universidad de león

Escuela de Ingenierías Industrial, Informática y Aeroespacial

Grado en Ingeniería Informática

Sistemas de información de Gestión y Business Intelligence

Memoria:

Aplicación Web *InJour*

Autor: Álvaro Muñiz Sierra

Índice

Introducción

En este documento se tratara de exponer las razones de la creación de la aplicación *InJour* así como sus componentes y herramientas utilizadas en su desarrollo. Posteriormente se realizará un estudio de los resultados del funcionamiento de la aplicación mediante la muestra de varios ejemplos de su funcionamiento.

Luego se realizara un análisis DAFO de la aplicación junto con un análisis de las posibles líneas de futuro a desarrollar que nos ofrece la aplicación y el tema del trabajo, y para finalizar, expondré lo que este trabajo me ha enseñado y las conclusiones a las que he llegado al terminarlo.

Descripción del problema

A la hora de realizar investigaciones o artículos de investigación, además de los propios resultados que haya obtenido uno mismo de su propio trabajo, es necesario realizar una cierta investigación previa sobre otros artículos que hayan tratado el tema en concreto sobre el que se quiere trabajar ya sea, para poder utilizar ciertos resultados como una base para tus propio trabajo o utilizar ciertos resultados obtenidos por otras personas como un apoyo para reafirmar los estudios que has realizado.

También es posible que aunque tu trabajo no este destinado a publicarse en revistas, si quieras tener una fuente de información para su realización que tenga cierta fiabilidad a la hora de usarlas para poder tener una cierta seguridad.

Por esto nos planteamos dos preguntas principales: ¿Qué revista puede ofrecerme los artículos relativos a mi campo de investigación que me puedan resultar más útiles? ¿Qué artículo en concreto relacionado con mi tema de trabajo puedo utilizar estando mínimamente seguro de que sus datos y resultados nos son útiles?

Con este trabajo se espera crear una aplicación capaz de solucionar el problema planteado por la primera pregunta, permitiendo al usuario ser capaz de elegir las revistas que más le puedan interesar dentro de un pequeño grupo que se le ofrezca como recomendación por el algoritmo construido.

Aun así para poder completar la aplicación dentro del tiempo y mostrar el potencial de la misma, he decidido limitar el alcance de la misma a un solo tipo de revistas y a única área de estudio: revistas Open Source sobre Computer Science.

Herramientas utilizadas

Para la implementación de la aplicación pensada para solucionar el problema planteado en el anterior apartado se nos ha dejado completa libertad a la hora de elegir como lo haremos, a excepción de una imposición que nos obliga a usar Neo4J [4] para la creación y manejo de la base de datos.

Con esto dicho he decido realizar una aplicación web, por lo tanto la aplicación estará dividida en tres secciones principales (ya hablaremos de ellas con más detalle en secciones posteriores) y las herramientas necesarias para su desarrollo serán las siguientes:

- Frontend: Parte del código encargada de las vistas y la interacción con el usuario. Para su creación se han utilizado las siguientes herramientas:
    - Vue JS [2]: Framework ampliamente utilizado para la creación de aplicaciones web que además permite la introducción de diferentes librerías y plugins.
    - Vuetify [3]: Plugin de Vue [2] utilizado para la creación estética de la interfaz del usuario.
    - Axios: Librería utilizada para la conexión del frontend con el backend para la ejecución de las diversas funciones.
- Backend: Sección encargada de los procesos necesarios para que la aplicación funcione correctamente. Para su desarrollo se ha utilizado la siguiente herramienta:
    - Node JS: Es un entorno de código abierto multiplataforma utilizado entre otras funciones para la creación de backend de aplicaciones web. Dentro de este entorno se han utilizado los drivers de Neo4J [4] para la conexión con la base de datos y el marco de Express JS para la construcción del backend.

También hay que destacar que el lenguaje utilizado para programar tanto el backend como el frontend es JavaScript

- Base de datos: Es la parte donde se almacenara toda la información relevante de la aplicación. Para su creación usaremos el software libre Neo4J [4] que consiste en un motor para la creación de bases de datos orientadas a grafos. Neo4J [4] almacena los datos en diferentes

nodos, los cuales se relacionan entre sí indiferentemente del tipo de nodo. Una característica interesante de estas bases de datos es que las relaciones, a diferencia de las bases de datos tradicionales, pueden tener características propias y almacenar información.

Figura 1: Logo de Neo4J

Neo4J [4] utiliza un lenguaje de programación único conocido como Cypher para la manipulación y creación de sus bases de datos, el cual está basado en el lenguaje SQL.

Descripción de la aplicación

La aplicación puede ser divida en tres componentes principales: la Base de datos donde se almacenan todos los datos necesarios para el funcionamiento de la aplicación, el backend donde se produce toda la parte lógica de la aplicación y es donde se encuentran desarrollados los algoritmos, además de ser el extremo que se conecta con la base de datos para realizar todas las peticiones; y por último el frontend, la parte con la que el usuario tiene interacción y que se encarga de enviar las peticiones con los datos introducidos por los usuarios al backend.

Una vez explicada la mecánica general de cómo se relacionan las tres partes principales voy explicar cada una de ellas más detalladamente:

- Base de Datos:

Para realizar la base de datos se ha utilizado el sistema de Neo4J Desktop que se encarga de gestionar las bases de datos de nodos de conocimiento. Este gestor nos permite, mediante el lenguaje Cypher, crear el grafo de forma manual o automáticamente del CSV con los datos necesarios para el funcionamiento de la aplicación. En este caso los datos fueron obtenidos de la página de ScimagoJR [1], en la cual se puede descargar un CSV con datos de las revistas de investigación. El problema de este CSV es que ciertos apartados no vienen correctamente separados para poder crearla de forma automática, por lo que decidí extraer datos del archivo de forma manual y crearla de igual manera.

1	journal	15487660	9,444	Q1	129	52	235	2512	2868	235	9,42	48,31	United State	Northern An	University of	1996-2020	Software (Q1); Statistics and Probability (Q1); Statistics, P
2	journal	17444292	7,338	Q1	140	51	178	3218	1458	154	8,84	63,1	United State	Northern An	Wiley-Black	2005-2020	Agricultural and Biological Sciences (miscellaneous) (Q1)
3	journal	20566387	4,098	Q1	24	112	115	5373	909	111	6,83	47,97	United Kingc	Western Eur	Nature Partr	2015-2020	Computational Theory and Mathematics (Q1); Computer
4	journal	23795077	3,489	Q1	28	205	232	11005	1723	226	6,84	53,68	United State	Northern An	American So	2016-2020	Biochemistry (Q1); Computer Science Applications (Q1); E
5	journal	20573960	3,44	Q1	25	129	163	8228	1659	158	8,84	63,78	United Kingc	Western Eur	Nature Publi	2015-2020	Computer Science Applications (Q1); Materials Science (r
6	journal	20524463	3,099	Q1	48	379	670	3038	4577	594	5,93	8,02	United Kingc	Western Eur	Nature Publi	2014-2020	Computer Science Applications (Q1); Education (Q1); Infc
7	journal	15537358, 15	2,91	Q1	166	703	1804	39546	8501	1682	4,8	56,25	United State	Northern An	Public Librar	2005-2020	Cellular and Molecular Neuroscience (Q1); Computationa
8	journal	2047217X	2,639	Q1	40	159	360	5596	1948	252	7,55	35,19	United Kingc	Western Eur	Oxford Univ	2012-2020	Computer Science Applications (Q1); Health Informatics (
9	journal	17580463	2,248	Q1	55	154	422	824	1312	413	2,33	5,35	United Kingc	Western Eur	Oxford Univ	2009-2020	Agricultural and Biological Sciences (miscellaneous) (Q1)
10	journal	15337928, 15	2,219	Q1	188	184	694	8926	3950	687	3,99	48,51	United State	Northern An	Microtome P	2001-2019	Artificial Intelligence (Q1); Control and Systems Engineer
11	journal	23780967	2,165	Q1	22	149	236	6676	1202	229	5,33	44,81	United State	Northern An	AIP Publishi	2016-2020	Atomic and Molecular Physics, and Optics (Q1); Compute
12	journal	20567189	2,043	Q1	14	43	99	2071	442	97	4,14	48,16	United Kingc	Western Eur	Nature Publi	2015-2020	Applied Mathematics (Q1); Biochemistry, Genetics and M
13	journal	10943501	2,011	Q1	69	27	104	1333	315	93	2,79	49,37	United State	Northern An	University of	1997-1998, 2	Computer Science Applications (Q1); Education (Q1); Lan
14	journal	20563051	1,993	Q1	23	73	250	3550	1003	247	3,85	48,63	United Kingc	Western Eur	SAGE Public	2015-2020	Communication (Q1); Computer Science Applications (Q1
15	journal	20010370	1,782	Q1	37	142	163	11539	1116	161	7,25	81,26	Sweden	Western Eur	Research Ne	2012-2020	Biochemistry (Q1); Biophysics (Q1); Biotechnology (Q1); (
16	journal	14712105	1,626	Q1	196	763	1669	29264	5615	1620	3,52	38,35	United Kingc	Western Eur	BioMed Cen	2000-2020	Applied Mathematics (Q1); Biochemistry (Q1); Computer
17	journal	23765992	1,601	Q1	18	63	129	2993	672	129	4,15	47,51	United State	Northern An	PeerJ Inc.	2015-2020	Computer Science (miscellaneous) (Q1)

Figura 2: Muestra de los datos sacados manualmente de la base de datos.

En consecuencia el código necesario para crearla quedaría de la siguiente forma (Se puede ver el código completo en archivo “CrearBBDD.txt”):

```
CREATE (staticalsoftware:Journal{nombre:"Journal of Statistical Software",emision:"1996-2020",sjr:9.444,publisher:"University of California at Los Angeles"}),
(molecularsystem:Journal{nombre:"Molecular Systems Biology",emision:"2005-2020",sjr:7.338,publisher:"Wiley-Blackwell"}),
(quantum:Journal{nombre:"npj Quantum Information",emision:"2015-2020",sjr:4.098,publisher:"Nature Partner Journals"}),
(msystem:Journal{nombre:"mSystems",emision:"2016-2020",sjr:3.489,publisher:"American Society for Microbiology"}),
(npj:Journal{nombre:"npj Computational Materials",emision:"2015-2020",sjr:3.44,publisher:"Nature Publishing Group"}),
```

```
(pakis:Nacion{nombrePais:"Pakistan"}),
(bahrain:Nacion{nombrePais:"Bahrain"}),
(norway:Nacion{nombrePais:"Norway"}),

(categoria:Area{nombre_area:"Computer Science"}),

(software:Subcategoria{nombreCategoria:"Software"}),
(compusystemsubcat:Subcategoria{nombreCategoria:"Computational Theory and Mathematics"}),
(aisubcat:Subcategoria{nombreCategoria:"Artificial Intelligence"}),
```

Figuras 3 y 4: Muestra de código para la creación de los Nodos.

Con estos se crearon 4 tipos de Nodos que componen la base de datos:

- Los Nodos de Nacion, Area y Subcategoria cuya función es representar un país, un área de estudio y la subcategoría del área de estudio respectivamente; y cuya única propiedad dentro de los nodos es su nombre.
- Los Nodos Journal cuya función es representar las diferentes revistas disponibles y cuyas propiedades son las siguientes:
    - **Nombre**: Hace referencia al nombre utilizado por la revista en concreto.
    - **Emisión**: Se refiere al tiempo que lleva en emisión una revista haciendo referencia a su año de inicio y a su año de finalización.
    - **Sjr**: Valoración que le da la página de ScimagoJR [1] al impacto que puede tener los artículos de esa revista. Para calcularlo se utiliza una formula propia que emplea el número de citaciones de los artículos de ese año y el número de artículos publicados los últimos tres años.
    - **Publisher**: Indica el nombre de la organización encargada de la publicación de la revista en cuestión.

  Aunque del CSV se pueden obtener más datos en este caso no he querido incluirlos ya que únicamente engordarían la tabla a la hora de mostrar los resultados pero no aportarían ninguna función real o relevante.

Una vez creados los nodos se crearían las relaciones entre los mismos:

```
(crosstalk)-[:PAIS_PROCEDENCIA]->(usa),

(categoria)-[:CONTIENE]->(software),
(categoria)-[:CONTIENE]->(compusystemsubcat),
(categoria)-[:CONTIENE]->(aisubcat),
(categoria)-[:CONTIENE]->(compunetworkcomuni),
(categoria)-[:CONTIENE]->(compuscienceapimiscellaneous),
(categoria)-[:CONTIENE]->(campuscienceapi),
(categoria)-[:CONTIENE]->(compugraph),
(categoria)-[:CONTIENE]->(compuvision),
(categoria)-[:CONTIENE]->(hardwarearch),
(categoria)-[:CONTIENE]->(humancomputer),
(categoria)-[:CONTIENE]->(signaprocess),
(categoria)-[:CONTIENE]->(infosystemsubcat),


(staticalsoftware)-[:TIENE_ARTICULOS_SOBRE{valor:4,cuartil:"Q1"}]->(software),

(molecularsystem)-[:TIENE_ARTICULOS_SOBRE{valor:4,cuartil:"Q1"}]->(compusystemsubcat),
(molecularsystem)-[:TIENE_ARTICULOS_SOBRE{valor:4,cuartil:"Q1"}]->(infosystemsubcat),
```

Figura 5: Muestra de código de la creación de las relaciones entre nodos

Con esto se pueden crear tres tipos de relaciones:

- Las relaciones entre los nodos Nacion y Journal que no tienen ningún tipo de propiedad
- Las relaciones entre los nodos Area y Subcategoria que tampoco tienen ninguna propiedad
- La relación entre los nodos Journal y Subcategoria que si tienen dos propiedades:
    - **Cuartil**: Indica a que cuartil pertenecen los artículos de una revista que corresponden a dicha subcategoría. Estos cuartiles indican el impacto y relevancia de dichos artículos siendo el Q1 el más importante.
    - **Valor**: se trata de una representación numérica que le he dado al cuartil para facilitarme el programar, además está planteado para un algoritmo que no se ha implementado en este prototipo.

Con todo esto creado se obtendría una base de datos de 93 nodos y 161 relaciones que tendría el siguiente aspecto:

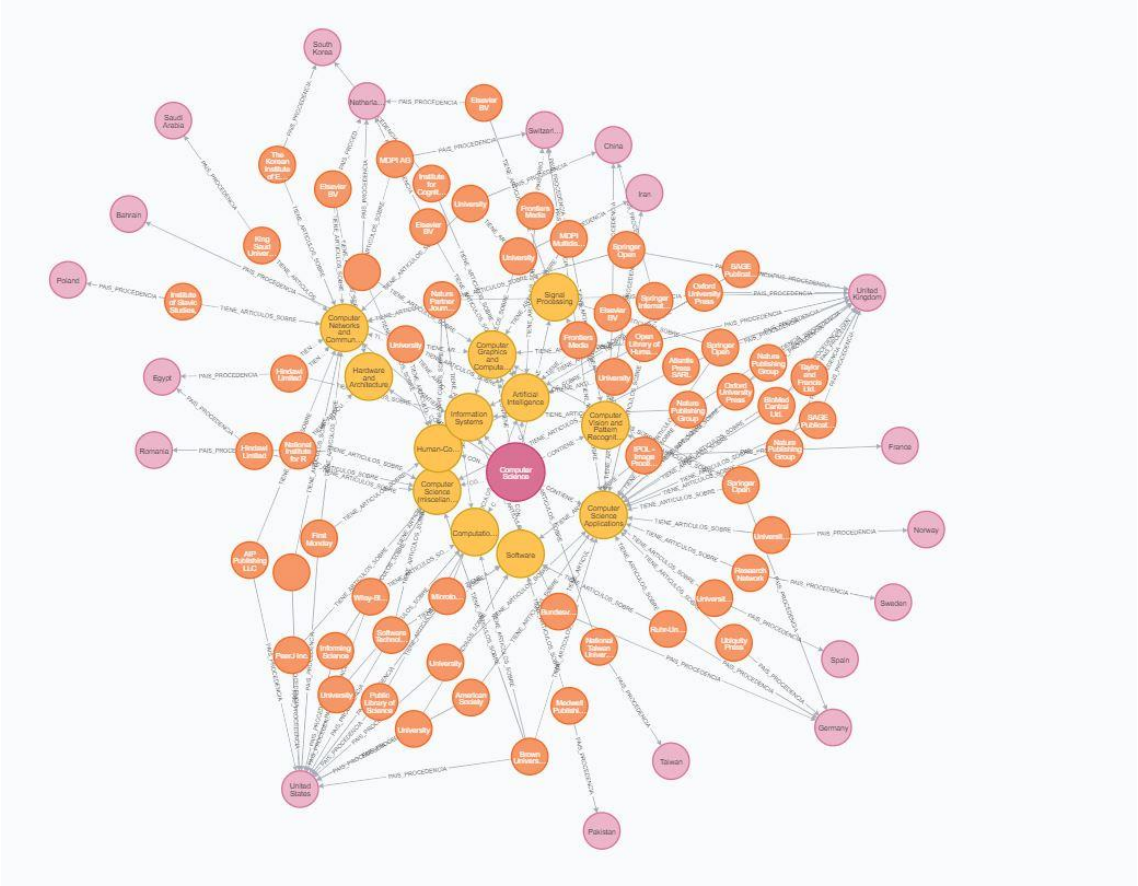


Figura 6: Modelo final de la base de datos

- Backend

Como se dijo en el aparatado anterior, para realizar el backend se ha utilizado Node JS y como lenguaje JavaScript, además de importar los drivers de Neo4J [4] para poder conectarlo con la base de datos, y express para poder realizar la conexión con el frontend. Todo el código de la implementación se encuentra en único archivo con el nombre “app.js”. El código se encuentra dividido en distintas funciones, las cuales son llamadas por el frontend para poder ejecutar alguna función y que vamos a explicar a continuación:

- cogerPaises

```
app.get("/cogerPaises",(req,res)=>{
    var session = driver.session();
    console.log("Parte de dar el Pais para hacer listado");
    var paises=[];
    var query="match(n:Nacion) return n.nombrePais"
    const resultadoPromesa = session.run(query).subscribe({
        onNext: function (result) {
            //console.log(result.get(0));
            paises.push(result.get(0));
        },
        onCompleted: function () {
            console.log(paises.length);
            res.send(paises);
            session.close();

        },
        onError: function (error) {
            console.log(error + " error interesante");
        }
    })
});
```

Figura 7: Código función cogerPaises

Esta función devuelve los países que se encuentran actualmente en la base de datos. Estos países serán posteriormente utilizados para la realización de los filtros que encontramos en el fronted y que se usaran para el algoritmo de búsqueda.

Al igual que esta función, tenemos dos funciones similares que también devuelven datos para la creación de filtros: cogerArea y cogerSubcategoria.

```
app.get("/cogerSubcategorias",(req,res)=>{
    var session = driver.session();
    console.log("Parte de dar la Subcategoria para hacer listado");
    var subcat=[];
    var query="match(n:Subcategoria) return n"
    const resultadoPromesa = session.run(query).subscribe({
        onNext: function (result) {
            //console.log(result.get(0));
            subcat.push(result.get(0));
        },
        onCompleted: function () {
            console.log(subcat.length);
            res.send(subcat);
            session.close();

        },
        onError: function (error) {
            console.log(error + " error interesante");
        }
    })
});
```

Figura 8: código función cogerSubcategoria

```
app.get("/cogerArea",(req,res)=>{
    var session = driver.session();
    console.log("Parte de dar el Area de estudio para hacer listado");
    var area=[];
    var query="match(n:Area) return n.nombre_area"
    const resultadoPromesa = session.run(query).subscribe({
        onNext: function (result) {
            //console.log(result.get(0));
            area.push(result.get(0));
        },
        onCompleted: function () {
            console.log(area.length);
            res.send(area);
            session.close();

        },
        onError: function (error) {
            console.log(error + " error interesante");
        }
    })
});
```

Figura 9: Código función cogerArea

- recomendRevista:

```
app.post("/recomendRevista",(req,res)=>{
    var session = driver.session();
    console.log("Devuelve las tres revistas mas utiles");
    var subcat=req.body.subcategoria;
    var pais=req.body.nacion;
    var query="";
    var revistas=[];
    var revistasAux=[];
    if (pais!="Elige Nación de procedencia si quiere"){
        query+="match (n)-[:PAIS_PROCEDENCIA]->(p:Nacion) where p.nombrePais='"+ pais +"' with n ";
    }
    //console.log(subcat);

    query+="match (n:Journal)-[a:TIENE_ARTICULOS_SOBRE]->(s:Subcategoria) where s.nombreCategoria='"+ subcat +"' with  n order by a.valor descending limit 3 re
    const resultadoPromesa = session.run(query).subscribe({
        onNext: function (result) {
            //console.log(result.get(0));
            revistasAux.push(result.get(0));
        },
        onCompleted: function () {
            //console.log(revistasAux);

            for (let index = 0; index < revistasAux.length; index++) {
                var element = revistasAux[index].properties;
                console.log(element);
                revistas.push(element)
            }
            res.send(revistas);
            session.close();

        },
        onError: function (error) {
            console.log(error + " error interesante");
        }
    })
});

app.post("/busquedaRevista",(req,res)=>{
```

Figura 10: código de recomendRevista

Esta función se encarga de implementar uno de los algoritmos principales de la aplicación. Utilizando las opciones introducidas por el usuario tiene que devolver un máximo de tres revistas para recomendar al usuario. También se puede observar que hay una opción que no se activa siempre dentro de la query. De esto hablare cuando tratemos el tema de análisis de resultados.

- busquedaRevistas

```
app.post("/busquedaRevista",(req,res)=>{
    var session = driver.session();
    console.log("Buscador de revistas");

    var area=req.body.area;
    var subcat=req.body.subcategoria;
    var pais=req.body.nacion;
    var nombre_publisher=req.body.nombre;
    var datoBuscado=false;
    var queryCompleta=false;
    var query="";
    var revistas=[];
    var revistasAux=[];

    /*if(area!="Elige el area de la reviosta"){
        //Para futuro En el que haya más Areas en la base de datos
    }*/
    if (pais!="Elige Nación de procedencia"){
        query+="match (n)-[:PAIS_PROCEDENCIA]->(p:Nacion) where p.nombrePais='"+ pais +"' with n ";
        datoBuscado=true;
        console.log("a");
    }

    if(subcat!="Elige una categoria"){
        query+="match (n)-[:TIENE_ARTICULOS_SOBRE]->(p:Subcategoria) where p.nombreCategoria='"+ subcat +"' with n ";
        datoBuscado=true;
        console.log("b");
    }

    if(nombre_publisher!=""){
        query+="match (n:Journal) where n.nombre='"+nombre_publisher+"' or n.publisher='"+nombre_publisher+"' return n";
        queryCompleta=true;
```

Figura 11: Fragmento del código de busquedaRevistas

Esta es otra de las funciones implementa otro de los algoritmos principales. En este caso su función es devolver la revista que el usuario está buscando a partir de los parámetros que el usuario ha introducido. Hay que tener en cuenta que el usuario no tiene que introducir o conocer todos los parámetros de la revista para querer buscarla por lo tanto esto se ha tenido en cuenta a la hora de realizar el código.

- mostrarTodasRevistas:

```
app.post("/mostrarTodasRevista",(req,res)=>{
    var session = driver.session();
    console.log("Mostrar todas las revistas");

    var query="";
    var revistas=[];
    var revistasAux=[];

    query+= "match (n:Journal) return n";

    const resultadoPromesa = session.run(query).subscribe({
            onNext: function (result) {
                //console.log(result.get(0));
                revistasAux.push(result.get(0));
            },
            onCompleted: function () {
               for (let index = 0; index < revistasAux.length; index++) {
                var element = revistasAux[index].properties;
                console.log(element);
                revistas.push(element)
            }
            res.send(revistas);
            session.close();

            },
            onError: function (error) {
                console.log(error + " error interesante");
            }
        })
});
```

Figura 12: Código de la función mostrarTodasRevistas

Esta función es se encarga de devolver todas las revistas existentes actualmente en la base de datos para que el frontend pueda mostrarlas en la tabla de resultados.

- Frontend

Esta parte aunque es la parte con la que el usuario interacciona es la que menos peso tiene en la lógica de la aplicación, ya que solo se encarga de llamar a las funciones del backend, enviarles parámetros y tratar sus respuestas para poder mostrárselas a los usuarios. Para su construcción se ha utilizado Vue [2], más concretamente un plugin de Vue conocido como Vuetify [3] para la construcción de las vistas y se ha utilizado la herramienta Axios para conectar con el backend. Además se ha utilizado JavaScript para realizar ciertas funciones necesarias para su funcionamiento.

Se ha dividido en tres vistas principales de las cuales hablare a continuación:

- Página Principal:

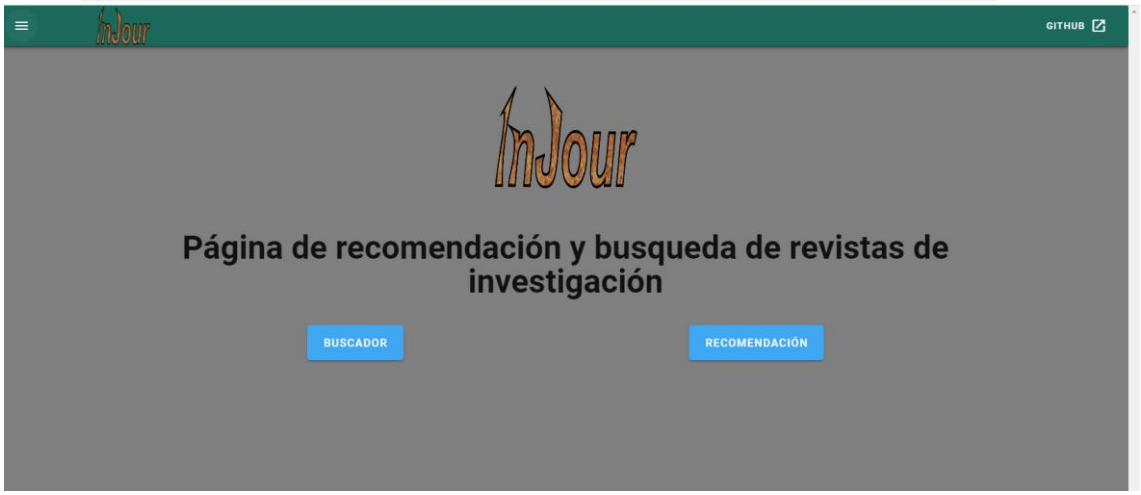


Figura 13: Pagina inicial de la aplicación

Esta vista únicamente tiene la función de dar la bienvenida al usuario y permitirle acceder rápidamente a las otras páginas donde se desarrolla la funcionalidad de la aplicación a través de los botones azules. Tiene un menú desplegable que se abre desde el botón de la esquina superior izquierda que nos permite también viajar entre las diferentes vistas. También está usada como molde para la creación del resto de páginas para que todas

tengan así una apariencia homogénea y mantengan el menú desplegable. Por último, la página está dispuesta de tal manera que si se produjeran ampliaciones de vistas en la aplicación solo se tendrían que añadir los botones que condujesen a estas vistas.

- Página de Recomendación:

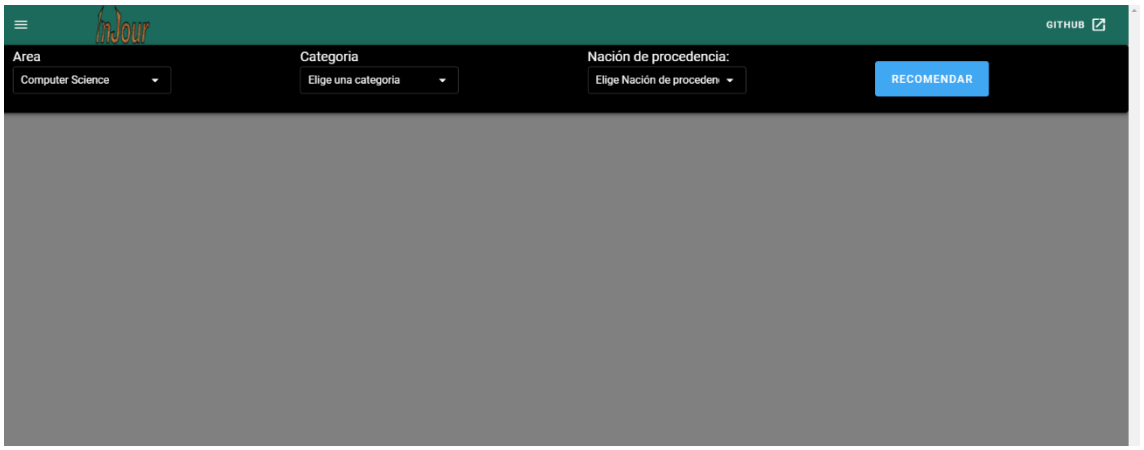


Figura 14: Pagina de recomendación

En esta vista se produce llevan a cabo las recomendaciones de las revistas. Se puede observar una apariencia básica similar a la principal incluyendo una toolbar donde se encuentran las opciones para las recomendaciones y el botón para la ejecución del algoritmo. Los parámetros a introducir se encuentran bien identificados cada uno por sus diferentes títulos que se encuentran ubicados encima de ellos.

- Página de Busqueda:

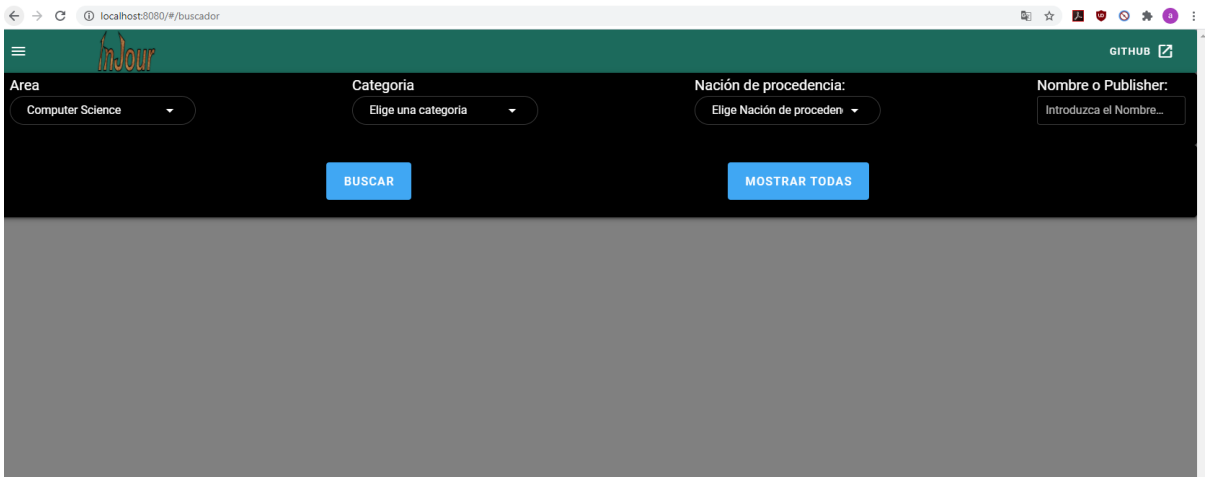


Figura 15: Pagina de búsqueda

Como la anterior, tiene una apariencia muy similar a la principal. Esta se encarga de ejecutar la parte de búsqueda de revistas o mostrar un listado con todas las revistas existentes actualmente en la base de datos. Los parámetros están debidamente marcados al igual que en la de recomendación y la función que realiza cada botón queda bastante clara con su nombre.

Análisis de Resultados

Cuando entramos en la primera página que nos encontramos es la página inicial del frontend desde la cual podemos acceder a las páginas donde se realiza la funcionalidad principal (aunque hay que recordar que tenemos un menú desplegable arriba a la izquierda con el que podemos realizar la misma función). Por ello vamos a dividir este apartado en dos partes:

- Página de Recomendaciones:

Figura 16: Inicio página de recomendación

Como se puede ver en la imagen, para realizar la recomendación existen dos campos que el usuario tiene que rellenar para que la aplicación pueda hacer una recomendación: uno obligatorio, la categoría, y otro que es opcional a elección del usuario pero que afecta a la recomendación en cierta medida, la nación de procedencia. Además se añadido dos pequeñas características a la funcionalidad debido a ser un prototipo, ya que la base de datos introducida para la aplicación es limitada: si los datos introducidos no producen ningún resultado se le avisara al usuario con una pequeña alerta. Esta alerta también se puede producir si no se introduce ningún parámetro. La otra es que los parámetros utilizados se reinician a su valor por defecto al realizar la recomendación.

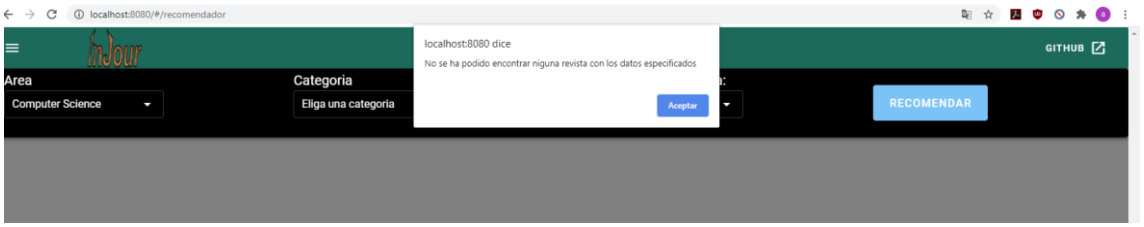


Figura 17: Alerta de error en la búsqueda de una recomendación

Con todo esto explicado voy mostrar dos ejemplos, uno general de como mostraría los resultados la página, y otro mostrando como influye en la recomendación incluir la variable opcional.

1. **Ejemplo general**: Al estar basado el problema en un área de estudios en concreto por defecto la página la tendrá seleccionada además de que no será posible su alteración. Luego seleccionaremos la categoría, en este caso 'Artificial Intelligence', y dejaremos la nacionalidad sin escoger.

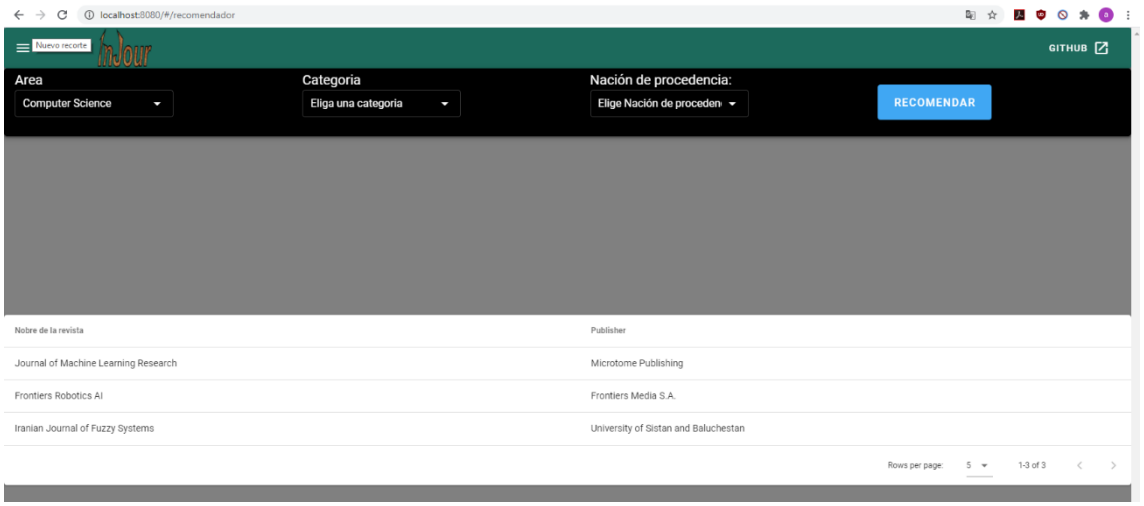


Figura 18: Resultado de la recomendación del ejemplo general

Como se puede observar en la imagen superior, la aplicación recomienda tres revistas de mayor a menor importancia (determinada por su valor SJR que indica el impacto y prestigio de la misma) para que el usuario pueda utilizar y además tenga cierta capacidad de elección.

2. **Ejemplo con el parámetro de nación de procedencia**: Vamos a utilizar los mismos parámetros que en ejemplo general, pero esta vez añadiendo ‘Netherlands’ como parámetro opcional.

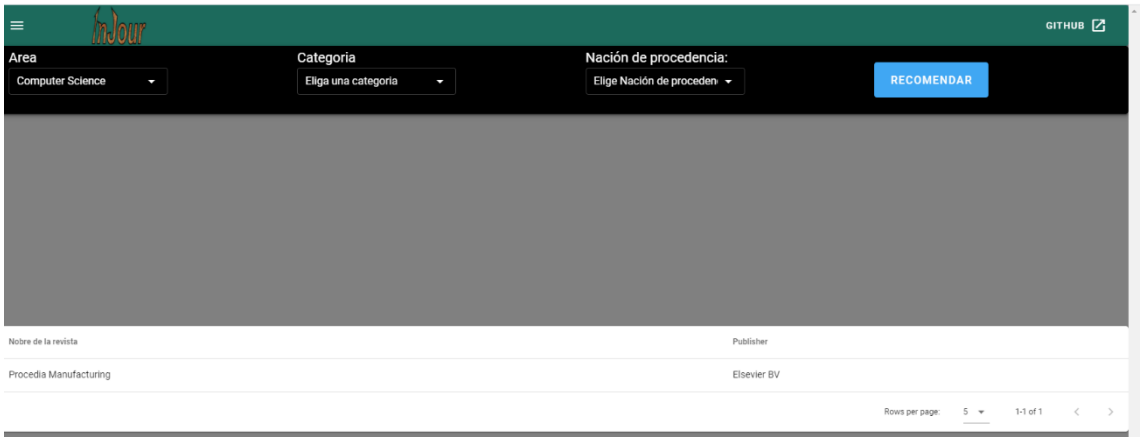


Figura 19: Resultado de la recomendación del segundo ejemplo

Se puede observar que la recomendación varía enormemente ya que primeramente el algoritmo hace un cribado de las revistas, escogiendo las provenientes de esa nación antes de ejecutar el algoritmo de recomendación propiamente.

- Página del Buscador:

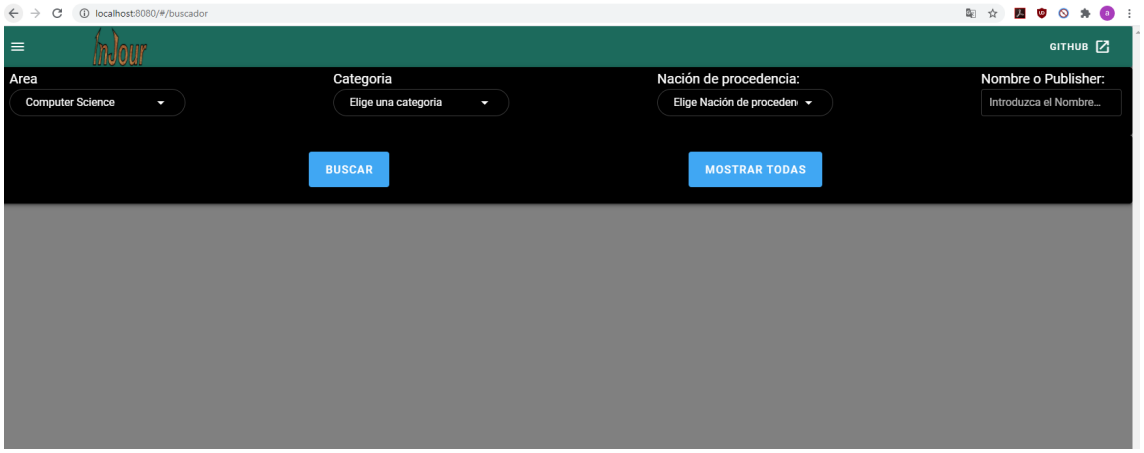
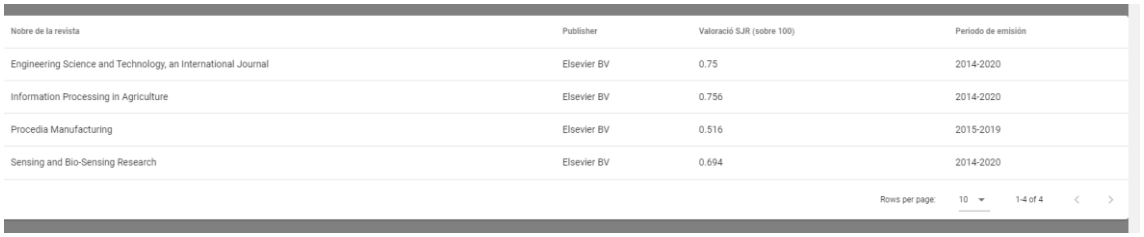


Figura 20: Página de inicio del Buscador

La página del buscador es bastante similar a la página de recomendación, tenemos los diferentes parámetros que podemos utilizar para hacer el filtrado a la hora de buscar una revista y podemos observar dos botones que ejecutan los algoritmos: el botón buscar, que ejecuta una búsqueda en la base de datos a partir de los parámetros que introduzca el usuario; y el botón 'Mostrar Todas´, que muestra todas las revistas existentes en la base de datos. Como ocurría en el apartado de recomendación si el usuario no introduce ningún parámetro o los resultados de la búsqueda con los parámetros introducidos son infructuosos, saltara una notificación indicándoselo. También decir que en cuadro de texto donde se puede introducir el nombre, podemos poner el nombre de la revista o del Publisher.

A continuación voy a mostrar dos ejemplos del funcionamiento de la página, uno con la función de mostrar todas las revistas y otra con una búsqueda utilizando los diferentes parrametros.

- Ejemplo de búsqueda: utilizaremos el cuadro de texto para buscar las revistas del siguiente Publisher: Elsevier BV, sin tener en cuenta las categorías que tienen sus artículos.

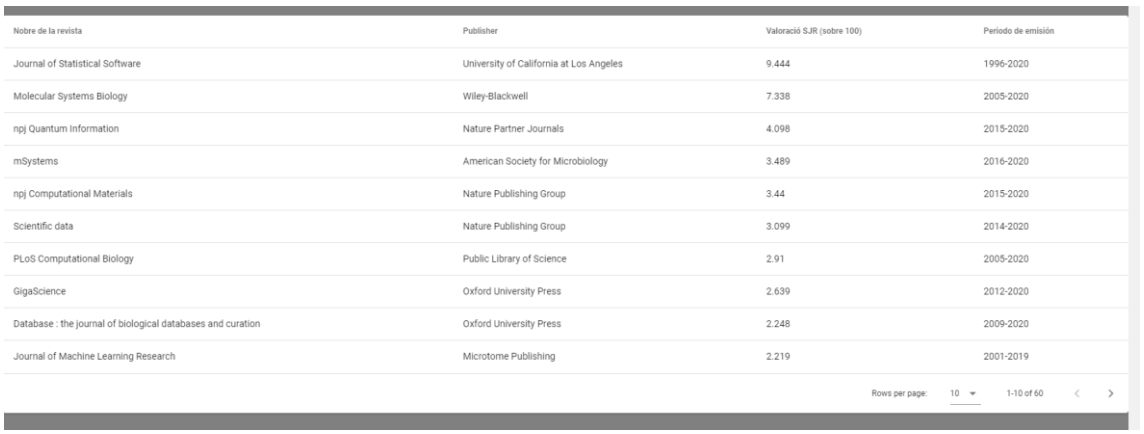


Nobre de la revista	Publisher	Valoració SJR (sobre 100)	Periodo de emisión
Engineering Science and Technology, an International Journal	Elsevier BV	0.75	2014-2020
Information Processing in Agriculture	Elsevier BV	0.756	2014-2020
Procedia Manufacturing	Elsevier BV	0.516	2015-2019
Sensing and Bio-Sensing Research	Elsevier BV	0.694	2014-2020

Rows per page: 10 1-4 of 4

Figura 21: Resultado de la búsqueda

- Ejemplo de mostrar todas las revistas:



Nobre de la revista	Publisher	Valoració SJR (sobre 100)	Periodo de emisión
Journal of Statistical Software	University of California at Los Angeles	9.444	1996-2020
Molecular Systems Biology	Wiley-Blackwell	7.338	2005-2020
npj Quantum Information	Nature Partner Journals	4.098	2015-2020
mSystems	American Society for Microbiology	3.489	2016-2020
npj Computational Materials	Nature Publishing Group	3.44	2015-2020
Scientific data	Nature Publishing Group	3.099	2014-2020
PLoS Computational Biology	Public Library of Science	2.91	2005-2020
GigaScience	Oxford University Press	2.639	2012-2020
Database : the journal of biological databases and curation	Oxford University Press	2.248	2009-2020
Journal of Machine Learning Research	Microtome Publishing	2.219	2001-2019

Rows per page: 10 1-10 of 60

Figura 22: Resultado de mostrar todas

Dentro de esta tabla podemos escoger la cantidad de revistas que se muestran dentro de la tabla, y también podemos ordenarlas a partir de su valoración, pulsando donde pone ‘Valoración SJR’.

DAFO

- **Debilidades**

La aplicación presenta varias debilidades importantes:

Primeramente la aplicación usa algoritmos bastante simples por lo que aunque cumplen su función es probable que al ampliar la base de datos de forma exponencial para albergar todos los datos no incluidos en el prototipo y que se pueden obtener de la página de ScimagoJR[1] ralenticen la obtención de los resultados.

También al estar planteado de una manera más general, las recomendaciones ofrecidas pueden no satisfacer completamente las necesidades del usuario. Aunque esto se ha tratado de subsanar ofreciendo 3 revistas en cada recomendación, no quiere decir que aun así cumpla de forma concisa con la necesidad del usuario. Una forma de solucionar esto será tratada en el siguiente apartado.

Por último, el tiempo utilizado para el desarrollo ha sido relativamente corto por lo que se pueden ver errores menores que necesitan ser pulidos pero que no afectan al correcto funcionamiento de la misma, esto se puede ver sobre todo en la interfaz que es la parte visible por el usuario.

- **Amenazas**

Se trata de una aplicación que aunque ofrece de forma distinta una solución al problema planteado, en esencia no es una nueva invención por lo que los usuarios pueden continuar usando otras páginas que ofrecen servicios similares. Además los datos utilizados para la creación de la aplicación están disponibles de forma gratuita en la página de donde los he obtenido por lo que cualquiera podría replicar la aplicación o mejorarla.

- **Fortalezas**

La interfaz es simple y accesible, lo que permite a los usuarios utilizar a los usuarios las diferentes funcionalidades de la aplicación sin tener que haberla utilizado previamente para aprender su uso.

Aunque la base de datos necesita ser actualizada periódicamente para mantenerse al día con los cambios en las revistas, los valores importantes que afectan a los algoritmos no varían a lo largo del tiempo o tienen una variación mínima en periodos cortos, por lo que el mantenimiento de la base de datos resulta bastante simple y no afectaría el realizar estos mantenimientos dentro de periodos de tiempo bastante largos.

Aunque el algoritmo es simple y como he comentado en debilidades si lo implementamos para todas las revistas de investigación existentes y todos sus campos posibles puede tener ciertos retardos que aunque cortos sean visibles para el usuario, marcado en el contexto de este problema de una sola área de estudio, sus resultados son instantáneos.

- **Oportunidades**

La base de datos actual es fácilmente ampliable lo cual permite pasar de un prototipo a una aplicación viable debido a la forma en la que se ha estructurado la misma base de datos. Además, las opciones de expansión de la aplicación así como la cantidad de funciones posibles que se pueden añadir son enormes por lo que si se desarrolla correctamente, puede llegar a ser una aplicación muy útil para cualquier usuario interesado en la investigación o incluso a estudiantes para la realización de sus trabajos.

Líneas de futuro

Con la temática propuesta para este trabajo y la aplicación que he realizado existen varias ramas por las que se puede ampliar el proyecto, por lo tanto para una mayor organización, voy a dividir este apartado en tres, teniendo en cuenta el tiempo que se tardaría en implementar o en hacer evolucionar la aplicación:

- A corto plazo:

Aunque esta se trata de una aplicación que tiene una función de prototipo, la base principal sobre la que se asienta es los datos obtenidos de la página de ScimagoJR [1]. Por lo tanto para poder ampliar la aplicación primeramente habría que crear una base de datos que se pudiese crear o actualizar y ampliar a partir de los CSV que se obtienen de la propia página ya que en este momento los datos de la aplicación son una recopilación hecha a mano de los datos de estos CSV. Para ello actualmente se me ocurren dos maneras principales de abordar este problema: la primera es buscar la forma mediante el uso de Cypher para poder adaptar el importe de los datos de forma adecuada para la base de datos ya que hay parte importante de ellos que se encuentran de forma que no pueden ser extraídos fácilmente para la forma en la que está planteada esta base de datos; y la segunda sería crear un script para poder reorganizar los datos del CSV de forma automática para que puedan ser fácilmente importados a la base de datos (aunque esta opción se saldría del estudio de Neo4J [4] y llevaría más tiempo que probablemente que la primera opción).

- A medio plazo:

Dentro del cuaderno de trabajo OSF he planteado la posibilidad de la creación de un algoritmo similar al que usan otras muchas páginas o aplicaciones. Se trataría de un algoritmo que mostraría revistas relacionadas con los temas de la revista que el usuario está buscando, pero además estas revistas tienen que tener una cierta relevancia dentro de la categoría con la que hacen relación con la revista buscada. Para implementarlo tendríamos también que un 'layout' para crear páginas individuales para las diferentes revistas donde se pudiesen mostrar las características de estas junto con los resultados de este algoritmo.

- A largo plazo:

Dentro de este apartado podríamos ampliar la aplicación de dos posibles formas:

La aplicación está pensada para que cualquiera pueda acceder a sus servicios sin necesidad de registrarse, es decir las recomendaciones son generales independientemente de si el usuario podrían interesarle otros campos de las revistas que pide que le recomienden. Para corregir esto, habría que cambiar la aplicación de manera que, aunque se mantenga los algoritmos y la parte ya creada de la aplicación, se puedan crear usuarios. A partir de estos usuarios y las búsquedas que hiciesen se crearía un algoritmo que mediante machine learning pudiese ofrecer recomendaciones especializadas a las necesidades y los campos en los que suele investigar el usuario.

Además de la modificación anteriormente mencionada, otra forma de ampliar la aplicación es añadir los artículos publicados en estas revistas para poder resolver el problema planteado de forma completa. Esto tendría una serie de cambios bastante grandes dentro de la aplicación ya que obligaría a modificar e incluso en algunos casos rehacer los algoritmos ya implementados dentro de la aplicación. Además, la información respecto a estos artículos no es de fácil acceso y se van añadiendo artículos de forma constante por lo que el mantenimiento y actualización de la base de datos aumentaría de forma enorme.

Lecciones Aprendidas

Una vez repasados todos los aspectos del trabajo realizado y la dinámica utilizada durante el proceso de realización del mismo he llegado a las siguientes conclusiones:

Mi experiencia con las bases de datos en general y con el lenguaje SQL en particular nunca ha sido la mejor; ya sea por problemas a la hora de implementarla dentro de los programas que he realizado, problemas con las interfaces a la hora de crear las bases de datos, o la complejidad a la hora de realizar las 'querys' para obtener los datos en SQL, etc.; pero con este trabajo al menos he dejado de aborrecerlas. Ya sea el tipo de base de datos utilizado por Neo4J [4], el cual resulta mucho más intuitiva para los usuarios a la hora de su creación ya que están basados en grafos, o el lenguaje Cypher utilizado para la creación de las 'querys' necesarias para los algoritmos utilizados o la obtención de los datos ha conseguido que sea capaz de trabajar de manera más o menos eficiente con ella. Desde mi experiencia actual, el lenguaje Cypher si lo tuviese que comparar con el SQL, tendría que decir que resulta mucho más sencillo de implementar y más intuitivo para el usuario a la hora de trabajar con él ya que su aplicación es mucho más directa en relación con el modelo de la base de datos. Dicho todo esto, si tuviera que recomendar un sistema y lenguaje de bases de datos a alguien sin experiencia previa, le recomendaría Cypher y el modelo de grafos (en este caso Neo4J [4]) para realizar su trabajo, ya que excluyendo el periodo de aprendizaje, el cual es el más complejo en cualquiera de los procesos que se realizan desde mi punto de vista, el desarrollo le será mucho más cómodo que con SQL.

Respecto a la metodología de trabajo. He sido capaz de organizarme para ser capaz de tener realizados los diferentes apartados del proyecto dentro de unos márgenes de tiempo establecidos bastante aceptables respecto a otros trabajos que he realizado en solitario. Aun así, puedo decir que esto tiene un margen de mejora bastante grande, ya que ciertos apartados del mismo se podrían haber realizado en un menor periodo de tiempo y de manera más eficiente si hubiese compartido dudas y diferentes cosas aprendidas con los diferentes compañeros de la asignatura durante las primeras etapas de la realización del mismo. A parte de esto, también puedo mejorar en la organización de tareas cuando tengo ciertos eventos que influyen en mi capacidad de trabajo, como pueden ser exámenes u

otros eventos repentinos, ya que debido a estos motivos el proyecto estuvo sin avances importantes por casi dos semanas.

Por último, quiero hablar sobre el cuaderno de trabajo OSF que he realizado durante la investigación y desarrollo del proyecto. Lo considero una herramienta bastante útil y que puedo utilizar ya sea para futuros trabajos de la universidad o bien para mis propios proyectos, ya que me permite mantener un cierto orden en los mismos y además como forma de revisar mis avances o la metodología usada que me podría ser útil para otros proyectos. Aun así, centrándome en el realizado para este trabajo, tengo que decir que, aunque estoy bastante contento en como lo estructurado de una manera general y sobre todo en el apartado de investigación; creo que en el apartado donde hablo del desarrollo del proyecto tengo que mejorar de una manera bastante sustancial: aunque la estructura de este apartado está bien distribuida para reflejar los diferentes componentes de la aplicación, el desarrollo del mismo durante los diferentes periodos de tiempo no lo está, ya que mi idea de usar las marcas de tiempo que se guardan cuando realizas cambios en la página no pueden ser usadas de manera tan exacta a como tenía pensado en un primer momento. Esto es debido a que no estoy acostumbrado a mantener este tipo de cuadernos donde voy manteniendo un registro del proceso por lo tanto ha habido avances semanales que no han quedado bien registrados ya que los cambios en la Wiki han sido escritos más tarde de lo que deberían. Por lo tanto, debería mejorar en este aspecto para futuros proyectos.

Fuentes y Referencias Bibliográficas

1. ScimagoJR: https://www.scimagojr.com/
2. Vue: https://vuejs.org/
3. Vuetify: https://vuetifyjs.com/en/
4. Neo4j: https://neo4j.com/

- Blog Jortilles: http://blog.jortilles.com/neo4j/
- VueJS crear aplicación con múltiples layouts: https://nmariasdev.medium.com/vue-js-crear-una-aplicaci%C3%B3n-con-m%C3%BAltiples-layouts-bcfdede95ba6