**Idea de proyecto:**

Almacén de Ropa

|  |  |
| --- | --- |
| **INTEGRANTES** | **ROL** |
| Jesus Antonio Sanchez Triviño Cod 78257 | *Product owner* |
| Oscar Leonardo Fuentes Cod.78161 | *Scrum master* |
| Alberto Cabrera Vanegas Cod.78272 | Equipo de desarrollo |
| Gustavo Adolfo Correa Rodriguez Cod.78256 | Equipo de desarrollo |
| German Andres Saenz Morales Cod. 782585 | Equipo de desarrollo |

**Idea de proyecto:**

**Proyecto: Sistema de Gestión de Almacén de Ropa**

**Descripción del Proyecto**

Desarrollar una aplicación para gestionar el inventario de un almacén de ropa. La aplicación permitirá registrar, actualizar y consultar el stock de prendas, gestionar pedidos y realizar un seguimiento de las ventas.

**Roles en Scrum**

* **Product Owner**: Responsable de definir las características y prioridades del producto.
* **Scrum Master**: Facilita el proceso Scrum y elimina impedimentos.
* **Equipo de Desarrollo**: Desarrolladores y testers que implementan las funcionalidades.

**Backlog del Producto**

1. **Gestión de Inventario**:
   * Registrar nuevas prendas.
   * Actualizar información de prendas existentes.
   * Consultar stock disponible.
2. **Gestión de Pedidos**:
   * Crear y gestionar pedidos de clientes.
   * Actualizar el estado de los pedidos.
   * Generar facturas.
3. **Seguimiento de Ventas**:
   * Registrar ventas realizadas.
   * Generar reportes de ventas.
   * Analizar tendencias de ventas.
4. **Autenticación y Autorización**:
   * Registro y login de usuarios.
   * Roles y permisos (administrador, empleado).
5. **Interfaz de Usuario**:
   * Diseño de una interfaz amigable y fácil de usar.
   * Panel de control con estadísticas y gráficos.

**Sprints**

Dividir el proyecto en sprints de 2 semanas. Cada sprint debe tener un objetivo claro y entregables específicos.

**Sprint 1**: Configuración del entorno y gestión de inventario.

* Configurar el entorno de desarrollo.
* Implementar la funcionalidad de registro y consulta de prendas.

**Sprint 2**: Gestión de pedidos.

* Implementar la creación y gestión de pedidos.
* Actualizar el estado de los pedidos.

**Sprint 3**: Seguimiento de ventas.

* Registrar ventas y generar reportes.
* Analizar tendencias de ventas.

**Sprint 4**: Autenticación y autorización.

* Implementar el registro y login de usuarios.
* Definir roles y permisos.

**Sprint 5**: Interfaz de usuario.

* Diseñar y desarrollar la interfaz de usuario.
* Integrar estadísticas y gráficos en el panel de control.

**Tecnologías**

* **Backend**: Python (Django o Flask)
* **Frontend**: HTML, CSS, JavaScript (React o Vue.js)
* **Base de Datos**: PostgreSQL o MySQL
* **Control de Versiones**: Git (Repositorio)

**Herramientas de Scrum**

* **Tablero POWER BI**: Para visualizar el progreso de las tareas.
* **Reuniones diarias**: Para sincronizar al equipo y resolver impedimentos.
* **Revisión de Sprint**: Para demostrar el trabajo completado.
* **Retrospectiva de Sprint**: Para identificar mejoras en el proceso.

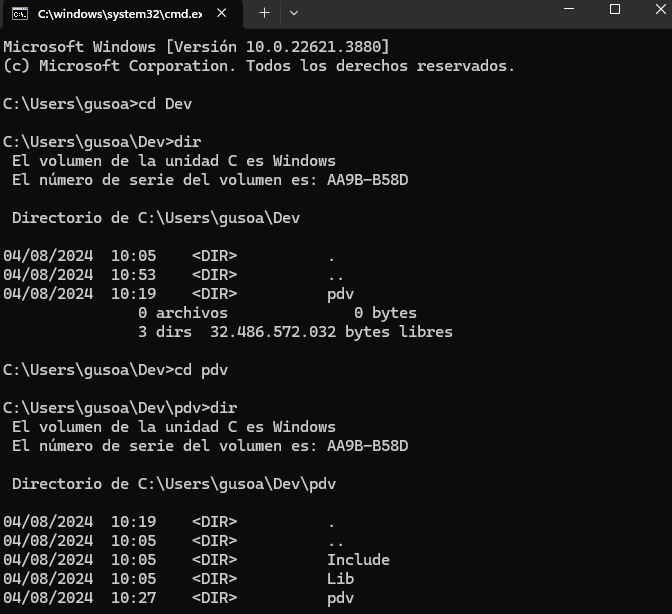
Este proyecto no solo te permitirá aplicar Scrum de manera efectiva, sino que también te dará la oportunidad de desarrollar una aplicación completa y funcional en Python.

1. **Diagrama de Casos de Uso**:
   * Representa las interacciones entre los actores (usuarios o sistemas externos) y los casos de uso del sistema.
   * Puedes mostrar cómo los usuarios interactúan con las funcionalidades, como registrar prendas, gestionar pedidos, etc.
2. **Diagrama de Clases**:
   * Muestra las clases (modelos) en tu aplicación y sus relaciones.
   * Incluye entidades como “Prenda”, “Pedido”, “Usuario”, etc.
   * Indica las asociaciones, herencias y atributos de cada clase.
3. **Diagrama de Secuencia**:
   * Describe la secuencia de interacciones entre objetos o componentes en un escenario específico.
   * Puedes utilizarlo para mostrar cómo se realiza una venta, cómo se actualiza el inventario, etc.
4. **Diagrama de Componentes**:
   * Representa los componentes físicos o lógicos de tu sistema.
   * Muestra cómo se agrupan los módulos o paquetes en tu aplicación.
5. **Diagrama de Despliegue**:
   * Describe la arquitectura física de tu aplicación, incluyendo servidores, bases de datos y otros componentes.
   * Indica cómo se despliegan los artefactos en la infraestructura.

Recuerda que estos diagramas son herramientas poderosas para comunicar tus ideas y decisiones de diseño a todo el equipo. Puedes utilizar herramientas como draw.io, Lucidchart o incluso herramientas específicas de modelado UML para crearlos.

CREAR LA APLICACION

1. **Configuración del Entorno**:
   * Asegúrate de tener Python instalado en tu sistema. Puedes descargarlo desde python.org.
   * Instala un entorno virtual (por ejemplo, virtualenv) para aislar las dependencias de tu proyecto.
2. **Backend**:
   * Elige entre Django o Flask como tu framework de backend:
     + **Django**:
       - Crea un nuevo proyecto Django: django-admin startproject myinventory.
       - Define tus modelos (por ejemplo, Producto, Categoría, etc.) en models.py.
       - Genera las migraciones y aplica los cambios a la base de datos: python manage.py makemigrations y python manage.py migrate.
       - Crea vistas y rutas en views.py y urls.py.
       - Implementa la lógica de negocio para agregar, modificar y eliminar productos.
     + **Flask**:
       - Crea una aplicación Flask: crea un archivo app.py.
       - Define tus rutas y vistas en app.py.
       - Utiliza SQLAlchemy para manejar la base de datos (PostgreSQL o MySQL).
3. **Frontend**:
   * Elige entre React o Vue.js como tu framework de frontend:
     + **React**:
       - Crea componentes para listar productos, agregar nuevos, modificar y eliminar.
       - Utiliza Axios para realizar solicitudes HTTP al backend.
     + **Vue.js**:
       - Crea componentes Vue para las mismas funcionalidades.
       - Utiliza Vue Router para la navegación.
4. **Base de Datos**:
   * Configura PostgreSQL o MySQL como tu base de datos.
   * Define los modelos de datos (por ejemplo, Producto, Categoría) con sus relaciones.
5. **Control de Versiones**:
   * Inicializa un repositorio Git en tu proyecto: git init.
   * Realiza commits periódicos para mantener un historial claro.
6. **Despliegue**:
   * Considera desplegar tu aplicación en un servidor (por ejemplo, Heroku o DigitalOcean)



Texto

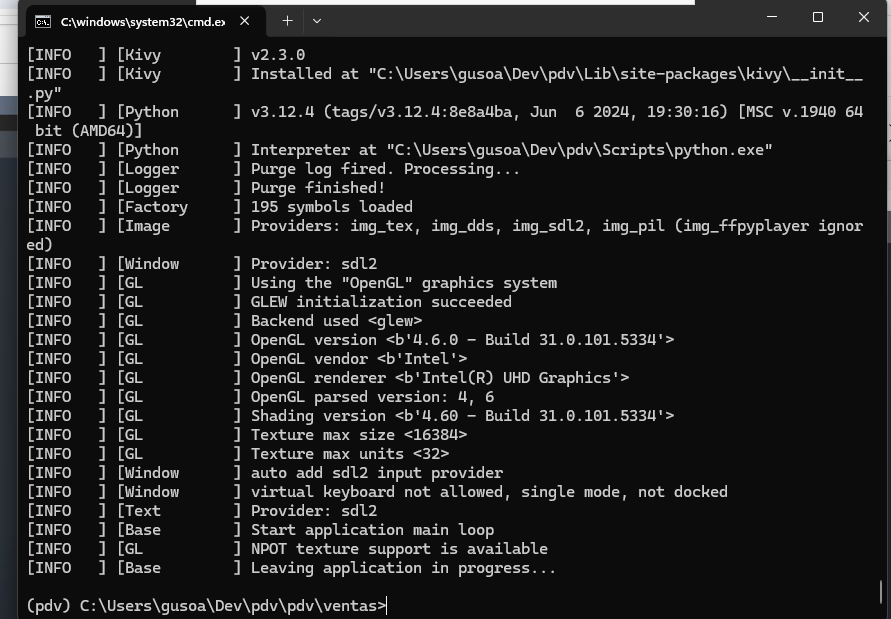
Descripción generada automáticamente

Pantalla de computadora con un texto en blanco

Descripción generada automáticamente con confianza media

Texto

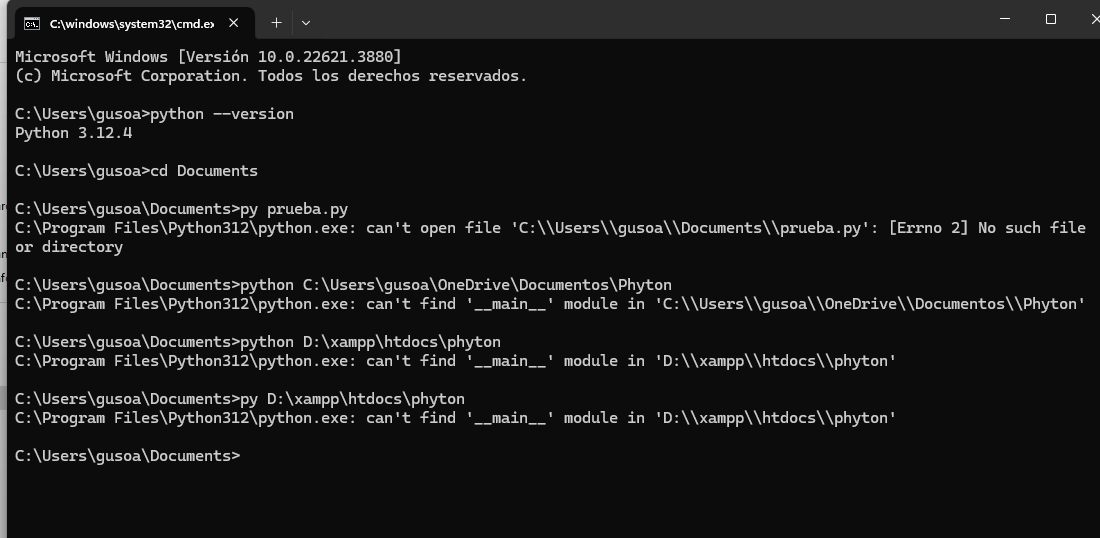
Descripción generada automáticamente



Hexadecimal color en Google OJOOO 65686c gris claro

[Guía: Cómo hacer el backend con Python - Paso a paso (scriptologia.com)](https://scriptologia.com/guia-como-hacer-el-backend-con-python-paso-a-paso/#Crear_un_nuevo_proyecto_de_Python)

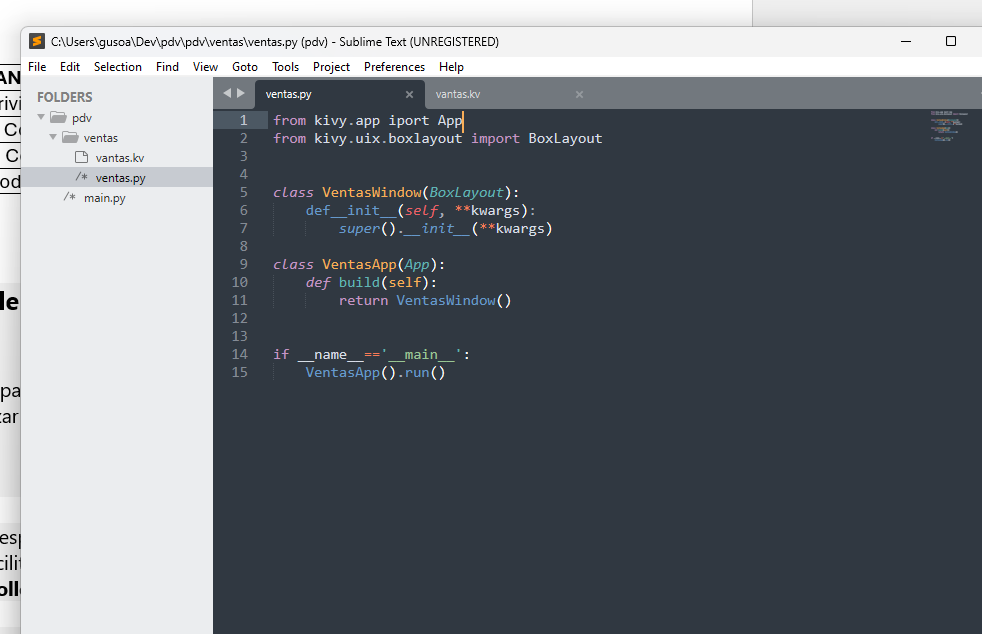
[InventarioSQL/\_\_pycache\_\_ at main · valentinbellini/InventarioSQL · GitHub](https://github.com/valentinbellini/InventarioSQL/tree/main/__pycache__)

<https://www.youtube.com/watch?v=6-D14yA86jQ&list=PLGUsAPwPODlhPwx_t66va4vIV7fkluAa1> programa pyton para crear almacen

Texto

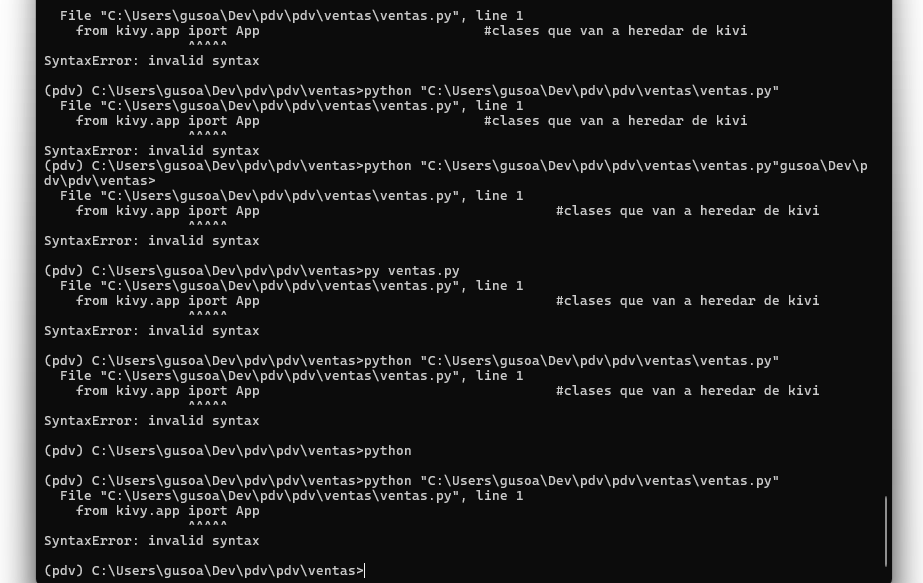
Descripción generada automáticamente

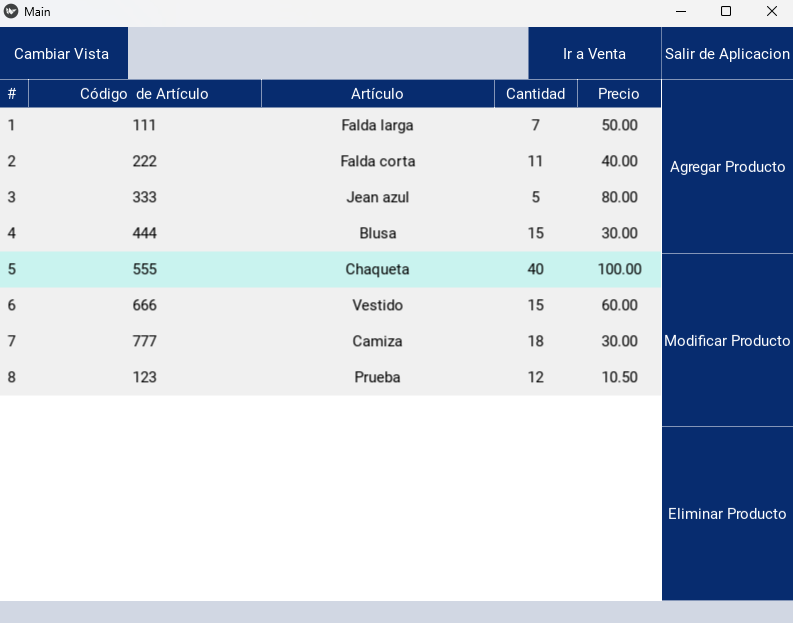
**sublimit**

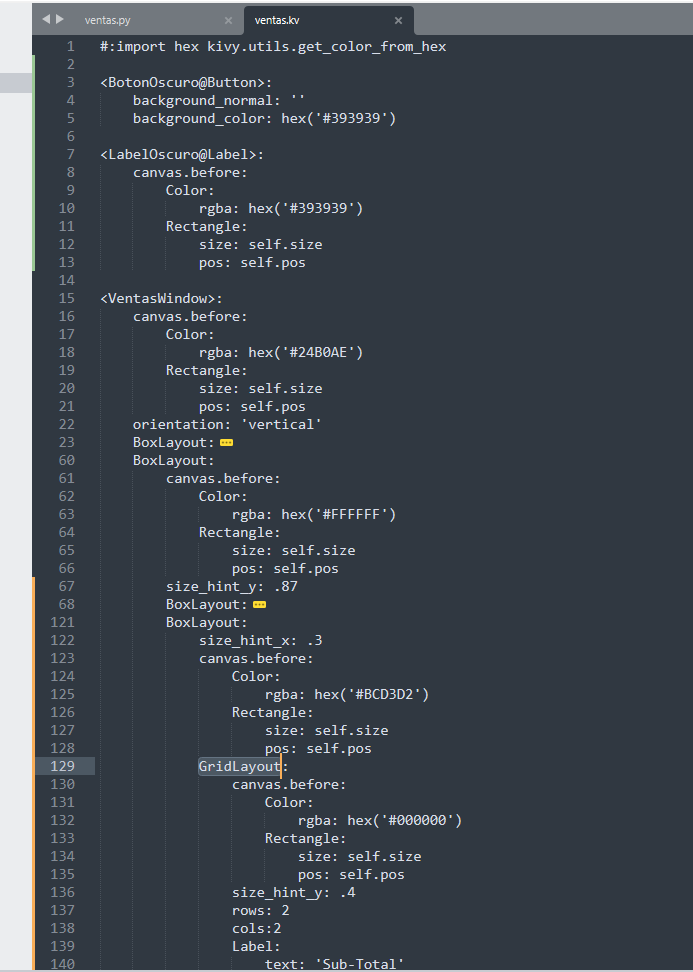


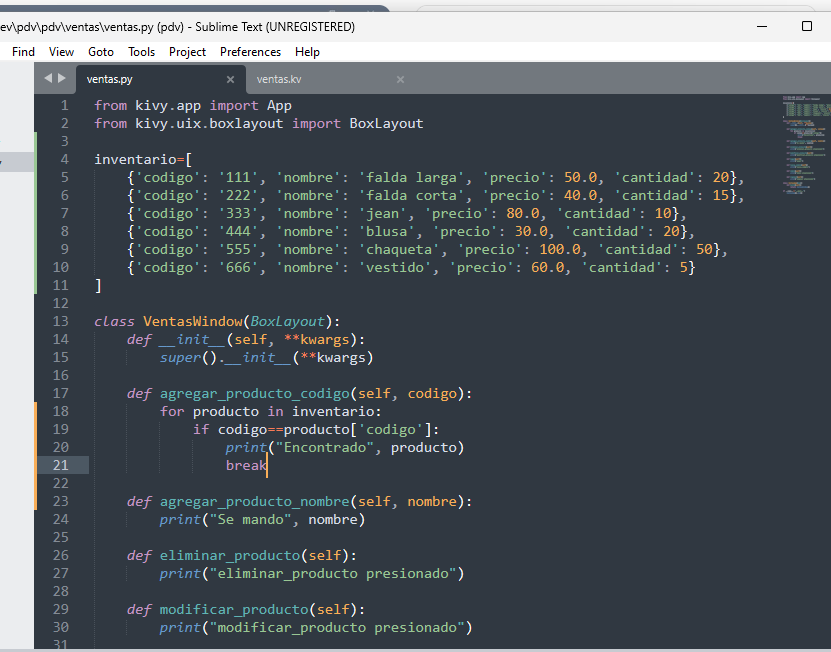
Texto

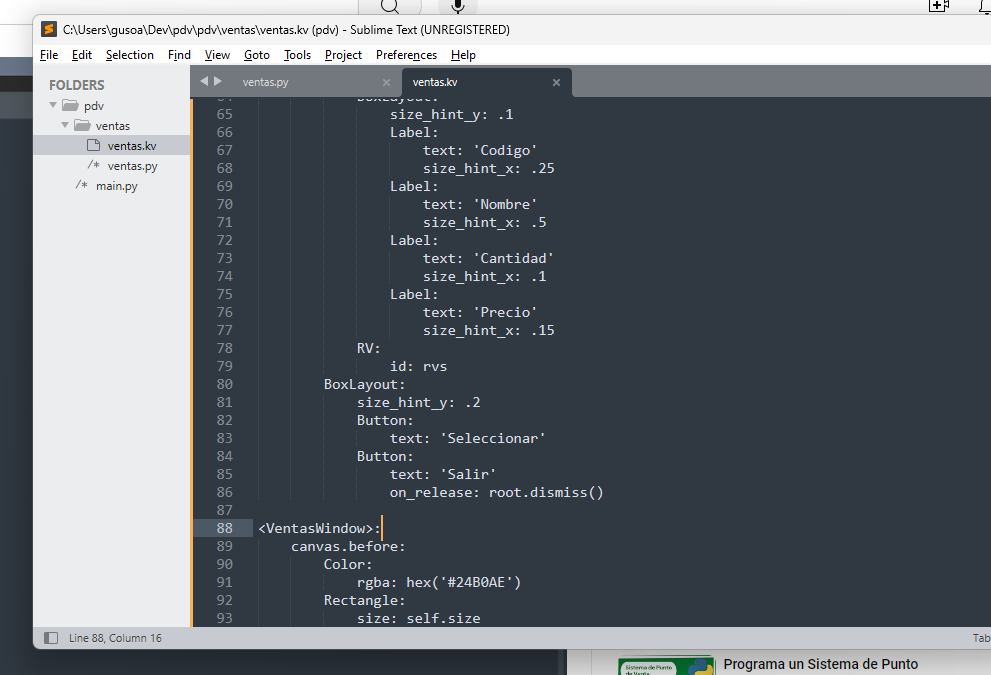
Descripción generada automáticamente



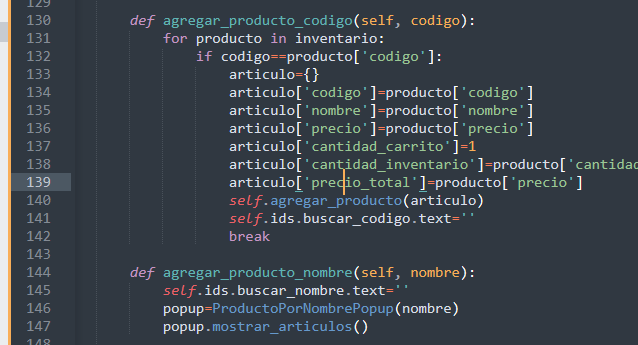


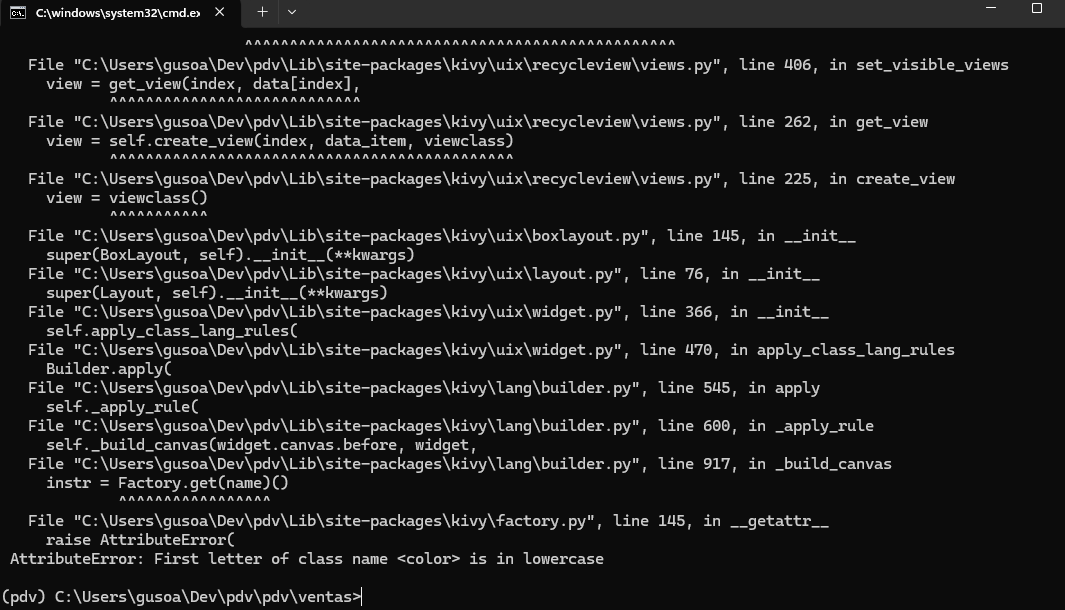


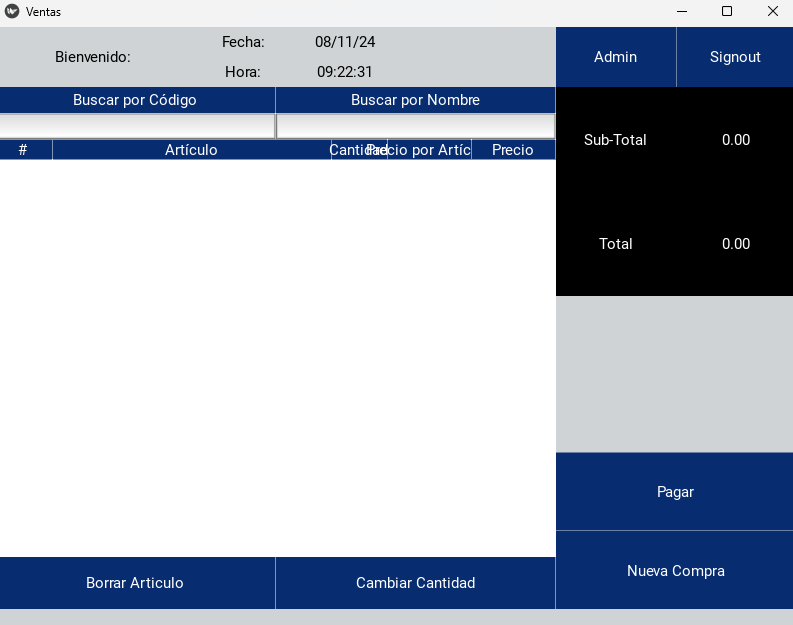




Error







[sqlite3](https://docs.python.org/es/3.8/library/sqlite3.html#module-sqlite3) — DB-API 2.0 interfaz para bases de datos SQLite

**Código fuente:** [Lib/sqlite3/](https://github.com/python/cpython/tree/3.8/Lib/sqlite3/)

SQLite es una biblioteca de C que provee una base de datos ligera basada en disco que no requiere un proceso de servidor separado y permite acceder a la base de datos usando una variación no estándar del lenguaje de consulta SQL. Algunas aplicaciones pueden usar SQLite para almacenamiento interno. También es posible prototipar una aplicación usando SQLite y luego transferir el código a una base de datos más grande como PostgreSQL u Oracle.

Texto

Descripción generada automáticamente