Universidad de San Carlos de Guatemala
Escuela de Ingeniería en Ciencias y Sistemas, Facultad de Ingeniería
Introducción a la programación y computación 2, 1er. Semestre 2025

PROYECTO 1

201942079 – Luz de María José Castillo Flores

Resumen

El manejo y análisis eficiente de datos experimentales requiere soluciones de software robustas y escalables. Este proyecto, desarrollado en Python, presenta un sistema integral que integra técnicas avanzadas de ingeniería de software con estructuras de datos dinámicas y visualización gráfica. En el núcleo de su diseño se aplican principios de POO, permitiendo una arquitectura modular que simplifica el mantenimiento y facilita futuras mejoras. El sistema emplea archivos XML para la entrada estandarizada de datos, garantizando que los detalles experimentales se procesen con precisión. Además, se utilizan listas enlazadas simples como estructura dinámica, optimizando el almacenamiento y la manipulación de información sin las limitaciones de estructuras estáticas. Para mejorar la comprensión de los resultados, se integra Graphviz, que genera visualizaciones interactivas de las transformaciones de datos. Aunque inicialmente motivado por aplicaciones en investigación oncológica, para realización de fármacos. La principal contribución del proyecto es su arquitectura de software, que automatiza procesos, optimiza datos y mejora el análisis científico.

Palabras clave

Cáncer, software, POO, XML, estructuras de datos.

Abstract

Efficient management and analysis of experimental data require robust and scalable software solutions. This project, developed in Python, presents a comprehensive system that integrates advanced software engineering techniques with dynamic data structures and graphical visualization. At its core, object-oriented programming principles are applied, enabling a modular architecture that simplifies maintenance and facilitates future enhancements. The system employs XML files for standardized data input, ensuring that experimental details are processed precisely. Additionally, singly linked lists are used as a dynamic data structure, optimizing the storage and manipulation of information without the limitations of static structures. To enhance the understanding of results, Graphviz is integrated, generating interactive visualizations of data transformations. Although initially motivated by applications in oncological research for drug development, the primary contribution of the project is its software architecture, which automates processes, optimizes data, and improves scientific analysis.

Keywords

Cancer, software, OOP, XML, data structures.

Introducción

En la era digital actual, el procesamiento y la automatización de datos experimentales son fundamentales para el avance de la investigación científica. Este proyecto, desarrollado en Python, aborda estos desafíos mediante la creación de un sistema de software robusto y escalable. La herramienta se basa en un diseño modular apoyado en la programación orientada a objetos, lo que permite estructurar el código en componentes independientes y especializados.

Mediante el uso de archivos XML, el sistema estandariza y organiza la información experimental, facilitando su carga y procesamiento. Asimismo, la implementación de listas enlazadas simples como estructura de datos dinámica optimiza la manipulación y el acceso a la información sin las limitaciones de estructuras estáticas. Complementariamente, la integración de Graphviz posibilita la generación de visualizaciones gráficas interactivas, que son cruciales para analizar y validar los resultados de los experimentos.

Aunque el contexto original del proyecto se relaciona con aplicaciones oncológicas, el enfoque principal reside en la innovación y eficiencia del software, destacando la importancia de un diseño orientado a objetos, el manejo avanzado de datos y la capacidad de visualización para mejorar la interpretación y el análisis de procesos experimentales.

Desarrollo del tema

Python como Lenguaje:

Python ha sido la base fundamental para la implementación de este proyecto, gracias a su flexibilidad, legibilidad y amplia variedad de bibliotecas integradas. Su sintaxis clara y estructurada ha permitido desarrollar un código organizado, siguiendo buenas prácticas de programación como la indentación adecuada y convenciones de nomenclatura, lo que facilita su comprensión y mantenimiento.

Además, el uso de bibliotecas estándar de Python ha optimizado la implementación del sistema sin necesidad de recurrir a librerías externas. Módulos como xml.etree.ElementTree han permitido el procesamiento eficiente de archivos XML, facilitando la gestión de los datos experimentales. A su vez, subprocess ha sido clave para la ejecución de comandos externos, en particular para la generación de gráficos con Graphviz, mientras que tkinter ha brindado una solución práctica para la selección de archivos a través de una interfaz gráfica.

El aprovechamiento de estas herramientas nativas ha garantizado una implementación más sencilla y una mayor compatibilidad con distintos entornos, asegurando la estabilidad y eficiencia del sistema. Python no solo ha permitido un desarrollo ágil, sino que también ha proporcionado un marco robusto para la manipulación de datos y la automatización de procesos dentro del experimento.

Desarrollo del tema Implementación de Programación Orientada a Objetos (POO):

La Programación Orientada a Objetos (POO) es un enfoque fundamental en el desarrollo de software moderno, ya que permite estructurar el código en torno a entidades llamadas objetos, los cuales encapsulan tanto datos como comportamientos. Esta metodología facilita la organización del sistema, promoviendo la reutilización del código y la separación de responsabilidades, aspectos clave en el diseño de aplicaciones escalables y mantenibles.

En el desarrollo de este proyecto, se implementaron diversas clases para modelar los diferentes componentes del sistema. La representación de las proteínas se logró mediante estructuras que encapsulan su información y establecen una relación jerárquica entre ellas. De igual manera, la gestión de los experimentos se realizó a través de clases específicas, responsables de almacenar y manipular los datos esenciales, como el nombre del experimento, las dimensiones de la rejilla y las parejas de proteínas involucradas. Además, se integraron mecanismos para la lectura y procesamiento de archivos XML, permitiendo automatizar la ejecución de los experimentos y mejorar la eficiencia del sistema.

Estructura de Datos Abstractas (TDA) en la gestión del sistema:

El uso de estructuras de datos abstractas (TDA) es un pilar fundamental en la implementación de este sistema, permitiendo la organización eficiente de la información y la optimización de los procesos. En este contexto, se han utilizado listas enlazadas simples para modelar dos aspectos clave del proyecto: la rejilla de proteínas y la gestión de los experimentos.

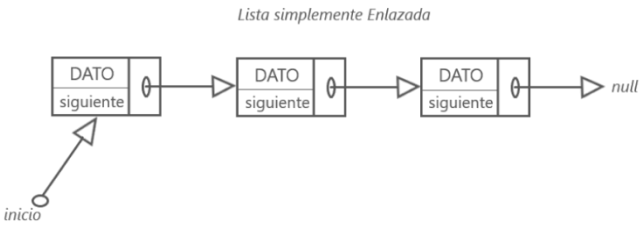

Figura 1. Estructura de una lista simplemente enlazada.

Fuente: elaboración propia.

Por un lado, la rejilla de proteínas se implementa mediante una lista enlazada simple, donde cada nodo, representado por la clase NodoProteina, almacena el valor de la proteína, una bandera que indica si es inerte y referencias tanto al siguiente nodo en la fila como al nodo de la fila inferior. Esta estructura permite representar la disposición bidimensional de la rejilla sin necesidad de utilizar matrices convencionales, lo que facilita la manipulación dinámica de los datos. La clase ListaProteina gestiona la estructura, permitiendo la carga de la rejilla a partir de un texto, la búsqueda de parejas de proteínas y el cálculo del porcentaje de nodos inertes. Gracias al uso de punteros, la rejilla puede recorrerse tanto de forma horizontal como vertical, proporcionando flexibilidad en el procesamiento de la información.

Además, para la gestión de los experimentos, se ha implementado otra lista enlazada simple, compuesta por las clases NodoExperimento y ListaExperimento. NodoExperimento almacena información clave de cada experimento, como su nombre, dimensiones de la rejilla y las parejas de proteínas involucradas. ListaExperimento, por su parte, permite manipular la colección de experimentos mediante métodos específicos para insertar, buscar y modificar registros, asegurando un acceso eficiente a la información almacenada.

El uso de listas enlazadas en ambas estructuras permite un manejo dinámico de los datos, sin las restricciones de tamaño de las estructuras estáticas. Además, la implementación de estos TDAs favorece la abstracción en la manipulación de la información, ocultando los detalles internos y proporcionando una interfaz clara y accesible para la interacción con los datos. Este modularidad no solo mejora la organización del código, sino que también facilita futuras modificaciones y extensiones del sistema, asegurando su escalabilidad y mantenibilidad.

Uso de Graphviz para Visualización:

La representación gráfica de los datos es un aspecto fundamental para comprender la evolución del experimento y analizar sus resultados de manera efectiva. En este sentido, la integración de Graphviz ha permitido generar visualizaciones claras y estructuradas, facilitando la interpretación del comportamiento del sistema.

Graphviz es una herramienta de código abierto utilizada para la generación de diagramas y representaciones gráficas a partir de descripciones en texto mediante el lenguaje DOT. Su capacidad para visualizar estructuras de datos, redes y relaciones complejas lo hace ampliamente utilizado en campos como la ingeniería de software, la bioinformática y la gestión de bases de datos. Además, permite exportar gráficos en formatos como SVG y PNG, proporcionando una solución versátil para el análisis y la interpretación visual de la información.

Aprovechando estas capacidades, el sistema genera archivos en formato DOT, que describen la rejilla experimental utilizando etiquetas HTML. En cada celda de la tabla se almacena la proteína correspondiente y, en el caso de los nodos inertes, se aplica un cambio de color de fondo para resaltar su estado. Esta metodología permite representar visualmente la transformación de la rejilla de manera organizada y comprensible.

Posteriormente, estos archivos DOT se procesan utilizando Graphviz, que convierte la descripción textual en gráficos en formato SVG. A través del módulo subprocess de Python, el sistema ejecuta automáticamente la conversión, permitiendo obtener representaciones gráficas tanto del estado inicial como del estado final del experimento.

Estado: 1

	1	2	3	4
1	AGVST	LEGAG	KRRRL	MNPQR
2	TUVWX	YZABC	DEFGH	IJKLM
3	NOPQR	STUVW	XYZA1	23456

Figura 2. Ejemplo de gráficos generados por Graphviz.

Fuente: elaboración propia.

Además, la generación dinámica de gráficos en el modo paso a paso proporciona una visión detallada de cómo evoluciona la interacción entre las proteínas y la propagación de los efectos en la rejilla. Esta funcionalidad no solo contribuye a la comprensión del proceso experimental, sino que también se convierte en una herramienta clave para la validación de los resultados y la identificación de patrones en la reacción de las proteínas.

La incorporación de Graphviz en el sistema no solo mejora la visualización de los datos, sino que también fortalece el análisis del experimento, proporcionando una representación estructurada que facilita la interpretación y evaluación de los resultados obtenidos.

Procesamiento de XML para gestión de datos:

La estructuración y el manejo eficiente de los datos son esenciales para el desarrollo de un sistema robusto y escalable. En este contexto, el formato XML ha sido utilizado como el medio principal para la entrada y organización de la información relacionada con los experimentos. Su flexibilidad y estructura jerárquica permiten representar de manera clara los distintos elementos involucrados en cada experimento, garantizando un procesamiento eficiente de los datos.

El sistema emplea la biblioteca xml.etree.ElementTree para la extracción y análisis del contenido de los archivos XML. A través de este módulo, se cargan y procesan los datos esenciales, como el nombre del experimento, las dimensiones de la rejilla (filas y columnas), la distribución de las proteínas y las combinaciones de pares que participan en la reacción. Esta metodología permite convertir la información almacenada en el archivo en una estructura manejable dentro del sistema.

```xml
<?xml version = "1.0" encoding = "UTF-8"?>
<experimentos>
    <experimento nombre = "Prueba01">
        <tejido filas = "5" columnas ="5">
            <rejilla>
                LAV LAV VAL VAL VAL
                VAR RAV GHI VAL VAL
                LAV AVL ALV LAV LAV
                VAL ILL KEQ DCC LAV
                VAR ILL KQA CDC LAV
            </rejilla>
        </tejido>
        <proteinas>
            <pareja>
                LAV VAL
            </pareja>
        </proteinas>
    </experimento>
</experimentos>
```

Figura 2. Ejemplo de archivo de entrada xml.

Fuente: elaboración propia.

Una vez extraídos, estos datos se mapean a objetos, instanciando clases como NodoExperimento y ListaExperimento, lo que facilita su manipulación mediante un enfoque orientado a objetos. Gracias a este modelo, se pueden aplicar métodos específicos para modificar, gestionar y ejecutar los experimentos sin necesidad de alterar la estructura de los datos en su origen. Este nivel de abstracción contribuye a una mayor organización del código y mejora la reutilización de los componentes del sistema.

Para garantizar la fiabilidad del procesamiento, se ha incorporado una validación y manejo de errores, evitando que un archivo XML mal estructurado o con información incompleta afecte el funcionamiento del sistema. Mediante estructuras de control y excepciones, se asegura que cualquier inconsistencia en el archivo sea identificada y gestionada adecuadamente, lo que fortalece la estabilidad y seguridad del programa.

El uso de XML no solo proporciona un formato estandarizado para la entrada de datos, sino que también facilita la integración y escalabilidad del sistema. Su flexibilidad permite la incorporación de nuevos experimentos sin necesidad de modificar la lógica interna del programa, favoreciendo la adaptabilidad del sistema a futuras mejoras y ampliaciones.

Conclusiones

El sistema desarrollado demuestra la eficacia de un diseño modular basado en programación orientada a objetos para gestionar y analizar datos experimentales. La implementación de listas enlazadas simples y el procesamiento de archivos XML permiten un manejo dinámico y eficiente de la información, mientras que la integración de Graphviz facilita la interpretación visual de los resultados. En conjunto, esta solución optimiza la automatización y el análisis de experimentos, ofreciendo una base sólida para futuras mejoras.

Referencias bibliográficas

- Guttag, J. V. (2016). Introduction to Computation and Programming Using Python. MIT Press.
- Python Software Foundation. (2021). Documentación oficial de Python: Tutorial de clases. Recuperado de https://docs.python.org/3/tutorial/classes.html
- Graphviz. (s.f.). Graphviz - Graph Visualization Software. Recuperado de https://graphviz.org/
- Geekflare. (2024). Comprender la implementación de listas enlazadas en Python. Recuperado de https://geekflare.com/es/python-linked-lists/
- W3C. (s.f.). Extensible Markup Language (XML) 1.0. Recuperado de https://www.w3.org/TR/xml/

Anexos

Universidad de San Carlos de Guatemala
Escuela de Ingeniería en Ciencias y Sistemas, Facultad de Ingeniería
Introducción a la programación y computación 2, 1er. Semestre 2025

Diagrama de clases:

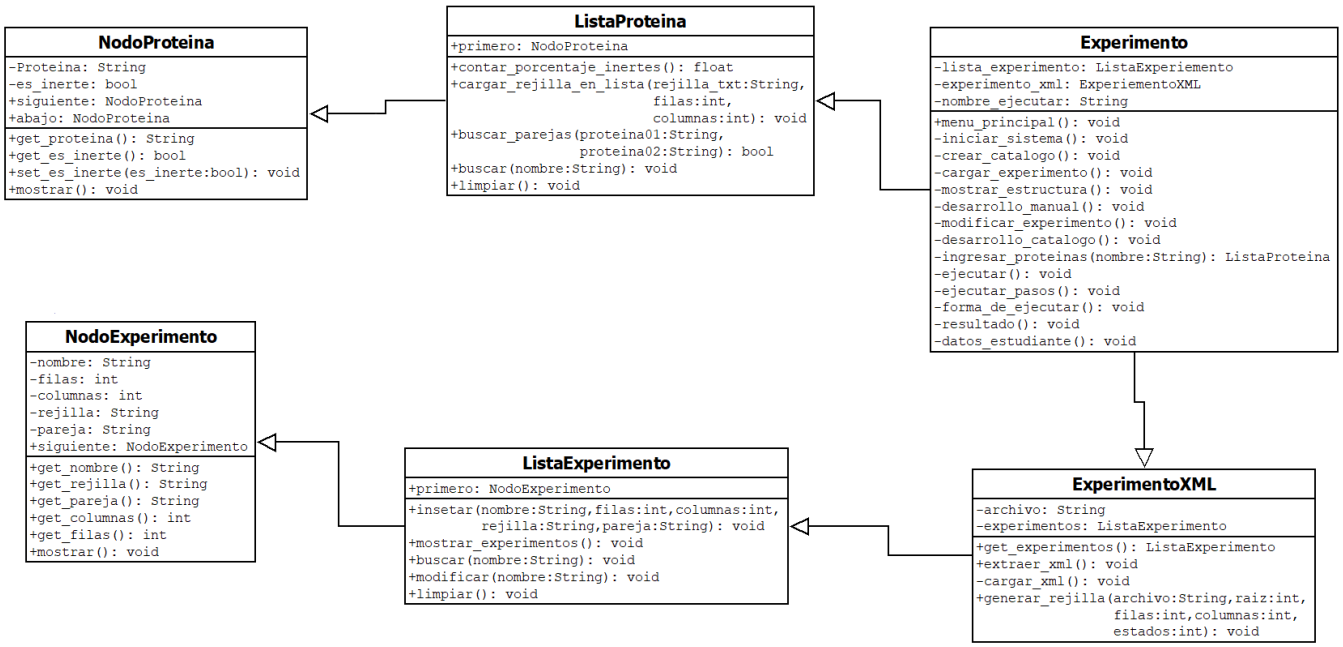

Diagrama de actividades: