

1ra Retrasada, Gestion de facturación*

Kevin Juergen Francisco Hernández Hernández, 201904037¹

¹Facultad 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.



Visual Studio Code

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

```
1 import psycpg2
2 import os
3 import re
```

* Proyectos de Computación Aplicados a Ingeniería Electronica, Sección A

```

4 from datetime import datetime
5
6 # Configuraci n de la base de datos
7 DB_CONFIG = {
8     'host': 'localhost',
9     'database': 'retrial',
10    'user': 'postgres',
11    'password': 'hidrogeno',
12    'port': '5432'
13 }
14
15 # Expresiones regulares para validaci n
16 NOMBRE_REGEX = re.compile(r'^[a-zA-Z\s]{2,50}$')
17 EMAIL_REGEX = re.compile(r'^[a-zA-Z0-9_+-.]+@[a-zA-Z0-9-]+\.[a-zA-Z0-9-]+\.$')
18
19 def conectar_db():
20     """Establece conexi n con la base de datos PostgreSQL"""
21     try:
22         conn = psycopg2.connect(**DB_CONFIG)
23         return conn
24     except Exception as e:
25         print(f"\n[!] Error de conexi n a la base de datos: {e}")
26         return None
27
28 def validar_nombre(nombre):
29     """Valida que el nombre solo contenga letras y espacios"""
30     if not NOMBRE_REGEX.match(nombre):
31         print("[!] Nombre inv lido. Solo debe contener letras y espacios (2-50 caracteres)")
32         return False
33     return True
34
35 def validar_email(email):
36     """Valida el formato b sico de un email"""
37     if not EMAIL_REGEX.match(email):
38         print("[!] Email inv lido. Formato debe ser usuario@dominio.com")
39         return False
40     return True
41
42 def input_numerico(mensaje, min_val=0, max_val=None):
43     """Solicita y valida un valor num rico"""
44     while True:
45         valor = input(mensaje).strip()
46         if valor.isdigit():
47             num = int(valor)
48             if min_val is not None and num < min_val:
49                 print(f"[!] El valor debe ser mayor o igual a {min_val}")
50                 continue
51             if max_val is not None and num > max_val:
52                 print(f"[!] El valor debe ser menor o igual a {max_val}")
53                 continue
54             return num
55         print("[!] Entrada inv lida. Debe ser un n mero entero")
56
57 def registrar_cliente():
58     """Registra un nuevo cliente o recupera existente"""
59
60     print("\n--- REGISTRO DE CLIENTE ---")
61
62     # Validar nombre
63     while True:
64         nombre = input("Nombre del cliente: ")
65         if validar_nombre(nombre):
66             break
67
68     # Validar email
69     while True:
70         email = input("Email del cliente: ")
71         if validar_email(email):
72             break
73
74     conn = conectar_db()
75     if conn is None:
76         return None
77
78     try:
79         cursor = conn.cursor()
80         # Verificar si el cliente existe
81         cursor.execute(
82             "SELECT id FROM clientes WHERE nombre_cliente = %s OR email = %s",
83             (nombre, email)
84         )
85         cliente_existente = cursor.fetchone()
86
87         if cliente_existente:
88             print("\n[ ] Cliente ya registrado. Usando datos existentes")
89             return cliente_existente[0]
90         else:
91             # Insertar nuevo cliente
92             cursor.execute(
93                 "INSERT INTO clientes (nombre_cliente, email) VALUES (%s, %s) RETURNING id",
94                 (nombre, email)
95             )
96             cliente_id = cursor.fetchone()[0]
97             conn.commit()
98             print("\n[ ] Nuevo cliente registrado exitosamente")
99             return cliente_id
100     except Exception as e:
101         print(f"\n[!] Error al registrar cliente : {e}")
102         conn.rollback()
103         return None
104     finally:
105         conn.close()
106
107 def mostrar_productos():
108     """Muestra todos los productos disponibles"""
109
110     conn = conectar_db()
111     if conn is None:
112         return []
113
114     try:
115         cursor = conn.cursor()
116         cursor.execute("SELECT id, nombre_producto, precio_unitario FROM productos")
117         productos = cursor.fetchall()
118
119         if not productos:

```

```

118         print("\n[!] No hay productos
disponibles en el inventario")
119         return []
120
121         print("\n--- PRODUCTOS DISPONIBLES ---")
122         print("ID | Nombre".ljust(35) + " |
Precio Unitario")
123         print("-" * 55)
124         for prod in productos:
125             print(f"{str(prod[0]).ljust(3)} | {
prod[1].ljust(30)} | Q{prod[2]:.2f}")
126
127         return productos
128     except Exception as e:
129         print(f"\n[!] Error al cargar productos:
{e}")
130         return []
131     finally:
132         conn.close()
133
134 def seleccionar_productos():
135     """Permite al usuario seleccionar productos
y cantidades"""
136     productos = mostrar_productos()
137     if not productos:
138         return []
139
140     seleccionados = []
141     ids_disponibles = [str(p[0]) for p in
productos]
142
143     while True:
144         print("\nIngrese ID del producto (0 para
finalizar)")
145         prod_id = input("> ").strip()
146
147         if prod_id == '0':
148             if seleccionados:
149                 break
150             else:
151                 print("[!] Debe seleccionar al
menos un producto")
152                 continue
153
154         # Validar ID num rico
155         if not prod_id.isdigit():
156             print("[!] ID debe ser un n mero.
Intente nuevamente")
157             continue
158
159         prod_id = int(prod_id)
160
161         # Verificar existencia del producto
162         producto = next((p for p in productos if
p[0] == prod_id), None)
163         if not producto:
164             print("[!] ID de producto inv lido")
165
166         continue
167
168         # Validar cantidad
169         cantidad = input_numerico(
170             f"Cantidad para '{producto[1]}': ",
171             min_val=1,
172             max_val=1000
173         )
174
175         subtotal = cantidad * producto[2]
176         seleccionados.append({
177             'id': producto[0],
178             'nombre': producto[1],
179             'precio': producto[2],
180             'cantidad': cantidad,
181             'subtotal': subtotal
182         })
183
184         print(f"[ ] Agregado: {cantidad} x {
producto[1]} = Q{subtotal:.2f}")
185
186     return seleccionados
187
188 def generar_factura(cliente_id, productos):
189     """Guarda la factura en la base de datos y
genera archivo"""
190     if not productos:
191         print("\n[!] Operaci n cancelada. No
hay productos para facturar")
192         return
193
194     conn = conectar_db()
195     if conn is None:
196         return
197
198     try:
199         cursor = conn.cursor()
200
201         # Obtener datos del cliente
202         cursor.execute("SELECT nombre_cliente,
email FROM clientes WHERE id = %s", (
203             cliente_id,))
204         cliente = cursor.fetchone()
205
206         if not cliente:
207             print("\n[!] Cliente no encontrado
en la base de datos")
208             return
209
210         nombre_cliente, email_cliente = cliente
211         total = sum(p['subtotal'] for p in
productos)
212
213         # Insertar factura
214         cursor.execute(
215             "INSERT INTO facturas (cliente_id,
total) VALUES (%s, %s) RETURNING id",
216             (cliente_id, total)
217         )
218         factura_id = cursor.fetchone()[0]
219
220         # Insertar detalles
221         for prod in productos:
222             cursor.execute(
223                 "INSERT INTO detalles_factura (
factura_id, producto_id, cantidad, subtotal)
VALUES (%s, %s, %s, %s)",
224                 (factura_id, prod['id'], prod['
cantidad'], prod['subtotal'])
225             )
226
227         conn.commit()
228
229         # Mostrar resumen
230         print("\n--- RESUMEN DE FACTURA ---")
231         print(f"Cliente: {nombre_cliente}")
232         print(f>Email: {email_cliente}")
233         print("\nProductos:")
234         for prod in productos:
235             print(f"- {prod['nombre']}: {prod['
cantidad']} x Q{prod['precio']:.2f} = Q{prod[

```

```

235         ['subtotal':.2f}"])
236         print("-" * 40)
237         print(f"TOTAL A PAGAR: Q{total:.2f}")
238         print("-" * 40)
239
240         # Generar archivo de texto
241         guardar_factura_txt(factura_id,
242                             nombre_cliente, email_cliente, productos,
243                             total)
244
245     except Exception as e:
246         print(f"\n[!] Error al generar factura:
247         {e}")
248         conn.rollback()
249     finally:
250         conn.close()
251
252 def guardar_factura_txt(factura_id, cliente,
253                         email, productos, total):
254     """Guarda la factura en un archivo de texto"""
255
256     try:
257         fecha = datetime.now().strftime("%d/%m/%
258         Y %H:%M:%S")
259         nombre_archivo = "factura.txt"
260         modo = 'a' if os.path.exists(
261             nombre_archivo) else 'w'
262
263         with open(nombre_archivo, modo, encoding
264                   ='utf-8') as f:
265             f.write("\n" + "=" * 60 + "\n")
266             f.write(f"FACTURA ELECTR NICA -
267             RETRA1\n")
268             f.write("=" * 60 + "\n")
269             f.write(f"ID Factura: {factura_id}\n
270             ")
271             f.write(f"Fecha: {fecha}\n")
272             f.write(f"Cliente: {cliente}\n")
273             f.write(f"Email: {email}\n")
274             f.write("-" * 60 + "\n")
275             f.write("DETALLE DE COMPRA:\n\n")
276             f.write("{:<5} {:<30} {:<10} {:<10}
277             {:<10}\n".format(
278                 "CANT", "PRODUCTO", "P. UNIT", "
279                 SUBTOTAL", "ID"))
280             f.write("-" * 60 + "\n")
281
282             for prod in productos:
283                 f.write("{:<5} {:<30} Q{:<9.2f}
284                 Q{:<9.2f} {:<10}\n".format(
285                     prod['cantidad'],
286                     prod['nombre'],
287                     prod['precio'],
288                     prod['subtotal'],
289                     prod['id']))
290
291             f.write("-" * 60 + "\n")
292             f.write(f"TOTAL: Q{total:.2f}\n")
293             f.write("=" * 60 + "\n\n")
294
295             print(f"\n[ ] Factura guardada en '{
296             nombre_archivo}'")
297     except Exception as e:
298         print(f"\n[!] Error al guardar archivo:
299         {e}")
300
301 def consultar_facturas():
302     """Consulta facturas anteriores por email de
303     cliente"""
304     print("\n--- CONSULTA DE FACTURAS ---")
305
306     # Validar email
307     while True:
308         email = input("Ingreso email del cliente
309         : ").strip().lower()
310         if validar_email(email):
311             break
312
313     conn = conectar_db()
314     if conn is None:
315         return
316
317     try:
318         cursor = conn.cursor()
319         # Obtener cliente
320         cursor.execute(
321             "SELECT id, nombre_cliente FROM
322             clientes WHERE email = %s",
323             (email,))
324         cliente = cursor.fetchone()
325
326         if not cliente:
327             print("\n[!] Cliente no encontrado")
328             return
329
330         cliente_id, nombre_cliente = cliente
331
332         # Obtener facturas del cliente
333         cursor.execute(
334             "SELECT id, fecha, total FROM
335             facturas WHERE cliente_id = %s ORDER BY
336             fecha DESC",
337             (cliente_id,))
338         facturas = cursor.fetchall()
339
340         if not facturas:
341             print("\n[!] No se encontraron
342             facturas para este cliente")
343             return
344
345         print(f"\nFacturas de {nombre_cliente}
346         ({email}):")
347         print("-" * 50)
348         for fact in facturas:
349             print(f"Factura #{fact[0]} - {fact
350             [1].strftime('%d/%m/%Y')} - Total: Q{fact
351             [2]:.2f}")
352
353         # Detalle de una factura espec fica
354         fact_id = input("\nIngreso ID de factura
355         para detalles (0 para salir): ").strip()
356         if fact_id == '0':
357             return
358
359         # Validar ID num rico
360         if not fact_id.isdigit():
361             print("[!] ID de factura debe ser un
362             n mero")
363             return
364
365         fact_id = int(fact_id)
366
367         if fact_id not in [f[0] for f in
368         facturas]:
369             print("[!] ID de factura inv lido")
370             return
371
372         cursor.execute(
373             "SELECT p.nombre_producto,
374             d.cantidad, d.subtotal, d.producto_id ")

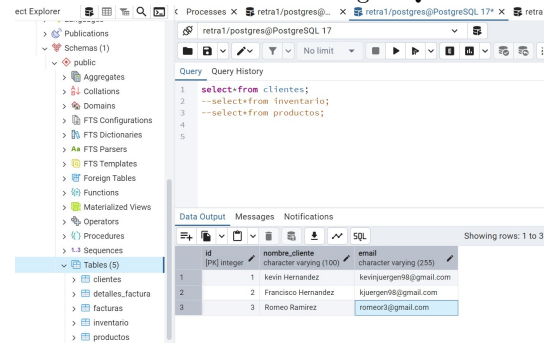
```

```

346         "FROM detalles_factura d "
347         "JOIN productos p ON d.producto_id =
           p.id "
348         "WHERE d.factura_id = %s",
349         (fact_id,))
350         detalles = cursor.fetchall()
351
352         print(f"\nDETALLE FACTURA #{fact_id}")
353         print("-" * 60)
354         for det in detalles:
355             print(f"{det[1]} x {det[0]} (ID: {
det[3]}) = Q{det[2]:.2f}")
356             print("-" * 60)
357
358     except Exception as e:
359         print(f"\n[!] Error en consulta: {e}")
360     finally:
361         conn.close()
362
363 def menu_principal():
364     """Muestra el men principal del sistema"""
365     while True:
366         print("\n--- SISTEMA DE FACTURACI N
RETRA1 ---")
367         print("1. Registrar nuevo cliente")
368         print("2. Crear nueva factura")
369         print("3. Consultar facturas existentes")
370     )
371         print("4. Salir")
372
373         opcion = input("\nSeleccione una opci n
: ").strip()
374
375         if opcion == '1':
376             registrar_cliente()
377         elif opcion == '2':
378             cliente_id = registrar_cliente()
379             if cliente_id:
380                 productos =
seleccionar_productos()
381                 if productos:
382                     generar_factura(cliente_id,
productos)
383             elif opcion == '3':
384                 consultar_facturas()
385             elif opcion == '4':
386                 print("\n[ ] Sistema cerrado.
Hasta pronto!")
387                 break
388             else:
389                 print("\n[!] Opci n inv lida.
Intente nuevamente")
390
391 if __name__ == "__main__":
392     print("\n" + "=" * 50)
393     print("SISTEMA DE FACTURACI N PARA TIENDA
DE ELECTR NICA")
394     print("=" * 50)
395     menu_principal()

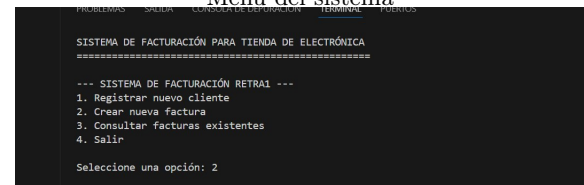
```

Figura No.1
Base de datos en PostgreSQL



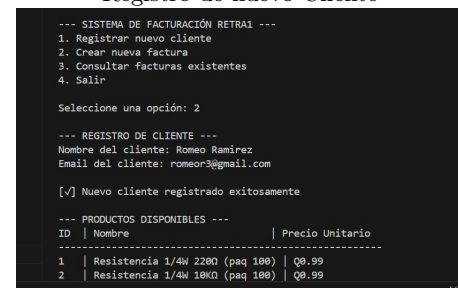
Fuente: Elaboración propia, 2025

Figura No.2
Menu del sistema



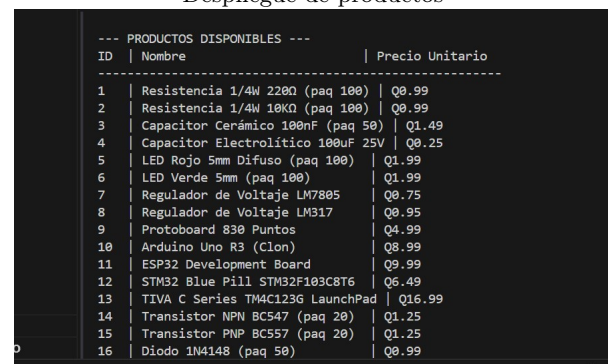
Fuente: Elaboración propia, 2025

Figura No.3
Registro de nuevo Cliente



Fuente: Elaboración propia, 2025

Figura No.4
Despliegue de productos



Fuente: Elaboración propia, 2025

Figura No.5

Control de errores en ingreso de datos cliente

```

--- REGISTRO DE CLIENTE ---
Nombre del cliente: kevin 32
[!] Nombre inválido. Solo debe contener letras y espacios (2-50 caracteres)
Nombre del cliente: 343"##
[!] Nombre inválido. Solo debe contener letras y espacios (2-50 caracteres)
Nombre del cliente: Kevin Hernandez
Email del cliente: kevinjuergen98@gmail.com

[✓] Cliente ya registrado. Usando datos existentes

```

Fuente: Elaboración propia, 2025

Figura No.6

Control de errores de ingreso de productos

```

Ingrese ID del producto (0 para finalizar)
> ee
[!] ID debe ser un número. Intente nuevamente

Ingrese ID del producto (0 para finalizar)
> #
[!] ID debe ser un número. Intente nuevamente

Ingrese ID del producto (0 para finalizar)
> 7
Cantidad para 'Regulador de Voltaje LM7805': 2
[✓] Agregado: 2 x Regulador de Voltaje LM7805 = Q1.50

Ingrese ID del producto (0 para finalizar)
> 4

```

Fuente: Elaboración propia, 2025

Figura No.7

Confirmacion de facturacion

```

--- RESUMEN DE FACTURA ---
Cliente: kevin Hernandez
Email: kevinjuergen98@gmail.com

Productos:
- Regulador de Voltaje LM7805: 2 x Q0.75 = Q1.50
- Capacitor Electrolítico 100uF 25V: 2 x Q0.25 = Q0.50
-----
TOTAL A PAGAR: Q2.00
-----

[✓] Factura guardada en 'factura.txt'

```

Fuente: Elaboración propia, 2025

Figura No.8

Consulta de historial de facturas

```

3. Consultar facturas existentes
4. Salir

Seleccione una opción: 3

--- CONSULTA DE FACTURAS ---
Ingrese email del cliente: kevinjuergen98@gmail.com

Facturas de kevin Hernandez (kevinjuergen98@gmail.com):
-----
Factura #4 - 29/05/2025 - Total: Q2.00
Factura #3 - 29/05/2025 - Total: Q14.90
Factura #2 - 29/05/2025 - Total: Q3.97
Factura #1 - 29/05/2025 - Total: Q17.96

Ingrese ID de factura para detalles (0 para salir): 4

DETALLE FACTURA #4
-----

```

Fuente: Elaboración propia, 2025

Figura No.9

Factura.

```

factura.txt X
factura.txt
19
20
21
22 FACTURA ELECTRÓNICA - RETRA1
23
24 ID Factura: 3
25 Fecha: 29/05/2025 23:10:00
26 Cliente: kevin Hernandez
27 Email: kevinjuergen98@gmail.com
28
29 DETALLE DE COMPRA:
30
31 CANT PRODUCTO P. UNIT SUBTOTAL ID
32
33 18 Resistencia 1/4W 2200 (paq 100) Q0.90 Q9.90 1
34 4 Placa de circuito Impreso Universal Q1.25 Q5.00 28
35
36 TOTAL: Q14.90
37
38
39
40
41 FACTURA ELECTRÓNICA - RETRA1

```

Fuente: Elaboración propia, 2025

III. CONCLUSIONES

1. Python es un lenguaje de programación de alto nivel el cual nos permite realizar una gran variedad de códigos ya que sus librerías nos permiten ahorrar código a la hora de programar.
2. PostgreSQL es una gran herramienta, ya que es un software, que junto a su administrador visual PGAdmin nos permiten gestionar las bases de datos de una manera más cómoda y más entendible, además de contar con una gran cantidad de herramientas de ayuda.
3. Al emplear tanto Python y PostgreSQL en un proyecto nos permite tener una buena gestión de las bases de datos con PostgreSQL y el manejo dinámico 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/1FGs8Fddjgk0aK5CwWdvkFup9XnNz0QuP/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>