XDATA

Proyecto Final 2017

Integrantes:

- Fernández, Pablo
- Lata, Andrea

Tutor:

- Aizemberg, Diego Ariel

Índice

Propuesta

El objetivo del Proyecto Final propuesto es mejorar un portal web, focalizado en la visualización de noticias. Dentro del portal se puede acceder a diferentes visualizaciones, aplicar filtros y agrupar la información para crear la visualización más conveniente para el usuario. Dicho proyecto tomará como base uno realizado previamente por el profesor Ariel Aizemberg.

Los principales puntos a desarrollar son:

1. En primer lugar se deben agregar nuevos medios de información para generar una base de datos. Se espera que la información provenga de diversas fuentes, en consecuencia se debe implementar un parser para procesar la información y almacenarla de forma que pueda ser reutilizada fácilmente. Actualmente hay un parser en python de RSS pero habría que analizar si es la mejor alternativa, en el caso que no lo sea realizar una mejora.
2. Diseñar e implementar una DATA API para consultar la información almacenada en una base de datos PostgreSQL. Se busca mejorar la performance y la eficiencia, ya que en este momento las consultas se tornan muy lentas.
3. Generar distintas visualizaciones en las que se puedan apreciar claramente distintas características de los datos obtenidos. Las mismas deben transmitir clara y efectivamente la información. En este punto se pondrá en práctica los conocimientos aprendidos en la materia “visualización de la información”

Metodología de trabajo

El proyecto se dividió en 3 etapas con el fin de ir presentando parcialmente funcionalidades nuevas dando visibilidad de los avances. Se tuvo en cuenta que una de las prioridades era la recolección de noticias y su almacenamiento para evitar la pérdida de datos y tener una base de datos más grandes para futuras visualizaciones.

Las etapas son las siguientes:

Relevamiento

La etapa inicial de relevamiento consistió de un análisis funcional de la aplicación existente para conocerla en profundidad y evaluar la experiencia de usuario. A continuación se evaluaron las tecnologías a utilizar para desarrollar la solución y se discutió la propuesta con el tutor.

Desarrollo de versiones

La funcionalidad del proyecto fue dividida en versiones más pequeñas que fueron entregadas y puestas en producción[1] regularmente con el fin de ir iterando sobre ellas. De esta manera desde el comienzo se pudo recolectar noticias, realizar pequeñas consultas y eventualmente contar con visualizaciones más complejas.

Mantenimiento y nuevas funcionalidades

La última etapa consistió en corregir los errores que se encontraron luego de los desarrollos y mejorar la experiencia del usuario evaluando globalmente el proyecto. A partir de esto se optó por incluir algunas funcionalidades nuevas.

[1] El departamento XDATA del Instituto Tecnológico de Buenos Aires ofreció una computadora para instalar la aplicación y tenerla en ejecución las 24 horas del día.

Análisis funcional

El análisis funcional se hizo sobre la aplicación original desde la perspectiva del usuario final y el administrador. Se entiende que el usuario final es el que utiliza la aplicación para informarse mientras que el administrador es aquel que la mantiene actualizada e incorpora nuevos medios.

Desde el punto de vista del usuario final

Sección títulos

Dicha sección es la principal (o "home") lista los títulos de las noticias publicadas por todos los medios el día de la fecha ordenadas alfabéticamente. Inicialmente mostraba la repercusión[2] en facebook y twitter de cada noticia, sin embargo, al momento del análisis no funcionaba porque dichas redes sociales dieron de baja ese servicio.

Las noticias se paginan en grupos de 50 permitiendo aplicar todos los filtros de búsqueda[3]. Al hacer foco sobre una noticia se abre un acordeón y muestra el resumen de la misma. Dentro de la sección desplegada se puede clickear en el link "Ver la noticia completa" que redirige a la original.

La imagen 1 muestra una captura de pantalla de la sección.

[2] Cantidad de veces que se compartió dicha noticia en una red social.
[3] Los filtros incluyen: fecha de publicación, el medio de origen y palabras dentro del título.

Imagen 1: Sección títulos

Sección Nube de palabras

La sección contiene una visualización que lista las palabras contenidas en los títulos de la noticias. Se ordenan las palabras de mayor a menor según la cantidad de veces que aparecen en los títulos. Las palabras van disminuyendo su tamaño a medida que la cantidad de repeticiones disminuye. Se pueden aplicar todos los filtros. Al clickear sobre una palabra uno es redirigido a la sección de títulos con el filtro de palabra aplicado.

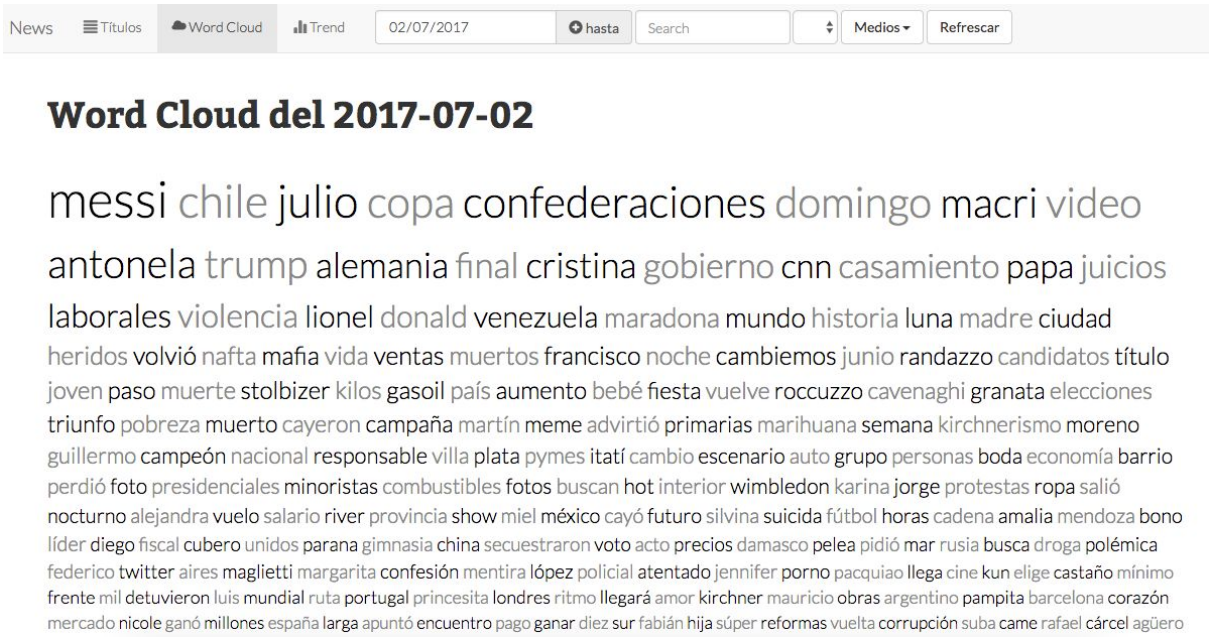


Imagen 2: Sección WordCloud

Sección Tendencias

Permite ver el impacto de una o más palabras en los títulos de las noticias en un período de tiempo. La visualización utilizada es un gráfico de línea que muestra en el eje X la fecha y en el eje Y la cantidad de veces que apareció la palabra en los títulos de ese dia.

Ofrece como opciones rápidas para filtrar la fecha: últimos 7 días, últimos 15 días, último mes, últimos 2 meses y últimos 3 meses. Si no se especifica ninguna fecha muestra los últimos 7 días. Es posible aplicar los mismos filtros que existen en las otras secciones.

Cuando uno ingresa a la sección te indica con el siguiente mensaje como usarla: “Buscar palabra: Tiene que buscar algunas palabras para ver el gráfico. Las palabras deben estar separadas por comas. Por ejemplo : "buenos aires, argentina" va a buscar "buenos aires" y "argentina".”

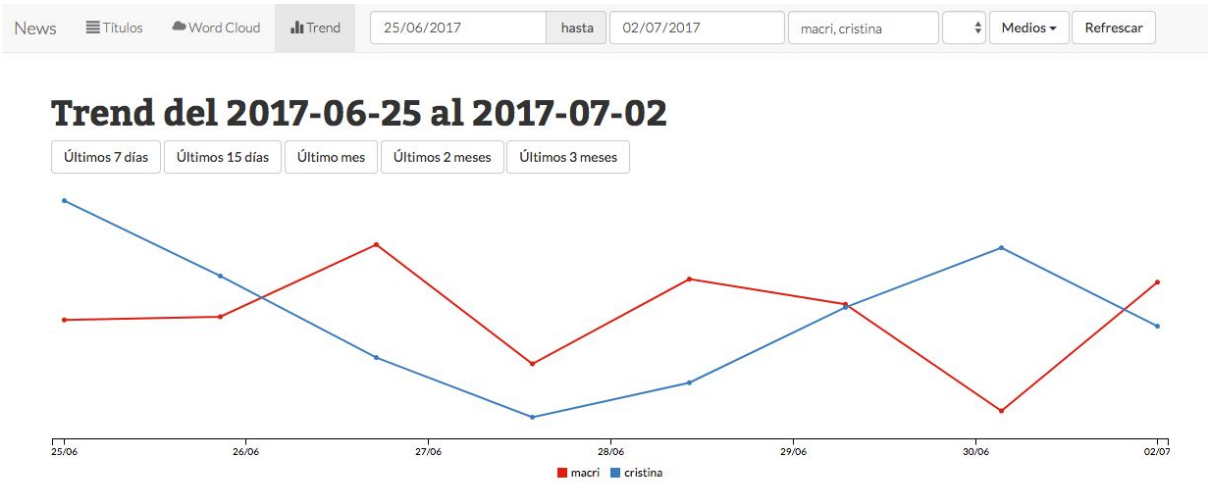


Imagen 3: Tendencias

Filtros

La aplicación tiene un encabezado en todas las secciones con un formulario para realizar búsquedas. Dichos filtros se conservan al pasar de una sección a otra manteniendo así el contexto.

Fecha

Filtra las noticias que se publicaron en un periodo de tiempo. Las opciones son: desde-hasta y desde.El campo desde tiene como valor predeterminado el día actual.

Medios

Muestra la lista de medios en un menú desplegable y permite filtrar por uno o más de ellos.

Texto

Busca los titulares de las noticias que contienen en el texto ingresado.

Desde el punto de vista del administrador

Administración de medios

Las noticias son obtenidas a través de los RSS[4]. Para su configuración el usuario administrador utiliza el archivo *feeds_list.py*. En este archivo debe incluir las urls de los mismos y una serie de campos de configuración que indican cómo interpretar el RSS. Por ejemplo, si incluye un resumen de noticia, en qué campo se encuentra la fecha de publicación, entre otros. A partir de este archivo el parser automáticamente buscará las noticias.

El formato para agregar los rss en el archivo es el siguiente:

```
feeds.append(['url_source',title, link, summary, date, category, content, media])
```

[4] https://es.wikipedia.org/wiki/RSS : **RSS** son las siglas de **Really Simple Syndication**, un formato XML para sindicar o compartir contenido en la web. Se utiliza para difundir información actualizada frecuentemente a usuarios que se han suscrito a la fuente de contenidos.

Elección de tecnología

Diseño de la aplicación

El interfaz visual y la API de acceso a los datos de la aplicación original están desarrollados con PHP y almacena la información en una base de datos PostgreSQL. Además cuenta con múltiples scripts en python para parsear los RSS. Estos programas realizan un solicitud sobre la url, interpretan la respuesta y guardan los datos en la base. Existe un programa genérico para parsear cualquier medio y programas individuales por medios específicos.

Teniendo en cuenta el diseño inicial se decidió utilizar un lenguaje y framework de desarrollo web más actual que permita crear una aplicación más robusta, mantenible y escalable. Además que permita un desarrollo iterativo y veloz ajustándose así a la metodología de trabajo. También se tuvo en cuenta los conocimientos del equipo y del tutor quién será el responsable de mantener la aplicación en el futuro.

Por estas razones el framework web y lenguaje elegidos fueron Django[5] y Python. Django es un framework web programado en python que ofrece la posibilidad de crear aplicaciones rápida y sencillamente. Se ajusta a las necesidades del proyecto ya que permite tener una aplicación en producción sin mayor esfuerzo e ir incorporando funcionalidades nuevas progresivamente. Cuenta con una documentación[6] muy completa y clara además de tener una comunidad activa. La versión elegida fue la 1.8 siendo la más nueva al momento con soporte a largo plazo (Long term support). Django ofrece un ORM[7] que mapea a base de datos los modelos definidos. Si bien no tiene una buena performance el mismo no fue utilizado para procesamiento de datos que requieren una alta performance.

Una de las ventajas de Django es el ABM que viene incorporado. Esto permitió pasar todo el manejo de medios y RSS que se hacían desde un archivo de texto a una interfaz visual más cómoda. Dando más flexibilidad al administrador a la hora de incorporar nuevos medios, testear su correcto funcionamiento y eliminarlos, cumpliendo así con el primer requerimiento del proyecto.

[5] https://www.djangoproject.com/
[6] https://docs.djangoproject.com/en/1.8/
[7] Mapeo de objeto-relacional

El consumo de RSS se lleva a cabo una vez por hora a través de un comando customizado de Django que se ejecuta con un cron de Linux. El parser de RSS se creó a partir del script genérico de la aplicación inicial con múltiples correcciones en el código.

Modelado del problema

Elección de base de datos

Las entidades existentes en la aplicación antigua son medio y noticias.

La cantidad de medios online en la Argentina que ofrecen RSS ronda el orden de las centenas. El set de datos más importante que se debe manejar es el de noticias el cual crece diariamente a una razón de 2,500 noticias al momento del análisis. Se estima que este número incrementará al incorporar nuevos medios y arreglar otros mal cargados en la aplicación. Sobre este conjunto de datos se espera hacer consultas complejas para realizar visualizaciones de interés. Por lo tanto se evaluó si era más conveniente utilizar una base de datos relacional, no relacional o ambas.

Inicialmente se consideró que la mejor alternativa era tener dos bases de datos, una relacional y la otra no relacional. La relacional manejaría las entidades de Medios y Usuarios mientras que la no relacional guardaría las entidades de Noticias. Cassandra y MongoDB fueron las bases de datos no relacionales consideradas. Cassandra fue descartada por no ofrecer la posibilidad de buscar strings con la opción “like” como otras bases. Luego se realizaron pruebas de concepto con MongoDB y PostgreSQL con un set de datos de IMDB[8] de 4 millones de registros para determinar si se justificaba a nivel performance tener una o ambas. Los resultados mostraron una gran diferencia en cuanto al rendimiento cuando se realizaba una consulta sin condiciones a favor de MongoDB pero a la hora de filtrar y ordenar estos registros sus tiempos de respuesta eran similares. Por lo tanto se decidió comenzar con una única base de datos PostgreSQL teniendo en cuenta la baja performance de consultas complejas.

Diseño de base de datos

Se definieron tres modelos que serían mapeados a base de datos. Estos son: Noticia (News), Medio (Source) y rss (SourceRSS).

[8] IMDB es un sitio web con información de películas y ofrece un dataset.

El modelo News contiene la información sobre una noticia: título, url, categoría, resumen y una relación al medio que la generó. No existe en la base de datos dos noticias con igual título, url, categoría y medio.

El modelo Source representa un medio con su nombre, url y favicon. También cuenta con un campo “activo” que determina si el sistema debe buscar o no noticias de este diario en particular. Esto permite dejar de buscar noticias de un diario que ya no tiene RSS pero tuvo en algún momento.

El modelo SourceRSS por su parte corresponde a cada RSS que ofrece un medio dado. Incluye la url al rss, una relación hacia su medio y un campo de fecha con la última actualización. Este último campo es utilizado para determinar desde el ABM si la RSS está funcionando o no.

Existe el modelo de usuario (User) en la base de datos para acceder al ABM pero es propio de Django. Este modelo define los usuarios que pueden ingresar al ABM y que modelos pueden modificar.

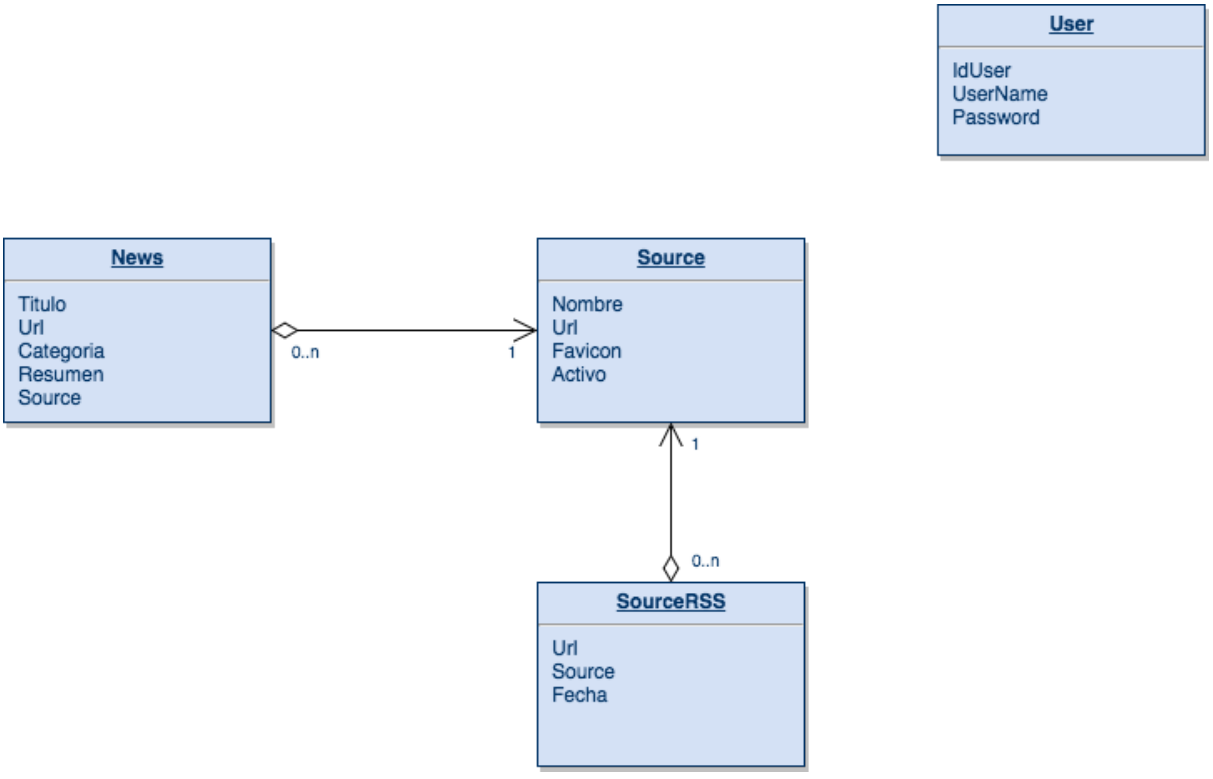


Imagen 4: Diagrama UML

Primera solución planteada

En base a los requerimientos la solución incluye:

1. Data API y performance

   Se construirá desde cero la aplicación web utilizando el framework Django. Permite definir fácilmente un API donde se expondrá los resultados de una búsqueda por medio de un JSON. La base de datos será PostgreSQL de acuerdo al análisis realizado.

2. ABM de medios y RSS.

   Utilizar el ABM provisto por Django para estas dos entidades. Además incluir la posibilidad de testear un RSS y notificar si no está funcionando. Realizar un rediseño del parser de RSS para evitar que se trabe como ocurre en el sistema viejo.

3. Front-end

   Dado que se propone utilizar Django para el backend sería conveniente utilizarlo para servir el contenido de la interfaz visual aprovechando al máximo las funcionalidades de este framework. Para la parte de visualizaciones se usará d3.js y c3.js. Además de un template que mejore la experiencia del usuario.

4. Agregar nuevas visualizaciones

Otras mejoras:

- En la sección de noticias agregar en un tooltip con el resumen y al clickearla redirigir a su url.
- En la misma sección mostrar el favicon correspondiente a cada diario junto con el título de la noticia.

Desarrollo de versiones

El desarrollo fue dividido en versiones que se le fueron presentando al tutor en reuniones cada 2 o 3 semanas. Estas presentaciones ayudaron a detectar algunos bugs y a ajustar funcionalidades. A continuación se detalla la última versión presentada previa al comienzo del período de mantenimiento.

ABM de medios y RSS

El ABM es una sección protegida del sitio donde se ingresa con un usuario administrador. Allí es posible acceder al listado de fuentes, agregar nuevas y modificar las existentes. Es posible ver el estado de cada medio para identificar si alguno no está obteniendo noticias como es debido. A su vez existe la opción de simular la obtención de datos de un medio sin guardarlo en la base de datos con el objetivo de probar fácilmente un medio en particular.

Imagen 5: ABM de Medios

Sección títulos

Lista los títulos de las noticias con el favicon del medio correspondiente ordenados por la fecha de publicación, las noticias más recientes aparecen primero. Las noticias se paginan en grupos de 20 permitiendo aplicar todos los filtros de búsqueda.

Al clickear un título se redirige a la noticia original. Es posible ver la descripción de la noticia si se deja el mouse sobre el link.

Imagen 6: Sección títulos

Sección nube de palabras

Ofrece visualizar las 200 palabras más influyentes en los títulos de las noticias según la búsqueda realizada. Si no se aplica ningún criterio de búsqueda se muestran las palabras más mencionadas del día. La sección cuenta con dos visualizaciones: ordenada (*imagen 1*) y desordenada (*imagen 2*). La visualización ordenada muestra el listado de palabras ordenadas de mayor a menor según la cantidad de veces que aparecen en los títulos. La desordenada muestra las palabras ubicadas de forma vertical y horizontal aleatoriamente y el tamaño de cada palabra crece con su frecuencia de aparición. Al hacer

foco sobre una palabra uno es redirigido a la sección de títulos con el filtro de palabra aplicado.

Imagen 7: Modo ordenado

Imagen 8: Modo desordenado

Sección cantidad de noticias

Muestra la cantidad de noticias publicadas según la búsqueda realizada. Si se busca en un rango de fechas, se ve una visualización que muestra cómo varía la cantidad de noticias a lo largo de ese periodo. Para realizar esta visualización se utilizó c3.js

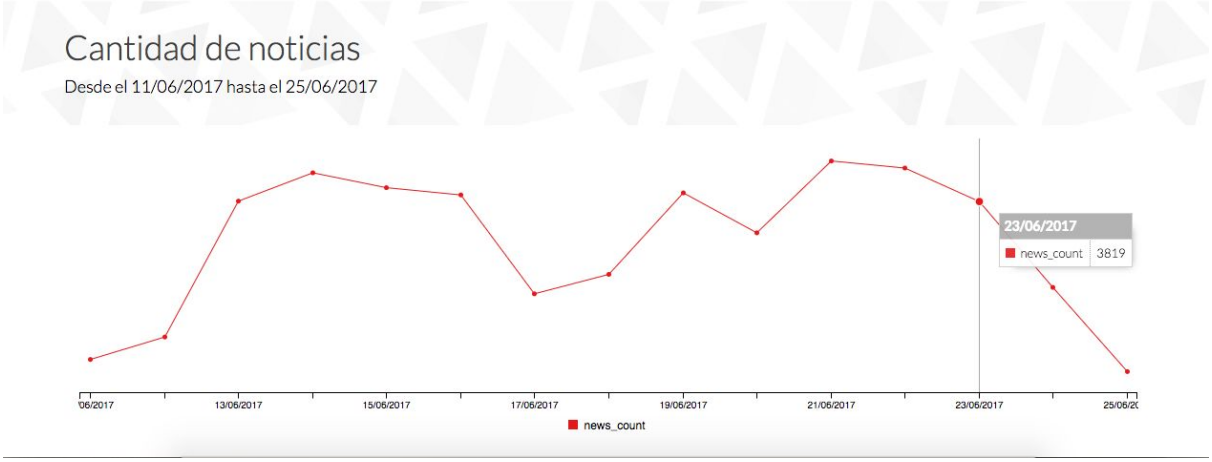


Imagen 9: Cantidad de noticias en un periodo de tiempo

Sección Tendencias

Muestra la cantidad de veces que se repite una o más palabras en los títulos de las noticias en un periodo de tiempo. Se utilizó la misma visualización que en su versión original. Se ofrece como opciones rápidas para filtrar fecha: últimos 7 días, últimos 15 días, últimos 30 días.

Cuando uno ingresa a la sección te indica con el siguiente mensaje como usarla: “Busca una palabra: Cuando ingreses una palabra podrás ver la visualización. La idea es ver la influencia que tuvo esa palabra en el periodo de tiempo seleccionado. Se puede ingresar más de una palabra, en ese caso deben estar separadas por **comas**. Por ejemplo : "buenos aires, argentina" va a buscar "buenos aires" y "argentina".”

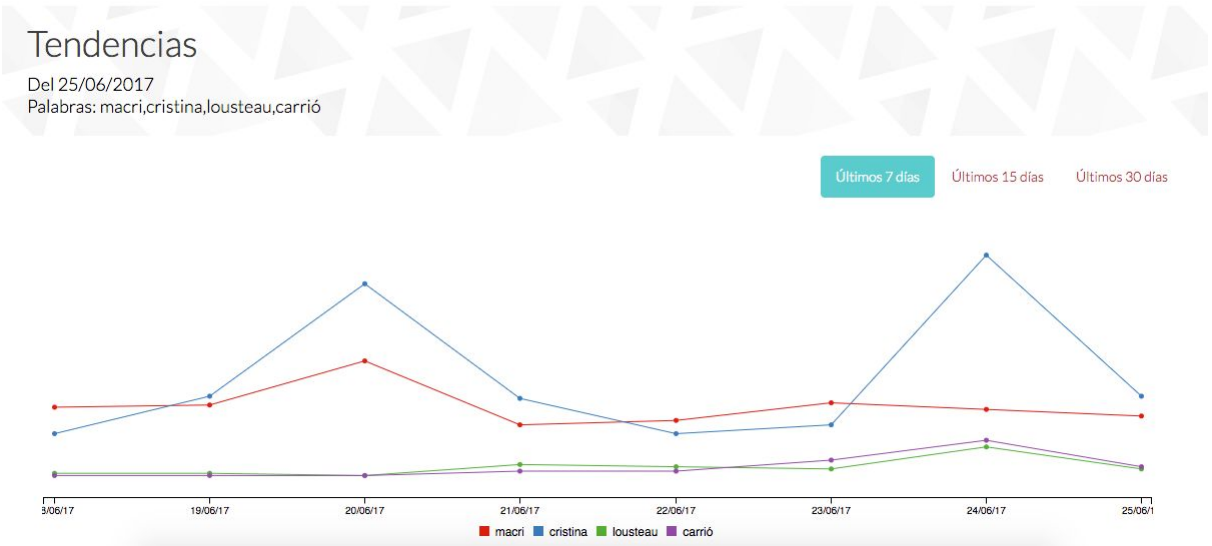


Imagen 10: Tendencias

API

Expone los conjuntos de datos utilizados en las diferentes secciones en un JSON. Se utiliza el mismo queryString que en el frontEnd para incluir filtros. Para más información consultar la documentación de la API[9] que detalla el funcionamiento de la misma.

El sistema tiene configurado CORS (Cross-origin resource sharing) para permitir que sea consultado desde un dominio distinto al del servidor. Actualmente no existe una lista de dominios permitidos a consultar si no que cualquiera puede consultar libremente. Es posible configurar una lista de dominios habilitados en el archivo **settings.py** del proyecto.

[9] http://pf2.it.itba.edu.ar/api/documentacion

Mantenimiento y nuevas funcionalidades

Finalizado el desarrollo de la aplicación hubo un período de mantenimiento en el cual se probó la aplicación en producción, se identificaron bugs y mejoras visuales a realizar. A su vez se incorporó nueva funcionalidad que agregaron valor al producto final. Las funcionalidades incorporadas fueron:

- Elastic Search

  El ORM de Django tiene una mala performance con lo cual realizar búsquedas con filtros sobre la base de datos PostgreSQL resulta lento y afecta la experiencia del usuario. Como alternativa se incluyó Elasticsearch. Es decir, las noticias se almacenan en la base de datos PostgreSQL y se indexan en el motor de Elastic. Las consultas ahora se realizan sobre este motor haciendo que la performance mejore drásticamente.

- Trends

  Inicialmente la visualización de Trends recibía 2 o más palabras y mostraba en un gráfico de líneas la cantidad de titulares que contenían cada palabra por día en un intervalo de tiempo definido. Esto fue cambiado para reemplazar el número de cantidad por un valor porcentual que relaciona la cantidad total de noticias del día que incluyen una palabra en particular con el total de noticias registradas un día dado.

  El algoritmo para determinar las curvas de trends es el siguiente:

  1. Contar por día cuántas veces aparece cada una de las palabras a analizar.
  2. Para cada día y por cada una de las palabras hacer la división entre la cantidad de noticias que incluyen dicha palabra sobre el total de noticias para ese día.
  3. Determinar cuál es el valor máximo del paso 2.
  4. Para cada día y por cada palabra obtener la división entre valor obtenido en el paso 2 y el paso 3. Luego multiplicarlo por 100 obteniendo el valor final a graficar.

- Word Tree

  Es una librería desarrollada por Google para construir un árbol con los titulares. Al buscar una palabra en particular se genera la visualización que permite entender el contexto en un período de tiempo particular.

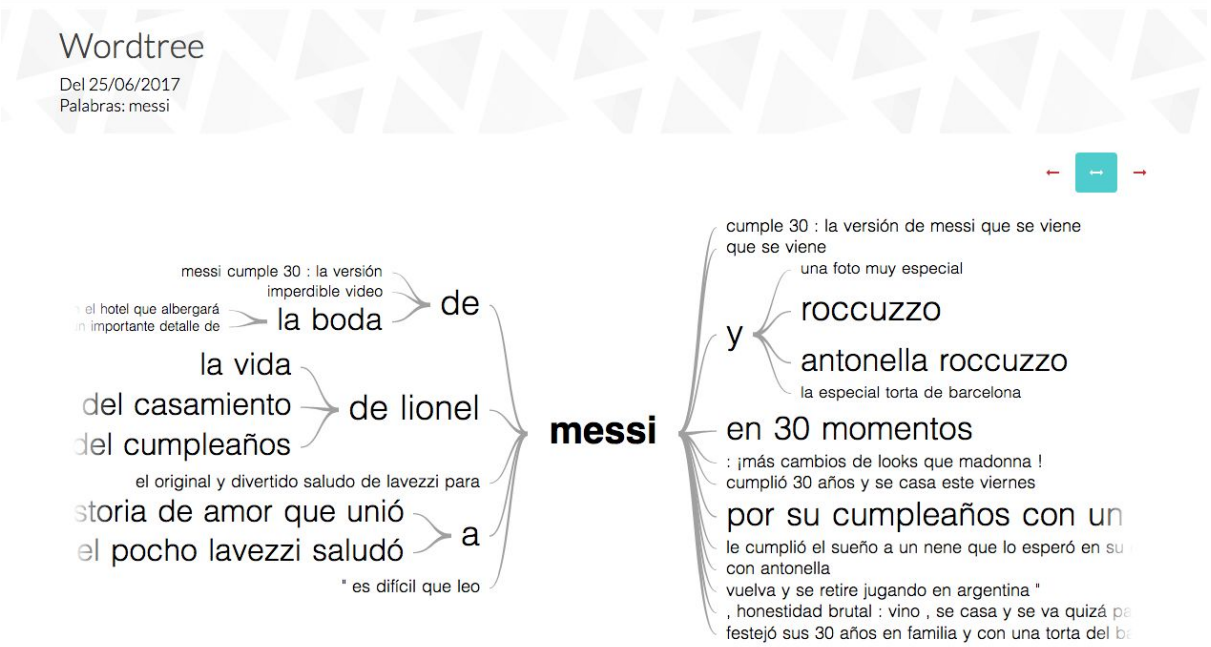


Imagen 11: Word Tree

- *Heatmap*

  Es una visualización desarrollada con d3.js inspirada en el *Heatmap* de github. La misma muestra una grilla con cuadrados de colores donde cada fila corresponde a un medio y cada columna a un día en particular. El color de cada cuadro en la grilla varía de blanco a azul oscuro dependiendo de la cantidad de noticias obtenidas en un día dado. De esta manera se puede comparar visualmente cual es el medio que ofrece más información.

Imagen 12: Heatmap

- Contexto de búsqueda

  Los parámetros de las búsquedas fueron agregados al query string de la URL con el objetivo de poder compartir con terceros una visualización sin la necesidad de que este vuelva a introducir los filtros. Además al cambiar de sección los parámetros se mantienen pudiendo así ver las distintas visualizaciones con los mismos criterios.

  Por otro lado se unieron visualizaciones. En cantidad de noticias es posible clickear sobre el gráfico de línea para un día en particular que redirigirá a la sección de títulos para ese día. En la sección nube de palabras es posible clickear sobre una palabra para ir al *WordTree* y ver el contexto en el que apareció esa palabra. Lo mismo se puede hacer desde la sección tendencias para ver el contexto de esa palabra en ese dia.

- Stop words

  Las “Stop words” son palabras que no se quieren ver en una nube de palabras ya que no ayudan a entender cual es el contexto. El objetivo de un word cloud es identificar inmediatamente las palabras relevantes. Las preposiciones, por ejemplo, son palabras que aparecen en casi todos los titulares pero que no permiten entender cuáles son los temas importantes del día. Para resolver esto se agregó un nuevo modelo a la base de datos, editable vía el ABM, que permite agregar “*stop words*” que serán ignoradas al generar el dataset para el wordcloud.

Imagen 13: ABM de Stop Words

- Sección ayuda

  Existe una sección de ayuda en la aplicación donde se describe el funcionamiento de cada sección. Se incorporó la sección luego de que varios usuarios indicaron que no comprendían bien algunas de las visualizaciones.

- Sección API

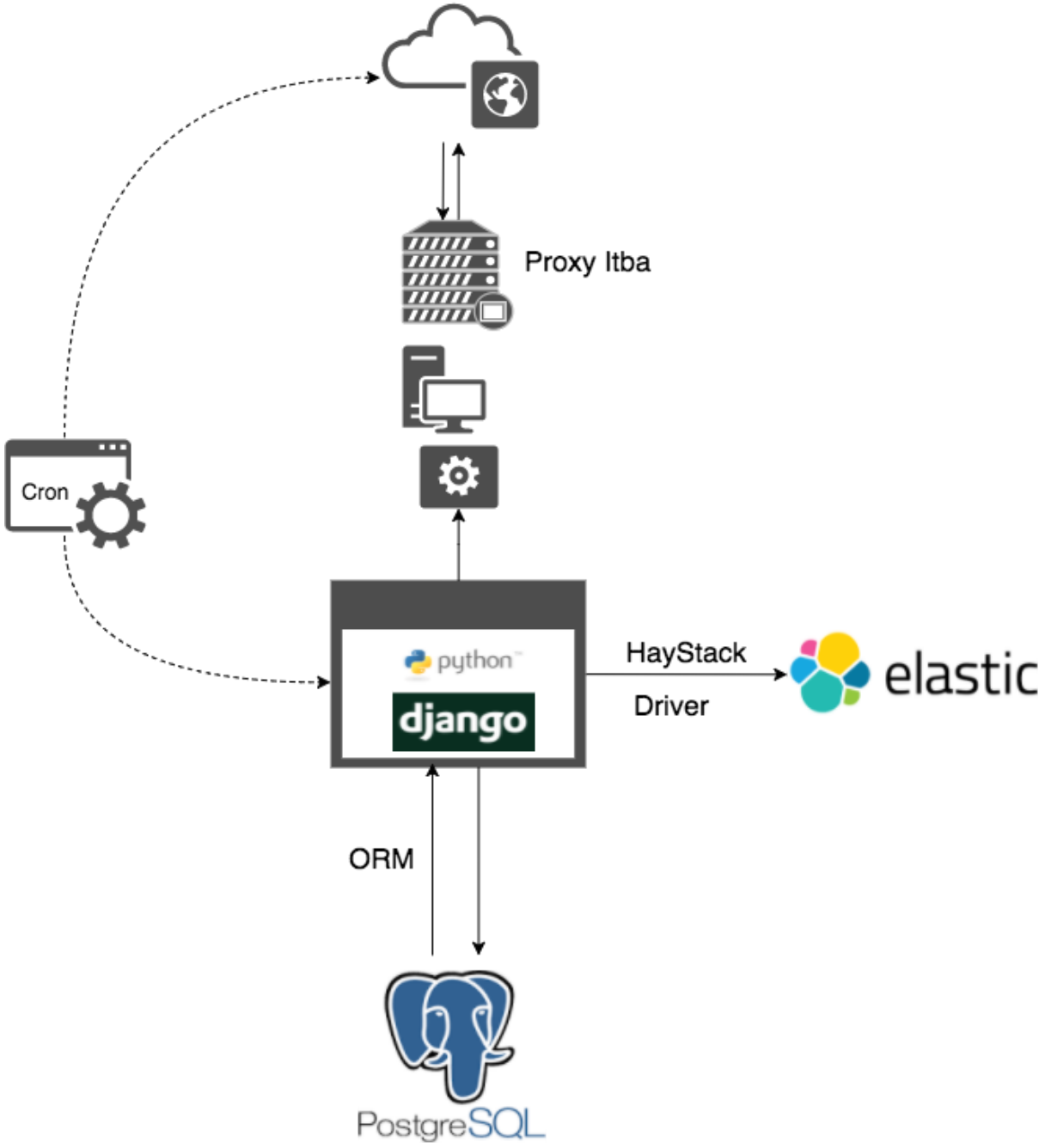
  En esta sección se describe el uso del API para obtener los datasets usados en cada visualización.

- Sección Documentación

  Dicha sección contiene los documentos asociados a la instalación y mantenimiento de la aplicación. Actualmente cuenta con 2 documentos: manual de instalación, que detalla cómo poner en producción la aplicación, y manual de funcionalidad, que describe cómo incorporar vistas y extender el API.

Diseño de arquitectura

El siguiente diagrama muestra la arquitectura de la aplicación final presentada.



Pruebas de Estrés

Las pruebas de estrés fueron realizadas con JMeter con el objetivo de medir el tiempo de respuesta de los servidores de la aplicación nueva y la antigua. El resultado esperado frente a los cambios de arquitectura y tecnologías utilizadas era encontrar una disminución considerable en los tiempos de respuesta así como una menor cantidad de errores y timeouts cuando el tráfico aumenta.

Configuración

Las pruebas se hicieron contra las APIs de títulos y nube de palabras. Las búsquedas tenían un filtro para el día 30 de Junio de 2017 con la palabra clave “Messi”. Dichos filtros fueron elegidos dado que ese día fue el casamiento de Lionel Messi y tuvo un impacto en las noticias.

Las URLs utilizadas en las pruebas para la versión antigua fueron:

- http://infovis.it.itba.edu.ar/news/titulos_data.php?key=21232f297a57a5a743894a0e4a801fc3&date1=2017-06-30&keyword=messi&medios=all
- http://infovis.it.itba.edu.ar/news/wc_data.php?key=21232f297a57a5a743894a0e4a801fc3&date1=2017-06-30&keyword=messi&medios=all

Las URLs de la versión nueva fueron:

- http://pf2.it.itba.edu.ar/api/busqueda?d_from=2017-06-30&d_to=2017-06-30&words=messi
- http://pf2.it.itba.edu.ar/api/nube-de-palabras?d_from=2017-06-30&d_to=2017-06-30&words=messi&

Cada ejecución de un test consistió en tener 10 usuarios simultáneos en un período de 1 minuto consultando las APIs. Los tests se repitieron 5 veces para obtener un resultado estadísticamente significativo.

El servidor utilizado para realizar las pruebas fue la computadora ofrecida por el Instituto. La misma cuenta con 64gb de disco duro, 4 gb de memoria RAM y un procesador Genuine Intel de 2.4Ghz con dos núcleos.

Resultados

- Búsqueda

  Los resultados para las pruebas sobre el API de búsqueda se encuentran en la siguiente tabla. Los tiempos están expresados en segundos. Se destaca que el tiempo medio para la respuesta de la aplicación antigua es de casi 2 minutos mientras que la nueva versión tiene una media de 1,39 segundos. Incluso el tiempo máximo de respuesta de la nueva versión es significativamente menor al tiempo de la otra aplicación.

	Media (segundos)	**Mínimo (segundos)**	**Máximo (segundos)**
V0	118,17	82,96	147,69
V1	1,39	0,072	16,84

Tabla 1 - Tiempo de Respuesta del API de Búsqueda

- Nube de Palabras

  Los resultados para las pruebas sobre el API de nube de palabras se encuentran en la tabla 2 expresados en segundos. Nuevamente los tiempos de respuesta de la nueva versión son ampliamente inferiores.

	Media (segundos)	**Mínimo (segundos)**	**Máximo (segundos)**
V0	116,66	83,53	145,61
V1	0,75	0,15	6,33

Tabla 2 - Tiempo de Respuesta del API de Nube de Palabras

Los resultados muestran que ambas APIs de la versión nueva responden mucho más rápido que la otra versión. La incorporación de Elasticsearch permite hacer consultas más veloces junto con el framework de Django que puede manejar múltiples usuarios simultáneamente.

En pruebas adicionales más exigentes se detectó que la versión de Django puede recibir más de 200 usuarios en un período de 1 minuto sin tener errores. Mientras que el servidor antiguo no soporta 20 usuarios en 1 minuto, frente a esta carga el servidor deja de responder.

Problemas encontrados

A continuación se enumeran algunos de los problemas encontrados a lo largo del desarrollo del proyecto.

- No estandarización de RSS

  Los RSS consumidos no siguen un estándar. Es decir, utilizan distintos tags para mandar la misma información y el formato no siempre es el mismo. El caso más destacado es el de la fecha ya que existen muchos formatos distintos. Para resolver el problema de la fecha se utiliza una función propia de python que recibe un string de una fecha en cualquier formato y lo estandariza en un objeto de tipo fecha.

- Categorías

  Habitualmente una noticia tiene una categoría asociada (por ejemplo: Política, Deportes, Espectáculos, etc). Cada medio utiliza categorías distintas y no todos lo incluyen en las RSS por esta razón la aplicación no tiene un filtro por categorías.

- API de Facebook y Twitter

  Las redes sociales Facebook y Twitter cerraron las APIs públicas que permitían consultar la cantidad de veces que se compartía una noticia dada. Esto impidió que se pudiera arreglar la funcionalidad que involucraba las redes sociales.

- Hardware

  La aplicación se encuentra instalada en una máquina de la facultad. Dicha máquina tiene un hardware bastante limitado que no ayuda a mejorar la performance en general. Al correr el sistema en una Macbook Pro de 256gb de disco duro, 16gb de RAM y un procesador Intel Core i5-3210M de 2.50GHz notamos un incremento de performance.

Posibles mejoras

El proyecto tiene mucho potencial para crecer y ofrecer funcionalidades nuevas para usuarios que quieran informarse u obtener estadísticas de lo que ocurre diariamente. Algunas posibilidades son las siguientes.

- Versión mobile

  Hoy en día los dispositivos mobile son usados constantemente para consumir información. Sería importante desarrollar una versión web mobile o nativa de la misma aprovechando el API que expone la información.

- Relevancia de medios

  El sistema tiene cargados más de 70 medios distintos. Algunos tienen mayor impacto y son consumidos más o menos por el público. Sería interesante poder ordenar las noticias no solo por fecha de publicación si no también por la relevancia del medio.

- Blogs

  Hoy en día los medios solo incluyen diarios online. Sería conveniente incluir Blogs locales que también pueden dar información de los eventos diarios. Además incluiría distintas perspectivas no solo de periodistas si no de gente capacitada en otras áreas que no escriben en diarios.

- Preprocesamiento

  Los datasets para las visualizaciones se construyen en el momento a partir de los parámetros: títulos, medios y fecha de publicación. Con el objetivo de mejorar la performance y bajar el tiempo de procesamiento se podría hacer un pre procesamiento en el momento que se carga la noticia en la base de datos. Por ejemplo, eliminar los signos de puntuación para que sea más rápido generar la nube de palabras.

- Imágenes

  Algunos medios envían un link a las imágenes asociadas a las noticias. Si bien no todos lo hacen sería bueno incluir las imágenes en el home de la página junto con cada titular o crear una nueva sección con las imágenes de los temas más hablados en el día.

- Compartir visualizaciones

  Agregar la opción de compartir visualizaciones en redes sociales como Facebook, Twitter o Instagram.

Apéndice

Diversas librerías y recursos externos fueron utilizados en el desarrollo del proyecto. A continuación se lista cada uno de ellos.

- D3.js una librería de Javascript para desarrollar visualizaciones.[10]
- C3 una librería de Javascript para armar gráficos.[11]
- Google Charts una librería de Javascript para crear gráficos.[12]
- Haystack un plugin de Django para indexar elementos en Elasticsearch y buscar.[13]
- AsciiFoldingBackend, un backend para hacer búsquedas ignorando acentos en Elasticsearch con Haystack.[14]
- Django CORS, librería para manejar CORS en el servidor.[15]
- Triangle[16] template de html, css y Javascript que se utilizó como base para crear la página
- Bootstrap-select[17] es un plugin de JQuery que utiliza Bootstrap's dropdown.js y agrega funcionalidad extra a los select
- d3.layout.cloud.js[18] para la nube de palabras.

[10] https://d3js.org/
[11] http://c3js.org/
[12] https://developers.google.com/chart/interactive/docs/gallery/wordtree
[13] http://django-haystack.readthedocs.io/en/v2.6.0/index.html
[14]https://mounirmesselmeni.github.io/2015/12/13/enable-asciifolding-in-elasticsearchhaystack/
[15] https://github.com/ottoyiu/django-cors-headers
[16] http://demo.themeum.com/html/triangle/1.1/index.html
[17] https://silviomoreto.github.io/bootstrap-select/
[18] https://github.com/jasondavies/d3-cloud

Conclusión

El proyecto desarrollado ofrece múltiples visualizaciones para analizar y entender las noticias en un período de tiempo determinado. Las conexiones entre visualizaciones y el contexto que se mantiene entre ellas ayudan a ver la misma información para un intervalo de tiempo dado de distintas formas apreciando diversas características y pudiendo obtener una visión más global de las mismas. Así también apreciar las relaciones entre las diferentes palabras o noticias, y detectar rápidamente cuales son los temas destacados o relevantes al momento buscado

Por otra parte, la incorporación de nuevas tecnologías resaltando el uso del framework Django exclusivo para desarrollo de aplicaciones web junto con el motor Elasticsearch permite ofrecer un alto rendimiento así como atender una gran cantidad de usuarios simultáneamente.

La visualización de noticias y extracción de información tiene múltiples usos y aplicaciones. El proyecto tiene potencial para crecer y explorar nuevas alternativas. En la actualidad el mismo ofrece una base robusta para iterar sobre el mismo incorporando funcionalidad. En particular la exposición de la API para que terceros puedan consumirlas y generar visualizaciones, reportes o nuevas aplicaciones.