Proyecto Web Scraping

Briefing: Proyecto de Web Scraping

Planteamiento

La empresa XYZ Corp está pensando en utilizar una frase que se identifique con sus valores y su misión. El objetivo de este proyecto es desarrollar un programa en Python que realice web scraping para extraer todas las frases de la web https://quotes.toscrape.com/ , además de las frases se quieren recuperar el autor de cada frase, los tags asociados a cada frase, y la pagina "about" con información de los autores. Los datos extraídos deben ser formateados y almacenados adecuadamente.

Objetivos del Proyecto

1. **Acceder a una web preparada para ser scrapeada**: La web contiene muchas frases, con información relacionada .
2. **Extraer información relevante**: Utilizar técnicas de web scraping en Python para obtener todas las frases con la información extra (autor, tags, about).
3. **Formatear los datos**: Asegurarse de que los datos extraídos estén limpios y organizados de manera coherente.
4. **Almacenar los datos en una base de datos**: Utilizar una base de datos SQL o NoSQL para guardar la información extraída.

Plazos

Se os dan 4 días para realizar la entrega y presentar la solución. Las presentaciones serán el martes 30 de Julio.

Condiciones de Entrega

Para el día de la entrega, será necesario presentar:

1. **Repositorio en GitHub**: El repositorio con todo el código fuente desarrollado. 2. **Demo del programa**: Presentar una demo donde se muestre el funcionamiento del programa de scraping.
3. **Presentación de negocio para un publico técnico**: Explicar el objetivo del proyecto y cómo se llevó a cabo, así como una breve descripción del código desarrollado, y las tecnologías empleadas, formato libre, una sola presentación del proyecto. (5 min)
4. **Tablero Kanban**: El enlace al tablero Kanban (Trello, Jira, etc.) utilizado para organizar el trabajo.

Tecnologías útiles

Git/GitHub
Docker
Python (bibliotecas: BeautifulSoup, Scrapy, Requests, etc.)
SQL/NoSQL (MySQL, PostgreSQL, MongoDB, etc.)
Herramientas de gestión de proyectos (Trello, Jira)

Niveles de Entrega

Nivel Esencial:

Un script que accede a la web, extrae las frases y la información asociada y la imprime en la consola.
Limpieza básica de los datos extraídos.
Documentación básica del código y README en GitHub.

Nivel Medio:

Almacenamiento de los datos extraídos en una base de datos.
Implementación de un sistema de logs para la trazabilidad del código.
Test unitarios para asegurar el correcto funcionamiento del scraper.

Nivel Avanzado:

Programación orientada a objetos (OOP) para estructurar mejor el código.
Gestión de errores robusta para manejar excepciones comunes en web scraping.
Un script que actualiza automáticamente la base de datos con nuevos datos a intervalos regulares.

Nivel Experto:

Dockerización del proyecto para asegurar un entorno de ejecución consistente.
Implementación de un frontend básico para visualizar los datos extraídos.
Despliegue de la aplicación en un servidor web accesible públicamente.
Utilizar otras webs de frases para aumentar la cantidad de frases scrapeadas.

Actividad Sugerida

Objetivo: Familiarizarse con las técnicas de web scraping y el manejo de bases de datos para almacenar datos extraídos.

Instrucciones:

1. **Preparación**:
    Familiarizarse con las herramientas de web scraping en Python (BeautifulSoup, Scrapy, Requests).
    Configurar un entorno de desarrollo (recomendado: Virtualenv, Docker).
2. **Desarrollo del Scraper**:
    Crear un script que acceda a la web y extraiga los datos necesarios.
    Limpiar y formatear los datos para asegurar su consistencia.
3. **Almacenamiento en Base de Datos**:
    Configurar una base de datos (SQL/NoSQL) para almacenar los datos.

Desarrollar el código para insertar los datos extraídos en la base de datos. 4.

Documentación y Presentación:

Documentar el código y crear un README detallado.

Preparar una presentación explicativa para público no técnico.

Presentar el código y su funcionamiento a un público técnico, explicando las decisiones de diseño y los desafíos encontrados.