Universidad de San Carlos de Guatemala
Escuela de Ingeniería en Ciencias y Sistemas, Facultad de Ingeniería
Introducción a la programación y computación 2, 2do. Semestre 2022.

PRIMER PROYECTO IPC2

201902363 – CHRISTOPHER IVÁN MONTERROSO ALEGRIA

Resumen

Aplicación creada para investigación epidemiológica de Guatemala con el fin de calcular los patrones de sus de las células de sus pacientes para determinar la evolución de su enfermedad y con esto concluir el estado final o en ciertos periodos del paciente. Para esto se creó una aplicación con la capacidad de analizar archivos tipo xml, los cuales son cargados a una lista enlazada propia hecha desde cero, con este tipo de lista, se manejaron los archivos de lectura como también se utilizó para cada gráfica en su respectivo estado de cada paciente, y se utilizó para manejar los reportes finales de las investigaciones de los pacientes.

Palabras clave
Lista enlazada, Nodo, Matriz, Optimización.

Abstract

Application created for epidemiological research in Guatemala in order to calculate the patterns of their patients' cells to determine the evolution of their disease and with this conclude the final state or in certain periods of the patient. For this, an application was created with the ability to analyze xml type files, which are loaded into a linked list of its own made from scratch, with this type of list, the reading files were handled as was also used for each graph in its respective status of each patient, and was used to manage the final reports of patient investigations.

Keywords
Linked list, Node, Matrix, Optimization.

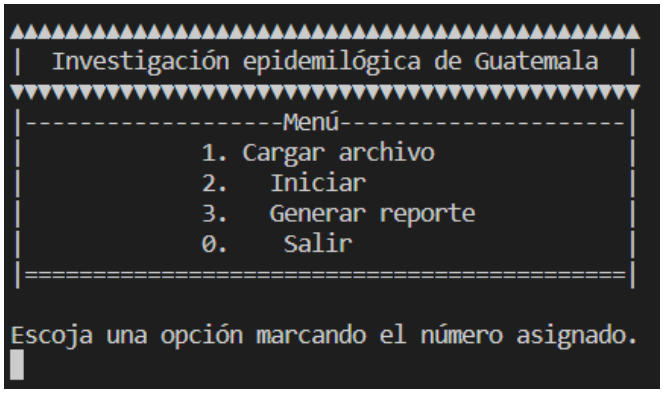
Introducción

La aplicación lee un archivo xml haciendo uso de la librería xml.etree.ElementTree, extrayendo la información, guardando en una lista, los datos del paciente como el nombre, edad, las dimensiones de su célula, los periodos a realizar de la investigación y sus células contagiadas, una vez cargada la información de cada paciente se presenta la lista de los pacientes para que se escoja el que se quiere investigar, para poder realizar la investigación se crea una matriz la cual también se usa para visualizar de forma gráfica las células de los pacientes, para el cálculo de la investigación se tiene la opción de avanzar un periodo o avanzar la totalidad de periodos permitidos según el paciente, para así dar con su estado final y su gráfica final. Para las gráficas se usó la librería graphviz.

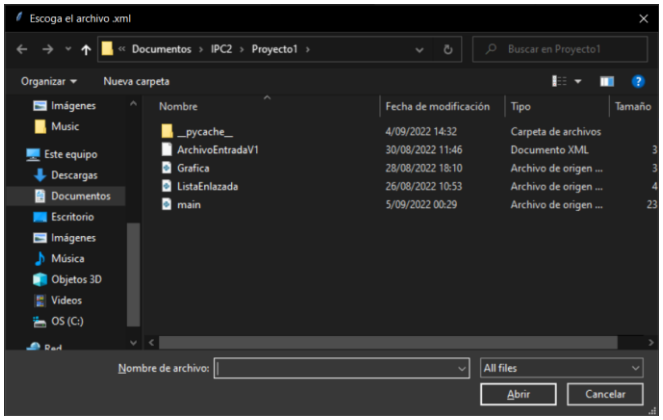
Desarrollo del tema

La aplicación inicia en su menú principal, el cuál nos muestra las siguientes opciones:

1. Carga de datos.
2. Iniciar.
3. Generar reporte.
4. Salir



La opción cargar datos nos presenta una ventana emergente la cual nos permite buscar nuestro archivo a cargar. Una vez cargado podemos iniciar el la visualización de pacientes dándole entrada al programa 2, sí ya hemos hecho la investigaciones requeridas podemos generar nuestro reporte dando una entrada de 3 y si deseamos salir del programa únicamente se debe dar entrada de 0.

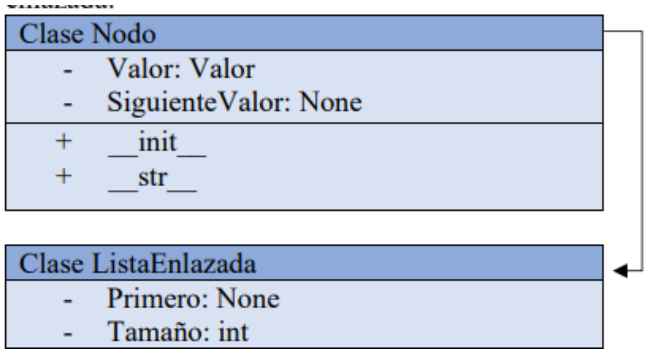


La carga de datos tiene un funcionamiento complejo, puesto que usa la librería xml, para su carga, se usa un solo método encargado de llevar la información a una variable global para su manejo de información. En dicho método se va creando una serie de listas con nodos de tipo listas para el almacenamiento de datos.

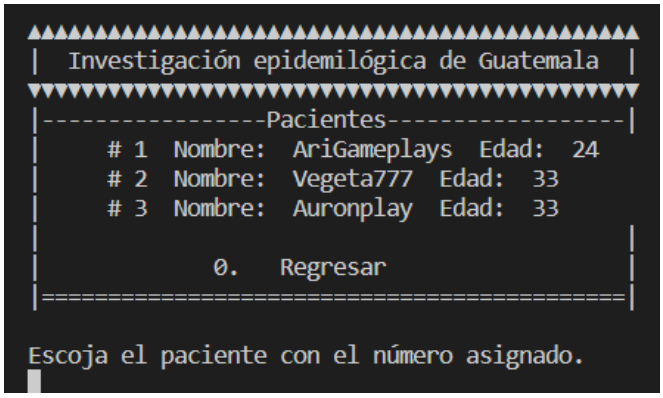
```
def CargarDatos():
    A=True
    while A:
        try:

            tree = ET.parse(easygui.fileopenbox(title="Escoga el archivo .xml"))
            root = tree.getroot()
            for i in range(len(root)):
                Datos = ListaEn()
                aux1 = ListaEn()
                aux2 = ListaEn()
                Datos.append(root[i][0][0].text)
                Datos.append(root[i][0][1].text)
                Datos.append(root[i][1].text)
                Datos.append(root[i][2].text)
                print(i+1,": paciente agregado")
                for j in range(len(root[i][3])):
                    aux1.append(root[i][3][j].get('f'))
                Datos.append(aux1)
                for k in range(len(root[i][3])):
                    aux2.append(root[i][3][k].get('c'))
                Datos.append(aux2)
                Lista.append(Datos)
                ids.append(i+1)

            A=False
        except :
            print("Error : Escoja un archivo válido")
```

Método para la carga de datos

Se utilizó una lista enlazada la cual funcionó como una lista preconfigurada de Python.

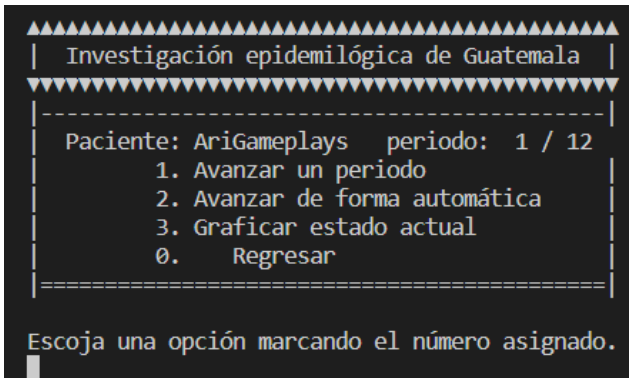


Una vez iniciado la carga de pacientes podremos escoger dando entrada con el número de paciente según nos presente la aplicación.



Una vez escogido el paciente a investigar, la aplicación nos presentará un menú con las siguientes opciones:

1. Avanzar un periodo.
2. Avanzar de forma automática los periodos.
3. Graficar el estado actual del paciente.



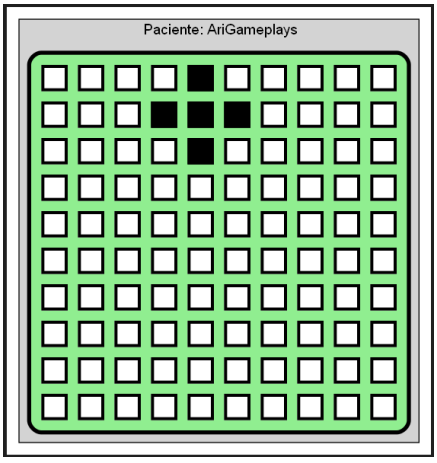
Para calcular el siguiente periodo se utilizó el siguiente método considerando las condiciones del mismo:

```
def Nuevo_periodo(matriz):
    CoordenadasX_Sanas=ListaEn()
    CoordenadasY_Sanas=ListaEn()
    CoordenadasX_Contagiadas=ListaEn()
    CoordenadasY_Contagiadas=ListaEn()
    final = len(matriz[0])-1
    for i in range(len(matriz)):
        for j in  range(len(matriz)):
            #Sanas
            if matriz[i][j]==0:

                # i+ mas es abajo
                # i- es arriba
                # j+ es derecha
                # j- es izquierda

                # esquina superior izquierda
                if i == 0 and j == 0:
                    if matriz[i+1][j]==1 and matriz[i][j+1]==1 and matriz[i+1][j+1]==1:
                            matriz[i][j]=1
                #esquina inferior izquierda
                elif i ==final and j==0:
                    if matriz[i-1][j]==1 and matriz[i][j+1]==1 and matriz[i-1][j+1]==1:
                            matriz[i][j]=1


                #esquina superior derecha
                elif i ==0 and j==final:
                    if matriz[i+1][j]==1 and matriz[i][j-1]==1 and matriz[i+1][j-1]==1:
                            matriz[i][j]=1


                #esquina inferior derecha
                elif i ==final and j==final:
                    if matriz[i-1][j]==1 and matriz[i][j-1]==1 and matriz[i-1][j-1]==1:
                            matriz[i][j]=1

                #pared izquierda
                elif i>0 and i<final and j==0 :
                    if (matriz[i-1][j]==1 and matriz[i-1][j+1]==1 and matriz[i][j+1]==1
                    or matriz[i-1][j+1]==1 and matriz[i][j+1]==1 and matriz[i+1][j+1]==1
                    or matriz[i][j+1]==1 and matriz[i+1][j+1]==1 and matriz[i+1][j]==1):
                        CoordenadasX_Sanas.append(j)
                        CoordenadasY_Sanas.append(i)

                #pared superior
                elif i==0 and j>0 and j<final:
                    if (matriz[i][j-1]==1 and matriz[i+1][j-1]==1 and matriz[i+1][j]==1
                    #  abajo - abajo derecha - derecha
                    or matriz[i+1][j]==1 and matriz[i+1][j+1]==1 and matriz[i][j+1]==1
                    #solo abajo
                    or matriz[i+1][j]==1 and matriz[i+1][j+1]==1 and matriz[i+1][j-1]==1
                    ):
                        CoordenadasX_Sanas.append(j)
                        CoordenadasY_Sanas.append(i)
                #pared derecha
                elif i>0 and i <final and j==final :
                    if (matriz[i-1][j]==1 and matriz[i-1][j-1]==1 and matriz[i][j-1]==1
                    or matriz[i-1][j-1]==1 and matriz[i][j-1]==1 and matriz[i+1][j-1]==1
                    or matriz[i][j-1]==1 and matriz[i+1][j-1]==1 and matriz[i+1][j]==1):
                        CoordenadasX_Sanas.append(j)
                        CoordenadasY_Sanas.append(i)
                #pared inferior
                elif i==final and j>0 and j<final:
```

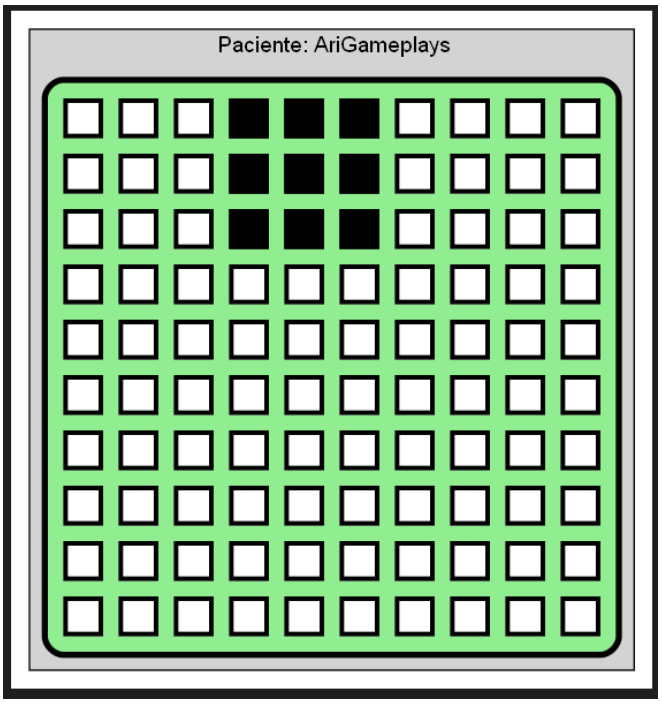
```
            elif i==final and j>0 and j<final:
                if (matriz[i][j-1]==1 and matriz[i-1][j-1]==1 and matriz[i-1][j]==1

                or matriz[i-1][j]==1 and matriz[i-1][j+1]==1 and matriz[i][j+1]==1
                #solo abajo
                or matriz[i-1][j]==1 and matriz[i-1][j+1]==1 and matriz[i-1][j-1]==1
                ):
                    CoordenadasX_Sanas.append(j)
                    CoordenadasY_Sanas.append(i)
            else:
                # izquierda - izquierda abajo   abajo
                if (matriz[i][j-1]==1 and matriz[i+1][j-1]==1 and matriz[i+1][j]==1):
                    CoordenadasX_Sanas.append(j)
                    CoordenadasY_Sanas.append(i)
                #   abajo - abajo derecha - derecha
                elif matriz[i+1][j]==1 and matriz[i+1][j+1]==1 and matriz[i][j+1]==1:
                    CoordenadasX_Sanas.append(j)
                    CoordenadasY_Sanas.append(i)
                # izquierda - izquierda arriba - arriba
                elif matriz[i][j-1]==1 and matriz[i-1][j-1]==1 and matriz[i-1][j]==1 :
                    CoordenadasX_Sanas.append(j)
                    CoordenadasY_Sanas.append(i)
                #arriba - arriba derecha - derecha
                elif matriz[i-1][j]==1 and matriz[i-1][j+1]==1 and matriz[i][j+1]==1:
                    CoordenadasX_Sanas.append(j)
                    CoordenadasY_Sanas.append(i)
                #todo arriba
                elif matriz[i-1][j]==1 and matriz[i-1][j+1]==1 and matriz[i-1][j-1]==1:
                    CoordenadasX_Sanas.append(j)
                    CoordenadasY_Sanas.append(i)

                    #todo abajo

            #todo abajo
            elif matriz[i+1][j]==1 and matriz[i+1][j+1]==1 and matriz[i+1][j-1]==1:
                CoordenadasX_Sanas.append(j)
                CoordenadasY_Sanas.append(i)

            #todo izquierda
            elif matriz[i-1][j-1]==1 and matriz[i][j-1]==1 and matriz[i+1][j-1]==1:
                CoordenadasX_Sanas.append(j)
                CoordenadasY_Sanas.append(i)

            #todo derecha

            elif matriz[i-1][j+1]==1 and matriz[i][j+1]==1 and matriz[i+1][j+1]==1:
                CoordenadasX_Sanas.append(j)
                CoordenadasY_Sanas.append(i)

        #CONTAGIADAS ------------------------------------------------------------------------------
        elif matriz[i][j]==1:
            if i == 0 and j == 0:
                if matriz[i+1][j]==0 and matriz[i+1][j+1]==0 or matriz[i][j+1]==0 and matriz[i+1][j+1]==0:
                    matriz[i][j]=0

            elif i ==final and j==0:
                if matriz[i-1][j]==0 and matriz[i-1][j+1]==0 or matriz[i][j+1]==0 and matriz[i-1][i+1]:
                    matriz[i][j]=0

            elif i ==0 and j==final:
                if matriz[i+1][j]==0 and matriz[i+1][j-1]==0 or matriz[i][j-1]==0 and matriz[i+1][j-1]==0:
                    matriz[i][j]=0
```

Dando como resultado la nueva matriz de células del paciente.

Estado inicial del paciente.



Estado del paciente después de un periodo de investigación.



Para la creación de graficas de utilizó el siguiente método haciendo uso de graphviz.

```
import graphviz
class Graficar:
    def __init__(self,matriz,paciente):
        self.matriz=matriz
        self.paciente=paciente
        self.n=""
        self.Crear()

    def Crear(self):
        #Ruta=self.Calcular()
        RutaPng="Gráfica del paciente "+str(self.paciente)

        text="""digraph L{
    node[shape=plain fontname=\"Arial\"]

    subgraph cluster1{
        label = \" Paciente: """+str(self.paciente)+"""\"
        fontname=\"Arial\"
        foreground=\"15\"
        bgcolor = lightgrey
        a0 [label=<
        <TABLE border=\"4\" cellspacing=\"10\" cellpadding=\"10\" style=\"rounded\" bgcolor=\"lightgreen\">"""
```

```
            matriz=self.matriz

            for i in range(len(matriz)):
                text+="\n<TR>"
                for j in range(len(matriz[0])):
                    color=""
                    if int(matriz[i][j])==0:
                        color="white"
                    else:
                        color="black"
                    text+="<TD border=\"3\"  bgcolor=\""+str(color)+"\"></TD>"
                text+="</TR>"
            text+="""
                    </TABLE>>];
        }
    }
```

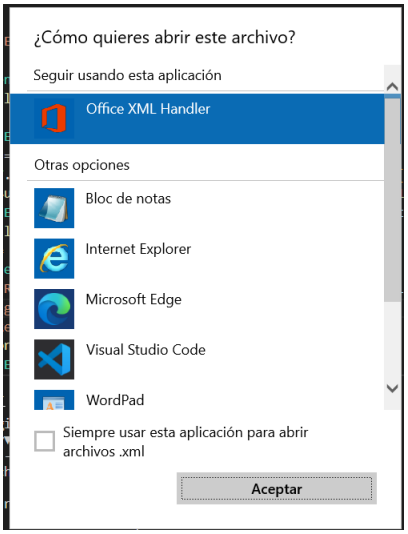
Una vez realizadas las investigaciones nos podremos dirigir al menú principal para así poder generar nuestro reporte.

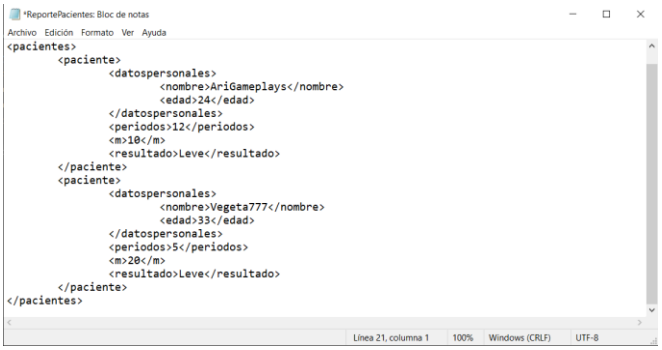
Para la creación del reporte se realizó el siguiente método.

```
def Generar():
    root = ET.Element("pacientes")
    for i in range(len(Lista_Reporte)):

        paciente = ET.SubElement(root,"paciente")
        personales = ET.SubElement(paciente,"datospersonales")
        nombre = ET.SubElement(personales,"nombre").text=Lista_Reporte[i][0]
        edad = ET.SubElement(personales,"edad").text=Lista_Reporte[i][1]
        periodos = ET.SubElement(paciente,"periodos").text=Lista_Reporte[i][2]
        m = ET.SubElement(paciente,"m").text=Lista_Reporte[i][3]
        resultado = ET.SubElement(paciente,"resultado").text="Leve"
    archivo = ET.ElementTree(root)
    archivo.write("ReportePacientes.xml",encoding="utf-8",xml_declaration=True)
    os.startfile("ReportePacientes.xml")
def Agregar_al_Reporte(nombre,edad,periodos,m):
    ListaAux= ListaEn()
    ListaAux.append(nombre)
    ListaAux.append(edad)
    ListaAux.append(periodos)
    ListaAux.append(m)
    ListaAux.append("Leve")
    Lista_Reporte.append(ListaAux)
```

Una vez generado el reporte automáticamente se nos abrirá una ventana para escoger que app se desea utilizar para visualizar el archivo xml, si no se tiene una predefinida.

En este caso al ser un archivo que puede ser visualizado con muchas aplicaciones, se utilizó bloc de notas para la demostración





Dando, así como finalizado el programa.

Conclusiones

La utilización de listas enlazadas creadas desde cero, son más funcionales que las listas por defecto de un lenguaje de programación.
La optimización con el uso de listas es complicada si no se usa una lista de tipo de pila

Referencias bibliográficas

MÉNDEZ, Mariano. 75.41 Algoritmos y Programación II Tda Lista y sus Derivados.

OJEDA, Luis Roberto. Tda Programacion Orientado a Objetos en Turbo. Univ. Nacional de Colombia.