

---

## PROYECTO 2

---

202247844 – Josué Samuel de la Cruz Medina

### Resumen

Este proyecto se centra en la carga, el proceso, y la salida de archivos con formato XML. En el programa llevamos a cabo una simulación de ensamblaje de productos en brazos mecánicos o líneas de componentes. El objetivo principal es comprobar y llegar al menor tiempo posible para la realización del ensamblaje de manera eficiente.

Para ello, se toman en cuenta las distintas líneas de ensamblaje que puede tener la empresa, los distintos pasos que se debe de seguir para hacer un ensamblaje exitoso.

Se realizó una implementación de una lista enlazada simple, no nativa de Python, para el mismo proyecto. Además, se crearon varias clases siguiendo los principios de Programación Orientada a Objetos, Producto, Maquina, Resultado, entre otras.

Gracias a la capacidad de calcular el tiempo óptimo del ensamble, la empresa puede lograr un ahorro significativo en términos de dinero y materiales.

### Palabras clave

Lista Enlazada, archivo, ensamble, línea de ensamble, reporte.

### Abstract

*This project focuses on loading, processing, and outputting XML formatted files. In the program we carry out a simulation of product assembly in mechanical arms or component lines. The main objective is to check and arrive in the shortest possible time to carry out the assembly efficiently.*

*To do this, the different assembly lines that the company may have are taken into account, the different steps that must be followed to make a successful assembly.*

*A simple linked list implementation, non-native to Python, was made for the same project. In addition, several classes were created following the principles of Object Oriented Programming, Product, Machine, Result, among others.*

*Thanks to the ability to calculate the optimal assembly time, the company can achieve significant savings in terms of money and materials.*

### Keywords

*Linked List, file, assembly, assembly line, report.*

## Introducción

La empresa Digital Intelligence, S.A. ha solicitado que se cree un programa capaz de simular el ensamble de uno o varios productos en una o varias máquinas de ensamble, también se calcula el tiempo óptimo de ensamble. Se hace uso del framework Flask en Python.

Para esto se carga un archivo XML con las distintas máquinas y productos, se realiza una simulación individual de cada uno de estos y se muestra en pantalla, ya sea en la misma página web o en otro HTML.

Por último, se genera un archivo XML de salida de los productos ya con el tiempo y los cambios hechos.

## Desarrollo del tema

Para el desarrollo del programa se utilizó Python como lenguaje de programación junto a Flask para realizar una página web. Dado que Python es ampliamente conocido, fácil y flexible de utilizar, se decidió utilizar este lenguaje. También facilita el manejo de Listas no nativas.

Por otro lado Flask es un framework para el desarrollo de API (Interfaz de Programación de Aplicaciones). Flask ofrece una gran compatibilidad con Python y no resulta un mayor problema.

### a. Flask

Flask resulta fácil e intuitivo de usar, y gracias a que tiene compatibilidad con CSS es posible decorar nuestras páginas web.

Para crear una dirección en Flask es tan sencillo como declarar la dirección junto a sus métodos, ya sea get o post y declarar una función debajo de está misma que contenga la lógica o parte de la lógica de lo que se va a realizar en la página web, esto generalmente se realiza en el *main* de nuestro programa.

Por otro lado, se debe de crear las distintas plantillas en HTML que se van a usar con cada dirección declarada con anterioridad en el *main*, y se puede incrustar código en los mismos archivos HTML para la lógica del programa.

### b. Diseño Frontend

Como mencione antes, se utilizó HTML junto a CSS para darle decoración a la página web. Se procuró crear una página web sencilla y que fuera amigable con los usuarios, de esta manera sería fácil de utilizar y no generaría tantos problemas.

Lamentablemente el uso de Javascript estaba prohibido, por lo tanto no se utilizó, aunque no fue necesario. Aprendiendo a usar CSS se puede lograr una decoración sencilla de cualquier documento HTML.

### c. Lógica Backend

Junto a videos en la plataforma de Youtube y artículos en internet, se logró idear la lógica para el Backend. El Backend es la parte del sistema que se ocupa de tareas como almacenar, recuperar datos de una base de datos, procesar formularios y gestionar la seguridad de algún sitio web.

En el Backend del programa se gestionan las diferentes listas enlazadas que fueron de utilidad para la creación del mismo programa, al igual que las diferentes clases de la filosofía de Programación Orientada a Objetos, como lo son Producto y Máquina. Las cuales cuentan con sus propios atributos y métodos que fueron esenciales para el procesamiento del archivo.

También se implementaron funciones y métodos para simular el ensamblaje de los diferentes productos, junto a sus reportes en HTML y graphviz, y por último a su archivo de salida en formato XML.

### d. APIs

Las APIs son utilizadas para la integración de funcionalidades en sistemas, las cuales pueden ser reutilizadas en otros programas. El proyecto utilizó varias, una de las principales es la API para manejar archivos XML en Python, sin la

cual no se podría haber comenzado el proyecto ya que el primer paso es cargar un archivo XML.

Asimismo, esta API sirve para la creación del archivo de salida en formato XML. Otra API utilizada es Werkzeug, que ayuda a Flask a ser capaz de descargar y abrir documentos.

### e. Librerías

En el proyecto se utilizan varias librerías, las cuales son conjuntos de funciones y procedimientos predefinidos por desarrolladores en el pasado para realizar tareas específicas. Entre las librerías utilizadas en el proyecto están OS y graphviz.

La librería OS sirve para abrir y editar documentos. Graphviz sirve para generar gráficas.

### f. Funciones importantes del programa

Entre las principales funciones del programa tenemos:

- cargarArchivo()
  - Esta función es la que maneja la lógica al subir un archivo XML en la página web.
- serializarLista()
  - Con esta función podemos serializar las listas que tenemos.
- generarXmlSalida()
  - Gracias a esta función podemos generar el archivo XML de salida,

ya con los cambios en los productos.

- `index()`
  - Nuestra función que maneja la lógica de la página de inicio.
- `reportes()`
  - La función encargada de la lógica para realizar los reportes de los productos.

#### g. Menú principal

Para empezar deberemos de iniciar nuestro ambiente virtual y luego nuestro archivo Main. Ya con esto nos dirigimos a la dirección indicada en nuestra consola y podremos observar el menú principal. Donde nos da unas pequeñas instrucciones que debemos de seguir.

#### h. Cargar archivo

Luego de esto nos topamos con Cargar archivo, página donde debemos de seleccionar y subir nuestro archivo XML de entrada, con los diferentes productos y máquinas.

#### i. Reportes

Ahora, en Reportes, debemos de seleccionar nuestra máquina y productos a Simular, queda a la disposición del usuario el elegir o no tiempo.

Por último, podemos elegir entre tener el reporte en HTML, el reporte de elaboración en un gráfico

hecho con Graphviz y el archivo de salida en formato XML.

#### j. Listas enlazadas

A continuación, enlistaré ventajas y desventajas de las listas enlazadas.

**Control total:** Se tiene control total sobre el funcionamiento y estructura de esta.

**Acceso a nodos eficiente:** A diferencia de las listas estándar de Python, que pueden tener un alto costo al insertar o eliminar elementos en lugares intermedios, en una lista enlazada solo es necesario modificar las referencias de los nodos que rodean la posición afectada. Esto les da a las listas enlazadas una ventaja en términos de eficiencia para estas operaciones.

**Mayor complejidad:** Ya que no tiene las operaciones nativas de Python, se encontraron con diferentes problemas a medida que se avanza en la elaboración del proyecto.

**Consumo de memoria:** Las listas enlazadas consumen mucha más memoria que las estructuras de datos nativas de Python.

#### k. Proceso de simulación

Al tener los datos de las máquinas cargados en una variable global llamada `máquinasGlobal`, simple y sencillamente se recorre esta lista para que se nos muestren las distintas máquinas que se encuentran cargadas.

Luego de esto se valida el tiempo, si el usuario ingreso uno o se debe de usar el tiempo óptimo.

Haciendo uso de la implementación de Listas enlazadas, se almacenan valores dentro de estas, los cuales serian resultados de simulación y las instrucciones de ensamblaje. Para las instrucciones de ensamblaje se busca en el XML de carga inicial toda la parte de la etiqueta elaboración para seguirla al pie de la letra y no tener errores.

Se inicia un bucle durante la simulación durante el tiempo especificado. Durante cada iteración se pueden obtener tres estados: Se actualiza el estado de cada línea de producción, se mueve el brazo mecánico o simple y sencillamente no hace nada.

Al finalizar la simulación, se almacenan los resultados y las instrucciones. Luego se representan en una tabla HTML para que el usuario pueda ver el resultado final. Por último, se tienen las opciones para generar la tabla más personal, un gráfico para la elaboración y/o un archivo de salida.

Si llegará a aparecer algún error la simulación deberá de ser capaz de devolver un mensaje de error apropiado.

## Conclusiones

Este proyecto representa un claro ejemplo del uso del framework Flask en Python, combinado con la implementación de listas enlazadas simples y la utilización de APIs y librerías seleccionadas cuidadosamente para asegurar un funcionamiento adecuado.

La combinación de estas diversas herramientas posibilita la creación de un simulador eficiente para la elaboración de productos, optimizando así el proceso de ensamblaje y permitiendo un mejor manejo de los recursos.

## Referencias bibliográficas

C. J. Date, (1991). *An introduction to Database Systems*. Addison-Wesley Publishing Company, Inc.

Maldeadora (2018) Qué es frontend y backend: Características, Diferencias y Ejemplos, Platzi. Available at: <https://platzi.com/blog/que-es-frontend-y-backend/>

W3schools.com (no date) W3Schools Online Web Tutorials. Available at: <https://www.w3schools.com/>.

## Apéndices:

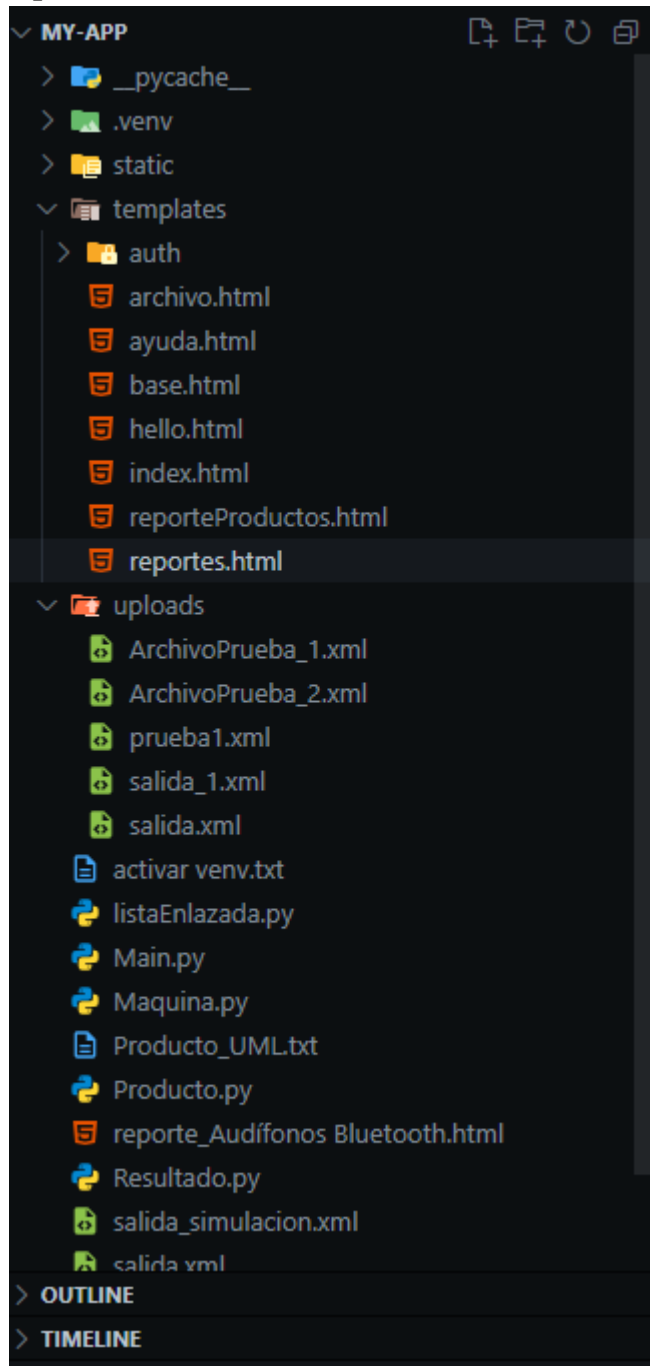
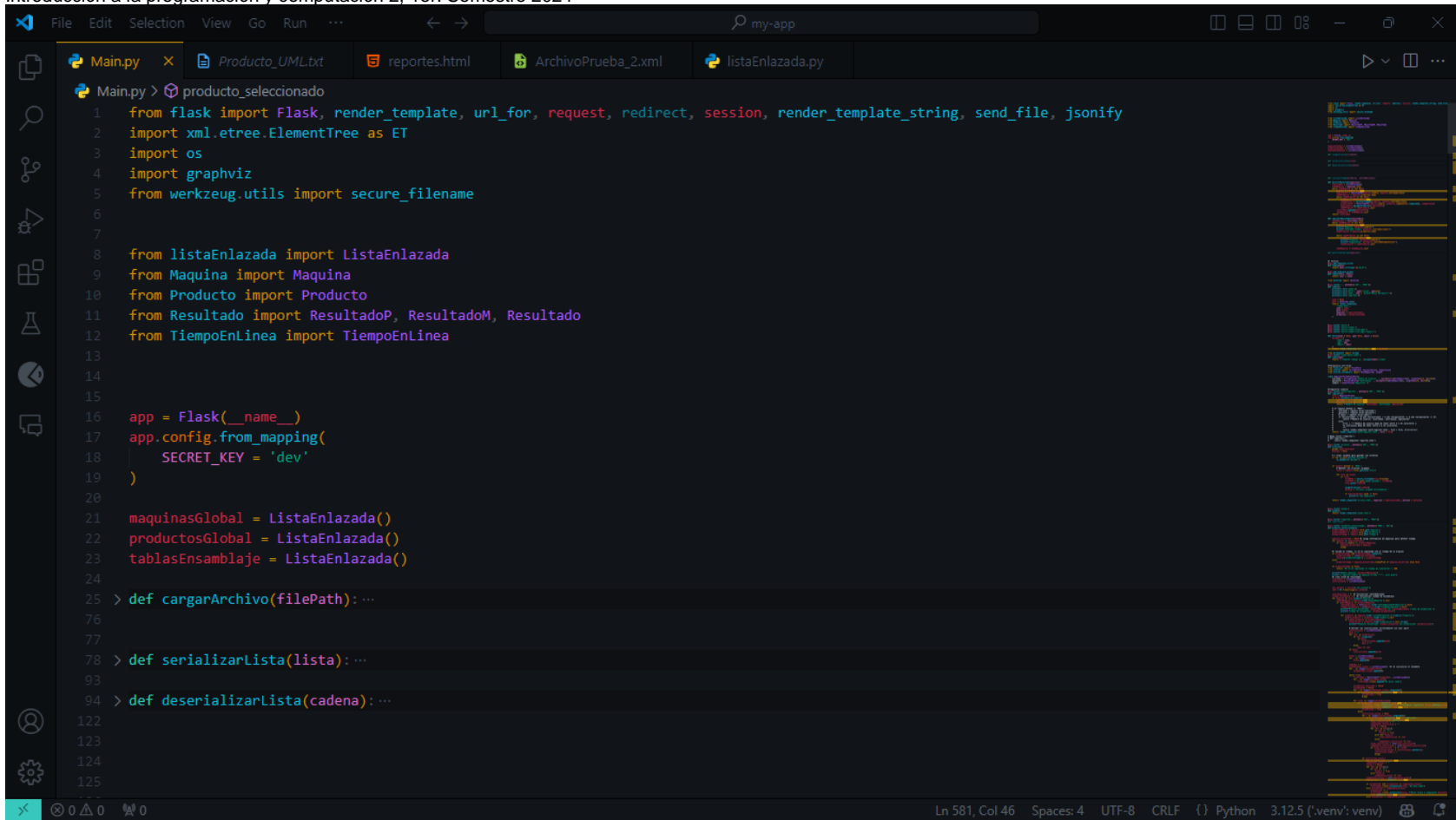


Figura 1 archivos en mi proyecto

Fuente: Elaboración propia



```
1 from flask import Flask, render_template, url_for, request, redirect, session, render_template_string, send_file, jsonify
2 import xml.etree.ElementTree as ET
3 import os
4 import graphviz
5 from werkzeug.utils import secure_filename
6
7
8 from listaEnlazada import ListaEnlazada
9 from Maquina import Maquina
10 from Producto import Producto
11 from Resultado import ResultadoP, ResultadoM, Resultado
12 from TiempoEnLinea import TiempoEnLinea
13
14
15
16 app = Flask(__name__)
17 app.config.from_mapping(
18     SECRET_KEY = 'dev'
19 )
20
21 maquinasGlobal = ListaEnlazada()
22 productosGlobal = ListaEnlazada()
23 tablasEnsamblaje = ListaEnlazada()
24
25 > def cargarArchivo(filePath): ...
26
27
28
29 > def serializarLista(lista): ...
30
31
32
33 > def deserializarLista(cadena): ...
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
```

Figura 2 Main.py  
Fuente: Elaboración propia

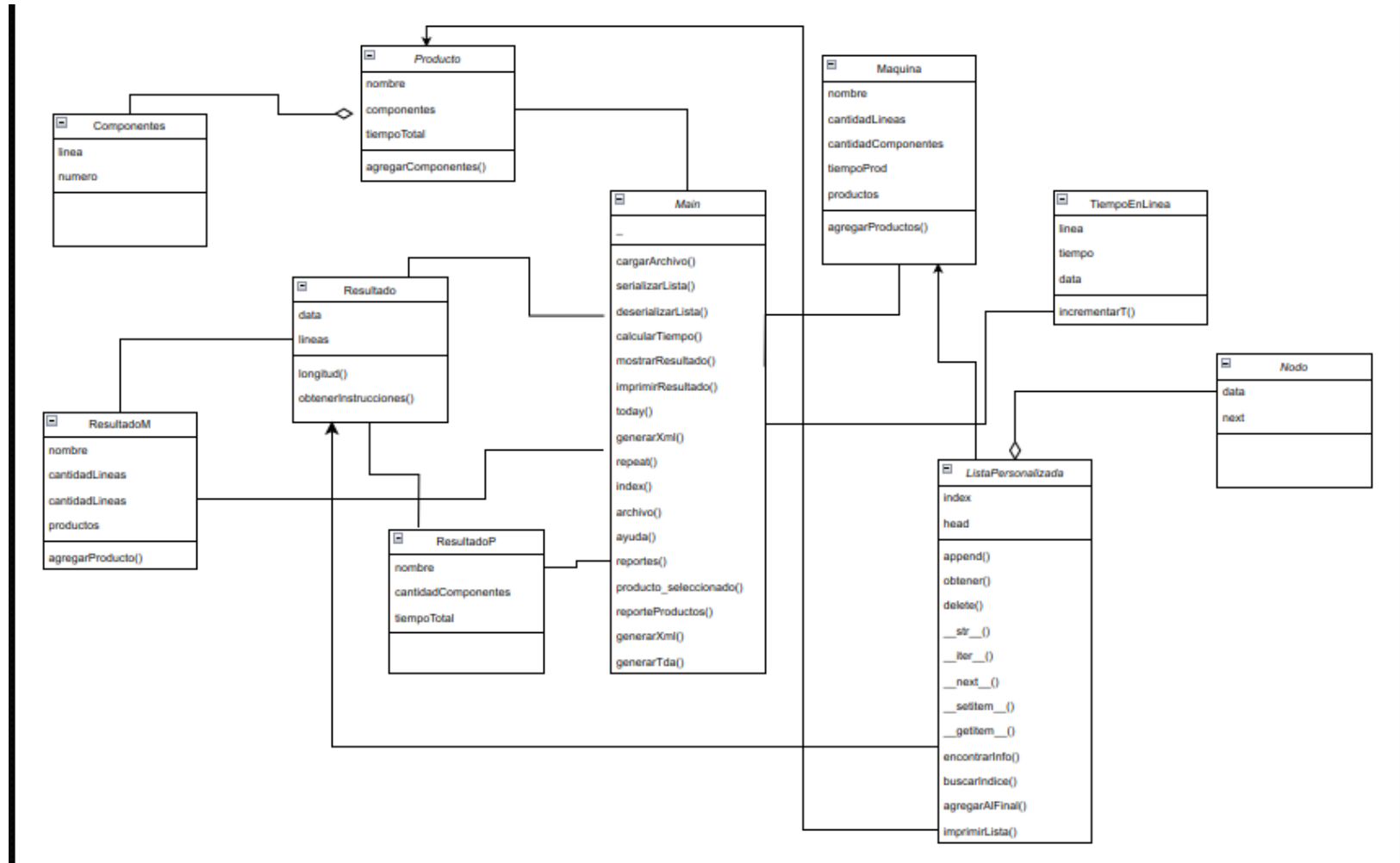


Figura 3 Diagrama de clases  
Fuente: Elaboración propia





Figura 4 Diagrama de Actividades  
Fuente: Elaboración propia