Programación Orientado a Objetos (Herencia y Relaciones)

Emmanuel Velez Lopez
Docente: Alejandro Rodas Vásquez
Universidad Tecnológica de Pereira

30 de abril de 2024

Introducción

Una de las claves para realizar este proyecto es aplicar el concepto de modularidad en la construcción de la Arquitectura de Software que soporta la aplicación.

1. Requerimientos Funcionales

Usted ha sido contratado para realizar un sistema de facturación para una tienda agrícola. Donde cada factura (o Pedido) está compuesto de los productos que serán comprados.

Esta tienda solamente maneja Productos de Control (Fertilizantes y Controles de plagas) y medicina para animales de granja, precisamente antibióticos.

Los Productos de Control tendrán como características un registro ICA, el nombre del producto y la frecuencia de aplicación (es decir, cada cuanto periodo se aplica el producto. Cada 15 días, cada 30 días, etc) así como también el valor del producto. Tenga en cuenta que el Control de Plagas y el Control de Fertilizantes son *un tipo de* Productos de Control, donde el primero tiene como característica un periodo de carencia (es el tiempo legalmente establecido, expresado usualmente en número de días que debe transcurrir entre la última aplicación de un fitosanitario y la cosecha) y el segundo la fecha de la última aplicación de este Producto.

Por otro lado, en la tienda se venden antibióticos para bovinos y porcinos donde las características de este producto son: nombre del producto, dosis (entre 400Kg y 600Kg), tipo de animal al que se le puede aplicar (Bovinos, caprinos o porcinos) y precio.

Tenga en cuenta que al ser una tienda agrícola los Clientes (con atributos nombre y cédula) son habituales por lo tanto el mismo cliente puede tener dentro de su historial de

compras, muchas Pedidos (o Facturas) asociadas. Una Factura como tal debe tener fecha en que se realizó la factura y el valor total de la compra.

> Tenga en cuenta que todos los atributos de las clases son *obligatorios*. Con esta información puede diseñar los *casos de prueba*.

2. Requerimientos de la Arquitectura de Software

Esta aplicación debe de ser construida bajo los siguientes parámetros arquitectónicos:

- Los componentes para separar responsabilidades (Modelo, Test, UI, CRUD) (Figura 1).
- Debe de realizar la pruebas donde se verifique las asociaciones entre las clases *(Relaciones y Herencias)*
- Utilicen el concepto de *Módulos y NameSpace*.
- Cada *Clase* debe de estar en un archivo separado dentro del *Componente* de *Modelo*.

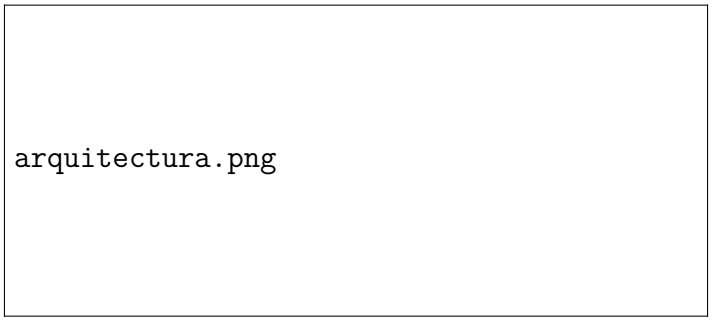


Figura 1: Arquitectura por Capas del Sistema

El sistema deberá:

1. Crear un registro con los *Clientes* de la tienda con sus correspondientes compras (*Facturas*).
2. Vender los Productos de Control señalados.
3. Implementar la función *buscar_por_cedula()* para obtener información de *todas las Facturas asociadas al Cliente; y Mostras los Productos vendidos*.

3. ¿Cómo realizo la entrega?

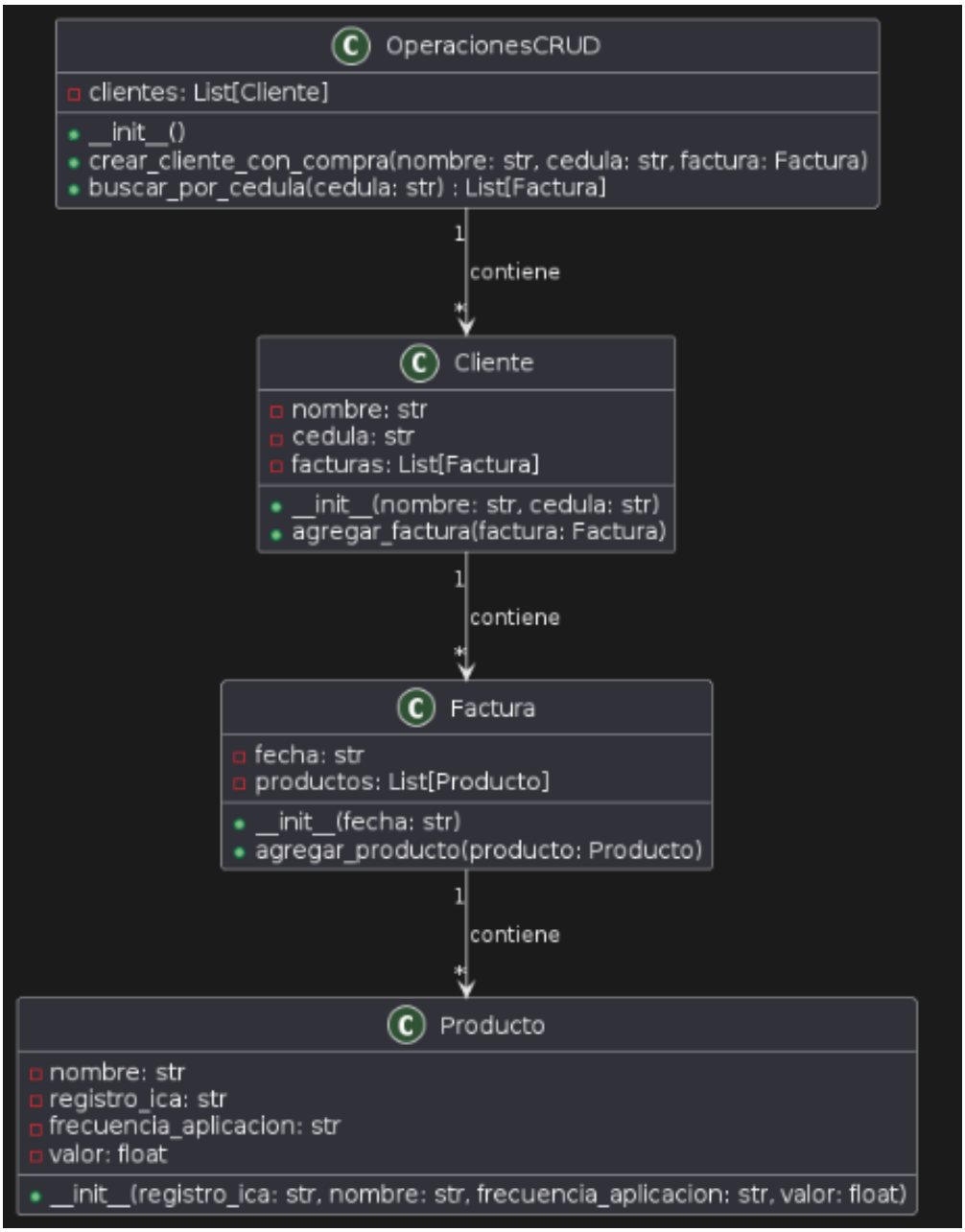
1. Usted debe de entregar el código fuente en su repositorio de *github*.
2. Pantallazos donde se corrobore que las pruebas unitarias han pasado.
3. Pantallazos donde se corrobore el uso del **debug**. En estas imagenes de debe de observar la composición del objeto. Es decir, se envidencia que el objeto x tiene *asociado* n *insntacias* del objeto y. Lo mismo con la herencia.
4. Realizar el Diagrama de Clases y Diagrama de Componentes.

4. Evidencias

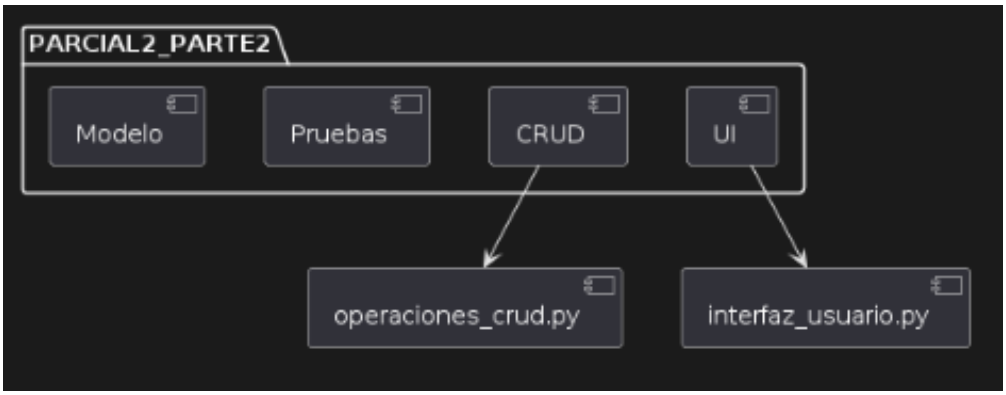
4.1. Repositorio de GitHub

PROGRMA MODELO

4.2. Diagrama de Clases



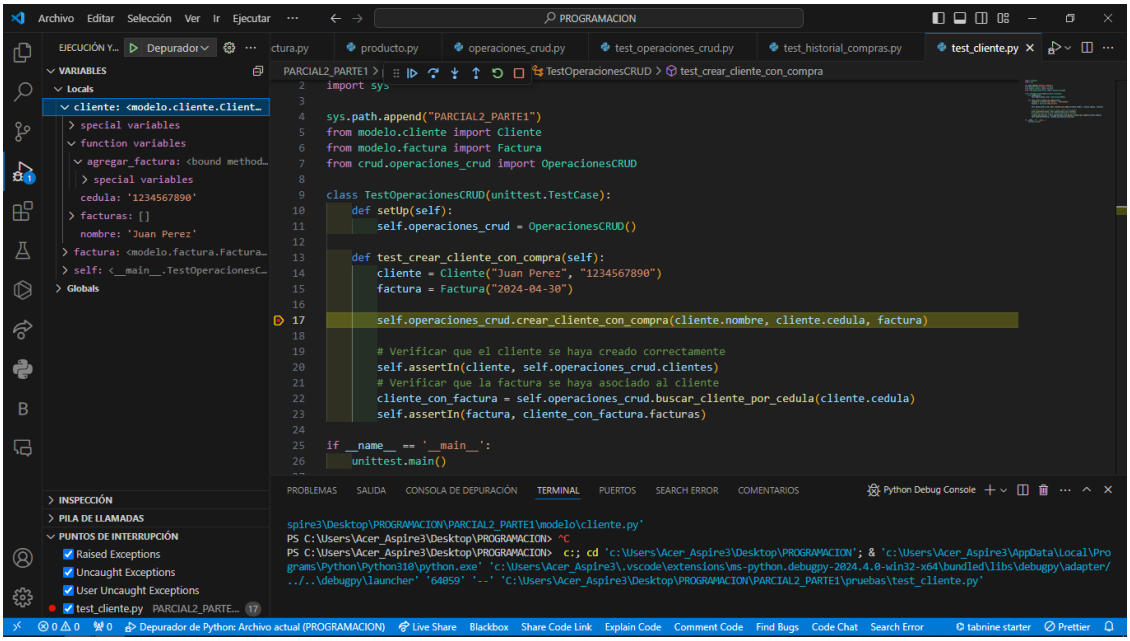
4.3. Diagrama de Componentes

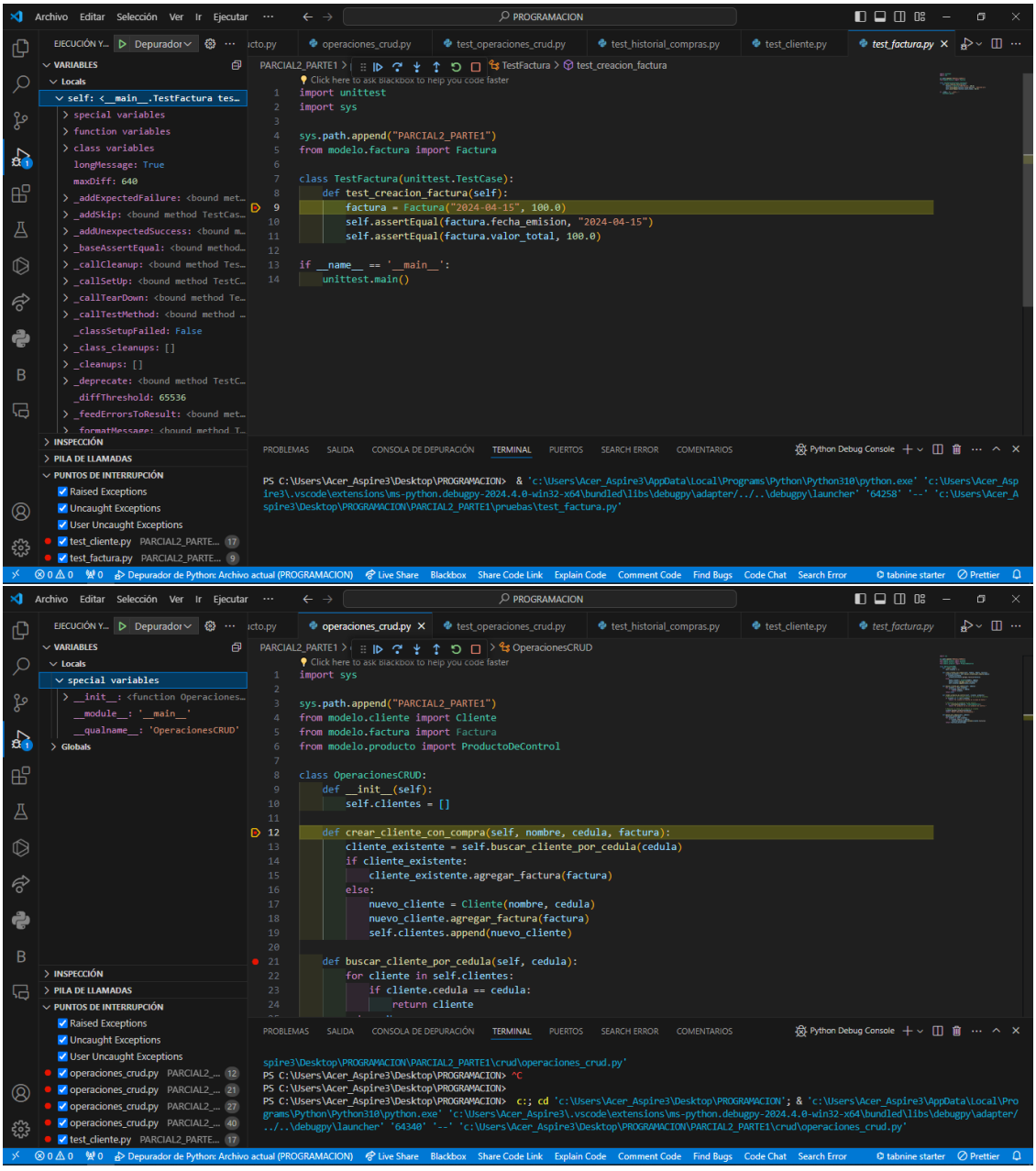


4.4. Pantallazos Pruebas Unitarias

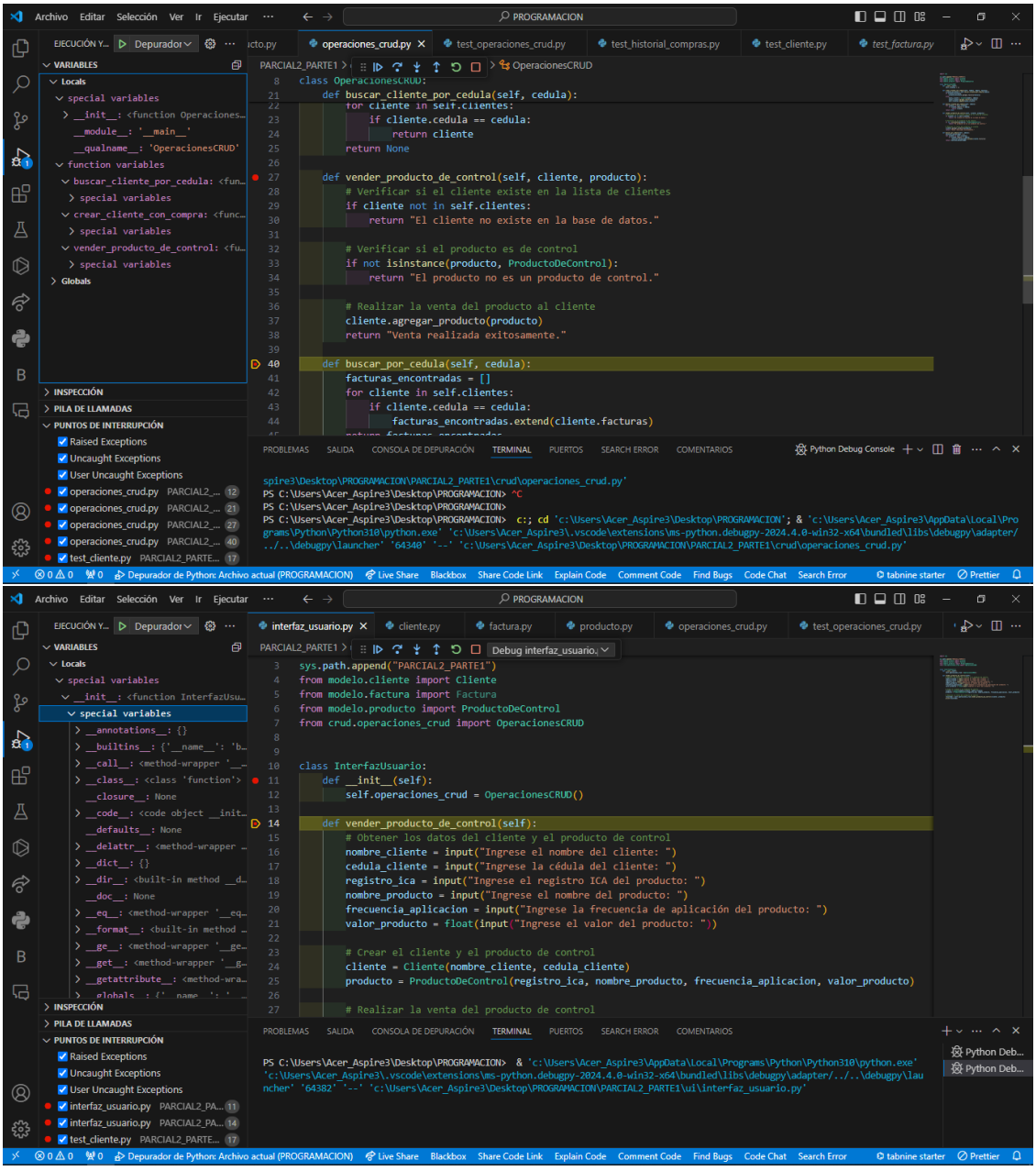
```
[Running] python -u "c:\Users\Acer_Aspire3\Desktop\PROGRAMACION\PARCIAL2_PARTE1\crud\operaciones_crud.py"

[Done] exited with code=0 in 0.251 seconds

[Running] python -u "c:\Users\Acer_Aspire3\Desktop\PROGRAMACION\PARCIAL2_PARTE1\ui\interfaz_usuario.py"

[Done] exited with code=0 in 0.258 seconds
```

4.5. Pantallazos Debug




Archivo Editar Selección Ver Ir Ejecutar
PROGRAMACION
test_factura.py
PARCIAL2_PARTE1 > TestFactura > test_creacion_factura
Click here to ask Blackbox to help you code faster
import unittest
import sys
sys.path.append("PARCIAL2_PARTE1")
from modelo.factura import Factura
class TestFactura(unittest.TestCase):
def test_creacion_factura(self):
factura = Factura("2024-04-15", 100.0)
self.assertEqual(factura.fecha_emision, "2024-04-15")
self.assertEqual(factura.valor_total, 100.0)
if __name__ == '__main__':
unittest.main()
PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN TERMINAL PUERTOS SEARCH ERROR COMENTARIOS
Python Debug Console
PS C:\Users\Acer_Aspire3\Desktop\PROGRAMACION> & 'c:\Users\Acer_Aspire3\AppData\Local\Programs\Python\Python310\python.exe' 'c:\Users\Acer_Aspire3\.vscode\extensions\ms-python.debugpy-2024.4.0-win32-x64\bundled\libs\debugpy\adapter/../..\debugpy\launcher' '64258' '--' 'c:\Users\Acer_Aspire3\Desktop\PROGRAMACION\PARCIAL2_PARTE1\pruebas\test_factura.py'

Archivo Editar Selección Ver Ir Ejecutar
PROGRAMACION
operaciones_crud.py
PARCIAL2_PARTE1 > OperacionesCRUD
Click here to ask Blackbox to help you code faster
import sys
sys.path.append("PARCIAL2_PARTE1")
from modelo.cliente import Cliente
from modelo.factura import Factura
from modelo.producto import ProductoDeControl
class OperacionesCRUD:
def __init__(self):
self.clientes = []
def crear_cliente_con_compra(self, nombre, cedula, factura):
cliente_existente = self.buscar_cliente_por_cedula(cedula)
if cliente_existente:
cliente_existente.agregar_factura(factura)
else:
nuevo_cliente = Cliente(nombre, cedula)
nuevo_cliente.agregar_factura(factura)
self.clientes.append(nuevo_cliente)
def buscar_cliente_por_cedula(self, cedula):
for cliente in self.clientes:
if cliente.cedula == cedula:
return cliente
PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN TERMINAL PUERTOS SEARCH ERROR COMENTARIOS
Python Debug Console
spire3\Desktop\PROGRAMACION\PARCIAL2_PARTE1\crud\operaciones_crud.py'
PS C:\Users\Acer_Aspire3\Desktop\PROGRAMACION> ^C
PS C:\Users\Acer_Aspire3\Desktop\PROGRAMACION>
PS C:\Users\Acer_Aspire3\Desktop\PROGRAMACION> c:; cd 'c:\Users\Acer_Aspire3\Desktop\PROGRAMACION'; & 'c:\Users\Acer_Aspire3\AppData\Local\Programs\Python\Python310\python.exe' 'c:\Users\Acer_Aspire3\.vscode\extensions\ms-python.debugpy-2024.4.0-win32-x64\bundled\libs\debugpy\adapter/../..\debugpy\launcher' '64340' '--' 'c:\Users\Acer_Aspire3\Desktop\PROGRAMACION\PARCIAL2_PARTE1\crud\operaciones_crud.py'


Archivo Editar Selección Ver Ir Ejecutar
PROGRAMACION
operaciones_crud.py
test_operaciones_crud.py
test_historial_compras.py
test_cliente.py
test_factura.py
OperacionesCRUD
class OperacionesCRUD:
def buscar_cliente_por_cedula(self, cedula):
for cliente in self.clientes:
if cliente.cedula == cedula:
return cliente
return None
def vender_producto_de_control(self, cliente, producto):
# Verificar si el cliente existe en la lista de clientes
if cliente not in self.clientes:
return "El cliente no existe en la base de datos."
# Verificar si el producto es de control
if not isinstance(producto, ProductoDeControl):
return "El producto no es un producto de control."
# Realizar la venta del producto al cliente
cliente.agregar_producto(producto)
return "Venta realizada exitosamente."
def buscar_por_cedula(self, cedula):
facturas_encontradas = []
for cliente in self.clientes:
if cliente.cedula == cedula:
facturas_encontradas.extend(cliente.facturas)
VARIABLES
Locals
special variables
function variables
Globals
INSPECCIÓN
PILA DE LLAMADAS
PUNTOS DE INTERRUPCIÓN
Raised Exceptions
Uncaught Exceptions
User Uncaught Exceptions
PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN TERMINAL PUERTOS SEARCH ERROR COMENTARIOS
Python Debug Console
PS C:\Users\Acer_Aspire3\Desktop\PROGRAMACION> ^C
PS C:\Users\Acer_Aspire3\Desktop\PROGRAMACION>
PS C:\Users\Acer_Aspire3\Desktop\PROGRAMACION> c:; cd 'c:\Users\Acer_Aspire3\Desktop\PROGRAMACION'; & 'c:\Users\Acer_Aspire3\AppData\Local\Programs\Python\Python310\python.exe' 'c:\Users\Acer_Aspire3\.vscode\extensions\ms-python.debugpy-2024.4.0-win32-x64\bundled\libs\debugpy\adapter/../..\debugpy\launcher' '64340' '--' 'c:\Users\Acer_Aspire3\Desktop\PROGRAMACION\PARCIAL2_PARTE1\crud\operaciones_crud.py'
Depurador de Python: Archivo actual (PROGRAMACION)
Live Share Blackbox Share Code Link Explain Code Comment Code Find Bugs Code Chat Search Error tabnine starter Prettier

Archivo Editar Selección Ver Ir Ejecutar
PROGRAMACION
interfaz_usuario.py
cliente.py
factura.py
producto.py
operaciones_crud.py
test_operaciones_crud.py
sys.path.append("PARCIAL2_PARTE1")
from modelo.cliente import Cliente
from modelo.factura import Factura
from modelo.producto import ProductoDeControl
from crud.operaciones_crud import OperacionesCRUD
class InterfazUsuario:
def __init__(self):
self.operaciones_crud = OperacionesCRUD()
def vender_producto_de_control(self):
# Obtener los datos del cliente y el producto de control
nombre_cliente = input("Ingrese el nombre del cliente: ")
cedula_cliente = input("Ingrese la cédula del cliente: ")
registro_ica = input("Ingrese el registro ICA del producto: ")
nombre_producto = input("Ingrese el nombre del producto: ")
frecuencia_aplicacion = input("Ingrese la frecuencia de aplicación del producto: ")
valor_producto = float(input("Ingrese el valor del producto: "))
# Crear el cliente y el producto de control
cliente = Cliente(nombre_cliente, cedula_cliente)
producto = ProductoDeControl(registro_ica, nombre_producto, frecuencia_aplicacion, valor_producto)
# Realizar la venta del producto de control
VARIABLES
Locals
special variables
INSPECCIÓN
PILA DE LLAMADAS
PUNTOS DE INTERRUPCIÓN
Raised Exceptions
Uncaught Exceptions
User Uncaught Exceptions
PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN TERMINAL PUERTOS SEARCH ERROR COMENTARIOS
PS C:\Users\Acer_Aspire3\Desktop\PROGRAMACION> & 'c:\Users\Acer_Aspire3\AppData\Local\Programs\Python\Python310\python.exe' 'c:\Users\Acer_Aspire3\.vscode\extensions\ms-python.debugpy-2024.4.0-win32-x64\bundled\libs\debugpy\adapter/../..\debugpy\launcher' '64382' '--' 'c:\Users\Acer_Aspire3\Desktop\PROGRAMACION\PARCIAL2_PARTE1\ui\interfaz_usuario.py'
Depurador de Python: Archivo actual (PROGRAMACION)
Live Share Blackbox Share Code Link Explain Code Comment Code Find Bugs Code Chat Search Error tabnine starter Prettier