1ra Retrasada, Gestion de facturación*

Kevin Juergen Francisco Hernández Hernández, 201904037[1]

[1] *Facultad de Ingeniería, Escuela de Mecánica Eléctrica, Universidad de San Carlos de Guatemala, Edificio T1, Ciudad Universitaria, Zona 12, Guatemala.*

En este reporte de remarca y se observa el desarrollo de la tarea de 1era retrasada el cual consiste en desarrollar un sistema de gestion de facturacion para una tienda de productos electronicos el cual debe contar con una base de datos tanto para los clientes registrados como de los productos que se ofrecen, ademas de almacenar en ella los registros de facturacion.

I. MARCO TEORICO

A. Visual Studio Code

Visual Studio Code (VS Code) es un editor de código fuente desarrollado por Microsoft que ha ganado mucha popularidad entre desarrolladores de diferentes lenguajes y plataformas debido a su flexibilidad y capacidad de personalización.VS Code es muy valorado por su flexibilidad y extensibilidad, lo que lo hace adecuado tanto para desarrolladores novatos como experimentados.

Simbolo de Visual studio code

B. Python

Python es un lenguaje de programación conocido por su simplicidad, legibilidad y versatilidad. Esto significa que, a diferencia de otros lenguajes más complicados, incluso un principiante puede empezar a programar rápidamente con Python.
Python fue concebido con la idea de crear un lenguaje que fuera fácil de leer y de escribir, permitiendo a los desarrolladores expresar conceptos en menos líneas de código que lenguajes como C++ o Java. Además, se buscó que fuese un lenguaje extensible y que pudiera ser embebido en otras aplicaciones.

Simbolo de Python

C. PostgreSQL

PostgreSQL, comúnmente pronunciado "Post-GRES", es una base de datos de código abierto que tiene una sólida reputación por su fiabilidad, flexibilidad y soporte de estándares técnicos abiertos. A diferencia de otros RDMBS (sistemas de gestión de bases de datos relacionales), PostgreSQL (enlace externo a ibm.com) soporta tipos de datos relacionales y no relacionales. Esto la convierte en una de las bases de datos relacionales más compatibles, estables y maduras disponibles actualmente.

Logo de PostgreSQL

II. RESULTADOS

Codigo No.1
Codigo del Programa en Python

```
import psycopg2
import os
import re
```

* Proyectos de Computación Aplicados a Ingenieria Electronica, Sección A

```python
from datetime import datetime

# Configuraci n de la base de datos
DB_CONFIG = {
    'host': 'localhost',
    'database': 'retra1',
    'user': 'postgres',
    'password': 'hidrogeno',
    'port': '5432'
}

# Expresiones regulares para validaci n
NOMBRE_REGEX = re.compile(r'^[a-zA-Z \s]{2,50}$')
EMAIL_REGEX = re.compile(r'^[a-zA-Z0-9_.+-]+@[a-zA-Z0-9-]+\.[a-zA-Z0-9-.]+$')

def conectar_db():
    """Establece conexi n con la base de datos PostgreSQL"""
    try:
        conn = psycopg2.connect(**DB_CONFIG)
        return conn
    except Exception as e:
        print(f"\n[!] Error de conexi n a la base de datos: {e}")
        return None

def validar_nombre(nombre):
    """Valida que el nombre solo contenga letras y espacios"""
    if not NOMBRE_REGEX.match(nombre):
        print("[!] Nombre inv lido. Solo debe contener letras y espacios (2-50 caracteres)")
        return False
    return True

def validar_email(email):
    """Valida el formato b sico de un email"""
    if not EMAIL_REGEX.match(email):
        print("[!] Email inv lido. Formato debe ser usuario@dominio.com")
        return False
    return True

def input_numerico(mensaje, min_val=0, max_val=None):
    """Solicita y valida un valor num rico"""
    while True:
        valor = input(mensaje).strip()
        if valor.isdigit():
            num = int(valor)
            if min_val is not None and num < min_val:
                print(f"[!] El valor debe ser mayor o igual a {min_val}")
                continue
            if max_val is not None and num > max_val:
                print(f"[!] El valor debe ser menor o igual a {max_val}")
                continue
            return num
        print("[!] Entrada inv lida. Debe ser un n mero entero")

def registrar_cliente():
    """Registra un nuevo cliente o recupera existente"""
    print("\n--- REGISTRO DE CLIENTE ---")

    # Validar nombre
    while True:
        nombre = input("Nombre del cliente: ").strip()
        if validar_nombre(nombre):
            break

    # Validar email
    while True:
        email = input("Email del cliente: ").strip().lower()
        if validar_email(email):
            break

    conn = conectar_db()
    if conn is None:
        return None

    try:
        cursor = conn.cursor()
        # Verificar si el cliente existe
        cursor.execute(
            "SELECT id FROM clientes WHERE nombre_cliente = %s OR email = %s",
            (nombre, email)
        )
        cliente_existente = cursor.fetchone()

        if cliente_existente:
            print("\n[   ] Cliente ya registrado. Usando datos existentes")
            return cliente_existente[0]
        else:
            # Insertar nuevo cliente
            cursor.execute(
                "INSERT INTO clientes (nombre_cliente, email) VALUES (%s, %s) RETURNING id",
                (nombre, email)
            )
            cliente_id = cursor.fetchone()[0]
            conn.commit()
            print("\n[   ] Nuevo cliente registrado exitosamente")
            return cliente_id
    except Exception as e:
        print(f"\n[!] Error al registrar cliente: {e}")
        conn.rollback()
        return None
    finally:
        conn.close()

def mostrar_productos():
    """Muestra todos los productos disponibles"""
    conn = conectar_db()
    if conn is None:
        return []

    try:
        cursor = conn.cursor()
        cursor.execute("SELECT id, nombre_producto, precio_unitario FROM productos")
        productos = cursor.fetchall()

        if not productos:
```

```
            print("\n[!] No hay productos disponibles en el inventario")
            return []

        print("\n--- PRODUCTOS DISPONIBLES ---")
        print("ID  | Nombre".ljust(35) + " | Precio Unitario")
        print("-" * 55)
        for prod in productos:
            print(f"{str(prod[0]).ljust(3)} | {prod[1].ljust(30)} | Q{prod[2]:.2f}")

        return productos
    except Exception as e:
        print(f"\n[!] Error al cargar productos: {e}")
        return []
    finally:
        conn.close()

def seleccionar_productos():
    """Permite al usuario seleccionar productos y cantidades"""
    productos = mostrar_productos()
    if not productos:
        return []

    seleccionados = []
    ids_disponibles = [str(p[0]) for p in productos]

    while True:
        print("\nIngrese ID del producto (0 para finalizar)")
        prod_id = input("> ").strip()

        if prod_id == '0':
            if seleccionados:
                break
            else:
                print("[!] Debe seleccionar al menos un producto")
                continue

        # Validar ID num rico
        if not prod_id.isdigit():
            print("[!] ID debe ser un n mero. Intente nuevamente")
            continue

        prod_id = int(prod_id)

        # Verificar existencia del producto
        producto = next((p for p in productos if p[0] == prod_id), None)
        if not producto:
            print("[!] ID de producto inv lido")
            continue

        # Validar cantidad
        cantidad = input_numerico(
            f"Cantidad para '{producto[1]}': ",
            min_val=1,
            max_val=1000
        )

        subtotal = cantidad * producto[2]
        seleccionados.append({
            'id': producto[0],
            'nombre': producto[1],
            'precio': producto[2],
            'cantidad': cantidad,
            'subtotal': subtotal
        })

        print(f"[   ] Agregado: {cantidad} x {producto[1]} = Q{subtotal:.2f}")

    return seleccionados

def generar_factura(cliente_id, productos):
    """Guarda la factura en la base de datos y genera archivo"""
    if not productos:
        print("\n[!] Operaci n cancelada. No hay productos para facturar")
        return

    conn = conectar_db()
    if conn is None:
        return

    try:
        cursor = conn.cursor()

        # Obtener datos del cliente
        cursor.execute("SELECT nombre_cliente, email FROM clientes WHERE id = %s", (cliente_id,))
        cliente = cursor.fetchone()

        if not cliente:
            print("\n[!] Cliente no encontrado en la base de datos")
            return

        nombre_cliente, email_cliente = cliente
        total = sum(p['subtotal'] for p in productos)

        # Insertar factura
        cursor.execute(
            "INSERT INTO facturas (cliente_id, total) VALUES (%s, %s) RETURNING id",
            (cliente_id, total)
        )
        factura_id = cursor.fetchone()[0]

        # Insertar detalles
        for prod in productos:
            cursor.execute(
                "INSERT INTO detalles_factura (factura_id, producto_id, cantidad, subtotal) "
                "VALUES (%s, %s, %s, %s)",
                (factura_id, prod['id'], prod['cantidad'], prod['subtotal'])
            )

        conn.commit()

        # Mostrar resumen
        print("\n--- RESUMEN DE FACTURA ---")
        print(f"Cliente: {nombre_cliente}")
        print(f"Email: {email_cliente}")
        print("\nProductos:")
        for prod in productos:
            print(f"- {prod['nombre']}: {prod['cantidad']} x Q{prod['precio']:.2f} = Q{prod
```

```
    ['subtotal']:.2f}")
        print("-" * 40)
        print(f"TOTAL A PAGAR: Q{total:.2f}")
        print("-" * 40)

        # Generar archivo de texto
        guardar_factura_txt(factura_id,
    nombre_cliente, email_cliente, productos,
    total)

    except Exception as e:
        print(f"\n[!] Error al generar factura:
    {e}")
        conn.rollback()
    finally:
        conn.close()

def guardar_factura_txt(factura_id, cliente,
    email, productos, total):
    """Guarda la factura en un archivo de texto"
    ""
    try:
        fecha = datetime.now().strftime("%d/%m/%
    Y %H:%M:%S")
        nombre_archivo = "factura.txt"
        modo = 'a' if os.path.exists(
    nombre_archivo) else 'w'

        with open(nombre_archivo, modo, encoding
    ='utf-8') as f:
            f.write("\n" + "=" * 60 + "\n")
            f.write(f"FACTURA ELECTR NICA -
    RETRA1\n")
            f.write("=" * 60 + "\n")
            f.write(f"ID Factura: {factura_id}\n
    ")
            f.write(f"Fecha: {fecha}\n")
            f.write(f"Cliente: {cliente}\n")
            f.write(f"Email: {email}\n")
            f.write("-" * 60 + "\n")
            f.write("DETALLE DE COMPRA:\n\n")
            f.write("{:<5} {:<30} {:<10} {:<10}
    {:<10}\n".format(
                "CANT", "PRODUCTO", "P. UNIT", "
    SUBTOTAL", "ID"))
            f.write("-" * 60 + "\n")

            for prod in productos:
                f.write("{:<5} {:<30} Q{:<9.2f}
    Q{:<9.2f} {:<10}\n".format(
                    prod['cantidad'],
                    prod['nombre'],
                    prod['precio'],
                    prod['subtotal'],
                    prod['id']))

            f.write("-" * 60 + "\n")
            f.write(f"TOTAL: Q{total:.2f}\n")
            f.write("=" * 60 + "\n\n")

        print(f"\n[   ] Factura guardada en '{
    nombre_archivo}'")
    except Exception as e:
        print(f"\n[!] Error al guardar archivo:
    {e}")

def consultar_facturas():
    """Consulta facturas anteriores por email de
     cliente"""
    print("\n--- CONSULTA DE FACTURAS ---")

    # Validar email
    while True:
        email = input("Ingrese email del cliente
    : ").strip().lower()
        if validar_email(email):
            break

    conn = conectar_db()
    if conn is None:
        return

    try:
        cursor = conn.cursor()
        # Obtener cliente
        cursor.execute(
            "SELECT id, nombre_cliente FROM
    clientes WHERE email = %s",
            (email,))
        cliente = cursor.fetchone()

        if not cliente:
            print("\n[!] Cliente no encontrado")
            return

        cliente_id, nombre_cliente = cliente

        # Obtener facturas del cliente
        cursor.execute(
            "SELECT id, fecha, total FROM
    facturas WHERE cliente_id = %s ORDER BY
    fecha DESC",
            (cliente_id,))
        facturas = cursor.fetchall()

        if not facturas:
            print("\n[!] No se encontraron
    facturas para este cliente")
            return

        print(f"\nFacturas de {nombre_cliente}
    ({email}):")
        print("-" * 50)
        for fact in facturas:
            print(f"Factura #{fact[0]} - {fact
    [1].strftime('%d/%m/%Y')} - Total: Q{fact
    [2]:.2f}")

        # Detalle de una factura espec fica
        fact_id = input("\nIngrese ID de factura
     para detalles (0 para salir): ").strip()
        if fact_id == '0':
            return

        # Validar ID num rico
        if not fact_id.isdigit():
            print("[!] ID de factura debe ser un
     n mero")
            return

        fact_id = int(fact_id)

        if fact_id not in [f[0] for f in
    facturas]:
            print("[!] ID de factura inv lido")
            return

        cursor.execute(
            "SELECT p.nombre_producto,
    d.cantidad, d.subtotal, d.producto_id "
```

```
            "FROM detalles_factura d "
            "JOIN productos p ON d.producto_id = p.id "
            "WHERE d.factura_id = %s",
            (fact_id,))
        detalles = cursor.fetchall()

        print(f"\nDETALLE FACTURA #{fact_id}")
        print("-" * 60)
        for det in detalles:
            print(f"{det[1]} x {det[0]} (ID: {det[3]}) = Q{det[2]:.2f}")
        print("-" * 60)

    except Exception as e:
        print(f"\n[!] Error en consulta: {e}")
    finally:
        conn.close()

def menu_principal():
    """Muestra el men  principal del sistema"""
    while True:
        print("\n--- SISTEMA DE FACTURACI N RETRA1 ---")
        print("1. Registrar nuevo cliente")
        print("2. Crear nueva factura")
        print("3. Consultar facturas existentes")
        print("4. Salir")

        opcion = input("\nSeleccione una opci n: ").strip()

        if opcion == '1':
            registrar_cliente()
        elif opcion == '2':
            cliente_id = registrar_cliente()
            if cliente_id:
                productos = seleccionar_productos()
                if productos:
                    generar_factura(cliente_id, productos)
        elif opcion == '3':
            consultar_facturas()
        elif opcion == '4':
            print("\n[   ] Sistema cerrado. Hasta  pronto!")
            break
        else:
            print("\n[!] Opci n inv lida. Intente nuevamente")

if __name__ == "__main__":
    print("\n" + "=" * 50)
    print("SISTEMA DE FACTURACI N PARA TIENDA DE ELECTR NICA")
    print("=" * 50)
    menu_principal()
```

codigo p1.jpg

Fuente: Elaboración propia, 2025

codigo p2.jpg

Fuente: Elaboración propia, 2025

codigo p3.jpg

Fuente: Elaboración propia, 2025

codigo p4.jpg

Fuente: Elaboración propia, 2025

Figura No.2
LLamada del programa desde la ventana de comandos ejecutandose y dezplegando su menu.

menu.jpg

Fuente: Elaboración propia, 2025

Figura No.3
Programa ejecutandose, desde opción 1 - 2.

funcionamiento p1.jpg

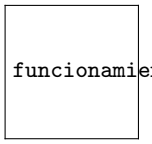
Fuente: Elaboración propia, 2025

Figura No.4
Programa ejecutandose, desde opción 3
Mostrar el historial de datos de base de datos.

funcionamiento p2.jpg

Fuente: Elaboración propia, 2025

Figura No.5
Programa ejecutandose, desde opción 3 - 5.



Fuente: Elaboración propia, 2025

Figura No.6
Base de datos desde PGAdmin4.

Fuente: Elaboración propia, 2025

Figura No.7
Archivo de texto donde se almacenan los datos.

Fuente: Elaboración propia, 2025

III. CONCLUSIONES

1. Python es un lenguaje de programacion de alto nivel el cual nos permite realizar una gran variedad de codigos ya que sus librerias nos permiten ahorrar codigo a la hora de programar.

2. Postgresql es una gran herramienta, ya que es un sofware, que junto a su administrador visual PGAdmin nos permiten gestionar las bases de datos de una manera mas comoda y mas entendible, ademas de contar con una gran cantidad de herramientas de ayuda.

3. Al emplear tanto python y postgresql en un proyeto nos permite tener una buena gestion de las base de datos con postgres y el manejo dinamico de Python313.

IV. REPOSITORIO

Enlace Repositorio: `https://github.com/KevinJ-1506/PCAIE/tree/0fa3b558faa96d4fd930eec425b8eab890b876b8/retra1`

V. VIDEO

Enlace de drive: `https://drive.google.com/file/d/1FGs8FddjgkOaK5CwWdvkFup9XnNzOQuP/view?usp=sharing`

[1] PALACIN, Alfredo Borque. El osciloscopio. Electronica y comunicaciones, 2008, vol. 28, no 233, p. 32-35.

[2] COOPER, William. Instrumentación electronica moderna y tecnicas de medicion. Mexico: Prentice Hall, 1991.

[3] Ruiz, G. Electronica basica para ingenieros. España: Servicio de Reprografía, Facultad de Ciencias, Universidad de Cantabria, 2001.

[4] ON Semiconductor. (n.d.). *P2N2222A: NPN general purpose transistor* [Data sheet]. Retrieved August 14, 2024, from https://www.onsemi.com/pdf/datasheet/p2n2222a-d.pdf

[5] Vishay. (n.d.). *NTC thermistors* [Data sheet]. Retrieved August 14, 2024, from https://www.rinconingenieril.es/wp-content/uploads/2017/08/NTC-Vishay.pdf