

**ESCUELA TECNICA SUPERIOR DE INGENIERÍA INFORMÁTICA**

**Trabajo fin de Grado**

Grado en Ingeniería Informática – Ingeniería del Software

**Análisis del mercado laboral a partir de datos de LinkedIn**

**Realizado por**

Santiago Zuleta de Reales Toro

Dirigido por

Jose Antonio Troyano Jiménez

Departamento

Departamento de Lenguajes y Sistemas Informáticos

Curso

2023/24

# Resumen

El mercado laboral en el sector de la tecnología y el desarrollo software está en constante cambio y avance. Además, en el ámbito tecnológico, el mercado avanza a pasos agigantados. Esto hace que surjan nuevas tendencias y prácticas que las empresas quieren aplicar a su modelo de negocio, aumentos y descensos en la demanda de uso de herramientas provocado por tecnologías emergentes, nuevos sectores en las plantillas de las empresas e incluso nuevos puestos de trabajo.

Este cambio se ve reflejado en las ofertas de empleo publicadas en el día a día y en las exigencias de las empresas que les hacen decantarse por un aplicante u otro.

LinkedIn es la mayor red social de ámbito profesional con más de 300 millones de usuarios en activo. Profesionales, empresas, y empleados de recursos humanos utilizan este portal como herramienta de trabajo o como plataforma para impulsar su carrera profesional. Los datos disponibles en esta plataforma son muy extensos y variados y contienen información muy relevante y que pueden reflejar tendencias en ofertas de empleo y el mercado laboral.

Este proyecto trata del desarrollo de un sistema de información que sea capaz de realizar una extracción periódica de datos del mercado laboral a partir del portal de empleo de LinkedIn. Estos datos serán almacenados y sometidos a un análisis que el usuario podrá consultar e incluso obtener recomendaciones según ciertos parámetros dados.

En cierto modo, el sistema implementará un proceso de vigilancia tecnológica dentro del propio sector laboral tecnológico. Los análisis periódicos permiten comparar tendencias, ya sea en alza o en baja del mercado laboral. Este proceso de vigilancia periódica pone a disposición del usuario un reflejo de la demanda laboral en la actualidad.

La motivación de este proyecto es tratar de ofrecer orientación a cualquier persona que esté en proceso de búsqueda de empleo ofreciendo un resumen general del estado en el que se encuentra la demanda laboral en el sector de la tecnología y el desarrollo software conociendo las tendencias y requerimientos en auge. También se aportarán recomendaciones de las exigencias más comunes de las empresas dependiendo del perfil del usuario y del puesto de trabajo que busquen.

# Tabla de contenido

[Resumen 2](#_Toc151138056)

[Tabla de contenido 3](#_Toc151138057)

[1) Fases del proyecto 4](#_Toc151138058)

[1.1) Exploración de herramientas o proyectos similares 4](#_Toc151138059)

[1.1.1) Herramientas de extracción de datos de sitios web 4](#_Toc151138060)

[1.1.2) Herramientas de extracción de datos de LinkedIn para marketing y ventas 8](#_Toc151138061)

[1.1.3) Proyectos de análisis (GitHub) 10](#_Toc151138062)

[1.2) Análisis de herramientas 12](#_Toc151138063)

[1.2.1) Scraping 12](#_Toc151138064)

[1.3) Elección de herramientas 16](#_Toc151138065)

[1.3.1) Selenium y BeautifulSoup 16](#_Toc151138066)

# Fases del proyecto

El proyecto está dividido en varias fases. Cada fase tendrá un propósito concreto en cuanto al objetivo del proyecto y se realizarán de manera secuencial.

Se distinguen las siguientes fases en el proyecto:

## Exploración de herramientas o proyectos similares

En esta fase previa al desarrollo del software que da soporte al proyecto se ha indagado sobre herramientas, servicios o repositorios de proyectos en GitHub en los que se implementen funcionalidades similares a la propuesta del proyecto.

Tras una búsqueda encontramos varias herramientas que ofrecen servicios similares a los que hemos mencionado, tanto de extracción, procesamiento y almacenamiento de datos sobre el mercado laboral como de análisis de los datos.

Las clasificamos en las siguientes secciones:

### Herramientas de extracción de datos de sitios web

El propósito final del proyecto es extraer conclusiones de un conjunto de datos a partir de un análisis y ser capaces de captar tendencias en las exigencias al aplicante de la oferta de trabajo. Pero para llegar a estas conclusiones necesitamos un conjunto de datos consistente que nos permita realizar un estudio sobre las ofertas de empleo.

Los datos de las ofertas de trabajo son accesibles de manera pública en la web de LinkedIn, pero necesitamos que estos datos estén almacenados de manera estructurada para poder realizar consultas concretas y análisis sobre ellos.

Cualquier persona puede extraer datos de una web pública a través de técnicas que se conocen como “Web Scrapping”. Esta técnica consiste en identificar en el código de la página web los elementos que contienen la información que nos interesa para extraer los datos.

Sin embargo, según a que web estemos accediendo, la tarea se puede complicar. Las personas que desarrollan la web pueden poner ciertas barreras para proteger los datos de la web en cuestión. Algunas de las barreras que nos encontramos son CAPTCHAs, bloqueos de IPs que realizan peticiones masivas o bloqueos de bots de automatización. En concreto, LinkedIn pone bastantes barreras a la hora de extraer datos de su web, por lo que puede ser interesante recurrir a terceros para que realicen este trabajo por nosotros.

El tipo de herramienta que se expone en primer lugar son servicios de empresas que se dedican a la extracción de datos de sitios web. Esto es interesante porque nos ahorra todos los problemas y complicaciones que supone extraer los datos de una web por cuenta propia.

Hay múltiples empresas que ofrecen esta posibilidad. Algunas de ellas son utilizables sobre cualquier sitio web, y otras en webs concretas. En nuestro caso nos interesan aquellas que nos permitan extraer información de la web de LinkedIn.

Para acceder a los datos cada herramienta es diferente, normalmente las consultas se pueden realizar a partir de una interfaz gráfica o haciendo consultas a una API.

A continuación, se muestran algunos ejemplos:

#### **ScrapingDog**

ScrapingDog ofrece servicios de extracción de datos de varias fuentes. Ofrece acceso a una API para realizar consultas que devuelven los datos en formato JSON.

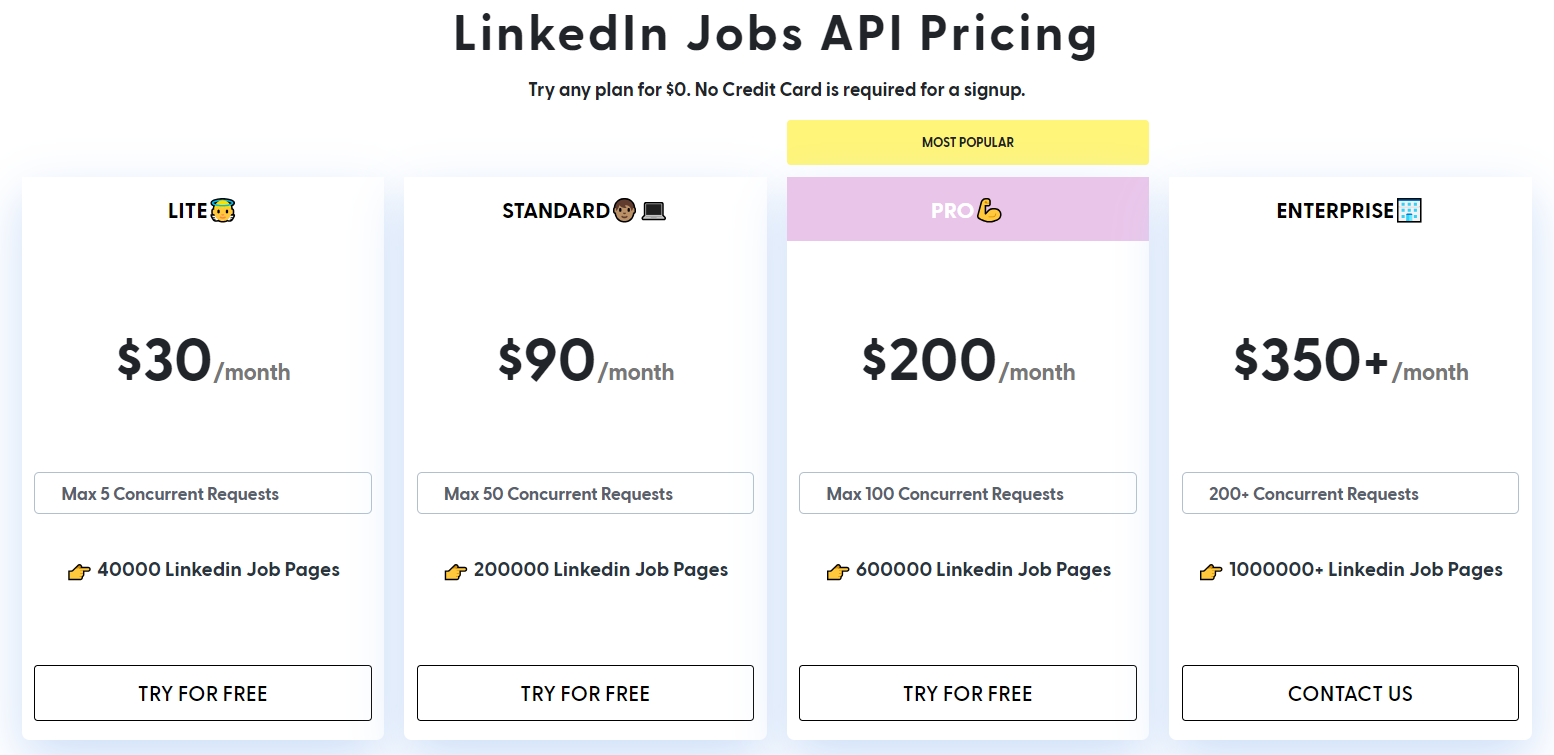
La herramienta pone a disposición una API diferente según la fuente de extracción de los datos. Estas son las APIS a través de las que obtenemos los datos:

* General Scraper API: nos permite extraer HTML de cualquier página web. Realiza el trabajo de automatizamos navegadores, proxies, interpretación de JavaScript y resolución de CAPTCHAs.
* Google Scraper API: la API permite obtener resultados de búsquedas de Google en formato JSON. Se pueden obtener más de 1000 resultados en un día
* Amazon Scraper API: permite automatizar la extracción de datos de productos que están en venta en la web de Amazon. Estos datos incluyen precios, reseñas o detalles del producto entre otros.
* LinkedIn Profile Scraper API: permite extraer datos de perfiles de usuarios y empresas.
* LinkedIn Jobs Scraper: permite automatizar la extracción de publicaciones de ofertas de trabajo en el portal de empleo de LinkedIn

Concretamente nos interesa la API LinkedIn Jobs Scraper, ya que proporciona los datos que nos interesan para realizar un análisis del mercado laboral.

Adicionalmente, la herramienta permite analizar estos datos laborales extraídos para crear informes de mercado personalizados, ayudando a los usuarios a comprender las tendencias de la industria, los puntos de referencia salariales y la demanda regional de roles o habilidades específicos.

Como ya hemos mencionado, los impedimentos que pone LinkedIn para realizar scraping sobre su sitio web son considerables, por lo que el precio del servicio que nos da acceso a la API que extrae los datos de ofertas de trabajo es elevado con respecto al resto de herramientas.  
Estos son los precios de la herramienta LinkedIn Jobs API:



#### **LinkedIn Jobs Scraper (XByte)**

La empresa XByte se dedica a la implementación de herramientas de crawling y scraping sobre portales de ventas, redes sociales, buscadores y comparadores de precios para sus clientes. Gracias a esta herramienta, podemos acceder a información como datos de productos en venta en Amazon, resultados de búsquedas de Google, publicaciones en Twitter o Facebook, datos de publicaciones de portales de alquiler y venta de viviendas y muchos otros además de las publicaciones en el portal de oferta de empleo de LinkedIn.

Concretamente nos interesa la herramienta LinkedIn Jobs Scraper, que extrae los datos del listado de ofertas de empleo por nosotros. Para extraer los datos, se nos pide que realicemos una búsqueda en el portal de empleo. Una vez realizada la búsqueda, debemos introducir la URL de la búsqueda en la interfaz de la aplicación y el mismo programa se encarga de extraer los datos del documento HTML. Una vez extraídos los datos, son ordenados y estructurados y se ponen a nuestra disposición de manera que podemos descargarlos en formato, XLS, CSV o JSON.

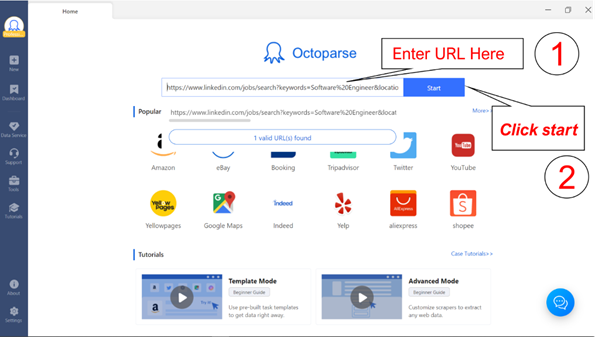
En cuanto al precio del servicio, no hay una cuota fija, debido a que el precio puede variar dependiendo de la fuente de los datos y otros factores como el volumen de los datos a extraer, la frecuencia con la cual se extraen los datos o la complejidad de extracción de estos, teniendo en cuenta barreras como CAPTCHAs, bloqueos de IP o detectores de bots de automatización.

#### **Octoparse (LinkedIn Jobs Scraper)**

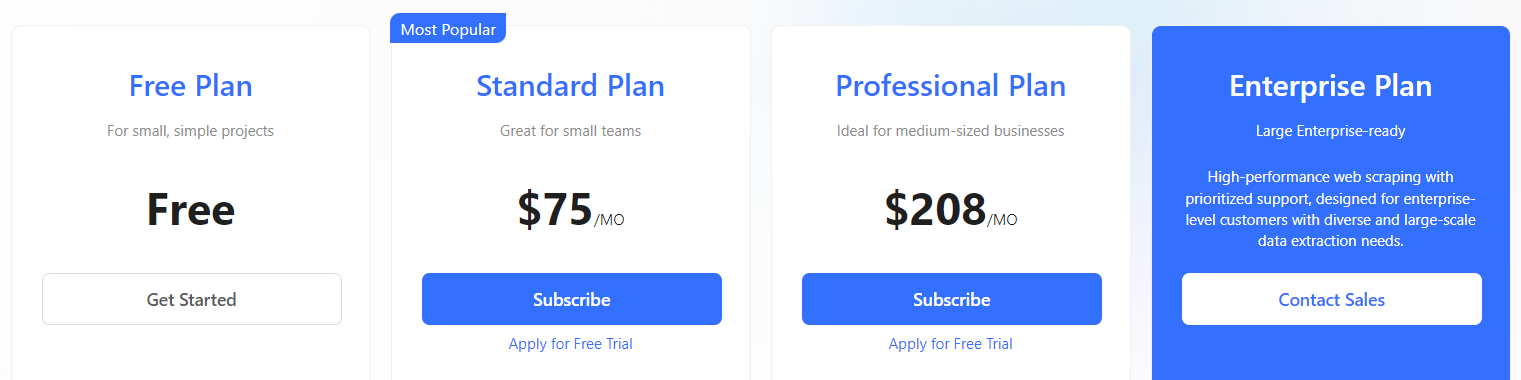
La empresa Octoparse ofrece servicios de extracción de datos de redes sociales, plataformas de eCommerce y publicaciones en LinkedIn entre algunos otros.

El funcionamiento de su herramienta es parecido a la anterior. Ofrecen un software en el que se pide introducir una URL de búsqueda de empleo y especificar ciertos parámetros para que automáticamente se extraigan los datos del listado de publicaciones.

Una vez la herramienta termina el proceso de scraping, nos permite consultar los datos en la propia aplicación o exportarlos en cualquier formato.

Esta es la interfaz de la aplicación, a través de la que podemos tanto realizar las peticiones como visualizar y exportar los datos obtenidos.   


En cuanto al precio de esta herramienta se ofrece un plan gratuito con funcionalidades limitadas. Este no nos serviría ya que el scraping en LinkedIn precisa de técnicas de rotación de IPs, rotación de proxys y solucionadores de CAPTCHAs para extraer los datos de la web. El plan estándar es el más económico de todos los planes de pago y ofrece todas las funcionalidades necesarias para hacer scraping en LinkedIn. El precio mensual es de 75$.



Como vemos, en Internet podemos encontrar varias empresas que ofrecen servicios de extracción de datos de LinkedIn. Sin embargo, este es un servicio costoso. En comparación a otros sitios web, LinkedIn nos poner obstáculos considerables a la hora de extraer información de su sitio web, esto hace que el precio del servicio sea bastante alto en comparación a otros servicios de extracción de datos. Por lo tanto, optaremos por la extracción de datos por cuenta propia utilizando técnicas de web scraping.

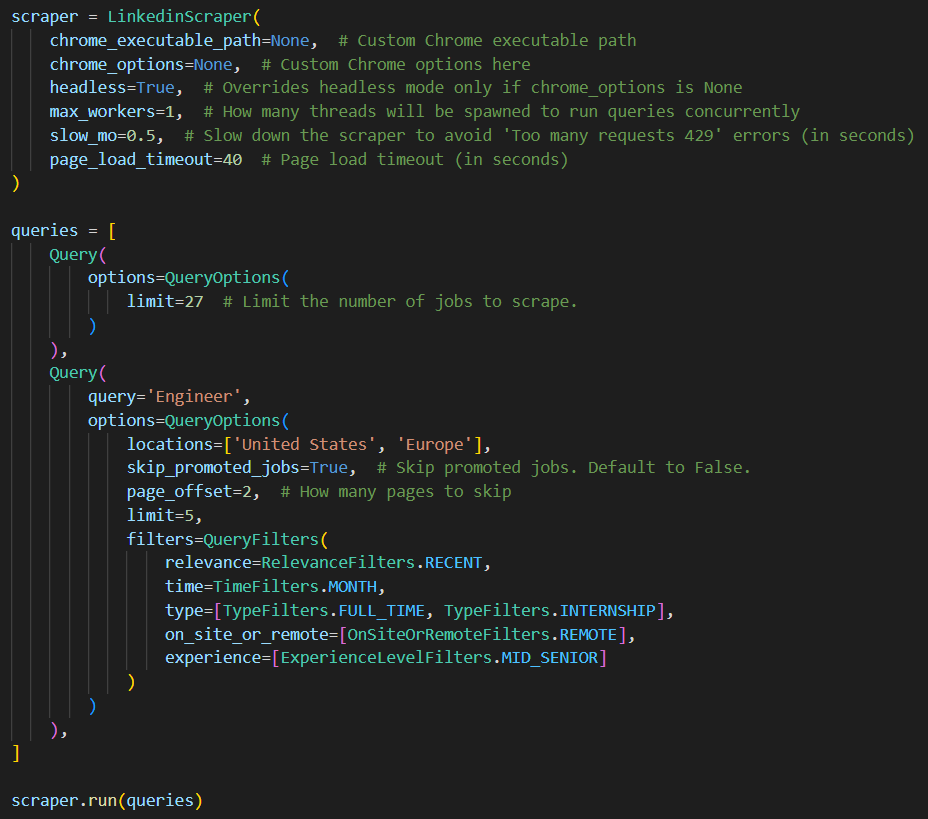
**linkedin-jobs-scraper**

Todas las herramientas de extracción de datos mencionadas se venden como servicio por parte de una empresa. Por otro lado, encontramos la librería de Python *linkedin-jobs-scraper*. Esta librería está disponible a través del instalador *pip*.

Esta librería nos permite extraer los siguientes datos a partir de las ofertas de empleo:  
 - *job\_id* - *link* - *apply\_link* - *title* - *company* - *company\_link*  
 - *company\_img\_link*  
 - *place*  
 - *description*  
 - *description\_html*  
 - *date*  
 - *insights*

Además, se ofrece filtros para aplicar a las consultas realizadas. Algunos de los parámetros de filtro son: el tipo de jornada de trabajo, si es presencial o remoto o el sector del trabajo que se oferta.

A continuación, se muestra un ejemplo de código en el que se realiza una petición a través de la librería:



En definitiva, esta librería ofrece un servicio gratuito de extracción de datos muy completo.

### Herramientas de extracción de datos de LinkedIn para marketing y ventas

Otra de las principales herramientas que ofrecen servicios de extracción de datos de LinkedIn es un modelo de aplicación que automatiza la extracción de datos de perfiles de la red social y su almacenamiento. Los datos que se pueden obtener a partir de un perfil son el nombre completo, la dirección de correo electrónico, el ámbito laboral, la empresa para la que trabaja, el puesto de trabajo, además de las principales habilidades y nombres de colegios o universidades en los que se ha formado la persona en cuestión. Estos datos pueden ser filtrados, ordenados y exportados desde la misma aplicación, lo que facilita su análisis.

Una vez obtenidos los datos, las aplicaciones de este tipo ofrecen funcionalidades adicionales de automatización de LinkedIn, como solicitudes de conexión automatizadas y mensajería destinada principalmente a campañas de marketing.

La automatización de estos eventos se suele realizar a través de *“workflows”* que ejecutan las acciones dependiendo de ciertas condiciones.

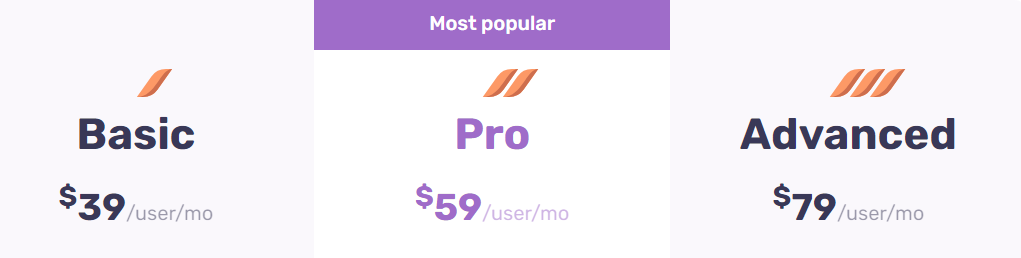
En definitiva, aunque estos servicios permiten realizar procesos de extracción de datos de perfiles de LinkedIn, el objetivo principal es ofrecer a los equipos de ventas y profesionales de marketing de una empresa una herramienta que facilite la tarea de expansión del negocio, el aumento de la captación de clientes y el aumento de ventas. Sin embargo, realizan el proceso de extracción de datos de la red social.

A continuación, se exponen algunos ejemplos de estas herramientas:

#### **Dripify**

Dripify es un software en forma de servicio que permite automatizar la extracción y exportación de datos de la plataforma de LinkedIn.

La funcionalidad principal de Dripify ofrecer funciones avanzadas de automatización de LinkedIn, como solicitudes de conexión automatizadas y mensajería. Estas funciones te permiten expandir la red del perfil de tu empresa y contactar de manera más eficiente con posibles clientes, socios o candidatos.

A continuación, una imagen con el plan de precios de Dripify:  


#### **Octopus CRM**

Octopus CRM es también una herramienta de extracción de datos de LinkedIn que te permite obtener información de perfiles de LinkedIn y automatizar diversas tareas relacionadas con la generación de “*leads”*, el alcance de ventas y las campañas de marketing.

Octopus CRM también ofrece herramientas de análisis e informes, lo que te permite realizar un seguimiento del rendimiento de tus campañas y optimizar tus esfuerzos de alcance con el tiempo.

#### **Linked Helper 2**

Linked Helper 2 es una herramienta de automatización de LinkedIn que incluye la funcionalidad de extracción de datos. Está diseñada para ayudar a empresas y particulares a automatizar sus esfuerzos de alcance y generación de clientes para su empresa a partir de perfiles de LinkedIn.

Utilizando esta herramienta, puedes extraer datos como direcciones de correo electrónico y nombres de perfiles de LinkedIn, y enviar los datos extraídos directamente a tu CRM para nutrirlo y enriquecerlo. Puedes utilizar los datos extraídos para crear listas para campañas de alcance y generación de clientes.

En definitiva, estas herramientas son útiles para la extracción de datos de LinkedIn y para su almacenamiento, sin embargo, el objetivo principal de esta herramienta es facilitar la captación de clientes potenciales para un negocio para aumentar las ventas y en consecuencia los ingresos.

A continuación, se muestra el precio de la suscripción de la herramienta:

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

Descripción generada automáticamente

### Proyectos de análisis (GitHub)

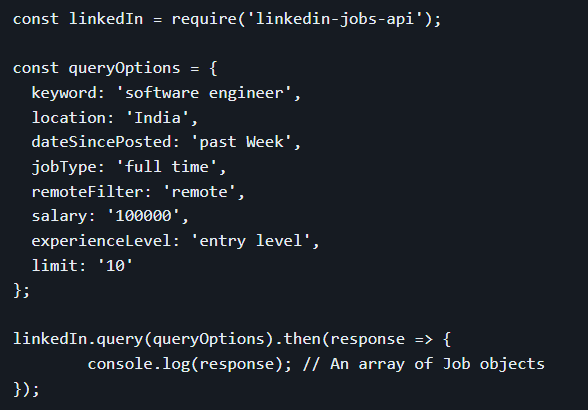
Hemos visto que en internet podemos encontrar varias herramientas y empresas que ofrecen de extracción de datos de LinkedIn, pero no encontramos ninguna que implemente la funcionalidad del análisis de los propios datos. Para buscar herramientas similares, buscamos proyectos que ofrezcan esta funcionalidad en GitHub.

GitHub es una plataforma donde los desarrolladores y equipos de desarrollo software pueden almacenar el código de sus proyectos en repositorios.

Realizando una búsqueda entre los repositorios públicos de GitHub, encontramos repositorios de código que implementan tanto la funcionalidad de extracción de datos de la web como la del proceso de análisis del mercado laboral a partir de los datos extraídos. A continuación, mostramos varios repositorios.

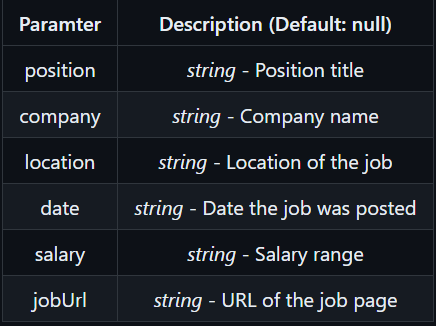
#### **Linkedin-jobs-api** de **@VishwaGauraln**

Este proyecto consiste en la implementación de una API que nos permite hacer peticiones para obtener datos de ofertas de trabajo del portal de LinkedIn. Esta implementada en JavaScript y disponible para instalar a través del gestor de paquetes npm. A continuación, se muestra un fragmento de código en el que se realiza una petición a la API:



Como podemos ver en el código anterior, el método que realiza la petición recibe un objeto de tipo JSON como parámetro. Los datos que incluyen este objeto son los parámetros que se utilizarán en la petición para obtener diferentes datos como respuestas a las peticiones.

La respuesta de la API viene en forma de array. Cada elemento del array es un objeto que corresponde a una oferta de trabajo concreta. Esta es la estructura de los objetos y los atributos que lo componen:



En resumen, esta API nos permite obtener datos de ofertas de empleo en LinkedIn de una manera sencilla y en un formato bien estructurado.

#### **Job-Market-Analysis** de **@saadabbass84**

Este proyecto consiste en la implementación de una herramienta que realiza un análisis de la oferta de trabajo en el campo del tratamiento y análisis de datos en Sydney y Melbourne.

Este proyecto está dividido en tres *Notebooks* de Jupyter:

El primero de todos se basa en la obtención de los datos y su almacenamiento en formato *CSV*. Los datos se obtienen mediante técnicas de webs scraping sobre el portal web “Seek”, en el que se publican ofertas de empleo para varios países.

El segundo realiza un preprocesado y un análisis exploratorio de los datos utilizando librerías de Python como *pandas* para el propio análisis y *Matplotlib* y *Seaborn* para la representación de gráficas.

El tercero consiste en la construcción de dos modelos predictivos de *Machine Learning*. Uno de ellos predice el salario de un empleo a partir del puesto de trabajo, el tipo de contrato y el sector laboral. El otro realiza una extracción de palabras clave usando técnicas de procesamiento del lenguaje natural.

#### **AI-Powered-Resume-Analyzer-and-LinkedIn-Scraper-with-Selenium** de **@gopiashokan**

Este proyecto es una aplicación web que pretende simplificar el proceso de búsqueda de empleo. La funcionalidad principal es una herramienta procesa un currículum que se ofrece como entrada al programa, y que es analizado mediante técnicas de procesamiento de lenguaje natural. El análisis ofrece un resumen, fortalezas, debilidades y sugerencias a añadir para mejorar el contenido del currículum.

Además, también existe la funcionalidad de extracción de datos usando Selenium. El usuario introduce un puesto de trabajo, y la herramienta automatiza la extracción de datos y los muestra en la página web.

#### **IT-Job-Market-Analyzer** de **@maks-maltsev**

Este proyecto es una aplicación web que ofrece valoraciones y conclusiones sobre las ofertas de empleo más demandadas, las competencias necesarias para los candidatos y los marcos tecnológicos más populares en el sector de las TI.

El análisis de las ofertas de trabajo consiste en la creación de varios gráficos de barras con la cantidad de ofertas con ciertas palabras clave:   
 - Gráfico con la cantidad de apariciones de un puesto de trabajo en concreto en búsquedas de empleo en sector de IT.  
 - Grafico con la cantidad de apariciones de varios frameworks de desarrollo en búsquedas de empleo.   
 - Gráfico con cantidad de apariciones de requerimientos para acceder a los puestos de trabajo que se ofrecen en las búsquedas.

Hay que recalcar que tanto de la extracción de datos como el análisis de estos se realizan sobre el mercado laboral ucraniano y en portales de empleo locales, no en LinkedIn.

## Análisis de herramientas

### Scraping

A continuación, nos centramos en las tecnologías que nos permitirán obtener los datos de LinkedIn para formar un conjunto de datos sólidos.

Web scraping es el proceso de extracción de datos e información a partir de sitios web de forma automatizada mediante la utilización de programas o scripts.

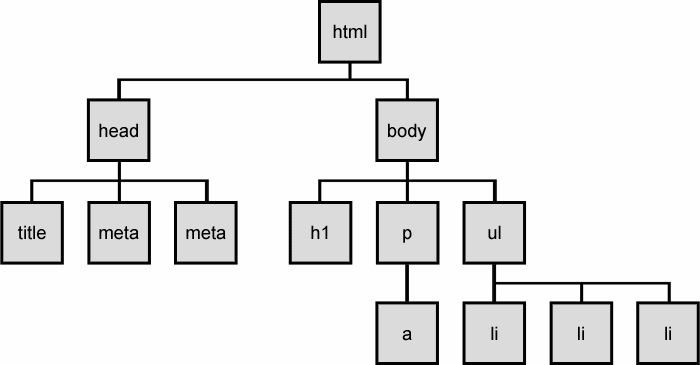
Para esta técnica de extracción de datos, hay que realizar una exploración previa de la estructura de la web, ya que es a partir del documento HTML como vamos a obtener los datos que nos interesan.

A continuación, se realiza una exposición de algunas librerías y herramientas de scraping:

#### **BeautifulSoup**

BeautifulSoup es una librería de Python que permite extraer información de contenido en formato HTML o XML. Para usarla, es necesario especificar un “*parser”*, que es responsable de transformar un documento HTML o XML en un árbol objetos Python. Esto permite, por ejemplo, que podamos interactuar con los elementos de una página web como si estuviésemos utilizando las herramientas del desarrollador de un navegador.

El árbol que genera BeautifulSoup es algo parecido a un árbol DOM (Document Object Model). Un árbol DOM sigue la siguiente estructura:



Al *“parsear”* el documento HTML se genera un árbol de objetos con clases provistas por la propia librería, y nos permite navegar por el árbol, buscar y obtener elementos y su contenido según la etiqueta, clase y otros selectores.

BeautifulSoup no permite realizar peticiones para obtener los documentos, por lo que se suele utilizar junto a la librería requests, que es la que nos permite realizar las peticiones a las webs que tienen los datos que nos interesan.

El proceso de obtención de datos sería el siguiente:  
 - Realizar una petición a una URL con la librería requests para obtener el documento HTML.   
 - *Parsear* el documento HTML con BeautifulSoup para formar el árbol de objetos.   
 - Utilizar los métodos y atributos de las clases de BeautifulSoup para navegar sobre el árbol y obtener los elementos que contienen los datos que buscamos.

Una hemos almacenado los datos podemos realizar una limpieza y post-procesado de los mismos y luego almacenarlos en un fichero o en una base de datos.

A continuación, se presentan algunas ventajas e inconvenientes al usar esta librería:

**Ventajas:** - Facilidad de uso: BeautifulSoup es conocido por su sintaxis simple y amigable con el usuario. Es fácil de aprender y utilizar, lo que lo hace una excelente elección para principiantes en web scraping.  
 - Integración con Python: BeautifulSoup se integra bien con Python. Esto facilita la combinación de BeautifulSoup con otras bibliotecas y módulos de Python para realizar tareas específicas relacionadas con el web scraping.  
 - Documentación y comunidad: BeautifulSoup tiene una documentación sólida y una comunidad activa de usuarios, lo que facilita encontrar ejemplos, tutoriales y soluciones a problemas comunes.

**Desventajas**:  
 - No aplicable en páginas web dinámicas: BeautifulSoup se utiliza comúnmente para analizar y extraer datos de páginas web estáticas, es decir, páginas web cuyo contenido no cambia dinámicamente después de que se carga la página en el navegador. Para usar BeautifulSoup sobre webs que utilizan tecnologías como JavaScript para cargar contenido de manera asíncrona o realizar cambios en el DOM después de la carga inicial de la página se requiere el uso de bibliotecas de automatización del navegador como Selenium.  
 - Falta de capacidad para realizar solicitudes HTTP: BeautifulSoup se enfoca en analizar documentos HTML y XML, pero no ofrece funcionalidades para realizar solicitudes HTTP. Esto significa que necesitas utilizar otra biblioteca, como *requests*, para descargar la página web que deseas analizar, lo que aumenta la complejidad del proceso.  
 - No es una solución completa para el web scraping: BeautifulSoup por sí solo no proporciona las capacidades avanzadas necesarias para lidiar con desafíos comunes en el web scraping, como la gestión de sesiones, la manipulación de cookies, la detección de CAPTCHAs o la gestión de proxies.  
 - No es tan rápido como algunas alternativas: BeautifulSoup es una biblioteca de análisis de documentos, por lo que, en comparación con algunas bibliotecas de web scraping más especializadas como Scrapy, puede no ser tan eficiente en términos de velocidad y rendimiento en proyectos de web scraping a gran escala.

En resumen, BeautifulSoup es una herramienta popular y útil para el web scraping debido a su facilidad de uso, integración con Python, capacidad para analizar documentos HTML y XML, y flexibilidad en la extracción de datos. Aunque tiene algunas limitaciones en proyectos más complejos, sigue siendo una elección sólida para tareas de web scraping más simples o cuando la simplicidad y la legibilidad del código son prioritarias.

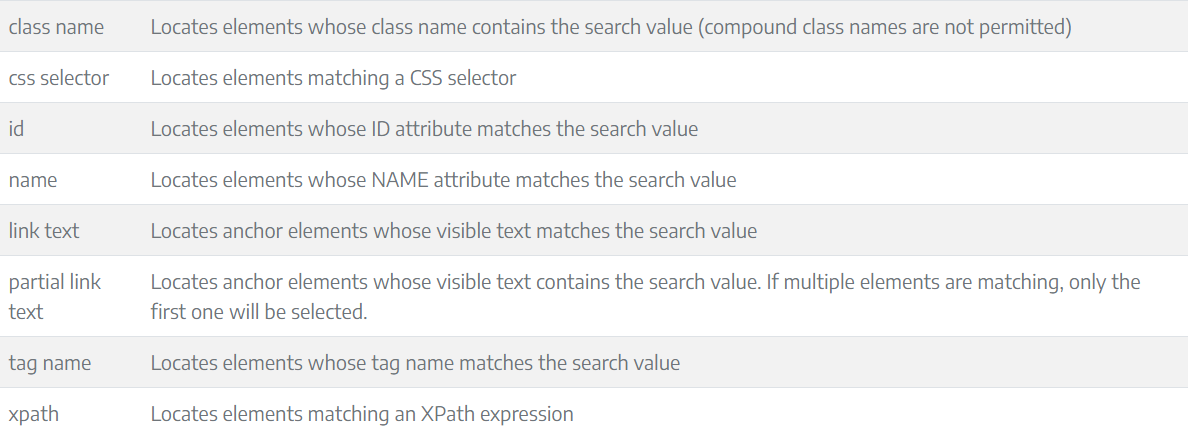
#### **Selenium**

Selenium es una herramienta de código abierto creada originalmente para la automatización de testing web. Soporta varios de los navegadores más usados del mercado y se puede programar en multitud de lenguajes. La librería nos permite programar un script que de instrucciones específicas a un navegador para automatizar su funcionamiento.

Sin embargo, adicionalmente, Selenium integra una variedad de herramientas y bibliotecas que hacen posibles la implementación de otras funcionalidades como puede ser la extracción de datos de sitios web.

La posibilidad de automatizar el funcionamiento del navegador resulta muy útil para cargar webs con contenido dinámico. Esto combinado con los métodos de selección de elementos HTML hacen de Selenium una biblioteca muy completa para realizar web scraping.

Los métodos que nos permiten acceder a los elementos HTML que nos interesan son los llamados selectores o *“locators”*. Estos son los selectores que Selenium nos ofrece para obtener elementos del DOM.



Como se puede ver en la captura, tenemos varios selectores disponibles para acceder a la información de diferentes maneras. Podemos seleccionar elementos según su clase, su id o su atributo *“name”*. Para selecciones mas flexibles podemos utilizar los selectores *“css selector”* o *“xpath”*. Esto nos permite seleccionar elementos con ciertas relaciones o estructuras determinadas.

A continuación, se presentan algunas ventajas e inconvenientes al usar esta librería:

**Ventajas:** - Interacción con páginas web dinámicas: Una de las mayores ventajas de Selenium es su capacidad para interactuar con páginas web dinámicas que utilizan JavaScript. Esto permite hacer clic en botones, llenar formularios, navegar por páginas, esperar a que se cargue contenido dinámico, y más.  
 - Flexibilidad y compatibilidad: Puedes realizar web scraping en una variedad de sitios web, independientemente de la tecnología utilizada para crearlos, ya que Selenium automatiza y simula la interacción con el navegador sin tener en cuenta la base sobre la que está construido.  
 - Amplia comunidad y documentación: Al igual que en el caso anterior, Selenium tiene una gran comunidad activa de usuarios y una amplia documentación disponible en línea. Esto implica facilidad a la hora de encontrar soluciones a problemas comunes y obtener ayuda de otros desarrolladores.

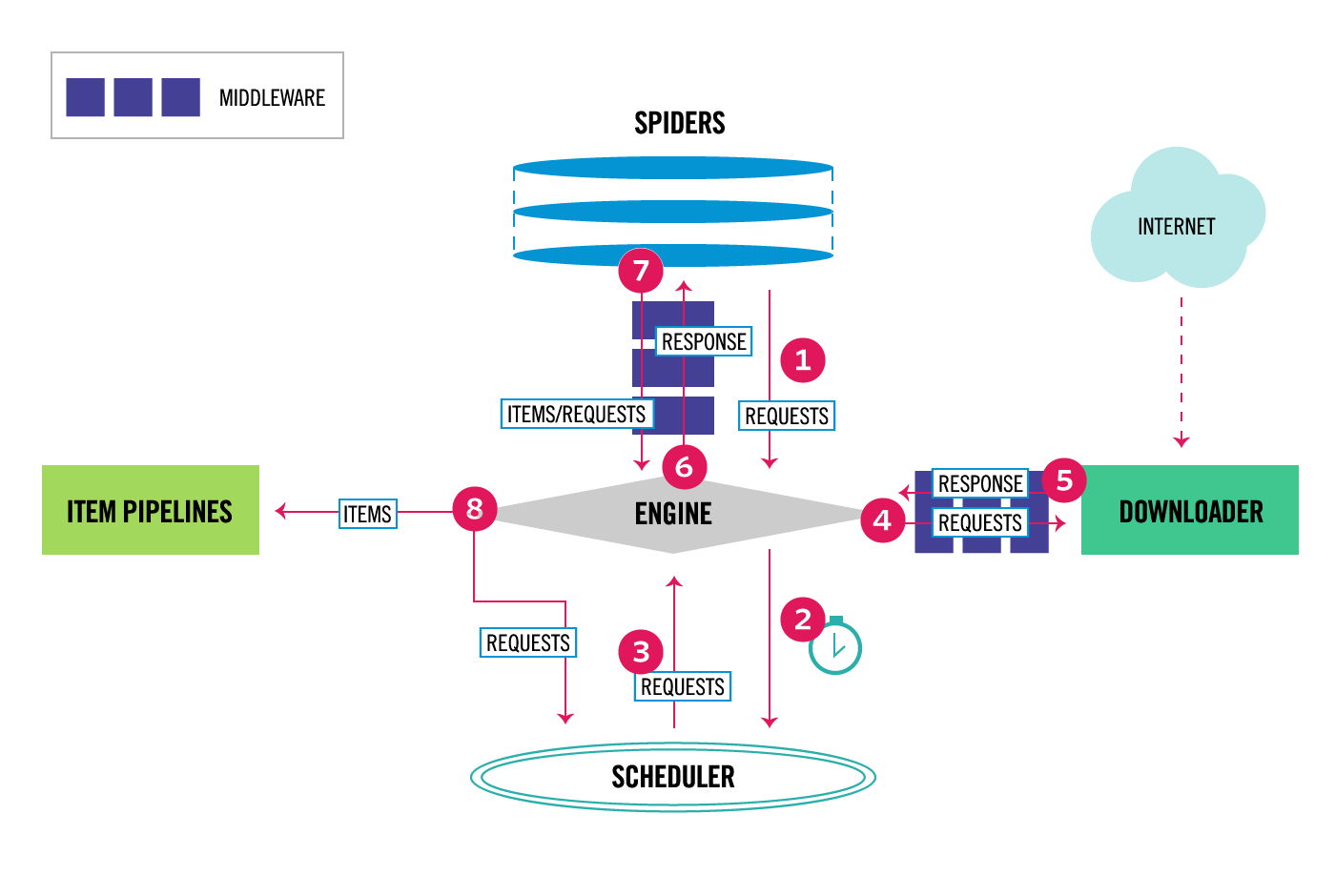
**Desventajas:** -Alto coste de mantenimiento: Las actualizaciones de los y sitios web de los navegadores pueden afectar la estabilidad de los scripts de Selenium, lo que puede requerir mantenimiento continuo.  
 - Posible detección por parte del sitio web: Al utilizar Selenium, el sitio web puede detectar la automatización y tomar medidas para evitar el web scraping. Esto puede llevar a bloqueos IP o restricciones.  
 - Velocidad y recursos: Debido a la necesidad de simular la interacción de un navegador real, Selenium puede ser más lento que otras herramientas de web scraping más livianas. Además, consume más recursos de CPU y memoria.

#### **Scrapy**

Scrapy es un framework de extracción de datos en la web de alto nivel y rápido, utilizado para explorar sitios web y extraer datos estructurados de sus páginas. Ofrece un conjunto de funcionalidades y bibliotecas que facilitan la creación de arañas web (web spiders) personalizadas para obtener datos de sitios web de manera eficiente y organizada.

Actualmente es la herramienta más potente disponible para hacer web scraping. No solo nos permite extraer los elementos del HTML y su contenido, sino que sus varios componentes mantienen todas las necesidades cubiertas evitando tener que depender de otras librerías. Algunas de las funcionalidades que implementan los componentes que forman el framework de Scrapy son el envío, recepción y gestión de peticiones y respuestas, el procesamiento del documento y extracción los datos, o la limpieza, formateo y persistencia de datos.

Al tratarse de un framework, existe una arquitectura predefinida que proporciona un marco para el desarrollo de una aplicación. Vamos a ver una descripción general de la arquitectura de Scrapy:



En el diagrama se muestran los componentes del framework y el flujo de datos. Los componentes son los siguientes:  
 - Motor (Engine): El motor de Scrapy se encarga de controlar el flujo de datos entre todos los componentes del sistema y de activar eventos cuando se producen determinadas acciones.  
 - Planificador (Scheduler): El planificador recibe peticiones del motor y las pone en cola para volver a proporcionarselas (también al motor) cuando éste las solicite.  
 - Downloader: El *“downloader”* se encarga de buscar y recibir los documentos HTML de las páginas web y enviarlas al motor que, a su vez, las envía a las arañas.  
 - Arañas (Spiders): Las arañas son clases personalizadas para una aplicación en concreto escritas por los desarrolladores para analizar respuestas y extraer elementos de ellas o peticiones adicionales a seguir.   
 - Tuberías de objetos (Item Pipelines): Las tuberías de objetos se encargan de procesar los objetos una vez que han sido extraídos por las arañas. Las tareas típicas incluyen la limpieza, la validación y la persistencia (como el almacenamiento del elemento en una base de datos).  
 - Middlewares: Los *“middlewares”* son scripts escritos por los desarrolladores de la aplicación que actúan a modo de *“hook”* y que permiten a los desarrolladores extender y modificar el flujo de ejecución de la aplicación. De esta forma podemos realizar ciertas acciones o tratamientos sobre los datos, peticiones o respuestas.   
Existen tanto *“spider middlewares”* como *“downloader middlewares”*. Los *“spider middlewares”* se utilizan para tareas como el post-procesado de la salida de nuestras arañas o para controlar excepciones de estas. Por otro lado, los *“downloader middlewares”* nos permiten cambiar parámetros sobre peticiones y respuestas, lo que resulta útil para lidiar con los obstáculos o bloqueos que podamos encontrar.

Todo esto hace que la elección de Scrapy sea la herramienta más potente para realizar procesos de extracción de datos de la web en proyectos de gran tamaño y complejidad.

## Elección de herramientas

### Selenium y BeautifulSoup

Tras haber realizado un repaso a las herramientas que nos permiten aplicar web scraping, se utilizará la combinación de Selenium y BeautifulSoup para la extracción de datos. La motivación principal para la elección de estas tecnologías es la simplicidad de uso, a parte del interés personal por aprenderlas. Además, esta combinación cubre todas las necesidades al suplir el problema de la carga dinámica de la web de LinkedIn con Selenium.

A continuación, se realizará un tutorial básico del uso de estas herramientas:

Comenzamos con Selenium, la herramienta de automatización que nos permite simular el funcionamiento de un navegador para interactuar con la web y cargar contenido dinámico de la página.

En primer lugar, es necesaria la instalación de la librería a través del gestor de paquetes de Python *pip*. Instalamos la librería ejecutando el siguiente comando:

**pip install selenium**

Una vez realizada la instalación podemos empezar a implementar el código que automatizará el funcionamiento del navegador por nosotros, pero antes, vamos a ver con algo más de detalle cómo funciona la librería.

#### **Web Driver**

Los *Web Drivers* son programas que permiten a Selenium interactuar con un navegador específico. Son esenciales ya que son los que permiten la automatización del navegador. Resumidamente, su función es actuar como intermediarios entre Selenium y el navegador. Cada navegador tiene su propio driver (por ejemplo, ChromeDriver para Google Chrome, GeckoDriver para Firefox, etc.). En nuestro caso vamos a utilizar Chrome.

La instalación de Selenium nos provee de un módulo que contiene todas las clases y scripts necesarios para hacer uso de un Web Driver en nuestro código. Para utilizarlo, únicamente hay que realizar la importación del módulo *webdriver*:

**from selenium import webdriver**

Una ver importado el módulo, podemos crear una instancia de la clase del driver que vayamos a utilizar:

**driver = webdriver.Chrome(options=chrome\_options)**

En la línea anterior creamos una instancia del driver de Chrome. Podemos notar que se ha añadido un parámetro “*options”* al constructor de la instancia. Este es uno de los parámetros que puede recibir el constructor, y su contenido determinará el funcionamiento del navegador.

Si no especificamos nada, ChromeDriver usara sus opciones por defecto. Podemos crear una instancia de la clase “*ChromeOptions”* para añadir los parámetros que creamos convenientes de la siguiente forma:

**chrome\_options = webdriver.ChromeOptions()**

A continuación, se muestran varias de las opciones que podemos configurar:

- No mostrar interfaz gráfica: Al añadir el parámetro “*headless”* a las opciones, hacemos que el navegador ejecute sus instrucciones de automatización sin mostrar interfaz gráfica. Esto es útil porque implica un ahorro de recursos y mejora del rendimiento. Cuando desarrollamos el script de automatización, puede ser conveniente ver como el navegador ejecuta sus instrucciones en la interfaz gráfica, pero esto no es necesario cuando el código de la aplicación está en producción en un servidor (muchos de ellos no tienen interfaz gráfica). Podemos añadir esta opción con la siguiente línea de código.

**chrome\_options.add\_argument("--headless")**

- No obtener imágenes: Desactivar la carga de imágenes, lo que puede acelerar la velocidad de carga de la página y reducir el consumo de ancho de banda. En casos de web scraping, es posible que nos interese no activar esta opción, debido a que obtener imágenes es una práctica común. La opción que añadir para no imágenes es:

**chrome\_options.add\_argument("--blink-settings=imagesEnabled=false")**

- Utilizar proxy: Las opciones del driver de Chrome nos permite utilizar un proxy en las peticiones que realiza el navegador. El uso de proxys es común para temas de privacidad y simulación de localizaciones. En el ámbito del web scraping, una técnica muy común es la rotación de proxys. Esto nos permite evitar bloqueos de IP y debido a la realización de peticiones masivas. Esta opción nos ayuda a sortear estos bloqueos. La opción que añadir para utilizar proxys en las peticiones es:

**chrome\_options.add\_argument("--proxy-server={PROXY\_SERVER}")**

donde PROXY\_SERVER es la dirección del servidor proxy.

- Especificar un User-Agent (UA): El User-Agent es un parámetro que un navegador web envía al servidor web para identificarse. La cadena del User-Agent proporciona información sobre el navegador, la versión y, en algunos casos, el sistema operativo y otros detalles relacionados con el cliente. En el contexto de uso de Selenium para web scraping se pueden alternar varios User-Agents para simular que las peticiones se realizan desde diferentes navegadores o dispositivos y, en algunos casos, evitar la detección de actividad automatizada en por parte de sitios web. Podemos establecer un UA determinado de la siguiente forma:

**chrome\_options.add\_argument(f"--user-agent={USER\_AGENT}")**

donde USER\_AGENT es la cadena de texto que identifica el UA a utilizar.

- Gestión de cookies: Otra de las opciones que podemos configurar son las cookies del navegador. Podemos definir una cookie en nuestro código de esta forma:



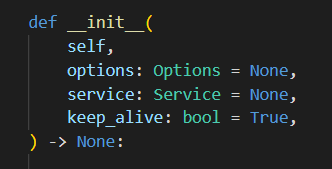
Una vez la cookie está definida la añadimos de la siguiente manera:

**chrome\_options.add\_cookie(cookie)**

Esto es muy útil ya que el análisis de las cookies por parte del servidor que responde a las peticiones de los navegadores es de las prácticas más utilizadas para bloqueos a programas de extracción de datos automatizados.

Estos son algunos de los muchos parámetros configurables en el driver de Selenium.

Si nos fijamos en el código que implementa el constructor del driver del navegador, vemos otros dos parámetros:



El parámetro *“service”* es un objeto recibido por el navegador que permite establecer pautas adicionales, como modificar la configuración de logs entre otros. Por otro lado, el parámetro *“keep\_alive”* especifica si la conexión TCP entre navegador y servidor se mantendrá abierta al realizar peticiones HTTP .

Tras revisar configuraciones previas y la creación de una instancia de un navegador de Selenium, pasamos a ver como realizar una petición y métodos de automatización del navegador.

#### **Navegación con Selenium**

Al crear una instancia de un navegador con Selenium, la primera acción a realizar es realizar una petición de una URL en concreta, de manera que podemos cargar y navegar sobre la web a la que corresponde dicha URL. Para realizar una petición utilizamos el método *“get”*:

**driver.get("https://www.linkedin.com/jobs/")**

Otras opciones de navegación típicas son retroceder y avanzar en el historial de navegación.

**driver.back()**

**driver.forward()**

O hacer *“scroll”* hacia el final de la página. Esto es especialmente útil cuando ciertos elementos de la web se cargan dinámicamente al hacer scroll hasta el final de la página. Podemos realizar esta acción mediante el siguiente comando:

**driver.execute\_script("window.scrollTo(0, document.body.scrollHeight);")**

Una vez cargado el documento de la web, podemos identificar los elementos que lo componen haciendo uso del método *“find\_element”* junto a la clase *“By”*. El método “find\_element” devuelve un objeto de la clase *“WebElement”* de Selenium que corresponde a un elemento del documento HTML concreto. La clase *“By”* nos sirve para especificar el valor de que atributo se utilizará para la búsqueda. En la siguiente imagen vemos la implementación de la clase “By” en la que podemos ver todos los selectores disponibles:

Texto

Descripción generada automáticamente

A continuación, se muestran ejemplos para de selección de elementos con el método mencionado:

**element1 = driver.find\_element(By.ID, “id”)**

**element2 = driver.find\_element(By.NAME, “name”)**

**element3 = driver.find\_element(By.CLASS\_NAME, “class”)**

**element4 = driver.find\_element(By.TAGNAME, “tag”)**

Hay que resaltar que el selector más potente es Xpath. Este nos permite seleccionar elementos dada una expresión Xpath. Las expresiones Xpath describen la ubicación de un elemento en un documento XML o HTML, y pueden especificar rutas a través de la estructura del documento. Esta herramienta es más potente y versátil ya que nos permite seleccionar teniendo en cuenta el valor de varios atributos.

Una vez tenemos un elemento seleccionado, podemos realizar acciones sobre él. Dependiendo del elemento que hayamos obtenido podremos realizar unas acciones u otras. Estos son algunos de los métodos más comunes

**element.send\_keys("Texto a escribir")**

**element.clear()**

Esto nos permite introducir una cadena de texto en un elemento de entrada y limpiar el contenido del campo de entrada respectivamente.

También podemos simular un click en un botón con el método *”click”*.

**element.click()**

Se ha mencionado que los objetos devueltos por los selectores son de la clase *“WebElement”*. Una de las propiedades de esta clase es *“text”* y en ella se almacena el contenido de un elemento HTML. Esta propiedad es muy utilizada en el contexto del web scraping, ya que en este texto se incluyen la mayoría de los datos que interesan extraer. Podemos acceder al contenido de un elemento HTML utilizando la propiedad *“text”* de esta forma:

**texto = element.text**

Sin embargo, en este proyecto, no se hará uso de esta propiedad, ya que, para extraer los datos, se utilizarán las herramientas provistas por BeautifulSoup.

En primer lugar, realizaremos una breve introducción sobre un caso reciente en el que LinkedIn y la empresa HiQLabs se vieron involucrados en procesos judiciales por temas relacionados con la extracción de datos de la web de LinkedIn.

HiQ es una compañía de base tecnológica cuyo principal servicio consiste en proporcionar a empleadores y personal de recursos humanos una herramienta para determinar los riesgos de rotación de personal de manera anticipada, identificar problemas relacionados con las habilidades del personal y colaborar con acciones que permitan resolver el problema de retención del talento.

HiQ recolectaba parte de los datos de las bases de perfiles públicos de Linkedin mediante técnicas de web scraping, que son definidas como técnicas que permiten la captura automatizada de todo tipo de información disponible en la red, para uso y tratamiento con fines investigativos, comerciales, publicitarios, etc.

Como argumento legal, LinkedIn se aferra a la Ley de abuso y fraude informático de Estados Unidos (CFAA, por sus siglas en inglés). Esta última prohíbe acceder a una computadora sin autorización (o en exceso de la autorización que podría haber sido obtenida) para obtener información protegida de esta computadora. Claramente, este supuesto es bastante lejano al caso de web scraping de HiQ Labs, ya que los datos que recolectan son accesibles de manera pública.

Ante el fallo a favor de la empresa HiQ Labs, LinkedIn emitió una orden de cese y desistimiento en 2017. La carta decía que HiQ estaba violando el acuerdo de usuario de LinkedIn, así como las leyes federales y de California, incluido el CFAA, entre otros. LinkedIn también dijo que técnicamente bloquearía los esfuerzos de HiQ para extraer datos de su web.

HiQ demandó por una orden judicial preliminar contra LinkedIn y ganó a nivel de tribunal de distrito. El tribunal ordenó a LinkedIn que permitiera el acceso de HiQ al contenido nuevamente. LinkedIn apeló al Noveno Circuito.

Los problemas entre empresas de tecnología por la utilización de técnicas de web scraping han venido creciendo de la mano de los desarrollos tecnológicos. En Europa hay varios pronunciamientos protegiendo las empresas de actos que afectan la libre competencia, la violación a la propiedad intelectual sobre bases de datos y el derecho fundamental de las personas al habeas data. En Estados Unidos, el tema se ha tratado alrededor de la violación de los términos y condiciones.

OTRAS DUDAS