Instituto de Educación Superior Privado TECSUP

DEPARTAMENTO DE TECNOLOGÍA DIGITAL

Carrera de Diseño y desarrollo de software

C24 - III - B

Sistema de gestión de pedidos de una tienda ropa online

Profesor:

Camasca Macedo, Charles Dummar

Integrantes:

Castro Peñaloza Hector Hanmer

Dávila Perez, Alessandro Alberto

Huaytalla Rodriguez, Franklin Alvaro

Perez Vengoa, Hector Henrique

Rodriguez Ordoñez, Juan Daniel

Lima – Perú

2024 - II

ÍNDICE

INTRODUCCIÓN **3**
1. Problemática 4
Planteamiento del problema 4
Objetivo general. 4
Objetivos específicos. 4
Justificación 4
Justificación técnica. 4
Justificación financiera 5
2. Metodología 7
Marco teórico 7
Metodología de programación 7
Diagrama de casos 8
3. Desarrollo 9
Diseño 9
Codificación 9
Resultados 14
4. Conclusiones 17
5. Recomendaciones 18
6. Bibliografía 19
7. Anexos 20

INTRODUCCIÓN

En este informe se presentará, en primer lugar, el planteamiento de la problemática que enfrentan las empresas hoy en día, señalando los principales retos que afectan su funcionamiento y crecimiento. Luego, se explicarán los objetivos que se buscan lograr con este proyecto, así como la razón por la que es importante y necesario llevarlo a cabo.

A continuación, se describirán de forma clara la metodología que se seguirá para desarrollar el proyecto, detallando las fases y los pasos que se tomarán para cumplir con los objetivos planteados.

Además, se discutirá cómo este proyecto puede beneficiar a las empresas y mejorar sus procesos en la gestión de pedidos.

También se presenta el diseño de la interfaz, que facilitará una visualización clara y ordenada de los datos. Además, se incluirá el código utilizado y los resultados generados, para que puedan ver cómo queda la interfaz de la aplicación en funcionamiento.

1. Problemática

a. Planteamiento del problema

Las nuevas empresas y las ya establecidas enfrentan desafíos importantes, como la pérdida de pedidos, retrasos en los tiempos de entrega y la falta de control del estado en el que se encuentra un pedido.

Objetivos

- Objetivo general: Desarrollar un sistema de gestión de pedidos y envíos para una tienda online de ropa.
- Objetivos específicos:
    1. Analizar los requerimientos técnicos en un sistema de gestión de pedidos.
    2. Justificar la viabilidad financiera del sistema mediante un análisis costo beneficio.
    3. Examinar las metodologías y estructuras de datos adecuadas para desarrollar el sistema.

b. Justificación

- Justificación técnica:

  El sistema de gestión de pedidos se desarrollará utilizando Python 3.10 o superior y la biblioteca tkinter para la interfaz gráfica. A continuación se detallan los requisitos mínimos y recomendables para su correcto funcionamiento:

  Requisitos mínimos:

  Procesador: 1.6 GHz o superior.

  Memoria RAM: 2 GB.

  Espacio en disco: 1 GB.

  Sistema operativo: Windows 7 o superior.

  Python: Python 3.10 o superior.

  Requisitos recomendables:

  Procesador: 2.0 GHz o superior, idealmente con 4 núcleos.

Memoria RAM: 4 GB o más.

Espacio en disco: 2 GB o más.

Sistema operativo: Windows 10 o superior.

Python: última versión estable.

Los requisitos mínimos aseguran que el sistema sea accesible para la mayoría de los equipos actuales; esto permite que la tienda comience a usar el sistema sin necesidad de tener equipos con alto rendimiento. Estos requisitos permitirán que el sistema realice las funciones básicas de manera eficiente.

Sin embargo, los requisitos recomendables ayudan a mejorar la experiencia del usuario. Tener un procesador más rápido, más memoria RAM y un disco más grande hace que el sistema funcione mucho mejor, sin interrupciones ni tiempos de espera innecesarios. Si la tienda tiene un alto volumen de pedidos, es necesario contar con un equipo más potente que permitirá que el sistema crezca de manera eficiente sin que se vuelva lento o inestable.

- Justificación financiera:

  Este proyecto es una buena oportunidad financiera porque en cada mes los ingresos son mayores que los gastos, lo que asegura que se genere dinero en lugar de perderlo. En un año, se espera ganar S/ 683,000, mientras que los gastos sumarán S/ 255,158. Esto significa que quedarán S/ 427,842 como ganancia, lo que representa un crecimiento del 159.64%. Estos resultados muestran que el proyecto es rentable y podrá cubrir los gastos necesarios, además de dejar una buena cantidad extra, haciendo una inversión conveniente para la organización.

  Si se proyecta este rendimiento óptimo a cinco años, considerando un progreso constante y estable , generará aproximadamente S/ 3415,000 en costos de S/ 1,296,905 dejando un flujo acumulado de S/ 2,120,095.

Tabla 1

Ingresos y egresos promedio.

	Octubre	Noviembre	Diciembre	Enero	Febrero	Marzo	Abril	Mayo	Junio	Julio	Agosto	Septiembre
Ingresos	50000	45000	70000	65000	62000	57000	47000	63000	53000	54000	51000	66000
						Egresos						
Proyecto	5000			5000			5000			5000		
Costo fijo	10000	10000	10000	10000	10000	10000	10000	10000	10000	10000	10000	10000
Costo variable	10505	10328	8962	8446	9989	9499	10582	10832	8674	10477	10977	9710
Suma Costo	25505	20328	18962	23446	19989	19499	25582	20832	18674	25477	20977	19710

Nota. Elaboración propia. Se pueden identificar los ingresos y egresos promedio con respecto a los 12 meses de la empresa.

Tabla 2

Cálculo del ingreso total y suma de costos

Ingreso total	683000
Suma de costos	258981

Nota. Elaboración propia.

Tabla 3

Cálculo de flujo de caja anual

Flujo de caja anual	
424019	161,08%

Nota. Elaboración propia.

2. Metodología

a. Marco teórico

Listas enlazadas: Las listas enlazadas son estructuras de datos fundamentales en la programación que consisten en una secuencia de elementos llamados nodos, los cuales se componen de dos partes: ‘Datos’ que almacenan la información en el nodo y ‘Referencia’, el cual sirve como puntero al siguiente nodo, como también permiten añadir, eliminar o buscar elementos, funciones esenciales en aplicaciones de gestión.

Colas: Son estructuras de datos que siguen la política FIFO (First In, First Out), lo que significa que el primer elemento en ingresar en la cola es también el primero en salir. Una cola tiene dos operaciones principales: ‘Encolar’ la cual añade un elemento al final de la cola, y ‘Desencolar’ que remueve el elemento final, y donde este elemento se puede mover direccionando a una lista.La utilización de las colas nos permite gestionar los envíos en el mismo orden en el cual llegaron, el mantener el orden de entrada garantiza que los pedidos se procesan en el orden adecuado, evitando que un pedido reciente se envíe antes que uno ingresado anteriormente.

Árboles binarios de búsqueda: Un árbol binario de búsqueda es una estructura de datos donde cada nodo contiene un elemento y dos referencias, los nodos en el subárbol izquierdo contienen valores menores que el nodo raiz, mientras que los derechos contienen valores mayores.Proporciona un control en las entregas de los pedidos gracias a que estos árboles ordenan los datos mediante un criterio, en este caso por el orden de llegada de cada pedido.

b. Metodología de programación

1. Fase de análisis: Identificación de las necesidades del sistema, análisis de los flujos de pedidos y envíos.
2. Diseño del sistema: Definición de las estructuras de datos que se usarán para cada parte del sistema.

3. Implementación: Desarrollo del sistema usando las estructuras de datos definidas, seguido de pruebas unitarias para asegurar el correcto funcionamiento.
4. Pruebas y evaluación: Verificación del comportamiento del sistema bajo diferentes escenarios de carga.

c. Diagrama de casos

Figura 1

Diagrama de casos de uso

Nota. Elaboración propia.

3. Desarrollo
    a. Diseño

    Este es el diseño que se mostrará al realizar diferentes operaciones, como agregar, eliminar y editar. Además, se está incorporando una función de búsqueda y una tabla para visualizar los datos que se guardarán.

    b. Codificación

    Estructura del código de listas, árboles, nodos y colas

```
class Cliente:
    def __init__(self, nombre, apellido, correo, telefono,
direccion, dni, id=None):
        self.id = id
        self.dni = dni
        self.nombre = nombre
        self.apellido = apellido
        self.correo = correo
        self.telefono = telefono
        self.direccion = direccion

class Nodo:
    def __init__(self, dato):
        self.dato = dato
        self.siguiente = None

class ListaEnlazada:
    def __init__(self):
        self.cabeza = None

    def agregar(self, cliente):
        nuevo_nodo = Nodo(cliente)
        if not self.cabeza:
            self.cabeza = nuevo_nodo
        else:
            actual = self.cabeza
            while actual.siguiente:
```

```
                actual = actual.siguiente
            actual.siguiente = nuevo_nodo

    def buscar(self, dni):
        actual = self.cabeza
        while actual:
            if actual.dato.dni == dni:
                return actual.dato
            actual = actual.siguiente
        return None

    def obtener_todos(self):
        clientes = []
        actual = self.cabeza
        while actual:
            clientes.append(actual.dato)
            actual = actual.siguiente
        return clientes)


class Pedido:
    def __init__(self, id, dni, nombres, correo, producto,
talla, cantidad, precio, total, direccion, fecha, estado):
        self.id = id
        self.dni = dni
        self.nombres = nombres
        self.correo = correo
        self.producto = producto
        self.talla = talla
        self.cantidad = cantidad
        self.precio = precio
        self.total = total
        self.direccion = direccion
        self.fecha = fecha
        self.estado = estado

class ColaPedidos:
```

```
    def __init__(self):
        self.frente = None
        self.final = None

    def encolar(self, pedido):
        nuevo_nodo = NodoPedido(pedido)
        if not self.frente:
            self.frente = nuevo_nodo
            self.final = nuevo_nodo
        else:
            self.final.siguiente = nuevo_nodo
            self.final = nuevo_nodo

    def desencolar(self):
        if not self.frente:
            return None
        pedido = self.frente.pedido
        self.frente = self.frente.siguiente
        if not self.frente:
            self.final = None
        return pedido

class ArbolBinario:
    def __init__(self):
        self.raiz = None

    def insertar(self, pedido):
        if not self.raiz:
            self.raiz = NodoArbol(pedido)
        else:
            self._insertar_recursivo(self.raiz, pedido)

    def _insertar_recursivo(self, nodo, pedido):
        if pedido.id < nodo.pedido.id:
            if nodo.izquierda:
```

```
                self._insertar_recursivo(nodo.izquierda,
pedido)
            else:
                nodo.izquierda = NodoArbol(pedido)
        else:
            if nodo.derecha:
                self._insertar_recursivo(nodo.derecha,
pedido)
            else:
                nodo.derecha = NodoArbol(pedido)

    def buscar(self, id_pedido):
        return self._buscar_recursivo(self.raiz, id_pedido)

    def _buscar_recursivo(self, nodo, id_pedido):
        if not nodo:
            return None
        if nodo.pedido.id == id_pedido:
            return nodo.pedido
        elif id_pedido < nodo.pedido.id:
            return self._buscar_recursivo(nodo.izquierda,
id_pedido)
        else:
            return self._buscar_recursivo(nodo.derecha,
id_pedido)
```

Creación de NODOS:

```
class NodoPedido:
    def __init__(self, pedido):
        self.pedido = pedido
        self.siguiente = None
```

```
class NodoArbol:
    def __init__(self, pedido):
        self.pedido = pedido
```

```
        self.izquierda = None
        self.derecha = None
```

```
class NodoTalla:
    def __init__(self, talla, cantidad):
        self.talla = talla
        self.cantidad = cantidad
        self.siguiente = None
```

Creación del objeto Productos:

```


# Clase ListaTallas: Representa una lista enlazada de tallas con cantidades
class ListaTallas:
    def __init__(self):
        self.cabeza = None

    def agregar_talla(self, talla, cantidad):
        nuevo_nodo = NodoTalla(talla, cantidad)
        if not self.cabeza:
            self.cabeza = nuevo_nodo
        else:
            actual = self.cabeza
            while actual.siguiente:
                actual = actual.siguiente
            actual.siguiente = nuevo_nodo

    def obtener_tallas(self):
        tallas = []
        actual = self.cabeza
        while actual:
            tallas.append((actual.talla, actual.cantidad))
            actual = actual.siguiente
        return tallas
```

```
# Clase Producto: Representa un producto asociado a una
lista de tallas
class Producto:
    def __init__(self, nombre, prenda, marca,
precio_unitario):
        self.nombre = nombre
        self.prenda = prenda
        self.marca = marca
        self.precio_unitario = precio_unitario
        self.tallas = ListaTallas()
```

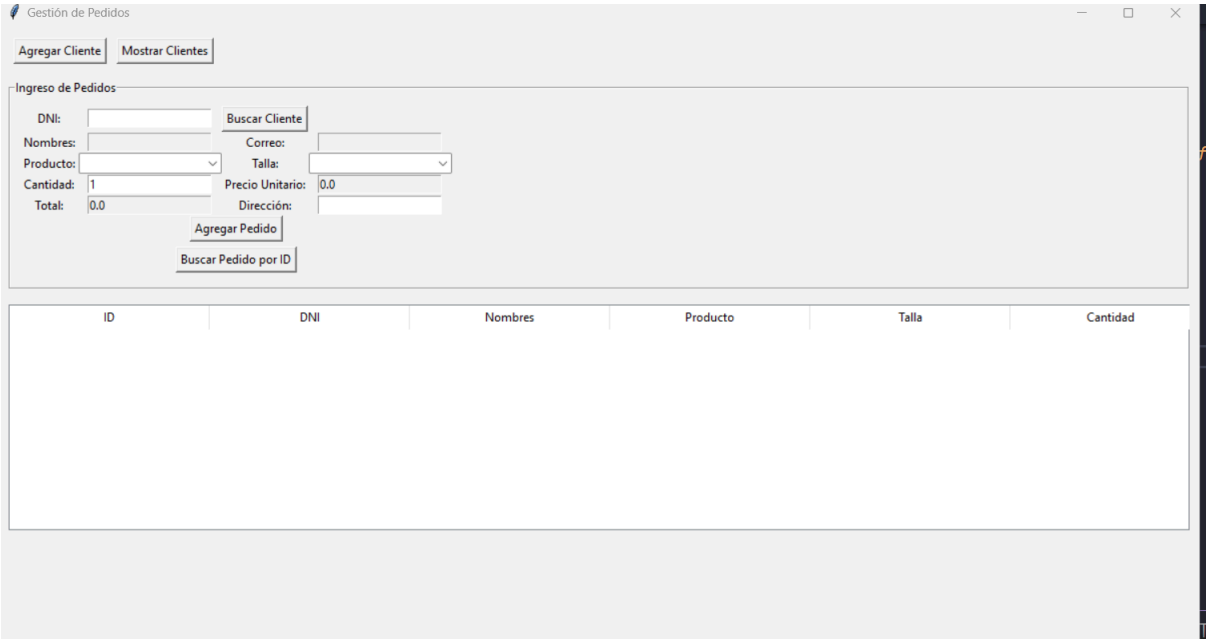
Creación del objeto Pedido:

```
class Pedido:
    def __init__(self, id, dni, nombres, correo, producto,
talla, cantidad, precio, total, direccion, fecha, estado):
        self.id = id
        self.dni = dni
        self.nombres = nombres
        self.correo = correo
        self.producto = producto
        self.talla = talla
        self.cantidad = cantidad
        self.precio = precio
        self.total = total
        self.direccion = direccion
        self.fecha = fecha
        self.estado = estado
```

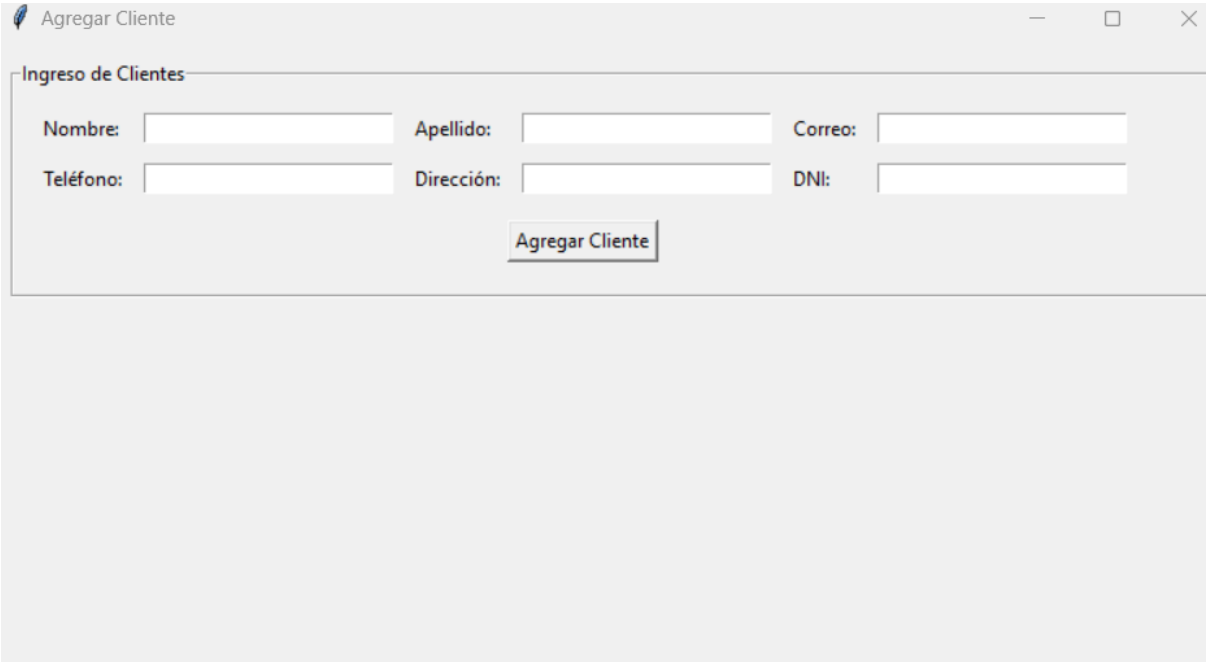
En esta parte se puede visualizar el diseño del sistema en Tkinder y cómo se van a ingresar los datos, aparte de las diferentes opciones que va a presentar este proyecto.

- **Ventana principal.**

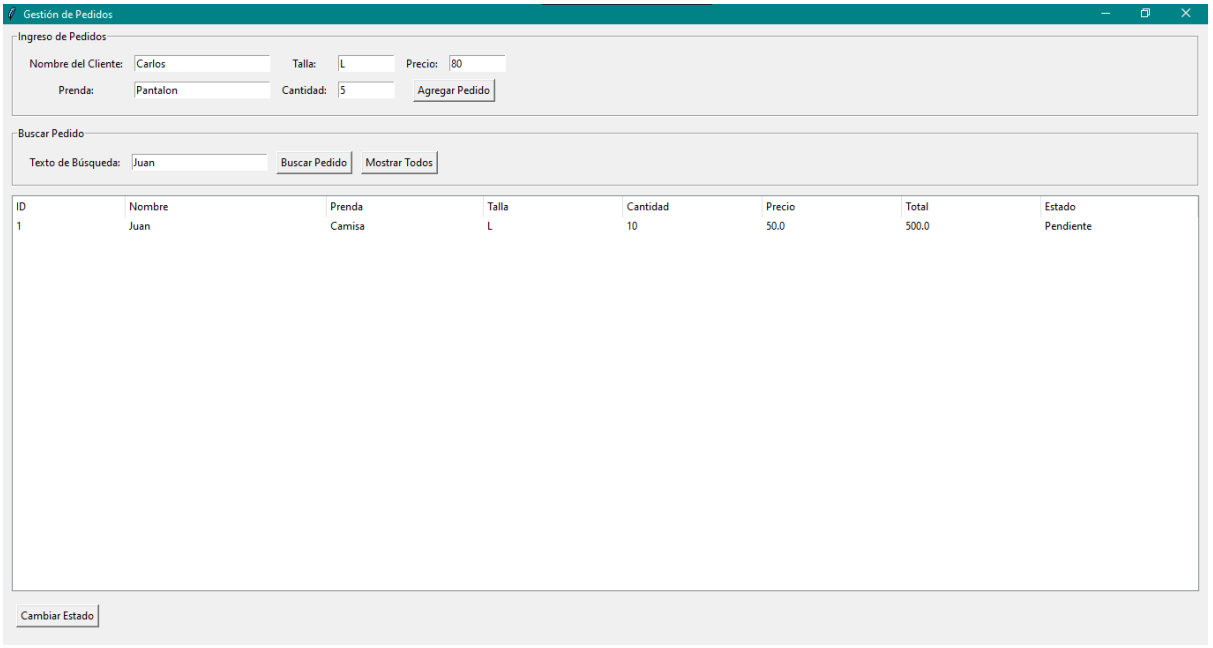
Figura.



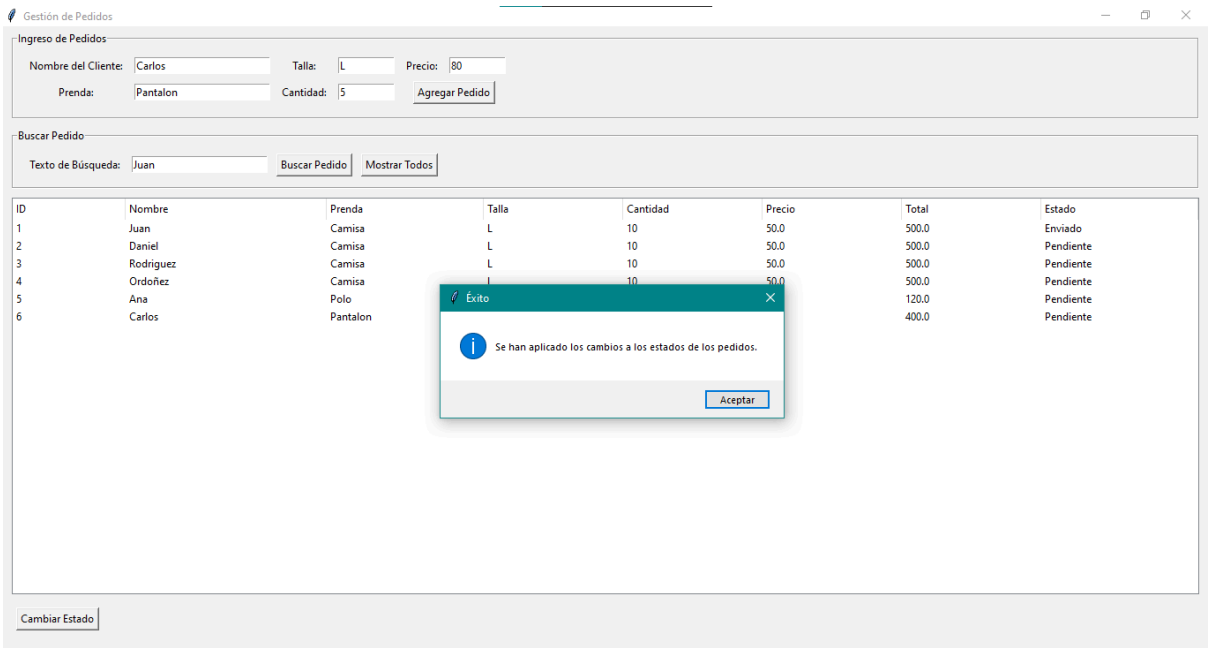
- **Agregando clientes**



- **Búsqueda de pedido por nombres**



- **Cambio de estado del pedido**



4. Conclusiones

Se logró desarrollar con éxito un sistema de gestión de pedidos y envíos para una tienda online de ropa, el cual cumple con las funcionalidades necesarias para su funcionamiento.

Se realizó el análisis de los requerimientos del sistema de gestión de pedidos, abarcando tanto los aspectos relacionados con el software. Durante este proceso, se establecieron los requisitos mínimos necesarios para el correcto funcionamiento del programa.

Se realizó una justificación financiera para evaluar la rentabilidad del programa en un plazo de 5 años, utilizando el flujo de caja anual, el cual muestra los ingresos y egresos del proyecto. Esto permitió evaluar la capacidad para generar efectivo.

Se utilizaron diferentes metodologías en el desarrollo del proyecto. En primer lugar, se elaboró el marco teórico, donde se establecieron los temas que servirían de base para su realización. Además, se empleó la metodología de programación, que guió el proceso de desarrollo del software de manera estructurada y eficiente. Por último, se utilizó el diagrama de casos de uso, lo que permitió visualizar la interacción del usuario con el programa.

5. Recomendaciones

Se recomienda realizar monitoreos continuos al sistema, garantizando el correcto rendimiento del sistema y sus funcionalidades, identificando problemas y errores técnicos que puedan surgir por su operación.

Realizar pruebas constantes del programa para verificar su correcto funcionamiento, asegurando que cumpla con los requisitos establecidos y satisfaga las expectativas de los usuarios.

Verificar que la ganancia generada por el programa sea la esperada. En caso contrario, se recomendaría explorar formas de optimizar el rendimiento del programa para asegurar una ganancia óptima acorde a las expectativas.

En caso de que las metodologías planteadas no sean las más adecuadas, se recomienda explorar otras metodologías que puedan facilitar el análisis del programa, enfocándose en el cumplimiento de los objetivos establecidos y asegurando la satisfacción del cliente.

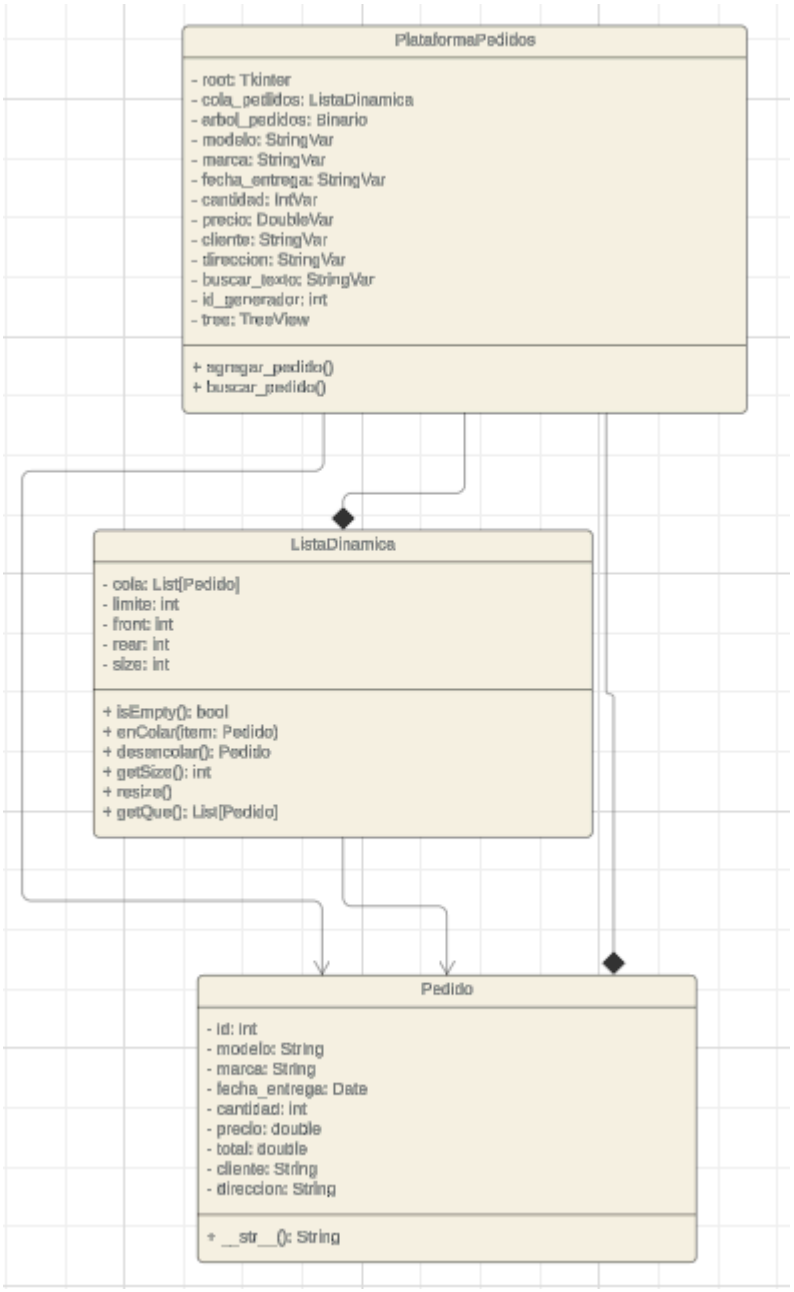
6. Bibliografía

Informatec Digital. (s.f.). *Requisitos del software: Cómo definirlos correctamente*. https://informatecdigital.com/desarrollo/requisitos-del-software/

Informatec Digital. (s.f.). *Las etapas del desarrollo de software: una guía completa*. https://informatecdigital.com/desarrollo/las-etapas-del-desarrollo-de-software-una-guia-completa/

VidaSoft. (s.f.). *Ciclo de vida del desarrollo de software: fases, ejemplos, modelos y mejores prácticas*. https://vidasoft.es/blog/desarrollo-producto/ciclo-de-vida-del-desarrollo-de-software-fases-ejemplos-modelos-y-mejores-practicas/

7. Anexos


PlataformaPedidos
- root: Tkinter
- cola_pedidos: ListaDinamica
- arbol_pedidos: Binario
- modelo: StringVar
- marca: StringVar
- fecha_entrega: StringVar
- cantidad: IntVar
- precio: DoubleVar
- cliente: StringVar
- direccion: StringVar
- buscar_texto: StringVar
- id_generador: int
- tree: TreeView
+ agregar_pedido()
+ buscar_pedido()
ListaDinamica
- cola: List[Pedido]
- limite: int
- front: int
- rear: int
- size: int
+ isEmpty(): bool
+ enColar(item: Pedido)
+ desencolar(): Pedido
+ getSize(): int
+ resize()
+ getQue(): List[Pedido]
Pedido
- id: int
- modelo: String
- marca: String
- fecha_entrega: Date
- cantidad: int
- precio: double
- total: double
- cliente: String
- direccion: String
+ __str__(): String