
DIGITAL INTELLIGENCE ASSEMBLER – PROYECTO 2

201900131 – Elías Abraham Vasquez Soto

Resumen

Digital Intelligence Assembler es una aplicación desarrollada para la empresa Digital Intelligence, S. A., la cual simula el funcionamiento de una máquina desarrollada por la mencionada empresa, capaz de ensamblar las partes de cualquier producto, con “n” líneas de ensamblaje y cada línea de ensamblaje con “m” posibles componentes a seleccionar de forma que pueda predecir el tiempo “óptimo” para elaborar cualquier producto que pueda ser ensamblado en la máquina.

La resolución del proyecto significa elevar la eficiencia de la máquina, ya que, al conocer los tiempos de ensamblaje óptimos, se puede realizar una mejor planificación del tiempo que la empresa dispone para el ensamblaje de productos.

Para la realización del programa, se desarrollaron conocimientos como la implementación de listas enlazadas utilizando el paradigma de programación orientado a objetos, la manipulación de archivos XML y la utilización de la herramienta Graphviz para la generación de las colas de secuencia, con el estado del ensamble en un segundo determinado.

Palabras clave

Archivo XML, clases, interfaz gráfica, listas, nodos.

Abstract

Digital Intelligence Assembler is an application developed for the company Digital Intelligence, S. A., which simulates the operation of a machine developed by the mentioned company, capable of assembling the parts of any product, with "n" assembly lines and each assembly line with "m" possible components to be selected so that it can predict the "optimal" time to produce any product that can be assembled on the machine.

The resolution of the project means increasing the efficiency of the machine, since, by knowing the optimal assembly times, a better planning of the time available to the company for the assembly of products can be carried out.

For the realization of the program, knowledge was developed such as the implementation of linked lists using the object-oriented programming paradigm, the manipulation of XML files and the use of the Graphviz tool for the generation of sequence queues, with the state of the assembly in a given second.

Keywords

XML file, classes, graphic interface, lists, nodes.

Introducción

El presente proyecto se centra en la creación de un programa, que sea capaz de cargar distintas máquinas, entregadas en formato XML, para determinar el tiempo óptimo que le toma a la máquina ensamblar cierto producto. Además, es posible cargar archivos de simulación en formato XML, si lo que se desea es calcular el tiempo óptimo de ensamblaje de múltiples productos.

Para el desarrollo del programa, se utilizó el lenguaje de Python, el cual soporta el paradigma de programación orientado a objetos. Dicho paradigma se utilizó para la creación de diversas clases, cuyas instancias, representan las máquinas, productos, componentes, etc., así como las listas enlazadas y nodos utilizados para almacenar dichos objetos de manera dinámica.

En este ensayo, se busca familiarizar al lector con las clases y objetos propios del POO, así como de su implementación en la creación de las diversas listas enlazadas y nodos, todo esto para llegar a la resolución del problema principal planteado.

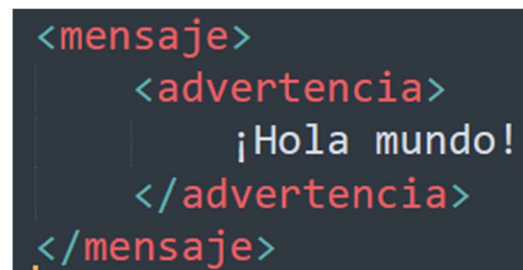
Desarrollo del tema

Para el desarrollo del programa, fue necesaria la aplicación de múltiples conocimientos en diversos temas, tales como manejo de archivos XML, manejo del paradigma de programación orientado a objetos, entre otros. A continuación, se detallan los conocimientos adquiridos y aplicados en este programa, además de la lógica implementada para ciertos métodos propios de las clases creadas.

Estos conocimientos son:

a. XML - ElementTree

XML es un lenguaje de marcado similar a HTML. Significa Extensible Markup Language (Lenguaje de Marcado Extensible). Esto significa que, a diferencia de otros lenguajes de marcado, XML no está predefinido, por lo que debes definir tus propias etiquetas. El propósito principal del lenguaje es compartir datos a través de diferentes sistemas, como Internet.



```
<mensaje>
  <advertencia>
    ¡Hola mundo!
  </advertencia>
</mensaje>
```

Figura 1. Ejemplo práctico XML.

Fuente: Elaboración propia.

XML posee diversas características:

- Está compuesto por múltiples pares de etiquetas
- Las etiquetas pueden tener atributos
- Los datos correspondientes a la etiqueta se colocan en el medio del par de etiquetas.
- Pueden anidarse etiquetas dentro de otras

Element Tree, el cual es un módulo de Python, es un procesador XML simple y liviano para crear, analizar y procesar datos XML. Las clases diseñadas en Element Tree son:

- ElementTree: Representa toda la jerarquía XML.
- Element: Representa todos los nodos principales en la estructura de árbol.

- SubElement: Representa todos los nodos secundarios en la estructura de árbol.

La manera en que maneja Element Tree las distintas características en un archivo XML, y que se usaron en este proyecto son:

Tabla I.

Manejo de XML con Element Tree.

CARACTERÍSTICA	ATRIBUTO
Nombre de la etiqueta	tag
Atributos	attrib
Valor de la etiqueta	text
Y la próxima etiqueta	tail

Fuente: Elaboración propia.

b. Paradigma de programación orientada a objetos

El paradigma de programación utilizado para la resolución de este problema fue el orientado a objetos. Este facilita el desarrollo, al facilitar la reutilización, creando la cantidad de objetos que sean necesarios sin límite, de manera relativamente sencilla. Para comprender este paradigma, se deben hablar de dos componentes importantes del mismo, los objetos y las clases.

Un objeto posee identidad (un identificador único), estado (propiedades o atributos) y comportamiento (un conjunto de operaciones o métodos). A lo largo del presente ensayo, se detallan la identidad, el estado y el comportamiento de los distintos tipos de objetos programados.

Una clase es la representación de la estructura y comportamiento de un objeto. Es un patrón para la definición de atributos y métodos para un tipo particular de objetos. Una característica de las clases es que todos los objetos que provengan de una clase dada son idénticos en estructura y comportamiento,

pero son únicos, aunque posean los mismos valores en sus atributos.

Parafraseando lo anterior expuesto, una clase es un patrón o plantilla, de donde crearemos a todos nuestros objetos. Para la realización del programa, fue necesaria la creación de 22 clases, descritas posteriormente en el diagrama de clases.

Existen 4 pilares de la programación orientada a objetos: la abstracción, el encapsulamiento, la herencia y el polimorfismo. Para el programa desarrollado, no fue necesario implementar la herencia ni el polimorfismo, por lo que todas las instancias que se crearon, aunque algunas se encuentran anidadas dentro de otras, ninguna es “hija” de otra.

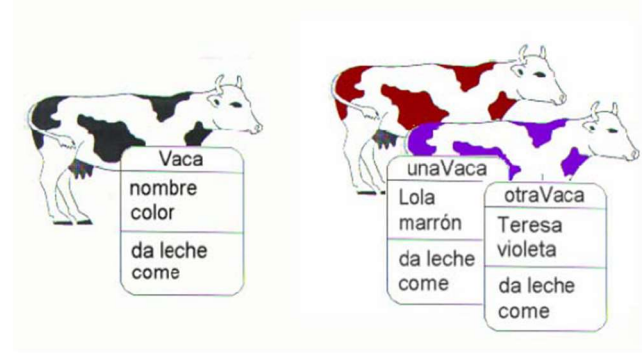


Figura 2. Ejemplo práctico de clases y objetos.

Fuente: José M. Ordax. *Principios de la Tecnología de Objetos.* 2004.

c. Lista enlazada

Una lista enlazada es una colección de elementos, denominados nodos. Cada nodo contiene dos componentes:

- Enlace o apuntador: contiene la dirección de memoria que apunta al próximo elemento.
- Datos: es la información que se necesita almacenar.
- Cabecera: Este elemento posee una referencia al primer elemento que se encuentra en la lista. Al instanciar una lista (crear un objeto

tipo lista enlazada), el constructor inicia automáticamente esta cabecera en nulo (None en python).

El orden de los nodos es determinado por la dirección de memoria donde se almacenan.

Las operaciones generales que se pueden realizar con este tipo de listas (porque existen varios tipos de listas con apuntadores), y que se utilizaron son:

- Insertar un dato a la lista

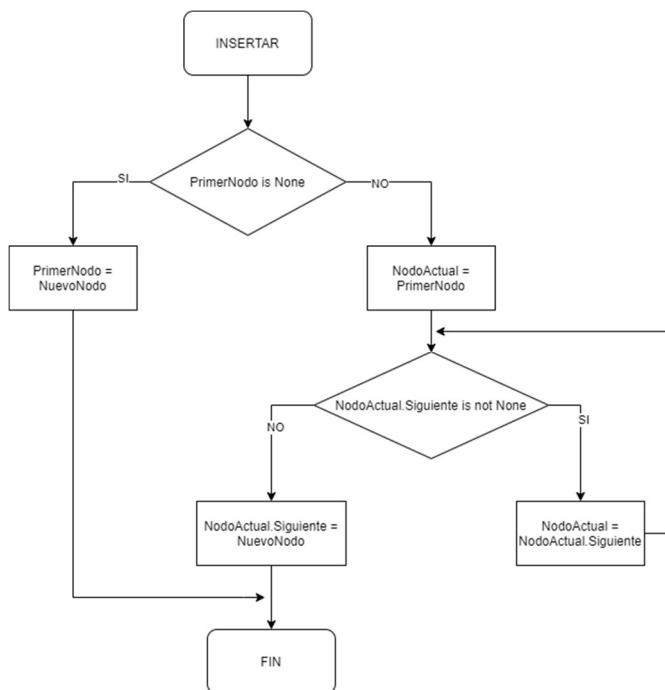


Figura 3. Diagrama de flujo, método insertar nodo.

Fuente: Elaboración propia

- Recorrer la lista

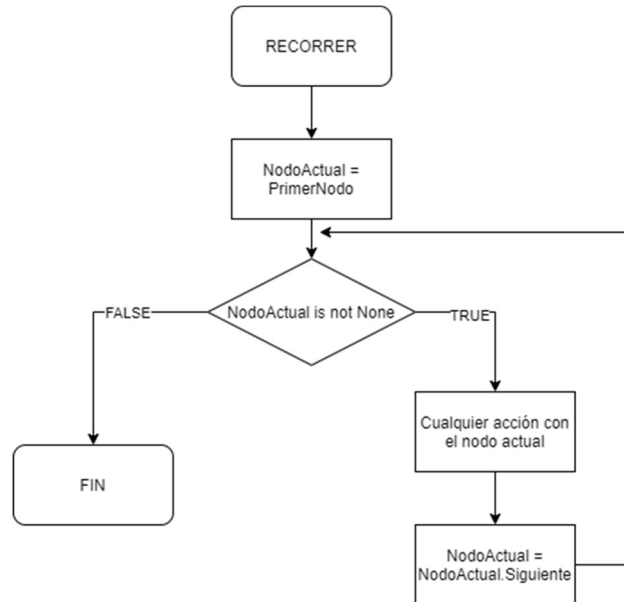


Figura 4. Diagrama de flujo, método recorrer lista

Fuente: Elaboración propia

- Buscar en elemento en específico de la lista

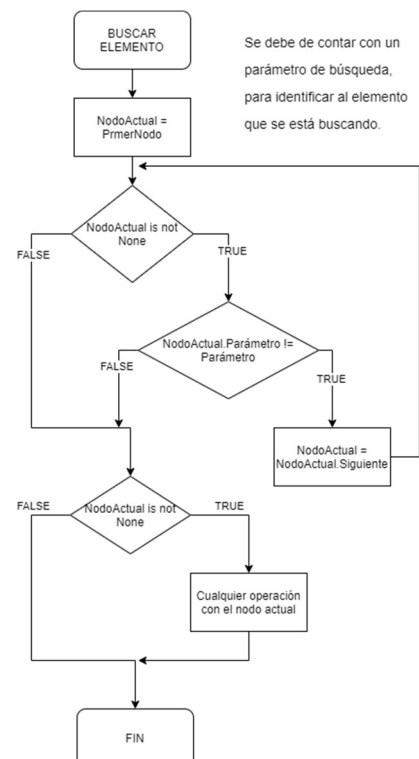


Figura 5. Diagrama de flujo, método buscar elemento

Fuente: Elaboración propia

- Eliminar un nodo de la lista

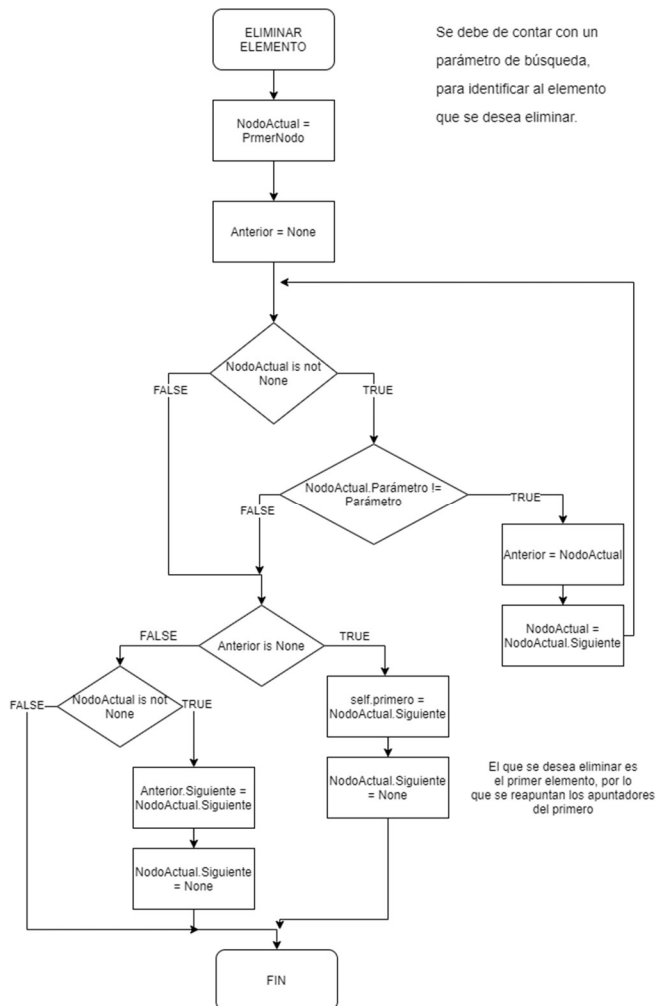


Figura 6. Diagrama de flujo, método eliminar nodo

Fuente: Elaboración propia

Se usaron listas para los siguientes casos:

- Lista de máquinas, en donde cada nodo guarda una máquina distinta.
- Lista de líneas por máquina, en donde cada nodo guarda una línea de ensamblaje.
- Lista de componentes por línea, en donde cada nodo guarda un componente.
- Lista de productos por máquina, en donde cada nodo guarda un producto en la máquina, el cual puede ser ensamblado.

- Lista de comandos por producto, en donde cada nodo guarda un comando de ensamble.
- Lista de acciones por producto, en donde cada nodo guarda una acción realizada por la máquina mientras se encuentra en proceso de ensamblaje.
- Lista de simulaciones, en donde cada nodo guarda una simulación o conjunto de nombres de productos que pueden ser ensamblados.
- Lista de nombres de productos por simulación, en donde cada nodo guarda un nombre de producto.

d. Interfaz gráfica de usuario con Tk

Tk, el cual es una parte integral de Python, proporciona un conjunto de herramientas robusto e independiente de la plataforma para administrar ventanas. Disponible para desarrolladores a través del paquete **tkinter** y sus extensiones, los módulos **tkinter.tix** y **tkinter.ttk**.

Para el desarrollo del presente proyecto y su interfaz gráfica, se utilizó el paquete **tkinter**. Este paquete es una capa delgada orientada a objetos encima de Tk. El módulo **tkinter** es un conjunto de funciones que envuelven a los distintos widgets Tk como clases de Python.

Las ventajas de los módulos **tkinter** son su velocidad y que generalmente se suministra de forma nativa con Python.

e. Software utilizado

El proyecto se realizó en lenguaje Python, en su versión más actual a la fecha de redacción de este

ensayo (3.9.7). El mismo se ejecuta por medio de una interfaz gráfica.

El IDE se dejó a criterio del desarrollador, por lo que se escogió Visual Studio Code versión 1.60.1, por la facilidad de personalización y múltiples extensiones que vuelven el desarrollo amigable a la vista.

Para las gráficas que se generan de las colas de secuencia, se utilizó la herramienta Graphviz, por lo que se recomienda que esta se encuentre instalada en la máquina del usuario, para garantizar el correcto funcionamiento de la aplicación.

Conclusiones

El manejo de archivos XML es un tema que se encuentra presente en el manejo de datos, principalmente en Internet, por lo que se vuelve de vital importancia, que se aprenda a procesar los datos de dichos archivos de manera correcta (tal aprendizaje se dio en el desarrollo del presente proyecto).

El uso de listas enlazadas, las cuales se llenan de forma dinámica, simplifica la resolución de problemas y aumenta la eficiencia del manejo de

memoria de la máquina. Esto va ligado al paradigma de programación orientado a objetos, que fue la base del desarrollo del programa.

Por último, se recalca la importancia de que el programador desarrolle la habilidad de diseñar interfaces gráficas amables e intuitivas al usuario, en donde se busca que la curva de aprendizaje en ella se nula, otorgándole al usuario la capacidad de dominar el programa al poco tiempo de interactuar con ella.

Referencias bibliográficas

Archivo de Python XML y procesamiento de archivos de configuración. Página web: <https://programmerclick.com/article/4079327273/>
D. Gilberto. *Listas Enlazadas.* Universidad de los Andes, Facultad de Ingeniería, Escuela de Sistemas.

O. José, (2004). *Conceptos básicos de la Orientación a Objetos.* Documentación de javaHispano.

O. José, (2004). *Paradigmas de la Orientación a Objetos.* Documentación de javaHispano.

Apéndices

Autómata implementado para reconocer los distintos componentes requeridos por producto.

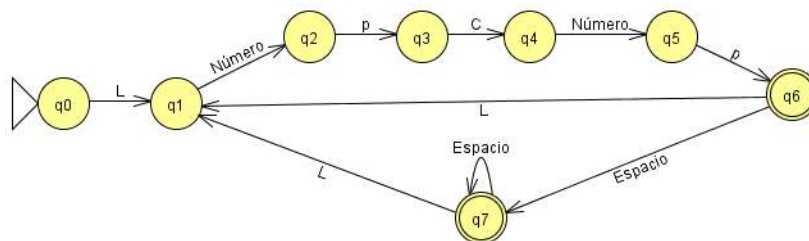


Figura 7. Autómata que reconoce comandos LnpCnp.

Fuente: Elaboración propia

Diagrama de clases: Lista global de máquinas.

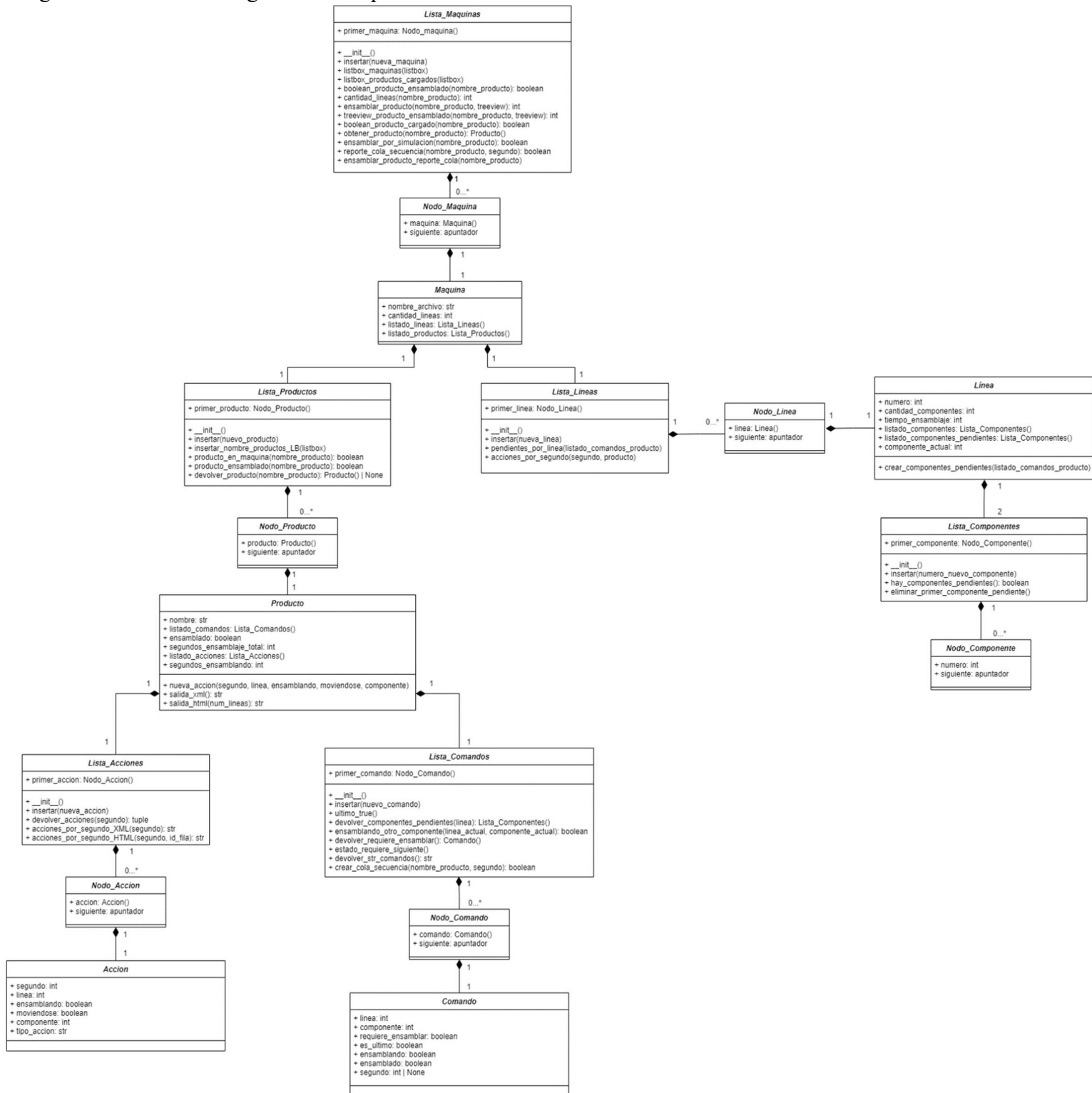


Figura 8. Diagrama de clases utilizadas para Lista global de máquinas.

Fuente: Elaboración propia

Diagrama de clases: Lista global de simulaciones.

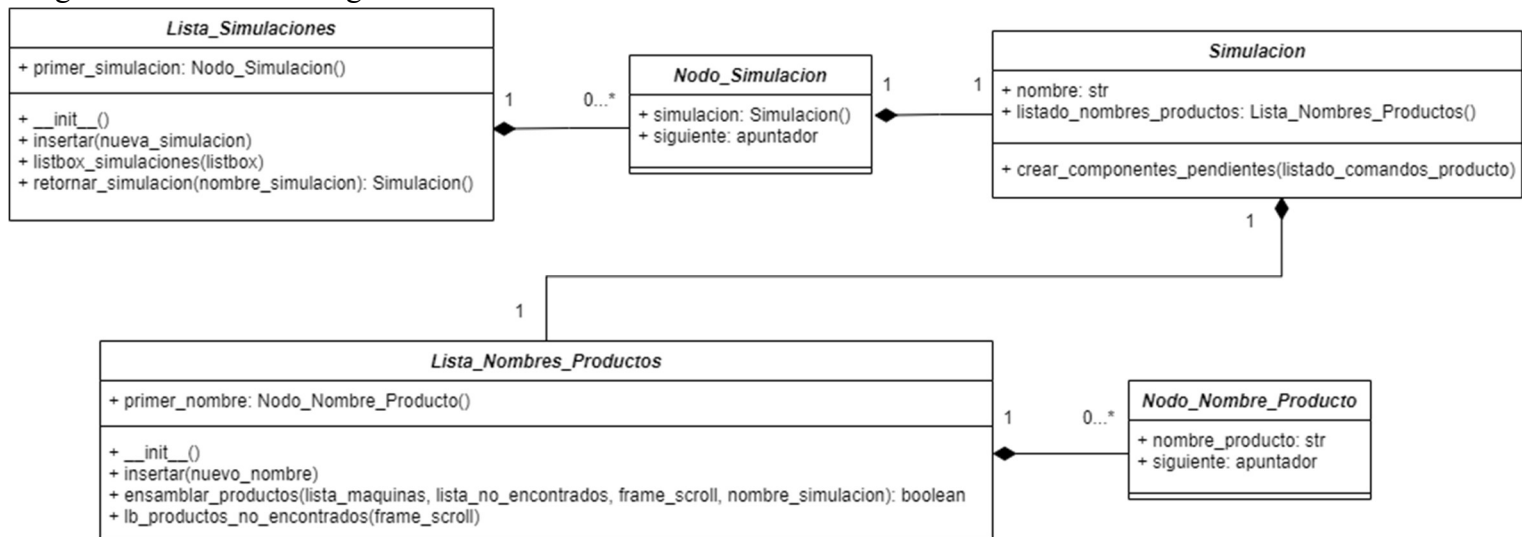


Figura 9. Diagrama de clases utilizadas para Lista global de simulaciones.

Fuente: Elaboración propia