Departamento de tecnoloxías da información
Facultade de Informática
UNIVERSIDADE DA CORUÑA

TRABAJO FIN DE GRADO
GRADO EN INGENIERÍA INFORMÁTICA
MENCIÓN EN TECNOLOGÍAS DE LA INFORMACIÓN

Desarrollo de una aplicación para el análisis automatizado de datos de producción científica

Autor: Marco Rodríguez Ocampo
Directores: Basilio Bernardo Fraguela Rodríguez
Diego Andrade Canosa

A Coruña, 18 de noviembre de 2019

A todos aquellos interesados en conocer los datos relacionados con los grandes autores en el mundo de la ciencia

Agradecementos

Quisiera dedicar este proyecto a mi familia, que me ha apoyado, siendo un pilar fundamental durante todos estos años, sobre todo para acabar este trabajo que resultó ser el reto más grande del grado. También a mis amigos de siempre, los cuales me animaron para acabar este proyecto, y a los amigos que encontré en la facultad, de los que aprendí mucho y con los que compartí buenos momentos. Por último, agradecer a aquellos profesores que despertaron mi interés en campos de la informática que nunca esperaría.

Resumen

En este proyecto se ha elaborado una aplicación Web destinada al colectivo de usuarios que desean o necesitan saber información relacionada con citas de revistas de artículos científicos. Dada la gran cantidad de datos que existen por la red y la dificultad para realizar un análisis de forma rápida, reunir la información resulta un tanto tedioso y complejo. Con esta nueva aplicación que integra datos de diversas fuentes de forma transparente al usuario, se intenta ofrecer un modo de uso sencillo y rápido de trabajar.

La idea principal es un sistema que permita al usuario buscar autores según su nombre y apellidos, para posteriormente observar sus datos y trabajar con ellos. La aplicación permitirá filtrar por diversos criterios de forma que se pueda observar los datos que se necesiten y no siempre los datos globales. También proporcionará la posibilidad de descargar los datos en formato CSV, facilitando de este modo las consultas y operaciones sobre el dato, cuando es necesario profundizar más en él de lo que una interfaz pueda permitir. Como la mayoría de las aplicaciones Web, cubrirá las típicas necesidades como son registro para información personalizada, atajos a perfiles de autores mediante el uso de favoritos, etc.

El conjunto de datos utilizados vendrá de Scopus, Web of Science, Google Scholar y GGS, consiguiendo una amplia cobertura independientemente de la información que tenga el autor buscado. Algunas de estas fuentes de datos serán accedidas mediante una API o mediante técnicas de extracción de información, y otras mediante el uso de documentos locales. La propuesta trata de conseguir que esta integración de datos sea abstracta al usuario, ofreciendo toda la información unificada.

La aplicación se implementó en un PC con prestaciones estándar, con sistema operativo Linux, y fue desarrollada con varios lenguajes de programación (Python, HMTL y JavaScript) sobre el Framework Django.

Palabras clave:

- Autor
- Scopus
- WoS
- API
- Artículo
- Django
- URL
- Integración

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Introducción

Como parte inicial de la memoria, en este capítulo se expondrá de forma lo más clara y concisa posible el propósito del trabajo realizado, de forma que el lector pueda formarse una idea básica que le ayude a conectar con el grueso del trabajo. Se dividirá en dos partes, motivación, que muestra las razones que han dado lugar al proyecto, y objetivos, donde se enumeran los objetivos fundamentales a conseguir.

1.1 Motivación

El uso páginas Web, como pueden ser la de Scopus o la de Web of Science (por ejemplo a través de la FECYT o Fundación Española para la Ciencia y la Tecnología), resulta intuitivo y de utilidad para poder consultar información relacionada con bibliografías resumen y citas científicas. A nivel usuario y con un propósito de acceso puntual, reunir información sobre un autor, sobre un artículo o sobre una revista resulta bastante sencillo. Esto cambia cuando se necesita trabajar con estos datos. Para este nuevo tipo de usuario, ya no es necesaria una interfaz agradable donde pueda observar la información global, si no que precisa un buen nivel de detalle y buenas opciones de filtración. Sobre todo, lo que no puede permitirse este usuario, es la cantidad de tiempo que hay que invertir para reunir los datos de todas las fuentes de información que necesita, traspasarlos a una hoja y desde ella poder trabajar con ellos. Esta aplicación nace por la necesidad de un colectivo de personas de recavar información de estas fuentes de datos para después poder trabajar con ellos. En concreto, se reunirán los datos pertenecientes a Scopus, una parte de Web of Science y Google Scholar. Necesitan poder jugar con los datos a demanda, sin tener que buscar cada información por separado.
En este proyecto se intenta proporcionar una base a esta necesidad, pero siendo realista en la capacidad de la aplicación, dado el tamaño del equipo de desarrollo. Para ello, se ha traducido a un conjunto de requisitos mínimos, que se pueden observar en el apartado 2 de la memoria.

1.2 Objetivos

La función principal de este sistema, como se ha introducido en el apartado anterior, es la de ofrecer un servicio Web para búsqueda y análisis de autores, citas, artículos y revistas. A partir de este punto, se elaboran unos objetivos iniciales que la aplicación debe cumplir:

- Búsqueda de autores mediante nombre y apellidos.
- Posibilidad de ver toda la información de un autor, obtenida de las diversas fuentes de datos, de forma unificada y sin tener que acceder a distintas pestañas o ventanas.
- Jugar con los datos antes de exportarlos, mediante filtraciones relacionadas, tanto con las métricas de cualificación más comunes, como con ordenaciones y/o filtros por fechas y/o texto.
- Exportar los datos a un documento CSV, donde el usuario pueda llevar a cabo funciones más complejas de obtención de información, de forma manual, o inyectando en otro Software que permita analizar estos datos.
- Almacenar, para un usuario, estos documentos CSV descargados en la aplicación, de forma que pueda recuperarlos en cualquier momento y no le suponga repetir el proceso.
- Poder guardar autores como favoritos, para acceder a ellos directamente sin necesidad de búsqueda.
- Tener los datos lo más actualizados posibles en relación al origen, pero también intentar ofrecer el mayor rendimiento a la hora de ofrecer la información recopilada.
- El usuario debería poder también añadir información, que pueda no estar vigente en alguna fuente, pero que él tenga la seguridad de que sea correcta.

Como se puede observar, la mayoría de los servicios están destinados a un usuario que esté registrado en la aplicación, de forma que pueda tener sus atajos y datos personalizados. No obstante, la propuesta intenta ofrecer el máximo conjunto de funcionalidades posibles para un usuario sin credenciales que desee consultar la aplicación de forma puntual.

1.3 Organización de la memoria

La memoria se divide en un total de 5 grandes partes o capítulos:

- **Introducción:** es el capítulo actual, realiza una descripción global de cara a que el usuario entienda la idea general del proyecto antes de profundizar más.

- **Catálogo de requisitos:** información sobre los requisitos obtenidos para la elaboración del desarrollo, vitales para la planificación y elaboración del mismo.

- **Tecnologías implicadas:** enumeración y descripción de las tecnologías empleadas en el proyecto.

- **Diseño e implementación:** parte posterior del desarrollo a los requisitos. En esta fase se elabora realmente la aplicación, de forma teórica en el diseño y de forma práctica en la implementación.

- **Pruebas:** fase final del desarrollo, donde se testea el funcionamiento de la aplicación para asegurar su correcta funcionalidad.

- **Conclusiones:** último capítulo, que cierra la memoria y que expone una visión más personal.

Al final de todo también se ofrecen dos glosarios (de acrónimos y de términos), y la bibliografía completa referente a toda la información obtenida.
Como referencia, esta memoria ha sido escrita en LaTeX en el SaaS (Software as a Service) on-line de Overleaf, a partir de la plantilla estándar para realización de Trabajos de Fin de Grado ofrecida por la Facultad de Informática.

1.4 Aplicaciones similares

Scimago es una aplicación Web que ofrece revistas e indicadores científicos de países a partir de la información que ofrece Scopus. Está muy cuidada visualmente, lo que hace que utilizar sus funcionalidades se haga muy cómodo para el usuario. Ofrece, además de datos concretos, gráficas detalladas sobre citas. Una curiosidad a mayores que tiene, es que permite exportar la información por ejemplo de una revista, en forma de Widget, lo que otorga una gran portabilidad hacia otras aplicaciones. En cuanto a la parte visual, Scimago posee mucho más potencial que el que tendrá esta aplicación, pero se ve más limitada en cuanto a funcionalidades como exportación de información en texto plano o documentos estilo CSV.
A continuación se muestran las siguientes figuras, 1.1 y 1.2, las cuales muestran un ejemplo de la interfaz de Scimago como resultado de la búsqueda de una revista por ISSN. En concreto, la primera muestra los datos globales de la misma, y la segunda una serie de gráficas (no todas) que ofrecen un pequeño análisis de datos sobre publicaciones y citas.

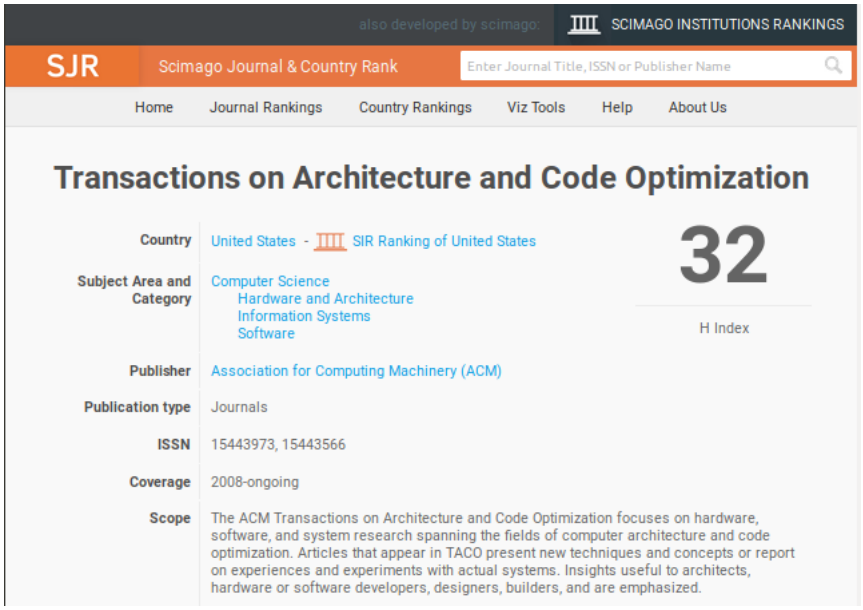

Figura 1.1: Datos de revista cuyo ISSN es 1544-3566

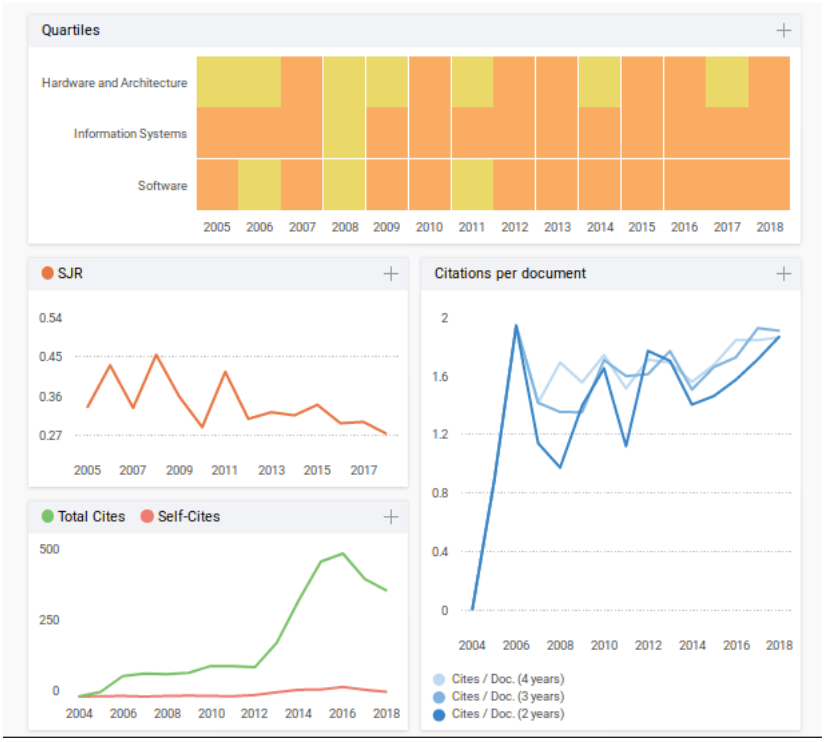

Figura 1.2: Datos gráficos de revista cuyo ISSN es 1544-3566

Capítulo 2

Catálogo de requisitos

A continuación comienza el segundo capítulo de la memoria, donde se detallan los requisitos recopilados para la elaboración de la aplicación. Hay que puntualizar antes de nada, que este proyecto desarrolla una aplicación Web, y que eso significa que se construirá un sistema pasando por todas las capas (back-end o modelo y front-end o interfaz). Estos requisitos establecen las bases del diseño, de forma que quede claro tanto para el desarrollador como para el cliente el qué y el cómo sobre el comportamiento del software, tanto del back-end como del front-end. En la primera sección se enumeran los requisitos funcionales, que describen las funcionalidades del sistema, seguidos de la definición de los actores que partician en las funcionalidades y de los casos de uso. Posteriormente se observan los requisitos no funcionales, que tratan el cómo de cada funcionalidad. Por último, se detalla el prototipo de la aplicación de forma visual, con sus respectivos mock-ups y un diagrama de transiciones, que conecta estas pantallas o mock-ups según como se pueda viajar entre ellas.

2.1 Requisitos funcionales

Un requisito funcional es aquel que describe una función o servicio que el sistema debe ofrecer, estableciendo el comportamiento del software. Estos requisitos son definidos en las tablas que se sitúan abajo y complementados con los casos de uso del sistema, los cuales son la base para el análisis. Se identificará el requisito, su código para poder referirse a él posteriormente, su descripción y las entradas y salidas del mismo [2] [3] [4].

REQF-01	Buscar autor
Descripción	Buscar autores según nombre y apellidos
Entradas	Dos cadenas de caracteres que simbolizan el nombre y apellidos del autor/es buscado/s
Salidas	Una lista de autores que reflejan en resultado de la búsqueda, puede ser vacía, de un elemento o de varios

Cuadro 2.1: Requisito funcional (Buscar autor)

REQF-02	Ver perfil del autor
Descripción	Ver datos del perfil del autor (incluye artículos, datos globales y gráficas)
Entradas	-
Salidas	Perfil del autor con sus respectivos datos, artículos en forma de lista y gráficas

Cuadro 2.2: Requisito funcional (Ver perfil del autor)

REQF-03	Registro y login
Descripción	Mediante credenciales, acceder al sistema como usuario registrado, o registrarse en el mismo
Entradas	Dos cadenas de caracteres, nombre de usuario y contraseña del mismo (en caso de registro, una tercera cadena que simboliza la repetición de la contraseña)
Salidas	-

Cuadro 2.3: Requisito funcional (Registro y login)

REQF-04	Filtro datos del autor
Descripción	Filtración de los datos del perfil del autor, 2.2
Entradas	Una serie de filtros relacionados con el historial de publicación del autor y los artículos a filtrar
Salidas	Perfil del autor filtrado

Cuadro 2.4: Requisito funcional (Filtro datos del autor)

REQF-05	Obtención de documento resumen
Descripción	Obtener un documento resumen del perfil del autor
Entradas	Perfil del autor
Salidas	Documento de tipo CSV con los datos del perfil del autor (pueden estar filtrados)

Cuadro 2.5: Requisito funcional (Obtención de documento resumen)

REQF-06	Marcar/desmarcar autores favoritos
Descripción	Añadir o eliminar un autor (puede estar filtrado) de la lista de autores favoritos para un usuario logueado
Entradas	Autor y un booleano conforme marque o desmarque el autor de favoritos
Salidas	-

Cuadro 2.6: Requisito funcional (Marcar/desmarcar autores favoritos)

REQF-07	Posibilidad de fusión de perfiles ambiguos
Descripción	Fusionar perfiles que puedan estar separados en Scopus, pero que pertenezcan al mismo autor
Entradas	Perfiles de autores con ciertas características comunes
Salidas	Un perfil que unifica los datos del resto

Cuadro 2.7: Requisito funcional (Posibilidad de fusión de perfiles ambiguos)

2.1.1 Definición de los actores

A continuación se describen los actores del sistema, siendo un actor un rol jugado bien por un ser humano, o bien por otra entidad o sujeto. En este caso los actores son solo tres y son sencillos, ya que siempre es una persona la que desempeña los correspondientes roles [5].

ACT-01	Administrador
Descripción	Representa a un encargado del mantenimiento de la aplicación, que se encarga de mantener las fuentes de datos actualizados

Cuadro 2.8: Definición de actores (Administrador)

ACT-02	Usuario no autenticado
Descripción	Representa a un usuario que no se ha autenticado en el sistema, por lo cual no goza de plenas funcionalidades

Cuadro 2.9: Definición de actores (Usuario no autenticado)

ACT-03	Usuario autenticado
Descripción	Representa a un usuario que se ha autenticado en el sistema, por lo cual sí goza de plenas funcionalidades

Cuadro 2.10: Definición de actores (Usuario autenticado)

2.1.2 Casos de uso del sistema

Los casos de uso permiten al desarrollador especificar los requisitos del sistema (normalmente se suelen realizar previamente a la elaboración de los requisitos funcionales), con el fin de mostrar cómo será la interacción del usuario con el sistema para cada funcionalidad. Describen, de la forma más comprensiva posible para el cliente, los pasos que debe llevar a cabo para conseguir realizar cualquier tarea que la aplicación ofrezca [6].
Abajo se muestran los casos de uso para este proyecto, incluyendo los siguientes campos. Si el caso de uso no tiene pre o post condiciones, no se mostrarán en las tablas.

- Título.
- Código, que permitirá identificar al caso de uso en adelante.
- Objetivos, es decir, a qué actor va dedicado.
- Breve descripción.
- Secuencia de pasos a seguir por el actor.
- Precondiciones y postcondiciones, que son las condiciones que se deben de cumplir para llevar a cabo el caso de uso, antes (pre) y después (post) del mismo.
- Excepciones, que pueden hacer que el caso de uso finalice sin el resultado deseado.

CU-01	Buscar autores por nombre y apellidos	
Objetivos	Servicios al usuario	
Descripción	El usuario puede buscar autores en función de su nombre y apellidos, siendo este último campo obligatorio	
Secuencia normal	Paso	Acción
	1	Introducir nombre en su campo de entrada correspondiente (opcional)
	2	Introducir apellido en su campo de entrada correspondiente (obligatorio)
	3	Presionar el botón ”Buscar”
	4	El sistema muestra una lista de autores que corresponden a los campos de búsqueda y el caso de uso finaliza
Excepciones	Paso	Acción
	1	Si no se introduce el campo apellidos la búsqueda no funciona, reseteando los campos a vacío para volver a escribir

Cuadro 2.11: Caso de uso (Buscar autores por nombre y apellidos)

CU-02	Acceder al perfil de un autor en concreto	
Objetivos	Servicios al usuario	
Descripción	El usuario puede acceder a los datos de un autor en concreto, escogiéndolo desde la lista obtenida en CU-01	
Precondición	Haber buscado por nombre y apellidos	
Secuencia normal	Paso	Acción
	1	Buscar el autor que desee el usuario y pulsar ”Ver autor”
	2	Dependiendo de si el autor tiene sus datos en varios perfiles en Scopus, la aplicación puede redirigir al caso de uso CU-06
	3	El sistema muestra la información del autor y el caso de uso finaliza

Cuadro 2.12: Caso de uso (Acceder al perfil de un autor en concreto)

CU-03	Filtrar datos del autor según campos	
Objetivos	Servicios al usuario	
Descripción	El usuario puede filtrar los artículos publicados por el autor según diversos campos, pudiendo aplicar varios filtros de forma simultánea	
Precondición	Haber accedido al perfil del autor (CU-02)	
Secuencia normal	Paso	Acción
	1	El sistema ofrece un checkbox para ordenar por fecha ascendente o descendente, y varios checkboxes para filtrar: • Filtro por años (también dos campos de selección de fecha, tanto inicial(desde), como final(hasta)) • Dos filtros por texto, uno simboliza que los artículos contengan en el título la cadena introducida, y otro, que no exista en el título esa cadena • Filtro por pertenencia de la revista a la Wos (JCR) • Filtro por pertenencia de la conferencia a la GGS • Filtro por ranking del cuartil (Q1) • Filtro por ranking del cuartil (Q2) • Filtro por ranking del cuartil (Q3) • Filtro por ranking del cuartil (Q4)
	2	El usuario pulsa cualquiera de estos checkboxes y el sistema filtra automáticamente, volviendo a ver la información del autor, pero ahora ya bajo los filtros indicados
Excepciones	Paso	Acción
	1	Los filtros de pertenencia a la Wos y a GII son excluyentes, es decir, no se puede filtrar por ambos a la vez. En caso de intentarlo, el usuario recibirá un mensaje de alerta

Cuadro 2.13: Caso de uso (Filtrar datos del autor según campos)

CU-04	Registrarse	
Objetivos	Servicios al usuario	
Descripción	El usuario puede registrarse en el servidor introduciendo sus datos	
Precondición	El usuario no debe estar logueado previamente, y tampoco haberse registrado previamente (al menos no con los mismos datos que usará cuando se registre ahora)	
Secuencia normal	Paso	Acción
	1	Introducir el nombre de usuario y la contraseña (será requerida por duplicado para evitar errores) en su campo de entrada correspondiente y registrarse
Postcondición	El usuario pasa a estar registrado	
Excepciones	Paso	Acción
	3	Si no se introducen valores iguales en los campos de la contraseña, o ya existe una cuenta con el mismo nombre de usuario que el indicado, el sistema no permite el registro

Cuadro 2.14: Caso de uso (Registrarse)

CU-05	Login	
Objetivos	Servicios al usuario	
Descripción	El usuario puede loguearse en el servidor introduciendo sus credenciales	
Precondición	El usuario no debe estar logueado previamente, pero si haberse registrado	
Secuencia normal	Paso	Acción
	1	Introducir el nombre de usuario y la contraseña en su campo de entrada correspondiente y loguearse
Postcondición	El usuario pasa a estar logueado	
Excepciones	Paso	Acción
	3	Si no se introducen valores correctos en cualquiera de los dos campos, el sistema no permite el logueo

Cuadro 2.15: Caso de uso (Login)

CU-06	Fusionar perfiles de Scopus	
Objetivos	Servicios al usuario	
Descripción	El usuario puede fusionar los datos de uno o varios perfiles de Scopus si considera que pertenecen a un único autor	
Precondición	El autor buscado ha de tener más perfiles similares al suyo en la Base de Datos de Scopus	
Secuencia normal	Paso	Acción
	1	Cuando el usuario accede al perfil del autor, el sistema le redirige a una página donde enseña los posibles perfiles a fusionar con el autor actual. El usuario puede seleccionar el/los perfiles que considere para fusionar o, directamente, hacer saber al sistema que no desea tal fusión
	2	En caso de acceder a la fusión, el sistema actualiza los datos y ofrece una visión única de ambos perfiles
	3	En caso de no aceptar la fusión, el sistema muestra la información del autor con normalidad (CU-02)

Cuadro 2.16: Caso de uso (Fusionar perfiles de Scopus)

CU-07	Añadir información restante a un artículo	
Objetivos	Servicios al usuario	
Descripción	El usuario puede añadir el cuartil al que pertenece la revista o conferencia del artículo, junto con el factor de impacto de la misma, siempre y cuando estos no vengan determinados por la Wos o por la GII	
Precondición	Haber accedido al perfil del autor (CU-02)	
Secuencia normal	Paso	Acción
	1	Introducir cuartil en su campo de entrada correspondiente
	2	Introducir factor de impacto en su campo de entrada correspondiente
	3	Presionar el botón ”Añadir datos”
	4	El sistema actualiza la información del autor teniendo en cuenta los nuevos datos añadidos
Excepciones	Paso	Acción
	3	Si no se introducen valores correctos en cualquiera de los dos campos, el sistema emite un mensaje de alerta mostrando los posibles valores, para que el usuario corrija la información que va a introducir

Cuadro 2.17: Caso de uso (Añadir información restante a un artículo)

CU-08	Descargar documento resumen CSV	
Objetivos	Servicios al usuario	
Descripción	El usuario puede descargar un documento CSV con datos referentes al autor que visualiza en pantalla (CU-02), y teniendo en cuenta las filtraciones (CU-03), junto con cualquier información añadida (CU-07)	
Precondición	Haber accedido al perfil del autor (CU-02)	
Secuencia normal	Paso	Acción
	1	El usuario puede descargar el documento durante la visión del perfil del autor siempre que quiera
Postcondición	Si el usuario está logueado, el documento se guarda en la BD del servidor, junto con información de la fecha en la que se descargó y los filtros que se aplicaron	

Cuadro 2.18: Caso de uso (Descargar documento resumen CSV)

CU-09	Descargar documento resumen CSV guardado	
Objetivos	Servicios al usuario	
Descripción	El usuario puede descargar un documento CSV con datos referentes a un autor que ya ha descargado antes	
Precondición	Haber accedido al perfil del autor (CU-02), que el usuario esté logueado (CU-05) y haber descargado el documento CSV en alguna ocasión (CU-08)	
Secuencia normal	Paso	Acción
	1	El usuario puede descargar el documento durante la visión del perfil del autor siempre que quiera
Postcondición	El documento se actualiza en la BD del servidor la fecha de descarga	

Cuadro 2.19: Caso de uso (Descargar documento resumen CSV guardado)

CU-10	Marcar/desmarcar autor como favorito	
Objetivos	Servicios al usuario	
Descripción	El usuario puede marcar/desmarcar como favorito un autor habiendo o no filtrado sus datos	
Precondición	Haber accedido al perfil del autor (CU-02) y que el usuario esté logueado (CU-05)	
Secuencia normal	Paso	Acción
	1	Cuando el usuario accede al perfil del autor, este ofrece la opción de marcar (sino ha sido marcado antes) o desmarcar el mismo como favorito, lo que guardará para el usuario la URL del autor con los filtros (si los hubiese)
Postcondición	El autor quedará guardado como favorito si se marca como tal, o borrado de los favoritos si se desmarca.	

Cuadro 2.20: Caso de uso (Marcar/desmarcar autor como favorito)

CU-11	Acceder a autor favorito	
Objetivos	Servicios al usuario	
Descripción	El usuario puede acceder al perfil de un autor que haya marcado como favorito, junto con los filtros que tenía activos cuando lo marcó (CU-10)	
Precondición	El usuario debe estar logueado y en la página inicial de búsqueda	
Secuencia normal	Paso	Acción
	1	El usuario verá una lista de favoritos(que ha ido guardando anteriormente) de la cual puede escoger el que quiera para acceder a su perfil

Cuadro 2.21: Caso de uso (Acceder a autor favorito)

CU-12	Actualizar información de la WOS o de la GGS	
Objetivos	Servicios al administrador	
Descripción	El administrador puede actualizar los datos de la WOS o de GGS subiendo un archivo CSV	
Precondición	El administrador debe estar logueado y en la página inicial de administración	
Secuencia normal	Paso	Acción
	1	El administrador verá las 3 fuentes de datos presentes en la aplicación relativas a la clasificación de revistas y/o conferencias, las cuales son 2 de JCR (SCCI y SCIE) y una de GGS
	2	El administrador podrá escoger entre una de estas 3 fuentes y añadir, eliminar o actualizar el documento CSV del que se nutrirá de datos el servidor
Postcondición	Los datos se actualizarán en la BD de Django respecto al nuevo CSV	

Cuadro 2.22: Caso de uso (Actualizar información de la WOS o de la GGS)

2.2 Requisitos no funcionales

Mientras que los requisitos funcionales indican el qué de una operación o funcionalidad del sistema, estos requisitos indican el cómo, es decir, establecer para cada operación las características de la misma. La mayoría de los requisitos son comunes para cualquier sistema, variando en métricas según su magnitud [7] [8].
Abajo se simplifican en la tabla 2.23 los requisitos no funcionales de este proyecto [3] .

Requisito	Categoría	Descripción
REQNF-01	Seguridad	Permisos de acceso y modificación de fuente de datos sólo posible para administrador
REQNF-02	Seguridad	Los datos de entrada son validados, eliminando riesgos de ataques como XSS, CSRF, SQL Injection, etc
REQNF-03	Usabilidad	El tiempo de aprendizaje es escaso, la aplicación debe ser intuitiva y sencilla de utilizar
REQNF-04	Usabilidad	El sistema debe ofrecer unos mensajes de error que contengan una información clara y simplificada para el usuario
REQNF-04	Usabilidad	El sistema debe ser Responsive, es decir, que adapte su interfaz a diferentes dispositivos, ya sean móviles, tabletas o computadoras
REQNF-05	Eficiencia	La aplicación no puede ser lenta, es decir, cualquier operación debe durar el menor tiempo posible en ofrecer un resultado, o mínimo, un feedback

Cuadro 2.23: Requisitos no funcionales

2.3 Prototipo

2.3.1 Imágenes

Abajo se observan unos mockups realizados con la herramienta Balsamiq Mockups 3, que permiten hacer una descripción visual básica de la aplicación. En general, este apartado de la ingeniería software es utilizado para enseñar al cliente el aspecto que tendría el sistema que él solicita, y poder llegar así a un acuerdo. Por este motivo, el prototipo de una aplicación no tiene porqué ser exactamente igual que el resultado final, si no una forma de dejar unas bases establecidas en lo que al diseño de la interfaz se refiere [9]. Se llegó al acuerdo, para este proyecto en concreto, de primar la funcionalidad al diseño del front-end o interfaz de la aplicación, así que, como se puede observar, no va a tener una apariencia extremadamente cuidada y elegante.
A la hora de comprender todos los mockups, hay que tener en cuenta que las imágenes que tengan el usuario autenticado implican que esa funcionalidad sólo debe estar presente si ha habido un logueo previamente.

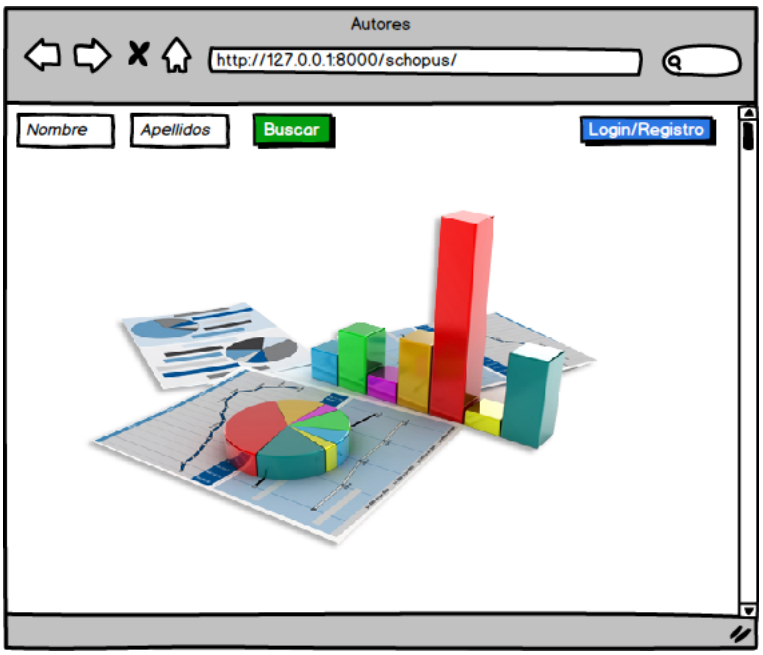

Figura 2.1: Imagen PROT1 (Vista principal)

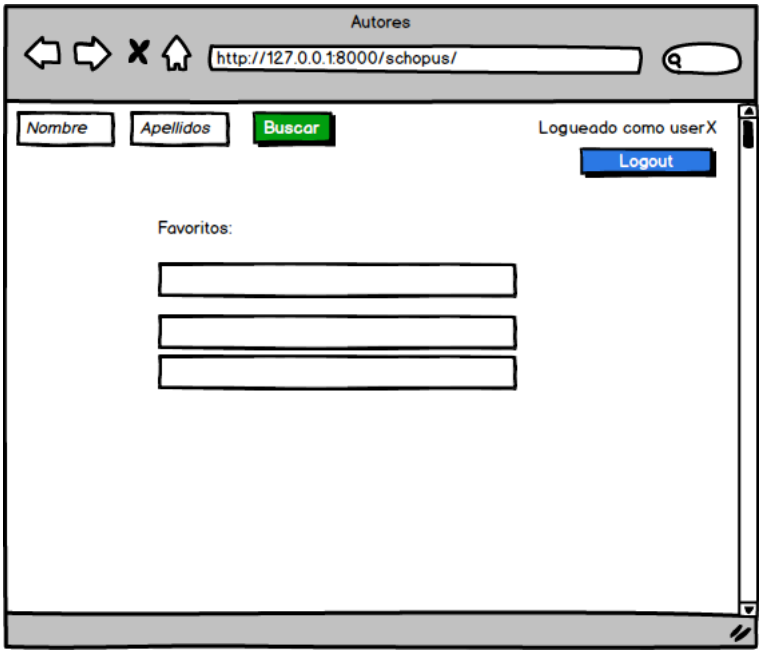

Figura 2.2: Imagen PROT2 (Vista principal con usuario logueado)

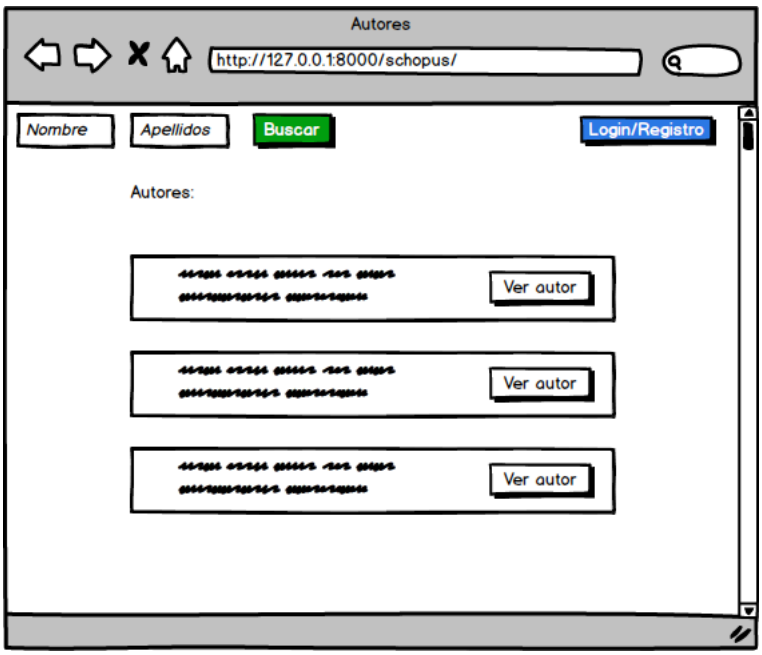

Figura 2.3: Imagen PROT3 (Vista con lista de autores según búsqueda)

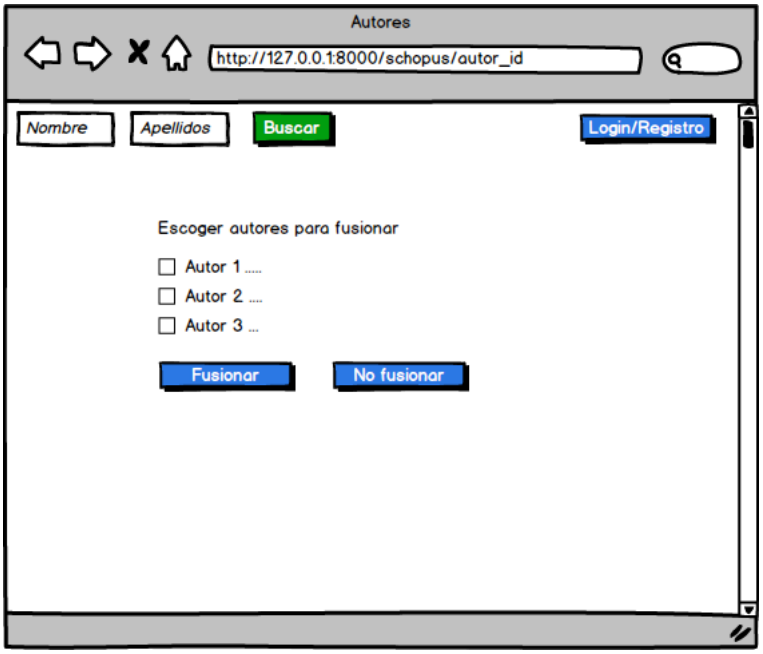

Figura 2.4: Imagen PROT4 (Vista de selección en caso de fusión)

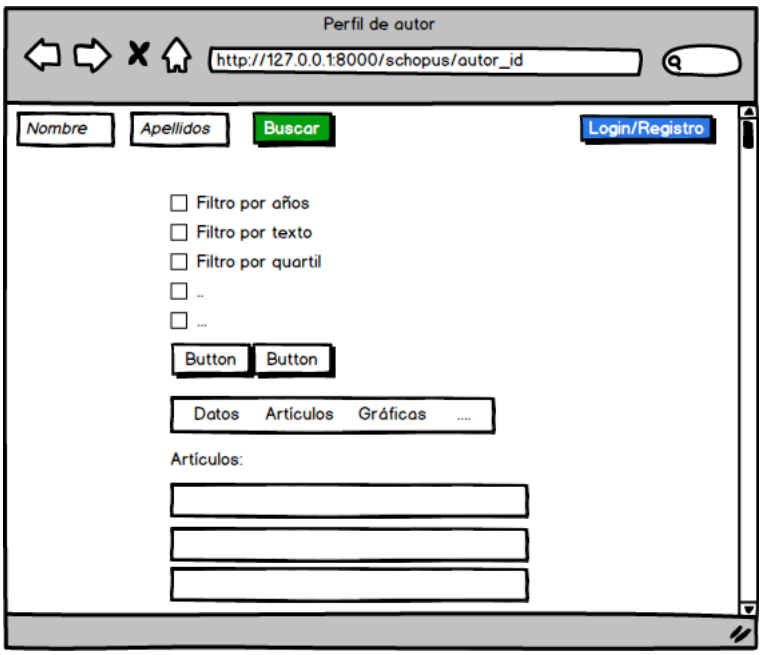

Figura 2.5: Imagen PROT5 (Primera vista principal de perfil de autor)

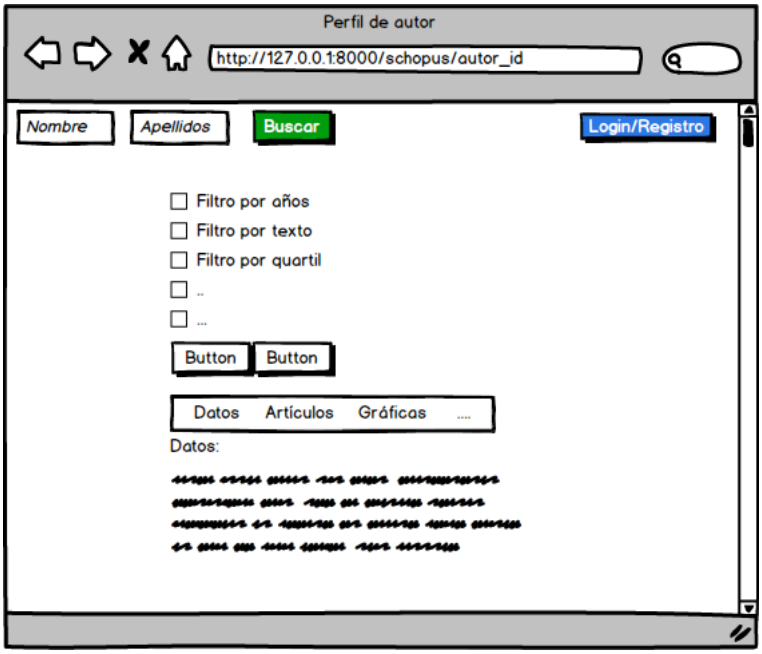

Figura 2.6: Imagen PROT6 (Segunda Vista principal de perfil de autor)

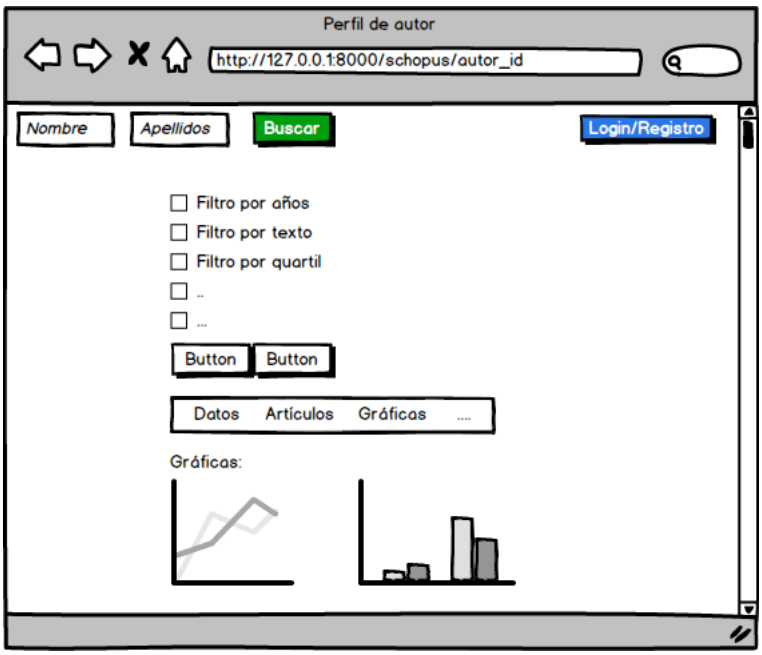

Figura 2.7: Imagen PROT7 (Tercera vista principal de perfil de autor)

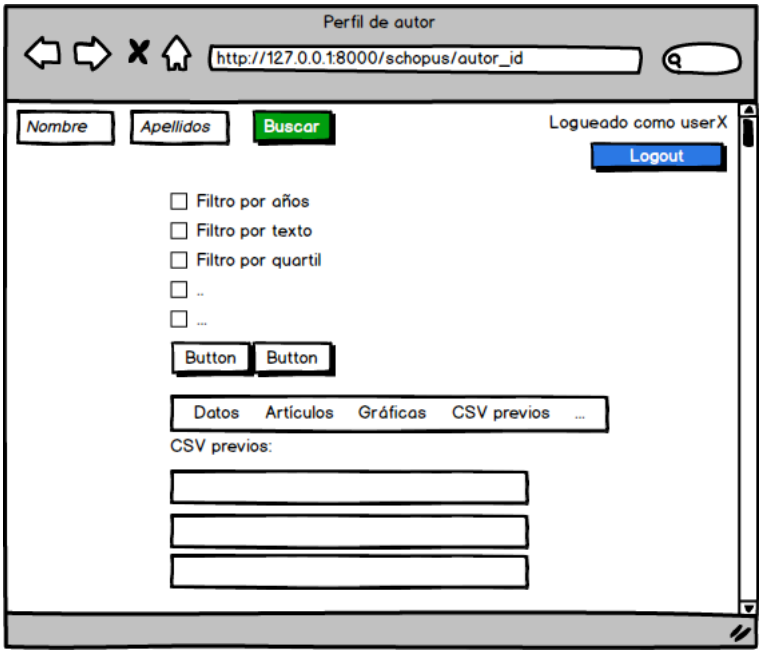

Figura 2.8: Imagen PROT8 (Cuarta vista principal de perfil de autor)

2.3.2 Diagrama de transiciones

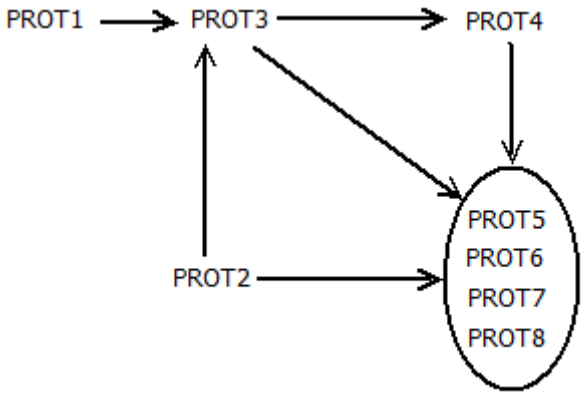

Figura 2.9: Imagen (Diagrama de transiciones entre mockups)

Capítulo 3

Tecnologías implicadas

Comienza el tercer capítulo, donde se tratarán las tecnologías que se han utilizado en este proyecto para la realización de la aplicación.
En primer lugar, se hablará de las fuentes de información y el acceso a las mismas (3.1). Estas fuentes, que como se comentó en la introducción, serán Scopus, WoS, Google Scholar y GGS, proporcionarán los datos necesarios para nutrir la aplicación.
En segundo lugar, se introducirán los lenguajes de programación utilizados para la elaboración del código que mueve la aplicación 3.1.5. Se ha construido una aplicación con 3 lenguajes, siendo Python el encargado de todo el entorno del modelo, y HTML con JavaScript los destinados al desarrollo de la interfaz.
Posteriormente aparecen los Frameworks utilizados 3.1.9, siendo Bootstrap el que complementa el HTML, y Django el que proporciona la base de la aplicación, ofreciendo el entorno de trabajo al completo.
En cuarto y último lugar están las Bases de Datos 3.2, encargadas de almacenar la información no volátil, como puede ser la gestión de usuarios, los autores favoritos o el almacenamiento de los documentos CSV descargados con anterioridad. Se implementaron SQLite y MongoDB.

3.1 Acceso a las fuentes de información

3.1.1 Scopus

Scopus es una base de datos de citas y resúmenes bibliográfico que arroja resultados de una gran variedad de campos (ciencia, tecnología, medicina, ciencias sociales y artes y humanidades) [10]. Para acceder desde la aplicación se usará la API que proporciona Elsevier, para la cual sólo se podrán realizar peticiones desde la intranet de la UDC [11]. Creando un usuario en Elsevier, se puede generar unas claves que permitan las peticiones de forma autorizada. Además de la utilización directa de la misma, usaremos el módulo de Python denominado

como ”elsapy”, en concreto la versión 0.5.0, que es la más reciente [12]. Ya que se basará la mayor parte del trabajo en esta fuente de datos, se expondrá a continuación, para cada dato que se necesite, que petición se ha de realizar, y la información que esta aporta.

- Búsqueda de un autor por nombre y apellidos
  - Objetivo: lista de autores según la búsqueda
  - Método:
    - Uso de elsapy
    - ElsSearch('authlast(apellido) and authfirst(nombre),author')
  - Datos obtenidos:
    - ID de autor de Scopus y eid
    - Nombre
    - Iniciales
    - Afiliación (nombre de la misma e ID)
    - Número de publicaciones
- Obtención de métricas del perfil de un autor
  - Objetivo: datos numéricos (métricas) que reflejan la importancia del autor
  - Método:
    - Petición por URL
    - URL= http://api.elsevier.com/content/author?author_id=autorid&view=metrics
    - Autorid es el ID del autor y metrics especifica que se obtengan ciertos datos numéricos de importancia.
  - Datos obtenidos:
    - Citas totales
    - H-Index
- Obtención de artículos de un autor
  - Objetivo: lista de artículos para un autor en concreto
  - Método:
    - Petición por URL
    - URL= http://api.elsevier.com/content/search/scopus?query=auid(autorid)&count=count

        * Autorid es el ID del autor y count especifica cuantos ítems tiene que devolver la petición (siempre indicamos los máximos que tenga el autor)
    - Datos obtenidos:
        * Identificador (DOI), título, tipo de publicación y fecha
        * Medio de publicación
            · ISSN y EISSN
            · Título y abreviatura
            · Tipo (revista, acta de conferencia, libro..)
        * Citas
        * Creador

- Obtención de coautores
    - Objetivo: lista de coautores para cada uno de los artículos
    - Método:
        * Petición por URL
        * URL= http://api.elsevier.com/content/abstract/doi/doi
        * Doi es el DOI de cada artículo del autor (es decir, se necesita una petición para cada artículo en concreto)
    - Datos obtenidos:
        * ID del autor de Scopus
        * Nombre del autor

Aunque este conjunto de datos está lejos de ser el total del contenido devuelto de cada petición, se considera que para el usuario no es preciso recabar mas información para cada autor.

3.1.2 WoS

Web of Science (WoS) es la famosa base de datos de citas y publicaciones, cuya licencia es gestionada por la FECYT a través de Clarivate Analytics (su propietario). Es utilizada para diversos fines, como el de darle una categoría a cada revista y/o a cada artículo, en base a diferentes métricas relacionadas con el índice de citas que reciben. De esta forma obtienen, según exponen en su web, ”una visión completa y segura de más de 115 años de investigación de la mejor calidad para cada ” [13][14][15].
Para este proyecto, se ha optado por usar el JCR (Journal Citation Reports), que mide el impacto de una revista según las citas que se recogen en la anteriormente introducida WoS. Hay

que aclarar que no todo lo que hay en la Web of Science está en el JCR. En concreto, de los 10 índices que hay, sólo 2 están en el JCR, y son SSCI (índice de citas de ciencias sociales) y SCIE (índice extendido de citas de ciencias) [16]. Aunque en este proyecto tiene más relevancia la búsqueda "científica", usaremos también el SSCI para aumentar el rango de publicaciones cubiertas. Gracias a estos índices, podemos asignarle un mayor o menos valor a los artículos dependiendo de la revista a la que pertenezcan.
Los indicadores que aportarán la métrica de calidad serán:

-Impact Factor o Factor de impacto: resultado de dividir el número de citas que ha recibido una revista en un año ($citas_i$) por la cantidad de artículos publicados por una revista los dos años anteriores ($articulos_j$). Visualmente, para el año x, el factor de impacto de una revista será:

$$F = \frac{\sum_{i=X-2}^{2} citas_i}{\sum_{j=X-2}^{2} articulos_j}$$

-Quartile Rank o rango de cuartil: divide una revista en 4 categorías dependiendo del rango percentil, al que llamaremos R, que ocupa en su categoría. Este valor irá determinado por indicadores como JIF Percentile o percentil de Factor de Impacto, que indica el puesto que ocupa esta revista en una categoría en un percentil) [17]. Los posibles valores son:

-Q1: 00 < R ≤ 0.25
-Q2: 0.25 < R ≤ 0.5
-Q3: 0.5 < R ≤ 0.75
-Q4: 0.75 < R

Para concluir esta introducción a la WoS y a JCR, hay que mencionar que los datos usados en este proyecto serán la recopilación de 2016 de JCR obtenidos como archivo CSV, bajo permiso por los tutores para uso exclusivo de la aplicación. De todos los datos que se pueden obtener de estos dos archivos, los que tendrán importancia para este proyecto son:

- ISSN
- Título
- Quartile Rank
- Impact Factor

3.1.3 Google Scholar (o Google Académico)

Google Scholar o Google Académico, es un buscador de Google, lanzado como beta en 2005, que se enfoca en artículos, tesis, libros, patentes, documentos relativos a congresos y resúmenes. A la hora de buscar, se puede indicar el criterio de búsqueda, por autor o por artículo (con todos los posibles datos relativos que puedan identificarlos) [18][19]. Es sencillo de utilizar y abarca mucha información, pero no puede hacer frente a los anteriormente introducidos Web Of Science y Scopus. Mediante la tabla 3.1 y 3.1, adjuntadas más abajo, se ofrece una comparativa sobre lo distintas que pueden llegar a ser estas fuentes de datos [20].

Aspectos positivos		
Scopus	WoS	Google Scholar
• Grandes informes • Cobertura disciplinaria internacional y especializada • Incluye artículos de prensa	• Sólo revistas de influencia • Cobertura hasta 1900 • Editor neutral y objetivo	• Todo tipo de documentos • Encuentra más citas en más áreas • Cobertura internacional e interdisciplinaria

Cuadro 3.1: Puntos positivos de las fuentes de datos

Aspectos negativos		
Scopus	WoS	Google Scholar
• En sus inicios, debilidad en ciencias sociales y humanidades y a día de hoy, aún la tiene en sociología, física y astronomía • Errores tipográficos	• Sólo cubre diarios de influencia • Dificultad en búsqueda de autores con formatos extraños • Problemas de puntuación	• Dificultad para restringir en la búsqueda nombres de autores comunes • Poca clasificación • Calidad de datos cuestionable • Muchas fuentes no revisadas por pares • Necesidad de creación de perfil de citas para crear informes

Cuadro 3.2: Puntos negativos de las fuentes de datos

Para este sistema se tomaron en cuenta los puntos positivos y los puntos negativos de Scholar, y se decidió que sería un buen complemento ofrecer al usuario alguna información y que pueda contrastar su diferencia con las otras fuentes de datos. En concreto, se obtuvo el H-Index del autor y la gráfica de citas/año. Debido a la naturaleza de las fuentes de datos y del H-Index, este dato recogido va a tener un mayor valor que en Scopus o que en la WoS.

3.1.4 GGS Conference Rating

La calificación de conferencias GGS o GII-GRIN-SCIE Conference Rating, permite otorgar una puntuación a las conferencias en el mundo de la informática. Está patrocinada por GII (grupo de profesores italianos de ingeniería informática), GRIN (grupo de profesores italianos de informática) y SCIE (no teniendo que ver con el anterior del JCR, son la Sociedad Española de Informática).
De estos grupos, un conjunto de miembros se reúnen para calificar las conferencias mediante un algoritmo expuesto en su página. Para este proyecto, el algoritmo se puede resumir en una tabla de equivalencias, que lo relacionan con las explicaciones anteriores de rango de cuartil y de factor de impacto de JCR [21]:

Tier	Class	Impact factor	cuartil Rank
1	A++	>25	Q1
	A+	23-25	Q2
2	A	18-23	Q3
	A-	16-18	Q4
3	B	12-16	Not enough
	B-	10-12	Not enough
-	Work in progress	<10	Not enough

Cuadro 3.3: Correspondencia de categorías GGS.

Los datos completos que se usan en el sistema se obtienen de la página oficial del GGS [22], donde se puede descargar de forma anónima y gratuita un archivo de tipo Excel. Como con los datos de la WoS se trabaja sobre un documento CSV, aquí se exportarán estos datos a un documento del mismo tipo, de forma que resulte mas fácil su posterior extracción. También, en este caso, no todos los datos que nos ofrece el documento son de importancia vital para el sistema, usando sólo los enumerados abajo:

- Título
- Clase
- Nota
- Quartile Rank (Rango de cuartil)
- Impact Factor (Factor de Impacto)

3.1.5 Lenguajes de Programación

3.1.6 Python

Python es un lenguaje de programación que fue creado a finales de los 80 por Guido van Rossum en el Centro para las Matemáticas y la Informática (CWI). Se trata de un lenguaje de programación multiparadigma, es decir, permite realizar programación funcional, orientada a objetos e imperativa, lo que da al programador una gran libertad a la hora de elaborar código. A diferencia de otros lenguajes como Java, Python tiene un tipado dinámico, es decir, sólo evalúa los tipos a la hora de la ejecución del programa, no de la compilación. Se dice que este lenguaje es muy adecuado para la formación, debido a su filosofía o estilo ”pythonico”, que busca la legibilidad y transparencia del código. Un ejemplo sencillo es el hecho de que si se intenta lanzar un código que posee errores de indexación (tabulación incorrecta del la estructura del código), dará un error de compilación. Otro aspecto positivo del lenguaje es que

posee un compilador interactivo (intérprete de comandos), que permite ejecutar sentencia a sentencia, con el fin de poder visualizar el resultado de forma inmediata. Una de las curiosidades de Python es que a la hora de tratar los parámetros de los métodos o las funciones, el paso de las variables será por valor en caso de que sea un tipo básico e inmutable, y por referencia en caso de ser un objeto o de tipo mutable. Es por este motivo por lo que en muchas ocasiones será necesario enviar una copia para garantizar la no modificación del dato dentro del método o función [23].
Para este proyecto se escogió este lenguaje por la sencilla razón de que el Framework de Django está escrito en Python, aunque conforme se fue realizando la implementación, se agradecieron factores como el tipado dinámico, que facilitó muchas tareas (por ejemplo listas con diferentes subclases en su interior). La versión utilizada en concreto fue Python 3.6.8.

3.1.7 HTML

HTML 5 o HyperText Markup Language es el lenguaje por excelencia para la elaboración y representación de páginas web y está estandarizado por W3C (World Wide Web Consortium). El nombre de HyperText corresponde a la idea de correlacionar una página Web con otra, y el de Markup, a que se basa en la utilización de elementos o etiquetas (como ejemplo de Markup tenemos otros lenguajes como el famoso XML). Estas etiquetas tiene una estructura simple, en la cual el nombre de la etiqueta se incluye entre los caracteres <> para abrir, y </> para cerrar [24] [25] .

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE html>
<html lang="en-us" xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Test</title>
  </head>
  <body>
    <ul>
      <li>
        <code>foo:bar</code>
      </li>
    </ul>
  </body>
</html>
```

Figura 3.1: Ejemplo de la estructura en HTML.

En el caso de Django, HTML sólo se usará en los templates, para dar una apariencia visual a la información que se envía desde la vista.
Hay que puntualizar que HTML ofrece el contenido de la página Web, pero no su funcionalidad. Existen lenguajes que sirven para ofrecer esta funcionalidad, como JavaScript- Otros, como CSS, se utilizan para indicar con precisión y de forma abstracta al HTML la apariencia de lo que se muestra. En este proyecto se usan ambos, y aunque la apariencia del front-end de la aplicación es importante, la parte de JS lleva más dedicación y más relevancia a la hora de cumplimentar los casos de uso y ofrecer una funcionalidad completa a la aplicación.

3.1.8 Javascript

JavaScript es un lenguaje de programación orientado a la realización de métodos complejos en páginas web que lenguajes como HTML no pueden realizar. Aunque se suele utilizar para crear pequeñas funciones o scripts, es un lenguaje en el que se puede utilizar programación funcional, OO e imperativa. Mientras que HTML es ”estático”, JS permite darle cierta dinamicidad a la interfaz, un comportamiento en función a la interacción con el usuario o simplemente a cambios externos a él [26]. Tiene numerosas características, como por ejemplo que pueda crear una interfaz entera sólo con JS mediante el DOM. No resulta lo más eficiente, por lo que no se suele realizar, pero sí pequeños trozos que dependan de factores condicionales para mostrarse. Así mismo, también se pueden detectar cambios en la interfaz, como clicks o pulsaciones de teclado.
Para esta aplicación se usó de forma considerable JS, con ayuda en algunas ocasiones de la librería jQuery [27], que posee ciertas funciones específicas de control muy útiles. Hay que recordar que en ningún caso JS sustituyó a HTML, sino que completó la funcionalidad, aportando en los aspectos que el otro lenguaje no tenía.

3.1.9 Frameworks

3.1.10 Django 2.2.1

Django es un Framework de alto nivel orientado al desarrollo de aplicaciones Web mediante Python. Comenzó su desarrollo en 2003, siendo públicamente conocido como Django en julio 2005. Como buen proyecto de Open Source, ha ido mejorándose con cada versión hasta llegar a la actual (2.2.1), la cual será la que se utilice aquí. Es considerado como completo y versátil, ya que posee la base que cualquier aplicación web puede utilizar, pero que puede orientarse hacia cualquier ejemplo concreto. También posee características como escalabilidad y gestión de la seguridad (esta última será detallada más adelante en el apartado 4.3.1)[28] [29].
Para explicar un poco su funcionamiento, se utilizará la estructura y un pequeño dibujo que

simboliza como viaja la petición http hacia Django, y como se devuelve su procesamiento. A mayores de la estructura base que se indica, en esta aplicación existen diversos ficheros con extensión .py , que pueden ser tanto clases, como módulos de lógica de negocio para no saturar el views.py.

Estructura típica de árbol de un proyecto en Django:

- manage.py
- proyecto/
  - settings.py
  - urls.py
  - wsgi.py
- app/
  - admin.py
  - apps.py
  - models.py
  - tests.py
  - views.py
  - __init__.py
  - migrations/
  - templates/

A continuación se puede observar la figura 3.1.10, que muestra la comunicación entre capas de Django para contestar a cada petición HTTP. Es un dibujo genérico, lo que quiere decir que no hay necesidad de adoptar esta fórmula para todos los proyectos de este Framework, pero sirve para sentar unas bases y comenzar a desarrollar posteriormente. Del mismo modo que con el apartado de seguridad, más adelante se detalla el uso y comportamiento de cada capa en el apartado 4.2.1 (Model-View-Template).

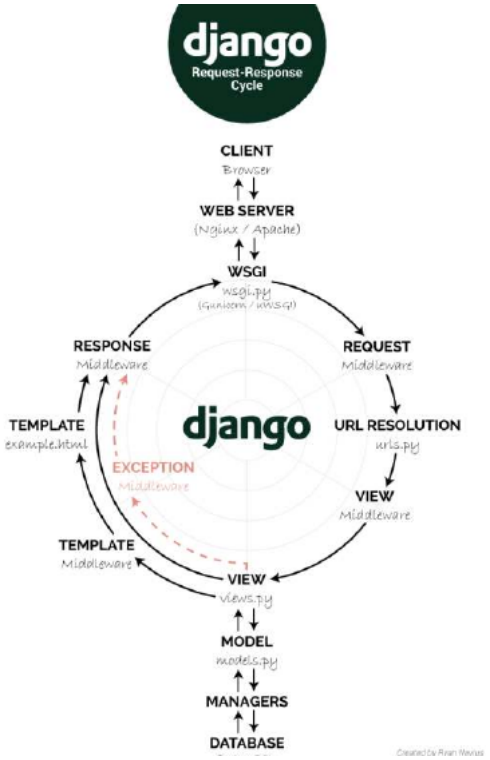

Figura 3.2: Viaje de una petición en Django

3.1.11 Bootstrap 4.1

Bootstrap 4 es un framework CSS que fue desarrollado inicialmente por la empresa Twitter, cuyo origen data de 2011. Bootstrap ofrece la posibilidad de crear interfaces web con diseño adaptativo o responsive design, es decir, que se adaptan al tamaño del dispositivo que la utilice, mediante un sistema de columnas. En concreto, hace uso de 12 columnas para añadir el contenido deseado [30] [31] [32]. A su vez, posee un amplio catálogo de elementos HTML con un diseño minimalista pero efectivo, fácil de usar y de configurar. Es por esto que Boostrap apuesta sobre todo por buscar la eficiencia de la interfaz y no por un gran de nivel de detalle (esto se puede conseguir de forma más personalizada). Así pues, se convierte en es uno de los Frameworks de front-end más adecuados para este proyecto, después de haberlo comparado con otros como Pure.CSS o Semantic UI.

3.2 Bases de datos

3.2.1 SQLite

Django trae por defecto un gestor de base de datos llamado SQLite. Este es un sistema de gestión, compatible con ACID y escrito en C. Tiene una curiosidad, y es que trabaja de forma ”enlazada” con el programa con el que se comunica, formando parte de él. De esta forma, se puede usar las funcionalidades con métodos y funciones simples que permiten una gran transparencia para el programador. Para conseguirlo, SQLite guarda todo en un sólo fichero de la máquina donde se ejecuta el programa [33]. Son estas características las que hicieron que el desarrollo se decantase por no cambiar el gestor de bases de datos por defecto. Para acceder a la BD, Django tiene un archivo denominado models.py, donde hay que definir los ”objetos”. A continuación se observa la facilidad con la que se puede declarar una relación, evitando cualquier tipo de consulta SQL. En concreto se trata de un dato que representa, para un usuario, su autor favorito (pudiendo tener más de uno claro está).

```
class Favorite(models.Model):

    id_usuario=models.IntegerField(default=0)
    url=models.CharField(max_length=1000)
    filtro=models.CharField(max_length=1050,default="")
    name=models.CharField(max_length=200,default="")
```

Una vez definido el dato y sus atributos, podemos usar funciones sencillas para consultar. Se usará el dato anteriormente definido para los ejemplos.

- Crear un nuevo dato:
  - conferencia = Conference(id_usuario=1,url=”http://ejemplo”,filtro=[],name=”ejemplo”)
- Eliminar todos los datos, o uno/varios elementos (por ejemplo, los de un usuario con ID 1):
  - Favorite.objects.all().delete()
  - Favorite.objects.filter(id_usuario=1).delete()
- Actualizar un elemento (cambiar nombre de un favorito a ”Autor_nuevo” para un usuario con ID 1):
  - Favorite.objects.filter(id_usuario=1).update(name=”Autor_nuevo”)

- Buscar por atributos (por ejemplo, favoritos para un usuario con ID 1):
  - favoritos=Favorite.objects.filter(id_usuario=1)

Tiene otras particularidades, como por ejemplo, poder sobrescribir el método save(). Es como si se pudiera redefinir el constructor para una clase de Python, pudiendo así añadir complejidad a las operaciones de guardado.
Como se puede observar, con SQLite integrado a Django se logra implementar la capa de acceso a datos con gran abstracción y facilidad.

3.2.2 MongoDB

Mongo 4.0.12 es una Base de datos NoSQL, que comenzó a desarrollarse en 2007 por 10gen Inc. y acabo siendo lanzada como producto independiente y OpenSource en 2009. Es orientada a documentos, que son almacenados en formato BSON (representación binaria de JSON), dentro de colecciones [34][35] .
En comparación a una BD estándar relacional, Mongo tiene una serie de ventajas e inconvenientes, lo que le hacen ser mas adecuada para ciertos ámbitos que para otros[36] .

Ventajas:

- El dato almacenado no tiene porque ser ”consistente” ni estar formalizado como en una BD relacional, es decir, se pueden guardar documentos con un contenido o esquema diferente.

- No se necesita un conocimiento de SQL, ya que posee varios métodos de consulta (todos orientados a los campos de los documentos), haciendo muy fácil la programación de las queries.

- Es sencillo indexar por campos del documento, aunque Mongo ya ofrece un ObjectId como índice por defecto.

- Ofrece una alta disponibilidad (la consulta SELECT es mucho más rápida en Mongo que en MySQL por ejemplo).

- Soporta muy bien un gran volumen de datos.

Inconvenientes:

- Al no poner un filtro al documento que se almacena, puede haber problemas al realizar lecturas, módulos que no existen, formatos diferentes, etc.

- No se lleva muy bien con las transacciones complejas, aunque desde la versión 4.0 esto se ha corregido en gran medida [37].

En este proyecto se ha usado esta BD a mayores de SQLite como respaldo para conseguir ofrecer la funcionalidad CU-08 y CU-09 (2.18, 2.19). Desgranando un poco más, cuando queremos guardar un CSV como tal tenemos dos opciones:

- Guardar una copia del autor del que fue descargado el documento, con todos sus datos (tanto propios, como de sus artículos) y el filtro que se aplicó.

- Guardar directamente una copia del documento descargado, de forma que no se tengan que volver a efectuar todos los procedimientos que devuelven un CSV al usuario (filtrar al autor, pasar sus datos a diccionario...).

A primera vista ya es fácil distinguir cuál de las opciones es la más viable, pero cuando se reflexiona sobre la posibilidad de llevar a cabo la primera (por ejemplo en MySQL), aparece la problemática (típica de toda BD relacional) de tener que realizar un diseño, cierta cantidad de tablas, reglas que pueden entorpecer las inserciones, etc. En cambio, con Mongo, se puede almacenar el diccionario que representa al documento descargado, sin más modificaciones que añadir los pares clave-valor de fecha, ID de usuario y filtro, de forma que luego cuando se necesite obtener los valores, sea tan sencillo como plasmar ese documento en un CSV para el usuario.
Para integrar su uso con Django, se escogió utilizar el módulo de Python PyMongo versión 3.4.0, como alternativa más simple y efectiva. Permite con funciones como ”find_one(’key’:value)” obtener los datos, con ”insert_one(diccionario)” insertarlos, y con muchas otras similares completar la funcionalidad de un acceso a una base de datos, con una abstracción total de lenguaje de consultas.

Capítulo 4

Diseño e implementación

Para introducir este capítulo, conviene dividir cada sección según si pertenece al diseño o a la implementación.

- **Diseño:**
  - Diagrama de clases UML 4.1, donde se muestra como estarán diseñadas las clases, en caso de tratarse de un sistema de orientado a objetos.
  - Los patrones de diseño que se utilizarán, los cuales sirven como base para el diseño, siguiendo una metodología estándar y de buenas prácticas. 4.2
  - Una sección correspondiente a la seguridad 4.3, un aspecto de vital importancia para ofrecer una aplicación robusta y segura, que no permita la obtención de información sensible o la modificación de los datos.
- **Implementación:**
  - Introducción de la técnica Web Scrapping 4.4, utilizada para obtener los datos de la fuente Google Scholar 3.1.3.
  - Metodología de integración de datos 4.5, donde se detalla cómo se juega con los datos de las diferentes fuentes para ofrecer una información unificada.
  - Unión de perfiles unificados 4.6, explicando el procedimiento de la desambiguación de los perfiles en Scopus.
  - Enumeración de las URLs que conforman la aplicación, para comprender que vista o dato representa cada URL 4.7.
  - Se acaba el capítulo con los detalles del despliegue en Docker 4.8, y la enumeración de las ventajas que esto implica.

4.1 Diagrama de clases UML

En esta sección se expondrá, mediante la figura 4.1, un diagrama de clases UML (Lenguaje Unificado de Modelado). Este diagrama muestra las clases utilizadas en la aplicación, y las relaciones existentes entre ellas. Las buenas prácticas establecen que la realización del diagrama debe ser previa a la implementación del código, de forma que es de vital importancia la buena elaboración del mismo. Sin embargo, en este proyecto el diagrama sufrió varias modificaciones durante el desarrollo para adaptarse a la aparición de nuevas funcionalidades.
Realizando un pequeño resumen, la aplicación se centra en autores (Author y Author2, siendo este último el que tiene más datos) y artículos (Article, que posee una lista de coautores o Coauthor y está asociado a una revista o Journal, donde se diferencian revistas de actas de conferencia o Conference). A mayores aparecen clases como el documento CSV (Csvdoc), el filtro que puede aplicarse a un perfil (Filtro), y una excepción personalizada, en caso de error en la respuesta de peticiones a las fuentes de datos (Exception_api).

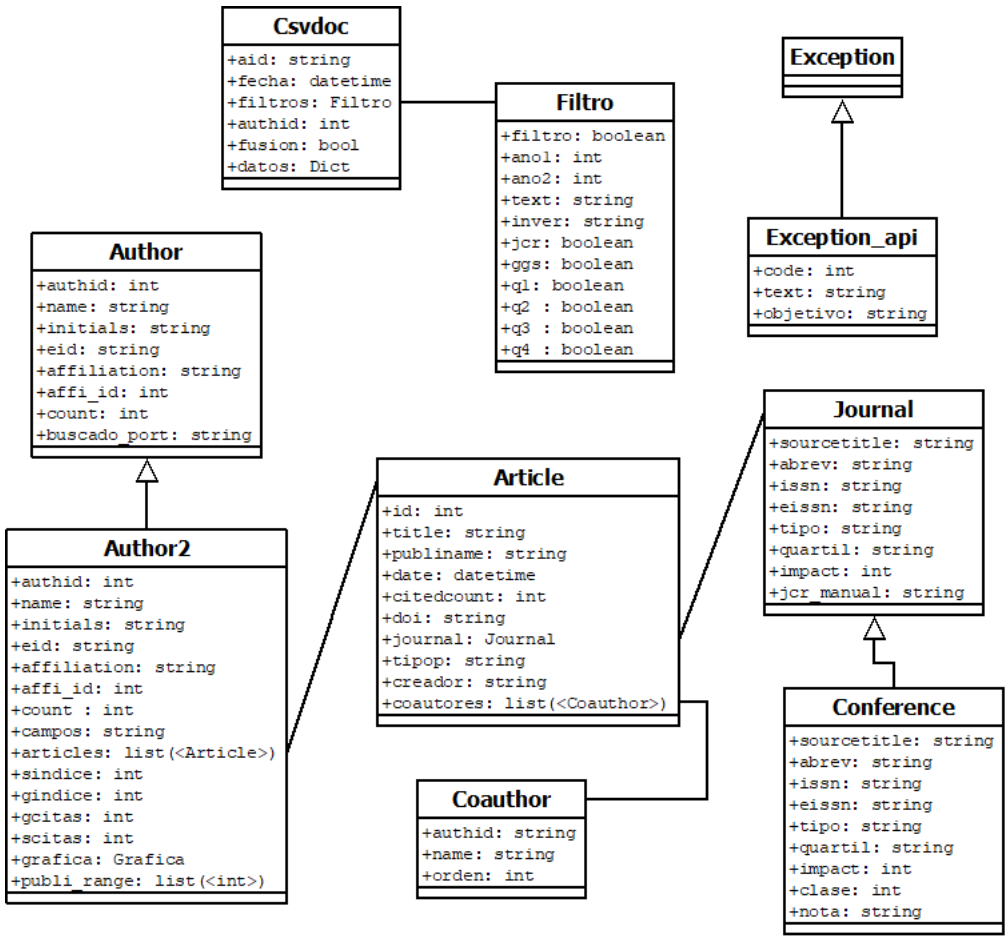

Figura 4.1: Diagrama de de clases UML

4.2 Patrones de diseño

4.2.1 Model-View-Template

Dentro de los patrones de diseño en ingeniería software, el MTV o patrón Model View y Template es el que integra Django en su estructura [38] . Es una vertiente del famoso patrón MVC o Model View Controller [39], pero difiriendo sobre todo en la capa de la vista y el controlador. En la siguiente tabla se hace una comparativa de ambos que ilustra las diferencias y similitudes que poseen.

Modelo	Componente	
MTV	Model	Capa de acceso a las bases de datos. En Django sirve también para definir las clases que tendrán su lugar en la BD, especificando sus campos (así se evita realizar la consulta sobre el SQL directamente) [40]
	Template	Aquí se indica la apariencia, es decir, el cómo lucirá la interfaz web, totalmente abstraído de los datos que le llegan desde views [41]
	View	Recibe las peticiones mediante la URL, las procesa (es decir, incorpora la lógica de negocio) accediendo a demanda a la capa del modelo, y termina devolviendo un template con los datos requeridos. También puede especificar el aspecto enviando elementos a los templates ya definidos [42]
MVC	Model	Igual que en el MTV, se encarga de el acceso a las bases de datos, pero también incluye la lógica de negocio
	View	Gestiona todo lo relativo a la interfaz, de forma completamente abstracta al modelo
	Controller	Es la capa encargada de la comunicación entre modelo y vista

Cuadro 4.1: Diferencias entre MTV y MVC

En la siguiente figura 4.2 se observan las capas del patrón MTV y como están conectadas entre sí para la transmisión de información.

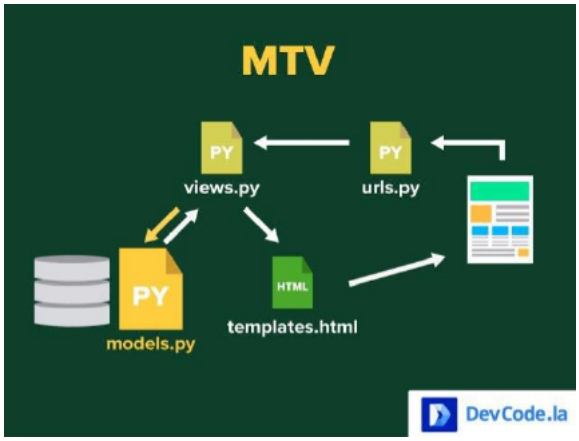

Figura 4.2: El patrón MTV en Django [1]

4.2.2 Caché a-side

El segundo patrón de diseño que se emplea en este proyecto es el Patrón Cache-Aside. El uso del mismo nace de la naturaleza de la aplicación, la cual necesita un respaldo en memoria para guardar la información que se va recuperando de las APIs [43]. Por ejemplo, una petición a Scopus tiene un tiempo de respuesta bastante bajo, pero el proceso completo de elaboración del perfil de un autor exige una cantidad de tiempo considerable (una cantidad que el usuario no va a estar dispuesto a invertir cada vez que haga alguna operación). Además, dado que esta aplicación integra datos de diversas fuentes, existen operaciones como relacionar una revista o una conferencia con las que están enumeradas en los datos de la WoS o de la GGS que conllevan un tiempo ”elevado” de ejecución, pese a no tener que realizar peticiones a ninguna API.
Así pues, se opta por escoger este patrón de diseño, que en rasgos generales sigue esta estrategia:

-Buscar si el dato que queremos está en caché.

-Si lo está, cogerlo de aquí, sino lo está, cogerlo de la BD (en nuestro caso, las APIs).

-Guardar una copia del dato en la memoria caché.

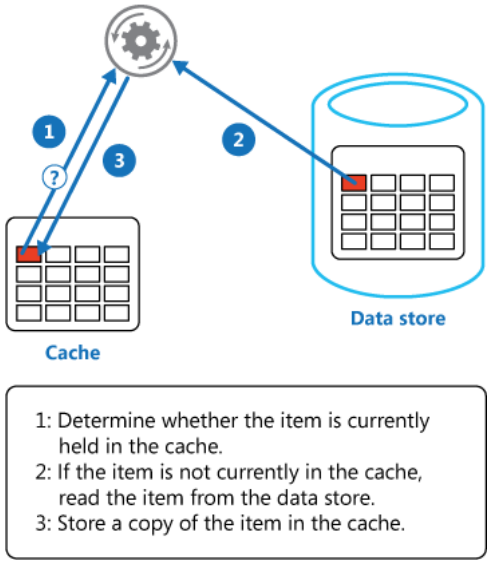

Figura 4.3: Representación del uso de memoria caché

Una de las grandes desventajas de usar este patrón es la premisa de que el diseñador debe establecer un tiempo de validez del dato en caché, es decir, un tiempo en el que el dato de no volver a guardarse, será eliminado. Dada la naturaleza de los datos, en este caso no se pueden guardar durante un periodo grande, ya que es posible que en el transcurso del tiempo, los datos obtenidos sean diferentes a los anteriores (por ejemplo, si el H-Index de Google Scholar cambia, la aplicación debería reflejar ese cambio en el menor tiempo posible). De esta forma, usaremos un tiempo medio de 3600 segundos (una hora), que aumente el rendimiento del sistema pero no lo penalice demasiado en cuanto a la consistencia de la información ofrecida respecto a las fuentes de datos. Este tiempo se basa en la suposición de que el usuario estará utilizando la aplicación de forma continuada durante como mucho una hora. Como consecuencia se puede evitar afrontar otro de los grandes inconvenientes de la caché, que es el límite de memoria existente. Este límite no se debería de ver afectado dado que los datos no permanecen mucho tiempo en memoria (y son relativamente pequeños).
En cuanto a la implementación del patrón, lejos de utilizar un gestor de BDs auxiliar o los anteriormente mencionados, nos servimos de un módulo de Django ya especializado para estos casos [44]. Ofrece diversos métodos, de los cuales sólo se utilizaron los dos siguientes:

cache.set(key, dato, t)

cache.get(key)

Como el nombre indica, el primero sirve para guardar el dato en la memoria caché, asociándolo a la key o llave introducida (ésta deberá ser un string sin caracteres extraños ni espacios) y estableciendo un tiempo t de expiración, el cual será la duración del objeto en memoria. Mediante el get, simplemente obtenemos el objeto asociado a la llave en caso de que siga en

memoria, o un None en caso contrario.

4.2.3 Singleton

El patrón de diseño Singleton o instancia única es aquel que controla la creación de objetos relativos a una clase, obligando a la existencia de una sola instancia por clase. Sólo creará una instancia si aún no existe una previa. Para regular este comportamiento, se implementa el patrón en el constructor de la clase, controlando desde el primer momento la creación de instancias [45].
Para esta aplicación, se considera útil la implementación del patrón Singleton para el acceso a la base de datos MongoDB. Para poder interactuar con la base de datos, se necesita la creación de una conexión cliente. Si cada vez que el sistema necesita acudir a Mongo para realizar alguna operación se instancia un nuevo acceso, se crearían múltiples conexiones que podrían quedar abiertas durante un tiempo incierto. Gracias a este patrón, se consigue que sólo se tenga una instancia por sesión, denegando la existencia de más de una conexión a la BD.
Como se muestra a continuación, en Python resulta relativamente sencillo implementar el patrón Singleton [46].

```
class Cliente:

        _instance=None

        def __init__(self,cliente):
                self.cliente=cliente

def Singleton(cls):

    if not cls._instance:
        print ('primera instancia del Singleton de Mongo')
        cls._instance = cls(MongoClient('localhost', 27017))
    return cls._instance


def inicializaDb():

        client=Singleton(Cliente)
```

```
        return client.cliente
```

4.3 Seguridad

4.3.1 La seguridad que Django ya nos ofrece

El Framework de Django ofrece una capa de seguridad de forma automática, que evita tener que realizar comprobaciones a la hora de desarrollar una aplicación Web. A continuación se enumerarán algunos de los sistemas de ataque más habituales para los cuales Django tiene protección [47].

- **XSS (Cross Site Scripting):** Este tipo de ataque se basa en la inyección de código maligno en el servidor a través del navegador. Existen tres tipos de ataques, los que afectan al DOM o Modelo de Objetos del Documento (lado del cliente), los que se producen cuando el usuario accede a un link o URL contaminado (no persistente) y los que se almacenan en el server de forma que cuando cualquier usuario quiera acceder al sitio web ejecute el código malicioso sin percatarse (persistente). Para protegerse ante esto, Django integra seguridad en los templates para caracteres sospechosos que puedan pertenecer a un código JavaScript malicioso.

- **CSRF (Cross Site Request Forgery):** Aquí el atacante tiene la finalidad de suplantar la identidad de usuario de la víctima, robando las credenciales y disponiendo de ellas para fines maliciosos. Para esto, el atacante debe modificar el código HTML de la aplicación, y es aquí donde Django integra su protección, ya que para cualquier formulario cuyo método sea POST, exige la utilización de un token {% csrf_token %}, que garantiza la legitimidad entre server y usuario.

- **SQL Injection:** Para realizar SQL Injection o Inyección de SQL malicioso, el atacante ha de conseguir lanzar una consulta a su voluntad fuera de las funcionalidades del sistema. Dado que Django no se ve mezclado en la capa de alto nivel con consultas SQL (sino que lo hace a través de la capa del modelo), resulta imposible introducir una sentencia SQL desde el exterior.

- **Clickjacking:** La finalidad de este ataque es que la víctima haga clicks en una página aparentemente legítima pero que en realidad no lo es. En caso de que el navegador lo soporte, Django integra protección contra el Clickjacking, en forma del middleware X-Frame-Options.

4.3.2 HTTPS

HTTPS o HiperText Transfer Protocol Secure es un protocolo cuya finalidad es garantizar la seguridad en la comunicación entre servidor y cliente, a fin de que esta comunicación no sea secuestrada. Esta seguridad se implementa en la capa de transporte y ofrece 3 capas [48]:

- Cifrado, se cifran los datos enviados entre cliente-servidor.
- Integridad de los datos, los datos no pueden alterarse en el medio de la comunicación por un agente externo, sin ser detectado.
- Autenticación, proporcionando seguridad contra ataques ”man-in-the-middle” y confianza al usuario.

Para conseguir esto, HTTPS basa su funcionamiento en el uso de certificados. El servidor envía al cliente (navegador) un certificado, el cual debe verificar la autenticidad y enviar una clave al servidor. Esta clave será la utilizada para cifrar los datos en el futuro. Para garantizar el correcto funcionamiento de este intercambio, se deben usar certificados de seguridad fiables, cuya CA pueda verificar de forma correcta la dirección web [49].
Para este proyecto no se ha implementado este protocolo por falta de tiempo y recursos, pero es prácticamente obligatorio mencionarlo a la hora de elaborar una aplicación de servicios Web. Además, en este caso, la información que se mueve no implica datos personales de usuario, por lo que este sistema estaría exento de riesgos que comprometiesen información sensible.

4.4 Web Scrapping

El acceso a datos que se ha visto para este proyecto tenía su base en solicitudes a BDs o peticiones a ciertas APIs vía HTTP. Si bien esto hace muy cómoda la labor, no todas las fuentes de datos proporcionan este tipo de accesos, y esto es lo que ocurre con Google Scholar. Aquí aparece la necesidad de usar el Web Scrapping, una técnica que consiste en obtener información a partir de la que está en una página web de forma lícita (aunque ha habido problemas por la cantidad de posibilidades que Web Scrapping ofrece a un usuario con malas intenciones). Su funcionamiento es, en resumen, simular el acceso de un usuario humano sin necesidad del mismo, y obtener todo los datos que se podrían observar desde la pantalla, o incluso más [50][51].
Dado que la aplicación está desarrollada en un entorno Python, se escogió la librería BeautifulSoup [52][53]. El proceso es sencillo, basta con realizar una petición a la URL correspondiente de Google, introduciendo los datos que identifiquen al autor del que queremos la información, y procesar el resultado de esta petición con BeautifulSoup, que será estructurado de

forma anidada. Entendiendo un poco la estructura de un documento de etiquetas HTML para poder moverse a través de sus nodos y mediante los diversos métodos de la librería, se puede acceder a la información de forma rápida y sencilla. Así es como se extrae el H-Index y los datos de la gráfica del autor. Además, al no necesitar autenticación para ver los datos del autor, se evita la complicación que pueda suponer un login con esta técnica.

4.5 Metodología de integración de datos

Una vez expuestas las diversas fuentes de datos utilizadas en el proyecto, llega el momento de explicar con un poco de detalle el proceso al que se someten los datos para lograr obtener una salida consistente, unificada y de valor para el usuario. Los datos de cada fuente ya han sido explicados previamente, pero no como se integran.

- **Scopus**
  - Método de obtención de datos: Llamadas a la API de Scopus de Elsevier.
- **Google Scholar**
  - Método de obtención de datos: Web Scrapping a la vista del perfil el autor (se usó BeatifulSoup).
- **JCR**
  - Método de obtención de datos: Consulta a la BD de Django.
- **GGS**
  - Método de obtención de datos: Consulta a la BD de Django.

A mayores de estos datos conviene recordar que:

- Un autor puede tener de 0 a muchos artículos.
- Un artículo tiene que estar publicado en solo 1 revista, acta de conferencia, libro, etc.
- El método que relaciona un medio de publicación de Scopus con JCR y GGS varía con el tipo del medio de publicación, y esto es porque:
  - Si el medio es una revista, ésta suele tener un ISSN asociado, lo cual ya te da una correlación directa con la información que hay en JCR, simplificando mucho la asociación. En caso de que la revista no tenga ISSN en Scopus, se procederá a comprobar si encajan los títulos.

  - Si el medio es una conferencia, existe una mayor complejidad, ya que éstas no tienen una clave en GGS como puede ser el ISSN de las revistas, sólo poseen el título, y algunas un acrónimo. Así pues, el primer paso es mirar si para cada conferencia que obtenemos de Scopus, hay alguna coincidencia con el acrónimo de GGS. En caso positivo, la relación finaliza y obtenemos de esta segunda fuente de datos todo lo necesario. En caso negativo, se recorrerá toda la lista de conferencias analizadas en la GGS, y se aplicará un método denominado ”sequencematcher” con el título. Este método compara ambos títulos para obtener un porcentaje de ratio de similitud entre ambos. Se cogerán todas las revistas que posean un ratio mayor a 0.8 de coincidencia y posteriormente, se seleccionará de éstas la mejor.
    Es de utilidad suprimir de los títulos obtenidos de Scopus palabras clave como ”Proceedings”, que no van a aparecer en el GGS y bajan el ratio de coincidencia, pudiendo dar lugar a error.

Se va a asociar cada vista o acción que provoque una petición de datos a cualquiera de las fuentes con los datos que se integran para poder ver el proceso con claridad.

- Vista nº 1: Listas de autores.
  - Fuentes usadas: Scopus.
  - Dato final obtenido: Lista de autores.
  - Procedimiento: Se obtienen de Scopus la lista de autores correspondientes a la búsqueda.
- Vista nº 2: Perfil de un autor.
  - Fuentes usadas: Scopus, Google Scholar, GGS, JCR.
  - Dato final obtenido: Perfil del autor casi completo.
  - Procedimiento:
    * Se obtienen todos los datos posibles de Scopus según las peticiones.
    * Se intenta buscar la correlación entre revistas y JCR para asignar a cada artículo del autor la cualificación que le corresponde según su medio de publicación.
    * Se hace lo mismo para la correlación entre artículos que han sido publicados en actas de conferencia y GGS.
    * Se realiza el WebScrapping en Google Scholar para obtener los datos correspondientes y añadirlos al autor.
- Vista nº 3: Descarga de documento CSV.

  - Fuentes usadas: Scopus.
  - Dato final obtenido: Perfil del autor completo.
  - Procedimiento:
    * Se realiza para cada publicación del autor una llamada a Scopus que le obtenga su lista de coautores 3.1.1.
    * Llegados a este punto, puede ser que el usuario haya filtrado o no las publicaciones del autor por diversos parámetros. Como realizar este proceso resulta muy costoso en cuanto al rendimiento del sistema se refiere, se sigue la siguiente estrategia:
      · Solo se solicitan los datos de los artículos que están visualmente expuestos, es decir, después del resultado de la filtración. Si no hay filtros, todos.
      · Se guarda en cache el perfil del autor con todos los artículos, no sólo los filtrados, pero ya con estas publicaciones asignadas a su lista particular de coautores.
      · Cuando se vuelva a solicitar la descarga, se repite el proceso, pero si la publicación ya tiene la lista, no hace falta hacer la llamada a la API (evitando recargar de forma innecesaria datos que ya están calculados).

4.6 Unión de perfiles duplicados en Scopus

Aunque Scopus es, de las fuentes de información escogidas, una de las más fiables y consistentes, tiene entre otros un defecto, y es que un autor puede tener sus datos divididos entre uno o más perfiles dentro de la base de datos. Ante esta situación, hay que explorar las posibilidades para poder unir la información de los diversos perfiles que pueda tener un único autor y ofrecer esa información al usuario.

Consideraciones:

- Si la propia base de datos es incapaz de unificar los perfiles, tiene sentido pensar que este sistema no va a ser capaz de hacerlo tampoco, al menos de forma automática sin intervención humana.
- Un autor que tenga varios perfiles, tiene una particularidad que ayuda a la fusión de ambos, y es que todos tienen la misma afiliación actual. Habría que despreciar cualquier perfil en caso contrario (que hubiese varios perfiles para un mismo autor con distinta afiliación actual), ya que eso supondría inyectar al sistema información errónea.

- Los perfiles duplicados suelen ser ocasionados por un error a la hora de situar el nombre como nombre y los apellidos como apellidos. Por ejemplo, para una misma investigadora, encontramos perfiles con los siguientes nombres:

  Costa Martínez, Catalina
  Martínez, Catalina
  Martinez Costa, Catalina

Teniendo esto en cuenta, se optó por seguir la siguiente metodología,una vez el usuario busca un autor por nombre y apellidos y después escoge en la lista ofrecida.

- En primer lugar, se busca, para cada combinación de nombre y apellidos, perfiles que coincidan con la afiliación actual del autor escogido.
- Al final del proceso, puede que el sistema no haya encontrado ningún ”duplicado”, o puede que haya encontrado uno o más posibles perfiles.
- Para hacer la fusión de la forma más correcta de cara al usuario, se le ofrecerán en caso de existir, una lista con el/los perfiles que coinciden con el autor que está consultando, y para que sea el usuario el que decida si quiere o no unir estos datos.

Una vez seleccionados los perfiles a unir por parte del usuario (en caso de haber escogido esta opción), el sistema ofrecerá un único perfil con los datos de todos a la vez, es decir:

- La lista de artículos estará compuesta por todos los artículos de los perfiles fusionados.
- Los datos a nivel personal estarán divididos en tarjetas por perfil.
- A la hora de obtener el CSV, y dada la naturaleza de los datos, se fijará el mayor H-Index del autor como H-Index global, y se hará un sumatorio de citas como citas totales.

4.7 URLs

De acuerdo al funcionamiento de la aplicación de Django, las peticiones son redirigidas al archivo urls.py, donde según la URL de la misma, acudirá a la función correspondiente del archivo views.py (donde se elabora la respuesta). Para la parte de administración ofrecida por Django, la URL base es http://127.0.0.1/admin/, desde donde se puede ejecutar el caso de uso 2.22. En cuanto al resto de funcionalidades para el usuario, a continuación se muestran cada una de las posibles URLs con su correspondiente representación y con sus tipo de petición (GET o POST). Se debe tener en cuenta, que todas las URLs mostradas son parciales, es decir, se crean a partir de http://127.0.0.1/data/.

- / Cubre las vistas 2.1 y 2.2
  - GET /
  - GET /?second=&first=
  - POST / autenticarse
    - Parámetros:
      - user (Usuario)
      - passwd (Contraseña)
  - POST / registrarse
    - Parámetros:
      - userr (Usuario)
      - passwdr (Contraseña)
      - passwdr2 (Confirmación de contraseña)
- GET /<int: auth_id> (Abajo observamos los filtros posibles según activación de checkboxes, pueden ir todos juntos en la URL o por separado como están). Cubre las vistas 2.3, 2.4, 2.5, 2.6 y 2.7
  - POST / seleccionar el número de perfiles a fusionar si los hay
  - POST / requiere estar autenticado, guardar o eliminar favorito
  - POST / añadir cualificación al artículo
    - Parámetros:
      - quart(Quartile Rank)
      - impact(Factor de Impacto)
      - id_art(Id de artículo)
  - GET /?ano1=&ano2=&filte=&filteinv=&filtroanual=on
  - GET /filte=&filteinv=&text=on
  - GET /q1=on
  - GET /q2=on
  - GET /q3=on
  - GET /q4=on
  - GET /checkjr=on
  - GET /checggs=on
- GET /downloadcsv/<int:auth_id>

- GET /downloadoldcsv/<int:auth_id>
- GET /fusion/<str:auth_id tiene los mismos filtros que /<int: auth_id> . En este caso se utiliza un tipo str (string o cadena de caracteres) porque irán todos los auth_ids de los perfiles unificados

4.8 Despliegue en Docker

Este proyecto ha sido desarrollado en un entorno de multi-versiones como es GitLab. El hecho de que sólo exista un desarrollador penaliza en gran medida la potencia de un repositorio de versiones, pero se determinó que usar git es una solución ya prácticamente obligatoria en el día de hoy. A mayores, hay 4 razones esenciales por las que se utilizó git:

- Se desconoce si en el futuro el proyecto será ampliado o si el número de desarrolladores cambiará.
- En caso de ocurrir algún percance con la infraestructura donde se desarrolla la aplicación, el código al completo estará a salvo en el repositorio remoto.
- El desarrollador nunca sabe si en algún momento tendrá que recuperar código borrado, lo cual mediante el sistema de los commits es posible.
- Se puede ligar un repositorio a Docker. De este modo se puede separar el desarrollo de la aplicación de el despliegue de la misma.

Docker es un sistema de contenedores portable, cuyo fin es poder ejecutar una aplicación en cualquier sistema, independientemente del software y del SO instalado en éste. Explicado de otro modo, Docker permite encapsular una aplicación en un contenedor y transportarla a cualquier entorno sólo con la necesidad de tener Docker instalado [54]. Por ejemplo, para levantar el servicio de esta aplicación, es obligatorio un sistema Unix, tener Python y sus módulos correctamente instalados, que correspondan las versiones de los mismos... Con Docker, un usuario podría realizar un despliegue de la aplicación siendo totalmente ajeno al código y al funcionamiento de la misma. Según RedHat [55], varias de las ventajas de los contenedores de Docker son:

- **Modularidad:** permite ir a una parte de la aplicación y no obligatoriamente al global.
- **Control de versiones de imágenes y capas:** las imágenes de Docker están formadas por capas, las cuales representan cada cambio que se vaya realizando.
- **Restauración:** basada en las versiones, permite recuperar capas antiguas en caso de necesidad.

- **Implementación rápida:** evita pérdida de tiempo en la implementación y despliegue.

A la hora de dockerizar la aplicación, se crean los 3 típicos archivos con la información concreta para la aplicación. El fichero comúnmente conocido como *docker-compose.yml* es necesario, ya que se va a desplegar más de un servicio en el contenedor (el propio Django y MongoDB). En las figuras 4.4, 4.6 y 4.5 se muestra este contenido, que si se analiza sería:

- Dockerfile
  - Se usa una imagen de Python con la versión correcta (3.6.7) como base.
  - Se prepara el entorno /data, donde estarán todos los archivos del código.
  - Se ejecuta el git clone, que es el encargado de descargar del repositorio remoto a un repositorio local los archivos del código. Se utiliza un token personal de la cuenta de GitHub para autorizar la descarga.
  - Se copia el contenido del repositorio al entorno.
  - Se instalan todos los módulos detallados en el requirements.txt.

```
FROM python:3.6.7

ENV PYTHONUNBUFFERED 1
RUN mkdir /data
RUN chmod -R 777 /data
WORKDIR /data

RUN git clone -b docker https://<git_token>:x-oauth-
basic@github.com/marcoro2/tfgmro.git .

ADD . /data/

COPY . /data/

RUN pip install --upgrade --user pip

RUN pip3 install --trusted-host pypi.python.org -r
requirements.txt
```

Figura 4.4: Dockerfile

- docker-compose.yml
  - MongoDB

    - En el campo *volumes* se añade un volumen donde pueda almacenar cualquier fichero que genere en el futuro.
    - En *enviroment* se indican los tokens de acceso.
    - 27017:27017 es el puerto por el que ofrecerá el servicio.
  - Django
    - Comando que hay que ejecutar para levantar Django.
    - En *volumes* se especifica el directorio donde se ubican los archivos. Situamos el volumen donde está descargado el repositorio del git.
    - El puerto por donde se va a ofrecer el servicio viene indicado por el *network mode host* , que simboliza la dirección de localhost o 127.0.0.1.
    - 8000:8000 es el puerto por el que ofrecerá el servicio.

```
version: '3'

volumes:
  data-db:

services:

  mongodb:
    image: mongo
    restart: always
    volumes:
      - ./data-db:/data/db
    environment:
      MONGO_INITDB_ROOT_USERNAME: root
      MONGO_INITDB_ROOT_PASSWD: mongoadmin
      MONGO_INITBD_DATABASE: tfgcient
    ports:
      - "27017:27017"

  web:
    build: .
    command: python3 manage.py runserver
    volumes:
      - ./data
    ports:
      - "8000:8000"
    network_mode: host
```

Figura 4.5: Docker-compose.yml

- requirements.txt

– Este archivo recoge los módulos de Python que son necesarios instalar en el entorno de trabajo para que la aplicación pueda funcionar de forma correcta.

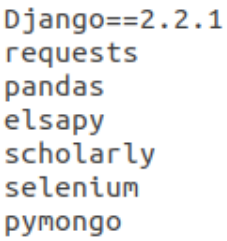

Figura 4.6: Requirements

Para concluir este apartado de despliegue, deberían tenerse en cuenta los siguientes puntos:

- Para levantar el servicio habrá que utilizar el comando *docker-compose up*, que levantará ambos servicios para proporcionar.
- Django no posee las características propias que debería tener un servidor para un entorno de producción real y debería integrarse con algo más compacto, como un Apache Tommcat. Para un posible futuro de la aplicación de este tipo, convendría integrar al *docker-compose.yml* este servicio.

Capítulo 5

Pruebas

Para garantizar que la aplicación cumpla los requisitos documentados con anterioridad y tenga el rendimiento esperado frente a cualquier tipo de situación, se planea la elaboración de la fase de pruebas, una fase vital en todo ciclo de desarrollo [56].
A pesar de que al desarrollador crea que su proyecto funciona de manera correcta, siempre puede haber un caso que no haya repasado si no realiza un testeo con profundidad. En este caso, se considera que con las pruebas de unidad y aceptación es suficiente para dar una buena cobertura al código y probar los casos de uso.

5.1 Pruebas de unidad

Las pruebas de unidad son aquellas pruebas que evalúan cada función o procedimiento creado en el código de forma separada (de ahí el nombre ”prueba de unidad o unitarias”). Es durante la elaboración del código cuando se deben ir creando este tipo de pruebas, de forma que cada vez que se modifique algo, se puedan correr los tests para verificar que los métodos siguen funcionando de forma correcta. Además, se hace casi obligatorio que este tipo de pruebas se ejecuten en un entorno o módulo independiente, que permita su realización de forma paralela al programa y no corrompa sus datos. Entre las características de una prueba unitaria encontramos que deben ser [57] [58]:

- **Automatizables:** evitar la necesidad de una intervención manual
- **Completas:** obtener la mayor cobertura del código posible
- **Repetibles o reutilizables:** las pruebas deben de poder utilizarse en más de una ocasión
- **Independientes:** las pruebas no deben afectarse entre ellas, es decir, cualquier operación que se realice en una prueba no puede verse reflejada en el resto

Django posee un archivo llamado tests.py dedicado exclusivamente a desarrollar las pruebas [59]. El procedimiento a seguir es crear para cada archivo Python del árbol de Django una subclase que hereda de TestCase, y para cada función o método diversas comprobaciones con cada posible situación, es decir, que atraviese todas las líneas de código, que compruebe datos extraños, corrompidos, listas vacías, etc. Una vez elaboradas, se utilizará el comando ***python3 manage.py test data***, estando situados en la carpeta tfg.

5.2 Pruebas de aceptación

Las pruebas de aceptación o User Acepttance Testing (UAT) son aquellas que se realizan en la parte final del desarrollo, cuando la aplicación está cercana a la fase de entrega al cliente. Su función última es saber si todos los aspectos del sistema cumplen las expectativas del usuario final que va a disfrutar de sus servicios, no solo a nivel de capa de negocio, sino también en cuanto a la parte de la vista o interfaces. Es por esto que estas pruebas necesitan un tercer agente a mayores del tester, que será un grupo de usuarios que prueben las funcionalidades y otorguen una opinión para cada una. De este modo se puede, no sólo descubrir algún bug o error que el propio diseñador no pudo prevenir, sino también aspectos como la comodidad del diseño de la interfaz de cara al usuario , alguna funcionalidad que no existente que se descubra durante la prueba, facilidad de uso, tiempos de respuesta, etc [60].
Para este proyecto se simularon estas pruebas que dieron como resultado las siguientes tablas. Los usuarios que las ejecutaron fueron los directores del proyecto, ya que según el concepto de Ingeniería Software, ellos desempeñan el papel de cliente. Para terminar se adjunta una última tabla que reúne los comentarios que se produjeron durante las pruebas, ampliando la información que pueda otorgar el haber superado cada prueba con o sin éxito, y aportando ideas para las futuras líneas de desarrollo.

PRUA-1	Buscar autor y ver sus datos
Descripción	El usuario quiere buscar a cierto autor, introduciendo para ello su nombre y apellidos. Posteriormente, escogerá de la lista ofrecida el que desee y podrá observar sus datos
Paso	Para buscar el autor, seguir los pasos del caso de uso CU-01, 2.11
	Para ver el perfil del autor, seguir los pasos del caso de uso CU-02, 2.12
Resultado esperado	El usuario debería saber donde introducir los campos de búsqueda, y saber seleccionar el autor que desea consultar. La interfaz debe ser intuitiva, tanto para buscar el autor, como para observar posteriormente los datos de su perfil
Resultado obtenido	El resultado esperado

Cuadro 5.1: Prueba de Aceptación PRUA-1

PRUA-2	Filtrar los datos de un autor
Descripción	Una vez situado en el perfil de un autor, el usuario deber poder filtrar los datos del mismo a voluntad
Pasos	Seguir los pasos del caso de uso CU-03, 2.13 para filtrar
Resultado esperado	El usuario podrá filtrar de forma sencilla los datos del autor, sabiendo en todo momento el porqué del resultado ofrecido después de haber filtrado
Resultado obtenido	El resultado esperado

Cuadro 5.2: Prueba de Aceptación PRUA-2

PRUA-3	Fusionar un autor con perfiles duplicados
Descripción	Cuando se da el caso de que el autor puede tener varios perfiles separados, el usuario debe saber escoger si fusionar o no estos perfiles, y saber hacerlo
Pasos	Seguir los pasos del caso de uso CU-06, 2.16
Resultado esperado	El usuario podrá ver de forma intuitiva el/los distintos perfiles posibles a unificar, saber fusionarlos o saber no fusionarlos y, posteriormente, entender en la vista del autor el resultado de esta fusión
Resultado obtenido	El resultado esperado

Cuadro 5.3: Prueba de Aceptación PRUA-3

PRUA-4	Descargar documento resumen
Descripción	Se intentará descargar el documento resumen de tipo CSV para un autor, esté fusionado o no, y esté filtrado o no
Pasos	Seguir los pasos del caso de uso CU-08, 2.18 para el actual y los del caso de uso CU-09, 2.19 para los que están guardados
Resultado esperado	El usuario sabrá donde clickar para descargar el documento, y entenderá cuando lo lea los datos que hay en el, para el estado en el que esté el perfil del autor (normal, filtrado o fusionado)
Resultado obtenido	El resultado esperado

Cuadro 5.4: Prueba de Aceptación PRUA-4

PRUA-5	Registrarse
Descripción	El usuario intentará crear un perfile en la aplicación
Pasos	Seguir los pasos del caso de uso CU-04, 2.14
Resultado esperado	No puede haber dudas en el acto del registro, sabiendo en todo momento como hacerlo, y resultando fácil y rápido para el usuario
Resultado obtenido	El resultado esperado

Cuadro 5.5: Prueba de Aceptación PRUA-5

PRUA-6	Login
Descripción	El usuario intentará iniciar sesión con sus credenciales
Pasos	Seguir los pasos del caso de uso CU-05, 2.15
Resultado esperado	Al igual que la anterior prueba, hacer login debería ser sencillo, rápido e intuitivo para el usuario
Resultado obtenido	El resultado esperado

Cuadro 5.6: Prueba de Aceptación PRUA-6

PRUA-7	Guardar un autor como favorito
Descripción	El usuario marcará un autor como favorito, y podrá consultarlo posteriormente de forma directa, sin tener que buscarlo. También puede eliminarlo de favoritos
Pasos	Seguir los pasos del caso de uso CU-10, 2.20
Resultado esperado	El usuario debe saber como guardar o eliminar un autor de favoritos, y además, saber acceder a el por favoritos sin tener que buscarlo (haciéndolo mucho más rápido y sencillo)
Resultado obtenido	El resultado esperado

Cuadro 5.7: Prueba de Aceptación PRUA-7

PRUA-8	Añadir o modificar cuartil e IF
Descripción	El usuario añadirá los datos de cuartil y factor de impacto para aquellos artículos que desee, siempre y cuando estos no hayan podido conseguir previamente los datos correspondientes en ninguna fuente de datos. También podrá modificar los datos que haya cambiado anteriormente
Pasos	Seguir los pasos del caso de uso CU-07, 2.17
Resultado esperado	El usuario debe saber como guardar nuevos datos o modificar datos que haya insertado antes. El sistema debe almacenar estos datos de forma correcta, mostrando además la fecha del cambio.
Resultado obtenido	El resultado esperado

Cuadro 5.8: Prueba de Aceptación PRUA-8

Comentarios del cliente	
Cliente 1	• Cuando descarga el CSV, el cliente nº 1 se percata de que le gustaría observar al lado de cada artículo, su año de publicación. Además, quiere saber la clase y la nota de los artículos de conferencia, no sólo su cuartil • Observa que no hay una funcionalidad que permita eliminar los documentos CSV descargados con anterioridad, lo que implica un gran crecimiento de la lista, que puede ser incómodo para el uso • En cuanto a la filtración, este cliente se ve un poco limitado cuando quiere filtrar por texto. Le gustaría poder filtrar por más de una cadena, sobre todo para eliminar varios artículos según diferentes trozos de texto
Cliente 2	• Observa que la gráfica de artículos de cada cuartil por año no ofrece información en el eje de las Y. El cliente se conforma cuando se da cuenta de que, poniendo el ratón sobre la gráfica, se revelan estos datos. Aún así sugiere que puedan visualizarse de forma inicial
Comentarios comunes	• Ambos clientes aprecian que el tiempo de respuesta de la aplicación para la descarga del documento CSV es algo lenta. Durante el proceso se les explica el porqué, y un tiempo después, cuando la caché comienza a alimentarse, los clientes notan como la aplicación ofrece una respuesta ya casi automática.

Cuadro 5.9: Comentarios de las pruebas de Aceptación

Capítulo 6

Conclusiones

Comienza el capítulo que cierra la memoria, exponiendo las conclusiones y lecciones aprendidas del proceso, y un pequeño balance de éxito o fracaso en cuanto a los objetivos definidos en la introducción.

Este proyecto ha llevado una línea bastante regular en cuanto a lo que progreso se refiere. Se optó por una metodología en la que los requisitos funcionales fueron apareciendo de forma incremental, de forma que se pudo ir mejorando y creando el contenido con cada revisión. Como consecuencia de esto, he aprendido mucho acerca de la comunicación con el cliente, la parte que quizás menos te prepara el grado, y la que creo que va a ayudarme mucho de cara al futuro profesional. El tiempo de realización se ha alargado más de lo esperado, en buena medida por este aumento gradual de requisitos en el estilo de ingeniería software que decidimos tomar. Se debe tener siempre en cuenta que este proyecto es una simulación de uno real, pero a pequeña escala. Esto quiere decir que hay metodologías que no tiene sentido aplicar cuando sólo existe un desarrollador y no un equipo.

En cuanto al procedimiento de la implementación, este no fue el ideal, ya que no se realizó el desarrollo del código y la ejecución de las pruebas totalmente en paralelo para cada revisión. Aún así, se han intentado seguir las buenas prácticas y bajo mi juicio se puede decir que se ha conseguido. Se han utilizado los patrones de diseño de forma correcta y se ha documentado de forma detallada su uso.

El resultado de la aplicación ha sido muy correcto, ya que se han cumplido absolutamente todos los requisitos funcionales. Además, el rendimiento ha ido mejorando mucho durante el desarrollo, consiguiendo que la aplicación en general responda muy bien cuando se trabaja con ella.

Para finalizar, he de decir que ha sido debido a una asignatura de tercero de mención, en la que hay una pequeña introducción a Django, el motivo por el que opté por realizar este trabajo. Aunque he de reconocer que los aspectos de front-end no son mis favoritos, estoy muy contento de haber trabajado con este Framework, ya que la elaboración del back-end y la co-

municación con las fuentes de datos ha sido una gran experiencia. En mi opinión se trata de un proyecto muy completo, en el que se tocaron muchos aspectos interesantes.

6.1 Líneas futuras

En esta sección de las conclusiones, se enumerarán ciertas mejoras o añadidos que debería acoplar el código en una futura nueva versión de la aplicación. Esto estará basado en el resultado de las pruebas de aceptación de cara a mejorar el nivel de satisfacción del cliente, y en ciertas apreciaciones por parte del desarrollador.

- En cuanto al caso de uso CU-08 2.18:
  - Añadir, a petición del cliente, información del año de cada artículo, y de métricas para los artículos de conferencia.
  - Por parte del desarrollador, dividir cada tabla en diferentes hojas para facilitar la comprensión del documento, y sobre todo, no mezclar datos en la misma hoja.
- Adicción del eje de las Y para la gráfica de artículos de cada cuartil por año.
- Ofrecer la opción de eliminar los documentos CSV descargados con anterioridad, para no saturar la lista de cada usuario.
- Implementar un filtro por texto más exhaustivo, que permita al cliente combinar diferentes cadenas de texto.
- De cara a la eficiencia, buscar un método que permita que la aplicación almacene datos de la API y que estos estén lo más actualizados posible. Otra opción es que la carga de datos sea totalmente transparente al tiempo de carga de la aplicación, de forma que pueda ir realizando las peticiones a la API por detrás.
- Añadir más campos posibles de búsqueda, para aquellos usuarios que no tengan claro cual es el nombre y apellidos del autor a buscar y puedan buscarlo por otros datos.
- Por último, de cara a una nueva versión, se puede explorar la opción de ofrecer una interfaz más cuidada, pero que mantenga la funcionalidad y la sencillez de la presente.

Apéndices

Apéndice A

Glosario de acrónimos

ACID *Atomicity, Consistency, Isolation and Durability*

API *Aplication Programming Interface*

CA *Autoridad Certificadora*

CSS *Cascading Style Sheets*

CWI *Centro de Matemáticas e Informática*

BD *Base de datos*

CSRF *Cross Site Request Forgery*

DOI *Digital Object Identifier*

DOM *Modelo de Objetos del Documento*

GGS *GII-GRIN-SCIE Conference Rating*

FECYT *Fundación Española para la Ciencia y Tecnología*

HTML *HyperText Markup Language*

HTTP *Hypertext Transfer Protocol*

HTTPS *Hypertext Transfer Protocol Secure*

IF *Factor de Impacto*

ISSN *International Standard Serial Number*

JCR *Journal Citation Reports*

JS *JavaScript*

MTV *Model View Template*

SaaS *Software as a Service*

SCIE *Indice Extendido de Citas de Ciencias*

SSCI *Indice de Citas de Ciencias Sociales*

OO *Orientado a Objetos*

SQL *Structured Query Language*

UML *Unified Modeling Language*

URL *Uniform Resource Locator*

WoS *Web of Science*

W3C *World Wide Web Consortium*

XML *Extensible Markup Language*

XSS *Cross Site Scripting*

Apéndice B

Glosario de términos

API Conjunto de funciones y procedimientos utilizados con el fin de desarrollar y/o integrar software, permitiendo añadir una capa de abstraccion entre la aplicación utilizada y el software desarrollado/integrado.

Framework Código independiente de la máquina que generan compiladores de determinados lenguajes (Java, Erlang,...) y que es ejecutado por el correspondiente intérprete.

DOI Identificación de un objeto digital que, aunque la URL que contenga al objeto varíe, éste no se verá afectado. Suele usarse en publicaciones electronicas (revistas cientificas, etc).

ISSN Código numérico reconocido de forma internacional para identificar publicaciones seriadas. Está fuertemente asociado al título de la publicación, y de haber una modificación en este, puede arrastrarse al ISSN.

NoSQL No sólo SQL, abarca los sistemas gestores de bases de datos que ni son relacionales ni, como el nombre indica, utilizan SQL como lenguaje de consulta. Como características principales suelen tener consistencia eventual,flexibilidad en el esquema (no se requieren tablas), buena escalabilidad horizontal y no todos garantizan los principios ACID.

Bibliografía

[1] Django mtv, 2018. [En línea]. Disponible en: https://www.javatpoint.com/django-mvt

[2] Requisito funcional, 2019. [En línea]. Disponible en: https://es.wikipedia.org/wiki/Requisito_funcional

[3] Técnicas para identificar requisitos funcionales y no funcionales, 2019. [En línea]. Disponible en: https://sites.google.com/site/metodologiareq/capitulo-ii/tecnicas-para-identificar-requisitos-funcionales-y-no-funcionales

[4] Requerimientos funcionales: Ejemplos, 2017. [En línea]. Disponible en: www.pmoinformatica.com/2017/02/requerimientos-funcionales-ejemplos.html

[5] Actor (UML), 2019. [En línea]. Disponible en: https://es.wikipedia.org/wiki/Actor_(UML)

[6] Guía para redacción de casos de uso. [En línea]. Disponible en: http://www.juntadeandalucia.es/servicios/madeja/contenido/recurso/416

[7] Requisito no funcional, 2019. [En línea]. Disponible en: http://www.pmoinformatica.com/2015/05/requerimientos-no-funcionales-ejemplos.html

[8] Ejemplos de requerimientos no funcionales, 2019. [En línea]. Disponible en: https://es.wikipedia.org/wiki/Requisito_no_funcional

[9] Modelo de prototipos, 2019. [En línea]. Disponible en: https://es.wikipedia.org/wiki/Modelo_de_prototipos

[10] ¿Qué es scopus?, 2019. [En línea]. Disponible en: https://biblioguias.biblioteca.deusto.es/Scopus

[11] Elsevier y Scopus. [En línea]. Disponible en: https://www.elsevier.com/es-es/solutions/scopus

[12] Elsapy (módulo de Python para Scopus), 2019. [En línea]. Disponible en: https://pypi.org/project/elsapy/

[13] Evaluación de revistas con JCR. [En línea]. Disponible en: https://www.recursoscientificos.fecyt.es/sites/default/files/incites_jcr_dec_2016.pdf

[14] Web of Science, 2019. [En línea]. Disponible en: https://clarivate.com/webofsciencegroup/solutions/web-of-science/

[15] Recursos científicos. [En línea]. Disponible en: https://recursoscientificos.fecyt.es/

[16] JCR y Wos, 2019. [En línea]. Disponible en: https://www.lluiscodina.com/jcr-wos-humanidades/

[17] Rango de percentil en JCR. [En línea]. Disponible en: http://help.prod-incites.com/incitesLiveJCR/JCRGroup/jcrJournalProfile/jcrJournalProfileRank.html

[18] Google Académico, 2019. [En línea]. Disponible en: https://es.wikipedia.org/wiki/Google_Acad%C3%A9mico

[19] Google Académico, paso a paso, 2019. [En línea]. Disponible en: https://biblioguias.uam.es/tutoriales/google_academico

[20] Scopus vs Google Scholar vs Wos, 2019. [En línea]. Disponible en: https://instr.iastate.libguides.com/c.php?g=901522&p=6492159

[21] Datos y métricas de ggs, 2018. [En línea]. Disponible en: http://gii-grin-scie-rating.scie.es/conferenceRating.jsf;jsessionid=378633E273517E882CB22577C4EF44AB

[22] Obtener datos sobre conferencias, 2018. [En línea]. Disponible en: http://gii-grin-scie-rating.scie.es/

[23] Python, 2019. [En línea]. Disponible en: https://es.wikipedia.org/wiki/Python

[24] HTML, 2019. [En línea]. Disponible en: https://developer.mozilla.org/es/docs/Web/HTML

[25] HTML, 2019. [En línea]. Disponible en: https://es.wikipedia.org/wiki/HTML

[26] Seguridad de las aplicaciones web Django, 2019. [En línea]. Disponible en: https://developer.mozilla.org/es/docs/Learn/JavaScript/First_steps/Qu%C3%A9_es_JavaScript

[27] Seguridad de las aplicaciones web Django, 2019. [En línea]. Disponible en: https://es.wikipedia.org/wiki/JQuery

[28] Introducción a Django, 2019. [En línea]. Disponible en: https://developer.mozilla.org/es/docs/Learn/Server-side/Django/Introducci%C3%B3n

[29] ¿Qué es Django?, 2019. [En línea]. Disponible en: https://tutorial.djangogirls.org/es/django/

[30] W3C, empezando con Bootstrap, 2019. [En línea]. Disponible en: https://www.w3schools.com/bootstrap4/bootstrap_get_started.asp

[31] Bootstrap, qué es y como funciona, 2019. [En línea]. Disponible en: https://www.axarnet.es/blog/bootstrap/

[32]

[33] SQLite, 2019. [En línea]. Disponible en: https://es.wikipedia.org/wiki/SQLite

[34] MongoDB, 2019. [En línea]. Disponible en: https://es.wikipedia.org/wiki/MongoDB

[35] MongoDB, página oficial, 2019. [En línea]. Disponible en: https://www.mongodb.com/es

[36] MySQL vs MongoDB, 2018. [En línea]. Disponible en: https://guiadev.com/mysql-vs-mongodb/

[37] Transacciones en mongo 4.0, 2018. [En línea]. Disponible en: https://www.gpsos.es/2018/08/transacciones-en-mongodb-4-0-ejemplos-de-uso/

[38] Estructura de Django, 2019. [En línea]. Disponible en: https://djangobook.com/mdj2-django-structure/

[39] Modelo-Vista-Controlador, 2019. [En línea]. Disponible en: https://es.wikipedia.org/wiki/Modelo%E2%80%93vista%E2%80%93controlador

[40] Djando model, 2018. [En línea]. Disponible en: https://www.javatpoint.com/django-model

[41] Djando template, 2018. [En línea]. Disponible en: https://www.javatpoint.com/django-template

[42] Djando view, 2018. [En línea]. Disponible en: https://www.javatpoint.com/django-view

[43] Patrón Caché-Aside, 2018. [En línea]. Disponible en: https://docs.microsoft.com/es-es/azure/architecture/patterns/cache-aside

[44] Framework de caché en Django, 2019. [En línea]. Disponible en: https://docs.djangoproject.com/en/2.2/topics/cache/

[45] Singleton, 2019. [En línea]. Disponible en: https://es.wikipedia.org/wiki/Singleton

[46] Implementar el patrón singleton en python, 2016. [En línea]. Disponible en: https://puropython.blogspot.com/2016/09/el-patron-singleton-o-una-instanciar.html

[47] Seguridad de las aplicaciones web Django, 2019. [En línea]. Disponible en: https://developer.mozilla.org/es/docs/Learn/Server-side/Django/web_application_security

[48] Proteger sitios web con HTTPS, 2019. [En línea]. Disponible en: https://support.google.com/webmasters/answer/6073543?hl=es-419

[49] HTTPS, 2019. [En línea]. Disponible en: https://es.ryte.com/wiki/HTTPS

[50] Web Scrapping, 2019. [En línea]. Disponible en: https://es.wikipedia.org/wiki/Web_scraping

[51] ¿Qué es el Web Scrapping?, 2016. [En línea]. Disponible en: https://sitelabs.es/web-scraping-introduccion-y-herramientas/

[52] Python + Web Scrapping = BeatifulSoup. [En línea]. Disponible en: https://www.geeksforgeeks.org/implementing-web-scraping-python-beautiful-soup/

[53] Documentación de BeatifulSoup. [En línea]. Disponible en: https://www.crummy.com/software/BeautifulSoup/bs4/doc/

[54] ¿qué es docker?, 2019. [En línea]. Disponible en: https://www.javiergarzas.com/2015/07/que-es-docker-sencillo.html

[55] Contenedores, qué es docker?, 2019. [En línea]. Disponible en: https://www.redhat.com/es/topics/containers/what-is-docker

[56] Continua integración, implementación y entrega, 2017. [En línea]. Disponible en: https://www.digitalocean.com/community/tutorials/an-introduction-to-continuous-integration-delivery-and-deployment

[57] Beneficios de las pruebas unitarias, 2017. [En línea]. Disponible en: https://apiumhub.com/es/tech-blog-barcelona/beneficios-de-las-pruebas-unitarias/

[58] Prueba unitaria, 2019. [En línea]. Disponible en: https://es.wikipedia.org/wiki/Prueba_unitaria

[59] Pruebas automatizadas en django. [En línea]. Disponible en: https://docs.djangoproject.com/es/2.2/intro/tutorial05/

[60] Pruebas de aceptación, 2019. [En línea]. Disponible en: https://es.wikipedia.org/wiki/Pruebas_de_aceptaci%C3%B3n_(inform%C3%A1tica)