Universidad de San Carlos de Guatemala
Escuela de Ingeniería en Ciencias y Sistemas, Facultad de Ingeniería
Introducción a la programación y computación 2, Vacaciones de junio 2024

PROYECTO 1 IPC2 - IPCMARKET

202300434 – Cerezo Paredes, Fátima Florisel

202300512 – Hernandez Flores, Daniel Andreé

Resumen

El ensayo presenta la explicación del desarrollo de IPCmarket, una aplicación de escritorio que permite a los usuarios hacer compras diarias con una interfaz gráfica fácil de usar. Los administradores y los compradores son los dos tipos de usuarios que distingue esta aplicación. Utilizando estructuras de datos avanzadas como listas doblemente enlazadas y colas, los administradores pueden cargar información masiva a través de archivos XML y administrar productos y usuarios. Los consumidores pueden ver reportes gráficos mientras navegan por los productos, los agregan a un carrito de compras y realizan sus compras.

Es importante considerar que el proyecto requiere del uso de programación orientada a objetos (POO) en Python, manejar estructuras de datos dinámicas y crear visualizaciones con Graphviz para llevar a cabo el proyecto. Se requiere autenticación de usuario y el administrador tiene credenciales predeterminadas.

Palabras clave
XML, POO, Python, TDA, Interfaz.

Abstract

The essay presents the explanation of the development of IPCmarket, a desktop application that allows users to make daily purchases with an easy-to-use graphical interface. Administrators and buyers are the two types of users the application distinguish. Using advanced data structures such as double-linked lists and queues, administrators can upload massive information through XML files and manage products and users. Consumers can view graphical reports as they browse products, add them to a shopping cart and make purchases.

It is important to consider that the project requires the use of object-oriented programming (POO) in Python, managing dynamic data structures and creating visualizations with Graphviz to carry out the project. User authentication is required and the administrator has default credentials.

Keywords
XML, POO, Python, TDA, Interface.

Introducción

IPCmarket fue una aplicación de escritorio diseñada para que los usuarios pudieran hacer compras diarias con una interfaz gráfica fácil de entender. La aplicación permite la carga de configuraciones a través de archivos XML, lo que permite la gestión de información de usuarios y productos en listas dinámicas. Los administradores administran la carga de datos y la aceptación de compras, mientras que los compradores seleccionan productos y compran. Los administradores pueden cargar archivos XML con datos de usuarios, productos, empleados y actividades y almacenarlos en estructuras de datos como listas circulares y doblemente enlazadas. Por el contrario, los clientes pueden navegar por los productos, agregarlos a su carrito de compras y finalmente confirmar sus pedidos.

Desarrollo del tema

La Programación Orientada a Objetos (POO) es fundamental para el desarrollo de aplicaciones modernas porque puede modelar elementos del mundo real de manera intuitiva. La OOP de IPCmarket facilita la reutilización y la modularidad del código, lo que es esencial para gestionar el complejo sistema de compra de productos. Cada entidad del dominio, como los usuarios, los productos y las compras, se representa como un objeto con estados y comportamientos particulares. Al mejorar la legibilidad del código y reducir la probabilidad de errores, esto mejora la escalabilidad y el mantenimiento del sistema.

IPCmarket maneja datos utilizando estructuras de datos abstractas (TDA) como listas enlazadas, pilas y colas. Por ejemplo, las listas enlazadas permiten la gestión dinámica de productos y usuarios, permitiendo la inserción y eliminación de productos sin los costos fijos de redimensionamiento asociados con las listas enlazadas. Las pilas facilitan el manejo del carrito de compras, permitiendo agregar y eliminar productos de manera simple y eficiente. Las colas son esenciales para garantizar un procesamiento ordenado y secuencial de las solicitudes de compra.

La interfaz de usuario creada por PyQt5 sirve como puente entre el usuario y la compleja lógica comercial. Los slots y la señalización de PyQt5 facilitan la comunicación entre la lógica comercial básica y la interfaz gráfica, lo que permite a los usuarios responder de inmediato a las interacciones. Por ejemplo, cuando un cliente elige un producto para agregar al carrito de compras, se emite una señal que activa el slot correspondiente en el código del negocio. Este slot controla cómo agregar el producto al carrito de compras. Esta distinción clara entre la lógica y la interfaz mejora la organización del código y facilita la depuración.

La gestión eficiente de operaciones concurrentes, como la actualización de inventarios y la gestión de múltiples sesiones de usuarios, fue uno de los principales desafíos enfrentados durante el desarrollo de IPCmarket. Para resolver esto, las estructuras de datos incorporaron mecanismos de bloqueo y control de concurrencia para garantizar que se llevaran a cabo operaciones críticas de manera atómica. Además, se implementaron servicios web que permitieron a IPCmarket administrar un mayor número de usuarios y transacciones sin disminuir el rendimiento, lo que mejoró la escalabilidad del sistema.

El programa podrá ser totalmente funcional siempre y cuando los archivos XML contengan la siguiente estructura:

a) Usuarios:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<usuarios>
    <usuario id="1" password="12345">
        <nombre>Juan Perez</nombre>
        <edad>30</edad>
        <email>juan.perez@example.com</email>
        <telefono>12345678</telefono>
    </usuario>
```

Figura 1. Estructura del archivo XML de entrada.

Fuente: elaboración propia, 2024.

Donde:

“usuarios” es la etiqueta padre.

“id” es la etiqueta que se proporciona el dato del identificador único del usuario. El cual se usa para configurar su acceso al módulo de usuarios “password” es la etiqueta que nos proporciona la contraseña para acceder al módulo de usuarios.

“nombre” es la etiqueta que indica el nombre del usuario.

“edad” es la etiqueta que indica la edad del usuario.

“email” es la etiqueta que nos proporciona el correo electrónico del usuario.

“telefono” es la etiqueta que nos proporciona el número telefónico del usuario.

b) Empleados:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<empleados>
    <empleado codigo="001">
        <nombre>Ana Garcia</nombre>
        <puesto>Gerente</puesto>
    </empleado>
```

Figura 2. Estructura del archivo XML de entrada.

Fuente: elaboración propia, 2024.

Donde:

“empleados” es la etiqueta padre.

“empleado” es la etiqueta que nos indicar que el

“codigo” es la etiqueta que nos proporciona el número único que identifica a cada empleado.

“nombre” es la etiqueta que nos proporciona el nombre del empleado.

“puesto” es la etiqueta que nos proporciona el puesto del empleado.

c) Productos

```xml
<productos>
    <producto id="3">
        <nombre>Laptop DEF</nombre>
        <precio>750.00</precio>
        <descripcion>Laptop DEF de alta gama</descripcion>
        <categoria>Electrónica</categoria>
        <cantidad>8</cantidad>
        <imagen>imagen3.jpg</imagen>
    </producto>
```

Figura 3. Estructura del archivo XML de entrada.

Fuente: elaboración propia, 2024.

Donde:

“productos” es la etiqueta padre.

“id” es la etiqueta que nos proporciona el identificador único del producto.

“producto” es la etiqueta que nos indica que hay productos dentro.

“nombre” es la etiqueta que nos brinda el nombre del producto.

“precio” es la etiqueta que nos brinda el precio del producto.

“descripción” es la etiqueta que nos indica la descripción del producto.

“categoria” es la etiqueta que nos indica a que categoría se asocia el producto.

“cantidad” es la etiqueta que nos indica a cuantas unidades del producto hay en existencia en el sistema.

"imagen" es la etiqueta que brinda la ruta a la imagen del producto.

d) Actividades:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<actividades>
    <actividad id="101">
        <nombre>Reunión de Ventas</nombre>
        <descripcion>Reunión mensual del equipo de ventas</descripcion>
        <empleado>001</empleado>
        <dia hora="7">1</dia>
    </actividad>
```

Figura 4. Estructura del archivo XML de entrada.

Fuente: elaboración propia, 2024.

Donde:

"actividades" es la etiqueta padre.

"actividad" es la etiqueta que nos indica que existen actividades en el documento.

"id" es la etiqueta que nos indica el identificador único de la actividad.

"nombre" es la etiqueta que nos brinda el nombre de la actividad.

"descripcion" es la etiqueta que nos brinda la descripción de la actividad.

"empleado" es la etiqueta que nos brinda a que empleado le pertenece la actividad. Esta se enlaza automáticamente con el empleado.

"hora" es la etiqueta que nos brinda la hora a la que se realiza la actividad.

"día" proporciona un numero del 1 al 7 que nos indica que día de la semana se lleva a cabo la actividad.

A continuación, se presenta una breve descripción de los métodos más importantes del programa.

Interfaz Gráfica:

Funcionalidad: Este método configura la interfaz gráfica principal de la aplicación utilizando PyQt5. Establece los widgets, botones, y otros componentes visuales, además de definir su comportamiento y estilo.

Importancia: Es crucial para definir la apariencia y la estructura inicial del programa, asegurando que los usuarios interactúen de manera intuitiva con la aplicación.

Mostrar actividades:

Funcionalidad: Filtra y muestra las actividades diarias de los empleados basadas en el día actual. Utiliza las listas enlazadas para acceder a la información relevante y despliega los detalles en la interfaz.

Importancia: Este método es vital para la gestión de recursos humanos, permitiendo a los administradores ver las tareas asignadas y su cumplimiento en tiempo real.

Aceptar o rechazar compras:

Funcionalidad: Estos métodos gestionan las compras en la cola. aceptar_compra añade la compra a una lista de compras aceptadas y rechazar_compra elimina la compra de la cola.

Importancia: Son esenciales para el control de inventario y flujo de caja, permitiendo a los administradores aprobar o declinar transacciones basadas en criterios establecidos.

Actualizar:

Funcionalidad: Actualiza la visualización de compras en la interfaz gráfica, mostrando la información más reciente de las transacciones en espera.

Importancia: Fundamental para mantener a los usuarios informados sobre el estado actual de sus compras y para la administración efectiva de la cola de compras.

Cargas Masivas:

Funcionalidad: Permite la carga de datos desde archivos XML para diferentes tipos de entidades como usuarios, productos, empleados y actividades.

Importancia: Este método es crucial para la inicialización y actualización de la base de datos del sistema, facilitando la integración y el mantenimiento de grandes volúmenes de información.

Graficar :

Funcionalidad: Genera una representación gráfica de los TDA, mostrando la información almacenada en su nodos.

Importancia: Mejora la experiencia del usuario y del administrador, ya que proporciona un la información almacenada gráficamente .

Manejo de ventanas de la aplicación:

Funcionalidad: Maneja el retorno a la pantalla de login desde cualquier punto de la aplicación, asegurando que los usuarios puedan cambiar de cuenta o cerrar sesión de manera segura.

Importancia: Es esencial para la gestión de sesiones y seguridad del sistema, asegurando que los accesos y permisos sean correctamente administrados.

Conclusiones

El diseño e implementación de IPCmarket demuestran cómo el uso estratégico de TDAs y la combinación de principios POO sólidos pueden conducir al desarrollo de sistemas robustos y eficientes. El sistema utiliza PyQt5 para mantener una arquitectura limpia que separa la presentación de la lógica comercial de una interfaz amigable y reactiva. El desarrollo requirió soluciones innovadoras para el manejo de datos y la concurrencia debido a los problemas encontrados. Estos dos elementos son esenciales para el éxito de cualquier aplicación comercial en la actualidad.

Referencias bibliográficas

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). Introduction to algorithms (3rd ed.). MIT Press.

Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley.

Silberschatz, A., Korth, H. F., & Sudarshan, S. (2013). Database system concepts (6th ed.). McGraw-Hill.

USAC, T.d (16 de Junio de 2024), Proyecto 1, UEDi. Obtenido de: https://uedi.ingenieria.usac.edu.gt/campus/course/view.php?id=17518

Anexos

A continuación se presentan diagramas de las estructuras en dónde se almacenan los distintos datos del programa:

Nodo simple:

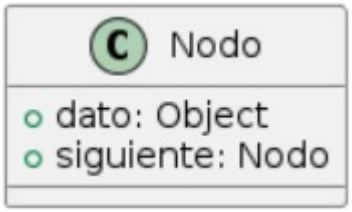

Figura 5. Estructura del nodo simple.

Fuente: elaboración propia, 2024.

Nodo Doble:

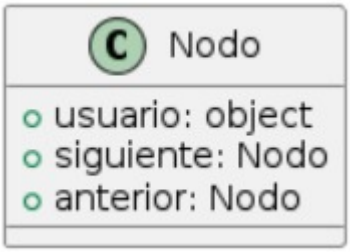

Figura 6. Estructura del nodo doble.

Fuente: elaboración propia, 2024.

Nodo doble de una lista circular doblemente enlazada:

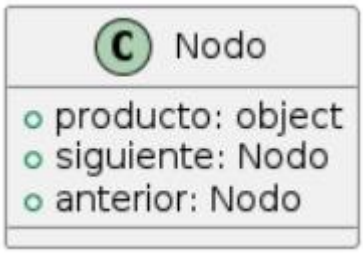

Figura 7. Estructura del nodo doble de una lista circular.

Fuente: elaboración propia, 2024.

Nodo Celda:

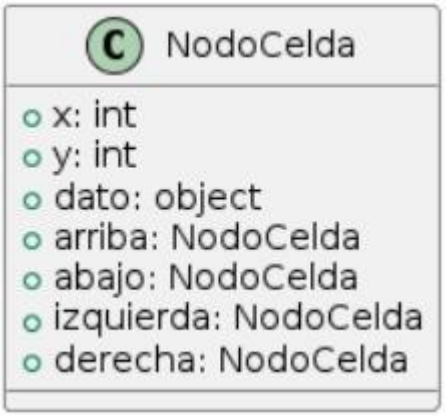

Figura 8. Estructura del nodo celda.

Fuente: elaboración propia, 2024.

Nodo simple para una lista circular simplemente enlazada:

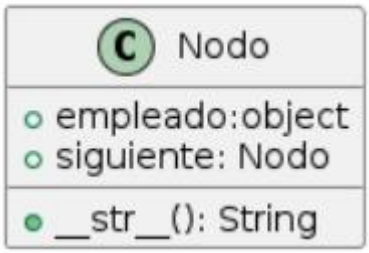

Figura 9. Estructura del nodo simple de una lista circular.

Fuente: elaboración propia, 2024.

Nodo cabecera:

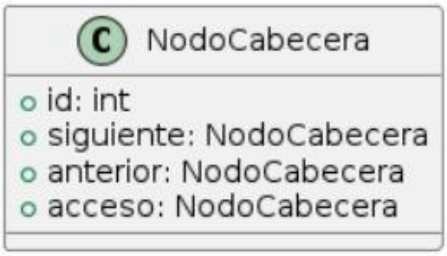

Figura 10. Estructura del nodo cabecera.

Fuente: elaboración propia, 2024.

Lista doblemente enlazada:

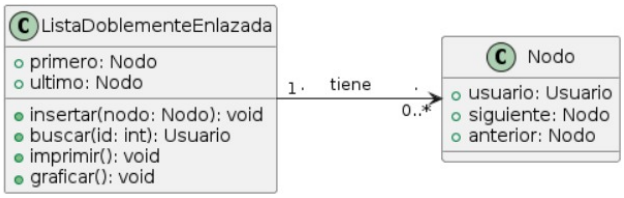

Figura 11. Estructura de la lista doblemente enlazada.

Fuente: elaboración propia, 2024.

Lista circular doblemente enlazada:

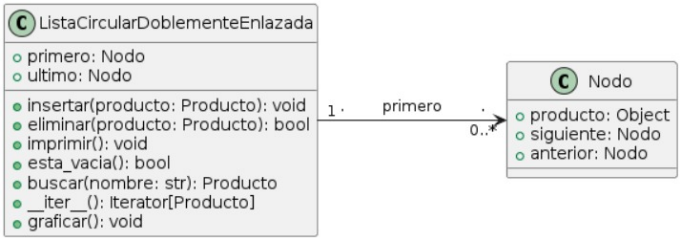

Figura 12. Estructura de la lista circular doblemente enlazada.

Fuente: elaboración propia, 2024.

Lista circular simplemente enlazada:

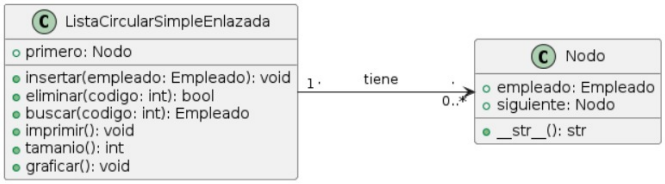

Figura 13. Estructura de la lista circular simplemente enlazada.

Fuente: elaboración propia, 2024.

Lista simplemente enlazada:

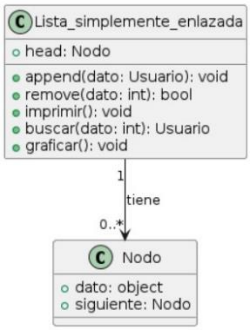

Figura 14. Estructura de la lista simplemente enlazada.

Fuente: elaboración propia, 2024.

Pilas:

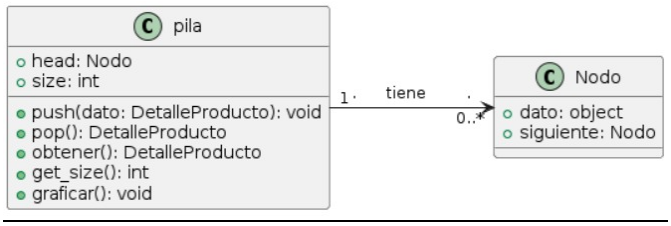

Figura 15. Estructura de la pila.

Fuente: elaboración propia, 2024.

Cola:

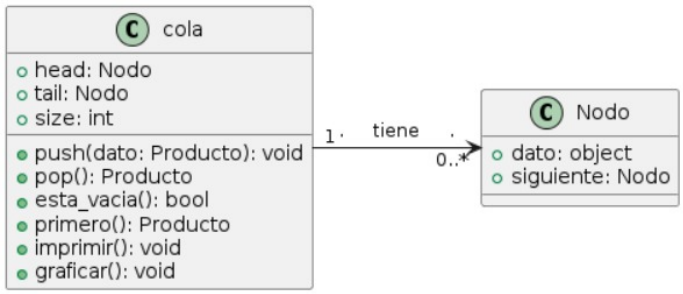

Figura 16. Estructura de la cola.

Fuente: elaboración propia, 2024.

Matriz dispersa;

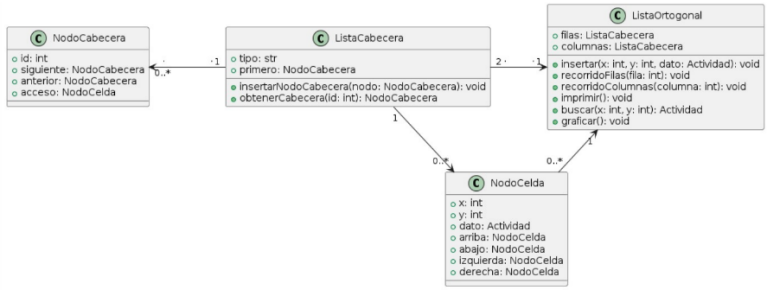

Figura 17. Estructura de la matriz dispersa.

Fuente: elaboración propia, 2024.

Universidad de San Carlos de Guatemala
Escuela de Ingeniería en Ciencias y Sistemas, Facultad de Ingeniería
Introducción a la programación y computación 2, Vacaciones de junio 2024

Diagrama de clase del proyecto en general: