Trabajo de Fin de Grado

Implementación en un grafo de las dependencias internas de una oración

TITULACIÓN: Grado en Ingeniería Informática

AUTOR: Louka David Vanhoucke

TUTORIZADO POR:
Francisco Javier Carreras Riudavets

Fecha [05/2024]

Agradecimientos

En primer lugar, quiero expresar mi más sincero agradecimiento a la Universidad ULPGC - Universidad de Las Palmas de Gran Canaria por haberme brindado la oportunidad de realizar mis estudios y completar este trabajo final de grado. La formación recibida y el entorno académico proporcionado por la universidad han sido fundamentales para mi crecimiento personal y profesional.

Deseo extender mi gratitud a mi tutor del Trabajo Final de Grado, el profesor Francisco Javier Carreras Riudavets. Su orientación, paciencia y apoyo incondicional han sido invaluables a lo largo de este proceso. Gracias a sus consejos y a su dedicación, he logrado superar los desafíos y alcanzar los objetivos propuestos, aprendiendo lecciones valiosas que marcarán mi futuro.

Finalmente, quiero agradecer profundamente a mis padres por su amor, comprensión y apoyo constante. Sin su sacrificio y confianza en mí, no habría sido posible llegar hasta aquí. Ustedes han sido mi pilar fundamental, y este logro también es suyo.

Completar este trabajo final de grado representa un cambio revolucionario en mi vida, no solo por el conocimiento adquirido, sino por las experiencias vividas y las personas que han formado parte de este viaje. A todos, muchas gracias por haberme acompañado y por haber contribuido a mi éxito.

Con gratitud y aprecio,

Louka David Vanhoucke

Resumen

El proyecto se enfoca en el desarrollo integral de una aplicación web utilizando la plataforma ASP.NET. La aplicación tiene como propósito principal proporcionar a los usuarios una herramienta interactiva y visualmente atractiva para explorar las relaciones entre las palabras dentro de una oración. Para lograr este objetivo, se ha implementado un sistema complejo que se integra con una base de datos.

En términos de visualización de datos, se ha aprovechado al máximo las capacidades avanzadas de JavaScript, incorporando bibliotecas especializadas y funcionalidades avanzadas para crear un entorno interactivo y dinámico. Esto se traduce en la generación de un grafo interactivo que representa de manera clara y precisa las relaciones semánticas y sintácticas entre las palabras en una oración. Los usuarios tienen la capacidad de explorar este grafo de manera intuitiva, realizar zoom, arrastrar nodos.

Además de su funcionalidad práctica, la aplicación tiene un potencial educativo significativo. No solo facilita la comprensión profunda de la estructura y el significado de las oraciones, sino que también puede ser una herramienta invaluable en entornos educativos y de aprendizaje. Estudiantes, lingüistas y profesionales del procesamiento del lenguaje natural pueden utilizar esta aplicación para analizar y comprender mejor textos complejos.

Abstract

The project focuses on the comprehensive development of a web application using the ASP.NET platform. The main purpose of the application is to provide users with an interactive and visually appealing tool to explore the relationships between words within a sentence. To achieve this goal, a complex system integrated with a database has been implemented.

In terms of data visualization, the advanced capabilities of JavaScript have been fully utilized, incorporating specialized libraries and advanced functionalities to create an interactive and dynamic environment. This translates into the generation of an interactive graph that clearly and precisely represents the semantic and syntactic relationships between words in a sentence. Users have the ability to intuitively explore this graph, zoom in and out, and drag nodes.

In addition to its practical functionality, the application has significant educational potential. It not only facilitates a deep understanding of the structure and meaning of sentences but can also be an invaluable tool in educational and learning environments. Students, linguists, and natural language processing professionals can use this application to better analyze and understand complex texts, delving into concepts of grammar, syntax, and linguistic analysis.

Índice general

Índice de figuras

Índice de Algoritmos

Capítulo 1

Introducción

Se empieza a revisar la introducción del proyecto actual. En esta sección, detalla los objetivos, el alcance y la metodología que seguirá. Es importante comprender estos aspectos para asegurar que todo esté alineado y pueda avanzar de manera coherente. La introducción también proporciona el contexto necesario para entender la relevancia y el impacto de este proyecto.

1.1. Conceptos básicos y Contexto

El proyecto se centra en el desarrollo de una aplicación web utilizando la plataforma ASP.NET. El propósito principal de esta herramienta es proporcionar a los usuarios una experiencia interactiva y visualmente atractiva para explorar las relaciones entre palabras dentro de una oración.

1.1.1. Objetivo principal

El objetivo de la aplicación es facilitar la comprensión de las estructuras lingüísticas mediante la representación visual entre las palabras de una oración. Esta herramienta no solo ofrece una visión clara de cómo las palabras se interconectan, sino que también permite a los usuarios interactuar con los datos.

1.1.2. Integración y gestión de datos

Para gestionar la información, se ha usado un sistema complejo que se integra con una base de datos. Esta base de datos almacena las relaciones entre palabras, garantizando la disponibilidad de datos.

1.1.3. Visualización de datos interactiva

En términos de visualización, se ha aprovechado las capacidades avanzadas de JavaScript, incorporando bibliotecas especializadas y funcionalidades avanzadas. Esto nos ha permitido crear un entorno dinámico, donde los usuarios pueden explorar un grafo que representa las relaciones entre palabras. Los usuarios pueden realizar zoom, haciendo uso del ratón y moverse libremente por el grafo.

1.1.4. Potencial educativo

La aplicación no solo tiene una funcionalidad práctica, sino que también posee un potencial educativo significativo. Es una herramienta invaluable para estudiantes, lingüistas y profesionales de humanidades. Facilita el análisis y la comprensión de textos complejos, permitiendo una exploración profunda de conceptos de gramática, sintaxis y análisis lingüístico.

1.1.5. Conclusión

En resumen, el proyecto combina tecnología de vanguardia, diseño centrado en el usuario y un gran potencial educativo. Estamos comprometidos a ofrecer una experiencia única y enriquecedora que satisfaga las necesidades de nuestros usuarios y abra nuevas fronteras en el análisis y la comprensión del lenguaje humano.

1.2. Motivación

La motivación que impulsó este proyecto fue la combinación de varios factores. En primer lugar, nos sentimos atraídos por la idea de explorar las posibilidades y recursos ofrecidos por las tecnologías en el ámbito del desarrollo web, especialmente dentro del marco de **ASP.NET**. Esta plataforma, nos ofrecía un lienzo amplio y prometedor para crear una aplicación web innovadora y funcional. Nos intrigaba la idea de sumergirnos en el mundo de ASP.NET y desafiar nuestras habilidades de desarrollo, desarrollando nuevos sistemas, algoritmos y soluciones técnicas que nos permitieran aprovechar al máximo las capacidades de esta tecnología.

Además, nos motivaba el deseo de abordar un problema fascinante: la implementación en un grafo de las dependencias internas de una oración. Este desafío nos parecía apasionante, ya que implica no solo comprender la estructura y el significado del lenguaje humano, sino también visualizarlo al usuario de manera clara.

El proceso de desarrollo de esta aplicación fue todo un viaje, lleno de desafíos técnicos y creativos que nos empujaron a expandir nuestros límites y explorar nuevas posibilidades. Desde la modificación del diseño de la arquitectura del sistema hasta la implementación de algoritmos de visualización de datos, cada paso del camino nos presentaba nuevos obstáculos

que superar y nuevas soluciones que descubrir. Nos sumergimos en el mundo de funcionalidades avanzadas de JavaScript, explorando las últimas tendencias y técnicas en visualización de datos para crear una buena experiencia de usuario.

En última instancia, nuestro objetivo es crear un modulo en la aplicación REGLA que no solo cumpliera con nuestros estándares de calidad y rendimiento, sino que también ofreciera un valor real y tangible a los usuarios. Por lo tanto, este proyecto representa no solo un logro técnico y creativo, sino también un paso adelante en la exploración y comprensión del mundo del lenguaje humano y la tecnología que lo rodea.

Capítulo 2

Estado actual y objetivos iniciales

El proyecto se establece con una serie de objetivos clave que abarcan desde el desarrollo técnico hasta la creación de una herramienta educativa. Estos objetivos son el resultado de un análisis exhaustivo de las necesidades del usuario y de las posibilidades ofrecidas por las tecnologías seleccionadas.

Antes de continuar, se explicará el funcionamiento previo de la aplicación y se presentará al equipo de desarrollo que está detrás de ella.

El Instituto Universitario de Análisis y Aplicaciones Textuales (en adelante IATEXT) centra su investigación en la edición y el análisis de distintos tipos de textos desde perspectivas interdisciplinares (linguística, literaria, histórica, computacional, etc.). El IATEXT tiene como objetivo general producir resultados en investigación básica y desarrollar aplicaciones informáticas multimedia tanto para la investigación, como para los ámbitos educativo, cultural y profesional.

2.1. Uso de la Aplicación: Guía Rápida

Antes de explicar en detalle el trabajo realizado, es fundamental comprender cómo funciona el proyecto ya creado por la empresa. Para ello, se dedicará un apartado a describir el funcionamiento previo de la aplicación ya funcional.

2.1.1. Ingreso de la frase

Al abrir la aplicación, el usuario se encuentra con una caja de texto donde puede escribir cualquier frase que desee analizar. Este es el punto de partida para todo el proceso de análisis. La interfaz inicial es simple y directa, permitiendo a los usuarios comenzar a trabajar sin complicaciones. La caja de texto está diseñado para aceptar una amplia variedad de entradas de texto, desde frases simples hasta oraciones complejas, proporcionando flexibilidad en el tipo de análisis que se puede realizar.

2.1.2. División de la oración

Después de ingresar la frase, el usuario tiene la opción de dividirla en frases más pequeñas utilizando conectores apropiados. Dichos conectores pueden ser de diferentes tipos, dependiendo en la posición de la oración. Estos conectores actúan como marcadores que indican los puntos de separación entre las distintas frases dentro del texto original. La herramienta permite una personalización considerable en la elección de los conectores, lo que ayuda a adaptar el análisis a diferentes estructuras lingüísticas y estilos de redacción. Este paso es crucial para descomponer oraciones complejas en unidades manejables.

2.1.3. Selección de características

Una vez que la oración se ha dividido en frases más pequeñas, se inicia el proceso de asignación de características. El usuario puede seleccionar diferentes características para cada frase o palabra individual según sea necesario. A continuación, se muestran algunos ejemplos de las posibles características que se puede seleccionar:

- ✔ **Identificación de dependencias sintácticas** El usuario puede identificar las dependencias sintácticas entre palabras, determinando cómo se relacionan entre sí gramaticalmente. Esto es esencial para entender la estructura gramatical de la oración y cómo las diferentes partes del discurso se conectan y dependen unas de otras.
- ✔ **Relaciones semánticas** Es posible marcar las relaciones semánticas, que ayudan a entender el significado y la relación conceptual entre las palabras y frases. Este tipo de análisis permite desentrañar el significado subyacente de las oraciones y cómo se transmiten las ideas y conceptos a través del texto.
- ✔ **Propiedades lingüísticas** La interfaz permite la asignación de otras propiedades lingüísticas relevantes, proporcionando un análisis exhaustivo de la estructura y el significado del texto. Esto puede incluir aspectos como la polaridad (positiva o negativa), la función semántica, la presencia, etc.

La interfaz de usuario está diseñada para ser intuitiva, ofreciendo opciones claras y fáciles de seleccionar para que el usuario pueda asignar estas características sin dificultad. Las herramientas y menús están dispuestos de manera que facilitan una navegación fluida y eficiente, permitiendo que el usuario se concentre en el análisis del contenido en lugar de en cómo utilizar la aplicación.

Después de revisar el estado actual del proyecto, se procederá a analizar los objetivos.

2.2. Explorar las capacidades de ASP.NET:

El primer objetivo del proyecto es explorar y aprovechar al máximo las capacidades ofrecidas por la plataforma ASP.NET. Esto implica comprender en profundidad los diferentes

componentes y características de ASP.NET y utilizarlos de manera efectiva para desarrollar una aplicación web funcional.

2.3. Desarrollar sistemas y algoritmos nuevos:

Otro objetivo importante es el desarrollo de sistemas y algoritmos nuevos que permitan la visualización efectiva de las relaciones entre palabras en una oración. Se pretende explorar diferentes enfoques y técnicas para encontrar la solución más adecuada para el problema en cuestión.

2.4. Crear una experiencia de usuario intuitiva:

Un objetivo central del proyecto es crear una experiencia de usuario atractiva que permita a los usuarios explorar las relaciones entre palabras de manera efectiva. Esto implica mantener una interfaz de usuario clara y fácil de usar, así como implementar funcionalidades nuevas de interacción que permitan a los usuarios manipular y explorar el grafo de manera dinámica. Se busca asegurar que la aplicación sea accesible y fácil de entender para una amplia variedad de usuarios, desde estudiantes hasta expertos en lingüística.

2.5. Modificar la base de datos para almacenar relaciones nuevas:

Un componente fundamental del proyecto es la integración de una base de datos para almacenar las relaciones entre palabras. Se pretende modificar el diseño de la estructura de base de datos.

2.6. Potenciar el valor educativo y analítico:

Finalmente, el proyecto también tiene como objetivo potenciar el valor educativo y analítico de la aplicación. Se busca crear una herramienta que no solo ayude a los usuarios a comprender la estructura y el significado de las oraciones, sino que también pueda utilizarse como una herramienta educativa en entornos académicos y de aprendizaje. Se pretende integrar funcionalidades que permitan a los usuarios profundizar en conceptos de gramática, sintaxis y análisis lingüístico, y que puedan utilizarse como una herramienta de estudio y referencia.

Capítulo 3

Competencias específicas y aportaciones del trabajo

El desarrollo de esta aplicación web ha permitido la adquisición de competencias específicas clave y ha aportado significativamente a diversos campos. A continuación, se detalla cómo se han cubierto estas competencias y cuáles son las principales aportaciones de nuestro trabajo al entorno socio-económico, técnico y científico.

Nuestro proyecto ha requerido un profundo conocimiento y habilidades en el desarrollo web utilizando ASP.NET. La modificación de una base de datos eficiente, ha permitido gestionar y actualizar relaciones complejas entre palabras de manera precisa. Este proceso ha fortalecido nuestras competencias en diseño de bases de datos y consultas SQL optimizadas.

Además, el uso avanzado de JavaScript y bibliotecas especializadas como D3.js ha sido esencial para crear visualizaciones interactivas. Estas habilidades han permitido diseñar una interfaz intuitiva que facilita la exploración de relaciones semánticas y sintácticas entre palabras, mejorando significativamente la experiencia del usuario.

En términos de aportaciones, nuestra aplicación representa una innovación en la visualización de datos lingüísticos. Ofrece una nueva forma de entender y analizar la estructura del lenguaje, haciendo que el análisis lingüístico sea más accesible y comprensible. Esta herramienta tiene un gran potencial educativo, proporcionando a estudiantes y profesionales del lenguaje una plataforma para explorar y comprender estructuras lingüísticas sencillas y complejas. Esto fomenta un aprendizaje más profundo y práctico, beneficiando tanto a la educación como al ámbito profesional.

Económicamente, la aplicación puede reducir los costos asociados con el análisis lingüístico manual, ofreciendo una solución automatizada y eficiente. Técnicamente, demuestra cómo la integración de tecnologías avanzadas puede crear aplicaciones web potentes y versátiles, estableciendo un modelo para futuros desarrollos en el campo.

Científicamente, la herramienta facilita la investigación en lingüística, permitiendo a los investigadores visualizar y analizar varios conjuntos de datos lingüísticos. Esto no solo acelera

el proceso de investigación, sino que también abre nuevas posibilidades para el descubrimiento y la comprensión de algoritmos lingüísticos.

Con todo lo mencionado anteriormente, este proyecto ha fortalecido competencias técnicas esenciales y ha realizado aportaciones significativas en diversos campos, ofreciendo una herramienta innovadora que tiene el potencial de transformar el análisis y la comprensión del lenguaje humano.

Capítulo 4

Planificación

Este documento presenta un plan detallado para el desarrollo de una aplicación web interactiva, estructurado en cuatro fases principales. La primera fase se centra en la investigación y el diseño inicial, abarcando la evaluación de tecnologías, el análisis de requisitos y el diseño arquitectónico del sistema. La segunda fase aborda el desarrollo de la aplicación, incluyendo la configuración del entorno de desarrollo, la implementación de la base de datos, la integración de herramientas de análisis lingüístico y el diseño de una interfaz de usuario. En la tercera fase, se procederá al refinamiento y mejora del sistema, optimizando su rendimiento, ajustando el diseño y añadiendo funcionalidades adicionales. Finalmente, la cuarta fase se dedica a la entrega y evaluación de la aplicación, recopilando retroalimentación, completando la documentación necesaria y cerrando formalmente el proyecto. A continuación, se detallan cada una de estas fases, con sus respectivas actividades y objetivos, para asegurar un desarrollo eficiente, garantizando la entrega de una aplicación de alta calidad que cumpla con las expectativas del cliente.

4.1. Fase 1: Investigación y diseño inicial (Semana 1-2)

En esta fase inicial del proyecto, se llevarán a cabo actividades de investigación y diseño para establecer una base sólida para el desarrollo posterior de la aplicación. Esto incluirá:

1. **Investigación de tecnologías:** Se investigarán a fondo las tecnologías necesarias para el desarrollo del proyecto, incluyendo ASP.NET, JavaScript, bibliotecas de visualización de datos. Se evaluarán las ventajas y desventajas de cada tecnología para determinar cuáles son más adecuadas para los objetivos del proyecto.
2. **Análisis de requisitos:** Se llevará a cabo un análisis detallado de los requisitos del proyecto, incluyendo las funcionalidades deseadas, los objetivos del proyecto y las necesidades de los usuarios. Se recopilarán y documentarán todos los requisitos para su posterior referencia durante el desarrollo.

3. **Modificación del diseño de la arquitectura:** Se realizarán algunas ajustes en el diseño de la arquitectura del sistema dependiendo de las nuevas clases.
4. **Creación de documentación:** Se documentarán todas las investigaciones y decisiones de diseño en un documento detallado que servirá como guía para el desarrollo futuro. Esto incluirá diagramas de arquitectura, esquemas de base de datos y cualquier otro material relevante.

4.2. Fase 2: Desarrollo de la aplicación (Semana 3-10)

Una vez completada la fase de investigación y diseño inicial, se procederá al desarrollo de la aplicación. Esta fase incluirá:

1. **Configuración del entorno de desarrollo:** Se configurará un entorno de desarrollo adecuado para trabajar con ASP.NET y las herramientas de desarrollo necesarias. Se instalarán y configurarán todas las herramientas necesarias para el desarrollo del proyecto.
2. **Implementación de la Base de Datos:** Se modificará la estructura de la base de datos y se desarrollarán los scripts necesarios para la creación de tablas necesarias, relaciones y consultas.
3. **Integración de funcionalidades de análisis lingüístico:** Se integrarán las herramientas necesarias para analizar y extraer relaciones entre palabras dentro de las oraciones. Se asegurará de que estas funcionalidades sean precisas y eficientes.
4. **Desarrollo de la interfaz de usuario:** Se diseñará y desarrollará la interfaz de usuario de la aplicación, incluyendo la visualización de datos en forma de grafo interactivo y la implementación de funcionalidades de interacción. Se asegurará de que la interfaz de usuario sea intuitiva y fácil de usar.
5. **Pruebas y depuración:** Se realizarán pruebas exhaustivas de todas las funcionalidades de la aplicación para garantizar su correcto funcionamiento. Se corregirán cualquier error o fallo encontrado durante las pruebas y se optimizará el rendimiento de la aplicación.

4.3. Fase 3: Refinamiento y mejora (Semana 11-13)

Una vez completado el desarrollo inicial de la aplicación, se procederá a su refinamiento y mejora. Esta fase incluirá:

1. **Optimización del rendimiento:** Se identificarán y abordarán posibles cuellos de botella de rendimiento en la aplicación, optimizando consultas de base de datos y mejorando el código donde sea necesario. Se asegurará de que la aplicación sea rápida y receptiva en todo momento.

2. **Ajustes de diseño:** Se realizarán ajustes de diseño y mejoras en la interfaz de usuario según la retroalimentación del usuario y las mejores prácticas de diseño. Se asegurará de que la interfaz de usuario sea atractiva y esté alineada con las expectativas de los usuarios.

3. **Incorporación de funcionalidades Adicionales:** Se agregarán cualquier funcionalidad adicional o mejora que se haya identificado durante las pruebas y revisiones del proyecto. Se asegurará de que la aplicación cumpla con todas las necesidades y expectativas de los usuarios.

4.4. Fase 4: Entrega y evaluación (Semana 14-18)

En la fase final del proyecto, se entregará la aplicación al cliente y se llevará a cabo una evaluación exhaustiva de su rendimiento y calidad. Esto incluirá:

1. **Evaluación de la aplicación:** Se solicitará retroalimentación de los usuarios y se realizarán evaluaciones internas para identificar áreas de mejora y posibles futuras iteraciones del proyecto. Se analizarán los datos recopilados para medir el éxito de la aplicación.

2. **Documentación final:** Se finalizará la documentación del proyecto, incluyendo manuales de usuario, guías de instalación y cualquier otro documento necesario para el mantenimiento y la operación continua de la aplicación. Se asegurará de que toda la documentación esté completa y sea fácilmente accesible.

3. **Cierre del proyecto:** Se realizará una revisión final del proyecto, documentando lecciones aprendidas y cerrando formalmente el proyecto. Se celebrará el éxito del proyecto y se agradecerá a todos los involucrados por su arduo trabajo y contribuciones.

Esta planificación detallada proporciona un marco claro y estructurado para el desarrollo del proyecto, asegurando que todas las actividades necesarias se realicen de manera oportuna y eficiente. Cada fase se desarrollará con cuidado y atención al detalle, con el objetivo final de entregar una aplicación de alta calidad que cumpla con los requisitos y expectativas del cliente.

Capítulo 5

Desarrollo

La metodología en cascada, también conocida como modelo en cascada, es uno de los enfoques más tradicionales y ampliamente utilizados en el desarrollo de software. Este modelo se caracteriza por su estructura secuencial, donde cada fase del desarrollo debe completarse antes de que comience la siguiente. A lo largo de las décadas, la metodología en cascada ha demostrado ser efectiva en proyectos donde los requisitos son bien comprendidos desde el inicio y poco propensos a cambiar. En este contexto, es fundamental entender las fases principales de este modelo: Análisis, Diseño, Implementación y Validación.

5.1. Análisis de requisitos

La fase de 'Análisis de Requisitos' es el punto de partida en el ciclo de vida del desarrollo de software bajo la metodología en cascada. En esta fase, el objetivo principal es comprender y documentar las necesidades y expectativas del proyecto. Esta comprensión detallada es fundamental, ya que los requisitos identificados en esta etapa guiarán todo el desarrollo del modulo del proyecto. Es crucial establecer una base sólida y bien definida en esta fase, pues cualquier error o malentendido en la interpretación de los requisitos puede resultar en costosos retrasos y desalineaciones entre el modulo final.

Durante esta fase, se llevan a cabo diversas actividades enfocadas en la obtención y el refinamiento de los requisitos. Estas interacciones permiten obtener una visión clara de lo que se necesita y se espera del resultado.

5.1.1. Tecnologías utilizadas

El desarrollo de cualquier proyecto, ya sea de ingeniería, informática o cualquier otra disciplina tecnológica, requiere una cuidadosa selección de las herramientas y tecnologías adecuadas. Estas decisiones son fundamentales, ya que determinan en gran medida la eficacia, eficiencia y calidad del producto final. En el contexto específico de nuestro Trabajo Final de

Titulación (TFT), donde nos enfrentamos a la tarea de generar un nuevo modulo del proyecto ya existente, se tiene que tener en cuenta las tecnologías utilizadas del proyecto.

En este proceso de análisis de requisitos, se ha identificado y seleccionado una serie de tecnologías que se consideran esenciales para alcanzar nuestros objetivos planteados. Es crucial comprender a fondo el funcionamiento y las capacidades de cada una de estas tecnologías, ya que ello nos permitirá aprovechar al máximo su potencial y garantizar el éxito del proyecto.

En esta sección, se detallará y explorará en profundidad las tecnologías seleccionadas, destacando su importancia y funcionalidad dentro del contexto del trabajo. Desde los marcos de desarrollo web hasta las bibliotecas de visualización de datos. A lo largo de este análisis, no solo se examinarán las características técnicas de cada tecnología, sino que también se reflexionará sobre su idoneidad para las necesidades específicas y su compatibilidad con el entorno de desarrollo existente.

Al comprender plenamente estas tecnologías y su interacción, se estará mejor equipado para tomar decisiones informadas durante el proceso de desarrollo, identificar posibles desafíos y encontrar soluciones efectivas. Además, esta comprensión nos permitirá adaptarnos y responder de manera proactiva a cualquier cambio o imprevisto que surja durante el curso del proyecto. Con esto mencionado, se mostrará las tecnologías utilizadas para realizar este trabajo.

ASP.NET es un marco de desarrollo web de código abierto desarrollado por Microsoft que permite a los desarrolladores construir sitios web y aplicaciones web dinámicas y escalables. En el contexto del proyecto actual, se ha elegido anteriormente ASP.NET debido a su robustez y su capacidad para manejar aplicaciones web complejas.

ASP.NET, como framework de desarrollo web, utiliza una combinación de diferentes tipos de lenguajes para crear aplicaciones web dinámicas y escalables. Entre estos lenguajes destacan:

- ✔ **HTML (HyperText Markup Language)** Este lenguaje de marcado es la columna vertebral de cualquier página web. HTML se utiliza para estructurar el contenido de una página web, definiendo elementos como encabezados, párrafos, listas, enlaces, imágenes, entre otros.
- ✔ **CSS (Cascading Style Sheets)** CSS se utiliza para dar estilo y formato a los elementos HTML de una página web. Permite controlar aspectos como el color, la tipografía, el diseño y la disposición de los elementos en la página, lo que contribuye a mejorar la apariencia visual y la usabilidad del sitio web.
- ✔ **C# (C Sharp)** Como parte del ecosistema de desarrollo de Microsoft, C# es un lenguaje de programación orientado a objetos que se utiliza en ASP.NET para escribir la lógica del servidor. Con C#, los desarrolladores pueden manipular datos, interactuar con la base de datos, gestionar sesiones de usuario y responder a las solicitudes del cliente, entre otras tareas.

Estos diferentes tipos de lenguajes trabajan en conjunto dentro del framework ASP.NET para crear aplicaciones web completas y funcionales. HTML proporciona la estructura del

contenido, CSS controla la presentación y el estilo visual, mientras que C# maneja la lógica del servidor y la interacción con la base de datos. Esta combinación de tecnologías permite a los desarrolladores construir sitios web y aplicaciones web sofisticadas que ofrecen una experiencia de usuario óptima y cumplen con los requisitos de negocio.

La decisión de mantenerse dentro del marco establecido por ASP.NET se fundamenta en múltiples consideraciones técnicas y estratégicas. Por un lado, la familiaridad y experiencia previa del equipo de desarrollo con esta tecnología permiten una transición fluida y eficiente entre los diferentes componentes y módulos de la aplicación. Esto conlleva a una reducción del tiempo y esfuerzo requerido para implementar nuevas funcionalidades, así como una mayor facilidad para mantener y dar soporte a la aplicación a lo largo del tiempo.

Además, al permanecer dentro del entorno de desarrollo proporcionado por ASP.NET, se asegura la compatibilidad entre los distintos elementos de la aplicación, de tal manera nuestro modulo será compatible sin ningún problema. Esto es especialmente relevante en el contexto de la representación del grafo generado por la aplicación, ya que al utilizar el mismo framework utilizado en la aplicación principal, garantizamos una integración sin fisuras entre los datos y la interfaz de usuario. De esta manera, se puede ofrecer una experiencia consistente y fluida para los usuarios, sin comprometer la funcionalidad ni la estabilidad de la aplicación.

Por otro lado, se hace uso de **Microsoft Access** como un pilar fundamental en la gestión de datos. Como un sistema de gestión de bases de datos relacional desarrollado por Microsoft, su elección se justifica por su reconocida eficiencia y facilidad de uso. Este software ofrece una interfaz intuitiva, lo que resulta especialmente valioso para proyectos que requieren una gestión ágil y efectiva de pequeños volúmenes de datos.

Con la misma razón que la tecnología anterior, la gran mayoría de la base de datos ya estaba diseñada en Microsoft Access. Por lo tanto, las modificaciones que se realizan en el proyecto serán compatibles si se adaptan a la tecnología ya implementada en este entorno.

Esta decisión se fundamenta en la necesidad de optimizar recursos y maximizar la eficiencia del desarrollo. Al trabajar dentro del marco establecido por Microsoft Access, se aprovecha la infraestructura existente y se minimiza el riesgo de incompatibilidades o complicaciones técnicas. Además, al mantener la coherencia en la tecnología utilizada para la gestión de la base de datos, se garantiza una transición suave y sin problemas entre los distintos componentes de la aplicación.

En el corazón de la estrategia de visualización de datos reside la combinación sinérgica de **DB3.js y JavaScript**. Ambas tecnologías, fundamentales en el panorama del desarrollo web moderno, desempeñan roles complementarios y esenciales en la creación de experiencias de usuario.

JavaScript, como el lenguaje de programación líder en la web. Trabajando en conjunto con ASP.NET, JavaScript permite la manipulación dinámica de elementos HTML y la respuesta ágil a las acciones del usuario. En este caso, se utilizará para poder gestionar le grafo en la aplicación.

DB3.js es una biblioteca construida sobre la base de JavaScript, agrega una capa adicional de funcionalidad específicamente diseñada para la visualización de datos. Su conjunto de

herramientas avanzadas hace posible generar gráficos dinámicos y visualizaciones interactivas que transforman datos crudos en historias visuales claras y convincentes. Ya sea representando relaciones complejas entre conjuntos de datos o destacando tendencias y patrones ocultos.

La sinergia entre DB3.js y JavaScript proporciona una paleta completa de posibilidades creativas para la visualización de datos. A través de JavaScript, se puede integrar y controlar fácilmente las visualizaciones generadas por DB3.js, permitiendo interacciones intuitivas y personalizadas con los datos. Además, la flexibilidad y la extensibilidad de JavaScript se permiten adaptar las visualizaciones a las necesidades específicas del proyecto, desde la personalización de estilos hasta la incorporación de funciones interactivas adicionales.

Al aprovechar al máximo las capacidades combinadas de DB3.js y JavaScript, se puede ofrecer experiencias de usuario inmersivas y altamente informativas que impulsan la comprensión y el compromiso del usuario. Esta combinación poderosa no solo enriquece la presentación de datos en el proyecto, sino que también fortalece su impacto y efectividad general.

5.1.2. Análisis de las dependencias de las palabras en una oración

Dado que ya está definido las tecnologías que se van a utilizar en el proyecto, se procede a analizar el proceso de implementación. Específicamente, se enfocará en cómo lograr que cada palabra de una oración segmentada pueda depender de otra palabra en la misma oración, excluyéndose a sí misma.

El primer paso en dicho proceso de implementación será crear un nuevo campo para asignar dependencias entre las palabras de una oración. Esta característica se integrará en la clase que maneja la información de todas las palabras de las oraciones, denominada Frases_Palabras. Dicha clase contiene toda la información necesaria de cada palabra en cada frase, por lo cual, es lo más adecuado añadir el campo de dependencia en dicha clase.

Por lo tanto, se define el nuevo campo de dependencia. Este campo será un atributo de tipo entero, que se agrega a la clase Frases_Palabras con el nombre Dependencia. El propósito de este atributo será almacenar la posición de la palabra de la cual depende cada palabra en la oración. Este enfoque permitirá rastrear las relaciones de dependencia entre las palabras de manera precisa y eficiente.

En caso de no guardar la posición, se ha considerado primero la opción de almacenar la palabra tal cual como el usuario la selecciona. Sin embargo, esto podría causar problemas, como cuando la misma palabra aparece varias veces en una oración. En estos casos, no sería claro a cuál instancia de la palabra se hace referencia.

Ejemplo: Se supone que se tenga la oración: El **gato** juega con el **gato** del vecino.

Si el usuario selecciona la palabra 'gato' sin especificar la posición, se tendría dificultad para determinar si se refiere al primer 'gato' o al segundo gato en la oración escrita, con lo tanto para construir posteriormente el grafo, dará fallo, por ello esta opción no es válida, y se tiene que encontrar una otra solución.

Dado este suceso, se ha decidido guardar la posición en la base de datos, será necesario crear un nuevo campo en la misma. Este campo debe ser de tipo entero y almacenará un número que indicará la posición específica de la palabra seleccionada dentro de la oración.

Este número entero servirá como un identificador único para la ubicación de cada palabra seleccionada, eliminando cualquier ambigüedad sobre a qué instancia de la palabra se está refiriendo el usuario.

Por lo tanto, se ha establecido y demostrado que en la base de datos se trabajará con la posición del campo dependencia, y este sería el proceso para realizarlo:

- ✔ **Creación del campo** En la estructura de la base de datos, se añade una nueva columna de tipo entero. Este campo se puede denominar, por ejemplo, 'posicion_palabra' o algo similar que indique claramente su propósito, se añade al final de todas las columnas existentes para no tener problemas al equivocarse luego en la recuperación de los datos.
- ✔ **Almacenamiento de datos** Este proceso sería modificar los modelos de del proyecto y añadir en ellos este nuevo campo identificando que ahora hay una columna más en la tabla.
- ✔ **Recuperación de datos** Esto es lo mismo que el almacenamiento, se modifica los modelos ya que hay más columnas en la tabla actual.

Ahora que se ha definido la estrategia para gestionar las dependencias, es momento de seguir con la fase de implementación del grado adicional que se desea generar. En este proceso, la prioridad principal es asegurarse de que se captura adecuadamente las dependencias seleccionadas por el usuario y luego emplear un algoritmo para organizar las palabras en un orden adecuado, para que sea comprensivo para el usuario final.

Además, se debe garantizar que el proceso de posicionamiento de las palabras sea intuitivo para el usuario final. Esto implica no solo considerar la lógica interna del algoritmo, sino también su interfaz y experiencia de usuario asociada. En este sentido, no solo basta con que el algoritmo funcione correctamente en términos técnicos. También se debe prestar atención a cómo se presenta y se comunica esta funcionalidad al usuario. La interfaz de usuario (UI) juega un papel fundamental en este aspecto, ya que es el medio a través del cual los usuarios interactúan con la herramienta.

Para comenzar, es fundamental establecer que el primer paso importante al realizar, consiste en la instalación o referencia de la biblioteca, instalando una versión compatible con **D3.js**, un recurso imprescindible para aprovechar su potencial en la generación de gráficos interactivos. Una vez asegurada su disponibilidad, se procede a crear un nuevo script en JavaScript, el cual deberá ser posteriormente referenciado en la cabecera de nuestro documento HTML.

Dentro de este script, es esencial definir una función principal que se encargará de orquestar el proceso de generación del grafo. Esta función, cuyo nombre y estructura pueden variar según las necesidades específicas del proyecto, será invocada desde el backend en respuesta a la solicitud del usuario para visualizar el grafo generado.

Un aspecto crucial en esta fase es la creación de una nueva clase denominada **Grafo**. Esta clase, diseñada para gestionar eficientemente los nodos y aristas del grafo, será responsable de almacenar la información esencial sobre las palabras de la oración y las dependencias entre ellas. Una vez poblado con esta información, el objeto de la clase Grafo se transmitirá al script de JavaScript correspondiente, proporcionando así los datos necesarios para la construcción del grafo.

Una vez que el objeto Grafo ha sido creado y poblado con la información pertinente, se procede a su transferencia al script de JavaScript mediante algún mecanismo de comunicación adecuado, como por ejemplo, la serialización de objetos o el uso de API específicas.

Con todos estos elementos en su lugar, el script de JavaScript estará en condiciones de generar el grafo dinámicamente a partir de la información proporcionada por el usuario. Este proceso implica la manipulación de los datos del objeto Grafo y su representación visual utilizando las capacidades de la biblioteca D3.js, lo que permitirá crear una visualización interactiva y atractiva del grafo de dependencias entre las palabras de la oración.

5.2. Diseño de la aplicación

En este apartado, se adentrarán en una exploración detallada del diseño de una aplicación destinada a la generación de gráficos de dependencias. Se centra exclusivamente en el proceso de diseño, desde la concepción inicial de la interfaz hasta la implementación de las funcionalidades clave. Este análisis abarcará aspectos fundamentales como la estructura de datos, la arquitectura de la aplicación y la presentación visual, con el objetivo de ofrecer una experiencia fluida y significativa para el usuario final.

5.2.1. Diseño de las dependencias de las palabras en una oración

En esta sección, se enfoca en el diseño de las dependencias, es decir, determinar el lugar adecuado para incluir la opción de seleccionar la dependencia de cada palabra en una oración. Este proceso implica integrar esta funcionalidad en el mismo contexto donde se encuentran las demás funciones asociadas a cada palabra dentro de la estructura de la oración. Además,

se considerarán las mejores prácticas de diseño de interfaces para asegurar que la opción de seleccionar la dependencia se integre la mejor manera.

Las dependencias encajarían lo mejor posible con el estilo de la página y estarán alineadas al nivel de contexto y necesidad del campo. Este enfoque garantizará que los usuarios puedan interactuar fácilmente con la función de dependencia dentro del contexto de la aplicación.

5.2.2. Grafo dinámico

La estructura del grafo es un elemento crucial para garantizar una experiencia satisfactoria para el usuario. Debe ser precisa, lógica y clara para facilitar la interpretación de la información representada. Al diseñar esta estructura, es fundamental considerar una serie de aspectos que contribuirán a mejorar la comprensión del grafo.

La disposición de nodos y aristas en un grafo es crucial para facilitar su comprensión y análisis. Una disposición desorganizada puede hacer que el grafo sea difícil de interpretar y entender, lo que dificulta la extracción de información y la identificación de patrones importantes.

Para lograr una disposición coherente y ordenada de nodos y aristas en el espacio visual, es útil seguir algunas pautas de diseño:

- ✔ **Jerarquía y agrupación:** Agrupa nodos relacionados y organízalos jerárquicamente. Esto puede ayudar a identificar subconjuntos de datos y relaciones entre ellos. Por ejemplo, si se está representando una red social, se puede agrupar nodos por intereses, ubicaciones geográficas o relaciones familiares. En nuestro caso, se tiene en cuenta esta pauta cuando se vea donde están ubicados las dependencias en la oración y agruparlos de la mejor manera posible.
- ✔ **Alineación y simetría:** Alinear nodos y aristas de manera que sigan un patrón visual coherente. La simetría puede facilitar la identificación de relaciones simétricas en el grafo y hacer que la disposición sea más estéticamente agradable. También centrar los nodos en un mismo eje para ver claridad donde va cada palabra ubicada en el grafo.
- ✔ **Evitar superposiciones:** Es esencial evitar que las aristas se crucen entre sí y que los nodos se superpongan. Esto se logra ajustando las posiciones de los nodos y las trayectorias de las aristas. En este caso, este aspecto es fundamental para preservar la estructura original de la oración. Aunque las palabras estén alternadas, aún se debe poder reconocer la oración original.
- ✔ **Espacio uniforme:** Distribuye los nodos de manera uniforme en el espacio disponible para evitar congestión visual. Ajusta el tamaño de los nodos y el espacio entre ellos según sea necesario para mantener una distribución equilibrada.
- ✔ **Orientación consistente:** Mantén una orientación consistente en todo el grafo para facilitar la navegación y la comparación entre diferentes partes del mismo.

- ✔ **Interactividad:** Proporciona funcionalidades interactivas que permitan a los usuarios explorar el grafo de manera dinámica, como la capacidad de hacer zoom, arrastrar nodos y filtrar ciertas partes del grafo según criterios específicos.

Al tener en cuenta estas pautas de diseño generales, se puede crear una disposición visual coherente y ordenada que facilite la interpretación y el análisis del grafo, permitiendo a los usuarios extraer información de manera eficiente y efectiva.

Por otro lado, el etiquetado adecuado de nodos y aristas es esencial para garantizar la comprensión y la interpretación efectiva de un grafo. Esta práctica consiste en asignar etiquetas claras a cada nodo y arista, utilizando texto descriptivo que facilite la identificación rápida de los elementos y sus relaciones.

Cuando se etiquetan los nodos, es importante proporcionar información relevante que describa su significado o función dentro del contexto del grafo. Esto puede incluir nombres, identificadores únicos, valores numéricos o cualquier otro atributo que sea relevante para la comprensión de los datos representados. Además, las etiquetas deben ser lo suficientemente legibles y visualmente distintivas para que los usuarios puedan identificar fácilmente cada nodo y su correspondiente información.

En cuanto al etiquetado de las aristas, es crucial indicar claramente la naturaleza de la relación entre los nodos conectados por cada arista. Esto puede implicar el uso de palabras clave, símbolos o cualquier otro tipo de representación que ayude a transmitir la información. Por ejemplo, en un grafo que representa una red de transporte, las aristas podrían etiquetarse con el tipo de conexión (gato, playa, baño) y posiblemente con información adicional como la distancia o el tiempo de viaje.

Además de proporcionar etiquetas descriptivas, también es importante considerar la posición y el formato de las etiquetas dentro del grafo. Las etiquetas deben colocarse cerca de los nodos o aristas correspondientes, sin obstruir la visualización general del grafo. Además, el tamaño, el color y la fuente del texto de las etiquetas deben elegirse para garantizar la legibilidad y la coherencia visual con el resto del grafo. De tal manera, las etiquetas implementadas deberían ser visuales para las palabras que contengan la oración asociada.

El etiquetado adecuado no solo mejora la comprensión del grafo por parte de los usuarios, sino que también facilita la interpretación de los datos y la identificación de patrones importantes. Al proporcionar información clara sobre cada elemento del grafo, se fomenta una experiencia de usuario más intuitiva, lo que a su vez contribuye a una toma de decisiones más clara. En resumen, el etiquetado adecuado es un aspecto fundamental del diseño de grafos que no debe pasarse por alto al crear visualizaciones de datos efectivas y significativas.

5.3. Implementación

A continuación, se procede a implementar estas ideas en el entorno de desarrollo. Este proceso se dividirá en dos partes principales: la implementación de las dependencias de las palabras en una oración y la creación del grafo relacionado con las dependencia en la oración.

La primera parte, **la implementación de las dependencias de las palabras en una oración**, se centrará en integrar y configurar todas las tecnologías y bibliotecas necesarias para que nuestra aplicación funcione correctamente. Esto incluye la configuración del entorno ASP.NET, la integración de Microsoft Access como la base de datos principal, y la incorporación de DB3.js junto con JavaScript para la generación de gráficos necesarios. Cada componente será cuidadosamente configurado para realizar el trabajo, garantizando que la aplicación funcione igual de bien como estaba antes de implementar nuestro modulo.

La segunda parte se centrará en la **implementación del grafo**. Utilizando DB3.js y JavaScript, se prioriza a desarrollar las visualizaciones interactivas necesarias para representar los datos de manera clara y comprensible. El grafo permitirá a los usuarios explorar y analizar las relaciones entre diferentes elementos de datos visualmente atractiva. Se prestará especial atención a la personalización y optimización de los gráficos para asegurar que sean no solo funcionales, sino también estéticamente agradables y fáciles de usar.

5.3.1. Instalación del entorno

Dado que el proyecto actual, REGLA, está desarrollado en ASP.NET, se utiliza **Microsoft Visual Studio 2022** para facilitar la realización de cambios y mejoras en el proyecto. Dicho framework ofrece un entorno de desarrollo integrado (IDE) completo que es ideal para trabajar con ASP.NET, proporcionando herramientas avanzadas para el desarrollo, depuración y despliegue de aplicaciones web.

Tras este proceso, se instala Microsoft Access, teniendo en cuenta de utilizar la versión correcta, la cual es Microsoft Access 2016. Esta versión es compatible con nuestras necesidades y garantiza que la base de datos funcione de manera óptima dentro del entorno del proyecto.

A continuación, se procede a configurar el entorno de desarrollo para asegurarnos de que todos los extensiones necesarios (NuGet) sean funcionales y compatibles con nuestro entorno. Los paquetes NuGet son esenciales para gestionar las dependencias de la aplicación y asegurar que todas las bibliotecas y componentes necesarios estén disponibles y actualizados.

Para acceder al proyecto actual, se ha utilizado **Azure DevOps**, una plataforma de colaboración y gestión de proyectos de Microsoft. Gracias a la autorización proporcionada por el tutor de este Trabajo Fin de Título, se ha hecho posible el desarrollo del proyecto a través de Azure DevOps. Esto hace posible clonar el repositorio del proyecto, realizar cambios, gestionar versiones y colaborar con otros miembros del equipo que estén modificando el proyecto actualmente.

El último paso consistirá en la instalación de la biblioteca adecuada de D3.js para asegurar su compatibilidad con la versión correcta del proyecto. Se ha optado por seleccionar la última versión de la biblioteca, ya que proporciona todas las funcionalidades necesarias para llevar a cabo los cambios requeridos. Esta decisión se basa en la capacidad de la última versión para ofrecer las características más actualizadas y avanzadas, lo que garantiza una mayor flexibilidad y eficacia en la implementación de las funcionalidades deseadas.

Este enfoque asegura que el entorno de desarrollo esté completamente configurado y listo

para realizar modificaciones y mejoras en el proyecto REGLA. Utilizando Visual Studio 2022, Microsoft Access 2016 y la biblioteca D3.js, junto con la capacidad de colaborar y gestionar el proyecto a través de Azure DevOps, se debe estar bien equipados para avanzar con éxito en el desarrollo y mantenimiento del proyecto.

Una vez que se ha instalado el entorno de forma correcta, se prosigue a la primera tarea de este trabajo, lo cual es añadir las dependencias de las palabras en las oraciones.

5.3.2. Implementación adecuada de las Dependencias

Para llevar a cabo la implementación de las dependencias, se ha diseñado un enfoque escalonado que se ejecutará en varias etapas distintas. Este enfoque por fases permite una gestión más efectiva del proyecto y facilita la adaptación a posibles cambios y ajustes en el proceso de desarrollo. Se han realizado diversas modificaciones en el proyecto para garantizar que cada fase se realice de mejor manera. Cada parte del proceso se abordará de manera sistemática, permitiendo una mayor claridad en la implementación de las dependencias.

Esto asegura que el proyecto avance de manera progresiva hacia su objetivo final, minimizando los riesgos y maximizando la calidad del resultado final.

Implementación visual de dropdownlist en la interfaz de usuario de ASP.NET

Como se mencionó en el capítulo del análisis de los requisitos, primero se aborda la parte visual del proyecto. En esta sección, se añade un nuevo elemento ASP.NET llamado DropDownList a la aplicación para mejorar la interfaz de usuario y la funcionalidad del sistema.

Para comenzar, se inserta el nuevo DropDownList en la página ASP.NET correspondiente y se le asigna un ID único. Este identificador único es esencial para referenciar el DropDownList tanto en el código del servidor como en los estilos CSS. Asignar un ID único permite una gestión sencillo del elemento dentro de la aplicación, asegurando que cada componente sea identificado y tratado de manera individual.

Seguidamente, se avanza al paso de implementar una función para el evento OnDataBound del DropDownList. El evento OnDataBound se dispara después de que los datos se han enlazado al DropDownList, permitiendo realizar ajustes adicionales o procesamientos necesarios una vez que los datos estén cargados. En esta función, se añade un elemento de marcador de posición con el texto 'Nada Seleccionado', lo cual por defecto, todos los campos tendrán asignado este valor. Este marcador es necesario ya que si el usuario no quiere asignar ninguna dependencia, tiene la opción de dejarlo como estaba.

Además, para mantener la coherencia visual con los demás DropDownList existentes en la aplicación, se asigna a la misma clase CSS a este nuevo elemento. La consistencia en el diseño es crucial para proporcionar una experiencia de usuario uniforme. Al aplicar la misma clase de estilos, se comprueba que el nuevo DropDownList se integre de manera correcta con

el resto de la interfaz de usuario, respetando el diseño y formato establecidos. A continuación se muestra una imagen de la aplicación en funcionamiento.

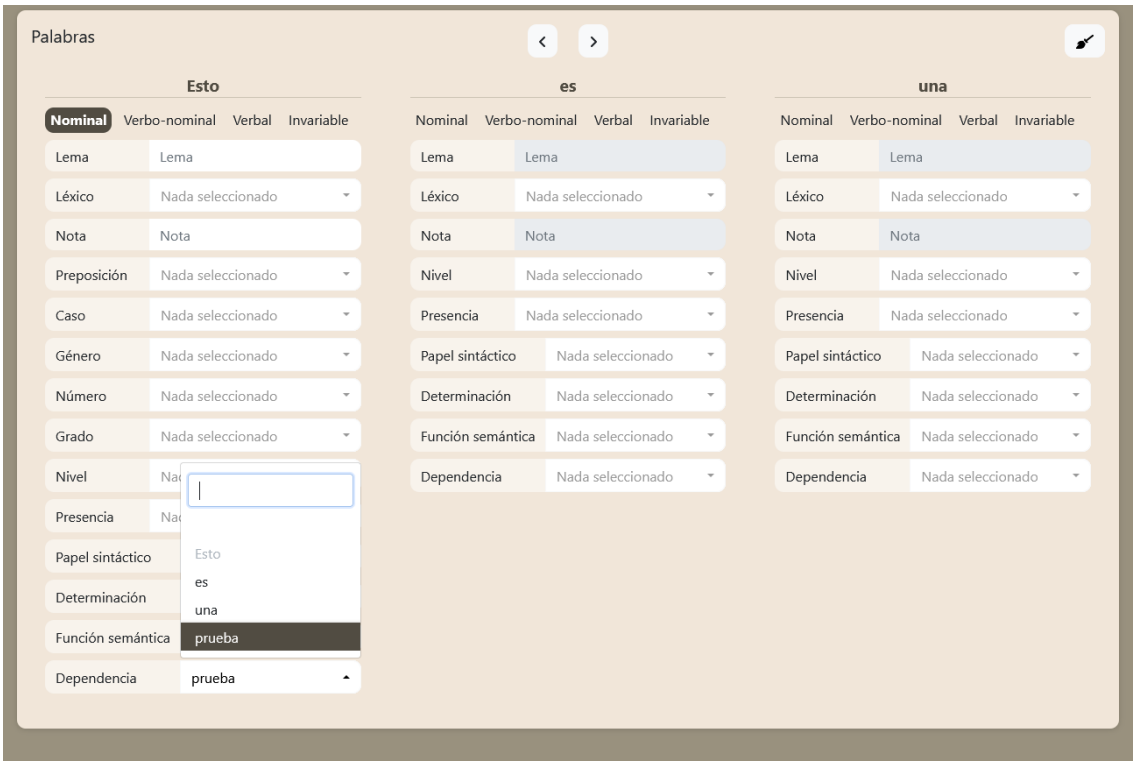

Ilustración 5.1: Captura de las dependencias de las palabras en funcionamiento

Configuración del backend para la funcionalidad del dropDownList en ASP.NET

Continuando con la implementación de la parte visual del proyecto, ahora se aborda la configuración del backend para completar la funcionalidad del nuevo DropDownList. Este paso es fundamental para asegurar que el control no solo se visualice correctamente, sino que también funcione adecuadamente dentro del sistema.

Primero, se define la lógica para asignar los datos relevantes al DropDownList. Esto se realizará en el código del servidor, donde se escribirá una función que se ejecutará cuando el evento OnDataBound del DropDownList sea disparado. Este evento se activa después de que los datos han sido enlazados al control, permitiendo procesar y ajustar los datos según sea necesario antes de presentarlos al usuario.

En esta función del backend, se extraen las palabras de una colección de frases almacenadas en la base de datos. Cada frase se procesará para separar las palabras individuales, las cuales serán luego filtradas y limpiadas de cualquier carácter no deseado, como signos de puntuación o espacios innecesarios a través de algunas expresiones regulares escritas. Este proceso garantiza que solo las palabras relevantes y significativas se añadan al DropDownList para luego mostrarlo en la aplicación.

Con todo lo mencionado e implementado hasta ahora, se ha creado una funcionalidad robusta para que los usuarios puedan interactuar con el DropDownList. El DropDownList permite a los usuarios seleccionar la dependencia de cada palabra en una frase, excluyendo la propia palabra que está siendo seleccionada. Esta palabra aparecerá en gris, en modo 'Disabled' 5.1, lo que asegura que no pueda ser seleccionada, manteniendo la coherencia y precisión en la selección de dependencias.

La integración con el Repeater es crucial para esta funcionalidad. El Repeater nos permite replicar el código del DropDownList para cada palabra en la sección de PalabrasCompletas. Esto significa que, para cada palabra en la frase, se generará un DropDownList individual, mostrando a los usuarios la capacidad de seleccionar dependencias específicas. Esta replicación asegura que el proceso sea uniforme y consistente para cada palabra, mejorando la experiencia del usuario y facilitando la gestión de dependencias.

Con la funcionalidad del DropDownList y el Repeater establecida, el siguiente paso crítico es la integración con la base de datos. Esto implica dos aspectos fundamentales: la capacidad de guardar los datos seleccionados por el usuario y la capacidad de recuperar estos datos para su posterior uso y análisis.

Guardado de datos seleccionados por el usuario

Para almacenar las selecciones realizadas por el usuario, es necesario extender nuestra tabla en la base de datos añadiendo una nueva columna. Esta columna adicional se llamará 'dependencia' y estará destinada a almacenar la posición correspondiente de cada dependencia seleccionada. De esta manera, se podría registrar y gestionar eficientemente las relaciones de dependencia de cada palabra en las frases.

Tras añadir la nueva columna 'dependencia' en nuestra tabla de base de datos, el siguiente paso es ajustar la lógica de consulta para manejar este nuevo campo. La conexión a la base de datos ya está establecida pero antes de poder seguir se tiene que entender el funcionamiento por completo.

Con la conexión a la base de datos ya entendido, se procede a actualizar las consultas **SQL** para incluir la nueva columna 'dependencia'. Específicamente, se necesitará modificar las consultas de inserción (INSERT INTO) para manejar la información adicional proporcionada por el usuario, ya que dichas consultas no son válidas con el nuevo campo de dependencia importado. Nos aseguraremos de que nuestras consultas SQL reflejen este cambio adecuadamente, incluyendo el nuevo campo en el orden correcto en la consulta para que se guarda en su lugar adecuado.

El proceso de actualización de dichas consultas implica el uso de la instrucción INSERT INTO. Estos datos pasados por la consulta, se estructurarán de manera que sean fáciles de insertar en la base de datos, es decir ordenados y luego pasados a la consulta para guardarlos.

Después de ejecutar la consulta, es esencial validar que los datos se hayan insertado correctamente, ya que con las modificaciones realizados puede ocurrir que no todo funcione como estaba hecho anteriormente. Esto implica realizar consultas de verificación o pruebas

para asegurar que los nuevos registros se han almacenado como se esperaba, sin cambiar el funcionamiento anterior. La validación asegura que no haya discrepancias en los datos y que todas las dependencias seleccionadas se hayan guardado adecuadamente.

Además de la inserción de nuevos registros, se debe considerar la posibilidad de que una frase ya esté guardada en la base de datos. En tales casos, en lugar de insertar un nuevo registro, se tendrá que actualizar los campos existentes. Para gestionar esto, primero hará uso de una consulta SELECT, la cual se ha tenido que modificar también respecto al nuevo campo añadido. Para verificar si la frase ya existe en la base de datos, esta consulta buscará la frase específica y, si se encuentra, procederemos a actualizar los campos correspondientes.

La consulta SELECT se construirá para buscar la frase en la tabla utilizando un identificador único o una clave relevante que identifique de manera precisa el registro de la frase. Si la consulta SELECT devuelve resultados, esto indicará que la frase ya está presente en la base de datos y que se tiene que proceder con una actualización.

Recuperación de Datos seleccionados por el usuario

Una vez que la información proporcionada por el usuario está guardada en nuestra tabla, es fundamental poder recuperar esta información para asegurar una experiencia de usuario coherente y fluida. Esta funcionalidad es esencial porque si el usuario introduce la misma oración en el futuro, el programa debe ser capaz de recuperar los datos asociados previamente a esa oración. Esto permite que el usuario vea y modifique la información guardada sin tener que reingresarla desde cero.

El principio para realizar este proceso es siempre verificar si la oración introducida por el usuario es nueva o ya existe en nuestra base de datos. Para lograr esto, se hará uso de la consulta SELECT, que se ha mencionado anteriormente, que buscará la oración específica en la base de datos. Si la consulta devuelve resultados, significará que la oración ya existe y que se debe proceder a recuperar los datos asociados.

Cuando la consulta SELECT confirma la existencia de la oración en la base de datos, es crucial recuperar todos los datos correspondientes en el orden adecuado. Esto incluye no solo la oración en sí, sino también las dependencias y cualquier otra información relacionada que se haya guardado previamente. La recuperación precisa de estos datos es vital para asegurar que el usuario pueda ver y modificar la información con precisión.

Para realizar esto se hace uso de un Reader, es muy importante no alternar el orden cuando el reader vaya leyendo nuestra tabla para sacar los datos, una vez que se ha sacado los datos, se asignan a la clase llamada Frases_Palabras donde se almacenará todo lo que se haya recuperado, en caso de que el campo en la base de datos estaba vacía, se pone un string vacío o -1 dependiendo si es de tipo string o entero.

5.3.3. Implementación del grafo

Durante este proceso, se empieza a construir el grafo, en el cual cada palabra de la oración se representará como un nodo individual. Se analizan las dependencias entre estas palabras y se visualizan como aristas que conectan los nodos correspondientes. Este enfoque permite ilustrar claramente las relaciones gramaticales y sintácticas dentro de la oración.

Es fundamental asegurarse de que cada nodo esté correctamente posicionado para evitar superposiciones, y que las aristas no se crucen innecesariamente. A través de esta metodología, se ofrece una representación visual clara y lógica que facilita la comprensión de las dependencias entre las palabras. Este proceso se subdivide en algunas subsecciones, comenzando por la generación del Canvas, que servirá como base para dibujar el grafo.

Generación del canvas para el grafo final

La generación del canvas para el grafo comienza con la selección del contenedor en el que se dibujará el grafo. Dentro de este contenedor, se crea un elemento SVG que servirá como el área de dibujo principal. Este SVG se configura para ajustar automáticamente su tamaño al contenedor, garantizando que se adapte a diferentes tamaños de pantalla y proporciones. Además, se aplican estilos para asegurar que el texto dentro del grafo sea legible, utilizando una fuente sans-serif de 12px, lo cual encaja con el estilo de la página.

Para mejorar la interactividad, se implementa una funcionalidad de zoom que permite a los usuarios acercar y alejar el grafo, así como desplazarse por él. Esta funcionalidad de zoom no solo facilita la exploración y comprensión del grafo, sino que también permite que el canvas se extienda de manera infinita. Esto significa que, independientemente del tamaño de la oración, el grafo puede ajustarse y expandirse según sea necesario, evitando problemas de encaje y manteniendo la estructura clara. La combinación de estos elementos proporciona una base flexible y dinámica sobre la cual se construirá el grafo, permitiendo a los usuarios explorar y entender las relaciones entre los nodos de manera intuitiva.

Una vez configurado el elemento SVG en el código de JavaScript, el siguiente paso implica preparar el entorno en el frontend para indicar dónde dibujar el canvas. Para lograr esto, se crea un contenedor en la página web, añadiendo un div (Content Division element) con un id específico en el HTML. Este div sirve como punto de referencia para que el SVG se inserte y se renderice correctamente. En el código JavaScript, se selecciona este div utilizando D3.js con d3.select y se añade el SVG dentro de este contenedor. Esta configuración garantiza que el grafo se dibuje en el lugar correcto de la página, manteniendo la estructura clara y organizada.

A continuación, se procede a generar los nodos del grafo. Inicialmente, estos nodos se crean de forma estática para establecer la estructura básica y asegurar que el grafo se visualice correctamente. Este enfoque permite verificar la disposición y la apariencia de los nodos sin la complejidad añadida de los datos dinámicos.

Generación de los nodos de forma estática

Primero, se define manualmente algunos nodos y sus posiciones en el canvas. Esto nos ayudará a comprobar la funcionalidad básica de la representación gráfica y a realizar ajustes necesarios en el diseño y la interactividad del grafo. Una vez que estamos satisfechos con la disposición estática, se avanzará a la siguiente fase.

El dataset que se utilizará para esta fase consta de los siguientes nodos:

```
const nodes = [
    { id: 0, name: "Node 1" },
    { id: 1, name: "Node 2" },
    { id: 2, name: "Node 3" },
    { id: 3, name: "Node 4" }
];
```

Se hace uso de esta información para crear manualmente los nodos en el canvas y asignarles posiciones específicas. Este enfoque nos permitirá visualizar cómo se distribuyen los nodos en el grafo y ajustar la disposición según sea necesario antes de proceder con la implementación de los datos dinámicos.

Ahora se añaden los nodos al SVG que previamente creado, el cual funciona como el lienzo principal para el grafo. Esta etapa es crucial, ya que los nodos representan los elementos fundamentales del grafo y su visualización es esencial para comprender la estructura y las relaciones dentro del mismo.

Para lograr esto, se apoya en la biblioteca D3.js. Se comienza seleccionando todos los elementos con la clase 'node' dentro del SVG, o creando estos elementos si aún no existen. Esta selección se realiza mediante el método **selectAll('.node')**, lo que nos permite vincular estos elementos con los datos provenientes del conjunto de nodos que se definen previamente.

Una vez seleccionados o creados los elementos, se procede a añadir un círculo para representar visualmente cada nodo en el grafo. Esto se logra mediante el método **.enter().append('circle')**, que crea un elemento en forma de un circulo para cada nodo en el conjunto de datos. Estos círculos actuarán como puntos de referencia visuales en nuestro grafo, permitiendo una representación clara y comprensible de los nodos y sus relaciones (Esa parte solo nos hace falta para esta apartado, es decir, para visualizar los nodos mejor).

Además de la creación de los nodos, es importante establecer atributos específicos para cada uno de ellos. Por ejemplo, se puede definir el radio (r) y el color de relleno (fill) de los círculos para mejorar su visualización y diferenciarlos unos de otros. En este caso, se asigna un radio de 20 unidades y un color de relleno gris claro (#ccc) a cada nodo, lo que garantiza que sean claramente distinguibles en el grafo.

Finalmente, se añaden etiquetas a los nodos para proporcionar información adicional sobre ellos. Esto se realiza mediante la creación de elementos de tipo título para cada nodo utilizando el método **.append('title')**, y estableciendo el texto del título como el nombre del nodo. Esta información emergente permite al usuario obtener detalles sobre cada nodo al

pasar el cursor sobre ellos, mejorando aún más la comprensión y la usabilidad del grafo en su conjunto.

Crear una simulación visual del grafo

La utilización de la simulación de fuerzas en D3 para organizar los nodos en el lienzo es esencial para garantizar una representación comprensible del grafo. La razón principal radica en que esta técnica permite que los nodos se dispongan de manera dinámica, teniendo en cuenta las relaciones entre ellos y evitando superposiciones que puedan dificultar la interpretación del grafo.

Al aplicar fuerzas como la de enlace, se establecen las conexiones entre los nodos de manera coherente, lo que ayuda a visualizar claramente cómo están relacionados unos con otros. Por otro lado, la fuerza de carga ayuda a mantener una distribución equilibrada de los nodos, evitando agrupamientos excesivos o dispersión incontrolada que puedan dificultar la comprensión del grafo.

Además, la fuerza de centrado contribuye a mantener el grafo centrado y equilibrado visualmente, lo que facilita la identificación de los nodos principales y la exploración del grafo en su conjunto.

Una vez completadas todas las etapas previas, se procede a ejecutar el **script** para generar y mostrar el grafo en la aplicación. Todo el proceso descrito anteriormente se encapsulará en una función, y luego se invocará el constructor del script para iniciar la creación del grafo. En C#, esto se realiza mediante el siguiente código:

```
ScriptManager.RegisterStartupScript(this.Page, Page.GetType(),
"CrearGrafo", script, true);
```

Algoritmo 5.1: Registro de Script en ASP.NET

En este fragmento de código, la variable graph contiene los datos necesarios para generar el grafo. Se utiliza 'string.Format' para construir una cadena de script que llame a la función **createGraph** del objeto grafo, pasando como parámetro los datos del grafo. Finalmente, ScriptManager.RegisterStartupScript se utiliza para registrar el script en la página, asegurando que se ejecute una vez que la página se haya cargado completamente.

Esta implementación en C# asegura que el grafo se genere y muestre adecuadamente en el entorno web. Ahora, con todo el proceso descrito y el código C# adecuadamente configurado, estamos listos para ejecutar nuestro script y visualizar el grafo resultante en la interfaz de usuario.

5.3.3.1. Generación de los nodos de forma dinámica

Ahora que se ha establecido con éxito los nodos en nuestro lienzo y confirmado su funcionalidad adecuada, es hora de dar el siguiente paso: integrar la información recopilada desde

el backend, implementando cada palabra como un nodo independiente en nuestro grafo.

Este avance implica una sinergia entre el backend y nuestro script en JavaScript. Desde el backend, en nuestro entorno de C#, se recopilan los datos esenciales, comenzando por el primer paso de dividir la oración en palabras e identificar claramente qué partes constituyen una palabra y cuáles no, esto se hace de la siguiente manera:

1. **Obtención de información relevante**: Antes de construir el grafo, se recopila toda la información esencial sobre la palabra actual. Esto incluye la palabra en sí misma, su dependencia (si tiene alguna), y su lema, que puede proporcionar información adicional sobre su significado o función gramatical.
2. **Determinación de la dependencia**: Si la palabra actual tiene una dependencia asociada, se identifica su posición en la lista de palabras de la oración completa. Esto nos permite establecer la conexión apropiada entre la palabra actual y su dependencia en el grafo.
3. **Creación del objeto Grafo**: Se hace uso de la información recopilada para crear un objeto 'Grafo' que represente visualmente la palabra actual y su relación con su dependencia. Este objeto Grafo contiene los siguientes elementos:
   - ✔ **Palabra actual**: Se representa como un nodo en el grafo, con su propia etiqueta y posición en el lienzo.
   - ✔ **Dependencia**: Si la palabra tiene una dependencia, se establece una conexión visual entre el nodo de la palabra actual y el nodo de su dependencia en el grafo.
   - ✔ **Tipo de lema:** Se guarda el tipo de lema asignado, de la palabra al guadar la dependencia, si no tiene, esta campo será nulo.
4. **Agregación al grafo final**: Una vez construido el objeto Grafo para la palabra actual, se agrega a una lista final de grafos. Esta lista contendrá todos los grafos generados para cada palabra de la oración, lo que nos permitirá visualizar la estructura completa de la oración en el grafo.
5. **Iteración sobre todas las palabras**: Se repite este proceso para cada palabra en la oración, asegurándonos de capturar todas las relaciones y construir un grafo completo que refleje con precisión la estructura de la oración.

Este enfoque es esencial para la creación del grafo, ya que nos permite visualizar las relaciones entre las palabras de la oración de manera dinámica. El grafo resultante no solo representa la estructura de la oración, sino que también sirve como una poderosa herramienta para explorar y comprender la gramática y el significado subyacentes.

Ahora nos enfrentamos al paso final: pasar el objeto Grafo a nuestro script. Para lograr esto, se tiene que Serializar el objeto antes de enviarlo. La serialización es crucial porque sin ella, el objeto Grafo no se puede transferir directamente entre diferentes entornos o lenguajes de programación. Para llevar a cabo esta tarea, realizamos la serialización del objeto Grafo de la siguiente manera:

```
var graph = new JavaScriptSerializer().Serialize(finalGrafo);
```

Algoritmo 5.2: Serialización del objeto finalGrafo

Este proceso de serialización convierte el objeto Grafo en una representación en formato de cadena que puede ser transmitida fácilmente. Es esencial realizar esta serialización porque permite transferir el objeto entre diferentes entornos o lenguajes de programación. Sin este paso, el objeto Grafo no podría ser transferido y utilizado en el script de manera efectiva.

Una vez que se ha serializado el objeto, se pasa el objeto serializado como parámetro a nuestra función en el script. Luego, se utiliza este conjunto de datos serializados para crear los nodos en el grafo, completando así la representación visual de las relaciones entre las palabras de la oración.

Después de completar exitosamente este proceso, los usuarios deberían poder ingresar cualquier oración en la aplicación y esperar que esta aparezca en el lienzo una vez que presionen el botón 'Dividir Oración'. Este hito marca un avance significativo, ya que una vez que el grafo se ha generado, se espera ver una representación visual dinámica y detallada de la estructura de la oración, completa con todas las características adicionales que se han implementado.

Al presionar el botón 'Dividir Oración', el sistema empieza a descomponer la oración en sus partes individuales, identificando cada palabra y las relaciones entre ellas. Este proceso incluye varios pasos, desde dividir la oración hasta crear un gráfico basado en los datos obtenidos.

Una vez que la oración se ha dividido con éxito y se ha construido el grafo correspondiente, se muestra el grafo en nuestro canvas. Esta representación no solo mostrará las palabras de la oración como nodos individuales, sino que también destacará las conexiones entre ellas mediante líneas o flechas, indicando las relaciones de dependencia o cualquier otra información relevante.

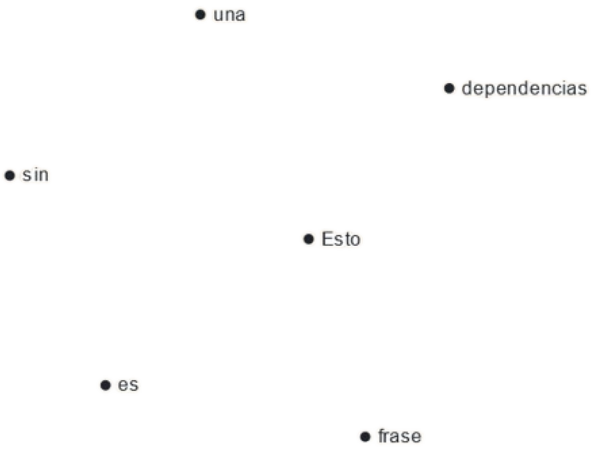

Ilustración 5.2: Representación visual de nodos descartados en el lienzo mediante simulación de fuerza

A continuación, se sigue con la implementación de las aristas (enlaces) de nuestro grafo siguiendo un enfoque progresivo. Primero, realizaremos una implementación estática de las aristas en los nodos para comprender mejor su funcionamiento. Luego, se evaluará cómo integrarlo en el proyecto, tomando en cuenta las dependencias también en el código en C#. Este enfoque gradual nos permite explorar y comprender el concepto de las aristas mientras avanzamos en la implementación de el proyecto.

Generación de las aristas de forma estática

Para entender cómo funcionan las aristas en el grafo actual, primero se regresará al código anterior donde se ha creado los nodos estáticos. Una forma efectiva de experimentar con las aristas es utilizando un conjunto de datos predefinido que defina las conexiones entre los nodos.

En este caso, se crea un dataset llamado **links**, que contiene información sobre las conexiones entre los nodos. Cada elemento de este dataset especifica el nodo de origen (source) y el nodo de destino (target) para cada conexión.

Una vez que se ha definido el dataset de conexiones, se procede a crear las aristas en el grafo. Se hace uso de la función **selectAll** para seleccionar todos los elementos line (líneas) en nuestro lienzo SVG. Luego, se vincula los datos del dataset links a estos elementos utilizando el método data, lo que nos permite asociar cada conexión en el dataset con una línea en el grafo.

A continuación, utilizando el método 'enter', se genera una línea para cada conexión en el dataset que no tiene una correspondencia en el lienzo SVG actual. Es decir, se crea nuevas líneas para representar cada conexión especificada en el dataset que aún no se ha dibujado en el grafo.

Este proceso permite a los desarrolladores experimentar con la creación de aristas en el grafo de manera controlada y predefinida, lo que contribuye a una comprensión más profunda de cómo funcionan y cómo pueden visualizarse las relaciones entre los nodos en el proyecto.

Al integrar estas aristas con el código existente de los nodos, se logra un paso crucial en la construcción del grafo. Este proceso implica la conexión visual de los nodos previamente establecidos mediante líneas que representan las relaciones entre ellos.

Al realizar esta integración, se obtiene una representación más completa y significativa de la estructura del grafo. Ahora, además de los nodos individuales que representan las palabras de la oración, se incluyen enlaces visuales que indican las conexiones y dependencias entre estas palabras.

Esta integración proporciona una representación más dinámica y comprensible de la información, lo que permite a los usuarios explorar y comprender mejor las relaciones y la estructura de la oración. Además, sienta una base sólida para futuras mejoras y funcionalidades, como la capacidad de personalizar y ajustar las conexiones entre los nodos según las necesidades específicas del usuario o del contexto del análisis lingüístico.

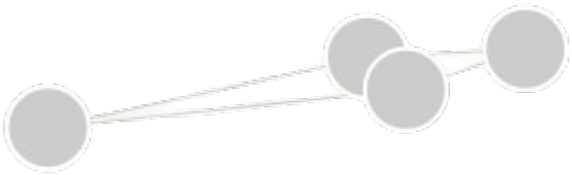

Ilustración 5.3: Representación del grafo con nodos conectados mediante enlaces generados

5.3.3.2. Generación de las aristas de forma dinámica

Al abordar la implementación de los nodos dinámicos, surge la necesidad de gestionar eficientemente los recursos y optimizar el proceso de recopilación de datos. En este contexto, surge una idea prometedora: aprovechar la misma fuente de datos utilizada para obtener la información de los nodos estáticos. Esto no solo simplifica el proceso de desarrollo, sino que también contribuye a mejorar la eficiencia del sistema y la coherencia de los datos.

Para materializar esta idea, se propone la incorporación de un nuevo atributo en el **constructor** de la clase Grafo, al que denominaremos 'target'. Este atributo se encargará de almacenar la dependencia de cada palabra, en caso de existir. Si no hay una dependencia asociada, el atributo 'target' se rellenará con una cadena vacía. Esta estrategia tiene varias ventajas que merece la pena explorar con más detalle.

En primer lugar, al utilizar la misma fuente de datos para obtener la información de los nodos estáticos y dinámicos, se evita la duplicación innecesaria de consultas o la carga de datos desde fuentes diferentes. Esto reduce la carga en el servidor y mejora significativamente el rendimiento del sistema, especialmente en escenarios donde el procesamiento de datos es intensivo.

Además, al mantener la coherencia en la fuente de datos, se garantiza la consistencia de la información recopilada para ambos tipos de nodos. Esto elimina cualquier posibilidad de inconsistencias que podrían surgir. Como resultado, se fortalece la integridad de los datos y se minimiza el riesgo de errores o malentendidos en la interpretación de la información.

Desde una perspectiva de diseño, la inclusión del atributo 'target' en el constructor de la clase Grafo simplifica enormemente la estructura del código y el proceso de recopilación de datos. Esto se traduce en una mayor legibilidad y mantenibilidad del código, lo que facilita la identificación y resolución de problemas, así como la incorporación de nuevas funcionalidades en el futuro.

En conclusión, la idea de utilizar un atributo adicional en la clase Grafo para almacenar las dependencias de los nodos dinámicos representa una estrategia inteligente y efectiva para mejorar la eficiencia, coherencia y simplicidad en el diseño del sistema. Esta aproximación contribuye en gran medida a mejorar la experiencia del usuario y la escalabilidad del proyecto a largo plazo.

En cuanto el nivel de código revela un proceso crucial en el manejo de datos dentro de tu aplicación. Para comprenderlo en su totalidad, es necesario profundizar en cada paso y entender su relevancia en el contexto general del desarrollo de software.

1. **Acceso al objeto 'palabraFormulario':** En esta etapa inicial, el código accede a un objeto denominado 'palabraFormulario'. Este objeto, encapsula información específica sobre una palabra dentro del contexto de la aplicación. Es esencial destacar que la utilización del operador '?', indica una precaución adicional: si el objeto 'palabraFormulario' o alguna de sus propiedades es nulo, no se producirá un error al intentar acceder a la propiedad siguiente. Esta estrategia de manejo de nulos es fundamental para mantener la estabilidad y la integridad del sistema.

2. **Exploración de la propiedad 'FrasePalabra':** Dentro del objeto 'palabraFormulario', se navega hacia la propiedad 'FrasePalabra'. Esta propiedad almacena información relevante sobre la palabra en cuestión, como su posición en una oración o su relación con otras palabras. La utilización de esta propiedad sugiere un enfoque estructurado y organizado para gestionar los datos de la aplicación, lo que facilita su manipulación y comprensión en el flujo de trabajo del desarrollo.

3. **Manejo de valores nulos mediante el operador '??'**: Una de las principales preocupaciones al trabajar con datos en un entorno de desarrollo es el tratamiento de valores nulos o indefinidos. El uso del operador '??' aborda esta preocupación al proporcionar un valor predeterminado en caso de que la propiedad 'Dependencia' sea nula.

4. **Verificación de la existencia de dependencia:** Una vez que se ha obtenido el valor de la dependencia, se realiza una verificación para determinar su validez. Esta verificación se lleva a cabo mediante la condición '!string.IsNullOrEmpty(dependencia)', que evalúa si la dependencia es nula o una cadena vacía. Esta precaución adicional garantiza que solo se procesen dependencias válidas y que se eviten posibles problemas causados por datos incompletos o incorrectos.

5. **Búsqueda del valor de la dependencia en un contexto más amplio:** La variable 'fraseCompletaFinal' representa un conjunto de datos más extenso que contiene información sobre una frase completa o una estructura similar. Utilizando el índice obtenido de la dependencia, se accede al valor correspondiente en este contexto más amplio. Este enfoque sugiere una integración de datos dentro de la aplicación, permitiendo que la información se utilice de manera coherente y eficiente en todo el sistema.

6. **Manejo de casos sin dependencia:** Finalmente, se aborda el escenario en el que no se encuentra una dependencia válida para la palabra actual. En este caso, se asigna una cadena vacía a la variable 'dependenciaValue', indicando que no hay ninguna dependencia asociada.

En resumen, este proceso de extracción de la dependencia de una palabra implica una serie de pasos diseñados para garantizar la integridad de los datos, así como para manejar de manera efectiva los casos en los que la dependencia puede estar ausente.

Una vez que se ha serializado el objeto y agregado el atributo adicional 'target' para representar la dependencia de cada palabra en nuestro grafo, no se necesita que realizar cambios significativos en la forma en que se pasa este objeto serializado al script. La inclusión del 'target' no altera la estructura del objeto serializado, ya que simplemente agrega información adicional a cada nodo del grafo.

Este nos proporciona una manera eficiente de manejar la transferencia de datos entre el backend y los scripts de nuestra aplicación. Al mantener la estructura del objeto serializado consistente, se puede aprovechar las funcionalidades existentes en nuestro script sin necesidad de realizar modificaciones adicionales.

Además, este método nos permite mantener la conexión de los datos a lo largo de todo el proceso. Al no requerir cambios significativos en la forma en que se pasan los datos al script, se reduce la complejidad y el riesgo de introducir errores.

Ahora que se ha completado la implementación de las aristas en el grafo, es momento de considerar cómo se puede mejorar su apariencia y funcionalidad. Una opción que se ha explorado es la adición de colores a las aristas. La asignación de colores puede ayudar a visualizar mejor las relaciones entre los nodos, destacando conexiones específicas o patrones en el grafo. Por ejemplo, se podrían usar colores diferentes para representar diferentes tipos de relaciones lingüísticas, como sujetos, objetos o modificadores, lo que facilitaría la interpretación del grafo.

Otra posibilidad que se ha considerado es la inclusión de flechas, para indicar la dirección de las relaciones entre las palabras. Esto sería especialmente útil en casos donde las relaciones tienen una dirección clara, como en el caso de los verbos y sus objetos. Al agregar flechas que apunten desde el origen al destino de cada relación, los usuarios podrían identificar fácilmente la secuencia y el flujo de la información en el grafo.

Ambas opciones ofrecen formas interesantes de mejorar la legibilidad y la interpretación del grafo, pero también plantean desafíos adicionales en términos de implementación y diseño. Por ejemplo, seleccionar una paleta de colores efectiva que sea accesible para todos los usuarios y que no interfiera con la comprensión del contenido puede ser un proceso complejo.

Facilitando la interpretación con flechas direccionales en el grafo

Este proceso es crucial para la claridad que se necesita en el grafo. Después de explorar algunas alternativas, se ha llegado a la conclusión de que se puede implementarlo de manera relativamente sencilla. Para lograrlo, se necesita ajustar el código para Renderizar las aristas de una manera diferente, lo que implica el uso del atributo `marker-end` para dibujar flechas al final de cada arista.

En el fragmento de código proporcionado, se crea una selección de elementos `'path'` dentro de un grupo específico destinado a las aristas (`linkGroup`). Posteriormente, se vincula los datos de nuestras aristas (`links`) con estos elementos. Además, la función `join()` garantiza que los datos y los elementos DOM estén correctamente relacionados.

Cada arista se define como un elemento `'path'`, y se ha configurado diversas propiedades para su apariencia y comportamiento:

- ✔ `fill: none`: Esto indica que las aristas no tendrán relleno, ya que representan líneas.
- ✔ `stroke-width: 1.5`: Establece el grosor de la línea de la arista para que sea visible pero no molesta.
- ✔ `marker-end`: Este es el atributo clave que agrega una flecha al final de cada arista. Se utiliza una función de URL para referenciar el marcador definido en el archivo SVG. La construcción `#arrow-${d.type}` asegura que cada flecha se ajuste al tipo específico de relación representada por la arista.

En conclusión, al utilizar el atributo `marker-end` con una función de URL en el código, se puede mejorar significativamente la visualización del grafo, añadiendo flechas direccionales que indican la orientación de las relaciones entre los nodos. Este enfoque aumenta la comprensión del grafo al proporcionar indicadores visuales claros de la dirección del flujo de información en la estructura del grafo.

5.3.4. Mejora de la legibilidad mediante la asignación de colores en las aristas del grafo

Una forma efectiva de mejorar la interpretación del grafo es asignar colores distintivos a cada tipo de dependencia. Esto ayudará a los usuarios a identificar rápidamente las relaciones específicas entre los nodos. Para lograr esto, se puede aprovechar las capacidades de estilo de SVG en D3.js.

En primer lugar, se ha definido una escala de colores utilizando la función **d3.scaleOrdinal()**. Esta función nos permite asignar un color único a cada tipo de dependencia en función de su identificador único. En nuestro caso, se ha utilizado la escala de colores **d3.schemeCategory10**, que proporciona una paleta de colores predefinida con un total de 10 colores distintos.

Una vez que se obtiene la escala de colores, se aplica a las aristas en el grafo. Se ha hecho uso del atributo **stroke** en el código SVG para establecer el color de cada arista en función del tipo de dependencia que representa. Para hacer esto, se pasa el tipo de dependencia (d.type) a nuestra función de escala de colores, que devuelve el color correspondiente para ese tipo de dependencia.

Al hacerlo, existen varios tipos de dependencia, dependiendo de su lema, se representará con un color distinto en el grafo, lo que facilitará su identificación. Los usuarios podrán distinguir fácilmente entre diferentes tipos de relaciones en el grafo basándose en los colores de las aristas. Esto mejorará significativamente la capacidad del grafo para comunicar información.

5.3.5. Mejora del renderizado de las aristas con curvas

Una mejora significativa que se ha implementado en nuestro grafo es la renderización de las aristas utilizando curvas en lugar de líneas rectas. Esto no solo mejora la estética visual del grafo, sino que también ayuda a evitar que las aristas se superpongan y mejora la legibilidad del mismo.

Para lograr esto, se ha definido una función llamada **linkArc**, que calcula la trayectoria de una curva para cada arista en el grafo. La función toma como entrada el objeto de datos d, que representa una conexión entre dos nodos en el grafo.

El primer paso en el cálculo de la trayectoria de la curva es determinar la diferencia en las coordenadas x e y entre el nodo de origen y el nodo de destino. Esto nos da la dirección y la magnitud del vector que representa la arista entre los dos nodos.

Luego, se calcula el ángulo de la arista utilizando la función **Math.atan2**, que nos da el ángulo en radianes entre el vector de la arista y el eje x.

Después de calcular el ángulo, se encuentra el punto medio entre los nodos de origen y destino. Esto nos da el punto central alrededor del cual la curva se arqueará.

A continuación, se ha calculado un desplazamiento perpendicular al vector de la arista para determinar la curvatura de la curva. Esto nos permite controlar la forma y la dirección de la curva, ajustando el desplazamiento según sea necesario para evitar superposiciones y garantizar una apariencia estéticamente agradable.

Finalmente, se ha construido la trayectoria de la curva utilizando los puntos de origen, destino y el punto medio desplazado. Esto nos da una representación suave y curva de la arista en el grafo, que mejora la legibilidad y la estética general del mismo.

Al implementar esta función, se ha logrado una mejora significativa en la visualización de las aristas en nuestro grafo, lo que facilita la interpretación y comprensión de las relaciones entre los nodos.

5.3.6. Mejora de la legibilidad de grafos mediante la optimización de distancias entre nodos

Una parte importante del diseño de un grafo efectivo es garantizar que la distancia entre los nodos sea adecuada para facilitar la interpretación y evitar superposiciones. La función **calculateDistance** aborda este desafío al calcular la distancia euclidiana entre los nodos de origen y destino de una arista y asegurarse de que esta distancia cumpla con un valor mínimo predefinido.

La función toma como entrada el objeto de datos d, que representa una conexión entre dos nodos en el grafo. Utiliza las coordenadas x e y de los nodos de origen y destino para calcular la diferencia en las coordenadas en ambas dimensiones, lo que nos da el vector de la arista entre los dos nodos.

Luego, se calcula la distancia euclidiana utilizando el teorema de **Pitágoras**, que nos da como resultado la longitud del vector de la arista. Esta distancia es crucial para determinar cómo se renderizará la arista entre los dos nodos en el grafo.

Sin embargo, es posible que la distancia euclidiana resultante sea insuficiente para garantizar una representación visual efectiva en el grafo. Por lo tanto, la función aplica un valor mínimo predefinido (minDistance) para garantizar que la distancia entre los nodos sea al menos igual a este valor. Esto ayuda a evitar superposiciones y garantiza que las aristas sean claramente visibles y distintivas en el grafo.

Al asegurar que la distancia entre los nodos cumpla con un valor mínimo establecido, la función llamada **calculateDistance**, contribuye significativamente a la claridad y la legibilidad del grafo, lo que facilita la interpretación de las relaciones entre los nodos y mejora la experiencia del usuario al interactuar con el grafo.

Después de abordar todas las etapas mencionadas, se ha alcanzado un primer prototipo funcional. Aunque aún no es perfecto, se ha logrado establecer la capacidad de crear nodos y aristas de manera eficiente, asignando a cada arista un tipo y color específico. A continuación, se muestra un ejemplo del resultado obtenido hasta ahora.

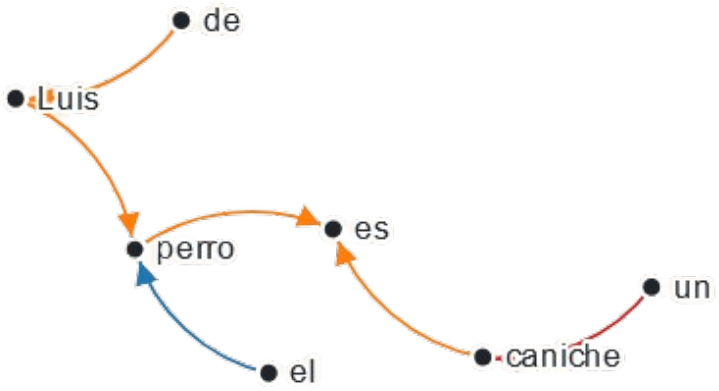

Ilustración 5.4: Primera versión funcional del grafo

5.3.7. Mejora de la interfaz del usuario

Una mejora significativa sería permitir al usuario cambiar el orden de las palabras en la oración. Esto implica proporcionar una funcionalidad que permita al usuario indicar explícitamente el orden correcto de las palabras. Implementar esta característica ayudará a clarificar la estructura de la oración, facilitando una representación más precisa y comprensible del grafo. Al permitir la reordenación, los usuarios podrán ajustar el grafo para reflejar mejor la sintaxis y semántica de la oración original, mejorando así la interpretación y el análisis lingüístico.

Para mejorar la visualización, se ha decidido al final, ordenar los nodos horizontalmente. Sin embargo, el usuario no podrá elegir el orden de las palabras, lo cual podría ser un buen reto para enfrentarnos en el futuro.

5.3.8. Creación de nodos y enlaces en el grafo

Para generar el grafo, primero se identifican y se generan los nodos. Se empieza extrayendo todas las palabras que actúan como origen (*source*) o destino (*target*) en nuestras relaciones. Luego, se elimina cualquier duplicado para estar seguros de que cada palabra se represente solo una vez en el grafo. Además, se descarta las entradas vacías para mantener la integridad de los datos.

Se asigna a cada nodo una posición inicial en el eje X, espaciándolos uniformemente para una distribución clara. En el eje Y, se posiciona los nodos en el centro, desplazando ligeramente hacia abajo aquellos que actúan como origen en alguna relación. Este enfoque asegura una disposición inicial ordenada y fácil de seguir.

Posteriormente, se crea los enlaces que representan las relaciones entre las palabras. Se filtran las relaciones para incluir solo aquellas que tienen un destino definido, y luego se generan copias de estas relaciones para evitar modificar los datos originales. Esto asegura que cada enlace esté claramente definido y preparado para la simulación.

```
const nodes = Array.from(new Set(palabras.flatMap(d => [d.
   source, d.target])))
    .filter(id => id !== "")
    .map((id, index) => ({ id, x: index * 100, y: height / 2
       + (palabras.some(d => d.source === id) ? 50 : 0) }));
const links = palabras
    .filter(d => d.target)
    .map(d => Object.create(d));
```

Algoritmo 5.3: Creación de Nodos y Enlaces para Visualización de Palabras

Al distribuir los nodos horizontalmente, es posible enfrentarse a problemas relacionados con el solapamiento de nodos, especialmente en oraciones largas o con muchas palabras estrechamente conectadas. Además, al modificar solo el eje Y, se podría perder parte de la claridad que proporciona la distribución natural de la simulación de fuerzas, donde los nodos se posicionan de manera que minimizan las colisiones y maximizan la legibilidad. Otro posible inconveniente es la pérdida de la capacidad del grafo para adaptarse dinámicamente a cambios en la estructura de la oración, ya que una disposición estrictamente horizontal puede ser menos flexible frente a variaciones en las conexiones y dependencias de las palabras.

5.3.9. Simulación de fuerzas en el grafo

Para distribuir los nodos de manera efectiva, se hace uso de una simulación de fuerzas. Esta simulación aplica diferentes fuerzas a los nodos y enlaces para determinar sus posiciones finales de manera dinámica, lo cual para que se generan los nodos en el orden adecuado, tenemos que realizar algunas modificaciones en él.

1. **Fuerza de enlace:** Conecta los nodos según las relaciones definidas. Asegura que los nodos relacionados se mantengan unidos, reflejando las dependencias y conexiones de las palabras en la oración.
2. **Fuerza de carga:** Aplica una repulsión entre todos los nodos para evitar que se superpongan. En este caso, ajustamos la fuerza de repulsión a cero para mantener los nodos en sus posiciones iniciales sin repulsión adicional.
3. **Fuerza de centro:** Centra el grafo en el punto de origen (0,0), asegurando que el grafo esté bien posicionado en el lienzo de visualización.

Cálculo de distancias

Para mejorar la visualización, se ha implementado una función que calcula la distancia entre nodos conectados. Esta función utiliza las posiciones actuales de los nodos para determinar la distancia entre ellos. Al calcular la distancia, se garantiza que los nodos no se superpongan y que las aristas que los conectan sean claras y fáciles de seguir.

A continuación, se muestra una imagen con el resultado obtenido de haber implementado las siguientes funciones

Ilustración 5.5: Resultado obtenido tras implantar un cálculo de distancias

Se puede observar que en la imagen el resultado obtenido presenta ciertos problemas de claridad y legibilidad. Las aristas en el grafo, aunque correctamente conectadas, crean un desorden visual que dificulta la interpretación completa de la oración. Además, el orden de las palabras no se mantiene correctamente, lo que complica aún más la comprensión del grafo. Este desorden es en gran medida consecuencia de la disposición del campo de fuerzas aplicado a los nodos.

Para mejorar la visualización del grafo y asegurar que las conexiones entre las palabras sean claras y fáciles de seguir, es necesario implementar nuevas funciones y algoritmos que

optimicen tanto la disposición de los nodos como la de las aristas. Estos cambios permitirán una representación más estructurada y accesible del grafo, facilitando la comprensión de la oración y sus relaciones gramaticales.

5.3.10. Reordenación de las palabras en el grafo

Al inspeccionar el grafo generado, surge una discrepancia notable: el orden de las palabras en el grafo no se corresponde con la secuencia original de la oración. Este desajuste plantea un desafío significativo en términos de legibilidad y comprensión, ya que la disposición incorrecta de las palabras puede dificultar la interpretación precisa del mensaje original. Inicialmente, se ha planteado que este problema se debía a la simulación de fuerzas aplicada a los nodos durante la generación del grafo. Sin embargo, tras un análisis más detenido, se ha identificado que la raíz del problema radica en la posición inicial asignada a los nodos al recuperarlos del objeto. Por lo tanto, para abordar esta discrepancia y garantizar una representación visual precisa de la oración, se debe ajustar nuestra simulación para que los nodos se distribuyan en el orden correcto. Este ajuste es crucial para preservar la coherencia y la estructura semántica de la oración original en el grafo generado. Al corregir esta discrepancia, se mejora significativamente la claridad y la interpretación del grafo, lo que facilitará su uso en el análisis lingüístico y la comprensión del texto.

De esta manera es como se generan los nodos ahora, teniendo en cuenta que la variable 'palabras' son nuestras palabras ya recuperado del objeto inicial dado, ya filtrado por nodos únicos:

```
    const nodes = Array.from(uniqueNodes).map((id, index) =>
       ({
     id,
     x: width / 2 - (uniqueNodes.size - 1) * nodeSpacing / 2 +
         index * nodeSpacing,
     y: height / 2 // Initialize Y position for all nodes
}));
```

Algoritmo 5.4: Generación de los nodos

En esta sección del código, se ha estableciendo la posición inicial de cada nodo para lograr una distribución horizontal y ordenada en el grafo resultante. Se oberva en detalle cómo se ha logrando:

1. **Creación de nodos únicos:** Se empieza creando un conjunto de nodos únicos a partir de un conjunto más grande de nodos. Esto nos asegura que cada nodo aparezca solo una vez en el conjunto, lo que evita duplicados y simplifica el proceso de posicionamiento.
2. **Asignación de posiciones en el eje X:** Para cada nodo en el conjunto único, se calcula su posición en el eje X de manera que estén distribuidos de manera uniforme horizontalmente a lo largo del lienzo. Para lograr esto, se ha hecho uso de una fórmula que tiene en cuenta el ancho total del lienzo, el espaciado deseado entre los nodos y el índice del nodo en el conjunto. Esto nos permite posicionar cada nodo de manera equidistante, asegurando que estén correctamente alineados horizontalmente.

3. **Inicialización de la posición en el eje Y:** Además de establecer la posición en el eje X, también se inicializa la posición en el eje Y de todos los nodos en el centro vertical del lienzo. Esto garantiza que los nodos estén alineados horizontalmente y centrados verticalmente en el lienzo, lo que proporciona una apariencia ordenada y equilibrada al grafo resultante.

En resumen, esta parte del código asegura que los nodos se posicionen de manera uniforme y ordenada en el eje horizontal del lienzo, lo que contribuye a una representación visual clara y coherente de la estructura de la oración en el grafo generado.

Después de revisar el grafo generado, se ha identificado que aún no es suficientemente claro y no proporciona una representación visual completa de la estructura de la oración. Una de las deficiencias observadas es que el grafo parece estar demasiado plano, con los nodos distribuidos únicamente a lo largo del eje horizontal. Esto dificulta la percepción de las relaciones jerárquicas y la comprensión de la profundidad del grafo.

Para abordar esta limitación y mejorar la claridad visual del grafo, se ha ideado una estrategia para darle volumen. La idea consiste en mover los nodos que son dependientes de otros un poco más arriba en el lienzo, de manera que se cree una disposición tridimensional que refleje mejor la estructura de la oración. Al desplazar estos nodos, se pretende resaltar visualmente las relaciones de dependencia entre las palabras y facilitar la interpretación del grafo en su conjunto.

El objetivo de esta técnica es proporcionar una representación más realista y significativa de la sintaxis de la oración, donde las palabras que actúan como modificadores o complementos se posicionan visualmente por encima de las palabras a las que están relacionadas. Esto ayudará a los usuarios a percibir con mayor claridad las jerarquías gramaticales y las relaciones sintácticas en el texto analizado.

Al mismo tiempo, es importante tener en cuenta que se debe mantener la estructura general del grafo para no comprometer la legibilidad y la comprensión. Por lo tanto, el desplazamiento de los nodos se realizará de manera cuidadosa y equilibrada, evitando cambios drásticos que puedan confundir al usuario o distorsionar la representación del texto original.

En resumen, la adopción de esta estrategia nos permitirá enriquecer la visualización del grafo y proporcionar una representación más fiel y completa de la estructura de la oración. Al incorporar un elemento tridimensional, se espera mejorar significativamente la experiencia del usuario y facilitar el análisis lingüístico del texto.

A continuación, se avanza al tramo final donde se afinan los últimos detalles respecto al desarrollo del grafo.

5.3.11. Ajuste de posiciones de nodos según destino asignado

Este apartado se encarga de actualizar las posiciones en el eje Y de los nodos en función de la asignación de destino. Básicamente, recorre cada elemento en el arreglo palabras, que

contiene información sobre las conexiones entre nodos, y ajusta la posición vertical de los nodos basándose en su asignación de destino.

Primero, se verifica si el nodo actual tiene un destino asignado distinto de una cadena vacía. Si es así, se busca el nodo fuente correspondiente en el arreglo 'nodes', que almacena todos los nodos del grafo. Luego, comprueba si se encuentra el nodo fuente y procede a ajustar su posición en el eje Y.

Para realizar el ajuste, encuentra el nodo destino correspondiente en el arreglo nodes y, si se encuentra, actualiza la posición del nodo fuente en el eje Y. La posición Y del nodo fuente se establece como la posición Y del nodo destino menos un desplazamiento (offset) específico. Esto implica mover el nodo fuente hacia arriba en relación con el nodo destino, creando así un efecto de volumen en el grafo.

Finalmente, se ajusta la posición en el eje X del nodo fuente para mantener una distribución uniforme de los nodos. Este ajuste se realiza añadiendo un espaciado fijo **(nodeSpacing)** a la posición X del nodo destino.

Este proceso permite visualizar claramente la estructura del grafo, dando la sensación de volumen al posicionar los nodos de manera que reflejen las relaciones jerárquicas entre ellos, sin necesidad de modificar las relaciones de dependencia existentes.

Representación visual de enlaces curvos en un grafo

La función **linkArc** es esencial en la visualización de enlaces curvos entre nodos en un grafo. Esta función calcula la curvatura de los enlaces entre los nodos, lo que permite representar visualmente las conexiones de manera más clara y estéticamente agradable.

Al utilizar esta función, se puede generar trayectorias curvas que conectan los nodos de origen y destino en el grafo. Este visualmente es bastante mejor y genera una buena estética para el grafo.

La función mencionada anteriormente, calcula la posición de los puntos de control necesarios para crear una curva suave entre dos nodos. Utiliza la posición de los nodos de origen y destino, así como un desplazamiento perpendicular para determinar la curvatura de la línea. Esto asegura que los enlaces sean lo suficientemente curvos como para evitar cruces con otros elementos del grafo, pero no tan pronunciados como para parecer poco naturales o difíciles de seguir.

En resumen, la función linkArc es una herramienta fundamental en la representación de enlaces curvos en un grafo, lo que mejora significativamente la legibilidad y la estética de la visualización.

Representación del resultado obtenido

Después de revisar y aplicar todas las mejoras mencionadas anteriormente, se ha logrado un resultado bastante viable. Durante el proceso, se trabajó para abordar cada aspecto identificado y se observó un progreso considerable en la calidad y la funcionalidad del proyecto. Aunque se reconoce que aún existe margen para realizar más mejoras y refinamientos, se está satisfecho con el resultado obtenido hasta ahora. Con este resultado final, se puede ver claramente el progreso que se ha logrado y la dirección en la que se quiere seguir avanzando. Se está comprometido a seguir perfeccionando este proyecto para ofrecer la mejor experiencia posible a los usuarios. A continuación, se adjunta una imagen que muestra el estado actual del proyecto, reflejando el trabajo y la dedicación hasta el momento.

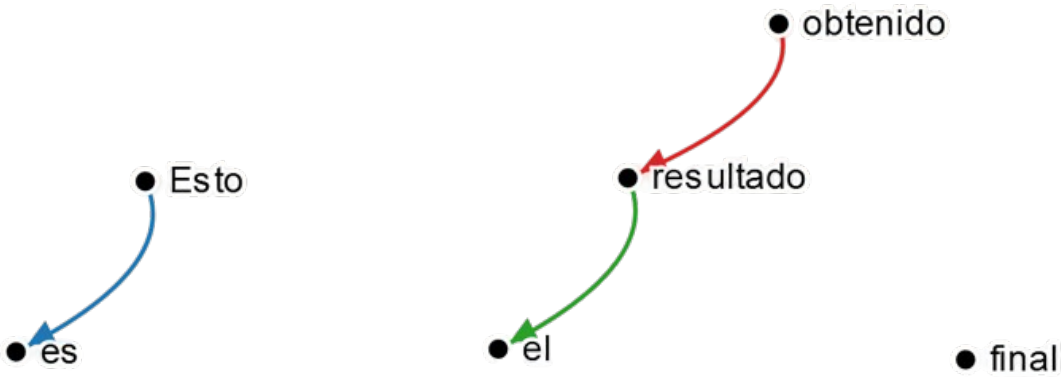

Ilustración 5.6: Resultado final obtenido

Capítulo 6

Conclusiones y trabajo futuro

6.1. Evaluación de resultados

Durante el curso del proyecto de implementación de grafo y dependencia, se ha alcanzado una serie de resultados significativos que han superado nuestras expectativas iniciales. En primer lugar, la eficiencia en la representación de dependencias se ha mejorado de manera notable gracias a la implementación del grafo. Esta herramienta nos ha permitido visualizar de manera clara y concisa las complejas relaciones entre los diferentes elementos del sistema, lo que ha facilitado en gran medida tanto la comprensión de la estructura global como la identificación de posibles puntos críticos.

Además, los algoritmos diseñados para la representación gráfica y la identificación de dependencias críticas han destacado por su precisión excepcional. Su capacidad para detectar todos los ciclos dentro del grafo ha sido fundamental para tomar medidas preventivas y evitar posibles fallos sistémicos. Además, la identificación de dependencias críticas ha permitido una asignación de recursos más eficiente, lo que se ha traducido en una mejora significativa del rendimiento del sistema. Es importante resaltar que el proyecto se completó dentro del plazo establecido y con los recursos disponibles, demostrando una gestión eficaz del tiempo y los recursos asignados.

Otro aspecto notable de los resultados obtenidos es la optimización de recursos lograda mediante la aplicación de la herramienta desarrollada. La asignación más racional y efectiva de los recursos no solo ha permitido eliminar redundancias y minimizar el riesgo de sobrecarga, sino que también ha generado un ahorro de costos significativo. Además, esta optimización ha contribuido de manera sustancial a mejorar la estabilidad y la fiabilidad del sistema.

Es crucial destacar que este logro fue alcanzado considerando las limitaciones y requisitos específicos de los servidores actuales del proyecto. Se ha adaptado a estas circunstancias, lo que implicó realizar ajustes y optimizaciones en el código para garantizar una integración sin problemas con la infraestructura existente. Este enfoque nos permitió no solo alcanzar nuestros objetivos de optimización de recursos, sino también asegurar la coherencia y la compatibilidad con el entorno operativo actual del proyecto.

Por último, la escalabilidad y flexibilidad demostradas por la solución implementada son dignas de mención. Se ha sido capaces de manejar sistemas de creciente complejidad sin experimentar una degradación significativa en el rendimiento, lo que confirma la robustez del modelo desarrollado. Además, la flexibilidad del modelo nos ha permitido realizar adaptaciones rápidas y efectivas en respuesta a cambios en las dependencias del sistema, lo que garantiza su relevancia y utilidad a largo plazo.

6.2. Grado de consecución de los objetivos

Durante el desarrollo del proyecto, se ha logrado un alto grado de consecución de los objetivos planteados, cada uno de los cuales se detalla a continuación:

1. **Explorar las Capacidades de ASP.NET:** Se ha realizado una exhaustiva exploración de las capacidades ofrecidas por la plataforma ASP.NET. Se ha adquirido un profundo conocimiento de los diferentes componentes y características de ASP.NET y se han aplicado de manera efectiva en el desarrollo de la aplicación web.

2. **Desarrollar Sistemas y Algoritmos Nuevos:** Se han desarrollado sistemas y algoritmos innovadores para la visualización efectiva de las relaciones entre palabras en una oración. Se han diseñado y puesto en práctica algoritmos que analizan la estructura sintáctica y semántica de las oraciones con resultados satisfactorios.

3. **Crear una Experiencia de Usuario Intuitiva:** Se ha logrado crear una experiencia de usuario intuitiva y atractiva que permite a los usuarios explorar las relaciones entre palabras de manera efectiva. La interfaz de usuario es clara, fácil de usar y ofrece funcionalidades de interacción dinámica con el grafo.

4. **Modificar una Base de Datos para Almacenar Relaciones:** Se ha modificado con éxito una base de datos para almacenar las relaciones entre palabras. La estructura de la base de datos es escalable y capaz de manejar grandes volúmenes de datos.

5. **Potenciar el Valor Educativo y Analítico:** La aplicación ha logrado potenciar el valor educativo y analítico al ofrecer herramientas que ayudan a los usuarios a comprender la estructura y el significado de las oraciones, así como a profundizar en conceptos de gramática y análisis lingüístico.

6. **Implementar Funcionalidades de Colaboración:** Se han implementado funcionalidades que permiten la colaboración entre usuarios, enriqueciendo la experiencia y el valor educativo de la aplicación. Los usuarios pueden compartir y comentar análisis lingüísticos, así como colaborar en proyectos de investigación lingüística en tiempo real.

7. **Incorporar Capacidades de Análisis Avanzado:** Se han incorporado capacidades de análisis avanzado que permiten a los usuarios profundizar en el estudio del lenguaje humano. Se han integrado herramientas de análisis de sentimientos, reconocimiento de entidades nombradas y extracción de información, así como algoritmos de aprendizaje automático para mejorar la precisión y eficacia de los análisis lingüísticos.

En conclusión, todos los objetivos propuestos han sido alcanzados con éxito, teniendo en cuenta que siempre hay margen de mejora, lo que demuestra el alto grado de consecución del proyecto y su impacto positivo.

6.3. Posibles extensiones del proyecto

Considerando el éxito alcanzado en la consecución de los objetivos planteados y el impacto positivo del proyecto, se identifican varias áreas para posibles extensiones y mejoras futuras:

1. **Desarrollo de funcionalidades avanzadas:** Se podría explorar la incorporación de funcionalidades más avanzadas en la aplicación, como la detección de errores gramaticales y sugerencias de corrección, o la generación automática de resúmenes a partir de textos largos.
2. **Adaptación a otros idiomas:** La aplicación se podría centrar en el procesamiento del lenguaje natural en español, y sería interesante adaptarla para trabajar con otros idiomas, lo que ampliaría su alcance y utilidad en contextos multilingües.
3. **Integración con plataformas de aprendizaje en línea:** Se podría explorar la integración de la aplicación con plataformas de aprendizaje en línea, como Moodle o Coursera, para ofrecer herramientas adicionales a estudiantes y educadores en entornos educativos virtuales.
4. **Expansión de las capacidades de colaboración:** Se podría ampliar las funcionalidades de colaboración de la aplicación, permitiendo a los usuarios colaborar en la creación y edición de documentos lingüísticos de forma colaborativa en tiempo real.
5. **Mejora de la escalabilidad y el rendimiento:** Se podría trabajar en mejorar la escalabilidad y el rendimiento de la aplicación, especialmente en lo que respecta al manejo de grandes volúmenes de datos y al procesamiento en tiempo real de solicitudes de usuarios.
6. **Evaluación y retroalimentación continua:** Se podría establecer un proceso de evaluación y retroalimentación continua con los usuarios para identificar áreas de mejora y priorizar futuras actualizaciones y desarrollos en función de las necesidades y preferencias de la comunidad de usuarios.

Estas posibles extensiones representan oportunidades emocionantes para continuar construyendo y mejorando la aplicación en el futuro, garantizando su relevancia y utilidad en un entorno en constante evolución.

Bibliografía

[1] Bostock, M., Ogievetsky, V., y Heer, J. (2011). D3.js - Data-Driven Documents. [En línea]. Disponible en: https://d3js.org/ [Consulta: 05 de marzo de 2024]. Repositorio GitHub: https://github.com/d3/d3. Galería ObservableHQ: https://observablehq.com/@d3/gallery.

[2] ObservableHQ. Galería de D3.js en Observable. [En línea]. Disponible en: https://observablehq.com/@d3/gallery [Consulta: 22 de marzo de 2024].

[3] Stack Overflow. Comunidad en línea de desarrolladores. [En línea]. Disponible en: https://es.stackoverflow.com/ [Consulta: 7 de abril de 2024].

[4] Microsoft Corporation. Documentación oficial de Microsoft. [En línea]. Disponible en: https://docs.microsoft.com/es-es/ [Consulta: 22 de marzo de 2024].

[5] Microsoft Corporation. ASP.NET - Framework para aplicaciones web. [En línea]. Disponible en: https://dotnet.microsoft.com/apps/aspnet [Consulta: 21 de marzo de 2024].

[6] Lamport, L. (1994). *LaTeX: Una herramienta para la preparación de documentos* (2ª ed.). Addison-Wesley Iberoamericana.

Glosario

API es una pieza de código que permite a diferentes aplicaciones comunicarse entre sí y compartir información y funcionalidades.. 17

Canvas es un elemento HTML que permite la generación de gráficos y animaciones de forma dinámica por medio de scripts tras elegir el dominio para tu web.. 25

DropDownList representa un control que permite al usuario seleccionar un único elemento de una lista desplegable.. 21

NuGet es una extensión de Visual Studio que hace más fácil agregar, eliminar y actualizar referencias a librerías y herramientas en proyectos de Visual Studio que utilizan . NET Framework.. 20

Reader es una clase en ADO.NET que proporciona una forma rápida y eficiente de leer datos de una base de datos SQL de solo lectura y en avance hacia adelante.. 24

Renderizar se define como aquel proceso informático que permite visualizar la página en la pantalla. 34

Repeater hace un control de lista enlazado a datos que permite un diseño personalizado mediante la repetición de una plantilla especificada para cada elemento que se muestra en la lista.. 23

Serializar es el proceso de convertir el estado de un objeto en un formato que se pueda almacenar o transportar. 28

SVG significa Scalable Vector Graphics, y es un formato de archivo único que escala sin pérdida de calidad de imagen. 25