

# Plataforma de colaboración para docentes

Por

**Carlos Ignacio Provoste Viveros**

**Patrocinante:** Julio Godoy del Campo

Memoria presentada

para la obtención del título

**Ingeniero Civil Informático**

Departamento de Ingeniería Civil Informática

y Ciencias de la Computación

De la

**Universidad de Concepción**



Concepción, Chile

Agosto 2019

<b>Resumen</b>	<b>3</b>
<b>1. Introducción</b>	<b>4</b>
1.1 Objetivo General	5
1.2 Objetivos Específicos	5
<b>2. Discusión Bibliográfica</b>	<b>6</b>
2.1 Procesamiento de Lenguaje Natural	6
2.1.1 Wordnet	7
2.1.2 NLTK	7
<b>3. Método Desarrollado</b>	<b>8</b>
3.1 Herramientas utilizadas	9
3.2 Base de Datos	11
3.3 Plataforma Web	13
3.3.1 Modelo	14
3.3.2 Vista	17
3.3.3 Controlador	22
3.4 Módulos para el procesamiento de textos y actualización de la base de datos	23
3.4.1 Normalización de textos	24
3.4.2 Procesamiento de textos	25
3.4.3 Comparación de textos	27
3.4.4 Comparación de intereses y expertises	28
<b>4. Experimentos y Resultados</b>	<b>29</b>
4.1 Similitud entre palabras	29
4.2 Bonificación por repetición de palabras	31
4.3 Comparación de textos	32
4.4 Puntaje	33
<b>5. Conclusiones</b>	<b>37</b>
5.1 Trabajo Futuro	38
<b>6. Bibliografía</b>	<b>39</b>

## Resumen

El procesamiento de lenguaje natural es un área de las ciencias de la computación e informática que permite estudiar la interacción entre el lenguaje y las computadoras. Esta área está muy utilizada hoy en día para tareas como la traducción automática y la corrección ortográfica. En esta memoria de título se diseñó un sistema informático que hace uso del procesamiento de lenguaje natural para hallar similitudes entre trabajos que han realizado los docentes o investigadores que hagan uso de ella, así como encontrar intereses o conocimientos en común que posean, además de una plataforma web que permita la fácil interacción docente - computadora para poder subir dichos trabajos y configurar intereses y conocimientos según se estime conveniente.

# 1. Introducción

Actualmente existen problemas que aquejan a la sociedad y que necesitan de una atención de manera urgente para evitar que estos puedan agravarse aún más. Algunos de estos más grandes problemas son la pobreza, desigualdad, cambio climático, acceso a la comida y agua, etc [1].

Para poder enfrentar estos problemas, se requiere una mirada multidisciplinaria y una cooperación de expertos de distintas áreas del conocimiento. En el ámbito académico, donde se encuentra mayoritariamente el capital humano avanzado del país, suele ser que los profesionales de un área tengan contacto solo con profesionales de su misma área. Por lo tanto, formar un grupo de trabajo multidisciplinario enfrenta un conjunto de desafíos.

Para poder conformar grupos de trabajo multidisciplinario, más allá de aplicar ciertas estrategias para mantener un ambiente laboral sano y eficiente, y también lograr mantener a un equipo unido y feliz. Estrategias como compartir durante los horarios de comida, relajarse en conjunto, etc. Ayudan a estos objetivos [2], pero también es necesario que exista una compatibilidad entre sus miembros, ya sea mediante intereses en común, que compartan conocimientos en una misma área o que hayan realizado trabajos investigativos similares.

Juntar a personas de diferentes disciplinas dependiendo de sus conocimientos e intereses se facilita cuando se sabe que es lo que le interesa o en que se ha perfeccionado cada experto, además de saber qué afinidad tienen 2 investigadores en base a los trabajos previos que han realizado, todo esto de una manera automática y sencilla, pero para esto se debe hacer un análisis de estos trabajos y compararlos con el resto de trabajos de todos los otros investigadores, sin duda un trabajo arduo. Para determinar esa similitud entre los trabajos se hace uso del procesamiento del lenguaje natural y, de esta forma, hallar similitudes entre ellos.

Esta memoria de título busca facilitar las tareas de comparación de intereses y expertises, además del análisis y comparación de trabajos realizados por los investigadores haciendo uso del procesamiento de lenguaje natural, el cual permitirá procesar dichos trabajos sacando las palabras relevantes de cada uno y de esta forma poder compararlo, no sin antes normalizar los textos. También se creó una plataforma web donde los docentes pueden subir sus trabajos y configurar sus intereses y expertises de manera sencilla y rápida, además de herramientas para la comunicación entre los usuarios de la plataforma y listas de recomendaciones que permiten encontrar quienes tienen más similitudes entre ellos.

## 1.1 Objetivo General

El objetivo de esta memoria es crear un sistema informático que fomente la colaboración en la Facultad de Ingeniería, permitiendo a investigadores (nuevos y ya establecidos) conectarse con investigadores de otras unidades académicas, con áreas de investigación, trabajos o problemas de interés en común.

## 1.2 Objetivos Específicos

- Crear sistema web que permita a investigadores ingresar datos tales como nombre, departamento al que pertenece, correo, áreas de investigación, trabajo realizado y problemas de interés.
- Crear una base de datos de áreas de investigación, trabajos hechos y problemas de interés de un número representativo de académicos de cada unidad de la Facultad de Ingeniería.
- Determinar similitud (mediante herramientas de procesamiento del lenguaje natural) entre las temáticas de investigación de los académicos que ingresen su información en pro del objetivo anterior, con el objetivo de establecer un primer contacto entre potenciales colaboradores.

## 2. Discusión Bibliográfica

En esta sección se presentarán algunas herramientas utilizadas o trabajos que influenciaron esta memoria. Estos trabajos hacen uso de las herramientas descritas en esta sección como Wordnet y NLTK y permiten dar un vistazo a estas funciones de Python y su uso en el procesamiento del lenguaje natural.

### 2.1 Procesamiento de Lenguaje Natural

El procesamiento del lenguaje natural (PLN o NLP, por sus siglas en inglés), es el campo de la informática, inteligencia artificial y lingüística que estudia las interacciones entre las computadoras y el lenguaje humano [7].

El PLN se encarga de la investigación de formas eficientes de establecer una comunicación entre personas y computadoras por medio del lenguaje natural. No trata de que la comunicación sea de forma abstracta, sino que se encarga de diseñar mecanismos para una comunicación eficiente computacionalmente que se puedan realizar por medio de programas que simulen la comunicación.

Actualmente, el procesamiento del lenguaje natural tiene muchos usos en el campo de las ciencias de la computación, tales como:

- Traducción automática de textos.
- Sistemas conversacionales.
- Detectar topics automáticamente, etc.

A pesar de su utilidad, hacer uso de herramientas para PLN conlleva algunas dificultades, como por ejemplo palabras cuyo significado cambia dependiendo del contexto en el que se encuentran. También existe la posibilidad de que una oración no tenga el significado que debiese, cosas como la ironía son importantes a la hora de interpretar una frase.

En definitiva, el PLN es parte fundamental de esta memoria en todo lo que conlleva la comparación de los trabajos realizados por los investigadores,, ya que permitirá analizar los texto que decidan subir a la plataforma y compararlos con los textos de otros investigadores y, de esta manera, encontrar similitudes entre ellos y así poder formar grupos de trabajos multidisciplinarios.

Existen herramientas en Python que son utilizadas en esta memoria y permiten realizar tareas de PLN, dichas herramientas son NLTK y Wordnet.

### 2.1.1 Wordnet

Wordnet es una base de datos léxica para el lenguaje inglés. Es usado en múltiples tareas de procesamiento de lenguaje natural tales como respuestas a preguntas, recuperación de información, etc. [3]

Wordnet consiste en 4 base de datos: 117097 sustantivos, 22141 adjetivos, 11488 verbos y 4601 adverbios. Las cantidades anteriores son de Diciembre de 2006 [4].

Esta herramienta fue utilizada en esta memoria para la comparación de textos, ya que permite la búsqueda de sinónimos, antónimos y lo más importante, hallar similitudes entre palabras (para más detalle, ver sección 3.2).

### 2.1.2 NLTK

Natural Language ToolKit es una herramienta para trabajo con lenguaje natural en formato texto en Python. Esta librería incluye una serie de herramientas que ayudan al procesamiento de lenguaje natural, tales como Lemmatization, Tokenization, etc. [5]

Es justamente por lo anterior que esta librería es utilizada en esta memoria, ya que permite el procesamiento de un texto y la obtención de sus palabras más relevantes dentro del mismo (Para más detalles, ver sección 3.2)

Un trabajo que influenció esta memoria fue “Natural Language Processing using NLTK and Wordnet”, ya que enseña el uso y la importancia de estas herramientas para una tarea tan importante como la interacción de las computadoras con el lenguaje humano. [6]. Otro trabajo que influyó fue “Text Analytic for Beginners using NLTK”, el cual complementa al trabajo anterior y hace uso de más herramientas para el procesamiento del lenguaje natural [7].

### 3. Método Desarrollado

El trabajo desarrollado en esta memoria se divide en 2 partes:

- Plataforma Web
- Módulos de procesamiento de lenguaje natural para manejo de base de datos y preprocesamiento de los textos.

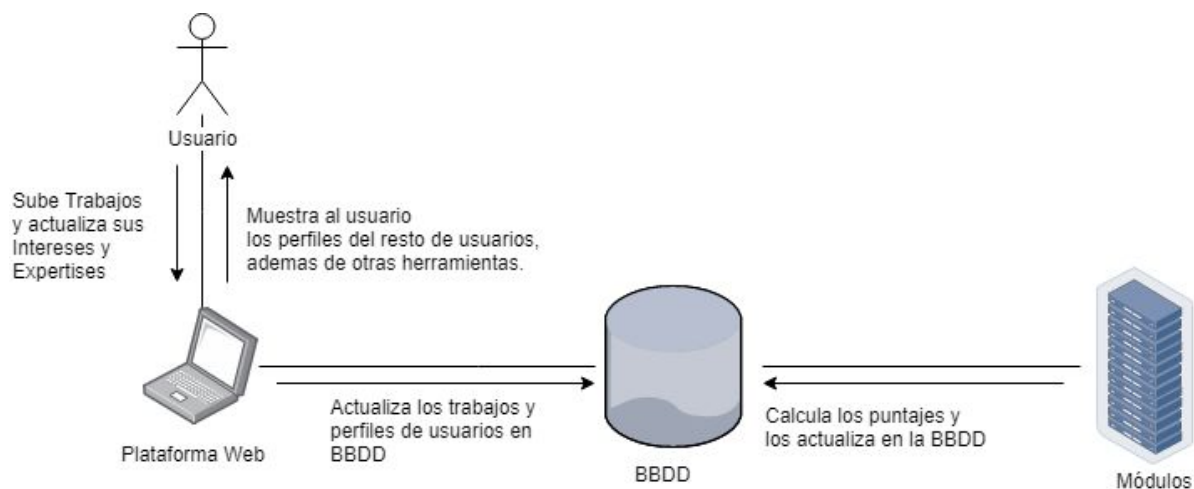


Figura 1: Diagrama que ejemplifica las relaciones entre los componentes del proyecto y el usuario

En la figura 1 se muestra un diagrama que ayuda a entender cómo interactúa el cliente con la plataforma, subiendo sus trabajos y configurando su perfil y obtiene de vuelta los resultados extraídos desde la base de datos. También muestra que los módulos y la plataforma no interactúan entre sí, sino que cada uno lo hace con la base de datos, actualizando lo que les corresponda y obteniendo los datos que necesitan.

La arquitectura elegida para el desarrollo de esta memoria fue la de Cliente - Servidor [8], ya que esta arquitectura permite que el usuario haga sus tareas y el servidor haga las suyas sin que uno interfiera con el trabajo del otro. Debido a que el servidor experimenta cargas importantes, sobre todo a la hora de procesar los trabajos de los investigadores, es de suma importancia que el usuario no se vea perjudicado por esta carga de trabajo y deba esperar grandes cantidades de tiempo para continuar usando la plataforma.



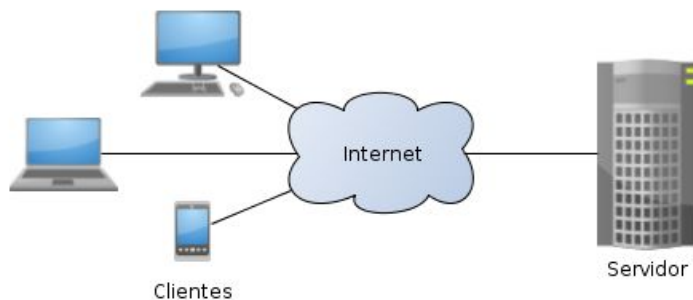


Figura 2: Diagrama de la arquitectura Cliente - Servidor [9]

Por otro lado, la plataforma Web fue realizada siguiendo el patrón de arquitectura Modelo - Vista - Controlador [10], ya que este posee ventajas que son muy útiles a la hora de programar herramientas como esta, donde los datos lógicos y manejo de información van separados de aquellos módulos que permiten la interacción del usuario con la plataforma.

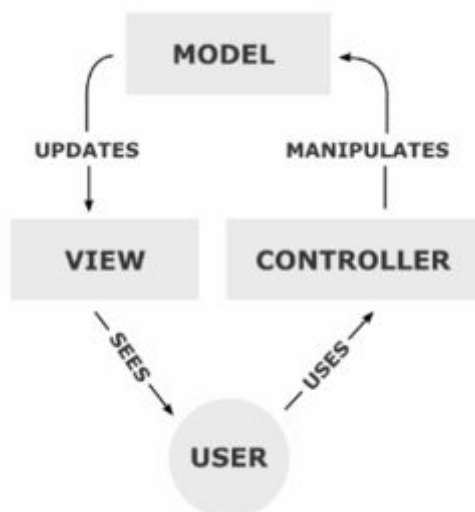


Figura 3: Esquema del Modelo - Vista - Controlador [11]

A continuación, se procederá a explicar en profundidad como actúan cada una de estas dos partes, así como las herramientas utilizadas durante el desarrollo:

### 3.1 Herramientas utilizadas

Durante el desarrollo de esta memoria, se utilizaron distintas herramientas de carácter informático:

- 1) Equipo: Todo el trabajo fue desarrollado en un computador ubicado en la sala de memoristas del Edificio de Ingeniería en Sistemas, 3° piso. Las características del equipo son:
  - a) Procesador: Intel Core i5 - 2300 CPU @ 2.80 GHz
  - b) Memoria RAM: 8 GB
  - c) HDD: 232 GB
  - d) Windows 10 Pro
- 2) Netbeans: Es un entorno de desarrollo libre, hecho principalmente para el lenguaje de programación Java [12]. Esta herramienta fue usada para la creación de la plataforma Web debido a ciertas ventajas que ofrece:
  - a) Es gratuito.
  - b) Existe una gran cantidad de módulos que pueden ser descargados y usados de manera libre en el proyecto.
  - c) Es de fácil uso y permite realizar trabajos con rapidez y sencillez.
  - d) Es multiplataforma.
- 3) Visual Studio Code: Es un editor de código fuente desarrollado por Microsoft y lanzado en el año 2015 [13]. Fue usado para el desarrollo de los módulos debido a que cumplía ciertos requisitos necesarios para este trabajo:
  - a) Compatible con Windows 10.
  - b) Gratuito.
  - c) Soporta una gran cantidad de lenguajes de programación, en especial Python, lenguaje elegido para el desarrollo de esta parte de la memoria.
- 4) MySQL: Es considerado el gestor de base de datos libre más popular del mundo, tiene una gran comunidad por detrás, lo que facilita la labor de buscar documentación para el uso de esta herramienta [14]. Otras características que posee y que hicieron posible su elección son:
  - a) Gratuito.
  - b) Multiplataforma.
  - c) Es de fácil instalación a la hora de montar un servidor local para su uso.
- 5) NLTK: Posee muchas herramientas para procesamiento de lenguaje natural en Python, lo que permite realizar todas las tareas necesarias para el procesamiento de los textos. Su elección fue debido a la gran cantidad de herramientas que ofrece y la documentación que posee, la cual es clara y concisa.

- 6) Wordnet: Ofrece funciones claves en esta memoria, como la posibilidad de comparar 2 palabras mediante funciones de similaridad, como la usada en este trabajo, la cual es `wup_similarity`. Acá la elección fue debido a que es una librería que se encuentra dentro de NLTK.

En resumen, la herramientas elegidas son todas gratuitas, salvo el equipo que fue facilitado por el departamento de Ingeniería Civil Informática, ya que esta memoria no cuenta con recursos para su realización, además de ser básicamente editores de texto, gestores de base de datos y librerías de Python debido a que está todo el código desarrollado desde 0 haciendo uso de estas herramientas que ofrece este lenguaje.

## 3.2 Base de Datos

Como se mencionó anteriormente, se ha hecho uso del gestor de base de datos MySQL por motivos ya mencionados.

La construcción de esta base de datos se basa en el modelo Entidad - Relación [15], donde tienen a las entidades necesarias con las relaciones necesarias para lograr un óptimo funcionamiento.

- Lista de entidades:
  - Docente: Posee atributos tales como el nombre del docentes, una descripción, una foto para colocar en su perfil y su respectivo usuario y contraseña para poder acceder a la plataforma.
  - Trabajo: Tabla que guarda los trabajos subidos en la plataforma, junto a un nombre elegido por quien lo sube y además la ID de quien subió dicho trabajo.
  - Interés: Tabla que guarda una lista de intereses que podrán ser elegidos por un docente para guardar en su perfil.
  - Expertise: Tabla que guarda una lista de expertises que el docente podrá elegir para mostrar que conocimientos posee.
- Lista de relaciones:
  - *DocenteTieneTrabajo*: Relación que une a los docentes con los respectivos trabajos que ha subido a la plataforma.
  - *DocenteTieneInteres*: Relación que une a los docentes con los respectivos intereses que ha elegido para mostrar en su perfil.
  - *DocenteTieneExpertise*: Relación que une a los docentes con los respectivos conocimientos que posee y ha decidido mostrar en su perfil.

- *TrabajosSimilares*: Relación que une a 2 docentes junto a un puntaje que dependerá de la similitud de los trabajos que han realizado.
- *InteresesSimilares*: Relación que une a 2 docentes junto a un puntaje asignado dependiendo de la similitud que tengan los intereses que han decidido guardar en la relación “DocenteTieneInteres”.
- *ExpertisesSimilares*: Relación que une a 2 docentes junto a un puntaje asignado dependiendo de la similitud que tengan las expertises que han decidido guardar en la relación “DocenteTieneExpertise”.

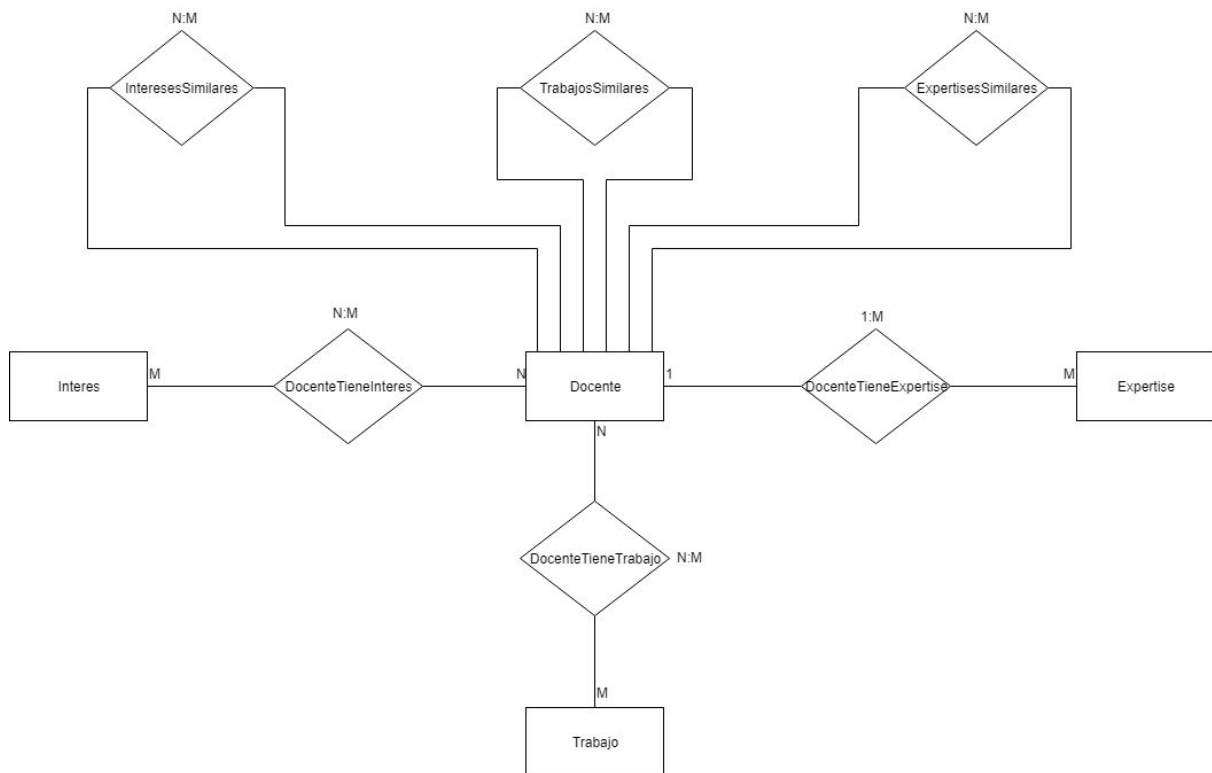


Figura 4: Modelo Entidad - Relación de la base de datos creada para el proyecto.

Como se puede ver en la figura 4, y a modo de aclaración, las relaciones “*InteresesSimilares*”, “*TrabajosSimilares*” y “*ExpertisesSimilares*” son relaciones reflexivas de la entidad “*Docente*”, ya que relacionan a 2 docentes con los respectivos puntajes que correspondan.

### 3.3 Plataforma Web

La decisión de desarrollar una plataforma web viene por el hecho de que los investigadores, en su mayoría, cuentan con equipos de escritorio en su oficina, además de que el uso de este tipo de plataforma es mucho más sencillo en un equipo de escritorio que en un celular, por ejemplo, a la hora de subir un trabajo, es mucho más común que un investigador lo tenga guardado en su página web, pendrive o equipo, no en su celular, por lo que, en caso de tener que descargarlo, es más fácil hacerlo desde su computador, lo mismo ocurre cuando quiera subirlo a la plataforma, se le hace la tarea más sencilla desde su equipo que desde su celular.

A continuación, se explicará de manera detallada el proceso seguido para desarrollar esta plataforma:

Para empezar, se ha optado por desarrollar esta parte de la memoria con el lenguaje de programación Java, los motivos de esta elección son el hecho de que Java es uno de los lenguajes de desarrollo más populares del mundo, por lo tanto posee una comunidad gigante que está constantemente desarrollando herramientas libres para ser usadas. Java es también un lenguaje orientado a objetos que da las herramientas necesarias para la construcción de plataformas siguiendo el patrón de arquitectura Modelo-Vista-Controlador de una manera eficiente y sencilla. Por último, Java es totalmente compatible con el gestor de bases de datos MySQL, teniendo librerías exclusivas para este gestor y funciones altamente eficientes para hacer consultas a las bases de datos correspondientes.

Siguiendo el patrón Modelo - Vista - Controlador, se dividió esta parte del proyecto en 3 partes:

- Clases para las entidades de la base de datos y funciones para extraer dichos datos (Modelo)
- Interfaz de Usuario (Vista)
- Servlets (Controlador)

Para una explicación más sencilla de entender, se explicará cada parte por separado:

En las subsecciones 3.3.1, 3.3.2 y 3.3.3 se explican cómo se adapta el patrón de diseño Modelo-Vista-Controlador al proyecto y se detallan cada uno de ellos.

### 3.3.1 Modelo

Se crearon 10 clases que representan únicamente las entidades en la base de datos. Básicamente cada una de estas 10 clases posee sus atributos, constructor, getters y setters.

Por ejemplo, en la base de datos se tiene la entidad docente, cuyos atributos son:

- Nombre
- Apellido
- Foto
- Usuario
- Contraseña
- Descripción

Lo que significa, que a la hora de crear la clase “Docente” en el proyecto, crearemos también los atributos:

- Nombre → String
- Apellidos → String
- Usuario → String
- Contraseña → String
- Descripción → String
- Foto → Byte[]

```
public class Docente {  
  
    private String docente;  
    private String pass;  
    private String nombre;  
    private String apellido;  
    private String descripcion;  
    private byte[] foto;  
}
```

Figura 5: Clase Docente con sus atributos creados

```
public Docente(String docente, String pass, String nombre, String apellido, String descripcion, byte[] foto) {  
    this.docente = docente;  
    this.pass = pass;  
    this.nombre = nombre;  
    this.apellido = apellido;  
    this.descripcion = descripcion;  
    this.foto = foto;  
}
```

Figura 6: Constructor de la clase “Docente”

Cada una de estas clases posee sus respectivos getters y setters, los cuales son métodos o funciones que permiten acceder a dichos atributos para conocer su valor o cambiarlos dependiendo de la situación.

Lo explicado anteriormente se realiza para cada una de las 10 clases necesarias para el proyecto, las cuales son:

- Docente
- Interes
- Expertise
- Documento
- DocenteTieneInteres
- DocenteTieneExpertise
- TrabajosSimilares
- InteresesSimilares
- ExpertisesSimilares
- Mensaje

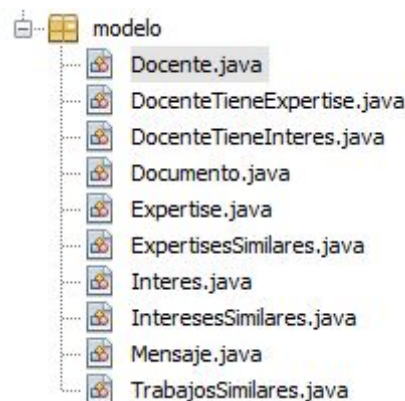


Figura 7: Carpeta contenedora de las 10 clases del Modelo

También existen 10 clases, que llamaremos clases “hermanas” a las 10 ya mencionadas.

La función de esas 10 clases hermanas, es hacer uso de las clases mencionadas anteriormente y proporcionar métodos que permitan la manipulación de la base de datos. Por ejemplo, se tiene la clase “Docente”, dicha clase tiene a su hermana “DocenteDB” (Cada clase hermana tiene el apellido DB como parte de su nombre). DocenteDB tiene métodos que controlan la tabla “Docente” que existe en la base de datos, por ejemplo, si se necesita obtener el perfil de un docente, la clase DocenteDB tiene un método que realiza dicha acción, llamado “getPerfil”, el cual sigue los siguientes pasos:

- Realiza la consulta en la base de datos para obtener el perfil solicitado.
- Recibe los atributos solicitados y los guarda en variables.
- Crea un objeto de clase “Docente” y asigna a cada atributo de dicha clase el valor correspondiente de acuerdo a lo obtenido desde la base de datos.
- Una vez asignados los valores, retorna dicho objeto de clase Docente.

Lo anterior es una de las 10 clases hermanas existentes, las 9 restantes con sus respectivos métodos son:

- 1) InteresDB:
  - a) `getIntereses`: Realiza una consulta a la base de datos que obtiene todos los intereses registrados. Retorna una lista con los datos solicitados.
  - b) `agregarIntereses`: Inserta en la base de datos un interés que algún docente quiera registrar, pero que no se encuentra en la base de datos.
- 2) ExpertiseDB:
  - a) `getExpertises`: Realiza una consulta a la base de datos que obtiene todas las expertises registradas. Retorna una lista con los datos solicitados.
  - b) `agregarExpertises`: Inserta en la base de datos una expertise que algún docente quiera registrar, pero que no se encuentra en la base de datos.
- 3) DocumentoDB:
  - a) `subirDocumento`: Método que permite agregar un documento a la base de datos cuando el docente lo requiera.
  - b) `listaDocumentosPorIdDocente`: Este método retorna una lista con todos los documentos que estén ligados a un docente de acuerdo a su ID, se usa para crear una tabla que muestre los trabajos desarrollados por el docente y mostrarlos en su perfil.
  - c) `mostrarPDF`: Este método es utilizado para obtener el conjunto de bytes que conforman el trabajo y, de esta manera, mostrar dicho PDF en el navegador.
- 4) DocenteTieneInteresDB:
  - a) `getInteresesPorId`: Método que retorna una lista de todos los intereses que tiene registrados un docente, los intereses retornados depende de la ID del docente que solicita esta lista.
  - b) `insertarIntereses`: A diferencia del método “`agregarIntereses`” de la clase `Interes`, este método inserta en la tabla “`DocenteTieneInteres`” tanto al interés del docente, como la ID de este.
- 5) DocenteTieneExpertiseDB:
  - a) `getExpertisesPorId`: Método que retorna una lista de expertises que están relacionadas con el docente cuya ID haya sido ingresada a la hora de ejecutar esta función.
  - b) `insertarExpertise`: Método encargado de insertar en la base de datos una expertise junto a la ID del docente que posea dicha expertise.
- 6) InteresesSimilares:
  - a) `getInteresesSimilares`: Método que recibe la ID de un docente y retorna una lista conformada por todos los otros docentes que posean intereses similares con quien esté solicitando, dicha lista está ordenada de forma descendente de acuerdo al puntaje.
- 7) ExpertisesSimilares:
  - a) `getExpertisesSimilares`: A partir de la ID de un docente, se retorna una lista ordenada descendientemente de acuerdo a un puntaje, con todos los otros docentes, de acuerdo a la similitud de expertises que tengan con el docente solicitante.
- 8) TrabajosSimilares:



- a) `getTrabajosSimilares`: Este método retorna una lista ordenada descendientemente, constituida por los docentes cuyos trabajos tienen similitudes con los trabajos realizados por el docente que solicita dicha lista.
- 9) `MensajeDB`:
- a) `getMessage`: Retorna un mensaje, se usa la ID de este para realizar la búsqueda en la base de datos.
  - b) `getTodosMensajesReceptor`: Retorna una lista con todos los mensajes que ha recibido un docente.
  - c) `cambiarEstadoAMensajeLeido`: Método que cambia el estado de un mensaje de “No leído” a “Leído”.

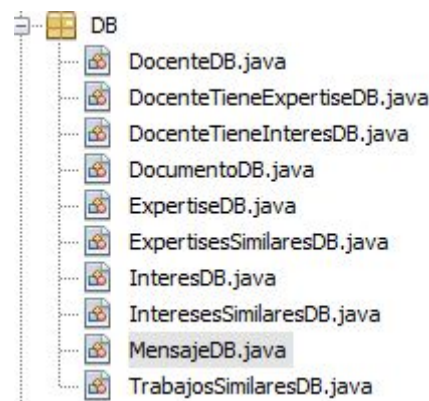


Figura 8: Carpeta que contiene a las clases hermanas

### 3.3.2 Vista

La vista del proyecto consisten en una serie de archivos JSP (Java Server Pages), programados en lenguaje HTML y Java.

Existen 9 vistas en el proyecto, cada una cumple un objetivo específico que se explicará a continuación:

1. Index: Vista para el Login del docente en la aplicación

## Iniciar Sesión

Usuario

Contraseña

Login

Figura 9: Interfaz para la vista de Login

2. Inicio: Vista que muestra el perfil del docente que ha accedido a la plataforma, acá puede subir trabajos y manejar sus respectivos conocimientos e intereses. Además, en el lado derecho aparecen 3 tablas que contienen (Hasta un máximo de 3) docentes con similitudes en alguno de los 3 apartados ya mencionados.

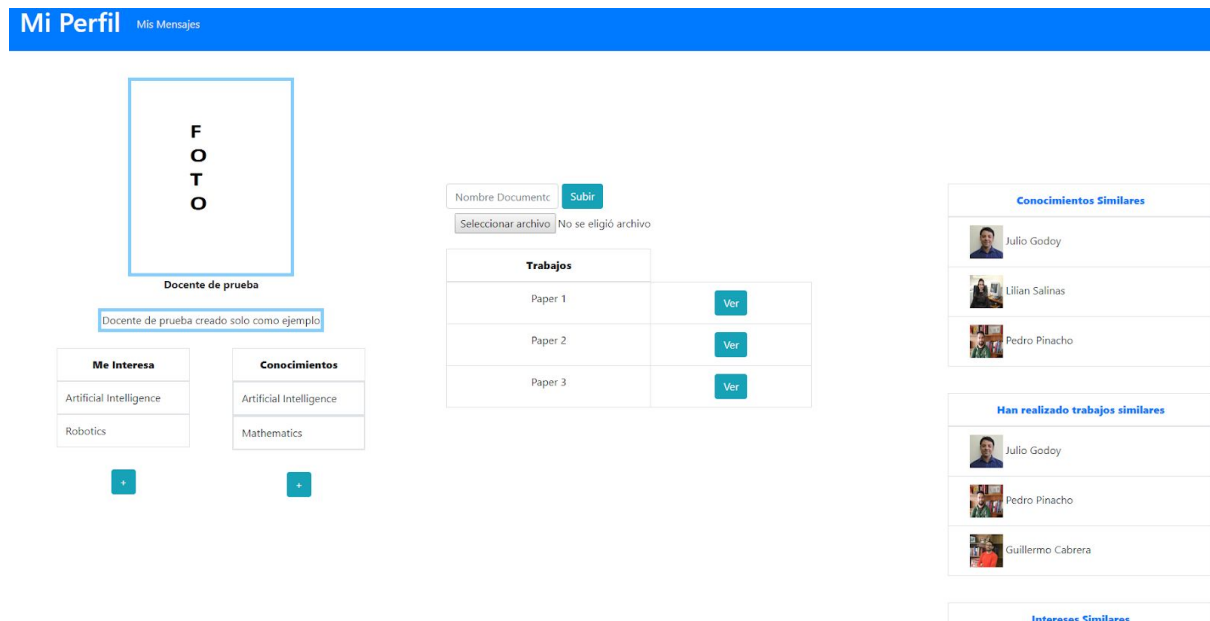


Figura 10: Interfaz de la vista para el perfil del Docente

3. VerPerfil: JSP usada para ver los perfiles de otros docentes. Es muy similar a la JSP inicio, con la diferencia de que acá no se puede modificar nada, así como tampoco aparecen las tablas de la derecha con las recomendaciones de docentes.

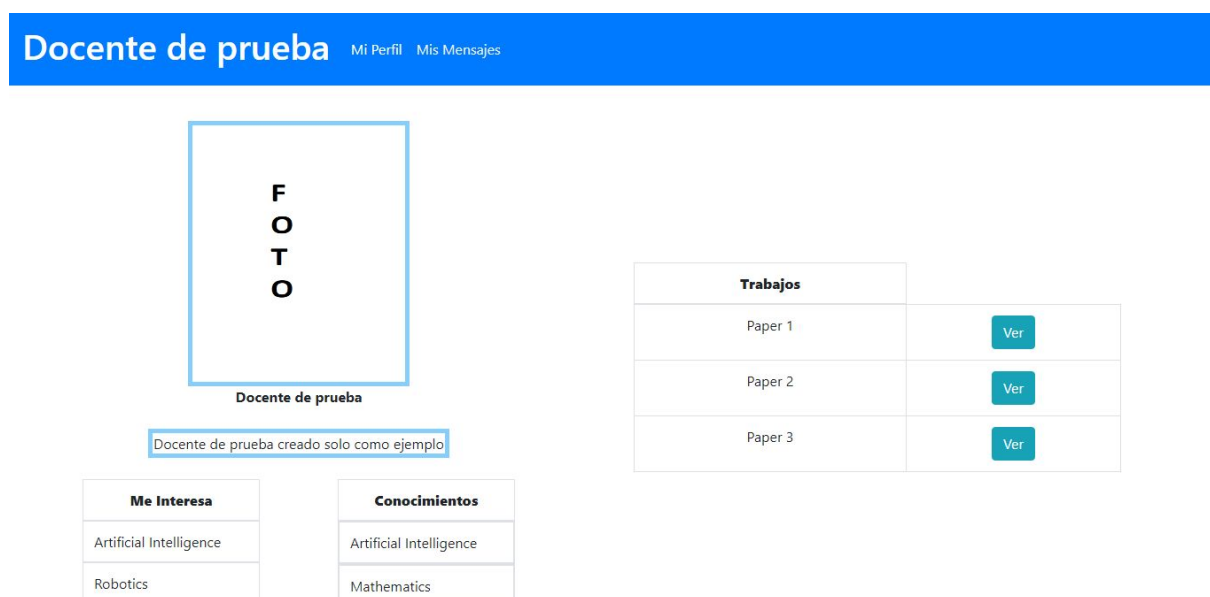


Figura 11: Interfaz a la hora de acceder a un perfil de otro docente

4. **TodoExpertisesSimilares**, **TodoTrabajosSimilares** y **TodoInteresesSimilares**: Son 3 vistas distintas, pero ambas cumplen funciones similares, expandir las listas de similitudes de las tablas que aparecen en el perfil del docente, pero ahora sin la restricción de un máximo de 3, acá se despliegan todos los otros docentes, ordenados de forma descendente de acuerdo a un puntaje.







<b>Han realizado trabajos similares</b>		
	<b>Julio Godoy</b>	<a href="#">Enviar Mensaje</a>
	<b>Pedro Pinacho</b>	<a href="#">Enviar Mensaje</a>
	<b>Guillermo Cabrera</b>	<a href="#">Enviar Mensaje</a>
	<b>Medi cina</b>	<a href="#">Enviar Mensaje</a>
	<b>Lilian Salinas</b>	<a href="#">Enviar Mensaje</a>
	<b>IA IA</b>	<a href="#">Enviar Mensaje</a>

Figura 12: Ejemplo de una lista para la tabla “Trabajos Similares”

5. Mis Mensajes: Vista que despliega una lista con los mensajes recibidos por el docente que hace uso de la plataforma.

#	De:	Asunto:	Recibido:
1	Julio Godoy	Hola Pedro	2019-06-18
2	Julio Godoy	Hola	2019-06-21

Figura 13: Interfaz que muestra la lista de mensajes recibidos

6. Enviar Mensaje: Interfaz desplegada que permite escribir un mensaje a otro docente y enviarlo de manera privada.

**Destinatario: Docente de prueba**

Asunto

Mensaje

Figura 14: Interfaz para enviar mensajes

7. Leer Mensaje: Vista mostrada cuando se hace click en un mensaje para leerlo.

Mensaje de prueba		
F O T O	De	Docente de prueba
	Destinatario	Julio Godoy
	Fecha	26/06/2019

Este es un mensaje de prueba creado solo para la demostración de la interfaz.

Figura 15: Interfaz para la lectura de los mensajes recibidos

### 3.3.3 Controlador

En esta memoria, la parte controladora del proyecto está incluida en los mismos archivos JSP. Se han incluido secciones de código Java cuya función es conectar la vista con el modelo y poder controlar la plataforma de forma correcta.

Otra parte importante del controlador, son los Servlets. Un servlet es un archivo Java que recibe parámetros de una lista a través de distintas funciones de comunicación y los manipula para realizar una acción u otra.

En concreto, existen 10 Servlet en el proyecto, cuyo nombre y función se explicarán a continuación:

- 1) **AgregarExpertise:** Recibe una expertise que haya sido agregada por un docente para su perfil y la agrega a la base de datos para formar la relación Docente - Expertise.
- 2) **AgregarIntereses:** Recibe un interés que haya sido agregado por un docente para su perfil y lo agrega a la base de datos para formar la relación Docente - Interés.
- 3) **AgregarNuevaExpertise:** Se activa en caso de que un docente agregue una expertise que no se encuentra registrada en la base de datos, toma como parámetro el nombre de la expertise y la agrega para que en un futuro, sea desplegada en la lista de expertises disponibles para colocar en un perfil de docente.
- 4) **AgregarNuevoInteres:** Mismo funcionamiento que el servlet anterior, pero, en este caso, realiza la tarea para los intereses.
- 5) **Conexion:** Posee todos los parámetros y funciones para conectar la plataforma con la base de datos.
- 6) **EnviarMensaje:** Recibe todo lo necesario para el envío de un mensaje privado, manipula tanto el texto, asunto, fecha de envío, remitente y receptor.

- 7) Login: Recibe el usuario y contraseña ingresados para poder acceder a la plataforma, en caso de que sean correctos, acepta el login y crea una sesión para el usuario, si son incorrectos rechaza la solicitud y no se puede ingresar.
- 8) MostrarFoto: Obtiene una imagen de la base de datos y la muestra en la vista que sea necesaria.
- 9) MostrarPDF: Permite poder visualizar los archivos PDF en el navegador.
- 10) SubirPDF: Encargado de recibir un archivo PDF y subirlo a la base de datos.

### 3.4 Módulos para el procesamiento de textos y actualización de la base de datos.

En esta memoria se hacen uso de varios módulos cuyas funciones son:

- Normalización de textos
- Procesamiento de textos.
- Comparación de intereses y expertises entre los docentes registrados.
- Comparación de textos.
- Actualización de la base de datos.

Estos módulos están desarrollados en el lenguaje de programación Python, por los siguientes motivos:

- Python es un lenguaje de programación muy popular actualmente [16].
- Posee una comunidad gigante que está constantemente desarrollando herramientas y librerías para este lenguaje.
- Posee librerías extremadamente útiles para este proyecto, librerías como Wordnet y PyPDF2 permiten leer y procesar textos con facilidad.
- El acceso a archivos y directorios está muy simplificado en este lenguaje, facilitando la tarea de manipular los textos y asociarlos al docente que corresponda.

Se consideró la posibilidad de ejecutar ciertos módulos cuando se suba un documento, pero se descartó al simular varios documentos subidos al mismo tiempo. Lo anterior provoca una sobrecarga en la memoria que trae como consecuencia el congelamiento del equipo debido al alto consumo de recursos.

Finalmente, la ejecución de los módulos se programa para una hora con bajo uso de la plataforma, en ese momento, se realizan los pre-procesamientos y comparaciones de todos los textos subidos durante el día, se realizan comparaciones de intereses y expertises

y se actualiza la base de datos, permitiendo ver los cambios reflejados en la plataforma una vez que este proceso termine.

A continuación se presentará con detalle el funcionamiento de cada uno de estos procesos realizados:

### 3.4.1 Normalización de textos

Antes de hacer uso de técnicas de procesamiento de lenguaje natural, es importante realizar ciertas acciones que serán indicadas a continuación.

Normalizar un texto es la acción de unificar todas las palabras que tienen un mismo significado, por ejemplo, si en un texto existen las palabras “ventana”, “Ventana” y “Ventanas”, todas ellas se unifican bajo una misma palabra → ventana.

En esta memoria se realizan varios pasos a la hora de normalizar el texto, pero es importante señalar que todo el proceso siguiente solo sirve para texto en inglés debido a limitaciones de las librerías, que hacen uso de diccionarios en esa lengua. Los pasos a seguir son:

- 1) Tokenizar: Esta acción permite separar todas las palabras del texto, crea una lista de palabras mediante la detección de espacios.
- 2) Remove caracteres No-Ascii: Como segundo paso, se elimina cualquier carácter que aparezca en la lista de palabras y no sea uno de los 255 caracteres que pertenecen a la tabla ascii.
- 3) Minúsculas: Luego de quitar los caracteres No-Ascii, es necesario eliminar toda mayúscula que aparezca en el texto, esto permitirá detectar la raíz a la que pertenece cada palabra.
- 4) Eliminar números: Si bien a la hora de normalizar los textos se debe transformar los números a palabras, por ejemplo, el número 4 se debe cambiar a “four”, en este caso se optó por eliminarlos, debido a que el método de comparación usado relaciona de manera directa cada número, haciendo que las comparaciones no sean realistas o correctas, además los se detectó que los números no son importantes a la hora de elegir las palabras que representan un texto.
- 5) Remove Stopwords: Las stopwords (O palabras vacías en español) son palabras que carecen de un significado por sí solas, en los pronombres,

conjunciones, etc. Son palabras vacías que pueden ser eliminadas en estos casos. Por ejemplo, en la frase “Caminar al parque”, en las palabras “Caminar” y “Parque” podemos inferir algo, por el contrario “Al” carece de significado y por separado no aporta nada. En este paso, se detectan todas aquellas palabras que no aportan al texto cuando están separadas del resto y se eliminan para aliviar la tarea de comparación.

- 6) Lemmatize: Este es un proceso algorítmico muy poderoso, consiste en transformar las palabras a su forma base, es muy útil sobre todo en verbos, por ejemplo, en inglés tenemos el verbo “cries”, al aplicar *lemmatize* vamos a obtener esa palabra en su forma base “cry”. Esto ayuda a evitar la ambigüedad en un texto y obtener todas las palabras en su forma base.

Los 6 pasos anteriores es todo lo que se realiza a los textos antes de su procesamiento, teniendo como resultado una lista de palabras, las cuales están todas en su forma base, quitan puntuaciones innecesarias y eliminando números. Todo lo anterior nos permite realizar un procesamiento del texto, ya que es necesario, pero también facilita la tarea eliminando palabras y con ello, una carga innecesarias al servidor.

Para finalizar esta subsección, y a modo de observación, los 6 pasos mencionados anteriormente se unifican bajo la palabra “Normalización” o “Normalizar”.

### 3.4.2 Procesamiento de textos mediante PLN

Luego de tener el texto normalizado, se debe procesar para así obtener las palabras más relevantes de cada trabajo y poder compararlos con los documentos de los otros docentes.

El proceso se divide en 2 partes, primero se debe obtener una idea sobre qué temas abarca el texto y luego, a partir de esas palabras más relevantes, analizar el resto de palabras y procesar el documento en su totalidad.

- 1) Extracción de palabras relevantes: Esto se logra analizando la primera página de cada documento, esto es porque en esa primera página se encuentra el abstract y la introducción, lo que permite tener una noción del texto completo analizando solo el inicio de este. Para ello se debe extraer todo el contenido de esta primera página, normalizar cada palabra y guardarlas en una lista de palabras relevantes. Una vez finalizado este proceso, podemos pasar al siguiente paso.



- 2) Análisis texto completo: Luego de extraer las palabras relevantes de la primera página, debemos extraer el contenido del resto del texto y normalizarlo, una vez hecho eso, se debe hacer una comparación entre las palabras relevantes obtenidas en el primer paso y compararlas con las palabras extraídas del resto del texto, aquellas palabras que cumplan con ciertas condiciones pasan a la lista de palabras finales de cada texto.

Ahora que el proceso está explicado, se detallarán ciertas condiciones y métodos utilizados para lograr esta tarea tales como la función **wup\_similarity**.

**Wup\_similarity** es una función que pertenece a la librería *wordnet* de Python. Su nombre es una simplificación de Wu-Palmer Similarity [17] y su tarea es la de retornar un puntaje basado en que tan parecido sean 2 palabras dependiendo de la distancia que exista entre ellas.

El puntaje basado en la distancia se calcula de la siguiente manera:

$$\text{SimWP} = 2 * N / (N1 + N2)$$

Para saber qué son esas variables, ver figura 16.

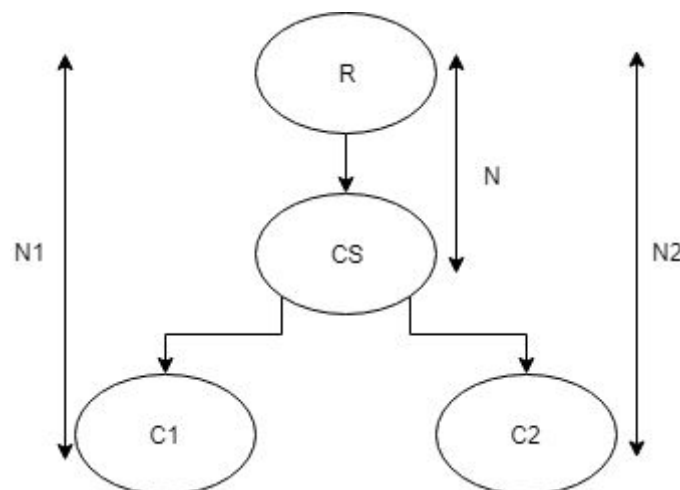


Figura 16: Ejemplo para entender cálculo de similitud entre 2 palabras

R → Nodo raíz.

CS → Closest Common Ancestor: Ancestro común más cercano entre ambas palabras

C1 y C2 → Palabras a las que se le calcula la similitud

N1 y N2 → Distancia entre las palabras respectivas y el nodo raíz

N → Distancia entre el nodo raíz y CS

A partir de este cálculo, wup\_similarity entrega un puntaje que va desde 0 hasta 1.

Gracias a esta función, es posible calcular la similitud que existe entre 2 palabras y, de esta forma, hacer una selección de aquellas palabras más importantes que existen en un texto. Para lograr esto, comparamos las palabras seleccionadas de la primera página con cada una de las seleccionadas en el resto del texto. Si existe una palabra de la primera lista que, al ser comparada con otra de la segunda lista y obtenga un puntaje superior a 0.8, pasa a formar parte de las palabras finales de ese texto.

### Algoritmo 1: CompararPalabras

**Input:** PalabrasPagina1, PalabrasRestoDelTexto

**Output:** PalabrasFinales

```
for palabra1 in PalabrasPagina1 do:
    for palabra2 in PalabrasRestoDelTexto do:
        if palabra1.wup_similarity(palabra2) > 0.8 then:
            PalabrasFinales.add(palabra2)
return PalabrasFinales
```

Ya terminado este proceso y obtenidas las palabras finales del texto, estas se guardan en un archivo de texto con un nombre igual a la ID en la base de datos del documento procesado. Esto se hace para ahorrar tiempo y no tener que procesar cada texto cuando se quiera hacer una comparación.







 1.txt	01-07-2019 11:18	Documento de tex...	4 KB
 4.txt	01-07-2019 11:19	Documento de tex...	4 KB
 5.txt	01-07-2019 11:19	Documento de tex...	2 KB
 6.txt	01-07-2019 11:20	Documento de tex...	3 KB
 7.txt	01-07-2019 11:20	Documento de tex...	1 KB
 8.txt	01-07-2019 11:22	Documento de tex...	5 KB

Figura 17: Carpeta de un docente con todos sus textos ya procesados

### 3.4.3 Comparación de textos

Ya teniendo todos los textos procesados, se debe hacer una comparación entre aquellos texto de un docente, con todos los otros de cada docente restante, así se consigue un puntaje de acuerdo a la similitud que existan entre estos trabajos y poder identificar quienes han realizado investigaciones similares. El proceso es similar al paso anterior, se hace uso de la función `wup_similarity` para cálculo de puntaje.

Primero se debe identificar si es que existe un docente nuevo en la base de datos, en caso de que exista uno, se debe crear su carpeta y procesar todos sus textos, una vez hecho, se debe comparar cada uno de sus textos con el resto de texto de los otros docentes y calcular su puntaje basado en estas comparaciones.

En caso de que no exista un docente nuevo, se verifica si alguno de los ya existentes ha subido algún documento en el último día, en caso de ser así, se debe procesar ese texto nuevo y compararlos con el resto de profesores, solo se hace con los textos nuevos, no es necesario volver a comparar todos los textos nuevamente en estos casos.

Si no hay docentes ni trabajos nuevos, la base de datos no se toca.

El proceso de comparación entre textos es el siguiente, se compara cada palabra procesada de un texto con todas las palabras procesadas del otro texto, se calcula el puntaje entre ellas y al finalizar, se suma al puntaje acumulado el mayor puntaje que se haya obtenido entre una palabra y todas las otras del otro documento. Al finalizar se le suma un bono que se calcula a partir de la cantidad de veces que se repite una palabra con la otra, al sumarle el bono, el puntaje obtenido en esa comparación entre 2 documentos se suma al puntaje acumulado que poseen esos 2 docentes, así se evita tener que calcular todos los textos nuevamente cada vez que se suba uno nuevo.

### **Algoritmo 2:** CompararTextos

**Input:** Documento1, Documento2

**Output:** Puntaje

```
puntaje ← 0
comparaciones ← 0
for palabra1 in Documento1 do:
    similitudMayor ← 0
    frecuencia ← 0
    for palabra2 in Documento2 do:
        if palabra1.wup_similarity(palabra2) > mayor then:
            mayor ← palabra1.wup_similarity(palabra2)
            frecuencia ← repeticionesPalabra1 + repeticionesPalabra2
        bono ← 1 + ((frecuencia * frecuencia) / (len(Documento1)*len(Documento2)))
        puntaje += (mayor * frecuencia) * bono
return puntaje
```

Al finalizar todas las comparaciones, se actualiza la base de datos con los puntajes correspondientes para cada par de docentes, de esta manera se puede ver quienes han realizado trabajos similares, ya que aquellos que puedan trabajar juntos obtendrán más puntaje que aquellos que no puedan.

## 3.4.4 Comparación de intereses y expertises

La función de este módulo es, como el nombre de la sección lo indica, comparar todos los intereses y expertises de los docentes registrados, accediendo a sus perfiles, tomando cada interés y expertises que el docente haya colocado y comparándolos con el resto.

### **Algoritmo 1:** CompararIntereses

**Input:** InteresesDocente1, InteresesDocente2

**Output:** Puntaje

```
contador → 0
for interesDocente1 in InteresesDocente1 do:
    for interesDocente2 in InteresesDocente2 do:
        if interesDocente1 == interesDocente2 then:
            contador += 1
        else:
            contador += similaridad entre interesDocente1 e interesDocente2
return contador
```

Del algoritmo anterior obtenemos un puntaje de acuerdo a la similitud que existe entre los intereses de ambos docentes, ese puntaje permitirá actualizar la base de datos y ordenarlos de manera descendente.

Para el caso de las expertises, su funcionamiento es exactamente igual, con la diferencia de que ahora se accede a la lista de expertises en vez de los intereses, es por ello que se omiten explicaciones.

Ese es todo el trabajo que realizan los módulos de Python creados para esta memoria y la explicación del desarrollo en general.

## 4. Experimentos y resultados

Durante el desarrollo de esta memoria, se realizaron varias pruebas para mejorar la plataforma. Dichas pruebas consideraban aspectos de eficiencia como el tiempo que tardaba en realizar comparaciones de textos y la calidad de dichas comparaciones.

A continuación se expondrán las pruebas más importantes que se realizaron a la plataforma y los resultados obtenidos.

### 4.1 Similitud entre palabras

Dentro del proceso de comparación de textos existe una parte en la cual se comparan las palabras relevantes de cada uno, para ellos se utiliza una función llamada “wup\_similarity” (Ver sección 3.4.2) la cual toma un par de estas palabras y entrega un número entre 0 y 1 que representa qué tan cercanas están en la ontología que conforma Wordnet.

En esta subsección se realizan experimentos de comparación de palabras y se mostrarán los resultados para verificar que tan acertado es la función wup\_similarity.

A continuación se muestra una lista de pares de palabras y el puntaje obtenido por cada uno de estos pares, que como se mencionó con anterioridad, un 1 representa palabras iguales mientras que un 0 son palabras sin ninguna relación.

Palabra 1	Palabra 2	Puntaje
Informatics	Science	0.941
Decisions	Actions	0.875
Work	Process	0.857
City	State	0.823
Interaction	Contact	0.933
Metal	Alum	0.875
Sensor	System	0.8
Development	Introduction	0.8
Geographic	System	0
Mathematics	Geographic	0
Chemical	Text	0.285
Space	Planet	0.769
Vector	Geographic	0
Software	Framework	0.235
Mind	Body	0.181
Sport	Tennis	0.842

Tabla 1: Comparación de palabras presentes en trabajos y el puntaje obtenido

En los resultados expuestos en la tabla 1 se puede apreciar la manera en que trabaja wup\_similarity. Pares de palabras como “Informatics” y “Science” obtienen un puntaje muy alto, ya que el campo de la Informática es justamente una ciencia. Por otro lado, palabras como “Vector” y “Geographic” obtienen un puntaje de 0, indicando que no existe relación cercana entre ellos dentro de la ontología que conforma a Wordnet.

## 4.2 Bonificaciones por repetición de palabras

Como se mencionó anteriormente, durante las comparaciones de los textos ya procesados, existe una variable llamada “Bono”, cuyo objetivo es bonificar aquellas palabras que tengan más peso dentro de un texto con respecto al resto de palabras.

Cuando se habla de “Peso” de una palabra, se refiere a la cantidad de veces que dicha palabra aparece dentro del texto, por ejemplo, sería común que en un trabajo de Inteligencia Artificial se encontrara la palabra “Heurística” repetida una cantidad considerable de veces, de ser así, dicha palabra añadirá más puntaje cuando sea comparada.

### **Fórmula de bonificación:**

$$\text{(Frecuencia Palabra1 * Frecuencia Palabra2) / Comparaciones Totales}$$

En resumen, la bonificación toma las 2 palabras elegidas para la comparación, considera las veces que se repite cada una dentro del texto y las multiplica, esa multiplicación es dividida por la cantidad de comparaciones totales entre ambos textos, finalmente le suma el valor 1 al resultado.

En el gráfico (Ver figura 18) se muestran los resultados obtenidos luego de comparar a un docente con otros 6 docentes que tengan trabajos subidos a sus perfiles. La barra azul representa el puntaje obtenido cuando el bono es aplicado, mientras que la barra roja muestra el mismo proceso, pero sin aplicar bonificación alguna. Los puntajes señalados van en una escala 0 - 100, donde 0 es el puntaje mínimo que se puede obtener, mientras que 100 es el máximo.

### Con bonificación v/s Sin bonificación

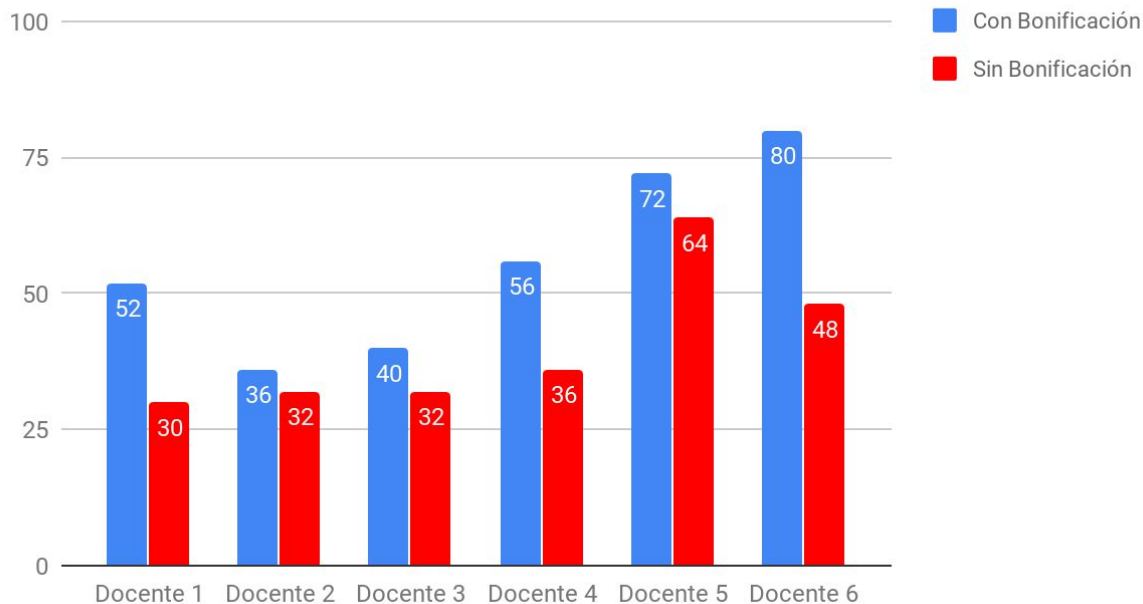


Figura 18: Gráfico comparativo de referencia entre un docente cualquiera con otros 6 al azar

Se puede observar que los puntajes aumentan en todos los casos cuando se aplica la bonificación, sin embargo, hay casos donde aumenta más que en otros, esto se debe a que en aquellos casos donde el aumento es mayor, coincide que los trabajos entre los docentes es más similar que en aquellos donde la similitud es menor.

## 4.3 Comparación de textos

Este es un aspecto que ha cambiado durante el desarrollo de la presente memoria. En un principio, la comparación de textos se hacía de forma directa entre ellos, es decir, se tomaban 2 textos, se seleccionan sus palabras relevantes y se comparaban, osea, el resultado es el mismo que el que se tiene ahora, con la excepción que el proceso para lograrlo era completo durante cada comparación, porque seleccionar las palabras relevantes de un documento era una tarea que debía hacerse en cada comparación.

Lo anterior lógicamente incrementaba de gran manera los tiempos de ejecución y carga en el servidor, por lo que se debía buscar una manera de optimizar dicho proceso.

Finalmente se decidió procesar cada texto y guardar el resultado en un documento de texto, así que cada vez que se detecta un documento nuevo, se procesa previo a la comparación, lo que disminuye los tiempos de ejecución de manera considerable.



### **Tiempo de ejecución al comparar 2 textos:**

**Sin procesar: 2 minutos**

**Ya procesados: 15 segundos**

Los tiempos anteriores son para documentos considerados de longitud promedio, con 8 páginas aproximadamente.

## **4.4 Puntaje**

Un tema crítico en cómo se determina la similaridad entre docentes es el del puntaje, ya que en caso de que este no refleje correctamente lo que debe, el sistema fallaría a la hora de dar las recomendaciones, o sea, recomendaría a docentes cuyos trabajos no son parecidos a los realizados por quien utiliza la plataforma.

En esta subsección enfatizará en el puntaje asignado de acuerdo a la similitud existente entre los trabajos.

Durante el desarrollo, el método de comparación se ha ido modificando varias veces en pos de mejorar los resultados, los cuales han ido variando de gran manera llegando al punto que se presentan hoy.

Durante las pruebas iniciales, los puntajes eran bien similares a pesar de las diferencias que pudiesen existir entre los trabajos, aunque existían variaciones en los puntajes, y los trabajos más parecidos obtenían un puntaje mayor que aquellos no tan parecidos, la diferencia no era tan notoria y tenía margen de mejora. La razón de ello era que la selección de palabras no era la mejor, ya que el algoritmo inicial consideraba como importantes palabras que no lo eran tanto, como meses del año, ciudades, países, etc. Por lo que se crearon distintas funciones que detectaban dichas palabras poco relevantes y las eliminaba de la lista final de palabras seleccionadas, lo que mejoraba los puntajes y permitía apreciar diferencias más notables dependiendo de los grados de similitud de los trabajos.

Otra mejora importante fue la inclusión de la bonificación, el cual se explica en la sección 4.1, esto ayudó a dar énfasis a las palabras con más peso dentro del texto y mejoró aún más el resultado de la asignación de puntajes.

Para mostrar los resultados, se seleccionó a un docente, que llamaremos Docente A, el cual tiene en su perfil trabajos sobre Inteligencia Artificial. Se comparará a dicho Docente A con otros 4 docentes, cuyos nombres y características son:

- Docente B: Trabajos de Inteligencia Artificial, pero diferentes que Docente A.
- Docente C: Trabajos sobre redes booleanas.
- Docente D: Trabajos de medicina usando Inteligencia Artificial.
- Docente E: Exactamente mismos trabajos que Docente A.

Los resultados obtenidos fueron:

<b>Docente 1</b>	<b>Docente 2</b>	<b>Puntaje</b>
<i>Docente A</i>	<i>Docente B</i>	<i>16.6</i>
<i>Docente A</i>	<i>Docente C</i>	<i>7.57</i>
<i>Docente A</i>	<i>Docente D</i>	<i>9.4</i>
<i>Docente A</i>	<i>Docente E</i>	<i>19.8</i>

Tabla 2: Comparación entre Docente A y otros 4 docentes

Se puede observar en la tabla 2 que, a medida que los trabajos son más similares, más puntaje se obtiene entre los docentes.

Para ampliar la experimentación se han creado 3 perfiles más que representan a 3 docentes de 3 distintos departamentos de ingeniería de la Universidad de Concepción:

- Departamento Ingeniería Civil Matemática.
- Departamento Ingeniería Civil Electrónica.
- Departamento Ingeniería Civil Química.

Para cada uno de esos 3 perfiles se creó un perfil al que se le denomina “Clon”. Dicho clon posee trabajos similares, pero no iguales. Todo esto es para simular una situación estándar, donde en la plataforma existen docentes de distintos departamentos con trabajos distintos, pero también existen docentes cuyos trabajos son similares, por lo que obtienen un puntaje mayor.

Para la primera de estas 3 comparaciones, el docente principal será el perteneciente al departamento de ingeniería civil electrónica, se denominará “Electrónico” para simplificar. Dicho docente se ha comparado con los otros 2 docentes nuevo, denominados “Químico” y “Matemático” debido a los departamentos a los que pertenecen. También se ha comparado a “Electrónico” con su clon y con un “Informático” para ampliar los resultados, los cuales fueron:

Docente 1	Docente 2	Puntaje
Electrónico	Clon	16.6
Electrónico	Informático	9.22
Electrónico	Matemático	5.68
Electrónico	Químico	1.6

Tabla 3: Comparación entre docente “Electrónico” con otros 4 docentes.

En este caso, si nos fijamos en la tabla 3, nuevamente el clon de “Electrónico” es quien obtiene un mayor puntaje, ya que como se explicó anteriormente, es quien posee los trabajos más similares. Luego tenemos que “Informático” obtiene el 2° mayor puntaje, con bastante diferencia con respecto a “Clon”, pero a la vez bastante mayor que los docentes restantes, lo cual indica la similaridad existente entre los trabajos que un Ing. Electrónico y un Ing. Informático realizan.

Para la siguiente comparación, usaremos casi los mismos docentes de la tabla 2, solo que ahora el principal será “Químico” y se cambiará el clon por el que le corresponda al docente principal, o sea, el “Clon Químico”.

Docente 1	Docente 2	Puntaje
Químico	Clon	5.67
Químico	Matemático	2.89
Químico	Informático	1.97
Químico	Electrónico	1.6

Tabla 4: Comparación entre docente “Químico” con otros 4 docentes.

En la tabla 4 se puede observar que los puntajes obtenidos son más bajos que en Tabla 2 y Tabla 3, esto es debido a que las palabras relevantes de los trabajos subidos por el docente “Químico” son más técnicas (chemical, polygon, tube, etc. Son ejemplos de esas palabras), o sea, demasiado apegadas a su especialidad, lo que puede traer dificultades a Wordnet a la hora de analizarlas y compararlas.

Ahora para la última comparación, el docente principal será “Matemático” junto a su respectivo clon, los otros 3 docentes quedan intactos.

Docente 1	Docente 2	Puntaje
Matemático	Clon	7.8
Matemático	Electrónico	5.68
Matemático	Informático	5.22
Matemático	Químico	2.89

Tabla 5 Comparación entre “Matemático” con otros 4 docentes.

Como se puede observar en los 4 experimentos anteriores, los puntajes entre cada par de docentes siempre son mayores a medida que la similitud entre sus trabajos sea más grande. Por lo que los módulos de comparación de textos y asignación de puntajes realizan su trabajo de manera correcta y dentro de lo que se podría esperar de ellos.

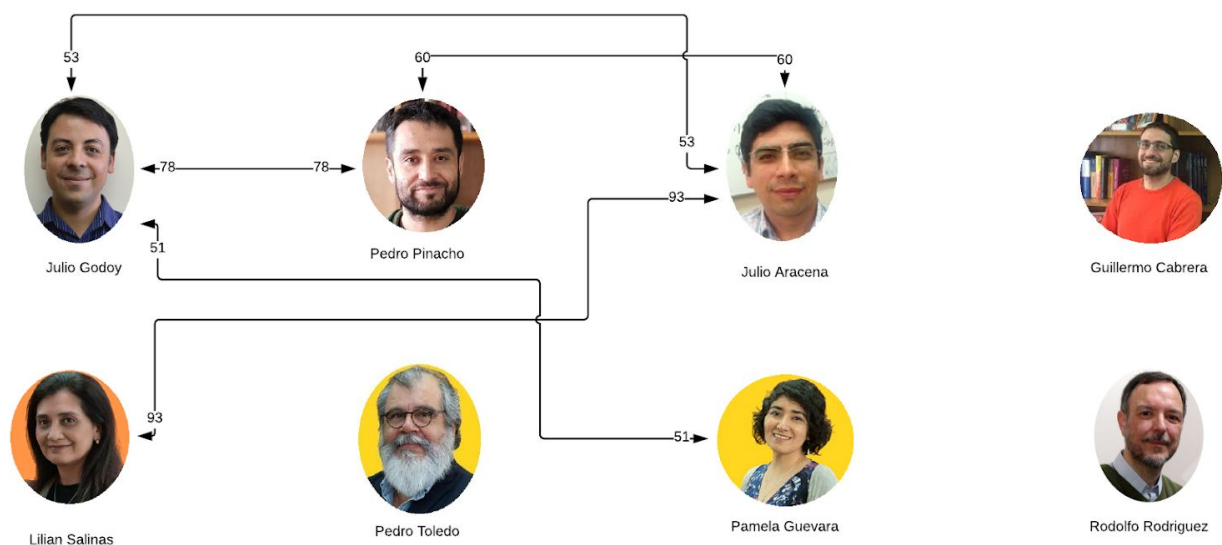


Figura 19: Gráfica que muestra las conexiones entre docentes existentes en la plataforma

En la figura 19 se muestra un ejemplo gráfico de cómo quedan unidos ciertos docentes de la Universidad de Concepción y el puntaje que les asignó la plataforma a cada par. Para realizar dichas conexiones se tomó como mínimo un puntaje de 50 en una escala de 1 - 100 para ejemplificar mejor, pero se estima que a partir de un puntaje de 70 es cuando más similitud existe entre ellos. Los docentes de la imagen son: Julio Godoy, Pedro Pinacho, Lilian Salinas y Guillermo Cabrera del Departamento de Informática, Pamela Guevara del departamento de Ing. Eléctrica, Pedro Toledo de Ing. Química, Julio Aracena y Rodolfo Rodríguez del Depto de Ing. Matemática.

## 5. Conclusiones

En esta memoria de título se propuso la creación de una herramienta que ayude en la conformación de equipos de trabajo multidisciplinarios a través de ciertas características que posean los docentes participantes. Dichas características son los intereses que tiene cada uno dentro de distintas áreas de trabajo, así como los conocimientos que poseen y los trabajos que hayan realizado con anterioridad. Después de mucho trabajo y aprendizaje durante el desarrollo de esta memoria, se puede decir que los resultados son satisfactorios (Como se puede ver en la sección 4 de Resultados).

Durante el desarrollo de este proyecto, se adquirieron diversos conocimientos que abarcan desde la conformación de un equipo de trabajo multidisciplinario, hasta las técnicas de procesamiento de lenguaje natural que existen a día de hoy y las mejoras futuras que vendrán, las que permitirán mejorar las herramientas que utilizamos a diario, tales como autocorrectores de ortografía o traductores automáticos.

Otro conocimiento importante adquirido durante el desarrollo, fue la de adaptarse a los recursos disponibles para poder desarrollar este tipo de herramientas, debido a las limitaciones de tiempo y recursos, además de otros problemas que aparecieron durante el semestre, se debieron descartar algunas ideas que posiblemente hubiesen ayudado a mejorar en parte el funcionamiento de este trabajo.

A pesar de todo, el trabajo realizado se puede calificar de satisfactorio, logrando los resultados esperados y obteniendo a cambio una plataforma funcional que cumple con los objetivos propuestos.

## 5.1 Trabajo futuro

En esta sección se detallarán algunas mejoras que quedaron pendientes en el proyecto, además de detallar algunas sugerencias que pueden ser implementadas si es que se cree necesario.

### 5.1.1 Mejorar métodos de comparación de textos

A pesar de que los métodos de comparación implementados en el proyecto entregan resultados satisfactorios, es sabido que pueden implementarse algoritmos de procesamiento de lenguaje natural que mejoren en cierta medida los resultados, pero que requerirán de equipos con mejores recursos para su funcionamiento.

### 5.1.2 Añadir mejoras a la plataforma

La plataforma web cumple con lo básico que requiere el proyecto, pero podrían añadirse algunas funcionalidades extras que mejoren la experiencia del usuario y hagan aún más fácil el uso de la misma. Algunas sugerencias son:

- Sistema de notificaciones para mensajes privados.
- Separación de trabajos subidos a la plataforma por secciones, para así permitir buscar similitudes por temas y que la comparación de trabajos similares no sea tan general.
- Hacer uso de las cuentas UdeC en la plataforma, para que cada docente tenga su propio perfil de forma inmediata con los mismos datos que posee en su cuenta UdeC.

## 6. Bibliografía

- [1] World Economic Forum. Global Sharper Survey, 2017.
  
- [2] Ciotti, G. (2014). 10 lecciones para formar un equipo. Recuperado de <https://www.entrepreneur.com/article/267237>
  
- [3] Wordnet, página oficial. 2019. <https://wordnet.princeton.edu/>
  
- [4] Jesper Segeblad. Wordnet - Structure and use in natural language processing.
  
- [5] NLTK, Página oficial, 2019. <http://www.nltk.org/>
  
- [6] Alabhya Farkiya, Prashant Saini, Shubham Sinha. Natural Language Processing using NLTK and Wordnet.
  
- [7] Avinash Navlani. Text Analytics for Beginners using NLTK. Recuperado de <https://www.datacamp.com/community/tutorials/text-analytics-beginners-nltk>
  
- [8] Wikipedia. Procesamiento de lenguajes naturales. [https://es.wikipedia.org/wiki/Procesamiento\\_de\\_lenguajes\\_naturales](https://es.wikipedia.org/wiki/Procesamiento_de_lenguajes_naturales)
  
- [9] Emiliano Marini. El Modelo Cliente/Servidor, 2012.
  
- [10] De Jesus, T.(2013). Modelo Cliente - Servidor[Figura]. Recuperado de <https://es.wikipedia.org/wiki/Cliente-servidor>
  
- [11] IBM Knowledge. Patrón de diseño Modelo - Vista - Controlador. Recuperado de

[https://www.ibm.com/support/knowledgecenter/es/SSZLC2\\_8.0.0/com.ibm.commerce.dev.elper.doc/concepts/csdmvcdespat.htm](https://www.ibm.com/support/knowledgecenter/es/SSZLC2_8.0.0/com.ibm.commerce.dev.elper.doc/concepts/csdmvcdespat.htm)

[12] Frey, R.(2010). The model, view and controller (MVC) relative to user[Figura]. Recuperado de <https://es.wikipedia.org/wiki/Modelo%E2%80%93vista%E2%80%93controlador>

[13] Netbeans, página oficial, 2019. Recuperado de <https://netbeans.org/>

[14] Adrienne Watt. *Database Design 2nd edition*, Chapter 8.

[15] Visual Studio Code, página oficial, 2019. Recuperado de <https://code.visualstudio.com/>

[16] Documentación de MySQL, 2019. Recuperado de <https://dev.mysql.com/doc/>

[17] Chan, R.(2019). The 10 most popular programming languages, according to the 'Facebook for programmers'. Recuperado de <https://www.businessinsider.com/the-10-most-popular-programming-languages-according-to-github-2018-10#2-java-9>

[18] Documentación de Wordnet. Recuperado de <http://www.nltk.org/howto/wordnet.html>