**Desarrollo Programa de Gestión de Compra y Venta**

**Descripción del Proyecto**

Desarrollar un programa que permita gestionar y controlar la compra y venta de vehículos, lotes, casas y artículos varios. El programa incluye las siguientes funcionalidades:

1. **Registro de Productos**:
   * Añadir nuevos productos (vehículos, lotes, casas, artículos varios) con detalles como nombre, descripción, categoría, precio, estado (nuevo/usado), y cantidad disponible.
   * Editar y eliminar productos existentes.
2. **Gestión de Inventario**:
   * Actualizar la cantidad de productos disponibles.
   * Registrar entradas y salidas de productos.
   * Generar alertas cuando el stock de un producto esté por debajo de un nivel mínimo establecido.
3. **Gestión de Compras y Ventas**:
   * Registrar compras y ventas de productos.
   * Generar facturas y recibos para las transacciones.
   * Consultar el historial de compras y ventas.
4. **Consultas y Reportes**:
   * Consultar el estado del inventario en tiempo real.
   * Generar reportes de inventario, incluyendo productos más vendidos, productos con bajo stock, y movimientos de inventario.
   * Generar reportes de ventas y compras, incluyendo ingresos y gastos.
5. **Interfaz de Usuario**:
   * Crear una interfaz amigable y fácil de usar para los empleados.
   * Incluir opciones de búsqueda y filtrado de productos.
6. **Seguridad**:
   * Implementar autenticación de usuarios.
   * Asignar roles y permisos para diferentes niveles de acceso.

**Requisitos Técnicos**

* El programa se desarrollará en Python.
* Utilizar una base de datos Excel o una base de datos fácil de manipular.
* La interfaz de usuario debe ser desarrollada utilizando HTML/CSS o una fácil de manipular.

**Estructura de Archivos**

gestion\_inventario/

├── main.py

├── inventory.py

├── sales.py

├── purchases.py

├── reports.py

├── user\_management.py

├── app/

│ ├── \_\_init\_\_.py

│ ├── models.py

│ ├── routes.py

│ ├── templates/

│ │ └── ... (archivos HTML)

│ ├── static/

│ │ └── ... (archivos CSS, JavaScript)

├── .env

├── init\_db.py

├── venv/ # Entorno virtual

**Paso a Paso para Crear y Configurar los Archivos**

**Paso 1: Crear las Carpetas y Archivos**

1. **Crea la carpeta principal del proyecto**:
2. mkdir gestion\_inventario
3. cd gestion\_inventario
4. **Crea los archivos principales**:
5. type nul > main.py
6. type nul > inventory.py
7. type nul > sales.py
8. type nul > purchases.py
9. type nul > reports.py
10. type nul > user\_management.py
11. **Crea la estructura de la aplicación Flask**:
12. mkdir app
13. mkdir app\templates
14. mkdir app\static
15. type nul > app\\_\_init\_\_.py
16. type nul > app\models.py
17. type nul > app\routes.py
18. type nul > .env
19. type nul > init\_db.py

**Paso 2: Configurar los Archivos**

1. **Archivo .env**:
   * Abre el archivo .env y añade la configuración de la base de datos:
   * DATABASE\_URI=sqlite:///site.db
2. **Archivo app/\_\_init\_\_.py**:
   * Configura la aplicación Flask y la base de datos:
   * from flask import Flask
   * from flask\_sqlalchemy import SQLAlchemy
   * from dotenv import load\_dotenv
   * import os
   * load\_dotenv()
   * db = SQLAlchemy()
   * def create\_app():
   * app = Flask(\_\_name\_\_)
   * app.config['SQLALCHEMY\_DATABASE\_URI'] = os.getenv('DATABASE\_URI')
   * db.init\_app(app)
   * with app.app\_context():
   * from . import routes, models
   * db.create\_all()
   * return app
3. **Archivo app/models.py**:
   * Define los modelos de la base de datos:
   * from . import db
   * class Producto(db.Model):
   * id = db.Column(db.Integer, primary\_key=True)
   * nombre = db.Column(db.String(100), nullable=False)
   * descripcion = db.Column(db.Text)
   * categoria = db.Column(db.String(50))
   * precio = db.Column(db.Float)
   * estado = db.Column(db.String(20))
   * cantidad = db.Column(db.Integer)
   * # Define otros modelos (Venta, Compra, etc.) de manera similar
4. **Archivo app/routes.py**:
   * Define las rutas de la aplicación:
   * from flask import render\_template, request, redirect, url\_for
   * from .models import Producto
   * from . import db, create\_app
   * app = create\_app()
   * @app.route('/')
   * def index():
   * productos = Producto.query.all()
   * return render\_template('index.html', productos=productos)
   * @app.route('/add', methods=['POST'])
   * def add\_producto():
   * nombre = request.form.get('nombre')
   * descripcion = request.form.get('descripcion')
   * categoria = request.form.get('categoria')
   * precio = request.form.get('precio')
   * estado = request.form.get('estado')
   * cantidad = request.form.get('cantidad')
   * nuevo\_producto = Producto(nombre=nombre, descripcion=descripcion, categoria=categoria, precio=precio, estado=estado, cantidad=cantidad)
   * db.session.add(nuevo\_producto)
   * db.session.commit()
   * return redirect(url\_for('index'))
5. **Archivo init\_db.py**:
   * Inicializa la base de datos:
   * from app import create\_app
   * app = create\_app()
6. **Archivos inventory.py, sales.py, purchases.py, reports.py, user\_management.py**:
   * Define las funciones y clases necesarias para gestionar el inventario, ventas, compras, reportes y gestión de usuarios.

**Implementaciones Realizadas**

**Configuración de la Base de Datos**

1. **Archivo .env**:
   * Crea un archivo .env en la raíz del proyecto y agrega la configuración de la base de datos:
   * DATABASE\_URI=sqlite:///site.db
2. **Archivo app/\_\_init\_\_.py**:
   * Configura la aplicación Flask y la base de datos:
   * from flask import Flask
   * from flask\_sqlalchemy import SQLAlchemy
   * from dotenv import load\_dotenv
   * import os
   * load\_dotenv()
   * db = SQLAlchemy()
   * def create\_app():
   * app = Flask(\_\_name\_\_)
   * app.config['SQLALCHEMY\_DATABASE\_URI'] = os.getenv('DATABASE\_URI')
   * db.init\_app(app)
   * with app.app\_context():
   * from . import routes, models
   * db.create\_all()
   * return app
3. **Archivo app/models.py**:
   * Define los modelos de la base de datos (productos, ventas, compras, etc.):
   * from . import db
   * class Producto(db.Model):
   * id = db.Column(db.Integer, primary\_key=True)
   * nombre = db.Column(db.String(100), nullable=False)
   * descripcion = db.Column(db.Text)
   * categoria = db.Column(db.String(50))
   * precio = db.Column(db.Float)
   * estado = db.Column(db.String(20))
   * cantidad = db.Column(db.Integer)
   * # Define otros modelos (Venta, Compra, etc.) de manera similar
4. **Inicializar la Base de Datos**:
   * Crea un archivo init\_db.py en la raíz de tu proyecto:
   * from app import create\_app
   * app = create\_app()
   * Ejecuta el archivo init\_db.py para inicializar la base de datos:
   * python init\_db.py

**Herramientas Útiles**

* **Git**: Para el control de versiones.
* **SQLiteStudio o DB Browser for SQLite**: Para visualizar y gestionar bases de datos SQLite.
* **Postman**: Para probar APIs (si decides crear una API).