



Universidad Nacional Autónoma de México

Facultad de Ingeniería

*Base de datos*

PROYECTO FINAL

*ING. FERNANDO ARREOLA FRANCO*

*Carrillo Cervantes Ivette Alejandra*

*Chilpa Navarro Martin Enrique*

*Juárez Juárez María José*

*Nava Alberto Vanessa*

*Sotelo Ramírez Axel Rodrigo*

*10 Junio 2023*

# Índice

<b>1. Introducción</b>	<b>2</b>
<b>2. Objetivo</b>	<b>2</b>
<b>3. Plan de trabajo</b>	<b>2</b>
3.1. Análisis de requerimientos . . . . .	2
3.2. Modelo conceptual . . . . .	3
3.3. Modelo lógico . . . . .	3
3.4. Normalización . . . . .	4
3.5. Representación del modelo lógico . . . . .	4
3.6. Modelo físico . . . . .	4
3.7. Interfaz gráfica . . . . .	5
3.8. Pruebas . . . . .	5
3.9. Extra . . . . .	5
<b>4. Conocimientos previos</b>	<b>5</b>
<b>5. Diseño</b>	<b>6</b>
5.1. Análisis de requerimientos: . . . . .	6
5.2. Diseño conceptual: . . . . .	6
5.3. Diseño lógico: . . . . .	6
5.4. Diseño físico: . . . . .	6
<b>6. Modelo entidad relación</b>	<b>7</b>
<b>7. Modelo relacional</b>	<b>7</b>
7.1. Intermedio . . . . .	7
7.2. Final . . . . .	8
<b>8. Script para creación de base de datos</b>	<b>9</b>
<b>9. Implementación</b>	<b>14</b>
9.1. Creación de la base de datos . . . . .	14
9.2. Creación de tablas . . . . .	14
9.3. Consultas . . . . .	14
9.4. Triggers . . . . .	14
9.5. Functions . . . . .	14
<b>10.Datos de tablas</b>	<b>15</b>
10.1. Tabla: Telefono . . . . .	15
10.2. Tabla: Categoria . . . . .	15
10.3. Tabla: Cliente . . . . .	16
10.4. Tabla: Surte . . . . .	16
10.5. Tabla: Sucursal . . . . .	17
10.6. Tabla: Proveedor . . . . .	17
10.7. Tabla: Venta . . . . .	18
10.8. Tabla: Guarda . . . . .	18
10.9. Tabla: Producto . . . . .	19
<b>11.Interfaz gráfica</b>	<b>19</b>

<b>12. Conclusiones</b>	<b>22</b>
12.1. Carrillo Cervantes Ivette Alejandra . . . . .	22
12.2. Chilpa Navarro Martin Enrique . . . . .	22
12.3. Nava Alberto Vanessa . . . . .	22
12.4. Juárez Juárez María José . . . . .	22
12.5. Sotelo Ramírez Axel Rodrigo . . . . .	22

## 1. Introducción

La gestión eficiente de la información es crucial en cualquier negocio, y las bases de datos desempeñan un papel fundamental en este proceso. En este proyecto, abordaremos el diseño y desarrollo de un sistema de base de datos para una papelería, con el objetivo de llevar un control exhaustivo de los proveedores, clientes, inventario, productos y demás.

El enfoque principal de este proyecto es proporcionar una solución integral que optimice la gestión de la papelería, permitiendo un seguimiento preciso de las transacciones, un mejor control de stock y una atención más eficiente a los clientes. A través del uso de una base de datos, buscamos agilizar las operaciones diarias y brindar una visión clara y actualizada de todos los aspectos relacionados con el negocio.

En las secciones siguientes, presentaremos el modelo entidad-relación y el modelo relacional propuestos, así como los diagramas y tablas que ilustran la estructura de la base de datos. Además, se describirán detalladamente las funcionalidades clave del sistema, abordando aspectos como la gestión de proveedores, clientes, inventario, productos y transacciones. Por último, se discutirán las tecnologías y herramientas utilizadas para la implementación del sistema, así como las consideraciones de seguridad y privacidad de la información.

## 2. Objetivo

El objetivo principal de este proyecto fue diseñar desde el modelo entidad relación hasta el diseño físico de una base de datos la cual ayude en la administración de una papelería, llevando el control de los productos, ventas, categorías, etc. También se planteó como objetivo optimizar el rendimiento del sistema de base de datos. Se identificaron consultas y operaciones lentas o ineficientes, y se aplicaron técnicas y estrategias para mejorar la velocidad de respuesta del sistema. Se llevaron a cabo ajustes en los índices, optimización de consultas y ajustes en la configuración del servidor de base de datos. El objetivo final era garantizar un rendimiento óptimo del sistema.

## 3. Plan de trabajo

### 3.1. Análisis de requerimientos

Actividad	Responsable	Grado de Prioridad	Fecha de Inicio	Fecha final
Análisis de requerimientos				
Identificación de requerimientos necesarios para el proyecto.	Carrillo Chilpa Juárez Nava Sotelo	Alta	8/05/2023	10/05/2023
Definición de entidades y atributos.	Carrillo Chilpa Juárez Nava Sotelo	Alta	10/05/2023	13/05/2023

Figura 2: Análisis de requerimientos

### 3.2. Modelo conceptual

Modelo Conceptual				
Realización de propuestas individuales del MER.	Carrillo Chilpa Nava	Alta	15/05/2023	20/05/2023
Unión de propuestas dada.	Nava	Media	20/05/2023	23/05/2023
Validación del MER.	Carrillo Chilpa Juárez Nava	Alta	23/05/23023	25/05/2023
Ajustes finales del MER.	Nava	Media	25/05/2023	26/05/2023
Validación Final del MER.	Carrillo Chilpa Juárez Nava Sotelo	Alta	26/05/2023	27/05/2023

Figura 3: Modelo conceptual

### 3.3. Modelo lógico

Modelo lógico				
Transformación del diagrama a relaciones.	Juárez Nava	Media	27/05/2023	28/05/2023
Corrección de la representación intermedia.	Carrillo Chilpa	Alta	28/05/2023	30/05/2023
Validación de la representación intermedia.	Carrillo Chilpa Juárez Nava	Alta	30/05/2023	30/05/2023

Figura 4: Modelo lógico

### 3.4. Normalización

Normalización				
Aplicación de 1FN, 2FN y 3FN.	Carrillo Chilpa Juárez Nava	Media	30/05/2023	31/05/2023
Validación de la normalización.	Carrillo Chilpa Juárez Nava Sotelo	Alta	31/05/2023	31/05/2023

Figura 5: Normalización

### 3.5. Representación del modelo lógico

Representación del modelo lógico				
Mapeo final en un software especializado	Juárez Nava	Media	31/05/2023	31/05/2023
Modificación del mapeo final	Carrillo Chilpa	Media	1/06/2023	1/06/2023
Validación del mapeo final.	Carrillo Chilpa Juárez Nava	Alta	2/06/2023	3/05/2023

Figura 6: Representación del modelo lógico

### 3.6. Modelo físico

Modelo físico				
Creación del código.	Carrillo Chilpa Juárez	Alta	3/06/2023	4/06/2023
Creación de registros y creación de csv.	Nava	Media	3/06/2023	4/06/2023
Planeación y programación de consultas.	Carrillo Chilpa Juárez Nava Sotelo	Media	4/06/2023	4/06/2023
Validación del modelo físico.	Carrillo Chilpa Juárez Nava	Alta	5/06/2023	6/06/2023

Figura 7: Modelo físico

### 3.7. Interfaz gráfica

Interfaz gráfica				
Creación y modificación de la interfaz gráfica.	Carrillo	Media	1/06/2023	9/06/2023
Conectar la base de datos con la interfaz.	Chilpa	Alta	9/06/2023	9/06/2023
Finalización y validación de interfaz gráfica.	Carrillo Chilpa	Alta	9/06/2023	9/06/2023

Figura 8: Interfaz gráfica

### 3.8. Pruebas

Pruebas				
Pruebas de la interfaz.	Carrillo Chilpa Juárez Nava	Media	10/06/2023	10/06/2023
Ajustes finales.	Carrillo Chilpa Juárez Nava Sotelo	Alta	10/06/2023	10/06/2023

Figura 9: Pruebas

### 3.9. Extra

Extra				
Creación y modificación del documentación	Carrillo Chilpa Juárez Nava Sotelo	Alta	10/05/2023	10/06/2023
Modificación de archivos csv.	Nava	Media	4/06/2023	9/06/2023

Figura 10: Extra

## 4. Conocimientos previos

Antes de adentrarnos en el diseño y desarrollo del sistema de base de datos para la papelería, es importante contar con algunos conocimientos previos que nos permitirán comprender mejor los conceptos y principios fundamentales de las bases de datos. A continuación, se presentan algunos conceptos clave que serán útiles en el contexto de este proyecto:

- **Base de datos:** Una base de datos es un conjunto estructurado de datos relacionados entre sí, organizados y almacenados de manera sistemática para su posterior consulta y manipulación. Proporciona un medio eficiente para almacenar y recuperar información de manera confiable.

- **Sistema de gestión de bases de datos (SGBD):** Un SGBD es un software que permite la creación, manipulación y administración de bases de datos. Facilita el acceso a los datos, la implementación de consultas, la gestión de transacciones y la garantía de integridad y seguridad de la información.
- **Modelo entidad-relación (ER):** El modelo entidad-relación es una representación gráfica que describe las entidades (objetos o conceptos del mundo real) y las relaciones entre ellas. Proporciona una vista abstracta de la estructura de la base de datos y sirve como base para el diseño de la misma.
- **Modelo relacional:** El modelo relacional es un modelo de datos que representa la información en forma de tablas o relaciones. Se basa en el concepto de conjuntos y utiliza claves primarias y claves foráneas para establecer las relaciones entre las tablas.
- **SQL (Structured Query Language):** SQL es un lenguaje de consulta estructurado utilizado para interactuar con bases de datos relacionales. Permite realizar consultas, inserciones, actualizaciones y eliminaciones de datos, así como definir la estructura y las restricciones de la base de datos.

Estos conceptos serán fundamentales para comprender y desarrollar adecuadamente el sistema de base de datos para la papelería. A medida que avancemos en el proyecto, profundizaremos en cada uno de ellos y los aplicaremos en el diseño y la implementación del sistema.

## 5. Diseño

Durante el proceso de diseño de la base de datos, se llevaron a cabo varias fases para lograr un resultado óptimo que cumpliera con los requerimientos planteados por la cadena de papelerías. A continuación, se describen brevemente las fases realizadas y los resultados obtenidos en cada una de ellas.

### 5.1. Análisis de requerimientos:

En esta fase, se recopilaron y analizaron detalladamente los requerimientos de la cadena de papelerías. Se identificaron las entidades clave, como proveedores, clientes, productos, ventas y artículos, así como sus atributos y relaciones. Se obtuvo una visión clara de las necesidades de la empresa y los datos que debían ser almacenados en la base de datos.

### 5.2. Diseño conceptual:

Con base en el análisis de requerimientos, se desarrolló un diseño conceptual de la base de datos utilizando un enfoque de modelo entidad-relación. Se crearon diagramas que representaban las entidades, sus atributos y las relaciones entre ellas. Se obtuvo una representación visual de la estructura general de la base de datos y cómo se relacionaban los diferentes elementos.

### 5.3. Diseño lógico:

En esta etapa, se tradujo el diseño conceptual en un modelo lógico de la base de datos utilizando el modelo relacional. Se definieron las tablas y se especificaron las claves primarias y foráneas, asegurando la integridad de los datos. Se realizaron ajustes y refinamientos en el diseño para garantizar la eficiencia y consistencia de la base de datos.

### 5.4. Diseño físico:

En esta fase, se llevaron a cabo decisiones relacionadas con la implementación técnica de la base de datos. Se seleccionaron los tipos de datos adecuados para cada atributo, se establecieron restricciones y se definieron índices para mejorar el rendimiento de las consultas. Además, se consideraron aspectos de rendimiento y escalabilidad para garantizar un funcionamiento eficiente de la base de datos.

Los resultados obtenidos de estas fases de diseño fueron una estructura de base de datos bien definida, con tablas y relaciones claras, asegurando la integridad y consistencia de los datos almacenados. Se logró un diseño lógico y físico eficiente, optimizado para realizar consultas rápidas y precisas. Además, se establecieron índices que mejoraron la velocidad de búsqueda de datos y se aplicaron restricciones para garantizar la integridad de los datos.

## 6. Modelo entidad relación

En el modelo entidad relación se muestran 6 entidades, siendo estas: Proveedor, Producto, Sucursal, Venta, Cliente, Categoría. De igual forma se muestran 5 relaciones que hacen que el modelo entidad relación tenga sentido. Del mismo modo cada entidad tiene consigo sus respectivos atributos, entando marcado si son multivaluados, opcionales, derivados, etc. En el diagrama se muestra, de igual forma, las cardinalidades que tiene cada relación, esto es de gran ayuda, ya que nos permite entender cómo funciona nuestro modelo.

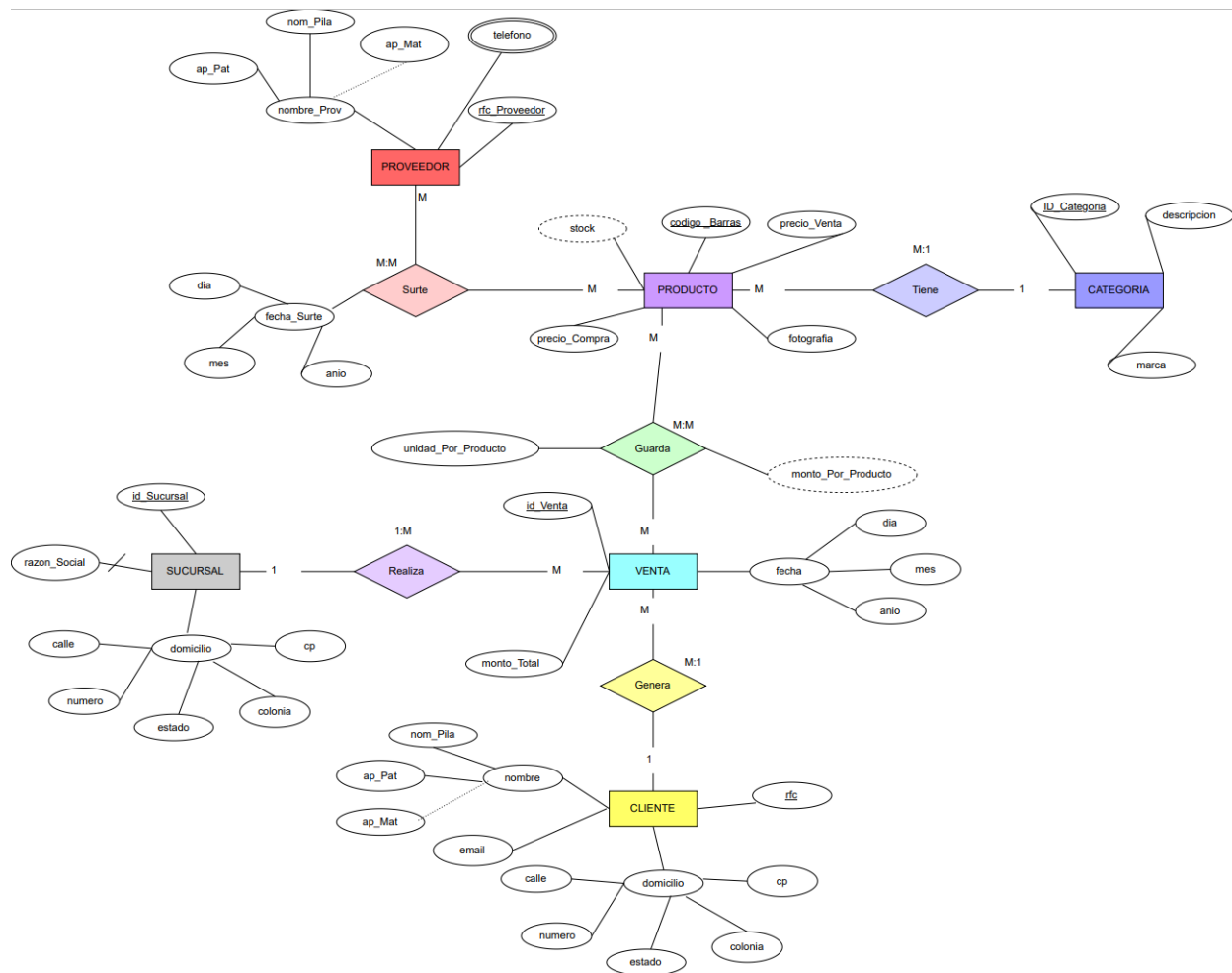


Figura 11: Modelo entidad relación

## 7. Modelo relacional

### 7.1. Intermedio

El modelo relacional intermedio nos ayuda bastante ya que con él podemos ver de manera más clara los atributos y su respectivo tipo de dato. También este modelo nos ayuda a visualizar cómo las relaciones, dependiendo de su cardinalidad, se vuelven tablas o llaves foráneas.

A continuación se presenta el diseño del modelo relacional intermedio que resultó de convertir el modelo entidad relación.



Cuadro 1: Modelo intermedio

Nombre	Atributos	Tipo de dato	Restricción
PRODUCTO	codigo_barras	BIGINT	PK
	precio_venta	NUMBER	-
	precio_compra	NUMBER	-
	fotografía	TEXT	-
	stock	SMALLINT	C
	id_categoria	INTEGER	FK
CATEGORIA	Id_categoria	INTEGER	PK
	marca	VARCHAR[50]	-
	descripción	TEXT	-
PROVEEDOR	RFC_proveedor	CHAR [13]	PK
	nombre	VARCHAR[60]	-
	ap_materno	VARCHAR[60]	N
	ap_paterno	VARCHAR[60]	-
CLIENTE	RFC_cliente	CHAR[13]	PK
	nom_pila	VARCHAR[60]	-
	ap_pat	VARCHAR[60]	-
	ap_mat	VARCHAR[60]	N
	calle	VARCHAR[70]	-
	numero	SMALLINT	-
	estado	VARCHAR[60]	-
	colonia	VARCHAR[60]	-
	CP	SMALLINT	-
VENTA	id_venta	VARCHAR[10]	PK
	fecha	DATE	-
	monto_final	FLOAT	-
	RFC_cliente	CHAR[13]	FK
	id_sucursal	VARCHAR[10]	FK
SUCURSAL	id_sucursal	VARCHAR[10]	PK
	razon_social	VARCHAR[100]	U
	calle	VARCHAR[70]	-
	numero	SMALLINT	-
	estado	VARCHAR[60]	-
	colonia	VARCHAR[60]	-
TELEFONO	num_tel	BIGINT	PK
	RFC_proveedor	CHAR[13]	FK
SURTE	fecha_surte	DATE	-
	RFC_proveedor	CHAR[13]	PK, FK
	codigo_barras	BIGINT	PK, FK
GUARDA	monto_producto	FLOAT	C
	cantidad_producto	SMALLINT	-
	id_venta	VARCHAR[10]	PK, FK
	codigo_barras	BIGINT	PK, FK

## 7.2. Final

El modelo relacional final resultó ser una herramienta invaluable en el proceso de diseño de la base de datos. Nos permitió tener una visión clara y detallada de los atributos y sus respectivos tipos de datos. Esta representación visual nos facilitó comprender la estructura de la base de datos y la forma en que los datos se relacionan entre sí.

Gracias al modelo relacional, pudimos identificar y definir las relaciones entre las entidades. Dependiendo de la cardinalidad de estas relaciones, algunas se convirtieron en tablas separadas, mientras que otras se representaron como claves foráneas en las tablas principales. Esta organización nos brindó una comprensión clara de cómo los datos se conectan y cómo se pueden acceder de manera eficiente.

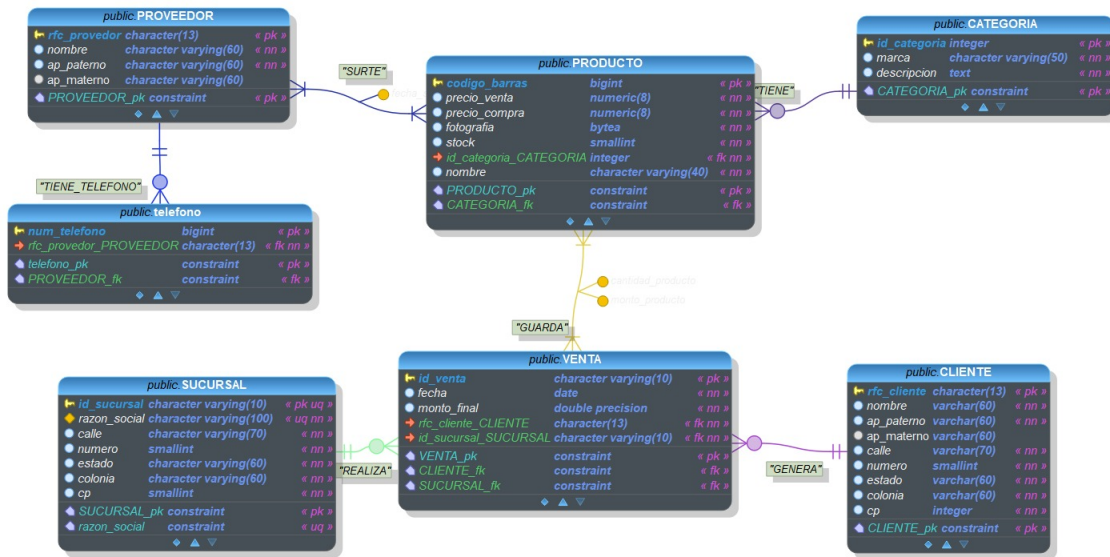


Figura 12: Caption

## 8. Script para creación de base de datos

```
-- Database generated with pgModeler (PostgreSQL Database Modeler).
-- pgModeler version: 1.0.3
-- PostgreSQL version: 15.0
-- Project Site: pgmodeler.io
--Autores:
-- Carrillo, Chilpa, Jurez, Sotelo y Nava.

-----Creacin de base de datos

CREATE DATABASE new_database;

-----Creacin de tablas

-- object: public.PRODUCTO
CREATE TABLE public.PRODUCTO (
    codigo_barras bigint NOT NULL,
    precio_venta numeric(8) NOT NULL,
    precio_compra numeric(8) NOT NULL,
    fotografia bytea NOT NULL,
    stock smallint NOT NULL,
    id_categoria_CATEGORIA integer NOT NULL,
    nombre character varying NOT NULL,
    CONSTRAINT PRODUCTO_pk PRIMARY KEY (codigo_barras)
);

-- object: public.CATEGORIA
CREATE TABLE public.CATEGORIA (
    id_categoria integer NOT NULL,
    marca varchar(50) NOT NULL,
    descripcion text NOT NULL,
    CONSTRAINT CATEGORIA_pk PRIMARY KEY (id_categoria)
);

-- object: public.PROVEEDOR
CREATE TABLE public.PROVEEDOR (
```

```

    rfc_proveedor char(13) NOT NULL,
    nombre varchar(60) NOT NULL,
    ap_paterno varchar(60) NOT NULL,
    ap_materno varchar(60),
    CONSTRAINT PROVEEDOR_pk PRIMARY KEY (rfc_proveedor)
);

-- object: public.CLIENTE
CREATE TABLE public.CLIENTE (
    rfc_cliente char(13) NOT NULL,
    nombre varchar(60) NOT NULL,
    ap_paterno varchar(60) NOT NULL,
    ap_materno varchar(60),
    calle varchar(70) NOT NULL,
    numero smallint NOT NULL,
    estado varchar(60) NOT NULL,
    colonia varchar(60) NOT NULL,
    cp integer NOT NULL,
    CONSTRAINT CLIENTE_pk PRIMARY KEY (rfc_cliente)
);

-- object: public.SUCURSAL
CREATE TABLE public.SUCURSAL (
    id_sucursal varchar(10) NOT NULL,
    razon_social varchar(100) NOT NULL,
    calle varchar(70) NOT NULL,
    numero smallint NOT NULL,
    estado varchar(60) NOT NULL,
    colonia varchar(60) NOT NULL,
    cp smallint NOT NULL,
    CONSTRAINT SUCURSAL_pk PRIMARY KEY (id_sucursal),
    CONSTRAINT razon_social UNIQUE (id_sucursal,razon_social)
);

-- object: public.VENTA
CREATE TABLE public.VENTA (
    id_venta varchar(10) NOT NULL,
    fecha date NOT NULL,
    monto_final float NOT NULL,
    rfc_cliente_CLIENTE char(13) NOT NULL,
    id_sucursal_SUCURSAL varchar(10) NOT NULL,
    CONSTRAINT VENTA_pk PRIMARY KEY (id_venta)
);

-- object: public.telefono
CREATE TABLE public.telefono (
    rfc_proveedor_PROVEEDOR char(13) NOT NULL,
    num_telefono bigint NOT NULL,
    CONSTRAINT telefono_pk PRIMARY KEY (num_telefono)
);

-- object: public.GUARDA
CREATE TABLE public.GUARDA (
    codigo_barras_PRODUCTO bigint NOT NULL,
    id_venta_VENTA varchar(10) NOT NULL,
    cantidad_producto smallint NOT NULL,

```

```

        monto_producto float NOT NULL,
        CONSTRAINT GUARDA_pk PRIMARY KEY (codigo_barras_PRODUCTO,id_venta_VENTA)
);

-- object: public.SURTE
CREATE TABLE public.SURTE (
    rfc_proveedor_PROVEEDOR char(13) NOT NULL,
    codigo_barras_PRODUCTO bigint NOT NULL,
    fecha_surte date NOT NULL,
    CONSTRAINT SURTE_pk PRIMARY KEY (rfc_proveedor_PROVEEDOR,codigo_barras_PRODUCTO)
);

-----Estableciendo las Condiciones
-- object: PRODUCTO_fk
ALTER TABLE public.GUARDA ADD CONSTRAINT PRODUCTO_fk FOREIGN KEY (codigo_barras_PRODUCTO)
REFERENCES public.PRODUCTO (codigo_barras) MATCH FULL
ON DELETE RESTRICT ON UPDATE CASCADE;

-- object: VENTA_fk
ALTER TABLE public.GUARDA ADD CONSTRAINT VENTA_fk FOREIGN KEY (id_venta_VENTA)
REFERENCES public.VENTA (id_venta) MATCH FULL
ON DELETE RESTRICT ON UPDATE CASCADE;

-- object: PROVEEDOR_fk
ALTER TABLE public.SURTE ADD CONSTRAINT PROVEEDOR_fk FOREIGN KEY (rfc_proveedor_PROVEEDOR)
REFERENCES public.PROVEEDOR (rfc_proveedor) MATCH FULL
ON DELETE RESTRICT ON UPDATE CASCADE;

-- object: PRODUCTO_fk
ALTER TABLE public.SURTE ADD CONSTRAINT PRODUCTO_fk FOREIGN KEY (codigo_barras_PRODUCTO)
REFERENCES public.PRODUCTO (codigo_barras) MATCH FULL
ON DELETE RESTRICT ON UPDATE CASCADE;

-- object: CATEGORIA_fk
ALTER TABLE public.PRODUCTO ADD CONSTRAINT CATEGORIA_fk FOREIGN KEY (id_categoria_CATEGORIA)
REFERENCES public.CATEGORIA (id_categoria) MATCH FULL
ON DELETE RESTRICT ON UPDATE CASCADE;

-- object: PROVEEDOR_fk
ALTER TABLE public.telefono ADD CONSTRAINT PROVEEDOR_fk FOREIGN KEY (rfc_proveedor_PROVEEDOR)
REFERENCES public.PROVEEDOR (rfc_proveedor) MATCH FULL
ON DELETE RESTRICT ON UPDATE CASCADE;

-- object: CLIENTE_fk
ALTER TABLE public.VENTA ADD CONSTRAINT CLIENTE_fk FOREIGN KEY (rfc_cliente_CLIENTE)
REFERENCES public.CLIENTE (rfc_cliente) MATCH FULL
ON DELETE RESTRICT ON UPDATE CASCADE;

-- object: SUCURSAL_fk
ALTER TABLE public.VENTA ADD CONSTRAINT SUCURSAL_fk FOREIGN KEY (id_sucursal_SUCURSAL)
REFERENCES public.SUCURSAL (id_sucursal) MATCH FULL
ON DELETE RESTRICT ON UPDATE CASCADE;

----- Realizando las consultas
    requeridas
--Calculando la utilidad de los productos.

```

```

CREATE OR REPLACE FUNCTION calcular_utilidad(p_codigo_barras bigint)
RETURNS TABLE (
    codigo_barras bigint,
    utilidad numeric,
    veces_vendido integer,
    utilidad_total numeric
) AS $$
BEGIN
    RETURN QUERY
    SELECT P.codigo_barras,
           (P.precio_venta - P.precio_compra) AS utilidad,
           COUNT(G.id_venta_VENTA)::integer AS veces_vendido,
           (P.precio_venta - P.precio_compra) * COUNT(G.id_venta_VENTA)::numeric AS utilidad_total
    FROM PRODUCTO P
    JOIN GUARDA G ON P.codigo_barras = G.codigo_barras_PRODUCTO
    WHERE P.codigo_barras = p_codigo_barras
    GROUP BY P.codigo_barras, P.precio_venta, P.precio_compra;
END;
$$ LANGUAGE plpgsql;
--Haciendo la consulta
select *
from calcular_utilidad(14325);

--Actualizacin del stock
CREATE OR REPLACE FUNCTION actualizar_stock() RETURNS TRIGGER AS $$
DECLARE
    codigoIngresado integer;
    codigoAlmacenado integer;
    stockAlmacenado smallint;
BEGIN
    codigoIngresado := NEW.codigo_barras_producto;

    SELECT codigo_barras INTO codigoAlmacenado
    FROM producto
    WHERE codigo_barras = NEW.codigo_barras_producto;

    SELECT stock INTO stockAlmacenado
    FROM producto
    WHERE codigo_barras = NEW.codigo_barras_producto;

    IF NEW.cantidad_producto < stockAlmacenado AND
       NEW.cantidad_producto > 0 THEN
        UPDATE producto
        SET stock = stock - NEW.cantidad_producto;
    ELSE
        RAISE NOTICE 'Nmero de productos no disponible o agotados.';
        RAISE NOTICE 'Intente ingresar una cantidad menor';
    END IF;

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

--Creando un disparador
CREATE TRIGGER actualizar_stock
AFTER INSERT ON guarda

```

```

FOR EACH ROW
EXECUTE FUNCTION actualizar_stock();

--Consulta
CREATE OR REPLACE FUNCTION stock_minimo()
RETURNS SETOF producto AS $$
BEGIN
    RETURN QUERY
    SELECT * FROM producto WHERE stock < 3;
END;
$$ LANGUAGE plpgsql;

--Creacin para insertar un cliente nuevo
CREATE OR REPLACE FUNCTION insertar_cliente(
    rfc character(13),
    name character varying(60),
    ap_paterno character varying(60),
    ap_materno character varying(60),
    calle character varying(70),
    numero smallint,
    estado character varying(60),
    colonia character varying(60),
    cp integer
)

RETURNS VOID AS $$
BEGIN
    INSERT INTO cliente( rfc_cliente, nombre, ap_paterno, ap_materno, calle, numero, estado,
        colonia, cp)
    VALUES (rfc, name, ap_paterno, ap_materno, calle, numero, estado, colonia, cp);
END;
$$ LANGUAGE plpgsql;

--Creacin para realizar una nueva venta
CREATE OR REPLACE FUNCTION ingresar_venta(
    p_id_venta character varying(10),
    p_fecha date,
    p_rfc_cliente character(13),
    p_id_sucursal character varying(10),
    p_codigo_barras bigint,
    p_cantidad_producto smallint,
)
RETURNS void AS $$
BEGIN

    INSERT INTO venta (id_venta, fecha, rfc_cliente_cliente, id_sucursal_sucursal)
    VALUES (p_id_venta, p_fecha, p_rfc_cliente, p_id_sucursal);

    INSERT INTO guarda (codigo_barras_producto, id_venta_venta, cantidad_producto)
    VALUES (p_codigo_barras, p_id_venta, p_cantidad_producto);

    RETURN;
END;
$$ LANGUAGE plpgsql;

```

## 9. Implementación

### 9.1. Creación de la base de datos

Se crea una nueva base de datos llamada "new\_database" utilizando el comando CREATE DATABASE.

### 9.2. Creación de tablas

PRODUCTO: contiene información sobre los productos, como código de barras, precio de venta, precio de compra, fotografía, stock, ID de categoría y nombre.

CATEGORIA: almacena información sobre las categorías de productos, como ID de categoría, marca y descripción.

PROVEEDOR: guarda información sobre los proveedores, incluyendo su RFC, nombre, apellidos y otros datos.

CLIENTE: almacena información sobre los clientes, como su RFC, nombre, apellidos, dirección y código postal.

SUCURSAL: contiene información sobre las sucursales, incluyendo su ID, razón social, dirección y código postal.

VENTA: guarda información sobre las ventas realizadas, como ID de venta, fecha, monto final, RFC del cliente y ID de sucursal.

telefono: almacena información sobre los teléfonos de los proveedores, incluyendo el RFC del proveedor y el número de teléfono.

GUARDA: tabla de relación entre PRODUCTO y VENTA, que registra la cantidad y el monto de cada producto en una venta.

SURTE: tabla de relación entre PROVEEDOR y PRODUCTO, que registra la fecha en que un proveedor surte un producto.

### 9.3. Consultas

Se crea la función calcular\_utilidad que acepta un código de barras y calcula la utilidad de un producto, la cantidad de veces que ha sido vendido y la utilidad total generada. Se realiza una consulta utilizando la función calcular\_utilidad con un código de barras específico (14325). Se crea la función actualizar\_stock que se dispara después de insertar datos en la tabla GUARDA. Esta función actualiza el stock de un producto en base a la cantidad ingresada en la tabla GUARDA. Se crea la función stock\_minimo que devuelve un conjunto de productos cuyo stock es menor que 3. Se crea la función insertar\_cliente para insertar un nuevo cliente en la tabla CLIENTE. Se crea la función ingresar\_venta para registrar una nueva venta en las tablas VENTA y GUARDA.

### 9.4. Triggers

se crea un trigger llamado "actualizar\_stock" que se dispara después de insertar datos en la tabla GUARDA y actualiza el stock de un producto en función de la cantidad ingresada.

### 9.5. Functions

Se definen varias funciones almacenadas, como "calcular\_utilidad" para calcular la utilidad de un producto, "stock\_minimo" para obtener productos con stock mínimo, "insertar\_cliente" para insertar un nuevo cliente y "ingresar\_venta" para registrar una nueva venta.

Las funciones implementadas permiten calcular la cantidad total vendida y la ganancia correspondiente en un período determinado, brindando información útil para el análisis de ventas.

La vista creada simula una factura de compra, proporcionando una representación visual de los datos relacionados con una venta específica.

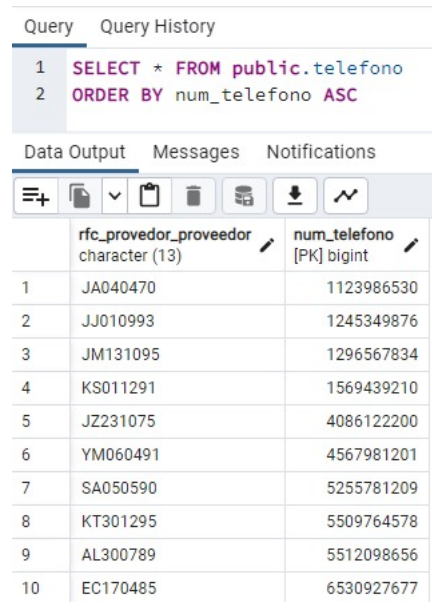
El trigger diseñado se encarga de decrementar automáticamente el stock de los artículos vendidos en el inventario, garantizando que la información esté actualizada y evitando transacciones inválidas.

La secuencia utilizada genera números de venta en un formato predefinido, asegurando la consistencia y facilitando la identificación de las transacciones.

Además, los stored procedures implementados permiten generar la vista de factura, actualizar el stock y generar alertas cuando el stock de un artículo es bajo. Estos procedimientos automatizan tareas recurrentes y mejoran la eficiencia en la gestión de la base de datos.

## 10. Datos de tablas

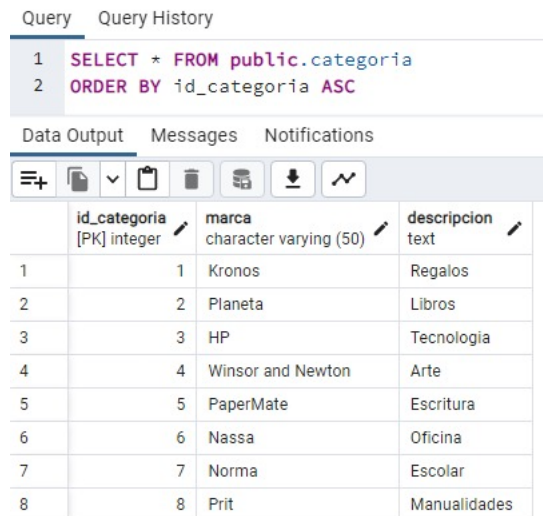
### 10.1. Tabla: Telefono



Query		Query History	
1	SELECT	*	FROM public.telefono
2	ORDER BY	num_telefono	ASC
Data Output			
Messages			
Notifications			
	rfc_proveedor_proveedor	num_telefono	
	character (13)	[PK] bigint	
1	JA040470	1123986530	
2	JJ010993	1245349876	
3	JM131095	1296567834	
4	KS011291	1569439210	
5	JZ231075	4086122200	
6	YM060491	4567981201	
7	SA050590	5255781209	
8	KT301295	5509764578	
9	AL300789	5512098656	
10	EC170485	6530927677	

Figura 13: Telefono

### 10.2. Tabla: Categoria



Query		Query History	
1	SELECT	*	FROM public.categoria
2	ORDER BY	id_categoria	ASC
Data Output			
Messages			
Notifications			
	id_categoria	marca	descripcion
	[PK] integer	character varying (50)	text
1	1	Kronos	Regalos
2	2	Planeta	Libros
3	3	HP	Tecnologia
4	4	Winsor and Newton	Arte
5	5	PaperMate	Escritura
6	6	Nassa	Oficina
7	7	Norma	Escolar
8	8	Prit	Manualidades

Figura 14: Categoria



### 10.3. Tabla: Cliente

Query Query History

```

1 SELECT * FROM public.cliente
2 ORDER BY rfc_cliente ASC

```

Data Output Messages Notifications

	rfc_cliente [PK] character (13)	nombre character varying (60)	ap_paterno character varying (60)	ap_materno character varying (60)	calle character varying (70)	numero smallint	estado character varying (60)	colonia character
1	AQ260402	Alexa	Baulista	Quero	Margaritas	121	Colima	Pozitos
2	AR080302	Alejandra	Rosales	Piña	Doctores	211	CDMX	Condes
3	CJ090785	Carla	Jimenez	Garcia	Copilco	102	CDMX	Roma
4	GR131102	Gamaliel	Rios	Lira	Derecho	98	Estado de México	Las Peñ
5	HP251202	Heber	Garcia	Piñares	Xocongo	105	Yucatán	Polancc
6	JP200402	Jay	Park	[null]	Nicolas Romero	44	Estado de México	Las Piñi
7	JS041295	Jimena	Sanchez	Ortega	Newton	56	CDMX	Roma
8	JW280494	Jackson	Wang	[null]	Torres	278	Quintana Roo	Condes
9	KC011102	Karla	Lopez	Carrasco	Ingenieria	591	Hidalgo	Miguel I
10	LC271102	Leonardo	Chagoya	Gonzalez	Roma	13	CDMX	Roma
11	MS250602	Marco	Sanchez	Hernandez	Odontologia	431	CDMX	Coyoac
12	RS230670	Ramon	Solis	Garcia	Arquitectura	342	CDMX	Rosales
13	RU110899	Rodrigo	Ulises	Sanchez	Rosas	777	Puebla	San Fra
14	SM170302	Saul	Morales	Rosales	Los Angeles	10	Puebla	Los Angeles

Total rows: 15 of 15 Query complete 00:00:00.195 Ln 1, Col 1

Figura 15: Cliente

### 10.4. Tabla: Surte

Query Query History

```

1 SELECT * FROM public.surte
2 ORDER BY rfc_proveedor_proveedor ASC, codigo_barras_producto ASC

```

Data Output Messages Notifications

	rfc_proveedor_proveedor [PK] character (13)	codigo_barras_producto [PK] bigint	fecha_surte date
1	AL300789	10103	2023-01-05
2	EC170485	14325	2023-01-03
3	JA040470	98981	2023-01-03
4	JJ010993	56789	2022-12-28
5	JM131095	98981	2022-12-27
6	JZ231075	23456	2023-01-07
7	KS011291	14325	2022-12-30
8	KT301295	41234	2022-12-28
9	SA050590	76589	2023-01-07
10	YM060491	34564	2022-12-30

Figura 16: Surte

## 10.5. Tabla: Sucursal

Query

Query History

1

SELECT \* FROM public.sucursal

2

ORDER BY id\_sucursal ASC

Data Output

Messages

Notifications

≡

+

📄

▼

📋

🗑

🗄

⬇

📈

	id_sucursal [PK] character varying (10) 	razon_social character varying (100) 	calle character varying (70) 	numero smallint 	estado character varying (60) 	colonia character varying (60) 	cp smallint 
1	131313	123	Flores Magon	121	CDMX	Alvaro Obregon	7700

Figura 17: Sucursal

## 10.6. Tabla: Proveedor

Query

Query History

1

SELECT \* FROM public.proveedor

2

ORDER BY rfc\_proveedor ASC

Data Output

Messages

Notifications

	rfc_proveedor [PK] character (13)	nombre character varying (60)	ap_paterno character varying (60)	ap_materno character varying (60)
1	AL300789	Antonio	Lopez	[null]
2	EC170485	Eduardo	Cruz	Alberto
3	JA040470	Jorge	Nava	Sanchez
4	JJ010993	John	Jun	[null]
5	JM131095	Jimin	Park	[null]
6	JZ231075	Julia	Zurita	[null]
7	KS011291	Kim	Song	[null]
8	KT301295	Tae	Kim	[null]
9	SA050590	Sarahi	Arenas	Arenas
10	YM060491	Yoongi	Min	[null]

Figura 18: Proveedor

## 10.7. Tabla: Venta

Query Query History

```

1 SELECT * FROM public.venta
2 ORDER BY id_venta ASC

```

Data Output Messages Notifications

	id_venta [PK] character varying (10)	fecha date	monto_final double precision	rfc_cliente_cliente character (13)	id_sucursal_sucursal character varying (10)
1	VENT-001	2023-06-05	800	LC271102	131313
2	VENT-002	2023-11-13	243	JP200402	131313
3	VENT-003	2023-04-30	856	AQ260402	131313
4	VENT-004	2023-03-08	40	JW280494	131313
5	VENT-005	2023-01-26	887	HP251202	131313
6	VENT-006	2023-06-08	20	AR080302	131313
7	VENT-007	2023-05-03	189	MS250602	131313
8	VENT-008	2023-02-24	276	KC011102	131313
9	VENT-009	2023-04-21	2100	GR131102	131313
10	VENT-010	2023-05-05	100	UC200497	131313
11	VENT-011	2023-05-06	90	RS230670	131313
12	VENT-012	2023-05-07	50	CJ090785	131313
13	VENT-013	2023-05-08	3849	JS041295	131313
14	VENT-014	2023-05-09	120	SM170389	131313

Total rows: 17 of 17 Query complete 00:00:00.136

Figura 19: Venta

## 10.8. Tabla: Guarda

Query Query History

```

1 SELECT * FROM public.guarda
2 ORDER BY codigo_barras_producto ASC, id_venta_venta ASC

```

Data Output Messages Notifications

	codigo_barras_producto [PK] bigint	id_venta_venta [PK] character varying (10)	cantidad_producto smallint	monto_producto double precision
1	10103	VENT-013	1	3599
2	10103	VENT-015	3	3599
3	14325	VENT-002	2	87
4	14325	VENT-003	1	87
5	14325	VENT-005	2	87
6	14325	VENT-008	1	87
7	23456	VENT-011	1	70
8	23456	VENT-016	10	70
9	34564	VENT-002	1	69
10	34564	VENT-003	1	69
11	34564	VENT-007	1	69
12	34564	VENT-008	1	69
13	41234	VENT-004	2	20
14	41234	VENT-006	1	20

Total rows: 32 of 32 Query complete 00:00:00.124

Figura 20: Guarda

## 10.9. Tabla: Producto

Query Query History

```
1 SELECT * FROM public.producto
2 ORDER BY codigo_barras ASC
```

Data Output Messages Notifications

	codigo_barras [PK] bigint	precio_venta numeric (8)	precio_compra numeric (8)	fotografia bytea	stock smallint	id_categoria_categoria integer	nombre character varying (40)
1	10103	3599	135	[binary data]	2	1	Reloj
2	14325	87	70	[binary data]	6	6	Pegamento en barra
3	23456	70	50	[binary data]	5	2	Lista vocabulario en ingles
4	34564	69	60	[binary data]	7	7	Paquete de dos cuadernos
5	41234	20	10	[binary data]	9	8	Plumas
6	56789	100	80	[binary data]	8	5	Carpetas para documentos
7	76589	50	25	[binary data]	6	3	USB
8	98981	700	550	[binary data]	10	4	Kit de regalo

Figura 21: Producto

## 11. Interfaz gráfica

Se realizó una interfaz gráfica utilizando el framework flask. El sistema se muestra en a través de algún navegador web ya que se trabajó con HTML, CSS, Python, etc. Se adjunta evidencia de la misma.



Figura 22: Página de inicio

En la siguiente imagen se muestra la interfaz para que el usuario pueda registrar un cliente.





Figura 23: Dar clic en la imagen para registrar cliente

En la siguiente imagen se muestra la interfaz para poder registrar una venta.



Figura 24: Dar clic en la imagen para registrar una venta

Una vez dado clic al enlace que se mostró anteriormente, nos lleva a una sección en la cual se muestra un formulario para poder agregar los datos requeridos por el usuario, estos campos cumplen con la restricciones de null o not null dependiendo el atributo.

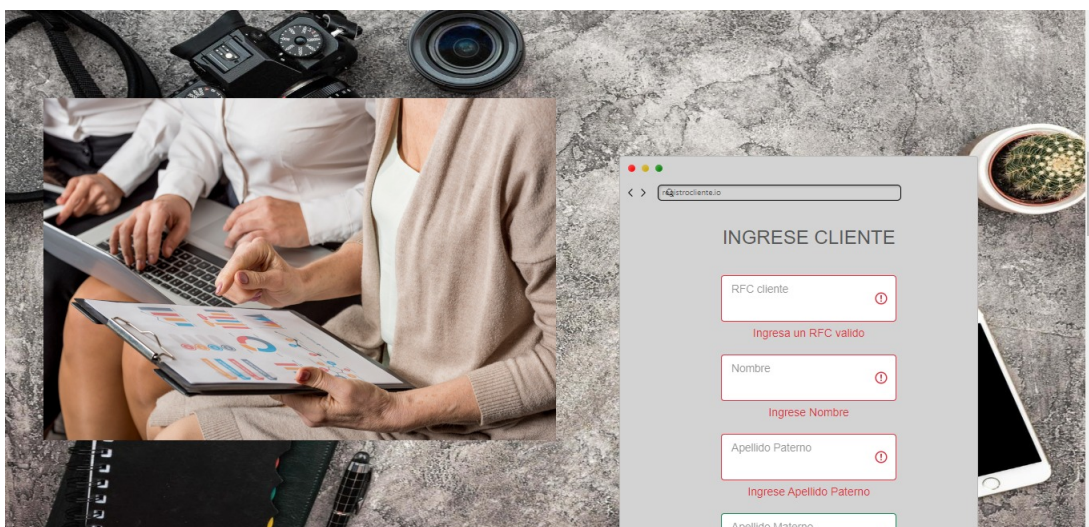


Figura 25: Registrar cliente

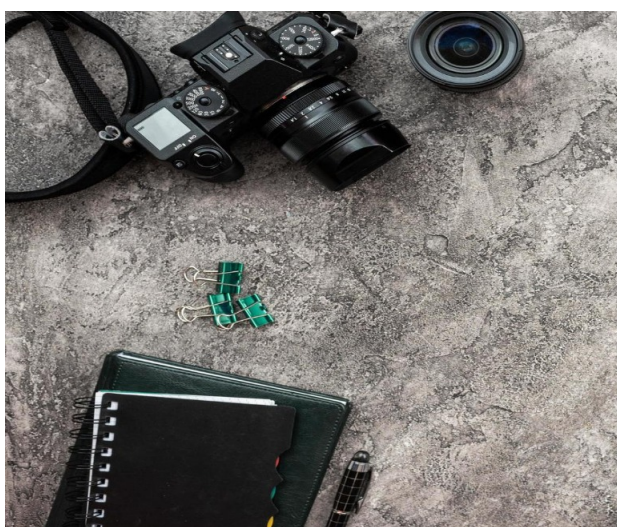
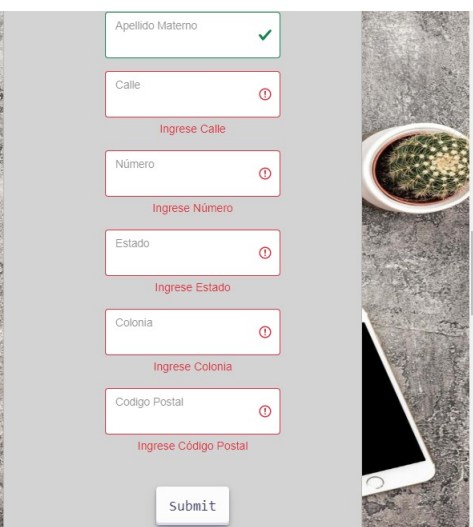



Figura 26: Registrar cliente (continua)

En la siguiente imagen se muestra igual un formulario que nos ayuda a insertar valores de ventas.


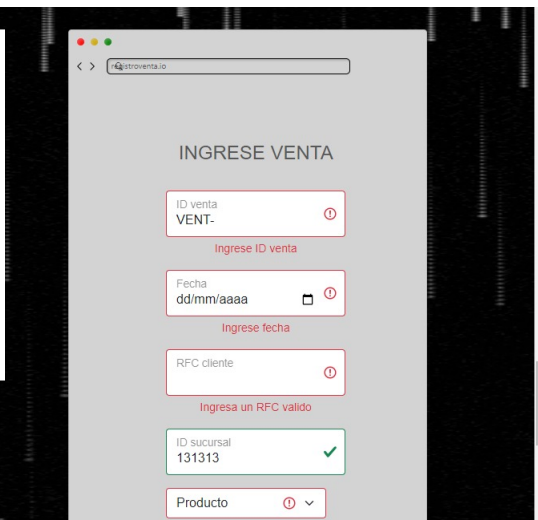



Figura 27: Registrar venta

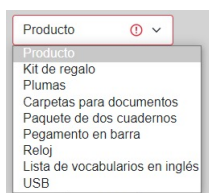


Figura 28: Elegir producto

## **12. Conclusiones**

### **12.1. Carrillo Cervantes Ivette Alejandra**

En este proyecto implementamos todos los conocimientos adquiridos a lo largo del curso, con el fin de crear una base de datos para la administración de una papelería etapa por etapa, desde el análisis de los requerimientos, hasta implementarlo, o bien, conectarlo a una interfaz gráfica, manteniendo la integridad en cada uno de sus datos.

Cada etapa de este proyecto fue un reto para nosotros, pues con forme íbamos avanzando en este, veíamos las etapas anteriores y se nos ocurrían más ideas por haber implementado o nos dábamos cuenta de errores que debíamos de corregir, por lo que debíamos de ir actualizando todos los pasos anteriores además de los actuales. No obstante, nuestra implementación fue cada día mejor y era más eficiente. La parte más complicada de esta implementación fue la parte de programación, pues en lo personal fue mi primer proyecto grande trabajando con bases de datos, por lo que el utilizar TRIGGERS, FUNCIONES, ect. en la base de datos fue un poco complicado, nos tardábamos mucho realizándolas, a veces por causa de el error en el tipo de dato de una entidad, o por que estaba mal la lógica; sin embargo, al ir investigando o al realizar sesiones de llamada entre los integrantes del equipo, se iban resolviendo las dudas.

A pesar de que no se pudo implementar del todo la conexión entre la base de datos y la interfaz gráfica, me agrado mucho el proceso del diseño de un proyecto web para administrar una base de datos.

Este proyecto, a parte de lo teórico y práctico, fue de gran impacto al momento de tomar decisiones a lo largo de su implementación, esto causado a problemas internos dentro del equipo que se lograron solucionar.

### **12.2. Chilpa Navarro Martin Enrique**

La resolución del problema basado en un requerimiento sobre un problema sobre la necesidad de tener una organización en la información de una cadena de papelerías se realizó de forma satisfactoria realizando lo necesario partiendo desde un punto cero, es decir la creación del modelo entidad relación donde se definieron las entidades necesarias hasta la implementación considerando de primera parte al cliente, ofreciendo soluciones a nivel SQL con funciones, vistas y disparadores para poder automatizar lo más posible la solución.

### **12.3. Nava Alberto Vanessa**

Se cumplieron los objetivos del proyecto satisfactoriamente, fue un proyecto retador individualmente y como equipo, pues se necesitó de mucha organización como equipo para poder terminar el proyecto de manera correcta y satisfactoria. En mi opinión fue un proyecto muy completo, pues pudimos aplicar todo lo que se vio en el curso y en lo personal me gustó bastante, pero también me di cuenta que no es tan sencillo como parece, aún así fue un buen reto terminar el proyecto, aunque hubo desacuerdos con un compañero se lograron arreglar. Sin embargo así es como uno se da cuenta de que no siempre se va a ser compatible con todos los compañeros de equipo. Uno de los retos más grandes para mí fue la sección de interfaz, porque jamás en mi vida me habían utilizado una interfaz desde cero, entonces al inicio daba miedo pues no conocía nada, pero aún así sería una experiencia que volvería a aprender.

### **12.4. Juárez Juárez María José**

Este proyecto fue acerca de crear una base de datos para una papelería, utilizamos un modelo entidad-relación y lo convertimos a modelo relacional para posteriormente implementar la base de datos. Durante el desarrollo, enfrentamos desafíos para la creación de consultas eficientes para poder extraer información relevante, como algún producto en específico, inventario o ventas. Además, se emplearon triggers y funciones en PostgreSQL para lograr automatizar algunas acciones y mejorar la manipulación de los datos. La herramienta pgAdmin facilitó el desarrollo al proporcionarnos una interfaz gráfica. En conclusión, el proyecto implicó una planificación, diseño e implementación de la base de datos, donde las consultas eficientes y el uso de triggers y funciones fueron algunos aspectos complejos pero fundamentales para su realización.

### **12.5. Sotelo Ramírez Axel Rodrigo**

La realización de este proyecto representó un reto para todos los integrantes del equipo, ya que tuvimos que idear maneras de implementar soluciones que ayuden a resolver los problemas planteados en la hoja de requerimientos del proyecto. Utilizamos los conocimientos adquiridos en clase para sobrellevar todas las adversidades y supimos adaptarnos para trabajar en equipo.