Universidad Politécnica
de Madrid

Escuela Técnica Superior de
Ingenieros Informáticos

Grado en Ingeniería Informática

Trabajo Fin de Grado

Automatización de ingesta de datos y publicación en web de datos de educación

Autor: Miguel Fernández Díaz

Tutor(a): Elena Montiel Ponsoda, Víctor Rodríguez Donciel

Madrid, enero 2024

Este Trabajo Fin de Grado se ha depositado en la ETSI Informáticos de la Universidad Politécnica de Madrid para su defensa.

Trabajo Fin de Grado
Grado en Ingeniería Informática
Título: Automatización de ingesta de datos y publicación en web de datos de educación
Mes Año: enero, 2024

Autor: Miguel Fernández Díaz

Tutor:

Elena Montiel Ponsoda
LINGÜÍSTICA APLICADA A LA CIENCIA Y A LA TECNOLOGÍA
ETSI Informáticos
Universidad Politécnica de Madrid

Víctor Rodríguez Doncel
INTELIGENCIA ARTIFICIAL
ETSI Informáticos
Universidad Politécnica de Madrid

Resumen

El auge de la inteligencia artificial en los últimos años ha provocado gran cantidad de cambios en el mundo laboral, empujando a los trabajadores a expandir su formación y adquirir nuevas competencias para adaptarse a estos cambios.

Numerosas son las iniciativas y proyectos que se están llevando a cabo para intentar paliar este problema, desde análisis del mercado laboral para identificar trabajos duraderos hasta portales de recomendación de cursos de formación universitarios. En este contexto, el presente trabajo pretende ser una contribución a esas iniciativas cuyo objetivo es facilitar el acceso a formación a trabajadores y empresas. Con este proyecto se propone un punto único de acceso a las ofertas formativas de los mayores portales de cursos online, y un algoritmo para la estructuración y visualización de los datos de forma eficiente.

Mediante un proceso de *scraping* automatizado se recopilan semanalmente datos de cada curso de los portales *Coursera*, *edX* y *Udemy*. Estos datos se procesan para ofrecer un resumen en forma de diagrama de acuerdo con los conceptos de la taxonomía Bloom. De esta forma se ofrece un formato constante entre todas las distintas fuentes que existen, facilitando la comprensión de la información presentada.

Esta página está disponible en: https://ai4labour.linkeddata.es/miguel

Palabras clave: *web scraping*, taxonomía Bloom, automatización, *cron*, diseño de páginas web, Python, HTML, CSS, Scrapy, BeautifulSoup, Flask.

Abstract

In recent years, the rise of artificial intelligence has caused a great number of changes in the labor market, creating the need for workers to work on their education and add to their skills to keep up with these changes.

There are a lot of ongoing initiatives and projects trying to deal with this issue, from an analysis of the labor market in order to find long-lasting jobs to a university course recommendation portal. In this context, this project aims to contribute to those initiatives aiming to provide workers and companies with an easy access to information. With this proposal comes an access point to what some of the biggest online education platforms have to offer, as well as an algorithm that creates a common structure for data from every source and a clear and efficient way of displaying it.

With an automated scraping process, the data from the learning platforms *Coursera*, *edX* and *Udemy* is compiled weekly. This information is processed in order to provide a summary in the form of a diagram in accordance with the principles of the Bloom taxonomy. This way the user is provided with a consistent format in data from all available sources, improving the understanding of the presented information.

This page is available at: https://ai4labour.linkeddata.es/miguel

Key words: web scraping, Bloom taxonomy, automation, cron, web page design, Python, HTML, CSS, Scrapy, BeautifulSoup, Flask.

Tabla de contenidos

1 Introducción 1
1.1 Contexto 1
1.2 Objetivos 1
1.3 Estado del arte 2
1.3.1 Scraping ..... **¡Error! Marcador no definido.**
1.3.2 Plataformas de cursos 3
1.3.2.1 Coursera 3
1.3.2.2 edX 5
1.3.2.3 Udemy 6
1.3.3 Publicación de datos 7
1.4 Contribuciones 9
1.4.1 Scraping ..... **¡Error! Marcador no definido.**
1.4.2 Publicación de datos en la página web 10
2 Desarrollo 11
2.1 Scraping ..... **¡Error! Marcador no definido.**
2.1.1 Coursera 11
2.1.2 edX 14
2.1.3 Udemy 16
2.2 Publicación de datos 19
2.2.1 Resultados de scraping 19
2.2.2 Lógica de la página web 20
2.2.3 Diseño de la página web 23
3 Resultados y conclusiones 33
4 Análisis de Impacto 34
5 Bibliografía 36
6 Anexos 39
6.1 Anexo 1 39
6.2 Anexo 2 41

1 Introducción

1.1 Contexto

En los últimos años, especialmente desde la pandemia de COVID-19, la inteligencia artificial (IA) se ha convertido en una parte vital del mundo laboral. Estudios detallados en los artículos "*AI4Labour: Reshaping labour force participation with artificial intelligence*"[1] y "*Jobs lost, jobs gained: What the future of work will mean for jobs, skills, and wages*"[2] estiman que en 2030 aproximadamente el 30% de horas trabajadas a nivel mundial podrían ser automatizadas mediante IA, dejando multitud de trabajos obsoletos y a sus trabajadores sin saber cómo adaptarse.

Entre las opciones disponibles para estos trabajadores está la de encontrar formas de adquirir nuevas competencias enfocadas a aquellos trabajos de los que la IA no puede hacerse cargo. Sin embargo, la gran cantidad de opciones disponibles puede ser tediosa e incluso abrumadora para alguien que necesita soluciones lo antes posible.

Existen proyectos como AI4LABOUR[3] que pretenden contribuir a la solución a este problema mediante un análisis del mercado laboral para elaborar una predicción de qué trabajos tendrán un futuro sólido y cuáles, por el contrario, están destinados a ser realizados por IA.

Un problema que surge buscando cursos orientados a estos trabajos es lo dispersa que está la información. Existen una gran cantidad de portales de cursos *online* con miles de cursos diferentes cada uno, dificultando la comparación de opciones y por lo tanto la toma de decisiones.

Navegando por las distintas páginas se descubre que la información de los cursos está además presentada en formatos diferentes, complicando la comparación directa entre distintos portales.

Este trabajo, enmarcado en el contexto de AI4LABOUR, ofrece un único punto de acceso en el que cotejar las diferentes ofertas educativas en distintos portales de educación. Esto se consigue mediante un análisis de distintas plataformas de aprendizaje en línea a través de un proceso de *scraping* automatizado, su procesado y publicación a una página web para, de esta forma, facilitar a los individuos la toma de decisiones sobre su formación y desarrollo profesional mediante información constantemente actualizada y comprensible.

1.2 Objetivos

Como se ha explicado en el apartado anterior, el problema al que pretende poner fin este proyecto es la dificultad que supone buscar e interpretar la gran cantidad de información disponible en los distintos portales de cursos. Este trabajo aspira a facilitar ese proceso proporcionando en un único punto de acceso la información de forma sencilla y clara con los siguientes pasos:

- Diseño e implementación de sistema automatizado de *scraping* o extracción de información de sitios web.
- Análisis de los datos extraídos de cada una de las páginas (*Coursera*[4], *edX*[5] y *Udemy*[6]) y estructuración para su consumo por terceras aplicaciones.
- Diseño e implementación de interfaz de visualización de los datos extraídos.

1.3 Estado del arte

1.3.1 Scraping

Los principales módulos de Python empleados para el *scraping* de páginas web considerados para este proyecto son BeautifulSoup, Selenium y Scrapy[7].

BeautifulSoup[8] es una librería utilizada para obtener información de archivos HTML y XML. En el ámbito de *web scraping* lo que hace es recolectar el código asociado a la página web asignando a cada elemento una etiqueta concreta y un método para ser extraído. Como tal, BeautifulSoup es solo una herramienta para obtener información estructurada de una página web que necesita de otras librerías para realizar el proceso de *scraping*, llevando a una gran cantidad de código y un tiempo de ejecución lento comparado con las otras opciones.

Selenium[9] es otra librería utilizada para la obtención de datos de páginas web dinámicas. Está diseñado para simular la interacción humana y extraer los datos una vez se haya cargado la página completamente. Su principal característica es que permite emular el comportamiento humano en navegadores, por ello tiene gran potencial no solo cuando se trata de *scraping* sino también a la hora realizar otras tareas como testear aplicaciones web. Para utilizarlo es necesario tener instalados en la máquina local algún navegador ("Chrome", "Firefox", etc.) y un *webdriver* (utilizado para manejar el navegador). A la hora de ejecutar el programa se crea una nueva instancia del navegador, consumiendo más recursos que Scrapy y BeautifulSoup que se ejecutan enteramente desde la terminal de comandos.

Scrapy[10] cuenta con la ventaja de que puede mandar peticiones asíncronas, consiguiendo una velocidad de ejecución considerablemente mayor a la de los demás métodos. También tiene la ventaja de que puede ser utilizado para la obtención de datos a gran escala de forma eficiente y sin ocupar mucha memoria. Ofrece una estructura flexible y efectiva no solo para *web scraping* sino para otros ámbitos como *data mining* o testeo de aplicaciones.

Para este proyecto la librería elegida es Scrapy, por su eficiencia, su velocidad y su flexibilidad. Sin embargo, dada la naturaleza del portal de cursos *Udemy*, se ha decidido reutilizar el código original con BeautifulSoup (razones explicadas en el apartado 1.4.1 Contribuciones).

A la hora de automatizar procesos existen varias tecnologías, todas útiles y eficientes en su propio ámbito y vitales a la hora de diseñar un sistema de *scraping* automático y eficiente:

- *Workflow Automation*: Secuencias de pasos según una lógica predefinida. Herramientas como *Zapier* o *Microsoft Power Automate* permiten diseñar, ejecutar y gestionar flujos de trabajo, asignando tareas, estableciendo reglas y facilitando la colaboración entre equipos. Siendo este proyecto individual se decide no aplicar esta metodología.
- Automatización Robótica de Procesos (RPA): Uso de software especializado (bots) para automatizar tareas repetitivas y reglas predefinidas en sistemas informáticos, emulando la interacción humana con las interfaces de usuario. Esto se consigue mediante herramientas como UiPath y BluePrism. Dado que estas herramientas son de pago y el objetivo de la automatización es la ejecución de *scripts*, se descarta esta opción.
- Automatización cognitiva: Uso de la inteligencia artificial y el aprendizaje automático para automatizar tareas que requieren análisis, toma de decisiones y procesamiento de información compleja, como la extracción

de datos no estructurados. Para este proyecto solo se necesita ejecutar de forma automática *scripts* existentes, se decide buscar otra herramienta.
- *Scripting* y Programación: Automatización a través de scripts de Python, JavaScript o PowerShell. Útil para tareas complejas, pero más accesible y manejable que los métodos anteriores. Sin embargo, la ejecución se realiza en una máquina virtual Linux, por lo que una solución más específica es preferible.
- Automatización de tareas en sistemas de TI: Automatización de tareas administrativas y de mantenimiento de sistemas con herramientas como c*rontab* en Linux o "Programador de tareas" en Windows.

Como ya se ha mencionado, este proyecto está desarrollado enteramente en una máquina virtual Linux, por lo que la mejor forma de automatizar los scripts de *scraping* es mediante *crontab*, herramienta nativa de Linux y, por lo tanto, la más eficiente para este trabajo. De esta forma, se ejecutan los *scripts* una vez por semana para garantizar que la información esté lo más actualizada posible.

1.3.2 Plataformas de cursos

La variedad de estructuras y tipos de datos a través de los siguientes portales hace imperativa la implementación de un sistema de procesado de datos para presentar un único formato, así como la existencia de tres *scripts* de *scraping* diferentes, uno para cada portal.

1.3.2.1 Coursera

Coursera es un portal de cursos *online* fundado en 2012 que trabaja junto con universidades y otras organizaciones para ofrecer cursos relevantes, flexibles y relevantes para la búsqueda de empleo. Es utilizado por empresas buscando afrontar los cambios que presenta la nueva economía digital; por universidades tratando de adaptarse a la modalidad híbrida de enseñanza y por gobiernos para hacer frente a la creciente cifra de desempleo a manos de la automatización de procesos.

Con 118 millones de usuarios en 2022, es uno de los mayores portales de educación disponibles.

Ofrece una gran variedad de cursos que con diferentes contenidos, precios y tiempo aplicado:

- Proyectos guiados: enfocados a la obtención de conocimientos específicos a un trabajo o herramienta de empresa lo más rápido posible (1-2 horas).
- Cursos: contenido más detallado cuyo objetivo principal es aprender algo nuevo (4-12 horas).
- Especializaciones: enfocadas a una habilidad específica (1-3 meses).
- Certificados profesionales: titulaciones orientadas a la búsqueda de empleo (1-6 meses).
- Certificados *MasterTrack* ®: además de un título proporcionado por una universidad, se obtienen créditos universitarios (4-7 meses).
- Grado: licenciatura o máster (2-4 años)

Para el proceso de *scraping* se consideran todos los tipos de cursos.

La forma en que *Coursera* presenta sus datos es la siguiente:

- En primer lugar se especifica el nombre e idioma del curso.

- Después se establecen las habilidades que se obtendrán una vez finalizado en forma de lista.
- El siguiente apartado es la descripción del curso, donde se explica detalladamente la información relevante en un elemento "*rc-TogglableContent about-section collapsed*". Este apartado presenta la dificultad añadida de que parte de la descripción está oculta en una sección "*aria-hidden*", accesible a través de un botón "Averiguar más". Inicialmente tiene valor *true* y al pulsar dicho botón adopta el valor *false*.
- Finalmente aparecen los módulos en los que se divide el curso, tampoco accesibles a primera vista, contenidos en una sección "*AccordionRoot*" cada uno con un botón "Detalles del módulo".

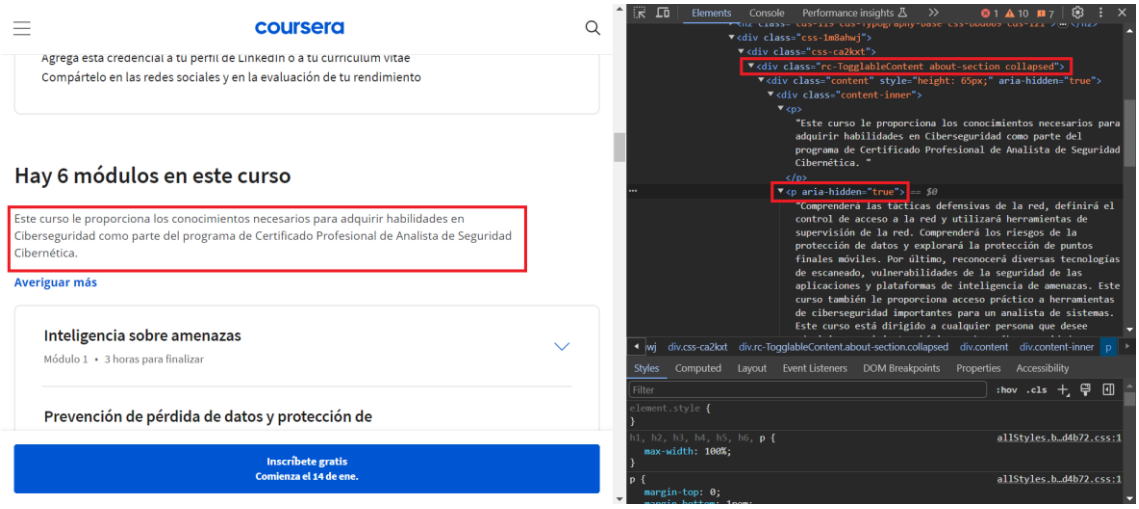

Ilustración 1: Ejemplo estructura de "Descripción" sin presionar el botón "Averiguar más" en https://www.coursera.org/learn/ibm-cyber-threat-intelligence?specialization=security-analyst-fundamentals.

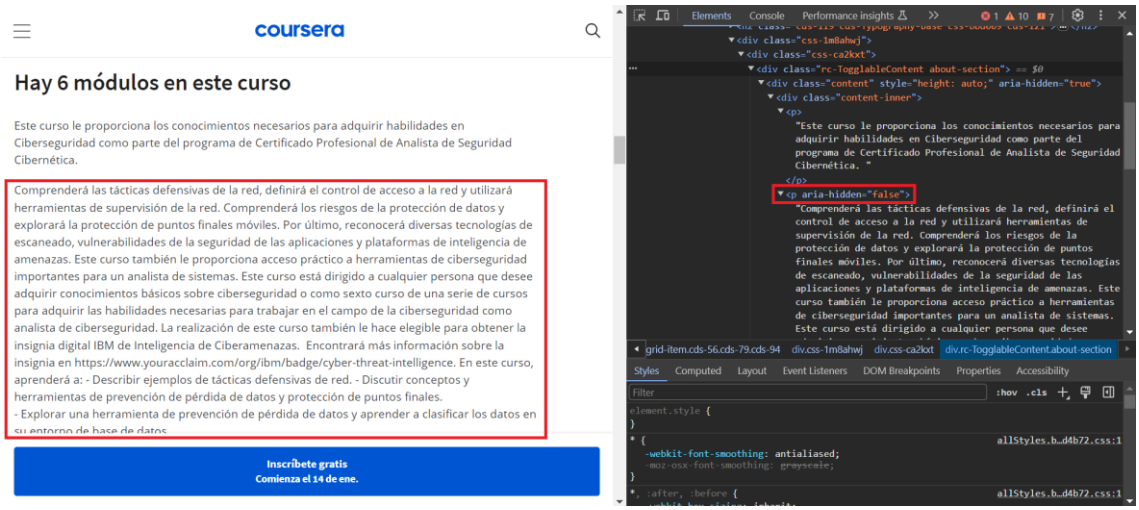

Ilustración 2: Ejemplo estructura de "Descripción" sin habiendo presionado el botón "Averiguar más" en https://www.coursera.org/learn/ibm-cyber-threat-intelligence?specialization=security-analyst-fundamentals.

1.3.2.2 edX

La plataforma *edX* fue fundada por la universidad de Harvard y comprada por la empresa 2U en 2021. Está asociada con universidades como Harvard y Princeton y organizaciones como *"Linux Foundation"*[11] y "*Web 3 Foundation*"[12].

Ofrece a más de 76 millones de usuarios un lugar en el que desarrollar competencias y fomentar la educación[13].

Sus ofertas *online* son:

- Licenciaturas
- Certificados
- Doctorados
- Másteres
- *MicroBachelors* ®: enfocados a la preparación para licenciaturas.
- *MicroMasters*®: ofrecen la posibilidad de obtener créditos universitarios, reduciendo así el tiempo y el coste de un máster.
- Certificados profesionales
- Programas *XSeries*: cursos avanzados de temas actuales y con alta demanda.

En este caso también se tienen en cuenta todos los tipos de cursos para el proceso de *scraping*.

En cuanto a la distribución de sus datos, *edX* está organizada de la siguiente manera en orden de aparición de principio a final de la página:

- Resultados de aprendizaje en forma de lista.
- Descripción del curso, oculta en una clase "*pgn_collapsible p-2 collapsible-card is-open*" dentro de "*Program overview*" solo accesible pulsando el botón ubicado a su derecha.
- Los temas tratados a lo largo del curso en una lista de clase "*pathway*".

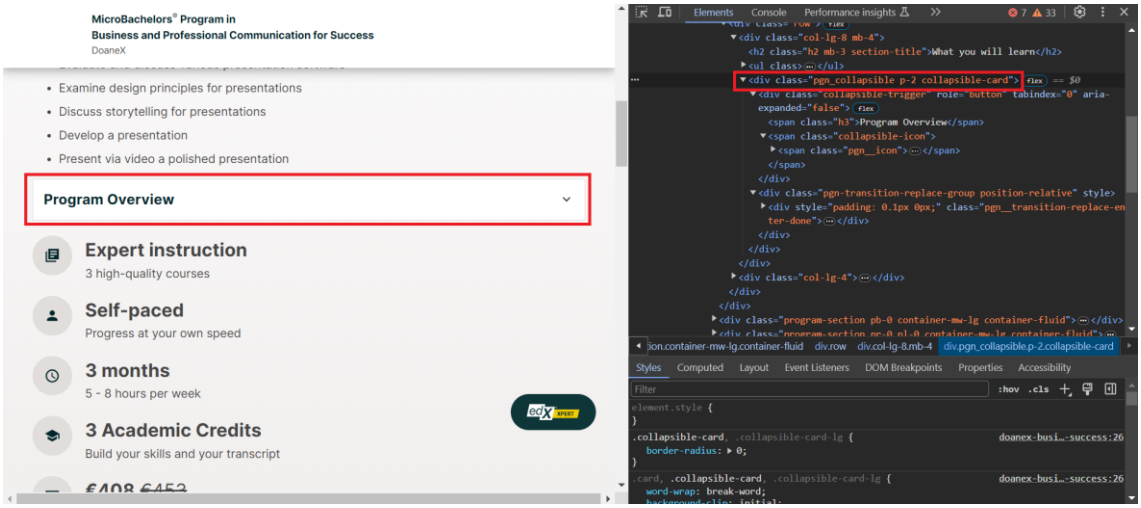

Ilustración 3: Ejemplo estructura de "Descripción" sin presionar el botón para expandir en https://www.edx.org/bachelors/microbachelors/doanex-business-and-professional-communication-for-success.

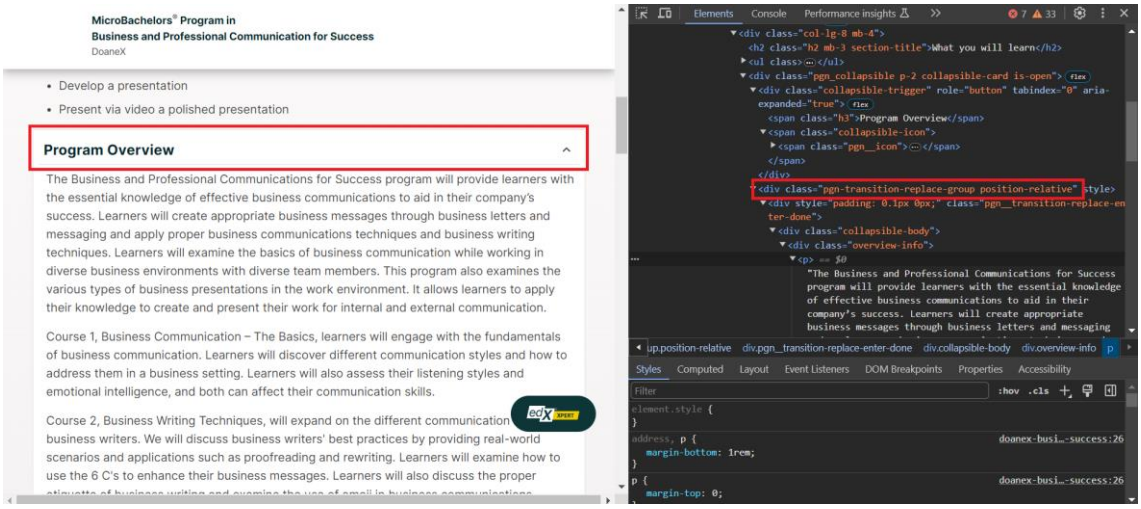

Ilustración 4: Ejemplo estructura de "Descripción" habiendo presionado el botón para expandir en https://www.edx.org/bachelors/microbachelors/doanex-business-and-professional-communication-for-success.

1.3.2.3 Udemy

Udemy es una plataforma de aprendizaje con 62 millones de estudiantes que ofrece 183000 cursos en 75 idiomas diferentes.

Sus cursos están divididos por categorías como diseño, desarrollo, marketing, negocios, etc.

Sus cursos más populares son de Python, Excel y Desarrollo Web.

Para *Udemy*, debido a la herramienta utilizada y sus limitaciones, el sistema de *scraping* solo trata la categoría "*courses*".

La estructura de datos en *Udemy* es la siguiente:

- Nombre del curso.
- Idioma.
- Resultados de aprendizaje en forma de lista.
- Contenidos del curso en un conjunto de elementos de clase "*accordion-panel*".
- Descripción en un elemento "*tabindex = "-1"* " , el cual está oculto hasta que se presiona el botón "Ver más". En ese momento pasa a ser "*tabindex = "0"* ".

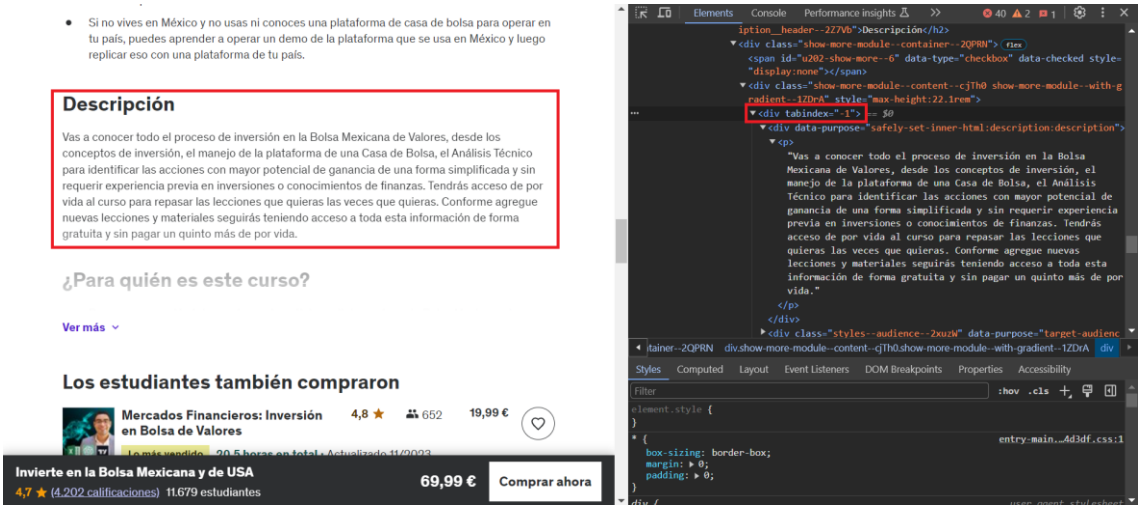

Ilustración 5: Ejemplo estructura de "Descripción" sin presionar el botón "Ver más" en https://www.udemy.com/course/invierte-en-la-bmv/.

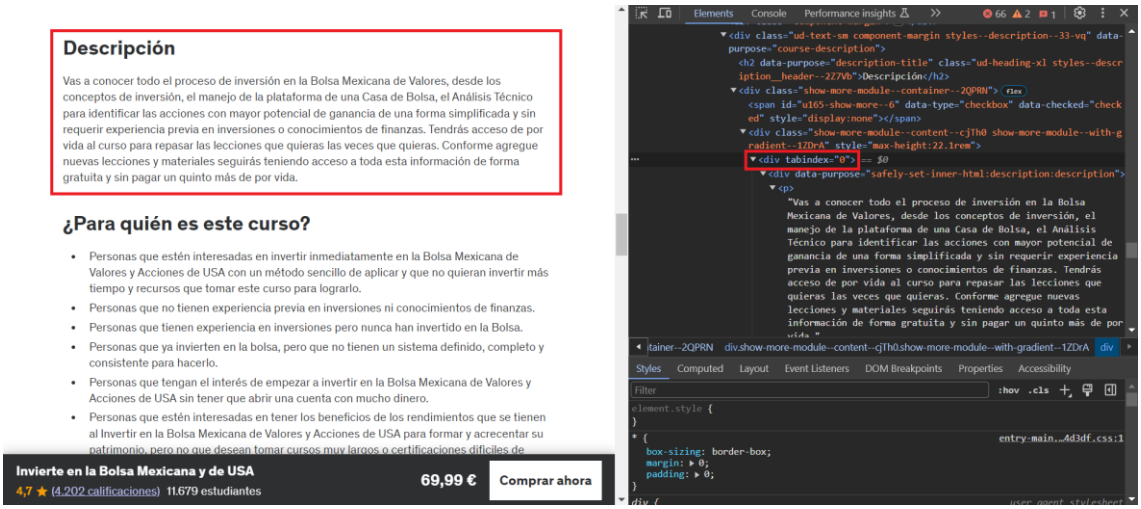

Ilustración 6: Ejemplo estructura de "Descripción" habiendo presionado el botón "Ver más" en https://www.udemy.com/course/invierte-en-la-bmv/.

1.3.3 Publicación de datos

Para la creación de páginas web existen distintas opciones para el *frontend* y para el *backend*. En cuanto al *frontend* se consideran los lenguajes HTML y CSS, React, Angular o Vue.js, así como otras herramientas enfocadas a la creación de páginas web sin código (*CMS*).

React[14] es una librería diseñada para la creación de interfaces web con gran flexibilidad en cuanto a programación. Su código está formado por componentes que son funciones de JavaScript escritos en una sintaxis llamada "JSX" que permite la creación de programas tanto *frontend* como *full-stack*, aunque para este propósito es recomendable usar otro *framework* como Next.js[15] o Remix[16]. Además, ofrece la posibilidad de diseñar aplicaciones web o nativas (específicas a Android o iOS).

HTML[17] (*HyperText Markup Language*) es un lenguaje estándar común a todos los navegadores que establece la estructura de las páginas web. Sirve para

asignar etiquetas únicas a elementos concretos y, junto con otros lenguajes como CSS y Javascript, permite personalizar tanto el estilo como la funcionalidad. Una estructura clara HTML no solo proporciona una mejor experiencia al usuario, sino que permite a los buscadores analizar la página y, de esta manera, ganar visibilidad.

Angular[18] es una plataforma de diseño de aplicaciones basada en TypeScript. Como tal, requiere conocimientos básicos de TypeScript así como de JavaScript y *Command Line Interface* (*CLI*). Sus principales características son su escalabilidad, su integración de librerías y su conjunto de herramientas de desarrollador para ayudar con el proceso de creación de aplicaciones.

Vue.js[19] es un *framework* basado en JavaScript que funciona junto con HTML, CSS y JavaScript para crear una interfaz de usuario de forma flexible y adaptable. Está basado en una estructura *SFC* (*Single File Component*), que encapsula toda la lógica de los componentes (JavaScript), los *templates* (HTML) y los estilos (CSS) en un único archivo.

Aunque son opciones válidas para este trabajo, la falta de familiaridad con estos lenguajes y el límite de tiempo establecido hace inviable el uso de React, Angular o Vue.js.

Una vez establecido HTML como la mejor opción a la hora de diseñar una página web mediante código, se compara con los llamados *CMS* (*Content Management System*)[20]. Estos permiten crear páginas web sin necesidad de escribir código y cuentan con funcionalidades como la creación y publicación de páginas web, edición de textos, instalación de *plugins* para mayor número de funciones, etc., todo esto mediante una interfaz de usuario intuitiva. El *CMS* más popular es *WordPress*[21], con el 43,2% de páginas web a nivel mundial creadas con esta herramienta[22].

Siendo esta otra opción válida, se decidió no seguir adelante con ella ya que en general los *CMS* son de pago, y aunque cuentan con planes gratuitos, la flexibilidad del código HTML lo hace preferible para este proyecto.

Dada su sencillez, accesibilidad y familiaridad en el momento del desarrollo, se decide elaborar los *scripts* del *frontend* en HTML y CSS.

Para el *backend* se ha decidido utilizar Python. Las dos herramientas más populares según "*Python Developers Survey 2022 Results*" publicados por JetBrains[23] son Django y Flask.

Django[24] es un *framework* gratis y *open source* diseñado para la creación de aplicaciones web complejas de forma rápida y escalable. Incluye herramientas que se encargan automáticamente de tareas como autentificación de usuarios, administración de contenido, *sitemaps* y *RSS feeds* (encargada de mandar actualizaciones de nuevo contenido a los usuarios). Incluye además una base de datos relacional y la posibilidad de modificar su estructura con código Python.

Flask[25] ofrece una experiencia sencilla de creación de páginas web, solo siendo necesaria la inicialización del objeto Flask y la definición de las URLs. Ofrece también la posibilidad de cargar *templates* HTML para cada URL y, en general, la flexibilidad de añadir la funcionalidad necesaria para cada contexto.

Dado que para este proyecto no es necesaria una página web compleja ni con funcionalidades como *login* o interfaces de administrador se decide utilizar Flask, siendo la opción mejor más adecuada para las necesidades de este trabajo.

En cuanto a la visualización de datos, se toma inspiración de la página web existente[26] cuyo objetivo es también proporcionar un único punto de acceso a cursos formativos, pero de universidades. En esta página el contenido está presentado mediante un diagrama con la taxonomía de Bloom de cada curso, ofreciendo una herramienta intuitiva y eficaz para facilitar la comprensión de la información.

La taxonomía Bloom[27][28] es un marco educativo que clasifica los objetivos educativos en diferentes categorías que van desde los niveles más básicos de pensamiento hasta los más complejos:

- Recordar: retener conceptos sin necesariamente asimilarlos o comprender su significado. Implica la capacidad de memorización.
- Entender: comprender la información presentada, siendo capaz de explicar conceptos y utilizar ejemplos.
- Aplicar: utilizar lo aprendido en situaciones nuevas, aplicando conceptos teóricos a situaciones prácticas.
- Analizar: descomponer la información en partes, identificando las causas para identificar patrones y comprender las partes que conforman el todo.
- Evaluar: ser capaz de emitir un juicio con respecto a la información, ser capaz de presentar y defender las opiniones.
- Crear: utilizar lo aprendido para crear algo nuevo o proponer soluciones alternativas a un problema existente.

En la página web se muestra a través un diagrama de barras a qué jerarquías está más enfocado el curso. Mediante un análisis de verbos en la descripción de los cursos se determina qué porcentaje hay de cada una y así proporcionar al usuario una información más clara y fácil de entender. El diagrama consta de seis barras, cada una asociada a una jerarquía, cuya altura corresponde al porcentaje de verbos encontrados de cada categoría, permitiendo saber con un único golpe de vista a qué está orientado el curso y si es o no lo que se está buscando.

1.4 Contribuciones

1.4.1 Scraping

Este proyecto se compone de 3 scripts de *scraping*, uno existente[29] y otros dos realizados desde cero[30]. En el primero (correspondiente a *Udemy*) se ha cambiado la forma de almacenar los resultados, pasando de un sistema de guardado con números (1, 2, etc.) según el orden de ejecución, lo cual podía dar lugar a duplicados con distinto nombre, a ficheros de extensión JSON nombrados con un hash creado a partir de su URL, consiguiendo de esta forma eliminar los duplicados. También se incluye en los tres un método para tratar los caracteres especiales, lo cual es un problema en los scripts originales.

En los scripts de *Coursera* y *edx* se realiza el *scraping* desde cero basándose en el módulo Scrapy. De esta forma se consiguen resultados similares, pero con una gran reducción de tiempo. De esta forma, los ficheros de cursos individuales no se guardan en carpetas separadas para cada tipo, sino todas en la carpeta correspondiente a su plataforma, por lo que no es necesario guardar el tipo de curso. Además, se filtran para que solo se guarden aquellos cursos en español o inglés.

Cabe destacar que no se ha podido actualizar el código para el portal *Udemy*, ya que usando Scrapy la solicitud GET a la página devuelve el código de error 403 (*forbidden*). Esto se debe a que en la página

"https://www.udemy.com/robots.txt", que especifica qué *user agents* tienen permitido acceder a qué páginas dentro del dominio, se prohíbe el acceso al de Scrapy, haciendo imposible el proceso de *scraping*.

Una vez completados, se actualiza el *cron* de la máquina virtual para que se ejecuten los *scripts* automáticamente una vez por semana.

1.4.2 Publicación de datos en la página web

A la hora de decidir qué datos mostrar al realizar las consultas, se tomó la decisión en base a los scripts "Web demo del proyecto ai4labour"[31] de mostrar:

- Un enlace con el que poder acceder al curso.
- El nombre del curso y la plataforma en la que se encuentra.
- Una descripción acortada con un botón "Ver más" que abre un cuadro de texto en el que está detallada toda la descripción del curso.
- La taxonomía Bloom correspondiente.

Para facilitar la subida de los datos a la página web, el script "combine_json.py" recorre las carpetas de cada curso y recopila todos los archivos individuales en un único fichero llamado "all.json", ubicado en la carpeta inmediatamente superior. El fichero "combine_json.py" también está incluido en el *cron* para ejecutarse una vez por semana con los nuevos datos recolectados.

En cuanto a la página web, se crea una página web desde cero[6] basada en Python empleando la librería Flask, la librería json para interpretar los datos y una serie de ficheros de extensión HTML para su visualización.

La información de los cursos se presenta en forma de un diagrama de barras de su taxonomía Bloom para, de esta forma, poder entender rápidamente a qué aspecto de la taxonomía está más enfocado el curso. Cada barra está asociada a un nivel de la jerarquía previamente explicada y se obtiene mediante un análisis de palabras clave en las descripciones de cada curso.

2 Desarrollo

En este capítulo se detallan los pasos y decisiones tomadas en el proceso de cumplimiento de los resultados especificados anteriormente.

2.1 Scraping

Todos los scripts de scraping tienen una serie de funciones comunes, utilizadas para facilitar tanto la recopilación de datos como su visualización tras la ejecución.

En primer lugar, para tratar los caracteres especiales se ha recurrido a la siguiente función:

```
Función convert_to_utf8(var):
    return unicodedata.normalize('NFKD', var).encode('ascii', 'ignore').decode()
Fin Función
```

También cuentan con una función para generar un código hash basado en la URL de cada curso para poder ser guardada la información de cada curso de forma eficiente:

```
Función generate_hash(url):
   hash_obj = hashlib.sha256(url.encode())
   return hash_obj.hexdigest()
Fin Función
```

2.1.1 Coursera

En la clase *CourseraScrapper* se definen las URLs iniciales y los algoritmos necesarios para obtener los datos deseados de cada página. Tiene la siguiente estructura:

```
class CourseraScrapper(scrapy.Spider):
      name = ‘nombre del scrapper’
      sitemap_urls = [
            ‘lista de URLs a scrappear’
      ]

      Función parse_sitemap
            Para cada url ∈ sitemap_urls  Hacer
                  Función parse
            Fin Hacer
      Fin Función

      Función parse
            language = response.css(etiqueta_html)
```

```
        Si language = Spanish o language = English Entonces
              description = response.css(etiqueta_html)
              Si description != None Entonces
                    name = response.css(etiqueta_html)
                    url = response.url
                    topics = response.css(etiqueta_html)
                    skills = response.css(etiqueta_html)
                    outcomes = response.css(etiqueta_html)
                    data = conjunto_datos
                    Escribir 'Processing url'
                    Escribir data en hash_url.json
              Fin Si
        Si no Entonces
              Escribir 'Exception: unsupported language.'
        Fin Si
  Fin Función
```

De esta forma se consiguen analizar cinco *sitemaps* distintos: "https://www.coursera.org/sitemap~www~courses.xml", "https://www.coursera.org/sitemap~www~professional-certificate.xml", "https://www.coursera.org/sitemap~www~onDemandSpecializations.xml", "https://www.coursera.org/sitemap~www~mastertrack.xml" y "https://www.coursera.org/sitemap~www~guided-projects.xml".

Cada uno de estos enlaces contiene las páginas de todos los tipos de curso ofrecidos en *Coursera*: "*course*", "*certificate*", "*specialization*", "*mastertrack*" y "*project*".

A la hora de analizar las páginas y dar con la etiqueta HTML adecuada para cada elemento, se ha empleado el comando `scrapy shell 'url'`. Ejecutando este comando con la URL de cualquier curso de *Coursera*, ya que la estructura HTML es idéntica para todos los cursos del portal, se van haciendo pruebas junto con la herramienta "Inspeccionar" de *Google* hasta que se encuentra la información requerida.

Empleando dicha herramienta en la página del curso deseado se analizan las etiquetas de cada elemento y se va comprobando el resultado como se ve en la imágen. Para este ejemplo se buscan los *outcomes* en la página: https://www.coursera.org/learn/advanced-android-development:

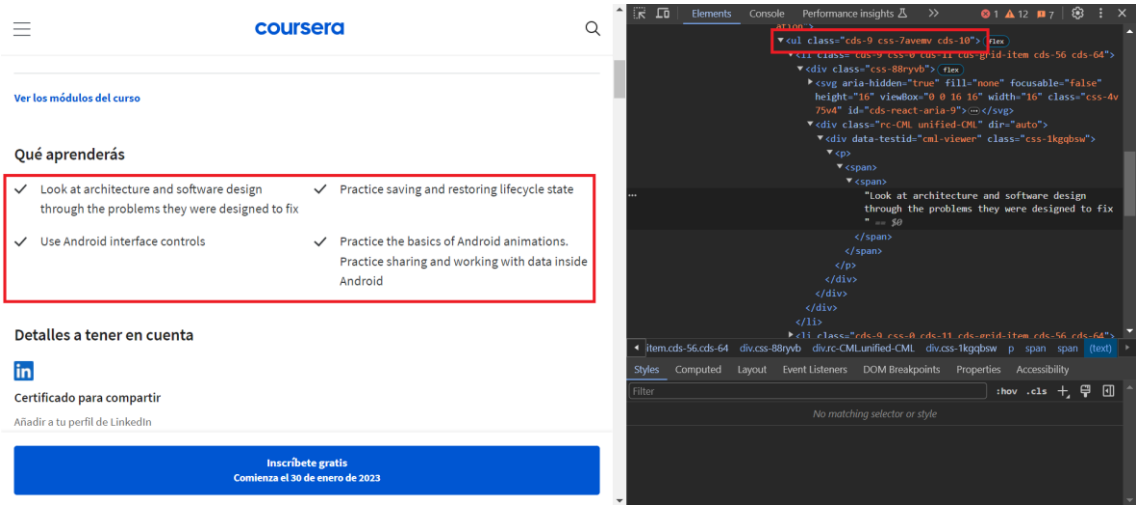

Ilustración 7: Análisis de estructura HTML de Coursera con la herramienta Inspeccionar.

En la terminal se ejecuta el comando `scrapy shell 'url'` y se realiza la prueba con el elemento y la clase señalados en la imagen anterior:

```
2024-01-10 20:59:43 [asyncio] DEBUG: Using proactor: IocpProactor
[s] Available Scrapy objects:
[s]   scrapy     scrapy module (contains scrapy.Request, scrapy.Selector, etc)
[s]   crawler    <scrapy.crawler.Crawler object at 0x000001F3C1F25070>
[s]   item       {}
[s]   request    <GET https://www.coursera.org/learn/advanced-android-development>
[s]   response   <200 https://www.coursera.org/learn/advanced-android-development>
[s]   settings   <scrapy.settings.Settings object at 0x000001F3C1F25520>
[s]   spider     <DefaultSpider 'default' at 0x1f3c243ae50>
[s] Useful shortcuts:
[s]   fetch(url[, redirect=True]) Fetch URL and update local objects (by default, redirects are followed)
[s]   fetch(req)                  Fetch a scrapy.Request and update local objects
[s]   shelp()           Shell help (print this help)
[s]   view(response)    View response in a browser
2024-01-10 20:59:43 [asyncio] DEBUG: Using proactor: IocpProactor
In [1]: response.css('ul.cds-9.css-7avemv.cds-10 ::text').getall()
Out[1]:
['Look at architecture and software design through the problems they were designed to fix ',
 'Practice saving and restoring lifecycle state',
 'Use Android interface controls',
 'Practice the basics of Android animations. Practice sharing and working with data inside Android']
```

Ilustración 8: Proceso de scraping en la terminal mediante "scrapy shell 'url'".

Mediante este proceso de prueba y error se consiguen todos los datos necesitados.

Antes de guardar los resultados, se aplica a cada uno la función `convert_to_utf8(var)` para asegurar el procesado sin caracteres especiales que puedan dar lugar a errores en la visualización.

```
name_utf8 = convert_to_utf8(name)
```

En el caso de que los resultados obtenidos estén en forma de lista, como *outcomes* en el ejemplo, se utiliza el algoritmo:

```
outcomes_utf8 = []
Para cada outcome ∈ outcomes Hacer
        outcomes_utf8.append(convert_to_utf8(outcome))
```

```
Fin Hacer
```

Para cada página se obtiene un diccionario `data` con la siguiente información:

```
data = {
      "url":url,
      "name":name_utf8,
      "topics":topics_utf8,
      "skills":skills_utf8,
      "description":description_utf8,
      "language":language,
      "outcomes":outcomes_utf8
}
```

Una vez terminado el proceso de *scraping* de la página, se escribe el resultado en un fichero de nombre el hash generado por su URL de la siguiente forma:

```
with open(f'coursera_data/{generate_hash(url)}.json', 'w') as f:
      json.dump(data,f,indent=4)
```

2.1.2 edX

La clase *EdxScrapper* tiene una estructura diferente a *CoruseraScrapper* ya que utiliza el módulo "scrapy.spiders". En esta clase se definen `name` y `start_urls` igual que en el anterior, pero también se definen una serie de `rules`, que establecen a qué función llamar en el caso de encontrar un enlace que contenga una palabra clave:

```
Class EdxScrapper(CrawlSpider):
    name = 'edx_spider'
    start_urls = ['https://www.edx.org/sitemap']


    rules = (
        Rule(LinkExtractor(allow='doctorate'), callback='parse_items'),
        Rule(LinkExtractor(allow='bachelors'), callback='parse_items'),
        Rule(LinkExtractor(allow='masters'), callback='parse_items'),
        Rule(LinkExtractor(allow='certificates'),
callback='parse_items'),
        Rule(LinkExtractor(allow='xseries'), callback='parse_items')
    )
```

De esta forma, en caso de encontrar en la lista de URLs alguna que contenga la URL inicial se llama a la función `parse_items()`:

```
Función parse_items (self, response)
      language = response.css(etiqueta_html)
            Si language = Spanish o language = English Entonces
                  description = response.css(etiqueta_html)
                  Si description != None Entonces
                        name = response.css(etiqueta_html)
                        url = response.url
                        topics = response.css(etiqueta_html)
                        skills = response.css(etiqueta_html)
                        outcomes = response.css(etiqueta_html)
                        data = conjunto_datos
                        Escribir 'Processing url'
                        Escribir data en hash_url.json
                  Fin Si
            Si no Entonces
                  Escribir 'Exception: unsupported language.'
            Fin Si
Fin Función
```

Como se puede observar, la función `parse_items()` tiene la misma estructura que la función `parse()` en la clase *CourseraScrapper*. La diferencia está en las distintas etiquetas HTML, obtenidas mediante el mismo proceso de prueba y error usando `scrapy shell 'url'` esta vez con la URL de prueba: https://www.edx.org/bachelors/microbachelors/doanex-business-and-professional-communication-for-success.

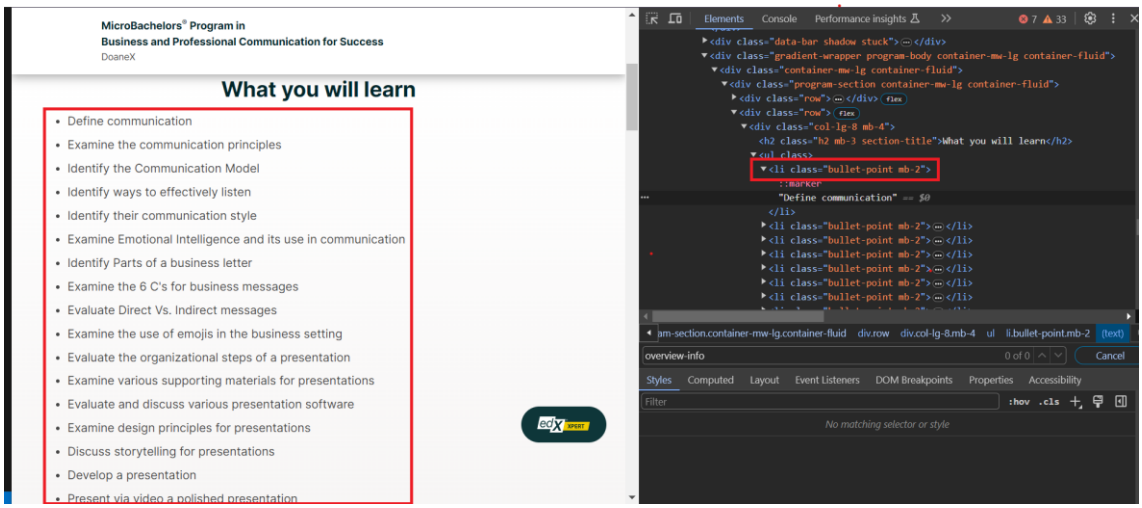

Ilustración 9: Análisis de la estructura HTML de edX con la herramienta Inspeccionar.

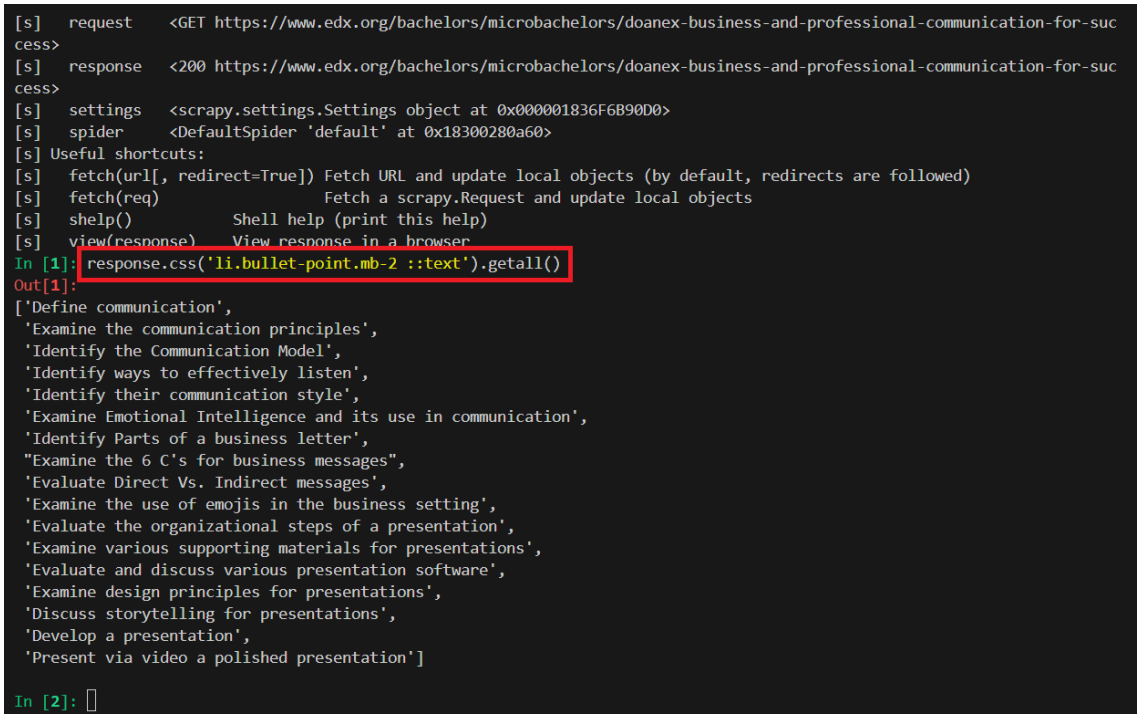

Ilustración 10: Proceso de scraping en la terminal mediante "scrapy shell 'url'".

Una vez obtenidos, se realiza el mismo tratamiento con las funciones `convert_to_utf8` y `generate_hash(url)` para guardar un fichero de extensión JSON de cada curso en su carpeta correspondiente:

```
with open(f'edx_data/{generate_hash(url)}.json', 'w') as f:
        json.dump(data,f,indent=4)
```

2.1.3 Udemy

Como ya se ha explicado en apartados anteriores, este código está basado en los scripts existentes con modificaciones para mejorar el guardado de archivos y la presentación de información. Esto se consigue con las funciones `convert_to_utf8(var)` y `generate_hash(url)`.

En primer lugar, se establece conexión con la URL deseada mediante el módulo "requests" y se crea un objeto "BeautifulSoup" con el contenido HTML de la página. El *sitemap* que se analiza es una lista de las páginas con URLs de cursos en Udemy:

```
url = "https://www.udemy.com/sitemap.xml"

response = requests.get(url)

xml = BeautifulSoup(response.text, 'lxml-xml')
```

Mediante la herramienta "Inspeccionar" del navegador se analiza la estructura HTML del *sitemap* y se concluye que los enlaces a los cursos tienen la etiqueta "`loc`". Una vez obtenidas las URLs se guardan aquellas que contengan la palabra clave "`courses`" y se procede al scraping de datos.

```
urls = xml.find_all("loc")
```

```
urls = [u.text for u in urls if “courses” in u.text]
```

Ilustración 11: Análisis de la estructura HTML del sitemap con la herramienta “Inspeccionar”.

Cada enlace contiene a su vez una lista de URLs de cursos, también identificados con la etiqueta HTML “loc” que se analizan individualmente.

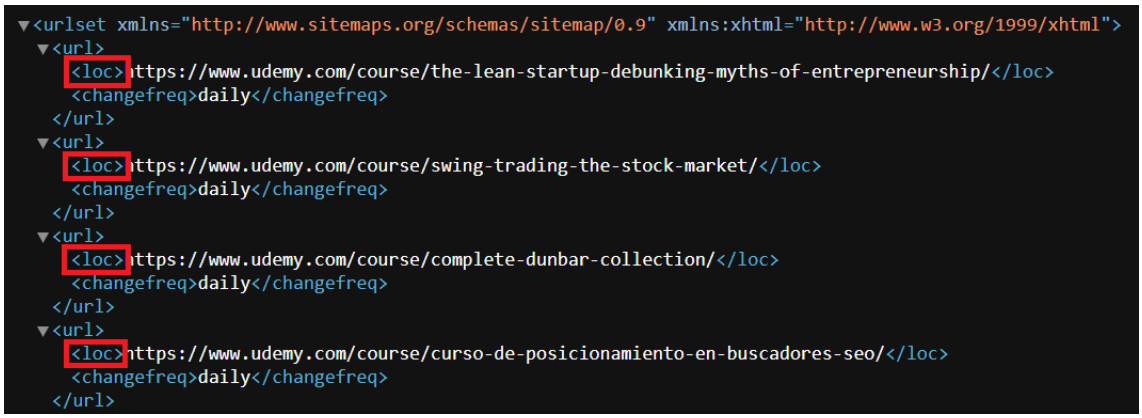

Ilustración 12: Análisis de la estructura HTML del sitemap de cursos con la herramienta "Inspeccionar".

```
Para cada url_tag ∈ urls Hacer
      response2 = requests.get(url_tag2)
      xml2 = BeautifulSoup(response2.text, 'lxml-xml')
      courses_urls = xml2.find_all("loc")

      Para cada url_tag2 ∈ courses_urls Hacer
```

```
            url = url_tag2.text
            url_response = requests.get(url)
            url_soup = BeautifulSoup(url_response.text, 'html.parser')

            name = convert_to_utf8(url_soup.title.string)
            skills = url_soup.find_all("etiqueta_html")
            skills_utf8 = convert_to_utf8(skills)
            description = url_soup.find_all("etiqueta_html")
            description_utf8 = convert_to_utf8(description)
            topics = url_soup.find_all("etiqueta_html")
            topics_utf8 = convert_to_utf8(topics)
            language = url_soup.find_all("etiqueta_html")
            language_utf8 = convert_to_utf8(language)
            outcomes = url_soup.find_all("etiqueta_html")
            outcomes_utf8 = convert_to_utf8(outcomes)

            data = conjunto_datos

            with open(f'udemy_data/{generate_hash(url)}.json', 'w') as
      f:
                  json.dump(data,f,indent=4)

      Fin Hacer
Fin Hacer
```

Los intentos de mejorar este código implementando *Scrapy* resultaron en vano, ya que al intentar establecer conexión se devuelve un error 403 (*forbidden*):

```
[s]   request    <GET https://www.udemy.com/sitemap/courses.xml>
[s]   response   <403 https://www.udemy.com/sitemap/courses.xml>
[s]   settings   <scrapy.settings.Settings object at 0x0000028365A4B070>
[s]   spider     <DefaultSpider 'default' at 0x28376630940>
[s] Useful shortcuts:
[s]   fetch(url[, redirect=True]) Fetch URL and update local objects (by default, redirects are followed)
[s]   fetch(req)                  Fetch a scrapy.Request and update local objects
[s]   shelp()           Shell help (print this help)
[s]   view(response)    View response in a browser
In [1]: response
Out[1]: <403 https://www.udemy.com/sitemap/courses.xml>

In [2]:
```

Ilustración 13: Intento de scraping con Scrapy.

2.2 Publicación de datos

2.2.1 Resultados de scraping

Para la recuperación de datos a la hora de realizar búsquedas en la página web se utiliza un fichero “combine_json.py” que combina todos los ficheros individuales de las tres plataformas en un único fichero “all.json” y en otro “all.csv”.

En primer lugar se definen las rutas con los ficheros de cursos individuales y la ruta de los dos ficheros de salida.

```
res = []
Para cada folder ∈ data_folders Hacer
      Para cada files ∈ folder Hacer
            Para cada file ∈ files Hacer
                  data = json.load(file)
                  res.append(data)
            Fin Hacer
      Fin hacer
Fin Hacer
```

El siguiente paso es escribir la información en los ficheros JSON y CSV:

```
with open(all_json_route, 'w') as outfile:
        json.dump(res, outfile, indent=4)

df = pd.DataFrame(res)
df.to_csv(all_csv_route, index=False)
```

La información para la página web se obtiene del fichero “all.json”, que contiene todos los cursos y sus datos presentados de forma ordenada y fácilmente recuperable. El archivo “all.csv” no es usado por la página web, pero existe para facilitar la depuración del código y comprobar que los resultados son los esperados.

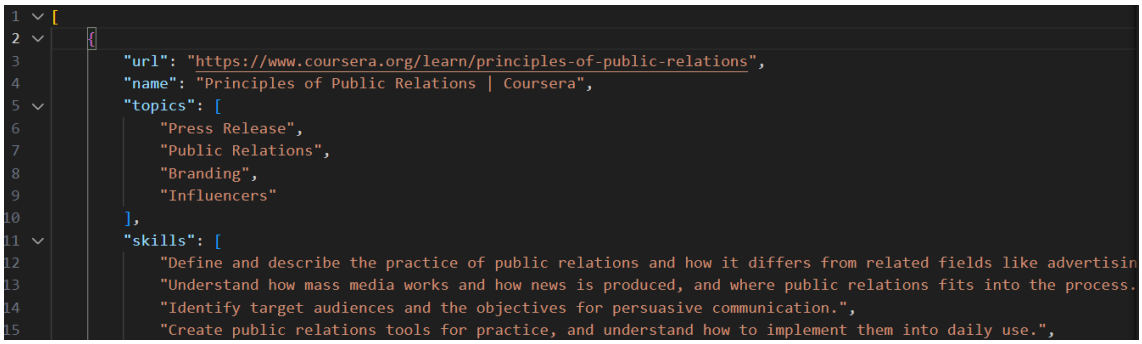

Ilustración 14: Estructura del fichero "all.json".

	A	B
1	url	name
2	https://www.coursera.org/learn/principles-of-public-relations	Principles of Public Relations | Coursera
3	https://www.coursera.org/learn/optimize-machine-learning-model-performance	Optimizing Machine Learning Performance | Coursera
4	https://www.coursera.org/learn/shape-and-property-control-of-metals-1-and-2	Shape and Property Control of Metals I & II | Coursera
5	https://www.coursera.org/learn/physiology	Introductory Human Physiology | Coursera
6	https://www.coursera.org/learn/build-and-operate-machine-learning-solutions-with-azure	Build and Operate Machine Learning Solutions with Azure | Coursera
7	https://www.coursera.org/projects/create-virtual-private-cloud-vpc-aws	Create a Virtual Private Cloud (VPC) Using AWS
8	https://www.coursera.org/learn/stepping-up-leading-others	Stepping Up: Leading Others | Coursera

Ilustración 15: Estructura del fichero "all.csv".

2.2.2 Lógica de la página web

Toda la lógica detrás de los resultados mostrados en la página web está definida en el fichero "app.py". El primer paso es crear la página mediante *Flask*, definiendo las rutas a las que se accede a la hora de realizar una búsqueda:

```
app = Flask(__name__)


@app.route('/', methods=['GET','POST'])

...


@app.route('/search', methods=['GET','POST'])

...


if __name__ == '__main__':
    app.run(debug=True)
```

Este código establece que al cargar por primera vez la página la URL es "url/" y una vez se realice una búsqueda se salta a "url/search".

Cabe destacar que al inicializar el script se asocia a la página web la dirección local del puerto 5000 (http://127.0.0.1:5000), pero esta solo está disponible si el fichero se ejecuta en la máquina local. Para solucionar esto, se ha añadido la URL generada por la librería Flask a un servidor apache2 existente en la máquina virtual asociada al proyecto AI4LABOUR. De esta forma se consigue una URL permanente siempre que la máquina virtual esté disponible.

Lo primero que se hace es cargar el *template* "index.html" (detallado en el siguiente apartado) mediante la función declarada en la primera ruta:

```
@app.route('/', methods=['GET','POST'])

Función index

      return render_template('index.html')
```

```
Fin Función
```

Después se define la función para buscar los cursos según el parámetro de búsqueda. Esta solo busca el número de cursos equivalente al máximo de resultados que se quieren mostrar en pantalla, ya que al tratarse de muchos datos si analizase todos los cursos provocaría problemas de rendimiento:

```
Función get_course (query,max)
      res = []
      with open('all.json','r') as f:
        data = json.load(f)

      i = 1
      Para course ∈ data Hacer
            Si query in course Entonces
                  res.append(course)
                  i += 1
            Fin Si
            Si i == max Hacer
                  break
            Fin Hacer
      Fin Hacer

      Escribir f'Courses shown: {len(res)}'
Fin Función
```

Una vez encontrados los cursos es necesario analizar la descripción de cada uno para definir su taxonomía Bloom según una lista de verbos predefinida (Anexo 1) que refleja cada jerarquía:

```
Función count_verbs (course_description, verbs_list)
      sum = 0
      Para verb in verb_list Hacer
            sum += course_description.count(verb)
      Fin Hacer
      return sum
Fin Función

Función count_bloom_verbs (course_description)
```

```
      total_verbs = 0
      bloom_count = []
      res = []
      Para verb_list ∈ bloom_verbs_list Hacer
            count = count_verbs(course_description,verb_list)
            total_verbs += count
            bloom_count.append(count)
      Fin Hacer
      Para num ∈ bloom_count Hacer
            Intentar
                  res.append('{:.1f}'.format(num/total_verbs*100))
            Excepción
                  return [0,0,0,0,0,0]
            Fin Intentar
      Fin Hacer
      return res
Fin Función
```

Al presionar el botón de búsqueda se ejecuta la función declarada en la ruta "/search". Esta es la encargada de:

- Definir cuántos resultados como máximo aparecen en la página de resultados.
- Obtener los cursos que coincidan con la palabra clave introducida.
- Ejecutar el análisis necesario para realizar la taxonomía Bloom.
- Invocar el siguiente *template* según el resultado que se obtenga ("results.html" o "no_results.html")

```
max_results = 20


@app.route('/search', methods=['GET','POST'])
Función search
      search_query = request.form['search']
      courses = get_course(search_query)


      Si courses = [] Entonces
            return render_template('no_results.html')
      Fin Si


      bloom_arr = []
      Para course ∈ courses Hacer
```

```
            bloom_count = count_bloom_verbs(course['description'])
            bloom_arr.append(bloom_count)
      Fin Hacer
      return      render_template('results.html',      courses=courses,
bloom_nums=bloom_arr, limit=min(max_results, len(courses)))


Fin Función
```

Con este código se consiguen obtener los cursos que coinciden con la solicitud de búsqueda así como la información asociada que se ha de mostrar en pantalla.

2.2.3 Diseño de la página web

Como se ha mencionado previamente, para el diseño de la página web se emplean tres ficheros HTML ubicados en la carpeta *templates*. Cada uno de ellos cumple una función concreta, pero todos tienen elementos en común.

En el caso de "index.html", el título de la página, la barra superior con la descripción de su funcionalidad y la imagen del proyecto *AI4Labour* son comunes a todos los demás ficheros:

```
<head>
<title>TFG Miguel Fernández Díaz</title>
<style>
      body {
            font-family: Arial, sans-serif;
            margin: 0;
            padding: 0;
      }
      .navbar {
            background-color: #333;
            color: #fff;
            padding: 15px 20px;
            text-align: center;
      }

      .navbar img {
            width: 100px;
```

```
            height: auto;
      }
</style>
</head>

<body>
<div class="navbar">
        <img src="https://ai4labour.com/wp-
content/uploads/2021/11/AI4labour-Logo_page-0001-scaled-300x187.jpg">
        <h1>Sistema de Recomendación de Cursos</h1>
</div>
</body>
```

Ilustración 16: navbar y título.

También está definida en todos los scripts la barra de búsqueda, el botón "Buscar" y su funcionalidad, así como su aspecto mediante código CSS:

```
<style>
      .search-container {
            display: flex;
            justify-content: center;
            margin-top: 20px;
      }

      .search-form {
            display: flex;
            align-items: center;
            background-color: #fff;
            border-radius: 5px;
            box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);
```

```
            padding: 8px;
      }

      input[type="text"] {
            padding: 8px;
            border: 1px solid #ccc;
            border-radius: 3px;
            margin-right: 5px;
            width: 300px;
      }

      input[type="submit"] {
            padding: 8px 20px;
            background-color: #007bff;
            color: #fff;
            border: none;
            border-radius: 3px;
            cursor: pointer;
      }

      input[type="submit"]:hover {
            background-color: #0056b3;
      }
</style>

<div class="search-container">
        <form class="search-form"
action="https://ai4labour.linkeddata.es/miguel/search" method="post">
            <input type="text" id="search" name="search"
placeholder="Introduce una habilidad (biología, java, etc.)">
            <input type="submit" value="Buscar">
```

```
        </form>
</div>
```

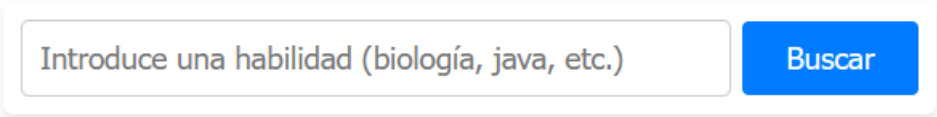

Ilustración 17: Barra de búsqueda.

Definiendo la *action* de la barra de búsqueda con la URL asignada tras la creación del servidor apache2 en la máquina virtual se consigue que al realizar una petición se salte a esa URL.

En ese momento se carga el *template* "results.html", cuya estructura es idéntica a "index.html" salvo que añadiendo una tabla con los resultados de la búsqueda.

```
<style>
      .result-container {
            display: flex;
            justify-content: space-evenly;
            margin-top: 20px;
      }
</style>


<table class="result-container" style="border-collapse: collapse; font-size: 14px;">
      <tbody id="courses-container">
            <tr>
                <th>Enlace</th>
                <th>Nombre y Plataforma</th>
                <th>Descripción</th>
                <th>
                    <a href="https://www3.gobiernodecanarias.org/medusa/edublog/cprofestenerifesur/2015/12/03/la-taxonomia-de-bloom-una-herramienta-imprescindible-para-ensenar-y-aprender/">
                        Taxonomía Bloom
                    </a>
                </th>
            </tr>
```

```
            ...
      </tbody>
</table>
```

En esta primera parte de código se crea la cabecera de la tabla, estableciendo cuatro columnas: Enlace, Nombre y Plataforma, Descripción y Taxonomía Bloom. La última contiene además un enlace a la explicación de la taxonomía Bloom y su importancia en el ámbito del aprendizaje.

Cada fila de la tabla de resultados se define de la siguiente forma (código de <style> para cada fila en Anexo 2):

```
<table>

...
      {% for i in range(limit) %}
      <tr>
            <td class="result-item">
                  <a href="{{ courses[i]['url'] }}">
                        <img
                  src="https://media.istockphoto.com/id/1135273220/vec
                  tor/link-icon-hyperlink-chain-symbol-external-link-
                  symbol-vector-icon-download-share-and-
                  load.jpg?s=612x612&w=0&k=20&c=RamInH2oNHWH_H0ESESgVd
                  J4GPXkFuffI-xgCZHj7R8=" alt="Link" width="50"
                  height="50">
                  </a>
            </td>
            <td class="result-item">{{ courses[i]['name'] }}</td>
            <td class="result-item">
                  <p>{{ courses[i]['description']|truncate(200, True,
            '...') }}</p>
                  {% set course_description =
            courses[i]['description'].replace('\n',
            '\\n').replace("'", "\\'").replace('\r', '') %}
                  <button id="ver_mas_button_{{ i }}"
            onclick="open_description('{{ course_description }}')">Ver
            más</button>
            </td>
            <td class="result-item">
                  <table class="graph">
```

```
                        <thead>
                              <tr>
                                    <th scope="col">Item</th>
                                    <th scope="col">Percent</th>
                              </tr>
                        </thead>
                        <tbody>
                              <tr
                        style="height:{{ bloom_nums[i][0] }}%">
                                    <th scope="row">Recordar</th>
                                    <td><span>{{ bloom_nums[i][0] }}%</
                                    span></td>
                              </tr>
                             <tr style="height:{{bloom_nums[i][1]}}%">
                                     <th scope="row">Entender</th>
                                    <td><span>{{bloom_nums[i][1]}}%</sp
                                    an></td>
                              </tr>
                               <tr
style="height:{{bloom_nums[i][2]}}%">
                                    <th scope="row">Aplicar</th>
                                    <td><span>{{bloom_nums[i][2]}}%</sp
                                    an></td>
                              </tr>
                             <tr style="height:{{bloom_nums[i][3]}}%">
                                    <th scope="row">Analizar</th>
                                    <td><span>{{bloom_nums[i][3]}}%</sp
                                    an></td>
                              </tr>
                             <tr style="height:{{bloom_nums[i][4]}}%">
                                    <th scope="row">Evaluar</th>
```

```
                                    <td><span>{{bloom_nums[i][4]}}%</sp
                                    an></td>

                                </tr>

                                <tr style="height:{{bloom_nums[i][5]}}%">

                                    <th scope="row">Crear</th>


                                    <td><span>{{bloom_nums[i][5]}}%</sp
                                    an></td>

                                </tr>

                            </tbody>

                        </table>

                    </td>

                </tr>

                {% endfor %}

</table>
```

En primer lugar, dado que a “results.html” se le invoca con una lista de cursos se crea un bucle para mostrar de forma dinámica cada uno de los cursos, llegando a un máximo de 20. Esto se consigue con la línea de código `{% for i in range(limit) %}`.

Para cada curso de la lista se define el enlace de la primera columna, asignando a la imagen la URL del curso correspondiente mediante `{{ courses[i]['url'] }}`.

El nombre y plataforma se consiguen de la misma forma, usando `{{ courses[i]['name'] }}`.

Para la descripción se decide acortar lo que se muestra en la celda para mayor comodidad, dejando visibles dos líneas y añadiendo un botón “Ver más”. Este abre un cuadro de texto con la descripción completan y está definido mediante dos funciones:

```
<style>

        .description-box {

            display: none;

            position: fixed;

            z-index: 1;

            left: 0;
```

```
            top: 0;
            width: 100%;
            height: 100%;
            overflow: auto;
            background-color: rgba(0,0,0,0.7);
      }
      .description-box-content {
            background-color: #fefefe;
            margin: 15% auto;
            padding: 20px;
            border: 1px solid #333;
            width: 80%;
      }
      .close {
            color: #aaa;
            float: right;
            font-size: 28px;
            font-weight: bold;
      }
      .close:hover, .close:focus {
            color: black;
            text-decoration: none;
            cursor: pointer;
      }
</style>

<script>
    function open_description(description) {
        var descriptionContent = document.getElementById('full-
description');
        descriptionContent.innerText = description;
```

```
        var descriptionBox = document.getElementById('description-
box');
        descriptionBox.style.display = 'block';
    }

    function close_description() {
        var description = document.getElementById('description-box');
        description.style.display = 'none';
    }
</script>
```

Estas funciones son llamadas mediante el parámetro `onclick="open_description('{{ course_description }}')` del botón "Ver más" y `onclick="close_description()` de "description-box".

La taxonomía Bloom consiste en un diagrama[32] basado en los resultados del análisis de verbos de las descripciones de los cursos. Se crean de forma dinámica 6 barras, cada una con un porcentaje asociado al verbo de la jerarquía correspondiente que determina su altura, para que con un simple golpe de vista se pueda tener una idea aproximada de qué trata el curso. Estas barras cuentan con la funcionalidad de que al poner el cursor sobre ellas se ve el porcentaje de verbos relacionados con cada nivel de taxonomía en la propia descripción.

En conjunto, esto es lo que aparece al realizar una búsqueda:

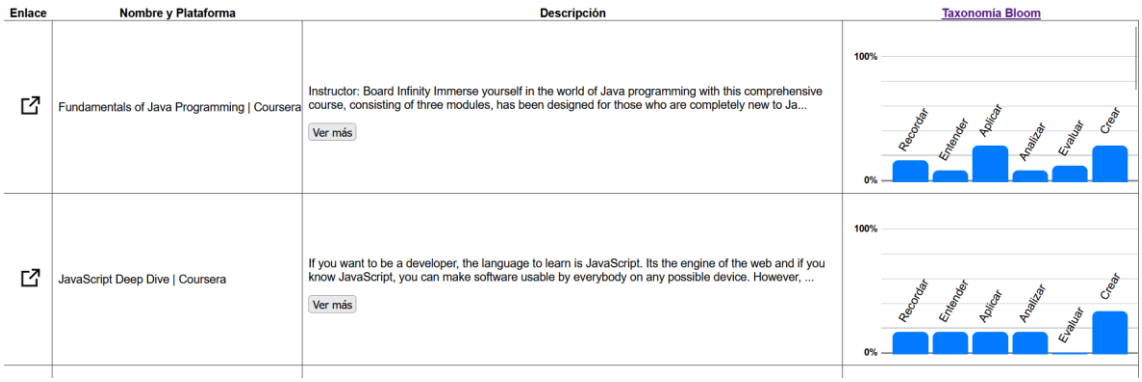

Ilustración 18: Tabla de resultados.

En el caso en que ningún curso coincida con los criterios de búsqueda, el programa invoca el *template* "no_results.html", que muestra debajo de la barra de búsqueda una "message-box" con el texto "No se han encontrado resultados" y una imagen que representa esa idea.

```
<style>
```

```
	.message-box {
		text-align: center;
	}
	.message-box img {
		width: 100px;
		height: auto;
	}
</style>

<div class="message-box">
	<h2>No se han encontrado resultados</h2>
	<img src="https://cdn-icons-
png.flaticon.com/512/6134/6134116.png">
</div>
```

Esto es lo que se muestra en pantalla:

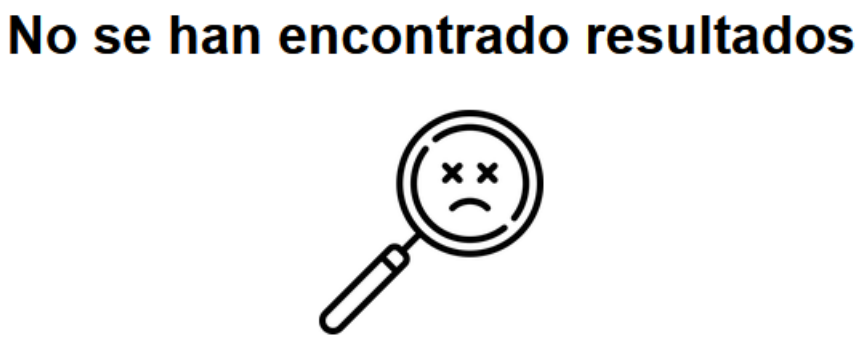

Ilustración 19: Mensaje de no resultados.

3 Resultados y conclusiones

El código de este proyecto ha sido publicado en "Zenodo"[33] conforme a los principios FAIR[34] con *DOI*: 10.5281/zenodo.10479337. De esta forma se garantiza la disponibilidad a largo plazo de este trabajo.

También ha sido publicado en "GitHub" siguiendo las buenas prácticas de publicación de repositorios en "GitHub"[35], contando con ficheros "README.md", "requirements.txt"[36] y "LICENSE" así como mensajes de *commits* descriptivos, claros y concisos que aportan contexto a los cambios realizados.

En el marco de este proyecto, se ha llevado a cabo la creación de unos sistemas de *scraping* y una página web encargada de la visualización de los datos obtenidos. Las páginas de cursos sometidas al proceso de *web scraping* han conformado un único fichero con los detalles considerados más importantes a la hora de analizar los cursos y la información recopilada se ha organizado en un diagrama con la taxonomía de Bloom de cada curso.

Como se estableció anteriormente, el objetivo de este trabajo es la recopilación de la gran cantidad de información disponible en portales diferentes y su presentación en un único punto unificado para su fácil análisis y comprensión. En este sentido, se ha cumplido el objetivo, consiguiendo crear una única página web en la que realizar búsquedas de cursos disponibles en varias plataformas online mediante palabras clave. Sin embargo, hay varias cosas que se pueden mejorar para hacer el sistema lo más eficiente posible.

En primer lugar, ofrecer la posibilidad de descargar el fichero JSON de cada curso de forma individual podría facilitar un análisis más exhaustivo de cada curso sin necesidad de seguir la URL al portal de cursos correspondiente.

Otro ámbito mejorable es el sistema de *scraping* de la página *Udemy*. Como se ha explicado a lo largo de este trabajo, no se ha podido aplicar la librería Scrapy a este portal debido a las especificaciones de su "robots.txt". El sistema actual no es tan eficiente en cuanto a tiempo comparado con los otros sistemas en los que sí se ha implementado Scrapy.

Tal y como está diseñado el sistema de visualización de cursos, hay datos que no se llegan a utilizar, principalmente los resultados de aprendizaje, ya que no se muestran en la página ni se tienen en cuenta para la taxonomía Bloom. Una posible mejora sería ofrecer la opción de mostrar todos los datos obtenidos de un curso en la página web y así tener toda la información disponible en un único lugar.

También se puede mejorar la forma de almacenar los datos. Un único fichero con toda la información de los cursos no es lo más eficiente y puede incluso alargar el tiempo de ejecución cuando se manejan muchos datos. La mejor forma de hacerlo sería mediante una base de datos, cuya implementación no ha sido posible en este trabajo debido a la limitación de tiempo.

A nivel personal, este proyecto ha sido un gran ejercicio de técnicas de *web scraping* y diseño de páginas web, áreas en las que antes de empezar solo poseía conocimientos básicos. También ha sido extremadamente útil en cuanto a la redacción de documentos oficiales, habilidad que probará ser útil a lo largo de mi carrera profesional.

4 Análisis de Impacto

Este proyecto prevé tener un mayor impacto entre aquellas personas u organizaciones cuya situación laboral se haya visto afectada por el auge de la IA y necesiten una forma de analizar las opciones para seguir adelante con su educación.

Para las empresas, este proyecto se presenta como una herramienta de modernizar sus empleos y conseguir empleados preparados y formados en campos relevantes.

Mediante la publicación de este trabajo al dominio público se pretende contribuir a los Objetivos de Desarrollo Sostenible[37], concretamente al "Objetivo 8: Promover el crecimiento económico inclusivo y sostenible, el empleo y el trabajo decente para todos". La meta 8.6: "De aquí a 2030, reducir considerablemente la proporción de jóvenes que no están empleados y no cursan estudios ni reciben capacitación" es relevante ya que promueve la educación y proporciona un portal único en que realizar consultas sobre los cursos disponibles en distintos portales. También es relevante la meta 8.2: "Lograr niveles más elevados de productividad económica mediante la diversificación, la modernización tecnológica y la innovación, entre otras cosas centrándose en los sectores con gran valor añadido y un uso intensivo de la mano de obra". En este sentido, este trabajo puede promover la diversificación y modernización de los trabajadores en el ámbito laboral, consiguiendo trabajadores más preparados para empleos más modernos.

A nivel ético, el *scraping* de páginas web ha llevado a gran cantidad de debates sobre su legalidad y sus implicaciones éticas y morales[38].

Por un lado, el *scraping* es una parte importante de internet que puede beneficiar tanto a empresas como consumidores, con ejemplos como páginas de comparación de precios o estudios de mercado. Por otro lado, el uso de los datos obtenidos puede ser preocupante, ya que se pueden dar casos en los que se haga *scraping* para conseguir contenido protegido por copyright o información personal sin consentimiento.

La toma de acciones legales contra el web *scraping* depende en gran parte del contexto y de la jurisdicción en la que se lleva a cabo. Por ejemplo, en EEUU se considera legal bajo el "*Computer Fraud and Abuse Act*"[39] (1986) y el "*Digital Millennium Copyright Act*"[40] (1998); en Europa existe la regulación "*General Data Protection Regulation (EU) 2016/679*"[41], que establece una lista de situaciones en las que se considera legal y busca evitar la obtención de información personal de los miembros de la Unión Europea.

En 2000, la compañía *eBay* ganó el juicio contra *Bidder's Edge*[42] por realizar *scraping* en su página web, alegando que la actividad saturaba el sistema y podría causar más daño a largo plazo.

En el caso de *Facebook* contra *Power Ventures* de 2009[43], el jurado determinó que el *scraping* de datos de *Facebook* correspondía una violación de la privacidad de sus usuarios.

En 2019, en el caso de *LinkedIn* contra *hiQ Labs*[44] se decretó que el *scraping* de páginas web es legal siempre y cuando sea sobre datos públicos.

Recientemente con el auge de la IA y los "*Large Language Models*" (*LLMs*) el debate sobre el *scraping* ha resurgido de nuevo, ya que este es vital para el entreno de los modelos de IA y los *LLMs* como GPT-4, que requieren grandes cantidades de información para su correcto funcionamiento.

En 2023, varios autores denunciaron a la compañía *OpenAI*[45] por utilizar sus libros, protegidos por *copyright*, para entrenar su modelo ("*ChatGPT*") sin consentimiento[46]. Autores como George RR Martin se han unido recientemente a la protesta alegando "robo sistemático a gran escala"[47]. Además, en diciembre de 2023 la editorial *New York Times* también ha denunciado a *OpenAI* y *Microsoft* por infracción de *copyright*[48], advocando por una regulación en el proceso de *scraping*.

Hay quienes defienden que para poder alcanzar el máximo desarrollo posible de la inteligencia artificial es necesario poder acceder a cualquier tipo de información. Otros consideran problemático el hecho de que no se apliquen las leyes de propiedad intelectual ni se respete la privacidad de los usuarios de internet. A estas preocupaciones se ha añade también la dificultad de eliminar datos una vez estos han sido utilizados para entrenar la inteligencia artificial.

En conclusión, el *scraping* de páginas web no se considera ilegal, pero puede llegar a ser utilizado de forma cuestionable. En el caso de este proyecto, al hacerse con fines académicos y recopilar información exclusivamente de dominio público no habría problemas de legalidad.

5 Bibliografía

[1] M. Navas-Loro, J. Arenas-Guerrero, y E. Montiel-Ponsoda, «AI4Labour: Reshaping labour force participation with artificial intelligence», CEUR Workshop Proc.

[2] J. Manyika et al., «Jobs lost, jobs gained: What the future of work will mean for jobs, skills, and wages», McKinsey & Company, 2017.

[3] «AI4Labour», Ai4labour.com. [En línea]. Disponible en: https://ai4labour.com/.

[4] T. Statement, «Modern Slavery Act», Amazonaws.com. [En línea]. Disponible en: https://coursera_assets.s3.amazonaws.com/footer/Modern+Slavery+Statement+26+April+2023.pdf.

[5] «Modern slavery statement», edX. [En línea]. Disponible en: https://www.edx.org/modern-slavery-statement.

[6] Udemy.com. [En línea]. Disponible en: https://www.udemy.com/.

[7] Zenrows.com. [En línea]. Disponible en: https://www.zenrows.com/blog/python-web-scraping-library#best-python-scraping-libraries.

[8] E. Pavlovskytė, «Scrapy vs. Beautiful soup: A comparison of web scraping tools», Oxylabs.io, 21-jul-2023. [En línea]. Disponible en: https://oxylabs.io/blog/scrapy-vs-beautifulsoup.

[9] «The selenium browser automation project», Selenium. [En línea]. Disponible en: https://www.selenium.dev/documentation/.

[10] «Scrapy 2.11 documentation — Scrapy 2.11.0 documentation», Scrapy.org. [En línea]. Disponible en: https://docs.scrapy.org/en/latest/.

[11] «The Linux Foundation», edX. [En línea]. Disponible en: https://www.edx.org/school/linuxfoundationx.

[12] «Web3 foundation», edX. [En línea]. Disponible en: https://www.edx.org/school/web3x.

[13] «About us», edX. [En línea]. Disponible en: https://www.edx.org/about-us.

[14] «React», React.dev. [En línea]. Disponible en: https://react.dev/.

[15] «API Reference», Nextjs.org. [En línea]. Disponible en: https://nextjs.org/docs/pages/api-reference.

[16] «Remix docs», Remix. [En línea]. Disponible en: https://remix.run/docs/en/main.

[17] B. FutureSchool, «Why is HTML Used in Web Pages?», BYJU'S Future School Blog, 14-mar-2023. [En línea]. Disponible en: https://www.byjusfutureschool.com/blog/why-is-html-used-in-web-pages/.

[18] «Angular», Angular.io. [En línea]. Disponible en: https://angular.io/.

[19] «Vue.Js», Vuejs.org. [En línea]. Disponible en: https://vuejs.org/.

[20] V. Coutinho, «Sistema de gestión de contenidos (CMS): ¿por qué implementarlo en tu empresa?», Rock Content - ES, 21-ago-2020. [En línea]. Disponible en: https://rockcontent.com/es/blog/cms/.

[21] «Wordpress • SDi», SDi Digital Group, 28-sep-2023. [En línea]. Disponible en: https://www.sdi.es/tecnologias/wordpress/.

[22] M. Woodward, «WordPress statistics: How many people use WordPress in 2024?», SearchLogistics, 31-may-2023. [En línea]. Disponible en: https://www.searchlogistics.com/learn/statistics/wordpress-statistics/.

[23] «Python Developers Survey 2022 Results», JetBrains: Developer Tools for Professionals and Teams. [En línea]. Disponible en: https://lp.jetbrains.com/python-developers-survey-2022/.

[24] «Getting started with Django», Django Project. [En línea]. Disponible en: https://www.djangoproject.com/start/.

[25] «Welcome to flask — flask documentation (3.0.X)», Palletsprojects.com. [En línea]. Disponible en: https://flask.palletsprojects.com/en/3.0.x/.

[26] «AI4Labour», Linkeddata.es. [En línea]. Disponible en: https://ai4labour.linkeddata.es/crec/.

[27] G. del docente MX, «¿Qué es la taxonomía de Bloom? Una definición para maestros», Guía del docente, 19-sep-2019. [En línea]. Disponible en: https://guiadeldocente.mx/que-es-la-taxonomia-de-bloom-una-definicion-para-maestros/.

[28] «La taxonomía de Bloom, una herramienta imprescindible para enseñar y aprender», CENTRO DEL PROFESORADO Tenerife Sur, 03-dic-2015. [En línea]. Disponible en: https://www3.gobiernodecanarias.org/medusa/edublog/cprofestenerifesur/2015/12/03/la-taxonomia-de-bloom-una-herramienta-imprescindible-para-ensenar-y-aprender/.

[29] mfdiaz, course_crawler: Copia de los scripts de scraping del proyecto Ai4Labour. .

[30] mfdiaz, TFGMiguelFernandez: TFG «Automatización de ingesta de datos y publicación en web de datos de educación». .

[31] Github.com. [En línea]. Disponible en: https://github.com/mfdiaz308/ai4labour.git.

[32] Codepen.io. [En línea]. Disponible en: https://codepen.io/inegoita/pen/YMrJGY.

[33] Redalyc.org. [En línea]. Disponible en: https://www.redalyc.org/journal/1804/180473611016/html/.

[34] M. D. Wilkinson et al., «The FAIR Guiding Principles for scientific data management and stewardship», Sci. Data, vol. 3, n.o 1, pp. 1-9, 2016.

[35] D. E. Fernández, «Buenas prácticas al trabajar con Git · The Git, the Bad and the Ugly», Gitbooks.io. [En línea]. Disponible en: https://david-estevez.gitbooks.io/the-git-the-bad-and-the-ugly/content/es/buenas-practicas-al-trabajar-con-git.html.

[36] X. Rigoulet, «The Python requirements file and how to create it», Learnpython.com, 13-ene-2022. [En línea]. Disponible en: https://learnpython.com/blog/python-requirements-file/.

[37] M. J. Gamez, «Objetivos y metas de desarrollo sostenible», Desarrollo Sostenible, 17-sep-2015. [En línea]. Disponible en:

https://www.un.org/sustainabledevelopment/es/objetivos-de-desarrollo-sostenible/.

[38] E. Hasson, «Is web scraping illegal?», Blog, 07-dic-2023. [En línea]. Disponible en: https://www.imperva.com/blog/is-web-scraping-illegal/.

[39] «NACDL - Computer Fraud and Abuse Act (CFAA)», NACDL - National Association of Criminal Defense Lawyers. [En línea]. Disponible en: https://www.nacdl.org/Landing/ComputerFraudandAbuseAct.

[40] U.S. Copyright Office, «The digital millennium copyright act», Copyright.gov. [En línea]. Disponible en: https://www.copyright.gov/dmca/.

[41] «General Data Protection Regulation (GDPR) – official legal text», General Data Protection Regulation (GDPR). [En línea]. Disponible en: https://gdpr-info.eu/.

[42] «EBay, Inc. V. Bidder's Edge, Inc., 100 F. supp. 2d 1058», Casetext.com, 24-may-2000. [En línea]. Disponible en: https://casetext.com/case/ebay-v-bidders-edge.

[43] «Facebook, Inc. v. Power Ventures, Inc., Case Number C 08-5780 JF (RS), [re: doc. no. 49]», Casetext.com, 22-oct-2009. [En línea]. Disponible en: https://casetext.com/case/facebook-inc-v-power-ventures.

[44] «HIQ LABS, INC. V. linkedin CORPORATION, no. 17-16783 (9th cir. 2022)», Justia Law. [En línea]. Disponible en: https://law.justia.com/cases/federal/appellate-courts/ca9/17-16783/17-16783-2022-04-18.html.

[45] «OpenAI», Openai.com. [En línea]. Disponible en: https://openai.com/.

[46] E. Creamer, «Authors file a lawsuit against OpenAI for unlawfully 'ingesting' their books», The guardian, The Guardian, 05-jul-2023.

[47] Associated Press, «George RR Martin and John Grisham among group of authors suing OpenAI», The guardian, The Guardian, 20-sep-2023.

[48] Reuters.com. [En línea]. Disponible en: https://www.reuters.com/legal/transactional/ny-times-sues-openai-microsoft-infringing-copyrighted-work-2023-12-27/.

6 Anexos

6.1 Anexo 1

Lista de palabras clave para el análisis de las descripciones de los cursos de cara a elaborar su taxonomía Bloom.

```
bloom_verbs_lists = [
      [
            "remember","cite","define","describe","draw","enumerate","
      identify","index","indicate","label","list","match","meet","name
      ","outline","point","quote","read","recall","recite","recognize"
      ,"record","repeat","reproduce","review","select","state","study"
      ,"tabulate","trace","write","interpret","observe","paraphrase","
      picture
      graphically","predict","review","rewrite","subtract","summarize"
      ,"translate","visualize","personalize","plot","practice","predic
      t","prepare","price","process","produce","project","provide","re
      late","round
      off","sequence","show","simulate","sketch","solve","subscribe","
      tabulate","transcribe","translate","use","recordar",
      "citar","definar","describir","dibujar","enumerar","identificar"
      ,"memorizar","indicar","etiquetar","listar","asignar","caracteri
      zar","nombrar","recitar","resumir","citar","leer","recordra","me
      morizar","reconocer","acordar","repetir","reproducir","revisar",
      "seleccionar","afirmar","estudiar","tabular","trazar","escribir"
      ,"interpretar","observar","parafrasear","encontrar","predecir","
      revisar","reescribir","restar","resumir","traducir","visualizar"
      ,"personalizar","dibujar","graficar","predecir","preparar","just
      ificar","procesar","producir","proyectar","proporcionra","relata
      r","narrar","secuenciar","mostrar","simular","bosquejar","resolv
      er","suscribir","tabular","transcribir","traducir","usar"
      ],
       [
            "understand","add","approximate","articulate","associate",
      "characterize","clarify","classify","compare","compute","contras
      t","convert","defend","describe","detail","differentiate","discu
      ss","distinguish","elaborate","estimate","example","explain","ex
      press","extend","extrapolate","factor","generalize","give","infe
      r","interact","interpolate","express","factor","figure","graph",
      "handle","illustrate","interconvert","investigate","manipulate",
      "modify","operate","proofread","query","relate","select","separa
      te","subdivide","train","transform","entender","añadir","aproxim
      ar","articular","asociar","caracterizar","aclarar","clasificar",
      "comparar","computar","contrastar","convertir","defender","descr
      ibir","detallar","diferenciar","discutir","identificar","elabora
      r","estimar","ejemplificar","explicar","expresar","extender","ex
      trapolar","reescribir","generalizar","dar","inferir","interactua
      r","interpolar","expresar","factorizar","figurar","graficar","ma
      nipular","ilustrar","convertir","investigar","manipular","modifi
      car","operar","revisar","consultar","relacionar","seleccionar","
      separar","subdividir","entrenar","transformar"
      ],
```

```
[
      "analyze","
analyze","audit","blueprint","breadboard","break
down","characterize","classify","compare","confirm","contrast","
correlate","detect","diagnose","diagram","differentiate","discri
minate","dissect","distinguish","document","ensure","examine","e
xplain","explore","figure
out","file","group","identify","illustrate","infer","interrupt",
"validate","verify","overhaul","plan","portray","prepare","presc
ribe","produce","program","rearrange","reconstruct","relate","re
organize","revise","rewrite","specify","summarize","analizar","a
uditar","ponderar","considerar","particularizar","caracterizar",
"clasificar","comparar","confirmar","contrastar","correlar","det
ectar","diagnosticar","dibujar","diferenciar","discriminar","dis
eccionar","distinguir","documentar","asegurar","examinar","expli
car","explorar","archivar","implementar","agrupar","identificar"
,"ilustrar","inferir","interrumpir","validar","verificar"
],
[
      "apply","acquire","adapt","allocate","alphabetize","apply"
,"ascertain","assign","attain","avoid","back
up","calculate","capture","change","classify","complete","comput
e","construct","customize","demonstrate","depreciate","derive","
determine","diminish","discover","draw","employ","examine","exer
cise","explore","expose","inventory","investigate","layout","man
age","maximize","minimize","optimize","order","outline","point
out","prioritize","aplicar","adquirir","adaptar","ubicar","alfab
etizar","determinar","asignar","obtener","evitar","respaldar","c
alcular","capturar","desarrollar","cambiar","clasificar","comple
tar","computar","programar","demostrar","implementar","derivar",
"determinar","reducir","simplificar","descubrir","dibujar","empl
ear","examinar","explorar","exponer","inventar","investigar","tr
azar","gestionar","maximizar","optimizar","minimizar","ordenar",
"compilar","calibrar","priorizar"
],
[
      "evaluate","appraise","assess","compare","conclude","contr
ast","counsel","criticize","critique","defend","determine","disc
riminate","estimate","evaluate","explain","grade","hire","interp
ret","judge","justify","measure","predict","prescribe","rank","r
ate","recommend","release","select","summarize","support","test"
,"improve","incorporate","integrate","interface","join","lecture
","model","modify","network","organize","outline","evaluar","val
orar","assess","comparar","concluir","contrastar","aconsejar","c
riticar","critique","defender","determinar","discriminar","estim
ular","evaluar","explicar","puntuar","contratar","interpretar","
juzgar","justificar","medir","predecir","anticipar","prescribir"
,"ordenar","recomendar","lanzar","seleccionar","resumir","apoyar
","testear","mejorar","incorporar","integrar","combinar","presen
tar","modelar","modificar","conectar","organizar"
```

```
      ],
      [
            "create","abstract","animate","arrange","assemble","budget
      ","categorize","code","combine","compile","compose","construct",
      "cope","correspond","create","cultivate","debug","depict","desig
      n","develop","devise","dictate","enhance","explain","facilitate"
      ,"format","formulate","generalize","generate","handle","import",
      "crear","abstraer","animar","diseñar","ensamblar","presupuestar"
      ,"categorizar","codificar","combinar","compilar","componer","con
      struir","resolver","corresponder","crear","cultivar","depurar","
      mostrar","diseñar","desarollar","concebir","dictar","mejorar","e
      xplicar"
      ]
]
```

6.2 Anexo 2

Código CSS de cada fila de la tabla de resultados en la página web, incluyendo el de los gráficos de la taxonomía Bloom.

```
<style>
      .result-item {
            border: 1px solid #333;
            padding-left: 7px;
            background-color: #fff;
        }


        .graph {
            margin-bottom:1em;
            font:normal 100%/150% arial,sans-serif;
        }


        .graph caption {
            font:bold 150%/120% arial,sans-serif;
            padding-bottom:0.33em;
        }
```

```
.graph tbody th {
    text-align:right;
}

@supports (display:grid) {

    @media (min-width:32em) {

        .graph {
            display:block;
            width:350px;
            height:150px;
        }

        .graph caption {
            display:block;
        }

        .graph thead {
            display:none;
        }

        .graph tbody {
            position:relative;
            display:grid;
            grid-template-columns:repeat(auto-fit, minmax(2em,
1fr));
            column-gap:2.5%;
            align-items:end;
            height:100%;
            margin:3em 0 1em 2.8em;
```

```
    padding:0 1em;
    border-bottom:2px solid rgba(0,0,0,0.5);
    background:repeating-linear-gradient(
    180deg,
    rgba(170,170,170,0.7) 0,
    rgba(170,170,170,0.7) 1px,
    transparent 1px,
    transparent 20%
    );
}

.graph tbody:before,
.graph tbody:after {
    position:absolute;
    left:-3.2em;
    width:2.8em;
    text-align:right;
    font:bold 80%/120% arial,sans-serif;
}

.graph tbody:before {
    content:"100%";
    top:-0.6em;
}

.graph tbody:after {
    content:"0%";
    bottom:-0.6em;
}

.graph tr {
```

```
    position:relative;
    display:block;
}

.graph tr:hover {
    z-index:999;
}

.graph th,
.graph td {
    display:block;
    text-align:center;
}

.graph tbody th {
    position:absolute;
    top:-3em;
    left:0;
    width:100%;
    font-weight:normal;
    text-align:center;
    white-space:nowrap;
    text-indent:0;
    transform:rotate(-60deg);
}

.graph tbody th:after {
    content:"";
}

.graph td {
```

```
    width:100%;
    height:100%;
    background:#007bff;
    border-radius:0.5em 0.5em 0 0;
    transition:background 0.5s;
}

.graph tr:hover td {
    opacity:0.7;
}

.graph td span {
    overflow:hidden;
    position:absolute;
    left:50%;
    top:50%;
    width:0;
    padding:0.5em 0;
    margin:-1em 0 0;
    font:normal 85%/120% arial,sans-serif;
    font-weight:bold;
    opacity:0;
    transition:opacity 0.5s;
    color:white;
}

.toggleGraph:checked + table td span,
.graph tr:hover td span {
    width:4em;
    margin-left:-2em;
    opacity:1;
```

```
                }
            }


        }
</style>
```

Este documento esta firmado por

	**Firmante**	CN=tfgm.fi.upm.es, OU=CCFI, O=ETS Ingenieros Informaticos - UPM, C=ES
	Fecha/Hora	Sun Jan 14 21:00:25 CET 2024
	Emisor del Certificado	EMAILADDRESS=camanager@etsiinf.upm.es, CN=CA ETS Ingenieros Informaticos, O=ETS Ingenieros Informaticos - UPM, C=ES
	Numero de Serie	561
	Metodo	urn:adobe.com:Adobe.PPKLite:adbe.pkcs7.sha1 (Adobe Signature)