PROYECTO WEB SCRAPING

DESCRIPCIÓN BREVE

Documentación correspondiente al proyecto de Web Scrapping sobre el sitio web GO-GO UFO.

Alejandro López y Gaspar Novel
Desarrollo de aplicaciones Web, CIFP Francesc de Borja Moll

Índice

Definición general del proyecto

Los retos principales de este proyecto consisten en crear un sitio web funcional, diseñar un programa de *Web Scraping* que pueda almacenar la información de los productos que se encuentren en el sitio web en una base de datos.

Requisitos del proyectos

Los requisitos del proyecto se encuentran divididos por módulos.

Lenguaje de Marcas

- Crear un sitio web corporativo de la organización que te hayan asignado (en este caso un sitio web de reserva de UFOs). Inventar los productos y servicios hasta conseguir un sitio funcional y representativo del negocio.
- El esquema de los documentos ha de ser pactado con el equipo de segundo de Dual que consumirá los JSON para extender el negocio.
- Utilizar etiquetas semánticas que ofrece HTML5 para etiquetar la información de los productos y servicios que ofrece el negocio.
- Utilizar CSS3 para garantizar una buena experiencia de usuario y usabilidad del sitio.
- Utilizar HTML5 y CSS3 para garantizar la accesibilidad del sitio.
- Validar el código HTML5 y CSS con los servicios de validación que ofrece W3CC.
- Utilizar el esquema que has propuesto en el módulo de bases de datos para definir los elementos HTML que etiquetarán la información sobre los productos y servicios.
- Definir los espacios de nombres que sean necesarios.

Entornos de Desarrollo

- Crear un repositorio del proyecto en GitHub.
- Crear un fichero README:MD que describa adecuadamente el proyecto. Utilizar Markdown para preparar documentación ágil sobre el proyecto.
- Realizar el control de versiones y trabajar de manera colaborativa con tu pareja de programación.
- Utilizar ramas para realizar los evolutivos, testing y debugginng, siguiendo el flujo de trabajo que se especifica en las sesiones presenciales.
- Documentar el proceso.

- Escribir las historias de usuario/a = identificar y describir los requisitos funcionales de la aplicación.
- Elegir el ciclo de desarrollo que consideres más adecuado la manera de trabajar y al proyecto.
- Utilizar Clockify para llevar un seguimiento del tiempo de desarrollo empleado en cada una de las fases del ciclo de vida del proyecto. Ser riguroso porque los informes generados por esta herramienta serán un artefacto para entregar en la defensa del proyecto.

Programación

- Escribir los componentes de Python que extraigan los datos del negocio de los sitios web y que genere un sistema de almacenamiento en formato JSON para estos datos extraídos.
- Dividir el código en rutinas de modo que sea SRP y OCP.
- Emplear precondiciones y postcondiciones para chequear las invariantes (las lecturas de datos) que maneja el programa y que intercambian las distintas rutinas y módulos que componen el sistema.
- No se pueden utilizar las librerías disponibles en lenguaje Python para parsear el DOM o hacer scraping web. Puedes observar las funcionalidades que ofrecen, estudiar su código y la lógica de la aplicación.
- Elegir la estructura de datos adecuada para representar y manipular en memoria los documentos.
- Utilizar bloques try / except para capturar las excepciones que se puedan producir durante la ejecución del programa.
- Crear una colección de documentos JSON con los productos y servicios resultado del scraping del sitio web.
- Invocar desde la consola el programa y pasarle las opciones adecuadas.

Sistemas Informáticos

- Instalar desde la línea de comandos (CLI) todo el software necesario para desarrollar el proyecto, tanto para la estructura de código Python, como HTML5, CSS3, y el sistema gestor de bases de datos.
- Instalar desde CLI Python 3.X.
- Invocar desde la línea de comandos el programa Python y averiguar qué parámetros admite.
- Averiguar qué errores genera una mala invocación del programa.

- Instalar desde CLI el entorno de desarrollo, sus extensiones y las herramientas de testing y debugging.
- Instalar el gestor de paquetes pip desde línea de comandos.
- Instalar las librerías necesarias de Python con el gestor de dependencias pip.
- Instalar desde la línea de comandos (CLI) el control de versiones git.
- Utilizar desde CLI el control de versiones.
- Crear desde línea de comandos en tu máquina un directorio donde almacenar los ficheros con los documentos JSON. Referido como biblioteca de documentos,
- Copiar o mover desde línea de comandos los documentos de la biblioteca.
- Ejecutar el programa desde la línea de comandos para extraer determinada información del sitio web sobre el que realizas scraping.
- Instalar desde CLI MongoDB y Compass y configurar el motor de BBDD.
- Arrancar desde línea de comandos MongoDB y Compass y realizar operaciones CRUD sobre la base de datos.
- Arranca desde línea de comandos Compass para estudiar el Esquema de los documentos de la base de datos.

Base de datos

- Realizar el modelo del esquema de la base de datos orientada a documentos que almacenará los JSON resultado del scraping del sitio web.
- Crear una colección de documentos JSON con los productos y servicios resultado del scraping del sitio web.
- Definir el tipo de dato de cada campo del documento.
- Crear una base de datos en la nube mediante el servicio de Atlas de MongoDB.
- Definir la autentificación y la autorización sobre la base de datos.
- Definir las colecciones de documentos que se estimen oportunas.
- Utilizar la utilidad Compass de MongoDB para:
    - Consultar la información almacenada en la base de datos.
    - Visualizar las estadísticas sobre uso, presencia, rangos de valores, tipos y demás parámetros que ofrece Compass.
- Utilizar el lenguaje de manipulación de datos de MongoDB para:
    - Consultar la información almacenada en la base de datos.
    - Modificar la información almacenada en la base de datos.

Procedimientos de desarrollo

Herramientas utilizadas

Nuestra principal herramienta de trabajo ha sido Visual Studio Code tanto en la parte que toca a Lenguaje de Marcas y Programación gracias a sus múltiples extensiones que facilitan el trabajo. Cabe destacar el uso de la extensión Conventional Commits para facilitar la integración de los cambios al código.

También hemos utilizado TortoiseGit para simplificar el uso de GitHub, Mongo Shell para crear la base de datos donde almacenamos los documentos, Photoshop para crear el catálogo de UFOs y el propio CLI para la instalación de algunos programas como Python.

Planificación

El primer objetivo era concretar un ciclo de desarrollo para concretar nuestro modo de acutación. Cogiendo de ejemplo el modelo en cascada lo adaptamos a nuestras necesidades ya que no lo hemos seguido totalmente. En cuanto a la parte de la creación del sitio web usamos la metodología ágil Scrum para dividir el trabajo a realizar por orden de prioridad y efectuando pequeñas reuniones periódicas.

Para empezar discutimos la forma que debía tener la base de datos para poder comentarla con el superior de segundo. Para ello nos reunimos durante varias sesiones y fue complejo debido a la poca experiencia que teníamos ambos , pero tras invertir tiempo fue más sencillo encaminar la base a una idea concreta, sobretodo más tarde cuando nos dieron los conocimientos para crear un diagrama e/r.

Lo segundo fue crear las herramientas para poder trabajar en común, como un repositorio en Github o instalar el Live share para hacer P3.

Lo tercero fue el sitio web ya que teníamos más experiencia usando tipos de lenguajes de programación y para eso buscamos otros sitios webs de reserva de vehículos para poder coger ideas y encaminar el proyecto, sobretodo nos centramos en Uber. Lo siguiente fue dividirnos las diferentes páginas y apartados del sitio web para una máxima productividad.

Por último toco ponerse manos a la obra y empezar la parte dura con el código, usamos python como lenguaje y poco a poco fuimos creando los módulos que más tarde relacionaríamos con un modelo de arquitectura.

Requisitos no funcionales

La funcionalidad del proyecto es hacer web scraping sobre una página propia y obtener nuestros productos en un formato Json para posteriormente subirlo a un cluster.

Para ello hemos tenido que generar unos modulos (crawler.py ,get_products.py, make_diccionary, remove_html, etc..) con intención de poder realizar éste algoritmo con éxito y conectarlos entre si dando un resultado para volcarlo en una base de datos.

Url Repositorio

Para ver toda la documentación y los archivos descritos en el documento hay que dirigirse al siguiente enlace:

https://github.com/Alopezmur/proyecto_transversal.git

Arquitectura del sistema

La arquitectura del sistema es el nucleo de un programa, es la forma en la que está están estructurados los módulos que contiene dicho programa. En nuestro caso hemos aplicado una arquitectura modular donde el main llamará a los distintos módulos para ejecutar el programa.

Descripción individual de los módulos:

En éste apartado se explicará detalladamente la descripción general de cada modulo del proyecto:

crawler.py:

-Coge el html y lo tranforma en un string, si no consigue procesarlo dice que ha ido mal.

-Crea una lista con todos los enlaces de html.

-Dentro de la página que se le da busca todos los enlaces.

-Crea una 'webpage' para despues checkear los html (uno a uno).

-Crea un diccionario con la 'webpage' (hasta el úlimo '/' y la lista de links para después checkear que html tiene los productos).

get_products.py:

-Busca cual de los html de list_links es el html con los productos.

-Crea una lista con todos los productos.

-Dentro del html con los productos, busca cada producto a partir de

'class="producto"'.

remove_html:

-Por cada item de la lista de get_products, lo limpia de marcas que puedan dar fallos.

-Añade cada item limpio en una nueva lista.

-Invoca esta función para coger los productos del hmtl.

-Limpia el html de espacios y otras marcas que puedan provocar un error.

make_diccionary:

-En este módulo entra el url y saca una lista de diccionarios de los productos.

main.py:

-Este módulo llama al make_diccionary que es la reunión de todos los demás módulos, para convertirse en el módulo ejecutable sin tener que ir uno por uno.

Diagrama de módulos:

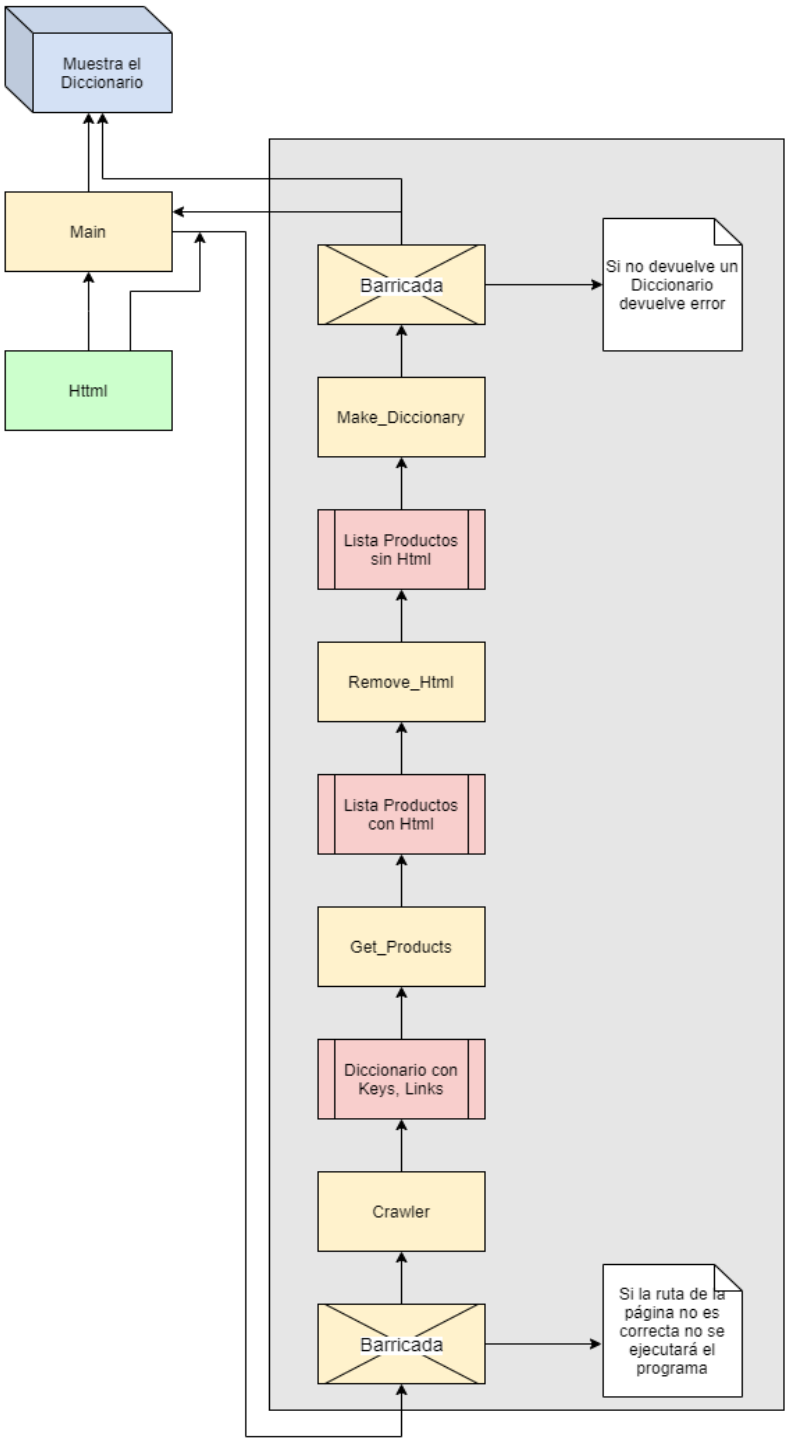
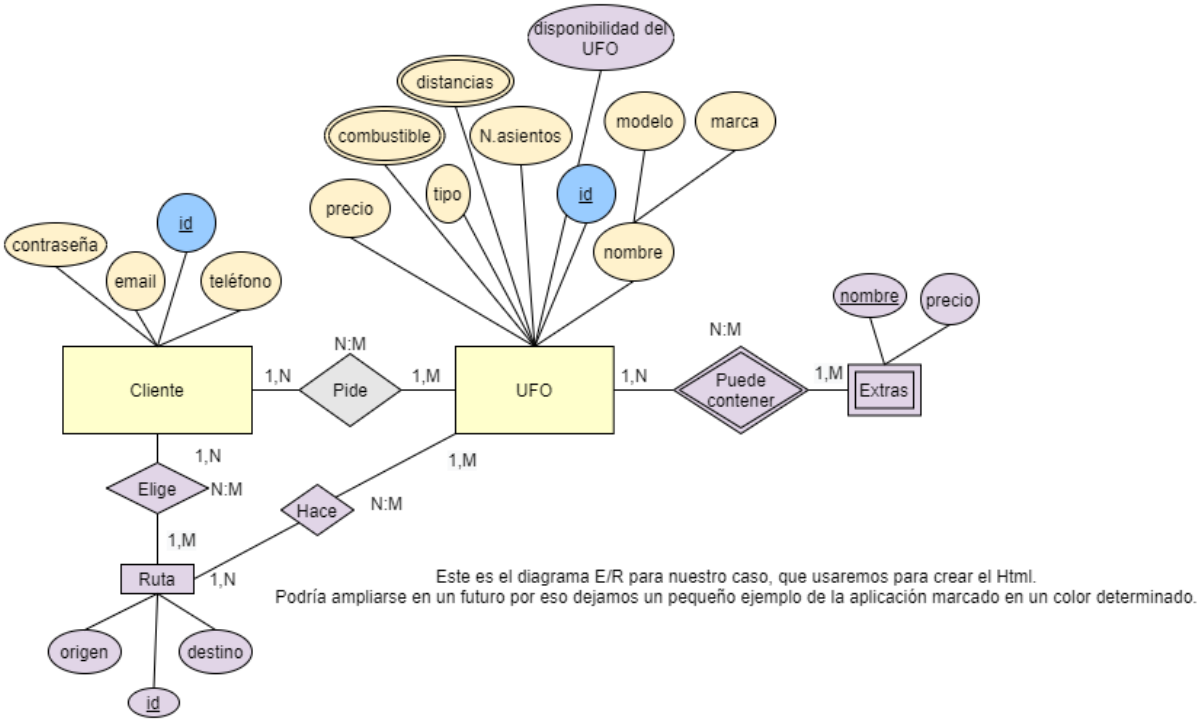


Diagrama E-R



Planificación y duración Clockify:

-https://github.com/Alopezmur/proyecto_transversal/tree/main/Doc

Dificultades y posibles mejoras

Nuestra mayor dificultad ha sido saber donde por donde empezar ya que hasta un cierto momento después de una reunión con el tutor no se supo bien por donde coger las riendas del proyecto ni que pasos a seguir, también la búsqueda de información por los problemas ocasionales en páginas de terceros para poder completar el proyecto con éxito.

Respecto a las posibles mejoras, nuestro proyecto es bastante ampliable ya que se podría abrir más bases de datos con usuarios por ejemplo, no habría que dejar todo para lo último sino ir haciendo pequeñas cosas a los largo del tiempo para cumplir con los strings marcados..

Bibliografía

- https://docs.google.com/spreadsheets/d/1L-0V6dh-FOICIMtqseQoM_PuL8OZCPYSTuevgHgh7OU/edit#gid=0
- https://stackoverflow.com/
- Archivos pasados por Slack, Clasroom, GoogleDrive o de repositorios de Github.
- https://www.youtube.com/
- Y un largo etc...