Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación

IIC2513 - Sección 1 — Tecnologías y Aplicaciones Web

Tarea 1

Entrega

- **Fecha y hora**: Viernes 22 de marzo del 2024, a las 22:00
- **Lugar:** Repositorio individual en la organización del curso en GitHub

Objetivos

- **Construir** programas usando herramientas para interactuar con aplicaciones web
- **Utilizar** la herramienta *DevTools* para explorar páginas web
- **Producir** documentación efectiva y clara que permita el entendimiento del proceso realizado

Descripción

En esta tarea, podrán adentrarse en el mundo de la automatización web y el análisis de páginas a través de herramientas avanzadas de programación, realizando *web scraping* . El objetivo principal es desarrollar habilidades prácticas en la construcción de programas que interactúan con aplicaciones web, utilizando para ello la biblioteca de **Selenium**. Una parte crucial de esta tarea será el uso efectivo de las herramientas DevTools, que les permitirá inspeccionar y comprender la estructura de las páginas web.

Para poder desarrollar correctamente esta tarea, contaran con 2 librerías. A continuación se encuentran los *links* de instalación:

- **Selenium** (ver la sección *Python*)
- **webdriver-manager**

Los objetivos de esta tarea son:

- Desarrollar un controlador de un navegador, para el correcto manejo de interacciones automatizadas con páginas web, utilizando un webdriver.
- Aprender a configurar y utilizar el webdriver para interactuar de manera eficiente con elementos web, como formularios, botones y enlaces, para realizar tareas específicas como la extracción de datos o la automatización de pruebas web.
- Familiarizarse con la estructura HTML de páginas web funcionales, y utilizar *DevTools* para inspeccionar páginas web.

Ayuda al profesor Samuel Oak

Figura 1: Imagen profesor Oak, quien requiere tu ayuda para sus investigaciones Pokémon.

El profesor **Samuel Oak**, de la región de **Kanto**, te ha encomendado una tarea crucial: necesitas investigar a trece Pokémon específicos. Esta misión es de suma importancia, ¡y el destino de estos Pokémon depende de ti! Deberás "scrapear" información detallada sobre los siguientes Pokémon:

- **Pikachu**
- **Charizard**
- **Snorlax**
- **Gyarados**
- **Lugia**
- **Eevee**
- **Rowlet**
- **Greninja**
- **Lucario**
- **Crobat**
- **Kingambit**
- **Salandit**
- **Entei**

Para lograr esta misión, utilizando las librerías entregadas, deberás seguir los siguientes pasos:

1. Inicializar el **driver**.
2. Abrir la WikiDex.
3. Utilizar el buscador e ingresar el Pokémon deseado.
4. Del apartado de datos, extraer los siguientes datos:
   a. **Tipo**
   b. **Categoría**
   c. **Peso**
   d. **Altura**
5. Guardar los datos en el archivo `pokemons.csv`. Estos deben ir ordenados[1], y deben seguir el siguiente formato:

   ```
   NOMBRE;TIPO1,TIPO2,...;CATEGORIA;PESO;ALTURA
   ```

6. Finalmente, deberás cerrar el navegador.

[1] Una vez que guardes la información extraída de los pokémons, deberás ordenar el archivo por el peso de los pokémons de manera descendente.

IMPORTANTE: Recordar que uno de los objetivos de esta tarea es familiarizarse con las *DevTools*, por lo que es indispensable que antes de utilizar Selenium se inspeccione la página y analice su composición para saber dónde está la información deseada y pensar posibles maneras de acceder a esta. De ese modo, se podrían encontrar múltiples formas de obtener los datos solicitados.

Código base

Para facilitar la corrección de esta tarea, se proporcionan dos archivos principales, en los que deberán completar los métodos de cada clase, además de un instalador de paquetes necesarios para la tarea. A continuación, se explica los métodos a completar en el archivo `"Web_driver.py"`:

```
class WebDriver:

    # NO MODIFICAR
    def __init__(self):
        self.active_tab = 0
        self.driver = None
        self.tabs = []

    # COMPLETAR
    # Deberas utilizar la librera webdriver para instanciar el driver y los tabs
    # No tiene atributos, no retorna nada
    def initialize_driver(self) -> None:
        pass

    # COMPLETAR
    # Funcion para cargar una pagina
    # Recibe un str, no retorna nada
    def load_page(self, page: str) -> None:
        pass

    # COMPLETAR
    # Funcion para abrir una nueva pestana, agregar un tab y actualizar el active_tab
    # Recibe un str, no retorna nada
    def new_tab(self, page: str) -> None:
        pass

    # COMPLETAR
    # Funcion que actualiza el atributo active_tab y cambia de pestana del driver
    # En caso que no queden pestanas, se debe resetar el driver
    # Recibe un int, no retorna nada
    def change_tab(self, page_index: int) -> None:
        pass

    # COMPLETAR
    # Funcion para cerrar una pestana, debe actualizar los atributos
    # No recibe ni retorna nada
    def close_tab(self) -> None:
        pass

    # COMPLETAR
    # Funcion para hacer click en un elemento
    # Recibe el tipo de busqueda y el valor, y hace click en el elemento. No retorna nada
```

```
    def click_element(self, by: By, value: str) -> None:
        pass

    # COMPLETAR
    # Funcion para encontrar un elemento
    # Recibe el tipo de busqueda y el valor, y retorna el elemento
    def find_element(self, by: By, value: str) -> SeleniumWebElement:
        pass

    # COMPLETAR
    # Funcion para escribir en un elemento
    # Recibe el tipo de busqueda, el valor y el texto a escribir, y escribe el texto en el
        ↪ elemento
    def write_in_element(self, by: By, value: str, text: str) -> None:
        pass

    # COMPLETAR
    # Funcion para extraer texto
    # Recibe el tipo de busqueda y el valor, y retorna el texto del elemento
    def get_text(self, by: By, value: str) -> str:
        pass

    # COMPLETAR
    # Funcion para extraer atributo title
    # Recibe el tipo de busqueda y el valor, y retorna el atributo title del elemento
    def get_title(self, by: By, value: str) -> str:
        pass

    # COMPLETAR
    # Funcion para extraer url
    # Retorna la url de la pagina actual
    def get_url(self) -> str:
        pass

    # COMPLETAR
    # Funcion para cerrar el driver
    # Cierra el driver y resetea el atributo a None
    def quit_driver(self) -> None:
        pass
```

```
class Scrapper:

    # No modificar
    def __init__(self, chrome: WebDriver):
        self.chrome = chrome

    # COMPLETAR
    # Debe buscar el pokemon en la pagina
    # Recibe el nombre del pokemon y no retorna nada
    def find_pokemon(self, nombre: str) -> None:
        pass

    # COMPLETAR
    # Debe buscar la informacion de cada pokemon en la pagina
    # Recibe una lista de nombres de pokemones y retorna una lista de listas con la informacion
        ↪  de cada pokemon.
```

```python
    def extract_pokemon_info(self, pokemon_list: list[str]) -> list:
        info_pokemons = []
        return info_pokemons

    # COMPLETAR
    # Debe ordenar la informacion de los pokemones por peso
    # Recibe una lista de listas con la info de cada pokemon y retorna una lista de listas con
        ↪ la info de cada pokemon ordenada
    def sort_by_weight(self, info: list) -> list:
        info_ordenada = []
        return info_ordenada

    # COMPLETAR
    # Debe escribir la informacion de los pokemones en un archivo csv (Incluyendo el header)
    # Recibe una lista de listas con la info de cada pokemon y no retorna nada
    def write_csv(self, info: list) -> None:
        header = "NOMBRE;TIPOS;CATEGORIA;PESO;ALTURA\n"
        filename = 'pokemons.csv'
        print(f"Se ha creado el archivo {filename}\n")
```

1. `Web_driver.py`: Esta clase está destinada a ser utilizada como controlador del driver del navegador. Según la documentación del paquete `webdriver-manager`, la biblioteca soporta varios navegadores; sin embargo, se recomienda utilizar `ChromeDriver` por su compatibilidad y estabilidad.

2. `Scrapper.py`: El propósito de esta clase es realizar la tarea de "scrapear"la información solicitada, utilizando como apoyo la clase `WebDriver`.

3. `Packages.py`: Este script se encarga de la gestión de dependencias, instalando automáticamente los paquetes necesarios para la ejecución de la tarea.

4. `Main.py`: Este script automatiza la navegación web y extrae información específica sobre Pokémon desde un sitio web dedicado. Para lograr esto se deben seguir los siguientes pasos:

   a) **Instanciar el `WebDriver` e inicializarlo:** Crea una nueva instancia de `WebDriver` y usa el método para inicializar el navegador web.

   b) **Cargar la página deseada:** Utiliza el `WebDriver` para navegar a la página de Wikidex.

   c) **Instanciar el `Scrapper`:** Crea una instancia de la clase `Scrapper`.

   d) **Extraer la información de cada Pokémon:** Utiliza el `Scrapper` para recopilar información detallada de cada Pokémon.

   e) **Guardar la información extraída:** Escribe los datos recopilados en un archivo CSV llamado *pokemons.csv*, manteniendo el orden solicitado (por peso).

   f) **Cerrar el driver:** Cierra el navegador web.

[2] Es importante considerar que no es obligatorio emplear todos los métodos disponibles en tu Scraper de la clase WebDriver. Sin embargo, es esencial completar la implementación de todos los métodos de dicha clase para asegurar su funcionalidad completa.

Documentación del proceso

Este ítem tiene como propósito comprender el proceso detallado de cómo se llevó a cabo la obtención de datos. En primer lugar, se espera que expliquen cómo se familiarizaron con la composición de la página y sus respectivas herramientas de desarrollo del navegador (*Dev Tools*). Deben describir los pasos realizados para navegar hasta secciones específicas de la página, así como la ubicación de información relevante dentro de los archivos y componentes.

Posteriormente, es necesario describir por qué esta exploración previa es crucial para poder automatizar la navegación y llevar a cabo la búsqueda de los datos requeridos de manera eficiente.

Para asegurar una correcta entrega de la documentación, se les hará entrega de una plantilla con la estructura que debe seguir su descripción y las preguntas que deben responder.

Entregables

Cada entrega deberá incluir los siguientes archivos:

- Los archivos de extensión `.py` con el código de las clases completados para realizar el proceso de *web scraping*
- El archivo `pokemons.csv` con la información requerida de los pokemons solicitados.
- `README.md` con el paso a paso de las acciones realizadas durante el proceso de *web scraping*, así como también de una descripción de que fue lo que extrajeron y cómo lo hicieron.

Rúbrica

A continuación, se describe el criterio de cada uno de los ítem de la rúbrica con la que se evaluará esta entrega. La nota se calcula al 50 % considerando un total de 16 puntos.

- **Clase `Web_driver.py` [2.5 puntos]**: Que al ejecutar el código, la clase pueda inicializar el navegador y pueda utilizar todos los métodos descritos anteriormente, además de que reciba y retorne los parámetros especificados correctamente.
- **Clase `Scrapper.py` [2.5 puntos]**: Que pueda inspeccionar el documento HTML de la página y que sea capaz de buscar y extraer la información de cada pokemon, además de ordenar y guardar la información extraída en el formato solicitado.
- **Archivo .csv [1.5 puntos]**: Debe entregarse el archivo `pokemons.csv` la información solicitada.
- **`Main.py` [1.5 puntos]**: Este es el archivo principal de su programa, por lo al ejecutarse, debe cumplir con todos los pasos descritos en la misión.
- **Test cases (ocultos) [3 puntos]**: Los códigos serán revisados por test cases, por lo que es importante que cumplan con la declaración de parámetros que recibe y retorna cada función, además de tener en cuenta la actualización de las variables de la clase `Web_driver.py` al momento de utilizar algún método que modifique el estado de estas.
- **Comprensión y documentación del proceso [5 puntos]**: Que se explique de manera clara y ordenada el paso a paso de la realización de la tarea. Comenzar con el comando que se debe ejecutar para dar inicio al *web scraping* y con la explicación del funcionamiento de tu código. Luego comentar el análisis y familiarización de la página web, seguir con como este se relaciona con el uso de Selenium y la utilidad de tener un controlador web. Finalmente, terminar con las sugerencias que te hubieran ayudado a realizar de manera más fácil el scraping.

Dudas

Pueden dejar sus preguntas sobre instalación o enunciado en las ***issues*** del repositorio del curso. No se van a responder dudas por correo.