



# UNIVERSIDAD NACIONAL DE TRUJILLO

Facultad de Ingeniería  
Programa de Ingeniería Mecatrónica

---

## INFORME DE PROYECTO

“PROVEEDORA DE GASEOSAS”

---

### PROGRAMACIÓN II

ESTUDIANTE(S) :

1. Bonifacio Julian Royer Matthew
2. Castillo Avila Renzo Arturo
3. Chacon Nontol Erwin Gleissner
4. Cruzado Rosas Italo Geremia

DOCENTE :

Lujan Segura Edwar Glorimer

CICLO :

2024 - IV

Trujillo - Perú  
2024

# ÍNDICE

ÍNDICE	II
1. Introducción.....	1
2. Objetivos	
2.1. Objetivo específico .....	1
2.2. Objetivos generales .....	1
3. Marco teórico .....	2
4. Desarrollo	
4.1. Creación de base de datos .....	7
4.2. Modelo Entidad-Relación .....	11
4.3. Instrucciones de Base de Datos .....	12
4.4. Desarrollo de interfaz .....	20
5. Conclusiones .....	57
6. Referencias bibliográficas .....	58

# 1. Introducción

En la era actual, la tecnología se ha convertido en el motor que impulsa nuestras actividades diarias, permeando todos los aspectos de la vida y ofreciendo soluciones innovadoras en diversos ámbitos. En el ámbito comercial, por ejemplo, la automatización ha revolucionado la manera en que se gestionan los procesos, permitiendo una administración más eficiente y ágil. Una aplicación concreta de esta tecnología se observa en las empresas proveedoras de gaseosas, donde la implementación de interfaces digitales juega un papel crucial.

Este informe detalla el desarrollo de una interfaz utilizando Python, en combinación con las bibliotecas Tkinter y Matplotlib, así como el empleo de un sistema de gestión de bases de datos MySQL. Estas herramientas no solo facilitan la interacción entre comprador y máquina, sino que también optimizan la comunicación entre administrador y sistema, abriendo nuevas posibilidades para su implementación en entornos reales. A través de este análisis exhaustivo, se explorarán las mejoras potenciales de esta interfaz, con el objetivo de maximizar su utilidad y eficacia en el mundo comercial actual.

## 2. Objetivos

### 2.1. Objetivo específico:

- Desarrollar una interfaz intuitiva y funcional usando Python y MySQL, para mejorar la administración de inventario y ventas de una Proveedor de Gaseosas.

### 2.2. Objetivos generales:

- Integrar un sistema de gestión de bases de datos MySQL para garantizar la precisión y disponibilidad de la información almacenada.

- Utilizar la herramienta tkinter para diseñar e implementar una interfaz eficiente que permita la automatización de procesos y la visualización de datos relevantes en la que interactúa la persona para la toma de decisiones.
- Crear reportes de inventario, clientes e ingreso de la Proveedora de Gaseosas, usando python.

### 3. Marco teórico

#### **Empresa**

Según Rodrigo Uría (2005), la empresa sería «el ejercicio profesional de una actividad económica planificada, con la finalidad de intermediar en el mercado de bienes o servicios».

Según Donaire (2006):

La empresa es una organización, de duración más o menos larga, cuyo objetivo es la consecución de un beneficio a través de la satisfacción de una necesidad de mercado.

La satisfacción de las necesidades que plantea el mercado se concreta en el ofrecimiento de productos (empresa agrícola o sector primario, industrial o sector secundario, servicios o sector terciario), con la contraprestación de un precio. (p.1)

Por otra parte las empresas, bajo la dirección y responsabilidad del empresario, construyen o realizan un conjunto de bienes y servicios con el objetivo de satisfacer las necesidades del mercado mediante la contraprestación del precio que existe actualmente en el mercado.

Para determinar o fijar con precisión los límites del mercado debemos distinguir entre:

1. **Ámbito geográfico:** Delimitación geográfica del entorno de actividad de la empresa.

Ej.: local, interior, exterior, de un país o región, etc.

2. **Ámbito conceptual:** Delimitación conceptual del mercado, relativa a la definición del producto o servicio (informático, financiero, etc.) o bien, delimitación referida al

colectivo de personas o entidades potencialmente usuarias de los productos o servicios (infantil, profesional, etc.). (Donaire,2006,p.1).

### **Empresa Virtual**

La denominada "empresa virtual" o "corporación modular" se refiere a una estructura organizativa adaptable que responde ágilmente a las demandas del mercado actual, aprovechando de manera intensiva las Tecnologías de la Información (TI). Este concepto se originó en la propuesta de la "organización trébol" de Charles Handy en 1989, y en la década de los noventa evolucionó hacia la idea de una "red temporal de empresas" que se unen para capitalizar oportunidades específicas de mercado mediante la utilización de capacidades tecnológicas. La esencia de la empresa virtual radica en su capacidad para establecer alianzas estratégicas y colaboraciones temporales, permitiendo la participación en proyectos específicos sin comprometerse a estructuras organizativas rígidas y permanentes, gracias a la eficiente comunicación y coordinación que posibilitan las tecnologías de la información (Marcial,2021).

### **Proveedora de gaseosas**

Las responsabilidades de una empresa proveedora de gaseosa pueden incluir la gestión del inventario, el almacenamiento adecuado de los productos, la coordinación de la logística para la entrega eficiente a los clientes, y la promoción de sus productos en el mercado local. Además, estas empresas pueden ofrecer servicios adicionales, como la instalación de equipos de dispensación de gaseosas en los puntos de venta.

En muchos casos, las empresas proveedoras de gaseosa establecen acuerdos con fabricantes de marcas populares para distribuir sus productos en una región específica. Estos acuerdos suelen incluir términos de exclusividad que les dan a las empresas proveedoras derechos exclusivos para comercializar y vender ciertas marcas en un área geográfica determinada.

## **El inventario**

Mendoza Gómez et al. (2019) menciona que: El inventario es capital en forma de material, ya que éste tiene un valor para las compañías, sobre todo para aquellas que se dedican a la venta de productos. Es por esto que es de suma importancia, ya que permite a la empresa cumplir con la demanda y competir dentro del mercado. (p. 25) . Por esta razón la empresa debe tener seguimientos continuos de los productos, ya que ha invertido su capital y necesita obtener liquidez para cubrir las diferentes obligaciones que ha incurrido para adquirirla, de modo que si mantiene productos inmovilizados solo generarán costos adicionales que afectan la utilidad de la misma.

## **La importancia del inventario para la empresa**

Arenal Laza (2020) menciona que: La importancia del inventario físico radica en los siguientes puntos: Permite verificar que lo anotado en los registros contables (presumiblemente digitalizados) efectivamente exista. Confirma la rotación de los productos, que es la rapidez con la que se venden o circulan los bienes. (p. 10)

Por lo tanto, es de vital importancia identificar y determinar los niveles de stock para hacer frente a la demanda de los clientes, mediante el control de inventario se podrá conocer la cantidad, la ubicación, el ingreso y la salida de cada uno de los productos permitiéndole a la administración tomar decisiones con el fin de reducir costos, prevenir robos hormigas, pérdidas por obsolescencia o deterioro y realizar abastecimientos óptimos y adecuados.

## **Marketing**

El marketing como toda disciplina se ha ido adaptando a los cambios que se han dado a lo largo del tiempo, es comprensible entonces que durante el proceso evolutivo del marketing su enfoque y orientación hayan ido variando, originando así etapas marcadas por cada transición.

Las ventas son un punto de suma importancia para todo tipo de empresa ya que por medio de estas la empresa puede subsistir y posteriormente generar utilidades, es de entenderse que las empresas siempre están en busca del incremento de sus ventas, sin embargo, el incremento de ventas es un proceso que requiere de tiempo y paciencia, pero tiene grandes beneficios.

## **Conceptos de programación empleados en el proyecto**

### **Python**

Según Gonzales Duque (2008) :

Python es un lenguaje de programación creado por Guido van Rossum a principios de los años 90 cuyo nombre está inspirado en el grupo de cómicos ingleses “Monty Python”. Es un lenguaje similar a Perl, pero con una sintaxis muy limpia y que favorece un código legible. Se trata de un lenguaje interpretado o de script, con tipado dinámico, fuertemente tipado, multiplataforma y orientado a objetos.

### **Interfaz gráfica**

La interfaz de usuario constituye la parte visible y perceptible de las aplicaciones. En consecuencia, se está dando cada vez más importancia y atención a su desarrollo. En el ámbito del desarrollo de software, la creación de interfaces de usuario ha experimentado una evolución en los últimos años, y este progreso continúa en la actualidad.

Se puede indicar que la interfaz de usuario es la primera impresión que se tiene de una aplicación, y si no alcanza los estándares de calidad adecuados, puede generar rechazo por parte del usuario. De hecho, una interfaz tiene el potencial de imponer restricciones en la comunicación entre la máquina y el usuario. Todo aquello que no pueda expresarse a través de la interfaz se perderá, por lo tanto, el diseño de la interfaz juega un papel crucial en el desarrollo de una aplicación (Montalvo,2018).

## **Cómo usar la interfaz dentro de Python**

Para el uso de interfaz gráfica o su creación dentro de Python es necesario importar el paquete de Tkinter, el cual es una capa destinada a objetos basados en Tcl y Tk.

A partir de este paquete (Tkinter), importándolo antes de iniciar nuestra programación dentro de Python, es que nos permite crear una interfaz gráfica según como queramos nosotros mismos diseñarlos ya que nos da la facilidad de utilizar más códigos para que nuestra interfaz se vea más creativa y se adapte a nuestras condiciones.

## **Base de datos**

Cruz (2011) señala que una base de datos se define como una agrupación de archivos interrelacionados diseñados para gestionar la información de una empresa. Cada uno de estos archivos puede concebirse como una recopilación de registros, y cada registro se compone de una serie de campos. Cada uno de los campos en un registro almacena información sobre un atributo específico de una entidad en el mundo real. En términos visuales, se puede imaginar un archivo de base de datos como una tabla con filas y columnas, donde cada fila representa un registro del archivo y cada columna corresponde a un campo.

## **MySQL Workbench**

Es una herramienta visual y entorno integrado de desarrollo diseñado para trabajar con bases de datos MySQL, el cual fue desarrollado por Oracle, el cual permite un diseño y modelado de base de datos, desarrollo de consultas SQL, administración de usuarios y privilegios, migración de datos, monitoreo y optimización del rendimiento, generación de informes visuales y automatización de tareas

Además, Workbench es un cliente de base de datos, lo cual significa que se tiene que conectar a una base de datos que ya existe, entonces es primordial instalar y conectarlo con el software Xampp.



## 4. Desarrollo:

### 4.1. Creación de base de datos:

- Creamos una base de datos en MySQL Workbench, luego en python creamos las diferentes tablas que nos serán de uso para poder desarrollar el proyecto de una manera ordenada, las tablas fueron:

- Administrador: Esta tabla sera la encargada de almacenar los usuarios y claves de los administradores, para poder ingresar al apartado de administración.

id_usuario	Usuario	Clave
1	10001	MECATRONICA
2	10002	MECA02
3	10006	MECAR5
4	10008	MAT5
5	10007	MECATRO
6	10006	MECATRONICA0

Tabla 1. Administrador

- Gaseosas: Esta tabla e sla encargada de almacenar el inventario, osea las gaseosas que están disponibles para su venta.

Codigo	Nombre	SixPack	Precio	Litros
10010	COACOLA	25	15	1.5
10011	GUARANA	11	10	0.45
10012	PEPSI	8	11.5	0.5
10013	INKACOLA	33	14	0.5
10014	KR	20	7.5	0.35
10015	COACOLA	13	11	0.5

Tabla 2. Gaseosas

- Cliente: Esta tabla es encargada de almacenar los datos personales de los clientes que realizarán sus pedidos.

PedidoID	Nombre	NombreTienda	Ruc	Telefono	Fecha
1	ROYER	PEPITO	12345678945	949652317	2024-01-29 15:42:07
2	LUIS	COLIBRI	45678912355	949785456	2024-01-29 15:47:02
3	ANDRES	PEDRETE	78945612345	978546321	2024-02-01 00:04:45
4	LUISA	LUCHITA	65478912356	987458732	2024-02-01 15:56:11

Tabla 3. Cliente

- Detalle\_cliente: Esta tabla está encargada de almacenar los datos de la residencia del cliente como Calle y Número.

PedidoID	Calle	Numero
1	SAN ANDRES	241
2	AV ESPANA	789
3	AV JUAN	785
4	SAN ANTONIO	784

*Tabla 4. Detalle\_cliente*

- Pedido: Esta tabla es la principal, en cargada de almacenar los pedidos realizados por los clientes.

PedidoID	Gaseosa	Cantidad	Litros	Costo	ClienteId
1.1	GUARANA	5	0.45	10	1
1.2	KR	6	0.35	7.5	1
2.1	KR	3	0.35	7.5	2
2.2	PEPSI	4	0.5	11.5	2
2.3	COCACOLA	2	0.5	13.5	2
3.1	COCACOLA	8	0.5	13.5	3
3.2	INKACOLA	10	0.5	14	3
3.3	PEPSI	7	0.5	11.5	3
4.1	KR	8	0.35	7.5	4
4.2	PEPSI	6	0.5	11.5	4
4.3	GUARANA	9	0.45	10	4

*Tabla 5. Pedido*

- Detalle\_factura: Esta tabla está conformada por columnas derivadas, en la que se almacenará el subtotal, igv y total de cada pedido realizado.

PedidoID	Subtotal	Igv	Total
1.1	50	9	59
1.2	45	8.1	53.1
2.1	22.5	4.05	26.55
2.2	46	8.28	54.28
2.3	27	4.86	31.86
3.1	108	19.44	127.44
3.2	140	25.2	165.2
3.3	80.5	14.49	94.99
4.1	60	10.8	70.8
4.2	69	12.42	81.42
4.3	90	16.2	106.2

*Tabla 5. Detalle\_factura*

- El código para la creación de las tablas fue el siguiente:

```
import mysql.connector
conexion = mysql.connector.connect(
    host = "localhost",
    user = "root",
    password = "",
    database = "proyecto"
)
cursor = conexion.cursor()
cursor.execute("DROP TABLE IF EXISTS gaseosas")
cursor.execute("CREATE TABLE gaseosas(Codigo INT AUTO_INCREMENT PRIMARY
        KEY , Nombre VARCHAR(255) , SixPack INT, Precio FLOAT, Litros
        DOUBLE, INDEX idx_nombre (Nombre)) AUTO_INCREMENT = 10010 ")
valores = [
    ("COCACOLA", 20, 13.5, 0.5),
    ("GUARANA", 25, 10, 0.45),
    ("PEPSI", 30, 11.50, 0.5),
    ("INKACOLA", 48, 14, 0.5),
    ("KR", 35, 7.5, 0.35)
]
cursor.executemany("INSERT INTO gaseosas (Nombre, SixPack, Precio, Litros)
        VALUES (%s,%s,%s, %s)", valores)
cursor.execute("CREATE TABLE Cliente( PedidoID INT AUTO_INCREMENT PRIMARY
        KEY, Nombre VARCHAR(255), NombreTienda VARCHAR(255), Ruc
        BIGINT, Telefono BIGINT, Fecha TIMESTAMP DEFAULT
        CURRENT_TIMESTAMP)")
cursor.execute("CREATE TABLE Pedido( PedidoID VARCHAR(30), Gaseosa
        VARCHAR(255), Cantidad INT, Litros DOUBLE, Costo FLOAT, INDEX
        idx_pedidoid (PedidoID), FOREIGN KEY (Gaseosa) REFERENCES
        gaseosas(Nombre))")
cursor.execute("CREATE TABLE Administrador( id_usuario INT AUTO_INCREMENT
        PRIMARY KEY, Usuario BIGINT, Clave VARCHAR(255))")
```

```

cursor.execute("CREATE TABLE Detalle_cliente(PedidoID INT UNIQUE, Calle
          VARCHAR(50), Numero INT, FOREIGN KEY (PedidoId) REFERENCES
          Cliente(PedidoId))")
cursor.execute("CREATE TABLE Detalle_factura ( PedidoID VARCHAR(30) ,
          Subtotal FLOAT, Igv FLOAT GENERATED ALWAYS AS (Subtotal*0.18)
          VIRTUAL, Total FLOAT GENERATED ALWAYS AS (Subtotal + Igv)
          VIRTUAL, FOREIGN KEY (PedidoID) REFERENCES
          Pedido(PedidoID))")
cursor.execute("INSERT INTO Administrador (Usuario,Clave) VALUES (10001,
          'MECATRONICA')")
conexion.commit()
conexion.close()

```

- También se realizaron unos códigos en Workbench:

```

DELIMITER //
CREATE TRIGGER insertar_en_Detalle_factura AFTER INSERT ON pedido
FOR EACH ROW
BEGIN
    INSERT INTO Detalle_factura (PedidoID)
    VALUES (NEW.PedidoID);
END;
//
DELIMITER ;
DELIMITER //
CREATE TRIGGER actualizar_subtotal AFTER UPDATE ON pedido
FOR EACH ROW
BEGIN
    UPDATE detalle_factura
    SET subtotal = NEW.cantidad * NEW.costeo
    WHERE PedidoID = NEW.PedidoID;
END;
//

DELIMITER ;

```

. La cuál es encargada de insertar y actualizar la tabla “detalle\_factura” de manera automática cuando se inserte un pedido en la tabla “pedido”.

```
ALTER TABLE pedido
ADD COLUMN ClienteId INT,
ADD CONSTRAINT fk_ClienteId
FOREIGN KEY (ClienteId)
REFERENCES cliente(PedidoID);
```

. Y agregamos una columna “ClienteID” a la tabla pedido, con un FOREIGN KEY con referencia a la columna PedidoID de la tabla cliente

#### 4.2. Modelo Entidad – Relación

. Para la creación del modelo entidad – relación se realizó en el software “DIA”, en este modelo podemos observar las relaciones (FOREIGN KEY, UNIQUE) que existen entre las diversas tablas creadas junto con la cardinalidad existente entre ellas.

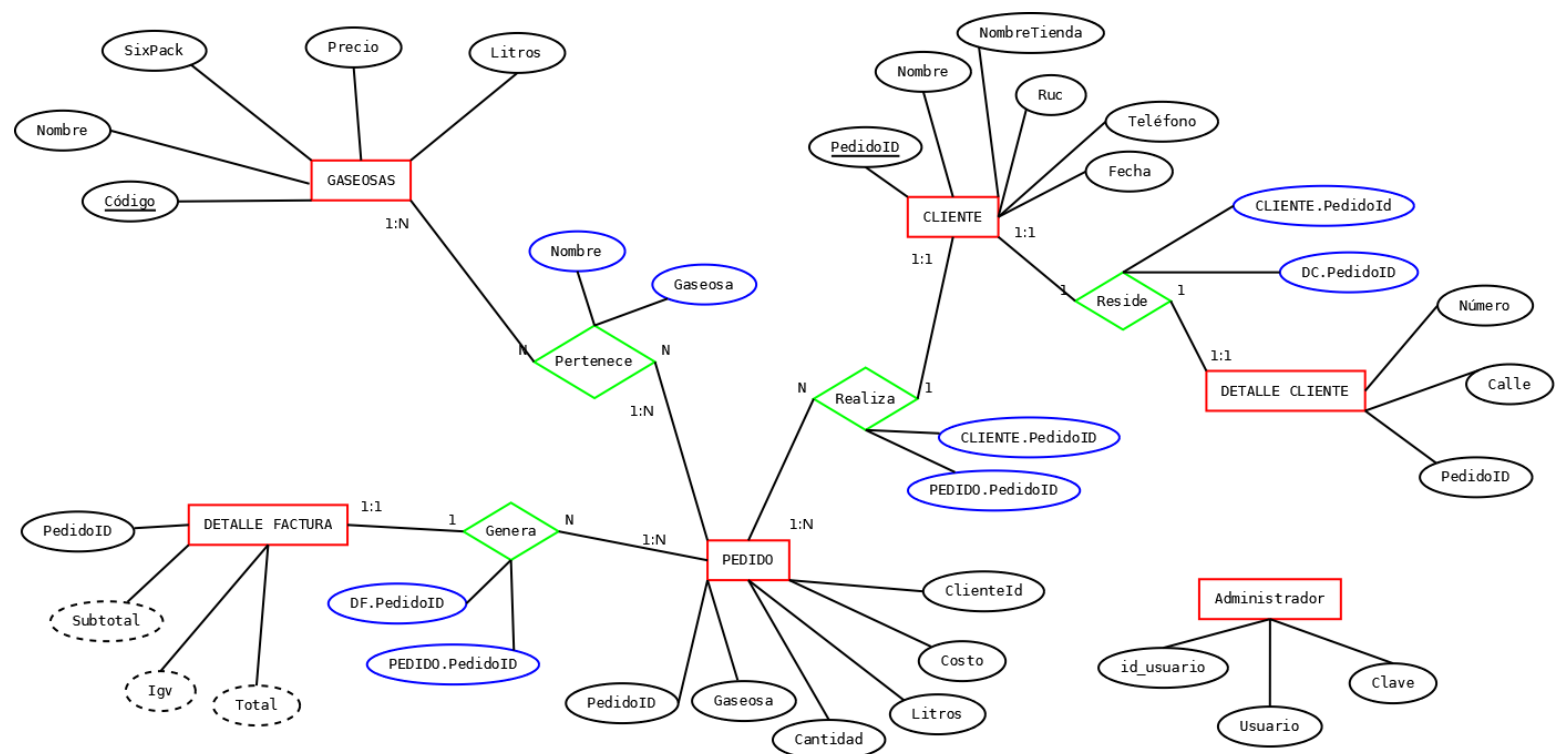


Figura 1. Modelo entidad-relación

#### **4.3.Instrucciones de Base de Datos:**

- Al momento de interactuar con el programa se haran diferentes acciones que involucren a la base de datos, como ingresar datos nuevos, modificar datos, eliminar datos o extraer datos para poder utilizarlos como labels; para eso hemos desarrollado diversas instrucciones en python capaces de realizar estas acciones.
- INSTRUCCIONES DE ADMINISTRACIÓN:
  - . Encargada de verificar si el usuario y contraseña se encuentra en la base de datos, si es así permitirá el ingreso, de lo contrario lo negará.

```
def signin(usuario, clave):  
    conn = mysql.connector.connect(  
        host = "localhost",  
        user = "root",  
        password = "",  
        database = "proyecto")  
  
    cursor = conn.cursor()  
  
    query = "SELECT * FROM administrador WHERE Usuario =  
        %s AND Clave = %s"  
    cursor.execute(query, (usuario, clave))  
    result = cursor.fetchone()  
  
    cursor.close()  
    conn.close()  
  
    if result:  
        return True  
    else:  
        return False
```

. Encargado de insertar a la base de datos un nuevo usuario y clave al momento de hacer el registro, si es que la clave de donfirmación es la correcta.

```
def verificar_clave(clave):
    conn = mysql.connector.connect(
        host = "localhost",
        user = "root",
        password = "Hoyesdia21",
        database = "proyecto")
    cursor = conn.cursor()
    query = "SELECT Clave FROM Administrador"
    cursor.execute(query)
    result = cursor.fetchone()
    if result and result[0] == clave:
        conn.close()
        return True
    else:
        conn.close()
        return False

def insertar_usuario(usuario, clave):
    conn = mysql.connector.connect(
        host = "localhost",
        user = "root",
        password = "Hoyesdia21",
        database = "proyecto")
    cursor = conn.cursor()

    cursor = conn.cursor()

    query = "INSERT INTO Administrador (Usuario, Clave) VALUES
        (%s, %s)"
    cursor.execute(query, (usuario, clave))
    conn.commit()
    cursor.close()
    conn.close()
```

. La clase Data es la que se utilizará para poder mostrar el inventario de gaseosas, se encarga de modificar, agregar o eliminar las gaseosas del inventario recuerdo a lo pedido.

```
class Data:
    def __init__(self):
        self.conn = mysql.connector.connect(
            host = "localhost",
            user = "root",
            password = "",
            database = "proyecto")
        self.cursor = self.conn.cursor()

    def InsertItems(self, element):
        sql = "insert into gaseosas(Nombre, SixPack, Precio, Litros)
            values('{}', '{}',
            '{}', '{}')".format(element[0], element[1], element[2],
            element[3])
        self.cursor.execute(sql)
        self.conn.commit()

    def ReturnOneItem(self, ref):
        sql = "select * from gaseosas where Nombre = '{}'".format(ref)
        self.cursor.execute(sql)
        return self.cursor.fetchone()

    def returnAllElements(self):
        sql = "select * from gaseosas"
        self.cursor.execute(sql)
        return self.cursor.fetchall()

    def Delete(self, ref):
        sql = "delete from gaseosas where Nombre = '{}'".format(ref)
        self.cursor.execute(sql)
        self.conn.commit()

    def UpdateItem(self, element, ref):
        sql = "UPDATE gaseosas SET Nombre = %s, SixPack = %s, Precio =
            %s, Litros = %s WHERE Codigo = %s"
        self.cursor.execute(sql, (element[1], element[2], element[3],
            element[4], ref))
        self.conn.commit()
```



. La clase Data\_1admi, se encarga de extraer los datos que se utilizarán en el reporte, por ejemplo, los datos de las gaseosas para realizar el círculo de porcentaje, cantidad de gaseosas vendidas, gaseosa más vendida.

```
class Data_1admi():
    def __init__(self):
        self.conn = mysql.connector.connect(
            host = "localhost",
            user = "root",
            password = "",
            database = "proyecto")
        self.cursor = self.conn.cursor()

    def circulo(self):
        sql = "SELECT Gaseosa, SUM(Cantidad) FROM Pedido GROUP BY
            Gaseosa"
        self.cursor.execute(sql)
        dato = self.cursor.fetchall()
        return dato

    def total(self):
        sql = "SELECT SUM(Cantidad) FROM Pedido "
        self.cursor.execute(sql)
        dato = self.cursor.fetchone()[0]
        return dato

    def max_gaseosa(self):
        sql = "SELECT Gaseosa FROM Pedido GROUP BY Gaseosa ORDER BY
            SUM(Cantidad) DESC LIMIT 1 "
        self.cursor.execute(sql)
        dato = self.cursor.fetchone()[0]
        return dato
```

. Para el reporte de los clientes, la siguiente instrucción se encarga de extraer solo los datos entre las fechas seleccionadas.

```
def cliente(self, fecha_inicio, fecha_fin):
    sql = f"""SELECT C.PedidoId, C.Nombre, C.NombreTienda, D.Calle,
                    D.Numero, C.Ruc, C.Telefono, C.Fecha
    FROM Cliente C,Detalle_Cliente D WHERE C.PedidoID =
    D.PedidoID AND DATE(C.Fecha) BETWEEN '{fecha_inicio}'
    AND '{fecha_fin}' """
    self.cursor.execute(sql)
    dato = self.cursor.fetchall()
    return dato
```

. Para el reporte de los ingresos generados, la siguiente instrucción devuelve los pedidos realizados junto con la suma de los ingresos totales.

```
def cliente(self, fecha_inicio, fecha_fin):
    sql = f"""SELECT p.PedidoID,
                    p.Gaseosa, p.Cantidad, p.Litros, d.Total FROM Pedido p
    JOIN Detalle_factura d ON p.PedidoID = d.PedidoID
    JOIN Cliente c ON p.PedidoID LIKE CONCAT(c.PedidoID,
    '%') AND DATE(c.Fecha) BETWEEN '{fecha_inicio}' AND
    '{fecha_fin}' """
    self.cursor.execute(sql)
    dato = self.cursor.fetchall()
    return dato

def ingreso(self, fecha_inicio, fecha_fin):
    sql = f"""SELECT SUM(d.Total) FROM Detalle_factura d
    JOIN Pedido p ON p.PedidoID = d.PedidoID
    JOIN Cliente c ON p.PedidoID LIKE CONCAT(c.PedidoID,
    '%') AND DATE(c.Fecha) BETWEEN '{fecha_inicio}' AND
    '{fecha_fin}' """
    self.cursor.execute(sql)
    dato = self.cursor.fetchone()[0]
    return round(dato, 2)
```

- INSTRUCCIONES DE PEDIDO:

. Esta instrucción se encarga de insertar en la tabla cliente y detalle\_cliente los datos del cliente ingresados.

```
class Data:

    def InsertItems(self, element):
        sql = "insert into Cliente(Nombre, NombreTienda, Ruc, Telefono)
              values('{ }', '{ }',
                    '{ }', '{ }')".format(element[0],element[3],element[4],element
                    [5])
        self.cursor.execute(sql)
        self.cursor.execute("SELECT PedidoID FROM Cliente ORDER BY PedidoID
                             DESC LIMIT 1")
        pedido_id = self.cursor.fetchone()[0]
        sql = "insert into Detalle_cliente(PedidoID,Calle, Numero)
              values('{ }',
                    '{ }', '{ }')".format(pedido_id,element[1],element[2])
        self.cursor.execute(sql)
        self.conn.commit()

    def ReturnElements(self):
        sql = "SELECT * FROM Cliente ORDER BY PedidoID DESC LIMIT 1"
        self.cursor.execute(sql)
        element = self.cursor.fetchone()
        return element
```

. La clase Data2 se encarga de insertar los pedidos en la tabla pedido, retornar las gaseosas disponibles para su compra, actualizar la cantidad de gaseosas disponibles luego de realizar la compra.

```
class Data2:

    def InsertItems2(self, element,i):
        self.cursor.execute("SELECT PedidoID FROM Cliente ORDER BY
                             PedidoID DESC LIMIT 1")
        pedido_id = self.cursor.fetchone()[0]
        codigo_ped = f"{pedido_id}.{i}"
        sql = "insert into Pedido(PedidoID,Gaseosa,Cantidad, Litros)
              values('{ }', '{ }',
                    '{ }', '{ }')".format(codigo_ped,element[0],element[1],
                    element[2])
        self.cursor.execute(sql)
        update_sql = f""""UPDATE Pedido SET Costo = (SELECT Precio FROM
                gaseosas WHERE gaseosas.Nombre = UPPER(Pedido.Gaseosa)
                AND gaseosas.Litros = Pedido.Litros) WHERE
                PedidoID = '{codigo_ped}'""""
        self.cursor.execute(update_sql)
        update_id = f"UPDATE Pedido SET ClienteId = '{pedido_id}' WHERE
                PedidoID LIKE '{pedido_id}%"
        self.cursor.execute(update_id)
        self.conn.commit()
```

```

def returnAllElements(self):
    sql = "select Nombre, SixPack, Precio, Litros from gaseosas"
    self.cursor.execute(sql)
    return self.cursor.fetchall()

def UpdateSix(self, litro, six, Nombre):
    self.cursor.execute(f"SELECT SixPack FROM gaseosas WHERE Litros = '{litro}' AND Nombre = '{Nombre}'")
    resultado = self.cursor.fetchone()
    valor_columna = resultado[0] if resultado else None
    valor = int(valor_columna)
    resultado = valor - six
    self.cursor.execute(f"UPDATE gaseosas SET SixPack = '{resultado}' WHERE Litros = '{litro}' AND Nombre = '{Nombre}'")
    self.conn.commit()

```

. Al momento de insertar una fila de pedido en la tabla pedido, se insertará automáticamente su suvtotal, igv y total en la tabla detalle\_factura, por lo tanto estas funciones se encargan de devolver los pedidos el subtotal, igv y total de los pedidos realizados para el resumen del pedido.

```

def resumen(self):
    self.cursor.execute("SELECT PedidoID FROM Cliente ORDER BY PedidoID DESC LIMIT 1")
    pedido_id = self.cursor.fetchone()[0]
    sql = f"SELECT * FROM Pedido WHERE PedidoId LIKE '{pedido_id}%"
    self.cursor.execute(sql)
    result = self.cursor.fetchall()
    return result

def Subtotal(self):
    self.cursor.execute("SELECT PedidoID FROM Cliente ORDER BY PedidoID DESC LIMIT 1")
    pedido_id = self.cursor.fetchone()[0]
    sql = f"SELECT Subtotal FROM Detalle_Factura WHERE PedidoId LIKE '{pedido_id}%"
    self.cursor.execute(sql)
    result = self.cursor.fetchall()
    return result

```

```

def igv(self):
    self.cursor.execute("SELECT PedidoID FROM Cliente ORDER BY PedidoID
                        DESC LIMIT 1")
    pedido_id = self.cursor.fetchone()[0]
    sql = f"SELECT Igv FROM Detalle_Factura WHERE PedidoId
            LIKE'{pedido_id}%"
    self.cursor.execute(sql)
    result =self.cursor.fetchall()
    val = 0
    for i in result:
        for j in i:
            val+=j
    return round(val,2)

def Total(self):
    self.cursor.execute("SELECT PedidoID FROM Cliente ORDER BY PedidoID
                        DESC LIMIT 1")
    pedido_id = self.cursor.fetchone()[0]
    sql = f"SELECT Total FROM Detalle_Factura WHERE PedidoId
            LIKE'{pedido_id}%"
    self.cursor.execute(sql)
    result =self.cursor.fetchall()
    val = 0
    for i in result:
        for j in i:
            val+=j
    return val

```

. Estas instrucciones la usamos al momento de generar la factura, las cuales devolverán, la dirección del cliente, la fecha del pedido y los pedidos para poder ponerlos en la factura.

```

def direccion(self):
    self.cursor.execute("SELECT Calle, Numero FROM Detalle_cliente ORDER
                        BY PedidoID DESC LIMIT 1")
    result =self.cursor.fetchone()
    return result

def fecha(self):
    self.cursor.execute("SELECT Fecha FROM Cliente ORDER BY PedidoID DESC
                        LIMIT 1")
    result =self.cursor.fetchone()[0]
    return result

```

```

def factura(self):
    self.cursor.execute("SELECT PedidoID FROM Cliente ORDER BY PedidoID
                        DESC LIMIT 1")
    pedido_id = self.cursor.fetchone()[0]
    sql = f"SELECT Gaseosa,Cantidad,Litros, Costo FROM Pedido WHERE
            PedidoId LIKE'{pedido_id}%'
    self.cursor.execute(sql)
    result =self.cursor.fetchall()
    return result

```

#### **4.4. Desarrollo de interfaz:**

- Para el desarrollo de la interfaz hemos utilizado tkinter, haciendo uso principalmente de labels, entrys, buttons, matplotlib, Radiobuttons y ttk, la cual es una herramienta que nos permite mostrar en tabla los datos de MySQL.

. PÁGINA PRINCIPAL:

```

from tkinter import *
from databaseadmi import *
from tkinter import messagebox
from tkinter import ttk
from principaladmi import *
from principalpedido import *
from ingreso_registro import *
pagina = Tk()
pagina.geometry("700x600")
pagina.config(bg="light cyan")
pagina.title("PEDIDO DE GASEOSAS")
frame = Frame(pagina)
frame.pack(expand=True)
etiqueta = Label(pagina, text="PROVEEDORA DE GASEOSAS",fg = "black", bg =
"light cyan" ,font=("courier new", 25, "bold")).place(x=150,y=50)
imagen = PhotoImage(file="gaseosas.png")
imagen_label = Label(pagina, image=imagen, bg = "light cyan")
imagen_label.place (x =200,y =90)
boton1=Button(pagina, text = "ADMINISTRACION", bg = "light grey", font =
("courier new", 20, "bold"), command = seguridad_admi,border = 5).place(x =
80, y = 400)
boton2 = Button(pagina, text = "PEDIDO", bg = "light grey", font = ("courier
new", 20, "bold"),command=ventana_ped, border = 5)
boton2.place(x = 450, y = 400)
pagina.mainloop()

```



*Figura 2. Interfaz Principal*

. PÁGINA DE ADMINISTRACIÓN:

```
from tkinter import *
from principaladmi import *

def seguridad_admi():

    pagina2 =Toplevel()
    pagina2.title("USUARIO DE ADMINISTRADOR")
    pagina2.geometry("600x400")
    pagina2.config(bg="light cyan")
    Label(pagina2, text="INGRESE USUARIO DE ADMINISTRADOR",fg = "black", bg =
        "light cyan" ,font=("courier new", 20, "bold")).place(x=40,y=30)
    Label(pagina2, text="USUARIO:",fg = "black", bg = "light cyan"
        ,font=("courier new", 14, "bold")).place(x=120,y=120)
    Label(pagina2, text="CONTRASEÑA:",fg = "black", bg = "light cyan"
        ,font=("courier new", 14, "bold")).place(x=120,y=200)

    user_val = StringVar()
    key_val = StringVar()
    Entry(pagina2,font=('Arial', 12),relief="flat", background="white"
        ,textvariable=user_val).place(x=260, y=120, height=25, width=150)
    Entry(pagina2,font=('Arial', 12),relief="flat", background="white"
        ,textvariable=key_val, show="*").place(x=260, y=200, height=25,
        width=150)
```


```

def verificacion():
    usuario = user_val.get()
    clave = key_val.get()

    if signnin(usuario, clave):
        administrador()
        pagina2.destroy()
    else:
        messagebox.showwarning(title="USUADIO DE ADMINISTRADOR", message=
            "Usuario y/o contraseña incorrecto")

Button(pagina2, text = "INGRESAR", bg = "light grey", font = ("courier
    new", 20, "bold"), border = 5, command=verificacion).place(x=130,
    y=300)
Button(pagina2, text = "REGISTRAR", bg = "light grey", font = ("courier
    new", 20, "bold"), border = 5,command=login).place(x=330, y=300)

```



USUARIO DE ADMINISTRADOR

**INGRESE USUARIO DE ADMINISTRADOR**

USUARIO:

CONTRASEÑA:

**INGRESAR** **REGISTRAR**

*Figura 3. Página de administrador*



. PÁGINA DE REGISTRO:

```
def login():
    pagina4 = Toplevel()
    pagina4.title("REGISTRO")
    pagina4.geometry("450x400")
    pagina4.config(bg="light cyan")
    Label(pagina4, text="REGISTRAR NUEVO USUARIO",fg = "black", bg = "light
        cyan" ,font=("courier new", 20, "bold")).place(x=40,y=30)
    Label(pagina4, text="NUEVO USUARIO:",fg = "black", bg = "light cyan"
        ,font=("courier new", 14, "bold")).place(x=50,y=120)
    Label(pagina4, text="NUEVA CONTRASEÑA:",fg = "black", bg = "light cyan"
        ,font=("courier new", 14, "bold")).place(x=50,y=200)

    global new_user
    global new_key
    new_user = StringVar()
    new_key = StringVar()

    Entry(pagina4,font=('Arial', 12),relief="flat", background="white"
        ,textvariable=new_user).place(x=250, y=120, height=25, width=150)
    Entry(pagina4,font=('Arial', 12),relief="flat", background="white"
        ,textvariable=new_key).place(x=250, y=200, height=25, width=150)

def confirmacion_login():
    pagina5 = Toplevel()
    pagina5.title("REGISTRO")
    pagina5.geometry("450x300")
    pagina5.config(bg="light cyan")
    Label(pagina5, text="CONFIRMACION DE REGISTRO",fg = "black", bg =
        "light cyan" ,font=("courier new", 20, "bold")).place(x=30,y=30)
    Label(pagina5, text="CONTRASEÑA PRINCIPAL:",fg = "black", bg = "light
        cyan" ,font=("courier new", 14, "bold")).place(x=20,y=90)

    global confirm_key
    confirm_key = StringVar()
```

```

Entry(pagina5,font=('Arial', 12),relief="flat", background="white"
      ,textvariable=confirm_key, show = "*").place(x=120, y=150,
      height=25, width=200)
Button(pagina5, text = "CONFIRMAR REGISTRO", bg = "light grey", font
      = ("courier new", 20, "bold"), border = 4, command=
      lambda:[pagina5.destroy(),verificar_e_insertar()]).place(x=60,
      y=220,height=40, width=325)

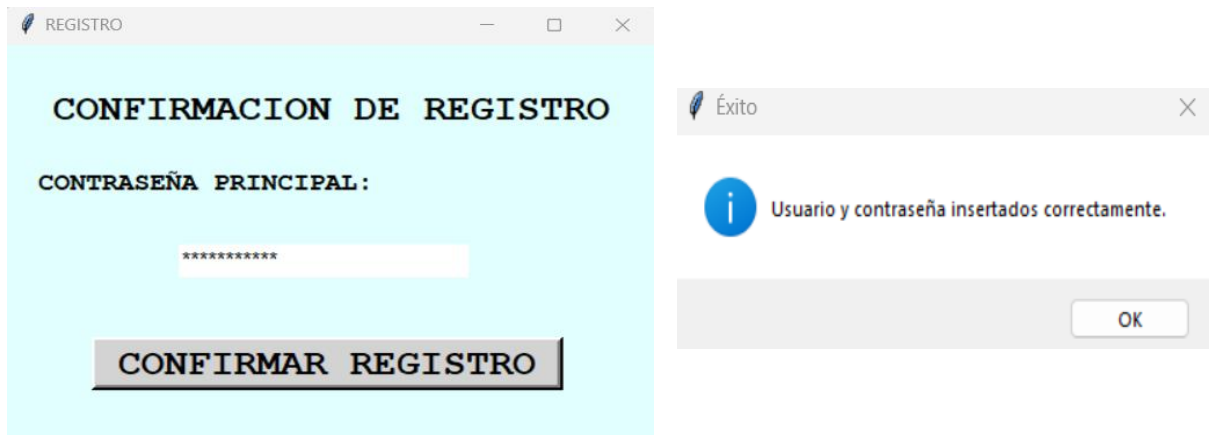
def verificar_e_insertar():
    clave_ingresada = confirm_key.get()
    if verificar_clave(clave_ingresada):
        usuario = new_user.get()
        contraseña = new_key.get()
        insertar_usuario(usuario, contraseña)
        messagebox.showinfo("Éxito", "Usuario y contraseña insertados
            correctamente.")
    else:
        messagebox.showerror("Error", "La clave ingresada no coincide.")

Button(pagina4, text = "REGISTRAR", bg = "light grey", font =
      ("courier new", 20, "bold"), border = 4,
      command=lambda:[pagina4.destroy(),confirmacion_login()]).place(
      x=130, y=300)

```

The screenshot shows a Tkinter window titled "REGISTRO". Inside the window, the text "REGISTRAR NUEVO USUARIO" is displayed at the top. Below this, there are two input fields. The first is labeled "NUEVO USUARIO:" and contains the text "10006". The second is labeled "NUEVA CONTRASEÑA:" and contains the text "MECATRONICA0". At the bottom of the form is a button labeled "REGISTRAR".

*Figura \_4 Registro de usuario*



*Figura 5. Confirmación de registro*

. PÁGINA DE INGRESO:

```
def mostrar_reporte():
    reporte()

def administrador ():

    pagina3 = Toplevel()
    pagina3.title("ADMINISTRACION")
    pagina3.geometry("400x400")
    pagina3.config(bg="light cyan")
    modificar = Button(pagina3, text = "MODIFICAR PRODUCTO", bg = "light
        grey", font = ("courier new", 20, "bold"), border = 5,
        command= ventana_mod)
    modificar.place(x=50, y=70)
    reporte = Button(pagina3, text = "MOSTRAR REPORTE", bg = "light grey",
        font = ("courier new", 20, "bold"), border = 5,command =
        mostrar_reporte)
    reporte.place(x=70, y=200)
    back= Button(pagina3, text = "ATRÁS", bg = "red3", fg = "white", font =
        ("courier new", 20, "bold"), border = 5, command =
```



*Figura 6. Administración*

. PÁGINA MODIFICAR PRODUCTO:

```
class App:
    def __init__(self, master):
        self.frame = master
        self.DrawEntry()
        self.DrawButtons()
        self.DrawLabel()
        self.DrawList()

    def DrawLabel(self):
        self.lbl_title = Label(self.frame,
                                foreground="black", font=("arial", 15, "bold"), background="light
                                cyan", text="AGREGAR").place(x=120, y=70)
        self.lbl_name = Label(self.frame,
                                foreground="black", font=("arial", 13, "bold"), background="light
                                cyan", text="Nombre:").place(x=60, y=110)
        self.lbl_six = Label(self.frame,
                                foreground="black", font=("arial", 13, "bold"), background="light
                                cyan", text="Cantidad(six pack):").place(x=60, y=160)
```

```

self.lbl_price = Label(self.frame,
                        foreground="black",font=("arial",13,"bold"), background="light
                        cyan", text="Precio:").place(x=60, y=210)
self.lbl_liter = Label(self.frame,
                        foreground="black",font=("arial",13,"bold"), background="light
                        cyan", text="Capacidad (L):").place(x=60, y=260)
def DrawEntry(self):
    self.name = StringVar()
    self.six = StringVar()
    self.price = StringVar()
    self.liter = StringVar()
    self.txt_name = Entry(self.frame,font=('Arial', 12),relief="flat",
                           background="white" ,textvariable=self.name)
    self.txt_name.place(x=140, y=110, height=25, width=150)
    self.txt_six = Entry(self.frame,font=('Arial', 12),relief="flat",
                           background="white" ,textvariable=self.six)
    self.txt_six.place(x=240, y=160, height=25, width=50)
    self.txt_price = Entry(self.frame,font=('Arial', 12),relief="flat",
                           background="white" ,textvariable=self.price)
    self.txt_price.place(x=140, y=210, height=25, width=150)
    self.txt_liter = Entry(self.frame,font=('Arial', 12),relief="flat",
                           background="white" ,textvariable=self.liter)
    self.txt_liter.place(x=200, y=260, height=25, width=90)
    # Asocia la función convertir_a_mayusculas al evento de cambio de texto
    en el Entry
    self.txt_name.bind("<KeyRelease>", self.convertir_a_mayusculas)

def convertir_a_mayusculas(self,event):
    # Obtiene el valor actual del Entry y lo convierte a mayúsculas
    nuevo_valor = self.txt_name.get().upper()
    self.name.set(nuevo_valor)

def DrawButtons(self):
    self.btn_confirm = Button(self.frame,foreground="white",
                              text="Guardar",borderwidth=2,relief="flat",
                              cursor="hand1",overrelief="raise",background="blue4",
                              command=lambda:self.confirmProcess()).place(x=1100, y=350,
                              width=90)
    self.btn_cancel = Button(self.frame,
                              text="Cancelar",foreground="white",borderwidth=2,relief="flat",
                              cursor="hand1",overrelief="raise",background="red", command=
                              lambda:self.canceProcess()).place(x=1200, y=350, width=90)

```

```

def DrawList(self):
    self.list_elems = ttk.Treeview(self.frame, columns=(1, 2, 3, 4,5),
        show="headings", height="8")
    # --- STYLE ---
    style = ttk.Style()
    style.theme_use("clam")
    style.configure("Treeview.Hheading",
        background="Royalblue4",relief="flat",foreground="white")
    style.map("Treeview", background=[('selected', 'yellow')],
        foreground=[('selected', 'black')])

    #--- Evento---AL darle doble click a un elemento de la tabla, abra el
        metodo getRow
    self.list_elems.bind("<Double 1>", self.getRow)
    #---- fin ---

    self.list_elems.heading(1, text="Codigo")
    self.list_elems.heading(2, text="Nombre")
    self.list_elems.heading(3, text="Cantidad(six pack)")
    self.list_elems.heading(4, text="Precio")
    self.list_elems.heading(5, text="Capacidad (L)")
    self.list_elems.column(1, anchor=CENTER)
    self.list_elems.column(2, anchor=CENTER)
    self.list_elems.column(3, anchor=CENTER)
    self.list_elems.column(4, anchor=CENTER)
    self.list_elems.column(5, anchor=CENTER)

    # -- LLENAR LIST--
    d = Data()
    self.rows = d.returnAllElements()
    for i in self.rows:
        self.list_elems.insert('', 'end', values=i)
    # ----- fin -----

    self.list_elems.place(x=340, y=90)

    scrollbar = ttk.Scrollbar(self.frame, orient="vertical",
        command=self.list_elems.yview)
    scrollbar.place(x=1340, y=90, height=190)
    self.list_elems.configure(yscrollcommand=scrollbar.set)

```

```

def getRow(self, event):
    self.na = StringVar()
    self.sx = StringVar()
    self.pr = StringVar()
    self.li = StringVar()
    item = self.list_elems.item(self.list_elems.focus())
    if item is not None:
        n = item['values'][1]
        s = item['values'][2]
        p = item['values'][3]
        l = item['values'][4]

        self.na.set(n)
        self.sx.set(s)
        self.pr.set(p)
        self.li.set(l)

    pop = Toplevel(self.frame)
    pop.geometry("400x250")
    lbl_tittle = Label(pop, text="ACTUALIZAR", font=("courier new", 18,
        "bold")).place(x=125, y=10)
    lbl_n = Label(pop, text="Nombre:", font=('Arial', 12),
        fg="black").place(x=50, y=40)
    lbl_s = Label(pop, text="Cantidad(six pack):", font=('Arial', 12),
        fg="black").place(x=50, y=80)
    lbl_p = Label(pop, text="Precio:", font=('Arial', 12),
        fg="black").place(x=50, y=120)
    lbl_l = Label(pop, text="Capacidad (L):", font=('Arial', 12),
        fg="black").place(x=50, y=160)
    txt_n = Entry(pop, textvariable=self.na).place(x=200, y=40)
    txt_s = Entry(pop, textvariable=self.sx).place(x=200, y=80)
    txt_p = Entry(pop, textvariable=self.pr).place(x=200, y=120)
    txt_l = Entry(pop, textvariable=self.li).place(x=200, y=160)
    btn_change = Button(pop, text="Actualizar", relief="flat",
        background="green2", foreground="white", command=lambda:
        self.editar(item)).place(x=180, y=200, width=90)
    btn_delete = Button(pop, text="Eliminar", relief="flat",
        background="red", foreground="white", command=lambda:
        self.eliminar(n)).place(x=290, y=200, width=90)

```

```

def eliminar(self, n):
    d = Data()
    d.Delete(n)
    messagebox.showinfo(title="Eliminación", message="Se ha eliminado el
                        elemento")
    self.ClearList()
    self.DrawList()
    self.ClearEntry()

def editar(self, item):
    na = self.na.get()
    sx = self.sx.get()
    pr = self.pr.get()
    li = self.li.get()
    updated_item = [item['values'][0], na, sx, pr, li]
    d = Data()
    d.UpdateItem(updated_item, item['values'][0])
    messagebox.showinfo(title="Actualización", message="Se han actualizado
                        los datos")

    self.ClearList()
    self.DrawList()
    self.ClearEntry()

def ClearList(self):
    self.list_elems.delete(*self.list_elems.get_children())

def cancelProcess(self):
    self.ClearEntry()

def ClearEntry(self):
    self.name.set("")
    self.six.set("")
    self.price.set("")
    self.liter.set("")

```



```

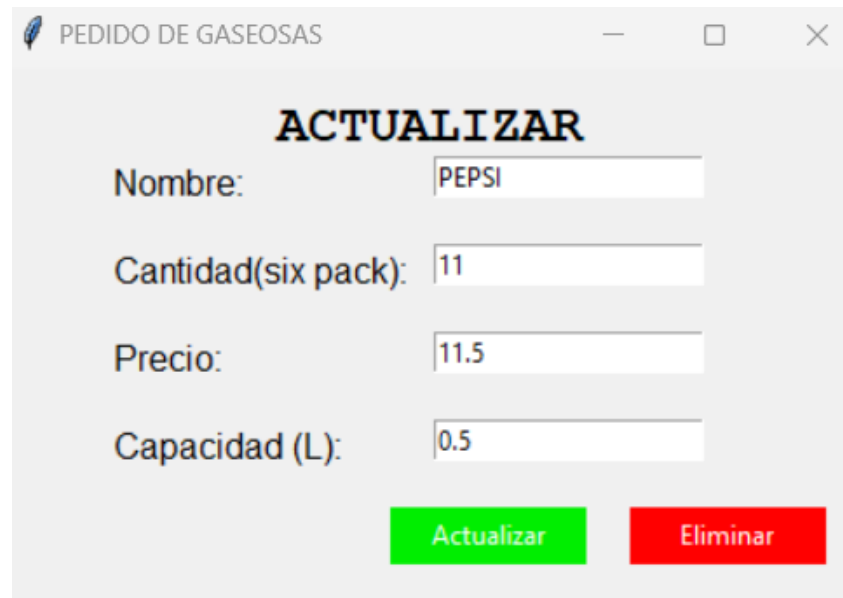
def confirmProcess(self):
    if self.name.get() != "" and self.six.get() != "" and self.price.get()
        != "" and self.liter.get() != "":
        d = Data()
        arr = [self.name.get(), self.six.get(),
            self.price.get(),self.liter.get()]
        d.InsertItems(arr)
        messagebox.showinfo(title="Alerta", message="Se inserto
            correctamente!")
        self.ClearList()
        self.DrawList()
        self.ClearEntry()
    else:
        messagebox.showinfo(title="Error", message="Debe llenar los campos
            para poder guardar!")
def ventana_mod():
    root = Toplevel()
    root.title("MODIFICACIÓN DE GASEOSAS")
    root.config(background="light cyan")
    root.geometry("1400x400")
    Label(root, text="INVENTARIO", fg = "black", bg = "light cyan",font =
        ("courier new", 28, "bold")).place(x=580, y = 20)
    App(root)

```

The screenshot shows a window titled "MODIFICACIÓN DE GASEOSAS" with a light cyan background. At the top center, the word "INVENTARIO" is displayed in a large, bold, black font. On the left side, under the heading "AGREGAR", there are four input fields labeled "Nombre:", "Cantidad(six pack):", "Precio:", and "Capacidad (L):". To the right of these fields is a table with five columns: "Codigo", "Nombre", "Cantidad(six pack)", "Precio", and "Capacidad (L)". The table contains five rows of data. At the bottom right of the window, there are two buttons: "Guardar" (blue) and "Cancelar" (red).

Codigo	Nombre	Cantidad(six pack)	Precio	Capacidad (L)
10010	COCACOLA	10	13.5	0.5
10011	GUARANA	11	10.0	0.45
10012	PEPSI	11	11.5	0.5
10013	INKACOLA	38	14.0	0.5
10014	KR	20	7.5	0.35

*Figura 7. Modificar producto*



**PEDIDO DE GASEOSAS**

**ACTUALIZAR**

Nombre:

Cantidad(six pack):

Precio:

Capacidad (L):

*Figura 8. Actualizar producto*

. PÁGINA DE REPORTE:

```
def reporte():
    root = Toplevel()
    root.title("REPORTES")
    root.config(background="light cyan")
    root.geometry("400x500")
    Button(root, text="VENTA DE GASEOSAS",
           font=("arial",18,"bold"),background="light grey", border = 8, command=
           reporte_gaseosa).place(x=50, y=70, width=300)
    Button(root, text="DETALLE CLIENTES",
           font=("arial",18,"bold"),background="light grey", border = 8, command=
           reporte_cliente).place(x=50, y=200, width=300)
    Button(root, text="INGRESOS", font=("arial",18,"bold"),background="light
           grey", border = 8,command= reporte_ingresos).place(x=50, y=330,
           width=300)
```



*Figura 9. Opciones de reporte*

. REPORTE – VENTA DE GASEOSAS:

```
class App2():
    def __init__(self, master):
        self.frame = master
        self.d = Data_ladmi()
        self.grafico()
        self.imagen()
        self.table()
        self.label()
        self.button()

    def grafico(self):

        valor_grafico = self.d.circulo()

        labels = [row[0] for row in valor_grafico]
        values = [row[1] for row in valor_grafico]
```

```

plt.figure(figsize=(6, 5))
plt.pie(values, labels=labels, autopct='%1.1f%%', startangle=140)
plt.title("Porcentaje de ventas de gaseosas")
plt.axis('equal')

if os.path.exists("grafico_circulo.png"):
    os.remove("grafico_circulo.png")

plt.savefig("grafico_circulo.png")

def imagen(self):

    self.img = PhotoImage(file="grafico_circulo.png")
    label_img = Label(self.frame, image = self.img)
    label_img.image = self.img
    label_img.place(x = 50, y = 70)

def table(self):

    self.list_elems = ttk.Treeview(self.frame, columns=(1, 2),
        show="headings", height= "5")

    style = ttk.Style()
    style.theme_use("clam")
    style.configure("Treeview.Hheading",
        background="Royalblue4", relief="flat", foreground="white")

    self.list_elems.heading(1, text="Gaseosa")
    self.list_elems.heading(2, text="Cantidad")
    self.list_elems.column(1, anchor=CENTER, width=170)
    self.list_elems.column(2, anchor=CENTER, width= 170)

    self.rows = self.d.circulo()
    for i in self.rows:
        self.list_elems.insert('', 'end', values=i)

    self.list_elems.place(x=700, y=150)

```

```

        scrollbar = ttk.Scrollbar(self.frame, orient="vertical",
                                   command=self.list_elems.yview)
        scrollbar.place(x=1044, y=150, height=130)
        self.list_elems.configure(yscrollcommand=scrollbar.set)

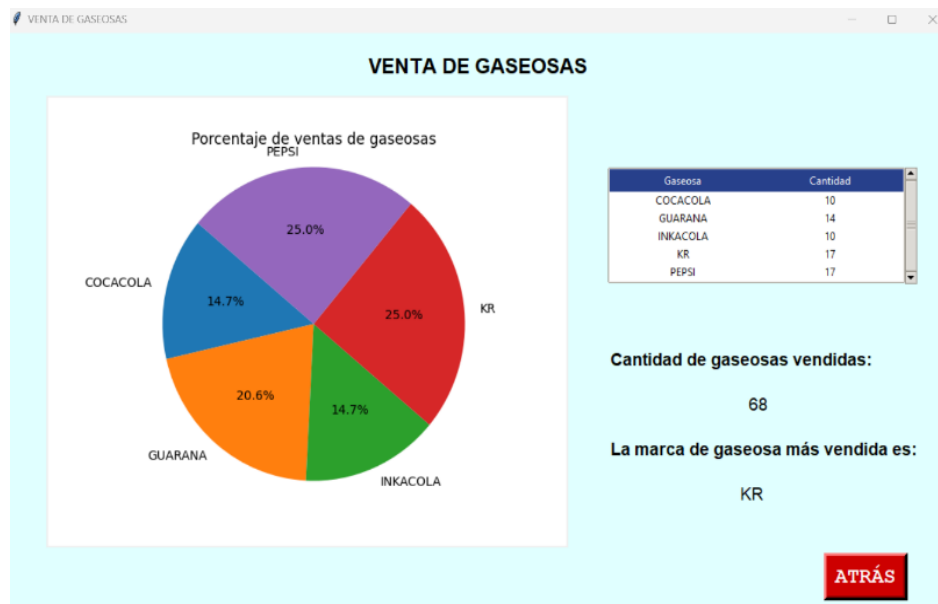
def label(self):
    maxima = self.d.max_gaseosa()
    total = self.d.total()
    self.label_total = Label(self.frame,
                              foreground="black", font=("arial", 14, "bold"),
                              background="light cyan", text="Cantidad de
                              gaseosas vendidas:").place(x=700, y=350)
    self.label_total_val = Label(self.frame,
                                  foreground="black", font=("arial", 14),
                                  background="light
                                  cyan", text=total).place(x=860, y=400)

    self.label_gaseosa = Label(self.frame,
                                foreground="black", font=("arial", 14, "bold"),
                                background="light cyan", text="La marca de
                                gaseosa más vendida es:").place(x=700, y=450)
    self.label_gaseosa_val = Label(self.frame,
                                    foreground="black", font=("arial", 14),
                                    background="light
                                    cyan", text=maxima).place(x=850, y=500)

def button(self):
    self.back = Button(self.frame, text = "ATRÁS", bg = "red3", fg =
                        "white", font = ("courier new", 17, "bold"), border = 5,
                        command = self.frame.destroy)
    self.back.place(x=950, y=580)

def reporte_gaseosa():
    root = Toplevel()
    root.title("VENTA DE GASEOSAS")
    root.config(background="light cyan")
    root.geometry("1100x650")
    Label(root, foreground="black", font=("arial", 17, "bold"),
          background="light cyan", text="VENTA DE GASEOSAS").pack(pady = 20)
    App2(root)

```



*Figura 10. Reporte de venta de gaseosas*

#### . REPORTE – DETALLE CLIENTES:

```
class App3():
    def __init__(self, master):
        self.frame = master
        self.label()
        self.entry()
        self.button()

    def label(self):
        self.label_inicio = Label(self.frame,
                                   foreground="black", font=("arial", 14, "bold"), background="light
                                   cyan", text="Desde (YYYY-MM-DD):").place(x=50, y=100)
        self.label_fin = Label(self.frame,
                                foreground="black", font=("arial", 14, "bold"), background="light
                                cyan", text="Fin (YYYY-MM-DD):").place(x=50, y=200)

    def formatear_fecha(self, event, entry):
        self.fecha_str = entry.get()
        try:
            fecha = datetime.strptime(self.fecha_str, "%Y-%m-%d")
            fecha_mysql = fecha.strftime("%Y-%m-%d")
            entry.delete(0, END)
            entry.insert(0, fecha_mysql)
        except ValueError:
            print("Error: Formato de fecha incorrecto")
```

```

def entry(self):
    self.entry_fecha_inicio = Entry(self.frame, font=('Arial',
        12), relief="flat", background="white")
    self.entry_fecha_inicio.place(x=300, y=100, height=25, width=150)
    self.entry_fecha_inicio.bind('<FocusOut>', lambda event:
        self.formatear_fecha(event, self.entry_fecha_inicio))
    self.entry_fecha_fin = Entry(self.frame, font=('Arial',
        12), relief="flat", background="white")
    self.entry_fecha_fin.place(x=300, y=200, height=25, width=150)
    self.entry_fecha_fin.bind('<FocusOut>', lambda event:
        self.formatear_fecha(event, self.entry_fecha_fin))

def obtener_fecha_inicio(self):
    return self.entry_fecha_inicio.get()

def obtener_fecha_fin(self):
    return self.entry_fecha_fin.get()

def button(self):
    self.generar = Button(self.frame, text = "GENERAR", bg = "blue4",
        fg = "white", font = ("courier new", 17, "bold"), border = 5,
        command=self.abrir_reporte)
    self.generar.place(x=180, y=300)

def abrir_reporte(self):
    reporte_cliente_2(self)

class App4():
    def __init__(self, master, app3_instance):
        self.frame = master
        self.cursor = self.conn.cursor()
        self.tabla_cliente()
        self.button()

```

```

def tabla_cliente(self):
    self.list_elems = ttk.Treeview(self.frame, columns=(1,
        2,3,4,5,6,7,8), show="headings", height= "5")

    style = ttk.Style()
    style.theme_use("clam")
    style.configure("Treeview.Heading",
        background="Royalblue4",relief="flat",foreground="white")

    self.list_elems.heading(1, text="ID")
    self.list_elems.heading(2, text="CLIENTE")
    self.list_elems.heading(3, text="NOMBRE DE TIENDA")
    self.list_elems.heading(4, text="CALLE")
    self.list_elems.heading(5, text="NUMERO")
    self.list_elems.heading(6, text="RUC")
    self.list_elems.heading(7, text="TELÉFONO")
    self.list_elems.heading(8, text="FECHA")

    self.list_elems.column(1, anchor=CENTER, width=150)
    self.list_elems.column(2, anchor=CENTER, width= 150)
    self.list_elems.column(3, anchor=CENTER, width=150)
    self.list_elems.column(4, anchor=CENTER, width= 150)
    self.list_elems.column(5, anchor=CENTER, width=150)
    self.list_elems.column(6, anchor=CENTER, width= 150)
    self.list_elems.column(7, anchor=CENTER, width=150)
    self.list_elems.column(8, anchor=CENTER, width= 150)

    fecha_inicio = self.app3_instance.obtener_fecha_inicio()
    fecha_fin = self.app3_instance.obtener_fecha_fin()

    self.rows = self.cliente(fecha_inicio, fecha_fin)
    for i in self.rows:
        self.list_elems.insert('', 'end', values=i)

    self.list_elems.place(x=50, y=70)

    scrollbar = ttk.Scrollbar(self.frame, orient="vertical",
        command=self.list_elems.yview)
    scrollbar.place(x=1250, y=70, height=130)
    self.list_elems.configure(yscrollcommand=scrollbar.set)

```



```

def button(self):
    self.back = Button(self.frame, text = "ATRÁS", bg = "red3", fg =
        "white", font = ("courier new", 17, "bold"), border = 5,
        command = self.frame.destroy)
    self.back.place(x=1150,y=220)

def reporte_cliente():
    root = Toplevel()
    root.title("DETALLES CLIENTE")
    root.config(background="light cyan")
    root.geometry("500x400")
    Label(root, foreground="black",font=("arial",17,"bold"),
        background="light cyan",text="DETALLES DE CLIENTES").pack(pady = 20)
    app3_instance = App3(root)

def reporte_cliente_2(app3_instance):
    root = Toplevel()
    root.title("DETALLE DE CLIENTES")
    root.config(background="light cyan")
    root.geometry("1300x300")
    Label(root, foreground="black",font=("arial",19,"bold"),
        background="light cyan",text="DETALLES DE CLIENTES").pack(pady = 20)
    App4(root, app3_instance)

```



*Figura 11. Detalle de clientes*

DETALLES DE CLIENTES							
ID	CLIENTE	NOMBRE DE TIENDA	CALLE	NUMERO	RUC	TELÉFONO	FECHA
1	ROYER	PEPITO	SAN ANDRES	241	12345678945	949652317	2024-01-29 15:42:07
2	LUIS	COLIBRI	AV ESPANA	789	45678912355	949785456	2024-01-29 15:47:02
3	ANDRES	PEDRETE	AV JUAN	785	78945612345	978546321	2024-02-01 00:04:45
4	LUISA	LUCHITA	SAN ANTONIO	784	65478912356	987458732	2024-02-01 15:56:11

ATRÁS

Figura 12. Reporte de clientes

. REPORTE – INGRESOS:

```
class App5():
    def __init__(self, master):
        self.frame = master
        self.label()
        self.entry()
        self.button()

    def label(self):
        self.label_inicio = Label(self.frame,
                                   foreground="black",font=("arial",14,"bold"), background="light
                                   cyan",text="Desde (YYYY-MM-DD):").place(x=50, y=100)
        self.label_fin = Label(self.frame,
                                foreground="black",font=("arial",14,"bold"), background="light
                                cyan",text="Fin (YYYY-MM-DD):").place(x=50, y=200)

    def formatear_fecha(self,event, entry):
        self.fecha_str = entry.get()
        try:
            fecha = datetime.strptime(self.fecha_str, "%Y-%m-%d")
            fecha_mysql = fecha.strftime("%Y-%m-%d")
            entry.delete(0, END)
            entry.insert(0, fecha_mysql)
        except ValueError:
            print("Error: Formato de fecha incorrecto")
```

```

def entry(self):
    self.entry_fecha_inicio = Entry(self.frame, font=('Arial',
        12), relief="flat", background="white")
    self.entry_fecha_inicio.place(x=300, y=100, height=25, width=150)
    self.entry_fecha_inicio.bind('<FocusOut>', lambda event:
        self.formatear_fecha(event, self.entry_fecha_inicio))

    self.entry_fecha_fin = Entry(self.frame, font=('Arial',
        12), relief="flat", background="white")
    self.entry_fecha_fin.place(x=300, y=200, height=25, width=150)
    self.entry_fecha_fin.bind('<FocusOut>', lambda event:
        self.formatear_fecha(event, self.entry_fecha_fin))

def obtener_fecha_inicio(self):
    return self.entry_fecha_inicio.get()

def obtener_fecha_fin(self):
    return self.entry_fecha_fin.get()

def button(self):
    self.generar = Button(self.frame, text = "GENERAR", bg = "blue4", fg
        = "white", font = ("courier new", 17, "bold"), border = 5,
        command=self.abrir_reporte)
    self.generar.place(x=180, y=300)

def abrir_reporte(self):
    reporte_ingresos2(self)

class App6():
    def __init__(self, master, app5_instance):
        self.frame = master
        self.app5_instance = app5_instance
        self.cursor = self.conn.cursor()
        self.tabla_cliente()
        self.label()
        self.button()

    def tabla_cliente(self):
        self.list_elems = ttk.Treeview(self.frame, columns=(1, 2, 3, 4, 5),
            show="headings", height= "8")

```

```

style = ttk.Style()
style.theme_use("clam")
style.configure("Treeview.Heading",
                background="Royalblue4",relief="flat",foreground="white")

self.list_elems.heading(1, text="PEDIDO ID")
self.list_elems.heading(2, text="GASEOSA(six)")
self.list_elems.heading(3, text="CANTIDAD")
self.list_elems.heading(4, text="LITROS")
self.list_elems.heading(5, text="PRECIO TOTAL")
self.list_elems.column(1, anchor=CENTER, width=150)
self.list_elems.column(2, anchor=CENTER, width= 150)
self.list_elems.column(3, anchor=CENTER, width=150)
self.list_elems.column(4, anchor=CENTER, width= 150)
self.list_elems.column(5, anchor=CENTER, width=150)

self.fecha_inicio = self.app5_instance.obtener_fecha_inicio()
self.fecha_fin = self.app5_instance.obtener_fecha_fin()
self.rows = self.cliente(self.fecha_inicio, self.fecha_fin)
for i in self.rows:
    self.list_elems.insert('', 'end', values=i)
self.list_elems.place(x=80, y=120)

scrollbar = ttk.Scrollbar(self.frame, orient="vertical",
                          command=self.list_elems.yview)
scrollbar.place(x=835, y=120, height=190)
self.list_elems.configure(yscrollcommand=scrollbar.set)

def label(self):
    self.label_desde = Label(self.frame,
                             foreground="black",font=("arial",14,"bold"), background="light
                             cyan",text=f"Desde: {self.fecha_inicio}").place(x=80, y=70)
    self.label_hasta = Label(self.frame,
                             foreground="black",font=("arial",14,"bold"), background="light
                             cyan",text=f"Hasta: {self.fecha_fin}").place(x=400, y=70)
    ingreso_total = self.ingreso(self.fecha_inicio, self.fecha_fin)
    self.label_ingreso =
        Label(self.frame,foreground="black",font=("arial",14,"bold"),
              background="light cyan",text=f"Ingreso total en esa fecha:
              {ingreso_total}").place(x=300, y=340)

```

```

def button(self):
    self.back = Button(self.frame, text = "ATRÁS", bg = "red3", fg =
        "white", font = ("courier new", 17, "bold"), border = 5,
        command = self.frame.destroy)
    self.back.place(x=780,y=330)
def reporte_ingresos():
    root = Toplevel()
    root.title("DETALLES CLIENTE")
    root.config(background="light cyan")
    root.geometry("500x400")
    Label(root, foreground="black",font=("arial",17,"bold"),
        background="light cyan",text="DETALLES DE CLIENTES").pack(pady = 20)
    app5_instance = App5(root)

def reporte_ingresos2(app5_instance):
    root = Toplevel()
    root.title("INGRESOS")
    root.config(background="light cyan")
    root.geometry("900x400")
    Label(root, foreground="black",font=("arial",19,"bold"),
        background="light cyan",text="DETALLES DE INGRESOS").pack(pady = 20)
    App6(root, app5_instance)

```



**DETALLES DE INGRESOS**

**Desde: 2024-01-29                      Hasta: 2024-02-01**

PEDIDO ID	GASEOSA(six)	CANTIDAD	LITROS	PRECIO TOTAL
1.1	GUARANA	5	0.45	59.0
1.2	KR	6	0.35	53.1
2.1	KR	3	0.35	26.55
2.2	PEPSI	4	0.5	54.28
2.3	COCACOLA	2	0.5	31.86
3.1	COCACOLA	8	0.5	127.44
3.2	INKACOLA	10	0.5	165.2
3.3	PEPSI	7	0.5	94.99

**Ingreso total en esa fecha: 870.84**

**ATRÁS**

*Figura 13. Reporte de ingresos*

. PÁGINA DE REGISTRO DE CLIENTE:

```
class App:
    def __init__(self, master):
        self.frame = master
        self.DrawEntry()
        self.DrawButtons()
        self.DrawLabel()

    def DrawLabel(self):
        self.lbl_name = Label(self.frame,
                               foreground="black",font=("arial",13,"bold"), background="light
                               cyan",text="Nombre:").place(x=50, y=100)
        self.lbl_place = Label(self.frame,
                               foreground="black",font=("arial",13,"bold"), background="light
                               cyan", text="Dirección:").place(x=50, y=170)
        self.lbl_streat = Label(self.frame,
                                foreground="black",font=("arial",12), background="light cyan",
                                text="- Calle:").place(x=90, y=220)
        self.lbl_streat_num = Label(self.frame,
                                    foreground="black",font=("arial",12), background="light cyan",
                                    text="- Número:").place(x=90, y=290)
        self.lbl_mname = Label(self.frame,
                                foreground="black",font=("arial",13,"bold"), background="light
                                cyan", text="Nombre de Tienda:").place(x=50, y=360)
        self.lbl_ruc = Label(self.frame,
                              foreground="black",font=("arial",13,"bold"), background="light
                              cyan", text="RUC:").place(x=50, y=430)
        self.lbl_phone = Label(self.frame,
                                foreground="black",font=("arial",13,"bold"), background="light
                                cyan", text="Teléfono:").place(x=50, y=500)

    def DrawEntry(self):
        self.name = StringVar()
        self.streat = StringVar()
        self.streat_num = IntVar()
        self.mname = StringVar()
        self.ruc = StringVar()
        self.phone = IntVar()
```

```

self.txt_name = Entry(self.frame,font=('Arial', 12),relief="flat",
    background="White" ,textvariable=self.name)
self.txt_name.place(x=200, y=100, height=25, width=250)
self.txt_streat = Entry(self.frame,font=('Arial', 12),relief="flat",
    background="White" ,textvariable=self.streat)
self.txt_streat.place(x=200, y=220, height=25, width=250)
self.txt_streat_num = Entry(self.frame,font=('Arial',
    12),relief="flat", background="White"
    ,textvariable=self.streat_num)
self.txt_streat_num.place(x=200, y=290, height=25, width=250)
self.txt_mname = Entry(self.frame,font=('Arial', 12),relief="flat",
    background="White" ,textvariable=self.mname)
self.txt_mname.place(x=215, y=360, height=25, width=240)
self.txt_ruc = Entry(self.frame,font=('Arial', 12),relief="flat",
    background="White" ,textvariable=self.ruc)
self.txt_ruc.place(x=200, y=430, height=25, width=250)
self.txt_phone = Entry(self.frame,font=('Arial', 12),relief="flat",
    background="White" ,textvariable=self.phone)
self.txt_phone.place(x=200, y=500, height=25, width=250)

self.txt_name.bind("<KeyRelease>", self.convertir_a_mayusculas)
self.txt_streat.bind("<KeyRelease>", self.convertir_a_mayusculas)
self.txt_mname.bind("<KeyRelease>", self.convertir_a_mayusculas)

def convertir_a_mayusculas(self,event):
    nuevo_valor = self.txt_name.get().upper()
    self.name.set(nuevo_valor)
    nuevo_valor = self.txt_streat.get().upper()
    self.streat.set(nuevo_valor)
    nuevo_valor = self.txt_mname.get().upper()
    self.mname.set(nuevo_valor)
def DrawButtons(self):
    self.btn_save = Button(self.frame,foreground="white",
        text="Guardar",borderwidth=2,relief="flat",
        cursor="hand1",overrelief="raise",background="blue4",
        command=self.confirmProcess).place(x=280, y=580, width=90)
    self.btn_continue = Button(self.frame,foreground="white",
        text="Siguiete",borderwidth=2,relief="flat",
        cursor="hand1",overrelief="raise",background="blue4",
        command=lambda:[self.frame.destroy(),ventana_ped2()]).place(x=380,
        y=580, width=90)

```

```

def confirmProcess(self):
    if self.name.get() != "" and self.streat.get() != "" and
        self.streat_num.get() != "" and self.mname.get() != "" and
        self.ruc.get() != "" and self.phone.get() != "":
        d = Data()
        arr = [self.name.get(), self.streat.get(), self.streat_num.get(),
            self.mname.get(), self.ruc.get(), self.phone.get()]
        d.InsertItems(arr)
        messagebox.showinfo(title="Gracias", message="Sus datos han sido
            registrados, puede continuar.")
    else:
        messagebox.showinfo(title="Error", message="Debe llenar los
            campos para poder seguir!")
        self.frame.focus_force()
def ventana_ped():
    root = Toplevel()
    root.title("REALIZAR PEDIDO")
    root.config(background="light cyan")
    root.geometry("500x650")
    Label(root, text="Ingrese sus datos", fg = "black", bg = "light cyan", font
        = ("courier new", 20, "bold")).place(x=30, y = 30)
    App(root)

```

*Figura 14. Ingreso de datos del cliente*



. PÁGINA DE PEDIDO:

```
class App2:
    def __init__(self, master):
        self.frame = master
        self.DrawEntry()
        self.DrawList()
        self.DrawLabel()
        self.DrawRadioButtons()
        self.i = 1

    def DrawLabel(self):
        self.lbl_name = Label(self.frame,
                               foreground="black",font=("arial",13,"bold"), background="light
                               cyan",text="Gaseosa:").place(x=300, y=300)
        self.lbl_six = Label(self.frame,
                              foreground="black",font=("arial",13,"bold"), background="light cyan",
                              text="Cantidad:").place(x=300, y=350)
        self.lbl_liter = Label(self.frame,
                                foreground="black",font=("arial",13,"bold"), background="light cyan",
                                text="Litros:").place(x=300, y=400)
        self.lbl_question = Label(self.frame,
                                   foreground="black",font=("arial",13,"bold"), background="light cyan",
                                   text="Desea agregar otra gaseosa?").place(x=200, y=450)

    def DrawEntry(self):
        self.name = StringVar()
        self.six = IntVar()
        self.liter = StringVar()
        self.txt_name = Entry(self.frame,font=('Arial', 12),relief="flat",
                               background="CadetBlue2" ,textvariable=self.name)
        self.txt_name.place(x=400, y=300, height=25, width=150)
        self.txt_six= Entry(self.frame,font=('Arial', 12),relief="flat",
                              background="CadetBlue2" ,textvariable=self.six)
        self.txt_six.place(x=400, y=350, height=25, width=150)
        self.txt_liter= Entry(self.frame,font=('Arial', 12),relief="flat",
                               background="CadetBlue2" ,textvariable=self.liter)
        self.txt_liter.place(x=400, y=400, height=25, width=150)

        self.txt_name.bind("<KeyRelease>", self.convertir_a_mayusculas)
```

```

def convertir_a_mayusculas(self,event):
    nuevo_valor = self.txt_name.get().upper()
    self.name.set(nuevo_valor)
def DrawRadioButtons(self):
    self.opcion = IntVar()
    self.yes = Radiobutton(self.frame, text = "Si",font=('Arial', 12), bg =
        "light cyan",variable= self.opcion, value = 1, command=self.Save).
        place(x= 340, y = 490)
    self.no = Radiobutton(self.frame, text = "No",font=('Arial', 12), bg =
        "light cyan", variable= self.opcion, value = 2,command = self.Save).
        place(x= 540, y = 490)

def ClearEntry(self):
    self.name.set("")
    self.six.set("")
    self.liter.set("")

def confirmProcess2(self):

    if self.name.get() != "" and self.six.get() != ""and self.liter.get()
        != "":
        d = Data2()
        arr = [self.name.get(), self.six.get(),self.liter.get()]
        d.InsertItems2(arr,self.i)
        d.UpdateSix(self.liter.get(),self.six.get(),self.name.get())

def Save(self):
    if self.opcion.get() == 1 :
        self.confirmProcess2()
        self.i+=1
        self.DrawList()
        self.ClearEntry()
        self.DrawLabel()
        self.ClearEntry()
        self.DrawRadioButtons()
    elif self.opcion.get() == 2 :
        self.confirmProcess2()
        self.i=1
        self.DrawList()
        self.frame.destroy()
        ventana_ped3()

```

```

def DrawList(self):
    self.list_elems = ttk.Treeview(self.frame, columns=(1, 2, 3, 4),
                                   show="headings", height="8")

    style = ttk.Style()
    style.theme_use("clam")
    style.configure("Treeview.Heading",
                    background="Royalblue4", relief="flat", foreground="white")

    self.list_elems.heading(1, text="Nombre")
    self.list_elems.heading(2, text="Cantidad(six pack)")
    self.list_elems.heading(3, text="Precio")
    self.list_elems.heading(4, text="Capacidad (L)")
    self.list_elems.column(1, anchor=CENTER)
    self.list_elems.column(2, anchor=CENTER)
    self.list_elems.column(3, anchor=CENTER)
    self.list_elems.column(4, anchor=CENTER)

    d = Data2()
    self.rows = d.returnAllElements()
    for i in self.rows:
        self.list_elems.insert('', 'end', values=i)

    self.list_elems.place(x=50, y=80)
    scrollbar = ttk.Scrollbar(self.frame, orient="vertical",
                              command=self.list_elems.yview)
    scrollbar.place(x=850, y=80, height=190)
    self.list_elems.configure(yscrollcommand=scrollbar.set)

def ventana_ped2():
    root = Toplevel()
    root.title("REALIZAR PEDIDO")
    root.config(background="light cyan")
    root.geometry("900x550")
    Label(root, text="Pedido:", fg = "black", bg = "light cyan", font = ("courier
        new", 20, "bold")).place(x=30, y = 30)
    App2(root)

```

**Pedido:**

Nombre	Cantidad(six pack)	Precio	Capacidad (L)
COCACOLA	25	15.0	1.5
GUARANA	8	10.0	0.45
PEPSI	8	11.5	0.5
INKACOLA	33	14.0	0.5
KR	14	7.5	0.35
COCACOLA	13	11.0	0.5

**Gaseosa:** PEPSI

**Cantidad:** 4

**Litros:** 0.5

**Desea agregar otra gaseosa?**

☐ Si ☐ No

*Figura 15. Pedido*

. RESUMEN DE PEDIDO:

```
def ventana_ped3():
    root = Toplevel()
    root.title("REALIZAR PEDIDO")
    root.config(background="light cyan")
    root.grid_columnconfigure(0, weight=1)

    lbl_tittle = Label(root, text="RESUMEN DE PEDIDO", fg = "black", bg =
        "light cyan",font = ("courier new", 20, "bold"))
    lbl_tittle.grid(row=0, column=0,columnspan=5, sticky="nsew",pady=(20, 0))
    for col in range(5):
        root.grid_columnconfigure(col, weight=1)
    spacer = Label(root, text="", bg="light cyan")
    spacer.grid(row=1, column=0)
    lbl_name = Label(root, foreground="black", font=("arial", 13, "bold"),
        background="light cyan", text="Gaseosa").grid(row=2, column=0,
        sticky="nsew")
    lbl_six = Label(root, foreground="black", font=("arial", 13, "bold"),
        background="light cyan", text="Cantidad").grid(row=2, column=1,
        sticky="nsew")
```

```

lbl_liter = Label(root, foreground="black", font=("arial", 13, "bold"),
    background="light cyan", text="Capacidad(L)").grid(row=2, column=2,
    sticky="nsew")
lbl_price = Label(root, foreground="black", font=("arial", 13, "bold"),
    background="light cyan", text="Precio (u)").grid(row=2, column=3,
    sticky="nsew")
lbl_pricet = Label(root, foreground="black", font=("arial", 13, "bold"),
    background="light cyan", text="Precio Total (s/.)").grid(row=2,
    column=4, sticky="nsew")
spacer = Label(root, text="", bg="light cyan")
spacer.grid(row=3, column=0)
d = Data2()
order = d.resumen()

row_position = 4

for i in order:
    col_position = 0
    for j in i[1:]:
        lbl_order = Label(root, foreground="black", font=("arial", 12),
            background="light cyan", text=j).grid(row=row_position,
            column=col_position, sticky="nsew")
        spacer = Label(root, text="", bg="light cyan")
        spacer.grid(row=row_position+1, column=0)
        col_position += 1

    row_position += 2

subtotal = d.Subtotal()
val_subtotal = 0

row_position_1 = 4

```

```

for i in subtotal:
    for j in i:
        lbl_order = Label(root, foreground="black", font=("arial", 12),
                           background="light cyan", text=j).grid(row=row_position_1,
                           column=4, sticky="nsew")
        spacer = Label(root, text="", bg="light cyan")
        spacer.grid(row=row_position+1, column=0)
        val_subtotal+= j
    row_position_1 += 2

spacer = Label(root, text="", bg="light cyan")
spacer.grid(row=row_position, column=0)
lbl_line = Label(root, foreground="black", font=("arial", 13),
                  background="light cyan", text="-----
                  ").grid(row=row_position, column=4, sticky="s")

lbl_subtotal = Label(root, foreground="black", font=("arial", 13, "bold"),
                     background="light cyan", text="SubTotal:").grid(row=row_position+1,
                     column=3, sticky="e")
lbl_pay = Label(root, foreground="black", font=("arial", 13, "bold"),
                background="light cyan", text=val_subtotal).grid(row=row_position+1,
                column=4, sticky="nsew")

igv = d.igv()
lbl_igv = Label(root, foreground="black", font=("arial", 13),
                background="light cyan", text="IGV (18%):").grid(row=row_position+2,
                column=3, sticky="e")
lbl_igv_value = Label(root, foreground="black", font=("arial", 13),
                      background="light cyan", text=igv).grid(row=row_position+2,
                      column=4, sticky="nsew")

total = d.Total()
lbl_total = Label(root, foreground="black", font=("arial", 13, "bold"),
                  background="light cyan", text="Pago
                  Total:").grid(row=row_position+3, column=3, sticky="e")
lbl_total_value = Label(root, foreground="black", font=("arial", 13,
                  "bold"), background="light cyan",
                  text=total).grid(row=row_position+3, column=4, sticky="nsew")

```

```

spacer = Label(root, text="", bg="light cyan")
spacer.grid(row=row_position+4, column=0)
spacer = Label(root, text="", bg="light cyan")
spacer.grid(row=row_position+5, column=0)
button_save = Button(root, foreground="white", text="CONFIRMAR PEDIDO",
                      borderwidth=2, relief="flat", cursor="hand1", overrelief="raise",
                      background="SeaGreen3", width=17, height=1, command=
                      lambda:[root.destroy(), factura()]).grid(row=row_position+6,
                      column=4)

y = row_position*60
root.geometry(f"1100x{y}")

```

RESUMEN DE PEDIDO				
Gaseosa	Cantidad	Capacidad(L)	Precio (u)	Precio Total (s/.)
GUARANA	3	0.45	10.0	30.0
KR	6	0.35	7.5	45.0
PEPSI	4	0.5	11.5	46.0
			<b>SubTotal:</b>	<b>121.0</b>
			IGV (18%):	21.78
			<b>Pago Total:</b>	<b>142.78</b>
				<a href="#">CONFIRMAR PEDIDO</a>

*Figura 15. Resumen de pedido*

. FACTURA:

```
def factura():
    root = Toplevel()
    root.title("FACTURA")
    root.config(background="white")

    d = Data()
    d_2 = Data2()

    lbl_tittle = Label(root, text="FACTURA", fg = "black", bg = "white",font =
        ("courier new", 14)).pack(pady=10)
    lbl_tittle2 = Label(root, text="DISTRIBUIDORA DE GASEOSAS", fg = "black",
        bg = "white",font = ("courier new", 22, "bold")).pack()
    gaseosa_ruc = Label(root, text="RUC: 45236548719", fg = "black", bg =
        "white",font = ("courier new", 12, "bold")).place(x=900, y = 55)

    fecha_label = Label(root, text="", font=("Arial", 8), bg="white")
    fecha_label.place(x=900, y=15)
    fecha = d_2.fecha()
    fecha_label.config(text=f"Fecha y Hora: {fecha}")

    cliente_name = Label(root, text="Nombre: ", fg = "black", bg = "white",font
        = ("courier new", 18, "bold")).place(x=30, y = 100)
    cliente_mname = Label(root, text="Nombre de Tienda:", fg = "black", bg =
        "white",font = ("courier new", 18, "bold")).place(x=30, y = 130)
    cliente_ruc = Label(root, text="RUC:", fg = "black", bg = "white",font =
        ("courier new", 18, "bold")).place(x=30, y = 160)
    cliente_phone = Label(root, text="Teléfono:", fg = "black", bg =
        "white",font = ("courier new", 18, "bold")).place(x=30, y = 190)
    cliente_place = Label(root, text="Dirección: ", fg = "black", bg =
        "white",font = ("courier new", 18, "bold")).place(x=30, y = 220)

    global imagen
    imagen = PhotoImage(file="gaseosas2.png")
    lbl_imagen = Label(root, image=imagen, bg = "white")
    lbl_imagen.place(x =850,y =150)
```



```

cliente = d.ReturnElements()
n_factura = Label(root, text="#Factura: ", fg = "black", bg = "white",font
    = ("courier new", 16, "bold")).place(x=820, y = 100)
valor_factura =Label(root, text=cliente[0], fg = "black", bg = "white",font
    = ("courier new", 16)).place(x=980, y = 100)

d_y = 100
for i in cliente[1:5]:
    cliente_dato = Label(root, text=i, fg = "black", bg = "white",font =
        ("courier new", 16 )).place(x=300, y = d_y)
    d_y += 30

direccion = d_2.direccion()
d_x = 300
for i in direccion:
    cliente_direccion = Label(root, text=i, fg = "black", bg = "white",font
        = ("courier new", 16 )).place(x=d_x, y = 220)
    d_x += 180

list_elems = ttk.Treeview(root, columns=(1, 2, 3, 4,5), show="headings",
    height="8")

style = ttk.Style()
style.theme_use("clam")
style.configure("Treeview.Heading",
    background="black",relief="flat",foreground="white", font = ("courier
    new",12, "bold"))

list_elems.heading(1, text="GASEOSA")
list_elems.heading(2, text="CANTIDAD(six pack)")
list_elems.heading(3, text="CAPACIDAD(L)")
list_elems.heading(4, text="PRECIO(u)")
list_elems.heading(5, text="PRECIO")
list_elems.column(1, anchor=CENTER)
list_elems.column(2, anchor=CENTER)
list_elems.column(3, anchor=CENTER)
list_elems.column(4, anchor=CENTER)
list_elems.column(5, anchor=CENTER)

```

```

rows = d_2.factura()
for i in rows:
    list_elems.insert('', 'end', values=i)

subtotal = d_2.Subtotal()
subtotal_val=0
filas = list_elems.get_children()

for fila, valores in zip(filas, subtotal):
    valor_columna_5 = valores[-1]
    list_elems.item(fila, values=[*list_elems.item(fila)['values'][:4],
        valor_columna_5])
    subtotal_val += valor_columna_5

num_rows = len(list_elems.get_children())
list_elems["height"] = num_rows
list_elems.pack()
list_elems.place(x=50, y=300)

lbl_subtotal = Label(root, foreground="black", font=("arial", 13, "bold"),
    background="white", text="SubTotal:").place(x=850, y = 330+50*num_rows)
lbl_subtotal_val = Label(root, foreground="black", font=("arial", 13,
    "bold"), background="white", text=subtotal_val).place(x=950, y =
    330+50*num_rows)
lbl_igv = Label(root, foreground="black", font=("arial", 13, "bold"),
    background="white", text="IGV(18%):").place(x=850, y = 370+50*num_rows)
igv = d_2.igv()
lbl_igv_val = Label(root, foreground="black", font=("arial", 13, "bold"),
    background="white", text=igv).place(x=950, y = 370+50*num_rows)

lbl_total = Label(root, foreground="black", font=("arial", 13, "bold"),
    background="white", text="Pago Total:").place(x=850, y =
    410+50*num_rows)
total = d_2.Total()
lbl_total_val = Label(root, foreground="black", font=("arial", 13, "bold"),
    background="white", text=total).place(x=950, y = 410+50*num_rows)
lbl_thank = Label(root, foreground="black", font=("arial", 9, "italic"),
    background="white", text="Gracias por su
    compra!!!").pack(side="bottom", pady=10, anchor="s")
y = 400+num_rows*100
root.geometry(f"1100x{y}")

```

## DISTRIBUIDORA DE GASEOSAS

RUC: 45236548719

Nombre: MARIO  
Nombre de Tienda: LOS SAPITOS  
RUC: 45632187956  
Teléfono: 949758612  
Dirección: PEDRO RUIZ 785

#Factura: 7



GASEOSA	CANTIDAD (six pack)	CAPACIDAD (L)	PRECIO (u)	PRECIO
GUARANA	3	0.45	10.0	30.0
KR	6	0.35	7.5	45.0
PEPSI	4	0.5	11.5	46.0

SubTotal: 121.0

IGV(18%): 21.78

Pago Total: 142.78

Gracias por su compra!!!

Figura 16. Factura

## 5. Conclusiones

1. Se logró desarrollar la interfaz robusta que proporciona a la Distribuidora de Gaseosas una herramienta eficaz para gestionar su inventario y ventas de manera más eficiente. Al integrar Python y MySQL, hemos creado un sistema que facilita la visualización y actualización de datos en tiempo real, mejorando significativamente los procesos de administración.
2. Se ha logrado establecer la integración del sistema de gestión de bases de datos MySQL garantizando la precisión y disponibilidad de la información almacenada en la Distribuidora de Gaseosas, gracias a ello, los datos se gestionan de manera

segura y eficiente, lo que proporciona una base sólida para la toma de decisiones informadas y la optimización de los procesos internos.

3. La implementación de tkinter ha permitido diseñar una interfaz intuitiva y eficiente que automatiza procesos y facilita la visualización de datos relevantes. Esta herramienta ha mejorado la experiencia del usuario al interactuar con el sistema, proporcionando una plataforma que simplifica la toma de decisiones y aumenta la productividad en la Distribuidora de Gaseosas.
4. Se han creado con éxito reportes detallados de inventario, clientes e ingresos utilizando Python. Estos reportes proporcionan una visión completa de la situación financiera y operativa de la Distribuidora de Gaseosas, permitiendo una mejor comprensión de los datos y facilitando la toma de decisiones estratégicas. El uso de Python ha demostrado ser una herramienta poderosa para analizar y presentar información clave de manera clara y concisa.

## 6. Referencias bibliográficas:

1. Espinoza Roggero. (2019). *UNIVERSIDAD SEÑOR DE SIPÁN*. Edu.pe. Recuperado el 7 de febrero de 2024, de <https://repositorio.uss.edu.pe/bitstream/handle/20.500.12802/7250/Espinoza%20Roggero%20Gabriela.pdf>
2. (S/f). La empresa y su organización. Mheducation.es. Recuperado el 7 de febrero de 2024, de <https://www.mheducation.es/bcv/guide/capitulo/8448199359.pdf>
3. Donaire, Manuel. (2006). La empresa: concepto, elementos, funciones y clase. <https://www3.gobiernodecanarias.org/medusa/ecoblog/cperpad/files/2012/05/tema1empresa.pdf>
4. Marcial. (2021). La empresa. <https://www.marcialpons.es/media/pdf/book-attachment-6089.pdf>

5. Gonzáles,D.(2008).Python para todos.  
<https://persoal.citius.usc.es/eva.cernadas/informaticaparacientificos/material/libros/Python%20para%20todos.pdf>
- 6.Montalvo.(2018).Interfaces gráficas en Java.  
<https://www.cerasa.es/media/areces/files/book-attachment-3027.pdf>
- 7.Cruz,C.(2011).Base de datos,conceptos y sus características.  
<http://www.gridmorelos.uaem.mx/~mcruz/cursos/miic/bd1.pdf>