



Tecnicatura Superior en Ciencia de Datos e Inteligencia Artificial

Modulo: Innovacion y Gestión de Datos

PROYECTO FINAL

Grupo:

Hebe Carolina Pereyra,
Jesica Valdivia
Mariela Rosa Suarez
Victoria Margarita Picco
Ulises Martinelli
Consuelo Viramonte,

Nombre del proyecto: Control de Stock a Granel

Link al repositorio en GitHub:

<https://github.com/ISPC-PROYECTOS/InnovGestDatos>

El Proyecto

El proyecto intenta desarrollar una aplicación para la gestión de datos, centrada en el control de stock de productos en un negocio que comercializa productos comestibles *Pachamama*, ubicado en la ciudad de La Falda. Este negocio recibe muchos de sus productos a granel y los fracciona en presentaciones más pequeñas. La solución busca gestionar el inventario de manera eficiente y proporcionar capacidades de análisis de datos.

La aplicación, desarrollada en Python, incluye un módulo de gestión de usuarios, para que cada sección tenga un responsable principal. En ese módulo se encuentran las operaciones CRUD sobre una base de datos MySQL. A través de la aplicación, los usuarios pueden gestionar proveedores, productos, presentaciones y ventas, lo cual facilita la toma de decisiones en el manejo del inventario.

A pedido de la cátedra se desarrolló también módulos de registros pluviales mediante manejo de archivos y las librerías **Pandas**, **Numpy** y **Matplotlib**, elaborando código para diferentes consultas y gráficos.

Justificación

El proyecto es relevante porque aborda una necesidad crítica en la gestión de inventario de un microemprendimiento que comercializa productos naturales a granel. Este negocio necesita una herramienta que le permita controlar de manera precisa la cantidad de productos disponibles, optimizar el almacenamiento y prevenir pérdidas por vencimientos, como así también prevenir pérdidas económicas por ventas no realizadas debido a falta de stock. Proporciona una solución económica y sencilla para gestionar su inventario. Contribuye al fortalecimiento de pequeños emprendimientos, ayudando a gestionar los recursos de manera eficiente y sostenible.

A la vez, facilita el aprendizaje y la aplicación de técnicas de programación, gestión de bases de datos, y habilidades de análisis.

Para justificar esta necesidad, se realizaron entrevistas a una empleada del negocio (que pertenece a este grupo de trabajo) que señaló la dificultad de gestionar productos a granel, especialmente en lo que respecta a la cantidad disponible y el seguimiento de fechas de vencimiento.

La inclusión de un módulo para manejo de usuarios, de archivos binarios y de las librerías **Pandas**, **Numpy** y **Matplotlib**, enriquece al proyecto y da herramientas a los desarrolladores.

Objetivos

Objetivo General

Desarrollar un sistema de gestión de stock a granel y su fraccionamiento, que optimice la administración del inventario en este negocio pero que

puede usarse para negocios similares, integrando el registro, análisis y visualización de datos y así mejorar la eficiencia operativa.

Objetivos Específicos

1. Diseñar y construir una base de datos en MySQL que almacene información relevante sobre productos, proveedores y ventas.
2. Utilizar sentencias de **SQL** para la gestión de bases de datos y así facilitar el acceso a la información de manera eficiente.
3. Desarrollar una aplicación en Python que permita a los usuarios gestionar el inventario de manera intuitiva aplicando los principios de **Programación Orientada a Objetos (POO)** para desarrollar los módulos clave del sistema.
4. Reforzar la seguridad y la integridad del sistema con buenas prácticas de programación.
5. Implementar herramientas de análisis de datos y visualización utilizando librerías como Pandas y Matplotlib para ayudar a los usuarios a tomar decisiones informadas.

Metodología Aplicada

Para alcanzar los objetivos propuestos, en cada una de las evidencias fuimos organizando y realizando reuniones para ponernos de acuerdo, comunicar los avances y dificultades presentadas. En estas reuniones propusimos objetivos específicos y dividimos tareas. Luego, íbamos haciendo un seguimiento casi diario en forma de mensajes de whatsapp. En la última entrega, aplicamos la metodología SCRUM que aprendimos en el ISPC en este último cuatrimestre.

El proyecto fue desarrollado por un equipo de seis estudiantes, dividiendo las tareas según capacidades (gestión de usuarios, control de stock, base de datos, etc.). Se utilizó Python para la programación de la aplicación y MySQL para la gestión de la base de datos. También se aprovechó el uso de herramientas colaborativas como GitHub para las versiones del código y la documentación. Como se emplearon herramientas, software libre y nuestras computadoras personales y conexión a internet, no se requirieron recursos económicos significativos

Actividades desarrolladas:

Planeación:

- Entrevista a la dueña del local para recaudar información.
- Análisis de los requerimientos y diseño conceptual del sistema.

Diseño:

- Diseño del esquema conceptual de la base de datos mediante DER con notación de Chen identificando las entidades, sus atributos y las relaciones para luego pasar al diseño lógico en un DER con notación Crow's Foot representando las entidades mediante tablas, determinando claves primarias y sus relaciones a través de las claves foráneas.

Desarrollo:

- Transformación de diagramas al modelo físico creando la BBDD en MySQL Workbench, sus tablas.
- Pruebas de funcionamiento correcto del diseño antes de ser usado en la aplicación simulando datos y consultas, y modificaciones necesarias
- Desarrollo de la aplicación en Python y creación del módulo de análisis de datos.
- Conexión de la BBDD con la aplicación mediante código de lenguaje Python.
- Implementación de las funcionalidades CRUD para la base de datos.
- Desarrollo de pruebas para asegurar la funcionalidad del sistema.

Evaluación:

- Pruebas del sistema con datos reales proporcionados por un microemprendimiento local.
- Ajustes y mejoras de la aplicación según los resultados obtenidos.

Cronograma

El cronograma está basado en el calendario académico y los plazos asignados por el módulo "Innovación en Gestión de Datos":

Etapa	Actividad	Duración	Fecha
Planeación	Definición de requerimientos	1 semana	16/8 al 23/08
Diseño	Diseño de la base de datos y la aplicación	2 semanas	24/08 al 06/09
Etapa 1 -Desarrollo	Implementación en Python y MySQL	3 semanas	16/9 al 6/10
Etapa 2-Desarrollo	Implementación gestión de usuarios y archivos	2 semanas	13/10 al 27/10
Pruebas y Ajustes	Pruebas unitarias y de integración	5 días	28/10 al 1/11
Documentación	Capturas de pantalla y documentación final	2 días	31/10 al 1/11

Presentación del Proyecto

Script:

main.py

```
import sys
import os
sys.path.append(os.path.abspath(os.path.join(os.path.dirname(__file__), '..')))

from modulos_main import menu_usuarios
from inicio_sesion import iniciar_sesion
from registros_pluviales import menu_registros_pluviales
from BDII.conexion_bd import conectar_base_datos, ErrorConexion,
cerrar_conexion

def main():
    salir = False
    while not salir:
        print('-----> MENÚ PRINCIPAL <-----')
        print('1. USUARIOS Y ACCESOS')
        print('2. INICIAR SESIÓN')
        print('3. ANÁLISIS DE DATOS')
        print('4. SALIR')
        opcion = input("Seleccione una opción: ")

        if opcion == '1':
            menu_usuarios()
        elif opcion == '2':
            try:
                sesion_iniciada = iniciar_sesion()
                conn, cursor = conectar_base_datos()

                if sesion_iniciada is True:
                    from BDII import index
                    index.menu_stock_granel()
                    cerrar_conexion(conn, cursor)
            except ErrorConexion as e:
                print(e)
                print("Por favor, configure los datos de conexión y
pruebe nuevamente.")
        opcion = input("¿Deseas volver al menú principal?
(s/n) : ")
        if opcion.lower() == 's':
            continue
```

```

        else:
            salir = True
    elif opcion == '3':
        menu_registros_pluviales()
    elif opcion == '4':
        print("Saliendo de la aplicación.")
        salir = True
    else:
        print("Opción no válida, intente nuevamente.")

if __name__ == '__main__':
    main()

```

crud.py

```

from funciones_main import agregar_usuario, modificar_usuario,
eliminar_usuario, buscar_usuario, mostrar_usuarios

class CrudUsuario:
    def __init__(self):
        pass

    def crear_usuario(self, dni, username, email, contrasena):
        agregar_usuario(dni, username, contrasena, email)

    def actualizar_usuario(self, dni, nuevo_username=None,
nuevo_email=None, nueva_contrasena=None):
        modificar_usuario(dni, nuevo_username, nueva_contrasena,
nuevo_email)

    def eliminar_usuario(self, dni):
        eliminar_usuario(dni)

    def buscar_usuario(self, dni):
        return buscar_usuario(dni)

    def mostrar_usuarios(self):
        mostrar_usuarios()

```

Funciones_main.py

```

from gestionUsuario import guardar_usuario,
guardar.todos.los.usuarios,cargar_datos

```

```
from validaciones import validar_email, validar_dni, validar_usuario,
validar_contrasena
from usuario import Usuario


def agregar_usuario():
    dni = input('Ingrese el DNI del usuario: ')
    while not validar_dni(dni):
        print('El dato ingresado es incorrecto, debe contener 7 u 8
números, intentelo de nuevo: ')
        dni = input('Ingrese el DNI del usuario: ')

    username = input("Ingrese el nombre del usuario: ").lower()
    while not validar_usuario(username):
        print('El dato ingresado es incorrecto, debe contener sólo
letras, intentelo nuevamente: ')
        username = input('Ingrese el nombre del usuario: ').lower()

    email = input('Ingrese el email: ').lower()
    while not validar_email(email):
        print('El dato ingresado es incorrecto, intentelo nuevamente:
')
        email = input('Ingrese el email: ').lower()

    contrasena = input("Ingrese la contraseña, mínimo 8 caracteres,
minúsculas y números: ").lower()
    while not validar_contrasena(contrasena):
        print('La contraseña no cumple con los requisitos, intentelo
nuevamente: ')
        contrasena = input('Ingrese la contraseña: ').lower()

    # Crear una instancia de Usuario
    nuevo_usuario = Usuario(dni, username, email, contrasena)

    # Guardar la instancia de Usuario
    guardar_usuario(nuevo_usuario)

    print("Usuario agregado exitosamente.")


def modificar_usuario():
    usuarios = cargar_datos()
    if not usuarios:
        print("No hay usuarios para modificar.")
        return
```

```
usuario_a_modificar = input("Ingrese el DNI del usuario a modificar: ")
for usuario in usuarios:
    if usuario.get_dni() == usuario_a_modificar:
        nuevo_usuario = input("Ingrese el nuevo nombre de usuario:")
    nuevo_email = input("Ingrese el nuevo email: ")
    nueva_contrasena = input("Ingrese la nueva contraseña: ")

    if not validar_usuario(nuevo_usuario):
        print('Usuario no válido.')
        return
    if not validar_email(nuevo_email):
        print("Email no válido.")
        return
    if not validar_contrasena(nueva_contrasena):
        print("Contraseña no válida.")
        return

    usuario.set_username(nuevo_usuario)
    usuario.set_email(nuevo_email)
    usuario.set_contrasena(nueva_contrasena)

    guardar.todos_los_usuarios(usuarios)
    print("Usuario modificado exitosamente.")
    return

print("Usuario no encontrado.")

def eliminar_usuario():
    usuarios = cargar_datos()
    if not usuarios:
        print("No hay usuarios para eliminar.")
        return

    usuario_a_eliminar = input("Ingrese el DNI del usuario a eliminar:")
    usuarios_actualizados = [usuario for usuario in usuarios if
    usuario.get_dni() != usuario_a_eliminar]

    if len(usuarios) == len(usuarios_actualizados):
        print("Usuario no encontrado.")
```

```

else:
    guardar.todos._los_usuarios(usuarios.actualizados)
    print("Usuario eliminado exitosamente.")

def mostrar_usuarios():
    usuarios = cargar_datos()
    if not usuarios:
        print("No hay usuarios registrados.")
        return
    for usuario in usuarios:
        print(f"DNI: {usuario.get_dni()}, Usuario:
{usuario.get_username()}, Email: {usuario.get_email()}\n")

def ordenar_usuarios_burbuja():
    usuarios = cargar_datos()
    for i in range(len(usuarios)):
        for j in range(0, len(usuarios)-i-1):
            if usuarios[j].get_username() >
    usuarios[j+1].get_username():
                usuarios[j], usuarios[j+1] = usuarios[j+1], usuarios[j]
    guardar.todos._los_usuarios(usuarios)
    return usuarios

def ordenar_usuarios_sort():
    usuarios = cargar_datos()
    usuarios_ordenados = sorted(usuarios, key=lambda x:
x.get_username())
    guardar.todos._los_usuarios(usuarios_ordenados)
    return usuarios_ordenados

def buscar_usuario_secuencial(username, usuarios):
    for usuario in usuarios:
        if usuario.get_username() == username:
            print(f"Usuario encontrado: {usuario.mostrar_datos()}")
            print("La búsqueda se realizó utilizando la técnica de
búsqueda secuencial.")
            return usuario
    print("Usuario no encontrado.")
    return None

def buscar_usuario_binaria(username, usuarios):
    inicio = 0
    fin = len(usuarios) - 1

```

```

while inicio <= fin:
    medio = (inicio + fin) // 2
    if usuarios[medio].get_username() == username:
        print(f"Usuario encontrado:")
        {usuarios[medio].mostrar_datos()}")
        print("La búsqueda se realizó utilizando la técnica de
búsqueda binaria.")
        return usuarios[medio]
    elif usuarios[medio].get_username() < username:
        inicio = medio + 1
    else:
        fin = medio - 1
print("Usuario no encontrado.")
return None

def buscar_usuario():
    usuarios = cargar_datos()
    if not usuarios:
        print("No hay usuarios registrados.")
        return
    usuario_a_buscar = input("Ingrese el nombre de usuario a buscar: ")
    if not validar_usuario(usuario_a_buscar):
        print("Nombre de usuario no válido.")
        return
    print("Seleccione el tipo de búsqueda:")
    print("1. Búsqueda Secuencial")
    print("2. Búsqueda Binaria (requiere que los usuarios estén
ordenados por username)")
    opcion = input("Ingrese su elección (1 o 2): ")
    if opcion == "1":
        buscar_usuario_secuencial(usuario_a_buscar, usuarios)
    elif opcion == "2":
        usuarios.sort(key=lambda u: u.get_username())
        buscar_usuario_binaria(usuario_a_buscar, usuarios)
    else:
        print("Opción no válida.")

def mostrar_datos_accesos():
    try:
        with open('accesos.ispc', 'r') as archivo:
            datos = archivo.readlines()
            print("\n-----> DATOS DE ACCESOS <-----")
            for linea in datos:

```

```

        print(linea.strip())
    except FileNotFoundError:
        print("El archivo accesos.ispc no se encontró.")

def mostrar_intentos_fallidos():
    try:
        with open('logs.txt', 'r') as archivo:
            logs = archivo.readlines()
            print("\n-----> LOGS DE INTENTOS FALLIDOS <-----")
            for log in logs:
                print(log.strip())
    except FileNotFoundError:
        print("El archivo logs.txt no se encontró.")

def buscar_por_dni(dni_buscar):
    usuarios = cargar_datos()
    for usuario in usuarios:
        if usuario.get_dni() == dni_buscar:
            print("Usuario encontrado:", usuario.mostrar_datos())
            return
    print("No se encontró el usuario con el DNI ingresado.")

def buscar_por_email(email_buscar):
    usuarios = cargar_datos()
    for usuario in usuarios:
        if usuario.get_email() == email_buscar:
            print("Usuario encontrado:", usuario.mostrar_datos())
            return
    print("No se encontró el usuario con el email ingresado.")

```

gestionAcceso.py

```

from datetime import datetime
import json

class Acceso:
    def __init__(self, usuarioLogueado):
        self.id_acceso = None
        self.fecha_ingreso = None
        self.fecha_salida = None
        self.__usuarioLogueado = usuarioLogueado
        self.id_acceso = self.generar_id_acceso() # Generar un ID de acceso único

```

```

def generar_id_acceso(self):
    return datetime.now().timestamp() # Ejemplo de ID único

def login(self, email, contrasena):
    if self.__usuarioLogueado.get_email() == email and
self.__usuarioLogueado.get_contrasena() == contrasena:
        self.fecha_ingreso = datetime.now()
        self.registrar_acceso() # Llama al método para registrar
el acceso
        return True
    else:
        return False

def registrar_acceso(self):
    acceso_data = {
        "id_acceso": self.id_acceso,
        "email": self.__usuarioLogueado.get_email(),
        "fecha_ingreso": self.fecha_ingreso.isoformat()
    }
    with open('accesos.ispc', 'a') as file:
        file.write(json.dumps(acceso_data) + '\n')

def logout(self):
    self.fecha_salida = datetime.now()
    acceso_data = {
        "id_acceso": self.id_acceso,
        "email": self.__usuarioLogueado.get_email(),
        "fecha_salida": self.fecha_salida.isoformat()
    }
    with open('accesos.ispc', 'a') as file:
        file.write(json.dumps(acceso_data) + '\n')

def get_usuarioLogueado(self):
    return self.__usuarioLogueado

def __str__(self):
    return(f"Identificacion de Acceso : {self.id_acceso}, Usuario:
{self.get_usuarioLogueado()} , "
           f"Ingreso: {self.fecha_ingreso}, Salida:
{self.fecha_salida}")

def registrar_fallo(mensaje):
    with open('logs.txt', 'a') as log_file:

```

```
fecha_hora = datetime.now()
log_file.write(f"[{fecha_hora}] {mensaje}\n")
```

gestionUsuario.py

```
import pickle
from usuario import Usuario
import datetime

# Función para guardar usuario
def guardar_usuario(usuario):
    try:
        with open('usuarios.ispc', 'ab') as archivo:
            pickle.dump(usuario, archivo)
        print("Datos guardados exitosamente.")
    except Exception as e:
        print(f"Error al guardar los datos: {e}")

# Función para mostrar los datos guardados
def mostrar_datos():
    try:
        with open('usuarios.ispc', 'rb') as archivo:
            while True:
                try:
                    usuario = pickle.load(archivo)
                    if isinstance(usuario, Usuario):
                        print(f"DNI: {usuario.get_dni()}, Usuario: "
{usuario.get_username()}, Email: {usuario.get_email()}")
                    else:
                        print("Objeto no es una instancia de Usuario")
                except EOFError:
                    # Cuando se alcanza el final del archivo,
terminamos
                    break
    except FileNotFoundError:
        print("No se ha encontrado el archivo de datos.")
    except Exception as e:
        print(f"Error al cargar los datos: {e}")

# Función para cargar los datos desde un archivo binario
def cargar_datos():
    usuarios = []
    try:
```

```
with open('usuarios.ispc', 'rb') as archivo:
    while True:
        try:
            usuario = pickle.load(archivo)
            if isinstance(usuario, Usuario):
                usuarios.append(usuario)
            else:
                print("Objeto no es una instancia de Usuario")
        except EOFError:
            break
    except FileNotFoundError:
        print("El archivo 'usuarios.ispc' no existe.")
    except Exception as e:
        print(f"Error al cargar los datos: {e}")
    return usuarios

# Función para guardar todos los usuarios en un archivo binario
def guardar.todos._los_usuarios(usuarios):
    try:
        with open('usuarios.ispc', 'wb') as archivo:
            for usuario in usuarios:
                if isinstance(usuario, Usuario):
                    pickle.dump(usuario, archivo)
                else:
                    print("Objeto no es una instancia de Usuario")
        print("Todos los datos guardados exitosamente.")
    except Exception as e:
        print(f"Error al guardar los datos: {e}")

def verificar_usuarios():
    usuarios = []
    try:
        with open('usuarios.ispc', 'rb') as file:
            while True:
                try:
                    usuario = pickle.load(file)
                    if isinstance(usuario, Usuario):
                        usuarios.append(usuario)
                    else:
                        print("Objeto no es una instancia de Usuario")
                except EOFError:
                    break
    except:
```

```

except FileNotFoundError:
    print("El archivo usuarios.ispc no existe.")
return usuarios

def guardar_usuarios_ordenados(usuarios, filename='usuarios.ispc'):
    with open(filename, 'w') as file:
        for usuario in usuarios:
            file.write(f"{usuario['usuario']}\n")

```

Inicio_sesion.py

```

from gestionUsuario import verificar_usuarios
from gestionAcceso import Acceso

def iniciar_sesion():
    email = input("Ingrese su email: ")
    contrasena = input("Ingrese su contraseña: ")

    try:
        usuarios = verificar_usuarios()

        for usuario in usuarios:
            if usuario.get_email() == email:
                acceso = Acceso(usuario)
                if acceso.login(email, contrasena):
                    print("Inicio de sesión exitoso")
                    return True
                else:
                    print("Contraseña incorrecta")
                    Acceso.registrar_fallo(f"Intento fallido:
Contraseña incorrecta para el usuario {email}")
                    return False
            print("Usuario no encontrado")
            Acceso.registrar_fallo(f"Intento fallido: Usuario no encontrado
con el email {email}")
            return False
    except Exception as e:
        print(f"Error durante el inicio de sesión: {e}")
        Acceso.registrar_fallo(f"Error durante el inicio de sesión:
{e}")

```

```
    return False
```

Modulos_main.py

```
from funciones_main import agregar_usuario,
modificar_usuario,eliminar_usuario, buscar_usuario, mostrar_usuarios,
ordenar_usuarios_burbuja, mostrar_datos_accesos,
mostrar_intentos_fallidos, buscar_por_dni, buscar_por_email

def menu_usuarios():
    salir = False

    while not salir:
        print('-----> MENÚ USUARIOS Y ACCESOS <-----')
        print('1. CRUD')
        print('2. MOSTRAR ACCESOS')
        print('3. ORDENAMIENTO Y BÚSQUEDA')
        print('4. SALIR')

        opcion = input("Seleccione una opción: ")

        if opcion == '1':
            menu_crud()
        elif opcion == '2':
            menu_accesos()
        elif opcion == '3':
            ordenamiento_busqueda()
        elif opcion == '4':
            salir = True
        else:
            print("Opción no válida, intente nuevamente.")

def menu_crud():
    salir = False

    while not salir:
        print('-----> MENÚ CRUD <-----')
        print('1. AGREGAR USUARIO')
        print('2. MODIFICAR USUARIO')
        print('3. ELIMINAR USUARIO')
        print('4. SALIR')

        opcion = input("Seleccione una opción: ")
```

```
if opcion == '1':
    agregar_usuario()
    pass
elif opcion == '2':
    modificar_usuario()
    pass
elif opcion == '3':
    eliminar_usuario()
    pass
elif opcion == '4':
    salir = True
else:
    print("Opción no válida, intente nuevamente.")

def menu_accesos():
    salir = False

    while not salir:
        print('-----> MENÚ ACCESOS <-----')
        print('1. MOSTRAR ACCESOS')
        print('2. MOSTRAR INTENTOS FALLIDOS')
        print('3. SALIR')

        opcion = input("Seleccione una opción: ")

        if opcion == '1':
            mostrar_datos_accesos()
        elif opcion == '2':
            mostrar_intentos_fallidos()
        elif opcion == '3':
            salir = True
            print("Volviendo al menú anterior...")
        else:
            print("Opción no válida, intente nuevamente.")

def ordenamiento_busqueda():
    salir = False

    while not salir:
        print('1. ORDENAR USUARIOS')
```

```
print('2. BÚSQUEDA POR DNI')
print('3. SALIR')

opcion = input("Seleccione una opción: ")

if opcion == '1':
    ordenar_usuarios_burbuja()
elif opcion == '2':
    buscar_mostrar()
elif opcion == '3':
    salir = True
else:
    print("Opción no válida, intente nuevamente.")

def buscar_mostrar():
    salir = False

    while not salir:
        print('1. BÚSQUEDA POR DNI')
        print('2. BÚSQUEDA POR USERNAME')
        print('3. BÚSQUEDA POR EMAIL')
        print('4. MOSTRAR USUARIOS')
        print('5. SALIR')

        opcion = input("Seleccione una opción: ")

        if opcion == '1':
            dni_buscar = input("Ingrese el DNI a buscar: ")
            buscar_por_dni(dni_buscar)
        elif opcion == '2':
            buscar_usuario()
        elif opcion == '3':
            email_buscar = input("Ingrese el email a buscar: ")
            buscar_por_email(email_buscar)
        elif opcion == '4':
            mostrar_usuarios()
        elif opcion == '5':
            salir = True
        else:
            print("Opción no válida, intente nuevamente.")
```

```
registros_pluviales.py
import os
import random
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

def generar_registros_aleatorios():
    dias_por_mes = [31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31]
    registros = []

    for dias in dias_por_mes:
        mes_registros = [random.randint(0, 500) for _ in range(dias)]
        mes_registros += [0] * (31 - dias)
        registros.append(mes_registros)

    return registros

def cargar_registros_pluviales():
    año = input("Ingrese el año para cargar los registros pluviales: ")
    carpeta = 'registros_pluviales'

    archivo_csv = os.path.join(carpeta, f'registroPluvial{año}.csv')

    if os.path.exists(archivo_csv):
        print(f"Archivo encontrado: {archivo_csv}")
        datos = pd.read_csv(archivo_csv)
    else:
        print(f"No se encontró el archivo. Generando registros aleatorios para el año {año}.")
        registros = generar_registros_aleatorios()
        guardar_en_csv(registros, año, carpeta)
        datos = pd.DataFrame({f'Mes {i+1}': registros[i] for i in range(len(registros))})

    return datos, año

def guardar_en_csv(registros, año, carpeta):
    df = pd.DataFrame({f'Mes {i+1}': registros[i] for i in range(len(registros))})
    archivo_csv = os.path.join(carpeta, f'registroPluvial{año}.csv')
    df.to_csv(archivo_csv, index=False)
```

```

print(f"Registros pluviales guardados en {archivo_csv}.")
```

```

def generar_grafico_barras(registros, mes, año):
    dias = list(range(1, len(registros) + 1))
    lluvias = registros

    plt.figure(figsize=(10, 6))
    plt.bar(dias, lluvias, color='skyblue')
    plt.title(f'Lluvias del mes {mes} en el año {año}')
    plt.xlabel('Días')
    plt.ylabel('Lluvias (mm)')
    plt.xticks(rotation=45)
    plt.tight_layout()
    plt.show()
```

```

def generar_grafico dispersion(registros, mes, año):
    dias = list(range(1, len(registros) + 1))
    lluvias = registros

    plt.figure(figsize=(10, 6))
    plt.scatter(dias, lluvias, color='orange')
    plt.title(f'Dispersión de lluvias del mes {mes} en el año {año}')
    plt.xlabel('Días')
    plt.ylabel('Lluvias (mm)')
    plt.xticks(np.arange(1, len(dias) + 1, 1), rotation=45)
    plt.grid()
    plt.tight_layout()
    plt.show()
```

```

def generar_grafico_circular(datos, año):
    lluvias_totales_por_mes = [datos[f'Mes {i+1}'].sum() for i in
range(12)]
    meses = ['Enero', 'Febrero', 'Marzo', 'Abril', 'Mayo', 'Junio',
'Julio', 'Agosto', 'Septiembre', 'Octubre', 'Noviembre',
'Diciembre']

    plt.figure(figsize=(10, 10))
    plt.pie(lluvias_totales_por_mes, labels=meses, autopct='%.1f%%',
startangle=140,
```

```

colors=plt.cm.Paired(np.arange(len(lluvias_totales_por_mes))))
    plt.title(f'Porcentaje de lluvias por mes en el año {año}')
    plt.axis('equal')
```

```
plt.tight_layout()
plt.show()

def menu_registros_pluviales():
    datos, año = cargar_registros_pluviales()
    salir = False
    while not salir:
        print("\n---> MENÚ RESGISTROS PLUVIALES <---")
        print("1. Mostrar registros de lluvias por mes")
        print("2. Gráfico de barras de lluvias por mes")
        print("3. Gráfico de dispersión de lluvias por mes")
        print("4. Gráfico circular de lluvias anuales")
        print("5. Salir")

        opcion = input("Ingrese la opción que desea ejecutar: ")

        if opcion == "1":
            mes = int(input("Ingrese el mes (1-12) para mostrar los
registros: "))
            if 1 <= mes <= 12:
                registros_mes = datos[f'Mes {mes}'].dropna().tolist()
                print(f"Registros pluviales para el mes {mes} del año
{año}:")
                print(registros_mes)
            else:
                print("Mes inválido. Debe estar entre 1 y 12.")

        elif opcion == "2":
            mes = int(input("Ingrese el mes (1-12) para generar el
gráfico de barras: "))
            if 1 <= mes <= 12:
                registros_mes = datos[f'Mes {mes}'].dropna().tolist()
                generar_grafico_barras(registros_mes, mes, año)
            else:
                print("Mes inválido. Debe estar entre 1 y 12.")

        elif opcion == "3":
            mes = int(input("Ingrese el mes (1-12) para generar el
gráfico de dispersión: "))
            if 1 <= mes <= 12:
                registros_mes = datos[f'Mes {mes}'].dropna().tolist()
                generar_grafico_dispersion(registros_mes, mes, año)
            else:
```

```
        print("Mes inválido. Debe estar entre 1 y 12.")

    elif opcion == "4":
        generar_grafico_circular(datos, año)

    elif opcion == "5":
        print("Saliendo de registros pluviales.")
        salir = True

else:
    print("Opción no válida. Intente de nuevo.")
```

usuario.py

```
class Usuario:
    def __init__(self, dni, username, email, contrasena):
        self.__dni = dni
        self.__username = username
        self.__email = email
        self.__contrasena = contrasena

    def get_dni(self):
        return self.__dni

    def get_username(self):
        return self.__username

    def set_username(self, username):
        self.__username = username

    def get_email(self):
        return self.__email

    def set_email(self, email):
        self.__email = email

    def set_contrasena(self, contrasena):
        self.__contrasena = contrasena

    def get_contrasena(self):
        return self.__contrasena
```

```
def mostrar_datos(self):
    return f"DNI: {self.__dni}, Usuario: {self.__username}, Email: {self.__email}"
```

validaciones.py

```
import re

def validar_dni(dni):
    if dni.isdigit() and (len(dni) == 8 or len(dni) == 7):
        return True
    return False

def validar_usuario(usuario):
    usuario = usuario.lower()
    if usuario.isalpha():
        return usuario
    return False

def validar_email(correo):
    correo = correo.lower()
    if '@' in correo and '.com' in correo:
        return correo
    return False

def validar_contrasena(contrasena):
    if len(contrasena) < 8:
        return False
    if not re.search(r"[a-z]", contrasena):
        return False
    if not re.search(r"[0-9]", contrasena):
        return False
    return True
```

A continuación se muestran capturas de pantallas de todas las funcionalidades dentro del menú principal y los sub-menús:

Menu Principal

```
C:\Users\Bright!Tax\Documents\Repositorio ISPC\InnovacionFinal> python proyecto_final\menu.py
-----> MENÚ PRINCIPAL <-----
1. USUARIOS Y ACCESOS
2. INICIAR SESIÓN
3. ANÁLISIS DE DATOS
4. SALIR
Seleccione una opción: 1
-----> MENÚ USUARIOS Y ACCESOS <-----
1. CRUD
2. MOSTRAR ACCESOS
3. ORDENAMIENTO Y BÚSQUEDA
4. SALIR
Seleccione una opción: 1
-----> MENÚ CRUD <-----
1. AGREGAR USUARIO
2. MODIFICAR USUARIO
3. ELIMINAR USUARIO
4. SALIR
Seleccione una opción: 1
```

Menu Principal>Usuarios y Accesos > CRUD >Agregar usuario

```
C:\Users\Bright!Tax\Documents\Repositorio ISPC\InnovacionFinal> python proyecto_final\menu.py
-----> MENÚ PRINCIPAL <-----
1. USUARIOS Y ACCESOS
2. INICIAR SESIÓN
3. ANÁLISIS DE DATOS
4. SALIR
Seleccione una opción: 1
-----> MENÚ USUARIOS Y ACCESOS <-----
1. CRUD
2. MOSTRAR ACCESOS
3. ORDENAMIENTO Y BÚSQUEDA
4. SALIR
Seleccione una opción: 1
-----> MENÚ CRUD <-----
1. AGREGAR USUARIO
2. MODIFICAR USUARIO
3. ELIMINAR USUARIO
4. SALIR
Seleccione una opción: 1
Ingrese el DNI del usuario: 25411481
Ingrese el nombre del usuario: Consuelo
Ingrese el email: Consuelo@gmail.com
Ingrese la contraseña, mínimo 8 caracteres, minúsculas y números: Consuelo123!
Datos guardados exitosamente.
Usuario agregado exitosamente.
-----> MENÚ CRUD <-----
1. AGREGAR USUARIO
2. MODIFICAR USUARIO
3. ELIMINAR USUARIO
4. SALIR
Seleccione una opción: 1
```

Menu Principal>Usuarios y Accesos > CRUD >Modificar usuario

```
-----> MENÚ CRUD <-----  
1. AGREGAR USUARIO  
2. MODIFICAR USUARIO  
3. ELIMINAR USUARIO  
4. SALIR  
Seleccione una opción: 2  
Ingrese el DNI del usuario a modificar: 25411481  
Ingrese el nuevo nombre de usuario: Consue  
Ingrese el nuevo email: consuelo@gmail.com  
Ingrese la nueva contraseña: Consuelo123!  
Todos los datos guardados exitosamente.  
Usuario modificado exitosamente.  
-----> MENÚ CRUD <-----  
1. AGREGAR USUARIO  
2. MODIFICAR USUARIO  
3. ELIMINAR USUARIO  
4. SALIR  
Seleccione una opción: |
```

Menu Principal>Usuarios y Accesos > CRUD >Eliminar usuario

```
-----> MENÚ CRUD <-----  
1. AGREGAR USUARIO  
2. MODIFICAR USUARIO  
3. ELIMINAR USUARIO  
4. SALIR  
Seleccione una opción: 3  
Ingrese el DNI del usuario a eliminar: 11111111  
Todos los datos guardados exitosamente.  
Usuario eliminado exitosamente.  
-----> MENÚ CRUD <-----  
1. AGREGAR USUARIO  
2. MODIFICAR USUARIO  
3. ELIMINAR USUARIO  
4. SALIR  
Seleccione una opción: |
```

Menu Principal>Usuarios y Accesos > CRUD >Salir

```
-----> MENÚ CRUD <-----  
1. AGREGAR USUARIO  
2. MODIFICAR USUARIO  
3. ELIMINAR USUARIO  
4. SALIR  
Seleccione una opción: 4  
-----> MENÚ USUARIOS Y ACCESOS <-----  
1. CRUD  
2. MOSTRAR ACCESOS  
3. ORDENAMIENTO Y BÚSQUEDA  
4. SALIR  
Seleccione una opción: |
```

Menu Principal>Usuarios y Accesos > Mostrar Accesos

```
3. ORDENAMIENTO Y BÚSQUEDA
4. SALIR
Seleccione una opción: 2
-----> MENÚ ACCESOS <-----
1. MOSTRAR ACCESOS
2. MOSTRAR INTENTOS FALLIDOS
3. SALIR
Seleccione una opción: 1

-----> DATOS DE ACCESOS <-----
[{"id_acceso": 1729694061.945483, "email": "susi@gmail.com", "fecha_ingreso": "2024-10-23T11:34:21.945483"}, {"id_acceso": 1729694166.248103, "email": "susi@gmail.com", "fecha_ingreso": "2024-10-23T11:36:06.248103"}, {"id_acceso": 1730217786.066116, "email": "hebe@gmail.com", "fecha_ingreso": "2024-10-29T13:03:06.066116"}, {"id_acceso": 1730217904.291437, "email": "hebe@gmail.com", "fecha_ingreso": "2024-10-29T13:05:04.291437"}, {"id_acceso": 1730218005.405221, "email": "hebe@gmail.com", "fecha_ingreso": "2024-10-29T13:06:45.405221"}, {"id_acceso": 1730218351.134099, "email": "hebe@gmail.com", "fecha_ingreso": "2024-10-29T13:12:31.134099"}, {"id_acceso": 1730218485.162378, "email": "hebe@gmail.com", "fecha_ingreso": "2024-10-29T13:14:45.162378"}, {"id_acceso": 1730238473.647488, "email": "usuariodeprueba@gmail.com", "fecha_ingreso": "2024-10-29T18:47:53.650152"}, {"id_acceso": 1730238708.795384, "email": "usuariodeprueba@gmail.com", "fecha_ingreso": "2024-10-29T18:51:48.795384"}, {"id_acceso": 1730239177.329612, "email": "usuariodeprueba@gmail.com", "fecha_ingreso": "2024-10-29T18:59:37.330611"}, {"id_acceso": 1730239294.580859, "email": "usuariodeprueba@gmail.com", "fecha_ingreso": "2024-10-29T19:01:34.580859"}, {"id_acceso": 1730239486.376263, "email": "usuariodeprueba@gmail.com", "fecha_ingreso": "2024-10-29T19:04:46.378664"}, {"id_acceso": 1730239583.67195, "email": "usuariodeprueba@gmail.com", "fecha_ingreso": "2024-10-29T19:06:23.671950"}, {"id_acceso": 1730286519.643073, "email": "usuarioprueba2@gmail.com", "fecha_ingreso": "2024-10-30T08:08:39.646033"}, {"id_acceso": 1730290523.768663, "email": "usuarioprueba2@gmail.com", "fecha_ingreso": "2024-10-30T09:15:23.770661"}, {"id_acceso": 1730290906.015662, "email": "usuarioprueba2@gmail.com", "fecha_ingreso": "2024-10-30T09:21:46.053019"}, {"id_acceso": 1730291049.13876, "email": "usuarioprueba2@gmail.com", "fecha_ingreso": "2024-10-30T09:24:09.139759"}, {"id_acceso": 1730298674.145521, "email": "vicki@gmail.com", "fecha_ingreso": "2024-10-30T11:31:14.145521"}, {"id_acceso": 1730298732.768325, "email": "vicki@gmail.com", "fecha_ingreso": "2024-10-30T11:32:12.768325"}, {"id_acceso": 1730298945.93386, "email": "vicki@gmail.com", "fecha_ingreso": "2024-10-30T11:35:45.933860"}, {"id_acceso": 1730298968.981803, "email": "vicki@gmail.com", "fecha_ingreso": "2024-10-30T11:36:08.981803"}, {"id_acceso": 1730299125.526489, "email": "vicki@gmail.com", "fecha_ingreso": "2024-10-30T11:38:45.526489"}, {"id_acceso": 1730302866.544984, "email": "vicki@gmail.com", "fecha_ingreso": "2024-10-30T12:41:06.544984"}, {"id_acceso": 1730303187.965683, "email": "vicki@gmail.com", "fecha_ingreso": "2024-10-30T12:46:27.965683"}, {"id_acceso": 1730303257.272114, "email": "vicki@gmail.com", "fecha_ingreso": "2024-10-30T12:47:37.272114"}, {"id_acceso": 1730303561.71285, "email": "vicki@gmail.com", "fecha_ingreso": "2024-10-30T12:52:41.712850"}, {"id_acceso": 1730303697.015975, "email": "vicki@gmail.com", "fecha_ingreso": "2024-10-30T12:54:57.015975"}, {"id_acceso": 1730306138.001764, "email": "vicki@gmail.com", "fecha_ingreso": "2024-10-30T13:35:38.001764"}, {"id_acceso": 1730306383.430344, "email": "vicki@gmail.com", "fecha_ingreso": "2024-10-30T13:39:43.430344"}, {"id_acceso": 1730306450.011243, "email": "vicki@gmail.com", "fecha_ingreso": "2024-10-30T13:40:50.011243"}, {"id_acceso": 1730306518.962243, "email": "vicki@gmail.com", "fecha_ingreso": "2024-10-30T13:41:58.962243"}, {"id_acceso": 1730306708.582927, "email": "vicki@gmail.com", "fecha_ingreso": "2024-10-30T13:45:08.582927"}, {"id_acceso": 1730306911.96582, "email": "vicki@gmail.com", "fecha_ingreso": "2024-10-30T13:48:31.965820"}, {"id_acceso": 1730307056.672938, "email": "vicki@gmail.com", "fecha_ingreso": "2024-10-30T13:50:56.672938"}, {"id_acceso": 17303090591.041518, "email": "vicki@gmail.com", "fecha_ingreso": "2024-10-31T13:03:11.041518"}, {"id_acceso": 17303090622.632093, "email": "vicki@gmail.com", "fecha_ingreso": "2024-10-31T13:03:42.632093"}, {"id_acceso": 17303090659.13735, "email": "vicki@gmail.com", "fecha_ingreso": "2024-10-31T13:04:19.137350"}, {"id_acceso": 1730483746.526849, "email": "victoria@gmail.com", "fecha_ingreso": "2024-11-01T14:55:46.526849"}, {"id_acceso": 1730498111.408219, "email": "hebe@gmail.com", "fecha_ingreso": "2024-11-01T18:55:11.408219"}]
-----> MENÚ ACCESOS <-----
1. MOSTRAR ACCESOS
```

Menu Principal>Usuarios y Accesos > Mostrar Accesos Fallidos

```
-----> MENÚ ACCESOS <-----  
1. MOSTRAR ACCESOS  
2. MOSTRAR INTENTOS FALLIDOS  
3. SALIR  
Seleccione una opción: 2  
  
-----> LOGS DE INTENTOS FALLIDOS <-----  
[2024-10-23 11:35:23.535337] Intento fallido: Contraseña incorrecta para el usuario susi@gmail.com  
[2024-10-23 11:35:43.498150] Intento fallido: Usuario no encontrado con el email susi@gmail.com  
[2024-10-29 12:54:03.423234] Intento fallido: Contraseña incorrecta para el usuario vpicco@gmail.com  
[2024-10-29 12:54:28.518630] Intento fallido: Contraseña incorrecta para el usuario vpicco@gmail.com  
[2024-10-29 12:55:26.494432] Intento fallido: Contraseña incorrecta para el usuario juan@gmail.com  
[2024-10-29 12:57:01.484427] Intento fallido: Contraseña incorrecta para el usuario juan@gmail.com  
[2024-10-29 13:00:20.889727] Intento fallido: Contraseña incorrecta para el usuario vpicco@gmail.com  
[2024-10-29 13:00:32.778603] Intento fallido: Contraseña incorrecta para el usuario juan@gmail.com  
[2024-10-29 13:01:37.553072] Intento fallido: Contraseña incorrecta para el usuario vpicco@gmail.com  
[2024-10-29 13:01:48.657477] Intento fallido: Contraseña incorrecta para el usuario juan@gmail.com  
[2024-10-29 17:29:03.960548] Intento fallido: Usuario no encontrado con el email hebecpereyra@gmail.com  
[2024-10-29 17:30:21.297027] Intento fallido: Contraseña incorrecta para el usuario hebecpereyra@gmail.com  
[2024-10-29 17:30:45.075892] Intento fallido: Contraseña incorrecta para el usuario hebecpereyra@gmail.com  
[2024-10-29 17:32:29.723577] Intento fallido: Usuario no encontrado con el email p_hebe@hotmail.com  
[2024-10-29 17:35:43.280678] Intento fallido: Contraseña incorrecta para el usuario prueba@gmail.com  
[2024-10-29 17:47:04.418850] Intento fallido: Contraseña incorrecta para el usuario hebecpereyra@gmail.com  
[2024-10-29 18:42:00.847724] Intento fallido: Usuario no encontrado con el email usuariodos@gmail.com  
[2024-10-29 19:06:00.140249] Intento fallido: Contraseña incorrecta para el usuario usuariodeprueba@gmail.com  
[2024-10-30 08:06:42.620156] Intento fallido: Usuario no encontrado con el email usuarioprueba@gmail.com  
[2024-10-30 09:20:20.551058] Intento fallido: Usuario no encontrado con el email usuarioprueba2@gmail.com  
[2024-10-30 09:21:11.178472] Intento fallido: Contraseña incorrecta para el usuario usuarioprueba2@gmail.com  
[2024-10-30 11:29:34.547600] Intento fallido: Usuario no encontrado con el email vicki@gmail.com  
[2024-10-31 13:02:44.544498] Intento fallido: Usuario no encontrado con el email vicki@gmail.com  
[2024-11-01 14:55:08.442401] Intento fallido: Contraseña incorrecta para el usuario victoria@gmail.com  
[2024-11-01 14:55:28.885640] Intento fallido: Usuario no encontrado con el email vicki@gmail.com  
[2024-11-01 18:52:54.186375] Intento fallido: Contraseña incorrecta para el usuario hebe@gmail.com  
-----> MENÚ ACCESOS <-----  
1. MOSTRAR ACCESOS  
2. MOSTRAR INTENTOS FALLIDOS  
3. SALIR  
Seleccione una opción: |
```

Menu Principal>Usuarios y Accesos > Salir

```
-----> MENÚ ACCESOS <-----  
1. MOSTRAR ACCESOS  
2. MOSTRAR INTENTOS FALLIDOS  
3. SALIR  
Seleccione una opción: 3  
Volviendo al menú anterior...  
-----> MENÚ USUARIOS Y ACCESOS <-----  
1. CRUD  
2. MOSTRAR ACCESOS  
3. ORDENAMIENTO Y BÚSQUEDA  
4. SALIR  
Seleccione una opción: |
```

Menu Principal>Ordenamiento y Busqueda

```
1. MOSTRAR ACCESOS
2. MOSTRAR INTENTOS FALLIDOS
3. SALIR
Seleccione una opción: 3
Volviendo al menú anterior...
-----> MENÚ USUARIOS Y ACCESOS <-----
1. CRUD
2. MOSTRAR ACCESOS
3. ORDENAMIENTO Y BÚSQUEDA
4. SALIR
Seleccione una opción: 3
1. ORDENAR USUARIOS
2. BUSCAR Y MOSTRAR USUARIOS
3. SALIR
Seleccione una opción: |
```

Menu Principal>Ordenamiento y Busqueda> Ordenar Usuarios

```
-----> MENU ACCESOS <-----
1. MOSTRAR ACCESOS
2. MOSTRAR INTENTOS FALLIDOS
3. SALIR
Seleccione una opción: 3
Volviendo al menú anterior...
-----> MENÚ USUARIOS Y ACCESOS <-----
1. CRUD
2. MOSTRAR ACCESOS
3. ORDENAMIENTO Y BÚSQUEDA
4. SALIR
Seleccione una opción: 3
1. ORDENAR USUARIOS
2. BUSCAR Y MOSTRAR USUARIOS
3. SALIR
Seleccione una opción: 1
Todos los datos guardados exitosamente.
1. ORDENAR USUARIOS
2. BUSCAR Y MOSTRAR USUARIOS
3. SALIR
Seleccione una opción: |
```

Menu Principal>Ordenamiento y Busqueda> Buscar y Mostrar Usuarios

```
-----> MENÚ ACCESOS <-----  
1. MOSTRAR ACCESOS  
2. MOSTRAR INTENTOS FALLIDOS  
3. SALIR  
Seleccione una opción: 3  
Volviendo al menú anterior...  
-----> MENÚ USUARIOS Y ACCESOS <-----  
1. CRUD  
2. MOSTRAR ACCESOS  
3. ORDENAMIENTO Y BÚSQUEDA  
4. SALIR  
Seleccione una opción: 3  
1. ORDENAR USUARIOS  
2. BUSCAR Y MOSTRAR USUARIOS  
3. SALIR  
Seleccione una opción: 1  
Todos los datos guardados exitosamente.  
1. ORDENAR USUARIOS  
2. BUSCAR Y MOSTRAR USUARIOS  
3. SALIR  
Seleccione una opción: 2  
1. BÚSQUEDA POR DNI  
2. BÚSQUEDA POR USERNAME  
3. BÚSQUEDA POR EMAIL  
4. MOSTRAR USUARIOS  
5. SALIR  
Seleccione una opción: []
```

Menu Principal>Ordenamiento y Busqueda> Buscar y Mostrar Usuarios> Busqueda por DNI

```
Seleccione una opción: 2  
1. BÚSQUEDA POR DNI  
2. BÚSQUEDA POR USERNAME  
3. BÚSQUEDA POR EMAIL  
4. MOSTRAR USUARIOS  
5. SALIR  
Seleccione una opción: 1  
Ingrese el DNI a buscar: 34980147  
Usuario encontrado: DNI: 34980147, Usuario: jesica, Email: jesica@gmail.com  
1. BÚSQUEDA POR DNI  
2. BÚSQUEDA POR USERNAME  
3. BÚSQUEDA POR EMAIL  
4. MOSTRAR USUARIOS  
5. SALIR  
Seleccione una opción: []
```

Menu Principal>Ordenamiento y Busqueda> Buscar y Mostrar Usuarios> Busqueda por Username>Busqueda Secuencial

```
1. BÚSQUEDA POR DNI
2. BÚSQUEDA POR USERNAME
3. BÚSQUEDA POR EMAIL
4. MOSTRAR USUARIOS
5. SALIR
Seleccione una opción: 2
Ingrese el nombre de usuario a buscar: consuelo
Seleccione el tipo de búsqueda:
1. Búsqueda Secuencial
2. Búsqueda Binaria (requiere que los usuarios estén ordenados por username)
Ingrese su elección (1 o 2): 1
Usuario no encontrado.
1. BÚSQUEDA POR DNI
2. BÚSQUEDA POR USERNAME
3. BÚSQUEDA POR EMAIL
4. MOSTRAR USUARIOS
5. SALIR
Seleccione una opción: 
```

Menu Principal>Ordenamiento y Busqueda> Buscar y Mostrar Usuarios> Busqueda por Username>Busqueda Binaria

```
1. BÚSQUEDA POR DNI
2. BÚSQUEDA POR USERNAME
3. BÚSQUEDA POR EMAIL
4. MOSTRAR USUARIOS
5. SALIR
Seleccione una opción: 2
Ingrese el nombre de usuario a buscar: hebe
Seleccione el tipo de búsqueda:
1. Búsqueda Secuencial
2. Búsqueda Binaria (requiere que los usuarios estén ordenados por username)
Ingrese su elección (1 o 2): 2
Usuario encontrado: DNI: 33117717, Usuario: hebe, Email: hebe@gmail.com
La búsqueda se realizó utilizando la técnica de búsqueda binaria.
1. BÚSQUEDA POR DNI
2. BÚSQUEDA POR USERNAME
3. BÚSQUEDA POR EMAIL
4. MOSTRAR USUARIOS
5. SALIR
Seleccione una opción: 
```

Menu Principal>Ordenamiento y Busqueda> Buscar y Mostrar Usuarios> Busqueda por email

- 1. BÚSQUEDA POR DNI
- 2. BÚSQUEDA POR USERNAME
- 3. BÚSQUEDA POR EMAIL
- 4. MOSTRAR USUARIOS
- 5. SALIR

Seleccione una opción: 3

Ingrese el email a buscar: hebe@gmail.com

Usuario encontrado: DNI: 33117717, Usuario: hebe, Email: hebe@gmail.com

- 1. BÚSQUEDA POR DNI
- 2. BÚSQUEDA POR USERNAME
- 3. BÚSQUEDA POR EMAIL
- 4. MOSTRAR USUARIOS
- 5. SALIR

Seleccione una opción: 1

Menu Principal>Ordenamiento y Busqueda> Buscar y Mostrar Usuarios> Mostrar Usuarios

- 1. BÚSQUEDA POR DNI
- 2. BÚSQUEDA POR USERNAME
- 3. BÚSQUEDA POR EMAIL
- 4. MOSTRAR USUARIOS
- 5. SALIR

Seleccione una opción: 4

DNI: 25411481, Usuario: Consue, Email: consuelo@gmail.com

DNI: 33117717, Usuario: hebe, Email: hebe@gmail.com

DNI: 34980147, Usuario: jesica, Email: jesica@gmail.com

DNI: 21718397, Usuario: mariela, Email: mariela@gmail.com

DNI: 11111111, Usuario: prueba, Email: prueba@gmail.com

DNI: 25342899, Usuario: ulises, Email: ulises@gmail.com

DNI: 39546009, Usuario: victoria, Email: victoria@gmail.com

- 1. BÚSQUEDA POR DNI
- 2. BÚSQUEDA POR USERNAME
- 3. BÚSQUEDA POR EMAIL
- 4. MOSTRAR USUARIOS
- 5. SALIR

Seleccione una opción: 1

Menu Principal>Menu Usuarios y Accesos >Ordenamiento y Busqueda> Buscar y Mostrar Usuarios> Salir

Menu Principal>Menu Usuarios y Accesos >Ordenamiento y Busqueda> Salir

Menu Principal>Menu Usuarios y Accesos > Salir

```
1. BÚSQUEDA POR DNI
2. BÚSQUEDA POR USERNAME
3. BÚSQUEDA POR EMAIL
4. MOSTRAR USUARIOS
5. SALIR
Seleccione una opción: 5
1. ORDENAR USUARIOS
2. BUSCAR Y MOSTRAR USUARIOS
3. SALIR
Seleccione una opción: 3
-----> MENÚ USUARIOS Y ACCESOS <-----
1. CRUD
2. MOSTRAR ACCESOS
3. ORDENAMIENTO Y BÚSQUEDA
4. SALIR
Seleccione una opción: 4
-----> MENÚ PRINCIPAL <-----
1. USUARIOS Y ACCESOS
2. INICIAR SESIÓN
3. ANÁLISIS DE DATOS
4. SALIR
Seleccione una opción: |
```

Menu Principal>Iniciar Sesion

```
-----> MENÚ PRINCIPAL <-----
1. USUARIOS Y ACCESOS
2. INICIAR SESIÓN
3. ANÁLISIS DE DATOS
4. SALIR
Seleccione una opción: 2
Ingrese su email: hebe@gmail.com
Ingrese su contraseña: hebe123!
Inicio de sesión exitoso
Conexión a la base de datos exitosa.

-----> MENÚ DE OPCIONES <-----
----->

1 - REGISTRO VENTA
2 - CONSULTAR STOCK
3 - ACTUALIZAR STOCK
4 - ALTA
5 - BAJA
6 - MODIFICAR
7 - SALIR
Ingrese una opción: |
```

Menu Principal>Iniciar Sesión>Registro Venta

-----> MENÚ PRINCIPAL <-----

1. USUARIOS Y ACCESOS
2. INICIAR SESIÓN
3. ANÁLISIS DE DATOS
4. SALIR

Seleccione una opción: 2

Ingrese su email: hebe@gmail.com

Ingrese su contraseña: hebe123!

Inicio de sesión exitoso

Conexión a la base de datos exitosa.

-----> MENÚ DE OPCIONES <---

1 - REGISTRO VENTA

2 - CONSULTAR STOCK

3 - ACTUALIZAR STOCK

4 - ALTA

5 - BAJA

6 - MODIFICAR

7 - SALIR

Ingrese una opción: 1

PRODUCTOS DISPONIBLES:

ID - DESCRIP - CANT - TAMAÑO - UM...

Conexión a la base de datos exitosa.

PRODUCTOS FRACCIONADOS DISPONIBLES:

- (1, 'HARINA INTEGRAL', 2, 250, 'gr', 10, 3, 1)
- (2, 'HARINA INTEGRAL', 3, 500, 'gr', 7, 3, 1)
- (3, 'HARINA INTEGRAL', 5, 1000, 'gr', 5, 2, 1)
- (4, 'LENTEJAS', 5, 1000, 'gr', 25, 5, 2)
- (5, 'ALMOHADITAS FRUTILLA', 2, 250, 'gr', 15, 5, 3)
- (6, 'ALMOHADITAS FRUTILLA', 4, 500, 'gr', 10, 2, 3)
- (7, 'NUECES', 4, 250, 'gr', 8, 2, 4)
- (8, 'NUECES', 4, 500, 'gr', 10, 5, 4)
- (9, 'QUINOA', 2, 500, 'gr', 12, 4, 5)
- (10, 'QUINOA', 3, 1000, 'gr', 8, 2, 5)

Ingresá la fecha(YYYY-MM-DD): █

Menu Principal>Iniciar Sesion>Registro Venta

---> MENÚ DE OPCIONES <---

-
- 1 - REGISTRO VENTA
 - 2 - CONSULTAR STOCK
 - 3 - ACTUALIZAR STOCK
 - 4 - ALTA
 - 5 - BAJA
 - 6 - MODIFICAR
 - 7 - SALIR

Ingrese una opción: 1

PRODUCTOS DISPONIBLES:

ID - DESCRIP - CANT - TAMAÑO - UM...

Conexión a la base de datos exitosa.

PRODUCTOS FRACCIONADOS DISPONIBLES:

- (1, 'HARINA INTEGRAL', 2, 250, 'gr', 10, 3, 1)
- (2, 'HARINA INTEGRAL', 3, 500, 'gr', 7, 3, 1)
- (3, 'HARINA INTEGRAL', 5, 1000, 'gr', 5, 2, 1)
- (4, 'LENTEJAS', 5, 1000, 'gr', 25, 5, 2)
- (5, 'ALMOHADITAS FRUTILLA', 2, 250, 'gr', 15, 5, 3)
- (6, 'ALMOHADITAS FRUTILLA', 4, 500, 'gr', 10, 2, 3)
- (7, 'NUECES', 4, 250, 'gr', 8, 2, 4)
- (8, 'NUECES', 4, 500, 'gr', 10, 5, 4)
- (9, 'QUINOA', 2, 500, 'gr', 12, 4, 5)
- (10, 'QUINOA', 3, 1000, 'gr', 8, 2, 5)

Ingresa la fecha(YYYY-MM-DD): 2024-11-01

Ingresa la cantidad: 1

Ingresa el ID del producto vendido: 2

Conexión a la base de datos exitosa.

SU VENTA SE REGISTRÓ EXITOSAMENTE.

Menu Principal>Iniciar Sesion>Consultar Stock> Total

---> MENÚ DE OPCIONES <---

- 1 - REGISTRO VENTA
- 2 - CONSULTAR STOCK
- 3 - ACTUALIZAR STOCK
- 4 - ALTA
- 5 - BAJA
- 6 - MODIFICAR
- 7 - SALIR

Ingrese una opción: 2

Ingrese la opción de STOCK que desea CONSULTAR:

- 1 - TOTAL
- 2 - GRANEL
- 3 - FRACCIONADO
- 4 - SALIR

Ingrese una opción: 1

Conexión a la base de datos exitosa.

PRODUCTOS FRACCIONADOS DISPONIBLES:

- (1, 'HARINA INTEGRAL', 2, 250, 'gr', 10, 3, 1)
- (2, 'HARINA INTEGRAL', 2, 500, 'gr', 7, 3, 1)
- (3, 'HARINA INTEGRAL', 5, 1000, 'gr', 5, 2, 1)
- (4, 'LENTEJAS', 5, 1000, 'gr', 25, 5, 2)
- (5, 'ALMOHADITAS FRUTILLA', 2, 250, 'gr', 15, 5, 3)
- (6, 'ALMOHADITAS FRUTILLA', 4, 500, 'gr', 10, 2, 3)
- (7, 'NUECES', 4, 250, 'gr', 8, 2, 4)
- (8, 'NUECES', 4, 500, 'gr', 10, 5, 4)
- (9, 'QUINOA', 2, 500, 'gr', 12, 4, 5)
- (10, 'QUINOA', 3, 1000, 'gr', 8, 2, 5)

Conexión a la base de datos exitosa.

Ingresa el nombre del producto: 8

DEPÓSITO FRACCIONADO-----DEPÓSITO GRANEL

Desc.-Cant.-Tamaño-UM-// -Desc.-Cant.-Tamaño-UM

No hay productos en la base de datos.

Consulta STOCK TOTAL realizada con éxito.

Ingrese la opción de STOCK que desea CONSULTAR:

- 1 - TOTAL
- 2 - GRANEL
- 3 - FRACCIONADO
- 4 - SALIR

Ingrese una opción: 1

Menu Principal>Iniciar Sesion>Consultar Stock> Granel

```
Ingrese la opción de STOCK que desea CONSULTAR:  
1 - TOTAL  
2 - GRANEL  
3 - FRACCIONADO  
4 - SALIR  
Ingrese una opción: 2  
Conexión a la base de datos exitosa.  
  
PRODUCTOS A GRANEL DISPONIBLES:  
(1, 'HARINA INTEGRAL', 3, 10, 'kg', 5, 2, 1)  
(2, 'LENTEJAS', 8, 25, 'kg', 10, 3, 2)  
(3, 'ALMOHADITAS FRUTILLA', 4, 3, 'kg', 8, 2, 3)  
(4, 'NUECES', 7, 1, 'kg', 7, 2, 4)  
(5, 'QUINOA', 5, 5, 'kg', 5, 2, 5)  
DISPONIBLE:  
Conexión a la base de datos exitosa.  
Ingresa el nombre del producto a granel: LENTEJAS  
  
Descripción-Cant.-Tamaño-UM  
('LENTEJAS', 8, 25, 'kg')  
  
Consulta STOCK GRANEL realizada con éxito.
```

Menu Principal>Iniciar Sesion>Consultar Stock>Fraccionado

```
Ingrese la opción de STOCK que desea CONSULTAR:  
1 - TOTAL  
2 - GRANEL  
3 - FRACCIONADO  
4 - SALIR  
Ingrese una opción: 3  
Conexión a la base de datos exitosa.  
  
PRODUCTOS FRACCIONADOS DISPONIBLES:  
(1, 'HARINA INTEGRAL', 2, 250, 'gr', 10, 3, 1)  
(2, 'HARINA INTEGRAL', 2, 500, 'gr', 7, 3, 1)  
(3, 'HARINA INTEGRAL', 5, 1000, 'gr', 5, 2, 1)  
(4, 'LENTEJAS', 5, 1000, 'gr', 25, 5, 2)  
(5, 'ALMOHADITAS FRUTILLA', 2, 250, 'gr', 15, 5, 3)  
(6, 'ALMOHADITAS FRUTILLA', 4, 500, 'gr', 10, 2, 3)  
(7, 'NUECES', 4, 250, 'gr', 8, 2, 4)  
(8, 'NUECES', 4, 500, 'gr', 10, 5, 4)  
(9, 'QUINOA', 2, 500, 'gr', 12, 4, 5)  
(10, 'QUINOA', 3, 1000, 'gr', 8, 2, 5)  
Conexión a la base de datos exitosa.  
Ingresa el nombre del producto fraccionado: QUINOA  
  
Descripción-Cant.-Tamaño-UM  
('QUINOA', 2, 500, 'gr')  
('QUINOA', 3, 1000, 'gr')  
  
Consulta STOCK FRACCIONADO realizada con éxito.
```

Menu Principal>Iniciar Sesion>Consultar Stock> Salir

```
Ingrese la opción de STOCK que desea CONSULTAR:
```

- 1 - TOTAL
- 2 - GRANEL
- 3 - FRACCIONADO

4 - SALIR

```
Ingrese una opción: 4
```

```
Saliendo de la opción CONSULTA de STOCK...
```

```
-----  
--> MENÚ DE OPCIONES <---  
-----
```

- 1 - REGISTRO VENTA
- 2 - CONSULTAR STOCK
- 3 - ACTUALIZAR STOCK
- 4 - ALTA
- 5 - BAJA
- 6 - MODIFICAR
- 7 - SALIR

```
Ingrese una opción: 
```

Menu Principal>Iniciar Sesion>Actualizar Stock

```
-----  
--> MENÚ DE OPCIONES <---  
-----
```

- 1 - REGISTRO VENTA
- 2 - CONSULTAR STOCK
- 3 - ACTUALIZAR STOCK**
- 4 - ALTA
- 5 - BAJA
- 6 - MODIFICAR
- 7 - SALIR

```
Ingrese una opción: 3
```

```
Ingrese la opción de STOCK que desea ACTUALIZAR:
```

- 1 - GRANEL
- 2 - FRACCIONADO
- 3 - SALIR

```
Ingrese una opción: 
```

Menu Principal>Iniciar Sesion>Actualizar Stock>Granel

```
-----  
---> MENÚ DE OPCIONES <---  
-----  
  
1 - REGISTRO VENTA  
2 - CONSULTAR STOCK  
3 - ACTUALIZAR STOCK  
4 - ALTA  
5 - BAJA  
6 - MODIFICAR  
7 - SALIR  
Ingrese una opción: 3  
Ingrese la opción de STOCK que desea ACTUALIZAR:  
1 - GRANEL  
2 - FRACCIONADO  
3 - SALIR  
Ingrese una opción: 1  
Conexión a la base de datos exitosa.  
  
PRODUCTOS A GRANEL DISPONIBLES:  
(1, 'HARINA INTEGRAL', 3, 10, 'kg', 5, 2, 1)  
(2, 'LENTEJAS', 8, 25, 'kg', 10, 3, 2)  
(3, 'ALMOHADITAS FRUTILLA', 4, 3, 'kg', 8, 2, 3)  
(4, 'NUECES', 7, 1, 'kg', 7, 2, 4)  
(5, 'QUINOA', 5, 5, 'kg', 5, 2, 5)  
Conexión a la base de datos exitosa.  
Ingresa el ID del producto a actualizar: 2  
Ingresa la cantidad a agregar: 10  
Stock del producto con ID 2 actualizado en 10 unidades.  
  
Actualización STOCK GRANEL realizada con éxito.
```

Menu Principal>Iniciar Sesion>Actualizar Stock>Fraccionado

```
Ingrese la opción de STOCK que desea ACTUALIZAR:  
1 - GRANEL  
2 - FRACCIONADO  
3 - SALIR  
Ingrese una opción: 2  
Conexión a la base de datos exitosa.  
  
PRODUCTOS FRACCIONADOS DISPONIBLES:  
(1, 'HARINA INTEGRAL', 2, 250, 'gr', 10, 3, 1)  
(2, 'HARINA INTEGRAL', 2, 500, 'gr', 7, 3, 1)  
(3, 'HARINA INTEGRAL', 5, 1000, 'gr', 5, 2, 1)  
(4, 'LENTEJAS', 5, 1000, 'gr', 25, 5, 2)  
(5, 'ALMOHADITAS FRUTILLA', 2, 250, 'gr', 15, 5, 3)  
(6, 'ALMOHADITAS FRUTILLA', 4, 500, 'gr', 10, 2, 3)  
(7, 'NUECES', 4, 250, 'gr', 8, 2, 4)  
(8, 'NUECES', 4, 500, 'gr', 10, 5, 4)  
(9, 'QUINOA', 2, 500, 'gr', 12, 4, 5)  
(10, 'QUINOA', 3, 1000, 'gr', 8, 2, 5)  
Conexión a la base de datos exitosa.  
Ingresa el ID del producto fraccionado a actualizar: 5  
Ingresa la cantidad a agregar: 20  
Stock del producto fraccionado con ID 5 actualizado en 20 unidades.  
  
Actualización STOCK FRACCIONADO realizada con éxito.
```

Menu Principal>Iniciar Sesion>Actualizar Stock>Salir

```
Ingrese la opción de STOCK que desea ACTUALIZAR:  
1 - GRANEL  
2 - FRACCIONADO  
3 - SALIR  
Ingrese una opción: 3  
Saliendo de la opción ACTUALIZACIÓN de STOCK...  
----> MENÚ DE OPCIONES <---  
----  
1 - REGISTRO VENTA  
2 - CONSULTAR STOCK  
3 - ACTUALIZAR STOCK  
4 - ALTA  
5 - BAJA  
6 - MODIFICAR  
7 - SALIR  
Ingrese una opción: |
```

Menu Principal>Iniciar Sesion>Alta

```
----> MENÚ DE OPCIONES <---  
----  
1 - REGISTRO VENTA  
2 - CONSULTAR STOCK  
3 - ACTUALIZAR STOCK  
4 - ALTA  
5 - BAJA  
6 - MODIFICAR  
7 - SALIR  
Ingrese una opción: 4  
Ingresa la opción que desea dar de ALTA:  
1 - PRODUCTOS  
2 - PRESENTACIÓN  
3 - PROVEEDOR  
4 - SALIR
```

Menu Principal>Iniciar Sesion>Alta>Productos

---> MENÚ DE OPCIONES <---

- 1 - REGISTRO VENTA
- 2 - CONSULTAR STOCK
- 3 - ACTUALIZAR STOCK
- 4 - ALTA**
- 5 - BAJA
- 6 - MODIFICAR
- 7 - SALIR

Ingrese una opción: 4

Ingresa la opción que desea dar de ALTA:

- 1 - PRODUCTOS**
- 2 - PRESENTACIÓN
- 3 - PROVEEDOR
- 4 - SALIR

Ingrese una opción: 1

Conexión a la base de datos exitosa.

CATEGORÍAS DISPONIBLES:

- (1, 'HARINA')
- (2, 'CEREALES')
- (3, 'LEGUMBRES')
- (4, 'FRUTOS-SECOS')
- (5, 'OTROS')
- (6, 'HARINA')
- (7, 'CEREALES')
- (8, 'LEGUMBRES')
- (9, 'FRUTOS-SECOS')
- (10, 'OTROS')
- (11, 'HARINA')
- (12, 'CEREALES')
- (13, 'LEGUMBRES')
- (14, 'FRUTOS-SECOS')
- (15, 'OTROS')
- (16, 'HARINA')
- (17, 'CEREALES')
- (18, 'LEGUMBRES')
- (19, 'FRUTOS-SECOS')
- (20, 'OTROS')

Conexión a la base de datos exitosa.

Menu Principal>Iniciar Sesion>Alta >Presentacion

```
Ingresa la opción que desea dar de ALTA:
```

- 1 - PRODUCTOS
- 2 - PRESENTACIÓN
- 3 - PROVEEDOR
- 4 - SALIR

```
Ingrese una opción: 2
```

```
Corrobore que la presentación del producto no existe.
```

```
ID/ NOMBRE/ CANTIDAD/ TAMAÑO/ UM/ MAX/ MIN/ CATEGORÍA
```

```
Conexión a la base de datos exitosa.
```

```
PRODUCTOS A GRANEL DISPONIBLES:
```

- (1, 'HARINA INTEGRAL', 3, 10, 'kg', 5, 2, 1)
- (2, 'LENTEJAS', 18, 25, 'kg', 10, 3, 2)
- (3, 'ALMOHADITAS FRUTILLA', 4, 3, 'kg', 8, 2, 3)
- (4, 'NUECES', 7, 1, 'kg', 7, 2, 4)
- (5, 'QUINOA', 5, 5, 'kg', 5, 2, 5)
- (6, 'PISTACHOS', 21, 1, 'kg', 10, 1, 19)

```
Conexión a la base de datos exitosa.
```

```
Ingresa la descripción del producto: LENTEJAS
```

```
Ingresa la cantidad: 3
```

```
Ingresa el tamaño: 250
```

```
Ingresa la unidad de medida: gr
```

```
Ingresa la cantidad máxima: 100
```

```
Ingresa la cantidad mínima: 10
```

```
Ingresa el ID del producto a granel: 2
```

```
PRESENTACIÓN dada de ALTA con éxito.
```

```
Ingresa la opción que desea dar de ALTA:
```

- 1 - PRODUCTOS
- 2 - PRESENTACIÓN
- 3 - PROVEEDOR
- 4 - SALIR

```
Ingrese una opción: 1
```

Menu Principal>Iniciar Sesion>Alta >Proveedor

```
Ingresa la opción que desea dar de ALTA:  
1 - PRODUCTOS  
2 - PRESENTACIÓN  
3 - PROVEEDOR  
4 - SALIR  
Ingresar una opción: 3  
Conexión a la base de datos exitosa.  
Ingresa el CUIT(XX-XXXXXXX-XX): 30-25411481-8  
Ingresa el nombre del proveedor: Molinos de la Calera  
Ingresa la dirección: Camino a Calera Km 46  
Ingresa el número de teléfono: 351-52415487  
Ingresa el email: molinoscalera@gmail.com  
Ingresa en número de días el tiempo de entrega (ej: 7): 5  
  
PROVEEDOR dado de ALTA con éxito.
```

Menu Principal>Iniciar Sesion>Alta >Salir

```
Ingresa la opción que desea dar de ALTA:  
1 - PRODUCTOS  
2 - PRESENTACIÓN  
3 - PROVEEDOR  
4 - SALIR  
Ingresar una opción: 4  
  
Saliendo de la opción de ALTA...  
  
-----  
--> MENÚ DE OPCIONES <---  
-----  
  
1 - REGISTRO VENTA  
2 - CONSULTAR STOCK  
3 - ACTUALIZAR STOCK  
4 - ALTA  
5 - BAJA  
6 - MODIFICAR  
7 - SALIR  
Ingresar una opción: []
```

El Sistema de Almacenamiento de Datos se lleva a cabo mediante las siguientes tablas:

- **PROVEEDORES:** Guarda la información personal y de contacto de los proveedores y el tiempo de demora en cada entrega, utilizando el CUIT como clave primaria.
- **PRODUCTOSGRANEL:** Maneja la información de productos a granel, incluyendo el idProductosGranel como clave principal, descripción, unidad de medida, la categoría a la que pertenece mediante el idCategoría, límites de stock y cuál es su proveedor mediante el idProdIngreso de la tabla **PRODXPROV**
- **PRESENTACIONES:** Lleva un control de las presentaciones fraccionadas de los productos a granel relacionadas ambas tablas el mediante idProductoGranel, cantidades, su clave principal es idProductoFracc..
- **VENTAS:** Almacena los registros de las ventas mediante fecha, hora, cantidad, idProductoFracc, su clave principal es idVenta. El registro de nuevas operaciones provoca el cambio en la tabla PRESENTACIONES.
- **PRODXPROV:** contiene los datos que relacionan cada proveedor con el producto que provee, y la fecha en la que ingresa al almacén y la fecha de vencimiento. Se relaciona con la tabla proveedores mediante el CUIT del proveedor y con PRODUCTOSGRANEL con el idProductoGranel.
- **CATEGORIAS:** es una pequeña tabla que contiene la descripción de las diferentes categorías en las que se clasifican los productos, siendo el idCategoria su clave principal.

Se implementaron operaciones **CRUD** para añadir, actualizar y eliminar registros en la base de datos MySQL mediante menús. Ellos dan opciones para insertar nuevos registros en las tablas **ProductosGranel**, **Presentaciones** y **Proveedores**. También se pueden modificar registros, como la cantidad de productos tanto a granel como fraccionados o los datos de contacto del proveedor, de manera manual ante algún error o cambio. Y si fuera necesario también eliminar registros de las tablas **Prodprov** y **Proveedores**

Se implementó un **sistema de control de stock** desde el producto a granel hasta su forma fraccionada, por ahora todo de manera manual.

El circuito comienza con la llegada de los productos, se cargan en la tabla **Prodprov** cantidad y la fecha de vencimiento, y en la tabla **ProductosGranel** se actualiza la cantidad existente, luego al ser fraccionados se cargan en la de **Presentaciones** las nuevas cantidades y se restan de la tabla **ProductosGranel**. Cuando se vende, al registrar dicha venta, se actualiza automáticamente las cantidades en la tabla **Presentaciones**.

El menú también ofrece la posibilidad de realizar diferentes **Consultas SQL** para análisis de datos dando la opción de ver la cantidad de a Granel, Fraccionado o total de un producto determinado.

Capturas de pantalla de BBDD

MySQL Workbench - stockagranel

```
use stockagranel;
/* Inserción de datos en la tabla 'categorias' */
INSERT INTO categorias (categoria) VALUES
('HARINA'),
('CEREALES'),
('LEGUMBRES'),
('FRUTOS-SECOS'),
('OTROS');

/* Inserción de datos en la tabla 'proveedores' */
INSERT INTO proveedores (CUIT, proveedor, direccion, telefono, email, tiempoDeEntrega) VALUES
('30-71427900-5', 'BIO SALUD SRL', 'Bv. Los Húngaros 4676', '54 9 3543 40 5354', 'info@bio-salu',
('30-12345678-4', 'NAT S.A', 'San Luis 123', '543534123456', 'natsa@gmail.com', '10'),
('30-71565674-3', 'COMER SANO', 'Mendiolata 2541', '3515646393', 'comersanoche@mail.com', '3');
```

Action	Output
21	18:44:28 SELECT * FROM stockagranel.proveedores LIMIT 1; 3 row(s) returned
22	18:44:30 SELECT * FROM stockagranel.registroventa LIMIT 1; 0 row(s) returned

Consulta CRUD : Insertar → INSERT INTO

Producto:

```
INSERT INTO productosGranel (descripcion, cantidad, tamano, unidadMedida, cantidadMax, cantidadMin, idCategoria) VALUES ('PIMENTON DULCE', 1, 11, 'kg', 3, 2, 5);
```

Presentación:

```
INSERT INTO presentaciones (descripcion, cantidad, tamano, unidadMedida, cantidadMax, cantidadMin, idProductoGranel) VALUES ('PIMENTON DULCE', 10, 10, 'gr', 10, 3, 6);
```

Proveedor:

```
INSERT INTO proveedores (CUIT, proveedor, direccion, telefono, email, tiempoDeEntrega) VALUES ('30-91432930-5', 'BARRO HNOS SRL', 'Av. Patria 1502', 3543405354, 'info@barrohnos.com.ar', 10);
```

Consulta CRUD : Actualizar → UPDATE

Producto:

```
UPDATE productosGranel SET cantidad = 5 WHERE idProductoGranel=1;
```

Proveedor:

```
UPDATE proveedores SET telefono = 3516666666 WHERE idProveedor= 3;
```

Presentación:

```
UPDATE presentaciones SET unidadMedida = "Kg" WHERE  
idProductoFracc= 3;
```

Consulta CRUD : Eliminar → DELETE

PRODUCTO/PROVEEDOR:

```
DELETE FROM prodprov WHERE CUIT= '30-12345678-4';
```

PROVEEDOR:

```
DELETE FROM proveedores WHERE CUIT='30-12345678-4';
```

**Una sola tabla
(mostrando todos los datos sin usar *):**

```
SELECT idProductoFracc,
       descripcion, cantidad, tamano,
       unidadMedida, cantidadMax,
       cantidadMin,
       idProductoGranel FROM
stockagranel.presentaciones;
```

Esta consulta, muestra todo lo que hay en presentaciones.

idproductoFracc	descripcion	cantidad	tamano	unidadMedida	cantidadMax	cantidadMin	idproductoGranel
4	MANZANAS FRUTILLA	5	250	gr	25	5	2
5	MANZANAS FRUTILLA	2	250	gr	15	5	3
6	MANZANAS FRUTILLA	5	500	gr	30	2	3
7	NUCES	4	250	gr	8	2	4
8	NUCES	4	500	gr	10	5	4

**Una sola tabla con
where**

```
SELECT categoria
FROM
stockagranel.categorias
WHERE idCategoria =  
1;
```

**Esta consulta, muestra
una categoría de la
tabla categoria donde el
id es =1**

CATEGORIA
HARINA

Una sola tabla con Select
SELECT fechaBaja FROM
stockagranel.perdidas;
Esta consulta, muestra la
columna *fechaBaja* de la tabla
perdidas.

The screenshot shows the MySQL Workbench interface. In the top right, a query editor window displays the SQL command:

```
1 • SELECT fechaBaja FROM stockagranel.perdidas;
```

The results are shown in a grid titled "Result Grid". The column is labeled "fechaBaja" and contains two rows of data:

fechaBaja
2024-05-11
2024-05-25

Una sola tabla con where
utilizando between:
SELECT idProdIngreso,
fechaIngr, fechaVto,
cantidad, CUIT,
idProductoGranel FROM
stockagranel.prodprov
WHERE cantidad
between 0 AND 6;;
Acá selecciona todos los
productos de la tabla
prodprov donde la
cantidad está entre 0 y 6.

The screenshot shows the MySQL Workbench interface. In the top right, a query editor window displays the SQL command:

```
1 • SELECT idProdIngreso, fechaIngr, fechavto, cantidad, CUIT, idProductoGranel FROM stockagranel.prodprov WHERE cantidad between 0 AND 6;
```

The results are shown in a grid titled "Result Grid". The columns are "idProdIngreso", "fechaIngr", "fechavto", "cantidad", "CUIT", and "idProductoGranel". There are two rows of data:

idProdIngreso	fechaIngr	fechavto	cantidad	CUIT	idProductoGranel
1	2024-05-11	2027-09-02	3	38-7427900-5	1
2	2024-05-12	2025-02-02	4	36-7156003-5	3

At the bottom of the interface, the "prodprov" tab is selected in the "Session" tab bar, showing a list of recent queries and their execution details.

Tabla usando Limit :
SELECT descripcion, cantidad
FROM
productosgranel LIMIT 3;

Esta consulta en la tabla *productosgranel* y limita la búsqueda a 3, y muestra las columnas descripción y cantidad

The screenshot shows the MySQL Workbench interface. In the top right, the Query Editor window displays the following SQL code:

```
1 • SELECT descripcion, cantidad
2   FROM
3     productosgranel LIMIT 3;
```

Below the code, the Result Grid shows the retrieved data:

descripcion	cantidad
HARINA INTEGRAL	3
LENTEJAS	8
ALMOHADITAS FRUTILLA	4

At the bottom of the interface, the Action Output pane shows the history of actions taken on the database:

Time	Action	Message
39 19:51:14	SELECT registrosventas.idProductoFacc, registrosventa.idVenta, presentaciones.idProductoFacc, presentac...	1 row(s) returned
40 20:28:21	SELECT * FROM stockagranl.presentaciones LIMIT 0, 1000	10 row(s) returned
41 20:28:18	SELECT idProductoFacc, descripcion, cantidad, tamano, unidadMedida, cantidadMax, cantidadMin, idProd...	10 row(s) returned
42 20:32:45	SELECT * FROM stockagranl.productos WHERE cantidad between 0 AND 6 LIMIT 0, 1000	2 row(s) returned
43 20:33:44	SELECT idProductoFacc, fechaIngreso, fechaVto, cantidad, CUIT, idProductoGranel FROM stockagranl.producto...	2 row(s) returned
44 20:35:27	SELECT * FROM productosgranel LIMIT 3	3 row(s) returned
45 20:35:54	SELECT descripcion, cantidad FROM productosgranel LIMIT 3	3 row(s) returned

Una sola tabla con where utilizando limit:

SELECT descripcion, cantidad, unidadMedida
FROM productosgranel
WHERE unidadMedida =
'kg' LIMIT 4;

Esta consulta muestra las columnas descripción, cantidad y unidadMedida de la tabla *productosgranel* donde la columna unidadMedida es kg pero limita los resultados a 4

The screenshot shows the MySQL Workbench interface. In the top right, the Query Editor window displays the following SQL code:

```
1 • SELECT descripcion, cantidad, unidadMedida
2   FROM
3     productosgranel
4    WHERE
5      unidadMedida = 'kg' LIMIT 4;
```

Below the code, the Result Grid shows the retrieved data:

descripcion	cantidad	unidadMedida
HARINA INTEGRAL	3	kg
LENTEJAS	8	kg
ALMOHADITAS FRUTILLA	4	kg
NUCES	7	kg

At the bottom of the interface, the Action Output pane shows the history of actions taken on the database:

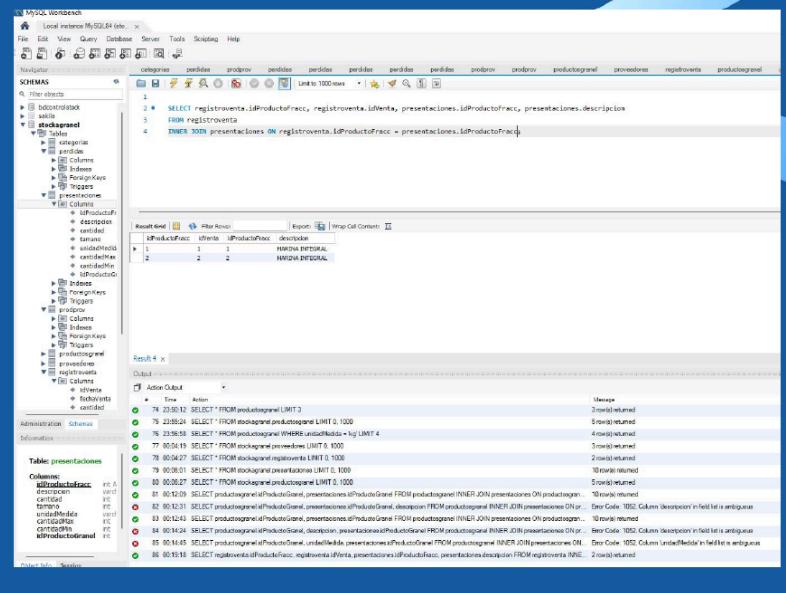
Time	Action	Message
44 20:35:27	SELECT * FROM productosgranel LIMIT 3	3 row(s) returned
45 20:35:54	SELECT descripcion, cantidad FROM productosgranel LIMIT 3	3 row(s) returned
46 20:38:35	SELECT * FROM productosgranel WHERE unidadMedida = 'kg' LIMIT 4	4 row(s) returned
47 20:39:10	SELECT descripcion, cantidad, unidadMedida FROM productosgranel WHERE unidadMedida = 'kg' LIMIT 4	4 row(s) returned
48 20:39:19	SELECT descripcion, cantidad, unidadMedida FROM productosgranel WHERE unidadMedida = 'kg' LIMIT 4	4 row(s) returned
49 20:39:27	SELECT descripcion, cantidad, unidadMedida FROM productosgranel WHERE unidadMedida = 'kg' LIMIT 4	4 row(s) returned
50 20:39:40	SELECT descripcion, cantidad, unidadMedida FROM productosgranel WHERE unidadMedida = 'kg' LIMIT 4	4 row(s) returned

Más de 1 tabla con inner join:

SELECT

```
registroventa.idProductoFracc,  
registroventa.idVenta,  
presentaciones.idProductoFracc,  
presentaciones.descripcion  
FROM registroventa  
INNER JOIN presentaciones ON  
registroventa.idProductoFracc =  
presentaciones.idProductoFracc;
```

Esta consulta selecciona de la tabla *registroventa* los productos que tienen igual valor para *idProductoFracc* y muestra las columnas *ifProductoFracc*, *idVenta* y descripción.



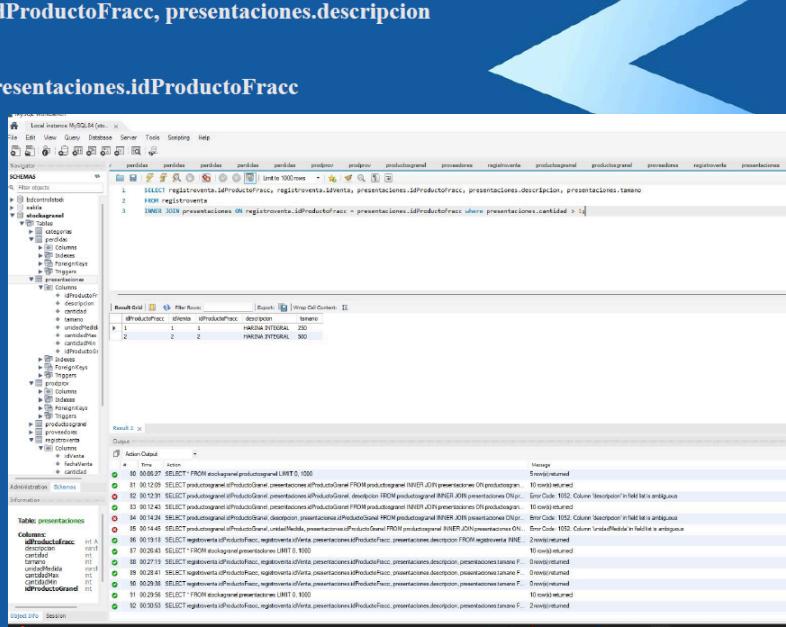
Más de 1 tabla con inner join con filtro: SELECT registroventa.idProductoFracc,

registroventa.idVenta

FROM registroventa
ENRER IONI

```
INNER JOIN presentaciones  
ON registroventa.idProductoFracc = presentaciones.idProducto  
where presentaciones.cantidad > 1;
```

Esta consulta selecciona de la tabla *registroventa* los productos que tienen igual valor para *idProductoFracc* y muestra las columnas *ifProductoFracc*, *idVenta* y descripción, para las presentaciones donde la cantidad es mayor a 1.



Una sola tabla con where utilizando limit:

```
SELECT descripcion, cantidad, tamano, cantidadMax FROM productosgranel WHERE unidadMedida = 'kg' ORDER BY idProductoGranel LIMIT 4;
```

Esta consulta muestra las columnas descripción, cantidad, tamano, cantidadMax la tabla productosgranel donde la columna unidadMedida es kg pero limita los resultados a 4 y los ordena teniendo en cuenta el idProductoGranel

The screenshot shows the MySQL Workbench interface. In the top panel, there is a query editor window titled 'Query 1' containing the SQL code: 'SELECT descripcion, cantidad, tamano, cantidadMax FROM productosgranel WHERE unidadMedida = 'kg' ORDER BY idProductoGranel LIMIT 4;'. Below the query editor is a results grid table with four rows of data. At the bottom of the interface, there is an 'Output' pane displaying several log entries.

descripcion	cantidad	tamano	cantidadMax
HARINA INTEGRAL	3	10	5
ARROZ	8	25	10
ALMOHADITAS FRUTILLA	4	3	9
NUECES	7	1	7

Action Output

- 47: 20:39:10 SELECT descripcion, cantidad, unidadMedida FROM productosgranel WHERE unidadMedida = 'kg' LIMIT 4 4 rows(s) returned
- 48: 20:39:13 SELECT descripcion, cantidad, unidadMedida FROM productosgranel WHERE unidadMedida = 'kg' LIMIT 4 4 rows(s) returned
- 49: 20:39:27 SELECT descripcion, cantidad, unidadMedida FROM productosgranel WHERE unidadMedida = 'kg' LIMIT 4 4 rows(s) returned
- 50: 20:39:40 SELECT descripcion, cantidad, unidadMedida FROM productosgranel WHERE unidadMedida = 'kg' LIMIT 4 4 rows(s) returned
- 51: 20:41:51 SELECT * FROM productosgranel WHERE unidadMedida = 'kg' ORDER BY idProductoGranel LIMIT 4 4 rows(s) returned
- 52: 20:42:36 SELECT * FROM productosgranel WHERE unidadMedida = 'kg' ... Error Code: 1054. Unknown column 'descripcion' in field list
- 53: 20:43:04 SELECT descripcion, cantidad, tamano, cantidadMax FROM productosgranel WHERE unidadMedida = 'kg' ... 4 row(s) returned

Módulo de Gestión de Usuarios y Accesos

Se implementan, utilizando conceptos de POO, operaciones de CRUD para los usuarios, creando las clases necesarias (*Usuario*, *Acceso*, *CrudUsuario*) almacenando los datos de los usuarios en un archivo binario llamado *usuarios.ispc*. Se crea una clase para registrar los intentos de acceso de usuarios en el archivo *accesos.ispc* y los intentos fallidos se registran en *logs.txt*. Validando cada dato en el módulo validaciones. Se pueden añadir, actualizar, eliminar, buscar tanto por DNI como por username y mostrar usuarios. Para estas últimas se implementaron métodos para ordenar y buscar usuarios por diferentes atributos (username, DNI, email) utilizando algoritmos como búsqueda secuencial y búsqueda binaria.

Módulo de Análisis de Datos utilizando Pandas, NumPy y Matplotlib:

Se crean scripts para leer datos de precipitaciones desde archivos CSV, realizar análisis y visualizar los mismos. Se generan conjuntos de datos aleatorios con NumPy para escenarios donde no se encuentre disponible un archivo CSV con los datos del año.

Se analizan datos de precipitaciones anuales y mensuales, calculando valores máximos, mínimos y promedio. Utilizando la librería Matplotlib se pueden visualizar datos a través de Gráficos de barras para las lluvias anuales, Gráficos de dispersión mostrando las precipitaciones diarias por mes, Gráficos circulares para la distribución de precipitaciones en los meses, etc.

Capturas de Pantalla:

Menu Principal>Analisis de Datos

```
-----> MENÚ PRINCIPAL <-----  
1. USUARIOS Y ACCESOS  
2. INICIAR SESIÓN  
3. ANÁLISIS DE DATOS  
4. SALIR  
Seleccione una opción: 3  
Ingrese el año para cargar los registros pluviales: 2024  
Archivo encontrado: registros_pluviales\registroPluvial2024.csv  
---> MENÚ RESGISTROS PLUVIALES <---  
1. Mostrar registros de lluvias por mes  
2. Gráfico de barras de lluvias por mes  
3. Gráfico de dispersión de lluvias por mes  
4. Gráfico circular de lluvias anuales  
5. Salir  
Ingrese la opción que desea ejecutar: |
```

Menu Principal>Analisis de Datos>Mostrar registros de lluvias por mes

```
--> MENÚ DE OPCIONES <---  
  
1 - REGISTRO VENTA  
2 - CONSULTAR STOCK  
3 - ACTUALIZAR STOCK  
4 - ALTA  
5 - BAJA  
6 - MODIFICAR  
7 - SALIR  
Ingrese una opción: 7  
Gracias por utilizar nuestro programa.  
Conexión a la base de datos cerrada.  
-----> MENÚ PRINCIPAL <-----  
1. USUARIOS Y ACCESOS  
2. INICIAR SESIÓN  
3. ANÁLISIS DE DATOS  
4. SALIR  
Seleccione una opción: 3  
Ingrese el año para cargar los registros pluviales: 2024  
Archivo encontrado: registros_pluviales\registroPluvial2024.csv  
---> MENÚ RESGISTROS PLUVIALES <---  
1. Mostrar registros de lluvias por mes  
2. Gráfico de barras de lluvias por mes  
3. Gráfico de dispersión de lluvias por mes  
4. Gráfico circular de lluvias anuales  
5. Salir  
Ingrese la opción que desea ejecutar: 1  
Ingrese el mes (1-12) para mostrar los registros: 5  
Registros pluviales para el mes 5 del año 2024:  
[238, 416, 297, 202, 489, 32, 297, 196, 105, 166, 142, 303, 352, 465, 147, 57, 133, 211, 408, 446, 156, 218, 425, 345, 85, 418, 133, 325, 391, 8, 205]  
---> MENÚ RESGISTROS PLUVIALES <---  
1. Mostrar registros de lluvias por mes  
2. Gráfico de barras de lluvias por mes  
3. Gráfico de dispersión de lluvias por mes  
4. Gráfico circular de lluvias anuales  
5. Salir  
Ingrese la opción que desea ejecutar: |
```

Menu Principal>Analisis de Datos>Grafico registros de lluvias por mes

---> MENÚ RESGISTROS PLUVIALES <---

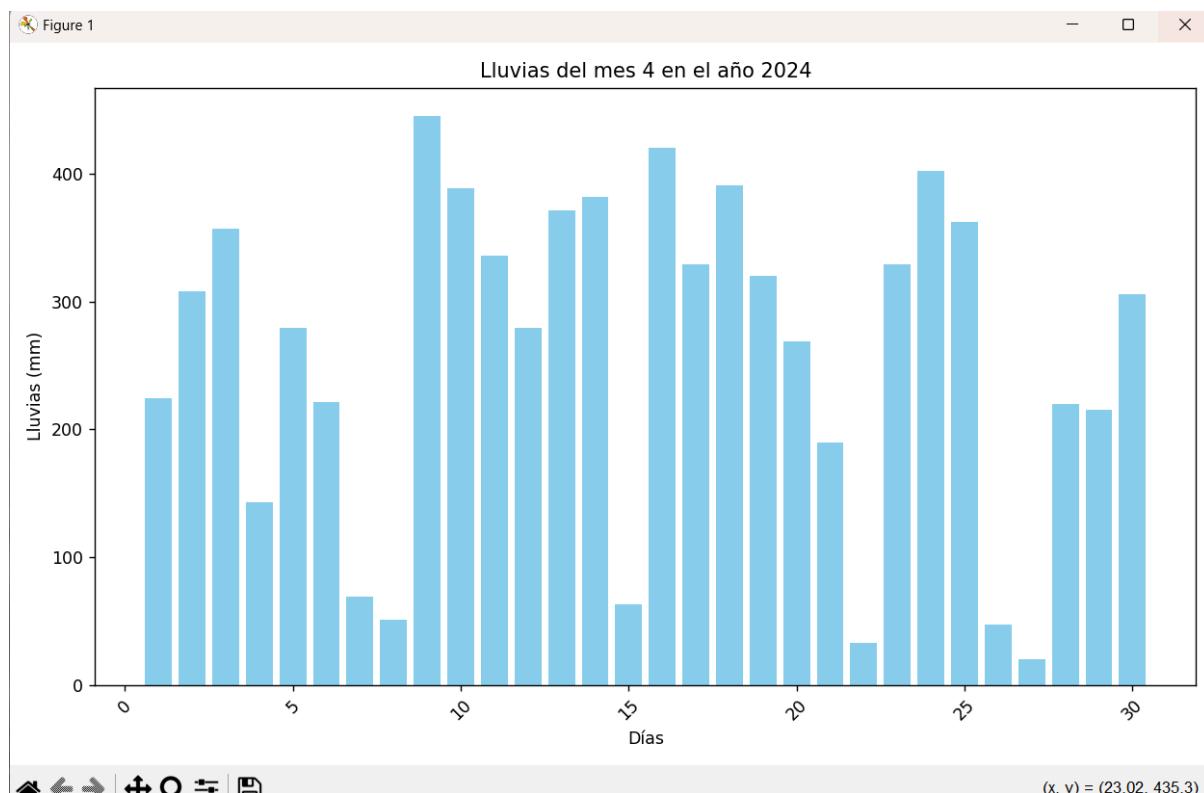
1. Mostrar registros de lluvias por mes
2. Gráfico de barras de lluvias por mes
3. Gráfico de dispersión de lluvias por mes
4. Gráfico circular de lluvias anuales
5. Salir

Ingrese la opción que desea ejecutar: 2

Ingrese el mes (1-12) para generar el gráfico de barras: 4

█

Grafico de barras:



Menu Principal>Analisis de Datos>Grafico de dispersion de lluvias por mes

---> MENÚ RESGISTROS PLUVIALES <---

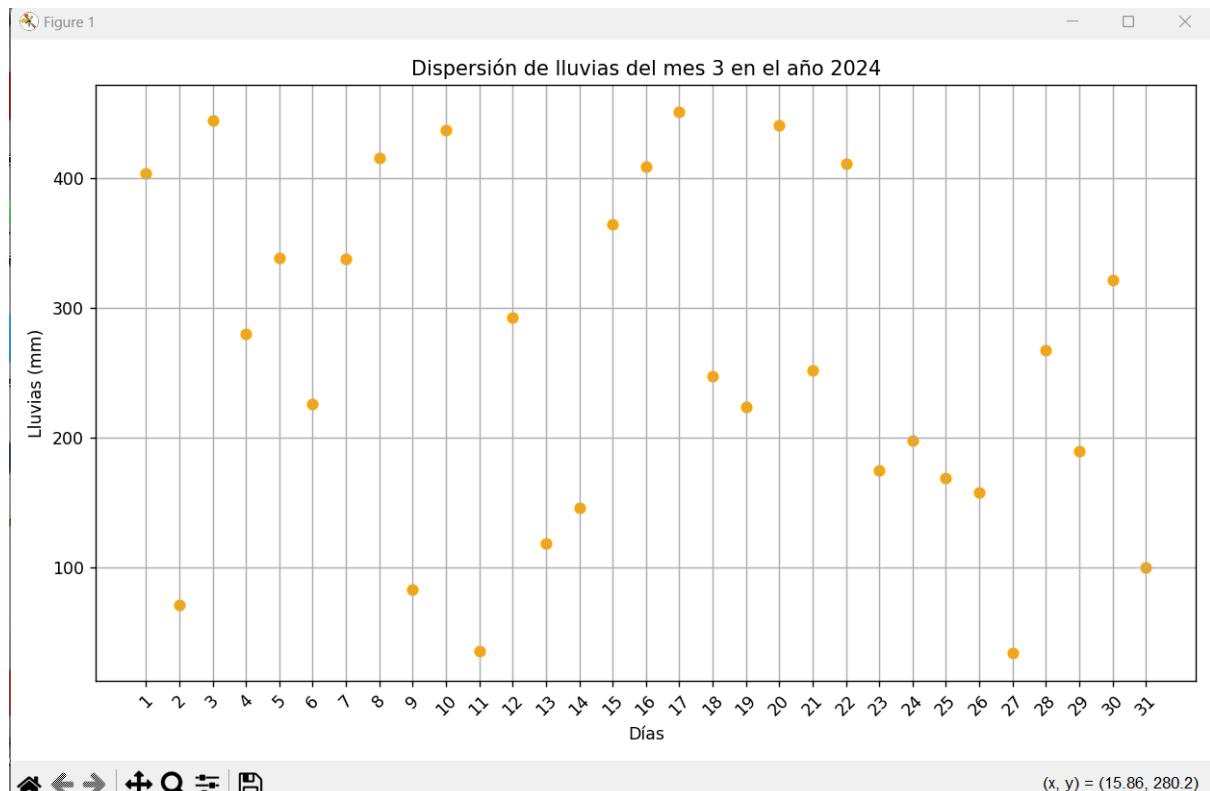
1. Mostrar registros de lluvias por mes
2. Gráfico de barras de lluvias por mes
3. Gráfico de dispersión de lluvias por mes
4. Gráfico circular de lluvias anuales
5. Salir

Ingrese la opción que desea ejecutar: 3

Ingrese el mes (1-12) para generar el gráfico de dispersión: 3

█

Grafico de dispersion:

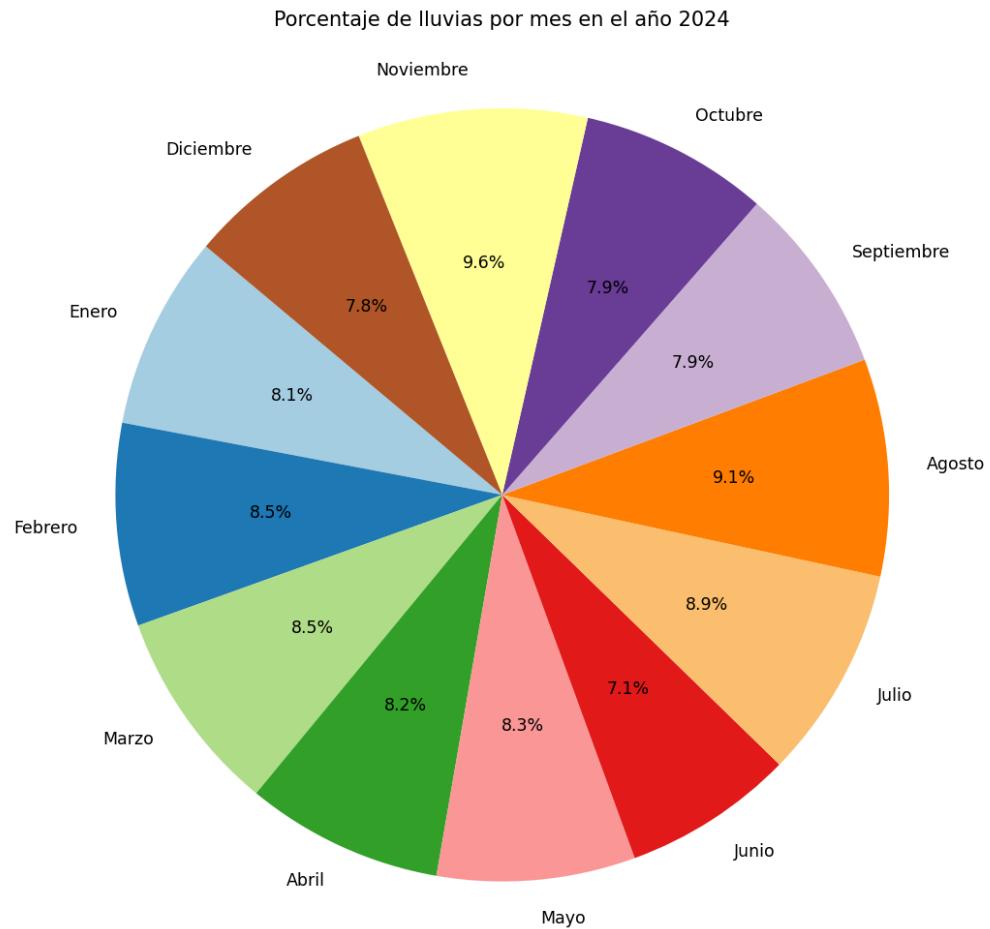


Menu Principal>Analisis de Datos>Grafico circular de lluvias anuales

- ```
--> MENÚ REGISTROS PLUVIALES <--
```
1. Mostrar registros de lluvias por mes
  2. Gráfico de barras de lluvias por mes
  3. Gráfico de dispersión de lluvias por mes
  - 4. Gráfico circular de lluvias anuales**
  5. Salir
- Ingrese la opción que desea ejecutar: 4

Grafico circular de lluvias anuales:

Figure 1



## Conclusión

El proyecto "Control de Stock a Granel" ha permitido desarrollar una solución práctica para la gestión de inventario, atendiendo específicamente a las necesidades de este negocio.

Durante el proceso, los estudiantes pudieron aplicar conocimientos adquiridos en el curso sobre POO, manejo de bases de datos, manejo de archivos, implementación del módulo captcha y análisis de datos. Desarrollar habilidades técnicas en Python y MySQL y habilidades transversales como la organización y la presentación de resultados.

Trabajar de manera colaborativa provocó un gran crecimiento en todos los participantes, destacando la cordialidad, la solidaridad y la empatía entre todos y cada uno de los integrantes del grupo. El proyecto también se vio favorecido por este método de trabajo, aunque quedan algunos items a mejorar e incorporar.

El proyecto también demostró ser una solución viable para el microemprendimiento local que proporcionó los datos, lo cual valida su impacto positivo en la gestión diaria de este tipo de negocios. El proyecto tiene mucho potencial y puede escalar cubriendo todas las necesidades del negocio y crecer con él.