ULPGC
Universidad de
Las Palmas de
Gran Canaria

Escuela de
Ingeniería Informática

Grafo de familias de palabras

Titulación: Grado en Ingeniería Informática

Autor: Álvaro Rodríguez Martín

Tutorizado por:

Francisco Javier Carreras Riudavets

Fecha: Junio 2021

Índice

Organización de la memoria

Comenzaremos este Trabajo de Fin de Grado con un breve resumen de los contenidos que veremos a continuación en la memoria.

- En la **introducción**, se hará una breve descripción del trabajo realizado, explicaremos en qué consisten los grafos y finalmente describiremos que son las familias de palabras
- Seguidamente, en el **estado del arte**, se verán algunas herramientas que existen en Internet en estos momentos, así como diferentes frameworks de desarrollo.
- En los **objetivos** se mencionarán los objetivos generales que se desean obtener con la realización de este Trabajo de Fin de Grado, junto con otros objetivos más personales que el alumno quiere obtener.
- En la **motivación**, se explicará porque se ha elegido realizar este proyecto en concreto.
- Luego, en la **justificación de las competencias**, se hará una comparativa entre el trabajo que se ha realizado y las competencias que se han adquirido durante la carrera universitaria.
- En los **recursos**, se mencionarán y se justificará que recursos se han usado durante la realización del proyecto.
- En el **análisis** se describirán los estudios realizados, previos al comienzo del trabajo y durante la realización de este.
- A continuación, en el **diseño** se mostrará el diseño base que se propuso para la aplicación web.
- Seguidamente, en el **desarrollo** se detallarán los pasos seguidos durante la realización de este trabajo.
- En la **conclusión**, se realizará un pequeño resumen de lo que se ha mencionado anteriormente en la memoria, junto con una breve reflexión sobre lo aprendido durante la creación de este Trabajo de Fin de Grado
- Por último, en los **trabajos futuros** se mencionarán cambios o mejoras que se pueden realizar en el trabajo si en un futuro se continúa trabajando en con proyecto.

Introducción

Descripción del Trabajo de Fin de Grado

Durante la realización de este trabajo de fin de grado titulado “Grafo de familias de palabras” se ha realizado una aplicación web en la cual el usuario introduce una palabra y la aplicación le muestra uno o varios botones. Dichos botones, aunque lo explicaremos más tarde de manera detallada, dependen de la forma canónica de la palabra introducida y de sus primitivas. Una vez el usuario hace clic en un botón, se despliega un nuevo contenedor donde aparecerá el grafo seleccionado.

Para este trabajo de fin de grado solo se muestran tres niveles del grafo. En el primer nivel, se encuentra la primitiva de la palabra introducida por el usuario, si existe. En el segundo nivel, se encuentran la palabra introducida por el usuario y sus hermanos, si dicha palabra tenía primitiva, en caso contrario, solo se muestra la palabra introducida por el usuario y, finalmente, en el último nivel, se muestran las derivadas de dicha palabra.

Para este trabajo de fin de grado solo se mostrarán 3 niveles del grafo. En el primer nivel, encontraremos la primitiva de la palabra introducida por el usuario si existe. En el segundo nivel encontraremos la palabra introducida por el usuario y sus hermanos si dicha palabra tenía primitiva en caso contrario solo se mostrará la palabra introducida por el usuario, y finalmente, en el último nivel se mostrará las derivadas de dicha palabra.

¿Qué es un grafo?

Un grafo es un conjunto de puntos unidos entre sí mediante un conjunto de conexiones en forma de líneas, utilizado para representar un proceso o relación funcional. Dichos puntos son comúnmente llamados nodos del grafo y las líneas que los conectan se conocen como aristas. Existen varios tipos de grafos de los cuales los más conocidos son los grafos dirigidos, no dirigidos y etiquetados.

Un grafo dirigido, también llamado dígrafo, se compone de un número de nodos y aristas, donde cada arista asocia un nodo con otro de una forma unidireccional, haciendo uso de una flecha en un extremo de la arista. Debido a esta fleca, el grafo solo se puede recorrer en una dirección determinada.

Un grafo no dirigido, es igual que un grafo dirigido, pero a diferencia del grafo dirigido, las aristas no tienen una flecha especificando en qué sentido se recorre el grafo, lo que significa que dicha arista puede recorrerse desde cualquiera de sus puntos y en cualquier dirección.

Por último, los grafos etiquetados o grafos dirigidos con pesos consisten en que las aristas de dicho grafo poseen información adicional, donde se suele plantear una relación entre los nodos que dicha arista conecta.[1]

En este trabajo, hemos utilizado los grafos dirigidos para expresar la relación entre la palabra raíz y sus derivadas.

¿Qué son las familias de palabras?

Una familia de palabras, también conocida como familia léxica, es un conjunto de palabras que derivan de la misma raíz, habitualmente, mediante la adición de un afijo que suele aportar un significado respecto al de su raíz, con lo que se establece un vínculo entre el lexema o raíz y la derivada.

Estas familias están divididas entre palabras primitivas y palabras derivadas. Una palabra se considera que es primitiva cuando no posee un origen que viene derivado de otra palabra, es decir, su origen no se encuentra en otra palabra del mismo idioma. Dicha palabra es la raíz de otras palabras que se generan a partir de ella, normalmente, mediante la adición de un sufijo o de un prefijo, o de ambos a la vez.

En cambio, una palabra derivada es aquella que se ha formado a partir de otra existente mediante la adición de un afijo derivativo. De esa manera, se consigue, partiendo de una palabra, obtener una nueva palabra derivada que comparte la misma raíz y un significado relacionado con la palabra de partida.

Las palabras de una familia se encuentran siempre en forma canónica, lo cual significa que dichas palabras no están flexionadas o conjugadas. Una forma canónica es la palabra que representa un conjunto de términos que se pueden generar desde ella mediante los procesos de flexión (género y número) o mediante su conjugación. Normalmente, estas palabras son las que se encuentran en los diccionarios. Esto es debido a que, si “aparcamiento” está relacionado con “aparcar”, también lo estará con su forma en plural “aparcamientos” y, por ello, es innecesario mostrar en el grafo todas las flexiones o conjugaciones de las palabras que pertenecen a la familia.

Con el fin de facilitar el conocimiento de los usuarios sobre estos temas de la aplicación web creada, se realiza un tratamiento lingüístico previo para identificar si la palabra introducida por el usuario es una flexión, una forma canónica o ambos casos, con el objetivo de aclarar la ambigüedad del lenguaje.

[1] Grafos https://www.grapheverywhere.com/tipos-de-grafos/ Fecha de consulta: 07/06/2021

Ambigüedad del español

Debido a las particularidades características del español, como las flexiones o palabras homógrafas, es necesario realizar un tratamiento previo de reconocimiento lingüístico para identificar la familia de la palabra que el usuario esta solicitando. Por ejemplo, la palabra “para” puede hacer referencia a la preposición “para” la cual no tiene familia, a varias formas conjugadas del verbo “parir”, a varias formas conjugadas del verbo “parar”. En caso de tener familia de palabras estas serían distintas. Otro ejemplo sería la palabra “vino”, la cual puede referirse al sustantivo “vino” de una bebida, o a la forma conjugada “vino” del verbo “venir” que pertenece a otra familia.

Estado del arte

El desarrollo de este proyecto ha implicado el estudio de diversas herramientas y tecnologías disponibles en la actualidad. De las herramientas que se han estudiado, se puede destacar el lenguaje Java, JavaScript y C#. Así como librerías para facilitar la representación gráfica de datos, de ellas, la que más se acercaba a cubrir las necesidades de este trabajo es la D3.js la cual se explicará posteriormente.

En la actualidad, no existe ninguna aplicación web de forma pública sin necesidad de registro, que permita obtener una representación gráfica de las familias de palabras. Por este motivo, se ha querido realizar este proyecto para aportar visibilidad gráfica.

La tecnología usada para este trabajo ha sido ASP.NET junto con los lenguajes C# y JavaScript. Esto ha sido así, ya que, una vez terminado este trabajo, se incorporará a las herramientas lingüísticas proporcionadas por el Instituto Universitario de Análisis y Aplicaciones Textuales (IATEXT) de la Universidad de Las Palmas de Gran Canaria (ULPGC) cuyo servidor es un Windows Server con IIS.

Objetivos

El objetivo principal de este Trabajo de Fin de Grado ha sido la realización de la aplicación web **Familias TIP** en el marco de la investigación del Instituto Universitario de Análisis y Aplicaciones Textuales. Dicha aplicación muestra, de forma gráfica, el grafo de la familia de la palabra que el usuario proporciona. Este grafo es interactivo y permite diferentes funcionalidades, las cuales se explicarán más adelante. Además, se muestra información morfológica de cada palabra y su categoría gramatical, con el objetivo de que el usuario pueda identificar con claridad el proceso de formación que ha sufrido cada palabra de la familia. La creación de este grafo se apoyará en una librería de representación gráfica de datos.

La información necesaria para representar el grafo se encuentra almacenada en una base de datos alojada en el servidor web de la División de Lingüística Computacional del IATEXT. Dado que la base de datos no es accesible desde fuera del Servidor ni desde la aplicación web, es necesaria la creación de un servicio web que haga de enlace entre la aplicación web y la base de datos para obtener las palabras de cada familia y el resto de información que se muestra al usuario.

Para identificar las posibles formas canónicas de cada palabra que pertenezcan a familias diferentes, se usará un servicio de lematización de la División de Lingüística Computacional de IATEXT. Este servicio ofrece la información lingüística y morfológica de cada término para detectar todas las posibles opciones y, posteriormente, mostrárselas al usuario para que pueda seleccionar que familia desea ver. Por tanto, un objetivo de este trabajo es analizar y tratar la información que devuelve dicho servicio de lematización.

Por otro lado, es frecuente que este tipo de aplicaciones sean atacadas por bots (programas autónomos que acceden a las webs de internet en busca de información), con el fin de obtener la información de la base de datos. Para combatir esto, se ha propuesto la creación de un módulo que se encargará de encriptar la información que se envía por la red.

Unos objetivos secundarios más personales del alumno relacionados con el TFG han sido:

- Aprendizaje de un nuevo lenguaje de programación, c#.
- Conocimiento y manejo de un nuevo framework de trabajo, ASP.NET y servicios Windows Communication Foundation (WCF)
- Familiarización con un nuevo entorno de desarrollo, Visual Studio 2019
- Aprendizaje de la librería D3.js para la representación gráfica de datos.

Motivación

El tratamiento de textos ha sido mi interés desde hace unos años, debido a que mi padre lleva muchos años investigando en lingüística computacional. Cuando vi este proyecto publicado en la bolsa de oferta de TFG en la página del Centro, hablé con el tutor correspondiente a este trabajo, para conseguir un poco más de información sobre este proyecto. Después de hablar con el tutor, me pareció interesante ya que involucraba el tratamiento textual junto con el diseño gráfico. También, era una oportunidad para aprender un lenguaje de programación y un entorno de desarrollo nuevo que no había estudiado en la carrera.

Justificación de las competencias

Durante la creación de este trabajo de fin de grado se han cubierto competencias específicas del grado de ingeniería informática las cuales detallaremos a continuación.

Formación básica (FB)

- FB04: Conocimientos básicos sobre el uso y programación de los ordenadores, sistemas operativos, bases de datos y programas informáticos con aplicación de la ingeniería.

  Durante la realización de este trabajo se ha hecho uso de distintos lenguajes de programación y aplicaciones. También, mediante llamadas a métodos se han hecho consultas a la base de datos para obtener los datos necesarios para el correcto funcionamiento de la aplicación.

Trabajo de fin de grado

- **TFG01:** Ejercicio original a realizar individualmente y presentar y defender ante un tribunal universitario, consistente en un proyecto en el ámbito de las tecnologías específicas de la Ingeniería en Informática de naturaleza profesional en el que se sinteticen e integren las competencias adquiridas en las enseñanzas.

  Este trabajo ha sido realizado por el alumno Álvaro Rodríguez Martín bajo la supervisión del tutor Francisco Javier Carreras Riudavets. En dicho trabajo se ha aplicado conocimiento adquirido a lo largo de la carrera como algunos lenguajes de programación web para la parte del front-end de la aplicación, y la programación orientada a objetos para la parte del backend.

Común a la ingeniería informática (CII)

- **CII08:** Capacidad para analizar, diseñar, construir y mantener aplicaciones de forma robusta, segura y eficiente, eligiendo el paradigma y los lenguajes de programación más adecuados.

  Antes de la realización del TFT01 se ha realizado un análisis para comprobar que lenguaje y herramientas eran las más adecuadas para la realización del trabajo.

- **CII014:** Conocimiento y aplicación de los principios fundamentales y técnicas básicas de la programación paralela, concurrente, distribuida y de tiempo real.

  Durante la creación de la aplicación se hace uso de métodos síncronos y asíncronos para el intercambio de información entre el servidor y la aplicación cliente.

Ingeniería del software (IS)

- **IS01:** Capacidad para desarrollar, mantener y evaluar servicios y sistemas software que satisfagan todos los requisitos del usuario y se comporten de forma fiable y eficiente, sean asequibles de desarrollar y mantener y cumplan normas de calidad, aplicando las teorías, principios, métodos y prácticas de la ingeniería del software.

  La aplicación web desarrollada cumple con los objetivos planteados antes de realizar este trabajo. Se ha tenido en cuenta el tiempo de respuesta de la aplicación, para que el periodo de tiempo entre la solicitud y la respuesta del servidor sea la menor posible.

- **IS05:** Capacidad de identificar, evaluar y gestionar los riesgos potenciales asociados que pudieran presentarse.

  Como se mencionó en el TFT01 se ha creado una aplicación con tests para comprobar si la aplicación web funciona correctamente.

Sistemas de información (SI)

- **SI01:** Capacidad de integrar soluciones de Tecnologías de la información y las comunicaciones y procesos empresariales para satisfacer las necesidades de información de las organizaciones, permitiéndoles alcanzar sus objetivos de forma efectiva y eficiente, dándoles así ventajas competitivas.

  La aplicación desarrollada, pretende ayudar a personas u organizaciones a tener una mejor educación en lengua española

Tecnologías de la información

- **TI06:** Capacidad de concebir sistemas, aplicaciones y servicios basados en tecnologías de red, incluyendo Internet, web, comercio electrónico, multimedia, servicios interactivos y computación móvil.

  Durante la realización de este trabajo, se ha realizado una aplicación web, uniendo servicios creados para procesar los datos obtenidos de la base de datos con el lado del cliente.

Aportaciones

Con el desarrollo de este trabajo, se ha conseguido realizar aportaciones en el ámbito educativo. Mediante el uso de la aplicación, los usuarios que estén estudiando el idioma español, o aquellas que ya lo dominen, pero estén interesadas en las familias de palabras, podrán observar las relaciones entre las distintas palabras que se encuentran dentro de una familia. Esto puede ser de gran interés, ya que puede ayudar al entendimiento del idioma español y a aprender de donde provienen algunas palabras.

Recursos usados

En esta sección de la memoria, se explican los distintos recursos tecnológicos que hemos utilizado durante el desarrollo de la aplicación web. Se mencionan primero los recursos hardware que han sido necesarios, y luego se explican los softwares que se han usado para el análisis de la aplicación, su posterior desarrollo y la creación de la memoria.

Hardware

En lo referente al hardware necesario para la realización de este trabajo, se ha hecho uso de un ordenador de sobremesa para el análisis y desarrollo de la aplicación y la creación de la memoria. También, debido a la situación de pandemia en la que nos encontramos actualmente, ha sido necesario disponer de unos auriculares, micrófono y cámara web, ya que todas las reuniones con el tutor se han realizado de forma telemática mediante la aplicación Microsoft Teams.

Las características del ordenador de sobremesa utilizado para desarrollar el proyecto son:

- Windows 10 Home
- Procesador Intel(R) Core (TM) i5-9600K CPU @ 3.70GHz 3.70 GHz
- Disco principal SSD de 512Gb
- Disco secundario HDD de 1Tb
- Memoria RAM DDR4 de 16Gb

Con respecto al alojamiento de la aplicación, se encuentra en producción en un servidor Windows Server 2016 con Internet Information Service (IIS).

Librerías

D3.js

Ilustración 1 Logo D3.js[2]

La librería D3.js es una librería de JavaScript usada para manipular documentos basados en datos. La librería hace posible mostrar los datos usando HTML, SVG y CSS. D3 permite unir los datos a un Document Object Model (DOM) y luego aplicarle transformaciones a dicho documento.[3]

Durante la realización de este TFG se ha usado la librería D3.js para la creación del grafo donde se muestran las relaciones entre la palabra introducida por el usuario y su primitiva o derivadas. Para un correcto funcionamiento de la librería, primero de ha de construir un JavaScript Object Notation (JSON), el cual tiene que contener toda la información del grafo y, luego, haciendo uso de los métodos de la librería se podrá crear el grafo con los datos proporcionados en el JSON.

JSON es un formato de archivo de estándar abierto para el intercambio de datos. Está formado por objetos con una clave-valor. Es comúnmente usado en las aplicaciones web para comunicarse con el servidor. [4]

CSS-Element-Queries

La librería CSS-Element-Queries es una librería en JavaScript la cual fue creada y está actualmente mantenida por Marc J. Schmidt. Es una librería basada en las dimensiones de un elemento que permite distinguir cuando se produce un cambio en las dimensiones [5]

[2] Todos los iconos han sido obtenidos de la página web freepng bajo una licencia de uso personal https://www.freepng.es/ Fecha de consulta: 07/06/2021
[3] Librería D3.js https://d3js.org/ Fecha de consulta: 19/05/2021
[4] JSON https://en.wikipedia.org/wiki/JSON Fecha de consulta: 07/06/2021
[5] Librería CSS-Element-Queries http://marcj.github.io/css-element-queries/ Fecha de consulta: 19/05/2021

Está librería es usada en la aplicación web para saber cuándo el usuario ha realizado un cambio en las dimensiones horizontales de la pantalla y entonces poder redibujar el grafo dentro del nuevo espacio disponible.

Net-Core-JS-Encryption-Decryption

La librería Net-Core-JS-Encryption-Decryption es una librería en JavaScript que permite encriptar y desencriptar una string en C# y JavaScript. Esta librería ha sido creada por la compañía Smart In Media GmbH & Co. para solucionar un problema de encriptación cuando estaban realizando un trabajo médico. Luego, donaron dicha librería a la comunidad haciendo todo el código público[6].

En este TFG se ha hecho uso de esta librería para encriptar el JSON creado en el back-end con C# y luego poder desencriptarlo con JavaScript en el front-end para posteriormente, poder crear el grafo.

Font Awesome

Font Awesome es una librería basada en CSS la cual te permite usar iconos ya desarrollados de una manera sencilla.

Esta librería ha sido usada para la creación de algunos iconos y tooltips en la aplicación web.

[6] Librería Net-Core-JS-Encryption-Decryption https://github.com/smartinmedia/Net-Core-JS-Encryption-Decryption Fecha de consulta: 19/05/2021

Bootstrap

Ilustración 2 Logo Bootstrap

Bootstrap es una librería gratis y de código abierto creado para añadirle diseño a las aplicaciones web de una manera sencilla y, a su vez, hacer dichas páginas responsive. La librería cuenta con ficheros HTML, CSS y JavaScript.

Durante la realización de este trabajo, se ha utilizado Bootstrap para la creación del navbar, las cards y los tooltips de la aplicación.

Tecnologías

ASP.NET

ASP.NET es un framework de código abierto, creado por Microsoft para construir aplicaciones web y servicios en .NET. La tecnología .NET es una plataforma de código abierto, también creada por Microsoft, con la que se pueden crear distintos tipos de aplicaciones. Uno de estos tipos de aplicación son las páginas Active Server Page Extended (ASPX), las cuales son usadas para la generación de páginas dinámicas[7].

En este Trabajo de Fin de Grado, se ha utilizado ASP.NET para el desarrollo del proyecto, tanto en el lado del cliente como en el lado del servidor.

Servicios WCF

Los servicios Windows Communication Foundation (WCF) son usados para la creación de aplicaciones orientadas a servicios. Usando los servicios WCF, se puede enviar datos como mensajes asincrónicos desde un endpoint a otro. Estos servicios son comúnmente usados con páginas ASPX. La configuración de trabajo predeterminada para los WCF es en paralelo con ASP.NET.

[7] ASP.NET https://dotnet.microsoft.com/learn/aspnet/what-is-aspnet Fecha de consulta: 04/06/2021

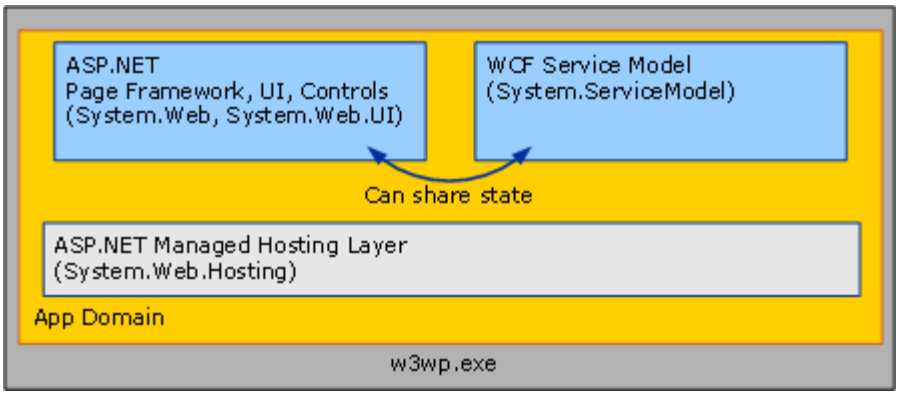

Ilustración 3 Estructura de una aplicación usando ASP.NET y servicios WCF[8]

Durante la realización de este trabajo, se han usado los servicios WCF para obtener los datos necesarios, desde la base de datos, para la creación del grafo.

Base de datos

La base de datos utilizada es el resultado de la tesis doctoral del tutor Don Francisco Javier Carreras Ruidavets. En esta base de datos, se pueden encontrar más de 600 mil palabras. Este conjunto de palabras mediante procesos de derivación, sufijación, prefijación, etc. han dado lugar a más de 62 mil familias de palabras.

En este trabajo se ha utilizado la base de datos mediante los servicios WCF para obtener la información necesaria para la creación del grafo de familias.

Aplicaciones

Microsoft Teams

Ilustración 4 Logo Microsoft Teams

Microsoft Teams es una aplicación de colaboración que permite a los usuarios reunirse en llamadas o reuniones grupales para hablar, compartir pantalla, escribir en el chat y transmitir ficheros.

[8] Servicios WCF https://docs.microsoft.com/es-es/dotnet/framework/wcf/feature-details/wcf-services-and-aspnet Fecha de consulta: 04/06/2021

Como se ha comentado anteriormente, debido a la situación actual de pandemia, todas las reuniones con el tutor se han realizado de forma virtual utilizando la herramienta Microsoft Teams.

1

Visual Studio 2019

Ilustración 5 Logo Visual Studio 2019

Visual Studio 2019 es un entorno de desarrollo integrado para Windows y iOs, desarrollado por la empresa Microsoft. Visual Studio 2019 es compatible con múltiples lenguajes de programación, como C#, Python, Java, JavaScript… La versión 2019 es la más actual.

Durante la realización de este TFG se ha utilizado Visual Studio 2019 para la creación de la página web haciendo uso de C# y JavaScript.

Figma

Ilustración 6 Logo Figma

Figma es una aplicación web que permite generar prototipos para una web o aplicación que se desea crear. Dicha aplicación se centra en la interfaz de usuario y el diseño de experiencia de usuario.

En este trabajo se ha utilizado Figma para la creación de prototipos de la aplicación web, pudiendo tener una primera versión del diseño esperado de la aplicación.

GitHub

Ilustración 7 Logo GitHub

GitHub es una plataforma que permite el alojamiento de proyectos utilizando el sistema de control de versiones Git.

Durante este TFG se ha usado GitHub para ir guardando versiones del proyecto. De esa manera, el tutor ha sido capaz de ir revisando y comprobando el estado del trabajo cuando el alumno subía una nueva versión a GitHub.

StarUML

Ilustración 8 Logo StarUML

StarUML es un programa que permite usar el Unified Modeling Language (UML) para crear diagramas de clases entre otras opciones disponibles.

En esta memoria se ha usado StarUML para crear un diagrama de las clases que se han usado durante el desarrollo de este trabajo y la relación de multiplicidad que existe entre ellas. Se ha decidido usar StarUML entre otros programas, ya que, durante la carrera, en la asignatura de Ingeniería del Software se ha usado dicho programa y por tanto era el más conocido por el alumno.

Navegador Web / Google Chrome

	Navegador	Adquisición
		Usuarios ↓
		8.347.155 % del total: 100,00 % (8.347.155)
☐	1. Chrome	**7.009.989** (83,01 %)
☐	2. Safari	**869.833** (10,30 %)
☐	3. Samsung Internet	**226.615** (2,68 %)
☐	4. Firefox	**123.593** (1,46 %)
☐	5. Edge	**74.515** (0,88 %)
☐	6. Android Webview	**38.470** (0,46 %)
☐	7. Internet Explorer	**37.787** (0,45 %)
☐	8. Opera	**35.199** (0,42 %)
☐	9. GoogleAnalytics	**7.028** (0,08 %)
☐	10. Android Browser	**6.464** (0,08 %)

Ilustración 9 Datos de peticiones

En la ilustración 9, se puede observar el número de peticiones realizado a las aplicaciones del IATEXT durante el periodo de un año, y los navegadores que se usaron para realizar dichas peticiones. Estos datos han sido proporcionados por el tutor mediante el uso de la herramienta Google Analytics.

Como se puede ver, Google Chrome es usado por un 83% de los usuarios que consultan las aplicaciones del instituto. Debido a ello, en nuestro trabajo, nos hemos centrado en usar el navegador Chrome cuando se debía realizar alguna comprobación de la estética y funcionamiento de la aplicación.

Ilustración 10 Logo Google Chrome

Google Chrome es un navegador web de código cerrado desarrollado por Google. Cuenta con más de 900 millones de usuarios, siendo considerado el navegador

web más usado en muchos países. Actualmente, el navegador está disponible para Windows, macOS, Linux, Android y iOS.[9]

Lenguajes

C#

Ilustración 11 Logo C#

C# es un lenguaje de programación desarrollado por la empresa Microsoft para su plataforma .NET. Al igual que algunos otros lenguajes, C# es un lenguaje orientado a objetos (OOP). Actualmente, C# no es solo usado en .NET, sino que se ha ido expandiendo a otras plataformas como Unity para la creación de videojuegos.

Durante la realización de este TFG, se ha usado C# en el back-end de la aplicación para conectar con los servicios WCF y obtener los datos necesarios según la petición del usuario.

HTML5

Ilustración 12 Logo HTML5

HTML5 es un lenguaje de marcado usado en la World Wide Web. Las siglas significan *HyperText Markup Language,* siendo esta la quinta versión de dicho lenguaje.

[9] Google Chrome https://es.wikipedia.org/wiki/Google_Chrome Fecha de consulta: 05/06/2021

En este TFG, se ha utilizado HTML5 para la creación de la estructura de la aplicación web.

CSS y SCSS

Ilustración 13 Logo CSS3

Cascading Style Sheets (CSS) es un lenguaje de diseño usado para proporcionar al HTML un diseño visual deseado. Sassy CSS (SCSS) es un lenguaje de scripts que luego es compilado a CSS

Durante la creación de la aplicación web se ha usado SCSS, posteriormente compilado a CSS, para darle el estilo anteriormente diseñado a dicha aplicación.

JavaScript

Ilustración 14 Logo JavaScript

JavaScript (JS) es un lenguaje de programación interpretado. Al principio era usado solamente para el lado cliente de la página web, proporcionando mejoras en la interfaz de usuario y modificando las páginas de una forma dinámica. Durante los últimos años, JS ha ido creciendo y expandiéndose a otros sectores, como el back-end de las aplicaciones web con Node.js o a la creación de videojuegos con Unity. Al igual que C#, JavaScript es un lenguaje OOP.

En este TFG se ha usado JavaScript para cambiar elementos de la aplicación web dependiendo de las acciones que realice el usuario. También, se han utilizado tres librerías en JavaScript para ayudar con el funcionamiento de la aplicación.

Resumen

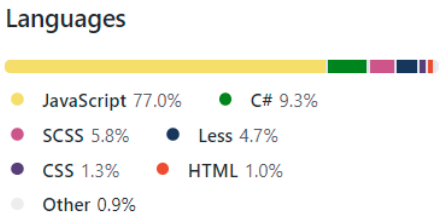

Ilustración 15 Análisis de lenguajes usado en GitHub

Como se puede observar en la ilustración 15, la cual ha sido obtenida por el repositorio de Github donde se ha subido el trabajo, se puede observar cómo JavaScript corresponde el 77% del lenguaje usado, frente al 9,3% de C#. Esto es debido a las tres librerías de JavaScript que se han importado, D3.js, Net-Core-JS-Encryption-Decryption y CSS-Element-Queries. El código de dichas librerías es calculado a la hora de obtener el porcentaje final de los lenguajes usados. Si las librerías no fueran tenidas en cuenta, obtendríamos que el código en JavaScript y C# tendrían un porcentaje mucho más similar.

Seguidamente, se puede encontrar SCSS y CSS que corresponden al diseño que se le ha dado a la aplicación web en SCSS y luego compilado a CSS. Parte de este código en SCSS y CSS junto con el código Less es proporcionado por la librería Font Awesome y Bootstrap, desde donde se han obtenido algunos iconos y CSS previamente ya creado.

Luego, se puede observar el código en HTML. Dicho código solo corresponde al 1% del proyecto, ya que mucho HTML es creado dinámicamente mediante código C# o por JavaScript.

Finalmente, en la parte de Other, podemos encontrar, entre otros, ficheros XML los cuales se crean automáticamente cuando se inicia el proyecto para la configuración de dicho trabajo y posterior subida a un servidor, y ficheros TXT que vienen con las librerías explicando las licencias proporcionadas por dichas librerías.

Análisis

Requerimiento de usuario

NO esta claro

Requerimiento de software

De entre los posibles lenguajes con los que se podría realizar la aplicación, el tutor sugiere que se realice la aplicación en C# ya que formará parte de un grupo de aplicaciones de lingüística, las cuales ya han sido desarrolladas en C#. También, se tuvo en cuenta que, al finalizar el trabajo de fin de grado, la aplicación será mantenida por el grupo informático del IATEXT, al que pertenece el tutor.

Una vez estudiado por el alumno este nuevo lenguaje de programación, se ha podido observar que es bastante similar a Java, por lo que no fue muy difícil entender los conceptos básicos de C#, y posteriormente, durante la realización del trabajo, seguir aprendiendo en más profundidad.

Posicionamiento SEO

El posicionamiento SEO, por sus siglas en inglés Search Engine Optimization, es un conjunto de acciones con el objetivo de mejorar el posicionamiento de un sitios web en la lista de resultados de los buscadores web[10].

Se ha decidido que el nombre de la aplicación web en español, debería contener familia y palabras. Para obtener un mejor posicionamiento SEO, se ha utilizado la herramienta Google Trends[11] para decidir qué palabras o combinación de palabras relacionadas con la aplicación son las más adecuadas para mejora el SEO. Se han realizado combinaciones con varias palabras, y se han obtenido cuatro posibilidades distintas: familia de palabra, familia de palabras, familias de palabras y familias de palabra. Con estas cuatro combinaciones, se ha podido obtener la gráfica que se puede ver a continuación.

[10] Posicionamiento SEO https://es.wikipedia.org/wiki/Posicionamiento_en_buscadores Fecha de consulta: 03/06/2021
[11] Google Trends https://trends.google.com/ Fecha de consulta: 03/06/2021

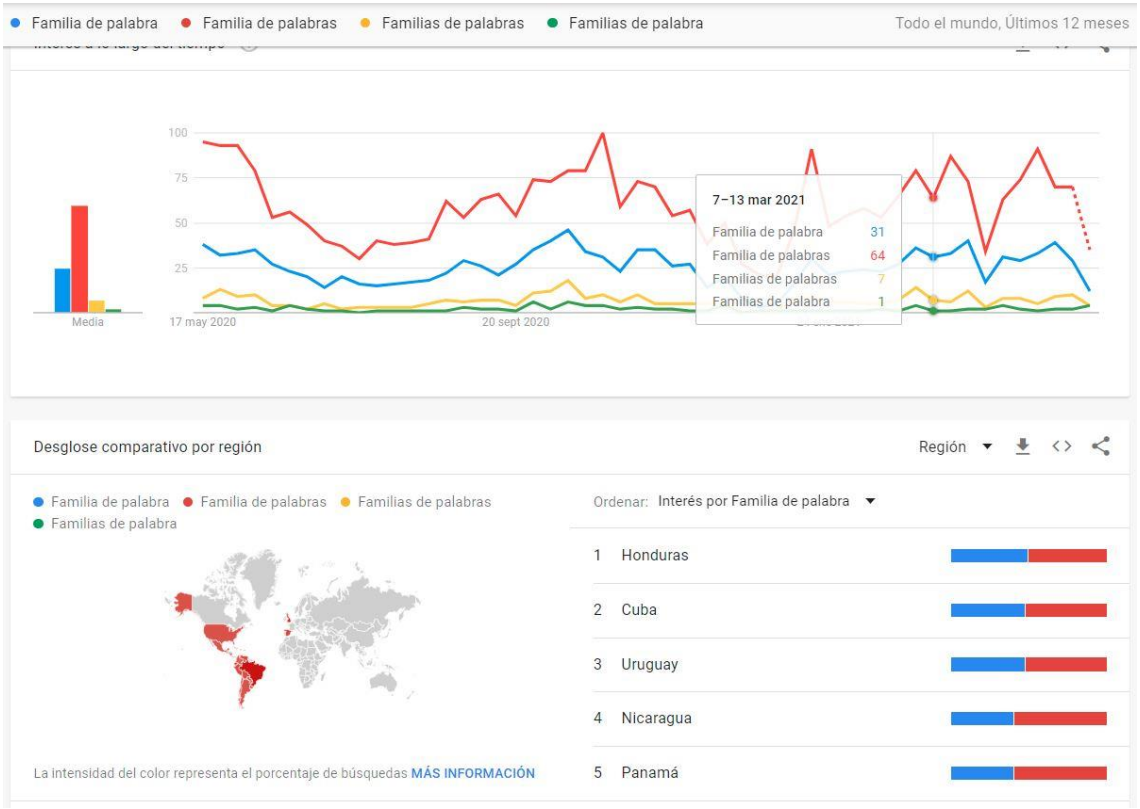

Ilustración 16 Google trends para url en español

Como se ha podido observar, el conjunto de palabras más usado es “Familia de palabras”. Por este motivo, se ha elegido poner este nombre como url de la aplicación web.

Así mismo, se ha vuelto a usar la herramienta para decidir el nombre de la aplicación en su versión inglesa. En este caso, se han traducido los cuatro conjuntos previos, obteniendo las siguientes opciones: families of words, families of word, family of words y family of word. Con estos conjuntos, estas son las gráficas obtenidas.

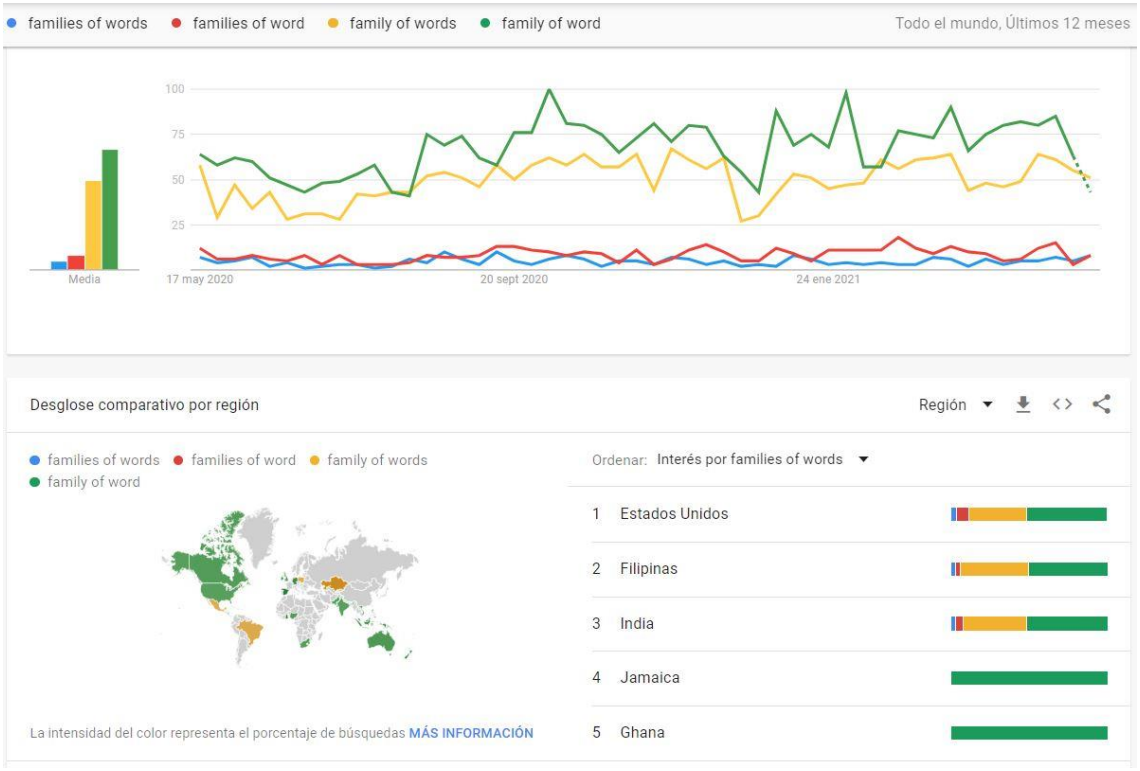

Ilustración 17 Google Trends para url en inglés

Se observa que el conjunto family of words y family of word, tienen una cantidad de uso muy similar. Para decidir el nombre final, se ha comprobado el mapamundi, donde se pueden ver las diferentes regiones donde cada conjunto de palabras es más usado. Se puede ver que el color verde, predomina en los países de habla inglesa por lo que, finalmente, se ha decidido que el nombre de la aplicación web en inglés sea “Family of word”.

Diseño

Diseño gráfico para ordenador

Como ha sido mencionado anteriormente, se ha usado la aplicación web Figma para crear unos prototipos de la aplicación web. Como se puede observar en la Figura vista a continuación, se ha intentado mantener un aspecto sencillo para facilitar al usuario la entrada de los datos.

Ilustración 18 Prototipo en Figma

Una vez que el usuario ha introducido la palabra, se le muestran unos botones que contienen las formas canónicas de dicha palabra, siempre que la forma canónica tenga familia. El usuario podrá hacer click en el botón deseado para ver un grafo u otro. Al mismo tiempo, justo debajo, se formará el grafo correspondiente al primer botón.

Ilustración 19 Prototipo en Figma

Una vez que el grafo ha sido formado, el usuario puede interactuar con él, pudiendo hacer zoom y mover el grafo dentro del espacio establecido. También, podrá poner el ratón encima de un nodo para desplegar un mensaje emergente, donde podrá encontrar más información de dicho nodo. Si el usuario desea ver información sobre los colores del grafo, podrá hacer click en el '+' para desplegar una ventana, donde se le mostrará una leyenda.

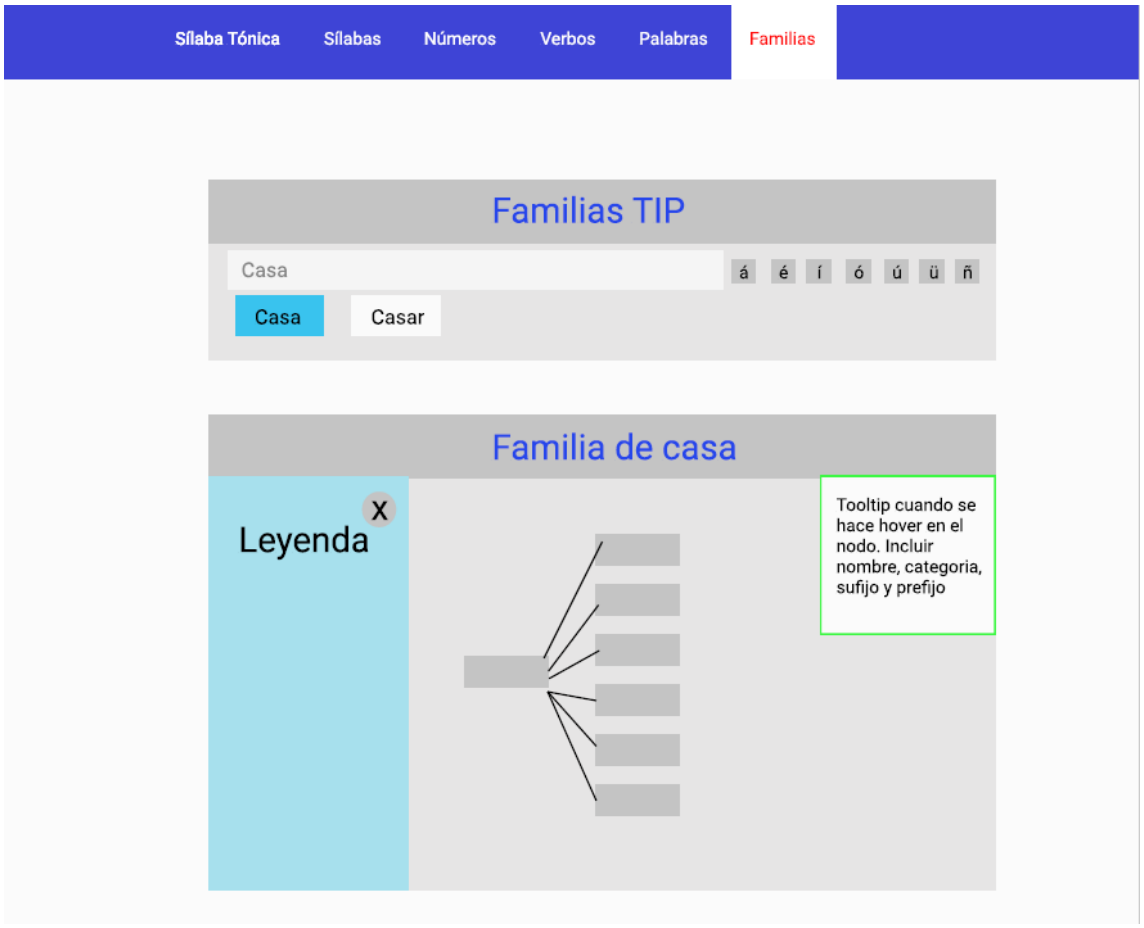

Ilustración 20 Prototipo en Figma

A continuación, se puede ver una captura de la aplicación final.

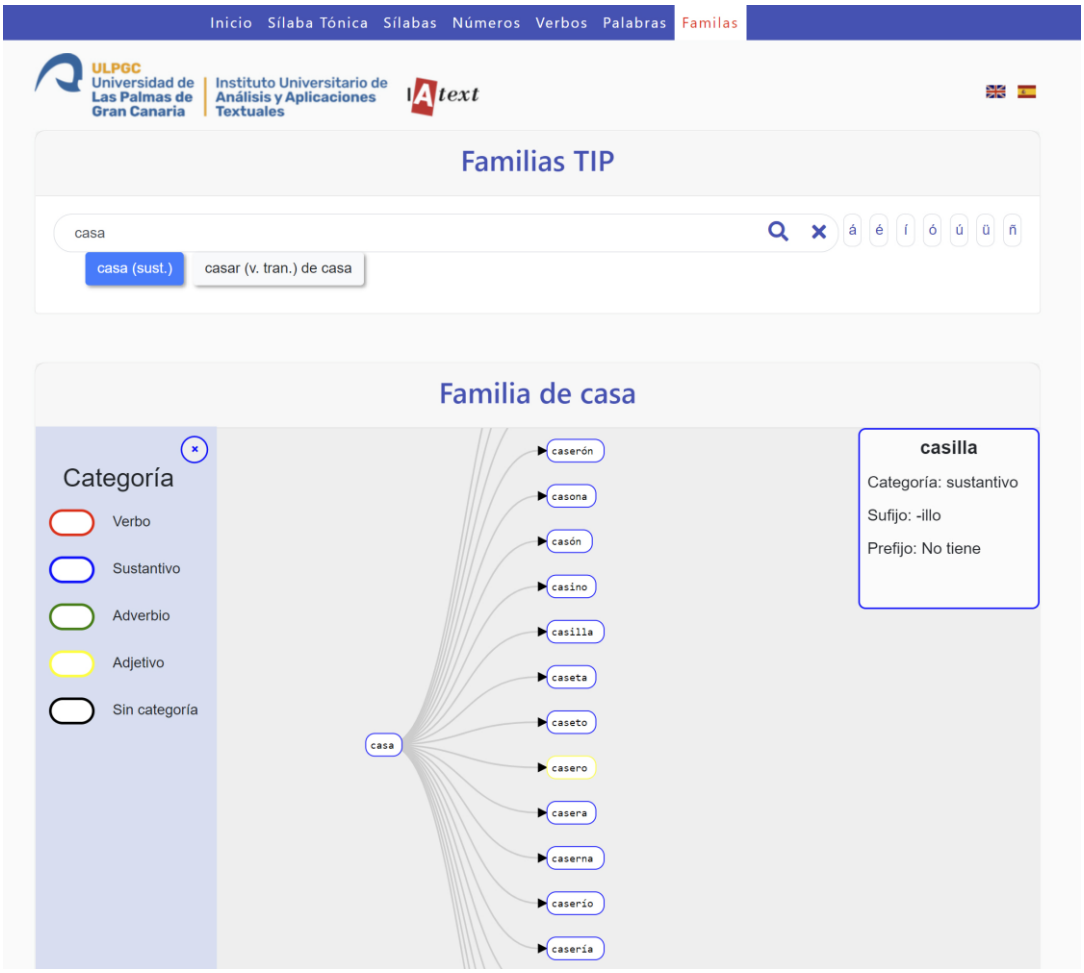

Ilustración 21 Versión final para ordenador

Diseño gráfico para dispositivos móviles

Con respecto al diseño para dispositivos móviles, se ha querido mantener la sencillez creada en la versión de ordenador. Para ello, se han agrupado los links del menú dentro de un desplegable y se ha mantenido una vista muy similar a la de ordenador donde el usuario podrá introducir la palabra.

Ilustración 22 Prototipo Figma para móvil

Una vez que el usuario ha introducido la palabra, al igual que en la versión de ordenador, se mostrarán los botones donde el usuario podrá seleccionar qué grafo desea ver.

Ilustración 23 Prototipo en Figma para móvil

Finalmente, se puede observar la aplicación finalizada en su versión de móvil.

Ilustración 24 Versión final en móvil

Diagrama de clases

En esta sección de la memoria se mostrará, de una forma breve, las clases que se han utilizado durante el desarrollo del trabajo. Posteriormente, en la sección de desarrollo se explicará todo con más detalle.

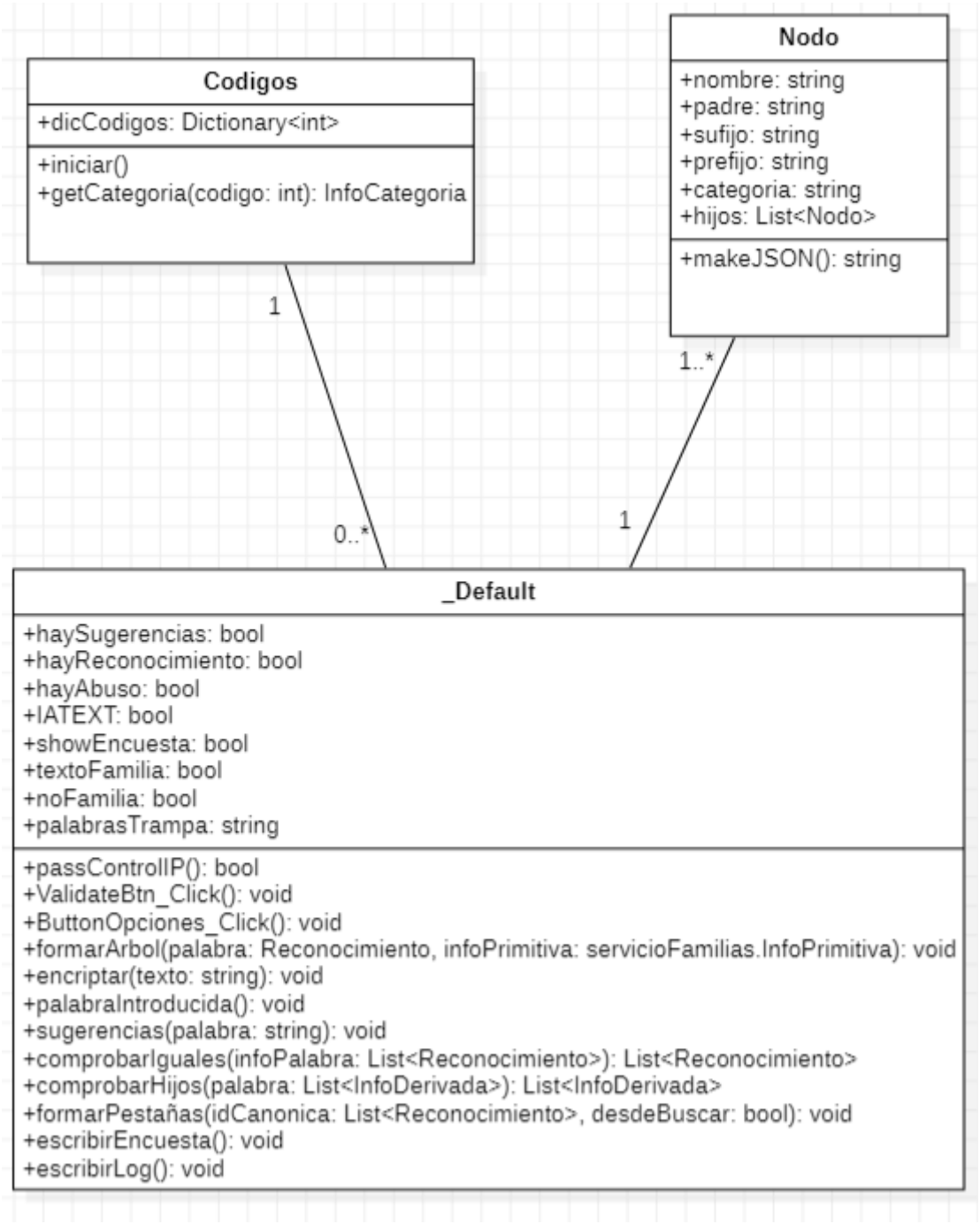

Ilustración 25 Diagrama de clases

Como se puede observar en el diagrama de clases, este Trabajo de Fin de grado, es una aplicación web formada por tres clases. Primero, encontramos la clase Codigos, la cual es una clase estática donde se cargarán todos los códigos de las categorías. Al ser una clase estática, solo existe una instancia de dicha clase en el servidor y es compartida por todos los usuarios que usen la aplicación.

Seguidamente, se puede ver la clase Nodo, la cual contiene la información necesaria para formar los nodos del grafo y posteriormente formar la string con el JSON.

Finalmente, se puede observar la clase _Default. Esta clase viene creada por defecto cuando se crea una aplicación web en ASP.NET. Aquí se tratan todos los procesos del servidor que se mostrarán en la aplicación web para que el usuario pueda interactuar con ella.

Desarrollo

En este apartado se tratará sobre los pasos realizados durante la creación de este Trabajo de Fin de Grado para, finalmente, obtener la aplicación web en funcionamiento.

Primero se mencionará como se construyó el grafo usando datos no dinámicos. Seguidamente, se modificará un servicio WCF necesario para obtener dichos datos de manera dinámica, dependiendo de la petición que el usuario realice. Luego realizaremos la parte del backend, usando el servicio recientemente modificado para obtener el JSON que el grafo necesita para ser formado. Finalmente, crearemos una aplicación de tests, los cuales comprobaran que el JSON formado en la aplicación web es el correcto.

Creación del grafo con datos de prueba

En esta primera parte del desarrollo, se ha estudiado la librería D3.js y luego sea creado la estructura del grafo con unos datos ya precargados para comprobar la correcta formación de dicho grafo. Finalmente, se añadirán métodos previamente discutidos con el tutor, como añadir la opción de esconder los hijos, poder hacer zoom en el grafo, poder mover el grafo dentro del div, etc, para mejorar y hacer más interactivo el grafo.

Primeramente, como se mencionó anteriormente, se comenzará con el estudio de la librería D3.js. Para ello, se ha accedido a la documentación de la librería, la cual se puede encontrar en este link. También, se han seguido algunos tutoriales en YouTube, donde se explicaba la creación de un grafo básico y algunos grafos ya creados y con código abierto, como el grafo creado por d3noob.

Lo primero que se ha creado para comenzar la creación del grafo, es el JSON desde el cual la librería crea el grafo. En dicho JSON, cada objeto tiene como mínimo un campo con el nombre del nodo, y si dicho nodo tiene hijos entonces, tiene otro campo children, el cual contiene un array con los nodos que son hijos de ese nodo. A continuación, se puede ver una imagen con un JSON básico

```
treeData = {
    "name": "Padre",
    "children": [
        {
            "name": "Hijo 1",
            "children": [
                {
                    "name": "Nieto 1",
                }
            ]
        }
    ]
};
```

Ilustración 26 Formato del JSON

Una vez establecido el JSON se ha procedido a la creación de los métodos que convertirán el JSON en un grafo. Para ello, como se comentó anteriormente, se ha usado un video de YouTube[12] donde explican la creación del grafo y los métodos usados para

[12] D3 Interactive Tree Graph: Part 3 https://www.youtube.com/watch?v=-3WdDnuHdkQ Fecha de consulta: 01/06/2021

ello. Una vez entendido como funciona la creación del grafo, se ha procedido a modificarlo para obtener un grafo que sirviera para representar las familias de palabras. A continuación, se muestra un grafo básico en el que se pueden ver los tres niveles. También, se han añadido unas flechas al final de los link para indicar que el hijo depende del padre y, seguidamente, el nieto depende del

Ilustración 27 Grafo base

Luego, como se comentó al comienzo de este apartado, se ha creado un método el cual permite esconder o mostrar nodos hijos. En las siguientes imágenes se puede ver como primero solo se ve el padre y los dos hijos, pero, después de hacer click sobre "Hijo 1" aparecen los hijos de ese nodo.

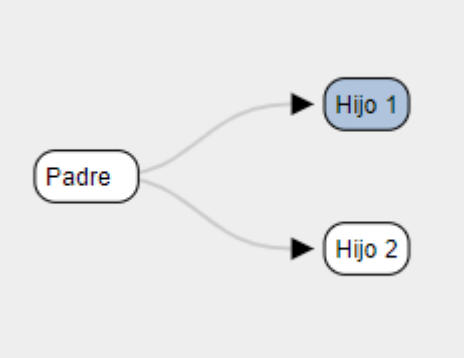

Ilustración 28 Grafo con hijos sin mostrar

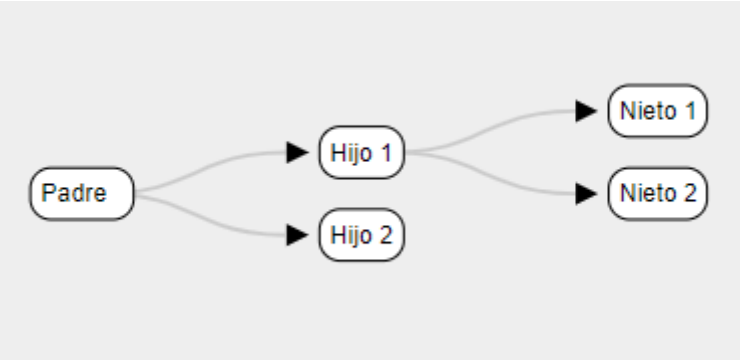

Ilustración 29 Grafo con hijos mostrados

A continuación, se ha añadido un campo categoría en el JSON del árbol. Dicho campo contendrá la categoría de la palabra y dependiendo de ello el borde del nodo cambiará de color. Como se puede ver en la siguiente figura, existen tres colores de bordes distintos, los cuales corresponden a sustantivos, adverbios y verbos. Actualmente, es solo una prueba para comprobar que funciona de forma correcta y, posteriormente, se añadirán el resto de las categorías.

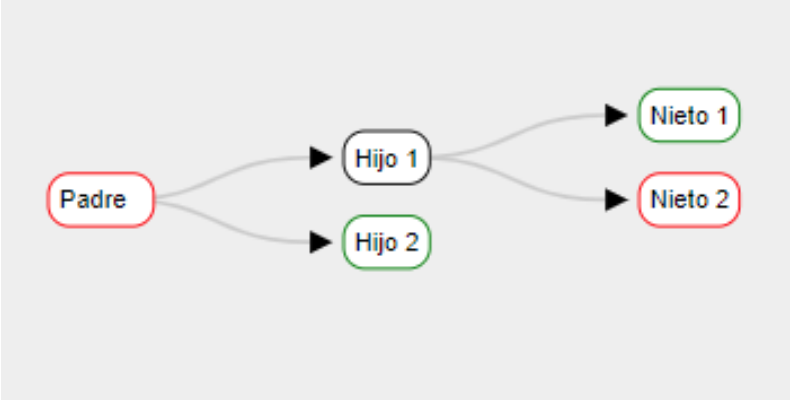

Ilustración 30 Grafo mostrando la categoría

Luego, se ha añadido un tooltip que muestra la información del nodo. La versión de la librería que se ha usado para crear al grafo no tiene incluido la creación de tooltips, por lo que se ha tenido que añadir otro script para crearlos. Dichos tooltips aparecerán cuando el usuario pase el ratón por encima del nodo y se mostrará los campos que se encuentre dentro del JSON. Debido a que actualmente solo está el campo del nombre y la categoría eso es lo que se muestra en el tooltip como se puede ver en la figura.

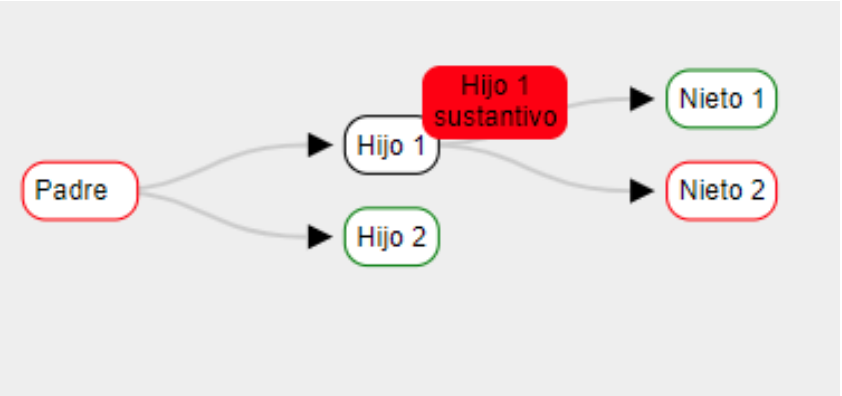

Ilustración 31 Grafo mostrando el tooltip creado

Seguidamente, se ha intentado añadir el método zoom, para acercar y alejar el grafo y poder moverlo dentro del espacio establecido. Para ello, D3 tiene un método llamado “behavior.zoom()”, pero debido a la versión que se estaba usando hasta este momento, la versión 3.5.17, no permitía añadir zoom al grafo. Para añadir el método zoom, se ha tenido que cambiar a la versión 3 básica. Esto ha significado que algunos métodos que se usaban antes para la creación del grafo no funcionaban en esta nueva versión y se han tenido que modificar partes del código desarrollado, ya que en esta nueva versión no podían ser utilizadas.

Creación del JSON con datos de prueba

Una vez terminada la creación del grafo con un JSON creado directamente en el fichero de JavaScript, se debe excluir dicho JSON del fichero para obtenerlo desde C#. En esta sección se explicará cómo se realizó una aplicación de prueba en la cual se creaban los datos en el backend con C# y luego se obtenía el JSON en el fichero JavaScript para formar el grafo.

Para empezar, se ha creado una aplicación separada de la anterior para realizar cambios e ir comparando con lo que teníamos en la aplicación de familias. En esta nueva aplicación web, se ha creado un botón el cual al ser pulsado deberá obtener los datos de prueba del servidor y, con ellos, crear el grafo en el lado cliente.

Primero, se ha estudiado como ejecutar código JavaScript desde C#, y se ha visto que la forma más común es usando el método ScriptManager. De esta manera la función de JavaScript pasada por parámetro en el ScriptManager será ejecutada cuando se termine de ejecutar todo el código en C#. Con ello se consigue obtener el JSON necesario y luego mostrar el grafo.

Con respecto a la creación del JSON necesario, Microsoft recomienda el uso del framework Json.Net, también llamado Newtonsoft. Como se puede observar en la siguiente figura, esta recomendación es debido a los bajos tiempos de creación que tiene el utilizar este framework respecto a otros.

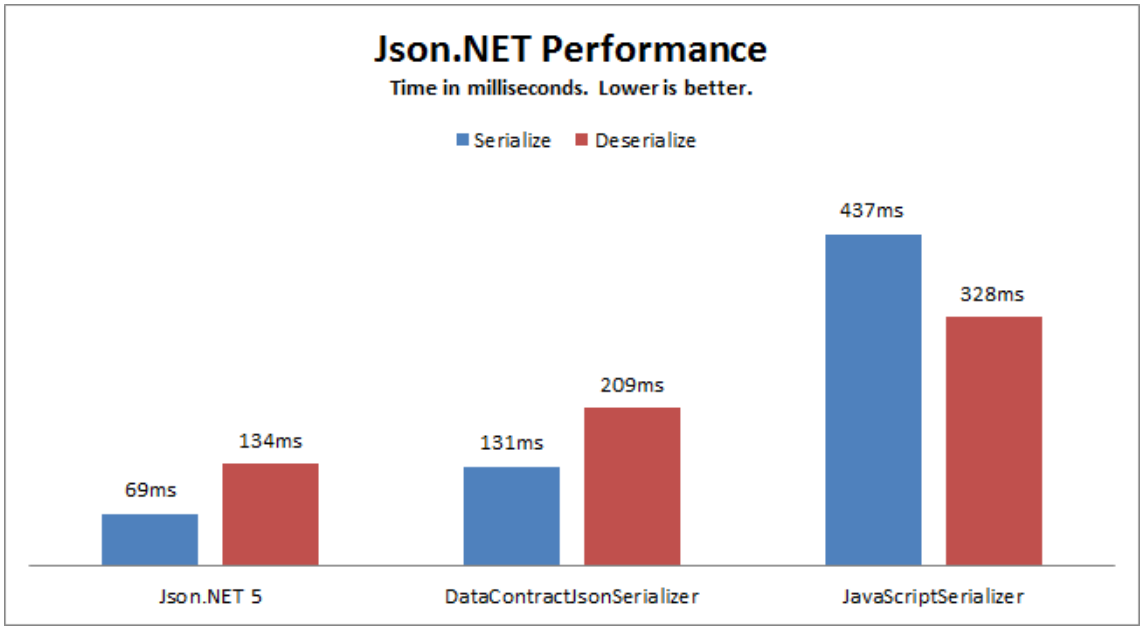

Ilustración 32 Comparativa entre Json.Net y otros métodos [13]

[13] Jsno.Net https://www.newtonsoft.com/json Fecha de consulta: 07/06/2021

Una vez decidido cómo crear el JSON, se procedió a probar con unos datos de prueba, pero al pasar la información al fichero JavaScript, se pudo observar que no sería posible usar Netwosoft, debido a que es necesario guardar los datos en una variable HiddenField, para luego poder obtener dichos datos con JavaScript. Como se puede observar en la documentación del HiddenField[14], el atributo Value solo admite una string, por lo que no es posible guardar en dicha variable un JSON. Para resolver dicho problema, se ha decidido por no usar el framework Newtonsoft y realizar un JSON en formato string y, luego, en JavaScript transformar dicha string a JSON.

Con el fin de obtener una string que tenga forma de JSON, se ha decidido crear una nueva clase llamada Nodo. Esta clase tendrá por parámetros la información que deseamos que el JSON contenga. Para realizar esta prueba los atributos serán: una string con el nombre del nodo, una string con la categoría de la palabra y un array de objetos de la clase Nodo, donde encontraremos todos los hijos de este nodo. En lo referente a los métodos, contendrá un contructor y un método llamado "makeJSON()" el cual, de forma recursiva, devuelve una string con formato JSON del nodo deseado. De esta manera, si el ejecutar el método "makeJSON()" del primer nodo, se obtendrá una string con toda la información necesaria para formar el grafo.

Una vez la string ha sido formada, se guarda el valor en el atributo value de la variable HiddenField previamente declarada. Luego, en el fichero JavaScript debemos acceder a dicha variable HiddenField en el HTML de la página mediante su id, y recuperar la string.

Finalmente, para obtener el objeto JSON necesario para la creación del grafo, esta string es pasada por parámetro al método JSON.parse() el cual devuelve un objeto tipo JSON de la string pasada por parámetro.

[14] Documentación HiddenField https://docs.microsoft.com/en-us/dotnet/api/system.web.ui.webcontrols.hiddenfield.value?view=netframework-4.8 Fecha de consulta: 02/06/2021

Modificación del servicio WCF

En esta sección del desarrollo, se explicará el procedimiento realizado para modificar el servicio WCF proporcionado por el tutor. Desde este servicio, se obtienen los datos necesarios para la formación del grafo. Para realizar esta modificación, se ha tenido que estudiar el funcionamiento de los servicios WCF, ya que es una tecnología nueva que el alumno no había usado anteriormente.

Una vez estudiado el servicio de familias proporcionado por el tutor, se ha procedido a añadir una nueva funcionalidad a este servicio. Esta nueva funcionalidad permite recuperar un conjunto de datos que permiten identificar las familias a las que pertenece la palabra dada.

Introducción de la palabra por el usuario

En esta sección, se explicarán los pasos que se han seguido cuando el usuario introduce una palabra y desea obtener el grafo de dicha palabra.

Primeramente, se ha creado en el HTML un asp: TextBox, donde el usuario podrá introducir la palabra deseada y un botón para enviar dicha palabra. También, se ha añadido un validador, haciendo uso de la "RegularExpressionValidator Class"[15], para comprobar que el valor introducido es una palabra y no contiene ningún carácter especial o número.

Una vez el usuario hace click para enviar la palabra, se ejecuta el método "ValidateBtn_Click" el cual comprueba si el valor introducido es válido con respecto a la expresión regular introducida previamente en el HTML. Si el valor no es válido, se le mostrará una alerta al usuario indicando que debe introducir una palabra válida para usar la aplicación. En caso contrario, si la palabra es válida, se ejecuta el método palabraIntroducida.

El método palabraIntroducida, recibirá la palabra suministrada por el usuario y, dependiendo de si esta existe y si tiene familia o no, se ejecutarán otros métodos. Primero, mediante el servicio de lematización, se obtendrá todos los posibles reconocimientos de la palabra dada. Este servicio, permite obtener el lema correspondiente de la palabra dada. El lema de una palabra es la forma de dicha palabra que se acepta como representante de las formas flexionadas de la palabra en cuestión.[16]

Este paso es necesario ya que en el lenguaje español existen palabras homógrafas, las cuales tienen una grafía idéntica, pero tienen un significado diferente. Por ejemplo, la palabra casa puede significar un lugar para habitar, pero también puede ser la conjugación del verbo casar en tercera persona del singular del presente de indicativo (él casa).

Seguidamente, se comprueba dicha lista de nuevas palabras, si la lista está vacía, significa que la palabra buscada no se encuentra en la base de datos, por lo que se le muestra un aviso al usuario. En caso contrario, se debe recorrer dicha lista y eliminar las palabras repetidas y las que no tienen familia, ya que para mostrar un grafo la palabra necesita tener una familia. Se considera que una palabra es igual cuando su forma canónica y el id de su categoría son iguales. Para realizar este tratamiento, se ejecuta el

[15] RegularExpressionValidator Class https://docs.microsoft.com/en-us/dotnet/api/system.web.ui.webcontrols.regularexpressionvalidator?view=netframework-4.8 Fecha de consulta: 02/06/2021
[16] Lematización https://es.wikipedia.org/wiki/Lematizaci%C3%B3n Fecha de consulta: 07/06/2021

método “comprobarIguales”, el cual devuelve una nueva lista con las palabras que han pasado la comprobación mencionada anteriormente.

Una vez se obtiene la nueva lista del método “comprobarIguales”, se comprueba si dicha lista está vacía. Una lista vacía significa que la palabra introducida por el usuario no tiene ninguna familia, por lo que se le mostrará un aviso al usuario. En caso contrario, se ejecutará el método “formarPestañas”, el cual se explicará en la siguiente sección y, finalmente, se mostrará el grafo de la primera palabra de la lista. Para ello, se ejecutará el método “formarArbol” el cual también se explicará en una sección posterior.

Creación de pestañas

En esta sección se explicará a que se refiere con pestañas y como se crean esas pestañas usando el método “formarPestañas” mencionado anteriormente.

Se ha decidido el uso de la palabra pestañas para señalar los botones que aparecerán en la aplicación cuando el usuario haya buscado una palabra y dicha palabra tenga más de una forma canónica. Dichas pestañas podrán ser pulsadas para seleccionar que grafo se desea visualizar. Se ha decidido usar la palabra pestaña, ya que su funcionamiento y apariencia es similar a las pestañas de un navegador, dependiendo en que pestaña nos encontremos, se podrá observar un grafo u otro.

Ilustración 33 Pestañas cuando se busca la palabra “casa”

Con respecto al método “formarPestañas”, este recibe por parámetro una lista con las palabras obtenidas anteriormente en “palabraIntroducida”. También, recibe un booleano para saber si dicho método se ha llamado desde “palabraIntroducida”.

La primera comprobación que se realiza dentro de este método es comprobar si la lista solo tiene una palabra y si dicha palabra solo tiene 1 familia. En caso de que esta comprobación sea cierta, el método se termina y no se genera ninguna pestaña. Esto es debido a que, durante una reunión con el tutor, se consideró que si una palabra solo tiene un grafo no se deben mostrar pestañas, ya que solo se mostraría una pestaña sin posibilidad de cambiar. Por ello, al terminar el método directamente, se generará el primer grafo desde el método “palabraIntroducida” y no se creará ninguna pestaña.

En caso contrario, si la lista tiene más de una palabra o dicha palabra tiene más de una familia, se procede a la creación de las pestañas. Para ello, se recorre la lista de palabras pasada por parámetro. Si dicha palabra tiene más de una familia, también se debe recorrer dicha lista de familias, ya que los grafos resultantes serán distinto. Si el grafo resultante es de tres niveles (padre, hijo y nieto), la pestaña incluirá el nombre de

la palabra primitiva, como se puede ver en la figura anterior en la segunda pestaña. En caso contrario, la pestaña solo contendrá la forma canónica de la palabra y su categoría gramatical.

Para acceder a la categoría gramatical de una palabra, se ha creado la clase estática llamada Códigos. Al ser una clase estática, solo existe una instancia de ella, por tanto, dicha instancia será compartida por todos los usuarios que accedan a la aplicación. De esta manera, se consigue que la aplicación web sea más eficiente, evitando que cada acceso a la aplicación cree una nueva instancia de la clase Códigos. Esta clase tiene un atributo de tipo Dictionary llamado dicCódigos, el cual contiene una clave int y un value tipo InfoCategoria. Con respecto a los métodos, cuenta con un método iniciar y un método getCategoria. El método iniciar se ejecuta al iniciar la aplicación web en el servidor y sirve para cargar en el atributo todas las claves y valores. Posteriormente, el método getCategorias será usado por la aplicación web cuando se tenga que consultar el valor al cual corresponde la clave que se le pasa por parámetro, siendo esta clave un int.

Creación del JSON

Para la creación del JSON, como fue mencionado anteriormente, se ha usado la clase Nodo que se probó en la aplicación de prueba. Una vez copiada la clase a la aplicación final, se han añadido los atributos: padre, sufijo y prefijo. Consecuentemente, se ha modificado el método "makeJSON()" para que incluya estos nuevos atributos. También se ha añadido un parámetro extra al constructor llamado "cultura".

En la sección de posicionamiento SEO se puede observar que la aplicación es posible visualizarla en dos idiomas, español e inglés. Dependiendo de la url que se está accediendo, se puede saber en qué idioma se encuentra la página accediendo al objeto CultureInfo.Name[17], el cual ha sido creado cuando el usuario entra en la aplicación web.

Una vez modificada la clase, se ha procedido a la creación del método "formarArbol". Este método se encarga de formar los objetos de la clase Nodo para luego poder crear la string con formato JSON. Para ello, el método obtiene por parámetro la palabra buscada por el usuario y su primitiva.

Al comenzar, se usa el servicio de familias para obtener las derivadas de la palabra buscada y de su primitiva. Una vez obtenidas todas las palabras, se procede a la creación de los objetos tipo Nodo, comenzando desde los nietos, luego los hijos y finalmente la palabra primitiva. Esto es debido a que el Nodo de la palabra primitiva y el Nodo de la palabra buscada por el usuario tienen un array de Nodos, por lo que dichos Nodos que están dentro del array tienen que haber sido creados previamente. Una vez obtenido el objeto Nodo de la raíz del grafo, se procede a convertir el conjunto de Nodos a una string con formato JSON con el método "makeJSON()".

Finalmente, se ejecuta un proceso de encriptación sobre la string creada y posteriormente ejecutar el código en JavaScript para mostrar el grafo. Este proceso de encriptado se verá en la siguiente sección.

[17] CultureInfo.Name
https://docs.microsoft.com/en-us/dotnet/api/system.globalization.cultureinfo.name?view=net-5.0
Fecha de consulta: 03/06/2021

Encriptación del JSON

En este apartado, se explicará el método “encriptar”, los problemas que hubo a la hora de realizarlo y la solución encontrada.

Una vez terminada la aplicación y con el fin de dificultar el robo de información, se implementó la encriptación de la string que contiene el JSON en el lado servidor y la desencriptación justo antes de la creación del grafo en el lado cliente. El primer problema que se encontró fue el uso de dos lenguajes de programación distintos ya que, primero, se ha de encriptar la string en el lado servidor usando C# y, luego, en el lado cliente se ha de desencriptar usando JavaScript. Como Martin Weihrauch explica[18], crear y mandar IV / SALTS las cuales no sean estáticas entre distintos lenguajes puede ser bastante complicado. Por ello, como fue mencionado en los recursos, su empresa ha creado una librería donde se puede encriptar y desencriptar en C# y JavaScript, lo cual es justamente lo que se necesita para este trabajo.

De esta manera, el método “encriptar” se encarga de encriptar la string formada previamente y obtenida por parámetros. A continuación, la nueva string encriptada es enviada al HTML. Finalmente, desde el fichero JavaScript se obtiene la string encriptada y con el uso de la librería se desencripta para obtener el JSON para la creación del grafo.

[18] Cita de Martin Weihrauch
https://stackoverflow.com/questions/39505636/encryption-in-c-sharp-decryption-in-js-cryptojs
Fecha de consulta: 03/06/2021

Aplicación de tests

Para comprobar el correcto funcionamiento de la aplicación web, se ha decidido crear una aplicación de test. Debido a que la aplicación que se ha creado tiene una salida gráfica, es muy difícil realizar un test sobre ello, ya que se necesitaría un humano para ir viendo que el grafo creado es el correcto. Por ello, se ha decidido dar un paso atrás en el desarrollo, y comprobar que el JSON formado, el cual luego crea el grafo, es el correcto.

Para comenzar, se ha copiado el código de la aplicación web con algunas modificaciones con el fin de poder obtener todos los JSON posible a partir de una palabra leída de un fichero txt. Dichos JSON serán guardados en un fichero txt para posteriormente ser usado en la comprobación.

Una vez terminada la creación de los JSON, se ha procedido a la comprobación de su validez. Debido a la poca información que se guarda en el JSON (nombre, padre, categoría, prefijo y sufijo), no se puede saber al 100% a que palabra se refiere ya que, como se explicó anteriormente, en el lenguaje español existen palabras homógrafas, las cuales no se podrían distinguir con esta información.

Empezando desde el root del JSON, se han buscado las flexiones de dicha palabra y, posteriormente con la lista obtenida, se ha ido recorriendo la lista hasta que se encontrara una palabra que tuviera el mismo número de derivadas que las encontradas en el JSON. Una vez encontrada dicha palabra, se ha procedido a comprobar que las derivadas obtenidas por el servicio coinciden con las escritas en el JSON. Se ha comprobado que coinciden todos los atributos que se encuentran en el JSON: nombre, padre, prefijo, sufijo y categoría. En el caso de que se encuentre alguna diferencia, se ha escrito en un fichero de logs, el cual se podrá consultar al final de la ejecución.

Si durante el recorrido de las derivadas una de ellas tiene un campo hijo, significa que esa palabra fue la introducida, por lo que el JSON es de tres niveles. Como se ha realizado anteriormente para el padre, volvemos a obtener las flexiones y luego buscar una que tenga el mismo número de derivadas, y se han comprobado dichas derivadas como se ha realizado anteriormente.

Finalmente, para comprobar el funcionamiento de esta nueva aplicación, el tutor ha proporcionado una lista de 263 palabras. Al terminar de ejecutarse y comprobar el log, podemos encontrar unos errores. Dichos errores son debido a que el programa a intentado buscar la palabra “[inexistente]” en la base de datos. Esta palabra no existe, ya que se consideran un eslabón perdido en la cadena de formación de palabras. Es necesario mostrarlo en el grafo ya que los hijos de esa palabra inexistente si son

hermanos. Por todo esto, los grafos que tienen en el nodo inicial la palabra [inexistente] no han podido ser comprobados, ya que se tendría que mirar el grafo realizado.

Sugerencias

Muy comúnmente, los usuarios pueden tener faltas de ortografía a la hora de escribir, por ello, para evitar que el usuario no pueda usar la aplicación web por un error ortográfico, se ha implementado el método “Sugerencias”. Este método es ejecutado cuando el usuario busca una palabra, pero no se encuentra en la base de datos. Con este nuevo método, se pretende mostrar al usuario una lista de palabras alternativas a la que acaba de buscar, con el fin de intentar mostrarle una alternativa próxima a la que él había introducido.

Esto se consigue gracias a otro servicio WCF que ofrece las aplicaciones del IATEXT, llamado Speller. Dicho servicio ya tiene implementado un método sugerencias, al cual se invoca desde este método para obtener una lista de palabras con un cierto parecido a la palabra introducida por el usuario.

Control IP

Para dar una capa extra de protección a la base de datos, se ha propuesto realizar un control mediante el cual se gestionan los accesos de los usuarios a la aplicación, permitiéndoles solo un número razonable de consultas por día.

Cuando un usuario supera un cierto número de consultas, se le mostrará en pantalla un aviso al cual debe contestar de forma correcta para comprobar que es un humano el que está realizando las consultas. En caso de que estas preguntas no sean satisfactoriamente superadas, se le avisará al usuario que su acceso a sido restringido durante un tiempo.

Si el usuario decide ignorar este aviso y continúa usando la aplicación, se solo se le mostrará un grupo de palabras previamente establecidas, sin tener en cuenta la palabra que este usuario haya introducido.

En caso de que el usuario piense que dicha restricción está injustificada, tiene la posibilidad de ponerse en contacto con el administrador de la aplicación web, proporcionando un código mostrado en pantalla. Posteriormente, el administrador podrá comprobar mediante el código enviado por el usuario si dicha restricción ha sido o no injustificada.

Este control de consultas de usuario hace uso de otro servicio WCF llamado ipControl, el cual ya es ofrecido en otras aplicaciones del IATEXT.

Conclusiones

Con este Trabajo de Fin de Grado, el alumno ha conseguido cumplir los objetivos que se marcaron al comienzo del trabajo. Además, el alumno ha sido capaz de adquirir nuevos conocimientos no impartidos durante la carrera, como el uso del lenguaje C#, ASP.NET framework, un entorno de desarrollo nuevo y librerías. También, al estar trabajando en coordinación con el tutor, le ha permitido al estudiante acercarse a la forma de trabajar dentro de un instituto de investigación como el IATEXT.

Con este trabajo, se ha descubierto la importancia de los tratamientos lingüísticos para el desarrollo de herramientas que permiten avanzar con facilidad en la lingüística computacional, así como la relación entre estos procesos con el avance de la inteligencia artificial.

Finalmente, gracias a la herramienta que proporciona Google*, Google Analytics*, se ha visto que hay un gran interés en herramientas de este tipo, ya que como se mencionó anteriormente, para el corto periodo de tiempo que lleva publicada, se ha podido observar un número de consultas interesante. En la imagen a continuación se pueden observar el número de consultas realizado entre el 1 de junio y el 7 de junio.

1.	Familias TIP - Familia de palabras en español	Peru	**40** (25.81%)
2.	Familias TIP - Familia de palabras en español	Spain	**28** (18.06%)
3.	Familias TIP - Familia de palabras en español	Venezuela	**24** (15.48%)
4.	Familias TIP - Familia de palabras en español	Argentina	**16** (10.32%)
5.	Familias TIP - Familia de palabras en español	Dominican Republic	**8** (5.16%)
6.	Familias TIP - Familia de palabras en español	United States	**8** (5.16%)
7.	Familias TIP - Familia de palabras en español	Ecuador	**6** (3.87%)
8.	Familias TIP - Familia de palabras en español	Guatemala	**6** (3.87%)
9.	Familias TIP - Familia de palabras en español	Mexico	**4** (2.58%)
10.	Familias TIP - Familia de palabras en español	Poland	**4** (2.58%)

Ilustración 34 Número de consultas a Familias TIP

Posibles trabajos futuros

En lo referente a posibles trabajos futuros, se había planteado realizar un juego en el cual el usuario introdujera una palabra para buscar la familia, pero el grafo mostrado no tendría texto en los nodos. Las palabras que deberían ir en los nodos se encontrarían en un lateral de la pantalla desde donde el usuario podría moverlos hasta el nodo que él cree que es el correcto. Una vez todas las palabras estuvieran colocadas en los nodos, se le mostraría al usuario que palabras colocó bien y cuales son incorrectas.

Con respecto a las palabras incorrectas, se podría mostrar al usuario cual es la posición correcta o también se podrían mover dichas palabras erróneas al lateral, donde empezaron, para que volviera a intentarlo.

También, como otras aplicaciones del IATEXT, se podría considerar traducir la aplicación a otros idiomas, como el alemán, italiano o chino, ya que esto aumentaría la cantidad de personas que podrían usar la aplicación.

Fuentes de información

- Grafos
  https://www.grapheverywhere.com/tipos-de-grafos/
  Fecha de consulta: 07/06/2021

- Iconos usados
  https://www.freepng.es/
  Fecha de consulta: 07/06/2021

- Google Chrome
  https://es.wikipedia.org/wiki/Google_Chrome
  Fecha de consulta: 05/06/2021

- ASP.NET
  https://dotnet.microsoft.com/learn/aspnet/what-is-aspnet
  Fecha de consulta: 04/06/2021

- Servicios WCF
  https://docs.microsoft.com/es-es/dotnet/framework/wcf/feature-details/wcf-services-and-aspnet
  Fecha de consulta: 04/06/2021

- Librería D3.js
  https://d3js.org/
  Fecha de consulta: 19/05/2021

- JSON
  https://en.wikipedia.org/wiki/JSON
  Fecha de consulta: 07/06/2021

- Librería CSS-Element-Queries
  http://marcj.github.io/css-element-queries/
  Fecha de consulta: 19/05/2021

- Librería Net-Core-JS-Encryption-Decryption
  https://github.com/smartinmedia/Net-Core-JS-Encryption-Decryption
  Fecha de consulta: 19/05/2021

- Posicionamiento SEO
  https://es.wikipedia.org/wiki/Posicionamiento_en_buscadores
  Fecha de consulta: 03/06/2021

- Google Trends
  https://trends.google.com/trends/
  Fecha de consulta: 03/06/2021

- D3 Interactive Tree Graph: Part 3
  https://www.youtube.com/watch?v=-3WdDnuHdkQ&ab_channel=CarlosTighe
  Fecha de consulta: 01/06/2021

- Documentación HiddenField
  https://docs.microsoft.com/en-us/dotnet/api/system.web.ui.webcontrols.hiddenfield.value?view=netframework-4.8
  Fecha de consulta: 02/06/2021

- RegularExpressionValidator Class
  https://docs.microsoft.com/en-us/dotnet/api/system.web.ui.webcontrols.regularexpressionvalidator?view=netframework-4.8
  Fecha de consulta: 02/06/2021

- Lematización
  https://es.wikipedia.org/wiki/Lematizaci%C3%B3n
  Fecha de consulta: 07/06/2021

- Documentación CultureInfo
  https://docs.microsoft.com/en-us/dotnet/api/system.globalization.cultureinfo.name?view=net-5.0
  Fecha de consulta: 03/06/2021

- Cita de Martin Weihrauch
  https://stackoverflow.com/questions/39505636/encryption-in-c-sharp-decryption-in-js-cryptojs
  Fecha de consulta: 03/06/2021