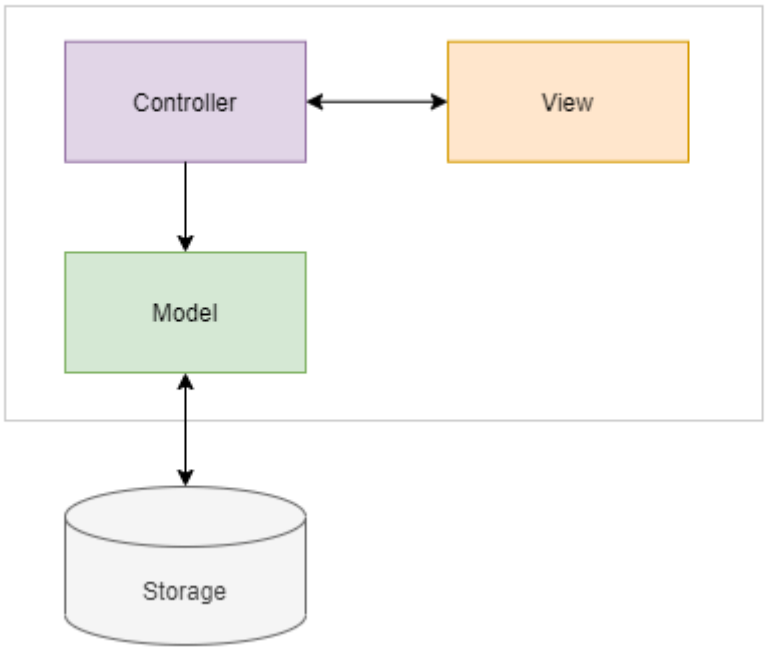
Proyecto Sistema de Gestión de Stock de vinoteca basado Tkinter MVC

Grupo nro 9

A medida que su aplicación crece, su complejidad también aumenta. Para que la aplicación sea más manejable, puede utilizar el patrón de diseño modelo-vista-controlador.

El patrón de diseño MVC le permite dividir la aplicación en tres componentes principales: modelo, vista y controlador. Esta estructura lo ayuda a concentrarse en la lógica de cada parte y hacer que sea más fácil de mantener, especialmente cuando la aplicación crece.

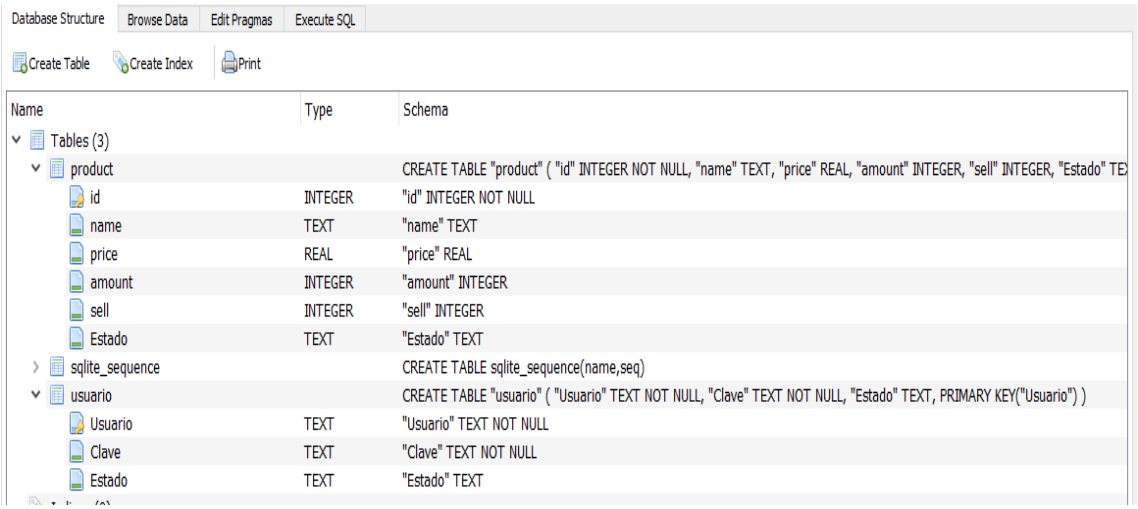
El siguiente diagrama ilustra el patrón de diseño de MVC:



Modelo

Un modelo en MVC representa los datos. Un modelo se ocupa de obtener datos o escribir datos en un almacenamiento, como una base de datos o un archivo. El modelo también puede contener la lógica para validar los datos para garantizar la integridad de los datos. El modelo no debe depender de la vista y el controlador.

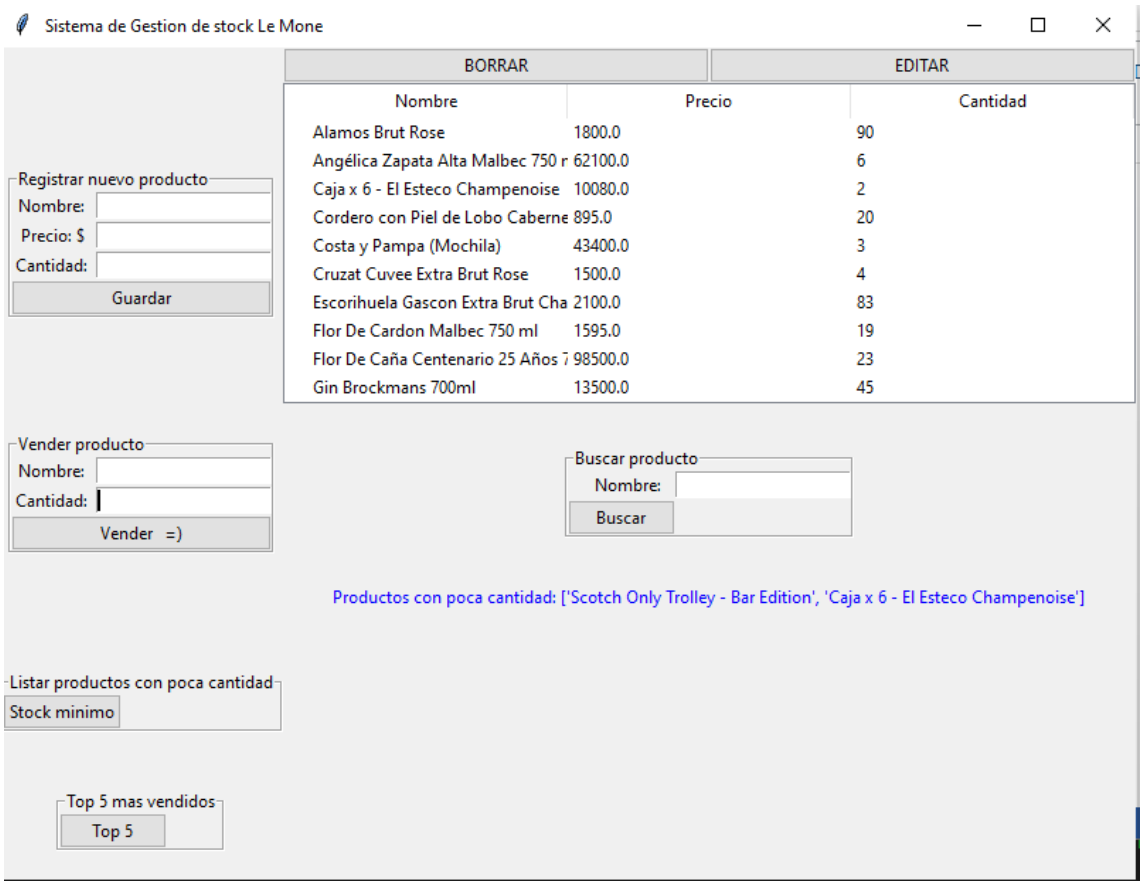
El proyecto tiene una única base de datos que consta de dos tablas, la primera se denomina product y almacena todos los atributos de producto, la segunda se llama usuario y almacena usuario y clave de acceso al sistema.



Vista

Una vista es la interfaz de usuario que representa los datos en el modelo. La vista no se comunica directamente con el modelo. La vista se comunica con el controlador directamente. En las aplicaciones Tkinker, la vista es la ventana raíz que consta de widgets.

Vista del proyecto.



Controlador

Un controlador actúa como intermediario entre las vistas y los modelos. El controlador enruta los datos entre las vistas y los modelos.

Por ejemplo, si los usuarios hacen clic en el botón Guardar en la vista, el controlador enruta la acción "Guardar" al modelo para guardar los datos en una base de datos y notificar a la vista que muestre un mensaje.

Controlador del proyecto.

El archivo stock.py tiene la definición de la conexión de la base de datos, los diferentes componentes que integran la vista, y las funciones para el CRUD de los registros de la base de datos en mysql.

```
from tkinter import ttk
from tkinter import *
import tkinter as tk

import sqlite3

import os.path

class Product:
    # Conexion a la base de datos
    db_name = 'database.db'

    def __init__(self, window):
        # Inicializaciones
        self.wind = window
        self.wind.title(' Sistema de Gestion de stock Le Mone')

        #######
        # Agregar Producto
        frame = LabelFrame(self.wind, text = 'Registrar nuevo producto')
        frame.grid(row = 1, column = 0, columnspan = 3, pady = 20)

        # Nombre de producto
        Label(frame, text = 'Nombre: ').grid(row = 1, column = 0)
        self.name = Entry(frame)
```

```python
# Busqueda de producto
def search_product(self, name_search):
    if name_search != '':
        query = 'SELECT * FROM product ORDER BY name DESC'
        db_rows = self.run_query(query)
        for row in db_rows:
            if(row[1] == name_search or name_search in row[1]):
                self.message['text'] = 'Nombre: '+row[1]+'   Precio: $'+str(row[2])+'   Cantidad: '+str(row[3])
                return
        self.message['text'] = '{} no esta en la lista'.format(name_search)
    else:
        self.message_error['text'] = 'Escriba un nombre para buscar'


# Function to Execute Database Querys
def run_query(self, query, parameters = ()):
    with sqlite3.connect(self.db_name) as conn:
        cursor = conn.cursor()
        result = cursor.execute(query, parameters)
        conn.commit()
    return result
```

```python
47
48
49  if __name__ == '__main__':
50      window = Tk()
51      application = Product(window)
52      window.mainloop()
53
```

Se seleccionó esta librería que se utiliza para realizar aplicaciones de escritorio en python, por la forma simple de implementación de POO en los objetos de la vista y la definición del controlador que ejecute las acciones de registro de producto, venta simplificada, búsqueda, modificación, eliminación de artículos de stock, como así también conocer el top 5 de los artículos más vendidos.