UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO
Facultad de Ingeniería
Ingeniería en Computación

Bases de Datos
grupo: 01

Proyecto final

Alumnos:

- Fonseca Huitrón Julise Aileen
- López Aniceto Saúl Isaac
- López González Kevin
- Martínez Vázquez Diego
- Ponce Soriano Armando

Profesor:
ING. Fernando Arreola Franco

12 de diciembre de 2021

Índice

1. Introducción

Este proyecto consiste en elaborar el diseño y la creación de una base de datos que utilizará una cadena de papelerías en la que se puede manipular y almacenar la información de todos los productos que esta cadena ofrece. Esta base de datos tendrá como finalidad llevar de manera ordenada y controlada la logística que la papelería sigue para tener control de todos los servicios que ofrece, así como llevar un control de las transacciones y accesos.

Para su correcta manipulación se contempla la creación de una página web, que le permita a los usuarios acceder de manera fácil y rápida a las funciones que se desarrollaron en la base de datos y se pueda ver de una forma gráfica. .

El desarrollo de esta base de datos será detallado en el presente documento con la finalidad de explicar los procedimientos, planes y metodologías que se fueron utilizadas para llevar a cabo este proyecto.

Este proyecto fue elaborado en POSTGRESQL, utilizando los recursos de PHP y AmazonWebServices

2. Plan de trabajo

A continuación, se describe el plan de trabajo utilizado para el desarrollo de este proyecto, principalmente consiste en asignar tareas específicas a todos los participantes, con base en el establecimiento de metas y fechas.

2.1. Descripción

Debido a la magnitud y características del proyecto se decidió utilizar una metodología ágil, que permitiera cumplir con todos los requerimientos del cliente. Por sus cualidades, Scrum fue la mejor alternativa, ya que, nos permitió:

- Definir el Product Backlog: ordenar de mayor a menor importancia las funcionalidades pedidas por el cliente.
- Desarrollar la lista de tareas de la iteración o Sprint Backlog.
- Realizar Sprint Planning Meeting a lo largo del Sprint Backlog con el equipo de trabajo para determinar el enfoque del proyecto, las etapas y los plazos.
- Durante todo el periodo de Sprint, realizar reuniones con el equipo encargado para conocer los avances, las tareas por terminar y las necesidades para terminar dichas tareas.
- Al concluir con los Sprint, realizar el Sprint Review, para revisar todos los avances del proyecto

2.2. Plan de actividades

En el siguiente cronograma se podrá consultar los Sprint estableciendo para la creación y desarrollo del sistema, estableciendo fechas y un límite de tiempo para cada tarea.

En la parte de asignaciones, se destina cada uno de los integrantes una tarea referida en el cronograma y que forma parte de la elaboración de este proyecto.

2.3. Cronograma

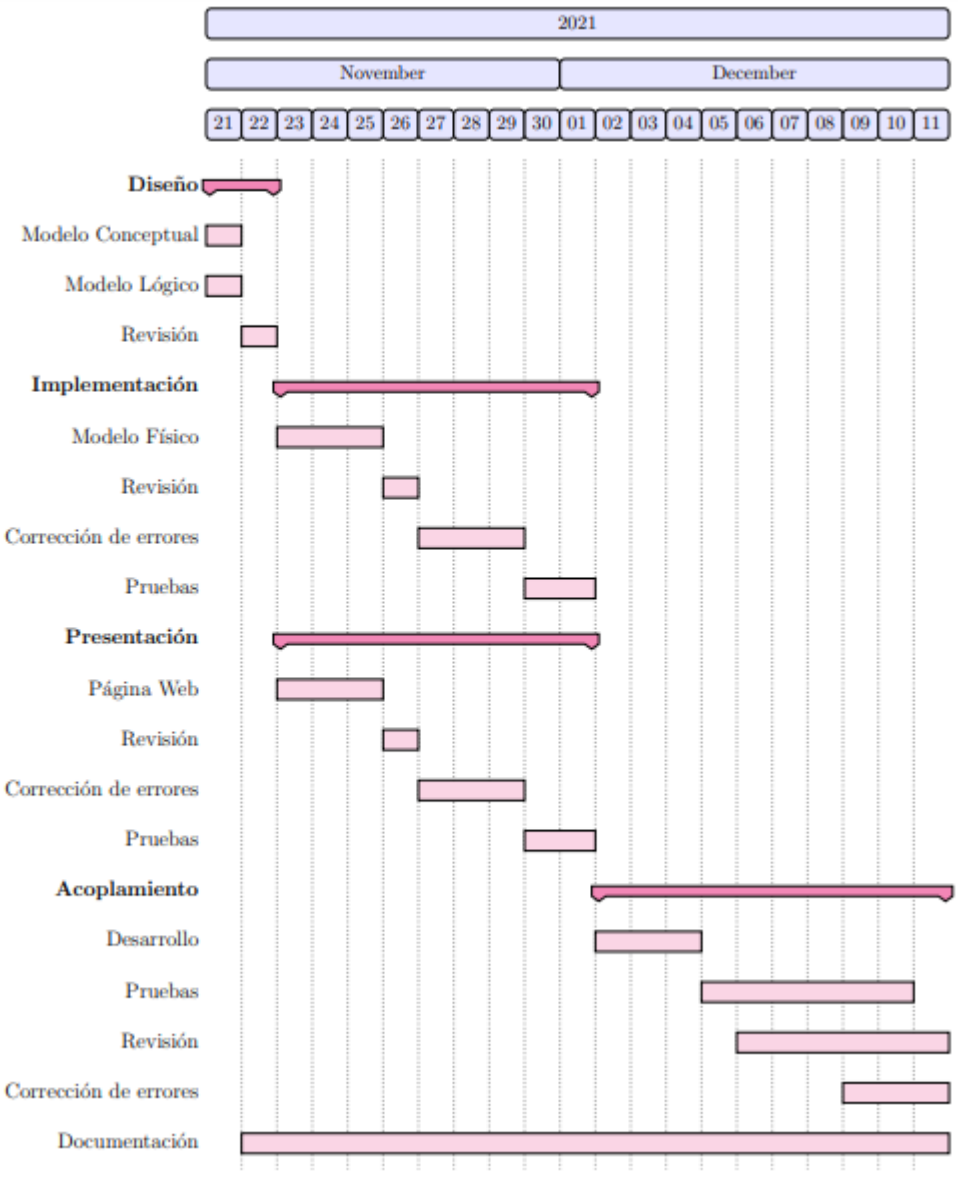


Figura 1: Cronograma

2.4. Aportaciones

	Diseño	Implementación	Presentación	Acoplamiento	Documentación
Fonseca Huitrón Julise	✓	✓			✓
López Aniceto Isaac	✓	✓			
López González Kevin	✓		✓	✓	✓
Martínez Vázquez Diego	✓	✓			✓
Ponce Soriano Armando	✓		✓	✓	

3. Diseño

3.1. Análisis de requerimientos

Para la elaboración de la base de datos se contempla que una papelería tendrá una serie de registros que a su vez contendrán información, se requiere almacenar información de los proveedores, se requiere almacenar información sobre un Inventario que almacena también información. Se deberá guardar información detallada de cada venta realizada.

Requerimos también toda la información relacionada con el cliente y también información específica del o los tipos de producto que tenemos a la venta.

Los requerimientos antes mencionados nos llevan a conceptualizar más las ideas, encontrando que hay dependencias en algunos de ellos, nos lleva a analizar la cardinalidad que hay entre las tablas.

El análisis de estos requerimientos es la base para la construcción de nuestros modelo Entidad-Relación y en general de todo el modelo Conceptual.

3.2. Modelo conceptual

Entidades

- PROVEEDOR: { id_Proveedor, razón social, domicilio (estado, código postal, colonia, calle y número), nombre, teléfonos }
- CLIENTE: {RFC, nomre (nombre, ap_Paterno, ap_Materno), domicilio (estado, código postal, colonia, calle y número), emails }
- INVENTARIO: {id_Inventario, precio_compra, fecha_compra, cantidad_ejemplares }
- PRODUCTO: {código_Barras, marca, descripción, precio, categoria}
- VENTA : {num_venta, fecha_venta, pago_Total, cantidad_articulo, pago_total_Articulo }

Relaciones

- Un proveedor surte a muchos inventarios.

- Un inventario es surtido por muchos proveedores.

- Un inventario almacena muchos productos.
- Un producto es almacenado por un inventario.

- Una venta contiene muchos productos.
- Un producto es contenido es muchas ventas.

- Un cliente concreta muchas ventas.
- Una venta es concretada por un cliente.

3.2.1. Modelo Entidad-Relación

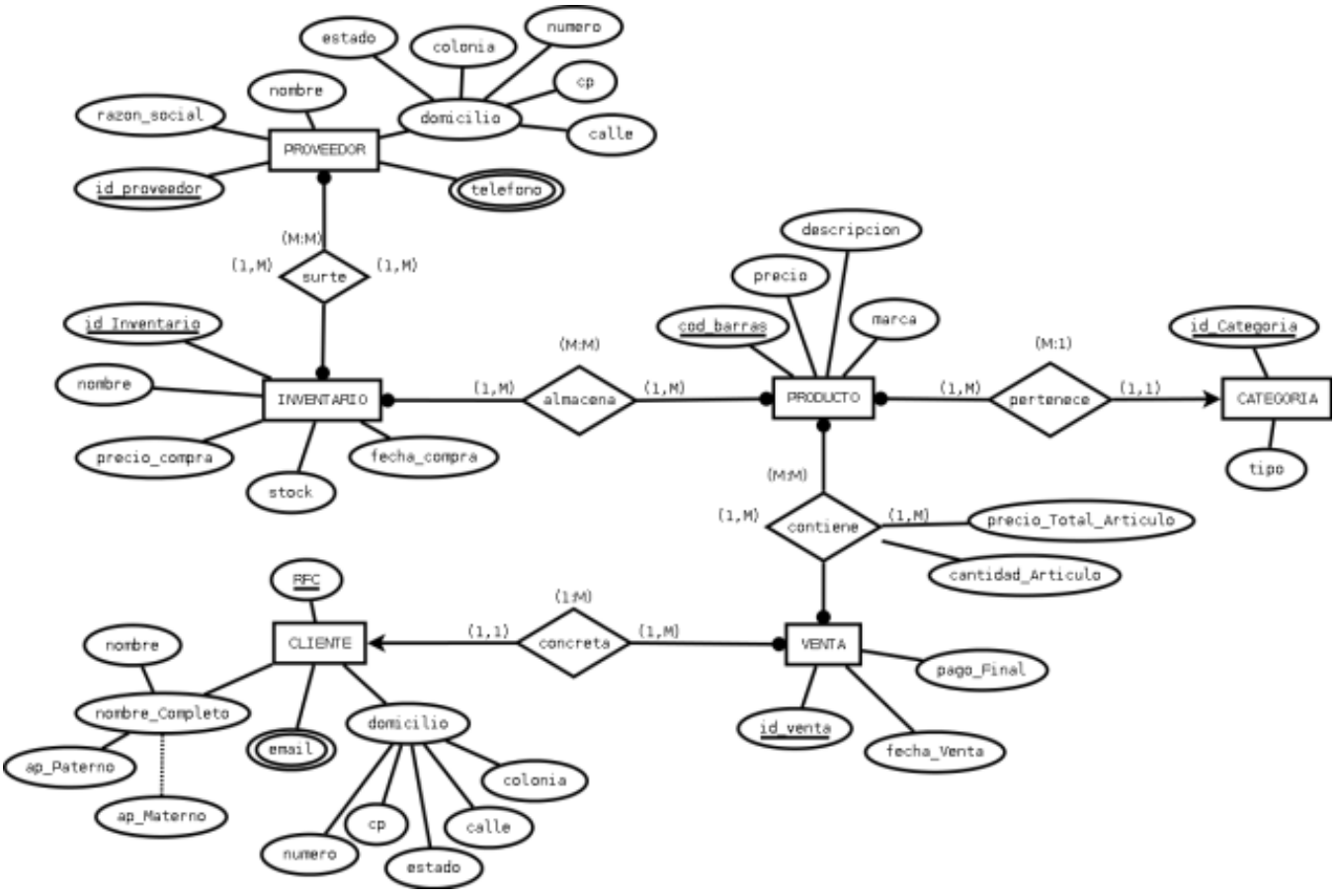


Figura 2: Modelo Entidad-Relación.

3.3. Modelo lógico

3.3.1. Representación Intermedia

Realizando el Modelo Entidad-Relación, pudimos seguir las reglas para de la transformación de MER a MR, nos derivó en las siguientes tablas:

- PROVEEDOR: { id_proveedor smallint (PK), nombre varchar 50, razón social varchar 50, estado varchar 50, colonia varchar 50, numero smallint, cp smallint, calle varchar 50}
- TELEFONO: {teléfono bigint(PK), id_proveedor smallint (FK)}
- INVENTARIO: {id_Inventario smallint (PK), nombre varchar(50)}
- SURTE: {[id_Provedor smaillint (FK), id_Inventario smallint (FK)] (PK)}
- PRODUCTO: {cod_barras integer PK, id_categoria smallint FK, precio smallint NOT NULL, marca varchar(20) NOT NULL, descripcion varchar(50)}
- CATEGORÍA: { id_categoria smallint PK, tipo varchar(20) NOT NULL}
- GUARDA: {[id_Inventario smallint (FK), cod_Barras int(FK)] (PK), dateprecio_compra decimal (10,2), stock smallint, fecha_compra date}
- CLIENTE: {RFC varchar(13) (PK), nombre varchar(20), ap_paterno varchar (20), ap_materno varchar (20) (N), cp smallint, numero smallint, estado varchar (32), calle varchar (32), colonia varchar (32)}
- EMAIL: {RFC varchar(13) (FK), email varchar (64) (PK)}
- VENTA: {id_venta int(PK), fecha_venta date, pago_final decimal(7,2), RFC varchar(13)(FK)}
- CONTIENE: { [cod_barras int , id_venta int](PK)(FK), precioTotalArt decimal(7,2), cantidad articulo int}

3.3.2. Transformación de MER a MR

Para la transformación de Modelo Entidad Relación a Modelo Relacional, primero debemos utilizar la transformación intermedia que obtuvimos y desarrollamos. Una vez finalizado ese procedimiento utilizamos las reglas para la trasformación de MER a MR y obtendremos nuestro modelo relacional.

3.3.3. Modelo Relacional

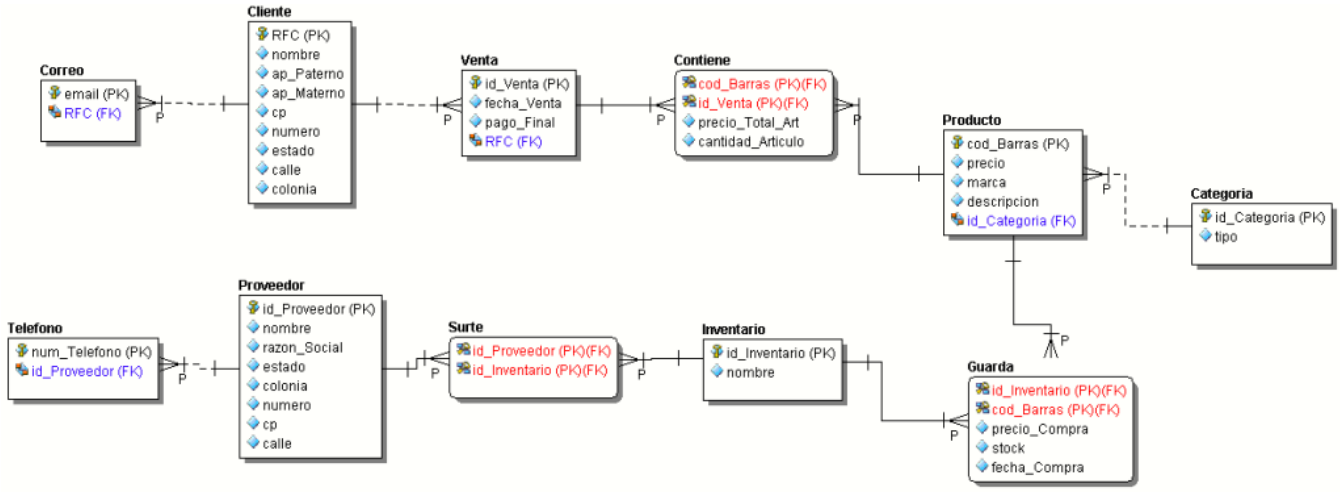


Figura 3: Modelo Relacional.

3.3.4. Normalización

Se cumple con la 1FN, ya que las tablas no presentan grupos de repetición y cada columna contiene valores atómicos.

Correo:cumple la 2FN debido a que PK es simple.
email → RFC

Cliente: cumple la 2FN debido a que PK es simple.
RFC → nombre, ap_Paterno, ap_Materno, cp, estado, calle, colonia

Venta: cumple la 2FN debido a que PK es simple
id_Venta →fecha_Venta, pago_Final, RFC

Contiene: No existen dependencias funcionales parciales
cod_Barras, id_Venta → precio_Total,cantidad_Articulo

Producto: cumple la 2FN debido a que PK es simple.
Cod_Barras → precio, marca, descripción, id_Categoia

Categoría: cumple la 2FN debido a que PK es simple.
id_Categoria → tipo

Inventario: cumple la 2FN debido a que PK es simple.
id_Inventario → nombre

Guarda: No existen dependencias funcionales parciales
id_Venta, cod_Barras → precio_Compra, stock, fecha_Compra

Proveedor: cumple la 2FN debido a que PK es simple.
Id_Proveedor: nombre, razón_Social, estado, colonia, numero, cp, cale

Teléfono:
num_Telefono →id_Proovedor

Se cumple con la 3FN, ya que se cumple con la 2FN y no hay transitividad en las tablas

4. Implementación

Para llevar a cabo la implementación de esta base de datos, primero tendremos que crear la base de datos en el manejador POSTGRESQL.

Una vez finalizada la creación de la base de datos, tendremos que crear las tablas que obtuvimos en nuestro modelo relacional. Estas tablas tendrán que tener sus respectivos constraints para las llevas primarias y foráneas que se utilizan en toda la base de datos.

Cuando las tablas estén creadas en las base de datos, podremos verificar su utilidad haciendo inserciones de prueba.

Para que el sistema resuelva cuando se reciba un código de barras regrese la utilidad, para que cuando haya la venta de un artículo y se decremente el stock por la cantidad vendida, dada una fecha o una fecha de inicio y de fin, regrese la cantidad total que se vendió y su ganancia correspondiente. Para obtener el nombre de aquellos productos que están es stock, que de forma automática genere vista que contenga información para desplegar una factura de compra y la creación del índice. Para todas esas funciones mencionadas, tendremos que hacer uso de la creación de Functions, Procedures y Triggers.

4.1. Modelo físico

4.1.1. IaaS

La infraestructura como servicio (IaaS) es un método que ofrece la funcionalidad de almacenar información a través de Internet.

Amazon Web Services (AWS) es la plataforma en la nube más adoptada y completa, permitiendo reducir costos, aumentar su agilidad e innovar de forma más rápida, además de su disponibilidad las 24 horas del día.

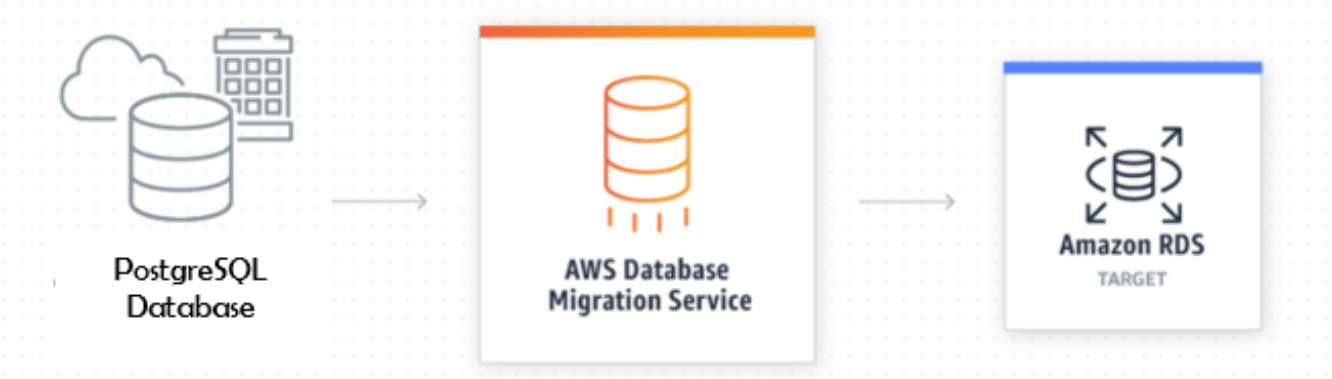


Figura 4: IaaS.

Debido a diferentes circunstancias sucitadas durante el desarrollo del proyecto, se tuvo que optar por el uso de una base de datos de forma local, aunque es importante aclararar que se lograron conexiones exitosas con una base de datos en la nube mediante estos servicios.

4.2. DDL

El lenguaje de definición de datos (Data Definition Language), permite definir las estructuras que almacenarán los datos, así como los procedimientos o funciones que permitan consultarlos.

La tabla cliente, permite guardar todos sus datos; debido a que un RFC nunca se repite se usara para comunicarse con las tablas Correo y Venta. En la siguiente tabla

```
CREATE TABLE Cliente(
    RFC             VARCHAR(13)     NOT NULL,
    nombre          VARCHAR(32)     NOT NULL,
    ap_Paterno      VARCHAR(32)     NOT NULL,
    ap_Materno      VARCHAR(32),
    cp              SMALLINT        NOT NULL,
    numero          SMALLINT        NOT NULL,
    estado          VARCHAR(32)     NOT NULL,
    calle           VARCHAR(32)     NOT NULL,
    colonia         VARCHAR(32)     NOT NULL,
    CONSTRAINT PK2 PRIMARY KEY (RFC)
);
```

Figura 5: Tabla Cliente.

Con la implementación de la tabla Correo el cliente podrá proporcionar al menos un correo.

```
CREATE TABLE Correo(
    email    VARCHAR(64)    NOT NULL,
    RFC      VARCHAR(13)    NOT NULL,
    CONSTRAINT PK1 PRIMARY KEY (email)
);
```

Figura 6: Tabla Correo.

La tabla Venta contiene los datos más generales de dicha acción.

```
CREATE TABLE Venta(
    id_Venta       INTEGER          NOT NULL,
    fecha_Venta    DATE             NOT NULL,
    pago_Final     DECIMAL(7, 2)    NOT NULL,
    RFC            VARCHAR(13)      NOT NULL,
    CONSTRAINT PK3 PRIMARY KEY (id_Venta)
);
```

Figura 7: Tabla Venta.

La siguiente tabla tiene los datos generales del producto vendido, es por eso que se relaciona con la tabla anterior y posterior.

```
CREATE TABLE Venta(
    id_Venta       INTEGER          NOT NULL,
    fecha_Venta    DATE             NOT NULL,
    pago_Final     DECIMAL(7, 2)    NOT NULL,
    RFC            VARCHAR(13)      NOT NULL,
    CONSTRAINT PK3 PRIMARY KEY (id_Venta)
);
```

Figura 8: Tabla Venta.

En la tabla Producto se guardan los datos particulares de cada uno de estos.

```
CREATE TABLE Producto(
    cod_Barras      INTEGER          NOT NULL,
    precio          DECIMAL(7, 2)    NOT NULL,
    marca           VARCHAR(120)     NOT NULL,
    descripcion     VARCHAR(50)      NOT NULL,
    id_Categoria    SMALLINT         NOT NULL,
    CONSTRAINT PK5 PRIMARY KEY (cod_Barras)
);
```

Figura 9: Tabla Venta.

Debido a que los productos son diversos, los seccionamos en cuatro categorías (regalos, papelería, impresiones y recargas), es por ello que establecemos un id para identificar a cada una.

```
CREATE TABLE Categoria(
    id_Categoria    SMALLINT        NOT NULL,
    tipo            VARCHAR(20)     NOT NULL,
    CONSTRAINT PK10 PRIMARY KEY (id_Categoria)
);
```

Figura 10: Tabla Categoría.

La tabla que se muestra a continuación permitirá tener los detalles particulares de la venta.

```
CREATE TABLE Guarda(
    id_Inventario    SMALLINT         NOT NULL,
    cod_Barras       INTEGER          NOT NULL,
    precio_Compra    DECIMAL(7, 2)    NOT NULL,
    stock            INTEGER          NOT NULL,
    fecha_Compra     DATE             NOT NULL,
    CONSTRAINT PK12 PRIMARY KEY (id_Inventario, cod_Barras)
);
```

Figura 11: Tabla Guarda.

Con el inventario se podrá tener acceso y control dela información referente a la venta.

```
CREATE TABLE Inventario(
    id_Inventario    SMALLINT        NOT NULL,
    nombre           VARCHAR(32),
    CONSTRAINT PK9 PRIMARY KEY (id_Inventario)
);
```

Figura 12: Tabla Inventario.

La tabla Surte permitirá comunicar al inventario con el proveedor.

```
CREATE TABLE Surte(
    id_Proveedor     SMALLINT    NOT NULL,
    id_Inventario    SMALLINT    NOT NULL,
    CONSTRAINT PK8 PRIMARY KEY (id_Proveedor, id_Inventario)
);
```

Figura 13: Tabla Surte.

Finalmente, esta tabla podrá tener el número telefónico de varios proveedores

```
CREATE TABLE Telefono(
    num_Telefono    BIGINT      NOT NULL,
    id_Proveedor    SMALLINT    NOT NULL,
    CONSTRAINT PK6 PRIMARY KEY (num_Telefono)
);
```

Figura 14: Tabla Telefono.

En el siguiente apartado, se podrán consultar las modificaciones realizadas a las tablas para su correcta conexión, usando las llaves primarias y foráneas, de acuerdo a lo establecido en el MR.

```
ALTER TABLE Correo ADD CONSTRAINT RefCliente2
    FOREIGN KEY (RFC)
    REFERENCES Cliente(RFC)
;

ALTER TABLE Venta ADD CONSTRAINT RefCliente3
    FOREIGN KEY (RFC)
    REFERENCES Cliente(RFC)
;

ALTER TABLE Contiene ADD CONSTRAINT RefVenta4
    FOREIGN KEY (id_Venta)
    REFERENCES Venta(id_Venta)
;

ALTER TABLE Contiene ADD CONSTRAINT RefProducto5
    FOREIGN KEY (cod_Barras)
    REFERENCES Producto(cod_Barras)
;

ALTER TABLE Producto ADD CONSTRAINT RefCategoria10
    FOREIGN KEY (id_Categoria)
    REFERENCES Categoria(id_Categoria)
;

ALTER TABLE Guarda ADD CONSTRAINT RefInventario11
    FOREIGN KEY (id_Inventario)
    REFERENCES Inventario(id_Inventario)
;

ALTER TABLE Guarda ADD CONSTRAINT RefProducto12
    FOREIGN KEY (cod_Barras)
    REFERENCES Producto(cod_Barras)
;

ALTER TABLE Surte ADD CONSTRAINT RefProveedor6
    FOREIGN KEY (id_Proveedor)
    REFERENCES Proveedor(id_Proveedor)
;

ALTER TABLE Surte ADD CONSTRAINT RefInventario7
    FOREIGN KEY (id_Inventario)
    REFERENCES Inventario(id_Inventario)
;

ALTER TABLE Telefono ADD CONSTRAINT RefProveedor8
    FOREIGN KEY (id_Proveedor)
    REFERENCES Proveedor(id_Proveedor)
;
```

Figura 15: Modificacion de las tablas.

En la siguiente sección se presenta una secuencia para incrementar de forma automática el numero cada vez que se realice una venta.

```
CREATE SEQUENCE Vent
START WITH 1
INCREMENT BY 1;
```

```
ALTER TABLE Venta ALTER id_Venta SET DEFAULT NEXTVAL('Vent');
```

Figura 16: Aumento del numero de venta.

La siguiente modificación implica que se devuelva la fecha actual

```
ALTER TABLE Venta ALTER fecha_Venta SET DEFAULT current_date;
```

Figura 17: Modificacion de la tabla Venta.

4.3. DML

Los datos que a continuación se presentan son los mínimos que se deben establecer para para posteriormente realizar tareas de consultas o modificación de dichos datos contenidos en la Base de Dato.

```
INSERT INTO PROVEEDOR VALUES (1,'Telcel','RADIOMOVIL DIPSA S.A. DE CV','Ciudad de Mexico','Col.Apliacion Granada',245,06500,'Lago
INSERT INTO PROVEEDOR VALUES (2,'Movistar','Telefónica Móviles México, S.A. de C.V.','Ciudad de Mexico','Col.Apliacion Granada',24
INSERT INTO PROVEEDOR VALUES (3,'Mapeed','MAPEED S.A. DE CV','Ciudad de Mexico','Col.Benito Juarez',233, 06700,'Fuentes');
INSERT INTO PROVEEDOR VALUES (4,'Bic','Industrial de Cuautitlán S.A. de C.V.','Jalisco','Col. Puerta de Hierro',240,04511,'Real');
INSERT INTO PROVEEDOR VALUES (5,'Epson','EPSON MEXICO- S.A. DE C.V.','Ciudad de Mexico','Col. Irrigacion',389,11500,'Manuel Avila
INSERT INTO PROVEEDOR VALUES (6,'Canon','CANON MEXICANA S. DE R.L. DE C.V.','Ciudad de Mexico','Col. Lomas de Chapultepec I Seccio
INSERT INTO PROVEEDOR VALUES (7,'Mattel','Mattel De Mexico S.a. De C.v.','Ciudad de Mexico','Col. Granada',193,11520,'Miguel de Ce
INSERT INTO PROVEEDOR VALUES (8,'Ferrero','Ferrero de México, S.A. de C.V.','Jalisco','Col. Jardines del Bosque',251,04452,'Marian

INSERT INTO TELEFONO VALUES (5556317219,1);
INSERT INTO TELEFONO VALUES (5526524662,2);
INSERT INTO TELEFONO VALUES (5573620400,3);
INSERT INTO TELEFONO VALUES (3350895900,4);
INSERT INTO TELEFONO VALUES (5557649090,5);
INSERT INTO TELEFONO VALUES (5552494905,6);
INSERT INTO TELEFONO VALUES (5529452792,7);
INSERT INTO TELEFONO VALUES (3331213242,8);
INSERT INTO PRODUCTO VALUES (53201,5.00,'Epson','Impresion a color tamaño carta',1);
INSERT INTO PRODUCTO VALUES (53202,1.00,'Epson','Impresion B/N tamaño carta',1);
INSERT INTO PRODUCTO VALUES (53203,10.00,'Canon','Impresion a color tamaño oficio',1);
INSERT INTO PRODUCTO VALUES (53204,3.00,'Canon','Impresion B/N tamaño oficio',1);
INSERT INTO PRODUCTO VALUES (53205,50.00,'Epson','Impresion OFFSET',1);
INSERT INTO PRODUCTO VALUES (53206,5.00,'Bic','Lapiz',2);
INSERT INTO PRODUCTO VALUES (53207,10.00,'Bic','Pluma',2);
INSERT INTO PRODUCTO VALUES (53208,5.00,'Mapeed','Goma',2);
INSERT INTO PRODUCTO VALUES (53209,5.00,'Mapeed','Sacapuntas',2);
INSERT INTO PRODUCTO VALUES (53210,20.00,'Mapeed','Colores',2);
INSERT INTO PRODUCTO VALUES (53211,49.99,'Mattel','Max Steel',3);
INSERT INTO PRODUCTO VALUES (53212,49.99,'Mattel','Barbie',3);
INSERT INTO PRODUCTO VALUES (53213,249.99,'Mattel','Triciclo',3);
INSERT INTO PRODUCTO VALUES (53214,199.99,'Ferrero','Ferrero Rocher grande',3);
INSERT INTO PRODUCTO VALUES (53215,99.99,'Ferrero','Nutella',3);
INSERT INTO PRODUCTO VALUES (53216,20.00,'Telcel','Recarga pequeña',4);
INSERT INTO PRODUCTO VALUES (53217,50.00,'Telcel','Recarga mediana',4);
INSERT INTO PRODUCTO VALUES (53218,100.00,'Telcel','Recarga grande',4);
INSERT INTO PRODUCTO VALUES (53219,50.00,'Movistar','Plan pequeño',4);
INSERT INTO PRODUCTO VALUES (53220,100.00,'Movistar','Plan grande',4);
INSERT INTO INVENTARIO VALUES (1,'Unidad Centro');
```

```
INSERT INTO GUARDA VALUES (1,53201,1.00,100,'2021-06-20');
INSERT INTO GUARDA VALUES (1,53202,0.25,100,'2021-06-20');
INSERT INTO GUARDA VALUES (1,53203,3.00,100,'2021-06-20');
INSERT INTO GUARDA VALUES (1,53204,1.00,100,'2021-06-20');
INSERT INTO GUARDA VALUES (1,53205,15.00,50,'2021-07-01');
INSERT INTO GUARDA VALUES (1,53206,1.00,50,'2021-04-15');
INSERT INTO GUARDA VALUES (1,53207,3.00,50,'2021-05-15');
INSERT INTO GUARDA VALUES (1,53208,2.00,25,'2021-04-15');
INSERT INTO GUARDA VALUES (1,53209,2.00,25,'2021-04-15');
INSERT INTO GUARDA VALUES (1,53210,5.00,10,'2021-04-01');
INSERT INTO GUARDA VALUES (1,53211,15.00,15,'2021-12-01');
INSERT INTO GUARDA VALUES (1,53212,15.00,15,'2021-12-01');
INSERT INTO GUARDA VALUES (1,53213,50.00,10,'2021-02-01');
INSERT INTO GUARDA VALUES (1,53214,40.00,20,'2021-02-01');
INSERT INTO GUARDA VALUES (1,53215,35.00,20,'2021-02-01');
INSERT INTO GUARDA VALUES (1,53216,5.00,50,'2021-11-09');
INSERT INTO GUARDA VALUES (1,53217,20.00,50,'2021-11-09');
INSERT INTO GUARDA VALUES (1,53218,35.00,25,'2021-11-09');
INSERT INTO GUARDA VALUES (1,53219,20.00,10,'2021-10-30');
INSERT INTO GUARDA VALUES (1,53220,35.00,10,'2021-10-30');
INSERT INTO SURTE VALUES (1,1);
INSERT INTO SURTE VALUES (2,1);
INSERT INTO SURTE VALUES (3,1);
INSERT INTO SURTE VALUES (4,1);
INSERT INTO SURTE VALUES (5,1);
INSERT INTO SURTE VALUES (6,1);
INSERT INTO SURTE VALUES (7,1);
INSERT INTO SURTE VALUES (8,1);
```

Figura 18: Datos en la BD.

4.4. Funciones

Para simular una factura, utilizamos la declaración CREATE VIEW, junto con el proceso Join para seleccionar los datos de las columnas que se intersectan de las diferentes tablas (Cliente, Venta, Contiene, Producto) y así generar la vista.

```
CREATE VIEW FACTURA
AS
SELECT Cliente.RFC, Cliente.nombre, Cliente.ap_Paterno, Cliente.ap_Materno, Cliente.cp,
Cliente.calle, Cliente.colonia, Venta.id_Venta, Venta.fecha_Venta, Venta.pago_Final,
Contiene.precio_Total_Art, Contiene.cantidad_Articulo, Producto.marca, Producto.descripcion FROM CLIENTE
INNER JOIN VENTA ON CLIENTE.RFC=VENTA.RFC INNER JOIN CONTIENE ON VENTA.id_Venta=CONTIENE.id_Venta
INNER JOIN PRODUCTO ON CONTIENE.cod_Barras=PRODUCTO.cod_Barras;
```

Figura 19: Vista Factura.

La siguiente función regresa una tabla, en la cual se muestre el número de productos en el inventario, la marca y su respectiva descripción, siempre y cuando el stock sea menor o igual a 3.

```
CREATE OR REPLACE FUNCTION menos_Thr() RETURNS TABLE (stock integer, marca varchar, descripcion varchar)
AS $$
DECLARE
v_stock integer;
nom_Marca varchar(120);
BEGIN
RETURN QUERY EXECUTE
'SELECT stock, marca, descripcion FROM guarda LEFT JOIN producto
ON guarda.cod_Barras=producto.cod_Barras
WHERE stock <= 3';
END;
$$
LANGUAGE plpgsql;
```

Figura 20: Funcion_menos_Thr.png

Con id_Venta_Funcion() se puede consultar el ultimo id de venta generado

```
CREATE OR REPLACE FUNCTION id_Venta_Funcion() RETURNS integer
AS
$$
DECLARE
V_id integer;
BEGIN
SELECT last_value INTO V_id FROM VENT;
RETURN V_id;
end;
$$
LANGUAGE plpgsql;
```

Figura 21: id_Venta_Funcion.

La siguiente función regresa la utilidad de cada producto, por medio de la intersección de las tablas Producto y Guarda, a partir de su código de barras.

```
CREATE OR REPLACE FUNCTION retorna_Utilidad(codigo INT)RETURNS DECIMAL(7,2) AS
$$
DECLARE
v_Cod_Barras INT;
v_Utilidad DECIMAL(7,2);
BEGIN
SELECT Guarda.cod_Barras, SUM(precio-precio_Compra) INTO v_Cod_Barras, v_Utilidad FROM Guarda
INNER JOIN Producto
ON Producto.cod_Barras = Guarda.cod_Barras
WHERE Guarda.cod_Barras = codigo
GROUP BY Guarda.cod_Barras;
RETURN v_Utilidad;
END;
$$
LANGUAGE plpgsql;
```

Figura 22: retorna_Utilidad.

Debido a las características de esta base de datos, se realizaron dos funciones distintas bajo el mismo nombre, con la finalidad de que, si una no cumplía con los parámetros proporcionados, el manejador seleccionara directamente la otra opción.

A partir de la fecha de venta, la función, suma el valor de cada venta para conocer la cantidad de dinero que se obtuvo ese día.

```
CREATE OR REPLACE FUNCTION retorna_Pago_Final(fecha DATE)RETURNS DECIMAL(7,2) AS
$$
DECLARE
v_Pago DECIMAL(7,2);
v_Fecha DATE;
BEGIN
SELECT fecha_Venta, SUM(pago_Final) INTO v_Fecha, v_Pago FROM Venta
WHERE fecha_Venta = fecha
GROUP BY fecha_Venta;
RETURN v_Pago;
END;
$$
LANGUAGE plpgsql;
```

Figura 23: retorna_Pago_Final.

A diferencia de la función anterior, en esta se deben proporcionar dos parámetros, el primero la fecha de inicio y el segundo la fecha de fin y así obtener la cantidad de dinero vendida en el periodo de tiempo señalado; para esto se suma la cantidad vendida de cada producto y se ocupa la palabra clave BETWEEN que permite establecer el rango de valores en donde se quiere realizar la consulta.

```
CREATE OR REPLACE FUNCTION retorna_Pago_Final(fecha_inicio DATE, fecha_fin DATE)RETURNS DECIMAL(7,2) AS
$$
DECLARE
v_Pago DECIMAL(7,2);
BEGIN
SELECT SUM(pago_Final) INTO v_Pago FROM Venta
WHERE fecha_Venta BETWEEN fecha_inicio AND fecha_fin;
RETURN v_Pago;
END;
$$
LANGUAGE plpgsql;
```

Figura 24: retorna_Pago_Final.

4.5. Trigger

El trigger o disparador se ejecuta en la función venta cuando se realice la instrucción de insertar sobre la tabla Contiene.

```
CREATE TRIGGER venta_trigger INSTEAD OF INSERT
ON CONTIENE FOR EACH ROW
EXECUTE PROCEDURE venta();
```

Figura 25: Creacion de Trigger

La función permite decrementar el stock cada que se realice una venta y debido a su estructura, si la cantidad de productos es mayor a la que se tiene en el inventario, esta lanzara una alerta donde se mencione que no hay suficientes productos y se cancelara la transacción,, si la venta se efectúa, se actualizara la actualización a la tabla Contiene para decrementar el stock del inventario, por otra parte si solo quedan 3 o menos de tres productos en el inventario, se lanzara otra alerta para prevenir que ya casi no quedan productos.

```
CREATE OR REPLACE FUNCTION venta() RETURNS TRIGGER AS $existe$
DECLARE decremento integer;
BEGIN
decremento= new.cantidad_Articulo;
IF (select stock FROM guarda WHERE cod_Barras = NEW.cod_Barras) < decremento THEN
raise notice 'No hay suficiente producto';
ROLLBACK;
RETURN stock;
END IF;
IF (select stock FROM guarda where cod_Barras = NEW.cod_Barras ) >= decremento THEN
UPDATE guarda SET stock= ((SELECT stock FROM guarda WHERE cod_Barras=NEW.cod_Barras) - decremento) WHERE cod_Barras = NEW.cod_Barras;
IF (select stock FROM guarda where  cod_Barras = NEW.cod_Barras ) <= 3 THEN
raise notice 'stock actualizado, REVISE SU INVENTARIO';
END IF;
RETURN NULL;
END IF;
RETURN NULL;
END;
$existe$ LANGUAGE plpgsql;
```

Figura 26: Funcion de Trigger

4.6. índice Tabla

Usamos un indice para venta debido a que es el atributo para realizar la comunicacion entre la tabla Venta y Contiene, las cuales son las encargadas de llenar cada uno de los datos para hacer una venta. Ademas se juntó con una secuencia para que este se incremente de forma automatica y el usuario no tenga que interactuar con eso.

Cabe mencionar que es el valor con el que mas se interactua ya que se hacen varias consultas ya sea para obtener datos requeridos o para cancelar una venta en caso de que sea abortada.

```
CREATE INDEX id_Vent ON VENTA(id_Venta);
```

Figura 27: Indice Tabla

5. Presentación

Una interfaz gráfica es un programa que nos permite manipular información a través de objetos gráficos que proporcionen un entorno visual, con el fin facilitar la interacción del usuario con la computadora.

En este caso, se optó por desarrollar una página web como interfaz gráfica que permita la manipulación de información de nuestra base de datos.

5.1. Django

Django es un framework de Python de alto nivel que permite diseñar aplicaciones web de una forma rápida, limpia y pragmática. Además, ayuda a los desarrolladores a evitar muchos errores de seguridad comunes, como la inyección de SQL, las secuencias de comandos entre sitios, la falsificación de solicitudes entre sitios y el secuestro de clics.

5.1.1. Mapeo Relacional de Objetos

El Mapeo Relacional de Objetos o ORM (*Object Relational Mapping*), es una tecnología que soluciona el desajuste entre las bases de datos relacionales y orientadas a objetos.

Por lo general, asigna una clase a una tabla uno a uno. Cada instancia de la clase corresponde a un registro en la tabla y cada atributo de la clase corresponde a cada campo en la tabla. ORM proporciona una asignación a la base de datos, en lugar de escribir código SQL directamente, solo es necesario manipular los datos de la base de datos como un objeto operativo.

Si bien, el ORM es una herrmienta muy util, no se utilizará en este proyecto, ya que preferimos escribir directamente la sentencia SQL para comunicarnos con la base de datos.

5.1.2. Ejecutar SQL personalizado directamente

El objeto **django.db.connection** representa la conexión por defecto entre django y la base de datos. Las funciones que utilizamos para la comunicación con la base de datos son las siguientes:

- **connection.cursor()**

  Para obtener un objeto cursor.

- **cursor.execute(sql, [params])**

  Para ejecutar sentencias SQL.

- **cursor.fetchall()**

  Para devolver las filas resultantes de la consulta.

5.2. Diseño

Se desarrolló una página simple que cumpliera con los requerimientos y objetivos del proyecto. Básicamente, desarrollamos una interfaz con 5 vistas principales:

- **Bienvenida** Es la vista por defecto de nuestra página. Únicamente contiene el nombre de la página y un mensaje.

- **Tienda** Es la vista donde se muestra la lista de productos que existen en el inventario y presenta la funcionalidad de organizar la lista de productos de acuerdo a su categoría. Es en esta sección donde podremos añadir productos a nuestro carrito para realizar la compra posteriormente. Si se agregan varios productos iguales, sólo se incrementará el número de unidades de dicho producto.

- **Herramientas** En esta sección se presentan 4 funciones que podemos nos ayudan para analizar la utilidad de los productos, ganancia de las ventas, artículos que están por agotarse y la opción de consultar facturas de ventas realizdas.

- **Registro Cliente** Es un formulario en el que se ingresan los datos del cliente para realizar su registro en la base de datos. Cuenta con verificación de datos para que la insersión en la base de datos se ejecute de forma correcta.

- **Carrito** Es la vista en la que observamos los productos que vamos a comprar. Tenemos la posibilidad de eliminar un producto del carrito y de limpiar el carrito.

Las vistas secundarias incluyen las validaciones de datos y procesos intermedios para realizar una compra o buscar una factura.

Cada una de las vistas se realizó en un documento HTML y se controlan por medio de funciones del marco de trabajo Django escritas en Python. A continuación, se muestran algunos códigos de las vistas más representativas.

```python
def home(request):
    return render(request,'home.html')
```

Figura 28: Vista home

```
{% extends "./base.html" %}
{% load static %}

{% block title %} Papel & Lápiz {% endblock %}

{% block body %}

<div class="container-fluid">
    <div class="row align-items-center">
        <div class="col-4">
            <img src="{% static 'img/papeleria.png' %}" class="img-fluid">
        </div>
        <div class="col-8 text-center">
            <h1 class="card-title">Papel & Lápiz</h1>
            <p class="card-text">¡Encuentra tus articulos favoritos en nuestra sucursal más cercana!</p>
        </div>
    </div>
</div>

{% endblock %}
```

Figura 29: home.html

```python
def herramientas(request):
    return render(request,'herramientas.html')
```

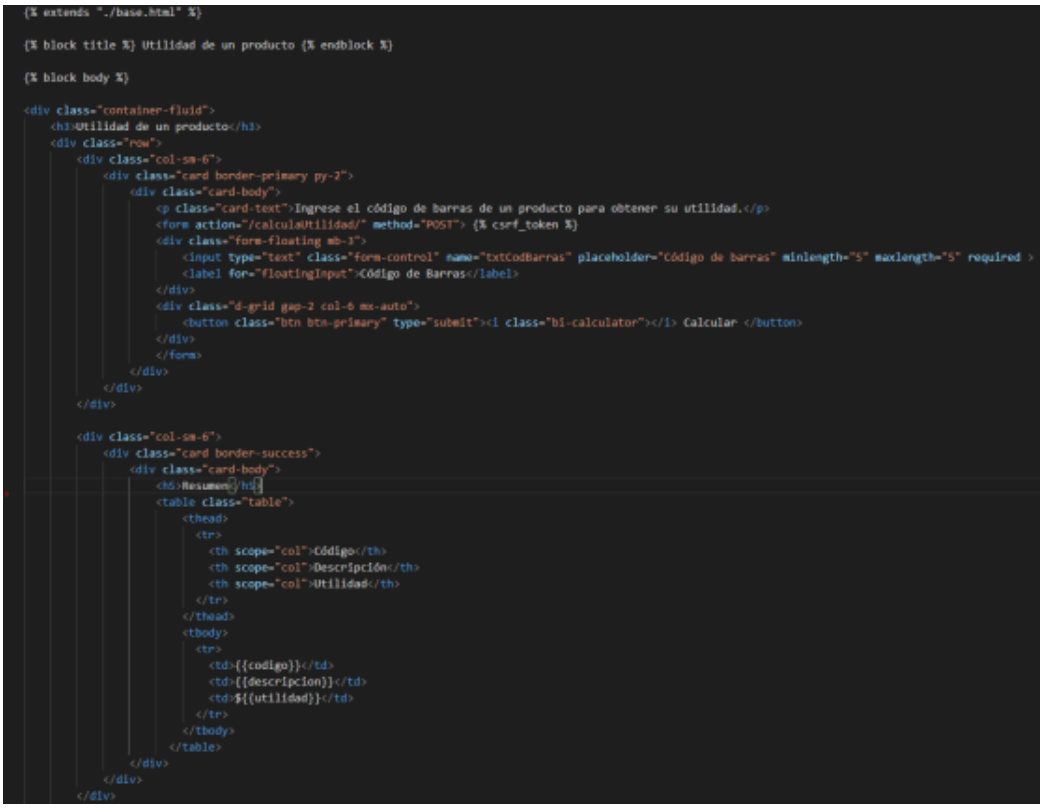
Figura 30: Vista herramientas

```
{% extends "./base.html" %}

{% block title %} Herramientas {% endblock %}

{% block body %}

<div class="container-fluid">
    <h3>Herramientas</h3>
    <div class="row">
        <div class="col-sm-3">
          <div class="card">
            <div class="card-body">
              <h5 class="card-title">Utilidad</h5>
              <p class="card-text">Obtenga la utlidad de un producto.</p>
              <a href="/utilidadProducto" class="btn btn-primary">Ir</a>
            </div>
          </div>
        </div>

        <div class="col-sm-3">
          <div class="card">
            <div class="card-body">
              <h5 class="card-title">Análisis de ventas</h5>
              <p class="card-text">Obtenga las ganancias de las ventas realizadas.</p>
              <a href="/analisisVentas" class="btn btn-primary">Ir</a>
            </div>
          </div>
        </div>

        <div class="col-sm-3">
            <div class="card">
              <div class="card-body">
                <h5 class="card-title">Revisión de inventario</h5>
                <p class="card-text">Conozca los productos que estan por agotarse.</p>
                <a href="/revisionInventario" class="btn btn-primary">Ir</a>
              </div>
            </div>
        </div>

        <div class="col-sm-3">
            <div class="card">
              <div class="card-body">
                <h5 class="card-title">Facturas</h5>
                <p class="card-text">Obtenga la información de una factura.</p>
                <a href="\busquedaVenta" class="btn btn-primary">Ir</a>
              </div>
            </div>
        </div>

    </div>
</div>

{% endblock %}
```

Figura 31: Herramientas.html

```
def utilidadProducto(request):
    return render(request, 'utilidadProducto.html')
```

Figura 32: Vista utilidadProducto



```
{% extends "./base.html" %}

{% block title %} Utilidad de un producto {% endblock %}

{% block body %}

<div class="container-fluid">
    <h3>Utilidad de un producto</h3>
    <div class="row">
        <div class="col-sm-6">
            <div class="card border-primary py-2">
                <div class="card-body">
                    <p class="card-text">Ingrese el código de barras de un producto para obtener su utilidad.</p>
                    <form action="/calculaUtilidad/" method="POST"> {% csrf_token %}
                    <div class="form-floating mb-3">
                        <input type="text" class="form-control" name="txtCodBarras" placeholder="Código de barras" minlength="5" maxlength="5" required >
                        <label for="floatingInput">Código de Barras</label>
                    </div>
                    <div class="d-grid gap-2 col-6 mx-auto">
                        <button class="btn btn-primary" type="submit"><i class="bi-calculator"></i> Calcular </button>
                    </div>
                    </form>
                </div>
            </div>
        </div>

        <div class="col-sm-6">
            <div class="card border-success">
                <div class="card-body">
                    <h5>Resumen</h5>
                    <table class="table">
                        <thead>
                          <tr>
                            <th scope="col">Código</th>
                            <th scope="col">Descripción</th>
                            <th scope="col">Utilidad</th>
                          </tr>
                        </thead>
                        <tbody>
                          <tr>
                            <td>{{codigo}}</td>
                            <td>{{descripcion}}</td>
                            <td>${{utilidad}}</td>
                          </tr>
                        </tbody>
                      </table>
                </div>
            </div>
        </div>
    </div>
```

Figura 33: utilidadProducto.html

```
def analisisVentas(request):
    return render(request, 'analisisVentas.html')
```

Figura 34: Vista analisisProducto

```
{% extends "./base.html" %}

{% block title %} Análisis de Ventas {% endblock %}

{% block body %}

<div class="container-fluid">
    <h3>Análisis de ventas</h3>
    <div class="row">
        <div class="col-sm-6">
            <div class="card border-primary py-2">
              <div class="card-body">
                <p class="card-text">Ingrese una fecha o una fecha de inicio y una fecha de fin.</p>
                <form action="/analisisVentasFecha/" method="POST"> {% csrf_token %}
                  <div class="form-group mb-3">
                    <label for="floatingInput">Fecha de inicio</label>
                    <input type="date" class="form-control" name="fDateS" placeholder="dd/mm/aaaa" required>
                  </div>
                  <div class="form-group mb-3">
                        <label for="floatingInput">Fecha de fin</label>
                        <input type="date" class="form-control" name="fDateE" placeholder="dd/mm/aaaa" >
                  </div>
                  <div class="d-grid gap-2 col-6 mx-auto">
                    <button class="btn btn-primary" type="submit"><i class="bi-calculator"></i> Calcular </button>
                  </div>
                </form>
              </div>
            </div>
          </div>
            <div class="col-sm-6">
                <div class="card border-success">
                    <div class="card-body">
                        <h5>Resumen</h5>
                        <table class="table">
                            <thead>
                              <tr>
                                <th scope="col">Periodo de tiempo</th>
                                <th scope="col">Ganancia total</th>
                              </tr>
                            </thead>
                            <tbody>
                              <tr>
                                <td>{{fecha1}} {{fecha2}}</td>
                                <td>${{ganancia}}</td>
                              </tr>
                            </tbody>
                          </table>
                    </div>
                </div>
            </div>

    </div>
</div>

{% endblock %}
```

Figura 35: analisisProducto.html

```python
def revisionInventario(request):
    consultaSQL = None
    productos = []
    with connection.cursor() as cursor:
        cursor.execute("SELECT menos_Thr();")
        consultaSQL = cursor.fetchall()

    for p in consultaSQL:
        reg, = p
        tup = tuple(reg[1:-1].replace('"','').split(','))
        prod = type('Producto', (object, ), dict(stock = tup[0], marca = tup[1], desc = tup[2]))
        productos.append(prod)

    return render(request, 'revInventario.html',{'productos':productos})
```

Figura 36: Vista analisisVentas

```
{% extends "./base.html" %}

{% block title %} Revisión de inventario {% endblock %}

{% block body %}

<div class="container-fluid">
    <div class="row">
        <div class="col-sm-2"></div>
        <div class="col-sm-8">
            <h3>Revisión de inventario</h3>
            <h6>Productos que estan por agotarse:</h6>

            <table class="table table-striped">
                <thead>
                    <tr>
                        <th scope="col">Descripción</th>
                        <th scope="col">Marca</th>
                        <th scope="col">Cantidad</th>
                    </tr>
                </thead>
                <tbody>
                    {% for p in productos %}
                    <tr>
                    <td>{{ p.desc}}</td>
                    <td>{{ p.marca }}</td>
                    <td style="color: red;">{{p.stock}}</td>
                    </tr>
                  {% endfor %}
                </tbody>
            </table>
        </div>
    </div>
</div>

{% endblock %}
```

Figura 37: revisionInventario.html

```python
def factura(request, idVenta):
    consulta = None
    with connection.cursor() as cursor:
        cursor.execute('SELECT * FROM FACTURA WHERE id_venta = %s;',[idVenta])
        consulta = cursor.fetchall()

        c = consulta[0]
        cliente = type('Cliente', (object,), dict(rfc=c[0], nom=c[1], apP=c[2], apM=c[3], cp=c[4], calle=c[5], col=c[6]))
        venta = type('Venta', (object,), dict( id=c[7], fecha=c[8], pago=c[9] ))
        productos = []
        for p in consulta:
            prod = type('Prod',(object,), dict(precio=p[10], cant=p[11], marca=p[12], desc=p[13], preciotot=p[11]*p[10]))
            productos.append(prod)
    return render(request, 'factura.html',{'cliente':cliente,'productos':productos, 'venta':venta})
```

Figura 38: Vista Factura

```
{% extends "./base.html" %}

{% block title %} Factura {% endblock %}

{% block body %}

<div class="container-fluid">
    <div class="row">
        <div class="col">
            <div class="card">
                <div class="card-body">

                    <div class="row mx-2 my-4">
                      <h3 class="card-title">Factura</h3>
                      <h4 class="card-subtitle mb-2 text-muted">Papel & Lápiz.</h4>
                    </div>

                    <div class="row mx-2">
                      <div class="col">
                        <p><b>FACTURAR A:</b><br>
                          Nombre: {{cliente.nombre}} {{cliente.apP}} {{cliente.apM}}<br>
                          RFC: {{cliente.rfc}}<br>
                          Dirección: {{cliente.calle}}, {{cliente.col}}. CP {{cliente.cp}} <br>
                        </p>
                      </div>

                      <div class="col"></div>

                      <div class="col">
                        <p><b>FACTURA:</b> FACT-{{venta.id}}</p>
                        <p><b>FECHA:</b> {{venta.fecha}}</p>
                      </div>
                    </div>
                    <div class="row mx-4">
                      <table class="table table-striped">
                        <thead>
                          <tr>
                            <th scope="col">Cantidad</th>
                            <th scope="col">Descripción</th>
                            <th scope="col">Precio Unitario</th>
                            <th scope="col">Precio Importe</th>
                          </tr>
                        </thead>
                        <tbody>
                          {% for p in productos %}
                          <tr>
                            <td>{{p.cant}}</td>
                            <td>{{p.desc}}</td>
                            <td>{{p.precio}}</td>
                            <td>{{p.preciotot}}</td>
                          </tr>
                          {% endfor %}
                          <tr>
                            <th scope="col"></th>
                            <th scope="col"></th>
                            <th scope="col">Total</th>
                            <th scope="col">{{venta.pago}}</th>
                          </tr>
                        </tbody>
                      </table>
                    </div>
                  </div>
                </div>
            </div>
        </div>
        <div class="row">
          <div class="col-11"></div>
          <div class="col-1">
            <a class="btn btn-primary my-3" href="\home" type="summit">Finalizar</a>
          </div>
        </div>
      </div>
    </div>
</div>

{% endblock %}
```

Figura 39: factura.html

```
def formularioCliente(request):
    return render(request, 'formularioCliente.html')
```

Figura 40: Vista FormularioCliente

Figura 41: FormularioCliente.html

```python
def carrito(request):
    productos = []
    for pr, li in miCarrito.canasta.items():
        producto = type('Prod', (object, ), dict(codigo=pr, cant=li[0], precio=li[1], desc=li[2], total=li[1]*li[0]) )
        productos.append(producto)

    return render(request, 'carrito.html',{'productos':productos})
```

Figura 42: Vista Carrito

```
{% extends "./base.html" %}

{% block title %} Mi Carrito {% endblock %}

{% block body %}

{% load static %}

<div class="container-fluid">
    <div class="row">
        <div class="col-1"></div>
        <div class="col-10">
            <h3>Mi Carrito</h3>
            <div class="card py-2">
                <div class="card-body">
                    <table class="table table-striped">
                        <thead>
                          <tr>
                            <th scope="col-2">Código</th>
                            <th scope="col-2">Descripción</th>
                            <th scope="col-2">Precio Unitario</th>
                            <th scope="col-2">Cantidad</th>
                            <th scope="col-2">Precio Total</th>
                            <th scope="col-2">Eliminar del carrito</th>
                          </tr>
                        </thead>
                        <tbody>
                          {% for p in productos %}
                          <tr>
                          <td>{{p.codigo}}</td>
                          <td>{{p.desc}}</td>
                          <td>${{p.precio}}</td>
                          <td> {{p.cant}} </td>
                          <td>${{p.total}}</td>
                          <td><a href="/eliminacionArticulo/{{p.codigo}}" class="btn btn-danger btnEliminar"><i class="bi-trash"></i></a></td>
                          </tr>
                          {% endfor %}
                        </tbody>
                    </table>
                    <a href="/cancelarCompra" class="btn btn-danger">Limpiar carrito</a>
                    <a href="/cliente" class="btn btn-primary">Siguiente</a>
                </div>
            </div>
        </div>
    </div>
</div>
<script src="//cdn.jsdelivr.net/npm/sweetalert2@11"></script>
<script src="{% static 'js/carrito.js' %}"></script>

{% endblock %}
```

Figura 43: Carrito.html

```python
def cliente(request):
    return render(request, 'clienteventa.html')
```

Figura 44: Vista Cliente

```
{% extends "./base.html" %}

{% block title %} Cliente {% endblock %}

{% block body %}

<div class="container-fluid">
    <div class="row">
        <div class="col-4"> </div>
        <div class="col-4">
          <h3>Cliente</h3>
          <div class="card py-2">
              <div class=" card-body" >
                  <form class="needs-validation" novalidate action="/validaCliente/" method="POST"> {% csrf_token %}
                      <p>Ingrese el RFC del cliente que realiza la compra:</p>
                      <div class="form-group col">
                        <div class="form-floating">
                          <input type="text" name="txtRFC" class="form-control" placeholder="RFC" minlength="13" maxlength="13" required >
                          <label for="floatingInput">RFC</label>
                        </div>
                        <div class="input-group-btn my-3">
                          <a href="/miCarrito" class="btn btn-secondary">Regresar</a>
                          <button class="btn btn-primary" type="submit">Siguiente</button>
                        </div>
                      </div>
                  </form>
              </div>
          </div>
        </div>

    </div>
</div>
{% block scripts %}

{% endblock %}
```

```
<script>
  // Example starter JavaScript for disabling form submissions if there are invalid fields
  (function() {
    'use strict';
    window.addEventListener('load', function() {
      // Fetch all the forms we want to apply custom Bootstrap validation styles to
      var forms = document.getElementsByClassName('needs-validation');
      // Loop over them and prevent submission
      var validation = Array.prototype.filter.call(forms, function(form) {
        form.addEventListener('submit', function(event) {
          if (form.checkValidity() === false) {
            event.preventDefault();
            event.stopPropagation();
          }
          form.classList.add('was-validated');
        }, false);
      });
    }, false);
  })();
  </script>
{% endblock %}
```

Figura 45: Cliente.html

```python
from django.urls import path
from . import views

urlpatterns = [
    path('', views.home),
    path('home/', views.home),
    path('formularioCliente/',views.formularioCliente),
    path('validaRegistroCliente/', views.validaRegistroCliente),
    path('formularioError/',views.formularioError),
    path('herramientas/', views.herramientas),
    path('utilidadProducto/',views.utilidadProducto),
    path('calculaUtilidad/',views.calculaUtilidad),
    path('utilidadError/',views.utilidadError),
    path('analisisVentas/',views.analisisVentas),
    path('revisionInventario/',views.revisionInventario),
    path('miCarrito/',views.carrito),
    path('tienda/<int:cat>',views.tienda),
    path('cliente/',views.cliente),
    path('clienteError/',views.clienteError),
    path('validaCliente/', views.validaCliente),
    path('factura/<int:idVenta>',views.factura),
    path('eliminacionArticulo/<int:codigo>', views.eliminar_Articulo),
    path('addCarrito/<int:codigo>', views.addCarrito),
    path('cancelarCompra/',views.cancelarCompra),
    path('venta/<rfc>',views.venta),
    path('busquedaVenta/', views.busquedaVenta),
    path('busquedaError/', views.busquedaError),
    path('validaVenta/', views.validaVenta),
    path('analisisVentasFecha/', views.analisisVentasFecha)
]
```

Figura 46: urls.py

```python
def validaRegistroCliente(request):
    #is_private = request.POST.get('is_private', False)
    nombre = request.POST['txtNombreform']
    apP = request.POST['txtApP']
    apM = request.POST['txtApM']
    rfc = request.POST['txtRFC']
    correo = request.POST['emCorreo']
    calle = request.POST['txtCalle']
    colonia = request.POST['txtColonia']
    estado = request.POST['txtEstado']
    numero = int(request.POST.get('txtNum',False))
    cp = int(request.POST.get('txtCP',False))

    with connection.cursor() as cursor:
        try:
            if len(apM) != 0:
                cursor.execute("INSERT INTO cliente VALUES(%s,%s,%s,%s,%s,%s,%s,%s,%s);",
                        [rfc, nombre, apP, apM, cp, numero, estado, calle, colonia])
            else:
                cursor.execute("INSERT INTO cliente VALUES(%s,%s,%s,NULL,%s,%s,%s,%s,%s);",
                        [rfc, nombre, apP, cp, numero, estado, calle, colonia])

            cursor.execute("INSERT INTO correo VALUES(%s,%s);",[correo, rfc])
        except:
            cursor.execute("ROLLBACK;")
            return redirect('/formularioError')

    return redirect('/formularioCliente')
```

Figura 47: Vista validaRegistroCliente

```python
def calculaUtilidad(request):
    codigo = int(request.POST['txtCodBarras'])
    consultaSQL = None
    desc = None
    with connection.cursor() as cursor:
        cursor.execute("SELECT retorna_Utilidad(%s);",[codigo])
        consultaSQL = cursor.fetchone()
        cursor.execute("SELECT descripcion FROM producto WHERE cod_Barras=%s;",[codigo])
        desc = cursor.fetchone()
    util, = consultaSQL

    if util != None:
        desc, = desc
        return render(request, 'utilidadProducto.html',{'codigo':codigo,'descripcion':desc,'utilidad':util})
    else:
        return redirect('/utilidadError')
```

Figura 48: Vista calculaUtilidad

```python
def analisisVentasFecha(request):
    fecha1 = request.POST['fDateS']
    fecha2 = request.POST['fDateE']
    consultaSQL = None
    desc = None
    with connection.cursor() as cursor:
        if fecha2 != '':
            cursor.execute("SELECT retorna_pago_Final(%s,%s);",[fecha1, fecha2])
        else:
            cursor.execute("SELECT retorna_pago_Final(%s);",[fecha1])
        consultaSQL = cursor.fetchone()
    ganancia, = consultaSQL

    if ganancia != None:
        return render(request, 'analisisVentas.html',{'fecha1':fecha1, 'fecha2':fecha2,'ganancia':ganancia})
    else:
        return redirect('/analisisVenta')
```

Figura 49: Vista analisisVentasFecha

```python
def revisionInventario(request):
    consultaSQL = None
    productos = []
    with connection.cursor() as cursor:
        cursor.execute("SELECT menos_Thr();")
        consultaSQL = cursor.fetchall()

    for p in consultaSQL:
        reg, = p
        tup = tuple(reg[1:-1].replace('"','').split(','))
        prod = type('Producto', (object, ), dict(stock = tup[0], marca = tup[1], desc = tup[2]))
        productos.append(prod)

    return render(request, 'revInventario.html',{'productos':productos})
```

Figura 50: Vista revisionInventario

```python
def tienda(request, cat):
    categorias = []
    productos = []
    categ = "Todos los productos"
    categoriasSQL = None
    productosSQL = None

    with connection.cursor() as cursor:
        cursor.execute("SELECT * FROM categoria;")
        categoriasSQL = cursor.fetchall()

        if cat == 0:
            cursor.execute("SELECT * FROM producto;")
        else:
            cursor.execute("SELECT tipo FROM categoria WHERE id_categoria=%s;",[cat])
            categ, = cursor.fetchone()
            cursor.execute("SELECT * FROM producto WHERE id_categoria=%s;",[cat])
        productosSQL = cursor.fetchall()

    for c in categoriasSQL:
        catObj = type('Categoria', (object, ), dict(id = c[0], tipo = c[1]))
        categorias.append( catObj )

    for p in productosSQL:
        productos.append( producto(p) )

    return render(request, 'categoria.html',{'idcat':cat,'categoria':categ,'categorias':categorias, 'productos':productos})
```

Figura 51: Vista tienda

```python
def venta(request, rfc):
    pagoVenta = miCarrito.compra()
    idVenta = None
    print(pagoVenta)
    with connection.cursor() as cursor:
        cursor.execute('INSERT INTO venta(pago_final,RFC) VALUES(%s,%s);',[pagoVenta, rfc])
        cursor.execute('SELECT id_Venta_Funcion();')
        idVenta, = cursor.fetchone()

        for pr, li in miCarrito.canasta.items():
            print("Dentro for")
            pagoTotal = li[0]*li[1]
            try:
                cursor.execute('INSERT INTO contiene(cod_Barras, id_Venta, precio_Total_Art, cantidad_Articulo) VALUES(%s,%s,%s,%s);',[int(pr), int(idVenta), float(pagoTotal) , li[0]])
            except:
                cursor.execute('DELETE FROM VENTA WHERE id_Venta = %s;',[idVenta])

    miCarrito.drop()
    return redirect('/factura/{}'.format(idVenta))
```

Figura 52: Vista Venta

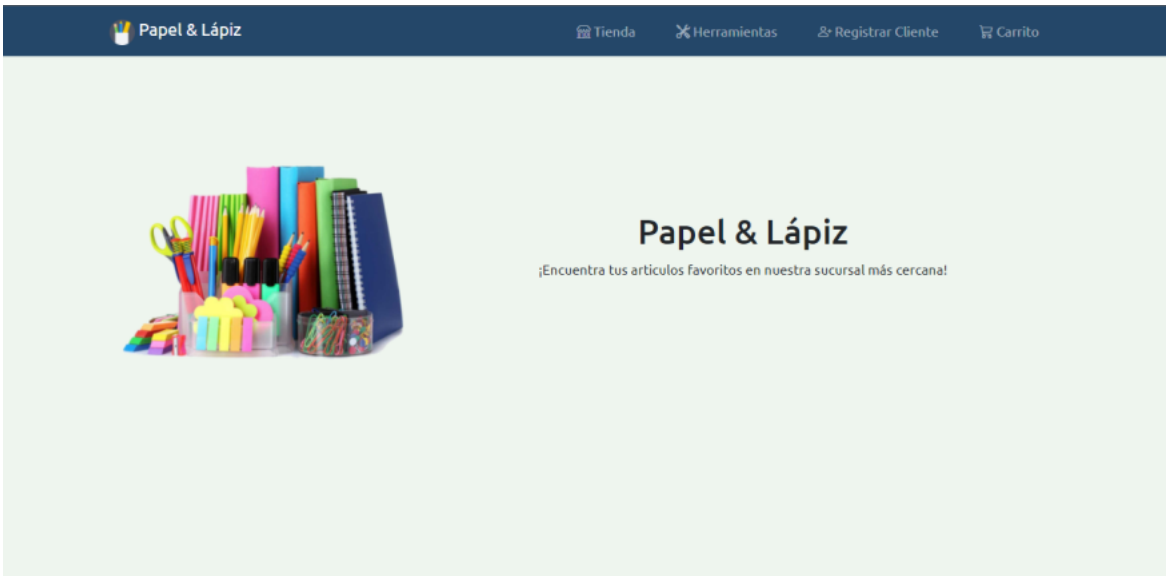


Figura 53: Pagina de inicio de la Papeleria

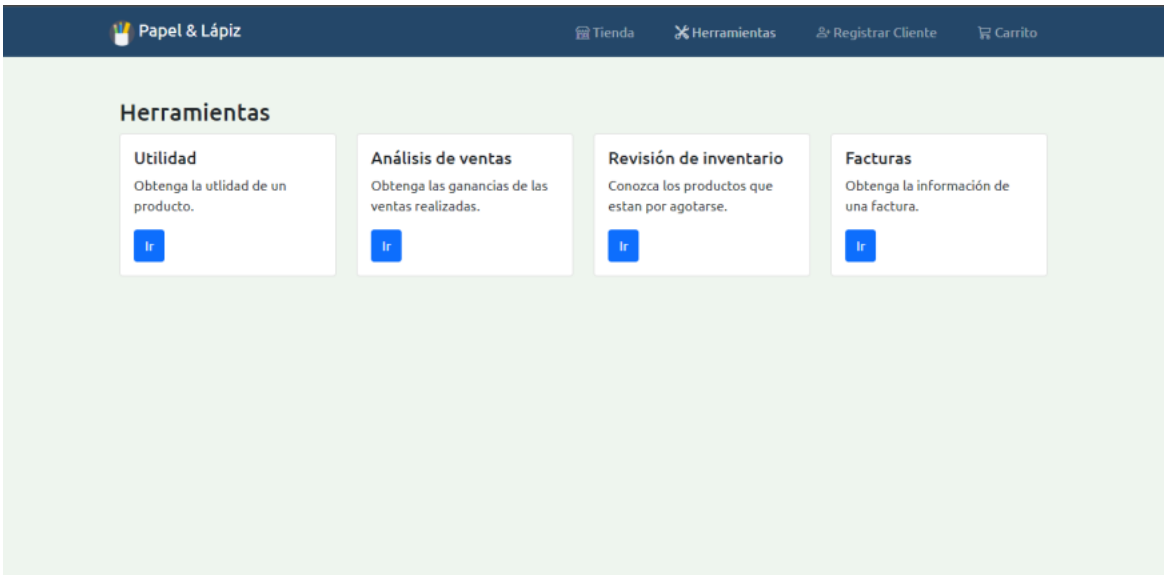


Figura 54: Pagina de herramientas

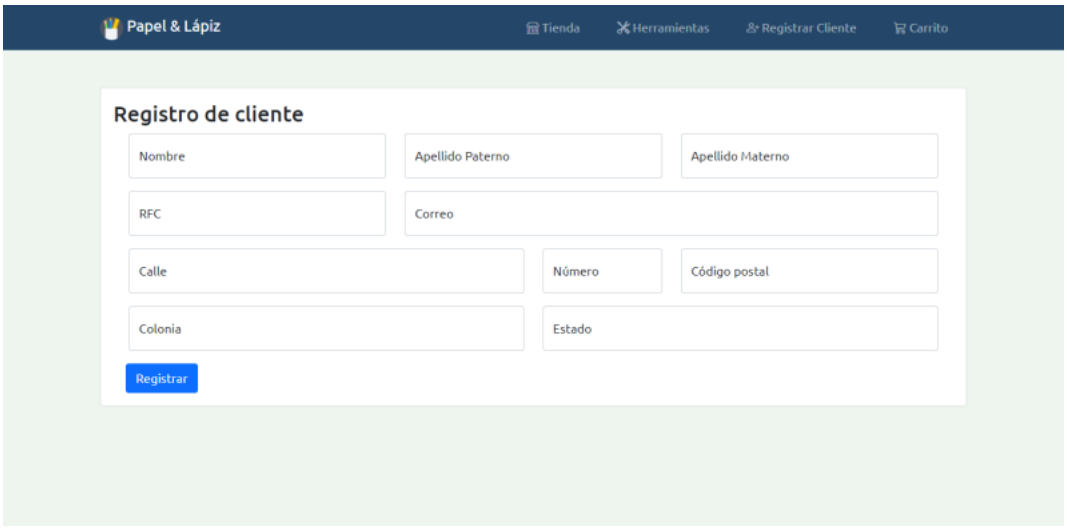


Figura 55: Pagina de registro de cliente

Figura 56: Factura

6. Conclusiones

- Fonseca Huitrón Julise Aileen:

  Gracias a la buena organización y trabajo en equipo los objetivos del proyecto se cumplieron, al realizar satisfactoriamente los requerimientos solicitados para la creación de la base de datos.

  Retomando el punto de la buena organización, este fue fundamental para poder designar tareas a cada integrante, a pesar de que las tareas fueron divididas en dos grupos (back y front end) se ocuparon herramientas colaborativas para realizar reuniones continuas e informar al resto del equipo sobre los avances y funcionamientos de cada parte, sin duda alguna la clave para que todo concordara fue que en la parte de diseño, es decir, el análisis de requerimientos, el modelo entidad relación, la representación intermedia, el modelo relacional, etc., todos los integrantes del equipo participaron.

  Debido a que en la parte de diseño se ocuparon todos los conocimientos adquiridos en clase, así como la participación de cada integrante, se logró un avance eficiente y eficaz. En la parte de implementación, presentación y acoplamiento se presentaron algunos contratiempos, estos fueron desde los diversos horarios, hasta el desconocimiento de funciones, trigger y herramientas para el uso de la página web. En estas ultimas se tuvieron que consultar tutoriales y documentación que nos permitieran tener las bases e ideas principales para la resolución del problema.

- López Aniceto Saúl Isaac:

  Gracias al proyecto comprendí más acerca de más bases de datos, de cómo estás pueden tener diferentes tipos de datos para la solución de problemas, que en este caso fue la de una papelería, y también el cómo crearlas desde cero para después implementarla a una página web con su respectiva interfaz y herramientas solicitadas. Al igual logré comprender y ejecutar las funciones requeridas por medio de PL/PGPSQL para cada una de las consultas y herramientas.

  Al igual comprendí más la importancia del trabajo en equipo y de como una buena organización puede hacer que la carga de trabajo sea menos para todos los integrantes y se pueda trabajar de una mejor manera para poder cumplir con el tiempo de trabajo solicitado.

  No estuve muy presente en la implementación del código Django debido a la división de trabajo de equipo, pero si estuve viendo cómo se fue programando, así que superficialmente aprendí acerca de este lenguaje que ayuda más a la implementación de la base gracias a todas las funciones que este ya trae consigo.

- López González Kevin:

  Al finalizar este proyecto aprendí a realizar una base de datos, desde su diseño conceptual hasta su diseño físico. Luego, de crear la base de datos, aprendí a desarrollar una página web en un servidor local, que sirviera de interfaz entre el usuario y la base de datos con el objetivo

de que el manejo de información sea más sencilla. Lo que realmente aprendí y disfruté mucho, fue a conectar una base de datos con el marco de trabajo de Django. Si bien, Django facilita la comunicación entre la página web y la base de datos gracias a su modelo nativo ORM, decidimos tomar el camino dificil, por lo que la comunicación se realizó mediante sentencias SQL. Esto tiene como ventaja el gran control que tenemos sobre la base de datos, ya que podemos ejecutar cualquier sentencia SQL directamente en la base de datos. Es recomendable tener los modelos de la base de datos como clases de Python, ya que aseguran la integridad de los datos, sin embargo, decidimos no usarlos para tener más control sobre la base de datos. Esto nos lleva a que probablemente tengamos problemas de integridad con nuestros datos, pero seguramente lo resolveremos en un futuro, al igual que colocar la página web en un servidor público y la base de datos en una plataforma de nube como AWS.

- Martínez Vázquez Diego:

  La realización de este proyecto fue muy enriquecedor, fueron muchos los beneficios que obtuve, al iniciar la planeación del proyecto la parte de análisis de requerimientos me dejó claro la importancia de los primeros temas de la materia, el concepto de entidades, relaciones, modelo entidad relación, modelo relacional y puede ver ahora si todo esto aplicado a un problema real. En esta parte la principal problemática es que como equipo teníamos soluciones diferentes y propuestas diferentes al problema, tuvimos que analizar de forma grupal y eso nos tomó algo de tiempo, afortunadamente logramos hacer un buen consenso y definimos como sería la representación del problema.

  En la parte de creación de la base de datos, fue un gran reto, ya que al principio todo fue un tanto sencillo con la creación de las tablas y la inserción de datos en algunas de ellas para las pruebas. Lo difícil comenzó al momento de crear las funciones, pues al momento de empezar de desarrollar esa parte aún no tenía los conocimientos suficientes para implementarlos, tuve que recurrir a diversos tutoriales en internet donde si bien tardé un poco por estar con prueba y error, prueba y error, finalmente pude empezar a crear las funciones que se me asignaron.

  Tomó algo de tiempo entender la parte de programación en base de datos, dría que ese fue un reto muy fuerte, así como fue un gran acierto comenzar a trabajar de forma autodidacta, hacer múltiples intentos y dedicarle días enteros a esta parte.

  En el desarrollo de la documentación, no consideraría que tuve alguna problemática en especial ya que mi función fue más de documentar algunas partes que ya estaban creadas como la base de datos, las tablas, así como la explicación del MR y MRE, también el desarrollo de la introducción y análisis de requerimientos. Diría que esta parte fue enriquecedora también ya que al estar documentando pude ver con más detalle cómo iba quedando nuestro trabajo, lo que funcionó en gran medida para afinar algunos detalles y darle una mejor estética al programa.

  Fue un reto importante también la comunicación entre compañeros, sobre todo al momento de hacer que algunas cosas de la base de datos fueran explicadas y detalladas para que los compañeros que se encargaban de la página web pudieran darle un mejor manejo tanto a la estructura de la base, como a los datos y funciones creadas sobre esta, yo diría que ese fue un reto, que se pudo solucionar con una video llamada de larga duración.

- Ponce Soriano Armando:

  Al termino de este proyecto, mis conocimientos referente al curso completo de Bases de datos fueron reforzados de gran manera, esto gracias a la creación de una base de datos enfocada a una problemática real, todo esto desde cero. Al haber comenzado desde la fase de planteamiento del problema, pasado por el diseño del modelo lógico, al modelo físico y finalmente implementándolo fuera del propio manejador de BD, todos mis conocimientos fueron puestos en prueba, reforzándolos de forma directa.
  Hablando propiamente de la parte del proyecto que me fue asignada, en este caso el diseño y creación de la página web, el uso de un framework como Django acelero el proceso de creación, ya que este brindo las herramientas para hacer que la conexión a nuestra BD fuera de una manera más sencilla. De igual forma, el uso de herramientas para diseño de páginas web como Bootstrap o Sweetalert nos permitieron brindarle un aspecto más profesional a nuestra página.

  En algún punto se planteo la idea de implementar la BD dentro de un servicio web de instancias en la nube, y aunque se logró una conexión exitosa con esta, debido a diversos problemas en la recta final del proyecto, esta idea tuvo que ser relegada parcialmente, seria ideal que en una futura iteración esto sea posible

  Me llevo de este proyecto mucha experiencia, mucho conocimiento y muchas ganas de seguir realizando proyectos de este tipo.