Proyecto 2

202109567 – Alvaro Norberto García Meza

Resumen

Dicho proyecto fue propuesto con la finalidad de crear un sistema de atención a clientes que pueda ser manejado por cualquier organización que necesite brindar servicios presenciales a sus clientes.

La aplicación brindará un servicio de atención presencial a sus clientes que estén suscritos al servicio de atención. Cada cliente podrá escoger que empresa y que punto de atención desea asistir, posteriormente ingresará el numero de transacciones que desea realizar en dicho lugar y se le proporcionará un numero de cola, en el cual al llegar al lugar será atendido conforme va llegando ya que el comportamiento es al de una cola. Por otra parte, la empresa que se registre en dicho sistema tendrá la posibilidad de registrar n cantidad de puntos, escritorios y transacciones. Además, se puede leer archivos con la extensión xml y programar un archivo de prueba para simular el comportamiento de un punto de atención donde al final mostrará tiempos de servicio y espera.

Palabras clave

Empresa, puntos de atención, clientes, cola, transacciones, tiempo y simulación

Abstract

This project was proposed to create a customer service system that can be managed by any organization that needs to provide face-to-face services to its customers.

The application will provide a face-to-face customer service to its customers who are subscribed to the customer service. Each client will be able to choose which company and which service point they wish to attend, later they will enter the number of transactions they wish to carry out in said place and they will be provided with a queue number, in which upon arrival at the place they will be attended as they arrive since the behavior is that of a queue. On the other hand, the company that registers in said system will have the possibility of registering n number of points, desks, and transactions. In addition, you can read files with the xml extension and program a test file to simulate the behavior of a service point where, at the end, it will show service and waiting times.

Keywords
Company, attention points, clients, queue, transactions, time and simulation

Introducción

El desarrollo de la aplicación se realizó con la ayuda de diferentes paradigmas de programación y el uso de estructuras de datos. Los paradigmas de programación implementados fueron Programación Orientada a Objetos, programación funcional y modular para desarrollar de manera ordenada las diferentes funcionalidades las cuales en algunos casos eran dependientes de otras.

La carga de los datos fue establecida al formato XML (Lenguaje de Marcado Extensible), donde almacenaba los atributos principales de una empresa, una lista de puntos de atención que a su vez contenía una lista de escritorios, luego otra lista que contenía transacciones. Posteriormente se analizó otro XML con diferente estructura, pero íntimamente ligado al primero ya que este estableció la configuración inicial para realizar la simulación de un cierto punto de atención con n cantidad de clientes. Cada lista anidada, representaba un objeto con diferentes atributos.

Para realizar la simulación, primero se limpiaron los datos tomando en cuenta únicamente los más importantes, se separó en dos listas simples enlazadas escritorios y clientes, el comportamiento de la lista simple enlazada de los clientes era igual al de una cola, donde el ultimo en entrar era el ultimo en salir. Al salir de la cola el cliente se almacenaba en la posición de un escritorio que estuviese vacío y activo, se demoraba el tiempo establecido por la cantidad de transacciones y luego salía, el escritorio volvía a estar vacío en espera de un nuevo cliente y así hasta que la cola no tuviera ningún cliente en espera.

Por último, se mostró los tiempos de espera en cola, y los tiempos de servicio de cada escritorio, además, se implementó la herramienta grahpviz la cual generaba por cada cambio que ocurría en la simulación, tanto el estado inicial, como cuando los escritorios estaban llenos y posteriormente vacíos.

Desarrollo del tema

La aplicación fue desarrollada con el lenguaje de programación “Python” debido a su gran facilidad de sintaxis y manejo de variables. Además, implementando librerías integradas del propio lenguaje.

A. Directorios y Módulos

Cada proyecto y aplicación debe contener cierta jerarquía y organización para un manejo ordenado del a información, en este caso un buen manejo de la funcionalidad y datos globales.

- **src:** El primero directorio de la jerarquía el cual contiene toda la aplicación
    - **Classes:** Es el primer directorio el cual contiene los módulos los cuales son todas las clases utilizadas para el desarrollo del programa.
        - **Attention_point.py**
        - **Client_trasactin.py**
        - **Client.py**
        - **Company.py**
        - **Desk.py**
        - **Init_desk.py**
        - **Initial_state.py**
        - **Transaction.py**

- **F_menu:** El primer directorio, es el encargado de gestionar el primer menú de la aplicación donde se puede cargar archivos xml y crear empresas, este contiene un archivo principal y un directorio con componentes auxiliares.
  - **F_menu.py**
  - **Components**
    - **Company.init_xml.py**
    - **Company_xml.py**
- **S_menu:** El segundo directorio es el encargado de manejar el segundo menú de la aplicación donde se puede escoger empresa y punto de atención.
  - **S_menu.py**
- **T_menu:** El tercer directorio es el encargado de manejar el tercer menú de la aplicación y en donde se encuentra la lógica de todo el programa como la simulación.
  - **T_menu.py**
- **Tdas:** En el ultimo directorio se encuentra la estructura de datos utilizada para toda la aplicación, la cual es una lista simple enlazada que contiene métodos que sirven para simular una cola y pila.
  - **linkedList.py**
  - **node.py**
- **main.py:** En este archivo se encuentra el menú principal, el cual gestiona los otros 3 directorios y hace que los datos se comuniquen correctamente.

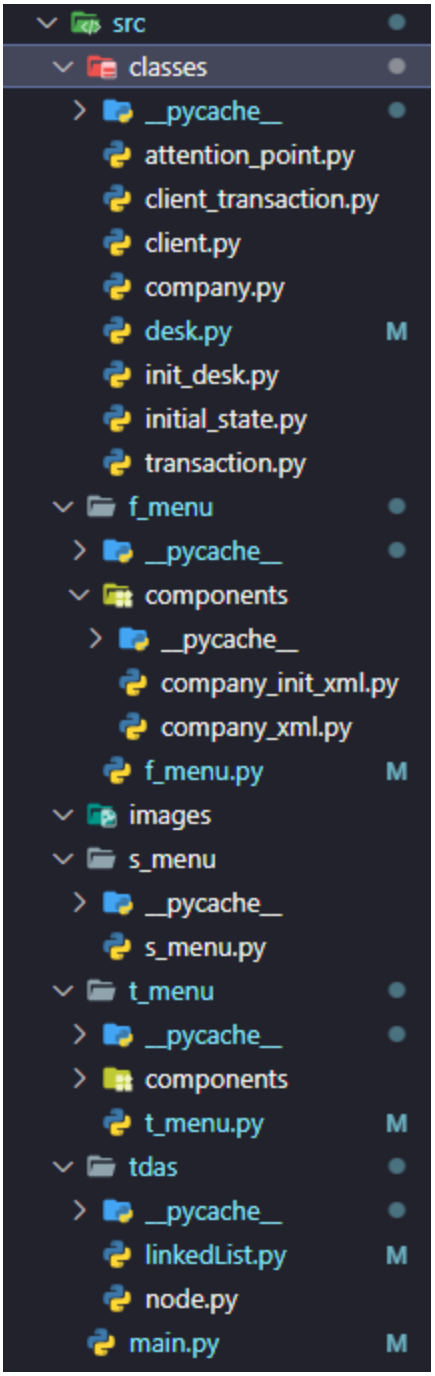

Figura 1. Directorios y Módulos.

Fuente: elaboración propia, 2022.

B. Clases

Al implementar distintos paradigmas de programación como Programación Orientada a Objetos, se implementó distintas clases para crear objetos que se complementarán entre si y fueran almacenados en una instancia de una lista simple enlazada.

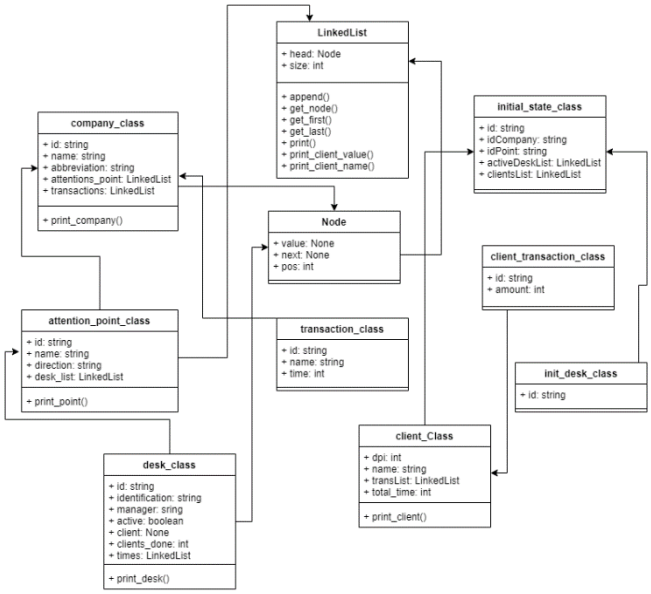

Figura 2. Diagrama de clases.

Fuente: elaboración propia, 2022.

C. Lista simple enlazada

Para la simulación de una cola y pila, se utilizo una lista simple enlazada la cual contenía diferentes métodos como por ejemplo para obtener sacar el primer valor de la lista, sacar el ultimo valor ingresado, retornar un valor de la lista simple e imprimir diferentes atributos de los nodos que contenía.

```python
class LinkedList:

    def __init__(self) -> None:
        self.head: Node = None
        self.size = 0

    # append a value to the list
    # this append at the end
    def append(self, value):
        if self.head is None:
            self.head = Node(value, self.size)
        else:
            aux = self.head
            while aux.next is not None:
                aux = aux.next
            aux.next = Node(value, self.size)

        self.size += 1

    # get the value
    def get_node(self, index):
        aux = self.head
        while aux is not None:
            if aux.pos == index:
                return aux.value
            aux = aux.next
        return None
    # get first node -> to FIFO
    def get_first(self):
        head = self.head
        self.head = self.head.next
        self.size -= 1
        return head.value

    # get last node -> LIFO
    def get_last(self):
        current = self.head
        next = current.next
        while next is not None:
            current = next
            next = current.next
        return current.value

    def print(self):
        current = self.head
        while current:
            print(current.value)
            current = current.next
    # print
    def print_client_value(self):
        current = self.head
        while current:
            print(current.value.print_client())
            current = current.next

    def print_client_name(self):
        current = self.head
        clients = ''
        while current:
            clients += f'C_{current.value.dpi}[label="{current.value.name}"]\n'
            current = current.next
        return clients
```

Figura 3. Lista simple enlazada.

Fuente: elaboración propia, 2022.

D. Módulos

La aplicación esta divida en diferentes directorios que cuentan con diferentes módulos que corresponden a una tarea especifica. Como primer punto tenemos al módulo principal que controla los demás módulos o sea los submenús.

Modulo: main.py

Como primer modulo tenemos el que controla todos los demás módulos de la aplicación y donde los datos se comunican entre sí.

```python
class Main_Menu:
    # constructor
    def __init__(self) -> None:
        self.companies = LinkedList()
        self.init_state = None
        self.selected = None

    def init_menu(self):
        option = 0

        while option != 4:
            print('Bienvenidos al servidor de Soluciones Guatemala SA')
            print('-----------------------------------------------------')
            print("""
        ¿Qué desea hacer?
        1. Configuración de empresa
        2. Selecionar una empresa
        3. Manejo de puntos de atención
        4. Salir
            """)
            option = int(input('Ingrese una opción: '))
            if option == 1:
                os.system('cls')
                menu_1 = first_menu() # call the first menu
                menu_1.init_menu()
                a , b = menu_1.get_attibutes()
                if a.size != 0:
                    for i in range(a.size):
                        company = a.get_node(i)
                        self.companies.append(company)
                self.init_state = b

            elif option == 2:
                os.system('cls')
                menu_2 = second_menu(self.companies)
                menu_2.init_menu()
                selected = menu_2.get_selected()
                self.selected = selected
            elif option == 3:
                os.system('cls')
                menu_3 = third_menu(self.companies,self.init_state)
                #menu_3.prepare_all()
                menu_3.ini_menu()
            elif option == 4:
                os.system('cls')
                print('ADIOS!')
            else:
                print('Ingrese una opción valida')
                os.system('cls')
```

Figura 4. Menú principal.

Fuente: elaboración propia, 2022

Módulo: f_menu.py

Este modulo se encarga de gestionar la carga de archivos XML a la aplicación, donde se escribe la dirección de ubicación de los archivos, se leen y se crean objetos de tipo empresa y estado inicial. Además, se pueden crear n empresas con n puntos de atención y transacciones.

Al salir del submenú los valores se devuelven en forma de una lista simple enlazada que se puede comunicar con otros módulos. Siempre que se quiera regresar a cargar nuevos archivos se podrá hacer

Figura 6. Módulo: f_menu.py.

Fuente: elaboración propia, 2022

Módulo: s_menu.py

En este módulo se muestran las empresas cargadas a la aplicación y da la posibilidad de escoger que empresa y en que punto se quiere realizar la simulación.

```
class second_menu:

    def __init__(self,companies) -> None:
        self.companies = companies
        self.selected = None


    def get_selected(self):
        return self.selected

    def init_menu(self):
        option = 0
        option_p = 0
        print('--------------------------')
        print('LISTA DE EMPRESAS')
        print('--------------------------')

        for i in range(self.companies.size):
            company = self.companies.get_node(i)
            print(f'''
    No. {i + 1}
    {company.print_company()}
    ''')

        option = int(input('\nQué empresa desea realizar la prueba? \n'))
        os.system('cls')


        selected = self.companies.get_node(option-1)
        self.selected = selected

        print(f'''
    Se he seleccionado la empresa: {selected.name}
        ''')
        for i in range(selected.attentions_point.size):
            point = selected.attentions_point.get_node(i)
            print(f'''
    Punto de atención No. {i+1}
    {point.print_point()}
            ''')

        option_p = int(input('\nQué punto de atención desea utilizar? \n'))
        selected_p = selected.attentions_point.get_node(option_p-1)
        os.system('cls')


        print(f'''
    Se ha seleccionado la empresa: {selected.name}
    En el punto de atención: {selected_p.name}
        ''')
        # ask again
        ask = 0
        while ask != 2:
            print('''
        1. Si
        2. No
            ''')
            ask = int(input('¿Desea escoger nuevamente?'))

            if ask == 1:
                os.system('cls')
                self.init_menu()
            elif ask == 2:
                os.system('cls')
```

Figura 7. Módulo: s_menu.py.

Fuente: elaboración propia, 2022

Módulo: t_menu.py

Este módulo es el más importante y grande ya que cuenta con toda la lógica de la aplicación, donde llegan los valores limpios, posteriormente se puede escoger que simulación se desea realizar, despliega el estado inicial de la simulación, permite activar o desactivar escritorios y al momento de terminar de ejecutar muestra los tiempos de servicio y espera. Además, muestra todo el comportamiento de la simulación mediante imágenes generadas en graphviz.

```
class third_menu:

    def __init__(self,companies,init_states) -> None:
        self.companies = companies
        self.init_state = None
        self.init_states = init_states
        self.active_desk = None
        self.clients_queue = None
        self.company = None
        self.total_time = 0
        self.times = 0
        self.point = None
        self.times_in_queue = []

    def ini_menu(self):
        for i in range(self.init_states.size):
            state = self.init_states.get_node(i)
            print(f'''
    Simulación No.{i+1}
    ------------------
    {state.id} - {state.idCompany} - {state.idPoint}
            ''')

        option_state = int(input('¿Qué simulación deseas realizar?\n'))
        self.init_state = self.init_states.get_node(option_state-1)
        os.system('cls')
        self.prepare_all()

        print('  ESTADO INICIAL  ')
        print('----------------------')
        self.print_grahpviz()
        self.print_queue()


        option = 0
        while option != 3:
            self.times += 1
            print('''
    ---------------------------------
    1. Iniciar Prueba
    2. Activar o desactivar escritorio
    3. No Iniciar prueba
    ---------------------------------
            ''')
            option = int(input('¿Qué desea hacer?\n'))
            if option == 1:
                os.system('cls')
                print('----------------------')
                print('  INICIANDO PRUEBA  ')
                print('----------------------')
```

Figura 8. Módulo: t_menu.py.

Fuente: elaboración propia, 2022

Conclusiones

El propósito de este proyecto es leer, generar e interpretar de manera correcta archivos con el formato XML, cumplir la necesidad que el programa solicita mediante el uso de todas las herramientas que la programación ofrece como los paradigmas de programación.

Dicho proyecto busca fortalecer los conocimientos de como las estructuras de datos funcionan ya que estas son la base de cómo se construyen las diferentes listas nativas de un lenguaje de programación y aplicar conceptos matemáticos para poder adaptar una herramienta a cualquier necesidad.

Por último, hacer uso de la herramienta grahpviz para la generación de grafos o en este caso listas enlazadas y sus derivados.

Referencias bibliográficas

The Graphviz Authors. (2022). Documentation. Graphviz. Recuperado 6 de septiembre de 2022, de https://graphviz.org/documentation/

A linked list program using one class. (2020, 19 septiembre). Stack Overflow. Recuperado 6 de septiembre de 2022, de https://stackoverflow.com/questions/63973503/a-linked-list-program-using-one-class

xml.etree.ElementTree — The ElementTree XML API — Python 3.10.6 documentation. (s. f.). Recuperado 6 de septiembre de 2022, de https://docs.python.org/3/library/xml.etree.elementtree.html

Anexos

```xml
<?xml version="1.0"?>
<listaEmpresas>
    <!-- Pueden haber más empresas -->
    <empresa id="1">
        <nombre> Claro Guateamala </nombre>
        <abreviatura> Claro </abreviatura>
        <listaPuntosAtencion>
        <!-- Pueden haber más puntos de atencion -->
            <puntoAtencion id="1">
                <nombre> Centro Claro </nombre>
                <direccion> Okland Mall 3er Nivel </direccion>
                <listaEscritorios>
                    <!-- Pueden haber más escritorios -->
                    <escritorio id="1">
                        <identificacion> #e1 </identificacion>
                        <encargado> Empleado Jorge </encargado>
                    </escritorio>
                    <!-- Pueden haber más escritorios -->
                    <!-- Pueden haber más escritorios -->
                    <escritorio id="2">
                        <identificacion> #e2 </identificacion>
                        <encargado> Empleado Mauricio </encargado>
                    </escritorio>
                    <!-- Pueden haber más escritorios -->
                    <!-- Pueden haber más escritorios -->
                    <escritorio id="3">
                        <identificacion> #e3 </identificacion>
                        <encargado> Empleado Paulo </encargado>
                    </escritorio>
                    <!-- Pueden haber más escritorios -->
                </listaEscritorios>
            </puntoAtencion>
        <!-- Pueden haber más puntos de atencion -->
        </listaPuntosAtencion>

        <listaTransacciones>
            <!-- Pueden haber más transacciones -->
            <transaccion id="1">
                <nombre> Pago servicios </nombre>
                <tiempoAtencion> 2 </tiempoAtencion>
            </transaccion>
            <!-- Pueden haber más transacciones -->
        </listaTransacciones>
    </empresa>
    <!-- Pueden haber más empresas -->
</listaEmpresas>
```

Figura 9. Documento de entrada 1 en XML.

Fuente: elaboración propia, 2022

```xml
<?xml version="1.0"?>
<listadoInicial>
    <!--Puede venir muchas más configuraciones -->
    <configInicial id="1" idEmpresa="1" idPunto="1">
        <escritoriosActivos>
            <!--Puede venir muchos escritorios activos -->
            <escritorio idEscritorio="1"/>
            <escritorio idEscritorio="2"/>
            <!--Puede venir muchos escritorios activos -->
        </escritoriosActivos>
        <listadoClientes>
            <!--Puede venir muchos clientes -->
            <cliente dpi="3233923923329">
                <nombre> Alvaro García</nombre>
                <listadoTransacciones>
                    <!--Puede venir muchas transacciones -->
                    <transaccion idTransaccion="1" cantidad="1"/>
                    <transaccion idTransaccion="1" cantidad="1"/>
                    <!--Puede venir muchas transacciones -->
                </listadoTransacciones>
            </cliente>
            <!--Puede venir muchos clientes -->
            <!--Puede venir muchos clientes -->
            <cliente dpi="32323232">
                <nombre> Julissa Reyes</nombre>
                <listadoTransacciones>
                    <!--Puede venir muchas transacciones -->
                    <transaccion idTransaccion="1" cantidad="1"/>
                    <transaccion idTransaccion="1" cantidad="1"/>
                    <!--Puede venir muchas transacciones -->
                </listadoTransacciones>
            </cliente>
            <!--Puede venir muchos clientes -->
            <!--Puede venir muchos clientes -->
            <cliente dpi="12323432132">
                <nombre> Kevin García</nombre>
                <listadoTransacciones>
                    <!--Puede venir muchas transacciones -->
                    <transaccion idTransaccion="1" cantidad="1"/>
                    <transaccion idTransaccion="1" cantidad="1"/>
                    <!--Puede venir muchas transacciones -->
                </listadoTransacciones>
            </cliente>
            <!--Puede venir muchos clientes -->
        </listadoClientes>
    </configInicial>
    <!--Puede venir muchas más configuraciones -->
</listadoInicial>
```

Figura 10. Documento de entrada 2 en XML.

Fuente: elaboración propia, 2022