HITO DE MEJORA PROGRAMACION

Iván Guijarro Soto

1º TRIMESTRE

Contenido

Hito individual

Fase 1(Teoría y Diseño)

Se requiere desarrollar una aplicación que permita a los posibles clientes registrarse y realizar pedidos de los productos disponibles en la tienda en línea. Los clientes pueden ser tanto nacionales como extranjeros, lo que plantea un desafío adicional en cuanto a la adaptación de los pedidos. La aplicación será programada en Python, aunque se llevará a cabo un análisis previo utilizando pseudocódigo y diagramas de flujo.

Las operaciones que nos pide el cliente son las siguientes:

1. Registro de clientes con datos personales
2. Selección de productos
3. Lista de Deseos
4. Pago de los productos
5. Enviar la factura de la compra en PDF al correo electrónico
6. Un seguimiento mediante un código que se enviara por correo electrónico y SMS

Para poder crear la solución para este tipo de operaciones necesitamos saber que es un algoritmo:

> -Un algoritmo es un conjunto de pasos o instrucciones precisas y ordenadas que se sigue para resolver un problema o realizar una tarea específica con un orden preciso.

Problema→Solución

- **Registro de clientes**
  Entrada: ID, Nombre, Apellidos, Email, Ciudad, Telf.
  Salida: Redirección a otra página donde pueda Iniciar Sesión

- **Selección de Productos**
  Entrada: Nombre_Producto, Precio, Unidades, Fecha_de_Compra
  Salida: Mensaje: Nombre_ Producto añadido con éxito.

- **Lista de Deseos**
  Entrada: Nombre_Producto, Precio, Unidades
  Salida: Mensaje: Nombre_Producto añadido a la Lista de Deseos con éxito

- **Pasarela de Pago**
  Entrada: Nombre_Tarjeta, Numero_Tarjeta, Fecha_Caducidad, CVV
  Salida: Mensaje: Pago Realizado con éxito

- **Enviar Factura en PDF**
  Entrada: Correo electrónico
  Salida: Mensaje: Factura_PDF enviada al correo electrónico

- **Enviar Seguimiento**
  Entrada: Telf, Correo Electrónico
  Salida: Mensaje: Código de seguimiento enviado al correo electrónico y al Telf vía SMS

Fase 2(Método/Proceso)

A continuación, nos enfocaremos en la creación de la aplicación. En esta etapa, es esencial utilizar Python para diseñar las clases, atributos, métodos y constructores necesarios. Es importante tener en cuenta que no se requiere el desarrollo de la interfaz gráfica de usuario, sólo la funcionalidad de cada caso. Utilizaremos programación imperativa, en particular la programación secuencial y la programación orientada a objetos (POO).

Clases

Cliente:

- ID, Nombre, Apellidos, Email, Ciudad, Telf.

Producto

- ID_Producto, Nombre_Producto, Precio, Unidades.

Pedido

- ID_Factura, Fecha Factura, Cliente, Producto.

Clase Clientes:

- ID, Nombre, Apellidos, Email, Ciudad, Telf

Método:

-Info_Cliente: Muestra la información del cliente

Clase Productos:

-ID_Producto, Nombre_Producto, Precio, Unidades,

Método:

-Info_Product: Muestra la información del Producto

Clase Pedido:

-ID_Factura, Fecha_de_Compra, Cliente, Producto

Método:

-Info_Pedido: Muestra la información de la Factura

-Add_Deseos: Se añade a la lista de deseos

-Pago: Realizar pago

-Fact_PDF: Envío de la factura en PDF al correo electrónico

-End_Shop: Finaliza la compra

-Seguimiento: Envía un SMS al cliente para poder realizar el seguimiento

Fase 3(Por qué)

He usado VSCode para realizar el desarrollo de este programa porque es un editor de código fuente que tiene una integración nativa con Python. Esto permite una mayor productividad y facilidad de uso a la hora de trabajar con él, además de tener una amplia gama de extensiones que te pueden ayudar a personalizarlo de la mejor manera para que te sientas bastante cómodo usándolo, como es mi caso.

Teoría

La programación imperativa se enfoca en cómo se deben realizar las tareas, especificando cada paso necesario, mientras que la programación declarativa se enfoca en qué se desea lograr, describiendo los datos y relaciones y dejando que el sistema determine cómo lograrlo. La programación imperativa se basa en modificar el estado de los datos y ejecutar instrucciones secuencialmente, mientras que la programación

declarativa se basa en describir el resultado deseado y dejar que el lenguaje o sistema encuentre la manera de obtenerlo.

Programación Orientada a Objetos (POO) en Python es una forma de programar en la que se utilizan objetos para representar conceptos del mundo real y se utilizan para modelar problemas de manera más precisa y fácil de entender.

-**Encapsulamiento**: Los objetos tienen atributos y métodos, y los atributos pueden ser privados o públicos. Esto permite controlar el acceso a los datos y la lógica de negocio.

-**Herencia**: Es posible crear una clase a partir de otra ya existente, heredando sus atributos y métodos. Esto permite crear jerarquías de clases y reutilizar código.

-**Polimorfismo**: Las clases pueden implementar métodos con el mismo nombre, pero con comportamientos diferentes, dependiendo de la clase. Esto permite tratar objetos de diferentes clases de manera similar.

```python
class Cliente:
    def __init__(self,id,nombre,apellidos,ciudad,email,telf):
        self.id=id
        self.nombre=nombre
        self.apellidos=apellidos
        self.ciudad=ciudad
        self.email=email
        self.telf=telf
    def Info_Cliente(self):
        print(f'El cliente se llama {self.nombre} {self.apellidos} , su email es: {self.email} y su telf es: {self.telf} y reside en:{self.ciudad}')
```

Primero he definido es una clase de Python para representar a un cliente. El método __init__ es un constructor de la clase, y se utiliza para inicializar los atributos de una instancia de la clase.

En este caso, el constructor recibe 6 parámetros: id, nombre, apellidos, ciudad, email, y telf. Cada uno de estos parámetros se asigna a un atributo de la clase. Por ejemplo, el parámetro id se asigna al atributo id de la clase utilizando self.id = id.

Puedes instanciar un objeto de esta clase y asignarle valores a los atributos utilizando los parámetros del constructor, como se muestra a continuación:

cliente1 = Cliente (1, "Juan", "Pérez","Madrid","juan.perez@gmail.com", "+34 123 456 789")

De igual manera, puedes acceder a los atributos del objeto:

print (cliente1.nombre)

La función Info_Cliente es un método de la clase Cliente, que tiene acceso a los atributos de la instancia mediante el uso de self.

En este caso, el método imprime información sobre el cliente utilizando una cadena de formato y los atributos de la instancia: nombre, apellidos, email, telf y ciudad.

Para llamar a este metodo debes crear una instancia de la clase Cliente y despues llamar al método

El cliente se llama Juan Pérez, su email es: juan.perez@gmail.com y su telf es: +34 123 456 789 y reside en: Madrid

```python
class Producto:
    def __init__(self,id_producto,nombre_producto,unidades,precio):
        self.id_producto=id_producto
        self.nombre_producto=nombre_producto
        self.unidades=unidades
        self.precio=precio
    def Info_Product(self):
        print(f'Los datos del producto son {self.nombre_producto} y cuesta: {self.precio}€')
```

Esta clase que he creado es igual que la anterior lo único que cambia son los parámetros y los atributos que se le asignan a cada parámetro.

La función Info_Product es un método de la clase Producto, que tiene acceso a los atributos de la instancia mediante el uso de self.

En este caso, el método imprime información sobre el producto utilizando una cadena de formato y los atributos de la instancia: nombre del producto y el precio del mismo.