PROYECTO 2

202104782 – Anthony Samuel Zea Herrera

Resumen

La empresa “Soluciones Guatemaltecas, S.A.” está construyendo un sistema de atención a clientes diseñado para cualquier organización que necesite brindar servicios presenciales a sus clientes. “Soluciones Guatemaltecas, S.A” está innovando la Experiencia del Cliente (CX) planteando la siguiente solución para empresas que requieren atender presencialmente a sus clientes:

1. Las empresas que brindan atención presencial a sus clientes se suscribirán al servicio de atención al cliente de “Soluciones Guatemaltecas, S.A.”.
2. Existirá una aplicación móvil gratuita
3. Deberá garantizar que los clientes son atendidos en el orden en que solicitan atención.
4. Brindará una aplicación Web que permitirá a las empresas suscritas Activar Escritorios de Servicios
5. Permitirá conocer en todo momento el estado de cada Escritorio de Servicio en cada punto de atención de la empresa.

Palabras clave
POO, Graphviz, lista enlazada, nodo, XML

Abstract

The company "Soluciones Guatemaltecas, S.A." is building a customer service system designed for any organization that needs to provide face-to-face services to its clients. "Soluciones Guatemaltecas, S.A" is innovating the Customer Experience (CX) proposing the following solution for companies that need to attend their customers in person:

1. *Companies that provide face-to-face service to their customers will subscribe to the service of customer service of "Soluciones Guatemaltecas, S.A.".*
2. *There will be a free mobile application*
3. *You must ensure that customers are served in the order in which they request attention.*
4. *Provide a Web application that will allow subscribed companies to Activate Service Desktops*
5. *It will allow to know at all times the status of each Service Desk in each point of attention of the company.*

Keywords
OOP, Graphviz, linked list, node, xml

Introducción

En la realización del proyecto se requeriría de un programa capaz de simular un sistema de atención al cliente, en el que se encontraban varios escritorios activos, para lo cual el cliente debía ser atendido por uno de los escritorios activos y así sucesivamente, hasta simular una cola.

El programa es capaz de realizar una lectura detallada de unos archivos XML, en la cual se deben encontrar los datos de cada empresa, de cada punto de atención, de las transacciones, y de los clientes, para posteriormente iniciar una prueba del sistema, como tal.

Para la manipulación sencilla de los datos, se utilizó una estructura de datos (lista enlazada), con sus respectivos nodos, y con ello manejar los atributos de dos dimensiones en una sola dimensión.

Gracias a la herramienta Graphviz, el programa genera un gráfico por estado evaluado para que el usuario vea detalladamente el sistema de atención al cliente.

Desarrollo del tema

Se requería de la realización de un programa con la capacidad de simular una atención al cliente, para que ciertos clientes fueran atendidos en distintos puestos los cuales se les denomina escritorios. Estos pueden estar activados o desactivados, y dependiendo su estado, podrá atender a un cliente.

Python es un lenguaje de programación bastante amigable con el usuario, teniendo la oportunidad de implementar distintos paradigmas de programación, entre ellos la Programación Orientada a Objetos, lo cual hace de Python uno de los lenguajes de alto nivel, más utilizados actualmente, entre ellos encontramos además de Python: Java, JavaScript, Go, C#, etc.

A menudo es considerado como un lenguaje “scripting”, ya que Python realmente es un lenguaje de propósito general, es por ello, que la implementación de dicho lenguaje de programación se veía obligado en la realización de dicho proyecto.

El proyecto se llevó a cabo implementando el paradigma de Programación Orientado a objetos, esto permitió el correcto uso de los atributos y datos rescatados durante todo el programa.

Diccionario de librerías

Durante el proyecto se implementaron ciertas librerías que permitieron un correcto funcionamiento y la forma de poder realizar el mismo, entre ellas están:

- **OS:** Esta librería o módulo provee una manera versátil de usar funcionalidades dependientes del sistema operativo.
- **ElementTree:** Esta biblioteca incluye herramientas para analizar XML usando APIs basadas en eventos y documentos, buscando documentos analizados con expresiones XPath, y creando nuevos o modificando documentos existentes.

Diccionario de Módulos:

Python ofrece la capacidad de trabajar toda la codificación en distintos archivos o módulos, es por ello por lo que dicho proyecto se dividió en distintos módulos que interactuaban sus funciones y atributos, los cuales son:

- **main:** En este modulo se encuentra toda la estructura de datos, en la que se visualiza la lista simple enlazada, con su respectivo nodo y funciones para la manipulación de datos.

- **menu:** Dicho modulo tiene la única finalidad de inicializar todo el programa mostrando el menú principal con todas las opciones disponibles para el usuario.
- **configInit:** Este módulo almacena las clases necesarias para la creación y almacenamiento de objetos del primer archivo XML, que se lee al inicio.
- **grafico:** Este modulo es el encargado de generar una imagen, utilizando la herramienta graphviz y así poder visualizar todo el contenido gráficamente
- **simulation:** Este modulo es el encargado de generar una simulación del programa, realizando la lectura y validaciones de ambos archivos para poder ver su contenido
- **testInit:** Este modulo contiene todas las clases necesarias para almacenar los objetos del segundo archivo XML.
- **xmlFile:** Este modulo es el encargado de realizar la lectura de los archivos xml, almacenando todos los datos en las listas enlazadas creadas en el programa.

Diccionario de clases:

Parte fundamental de la programación orientada a objetos son las clases, para lo cual se implementaron ciertas clases importantes durante todo el desarrollo del programa:

- **class App:** Dicha clase es la clase principal, la cual se inicializa al momento de ejecutar el programa, ya que esta contiene al menú principal con todas las funciones.
- **class Lista_Enlazada:** Dicha clase, es una de las mas importantes ya que dentro de esta se encuentra básicamente la estructura de datos utilizada a lo largo de todo el programa.
- **class Nodo:** Dicha clase cuenta únicamente con su método constructor el cual tiene la única finalidad de manejar toda la información en la lista enlazada. Mediante nodos.
- **class Empresas:** Dicha clase es utilizada para el almacenamiento de las empresas que se encuentran en el primer archivo de entrada y cuenta con sus demas clases para poder almacenar a puntos de atención, transacciones y escritorios
- **class Configuración_inicial:** Dicha clase es utilizada para el almacenamiento de objetos respectivos sobre la configuración y prueba inicial que se llevara a cabo por el programa, y cuenta con sus demas clases como; Escritorios_activos, Clientes, transacciones.

Todas estas clases se manejan a lo largo de todo el proyecto mediante objetos lo cual permite el acceso a cada uno de los métodos de estas clases. Mediante el diagrama de clases se observa se la siguiente manera:

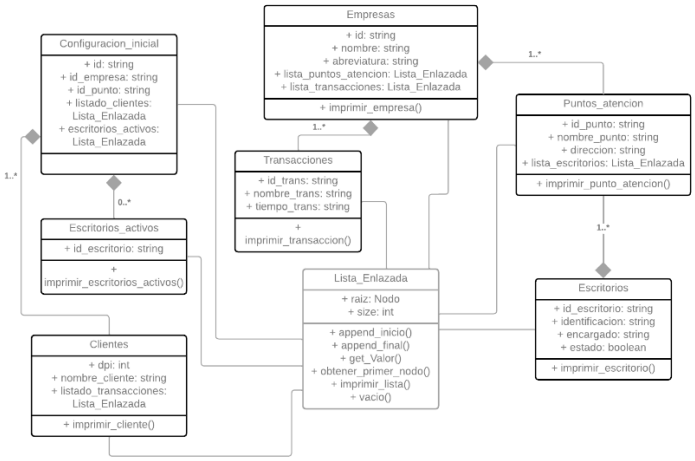

Figura 1. Diagrama de clases.
Fuente elaboración propia, 2022.

Mediante el diagrama de clases se puede observar detalladamente como se relaciona la clase Lista_Enlazada, con las clases de objetos más importantes las cuales son las relacionadas al manejo

de datos las empresas, clientes y configuraciones iniciales.

Generalmente se creo únicamente una estructura de datos la cual se implemento para almacenar todo tipo de datos, la cual hace referencia a un nodo, para poder manejarlo de manera mas optimizada y sencilla

```python
# Creamos clase Nodo, para todas las listas
class Nodo():
    def __init__(self, dato, pos) -> None:
        self.dato = dato
        self.pos = pos
        self.siguiente = None

# ---------------- Aqui viene todo el TDA (Linked List) -------------------- #
class Lista_enlazada():

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

    # Metodo para agregar datos al inicio
    def append_inicio(self, dato):
        if self.raiz == None:
            self.raiz = Nodo(dato, self.size)
        else:
            aux = self.raiz
            while aux.siguiente != None:
                aux = aux.siguiente
            aux.siguiente = Nodo(dato, self.size)

        self.size += 1

    # Metodo para agregar datos al final
    def append_final(self, dato):
        if not self.raiz:
            self.raiz = Nodo(dato = dato)
            return
        actual = self.raiz
        while actual.siguiente:
            actual = actual.siguiente
        actual.siguiente = Nodo(dato = dato)

    def get_Valor(self, indice):
        aux = self.raiz
        while aux is not None:
            if aux.pos == indice:
                return aux.dato
            aux = aux.siguiente
        return None

    def obtener_primer_nodo(self):
        request = self.raiz
        self.raiz = self.raiz.siguiente
        return request

    def imprimir_lista(self, llave):
        nodo = self.raiz
        print('\nEscritorio', end=' - ')
        while nodo != None:
            print(getattr(nodo.dato, llave), end=' - ')
            nodo = nodo.siguiente
        print('\n')

    # Método para verificar si la estructura de datos esta vacia
    def vacio(self):
        return self.raiz == None
```

Figura 2. Class Lista_enlazada
Fuente: Elaboración propia, 2022.

En este caso observamos ciertas funciones que manipulan toda la información relacionada a las empresas, configuración inicial, clientes, escritorios, etc. Y esto cumple la funcionalidad de poder llevar una estructura de datos de una lista simple enlazada con implementación de colas.

Cumpliendo con este paradigma, a lo largo de todo el programa objetos de esta clase son los encargados de almacenar todos los datos en todas las clases a las cuales se hace mencion

El proyecto consta de ciertas funciones importantes que cumplen el propósito del programa, que vienen desde la lectura y escritura de un archivo XML, imprimir en consola la simulación de la atención al cliente, o crear un gráfico mediante el uso de la herramienta Graphviz y generar en una carpeta las imágenes correspondientes a atención a cliente

Diccionario de funciones:

- **append_inicio:** Esta función es básicamente la encargada de añadir elementos u objetos a una lista enlaxada, al inicio de esta, simplificando el trabajo.
- **get_Valor:** Esta función recibe como parámetro un número, y dentro de esta, itera en la lista enlazada, hasta encontrar el numero que se le ha pasado como parámetro, esto con el único fin, de devolvernos el nodo en el cual hayamos indicado en el índice.
- **Obtener_primer_nodo:** Esta función representa lo que se denomina como “Cola”, ya que básicamente lo que hace, es tomar el primer nodo que se encuentra en la lista enlazada, y al mismo tiempo eliminándolo permanentemente de esta, simulando el proceso de una cola, o fila.
- **imprimir_lista:** Esta función es la encargada sencillamente de iterar sobre la lista enlazada e ir imprimiendo cada valor de ella, en este

caso nodos que contienen objetos, debido a que es una lista enlazada de objetos.

Graphviz

Graphviz es una colección de software para representar gráficos, muy flexible, pero requiere un cierto esfuerzo de aprendizaje. El paquete CPAN Graphviz nos ofrece una interfaz sencilla al software original, perdiendo algunas capacidades.

Gracias a la gran ampliación de bibliotecas y librerías que Python nos ofrece, nos permite trabajar con esta herramienta, de manera óptima e implementado código propio de Python junto con código de Graphviz. Dentro de Python se puede llamar a la biblioteca “os”, para poder acceder a las rutas del sistema, y es por ello por lo que mediante el uso de esta importación nos permite generar los dichos gráficos, sin embargo, anterior a ello, es necesario realizar la instalación ya sea mediante importación privada de Python o instalando el ejecutable de graphviz, ambas funcionalidades cumplen el mismo objetivo.

Con el pasar de los años Graphviz ha ido implementado muchas más funciones que le facilitan el trabajo al usuario, es por ello por lo que esta herramienta acepta código en lenguaje html, dándonos la posibilidad de una gran infinidad de gráficos realizados desde Python y graphviz.

XML

XML es el acrónimo de Extensible Markup Language, es decir, es un lenguaje de marcado que define un conjunto de reglas para la codificación de documentos. Esta basado en texto que tiene una estructura de autodescripción y puede definir efectivamente otro lenguaje de marcado. En este caso el XML, no permite ningún error en el código implementado.

Lista Enlazada

Una lista enlazada es una estructura de datos lineal que consta de un grupo de nodos donde cada nodo apunta al siguiente nodo a través de un puntero. Cada nodo se compone de datos y una referencia o enlace, al siguiente nodo de referencia.

Conclusiones

En conclusión, básicamente todo el proyecto se ve referenciado a un paradigma de programación orientado a objetos, ya que esto facilitó de gran manera la manipulación de datos durante todo el programa.

El enfoque principal para el manejo de datos estuvo presente en una estructura de datos lineal, la cual es la lista enlazada, permitiendo que cada posición se relacionara con su valor siguiente, y de esta manera se manejó todo el proceso de manipulación de las celdas y rejillas del programa.

Gracias a la herramienta XML y Graphviz, se pudo crear de manera intuitiva y estructurada un reporte de salida del paciente analizado debido a que graphviz genero una imagen por cada cliente atendido, y de esta manera visualizar todo mas intuitivamente y a su misma vez, atractivo para el usuario quien este ejecutando el programa.

Referencias bibliográficas

E. R. Schmidt, (2019). *xml.etree.ElementTree – Interfaz de programación de manipulación XML*. Sphinx. Recuperado el 5 de septiembre de 2022, de https://rico-schmidt.name/pymotw-3/xml.etree.ElementTree/index.html

The Graphviz Autors, (2022). *Graphviz.* Graphviz. Recuperado el 6 de septiembre de 2022, de https://graphviz.org/

Python Software Foundation. (2022). *Python 3.10.7 documentation.* Python. Recuperado el 7 de septiembre de 2022, de https://docs.python.org/3/

Anexos

```
# Metodo para agregar datos al inicio
    def append_inicio(self, dato):
        if self.raiz == None:
            self.raiz = Nodo(dato, self.size)
        else:
            aux = self.raiz
            while aux.siguiente != None:
                aux = aux.siguiente
            aux.siguiente = Nodo(dato, self.size)

        self.size += 1
```

Figura 3. Función append_inicio
Fuente: Elaboracion propia, 2022.

```
def get_Valor(self, indice):
        aux = self.raiz
        while aux is not None:
            if aux.pos == indice:
                return aux.dato
            aux = aux.siguiente
        return None
```

Figura 4. Función get_Valor
Fuente: Elaboracion propia, 2022.

```
def obtener_primer_nodo(self):
        request = self.raiz
        self.raiz = self.raiz.siguiente
        return request
```

Figura 5. Función obtener_primer_nodo
Fuente: Elaboracion propia, 2022.

```
def imprimir_lista(self, llave):
        nodo = self.raiz
        print('\nEscritorio', end=' - ')
        while nodo != None:
            print(getattr(nodo.dato, llave), end=' - ')
            nodo = nodo.siguiente
        print('\n')
```

Figura 6. Función imprimir_lista
Fuente: Elaboracion propia, 2022.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<listaEmpresas>
    <empresa id="25">
        <nombre> Gabriel y Asociados </nombre>
        <abreviatura> G.Y.A </abreviatura>
        <listaPuntosAtencion>
            <puntoAtencion id="2">
                <nombre> Majadas </nombre>
                <direccion> Centro comercial Majadas Once</direccion>
                <listaEscritorios>
                    <escritorio id="1">
                    <identificacion> #1f48</identificacion>
                    <encargado> Payaso Aymar </encargado>
                    </escritorio>
                    <escritorio id="2">
                    <identificacion> #1f49</identificacion>
                    <encargado> Nico Hagen </encargado>
                    </escritorio>
                </listaEscritorios>
            </puntoAtencion>
            <puntoAtencion id="4">
                <nombre> Cayalá </nombre>
                <direccion> Centro comercial Cayalá</direccion>
                <listaEscritorios>
                    <escritorio id="1">
                    <identificacion> #1f48</identificacion>
                    <encargado> Kun Aguero </encargado>
                    </escritorio>
                    <escritorio id="2">
                    <identificacion> #1f48</identificacion>
                    <encargado> Paco Memo </encargado>
                    </escritorio>
                </listaEscritorios>
            </puntoAtencion>
        </listaPuntosAtencion>
        <listaTransacciones>
            <transaccion id="249">
                <nombre> Pago Laptop </nombre>
                <tiempoAtencion> 30 </tiempoAtencion>
            </transaccion>
        </listaTransacciones>
    </empresa>
    <empresa id="28">
        <nombre> Patronilo y Asociados </nombre>
        <abreviatura> P.Y.A </abreviatura>
        <listaPuntosAtencion>
            <puntoAtencion id="3">
                <nombre> MetaMercado </nombre>
                <direccion> Centro comercial Meta</direccion>
                <listaEscritorios>
                    <escritorio id="1">
                    <identificacion> #1f48</identificacion>
                    <encargado> Lionel ronaldo Nazario </encargado>
                    </escritorio>
                    <escritorio id="2">
                    <identificacion> #1f49</identificacion>
                    <encargado> Courtouis </encargado>
                    </escritorio>
                    <escritorio id="3">
                    <identificacion> #1f50</identificacion>
                    <encargado> ElBicho </encargado>
                    </escritorio>
                </listaEscritorios>
            </puntoAtencion>
        </listaPuntosAtencion>
        <listaTransacciones>
            <transaccion id="249">
                <nombre> Pago PS5 </nombre>
                <tiempoAtencion> 10 </tiempoAtencion>
            </transaccion>
            <transaccion id="250">
                <nombre> Pago Xbox </nombre>
                <tiempoAtencion> 100 </tiempoAtencion>
            </transaccion>
        </listaTransacciones>
    </empresa>
</listaEmpresas>
```

Figura 7. Archivo1 XML de entrada

Fuente: Elaboracion propia, 2022.

```xml
<?xml version="1.0"?>
<listadoInicial>
    <configInicial id="21" idEmpresa="25" idPunto="2">
        <escritoriosActivos>
            <escritorio idEscritorio="1"/>
            <escritorio idEscritorio="2"/>
        </escritoriosActivos>
        <listadoClientes>
            <cliente dpi="145587">
                <nombre> Cristiano Ronaldo </nombre>
                <listadoTransacciones>
                    <transaccion idTransaccion="249" cantidad="1240"/>
                </listadoTransacciones>
            </cliente>
            <cliente dpi="101002">
                <nombre> Zidane </nombre>
                <listadoTransacciones>
                    <transaccion idTransaccion="249" cantidad="100"/>
                </listadoTransacciones>
            </cliente>
            <cliente dpi="353011">
                <nombre> Lio Messi </nombre>
                <listadoTransacciones>
                    <transaccion idTransaccion="249" cantidad="2800"/>
                </listadoTransacciones>
            </cliente>
            <cliente dpi="3111">
                <nombre> Jude Bellingham</nombre>
                <listadoTransacciones>
                    <transaccion idTransaccion="249" cantidad="100"/>
                </listadoTransacciones>
            </cliente>
        </listadoClientes>
    </configInicial>
</listadoInicial>
```

Figura 8. Archivo2 XML de entrada

Fuente: Elaboracion propia, 2022.