



Papeleria Papel & Lápiz

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

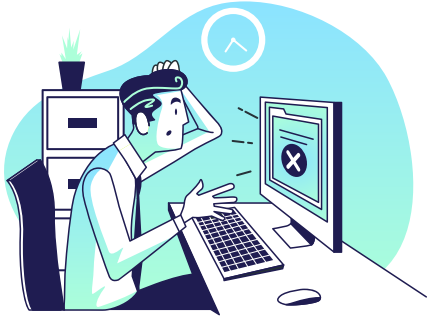
Objetivo

El siguiente proyecto tiene como meta elaborar desde cero una base de datos con el propósito de ser utilizada por una cadena de papelerías. Dicha base de datos se hará cargo de almacenar y manipular toda la información de inventario, clientes, proveedores y ventas de la cadena.

Para las interacciones Papelería-BD se realizará una página web en la que se tendrá acceso a funciones de la misma, tales como: La realización de una venta, una factura automática, etc.

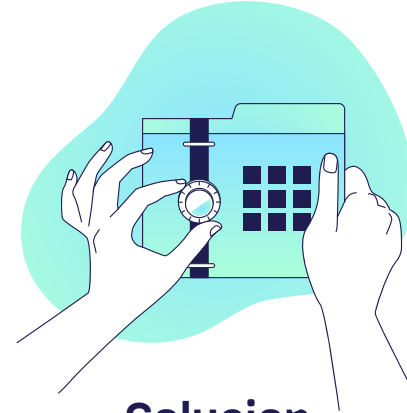


Planteamiento de la problemática



Problema

Una cadena de papelerías busca innovar la manera en que almacena su información



Solucion

Creación de una página web cuyos datos sean almacenados en una base de datos relacional

Plan de trabajo

Para la óptima realización del proyecto se decidió implementar de cierto modo metodologías ágiles (SCRUM) en todo el proceso.



SCRUM nos permitió

- Definir el Product Backlog: ordenar de mayor a menor importancia las funcionalidades pedidas por el cliente.
- Desarrollar la lista de tareas de 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

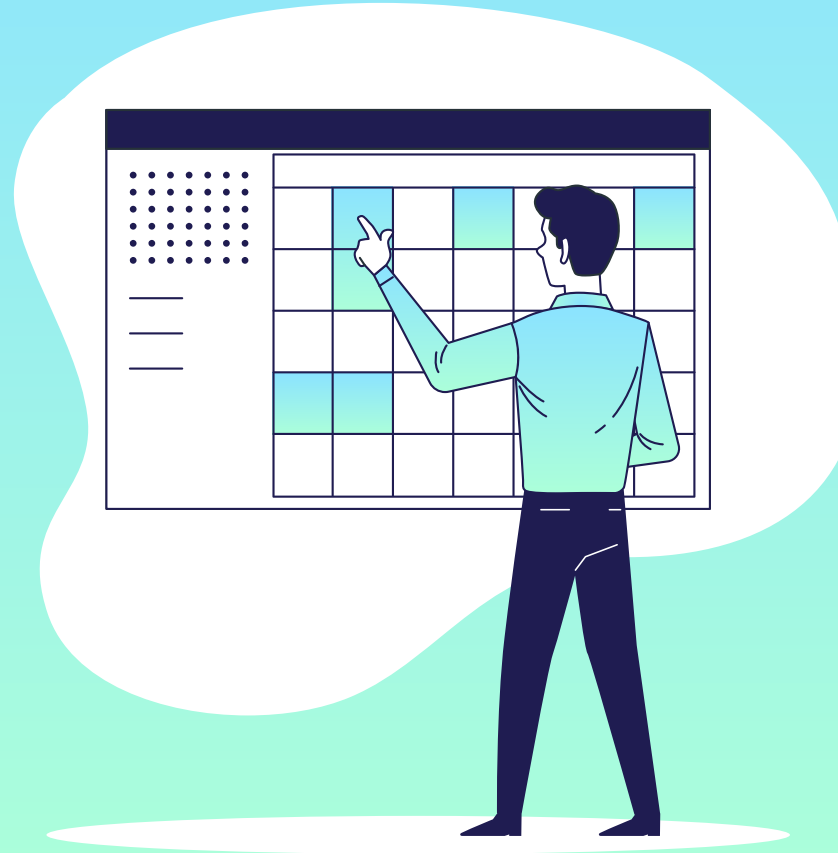


Aportaciones

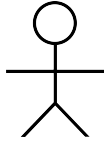
	Diseño	Implementación	Presentación	Acoplamiento	Documentación
Fonseca Huitrón Julise	✓	✓			✓
López Aniceto Isaac	✓	✓			
López González Kevin	✓		✓	✓	✓
Martínez Vázquez Diego	✓	✓			✓
Ponce Soriano Armando	✓		✓	✓	

Modelo Conceptual

Gracias al análisis de requerimientos hecho anteriormente, pudimos llevar a cabo el modelo conceptual, donde obtuvimos de manera más clara las entidades y las relaciones que hay para realizar el Modelo Entidad-Relación.



Entidades



PROVEEDOR: { id Proveedor, razón social, domicilio (estado, código postal, colonia, calle y número), nombre, teléfonos }

CLIENTE: {RFC, nombre (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, categoría}

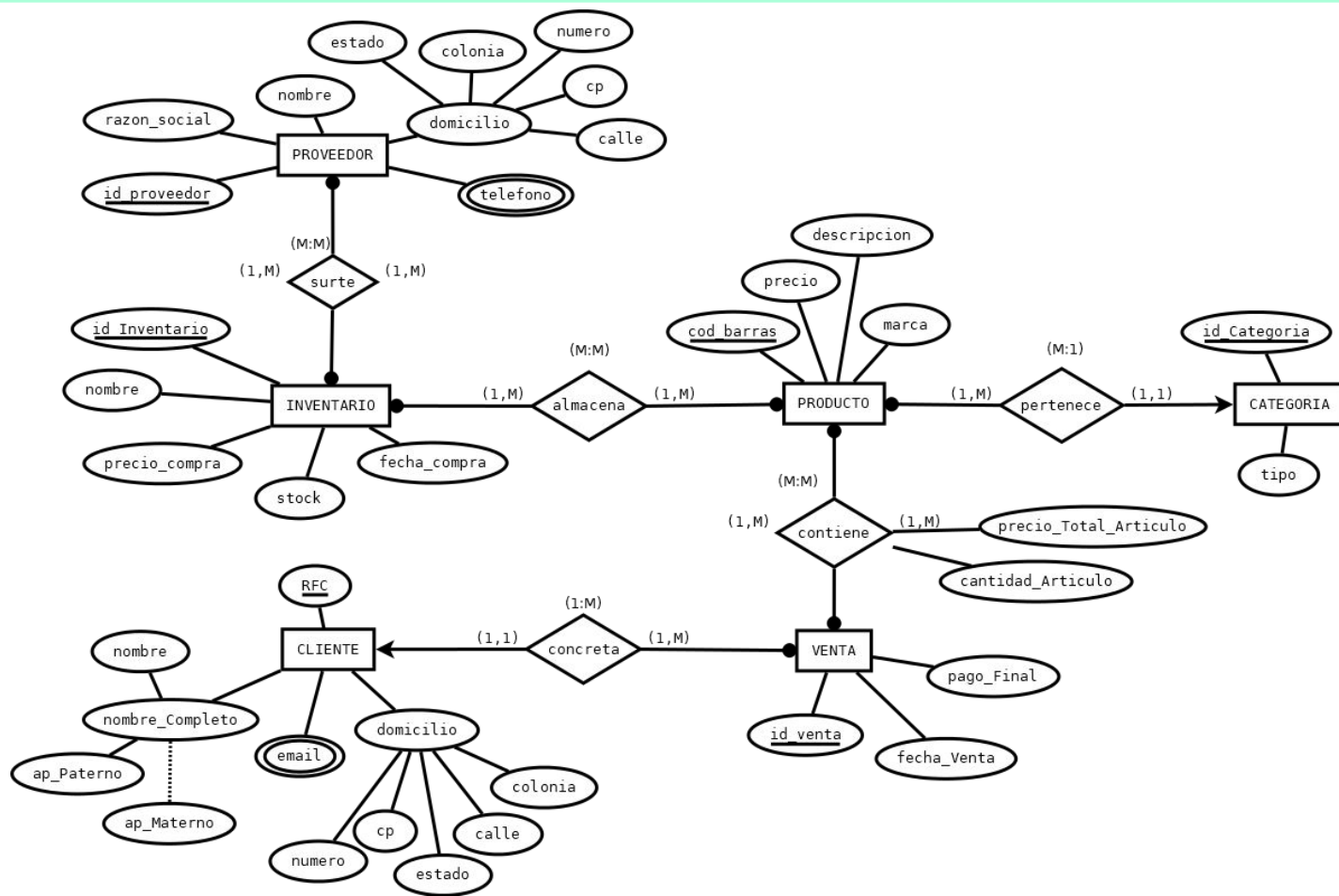
VENTA : {num venta, fecha venta, pago Total, cantidad artículo, pago total Artículo }

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.



Modelo Entidad-Relación



Representación Intermedia

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: {num Telefono bigint(PK), id proveedor smallint (FK)}

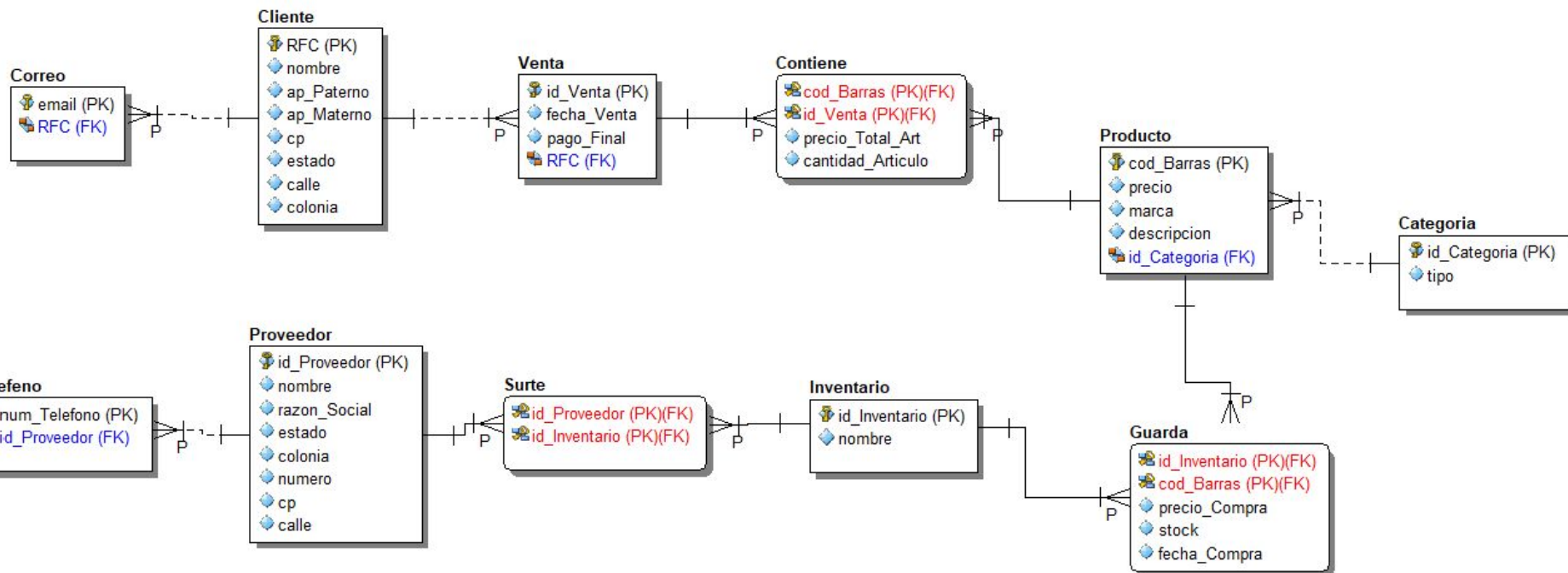
INVENTARIO: {id Inventario smallint (PK), nombre varchar(50)}

SURTE: {[id Proveedor smallint (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)}

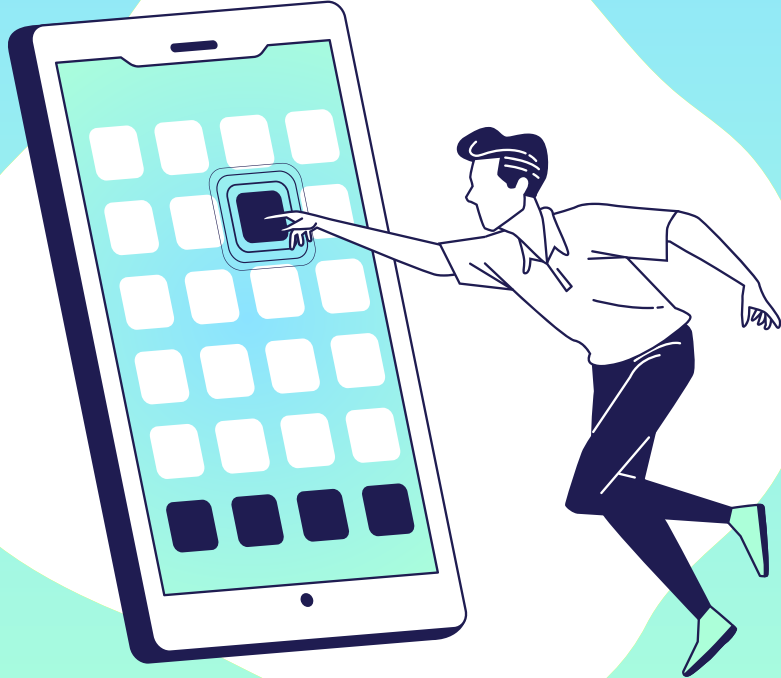
CATEGORÍA: { id categoria smallint PK, tipo varchar(20) NOT NULL}

Modelo Relacional



Modelo físico

Para el desarrollo de la base de datos y su implementación, tendremos que trabajar con la creación de tablas y sus respectivas inserciones en el manejador, tendremos que utilizar la programación en base de datos para que se logren hacer las funciones que se nos solicitan.



Creación de tablas

Tabla Cliente y Correo

```
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)  
);
```

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

Creación de tablas

```
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)  
);
```

Tabla Venta y Contiene

```
CREATE TABLE Contiene(  
    cod_Barras        INTEGER          NOT NULL,  
    id_Venta          INTEGER          NOT NULL,  
    precio_Total_Art  DECIMAL(7, 2)    NOT NULL,  
    cantidad_Articulo INTEGER          NOT NULL,  
    CONSTRAINT PK4 PRIMARY KEY (cod_Barras, id_Venta)  
);
```

Creación de tablas

Tabla Producto y Categoría

```
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)  
);
```

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


Creación de tablas

Tabla Guarda e Inventario

```
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)  
);
```

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

Creación de tablas

```
CREATE TABLE Proveedor(  
    id_Proveedor    SMALLINT    NOT NULL,  
    nombre          VARCHAR(50)  NOT NULL,  
    razon_Social    VARCHAR(50)  NOT NULL,  
    estado          VARCHAR(50)  NOT NULL,  
    colonia         VARCHAR(50)  NOT NULL,  
    numero         SMALLINT    NOT NULL,  
    cp             SMALLINT    NOT NULL,  
    calle          VARCHAR(50)  NOT NULL,  
    CONSTRAINT PK7 PRIMARY KEY (id_Proveedor)  
);
```

Tabla Surte y Proveedor

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

Creación de tablas

Tabla Teléfono

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

Creación de funciones

Vista Factura

```
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;
```

Creación de funciones

Función que nos da el nombre de los productos cuando hay menos de 3 en stock

```
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;
```

Creación de funciones

Función donde se puede consultar el último id de Venta agregado.

```
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;
```

Creación de funciones

Función que nos permite regresar la utilidad de un producto

```
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;
```

Creación de funciones

Función retorna pago final

```
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;
```


Creación de Función

Función que retorna el pago final a partir de un rango

```
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;
```

Creación de funciones

Función que retorna la ganancia de una fecha

```
CREATE OR REPLACE FUNCTION retorna_Pago_Ganancia(fecha DATE) RETURNS DECIMAL(7,2) AS
$$
DECLARE
v_Pago DECIMAL(7,2);
v_Bruto DECIMAL(7,2);
v_Ganancia DECIMAL(7,2);
BEGIN
SELECT SUM(pago_Final), SUM(precio_Compra*cantidad_articulo)
INTO v_Pago, v_Bruto FROM Venta
INNER JOIN (Contiene INNER JOIN (Producto INNER JOIN Guarda ON Producto.cod_Barras = Guarda.cod_Barras)
ON Contiene.cod_Barras = Producto.cod_Barras)
ON Venta.id_Venta = Contiene.id_Venta
WHERE Venta.fecha_venta = fecha;
v_Ganancia = v_Pago - v_Bruto;
RETURN v_Ganancia;
END;
$$
LANGUAGE plpgsql;
```

Creación de funciones

Función que retorna la ganancia entre dos fechas

```
CREATE OR REPLACE FUNCTION retorna_Pago_Ganancia (fecha_inicio DATE, fecha_fin DATE) RETURNS DECIMAL(7,2) AS
$$
DECLARE
v_Pago DECIMAL(7,2);
v_Bruto DECIMAL(7,2);
v_Ganancia DECIMAL(7,2);
BEGIN
SELECT SUM(pago_Final), SUM(precio_Compra*cantidad_articulo)
INTO v_Pago, v_Bruto FROM Venta
INNER JOIN (Contiene
INNER JOIN (Producto
INNER JOIN Guarda ON Producto.cod_Barras = Guarda.cod_Barras)
ON Contiene.cod_Barras = Producto.cod_Barras)
ON Venta.id_Venta = Contiene.id_Venta
WHERE Venta.fecha_Venta BETWEEN fecha_inicio AND fecha_fin;
v_Ganancia = v_Pago - v_Bruto;
RETURN v_Ganancia;
END;
$$
LANGUAGE plpgsql;
```

Creación de funciones

Función para decrementar el stock y cancelar transacciones

```
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;
```

Creación de Trigger

Disparador que se ejecuta en la función venta cuando se realice la instrucción de insertar sobre la tabla Contiene

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

Creación de Índice

Índice sobre tabla Venta

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

Secuencia para id_Vent

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

Edición de tablas

ALTER TABLE de Venta

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



Página Web

DESCRIPCIÓN

La página web se desarrolló como medio de conexión hacia la base de datos.

De esta forma, la comunicación entre el cliente y la BD es más sencilla y simplifica el proceso de realizar operaciones.

Además, se utilizaron iconos e imágenes con el fin de hacer una aplicación intuitiva para que cualquier usuario sea capaz de utilizarla.

Para la conexión y la creación de la página se implementó un framework basado en python llamado Django.

Para el estilo, utility responsive, alertas y presentación general del sitio se usaron herramientas como Bootstrap, SweetAlert y JavaScript



Inicio
Vista inicial de la página web.

01

04
Registrar Cliente
Formulario de datos para registrar un cliente.

Tienda
Sección donde se muestran los productos disponibles.

02

05
Carrito
Lista de productos seleccionados para comprar.

Herramientas
Sección de funciones para el análisis de ventas y consulta de facturas.

03


06
Comunicación con la BD
Vistazo rápido a las funciones medulares de la página.



Papel & Lápiz

¡Encuentra tus artículos favoritos en nuestra sucursal más cercana!

Vista de tienda












 **Papel & Lápiz**

TiendaHerramientasRegistrar ClienteCarrito


Categorías


- impresiones
- papeleria
- regalos
- recargas


Todos los productos


Código	Descripción	Marca	Precio	Añadir al carrito
53201	Impresion a color tamaño carta	Epson	5.00	
53202	Impresion B/N tamaño carta	Epson	1.00	
53203	Impresion a color tamaño oficio	Canon	10.00	
53204	Impresion B/N tamaño oficio	Canon	3.00	
53205	Impresion OFFSET	Epson	50.00	
53206	Lapiz	Bic	5.00	
53207	Pluma	Bic	10.00	
53208	Goma	Mapeed	5.00	
53209	Sacapuntas	Mapeed	5.00	
53210	Colores	Mapeed	20.00	
53211	Marshall	Marshall	40.00	


Vista de Herramientas

 **Papel & Lápiz**

 [Tienda](#)

 [Herramientas](#)

 [Registrar Cliente](#)

 [Carrito](#)

Utilidad

Obtenga la utilidad de un producto.

[Ir](#)

Análisis de ventas

Obtenga las ganancias de las ventas realizadas.

[Ir](#)

Revisión de inventario

Conozca los productos que estan por agotarse.


[Ir](#)





Facturas

Obtenga la información de una factura.

[Ir](#)

Herramientas: Utilidad de un Producto


 **Papel & Lápiz**

 [Tienda](#)  [Herramientas](#)  [Registrar Cliente](#)  [Carrito](#)

Utilidad de un producto

Ingrese el código de barras de un producto para obtener su utilidad.


Código de Barras

 **Calcular**

Resumen

Código	Descripción	Utilidad
		\$

Herramientas: Utilidad de un Producto

 **Papel & Lápiz**

Tienda

Herramientas

Registrar Cliente

Carrito

Utilidad de un producto

Ingrese el código de barras de un producto para obtener su utilidad.


Código de Barras

Calcular

Resumen

Código	Descripción	Utilidad
53204	Impresion B/N tamaÑo oficio	\$2.00

Herramientas: Análisis de Ventas

 **Papel & Lápiz**

Tienda

Herramientas


Registrar Cliente

Carrito


Análisis de ventas

Ingrese una fecha o una fecha de inicio y una fecha de fin.

Fecha de inicio



Fecha de fin




Calcular

Resumen

Periodo de tiempo	Ganancia total
	\$

Análisis de Ventas: Una fecha

 Papel & Lápiz

TiendaHerramientasRegistrar Clientecarrito

Análisis de ventas

Ingrese una fecha o una fecha de inicio y una fecha de fin.

Fecha de inicio

mm/dd/yyyy

Fecha de fin

mm/dd/yyyy

Calcular

Resumen

Periodo de tiempo	Ganancia total
2021-12-11	\$9500.50

Análisis de Ventas: Dos fechas



Papel & Lápiz

Tienda

Herramientas

Registrar Cliente

Carrito

Análisis de ventas

Ingrese una fecha o una fecha de inicio y una fecha de fin.

Fecha de inicio

mm/dd/yyyy



Fecha de fin

mm/dd/yyyy



Calcular

Resumen

Periodo de tiempo	Ganancia total
2021-12-09 2021-12-12	\$10482.50

Herramientas: Revisión del Inventario

 **Papel & Lápiz**

 [Tienda](#)  [Herramientas](#)  [Registrar Cliente](#)  [Carrito](#)

Revisión de inventario

Productos que estan por agotarse:

Descripción	Marca	Cantidad
Colores	Mapeed	0

Factura: Formulario Previo

 **Papel & Lápiz**

TiendaHerramientasRegistrar Clientecarrito

Busqueda por venta

Ingrese el número de la factura:

Num. Factura

RegresarSiguiente

Herramientas: Factura

Papel & Lápiz

Tienda

Herramientas

Registrar Cliente

Carrito

Factura

Papel & Lápiz.

FACTURAR A:

Nombre: Perez Lopez

RFC: GEC8501401411

Dirección: La Mision, Guerrero. CP 6300

FACTURA: FACT-15

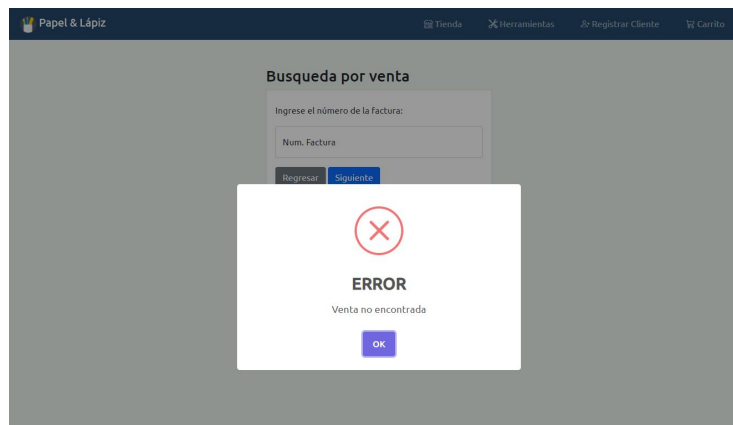
FECHA: 12 de Diciembre de 2021

Cantidad	Descripción	Precio Unitario	Precio Importe
1	Impresion a color tamaÃ±o carta	5.00	5.00
1	Impresion B/N tamaÃ±o carta	1.00	1.00
1	Impresion a color tamaÃ±o oficio	10.00	10.00
		Total	16.00

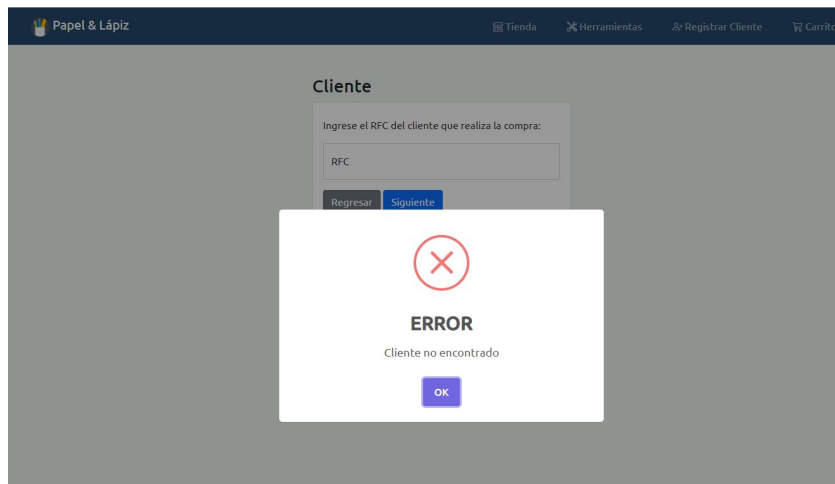
Finalizar

Vistas de Error relacionadas con la Factura


Desde Herramientas:



Desde el proceso de compra:



Formulario para el registro de un cliente

 **Papel & Lápiz**

Tienda

Herramientas

Registrar Cliente

Carrito

Registro de cliente

Nombre	Apellido Paterno	Apellido Materno
RFC	Correo	
Calle	Número	Código postal
Colonia	Estado	

Registrar

Error al crear al cliente

Papel & Lápiz

TiendaHerramientasRegistrar ClienteCarrito

Registro de cliente

Nombre

Apellido Paterno

Apellido Materno

RFC

Correo

Calle

Código postal

Colonia


Registrar




ERROR AL INSERTAR UN CLIENTE

Hubo un error al momento de insertar un cliente, por favor revisa los campos de RFC o correo

OK

Carrito de venta: Vista inicial


 **Papel & Lápiz**

 [Tienda](#)  [Herramientas](#)  [Registrar Cliente](#)  [Carrito](#)

Mi Carrito




Código	Descripción	Precio Unitario	Cantidad	Precio Total	Eliminar del carrito
<div><div>Limpiar carrito</div><div>Siguiente</div></div>					

Carrito de venta: Con artículos

 **Papel & Lápiz**

TiendaHerramientasRegistrar Clientecarrito

Mi Carrito

Código	Descripción	Precio Unitario	Cantidad	Precio Total	Eliminar del carrito
53202	Impresion B/N tamaño carta	\$1.0	1	\$1.0	
53207	Pluma	\$10.0	8	\$80.0	
53211	Max Steel	\$49.99	3	\$149.97	

Limpiar carrito

Siguiente

Comunicación con la Base de datos

El primer paso, y el más importante, es definir los parámetros de la base de datos a la que nos vamos a conectar. Esto se realiza en el archivo *settings.py* de nuestro proyecto en Django.

```
DATABASES = {  
    'default': {  
        'ENGINE': 'django.db.backends.postgresql_psycopg2',  
        'NAME': 'pp2',  
        'USER': 'kevin',  
        'PASSWORD': '*****',  
        'HOST': '127.0.0.1',  
        'DATABASE_PORT': '5432'  
    }  
}
```

Comunicación con la Base de datos

Para ejecutar las sentencias SQL directamente en la base de datos, se utilizó el objeto *cursor*, obtenido de la conexión de Django con la base de datos.

```
connection = psycopg2.connect("...")

from django.db import connection

with connection.cursor() as cursor:
    cursor.execute('SQL Query')
    response = cursor.fetchone()
    response = cursor.fetchall()
```

Tienda

```
def tienda(request, cat):  
  
    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()
```

Registro Cliente

```
def validaRegistroCliente(request):  
  
    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')
```

Venta

```
def venta(request, rfc):

    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():
            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])
```

Utilidad

```
def calculaUtilidad(request):  
  
    with connection.cursor() as cursor:  
        cursor.execute("SELECT retorna_Utilidad(%s);", [codigo])  
        consultaSQL = cursor.fetchone()
```

Análisis Ventas

```
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')
```


Revision Inventario

```
def revisionInventario(request):  
  
    with connection.cursor() as cursor:  
        cursor.execute("SELECT menos_Thr();")  
        consultaSQL = cursor.fetchall()
```

Factura

```
def factura(request, idVenta):  
  
    with connection.cursor() as cursor:  
        cursor.execute('SELECT * FROM FACTURA WHERE id_venta = %s;', [idVenta])  
        consulta = cursor.fetchall()
```

Gracias!

De parte de JADKI agradecemos su atención.