



UNIVERSIDAD DE MÁLAGA



GRADO EN INGENIERÍA INFORMÁTICA

HERRAMIENTA DE CRAWLING DE  
PUBLICACIONES CIENTÍFICAS

SCIENTIFIC PUBLICATIONS CRAWLING TOOL

Realizado por  
ENRIQUE LÓPEZ ENCINAS

Tutorizado por  
EDUARDO GUZMÁN DE LOS RISCOS

Departamento  
LENGUAJE Y CIENCIAS DE LA COMPUTACIÓN  
UNIVERSIDAD DE MÁLAGA

MÁLAGA, JUNIO DE 2024



UNIVERSIDAD  
DE MÁLAGA



ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA  
GRADUADO EN INGENIERÍA INFORMÁTICA

## **HERRAMIENTA DE CRAWLING DE PUBLICACIONES CIENTÍFICAS**

### **SCIENTIFIC PUBLICATIONS CRAWLING TOOL**

Realizado por  
**Enrique López Encinas**

Tutorizado por  
**Eduardo Guzmán De Los Riscos**

Departamento  
**Lenguaje y Ciencias de la Computación**

UNIVERSIDAD DE MÁLAGA  
MÁLAGA, JUNIO DE 2024

Fecha defensa: julio de 2024



# Resumen

En la era de la información, el acceso rápido y eficiente a publicaciones científicas es esencial para investigadores, académicos y estudiantes. Este proyecto presenta una aplicación web desarrollada en Angular que automatiza el proceso de recopilación de artículos científicos desde Google Scholar. La aplicación utiliza técnicas de *scraping* para extraer información clave de los artículos, incluyendo el número de citas, autores y fechas de publicación.

La funcionalidad principal de la aplicación permite a los usuarios especificar parámetros de búsqueda personalizados, como el título del artículo, el idioma de búsqueda y el número de páginas de Google Scholar que desean explorar. Esto facilita una búsqueda más precisa y relevante de la literatura científica, ahorrando tiempo y esfuerzo a los usuarios.

Una característica destacada de esta aplicación es la capacidad de almacenar las búsquedas realizadas en una base de datos. Esto permite a los usuarios cargar búsquedas anteriores sin necesidad de repetir el proceso de *scraping*, optimizando así el acceso a la información previamente obtenida.

Este proyecto no solo proporciona una herramienta valiosa para la búsqueda y gestión de artículos científicos, sino que también destaca el uso de tecnologías web modernas y métodos de *scraping* eficientes. A través de la implementación de Angular para el desarrollo frontend y la integración con bases de datos para el almacenamiento de información, esta aplicación representa una solución innovadora para los desafíos actuales en la recopilación y manejo de datos académicos.

## **Palabras clave:**

Web Scraping, Google Scholar, Artículos científicos, Angular, Python.



# Abstract

In the information age, quick and efficient access to scientific publications is essential for researchers, academics, and students. This project presents a web application developed in Angular that automates the process of collecting scientific articles from Google Scholar. The application uses *scraping* techniques to extract key information from the articles, including the number of citations, authors, and publication dates.

The main functionality of the application allows users to specify custom search parameters, such as the article title, search language, and the number of Google Scholar pages they want to scrape. This facilitates a more precise and relevant search of the scientific literature, saving users time and effort.

A standout feature of this application is the ability to store performed searches in a database. This enables users to load previous searches without needing to repeat the *scraping* process, thus optimizing access to previously obtained information.

This project not only provides a valuable tool for the search and management of scientific articles but also highlights the use of modern web technologies and efficient *scraping* methods. Through the implementation of Angular for frontend development and integration with databases for information storage, this application represents an innovative solution to the current challenges in the collection and management of academic data.

## **Keywords:**

Web scraping, Scientific articles, Google Scholar, Data management, Angular, Python.



# Índice

Resumen .....	1
Abstract .....	1
Índice .....	1
Introducción .....	1
1.1 Motivación .....	1
1.2 Objetivos .....	2
1.3 Contexto.....	2
1.4 Tecnologías utilizadas .....	3
1.5 Metodología empleada .....	5
1.6 Estructura de la memoria .....	6
Especificaciones y análisis.....	7
Análisis de requisitos .....	7
Casos de uso.....	8
Diagrama de secuencia .....	9
Aprendizaje de las tecnologías y lenguajes usados .....	11
Desarrollo del web scraper .....	13
Desarrollo del Backend .....	15
Script app.py .....	15
Base de datos SQLite .....	17
Desarrollo del Frontend.....	19
Integración y Pruebas .....	23
Estructura del proyecto.....	23
Pruebas sobre el script de <i>scraping</i> .....	23
Pruebas sobre la base de datos .....	24
Pruebas sobre el Frontend y Backend.....	24
Conclusiones y líneas futuras .....	26
Referencias .....	29
Listado de figuras .....	31
Manual de Instalación .....	33
Instalación y ejecución en Windows:.....	33
Manual de usuario .....	35





# 1

# Introducción

## 1.1 Motivación

La motivación detrás de este proyecto surge de la creciente necesidad de acceder de manera rápida y eficiente a la vasta cantidad de información científica disponible en línea. Los investigadores, académicos y estudiantes a menudo enfrentan el desafío de buscar y recopilar artículos relevantes entre una enorme cantidad de publicaciones, lo que puede ser un proceso tedioso y consume mucho tiempo.

Google Scholar es una fuente inestimable de artículos científicos, pero su interfaz no proporciona herramientas avanzadas para la recopilación masiva y el análisis de datos específicos. Esto plantea una barrera para aquellos que necesitan realizar revisiones extensivas de la literatura o mantener un seguimiento constante de nuevas publicaciones en su campo de estudio.

Desarrollar una aplicación web que automatice el proceso de extracción (*scraping*) en Google Scholar y permita la personalización de los parámetros de búsqueda ofrece una solución poderosa a este problema. Al facilitar la extracción y el almacenamiento de datos relevantes, los usuarios pueden optimizar su tiempo de investigación, centrándose en el análisis y la interpretación de la información en lugar de en su recolección.

Además, la posibilidad de guardar búsquedas y acceder a ellas posteriormente sin necesidad de repetir el proceso de *scraping* no solo mejora la eficiencia, sino que también proporciona un valor añadido significativo al permitir la construcción de una base de datos personalizada y reutilizable de artículos científicos.

Este proyecto combina la aplicación de tecnologías web modernas y técnicas de *scraping* para abordar una necesidad real en el ámbito académico, ofreciendo una herramienta que potencia la capacidad investigativa y mejora significativamente la gestión del conocimiento científico.

## 1.2 Objetivos

El objetivo principal de este proyecto es desarrollar una aplicación web eficiente y fácil de usar, basada en Angular, que automatice el proceso de búsqueda y recopilación de artículos científicos desde Google Scholar. La aplicación permitirá a los usuarios especificar parámetros de búsqueda personalizados, tales como títulos de artículos, idiomas de búsqueda y el número de páginas a extraer, para obtener resultados precisos y relevantes.

El desarrollo de esta aplicación ha presentado desafíos significativos debido a la falta de una API oficial para la recolección de datos desde Google Scholar y otras páginas web de artículos científicos. Para superar este obstáculo, se han analizado diferentes fuentes de artículos científicos y se han implementado técnicas avanzadas de *scraping* web para extraer la información necesaria.

Además, la aplicación tendrá la capacidad de almacenar estas búsquedas en una base de datos, facilitando el acceso a resultados anteriores sin la necesidad de realizar nuevas búsquedas, optimizando así el tiempo y esfuerzo de los usuarios. De esta manera, se busca proporcionar una herramienta que no solo mejore la eficiencia en la recolección de datos académicos, sino que también contribuya significativamente a la gestión y análisis de la literatura científica disponible, apoyando a investigadores, académicos y estudiantes en sus esfuerzos de investigación.

## 1.3 Contexto

Como se ha expuesto en secciones anteriores, esta aplicación surge de la necesidad de proporcionar a los usuarios información útil de artículos científicos de manera rápida y eficiente. Una de las primeras fases del proyecto consistió en determinar la fuente de datos más adecuada para recopilar esta información. Para ello, se analizaron diferentes páginas web, que se describen a continuación:

**Web of Science** (Clarivate, 2024): Es una plataforma de investigación en línea que ofrece acceso a una amplia colección de bases de datos de referencias bibliográficas y citas científicas. Es una herramienta esencial para académicos, investigadores y estudiantes que buscan información sobre artículos científicos, revistas académicas y otros recursos relacionados con la investigación. A pesar de sus numerosas cualidades, como la capacidad de realizar consultas en varias bases de datos, numerosos filtros y la posibilidad de crear alertas de publicaciones para temas específicos, esta plataforma fue descartada debido a su modelo de suscripción mensual.

**Open Researcher and Contributor ID (ORCID, 2024)**: Es un sistema de identificación digital único para investigadores y autores académicos. Proporciona un identificador persistente que distingue a los investigadores y asegura que sus contribuciones académicas sean correctamente atribuidas, independientemente de las variaciones en la forma de sus nombres. Sin embargo, se descartó esta opción ya que, en el momento de su evaluación, no se consideraba una forma eficaz de identificar a los investigadores, dado que muchos autores no están registrados en esta plataforma.

**Digital Bibliography & Library Project (DBLP, 2024):** Es una base de datos en línea que proporciona información bibliográfica detallada sobre publicaciones en los campos de informática y tecnología de la información. Es una herramienta fundamental para académicos, investigadores y estudiantes que buscan acceder a la literatura científica y técnica en estas áreas. No obstante, presenta dos problemas significativos: la falta de paginación (el contenido se carga conforme el usuario desplaza la página hacia abajo), lo que dificulta la recolección de datos mediante *scraping*, y su especialización en artículos del ámbito de la informática, lo que limita los perfiles de usuarios interesados en la aplicación web.

**Google Scholar (Google Scholar, 2024):** Es un motor de búsqueda especializado en literatura académica que permite a los usuarios encontrar artículos, tesis, libros, patentes y otros documentos científicos. Es ampliamente utilizado por estudiantes, investigadores y académicos para acceder a una vasta colección de publicaciones académicas.

Tras analizar diversas páginas web, se concluyó que Google Scholar era la opción más adecuada. Ofrece acceso gratuito a una amplia variedad de recursos, como artículos científicos, perfiles de investigadores, patentes y filtros de búsqueda, además de contar con datos bien estructurados.

## 1.4 Tecnologías utilizadas

El desarrollo de esta aplicación web ha requerido el uso de diversas tecnologías, cada una desempeñando un papel crucial en la implementación y funcionamiento del proyecto. A continuación, se describen las principales tecnologías utilizadas:

### Backend: Python 11

Para el backend, se ha utilizado Python 11, un lenguaje de programación robusto y versátil. En concreto, se han empleado las siguientes librerías:

- **pandas** (pandas development team, 2024): Esta librería es fundamental para la manipulación y análisis de datos. Ha permitido manejar de manera eficiente los datos extraídos de los artículos científicos creando marcos de datos que facilitan su estructuración y lectura.
- **beautifulsoup4** (Richardson, L. 2024): Utilizada para el *scraping* web, esta librería facilita la extracción de datos de las páginas HTML de Google Scholar, permitiendo una navegación y obtención de información precisa, así como el uso de filtros por nombres de clases o etiquetas sobre los elementos HTML de una página web.
- **requests** (Python Software Foundation, 2024): Empleada para realizar solicitudes HTTP, esta librería ha permitido descargar el contenido de las páginas web de las que desea extraer la información.

- **click** (Click, 2023): Esta librería proporciona una manera sencilla de crear interfaces de línea de comandos, facilitando la ejecución de scripts de *scraping* y otras tareas automatizadas. Permite declarar atributos que recibe el script al ejecutarse
- **sqlite3** (SQLite Consortium, 2024): Es una biblioteca integrada en Python que proporciona una API para trabajar con bases de datos SQLite. Permite crear, conectar, manipular y consultar bases de datos SQLite directamente desde el código Python utilizando sentencias SQL estándar.
- **flask** (Welcome To Flask — Flask Documentation (3.0.x), s.f.): Flask es un microframework para aplicaciones web en Python. Se ha utilizado para manejar las solicitudes del frontend, proporcionando una interfaz eficiente y ligera para la comunicación entre el cliente y el servidor.

### **Frontend: Angular**

El framework Angular se ha utilizado para el desarrollo del frontend de la aplicación. Angular es conocido por su capacidad para construir aplicaciones web dinámicas y de alto rendimiento, proporcionando una experiencia de usuario fluida e interactiva. Junto con Angular, se han empleado las siguientes tecnologías:

- **Bootstrap** (Bootstrap, 2024): Un framework de CSS que facilita el diseño responsivo y estilizado de la interfaz de usuario, asegurando que la aplicación se vea y funcione bien en diferentes dispositivos y tamaños de pantalla.
- **Angular Material** (Team, A. C. (s. f.)): Una librería de componentes de interfaz de usuario que sigue las directrices de Material Design de Google, proporcionando una apariencia moderna y consistente a la aplicación.
- **Tailwind CSS** (Tailwind, (s. f.)): Un framework de utilidades CSS que permite diseñar rápidamente interfaces atractivas y personalizables mediante clases de utilidad predefinidas.

### **Base de Datos: SQLite**

Para el almacenamiento de datos, se ha utilizado una base de datos SQLite. SQLite es una base de datos ligera y de código abierto que se integra fácilmente con aplicaciones Python. Sus principales ventajas incluyen:

- **Facilidad de uso:** SQLite no requiere una configuración de servidor compleja, lo que simplifica su integración y mantenimiento.
- **Eficiencia:** A pesar de ser ligera, SQLite es suficientemente robusta para manejar las necesidades de almacenamiento de la aplicación, permitiendo un acceso rápido y eficiente a los datos guardados.

## 1.5 Metodología empleada

En el desarrollo de este proyecto se ha optado por utilizar una metodología ágil, conocida por su gran flexibilidad a la hora de realizar diferentes tareas según se vayan necesitando. Esta metodología permite una adaptación continua y rápida a los cambios y requerimientos del proyecto, facilitando un desarrollo más dinámico y eficiente.

### Características Principales

- **Flexibilidad:**

La metodología ágil permite modificar y ajustar las tareas en función de las necesidades emergentes del proyecto. Esto incluye la capacidad de incorporar nuevos requisitos o cambios en cualquier fase del desarrollo.

- **Iteración y Revisión Continua:**

Aunque el proyecto se estructura en fases de trabajo, la metodología ágil facilita moverse entre estas fases de manera no lineal. Si se encuentran errores que necesitan ser corregidos urgentemente o si es necesario rehacer o modificar una parte anterior del proyecto, se puede retroceder y abordar estos problemas de inmediato.

- **Mejora Continua:**

La metodología ágil promueve la mejora continua a través de ciclos iterativos de desarrollo, donde cada iteración se revisa y ajusta según la retroalimentación recibida por el tutor. Esto asegura que el proyecto evoluciona constantemente hacia una mejor calidad y alineación con los objetivos y expectativas del proyecto.

- **Colaboración y Comunicación:**

Fomenta una comunicación abierta y continua entre el tutor y el alumno. Esta colaboración estrecha permite identificar rápidamente los problemas y encontrar soluciones efectivas de manera conjunta.

### Control de Versiones: Git y GitHub

Para el control de versiones, se han utilizado Git y GitHub, herramientas esenciales para gestionar el código fuente y colaborar de manera efectiva.

Git Es un sistema de control de versiones distribuido que permite realizar seguimiento de los cambios en el código, facilitando la colaboración entre varios desarrolladores y la gestión de distintas versiones del proyecto.

GitHub es una plataforma basada en la nube que permite alojar repositorios Git. GitHub proporciona herramientas adicionales para la colaboración, como *pull requests*, revisión de código, y la integración continua.

## 1.6 Estructura de la memoria

En este primer capítulo introductorio se han estudiado las motivaciones y objetivos que persigue el proyecto, así las fuentes de datos que se han analizado como paso previo al desarrollo de la aplicación. Posteriormente se describen las tecnologías que han hecho posible llevar a cabo el desarrollo de la aplicación web. Por último, se explica cuál ha sido la metodología de trabajo que se ha mantenido, detallando las ventajas de esta.

En el capítulo 2 se llevará a cabo un análisis de los requisitos del sistema, asegurando que todas las necesidades y expectativas de los usuarios finales y otras partes interesadas sean claramente comprendidas y documentadas. Se incluyen aquí varios diagramas que ayudarán a comprender mejor el alcance del proyecto.

Durante el capítulo 3 se contextualiza el proyecto, explicando de donde surge la necesidad de realizar este proyecto, el análisis previo de las páginas web que contienen publicaciones científicas o información relevante sobre las mismas y el motivo que justifica la elección de Google Scholar sobre todas ellas.

En el capítulo 4 se habla sobre el camino que ha seguido el estudiante para familiarizarse con todas las tecnologías y lenguajes que se han usado a lo largo del desarrollo de este trabajo, describiendo también los recursos que ha usado.

Durante los capítulos 5, 6 y 7 se profundiza más en el desarrollo del script de *scrapping*, el Backend y el Frontend respectivamente, introduciendo los conceptos necesarios y dejando claro la estructura de la aplicación web.

En el capítulo 8 se explica cómo se ha trabajado la documentación y se han planteado las pruebas sobre el código implementado para asegurar el correcto funcionamiento de todas sus partes.

El capítulo 9 trata sobre las conclusiones obtenidas tras acabar la implementación de todos los capítulos anteriores y se exponen posibles líneas futuras para seguir ampliando funcionalidades que ofrece esta aplicación web.

En el apartado Referencias se deja constancia de las referencias bibliográficas consultadas por el alumno durante el desarrollo de la aplicación, que incluyen libros, video tutoriales y guías repartidas por internet.

Como anexo se incluye una guía de instalación para los sistemas operativos Windows 10 y la distribución de Linux Ubuntu. De esta forma, se espera facilitar la implementación, por parte de futuros desarrolladores, de las líneas futuras propuestas en el capítulo 9.

# 2

## Especificaciones y análisis

### **Análisis de requisitos**

Tras un análisis de las funcionalidades que debería tener esta aplicación web, se han determinado los siguientes requisitos:

#### 1. Requisitos funcionales

- a. El usuario puede introducir parámetros de búsqueda: cadena de texto, idioma y número de páginas de Google Scholar a consultar.
- b. Los resultados de la búsqueda se mostrarán en una tabla ordenable por columnas (título, autor, año, etc.).
- c. Guardar Búsqueda: El usuario puede guardar la búsqueda en la base de datos para consultas futuras.
- d. Recargar Búsqueda: El usuario puede recargar búsquedas guardadas y visualizarlas nuevamente en la tabla.
- e. Borrar Búsqueda: El usuario puede borrar búsquedas guardadas si ya no son necesarias.
- f. El usuario puede consultar una guía de usuario de la aplicación web.



## 2. Requisitos no funcionales

- La aplicación debe ser capaz de procesar y mostrar resultados de búsquedas de varias páginas de Google Scholar de manera eficiente.
- La aplicación no debe ser bloqueada por Google al no tratarse de una búsqueda realizada por una persona.
- La interfaz debe ser intuitiva y fácil de usar.
- La guía de usuario debe estar accesible y ser clara en sus instrucciones.
- Las búsquedas guardadas deben estar disponibles y no deben perderse con el tiempo.

## Casos de uso

En la Figura 1 observamos los casos de uso de la aplicación. Un usuario puede consultar la guía de usuario para entender su funcionamiento, buscar artículos o consultar búsquedas que se hayan guardado con anterioridad.

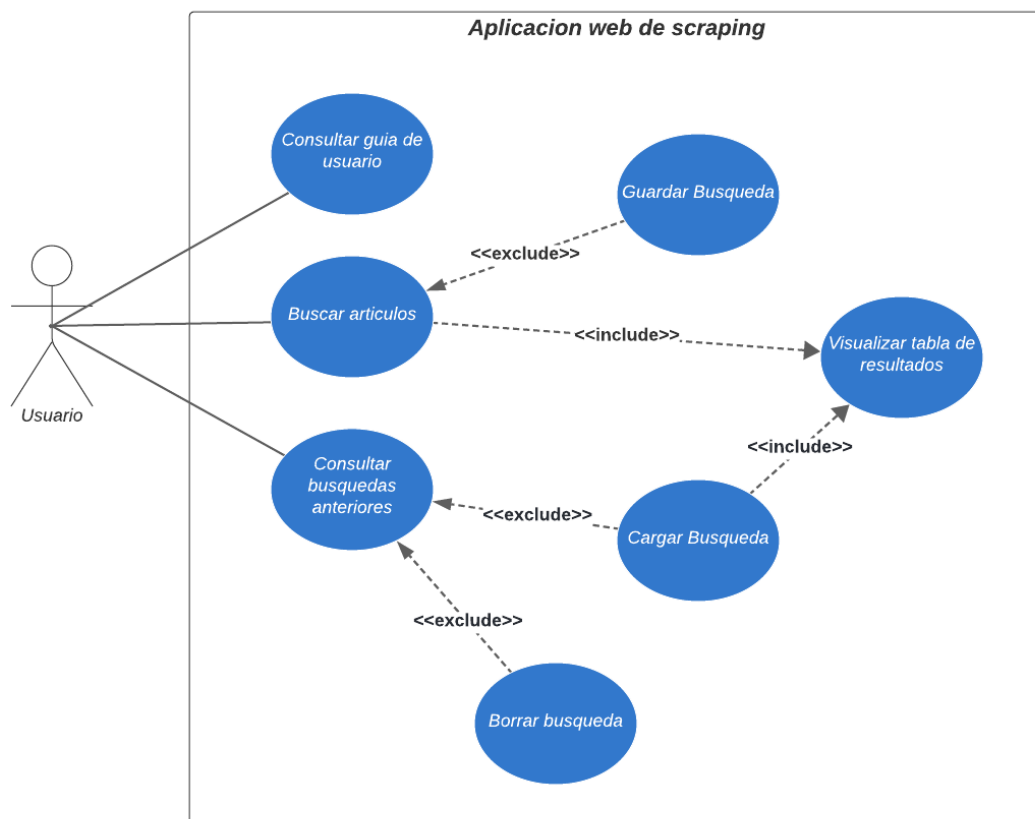
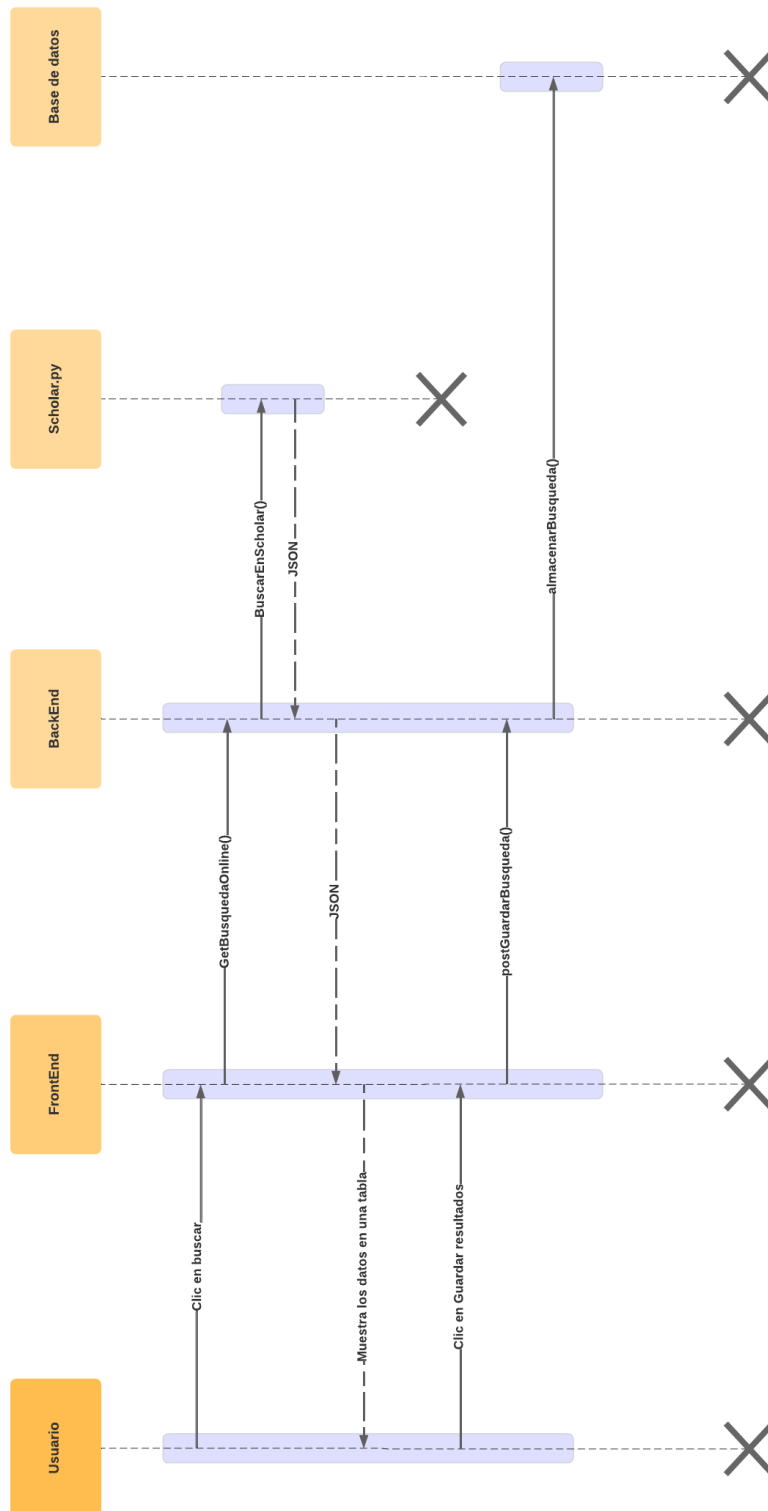


Figura 1: Casos de uso.

## Diagrama de secuencia

A continuación, en la Figura 2, se muestra el diagrama de secuencia correspondiente al caso de uso más frecuente de la aplicación, mostrando el proceso en el que un usuario realiza una búsqueda en la aplicación web, y después de revisar los datos obtenidos, decide guardarlos para poder consultarlos más tarde.



**Figura 2:** Diagrama de flujo del proceso más común.



# 3

## Aprendizaje de las tecnologías y lenguajes usados

Uno de los desafíos iniciales en la realización de este trabajo de fin de grado fue adquirir conocimientos sobre las tecnologías empleadas en el proyecto. La familiarización con las herramientas y plataformas necesarias para el desarrollo de la aplicación constituyó un componente crucial en el proceso de investigación y desarrollo. Este proceso de aprendizaje abarcó la comprensión de los lenguajes de programación, *frameworks*, bases de datos y otros recursos tecnológicos relevantes para la implementación del proyecto. La adquisición de estos conocimientos proporcionó una base sólida para el desarrollo efectivo de la aplicación y contribuyó significativamente al éxito del proyecto en su conjunto.

A continuación, se procede a describir los recursos utilizados como preparación previa al uso de las tecnologías ya mencionadas.

La página web GitHub Guides (git Guides. 2024) proporciona una serie de recursos y tutoriales diseñados para ayudar a los usuarios a aprender y comprender los conceptos básicos y avanzados de Git, así como las mejores prácticas para su uso eficaz en proyectos de desarrollo de software. Estos recursos incluyen guías paso a paso, tutoriales interactivos y documentación detallada sobre una variedad de temas relacionados con Git y GitHub. Desde la configuración inicial de un repositorio hasta la colaboración con otros usuarios en proyectos de código abierto, GitHub Guides ofrece una amplia gama de materiales educativos para usuarios de todos los niveles de experiencia.

Durante el proceso de aprendizaje de Python, se utilizaron dos recursos principales: la página web oficial de Python, python.org, y el asistente virtual ChatGPT.

- Python.org (Welcome To Python.org, 2024): La página oficial de Python proporciona una amplia variedad de recursos educativos, incluyendo documentación detallada, tutoriales y guías para aprender Python desde cero. Esta plataforma fue fundamental para adquirir conocimientos sobre los fundamentos del lenguaje y para explorar sus características más avanzadas. Además, Python.org sirvió como una referencia confiable para consultar la sintaxis, las bibliotecas estándar y las mejores prácticas de programación en Python.

Durante el proceso de aprendizaje de Angular, se utilizaron dos recursos principales: "Tour of Heroes" y Flexbox Froggy.

- "Tour of Heroes" - Guía Paso a Paso de Angular (Angular, s. f.): "Tour of Heroes" es una guía paso a paso proporcionada en la documentación oficial de Angular que te guía a través de la creación de una aplicación Angular completa desde cero. Esta guía es ideal para principiantes, ya que cubre los conceptos básicos de Angular mientras desarrollas una aplicación real. Desde la configuración inicial hasta la implementación de características avanzadas, "Tour of Heroes" te proporciona una introducción completa al desarrollo de aplicaciones con Angular.
- Flexbox Froggy - Juego Interactivo de Flexbox (Flexbox Froggy, (s. f.)): Flexbox Froggy es un juego interactivo en línea que enseña los conceptos básicos de Flexbox de una manera divertida y práctica. A través de una serie de desafíos, Flexbox Froggy te guía a través de los diferentes atributos y propiedades de Flexbox, permitiéndote experimentar directamente con la disposición y el diseño de elementos en una página web. Este recurso es excelente para visualizar y comprender cómo funciona Flexbox y cómo puedes utilizarlo para diseñar interfaces web flexibles y receptivas.

Además de estos recursos, ChatGPT ha demostrado ser de gran ayuda para consultas rápidas y aclaraciones sobre conceptos específicos de Python, Angular, Git y otras tecnologías. ChatGPT proporcionó respuestas rápidas y precisas a preguntas relacionadas con la sintaxis de estas herramientas, la resolución de problemas de código y la comprensión de conceptos complejos. Esta interacción bidireccional con ChatGPT ayudó a consolidar y reforzar el aprendizaje adquirido a través de otros recursos, permitiendo una comprensión más profunda y completa del lenguaje Python.

# 4

## Desarrollo del web scraper

El proceso de extracción de información y datos de sitios web de manera automatizada se conoce como Web Scraping. Se usan bots de software para analizar el contenido de las páginas web y extraer información relevante de ellas.

El *web scraper* navega por la página web, descarga el contenido HTML y luego extrae los datos específicos utilizando técnicas de análisis y manipulación de datos. Se pueden extraer textos, imágenes, enlaces o cualquier elemento que se pueda definir con el lenguaje HTML.

Esta técnica se usa en variedad de casos de uso, incluyendo la recopilación de datos para el análisis de mercado, monitorización de precios y generación de contenido automatizado, entre otros.

En este caso, el software se ha desarrollado en el lenguaje Python, con la ayuda de las librerías `beautifulsoup` y `requests`, como ya se ha explicado en el capítulo 1. A continuación se explica la estructura y los métodos que componen "scholar.py".

Gran parte de la recopilación y análisis de los datos se controla en la función denominada "buscar()", del archivo "scholar.py". La función comienza definiendo variables, entre los que se encuentran los filtros que se pueden aplicar a la búsqueda (Idioma, texto de la búsqueda y número de páginas). también se define una variable llamada "raiz" que almacena la dirección base de Google Scholar, sobre la que se añadirán los diferentes parámetros de búsqueda, tal y como se muestra en la Figura 3.

```

# direccion base en la que buscar (por ahora google scholar)
raiz = 'https://scholar.google.es/'

# Cambiamos el string a buscar para poder meterlo en la url
busqueda = busqueda.replace(' ', '+')

# iteramos por las diferentes paginas hasta procesar numero de paginas especificado
cont=0
while(cont <= paginas):
    # modifica la url para añadir los parametros de bsuqueda
    url = raiz+f'scholar?hl={idioma}&as_sdt=0%2C5&q={busqueda}&start={str(cont)}0'
    # obtenemos el contenido de lapagina y lo preprocesamos con bs4
    page = requests.get(url)
    soup = BeautifulSoup(page.content, 'html.parser')

```

**Figura 3:** Fragmento del código de scholar.py, en el método buscar() donde se modifica la URL para buscar en Scholar.

Se hace uso de la biblioteca BeautifulSoup para convertir el contenido recibido al a un objeto del tipo BeautifulSoup, que permite navegar por el árbol de elementos HTML. Existen dos métodos principales para encontrar los datos que se necesitan: mediante el uso de etiquetas y mediante el uso de atributos. En el desarrollo de este script se usaron ambos, y, junto con ayuda de expresiones regulares, se consiguió rescatar toda la información de esta página web, como se puede ver en la Figura 4.

```

#separamos la pagina en 2 variables, la que contiene el enlace al documento y el resto
cuerpo = i.find('div', class_='gs_ri')
enlace = i.find('div', class_='gs_or_ggsm')
if cuerpo is not None:

    # buscamos el titulo
    if cuerpo.find('h3', class_='gs_rt').find('a'):
        tit = cuerpo.find('h3', class_='gs_rt').find('a')
        #eliminamos los caracteres no alfanumericos del titulo
        titulos.append(re.sub(r'^a-zA-Z0-9\s', '', tit.text))
    else: titulos.append("No encontrado")

```

**Figura 4:** Fragmento del código de scholar.py donde se usan etiquetas y atributos para filtrar el contenido de la página web.

# 5

## Desarrollo del Backend

En este capítulo se detalla el desarrollo del Backend de la aplicación web, un componente esencial que se encarga de gestionar la lógica del servidor y la interacción con la base de datos. El Backend se ha implementado principalmente en dos partes: un script Python denominado *app.py* y una base de datos en SQLite.

El script *app.py* es el núcleo de la aplicación, ofreciendo funcionalidades clave que incluyen la búsqueda en la base de datos, la búsqueda en la web y la filtración de datos. Por otro lado, la base de datos en SQLite actúa como el almacenamiento principal de la aplicación.

### **Script *app.py***

Este archivo desarrollado en Python hace la función de servidor Backend con ayuda de la biblioteca Flask. Para facilitar su comprensión, el código posee varios comentarios explicando las partes de mayor dificultad. Consta de varios Endpoints que se explican a continuación:



```

# Busca los articulos de una determinada busqueda
@app.route('/api/busqueda-bd', methods = ['GET'])
def buscarPorId():
    # creamos la conexion con la base de datos (si no existe la db, la crea)
    conn = sqlite3.connect('databaseTFG.db')
    c = conn.cursor()
    # recibimos el id de la busqueda que queremos encontrar en la bd
    id = request.args.get('idBusqueda')
    # Ejecutamos la sentencia SQL
    c.execute("""
        SELECT a.titulo, a.citas, a.fecha_publicacion, a.enlace, a.aut_1, a.aut_2, a.aut_3 FROM articulos a
        INNER JOIN busquedas_articulos ba ON a.id = ba.articulo_id
        INNER JOIN busquedas b ON b.id = ba.busqueda_id
        WHERE b.id = ?; """, (id,))
    # guardamos en una variable (una lista) los resultados de la consulta
    filas = c.fetchall()
    # Iteramos sobre cada fila de la consulta SQL y la guardamos en una lista
    resultados = []
    for fila in filas:
        rdo = {
            'Titulo': fila[0],
            'Citas': fila[1],
            'Año': fila[2],
            'Enlace': fila[3],
            'Autor 1': fila[4],
            'Autor 2': fila[5],
            'Autor 3': fila[6],
        }
        resultados.append(rdo)
    # Creamos el json que devuelve la funcion
    resultados_json = json.dumps(resultados, indent=4)
    # cerramos la conexion con la bd
    conn.close()
    return resultados_json

```

Figura 5: Ejemplo de un Endpoint de app.py con los comentarios para facilitar su comprensión.

- **Endpoint `"/api/buscar-online"`:** Expone la función `"buscarEnScholar()"`, que utiliza tres parámetros pasados desde el FrontEnd (idioma, búsqueda y paginas) e invoca al método `"buscar()"` del *web scraper* que se encuentra en el archivo `scholar.py` (ya explicado en el capítulo 3).

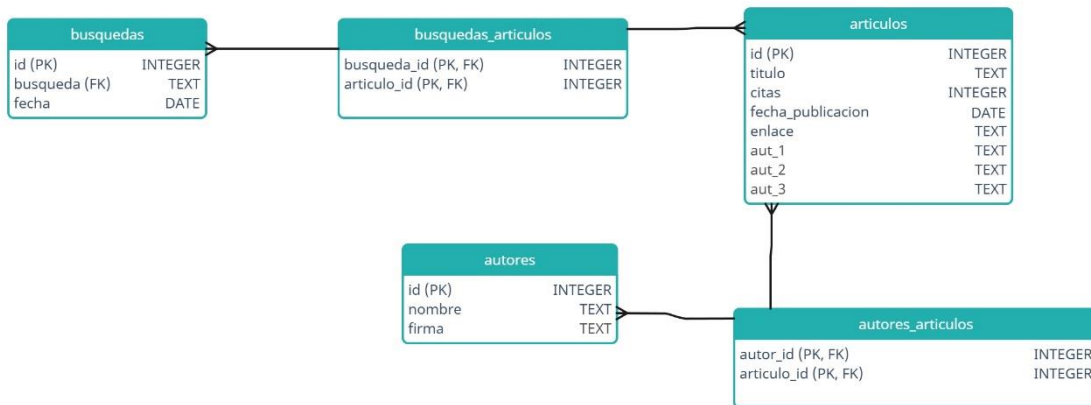
Devuelve el mismo JSON que se crea al final de la búsqueda en `scholar.py` con los datos de los artículos encontrados durante el proceso de recolección de datos.

- **Endpoint `"/api/busquedas-anteriores"`:** Se Invoca al metodo `"buscarbd()"` que consulta en la base de datos todas búsquedas almacenadas en la tabla de búsquedas y las envía al Frontend.
- **Endpoint `"/api/busqueda-bd"`:** Invoca el método `"buscarPorId()"` (Figura 5) que consulta la base de datos para obtener los artículos asociados a una determinada búsqueda con ayuda de un identificador que se le pasa desde el Frontend.
- **Endpoint `"/api/almacenarBusqueda"`:** Expone el método `"almacenarBusqueda()"` que recibe la información de la búsqueda realizada en el Frontend y la almacena en las tablas correspondientes, creando las relaciones entre la búsqueda, los artículos encontrados y los autores de dichos artículos.

- **Endpoint `"/api/borrarBusqueda"`:** Invoca el método `"borrarBusqueda()"`, que usa el identificador de la búsqueda que se desea borrar recibido desde el Frontend para eliminar de la base de datos dicha búsqueda junto con las relaciones que tuviera con los artículos almacenados.
- **Endpoint `"/api/test"`:** Este endpoint se usó como prueba para verificar la comunicación entre el FrontEnd y el Backend. Forma parte de las pruebas descritas en el capítulo 6.

## Base de datos SQLite

La base de datos está diseñada para gestionar información sobre autores, artículos académicos, búsquedas realizadas y las relaciones entre estas entidades. Está compuesta por cinco tablas: `autores`, `articulos`, `busquedas`, `busquedas_articulos` y `autores_articulos`, cuyo modelo Entidad-Relación se representa en la figura 6.



**Figura 6:** Diagrama entidad/relación de la base de datos de SQLite

- La tabla `autores_articulos` permite gestionar la relación de muchos a muchos entre autores y artículos, indicando qué autores han contribuido a qué artículos.
- La tabla `busquedas_articulos` permite gestionar la relación de muchos a muchos entre búsquedas y artículos, mostrando qué artículos están asociados con cada búsqueda.
- La tabla `búsquedas` almacena la información relativa a la búsqueda que ha realizado el usuario, almacenando la cadena de caracteres que se ha usado para obtener el resultado de la búsqueda y la fecha en la que se ha realizado. En caso de que se repita exactamente la misma búsqueda, la fecha queda actualizada a la última vez que se realizó.
- La tabla de `autores` almacena información relacionada con el autor. Actualmente no se recoge más información que el nombre, dejando el campo de firma para posibles ampliaciones del proyecto.

- La tabla de artículos almacena la información recopilada sobre los artículos de investigación encontrados en la búsqueda. Se recoge el título del artículo, el número de citas según Google Scholar, la fecha de publicación, el enlace al documento. además, en aut\_1 y aut\_2 se almacena el primer y segundo autor de la publicación, respectivamente, y en aut\_3 se almacena el resto de los autores de dicha publicación. Esto se debe a que el número de autores puede ser bastante amplio y no se desea perder esta información.

Las relaciones entre las entidades son las siguientes:

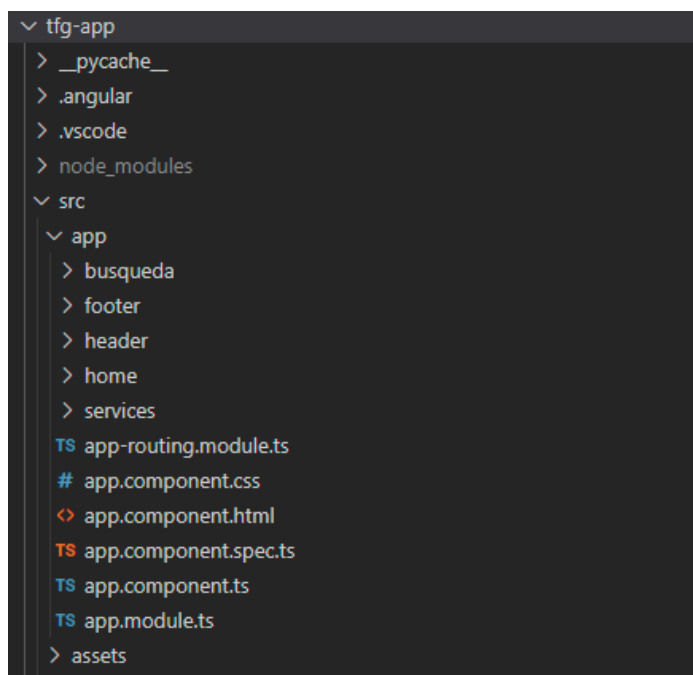
- Un autor puede estar relacionado con múltiples artículos (relación de muchos a muchos representada por la tabla autores\_articulos).
- Un artículo puede estar relacionado con múltiples búsquedas (relación de muchos a muchos representada por la tabla busquedas\_articulos).
- Una búsqueda puede estar relacionada con múltiples artículos (relación de muchos a muchos representada por la tabla busquedas\_articulos).

Para la creación de la base de datos, se ha usado un script de Python que crea todas las tablas que se ha nombrado "create\_bd.py". además de crear las tablas, inserta algunos datos de prueba en la base de datos y realiza algunas consultas para asegurar su correcta creación.

# 6

## Desarrollo del Frontend

El desarrollo del frontend se ha llevado a cabo utilizando Angular 16, conforme se ha detallado en el capítulo 1. Se ha buscado adoptar un enfoque altamente modular con el objetivo de mejorar la comprensión de los diversos componentes del proyecto y permitir futuras expansiones de manera más fluida. A continuación, se proporciona una descripción más detallada de su estructura.



**Figura 7:** Estructura de los archivos que componen el proyecto de Angular.

Dentro del directorio "src" se localizan todos los archivos esenciales para la ejecución del frontend de Angular. En este lugar, se destacan varios archivos significativos:

- El archivo `app-routing.module.ts` administra el enrutamiento entre las distintas páginas de la aplicación web.
- El archivo `app.component.html` constituye la página principal en la que se cargan las demás páginas de la aplicación, y contiene las rutas hacia estas.
- `App.module.ts` es el archivo donde se importan todas las bibliotecas y elementos utilizados en la aplicación web.

En el directorio "app", se han establecido los distintos componentes que conforman la página web. Como se puede apreciar en la figura 7, se definen los componentes de búsqueda, pie de página, encabezado e inicio.

- Los componentes de encabezado y pie de página corresponden, respectivamente, a la cabecera (donde se encuentran los enlaces para la navegación entre las distintas páginas del sitio web) y el pie de página que se sitúa al final de la página.
- El componente de inicio se encarga de mostrar la primera página que visualiza el usuario. En ella, se ha elaborado una breve guía del usuario para facilitar la navegación y el uso de la aplicación a nuevos usuarios.
- El componente de búsqueda es fundamental, ya que carga la página en la que el usuario puede realizar búsquedas de los artículos deseados, y desde la cual puede guardar o cargar información en la base de datos.

Cada componente está compuesto por varios archivos: un archivo HTML que define la estructura de la página web, un archivo CSS que proporciona estilos al código HTML, y dos archivos TypeScript en los que se definen diversas funcionalidades de dicho componente. En la figura 8 se ilustra cómo se presenta el contenido del componente de búsqueda.

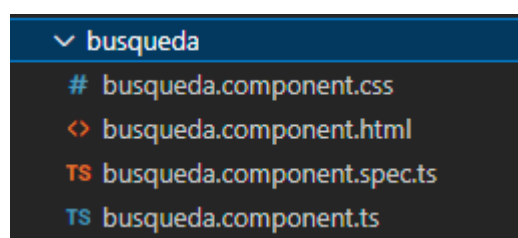


Figura 8: Contenido del componente "busqueda".

Además, en esta misma ubicación, se encuentra la carpeta "services", la cual alberga un archivo dedicado a la comunicación entre el backend y el frontend. Dentro de este archivo, se definen las rutas que la aplicación frontend utilizará para enviar o solicitar datos al backend. Asimismo, se especifican los métodos que emplean estas rutas. A continuación, en la figura 9, se muestra un fragmento del código contenido en este archivo.

```
export class AppService{

  constructor(private http: HttpClient){ }

  urlBuscarOnline: string = 'http://127.0.0.1:5000/api/buscar-online';
  urlBuscarBD: string = 'http://127.0.0.1:5000/api/búsquedas-anteriores'
  urlBusquedaArticulos: string = 'http://127.0.0.1:5000/api/busqueda-bd'
  urlAlmacenarBusqueda: string = 'http://127.0.0.1:5000/api/almacenarBusqueda'
  urlDescargar: string = 'http://127.0.0.1:5000/api/descargarCSV'
  urlBorrarBusqueda: string = 'http://127.0.0.1:5000/api/borrarBusqueda'
  urltest: string = 'http://127.0.0.1:5000/api/test'

  getBusquedaOnline(busqueda: string, idioma: string, paginas: number): Observable<any>{
    let params = new HttpParams()
    .set('idioma', idioma)
    .set('busqueda', busqueda)
    .set('paginas', paginas);
    return this.http.get(this.urlBuscarOnline, {params})
  }

  getBúsquedasAnteriores(): Observable<any>{
    let params = new HttpParams()
    return this.http.get(this.urlBuscarBD, {params})
  }

  getBúsquedasArticulos(id:number): Observable<any>{
    let params = new HttpParams()
    .set('idBusqueda', id);
    return this.http.get(this.urlBusquedaArticulos, {params})
  }

  postGuardarBusqueda(busqueda: string, articulos: Articulo[]): Observable<any>{
    return this.http.post(this.urlAlmacenarBusqueda, {busqueda, articulos})
  }

  postBorrarBusqueda(id:number){
    return this.http.post(this.urlBorrarBusqueda, {id})
  }
}
```

**Figura 9:** Fragmento del Código del archivo "app.services.ts", encargado de la comunicación entre el frontend y el backend.



# Integración y Pruebas

## Estructura del proyecto

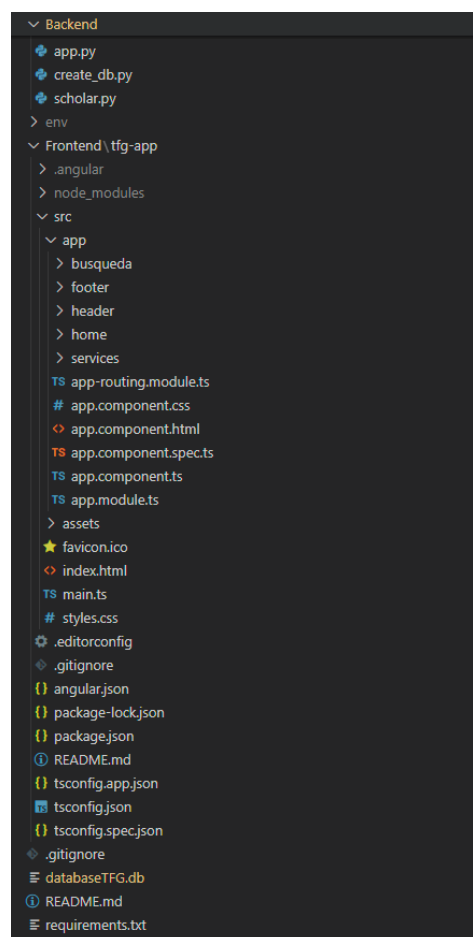
En el código fuente del proyecto, la carpeta "TFG" contiene todos los archivos desarrollados durante este proyecto, tal y como se observa en la figura 8, divididos en dos carpetas llamadas "Frontend" y "Backend". Dentro de la primera se encuentran todos los archivos del proyecto de Angular, y la segunda contiene un fichero llamado "app.py", el archivo "create\_db.py" usado para crear la base de datos y la propia base de datos "databaseTFG.db" y, por último, el fichero "scholar.py" que es el que recoge los datos de la página web de scholar.

Todas las partes se coordinan gracias al backend "app.py". Desde él, se obtiene o guarda información en la base de datos, se reciben peticiones o se manda información hacia el frontend o ejecuta el *scraping* de la web de Scholar.

## Pruebas sobre el script de *scraping*

El proyecto empezó con el desarrollo del script encargado de recoger los datos de la página web de Google Scholar, y, por lo tanto, fue también el primero en necesitar pruebas para verificar su correcto funcionamiento.

El desarrollo de las pruebas fue incremental según se iba escribiendo funcionalidades nuevas del código. Por ejemplo, se programaba la parte que extraía los datos en crudo, y se visualizaban para verificar que se estaban recibiendo datos. Posteriormente, se



**Figura 10:** Estructura de los archivos de la aplicación web.



```
# creamos los dataframes y guardamos los datos en csv (se han comentado las líneas que guardan en un archivo estas búsquedas)
df_autores = pd.DataFrame({"Nombre": listaAutores, "firma": ""}) # no se usa actualmente
#df_autores.to_csv("Lista de autores", index=False)
df_articulos = pd.DataFrame({"Titulo": titulos, 'Citas': citas, 'Año': anoPublicacion, 'Enlace': link, 'Autor 1': autores1, 'Autor 2': autores2, 'Autor 3': autores3, })
#df_articulos.to_csv("Lista de articulos", index=False)
```

Las últimas pruebas implicaban la creación de unos ficheros donde se almacenaban los datos resultantes de la ejecución de la búsqueda, tal y como se muestra en la figura 11. Esto fue de mucha ayuda a la hora de afinar los datos que se estaban recopilando y para ver posibles fallos en el formato. En la versión final del código, estas líneas se encuentran comentadas.

En el mismo fichero usado para crear la base de datos, se definieron algunas consultas que comprobaban que efectivamente los datos se estuvieran añadiendo y eliminando correctamente de esta base de datos.

```
#Creamos unos datos de prueba
c.execute("INSERT INTO articulos VALUES (1, 'La valoracion de la prueba', 490, 2018, 'http://www.derechopenalenlared.com/libros/la-valoracion-de-la-prueba-jordi-nieva.pdf',
c.execute("INSERT INTO articulos VALUES (2, 'Eportafolios en Procesos Blendedlearning Innovaciones de la Evaluaci n en los Crditos Europeos', 56, 2009, 'https://revistas.um.es/eportafolios')")
c.execute("INSERT INTO busquedas VALUES (1, 'prueba', '25-03-2024')")
c.execute("INSERT INTO busquedas_articulos VALUES (1,1)")
c.execute("INSERT INTO busquedas_articulos VALUES (1,2)")

c.execute('SELECT * FROM articulos')
articulos = c.fetchall() #puede ser fetchone() o fetchmany(x)
#print(articulos)

c.execute('SELECT * FROM busquedas')
busquedas = c.fetchall() #puede ser fetchone() o fetchmany(x)
#print(busquedas)

c.execute('SELECT * FROM busquedas_articulos')
busquedasArticulos = c.fetchall()
#print(busquedasArticulos)

c.execute("""
        SELECT a.* FROM articulos a
        INNER JOIN busquedas_articulos ba ON a.id = ba.articulo_id
        INNER JOIN busquedas b ON b.id = ba.busqueda_id
        WHERE b.id = '1'; """)

rdo = c.fetchall()
#print(rdo)
```

Observamos en la figura 12 cómo se introducen datos en las tablas de artículos y búsquedas y se crean las relaciones en la tabla de "busquedas\_articulos". Posteriormente se hace una consulta sobre cada tabla de forma independiente y por último una consulta algo más compleja, que realiza una búsqueda de los artículos relacionados con la búsqueda con id igual a 1.

En el caso las pruebas sobre el Frontend fueron más sencillas, pues en su mayor parte solo se debía que los elementos HTML se mostraban correctamente en la página web o que los enlaces hacia otras páginas funcionaban.

Para comprobar que los botones y búsquedas funcionaban antes de que estuviera el endpoint que recibe los datos a buscar en el backend, se usó la consola del navegador que se despliega al inspeccionar un elemento. En esta consola se podían imprimir datos definidos por el desarrollador. Por ejemplo, al rellenar los campos de búsqueda y hacer clic sobre el botón "Buscar", se podía observar en la consola los campos que se le pasarían más tarde al backend, verificando que la página web los recoge correctamente.

En las pruebas del backend, se definieron dos metodos; Uno que simplemente ejecutaba el script "scholar.py" y lo mostraba en la terminal, y otro llamado "saludar()". Este último era un endpoint que recibe los parámetros de idioma, búsqueda y número de páginas, los muestra en la terminal y devuelve un archivo JSON con un mensaje para verificar en el frontend si la comunicación se completaba con éxito. El código de estos métodos se observa en las figuras 13 y 14.

```
# EndPoint de prueba para comprobar que el Backend y el Frontend se pueden comunicar.
@app.route('/api/test', methods=['GET'])
def saludar():
    idioma = request.args.get('idioma')
    busqueda = request.args.get('busqueda')
    paginas = request.args.get('paginas')
    print(busqueda, idioma, paginas)
    data = {"mensaje": "Hola desde Flask!", "busqueda": busqueda}
    return jsonify(data)
```

Figura 13: Endpoint "saludar()" en app.py para probar la conexión entre el backend y el frontend.

```
getTest(busqueda: string, idioma: string, paginas: number): Observable<any>{
    let params = new HttpParams()
        .set('idioma', idioma)
        .set('busqueda', busqueda)
        .set('paginas', paginas);
    return this.http.get(this.urlTest, {params})
}
```

Figura 14: Fragmento del código en "app.service.ts" que hace una petición al endpoint "test" para probar la conexión entre el backend y el frontend.

# 8

## Conclusiones y líneas futuras

La aplicación desarrollada facilita la tarea del investigador de encontrar artículos científicos en la web de Google Scholar, asentando las bases de una herramienta que ahorra mucho tiempo en búsquedas por internet.

Como se describe en la guía de usuario de la propia aplicación web, en el apartado de "inicio", el usuario puede introducir los parámetros de búsqueda y, al hacer clic sobre "Buscar en Google Scholar" y, como resultado, se muestran los datos en una tabla. Los elementos de esta tabla se pueden ordenar para facilitar al usuario encontrar el nombre del artículo, o buscar solo aquellas publicaciones con más citas.

Si el usuario encuentra útil esta información, puede guardarla en la base de datos para no tener que volver a realizar la búsqueda, evitando así posibles bloqueos por parte de la página web consultada, que son bastante comunes al usar bots de recolección de datos.

En otra tabla se muestran búsquedas que el usuario ha realizado anteriormente, u con un par de botones puede decidir si volver a mostrar los resultados o borrarla si ya no le son útiles.

Entre las mayores dificultades enfrentadas, se encuentran:

- La recolección de datos de páginas web que no poseen una API. Los resultados de la búsqueda se deben filtrar por etiquetas que, en muchas ocasiones, tienen nombres poco descriptivos. también ocurre que no se sigue una estructura de la información en todos los casos, pudiéndose mostrar primero la fecha de un artículo y después el autor, o no existir fecha, etc. Esto dificulta el acceso a dicha información.

- Otro problema relacionado con la extracción de datos es la posibilidad de que la página web bloquee el acceso del ordenador que ejecuta la búsqueda, detectándola como un bot de recolección de datos. Este problema puede solucionarse con el uso de BeautifulSoup y las cabeceras que se pueden añadir a la búsqueda, simulando ser un usuario real. También se optó por congelar la ejecución del script entre página y página consultada por unos segundos, simulando la velocidad a la que una persona real navega por la web.
- Estimación del tiempo empleado en cada tarea: Al tratarse de un proyecto nuevo para el alumno, muchas de las tareas que se planteaban con una duración concreta, ha sido subestimada, resultando en más horas de trabajo. Tareas que parecen sencillas han resultado ser más complicada que en un principio y ha acabado retrasando la finalización del proyecto. Esta experiencia, sin embargo, resulta de vital importancia de cara al desempeño profesional del alumno de cara a futuros proyectos.

Como posibles continuaciones de este proyecto, se proponen las siguientes funcionalidades que mejorarían la experiencia y usabilidad del sistema:

- En primer lugar, la nueva versión de Angular 17 trae consigo muchas novedades, como la carga más rápida de los módulos, su nueva documentación más completa o los nuevos componentes independientes. Sería interesante actualizar el Frontend de su versión de Angular 16 a esta última versión.
- Incluir un control de acceso o *login* de cara al despliegue de la herramienta en un entorno público y real, donde se controle qué usuarios pueden almacenar o borrar datos de la base de datos.
- El problema con los nombres de los autores es que Scholar no proporciona en muchas ocasiones nombres completos, sino alias o abreviaciones que dificultan relacionar los artículos con sus autores, habiendo casos donde un mismo autor se presenta con varios alias diferentes. Podría implementarse alguna normalización de estos datos para poder enlazar correctamente cada autor con su artículo y así poder filtrar por nombres de autor.
- Por último, se puede enriquecer la información mostrada al usuario consultando varias páginas web como, por ejemplo, las analizadas en el capítulo 1. De esta forma, se puede evitar que se consulten varias páginas web y el usuario deba filtrar la información que cree que puede serle de utilidad.



# Referencias

1. Clarivate. (2024). Data, insights and analytics for the innovation lifecycle. Clarivate. <https://clarivate.com/> el 28 de mayo de 2024 (pág 2)
2. ORCID. (2024). Connecting Research and Researchers. ORCID. <https://orcid.org/> el 28 de mayo de 2024 (pág 2)
3. DBLP. (2024). DBLP: Computer Science Bibliography. DBLP. <https://dblp.org/> el 28 de mayo de 2024 (pág 3)
4. Google Scholar. (2024). Google Scholar. <https://scholar.google.es/> el 28 de mayo de 2024 (pág 3)
5. pandas development team. (2024). *pandas: Python Data Analysis Library*. <https://pandas.pydata.org/> el 28 de mayo de 2024. (pág 3)
6. Richardson, L. (2024). *beautifulsoup4* (Versión 4.12.3) [Software]. PyPI. <https://pypi.org/project/beautifulsoup4/> el 28 de mayo de 2024. (pág 3)
7. Python Software Foundation. (2022). Requests 2.32.2. Obtenido de <https://pypi.org/project/requests/> el 28 de mayo de 2024 (pág 3).
8. click. (2023, 17 agosto). PyPI. <https://pypi.org/project/click/> <https://pypi.org/project/click/> el 28 de mayo de 2024. (pág 4)
9. SQLite Consortium. (2024). *SQLite*. <https://sqlite.org/> el 28 de mayo de 2024. (pág 4)
10. Welcome to Flask — Flask Documentation (3.0.x). (s. f.). <https://flask.palletsprojects.com/en/3.0.x/> el 28 de mayo de 2024. (pág 4)
11. Bootstrap. (2024). *Bootstrap* (Versión 5.3.3) [Documentación]. <https://getbootstrap.com/> el 28 de mayo de 2024. (pág 4)
12. Team, A. C. (s. f.). Angular Material. <https://material.angular.io/> el 28 de mayo de 2024. (pág 4)
13. Tailwind CSS - Rapidly build modern websites without ever leaving your HTML. (s. f.). Tailwind CSS. <https://tailwindcss.com/> el 28 de mayo de 2024. (pág 4)

14. git Guides. (2024). GitHub. <https://github.com/git-guides> el 28 de mayo de 2024. (pág 11)
15. Welcome to Python.org. (2024, 28 mayo). Python.org. <https://www.python.org/> el 28 de mayo de 2024. (pág 12)
16. Angular. (s. f.). <https://docs.angular.lat/tutorial> el 28 de mayo de 2024. (pág 12)
17. Flexbox Froggy. (s. f.). A Game For Learning CSS Flexbox. <https://flexboxfroggy.com/#es> el 28 de mayo de 2024. (pág 12)

# Listado de figuras

**Figura 1:** Casos de uso. 8

**Figura 2:** Diagrama de flujo del proceso más común. 9

**Figura 3:** Fragmento del código de `scholar.py`, en el método `buscar()` donde se modifica la dirección URL para buscar en Scholar.. 14

**Figura 4:** Fragmento del código de `scholar.py` donde se usan etiquetas y atributos para filtrar el contenido de la página web. 14

**Figura 5:** Ejemplo de un Endpoint de `app.py` con los comentarios para facilitar su comprensión. 16

**Figura 6:** Diagrama entidad/relación de la base de datos de SQLite. 17

**Figura 7:** Estructura de los archivos que componen el proyecto de Angular. 19

**Figura 8:** Contenido del componente "búsqueda". 20

**Figura 9:** Fragmento del Código del archivo "`app.services.ts`", encargado de la comunicación entre el frontend y el backend. 17

**Figura 10:** Estructura de los archivos de la aplicación web. 19

**Figura 11:** Fragmento del código en el que se crean 2 ficheros de salida donde se escriben los datos obtenidos del scraping. 21

**Figura 12:** Consultas realizadas sobre la base de datos justo después de su creación para verificar su funcionamiento. 24

**Figura 13:** *Endpoint "saludar()"* en `app.py` para probar la conexión entre el backend y el frontend. 25

**Figura 14:** Fragmento del código en "`app.service.ts`" que hace una petición al endpoint "test" para probar la conexión entre el backend y el frontend. 25

**Figura 15:** Cabecera de la aplicación web. 35

**Figura 16:** Sección donde el usuario introduce los parámetros de búsqueda. 35

**Figura 17:** Tabla de resultados. 36

**Figura 18:** Tabla de búsquedas anteriores. 36





# Apéndice A

## Manual de Instalación

### Instalación y ejecución en Windows:

#### Configuración del Backend

1. Instale Python: Asegúrese de tener Python instalado en su sistema. Puede descargarlo desde [python.org](https://python.org).
2. Clone el repositorio: Abra una terminal y ejecute el siguiente comando para clonar el repositorio del proyecto:

```
git clone https://github.com/EnriqueIp/TFG.git
```

3. Acceda al directorio del proyecto y cree un entorno virtual:

```
cd TFG  
python -m venv env
```

4. Active el entorno virtual e instale las dependencias:

```
.\env\Scripts\activate  
pip install -r requirements.txt
```

5. Inicialice la base de datos

```
python .\Backend\create_db.py
```

6. Ejecute el backed

```
python .\Backend\app.py
```

## Configuración del Frontend

1. Instale Node.js: Descargue e instale Node.js versión 18 LTS desde [nodejs.org](https://nodejs.org).
2. Instale de NPM: Abra una terminal y ejecute el siguiente comando para asegurarse de que NPM está instalado:

```
npm install
```

3. Instale de Angular: En la terminal, ejecute el siguiente comando:

```
npm install -g @angular/cli@16
```

4. Navegue al directorio del proyecto angular y ejecute el Frontend:

```
cd .\TFG\Frontend\tfg-app\  
ng serve -o
```

## Creación de la base de datos

La base de datos se descarga junto con el resto de los archivos del repositorio de GitHub, pero si se borrara o no estuviera, se volvería a crear ejecutando el siguiente comando en la carpeta del proyecto:

```
python create_db.py
```

# Apéndice B

## Manual de usuario

La primera página que se visualiza de esta aplicación web es la pagina de inicio en la que se puede consultar el manual de usuario. En la parte superior se encuentra la cabecera con dos botones para navegar por la aplicación, tal y como se aprecia en la figura 15. El segundo boton lleva hacia la pagina donde se hacen las busquedas.

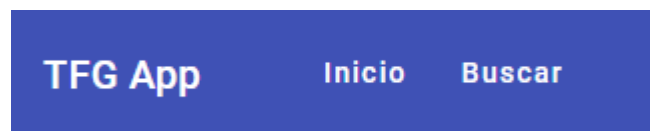


Figura 15: Cabecera de la aplicación web.

La página de búsqueda contiene varios elementos importantes que se proceden a explicar a continuación:

- El apartado para introducir los datos de búsqueda consta de un cuadro de texto y dos desplegables. En el primero de ellos se introduce la cadena de texto a buscar. En el segundo recuadro, que es el primer desplegable, muestra las opciones de idioma que pueden seleccionarse para realizar la búsqueda, y en el último recuadro, el segundo desplegable, permite al usuario escoger el número de páginas que va a consultar. En la figura 16 se observa el aspecto cuando el usuario aún no ha introducido los datos.

### Introduce los datos para buscar

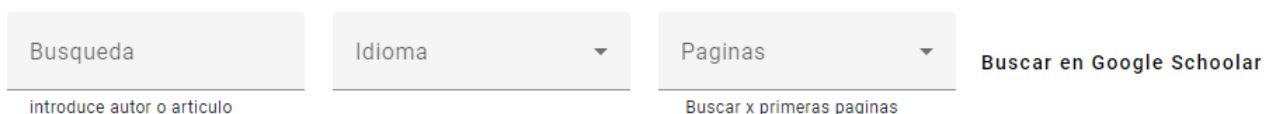




Figura 16: Sección donde el usuario introduce los parámetros de búsqueda.

- Justo debajo se encuentra la tabla donde se mostrarán los resultados encontrados. Las columnas mostradas son título, citas, autor 1, 2 resto de autores, año de publicación y enlace. En esta última columna, haciendo clic en el símbolo, se abre una nueva ventana del navegador mostrando el artículo para facilitar al usuario su lectura. Además, todas estas columnas se pueden ordenar haciendo clic sobre cada una de ellas.

Se observa un botón de color verde llamado "Guardar resultados" cuya finalidad es la de permitir al usuario guardar la búsqueda en la base de datos, si es que le

ha resultado de utilidad, para poder consultarla en otro momento sin necesidad de ejecutar una nueva búsqueda. En la figura 17 se muestra el resultado de una búsqueda de prueba, con la que se verifica su correcto funcionamiento.

Guardar resultados

Titulo	Citas	Autor 1	Autor 2	Resto de autores	Año de publicación	Enlace
La valoracion de la prueba	490	J Nieva Fenoll			2018	
Eportafolios en Procesos Blendedlearning Innovaciones de la Evaluacin en los Crditos Europeos	56	R Barragán	R García	O Buzón	2009	

Items per page: 5 0 of 0 |< < > >|

**Figura 17:** Tabla de resultados.

- Bajo la tabla de búsquedas antes mencionada se encuentra un desplegable que contiene las búsquedas realizadas anteriormente y que se encuentran en la base de datos. Posee 3 columnas: la primera corresponde con la cadena de texto que se introdujo. La segunda muestra la fecha en la que se guardó o actualizó dicha búsqueda, y la última tiene dos iconos.

El icono azul tiene la función de cargar en la tabla de búsquedas todos los resultados almacenados desde la base de datos. El icono rojo de la papelera permite borrar esta entrada de la base de datos si el usuario considera que ya no le es útil.

Esta tabla también cuenta con un filtro, útil cuando se poseen gran cantidad de búsquedas guardadas y se quiere encontrar más fácilmente la que el usuario desea.

La figura 18 muestra dos búsquedas ya realizadas por el usuario.

Busquedas anteriores Consulta las busquedas guardadas en la base de datos

Filtro

Busqueda	Fecha	
prueba	25-03-2024	 
Rafael Garcia	12-06-2024	 

**Figura 18:** Tabla de búsquedas anteriores.



UNIVERSIDAD  
DE MÁLAGA

| **uma.es**

E.T.S. DE INGENIERÍA INFORMÁTICA

E.T.S de Ingeniería Informática  
Bulevar Louis Pasteur, 35  
Campus de Teatinos  
29071 Málaga