



UNIVERSIDAD NACIONAL DE EDUCACIÓN A DISTANCIA

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

Proyecto de Fin de Grado en Ingeniería de las Tecnologías de la Información

FUSION DE BASES DE DATOS

FERNANDO DEVÍS RODRÍGUEZ

Dirigido por: FERNANDO LÓPEZ OSTENERO

Curso 2020-2021: Convocatoria de Octubre

FUSIÓN DE BASES DE DATOS BIBLIOGRÁFICAS



FUSION DE BASES DE DATOS

Proyecto de Fin de Grado en Ingeniería de las Tecnologías de la Información
de modalidad *general*

Realizado por: FERNANDO DEVÍS RODRÍGUEZ

Dirigido por: FERNANDO LÓPEZ OSTENERO

Fecha de lectura y defensa:

Tabla de contenido

Resumen	5
Abstract	5
Introducción	6
CAPITULO 1: La elección del entorno de ejecución	7
Programa, Interfaz Web, Servicio Web... ..	7
El publico objetivo de la “fusión de bases de datos bibliográficas”	8
Los navegadores web	8
Páginas Web Vs Aplicaciones Web Vs Apps Móviles	9
La elección final.....	11
CAPITULO 2: La elección del entorno de desarrollo	12
La pila o “stack” tecnológico para el desarrollo web*	12
La elección final: XAMPP	13
El Front-End. Usabilidad y accesibilidad.....	15
HTML, CSS y el diseño responsive.	15
Puestos a pedir que sea estéticamente moderno y atractivo.	16
Reusabilidad.	16
Twitter Bootstrap	17
El Back-End: El núcleo de la programación.....	18
Servidor	18
Aplicación	19
Base de Datos.....	20
Herramientas para el desarrollo.....	22
Pegados que me servirán	51
Conclusiones	51
Bibliografía	52
Anexos.....	52

FUSIÓN DE BASES DE DATOS BIBLIOGRÁFICAS

Resumen

El objetivo de este TFG consiste en asentar las bases de un sistema de fusión de bases de datos bibliográficas, de forma que no sea necesario realizar una búsqueda en diferentes lugares especializados.

Funciona mediante una sola búsqueda y unifica de toda la información posible en un solo lugar, además de extender su funcionalidad a la modificación, eliminación o agregación de más datos e incluso poder exportar dicha información a diferentes formatos.

Palabras Clave: Google Scholar, DBLP, BibTex, bibliografía, publicación, paper, búsqueda, unificación, PHP, MySQL, EndNote, XML, CSV, stack, responsive, usabilidad, accesibilidad, CRUD.

Abstract

The objective of this bachelor's degree Final Project is to lay the foundations of a system for merging bibliographic databases, so that it is not necessary to search in different specialized places.

It works by means of a single search and unifies all possible information in one place, as well as extending its functionality to the modification, elimination or aggregation of more data and even being able to export this information to different formats.

Introducción

La elección de esta temática ha obedecido a diferentes factores que lo hacían de mi interés; como es la programación de aplicaciones web, el uso y manejo de bases de datos, y por supuesto, adentrarme un poco mas en el mundo de las publicaciones científicas y ver como se gestionan sus búsquedas y referencias, mas allá de los buscadores genéricos.

En mi ámbito laboral, y pese a dedicarme más a la administración de sistemas e implantaciones de ERP, estoy adentrándome cada vez más en la programación, por lo que además de lo aprendido todos estos años en el Grado, también puedo aplicar los conocimientos que estoy adquiriendo en mi trabajo diario empresarial, y viceversa; este proyecto también me está ayudando a realizar nuevas aplicaciones en ella.

Y, siguiendo con esta relación entre universidad y entorno laboral, el desarrollo de este PFG, permite aplicar un gran número de aspectos clave en lo que laboralmente se conoce hoy en día como “Full Stack Developer”, cubriendo además de la programación apartados estudiados como:

- Usabilidad y accesibilidad: diseño de la interfaz de trabajo
- Gestión de Bases de Datos
- Aplicaciones Web
- Ingeniería de Software
- Seguridad.
- Despliegue
- Pruebas, etc..

CAPITULO 1: La elección del entorno de ejecución

Tal y como se ha comentado en el texto introductorio, la creación de un software de fusión de bases de datos, es un excelente ejercicio, ya no sólo para poner en práctica todo lo aprendido durante todos los años del grado, si no que además brinda la oportunidad de poder utilizar las herramientas y tecnologías actuales que se usan en ámbitos laborales; desde la primera parte del ciclo de vida del software, es decir, la planificación del proyecto, hasta su despliegue. Estas fases del ciclo se verán desarrolladas obviamente en esta memoria.

Una vez entendido el enunciado y teniendo claro el objetivo, me surgió la primera (de muchas posteriores) duda.

Programa, Interfaz Web, Servicio Web...

Si leemos el enunciado, en la parte de “componentes del sistema” se especifica que casi todo será “un programa”, o “una interfaz web”

Caso 1: “El primero será un programa que tome como entrada dos o más URLs de bases de datos bibliográficas....”

Caso 2: “El segundo será un programa que permita buscar en la base de datos bibliográfica resultante...”

Caso 3: “El tercero será una interfaz web que dará acceso a los programas anteriores...”

Por lo tanto, aquí surge la primera decisión respecto al entorno elegido; ¿uso un “programa” compilable tradicional ejecutable?, ¿lo hago en java?, ¿en C?, ¿Visual Basic?, ¿y con un IDE o un editor de texto? Y en estos casos; ¿Cómo lo entregaré?, ¿Cómo lo despliegue al presentarlo? ¿Lo compilo para Windows, usando una máquina virtual desde mi ordenador con Mac OS X? ¿Uso Vagrant para manejar el ciclo de desarrollo de la máquina virtual?, ¿tal vez Docker?

Cualquiera de estas opciones sólo derivaría en limitaciones y en problemas a la hora del despliegue y su presentación. En ese momento, todavía ignoro que tipo de defensa de proyecto se utilizará en estos tiempos de pandemia. Por lo que aquí empiezo mi

FUSIÓN DE BASES DE DATOS BIBLIOGRÁFICAS

debate interno para tomar esta decisión “crítica” que va a influir completamente en todo el posterior desarrollo de esta tarea.

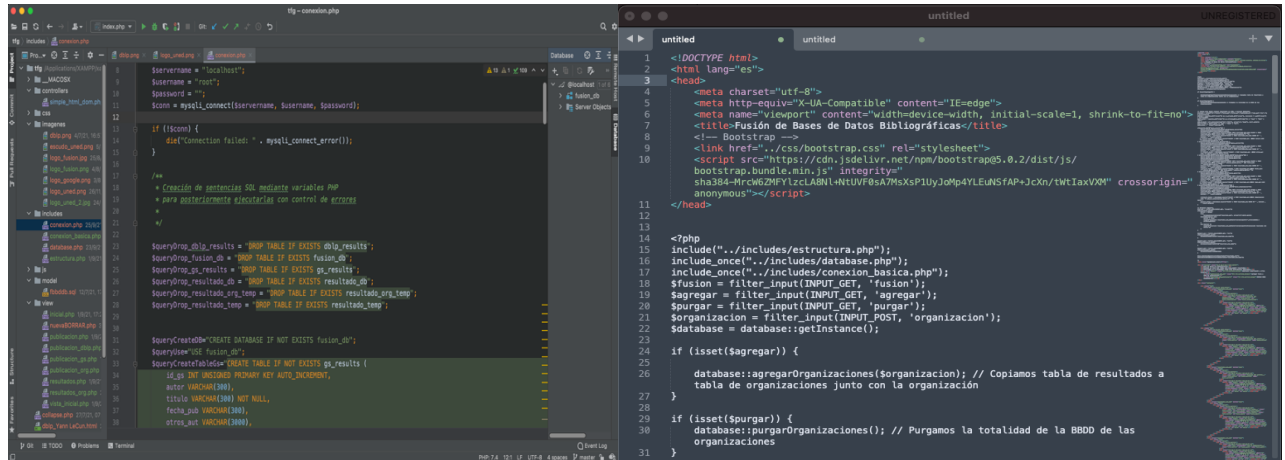


Figura: IDE (Jet Brains PHP storm) vs Editor de Código (Sublime Text 3)

El publico objetivo de la “fusión de bases de datos bibliográficas”

Una de las primeras observaciones que debemos tener en cuenta en la producción de un bien o servicio, es a quien se dirige dicho bien. En este caso y teniendo en cuenta la temática abordada, el usuario final serían instituciones y usuarios de carácter investigador o académico que necesitan realizar consultar y realizar gestiones rápidamente sobre bases de datos bibliográficas.

Una vez definido nuestro “publico objetivo”, el siguiente paso sería averiguar desde dónde se van a realizar dichas peticiones a las bases de datos.

Los navegadores web

A priori, y seguramente unos pocos años atrás, la decisión sobre que plataforma de destino elegiríamos para nuestro objetivo, hubiera tenido en cuenta sobre todo el hardware del usuario, tendríamos que haber realizado alguna investigación o estadística del hardware que usan investigadores, universidades, bibliotecas, etc. y tenerlos muy en cuenta para el desarrollo de la aplicación.

Hoy, en pleno 2021 ya no es una cuestión tan crítica como anteriormente, ya que, aunque efectivamente el usuario dispondrá o de un PC (o Mac, o incluso Linux), también

se disponen ya de otros dispositivos como tabletas, o el ya omnipresente “smartphone”. Es decir, el acceso a nuestra aplicación puede venir de un conjunto bastante heterogéneo de dispositivos, tal y como ocurre hoy en día con la mayoría del software dirigido al público en general.

Afortunadamente, ya desde hace varios años, INTERNET y mas específicamente, la WWW y el protocolo HTTP es el nexo entre gran cantidad de software y el usuario final, independientemente del dispositivo utilizado. Por lo que el público objetivo de este trabajo, en su gran mayoría disponen de un navegador web y conexión a internet.

Por lo tanto; la entrega del producto final será definitivamente UNA APLICACIÓN WEB accesible desde cualquier tipo de navegador, aprovechado pues la gran ventaja de poder acceder desde cualquier tipo de dispositivo, entre otras que veremos más adelante.

Páginas Web Vs Aplicaciones Web Vs Apps Móviles

Una vez seleccionada la plataforma de ejecución (un navegador web), nos encontramos que en la actualidad disponemos por un lado los conceptos de “aplicación web” y “páginas web” y, por otro lado, las aplicaciones móviles o “apps” cuyo crecimiento ha sido exponencial este último lustro gracias a la disponibilidad masiva de teléfonos inteligentes y tabletas entre el público en general.

Para poder profundizar un poco entre las similitudes y diferencias entre estas plataformas de ejecución, deberíamos antes tener una pequeña perspectiva histórica, obviamente no es objetivo de esta memoria realizar un compendio histórico de internet, pero si tener en cuenta esta gran evolución que ha tenido en muy poco tiempo.

Durante mucho tiempo; el lenguaje* HTML y el protocolo HTTP, debido a su objetivo inicial, fueron planteados como un sistema para poder visualizar contenido en navegadores, sin tener mucha mas pretensión respecto a la interacción con el usuario. La mayoría de las aplicaciones de software se desarrollaban para diferentes plataformas de ejecución, pese a que la gran mayoría de sistemas instalados en el mundo son

FUSIÓN DE BASES DE DATOS BIBLIOGRÁFICAS

Windows, otros sistemas pueden ser Mac, Linux, Unix, o incluso a nivel empresarial nos podemos encontrar con multitud de sistemas operativos. Es más, incluso dentro de la plataforma Windows existían diferentes plataformas de ejecución (x86, x64), por lo que cada aplicación debía ser ejecutada (y desarrollada) en diferentes plataformas, ofreciendo el fabricante sus opciones disponibles.

Sin embargo; la evolución del protocolo HTTP, permitió la integración y la creación de nuevos lenguajes de programación que podían incluso “mezclarse” con las últimas versiones de HTML. Estos “verdaderos” lenguajes de programación pueden ser interpretados en tiempo de ejecución por los navegadores actuales, e incluso pueden ser desarrollados para que parte de la ejecución se realice en el servidor, de manera que se “libera” al navegador de realizar tareas, permitiendo un completo aplicativo con la gran mayoría de dispositivos que puedan disponer de un navegador y por supuesto, una conexión a internet. En la actualidad es algo muy común, sin embargo, este gran paso del protocolo HTTP permitió esta revolución, logrando que multitud de aplicaciones puedan ser escritas en lenguajes de programación web, sin tener en cuenta la plataforma de destino.

Ahora; ya tenemos la tecnología necesaria para poder hacer completas aplicaciones ejecutables desde un navegador y no estar limitados, no obstante, estamos viviendo un auge de las “apps” para dispositivos móviles, que deben ser generalmente escritas y ejecutadas en diferentes plataformas, actualmente el 90% se escriben para Android o iOS. ¿Es esto una “involución” respecto a las webs? ¿Por qué el auge de estas “apps” si necesitan versiones diferentes y desarrollo algo diferente para cada plataforma? ¿Por qué solo existen para dispositivos móviles, y (generalmente) su correspondiente en PC es una página web?

La respuesta a estas preguntas también deben ser respondidas teniendo en cuenta el contexto evolutivo*, pero en la actualidad la existencia de estas apps se deben a dos factores fundamentales; la usabilidad y su idoneidad para el “marketing”.

Respecto a la usabilidad; debemos tener en cuenta la heterogeneidad de los dispositivos móviles, tanto respecto al hardware (marcas, procesadores, **tamaños de pantallas**) como al software (sucesivas versiones de Android e iOS). La “app” como tal permite un mayor control y optimización sobre el dispositivo sobre el que está

funcionando, y además facilita bastante su uso al usuario, y además **suelen tener sistemas de notificaciones y extracción de datos** del usuario mas completos que su equivalente web (pensemos ya simplemente en la localización), por lo que son una “mina de oro” para “minar” datos. Además en la actualidad existen lenguajes y “frameworks” de desarrollo de aplicaciones móviles que prácticamente permiten compilar el mismo código para Android e iOS, por lo que una de sus principales desventajas se desvanecen. Ejemplos de las mismas serían Flutter, Ionic, etc..

La elección final

Después de barajar las diferentes opciones disponibles, en nuestro caso, y salvo las habituales excepciones contamos con este escenario:

Usuario: personal docente, investigadores, estudiantes, etc..

Plataformas habituales de estos usuarios: generalmente PC's ya que el software no sólo va a servir para realizar consultas, sino se extiende con funcionalidades de personalización y construcción y exportación de las bases de datos resultantes.

Teniendo en cuenta estos dos pilares, la elección de plataforma de ejecución sin duda va a ser la de un **navegador web**. Dicha plataforma nos permite la flexibilidad completa cara al usuario; ya que con un simple ordenador conectado a internet podrá realizar todas las tareas propuestas por el enunciado de este PFG.

Además, iremos un poco mas allá y la ejecución tendrá en cuenta el amplio y heterogéneo conjunto de dispositivos y por supuesto de ~~usuarios~~ personas, por lo que conceptos que en la actualidad son factores clave como la usabilidad y accesibilidad, han sido tenidos en cuenta para todo el ciclo de uso de la aplicación.

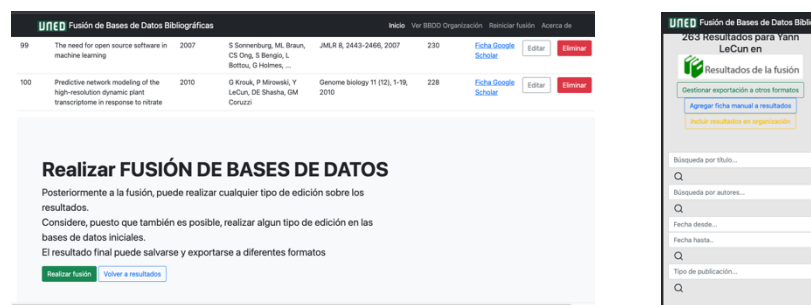


Figura: Vista desde escritorio / Vista desde dispositivo móvil

CAPITULO 2: La elección del entorno de desarrollo

Ya sabemos que queremos aprovecharnos de las ventajas actuales que nos brindan los modernos navegadores, por lo que ahora tenemos otra de las decisiones “críticas”; la elección del entorno de desarrollo.

En el mundo del software podemos encontrar diferentes acepciones cuando utilizamos el vocablo **entorno**; per se, suele referirse muchas veces al IDE* que se está utilizando para manejar el código. Realmente al referirnos a **entorno** debemos tener en cuenta todas las tecnologías que vamos a utilizar en la realización de nuestro proyecto, y dichas tecnologías, por supuesto, no incluyen únicamente lo que es el lenguaje (o lenguajes) de programación utilizados. Es mucho más.

La pila o “stack” tecnológico para el desarrollo web*

A la hora de abordar un proyecto de software, independientemente del ciclo de vida, debemos utilizar un conjunto de tecnologías adecuadas en relación con lo que es el desarrollo, entendiendo el desarrollo como la “codificación” o “implementación” del proyecto.

Estas tecnologías comprenden diferentes áreas, que podemos resumir en las siguientes:

- Sistema operativo
- Servidor web (Nos referimos al “software” como IIS, Apache, NginX..)
- Base de datos (Pueden ser relacionales o las mas recientes NOSQL)
- Interprete de lenguaje de programación o Framework *, que además suele diferir dependiendo del front-end o del back-end)

Los STACKS son agrupaciones que engloban estas áreas, su funcionamiento no es mediante combinaciones de todos a todos, si no que generalmente el uso de una determinada tecnología de estas 4, tiene más compatibilidad y rendimiento con otras del subconjunto. Es decir, si uso un servidor web **Apache**, por lo general, el sistema

operativo es **Linux**, o si por el contrario uso un servidor web **IIS** (de Microsoft), pues lógicamente el sistema operativo utilizado será **Windows** (Server).

En resumidas cuentas, estas pilas tecnológicas están configuradas de forma que cada una de ellas tiene bastante definida la tecnología de cada una de sus áreas.

Las más conocidas dentro del ámbito del desarrollo web * (hay más) son:

- XAMPP: Dependiendo del SO, puede especificarse como Wamp (Windows), Lamp (Linux), o Mamp (MacOs X). El acrónimo viene del resto de tecnologías utilizadas, “A” por el servidor web **Apache**, “M” por la BBDD **MySQL** y “P” por el lenguaje del backend, que suele ser **PHP o Perl**.
- MEAN: En este caso se usa la base de datos NoSQL* **MONGODB**, unido a **Express.JS** para manejar las solicitudes HTTP, **AngularJS** para el front-end* y **NodeJS** para el backend.
- WISA: En este caso hablamos de la pila de Microsoft, y como tal usa sus tecnologías: **Windows Server** como SO, **IIS (Internet Information Services)** como software de servidor web, **SQL Server** para la base de datos y toda la biblioteca de **ASP.NET** de lenguajes de programación.

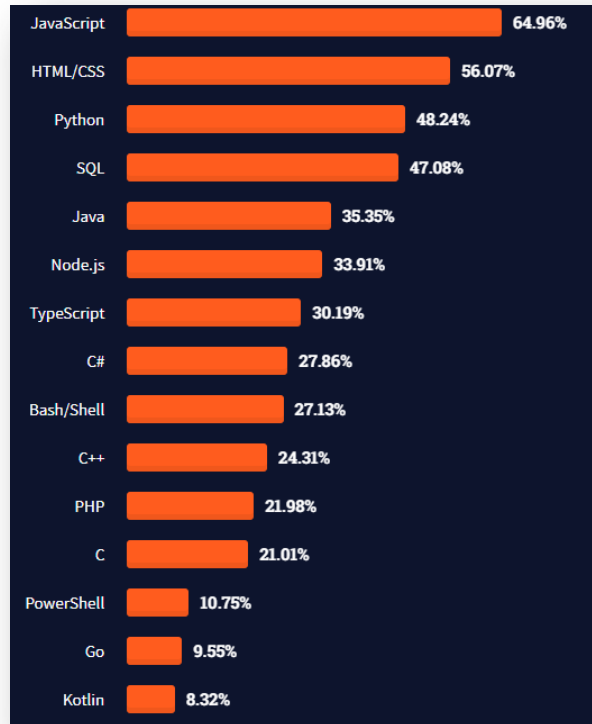
En los enlaces indicaremos muchos mas stacks de desarrollo para aplicaciones web, pero con esta información ya podemos tener una base para poder realizar la elección.

La elección final: XAMPP

Para poder elegir este Stack de desarrollo, he tenido en cuenta varios factores que han inclinado la balanza hacia esta pila. Como he comentado desde el principio, he intentado seguir realizando el enfoque de este PFG mediante un “mix” de lo que sería todo lo aprendido en el grado, y su utilización en el mundo laboral.

Si investigamos la encuesta anual de *StackOverflow** podemos comprobar como en la actualidad el lenguaje de programación para aplicaciones web mas utilizado sigue siendo desde hace varios años JavaScript:

FUSIÓN DE BASES DE DATOS BIBLIOGRÁFICAS



Sin embargo; que sea el mas utilizado desde hace varios años, no implica que la mayoría de los sitios web comerciales * están realizados en su mayoría en este lenguaje, y aunque seguramente en poco tiempo será el mas implementado, debemos tener en cuenta que un gran porcentaje de sitios web (sigo especificando empresariales) usan CMS del tipo WordPress o PrestaShop.

Estos gestores de contenido, al ser proyectos de cierta envergadura, y, sobre todo, al ser extensibles, utilizan muchos lenguajes de programación, pero el núcleo y la mayoría de su backend se desarrolla en **PHP**.

Como mi deseo realizando este proyecto ha sido acercarme lo máximo posible al mundo real del desarrollo, sin perder la perspectiva académica, he elegido el stack XAMPP (específicamente Mamp), ya que es el que utiliza PHP, y además creo que es el mas adecuado por estos motivos:

- Es una plataforma de desarrollo libre y gratuita.
- Tanto el lenguaje PHP como MySQL tienen un gran soporte y comunidad de usuarios
- Es fácilmente transportable.

Si entramos en detalles técnicos, probablemente la discusión sería, ¿y porque no usar otras pilas, por ejemplo, basadas en **Node.JS**, que además tienen ahora más proyección y su comunidad de usuarios está creciendo de manera sensible?

La elección de un “stack”, lenguaje de programación o “framework” no debe estar basadas únicamente en tendencias de mercado laboral, gustos laborales o facilidad de aprendizaje, si no que el factor principal va a ser el tipo de aplicación a desarrollar; en nuestro caso estamos ante una base de datos a la que le di un enfoque *relacional* desde un principio y para esto es mejor MySQL, que se lleva perfectamente con PHP. También cabe destacar que la ventaja principal de Node.JS es el “soporte de sub-procesos múltiples” y “soporte de servidor” incorporado. Teniendo en cuenta que la escalabilidad en este proyecto no es un factor clave, la balanza se inclina definitivamente hacia el trinomio PHP /MySQL/Apache que ofrece la pila XAMPP.

El Front-End. Usabilidad y accesibilidad.

En el stack elegido, vemos que hemos hablado casi todo del back-end, sin embargo, no hemos tenido en cuenta hasta ahora, la parte con la que interactúa el usuario, el **front-end**.

Anteriormente hemos comentado que uno de los motivos de la elección de un navegador como entorno de ejecución para el usuario era el hecho de tener mas flexibilidad tanto en los dispositivos disponibles de uso, como en la usabilidad del sistema, ¿pero como se logra este objetivo?, pues uno de los pilares es aprovechar la característica del tipo de vistas “responsive”* que tiene HTML5, y si además lo extendemos con un framework de frontend, podemos lograr un estupendo resultado, que además nos puede facilitar enormemente el desarrollo de la parte “visual” de nuestra aplicación.

HTML, CSS y el diseño responsive.

La base y lenguaje principal del front-end para el desarrollo web es HTML5, acompañado de CSS. Sin extendernos en los detalles técnicos, es de sobra conocido que HTML5 se encarga del contenido y CSS se encarga de la parte estética.

FUSIÓN DE BASES DE DATOS BIBLIOGRÁFICAS

En nuestro caso; la estética debe tener en cuenta que, pese a que el uso mayoritario será en ordenadores, no debemos dejar de lado su uso también en tabletas o teléfonos móviles, por lo que vamos a tener que lidiar con diferentes tamaños de pantallas, y por supuesto no debemos dejar de lado la “accesibilidad”, por lo tanto deberemos tener en cuenta aquellas personas que puedan tener ciertas dificultades a la hora de poder usar sus dispositivos, usando todas aquellas características disponibles a nuestro alcance para poder facilitar su uso a usuarios con diversidad funcional.

La característica “responsive” de HTML5 nos va a permitir desarrollar con facilidad nuestra aplicación para que pueda adaptarse a diferentes tamaños de pantalla, de forma transparente para el usuario, y también tendremos en cuenta además, ciertas características que agregaremos para personas con diversidad funcional, como grandes tipos de letras, contrastes bien definidos, uso de etiquetas alternativas* y situación de controles y elementos de la interfaz muy claros y definidos. Todo esto es mas sencillo de implementar mediante las herramientas “responsive” que nos ofrece HTML5 y CSS.

Puestos a pedir que sea estéticamente moderno y atractivo.

Llegados a este punto, es cuando vamos a tener que tomar otra decisión teniendo en cuenta todos los *requerimientos* de nuestro software, y además tenemos otro requisito, que, pese a que no hemos hablado de él, se sobreentiende, y es el aspecto estético. Si algo requiere la usabilidad, y pese a que hay gustos para todo, es diseñar una interfaz amigable y moderna acorde a las últimas tendencias en diseño web. No es precisamente la especialidad de los ingenieros de software, y es una tarea delegada a diseñadores, pero como hemos comentado desde el principio nuestro objetivo es ponernos en el perfil de un “Full Stack Developer”*

Reusabilidad.

La decisión respecto al apartado anterior es que si tenemos que ponernos desde cero a desarrollar la interfaz en HTML y CSS, nos ocuparía demasiado tiempo (y además no es realmente nuestra especialidad), por lo que como “fullstack” vamos a tener que asistarnos de uno de los conceptos que hemos aprendido a lo largo de estos años en el

grado, es decir la *reusabilidad* o la reutilización de código, que si se me permite, la explicación mas sencilla del concepto es “no reinventemos la rueda si ya existe”.

El caso de la reutilización de software no consiste en copiar directamente de otros, si no de apoyarnos en piezas de software existentes (precisamente diseñadas con el objetivo de su reutilización) con el fin de poder acelerar los proyectos de software.

Estas piezas de software tienen un peso en la industria primordial y su uso se ha generalizado completamente en los desarrollos de software, independientemente de su tamaño o lenguaje. Con cada lenguaje de programación tenemos a nuestra disposición varias de estas herramientas que se desarrollan específicamente para cada lenguaje o tecnología. En un principio (pese a que se siguen utilizando masivamente) podemos hablar de “librerías”, que están a nuestra completa disposición (suele ser open-source* o libres). Estas librerías, dependiendo del tipo de lenguaje, pueden tener métodos, funciones o herramientas que podemos utilizar sin tener que escribirlas nosotros, facilitando enormemente el trabajo del desarrollador.

No obstante; el siguiente paso en las librerías (y es en lo que vamos a apoyarnos en este proyecto para la parte front-end) es la construcción de diferentes *marcos de desarrollo* o más comúnmente denominados FRAMEWORKS*. Esta evolución de las librerías de software, no sólo tienen en cuenta la reutilización del código, si no también un *conjunto de prácticas* diferentes, que dependen de dicho framework elegido. Además, *el framework te da toda la estructura para un proyecto completo, integrando funcionalidades sin depender de librerías externas**.

Twitter Bootstrap

En nuestro caso; teniendo en cuenta que queremos reutilizar código para la parte frontend, pues lo suyo es investigar diferentes frameworks que se adecúen a este cometido. Efectuando una búsqueda rápida, veremos que al igual que ocurría con los “stacks tecnológicos” (incluso en ellos se incluyen algunos de estos frameworks) tenemos multitud de ellos dedicados a este aspecto de la programación, por ejemplo:

- **Vue:** bastante nuevo y diseñado para JavaScript, tiene por bandera su ligereza y facilidad de uso.

FUSIÓN DE BASES DE DATOS BIBLIOGRÁFICAS

- **React:** Creado inicialmente como una herramienta en Facebook, se ha extendido por la comunidad de desarrolladores. Está ligado a Javascript, y su característica principal es el uso del DOM virtual*
- **Angular:** Su fuerte es que su diseño y soporte es de Google, que es una garantía de estabilidad y fuerte comunidad de usuarios. Está ligado al lenguaje TypeScript
- **Bootstrap:** Su creación como herramienta para Twitter se ha extendido y es ampliamente usado como framework para plataforma web, independiente del dispositivo.

Teniendo en cuenta las opciones disponibles, el mas adecuado para el nuestro propósito es Bootstrap, por diferentes razones como facilidad de uso, trabaja directamente sobre HTML (el resto se basa en JavaScript, algo que apenas vamos a tocar), es de código libre y sobre todo se ocupa de la estética y usabilidad de la parte visual de la aplicación, sin interferir en nada mas.

El Back-End: El núcleo de la programación

En los apartados anteriores se ha detallado como funciona la “interfaz”, como el usuario interactúa con la aplicación y como se desarrolla e implementa la misma. Sin embargo, todo este resultado presentado, sería imposible si no se ha desarrollado antes mediante el “back-end”. En esta cita, se define bastante bien el concepto:

“Entonces, ¿qué hace posible la interfaz de un sitio web? ¿Dónde se almacenan todos esos datos? Aquí es donde entra el back-end.

El back-end de un sitio web consta de un servidor, una aplicación y una base de datos. Un desarrollador de back-end crea y mantiene la tecnología que impulsa esos componentes que, en conjunto, permiten que el lado del sitio web que mira al usuario exista incluso en primer lugar.

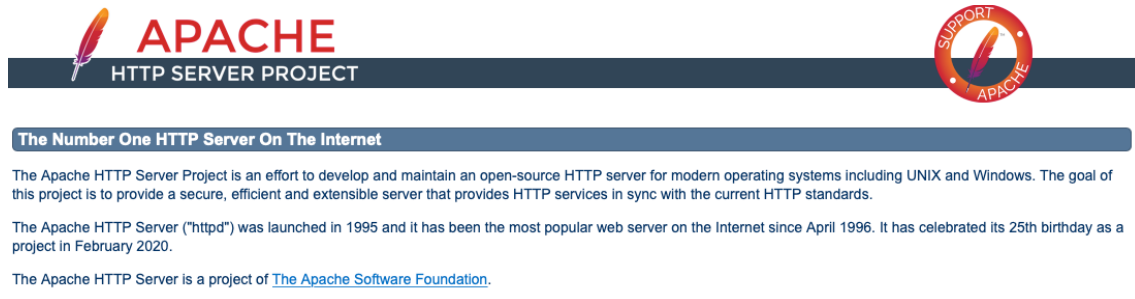
Fuente: <https://anincubator.com/diferencias-entre-frontend-y-backend-en-el-desarrollo-web/>

Al hablar del “stack” hemos comentado todo el asunto del servidor, ya que como sabemos, consta de:

Servidor

En el caso de uso de stack MAMP, el asociado es Apache (<https://httpd.apache.org>), cuyo funcionamiento, en nuestro caso es transparente para el usuario y prácticamente también para el desarrollador, ya que no necesitamos realizar ningún

tipo de “tuning” en el servidor, mas allá del encendido / apagado. Generalmente la configuración de estos entornos son tareas de sysadmins* más que de desarrolladores.



(figura: página principal del proyecto Apache Server)

Aplicación

En este caso la aplicación se refiere al lenguaje de programación en sí, que es PHP (en este caso 7.4, www.php.net). Se habla de “aplicación” como tal, ya que, para poder realizar el desarrollo en dicho lenguaje, es necesaria la instalación de la aplicación, algo que ocurre con gran cantidad de lenguajes de programación de Back-End. Nótese la diferencia con por ejemplo; JavaScript, que es directamente interpretable en el Navegador, o con Java, que actúa con una especie de máquina virtual en la máquina cliente que “traduce” las instrucciones. En el caso de PHP se instala la aplicación que interpreta el lenguaje y las librerías en el propio servidor, quien es quien se encarga de realizar la compilación a tiempo real. Y, por tanto, esto último también es una ventaja, ya que, al programar para este tipo de plataformas, no es necesario realizar compilaciones para su ejecución.

FUSIÓN DE BASES DE DATOS BIBLIOGRÁFICAS

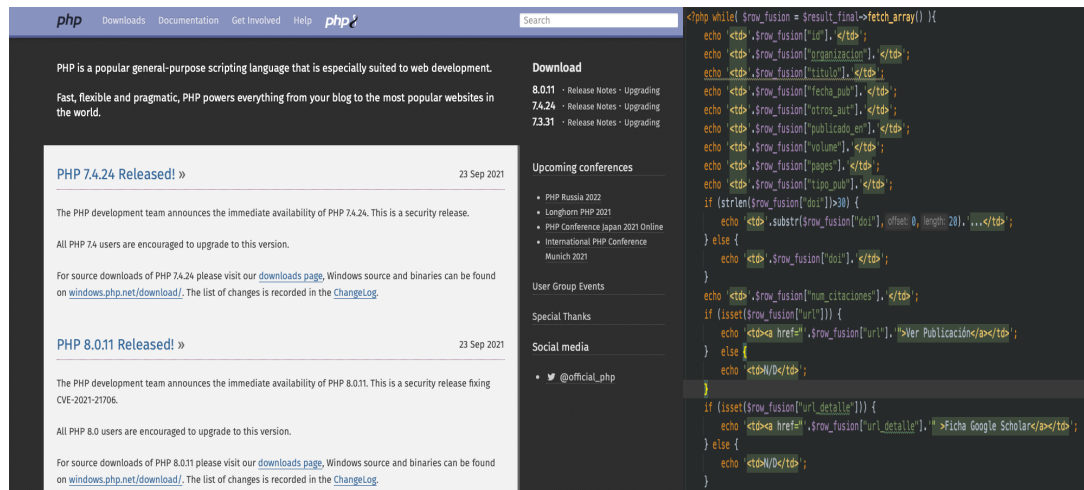


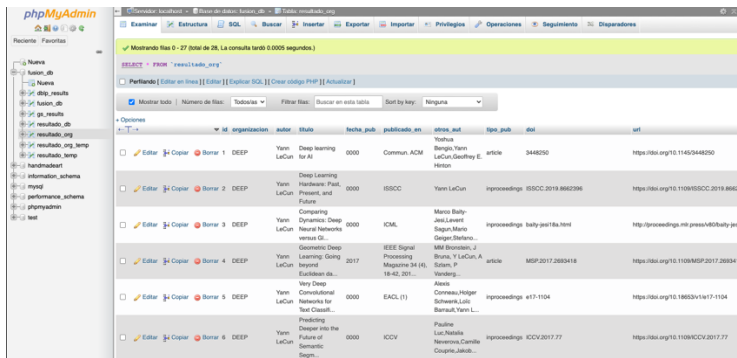
Figura: La web de PHP y algo de código (nótese el <?php)

Base de Datos

El lugar donde se guarda la información, el esquema y el modelo* de todos aquellos datos que se van almacenando, y que brinda la persistencia de los mismos, es la base de datos. Debemos tener en cuenta un pequeño matiz a la hora de estudiar las BBDD, ya que muchas veces se habla incorrectamente de ellas, haciendo referencia al **gestor de la base de datos**, y no a la base de datos en sí. Cabe destacar, además, que tenemos también en juego el **lenguaje de la BBDD**, y el **modelo** de la base de datos. Ya sabemos que al elegir XAMPP como stack de desarrollo, vamos a usar MySQL; y aquí debemos puntualizar unos pequeños detalles:

El **lenguaje** de nuestra base de datos es *mysql*, que está basado en SQL y es de código libre. Es bastante similar a otros lenguajes SQL como SQL Server o el SQL de Oracle Database.

El **gestor de la base de datos**, en este caso “gestores”, ya que he trabajado con dos de ellos; por un lado, el habitual *phpMyAdmin* incluido en XAMPP, que es una interfaz web para poder realizar todo tipo de instrucciones DDL y DDE*, y por otro lado DBWeaver, un gestor de escritorio cuya versión gratuita es mas que suficiente para nuestro propósito.



La interfaz de phpMyAdmin.

Respecto al **modelo** de la base de datos; *mysql* utiliza el llamado modelo *relacional*, como todas las bases de datos SQL, frente al más novedoso modelo NoSQL utilizado por recientes sistemas como MongoDB.

Además de que la pila XAMPP por defecto utiliza *mysql*, se ha utilizado un modelo relacional, ya que está mucho más asentado y conocido que un modelo NoSQL, además los modelos NoSQL están pensados para otro tipo de pilas, y pese a que ambos modelos hubieran funcionado en nuestra base de datos, el modelo SQL relacional es mucho más indicado:

“SQL permite combinar de forma eficiente diferentes tablas para extraer información relacionada, mientras que NoSQL no lo permite o muy limitadamente. NoSQL permite distribuir grandes cantidades de información; mientras que SQL facilita distribuir bases de datos relacionales”

Fuente: <https://www.facilecloud.com/noticias/sql-vs-nosql-which-one-should-i-use/>

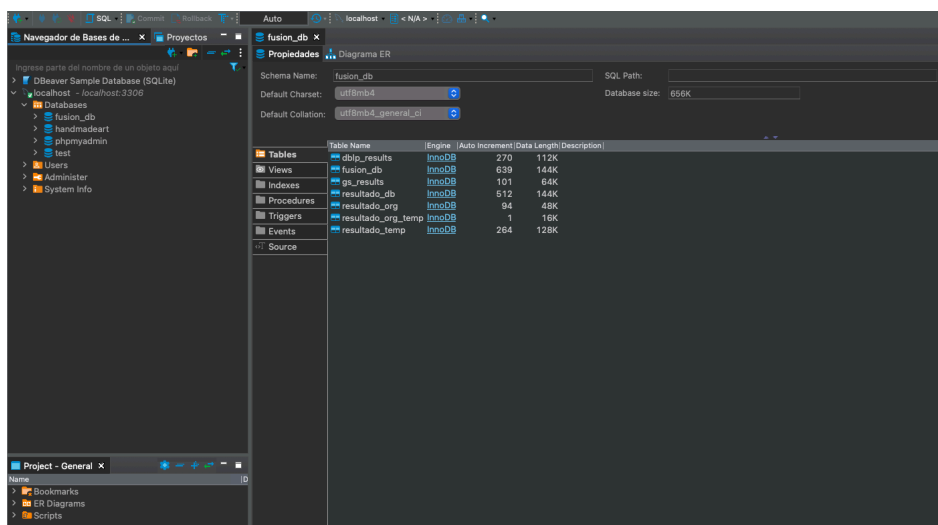


Figura: La interfaz del Gestor de BBDD “DBWeaver”

Herramientas para el desarrollo

Ya tenemos definida nuestra plataforma tanto de ejecución como de desarrollo, el siguiente paso consistiría en configurar nuestro ordenador con las herramientas de desarrollo dirigidas a las plataformas seleccionadas para nuestro proyecto y la decisión de que IDE o Editor de código utilizar.

Antecedentes:

Hardware utilizado: MacBook Pro con procesador Intel i5, 8 Gb de RAM (late 2017)

Sistema operativo; Mac Os Catalina.

Para el desarrollo del software con mi hardware tendría dos opciones iniciales:

- Usar un servidor web externo (gratuito o de pago)
- Configurar mi ordenador como servidor web local.

Como todavía no tengo decidido el despliegue final, voy a optar de momento por la opción segunda. De esta forma no dependo ni siquiera de tener una conexión a internet o de tener posibles cortes o problemas con algún servidor gratuito. Al tener el control total, mas adelante puedo desplegarlo en otro servidor o utilizar algún tipo de máquina virtual o contenedor.

También así tengo la ventaja de poder utilizar bajo mi control todo el “stack tecnológico” XAMPP, en el que obviamente existe versión para Mac Os, que funciona perfectamente.

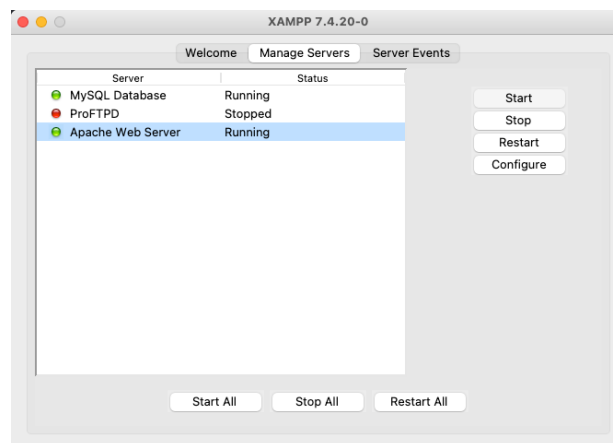


Figura: La interfaz principal de XAMPP

Respecto al IDE, desde hace un par de años Visual Studio Code es seguramente el referente para multitud de desarrolladores en el mundo, independientemente de la tecnología usada, sobre todo, gracias a sus plugins, que lo hacen ideal para cualquier proyecto, sobrepasando incluso a pesos pesados como Sublime Text, Brackets y Atom, personalmente, lo uso habitualmente como herramienta profesional, sin embargo, ya que la UNED tiene acuerdos con diferentes empresas de software, ¿porque no usar un IDE completo “de pago” dirigido a PHP y que casi toda la comunidad considera uno de los mejores? Es hora pues de lanzarse a ver que tal se desarrolla con **PHP Storm** de la conocida empresa **Jet Brains**, que gracias a la UNED podemos utilizar con una licencia gratuita con fines docentes.

Capítulo 3

Primeros pasos

Una vez seleccionada la plataforma de ejecución, desarrollo y resto de herramientas podemos ponernos en marcha con una lectura minuciosa del enunciado, punto de partida para poder realizar el diseño mediante el habitual ciclo de desarrollo del software que hemos estudiado en algunas asignaturas del grado y cuyo modelo se va a intentar seguir en la medida de lo posible.

El ciclo de vida del desarrollo del software

El SDLC (Systems Development Life Cycle) enumera las fases para realizar un análisis y validación de los requisitos del software, además de la verificación de los procedimientos, asegurando que los métodos son los convenientes. Su objetivo es realizar software de calidad con los recursos iniciales dotados, minimizando errores tardíos e intentar que los errores sean detectados en las primeras fases del desarrollo, ya que en la últimas es cuando resulta mas costoso el arreglo y las modificaciones.

FUSIÓN DE BASES DE DATOS BIBLIOGRÁFICAS

También como vimos durante el grado, existen normativas ISO al respecto del ciclo de vida, concretamente la ISO/IEC/IEEE 12207: 2017, que establece:

“Un marco común para los procesos del ciclo de vida de los programas informáticos, con una terminología bien definida, a la que pueda remitirse la industria del software. Contiene procesos, actividades y tareas aplicables durante la adquisición, el suministro, el desarrollo, el funcionamiento, el mantenimiento o la eliminación de sistemas, productos y servicios informáticos. Estos procesos del ciclo de vida se llevan a cabo mediante la participación de los interesados, con el objetivo final de lograr la satisfacción del cliente”.

En nuestro caso obviamente vamos a tener que realizar algunas modificaciones respecto al curso normal del ciclo, debido a la naturaleza de este proyecto, ya que una vez “entregado” y al no pasar a producción, veremos que existen algunas fases que no pueden / deben tenerse en cuenta. Sin embargo, creo conveniente realizar la estructura de esta memoria precisamente basándome en el proceso de ciclo de desarrollo, ya que considero que es el guion perfecto para el objetivo que nos propone el enunciado.

Pese a que existen varias interpretaciones del ciclo de desarrollo de software, vamos a basarnos en las siguientes fases:

- PLANIFICACIÓN
- ANALISIS
- DISEÑO
- IMPLEMENTACIÓN
- PRUEBAS
- INSTALACIÓN O DESPLIEGUE
- USO Y MANTENIMIENTO

Modelos del ciclo de vida del software

Además de las fases del ciclo, también estudiamos de forma profunda los diferentes modelos de ciclo de vida, modelos que afectan a la manera de abordar las etapas que hemos comentado anteriormente.

Existen multitud de modelos, aunque los mas conocidos son los modelos en cascada y los modelos iterativos.

Modelo en cascada.

Es un enfoque metodológico que ordena rigurosamente las etapas del ciclo de vida del software, de forma que el inicio de cada etapa debe esperar a la finalización de la inmediatamente anterior.

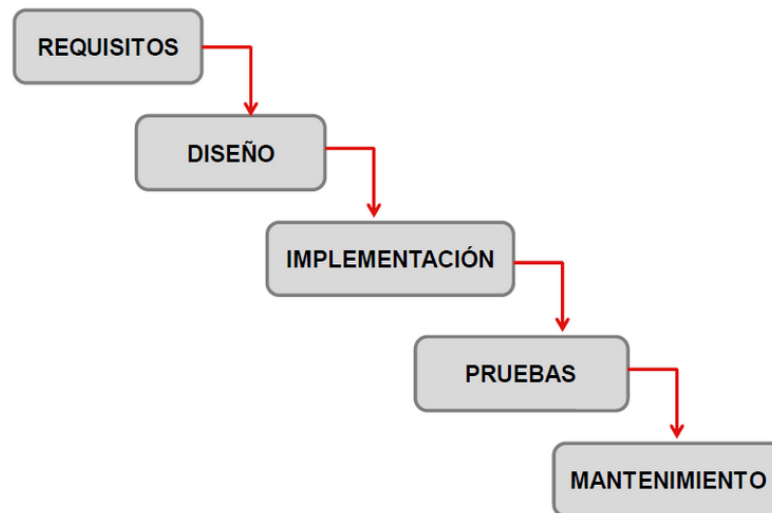


Figura: modelo en cascada

Modelo iterativo:

Consiste en la iteración de varios ciclos de vida en cascada. Al final de cada iteración se le entrega al cliente una versión mejorada o con mayores funcionalidades del producto. El cliente es quien, después de cada iteración, evalúa el producto y lo corrige o propone mejoras. Estas iteraciones se repetirán hasta obtener un producto que satisfaga las necesidades del cliente.

Este modelo se suele utilizar en proyectos en los que los requisitos no están claros por parte del usuario, por lo que se hace necesaria la creación de distintos prototipos para presentarlos y conseguir la conformidad del cliente.

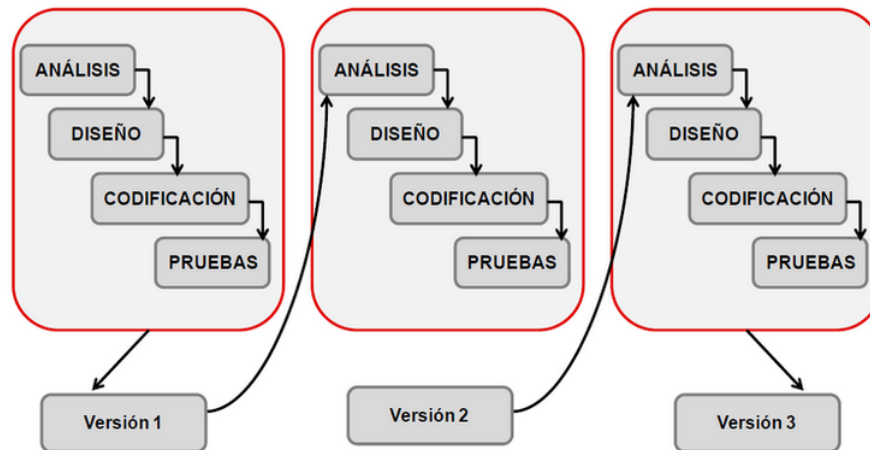


Figura: modelo iterativo

Elección de modelo

En el caso que nos ocupa, al ser únicamente una persona el desarrollador y tener bastante claros los requisitos del software, he optado por **el modelo en cascada**, ya que la evaluación final de un TFG es sobre todo el conjunto y consiste en una entrega final, sin opción a realizar futuras modificaciones. Además de tener muy claras “las necesidades del cliente” en el enunciado.

Capítulo 4: comenzamos

El siguiente capítulo contempla la parte principal de esta memoria.

Tal y como se ha comentado con anterioridad se va a aplicar el modelo en cascada del ciclo de desarrollo de software a modo de guion de todo el proyecto

Planificación del proyecto

En las fases del ciclo de vida, la planificación del proyecto suele hacer referencia sobre todo en ambientes organizativos a los recursos que van a ser utilizados, tanto económicos como de plazos de entrega. Debido a la naturaleza del proyecto, no nos preocupan los aspectos económicos, ni los recursos de asignación personales, pero creo

conveniente la realización de un marco temporal, con el fin de tener una idea de la ventana de tiempo utilizada y cuanto tiempo se consume en cada fase.

Esto puede servirnos como ejercicio para ver la diferencia de los recursos de tiempo que asignamos ahora, al principio del proyecto, a lo que será utilizado realmente a la finalización del trabajo.

El trabajo del TFG se establece como asignatura del 2º semestre del 4º curso del Grado de Ingeniería de las Tecnologías de la Información, y tiene una carga lectiva equivalente a la de 3 asignaturas, es decir 18 ECTS, lo que equivale a unas 450 horas de estudio.

En mi situación particular; mi objetivo es terminar el proyecto en septiembre, por lo que mi planificación aproximada sería la siguiente:

Fecha de inicio	Fecha de fin	Tarea	Días	Marzo	Abril	Mayo	Junio	Julio	Agosto	Septiembre
01/03/2021	15/03/2021	Lectura y comprensión y contacto con tutores	15							
01/03/2021	10/03/2021	Recopilación información general de TFG's	10							
11/03/2021	24/03/2021	Mapping de la base de datos en borrador	14							
17/03/2021	25/03/2021	Borrador de la estructura de la memoria	9							
25/03/2021	27/03/2021	Elección de entornos de ejecución y desarrollo	3							
04/04/2021	04/05/2021	Repaso general programación con PHP	31							
04/05/2021	17/05/2021	Diseño del esquema de la base de datos	14							
17/05/2021	30/05/2021	Prototipado: diseño de la interfaz de usuario	14							
01/06/2021	31/08/2021	Implementación y desarrollo	92							
01/09/2021	15/09/2021	Edición definitiva memoria	15							
01/09/2021	15/09/2021	Pruebas y casos de uso	15							
01/09/2021	15/09/2021	Instalación y despliegue en diferentes plataformas	15							
01/03/2021	15/09/2021	Memoria	199							

Análisis del proyecto.

El análisis es el proceso que tiene como fin realizar la descripción de los requisitos de la aplicación, y las características que debe tener.

Para el análisis nos vamos a apoyar obviamente en el enunciado del ejercicio, poniendo énfasis en los siguientes REQUISITOS.

- R.1 – Capacidad para fusionar diferentes bases de datos de bibliografías: el sistema va a tener en cuenta tanto a Google Scholar como a DBLP y será sencilla su extensibilidad para otras bases de datos.
<https://scholar.google.com/>
<https://dblp.org/>
- R.2 – Se podrán realizar búsquedas mediante URL, pero intentaremos extenderlo, con una sencilla búsqueda por autor.

FUSIÓN DE BASES DE DATOS BIBLIOGRÁFICAS

- R.3 – Se podrán manejar las bases de datos iniciales DBLP y Google Scholar para poder rectificar los resultados obtenidos por parte del usuario.
- R.4 - Poner herramientas a disposición del usuario para poder visualizar y exportar los datos a otros formatos.

Estos requisitos se irán comprobando a medida que la implementación se vaya desarrollando.

También, debemos mencionar a los **actores** que definen los diferentes perfiles de uso de la aplicación y pueden realizar sus **funciones**. En nuestro caso tenemos a:

- Usuario estándar: Realizar las tareas habituales de la aplicación; ver las bases de datos iniciales según su criterio de busca, edita resultados, fusiona, etc.
- Requisito deseado opcional: administrador de la aplicación: El administrador de la aplicación tendrá acceso además al **segundo caso de uso del enunciado** (generación de una única lista de publicaciones para una organización)



Figura X: Funciones / Actores

Diseño del proyecto

En este apartado nos extenderemos en la estructuración del proyecto y en el **modelado** de este. Estamos ante una aplicación web, por lo tanto, contamos con una

disposición de elementos muy diferenciados, como la base de datos sobre la que se apoya el proyecto, la interfaz web, y la implementación en sí del sistema.

En el momento que se desarrolla software de cierta complejidad con estructuras diferenciadas, debemos tener en cuenta los llamados **PATRONES DE DISEÑO**, vistos profundamente en el estudio del grado.

Existen varios de estos patrones, no obstante, se trata de elegir alguno que case con el tipo de aplicación que estamos desarrollando

Generalmente para las aplicaciones web, y en ésta concretamente que hace uso constante del *request / response* de HTTP, es adecuado implementar en la medida de lo posible el patrón **MODELO – VISTA – CONTROLADOR**.

Modelo – Vista – Controlador.

Si atendemos a la Wikipedia, la definición que tenemos del MVC es la siguiente:

“Modelo-vista-controlador (MVC) es un patrón de arquitectura de software, que separa los datos y principalmente lo que es la lógica de negocio de una aplicación de su representación y el módulo encargado de gestionar los eventos y las comunicaciones. Para ello MVC propone la construcción de tres componentes distintos que son el modelo, la vista y el controlador, es decir, por un lado define componentes para la representación de la información, y por otro lado para la interacción del usuario.¹² Este patrón de arquitectura de software se basa en las ideas de reutilización de código y la separación de conceptos, características que buscan facilitar la tarea de desarrollo de aplicaciones y su posterior mantenimiento.”

(<https://es.wikipedia.org/wiki/Modelo-vista-controlador>)

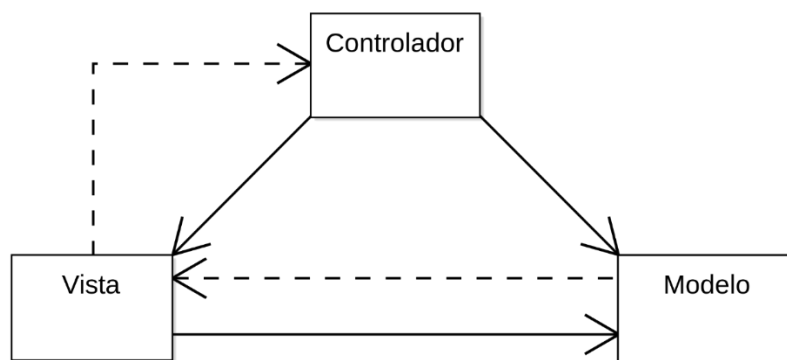


Figura: modelo vista controlador

FUSIÓN DE BASES DE DATOS BIBLIOGRÁFICAS

Así que, para nuestro proyecto, estos componentes comprenden las siguientes áreas de desarrollo, bien separadas y definidas.

MODELO: En este caso correspondería a todo el esquema relacional de la base de datos; desde la definición de tablas, sus restricciones, campos y relaciones entre ellas, así como las diferentes funciones tipo CRUD (Create, Read, Update, Delete – Creación, Lectura, Actualización y borrado) que les afecten.

VISTA: Al ser una aplicación web; la frontera delimitada por la vista es bastante clara, ya que se trata de toda la interfaz diseñada para interactuar con el usuario, teniendo en cuenta, por supuesto, todos aquellos detalles comentados anteriormente respecto a la usabilidad y accesibilidad.

CONTROLADOR: Sería el “puente” entre el MODELO y la VISTA; se encarga de gestionar las peticiones del usuario mediante el protocolo HTTP, de solicitud / respuesta. De forma que posteriormente se comunica con el modelo, realizando las funciones necesarias.

Modelado de la base de datos.

En este punto del desarrollo es cuando comienzan las dificultades, ya que debemos tener en cuenta que vamos a tener que mezclar bases de datos que pese a que tratan del mismo tema; bibliografía y publicaciones, al tomarlas de diferentes fuentes no van a ser completamente homogéneas entre sí. Es cuando llega el momento de realizar un “mapping” de las bases de datos, estudiando la estructura de las fuentes y traducirlas a un formato final, que es el que manejará nuestra aplicación.

El resultado de Google Scholar:

Si nos basamos en el ejemplo del enunciado, y utilizamos la URL del autor “Yann LeCun” en Google Scholar, nos ofrece la siguiente vista en HTML:



Yann LeCun

[FOLLOW](#)

Chief AI Scientist at Facebook & Silver Professor at the Courant Institute, [New York University](#)

Verified email at cs.nyu.edu - [Homepage](#)

[AI](#) [machine learning](#) [computer vision](#) [robotics](#) [image compression](#)

TITLE	CITED BY	YEAR
Deep learning Y LeCun, Y Bengio, G Hinton nature 521 (7553), 436-444	42189	2015
Gradient-based learning applied to document recognition Y LeCun, L Bottou, Y Bengio, P Haffner Proceedings of the IEEE 86 (11), 2278-2324	39026	1998
Backpropagation applied to handwritten zip code recognition Y LeCun, B Boser, JS Denker, D Henderson, RE Howard, W Hubbard, ... Neural computation 1 (4), 541-551	9156	1989
Convolutional networks for images, speech, and time series Y LeCun, Y Bengio The handbook of brain theory and neural networks 3361 (10), 1995	4981	1995
OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks P Sermanet, D Eigen, X Zhang, M Mathieu, R Fergus, Y LeCun International Conference on Learning Representations (ICLR 2014)	4843	2014
The MNIST database of handwritten digits Y LeCun, C Cortes	4496	1998
Handwritten digit recognition with a back-propagation network Y LeCun, B Boser, JS Denker, D Henderson, RE Howard, W Hubbard, ... Advances in neural information processing systems 2, NIPS 1989, 396-404	4230	1990
Efficient backprop Y LeCun, L Bottou, GB Orr, KR Müller Neural networks: Tricks of the trade, 9-48	4224	2012
Efficient backprop Y LeCun, L Bottou, GB Orr, KR Müller Neural networks: Tricks of the trade, 9-48	4221 *	1998
Optimal Brain Damage Y LeCun, JS Denker, SA Solla Advances in neural information processing systems 2, NIPS 1989 2, 598-605	4085	1990

Figura: Resultados de Yann LeCun en Google Scholar

En ella podemos ver los detalles principales de cada una de sus publicaciones, que corresponden a los siguientes campos o facetas:

TITLE	CITED BY	YEAR
Deep learning ⁽¹⁾ Y LeCun, Y Bengio, G Hinton ⁽²⁾ nature 521 (7553), 436-444 ^{(3),(4),(5)}	42189 ⁽⁶⁾	2015 ⁽⁷⁾

Figura: facetas o campos de una ficha de Google Scholar

FUSIÓN DE BASES DE DATOS BIBLIOGRÁFICAS

- (1) Título: "Deep Learning"
- (2) Autores: "Yann LeCun, Y Bengio, G Hinton"
- (3) Publicado en: "Revista Nature"
- (4) Volumen: "521"
- (5) Páginas: "436-444"
- (6) Numero de citas: "42189"
- (7) Año de publicación: "2015"

También es posible entrar en cada ficha para poder ver el detalle, aunque no obstante veremos mas adelante, que, al implementar el código para ver detalle, **Google nos penaliza** al tener demasiadas "requests" en poco tiempo, obligándonos a esperar o reiniciar el router para disponer de otra IP.

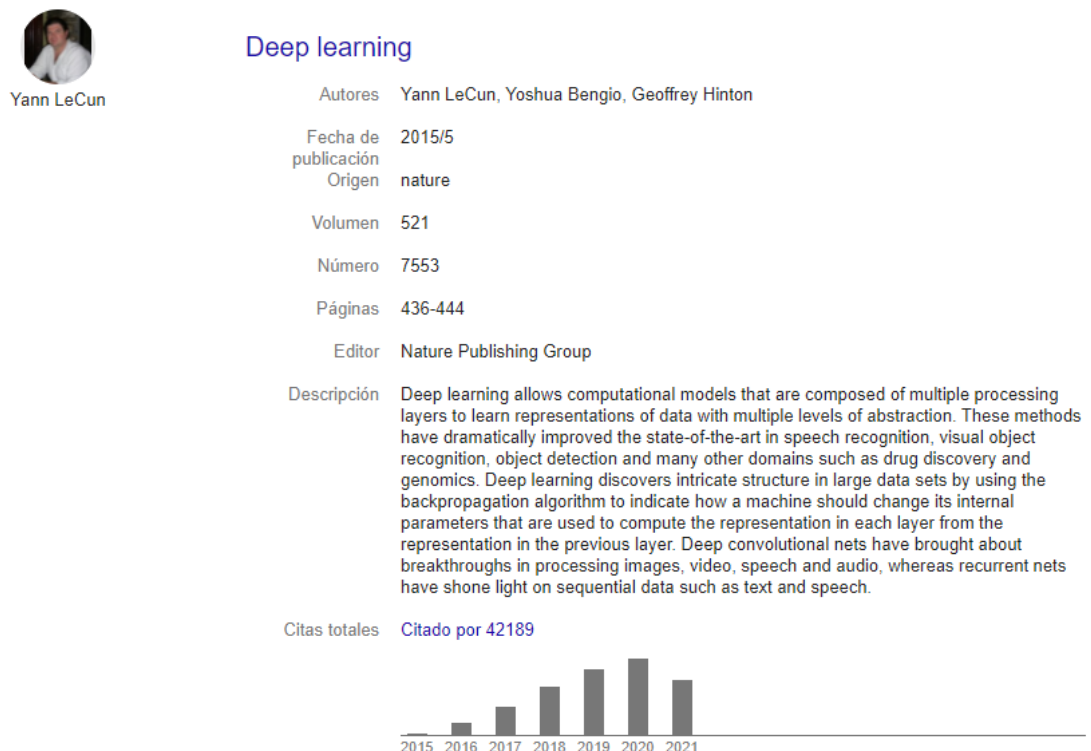


Figura: Ficha detallada de publicación en Google Scholar

En el detalle podemos disponer de un campo adicional llamado “descripción”, aunque finalmente se ha optado por no incluirlo debido a las restricciones de Google comentadas anteriormente:

Warning: file_get_contents(https://scholar.google.es/citations?view_op=search_authors&mauthors=Fernando+Ostenero&hl=es&oi=ao): failed to open stream: HTTP request failed! HTTP/1.0 429 Too Many Requests in /Applications/XAMPP/xamppfiles/htdocs/tfg/controllers/simple_html_dom.php on line 82

Figura: Error disparado por Google al extraer datos de la ficha detallada

El resultado de DBLP:

En DBLP, realizando lo mismo que para Google Scholar, obtenemos los siguientes resultados para Yann LeCun:

[–] Person information

- *affiliation:* New York University, Courant Institute of Mathematical Sciences, USA
- *affiliation:* Facebook
- *award (2018):* Turing Award

[–] 2020 – today

2021
























- [j36]    Yoshua Bengio, Yann LeCun, Geoffrey E. Hinton:
Deep learning for AI. Commun. ACM 64(7): 58-65 (2021)
- [j35]    Baptiste Rozière , Morgane Rivière, Olivier Teytaud, Jérémy Rapin, Yann LeCun, Camille Couprie 
Inspirational Adversarial Image Generation. IEEE Trans. Image Process. 30: 4036-4045 (2021)
- [c149]    Jure Zbontar, Li Jing, Ishan Misra, Yann LeCun, Stéphane Deny:
Barlow Twins: Self-Supervised Learning via Redundancy Reduction. ICML 2021: 12310-12320
- [i73]    Jure Zbontar, Li Jing, Ishan Misra, Yann LeCun, Stéphane Deny:
Barlow Twins: Self-Supervised Learning via Redundancy Reduction. CoRR abs/2103.03230 (2021)
- [i72]    Zeyu Yun, Yubei Chen, Bruno A. Olshausen, Yann LeCun:
Transformer visualization via dictionary learning: contextualized embedding as a linear superposition of transformer factors. CoRR abs/2103.15949 (2021)
- [i71]    Aishwarya Kamath, Mannat Singh, Yann LeCun, Ishan Misra, Gabriel Synnaeve, Nicolas Carion:
MDETR - Modulated Detection for End-to-End Multi-Modal Understanding. CoRR abs/2104.12763 (2021)
- [i70]    Adrien Bardes, Jean Ponce, Yann LeCun:
VICReg: Variance-Invariance-Covariance Regularization for Self-Supervised Learning. CoRR abs/2105.04906 (2021)

Figura: El resultado en DBLP

Si diseccionamos cada resultado, obtenemos los siguientes campos:

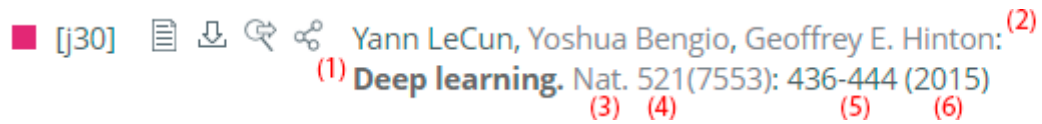


Figura: registro detallado en DBLP

- (1) Título: “Deep Learning”
- (2) Autores: “Yann LeCun, Y Bengio, G Hinton”
- (3) Publicado en: “Revista Nature”
- (4) Volumen: “521”
- (5) Páginas: “436-444”
- (6) Año de publicación: “2015”

Podemos comprobar que esta bibliografía no tiene en cuenta, por ejemplo, el dato de las “citaciones” que sí que tiene Google Scholar, por lo tanto, será un dato a tener en cuenta.

Sin embargo, así como notamos en falta el asunto de las citaciones, tenemos un acceso sencillo al XML, que al contrario de Scholar, vamos a poder utilizarlo sin ningún problema para poder ver todos los datos e importarlos a nuestras bases de datos fusionadas.

Y en este XML tenemos acceso a otros datos que no aparecen en nuestras fuentes en Google Scholar, por lo tanto, ya tenemos un punto de partida para realizar la fusión de ambas bases de datos.

```

▼<dblp>
  ▼<article key="journals/nature/LeCunBH15" mdate="2020-06-08">
    <author>Yann LeCun</author>
    <author>Yoshua Bengio</author>
    <author>Geoffrey E. Hinton</author>
    <title>Deep learning.</title>
    <pages>436-444</pages>
    <year>2015</year>
    <volume>521</volume>
    <journal>Nat.</journal>
    <number>7553</number>
    <ee>https://doi.org/10.1038/nature14539</ee>
    <ee>https://www.wikidata.org/entity/Q28018765</ee>
    <url>db/journals/nature/nature521.html#LeCunBH15</url>
  </article>
</dblp>

```

Figura: XML de registro en DBLP

Estos datos, son, por ejemplo; el DOI (una especie de código para las publicaciones, al igual que podemos hablar del ISBN para libros), y el sitio web de la propia publicación.

Google Scholar	DBLP
Titulo	Titulo
Fecha Publicación	Fecha Publicación
Autores	Autores
Publicado en	Publicado en
Páginas	Páginas
Volumen	Volumen
Descripción *	Tipo de publicación
url_detalle	url
Número de citas	doi

Esquemas de tabla para cada una de las dos fuentes:

Leyenda: **AMARILLO**: Existentes en ambos esquemas,

GRIS: Exclusivos de cada plataforma.

FUSIÓN DE BASES DE DATOS BIBLIOGRÁFICAS

* La **descripción** no será posible activar debido al “baneo” de Scholar.

La tabla fusión de resultados

Teóricamente nuestra tabla final debería contener los siguientes campos:

Resultados_fusión:
Id
titulo
autor
otros_aut
tipo_pub
publicado_en
fecha_pub
num_citaciones
doi
volume
pages
url
url_detalle
descripción*
organización.

- **id:** La identificación del autor. Sería el campo clave para identificar cada publicación
- **título:** El título de la publicación
- **autor:** El autor principal sobre el que se ha realizado la búsqueda. Generalmente este tipo de publicaciones tiene varios autores, no obstante, se guarda como autor “principal”. El término “principal” no corresponde a si es no el autor mas determinante en la publicación, es el principal para nosotros como usuarios, ya que es sobre el que se ha realizado la búsqueda.
- **otros_aut:** Se incluyen el resto de autores que han trabajado en la publicación
- **fecha_pub:** El año correspondiente a la publicación.
- **num_citaciones:** La cantidad de veces que ha sido citada dicha publicación (Google Scholar)
- **doi:** Una suerte de código identificativo para identificar unívocamente las publicaciones
- **pages:** el intervalo de páginas donde se haya la publicación escrita. No confundir con número de páginas, que en algunos casos aparece para libros completos.
- **publicado_en:** Si se ha publicado de manera dependiente de otra, aparece el origen de la misma (p. ej. revistas como “Nature”, etc..)
- **tipo_pub:** En el dominio de las citas y publicaciones bibliográficas existe un estándar de facto, que es el manejado por el formato cuyo origen es LaTeX*, llamado BibTex. Dentro de este formato, se indica el tipo de publicación referida, que suele ser una de este conjunto.
 - article
 - book
 - booklet
 - inbook
 - incollection

FUSIÓN DE BASES DE DATOS BIBLIOGRÁFICAS

- inproceedings
 - manual
 - mastersthesis
 - misc
 - phdthesis
 - proceedings
 - techreport
 - unpublished
- **url:** Acceso a la publicación
- **url_detalle:** Acceso a la ficha de Google Scholar; se ha incluido con el fin de poder ver el detalle de la misma sin incurrir en el problema de exceso de peticiones.
- **volumen:** número de volumen donde se encuentra la publicación
- **descripcion:** campo vacío para realizar las pruebas (desechado por error http de peticiones)
- **organización:** Este campo se incluye para que los administradores puedan realizar el **segundo caso de uso del enunciado**, de manera que puedan seleccionar un conjunto de publicaciones de varios autores para incluirlos dentro de una organización.

*Esquema de la base de datos **FUSION_DB***

Ya sabemos que la información va a venir de dos tablas diferentes, que se van a fusionar en una, sin embargo, hasta llegar a los resultados deseados y cumplir con todos los requisitos, van a ser necesarias varias tablas.

Además, debemos contar con que uno de los casos de uso hace referencia a poder gestionar las publicaciones dentro de una organización (en nuestro caso podremos

gestionar varias). También vamos a tener que definir el comportamiento en relación a la “persistencia” de los datos, como veremos más adelante.

Una vez tengamos las tablas “parseadas*” desde sus HTML’s correspondientes a SQL, tendremos primeramente estos resultados en estas dos tablas, una para Google Scholar y otra para DBLP:

gs_results		dblp_results	
123	id_gs	123	id_dblp
ABC	autor	ABC	autor
ABC	titulo	ABC	titulo
ABC	fecha_pub	ABC	fecha_pub
ABC	otros_aut	ABC	otros_aut
ABC	publicado_en	ABC	publicado_en
ABC	bdorigen	ABC	tipo_pub
ABC	descripcion	ABC	bdorigen
ABC	url_detalle	ABC	url
123	num_citaciones	ABC	pages
		ABC	volume
		ABC	doi

Para cada una de ellas, tendremos la siguiente **definición**:

#	Column Name	Data Type	Not Null	Auto Increment	Key	Default	Extra
1	123 id_gs	int(10) unsigned	[v]	[v]	PRI		auto_increment
2	ABC autor	varchar(300)	[]	[]			
3	ABC titulo	varchar(300)	[v]	[]			
4	ABC fecha_pub	varchar(300)	[]	[]			
5	ABC otros_aut	varchar(1000)	[]	[]			
6	ABC publicado_en	varchar(300)	[]	[]			
7	ABC bdorigen	varchar(300)	[]	[]			
8	ABC descripcion	varchar(3000)	[]	[]			
9	ABC url_detalle	varchar(3000)	[]	[]			
10	123 num_citaciones	int(10) unsigned	[]	[]			

FUSIÓN DE BASES DE DATOS BIBLIOGRÁFICAS

#	Column Name	Data Type	Not Null	Auto Increment	Key	Default	Extra
1	id_dblp	int(10) unsigned	[v]	[v]	PRI		auto_increment
2	autor	varchar(300)	[]	[]			
3	titulo	varchar(300)	[v]	[]			
4	fecha_pub	varchar(300)	[]	[]			
5	otros_aut	varchar(1000)	[]	[]			
6	publicado_en	varchar(300)	[]	[]			
7	tipo_pub	varchar(300)	[]	[]			
8	bdorigen	varchar(300)	[]	[]			
9	url	varchar(300)	[]	[]			
10	pages	varchar(300)	[]	[]			
11	volume	varchar(300)	[]	[]			
12	doi	varchar(300)	[]	[]			

Dichas definiciones vienen de operaciones DDL*

* “sentencias DDL son aquellas utilizadas para la creación de una base de datos y todos sus componentes: tablas, índices, relaciones, disparadores (triggers), procedimientos almacenados, etc. sentencias DML son aquellas utilizadas para insertar, borrar, modificar y consultar los datos de una base de datos”

Ejemplos de las operaciones DDL para la creación de dichas tablas (dblp_results)

```
CREATE TABLE `dblp_results` (  
  `id_dblp` int(10) unsigned NOT NULL AUTO_INCREMENT,  
  `autor` varchar(300) DEFAULT NULL,  
  `titulo` varchar(300) NOT NULL,  
  `fecha_pub` varchar(300) DEFAULT NULL,  
  `otros_aut` varchar(1000) DEFAULT NULL,  
  `publicado_en` varchar(300) DEFAULT NULL,  
  `tipo_pub` varchar(300) DEFAULT NULL,  
  `bdorigen` varchar(300) DEFAULT NULL,  
  `url` varchar(300) DEFAULT NULL,  
  `pages` varchar(300) DEFAULT NULL,  
  `volume` varchar(300) DEFAULT NULL,  
  `doi` varchar(300) DEFAULT NULL,
```


PRIMARY KEY (`id_dblp`)

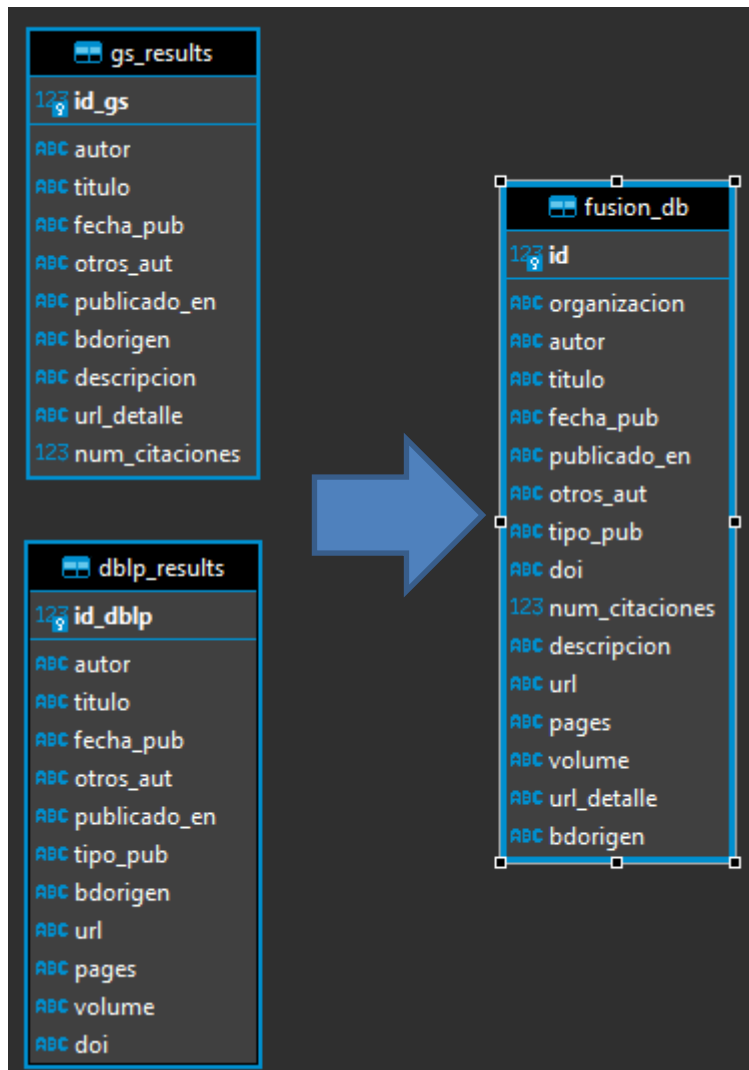
) ENGINE=InnoDB **AUTO_INCREMENT=33** **DEFAULT** CHARSET=latin1

Una vez tengamos los datos de ambas tablas, usaremos una tabla a modo de “saco” donde sumaremos todos los resultados obtenidos, sin ningún tipo de filtrado. Esta tabla la llamaremos, igual que la base de datos, **fusión_db** y tiene la siguiente definición:

#	Column Name	Data Type	Not Null	Auto Increment	Key	Default	Extra
1	123 id	int(10) unsigned	[v]	[v]	PRI		auto_increment
2	ABC organizacion	varchar(300)	[]	[]			
3	ABC autor	varchar(300)	[]	[]			
4	ABC titulo	varchar(300)	[v]	[]			
5	ABC fecha_pub	varchar(300)	[]	[]			
6	ABC publicado_en	varchar(300)	[]	[]			
7	ABC otros_aut	varchar(1000)	[]	[]			
8	ABC tipo_pub	varchar(300)	[]	[]			
9	ABC doi	varchar(300)	[]	[]			
10	123 num_citaciones	int(10) unsigned	[]	[]			
11	ABC descripcion	varchar(3000)	[]	[]			
12	ABC url	varchar(300)	[]	[]			
13	ABC pages	varchar(300)	[]	[]			
14	ABC volume	varchar(300)	[]	[]			
15	ABC url_detalle	varchar(3000)	[]	[]			
16	ABC bdorigen	varchar(300)	[]	[]			

Dicha tabla contiene toda la información que necesitamos procedente de las dos tablas de partida, y será la tabla en la que realizaremos todas aquellas **funciones** necesarias para obtener un resultado óptimo, siguiendo los requisitos del enunciado.

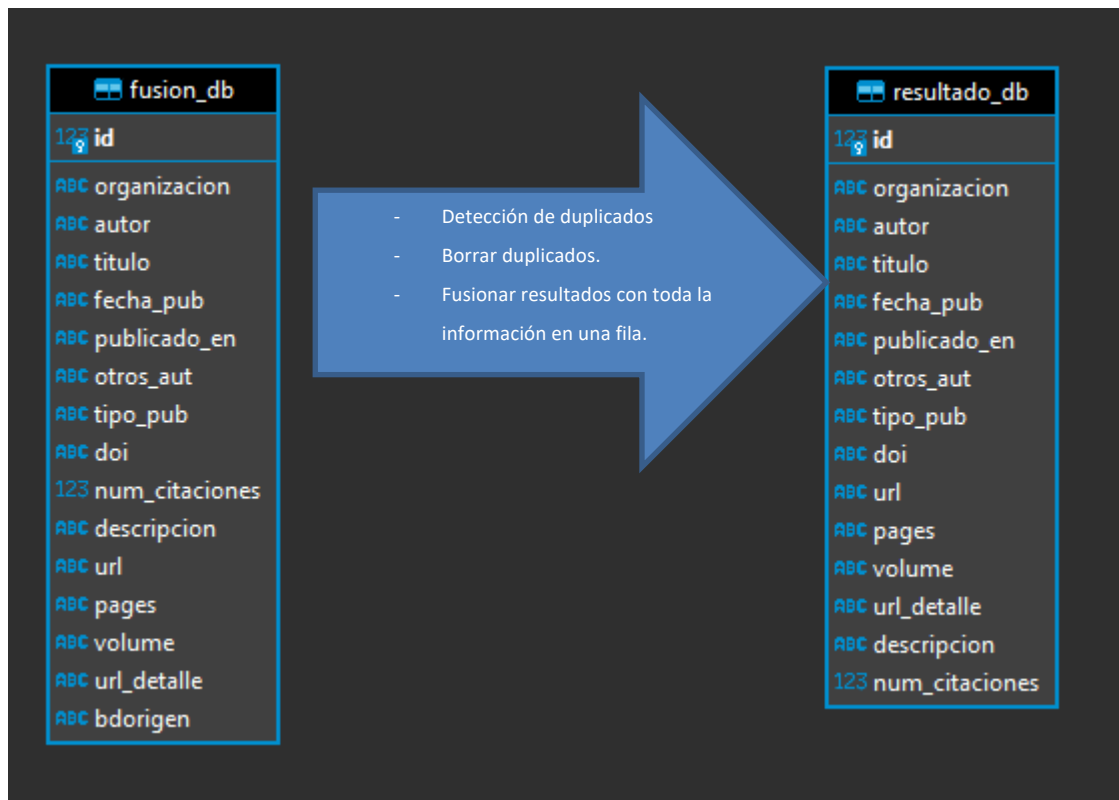
FUSIÓN DE BASES DE DATOS BIBLIOGRÁFICAS



Esta tabla **fusión_db** es pues una tabla de datos “en bruto” en la que posteriormente refinaremos en otra tabla llamada **resultados_db**, cuyos datos serán ya manipulables por el usuario y podrá realizar mediante la interfaz web varias operaciones sobre ella.

Esta tabla **resultados_db** contendrá ya la primera aproximación de la fusión.

Se trata de que el usuario deba realizar el mínimo mantenimiento posible en dicha tabla, y que en ella ya se encuentren casi todos los problemas resueltos sobre duplicados, falta de datos, etc..



Detección de duplicados, borrados y fusiones

Para detectar duplicados, el mejor sistema va a ser trabajar con cadenas de textos, ya que pese a la existencia del DOI, **no todos los documentos disponen de un identificador único y universal**, por lo que después de pensarlo detenidamente el sistema **va a basarse en los títulos para la detección de duplicados**.

En un principio había considerado la idea de tomar como duplicado a partir de una cantidad X de caracteres al comienzo del título que sean coincidentes. Sin embargo, he visto algunas fichas, que casi el mismo título, variando apenas algún carácter, pero que, sin embargo, son publicaciones diferentes, o una revisión de la anterior, por ejemplo:

57	Energy-based Generative Adversarial Network	2016	J Zhao, M Mathieu, Y LeCun	international Conference on Learning Representations (ICLR 2017), 2016	abs/1609.03126	5041-5049	article	1609.03126	1046
33	Energy-based Generative Adversarial Networks	2017	Junbo Jake Zhao, Michaël Mathieu, Yann LeCun	ICLR (Poster)	34	648-657	inproceedings	forum?id=ryh9pmcee	

FUSIÓN DE BASES DE DATOS BIBLIOGRÁFICAS

También podría darse el caso de que una publicación con **exactamente el mismo nombre** sea diferente (ediciones de años diferentes, por ejemplo), veamos este caso concreto:

Yann LeCun	Efficient backprop	2012	YA LeCun, L Bottou, GB Orr, KR Müller	Neural networks: Tricks of the trade, 9-48, 2012
Yann LeCun	Efficient backprop	1998	YA LeCun, L Bottou, GB Orr, KR Müller	Neural networks: Tricks of the trade, 9-48, 1998

Se trata también de una publicación que ha sido publicada dos veces, una primera en 2012 y otra revisada en 1998, por lo que debería tratarse **como dos registros independientes**.

Para casos así, se identificarán aquellos duplicados cuyo año no sea coincidente y se incluirán como resultados diferentes.

No creo necesario comparar con más campos, ya que podría darse el caso de estar en orígenes diferentes, pero realmente ser exactamente la misma publicación, como, por ejemplo:

NULL	Yann LeCun	A Closer Look at Spatiotemporal Convolutions for Action Recognition	2017	computer vision and pattern recognition conference (CVPR 2018), 2017	D Tran, H Wang, L Torresani, J Ray, Y LeCun, M Paluri	NULL
NULL	Yann LeCun	A Closer Look at Spatiotemporal Convolutions for Action Recognition	2017	ICML	Du Tran, Heng Wang, Lorenzo Torresani, Jamie Ray, Yann LeCun, Manohar Paluri	article 1711.11248

En este caso, estamos ante la misma publicación, publicada (valga la redundancia) en dos sitios diferentes; (conferencia CVPR, y en ICML)

En resumen; debido a la disparidad de datos, orígenes y sitios de publicaciones, no va a existir una regla que resulte exacta para realizar la fusión.

No obstante, el usuario final, podrá realizar todas aquellas ediciones que desee, tanto en los primeros resultados obtenidos por Google Scholar y DBLP para su posterior

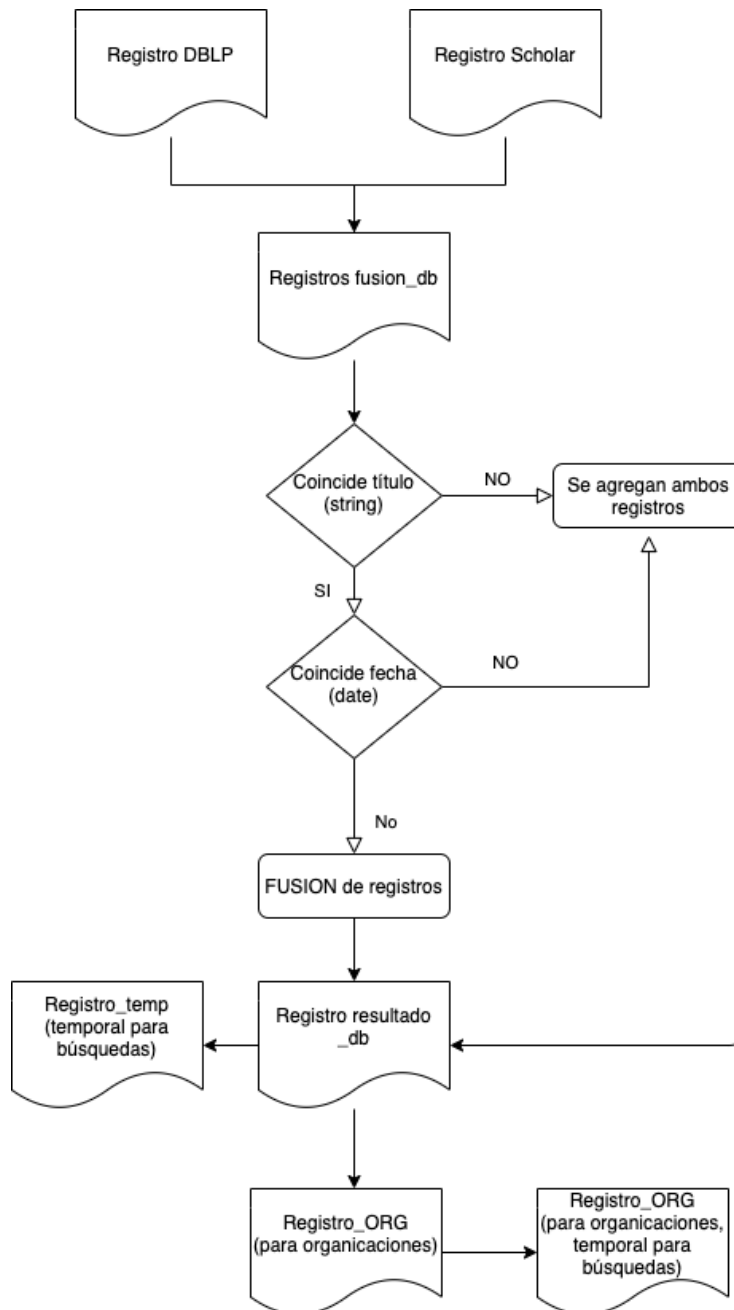
fusión, como en los resultados finales obtenidos, e incluso si decide realizar un volcado para la base de datos propia de una organización, también los datos serán editables de forma independiente para cada una de las tablas obtenidas, logrando una **persistencia** a medida del usuario.

Borrado de duplicados: no existe un borrado de duplicados como tal, si no que, tal y como se ha visto, no se agregan en la tabla de resultados aquellos registros que se consideran duplicados.

Fusionar resultados: La fusión de resultados se realiza por diferenciación, es decir, si para un mismo título no existen datos en DBLP, se obtienen los de Google Scholar y viceversa.

A continuación se puede ver en diagramas explicativos:

FUSIÓN DE BASES DE DATOS BIBLIOGRÁFICAS



Persistencia de datos

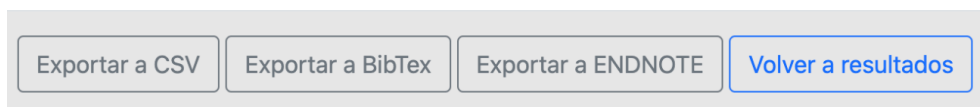
La persistencia hace referencia al guardado en el tiempo de los datos que gestionamos, independientemente de si cerramos el programa o apagamos el equipo informático.

En nuestro caso queremos tener la opción tanto de poder exportar y guardar datos en otros formatos, como poder gestionar las publicaciones de una (o varias) organizaciones, sin que se borren las bases de datos de estas.

Pese a ser un concepto que a priori puede parecer sencillo; debemos tratar y gestionar correctamente la persistencia de las bibliografías que gestionemos.

Teniendo en cuenta los casos de uso de la aplicación, debería estar habilitada en estos casos:

- Mediante exportación a CSV, Bibtex, XML, etc...:
 - **Registro temporal para búsquedas (registro_temp):** nos permite poder volcar datos a otro formato de nuestra elección y filtrado personalizado.

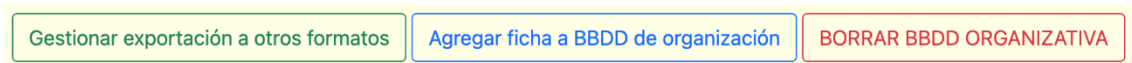


Será el escenario principal desde donde el usuario podrá realizar una persistencia totalmente acorde a sus preferencias, tanto de búsqueda como de formato elegido.

- **Registro de Organizaciones (Registro_ORG):** Para el apartado específico del caso de uso de organizaciones también es posible utilizar el método de exportación a diferentes formatos, exactamente de la misma manera
- **Registro temporal para búsquedas de Organizaciones (Registro_ORG_temp):** Igual que el anterior, pero incluyendo únicamente aquellos registros seleccionados mediante condiciones de búsqueda

FUSIÓN DE BASES DE DATOS BIBLIOGRÁFICAS

- Mediante persistencia en Base de Datos:
 - **Registro para Organizaciones (Registro_ORG):** cuando se inicia el software, todas las tablas de la BBDD se reinician mediante “drops”, no obstante, la tabla de las organizaciones queda persistente en el esquema relacional, siendo posible, no obstante, la realización de tareas de edición en ellas, y por supuesto el borrado total de la misma.



En el siguiente diagrama podemos observar la ampliación de las funcionalidades anteriormente descritas acerca de la persistencia.

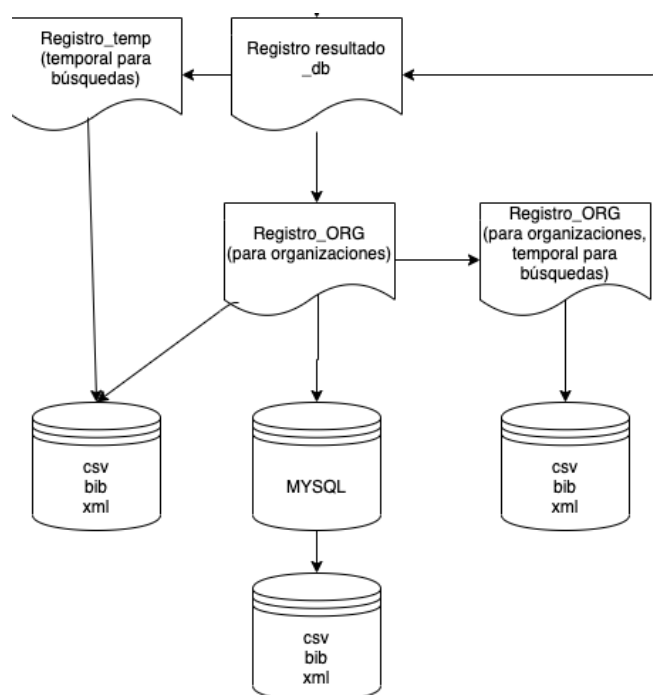


Figura: Diagrama de persistencia de la BBDD

Vista de la aplicación

Implementación

Pruebas

Instalación o despliegue

Uso y mantenimiento

FUSIÓN DE BASES DE DATOS BIBLIOGRÁFICAS

Pegados que me servirán

EndNote 20 - My EndNote Library.enl

File Edit References Groups Library Tools Window Help

Sync Configuration

All References 50

Imported References 50

Recently Added 50

Unfiled 50

Trash 0

MY GROUPS

My Groups

FIND FULL TEXT

GROUPS SHARED BY ...

ONLINE SEARCH

Library of Congress 0

LISTA (EBSCO) 0

PubMed (NLM) 0

Web of Science C... 0

more...

Imported References

Advanced search

Imported References

50 References

Author	Year	Title	Journal	Last Updated	Reference Type
AG Serrano, J.	2016	Visualización del significado de verbo...	Inteligencia...	30/08/2021	Journal Article
AG Serrano, J.	2016	Spanish Verbs Visualization: A study a...	Inteligencia...	30/08/2021	Journal Article
D Villarroel, ...	n/a	Assisting Quality Assessment (AQUA)...	n/a	30/08/2021	Journal Article
DW Oard, J ...	2004	Interactive Cross-Language Document...	Information...	30/08/2021	Journal Article
F López-Oste...	2001	Noun Phrase Translations for Cross-La...	Workshop o...	30/08/2021	Journal Article
F López-Oste...	2002	Interactive Cross-Language Searching...	Workshop o...	30/08/2021	Journal Article
F López-Oste...	2003	UNED at iCLEF 2003: Searching Cross...	Workshop o...	30/08/2021	Journal Article
F Lopez-Oste...	2003	I System Evaluation Experiments at CL...	Lecture Not...	30/08/2021	Journal Article
F López-Oste...	2004	Interactive Cross-Language Question ...	Workshop o...	30/08/2021	Journal Article
F López-Oste...	2005	Noun phrases as building blocks for cr...	Information...	30/08/2021	Journal Article
F Lopez-Oste...	2005	Part III-Interactive Cross-Language Inf...	Lecture Not...	30/08/2021	Journal Article
F López-Oste...	2005	UNED@ CL-SR CLEF 2005: Mixing diff...	Workshop o...	30/08/2021	Journal Article
F Lopez-Oste...	2006	Part VI-Cross-Language Speech Retri...	Lecture Not...	30/08/2021	Journal Article
F López-Oste...	2008	Interactive question answering: Is Cros...	Information...	30/08/2021	Journal Article

Attach file

UNED at iCLEF 2003: Searching Cross-Language Summaries

J. G. F López-Ostenero, F Verdejo

Workshop of the Cross-Language Evaluation Forum for European Languages, 450-461, 2003 2003 Vol. 7 Pages 450-461

DOI: 10.1016/S1569-2528(03)00033-3

Annotated Insert Copy

F López-Ostenero, J. G., F Verdejo (2003). "UNED at iCLEF 2003: Searching Cross-Language Summaries." *Workshop of the Cross-Language Evaluation Forum for European Languages*, 450-461, 2003. 7: 450-461.

Warning: file_get_contents(https://scholar.google.es/citations?view_op=search_authors&mauthors=Fernando+Ostenero&hl=es&oi=ao): failed to open stream: HTTP request failed! HTTP/1.0 429 Too Many Requests in /Applications/XAMPP/xamppfiles/htdocs/tfg/controllers/simple_html_dom.php on line 82

Conclusiones

Exposición breve sobre las hipótesis alcanzadas, opiniones, etc.

Bibliografía

La bibliografía debe incluir las referencias de las fuentes, en cualquier soporte, que se han consultado para realizar el TFG. El formato de las referencias es un estándar internacional, fijado por distintas normativas dependiendo del área de conocimiento. El tutor del TFG deberá indicar el formato más adecuado en cada caso. Recomendamos consultar la guía “Cómo citar bibliografía” para ver ejemplos y conocer los distintos formatos existentes:

http://portal.uned.es/pls/portal/docs/PAGE/UNED_MAIN/BIBLIOTECA/GUIAS/GUIABIBLIOGRAFIA.PDF

[Esta es la historia de las aplicaciones móviles | Skyscanner Espana](#)

[Criterios para elegir un stack tecnológico | by Daniel Estiven Rico Posada | Bancolombia Tech | Medium](#)

[Cómo elegir el stack de software adecuado para tu producto - Código Fuente \(codigofuente.org\)](#)

[Top 6 Tech Stacks That Reign Software Development in 2020 - Fingent Technology Stack Overflow Developer Survey 2021](#)

[Node JS vs PHP: Comparaciones, beneficios y tiempos de respuesta. \(luisjordan.net\)](#)

 [Framework vs Librería | EDteam](#)

[Ciclo de vida del software: todo lo que necesitas saber \(intelequia.com\)](#)

[Bib-it: Help - BibTeX Entry and Field Types \(sourceforge.net\)](#)

[\(Ciclo de vida de desarrollo de Software \(efectodigital.online\)\)](#)

Anexos

- En la medida de lo posible se intentará evitar el uso de Anexos, procurando incluir la totalidad de la información en la Memoria del TFG. No obstante, en función de la temática tratada, se precisará la inclusión de mapas, planos,

imágenes artísticas, etc., en cuyo caso podrán utilizarse cuantos anexos sean necesarios. Dichos anexos deberán estar adecuadamente referidos en el texto de la memoria. Asimismo, la lectura completa de la memoria debe permitir la comprensión completa del TFG desarrollado, quedando los anexos para detalles o cálculos poco significativos.

- Presupuesto: en el caso de que el alumno debe presentar como parte de su TFG un presupuesto o estudio económico de las actuaciones objeto del trabajo desarrollado.
- Mapas o planos: este documento es necesario cuando la representación gráfica de los contenidos del trabajo no puede reducirse únicamente a las figuras insertadas en el texto. Una buena práctica sería que cada uno de los planos utilizados estuviese referenciado en algún momento a lo largo del texto de la memoria o de los anejos.