ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

DESARROLLO DE VISUALIZACIONES WEB INTERACTIVAS PARA DISTINGUIR LA TENDENCIA DEL DISCURSO POLÍTICO DE LOS CANDIDATOS PRESIDENCIALES ECUATORIANOS.

PROYECTO PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN ELECTRÓNICA Y REDES DE INFORMACIÓN

DARÍO SEBASTIÁN HERRERA MONTERO
dario.herrera02@epn.edu.ec

DIRECTOR: DR. LUIS FELIPE URQUIZA AGUIAR
luis.urquiza@epn.edu.ec

Quito, 2022

AVAL

Certifico que el presente trabajo fue desarrollado por Darío Sebastián Herrera Montero, bajo mi supervisión.

PhD. LUIS FELIPE URQUIZA AGUIAR
DIRECTOR DEL TRABAJO DE TITULACIÓN

DECLARACIÓN DE AUTORÍA

Yo Darío Sebastián Herrera Montero, declaro bajo juramento que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración dejo constancia de que la Escuela Politécnica Nacional podrá hacer uso del presente trabajo según los términos estipulados en la Ley, Reglamentos y Normas vigentes.

DARÍO SEBASTIÁN HERRERA MONTERO

DEDICATORIA

A mis compañeros y amigos que fueron un participes de cada uno de estos años aportando su granito de arena a esta etapa de mi vida.

Al Dr. Luis Felipe Urquiza que tanto como docente de distintas asignaturas y como director de este trabajo de titulación ha sabido guiarme y apoyarme en cada circunstancia presentada.

Y principalmente a mi familia, a mis hermanos Carlos y Zohely que desde su desconocimiento de varias cosas fueron un apoyo emocional gigante cuando las cosas pintaban complicadas, y a mis padres José y Rocío mi infinito agradecimiento y mi deuda eterna hacia ellos por apoyarme, guiarme y sobre todo darme todo su cariño y amor en cada día de mi vida, sin ellos no sería ni una décima parte de lo que hoy soy como persona.

AGRADECIMIENTO

A Dios, gracias por permitirme vivir y alcanzar la meta propuesta.

A mis padres y hermanos, quienes supieron ser mi guía y mi apoyo en todas las circunstancias de mi vida.

A mis profesores, y en especial al Dr. Luis Urquiza, por su paciencia, compromiso y apertura a apoyarme durante el desarrollo de este proyecto de titulación.

A mis familiares y amigos, por todo el apoyo y confianza que me brindan.

ÍNDICE DE CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE CÓDIGOS

ÍNDICE DE TABLAS

RESUMEN

Este Proyecto de Titulación se enfoca en el desarrollo e implementación de tres visualizaciones web interactivas que permitirán distinguir la tendencia del discurso político de los candidatos presidenciales ecuatorianos utilizando los datos extraídos de los tweets de las cuentas oficiales de los presidenciables pertenecientes al proceso electoral 2020-2021.

La estructura de los capítulos que conforman la documentación del Proyecto se presenta a continuación.

En el Capítulo 1 se formula el problema, se delimita su alcance y se presenta la justificación de este, además de detallar tanto el objetivo general como los objetivos específicos.

En el Capítulo 2 se estudia los conceptos relacionados a la visualización de datos como: las etapas que corresponden al desarrollo de visualizaciones, la definición de mapeo de datos a objetos visuales, los tipos de datos que se usan para realizar distintas visualizaciones, entre otros.

En el Capítulo 3 se detalla el desarrollo y la implementación de las tres visualizaciones, haciendo énfasis en el mapeo realizado correspondiente a cada una de ellas.

En el Capítulo 4 se presentan las pruebas de funcionalidad y de usuario realizadas a las visualizaciones y a la interfaz web en la que están alojadas.

En el Capítulo 5 se exponen las conclusiones y recomendaciones a las cuales se llegó una vez finalizado el proyecto.

PALABRAS CLAVE: Visualizaciones, Política, d3.js, Twitter.

ABSTRACT

This project focuses on developing and implementing three interactive web visualizations that will allow distinguishing the trend of the political discourse of the Ecuadorian presidential candidates using the data extracted from the tweets of the official accounts of the presidential candidates belonging to the 2020-2021 electoral process.

The structure of the chapters that make up this project is presented below.

In Chapter 1, the problem is formulated, its scope is delimited, and its justification is presented. It also details the general and specific objectives.

In Chapter 2, the concepts related to data visualization are studied, such as the stages that correspond to the development of visualizations, the definition of data mapping to visual objects, and the types of data used to make different visualizations, among others.

Chapter 3 details the development and implementation of the three visualizations, emphasizing the mapping carried out corresponding to each one of them.

Chapter 4 presents the functionality and user tests performed on the visualizations and the web interface in which they are hosted.

Chapter 5 presents the conclusions and recommendations that we reached once the project was completed

KEYWORDS: Visualizations, Politics, d3.js, Twitter.

1. INTRODUCCIÓN

Ecuador ha sido partícipe de diversos procesos electorales de gran trascendencia debido a los innumerables problemas por los cuales atraviesa el país. Una de las problemáticas que atraviesa es la cantidad de presidenciables que se presentan en los distintos procesos electorales en busca de un lugar en Carondelet, factor que genera conflicto en el ciudadano al no saber qué prioridades tiene cada candidato debido a la gran cantidad de información existente. Quizá por esto la población ecuatoriana prefiere tomar en cuenta otro tipo de factores como popularidad, partidos tradicionalistas, logros deportivos, logros culturales, entre otros, que muy poco tiene que ver con un análisis coherente del discurso político de cada uno de los candidatos para llegar a tener una idea clara y bien estructurada sobre quien merece nuestro voto.

En base a la encuesta "¿Qué sabemos sobre los candidatos presidenciales?" [1] realizada mediante la plataforma Google Forms se pudo evidenciar que el contexto político ecuatoriano se ha visto sumergido en otra gran problemática, el desconocimiento y/o la falta de información que tienen los electores respecto a las propuestas que los candidatos presentan en los distintos procesos electorales por los cuales ha pasado el país. Sumado a la gran cantidad de desinformación que existe tanto en los medios de comunicación como en redes sociales que derivan en una falta de seguridad de parte de la ciudadanía al momento de ejercer su derecho al voto que a posteriori conlleva en una mala elección de nuestros representantes.

Los medios usados por los candidatos para difundir sus propuestas y su discurso político han venido en constante cambio a lo largo de los años pasando por eventos públicos, periódicos, radio, televisión y llegando a tener un gran impacto en lo que respecta a redes sociales. Actualmente, las redes sociales son un espacio utilizado por líderes políticos para dar a conocer detalles de sus propuestas, siendo Twitter, Facebook e Instagram las aplicaciones que más relevancia toman en este aspecto. y que son ampliamente usados en distintos tópicos. Uno de los problemas que tiene el ciudadano es la dificultad de identificar las tendencias de los políticos ante la extensa cantidad de información que se puede encontrar en estas redes sociales.

El presente trabajo de titulación plantea la creación de tres visualizaciones a partir de la información obtenida del proceso electoral 2020-2021, alojadas en una aplicación web que permitirá al ciudadano conocer si los candidatos hablan acerca de distintas temáticas trascendentales en un país como son educación, salud, economía, ambiente, entre otras, y como las publicaciones que realizan en redes sociales se asocian a las temáticas

previamente mencionadas, contribuyendo a que el usuario tenga una idea clara de la tendencia que tienen en su discurso político los distintos candidatos.

1.1. OBJETIVOS

El objetivo general de este Proyecto Técnico es: desarrollar visualizaciones web interactivas para distinguir la tendencia del discurso político de los candidatos presidenciales ecuatorianos.

Los objetivos específicos del Proyecto Técnico son:

- Identificar los tipos de mapeo apropiados para las tres visualizaciones.
- Esquematizar las tres visualizaciones acordes al mapeo correspondiente.
- Implementar las tres visualizaciones interactivas que reflejen la tendencia del discurso político de los candidatos.
- Desarrollar una aplicación web responsiva e interactiva para el correcto despliegue de las visualizaciones.
- Validar el funcionamiento de la aplicación.

1.2. ALCANCE

Este trabajo explorará el diseño de tres visualizaciones dinámicas e interactivas, basadas en información obtenida del proceso electoral 2020-2021, donde se mostrará las tendencias del discurso político de los candidatos presidenciales hacia doce temáticas: ambiente, corrupción, derechos, economía, educación, energía, juventud, pobreza, salud, trabajo, violencia y vivienda. Además, tendrá la posibilidad de realizar comparaciones entre candidatos de su interés, o comparar cómo varía su discurso político de un segmento de tiempo a otro.

Este proyecto se centra específicamente en el diseño de un aplicativo web basado en JavaScript mediante D3.js, biblioteca enfocada a la realización de visualizaciones interactivas basadas en datos. Uno de los beneficios de esta biblioteca es que permite tener un control completo sobre el resultado visual final. El procedimiento por seguir se compone de cuatro pasos como se puede observar en la Figura 1.1.

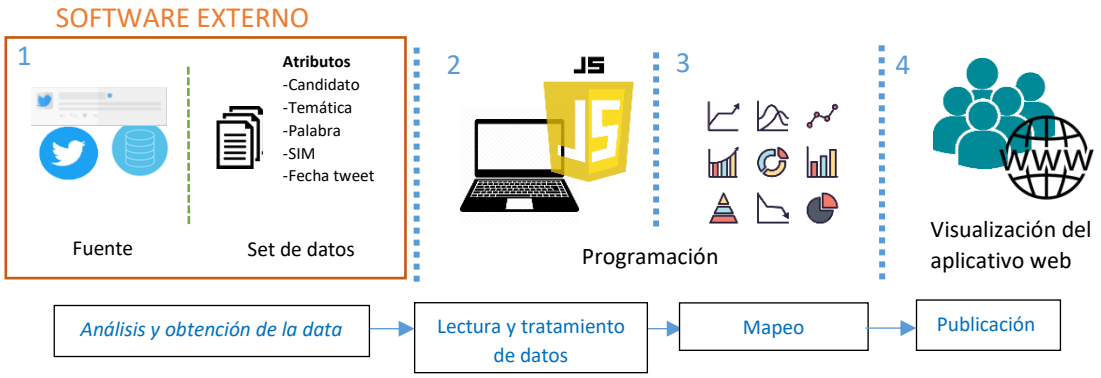

Figura 1.1. Pipeline visualizaciones

Los datos que se visualizarán se obtendrán de las cuentas oficiales de Twitter de 6 candidatos presidenciales de los 16 que actualmente se encuentran registrados en el CNE como contendientes para las elecciones de febrero de 2021.

El primer paso de este proyecto está basado en la obtención de un archivo en formato CVS, mismo que será proporcionado por el área de investigación de la Escuela Superior Politécnica del Litoral “Inferencia de conocimiento a partir de corpus de texto y meta data de publicaciones científicas y textos políticos en Latinoamérica usando técnicas de Procesamiento de Lenguaje Natural” mediante el uso de un software propiedad de este. Dicho archivo se compone de cinco atributos: candidato, temática, palabra, SIM (Valor de la similitud semántica calculada en base a la similitud de coseno) y la fecha del tweet. El SIM o índice de pertenencia representa la medida en que se relacionan dos palabras en específico, para nuestro caso de aplicación, una palabra de un tweet con una temática.

Como segundo paso, se realizará la lectura y el tratamiento de datos, filtrándolos de manera que se obtenga únicamente los registros necesarios para las visualizaciones que se realizarán, posteriormente se realizará el mapeo de los datos a objetos visuales acorde a las necesidades de cada visualización utilizando la biblioteca D3.js. Las visualizaciones permitirán interactividad con el usuario, siendo los candidatos y las temáticas las variables que el usuario podrá usar para filtrar y seleccionar datos a visualizar.

Mapear datos a objetos visuales consiste en la selección de la representación simbólica idónea para un conjunto de datos para la implementación de una visualización que busca un objetivo en concreto, ya sea presentar tendencias, proporciones, semejanzas, entre otros. El éxito de una visualización depende del mapeo que se realice de la misma.

Como tercer paso, una vez estén las visualizaciones correctamente programadas se someterán a pruebas de funcionamiento. Por último, al disponer ya de las visualizaciones

se procede a publicarlas en el aplicativo web y realizar pruebas de validación con usuarios potenciales.

El proyecto se basará en tres visualizaciones interactivas.

La primera visualización denominada “Tabla de tendencia del discurso político”, se basa en una tabla dinámica que muestra a qué temáticas se asocian las distintas palabras obtenidas de los tweets en base al índice de pertenencia de cada una. Esta visualización permitirá observar la relación que existe entre las palabras obtenidas de los tweets y las temáticas, lo que marcará la tendencia que tienen uno u otro candidato en su discurso político.

La segunda visualización denominada “Red de tendencia del discurso político”, se basa en una red en donde los nodos principales son las categorías y los nodos secundarios son las palabras obtenidas de los tweets demostrando mediante enlaces la relación existente entre los nodos. Al igual que la primera visualización ayudará a diferenciar las tendencias que tienen los candidatos además de permitir observar la existencia de distintas palabras que se relacionan a más de una temática.

La tercera visualización denominada “Comparativa de tendencia del discurso político” ayudará a comparar dos espacios de tiempo y cómo ha ido cambiando la tendencia de los distintos candidatos respecto a los dos tiempos planteados. La visualización se compondrá de 3 secciones ordenadas de izquierda a derecha, las secciones de borde representarán cada una a un segmento de tiempo en específico y mostrarán las palabras clave representadas por un nodo y la sección central mostrará las categorías representadas por un nodo y las palabras clave se asociarán a la categoría correspondiente mediante un enlace.

1.3. JUSTIFICACIÓN

Actualmente, ante la cantidad extensa de data de diversos tópicos que se tiene, en especial en campañas políticas se necesita de un mecanismo que permita centralizar, resumir los datos y brindar una idea concisa y clara sobre los discursos políticos existentes.

Las visualizaciones son el medio ideal para este propósito. Sus propiedades permiten procesar los datos obtenidos y representarlos de manera gráfica, comprimiendo la información y, dependiendo del tipo de mapeo realizado, brindar resultados enfocados a un tópico específico como es la tendencia de un discurso político.

El uso de la red social Twitter mediante la obtención de distintas palabras de los tweets de las cuentas oficiales de los candidatos presidenciales relacionándolas con una serie de temáticas de interés público, será el método y los elementos que se utilizarán en este proyecto para poder procesarlos, de manera que se pueda mapear a objetos visuales que brinden una idea clara sobre las tendencias de los discursos políticos de los candidatos.

Sitios web tales cómo “ECUADOR DECIDE” permite comparar propuestas de máximo tres candidatos presidenciales, apoyándose en la comprensión lectora de los usuarios. La información presentada por esta plataforma se basa específicamente en el plan de trabajo de cada uno de los candidatos. La fuente de datos del proyecto se basa en el discurso político que los candidatos presentan a través de la red social Twitter. Esta información se traduce a tres gráficas en las cuales se presenta las tendencias basadas en términos expresados por cada uno de los candidatos facilitando de esta manera realizar distintas comparaciones entre ellos.

La información obtenida de Twitter acoplándola a una filtración y mapeo a objetos visuales correcto es una vía válida para poder proporcionar información concisa y precisa sobre un tópico en específico. Es más sencillo para el ciudadano en general procesar una visualización informativa y sacar una conclusión mejor estructurada, a procesar una gran cantidad de información de la cual es fácil perder la secuencia y el objetivo de la misma.

1.4. MARCO TEÓRICO

1.4.1. INTRODUCCIÓN A LA VISUALIZACIÓN DE DATOS

Para entender de una mejor manera la definición de visualización de datos, debemos comprender la diferencia entre dato, información y conocimiento.

Como menciona Russell Ackoff’s [2], un dato es un símbolo que representa propiedades de un objeto o evento y el ambiente en el cual se desenvuelve. Información son los datos procesados para ser usados y que son capaces de dar respuestas a preguntas básicas: “Quién”, “Qué”, “Dónde” y “Cuándo” mientras el conocimiento es la aplicación de los datos e información dando respuestas a preguntas “Cómo”.

Llevando estos mismos tópicos y ambientándolos a la informática, Min Chen plantea las siguientes definiciones. Dato es la representación computarizada de modelos y atributos de entidades reales o simuladas. Información son el conjunto de datos resultado de un proceso computacional que le brinda un significado y que puede ser procesado por el ser humano. Mientras el conocimiento son los datos que representan los resultados de un proceso cognitivo simulado por computadora, como la percepción, el aprendizaje,

asociación y razonamiento, o las transcripciones de algunos conocimientos adquirido por los seres humanos.

En base a las definiciones previamente mencionadas, una visualización de datos consiste en la transmisión de la información obtenida de un conjunto de datos, mediante imágenes, símbolos y elementos interactivos que permiten al usuario final generar conocimiento basado en el análisis de las visualizaciones

La visualización de datos es una fuente de conocimiento que evoluciona conforme las fuentes de extracción de datos aumenta y por ende da apertura a nuevas formas de visualización de los mismos. La tecnología, desde los dispositivos móviles pasando por el boom de las redes sociales hasta la digitalización de distintos servicios de consumo social permite disponer de datos sobre distintas actividades humanas. Desde la aparición del internet la capacidad de generar, procesar y compartir datos ha crecido exponencialmente, lo que a la postre da como resultado cantidades exorbitantes de datos.

1.4.2. ETAPAS DE LAS VISUALIZACIONES

Existen diferentes autores que definen distintas etapas a las que se somete una visualización antes de llegar hasta el usuario final. En esta ocasión tomaremos las etapas planteadas por Colin Ware que basa su análisis en el aspecto sensorial y/o emocional que produce una visualización en una persona y que tienen una serie de circuitos de realimentación.

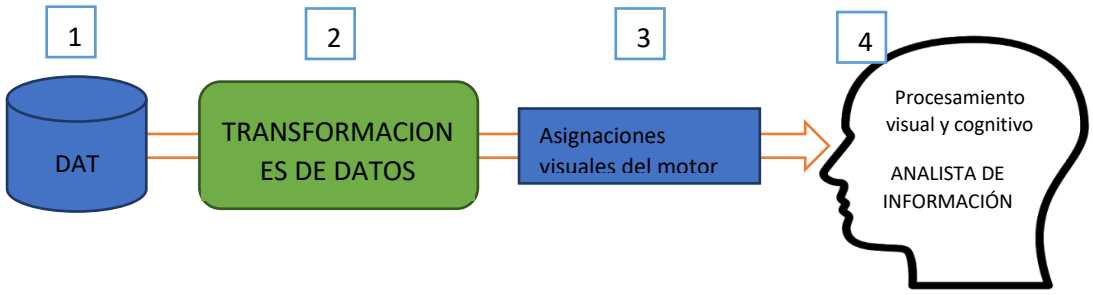

Figura 1.2. Etapas de las visualizaciones según Collin Ware

A continuación, se detallan las 4 etapas de visualización según Ware, y que se reflejan en la Figura 1.2 [3]:

1. Recopilación y almacenamiento de datos
2. Una etapa de preprocesamiento diseñada para transformar los datos en información que sea más fácil de manipular. Por lo general, hay alguna forma de

reducción de datos para revelar aspectos seleccionados. La exploración de datos es el proceso de cambiar el subconjunto que sé que se está viendo actualmente.

3. Mapeo de los datos seleccionados a una representación visual, que se logra a través de algoritmos informáticos que producen una imagen en la pantalla.
4. El sistema cognitivo y perceptivo humano.

1.4.3. TIPOS DE DATOS Y OPERACIONES

Aunque relativamente antiguo (data de 1996), en [4] se introduce un famoso mantra sobre cómo debe plantearse la búsqueda visual de información: *"Overview first, zoom and filter, then details-on-demand"*, es decir, primero una visión general, después seleccionar y ampliar la zona de interés, y finalmente añadir el detalle necesario según las necesidades del usuario. A partir de esta idea, Shneiderman define siete operaciones que pueden ser aplicadas a conjuntos de datos, que también son (coincidentemente) de siete tipos diferentes:

- *Datos 1-dimensionales:* datos organizados secuencialmente, que son consumidos según el orden como que se han generado. Por ejemplo, documentos de texto, listas de elementos, etc.
- *Datos 2-dimensionales:* datos que representan una cierta estructura espacial, por lo tanto, incluyendo el concepto de posición. Un ejemplo obvio son los mapas o diagramas.
- *Datos 3-dimensionales:* en algunos casos, se desea representar la posición de un conjunto de elementos en el mundo real, y en este caso es necesario utilizar tres dimensiones. Muchos de los primeros esfuerzos en el ámbito de la visualización de datos pertenecían a esta categoría, como por ejemplo los sistemas de información geográfica, los entornos de diseño asistidos por ordenador (CAD) para el modelado de piezas o en la arquitectura, o las imágenes resultantes de procesos de barrido del cuerpo humano.
- *Datos temporales:* son aquellos relacionados con eventos que tienen un inicio y un fin, y que pueden solaparse. Es importante no confundirlos con datos 1-dimensionales que son generados por un proceso con un cierto ritmo, como por ejemplo un conjunto de mediciones de la temperatura diaria en un lugar.

- *Datos multidimensionales o n-dimensionales:* en la mayoría de los casos, los ítems almacenados en una base de datos se describen mediante n atributos, lo cual dificulta su visualización dadas las limitaciones de nuestro mundo tridimensional.
- *Datos jerárquicos (Árboles):* en este caso se trata de datos que tienen una relación entre ellos, en la que cada elemento o nodo tiene un enlace a su (único) ancestro, excepto en el caso del nodo raíz.
- *Redes:* cuando las relaciones entre nodos son más generales y no existen restricciones, los árboles generalizan en grafos, los cuales pueden ser de muchos tipos diferentes (dirigido, bipartido, acíclico, etc.). Actualmente el análisis y visualización de datos de redes es una de las áreas con mayor interés (por ejemplo, el análisis del flujo de contenido en Twitter).

Sobre estos tipos de datos, Shneiderman define siete operaciones básicas, según la idea repetida en el mantra *("Overview first, zoom and filter, then details-on-demand")* [4]. Obviamente para cada tipo de datos, el significado de la operación puede ser ligeramente diferente:

- *Overview:* se trata de crear una vista general de todos los datos disponibles, como punto de entrada a la exploración de estos. La vista general puede no incluir todos los datos al mismo tiempo, pero entonces debe proporcionar un mecanismo para que el usuario pueda seleccionar otros datos al mismo nivel de detalle.
- *Zoom:* se trata de un ajuste de la vista anterior, seleccionando un subconjunto de datos, pero manteniendo la sensación de posición y el contexto, acercándose o alejándose de los datos. El zoom complementa la operación anterior para poder hacerse una idea de la estructura subyacente en los datos.
- *Filter:* el usuario puede seleccionar un subconjunto de los datos, de forma que solamente visualice aquellos que cumplan unos ciertos criterios, eliminando el resto. Los criterios pueden especificarse de muchas maneras, dependiendo del tipo de datos: en el caso de una variable numérica 1-dimensional, se puede especificar un rango, mientras que en una 2-dimensional se utiliza el concepto de bounding box, o caja que encierra un conjunto de datos.
- *Details-on-demand:* una vez, mediante las operaciones anteriores, ya se ha seleccionado un subconjunto reducido de datos, el usuario puede solicitar información de cada uno de ellos, normalmente mediante el uso del ratón, pasando

por encima o haciendo clic en los datos, de forma que aparezca la información adicional contenida en cada uno de ellos.

- *Relate:* en el caso de datos n-dimensionales, es posible establecer criterios de distancia entre los elementos del conjunto, según su tipología y el uso que se le quiera dar a la visualización. Mediante esta distancia es posible.
- *History:* se trata de mantener una lista de las operaciones realizadas, de forma que sea posible deshacer alguno de los pasos realizados durante el proceso de visualización de los datos.
- *Extract:* finalmente, una vez se han seleccionado los datos y la información adicional requerida, se trata de poder volcarlos para poder reutilizarlos en otro contexto, almacenándolos como un nuevo conjunto de datos.

1.4.4. MAPEO DE DATOS

El mapeo de datos a objetos visuales consiste en seleccionar elementos gráficos idóneos para el tipo de dato que se desea visualizar. Una visualización correcta, completa, concisa, precisa y que genera rápidas conclusiones acertadas en quién la usa refleja un correcto mapeo de datos. Se podrían definir 3 pilares fundamentales para un correcto mapeo de los datos [5].

El primer pilar consiste en analizar qué tipo de dato estamos usando, sean estos unidimensionales o multidimensionales, cuantitativos, cualitativos, entre otros. Cada tipo de datos tiene una manera concreta de ser representados como, por ejemplo, mapear los datos de cantidad de celulares fabricados, teniendo los registros de la cantidad de celulares fabricados año a año, tiene sentido hacerlo siguiendo el orden de los años y no por el valor más alto de unidades fabricadas. Es por esto que saber qué tipo de datos tengo es el primer paso para poder realizar una visualización correcta.

El segundo pilar consiste en definir la codificación visual, es decir, encontrar la mejor manera de mostrar visualmente los datos usando colores, tamaños, formas, posición. Según W. Cleveland y R. McGill [6], todos tienen distintas percepciones de las visualizaciones, pero existen técnicas generales que se pueden seguir para encontrar una estructura correcta para la visualización.

Y como tercer pilar está el uso, que básicamente responde a la pregunta ¿Qué queremos mostrar con los datos? Dependiente de los dos pilares previos, la difusión del gráfico y el correcto discernimiento da la información mostrada confirma el correcto funcionamiento del

mismo. Puede que un gráfico sea visualmente atractivo para el usuario, pero si no logra comunicar información que ayude al usuario a razonar o analizar datos pierde su valor.

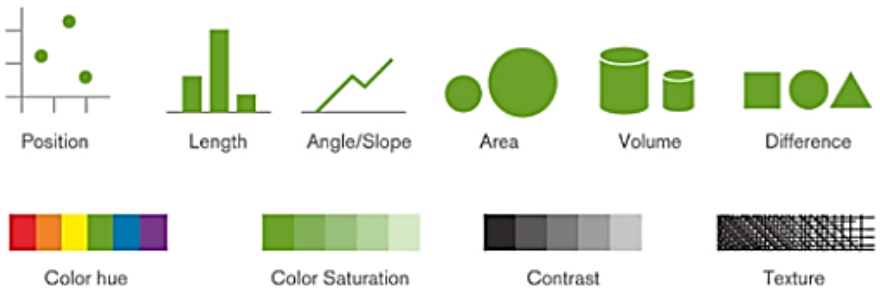

Figura 1.3. Opciones para codificación visual

1.4.5. TIPOS DE VISUALIZACIONES

Existen distintos tipos de visualizaciones, las cuales encontrarán su mejor presentación en base a los datos que se va a representar, sus relaciones y el objetivo de los mismos. A continuación, se muestran distintos ejemplos de visualizaciones [7].

1.4.5.1. DATOS DE SERIES DE TIEMPO

Los gráficos de series de tiempo es una de las formas más comunes de visualización de datos. Como su nombre indica, proporciona una forma idónea de visualizar datos que varían en el tiempo. Dichos datos son fundamentales para diversos dominios como son las finanzas, ciencia, políticas públicas, entre otros.

1.4.5.1.1. Gráficos de índice

Un gráfico de índice es un tipo de visualización de líneas interactivo que muestra cambios porcentuales para una colección de datos de series de tiempo basados en un punto de índice seleccionado. Por ejemplo, en la Figura 1.4 se muestra el porcentaje cambio de los precios de las acciones seleccionadas si se compran en enero de 2005.

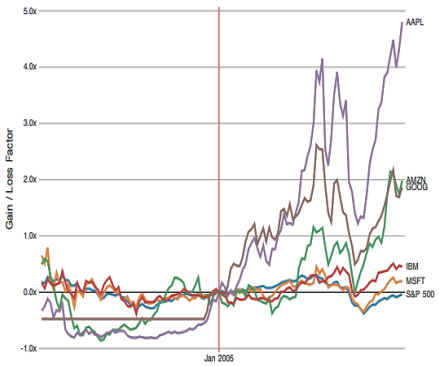

Figura 1.4. Visualización tipo índice

1.4.5.1.2. Gráficos apilados

Otra forma de representar datos de series de tiempo son los gráficos apilados que consiste en apilar gráficos de área uno encima de otro resultando una suma visual de valores de series de. Este tipo de gráfico también llamado gráfico de flujo representa patrones agregados y, a menudo, admite el desglose en un subconjunto de series individuales.

En la Figura 1.5 se muestra el número de trabajadores desempleados en los Estados Unidos durante la última década, subdividido por industria. Si bien estos gráficos han demostrado ser populares en los últimos años, tienen algunas limitaciones notables. Un gráfico apilado no admite valores negativos y no tiene sentido para datos que no se deben sumar (temperaturas, por ejemplo).

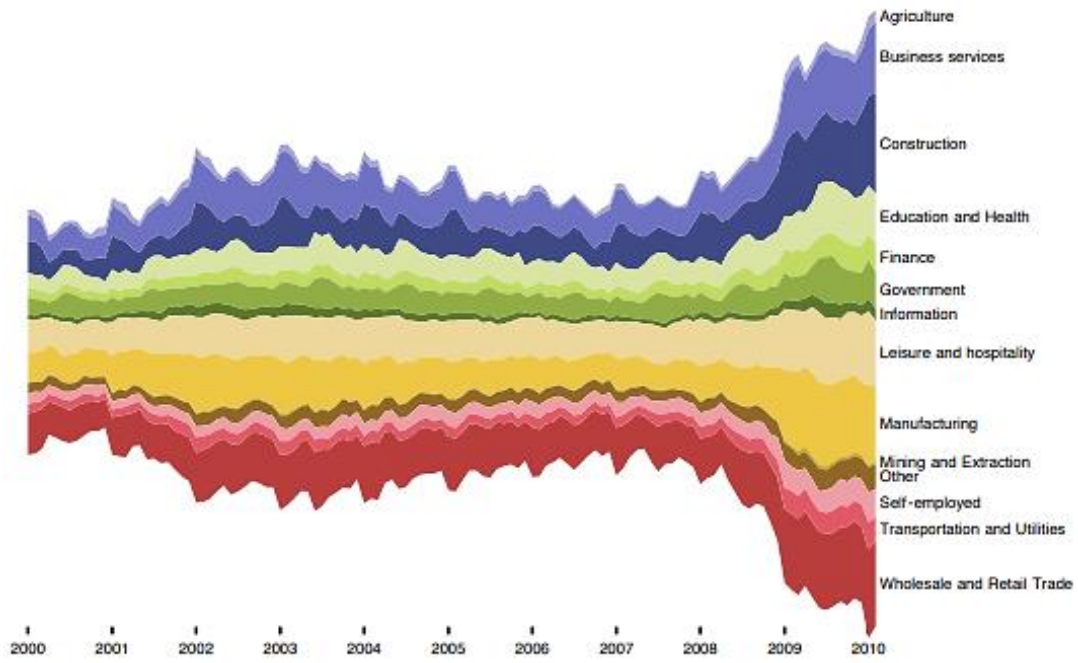

Figura 1.5. Visualización por apilamiento

1.4.5.1.3. Pequeños múltiplos

Usado como alternativa del gráfico apilado, consiste en trazar múltiples series de tiempo dentro de los mismos ejes cómo el gráfico de índice. Colocar varias series en un mismo espacio puede producir superposición de las mismas que reducen la legibilidad de la data. Sin embargo, como alternativa se puede hacer múltiplos pequeños: mostrar cada serie en su propio gráfico.

En la Figura 1.6 se puede ver el número de trabajadores desempleados pero normalizado dentro de cada industria en gráficos independientes logrando deducir tanto tendencias generales como patrones independientes para cada industria.

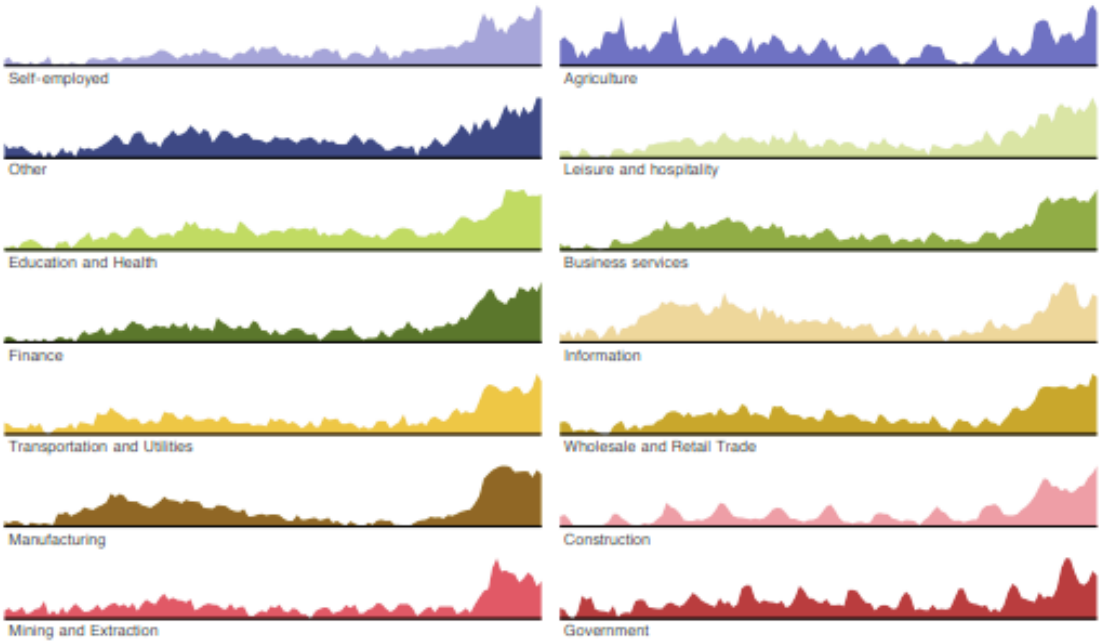

Figura 1.6. Visualización por pequeños múltiplos

1.4.5.2. DISTRIBUCIÓN ESTÁTICA

Se han diseñado otras visualizaciones para revelar cómo se distribuye distribuye un conjunto de números y así ayudar al analista a comprender mejor las propiedades estadísticas de los datos.

Los analistas a menudo quieren ajustar sus datos a modelos estadísticos, ya sea para probar hipótesis o predecir valores futuros, pero una elección incorrecta del modelo puede dar lugar a predicciones erróneas. Por lo tanto, un uso importante de las visualizaciones es el análisis de datos exploratorio: obtener información sobre cómo se distribuyen los datos para informar la transformación de los mismos y las decisiones de modelado.

Las técnicas comunes incluyen el histograma, que muestra la prevalencia de valores agrupados en contenedores, y el diagrama de caja y bigotes, que puede transmitir características estadísticas como la media, mediana, límites de cuartiles o valores atípicos extremos. Además, existen otras técnicas para evaluar una distribución y examinar las interacciones entre múltiples dimensiones.

1.4.5.2.1. Diagrama de tallo y hoja

Para evaluar una colección de números, una alternativa al histograma es el diagrama de tallo y hojas. En general, agrupa los números en relación al primer dígito significativo y luego agrupa los valores dentro de cada ubicación por el segundo dígito significativo.

Esta representación utiliza los datos en si para mostrar una distribución de frecuencia, reemplazando las barras de información vacía de un gráfico de barras de histograma tradicional permitiendo evaluar tanto la distribución general como el contenido de cada contenedor.

En la Figura 1.7, el diagrama de tallo y hoja nuestra la distribución de las tasas de finalización de los trabajadores que completan tareas colectivas en Mechanical Turk de Amazon.

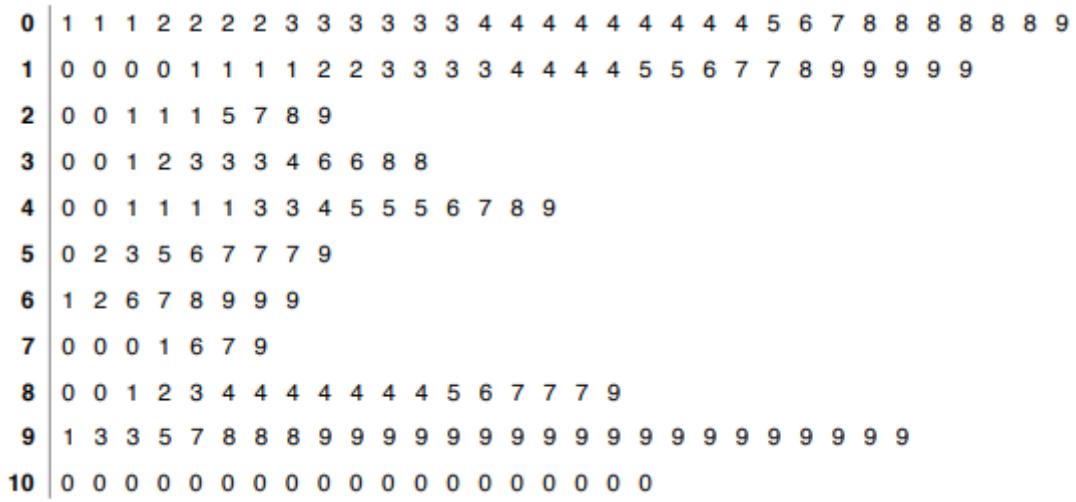

Figura 1.7. Visualización por diagrama de tallo y hoja

1.4.5.2.2. Diagrama Q-Q (Cuantil – Cuantil)

La gráfica Q-Q compara dos distribuciones de probabilidad al graficar sus cuantiles entre sí. Si los dos son similares, los valores trazados se ubicarán aproximadamente a lo largo de la diagonal central. Si los dos están relacionados linealmente, los valores volverán a estar a lo largo de una línea, aunque con pendiente e intersecciones variables.

La **Figura 1.8** muestra los mismos datos de participación de Mechanical Turk en comparación con tres distribuciones estadísticas. Los datos forman tres componentes distintos cuando se comparan con distribuciones uniformes y normales (gaussianas): esto sugiere que un modelo estadístico con tres componentes podría ser más apropiado y, de hecho, vemos en el gráfico final que una mezcla ajustada de tres distribuciones normales proporciona un mejor ajuste.

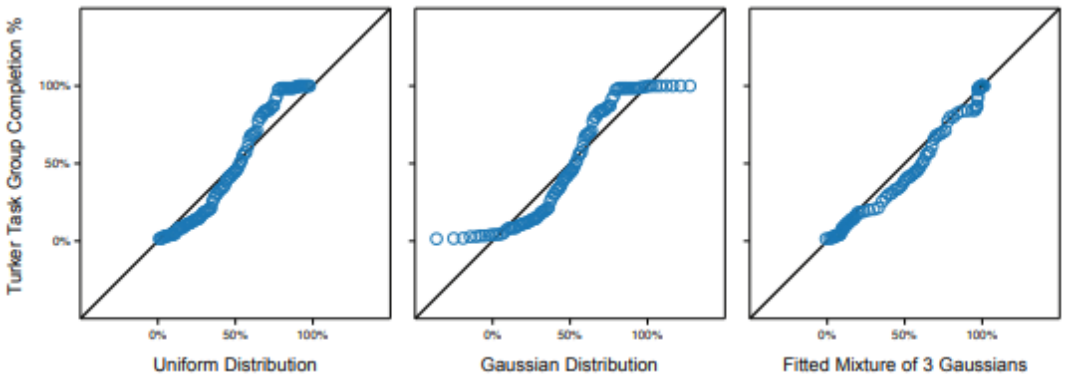

Figura 1.8. Visualización por diagrama Q-Q

1.4.5.2.3. SPLOM (Matriz de distribución)

Otras técnicas de visualización intentar representar las relaciones entre múltiples variables. Los datos multivariados ocurren con frecuencia y son muy difíciles de representar, en parte debido a la dificultad de representar mentalmente los datos en más de tres dimensiones. Una técnica para superar este problema es utilizar pequeños múltiplos de diagramas de dispersión que muestren un conjunto de relaciones por pares entre variables, creando así el SPLOM (Matriz de diagramas de dispersión). Un SPLOM permite la inspección visual de las correlaciones entre cualquier par de variables.

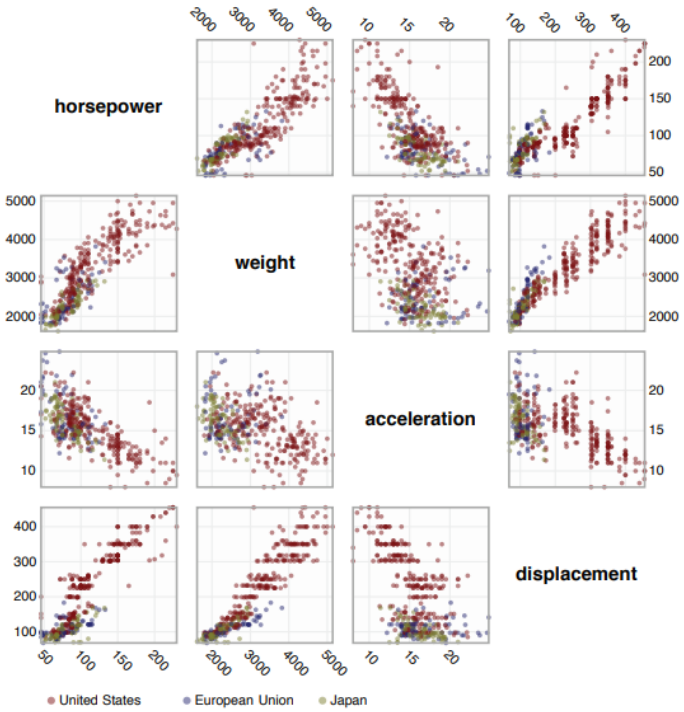

Figura 1.9. Visualización por SPLOM

En la **Figura 1.9** se usa una matriz de diagrama de dispersión para visualizar los atributos de una base de datos de automóviles, mostrando las relaciones entre caballos de fuerza, peso, aceleración y desplazamiento.

1.4.5.3. MAPAS

Pese a que un mapa puede ser la forma base de representar datos geográficos, tiene un trasfondo rico en historia de diseño. Muchos mapas se basan en una proyección cartográfica: una función matemática que asigna la geometría tridimensional de la Tierra a una imagen bidimensional. Otros mapas distorsionan o abstraen a sabiendas las características geográficas para contar una historia más rica o resaltar datos específicos.

1.4.5.3.1. Mapas de flujo

Al colocar líneas de trazos en la parte superior de un mapa geográfico, un mapa de flujo puede representar el movimiento de una cantidad en el espacio y (implícitamente) en el tiempo. Las líneas de flujo suelen codificar una gran cantidad de información multivariante: los puntos de ruta, la dirección, el grosor de la línea y el color se pueden utilizar para presentar las dimensiones de la información al espectador.

La Figura 1.10 es una interpretación moderna de la descripción de Charles Minard de la marcha de Napoleón sobre Moscú.

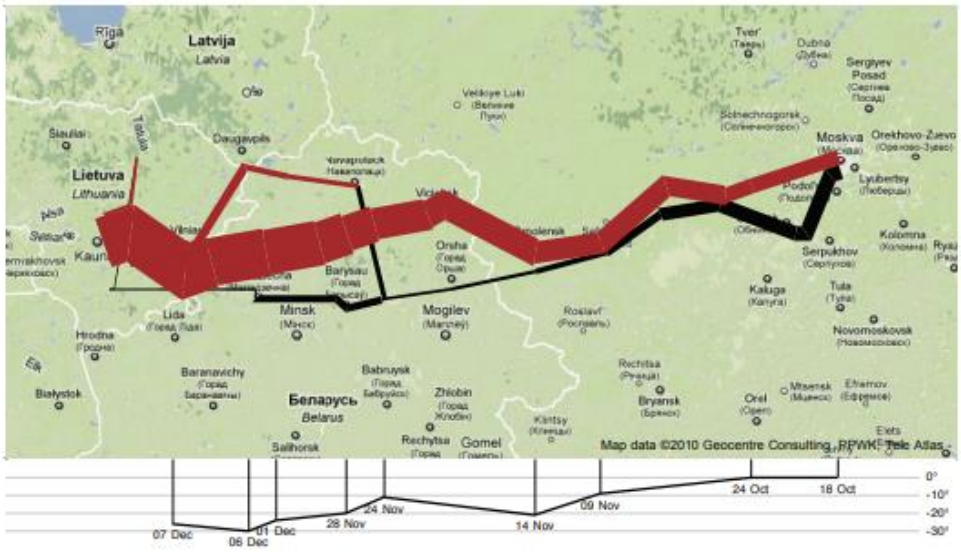

Figura 1.10. Visualización por mapa de flujo

1.4.5.3.2. Mapas de copletas

Los datos comúnmente se recopilan y se asignan por áreas geográficas como ciudades provincias y/o estados. Un enfoque estándar para representar estos datos es la utilización

del color como codificación del área geográfica, que da como resultado un mapa diferenciado por colores mejor conocido como coropletas. La **Figura 1.11** usa una codificación de colores para comunicar la prevalencia de la obesidad en cada estado de los EE. UU.

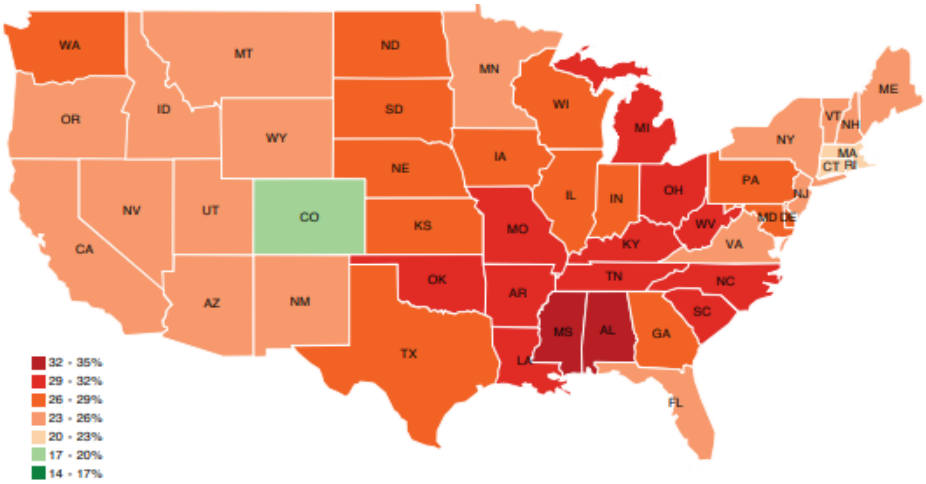

Figura 1.11. Visualización por mapa de coropletas

1.4.5.3.3. Cartograma

Los cartogramas distorsionan la forma de las regiones geográficas para que el área codifique directamente una variable de datos.

En la Figura 1.12, el área circular codifica el número total de personas obesas por estado y el color indica el porcentaje de la población total que es obesa.

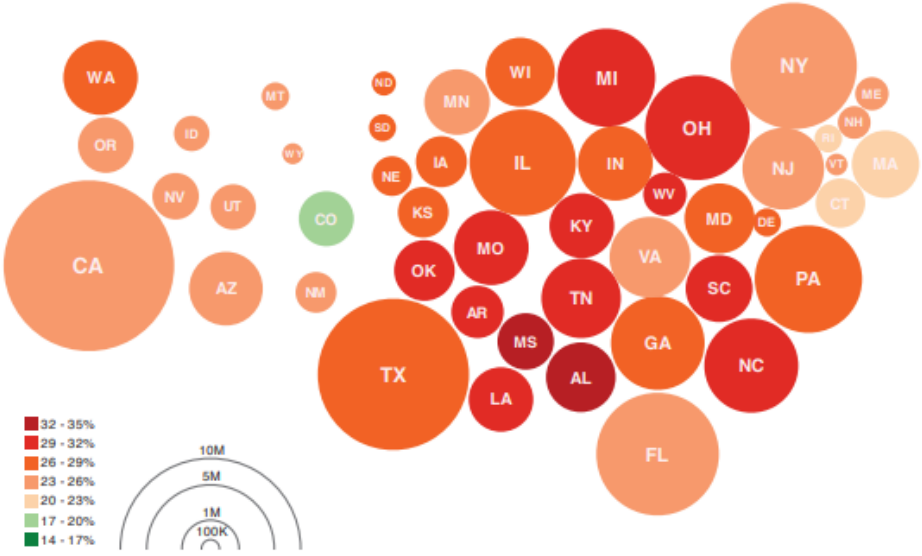

Figura 1.12. Visualización por mapa de cartograma

1.4.5.4. JERARQUÍAS

Si bien la mayor parte de datos son una colección plana de números, la gran mayoría de estos se pueden organizar mediante jerarquías. Existen varias técnicas de visualización especiales que aprovechan la estructura jerárquica permitiendo una rápida inferencia multiescalar: micro observaciones de elementos individuales y macro observaciones de grandes grupos.

1.4.5.4.1. Diagramas de nodo-enlace

La palabra árbol se usa indistintamente con jerarquía, ya que las ramas fractales de un roble pueden reflejar el anidamiento de datos. Si tomamos un plano bidimensional de un árbol, tenemos una opción popular para visualizar jerarquías: un diagrama de nodo-enlace. Se han diseñado muchos algoritmos de distribución de árboles diferentes; el algoritmo de Reingold-Tilford, utilizado en la Figura 1.13 en una jerarquía de paquetes de clases de software, produce un resultado ordenado con un mínimo de espacio desperdiciado.

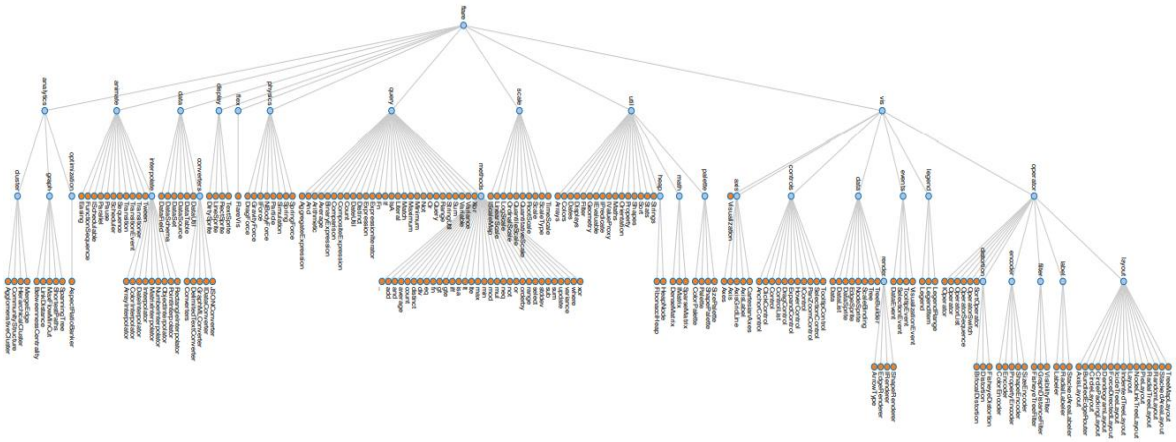

Figura 1.13. Visualización por diagrama de nodo-enlace

Un esquema de visualización alternativo es el algoritmo dendrograma (o clúster), que coloca los nodos de las hojas del árbol al mismo nivel. Así, en el diagrama de la Figura 1.14, las clases (nodos de color naranja) están en el diámetro del círculo, con los paquetes (nodos internos azules) adentro.

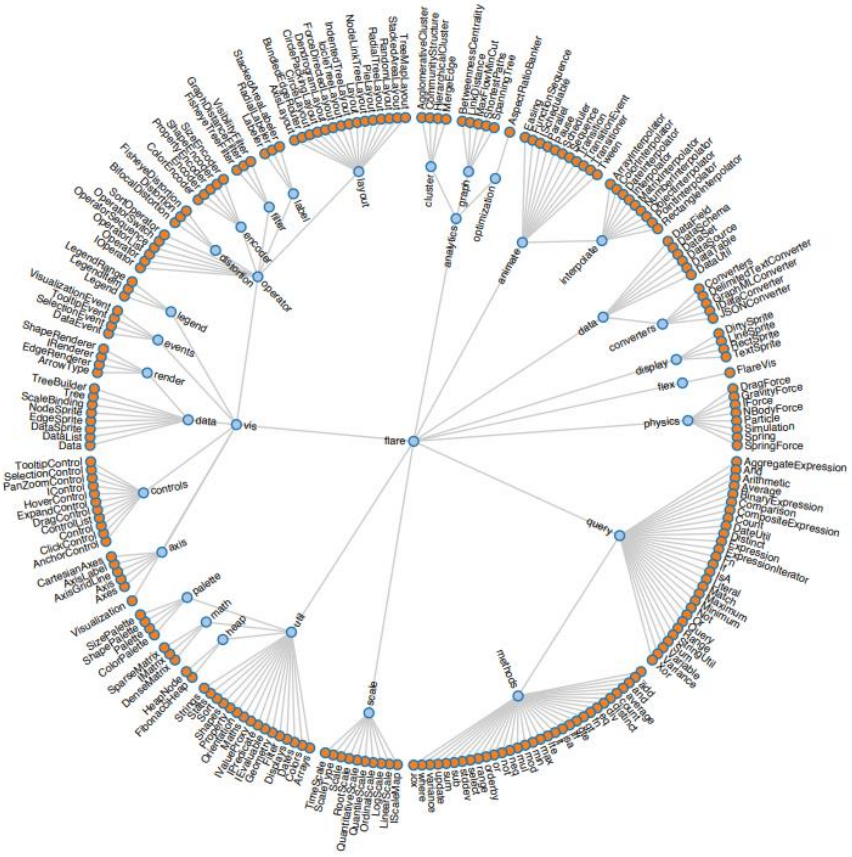

Figura 1.14. Visualización por diagrama de nodo-enlace 2

1.4.5.4.2. Diagramas de adyacencia

El diagrama de adyacencia es una variante del diagrama de nodo-enlace la cual en vez de ocupar un enlace como tal los nodos ocupan un espacio dentro del gráfico como áreas sólidas (arcos o barras) y su posición relativa a los nodos adyacentes (área sólida) indica su posición en la jerarquía.

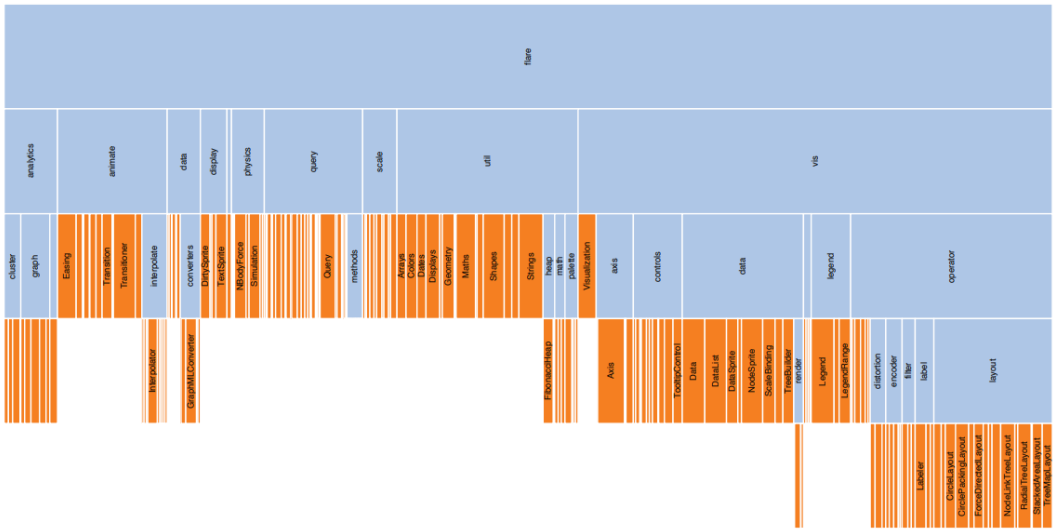

Figura 1.15. Visualización por diagrama de adyacencia en barras

En la Figura 1.15 se puede observar cómo el nodo raíz aparece en la parte superior y los nodos secundarios van ordenándose para abajo, de igual manera en la Figura 1.16 el nodo raíz viene a estar en el centro y los nodos secundarios se expanden hacia afuera, pero en forma de arco.

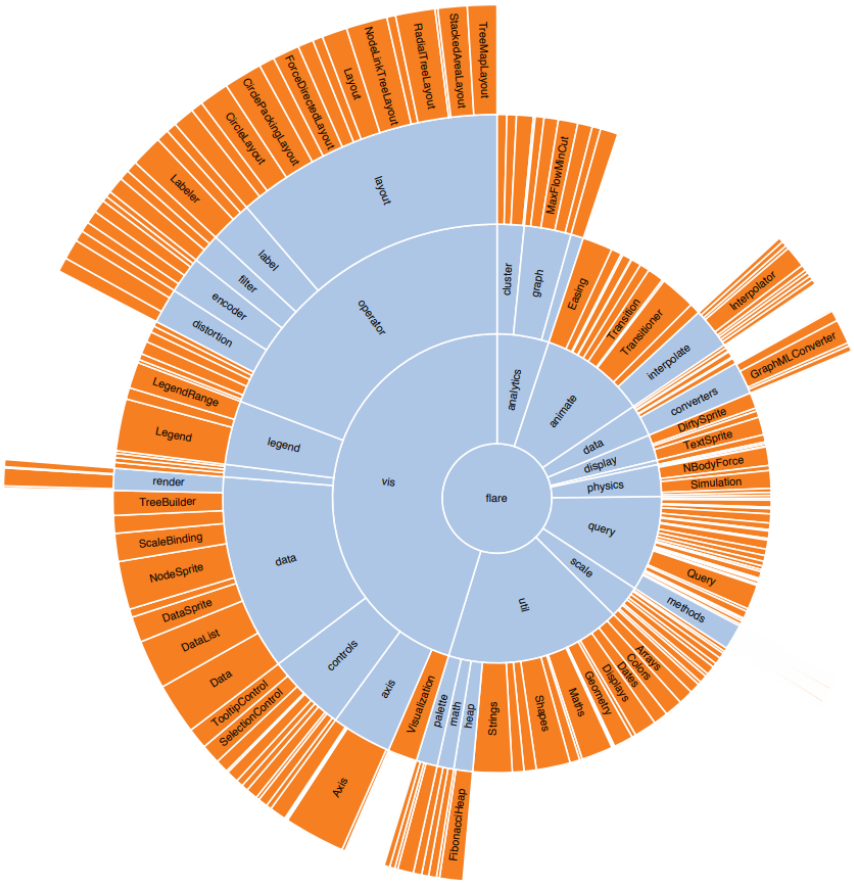

Figura 1.16. Visualización por diagrama de adyacencia en arcos

1.4.5.5. REDES

Mientras en los diagramas de jerarquía explorábamos la organización de los datos, al hablar de redes insertamos dentro de las características que nos brindan los gráficos la relación que existe entre nodos. Dicho esto, una jerarquía es una forma especializada de red: cada nodo tiene exactamente un enlace a su padre mientras el nodo raíz no tiene enlaces superiores.

Por lo tanto, los diagramas de enlace-nodo también se utilizan para visualizar redes perdiendo la característica de jerarquía e insertando como prioridad la visualización de las relaciones existentes entre nodos lo que conlleva al uso de algoritmos diferentes para el posicionamiento de los nodos.

1.4.5.5.1. Diseños dirigidos por la fuerza

Los nodos son partículas cargadas que se repelen entre sí, y los enlaces son resortes amortiguados que unen los nodos relacionados. Luego, una simulación física de estas fuerzas determina las posiciones de los nodos; Las técnicas de aproximación que evitan calcular todas las fuerzas por pares permiten el diseño de un gran número de nodos. Además, la interactividad permite al usuario dirigir el diseño y mover los nodos para eliminar la ambigüedad de los enlaces.

En la Figura 1.17 se usa un diseño dirigido por la fuerza para ver la red de coocurrencia de personajes en los capítulos de la novela clásica de Víctor Hugo, Los Miserables. Los colores de los nodos representan la pertenencia a un clúster calculada por un algoritmo de detección de la comunidad.

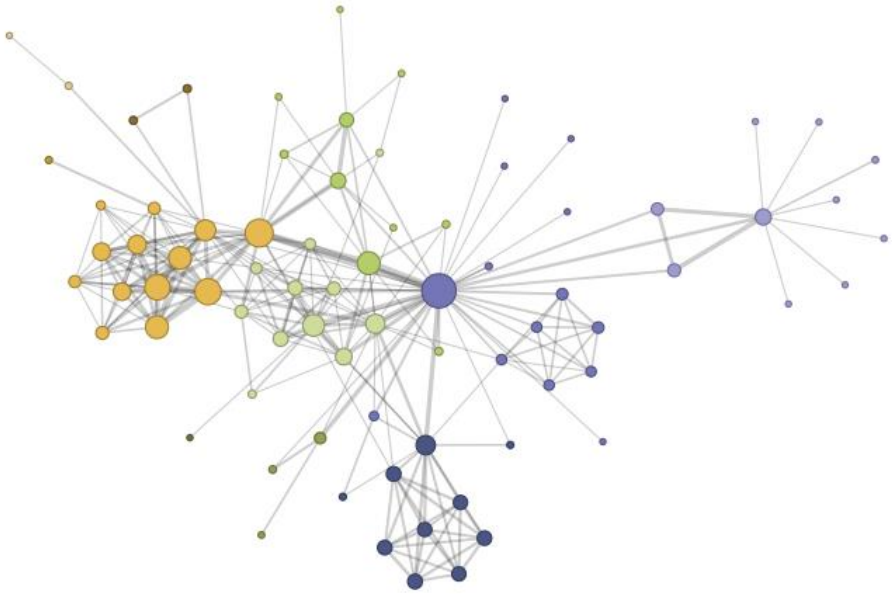

Figura 1.17. Visualización por diagrama de red dirigido por la fuerza

1.4.5.5.2. Diagramas de arco

Un diagrama de arco, como se muestra en la Figura 1.18, usa un diseño unidimensional de nodos, con arcos para representar los enlaces. Aunque este tipo de gráfico no muestra una estructura general del gráfico como lo hace el diagrama de fuerza, con una buena distribución de ellos nodos se facilita la visualización de las relaciones existentes entre los mismos.

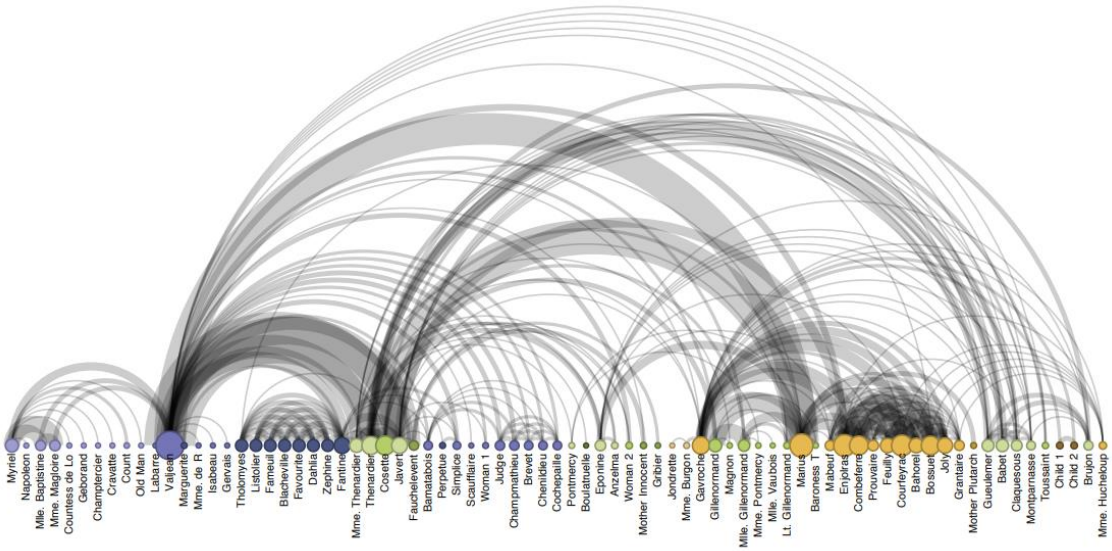

Figura 1.18. Visualización por diagrama de arco

1.4.6. BENEFICIOS DE LAS VISUALIZACIONES

El objetivo final del diseño de visualización interactiva es optimizar las aplicaciones para que nos ayuden a realizar el trabajo cognitivo de manera más eficiente. Optimizar un sistema requiere que tengamos al menos alguna concepción del valor. Usamos visualizaciones porque nos ayudan a resolver problemas más rápido o mejor, o nos permiten aprender algo nuevo. Entre los infinitos beneficios destacan los siguientes:

- Clave para desbloquear gran cantidad de datos.
- Permite una absorción casi instantánea de una gran cantidad de datos.
- Acelera el proceso de toma de decisiones.
- Revela patrones o tendencias.
- Muestra rápidamente la relación entre operaciones y resultados.
- Es interactiva
- Mejora la comunicación.

1.4.7. VISUALIZACIONES EN EL CONTEXTO POLÍTICO

En el contexto político y todo lo que este tópico abarca es una fuente sumamente grande de información, ya sea enfocado a los datos que arrojan los comicios o la campaña electoral previo a estos.

Varios han sido los trabajos desarrollados en este ámbito, localmente existe un gran exponente de este tipo de visualizaciones como lo es Roberto Camana Fiallos [8]. El objetivo de esta investigación fue utilizar herramientas para la visualización dinámica de los datos. El conjunto de datos se basó en las elecciones presidenciales de Ecuador en primera vuelta de los años 2006, 2009 y 2013. Como resultado se obtuvo tres visualizaciones.

En primer lugar, la votación histórica por partidos políticos (2006–2013), utilizando gráficos de burbujas; este análisis tenía la finalidad de conocer la evolución de la votación alcanzada por partidos políticos recientemente creados, frente al oficialismo, como se muestra en la siguiente figura.

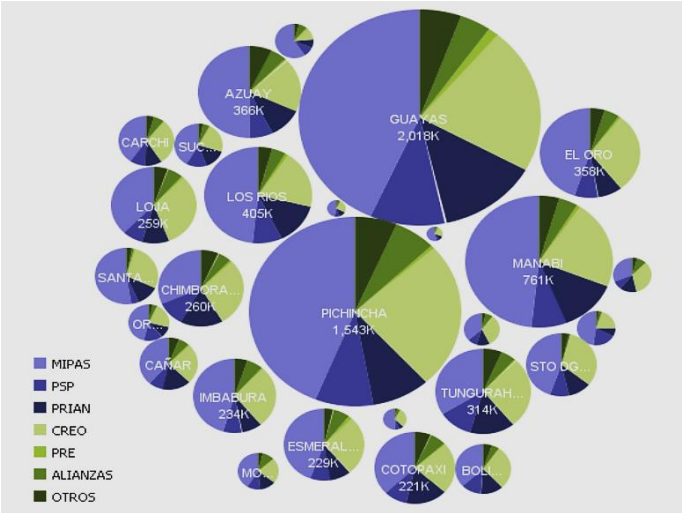

Figura 1.19. Votación histórica por partidos políticos

La segunda, resultados por cantones, divididos por género (hombre y mujer), concerniente a las elecciones presidenciales 2013, utilizando mapas geoespaciales, con el fin de determinar cuánto influye la decisión de género en las elecciones, como se muestra en la Figura 1.20.

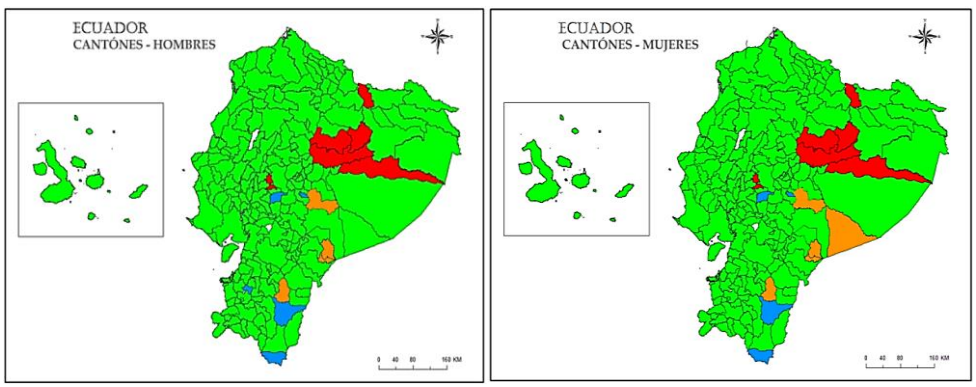

Figura 1.20. Resultados por cantones diferenciados por géneros

La última fue visualizar la cantidad total de votos nulos por cantones, con el propósito de establecer estrategias para campañas políticas, utilizando diagrama de árboles. Para tales efectos se utilizaron la aplicación gratuita en línea Many Eyes y el software comercial TreeMap. Para la visualización de datos geoespaciales, se manejó el software libre Quantum GIS. Estas estrategias permitieron lograr visualizaciones comprensibles y dinámicas para la toma de decisiones para candidatos, consultores, encuestadores y partidos políticos, en cara a futuras elecciones, como se muestra en la Figura 1.21.

Figura 1.21. Disgregación del voto, cantones por provincia. Tomado de [8]

El presente trabajo basará su análisis, a diferencia del trabajo expuesto anteriormente, en el discurso político que los candidatos presidenciales expresan en Twitter, siendo esta red social una fuente de datos de gran utilidad. Para obtener datos correctos para mapearlos a una visualización se aplica previamente una técnica de muestreo de las palabras u oraciones a vectores, generando una relación entre ellas aplicando el concepto de índice de similitud.

El índice de similitud o SIM es el valor de la similitud semántica existente entre dos palabras. Viene defino en un rango de entre 0 y 1, mientras más próximo a 1 sea su valor mayor será la similitud existente entre las palabras en cuestión.

La obtención de este valor se basa en el concepto de similitud de coseno, que, para este caso de estudio, trata a las palabras como vectores y obtiene la similitud entre ellas calculando el coseno existente entre ellos.

En la actualidad existen distintos algoritmos que realizan este proceso como es Word2Vec. Este algoritmo permite transformar las palabras o frases del lenguaje natural a vectores de números reales.

1.4.8. HERRAMIENTAS DE VISUALIZACIÓN

1.4.8.1. D3.JS

D3.js es una librería escrita en el lenguaje de programación interpretado Javascript para manipular documentos basándose en datos. Permite realizar representaciones visuales, dinámicas e interactivas en navegadores web, utilizando HTML5, CSS3 y SVG, de manera de manipular el DOM ajustándolo a los datos deseados.

Emplea una técnica llamada chain syntax, en la cual los métodos son encadenados entre sí con puntos, permitiendo realizar varias acciones en una sola línea.

Su primera versión se remonta al año 2011, es desarrollada y mantenida por Mike Bostock. Su desarrollo comenzó en 2009, primero con la librería Protovis que luego quedó obsoleta para continuar con d3.js. Es una de las herramientas más utilizadas hoy en día para la generación de visualizaciones en la web.

Entre sus principales funcionalidades se destacan:

- *Forces:* Característica que permite la creación de grafos dirigidos con aplicación de fuerzas utilizando integración del método de Verlet con velocidad. Se utiliza con frecuencia para calcular trayectorias de partículas en simulaciones de dinámica molecular y gráficos computacionales.
- *Geographies:* Característica que permite crear proyecciones geográficas a partir de datos de entrada.
- *Dragging:* Característica que permite drag and drop de elementos SVG, HTML o Canvas con el ratón o pantalla táctil.
- *Zoom:* Característica que permite el encuadre y zoom de elementos SVG, HTML o Canvas con el ratón o la pantalla táctil.
- *Voronoi Diagrams:* Característica que permite la construcción de un diagrama de Voronoi para un conjunto de puntos dados. El diagrama de Voronoi de un conjunto de puntos en el plano es la división de dicho plano en regiones, de tal forma, que a cada punto le asigna una región del plano formada por los puntos que son más cercanos a él que a ninguno de los otros objetos.

1.4.8.2. RAPHAËL

Raphaël 2 es una pequeña biblioteca de JavaScript que funciona con gráficos vectoriales en la web para la generación de visualizaciones. Raphaël usa SVG como base para crear gráficos, por lo que cada objeto gráfico que cree también es un objeto DOM, puede adjuntar controladores de eventos de JavaScript o modificarlos más tarde. El objetivo de Raphaël es proporcionar un adaptador que haga que el dibujo compatible con arte vectorial sea compatible con todos los navegadores.

1.4.8.3. PROCESSING.JS

Processing.js se basa en el popular lenguaje de programación Processing, diseñado para la web mediante el cual se pueden generar visualizaciones. Processing.js hace que las visualizaciones de datos, arte digital, animaciones interactivas, gráficos educativos, videojuegos, etc. funcionen utilizando estándares web y sin ningún complemento. Desarrollado originalmente por Ben Fry y Casey Reas, Processing comenzó como un lenguaje de programación de código abierto basado en Java para ayudar a las comunidades de arte electrónico y diseño visual a aprender los conceptos básicos de la programación de computadoras en el contexto de arte visual. Processing.js permite que el código de procesamiento sea ejecutado por cualquier navegador compatible con HTML5.

1.4.8.4. PREFUSE

Prefuse4 es un entorno de software basado en Java que permite la creación interactiva de aplicaciones de visualización de información. Se puede utilizar para crear aplicaciones independientes, componentes visuales y applets. Pretende simplificar los procesos de visualización, control y asignación de datos, así como la interacción del usuario. Permite la navegación rizoma siendo este un método para crear dinámicamente una interfaz de navegación para sistemas de datos, tales como bases de datos y sitios web en el que los vínculos de navegación que se presentan al usuario no están predefinidos, sino que se generan en respuesta al comportamiento de los usuarios y al análisis de otros datos.

2. METODOLOGÍA

Este capítulo está conformado por tres secciones. En la sección 2.1 se tiene el levantamiento de la información, los resultados de la encuesta realizada y los requerimientos obtenidos tras el análisis de esta. En la sección 2.2 se tiene el diseño de la aplicación web, en la cual se incluye el diseño tantos de las distintas visualizaciones como del aplicativo que alojará mismas. En la sección 2.3 se explica a detalle cómo se realiza la implementación del prototipo.

2.1. LEVANTAMIENTO DE INFORMACIÓN

2.1.1. ESTRUCTURA DE LA ENCUESTA

En la Tabla 2.1 se mostrará de manera resumida la estructura de la encuesta realizada a distintos usuarios. Las preguntas planteadas se repiten para cada uno de los candidatos que serán parte de este aplicativo.

La encuesta consta de 13 preguntas por cada candidato, siendo 7 los seleccionados para este prototipo y una pregunta final general. Para la explicación se tomará como ejemplo al candidato Guillermo Lasso.

Tabla 2.1. Descripción de las preguntas planteadas en la encuesta

N°	PREGUNTA	FORMATO
1	¿Cuánto considera que sabe acerca de las distintas temáticas que plantea en el discurso político el candidato presidencial GUILLERMO LASSO?	Pregunta de múltiples opciones: • Nada • Poco • Más o menos • Bastante
2	¿Qué habla GUILLERMO LASSO acerca del AMBIENTE?	Pregunta abierta
3	¿Qué habla GUILLERMO LASSO acerca de la CORRUPCIÓN?	Pregunta abierta
4	¿Qué habla GUILLERMO LASSO acerca de los DERECHOS?	Pregunta abierta
5	¿Qué habla GUILLERMO LASSO acerca de la ECONOMÍA?	Pregunta abierta
6	¿Qué habla GUILLERMO LASSO acerca de la EDUCACIÓN?	Pregunta abierta
7	¿Qué habla GUILLERMO LASSO acerca de la ENERGÍA?	Pregunta abierta
8	¿Qué habla GUILLERMO LASSO acerca de la JUVENTUD?	Pregunta abierta
9	¿Qué habla GUILLERMO LASSO acerca de la POBREZA?	Pregunta abierta
10	¿Qué habla GUILLERMO LASSO acerca de la SALUD?	Pregunta abierta
11	¿Qué habla GUILLERMO LASSO acerca del TRABAJO?	Pregunta abierta
12	¿Qué habla GUILLERMO LASSO acerca de la VIOLENCIA?	Pregunta abierta
13	¿Qué habla GUILLERMO LASSO acerca de la VIVIENDA?	Pregunta abierta
S/N	¿Aparte de los candidatos expuestos, liste el/los candidatos presidenciales con los que usted esté familiarizado con su discurso político.	Pregunta general abierta

El objetivo de la primera pregunta es tener un indicio de cuanta información tiene el usuario sobre el discurso político en general del candidato en cuestión.

De la pregunta 2 a la 12 están enfocadas al conocimiento que tiene el usuario sobre el discurso político de los distintos candidatos específicamente sobre categorías o temas de interés público.

La última pregunta no tiene diferenciador por candidato, el objetivo de la misma es tener un indicio acerca de la existencia de otro candidato/s de los cuales el usuario tenga conocimiento del discurso político.

2.1.2. ANÁLISIS DE LOS RESULTADOS DE LA ENCUESTA

En base a la respuesta de 20 usuarios encuestados, se tiene una conclusión dispersa acerca de cuanto conocen sobre los distintos candidatos y cuáles son las categorías de mayor interés y/o conocimiento.

Las respuestas de la primera pregunta se enfocan principalmente en un conocimiento mayor del discurso político de dos candidatos destacados: Guillermo Lasso y Andrés Arauz, siendo estos los de mayor interacción, alcance y aceptación dentro del contexto político y social ecuatoriano, además de ser los representantes de las dos corrientes políticas que han dominado las elecciones ecuatorianas por más de una década [9].

Mientras para el resto de los candidatos los usuarios demuestran poco conocimiento sobre su discurso político.

Para las preguntas de la 2 a la 12, en relación a los resultados de la primera pregunta, los usuarios tienen un conocimiento mayor sobre el discurso político enfocado a las categorías de: corrupción, educación y economía, mientras para el resto de las categorías hay poca o nula información.

Para la última pregunta, en base a las respuestas obtenidas, no se tiene otro candidato aparte de los presentados en la encuesta de los cuales el usuario tenga información o conocimiento sobre su discurso político.

2.1.3. REQUERIMIENTOS DEL PROTOTIPO

En la sección anterior se analizó las respuestas obtenidas en la encuesta realizada a distintos usuarios. En base a esta información se definen los distintos requerimientos que tendrá el prototipo. En la Tabla 2.2 se muestran los requerimientos funcionales del prototipo, mientras que en la Tabla 2.3 se muestran los requerimientos no funcionales del prototipo

Tabla 2.2. Requerimientos funcionales del prototipo

ID	REQUERIMIENTO	DESCRIPCIÓN
RF001	Enfoque del discurso político de los candidatos presidenciales.	El usuario requiere una plataforma donde pueda saber a qué tema o categoría de interés social se enfoca en mayor proporción el o los candidatos presidenciales.
RF002	Comparación entre candidatos presidenciales.	El usuario requiere una plataforma en donde pueda comparar los discursos políticos de los distintos candidatos presidenciales.
RF003	Diferenciación en segmentos de tiempo del discurso político del candidato presidencial.	El usuario requiere una plataforma en donde pueda tener una diferenciación entre espacios de tiempo sobre el discurso político de los candidatos presidenciales. Ejm: Discurso político durante la campaña presidencial y discurso político presidencial luego de la misma.

Tabla 2.3. Requerimientos no funcionales del prototipo

ID	Requerimiento	Descripción
RNF001	Presentación	La interfaz web debe ser amigable e intuitiva para el usuario.
RNF002	Acceso a través de la web	El prototipo debe ser accesible a través de internet.

2.1.4. ARQUITECTURA DEL PROTOTIPO

El prototipo constará únicamente de una capa de presentación en la cual el usuario podrá hacer uso e interactuar con el mismo en una serie de pestañas personalizadas, acorde a los requerimientos funcionales y no funcionales establecidos en las tablas anteriores.

2.2. DISEÑO

En esta sección se definiría el diseño del prototipo para solventar tanto los requerimientos funcionales como no funcionales detallados en la sección anterior. Además, se presentará el diagrama de casos de uso del prototipo. Para el desarrollo se usará la metodología de desarrollo de software Kanban mediante el uso del tablero que lleva su mismo nombre.

El tablero Kanban es una herramienta que permite la organización visual de las distintas tareas que compondrán el desarrollo total de un proyecto. Entre sus principales características se encuentran [10]:

- Limitar el trabajo en curso
- Visualizar el flujo de trabajo.

- Medir y optimizar el flujo de trabajo.
- Gestionar el trabajo cuantitativamente.

Un tablero Kanban en su definición más básica es una tabla o diagrama compuesta de columnas y cada columna se compone de un número limitado de tareas. Las columnas representan el estado actual en el que se encuentra cada tarea que compone el proyecto.

Para el desarrollo de este prototipo, al basarse únicamente en las actividades que realizará una sola persona se implementará un tablero con tres estados (columnas) diferentes: por hacer, en progreso y realizadas, las cuales proporcionarán un flujo de trabajo más organizado.

2.2.1. TABLERO KANBAN

Al inicio de la sección de diseño, como se puede evidenciar en la Tabla 2.4, el proyecto está en una fase inicial, donde se encuentran finalizadas tareas enfocadas principalmente al análisis de la problemática mediante la realización de encuestas y la definición de los requerimientos que tendrá el prototipo a desarrollarse.

Tabla 2.4. Tablero de Kanban al inicio de la fase de diseño

POR HACER	EN PROGRESO	REALIZADAS
Elaborar el diagrama de casos de uso.	Analizar el archivo de datos proporcionado	Analizar las herramientas de desarrollo necesarias para la implementación del prototipo.
Diseñar y el mapeo de datos a objetos visuales adecuado para las distintas visualizaciones definidas.		Realizar encuesta a usuarios potenciales del prototipo.
Instalación de las distintas herramientas para la implementación del prototipo		Analizar los resultados de la encuesta planteada
Codificar las distintas visualizaciones definidas		Definir las visualizaciones acordes a requerimientos funcionales.
Desarrollar la interfaz web donde se alojarán las gráficas codificadas.		
Seleccionar y configurar del host donde se alojará el aplicativo web.		

Seleccionar de dominio personalizado para la aplicación web.		
Realizar pruebas de validación de funcionalidad.		
Realizar pruebas de cada funcionalidad del prototipo mediante pruebas de usuario.		
Corregir errores detectados.		
Publicar prototipo en la nube con el dominio personalizado.		

2.2.2. DIAGRAMA DE CASOS DE USO

En la Figura 2.1 se muestra el diagrama de casos de uso del prototipo, en este diagrama se puede apreciar las acciones que puede hacer el usuario dentro de la aplicación web.

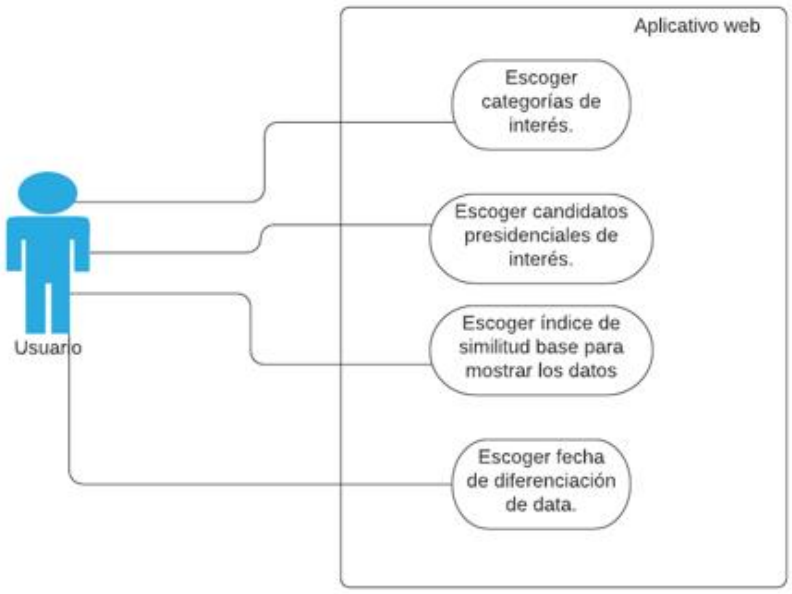

Figura 2.1. Diagrama de casos de uso del prototipo

El diagrama de casos de uso se presenta de una manera simple y concreta debido a que el prototipo va enfocado mayormente en las conclusiones que pueda generar el usuario a partir de la interacción con el prototipo. La facilidad de uso del prototipo se ve reflejado en las pocas acciones que debe realizar el usuario para poder obtener, a partir de un conjunto de datos extenso, información filtrada y acertada respecto al discurso político de los distintos candidatos. Dichas acciones se basan en la interacción con los filtros aplicables a cada una de las visualizaciones por desarrollarse.

El rol de administrador o desarrollador no aplica en este prototipo debido que, una vez desarrollado el mismo, este funcionará sin necesidad de que una persona en concreto tenga el control sobre el mismo, es decir, el software funcionara autónomamente a partir de la finalización del mismo.

2.2.3. DISEÑO DE VISUALIZACIONES

En esta sección se realizará el diseño de las tres visualizaciones que solventarán los requerimientos funcionales del prototipo haciendo énfasis en el mapeo a objetos visuales que se implementará para cada una de las visualizaciones.

2.2.3.1. GENERALIDADES

En esta sección se especifica aspectos generales que tendrán las tres visualizaciones a diseñar, siendo el archivo CSV proporcionado y sus data la fuente primordial de información para mapear a objetos visuales.

2.2.3.1.1. Formato de archivo CSV

El archivo necesario para la implementación del presente proyecto es proporcionado por el área de investigación de la Escuela Superior Politécnica del Litoral “Inferencia de conocimiento a partir de corpus de texto y metadata de publicaciones científicas y textos políticos en Latinoamérica usando técnicas de Procesamiento de Lenguaje Natural” mediante un software propietario el cuál toma los tweets de los diferentes candidatos presidenciales extrayendo palabras claves de los mismos y relacionándolos a diferentes categorías de interés social mediante el índice de similitud (SIM) el cual nos da valores cercanos a 1 si la palabra se relaciona en mayor proporción a la categoría y cercanos a 0 mientras menos relación tengan.

El archivo tiene un formato de cinco columnas, como se detalla en la siguiente tabla:

Tabla 2.5. Descripción de las columnas pertenecientes al archivo CSV

COLUMNA	DESCRIPCIÓN
Categoría	Categoría a la que se relaciona la palabra extraída.
Candidato	Candidato presidencial al cual le pertenece el tweet del que fue extraída la palabra.
Palabra (word)	Palabra extraída del tweet del candidato presidencial.
SIM	Índice de similitud entre la palabra y la categoría.
Fecha	Fecha de publicación del tweet.

Mientras que en la Figura 2.2 se tiene un extracto del archivo como ejemplo para facilitar el entendimiento general.

categoria	candidato	word	sim	fecha
vivienda	lasso	acceder	0,4696	16/3/2021
vivienda	lasso	creación	0,4133	17/3/2021
vivienda	lasso	empresas	0,3862	18/3/2021
vivienda	lasso	amazónicos	0,0153	19/3/2021

Figura 2.2. Extracto del archivo CSV

2.2.3.1.2. Categorías

Se ha seleccionado una lista de 12 categorías las cuales son de importancia social y, además, de gran impacto dentro de una campaña política, siendo las siguientes:

- Vivienda
- Violencia
- Trabajo
- Salud
- Pobreza
- Juventud
- Energías
- Educación
- Economía
- Derechos
- Corrupción
- Ambiente

2.2.3.1.3. Candidatos

Para este ítem se ha seleccionado a los candidatos más relevantes y de mayor impacto en el proceso de elecciones presidenciales 2020-2021 de la República del Ecuador. A continuación, se listan los candidatos elegidos:

- Guillermo Lasso
- Andrés Arauz
- Lucio Gutiérrez
- Yaku Pérez
- Gustavo Larrea
- Xavier Hervas

2.2.3.2. DISEÑO DE VISUALIZACIONES

En esta sección se presenta el diseño que tendrán cada una de las tres visualizaciones enfocándose en la solución de los requerimientos funcionales planteados previamente. En la Tabla 2.6 se presenta un resumen de los requerimientos que solucionará cada visualización.

Tabla 2.6. Resumen de los requerimientos aplicables a cada visualización

Requerimientos / ***Visualización***	*RF001*: Enfoque del discurso político de los candidatos presidenciales.	*RF002*: Comparación entre candidatos presidenciales.	*RF003*: Diferenciación en segmentos de tiempo del discurso político del candidato presidencial.
Visualización "Tabla de tendencia del discurso político"	X	X	
Visualización "Red de tendencia del discurso político"	X	X	
Visualización "Comparativa de tendencia del discurso político"	X	X	X

2.2.3.2.1. Tabla de tendencia del discurso político

La primera visualización basa su diseño en el tipo tabla de datos, esto debido a la facilidad de entendimiento del usuario ante este tipo de visualizaciones al ser manejado ampliamente en distintos ámbitos y de la cual es sencillo obtener conclusiones precisas respecto a la información presentada. Además, basándose en el mantra *"Overview first, zoom and filter, then details-on-demand"* [11], se definen para esta visualización las operaciones:

- Filter: La visualización permitirá obtener un subconjunto de datos en base a los distintos filtros categoría, candidato e índice de similitud.
- Details-on-demand: La visualización permitirá obtener información oculta mediante el uso del ratón en interacción con las palabras pertenecientes a cada categoría. Al pasar el ratón por encima de una palabra esta presentará numéricamente el índice de similitud que esta tiene.

El diseño de la primera visualización se enfoca en la solución a los requerimientos funcionales RF001 y RF002 los cuales requieren mostrar el discurso político enfocándolo a las categorías descritas teniendo la capacidad de mostrar a cuál de ellas se inclina la tendencia del discurso además de tener la capacidad de hacer una comparación entre candidatos presidenciales en categorías específicas.

A la tabla de datos se le añade interactividad proporcionándole 3 distintos filtros: candidato, categoría y SIM, mediante los cuales la tabla irá cambiando su estructura a medida que el usuario aplique o cambie las variables de sus filtros.

Figura 2.3. Filtros aplicables a la primera gráfica

En la Figura 2.3 se muestran los demos de los filtros aplicables para esta visualización.

Mapeo a objetos visuales

Para este ejercicio se ocupará únicamente las columnas: candidato, categoría, palabra (word), SIM. Cada una de estas se representarán de distinta forma dentro de la visualización. En la Figura 2.4, se muestra el esquema que se usará para la implementación de esta.

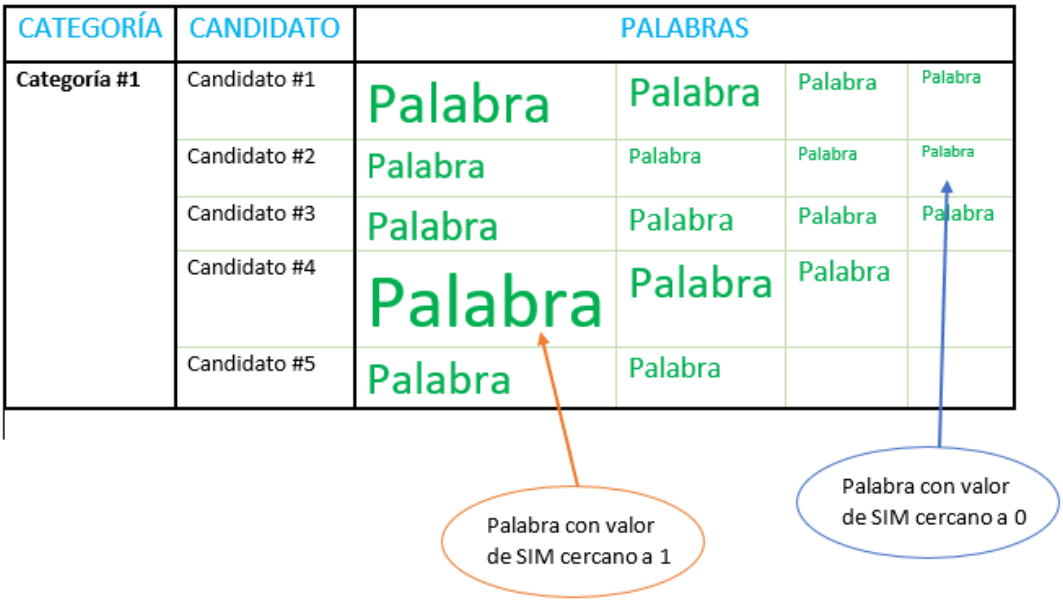

Figura 2.4. Esquema de diseño de la primera visualización

A continuación, se muestra el boceto de la primera visualización.

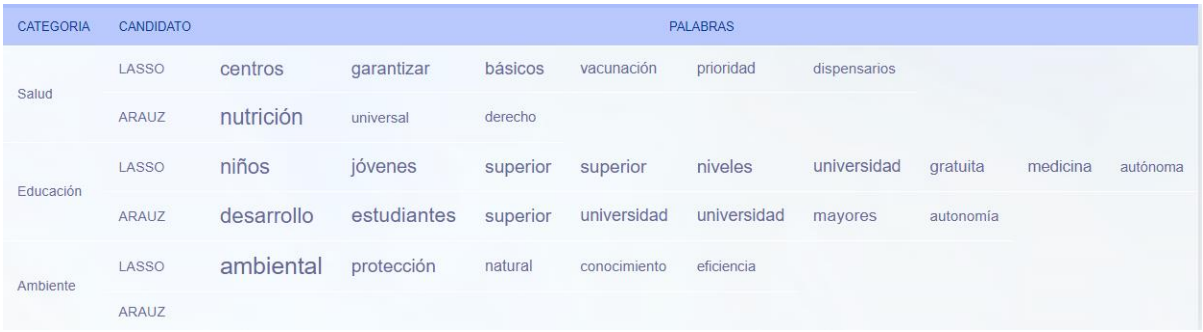

Figura 2.5. Boceto de la primera visualización

Con esta visualización se diferencia las categorías de las que habla en mayor proporción un candidato y que tan relacionado está lo que menciona en sus tweets respecto a la categoría en cuestión, además de poder hacer una comparación rápida entre los distintos candidatos.

2.2.3.2.2. Red de tendencia del discurso político

La segunda visualización se enfoca en un diseño dirigido por la fuerza, específicamente una red de datos, debido a la versatilidad que este tipo de visualización brinda, permitiendo modificar las características físicas de los elementos que la componen como son, color, tamaño, distancia de separación, entre otros y representando información específica en cada una de ellas. Además, conforme al tipo de información obtenida, es idóneo representar la relación que existen entre los datos mediante una red.

Basándose en el mantra *"Overview first, zoom and filter, then details-on-demand"* [11], se definen, para esta visualización, las operaciones:

- Overview: La visualización permitirá una vista general de todos los datos disponibles.
- Filter: La visualización permitirá obtener un subconjunto de datos en base a los distintos filtros categoría, candidato e índice de similitud.
- Details-on-demand: La visualización permitirá obtener información oculta mediante el uso del ratón en interacción con las palabras pertenecientes a cada categoría. Al pasar el ratón por encima de una palabra esta presentará el candidato a la cual pertenece y al pasar el ratón por encima del enlace que conecta la palabra con su categoría presentará el índice de similitud que tiene la palabra.

El diseño de esta visualización tiene el mismo objetivo que la presentada anteriormente, dar solución a los requerimientos funcionales RF001 y RF002, pero con un mapeo distinto, teniendo la capacidad de implementar los mismos filtros de la Figura 2.3.

A esta visualización se le adaptarán 3 tipos de filtro, los mismos que se aplican para la primera visualización, lo que le brindará mayor interactividad al usuario adaptándose a sus necesidades.

Mapeo a objetos visuales

Para este ejercicio se usará la data únicamente de las columnas: candidato, categoría, palabra (word) y SIM. Cada una de estas será representada de una manera específica dentro de la visualización. En la Figura 2.6 se puede ver el esquema de diseño para la visualización.

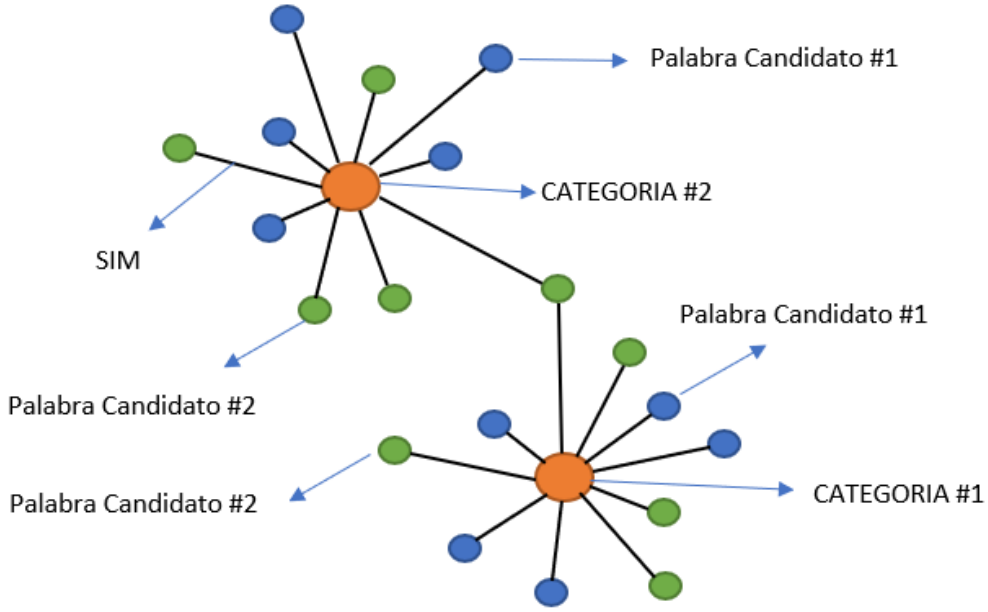

Figura 2.6. Esquema de diseño de la segunda visualización

Además, en la Tabla 2.7 se indica el resumen del mapeo a implementarse dependiendo del tipo de dato.

Tabla 2.7. Mapeo a objetos visuales por tipo de datos de la segunda visualización

COLUMNA	OBJETO VISUAL
Categoría	Nodos principales (Nodo central de mayor tamaño)
Palabra (word)	Nodos secundarios (Nodos periféricos de menor tamaño)
Candidato	Color de los nodos secundarios
SIM	Enlace entre nodo principal – nodo secundario. (Enlace más pequeño en longitud representa un valor de SIM mayor y viceversa)

A continuación, se muestra el boceto de la segunda visualización.

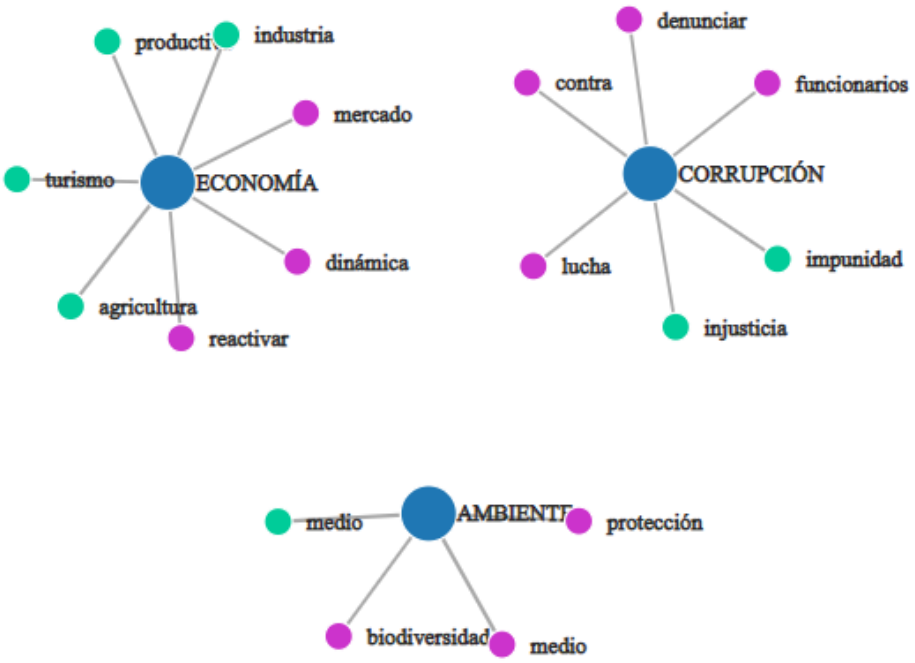

Figura 2.7. Boceto de la segunda visualización

Mediante esta visualización se puede obtener información acerca de las categorías de las que habla más un candidato y que tan relacionado está lo que menciona con la categoría en cuestión. Además, se puede realizar una comparativa a nivel de categorías y candidatos. Esta versatilidad de la visualización le da al usuario la facilidad de obtener varias conclusiones a partir de la misma visualización.

2.2.3.2.3. Comparativa de tendencia del discurso político

La tercera visualización, al igual que la segunda, se basa en un diseño dirigido por la fuerza. Para acoplar la red al diseño esperado para la representación de datos se modificará una de las características representativas de este tipo de visualización como lo es la posición de los nodos dentro de la misma.

Basándose en el mantra *"Overview first, zoom and filter, then details-on-demand"* [11], se definen, para esta visualización, las operaciones:

- Overview: La visualización permitirá una vista general de todos los datos disponibles.
- Filter: La visualización permitirá obtener un subconjunto de datos en base a los distintos filtros categoría, candidato, índice de similitud y fecha de comparación.
- Details-on-demand: La visualización permitirá obtener información oculta mediante el uso del ratón en interacción con las palabras pertenecientes a cada categoría. Al

pasar el ratón por encima de una palabra esta presentará el candidato a la cual pertenece y al pasar el ratón por encima del enlace que conecta la palabra con su categoría presentará el índice de similitud que tiene la palabra.

El diseño de la tercera visualización se enfoca en la solución de los requerimientos funcionales RF001, RF002 y en especial del RF003. Para los primeros dos requerimientos se plantea un diseño similar al de las dos visualizaciones presentadas anteriormente, pero con una distribución de sus elementos diferente. Para el último requerimiento.

Esta visualización se diferencia de la segunda en la ubicación en que se presentan los objetos que la componen, esta alteración de la visualización tiene como objetivo agregar una característica más como es la fecha de publicación de los diferentes tweets de los cuales se sacó la data utilizada. Esta visualización será capaz de interpretar cuatro filtros, los cuales se aprecian en la Figura 2.8.

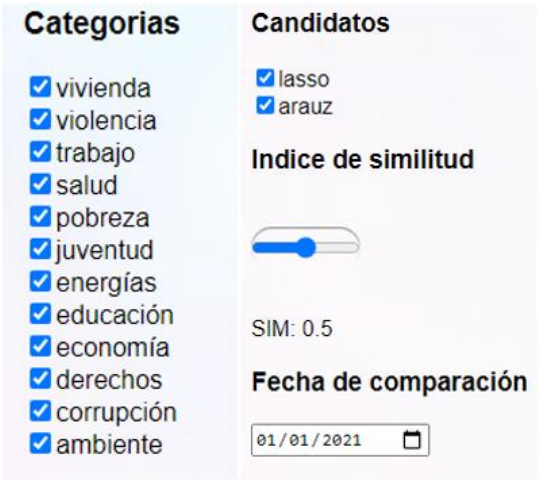

Figura 2.8. Filtros aplicables a la tercera gráfica

Mapeo a objetos visuales

Para este ejercicio se usará la data de las columnas: candidato, categoría, palabra (word), SIM y Fecha. Cada una de estas será representada de una manera específica dentro de la visualización. En la Figura 2.9 se puede ver el esquema de diseño para la visualización. En la Figura 2.10 se puede observar el esquema de funcionamiento del índice de similitud en base al filtro de fecha de corte.

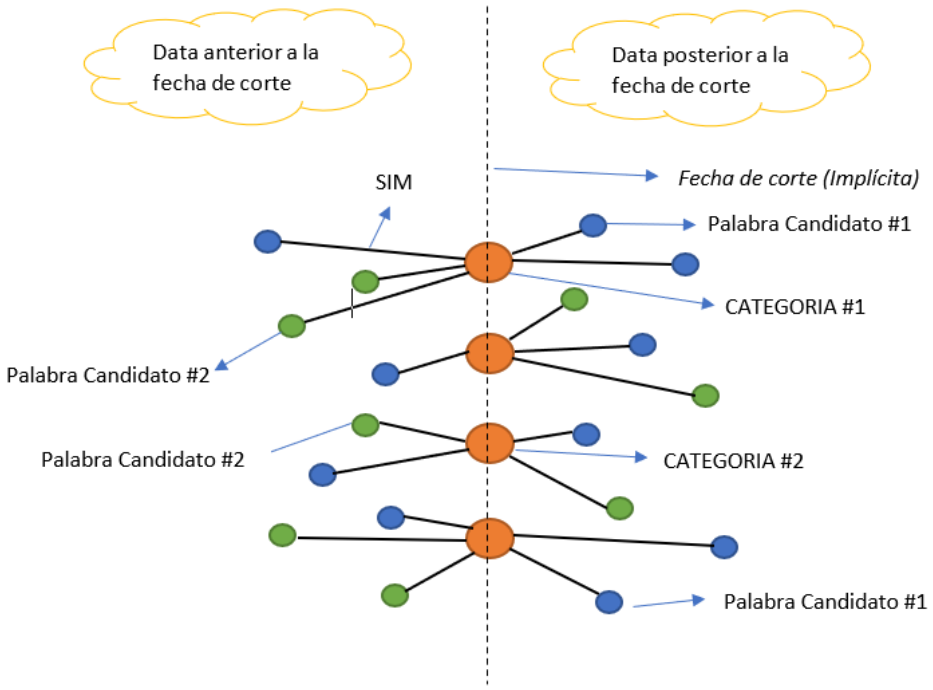

Figura 2.9. Esquema de diseño de la segunda visualización

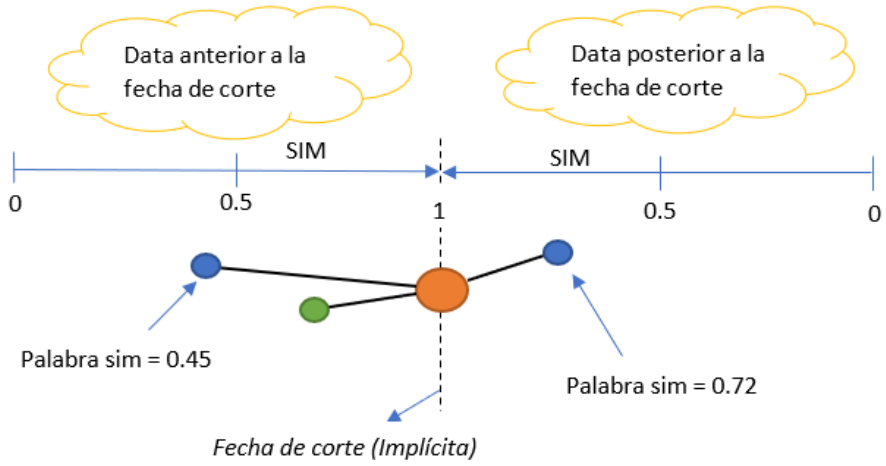

Figura 2.10. Esquema de funcionamiento de la SIM respecto a la fecha de corte

Además, en la siguiente tabla se indica el resumen del mapeo a implementarse dependiendo del tipo de dato.

Tabla 2.8. Mapeo a objetos visuales de los tipos de datos de la tercera visualización.

COLUMNA	OBJETO VISUAL
Categoría	Nodos principales (Nodo central de mayor tamaño)
Palabra (word)	Nodos secundarios (Nodos periféricos de menor tamaño)
Candidato	Color de los nodos secundarios
SIM	Enlace entre nodo principal – nodo secundario. (Enlace más pequeño en longitud representa un valor de SIM mayor y viceversa)
Fecha	Límite vertical implícito. Centro de la visualización

A continuación, en la Figura 2.11 muestra el boceto de la tercera visualización

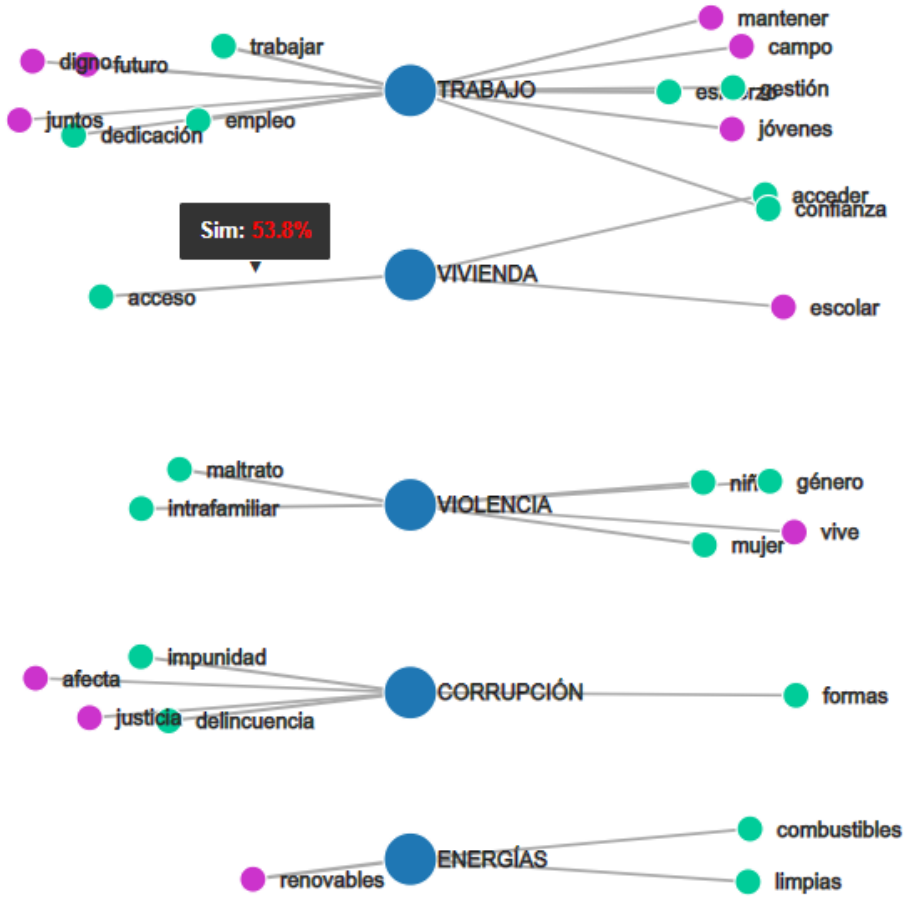

Figura 2.11. Boceto de la tercera visualización

Mediante esta visualización se puede obtener información acerca de cuán relacionado estuvo lo que mencionaron los candidatos políticos acerca de las diferentes categorías en su discurso político, personalizando la visualización con una fecha de corte a elección del usuario, lo que permite saber el enfoque del discurso político antes y después de una fecha específica, por ejemplo: antes de votaciones y después de votaciones o antes de primera vuelta y después de primera vuelta. Además, permite hacer una comparativa entre categorías y/o candidatos tomando en cuenta los factores antes mencionados.

2.3. IMPLEMENTACIÓN

En esta sección se presenta el proceso de implementación del prototipo de aplicación web en base al diseño especificado en la sección 2.2. Incluye el proceso de instalación de los

recursos necesarios para la codificación, implementación de las tres visualizaciones, implementación de la página web que alojará las tres visualizaciones y proceso de publicación de la página web en la nube.

2.3.1. ACTUALIZACIÓN DEL TABLERO KANBAN

Una vez finalizada la fase de diseño se tiene un avance significativo respecto al proyecto, teniendo concluidas tareas como la elección de las visualizaciones y el diseño de las mismas en base a mapeos a objetos visuales específicos, las cuales son importantes dentro de la ejecución del proyecto para solventar los objetivos del mismo. Se tienen pendientes tareas enfocadas principalmente al desarrollo técnico del prototipo.

Tabla 2.9. Tablero de Kanban al inicio de la fase de implementación.

POR HACER	EN PROGRESO	REALIZADAS
Codificar las distintas visualizaciones definidas	Instalación de las distintas herramientas para la implementación del prototipo	Analizar las herramientas de desarrollo necesarias para la implementación del prototipo.
Desarrollar la interfaz web donde se alojarán las gráficas codificadas.		Realizar encuesta a usuarios potenciales del prototipo.
Seleccionar y configurar del host donde se alojará el aplicativo web.		Elaborar el diagrama de casos de uso.
Seleccionar de dominio personalizado para la aplicación web.		Analizar el archivo de datos proporcionado.
Realizar pruebas de validación de funcionalidad.		Definir las visualizaciones acordes a requerimientos funcionales.
Realizar pruebas de cada funcionalidad del prototipo mediante pruebas de usuario.		Diseñar y seleccionar el mapeo de datos a objetos visuales adecuado para las distintas visualizaciones definidas.
Corregir errores detectados.		
Publicar prototipo en la nube con el dominio personalizado.		

2.3.2. INSTALACIÓN DE LAS HERRAMIENTAS DE SOFTWARE

Al ser una aplicación web únicamente con capa de presentación la herramienta necesaria para la codificación de esta será un editor de código fuente, en este caso se utilizará Visual Studio Code. El manual de instalación se encuentra en la página oficial de la aplicación [12].

Para la implementación de las visualizaciones se requerirán distintas bibliotecas, además de complementos (plugins) existentes en Visual Studio Code que servirán para la correcta codificación del prototipo. En la Tabla 2.10 se detallan las herramientas antes mencionadas.

Tabla 2.10. Herramientas de software necesarias

NOMBRE	TIPO	DESCRIPCIÓN	VERSIÓN
Visual Studio Code	Aplicación	Editor de código fuente	1.59.1
D3.js	Biblioteca	Biblioteca basada en javascript para generar a partir de datos visualizaciones dinámicas e interactivas.	5.16.0
Lodash.js	Biblioteca	Biblioteca de javascript enfocada a la optimización en el uso de objetos y/o arreglos	4.17.11
Live Server	Plugin	Simulador de servidor local	5.6.1

Para el caso de d3.js es la biblioteca en la que se basará este prototipo para realizar la implementación de las visualizaciones. Lodash.js se aplicará para un mejor manejo del set de datos y el plugin Live Server se usa durante el proceso de implementación para simular un servidor donde está alojado el archivo, esto para evitar problemas con las normas de seguridad que tienen los distintos navegadores web.

2.3.3. GENERALIDADES

En esta sección se especificará la implementación de distintas funciones que son generales para las tres visualizaciones que se van a codificar, tal como la lectura del archivo o la filtración de categorías y candidatos que tiene la data proporcionada.

2.3.3.1. LECTURA DEL ARCHIVO CSV

Al ser un archivo en formato CSV (Archivo separado por comas), a nivel de lenguaje JavaScript existe diferentes maneras de leer un archivo de este tipo para poder manipularlo desde el código. Para este prototipo se utilizará una función propia de la biblioteca d3.js que permite un mejor acoplamiento con las demás funciones que se utilizarán para generar las distintas visualizaciones. La sintaxis de uso del comando es el siguiente [13]:

```
1 d3.csv(url[[, accessor], callback])
```

Código 2.1. Sintaxis de uso d3.js

Existen distintas variaciones de esta función en base al formato en que tenga el archivo, por ejemplo, la existencia o no de un encabezado. Para esta implementación se tiene un archivo que constan de un encabezado en el que se definen el nombre de las columnas

que tiene la data. En la Figura 2.12 se tiene un extracto de un archivo CSV con encabezado y en la Figura 2.12 se muestra el resultado que arroja la función a nivel de código.

```
Year,Make,Model,Length
1997,Ford,E350,2.34
2000,Mercury,Cougar,2.38
```

Figura 2.12. Extracto de archivo CSV con encabezado. Tomado de [13]

```
[
  {"Year": "1997", "Make": "Ford", "Model": "E350", "Length": "2.34"},
  {"Year": "2000", "Make": "Mercury", "Model": "Cougar", "Length": "2.38"}
]
```

Figura 2.13. Resultado de la función d3.csv() Tomado de [13]

En el Código 2.2 se muestra el fragmento de código del comando d3.csv() que se utilizará en la presente implementación.

```
function loadData(){
      return Promise.all([
            d3.csv("elecciones_final.csv")
      ]).then(datasets => {
            store.dat = datasets[0];
            return store;
      })
}
```

Código 2.2. Lectura del archivo CSV mediante D3

La Figura 2.14 muestra un extracto del arreglo obtenido una vez que el archivo CSV otorgado para este proyecto ha sido leído mediate la función descrita arriba, basándose en la estructura definida en la Tabla 2.5.

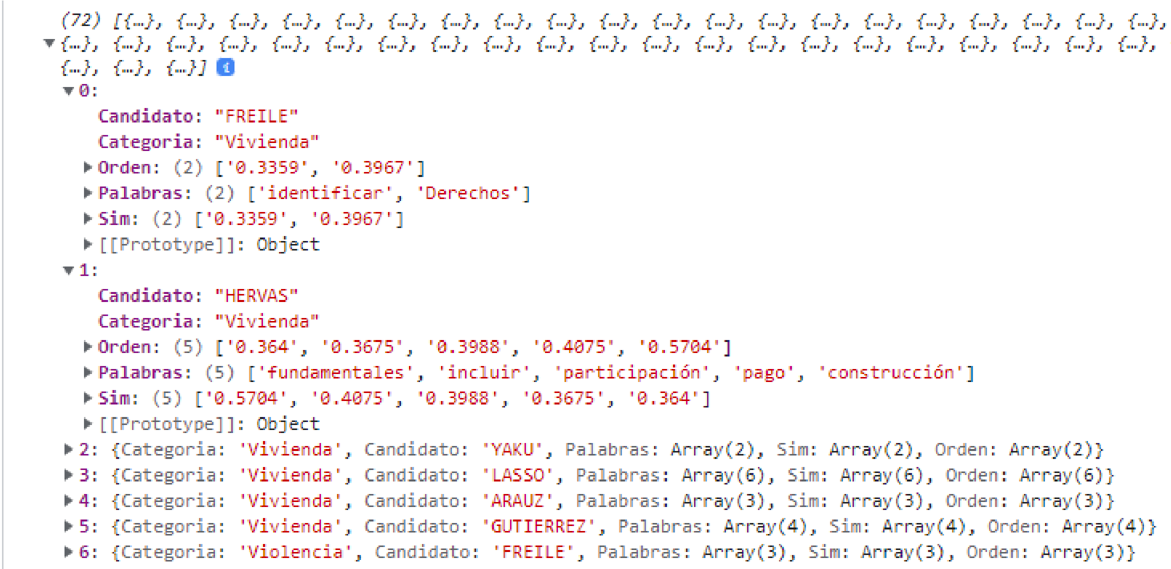

Figura 2.14. Resultado de la función d3.csv() en base al archivo CSV de entrada

2.3.3.2. OBTENCIÓN DE CATEGORÍAS Y CANTIDATOS

Se obtiene las categorías y candidatos a partir del archivo para mantener la autonomía de la aplicación web respecto al cambio del archivo que pueda darse en futuras implementaciones ya sea para otros candidatos, categorías o procesos electorales. De esta manera se evita la definición por defecto tanto de categorías como candidatos dentro del código. Para este fin se usa la función *reduce()* propia de JavaScript. Esta función, a diferencia de otras aplicables para cadenas y matrices, brinda un mejor performance en cuanto a iteración y generación de una nueva matriz o cadena respecto a otra de mayor tamaño.

La función ejecuta dentro de su definición una función reductora por valor de matriz, devuelve un único valor que es el acumulado de la función, no se ejecuta para elementos vacíos de la matriz y no altera la matriz original [14].

```
array.reduce(function(total, currentValue, currentIndex, arr), initialValue)
```

Código 2.3. Sintaxis de uso de la función reduce de JavaScript

En la Tabla 2.11 se detalla la descripción de los parámetros de la función, su obligatoriedad y descripción.

Tabla 2.11. Parámetros de la función reduce()

PARÁMETRO	ARGUMENTO	OBLIGATORIEDAD	DESCRIPCIÓN
function(total,currentValue,index,arr)	total	Si	Acumulador
	currentValue	Si	Valor del elemento actual.
	currentIndex	No	El índice de matriz del elemento actual.

	arr	No	Array sobre el cual se llamó el método.
initialValue		No	Valor que se pasará a la función como valor inicial.

En el Código 2.4 se muestra un ejemplo simple de uso de la función *reduce()* para arreglos de elementos, mientras que en el Código 2.5 se presenta la implementación para obtener los candidatos

```
const array1 = [1, 2, 3, 4];
const reducer = (acumulador, valorActual) => acumulador + valorActual;
//1 + 2 + 3 + 4
console.log (array1.reduce(reducer));
// resultado esperado: 10
```

Código 2.4. Ejemplo de uso de la función reduce()

```
function obtCand(datos){
    var aux=0;
    var result = datos.reduce((result,d) => {
        result = Object.keys(result).map(key => result[key])
        if(result != undefined){
           var checkRep = dataRepetidaCand(result, d.candidato);
            if(checkRep.length == 0){
                var temp = {"IdCand": aux + 1, "NomCand": d.candidato};
                result.push(temp);
                aux = aux + 1;
            }
        }
        return result;
    },{});
    result = Object.keys(result).map(key => result[key])
    candidatos = result;
    return result;
}
```

Código 2.5. Obtención de candidatos a partir del set de datos

En el Código 2.6 se muestra el código implementado en el prototipo para leer las categorías.

```
function obtCat(datos){
    var aux=0;
    var result = datos.reduce((result,d) => {
        result = Object.keys(result).map(key => result[key])
        if(result != undefined){
           var checkRep = dataRepetidaCat(result, d.categoria);
            if(checkRep.length == 0){
                var temp = {"CatID": aux + 1, "NomCat": d.categoria};
                result.push(temp);
                aux = aux + 1;
            }
        }
        return result;
    },{});
    categorias = result;
    return result;
}
```

Código 2.6. Obtención de categorías a partir del set de datos

2.3.3.3. CREACIÓN DE FILTROS DINÁMICOS

La creación de los filtros que se usarán en las visualizaciones obedece expresamente a la data que se obtendrá a partir del archivo, es decir, el aplicativo web será capaz de crear filtros a partir del archivo proporcionado lo que permite usar distintos archivos siempre y cuando obedezcan a la estructura detallada en la Tabla 2.5.

Para crear los filtros, tanto de candidatos como de categorías, se usan elementos HTML creados a partir de funciones JavaScript, lo que le proporciona el dinamismo deseado. Para este fin se usan las funciones detalladas en la Tabla 2.12.

Tabla 2.12. Funciones JavaScript utilizadas para la creación de filtros dinámicos

FUNCIÓN	DESCRIPCIÓN	SINTAXIS
createElement()	Crea elementos html mediante lenguaje JavaScript.	document.createElement(tagName, [options]);
innerHTML	Establece sintaxis HTML a un elemento HTML creado mediante JavaScript.	element.innerHTML = htmlString;

Los filtros, tanto de candidatos como de categorías se mostrarán en forma de checkbox, los cuales se crean en base al Código 2.7 destinado a categorías y el Código 2.8 destinado a candidatos.

```
1 function checkBCategorias(cat){
2     var aux = 0
3     cat.forEach(elemento => {
4       var node = document.createElement('div');
5       node.innerHTML = '<input type="checkbox" id="check'+aux+'" checked=""
onclick="isCheck()">
6        <label
for="check'+aux+'">'+elemento.NomCat.toUpperCase()+'</label><br>';
7       document.getElementById('cat').appendChild(node);
8       aux=aux+1;
9     })
10}
```

Código 2.7. Código de generación de filtros de categorías a partir del set de datos

```
1 function checkBCandidatos(cand){
2     var aux = 0
3     cand.forEach(elemento => {
4       var node = document.createElement('div');
5       node.innerHTML = '<input type="checkbox" id="checkC'+aux+'"
checked="" onclick="isCheck()">
6        <label
for="checkC'+aux+'">'+elemento.NomCand.toUpperCase()+'</label><br>';
7       document.getElementById('cand').appendChild(node);
8       aux=aux+1;
9     })
10}
```

Código 2.8. Código de generación de filtros de candidatos a partir del set de datos

La data que se mostrará en la visualización dependerá de los checkbox que sean seleccionados por el usuario. Los checkbox, como objetos HTML, tienen propiedades especificas basadas en su comportamiento. La propiedad *onclick* permite definir que función se ejecutará cada que el usuario de click sobre un checkbox cualquiera.

El

Código 2.9 muestra la función a ejecutarse a interacción con cualquier checkbox. Esta función creará dos arreglos en los cuales se especificará los candidatos y categorías que no han sido seleccionados, información que servirá para crear nuevamente la visualización omitiendo la data relacionada a los dos arreglos mencionados. La verificación de selección de un checkbox se hace mediante el uso de la propiedad *checked*.

```
function isCheck(){
    var aux;
    var aux1=0;
    var aux2;
    var aux3=0;
    var indexCat = [];
    var indexCand = [];
    for(aux = 0;aux < categorias.length;aux++){
       var isChecked = document.getElementById("check"+aux+"").checked;
          if(isChecked != true){
              indexCat[aux1] = categorias[aux].NomCat;
              aux1=aux1+1;
          }
      }
    for(aux2 = 0;aux2 < candidatos.length;aux2++){
       var isChecked2 = document.getElementById("checkC"+aux2+"").checked;
          if(isChecked2 != true){
              indexCand[aux3] = candidatos[aux2].NomCand;
              aux3=aux3+1;
          }
      }
     hide(indexCat,indexCand);
}
```

Código 2.9. Código ejecutable en la interacción con los checkbox

Además de los filtros que se tendrán basados en los candidatos y categorías que son dependientes del archivo proporcionado, tenemos un filtro que se basa en el índice de

```
<div class="slidecontainer">
    <h3>Indice de Similitud</h3>
    <input  type="range" min="0" max="1" step="0.005" value="0.33"
                    class="slider" id="myRange" onclick="actTabla()">
    <p>SIM: <span id="demo"></span></p>
</div>
```

similitud, este será visualmente definido como una slider entre 0 y 1 que dependiendo del valor que se escoja la gráfica mostrará los nodos (palabras) cuyo índice de similitud se encuentre por encima de dicho valor. La generación del mismo se realiza a partir del Código 2.10.

Código 2.10. Implementación del slider como filtro

2.3.4. VISUALIZACIÓN TABLA DE TENDENCIA DEL DISCURSO POLÍTICO

En esta sección se detallará la implementación de la primera visualización en base al diseño detallado en la sección 2.2.3.2. El lenguaje usado para este caso será JavaScript en conjunto con elementos HTML.

2.3.4.1. FORMATO DE LA INFORMACIÓN

Para esta visualización se da un formato específico al arreglo de objetos, basado en la data obtenida del archivo CSV, que se utilizará para generar la visualización. En la Tabla 2.13 se describen los atributos que tendrá cada objeto del arreglo planteado.

Tabla 2.13. Atributos pertenecientes a cada objeto del arreglo.

ATRIBUTOS	DESCRIPCIÓN
Candidato	Candidato al cual pertenece el array de palabras del objeto
Categoría	Categoría a la cual pertenece el array de palabras del objeto
Orden	Orden de los SIM de menor a mayor.
Palabras	Array de palabras ordenadas en base a su correspondiente SIM

Esta estructura permite tener en cada objeto, un arreglo de palabras ordenadas en base a su índice de similitud (SIM), diferenciadas por candidato y categoría. Lo que permitirá una mejor gestión de la información al momento de generar la tabla.

En la Figura 2.15 se tiene un extracto del arreglo, generado a partir de la data obtenida del archivo CSV, que será utilizado en la generación de la primera visualización.

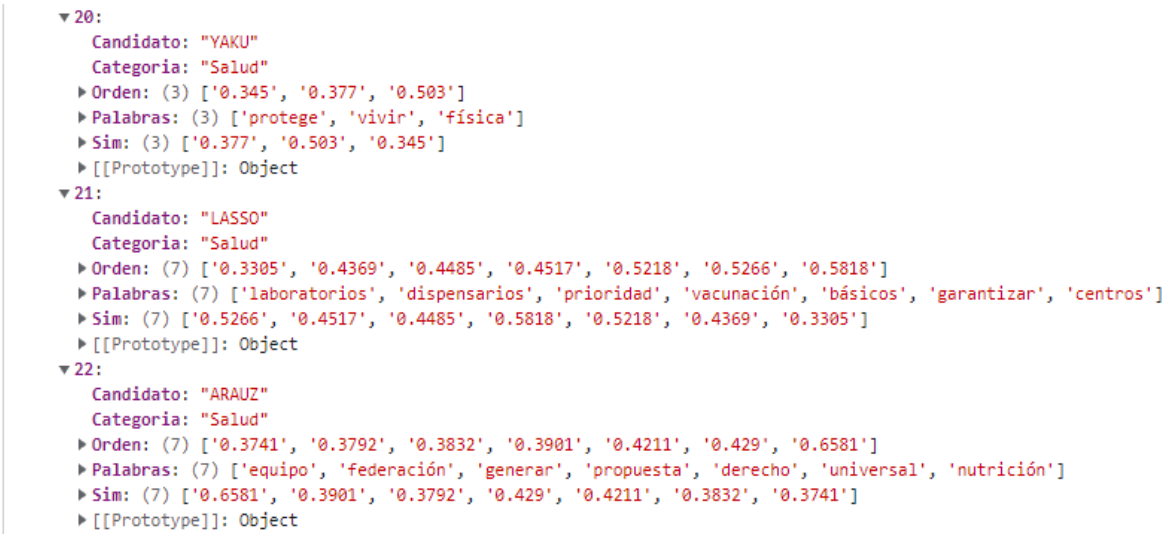

Figura 2.15. Estructura del arreglo para la primera visualización

2.3.4.2. CREACIÓN DE LA TABLA DINÁMICA

HTML consta de etiquetas básicas las cuales permiten al usuario crear una tabla básica y estática. Dichas etiquetas se resumen en la Tabla 2.14.

Tabla 2.14. Etiquetas para creación de una tabla HTML

```
<style>
    td, th{
        border: 1px solid Oblack;
    }
</style>
<table>
    <thead>
        <tr>
            <th>Columna 1</th>
            <th>Columna 2</th>
        </tr>
    </thead>
    <tbody>
        <tr>
            <td>Celda 1.1</td>
            <td>Celda 1.2</td>
        </tr>
        <tr>
            <td>Celda 2.1</td>
            <td>Celda 2.2</td>
        </tr>
    </tbody>
</table>
```

ETIQUETA	FUNCIÓN
<table>	Creación de una tabla
<thead>	Etiqueta contenedora de la cabecera de la tabla.
<tbody>	Etiqueta contenedora del cuerpo de la tabla.
<tr>	Creación de una fila
<th>	Creación de una celda perteneciente al encabezado de la tabla.
<td>	Creación de una celda perteneciente al cuerpo de la tabla.

En el Código 2.11 se presenta un ejemplo para una tabla básica basada en etiquetas HTML y en la Figura 2.16 se muestra el resultado del código planteado previamente.

Código 2.11. Ejemplo de creación de tabla HTML

Columna 1	Columna 2
Celda 1.1	Celda 1.2
Celda 2.1	Celda 2.2

Figura 2.16. Ejemplo de tabla creada con HTML

Para poder adaptar la idea de interactividad es necesario acoplar estos conceptos de etiquetas antes mencionados a un script desarrollado en lenguaje JavaScript que le permita

```
function dibujarTabla(dataPalabras){
    var numCND = 0;
    numCND = candidatos.length - lenghtCND;

    const $cuerpoTabla = document.querySelector("#cuerpoTabla");
    dataPalabras.forEach(pal => {
        const $tr = document.createElement("tr");
          let $tdCategoria = document.createElement("td");
          if(dataPalabras.indexOf(pal) % numCND == 0){
          $tdCategoria.textContent = pal.Categoria;
          $tdCategoria.rowSpan = numCND;
          $tr.appendChild($tdCategoria);
          }
          let $tdCand = document.createElement("td");
          $tdCand.textContent = pal.Candidato;
          $tr.appendChild($tdCand);
          var aux = pal.Sim.length;
          for (var i=pal.Palabras.length-1;i>=0;i--){
                  let $tdPalabras = document.createElement("td");
                  var tam = parseFloat(pal.Orden[aux-1])*35
                  $tdPalabras.style.fontSize = tam;
                  $tdPalabras.textContent = pal.Palabras[i];
                  $tdPalabras.title = "Sim: " +pal.Orden[aux-1]
                  aux--;
                  $tr.appendChild($tdPalabras);
          }
          $cuerpoTabla.appendChild($tr);
})
```

a la tabla agregar/quitar celdas dinámicamente y además cambiar las propiedades de su contenido acorde a los filtros que esta visualización permita.

La sintaxis de la función que permite acoplar etiquetas HTML a un script de JavaScript es la siguiente:

```
var element = document.createElement(tagName, [options]);
```

Código 2.12. Función para acoplar etiquetas HTML a un script JS

En el Código 2.13 se muestra la función de JavaScript definida para la creación de la tabla dinámica en base a elementos HTML y tomando el arreglo definido en la sección 2.3.4.1.

Código 2.13. Creación de la tabla dinámica para la primera visualización

En el Anexo A se muestra el código usado para generar la primera visualización.

2.3.5. VISUALIZACIÓN RED DE TENDENCIA DEL DISCURSO POLÍTICO

En esta sección se detallará la implementación de la segunda visualización en base al diseño detallado en la sección 2.2.3.2.2. El lenguaje usado para este caso será JavaScript haciendo uso de la biblioteca d3.js.

2.3.5.1. FORMATO DE LA INFORMACIÓN

La información a utilizarse en la generación de la segunda visualización será separada en dos arreglos: el primero que tendrá la información de todos los nodos tanto primarios como segundarios y el segundo que tendrá la información de los enlaces que existirán entre nodos.

La estructura que tendrán los dos arreglos se detalla en la Tabla 2.15.

Tabla 2.15. Estructura de los arreglos

Arreglo	Atributos por objeto (nodo) del arreglo	Descripción
Nodos (nodes)	Identificador (id)	Identificador secuencial del objeto.
	Palabra (label)	Palabra asociada al nodo.
	Grupo (group)	Grupo al que pertenece el nodo. (Primario o secundario)
	Candidato (cnd)	Candidato al cual pertenece la palabra.
Enlaces (edges)	Fuente (source)	Atributo tipo nodo. Especifica el nodo fuente del enlace.
	Destino (target)	Atributo tipo nodo. Especifica el nodo destino del enlace.
	SIM (weight)	Indica el peso del enlace. Se basa en el índice de similitud (SIM) que tenía el nodo secundario (palabra) con el nodo primario (categoría).

En la Figura 2.17 se muestra un extracto del arreglo de nodos que se utilizará en la generación de la segunda visualización.

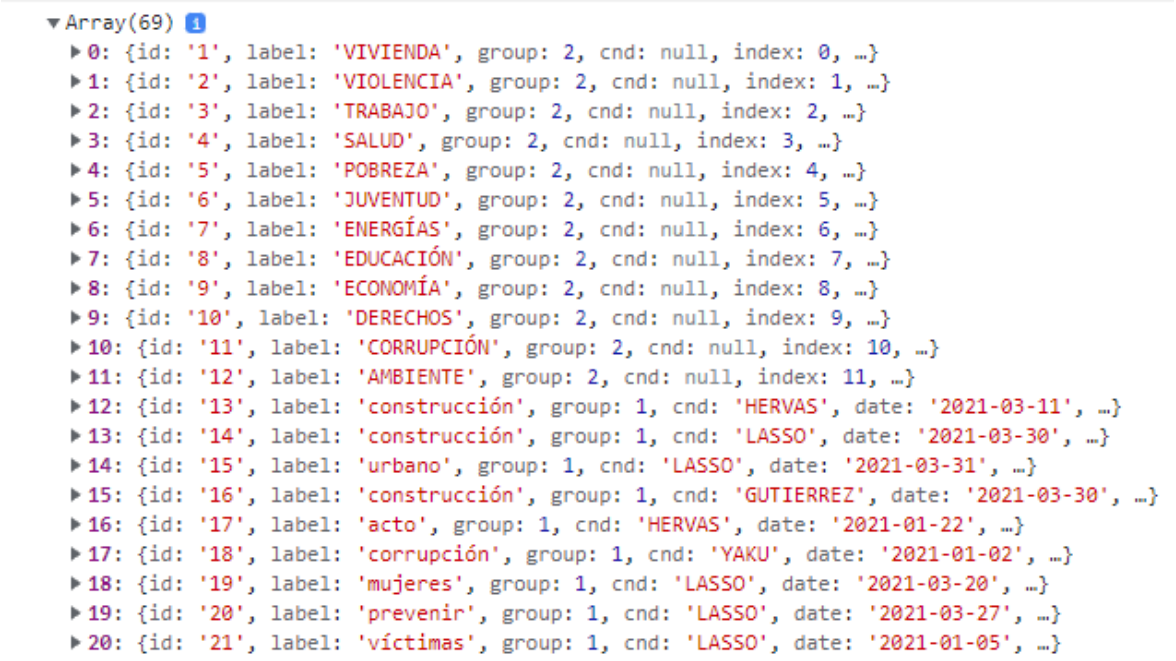

Figura 2.17. Extracto del arreglo de nodos a utilizarse en la segunda visualización

En la Figura 2.18 se muestra un extracto del arreglo de enlaces que se utilizará en la generación de la segunda visualización.

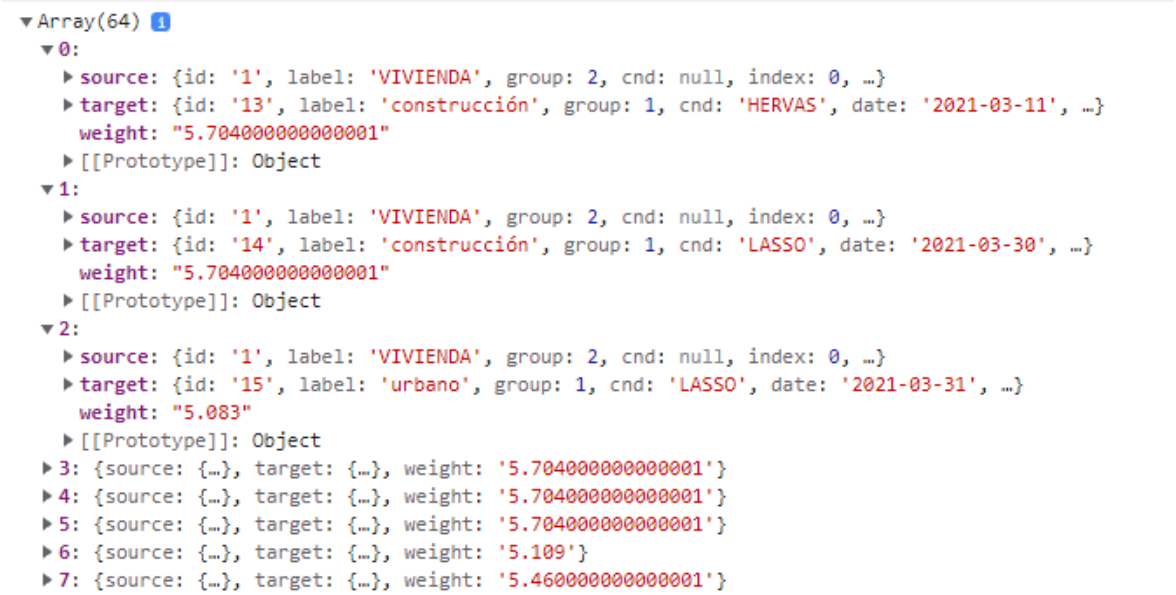

Figura 2.18. Extracto del arreglo de enlaces a utilizarse en la segunda visualización

2.3.5.2. CREACIÓN DE LA RED DE TENDENCIA

Existen diversas funciones que provee la biblioteca d3.js que permite realizar un sinfín de visualizaciones con distinta estructura y características. Para este caso en específico se

usará la función *force()* modificando sus propiedades para cumplir con el objetivo que tiene la segunda visualización.

En la siguiente tabla se muestran las propiedades que tienen la función *force()* y su impacto dentro la visualización a implementar.

Tabla 2.16. Propiedades de la función force()

PROPIEDAD	IMPACTO
force.size([*width, height*])	Define el tamaño de la visualización a implementar
force.linkDistance([*distance*])	Define la distancia entre nodos principales y nodos secundarios en base a la relación existente entre ellos (SIM).
force.charge([*charge*])	Define la atracción entre nodos.
force.gravity([*gravity*])	Define la atracción de los nodos al centro de la visualización
force.drag();	Habilita el movimiento de nodos mediante el uso del mouse.
force.start();	Inicializa la simulación asignando posiciones aleatorias a los nodos dentro del gráfico respetando aquellas propiedades que han sido definidas previamente.

La función *force.start()* es la encargada de inicializar la visualización, pero previamente necesita tener definida la información de la cual va a poder generarla. Dicha información se obtiene de los arreglos definidos anteriormente En el Código **2.14** se puede observar la estructura de la función previo asignación de la data necesaria.

```
1 force.nodes(nodes).links(edges).start();
```

Código 2.14. Inicialización de la visualización mediante la función force.start()

Una vez inicializada la visualización, la función force.start() añade atributos a cada objeto del arreglo de nodos aparte de los que ya se definió en su estructura.

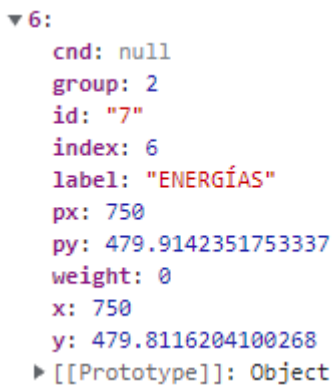

Figura 2.19. Extracto de un objeto del arreglo de nodos

Estos atributos contienen la ubicación en coordenadas rectangulares del nodo dentro del espacio de la visualización, que van variando conforme se interactúe con la misma. Dichas

coordenadas/posiciones se generan, a percepción del usuario, de manera aleatoria, pero que internamente la función va asignando posiciones a cada nodo hasta encontrar una estabilidad en el diagrama de fuerza (force). En la Figura 2.19 se puede observar los atributos de un objeto del arreglo de nodos una vez inicializada la visualización.

Otra de las funciones que toma más relevancia en esta visualización es *force.linkDistance()*. Esta función definirá la proximidad entre las palabras (nodos secundarios) y las categorías (nodos primarios) en base al índice de similitud (SIM) que tengan entre ellos. El

Código **2.15** muestra el fragmento de código que permite establecer una distancia específica por cada enlace en base a la relación que exista entre nodos (weight).

Código 2.15. Código de la función linkDistance() para la segunda visualización

```
1force.linkDistance(function(link) {
2    return 200-(parseFloat(link.weight)*20);
3});
```

El resto de las funciones detalladas en la Tabla 2.16, si bien no dejan de ser importantes, están enfocadas mayormente en el apartado estético de la visualización.

2.3.6. VISUALIZACIÓN COMPARATIVA DE TENDENCIA DEL DISCURSO POLÍTICO

En esta sección se detallará la implementación de la tercera visualización en base al diseño detallado en la sección 2.2.3.2.3. Al igual que la segunda visualización, el lenguaje usado para este caso será JavaScript haciendo uso de la biblioteca d3.js, de la cual se habló al principio del presente trabajo.

2.3.6.1. FORMATO DE LA INFORMACIÓN

Dado que el diseño de la visualización es similar a la segunda visualización implementada se mantiene la estructura de la información: un arreglo destinado a los nodos y un segundo arreglo destinado a los enlaces entre nodos. El primero de ellos se diferencia del arreglo de la segunda visualización ya que incorpora la fecha como nuevo campo por cada elemento del arreglo como se detalla en la

Tabla 2.17. Estructura de los arreglos

ARREGLO	ATRIBUTOS POR OBJETO (NODO) DEL ARREGLO	DESCRIPCIÓN
Nodos (nodes)	Identificador (id)	Identificador secuencial del objeto.
	Palabra (label)	Palabra asociada al nodo.
	Grupo (group)	Grupo al que pertenece el nodo. (Primario o secundario)
	Candidato (cnd)	Candidato al cual pertenece la palabra.
	Fecha (date)	*Fecha de publicación del tweet al que pertenece la palabra.*
Enlaces (edges)	Fuente (source)	Atributo tipo nodo. Especifica el nodo fuente del enlace.
	Destino (target)	Atributo tipo nodo. Especifica el nodo destino del enlace.
	SIM (weight)	Indica el peso del enlace. Se basa en el índice de similitud (SIM) que tenía el nodo secundario (palabra) con el nodo primario (categoría).

En Figura 2.20 la se muestra un extracto del arreglo de nodos utilizado en la implementación de la tercera visualización donde destaca la fecha como nuevo campo respecto a la estructura usada en la segunda visualización.


```
▼12:
   cnd: "HERVAS"
   date: "2021-03-11"
   group: 1
   id: "13"
   index: 12
   label: "construcción"
   px: 1049.7388965071755
   py: 838.2486345624496
   sim: "0.5704"
   weight: 1
   x: 1050.72
   y: 838.2986654154059
  ▶[[Prototype]]: Object
▶13: {id: '14', label: 'construcción', group: 1, cnd: 'LASSO', date: '2021-03-30', …}
▶14: {id: '15', label: 'urbano', group: 1, cnd: 'LASSO', date: '2021-03-31', …}
▶15: {id: '16', label: 'construcción', group: 1, cnd: 'GUTIERREZ', date: '2021-03-30', …}
▶16: {id: '17', label: 'acto', group: 1, cnd: 'HERVAS', date: '2021-01-22', …}
▶17: {id: '18', label: 'corrupción', group: 1, cnd: 'YAKU', date: '2021-01-02', …}
▶18: {id: '19', label: 'mujeres', group: 1, cnd: 'LASSO', date: '2021-03-20', …}
▶19: {id: '20', label: 'prevenir', group: 1, cnd: 'LASSO', date: '2021-03-27', …}
▶20: {id: '21', label: 'víctimas', group: 1, cnd: 'LASSO', date: '2021-01-05', …}
▶21: {id: '22', label: 'doméstica', group: 1, cnd: 'LASSO', date: '2021-01-14', …}
▶22: {id: '23', label: 'seguridad', group: 1, cnd: 'FREILE', date: '2021-01-01', …}
}
```


Figura 2.20. Extracto del arreglo de nodos a utilizarse en la tercera visualización

El arreglo de enlaces no tiene variación alguna respecto a la estructura utilizada en la segunda visualización que se presenta en la Figura 2.17.

2.3.6.2. FILTRO “FECHA DE COMPARACIÓN”

Además de los filtros definidos en la sección 2.2.3.2.2, esta visualización tiene un filtro adicional de tipo fecha, que altera la misma comparando la fecha definida en el filtro con el campo fecha que tiene cada uno de los elementos del arreglo de nodos. La generación del mismo se basa en el siguiente código.

Código 2.16. Implementación del filtro tipo fecha

Su funcionamiento respecto a los datos a utilizarse se evidencia en la ubicación que tendrán los nodos dentro de la visualización. Dependiendo la fecha elegida por el usuario cada nodo se ubicará de lado derecho de las categorías (fecha de nodo anterior a la fecha de filtro) o de lado izquierdo de las categorías (fecha de nodo posterior a la fecha de filtro) como se puede observar en la

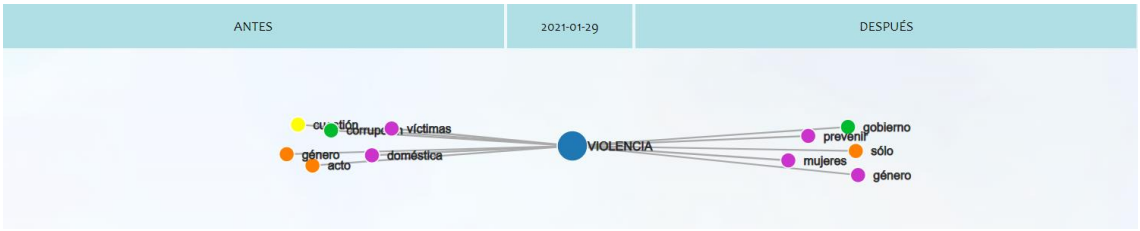

Figura 2.21. Funcionamiento del filtro fecha

2.3.6.3. CREACIÓN DE LA RED COMPARATIVA DE TENDENCIA

Al igual que la segunda visualización, al tratarse de una visualización tipo red, utiliza las mismas funciones definidas en la Tabla 2.16, con la diferencia que para este caso vamos a forzar la ubicación de los nodos alterando las posiciones que define automáticamente la función *force.start()*. Esto con el objetivo de enlazar el funcionamiento del filtro fecha explicado previamente y la distribución de los nodos dentro de la visualización.

```
<div id="cand" >
    <h3>Fecha de comparación</h3>
    <input      type="date"
                id="start"
                name="trip-start"
                value="2021-02-08"
                min="2020-12-30"
                max="2021-05-31"
                onchange="updateNetwork()">
</div>
```

Para personalizar la ubicación de los distintos nodos en base a la fecha definida en el filtro se utiliza el siguiente fragmento de código.

Código 2.17. Personalización de la ubicación de los nodos dentro de la visualización

Para el caso de los nodos que son de tipo categoría la ubicación respecto al eje X es la misma para todos mientras que para el eje Y será el diagrama de fuerza quien asigne el valor de este eje automáticamente. Para el caso de los nodos de tipo palabra las ubicaciones se verán definidas conforme a lo explicado en la sección 2.3.6.2.

La función *force.start()* se encarga de inicializar la visualización basándose en los arreglos de nodos y enlaces obtenidos de la data proporcionada como se muestra en el Código 2.18 y tomando en cuenta los filtros aplicables para la misma.

```
1 force.nodes(nodes).links(edges).start();
```

Código 2.18. Inicialización de la visualización mediante la función force.start()

2.3.7. INTERFAZ WEB

```
1 nodes.forEach(function(node){if(node.cnd == null){node.x = 750;}})
2
3 edges.forEach(function(edge){
4     edge.source.x = 750;
5     if(edge.target.date <= fechaInput){
6         edge.target.x = edge.target.sim*700+50;
7     }else if(edge.target.date > fechaInput){
8         edge.target.x = 1450-edge.target.sim*700;
9     }
10});
```

La interfaz web donde se alojarán las tres visualizaciones se basa en el lenguaje de etiquetas HTML, sencilla e intuitiva para el usuario final.

La interfaz constará de un sistema de tabs o pestañas, las cuales presentan una pantalla distinta al interactuar con cada una de ellas. Se tiene tres pestañas destinadas cada una a una visualización y una pestaña que presenta una reseña de las visualizaciones y su objetivo como se puede observar en la Figura 2.22.

Figura 2.22. Interfaz web basado en un sistema de tabs o pestañas

El desarrollo de esta interfaz se puede visualizar en el Código 2.19.

```
<ul class="tabs">
    <li><a href="#tab1"><span class="fa fa-home"></span><span class="tab-
    text">Inicio</span></a></li>
    <li><a href="#tab2"><span class="fa fa-group"></span><span class="tab-
    text">Tabla de tendencia</span></a></li>
    <li><a href="#tab3"><span class="fa fa-briefcase"></span><span
    class="tab-text">Red de tendencia</span></a></li>
    <li><a href="#tab4"><span class="fa fa-bookmark"></span><span
    class="tab-text">Comparativa de tendencia</span></a></li>
</ul>

<div class="secciones">
    <article id="tab1">
        <div class="about">
            <h1>RESEÑA</h1>
            <p>{texto}</p>
            <br>
            <p>{texto}</p>
            <br>
            <p>{texto}</p>
        </div>
    </article>
    <article id="tab2">
        <iframe src="./web/Tabla de tendencia del discurso
        político.html" width="100%" height="100%"></iframe>
    </article>
    <article id="tab3">
        <iframe src="./web/Red de tendencia del discurso político.html"
        width="100%" height="100%"></iframe>
    </article>
    <article id="tab4">
        <iframe src="./web/Comparativa de tendencia del discurso
        político.html" width="100%" height="100%"></iframe>
    </article>
</div>
```

Código 2.19. Desarrollo interfaz web basado en tags o pestañas

2.3.8. HOST

El host utilizado para publicar el aplicativo web en la nube es *GITHUB* mediante el recurso *GITHUB PAGES* el cual nos brinda un espacio en la nube o repositorio en el cual alojar las fuentes que generan el aplicativo web para posteriormente hacerla accesible desde el internet.

Además, brinda la posibilidad de adaptar un nombre de dominio personalizado para acceder a los recursos cargados en la nube, en este caso de realiza la compra del dominio www.ecuadordemocratico.com al cual se enlazó el aplicativo web. En la Figura 2.23 se puede observar la configuración que se le realiza al dominio obtenido para que este apunte al repositorio que contiene el código de la página web.

	Tipo	Nombre	Datos	TTL		
	A	@	185.199.108.153	600 segundos	Eliminar	Editar
	A	@	185.199.109.153	600 segundos	Eliminar	Editar
	A	@	185.199.110.153	600 segundos	Eliminar	Editar
	A	@	185.199.111.153	600 segundos	Eliminar	Editar
	NS	@	ns05.domaincontrol.com.	1 Hora	No se puede eliminar	No se puede editar
	NS	@	ns06.domaincontrol.com.	1 Hora	No se puede eliminar	No se puede editar
	CNAME	www	dhmontero.github.io.	1 Hora	Eliminar	Editar
	CNAME	_domainconnect	_domainconnect.gd.domaincontrol.com.	1 Hora	Eliminar	Editar
	SOA	@	Nombre del servidor principal: ns05.domaincontrol.com.	1 Hora	Eliminar	Editar

Figura 2.23. Configuración de domino personalizado.

El registro de tipo A obedece a las IP de los servidores públicos que brinda GITHUB para obtener sus recursos. El registro de tipo CNAME permite redirigir el acceso de un dominio hacia otro en este caso vamos a redirigir el acceso de dhmontero.github.io. que es el sitio donde está alojada nuestra página hacia el nombre de dominio personalizado que definimos anteriormente.

El resto de los registros son generados automáticamente por el servicio de dominios que en este caso es www.godaddy.com.

Para finalizar en el repositorio de git hub donde se encuentra la página en el apartado se settings/pages definimos el dominio personalizado en el cual se va a alojar el aplicativo web como se puede observar en la Figura 2.24. y finalmente se tendría publicado en el internet el aplicativo y a su vez se le ha asignado un nombre de dominio personalizado.

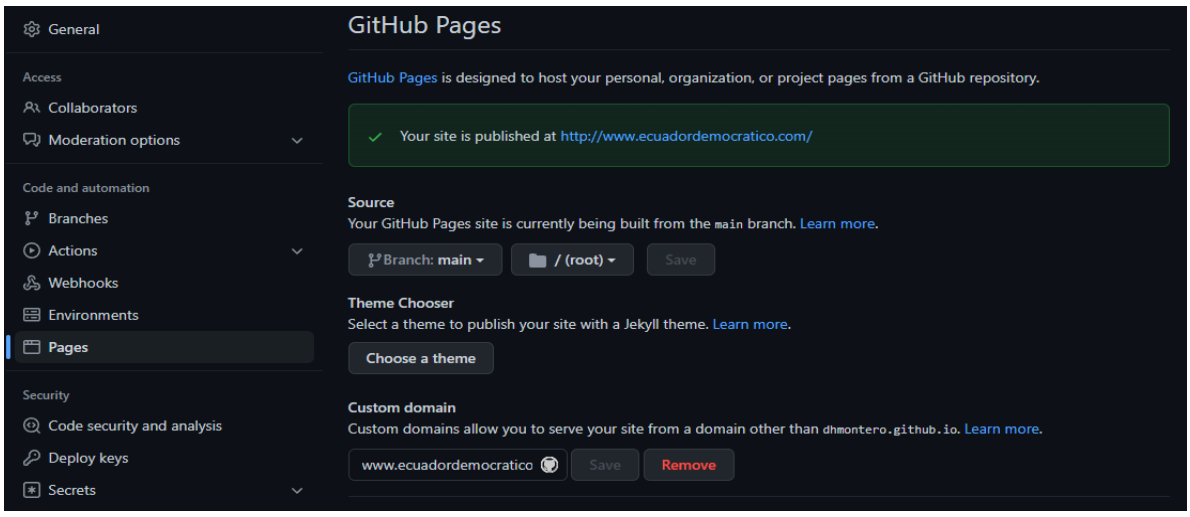

Figura 2.24. Configuración de dominio personalizado en GITHUB.

3. RESULTADOS Y DISCUSIÓN

En este capítulo se presentan las pruebas realizadas al aplicativo web y sus distintas funcionalidades basándose en los requerimientos funcionales y no funcionales definidos en el apartado 2.1.3

Se realizan encuestas de satisfacción aplicada a un grupo de usuarios determinado, el cual hará uso del aplicativo web y se expondrán los errores encontrados durante las pruebas de funcionamiento y la solución a los mismos.

3.1. ACTUALIZACIÓN DEL TABLERO KANBAN

Una vez finalizada la fase de implementación se tiene avanzado el proyecto notablemente, las tareas que respectan al análisis funcional y el desarrollo técnico del mismo se encuentran concluidas, únicamente quedan tareas de pruebas de validación de funcionalidad y pruebas de usuario por realizarse además de posibles correcciones que se deban hacer a partir de los resultados de dichas pruebas.

Como se evidencia en la Tabla 3.1 la tarea de validación de funcionalidad se encuentra en progreso debido a que en desarrollo de software mientras se va construyendo el prototipo se va probando constantemente los avances presentados, al final se hace pruebas de funcionalidad de punta a punta para corroborar el correcto desenvolvimiento del prototipo.

Tabla 3.1. Tablero de Kanban al inicio de la fase de resultados

POR HACER	EN PROGRESO	REALIZADAS
Realizar pruebas de cada funcionalidad del prototipo mediante pruebas de usuario.	Realizar pruebas de validación de funcionalidad.	Analizar las herramientas de desarrollo necesarias para la implementación del prototipo.
Corregir errores detectados.		Realizar encuesta a usuarios potenciales del prototipo.
		Analizar el archivo de datos proporcionado
		Elaborar el diagrama de casos de uso.
		Definir las visualizaciones acordes a requerimientos funcionales.
		Diseñar y el mapeo de datos a objetos visuales adecuado para las distintas visualizaciones definidas.

		Instalación de las distintas herramientas para la implementación del prototipo
		Codificar las distintas visualizaciones definidas
		Desarrollar la interfaz web donde se alojarán las gráficas codificadas.
		Seleccionar y configurar del host donde se alojará el aplicativo web.
		Seleccionar de dominio personalizado para la aplicación web.
		Publicar prototipo en la nube con el dominio personalizado.

3.2. PRUEBAS DE FUNCIONAMIENTO

En esta sección se presentará los resultados de las pruebas que validan el correcto funcionamiento de todas las características del aplicativo web. Las pruebas se realizarán directamente en la nube en el dominio que se ha definido para publicar el aplicativo web en Internet.

En las siguientes secciones se abordará distintos ítems que se van a validar durante las pruebas en las que se verán inmiscuidas las tres visualizaciones basándose en los requerimientos funcionales y no funcionales que se definieron en la sección 2.1.3.

3.2.1. FUNCIONAMIENTO INTERFAZ WEB

En la siguiente sección se valida el funcionamiento de la interfaz web en la cual se alojarán las tres visualizaciones que componen el aplicativo web. El acceso a esta interfaz web, así como su estructura solventan los requerimientos no funcionales RNF001 y RNF002 definidos para este aplicativo web.

El aplicativo web está disponible para todo usuario mediante el domino o URL https://www.ecuadordemocratico.com/ como se puede apreciar en la Figura 3.1.

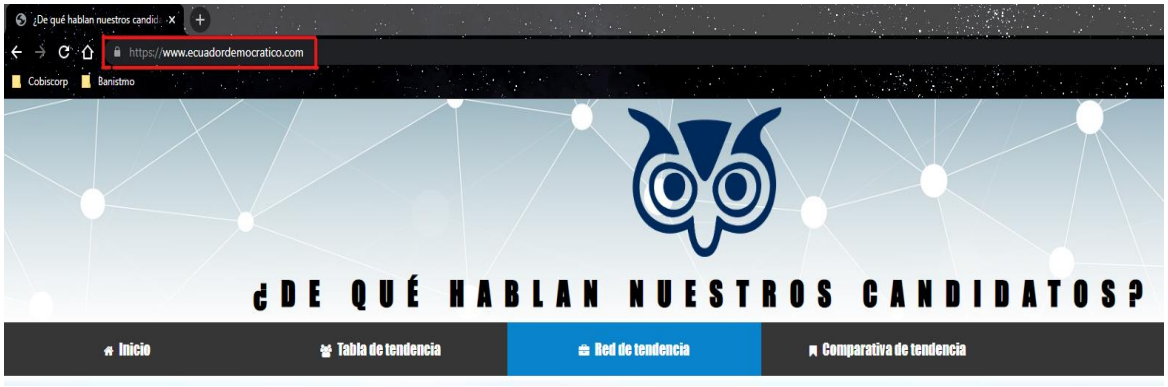

Figura 3.1. URL de acceso al aplicativo web mediante internet.

El aplicativo web presenta una estructura de pestañas o tabs las cuales a interacción del usuario con las mismas presentan una sección diferente: Inicio, Tabla de tendencia, Red de Tendencia y Comparativa de tendencia, pertenecientes a cada una de las visualizaciones definidas y una adicional que muestra una reseña del contexto político ecuatoriano y a su vez explica a breves rasgos la funcionalidad del aplicativo como se pueden apreciar en las siguientes figuras.

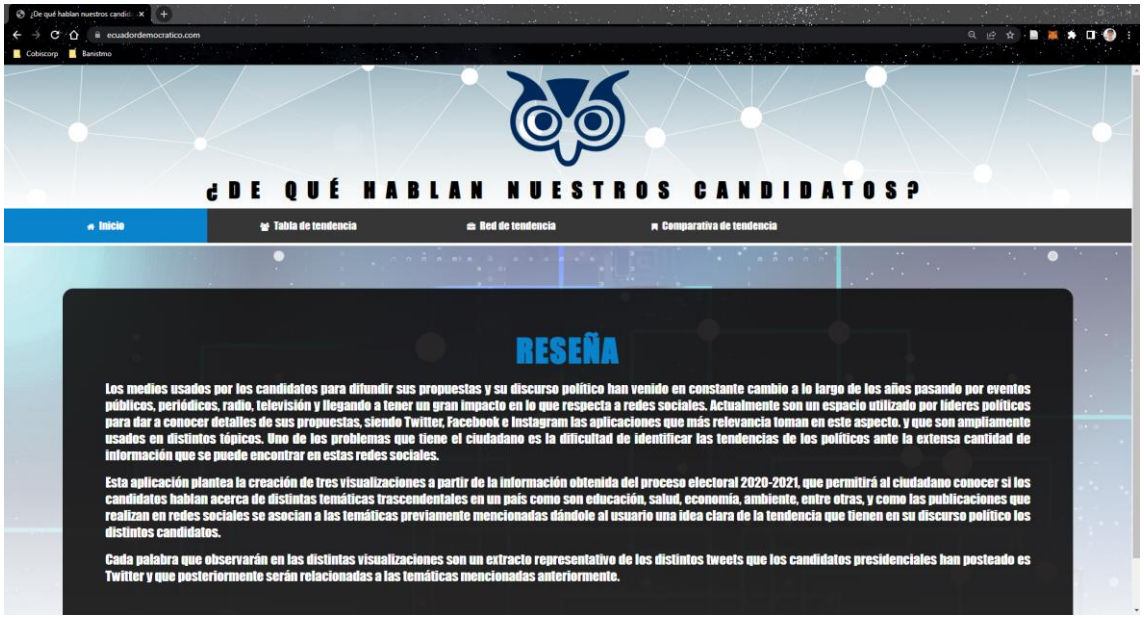

Figura 3.2. Pestaña de inicio del aplicativo web.

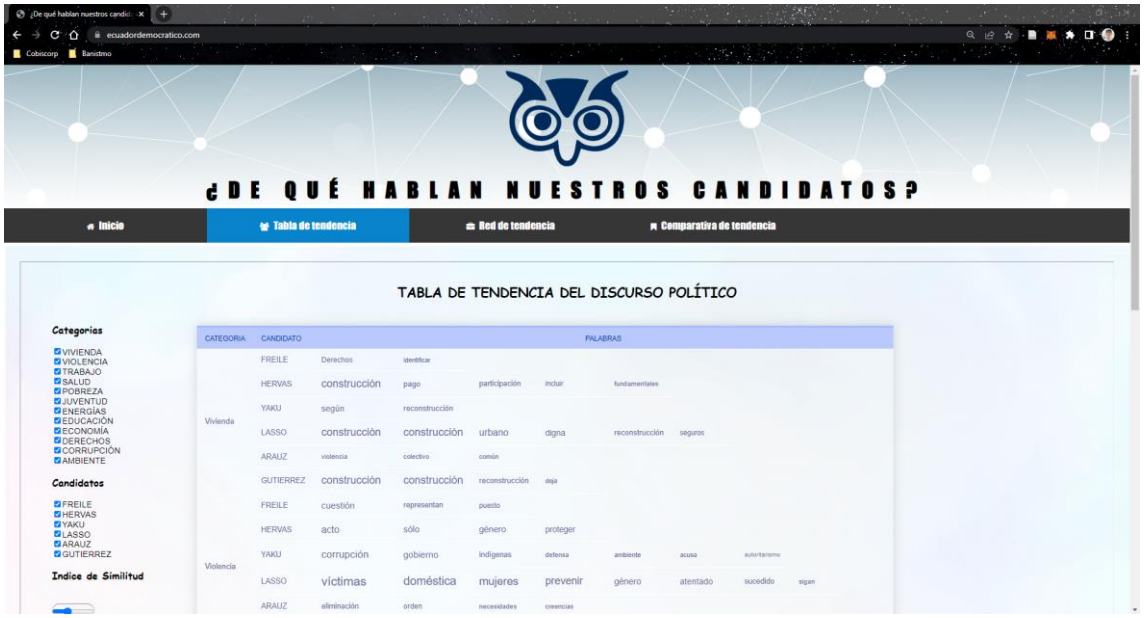

Figura 3.3. Pestaña perteneciente a la visualización “Tabla de tendencia del discurso político”.

Figura 3.4. Pestaña perteneciente a la visualización “Red de tendencia del discurso político”.

Figura 3.5. Pestaña perteneciente a la visualización “Comparativa de tendencia del discurso político”.

3.2.2. FUNCIONAMIENTO DE FILTROS

En la siguiente sección se valida el funcionamiento de los filtros: categoría candidato, índice de similitud y fecha de comparación, este último siendo exclusivo únicamente para la tercera visualización. Filtros que solventan los requerimientos funcionales RF001, RF002 y RF003 definidos para este aplicativo web.

Para la primera visualización “TABLA DE TENDENCIA DEL DISCURSO POLÍTICO” se ha escogido cuatro categorías, tres candidatos y un índice de similitud base de 0.33, en la Figura 3.6 se puede visualizar la correspondencia entre la información representada y los filtros definidos.

El índice de similitud se lo puede diferenciar mediante el tamaño de las palabras pertenecientes a cada categoría y candidato, las palabras con mayor similitud a la categoría correspondiente presentan un mayor tamaño y a medida que menos se relacionan con la categoría su tamaño va disminuyendo. Además, mientras menos se relacione la palabra, se ubicará más alejada de su categoría correspondiente en cuanto a columnas se refiere.

Figura 3.6. Validación de filtros en la primera visualización.

Para la segunda visualización “RED DE TENDENCIA DEL DISCURSO POLÍTICO” se ha definido cinco categorías, tres candidatos y un índice de similitud de 0.315, en la Figura 3.7 se puede verificar la correspondencia entre la información representada y los filtros definidos.

Las palabras, representadas por nodos o círculos pequeños, tienen como característica un color específico, los cuales representan a cada uno de los candidatos a los cuales pertenecen dichas palabras. Para este caso tenemos tres colores distintos los cuales obedecen a los tres candidatos escogidos mediante los filtros. Además, la distancia existente entre las palabras (nodos de borde) y las categorías (nodos principales) representan el índice de similitud existente entre ellas, mientras más corta sea la distancia que los separa, mayor será la relación que tenga dicha palabra con su correspondiente categoría.

La cantidad de palabras que concentre un candidato en una categoría específica, indica la cantidad de tweets en las que se hizo referencia implícitamente a dicha categoría dentro del discurso político del candidato, limitando este valor a las palabras que superen el valor de índice de similitud proporcionado por el filtro, en este caso 0.33.

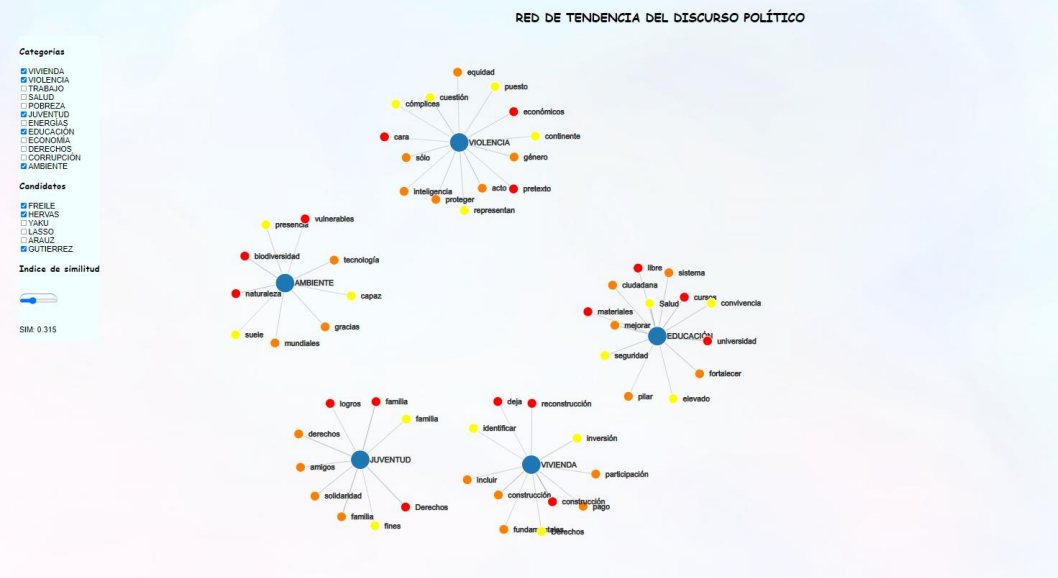

Figura 3.7. Validación de filtros en la segunda visualización.

Para la tercera visualización “COMPARATIVA DE TENDENCIA DEL DISCURSO POLÍTICO” se ha definido cinco categorías, cinco candidatos, índice de similitud de 0.5 y la fecha de comparación del 14 de enero de 2021, en la Figura 3.8 se puede verificar la correspondencia entre la información representada y los filtros definidos.

En este caso, los colores definidos para cada palabra, la cantidad de palabras por categoría para un candidato específico y la distancia entre palabra y categoría tienen la misma interpretación que visualización explicada previamente.

A esta visualización se le añade el filtro fecha, el cual nos permite diferenciar mediante la ubicación de la palabra si está pertenece a un tweet publicado antes o después de la fecha definida por el usuario, siendo a la izquierda del eje central (categorías) el tiempo anterior a dicha fecha y a la derecha el tiempo posterior a dicha fecha. Lo que le permite al usuario tener una comparación temporal del discurso político de los candidatos.

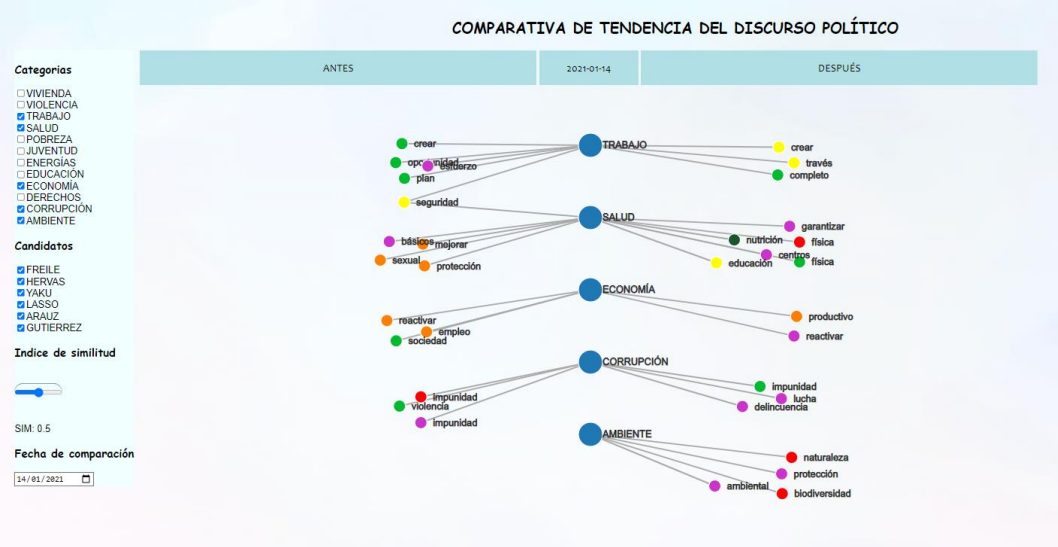

Figura 3.8. Validación de filtros en la tercera visualización.

3.3. PRUEBAS DE USUARIO

Una vez verificado el funcionamiento del aplicativo web, tanto a nivel funcional: segmentación y visualización de la información mediante el uso de filtros, así como a nivel no funcional: acceso al aplicativo web y la estructura del mismo. Se procede a realizar las pruebas a 20 usuarios finales, las mismas que se basan en la interacción con el aplicativo web y las conclusiones a las que pueda llegar el usuario en base a lo que pueda observar en las visualizaciones previamente definidas.

A continuación, se listan las actividades que realizaron los usuarios durante el proceso de pruebas.

Tabla 3.2. Actividades realizadas por los usuarios en el aplicativo web.

Actividad	Descripción
Acceso al aplicativo web	Los usuarios acceden al aplicativo web por medio de internet desde sus computadoras.
Navegación por el aplicativo web	Los usuarios navegan por las pestañas presentes en el aplicativo web familiarizándose con el mismo.
Interacción visualización “Tabla de tendencia del discurso político”	Los usuarios interactúan con la primera visualización generando diversas comparativas y obteniendo conclusiones propias respecto a la tendencia del discurso político de los candidatos.
Interacción visualización “Red de tendencia del discurso político”	Los usuarios interactúan con la segunda visualización generando diversas

	comparativas y obteniendo conclusiones propias respecto a la tendencia del discurso político de los candidatos.
Interacción visualización “Comparativa de tendencia del discurso político”	Los usuarios interactúan con la tercera visualización generando diversas comparativas y obteniendo conclusiones propias respecto a la tendencia del discurso político de los candidatos.
Conclusiones finales	Los usuarios obtienen una conclusión final acerca del funcionamiento y la utilidad del aplicativo web.

3.3.1. RESULTADOS

Para la obtención de los resultados de las pruebas de usuario realizadas y la satisfacción de los mismo se generó un cuestionario, los resultados del mismo se encuentran detallados en la tabla 3.3. en la cual se puede verificar que todas las respuestas son positivas y no se han registrado errores en la interacción de los usuarios con el aplicativo web, salvo ciertas recomendaciones acerca de las características visuales del prototipo.

Tabla 3.3. Resultados de la encuesta de satisfacción realizada a los usuarios.

#	Pregunta	Respuestas		
		SI	**NO**	**OTRAS**
1	¿Pudo acceder con facilidad al aplicativo web?	100%	0%	No aplica
2	¿La interacción con el aplicativo web fue sencilla?	100%	0%	No aplica
3	¿Modificaría usted el aspecto visual del aplicativo web?	40%	60%	Un usuario aumentaría un poco más de color al aplicativo además de aumentar el tamaño de los textos presentados.
4	¿Se tuvo claro la fuente de donde se extrae la información?	100%	0%	No aplica
5	¿Se tuvo claro el funcionamiento de cada uno de los filtros aplicado en las distintas visualizaciones?	100%	0%	No aplica

6	¿En la primera visualización, pudo realizar distintas comparaciones que le llevaron a una conclusión objetiva en base a la información mostrada?	100%	0%	No aplica
7	En la segunda visualización, ¿Pudo realizar distintas comparaciones que le llevaron a una conclusión objetiva en base a la información mostrada?	100%	0%	No aplica
8	En la tercera visualización, ¿Pudo realizar distintas comparaciones que le llevaron a una conclusión objetiva en base a la información mostrada?	100%	0%	No aplica
9	Al interactuar con las visualizaciones, ¿Pudo establecer con claridad la tendencia del discurso político de los candidatos?	100%	0%	No aplica
10	¿Encontró algún error durante la interacción con el aplicativo web?	0%	100%	No aplica
11	¿Considera satisfactoria su interacción con el aplicativo web?	100%	0%	No aplica
12	Comentarios y sugerencias	N/A	N/A	Los usuarios manifiestan que el aplicativo se muestra eficaz e idóneo para procesar bastante información que en su estado natural tomaría tiempo y generaría confusión.

En base a los resultados de la encuesta se procedió a obedecer a las sugerencias proporcionadas por los usuarios y se aumentó el color de la interfaz web y del prototipo además de aumentar el tamaño de los textos presentados para que estos sean más legibles.

3.4. ACTUALIZACIÓN FINAL DEL TABLERO KANBAN

Una vez finalizadas las pruebas pertinentes y la corrección de los errores presentados en las mismas, se concluye el proyecto y mediante el tablero Kanban se puede evidenciar la finalización del mismo al tener todas las tareas en estado de realizadas, lo que indica que no hay ninguna tarea pendiente por hacer.

Tabla 3.4. Tablero de Kanban al final de la fase de resultados

POR HACER	EN PROGRESO	REALIZADAS
		Analizar las herramientas de desarrollo necesarias para la implementación del prototipo.
		Realizar encuesta a usuarios potenciales del prototipo.
		Analizar el archivo de datos proporcionado
		Elaborar el diagrama de casos de uso.
		Definir las visualizaciones acordes a requerimientos funcionales.
		Diseñar y el mapeo de datos a objetos visuales adecuado para las distintas visualizaciones definidas.
		Instalación de las distintas herramientas para la implementación del prototipo
		Codificar las distintas visualizaciones definidas
		Desarrollar la interfaz web donde se alojarán las gráficas codificadas.
		Seleccionar y configurar del host donde se alojará el aplicativo web.
		Seleccionar de dominio personalizado para la aplicación web.
		Publicar prototipo en la nube con el dominio personalizado.
		Realizar pruebas de validación de funcionalidad.
		Realizar pruebas de cada funcionalidad del prototipo mediante pruebas de usuario.
		Corregir errores detectados.

4. CONCLUSIONES Y RECOMENDACIONES

4.1. CONCLUSIONES

Como resultado final se tiene un prototipo de aplicación web, que permite conocer y comparar la tendencia del discurso político de los candidatos presidenciales ecuatorianos. Para la implementación de este prototipo se adquirieron nuevos conocimientos acerca de javascript, css y html. A continuación, se presentan las conclusiones tras la culminación de este proyecto de titulación.

- Para conocer la cantidad y calidad de la información que la ciudadanía tenía respecto a la tendencia del discurso político de los candidatos presidenciales ecuatorianos, se realizó una encuesta a 20 personas escogidas aleatoriamente. En base a los resultados de la encuesta se evidenció la falta de conocimiento del ciudadano respecto al tema planteado, concluyendo que la población no tiene una herramienta en la cual compactar toda la información que se presenta a través de los distintos medios y obtener una comparación clara entre candidatos en lo que respecta a su discurso político.

- A partir de la problemática definida se estableció los requerimientos funcionales y no funcionales del prototipo. En base a los requerimientos planteados se necesitó tres visualizaciones como mecanismo de representación de la información obtenida del discurso político expresado en las cuentas de Twitter de los distintos candidatos presidenciales y mediante las mismas solventar cada uno de los requerimientos establecidos.

- La visualización “Tabla de tendencia del discurso político”, como su nombre lo indica es de tipo tabla de datos y acepta las operaciones filter y detail-on-demand en base a los datos que puede representar. La ventaja de esta visualización respecto a las otras es que permitió a los usuarios conocer que candidato mediante su discurso político se enfocó más a una categoría en específico

- La visualización “Red de tendencia del discurso político” de tipo red de datos, basado en un diseño dirigido por la fuerza permite las operaciones overview, filter y detail-on-demand en base a los datos que puede representar. La ventaja de esta visualización respecto a las otras es que permitió a los usuarios diferenciar que candidato en volumen abordó más una categoría en específico.

- La visualización “Comparativa de tendencia del discurso político” de tipo red de datos, basado en un diseño dirigido por la fuerza permite las operaciones overview,

filter y detail-on-demand en base a los datos que puede representar. La ventaja de esta visualización respecto a las otras es que permitió al usuario diferenciar temporalmente que candidatos hablaron más o menos sobre una categoría en específico y cuanto cambió el enfoque del discurso político en relación a una fecha en particular.

- Al ser un prototipo basado en el desarrollo de software, se definió la metodología Kanban como herramienta de gestión de proyecto lo que permitió tener una organización efectiva respecto a las tareas necesarias para el desarrollo de este trabajo de titulación, siendo esta una metodología visual permitió llevar un control preciso respecto al flujo de trabajo.

- Se eligió JavaScript como lenguaje base y su librería d3.js como recurso para la implementación de las tres visualizaciones. A diferencia de otras herramientas como lo es Raphaël o Processing.js, D3.js tiene la función force() la cual está optimizada para diagramas dirigidos por la fuerza, los cuales se ocupan para la segunda y tercera visualización. Mediante la modificación de los parámetros de la función force() se pudo configurar el tamaño de la visualización, la atracción entre nodos, la atracción de los nodos al centro de la visualización, entre otras características además de tener funciones de lectura de datos propias a diferencia de las otras herramientas en las cuales es necesario implementar nativamente funciones que permitan el tratamiento de los datos.

- Para el desarrollo de la interfaz web del prototipo donde se alojarán las tres visualizaciones se utilizó HTML como lenguaje de marcaje en conjunto con CSS para brindarle estilo a la misma. Finalmente, para el alojamiento del prototipo en la nube se usó los servicios que proporciona Git Hub Pages permitiendo una fácil gestión de las fuentes y evitando la creación de un servidor web.

- Se realizó pruebas de funcionamiento del mismo, para lo cual se definió distintas combinaciones posibles de los filtros a los que obedecen cada una de las visualizaciones, corroborando que la información representada visualmente sea la esperada además de validar el correcto funcionamiento de la interfaz web del prototipo. Posteriormente se realizó pruebas de usuario en las cuales se eligió un grupo de diez personas a las cuales se les permitió el libre uso del prototipo. Luego de un tiempo prudencial se les pidió su opinión acerca del uso del prototipo en cuanto a la interacción con el mismo, del cual surgieron ciertas recomendaciones en cuanto a las características del prototipo, enfocándose principalmente en el color

de los elementos visuales y el tamaño de los textos presentados, dichas sugerencias fueron aceptadas y el prototipo fue readecuado de acuerdo a las necesidades de los usuarios. Además, se les consultó acerca de las conclusiones obtenidas a partir de sus pruebas, de los cuales la totalidad de las personas supieron hacer una comparación entre distintos candidatos, conocer acerca de las tendencias que tiene cada uno de ellos y el tópico social al cual se acerca más cada uno.

4.2. RECOMENDACIONES

El prototipo de aplicación web para la comparación de la tendencia del discurso político de los candidatos presidenciales ecuatorianos cumple con los requerimientos funcionales y no funcionales planteados, sin embargo, para este prototipo existen posibles implementaciones futuras que brindarían una mayor versatilidad al mismo, las cuales se presentan a continuación:

- Actualmente el prototipo se rige únicamente a un archivo perteneciente al proceso electoral 2021 el cual proporciona la información que se representa en las tres visualizaciones. Para un mejor uso del prototipo se recomienda realizar una conexión a una base de datos en la cual exista información de distintos procesos electorales y el prototipo permita también comparar información entre procesos electorales lo que brindaría al usuario la posibilidad de realizar otros tipos de análisis a su conveniencia.
- El prototipo presentado está únicamente desarrollado a nivel de capa de aplicación o frontend, se recomienda hacer uso nuevas y versátiles tecnologías para la obtención de los datos a representar como son servicios Api Rest los cuales brindarían al prototipo un amplio rango de posibilidades respecto a los datos que se pueden mostrar en la visualización.
- Para la interfaz web se recomienda acoplar otras librerías o frameworks como lo son AngularJS o ReactJs, que permitirán realizar interfaces de usuario mucho más personalizadas a beneficio de los usuarios, además de brindar la posibilidad de tener un prototipo web responsivo y adaptable a cualquier dispositivo digital.
- Se recomienda para futuras implementaciones extender el número de visualizaciones que puede presentar el prototipo, dándole al usuario la posibilidad de hacer comparaciones más profundas no solo del discurso político, por ejemplo,

del historial político de los candidatos, eventos transcendentales en los que se han visto involucrados, y muchas otras posibilidades.

Para mejorar el desarrollo del prototipo u otros que se basen en los mismos recursos y tecnologías se presentan las siguientes recomendaciones técnicas:

- Se recomienda tener en cuenta las diferencia entre las versiones de las librerías y lenguajes que se están usando en el desarrollo del prototipo debido a que entre versiones pueden cambiar tanto las funciones como la metodología de implementación de las mismas.
- Se recomienda, en la etapa de esquematización de las visualizaciones tomar en cuenta como factor crítico la cantidad de información que se puede llegar a tener, debido a que al no dimensionar la cantidad de datos se puede escoger visualizaciones las cuales se vean saturadas de información y genere conflicto en el usuario al momento de intentar sacar conclusiones de la misma.
- Se recomienda usar funciones propias de JavaScript para el manejo de colecciones de datos o arrays, como es el caso de la función reduce() que simplifica la iteración de los datos y evita el uso de bucles explícitos dentro del código que generan lentitud en el procesamiento de la información salvo casos especiales.
- Se recomienda aplicar el concepto de código limpio y óptimo como buena práctica al momento de programar, derivando código a funciones independientes que pueden ser llamadas de acuerdo a las necesidades del desarrollador evitando así duplicidad de código.

5. REFERENCIAS BIBLIOGRÁFICAS

[1] D. Herrera, «¿De qué hablan nuestros candidatos?» [En línea]. Disponible en: https://docs.google.com/forms/d/1IaA_Kjsgn0znw6STNj0iIyHvSlzWoY3MjEeeIwOGFzc/prefill

[2] R. L. Ackoff, «From data to wisdom», *Journal of applied systems analysis*, vol. 16, n.º 1, pp. 3-9, 1989.

[3] C. Ware, «Foundation for a science of data visualization», *Information visualization: perception for design (2nd ed.). San Francisco: Morgan Kaufmann Publishers*, 2004.

[4] B. Shneiderman, «The eyes have it: A task by data type taxonomy for information visualizations», en *The craft of information visualization*, Elsevier, 2003, pp. 364-371.

[5] «Los 3 pilares del mapeo de datos en visualizaciones*», *Data IQ*, 14 de diciembre de 2017. https://dataiq.com.ar/blog/los-3-pilares-del-mapeo-de-datos-en-visualizaciones/ (accedido 13 de junio de 2022).

[6] W. S. Cleveland y R. McGill, «Graphical perception: Theory, experimentation, and application to the development of graphical methods», *Journal of the American statistical association*, vol. 79, n.º 387, pp. 531-554, 1984.

[7] J. Heer, M. Bostock, y V. Ogievetsky, «A tour through the visualization zoo», *Communications of the ACM*, vol. 53, n.º 6, pp. 59-67, 2010.

[8] R. Camana-Fiallos, «Visualización de Datos Electorales Presidenciales: Herramientas para la Visualización de Grandes Conjuntos de Datos», *CienciAmérica*, vol. 2, n.º 1, pp. 37-45, 2013.

[9] «Arauz vs Lasso: el duelo entre el correísmo y la derecha por la presidencia de Ecuador», *BBC News Mundo*. Accedido: 13 de junio de 2022. [En línea]. Disponible en: https://www.bbc.com/mundo/noticias-america-latina-56706606

[10] E. Cornejo Velázquez y E. Hernández Gutiérrez, «Uso de tableros virtuales Kanban como herramienta para mejorar productividad en equipos de trabajo», *Staobil lekilal ta lekil abtel*, n.º 9, pp. 79-95, 2015.

[11] B. Shneiderman, «The eyes have it: A task by data type taxonomy for information visualizations», en *The craft of information visualization*, Elsevier, 2003, pp. 364-371.

[12] anandmeg, «Install Visual Studio». https://docs.microsoft.com/en-us/visualstudio/install/install-visual-studio (accedido 13 de junio de 2022).

[13] «CSV - D3 wiki». https://d3-wiki.readthedocs.io/zh_CN/master/CSV/ (accedido 13 de junio de 2022).

[14] «JavaScript Array reduce() Method». https://www.w3schools.com/jsref/jsref_reduce.asp (accedido 13 de junio de 2022).

6. ANEXOS

A continuación, se presenta el código completo implementado para este prototipo, organizado de la siguiente manera:

ANEXO A. Código de implementación de la primera visualización “Tabla de tendencia del discurso político”

ANEXO B. Código de implementación de la segunda visualización “Red de tendencia del discurso político”

ANEXO C. Código de implementación de la tercera visualización “Comparativa de tendencia del discurso político”

ANEXO D. Código de implementación de interfaz web

ANEXO A

CÓDIGO DE IMPLEMENTACIÓN DE LA PRIMERA VISUALIZACIÓN “TABLA DE TENDENCIA DEL DISCURSO POLÍTICO”

```
<html>
   <head>
       <meta charset="utf-8">
       <script type="text/javascript"
src="https://cdnjs.cloudflare.com/ajax/libs/d3/5.16.0/d3.min.js"></script>
       <script type="text/javascript"
src="https://cdnjs.cloudflare.com/ajax/libs/c3/0.7.18/c3.min.js"></script>
       <script src="https://cdn.jsdelivr.net/npm/lodash@4.17.11/lodash.min.js"></script>
       <script src="https://unpkg.com/d3-simple-slider"></script>
       <link href="https://cdnjs.cloudflare.com/ajax/libs/c3/0.7.18/c3.css"
rel="stylesheet" type="text/css">
       <link rel="stylesheet" href="tabla.css">

      <style>
          body, html{
              font-family: Arial, Helvetica, sans-serif;
              overflow:hidden;
          }
          h1 {
            background-color: black;
            color: white;
            padding: 5px;
            text-align-last: center
            }
          table {     font-family: Arial, Helvetica, sans-serif; font-size:
12px;    margin: 45px;     width: 480px; text-align: left;    border-collapse: collapse;
}
          th {     font-size: 13px;     font-weight: normal;     padding:
8px;     background: #b9c9fe; border-top: 4px solid #aabcfe;    border-bottom: 1px solid
#fff; color: #039; }
          td {    padding: 8px;  border-bottom: 1px solid #fff; color: #669;    border-
top: 1px solid transparent; }
          tr:hover td { background: #d0dafd; color: #339; }
      </style>
   </head>

   <body>
        <h2>TABLA DE TENDENCIA DEL DISCURSO POLÍTICO</h2>
                <aside>
                    <div id="left" style = "float: left">
                        <div id="cat" ><h3>Categorias</h3></div>
                        <div id="cand" ><h3>Candidatos</h3></div>
                        <div class="slidecontainer">
```

```
                        <h3>Indice de Similitud</h3>
                        <input type="range" min="0" max="1" step="0.005"
value="0.33" class="slider" id="myRange" onclick="actTabla()">
                        <p>SIM: <span id="demo"></span></p>
                    </div>
                </div>
            </aside>
            <div>
               <div id="tabla_ten" style="float: left" class="tabla_ten">
                   <table class="styled-table">
                        <thead>
                            <tr class="active-row">
                                <th>CATEGORIA</th>
                                <th>CANDIDATO</th>
                                <th colspan="10" style="text-
align:center">PALABRAS</th>
                            </tr>
                        </thead>
                        <tbody id="cuerpoTabla">
                        </tbody>
                    </table>
                </div>
            </div>


    <script>

        let dataPalabras=[];
        let store=[];
        let candidatos = [];
        let categorias = [];
        let actCheck = 0;
        let lenghtCND = 0;

        var slider = document.getElementById("myRange");
        var output = document.getElementById("demo");
        output.innerHTML = slider.value;
        slider.oninput = function() {
            output.innerHTML = this.value;
        }

        function loadData(){
            return Promise.all([
                d3.csv("elecciones_final.csv")
            ]).then(datasets => {
                store.dat = datasets[0];
                return store;
            })
```

```
}

function dataRepetidaCand(elementos, nombre) {
    return elementos.filter(
        function(elementos){ return elementos.NomCand == nombre }
    );
}

function dataRepetidaCat(elementos, nombre) {
    return elementos.filter(
        function(elementos){ return elementos.NomCat == nombre }
    );
}

function obtCat(datos){
    var aux=0;
    var result = datos.reduce((result,d) => {
        result = Object.keys(result).map(key => result[key])
        if(result != undefined){
           var checkRep = dataRepetidaCat(result, d.categoria);
            if(checkRep.length == 0){
                var temp = {"CatID": aux + 1, "NomCat": d.categoria};
                result.push(temp);
                aux = aux + 1;
            }
        }
        return result;
    },{});
    categorias = result;
    return result;
}

function obtCand(datos){
    var aux=0;
    var result = datos.reduce((result,d) => {
        result = Object.keys(result).map(key => result[key])
        if(result != undefined){
           var checkRep = dataRepetidaCand(result, d.candidato);
            if(checkRep.length == 0){
                var temp = {"IdCand": aux + 1, "NomCand": d.candidato};
                result.push(temp);
                aux = aux + 1;
            }
        }
        return result;
    },{});
    result = Object.keys(result).map(key => result[key])
    candidatos = result;
```

```
    return result;
}

function ordenar(palabras,sim,orden){
    var newOrden = [];
    orden.forEach(function(el){
        var index = sim.indexOf(el);
        newOrden.push(palabras[index]);
    })
    return newOrden;
}

function getPalabras(){
    let dataTotal = store.dat;
    let palabras = [];
    let candidatos = obtCand(dataTotal)
    let categorias = obtCat(dataTotal)
    if (actCheck == 0){
        checkBCategorias(categorias)
        checkBCandidatos(candidatos)
        actCheck = 1;
    }
    categorias.forEach(ctg =>{
    let palCand = [];
          candidatos.forEach(cndd =>{
              let result5 = [];
              let result6 = [];
              let result7 = [];
              let aux = 0;
              let result2 = dataTotal.reduce((result,d) => {
                  if(d.categoria == ctg.NomCat && d.candidato == cndd.NomCand){
                      if(d.sim > slider.value){
                        let temp = d.word
                        let temp2 = d.sim
                        result5[aux] = temp;
                        result6[aux] = temp2;
                        result7[aux] = temp2;
                        aux = aux + 1 ;
                        }
                    }
                  return result5;
              });
                aux=0;
              result2 = Object.keys(result2).map(key => result2[key])
              result6 = Object.keys(result6).map(key => result6[key])
              result7 = result7.sort(function(a, b){return a - b});
              result2 = ordenar(result2,result6,result7);
```

```
                    let temp1 = {"Categoria": ctg.NomCat , "Candidato": cndd.NomCand,
"Palabras": result2, "Sim": result6, "Orden": result7 };
                    palabras.push(temp1);
                })
            })
          dataPalabras = palabras;
          dibujarTabla(palabras);
          isCheck();
          return palabras;
        }

        function dibujarTabla(dataPalabras){
            var numCND = 0;
            numCND = candidatos.length - lenghtCND;

            const $cuerpoTabla = document.querySelector("#cuerpoTabla");
            dataPalabras.forEach(pal => {
                const $tr = document.createElement("tr");
                  let $tdCategoria = document.createElement("td");
                  if(dataPalabras.indexOf(pal) % numCND == 0){
                  $tdCategoria.textContent = pal.Categoria;
                  $tdCategoria.rowSpan = numCND;
                  $tr.appendChild($tdCategoria);
                  }
                  let $tdCand = document.createElement("td");
                  $tdCand.textContent = pal.Candidato;
                  $tr.appendChild($tdCand);
                  var aux = pal.Sim.length;
                  for (var i=pal.Palabras.length-1;i>=0;i--){
                          let $tdPalabras = document.createElement("td");
                          var tam = parseFloat(pal.Orden[aux-1])*35
                          $tdPalabras.style.fontSize = tam;
                          $tdPalabras.textContent = pal.Palabras[i];
                          $tdPalabras.title = "Sim: " +pal.Orden[aux-1]
                          aux--;
                          $tr.appendChild($tdPalabras);
                  }
                  $cuerpoTabla.appendChild($tr);

            })
        }

         function actTabla(){
             var Table = document.getElementById("cuerpoTabla");
             Table.innerHTML = "";
             getPalabras();
         }
```

```
            function checkBCategorias(cat){
              var aux = 0
              cat.forEach(elemento => {
                var node = document.createElement('div');
                node.innerHTML = '<input type="checkbox" id="check'+aux+'" checked=""
onclick="isCheck()"><label
for="check'+aux+'">'+elemento.NomCat.toUpperCase()+'</label><br>';
                document.getElementById('cat').appendChild(node);
                aux=aux+1;
              })
          }

           function checkBCandidatos(cand){
              var aux = 0
              cand.forEach(elemento => {
                var node = document.createElement('div');
                node.innerHTML = '<input type="checkbox" id="checkC'+aux+'" checked=""
onclick="isCheck()"><label
for="checkC'+aux+'">'+elemento.NomCand.toUpperCase()+'</label><br>';
                document.getElementById('cand').appendChild(node);
                aux=aux+1;
              })
          }
           function isCheck(){
              var aux;
              var aux1=0;
              var aux2;
              var aux3=0;
              var indexCat = [];
              var indexCand = [];
              for(aux = 0;aux < categorias.length;aux++){
                 var isChecked = document.getElementById("check"+aux+"").checked;
                    if(isChecked != true){
                        indexCat[aux1] = categorias[aux].NomCat;
                        aux1=aux1+1;
                    }
                }
              for(aux2 = 0;aux2 < candidatos.length;aux2++){
                 var isChecked2 = document.getElementById("checkC"+aux2+"").checked;
                    if(isChecked2 != true){
                        indexCand[aux3] = candidatos[aux2].NomCand;
                        aux3=aux3+1;
                    }
                }
               hide(indexCat,indexCand);
          }
           function hide(indicesCategorias,indicesCandidatos) {
```

```
            var newCategorias=_.cloneDeep(categorias)
            var newCandidatos=_.cloneDeep(candidatos)
            var newPalabras=_.cloneDeep(dataPalabras);

            lenghtCND = indicesCandidatos.length;

           indicesCandidatos.forEach(function(el){
                 newPalabras.forEach(elemento => {
                     if(el == elemento.Candidato){
                         var temp = newPalabras.indexOf(elemento);
                         newPalabras.splice(temp,1);
                     }
                 })
             });
           indicesCategorias.forEach(el => {
                 newPalabras.forEach(elemento => {
                     if(el == elemento.Categoria){
                         var temp = newPalabras.indexOf(elemento);
                         newPalabras.splice(temp,candidatos.length);
                     }
                 })
             })
            var Table = document.getElementById("cuerpoTabla");
            Table.innerHTML = "";
            if(lenghtCND == candidatos.length){
               for (let i = newPalabras.length; i > 0; i--) {
                   newPalabras.pop();
               }
            }
            dibujarTabla(newPalabras);
         }
      loadData().then(getPalabras)
    </script>
  </body>
</html>
```

ANEXO B

CÓDIGO DE IMPLEMENTACIÓN DE LA PRIMERA VISUALIZACIÓN “RED DE TENDENCIA DEL DISCURSO POLÍTICO”

```
<!DOCTYPE html>
<meta charset="utf-8" />
<head>
    <link rel="stylesheet" href="tabla.css">
    <style>
      body {
          font-family:Arial, Helvetica, sans-serif;
          overflow: hidden
      }
      .link {
        stroke: #aaa;
      }
      .node text {
        stroke: #333;
        cursor: pointer;
      }
      .node circle {
        stroke: #fff;
        stroke-width: 1.5px;
        fill: #555;
      }
      .d3-tip {
        line-height: 1;
        font-weight: bold;
        padding: 14px;
        background: rgba(0, 0, 0, 0.8);
        color: #fff;
        border-radius: 2px;
      }
      .d3-tip:after {
        box-sizing: border-box;
        display: inline;
        font-size: 14px;
        width: 100%;
        line-height: 1;
        color: rgba(0, 0, 0, 0.8);
        content: "\25BC";
        position: absolute;
        text-align: center;
      }
      .d3-tip.n:after {
        margin: -1px 0 0 0;
```

```
        top: 100%;
        left: 0;
      }
    </style>
</head>
<body>
  <h2>RED DE TENDENCIA DEL DISCURSO POLÍTICO</h2>
  <aside  style = "float: left">
      <div>
        <div id="cat"><h3>Categorias</h3></div>
        <div id="cand"><h3>Candidatos</h3></div>
      </div>
      <div class="slidecontainer">
          <h3>Indice de similitud</h3>
          <input
            type="range"
            min="0"
            max="1"
            step="0.005"
            value="0.5"
            class="slider"
            id="myRange"
            onclick="updateNetwork()"
          />
          <p>SIM: <span id="demo"></span></p>
    </div>
  </aside>
  <div id="main3" class="main3" style = "float: left">
    <script type="text/javascript" src="https://d3js.org/d3.v3.min.js"></script>
    <script src="https://labratrevenge.com/d3-
tip/javascripts/d3.tip.v0.6.3.js"></script>
    <script>
      var store = [];
      var indexCat = [];
      var indexCand = [];
      var isNew = 0;
      var linkpath = "links.csv"; //archivo de enlaces
      var nodepath = "nodes.csv"; //archivo de nodos
      var width = 1500, //seteo de tamaño del grafo
        height = 1200;
      var color = d3.scale.category20(); //Gama de colores seteados por d3.js
      var slider = document.getElementById("myRange");
      var output = document.getElementById("demo");
      output.innerHTML = slider.value;
      slider.oninput = function () {
        output.innerHTML = this.value;
      };
      d3.csv("elecciones_final.csv", function (error, data) {
```

```
  store = data;
  createNetwork(data);
}); //Cargamos nuestro archivo de datos y ejecutamos la función create network
function createNetwork(edgelist) {
  var nodeHash = {};
  var nodes = [];
  var edges = [];
  var candidatos = [];
  var categorias = [];
  var aux = 0;
  var aux2 = 0;
  var tip = d3
    .tip()
    .attr("class", "d3-tip")
    .offset([-10, 0])
    .html(function (d) {
      return (
        "<strong>Candidato:</strong> <span style='color:pink'>" +
        d.cnd.toUpperCase() +
        "</span>"
      );
    });
  var tip2 = d3
    .tip()
    .attr("class", "d3-tip")
    .offset([-10, 0])
    .html(function (d) {
      return (
        "<strong>Sim:</strong> <span style='color:pink'>" +
        parseFloat(d.weight).toFixed(2) * 10 +
        "%" +
        "</span>"
      );
    });

  var svg = d3
    .select("body")
    .append("svg")
    .attr("width", width)
    .attr("height", height);
  svg.call(tip);
  svg.call(tip2);
  var force = d3.layout
    .force()
    .gravity(0.08) //atracción entre nodos principales
    .charge(-500) //separación entre nodos de borde
    .size([width, height]); // tamaño del gráfico
```

```
        force.linkDistance(function(link) {
          return 200-(parseFloat(link.weight)*20);
        });
        obtCand(store);
        obtCat(store);
        isNew = 1;
        edgelist.forEach(function (edge) {
          //Funcion que selecciona del archivo los nodos principales (categorias) y los
guarda en un archivo especifico
          if (
           !nodes.some(
              (node) => node.label === edge.categoria.toUpperCase()
            ) &&
            !indexCat.includes(edge.categoria) &&
            !indexCand.includes(edge.candidato)
          ) {
            //verifica que no se repita la categoria
            nodeHash[aux] = {
              id: String(aux + 1),
              label: edge.categoria.toUpperCase(),
              group: 2,
              cnd: null,
            };
            nodes[aux] = nodeHash[aux];
            aux++;
          }
        });
        edgelist.forEach(function (edge) {
          //Funcion que selecciona del archivo los nodos secundarias (palabras) y los
guarda en un archivo especifico
          if (
            !nodes.some(
              (node) => node.label === edge.word && node.cnd === edge.candidato
            ) &&
            edge.sim > slider.value &&
            !indexCat.includes(edge.categoria) &&
            !indexCand.includes(edge.candidato)
          ) {
            //verifica que no se repita la palabra
            nodeHash[aux] = {
              id: String(aux + 1),
              label: edge.word,
              group: 1,
              cnd: edge.candidato,
            };
            nodes[aux] = nodeHash[aux];
            aux++;
          }
```

```
        });
        edgelist.forEach(function (edge) {
          //Funcion que enlaza los nodos principales con los secundarios en pares de
acuerdo a su relación
          if (
            edge.sim > slider.value &&
            !indexCat.includes(edge.categoria) &&
            !indexCand.includes(edge.candidato)
          ) {
            var src =
              nodes[
                nodes.findIndex(
                  (node) => node.label === edge.categoria.toUpperCase()
                )
              ];
            var trg =
              nodes[
                nodes.findIndex(
                  (node) =>
                    node.label === edge.word && node.cnd === edge.candidato
                )
              ];
            var wgt = String(edge.sim * 10);
            var temp = { source: src, target: trg, weight: String(wgt) };
            edges[aux2] = temp;
            aux2++;
          }
        });
        force.nodes(nodes).links(edges).start();
        var link = svg
          .selectAll(".link")
          .data(edges) //Asignacion de la data para cada enlace
          .enter()
          .append("line") //Por cada enlace se genera una linea
          .attr("class", "link") //Se setean las caracteristica de .link
          .style("stroke-width", function (d) {
            return d.weight/5;
          });

        // Create the node circles.
        var node = svg
          .selectAll(".node")
          .data(nodes) //Asignación de la data para cada nodo
          .enter()
          .append("g") //Por cada nodo se agrega un circulo
          .attr("class", "node") //Se setean las caracteristicas de .node
          .call(force.drag);
        node
```

```
  .append("circle") //Caracteristicas de los nodos
  .attr("r", function (d) {
    return String(d.group * 10);
  })
  .style("fill", function (d) {
    if (d.cnd === null) {
      return color(d.group);
    } else if (d.cnd === "LASSO") {
      return "#CC33CC";
    } else if (d.cnd === "ARAUZ") {
      return "#1C542D";
    } else if (d.cnd === "YAKU") {
      return "#00BB2D";
    } else if (d.cnd === "HERVAS") {
      return "#FF8000";
    } else if (d.cnd === "FREILE") {
      return "#FFFF00";
    }else if (d.cnd === "GUTIERREZ") {
      return "#FF0000";
    }
  })
  .on("mouseover", tip.show)
  .on("mouseout", tip.hide);
node
  .append("text") //Caractersiticas del texto de los nodos
  .attr("dx", 20) //Ubicación respecto al nodo en X
  .attr("dy", ".35em") //Ubicacion respecto al nodo en Y
  .style("fill", "black") // Estilo del texto
  .text(function (d) {
    if (d.group == 2) {
      return d.label;
    } else {
      return d.label;
    }
  });
link.on("mouseover", tip2.show).on("mouseout", tip2.hide);
force.on("tick", function () {
  var k = 0;
  while (force.alpha() > 1 && k < 150) {
    force.tick();
    k = k + 10;
  }
  link
    .attr("x1", function (d) {return d.source.x;})
    .attr("y1", function (d) {return d.source.y;})
    .attr("x2", function (d) {return d.target.x;})
    .attr("y2", function (d) {return d.target.y;});
```

```
    node.attr("transform", function (d) {
      return "translate(" + d.x + "," + d.y + ")";
    });
  });
}
function updateNetwork() {
  d3.select("svg").remove();
  d3.csv("elecciones_final.csv", function (error, data) {
    createNetwork(store);
  });
}
  function dataRepetidaCand(elementos, nombre) {
      return elementos.filter(
          function(elementos){ return elementos.NomCand == nombre }
      );
  }
  function dataRepetidaCat(elementos, nombre) {
      return elementos.filter(
          function(elementos){ return elementos.NomCat == nombre }
      );
  }
function obtCand(datos) {
  var aux=0;
      var result = datos.reduce((result,d) => {
          result = Object.keys(result).map(key => result[key])
          if(result != undefined){
             var checkRep = dataRepetidaCand(result, d.candidato);
              if(checkRep.length == 0){
                  var temp = {"IdCand": aux + 1, "NomCand": d.candidato};
                  result.push(temp);
                  aux = aux + 1;
              }
          }
          return result;
      },{});
  result = Object.keys(result).map((key) => result[key]);
  //result = sinRep(result);
  candidatos = result;
  if (isNew == 0) {
    checkBCandidatos(candidatos);
  }
  return result;
}
function obtCat(datos) {
  var aux=0;
      var result = datos.reduce((result,d) => {
          result = Object.keys(result).map(key => result[key])
          if(result != undefined){
```

```
            var checkRep = dataRepetidaCat(result, d.categoria);
             if(checkRep.length == 0){
                 var temp = {"CatID": aux + 1, "NomCat": d.categoria};
                 result.push(temp);
                 aux = aux + 1;
             }
         }
         return result;
      },{});
  categorias = result;
  if (isNew == 0) {
    checkBCategorias(categorias);
  }
  return result;
}
function checkBCandidatos(cand) {
  var aux = 0;
  cand.forEach((elemento) => {
    var node = document.createElement("div");
    node.innerHTML =
      '<input type="checkbox" id="checkC' +
      aux +
      '" checked="" onclick="isCheck()"><label for="checkC' +
      aux +
      '">' +
      elemento.NomCand.toUpperCase() +
      "</label><br>";
    document.getElementById("cand").appendChild(node);
    aux = aux + 1;
  });
}
function checkBCategorias(cat) {
  var aux = 0;
  cat.forEach((elemento) => {
    var node = document.createElement("div");
    node.innerHTML =
      '<input type="checkbox" id="check' +
      aux +
      '" checked="" onclick="isCheck()"><label for="check' +
      aux +
      '">' +
      elemento.NomCat.toUpperCase() +
      "</label><br>";
    document.getElementById("cat").appendChild(node);
    aux = aux + 1;
  });
}
function isCheck() {
```

```
        var aux;
        var aux1 = 0;
        var aux2;
        var aux3 = 0;
        indexCand = [];
        indexCat = [];
        for (aux = 0; aux < categorias.length; aux++) {
          var isChecked = document.getElementById("check" + aux + "").checked;
          if (isChecked != true) {
            indexCat[aux1] = categorias[aux].NomCat;
            aux1 = aux1 + 1;
          }
        }
        for (aux2 = 0; aux2 < candidatos.length; aux2++) {
          var isChecked2 = document.getElementById(
            "checkC" + aux2 + ""
          ).checked;
          if (isChecked2 != true) {
            indexCand[aux3] = candidatos[aux2].NomCand;
            aux3 = aux3 + 1;
          }
        }
        console.log(indexCat);
        console.log(indexCand);
        updateNetwork();
      }
    </script>
  </div>
</body>
```

ANEXO C

CÓDIGO DE IMPLEMENTACIÓN DE LA PRIMERA VISUALIZACIÓN “COMPARATIVA DE TENDENCIA DEL DISCURSO POLÍTICO”

```
<!DOCTYPE html>
<meta charset="utf-8">
<head>
    <link rel="stylesheet" href="tabla.css">
    <link rel="stylesheet" href="comparativa.css">
    <style>
        body {
              font-family:Arial, Helvetica, sans-serif;
              overflow: hidden
          }
        .link {
          stroke: #aaa;
        }
        .node text {
        stroke:#333;
        cursor:pointer;
        }
        .node circle {
        stroke: #fff;
        stroke-width: 1.5px;
        fill:#555;
        }
        .d3-tip {
          line-height: 1;
          font-weight: bold;
          padding: 14px;
          background: rgba(0, 0, 0, 0.8);
          color: #fff;
          border-radius: 2px;
        }
        .d3-tip:after {
          box-sizing: border-box;
          display: inline;
          font-size: 14px;
          width: 100%;
          line-height: 1;
          color: rgba(0, 0, 0, 0.8);
          content: "\25BC";
          position: absolute;
          text-align: center;
        }
        .d3-tip.n:after {
```

```
            margin: -1px 0 0 0;
            top: 100%;
            left: 0;
        }
    </style>
</head
<body>
    <h2 align= center >COMPARATIVA DE TENDENCIA DEL DISCURSO POLÍTICO</h2>
    <div style="inline-block">
        <aside style = "float: left; display: inline-block" >
            <div>
                <div id="cat" ><h3>Categorias</h3></div>
                <div id="cand" ><h3>Candidatos</h3></div>
                <div class="slidecontainer">
                    <h3>Indice de similitud</h3>
                    <input type="range" min="0" max="1" step="0.005" value="0.5"
                    class="slider" id="myRange" onclick="updateNetwork()">
                    <p id='sim'>SIM: <span id="demo"></span></p>
                </div>
                <div id="cand" >
                    <h3>Fecha de comparación</h3>
                    <input type="date" id="start" name="trip-start" value="2021-02-08"
min="2020-12-30" max="2021-05-31" onchange="updateNetwork()">
                </div>
            </div>
        </aside>
        <div id="main3" class="main3" style="float: left; display: inline-block">
            <div id="labels" class="labels" style="float: center">
                <div id='before' style="display: inline-block; width:44%"><p align=
center>ANTES</p></div>
                <div id='date' style="display: inline-block; width:11%"><p id='fecha'
align= center>2021-02-08</p></div>
                <div id='after' style="display: inline-block; width:44%"><p align=
center>DESPUÉS</p></div>
                <div id='main4' style="width:100%"></div>
            </div>
        </div>
    </div>
<script type="text/javascript" src="https://d3js.org/d3.v3.min.js"></script>
<script src="https://labratrevenge.com/d3-tip/javascripts/d3.tip.v0.6.3.js"></script>
<script>
var store=[];
var indexCat = [];
var indexCand = [];
var isNew = 0;
var linkpath = ("links.csv"); //archivo de enlaces
var nodepath = ("nodes.csv"); //archivo de nodos
var width   = 1500, //seteo de tamaño del grafo
```

```
    height  = 1500;
var color = d3.scale.category20(); //Gama de colores seteados por d3.js
var slider = document.getElementById("myRange");
var output = document.getElementById("demo");
    output.innerHTML = slider.value;
    slider.oninput = function() {
        output.innerHTML = this.value;
    }
d3.csv("elecciones_final.csv",function(error,data) {store = data;
createNetwork(data);}); //Cargamos nuestro archivo de datos y ejecutamos la función
create network
function createNetwork(edgelist) {
  var nodeHash = {};
  var nodes = [];
  var edges = [];
  var candidatos = [];
  var categorias = [];
  var aux = 0;
  var aux2 = 0;
  var fechaInput = new Date();
  var tip = d3.tip()
  .attr('class', 'd3-tip')
  .offset([-10, 0])
  .html(function(d) {
    return "<strong>Candidato:</strong> <span style='color:pink'>" + d.cnd.toUpperCase()
+ "</span>";
  })
  var tip2 = d3.tip()
  .attr('class', 'd3-tip')
  .offset([-10, 0])
  .html(function(d) {
    return "<strong>Sim:</strong> <span style='color:red'>"
+parseFloat(d.weight).toFixed(2)*10+ "%"+ "</span>";
  })
  var svg = d3.select("#main4").append("svg")
    .attr("width", width)
    .attr("height", height);
  svg.call(tip);
  svg.call(tip2);
  var force = d3.layout.force()
    .gravity(.08)  //atracción entre nodos principales
    .distance(100) //separacion nodos principales - nodos de borde
    .charge(-500)  //separación entre nodos de borde
    .size([width, height]); // tamaño del gráfico

  obtCand(store);
  obtCat(store);
  isNew = 1;
```

```
  edgelist.forEach(function (edge) { //Funcion que selecciona del archivo los nodos
principales (categorias) y los guarda en un archivo especifico
    if (!nodes.some(
              (node) => node.label === edge.categoria.toUpperCase()
            ) && !(indexCat.includes(edge.categoria)) &&
!(indexCand.includes(edge.candidato))) { //verifica que no se repita la categoria
        nodeHash[aux] = {id: String(aux + 1), label: edge.categoria.toUpperCase(),
group: 2, cnd: null};
        nodes[aux] = nodeHash[aux];
        aux++;
    }
  })
  edgelist.forEach(function (edge) { //Funcion que selecciona del archivo los nodos
secundarias (palabras) y los guarda en un archivo especifico
    if(!(nodes.some(node => (node.label === edge.word && node.cnd === edge.candidato)))
&& edge.sim > slider.value && !(indexCat.includes(edge.categoria))&&
!(indexCand.includes(edge.candidato))){ //verifica que no se repita la palabra
        nodeHash[aux] = {id: String(aux + 1), label: edge.word, group: 1, cnd:
edge.candidato, date: edge.fecha, sim: edge.sim};
        nodes[aux] = nodeHash[aux];
        aux++;
    }
  })
  edgelist.forEach(function (edge) { //Funcion que enlaza los nodos principales con los
secundarios en pares de acuerdo a su relación
    if(edge.sim > slider.value && !(indexCat.includes(edge.categoria)) &&
!(indexCand.includes(edge.candidato)) ){
        var src = nodes[nodes.findIndex(node => node.label ===
edge.categoria.toUpperCase())];
        var trg = nodes[nodes.findIndex(node => (node.label === edge.word && node.cnd
=== edge.candidato))];
        var wgt = String(edge.sim*10);
        var temp = {source: src, target: trg, weight: String(wgt)}
        edges[aux2] = temp;
        aux2++;
    }
  })
    force.nodes(nodes).links(edges).start();
    console.log(edges);
    console.log(nodes);
    console.log(fechaInput);
    fechaInput = start.value;
    console.log(fechaInput);

var link = svg.selectAll(".link")
    .data(edges) //Asignacion de la data para cada enlace
    .enter().append("line") //Por cada enlace se genera una linea
    .attr("class", "link")  //Se setean las caracteristica de .link
```

```
    .style("stroke-width", function(d) { return Math.sqrt(d.weight); });
  var node = svg.selectAll(".node")
      .data(nodes) //Asignación de la data para cada nodo
      .enter().append("g") //Por cada nodo se agrega un circulo
      .attr("class", "node") //Se setean las caracteristicas de .node
      .call(force.drag);
        node
          .append("circle") //Caracteristicas de los nodos
          .attr("r", function (d) {
            return String(d.group * 10);
          })
          .style("fill", function (d) {
            if (d.cnd === null) {
              return color(d.group);
            } else if (d.cnd === "LASSO") {
              return "#CC33CC";
            } else if (d.cnd === "ARAUZ") {
              return "#1C542D";
            } else if (d.cnd === "YAKU") {
              return "#00BB2D";
            } else if (d.cnd === "HERVAS") {
              return "#FF8000";
            } else if (d.cnd === "FREILE") {
              return "#FFFF00";
            } else if (d.cnd === "GUTIERREZ") {
              return "#FF0000";
            }
          })
          .on("mouseover", tip.show)
          .on("mouseout", tip.hide);
  node.append("text") //Caractersiticas del texto de los nodos
      .attr("dx", 20) //Ubicación respecto al nodo en X
      .attr("dy", ".35em") //Ubicacion respecto al nodo en Y
      .style("fill","black") // Estilo del texto
      .text(function(d) {if(d.group == 2){return d.label}else{return d.label}}); //Texto
a mostrar por nodo
  link.on('mouseover', tip2.show)
      .on('mouseout', tip2.hide);
  force.on("tick", function() {
    var k = 0;
    while ((force.alpha() > 1) && (k < 150)) {
        force.tick();
        k = k + 10;
    }
    var ejeY = 100;
    nodes.forEach(function(node){
        if(node.cnd == null){
            node.x = 750;
```

```
            node.y = ejeY;
            ejeY = ejeY + 115;
        }
    });
    edges.forEach(function(edge){
        edge.source.x = 750;
        if(edge.target.date <= fechaInput){
            edge.target.x = edge.target.sim*700+50;
        }else if(edge.target.date > fechaInput){
            edge.target.x = 1450-edge.target.sim*700;
        }
    });
    link.attr("x1", function(d) { return d.source.x; })
        .attr("y1", function(d) { return d.source.y; })
        .attr("x2", function(d) { return d.target.x; })
        .attr("y2", function(d) { return d.target.y; });
    node.attr("transform", function(d) { return "translate(" + d.x + "," + d.y + ")";
});
  });
}
function updateNetwork(){
    updateDateLabel(start.value);
    d3.select("svg").remove();
    d3.csv("elecciones_final.csv",function(error,data) {createNetwork(store)});
}
function dataRepetidaCand(elementos, nombre) {
    return elementos.filter(
        function(elementos){ return elementos.NomCand == nombre }
    );
}
function dataRepetidaCat(elementos, nombre) {
    return elementos.filter(
        function(elementos){ return elementos.NomCat == nombre }
    );
}
function obtCand(datos){
        var aux=0;
            var result = datos.reduce((result,d) => {
                result = Object.keys(result).map(key => result[key])
                if(result != undefined){
                   var checkRep = dataRepetidaCand(result, d.candidato);
                    if(checkRep.length == 0){
                        var temp = {"IdCand": aux + 1, "NomCand": d.candidato};
                        result.push(temp);
                        aux = aux + 1;
                    }
                }
                return result;
```

```
            },{});
        result = Object.keys(result).map((key) => result[key]);
        //result = sinRep(result);
        candidatos = result;
    if(isNew == 0){
        checkBCandidatos(candidatos);
    }
    return result;
}
function obtCat(datos){
        var aux=0;
            var result = datos.reduce((result,d) => {
                result = Object.keys(result).map(key => result[key])
                if(result != undefined){
                   var checkRep = dataRepetidaCat(result, d.categoria);
                    if(checkRep.length == 0){
                        var temp = {"CatID": aux + 1, "NomCat": d.categoria};
                        result.push(temp);
                        aux = aux + 1;
                    }
                }
                return result;
            },{});
        categorias = result;
    if(isNew == 0){
         checkBCategorias(categorias);
    }
    return result;
}
function checkBCandidatos(cand){
    var aux = 0
     cand.forEach(elemento => {
        var node = document.createElement('div');
        node.innerHTML = '<input type="checkbox" id="checkC'+aux+'" checked=""
onclick="isCheck()"><label
for="checkC'+aux+'">'+elemento.NomCand.toUpperCase()+'</label><br>';
        document.getElementById('cand').appendChild(node);
        aux=aux+1;
    })
}
function checkBCategorias(cat){
    var aux = 0
    cat.forEach(elemento => {
        var node = document.createElement('div');
        node.innerHTML = '<input type="checkbox" id="check'+aux+'" checked=""
onclick="isCheck()"><label
for="check'+aux+'">'+elemento.NomCat.toUpperCase()+'</label><br>';
        document.getElementById('cat').appendChild(node);
```

```
        aux=aux+1;
    })
}
function sinRep(result){
    let aux = 0;
    let result1 = [];
    result.forEach(element => {
        if(result1.length == 0){
            result1.push(element);
        }
        else if (result1.length > 0){
            result1.forEach(el => {
            if(el.NomCand == element.NomCand){
           aux = 1;
            }
        })
    if(aux != 1){
        result1.push(element);
        aux = 0;
    }
    }
     })
    return result1;
}
function isCheck(){
    var aux;
    var aux1=0;
    var aux2;
    var aux3=0;
    indexCand=[];
    indexCat = [];
    for(aux = 0;aux < categorias.length;aux++){
        var isChecked = document.getElementById('check'+aux+'').checked;
        if(isChecked != true){
            indexCat[aux1] = categorias[aux].NomCat;
            aux1=aux1+1;;
        }
    }
    for(aux2 = 0;aux2 < candidatos.length;aux2++){
        var isChecked2 = document.getElementById('checkC'+aux2+'').checked;
        if(isChecked2 != true){
            indexCand[aux3] = candidatos[aux2].NomCand;
            aux3=aux3+1;
        }
    }
    console.log(indexCat);
    console.log(indexCand);
    updateNetwork();
```

```
}
function updateDateLabel(dateLabel){
    document.getElementById("fecha").innerHTML = ""
    var texto = document.createTextNode(dateLabel);
    document.getElementById('fecha').appendChild(texto);
}
</script>
</body>
```

ANEXO D

CÓDIGO DE IMPLEMENTACIÓN DE LA INTERFAZ WEB

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, user-scalable=no, initial-
scale=1.0, maximum-scale=1.0, minimum-scale=1.0">
    <title>¿De qué hablan nuestros candidatos</title>
    <link rel="stylesheet" href="estilos.css">
    <link rel="stylesheet" href="font-awesome.css">

    <script src="https://code.jquery.com/jquery-3.2.1.min.js"></script>
    <script src="main.js"></script>
</head>
<body>
    <div class="wrap">
        <div class="cabecera">
            <img src="./assets/img/buho.png" alt="">
            <h1>¿De qué hablan nuestros candidatos?</h1>
        </div>

        <ul class="tabs">
            <li><a href="#tab1"><span class="fa fa-home"></span><span class="tab-
text">Inicio</span></a></li>
            <li><a href="#tab2"><span class="fa fa-group"></span><span class="tab-
text">Tabla de tendencia</span></a></li>
            <li><a href="#tab3"><span class="fa fa-briefcase"></span><span class="tab-
text">Red de tendencia</span></a></li>
            <li><a href="#tab4"><span class="fa fa-bookmark"></span><span class="tab-
text">Comparativa de tendencia</span></a></li>
        </ul>

        <div class="secciones">
            <article id="tab1">
                <div class="about">
                    <h1>RESEÑA</h1>
                    <p>Los medios usados por los candidatos para difundir sus propuestas
y su discurso político han venido en constante cambio a lo largo de los años pasando por
eventos públicos, periódicos, radio, televisión y llegando a tener un gran impacto en lo
que respecta a redes sociales. Actualmente son un espacio utilizado por líderes
políticos para dar a conocer detalles de sus propuestas, siendo Twitter, Facebook e
Instagram las aplicaciones que más relevancia toman en este aspecto. y que son
ampliamente usados en distintos tópicos. Uno de los problemas que tiene el ciudadano es
la dificultad de identificar las tendencias de los políticos ante la extensa cantidad de
información que se puede encontrar en estas redes sociales.</p>
```

```
                    <br>
                    <p>Esta aplicación plantea la creación de tres visualizaciones a
partir de la información obtenida del proceso electoral 2020-2021, que permitirá al
ciudadano conocer si los candidatos hablan acerca de distintas temáticas trascendentales
en un país como son educación, salud, economía, ambiente, entre otras, y como las
publicaciones que realizan en redes sociales se asocian a las temáticas previamente
mencionadas dándole al usuario una idea clara de la tendencia que tienen en su discurso
político los distintos candidatos.</p>
                    <br>
                    <p>Cada palabra que observarán en las distintas visualizaciones son
un extracto representativo de los distintos tweets que los candidatos presidenciales han
posteado es Twitter y que posteriormente serán relacionadas a las temáticas mencionadas
anteriormente. </p>
                </div>
            </article>
            <article id="tab2">
                <iframe src="./web/Tabla de tendencia del discurso político.html"
width="100%" height="100%"></iframe>
            </article>
            <article id="tab3">
                <iframe src="./web/Red de tendencia del discurso político.html"
width="100%" height="100%"></iframe>
            </article>
            <article id="tab4">
                <iframe src="./web/Comparativa de tendencia del discurso político.html"
width="100%" height="100%"></iframe>
            </article>
        </div>
    </div>
</body>
</html>
```