PROYECTO 2 – SIMULACIÓN DE ENSAMBLAJE

202300596 – Xavi Alexander De León Perdomo

Resumen

Este ensayo aborda el desarrollo de un software de simulación para una máquina de ensamblaje automático, un tema de creciente relevancia en la industria manufacturera moderna. El proyecto plantea la creación de una aplicación en Python que modela el funcionamiento de una máquina capaz de ensamblar diversos productos mediante múltiples líneas de producción.

La novedad radica en la optimización del tiempo de ensamblaje y la flexibilidad para simular diferentes configuraciones de máquinas y productos. Este enfoque tiene impactos significativos a nivel técnico, mejorando la eficiencia en la planificación de producción, y económico, al permitir la predicción y reducción de tiempos de fabricación.

El proyecto enfatiza el uso de programación orientada a objetos, estructuras de datos personalizadas y una interfaz web, reflejando las tendencias actuales en desarrollo de software. La implementación de entrada/salida XML y generación de reportes gráficos aumenta su aplicabilidad en entornos industriales reales.

Las conclusiones destacan la importancia de la simulación en la optimización de procesos industriales y la necesidad de soluciones software flexibles y escalables en la manufactura moderna.

Palabras clave

- Simulación industrial
- Ensamblaje automático
- Optimización de producción
- Python
- Programación Orientada a Objetos

Abstract

This essay addresses the development of simulation software for an automatic assembly machine, a topic of growing relevance in modern manufacturing. The project involves creating a Python application that models the operation of a machine capable of assembling various products through multiple production lines.

The novelty lies in optimizing assembly time and the flexibility to simulate different machine and product configurations. This approach has significant technical impacts, improving production planning efficiency, and economic impacts by allowing prediction and reduction of manufacturing times.

The project emphasizes the use of object-oriented programming, custom data structures, and a web interface, reflecting current trends in software development. The implementation of XML input/output and graphical report generation increases its applicability in real industrial environments.

The conclusions highlight the importance of simulation in optimizing industrial processes and the need for flexible and scalable software solutions in modern manufacturing.

The project's comprehensive approach, combining programming skills with practical industrial applications, demonstrates the interdisciplinary nature of contemporary software engineering in addressing complex manufacturing challenges.

Keywords

- *Industrial simulation*
- *Automatic assembly*
- *Production optimization*
- *Python*
- *Object-Oriented Programming*

Introducción

La simulación de procesos mediante software es crucial en la industria manufacturera moderna, permitiendo optimizar tiempos y mejorar la flexibilidad en la producción. Este ensayo explora el desarrollo de un software en Python que modela una máquina de ensamblaje automático con múltiples líneas de producción, destacando su capacidad para simular diversas configuraciones y productos.

El objetivo principal es demostrar cómo esta herramienta puede mejorar la eficiencia en la planificación de la producción y reducir tiempos de fabricación, impactando positivamente en los resultados económicos. El uso de programación orientada a objetos, estructuras de datos personalizadas, entrada/salida XML y generación de reportes gráficos refuerza su aplicabilidad en entornos industriales.

El trabajo plantea preguntas clave: ¿Cómo puede la simulación optimizar procesos industriales? ¿Qué beneficios ofrece para la flexibilidad y escalabilidad en la manufactura? Estas interrogantes guían la reflexión sobre la relevancia de soluciones software en la industria actual.

Desarrollo del tema

Manual de Usuario:

Bienvenido a la aplicación, al comenzar la ejecución comenzaras en la siguiente página

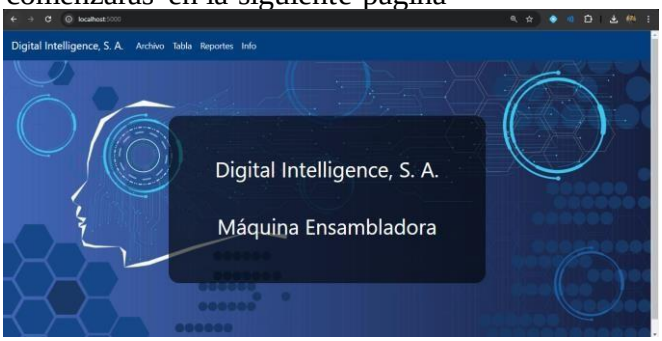

Figura 1. Inicio de la aplicación
Fuente: elaboración propia.

Lo primero es ir a la pestaña de archivo.

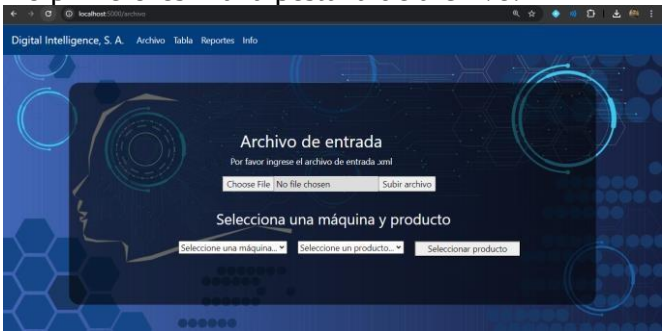

Figura 2. Pestaña archivo
Fuente: elaboración propia.

Una vez aquí debes cargar el archivo de entrada.

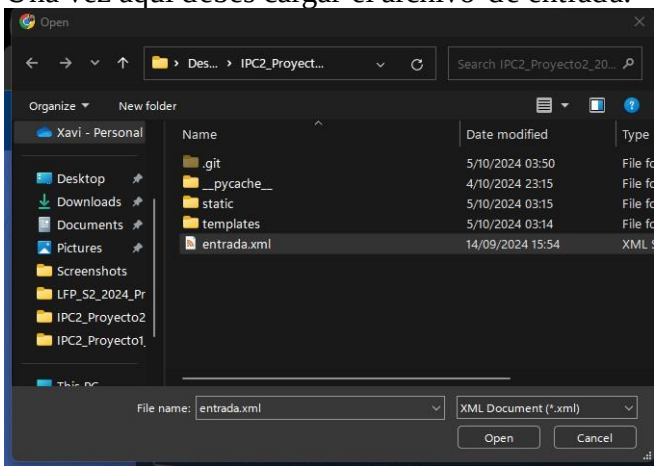

Figura 3. Cargar archivo
Fuente: elaboración propia.

Una vez cargado el archivo lo subimos y podremos seleccionar la máquina.

Figura 4. Seleccionar máquina
Fuente: elaboración propia.

Posterior a ello seleccionaremos el producto.

Figura 5. Seleccionar producto
Fuente: elaboración propia.

Una vez seleccionado el producto podremos ir a la pestaña de tabla y visualizar el proceso del producto.

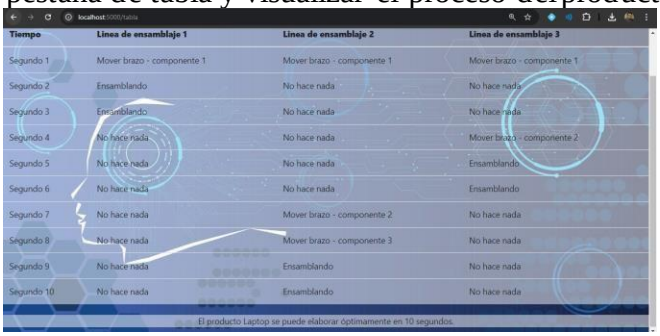

Figura 6. Tabla del producto
Fuente: elaboración propia.

Además, también podremos ir a la pestaña Reportes para observar el gráfico de las instrucciones que sigue la máquina ensambladora.

Figura 7. Reportes
Fuente: elaboración propia.

Por último, puedes dirigirte a la pestaña de info para ver la información del estudiante encargado de desarrollar la página web.

Figura 8. Info
Fuente: elaboración propia.

Manual Técnico:

Este manual explica los componentes técnicos clave del código presentado, como las clases, métodos e importaciones utilizadas.

1. Clases Principales

1.1 Clase Producto

La clase Producto representa un objeto que contiene información sobre el nombre del producto y su proceso de elaboración.

- **Atributos:**
  - nombre: Nombre del producto.
  - elaboracion: Proceso de fabricación del producto (cadena de texto).
  - siguiente: Referencia al siguiente producto (estructura tipo lista enlazada).
- **Métodos:**
  - __init__(self, nombre, elaboracion): Constructor que inicializa un producto con su nombre y proceso de elaboración.
  - __str__(self): Devuelve una representación en texto del producto.

1.2 Clase Maquina

Representa una máquina de ensamblaje y sus atributos relacionados con la producción.

- **Atributos:**
  - nombre: Nombre de la máquina.
  - produccion: Número de líneas de producción que tiene la máquina.
  - componentes: Cantidad de componentes utilizados.
  - tiempo: Tiempo necesario para ensamblar productos.
  - pila_productos: Pila que almacena productos asociados a la máquina.
  - siguiente: Referencia a la siguiente máquina (estructura tipo lista enlazada).
- **Métodos:**
  - __init__(self, nombre, produccion, componentes, tiempo, pila_productos): Constructor que inicializa la máquina con los atributos correspondientes.
  - __str__(self): Devuelve una representación en texto de la máquina.

1.3 Clase Instruccion

Modela una instrucción dentro del proceso de fabricación.

- **Atributos:**
  - linea: Línea de ensamblaje donde se ejecuta la instrucción.
  - componente: Componente que debe usarse en la instrucción.
  - siguiente: Referencia a la siguiente instrucción.
- **Métodos:**
  - __init__(self, linea, componente): Constructor que inicializa una instrucción con su línea y componente.
  - __str__(self): Representación en texto de la instrucción.

2. Clases Abstractas y Pilas

2.1 Clase Abstracta Pila

Esta es una clase base para manejar listas enlazadas de elementos, utilizada para gestionar productos, máquinas e instrucciones.

- **Atributos:**
  - primero: Referencia al primer elemento en la pila.
  - size: Tamaño actual de la pila (número de elementos).
- **Métodos:**
  - insertar(self): Método abstracto que debe ser implementado por las subclases.
  - buscar(self, nombre): Busca un nodo en la pila por su nombre.
  - getNames(self): Obtiene los nombres de todos los elementos en la pila.
  - clear(self): Limpia la pila, eliminando todos sus elementos.
  - __iter__(self): Iterador para recorrer la pila.

2.2 Clase PilaProductos

Subclase de Pila que gestiona productos.

- **Métodos:**
  - insertar(self, nombre, elaboracion): Inserta un nuevo producto en la pila.
  - print(self): Imprime los productos en la pila.
  - getProducto(self, nombre_producto): Busca y retorna un producto por su nombre.

2.3 Clase PilaMaquinas

Subclase de Pila que gestiona máquinas.

- **Métodos:**
  - insertar(self, nombre, produccion, componentes, tiempo, pila_productos): Inserta una nueva máquina en la pila.
  - print(self): Imprime las máquinas y sus productos.
  - getMaquina(self, nombre_maquina): Busca y retorna una máquina por su nombre.
  - getProductos(self, nombre_maquina): Obtiene los nombres de los productos asociados a una máquina.

2.4 Clase PilaInstrucciones

Subclase de Pila que gestiona las instrucciones de ensamblaje.

- **Métodos:**
  - insertar(self, linea, componente): Inserta una nueva instrucción en la pila.
  - print(self): Imprime las instrucciones almacenadas.

3. Importaciones

El código incluye varias importaciones esenciales para su funcionamiento:

- **abc:** Proporciona la capacidad de crear clases abstractas. La clase Pila extiende de ABC y utiliza el decorador @abstractmethod para definir métodos que deben ser implementados por las subclases.
- **Flask:** Utilizado para crear el servidor web y gestionar las rutas y solicitudes HTTP.
  - Flask: Para crear la aplicación.
  - render_template: Para renderizar plantillas HTML.
  - request: Para manejar solicitudes HTTP.
  - flash: Para enviar mensajes flash al usuario.
  - redirect: Para redirigir a otras rutas.
  - url_for: Para generar URLs dinámicas.
  - session: Para almacenar y recuperar datos de la sesión de usuario.
- **os:** Manejo de operaciones relacionadas con el sistema operativo.
- **xml.etree.ElementTree:** Permite trabajar con archivos XML, utilizado para cargar y analizar datos estructurados sobre las máquinas y productos.
- **graphviz:** Genera gráficos y diagramas visuales a partir de datos, usado para la visualización de las instrucciones de ensamblaje.
- **jinja2:** Manejador de plantillas utilizado por Flask para procesar archivos HTML.

4. Aplicación Flask: Clase MyApp

La clase MyApp integra toda la funcionalidad del sistema en una aplicación web basada en Flask.

- **Atributos:**
  - app: Instancia de Flask.
  - pila_maquinas, pila_productos, pila_instrucciones: Instancias de las pilas correspondientes.
- **Métodos:**
  - home(self): Carga la página de inicio.
  - archivo(self): Permite cargar y procesar un archivo XML con datos de máquinas y productos.
  - tabla(self): Genera una tabla con información de las máquinas y productos seleccionados.
  - reportes(self): Crea un reporte gráfico de las instrucciones.
  - info(self): Muestra información adicional de la aplicación.

5. Ejecución

El archivo ejecuta la aplicación Flask llamando al método run() de la clase MyApp en la condición if __name__== '_main_':. Esto inicia el servidor web en modo de depuración para pruebas locales.

Conclusiones

Esta sección debe orientarse a evidenciar claramente las principales ideas generadas, propuestas que deriven del análisis realizado y si existen, expresar las conclusiones o aportes que autor quiera destacar.

Enfatizando, lo importante es destacar las principales posturas fundamentadas del autor, que desea transmitir a los lectores.

Adicionalmente, pueden incluirse preguntas abiertas a la reflexión y debate, temas concatenados con el tema expuesto o recomendaciones para profundizar en la temática expuesta.