Instituto Tecnológico de Buenos Aires
Escuela de ingeniería y gestión

Visualización de noticias

Autores:

Leandro Matías Rivas (L: 51274)
Sebastián Andrés Maio (L: 50386)

Tutor:

Lic. Diego Ariel Aizemberg

Proyecto final presentado para la obtención del título de Ingeniero en Informática

Lugar: Ciudad Autónoma de Buenos Aires, Argentina
Fecha: 15/12/2020

Abstract

Este trabajo parte de 2 proyectos finales anteriores, XDATA[1] (Recopilación y visualización de noticias) y NERD[2] (Analizador de texto para reconocimiento de entidades nombradas), donde se buscó mejorar la visualización de noticias utilizando técnicas de reconocimiento de entidades.

El proyecto actual tuvo como objetivo mejorar la interfaz y usabilidad de XDATA al igual que ampliar sus funcionalidades e incorporar el reconocimiento de entidades en las mismas. Para esto se volvió a implementar el sitio de manera tal que la misma se comunica vía API con los servidores, desacoplando completamente sus implementaciones.

[1] https://pf2.it.itba.edu.ar/nosotros
[2] http://nerd.it.itba.edu.ar/

Índice

Abstract **1**

Justificación del trabajo **3**

Estado del arte **4**

Objetivos **5**

Usabilidad 5

Tecnologías 5

Funcionalidad 5

Planificación del proyecto **6**

Evaluación 6

Reimplementación 6

Extensión 6

Requerimientos **7**

Funcionales 7

Técnicos 7

Análisis **8**

Diseño **9**

Arquitectura 9

Datos de implementación **11**

Datos técnicos 11

Modificaciones en XDATA API 12

Visualizaciones 13

Prototipo **23**

Conclusiones **34**

Trabajo futuro **35**

Apéndice **36**

Justificación del trabajo

Anterior a este proyecto, el tutor lideró los proyectos XDATA y NERD, los cuales cumplieron con sus objetivos. En caso de XDATA se recuperaron diariamente las noticias a través de sus RSS, y ofrecieron algunas maneras de visualizar las mismas. Por otro lado, la NERD API desarrolló, entre otras cosas, el reconocimiento de entidades nombradas.

A partir de estos dos, surge la posibilidad de utilizar el analizador NERD, junto con la base de datos de XDATA (y las respectivas APIs) para conseguir nuevas formas de visualizar y analizar las noticias.

Hacer uso de ambos servicios permitiría disponer de datos que anteriormente el sitio no contaba, habilitando la posibilidad de utilizar nuevas visualizaciones y crear funcionalidades que mejoren la exploración de las noticias.

Estado del arte

Actualmente, las maneras de visualizar las noticias que nos brindan los medios tienden a ser simples, donde la noticia está formada por el título, una imagen y la noticia propiamente dicha.

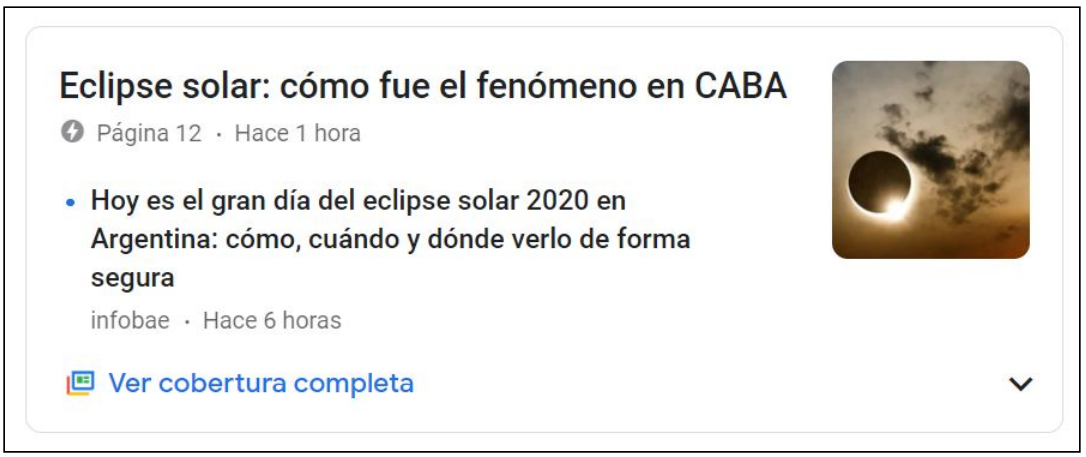

Figura 1. Noticia sobre el eclipse solar.

En ocasiones, cuando la información que se quiere brindar es de importancia, como puede ser las elecciones presidenciales de un país o la pandemia, estas visualizaciones son mucho más sofisticadas. No solamente muestran datos mediante una o más visualizaciones o gráficos interactivos, sino que también hacen un análisis del mismo e inclusive a veces conclusiones o resultados. Este tipo de visualizaciones suelen ser creados por equipos interdisciplinarios, donde podemos diferenciar a aquellos que realizan la recolección de datos, los que desarrollan el gráfico y quienes brindan el toque artístico a la visualización. Esto se puede apreciar en los trabajos realizados por el equipo de La Nacion Data[3].
Existen muchas librerías y programas con un amplio espectro de visualización y gráficos disponibles que podrían utilizarse para noticias, lamentablemente utilizarlas para dichos fines no es una tarea fácil.

[3] https://www.lanacion.com.ar/data

Objetivos

El objetivo fue mejorar la UI del portal XDATA para hallar nuevas maneras de explorar y visualizar las noticias. Las mejoras propuestas fueron tanto de usabilidad, tecnologías y funcionalidad.

Usabilidad

- Mejorar la utilización del sitio en dispositivos móviles.
- Mejorar los tiempos de respuesta.

Tecnologías

- Utilizar tecnologías modernas.
- Desacoplar la UI de la API.

Funcionalidad

- Investigar e incorporar nuevas maneras de visualizar las noticias.
- Incorporar reconocimiento de entidades a las visualizaciones preexistentes.
- El sitio debe permitir embeber visualizaciones en otros sitios.

Planificación del proyecto

El proyecto se dividió en 3 fases principales: Evaluación, re-implementación y extensión. Para organizar mejor las tareas se utilizó Trello, donde se hizo un seguimiento de las tareas a realizar.

Evaluación

En esta fase se analizó con qué tecnologías y cómo se había construido XDATA.
Se decidió que se haría una implementación nueva para la interfaz, desacoplando a esta del backend utilizando un *framework* moderno. En esta fase también se definieron los requerimientos del proyecto.

Reimplementación

Esta fase del proyecto fue la migración de lo que ya existía en XDATA a la nueva UI. Durante este periodo se realizaron reuniones mensuales con el tutor del proyecto, para mostrar los avances y discutir cambios, mejoras y modificaciones respecto de XDATA.

Extensión

Una vez terminada la reimplementación, el trabajo se realizó con una metodología ágil, se cambió la frecuencia a reuniones semanales con el tutor, donde se mostraba el trabajo hecho en la semana, compartían prototipos nuevos e ideas y se definía con que se seguiría la semana entrante.

Requerimientos

Los requerimientos se pueden dividir en las siguientes categorías: Funcionales, Técnicos.

Funcionales

- El sitio debía ofrecer nuevas maneras de visualizar las noticias
- El sitio debía ofrecer una manera fácil de ser embebido en otros sitios
- El sitio debe poder visualizarse correctamente en dispositivos móviles
- El sitio debía conectarse con NERD API para obtener entidades y visualizarlas

Técnicos

- El sitio debía estar hecho en una tecnología moderna
- El código del sitio debía estar desacoplado del de los servidores
- El sitio debía comunicarse con los servidores vía API REST
- El sitio debía comunicarse con servicios que utilicen protocolo HTTPS
- El sitio debía evitar hacer consultas evitables a los servidores para mejorar los tiempos de respuesta

Análisis

XDATA fue construido con DJANGO, tanto el *backend*, la API y el *frontend*, utilizando una arquitectura MVC. La base de datos está en PostgreSQL y se ha instalado Elasticsearch para agilizar las búsquedas.

Tras un análisis del código se decidió que era deseable separar el *backend* y el *frontend*, para que en un futuro proyecto este se pueda mejorar/reemplazar sin necesidad de reemplazar el *frontend*.

Esto permitió utilizar un *Framework* para desarrollo web, para esto se consideraron React y Angular. Ambas permiten crear *single page applications* (SPA), y tener el desprendimiento deseado entre el *frontend* y el *backend*.

Tras un análisis de ambos en el cual se determinó que los objetivos podrían cumplirse con ambos, por lo cual el factor determinante fue la experiencia previa que poseía el equipo con Angular.

Diseño

Arquitectura

Para la implementación se decidió crear una SPA, utilizando el *framework* Angular. Esto permite reducir tiempos de carga al navegar entre distintas rutas y delegar la navegación y carga de recursos al servidor web.
En vez de crear páginas o vistas, se diseñó una arquitectura de componentes tal como plantea Angular, en donde en vez de organizar el proyecto en paginas se organiza en componentes. Las componentes permiten una mejor separación de código y responsabilidades, mejorando la reusabilidad del mismo y reduciendo el código duplicado.
Cada componente cuenta con 3 archivos, un archivo de TypeScript donde se encuentra el comportamiento/lógica de la componente. Un archivo HTML donde está la estructura/cuerpo del documento y por último un archivo SCSS donde está el estilo.
La comunicación con las API es mediante servicios, los cuales se encargan de los pedidos HTTP, liberando a las componentes de esa lógica. Los servicios son clases de TypeScript que Angular permite inyectar en otros servicios o componentes, sirviendo así de facilitadores de funcionalidades o datos.
La aplicación web se comunica con dos APIs para conseguir datos, XDATA API y NERD API. De la primera se obtienen los datos relacionados a las noticias y a los medios de comunicación que las publicaron. La segunda es una API que mediante NLP (Natural Language Processing) devuelve entidades asociadas a un texto. Para el caso del sitio web, el texto enviado son las noticias, así pudiendo conseguirse las entidades intervinientes en las mismas.

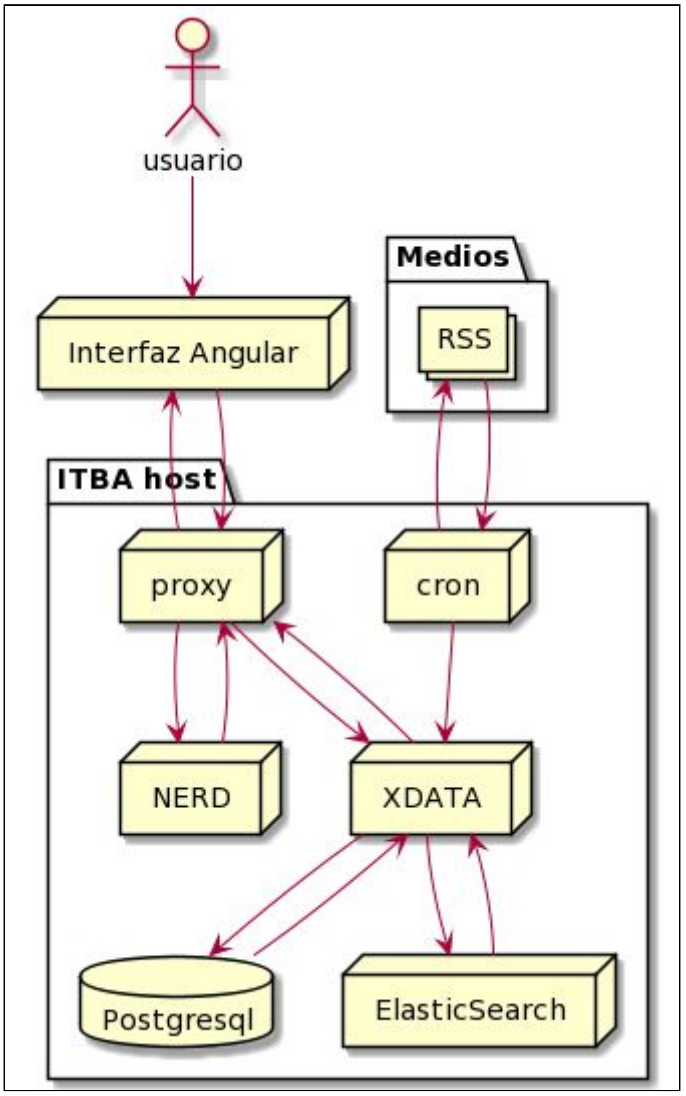

Figura 2. Diagrama de arquitectura del proyecto.

Datos de implementación

Datos técnicos

La implementación se hizo en Angular, se planteó una SPA organizada en componentes, servicios, directivas y modelos, lo cual respeta la organización que plantea el *framework*.
En los modelos se encuentra la representación en clases de los datos, destinados a facilitar su exposición y manejo principalmente.
Los servicios son clases inyectables, que se encargan de la obtención, producción y traspaso de datos. Todas las llamadas a las APIs para obtención de datos son a través de los servicios, al igual que el pase de datos entre componentes que no son padre e hijo, se realiza a través de los estos.
Las directivas indican cómo debe procesarse la información o comportamiento de cierto objeto HTML, en el proyecto se utilizaron para generar algunas visualizaciones.
Las componentes encapsulan comportamiento y vistas (o *templates* html), los cuales son utilizados por otras componentes para así generar la aplicación. Están compuestas por 3 archivos, un archivo HTML donde está la estructura, un archivo de TypeScript (.ts) donde se encuentra la lógica/comportamiento y un SASS donde se encuentra el estilo de la componente.
Un aspecto importante a resaltar, es la navegación y los *layouts*. Angular provee un *Router* el cual es el encargado de emular la navegación clásica de una página web y mostrar la componente correspondiente para la ruta solicitada.
En el *router* se define para cada ruta que componente será utilizada, pero además se puede asociar un *layout*. Un *layout* es un conjunto de componentes, que se muestran antes de mostrar la componente asociada a la ruta. Así es que para el sitio web se utiliza un *layout* que contiene la barra de navegación, los inputs de datos y el *footer*. En cambio cuando se quiere embeber el sitio, estos mismos no están presentes, ya que se llama desde el routing a un *layout* distinto que no posee dichos componentes, simplificando así la vista para que sea embebida.
La comunicación entre el sitio y el *backend*, al igual que con la nerd API es vía HTTPS. Para obtener información las componentes piden los datos a los servicios y estos lo piden a un servicio específico llamado DataService, el cual se encarga de de ensamblar la *query* que se enviara via HTTPS en base a los

filtros seleccionados por el usuario y, de ser el caso, los parámetros pasados por el servicio que está pidiendo dichos datos.
Además, DataService lleva un registro sobre el estado de los filtros, lo cual facilita que cada servicio pueda poseer un caché de memoria el cual almacena el resultado del último pedido realizado. Si algún componente vuelve a solicitar dicho pedido, el servicio válida contra el DataService si algún filtro cambio, en caso de que no hubiese cambios, devuelve el resultado que almacenó en vez de repetir el pedido anterior.
Como se ha mencionado el sitio permite ser embebido dentro de otros sitios si se antepone '/embed' a la ruta de la sección, por ejemplo, en vez de /títulos se va a /embed/títulos.
Cuando se utiliza dicha ruta, se muestra la visualización correspondiente pero sin la barra de navegación, selector de filtros ni *footer*. Los parámetros de los filtros pueden ser pasados vía la url. Así, en caso de embeber el sitio, se presenta una vista más limpia de la visualización.

Modificaciones en XDATA API

Para poder llevar a cabo nuevas visualizaciones se debió realizar cambios en el servicio REST para que éste acepte nuevos parámetros y, además, retorne nuevos datos. A continuación se listan los endpoints modificados:

GET /busqueda

Este *endpoint* siempre retornaba 20 noticias por consulta. Ahora, admite el parámetro *page_size* donde se le indica la cantidad de noticias por página. Este cambio nos permitió realizar menos consultas al servicio y obtener las respuestas en menos tiempo.

```
http://pf2.it.itba.edu.ar/api/busqueda?d_from=2020-12-12&d_to=2020-12-12&
words=coronavirus&page=1&sources=53&page_size=200
```

Figura 3. Consulta al servicio para obtener hasta 200 noticias para la palabra "coronavirus".

Además, este *endpoint* en su respuesta se le agregó el campo *date*. Esto permitió que se pueda realizar la visualización de la línea de tiempo de noticias.

```
{
  "status": "success",
```

```
  "data": [
    {
                                                                    "url":
"https://www.minutouno.com/notas/5154944-el-mundo-registra-las-peores-cif
ras-diarias-contagios-y-muertes-coronavirus",
      "source": "Minuto Uno",
       "summary": "Así lo informó la OMS en medio de la segunda ola en el
norte y el alerta por\nuna tercera. Suman 71 millones las personas
contagiadas y 1,6 millones las\nmuertas.",
      "date": "2020-12-12T20:24:00",
       "title": "El mundo registra las peores cifras diarias de contagios
y muertes por\ncoronavirus"
    }
  ]
}
```


Figura 4. Respuesta del endpoint Get /busqueda.

GET /medios

La modificación sobre este *endpoint* le permite retornar no solamente el nombre del medio y el id sino que también el *favicon*. Anteriormente, como el *frontend* y *backend* se encontraban acoplados, el *favicon* se obtenía consultando directamente a la base de datos.


```
{
  "status": "success",
  "data": [
    {
      "source": "Clarin",
      "id": 7,
      "favicon": "http://www.clarin.com/favicon.ico"
    }
  ]
}
```


Figura 5. Respuesta del endpoint Get /medios.

Visualizaciones

Títulos

En esta sección se pueden visualizar las noticias con sus respectivos resúmenes, hace cuanto tiempo se publicaron y la fuente que los publicó. El tamaño de página es variable, de acuerdo a un selector, pudiendo elegirse los valores

5-10-20-50-100. La vista *default* es la vista de lista, donde se pueden ver en una o dos columnas (según el tamaño del dispositivo) las noticias.

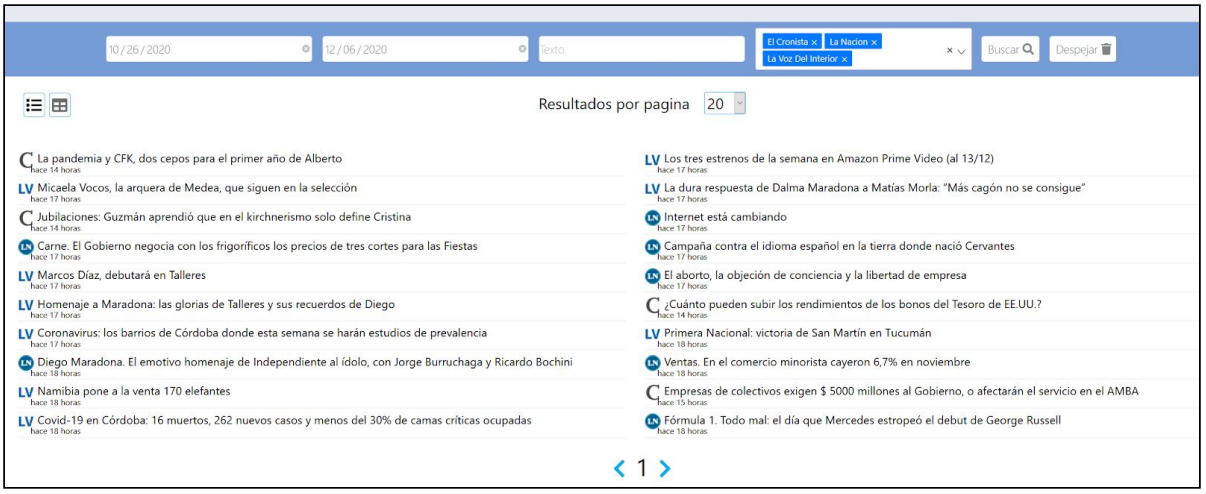

Figura 6. Títulos, vista de lista.

Asimismo esta sección ofrece una vista alternativa, la cual se puede ver pulsando los iconos ubicados arriba a la izquierda, el cual ofrece una variante visual en modo de tabla de cartas.
Esta vista alternativa ofrece un enlace al medio de publicación si se clickea el *favicon*.

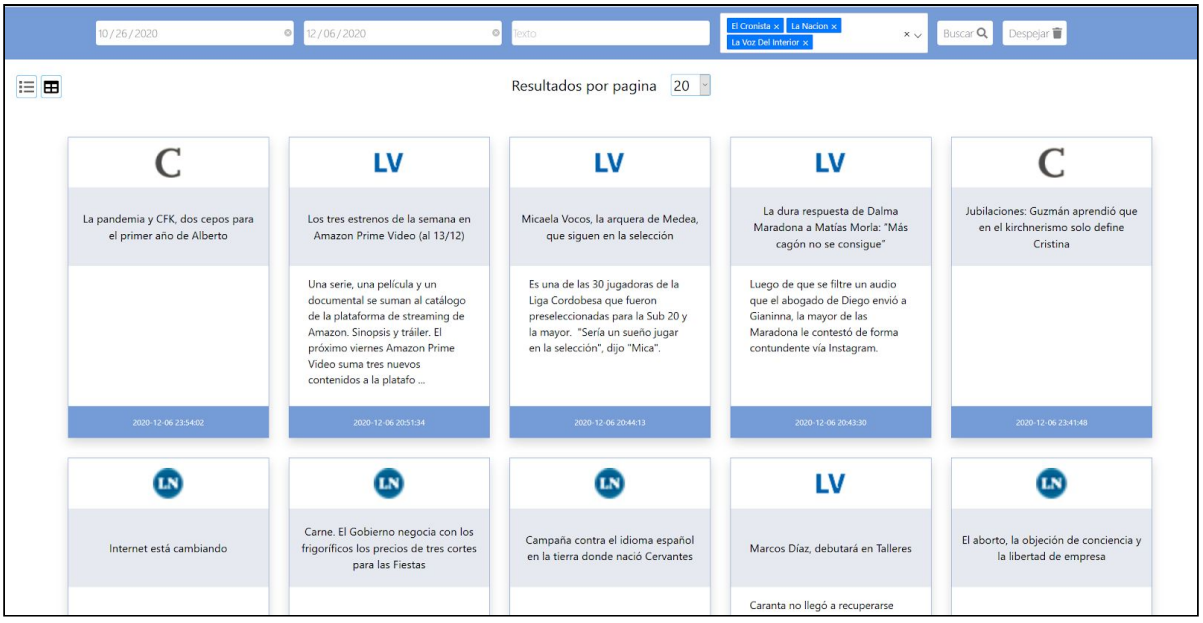

Figura 7. Títulos, vista de tabla de cartas.

Nube de palabras

Esta sección muestra la cantidad de incidencias que tuvo cada palabra en las noticias seleccionadas. Para ello se hicieron cuatro visualizaciones, un árbol de palabras, una nube de palabras ordenada, una nube de palabras desordenada y un gráfico de burbuja.
Para las cuatro visualizaciones, es posible seleccionar cuántas palabras se desean ver, este valor puede oscilar entre 0 y 200.

Treemap

En el treemap las palabras se ordenan de mayor frecuencia a menor frecuencia de arriba a abajo e izquierda a derecha. Entre más apariciones tenga la palabra, mayor es el cuadro que la contiene. Abajo de la palabra, se puede ver cuántas apariciones tuvo dicha palabra. Es posible desplazar con el *mouse* el gráfico, al igual que hacer clic en un cuadro para agrandarlo y centrar, o hacer zoom con la rueda del *mouse*. Esta visualización fue creada con la librería Apache ECharts.

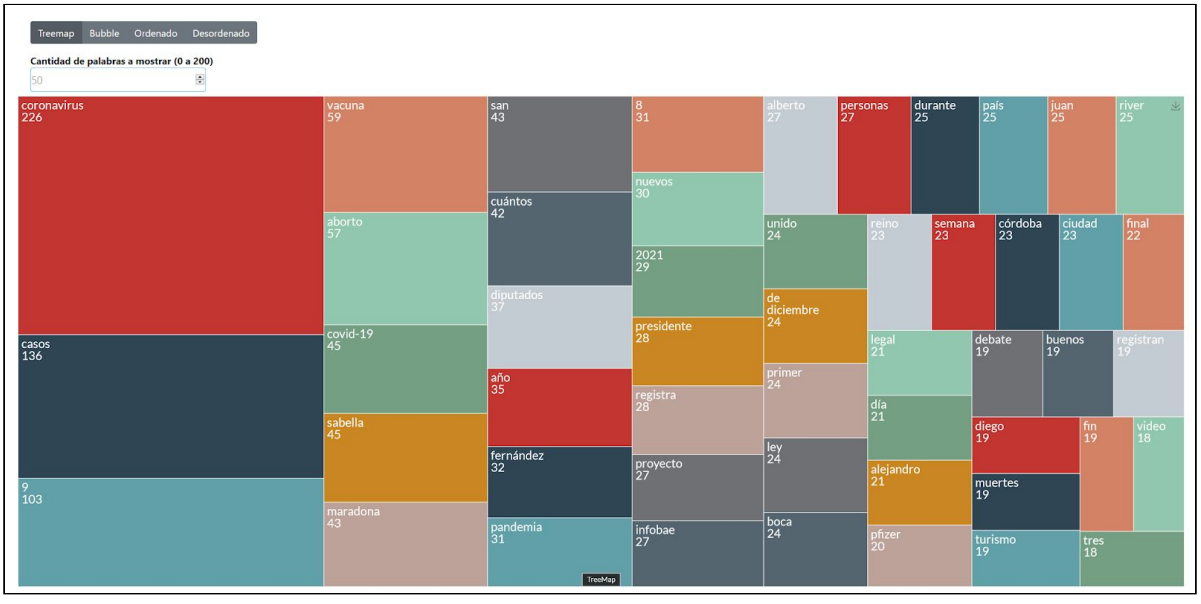

Figura 8. Nube de palabras, treemap.

Burbujas o Empaquetamiento de círculos

El tamaño de las burbujas es acorde a la cantidad de incidencias de la palabra, las más grandes están centradas. Debajo de cada palabra se puede apreciar la cantidad de incidencias que tuvo la palabra. Esta visualización fue creada con la librería D3.

Figura 9. Burbujas, gráfico de burbujas.

Nube de palabras desordenada

Las palabras son mostradas con un orden aleatorio, la cantidad de apariciones de cada palabra está representada por el tamaño de la misma. Esta visualización fue creada con la librería D3.

Figura 10. Nube de palabras, nube de palabras desordenadas.

Nube de palabras ordenada

Las palabras se ordenan de mayor a menor, según la frecuencia de aparición de cada palabra. El tamaño de cada palabra es proporcional a la frecuencia.

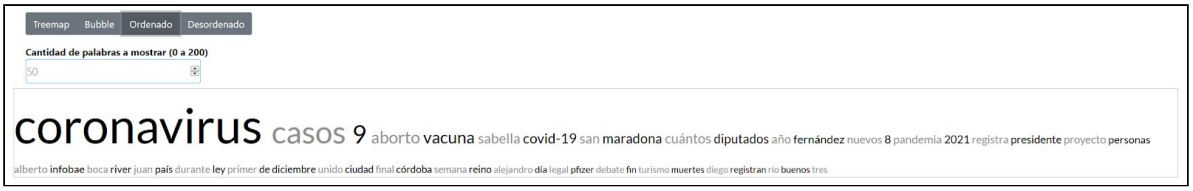

Figura 11. Nube de palabras, nube de palabras ordenadas.

Cantidad de noticias

Esta sección tiene dos posibles vistas, noticias del día y noticias del periodo.

Noticias del dia

Se muestra la cantidad de noticias que hubo en un día dado, el *default* es el día actual, en caso que no hubiera fechas seleccionadas. Si el rango de fechas seleccionado es de 1 solo dia, muestra la cantidad de noticias del dia seleccionado, de acuerdo a los filtros de búsqueda.

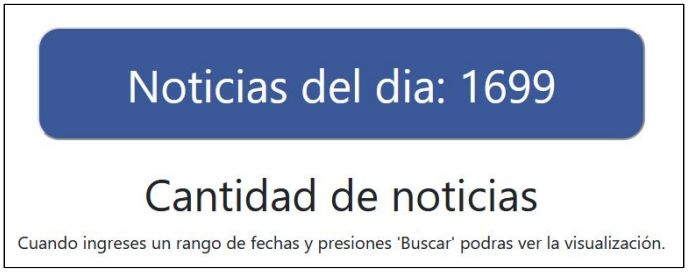

Figura 12. Cantidad de noticias, para un solo dia.

Noticias de un periodo

Permite la visualización de un gráfico de líneas, cuyos ejes muestran la cantidad de noticias de un dia (eje Y) y el dia en cuestion (eje X) para todas las fechas entre el periodo seleccionado.
Si no se especifica ninguna fuente, se muestra un gráfico con la cantidad de noticias de todos los medios representados en una sola línea.

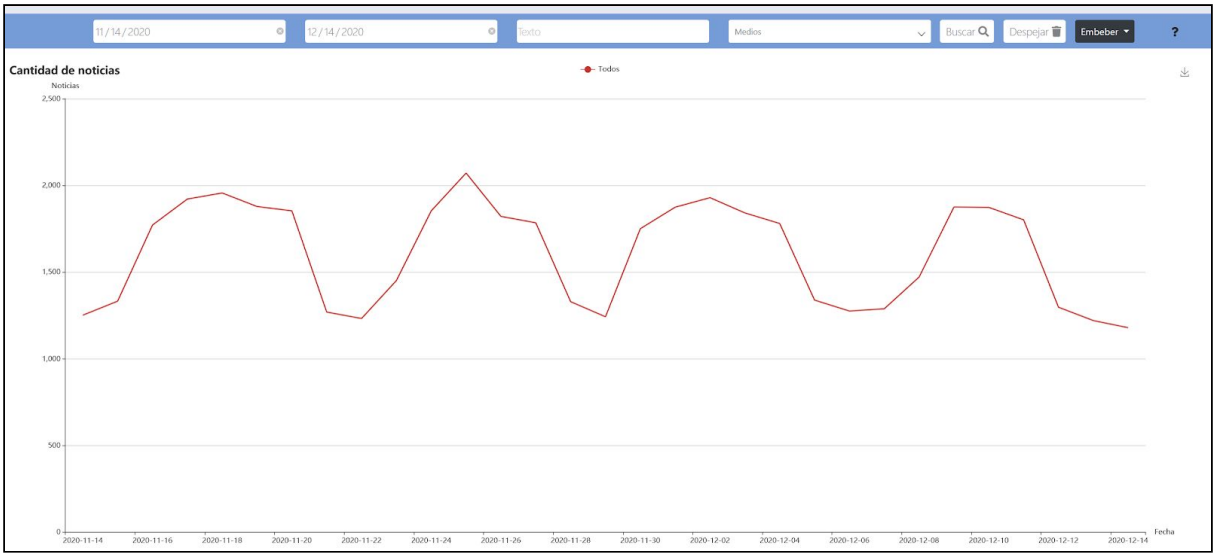

Figura 13. Cantidad de noticias, rango de días.

En cambio si se elige más de un medio, se muestra para cada fuente una línea con la cantidad de noticias de dicha fuente.
Se puede hacer clic sobre ellas o sobre la leyenda para ocultar/mostrar cada serie.

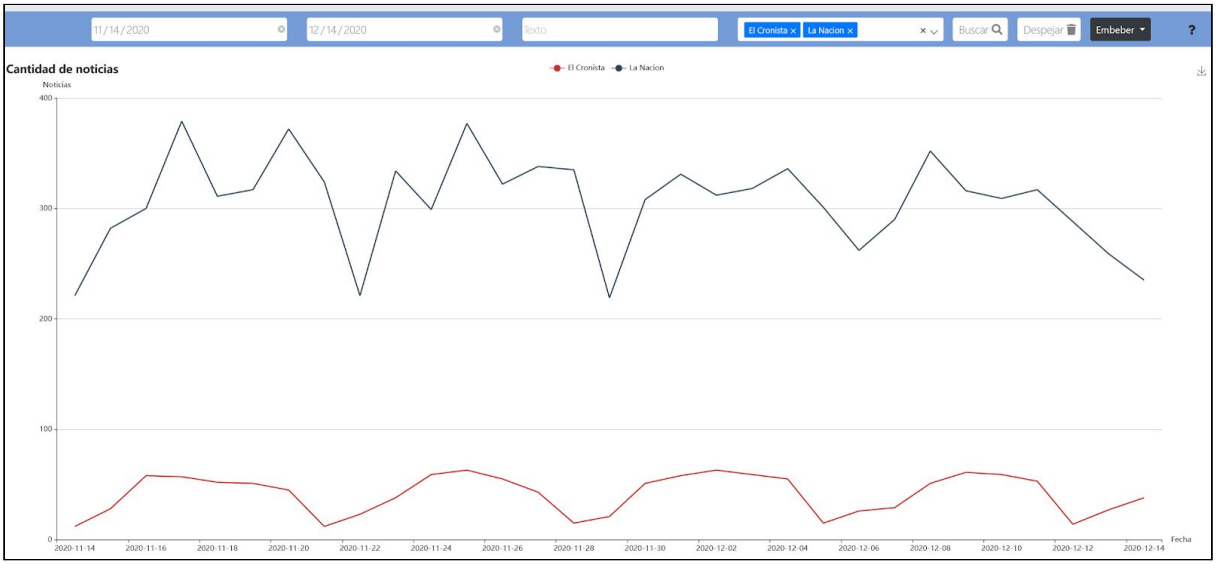

Figura 14. Cantidad de noticias para rango de días, y medios específicos.

Esta visualización fue creada con la librería Apache ECharts.

Tendencia

Aquí se muestra la tendencia que tuvo una palabra o varias (según la selección), para el periodo de fechas seleccionados. Para mostrar varias tendencias, las

palabras deben estar separadas por comas.

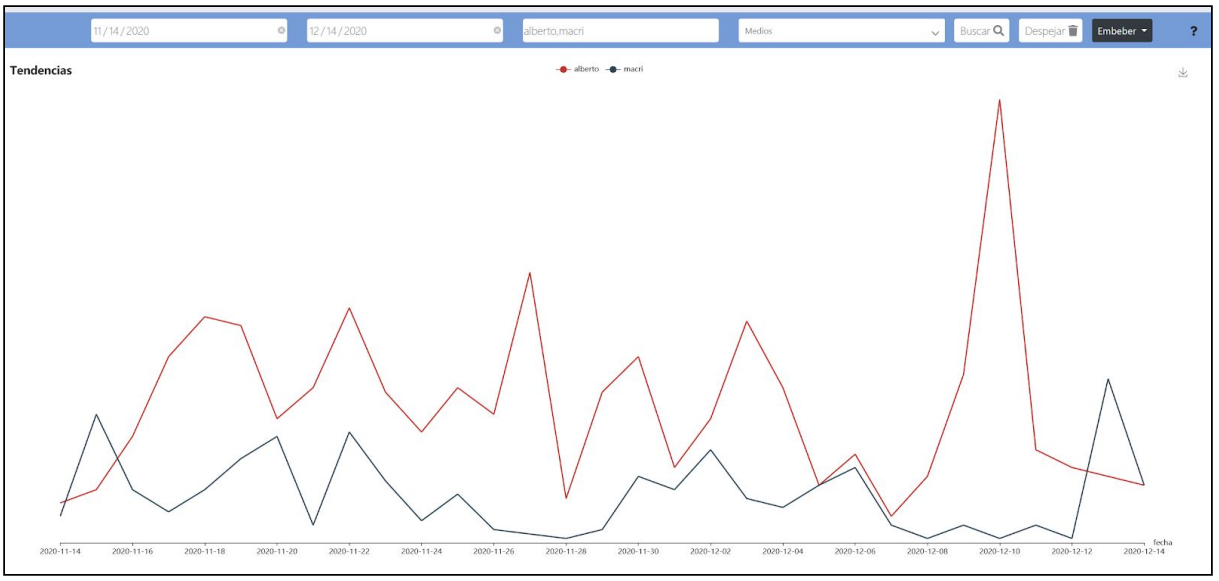

Figura 15. Tendencias, rango de fechas.

Si ningún período es elegido, por default muestra la tendencia de los últimos 15-30-90 días, pudiendo mostrarse una u otra con los botones de selección.

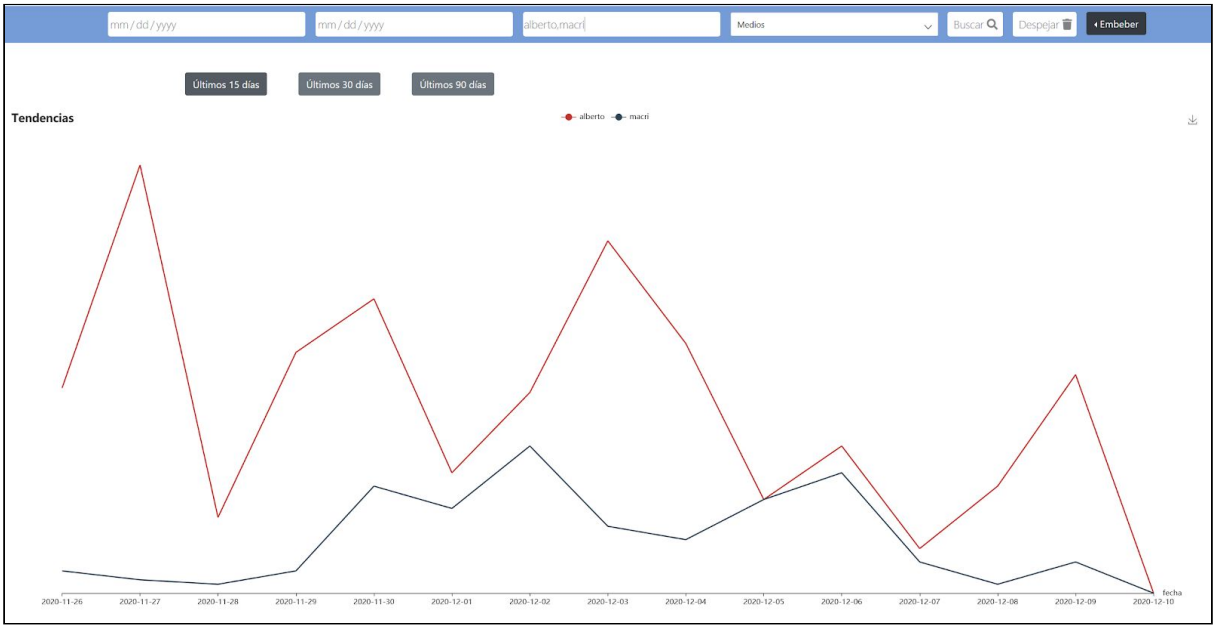

Figura 16. Tendencias, sin especificar rango de fechas.

Esta visualización fue creada con la librería Apache ECharts.

Árbol de palabras

El árbol de palabras permite visualizar las palabras que anteceden y preceden a la palabra en foco, la cual inicialmente es la palabra que se buscó en las noticias con el filtro de texto.
Con los controles se puede mostrar únicamente el árbol antecesor (flecha izq) o el árbol posterior (flecha derecha) o volver a la vista centrada (multiflecha).
Al hacer clic en las palabras es posible navegar el árbol. Esta visualización fue creada con la librería Google Charts.

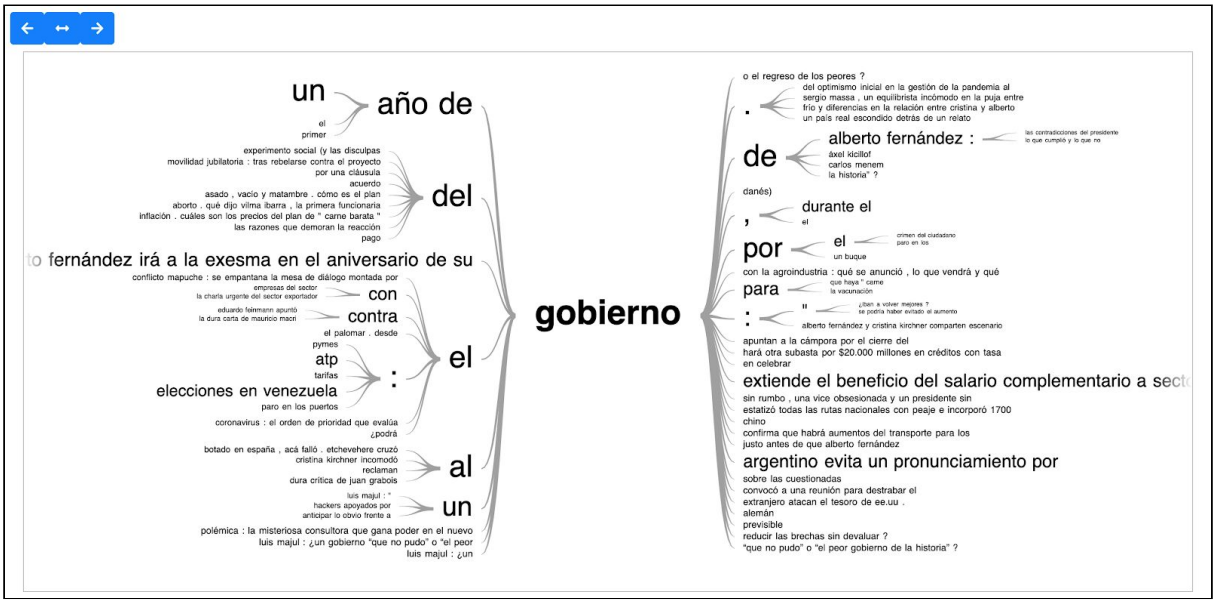

Figura 17. Árbol de palabras.

Calendario

Esta sección muestra un mapa de calor por cada medio seleccionado, donde el calor depende de la cantidad de noticias de la fecha dada. La escala de color es individual para cada mapa, variando entre la minima y maxima cantidad de noticias de dicho mapa. Esta visualización fue creada con la librería Apache ECharts.

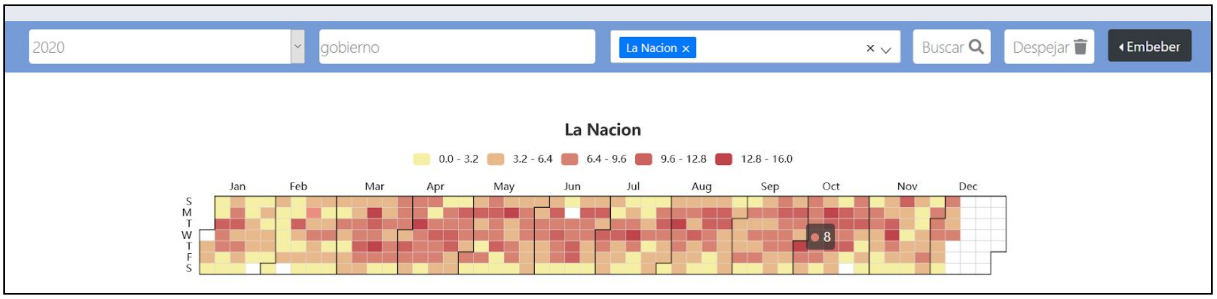

Figura 18. Mapa de calor para un solo medio.

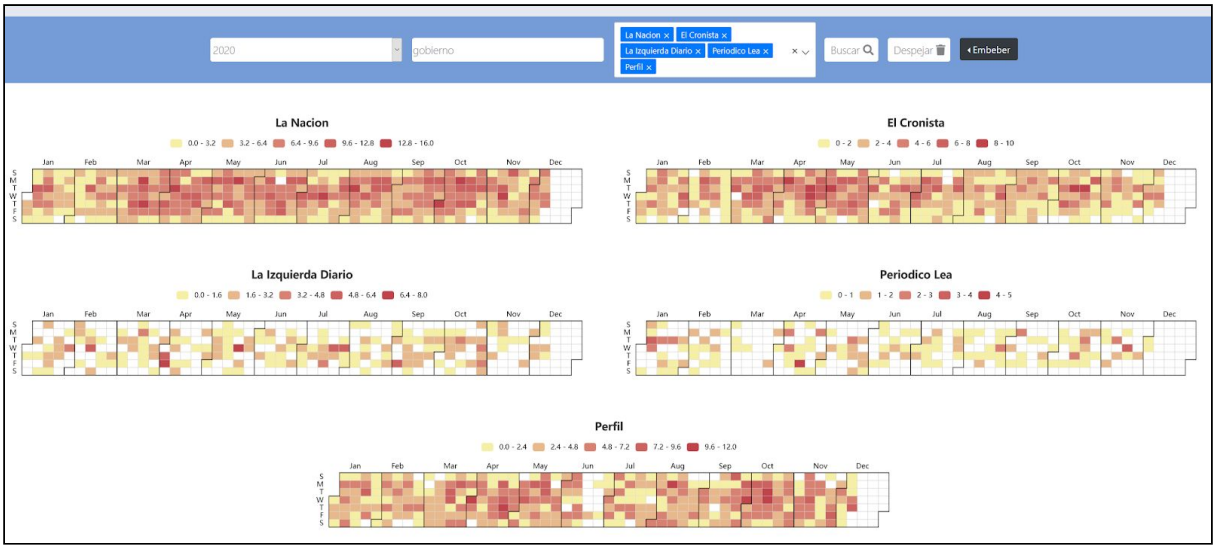

Figura 19. Mapa de calor para múltiples medios

Destacados

Esta sección toma las últimas 200 noticias que entran dentro de los filtros seleccionados (o las últimas 200 si ningún filtro fue seleccionado) y las analiza con la NERD API, obteniendo las entidades de dichas noticias. Selecciona las 5 que más se repiten de cada categoría (persona, ubicación, organización, misc) y las muestra. Al clickear sobre algún destacado, se expande la lista de noticias que contienen dicha entidad y además, un grafo de entidades-medios.

En el grafo se pueden visualizar las entidades y los medios que hablaron sobre ellos (dentro de las noticias analizadas), además la entidad seleccionada es enfocada y resaltada. por default solo se muestran las entidades del tipo de la entidad que se seleccionó. Si se selecciona una nueva, de una categoría distinta, el grafo se redibuja con la categoría deseada.

Puede seleccionarse manualmente la categoría a mostrar, si se elige mostrar todas, al seleccionar una entidad nueva, el grafo no se vuelve a redibujar, mostrando así todas a menos que se seleccione manualmente un subconjunto nuevamente.

Cada tipo de entidad posee un icono único en el grafo, al igual que cada medio es representado con su favicon.

El gráfico de nodos está hecho con la librería Vis-Network.

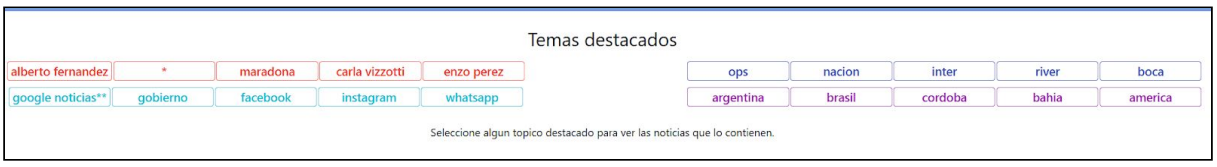

Figura 20. Entidades agrupadas por tipo.

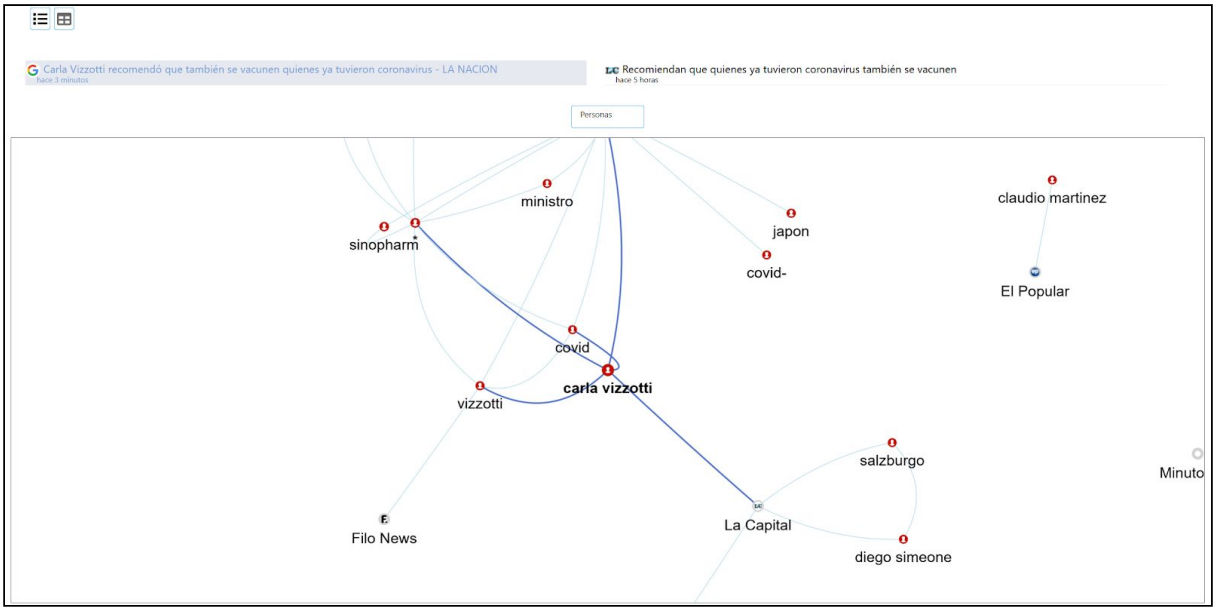

Figura 21. Entidad resaltada y sus aristas.

Línea de tiempo

La línea de tiempo permite visualizar cronológicamente las últimas noticias, mostrando su hora de publicación, y representando al medio que la publicó con su favicon. Al hacer *mouseover* sobre una noticia, se puede ver el contenido de la misma. Si se clickea, se abre en una nueva pestaña la noticia.
Mediante un input es posible alterar la cantidad de noticias a mostrar, el máximo posible es 1000, con los controles o con el *mouse* es posible hacer zoom, y desplazarse hacia los lados.
Esta visualización está hecha con la librería Vis-Network.

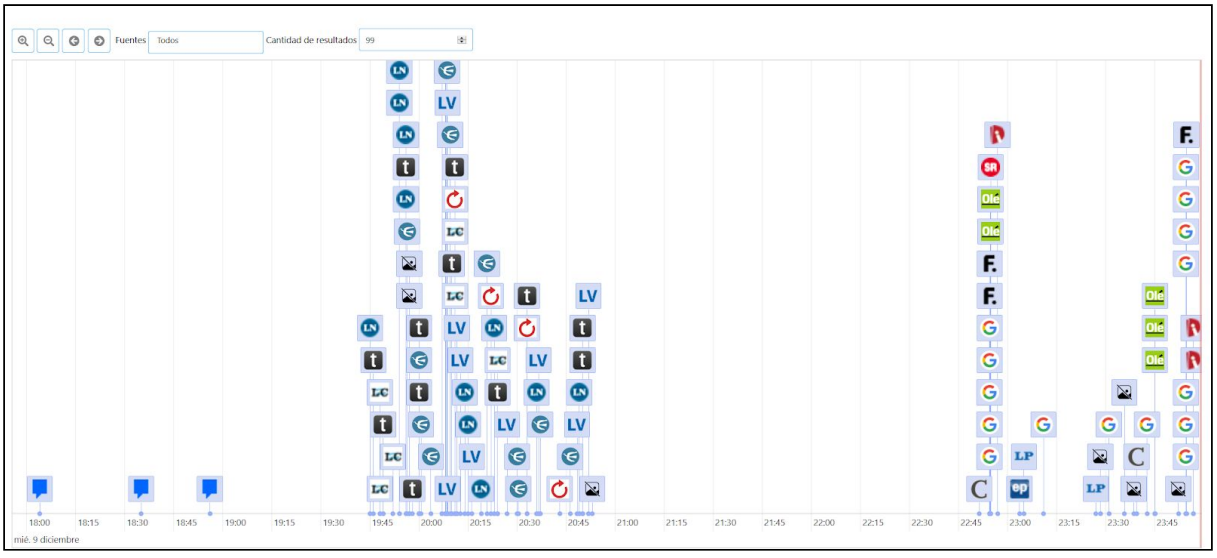

Figura 22. Línea de tiempo de noticias.

Prototipo

Las visualizaciones nuevas que se presentaron fueron prototipadas en Observable[4]. Se optó por realizarlas en este sitio porque permite el desarrollo de las visualizaciones de forma rápida, compartirlas y luego decidir si se implementa en el proyecto Angular. Los prototipos que se hicieron son los siguientes:

Noticias por provincia

Mapa de Argentina donde cada provincia está pintada de acuerdo a la cantidad de noticias que posee. Esta cantidad se obtiene mediante el *endpoint* `GET /cantidad-de-noticias`. Además, realizando clic sobre una de ellas, se lista las noticias relacionadas a la provincia seleccionada. Dichas noticias las retorna el *endpoint* `GET /busqueda`, que recibe como parametro el nombre de la provincia y luego se las muestra en pantalla en formato de *card*.
Observable url: https://observablehq.com/@matrivas/d3-mapa-provincias-argentina-news

[4] https://observablehq.com/

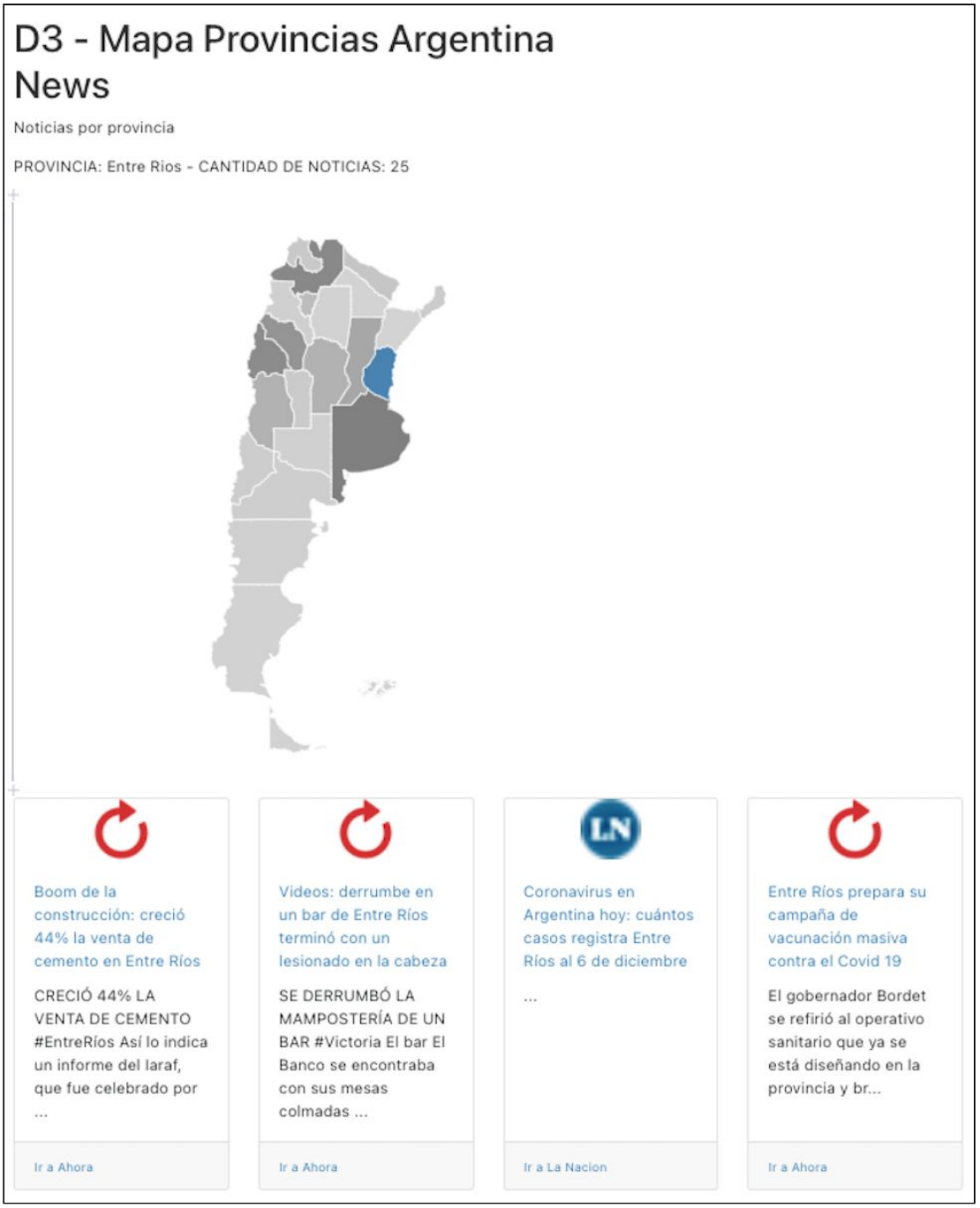

Figura 23. Se observa la provincia de Entre Ríos seleccionada y sus noticias relacionadas.

Noticias en formato carta

Se utilizó el *endpoint* `GET /busqueda` para obtener noticias de la última semana. Estas noticias se visualizan en forma de card mostrando el logo del medio, título y summary. Para esta visualización se utilizó la librería Bootstrap.
Observable url: https://observablehq.com/@matrivas/card-news

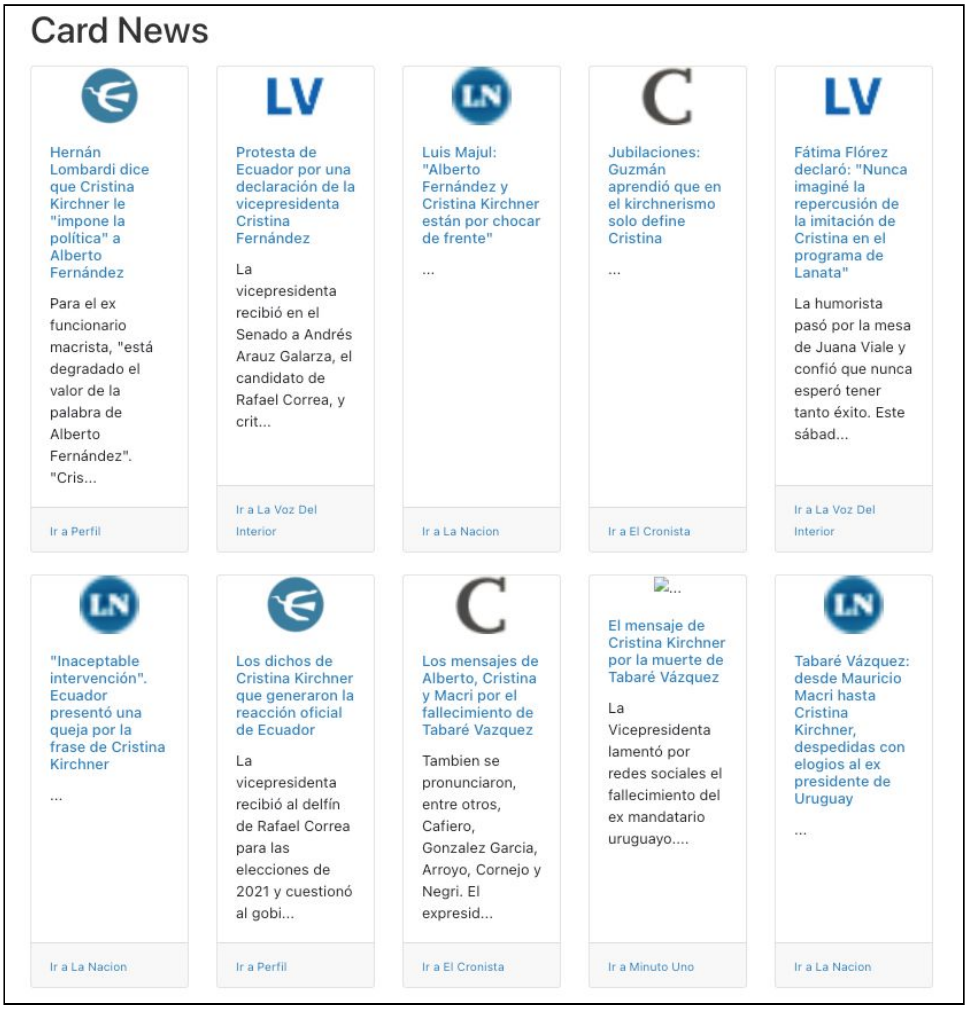

Figura 24. Noticias en formato tarjeta.

Línea de tiempo

El timeline se creó utilizando la librería Vis-Timeline. Para la obtención de las noticias se llamó al *endpoint* `GET /busqueda` pasando como parámetro un rango de fechas, palabra a buscar y medio. La respuesta obtenida se transformó de modo que pueda ser leída por la librería. Cada noticia posee un tooltip donde se muestra el título, summary y la fecha de publicación. Esta fecha se muestra respecto a la fecha y hora actual. Para lograr esto, se utilizó la librería moment.js.

Observable url: https://observablehq.com/@matrivas/timeline-news

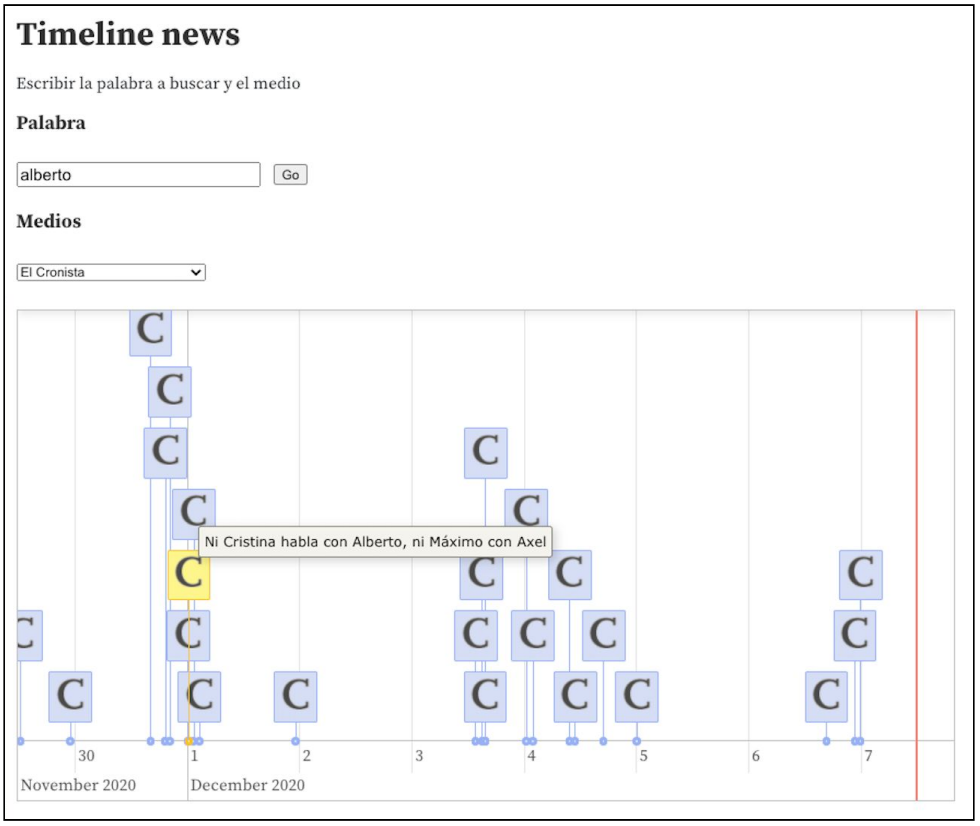

Figura 25. Línea de tiempo de noticias del medio El Cronista.

Grafo de entidades con vis-network

Se utilizó el *endpoint* `GET /nube-de-palabras` pasando como parámetros un rango de fechas y un medio. Una consulta por medio realizó debido a que la respuesta de retornar el servicio no hace la distinción. Luego, utilizando los datos obtenidos, se crea el grafo de tendencias utilizando la librería Vis-Network. El mismo muestra la relación entre las palabras que son tendencia y los medios que hacen referencia a las mismas.
Observable url: https://observablehq.com/@matrivas/vis-js-tendencia

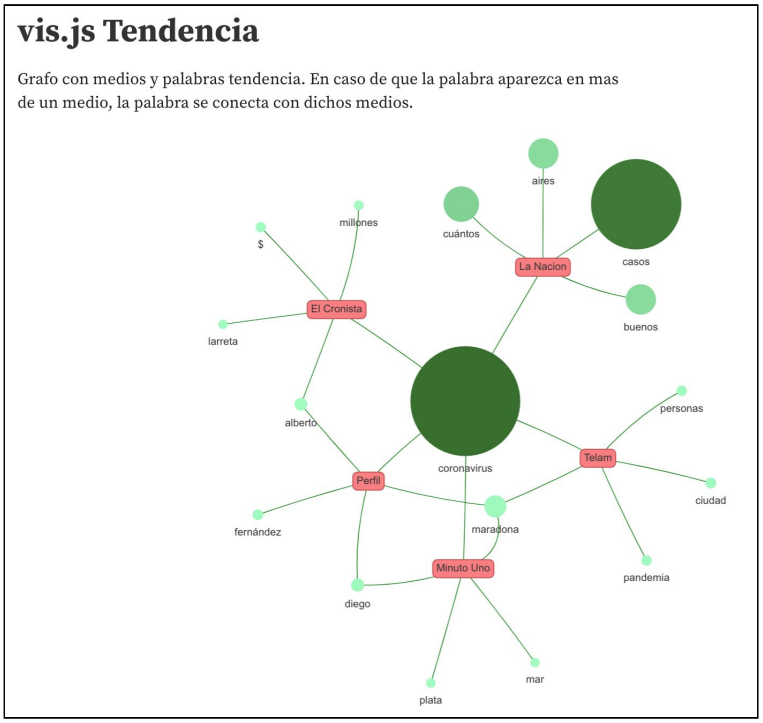

Figura 26. Grafo con nodos de palabras tendencia en círculo y nodos medios en rectángulo.

Grafo de entidades con ECharts

En este caso el grafo de tendencias se realizó utilizando la librería ECharts. De igual forma que el prototipo anterior, se utilizó el *endpoint* `GET /nube-de-palabras`.
Observable url: https://observablehq.com/@matrivas/echarts-grafo-de-tendencias

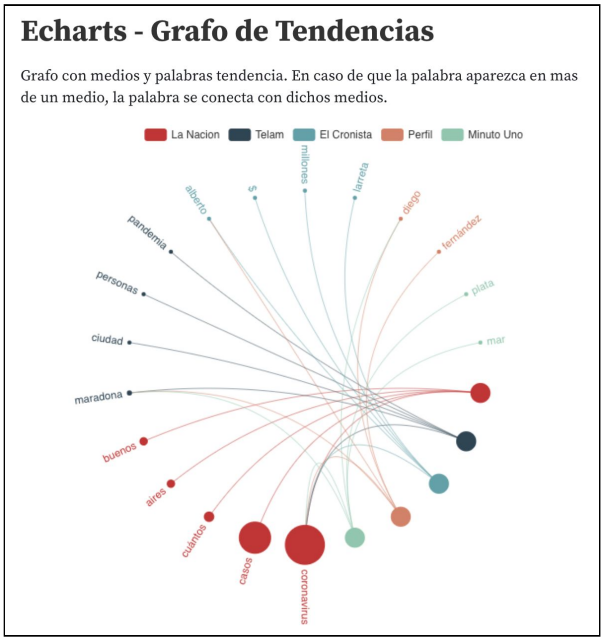

Figura 27. Grafo con palabras tendencia y conectadas con los medios en los que es tendencia.

Dashboard de noticias

Este prototipo consiste en el diseño de un dashboard de noticias donde se pretende seleccionar una palabra en el árbol de palabras y que esta acción lance una búsqueda de noticias, utilizando el *endpoint* `GET /busqueda`, relacionadas a dicha palabra y mostrarlas en una tabla. Además, se busca palabras relacionadas a la ya seleccionada utilizando el *endpoint* `GET /nube-de-palabras` y se visualizan en un gráfico de burbuja.

El objetivo de este prototipo fue buscar la interacción entre distintas visualizaciones y relacionar los datos que retornan distintos endpoints del servicio XDATA API.

Observable url: https://observablehq.com/@matrivas/news-dashboard

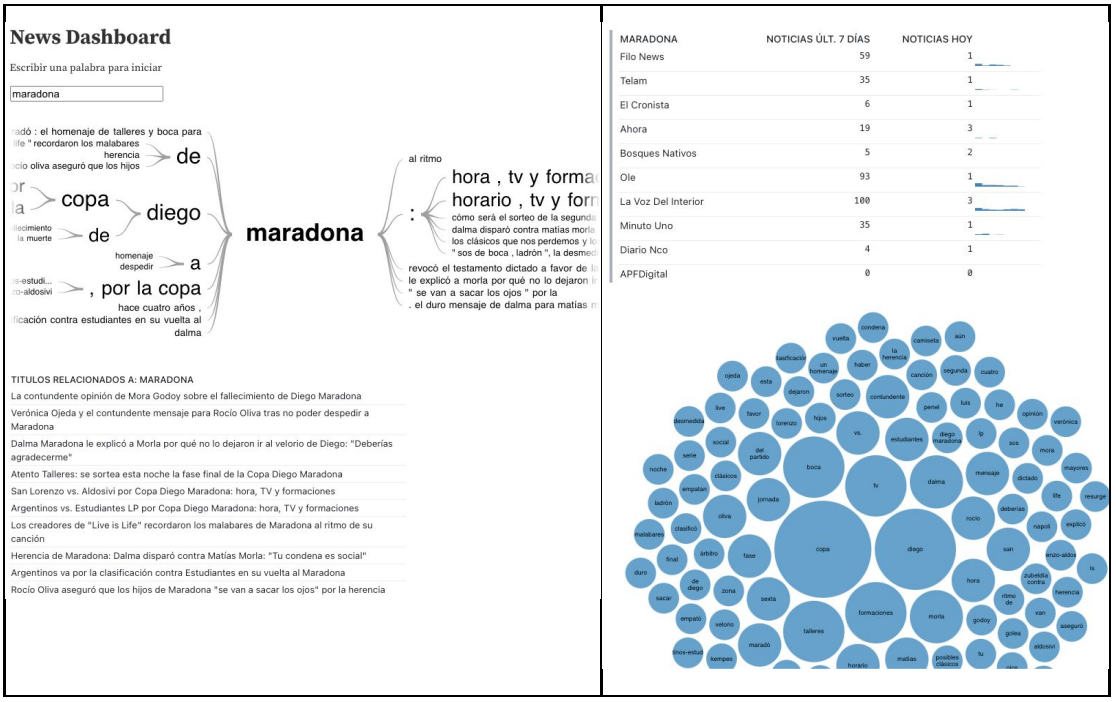

Figura 28. Dashboard con diferentes visualizaciones sobre la palabra "maradona".

Mapa de calor

El objetivo del mismo es mostrar la actividad de cada medio por día durante todo un año en formato calendario. Los datos se obtuvieron por medio del *endpoint* `GET /cantidad-de-noticias`.

Observable url: https://observablehq.com/@matrivas/echarts-heatmap

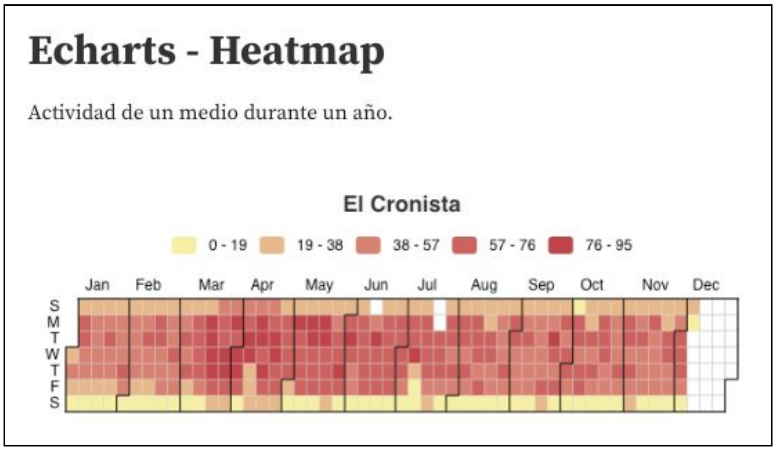

Figura 29. Mapa de calor sobre un medio en particular.

Cantidad de noticias por medio

El objetivo fue detallar la cantidad de noticias por semana y del día por cada medio. También se agregó el sparkline para visualizar la cantidad de noticias en

la semana.
Observable url: https://observablehq.com/@matrivas/noticias

Noticias

DÓLAR	NOTICIAS ÚLT. 7 DÍAS	NOTICIAS HOY
Filo News	11	2
Telam	6	1
El Cronista	18	1
Ahora	7	1
Bosques Nativos	1	1
Ole	1	1
La Voz Del Interior	3	1
Minuto Uno	8	1
Diario Nco	3	1
APFDigital	0	0

Figura 30. Aparición durante una semana de la palabra “dólar” sobre distintos medios.

Dashboard de noticias 2

Prototipo de dashboard de noticias el cual permite seleccionar un medio y se muestra las palabras tendencias, titulos del dia y el heatmap.
Nuevamente, se buscó que con una sola selección (medio) se muestre información relevante visualizada de distintas maneras.
Observable url: https://observablehq.com/@matrivas/news-dashboard-2

Figura 31. Dashboard con visualizaciones sobre un medio en particular.

Burbuja de palabras

Buscando otra forma de mostrar la nube de palabras, se prototipó un gráfico de burbuja que ofrece la librería D3. Las palabras se obtienen de la respuesta del *endpoint* `GET /nube-de-palabras`. Estas palabras se muestran dentro de un círculo donde su tamaño depende de la ponderación que la misma tenga.
Observable url: https://observablehq.com/@matrivas/d3-bubble-chart

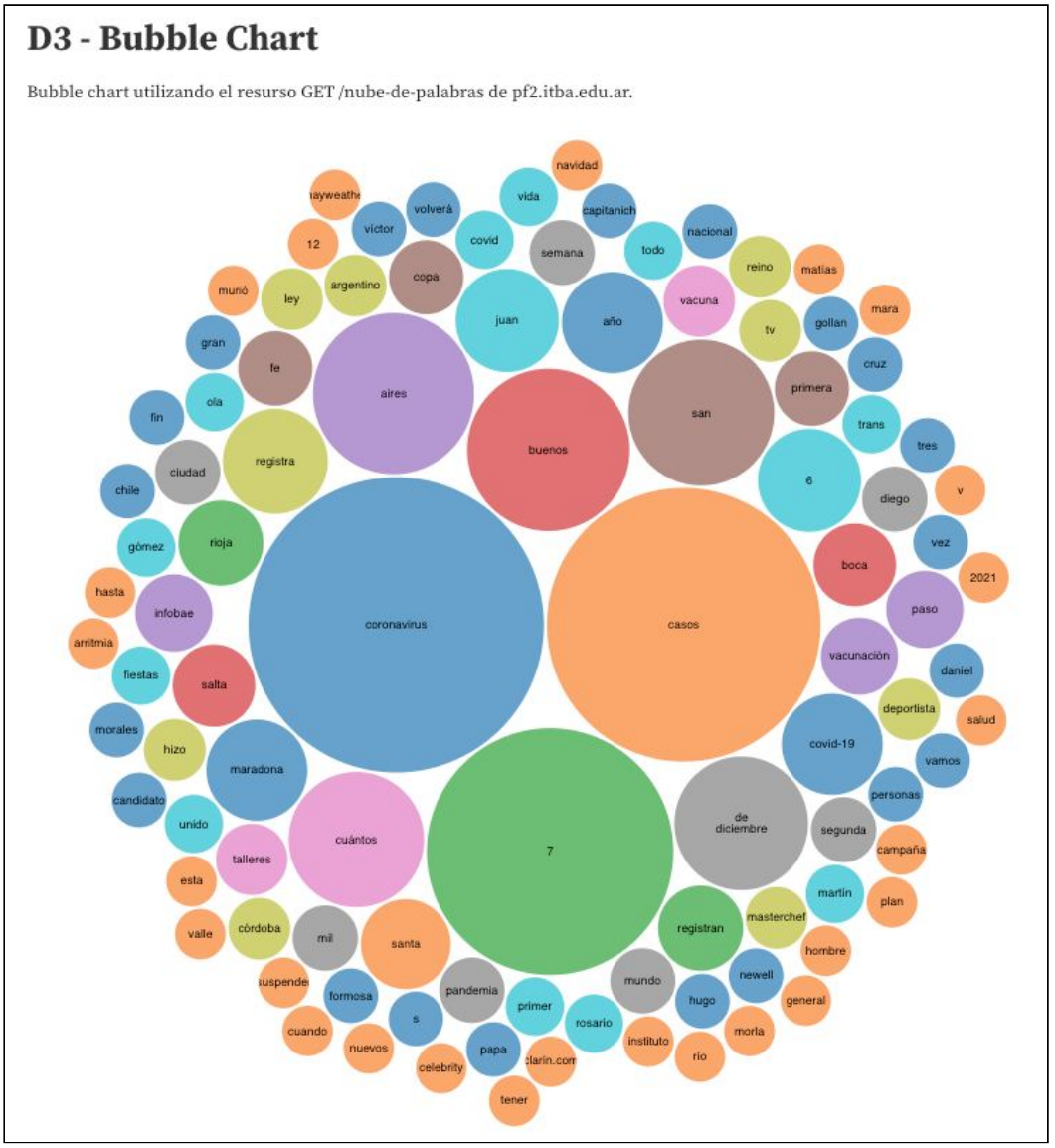

Figura 32. Palabra en círculos donde su tamaño corresponde a la cantidad de repeticiones.

Embeber visualización

Se generó la porción de código con la consulta al sitio y se lo importó a Observable.

Observable url: https://observablehq.com/@matrivas/embebed-xdata-2-0

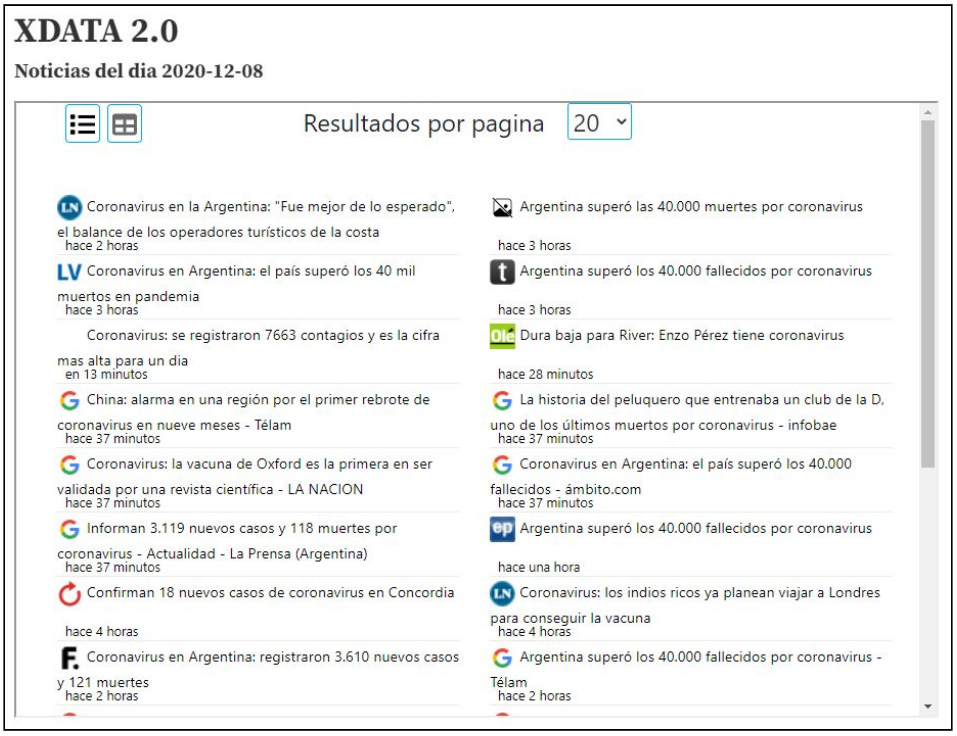

Figura 33. Visualización de títulos embebido en Observable.

Conclusiones

Tras el trabajo realizado, podemos concluir que existen muchas herramientas que permiten mejorar la visualización y exploración de noticias. Los sitios actuales podrían crear funcionalidades que mejoren la experiencia del usuario.

Pocos son los sitios donde sus visualizaciones son destacadas, ya sea por la creatividad con la que se las presenta como también la sencillez de poder mostrar datos y que los mismos sean fáciles de entender para el usuario promedio.

Lamentablemente dicha mejora podría llegar a implicar actualización de sus tecnologías, capacitar al personal o disponer de un equipo especializado en el análisis de datos y creación de visualizaciones, lo cual conlleva un costo económico.

Utilizar entidades nombradas dio resultados satisfactorios y abre la puerta a nuevas maneras de ver, buscar y relacionar las noticias, permitiendo asociar dos o más palabras a un solo término y dando más valor a dichos términos que a una palabra ordinaria.

Trabajo futuro

Entidades

Una vez que se tengan las entidades de cada noticia en la base de datos, se pueden agregar visualizaciones que utilicen las entidades de una mayor cantidad de noticias a la vez, ya que como actualmente se piden vía API, utilizar un número mayor a 200 generaba demoras en las visualizaciones.

Explotar más visualizaciones con las entidades

La gran limitante para crear visualizaciones con entidades fue la falta de las mismas en la base de datos. De tenerse estas mismas guardadas podría crearse más visualizaciones con búsquedas y filtros más avanzados sobre las entidades.

Apéndice

Para el proyecto se utilizaron las siguientes librerías y frameworks:
Angular[5] - Framework para desarrollo web.
D3.js[6] - Librería de javascript para desarrollar gráficos y visualizaciones.
Apache[7] ECharts - Librería de javascript para desarrollar gráficos y visualizaciones.
Vis-Network[8] - Librería de javascript para desarrollar gráficos y visualizaciones.
Observable[9] - Sitio para explorar, prototipar, visualizar y analizar información.
Google Charts[10] - Librería de javascript para desarrollar gráficos y visualizaciones.

Repositorio del proyecto
https://bitbucket.org/itba/pf-vis-news-nlp-2020

Hosting del proyecto
https://news-ui.herokuapp.com

Tecnologías y herramientas utilizadas para la comunicación del equipo de trabajo
https://trello.com
https://app.slack.com

NERD API
http://nerd.it.itba.edu.ar/
http://nerd.it.itba.edu.ar/api/doc

XDATA
http://pf2.it.itba.edu.ar/

[5] https://angular.io/features
[6] https://d3js.org/
[7] https://echarts.apache.org/en/index.html
[8] https://visjs.org
[9] https://observablehq.com/
[10] https://developers.google.com/chart