|  |
| --- |
| **Proyecto 1** |
| **202109567 – Alvaro Norberto García Meza** |

**Resumen**

Dicho proyecto fue propuesto con la finalidad de crear una aplicación a petición del laboratorio de investigación epidemiologia de Guatemala, para analizar la forma en que las enfermedades infectan las células del cuerpo humano y se expanden produciendo enfermedades graves e incluso la muerte.

Los científicos proveen una rejilla con un numero el cual representa las dimensiones de una matriz cuadrada y que esta compuesta por filas y columnas, donde cada celda puede o no estar infectada. Mediante un patrón estudiado se observa en un determinado número de periodos el desarrollo de esta rejilla para posteriormente devolver un diagnostico que determine la gravedad de la enfermedad y otras señales tales como: en qué periodo este patrón de células infectadas vuelve a repetirse y cada cuanto se en cicla dicho periodo si es que contiene.

**Palabras clave**

Rejilla, periodo, infectada, patrón y diagnóstico.

***Abstract***

*This project was proposed with the aim of creating an application, at the request of the Guatemalan Epidemiological Research Laboratory, to analyze how diseases infect the cells of the human body and spread, causing serious illness and even death.*

*The scientists provide a grid with a number that represents the size of a square matrix and is made up of rows and columns, in which each cell may or may not be infected. Through a studied pattern, the development of this grid is observed over a series of periods to later give a diagnosis that determines the severity of the disease and other signals such as: in which period this pattern of infected cells is repeated and how often this period occurs in cycle if it contains it.*

***Keywords***

*Grid, period, infected, pattern and diagnostic*

**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 contenía los pacientes a analizar con sus respectivos atributos tales como: nombre, edad, periodo, tamaño de la rejilla y apartado con las células especificas infectadas donde al leerlo y evaluarlo retornaba un mismo formato XML con algunos atributos diferentes tales como resultado, n y n1.

Para simular el comportamiento de las células en una matriz utilizando una lista simple enlazada, se implemento el concepto de row mayor, el cual con un simple cálculo matemático se podía obtener la posición de dicha célula en una matriz cuadrada.

Por último, se implementó la herramienta grahpviz la cual generaba una imagen del ultimo periodo ejecutado.

**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.

1. **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.

* **App:** El primero directorio de la jerarquía el cual contiene toda la aplicación
  + **Classes:** Es el primer directorio el cual con tiene los módulos los cuales son todas las clases utilizadas para el desarrollo del programa.
    - **Cell.py**
    - **List.py**
    - **Node.py**
    - **Patient.py**
  + **Components:** El segundo directorio contiene los componentes los cuales le dan la funcionalidad y lógica a la aplicación.
    - **Select\_and\_execute\_patient.py**
  + **App\_Consola.py:** Es el modulo principal donde todas las demás clases y módulos se conectan para crear la aplicación.

Pantalla de computadora con letras

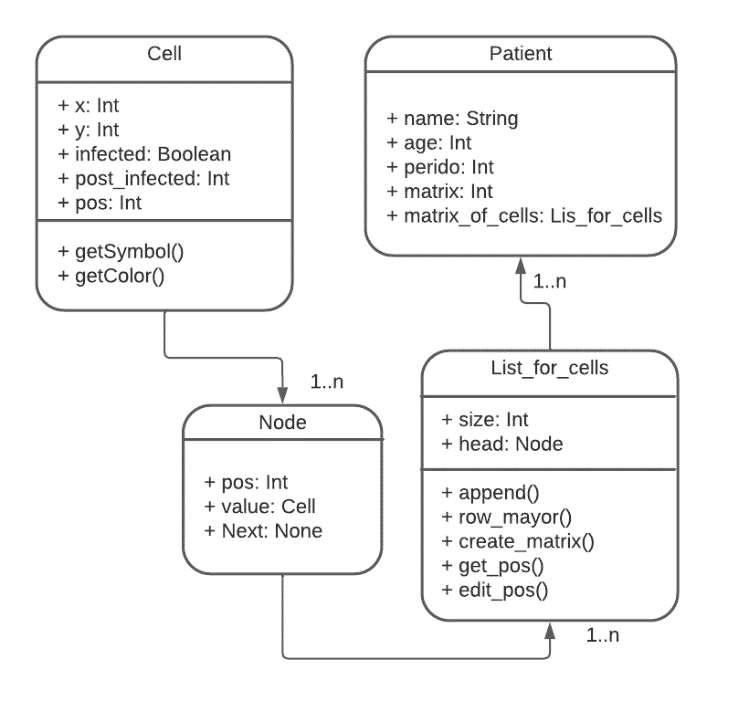
Descripción generada automáticamente con confianza media

*Figura 1.* Directorios y Módulos.

Fuente: elaboración propia, 2022.

1. **Clases**

Al implementar distintos paradigmas de programación como Programación Orientada a Objetos, se implementó distintas clases para almacenar pacientes, células y el uso de una clase nodo la cual se complementaba con la clase la cual era una lista simple enlazada.



*Figura 2.* Diagrama de clases.

Fuente: elaboración propia, 2022.

*Figura 1.* Título o descripción breve de la figura.

Fuente: elaboración propia, o citar al autor, año y página.

1. **Lista simple enlazada**

Para la simulación de una matriz utilizando listas simples la manera más sencilla de realizarlo es utilizando lo que se conoce como row\_mayor el cual con un simple cálculo matemático el retorna lo que es la suma de la posición de la fila y la multiplicación de la columna por el tamaño de la matriz. Con esta simple función se sabe en que posición de la matriz esta la célula. Además, de al ser una lista simple se debe manejar la posición de cada objeto con el uso de nodos el cual apunta al siguiente.

Captura de pantalla de computadora

Descripción generada automáticamente

*Figura 3.* Lista simple enlazada.

Fuente: elaboración propia, 2022.

1. **Módulos**

La aplicación esta divida en dos módulos, el primero contiene el desarrollo de todo el menú principal, donde se puede cagar archivos, ver los pacientes y salir y el segundo módulo contiene toda la funcionalidad lógica que el programa demanda tal como: escoger un paciente, evaluar un paciente, generar un gráfico de los resultados y a su vez devolver un diagnostico del paciente seleccionado. Además, tiene la posibilidad de terminar de evaluar un paciente y regresar al menú para salir.

**Modulo: App\_Consola.py**

Como primer punto en la figura 4. Es la recopilación de funciones las cuales juntas generan un menú sencillo con 3 opciones a elegir: cargar archivo, ver paciente y salir.

Al momento de seleccionar cargar un archivo la función de la figura 5 se ejecuta, lee el XML y convierte a cada paciente en un objeto, luego se almacenan en una lista nativa las células infectadas proporcionadas, posteriormente se crea el objeto de lista simple donde almacena el valor de las célula contagiadas y no contagiadas con su respectiva posición en memoria y simulacro de rejilla.

**Texto

Descripción generada automáticamente**

*Figura 4.* Menú principal.

Fuente: elaboración propia, 2022

**Texto

Descripción generada automáticamente**

*Figura 5.* Función upload\_file.

Fuente: elaboración propia, 2022

**Modulo: select\_and\_execute\_patient.py**

Este modulo es uno de lo que más funciones contiene ya que es lo que le da la funcionalidad total a la aplicación. Como primer punto en la figura 6 genera un submenú para escoger los usuarios cargados y evaluarlos conforme los periodos.

Desde la figura 7.1 hasta 7.4 ejecuta cada matriz del paciente seleccionado, guiándose por el patrón ya establecido para monitorear el comportamiento de las células con el pasar del tiempo.

En la figura 8 se almacenan 2 funciones sencillas que se complementan con las de la figura 7 las cuales tienen la finalidad de contar la cantidad de celdas vecinas infectadas de una célula y retornar un valor e imprimir en consola cada periodo seleccionado de la matriz respectivamente.

Por consiguiente, la figura 9 genera un diagnostico y este usará la información del paciente seleccionado y el ultimo periodo ejecutado para generar un formato XML con el resultado.

Por último, en la figura 10 contiene la función que genera una imagen png con listas simples en forma de matriz simulando el ultimo periodo ejecutado.

**Texto

Descripción generada automáticamente**

*Figura 6.* Función: select\_patient.

Fuente: elaboración propia, 2022

**Texto

Descripción generada automáticamente**

*Figura 7.1.* Función execute\_patient.

Fuente: elaboración propia, 2022

Texto

Descripción generada automáticamente

*Figura 7.2.* Función execute\_patient.

Fuente: elaboración propia, 2022

Texto

Descripción generada automáticamente

*Figura 7.3.* Función execute\_patient.

Fuente: elaboración propia, 2022

Texto

Descripción generada automáticamente

*Figura 7.4.* Función execute\_patient.

Fuente: elaboración propia, 2022

****

*Figura 8.* Función verified y print\_matrix.

Fuente: elaboración propia, 2022

**Texto

Descripción generada automáticamente**

*Figura 9.* Función xml\_response.

Fuente: elaboración propia, 2022

Texto

Descripción generada automáticamente

*Figura 10.* Función print\_grahpviz.

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 como 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**

**Texto

Descripción generada automáticamente**

*Figura 11.* Documento de entrada en XML.

Fuente: elaboración propia, 2022

**Texto

Descripción generada automáticamente**

*Figura 12.* librerías utilizadas.

Fuente: elaboración propia, 2022