

# Proyecto Final

Cabrera Barzalobre Jessica Jazmín

Ochoa Fernández Francine

Reyes Martínez Juan Carlos

## Bases de Datos

Diciembre 11, 2021

## 1. Introducción

### 1.1. Objetivo

Dar solución a la problemática propuesta aplicando los conocimientos adquiridos en la materia de Bases de datos.

### 1.2. Problemática y requerimientos

Una cadena de papelerías busca innovar la manera en que almacena su información, y los contratan para que desarrollen los sistemas informáticos para satisfacer los siguientes requerimientos:

Se desea tener almacenados datos como la razón social, domicilio, nombre y teléfonos de los proveedores, rfc, nombre, domicilio y al menos un email de los clientes. Es necesario tener un inventario de los productos que se venden, en el que debe guardarse el código de barras, precio al que fue comprado el producto, fecha de compra y cantidad de ejemplares en la bodega (stock). Se desea guardar la marca, descripción y precio

de los regalos, artículos de papelería, impresiones y recargas, siempre y cuando se tenga su correspondiente registro en el inventario. Debe también guardarse el número de venta, fecha de venta y la cantidad total a pagar de la venta, así como la cantidad de cada artículo y precio total a pagar por artículo. Adicional al almacenamiento de información, se requiere que el sistema resuelva lo siguiente:

- Al recibir el código de barras de un producto, regrese la utilidad.
- Cada que haya la venta de un artículo, deberá decrementarse el stock por la cantidad vendida de ese artículo. Si el valor llega a cero, abortar la transacción. Si el pedido se completa, pero quedan menos de 3 en stock, se deberá emitir una alerta.
- Dada una fecha, o una fecha de inicio y fecha de fin, regresar la cantidad total que se vendió y la ganancia correspondiente en esa fecha/periodo.
- Permitir obtener el nombre de aquellos productos de los cuales hay menos de 3 en stock.
- De manera automática se genere una vista que contenga información necesaria para asemejarse a una factura de una compra.
- Crear al menos, un índice, del tipo que se prefiera y donde se prefiera. Justificar el porqué de la elección en ambos aspectos.

Tomar en cuenta las siguientes consideraciones:

- Puede haber distintas soluciones al problema.
- Los requerimientos enlistados anteriormente, deberán ser realizados por medio de PgSQL, con los elementos que se consideren adecuados para resolverlos.
- El número de venta debe tener un formato similar a "VENT-001", prefijo VENT, seguido de un guión y un número secuencial.

- Donde esté presente el atributo domicilio, está compuesto por estado, código postal, colonia, calle y número.
- El diseño debe satisfacer todos los principios de diseño, los requerimientos anteriores y un buen manejo de información.

Una vez diseñada y lista la base de datos, se debe crear una interfaz gráfica vía app móvil o web, que permita:

1. Agregar la información de un cliente.
2. Ingresar una venta, de hasta 3 artículos, los cuales podrán seleccionarse de una lista de opciones, permitir ingresar la cantidad, calcular el costo total de cada artículo y el costo total de toda la venta. Ingresar dicha información en la base de datos, respetando todas las restricciones de integridad.

## 2. Plan de trabajo

### 2.1. Actividades a realizar

- **Lectura del proyecto final:** se leera a conciencia propia la documentación que se nos fue proporcionado para realizar este proyecto.
- **Modelo Entidad Relacional:** se propondrán diferentes MERs hasta lograr el definitivo que nos permita comprender y desarrollar de manera plena la base de datos correspondiente a la papelería.
- **Modelo Relacional:** al obtener el MER correspondiente al problema planteado se pasará a la transformación del diagrama para convertirlo en un MR que nos permita realizar las tablas correspondientes para la base de datos.
- **Creación de tablas e inserción de datos:** se comenzará la parte de la implementación a través de la creación de las tablas e inserción de datos.

- **Programación:** al finalizar las actividades anteriores se corresponderá a realizar la programación correspondiente para obtener los resultados deseados en los requerimientos del problema.

## 2.2. Organización

- **Lectura del proyecto final:** cada uno de los integrantes del equipo se comprometió a leer el proyecto final en sus tiempos libres para así llegar con una idea a la reunión que se desarrolló para discutir las partes teóricas del proyecto final.
- **Modelo Entidad Relacional:** nos reunimos a discutir y a plantear cada uno de los miembros del equipo un MER para así combinar las diferentes ideas en uno solo. Esto fue realizado por los tres miembros actuales del equipo.
- **Modelo Relacional:** al obtener el modelo gráfico para el problema planteado, los miembros nos dedicamos a transformar las entidades y relaciones del MER para obtener así el MR. La toma de decisión respecto al tipo de dato para las entidades y el diseño del MR fue realizado por todos los miembros durante la misma reunión donde se planteó el MER definitivo del equipo.
- **Creación de tablas e inserción de datos:** al finalizar la parte teórica del diseño final se procede a la creación de tablas e inserción de datos. Esta parte fue encargada de Carlos Reyes.
- **Programación:** teniendo las tablas y los datos insertados se procederá a la parte de programación de la cual estuvo a cargo Francine Ochoa.
- **Documentación:** finalmente, al obtener cada una de las partes del proyecto finalizadas, se procede a realizar el documento final en LATEX. Esta parte estuvo encargada de Jessica Cabrera.

## 2.3. Cronograma

ACTVIDADES	NOVIEMBRE				DICIEMBRE											
	SEMANA III			SEMANA I				SEMANA II								
	V 26	S 27	D 28	L 29	M 30	M 1	J 2	V 3	S 4	D 5	L 6	M 7	M 8	J 9	V 10	S 11
Lectura del proyecto																
Primera Reunión																
Modelo Entidad Relacional																
Modelo Relacional																
Segunda Reunión																
Creación de tablas e inserción de datos																
Programación																
Tercera Reunión																
Documentación																
Entrega de proyecto																

Figura 1: Cronograma

## 2.4. Herramientas empleadas

- **DISCORD:** es una plataforma de comunicación que esta destinada principalmente a los amantes de los videojuegos, sin embargo, también funciona muy bien para comunidades estudiantiles como la nuestra puesto que se puede realizar un grupo privado y organizar todo tipo de secciones dentro de ella.
- **WHATSAPP:** plataforma de FACEBOOK que nos permitio comunicarnos a través de mensajes durante todo este periodo de cuarentena y del proyecto.
- **POSTGRES SQL:** es el software principal de trabajo para el desarrollo de este proyecto. Un manejado de bases de datos de código abierto.
- **PYTHON:** lenguaje de programación que nos permitio realizar la conexión de nuestra base de datos y la terminal de nuestro proyecto.
- **DRAWIO.IO:** aplicación de diseños gráficos que hay en la web y que es parte de las herramientas permitidas por google drive.
- **OVERLEAF:** herramienta colaborativa para realizar en línea un documento basado en LATEX.

- **EXCEL:** software que está enfocado a la administración y a la realización de cálculos extensos e igualmente puede ser utilizado para bases de datos.
- **SLACK:** aplicación que se nos proporciona al inicio del semestre para obtener una retroalimentación por parte de nuestro profesor.

### 3. Diseño

#### 3.1. Modelo Entidad Relación

Para el modelo entidad relación (MER) se crearon las siguientes entidades con sus atributos y relaciones.

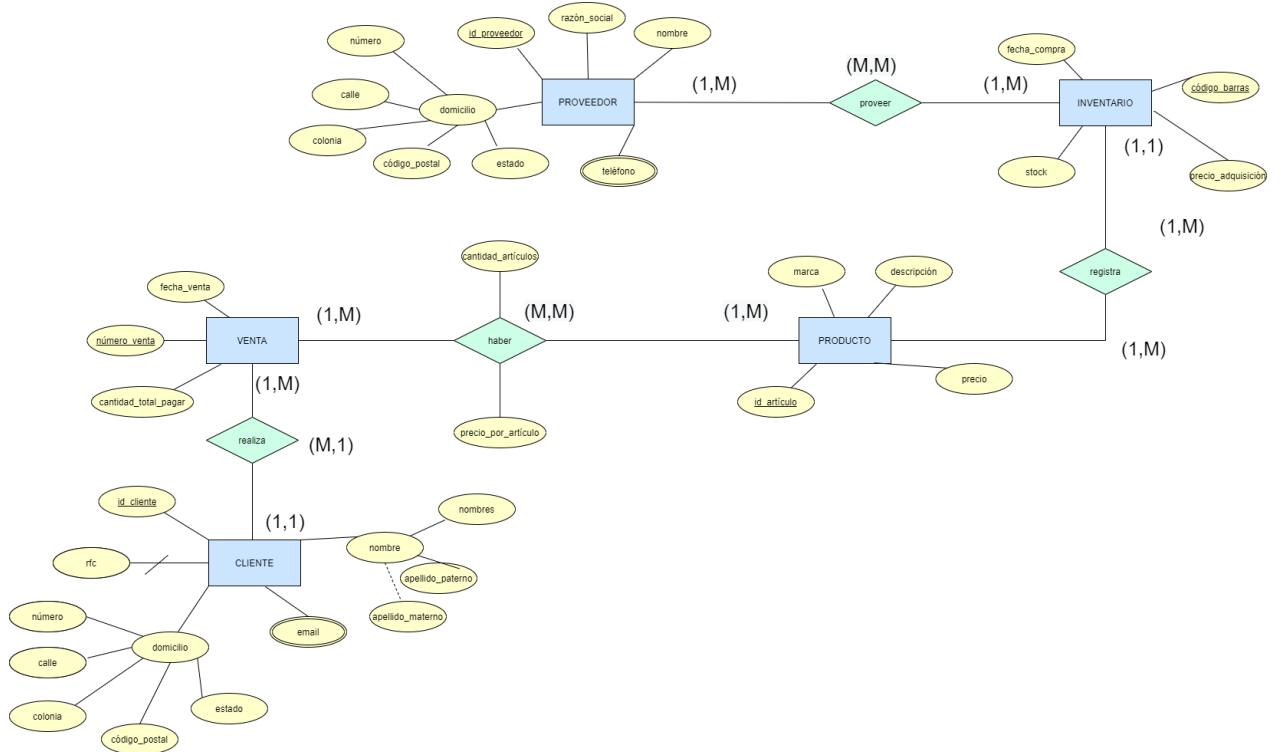


Figura 2: Modelo Entidad Relación

### 3.1.1. Entidad PROVEEDOR

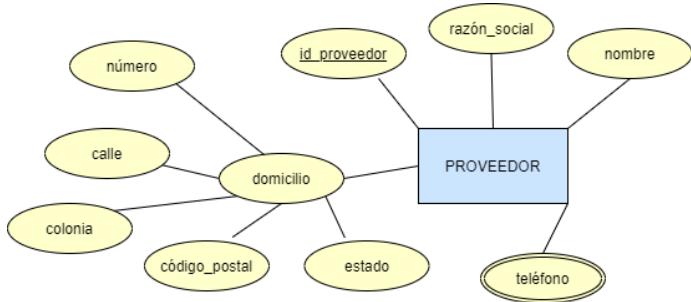


Figura 3: Entidad PROVEEDOR

El diseño de la entidad PROVEEDOR está realizado basado en las características planteadas en un inicio. Esta entidad contiene los siguientes atributos:

- **Id proveedor:** este atributo es la llave primaria que se utilizará para acceder a esta entidad en el modelo relacional y en la implementación. Esta llave fue creada para un manejo más simple en la fase de implementación.
- **Razón social:** un atributo simple que en su mayoría de veces podría llegar a considerarse como una posible llave primaria pues conceptualmente se considera como la denominación por la que se conoce de forma legal y oficial a una empresa. Sin embargo, la dejamos a un lado por cuestiones de diseño e implementación.
- **Nombre:** atributo simple que a su vez es compuesto dependiendo de su implementación. Este atributo suele constar del nombre, apellido paterno y apellido materno.
- **Domicilio:** atributo compuesto del código postal, estado, colonia, calle y número.

### 3.1.2. Relación proveer

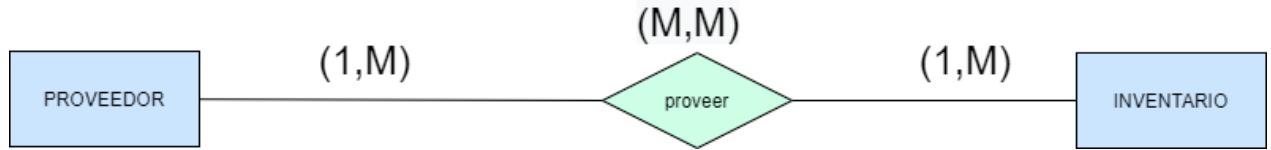


Figura 4: Relación proveer

La relación proveer existe entre las entidades PROVEEDOR e INVENTARIO. La cardinalidad que hay entre ellas es M:M, ya que un proveedor provee a muchos inventarios (1,M) y un inventario es provisto por muchos proveedores (1,M).

### 3.1.3. Entidad INVENTARIO

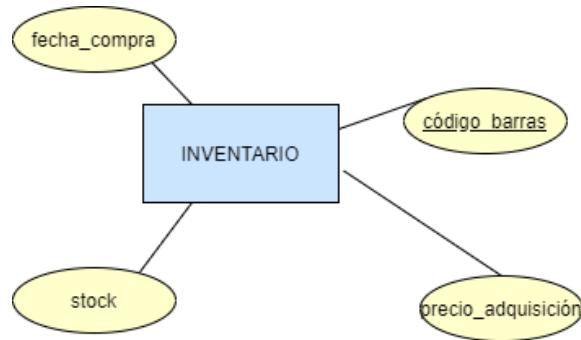


Figura 5: Entidad INVENTARIO

La entidad INVENTARIO está diseñada a partir de los siguientes cuatro atributos:

- **Código barras:** este atributo es la llave primaria que se utilizará para acceder a esta entidad en el modelo relacional y en la implementación. Esta llave es proporcionada de forma nativa en los requerimientos del problema.
- **Fecha compra:** un atributo simple que a su vez está compuesto por el día, mes y año.

- **Precio adquisición:** atributo simple que permite ver el precio por el que se adquirió algún artículo a través de los diversos proveedores existentes.
- **Stock:** atributo simple cuyo propósito es mostrar la cantidad de existencias en el inventario.

### 3.1.4. Relación registro

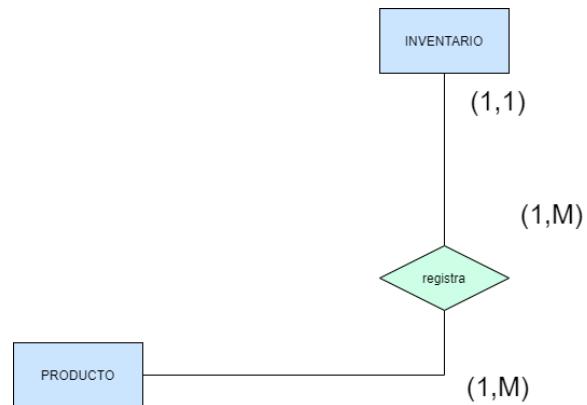


Figura 6: Relación registro

Es la relación existente entre las entidades INVENTARIO y PRODUCTO. La cardinalidad que hay entre ellas es 1:M, puesto que un inventario puede tener el registro de muchos productos (1,1) y un producto puede estar registrado en un solo inventario (1,M).

### 3.1.5. Entidad PRODUCTO

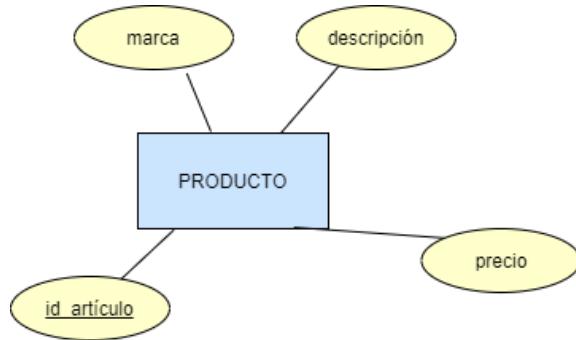


Figura 7: Entidad PRODUCTO

Para la entidad PRODUCTO se tomarón en cuenta los siguientes atributos:

- **Id artículo:** este atributo es la llave primaria que se utilizará para acceder a esta entidad en el modelo relacional y en la implementación. Esta llave fue creada para una obtener una mejor y más sencilla manipulación de datos en la fase de implementación.
- **Marca:** un atributo simple cuyo objetivo es mostrar la marca de cada artículo que yace en el inventario.
- **Descripción:** atributo simple cuya función será guardar una descripción precisa de los diversos artículos encontrados en el inventario.
- **Precio:** atributo simple el cual mostrará el precio al que se vende dicho artículo.

### 3.1.6. Relación haber

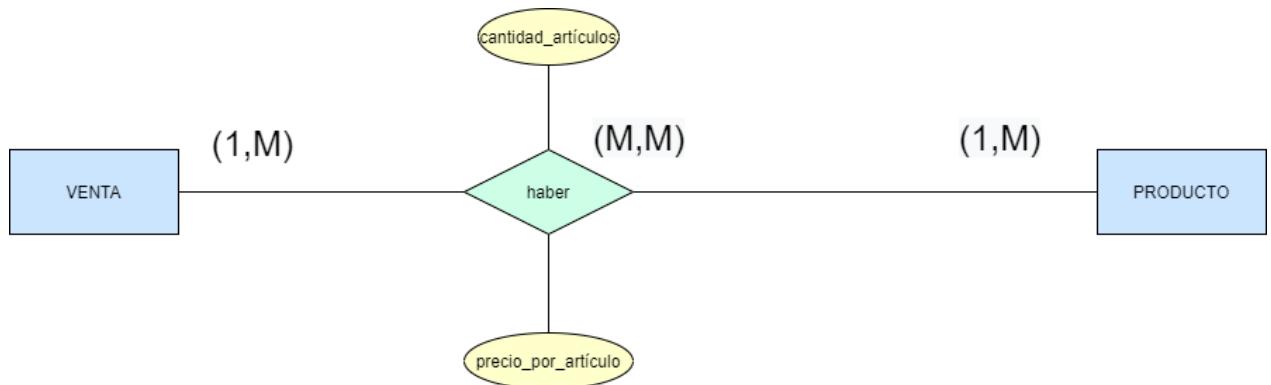


Figura 8: Relación haber

Esta relación existe entre las entidades PRODUCTO y VENTA. La cardinalidad que hay en esta relación es M:M, pues en una venta puede haber muchos productos (1,M) y un producto puede haberse contenido en muchas ventas (1,M). En esta relación existen dos atributos más los cuales son:

- **Cantidad artículo:** un atributo simple cuya finalidad es obtener la cantidad de artículos que se están registrando para la venta a realizar.
- **Precio por artículo:** un atributo simple que muestra el precio que se pagará por la cantidad de artículos en la venta.

### 3.1.7. Entidad VENTA

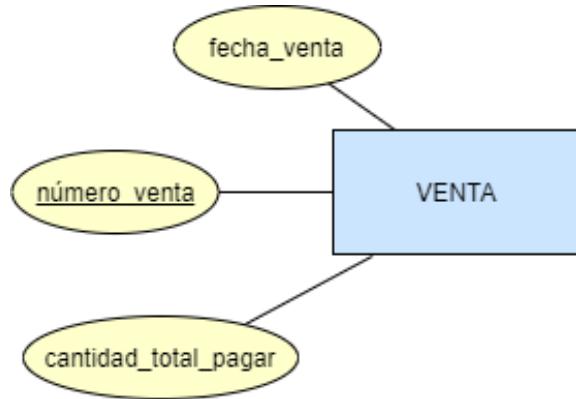


Figura 9: Entidad VENTA

La entidad VENTA fue expresada a partir de los siguientes atributos dados de forma nativa en los requerimientos:

- **Número venta:** este atributo es la llave primaria que se utilizará para acceder a esta entidad en el modelo relacional y en la implementación. Esta llave fue identificada de forma nativa en los requerimientos previamente planteados en el problema.
- **Fecha venta:** un atributo simple que a su vez puede estar compuesta por el día, mes y año.
- **Cantidad total pagar:** atributo simple que muestra la cantidad total por pagar para nuestro cliente.

### 3.1.8. Relación realiza

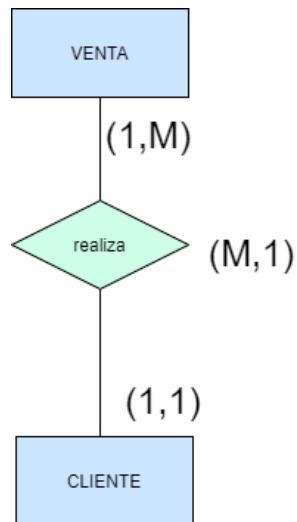


Figura 10: Relación realiza

La relación existente entre las entidades de VENTA y CLIENTE es realiza. Esta relación tiene una cardinalidad M:1, ya que una venta es realizada por un cliente (1:M) y pero un solo cliente puede realizar más de una venta (compra) (1,1).

### 3.1.9. Entidad CLIENTE

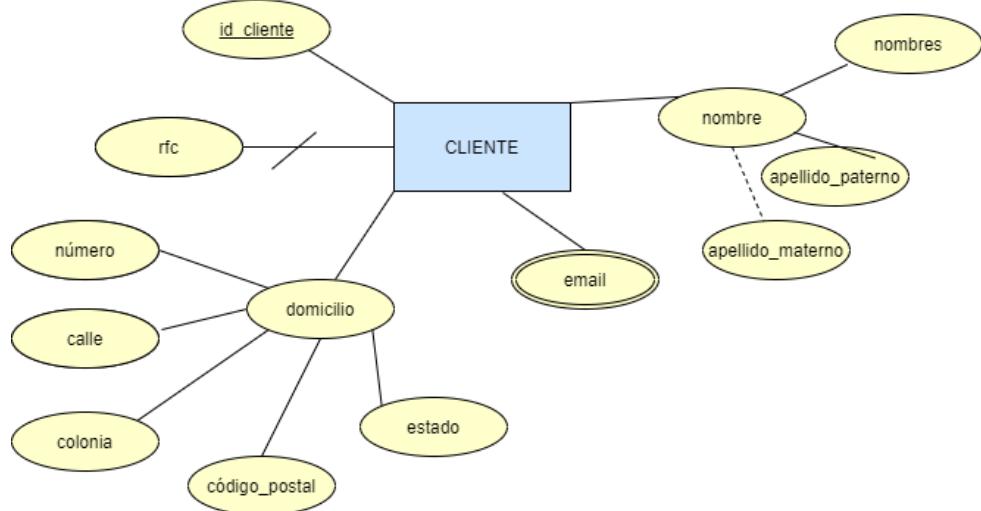


Figura 11: Entidad CLIENTE

En el diseño planteado para la entidad CLIENTE podemos encontrar los siguientes atributos:

- **Id cliente:** este atributo es la llave primaria que se utilizará para acceder a esta entidad en el modelo relacional y en la implementación. Esta llave fue creada para un manejo más simple en la fase de implementación.
- **RFC:** un atributo simple que en su momento podría considerarse como llave primaria. Sin embargo, la dejamos a un lado por cuestiones de diseño dejandola como una posible clave candidata. En su mayoría de veces tiene una extensión de quince caracteres.
- **Nombre:** atributo simple que a su vez es compuesto dependiendo de su implementación. Este atributo suele constar del nombre, apellido paterno y apellido materno.
- **Domicilio:** atributo compuesto del código postal, estado, colonia, calle y número.

- **Email:** atributo multivaluado que representa los emails por los cuales se puede llegar a contactar al cliente.

### 3.2. Modelo Relacional

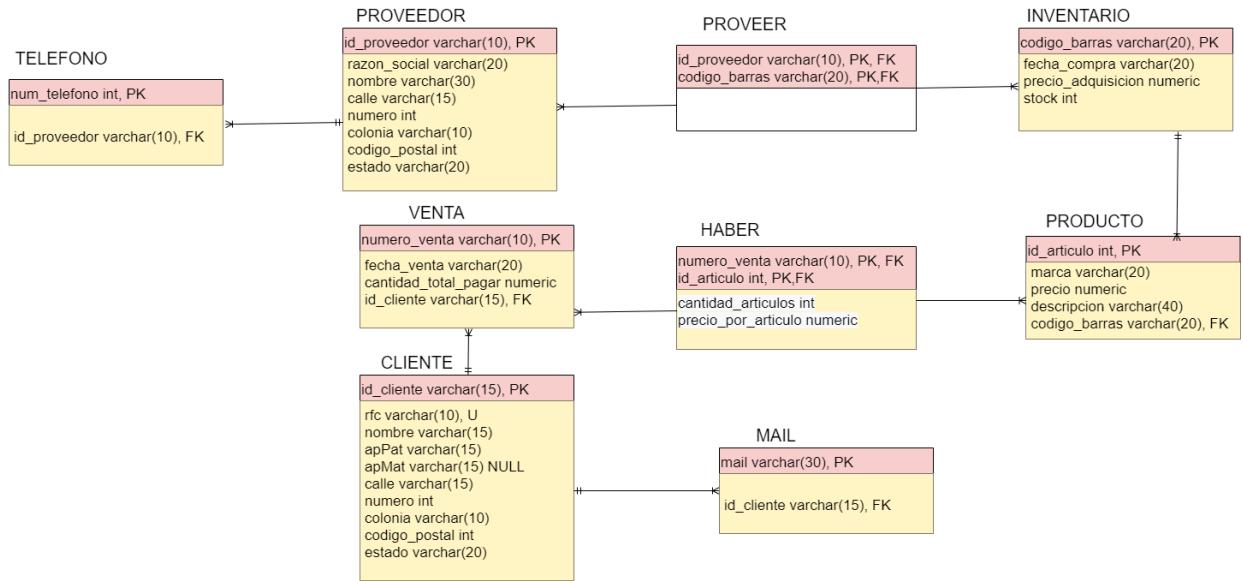


Figura 12: Modelo Relacional

El diagrama anterior es el Modelo Relacional (MR) que se obtiene al aplicar las correspondientes reglas de transformación a cada una de las entidades existentes en nuestro MER.

#### 3.2.1. Entidad TELEFONO

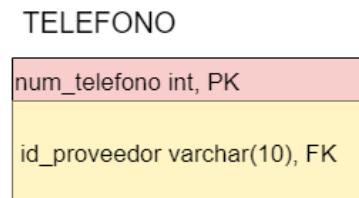


Figura 13: Entidad TELEFONO

Al realizar la transformación del MER al MR y al aplicar la correspondiente conversión en la entidad PROVEEDOR, se obtiene que el atributo teléfono al ser un atributo multivalorado su transformación consta de crear una nueva entidad con llave primaria así mismo y junto con ella la llave primaria de la entidad PROVEEDOR como llave foránea.

**TELEFONO:** {num\_telefono INT (PK) NOT NULL, id\_proveedor VARCHAR (10) (FK) NOT NULL}

### 3.2.2. Entidad PROVEEDOR

PROVEEDOR	
	<b>id_proveedor</b> varchar(10), PK
	razon_social varchar(20)
	nombre varchar(30)
	calle varchar(15)
	numero int
	colonia varchar(10)
	codigo_postal int
	estado varchar(20)

Figura 14: Entidad PROVEEDOR

La transformación correspondiente a la entidad PROVEEDOR quedara de la siguiente forma, tomando en cuenta la transformación del atributo multivalorado teléfono y la transformación de atributos compuestos la entidad quedaria de la siguiente forma:

**PROVEEDOR:** {id\_proveedor VARCHAR(10) (PK) NOT NULL, razon\_social VARCHAR (20) NOT NULL, nombre VARCHAR(30) NOT NULL, calle VARCHAR(15) NOT NULL, numero INT NOT NULL, colonia VARCHAR(10) NOT NULL, codigo\_postal INT NOT NULL, estado VARCHAR(20)}

### 3.2.3. Entidad PROVEER

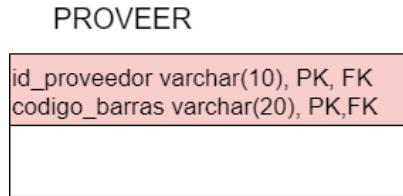


Figura 15: Entidad PROVEER

Al tener una cardinalidad M:M en lo que es la relación proveer entre las entidades PROVEEDOR e INVENTARIO tenemos que crear una nueva entidad cuya llave primaria es la combinación de las llaves primarias de las dos entidades que enlaza y al no tener ningún otro atributo queda de la siguiente manera:

**PROVEER:**{ [id\_proveedor VARCHAR(10) (FK) NOT NULL, codigo\_barra VARCHAR(20) (FK) NOT NULL] (PK)}

### 3.2.4. Entidad INVENTARIO

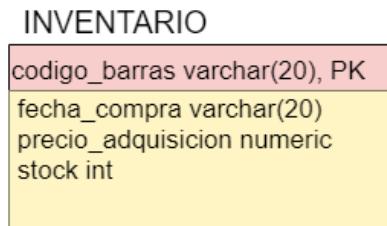


Figura 16: Entidad INVENTARIO

La transformación a realizar para la entidad INVENTARIO quedaría de la siguiente forma:

**INVENTARIO:**{codigo\_barra VARCHAR(PK) NOT NULL, fecha\_compra VARCHAR(20) NOT NULL, precio\_adquisicion NUMERIC NOT NULL, stock INT NOT NULL }

### 3.2.5. Entidad PRODUCTO

PRODUCTO	
id_articulo	int, PK
marca	varchar(20)
precio	numeric
descripcion	varchar(40)
codigo_barras	varchar(20), FK

Figura 17: Entidad PRODUCTO

Para la transformación de la entidad PRODUCTO se tiene que tomar en cuenta a su vez la transformación que se implementa en la relación 1:M de registro la cual pasa la llave primaria de la entidad INVENTARIO a la entidad PRODUCTO como llave foránea.

**PRODUCTO:**{id\_articulo INT (PK) NOT NULL, marca VARCHAR(20) NOT NULL, precio NUMERIC NOT NULL, descripcion VARCHAR(40) NOT NULL, codigo\_barras VARCHAR(20) NOT NULL (FK)}

### 3.2.6. Entidad HABER

HABER	
numero_venta	varchar(10), PK, FK
id_articulo	int, PK, FK
cantidad_articulos	int
precio_por_articulo	numeric

Figura 18: Entidad HABER

En la relación haber al tener una cardinalidad M:M formada por las entidades PRODUCTO y VENTA se aplica la misma regla de transformación que en la relación proveer. Se crea una nueva entidad donde la llave primaria es la combinación de las llaves primarias de las entidades PRODUCTO y VENTA junto con los atributos si

los hay, lo cual es este caso.

**HABER:**{[numero\_venta VARCHAR(10) (FK) NOT NULL, id\_articulo INT (FK) NOT NULL] (PK), cantidad\_articulo INT NOT NULL, precio\_por\_articulo NUMERIC NOT NULL}

### 3.2.7. Entidad VENTA

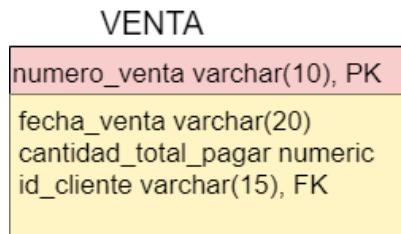


Figura 19: Entidad VENTA

La transformación desarrollada para la entidad VENTA es la siguiente, donde tenemos que considerar la transformación que se emplea en la relación realiza con cardinalidad M:1 entre las entidades de VENTA y CLIENTE:

**VENTA:**{numero\_venta VARCHAR(10) (PK) NOT NULL, fecha\_venta VARCHAR(20) NOT NULL, cantidad\_total\_pagar NUMERIC (C) NOT NULL, id\_cliente VARCHAR(15) (FK) NOT NULL}

### 3.2.8. Entidad CLIENTE

CLIENTE	
	id_cliente varchar(15), PK
rfc	varchar(10), U
nombre	varchar(15)
apPat	varchar(15)
apMat	varchar(15) NULL
calle	varchar(15)
numero	int
colonia	varchar(10)
codigo_postal	int
estado	varchar(20)

Figura 20: Entidad CLIENTE

Al transformar la entidad CLIENTE se toma en cuenta que el atributo mail es multivaluado, por lo cual se tiene que aplicar la transformación correspondiente para ello, igualmente para los atributos compuestos como lo son el nombre y la dirección. Esto queda:

**CLIENTE:**{id\_cliente VARCHAR(15) (PK) NOT NULL, rfc VARCHAR(10) (U) NOT NULL, nombre VARCHAR(15) NOT NULL, apPat VARCHAR(15) NOT NULL, apMat VARCHAR(15) NULL, calle VARCHAR(15) NOT NULL, numero INT NOT NULL, colonia VARCHAR(15) NOT NULL, codigo\_postal INT NOT NULL, estado VARCHAR(20) NOT NULL}

### 3.2.9. Entidad MAIL

MAIL	
	mail varchar(30), PK
	id_cliente varchar(15), FK

Figura 21: Entidad MAIL

Finalmente, al realizar la transformación de la tabla CLIENTE obtenemos una nueva entidad la cual nombramos como el mismo atributo, agregamos el atributo como llave primaria y la llave primaria de la entidad CLIENTE como llave foránea. Esta entidad queda de la siguiente forma:

**MAIL:**{mail VARCHAR(30) (PK), id\_cliente VARCHAR(15) (FK)}

## 4. Implementación

En esta sección se encuentra la parte que implica la implementación de nuestro análisis previo de la problemática a resolver.

### 4.1. Creación de Tablas

Para la crear las tablas que se utilizarán en la base de datos se utilizará como base principal la sintaxis para tablas en SQL.

```
CREATE TABLE tbl_name [(create_definition,...)] [table_options] [select_statement]
```

```
CREATE TABLE PROVEEDOR (
    id_proveedor varchar(10),
    razon_social varchar(30),
    nombre varchar(30),
    calle varchar(40),
    numero int,
    colonia varchar(40),
```

```
codigo_postal int,  
estado varchar(20),  
CONSTRAINT PROVEEDOR_PK PRIMARY KEY (id_proveedor)  
);
```

```
CREATE TABLE TELEFONO (   
telefono bigint,  
id_proveedor varchar(10),  
CONSTRAINT TELEFONO_PK PRIMARY KEY (telefono),  
CONSTRAINT PROVEEDOR_TELEFONO_FK FOREIGN KEY(id_proveedor)  
REFERENCES PROVEEDOR (id_proveedor)  
);
```

```
CREATE TABLE INVENTARIO (   
codigo_barras varchar(20),  
fecha_compra varchar(20),  
precio_adquisicion numeric,  
stock bigint,  
CONSTRAINT INVENTARIO_PK PRIMARY KEY (codigo_barras)  
);
```

```
CREATE TABLE PROVEER (   
id_proveedor varchar(10),  
codigo_barras varchar(20),  
CONSTRAINT PROVEER_PK PRIMARY KEY (id_proveedor,codigo_barras),  
CONSTRAINT PROVEER_PROVEEDOR_FK FOREIGN KEY(id_proveedor)  
REFERENCES PROVEEDOR (id_proveedor),  
CONSTRAINT PROVEER_INVENTARIO_FK FOREIGN KEY(codigo_barras)  
REFERENCES INVENTARIO (codigo_barras)  
);
```

```
CREATE TABLE CLIENTE (
    id_cliente varchar(15),
    rfc varchar(10), -UNIQUE
    nombre varchar(25),
    apPat varchar(25),
    apMat varchar(25) NULL,
    calle varchar(40),
    numero int
    colonia varchar(40),
    codigo_postal int,
    estado varchar(20),
    CONSTRAINT CLIENTE_PK PRIMARY KEY (id_cliente),
    CONSTRAINT CLIENTE_UK UNIQUE(rfc)
);
```

```
CREATE TABLE MAIL (
    mail varchar(40),
    id_cliente varchar(15),
    CONSTRAINT MAIL_PK PRIMARY KEY (mail),
    CONSTRAINT CLIENTE_MAIL_FK FOREIGN KEY(id_cliente)
    REFERENCES CLIENTE (id_cliente)
);
```

```
CREATE TABLE PRODUCTO (
    id_articulo int,
    marca varchar(20),
    precio numeric,
    descripcion varchar(50),
    codigo_barras varchar(20),
    CONSTRAINT PRODUCTO_PK PRIMARY KEY (id_articulo),
```

```
CONSTRAINT PRODUCTO_UK UNIQUE(descripcion),
CONSTRAINT PRODUCTO_INVENTARIO_FK FOREIGN KEY(codigo_barras)
REFERENCES INVENTARIO (codigo_barras)
);
```

```
CREATE TABLE VENTA (
numero_venta varchar(15),
fecha_venta varchar(20),
cantidad_total_pagar numeric,
id_cliente varchar(15),
CONSTRAINT VENTA_PK PRIMARY KEY (numero_venta),
CONSTRAINT VENTA_CLIENTE_FK FOREIGN KEY(id_cliente)
REFERENCES CLIENTE (id_cliente)
);
```

```
CREATE TABLE HABER (
id_articulo int,
numero_venta varchar(15),
cantidad_articulos int,
precio_por_articulo numeric,
CONSTRAINT HABER_PK PRIMARY KEY (id_articulo,numero_venta),
CONSTRAINT HABER_PRODUCTO_FK FOREIGN KEY(id_articulo)
REFERENCES PRODUCTO (id_articulo),
CONSTRAINT HABER_VENTA_FK FOREIGN KEY(numero_venta)
REFERENCES VENTA (numero_venta)
);
```

## 4.2. Inserción de datos en las tablas

Para la inserción de datos se utiliza la función INSERT INTO cuya sintaxis es la siguiente:

```
INSERT INTO table_name VALUES (value1, value2, value3, ...);
```

```
INSERT INTO PROVEEDOR VALUES ('ID86420', 'Vazquez y compañia', 'Proveedores Vazquez', 'Avenida Alvaro Obregon','567','Juarez','01588','CDMX');
```

```
INSERT INTO PROVEEDOR VALUES ('ID82020', 'S. en C.S.', 'Proveedores Historia', 'Eje 3 Norte','40','Polanco','45588','CDMX');
```

```
INSERT INTO PROVEEDOR VALUES ('ID74120', 'S.A.', 'Obrera Iii', 'Río de la Piedad','40','Compositores Mexicanos','45588','Monterrey');
```

```
INSERT INTO PROVEEDOR VALUES ('ID33020', 'S.A.', 'Parres El Guarda', 'Avenida Presidente Masaryk','987','El Prado','45588','Guadalajara');
```

```
INSERT INTO PROVEEDOR VALUES ('ID01020', 'S. en C. por A.', 'Tlatelolco Familia', 'Avenida Tláhuac','22','Angel Zimbron','45588','CDMX');
```

```
INSERT INTO PROVEEDOR VALUES ('ID56027', 'Carmen y compañia', 'Carmen Empresas', 'Calzada de los Misterios','678','Los Pastores','45588','Monterrey');
```

```
INSERT INTO PROVEEDOR VALUES ('ID35034', 'S. en C.S.', 'Proveedores Rinconada', 'Eje 2 sur','220','Barrio Norte','45588','Puebla');
```

```
INSERT INTO TELEFONO VALUES (5515846399, 'ID86420');
```

```
INSERT INTO TELEFONO VALUES (5548963281, 'ID82020');
```

```
INSERT INTO TELEFONO VALUES (5511462937, 'ID74120');
```

```
INSERT INTO TELEFONO VALUES (5551585958, 'ID33020');
```

```
INSERT INTO TELEFONO VALUES (5588692569, 'ID01020');  
INSERT INTO TELEFONO VALUES (5572025326, 'ID56027');  
INSERT INTO TELEFONO VALUES (5508142635, 'ID56027');  
INSERT INTO TELEFONO VALUES (5590603618, 'ID35034');  
INSERT INTO TELEFONO VALUES (5517283956, 'ID35034');  
  
INSERT INTO INVENTARIO VALUES('3456765','3-5-2019',345,32456);  
INSERT INTO INVENTARIO VALUES('2345467','1-5-2019',35,5764);  
INSERT INTO INVENTARIO VALUES('1239543','3-5-2019',345,32456);  
INSERT INTO INVENTARIO VALUES('9876542','1-5-2019',35,5764);  
INSERT INTO INVENTARIO VALUES('4354435','3-5-2019',345,45655);  
INSERT INTO INVENTARIO VALUES('2435674','6-2-2021',654,2400);  
INSERT INTO INVENTARIO VALUES('3546766','9-7-2018',765,246);  
INSERT INTO INVENTARIO VALUES('8765431','8-12-2020',987,35);  
  
INSERT INTO PROVEER VALUES('ID86420','3456765')  
INSERT INTO PROVEER VALUES('ID82020','2345467')  
INSERT INTO PROVEER VALUES('ID74120','1239543')  
INSERT INTO PROVEER VALUES('ID33020','9876542')  
INSERT INTO PROVEER VALUES('ID01020','4354435')  
INSERT INTO PROVEER VALUES('ID56027','2435674')  
INSERT INTO PROVEER VALUES('ID35034','3546766')
```

```
INSERT INTO PROVEER VALUES('ID35034','8765431')
```

```
INSERT INTO CLIENTE VALUES('IDC24354','ERTR6453','Ernesto','Fernandez','Castrejon','Avenida Alvaro Obregon','567','Juarez','01588','CDMX')
```

```
INSERT INTO CLIENTE VALUES('IDC32434','FGHJ5769','Hugo','Gomez','Flores','Eje 3 Norte','40','Polanco','45588','CDMX')
```

```
INSERT INTO CLIENTE VALUES('IDC76890','OIH35535','Martín','Lopez','Castrejon','Río de la Piedad','40','Compositores Mexicanos','45588','Monterrey')
```

```
INSERT INTO CLIENTE VALUES('IDC98763','KJHG3443','Pablo','Fernandez','María','Avenida Presidente Masaryk','987','El Prado','45588','Guadalajara')
```

```
INSERT INTO CLIENTE VALUES('IDCi9876','OIUYU900','Ximena','Reyes','Castrejon','Avenida Tláhuac','22','Angel Zimbron','45588','CDMX')
```

```
INSERT INTO CLIENTE VALUES('IDC89786','6578HJHJ','Jessica','Hernandez','Ribera','Calzada de los Misterios','678','Los Pastores','45588','Monterrey')
```

```
INSERT INTO CLIENTE VALUES('ID098768','KJHG3325','Ernesto','Rico','Valente','Eje 2 sur','220','Barrio Norte','45588','Puebla')
```

```
INSERT INTO MAIL VALUES('ernesto@gmial.com','IDC24354')
```

```
INSERT INTO MAIL VALUES('hugo@gmial.com','IDC32434')
```

```
INSERT INTO MAIL VALUES('martin@gmial.com','IDC76890')
```

```
INSERT INTO MAIL VALUES('pablo@gmial.com','IDC98763')
```

```
INSERT INTO MAIL VALUES('ximena@gmial.com','IDCi9876')
```

```
INSERT INTO MAIL VALUES('Jessica@gmial.com','IDC89786')
```

```
INSERT INTO MAIL VALUES('ernesto1@gmial.com','ID098768')
```

```
INSERT INTO MAIL VALUES('ximena234@gmial.com','IDCi9876')
INSERT INTO MAIL VALUES('Jessica333@gmial.com','IDC89786')
INSERT INTO MAIL VALUES('ernesto1789@gmial.com','ID098768')

INSERT INTO PRODUCTO VALUES(1,'BIC',4,'PLUMA','3456765')
INSERT INTO PRODUCTO VALUES(2,'LAPICITO',3,'LAPEZ','2345467')
INSERT INTO PRODUCTO VALUES(3,'FACTIS',5,'GOMA','1239543')
INSERT INTO PRODUCTO VALUES(4,'BARRILITO',9,'TIJERAS','9876542')
INSERT INTO PRODUCTO VALUES(5,'SCRIBE',20,'CUADERNO','4354435')
INSERT INTO PRODUCTO VALUES(6,'FABER-CASTELL',80,'COLORES','2435674')
INSERT INTO PRODUCTO VALUES(7,'MAPED',9,'SACAPUNTAS','3546766')
INSERT INTO PRODUCTO VALUES(8,'NORMA',30,'LIBRETA','8765431')

INSERT INTO VENTA VALUES ('VENT-001','8-8-2019',50,'IDC24354');
INSERT INTO VENTA VALUES ('VENT-002','18-12-2019',18,'ID098768');
INSERT INTO VENTA VALUES ('VENT-003','10-7-2018',12,'IDC89786');
INSERT INTO VENTA VALUES ('VENT-004','31-1-2020',24,'ID098768');
INSERT INTO VENTA VALUES ('VENT-005','8-12-2020',80,'IDC76890');
INSERT INTO VENTA VALUES ('VENT-006','5-1-2021',130,'IDCi9876');
INSERT INTO VENTA VALUES ('VENT-007','14-2-2020',138,'IDC98763');
INSERT INTO VENTA VALUES ('VENT-008','9-10-2019',4,'IDC24354');
```

```
INSERT INTO HABER VALUES(1,'VENT-001',100,6);
INSERT INTO HABER VALUES(5,'VENT-001',10,6);
INSERT INTO HABER VALUES(2,'VENT-002',50,5);
INSERT INTO HABER VALUES(3,'VENT-003',3,8);
INSERT INTO HABER VALUES(1,'VENT-004',5,7);
INSERT INTO HABER VALUES(3,'VENT-005',80,8);
INSERT INTO HABER VALUES(4,'VENT-006',90,12);
INSERT INTO HABER VALUES(5,'VENT-006',110,25);
INSERT INTO HABER VALUES(6,'VENT-007',130,100);
INSERT INTO HABER VALUES(7,'VENT-008',70,12);
INSERT INTO HABER VALUES(8,'VENT-008',40,35);
```

#### 4.3. Implementación de tablas e inserción de datos en PostgreSQL

Las tablas fueron creadas con la ayuda de un script para facilitar la inserción de datos.

```
Ahora esta conectado a la base de datos «papeleria» con el usuario «postgres».
CREATE TABLE
papeleria=# \l
                                         Listado de base de datos
  Nombre   | Dueo    | Codificacin | Collate           | Ctype      | Privilegios
-----+-----+-----+-----+-----+-----+
datos_examen | postgres | UTF8 | Spanish_Mexico.1252 | Spanish_Mexico.1252 |
ejemplotarea3 | postgres | UTF8 | Spanish_Mexico.1252 | Spanish_Mexico.1252 |
papeleria    | postgres | UTF8 | Spanish_Mexico.1252 | Spanish_Mexico.1252 |
postgres     | postgres | UTF8 | Spanish_Mexico.1252 | Spanish_Mexico.1252 |
template0    | postgres | UTF8 | Spanish_Mexico.1252 | Spanish_Mexico.1252 | =c/postgres
template1    | postgres | UTF8 | Spanish_Mexico.1252 | Spanish_Mexico.1252 | =c/postgres
                                         |           |           |           |           | postgres=CTc/postgres
                                         |           |           |           |           | postgres=CTc/postgres
                                         |           |           |           |           | postgres=CTc/postgres
(6 filas)
```

Figura 22: Creacion de tablas y verificacion de base de datos: papeleria

La insercion de igual manera se realizo mediante un script.

Figura 23: Insercion de datos en tablas

#### 4.4. Verificación de datos en tablas

```
papeleria=# SELECT *FROM TELEFONO;
telefono | id_proveedor
-----+-----
5515846399 | ID86420
5548963281 | ID82020
5511462937 | ID74120
5551585958 | ID33020
5588692569 | ID01020
5572025326 | ID56027
5508142635 | ID56027
5590603618 | ID35034
5517283956 | ID35034
(9 filas)
```

Figura 24: Tabla TELEFONO

Primeramente verificamos los datos que se insertaron en la tabla TELEFONO, se puede observar que se agrego de forma correcta tanto el número de teléfono (Llave primaria) y el id del proveedor en cuestión (Llave foránea).

```
papeleria=# SELECT *FROM HABER;
id_articulo | numero_venta | cantidad_articulos | precio_por_articulo
-----+-----+-----+-----
1 | VENT-001 | 100 | 6
5 | VENT-001 | 10 | 6
2 | VENT-002 | 50 | 5
3 | VENT-003 | 3 | 8
1 | VENT-004 | 5 | 7
3 | VENT-005 | 80 | 8
4 | VENT-006 | 90 | 12
5 | VENT-006 | 110 | 25
6 | VENT-007 | 130 | 100
7 | VENT-008 | 70 | 12
8 | VENT-008 | 40 | 35
(11 filas)
```

Figura 25: Tabla HABER

En la tabla HABER se observa que se agregaron de forma correcta los datos tanto para el artículo como para el de la venta. Se logra ver los datos propuestos para el id del artículo, número de venta, cantidad de artículos y el precio por artículo.

```
papeleria=# SELECT *FROM PROVEER;
id_proveedor | codigo_barras
-----+-----
ID86420      | 3456765
ID82020      | 2345467
ID74120      | 1239543
ID33020      | 9876542
ID01020      | 4354435
ID56027      | 2435674
ID35034      | 3546766
ID35034      | 8765431
(8 filas)
```

Figura 26: Tabla PROVEER

La tabla PROVEER también se logra observar la correcta inserción de los datos propuestos para realizar las pruebas correspondientes (id proveedor, código de barras).

```
papeleria=# SELECT *FROM VENTA;
numero_venta | fecha_venta | cantidad_total_pagar | id_cliente
-----+-----+-----+-----
VENT-001     | 8-8-2019   | 50 | IDC24354
VENT-002     | 18-12-2019 | 18 | ID098768
VENT-003     | 10-7-2018   | 12 | IDC89786
VENT-004     | 31-1-2020   | 24 | ID098768
VENT-005     | 8-12-2020   | 80 | IDC76890
VENT-006     | 5-1-2021    | 130 | IDC9876
VENT-007     | 14-2-2020   | 138 | IDC98763
VENT-008     | 9-10-2019   | 4  | IDC24354
(8 filas)
```

Figura 27: Tabla VENTA

Al realizar la verificación de los datos insertados en la tabla VENTA logramos ver que cada una de las columnas implementadas en esta entidad se realizo de manera correcta.

```
papeleria=# SELECT *FROM PRODUCTO;
+-----+-----+-----+-----+-----+
| id_articulo | marca | precio | descripcion | codigo_barras |
+-----+-----+-----+-----+-----+
| 1 | BIC | 10 | PLUMA | 3456765 |
| 2 | LAPICITO | 14 | LAPIZ | 2345467 |
| 3 | FACTIS | 50 | GOMA | 1239543 |
| 4 | BARRILITO | 90 | TIJERAS | 9876542 |
| 5 | SCRIBE | 20 | CUADERNO | 4354435 |
| 6 | FABER-CASTELL | 80 | COLORES | 2435674 |
| 7 | MAPED | 90 | SACAPUNTAS | 3546766 |
| 8 | NORMA | 30 | LIBRETA | 8765431 |
| 9 | NORMA | 34 | PLUMON | 4567838 |
+-----+-----+-----+-----+-----+
(9 filas)
```

Figura 28: Tabla PRODUCTO

Para la entidad PRODUCTO logramos verificar que la inserción de los datos fue exitosa para las cinco columnas por las cuales está formada la entidad.

```
papeleria=# SELECT *FROM INVENTARIO;
+-----+-----+-----+-----+
| codigo_barras | fecha_compra | precio_adquisicion | stock |
+-----+-----+-----+-----+
| 3456765 | 3-5-2019 | 3 | 324 |
| 2345467 | 1-5-2019 | 6 | 54 |
| 1239543 | 3-5-2019 | 4 | 326 |
| 9876542 | 1-5-2019 | 8 | 574 |
| 4354435 | 3-5-2019 | 5 | 455 |
| 2435674 | 6-2-2021 | 10 | 2400 |
| 3546766 | 9-7-2018 | 5 | 246 |
| 8765431 | 8-12-2020 | 9 | 35 |
| 4567838 | 8-12-2020 | 5 | 2 |
+-----+-----+-----+-----+
(9 filas)
```

Figura 29: Tabla INVENTARIO

Al introducir los datos para la entidad INVENTARIO también logramos visualizar que esto se logra de forma correcta. Las columnas código de barras, fecha de compra, precio de adquisición y stock tienen en sus filas los datos correspondiente para los artículos que yacen en el inventario.

mail	id_cliente
ernesto@gmial.com	IDC24354
hugo@gmial.com	IDC32434
martin@gmial.com	IDC76890
pablo@gmial.com	IDC98763
ximena@gmial.com	IDCi9876
Jessica@gmial.com	IDC89786
ernesto1@gmial.com	ID098768
ximena234@gmial.com	IDCi9876
Jessica333@gmial.com	IDC89786
ernesto1789@gmial.com	ID098768
(10 filas)	

Figura 30: Tabla MAIL

La entidad MAIL como fue planteada recibe los datos correspondientes al mail como llave primaria e id cliente como llave foránea. Se nota con facilidad que la inserción de llevo de forma correcta a pesar de los caracteres especiales como el @.

id_cliente	rfc	nombre	appat	apmat	calle	numero	colonia
codigo_postal	estado						
IDC24354	ERTR6453	Ernesto	Fernandez	Castrejon	Avenida Alvaro Obregon	567	Juarez
	1588	CDMX					
IDC32434	FGHJ5769	Hugo	Gomez	Flores	Eje 3 Norte	40	Polanco
	45588	CDMX					
IDC76890	OIH35535	Martijn	Lopez	Castrejon	Rio de la Piedad	40	Compositores Mexicanos
	45588	Monterrey					
IDC98763	KJHG3443	Pablo	Fernandez	Marja	Avenida Presidente Masaryk	987	El Prado
	45588	Guadalajara					
IDCi9876	OIUYU990	Ximena	Reyes	Castrejon	Avenida Tlalhuac	22	Angel Zimbron
	45588	CDMX					
IDC89786	6578H0H	Jessica	Hernandez	Ribera	Calzada de los Misterios	678	Los Pastores
	45588	Monterrey					
ID098768	KJHG3325	Ernesto	Rico	Valente	Eje 2 sur	220	Barrio Norte
	45588	Puebla					
(7 filas)							

Figura 31: Tabla CLIENTE

Volvemos a verificar para la entidad CLIENTE la correcta introducción de los datos propuestos para las pruebas a realizar en la base de datos. Las diez columnas por las cuales está formada la entidad CLIENTE tienen de forma correcta cada uno de los datos insertados previamente.

id_proveedor	razon_social	nombre	calle	numero	colonia
ID86420	Vazquez y compa ;ja	Provedores Vazquez	Avenida Alvaro Obregon	567	Juarez
	1588   CDMX				
ID82020	S. en C.S.	Provedores Historia	Eje 3 Norte	40	Polanco
	45588   CDMX				
ID74120	S.A.	Obrera Iii	R jo de la Piedad	40	Compositores Mexicanos
OS	45588   Monterrey				
ID33020	S.A.	Parres El Guarda	Avenida Presidente Masaryk	987	El Prado
	45588   Guadalajara				
ID01020	S. en C. por A.	Tlatelolco Familia	Avenida Tl ihuac	22	Angel Zimbron
	45588   CDMX				
ID56027	Carmen y compa ;ja	Carmen Empresas	Calzada de los Misterios	678	Los Pastores
	45588   Monterrey				
ID35034	S. en C.S.	Provedores Rinconada	Eje 2 sur	220	Barrio Norte
	45588   Puebla				
(7 filas)					

Figura 32: Tabla PROVEEDOR

Finalmente, igual que las anteriores entidades o tablas podemos verificiar que sus atributos contienen la informacin considerada y adecuada para cada uno de sus proveedores.

## 4.5. Funciones implementadas

- Al recibir el cdigo de barras de un producto, regresar la utilidad

```
papeleria=# --____1. Al recibir el cdigo de barras de un producto, regrese la utilidad._____
papeleria=#
papeleria=#
papeleria=# CREATE FUNCTION fcodigo_barras_utilidad (varcodigo_barras varchar(20) )
papeleria=# RETURNS int
papeleria=# LANGUAGE plpgsql
papeleria=# AS
papeleria=# $$
papeleria$$# DECLARE
papeleria$$#     utilidad integer;
papeleria$$# BEGIN
papeleria$$#     SELECT (precio-precio_adquisicion) into utilidad
papeleria$$#     FROM INVENTARIO, PRODUCTO WHERE INVENTARIO.codigo_barras = PRODUCTO.codigo_barras AND INVENTARIO.codigo_barras=varcodigo_barras;
papeleria$$#     RAISE NOTICE 'El cdigo de barras % tiene una utilidad de %',varcodigo_barras,utilidad;
papeleria$$#     return utilidad;
papeleria$$# END;
papeleria$$# $$;
CREATE FUNCTION
papeleria=#
papeleria=# SELECT fcodigo_barras_utilidad('3456765');
NOTICE: El cdigo de barras 3456765 tiene una utilidad de 7
fcodigo_barras_utilidad
-----
```

Figura 33: Utilidad mediante cdigo de barras

- Cada que haya la venta de un artículo, se decrementará el stock por la cantidad vendida de ese artículo. Si el valor llega a cero, se abortará la transacción. Si el pedido se completa pero quedan menos de 3 en stock, se emitirá una alerta.

```

papeleria=# 
papeleria=# CREATE OR REPLACE FUNCTION stock_articulos()
papeleria=# RETURNS TRIGGER AS
papeleria=# $$
papeleria#$# DECLARE
papeleria#$#     var_codigo_barras varchar;
papeleria#$#     varstock integer;
papeleria#$# BEGIN
papeleria#$#
papeleria#$#     SELECT codigo_barras INTO var_codigo_barras
papeleria#$#     FROM PRODUCTO WHERE id_articulo=NEW.id_articulo;
papeleria#$#     RAISE NOTICE 'Codigo de Barras: %',var_codigo_barras;
papeleria#$#
papeleria#$#     SELECT stock INTO varstock
papeleria#$#     FROM INVENTARIO WHERE codigo_barras=var_codigo_barras;
papeleria#$#     RAISE NOTICE 'Stock Inicial: %',varstock;
papeleria#$#
papeleria#$#     IF (varstock-NEW.cantidad_articulos)=0 THEN
papeleria#$#         RAISE NOTICE 'NO SE PUEDE SEGUIR CON VENTA: El Stock del producto que quiere comprar es %',varstock-NEW.cantidad_articulos;
papeleria#$#         RETURN NULL;
papeleria#$#     ELSEIF ((varstock-NEW.cantidad_articulos)<3) THEN
papeleria#$#         RAISE NOTICE 'ALERTA: El Stock restante del producto comprado es de %. Es muy bajo.',varstock-NEW.cantidad_articulos;
papeleria#$#         UPDATE INVENTARIO SET stock=varstock-NEW.cantidad_articulos WHERE codigo_barras=var_codigo_barras;
papeleria#$#         RETURN NULL;
papeleria#$#     ELSE
papeleria#$#         UPDATE INVENTARIO SET stock=varstock-NEW.cantidad_articulos WHERE codigo_barras=var_codigo_barras;
papeleria#$#         RAISE NOTICE 'Stock Final: %',varstock-NEW.cantidad_articulos;
papeleria#$#         RETURN NULL;
papeleria#$#     END IF;
papeleria#$#
papeleria#$# END;
papeleria#$$$
papeleria=# LANGUAGE PLPGSQL;
CREATE FUNCTION
papeleria=# 
papeleria=# 
papeleria=# CREATE TRIGGER trigger_stock_articulos
papeleria=# BEFORE INSERT ON HABER
papeleria=# FOR EACH ROW
papeleria=# EXECUTE FUNCTION stock_articulos();
CREATE TRIGGER
papeleria=# -- Caso cuando no se puede hacer la venta _____
papeleria=# --Primero se deve crear el numero de venta
papeleria=# INSERT INTO VENTA VALUES ('VENT-009','9-10-2019',300,'IDC89786');
INSERT 0 1
papeleria=# --Luego se puede ingresar la venta del articulo
papeleria=# INSERT INTO HABER VALUES(2,'VENT-009',54,35);
NOTICE: Codigo de Barras: 2345467
NOTICE: Stock Inicial: 54
NOTICE: NO SE PUEDE SEGUIR CON VENTA: El Stock del producto que quiere comprar es 0
INSERT 0 0
papeleria=# -- Caso si se hace venta pero manda alerta de poco stock_____
papeleria=# --Primero se deve crear el numero de venta
papeleria=# INSERT INTO VENTA VALUES ('VENT-010','9-10-2019',300,'IDC89786');
INSERT 0 1
papeleria=# --Luego se puede ingresar la venta del articulo
papeleria=# INSERT INTO HABER VALUES(9,'VENT-010',1,35);
NOTICE: Codigo de Barras: 4567838
NOTICE: Stock Inicial: 2
NOTICE: ALERTA: El Stock restante del producto comprado es de 1. Es muy bajo.
INSERT 0 0
papeleria=# -- Caso de venta normal_____
papeleria=# --Primero se deve crear el numero de venta
papeleria=# INSERT INTO VENTA VALUES ('VENT-011','9-10-2019',400,'IDC24354');
INSERT 0 1
papeleria=# --Luego se puede ingresar la venta del articulo
papeleria=# INSERT INTO HABER VALUES(1,'VENT-011',24,35);
NOTICE: Codigo de Barras: 3456765
NOTICE: Stock Inicial: 324
NOTICE: Stock Final: 300
INSERT 0 0
papeleria=#

```

Figura 34: Decremento de stock

- Dada una fecha, o una fecha de inicio y fecha de fin, se regresa la cantidad total que se vendió y la ganacia correspondiente en esa fecha/periodo.

Por cuestiones de tiempo, no se pudo terminar esta función. Se intentará terminar para el dia de la presentación.

```
papeleria=# -- ____3. Dada una fecha, o una fecha de inicio y fecha de fin, regresar la cantidad total que se vendió_____
papeleria=#   --____y la ganacia correspondiente en esa fecha/periodo._____
papeleria=
papeleria=# --NO SE LOGRÓ :( 
papeleria=
papeleria=# UPDATE VENTA SET fecha_venta=TO_DATE(fecha_venta,'DD-MM-YYYY');
UPDATE 11
papeleria=
papeleria# select * from venta where fecha_venta between '2019-08-08' and '2020-12-08'
papeleria#
papeleria#
papeleria# CREATE FUNCTION fecha_intervalo_venta (fecha1 date, fecha2 date)
papeleria# RETURNS int
papeleria# LANGUAGE plpgsql
papeleria# AS
papeleria# $$
papeleria$# DECLARE
papeleria$#   utilidad integer;
papeleria$# BEGIN
papeleria$#   --SELECT * from VENTA where fecha_venta between fecha1 and fecha2;
papeleria$#   SELECT (precio-precio_adquisicion) into utilidad
papeleria$#   FROM INVENTARIO, PRODUCTO WHERE INVENTARIO.codigo_barras = PRODUCTO.codigo_barras AND INVENTARIO.codigo_barras=varcodigo_barras;
papeleria$#   RAISE NOTICE 'El codigo de barras % tiene una utilidad de %',varcodigo_barras,utilidad;
papeleria$#   return utilidad;
papeleria$# END;
papeleria$# $$;
papeleria$# $$;
ERROR: error de sintaxis en o cerca de «»
LINEA 1: ...nta where fecha_venta between '2019-08-08' and '2020-12-08' ^
papeleria=
```

Figura 35: Regresa la cantidad total que se vendió y la ganacia correspondiente

- Permitir obtener el nombre de aquellos productos de los cuales hay menos de 3 en stock.

```
papeleria# -- ____4. Permitir obtener el nombre de aquellos productos de los cuales hay menos de 3 en stock._____
papeleria# CREATE VIEW articulos_stock AS
papeleria# SELECT descripcion as articulo ,stock FROM PRODUCTO, INVENTARIO WHERE PRODUCTO.codigo_barras=INVENTARIO.codigo_barras AND stock<3;
CREATE VIEW
papeleria#
papeleria# --DROP VIEW articulos_stock; --En caso necesario;
papeleria#
papeleria# SELECT * FROM articulos_stock;
  articulo | stock
-----+-----
 PLUMON  |    2
(1 fila)
```

Figura 36: Nombre de productos con stock menor a 3

- Generar de forma automática una vista que contiene la información necesaria para asemejarse a una factura de una compra.

```
papeleria=# --__5. De manera automática se genere una vista que contenga información necesaria_____
papeleria=#   --__ para asemejarse a una factura de una compra._____
papeleria=# CREATE VIEW factura AS
papeleria=# SELECT venta.numero_venta, id_cliente, fecha_venta, cantidad_total_pagar,id_articulo,cantidad_articulos,
precio_por_articulo
papeleria=# from venta, haber WHERE venta.numero_venta=haber.numero_venta;
CREATE VIEW
papeleria=#

```

Figura 37: Vista que genera el formato similar de una factura

- Crear al menos, un índice, del tipo que se prefiera y donde se prefiera. Justificar el porqué de la elección en ambos aspectos.

```
papeleria=# --__6.Crear al menos, un índice, del tipo que se prefiera y donde se prefiera. _____
papeleria=#   --__Justificar el porqué de la elección en ambos aspectos. _____
papeleria=# CREATE INDEX codigo_barras ON INVENTARIO("codigo_barras");
CREATE INDEX
papeleria=#
papeleria=# SELECT * from INVENTARIO WHERE codigo_barras IN ('1239543','2435674','8765431');
 codigo_barras | fecha_compra | precio_adquisicion | stock
-----+-----+-----+-----+
 1239543 | 3-5-2019 |          4 |    326
 2435674 | 6-2-2021 |         10 |   2400
 8765431 | 8-12-2020 |          9 |    35
(3 filas)
```

Figura 38: Creación de un índice

PostgreSQL proporciona varios tipos de índices: B-tree, Hash, GiST, SP-GiST y GIN. Cada tipo de índice utiliza un algoritmo diferente que se adapta mejor a diferentes tipos de consultas. Por defecto, el comando CREATE INDEX crea índices de árbol B, que se ajustan a las tareas más comunes. Los árboles B pueden manejar consultas de equivalencias y de rango sobre datos que pueden ser clasificados en algún orden. En particular, el manejador PostgreSQL considerará el uso de un índice de árbol B siempre que una columna indexada esté implicada en una comparación que utilice uno de estos operadores: <, <=, =, >=, >. Las instrucciones equivalentes a las combinaciones de estos operadores, como BETWEEN e IN, también pueden implementarse con una búsqueda de índice de árbol B. Además, una condición IS NULL o IS NOT NULL en una columna de índice puede utilizarse con un índice de árbol B.

## **5. Presentación**

### **5.1. PostgreSQL**

PostgreSQL es un sistema para gestionar bases de datos de muy alto nivel, completamente de software libre y con una licencia BSD, compatible con cualquier uso, ya sea personal o comercial. Este sistema tiene como precursor otro sistema gestor de bases de datos, llamado INGRES, que fue uno de los primeros intentos de implementar un sistema de bases de datos relacional. INGRES abrió el camino para muchos otros sistemas conocidos como Sybase, Informix o el propio SQL Server. El impulsor de INGRES lideró posteriormente también el desarrollo de PostgreSQL, cuyo nombre del proyecto hace referencia a su propia raíz (Post-Ingres).

PostgreSQL es un sistema de base de datos relacional de alta disponibilidad. Es capaz de funcionar de manera estable en el servidor y, por lo tanto, resulta robusto, una de las principales características que buscan las empresas. Además, es consistente y tolerante a fallos. Es compatible con el modelo relacional, ya que asegura siempre su integridad referencial.

### **5.2. Python**

Python es un lenguaje de programación interpretado cuya filosofía hace hincapié en la legibilidad de su código. Se trata de un lenguaje de programación multiparadigma, ya que soporta parcialmente la orientación a objetos, programación imperativa y, en menor medida, programación funcional. Es un lenguaje interpretado, dinámico y multiplataforma.

Python es un lenguaje de programación multiparadigma. Esto significa que más que forzar a los programadores a adoptar un estilo particular de programación, permite varios estilos: programación orientada a objetos, programación imperativa y programación funcional. Otros paradigmas están soportados mediante el uso de ex-

tensiones.

Python usa tipado dinámico y conteo de referencias para la administración de memoria.

Una característica importante de Python es la resolución dinámica de nombres; es decir, lo que enlaza un método y un nombre de variable durante la ejecución del programa (también llamado enlace dinámico de métodos).

Otro objetivo del diseño del lenguaje es la facilidad de extensión. Se pueden escribir nuevos módulos fácilmente en C o C++. Python puede incluirse en aplicaciones que necesitan una interfaz programable.

### 5.3. Bibliotecas de conexión

#### PySimpleGUI

PySimpleGUI es una biblioteca de Python que permite crear GUI's (Interfaz Gráfica de Usuario). Se especifica la ventana GUI usando un "diseño" que contiene widgets (se llaman <sup>E</sup>lementos.<sub>en</sub> PySimpleGUI). El diseño se utiliza para crear una ventana utilizando uno de los 4 marcos compatibles para mostrar e interactuar con su ventana. Los marcos compatibles incluyen tkinter, Qt, WxPython o Remi.

#### Psycopg2

Psycopg es un adaptador de base de datos PostgreSQL para el lenguaje de programación Python. Sus principales características son la implementación completa de la especificación Python DB API 2.0 y la seguridad de subprocesos (varios subprocesos pueden compartir la misma conexión). Fue diseñado para aplicaciones con muchos subprocesos múltiples que crean y destruyen muchos cursores y hacen una gran cantidad de INSERT o UPDATE simultáneos.

Varias versiones de Python son compatibles y se adaptan a los tipos de datos de

PostgreSQL coincidentes; La adaptación se puede ampliar y personalizar gracias a un sistema de adaptación de objetos flexible.

Psycopg 2 es compatible con Unicode y Python 3.

## 5.4. Interfaz Gráfica de Usuario

Para poder correr la interfaz es necesario haber creado con anterioridad la base de datos 'papeleria' junto con las tablas y su inserción de datos. Todo esto se encuentra en los scripts *1\_creacion\_tablas.sql* y *2\_insertando\_datos.sql*.

### 5.4.1. Pantalla de Inicio

La pantalla de inicio de la interfaz muestra una explicación breve y general sobre lo que se puede hacer en la interfaz.



Figura 39: Pantalla de inicio con instrucciones

#### 5.4.2. Pantalla de cliente

Pantalla de llenado para ingresar los datos de un cliente, en este formulario el usuario ingresa los datos del cliente, cabe resaltar que en caso de querer guardar los datos el usuario debe presionar el botón inferior de agregar datos a la base y en caso de querer borrar los datos debe presionar el botón cancel. Si se quiere ver datos de clientes previamente agregados debe presionar el botón de ver clientes agregados.

The screenshot shows a window titled "PAPELERIA" with a light yellow header. The main title "Agrega la información de tu cliente:" is centered in a dark green box. Below it is a list of fields with corresponding input fields:

id_cliente	[Input Field]
rfc	[Input Field]
nombre	[Input Field]
Apellido Paterno	[Input Field]
Apellido Materno	[Input Field]
Calle	[Input Field]
Numero	[Input Field]
Colonia	[Input Field]
CP	[Input Field]
Estado	[Input Field]

At the bottom of the form are three buttons: "Agregar datos a la base" (green), "Ver clientes agregados" (pink), and "Cancel" (red). Below the buttons is a navigation bar with "home" and page numbers "1" and "2".

Figura 40: Pantalla para ingresar datos de un cliente

#### 5.4.3. Pantalla de vista de clientes agregados

Pantalla de clientes agregados al oprimir botón de ver clientes agregados (sin haber agregado datos aún). En este caso al no tener datos de cliente alguno, sólo aparecen los campos vacíos.

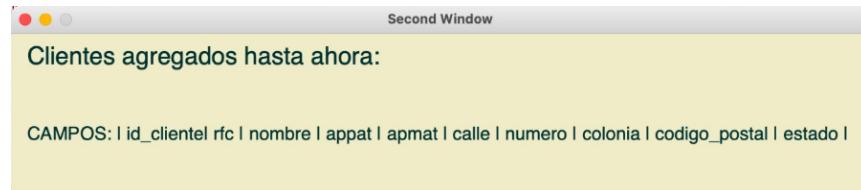


Figura 41: Pantalla vista de clientes agregdos, sin haber agregado datos.

#### 5.4.4. Pantalla para ingresar una venta

Pantalla que se muestra al usuario para ingresar una venta, el usuario marca la casilla del artículo y debe introducir la cantidad de los mismos, así como el id del cliente. Posteriormente presionar el botón de agregar venta a la base.

A screenshot of a window titled "PAPELERIA" with the sub-title "Ingresa una Venta". It says "Selecciona artículos comprados y cantidad de cada uno:" followed by a list of items with checkboxes and quantity input fields. The items are: PLUMA, LAPIZ, GOMA, TIJERAS, CUADERNO, COLORES, SACAPUNTAS, LIBRETA, and PLUMON. Below this is a text input field for "Ingresa tu id\_cliente:" and three buttons at the bottom: "Agregar venta a la base" (green), "Ver informacion de ventas" (pink), and "Cancel" (red). At the very bottom are navigation buttons: "home", "1" (highlighted in blue), and "2".

Figura 42: Pantalla de ingresar una venta

#### 5.4.5. Pantalla de ingreso de venta al oprimir ver info de ventas

Pantalla que se presenta al usuario de la información de ventas al oprimir el botón de ver información de ventas (sin haber puesto datos).

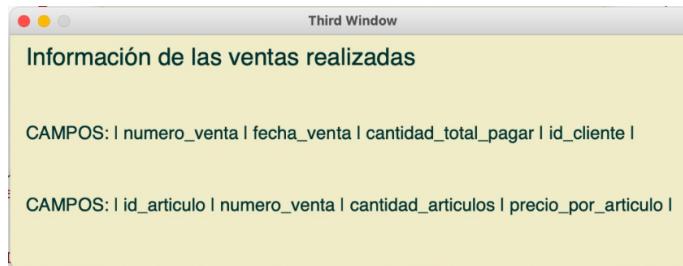


Figura 43: Pantalla de ingreso de venta al oprimir ver información de ventas

#### 5.4.6. Pantalla agregando información de un cliente

Se agrega datos del cliente llenando los campos correspondientes y posteriormente presionando el botón de agregar datos a la base.

A screenshot of a Mac OS X-style window titled "PAPELERIA". The main title bar says "Agrega la información de tu cliente:". Below it is a form with various input fields. The fields and their values are: id\_cliente (ID098261), rfc (FFGG3325), nombre (Gabriel), Apellido Paterno (Martinez), Apellido Materno (Pereda), Calle (Centenario), Numero (22), Colonia (Bosques), CP (78903), and Estado (CDMx). At the bottom of the form are three buttons: "Agregar datos a la base" (green), "Ver clientes agregados" (pink), and "Cancel" (red). Below the buttons are navigation links: "home" (blue), "1" (blue), and "2" (blue).

Figura 44: Pantalla agregando información de un cliente

#### 5.4.7. Pantalla al oprimir agregar datos

Al oprimir botón el agregar datos a la base, se limpia el campo y se agregan datos a la tabla correspondiente en postgres.

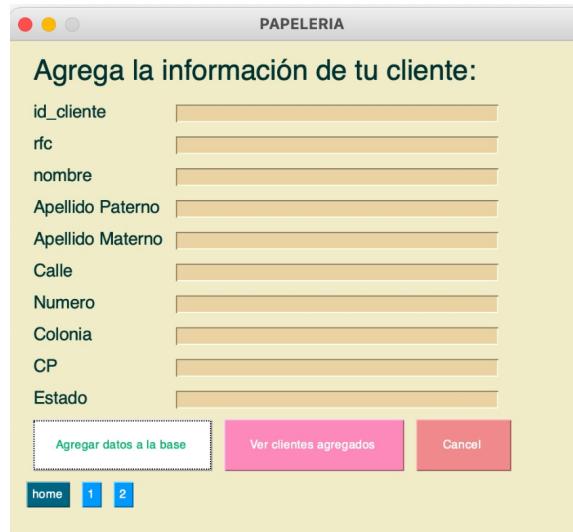


Figura 45: Pantalla al oprimir agregar datos

#### 5.4.8. Pantalla cliente al oprimir botón vista

Pantalla de clientes agregados, que se muestra al usuario al oprimir botón vista clientes agregados después de haber agregado datos. Se puede observar que se muestra el insert generado con la información dada a la base de datos, encima del insert aparecen los campos a la cual pertenece la información.

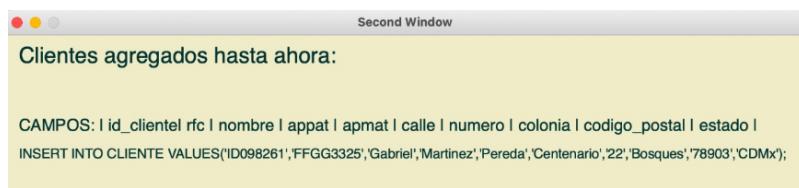


Figura 46: Pantalla cliente después de haber agregado datos

#### 5.4.9. Vista de tabla cliente en postgres antes y después de agregar datos

Vista del contenido de la tabla cliente en postgres antes y despues de agregar datos por medio de la interfaz. Se puede observar que el cliente agregado se encuentra hasta la parte inferior de la segunda tabla, que corresponde a la tabla después de haber agregado al cliente.

Figura 47: Comparativa de tabla cliente en postgres

#### 5.4.10. Agregando datos de otro cliente

Pantalla que se muestra al usuario para ingresar a otro usuario. Al terminar de ingresar los datos, el usuario debe presionar el botón de agregar datos a la base.

**PAPELERIA**

**Agrega la información de tu cliente:**

<b>id_cliente</b>	IDC16576
<b>rfc</b>	OIH35176
<b>nombre</b>	Susana
<b>Apellido Paterno</b>	Hernández
<b>Apellido Materno</b>	Fernandez
<b>Calle</b>	olimpo
<b>Numero</b>	789
<b>Colonia</b>	Tarango
<b>CP</b>	8902
<b>Estado</b>	Puebla

**Agregar datos a la base**    **Ver clientes agregados**    **Cancel**

**home**    **1**    **2**

Figura 48: Ingreso de datos de otro cliente

#### 5.4.11. Agregado de datos de un cliente

Al oprimir botón agregar datos a la base se limpia el campo y se agregan datos a la tabla correspondiente en postgres

The screenshot shows a window titled "PAPELERIA" with a light yellow background. The main title is "Agrega la información de tu cliente:". Below it is a list of fields with placeholder text:

- id\_cliente
- rfc
- nombre
- Apellido Paterno
- Apellido Materno
- Calle
- Numero
- Colonia
- CP
- Estado

At the bottom of the form are three buttons: "Agregar datos a la base" (in green), "Ver clientes agregados" (in pink), and "Cancel" (in red). Below the buttons are navigation links: "home" (dark blue), "1" (light blue), and "2" (light blue).

Figura 49: Agregado de datos de un cliente

#### 5.4.12. Pantalla cliente con elementos agregados

Pantalla que se muestra al usuario de clientes agregados al oprimir botón vista clientes agregados después de haber agregado datos. Se pueden ver los 2 clientes agregados.

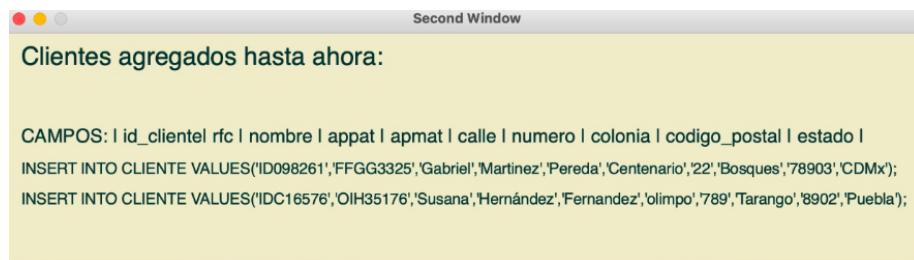


Figura 50: Pantalla cliente con elementos agregados hasta ahota

### 5.4.13. Comparativa de tabla cliente

Vista de tabla cliente en postgres cuando se agrego primer cliente y despues de agregar datos del segundo cliente, por medio de la interfaz. Se puede observar que en la parte inferior de la segunda tabla estan ingresados los datos del nuevo usuario.

id_cliente	rfc	nombre	appat	apmat	calle	numero	colonia	codigo_postal	estado
IDC24354	ERTR6453	Ernesto	Fernandez	Castrejon	Avenida Alvaro Obregon	567	Juarez	1588	CDMX
IDC32434	FGH35769	Hugo	Gomez	Flores	Eje 3 Norte	40	Polanco	45588	CDMX
IDC76898	OIH35535	Martin	Lopez	Castrejon	Rio de la Piedad	40	Compositores Mexicanos	45588	Monterrey
IDC98763	KJHG3443	Pablo	Fernandez	Maria	Avenida Presidente Masaryk	987	El Prado	45588	Guadalajara
IDC19876	OIUYU980	Ximena	Reyes	Castrejon	Avenida Tlalhuac	22	Angel Zimbron	45588	CDMX
IDC89786	6578H3HJ	Jessica	Hernandez	Ribera	Calzada de los Misterios	678	Los Pastores	45588	Monterrey
ID098768	KJHG3325	Ernesto	Rico	Valente	Eje 2 sur	220	Barrio Norte	45588	Puebla
ID098261	FFGG3325	Gabriel	Martinez	Pereda	Centenario	22	Bosques	78903	CDMx
(8 rows)									
id_cliente	rfc	nombre	appat	apmat	calle	numero	colonia	codigo_postal	estado
IDC24354	ERTR6453	Ernesto	Fernandez	Castrejon	Avenida Alvaro Obregon	567	Juarez	1588	CDMX
IDC32434	FGH35769	Hugo	Gomez	Flores	Eje 3 Norte	40	Polanco	45588	CDMX
IDC76898	OIH35535	Martin	Lopez	Castrejon	Rio de la Piedad	40	Compositores Mexicanos	45588	Monterrey
IDC98763	KJHG3443	Pablo	Fernandez	Maria	Avenida Presidente Masaryk	987	El Prado	45588	Guadalajara
IDC19876	OIUYU980	Ximena	Reyes	Castrejon	Avenida Tlalhuac	22	Angel Zimbron	45588	CDMX
IDC89786	6578H3HJ	Jessica	Hernandez	Ribera	Calzada de los Misterios	678	Los Pastores	45588	Monterrey
ID098768	KJHG3325	Ernesto	Rico	Valente	Eje 2 sur	220	Barrio Norte	45588	Puebla
ID098261	FFGG3325	Gabriel	Martinez	Pereda	Centenario	22	Bosques	78903	CDMx
IDC16576	OIH35176	Susana	Hernández	Fernandez	olimpio	789	Tarango	8902	Puebla
(9 rows)									

Figura 51: Comparativa de tabla cliente

### 5.4.14. Agregando datos de una venta

Pantalla que se muestra al usuario para ingresar datos de una venta, el usuario debe marcar las casillas que desee, y la cantidad de los mismos.

**PAPELERIA**

### Ingresar una Venta

Selecciona artículos comprados y cantidad de cada uno:

<input checked="" type="checkbox"/> PLUMA	Cantidad: <input type="text" value="3"/>
<input type="checkbox"/> LAPIZ	Cantidad: <input type="text"/>
<input checked="" type="checkbox"/> GOMA	Cantidad: <input type="text" value="5"/>
<input type="checkbox"/> TIJERAS	Cantidad: <input type="text"/>
<input checked="" type="checkbox"/> CUADERNO	Cantidad: <input type="text" value="1"/>
<input type="checkbox"/> COLORES	Cantidad: <input type="text"/>
<input type="checkbox"/> SACAPUNTAS	Cantidad: <input type="text"/>
<input type="checkbox"/> LIBRETA	Cantidad: <input type="text"/>
<input type="checkbox"/> PLUMON	Cantidad: <input type="text"/>

Ingresar tu id\_cliente:

**Agregar venta a la base**   **Ver información de ventas**   **Cancelar**

**home**   **1**   **2**

Figura 52: Agregando datos de una venta

#### 5.4.15. Oprimiendo botón agregar venta

El usuario al oprimir el botón de agregar venta a la base, se agrega la venta y se limpia el formulario, ademas se agregan los datos a la tabla correspondiente en postgres.

**PAPELERIA**

### Ingresar una Venta

Selecciona artículos comprados y cantidad de cada uno:

<input checked="" type="checkbox"/> PLUMA	Cantidad: <input type="text"/>
<input type="checkbox"/> LAPIZ	Cantidad: <input type="text"/>
<input checked="" type="checkbox"/> GOMA	Cantidad: <input type="text"/>
<input type="checkbox"/> TIJERAS	Cantidad: <input type="text"/>
<input checked="" type="checkbox"/> CUADERNO	Cantidad: <input type="text"/>
<input type="checkbox"/> COLORES	Cantidad: <input type="text"/>
<input type="checkbox"/> SACAPUNTAS	Cantidad: <input type="text"/>
<input type="checkbox"/> LIBRETA	Cantidad: <input type="text"/>
<input type="checkbox"/> PLUMON	Cantidad: <input type="text"/>

Ingresar tu id\_cliente:

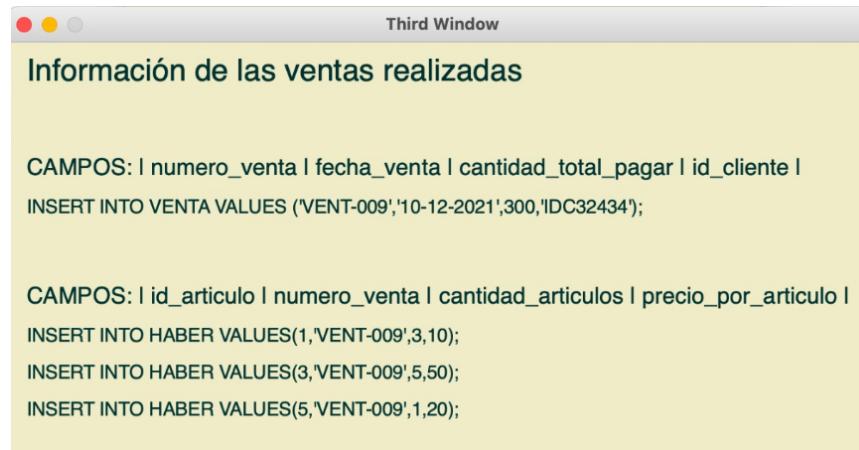
**Agregar venta a la base**   **Ver información de ventas**   **Cancelar**

**home**   **1**   **2**

Figura 53: Agregando datos de una venta

#### 5.4.16. Información de ventas realizadas

Pantalla de información de las ventas, posterior al haber oprimido el botón de ver informacion de als ventas y haber agregado datos de la venta. Se pueden observar los datos de ventas agregadas previamente, se observan los campos y los inserts de manera ordenada.



```
Third Window
Información de las ventas realizadas

CAMPOS: | numero_venta | fecha_venta | cantidad_total_pagar | id_cliente |
INSERT INTO VENTA VALUES ('VENT-009','10-12-2021',300,'IDC32434');

CAMPOS: | id_articulo | numero_venta | cantidad_articulos | precio_por_articulo |
INSERT INTO HABER VALUES(1,'VENT-009',3,10);
INSERT INTO HABER VALUES(3,'VENT-009',5,50);
INSERT INTO HABER VALUES(5,'VENT-009',1,20);
```

Figura 54: Información de ventas realizadas

#### 5.4.17. Comparativa de tablas sin y con elementos agregados.

Vista de las tablas venta y haber, antes y despues de haber ingresado datos. Se pueden observar en las tablas 3 y 4 las actualizaciones con los datos ingresados de manera respectiva.

```

|papeleria=# select * from venta;
+-----+-----+-----+-----+
| numero_venta | fecha_venta | cantidad_total_pagar | id_cliente |
+-----+-----+-----+-----+
| VENT-001 | 8-8-2019 | 50 | IDC24354 |
| VENT-002 | 18-12-2019 | 18 | ID098768 |
| VENT-003 | 10-7-2018 | 12 | IDC89786 |
| VENT-004 | 31-1-2020 | 24 | ID098768 |
| VENT-005 | 8-12-2020 | 80 | IDC76890 |
| VENT-006 | 5-1-2021 | 130 | IDCi9876 |
| VENT-007 | 14-2-2020 | 138 | IDC98763 |
| VENT-008 | 9-10-2019 | 4 | IDC24354 |
+-----+
(8 rows)

|papeleria=# select * from haber;
+-----+-----+-----+-----+
| id_articulo | numero_venta | cantidad_articulos | precio_por_articulo |
+-----+-----+-----+-----+
| 1 | VENT-001 | 100 | 6 |
| 5 | VENT-001 | 10 | 6 |
| 2 | VENT-002 | 50 | 5 |
| 3 | VENT-003 | 3 | 8 |
| 1 | VENT-004 | 5 | 7 |
| 3 | VENT-005 | 80 | 8 |
| 4 | VENT-006 | 90 | 12 |
| 5 | VENT-006 | 110 | 25 |
| 6 | VENT-007 | 130 | 100 |
| 7 | VENT-008 | 70 | 12 |
| 8 | VENT-008 | 40 | 35 |
+-----+
(11 rows)

|papeleria=# select * from venta;
+-----+-----+-----+-----+
| numero_venta | fecha_venta | cantidad_total_pagar | id_cliente |
+-----+-----+-----+-----+
| VENT-001 | 8-8-2019 | 50 | IDC24354 |
| VENT-002 | 18-12-2019 | 18 | ID098768 |
| VENT-003 | 10-7-2018 | 12 | IDC89786 |
| VENT-004 | 31-1-2020 | 24 | ID098768 |
| VENT-005 | 8-12-2020 | 80 | IDC76890 |
| VENT-006 | 5-1-2021 | 130 | IDCi9876 |
| VENT-007 | 14-2-2020 | 138 | IDC98763 |
| VENT-008 | 9-10-2019 | 4 | IDC24354 |
| VENT-009 | 10-12-2021 | 300 | IDC32434 |
+-----+
(9 rows)

|papeleria=# select * from haber;
+-----+-----+-----+-----+
| id_articulo | numero_venta | cantidad_articulos | precio_por_articulo |
+-----+-----+-----+-----+
| 1 | VENT-001 | 100 | 6 |
| 5 | VENT-001 | 10 | 6 |
| 2 | VENT-002 | 50 | 5 |
| 3 | VENT-003 | 3 | 8 |
| 1 | VENT-004 | 5 | 7 |
| 3 | VENT-005 | 80 | 8 |
| 4 | VENT-006 | 90 | 12 |
| 5 | VENT-006 | 110 | 25 |
| 6 | VENT-007 | 130 | 100 |
| 7 | VENT-008 | 70 | 12 |
| 8 | VENT-008 | 40 | 35 |
| 1 | VENT-009 | 3 | 10 |
| 3 | VENT-009 | 5 | 50 |
| 5 | VENT-009 | 1 | 20 |
+-----+
(14 rows)

```

Figura 55: Comparativa de tablas sin y con elementos agregados

#### 5.4.18. Agregando datos de otra venta

Pantalla que se muestra al usuario al agregar nueva información de una venta, seleccionando el artículo y la cantidad de los mismos, ademas del id del cliente.

The screenshot shows a software window titled "PAPELERIA". The main title is "Ingresa una Venta". Below it, a sub-instruction reads "Selecciona artículos comprados y cantidad de cada uno:". A list of items with checkboxes and quantity inputs follows:

<input type="checkbox"/> PLUMA	Cantidad: <input type="text"/>
<input type="checkbox"/> LAPIZ	Cantidad: <input type="text"/>
<input type="checkbox"/> GOMA	Cantidad: <input type="text"/>
<input checked="" type="checkbox"/> TIJERAS	Cantidad: <input type="text" value="1"/>
<input type="checkbox"/> CUADERNO	Cantidad: <input type="text"/>
<input checked="" type="checkbox"/> COLORES	Cantidad: <input type="text" value="30"/>
<input type="checkbox"/> SACAPUNTAS	Cantidad: <input type="text"/>
<input type="checkbox"/> LIBRETA	Cantidad: <input type="text"/>
<input checked="" type="checkbox"/> PLUMON	Cantidad: <input type="text" value="14"/>

Below the list, there is a text input field labeled "Ingresa tu id\_cliente:" containing the value "IDC98763". At the bottom of the window are three buttons: "Agregar venta a la base" (green), "Ver informacion de ventas" (pink), and "Cancel" (red). Navigation links at the bottom include "home", "1", and "2".

Figura 56: Agregando datos de otra venta

#### 5.4.19. Oprimiendo botón agregar venta

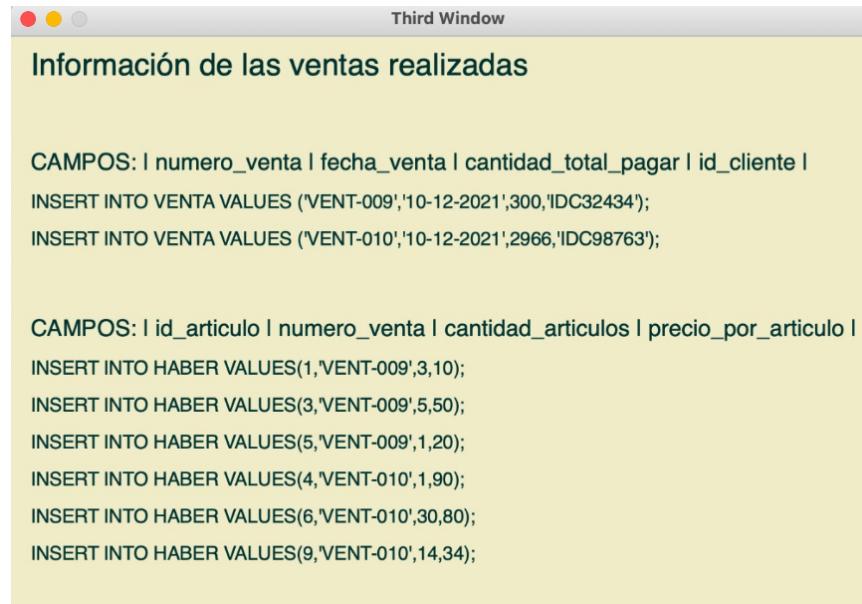
Al oprimir botón de agregar venta a la base, esta se limpia y se agregan los datos a la tabla correspondiente en postgres.



Figura 57: Oprimiendo boton agregar venta

#### 5.4.20. Pantalla de información de ventas realizadas

Pantalla que se muestra al usuario sobre sobre la información de las ventas realizadas, después de haber oprimido el botón de ver informacion ventas. Se puede observar que se muestran los inserts con los datos ingresados de todas las ventas, ademas de los campos de cada uno.



```
Third Window
Información de las ventas realizadas

CAMPOS: | numero_venta | fecha_venta | cantidad_total_pagar | id_cliente |
INSERT INTO VENTA VALUES ('VENT-009','10-12-2021',300,'IDC32434');
INSERT INTO VENTA VALUES ('VENT-010','10-12-2021',2966,'IDC98763');

CAMPOS: | id_articulo | numero_venta | cantidad_articulos | precio_por_articulo |
INSERT INTO HABER VALUES(1,'VENT-009',3,10);
INSERT INTO HABER VALUES(3,'VENT-009',5,50);
INSERT INTO HABER VALUES(5,'VENT-009',1,20);
INSERT INTO HABER VALUES(4,'VENT-010',1,90);
INSERT INTO HABER VALUES(6,'VENT-010',30,80);
INSERT INTO HABER VALUES(9,'VENT-010',14,34);
```

Figura 58: Pantalla de información de ventas realizadas

#### 5.4.21. Comparación de tablas antes y después de agregar la venta

Vista de las tablas venta y haber, antes y despues de haber ingresado datos. Sepueden observar en las tablas 3 y 4 las actualizaciones con los datos ingresados de manera respectiva.

```
papelaria=# select * from venta;
numero_venta | fecha_venta | cantidad_total_pagar | id_cliente
-----+-----+-----+-----+
VENT-001 | 8-8-2019 | 50 | IDC24354
VENT-002 | 18-12-2019 | 18 | ID098768
VENT-003 | 10-7-2018 | 12 | IDC89786
VENT-004 | 31-1-2020 | 24 | ID098768
VENT-005 | 8-12-2020 | 80 | IDC76890
VENT-006 | 5-1-2021 | 130 | IDC19876
VENT-007 | 14-2-2020 | 138 | IDC98763
VENT-008 | 9-10-2019 | 4 | IDC24354
VENT-009 | 10-12-2021 | 300 | IDC32434
(9 rows)

papelaria=# select * from haber;
id_articulo | numero_venta | cantidad_articulos | precio_por_articulo
-----+-----+-----+-----+
1 | VENT-001 | 100 | 6
5 | VENT-001 | 10 | 6
2 | VENT-002 | 50 | 5
3 | VENT-003 | 3 | 8
1 | VENT-004 | 5 | 7
3 | VENT-005 | 80 | 8
4 | VENT-006 | 90 | 12
5 | VENT-006 | 110 | 25
6 | VENT-007 | 130 | 100
7 | VENT-008 | 70 | 12
8 | VENT-008 | 40 | 35
1 | VENT-009 | 3 | 10
3 | VENT-009 | 5 | 50
5 | VENT-009 | 1 | 20
(14 rows)

papelaria=# select * from venta;
numero_venta | fecha_venta | cantidad_total_pagar | id_cliente
-----+-----+-----+-----+
VENT-001 | 8-8-2019 | 50 | IDC24354
VENT-002 | 18-12-2019 | 18 | ID098768
VENT-003 | 10-7-2018 | 12 | IDC89786
VENT-004 | 31-1-2020 | 24 | ID098768
VENT-005 | 8-12-2020 | 80 | IDC76890
VENT-006 | 5-1-2021 | 130 | IDC19876
VENT-007 | 14-2-2020 | 138 | IDC98763
VENT-008 | 9-10-2019 | 4 | IDC24354
VENT-009 | 10-12-2021 | 300 | IDC32434
VENT-010 | 10-12-2021 | 2966 | IDC98763
(10 rows)

papelaria=# select * from haber;
id_articulo | numero_venta | cantidad_articulos | precio_por_articulo
-----+-----+-----+-----+
1 | VENT-001 | 100 | 6
5 | VENT-001 | 10 | 6
2 | VENT-002 | 50 | 5
3 | VENT-003 | 3 | 8
1 | VENT-004 | 5 | 7
3 | VENT-005 | 80 | 8
4 | VENT-006 | 90 | 12
5 | VENT-006 | 110 | 25
6 | VENT-007 | 130 | 100
7 | VENT-008 | 70 | 12
8 | VENT-008 | 40 | 35
1 | VENT-009 | 3 | 10
3 | VENT-009 | 5 | 50
5 | VENT-009 | 1 | 20
4 | VENT-010 | 1 | 90
6 | VENT-010 | 30 | 80
9 | VENT-010 | 14 | 34
(17 rows)
```

Figura 59: Comparación de tablas antes y después de agregar la venta

#### 5.4.22. Terminal mientras se corría el código de python

Así se muestra la terminal mientras se ejecutaba el código de python (Screenshots anteriores), se nos muestra las ejecuciones realizadas y nos informa la ejecución de los comandos de manera correra, los inserts y las vistas de información realizadas.

```
[francineochoafernandez@Francines-MacBook-Pro Desktop % python3 ProyectoFinal.py
Comando corrió correctamente en PostgreSQL
1
2
1
Ver clientes agregados
2
Ver informacion de ventas
1
Aregar datos a la base
Comando corrió correctamente en PostgreSQL
Ver clientes agregados
Aregar datos a la base
Comando corrió correctamente en PostgreSQL
Ver clientes agregados
2
Aregar venta a la base
Comando corrió correctamente en PostgreSQL
Ver informacion de ventas
Aregar venta a la base
Comando corrió correctamente en PostgreSQL
Ver informacion de ventas
None
```

Figura 60: Terminal mientras se corría el código de python

## 6. Conclusiones

- **Cabrera Barzalobre Jessica Jazmín**

Al dar vida a este proyecto pude llegar a concluir que fue la manera correcta para dar cierre a todo lo aprendido en el curso de Bases de Datos. Desde el desarrollo conceptual de un problema a través de las herramientas del Modelo Entidad Relación, Modelo relacional y la normalización. Las fases conceptuales fueron fundamentales para lograr concebir la parte práctica con facilidad. Se lograron poner en práctica los conceptos básicos tanto del DDL como el del DML para así poner en funcionamiento la base de datos correspondiente a la problemática principal. Desde la creación de tablas hasta el desarrollo de funciones para así manipular de manera correcta la base de datos para la papelería.

- **Ochoa Fernández Francine**

Este proyecto me parecio una gran manera de aplicar todo lo aprendido en el curso. Se aplicó la teoría vista sobre diagramas entridad relacion, para despues hacer el modelo relacional. Al tener nuestros driagramas, por medio del manejador postgreSQL, con DML se pudieron crear las tablas junto con los constraints necesarios para posteriormente agregarle valores. Despues por medio de programacion en plpgsql se crearon las funciones, triggers y vistas necesarias para cumplir las necesidades del proyecto papeleria. Se hizo uso de python y las librerias necesarias para conectar los scripst de python con postgreSQL y hacer una interfaz que controlara la insercion y lectura de datos a la base 'papeleria'. Se apredió mucho y me parece que se lograron con exito los requerimientos del proyecto.

- **Reyes Martínez Juan Carlos**

En este proyecto arendí mucho sobre el trabajo colaborativo, ya que exigío nuestros conocimientos aprendidos en el curso y cómo complementar las habilidades de todo el equipo, en lo personal tuve que aprender y entender cómo usar LATEX para generar la documentación y tener al nivel los comandos de

postgres al igual que los comandos de sql oracle que es lo que utilizaba con mayor regularidad. Además de la importancia del uso herramientas colaborativas para generar la misma.

Durante el desarrollo del sistema de base de datos desde cero, se inició el análisis del problema para posteriormente tener claro los requerimientos del proyecto y así diseñar el sistema de base de datos. Lo que vimos en clase se implementó en este proyecto,y esto me hizo entender mejor los puntos clave al tener un problema de base de datos real.

## 7. Referencias

- 3.10.1 Documentation. (s. f.). Documentación Python. Recuperado 6 de diciembre de 2021, de <https://docs.python.org/3/>
- PostgreSQL: Documentation. (s. f.). The PostgreSQL Global Development Group. Recuperado 5 de diciembre de 2021, de <https://www.postgresql.org/docs/>