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 2024

ENSAYO DE PROYECTO 2

202302220 – Enner Esaí Mendizabal Castro

Resumen

El proyecto 2 de Introducción a la Programación y Computación 2 consistió en desarrollar un programa capaz de simular el funcionamiento de una máquina ensambladora con "n" líneas y "m" componentes. Para ello, se utilizó Python como lenguaje de programación, implementando programación orientada a objetos (POO) para crear listas dinámicas que permitieran almacenar temporalmente la información generada. La simulación del ensamblaje de cada producto se realizó mediante ciclos que recorrían los datos almacenados tras la lectura del archivo de entrada. Además, se implementó un frontend utilizando el framework Flask para crear una interfaz interactiva, y las plantillas HTML se generaron con el motor Jinja2 de Flask, lo que permitió lograr responsividad sin necesidad de utilizar JavaScript. Este enfoque facilitó la interacción entre el usuario y el sistema, proporcionando una experiencia dinámica y funcional a través de la web. El proyecto combinó programación backend y frontend para simular el proceso de ensamblaje de manera eficiente.

Palabras clave

Framework, Programación Orientada a Objetos (POO) , Responsividad, Frontend, Backend

Abstract

Project 2 of Introduction to Programming and Computing 2 consisted of developing a program capable of simulating the operation of an assembly machine with "n" lines and "m" components. To do this, Python was used as the programming language, implementing object-oriented programming (OOP) to create dynamic lists that allowed the generated information to be temporarily stored. The simulation of the assembly of each product was carried out using cycles that went through the data stored after reading the input file. In addition, a frontend was implemented using the Flask framework to create an interactive interface, and the HTML templates were generated with Flask's Jinja2 engine, which allowed responsiveness to be achieved without the need to use JavaScript. This approach facilitated the interaction between the user and the system, providing a dynamic and functional experience through the web. The project combined backend and frontend programming to simulate the assembly process efficiently.

Keywords

Framework, Object Oriented Programming (OOP), Responsiveness, Frontend, Backend

Introducción

Una empresa llamada Digital Intelligence, desarrolló una máquina capaz de ensamblar las partes de cualquier producto mediante una "n" cantidad de líneas que pueden acceder a una "m" cantidad de componentes distintos. Esta máquina tarda 1 segundo en colocarse sobre el recipiente siguiente y una cantidad "x" de tiempo para ensamblar el componente. Esta máquina tiene un orden específico de ensamblaje de cada uno de los componentes del producto.

En este proyecto se pretende desarrollar un software que sea capaz de simular el funcionamiento de la máquina. Este programa será capaz de predecir el tiempo óptimo para elaborar cualquier producto generando una tabla de salida con cada uno de los movimientos de los brazos en cada uno de los segundos que duró el proceso de ensamble del producto optimo, una gráfica del proceso de ensamblaje y más información mediante el framework de Flask, listas dinámicas creadas mediante POO y Python.

Desarrollo del tema

El Proyecto 2 de Introducción a la Programación y Computación 2 trataba sobre simulación del funcionamiento de una máquina ensambladora con una cantidad "n" de líneas de producción con un brazo cada línea y una cantidad "m" de componentes distintos sobre los cuales se movería el brazo para poder generar el producto final.

Listas doblemente enlazadas

Para generar la solución a este proyecto, se tenía prohibido el uso de listas nativas de Python, por tal motivo, se optó por la generación de listas dinámicas mediante POO utilizando clases y distintas funciones.

Para la generación de las estas listas, se creó una clase "nodo" que sería la que contendría la información de esa posición de la lista junto con la información del nodo anterior y posterior a este.

Posteriormente, para poder mantener la información de los nodos ordenada y bien almacenada, se creo una clase "listita" que contendría todos los nodos, un tamaño de la lista y todas las funciones para trabajar con la lista.

El código utilizado para esta sección del proyecto fue casi totalmente reutilizado del proyecto 1 debido a que en ese se usó lo mismo.

Clases para el almacenamiento de la información

Para almacenar la información de manera más ordenada, además del uso de las listas dinámicas, se utilizaron clases con distintos atributos para almacenar los datos, de esta manera se logró un acceso más controlado, ordenado y eficiente de la información de cada máquina y producto.

Lectura del archivo XML

Para la lectura del archivo XML se usó la librería ElementTree. Para este proceso, se comenzó cargando el archivo de entrada al programa. Posteriormente, se crearon variables temporales y se fue recorriendo el XML para almacenar los valores temporales de cada maquina y producto. Una vez con toda la información, se almacenó creando clases que se almacenarían a su vez dentro de listas dinámicas

Simulación

La simulación es probablemente el proceso más importante de este programa, ya que a partir de estas se generarían todos los reportes.

Para la lógica de la simulación, se optó por la creación de una clase que se crearía a partir de una máquina y

un producto, de tal forma de que una instancia de esta clase simulación contendría toda la información necesaria para realizar y que se obtendría después de una simulación de ese producto ingresado.

Dentro de la clase dedicada a la simulación, se crearon atributos, entre los cuales estaría uno que contendría los pasos de la elaboración del producto en una lista para que se pueda ir eliminando ese proceso de la cola y así posteriormente generar la gráfica; otro que contendría una matriz generada con listas dinámicas que almacenaría todos los datos para generar la tabla del reporte de la simulación de dicho producto; otro atributo que contendría una lista con el estado y posición de cada uno de los brazos; y un último atributo que contendría una variable de tipo string que contendría el código de la tabla HTML con la tabla del reporte. Claramente hay más atributos, pero probablemente estos mencionados son los más importantes.

Ahora para comenzar con la simulación, primero se generaron los atributos que se tuviera que generar mediante la información proporcionada del producto y la máquina. Una vez con todo lo necesario para comenzar la simulación, se comenzó iniciando un contador y algunas banderas y variables para validar y almacenar información temporal y se inició un ciclo que permitiría que el código dentro de este se continué ejecutando siempre y cuando haya elementos a elaborar en la lista de los pasos de la elaboración del producto o si se tiene que esperar a que termine el ensamble de un componente del producto.

Dentro de este ciclo hay un ciclo que limpia los estados temporales que deben de ser eliminados para cada brazo, otro que se encargar de la eliminación de un paso de la elaboración ya realizados un ciclo principal que recorrería cada uno de los pasos de la elaboración.

Dentro del ciclo que recorre cada uno de los pasos de la elaboración del producto, se colocó otro ciclo que compararía el paso de la elaboración con cada uno de los brazos, para que de esta forma el brazo correspondiente sea el que realice la acción de ensamblado tal y como dictaminan los pasos de elaboración.

Si se entra dentro del ciclo del brazo, este se moverá hacia adelante, atrás, no hará nada o ensamblará el producto. En esta parte del código es donde se generan las modificaciones en los brazos que permiten el ensamble y se almacenan cada uno de los pasos en una matriz dinámica creada con dos listas.

Adicional a esta función que simula la máquina y las que generan las listas de elaboración y brazos, se decidieron colocar uno que genera el reporte HTML recorriendo la matriz generada con la toda la información de la simulación y colocando la tabla en HTML en una variable de tipo string; y otra función que graficaría la elaboración recorriendo su lista correspondiente mediante graphviz.

Por último, para el programa, era necesario que se pudiera escoger un tiempo de ejecución del programa, por tal motivo, se creó una función paralela a la que simula la máquina, con la diferencia de que en vez de depender de que haya elementos en la lista de elaboración, dependería del valor de segundos que se quiere que se ejecute la simulación.

Generación del archivo de salida XML

Para realizar el archivo de salida, se recorrieron todas las listas en las que se había cargado la información y se fueron. Se recorrió toda la información de cada

máquina y se simuló cada producto para poder encontrar los valores de la elaboración optima.

Esta parte, a mi parecer, fue la más fácil de todo el proyecto a pesar de que previamente pensaba que no sería así debido a que consideraba que sería demasiado complejo en tener que trabajar con tantos ciclos para poder generar cada uno de los datos para la elaboración optima de cada uno de los productos de cada máquina.

Inicialización del programa

Para crear esta función, dentro de la clase "listita" se creó una función que eliminara todos los valores que esta contuviera, para que, de esta forma, se "limpiara". Adicionalmente, se reiniciaron los valores de todas las variables utilizadas.

Uso de Framework de Flask para la creación de la interfaz de usuario

Flask es un framework con mucha popularidad en Python debido a que es muy minimalista y permite un desarrollo web rápido y sencillo. (¿Qué es Flask?, s.f.)

Entre algunas de sus características principales, tiene que permite trabajar con diferentes URLs y que puede generar vistas dinámicas usando plantillas HTML mediante Jinda2, el cual es su motor de plantillas.

Ahora bien, para desarrollar la interfaz de usuario en este proyecto, se creó primeramente un "backend" que enrutaría todo y se encargaría de llamar cada una de las funciones previamente descritas, así como las plantillas de HTML.

Luego de eso, se crearon las plantillas HTML mediante Jinja2.

Con Jinja2 se pueden crear ciclos, condicionales y más dentro de los HTML, de tal forma que se puede controlar desde el backend la manera en la que se mostrará la página.

Para hacer de esta más dinámica, se crearon varios condicionales para mostrar modularmente cada una de las partes del página y ciclos para generar las listas para seleccionar la máquina y producto que se desea simular y generar reporte.

Uso de Bootstrap

Bootstrap es una biblioteca que facilita la creación de sitios web responsivos y atractivos. Logra esto gracias a que proporciona botones, formularios, barras de navegación y muchos más componentes predefinidos que se adaptan automáticamente a los tamaños de la pantalla. (A, 2023)

En el proyecto, se usó Bootstrap para las plantillas de HTML que se crearon mediante jinja2, de esta forma, tendría un mejor estilo de forma más rápida y simple.

Conclusiones

Una forma muy eficiente para almacenar información temporalmente dentro de un programa son las clases, dado a que estas permiten una mejor organización a la que se podría tener si se usarán únicamente estructuras como listas o matrices.

El Framework de Flask es muy simple y minimalista, esto permite que se pueda crear una página de forma más rápida y sin tanto esfuerzo.

El motor de plantillas de Flask, Jinja2, permite usar estructuras de un lenguaje de programa como condicionales y ciclos, esto ayuda a que la página pueda ser más interactiva y responsiva la necesidad de usar JavaScript.

Bootstrap es una muy buena forma para facilitar el trabajo a la hora de crear el frontend de cualquier

página gracias a sus distintos componentes predefinidos que son capaz de adaptarse estilizadamente a cualquier pantalla.

Referencias bibliográficas

¿Qué es Flask? (s.f.). Obtenido de devCamp: https://devcamp.es/que-es-flask/

A, D. (11 de enero de 2023). *¿Qué es Bootstrap? – Una guía para principiantes*. Obtenido de Hostinger: https://www.hostinger.mx/tutoriales/que-es-bootstrap

Apéndice

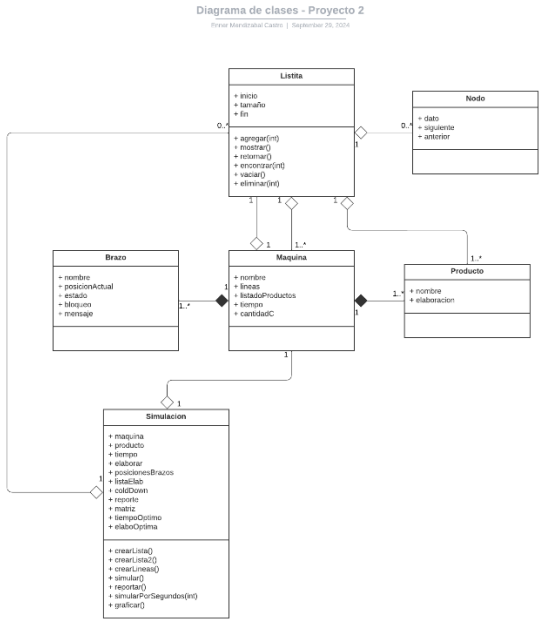

Figura 1. Diagrama de clases del proyecto
Fuente: Elaboración propia

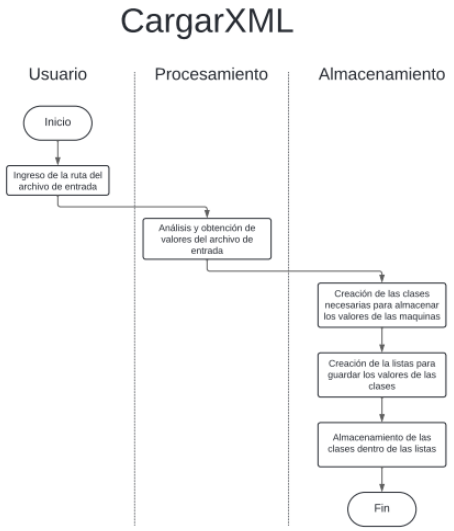

Figura 2. Diagrama de actividades de la función encargada de cargar el archivo XML
Fuente: Elaboración propia

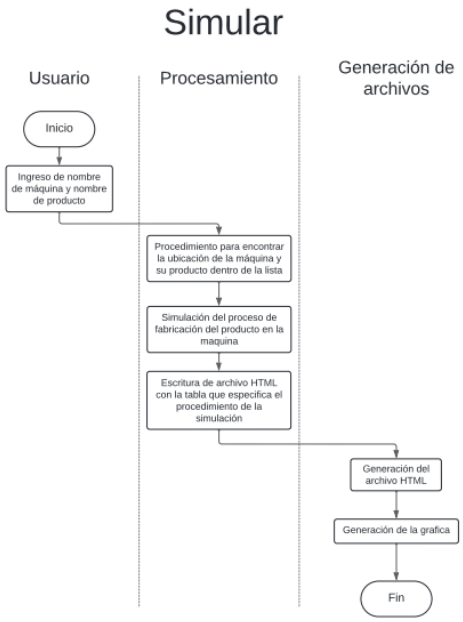

Figura 3. Diagrama de actividades de la función encargada de simular el funcionamiento óptimo de la máquina
Fuente: Elaboración propia

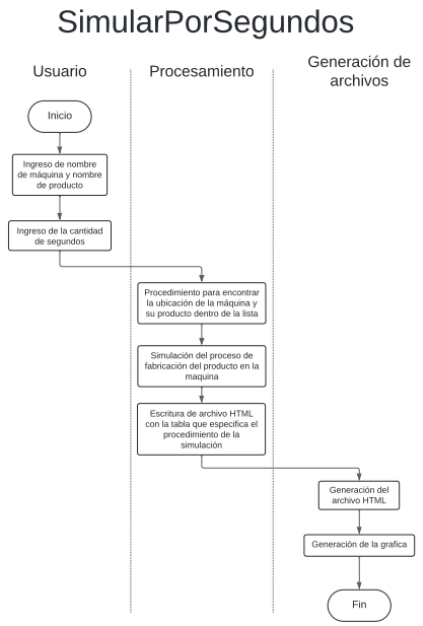

Figura 4. Diagrama de actividades de la función encargada de simular el funcionamiento de la máquina a partir de cierta cantidad de segundos proporcionados
Fuente: Elaboración propia

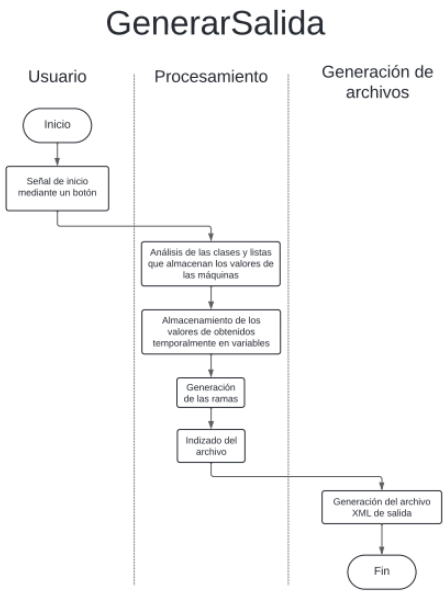

Figura 5. Diagrama de actividades de la función encargada de generar el archivo de salida XML
Fuente: Elaboración propia