Universidad San Jorge

Escuela de Arquitectura y Tecnología

Grado en Ingeniería Informática

Proyecto Final

<u>Easy Scholar Ranking</u>

Autor del proyecto: Jorge Aguirregomezcorta Aina

Directora del proyecto: María Francisca Pérez Pérez

Zaragoza, 11 de junio de 2021

universidad
SANJORGE
GRUPO SANVALERO

Este trabajo constituye parte de mi candidatura para la obtención del título de Graduado en Ingeniería Informática por la Universidad San Jorge y no ha sido entregado previamente (o simultáneamente) para la obtención de cualquier otro título.

Este documento es el resultado de mi propio trabajo, excepto donde de otra manera esté indicado y referido.

Doy mi consentimiento para que se archive este trabajo en la biblioteca universitaria de Universidad San Jorge, donde se puede facilitar su consulta.

Firma Fecha **11 de junio de 2021**

Tabla de contenido

Resumen

El ámbito de las ciencias de la computación cuenta con una gran cantidad de librerías y sitios web donde los investigadores difunden y distribuyen sus trabajos de investigación. Esta difusión ayuda al análisis y estudio de referencias que sirven para elevar trabajo a trabajo el conocimiento que la comunidad científica posee. Una de las necesidades de la industria es la ordenación de investigadores en base a su contribución para de esa manera poder estudiar y analizar los trabajos de investigación de aquellos investigadores con posiciones más altas.

Sin embargo, no existe ninguna plataforma que ofrezca al ámbito de las ciencias de la computación dicha necesidad. Es por esa razón que surge Easy Scholar Ranking, un proyecto con el objetivo de crear una aplicación web donde los investigadores pueden obtener rankings en base a diferentes parámetros seleccionados.

El resultado del proyecto incluye el estudio de tecnologías capaces de ofrecer la información necesaria sobre investigadores, la construcción de la aplicación web, y el despliegue de la misma. Ofreciendo los requerimientos necesarios para cubrir la ordenación de investigadores del ámbito de ciencias de la computación.

Abstract

The field of computer science features a great number of online libraries and web sites where researchers distribute their research papers. This diffusion helps the analysis and study of references that increase paper by paper the overall knowledge of the scientific community. One need the industry has is the ranking of researchers based on the number of contributions they have to allow researchers to study the papers of those standing in better positions.

However, there is no online platform that can offer researchers such functionality. It is because of this very reason that Easy Scholar Ranking is created, a project with the goal of creating a web application where researchers can create ranks by using different parameters.

The result of the project includes the study of different technologies capable of offering the information needed about the researchers, the development of the web application and the deployment of such application. Offering the necessary requirements to cover the ranking of computer science researchers.

1. Introducción

Un trabajo de investigación [1] es la culminación y producto final de un largo proceso en el que uno o más investigadores, a través de la aplicación del método científico, son capaces de razonar, analizar, organizar y componer una escritura académica del más alto nivel. Una forma de entender estos trabajos es pensar en ellos como si de seres vivos se trataran, ya que crecen y cambian conforme los investigadores los examinan, interpretan y evalúan fuentes relacionadas con un tema en específico. Las fuentes primarias y secundarias forman el núcleo de un trabajo de investigación, y corrigen el rumbo hacia el cual se acaba dirigiendo. Sin el respaldo e interacción con estas fuentes, el trabajo acabaría convirtiéndose en otro tipo de género de escritura. Un trabajo de investigación no solo sirve para avanzar el campo sobre el que escribe, pero también para proporcionar la excelente oportunidad a sus autores de incrementar su conocimiento sobre la materia.

La importancia de la difusión y distribución de trabajos es de vital importancia para poder seguir investigando sobre sus respectivos temas. Para ello, son muchos los investigadores que publican artículos en revistas y conferencias relacionadas con su área de investigación. Con el objetivo de poder consultar esta clase de artículos, existen páginas web que ofrecen a los investigadores, u otra clase de lectores, la oportunidad de consultar los artículos de investigación publicados por sus colegas. Un ejemplo de ello es dblp [2], un portal web que contiene un catálogo sobre artículos y conferencias del mundo de las ciencias de la computación.

Sin embargo, pese a los millones de registros que incluye dblp, no es posible comparar investigadores para medir su rendimiento o la posición en la que se encontrarían en sus respectivos campos. Tampoco pueden filtrarse los autores dependiendo de las conferencias o revistas en las que publican para conocer cuántos hay y la frecuencia con la que realizan publicaciones. Menos aún puede un investigador consultar estadísticas personales que ayudarían a visualizar sus datos más relevantes, lo cual podría servir para ayudar a mantener una ordenación de los autores, los campos de estudio, o incluso a motivar a dichos investigadores en la realización de sus trabajos.

La Figura 1 [3] presenta la búsqueda de un autor en la web de dblp. En la parte superior hay un campo de búsqueda que permite realizar búsquedas a partir de nombres de autores, fechas de publicación y nombres de revistas o conferencias. En la parte inferior se encuentra el resultado de la búsqueda, que muestra el autor identificado y las publicaciones en las que aparece. Finalmente, en la columna derecha se aprecian filtros de búsqueda relacionados con el autor, como autores con los que ha colaborado y revistas en las que ha publicado. Estos filtros ayudan a refinar la lista de publicaciones obtenida.

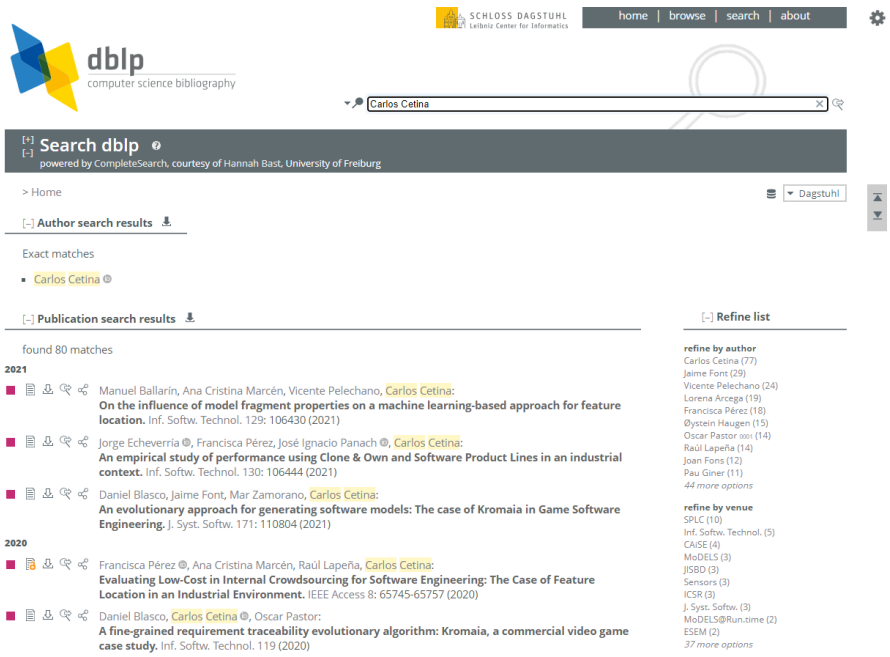


Figura 1. Búsqueda de un autor en dblp

El objetivo de Easy Scholar Ranking nace para proporcionar a la comunidad de investigadores de las ciencias de la computación una herramienta para poder consultar de forma sencilla información relevante sobre su trabajo y el de sus colegas profesionales. Concretamente, el proyecto tiene como objetivo construir una aplicación web donde los investigadores puedan ser capaces de realizar consultas que muestren tablas de clasificación ordenadas según el número de artículos publicados por estos. Otro de los objetivos es también que a través de otra serie de consultas puedan obtener información acerca de sus estadísticas como profesionales, y de esa manera tener una imagen concreta del nivel que desempeñan en sus respectivos campos.

El proyecto se estructura de la siguiente forma: el capítulo 2 presenta el estado del arte, donde se recogen y discuten las diferentes anteriores realizaciones del campo de estudio del proyecto. En el capítulo 3, se indican los objetivos planteados para la realización del proyecto y la correlación entre ellos. En el capítulo 4, se explica la metodología del proyecto. Es decir, cómo se ha realizado el trabajo, y como el flujo de alumnado-tutor funciona en las diferentes reuniones que se han dado a lo largo del desarrollo. A continuación, en el capítulo 5, se indica como se ha desarrollado la solución del proyecto, con sus consecuentes problemas, ideas y resoluciones. En el capítulo 6 se procede a la visualización del estudio económico, en el que se refleja tanto el presupuesto necesario como el beneficio económico que puede derivar del proyecto. En el capítulo 7 se discuten los resultados obtenidos tras la finalización de la solución creada para el proyecto. Finalmente, el capítulo 8 presenta las conclusiones generales del proyecto, las cuales reflejan el grado de cumplimiento de los objetivos planteados y las futuras ampliaciones del proyecto.

2. Estado del Arte

Para poder comprender la necesidad de Easy Scholar Ranking es preciso conocer o entender hasta cierto punto en que consisten las diferentes herramientas o plataformas que permiten a los investigadores obtener información sobre su campo de estudio. Tal y como se comenta en la introducción, el foco de estudio se realiza sobre el ámbito de las ciencias de la computación. Las plataformas que el ámbito actualmente presenta relacionadas con el proyecto, y que se estudiarán a continuación son las siguientes: dblp, Google Scholar, y ResearchGate.

2.1. Dblp

La bibliografía de dblp es la referencia online para consultar información bibliográfica acerca de publicaciones relacionadas con las ciencias de la computación. Dblp ha evolucionado mucho desde su primera versión, donde ofrecía un pequeño servidor web experimental, hasta la última, la cual cuenta con un servicio open-data disponible para toda la comunidad. La misión de dblp [4] es la de apoyar a los investigadores de ciencias de la computación en sus esfuerzos diarios proveyéndolos con acceso gratuito a una bibliografía gratuita y de la mayor calidad posible.

El servidor de dblp [5] cuenta con más de 4.4 millones de publicaciones y 2.2 millones de autores, representados en la Figura 2. La librería aumenta año tras año de manera considerable, y permite a los investigadores obtener información sobre los distintos tipos de publicaciones de las que disponen: libros, conferencias, artículos, tesis de máster, tesis doctoral, referencias, y publicaciones informales. Además, dblp permite consultar información en tiempo real sobre otros campos relevantes, como son el número de autores por publicación, el número de coautores por publicación, las publicaciones en una conferencia, las publicaciones en una revista, etc.

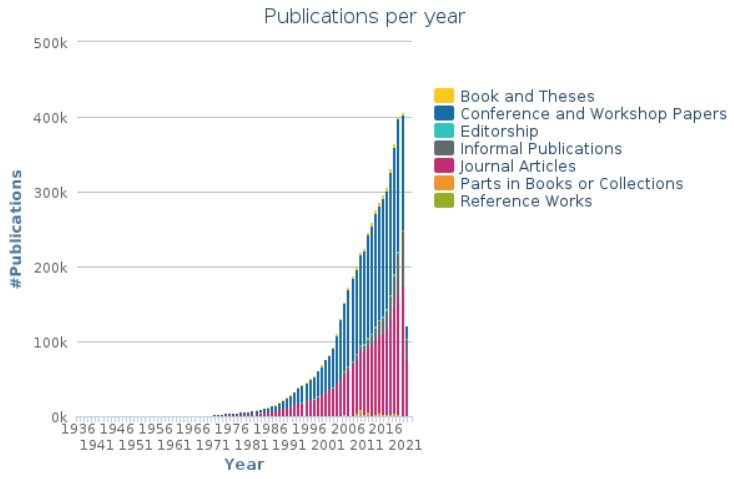


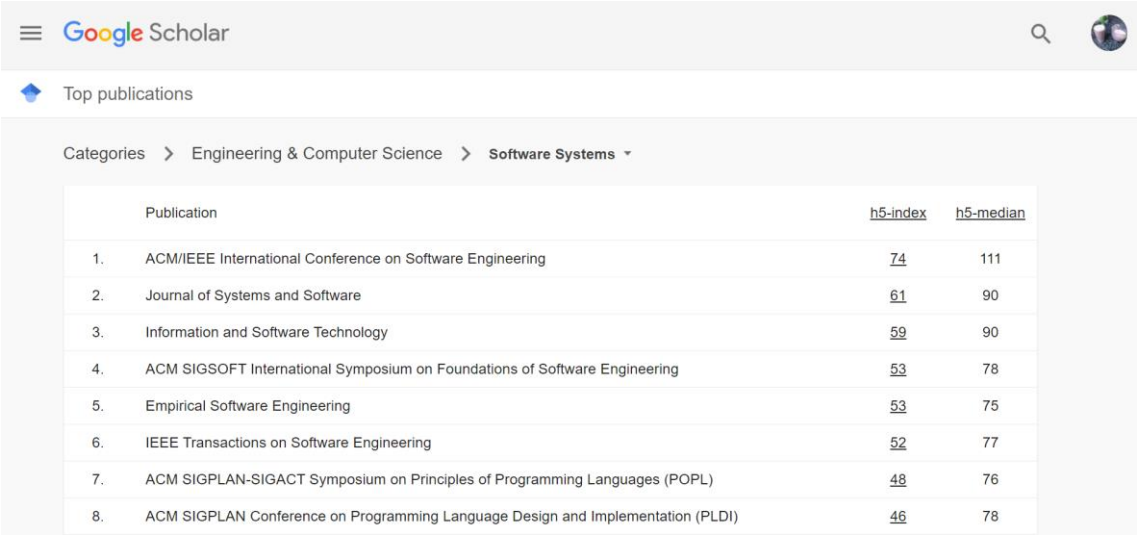
Figura 2. Publicaciones anuales registradas por dblp

Sin embargo, la cualidad más significativa que posee dblp es la de ofrecer a sus clientes la posibilidad de obtener toda la información disponible de las publicaciones, ordenándolas a partir de su fecha de publicación, revista, tipo de publicación y nombre de autores.

2.2. Google Scholar

Google Scholar [6] es otra herramienta que permite buscar información sobre literatura académica, aunque no solo se centran en las ciencias de computación. La plataforma ofrece buscar una gran cantidad de datos, incluyendo artículos, tesis, libros y resúmenes provenientes de publicaciones científicas, repositorios online, universidades y muchos otros sitios web. Una de las diferencias entre dblp y Google Scholar es que esta última permite obtener documentos enteros para poder consultarlos en tiempo real.

Otra de las funcionalidades que Google Scholar ofrece es la de ordenar documentos en base al índice h, que mide la calidad de las publicaciones. Cuando una publicación se sube al repositorio Google Scholar le da un valor basándose en el texto del documento, donde se publica, quien lo escribe, y cada cuando se cita en otras publicaciones. La asignación de una puntuación a una publicación permite a la plataforma ofreces rankings sobre las distintas revistas y conferencias que existen en sus muchos ámbitos de estudio. Un ejemplo de esta funcionalidad es el ranking de sistemas software, donde se encuentran las conferencias y revistas mejor calificadas del sector, como se puede apreciar en la Figura 3 [7].



	Publication	h5-index	h5-median
1.	ACM/IEEE International Conference on Software Engineering	74	111
2.	Journal of Systems and Software	61	90
3.	Information and Software Technology	59	90
4.	ACM SIGSOFT International Symposium on Foundations of Software Engineering	53	78
5.	Empirical Software Engineering	53	75
6.	IEEE Transactions on Software Engineering	52	77
7.	ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL)	48	76
8.	ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI)	46	78

Figura 3. Venues con mejor puntuación de Google Scholar

Finalmente, para concluir con el estudio de Google Scholar cabe resaltar el uso de perfiles que ofrece la plataforma. Al igual que en cualquier red social, permite a los autores crear un perfil con sus datos relevantes para luego poder mostrar todas las publicaciones a su nombre. Estos perfiles permiten conocer quien cita artículos ya publicados, el número de citaciones de un autor, las publicaciones a nombre del autor, su índice h, y la afiliación a una institución académica, entre

otros. Además, los perfiles públicos pueden ser encontrados a través del buscador como si de una publicación se tratara.

2.3. ResearchGate

ResearchGate [8], la última plataforma que se va a estudiar es una red social para científicos e investigadores. Con más de 20 millones de miembros de todo el mundo permite a sus usuarios compartir, descubrir y discutir sobre investigación. En definitiva, la plataforma surge para conectar el mundo de la ciencia y hacer que la investigación sea pública para todos.

ResearchGate permite a sus usuarios realizar una gran cantidad de operaciones, entre las que se encuentran compartir las publicaciones bajo el nombre del investigador, acceder a millones de ellas, conectar y colaborar con otros profesionales, obtener estadísticas de quién y cuánta gente lee una publicación, y finalmente, permitir generar foros de discusión para plantear y resolver problemas. La Figura 4 [9] presenta la página principal de la web de ResearchGate.

La diferenciación que esta plataforma logra con respecto a las otras dos es sin lugar a dudas la de poder realizar y buscar preguntas. A través de varios filtros, como pueden ser lenguajes de programación, frameworks, o incluso palabras claves, los investigadores pueden buscar información sobre temas específicos sin tener que buscar en decenas de artículos al respecto. Además, ResearchGate ofrece también la oportunidad de buscar empleos relacionados con las aptitudes profesionales de cada uno. En conclusión, no solo permite encontrar y publicar artículos, sino que también sirve para unir a la comunidad científica en un solo sitio web.

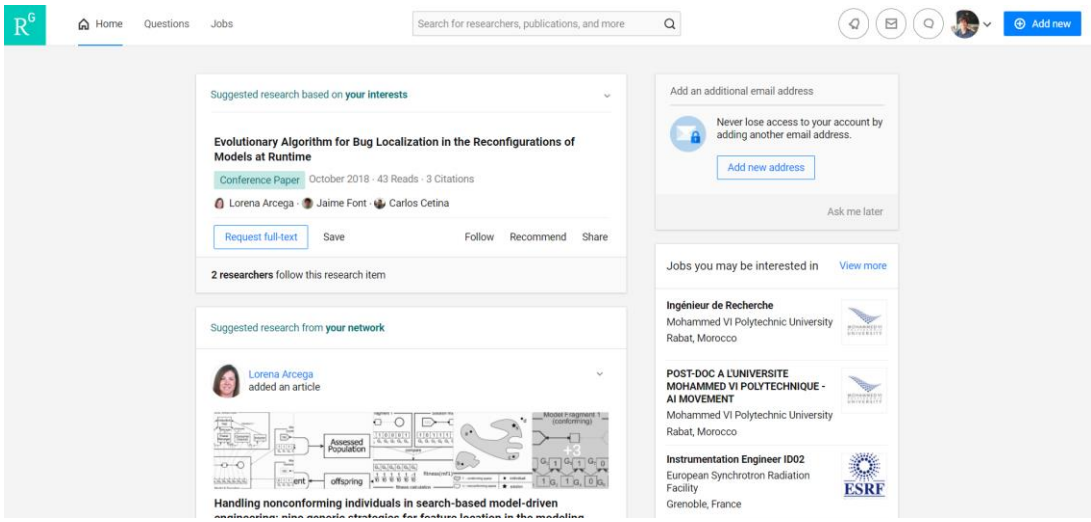


Figura 4. Página principal de ResearchGate

2.4. Comparativa

Tras la introducción a las diferentes plataformas relacionadas con el ámbito de las ciencias de la computación se puede apreciar que a pesar de la gran funcionalidad que ofrecen estas

herramientas, sus características específicas hacen que se deba hacer uso de múltiples plataformas para abarcar todas las necesidades de un investigador.

Mientras que dblp se centra en servir datos sobre las publicaciones a los investigadores, no permite obtener estadísticas sobre ellas. Google Scholar sí que permite darles una puntuación a dichas publicaciones, pero no alcanza la funcionalidad que podría presentar a la comunidad. Como por ejemplo creando rankings de publicaciones dentro de cada ámbito, y de esa manera asociarlos a los investigadores para obtener un ranking más completo. Finalmente, ResearchGate permite conectar con otros investigadores y buscar artículos o conferencias, pero no deja de ser una red social.

Easy Scholar Rank tiene como objetivo cubrir una necesidad que a día de hoy ninguna de las grandes plataformas considera, la creación de rankings de investigadores a partir de las conferencias donde publican, y la obtención de estadísticas personales sobre dichos investigadores.

Los objetivos de desarrollo de la funcionalidad se especifican en el siguiente capítulo, donde serán introducidos, relacionados entre ellos, y profundizados para comprender la necesidad de la que surgen y como en su totalidad forman el proyecto de Easy Scholar Ranking.

3. Objetivos

Para introducir el apartado en el cual se detallan los objetivos del proyecto es preciso reiterar, que tal y como se comenta en la introducción, la misión del proyecto es proporcionar a la comunidad de investigadores de las ciencias de la computación una herramienta para poder consultar de forma sencilla información relevante sobre su trabajo y campo de estudio.

El proyecto tiene como objetivo principal el desarrollo de un sistema o aplicación web capaz de ordenar investigadores dentro del ámbito de ciencias de la computación. A través de la aplicación, un investigador debería de ser capaz de filtrar y ordenar investigadores de una forma sencilla.

Para construir la aplicación web, y como es común en el desarrollo de software, se subdivide el objetivo principal en tareas más pequeñas y concretas. Estas tareas pueden ser clasificadas dependiendo de su naturaleza, entre las que encontramos: tareas de estudio y análisis, tareas de diseño, y finalmente, tareas de implementación. A continuación, se muestran las diferentes tareas necesarias para formar el proyecto dentro de sus correspondientes bloques de trabajo.

Tareas de estudio y análisis

- Estudiar fichero XML con datos de dblp
- Buscar frameworks de desarrollo de la aplicación
- Estudiar servicios en la nube para desplegar la aplicación

Tareas de diseño

- Crear modelo de dominio de la base de datos de dblp
- Crear modelo de datos para leer consultas de la api de dblp
- Diseñar interfaz de usuario

Tareas de implementación

- Obtener lista de revistas y conferencias de Google Scholar
- Obtener datos de la api de dblp
- Generar rankings por año y conferencia/revista
- Generar rankings por intervalo de años
- Generar rankings a partir de la universidad de un investigador
- Generar rankings usando la lista de revistas de Google Scholar
- Generar rankings de universidades
- Desplegar la aplicación web en un servicio en la nube

El cumplimiento de todas las tareas concluye en la creación de la aplicación web Easy Scholar Ranking, que permite a sus usuarios crear rankings para visualizar el posicionamiento de investigadores y obtener estadísticas dentro de revistas o conferencias de ciencias de la computación. De esa manera cubriendo la necesidad básica que se introduce en capítulos anteriores.

4. Metodología

La metodología hace referencia al conjunto de procedimientos que se siguen para alcanzar los objetivos de una tarea, proyecto o cualquier investigación de origen científica. Es la herramienta que hace que el desarrollo del proyecto siga siempre un curso estable y controlado.

Easy Scholar Ranking cuenta con un tutor académico y un cliente, lo que hace que el desarrollo del proyecto tenga siempre en cuenta las especificaciones y opiniones de este último. Para ello, se acuerda entre la tutora del proyecto y el alumno la periodicidad de las reuniones, el tipo de metodología y demás aspectos relevantes al progreso y seguimiento del desarrollo.

Dada la naturaleza del proyecto se elige utilizar características propias de metodologías ágiles [10], las cuales permiten adaptar la forma de trabajo a las condiciones del proyecto consiguiendo flexibilidad e inmediatez en la respuesta para amoldar el proyecto y su desarrollo a las circunstancias específicas del entorno. Las características del proyecto y la metodología de la cual se extraen son las siguientes:

Extreme programming XP

- Retroalimentación concreta y frecuente por parte del cliente
- Diseño simple
- Versiones pequeñas
- Entregas regulares y parciales del producto final
- Reuniones de planificación, revisión y retrospectiva
- Desarrollo en forma de sprints con duración máxima de 1 mes

El uso de una metodología como XP hace que el proyecto pueda hacer uso de las ventajas que ofrece el desarrollo ágil. Entre ellas se encuentran la mejora de la calidad del producto en base a la integración, comprobación y mejora continua del mismo. Otra de las ventajas es la satisfacción del cliente, que se ve involucrado en el proceso y puede vivir en tiempo real el desarrollo y evolución del proyecto. La gestión ágil [11] permite además ejercer un mayor control sobre el trabajo en base a la continua revisión y adaptación de este. Finalmente, la gestión ágil incrementa potencialmente la posibilidad de éxito del proyecto, ya que los errores se identifican a lo largo del proceso de desarrollo, haciendo que el gasto de recursos y capital sea mínimo en comparación a otro tipo de gestiones.

La metodología obtenida a partir de la combinación de características propias de XP se adapta a través de la división del desarrollo del proyecto en forma de iteraciones. Estas iteraciones implementarán objetivos pequeños y concretos que se deberán completar a lo largo de un periodo máximo de 1 mes. Además, previo a cada ciclo de desarrollo se planificará una reunión en la cual se revisarán los objetivos completados hasta el momento y se decidirán las implementaciones necesarias para completar la iteración. Finalmente, cabe decir que dichos objetivos se decidirán

a partir de tanto la retroalimentación del cliente y de la tutora del proyecto, quienes decidirán junto al alumno la forma adecuada de desarrollar el proyecto de Easy Scholar Ranking.

La planificación estimada del proyecto tiene una duración de 5 iteraciones con los siguientes objetivos en cuenta, ordenados en base a su prioridad para construir el producto final:

Primera iteración

- Estudio del fichero XML de dblp
- Creación del modelo de dominio de la base de datos de dblp
- Comparación de la obtención de datos entre el fichero XML y el servicio web de dblp

Segunda iteración

- Búsqueda de frameworks de desarrollo
- Diseño de la interfaz de usuario
- Generación de un ranking a partir de un año y revista

Tercera iteración

- Generación de un ranking a partir de un intervalo de años
- Generación de un ranking a partir de la universidad de un investigador

Cuarta iteración

- Obtención de la lista de revistas y conferencias de Google Scholar
- Generación de un ranking usando la lista de revistas de Google Scholar
- Generación de un ranking de universidades

Quinta iteración

- Búsqueda de servicios en la nube para desplegar la aplicación
- Despliegue de la aplicación web

La planificación se pone en práctica en el capítulo siguiente, donde se explica la implementación de la solución del proyecto. Esta planificación no es final, y podrá verse modificada a lo largo del capítulo de desarrollo.

5. Diseño e implementación

El capítulo de desarrollo explica cómo se ha diseñado e implementado la solución del proyecto. Para poder transmitir al lector de mejor manera el curso de este, se divide el capítulo en subapartados correspondientes a las diferentes iteraciones que ha habido. En ellos se discute sobre los objetivos o tareas propuestas en las reuniones y como se enfocan, analizan, y desarrollan, de esa manera pudiendo comprender el flujo de implementación del proyecto de la mejor forma posible.

5.1. Primera iteración – 29 días

El primer contacto con el proyecto se da a través de una reunión con la tutora, María Francisca Pérez, y el cliente, Carlos Cetina. En esta reunión introductoria se discute sobre la necesidad que cubre el proyecto, crear rankings para clasificar investigadores dependiendo de su campo. Para ello se utiliza como ejemplo la plataforma de EasyChair, que permite gestionar conferencias de forma sencilla. Una vez comprendida la idea detrás de la cual surge el proyecto se procede a hablar sobre las formas en las que podría afrontarse el desarrollo del mismo. Una de ellas es a través de la plataforma de dblp, la cual ofrece una api para realizar consultas. La otra, que también procede de dblp, trata de un fichero XML en el cual se encuentra toda la información de la base de datos del servidor. Ambas formas de afrontar el proyecto comienzan con la comprensión y análisis del fichero, el cual permite entender el modelo relacional de datos detrás de una de las mayores plataformas del ámbito de las ciencias computacionales. Finalmente, se concluye la reunión con la especificación de las tareas que se deben completar durante la iteración:

- Estudio del fichero XML de dblp
- Creación del modelo de dominio de la base de datos de dblp
- Comparación de la obtención de datos entre el fichero XML y el servicio web de dblp

5.1.1. Estudio del fichero XML de dblp

La primera de las tareas del proyecto consiste en el análisis del fichero XML que se encuentra en la página web de dblp. Este fichero cuenta con todas las entradas y tablas de la base de datos de la plataforma, lo cual resulta ideal para comprender como se distribuyen los datos en ella.

Para afrontar la tarea se estudian primero las formas en las que se puede manipular un fichero XML. Una de ellas es a través de parsers en Java, los cuales se usan para leer los ficheros de este estilo. Tras una búsqueda sobre esta herramienta se obtienen 5 parsers distintos que pueden resultar útiles para el proyecto, y que son: DOM, SAX, JDOM, XPath y DOM4J. Todos ellos funcionan de manera distinta, algunos son más rápidos en la lectura, otros en la escritura y hay alguno que crea estructuras de datos en forma de árbol. Para comprobar cuál es el que mejor

funcionaría en el proyecto se prueba a mostrar por consola los distintos tipos de publicaciones que existen en el fichero y cuantas veces aparecen.

La primera opción es JDOM, el estándar oficial de W3C [12], una comunidad internacional que desarrolla estándares que aseguran el crecimiento de la Web a largo plazo. Tras probar este parser se llega a la conclusión de que usa demasiada memoria para crear el árbol de nodos, lo cual hace que la memoria RAM llegue a su máximo y la aplicación colapse. Esto hace que se descarte. La siguiente opción es SAX, con la cual pasa exactamente lo mismo. También se prueba con el resto de parsers, pero el resultado es siempre el mismo.

Tras el fracaso del uso de parsers se propone usar aplicaciones que permiten crear y modificar bases de datos con tecnología XML. El primer programa que se usa es eXist [13], un proyecto de software de código abierto especializado en bases de datos NoSQL. Para entender bien el programa se intenta simular la serie de ejemplos de funcionamiento que ofrece la página web. Sin embargo, a la hora de realizar estos ejemplos ocurre un error ya que el fichero de dblp, 3GB, ocupa mucho más que el máximo límite permitido de 100MB.

Otra motor de bases de datos recomendado es BaseX, el cual cuenta con el apoyo de W3C y permite trabajar con ficheros de cualquier tamaño de forma sencilla. BaseX funciona con XQuery [14], un lenguaje de consultas diseñado para colecciones XML con una gran cantidad de ejemplos y documentación en la red. Esto hace que se elija como plataforma de trabajo para realizar las consultas y conocer en qué manera se distribuye el modelo de datos de dblp.

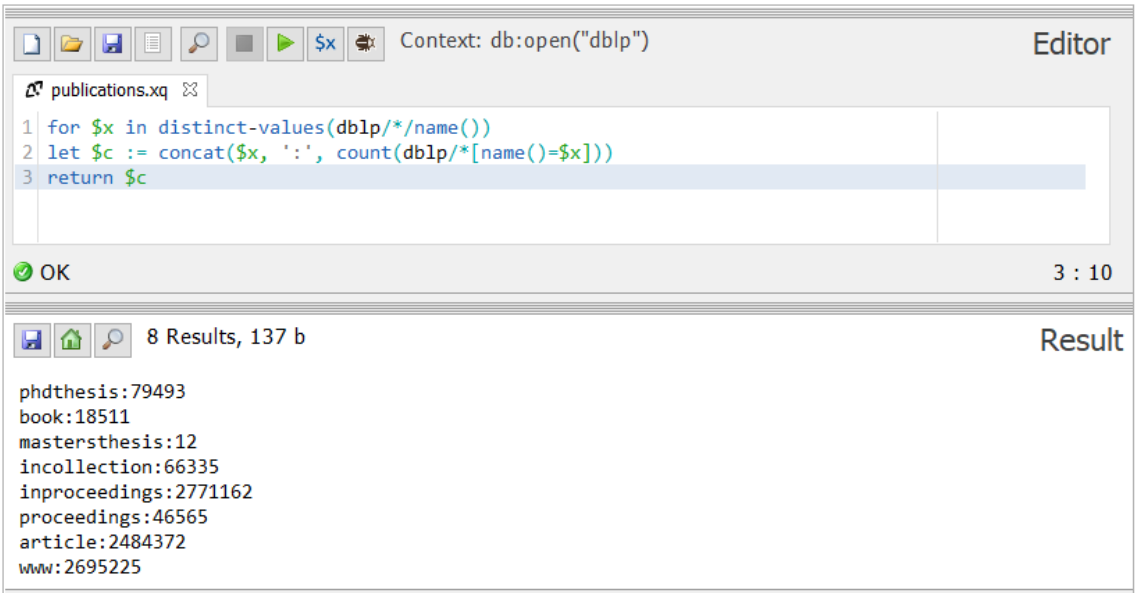


Figura 5. Consulta de datos en BaseX

Con BaseX se puede empezar a analizar la base de datos y comprender como se distribuyen las publicaciones, autores y otros datos relevantes. Por ejemplo, en la Figura 5 se pueden ver los

distintos tipos de publicaciones que existen acompañadas con el número de veces que aparecen en el documento. Si se distribuyen en forma de gráfico los datos obtenidos son:

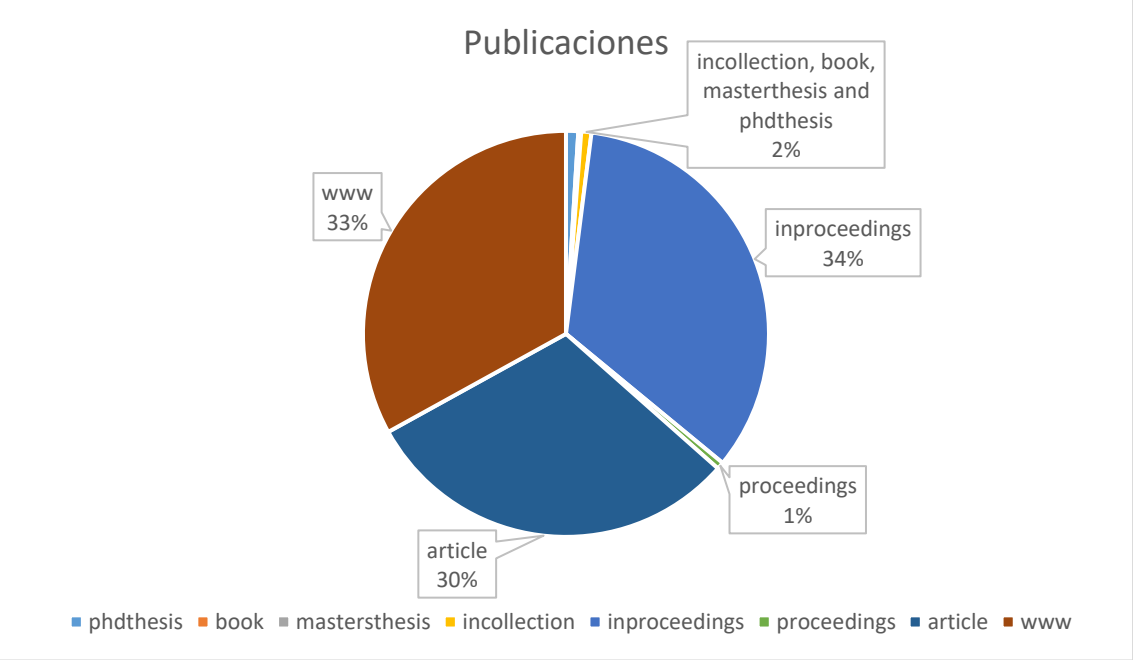


Figura 6. Distribución de publicaciones de dblp

Analizando la Figura 6, la cual muestra la distribución de publicaciones de dblp, se puede observar que existen 3 grupos de publicaciones que engloban la mayoría de los datos del fichero XML, siendo estos 'www', 'article' e 'inproceedings'. Comparando los datos obtenidos de la consulta con la información proveniente de la página web puede deducirse de forma sencilla a qué hace referencia cada grupo:

- Inproceedings: conferencias
- Article: artículos
- Www: información relacionada con el autor

Si de la misma forma que se han obtenido estos datos se realizan consultas para obtener información relacionada sobre cada tipo de publicación, se puede averiguar cuáles son los campos que componen cada una de estas agrupaciones:

<u>www:</u>

La mayoría de las propiedades de la tabla www aparecen en raras ocasiones, los únicos campos que suelen estar en todas las consultas son el nombre del autor y el título, que hace referencia a como se llama la página donde se encuentra información sobre el autor. Otro de los campos que puede resultar interesante es el de notas, donde puede encontrarse información sobre las instituciones académicas afiliadas al investigador. Sin duda una buena forma de afrontar el filtrado por universidad de los rankings. En la Figura 7 se puede observar las diferentes entrada de datos en la tabla www con el número de veces que aparecen el ella.

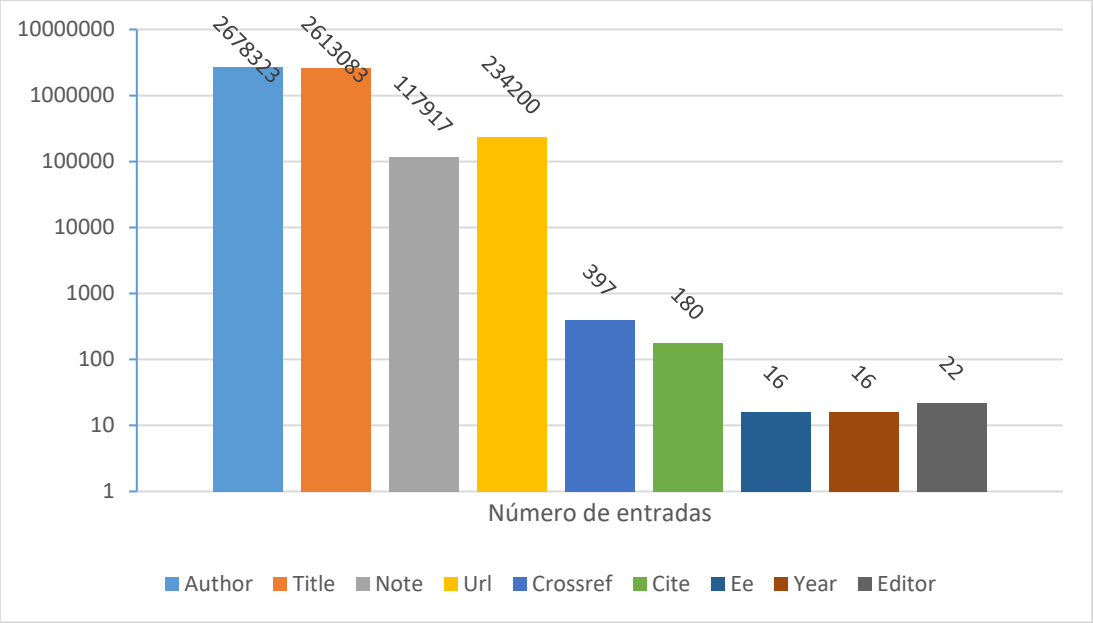


Figura 7. Entradas de datos en la tabla www

<u>Conferencias:</u>

La tabla de conferencias permite recuperar una gran cantidad de información relacionada con los autores y sus publicaciones, entre ellas se encuentran campos útiles como el nombre de la conferencia y libro en el que se publica, nombre de la publicación, el año y mucho más. Lo más importante de esta tabla es que al contrario que la de www parece que los datos de las propiedades suelen aparecer en cada una de las entradas, lo cual resulta de vital importancia en el desarrollo del proyecto. En la Figura 8 se puede observar las diferentes entrada de datos en la tabla conferencias con el número de veces que aparecen el ella.

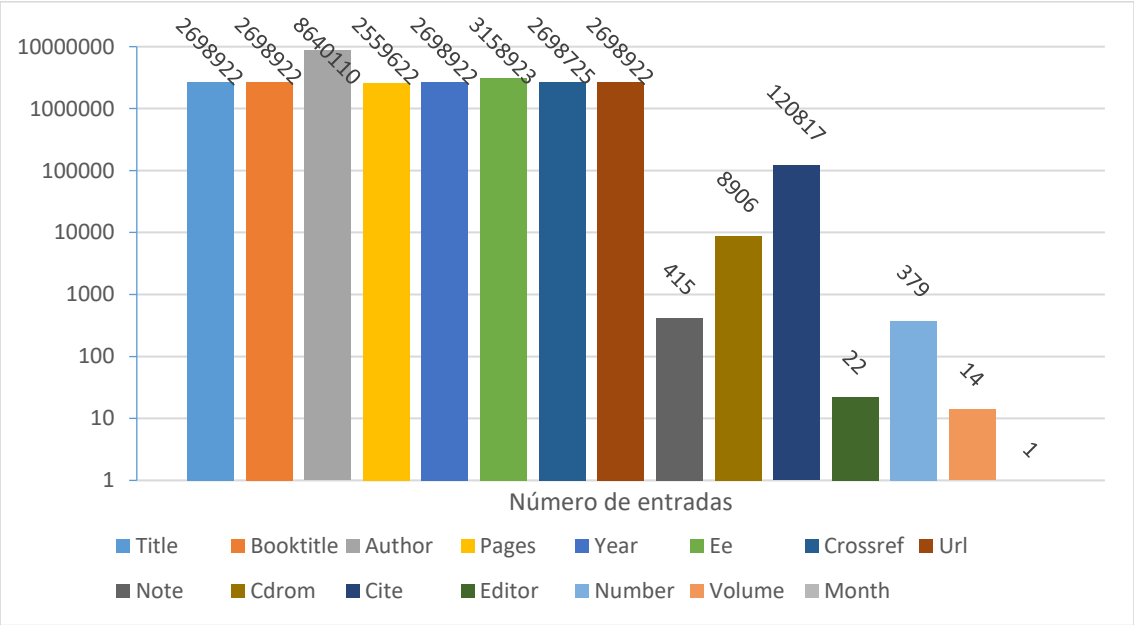


Figura 8. Entradas de datos en la tabla conferencias

Artículos:

De forma similar a la tabla de conferencias, la tabla de artículos presenta una gran cantidad de propiedades y entradas. La única diferencia entre las dos tablas es el contenido de la misma y a qué tipo de publicación hacen referencia. En la Figura 9 se puede observar las diferentes entrada de datos en la tabla artículos con el número de veces que aparecen el ella.

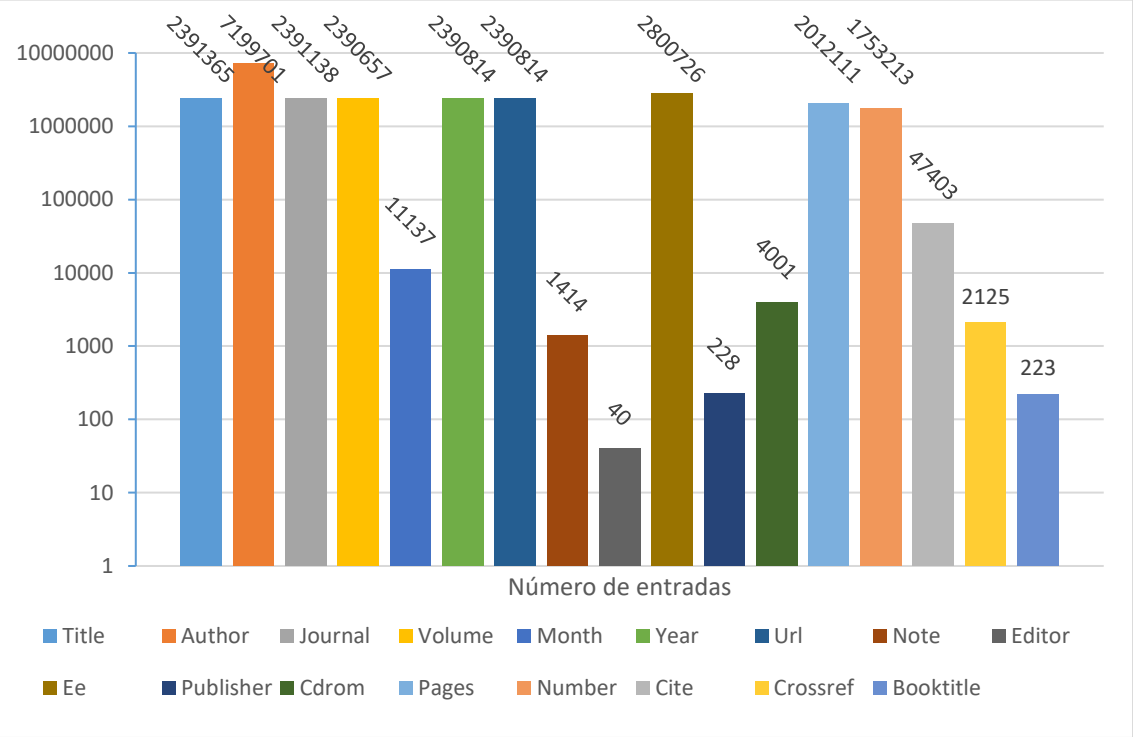


Figura 9. Entradas de datos en la tabla artículos

Tras comprobar las agrupaciones más importantes de datos se puede llegar a la conclusión de que la aplicación web de Easy Scholar Ranking debe centrarse solamente en obtener datos de artículos y conferencias, que ocupan un 64% del volumen total de datos y más del 95% de publicaciones que contiene el servidor de dblp.

Finalmente, habiendo comprendido la estructura de datos se puede crear el modelo de dominio de dblp, el segundo objetivo de la iteración actual.

5.1.2. Creación del modelo de dominio de la base de datos de dblp

Un modelo de dominio es un tipo de modelo conceptual del dominio que incorpora tanto el funcionamiento como los datos de un sistema. Para entender mejor cómo funciona la base de datos y cuáles son las relaciones entre las distintas entidades que la componen, se procede a mostrar el modelo de dominio de la base de datos XML de dblp:

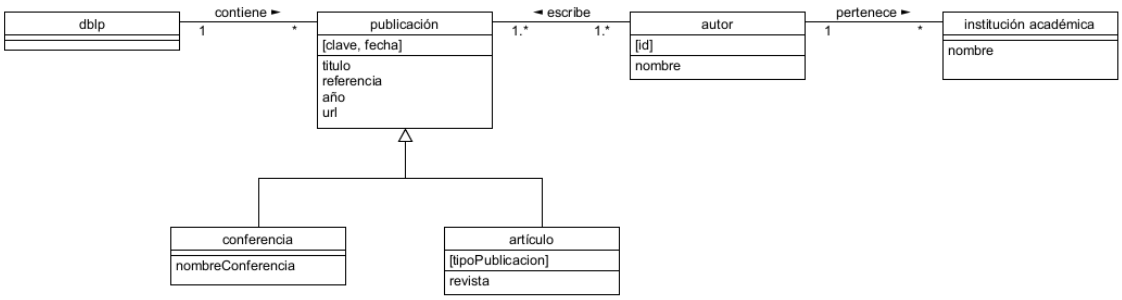


Figura 10. Modelo de dominio de dblp

En la Figura 10 se pueden observar las relaciones entre las distintas tablas que se encontrarían en la base de datos XML de dblp. Lo más importante del dominio es sin duda la relación entre autores y publicaciones, permitiendo a su vez asociar una publicación a una institución académica. Finalmente, tal y como se decide al completar la tarea anterior, se eligen los dos tipos más importantes de publicaciones, conferencias y artículos.

Tan sencillo como es, el modelo de dominio ayuda a comprender que es lo que pasa dentro de la base de datos, lo cual es realmente importante para poder comparar si esta forma de realizar consultas es más eficaz que a través del servicio web que proporciona la plataforma.

5.1.3. Comparación de la obtención de datos entre el fichero XML y el servicio web de dblp

El objetivo final de la primera iteración del proyecto consiste en la comparación de las dos herramientas que se presentan en la introducción de este para obtener datos sobre los investigadores. Para llegar a la mejor conclusión posible se prueba a obtener el mismo dato a través de las dos maneras, una publicación realizada por el cliente del proyecto, Carlos Cetina.

La primera forma de obtener la información deseada es a través de las consultas XQuery al fichero de datos. Tras realizar la consulta surge el primer inconveniente, existen caracteres que no se pueden recuperar, entre ellos acentos. Lo cual resulta un problema si una de las funcionalidades deseadas de Easy Scholar Ranking es poder obtener información de investigadores españoles. Otros inconvenientes surgen al considerar el uso de las consultas en la aplicación final, donde el sistema tendría que leer parámetros especificados por el usuario para luego realizar consultas. Con XQuery se deberían traducir los datos introducidos a dicho lenguaje, lo cual resultaría en una carga de trabajo demasiado elevada para el desarrollo de una funcionalidad. El último inconveniente de este tipo de consultas es que se realizan en local, y no en un servidor online. Esto haría que se necesitara de un servicio externo para servir las consultas, incrementando el coste de la aplicación. En definitiva, las consultas con XQuery resultan ideales para aprender sobre la arquitectura y funcionamiento de la base de datos, pero no para realizar dichas consultas en tiempo real desde una aplicación web.

A continuación, en la Figura 11, se puede observar el resultado de la consulta en la plataforma de BaseX:


```
<inproceedings mdate="2020-06-04" key="conf/caise/DomingoEPC20">
  <author>frica Domingo</author>
  <author>Jorge Echeverria</author>
  <author>Oscar Pastor</author>
  <author>Carlos Cetina</author>
  <title>Evaluating the Benefits of Model-Driven Development - Empirical Evaluation Paper.</title>
  <pages>353-367</pages>
  <year>2020</year>
  <booktitle>CAiSE</booktitle>
  <ee>https://doi.org/10.1007/978-3-030-49435-3_22</ee>
  <crossref>conf/caise/2020</crossref>
  <url>db/conf/caise/caise2020.html#DomingoEPC20</url>
</inproceedings>
```


Figura 11. Consulta de publicaciones en BaseX

La segunda forma de realizar consultas es a través de la api de dblp, la cual puede resultar algo difícil de usar en su primer contacto. Esta api contiene 3 servicios de búsqueda distintos, orientándose en la búsqueda de publicaciones, autores y escenarios. Esto permite simplificar las consultas en base a los distintos servicios para recibir información más específica. El único inconveniente que tiene la consulta de datos de la api es su poca flexibilidad, que viene explicada en la página web de dblp. Por suerte, este problema puede ser solucionado a través de la generación de un modelo de datos capaz de albergar los resultados que se obtienen, una de las próximas tareas a implementar. En conclusión, la api resulta una mejor opción a la hora de integrarla dentro de la aplicación web de Easy Scholar Ranking, ya que cubre todas las necesidades que se esperan de ella. Además, al estar en su propio servidor no hace falta tener que descargar un nuevo fichero XML cada mes para actualizar los datos, basta con llamar a la api que tendrá todas sus publicaciones actualizadas.

A continuación, en la Figura 12, se puede observar el resultado de la consulta con la api de dblp:


```
▼<hit score="1" id="318311">
  ▼<info>
    ▼<authors>
       <author pid="17/4106">Francisca Pérez</author>
       <author pid="151/5269">Jaime Font</author>
       <author pid="151/5211">Lorena Arcega</author>
       <author pid="67/4958">Carlos Cetina</author>
     </authors>
     <title>Collaborative feature location in models through automatic query expansion.</title>
     <venue>Autom. Softw. Eng.</venue>
     <volume>26</volume>
     <number>1</number>
     <pages>161-202</pages>
     <year>2019</year>
     <type>Journal Articles</type>
     <key>journals/ase/PerezFAC19</key>
     <doi>10.1007/S10515-019-00251-9</doi>
     <ee>https://doi.org/10.1007/s10515-019-00251-9</ee>
     <url>https://dblp.org/rec/journals/ase/PerezFAC19</url>
   </info>
   <url>URL#318311</url>
 </hit>
```


Figura 12. Consulta de publicaciones en la api de dblp

Con la finalización del estudio de consultas concluyen las tareas especificadas para la primera iteración, lo cual supone organizar una reunión con la tutora para discutir y analizar los próximos pasos del proyecto.

5.2. Segunda iteración – 28 días

La segunda iteración comienza, tal y como se propone en la metodología, con una reunión de estudio con la tutora del proyecto. En esta reunión se repasan los objetivos del primer sprint, se estudian las decisiones tomadas y, finalmente, se proponen una nueva lista de tareas a completar. Los objetivos son los siguientes:

- Buscar frameworks de desarrollo para la aplicación
- Generar rankings por año y revista/conferencia
- Diseñar interfaz de usuario

5.2.1. Buscar frameworks de desarrollo para la aplicación

El desarrollo de la segunda iteración ocurre durante el periodo de tiempo que el alumno realiza prácticas de empresa curriculares en Inycom. En la empresa el grupo de trabajo que se asigna al alumno es el de desarrollo .Net, un framework de Microsoft que permite desarrollar aplicaciones de cualquier tipo de forma sencilla. Tras consultar con el tutor de prácticas cuales serían buenas herramientas para desarrollar una aplicación web, su respuesta es utilizar el framework ASP .NET MVC. ASP .Net es la parte de .Net especializada en el desarrollo de aplicaciones web , y MVC es un patrón de arquitectura de datos que separa la lógica de control, los datos de la aplicación y la interfaz de usuario con el fin de desarrollar rápidamente la aplicación de forma modular y sostenible.

Tras un pequeño estudio del framework se decide elegirlo como plataforma de desarrollo de Easy Scholar Ranking en base a 3 factores. El primero de todos la familiarización del alumno en .Net gracias al desarrollo de las prácticas. El segundo es la masiva cantidad de ejemplos y documentación open source que posee el framework. Finalmente, la tercera razón es la constante actualización de la plataforma de desarrollo, lo cual hace que presente un gran nivel de rendimiento, llevándola ser un estándar web en muchas empresas. Esta serie de razones hace que se elija ASP .Net MVC para desarrollar la demo del proyecto, sin embargo, existe otro framework para ayudar al desarrollo de las vistas de usuario, uno de los puntos débiles del alumno. Este framework se conoce como Razor Pages, y facilita en gran manera el desarrollo de páginas web en modelos vista-controlador, similares al que se estudia anteriormente, por lo que también se introduce su uso en el proyecto, dando por finalizada la búsqueda de frameworks.

5.2.2. Diseñar interfaz de usuario

El último objetivo es la creación de la interfaz de usuario que mostrará las tablas de los rankings. Para ello se decide realizar unos bocetos simples en los que se aprecie la importancia de la tabla

en pantalla. También se diseña la pantalla principal de la aplicación, la cual el controlador servirá cuando el cliente acceda por primera vez.

Pantalla principal:

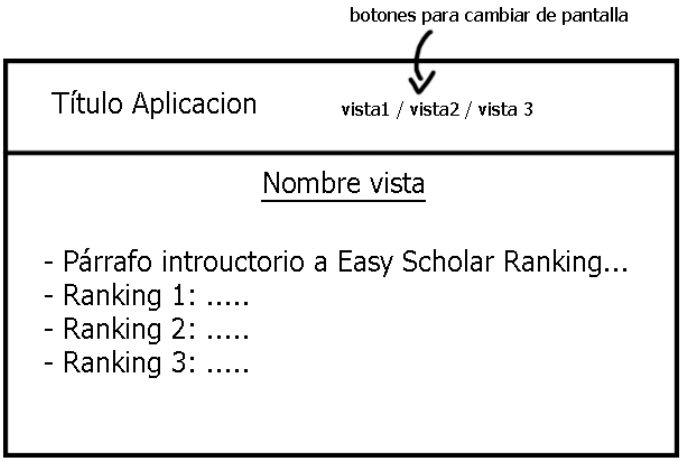


Figura 13. Diseño de la pantalla principal de Easy Scholar Ranking

La pantalla principal, cuyo diseño se muestra en la Figura 13, debe de contener solo la información necesaria para introducir la aplicación al cliente. Para ello se adopta un sistema de pestañas en la parte superior de la pantalla con las cuales el usuario puede saber cuáles son los distintos servicios que ofrece Easy Scholar Ranking. Por último, la pantalla principal debería describir brevemente la funcionalidad de la aplicación y de los diferentes rankings que se implementan en ella. En definitiva, fácil de usar y entender.

Pantalla de rankings:

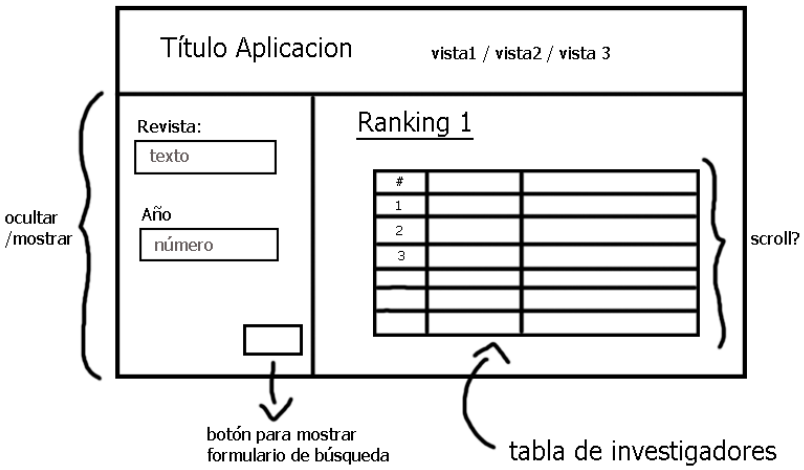


Figura 14. Diseño inicial de la pantalla de rankings

La pantalla de rankings, al igual de la principal cuenta con un diseño con los elementos importantes como protagonistas, en este caso la tabla. Para ello se diseña la pantalla de forma que con un botón se pueda abrir un formulario que aparecería a la izquierda. En el formulario se introducirían los datos necesarios para crear el ranking, el cual sería generado al cerrar el formulario. La tabla del ranking inicial contendría tres columnas: posición, nombre y número de publicaciones. Se propone también la opción de que la tabla sea desplazable para que la pantalla tenga siempre las mismas medidas y se pueda acceder a todos los elementos de la pantalla independientemente del número de entradas que haya en la tabla.

En definitiva, se prioriza la simplificación de la interfaz por encima de un diseño vistoso, haciendo un guiño al nombre de la aplicación, que es Easy. La Figura 14 presenta el diseño inicial de la pantalla de rankings.

5.2.3. Generar rankings por año y revista/conferencia

Después de haber encontrado los frameworks necesarios para la implementación, la siguiente tarea es la de crear un prototipo de aplicación. Para ello se debe crear una pantalla en la que se muestre un ranking conteniendo información sobre investigadores en base a dos parámetros distintos, año y nombre de la revista/conferencia. La obtención del ranking presenta el siguiente flujo de trabajo:

1. Filtrar publicaciones en la api por año y nombre de la revista
2. Obtener número de publicaciones por autor
3. Ordenar autores en base a su contribución numérica

Para conseguir crear el ranking es preciso construir primero la arquitectura del proyecto, que a pesar de estar basada en MVC, presenta algunos cambios debidos al uso de Razor Pages y a la consulta del servicio web de dblp. La Figura 15 presenta el diagrama de dicha arquitectura.

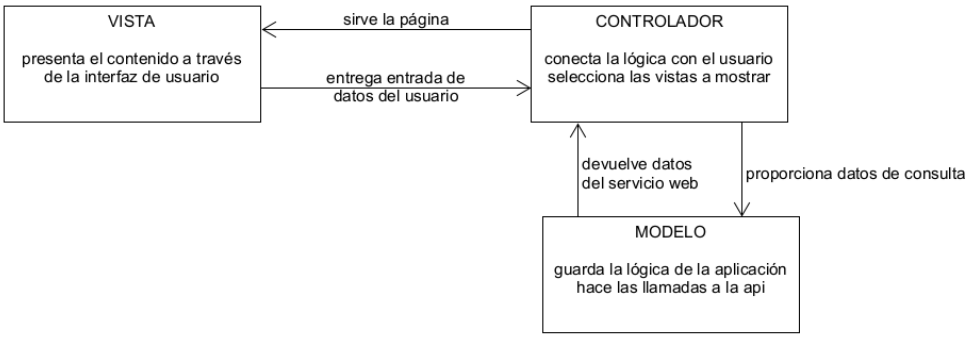


Figura 15. Diagrama de la arquitectura MVC de Easy Scholar Ranking

Para comprender bien el sencillo diagrama, es preciso conocer las responsabilidades de cada uno de los componentes [15].

Modelo

Representa el estado de la aplicación y cualquier lógica de negocio u operaciones que puedan ser llevadas a cabo por esta. La lógica se encapsula en el modelo, junto a cualquier implementación que modifique el estado de la aplicación. En Easy Scholar Ranking el modelo se encarga de mantener todas las clases usadas para ayudar en la lectura, manipulación y consulta de datos.

Vista

Es el componente responsable de presentar el contenido de la aplicación a través de la interfaz de usuario. Usa componentes como Razor Pages para incorporar lógica de .Net dentro de HTML. La lógica dentro de la vista es mínima, y se usa siempre para ayudar a la presentación de contenido.

Controlador

El controlador son los componentes que se encargan de la interacción con el usuario, la relación con el modelo, y finalmente, la selección de vistas para su posterior renderizado. La vista solo muestra la información, es el controlador quien controla la entrada de datos del usuario y las respuestas que debe pasar al resto de componentes. Es responsable de elegir el punto de entrada en la aplicación y seleccionar los modelos a mostrar.

Conociendo los componentes que llevan a cabo el funcionamiento de Easy Scholar Ranking se pueden realizar llamadas a la api de dblp para así conocer el modelo de datos de su respuesta. De los tres servicios que ofrece dblp se usa la búsqueda de publicaciones, con la cual se obtiene información sobre estas. Incluyendo investigadores involucrados, año de publicación y mucha más información relevante.

Para poder construir el modelo que permita obtener la lista de autores en una consulta se debe empezar por consultar el servicio de la forma más sencilla posible, que ya se dificultará en futuras iteraciones. Esta razón lleva a la creación de la consulta, la cual deberá obtener los datos sobre publicaciones correspondientes a la revista científica 'Information and Software Technology' en el año 2019, 'q=venue:Inf._Softw._Technol.:year:2019&format=json', con la cual se obtiene el siguiente modelo de datos:

```
"hit": [
    {
        "@score": "2",
        "@id": "607580",
        "info": {
            "authors": {
                "author": [
                    {
                        "@pid": "166/1051",
                        "text": "Tamer Mohamed Abdellatif"
                    },
                    {
                        "@pid": "c/LuizFernandoCapretz",
                        "text": "Luiz Fernando Capretz"
                    },
                    {
                        "@pid": "20/2202",
                        "text": "Danny Ho"
                    }
                ]
            },
            "title": "Automatic recall of software lessons learned for software project managers.",
            "venue": "Inf. Softw. Technol.",
            "volume": "115",
            "pages": "44-57",
            "year": "2019",
            "type": "Journal Articles",
            "key": "journals/infsof/AbdellatifCH19",
            "doi": "10.1016/J.INFSOF.2019.07.006",
            "ee": "https://doi.org/10.1016/j.infsof.2019.07.006",
            "url": "https://dblp.org/rec/journals/infsof/AbdellatifCH19"
        },
        "url": "URL#607580"
    },
```


Figura 16. Respuesta de una consulta a la api de dblp

Usando una herramienta como Postman se puede estudiar la respuesta del servicio, como se ilustra en la Figura 16. La respuesta manda un array de publicaciones, el cual a su vez contiene un objeto llamado authors con otro array de autores dentro. Una vez comprendida la respuesta se puede convertir en un objeto de tipo JSON para poder acceder a la lista de publicaciones contenidas en la propiedad 'hit'. Con la lista de publicaciones el siguiente paso es deserializar la información en un objeto creado para albergar los datos de una publicación:

```
4 references
class Authors
{
    0 references
    public List<AuthorClass> Author { get; set; }
}

1 reference
class AuthorClass
{
    0 references
    public string Text { get; set; }
}
```

Figura 17. Modelo de clases de Easy Scholar Ranking

En la Figura 17, que contiene el modelo de clases, se puede observar que la clase Authors contiene una lista de AuthorClass con la propiedad Text, que se usa para guardar el nombre de los investigadores. Aunque el nombre de las clases parezca confuso, se hace de esta manera para que el deserializado se realice de forma automática y sin ningún problema.

Si se realiza una consulta y se convierte la respuesta al modelo generado se obtiene una lista de todos los autores encontrados. Sin embargo, existe un problema cuando la publicación presenta un solo autor ya que, en vez de devolver un array de autores, devuelve un objeto con la información del autor dentro.

Para solucionar este problema se realiza una comprobación antes de deserializar el array de autores. En dicha comparación se cuenta el número de veces que aparece el campo 'Text' dentro de una publicación. Si el resultado es uno se deserializa el objeto como AuthorClass y se guarda dentro de la lista de una nueva instancia de tipo Authors. Por otra parte, si el resultado es distinto de uno se sigue la lógica mencionada anteriormente.

Con la lista de publicaciones generada se debe conocer la forma de diferenciar autores para poder contar cuantas veces aparecen en la lista. Por suerte, como se muestra en la Figura 18, cuando un nombre se repite dblp añade un número detrás de este, convirtiéndolo en la clave única necesaria para diferenciarlos.

```
{
    "@pid": "20/241-6",
    "text": "Muhammad Usman 0006"
}
```

Figura 18. Autor de dblp con número en el nombre

El último paso para obtener el ranking es iterar la lista de publicaciones para conseguir ordenarlos en base al número de veces que aparecen. Para ello se crea un mapa que usa como clave el nombre del autor y tiene por valor cuantas veces ha publicado en la revista. Obteniendo de esa manera el ranking de autores deseado para esta iteración:

Position	Name	Number of publications
1	Rajkumar Buyya	3
1	Chanchal K. Roy	3
1	Kevin A. Schneider	3
3+1	Xiapu Luo	2
4+1	Zhou Xu 0003	2

Venue: J._Syst._Softw. **Starting Year:** 2020 **Finishing Year:** 2020 **Number of entries:** 5 Search Ranking

Figura 19. Creación de un ranking en la primera iteración del proyecto

Con el ranking obtenido se puede concluir con la segunda iteración del proyecto, la cual establece el primer contacto con la api y abre las puertas a la creación de consultas más complejas que elevarán la funcionalidad y calidad del producto final de Easy Scholar Ranking. La Figura 19 presenta el ranking obtenido tras una consulta en la primera iteración del proyecto.

5.3. Tercera iteración – 22 días

Con la obtención de consultas y los datos del primer ranking obtenidos la reunión de planificación y revisión del progreso consiste principalmente en la introducción de nuevas funcionalidades y mejoras. Una de las principales mejoras a tener cuenta es la del aspecto visual de la aplicación de cara al usuario. En la anterior iteración se diseña la interfaz gráfica, pero no se llega a implementar completamente ya que la funcionalidad del ranking no es final. Relacionado con esto último se encuentra la representación de datos, siendo una de las mejoras propuestas por la tutora crear un gráfico para mostrar la información del ranking. Con estas mejoras se especifican los objetivos de este sprint:

- Mejorar visualmente el ranking
- Generar ranking por intervalo de años
- Mostrar estadísticas de una consulta

5.3.1. Mejorar visualmente el ranking

En la anterior iteración se diseña la pantalla de ranking, sin embargo, muchas de las ideas planteadas no llegan a implementarse. Además, en la reunión se habla sobre incluir algunas mejoras visuales para hacer que la aplicación proporcione una funcionalidad extra. Los cambios o mejoras son los siguientes:

<u>Tabla desplazable</u>

En el primer ranking que se genera solo se muestran las primeras cinco entradas de la tabla, sin embargo, en la implementación final se deberían de poder mostrar todas las entradas que el usuario especifique. Para ello es necesario que la tabla implemente la función de desplazamiento. Es decir, que siempre tenga la misma anchura y altura pero que el usuario se pueda mover a través de ella haciendo scroll con el ratón.

Esto requiere tomar una decisión de diseño en la aplicación, ya que como se comenta anteriormente, también se debe de mostrar una gráfica en la pantalla con las estadísticas de la tabla. Finalmente se realiza una modificación al diseño inicial de la pantalla de ranking, obteniendo así el segundo diseño de la pantalla de rankings, que se muestra en la Figura 20.

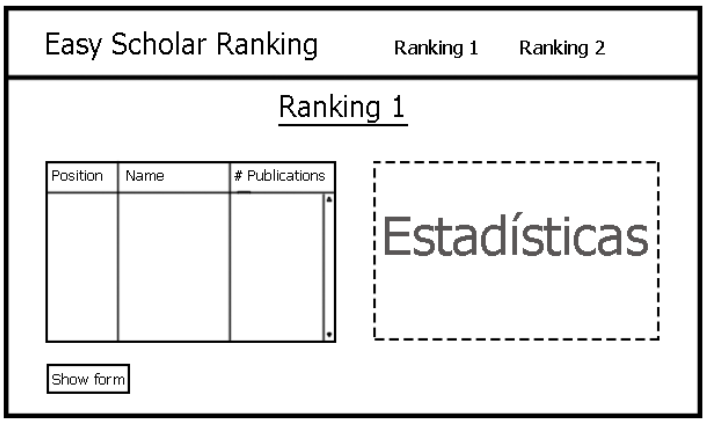


Figura 20. Segundo diseño de la pantalla de rankings

Este diseño se implementa haciendo uso de CSS en su mayoría, logrando obtener el resultado esperado sin ningún problema, como se muestra en la Figura 21.

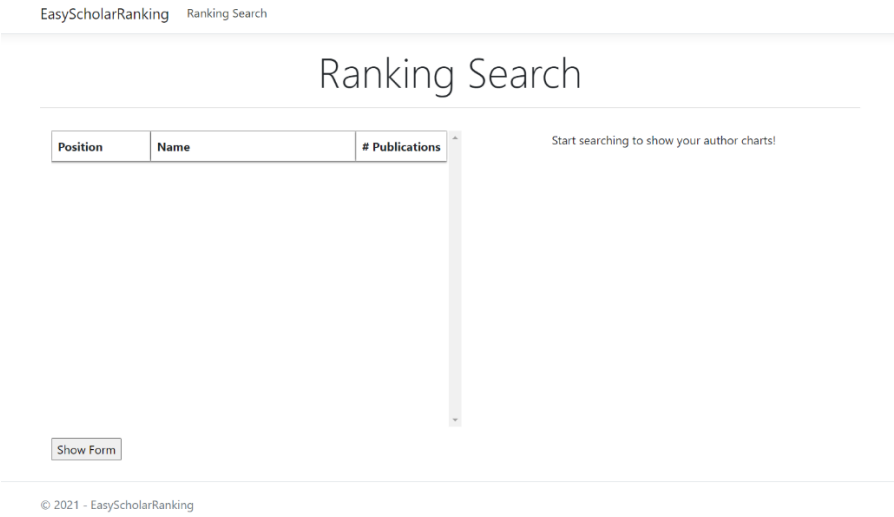


Figura 21. Implementación del diseño de la pantalla de rankings

Mostrar y esconder forma

Para hacer uso de todo el espacio, una de las primeras ideas de diseño es tener un botón para mostrar u ocultar la forma que utilizará el usuario para introducir los datos de búsqueda. Esta funcionalidad se logra gracias a Razor Pages, que permite generar y manipular código .Net dentro de HTML. Usando un booleano se puede mostrar u esconder el en el que se encuentran los datos de entrada. De la misma forma se cambia también el nombre del botón, logrando una sencilla funcionalidad de manera muy simple. El resultado de esta implementación se puede observar en la Figura 22.

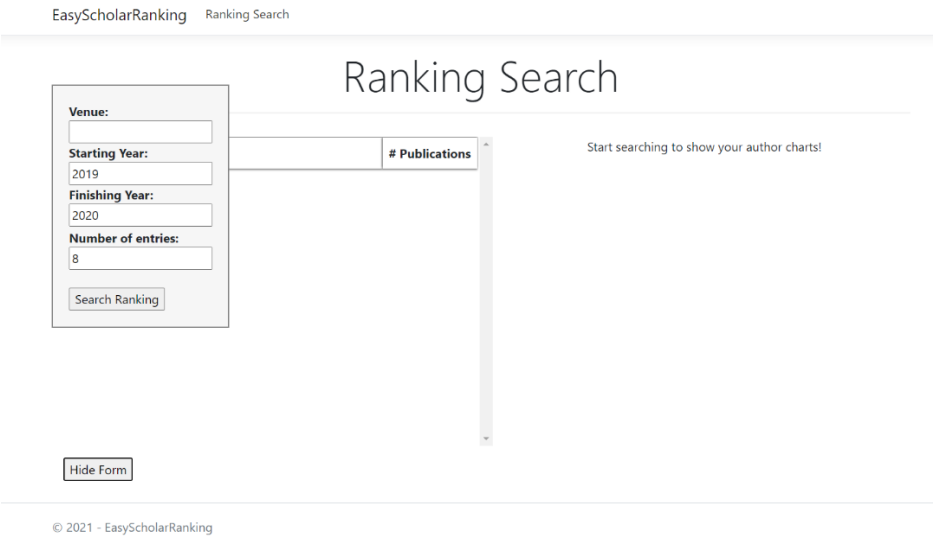


Figura 22. Pantalla de rankings con la forma de entrada de datos activada

Mostrar posiciones en la tabla

Para poder mostrar al usuario las posiciones de los investigadores en el ranking se elige utilizar el mismo formato que se usa en las competiciones deportivas. En estas cuando dos personas empatan en una posición, por ejemplo, el segundo puesto, a la siguiente persona del ranking no le corresponde la tercera posición, sino la cuarta, ya que es la cuarta persona con mejor puntuación de todas. Para implementar este funcionamiento basta con hacer uso de Razor Pages y asignar la posición de los investigadores una vez que se está formando la tabla comparando el número de publicaciones de un investigador con el anterior y así obteniendo el resultado deseado. La Figura 23 presenta el resultado y explicación de la clasificación de los autores del ranking.

Position	Name	# Publications
1	Fabio Palomba	3
1	Qing Wang 0001	3
1	Klaus Schmid	3
1	Silvia Teresita Acuña	3
1	Xin Xia 0001	3
1	Markku Oivo	3
1	Emilia Mendes	3
8	Tony Gorschek	2

Figura 23. Explicación visual de la clasificación de los autores

5.3.2. Generar ranking por intervalo de año

En la forma de la anterior imagen puede apreciarse que, al contrario que en la primera versión del ranking, los años de entrada se especifican en base al inicial y al final, es decir, un intervalo.

También puede observarse como se puede elegir el número de entradas de la tabla, ambas funcionalidades no implementadas pero que se pueden ver ya en la interfaz de usuario.

La implementación de esta última se logra también con Razor Pages, haciendo que los datos a mostrar solo se seleccionen en base a una variable especificada por el cliente.

Sin embargo, para introducir un rango dinámico de años en la búsqueda de investigadores es preciso modificar la lógica implementada hasta el momento, que se basa en la llamada a un método que realiza solamente una consulta al servicio web de dblp. La Figura 24 presenta el esquema del algoritmo implementado para obtener los datos de los autores a partir de un intervalo de años.

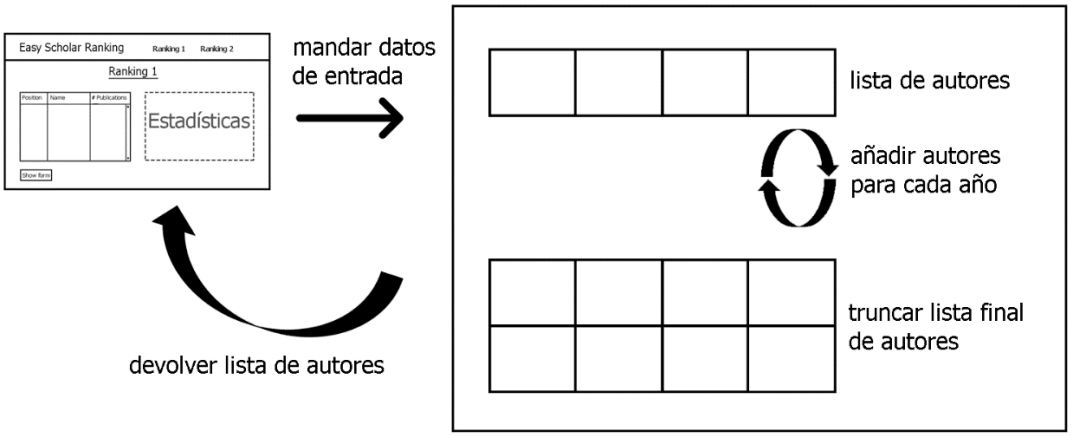


Figura 24. Esquema del algoritmo de obtención de datos en un intervalo de años

Esta nueva implementación de las consultas itera el funcionamiento anterior para añadir información a la lista de autores por cada año que se estipula en la forma. Esto implica que la variable que contiene toda la información se introducida en el método que incrementa los investigadores y sea devuelta cuando ya se hayan introducido en ella. Sin embargo, esta implementación genera un error que era muy difícil de generar en la anterior iteración. En anteriores versiones de dblp los datos del servidor tenían un modelo de datos distinto, lo que hace que puedan aparecer publicaciones antiguas con ningún autor dentro de ellas.

Para solucionar este problema se capturan las excepciones que se generan al no poder leer los objetos con el modelo que utiliza la aplicación. Una vez capturada si el sistema tiene publicaciones por leer sigue haciéndolo, y en el caso contrario, si la excepción se produce porque no hay entradas de datos en la respuesta, simplemente se devuelve la lista generada hasta el momento, incluso si permanece vacía. De esta forma el modelo es capaz de obtener de la base de datos toda la información relevante de forma exitosa.

La Figura 25 presenta una publicación sin autores obtenida a través de una consulta de dblp.

{{ "@score": "2", "@id": "5180780", "info": { "title": "Editor's corner - Software maintenance is a solution - Not a problem.", "venue": "J. Syst. Softw.", "volume": "11", "number": "2", "pages": "77-78", "year": "1990", "type": "Journal Articles", "key": "journals/jss/X90", "doi": "10.1016/0164-1212(90)90053-O", "ee": "https://doi.org/10.1016/0164-1212(90)90053-O", "url": "https://dblp.org/rec/journals/jss/X90" }, "url": "URL#5180780"}}

Figura 25. Publicación sin autores

5.3.3. Mostrar estadísticas de una consulta

Con la implementación de las consultas con un intervalo de tiempo el siguiente objetivo de la iteración es poder mostrar estadísticas acompañando a la tabla de datos. Una de las opciones que se puede implementar es a través de bibliotecas de jQuery, que funcionan muy bien de la mano de HTTP. Sin embargo, el proyecto hace uso de Razor Pages y la introducción de un nuevo framework de trabajo complica mucho la implementación. Para ello se busca diferentes herramientas de trabajo que funcionen con Razor Pages, y tras probar múltiples de ellas sin ningún resultado positivo se encuentra a Chart.js. Chart.js es una biblioteca de JavaScript especializada en crear y mostrar gráficos dinámicos que permiten interacción del usuario. Y, aunque la biblioteca este optimizada para JavaScript, funciona también en Razor Pages sin problema alguno.

Entre todas las opciones que permite usar la biblioteca, la gráfica que se elige es la circular, ya que puede servir para mostrar de forma muy clara la contribución de los autores en un ranking concreto. La implementación del gráfico es muy sencilla y se puede conseguir en poco tiempo usando la documentación de la página web de Chart.js [16].

El objetivo de la implementación del gráfico es mostrar como dentro de un ranking se distribuye el número de publicaciones de los investigadores. Para ello se comienza obteniendo la lista de autores de la consulta y guardando en un array las puntuaciones atribuidas a cada autor. Realizando estas simples instrucciones se pueden mostrar todos los investigadores en un solo gráfico circular. Esta implementación sirve cuando el número de investigadores es pequeño, es decir, cuando en el gráfico no está dividido en muchas entradas diferentes que dificultan la apreciación de información.

Para solucionar este problema se propone agrupar los autores que contribuyen por debajo de un valor en una sola entrada. Solo aparecerán en el grafico como entradas individuales los investigadores que:

1. su puntuación contribuya un valor por encima de un mínimo o,
2. su puntuación se repite por más de un investigador el porcentaje de autores repetidos este por encima de un valor máximo.

Este algoritmo ayuda a que los autores con puntuaciones más altas aparezcan en la tabla, y que si existen muchos autores con el mismo número de publicaciones se les pueda agrupar dentro de un mismo campo. Los valores usados para establecer el mínimo y máximo de la comparación se obtienen a partir de porcentajes de la tabla, como son la contribución de un valor de publicaciones sobre el total o la contribución de un autor dentro del total.

Con el algoritmo implementado los valores se consiguen leer sin ninguna dificultad y se aprecia claramente la contribución de los autores en el ranking, como se muestra en la Figura 26.

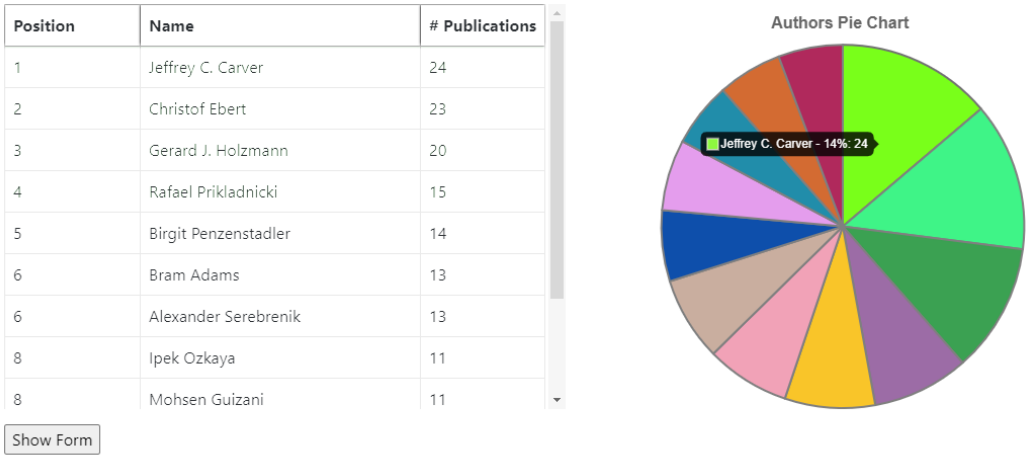


Figura 26. Implementación de un ranking junto a un gráfico circular

La implementación del gráfico concluye con la tercera iteración del proyecto, donde se corrigen y mejoran aspectos de la interfaz de usuario y se permite la consulta de rankings con un intervalo de años. Además, se logra proporcionar al usuario información visual de los rankings que se consultan, haciendo que Easy Scholar Ranking vaya adoptando una funcionalidad más completa, definida y profesional.

5.4. Cuarta iteración – 26 días

La reunión de la cuarta iteración se realiza el primer día después de las vacaciones de navidad. Un periodo con una gran carga de trabajo tanto para los docentes como para los alumnos. Por lo que se toma la decisión de realizar una iteración con una carga leve de trabajo basada en servir la aplicación web desde un servidor en internet. Es decir, hacer la aplicación pública para poder testearla y analizar los requerimientos implementados a lo largo de los sprints. Con el objetivo principal del sprint decidido se procede a la búsqueda de servicios en la nube para poder desplegar la aplicación web.

5.4.1. Estudiar servicios en la nube para desplegar la aplicación

Uno de los objetivos anteriores es la búsqueda de frameworks para desarrollar la aplicación, que resulta en el uso de una herramienta distribuida por Microsoft. Una de las razones detrás de la

decisión es la familiaridad con .Net gracias a las prácticas de empresa. De la misma forma, en las prácticas también se hace uso de servicios en la nube usando Microsoft Azure, que permite construir, desplegar y administrar aplicaciones en la nube de forma sencilla. El único problema que presenta Azure es el coste de mantener dichas aplicaciones activas. Sin embargo, Azure permite a sus usuarios probar la plataforma usando un bono de bienvenida de 200$, lo cual resulta ideal para testear Easy Scholar Ranking. Además, la plataforma de desarrollo de la aplicación, Visual Studio, permite conectar con los servicios en la nube de Azure de forma muy sencilla, desplegando aplicaciones con solo un clic. La Figura 27 presenta la página de bienvenida del portal de Microsoft Azure [17].

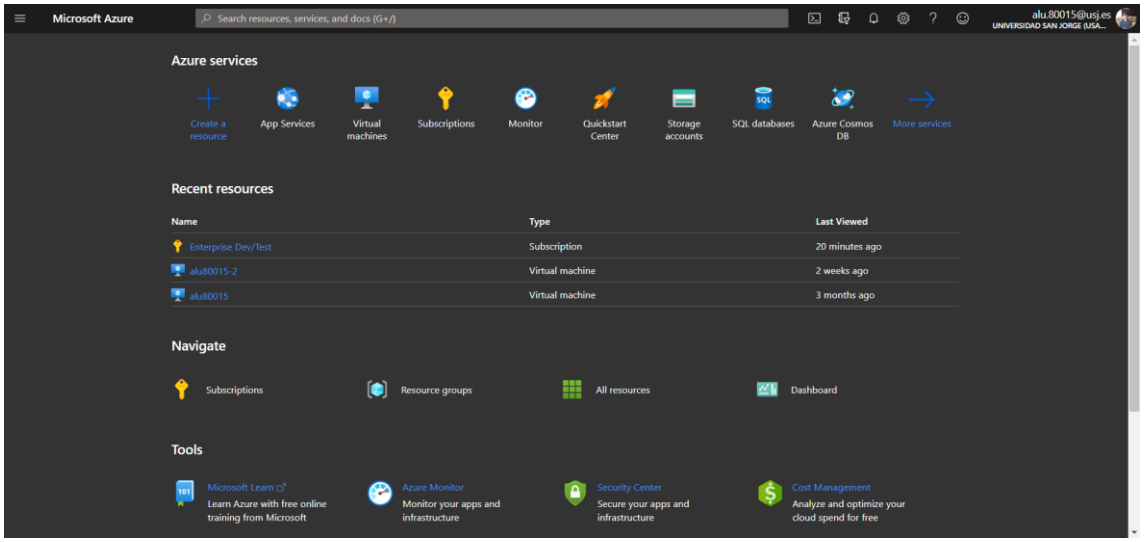


Figura 27. Pantalla de bienvenida del portal de Microsoft Azure

En conclusión, se elige desplegar la aplicación usando Microsoft Azure debido a la facilidad, familiaridad y compatibilidad entre el framework de desarrollo y la plataforma en la nube. El único inconveniente es que la prueba gratuita de los servicios dura 30 días, por lo que se deben estudiar otras opciones para conseguir desplegar la aplicación en futuros sprints.

Con la aplicación desplegada, tanto el cliente del proyecto como la tutora pueden observar el progreso y las funcionalidades que se han añadido a lo largo de las 3 primeras iteraciones de desarrollo. Tras testear la aplicación el cliente entrega una lista de mejoras y funcionalidades que podrían ser introducidas en Easy Scholar Ranking. Entre las mejoras se encuentra la tarea de hacer que la entrada de datos sea más sencilla para el usuario, ya que hasta ahora se debe introducir el nombre específico de las revistas en el formato que se usa para realizar las consultas a la api. Esto implica implementar un nuevo objetivo en la cuarta iteración antes de realizar el estudio de las funcionalidades del cliente.

5.4.2. Simplificar la entrada de datos de filtrado

Como bien se comenta en el párrafo anterior, la entrada de datos en el sistema resulta muy complicada para el usuario, lo cual difiere en gran manera con uno de los objetivos principales del proyecto, la construcción de una aplicación fácil de usar. La Figura 28 presenta la entrada de datos de una venue.

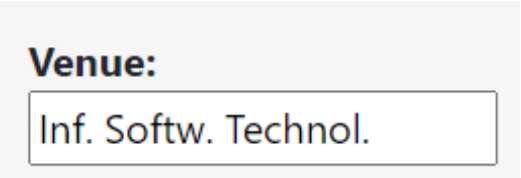


Figura 28. Ejemplo de entrada de datos compleja

Para solucionar el problema se hace uso de una de las herramientas propuestas por el cliente en la primera reunión informativa, la tabla de rankings de Google Scholar. Esta tabla contiene las 20 revistas y conferencias con la calificación más alta dentro del ámbito de ciencias de la computación. Por lo que se decide hacer uso de ella para que en vez de tener que escribir el nombre de la revista, el usuario pueda seleccionarla de dentro de un desplegable.

La implementación de esta funcionalidad se consigue a través de la creación de un fichero json compuesto por objetos que tienen tanto la clave usada para las consultas como el nombre completo de la revista. La búsqueda de claves se hace a través de la plataforma de dblp, donde a través del filtrado de revistas se puede llegar a obtener la clave usada en el servicio web. Una vez todas las claves son obtenidas basta con crear el fichero y hacer que los nombres de las revistas y conferencias se sirvan al usuario cuando este vaya a introducir el nombre de la conferencia o revista deseada. La Figura 29 presenta la mejora en la entrada de datos.

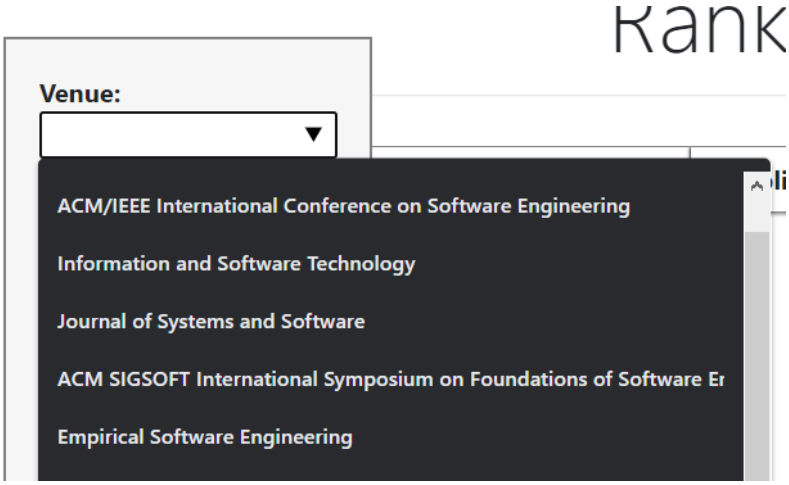


Figura 29. Entrada de datos simplificada a través del uso de una lista

Cuando el usuario realiza la consulta el sistema se obtiene la clave de la revista seleccionada a través del nombre de esta. Creando una pequeña capa de abstracción que ayuda a simplificar

notablemente la entrada de datos en el sistema. Sin embargo, esta funcionalidad hace que el usuario solo pueda buscar en 20 revistas/conferencias diferentes, lo cual resulta una gran limitación para el sistema. Para ello, se permite al usuario escribir el nombre de las revistas que desee, incluyendo un pequeño filtro para que no deba incluir también caracteres especiales.

```
{
  "name": "ACM/IEEE International Conference on Software Engineering",
  "key": "ICSE"
},
{
  "name": "Information and Software Technology",
  "key": "Inf._Softw._Technol."
},
{
  "name": "Journal of Systems and Software",
  "key": "J._Syst._Softw."
},
{
  "name": "ACM SIGSOFT International Symposium on Foundations of Software Engineering",
  "key": "SIGSOFT FSE"
},
{
  "name": "Empirical Software Engineering",
  "key": "Empir._Softw._Eng."
},
```

Figura 30. Objeto JSON de las venues de Google Scholar

En definitiva, con esta implementación se simplifica de gran manera la introducción de datos, pero a su vez se conserva la libertad de poder seleccionar las revistas y conferencias deseadas por el usuario, aprovechando así toda la funcionalidad que brinda el servicio web de dblp y concluyendo el desarrollo de la cuarta iteración del proyecto. En la Figura 30 se muestra el contenido del objeto JSON creado para realizar albergar los datos de las venues de Google Scholar.

5.5. Quinta iteración – 20 días

Durante el desarrollo de la cuarta iteración el cliente propone una serie de funcionalidades que deben ser estudiadas para considerar su implementación dentro de la aplicación web. Para poder comprender las decisiones tomadas en cuanto al desarrollo del proyecto se procede a mostrar la lista de funcionalidades entregada por el cliente:

1. Dada una revista y un periodo de tiempo, hacer un ranking de investigadores españoles que han publicado en esa revista. Tal vez DBLP tiene el país de cada investigador, o la universidad del investigador (en ese caso se puede hacer un listado de universidades españolas).
2. Dado un investigador español, una revista y un periodo de tiempo, indicar en qué posición del ranking de investigadores españoles se encuentra para esa revista en ese periodo.

3. Coger el listado de Google Scholar para un área de conocimiento y calcular el punto 2 para cada venue de esa área. Para no hacerlo con todos, hacerlo únicamente con ingeniería del software.
4. Dada una revista y un periodo de tiempo, calcular la media de papers que publican los investigadores en esa revista durante ese periodo de tiempo.
5. Dado un investigador, una revista, y un periodo de tiempo, calcular en qué percentil y cuartil se encuentra el investigador por papers publicados.
6. Coger el listado de Google Scholar para un área de conocimiento y calcular el punto 5 para cada venue de esa área. Para no hacerlo con todos, hacerlo únicamente con ingeniería del software.
7. Calcular para cada investigador la duración de su carrera. Para ello restar al año actual el primer año en el que tiene un paper. En los filtros de búsqueda (venue, inicio, fin, entradas) añadir uno nuevo para duración carrera.
8. Meto el nombre de un investigador y me da los resultados de 1 a 7 donde sale más favorecido.

Tras la lectura y análisis de las funcionalidades se discute con la tutora sobre las posibles implementaciones que se pueden realizar en el proyecto. La Tabla 1 presenta puntos de la lista del cliente a través de los cuales se obtienen los objetivos de la quinta iteración del proyecto.

Puntos	**Objetivos**
1 y 2	Filtrar investigadores por la localización de sus universidades
4	Obtener media de papers en un periodo de tiempo
3 y 6	Hacer filtrado usando todas las venues de Google Scholar
7	Filtrar investigadores por la duración de su carrera
5 y 8	Obtener estadísticas de un investigador

Tabla 1. Objetivos de la quinta iteración del proyecto

5.5.1. Obtener media de papers en un periodo de tiempo

Una forma de agregar algo más de información sobre los autores en el ranking es a partir de la obtención de la media de publicaciones en las que aparecen en el intervalo de tiempo seleccionado. Partiendo de la información que se obtiene en las consultas para crear la tabla se puede calcular fácilmente la media de publicaciones totales del ranking, la cual se añade al gráfico circular para que se muestre al usuario cada vez que realiza una consulta.

Sin embargo, solo mostrar un número no es suficiente para diferenciar autores con más o menos publicaciones, por lo que se propone pintar de un color las filas de los investigadores por encima

de la media de publicaciones del ranking. Para ello se crea un nuevo estilo con CSS para la tabla que se aplica cuando se crean las entradas de esta. Haciendo así distinción entre los investigadores por encima y por debajo de la media.

La Figura 31 presenta la pantalla de rankings con la diferenciación visual de autores en base a su media de publicaciones.

Position	Name	# Publications
1	Bogdan Vasilescu	8
2	David Lo 0001	7
3	Yang Liu 0003	6
3	Zhendong Su	6
3	Tien N. Nguyen	6
6	Gabriele Bavota	5
6	Massimiliano Di Penta	5
6	Gordon Fraser 0001	5

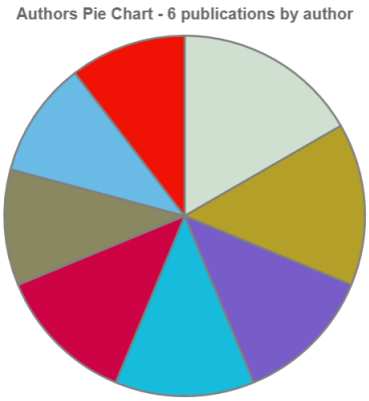


Figura 31. Diferenciación visual de autores en base a su media de publicaciones

5.5.2. Filtrar investigadores por la duración de su carrera

El segundo objetivo de la iteración consiste en filtrar investigadores en base al número de años que han estado en activo. La primera propuesta para implementar esta tarea consiste en calcular todos los años que un investigador ha estado activo, pero, al usar también el filtrado por intervalo de años, se acaba decidiendo que los años activos se calcularán dentro del intervalo. Es decir, si un investigador lleva 20 años publicando artículos, pero el filtrado que se realiza es en un intervalo de 5 años, el valor de años activo será de 5.

Para implementar esta funcionalidad se añade al modelo de autores unas propiedades para guardar el primer y último año en el que aparece un autor en la consulta. El valor del primer año se inicializa cuando se introduce el autor en la lista que se crea para un año. Una vez la lista final ha sido completada, se procede a recorrerla para crear el mapa de autores. En la creación del mapa se comprueba si un autor ha sido introducido y se actualizan los años de acuerdo a los valores establecidos anteriormente. De esta forma cuando se devuelve la lista ordenada con los investigadores se puede comprobar si la diferencia entre el primer y último año es superior a la establecida en la consulta, filtrando exitosamente los años activos de los investigadores.

5.5.3. Filtrar investigadores por la localización de sus universidades

Una de las primeras funcionalidades que se comentan cuando el cliente introduce el proyecto es la de poder filtrar investigadores en base a su nacionalidad o lugar de trabajo. Este objetivo se analiza en la reunión del quinto sprint, y tras una pequeña deliberación con la tutora se acaba

decidiendo que el filtrado basado en la afiliación a universidades de un país en concreto es la mejor forma de abordarlo.

En la primera iteración se realiza el estudio de la base de datos de dblp, y en ella se encuentra que existe una tabla con datos relevantes sobre la afiliación de los investigadores. Esta información procede del modelo de datos del fichero XML, pero no sirve para saber cómo se representa dicha información en el servicio web. Es por eso por lo que se utiliza la búsqueda de autores que incorpora la api de dblp. Con ella se buscan los datos de Carlos Cetina, el cual se sabe que está afiliado con la Universidad San Jorge. Realizando la búsqueda aparecen otros autores con un apellido similar, y entre todos ellos se encuentra uno con información relacionada con su afiliación. En la Figura 32 se muestra el resultado de obtener un autor con su afiliación definida dentro del objeto.


```
▼<result>
  ▼<query id="270247">
     <![CDATA[ Rengül* Çetin* Atalay* ]]>
   </query>
   <status code="200">OK</status>
   <time unit="msecs">1.03</time>
  ▼<completions total="1" computed="1" sent="1">
     <c sc="1" dc="1" oc="1" id="12665380">atalay</c>
   </completions>
  ▼<hits total="1" computed="1" sent="1" first="0">
    ▼<hit score="5" id="344846">
      ▼<info>
         <author>Rengül Çetin-Atalay</author>
        ▼<notes>
           <note type="affiliation">Bilkent University, Ankara, Turkey</note>
         </notes>
         <url>https://dblp.org/pid/33/3424</url>
       </info>
       <url>URL#344846</url>
     </hit>
   </hits>
 </result>
```


Figura 32. Respuesta de una consulta con un autor afiliado a una universidad

La afiliación del autor suele contener el país donde se encuentra la institución académica del investigador, por lo que merece la pena tratar de abordar el problema usando este campo. Para analizar de manera correcta el funcionamiento de este campo se procede a usar otra vez BaseX, el programa para realizar consultas en bases de datos XML. La primera consulta que se realiza es para conocer cuántos campos 'afiliación' existen en la base de datos, ya que en el estudio de la tabla 'www' se llegó a la conclusión de que muchos de sus campos permanecían vacíos la mayoría de las veces. Usando dicha consulta se obtiene que existen 125.000 entradas distintas en la base de datos con información relacionada con la afiliación de los investigadores. Tal y como se muestra en la Figura 33.

125.340 Results, ≥8192 kB (Chopped) Resu

```
<note type="affiliation">Nazarbayev University, Kazakhstan</note>
<note type="affiliation">Southern Methodist University, Department of Mechanical Engineering, Dallas, TX, USA</
note>
<note type="affiliation" label="PhD 2019">University of Glasgow, UK</note>
<note type="affiliation">University of Applied Sciences and Arts Northwestern Switzerland, Switzerland</note>
<note type="affiliation">University of Lisbon, Instituto Superior Tcnico, Portugal</note>
<note type="affiliation">Queen Mary University of London, UK</note>
<note type="affiliation">Technion, Andrew and Erna Viterbi Faculty of Electrical Engineering, Haifa, Israel</note>
<note type="affiliation">Thi-Qar University College of Computer Science and Mathematics, Thi-Qar, Iraq</note>
<note type="affiliation">University of Toronto, Canada</note>
<note type="affiliation">Beijing Institute of Technology, School of Aerospace Engineering, China</note>
<note type="affiliation">IIT Palakkad, Department of Electrical Engineering, Kerala, India</note>
<note type="affiliation">Institute for Systems and Robotics, Lisbon, Portugal</note>
<note type="affiliation">University of the Basque Country, Eibar, Spain</note>
<note type="affiliation">Jiangxi University of Finance and Economics, School of Software and Communication
Engineering, Nanchang, China</note>
<note type="affiliation">Indian Institute of Technology (ISM), Department of Applied Mathematics, Dhanbad, India</
note>
<note type="affiliation">A.P. Ershov Institute of Informatics Systems, Novosibirsk, Russia</note>
<note type="affiliation">Northeast Petroleum University, College of Electrical and Information Engineering, Daqing
, China</note>
<note type="affiliation">Xidian University, Department of Information Engineering, Xi'an, China</note>
```

Figura 33. Consulta en BaseX de la lista de afiliaciones existentes en dblp

Existen más de 2 millones de investigadores en la base de datos de dblp. Sin embargo, solo 125.000 contienen información sobre la institución académica donde trabajan, obteniendo un ratio de 1 institución cada 16 investigadores. Este número es realmente bajo para garantizar que usando este método se consiga una implementación exitosa del objetivo. Por otra parte, con este método no hace falta preocuparse por cambiar la implementación una vez completada, ya que los campos de dblp irán incrementando con el tiempo, consiguiendo el objetivo propuesto a largo plazo. Por esta misma razón, y considerando también que la búsqueda a partir de la afiliación permite no solo filtrar por país, sino por región y universidad, se decide continuar con la implementación del objetivo a partir de este campo.

El primer paso es actualizar la interfaz de usuario, y permitir que se pueda filtrar a partir de un nuevo campo llamado localización. Este parámetro es de tipo string y se usará para dar como correcta cualquier afiliación que contenga esa cadena de texto. Dentro de la lógica de la aplicación se requiere añadir nuevos métodos para poder recuperar esta información, ya que la consulta que se realiza para obtener publicaciones se hace en un endpoint distinto. Esta nueva consulta hace que tanto el tiempo como la memoria usada por la aplicación aumente de manera considerable, ya que se requerirá pedir la afiliación para cada autor que consiga pasar el filtrado del resto de parámetros.

Teniendo este pequeño problema en cuanta se propone que cada vez que se obtenga información sobre la afiliación de un autor, esta sea guardada en un fichero dentro de la aplicación. Para de esa manera crear con cada consulta una pequeña estructura de datos conteniendo la información deseada y buscando primero en ella antes de realizar la llamada a la api, funcionando de manera similar a como lo hace la memoria caché. Sin embargo, esta funcionalidad extra requeriría invertir una gran cantidad de tiempo necesario para implementar el resto de las funcionalidades

obligatorias, por lo que se propone realizar dicha implementación en el futuro, cuando la aplicación web sea totalmente funcional.

5.5.4. Hacer filtrado usando todas las venues de Google Scholar

El objetivo del filtrado usando las venues de Google Scholar es dar a los investigadores una idea de cómo se organizan los rankings en un nivel más global con respecto a una sola revista específica. Para implementar esta funcionalidad se empieza por pensar en cómo el usuario puede elegir este filtro. Una de las ideas es usar un botón de timo radio que seleccione si se elige una revista o todas las de Google Scholar, y si el botón esta activo bloquear el desplegable del resto de revistas. Con un poco más de consideración se decide descartar este modelo y simplemente introducir una nueva entrada en el json de revistas, de modo que cuando el usuario vaya a seleccionar una nueva revista o conferencia pueda ver también una nueva venue llamada Google Scholar Ranking. La Figura 34 presenta la implementación de la búsqueda de rankings a través de Google Scholar.

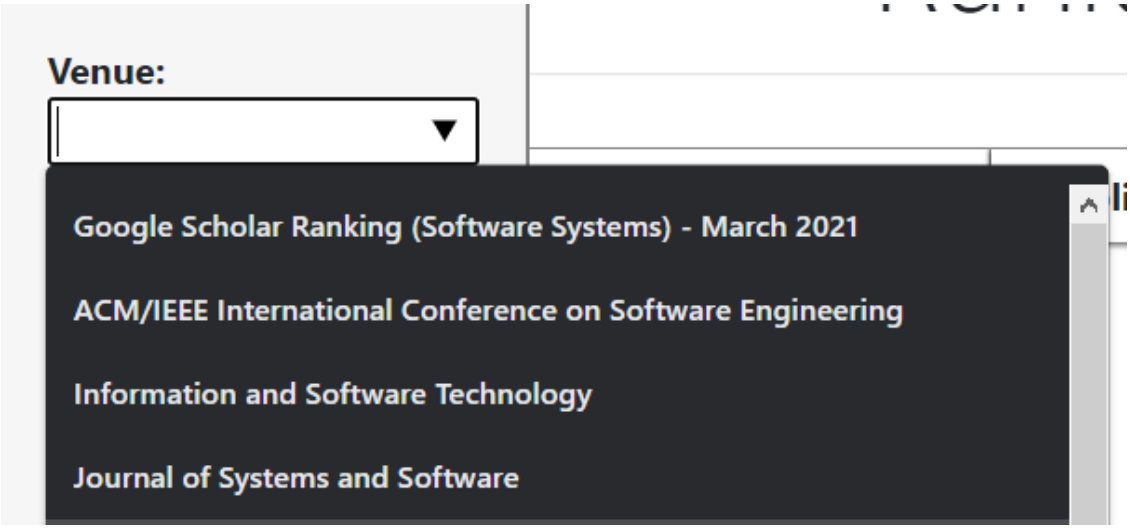


Figura 34. Implementación de la búsqueda de rankings con Google Scholar

Como se aprecia en la imagen, el ranking de revistas y conferencias de Google Scholar se obtiene en el mes de marzo, y actúa a ojos del usuario como otra revista o conferencia cualquiera. Sin embargo, su funcionalidad lógica no es tan simple como generar una sola consulta. Para recuperar los datos de todas las revistas para todos los años especificados el sistema tiene que construir y pedir una gran cantidad de consultas que hacen que el tiempo de respuesta de cara al usuario sea más largo de lo normal. Esto es un hecho, ya que si se realizan más consultas se tardará más tiempo, lo que puede dejar al cliente pensando si la aplicación sigue funcionando o no.

Para hacer ver que el sistema está buscando los datos pedidos por el usuario se genera en el momento que se pide la consulta una pequeña imagen en pantalla. Esta imagen tiene un texto que dice 'Please wait…' lo que se traduciría como 'Espere por favor'. Esta simple imagen cuya lógica funciona de manera asíncrona sirve para elevar ligeramente la calidad de la aplicación y

garantizar que el usuario esperará un poco más para poder visualizar sus consultas. La Figura 35 representa la imagen de espera usada cuando los usuarios esperan la respuesta de una consulta.

Figura 35. Implementación visual de la espera de datos

5.5.5. Obtener estadísticas de un investigador

El último objetivo del quinto sprint consiste en mostrar a los investigadores información relevante sobre ellos mismos más allá de los propios rankings. Con esta nueva funcionalidad un investigador debería ser capaz de recibir la siguiente información realizando una consulta:

- Posición en un ranking
- Percentil
- Cuartil
- Duración de la carrera
- Número de revistas en las que ha publicado
- Media de publicaciones

El cliente también indica la posibilidad de poder usar la tabla de Google Scholar para obtener la información anterior. Además, con este filtro el sistema podría también calcular en que revista sale más favorecido el investigador, añadiendo más funcionalidad a esta nueva búsqueda. Considerando la diferencia con la funcionalidad de la pantalla de búsqueda de rankings se llega a la decisión de crear una nueva pantalla para este tipo de búsqueda, llamada 'Author Search'.

De la misma forma que el resto de los objetivos implementados, el primer paso es diseñar como se van a mostrar los datos. La primera idea de diseño es dividir la pantalla en dos columnas, una para la búsqueda de revistas específicas y otra para la búsqueda de Google Scholar. De esta forma el sistema obtendría la información de la revista elegida y la mostraría a un lado de la pantalla, mientras que el resultado de la búsqueda de Google Scholar se mostraría al otro. Esta idea de diseño implica que si el usuario eligiera directamente Google Scholar no se mostrara información en un lado de la pantalla. Para solucionar esto se propone que cuando la búsqueda sea a través de Google Scholar, a un lado de la pantalla se muestre la información de la revista en la que el investigador sale más favorecido. La Figura 36 presenta el diseño de la pantalla de búsqueda de estadísticas de un investigador.

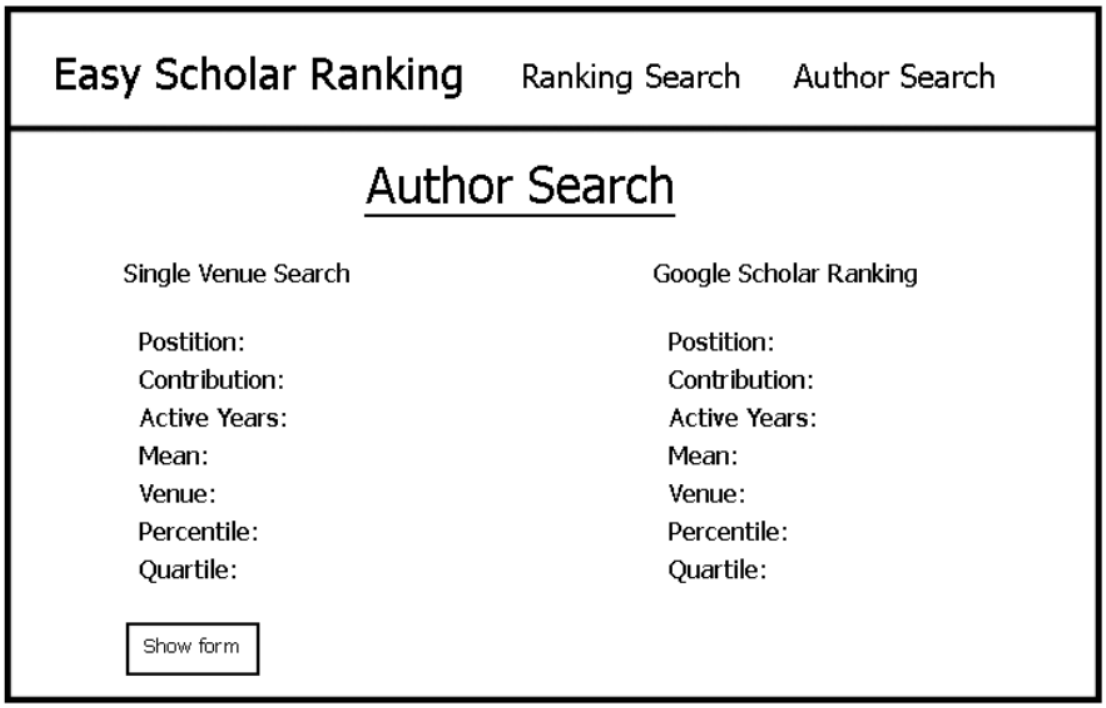


Figura 36. Diseño de la pantalla de búsqueda de estadísticas de un investigador

Con el diseño creado se puede proceder a la búsqueda de datos. Para empezar con la funcionalidad, se deja de lado la búsqueda de Google Scholar y se centran los esfuerzos en conseguir que se pueda obtener la información para una sola venue. Con los datos que ya se recuperaban en la anterior pantalla se puede obtener la información de los 5 primeros campos sin realizar ninguna operación complicada. Sin embargo, obtener el percentil y cuartil requiere realizar unos cálculos.

Para calcular el percentil se agrupan las diferentes puntuaciones que se obtienen de la lista de autores por su valor, generando un mapa de datos. La clave de este mapa es la puntuación obtenida, mientras que el valor de cada clave es el número de veces que aparece esa puntuación. El siguiente paso es calcular que porcentaje de aparición tiene cada valor en el mapa empezando desde la puntuación más baja y hasta llegar a la puntuación del investigador a estudiar, sumando el porcentaje obtenido en cada iteración que se realiza. Para la puntuación del investigador no se suma todo el porcentaje, sino que se divide entre dos para obtener la media porcentaje, ya que cuando no se conoce la posición dentro del grupo se elige el valor medio. Cuando se calcula este porcentaje se suma al anterior y se multiplica por cien para obtener percentil en el que se encuentra el investigador deseado. El cálculo del cuartil se hace en base al percentil obtenido, calculado el módulo en base 25 y restándole el resultado al número 4. Obteniendo así ambos datos deseados para un solo investigador.

La Figura 37 presenta el resultado de realizar una consulta usando una revista específica.

Single Venue Search

Position:	29 / 853
Contribution:	2 publications
Active Years:	1 years
Mean:	2 publications by year
Venue:	Inf._Softw._Technol.
Percentile:	92th
Quartile:	Q1

Author: Carlos Cetina, Venue: Information and Software Technology, Location: International, Time period: 2019 - 2020.

Figura 37. Resultado de realizar una consulta usando una revista específica

Tras obtener información sobre un autor en una revista específica, el siguiente paso es calcular el mismo proceso, pero para todas las revistas de Google Scholar. Para obtener el resultado esperado basta con usar la búsqueda que se realiza en la otra pantalla y calcular el percentil y cuartil de la misma forma que se explica anteriormente. Sin embargo, a pesar de haber realizado las mismas operaciones que en la columna de Single Venue, falta obtener el número de revistas en las que ha publicado el investigador en ese periodo de años.

Para guardar esta información se añade una nueva propiedad al modelo de autores llamada 'Venue', y se actualiza cuando se introduce el autor por primera vez en la lista. Además de este campo, también se crea una lista para poder guardar la iteración de dicha propiedad. Cuando se realiza el mapeo de autores en el que se calculan los años, se comprueba también la venue de la cual procede el autor y se guarda dentro de la lista, para que cuando se realice toda la iteración de datos se pueda conocer en que revistas ha aparecido un investigador. Finalmente, para obtener el número final basta con buscar al autor que se desea analizar y contar cuantas entradas tiene en su lista de venues.

La Figura 38 presenta el resultado de realizar una consulta usando Google Scholar Ranking.

Google Scholar Ranking (Software Systems)

Position:	**359 / 11058**
Contribution:	**5 publications**
Active Years:	**2 years**
Mean:	**2.5 publications by year**
Venues:	**published in 4 / 20 GSR venues**
Percentile:	**95th**
Quartile:	**Q1**

Author: Carlos Cetina, Venue: Google Scholar Ranking (Software Systems) - March 2021, Location: International, Time period: 2019 - 2020.

Figura 38. Resultado de realizar una consulta usando Google Scholar

Si con esta nueva funcionalidad se realiza una consulta se pueden obtener los siguientes datos:

Con la búsqueda de información de Google Scholar solo falta calcular cual es la venue en la que el investigador tiene mejor puntuación y mostrarla en la otra columna. Para obtener esta información se lee la puntuación del investigador para cada una de las revistas en las que aparece. Si la puntuación en la revista es mayor a las anteriores apariciones del autor, se guarda el valor y el nombre de la revista, para que cuando se realice la última consulta se pueda conocer en que venue sale más favorecido. Conocer esta información no basta para poder mostrarla en la pantalla, ya que lo que se devuelve a la vista tras realizar una consulta es una lista de autores. Para solucionar este pequeño problema de diseño se crea un nuevo autor llamado 'BestVenue' al cual se le asigna en la propiedad 'Venue' el nombre de la revista y en el campo de la puntuación el número de publicaciones que publica en dicha revista. De esta forma cuando la lista se consulta para obtener la información se puede extraer este 'Autor' y conocer la revista más favorecida.

Si se realiza la consulta de Google Scholar para un autor, se calcula también la revista más favorecida y se muestran los datos del autor en dicha revista. La Figura 39 presenta el resultado de realizar una consulta usando Google Scholar y obteniendo la venue en la que el investigador tiene una mayor puntuación.

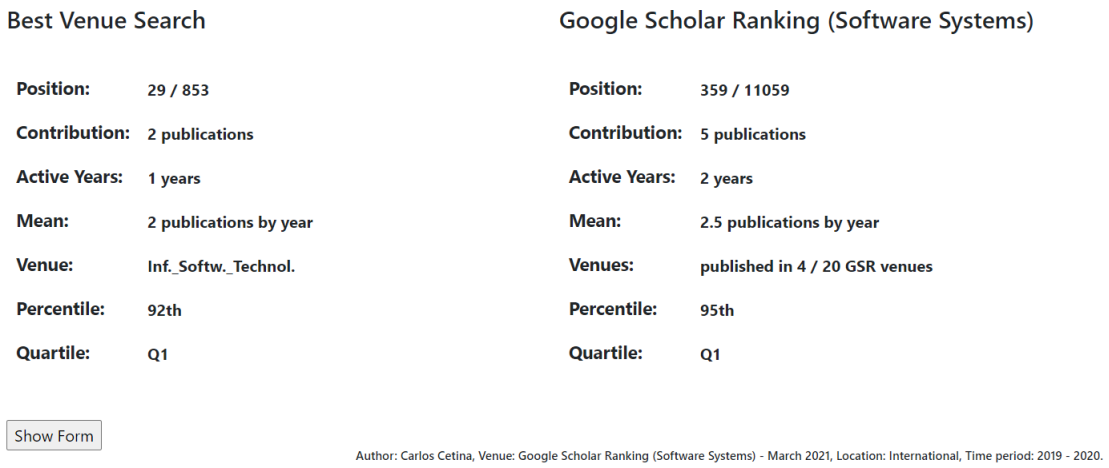


Figura 39. Resultado final de realizar una consulta usando Google Scholar

La implementación de la funcionalidad de Author Search se muestran los resultados a la tutora del proyecto, quien proporciona al alumno una cuenta en Microsoft Azure para poder volver a servir el proyecto desde la nube. Una vez subido a la plataforma se da por concluida la quinta iteración del desarrollo del proyecto y se organiza una nueva reunión para revisar y planificar el último sprint de Easy Scholar Ranking.

5.6. Sexta iteración – 32 días

La reunión introductoria a la última iteración del proyecto consiste en el testeo y análisis de las funcionalidades desarrolladas hasta el momento. En el testeo se encuentran una serie de bugs que se apuntan en una lista para que puedan ser localizados y corregidos a lo largo del desarrollo de la iteración. Otro de los aspectos más importantes de la reunión es el estudio del funcionamiento del filtrado por localización. Tras analizar las ventajas, inconvenientes y rendimiento de dicho filtrado se propone implementar la misma funcionalidad, pero usando un fichero de texto. Este fichero contendría los nombres de todos los investigadores afiliados con universidades españolas, y a través de este se podría ganar en tiempo lo que se perdería en libertad de elección.

Con la revisión y análisis de las funcionalidades terminada se especifica la lista de tareas a realizar para esta iteración del proyecto:

- Arreglar búsqueda de autores usando la localización de dblp
- Crear fichero con autores españoles
- Implementar búsqueda de autores con el fichero
- Crear pantalla de preguntas más frecuentes
- Crear pantalla mostrando lista de Google Scholar
- Corregir bugs

5.6.1. Arreglar búsqueda de autores usando la localización de dblp

Tras testear el funcionamiento del filtrado por localización implementado en el último sprint se llega a la conclusión de que la información obtenida es incorrecta. Con el filtrado se comprueba si los autores que aparecen al buscar el nombre del investigador en dblp tienen la afiliación que se estipula en el campo de localización. El problema con esto es que se comprueba si la afiliación aparece alguna vez entre todos los resultados obtenido, y si buscamos a 'David Lo', lo más probable es que obtengamos información también sobre 'David Pérez' o 'David López'. Y basta con que uno solo de ellos tenga el campo buscado en la afiliación para que se decida que 'David Lo' está afiliado a dicha universidad o país. Para corregir este bug se comprueba también el nombre del autor, y así el sistema se asegura de que se obtiene la localización para un investigador en concreto.

A partir de esta corrección surge otro fallo en la aplicación. Este error se genera porque el sistema trata de leer un solo campo de afiliación para cada investigador, siendo que estos pueden estar afiliados a múltiples instituciones académicas. Para solucionar el problema se implementa un algoritmo de lectura similar al que lee grupos de investigadores a partir de publicaciones, y se logra implementar la funcionalidad que se deseaba en la iteración anterior.

5.6.2. Corregir bugs

En el anterior sprint se implementa un pequeño texto que aparece cuando los usuarios hacen una consulta. Sin embargo, a pesar de que se oculte la forma de entrada de datos, el usuario puede seguir interactuando con ella, ya que simplemente se hace invisible. Para solucionar este pequeño problema basta con usar la propiedad de Razor Pages que dictamina cuando se muestran u ocultan elementos para bloquear de verdad la forma de entrada de datos. De esta manera el usuario solo puede esperar a que se carguen los datos y no hace cambios que pudieran poner en peligro el funcionamiento de la aplicación.

Otro de los bugs que se encuentran durante el ejercicio de testeo tiene que ver con el cálculo de los años activos de un investigador. En una de las iteraciones anteriores se calcula este valor a partir del primer y último año que se encontraba una publicación a nombre del autor. Cuando se encuentra una publicación con otro año se actualiza el último año, pero no el primero. Esto hace que si un investigador tiene un año con una fecha anterior a la que en su modelo aparece como primera no se actualice, haciendo que el cálculo de años activos no sea correcto. Para arreglar este comportamiento se compara también el primer año de cada investigador, obteniendo de esa manera la funcionalidad deseada.

5.6.3. Crear fichero con autores españoles

En la reunión del sprint se propone al alumno buscar en internet la lista de investigadores que trabajan en una universidad española. Sin embargo, a pesar de la búsqueda no se encuentra

ninguna forma de obtener esta información. Por suerte, la tutora del proyecto conoce páginas web donde obtener esta información y ofrece al alumno una serie de ficheros con información relevante al objetivo de la implementación.

Estos 7 ficheros contienen una tabla en formato texto con información sobre diferentes publicaciones procedentes de grupos de investigación de universidades españolas. Entre las muchas columnas una de ellas contiene los miembros del proyecto, por lo que se decide extraer esa información en cada uno de los ficheros. Con el objetivo de obtener solamente la columna de autores se utiliza Microsoft Excel para borrar el resto de las columnas y filas que no son útiles para el proyecto, de esta forma obteniendo en cada fichero un listado de investigadores.

Sin embargo, estos ficheros no se pueden usar en la aplicación, ya que ni aparecen en el formato deseado, ni es una buena práctica recorrer tantos ficheros para buscar un dato entre todos ellos. Se debe obtener toda la información en el mismo fichero y con el formato deseado. Para ello se crea un código en Java capaz de recorrer cada uno de los archivos quitando caracteres innecesarios y escribiendo en uno nuevo el nombre de los diferentes investigadores que aparecen. De esta forma se pasa de tener una serie de archivos con tablas ilegibles a uno solo con una lista de más de 4000 autores afiliados a universidades españolas.

5.6.4. Implementar búsqueda de autores con el fichero

Una vez obtenido el fichero con la lista de autores, el siguiente objetivo de la iteración es implementar un filtrado usando ese mismo fichero. El primer paso es decidir dónde se va a realizar el filtrado, y tras un pequeño análisis se decide incorporar la funcionalidad en las dos vistas de búsqueda. Para ello se añade con un botón radial la opción de elegir qué tipo de búsqueda se realizará para filtrar la localización, la api de dblp, o el fichero local. La Figura 40 muestra la selección del filtro de localización.

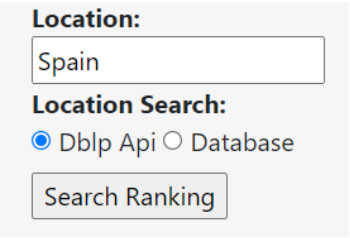


Figura 40. Selección del filtro de localización

Cuando el usuario selecciona 'Database' se realiza la búsqueda con el fichero. Para ello se abre y recorre el archivo y se compara el nombre del autor especificado por el usuario con cada una de las entradas del archivo. En el objetivo anterior se limpian los caracteres innecesarios, y entre ellos, se acaba eliminando también las comas de los nombres y apellidos. Es por esta razón que cuando se realiza la comparación se eliminan también los acentos del nombre del autor y se deja todo el texto en minúsculas, obteniendo así el resultado esperado.

Es preciso considerar que la implementación de la búsqueda se puede optimizar a partir de algoritmos de búsqueda, pero se considera que dicha optimización no se debe realizar durante el desarrollo del proyecto, y se guarda como futura mejora.

5.6.5. Crear pantalla de preguntas más frecuentes

Tras la implementación de todas las opciones de filtrado existen cuestiones que un usuario podría llegarse a hacer. Es por eso por lo que se propone crear una nueva vista donde el usuario pueda ver algunas de las dudas más comunes que puede llegar a tener cuando hace uso de Easy Scholar Ranking. El diseño de la vista en este caso se realiza de la misma forma que la pantalla principal, mostrando las preguntas en forma de párrafo para facilitar así su lectura.

Las diferentes preguntas con sus respectivas contestaciones en inglés son las siguientes:

- **Where does the data come from?** All the information about the authors and rankings are obtained from dblp's api.
- **How can I search specific venues?** Specific venues can be chosen. However, they require their specific searching name. This information can be acquired at dblp's page.
- **What is Google Scholar Ranking?** GSR are the top 20 venues in Google Scholar's Software Systems' ranking (March 2021). When searching this way, the system searches every individual venue and merges the final result.
- **Filter does not seem to be working, why?** The application may rarely fail to obtain information from the external api due to connection issues. If this happens to you, wait a little bit, and try again.
- **How long does it take to search an author or ranking?** Searching time depends on the parameters chosen. Factors that can increase the searching time are location and number of venues.
- **What is the difference between Dblp Api and Database search?** Dblp api uses the information obtained from dblp's service to check the affiliation of every author on real time. On the other hand, the database looks into the internal files of the application for such affiliations.
- **Can I search any location I want?** If you use dblp api search you can look for any location. The data is updated by dblp as authors register their affiliations, so it may not be very precise. If you wish to search by a specific location, try look at the database first and use it if your location is included. If not, try dblp api.
- **What do active years mean when searching?** Active years keeps track of the period of time the author has between its first and last publication inside the data **queried**. Meaning that if the searching interval is 5 years, the active time of the author will not be greater than 5.

- **What does international location refer to?** International search is the one that includes every author, regardless of their affiliation.

Estas preguntas engloban las dudas más frecuentes que el usuario puede tener al hacer uso de la aplicación en su versión actual. Y podrán ser modificadas en próximas iteraciones de Easy Scholar Ranking.

5.6.6. Crear pantalla mostrando lista de Google Scholar

A lo largo del desarrollo del proyecto se comenta en múltiples ocasiones el uso de la lista de revistas y conferencias de Google Scholar. Sin embargo, en ningún momento se muestra el contenido de la misma, y al hacer uso de ella en algunos de los filtros de la aplicación, se considera que debe incluirse una vista en la que se pueda visualizar cuáles son estas revistas y el orden en el que aparecen.

Similar al anterior objetivo, el diseño que se elige es igual al de la pantalla principal, haciendo hincapié en la importancia de las revistas y conferencias dentro del funcionamiento de Easy Scholar Ranking. Las venues que se utilizan en el proyecto y que se obtienen en marzo de 2021 son las siguientes:

1. ACM/IEEE International Conference on Software Engineering
2. Information and Software Technology
3. Journal of Systems and Software
4. ACM SIGSOFT International Symposium on Foundations of Software Engineering
5. Empirical Software Engineering
6. IEEE Transactions on Software Engineering
7. ACM SIGPLANT-SIGACT Symposium on Principles of Programming Languages
8. ACM SIGPLAN Conference on Programming Language Design and Implementation
9. IEEE/ACM International Conference on Automated Software Engineering
10. IEEE Software
11. Symposium on Operating Systems Principles
12. Software & Systems Modeling
13. Mining Software Repositories
14. International Conference on Software Analysis, Evolution, and Reengineering (SANER)
15. International Symposium on Software Testing and Analysis
16. International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)
17. IEEE International Conference on Software Maintenance and Evolution
18. Proceedings of the ACM on Programming Languages
19. Software: Practice and Experience

20. ACM SIGPLAN International Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA)

Una vez realizada la implementacion de todos los objetivos de la iteración se vuelve a analizar y testear la aplicación. Durante el testeo de la aplicación en el entorno de Microsoft Azure se observa que el filtrado de localización con el archivo no funciona de ninguna manera. Para encontrar el origen del problema se observa el log de la aplicación usando una de las herramientas que Azure incorpora para gestionar aplicaciones web [18].

```
2021-04-20 17:35:58.329 +00:00 [Information] Microsoft.AspNetCore.Routing.EndpointMiddleware: Executing endpoint '/
2021-04-20 17:35:58.329 +00:00 [Information] Microsoft.AspNetCore.Routing.EndpointMiddleware: Executing endpoint '/
2021-04-20 17:35:58.329 +00:00 [Information] Microsoft.AspNetCore.Routing.EndpointMiddleware: Executed endpoint '/_
2021-04-20 17:35:58.329 +00:00 [Information] Microsoft.AspNetCore.Hosting.Diagnostics: Request finished HTTP/1.1 PO
https://easyscholarrankingusj.azurewebsites.net/_blazor?id=2H8a3Yb8FlGhZfuKh3LZkQ text/plain;charset=UTF-8 44 - 200
2021-04-20 17:35:58.559 +00:00 [Error] Microsoft.AspNetCore.Components.Server.Circuits.CircuitHost: Unhandled excep
'BG2FFBciK_6BKdgR1A-_8VvNWt0325GgzzxQ60vJFIA'
System.IO.FileNotFoundException: Could not find file 'C:\home\site\wwwroot\Data\spanish-authors.txt'.
File name: 'C:\home\site\wwwroot\Data\spanish-authors.txt'
   at System.IO.FileStream.ValidateFileHandle(SafeFileHandle fileHandle)
   at System.IO.FileStream.CreateFileOpenHandle(FileMode mode, FileShare share, FileOptions options)
   at System.IO.FileStream..ctor(String path, FileMode mode, FileAccess access, FileShare share, Int32 bufferSize,
   at System.IO.StreamReader.ValidateArgsAndOpenPath(String path, Encoding encoding, Int32 bufferSize)
   at System.IO.StreamReader..ctor(String path, Encoding encoding)
   at System.IO.File.ReadLines(String path)
   at EasyScholarRanking.Data.AuthorFileService.SearchAuthor(String authorName) in
   C:\Jorge\Universidad\5\EasyScholarRanking\Projects\EasyScholarRanking\EasyScholarRanking\Data\AuthorFileService.
```

Figura 41. Log de errores en la aplicación

La Figura 41 muestra el log de errores de la aplicación, el cual dice que no se puede encontrar el fichero dentro del proyecto. Esto ocurre porque el archivo con los autores no se sube a Microsoft Azure con el resto de los archivos de datos. La solución que se realiza es incorporar manualmente el fichero a la ruta de la que lee la aplicación desde Azure, haciendo que se puedan realizar las búsquedas de autores a partir de su localización.

La corrección de este error concluye la implementación del desarrollo de la aplicación web de Easy Scholar Ranking. En este capítulo se estudia cómo se construye la demo del proyecto, incluyendo la adaptación de la metodología, el estudio de los objetivos y la superación de retos o problemas surgidos a lo largo del periodo de desarrollo.

6. Estudio Económico

El objetivo del estudio económico del proyecto es el cálculo de los distintos tipos de indicadores que informen sobre la viabilidad de este. El estudio económico, al contrario que un presupuesto, estima los costes de explotación. Es decir, los costes que se producen durante la vida útil del proyecto. Para el cálculo del estudio se efectuará el método económico, el cual evalúa la relación costes-ingresos [19].

6.1. Método económico

El método económico consiste en la estimación de los costes e ingresos de funcionamiento del proyecto para calcular los ratios de rentabilidad. La estimación de los costes se determina a partir de la naturaleza de estos, siendo los más importantes los costes anuales por naturaleza, los costes por secciones y finalmente, los costes por producto.

Los costes por naturaleza pueden clasificarse a su vez en costes de materiales, costes de mano de obra y costes por utilización de equipos e instalaciones, que a su vez dependen del coste de amortización, consumo y mantenimiento.

Por otra parte, el ingreso se obtiene a través del producto del precio de venta unitario y el volumen de ventas. El precio de venta se estima a partir del estudio de productos similares en el mercado, mientras que el volumen de ventas se trata como variable independiente debido a la dificultad que supone realizar su estimación.

Conociendo tanto costes como ingresos del proyecto se puede determinar el beneficio bruto de la siguiente manera:

$$Beneficio\ bruto\ (B) = Ingresos\ (I) - Costes\ totales(C)$$

$$Tasa\ de\ rentabilidad\ (r)$$

Otro concepto importante de entender y determinar es el punto de equilibrio, el cual indica el volumen de ventas necesario para compensar los costes totales del proyecto. Suponiendo que el precio de venta y el coste unitario del producto sea independiente del volumen de ventas.

$$Punto\ de\ equilibrio\ (qe) = \frac{Costes\ fijos\ (Cf)}{Precio\ de\ venta\ unitario(p)\ -\ Coste\ variable\ unitario\ (cv)}$$

El estudio a través del método económico resulta realmente interesante para poder estimar la relación ingresos-costes de Easy Scholar Ranking. Sin embargo, la monetización del proyecto no se estudia en ninguno de los apartados ya que se supone que será estudiada por el cliente del proyecto, y no por el alumno. Por esta razón se procede a analizar solamente los costes del desarrollo del proyecto, que se distribuyen de la siguiente manera:

Costes de mano de obra

Los costes de mano de obra abarcan todos los costes que se generan a través de la inversión de esfuerzo en forma de tiempo por los integrantes del proyecto. Este esfuerzo se calcula a través del número de horas invertidas en el proyecto, ya sean reuniones de estudio, análisis de herramientas o desarrollo del producto final.

Para calcular el número de horas invertidas se hace uso del apartado de desarrollo, en el cual se referencia el número de reuniones organizadas, siendo 6 las reuniones a lo largo del desarrollo. Además de las reuniones también se invierte tiempo en forma de testeo y análisis de requerimientos por parte de la tutora y cliente, por lo que se estima que relacionada con las reuniones se invierte un valor de esfuerzo de 10 horas.

Según el capítulo anterior se realizan 6 iteraciones a lo largo de 157 días. La carga de trabajo diaria no es constante a lo largo del proyecto, sino que es constante para cada sprint, obteniendo una estimación de 40 horas por iteración del proyecto. Esta estimación supone 40 horas invertidas en la fase de análisis, y 200 en la fase de desarrollo.

Para el desarrollo del estudio económico solo se considera el trabajo invertido por el alumno, y no por la tutora, ya que se desconocen los datos necesarios para realizar las estimaciones. La Tabla 2 muestra las tareas realizadas a lo largo del proyecto y las horas de trabajo invertidas en cada una de ellas.

Tarea	**Estimación de horas alumno**
Reuniones organizadas	10
Fase de análisis	40
Fase de desarrollo	200
Fase de redacción de la memoria	35
Total de horas	285

Tabla 2. Tabla de tareas y horas del alumno

Para calcular el coste de la mano de obra [20] [21] es preciso conocer el coste que supone mantener a un empleado con las características requeridas [22] para realizar las labores del alumno en las fases de análisis y desarrollo.

La Tabla 3 muestra la tabla de los salarios a través de los cuales se obtiene el coste bruto de mantener a un trabajador.

Posición	Salario neto por año	Salario nota por hora	Coste bruto por hora
Programador junior	20.000€	10.5€	10.5 + 24% = 13.02€
Analista junior	21.600€	11.25€	11.25 + 24% = 13.95€

Tabla 3. Tabla de salarios

Tras obtener el coste que le supone a la empresa mantener a un trabajador se puede calcular el coste total de la mano de obra, que se muestra a través de la Tabla 4.

Posición	Horas	Coste bruto por hora	Coste total trabajador
Programador junior	200 horas	13.02€ por hora	2473.8€
Analista junior	40 horas	13.95€ por hora	558€
Coste total de mano de obra			3031.8€

Tabla 4. Tabla de coste de mano de obra

Costes de materiales

El coste de materiales hace referencia al gasto que se ha realizado a través de la pérdida de valor de los equipos de trabajo usados durante el desarrollo del proyecto. Todo el desarrollo se lleva a cabo con un ordenador portátil, por lo que los costes de materiales se calculan a través de la pérdida de valor del equipo de trabajo, como se indica en la Tabla 5.

Equipo de trabajo	Coste del equipo	Tiempo de vida	Pérdida de valor
HP ProBook 440 G7	889€	6 años	3.22€

Tabla 5. Tabla de coste de materiales

Costes de instalaciones

El último tipo de coste se calcula a través del valor de las instalaciones u equipos usados a lo largo del proyecto. Dentro de esta categoría entraría el coste de mantener una aplicación web en la plataforma de Microsoft Azure [23].

La prueba que se utiliza para probar los servicios de Azure es gratis, mientras que el App Service que se usa para servir Easy Scholar es proporcionado por la universidad, haciendo que el coste de ambos servicios sea 0€. En caso de desarrollar el proyecto por cuenta propia, se debería de realizar un coste mensual de un mínimo de 8€ para adquirir el entorno de desarrollo en la nube.

La Tabla 6 presenta la tabla de coste de instalaciones.

Servicio usado	Coste mensual	Meses	Coste del servicio
Prueba suscripción Azure	Gratis	1	0€
App Service de desarrollo	Gratis	3	0€
Coste total de instalaciones			0€

Tabla 6. Tabla de coste de instalaciones

Costes totales del proyecto

Para concluir con el estudio económico se calculan los costes del proyecto en base a los costes previamente calculados, obteniendo de esa manera el coste total del proyecto. La Tabla 7 presenta el cálculo del coste total del proyecto.

Tipo de coste	Valor
Coste de mano de obra	3031.8€
Coste de materiales	3.22€
Coste de instalaciones	0€
Coste total del proyecto	2477.02€

Tabla 7. Tabla de coste total del proyecto

7. Resultados

Este capítulo de la memoria recoge los diferentes resultados obtenidos a lo largo del desarrollo del proyecto. Para una mejor comprensión se divide el capítulo en 2 puntos. En el primero se hace un análisis del artefacto generado y el rendimiento que ofrece. Mientras que en el segundo se estudian las desviaciones con respecto a la metodología propuesta y los objetivos que se han cumplido o modificado a lo largo del proyecto.

7.1. Análisis del artefacto

El objetivo principal del proyecto es la creación de una aplicación web funcional con la capacidad de generar rankings de investigadores. Este objetivo se cumple y a partir de él se obtiene la aplicación web de Easy Scholar Ranking. A pesar de no implementar todas las funcionalidades esperadas al principio del desarrollo, Easy Scholar Ranking ofrece la oportunidad a sus usuarios de generar algo más que solamente rankings.

Servida desde Microsoft Azure, la aplicación cuenta con su propio dominio web [24] que permite a los usuarios reconocerla sin ningún problema.

Si se accede a la aplicación la pantalla de bienvenida muestra un pequeño resumen de las diferentes funcionalidades que ofrece el servicio. Entre las que se encuentran la búsqueda de rankings, que permite generar rankings de investigadores en base a unos parámetros. Y la búsqueda de autores, que permite obtener información más detallada de un investigador dentro de las revistas más aclamadas por Google Scholar. La Figura 42 presenta la pantalla de bienvenida final.

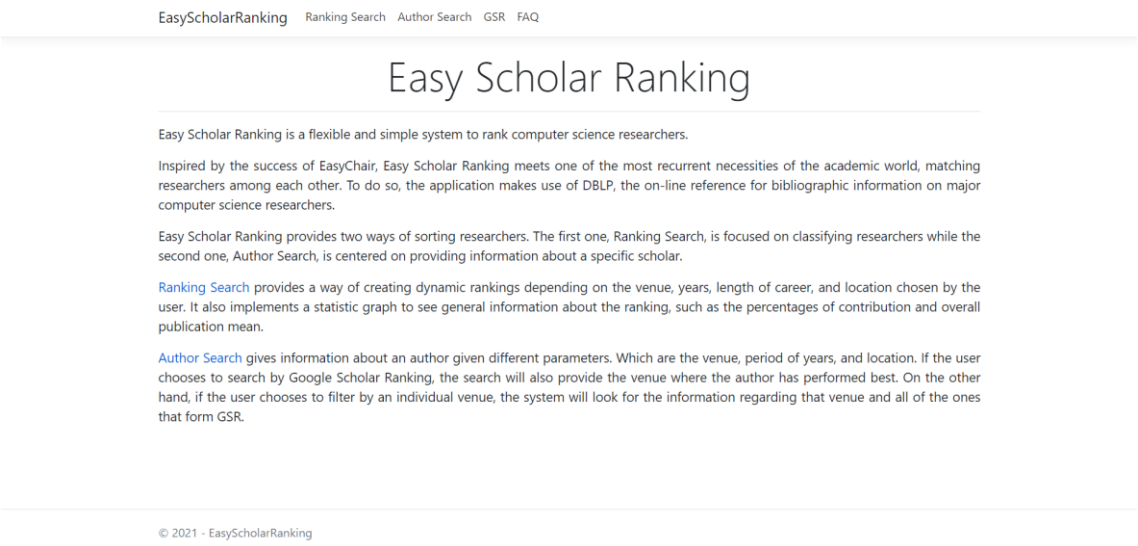


Figura 42. Pantalla de bienvenida final

7.1.1. Ranking Search

La primera de las pantallas funcionales que presenta Easy Scholar Ranking es la de Ranking Search. Esta pantalla permite a los usuarios crear rankings dinámicos dependiendo de la revista, intervalo de años, años activos y localización de los investigadores. Además de permitir modificar la longitud del ranking en todo momento.

Cuando un usuario busca un ranking el sistema se pone en contacto con el servicio web de dblp para obtener todos los datos deseados. Una vez en el sistema los datos se modifican para poder ser procesados correctamente y de esa manera mostrados al usuario. Con el ranking creado se genera una gráfica circular que permite observar cómo se distribuyen las publicaciones obtenidas. Además, se pinta de un color distinto aquellos investigadores con un número de publicaciones por encima de la media, para así visualizar más claramente aquellos autores que están por delante del resto. La Figura 43 presenta la pantalla de búsqueda de rankings final.

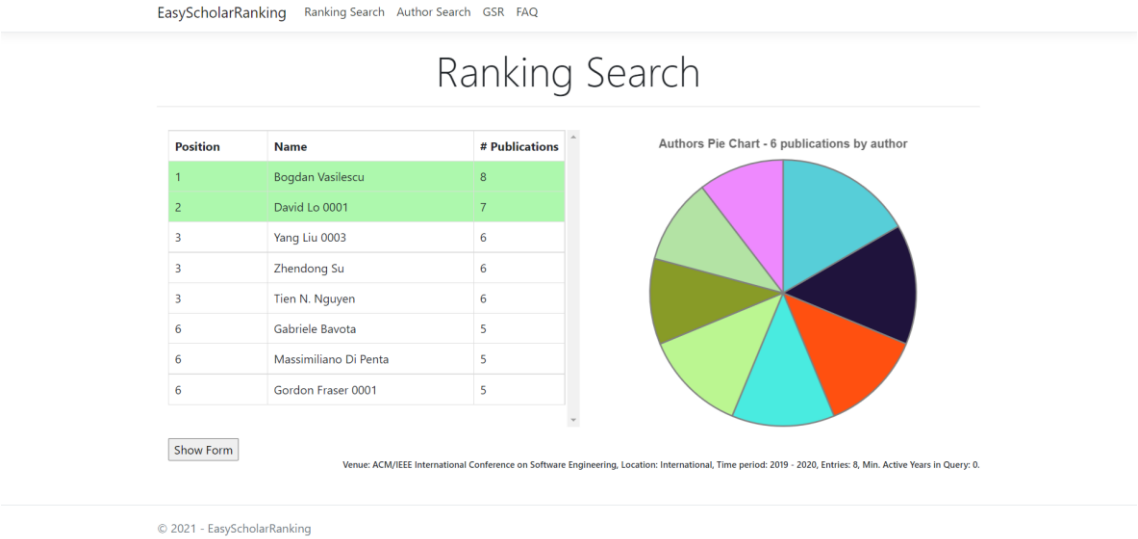


Figura 43. Pantalla de búsqueda de rankings final

Debajo de la gráfica pueden observarse los parámetros usados en la creación del ranking, y de esa manera poder realizar capturas de pantalla en las que se muestren tanto los datos obtenidos como los parámetros usados en la búsqueda.

Existen tres tipos de búsquedas que se pueden realizar a partir del filtrado de revistas. La primera es seleccionando una de las veinte que se obtienen del ranking de Google Scholar. Esta búsqueda obtiene la información sobre los investigadores que han publicado en ella y los muestra al usuario. La segunda es seleccionando la búsqueda de Google Scholar, la cual obtiene información sobre todas las revistas del ranking y las junta para poder así visualizarlas. Este tipo de búsqueda resulta bastante útil para observar cómo se distribuyen los rankings en las revistas más importantes del ámbito de las ciencias de la computación. Finalmente, se ofrece la oportunidad al usuario de

escribir la revista o conferencia que desee, siempre que el nombre sea igual que el que se usa para realizar las consultas al servicio web de dblp.

Otro de los parámetros usados para filtrar información son el intervalo de años. Eligiendo en que año empieza y acaba la búsqueda el usuario es capaz de consultar desde los primeros datos introducidos en la base de datos de dblp hasta la información más reciente.

El siguiente parámetro es años activos, el cual sirve para filtrar investigadores dependiendo de cuando publicó por primera y última vez en el intervalo de años establecido. Es decir, funciona junto al filtro del intervalo de años para comprobar si un investigador ha realizado publicaciones a lo largo del periodo de tiempo establecido.

La cuarta propiedad que puede modificar el usuario es el número de entradas que se pueden mostrar en la tabla, y que sirve para hacer el ranking más o menos grande. La longitud del ranking afecta también al gráfico, el cual puede llegar a agrupar investigadores si su contribución al ranking es muy pequeña.

Finalmente, el último filtro que ofrece la búsqueda de Rankings es el de localización. Este filtro permite al usuario escribir el lugar a través del cual desea filtrar a los investigadores, y ofrece dos formas de obtener estos datos. La primera es a través de la api de dblp, y en esta opción se permite al usuario escribir lo que quiera en el campo de texto. La opción elegida se usa entonces para comprobar si los investigadores están afiliados a una institución con ese nombre, e independientemente de si se ha elegido una ciudad, provincia, país o universidad se obtendrán los resultados deseados. Sin embargo, esta opción hace uso de una funcionalidad que no presenta mucha información en el servidor por lo que a pesar de ofrecer mucha libertad al usuario, su efectividad no es tan alta como pudiera desearse. La segunda forma de filtrar la localización es usando un fichero dentro de la aplicación. Con esta opción los usuarios solo pueden elegir entre una búsqueda internacional, que no filtra nada, y España. Con España los usuarios pueden obtener datos sobre los investigadores afiliados a universidades españolas. Esta opción es menos flexible que la anterior, pero también más rápida y precisa para buscar entradas en territorio español.

La Tabla 8 muestra el estudio del rendimiento de la búsqueda de rankings. En este estudio se analiza el tiempo de respuesta del sistema cuando se realizan diferentes consultas.

# venues	**Intervalo Años**	**Años activos**	**# entradas**	**Tipo de búsqueda**	**Tiempo de respuesta**
1	0	0	8	Internacional	0.6s
1	4	0	8	Internacional	1.2s
1	4	4	8	Internacional	1.15s
1	4	4	64	Internacional	1.2s

1	4	4	8	Dblp Api	23.4s
1	4	4	8	Database	2.3s
20	0	0	8	Internacional	2.5s
20	4	0	8	Internacional	14.5s
20	4	4	8	Internacional	12.7s
20	4	4	64	Internacional	15.9s
20	4	4	8	Dblp Api	51.6s
20	4	4	8	Database	14.8s

Tabla 8. Tabla de rendimiento de la búsqueda de rankings

Tras realizar el análisis de los tiempos de respuesta de la búsqueda de rankings a partir de los filtros usados se obtienen las siguientes conclusiones:

- El número de venues filtradas afecta ligeramente al tiempo de espera
- El intervalo de tiempo afecta al tiempo de espera proporcionalmente al número de venues que se filtren
- Los años activos no afectan al tiempo de espera
- Los número de entradas no afectan al tiempo de espera
- La búsqueda de localización con dblp afecta considerablemente al tiempo de espera
- La búsqueda de localización con el fichero de autores españoles afecta ligeramente al tiempo de espera

7.1.2. Author Search

La segunda pantalla de funcionalidad del proyecto permite a los usuarios obtener información acerca de los investigadores. La información que se muestra tiene que ver con su posición dentro de rankings, su contribución, años activos, media de publicaciones, número de revistas en las que han publicado y, percentil y cuartil en el que se encuentran.

En esta pantalla los usuarios pueden seguir filtrando por propiedades como el nombre del autor, venue, intervalo de años y localización. Dependiendo de la venue que elija el usuario el sistema se comporta de dos formas distintas. En la primera, si el usuario elige una venue normal el sistema busca la información relevante al autor en esa venue y también la información de acuerdo a todas las venues de Google Scholar, calculando en el proceso el número de revistas en las que publica el autor. La segunda opción se da lugar cuando el usuario decide elegir todas las venues de Google Scholar, haciendo que el sistema procese primero la información relacionada a todas las venues y calcule cuál es la revista en la que el investigador tiene más publicaciones. Tras obtener dicha revista el sistema procede a buscar la información permitente al autor en esa revista, mostrando entonces toda la información por pantalla.

La Figura 44 presenta el resultado de realizar una consulta en la pantalla de búsqueda de autor.

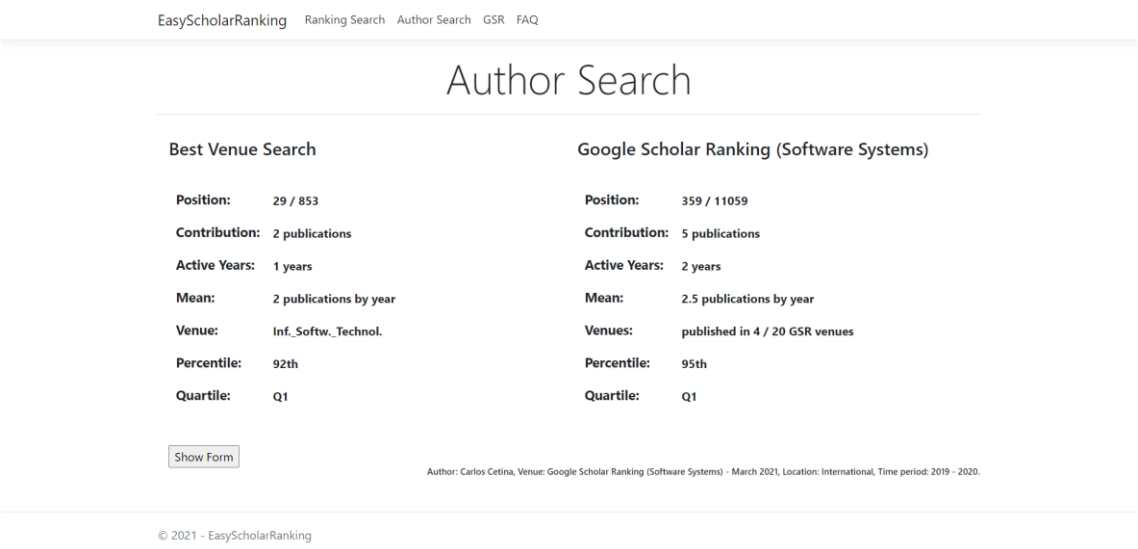


Figura 44. Resultado de realizar una consulta en la pantalla de búsqueda de autores

De la misma forma que en la anterior pantalla, se procede a continuación al análisis de tiempo de respuesta de la aplicación dependiendo de los parámetros de entrada, tal y como se muestra en la Tabla 9:

# venues	Intervalo Años	Tipo de búsqueda	Tiempo de respuesta
1	0	Internacional	12.7s
1	4	Internacional	12.8s
1	4	Dblp Api	> 120s
1	4	Database	> 120s
20	0	Internacional	13.3s
20	4	Internacional	14.2s
20	4	Dblp Api	> 120s
20	4	Database	> 120s

Tabla 9. Tabla de rendimiento de la búsqueda de autores

Tras realizar el análisis de los tiempos de respuesta de la búsqueda de rankings a partir de los filtros usados se obtienen las siguientes conclusiones:

- El número de venues e intervalo de años apenas afecta al tiempo de respuesta
- Tanto la búsqueda por localización con la api con la del fichero tienen tiempos de respuesta demasiado alto, por lo que se deberá optimizar en futuras implementaciones.

7.1.3. Google Scholar Ranking

Las revistas y conferencias obtenidas a partir del ranking de Google Scholar son una parte fundamental del funcionamiento de la aplicación. Es por eso por lo que se incluye una pantalla que muestra la lista ordenada con el nombre de las venues.

La lista obtenida es de las venues con mayor puntuación relacionadas con sistemas software de marzo de 2021. La Figura 45 presenta la pantalla de venues de Google Scholar final.

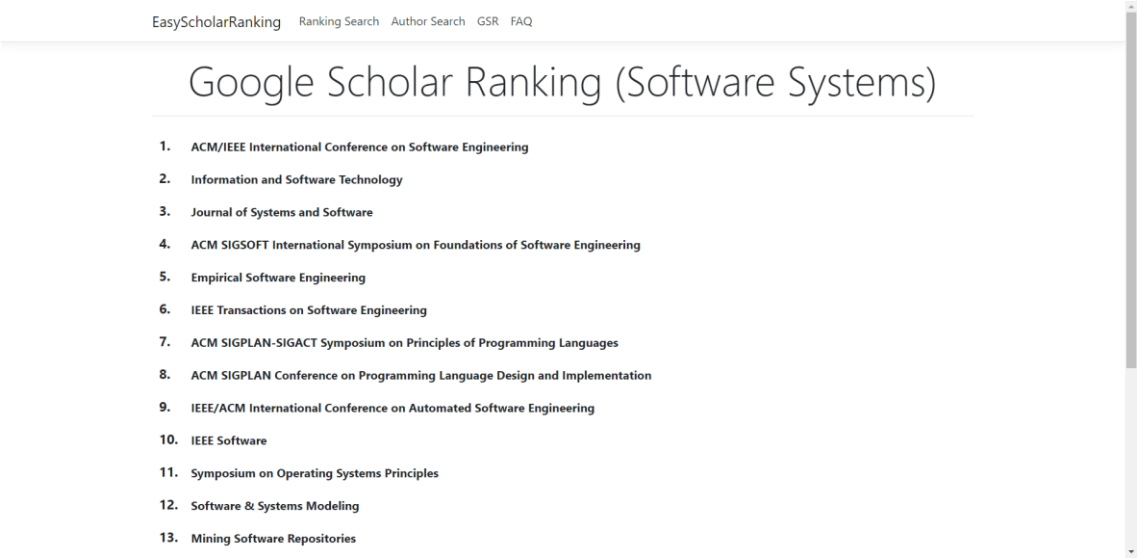


Figura 45. Pantalla de venues Google Scholar Ranking final

7.1.4. Frequently Asked Questions

Tal y como se comenta en la última iteración del capítulo de desarrollo, se introduce una vista para poder consultar las dudas más comunes y que cualquier usuario pueda utilizar la aplicación de forma sencilla. La Figura 46 presenta la pantalla de preguntas más frecuentes final.

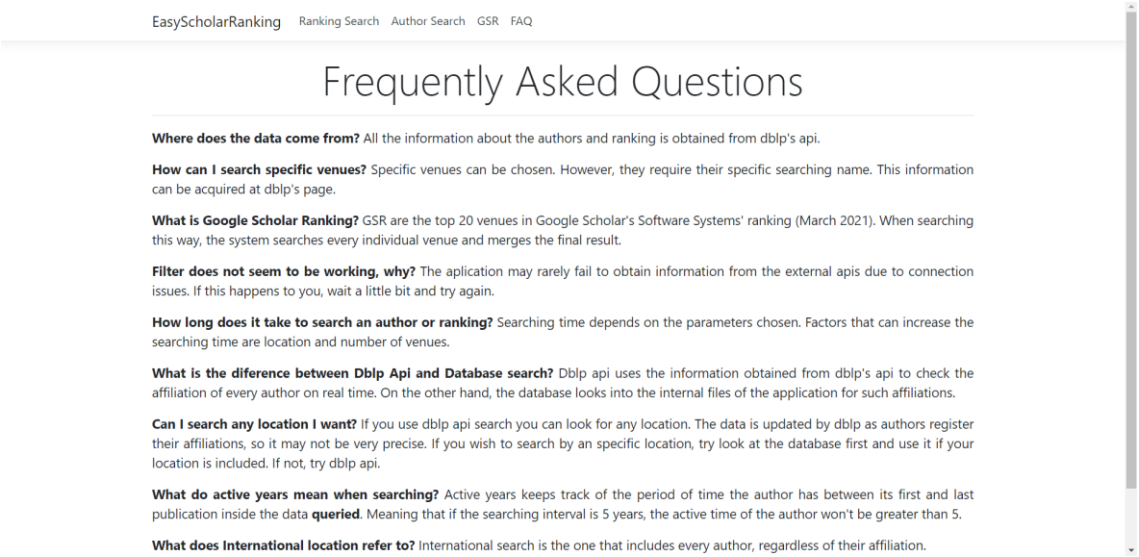


Figura 46. Pantalla de preguntas más frecuentes final

7.2. Desviaciones del proyecto

En el capítulo cuatro de la memoria se realiza una planificación estimada de la metodología a seguir a lo largo del desarrollo del proyecto. Sin embargo, en la fase de desarrollo se producen múltiples modificaciones que desvían al proyecto de la metodología original.

La primera modificación que se realiza está relacionada con la duración del desarrollo del proyecto y el número de iteraciones que se realizan. En la metodología original el proyecto se divide en cinco iteraciones con una duración máxima de un mes, estimando una duración del proyecto de cinco meses. Sin embargo, y a pesar de que se cumple con la condición de mantener las iteraciones por debajo de treinta días, el número de iteraciones acaba siendo de seis. Esto se debe a modificaciones de requerimientos durante las reuniones de análisis y estudio del progreso, extendiendo la duración del desarrollo de Easy Scholar Ranking a un total de seis meses.

La segunda desviación ocurre con la planificación estimada de los requerimientos que se deben desarrollar en cada iteración. En la planificación original el proyecto comienza con el estudio de herramientas y frameworks a través de los cuales se puede obtener información de los investigadores. Tras acabar con el estudio, los siguientes ciclos de desarrollo se centran en la implementación de rankings con distintas características, llegando a generar un total de cinco rankings diferentes. Finalmente, en última iteración comienza con el estudio de herramientas para desplegar la aplicación en la nube, con el posterior despliegue del servicio.

Sin embargo, la metodología final adoptada en el desarrollo acaba teniendo una gran desviación en cuanto al orden de implementación de los requerimientos y a los objetivos de desarrollo de cada iteración. Mientras que en la primera iteración si que se estudian las herramientas y se analizan las diferentes formas de obtener información, a partir de la segunda se comienza a priorizar requerimientos más importantes entrelazando de esa manera fases de búsqueda, estudio y desarrollo. Finalmente, mientras que en la planificación original se realizaba el despliegue de la aplicación en la última iteración, en el desarrollo se despliega la primera versión de la aplicación tras completar el primer ranking, obteniendo de esa manera la retroalimentación del cliente necesaria para generar nuevos requerimientos.

La desviación de los objetivos y la explicación detrás de dicha modificación se lleva a cabo en el capítulo final.

7.3. Open Science

Open Science [25] representa una nueva forma de abordar el progreso científico basada en trabajo cooperativo y nuevas formas de difundir el conocimiento a través de tecnologías digitales y herramientas colaborativas. La OECD (Organización para la Cooperación el Desarrollo Económicos) define Open Science como: "hacer los resultados de estudios fundados públicamente - publicaciones y datos de investigación – accesibles a todo el mundo a través de un formato

digital sin ninguna restricción", pero es más que eso. Open Science pretende extender los principios de franqueza en todo el ciclo de investigación, fomentando compartir y colaborar con el resto cuanto antes posible cambiando así la forma en la que la ciencia e investigación funcionan hasta el momento.

Easy Scholar Ranking se adhiere al movimiento de Open Science para así compartir con el resto de la comunidad científica el artefacto que se genera en el proyecto [26].

La Figura 47 presenta el repositorio público donde se encuentra el artefacto del proyecto.

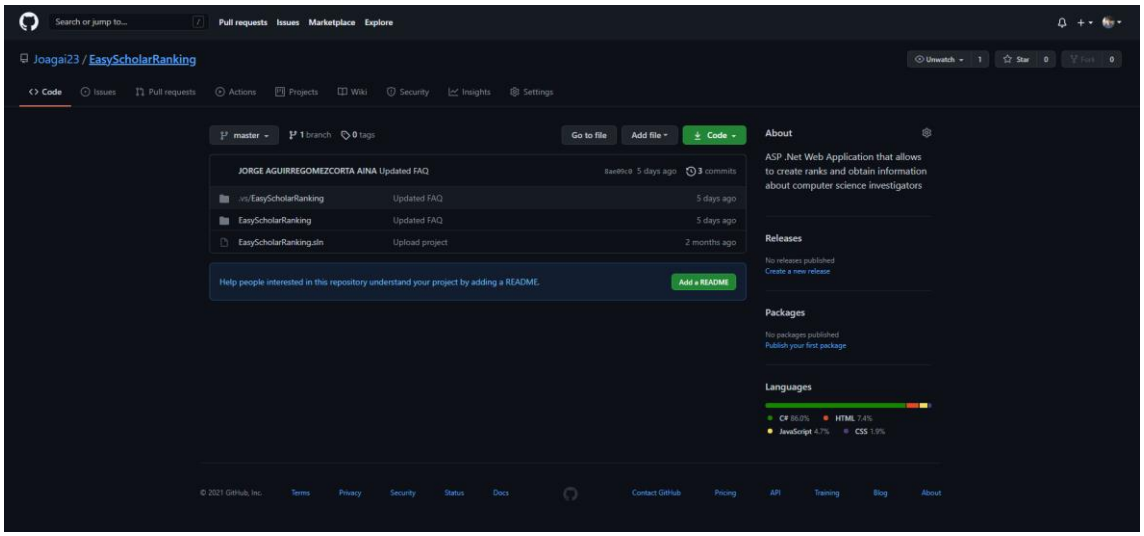


Figura 47. Repositorio público del artefacto del proyecto

8. Conclusiones

El capítulo ocho concluye el desarrollo del proyecto analizando el cumplimiento de los objetivos propuestos y futuras mejoras que se pudieran realizar para ofrecer más servicios o mejorar los actuales.

8.1. Cumplimiento de objetivos

A pesar de que se consigue completar el objetivo principal del proyecto, crear una aplicación web capaz de clasificar investigadores, algunos de los subobjetivos varían a lo largo del desarrollo.

Para poder abarcar el análisis de todos los objetivos modificados a lo largo del proyecto, se realiza una clasificación dependiendo de la naturaleza del objetivo, al igual que se realiza en el capítulo tres.

Tareas de estudio y análisis

Este grupo de tareas corresponde a aquellos objetivos necesarios para obtener información necesaria acerca de herramientas para de esa manera proceder con el desarrollo del proyecto. Todas las tareas propuestas al principio del desarrollo se cumplen una vez finalizada la implementación, aunque se añaden más a lo largo de la implementación.

Las tareas definidas a lo largo del proyecto son las siguientes:

- Buscar herramientas para leer el fichero XML
- Estudiar XQuery para obtener datos del fichero a través de BaseX
- Estudiar la api de dblp para obtener consultas
- Comparar obtención de datos a través de la api con las consultas XQuery
- Elegir y planificar metodología de trabajo
- Estudiar framework de aplicaciones web ASP.NET MVC
- Búsqueda de frameworks para mostrar gráficas
- Estudiar la plataforma de Microsoft Azure
- Estudiar api de dblp para obtener información sobre los autores
- Estudiar formas de obtener una lista de investigadores con afiliación a universidades españolas
- Estudiar algoritmos para leer un fichero de forma eficiente y rápida
- Estudiar rendimiento de la aplicación
- Estudiar y adaptar feedback del cliente
- Estudiar proyecto para obtener preguntas más frecuentes
- Estudiar y crear lista de mejoras para futuras versiones

Tareas de diseño

Las tareas de diseño sirven para obtener una imagen, esquema o modelo mental de cómo se van a realizar tareas de implementacion. Estas tareas son muy necesarias ya que ayudan a abordar implementaciones del proyecto de una forma clara y documentada. Al igual que las tareas de estudio y análisis, se logra completar los objetivos propuestos y se añaden más a lo largo del desarrollo.

Las tareas definidas a lo largo del proyecto son las siguientes:

- Crear arquitectura del proyecto en base a ASP.NET MVC
- Diseñar tabla de rankings
- Diseñar modelo para simplificar la entrada de propiedades de filtrado
- Diseñar modelo para obtener la afiliación de un investigador
- Diseñar nueva vista para obtener información sobre un investigador
- Actualizar modelo de datos para poder guardar información sobre un investigador
- Diseñar programa para juntar ficheros con lista de investigadores con afiliación a universidades españolas
- Actualizar diseño de la interfaz para mostrar nuevas opciones de filtrado
- Diseñar algoritmo de búsqueda de autores

Tareas de implementación

El último tipo de tareas que se realizan en el proyecto son las de implementación, que como el nombre sugiere se refiere a las tareas u objetivos que describen la integración de funcionalidad. En la planificación inicial se proponen una gran cantidad de tareas de implementación relacionadas con la generación de rankings. Esto se debe a que en un primer momento el objetivo del proyecto era la creación de diferentes rankings que ofrecieran una funcionalidad específica. Sin embargo, a lo largo del desarrollo se decide juntar las diferentes funcionalidades en un solo ranking con múltiples filtros. Por tanto no se completan algunas de las tareas, sino que se modifican para ser adaptada. Estas tareas u objetivos son los siguientes:

- Generar rankings a partir de la universidad de un investigador
- Generar rankings usando la lista de revistas de Google Scholar
- Generar rankings de universidades

Los rankings de universidades acaban formando el filtrado por localización, mientras que el ranking de revistas de Google Scholar se implementa como filtrado por venue.

El resto de las tareas definidas a lo largo del proyecto son:

- Implementar tabla de rankings
- Obtener datos de la consulta para generar estadísticas

- Implementar gráficos para mostrar estadísticas
- Implementar modelo para simplificar la entrada de propiedades de filtrado
- Subir aplicación web al servidor de Microsoft Azure
- Implementar modelo para filtrar investigadores en base a su afiliación
- Implementar búsqueda de consultas con la tabla de rankings de Google Scholar
- Optimizar código para facilitar su comprensión y funcionamiento
- Implementar filtrado de investigadores en base a los años que ha estado activo
- Obtener y mostrar media de publicaciones en una consulta
- Implementar programa para juntar ficheros con lista de investigadores con afiliación a universidades españolas
- Modificar fichero de investigadores para leerlo en un formato correcto en la aplicación
- Implementar algoritmo para hacer consultas al fichero
- Implementar método de búsqueda asíncrona para visualizar rueda de carga
- Corregir bugs
- Implementar algoritmo de búsqueda de autores
- Adherir proyecto a la política de Open-Science de Europa
- Implementar nueva vista de preguntas más frecuentes
- Implementar nueva vista con las conferencias y revistas usadas en la búsqueda de Google Scholar

La conclusión que se puede obtener a lo largo del desarrollo del proyecto es se cumple con todos los objetivos propuestos en la planificación, a pesar de que algunos hayan tenido que ser modificados. En definitiva, Easy Scholar Ranking es una aplicación web funcional que cubre la necesidad especificada desde el principio, crear rankings para investigadores del ámbito de las ciencias de la computación.

8.2. Futuras mejoras

El trabajo de fin de grado no abarca todas las posibles implementaciones que se podrían realizar en una aplicación web con las características de Easy Scholar Ranking, y es por eso por lo que se decide crear una lista de mejoras y ampliaciones que se podrían realizar en el futuro para cubrir el potencial que posee la aplicación web:

- Escalar la vista dependiendo de la resolución de la pantalla
- Optimizar algoritmo de localización a partir del servicio web de dblp
- Optimizar algoritmo de búsqueda de autores en el fichero
- Incluir más países en la búsqueda por localización
- Dividir Author Search en dos pantallas distintas
- Añadir gráficos a las pantallas de Author Search

- Crear archivo de autores para guardar la afiliación de investigadores cuando se busca con la api de dblp
- Utilizar un mapa de colores definido para mostrar el gráfico de Ranking Search
- Implementar un modo oscuro de la interfaz
- Añadir la posibilidad de cambiar el idioma del sistema

La implementacion actual de Easy Scholar Ranking cuenta con los requerimientos suficientes para poder cubrir una necesidad específica. Sin embargo, esta razón no puede hacer que se de por terminado el proyecto, ya que existen muchas formas de incrementar la calidad de la aplicación.

Es por eso por lo que el proyecto conocido como Easy Scholar Ranking no termina con la versión actual, sino que se continuará la implementación para construir de esa manera la mejor versión posible de la aplicación web.

9. Bibliografía

[1] Purdue University. (mayo de 29 de 2021). Genre and the Research Paper. Obtenido de https://owl.purdue.edu/owl/general_writing/common_writing_assignments/research_papers/genre_and_the_research_paper.html

[2] dblp computer science bibliography. (29 de mayo de 2021). Welcome to dblp. Obtenido de https://dblp.org/

[3] dblp computer science bibliography. (29 de mayo de 2021). Search dblp. Obtenido de https://dblp.org/search?q=Carlos%20Cetina

[4] dblp computer science bibliography. (29 de mayo de 2021). What is dblp? Obtenido de https://dblp.org/faq/What+is+dblp.html

[5] dblp computer science bibliography. (29 de mayo de 2021). Statistics - Publications per year. Obtenido de https://dblp.org/statistics/publicationsperyear.html

[6] Google. (29 de mayo de 2021). About Google Scholar. Obtenido de https://scholar.google.com/intl/es/scholar/about.html

[7] Google. (29 de mayo de 2021). Software Systems Top Publications. Obtenido de https://scholar.google.com/citations?view_op=top_venues&hl=en&vq=eng_softwaresystems

[8] ResearchGate. (29 de mayo de 2021). About ResearchGate. Obtenido de https://www.researchgate.net/about

[9] ResearchGate. (29 de mayo de 2021). ResearchGate . Obtenido de https://www.researchgate.net/

[10] Villán, V. R. (15 de marzo de 2019). Las metodologías ágiles más utilizadas y sus ventajas dentro de la empresa. Obtenido de https://www.iebschool.com/blog/que-son-metodologias-agiles-agile-scrum/

[11] StudentPlace. (9 de septiembre de 2018). Metodologia de Desarrollo de Software. Obtenido de https://studentplace98.blogspot.com/2018/09/metodologia-de-desarrollo-de-software.html

[12] W3C España. (30 de mayo de 2021). W3C España. Obtenido de https://www.w3c.es/

[13] Wikipedia. (20 de abril de 2021). eXist. Obtenido de Obtenido de https://en.wikipedia.org/wiki/EXist

[14] Wikipedia. (7 de noviembre de 2020). XQuery. Obtenido de https://es.wikipedia.org/wiki/XQuery#:~:text=XQuery%20es%20un%20lenguaje%20de,de%20Consulta%20XML%20del%20W3C

[15] Smith, S. (2 de diciembre de 2020). Overview of ASP.NET Core MVC. Obtenido de https://docs.microsoft.com/en-us/aspnet/core/mvc/overview?view=aspnetcore-5.0

[16] ChartJs. (1 de junio de 2021). Chart Samples Pie. Obtenido de https://www.chartjs.org/docs/latest/samples/other-charts/pie.html

[17] Microsoft. (1 de junio de 2021). Portal Azure. Obtenido de https://portal.azure.com/#home

[18] Microsoft. (3 de junio de 2021). Azure Advanced Tools. Obtenido de https://easyscholarrankingusj.scm.azurewebsites.net/

[19] UPV, U. P. (19 de octubre de 2017). EL ESTUDIO ECONÓMICO EN PROYECTOS DE INGENIERÍA. Obtenido de https://www.youtube.com/watch?v=rCMPE3utkEQ&ab_channel=UniversitatPolit%C3%A8cnicadeVal%C3%A8ncia-UPV

[20] Jobted. (4 de junio de 2021). Sueldo del Programador en España. Obtenido de https://www.jobted.es/salario/programador

[21] Jobted. (4 de junio de 2021). Sueldo del Analista de Sistemas en España. Obtenido de https://www.jobted.es/salario/analista-sistemas

[22] Trecet, J. (29 de marzo de 2021). IRPF 2021: así funcionan los tramos. Obtenido de https://www.finect.com/usuario/Josetrecet/articulos/como-funciona-renta-tramos-irpf

[23] Microsoft Azure. (4 de junio de 2021). Precios de App Service. Obtenido de https://azure.microsoft.com/es-es/pricing/details/app-service/windows/?cdn=disable

[24] Universidad San Jorge. (5 de junio de 2021). Easy Scholar Ranking. Obtenido de https://easyscholarrankingusj.azurewebsites.net/

[25] Fuente, G. B. (5 de junio de 2021). What is Open Science? Introduction. Obtenido de https://www.fosteropenscience.eu/content/what-open-science-introduction

[26] Aguirregomezcorta, J. (6 de junio de 2021). Joagai23/EasyScholarRanking. Obtenido de https://github.com/Joagai23/EasyScholarRanking