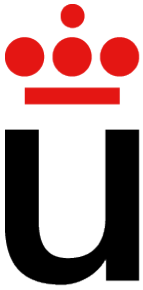

Universidad
Rey Juan Carlos

GRADO EN INGENIERÍA EN SISTEMAS AUDIOVISUALES Y MULTIMEDIA

Curso Académico 2021/2022

Trabajo Fin de Grado

IMPLEMENTACIÓN DE FUNCIONALIDADES EN LEARNINGML: RECONOCIMIENTO DE BASES DE DATOS

Autor : Ignacio Rueda Rodríguez

Tutor : Dr. Gregorio Robles Martínez

Co-tutor : Juan David Rodríguez García

Trabajo Fin de Grado

Implementación de Funcionalidades en LearningML: Reconocimiento de Bases de Datos

Autor : Ignacio Rueda Rodríguez

Tutor : Dr. Gregorio Robles Martínez

Co-tutor : Juan David Rodríguez García

La defensa del presente Proyecto Fin de Carrera se realizó el día de de 2022, siendo calificada por el siguiente tribunal:

Presidente:

Secretario:

Vocal:

y habiendo obtenido la siguiente calificación:

Calificación:

Fuenlabrada, a de de 2022

Dedicado a

mi familia, abuelos y amigos

II

Agradecimientos

Aquí vienen los agradecimientos... Aunque está bien acordarse de la pareja, no hay que olvidarse de dar las gracias a tu madre, que aunque a veces no lo parezca disfrutará tanto de tus logros como tú... Además, la pareja quizás no sea para siempre, pero tu madre sí.

Resumen

Aquí viene un resumen del proyecto. Ha de constar de tres o cuatro párrafos, donde se presente de manera clara y concisa de qué va el proyecto. Han de quedar respondidas las siguientes preguntas:

- ¿De qué va este proyecto? ¿Cuál es su objetivo principal?
- ¿Cómo se ha realizado? ¿Qué tecnologías están involucradas?
- ¿En qué contexto se ha realizado el proyecto? ¿Es un proyecto dentro de un marco general?

Lo mejor es escribir el resumen al final.

Summary

Here comes a translation of the “Resumen” into English. Please, double check it for correct grammar and spelling. As it is the translation of the “Resumen”, which is supposed to be written at the end, this as well should be filled out just before submitting.

Índice general

Índice de figuras

Capítulo 1

Introducción

En este capítulo se introduce el proyecto. Debería tener información general sobre el mismo, dando la información sobre el contexto en el que se ha desarrollado.

No te olvides de echarle un ojo a la página con los cinco errores de escritura más frecuentes[1].

Aconsejo a todo el mundo que mire y se inspire en memorias pasadas. Las memorias de los proyectos que he llevado yo están (casi) todas almacenadas en mi web del GSyC[2].

Dado que hoy

1.1. Sección

Esto es una sección, que es una estructura menor que un capítulo.

Por cierto, a veces me comentáis que no os compila por las tildes. Eso es un problema de codificación. Al guardar el archivo, guardad la codificación de "ISO-Latin-1" a "UTF-8" (o viceversa) y funcionará.

1.1.1. Estilo

Recomiendo leer los consejos prácticos sobre escribir documentos científicos en LaTeX de Diomidis Spinellis[3].

[1] `http://www.tallerdeescritores.com/errores-de-escritura-frecuentes`
[2] `https://gsyc.urjc.es/~grex/pfcs/`
[3] `https://github.com/dspinellis/latex-advice`

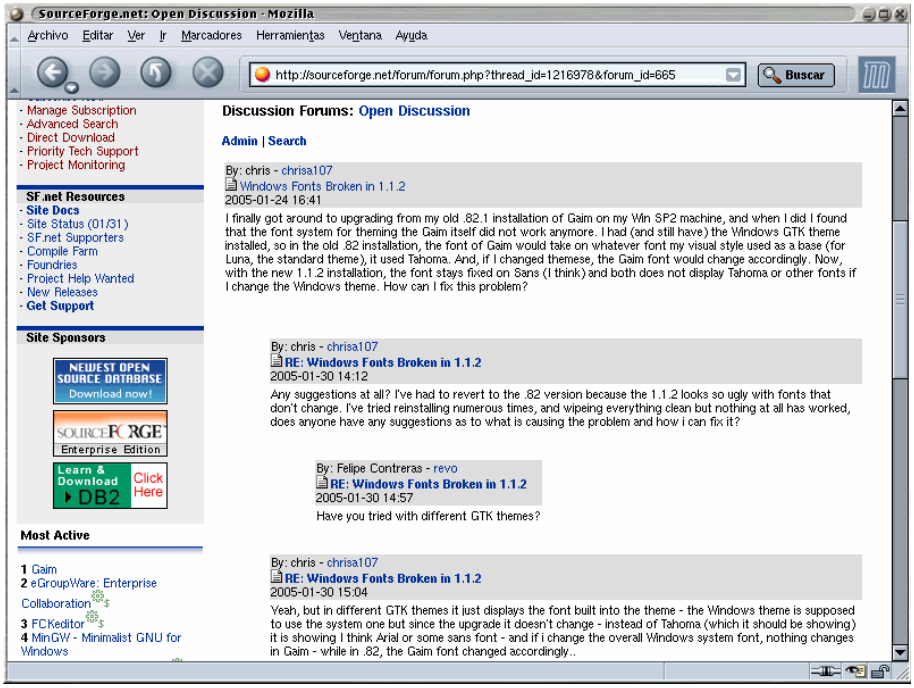

Figura 1.1: Página con enlaces a hilos

Lee sobre el uso de las comas[4]. Las comas en español no se ponen al tuntún. Y nunca, nunca entre el sujeto y el predicado (p.ej. en "Yo, hago el TFG" sobre la coma). La coma no debe separar el sujeto del predicado en una oración, pues se cortaría la secuencia natural del discurso. No se considera apropiado el uso de la llamada coma respiratoria o *coma criminal*. Solamente se suele escribir una coma para marcar el lugar que queda cuando omitimos el verbo de una oración, pero es un caso que se da de manera muy infrecuente al escribir un texto científico (p.ej. "El Real Madrid, campeón de Europa").

A continuación, viene una figura, la Figura 1.1. Observarás que el texto dentro de la referencia es el identificador de la figura (que se corresponden con el "label" dentro de la misma). También habrás tomado nota de cómo se ponen las "comillas dobles" para que se muestren correctamente. Nota que hay unas comillas de inicio (``) y otras de cierre (''), y que son diferentes. Volviendo a las referencias, nota que al compilar, la primera vez se crea un diccionario con las referencias, y en la segunda compilación se "rellenan" estas referencias. Por eso hay que compilar dos veces tu memoria. Si no, no se crearán las referencias.

A continuación un bloque "verbatim", que se utiliza para mostrar texto tal cual. Se puede

[4] `http://narrativabreve.com/2015/02/opiniones-de-un-corrector-de-estilo-11-recetas-par`
`html`

utilizar para ofrecer el contenido de correos electrónicos, código, entre otras cosas.

```
From gaurav at gold-solutions.co.uk  Fri Jan 14 14:51:11 2005
From: gaurav at gold-solutions.co.uk (gaurav_gold)
Date: Fri Jan 14 19:25:51 2005
Subject: [Mailman-Users] mailman issues
Message-ID: <003c01c4fa40$1d99b4c0$94592252@gaurav7klgnyif>

Dear Sir/Madam,
How can people reply to the mailing list?  How do i turn off
this feature? How can i also enable a feature where if someone
replies the newsletter the email gets deleted?
Thanks

From msapiro at value.net  Fri Jan 14 19:48:51 2005
From: msapiro at value.net (Mark Sapiro)
Date: Fri Jan 14 19:49:04 2005
Subject: [Mailman-Users] mailman issues
In-Reply-To: <003c01c4fa40$1d99b4c0$94592252@gaurav7klgnyif>
Message-ID: <PC17302005011410485105780lb04d55@msapiro>

gaurav_gold wrote:
>How can people reply to the mailing list?  How do i turn off
this feature? How can i also enable a feature where if someone
replies the newsletter the email gets deleted?

See the FAQ
>Mailman FAQ: http://www.python.org/cgi-bin/faqw-mm.py
article 3.11
```

1.2. Estructura de la memoria

En esta sección se debería introducir la esctura de la memoria.

Así:

- En el primer capítulo se hace una intro al proyecto.
- En el capítulo 2 (ojo, otra referencia automática) se muestran los objetivos del proyecto.

- A continuación se presenta el estado del arte en el capítulo 3.
- ...

Capítulo 2

Objetivos

2.1. Objetivo general

El trabajo fin de grado consiste en añadir a la aplicación web LearningML el reconocimiento de bases de datos.

Recuerda que los objetivos siempre vienen en infinitivo.

2.2. Objetivos específicos

Los objetivos específicos se pueden entender como las tareas en las que se ha desglosado el objetivo general. Y, sí, también vienen en infinitivo.

2.3. Planificación temporal

A mí me gusta que aquí pongáis una descripción de lo que os ha llevado realizar el trabajo. Hay gente que añade un diagrama de GANTT. Lo importante es que quede claro cuánto tiempo llevas (tiempo natural, p.ej., 6 meses) y a qué nivel de esfuerzo (p.ej., principalmente los fines de semana).

Capítulo 3

Estado del arte

3.1. Inteligencia Artificial

La IA (*Inteigencia Artificial*) [9] se puede definir como la habilidad de los ordenadores para hacer actividades que normalmente requieren inteligencia humana. Aunque de forma más técnica se puede definir como la capacidad de las máquinas para usar algoritmos, aprender de los datos y utilizar lo aprendido en la toma de decisiones como si se tratase de un humano.

El uso de la IA cada vez es mayor, ya que nos ayuda a beneficiarnos de mejoras significativas y disfrutar de una mayor eficiencia en casi todos los ámbitos de la vida. Pero debemos estar atentos para prevenir y analizar las posibles desventajas directas o indirectas que pueda generar el crecimiento del uso de la IA. La IA se puede aplicar a una inmensidad de situaciones, algunas de ellas son las siguientes:

- Cambiará la forma de hacer negocios debido a que las empresas que busquen entender y aplicar estas herramientas de forma rápida y eficaz obtendrán ventajas competitivas. Por ejemplo, mejoras del desempeño de la estrategia algorítmica comercial o detección y clasificación de objetos.

- La IA será capaz de ofrecernos sugerencias y predicciones relacionadas con temas importantes de nuestra vida, esto supondrá un impacto en áreas como la salud, el bienestar, la educación, el trabajo y las relaciones interpersonales. Por ejemplo, el procesamiento eficiente y escalable de datos de pacientes implicará que la atención médica sea más efectiva y eficiente.

- Permitirá que las máquinas y los robots realicen tareas que los humanos consideran difíciles, aburridas o peligrosas. Además que las máquinas no necesitan descansar y pueden analizar grandes volúmenes de información a la vez y con un porcentaje de error menor que los humanos.

3.2. Machine Learning

El machine learning o aprendizaje automático [9] es uno de los enfoques fundamentales de la inteligencia artificial. Se trata un elemento de la informática en el que los ordenadores o las máquinas tienen la capacidad de aprender sin estar programados para ello. El aprendizaje automático usa algoritmos para aprender de los patrones de datos. Un ejemplo es la personalización de los sitios de medios sociales como Facebook o los resultados del motor de búsqueda de Google. Otro ejemplo son los filtros de Spam en el correo electronico. Aprenden en base a unos patrones que tipo de mensajes son correo basura y cuales no y toman la decision de clasificarlos como tal o no.

En el aprendizaje automático se diferencian tres tipos de aprendizaje:

- **Aprendizaje supervisado:** es el humano quien tiene que suministrar datos etiquetados y organizados para indicar cómo tendría que ser categorizada la nueva información o entrada. Por ejemplo, enseñar previamente al algoritmo fotos donde aparezca un perro para que luego pueda identificar imágenes similares. Es el tipo de aprendizaje utilizado en LearningML.

- **Aprendizaje no supervisado:** es el algoritmo el que tiene que encontrar la forma de asignar a los datos una etiqueta sin información previa, de manera que no precisa de la intervención de un humano.

- **Aprendizaje por refuerzo:** el algoritmo aprende de la experiencia. Cuando el algoritmo acierta al clasificar una entrada recibe un refuerzo positivo y de esta forma mejora con el uso.

3.3. LearningML

3.4. Angular

Angular [6, 2] nació en 2010 con el nombre de AngularJS pero en el año 2016 pasó a llamarse Angular, en su versión 2.0 y ha ido evolucionando e implantando mejoras. La última versión es la 13 publicada en noviembre de 2021.

Angular es un *framework* de código abierto desarrollado por Google que mediante el uso de TypeScript y HTML permite crear aplicaciones de una sola página, denominadas SPA (*Single Page Application*).

Las ventajas que tiene Angular es que mantiene la aplicación más ordenada, simplifica el codigo y evita escribir código repetitivo gracias a que sigue un modelo MVC (*Modelo-Vista-Controlador*), a la vez que posibilita que las modificaciones y las actualizaciones de las aplicaciones sean rápidas y sencillas. Además separa el *frontend* y el *backend* en la aplicación.

La principal ventaja que tienen las SPA es que tiene una alta velocidad de carga entre las vistas que tiene la aplicación web, ya que cuando hay un cambio de vista no se recarga la página si no que las vistas se cargan de forma rápida, dinámica y reactiva. Esto se debe a que solamente hay una petición inicial al servidor y una respuesta HTML del mismo, luego funciona mediante routing.

Como cualquier *framework*, Angular tiene una serie de librerías que podemos importar para facilitar el desarrollo del código, o para solucionar problemas concretos. Las librerías amplían las funcionalidades.

La arquitectura de una aplicación Angular está basada en cuatro clases distintas, que se identifican a través de decoradores. Estos decoradores definen su tipo y proporcionan metadatos que le indican a Angular cómo usarlos. Las cuatro clases que utiliza son:

- **Módulos:** declaran un contexto de compilación para un conjunto de componentes. Los módulos juegan un papel fundamental en la estructuración de las aplicaciones Angular. Es donde se definen o declaran los componentes, las directivas y los sevicios que conforman la aplicación. De manera que representa una agrupación lógica de lo que podríamos llamar áreas funcionales de una aplicación. También definen las rutas que establecen las vistas de la aplicación y las dependencias con otros módulos, es decir, que módulos necesita

importar y que componentes o directivas exporta.

Cualquier aplicación de Angular tiene un módulo raiz, llamado `AppModule`, que proporciona el mecanismo de arranque que inicia la aplicación, pero no es el único, normalmente una aplicación contiene varios módulos funcionales, además de los propios del *framework*.

Los módulos de Angular pueden importar o exportar funcionalidades de otros módulos, esto hace que cada módulo sea independiente. La organización en distintos módulos ayuda a gestionar el desarrollo de aplicaciones complejas y a la reutilización de código.

Para definir una clase como módulo se utiliza el decorador `@NgModule`.

- **Componentes:** son los bloques de construcción que componen una aplicación, contienen la lógica y los datos de la aplicación. Cada componente tiene asociada una plantilla HTML. Las plantillas son las vistas que conforman la interfaz de usuario y se cargan cuando se cambia o modifica la URL.

  Una aplicación de Angular tiene al menos el componente raiz, llamado `AppComponent`, que conecta una jerarquía de componentes con el modelo de objeto del documento (DOM) de la página.

  Los componentes pueden tener uno o varios subcomponentes, estos pueden relacionarse entre sí de dos formas. Mediante eventos para actualizar datos cuando el usuario interacciona con la interfaz grafica y mediante las propiedades o interpolación para enviar datos a las plantillas que conforman las vistas. Esto hace que los cambios en el DOM pueden modificar los datos de la aplicación y los datos de la aplicación pueden modificar las plantillas.

  Para definir una clase como componente se utiliza el decorador `@NgComponent`.

- **Directivas:** son clases que agregan comportamiento adicional a los elementos en sus aplicaciones Angular para modificar de alguna manera el HTML. Antes de que se muestre una vista, Angular evalúa las directivas y resuelve la sintaxis vinculante en la plantilla para modificar los elementos HTML y el DOM.

  Hay tres tipos de directivas:

  * Directivas de componente: los componentes son un tipo de directiva.
  * Directivas de atributos: modifican el comportamiento o la apariencia de un elemento, componente o directiva.
  * Directivas estructuales: modifican la apariencia agregando y eliminando elementos del DOM.

  Se pueden crear directivas o Angular dispone de directivas integradas que se pueden usar para administrar formularios, listas, estilos y lo que ven los usuarios. Algunos ejemplos de directivas de atributos integradas son `NgClass`, `NgStyle` y `NgModel`, y de directivas estructurales `NgIf`, `NgFor` y `NgSwitch`.

- **Servicios:** proporcionan una funcionalidad específica que no está directamente relacionada con las vistas. Son clases con un propósito limitado y bien definido, deben hacer algo específico y de uso general. Se usan para cambiar datos entre componentes, obtener datos del servidor, validar la entrada del usuario o cualquier servicio que se repita en distintas componentes.

  Angular distingue los componentes de los servicios para aumentar la modularidad y la reutilización. Al separar la funcionalidad relacionada con la vista de un componente de otros tipos de procesamiento, puede hacer que sus clases de componentes sean sencillas y eficientes.

  Para definir una clase como servicio se utiliza el decorador `@Injectable`,ya que proporciona los metadatos que permiten inyectar otros proveedores como dependencias en su clase.

3.4.1. Angular CLI

Angular CLI (*Command Line Interface*) [7] es una herramienta de interfaz de línea de comandos desarrollada por Angular que se utiliza para inicializar, desarrollar, montar y mantener aplicaciones de Angular directamente desde un shell de comandos.

Es una herramienta que facilita el inicio de una aplicación de Angular, porque con una instrucción `ngnew`, crea una carpeta de trabajo y genera el esqueleto de aplicación. También tiene instrucciones que permiten crear nuevas clases de forma sencilla, creando los distintos

archivos que forman un módulo, componente, servicio o una directiva, además de incluirlos en los archivos necesrios y crear las dependencias. Las herramientas predefinidas más destacadas son el servidor web, el compilador y el sistema de testing.

3.4.2. MVC

Para el desarrollo de una aplicación web Angular utiliza el patrón MVC (*Modelo-Vista-Controlador*) [1] que se utiliza para separar en tres componentes los datos, la metodología y la interfaz gráfica de una aplicación, lo que permite modificar cada uno de ellos sin tener que modificar los demas. La arquitectura MVC está formada por tres componentes, que son:

- **Modelo:** es el encargado de la manipulación, gestión y actualización de los datos. En el caso de que haya una base de datos es donde se realizan las consultas, busquedas o actualizaciones.

- **Vista:** es la representación gráfica de los datos proporcionados por el controlador. Ni el controlador ni el modelo se preocupan de cómo se verán los datos, toda la parte del diseño de la interfaz es responsabilidad de la vista.

- **Controlador:** es el componente principal, se encarga de gestionar las instrucciones que se reciben del usuario, atenderlas y procesarlas. Después realiza las consultas al modelo y una vez se hayan obtenido dichos datos, se envía a la vista para producir una salida como respuesta a el evento.

Las ventajas que aporta esta arquitectura son:facilita el mantenimiento, permite la reutilización de componentes y mejora la escalabilidad, ya que separa cada tipo de lógica.

3.5. HTML5

HTML (*HyperText Markup Language*) [4, 5, 8] es un lenguaje de marcado que se utiliza para definir la estructura y el contenido de una página Web, y es el componente más básico de una web. Se escribe en texto plano y tiene una estructura de árbol de elementos y textos. La estructura básica de un documento HTML se compone de los siguientes elementos: `<!DOCTYPE>`, `<html>`, `<head>` y `<body>`. HTML provee los elementos estructurales y se complemeta

con CSS para modificar la apariencia de la pagina y con JavaScript para proveer dinamismo y construir aplicaciones web completamente funcionales.

La idea de desarrollar HTML5 nació en 2004, cuando el consorcio W3C (*World Wide Web Consortium*) que está a cargo del estandar deció dejar de evolucionar HTML, por lo que la asociación WHATWG (*Web HyperText Application Technology Working Group*), formada por Apple, Opera y Mozilla, decidió crear HTML5 y publicó el primer borrador de HTML5. En el año 2007 W3C formó un grupo de trabajo autorizado para trabajar con WHATWG en el desarrollo de HTML5, pero en 2011 se separaron debido a que tenían distintos objetivos, W3C quería publicar una version terminada, mientras que WHATWG quería seguir trabajando en una constante evolución de HTML5. En 2019, WHATWG y W3C firmaron un acuerdo para colaborar en una única versión de HTML en el futuro.

Fue creado con la intención de hacerlo más eficiente que las versiones anteriores, mantener la compatibilidad con versiones anteriores y facilitar el desarrollo web compatible con distintos navegadores. Las principales novedades de HTML5 son:

- Incorpora nuevas etiquetas que permiten una mejor estructuración de los documentos HTML, ya que antes solo estaba el `<div>` para definir secciones. Ahora se pueden utilizar también `<header>` para la cabecera, `<section>` para la información principal, `<nav>` para la barra de navegación y `<footer>` para el pie, entre otros.

- Permite incorporar elementos multimedia para reproducir audios y videos desde el propio navegador, con las etiquetas `<audio>` y `<video>`.

- Incluye numerosas APIs, algunas de ellas son:

    * Forms: incluye mejoras en los formularios para personalizar todos los aspectos de procesamiento y validación.

    * Canvas: permite dibujar, presentar gráficos en pantalla, animar y procesar imágenes y texto.

    * Geolocation: permite a los desarrolladores determinar la ubicación física real del usuario y e mostrar e interactuar con un mapa de Google Maps.

    * Drag and drop: permite arrastrar un elemento desde un lugar y luego soltarlo en otro.

* Web Storage: es una mejora de las cookies y permite el almacenamiento local en el lado del cliente, con `localStorage` cuando la información tiene que estar disponible solo durante la sesión y `sessionStorage` cuando tiene que ser preservada todo el tiempo que el usuario desee

3.6. CSS3

3.7. TypeScript

TypeScript [3] es un lenguaje de programación de código abierto desarrollado y mantenido por Microsoft. Es un superconjunto de JavaScript, es decir, amplía JavaScript con una nueva sintaxis que añade tipos estáticos y objetos basados en clases. Nació en 2012 como solución para el desarrollo de aplicaciones a gran escala, ya que con JavaScript era muy complicado. Los programas de JavaScript son compatibles en TypeScript, por lo que se puede integrar en proyectos ya existentes porque a través de un compilador de TypeScript se traducen a código JavaScript original. Los cambios más importantes respecto a JavaScript son:

Evita errores en tiempo de ejecución, ya que incorpora el tipado estático, es decir, al crear variables se puede añadir el tipo de dato.

Al añadir objetos basados en clases hace que la programación orientada a objetos sea más sencilla, esto implica que sea más funcional.

Capítulo 4

Diseño e implementación

Aquí viene todo lo que has hecho tú (tecnológicamente). Puedes entrar hasta el detalle. Es la parte más importante de la memoria, porque describe lo que has hecho tú. Eso sí, normalmente aconsejo no poner código, sino diagramas.

4.1. Arquitectura general

Si tu proyecto es un software, siempre es bueno poner la arquitectura (que es cómo se estructura tu programa a "vista de pájaro").

Por ejemplo, puedes verlo en la figura 4.1. LaTeX pone las figuras donde mejor cuadran. Y eso quiere decir que quizás no lo haga donde lo hemos puesto... Eso no es malo. A veces queda un poco raro, pero es la filosofía de LaTeX: tú al contenido, que yo me encargo de la maquetación.

Recuerda que toda figura que añadas a tu memoria debe ser explicada. Sí, aunque te parezca evidente lo que se ve en la figura 4.1, la figura en sí solamente es un apoyo a tu texto. Así que explica lo que se ve en la figura, haciendo referencia a la misma tal y como ves aquí. Por ejemplo: En la figura 4.1 se puede ver que la estructura del *parser* básico, que consta de seis componentes diferentes: los datos se obtienen de la red, y según el tipo de dato, se pasará a un *parser* específico y bla, bla, bla...

Si utilizas una base de datos, no te olvides de incluir también un diagrama de entidad-relación.

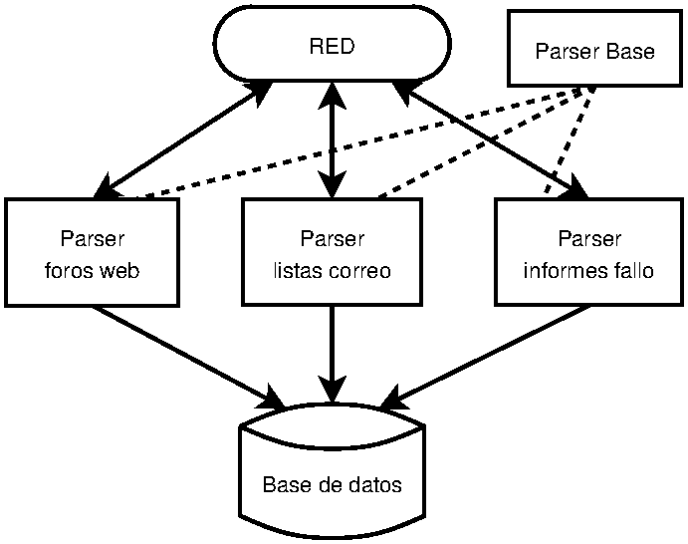

Figura 4.1: Estructura del parser básico.

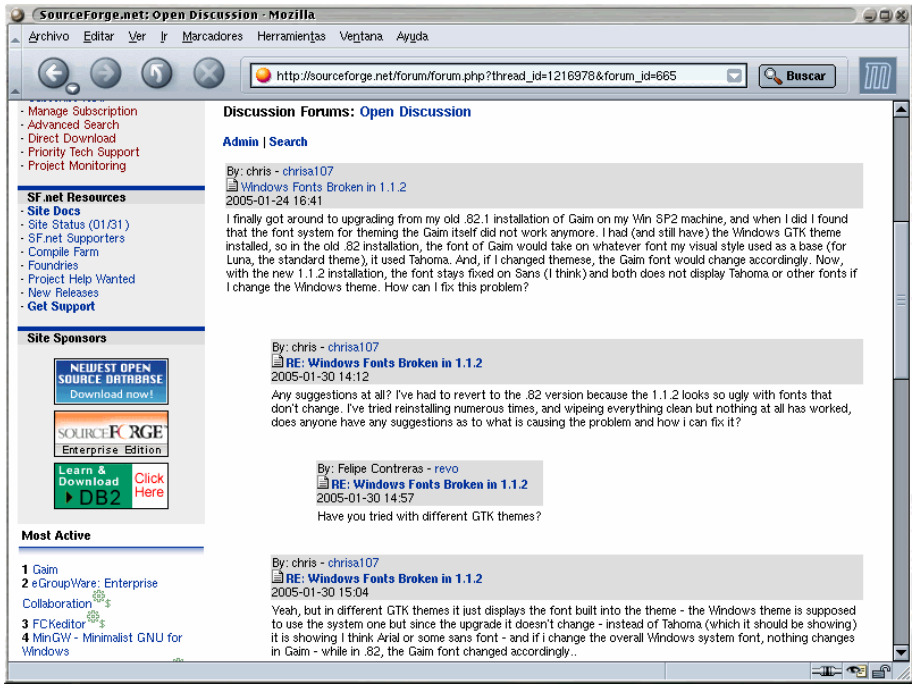

Figura 4.2: Página con enlaces a hilos

Capítulo 5

Experimentos y validación

Este capítulo se introdujo como requisito en 2019. Describe los experimentos y casos de test que tuviste que implementar para validar tus resultados. Incluye también los resultados de validación que permiten afirmar que tus resultados son correctos.

Capítulo 6

Resultados

En este capítulo se incluyen los resultados de tu trabajo fin de grado.

Si es una herramienta de análisis lo que has realizado, aquí puedes poner ejemplos de haberla utilizado para que se vea su utilidad.

Capítulo 7

Conclusiones

7.1. Consecución de objetivos

Esta sección es la sección espejo de las dos primeras del capítulo de objetivos, donde se planteaba el objetivo general y se elaboraban los específicos.

Es aquí donde hay que debatir qué se ha conseguido y qué no. Cuando algo no se ha conseguido, se ha de justificar, en términos de qué problemas se han encontrado y qué medidas se han tomado para mitigar esos problemas.

Y si has llegado hasta aquí, siempre es bueno pasarle el corrector ortográfico, que las erratas quedan fatal en la memoria final. Para eso, en Linux tenemos aspell, que se ejecuta de la siguiente manera desde la línea de *shell*:

```
aspell --lang=es_ES -c memoria.tex
```

7.2. Aplicación de lo aprendido

Aquí viene lo que has aprendido durante el Grado/Máster y que has aplicado en el TFG/TFM. Una buena idea es poner las asignaturas más relacionadas y comentar en un párrafo los conocimientos y habilidades puestos en práctica.

1. a

2. b

7.3. Lecciones aprendidas

Aquí viene lo que has aprendido en el Trabajo Fin de Grado/Máster.

1. Aquí viene uno.
2. Aquí viene otro.

7.4. Trabajos futuros

Ningún proyecto ni software se termina, así que aquí vienen ideas y funcionalidades que estaría bien tener implementadas en el futuro.

Es un apartado que sirve para dar ideas de cara a futuros TFGs/TFMs.

Apéndice A

Manual de usuario

Esto es un apéndice. Si has creado una aplicación, siempre viene bien tener un manual de usuario. Pues ponlo aquí.

Bibliografía

[1] Características mvc.
https://seopromarketing.online/que-es-mvc-ventajas.

[2] Curso de angular.
https://youtu.be/fXpMiweCC_o.

[3] Pagina de typescript.
https://www.typescriptlang.org.

[4] Pagina sobre html.
https://www.w3schools.com/html/html_intro.asp.

[5] Pagina sobre html.
https://html.spec.whatwg.org/multipage.

[6] Página de angular.
https://angular.io.

[7] Página de angular cli.
https://cli.angular.io.

[8] J. D. Gauchat. *El gran libro de HTML5, CSS3 y Javascript*. Marcombo, 2012.

[9] L. Rouhiainen. Inteligencia artificial. *Madrid: Alienta Editorial*, 2018.