



# BÚSQUEDA DE RESPUESTAS SOBRE SALTA

Enzo Rubén Notario

Ingeniería en informática

Facultad de Ingeniería

Universidad Católica de Salta

AÑO 2018



**Título**

Ingeniería en Informática

**Profesor Guía**

**Nombre:** Lic. Carolina Cardoso

**Firma:** \_\_\_\_\_

**Tribunal Evaluador**

**Nombre:** \_\_\_\_\_

**Firma:** \_\_\_\_\_

**Nombre:** \_\_\_\_\_

**Firma:** \_\_\_\_\_

**Nombre:** \_\_\_\_\_

**Firma:** \_\_\_\_\_

**Fecha de Exposición**

\_\_\_\_\_/\_\_\_\_\_/\_\_\_\_\_

## **Agradecimientos**

A la Lic. Carolina Cardoso, quien supo guiarme sabiamente y con total predisposición durante este trabajo y el trabajo de investigación.

A la Dra. Alicia Pérez, con quien obtuve una gran experiencia durante el trabajo de investigación.

A mis padres, Titina y Hugo, quienes hicieron posible todo esto.

A mis hermanos, Brisa y Eloy por el constante apoyo y cariño.

A mis amigos y compañeros, quienes me acompañaron durante esta carrera.

# Índice

Índice	1
Abstract	8
1. Introducción	9
1.1 Descripción del problema	9
1.2 Importancia del problema	10
1.3 Pasos a realizar	12
1.4 Criterios de éxito	12
2. Estado de la cuestión	13
2.1 Conceptos previos	13
Web Crawler	13
Web Scraping	13
Procesamiento del lenguaje natural (PLN)	14
Recuperación de la información (RI)	14
Lenguaje natural	14
Open Information Extraction (OIE)	14
Recall	15
Precisión	15
2.2 Antecedentes	15
ReVerb	15
ExtrHech	16
QARelaciones	17
2.3 Arquitectura de un SBR	19
Análisis de la pregunta	19
Recuperación de información	19
Extracción de la respuesta	20

2.4 Análisis de una oración	20
2.5 FreeLing	20
3. Definición del problema	24
3.1 Definición de objetivos y alcances	24
Objetivo general	24
Objetivos específicos	24
Alcance del proyecto	24
3.2 Situación actual	24
3.3 Alternativas tecnológicas	26
4. Propuesta	28
4.1 Solución propuesta	28
4.2 Herramientas utilizadas	28
Ubuntu	28
Apache	28
MySQL	29
FreeLing	29
Apache Lucene	29
Apache Solr	29
PHP	30
Laravel	30
Guzzle	30
Goutte	30
Solarium	30
Angular	30
Covalent	31
PhpStorm	31

Git	31
4.3 Metodología	31
Fase 1: Comprensión del negocio	33
Fase 2: Estudio y comprensión de los datos	33
Fase 3: Análisis de los datos y selección de características	34
Fase 4 y 5: Modelado y Evaluación	34
Fase 6: Despliegue	37
4.4 Arquitectura	37
4.4.1 Obtención de textos de la web	38
4.4.2 Extracción de información	43
4.5 Buscador de respuestas	52
Interfaz de usuario	55
Interfaz que muestra el proceso de extracción	55
4.6 Análisis FODA	57
Factores internos	57
Factores Externos	57
4.7 Análisis de factibilidad	58
Factibilidad económica	58
Factibilidad técnica	59
Factibilidad legal	59
4.8 Gestión de Riesgos	59
5. Resultados	61
5.1 Obtención de texto	61
5.2 Extracción de Documentos	61
5.3 Evaluación	61
Evaluación del componente de extracción de Documentos	61

Evaluación de las respuestas	62
6. Conclusiones	64
6.1 Futuras líneas de investigación	64
Bibliografía	65
Anexo 1 - Banco de preguntas	68
Anexo 2 - Resultados del Banco de preguntas	71
Anexo 3 – Resultado de la evaluación del componente de extracción de Documentos	75
Anexo 4 - Códigos fuente	85
Anexo 5 - Instalación y configuración	113
Instalación del sistema operativo	113
Instalaciones generales	118
Instalación de Solr	119
Configuración	119
Instalación de Freeling	123
Instalación de Laravel	123
Anexo 6 – Resolución Rectoral N° 839/13	125
Anexo 7 – Resolución Rectoral N° 421/15	126

## Índice de figuras

Figura 1 - Arquitectura del sistema .....	10
Figura 2 - Resultado en Google a la pregunta "¿Por qué ruta se llega al Parque Nacional El Rey?" .....	11
Figura 3 - Restricción Sintáctica para las relaciones extraídas por ReVerb.....	16
Figura 4 - Restricción sintáctica para las relaciones extraídas por ExtrHech .....	16
Figura 5 - Restricción sintáctica para los grupos nominales de ExtrHech .....	17
Figura 6 - Orden de los argumentos en el idioma Español e Inglés .....	17
Figura 7 - Patrones que usa QARelaciones para detectar la frase verbal en una oración .....	17
Figura 8 - Fragmento Destacado de Google.....	26
Figura 9 - Fases CRISP-DM.....	33
Figura 10 - Script de NLTK .....	36
Figura 11 - Visualización gráfica del resultado de NLTK .....	37
Figura 12 - Resultado de NLTK.....	37
Figura 13 - Formato de los archivos JSON extraídos de turismo.salta.gov.ar .....	39
Figura 14 - Formato de los archivos JSON extraídos de salta.gov.ar .....	40
Figura 15 - Formato de los archivos JSON extraídos de wikipedia.org.....	42
Figura 16 - Diagrama de Entidad-Relación.....	47
Figura 17 - Primer Documento que se extrae de la página de La Ciudad de Salta en Wikipedia .....	48
Figura 18 - Indexación en Solr del primer Documento que se extrajo de la página de La Ciudad de Salta en Wikipedia .....	51
Figura 19 - Documento que se extrae de una pregunta en lenguaje natural hecha por el usuario final.....	53
Figura 20 - Respuestas en formato JSON a una pregunta realizada en lenguaje natural por el usuario final .....	54



Figura 21 - Visualización en la Interfaz de Usuario de las respuestas en formato JSON a una pregunta realizada por el usuario final en lenguaje natural .....	54
Figura 22 - Comparación entre QARelaciones, Google y SBRS de resultados sobre el Banco de preguntas.....	63
Figura 23 - Rutas .....	85
Figura 24 - Tabla "documents" .....	85
Figura 25 - Tabla "visitedSites" .....	85
Figura 26 - Tabla "queuedSitesTable" .....	86
Figura 27 - Tabla "synonymous" .....	86
Figura 28 - Modelo "Document" .....	87
Figura 29 - Modelo "VisitedSite" .....	87
Figura 30 - Modelo "QueuedSite" .....	87
Figura 31 - Modelo "Synonym" .....	88
Figura 32 - Controlador "SearchController" .....	92
Figura 33 - Controlador "CorpusController" .....	93
Figura 34 - Controlador "ExtractorController" .....	95
Figura 35 - Job "ExtractorJob" .....	97
Figura 36 - Job "WikipediaCrawlerJob" .....	97
Figura 37 - Comando "ExtractorCommand" .....	98
Figura 38 - Comando "CrawlerCommand" .....	99
Figura 39 - Clase "Crawler" .....	103
Figura 40 - Clase "Synonymous" .....	105
Figura 41 - Clase "Extractor" .....	112

## Índice de tablas

Tabla 1 - Etiquetas EAGLES para adjetivos .....	21
Tabla 2 – Análisis morfológico de la palabra "Alegre" .....	21
Tabla 3 - Tripletas que logra extraer ExtrHech .....	35
Tabla 4 - Petición enviada a Solr para indexar un Documento .....	50
Tabla 5 - Tabla de riesgo .....	60
Tabla 6 - Resultados de la obtención de texto .....	61
Tabla 7 - Comparación entre QARelaciones, ExtrHech y SBRS .....	62
Tabla 8 - Banco de preguntas .....	70
Tabla 9 - Resultados del Banco de preguntas .....	74
Tabla 10 - Resultados de la evaluación frente a las sesenta y ocho oraciones tomadas de textos escolares.....	84

## Abstract

Este proyecto de grado consiste en el desarrollo de un Sistema de Búsqueda de Respuestas sobre Salta capaz de recibir preguntas en lenguaje natural y devolver una respuesta, también en lenguaje natural.

El sistema se diseñó utilizando herramientas para el procesamiento del lenguaje natural y tecnologías web. Se recopilaron textos extraídos de la World Wide Web, específicamente de dos sitios oficiales del Gobierno de la Provincia de Salta y Wikipedia.

De estos textos se extrajeron relaciones basadas en un verbo o una frase verbal, las cuales se indexaron formando la base de conocimiento. Una pregunta de un usuario realizada en lenguaje natural se convierte a una consulta al índice, el cual devuelve como resultado una serie de respuestas candidatas, entre las cuales se determina la mejor respuesta.

La comparación del sistema con otros del estado del arte, en cuanto a la exactitud de las respuestas obtenidas, demuestran la validez del enfoque planteado.

## 1. Introducción

Actualmente la *World Wide Web*, WWW de ahora en más, alberga una gran cantidad de información, la mayoría en forma de texto no estructurado y disperso en distintos sitios, lo cual hace difícil la tarea de localización y extracción de información concreta que responda a una pregunta realizada en el lenguaje hablado por los humanos: el natural. Además del texto útil que la WWW posee como información, también posee una gran cantidad de texto inútil, como los del menú del sitio web o pequeñas descripciones de otros artículos que suelen rodear al artículo central.

Es necesario, entonces, poder recoger de los distintos sitios y de una manera automatizada, la información que estos contienen, dejando de lado el gran volumen de texto innecesario que se genera día a día.

Por otra parte, para el usuario final es mucho más fácil realizar una búsqueda en el lenguaje que utiliza a diario, que tener que aprender y armar una consulta especial para obtener resultados de, por ejemplo, una Base de Datos Relacional.

Además, las repuestas deben mostrarse de manera sencilla y amigable ante el usuario, quien debería poder encontrar la respuesta a su pregunta entre los distintos resultados que el sistema brinda en la misma pantalla, sin que sea necesario navegar hacia otra URL para encontrar la respuesta entre un montón de texto.

El presente trabajo buscará aportar una solución a estos problemas enfocándose a brindar respuestas en lenguaje natural a preguntas también realizadas en lenguaje natural sobre la Provincia de Salta.

### 1.1 Descripción del problema

Actualmente la Provincia de Salta cuenta con distintos sitios oficiales de entre los cuales se destacan: [www.salta.gov.ar](http://www.salta.gov.ar) y [www.turismo.salta.gov.ar](http://www.turismo.salta.gov.ar). Estos sitios son actualizados día a día con información sobre la Provincia, incluyendo fechas de actividades turísticas, lugares a dónde ir, exposiciones, etc., y están dispuestos en forma de *Blog*: publicaciones, una tras otra, divididas en categorías y/o secciones. Otro sitio que aporta bastante información a pesar de no ser oficial es Wikipedia: [www.wikipedia.org](http://www.wikipedia.org), el cual también presenta dicha información en forma de texto plano sin un buscador.

Para un usuario final que busca una información en particular puede resultar complejo recorrer uno o más sitios web hasta encontrarlo. Si bien se puede utilizar un motor de búsqueda, como por ejemplo Google, el usuario obtendrá como respuesta una lista de enlaces a sitios web donde posiblemente encuentre la información buscada. Idealmente el usuario podría disponer de un sistema donde pueda realizar una pregunta, en lenguaje natural, y obtener sólo la respuesta.

Para solucionar este problema es necesario plantear un Sistema de Búsqueda de Respuestas, SBR de ahora en más, que implemente la arquitectura que se muestra en Figura 1: A partir de una gran cantidad de texto no estructurado obtenido de distintos sitios webs, se extrae información relevante en forma de relaciones que son indexadas y forman el índice. El usuario final puede ingresar su pregunta en lenguaje natural, la cual será convertida en una consulta al índice, y obtendrá una serie de resultados que contienen la respuesta, también en lenguaje natural, en vez de obtener enlaces a sitios webs donde podría encontrarse la respuesta.

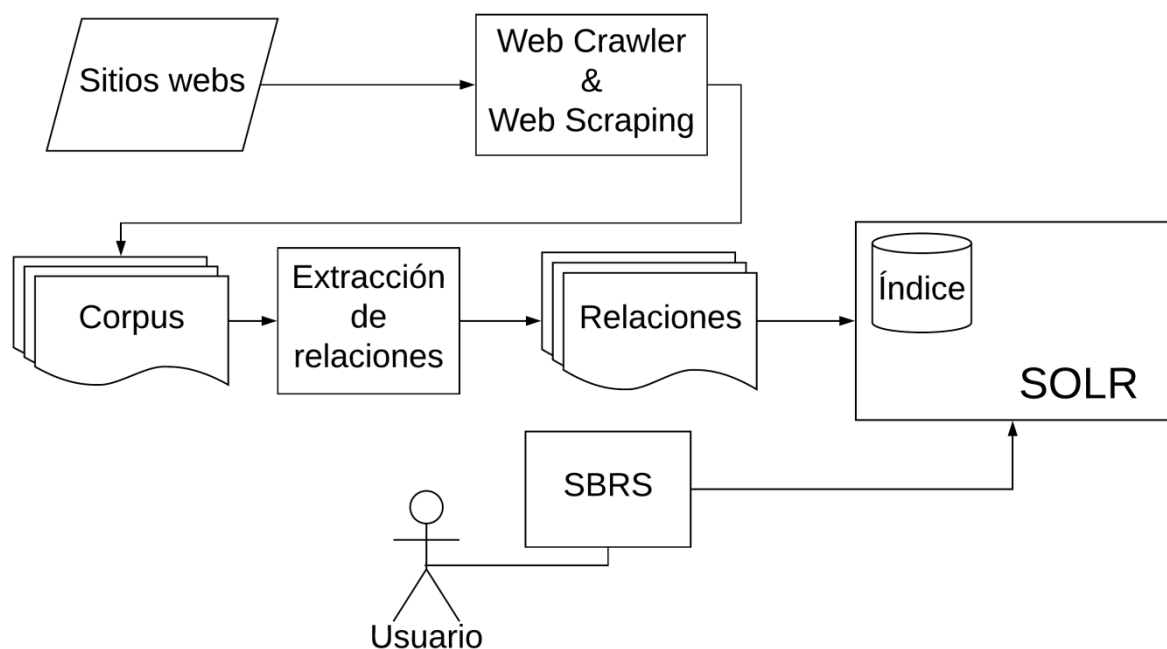


FIGURA 1 - ARQUITECTURA DEL SISTEMA

## 1.2 Importancia del problema

Para comprender la importancia del problema se buscará la respuesta a la pregunta *¿Por qué ruta se llega al Parque Nacional El Rey?* en Google:

The image is a screenshot of a Google search results page. At the top, the Google logo is on the left, and the search bar contains the text "¿Por qué ruta se llega al Parque Nacional El Rey?". To the right of the search bar are icons for voice search and a magnifying glass. Below the search bar, there are tabs for "Todos", "Maps", "Imágenes", "Noticias", "Videos", "Más", "Preferencias", and "Herramientas". The "Todos" tab is selected. Below the tabs, it says "Cerca de 809,000 resultados (0.66 segundos)". The first search result is titled "Parque Nacional El Rey - Administración de Parques Nacionales" with a green URL. The snippet describes the park's location and size. Below this is a section titled "Preguntas relacionadas" with four questions and dropdown arrows. Below that is a link to "Parque Nacional 'El Rey' - Salta - Portal de Salta" with a snippet about the park's location. This is followed by a link to the Wikipedia page for "Parque nacional El Rey" with a snippet about its climate. Next is a link to "Como Llegar Parque Nacional El Rey, Salta - Rutas y Distancias" with a snippet about the route. Then is a link to "Parque Nacional El Rey | Avistaje de aves | Viajes Argentina" with a snippet about bird watching. This is followed by a link to "Parque Nacional El Rey (Provincia de Salta) - consejos útiles antes de ..." with a snippet about the park's location and a rating. Next is a link to "Parque Nacional El Rey - Salta - Que hacer, que visitar y como llegar ..." with a snippet about the park's location. Then is a link to "Parque Nacional El Rey, en Salta - Sobre Argentina" with a snippet about the park's history. This is followed by a link to "Parque Nacional El Rey - Salta - Welcome Argentina" with a snippet about the park's location. Finally, there is a link to "CULTURA Y TURISMO SALTA: Parque Nacional El Rey" with a snippet about the park's location.

Google

¿Por qué ruta se llega al Parque Nacional El Rey?

Todos Maps Imágenes Noticias Videos Más Preferencias Herramientas

Cerca de 809,000 resultados (0.66 segundos)

**Parque Nacional El Rey - Administración de Parques Nacionales**  
<https://www.parquesnacionales.gob.ar/areas-protegidas/region-noroeste/pn-el-rey/> ▼  
El Parque Nacional El Rey se ubica a unos 200 kilómetros al Este de la ciudad ... Posee una superficie de 44.162 hectáreas que contienen muestras de las ... por Ruta Nacional N° 9 hasta Lumbreira, Ruta Provincial N° 5 hasta Paso de la ... sólo se llega hasta el paraje Paso de la Cruz, a 48km del Parque Nacional El Rey.

Preguntas relacionadas

- ¿Que se protege en el Parque Nacional El Rey? ▼
- ¿Cómo se creó el Parque Nacional El Rey? ▼
- ¿Cuáles son los parques nacionales de la provincia de Salta? ▼
- ¿Dónde se encuentra el Parque Nacional Los Cardones? ▼

Comentarios

**Parque Nacional "El Rey" - Salta - Portal de Salta**  
<www.portaldesalta.gov.ar/elrey.htm> ▼  
El Parque está encerrado por cordones montañosos que descienden ... Por dicha ruta, tras recorrer 35 km., se llega a la portada Parque Nacional, por la cual se ...

**Parque nacional El Rey - Wikipedia, la enciclopedia libre**  
[https://es.wikipedia.org/wiki/Parque\\_nacional\\_El\\_Rey](https://es.wikipedia.org/wiki/Parque_nacional_El_Rey) ▼  
El parque nacional El Rey se encuentra en el departamento de Anta, Provincia de Salta, ... Posee un clima tropical con temperaturas medias que oscilan entre los 22 y 25 °C. Las precipitaciones alcanzan los 2000 mm anuales. ... El estrato más alto se caracteriza por la presencia de grandes ejemplares de palo blanco ...

**Como Llegar Parque Nacional El Rey, Salta - Rutas y Distancias**  
[www.derutasydestinos.com/comollegar\\_parques\\_Parque\\_Nacional\\_El\\_Rey-Salta.html](www.derutasydestinos.com/comollegar_parques_Parque_Nacional_El_Rey-Salta.html) ▼  
Como Llegar a Parque Nacional El Rey, en General Guemes, Salta, con rutas, ... por la cantidad de litros de combustible totales que puede cargar el vehículo.

**Parque Nacional El Rey | Avistaje de aves | Viajes Argentina**  
<www.tripin.travel/blog/parque-nacional-el-rey-avisataje-aves-salta-pn-rey/> ▼  
6 mar. 2014 - El Parque Nacional El Rey en Salta es el lugar preferido para los amantes ... Siguiendo la RN 34, y luego RP 5 y RP 20, se llega al Parque Nacional El Rey, ambiente de ... Se caracteriza por los numerosos ríos que se originan en su ... el diseño de nuevos senderos, la identificación de rutas de enlace y ...

**Parque Nacional El Rey (Provincia de Salta) - consejos útiles antes de ...**  
<https://www.tripadvisor.com.ar/.../Actividades-en-Provincia-de-Salta> ▼  
★ ★ ★ ★ ★ Calificación: 3.5 - 8 opiniones  
Reserva Parque Nacional El Rey, Provincia de Salta en TripAdvisor: Consulta ... El camino de acceso atraviesa varios arroyos por lo que el ingreso se cierra los ...

**Parque Nacional El Rey - Salta - Que hacer, que visitar y como llegar ...**  
<argentinaturismo.org/salta/parque-nacional-el-rey/> ▼  
Parque Nacional El Rey. Provincia de Salta | por Nicolas Martinez. Se encuentra ubicado dentro de la provincia de Salta, en el Departamento de Anta 80 ...

**Parque Nacional El Rey, en Salta - Sobre Argentina**  
<https://sobreargentina.com/2008/10/21/parque-nacional-el-rey-en-salta/> ▼  
21 oct. 2018 - El Parque Nacional El Rey fue creado en 1948, para proteger una impresionante ... Se encuentra ubicado en el norte de la provincia de Salta. ... Por lo mismo, te aconsejo calzar zapatos altos, que proteja tus tobillos; ... allí sigues por la Ruta Provincial N°5, hasta llegar a Paso de la Cruz, allí ...

**Parque Nacional El Rey - Salta - Welcome Argentina**  
<https://www.welcomeargentina.com/salta/parque-nacional-el-rey.html> ▼  
De la estancia del siglo XVIII que sirviera de fuerte para el Virreynato del Río de la Plata, ... En la actualidad, el Parque Nacional El Rey, creado en el año 1948, ... serranías de la Cresta de Gallo en el oeste y por la Sierra del Piquete al este .... km del Parque, se debe tomar la ruta nacional N° 9, que coincide en este tramo ...

**CULTURA Y TURISMO SALTA: Parque Nacional El Rey**  
<turismo.salta.gov.ar/contenido/58/parque-nacional-el-rey> ▼  
Página: Que Hacer y Conocer: Parque Nacional El Rey - Turismo Salta ... Es frío y se caracteriza por la presencia de pinos del cerro, matos y alisos, que son ...

FIGURA 2 - RESULTADO EN GOOGLE A LA PREGUNTA "¿POR QUÉ RUTA SE LLEGA AL PARQUE NACIONAL EL REY?"

Como se puede ver en la Figura 2, sólo el primer resultado brinda una parte de la respuesta correcta: “... por Ruta Nacional N° 9 hasta Lumbrera, Ruta Provincial N° 5 hasta Paso de la ...”. El usuario debe ingresar al enlace y buscar, entre todo el texto, la respuesta a su pregunta. Esto dificulta la búsqueda y consume más tiempo.

### 1.3 Pasos a realizar

Para desarrollar el sistema que sea capaz de brindar una respuesta directa a una pregunta realizada en lenguaje natural se comenzará por obtener la mayor cantidad de texto posible de distintos sitios previamente seleccionados. A partir de estos textos se buscarán relaciones basadas en un verbo o una frase verbal, las cuales serán indexadas y formarán la base de conocimiento. La pregunta del usuario realizada en lenguaje natural será convertida a una consulta al índice y se le devolverán los resultados que éste devuelva.

### 1.4 Criterios de éxito

Los criterios de éxito que determinarán la utilidad de la solución propuesta son los siguientes:

- Construir un sistema de extracción de relaciones que extraiga el 80% de las relaciones de los textos que conforman el corpus
- Seguir adecuadamente la metodología propuesta.

## 2. Estado de la cuestión

La primera discusión sobre las características que debería cumplir un Sistema de Búsqueda de Respuesta, así como la primera aproximación a un sistema funcional (QUALM) fueron introducidos por Wendy Lenhart a finales de la década de 1970.

La investigación en SBR tuvo sus orígenes en el campo de la Inteligencia Artificial (IA). Inicialmente se consideró requisito indispensable que los SBR satisficieran todas y cada una de las características ideales de un sistema de IA. Sin embargo, hasta la fecha únicamente se han podido obtener buenos resultados a costa de restringir los dominios sobre los que se realizan las consultas.

Los sistemas actuales afrontan la tarea de búsqueda desde la perspectiva de un usuario casual: un usuario que realiza preguntas simples que requieren como contestación un hecho, situación o dato concreto (John Burger, 2003). La dificultad de localizar y verificar la respuesta precisa hace de la tarea de Búsqueda de Respuestas (BR) una tarea más desafiante que las tareas comunes de Recuperación de la Información (RI). Por esta razón, se hace necesario el uso de técnicas avanzadas de lenguaje natural más allá de la simple extracción y expansión de términos.

### 2.1 Conceptos previos

#### Web Crawler

Un Web Crawler, WC de ahora en más, es un programa que se encarga de inspeccionar la WWW de forma metódica y automatizada para luego realizar alguna acción en cada página que visita. Comienza con al menos una URL previamente definida e identifica los hipervínculos en cada URL que visita para luego añadirlas a la lista de URLs a visitar de manera recurrente, de acuerdo a un determinado conjunto de reglas.

#### Web Scraping

La técnica Web Scraping, WS de ahora en más, consiste en la transformación de datos no estructurados de la web en datos estructurados, de manera automatizada, *parseando* los documentos HTML y extrayendo sólo el texto contenido en determinadas secciones del DOM del documento HTML a través de *querys* XPath (XML Path Language), previamente definidas, o a través de algún mecanismo similar que permita ubicar una determinada sección del documento a través de selectores de etiquetas HTML, identificadores, clases, etc.



### Procesamiento del lenguaje natural (PLN)

Es el campo que combina las tecnologías de la ciencia computacional con la lingüística aplicada, con el objetivo de hacer posible la comprensión y el procesamiento asistidos por ordenador de información expresada en lenguaje humano para determinadas tareas, como la traducción automática, los sistemas de recuperación, elaboración automática de resúmenes, interfaces en lenguaje natural, etc. (Vicomtech, 2016).

### Recuperación de la información (RI)

Es un área que se viene desarrollando desde finales de la década de 1950, aunque en la actualidad adquiere un rol más importante debido al valor que tiene la misma. Se puede plantear que disponer o no de la información justa en tiempo y forma puede resultar en el éxito o fracaso de una operación. De aquí la importancia de los Sistemas de Recuperación de Información, de ahora en más SRI, que pueden manejar, con ciertas limitaciones, estas situaciones de manera eficaz y eficiente.

Croft dice que la recuperación de información es “el conjunto de tareas mediante las cuales el usuario localiza y accede a los recursos de información que son pertinentes para la resolución del problema planteado. En estas tareas desempeñan un papel fundamental los lenguajes documentales, las técnicas de resumen, la descripción del objeto documental, entre otras” (W.B., 1987). Por otro lado, Korfhage definió la RI como “la localización y presentación a un usuario de información relevante a una necesidad de información expresada como una pregunta” (Korfhage).

### Lenguaje natural

Cuando hablamos de *lenguaje natural* nos referimos al lenguaje hablado o escrito, empleado por humanos, para propósitos generales de comunicación. Se utiliza este término para diferenciarlo de los lenguajes formales, contruidos de forma artificial, como aquellos empleados en programación o en lógica formal.

### Open Information Extraction (OIE)

*Open Information Extraction* (OIE) consiste en generar una representación estructurada de textos, generalmente en forma de proposiciones de tres componentes llamadas tripletas, las cuales están formadas por dos argumentos y una relación entre ellos.

Por ejemplo, dada la siguiente oración: “Salta es una ciudad, un municipio y la capital de la provincia de Salta.”, un sistema OIE extraerá al menos una tripleta:

(Salta; es; una ciudad, un municipio y la capital de la provincia de Salta)

Donde “Salta” y “una ciudad, un municipio y la capital de la provincia de Salta” son los argumentos y “es” es la relación.

La relación de una tripleta se basa en un verbo o una frase verbal (Gamallo, Garcia, & Fernández-Lanza, 2012).

### Recall

Es una métrica utilizada para evaluar el componente de extracción de relaciones e indica el porcentaje de relaciones obtenidas del total de posibles relaciones correctas. El conjunto de posibles relaciones correctas está formado por las relaciones correctas obtenidas por el sistema más otras relaciones manualmente extraídas que se espera que el sistema sea capaz de encontrar.

### Precisión

Es una métrica similar a *recall* e indica el porcentaje de relaciones extraídas que son correctas.

## 2.2 Antecedentes

### ReVerb

ReVerb (Fader, Soderland, & Etzioni) es un sistema que implementa la extracción de relaciones de proposiciones en el idioma inglés en forma de tripletas basadas en verbos o frases verbales. El texto es sometido al etiquetado POS (*Part Of Speech*) que consiste en asignarle una etiqueta a cada palabra o símbolo que determina su categoría gramatical. Primero se detectan las relaciones (basadas en el verbo) y luego en torno a ellas los argumentos de las relaciones. Este modelo de relaciones buscadas se describe mediante dos restricciones una restricción sintáctica y una restricción léxica.

La restricción sintáctica busca reducir el número de extracciones incoherentes o no informativas utilizando la expresión regular que se muestra en Figura 3 para encontrar la relación de la tripleta.

La restricción léxica se refiere a que una relación sólo es válida si ésta aparece en el corpus con una cierta frecuencia y con una diversidad de argumentos.

V   VP   VW*P
<p>donde:  V es un verbo  W puede ser un nombre, adjetivo, adverbio, pronombre o artículo  P es una preposición que puede ir seguida de un infinitivo  * representa repetición de cero o más veces  V W* P es un verbo con palabras dependientes</p>

FIGURA 3 - RESTRICCIÓN SINTÁCTICA PARA LAS RELACIONES EXTRAÍDAS POR REVERB

Para cada oración, ReVerb encuentra la secuencia más larga de palabras que satisfacen la restricción sintáctica y léxica, lo cual conforma la relación de la tripleta. A partir de esta relación forma el primer argumento tomando el grupo nominal más cercano a la izquierda de la relación que no contenga un pronombre relativo, una partícula interrogativa que comience por *Wh* o la palabra *There*. El segundo argumento se forma tomando el grupo nominal más cercano a la derecha de la relación. La tripleta es devuelta sólo si se encontraron los dos argumentos y la relación.

ReVerb está limitado a relaciones basadas en un verbo, a relaciones con sólo dos argumentos, y al análisis del idioma inglés debido a su restricción sintáctica.

### ExtrHech

El sistema ExtrHech (Zhila & Gelbukh, 2013) es prácticamente una implementación de ReVerb para el español. Comienza buscando una frase verbal limitada a un verbo o a un verbo seguido inmediatamente de palabras dependientes hasta encontrar una preposición (*nació en*), o una preposición seguida inmediatamente de un infinitivo (*sirven para acentuar*). La expresión regular que captura esta restricción se muestra en Figura 4. Como puede observarse se trata prácticamente de la misma restricción sintáctica de ReVerb.

VREL → (V W* P)(V)
<p>donde:  V es un verbo, incluida la posibilidad de un pronombre reflexivo (se caracterizan) o de un participio (relacionado)  W puede ser un nombre, adjetivo, adverbio, pronombre o artículo  P es una preposición que puede ir seguida de un infinitivo  * representa repetición de cero o más veces  V W* P es un verbo con palabras dependientes</p>

FIGURA 4 - RESTRICCIÓN SINTÁCTICA PARA LAS RELACIONES EXTRAÍDAS POR EXTRHECH

A continuación el sistema busca grupos nominales, según la expresión regular de la Figura 5, a la izquierda y derecha de la frase verbal para ubicar los dos argumentos de la relación. Un ejemplo de grupo nominal es “*la historia de la civilización romana*”.

N (PREP N)?
<p>donde: N es un nombre precedido opcionalmente por un artículo, adjetivo o un número ordinal, o una combinación de estos. PREP es una preposición. ? representa repetición de cero o una vez</p>

FIGURA 5 - RESTRICCIÓN SINTÁCTICA PARA LOS GRUPOS NOMINALES DE EXTRHECH

Si después de un nombre hay un participio que da comienzo a una cláusula subordinada terminada con un nombre, se considera una tripleta separada. Por ejemplo la oración “*Los egipcios se caracterizaron por sus creencias relacionadas con la muerte.*” Da lugar a dos tripletas:

Los egipcios; se caracterizaron por; sus creencias

Sus creencias; relacionadas con; la muerte

Al ser una implementación de ReVerb para el idioma Español, no contempla las relaciones en las que el orden de los argumentos es *Verbo-Sustantivo* en vez de *Sustantivo-Verbo*, lo cual es más común en el idioma Español que en el Inglés, como muestra la Figura 6.

Español:	<i>Entró la maestra</i> [VS]
	<i>La maestra entró</i> [SV]
Inglés	<i>The teacher come in</i> [SV]

FIGURA 6 - ORDEN DE LOS ARGUMENTOS EN EL IDIOMA ESPAÑOL E INGLÉS

## QARelaciones

El sistema QARelaciones (Cardoso, Pérez Abelleira, & Notario, 2016) sigue el enfoque OIE de una manera particular, y diferente al de ExtrHech (Zhila & Gelbukh, 2013).

Las tripletas se extraen siguiendo el patrón que se muestra en la Figura 7.

V   VW*P   VW*P+
<p>V = RN? P0? VA? (VM   VS   VP) (VN   VP)? W = (NC   NP   AQ   AO   RG   PP   PD   PX   PI   DA) P = SP (VN   VG)? donde: * representa repetición de cero o más veces + representa repetición de una o más veces ? representa repetición de cero o una vez Las etiquetas utilizadas se pueden ver en Tabla 1</p>

FIGURA 7 - PATRONES QUE USA QARRELACIONES PARA DETECTAR LA FRASE VERBAL EN UNA ORACIÓN

ReVerb toma como frase verbal la secuencia más larga de palabras que satisface el patrón, QARelaciones considera además todas las secuencias de palabras posibles que satisfacen el patrón, no sólo la más larga.

La frase verbal  $r$  detectada por los patrones se almacena en la tripleta con el verbo en infinitivo y eliminando los pronombres.

A continuación se localizan los argumentos. Por cada frase verbal  $r$  obtenida se buscan *chunks* a la izquierda y a la derecha de la frase verbal ( $x$  e  $y$  respectivamente). Un *chunk* es una secuencia de palabras más larga que no contiene un verbo y por lo menos tiene un nombre común o propio. De esta forma se obtiene la primera tripleta candidata  $\langle x;r;y \rangle$ , que es la tripleta con mayor cantidad de palabra. La tripleta se agrega al conjunto  $T$ .

Por ejemplo, dada la oración *De la médula espinal nacen los nervios periféricos, que permiten movimientos voluntarios e involuntarios, sensaciones y reflejos* para la frase verbal *permiten* el *chunk* a la izquierda que es la secuencia más larga sin un verbo y por lo menos con un nombre es *los nervios periféricos*, que se convierte en el primer argumento  $x$ . El *chunk* a la derecha es *movimientos voluntarios e involuntarios, sensaciones y reflejos*, segundo argumento  $y$ . La tripleta resultante es por lo tanto:

$\langle \text{los nervios periféricos; permiten; movimientos voluntarios e involuntarios, sensaciones y reflejos} \rangle$

Nótese que en esta oración existe otro verbo, *nacen*, que da lugar a su vez a otra tripleta.

La tripleta inicial es procesada de distintas maneras para generar el conjunto final de tripletas  $T$  como sigue. Un argumento  $arg$  se divide para obtener nuevas tripletas según las siguientes heurísticas:

h1: si  $arg$  contiene texto entre paréntesis, este texto se elimina, obteniendo  $arg_1$ .

h2: si  $arg$  contiene una conjunción (y, o, e) o contiene uno o más separadores (; o :) se divide  $arg$  en los fragmentos de texto a ambos lados del separador(es)  $arg_2, arg_3, \dots$

Este proceso se realiza para ambos argumentos  $x$  e  $y$ . Se generan tripletas para  $r$  y cada una de las combinaciones de fragmentos  $x_1, x_2, x, \dots$  con  $y_1, y_2, y_3, \dots$

En el ejemplo anterior las tripletas resultantes son:

<los nervios periféricos; permiten; movimientos voluntarios e involuntarios>

<los nervios periféricos; permiten; sensaciones>

<los nervios periféricos; permiten; reflejos>

Cada nueva tripleta obtenida se agrega a *T* si no se ha agregado ya (no se almacenan tripletas repetidas) y si cada argumento contiene por lo menos un nombre común o nombre propio.

Las heurísticas descritas generan un gran número de tripletas, algunas de ellas redundantes, incoherentes o hasta incorrectas. Sin embargo, dada la aplicación de las tripletas a responder preguntas, se prefiere un gran número de tripletas para aumentar el *recall* aún a riesgo de disminuir la *precisión*.

## 2.3 Arquitectura de un SBR

Un SBR se basa en tres fases bien definidas: *análisis de la pregunta*, *recuperación de información* y *extracción de la respuesta*.

### Análisis de la pregunta

El proceso inicia obteniendo una pregunta en lenguaje natural por parte del usuario y llevando a cabo la *formación de la consulta*. Tiene generalmente los siguientes objetivos:

- Extraer los términos para formar la consulta
- Concretar el contexto semántico de las respuestas

### Recuperación de información

Este proceso tiene como objetivo encontrar los documentos relevantes a la pregunta entre los que se espera que se encuentre la respuesta. El principal problema de los sistemas de RI consiste en realizar la indexación correcta de un corpus que luego servirá como soporte del SBR.

Se comienza realizando la *tokenización* de los documentos que conforman el corpus, que consiste en identificar cada palabra o conjunto de palabras (por ejemplo: estar, Provincia de Salta, etc.) en dichos documentos, para luego crear el índice sobre el cual consultará el SBR.

### Extracción de la respuesta

La última fase llevada a cabo en un SBR es la *extracción de la respuesta*, en la que se procesa el conjunto de documentos obtenidos por el sistema de RI con el objetivo de localizar y extraer la respuesta buscada por el usuario.

## 2.4 Análisis de una oración

Una oración expresada en lenguaje natural se puede caracterizar en base a tres criterios:

- **Morfológico:** Determina la categoría gramatical de cada palabra, esto es, determinar si la palabra es un sustantivo, adjetivo, artículo, pronombre, verbo, adverbio, interjección, preposición o conjunción, además de su género y número, y, en el caso de los verbos, el número, persona, tiempo, modo, voz y aspecto.
- **Sintáctico:** Estudia la forma en que se estructuran y organizan las palabras de una oración
- **Semántico:** Se enfoca en determinar el significado que tiene una oración, en base al contexto.

El análisis morfológico se realiza en base a otros tres tipos de análisis:

- *Stemming*: Consiste en determinar la raíz (*stem*) de una palabra. Por ejemplo, para las palabras *bibliotecario* y *bibliotecas*, el *stem* es *bibliotec*.
- *Lemmatization*: Consiste en determinar el lema que represente todas las formas flexionadas de una palabra. Por ejemplo: el lema de *leíamos* es *leer*.
- *Tokenization*: Consiste en la segmentación de una oración en base a signos de puntuación, expresiones numéricas, símbolos o cualquier otro elemento de un texto.

## 2.5 FreeLing

FreeLing<sup>1</sup> es una librería de código abierto para el procesamiento multilingüe automático, que proporciona una amplia gama de servicios de análisis lingüístico para diversos idiomas. FreeLing ofrece a los desarrolladores de aplicaciones de PLN funciones de análisis y anotación lingüística de textos, con la consiguiente reducción del coste de construcción de dichas aplicaciones.

---

<sup>1</sup><http://nlp.lsi.upc.edu/freeling/node/1>

Está concebido como una librería sobre la cual se puedan desarrollar potentes aplicaciones de PLN ofreciendo distintos servicios para diferentes lenguajes.

Puede realizar un análisis morfológico de cada palabra o símbolo, es decir, que obtendremos, por cada uno, su lema y etiqueta EAGLES<sup>2</sup>, que es un conjunto de dos a ocho caracteres que identifican la categoría gramatical de la palabra analizada.

Por ejemplo, para los adjetivos utiliza 7 caracteres como muestra la Tabla 1. Teniendo en cuenta esto, el adjetivo “alegre” será analizado y etiquetado como se ve en la Tabla 2: adjetivo (A) calificativo (Q) común (C) singular (S)

Adjetivos			
Posición	Atributo	Valor	Código
1	Categoría	Adjetivo	A
2	Tipo	Calificativo	Q
3	Grado	Apreciativo	A
4	Género	Masculino	M
		Femenino	F
		Común	C
5	Número	Singular	S
		Plural	P
		Invariable	N
6	Caso	-	0
7	Función	Participio	P

TABLA 1 - ETIQUETAS EAGLES PARA ADJETIVOS

Forma	Lema	Etiqueta
Alegre	Alegre	AQ0CS00

TABLA 2 – ANÁLISIS MORFOLÓGICO DE LA PALABRA "ALEGRE"

Dado el siguiente párrafo extraído de Wikipedia:

Salta es una ciudad, un municipio y la capital de la provincia de Salta, República Argentina, que tiene una población de 535 303 habitantes, siendo la ciudad más poblada de la provincia y la séptima del país. Su área metropolitana, denominada Gran Salta,[1] está compuesta por once municipios y tiene a una población de 554 125 habitantes (INDEC, 2010).

Al ejecutar el analizador de FreeLing en una distribución GNU/Linux con los siguientes parámetros:

```
$ analyze -f /usr/share/freeling/config/es-ar.cfg --usr --fmap user_map_file.conf --ner --outlv tagged < corpus/ciudadessalta.txt > corpus/ciudadessalta.txt.pos0
```

<sup>2</sup><http://nlp.lsi.upc.edu/freeling-old/doc/tagsets/tagset-es.html>



Parámetro	Explicación
-f /usr/share/freeling/config/es-ar.cfg	Ruta del archivo de configuración de FreeLing
--usr	Le indicamos que debe aplicar un conjunto de etiquetas específicas para ciertas palabras, definidas con el parámetro <i>-fmap</i>
--fmap user_map_file.conf	Archivo que indica las etiquetas específicas a utilizar para ciertas palabras. En éste archivo se encuentra sólo una regla: Salta salta NP00000 Que indica que cuando encuentra la palabra "Salta", el lema será "salta" y la etiqueta será "NP00000", para que la clasifique como un sustantivo propio, en vez de un verbo, que es la clasificación original que FreeLing le da a ésta palabra
--ner	Habilitamos el reconocimiento de nombre de entidades
--outlv tagged	Indicamos el nivel de análisis que efectuará FreeLing. En este caso, "tagged" devolverá una salida con una palabra por línea, <i>tokenizada</i> , analizada morfológicamente, etiquetada y con oraciones separadas por una línea vacía.

Se obtiene el siguiente resultado:

<p>Salta salta NP00000 1 es ser VSIP3S0 1 una uno DIOFS0 0.951973 ciudad ciudad NCFS000 1 , , Fc 1 un uno DI0MS0 0.99698 municipio municipio NCMS000 1 y y CC 0.999989 la el DAOFS0 0.98926 capital capital NCCS000 0.949605 de de SP 0.999961 la el DAOFS0 0.98926 provincia provincia NCFS000 1 de de SP 0.999961 Salta salta NP00000 1 , , Fc 1 República_Argentina república_argentina NP00000 1 , , Fc 1 que que PR0CN00 0.550139 tiene tener VMIP3S0 1 una uno DIOFS0 0.951973 población población NCFS000 1 de de SP 0.999961 535 535 Z 1 Fz 1 303 303 Z 1 11habitantes habitante NCCP000 1 , , Fc 1 siendo ser VSG0000 1 la el DAOFS0 0.98926 ciudad ciudad NCFS000 1 más más RG 0.99993 poblada poblar VMP00SF 1 de de SP 0.999961 la el DAOFS0 0.98926 provincia provincia NCFS000 1 y y CC 0.999989 la el DAOFS0 0.98926 séptima 7 AO0FS00 0.983871 de de SP 1 el el DA0MS0 1 país país NCMS000 1</p>
--

.. Fp 1  
 Su su DP3CSN 1  
 área área NCFS000 1  
 metropolitana metropolitano AQ0FS00 0.481264  
 ,, Fc 1  
 denominada denominar VMP00SF 1  
 Gran\_Salta gran\_salta NP00000 1  
 ,, Fc 1  
 [[ Fca 1  
 1 1 Z 1  
 ]] Fct 1  
 está estar VMIP3S0 0.999398  
 compuesta componer VMP00SF 1  
 por por SP 1  
 once 11 Z 1  
 municipios municipio NCMP000 1  
 y y CC 0.999989  
 tiene tener VMIP3S0 1  
 a a SP 0.998775  
 una uno DI0FS0 0.951973  
 población población NCFS000 1  
 de de SP 0.999961  
 554 554 Z 1  
 Fz 1  
 125 125 Z 1  
 habitantes habitante NCCP000 1  
 ( ( Fpa 1  
 INDEC indec NP00000 1  
 ,, Fc 1  
 2010 2010 Z 1  
 )) Fpt 1  
 .. Fp 1

## 3. Definición del problema

### 3.1 Definición de objetivos y alcances

#### Objetivo general

Desarrollar un SBR para que un usuario pueda realizar preguntas en lenguaje natural sobre la Provincia de Salta y que el sistema sea capaz de brindar la respuesta también en lenguaje natural de manera directa, en vez de devolver enlaces a sitios webs en donde el usuario tenga que buscar la respuesta que podría estar presente o no.

#### Objetivos específicos

- Responder preguntas de interés sobre la Provincia de Salta.
- Investigar las herramientas y tecnologías que permiten la extracción de texto de la web de manera automática.
- Desarrollar una herramienta capaz de convertir texto no estructurado en información para formar la base de conocimiento.
- Formar la base de conocimiento a partir de texto no estructurado de la WWW.
- Utilizar tecnologías webs para el desarrollo del sistema.

#### Alcance del proyecto

El proyecto propone desarrollar un SBR que se acople a los sitios oficiales del Gobierno de la Provincia de Salta fundamentado en una base de conocimiento de información textual extraída de la web.

### 3.2 Situación actual

Actualmente el buscador más conocido es Google ([www.google.com](http://www.google.com)), que es un motor de búsqueda en el que el usuario puede ingresar una pregunta en lenguaje natural y éste le devolverá una lista de enlaces a páginas donde el usuario posiblemente podrá encontrar la respuesta a su pregunta, leyendo el texto de la página.

También es posible formular una consulta especial para lograr una búsqueda refinada utilizando ciertos operadores<sup>3</sup>, como por ejemplo el operador *site* que le indica a Google que debe realizar la búsqueda sólo en un sitio web específico. Por ejemplo, ingresando la consulta “Urtubey site:www.salta.gov.ar”, Google devolverá sólo aquellos enlaces relacionados a la palabra “Urtubey” que se encuentren en el sitio [www.salta.gov.ar](http://www.salta.gov.ar).

---

<sup>3</sup> <http://guiagoogles.es/buscar-en-google-1-refinar/>

Hace unos años Google añadió a su buscador los Fragmentos Destacados<sup>4</sup>, que incluyen un resumen de la respuesta, extraído de una página web, y un enlace a dicha página, así como su título y URL. Estos tienen un aspecto similar al que se muestra en la Figura 8. De esta manera, Google busca brindarle al usuario una respuesta directa, sin que éste tenga que continuar su búsqueda en los links que el buscador devuelve como resultado, aunque, como se puede ver, muchas veces la respuesta no es directa y el usuario aún tiene que deducirla leyendo el texto devuelto.

Existen otros buscadores similares a Google, tales como DuckDuckGo (<https://duckduckgo.com/>), que es un buscador que protege la privacidad del usuario, o Bing (<https://www.bing.com/>), el buscador de Microsoft que es una evolución del buscador Yahoo! (<https://espanol.yahoo.com/>), entre otros.

La particularidad de todos estos buscadores es que su base de conocimientos se construye a partir de las distintas páginas que conforman la WWW y que sus respuestas son *links a páginas*, acompañados de un breve resumen de ésta, donde el usuario puede encontrar la respuesta que busca si es que accede a estos sitios y logra localizar, de entre todo el texto, la información que necesita.

---

<sup>4</sup> <https://support.google.com/webmasters/answer/6229325?hl=es-419>

The image shows a Google search interface. The search bar contains the text "¿Quién es el gobernador de la Provincia de Salta?". Below the search bar, there are tabs for "Todos", "Imágenes", "Noticias", "Videos", "Maps", "Más", "Preferencias", and "Herramientas". The "Todos" tab is selected. Below the tabs, it says "Cerca de 397,000 resultados (0.59 segundos)". A prominent red box highlights the first search result, which is a Wikipedia entry for "Juan Manuel Urtubey". The text in the red box reads: "Juan Manuel Urtubey (Ciudad de Salta, 6 de septiembre de 1969) es un político, abogado y profesor universitario argentino, actual Gobernador de la Provincia de Salta. Fue elegido Gobernador por primera vez en el año 2007 con tan solo 37 años." Below this text, there is a link to the Wikipedia page: "Juan Manuel Urtubey - Wikipedia, la enciclopedia libre" with the URL "https://es.wikipedia.org/wiki/Juan\_Manuel\_Urtubey". At the bottom right of the red box, there are links for "Acerca de este resultado" and "Comentarios". Below the red box, there is a section titled "Preguntas relacionadas" with four questions: "¿Cómo se llama el actual gobernador de Salta?", "¿Cómo se llama la esposa del gobernador de Salta?", "¿Quién es el vicegobernador de la provincia de Salta?", and "¿Cuáles son los limites de la provincia de Salta?". Each question has a dropdown arrow. At the bottom right of this section, there is a link for "Comentarios". Below the "Preguntas relacionadas" section, there are two search results. The first is "Gobernador de Salta - Gobierno de la Provincia de Salta" with the URL "www.salta.gov.ar/organismos/funcionarios/gobernador-gobernacion/1" and a description: "Información referida a Gobernador de Gobernación de Salta, Dr. Juan Manuel ... Salta, 2015-2019; Gobernador de la Provincia de Salta 2011-2015; Gobernador de la ... 15/11/2017 Macri y Urtubey recorrieron las obras de agua y cloaca que ...". The second is "Gobernación de Salta, Gobierno de la Provincia de Salta, Argentina" with the URL "www.salta.gov.ar/organismos/gobernacion/1" and a description: "El Gobernador, mediante decreto, promulga las leyes de la Provincia disponiendo su publicación en la forma en que lo determine la reglamentación. Puede ...".

FIGURA 8 - FRAGMENTO DESTACADO DE GOOGLE

### 3.3 Alternativas tecnológicas

Para desarrollar este trabajo se analizaron distintas alternativas tecnológicas que se muestran a continuación. En la sección 4.2 se detalla la alternativa escogida y se justifica su elección.

- Lenguajes de programación del lado del servidor
  - Node.js
  - Java
  - PHP
- Lenguajes de programación/Frameworks del lado del cliente

- HTML
  - Javascript
  - JQuery
  - Angular
- Base de datos relacional
  - MySQL
  - SQL Server
  - PostgreSQL
- Frameworks y librerías para procesamiento de información textual
  - UIMA
  - SOLR
  - FreeLing
  - ExtrHech
  - NLTK
- Entornos de trabajo
  - SublimeText
  - PhpStorm
  - Visual Code
- Sistemas Operativos
  - Windows
  - GNU/Linux
  - MacOS
- Servidores web
  - Apache
  - Nginx
- Control de versiones
  - Git
  - SVN

## 4. Propuesta

### 4.1 Solución propuesta

Se propone como solución el desarrollo de un SBR llamado **SBRs** que sea capaz de responder preguntas sobre la Provincia de Salta hechas por el usuario en lenguaje natural, devolviendo resultados que respondan a la pregunta directamente.

Además, el sistema devolverá una serie de resultados los cuales contendrán, cada uno de ellos, el título de la página de donde se extrajo la respuesta, el subtítulo, la oración que responde la pregunta y el link a la página de origen, por si el usuario desea profundizar su búsqueda.

La interfaz será amigable y permitirá al usuario realizar la búsqueda desde un cualquier tipo de dispositivo con conexión a internet.

### 4.2 Herramientas utilizadas

#### Ubuntu

Ubuntu es una distribución de GNU/Linux basada en Debian. Proporciona un sistema operativo actualizado y estable para el usuario promedio, con un fuerte enfoque en la facilidad de uso y de instalación del sistema. Al igual que otras distribuciones se compone de múltiples paquetes de aplicaciones normalmente bajo una licencia libre o de código abierto (Ubuntu).

Se optó por una distribución GNU/Linux por sobre Windows y MacOS debido a la flexibilidad que el sistema provee y a que se trata de software libre.

Dicha distribución será instalada, en vez de usarla como un CD Vivo (*Live CD*, en inglés), ya que de esta manera los cambios permanecerán en el tiempo.

#### Apache

Apache es un proyecto de la Fundación de Software Apache<sup>5</sup>, con el objetivo de suministrar un servidor seguro, eficiente y extensible que proporcione servicios HTTP en sincronía con los estándares HTTP actuales (Apache).

Se escoge Apache sobre Nginx debido a la madurez del proyecto y facilidad de instalación e implementación.

---

<sup>5</sup><http://www.apache.org/>

## MySQL

MySQL es un sistema de gestión de base de datos relacional (RDBMS) de código abierto, basado en el lenguaje de consulta estructurado (SQL). Se ejecuta en prácticamente todas las plataformas, incluyendo GNU/Linux, UNIX y Windows (Rouse, 2015).

De entre todas las alternativas, se eligió MySQL debido a su facilidad de instalación, uso y perfecta integración con PHP/Laravel.

## FreeLing

FreeLing es una librería de código abierto para el procesamiento multilingüe automático, que proporciona una amplia gama de servicios de análisis lingüístico para diversos idiomas (Padró, 2012). En la presente solución se lo utiliza para obtener, a partir de un texto plano, cada palabra o símbolo analizado morfológicamente, una por línea y con una línea blanca que indique el fin de una oración.

## Apache Lucene

Lucene es un software de recuperación de información (RI) de código abierto. Es una API flexible que permite añadir capacidades de indexación y búsqueda a cualquier sistema que se esté desarrollando (Apache Lucene).

## Apache Solr

Solr es una plataforma de búsquedas basada en Apache Lucene, que funciona como un "servidor de búsquedas". Sus principales características incluyen búsquedas de texto completo, resaltado de resultados, *clustering* dinámico, y manejo de documentos ricos (como Word y PDF). Solr es escalable, permitiendo realizar búsquedas distribuidas y replicación de índices, y actualmente se está usando en muchos de los sitios más grandes de Internet (Seta, 2010).

Se escogió Solr ya que provee una interfaz más amigable para interactuar con el índice de Lucene, a diferencia de UIMA, en el que se deben implementar manualmente los distintos módulos.

Cabe destacar que la unidad de indexación de Solr es llamada *Documento*. En el presente trabajo, un Documento se conforma de una tripleta basada en un verbo y otros atributos relacionados a la oración de donde se extrajo la tripleta.



## PHP

PHP (acrónimo recursivo de PHP: *Hypertext Preprocessor*) es un lenguaje de código abierto muy popular especialmente adecuado para el desarrollo web y que puede ser incrustado en HTML (The PHP Group, 2017).

Se eligió este lenguaje ya que el autor de este trabajo posee experiencia previa y además existen una gran cantidad de librerías de código abierto que facilitan el trabajo.

## Laravel

Laravel es un framework de código abierto para el desarrollo de aplicaciones web en PHP que posee una sintaxis simple, expresiva y elegante. Fue creado en 2011 por Taylor Otwell, inspirándose en Ruby on Rails y Symfony, de los cuales ha adoptado sus principales ventajas (Gallego, 2017).

## Guzzle

Guzzle es un cliente HTTP para PHP que permite facilita el envío de peticiones HTTP (Dowling, 2017).

## Goutte

Goutte es una librería para PHP que realiza WS, construido sobre Guzzle (FriendsOfPHP).

## Solarium

Solarium es un cliente de Solr en PHP que modeliza con precisión los conceptos de Solr, además de permitir la comunicación con éste<sup>6</sup>.

## Angular

Angular es un framework de desarrollo para JavaScript creado por Google. La finalidad de Angular es facilitarnos el desarrollo de aplicaciones web SPA<sup>7</sup> y además darnos herramientas para trabajar con los elementos de una web de una manera más sencilla y óptima (Robles, 2017).

Se escoge Angular debido a que integra fácilmente HTML y Javascript, permitiendo el desarrollo de una interfaz para el cliente independiente del servidor.

---

<sup>6</sup> <https://github.com/solariumphp/solarium>

<sup>7</sup> SPA: Single Page Application, aplicación de una sola página.

## Covalent

Covalent es una plataforma desarrollada en Angular que suma ciertos componentes a Angular Material<sup>8</sup>, los cuales siguen las directrices de diseño de *Material Design*<sup>9</sup>.

Se elige esta plataforma debido a que provee una manera fácil y rápida de comenzar un proyecto en Angular, además de componentes útiles.

## PhpStorm

PhpStorm es un entorno de trabajo que provee una integración perfecta con el lenguaje de programación PHP, además de permitir trabajar con otros lenguajes tales como HTML, Javascript, CSS, etc.

Se escoge PhpStorm debido a que provee un verdadero entorno de trabajo, más que un simple editor de texto tal como Sublime Text, lo cual facilita el desarrollo.

## Git

Git es un sistema de control de versiones, que registra los cambios realizados sobre un archivo o conjunto de archivos a lo largo del tiempo, de modo que se pueda recuperar versiones específicas más adelante<sup>10</sup>.

Se elige Git debido a que la gran mayoría de librerías están disponibles en repositorios Git y a su facilidad de uso.

## 4.3 Metodología

El desarrollo del proyecto se realizará siguiendo la metodología CRISP-DM<sup>11</sup> (*Cross Industry Standard Process for Data Mining*) (Chapman, y otros, 2000), que proporciona una descripción normalizada del ciclo de vida de un proyecto estándar de análisis de datos, de forma análoga a como se hace en la ingeniería del software con los modelos de ciclo de vida de desarrollo de software. El modelo CRISP-DM cubre las fases de un proyecto, sus tareas respectivas y las relaciones entre estas tareas.

El ciclo de vida del proyecto de minería de datos consiste en seis fases mostradas en la Figura 9. La secuencia de las fases no es rígida: se permite movimiento hacia adelante y hacia

---

<sup>8</sup> <https://material.angular.io/>

<sup>9</sup> <https://material.io/guidelines/>

<sup>10</sup> <https://git-scm.com/book/es/v1/Empezando-Acerca-del-control-de-versiones>

<sup>11</sup> <http://crisp-dm.eu/>

atrás en diferentes fases. El resultado de cada fase determina qué fase, o que tarea particular de una fase, hay que hacer después. Las flechas indican las dependencias más importantes y frecuentes (Román, 2016).

Las seis fases son:

- Fase 1, Comprensión del negocio: Se enfoca en la comprensión de los objetivos del proyecto. Después se convierte este conocimiento de los datos en la definición de un problema de minería de datos y en un plan preliminar diseñado para alcanzar los objetivos.
- Fase 2, Estudio y comprensión de los datos: Comienza con la colección de datos inicial y continúa con las actividades que permiten familiarizarse con los datos, identificar los problemas de calidad, descubrir conocimiento preliminar sobre los datos, y/o descubrir subconjuntos interesantes para formar hipótesis en cuanto a la información oculta.
- Fase 3, Análisis de los datos y selección de características: Cubre todas las actividades necesarias para construir el conjunto final de datos (los datos que se utilizarán en las herramientas de modelado) a partir de los datos brutos iniciales. Las tareas incluyen la selección de tablas, registros y atributos, así como la transformación y la limpieza de datos para las herramientas que modelan.
- Fase 4, Modelado: Se seleccionan y aplican las técnicas de modelado que sean pertinentes al problema (cuanto más mejor), y se calibran sus parámetros a valores óptimos. Típicamente hay varias técnicas para el mismo tipo de problema de minería de datos. Algunas técnicas tienen requerimientos específicos sobre la forma de los datos. Por lo tanto, casi siempre en cualquier proyecto se acaba volviendo a la fase de preparación de datos.
- Fase 5, Obtención de resultados: En esta etapa del proyecto, se han construido uno o varios modelos que parecen alcanzar calidad suficiente desde una perspectiva de análisis de datos. Antes de proceder al despliegue final del modelo, es importante evaluarlo a fondo y revisar los pasos ejecutados para crearlo, comparar el modelo obtenido con los objetivos de negocio. Un objetivo clave es determinar si hay una cuestión importante de negocio que no haya sido considerada suficientemente. Al final de esta fase, se debería obtener una decisión sobre la aplicación de los resultados del proceso de análisis de datos.

- Fase 6, Despliegue: Generalmente, la creación del modelo no es el final del proyecto. Incluso si el objetivo del modelo es de aumentar el conocimiento de los datos, el conocimiento obtenido tendrá que organizarse y presentarse para que el cliente pueda usarlo. Dependiendo de los requisitos, la fase de desarrollo puede ser tan simple como la generación de un informe o tan compleja como la realización periódica y quizás automatizada de un proceso de análisis de datos en la organización.

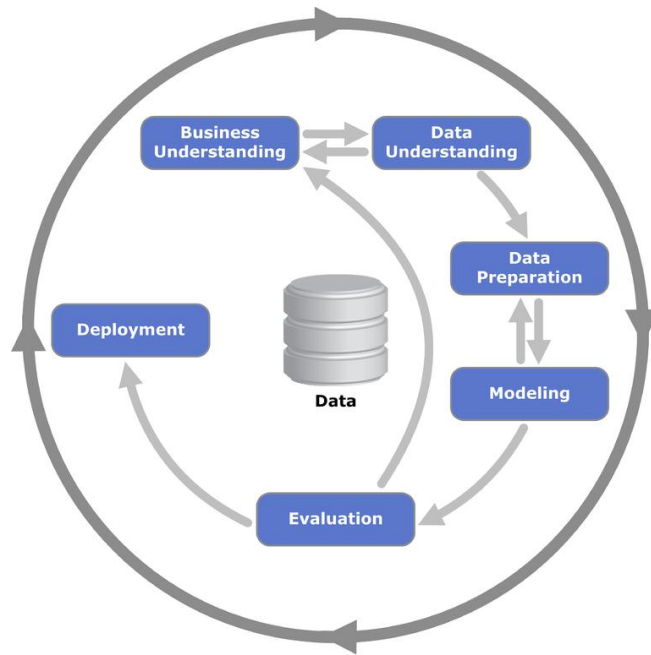


FIGURA 9 - FASES CRISP-DM

Las distintas fases de la metodología se fueron realizando iterativamente como se describe a continuación.

### Fase 1: Comprensión del negocio

Realizando un estudio de dos de los principales sitios oficiales del Gobierno de la Provincia de Salta: [www.turismo.salta.gov.ar](http://www.turismo.salta.gov.ar) y [www.salta.gov.ar](http://www.salta.gov.ar), se puede observar que estos carecen de un buscador. La información que poseen está presente en forma de texto plano distribuida en distintas páginas, muchas de ellas son noticias. Cada página incluye, generalmente, un título y un subtítulo. En el caso de [www.turismo.salta.gov.ar](http://www.turismo.salta.gov.ar), las noticias poseen como título la fecha de publicación de la misma.

### Fase 2: Estudio y comprensión de los datos

Los datos que se encuentran en los distintos sitios webs son datos no estructurados, es decir, texto plano que debe ser tratado para poder ser indexado en una base de conocimiento.

Además el volumen es inmenso y muchas veces poseen textos innecesarios que se deben identificar y eliminar para no entorpecer la base de conocimiento, como el de otros artículos que rodean al central.

Además se puede incorporar información obtenida de Wikipedia, que también la presenta en forma de texto plano, dividido en secciones y subsecciones. Luego de un análisis con distintas preguntas, pude observar que Wikipedia aporta información concreta de buena calidad, a pesar de ser una enciclopedia mantenida por personas comunes y corrientes en su mayoría: hechos históricos, datos sobre clima, ubicación, límites, rutas de acceso, etc., aunque incluye información de cualquier ámbito, no sólo de la Provincia de Salta. Por lo tanto es importante identificar y utilizar sólo aquellas páginas que estén relacionadas a la Provincia de Salta.

Los sitios oficiales aportan información que no se puede encontrar en Wikipedia, tales como determinadas fiestas en distintos pueblos de la Provincia, personajes históricos, etc., aunque también presentan muchas noticias, las cuales pueden aportar datos no informativos.

### Fase 3: Análisis de los datos y selección de características

De las páginas que se visiten se analiza el HTML que las constituye para determinar el título de la página, subtítulo y texto en idioma castellano, y almacenarlas sólo si están relacionadas a la Provincia de Salta. Se descartan imágenes, tablas, listas, y todo aquello que no sea texto contenido en párrafos, tales como menús, descripciones breves de otros artículos que suelen rodear al artículo central, etc.

### Fase 4 y 5: Modelado y Evaluación

Estas dos fases son iterativas y casi simultáneas. Durante el desarrollo del proyecto se probaron varias alternativas, hasta llegar a la solución propuesta.

Inicialmente se probó ExtrHech (Zhila & Gelbukh, 2013) para la extracción de tripletas, pero debido a su naturaleza, que no es más que una adaptación de ReVerb (Fader, Soderland, & Etzioni) para el idioma español. La calidad de las tripletas que logra extraer no es muy buena. Por ejemplo, dado el siguiente texto extraído de la página de La Ciudad de Salta en Wikipedia:

Salta es una ciudad, un municipio y la capital de la provincia de Salta, República Argentina, que tiene una población de 535 303 habitantes, siendo la ciudad más poblada de la provincia y la séptima del país. Su área metropolitana, denominada Gran Salta,[1] está compuesta por once municipios y tiene a una población de 554 125 habitantes (INDEC, 2010). Se encuentra ubicada al este de la cordillera de los Andes, en el Valle de Lerma, a 1187 msnm (3894 pies) y cruzada por el por el río Arenales que la divide en centro, norte y sur.
--

Se ejecuta el siguiente script, que en primer lugar solicita el análisis a FreeLing y luego ejecuta el script de ExtrHech:

```
$ analyze -f /usr/share/freeling/config/es-ar.cfg --usr --fmap user_map_file.conf --ner --outlv "tagged" <
corpus/ciudaddesalta.txt > corpus/ciudaddesalta.txt.pos0
$ python parse_pos_file.py corpus/ciudaddesalta.txt.pos0
```

El cual imprime en una terminal de GNU/Linux el resultado en forma de tripletas que se puede ver en la Tabla 3. Estas no son del todo representativas y por lo tanto se descarta esta herramienta.

Arg1	Rel	Arg2
Salta	es	es una ciudad ,
	tiene	tiene
	siendo	siendo
	poblada	poblada
	denominada está compuesta tiene	denominada está compuesta tiene
	encuentra ubicada	encuentra ubicada
	cruzada	cruzada
	divide	divide

TABLA 3 - TRIPLETAS QUE LOGRA EXTRAER EXTRHECH

Otra herramienta que se probó es NLTK (*Natural Language Toolkit*)<sup>12</sup>, que es un conjunto de librerías para PLN escrito en Python. Se instala fácilmente en una distribución de GNU/Linux usando *pip*:

```
$ sudo pip install -U nltk
```

El script que se muestra en la Figura 10 logra extraer las relaciones basándose en una simple expresión regular. La Figura 11 muestra gráficamente una primera aproximación y la Figura 12 la tripleta que se podría extraer.

En la solución que se propone en este proyecto se optó por extraer Documentos que poseen una tripleta basada en un verbo, entre otros atributos que le dan un contexto a la tripleta. Los argumentos de una tripleta se determinan usando una expresión regular diferente a las que se usan en QARelaciones y ExtrHech. El modelado se explica en el capítulo 4.4.2 y la evaluación en el capítulo 5.

---

<sup>12</sup><http://www.nltk.org/>

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import os
import sys
import nltk

def main(argv):
    # print [("Salta"), ("es"), ("una ciudad, un municipio y la capital de la provincia de Salta, República Argentina")]
    # print "\n\n\n"
    if len(argv) != 2:
        sys.stderr.write("Usage: %s POStagged_file OUT_File \n" % argv[0])
        sys.exit(-1)
    file = argv[1]
    sentence = []
    emptyLine = 0
    for s in open(file):
        if s.strip() == "":
            emptyLine += 1
            continue
        ws = s.strip().split()
        word = (ws[0].decode('utf-8'), ws[2][:2])
        sentence.append(word)
    chunk(sentence)
def chunk(sentence):
    grammar = """
    VERB: {<V.> *}
    NP:
        {<CC>? <Fc>? <D.>? <*>? <N.> *}
    <NN.>? {<.> *}
    <*> {<D.> *}
    <N.> {<N.> *}
        {<N.> *}
        {<P.> *}
        {<N.>? <Fc>? <CC>? <N.> *}
    VERBPHRASE:
        {<NP> <VERB> <NP>}
    """
    cp = nltk.RegexpParser(grammar)
    result = cp.parse(sentence)
    for s in result:
        isTuple = 0
        rel = ["", "", ""]
        argumentCounter = 0
        verbFound = 0
        arg1 = ""
        arg2 = ""
        for t in s:
            if type(t) is nltk.tree.Tree:
                isTuple = 1
                if t.label() == "VERB":
                    verbFound = 1
                rel[1] += getNodes(t)
                if verbFound == 1:
                    arg1 += getNodes(t)
                else:
                    arg2 += getNodes(t)
            if isTuple:
                rel[0] = arg1
                rel[2] = arg2
                print rel
                print '\n\n\n'
def getNodes(tree):
    words = ""
    for node in tree:
        if type(node) is nltk.tree.Tree:
            getNodes(node)
        else:
            words += node[0]
            words += " "
    # words += tree[0]
    # words += " "
    return words
if __name__ == "__main__":
    main(sys.argv)
```

FIGURA 10 - SCRIPT DE NLTK

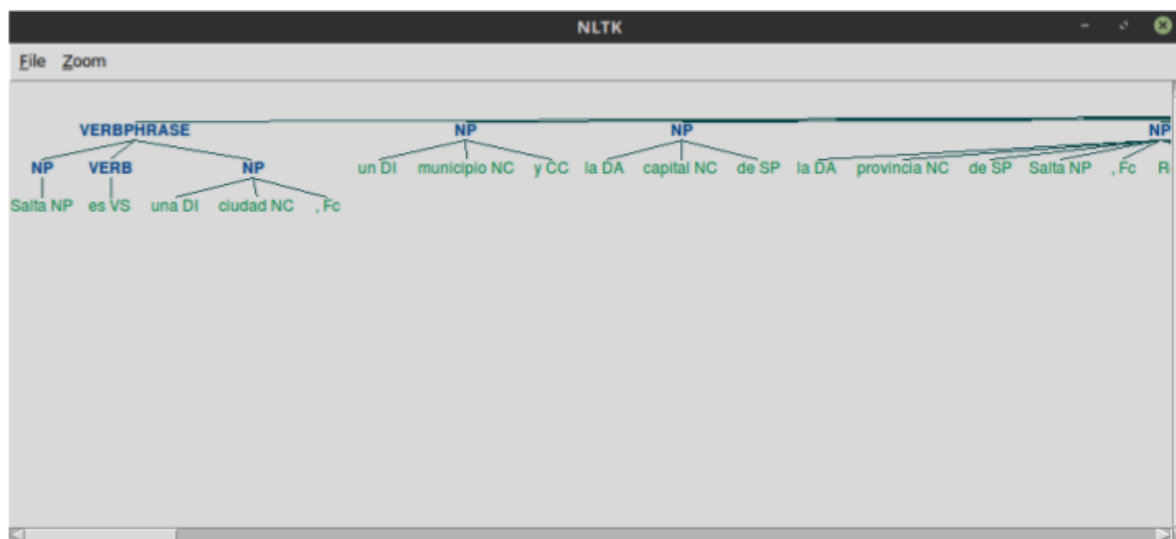


FIGURA 11 - VISUALIZACIÓN GRÁFICA DEL RESULTADO DE NLTK

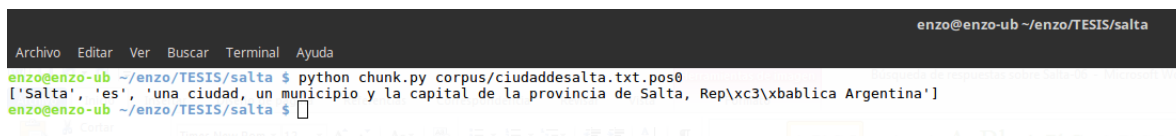


FIGURA 12 - RESULTADO DE NLTK

## Fase 6: Despliegue

El despliegue de la solución propuesta se puede realizar utilizando la imagen<sup>13</sup> que se adjunta en el proyecto, la cual posee las herramientas necesarias, instaladas y configuradas de tal manera que sólo es necesario volcarla a un servidor. Además en el Anexo 5 se indican los pasos para instalar y configurar todo desde cero.

## 4.4 Arquitectura

El sistema se apoyará sobre un proyecto de Laravel<sup>14</sup> que se comunicará con una instancia de Solr, quien será el encargado de armar el índice de Lucene y realizar las consultas, y FreeLing, todo corriendo sobre Ubuntu 16.04 y un servidor Apache. De esta manera se logra implementar absolutamente todo el sistema sobre tecnologías web. Gráficamente se puede observar la arquitectura en la Figura 1.

La instalación y configuración de las distintas herramientas se detallan en el Anexo 5.

<sup>13</sup> <https://github.com/enzonotario/sbrs>

<sup>14</sup> <https://github.com/enzonotario/sbrs-api>



#### 4.4.1 Obtención de textos de la web

La base de conocimiento del SBR se construyó a partir de un corpus formado de textos extraídos de distintos sitios webs, para lo cual fue necesario desarrollar distintos WC que aplican WS con *Goutte* en cada página que visitan. Los sitios seleccionados son dos oficiales: [www.salta.gov.ar](http://www.salta.gov.ar) y [www.turismo.salta.gov.ar](http://www.turismo.salta.gov.ar), y Wikipedia: [www.wikipedia.org](http://www.wikipedia.org).

Los distintos WC se han implementado en la Clase *Crawler* ubicada en *app/Crawler.php* y cuyo código se muestra en la Figura 39.

#### Extracción de texto del sitio [www.turismo.salta.gov.ar](http://www.turismo.salta.gov.ar)

Para la extracción de texto del sitio [www.turismo.salta.gov.ar](http://www.turismo.salta.gov.ar) se ha programado un WC que visita las distintas páginas siguiendo un patrón identificado en la estructura del sitio, en vez de comenzar por una URL e ir visitando los hiperenlaces que se identifiquen en cada página.

El patrón identificado surgió de una simple inspección del sitio: cada URL está formada por el nombre de dominio, seguido de: */contenido/{id}/{titulo}*. Por ejemplo:

<http://turismo.salta.gov.ar/contenido/1/geografia-salta>

<http://turismo.salta.gov.ar/contenido/2/clima-salta>

<http://turismo.salta.gov.ar/contenido/4/historia-salta>

...

<http://turismo.salta.gov.ar/contenido/contenido/{id}/{titulo}>

Para realizar esta tarea se creó un *Comando* en Laravel ejecutando el siguiente comando en una terminal GNU/Linux:

```
$ php artisan make:command CrawlerCommand
```

El cual crea un archivo en *app/Console/Commands/CrawlerCommand.php* y contiene el código que se muestra en la Figura 38.

Este comando se ejecuta por medio de una terminal de GNU/Linux con el comando *php artisan crawler*, seguido del sitio al que se desea realizar WC como parámetro:

```
$ php artisan crawler turismo.salta.gov.ar
```

El WC comienza visitando la página con *id* 1 y lo incrementa consecutivamente para obtener las demás páginas, hasta que visita 500 páginas consecutivas que no existan. Esto le indica al WC que más allá de cierto *id* no existen más páginas, ya que es imposible conocer de antemano la cantidad de páginas exactas, teniendo en cuenta que el sitio es dinámico y cambia día a día.

En cada visita extrae y almacena en variables el título, subtítulo, tipo de artículo, la descripción y el contenido del artículo a través de selectores CSS para luego escribirlo en un archivo JSON con el formato que se muestra en la Figura 13.

Cabe destacar que en algunas de estas páginas, el título es la fecha de la publicación, por lo que la interfaz de usuario podría mostrar la fecha de la publicación como título, acompañado de los demás atributos.

```
{
  "id": "...",
  "url": "...",
  "page": "...",
  "title": "...",
  "subtitle": "...",
  "text": [
    "...",
    "...",
    "...",
    "..."
  ]
}
```

FIGURA 13 - FORMATO DE LOS ARCHIVOS JSON EXTRAÍDOS DE TURISMO.SALTA.GOV.AR

### Extracción de texto del sitio [www.salta.gov.ar](http://www.salta.gov.ar)

La extracción de información del sitio [www.salta.gov.ar](http://www.salta.gov.ar) se hizo de manera similar a la del sitio [www.turismo.salta.gov.ar](http://www.turismo.salta.gov.ar). El patrón identificado en éste sitio es el siguiente:

<http://www.salta.gov.ar/contenidos/crawl/1>

<http://www.salta.gov.ar/contenidos/crawl/2>

...

<http://www.salta.gov.ar/contenidos/crawl/{id}>

El código se muestra en la función *saltaGovAr* de *CrawlerCommand* y se ejecuta, a través de una terminal en GNU/Linux, con el siguiente comando:

```
$ php artisan crawler salta.gov.ar
```

El WC comienza visitando la página con *id* 1 y termina cuando no encuentra 500 páginas consecutivas. Los archivos JSON que se almacenan tienen el formato que se muestra en la Figura 14.

```
{
  "id": "...",
  "url": "...",
  "page": "...",
  "title": "...",
  "subtitle": "...",
  "type": "...",
  "text": [
    "...",
    "...",
    "...",
    "..."
  ]
}
```

FIGURA 14 - FORMATO DE LOS ARCHIVOS JSON EXTRAÍDOS DE SALTA.GOV.AR

### Extracción de texto del sitio [www.wikipedia.org](http://www.wikipedia.org)

El WC para la extracción de texto de éste sitio comienza visitando la página de la Provincia de Salta en Wikipedia ([https://es.wikipedia.org/wiki/Provincia\\_de\\_Salta](https://es.wikipedia.org/wiki/Provincia_de_Salta)) y agrega a la cola de URLs por visitar todos aquellos hiperenlaces que encuentre y correspondan a direcciones dentro del sitio de Wikipedia.org. En cada visita busca que en todo el texto se encuentre la palabra “Salta”, de ésta manera espera obtener sólo información sobre la Provincia de Salta. Además extrae los títulos, subtítulos y párrafos de cada sección en variables para luego escribirlas en un archivo en formato JSON.

Éste WC se implementó usando *Jobs*, que no son más que Clases que se agregarán a una cola para permitir la ejecución asíncrona y el multiprocesamiento y así optimizar la velocidad de ejecución. El *Job* que implementa el WC se crea ejecutando el siguiente comando en una terminal de GNU/Linux:

```
$ php artisan make:job WikipediaCrawlerJob
```

Lo cual creará el archivo *app/Jobs/WikipediaCrawlerJob.php* y cuyo código se muestra en la Figura 36. Para hacer un uso asíncrono de los *Jobs* se debe ejecutar el siguiente comando en una terminal de GNU/Linux:

```
$ php artisan queue:table
```

Que generará la migración de tablas necesaria para llevar un registro de los diferentes *Jobs* a ejecutar. Además, para procesarlos, se debe ejecutar el siguiente comando en una terminal de GNU/Linux:

```
$ php artisan queue:work
```

Que creará un proceso encargado de procesar los diferentes *Jobs* a medida que se vayan creando.

Debido a que la ejecución de éste WC es asíncrona, se pierde un poco el control que la ejecución síncrona provee, aunque se gana velocidad. Por esto es necesario llevar un registro de las URLs que el sistema ya visitó y de las que ya agregó a la cola de *Jobs* por procesar, para evitar agregar las mismas URLs una y otra vez. Para esto se usó dos tablas en MySQL que se generan a partir de las migraciones que se crean con los siguientes comandos en una terminal de GNU/Linux:

```
$ php artisan make:migration create_visitedSites_table  
$ php artisan make:migration create_queuedSites_table
```

Los códigos de estas migraciones se muestran en la Figura 25 y Figura 26 respectivamente.

Para poder interactuar con estas tablas, se hace uso del *Object Relational Mapping* (ORM) de Laravel llamado *Eloquent*, lo cual implica crear un modelo para cada tabla con los siguientes comandos en una terminal de GNU/Linux:

```
$ php artisan make:model VisitedSites  
$ php artisan make:model QueuedSites
```

Estos Modelos poseen el código que se muestra en la Figura 29 y Figura 30, respectivamente.

El punto de entrada del WC se maneja en *CrawlerCommand*, ejecutando en una terminal de GNU/Linux el siguiente comando:

```
$ php artisan crawler wikipedia.org
```

Este, a su vez, ejecuta la función *wikipediaOrg* de la Clase *Crawler*, la cual realizará WS en cada página que visite (sólo si la página no fue previamente analizada), creará los distintos *Jobs* según los hiperenlaces que encuentre (si aún no fueron creados para estos sitios) y almacenará el contenido de cada página en un archivo JSON con el formato que se muestra en la Figura 15.

Luego de un tiempo, que dependerá de las características de la PC que ejecute los *Jobs*, obtendremos en la carpeta *storage/app/corpus/wikipedia.org* todos los archivos correspondientes a las distintas páginas de Wikipedia.

```
{
  "url": "...",
  "name": "...",
  "sections": [
    {
      "title": "...",
      "subsections": [
        {
          "subtitle": "",
          "paragraphs": [
            "...",
            "...",
            "..."
          ]
        }
      ]
    }
  ],
  {
    "title": "...",
    "subsections": [
      {
        "subtitle": "",
        "paragraphs": [
          "...",
          "...",
          "..."
        ]
      }
    ]
  }
],
  {
    "title": "...",
    "subsections": [
      {
        "subtitle": "",
        "paragraphs": [
          "...",
          "...",
          "..."
        ]
      }
    ]
  }
]
}
```

FIGURA 15 - FORMATO DE LOS ARCHIVOS JSON EXTRAÍDOS DE WIKIPEDIA.ORG

### Obtención de sinónimos de verbos

Cuando el usuario final realiza una pregunta en lenguaje natural, ésta es transformada y todos los verbos se reemplazan por su equivalente en infinitivo. Puede darse el caso en el que los Documentos indexados en realidad contengan un sinónimo del/los verbo/s que conforma/n la pregunta. Por este motivo, al momento de extraer los Documentos, también se almacena en un atributo los sinónimos de los verbos que conforman el atributo *rel* del Documento. Los sinónimos se obtienen de dos sitios diferentes: <http://sinonimos.woxikon.es/es/> y <http://www.sinonimos.org/>.

El WS se implementó en la Clase *Synonymous*, cuyo código se muestra en la Figura 40, y es la encargada de obtener los sinónimos de los dos sitios y devolverlos. Además los guarda en la base de datos para que la próxima vez que se necesiten los sinónimos de un verbo previamente analizado, se evite realizar de nuevo el WS y así mejorar la velocidad. Para esto es necesario generar el Modelo *Synonym* con el siguiente comando en una terminal de GNU/Linux:

```
$ php artisan make:model Synonym
```

El cual creará el archivo *app/Synonym.php* y cuyo código se muestra en la Figura 31. También es necesario generar la Migración que creará la tabla en la base de datos con el siguiente comando en una terminal de GNU/Linux:

```
$ php artisan make:migration create_synonymous_table
```

El cual creará un archivo con una marca de tiempo seguido del nombre de la tabla en *database/migrations* y debe contener el código que se muestra en la Figura 27.

#### 4.4.2 Extracción de información

La extracción de información se hizo siguiendo la idea básica de OIE: obtener tripletas basadas en un verbo, pero almacenando también todos los sustantivos propios y comunes que se obtiene de cada oración analizada, entre otros atributos, lo cual conforma un *Documento*.

El extractor se construyó en la Clase *Extractor* que se encuentra *app/Extractor.php*, y cuyo código se muestra en la Figura 41.

La acción de extraer los Documentos de los distintos sitios se realiza mediante un Comando de Laravel que se crea ejecutando:

```
$ php artisan make:command ExtractorCommand
```

*ExtractorCommand* contiene el código que se muestra en la Figura 37, y recibe como parámetro el sitio del cual se debe proceder a extraer los Documentos. Éste, a su vez, crea distintos *Jobs*. El *Job* se crea ejecutando el siguiente comando en una terminal de GNU/Linux:

```
$ php artisan make:job ExtractorJob
```

Y contiene el código que se muestra en la Figura 35.

Como se hace uso de FreeLing, es necesario ejecutarlo en modo cliente/servidor para que sea capaz de comunicarse a través de un *socket* con la clase *Extractor* de Laravel. Para esto ejecutamos el siguiente comando en una terminal de GNU/Linux:

```
$ analyze -f /usr/share/freeling/config/es-ar.cfg --usr --fmap user_map_file.conf --server --port 50005
```

Con lo cual se obtiene un servidor de FreeLing escuchando en el puerto 50005. Se debe tener en cuenta que en el directorio de trabajo se debe encontrar el archivo *user\_map\_file.conf* que define las etiquetas específicas a utilizar para ciertas palabras. En este caso en particular es necesario que el archivo contenga lo siguiente:

Salta salta NP00000
---------------------

Para que la palabra “Salta” sea considerada como un sustantivo propio y no un verbo, que es la clasificación original que FreeLing asigna a esta palabra.

A medida que la cola vaya procesando los diferentes *Jobs* que se crearon, cada uno de ellos leerá un archivo del corpus del sitio correspondiente y los párrafos, que pueden estar contenido dentro de una sección (como en el caso de wikipedia.org) o directamente en la misma página (como en el caso de turismo.salta.gov.ar y salta.gov.ar). Estos párrafos contienen el texto no estructurado que se obtuvo al aplicar WS en los distintos sitios. De cada uno se extraerán una serie de Documentos para ser indexados en Solr y así formar la base de conocimiento.

Por cada párrafo, *ExtractorJob* ejecuta el método *fromText* de la Clase *Extractor*, el cual solicita el análisis a FreeLing a través del método *analyzeText*, que devuelve como resultado una línea por cada palabra o símbolo, con su lema y etiqueta EAGLES, y una línea vacía que indica el fin de una oración. De éste resultado se obtienen las distintas oraciones, a través del método *getSentencesFromAnalyzedText* (teniendo en cuenta que cada oración está separada por una línea vacía), y cada una de ellas se lee línea por línea hasta que encuentra un verbo que no se encuentre dentro de paréntesis o corchetes. Cuando lo encuentra, almacena todo lo anterior de la oración en el atributo *arg1* del Documento, y el verbo que encontró en el atributo *rel* del Documento, pero continúa leyendo cada línea en busca de verbos consecutivos (ya que existen frases verbales compuestas por más de una palabra, como por ejemplo: “*está compuesto*”). Si encuentra otro verbo consecutivo, lo agrega al atributo *rel* del Documento, hasta que no encuentre otro verbo consecutivo. En tal caso, las palabras o símbolos a partir de esa posición y hasta el final de la oración, se almacenan en el atributo *arg2* del Documento.

Esto se puede resumir con la siguiente expresión regular:

$W^* V^+ W^+$
donde: W representa cualquier palabra o símbolo (incluido un verbo) que no se encuentre dentro de paréntesis o corchetes V representa un verbo * representa repetición de cero o más veces + representa repetición de una o más veces

El extractor lee absolutamente todas las líneas que devuelve FreeLing como resultado, y almacena en el atributo *nps* del Documento todos los sustantivos propios que encuentra en una misma oración, y en el atributo *ns* del Documento, todos los sustantivos comunes.

Además, por cada párrafo, almacena en una variable el primer sustantivo propio que encuentra para ser usado como atributo *arg1* en los casos en que el verbo de una oración esté al comienzo de la misma (lo que implica un sujeto tácito) y ser almacenado como un sustantivo propio, en el atributo *nps*, de todos los Documentos que se extraen de las oraciones del párrafo (para darle un contexto a cada Documento). Esto es algo que no se contempla en QARelaciones ni en ExtrHech.

Luego se vuelve a analizar el atributo *arg2* de cada Documento como si fuese una oración, ya que podría contener otros verbos y, en tal caso, extraer otros Documentos.

Con el método *cleanText* se limpian paréntesis y corchetes de los atributos *arg1* y *arg2*. Se decidió esta limpieza porque después de múltiples pruebas se verificó que la información contenida dentro de paréntesis y/o corchetes (por ejemplo una fecha) no aportaba mucha información y entorpecía el índice.

Una vez analizado todo un párrafo, se conforman los Documentos que en un principio contienen los siguientes atributos:

- *arg1*
- *rel*
- *arg2*
- *nps*
- *ns*

A cada Documento se le agregan los siguientes atributos, a través del método *formatToSolr*, que, sumado los anteriores, son los que Solr espera obtener:



- URL
- Nombre de la página
- Título de la sección del párrafo o de la página
- Subtítulo de la sección del párrafo o de la página
- Sitio web del cual fue extraído el párrafo
- Oración original
- Lista de verbos en infinitivo que forman el atributo *rel* del Documento
- Lista de sinónimos de verbos que forman el atributo *rel* del Documento

Finalmente, con el método *exportToSolr*, se realiza una petición HTTP a la API de Solr enviándole todos los Documentos que se obtuvieron de la extracción para que los *indexee* y así se forme la base de conocimiento. Cada petición que se realiza es similar a la que se puede ver en la Tabla 4.

También es posible guardar cada Documento extraído en la base de datos si se lo indica a la clase *Extractor* a través del método *withSave*, para tener un respaldo. En tal caso es necesario crear la migración de la tabla Documentos con el siguiente comando en una terminal de GNU/Linux:

```
$ php artisan make:migration create_documents_table
```

La cual debe contener el código que se muestra en la Figura 24, y el modelo con el siguiente comando:

```
$ php artisan make:model Document
```

Que debe contener el código que se muestra en la Figura 28.

En la Figura 16 se muestra el Diagrama de Entidad – Relación que resulta de ejecutar todas las migraciones.

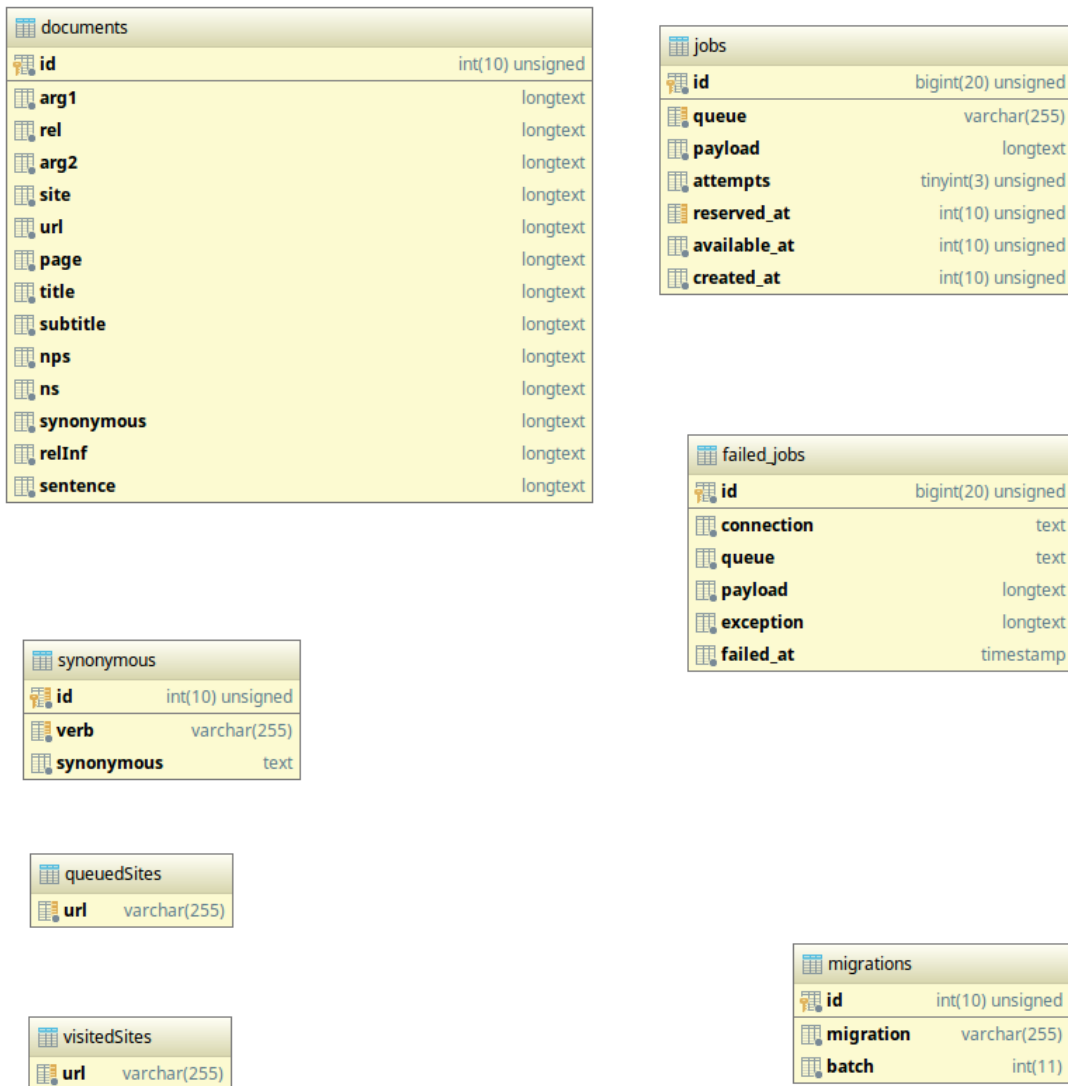


FIGURA 16 - DIAGRAMA DE ENTIDAD-RELACIÓN

Para ilustrar los resultados de éste proceso, se construyó una interfaz que muestra, en formato JSON, los Documentos que se extraen de un determinado archivo del corpus.

Por ejemplo, se puede ver gráficamente en la Figura 17 el primer Documento que se obtiene del archivo JSON que corresponde a la página de la Ciudad de Salta en Wikipedia<sup>15</sup>.

Este Documento es enviado a Solr, realizando la petición que se muestra en la Tabla 4, para que lo *indexee*. La Figura 18 muestra el Documento en el panel de administración de Solr, donde se puede observar, además, que Solr ya le asignó un *id* en el índice.

<sup>15</sup>[https://es.wikipedia.org/wiki/Salta\\_\(ciudad\)](https://es.wikipedia.org/wiki/Salta_(ciudad))

Este proceso se debe realizar cada cierto tiempo, según la precisión que se desee, que dependerá de la frecuencia con que sean actualizados los sitios web. Los WC están diseñados para comenzar a partir de cierto *id* (desde el último que se analizó, por ejemplo) o de buscar páginas que previamente no hayan sido analizadas en algún momento anterior.

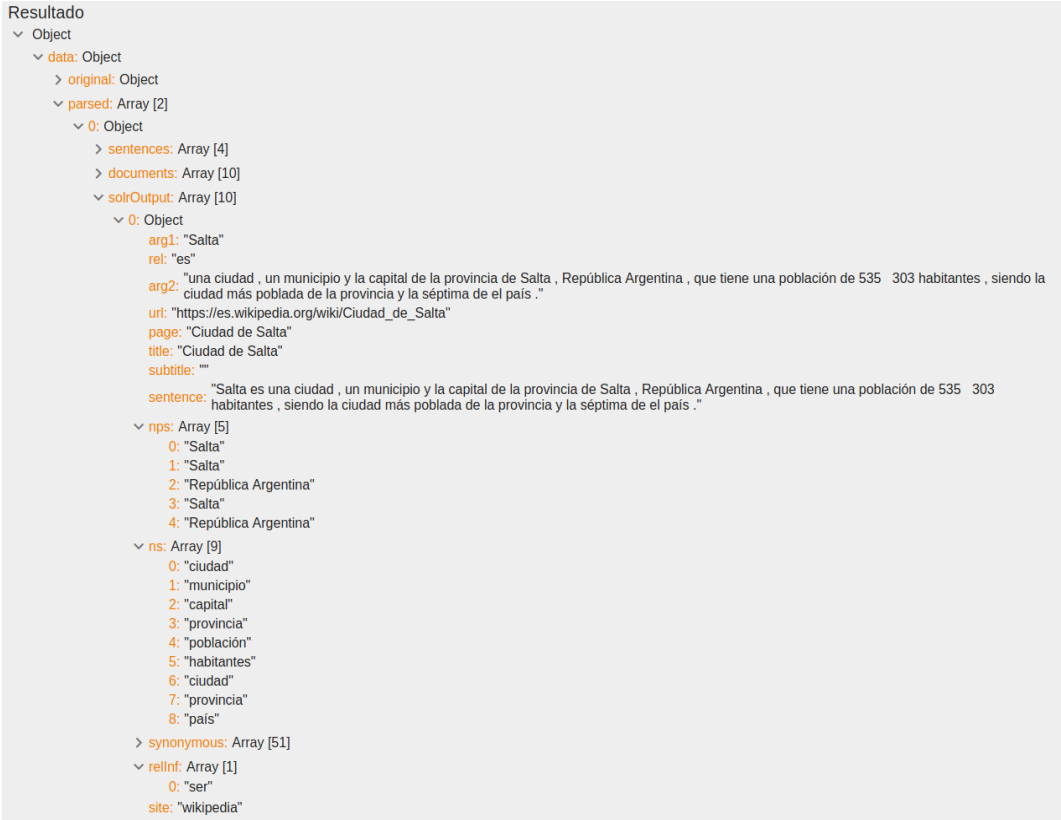


FIGURA 17 - PRIMER DOCUMENTO QUE SE EXTRAE DE LA PÁGINA DE LA CIUDAD DE SALTA EN WIKIPEDIA

Método	POST
Endpoint	<a href="http://localhost:8983/solr/tesis/update?_=1496584152882&amp;boost=1.0&amp;commitWithin=1000&amp;overwrite=true&amp;wt=json">http://localhost:8983/solr/tesis/update?_=1496584152882&amp;boost=1.0&amp;commitWithin=1000&amp;overwrite=true&amp;wt=json</a>
Body	[ { "arg1":"Salta", "rel":"es", "arg2":"una ciudad , un municipio y la capital de la provincia de Salta , Rep\u00fablica Argentina , que tiene una poblaci\u00f3n de 535 \u00a0 303 habitantes , siendo la ciudad m\u00e1s poblada de la provincia y la s\u00e9ptima de el pa\u00eds .", "url":"https://es.wikipedia.org/wiki/Ciudad_de_Salta", "page":"Ciudad de Salta", "title":"Ciudad de Salta", "subtitle":"", "sentence":"Salta es una ciudad , un municipio y la capital de la provincia de Salta , Rep\u00fablica Argentina , que tiene una poblaci\u00f3n de 535 \u00a0 303 habitantes , siendo la ciudad m\u00e1s poblada de la provincia y la s\u00e9ptima de el pa\u00eds .", "nps":[ "Salta", "Salta", "Rep\u00fablica Argentina", "Salta", "Rep\u00fablica Argentina" ], }, ]

	<p>"ns":[  "ciudad",  "municipio",  "capital",  "provincia",  "poblaci\u00f3n",  "habitantes",  "ciudad",  "provincia",  "pa\u00eds"]],  "synonymous":[  "pasar",  "acontecer",  "acaecer",  "suceder",  "ente",  "cosa",  "organismo",  "esp\u00e9cimen",  "individuo",  "sujeto",  "existir",  "vivir",  "quedar",  "haber",  "yacer",  "subsistir",  "naturaleza",  "materia",  "coexistir",  "prevalecer",  "estar",  "hallarse",  "entidad",  "hombre",  "mortal",  "humano",  "permanecer",  "quedarse",  "encontrarse",  "ubicarse",  "substancia",  "virtualidad",  "principio",  "existencia",  "propiedad",  "individualidad",  "unidad",  "germen",  "aborto",  "criatura",  "engendro",  "feto",  "elemento",  "objeto",  "cuerpo",  "persona",  "encarnar",  "figurar",  "simbolizar",  "constituir",  "personificar",  "significar",  "identidad",  "inerencia",  "esencia"]]</p>
--	---

	<pre>],   "relInf": [     "ser"   ],   "site": "wikipedia.org" } ]</pre>
--	--

TABLA 4 - PETICIÓN ENVIADA A SOLR PARA INDEXAR UN DOCUMENTO

```
{
  "responseHeader":{
    "zkConnected":true,
    "status":0,
    "QTime":259,
    "params":{
      "q":"sentence:\\\"Salta es una ciudad , un municipio y la capital de la provincia de Salta , República Argentina\\\"
      \"indent\":\"on\",
      \"wt\":\"json\",
      \"_\":\"1507148960067\"}},
    "response":{"numFound":4,"start":0,"docs":[
      {
        "arg1":"Salta",
        "rel":"es",
        "arg2":"una ciudad , un municipio y la capital de la provincia de Salta , República Argentina , que tiene (
        \"url\":\"https://es.wikipedia.org/wiki/Ciudad_de_Salta\",
        \"page\":\"Ciudad de Salta\",
        \"title\":\"Ciudad de Salta\",
        \"nps\":[\"Salta\",
          \"Salta\",
          \"República Argentina\",
          \"Salta\",
          \"República Argentina\"],
        \"ns\":[\"ciudad\",
          \"municipio\",
          \"capital\",
          \"provincia\",
          \"población\",
          \"habitantes\",
          \"ciudad\",
          \"provincia\",
          \"país\"],
        \"synonymous\":[\"existir\",
          \"vivir\",
          \"quedar\",
          \"haber\",
          \"yacer\",
          \"subsistir\",
          \"ser\",
          \"naturaleza\",
          \"materia\",
          \"pasar\",
          \"acontecer\",
          \"acaecer\",
          \"suceder\",
          \"existir\",
          \"coexistir\",
          \"prevalecer\",
          \"estar\",
          \"hallarse\",
          \"sujeto\",
          \"entidad\",
          \"hombre\",
          \"individuo\",
          \"mortal\",
          \"humano\",
          \"permanecer\",
          \"quedarse\",
          \"encontrarse\",
          \"ubicarse\",
          \"existir\",
          \"ente\",
          \"individualidad\",
          \"unidad\",
          \"germen\",
          \"aborto\",
          \"criatura\",
          \"engendro\",
          \"feto\",
          \"persona\",
          \"encarnar\",
          \"figurar\",
          \"simbolizar\",
          \"constituir\",
          \"personificar\",
          \"significar\",
          \"principio\",
          \"existencia\",
          \"substancia\",
          \"identidad\",
          \"propiedad\",
          \"inherencia\",
          \"esencia\"],
        \"relInf\":[\"ser\"],
        \"sentence\":\"Salta es una ciudad , un municipio y la capital de la provincia de Salta , República Argentina
        \"id\":\"480b3e75-7d28-4702-b1c0-02f57284f7eb\",
        \"_version_\":\"1580304734731894786\"},
```

FIGURA 18 - INDEXACIÓN EN SOLR DEL PRIMER DOCUMENTO QUE SE EXTRAJO DE LA PÁGINA DE LA CIUDAD DE SALTA EN WIKIPEDIA

## 4.5 Buscador de respuestas

El buscador de respuestas se ha construido en base a una API con la cual puede interactuar cualquier interfaz que se desee construir para el usuario final. Las rutas de esta API se muestran en la Figura 23.

Específicamente, la función que se encarga de devolver una respuesta a una pregunta hecha por el usuario es *search*, que se encuentra en el controlador *SearchController*, cuyo código se muestra en la Figura 32 y el *endpoint* es:

Método	Endpoint	Parámetros requeridos
POST	/api/search	"question": "La pregunta a realizar"

Esta función recibe la pregunta realizada en lenguaje natural por el usuario, le quita los signos de interrogación si es que los tiene y, a través de la clase *Extractor*, la analiza con FreeLing para obtener un Documento como se hizo a la hora de extraer los Documentos que forman la base de conocimiento.

De éste Documento se concatenan los atributos *arg1*, la lista de verbo en infinitivo, y *arg2*, lo cual se almacena en la variable *\$questionParsed*.

Una vez obtenida la variable *\$questionParsed*, se realizan tres consultas a Solr a través de Solarium: Una aplicando *Filter Queries*, otra aplicando *Fuzzy Match*, y otra sin aplicar *Fuzzy Match*, con la pregunta original, para complementar las respuestas.

La consulta del tipo *Filter Queries* implica que se busca, entre todo el índice, aquellos Documentos que más coincidan en los mismos atributos *arg1*, *arg2*, *relInf*, *synonymous*, *nps* y *ns*. Las otras dos consultas se realizan sobre todos los atributos de los Documentos.

La consulta del tipo *Fuzzy Match* implica que devolverá como resultado aquellos Documentos que contengan una respuesta cuyas palabras pueden diferir en un determinado número de letras a las de la pregunta original. En este caso se permite que pueda diferir en una letra.

Los resultados de estas tres tipos de consultas son analizados y sólo se retornarán como resultado a la búsqueda si previamente no se retornó otro Documento con la misma oración (ya que los resultados pueden ser distintos Documentos que en realidad corresponden a una misma oración). Además estos se agrupan según su página de origen.

Para ejemplificar este proceso, se muestran a continuación los diferentes pasos que se llevan a cabo al recibir una pregunta en lenguaje natural, por ejemplo: “¿Dónde está el Centro de Convenciones de Salta?”.

Se quitan los signos de interrogación y a través de la Clase *Extractor* se obtiene el primer Documento, como se muestra en la Figura 19. En la variable *\$questionParsed* se almacena la concatenación de los atributos *arg1*, *relInf* y *arg2* del Documento:

```
$questionParsed = "Donde estar el Centro de Convenciones de Salta"
```



FIGURA 19 - DOCUMENTO QUE SE EXTRAE DE UNA PREGUNTA EN LENGUAJE NATURAL HECHA POR EL USUARIO FINAL

Luego se envían las tres tipos consultas a Solr, una con *Filter Queries*, otra con *Fuzzy Match* y otra sin *Fuzzy Match*, y los distintos documentos que Solr devuelva como resultado se analizarán y se añadirán a una variable *\$response* sólo si previamente no fue añadido otro Documento que corresponda a una misma oración. Además, aquellos Documentos que



pertenezcan a la misma página de un Documento previamente agregado, serán agrupados en un mismo resultado.

Finalmente se devuelve la variable  $\$response$  en formato JSON, como se muestra en la Figura 20, para que la interfaz se lo muestre al usuario final, como se puede ver en la Figura 21.

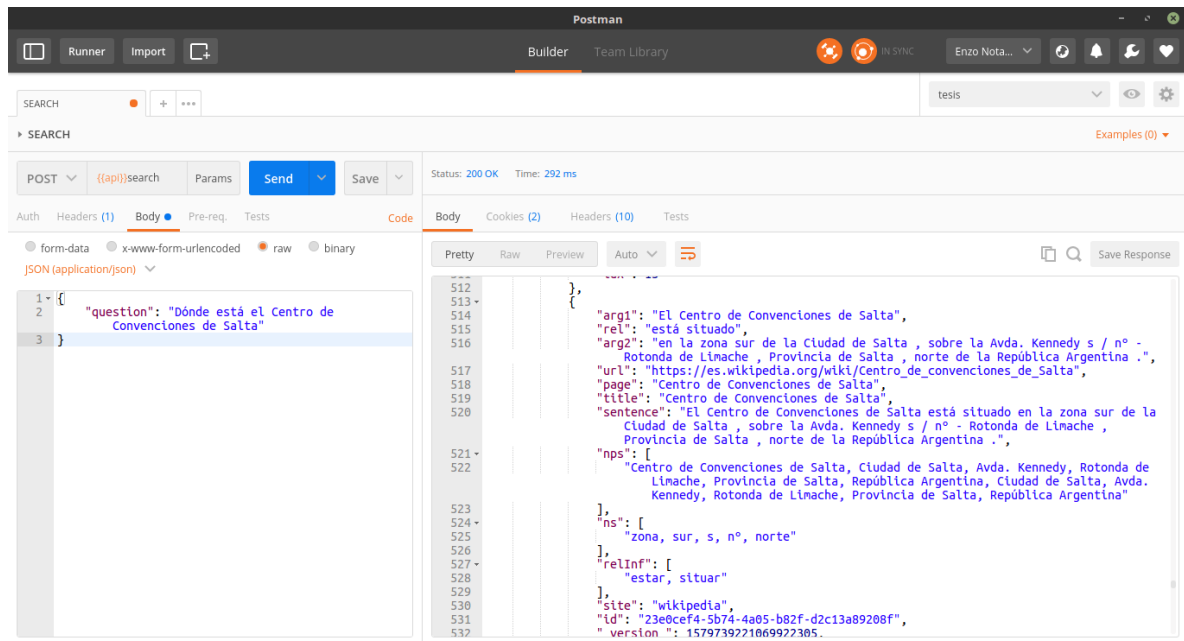


FIGURA 20 - RESPUESTAS EN FORMATO JSON A UNA PREGUNTA REALIZADA EN LENGUAJE NATURAL POR EL USUARIO FINAL

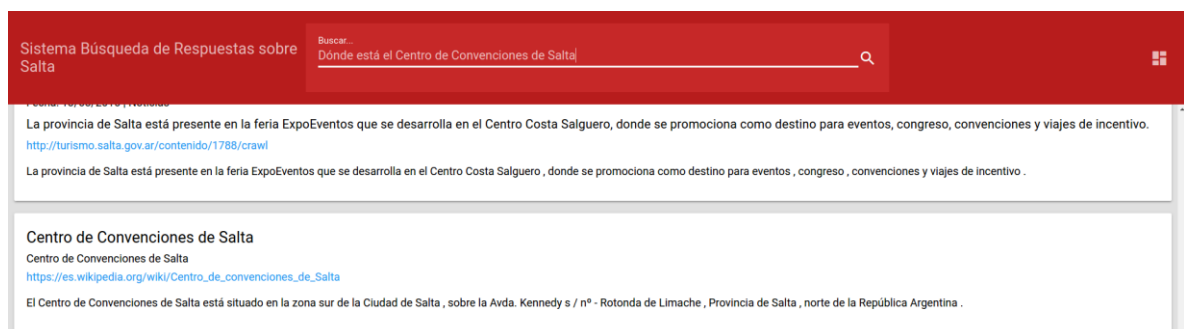


FIGURA 21 - VISUALIZACIÓN EN LA INTERFAZ DE USUARIO DE LAS RESPUESTAS EN FORMATO JSON A UNA PREGUNTA REALIZADA POR EL USUARIO FINAL EN LENGUAJE NATURAL

## Interfaz de usuario

Para que el buscador de respuesta sea fácilmente accesible para cualquier usuario, se ha construido una interfaz web amigable<sup>16</sup> que se comunica con la API construida en Laravel para responder las preguntas realizadas por el usuario en lenguaje natural. Esta interfaz fue construida en Angular 4 utilizando Covalent.

Para comenzar, se debe clonar el repositorio *Covalent Quickstart* en Github<sup>17</sup>, el cual contiene lo necesario para comenzar a programar sobre la plataforma Covalent.

## Interfaz que muestra el proceso de extracción

Para poder observar el proceso de extracción de manera gráfica, se desarrolló una interfaz que muestra los Documentos que se extraen de un determinado archivo o una oración. El objeto de esta interfaz es visualizar los resultados de las distintas pruebas realizadas durante el desarrollo de este proyecto y no está disponible para el usuario final. Consta de tres componentes: *DashboardComponent*, *DirectoryComponent* y *FileComponent* que se crearon con los siguientes comandos en una terminal de GNU/Linux:

```
$ ng g component dashboard  
$ ng g component directory  
$ ng g component file
```

*DashboardComponent* es el encargado de comunicarse con la API a través de distintos *endpoints*:

Método	Endpoint	Parámetros requeridos	Resultado
GET	/api/corpus		Devuelve el árbol de directorios y archivos que forman el corpus del sistema
POST	/api/extractor	text: El texto a ser analizado para extraer los Documentos filepath: La ruta del archivo a ser analizado para extraer los Documentos	Devuelve, en formato JSON, los Documentos extraídos

Estos *endpoints* son manejados por el controlador *CorpusController* y cuyo código se muestra en la Figura 33. *DashboardComponent* utiliza de manera recursiva *DirectoryComponent* y *FileComponent* para mostrar el árbol de directorios y archivos que conforman el corpus del sistema, y un componente de Covalent, *TdJsonFormatterComponent*, para mostrar el resultado de la extracción en formato JSON.

<sup>16</sup> <https://github.com/enzonotario/sbrs-client>

<sup>17</sup> <https://github.com/Teradata/covalent-quickstart>

El componente tiene el siguiente aspecto y permite ingresar cualquier preposición (sea o no una pregunta) en el campo que se encuentra en la parte izquierda, o seleccionar un archivo del corpus. Los Documentos extraídos se visualizan en la parte derecha:

Texto

Salta es una ciudad

EXTRAER

Resultado

- Object
  - data: Object
    - original: "Salta es una ciudad"
    - parsed: Object
      - solrOutput: Array [1]
        - 0: Object
          - arg1: "Salta"
          - rel: "es"
          - arg2: "una ciudad"
          - url: "-"
          - page: "-"
          - title: "-"
          - subtitle: "-"
          - sentence: "Salta es una ciudad"
          - nps: Array [1]
          - ns: Array [1]
          - synonymous: Array [51]
          - relnf: Array [1]
          - site: "-"

corpus

salta.gov.ar

turismo.salta.gov.ar

wikipedia

## Buscador

La interfaz del buscador se construyó en el componente *SearchComponent*, que se crea ejecutando el siguiente comando en una terminal de GNU/Linux:

```
$ ng g component search
```

*SearchComponent* se encarga de enviar la pregunta realizada en lenguaje natural por parte del usuario a la API, y de mostrarle los resultados que ésta devuelve. Esto lo hace a través del siguiente *endpoint*:

Método	Endpoint	Parámetros requeridos
GET	/api/search	question: La pregunta a realizar

Éste *endpoint* se maneja en el controlador *SearchController*, cuyo código se muestra en la Figura 32.

El usuario puede ingresar cualquier pregunta realizada en lenguaje natural en el campo que se encuentra en la parte superior. El sistema le devolverá una serie de resultados, indicándole el título, subtítulo, y URL de la página a la que corresponde la respuesta, como se ve a continuación:



## 4.6 Análisis FODA

### Factores internos

#### Fortalezas

- Se cuenta con el asesoramiento de las personas involucradas en el Proyecto de Investigación “Minería de textos: Búsqueda automática de respuestas”.
- Se posee un dominio considerable del idioma inglés.
- Se dispone de información libre y gratuita en la WWW para formar la base de conocimiento del SBR.
- Se posee conocimiento y experiencia en tecnologías webs.
- Experiencia previa en desarrollo de soluciones sobre tecnologías web.
- Experiencia adquirida al participar como alumno ayudante en el Proyecto de Investigación “Minería de textos: Búsqueda automática de respuestas”.

#### Debilidades

- La documentación de la gran mayoría de herramientas está sólo disponible en idioma inglés.

### Factores Externos

#### Oportunidades

- Existencia de herramientas de software libre que ayudarán a implementar el SBR.
- Los sitios del Gobierno de la Provincia de Salta carecen de un sistema de búsqueda con las características expuestas en este trabajo.

#### Amenazas

- Las herramientas utilizadas se actualizan rápidamente y pueden generarse incompatibilidad entre ellas.

- Los administradores de los sitios podrían cambiar la estructura de los mismos, provocando que el WC no sea capaz de ubicar correctamente ciertos fragmentos, tales como el título de la página, subtítulo, texto, etc.

## 4.7 Análisis de factibilidad

### Factibilidad económica

Las herramientas usadas son gratuitas y de código abierto, lo cual no genera ningún costo y hace posible su modificación, si fuese necesario.

Los costos del desarrollo del proyecto involucran dos recursos humanos y están dados según los honorarios del COPAIPA al día 30/09/2017<sup>18</sup>. Además se debe contar con dos computadoras, una para cada miembro del equipo de trabajo.

### Costos de recursos humanos

Profesional	Costo por hora [\$]	Horas de trabajo por día [h/d]	Días [d]	Costo total [\$]
Analista Funcional	240	2	40	19200
Analista Programador	200	2	80	32000

### Costos de equipamiento

Equipo	Unidades	Precio unitario [\$]	Costo total [\$]
Notebook HP 15-AY163NR Procesador: i7 RAM: 8GB Almacenamiento: 1TB Pantalla: 15.6"	2	16599 <sup>19</sup>	33198

### Costo total

Tipo de costo	Subtotal [\$]
Costos de recursos humanos	51200
Costos de equipamiento	33198
TOTAL: \$84398	

Con un desembolso inicial del 50% se comenzará el desarrollo. A partir del 2do mes se debe cancelar el 25% restante y al finalizar el proyecto, el 25% faltante.

Para el despliegue del proyecto se debe contratar un servidor que sea el que aloje el sistema, más un dominio. Los costos de despliegue por año, tomando el dólar a \$17.27 (precio del dólar el día 19/09/2017) son los siguientes:

<sup>18</sup> <http://www.copaipa.org.ar/informatica/>

<sup>19</sup> Valor tomado de MercadoLibre el día 30/09/2017. [https://articulo.mercadolibre.com.ar/MLA-663848780-notebook-hp-15-ay163nr-i7-7500u-1tb8gb156w10dvdwgtia-\\_JM](https://articulo.mercadolibre.com.ar/MLA-663848780-notebook-hp-15-ay163nr-i7-7500u-1tb8gb156w10dvdwgtia-_JM)

### Costos de despliegue:

Ítem	Tipo de pago	Costo [\$]
Instancia EC2 t2.small en Amazon Web Services	Anual	\$2367
Dominio en namecheap.com	Anual	\$153,40
		TOTAL: \$2520,40

### Beneficio económico esperado

Los beneficios que este sistema brinda son muy atractivos para el Gobierno de la Provincia de Salta, debido a que es una herramienta para aquellos que desean conocer más sobre la Provincia. Teniendo en cuenta esto, se podría plantear la herramienta ante el Gobierno y obtener, a cambio de publicidad, un monto similar al que el Gobierno realiza en otros medios digitales, el cual rondó los \$20.000 por mes en el año 2016, llegando a los \$80.000 mensuales<sup>20</sup>. Esto permitirá recuperar rápidamente el dinero invertido y poder obtener un beneficio económico.

### Factibilidad técnica

Técnicamente hablando, es posible realizar todos los procesos del SBR en cualquier PC, ya que también es posible instalar y usar todas las herramientas utilizadas en una distribución Ubuntu, en cualquiera de sus variantes.

### Factibilidad legal

Legalmente, la extracción de información se hizo sobre textos de dos sitios gubernamentales públicos: [www.salta.gov.ar](http://www.salta.gov.ar) y [www.turismo.salta.gov.ar](http://www.turismo.salta.gov.ar), los cuales además no definen ninguna restricción de acceso para los WC en sus archivos *robots.txt*<sup>21</sup> al igual que la enciclopedia libre Wikipedia.

## 4.8 Gestión de Riesgos

Todo proyecto lleva asociado un riesgo, que es un problema potencial que puede ocurrir o no. Particularmente en este proyecto no se ha detectado ningún riesgo que pueda afectar la factibilidad técnica, pero no así con la factibilidad legal, ya que podrían existir sitios que en sus archivos *robots.txt* prohíban el acceso de WC a su contenido.

Por otra parte, alguna de las herramientas utilizadas podrían convertirse en herramientas pagas, lo cual afectaría la factibilidad económica de una manera imposible de

---

<sup>20</sup> <http://www.saltatransparente.com/2017/07/pauta-virtual-el-gobierno-de-la.html>

<sup>21</sup> El archivo *robots.txt* define las restricciones de acceso que un WC tiene sobre un sitio.

prever, aunque al tratarse de software libre siempre es posible crear una bifurcación y continuar con su desarrollo de manera gratuita.

La Tabla 5 muestra el impacto que podría ocasionar y las medidas que se podrían llevar a cabo para mitigar cada riesgo.

Riesgo	Probabilidad	Impacto	Acción para mitigar
Un determinado sitio permite la ejecución de WC cada X tiempo	50%	Leve	Programar el WC para que respete los tiempos y así evitar un bloqueo de acceso
Un determinado sitio no permite el acceso a través de WC	5%	Alto	Contactarse con el administrador del sitio web y pedir una excepción
Una determinada herramienta se convierte en una herramienta paga	10%	Leve	Continuar el desarrollo de una nueva versión gratuita a partir de la última que estuvo disponible de manera gratuita.

TABLA 5 - TABLA DE RIESGO

## 5. Resultados

### 5.1 Obtención de texto

Durante el proceso de obtención de texto se extrajo texto de 2.042 páginas diferentes, produciendo un volumen de 10,7 *megabytes*. La Tabla 6 muestra los resultados por cada sitio.

Sitio	Número de archivos	Peso
<a href="http://www.salta.gov.ar">www.salta.gov.ar</a>	7	44,4 KB
<a href="http://www.turismo.salta.gov.ar">www.turismo.salta.gov.ar</a>	1676	4,4 MB
<a href="http://www.wikipedia.org">www.wikipedia.org</a>	359	6,3 MB

TABLA 6 - RESULTADOS DE LA OBTENCIÓN DE TEXTO

### 5.2 Extracción de Documentos

El proceso de extracción de Documentos obtuvo 133.806 Documentos y sinónimos de 22.385 verbos, lo cual conforma una base de datos de 240,3 MB.

### 5.3 Evaluación

La evaluación se hizo sobre el componente de extracción de Documentos y sobre las respuestas que el sistema brinda a una tabla de preguntas.

#### Evaluación del componente de extracción de Documentos

Para evaluar este componente se tienen en cuenta dos métricas diferentes pero relacionadas: *recall* y precisión.

El banco de oraciones<sup>22</sup> que se usó es el mismo que se usó para evaluar ExtrHech frente a ReVerb, y QARelaciones frente a ExtrHech, y consta de sesenta y ocho oraciones tomadas de textos escolares. Los autores de ExtrHech reportan una precisión del 87% y un recall del 70%. El sistema QARelaciones logra obtener 347 tripletas, de las cuales 239 se consideran correctas, lo cual lleva a una precisión del 69%. Para calcular el *recall*, se consideró que el número de tripletas que se podrían obtener de las sesenta y ocho oraciones es 267, de las cuales 239 son obtenidas correctamente, lo cual lleva a un *recall* del 90%.

Teniendo en cuenta que las relaciones que se deben extraer tan sólo consisten en la separación de una oración basándose en verbos, se deberían extraer 142 relaciones. SBRS logra extraer 142, siendo 141 de estas correctas. Esto implica un recall del 100% y una precisión del 99%.

---

<sup>22</sup>[https://www.gelbukh.com/resources/spanish-open-fact-extraction/ver1/Parallel\\_FactSpaCIC.xlsx](https://www.gelbukh.com/resources/spanish-open-fact-extraction/ver1/Parallel_FactSpaCIC.xlsx)



Además de las métricas *recall* y precisión, se calcula el valor F1, que es una métrica que combina la precisión y el *recall* y es frecuentemente usado en sistemas de RI.

La Tabla 7 resume los diferentes valores para los tres sistemas: ExtrHech, QARelaciones y SBRS.

Los resultados de esta evaluación sobre las sesenta y ocho oraciones se muestran en Tabla 10.

	Precisión	Recall	F1
ExtrHech	87%	70%	0.776
QARelaciones	69%	90%	0.781
SBRS	99%	100%	0.99

TABLA 7 - COMPARACIÓN ENTRE QARELACIONES, EXTRHECH Y SBRS

### Evaluación de las respuestas

La evaluación final de la solución propuesta consiste en comparar, en base a ciento cincuenta preguntas que se muestran en la Tabla 8, los resultados acertados que brinda frente a los que brinda QARelaciones y Google. Estas preguntas han sido utilizadas para evaluar QARelaciones frente a Google. Están escritas en lenguaje natural y tratan principalmente sobre la Provincia de Salta.

Se considera que un sistema responde correctamente una pregunta si brinda, entre los primeros cinco resultados, una respuesta adecuada. Los resultados se muestran en la Tabla 9. En el caso de Google, la respuesta puede aparecer de manera explícita (en un Fragmento Destacado o en el breve resumen que acompaña el enlace que devuelve el buscador) o bien estar contenida en uno de los enlaces propuestos. En tal caso, el usuario deberá ingresar al enlace y buscar la respuesta en la página a la que accedió. Si sucede este caso, en la Tabla 9 un signo menos (-) acompaña la posición del resultado devuelto por Google, además de no ser considerado como una respuesta correcta.

De las ciento cincuenta preguntas, Google mostró una respuesta explícita en el 57% de las preguntas y una página que contiene la respuesta en el 41% de los casos. En cuatro casos (3%) no devolvió una respuesta correcta.

El sistema QARelaciones devolvió una respuesta en el 88% de los casos y devolvió respuestas inadecuadas en el 12%.

SBRS, devolvió una respuesta correcta en el 93% de los casos y devolvió una respuesta inadecuada o más allá de la quinta posición en el 7%.

De este 7% (once preguntas) que SBRS no fue capaz de responder, en realidad nueve están más allá de la quinta posición (requisito establecido para considerar una respuesta válida). Esto se debe, en gran medida, al tipo de información que se puede encontrar en el sitio [www.turismo.salta.gov.ar](http://www.turismo.salta.gov.ar) que es del tipo periodístico, generando resultados anteriores a los de, por ejemplo, Wikipedia, que aporta datos más certeros. Sólo de dos preguntas no se obtiene una respuesta adecuada en ninguna posición. En estos dos casos se debe a que la respuesta en realidad está presente con algún sinónimo del verbo que forma parte de la pregunta.

La Figura 22 muestra un gráfico de barras comparativo entre los resultados de los tres sistemas sobre el banco de preguntas de la Tabla 8. En el eje horizontal se encuentran las posiciones en las que se obtuvo la respuesta correcta (el valor cero indica que no se obtuvo una respuesta correcta en una posición anterior a la quinta). En el eje vertical, la cantidad total de respuestas obtenidas en una determinada posición.

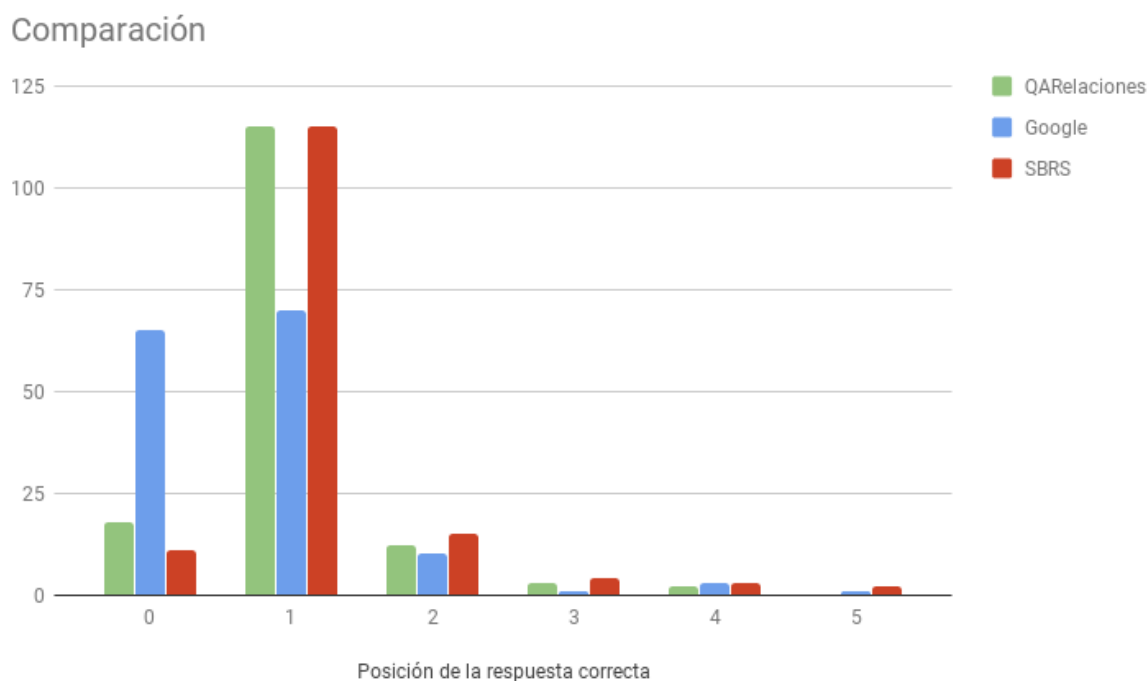


FIGURA 22 - COMPARACIÓN ENTRE QARELACIONES, GOOGLE Y SBRS DE RESULTADOS SOBRE EL BANCO DE PREGUNTAS

## 6. Conclusiones

El desarrollo de este proyecto concluyó en una alternativa con resultados comparable al gran trabajo que se hizo en QARelaciones, planteando un enfoque diferente para extraer la información que forma la base de conocimiento.

Se obtuvo un SBR construido completamente sobre tecnologías web que permite ser añadido a cualquier sitio web de manera fácil.

En cuanto a la experiencia adquirida, considero que me ha otorgado una gran experiencia en el desarrollo de un sistema que se comunica con otros a través de distintas APIs y Sockets, cada uno desarrollado en distintos lenguajes de programación. Además, al haber construido el Extractor de Documentos, pude profundizar mi conocimiento en PLN.

### 6.1 Futuras líneas de investigación

El presente proyecto se plantea como el inicio de futuros trabajos de investigación sobre los SBR, los cuales pueden consistir en:

- Análisis semántico de los textos, para poder extraer relaciones teniendo en cuenta el significado de las palabras, en vez de sólo tener en cuenta su categoría gramatical.
- Agregar el componente de Lucene “*Similarity*”, el cual devuelve Documentos similares a uno en particular. Esto podría usarse para devolver respuestas similares a las que el usuario está buscando, que podrían ser de interés.
- Ampliar el dominio, para brindar respuestas no sólo de la Provincia de Salta, sino sobre cualquier otro ámbito. Esto sólo implica recoger texto de otros sitios webs que contengan información del ámbito deseado.
- Realizar un WC que sea aplicable a cualquier sitio, sin previo análisis, ya que actualmente, para realizar WC sobre un sitio, es necesario estudiar la estructura del mismo para identificar dónde se ubica el título, subtítulo, texto y demás elementos. Esto será válido hasta que el administrador del sitio realice alguna modificación a nivel estructura, momento en el cual se deberá volver a analizar el sitio para determinar las nuevas ubicaciones de los elementos que se desean extraer.

## Bibliografía

- Apache. (s.f.). Recuperado el 21 de 09 de 2017, de OpenSUSE:  
<https://es.opensuse.org/Apache>
- Apache Lucene. (s.f.). Recuperado el 04 de 09 de 2017, de Apache Foundation:  
<https://apachefoundation.wikispaces.com/Apache+Lucene>
- Cardoso, A. C., Pérez Abelleira, A. M., & Notario, E. R. (2016). Extracción de relaciones para la búsqueda automática de respuestas. En A. C. Cardoso, A. M. Pérez Abelleira, & E. R. Notario (Ed.), *Simposio Argentino de Inteligencia Artificial, 45 Jornadas Argentinas de Informática (JAIIO)*, (págs. 25-32). Buenos Aires.
- Chapman, P., Clinton, J., Kerber, R., Khabaza, T., Reinartz, T., Shearer, C., y otros. (2000). *CRISP-DM 1.0: Step-by-step data mining guide*. SPSS.
- Documentación de Solarium. (s.f.). Recuperado el 04 de 09 de 2017, de Github:  
<https://github.com/solariumphp/solarium>
- Dowling, M. (14 de 08 de 2017). *Documentación de Guzzle*. Recuperado el 14 de 08 de 2017, de <http://docs.guzzlephp.org/en/stable/>
- Empezando - Acerca del control de versiones. (s.f.). Recuperado el 01 de 10 de 2017, de Git:  
<https://git-scm.com/book/es/v1/Empezando-Acerca-del-control-de-versiones>
- Fader, A., Soderland, S., & Etzioni, O. (s.f.). *Identifying Relations for Open Information Extraction*. Recuperado el 17 de 09 de 2017, de <http://www.aclweb.org/anthology/D11-1142>
- FriendsOfPHP. (s.f.). *Documentación de Goutte*. Recuperado el 14 de 08 de 2017, de Github:  
<https://github.com/FriendsOfPHP/Goutte>
- Fuente Ibáñez, L. (29 de 08 de 2016). *Recuperación y organización de la información*. Obtenido de Sistemas de Question-Answering: <http://question-answering.orgfree.com/>
- Gallego, A. J. (14 de 08 de 2017). *¿Qué es Laravel?* Recuperado el 14 de 08 de 2017, de <https://ajgallego.gitbooks.io/laravel-5/content/introduccion.html>
- Gamallo, P., Garcia, M., & Fernández-Lanza, S. (23-27 de 04 de 2012). *Dependency-Based Open Information Extraction*. Avignon, France.

- Google. (s.f.). *Fragmentos destacados en la búsqueda*. Recuperado el 15 de 08 de 2017, de <https://support.google.com/webmasters/answer/6229325?hl=es-419>
- John Burger, C. C.-Y. (2003). *Issues, Tasks and Program Structures in Question & Answering*.
- Korfhage, R. R. (s.f.). *Information Storage and Retrieval*. Wiley India Pvt.
- Moreno, L. P. (1999). *Introducción al procesamiento del Lenguaje Natural*. Publicaciones de la Universidad de Alicante.
- Padró, L. (2012). *Analizadores Multilingües en FreeLing*. Catalunya: Centro de Investigación TALP.
- Robles, V. (05 de 08 de 2017). *¿Que es Angular y para que sirve?* Recuperado el 04 de 09 de 2017, de Victor Robles: <http://victorroblesweb.es/2017/08/05/que-es-angular-y-para-que-sirve/>
- Román, J. V. (08 de 08 de 2016). *CRISP-DM: La metodología para poner orden en los proyectos de Data Science*. Recuperado el 19 de 09 de 2017, de singular data&analytics: <https://data.singular.team/es/art/25/crisp-dm-la-metodologia-para-poner-orden-en-los-proyectos-de-data-science>
- Rouse, M. (01 de 2015). *MySQL*. Recuperado el 21 de 09 de 2017, de Search Data Center: <http://searchdatacenter.techtarget.com/es/definicion/MySQL>
- Seta, L. d. (11 de 6 de 2010). *Apache Solr: una introducción*. Recuperado el 02 de 09 de 2017, de Dos Ideas: <https://dosideas.com/noticias/java/913-apache-solr-una-introduccion>
- The PHP Group. (14 de 08 de 2017). *¿Qué es PHP?* Recuperado el 14 de 08 de 2017, de <http://php.net/manual/es/intro-what-is.php>
- Ubuntu. (s.f.). Recuperado el 04 de 09 de 2017, de EcuRed: <https://www.ecured.cu/Ubuntu>
- Vicomtech. (29 de 08 de 2016). *Procesamiento del Lenguaje Natural*. Obtenido de vicomtech: <http://www.vicomtech.org/t4/e11/procesamiento-del-lenguaje-natural>
- W.B., C. (1987). *Approaches to intelligent information retrieval. Information Proccesing & Management*.

Zhila, A., & Gelbukh, A. (2013). *Comparison of open information extraction for English and Spanish*. Recuperado el 17 de 09 de 2017, de <http://www.dialog-21.ru/digests/dialog2013/materials/pdf/ZhilaA.pdf>

## Anexo 1 - Banco de preguntas

¿En qué departamento está el Abra del Acay?  
¿Dónde está situado el Abra del Acay?  
¿Cómo se conoce también al abra del Acay?  
¿Cómo se conoce también al Abra del Acay?  
¿Qué es una apacheta?  
¿Dónde está la laguna de Guayatayoc?  
¿En qué departamento está Aguaray?  
¿Con quién limita Aguaray?  
¿Dónde se encuentran los chané?  
¿Dónde hay asentamientos de chiriguano?  
¿Dónde tienen asentamientos los chiriguano?  
¿de dónde provienen los chané?  
¿cuál es la fauna autóctona de Aguaray?  
¿Cuál es la base de la economía de Aguaray?  
¿Quiénes son los aimara?  
¿En qué consistía la alimentación de los aimaras?  
¿Dónde se centra la civilización aimara?  
¿Por qué se caracterizan los aimaras?  
¿Por qué se caracterizan los aimaras?  
¿En qué zonas viven los Carangas?  
¿En qué zonas viven los carangas?  
¿Dónde viven los Carangas?  
¿Dónde viven los aimaras ahora?  
¿Cuál es el centro urbano de los aimaras?  
¿En qué provincia está Angastaco?  
¿En qué departamento está Angastaco?  
¿Cómo es el clima de Angastaco?  
¿Por dónde se llega a Angastaco?  
¿Quién es la patrona de Apolinario Saravia?  
¿Dónde está Apolinario Saravia?  
¿Cuáles son las tradiciones de los atacameños?  
¿de dónde se extrae el cobre?  
¿de dónde se extrae el oro?  
¿Quién dictó el decreto de la bandera de Salta?  
¿Qué ley crea la bandera de Salta?  
¿Quién reconstruyó el cabildo de Salta?  
¿Quién restauró el cabildo de Salta?  
¿Dónde está Cachi?  
¿Dónde está el Centro de Convenciones de Salta?  
¿Cuánto mide el cerro San Bernardo?  
¿Qué altura tiene el cerro San Bernardo?  
¿Cómo es el paisaje de Iruya?  
¿Quiénes son los mayores exponentes de la cultura argentina?  
¿Cuántas provincias tiene la República Argentina?  
¿Cuántas provincias han existido en la República Argentina?  
¿Cuántas provincias hay en la República Argentina?  
¿Dónde se ubica Villa San Lorenzo?  
¿Cuál es la temperatura media de Salta?  
¿Qué museos existen en la ciudad de Salta?  
¿Qué museos hay en la ciudad de Salta?  
¿Quién hizo el Cristo de la Humildad y de la Paciencia?  
¿Qué es Apolinario Saravia?  
¿Cuántos habitantes tiene Apolinario Saravia?  
¿Cuántas parroquias tiene Apolinario Saravia?  
¿Qué es La Caldera?  
¿Qué es El Bordo?  
¿Dónde está Campo Alegre?  
¿Qué es Campo Alegre?  
¿Qué produce Las Lajitas?  
¿Cuántos habitantes tiene Las Lajitas?  
¿Quién es Lucrecia Martel?  
¿Qué premios obtuvo Lucrecia Martel?  
¿Qué es el Museo de la Vid y el Vino?

¿Cuántas regiones hay en Argentina?  
 ¿En qué continente se ubica la República Argentina?  
 ¿Cuál es la capital de Argentina?  
 ¿Cuál es la capital de la República Argentina?  
 ¿Cuántos habitantes viven en la República Argentina?  
 ¿Cuál es la superficie de Argentina?  
 ¿Quién es el presidente de la República Argentina?  
 ¿Cuál es el deporte nacional de la República Argentina?  
 ¿Quiénes son los mejores futbolistas de la República Argentina?  
 ¿Cuál es la comida típica de Argentina?  
 ¿Cuál es la comida típica de la República Argentina?  
 ¿Quién fue Artidorio Cresseri ?  
 ¿Dónde nació Artidorio Cresseri?  
 ¿Qué obra escribió Artidorio Cresseri?  
 ¿Quién escribió la López Pereira?  
 ¿En qué año falleció Artidorio Cresseri?  
 ¿Qué significa Cachi?  
 ¿A qué altitud está Cachi?  
 ¿Dónde está la iglesia de Cachi?  
 ¿De qué siglo es la iglesia de Cachi?  
 ¿Quién fundó Cafayate?  
 ¿Qué uva se cultiva en Cafayate?  
 ¿en qué provincias habitaban los calchaquies?  
 ¿en qué años tuvieron lugar las guerras calchaquies?  
 ¿en qué años ocurrieron las guerras calchaquies?  
 ¿cómo eran las casas de los calchaquies?  
 ¿cómo eran las viviendas de los calchaquies?  
 ¿Quién fue el primer gobernador de Salta?  
 ¿Quién fue el primer gobernador autónomo de Salta?  
 ¿a quién nombró Belgrano como gobernador militar de la provincia de Salta?  
 ¿Quién ordenó la restauración de la cruz de la Batalla de Salta?  
 ¿Cuándo se construyó el Monumento a la Batalla de Salta?  
 ¿Cuáles son los límites del Parque Nacional El Rey?  
 ¿Quién es el máximo rival de Juventud Antoniana?  
 ¿Dónde está la cancha de Juventud Antoniana?  
 ¿qué nombre tiene el estado de Juventud Antoniana?  
 Dónde se celebra la Fiesta Nacional de la Chicha  
 ¿Qué actividades se realizan en Villa San Lorenzo?  
 ¿Qué actividades se realizan en el Dique Campo Alegre?  
 ¿A qué distancia de Molinos está la Laguna de Brealito?  
 ¿Dónde está la Laguna de Brealito?  
 ¿Qué actividades se realizan en el Dique Las Maderas?  
 ¿Qué circuito hay próximo al centro salteño?  
 ¿Dónde está la imagen del Cristo Redentor?  
 ¿Qué actividades se realizan en La Caldera?  
 ¿Dónde hay tradición de teleros?  
 ¿Qué producción hay en Vaqueros?  
 ¿Cuando se realiza la fiesta de San Cayetano en Vaqueros?  
 ¿Cómo es la arquitectura de Campo Quijano?  
 ¿Qué ríos confluyen en Cachi?  
 ¿A qué altura está la Recta del Tin Tin?  
 ¿Qué lugares forman parte del Circuito Sur?  
 ¿Cuáles eran los gauchos a las órdenes del general Güemes?  
 ¿Qué costumbres ancestrales se conservan en la Puna?  
 ¿Con qué se realizan los tejidos en la Puna?  
 ¿Cuántos kilómetros hay a San Antonio de los Cobres?  
 ¿Qué se observa desde la cumbre del Cerro Macón?  
 ¿Cuándo es la fiesta patronal de Santa Rosa de Tastil?  
 ¿Por qué ruta se llega al Parque Nacional El Rey?  
 ¿Dónde se encuentra el Hotel Termas?  
 ¿Dónde está el Hotel Termas?  
 ¿Dónde hay baños termales?  
 ¿Dónde está el Cerro el Crestón?  
 ¿Cuándo fue inaugurado el Templo de la Parroquia San José?  
 ¿Quiénes obtuvieron la victoria en el Combate de Las Piedras?  
 ¿Qué nombró el Gral. José de San Martín al Gral. Güemes?  
 ¿Dónde está el monumento natural a la Jura de la Bandera?



<p>¿Con qué río limita El Galpón? ¿Qué turismo se puede practicar en Mosconi? ¿Por qué ciudad pasa el Trópico de Capricornio? ¿Por dónde pasa el Trópico de Capricornio? ¿Cuál es la fiesta patronal de Tartagal? ¿Cuál es la economía base de Tartagal? ¿Cuáles son los diferentes atractivos de la Provincia? ¿Cuándo se realizará la Feria Internacional de Turismo? ¿Qué río atraviesa Rosario de Lerma? ¿Qué caracteriza el paisaje de Iruya? ¿Quién es el presidente del Consejo de Ciencias Económicas? ¿Qué manifestaciones culturales hay en la Puna? ¿Contra quién lucharon los gauchos a las órdenes del general Güemes? ¿Qué se pesca en el río Calchaquí? ¿Dónde hay pinturas rupestres? ¿Qué edificios se conservan en San Carlos? ¿Qué significa el nombre de Angastaco? ¿Con quién se compara a Esteco? ¿Cuándo fue inaugurado el Templo de la Parroquia San José de Metán? ¿Cómo es el clima de El Galpón?</p>
---

TABLA 8 - BANCO DE PREGUNTAS

## Anexo 2 - Resultados del Banco de preguntas

	Pregunta en lenguaje natural	QA	G	SBRS
1	¿En qué departamento está el Abra del Acay?	1	1-	1
2	¿Dónde está situado el Abra del Acay?	1	1	1
3	¿Cómo se conoce también al abra del Acay?	1	1	1
4	¿Cómo se conoce también al Abra del Acay?	1	1	1
5	¿Qué es una apacheta?	1	1	3
6	¿Dónde está la laguna de Guayatayoc?	1	1	1
7	¿En qué departamento está Aguaray?	1	1	1
8	¿Con quién limita Aguaray?	1	1	1
9	¿Dónde se encuentran los chané?	1	1-	2
10	¿Dónde hay asentamientos de chiriguano?	1	1-	1
11	¿Dónde tienen asentamientos los chiriguano?	1	1-	1
12	¿de dónde provienen los chané?	3	1-	1
13	¿cuál es la fauna autóctona de Aguaray?	1	1-	1
14	¿Cuál es la base de la economía de Aguaray?	1	1-	1
15	¿Quiénes son los aimara?	2	1-	2
16	¿En qué consistía la alimentación de los aimaras?	1	1	1
17	¿Dónde se centra la civilización aimara?	1	1	1
18	¿Por qué se caracterizan los aimaras?	1	1	1
19	¿Por qué se caracterizan los aymaras?	1	1	1
20	¿En qué zonas viven los Carangas?	0	2-	1
21	¿En qué zonas viven los carangas?	0	2-	1
22	¿Dónde viven los Carangas?	2	0	1
23	¿Dónde viven los aimaras ahora?	1	1	1
24	¿Cuál es el centro urbano de los aimaras?	1	1	1
25	¿En qué provincia está Angastaco?	1	1	1
26	¿En qué departamento está Angastaco?	1	1	1
27	¿Cómo es el clima de Angastaco?	1	5	1
28	¿Por dónde se llega a Angastaco?	1	1-	2
29	¿Quién es la patrona de Apolinario Saravia?	1	1	1
30	¿Dónde está Apolinario Saravia?	1	1-	2
31	¿Cuáles son las tradiciones de los atacameños?	1	1	1
32	¿de dónde se extrae el cobre?	1	3-	2
33	¿de dónde se extrae el oro?	1	1	4
34	¿Quién dictó el decreto de la bandera de Salta?	1	1	1
35	¿Qué ley crea la bandera de Salta?	1	1	3
36	¿Quién reconstruyó el cabildo de Salta?	1	1	1
37	¿Quién restauró el cabildo de Salta?	0	1	1
38	¿Dónde está Cachi?	1	1	9
39	¿Dónde está el Centro de Convenciones de Salta?	2	1	6
40	¿Cuánto mide el cerro San Bernardo?	4	2	0
41	¿Qué altura tiene el cerro San Bernardo?	1	1	1

42	¿Cómo es el paisaje de Iruya?	1	4-	2
43	¿Quiénes son los mayores exponentes de la cultura argentina?	0	1-	1
44	¿Cuántas provincias tiene la República Argentina?	0	2	1
45	¿Cuántas provincias han existido en la República Argentina?	1	1	1
46	¿Cuántas provincias hay en la República Argentina?	1	1	2
47	¿Dónde se ubica Villa San Lorenzo?	2	1	1
48	¿Cuál es la temperatura media de Salta?	3	1	4
49	¿Qué museos existen en la ciudad de Salta?	1	1	1
50	¿Qué museos hay en la ciudad de Salta?	1	1	1
51	¿Quién hizo el Cristo de la Humildad y de la Paciencia?	1	1-	1
52	¿Qué es Apolinario Saravia?	1	1	1
53	¿Cuántos habitantes tiene Apolinario Saravia?	1	1-	1
54	¿Cuántas parroquias tiene Apolinario Saravia?	1	1	1
55	¿Qué es La Caldera?	1	2	1
56	¿Qué es El Bordo?	1	2	1
57	¿Dónde está Campo Alegre?	1	3-	5
58	¿Qué es Campo Alegre?	1	2-	1
59	¿Qué produce Las Lajitas?	1	1-	3
60	¿Cuántos habitantes tiene Las Lajitas?	1	1	1
61	¿Quién es Lucrecia Martel?	1	1-	1
62	¿Qué premios obtuvo Lucrecia Martel?	1	1	1
63	¿Qué es el Museo de la Vid y el Vino?	1	1	1
64	¿Cuántas regiones hay en Argentina?	0	1	6
65	¿En qué continente se ubica la República Argentina?	2	1-	2
66	¿Cuál es la capital de Argentina?	0	2	1
67	¿Cuál es la capital de la República Argentina?	0	2-	1
68	¿Cuántos habitantes viven en la República Argentina?	1	1	2
69	¿Cuál es la superficie de la República Argentina?	2	2	1
70	¿Quién es el presidente de la República Argentina?	0	1-	6
71	¿Cuál es el deporte nacional de la República Argentina?	1	1	1
72	¿Quiénes son los mejores futbolistas de la República Argentina?	1	1-	1
73	¿Cuál es la comida típica de Argentina?	1	1-	1
74	¿Cuál es la comida típica de la República Argentina?	1	1-	1
75	¿Quién fue Artidorio Cresseri ?	1	1	1
76	¿Dónde nació Artidorio Cresseri?	1	1-	1
77	¿Qué obra escribió Artidorio Cresseri?	1	1	1
78	¿Quién escribió la López Pereira?	3	1	7
79	¿En qué año falleció Artidorio Cresseri?	0	1	1
80	¿Qué significa Cachi?	4	2	1
81	¿A qué altitud está Cachi?	1	1-	1
82	¿Dónde está la iglesia de Cachi?	2	1	2
83	¿De qué siglo es la iglesia de Cachi?	1	1-	1
84	¿Quién fundó Cafayate?	1	1-	5
85	¿Qué uva se cultiva en Cafayate?	1	1	1
86	¿en qué provincias habitaban los calchaquíes?	1	1-	6
87	¿en qué años tuvieron lugar las guerras calchaquíes?	1	1	1
88	¿en qué años ocurrieron las guerras calchaquíes?	1	1-	1
89	¿cómo eran las casas de los calchaquíes?	1	1	0
90	¿cómo eran las viviendas de los calchaquíes?	1	1	4
91	¿Quién fue el primer gobernador de Salta?	1	1	1
92	¿Quién fue el primer gobernador autónomo de Salta?	1	1	1
93	¿a quién nombró Belgrano como gobernador militar de la provincia de	1	1	1

	Salta?			
94	¿Quién ordenó la restauración de la cruz de la Batalla de Salta?	1	1	1
95	¿Cuándo se construyó el Monumento a la Batalla de Salta?	1	1-	1
96	¿Cuáles son los límites del Parque Nacional El Rey?	1	3-	7
97	¿Quién es el máximo rival de Juventud Antoniana?	1	1	1
98	¿Dónde está la cancha de Juventud Antoniana?	2	1-	2
99	¿qué nombre tiene el estado de Juventud Antoniana?	1	2	1
100	Dónde se celebra la Fiesta Nacional de la Chicha	1	2-	1
101	¿Qué actividades se realizan en Villa San Lorenzo?	1	4	1
102	¿Qué actividades se realizan en el Dique Campo Alegre?	1	1	2
103	¿A qué distancia de Molinos está la Laguna de Brealito?	0	2-	1
104	¿Dónde está la Laguna de Brealito?	2	1	1
105	¿Qué actividades se realizan en el Dique Las Maderas?	0	1-	2
106	¿Qué circuito hay próximo al centro salteño?	1	1	1
107	¿Dónde está la imagen del Cristo Redentor?	1	1	1
108	¿Qué actividades se realizan en La Caldera?	1	1	1
109	¿Dónde hay tradición de teleros?"	1	1	1
110	¿Qué producción hay en Vaqueros?	1	2-	3
111	¿Cuando se realiza la fiesta de San Cayetano en Vaqueros?	1	3	1
112	¿Cómo es la arquitectura de Campo Quijano?	1	1	1
113	¿Qué ríos confluyen en Cachi?	1	1-	2
114	¿A qué altura está la Recta del Tin Tin?	1	3-	1
115	¿Qué lugares forman parte del Circuito Sur?	1	2-	1
116	¿Cuáles eran los gauchos a las órdenes del general Güemes?	1	1-	1
117	¿Qué costumbres ancestrales se conservan en la Puna?	1	1-	1
118	¿Con qué se realizan los tejidos en la Puna?	1	-1	1
119	¿Cuántos kilómetros hay a San Antonio de los Cobres?	1	1-	1
120	¿Qué se observa desde la cumbre del Cerro Macón?	1	1	1
121	¿Cuándo es la fiesta patronal de Santa Rosa de Tastil?	1	1	1
122	¿Por qué ruta se llega al Parque Nacional El Rey?	1	1-	1
123	¿Dónde se encuentra el Hotel Termas?	1	1	1
124	¿Dónde está el Hotel Termas?	1	1	1
125	¿Dónde hay baños termales?	2	1	1
126	¿Dónde está el Cerro el Crestón?	1	2	2
127	¿Cuándo fue inaugurado el Templo de la Parroquia San José?	1	2	1
128	¿Quiénes obtuvieron la victoria en el Combate de Las Piedras?	1	1-	1
129	¿Qué nombró el Gral. José de San Martín al Gral. Güemes?	1	1-	1
130	¿Dónde está el monumento natural a la Jura de la Bandera?	1	0	1
131	¿Con qué río limita El Galpón?	1	1	1
132	¿Qué turismo se puede practicar en Mosconi?	2	1	1
133	¿Por qué ciudad pasa el Trópico de Capricornio?	1	4	1
134	¿Por dónde pasa el Trópico de Capricornio?	1	2-	1
135	¿Cuál es la fiesta patronal de Tartagal?	1	1-	1
136	¿Cuál es la economía base de Tartagal?	1	3-	1
137	¿Cuáles son los diferentes atractivos de la Provincia?	1	2-	1
138	¿Cuándo se realizará la Feria Internacional de Turismo?	1	1-	1
139	¿Qué río atraviesa Rosario de Lerma?	1	4	1
140	¿Qué caracteriza el paisaje de Iruya?	1	3-	1
141	¿Quién es el presidente del Consejo de Ciencias Económicas?	1	0	1
142	¿Qué manifestaciones culturales hay en la Puna?	1	2-	1
143	¿Contra quién lucharon los gauchos a las órdenes del general Güemes?	1	1-	1
144	¿Qué se pesca en el río Calchaquí?	0	3-	9

145	¿Dónde hay pinturas rupestres?	2	1	1
146	¿Qué edificios se conservan en San Carlos?	0	2-	9
147	¿Qué significa el nombre de Angastaco?	0	3-	1
148	¿Con quién se compara a Esteco?	0	0	1
149	¿Cuándo fue inaugurado el Templo de la Parroquia San José de Metán?	0	1	1
150	¿Cómo es el clima de El Galpón?	0	1	1

TABLA 9 - RESULTADOS DEL BANCO DE PREGUNTAS

## Anexo 3 – Resultado de la evaluación del componente de extracción de Documentos

Arg1	Rel	Arg2	Total obtenidas	Total correctas	Total esperadas
			<b>2</b>	<b>2</b>	<b>2</b>
Los egipcios se	caracterizaron	por sus creencias relacionadas con la muerte			
Los egipcios se caracterizaron por sus creencias	relacionadas	con la muerte			
			<b>2</b>	<b>2</b>	<b>2</b>
El libro de los muertos	es	otro de los escritos que se han encontrado en diversas tumbas egipcias			
El libro de los muertos es otro de los escritos que se	han encontrado	en diversas tumbas egipcias			
			<b>1</b>	<b>1</b>	<b>1</b>
El ritmo	es	la alternancia de las sílabas átonas con sílabas tónicas			
			<b>1</b>	<b>0</b>	<b>1</b>
Los datos de las fuentes	consultadas deben registrarse	en fichas bibliográficas			
			<b>3</b>	<b>3</b>	<b>3</b>
	Redactar	un borrador en el cual se deben considerar las partes que conforman un trabajo de investigación			
Redactar un borrador en el cual se	deben considerar	las partes que conforman un trabajo de investigación			
Redactar un borrador en el cual se deben considerar las partes que	conforman	un trabajo de investigación			
			<b>1</b>	<b>1</b>	<b>1</b>
Esta sección	debe ser	breve e interesante			
			<b>2</b>	<b>2</b>	<b>2</b>
Las conclusiones	sintetizan	los argumentos presentados en el desarrollo del trabajo			
Las conclusiones sintetizan los argumentos	presentados	en el desarrollo del trabajo			
			<b>2</b>	<b>2</b>	<b>2</b>
La bibliografía	es	el listado alfabético de todas las fuentes consultadas para la elaboración del trabajo			
La bibliografía es el listado alfabético de todas las fuentes	consultadas	para la elaboración del trabajo			
			<b>1</b>	<b>1</b>	<b>1</b>
La bibliografía	se estructura	con los datos de las fichas bibliográficas de esos textos			
			<b>2</b>	<b>2</b>	<b>2</b>
Las repeticiones	sirven	para acentuar las emociones y sentimientos del hablante lírico			
Las repeticiones sirven para	acentuar	las emociones y sentimientos del hablante lírico			
			<b>2</b>	<b>2</b>	<b>2</b>

	Es	muy común el empleo de llaves para estructurar un cuadro sinóptico			
Es muy común el empleo de llaves para	estructurar	un cuadro sinóptico			
			<b>3</b>	<b>3</b>	<b>3</b>
El epíteto heroico	es	la expresión que menciona una característica o cualidad del personaje u objeto nombrado			
El epíteto heroico es la expresión que	menciona	una característica o cualidad del personaje u objeto nombrado			
El epíteto heroico es la expresión que menciona una característica o cualidad del personaje u objeto	nombrado				
			<b>2</b>	<b>2</b>	<b>2</b>
El vocativo épico	es	un enunciado exclamativo intercalado en una oración			
El vocativo épico es un enunciado exclamativo	intercalado	en una oración			
			<b>2</b>	<b>2</b>	<b>2</b>
El vocativo épico	atrae	la atención del lector o del oyente y lo ubica en la trama del relato			
El vocativo épico atrae la atención del lector o del oyente y lo	ubica	en la trama del relato			
			<b>1</b>	<b>1</b>	<b>1</b>
Los documentos escritos antiguos	pertenecen	a la época de la dinastía Shang			
			<b>1</b>	<b>1</b>	<b>1</b>
Los métodos modernos de investigación	han permitido estudiar	al hombre prehistórico			
			<b>2</b>	<b>2</b>	<b>2</b>
La Química	usa	técnicas para analizar sustancias			
La Química usa técnicas para	analizar	sustancias			
			<b>2</b>	<b>2</b>	<b>2</b>
Se	usa	el guión menor para formar adjetivos compuestos			
Se usa el guión menor para	formar	adjetivos compuestos			
			<b>2</b>	<b>2</b>	<b>2</b>
Los documentos	escritos	más antiguos fueron encontrados en Mesopotamia y Egipto			
Los documentos escritos más antiguos	fueron encontrados	en Mesopotamia y Egipto			
			<b>3</b>	<b>3</b>	<b>3</b>
La Botánica	estudia	el polen fósil y ha logrado analizar las características de la vegetación e inferir los climas			
La Botánica estudia el polen fósil y	ha logrado analizar	las características de la vegetación e inferir los climas			
La Botánica estudia el polen fósil y ha logrado analizar las características de la vegetación e	inferir	los climas			
			<b>8</b>	<b>8</b>	<b>8</b>
La arqueología	usa	nuevas técnicas para excavar y			

		localizar y estudiar los restos materiales y huellas y señales que el hombre ha dejado en el pasado para reconstruir y comprender su vida en todos los aspectos posibles .			
La arqueología usa nuevas técnicas para	excavar	y localizar y estudiar los restos materiales y huellas y señales que el hombre ha dejado en el pasado para reconstruir y comprender su vida en todos los aspectos posibles .			
La arqueología usa nuevas técnicas para excavar y	localizar	y estudiar los restos materiales y huellas y señales que el hombre ha dejado en el pasado para reconstruir y comprender su vida en todos los aspectos posibles .			
La arqueología usa nuevas técnicas para excavar y localizar y	estudiar	los restos materiales y huellas y señales que el hombre ha dejado en el pasado para reconstruir y comprender su vida en todos los aspectos posibles .			
La arqueología usa nuevas técnicas para excavar y localizar y estudiar los restos materiales y huellas y señales que el hombre	ha dejado	en el pasado para reconstruir y comprender su vida en todos los aspectos posibles .			
La arqueología usa nuevas técnicas para excavar y localizar y estudiar los restos materiales y huellas y señales que el hombre ha dejado en el	pasado	para reconstruir y comprender su vida en todos los aspectos posibles .			
La arqueología usa nuevas técnicas para excavar y localizar y estudiar los restos materiales y huellas y señales que el hombre ha dejado en el pasado para	reconstruir	y comprender su vida en todos los aspectos posibles .			
La arqueología usa nuevas técnicas para excavar y localizar y estudiar los restos materiales y huellas y señales que el hombre ha dejado en el pasado para reconstruir y	comprender	su vida en todos los aspectos posibles .			
			<b>1</b>	<b>1</b>	<b>1</b>
La Prehistoria	abarca	desde la aparición de la humanidad hasta la invención de la escritura			
			<b>1</b>	<b>1</b>	<b>1</b>
El agua	es	indispensable para la vida			
			<b>1</b>	<b>1</b>	<b>1</b>
La numeración arábica	procede	de India			
			<b>1</b>	<b>1</b>	<b>1</b>
Benito Juárez	nació	en San Pablo Guelatao, Oaxaca, en 1806			
			<b>4</b>	<b>4</b>	<b>4</b>
Los primeros homínidos	eran	recolectores y sólo comían carne cuando encontraban los restos abandonados por otros animales .			



Los primeros homínidos eran recolectores y sólo	comían	carne cuando encontraban los restos abandonados por otros animales .			
Los primeros homínidos eran recolectores y sólo comían carne cuando	encontraban	los restos abandonados por otros animales .			
Los primeros homínidos eran recolectores y sólo comían carne cuando encontraban los restos	abandonados	por otros animales .			
			<b>1</b>	<b>1</b>	<b>1</b>
La civilización China nos	heredó	el papel , la pólvora , una forma de imprenta rudimentaria , y la brújula			
			<b>2</b>	<b>2</b>	<b>2</b>
El surgimiento de la agricultura	fue	posible por los cambios climáticos que crearon un ambiente propicio para la reproducción, el cuidado y la selección de plantas.			
El surgimiento de la agricultura fue posible por los cambios climáticos que	crearon	un ambiente propicio para la reproducción, el cuidado y la selección de plantas.			
			<b>2</b>	<b>2</b>	<b>2</b>
El mar Mediterráneo	es	un mar profundo, con escasas corrientes marinas, lo cual facilita la navegación			
El mar Mediterráneo es un mar poco profundo, con escasas corrientes marinas, lo cual	facilita	la navegación			
			<b>3</b>	<b>3</b>	<b>3</b>
Los primeros pobladores de América	llegaron hace	30 mil años, aproximadamente, siguiendo a las manadas de animales que acostumbraban cazar			
Los primeros pobladores de América llegaron hace 30 mil años, aproximadamente,	siguiendo	a las manadas de animales que acostumbraban cazar			
Los primeros pobladores de América llegaron hace 30 mil años, aproximadamente, siguiendo a las manadas de animales que	acostumbraban cazar				
			<b>1</b>	<b>1</b>	<b>1</b>
En América la agricultura	inició	entre el 8000 y el 5000 a.C.			
			<b>1</b>	<b>1</b>	<b>1</b>
La agrupación de seres humanos en un mismo espacio	favoreció	el intercambio de conocimientos y el desarrollo de las ciencias y el arte.			
			<b>2</b>	<b>2</b>	<b>2</b>
El mamut	era	un animal de gran tamaño al que se cazaba mediante diversas técnicas			
El mamut era una animal de gran tamaño al que se	cazaba	mediante diversas técnicas			
			<b>4</b>	<b>4</b>	<b>4</b>
Los mamuts	migraron	de África hace 3.5 millones de años y llegaron a vivir en Europa, Asia y América.			

Los mamuts migraron de África	hace	3.5 millones de años y llegaron a vivir en Europa, Asia y América.			
Los mamuts migraron de África					
hace 3.5 millones de años y	llegaron	a vivir en Europa, Asia y América.			
Los mamuts migraron de África					
hace 3.5 millones de años y					
llegaron a	vivir	en Europa, Asia y América.			
			1	1	1
Las civilizaciones agrícolas también	desarrollaron	la ciencia			
			2	2	2
Los cretenses	eran	un pueblo pacífico de navegantes que estuvo en contacto con Egipto y Medio Oriente			
Los cretenses eran un pueblo pacífico de navegantes que	estuvo en contacto	con Egipto y Medio Oriente			
			2	2	2
Los primeros griegos se	organizaron	en grupos que tenían lazos familiares			
Los primeros griegos se organizaron en grupos que	tenían	lazos familiares			
			1	1	1
Esparta	era gobernada	por reyes			
			1	1	1
En Atenas los gobernantes	eran elegidos	por el voto de los ciudadanos			
			1	1	1
El término democracia	significa	gobierno del pueblo			
			1	1	1
La democracia ateniense se	basaba	en la participación de todos los ciudadanos de la vida política			
			1	1	1
La cultura griega	alcanzó	su esplendor en el siglo V a.C.			
			2	2	2
La civilización helenística	llegó	a su fin en el siglo I a.C., cuando Roma consumó la conquista de Egipto.			
La civilización helenística llegó a su fin en el siglo I a.C., cuando Roma	consumó	la conquista de Egipto.			
			1	1	1
La historia de la civilización romana se	divide	en tres periodos			
			1	1	1
Roma	fue gobernada	por siete reyes , etruscos y latinos , en diferentes periodos			
			1	1	1
Durante la república	comenzó	la expansión de los romanos			
			2	2	2
El último periodo de la civilización romana	fue	el imperio, que abarcó desde el año 27 a.C. hasta el año 476 d.C.			
El último periodo de la civilización romana fue el imperio, que	abarcó	desde el año 27 a.C. hasta el año 476 d.C.			
			1	1	1

El primer emperador de Roma	fue	el político y militar Octavio Augusto			
			1	1	1
Roma no	imponía	ideas políticas o credos en sus territorios			
			1	1	1
Los habitantes de la antigua Roma se	ocupaban	en diversos trabajos			
			1	1	1
Los mesopotámicos nos	legaron	la rueda y la escritura			
			1	1	1
El pueblo griego nos	dejó	como herencia la democracia			
			4	4	4
La palabra Mesoamérica	fue creada	por un antropólogo en el siglo xx para definir el lugar en el que florecieron las culturas más desarrolladas de el México antiguo .			
La palabra Mesoamérica fue creada por un antropólogo en el siglo xx para	definir	el lugar en el que florecieron las culturas más desarrolladas de el México antiguo .			
La palabra Mesoamérica fue creada por un antropólogo en el siglo xx para definir el lugar en el que	florecieron	las culturas más desarrolladas de el México antiguo .			
La palabra Mesoamérica fue creada por un antropólogo en el siglo xx para definir el lugar en el que florecieron las culturas más	desarrolladas	de el México antiguo .			
			3	3	3
El preclásico	duró	aproximadamente 2700 años, ya que inició en el 2500 a.C. y concluyó hacia el 200 d.C.			
El preclásico duró aproximadamente 2700 años, ya que	inició	en el 2500 a.C. y concluyó hacia el 200 d.C.			
El preclásico duró aproximadamente 2700 años, ya que inició en el 2500 a.C. y	concluyó	hacia el 200 d.C.			
			1	1	1
El periodo clásico	abarcó	de el 200 al 900 d.C.			
			1	1	1
Para su estudio , el sistema nervioso se	divide	en sistema nervioso central y sistema nervioso periférico			
			3	3	3
El sistema nervioso periférico lo	conforman	los nervios que nacen de el cerebro y de la médula espinal y llegan a todas las partes de el cuerpo por medio de fibras nerviosas .			
El sistema nervioso periférico lo conforman los nervios que	nacen	de el cerebro y de la médula espinal y llegan a todas las partes de el cuerpo por medio de fibras nerviosas .			
El sistema nervioso periférico lo conforman los nervios que	llegan	a todas las partes de el cuerpo por medio de fibras nerviosas .			

nacen de el cerebro y de la médula espinal y					
			<b>3</b>	<b>3</b>	<b>3</b>
El encéfalo se	encuentra	dentro de el cráneo y consta de varios órganos, cada uno de éstos realiza distintas funciones			
El encéfalo se encuentra dentro de el cráneo y	consta	de varios órganos, cada uno de éstos realiza distintas funciones			
El encéfalo se encuentra dentro de el cráneo y consta de varios órganos, cada uno de éstos	realiza	distintas funciones			
			<b>5</b>	<b>5</b>	<b>5</b>
El Cerebro	es	el órgano más grande de el encéfalo, está dividido en dos mitades o hemisferios y presenta hendiduras y pliegues que le dan el aspecto de una nuez pelada			
El Cerebro es el órgano más grande de el encéfalo,	está dividido	en dos mitades o hemisferios y presenta hendiduras y pliegues que le dan el aspecto de una nuez pelada			
El Cerebro es el órgano más grande de el encéfalo, está dividido en dos mitades o hemisferios y	presenta	hendiduras y pliegues que le dan el aspecto de una nuez pelada			
El Cerebro es el órgano más grande de el encéfalo, está dividido en dos mitades o hemisferios y presenta hendiduras y pliegues que le	dan	el aspecto de una nuez pelada			
El Cerebro es el órgano más grande de el encéfalo, está dividido en dos mitades o hemisferios y presenta hendiduras y pliegues que le dan el aspecto de una nuez	pelada				
			<b>8</b>	<b>8</b>	<b>8</b>
El cerebro	almacena	enormes cantidades de información, realiza millones de actividades todos los días y es capaz de llevar a cabo varias acciones a el mismo tiempo, como interpretar lo que los ojos ven, pensar y controlar muchos de los movimientos de el cuerpo			
El cerebro almacena enormes cantidades de información,	realiza	millones de actividades todos los días y es capaz de llevar a cabo varias acciones a el mismo tiempo, como interpretar lo que los ojos ven, pensar y controlar muchos de los movimientos de el cuerpo			
El cerebro almacena enormes cantidades de información, realiza millones de actividades todos los días y	es	capaz de llevar a cabo varias acciones a el mismo tiempo, como interpretar lo que los ojos ven, pensar y controlar muchos de los movimientos de el cuerpo			
El cerebro almacena enormes cantidades de información, realiza millones de actividades	llevar a cabo	varias acciones a el mismo tiempo, como interpretar lo que los ojos ven, pensar y controlar			

todos los días y es capaz de		muchos de los movimientos de el cuerpo			
El cerebro almacena enormes cantidades de información, realiza millones de actividades todos los días y es capaz de llevar a cabo varias acciones a el mismo tiempo, como	interpretar	lo que los ojos ven, pensar y controlar muchos de los movimientos de el cuerpo			
El cerebro almacena enormes cantidades de información, realiza millones de actividades todos los días y es capaz de llevar a cabo varias acciones a el mismo tiempo, como interpretar lo que los ojos	ven	, pensar y controlar muchos de los movimientos de el cuerpo			
El cerebro almacena enormes cantidades de información, realiza millones de actividades todos los días y es capaz de llevar a cabo varias acciones a el mismo tiempo, como interpretar lo que los ojos ven,	pensar	y controlar muchos de los movimientos de el cuerpo			
El cerebro almacena enormes cantidades de información, realiza millones de actividades todos los días y es capaz de llevar a cabo varias acciones a el mismo tiempo, como interpretar lo que los ojos ven, pensar y	controlar	muchos de los movimientos de el cuerpo			
			2	2	2
El cerebro	es	un órgano tan complejo que no se conoce a el detalle su funcionamiento completo			
El cerebro es un órgano tan complejo que no se	conoce	a el detalle su funcionamiento completo			
			4	4	4
El Tálamo se	halla	en el centro de el encéfalo, recibe las señales enviadas por los sentidos y las reenvía a distintas áreas de el cerebro para su procesamiento			
El Tálamo se halla en el centro de el encéfalo,	recibe	las señales enviadas por los sentidos y las reenvía a distintas áreas de el cerebro para su procesamiento			
El Tálamo se halla en el centro de el encéfalo, recibe las señales	enviadas	por los sentidos y las reenvía a distintas áreas de el cerebro para su procesamiento			
El Tálamo se halla en el centro de el encéfalo, recibe las señales enviadas por los sentidos y las	reenvía	a distintas áreas de el cerebro para su procesamiento			
			4	4	4
El Cerebelo	es	el segundo órgano más grande de el encéfalo, sirve para mantener el equilibrio y controlar los movimientos finos			
El Cerebelo es el segundo órgano más grande de el	sirve	para mantener el equilibrio y controlar los movimientos finos			

encéfalo,					
El Cerebelo es el segundo órgano más grande de el encéfalo , sirve para	mantener	el equilibrio y controlar los movimientos finos			
El Cerebelo es el segundo órgano más grande de el encéfalo, sirve para mantener el equilibrio y	controlar	los movimientos finos			
			<b>3</b>	<b>3</b>	<b>3</b>
El Hipotálamo se	encarga	de algunas funciones corporales, como regular la temperatura y percibir la señal de sueño, hambre y sed			
El Hipotálamo se encarga de algunas funciones corporales, como	regular	la temperatura y percibir la señal de sueño, hambre y sed			
El Hipotálamo se encarga de algunas funciones corporales, como regular la temperatura y	percibir	la señal de sueño, hambre y sed			
			<b>1</b>	<b>1</b>	<b>1</b>
El Hipotálamo	es	el responsable de las manifestaciones emocionales (como la amistad, el cariño y el amor)			
			<b>3</b>	<b>3</b>	<b>3</b>
El Bulbo raquídeo	es	el encargado de transmitir mensajes entre el cerebro y el cuerpo, y controla funciones básicas como el latido de el corazón, la digestión y la respiración			
El Bulbo raquídeo es el encargado de	transmitir	mensajes entre el cerebro y el cuerpo, y controla funciones básicas como el latido de el corazón, la digestión y la respiración			
El Bulbo raquídeo es el encargado de transmitir mensajes entre el cerebro y el cuerpo, y	controla	funciones básicas como el latido de el corazón, la digestión y la respiración			
			<b>4</b>	<b>4</b>	<b>4</b>
La Médula espinal	es	la prolongación de el encéfalo, tiene forma de cordón y corre por dentro de la columna vertebral, que la protege			
La Médula espinal es la prolongación de el encéfalo,	tiene	forma de cordón y corre por dentro de la columna vertebral, que la protege			
La Médula espinal es la prolongación de el encéfalo, tiene forma de cordón y	corre	por dentro de la columna vertebral, que la protege			
La Médula espinal es la prolongación de el encéfalo, tiene forma de cordón y corre por dentro de la columna vertebral, que la	protege				
			<b>2</b>	<b>2</b>	<b>2</b>
De la médula espinal	nacen	los nervios periféricos , que permiten movimientos			

		voluntarios e involuntarios , sensaciones y reflejos .			
De la médula espinal nacen los nervios periféricos , que	permiten	movimientos voluntarios e involuntarios , sensaciones y reflejos .			
			<b>142</b>	<b>141</b>	<b>142</b>

TABLA 10 - RESULTADOS DE LA EVALUACIÓN FRENTE A LAS SESENTA Y OCHO ORACIONES TOMADAS DE TEXTOS  
ESCOLARES

## Anexo 4 - Códigos fuente

```
<?php

Route::post('search', 'SearchController@search');
Route::post('extractor', 'ExtractorController@extract');
Route::get('corpus', 'CorpusController@get');
```

FIGURA 23 - RUTAS

```
<?php

use Illuminate\Support\Facades\Schema;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;

class CreateDocumentsTable extends Migration
{
    public function up()
    {
        Schema::create('documents', function (Blueprint $table) {
            $table->increments('id');
            $table->text('arg1');
            $table->text('rel');
            $table->text('arg2');

            $table->string('site');
            $table->string('url');
            $table->string('page');
            $table->string('title');
            $table->string('subtitle');

            $table->text('nps');
            $table->text('ns');
            $table->text('synonymous');
            $table->text('rellnf');

            $table->text('sentence');
        });
    }
}
```

FIGURA 24 - TABLA "DOCUMENTS"

```
<?php

use Illuminate\Support\Facades\Schema;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;

class CreateVisitedSitesTable extends Migration
{
    public function up()
    {
        Schema::create('visitedSites', function (Blueprint $table) {
            $table->string('url');

            $table->index('url');
        });
    }
}
```

FIGURA 25 - TABLA "VISITEDSITES"

```
<?php

use Illuminate\Support\Facades\Schema;
```



```
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;

class CreateQueuedSitesTable extends Migration
{
    public function up()
    {
        Schema::create('queuedSites', function (Blueprint $table) {
            $table->string('url');

            $table->index('url');
        });
    }
}
```

FIGURA 26 - TABLA "QUEUEDSITESTABLE"

```
<?php

use Illuminate\Support\Facades\Schema;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;

class CreateSynonymousTable extends Migration
{
    public function up()
    {
        Schema::create('synonymous', function(Blueprint $table) {
            $table->increments('id');
            $table->string('verb');
            $table->text('synonymous');

            $table->index(['id', 'verb']);
        });
    }
}
```

FIGURA 27 - TABLA "SYNONYMOUS"

```
<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

/**
 * Modelo Document
 * Class Document
 * @package App
 */
class Document extends Model
{
    protected $table = 'documents';

    protected $fillable = [
        'arg1',
        'rel',
        'arg2',
        'site',
        'url',
        'page',
        'title',
        'subtitle',
        'nps',
        'ns',
        'synonymous',
        'rellnf',
    ];
}
```

```
'sentence',  
];  
  
public $timestamps = false;  
}
```

FIGURA 28 - MODELO "DOCUMENT"

```
<?php  
  
namespace App;  
  
use Illuminate\Database\Eloquent\Model;  
  
/**  
 * Modelo VisitedSite  
 * Class VisitedSite  
 * @package App  
 */  
class VisitedSite extends Model  
{  
    protected $table = 'visitedSites';  
  
    protected $fillable = [  
        'url',  
    ];  
  
    public $timestamps = false;  
}
```

FIGURA 29 - MODELO "VISITEDSITE"

```
<?php  
  
namespace App;  
  
use Illuminate\Database\Eloquent\Model;  
  
/**  
 * Modelo QueuedSite  
 * Class QueuedSite  
 * @package App  
 */  
class QueuedSite extends Model  
{  
    protected $table = 'queuedSites';  
  
    protected $fillable = [  
        'url',  
    ];  
  
    public $timestamps = false;  
}
```

FIGURA 30 - MODELO "QUEUEDSITE"

```
<?php  
  
namespace App;  
  
use Illuminate\Database\Eloquent\Model;  
  
/**  
 * Modelo Synonym  
 * Class Synonym  
 * @package App  
 */
```

```
class Synonym extends Model
{
    protected $table = 'synonymous';

    protected $fillable = [
        'verb',
        'synonymous',
    ];

    public $timestamps = false;
}
```

FIGURA 31 - MODELO "SYNONYM"

```
<?php

namespace App\Http\Controllers;

use App\Http\Requests\SearchRequest;
use Illuminate\Support\Facades\Storage;

/**
 * Perform a simple text replace
 * This should be used when the string does not contain HTML
 * (off by default)
 */
define('STR_HIGHLIGHT_SIMPLE', 1);

/**
 * Only match whole words in the string
 * (off by default)
 */
define('STR_HIGHLIGHT_WHOLEWD', 2);

/**
 * Case sensitive matching
 * (off by default)
 */
define('STR_HIGHLIGHT_CASESENS', 4);

/**
 * Overwrite links if matched
 * This should be used when the replacement string is a link
 * (off by default)
 */
define('STR_HIGHLIGHT_STRIPLINKS', 8);

/**
 * Se encarga de parsear la pregunta realizada por el usuario, enviársela a Solr
 * y devolver al usuario los resultados que Solr devuelve
 * Class SearchController
 * @package App\Http\Controllers
 */
class SearchController extends Controller
{
    protected $solarium;

    public function __construct(\Solarium\Client $solarium)
    {
        $this->solarium = $solarium;
    }

    public function search(SearchRequest $request)
    {
        $question = $this->cleanText($request->input('question'));

        $output = collect();
    }
}
```

```
$document = Extractor::fromText($question)
->formatToSolr()
->solrOutput
->first();

if ($document) {
    $questionParsed = $document['arg1'] . ' ';
    $questionParsed .= $document['relInf']->implode(' ') . ' ';
    $questionParsed .= $document['arg2'];
} else {
    $questionParsed = $question;
}

$output->push($this->queryDocument($document));
$output->push($this->withFuzzyMatch($questionParsed, 1));
$output->push($this->withFuzzyMatch($questionParsed, 0));

$output = $this->removeDuplicatedSentences($output->collapse());

return response()->json([
    'data' => [
        'count' => $output->count(),
        'docs' => $output,
    ]
]);
}

public function withFuzzyMatch($question, $fuzzyMatch = 1)
{
    $q = collect();

    foreach (explode(' ', $question) as $word) {
        $q->push($word . '~' . $fuzzyMatch);
    }

    return $this->query($q->implode(' '));
}

public function queryDocument($document)
{
    $output = collect();

    $query = $this->solarium->createSelect();
    $query->createFilterQuery('synonymous')->setQuery($document['synonymous']->implode(' '));
    $query->createFilterQuery('nps')->setQuery($document['nps']->implode(' '));
    $query->createFilterQuery('ns')->setQuery($document['ns']->implode(' '));

    $questionParsed = $document['arg1'] . ' ';
    $questionParsed .= $document['relInf']->implode(' ') . ' ';
    $questionParsed .= $document['arg2'];

    $query->setQuery($questionParsed);

    $hl = $query->getHighlighting();
    $hl->setFields(['sentence']);
    $hl->setSimplePrefix("");
    $hl->setSimplePostfix("");

    $resultset = $this->solarium->select($query);
    $results = json_decode($resultset->getResponse()->getBody(), true)['response'];

    $highlighting = $resultset->getHighlighting();

    foreach ($results['docs'] as $result) {
        $resultHighlighted = $highlighting->getResults()[$result['id']];
    }
}
```

```

        if (isset($resultHighlighted->getFields()['sentence'][0])) {
            $result['sentenceHighlighted'] = $this->str_highlight($result['sentence'], $resultHighlighted->getFields()['sentence'][0]);
        } else {
            $result['sentenceHighlighted'] = $result['sentence'];
        }

        $output->push($result);
    }

    return $output;
}

public function query($text)
{
    $output = collect();

    $query = $this->solarium->createSelect();
    $query->setQuery($text);

    $hl = $query->getHighlighting();
    $hl->setFields(['sentence']);
    $hl->setSimplePrefix("");
    $hl->setSimplePostfix("");

    $resultset = $this->solarium->select($query);
    $results = json_decode($resultset->getResponse()->getBody(), true)['response'];

    $highlighting = $resultset->getHighlighting();

    foreach ($results['docs'] as $result) {
        $resultHighlighted = $highlighting->getResults()[ $result['id'] ];

        if (isset($resultHighlighted->getFields()['sentence'][0])) {
            $result['sentenceHighlighted'] = $this->str_highlight($result['sentence'], $resultHighlighted->getFields()['sentence'][0]);
        } else {
            $result['sentenceHighlighted'] = $result['sentence'];
        }

        $output->push($result);
    }

    return $output;
}

public function removeDuplicatedSenentences($collection)
{
    $results = collect();
    $idx = 0;

    foreach ($collection as $result) {
        $result['sentence'] = trim($result['sentence']);
        $result['sentence'] = preg_replace('!\s+!', ' ', $result['sentence']); // Elimina múltiples espacios y
        deja sólo uno

        if (! $results->where('sentence', $result['sentence'])->first()) {
            if ($similar = $results->where('page', $result['page'])->first()) {
                $result['idx'] = $similar['idx'];
                $results->push($result);
            } else {
                $result['idx'] = $idx++;
                $results->push($result);
            }
        }
    }
}

```

```
$output = collect();

for ($i = 0; $i <= $results->count(); $i++) {
    $similarResults = $results->where('idx', $i);

    if (! $similarResults->count()) continue;

    $out = $similarResults->first();

    if ($similarResults->count() > 1) {
        $c = 0;
        $out['sentenceHighlighted'] = "";
        foreach ($similarResults as $similarResult) {
            if ($c++ != 0) $out['sentenceHighlighted'] .= "<br><br><br>";
            $out['sentenceHighlighted'] .= $similarResult['sentenceHighlighted'];
        }
    }

    $output->push($out);
}

return $output;
}

/**
 * Highlight a string in text without corrupting HTML tags
 * http://www.aidanlister.com/2004/04/highlighting-a-search-string-in-html-text/
 *
 * @author Aidan Lister <aidan@php.net>
 * @version 3.1.1
 * @link http://aidanlister.com/2004/04/highlighting-a-search-string-in-html-text/
 * @param string $text Haystack - The text to search
 * @param array|string $needle Needle - The string to highlight
 * @param bool $options Bitwise set of options
 * @param array $highlight Replacement string
 * @return Text with needle highlighted
 */
function str_highlight($text, $needle, $options = null, $highlight = null)
{
    // Default highlighting
    if ($highlight === null) {
        $highlight = '<strong>\1</strong>';
    }

    // Select pattern to use
    if ($options & STR_HIGHLIGHT_SIMPLE) {
        $pattern = '#(%s)#';
        $sl_pattern = '#(%s)#';
    } else {
        $pattern = '#(?:!<.*?)(%s)(?!^<>]*?>)#';
        $sl_pattern = '#<a\s(?:.*?)>(%s)</a>#';
    }

    // Case sensitivity
    if (! ($options & STR_HIGHLIGHT_CASESENS)) {
        $pattern .= 'i';
        $sl_pattern .= 'i';
    }

    $needle = (array) $needle;
    foreach ($needle as $needle_s) {
        $needle_s = preg_quote($needle_s);

        // Escape needle with optional whole word check
        if ($options & STR_HIGHLIGHT_WHOLEWD) {
            $needle_s = '\b' . $needle_s . '\b';
        }
    }
}
```

```
// Strip links
if ($options & STR_HIGHLIGHT_STRIPLINKS) {
    $sl_regex = sprintf($sl_pattern, $needle_s);
    $text = preg_replace($sl_regex, '\1', $text);
}

$regex = sprintf($pattern, $needle_s);
$text = preg_replace($regex, $highlight, $text);
}

return $text;
}

public function cleanText(string $text)
{
    $text = str_replace('¿', '', $text);
    $text = str_replace('?', '', $text);

    return $text;
}
}
```

FIGURA 32 - CONTROLADOR "SEARCHCONTROLLER"

```
<?php

namespace App\Http\Controllers;

use Illuminate\Support\Facades\Storage;

/**
 * Se encarga de devolver los distintos directorios y archivos que forman el corpus
 * Class CorpusController
 * @package App\Http\Controllers
 */
class CorpusController extends Controller
{
    public function get() {
        return response()->json([
            'data' => array_merge($this->getNode(), ['name' => '/']),
        ]);
    }

    public function getOne($name) {
        return response()->json([
            'data' => [
                'directories' => [
                    'data' => $this->getDirectories($name),
                ],
                'files' => [
                    'data' => $this->getFiles($name),
                ],
            ],
        ]);
    }

    public function getNode($name = null) {
        $output = collect();

        foreach ($this->getDirectories($name) as $directory) {
            $o = collect();
            $directories = $this->getDirectories($directory['name']);

            foreach ($directories as $d) {
                $o->push([
                    'name' => $d['name'],
                ]);
            }
        }
    }
}
```

```
        'children' => $this->getNode($d['name']),
    );
}

$files = $this->getFiles($directory['name']);
foreach ($files as $f) {
    $o->push([
        'name' => $f['name'],
        'isFile' => true,
    ]);
}

$output->push([
    'name' => $directory['name'],
    'children' => $o,
]);
}

if (! count($output)) return null;

return ['children' => $output];
}

public function getDirectories($name = '') {
    $directories = collect();

    foreach (Storage::disk('corpus')->directories($name) as $directory) {
        $directories->push(['name' => $directory]);
    }

    return $directories;
}

public function getFiles($directory = null) {
    $files = collect();

    foreach (Storage::disk('corpus')->files($directory) as $directory) {
        $files->push(['name' => $directory]);
    }

    return $files;
}
}
```

FIGURA 33 - CONTROLADOR "CORPUSCONTROLLER"

```
<?php

namespace App\Http\Controllers;

use Illuminate\Support\Facades\Storage;

/**
 * A partir de un texto o archivo, llama a la clase Extractor para extraer los Documentos
 * Class ExtractorController
 * @package App\Http\Controllers
 */
class ExtractorController extends Controller
{
    public function extract() {
        $filepath = \Request::input('filepath') or null;
        $text = \Request::input('text') or null;

        $json = null;

        if ($filepath) {
            $file = Storage::disk('corpus')->get($filepath);
        }
    }
}
```



```
$json = $this->fromFile($filepath, true, true);
} elseif ($text) {
    $json = $this->fromText($text, false);
}

if (! $json) {
    return response()->json(['data' => []]);
}

return response()->json([
    'data' => [
        'original' => isset($file) ? json_decode($file, true) : $text,
        'parsed' => $json,
    ]
]);
}

public function fromFile($filepath, $saveInDb, $exportToSolr)
{
    $output = collect();

    $json = collect(json_decode(Storage::disk('corpus')->get($filepath), true));

    $site = substr($filepath, 0, strpos($filepath, '/'));

    if (isset($json['sections'])) {
        foreach($json['sections'] as $section) {
            foreach ($section['subsections'] as $subsection) {
                foreach ($subsection['paragraphs'] as $paragraph) {
                    $documents = Extractor::fromText($paragraph)
                        ->withUrl($json['url'])
                        ->withPage($json['name'])
                        ->withTitle($section['title'])
                        ->withSubtitle($subsection['subtitle'])
                        ->withSave($saveInDb)
                        ->withSite($site)
                        ->formatToSolr();

                    if (count($documents)) {
                        $output->push($documents);
                    }

                    if ($exportToSolr) {
                        $documents->exportToSolr();
                    }
                }
            }
        }
    }

    return $output;
} else {
    $json->text = collect($json['text']);

    $page = collect();
    $page['url'] = $json['url'];
    $page['name'] = $json['page'];
    $page['sections'] = collect();

    $section = collect();
    $section['title'] = $json['title'];
    $section['subsections'] = collect();

    $subsection = collect();
    $subsection['subtitle'] = $json['subtitle'];
    $subsection['paragraphs'] = $json['text'];

    $section['subsections']->push($subsection);
}
```

```
$page['sections']->push($section);

foreach($page['sections'] as $section) {
    foreach ($section['subsections'] as $subsection) {
        foreach ($subsection['paragraphs'] as $paragraph) {
            $documents = Extractor::fromText($paragraph)
                ->withUrl($page['url'])
                ->withPage($page['name'])
                ->withTitle($section['title'])
                ->withSubtitle($subsection['subtitle'])
                ->withSave($saveInDb)
                ->withSite($site)
                ->formatToSolr();

            if (count($documents)) {
                $output->push($documents);

                if ($exportToSolr) {
                    $documents->exportToSolr();
                }
            }
        }
    }
}

return $output;
}

public function fromText($text, $save = false)
{
    return Extractor::fromText($text)
        ->withUrl('-')
        ->withPage('-')
        ->withTitle('-')
        ->withSubtitle('-')
        ->withSave($save)
        ->withSite('-')
        ->formatToSolr();
}
```

FIGURA 34 - CONTROLADOR "EXTRACTORCONTROLLER"

```
<?php

namespace App\Jobs;

use App\Http\Controllers\Extractor;
use Illuminate\Bus\Queueable;
use Illuminate\Queue\SerializesModels;
use Illuminate\Queue\InteractsWithQueue;
use Illuminate\Contracts\Queue\ShouldQueue;
use Illuminate\Foundation\Bus\Dispatchable;
use Illuminate\Support\Facades\Storage;

/**
 * Extrae Documentos a partir de un determinado archivo
 * Class ExtractorJob
 * @package App\Jobs
 */
class ExtractorJob implements ShouldQueue
{
    use Dispatchable, InteractsWithQueue, Queueable, SerializesModels;

    protected $filepath;
```

```
protected $saveInDb;
protected $exportToSolr;
protected $file;

public function __construct($filepath, $saveInDb = false, $exportToSolr = false)
{
    $this->filepath = $filepath;
    $this->saveInDb = $saveInDb;
    $this->exportToSolr = $exportToSolr;
    $this->file = Storage::disk('corpus')->get($this->filepath);
}

public function handle()
{
    $this->fromFile($this->filepath, $this->saveInDb, $this->exportToSolr);
}

public function fromFile($filepath, $saveInDb, $exportToSolr)
{
    $output = collect();

    $json = collect(json_decode(Storage::disk('corpus')->get($filepath), true));

    $site = substr($filepath, 0, strpos($filepath, '/'));

    if (isset($json['sections'])) {
        foreach($json['sections'] as $section) {
            foreach ($section['subsections'] as $subsection) {
                foreach ($subsection['paragraphs'] as $paragraph) {
                    $documents = Extractor::fromText($paragraph)
                        ->withUrl($json['url'])
                        ->withPage($json['name'])
                        ->withTitle($section['title'])
                        ->withSubtitle($subsection['subtitle'])
                        ->withSave($saveInDb)
                        ->withSite($site)
                        ->formatToSolr();

                    if (count($documents)) {
                        $output->push($documents);

                        if ($exportToSolr) {
                            $documents->exportToSolr();
                        }
                    }
                }
            }
        }
    }

    return $output;
} else {
    $json->text = collect($json['text']);

    $page = collect();
    $page['url'] = $json['url'];
    $page['name'] = $json['page'];
    $page['sections'] = collect();

    $section = collect();
    $section['title'] = $json['title'];
    $section['subsections'] = collect();

    $subsection = collect();
    $subsection['subtitle'] = $json['subtitle'];
    $subsection['paragraphs'] = $json['text'];

    $section['subsections']->push($subsection);
}
```

```
$page['sections']->push($section);

foreach($page['sections'] as $section) {
    foreach ($section['subsections'] as $subsection) {
        foreach ($subsection['paragraphs'] as $paragraph) {
            $documents = Extractor::fromText($paragraph)
                ->withUrl($page['url'])
                ->withPage($page['name'])
                ->withTitle($section['title'])
                ->withSubtitle($subsection['subtitle'])
                ->withSave($saveInDb)
                ->withSite($site)
                ->formatToSolr();

            if (count($documents)) {
                $output->push($documents);

                if ($exportToSolr) {
                    $documents->exportToSolr();
                }
            }
        }
    }
}

return $output;
}
```

FIGURA 35 - JOB "EXTRACTORJOB"

```
<?php

namespace App\Jobs;

use App\Crawler;
use Illuminate\Bus\Queueable;
use Illuminate\Queue\SerializesModels;
use Illuminate\Queue\InteractsWithQueue;
use Illuminate\Contracts\Queue\ShouldQueue;
use Illuminate\Foundation\Bus\Dispatchable;

/**
 * Realiza WC en una determinada página de Wikipedia
 * Class WikipediaCrawlerJob
 * @package App\Jobs
 */
class WikipediaCrawlerJob implements ShouldQueue
{
    use Dispatchable, InteractsWithQueue, Queueable, SerializesModels;

    protected $url;

    public function __construct($url)
    {
        $this->url = $url;
    }

    public function handle()
    {
        app(Crawler::class)->wikipediaOrg($this->url);
    }
}
```

FIGURA 36 - JOB "WIKIPEDIACRAWLERJOB"

```
<?php

namespace App\Console\Commands;

use App\Jobs\ExtractorJob;
use Illuminate\Console\Command;
use Illuminate\Support\Facades\Storage;

/**
 * Dispara los Jobs que extraerán los Documentos de un determinado sitio
 * Class ExtractorCommand
 * @package App\Console\Commands
 */
class ExtractorCommand extends Command
{
    protected $signature = 'extractor {site}';

    protected $description = 'Dispara los Jobs que extraeran los Documentos de un determinado sitio';

    protected $validSites;

    public function __construct()
    {
        parent::__construct();

        $this->validSites = [
            'turismo.salta.gov.ar',
            'salta.gov.ar',
            'wikipedia.org',
        ];
    }

    public function handle()
    {
        $site = $this->argument('site');

        if (! in_array($site, $this->validSites)) {
            $this->error('Sitio desconocido');
            return;
        }

        $files = Storage::disk('corpus')->files($site);

        foreach ($files as $filepath) {
            dispatch(new ExtractorJob($filepath, true, true));
        }

        $this->info('Jobs disparados');
    }
}
```

FIGURA 37 - COMANDO "EXTRACTORCOMMAND"

```
<?php

namespace App\Console\Commands;

use App\Crawler;
use Illuminate\Console\Command;

/**
 * Realiza WC en un determinado sitio
 * Class CrawlerCommand
 * @package App\Console\Commands
 */
class CrawlerCommand extends Command
{

```

```
protected $signature = 'crawler {site}';

protected $description = 'Realiza WC en un determinado sitio';

public function __construct()
{
    parent::__construct();
}

public function handle()
{
    switch ($this->argument('site')) {
        case 'turismo.salta.gov.ar':
            $this->info('WC turismo.salta.gov.ar');
            app(Crawler::class)->turismoSaltaGovAr();
            $this->info('ok');

            break;

        case 'salta.gov.ar':
            $this->info('WC salta.gov.ar');
            app(Crawler::class)->saltaGovAr();
            $this->info('ok');

            break;

        case 'wikipedia.org':
            $this->info('WC wikipedia.org');
            app(Crawler::class)->wikipediaOrg();
            $this->info('ok');

            break;

        default:
            $this->error('Sitio inválido');
            break;
    }
}
```

FIGURA 38 - COMANDO "CRAWLERCOMMAND"

```
<?php

namespace App;

use App\Jobs\WikipediaCrawlerJob;
use Illuminate\Support\Facades\Storage;
use Weidner\Goutte\GoutteFacade;

/**
 * Ejecuta el WC en un determinado sitio
 * Class Crawler
 * @package App
 */
class Crawler
{
    public function turismoSaltaGovAr()
    {
        $invalids = 0; // Contador de páginas inválidas (tienen que haber 500 páginas inválidas
        CONSECUTIVAS para que pare el crawler)
        $c = 1; // Contador que representa el id de la página a crawlear

        while ($invalids <= 500) {
            $page = collect();
            $page->id = $c;
            $page->url = "http://turismo.salta.gov.ar/contenido/{$page->id}/crawl";
```

```
$page->name = "";
$page->description = "";
$page->title = "";
$page->subtitle = "";
$page->text = collect();

$crawler = GoutteFacade::request('GET', $page->url);

$crawler->filter('#layout > div > div.container > div.box-container-full > div.box-container-titulo > h1')->each(function ($node) use ($page) {
    $page->name = $node->text();
});

$crawler->filter('#layout > div > div.container > div.box-container-full > div.box-container-inner > div > p')->each(function ($node) use ($page) {
    $page->type = $node->text();
});

$crawler->filter('#layout > div > div.container > div.box-container-full > div.box-container-inner > div > h2')->each(function ($node) use ($page) {
    $page->description = $node->text();
});

$crawler->filter('#layout > div > div.container > div.box-container-full > div.box-container-inner > div > div.noticia-contenido > *:not(table)')->each(function ($node) use ($page) {
    $page->text->push($node->text());
});

$crawler->filter('#layout > div > div.container > div.box-container-full > div.box-container-inner > div > h1')->each(function ($node) use ($page) {
    $page->title = $node->text();
});

if ($page->name == 'Noticias') {
    $page->name = $page->title;
}

$page->title = $page->type;

if (count($page->text)) {
    $invalids = 0;

    Storage::disk('corpus')->put("turismo.salta.gov.ar/{$page->id}.json", json_encode([
        'id' => $page->id,
        'url' => $page->url,
        'page' => $page->name,
        'title' => $page->title,
        'subtitle' => $page->description,
        'text' => $page->text,
    ]));
} else {
    $invalids++;
}

$c++;
}

}

public function saltaGovAr()
{
    $invalids = 0; // Contador de páginas inválidas (tienen que haber 500 páginas inválidas CONSECUTIVAS para que pare el crawler)
    $c = 1; // Contador que representa el id de la página a crawllear
    $output = collect();

    while ($invalids <= 500) {
        $page = collect();
```

```

$page->id = $c;
$page->url = "http://www.salta.gov.ar/contenidos/crawl/{$page->id}";
$page->type = 'Acerca de Salta';
$page->page = "";
$page->title = "";
$page->imageUrl = "";
$page->subtitle = "";
$page->date = "";
$page->text = collect();

$crawler = GoutteFacade::request('GET', $page->url);

$crawler->filter('#main > section:nth-child(1) > article > header > h1')->each(function ($node) use ($page) {
    $page->page = $node->text();
    $page->title = $node->text();
});

$crawler->filter('#main > section:nth-child(1) > article > header > h2')->each(function ($node) use ($page) {
    $page->type = $node->text();
});

$crawler->filter('#main > section:nth-child(1) > article > p')->each(function ($node) use ($page) {
    $page->text->push($node->text());
});

$crawler->filter('#main > section:nth-child(1) > article > div.box-dynamic-content')->each(function ($node) use ($page) {
    $page->text->push($node->text());
});

if ($page->title && count($page->text)) {
    $invalids = 0;

    $output->push($page);

    Storage::disk('corpus')->put("salta.gov.ar/{$page->id}.json", json_encode([
        'id' => $page->id,
        'url' => $page->url,
        'page' => $page->page,
        'title' => $page->title,
        'subtitle' => $page->subtitle,
        'type' => $page->type,
        'text' => $page->text,
    ]));
} else {
    $invalids++;
}

$c++;
}

}

public function wikipediaOrg($url = 'https://es.wikipedia.org/wiki/Provincia_de_Salta')
{
    if (VisitedSite::where('url', '=', $url)->first()) return;

    VisitedSite::create([
        'url' => $url,
    ]);

    $page = [
        'url' => $url,
        'name' => "",
        'sections' => collect(),
    ];
}

```



```
$links = collect();

$foundSalta = false;

$crawler = GoutteFacade::request('GET', $page['url']);

$crawler->filter('#firstHeading')->each(function ($node) use (&$page) {
    $page['name'] = $node->text();
    $page['sections']->push($this->initSection($node->text()));
});

$crawler->filter('#mw-content-text h2, #mw-content-text h3, #mw-content-text p, #mw-content-text a')->each(function ($node) use ($page, $links, &$foundSalta) {
    $tag = $node->nodeName();

    switch ($tag) {
        case 'h2':
            if ($node->text() == 'Índice') {
                return;
            }

            $page['sections']->push($this->initSection($node->text()));
            break;

        case 'h3':
            $page['sections']->last()['subsections']->push($this->initSubsection($node->text()));

            break;

        case 'p':
            if (str_contains($node->text(), 'Salta')) $foundSalta = true;

            $page['sections']->last()['subsections']->last()['paragraphs']->push($node->text());

            break;

        case 'a':
            $href = $node->attr('href');

            if (starts_with($href, '/')
                && ! strpos($href, '.')) {
                $links->push('https://es.wikipedia.org' . $node->attr('href'));
            }

            break;
    }
});

foreach ($links as $link) {
    if (! QueuedSite::where('url', '=', $link->first()) {
        QueuedSite::create([
            'url' => $link,
        ]);

        dispatch(new WikipediaCrawlerJob($link));
    }
}

if ($foundSalta) {
    Storage::disk('corpus')->put('wikipedia.asd/'. str_slug($page['name']) . '.json', json_encode($page));
}

public function initSection(string $title)
{
    $section = [
```

```
'title' => $this->cleanSquareBrackets($title),
'subsections' => collect(),
];

$section['subsections']->push($this->initSubsection(""));

return $section;
}

public function initSubsection(string $subtitle)
{
    $subsection = [
        'subtitle' => $this->cleanSquareBrackets($subtitle),
        'paragraphs' => collect(),
    ];

    return $subsection;
}

public function cleanSquareBrackets($text)
{
    $text = preg_replace('#\s*[\.\+]\s*#U', '', $text);
    $text = preg_replace('!\s+!', ' ', $text); // Elimina múltiples espacios y deja sólo uno

    return trim($text) ;
}
}
```

FIGURA 39 - CLASE "CRAWLER"

```
<?php

namespace App\Http\Controllers;

use App\Synonym;
use Illuminate\Support\Facades\Storage;
use Weidner\Goutte\GoutteFacade;

/**
 * Devuelve los sinónimos para un determinado verbo
 * Class Synonymous
 * @package App\Http\Controllers
 */
class Synonymous
{
    /**
     * Obtiene los sinónimos del verbo que se pasó como parámetro desde sinonimos.woxikon.es
     * y lo guarda en la BD
     * @param $verb
     * @return \Illuminate\Support\Collection
     */
    public static function get(string $verb)
    {
        // return collect();

        $verb = str_replace('_', ' ', $verb);

        $fromDb = Synonymous::getFromDb($verb);
        if ($fromDb) return $fromDb;

        $fromCorpus = Synonymous::getFromCorpus($verb);
        if ($fromCorpus->count()) return $fromCorpus;

        $output = collect();

        foreach (explode(' ', $verb) as $subverb) {
            $subverb = str_replace('.', '', $subverb);
        }
    }
}
```

```
    if ($fromDb = Synonymous::getFromDb($subverb)) {
        foreach ($fromDb as $synonym) {
            if ($output->search($synonym)) continue;

            $output->push($synonym);
        }

        continue;
    }

    $subverbSynonymous = collect();

    foreach (Synonymous::sinonimosWoxiconEs($subverb) as $synonym) {
        if ($output->search($synonym)) continue;

        $subverbSynonymous->push($synonym);
        $output->push($synonym);
    }

    foreach (Synonymous::sinonimosOrg($subverb) as $synonym) {
        if ($output->search($synonym)) continue;

        $subverbSynonymous->push($synonym);
        $output->push($synonym);
    }

    Synonym::create([
        'verb' => $verb,
        'synonymous' => $subverbSynonymous->implode(' '),
    ]);
}

if ($output->count()) {
    Synonym::create([
        'verb' => $verb,
        'synonymous' => $output->implode(' '),
    ]);
}

return $output;
}

public static function getFromDb($verb)
{
    $synonym = Synonym::where('verb', '=', $verb)->first();

    if (!$synonym) {
        return null;
    }

    return collect(explode(' ', $synonym->synonymous));
}

public static function getFromCorpus($verb)
{
    $synonymous = collect(json_decode(Storage::disk('corpus')->get('sinonimos.json'), true));

    $output = collect($synonymous->get($verb));

    $output = $output->filter(function($item) {
        // Limpio los valores vacíos

        if (!$item || trim($item) == "") {
            return false;
        }
    });
}
```

```

        return true;
    });

    if ($output->count()) {
        $synonym::create([
            'verb' => $verb,
            'synonymous' => $output->implode(' '),
        ]);
    }

    return $output;
}

public static function sinonimosWoxiconEs($verb)
{
    $synonymous = collect();

    $url = 'http://sinonimos.woxikon.es/es/' . $verb;

    $crawler = GoutteFacade::request('GET', $url);

    $crawler->filter('#content > ol > li.synonyms-list-item > div.synonyms-list-content')
        ->each(function ($nodes) use ($synonymous) {
            $nodes = explode(' ', trim(str_replace(["\r\n", "\n", "\r"], ' ', $nodes->text())));
            foreach ($nodes as $node) {
                if (preg_match('/({.+})\(.+\)[.+] /', $node)) {
                    $synonym = trim(preg_split('/({.+})\(.+\)[.+] /', $node)[0]);
                } else {
                    $synonym = trim($node);
                }

                if (!$synonymous->search($synonym)) {
                    $synonymous->push($synonym);
                }
            }
        });

    return $synonymous;
}

public static function sinonimosOrg($verb)
{
    $synonymous = collect();

    $url = 'http://www.sinonimos.org/' . $verb;

    $crawler = GoutteFacade::request('GET', $url);

    $crawler->filter('body > div:nth-child(2) > div:nth-child(5) > b')
        ->each(function ($node) use ($synonymous) {
            $synonymous->push($node->text());
        });

    return $synonymous;
}
}

```

FIGURA 40 - CLASE "SYNONYMOUS"

```

<?php

namespace App\Http\Controllers;

use App\Document;
use GuzzleHttp\Client;

/**

```

```

* Extrae Documentos a partir de un texto
* También se puede encargar de indexarlo en Solr
* Class Extractor
* @package App\Http\Controllers
*/
class Extractor
{
    protected $freeling;
    protected $guzzle;
    protected $text;
    protected $textAnalyzed;
    public $sentences;
    public $documents;
    protected $previousNp;
    protected $url;
    protected $page;
    protected $title;
    protected $subtitle;
    protected $save = false;
    public $solrOutput;
    protected $site = 'wikipedia';

    protected $export;

    public function __construct(Analyzer $freeling)
    {
        $this->freeling = $freeling;
        $this->guzzle = new Client([
            'base_uri' => env('SOLR_POST_URL'),
        ]);
        $this->sentences = collect();
        $this->documents = collect();
        $this->solrOutput = collect();
    }

    /**
     * Extrae información a partir de un string
     * @param string $text
     * @return mixed
     */
    public static function fromText(string $text)
    {
        $builder = app()->make(Extractor::class);
        $builder->text = $text;
        $builder->textAnalyzed = $builder->analyzeText($builder->text);
        $builder->sentences = $builder->getSentencesFromAnalyzedText($builder->textAnalyzed);

        foreach ($builder->sentences as $sentence) {
            $builder->extract($sentence);
        }

        return $builder;
    }

    /**
     * Extrae un Documento a partir de un array de items WLT.
     * Además de la tupla con ARG1, REL y ARG2, guarda todos los NP y NS que encuentre.
     * @param $items
     * @param null $arg1
     * @param null $nps
     */
    public function extract($wlts, $arg1 = null, $nps = null)
    {
        $document = [
            'arg1' => collect(),
            'rel' => collect(),
            'arg2' => collect(),
        ];
    }
}

```

```
'nps' => $nps?: collect(),
'ns' => collect(),
];

// Contador de paréntesis o corchetes abiertos sin cerrar
$pcC = 0;

// Los símbolos que identifican la apertura de paréntesis o corchetes
$pcA = [
    'Fpa',
    'Fca',
];

// Los símbolos que identifican el cierre de paréntesis o corchetes
$pcT = [
    'Fpt',
    'Fct',
];

$verbIdx = -1;

foreach ($wlts as $wltIdx => $wlt) {
    $tag = $wlt['tag'];

    if (in_array($tag, $pcA)) {
        $pcC++;
    }

    if (in_array($tag, $pcT)) {
        if ($pcC > 0) $pcC--;
    }

    if (preg_match('/^NP.+\/u', $tag)) {
        // NP

        if (!$this->previousNp) {
            // Éste NP será usado para las oraciones con sujeto tácito. Además se agregará a la lista
            // de nps
            $this->previousNp = $wlt;
        }

        $document['nps']->push($wlt);
    } else if (preg_match('/^N.+\/u', $tag)) {
        $document['ns']->push($wlt);
    }

    if (preg_match('/^V.+\/u', $tag) && $pcC == 0) {
        // Verbo que NO está dentro de paréntesis o corchetes

        if ($verbIdx == -1) {
            $verbIdx = $wltIdx;

            $document['rel']->push($wlt);

            if ($wltIdx == 0) {
                // Sujeto tácito

                if ($arg1) {
                    $document['arg1']->push($arg1);
                } else if ($this->previousNp) {
                    $document['arg1']->push([$this->previousNp]);
                }

                $document['nps']->push($wlt);
            } else {
                // No se pudo deducir ningún arg1
            }
        } else {
```

```
// Sujeto expreso
    if ($arg1) {
        $document['arg1']->push($arg1->merge(array_slice($wlts->toArray(), 0, $wltldx)));
    } else {
        $document['arg1']->push(array_slice($wlts->toArray(), 0, $wltldx));
    }
} else if ($wltldx == $verbldx + 1) {
    // Es un verbo de más de una palabra. Se agrega en REL

    $verbldx = $wltldx;

    $document['rel']->push($wlt);
}
}
}

if ($verbldx >= 0) {
    // A partir del verbo, y hacia el final de la oración, forma arg2
    foreach ($wlts as $wltldx => $wlt) {
        if ($wltldx > $verbldx) {
            $document['arg2']->push([
                'word' => $wlt['word'],
                'tag' => $wlt['tag'],
                'lemma' => $wlt['lemma'],
            ]);
        }
    }
}

if (isset($document['arg1'][0])) $document['arg1'] = collect($document['arg1'][0]);

if ($document['rel']->count() && $document['arg2']->count()) {
    // Sólo la agrego si tiene rel ^ arg2 (puede no tener arg1)

    $this->documents->push($document);
}

$this->extract($document['arg2'], $document['arg1']->merge($document['rel']), $document['nps']);
}
}

/**
 * A partir de un texto analizado con Freeling, que contiene en cada línea: WORD LEMMA TAG PROB
 * genera oraciones que en vez de contener las palabras como string, contiene arrays denominados
 "Item WLT",
 * es decir, que contiene: WORD LEMMA TAG.
 * @param $analyzedText
 * @return \Illuminate\Support\Collection
 */
public function getSentencesFromAnalyzedText($analyzedText)
{
    $sentences = collect();
    $sentence = collect();

    $lines = preg_split("/((\r?\n)|(\r\n?))/", $analyzedText);

    foreach ($lines as $line) {
        if ($line == "\n" || $line == "") {
            // Termina una oración
            $sentences->push($sentence);
            $sentence = collect();
        } else {
            $line = explode(' ', $line);

            if (count($line) != 4) {
                // Por alguna razón, Freeling no generó correctamente ésta línea,

```

```

        // ya que no contiene los 4 elementos que debe contener:
        // WORD LEMMA TAG PROB.
        continue;
    }

    $sentence->push([
        'word' => str_replace('_', ' ', $line[0]), // Limpio todos los "_" ya que luego generan problema
al buscar en SOLR
        'lemma' => $line[1],
        'tag' => $line[2],
    ]);
}
}

if ($sentence->isNotEmpty()) {
    $sentences->push($sentence);
}

return $sentences;
}

/**
 * Analiza un texto con Freeling, el cual devuelve por cada palabra o símbolo una línea que contiene:
 * `WORD LEMMA TAG PROB`
 * @param string $text
 * @return string
 */
public function analyzeText(string $text)
{
    return $this->freeling->analyze_text($text);
}

/**
 * Transforma un Documento para enviárselo a SOLR
 * @return $this
 */
public function formatToSolr()
{
    $this->solrOutput = collect();

    foreach ($this->documents as $document) {
        $arg1 = $this->cleanText($document['arg1']->pluck('word')->implode(' '));
        $rel = $this->cleanText($document['rel']->pluck('word')->implode(' '));
        $arg2 = $this->cleanText($document['arg2']->pluck('word')->implode(' '));

        if (! $arg1 && ! $rel && ! $arg2) {
            continue;
        }

        $sentence = $arg1 . ' ' . $rel . ' ' . $arg2;

        if ($this->save) {
            Document::create([
                'arg1' => $arg1,
                'rel' => $rel,
                'arg2' => $arg2,

                'site' => $this->site,
                'url' => $this->url,

                'page' => $this->page,
                'title' => $this->title,
                'subtitle' => $this->subtitle,

                'nps' => join(', ', $document['nps']->pluck('word')->toArray()),
                'ns' => join(', ', $document['ns']->pluck('word')->toArray()),
                'synonymous' => join(', ', Synonymous::get($document['rel']->pluck('lemma')->implode(' '))-

```



```
>toArray()),
    'rellnf' => join(' ', $document['rel']->pluck('lemma')->toArray()),

    'sentence' => $sentence,
  ]);
}

$this->solrOutput->push([
  'arg1' => $arg1,
  'rel' => $rel,
  'arg2' => $arg2,
  'url' => $this->url,
  'page' => $this->page,
  'title' => $this->title,
  'subtitle' => $this->subtitle,
  'sentence' => $sentence,
  'nps' => $document['nps']->pluck('word'),
  'ns' => $document['ns']->pluck('word'),
  'synonymous' => collect(Synonymous::get($document['rel']->pluck('lemma')->implode(' '))),
  'rellnf' => $document['rel']->pluck('lemma'),
  'site' => $this->site,
]);
}

return $this;
}

/**
 * Envía los Documentos generados a SOLR
 * @return $this
 */
public function exportToSolr()
{
    $this->guzzle->post(' ', [
        'headers' => [
            'Content-Type' => 'application/json',
        ],
        'body' => json_encode($this->solrOutput),
    ]);

    return $this;
}

/**
 * Indica la URL de la cual se procesó el texto
 * @param string $url
 * @return $this
 */
public function withUrl(string $url)
{
    $this->url = $url;

    return $this;
}

/**
 * Indica el nombre de la página de la cual se procesó el texto
 * @param string $page
 * @return $this
 */
public function withPage(string $page)
{
    $this->page = $page;

    return $this;
}
```

```
/**
 * Indica el título de la página de la cual se procesó el texto
 * @param string $title
 * @return $this
 */
public function withTitle(string $title)
{
    $this->title = $title;

    return $this;
}

/**
 * Indica el subtítulo de la página de la cual se procesó el texto
 * @param string $subtitle
 * @return $this
 */
public function withSubtitle(string $subtitle)
{
    $this->subtitle = $subtitle;

    return $this;
}

/**
 * Indica si debe guardar los documentos en la BD
 * @param bool $save
 * @return $this
 */
public function withSave(bool $save)
{
    $this->save = $save;

    return $this;
}

/**
 * Indica si debe exportar los Documentos a SOLR
 * @param bool $save
 * @return $this
 */
public function withExport(bool $export)
{
    $this->export = $export;

    return $this;
}

/**
 * Indica el sitio del cual fue procesado el texto. Puede ser:
 * @param string $site
 * @return $this
 */
public function withSite(string $site)
{
    $this->site = $site;

    return $this;
}

/**
 * Elimina todo lo que está dentro de paréntesis o corchetes
 * @param $text
 */
public function cleanText($text) {
    $text = preg_replace('#\s*(.+)\s*#U', '', $text);
    $text = preg_replace('#\s*[.+\]\s*#U', '', $text);
}
```

```
$text = str_replace('(', '$text);  
$text = str_replace(')', '$text);  
$text = str_replace('[', '$text);  
$text = str_replace(']', '$text);  
  
$text = preg_replace('!\s+!', ' ', $text); // Elimina múltiples espacios y deja sólo uno  
  
return $text;  
}
```

FIGURA 41 - CLASE "EXTRACTOR"

## Anexo 5 - Instalación y configuración

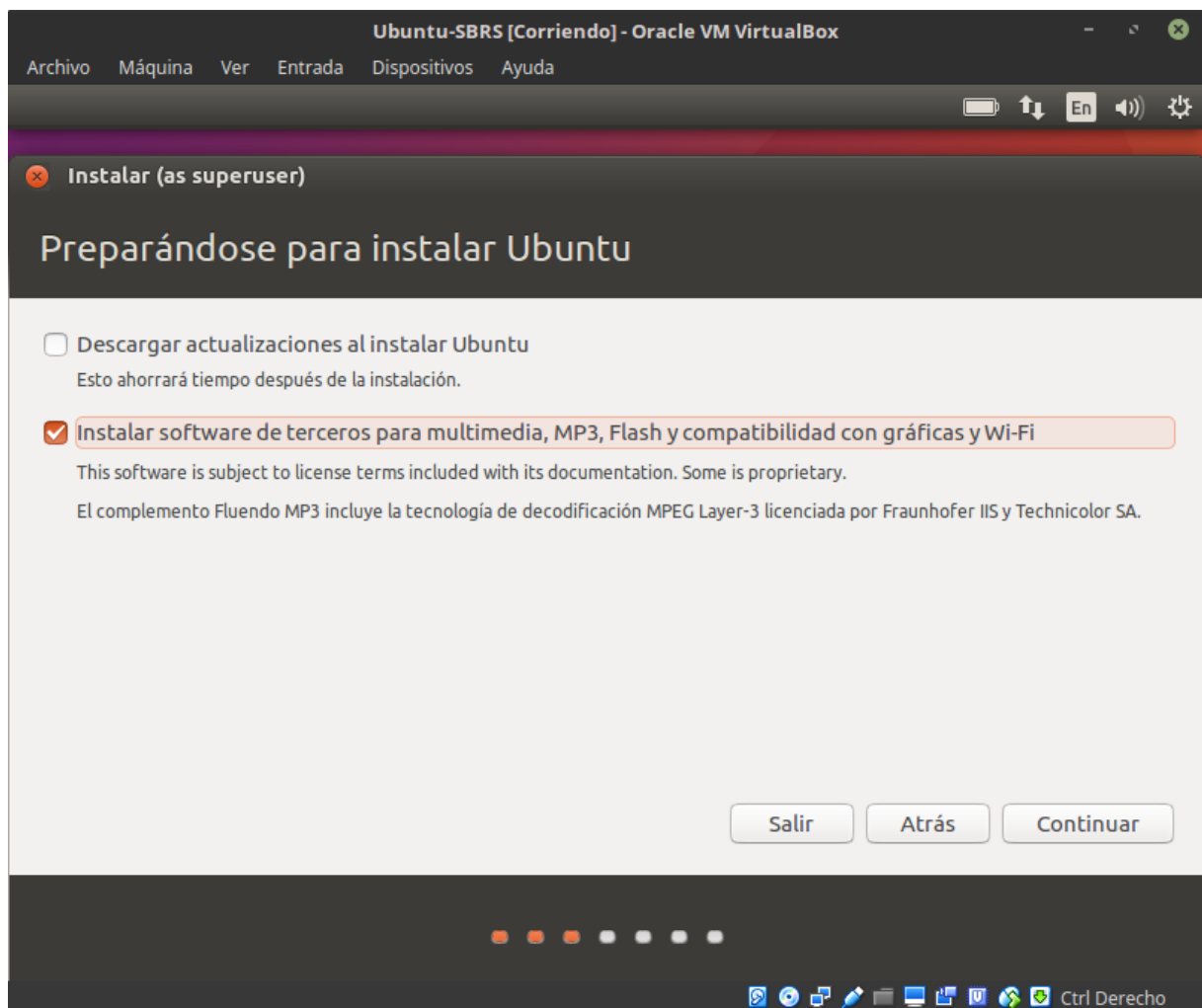
### Instalación del sistema operativo

Para comenzar se debe instalar Ubuntu Desktop 16.04.3 que se puede descargar desde la página oficial<sup>23</sup>. La máquina debe arrancar desde la imagen y a continuación seguir los siguientes pasos de instalación:

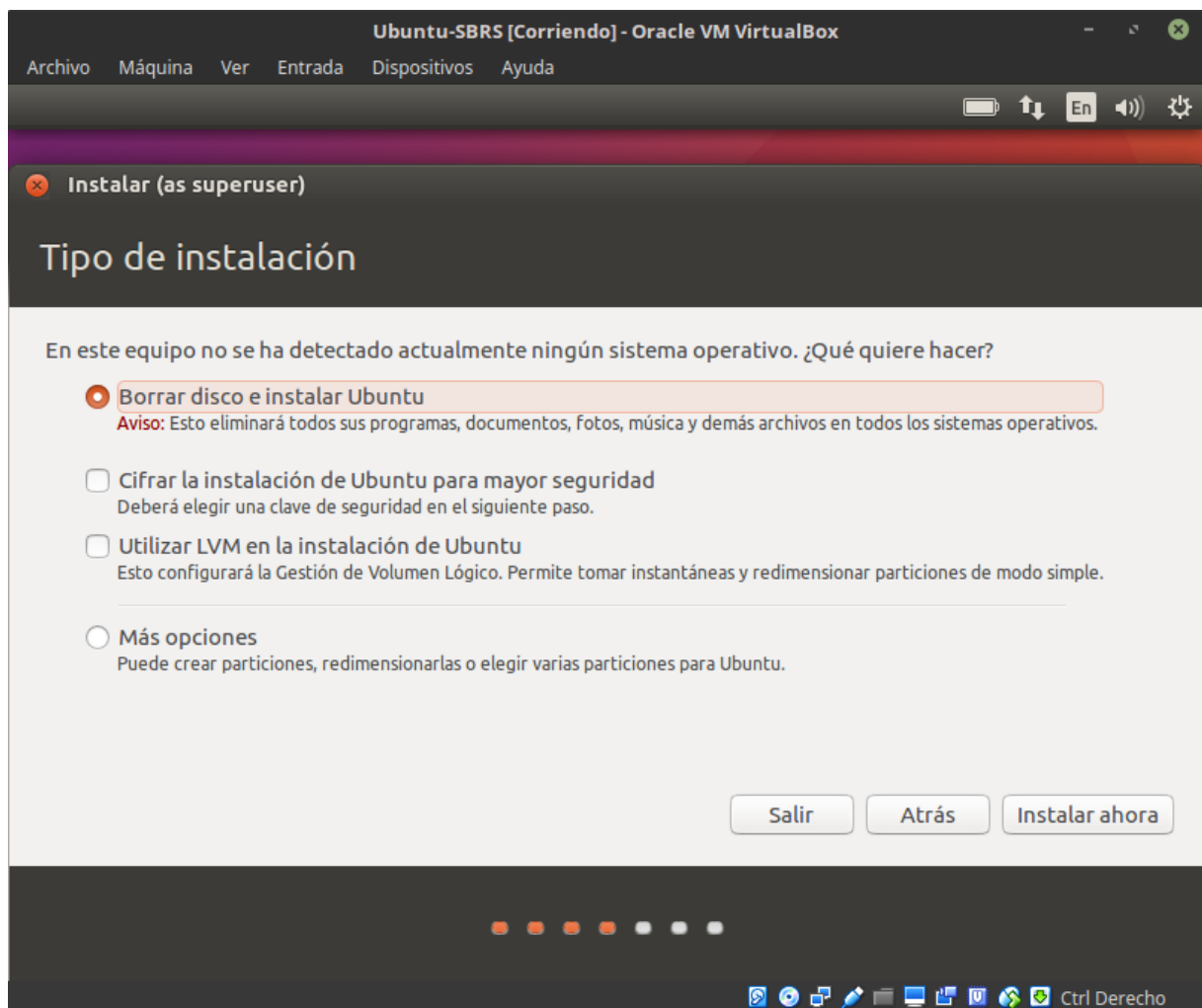


Elegir el idioma y seleccionar “**Instalar Ubuntu**”.

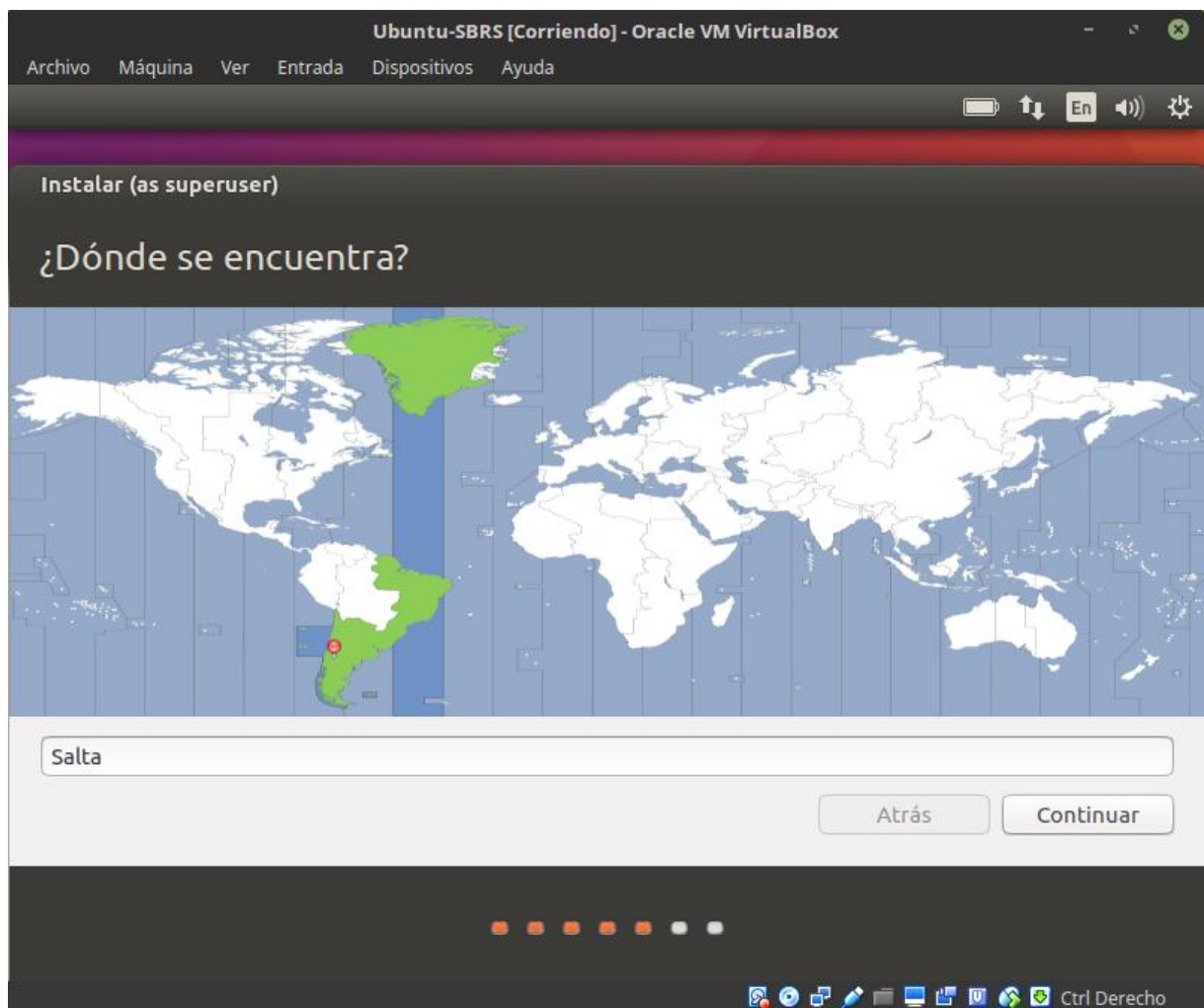
<sup>23</sup> <https://www.ubuntu.com/download/desktop>



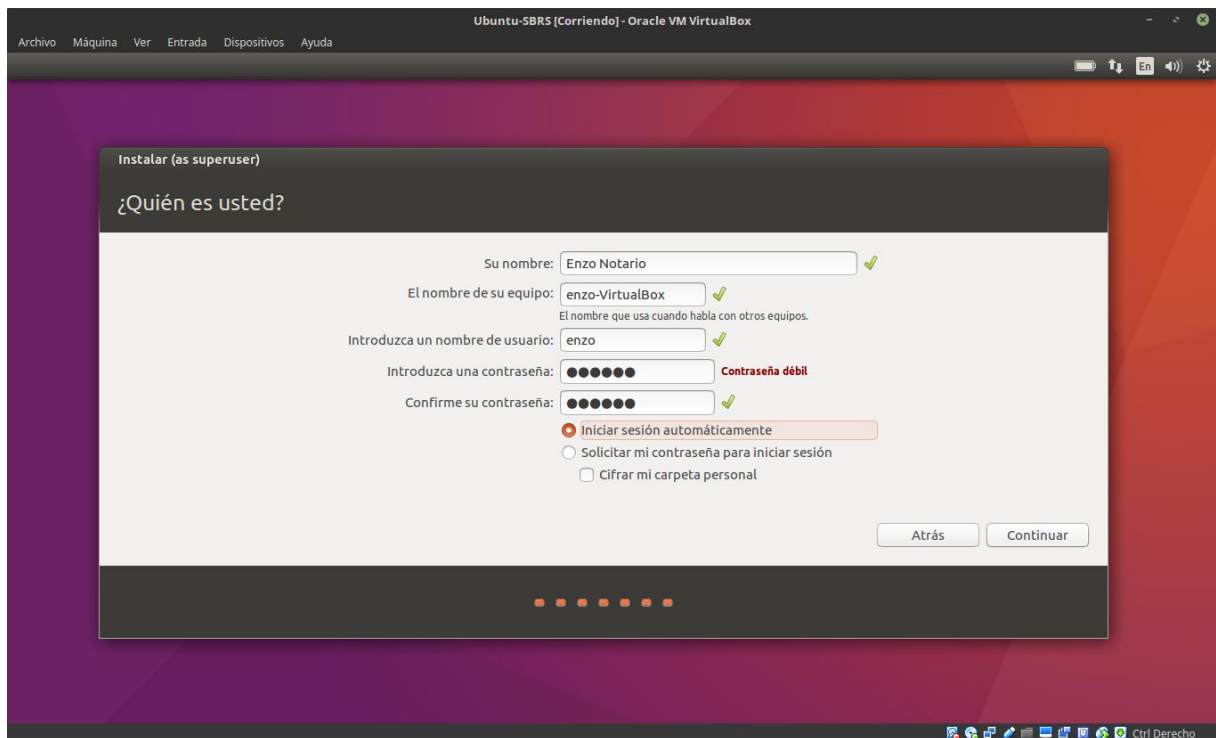
Tildar “Instalar software de terceros para multimedia, MPR, Flash y compatibilidad con gráficas y Wi-Fi” y continuar.



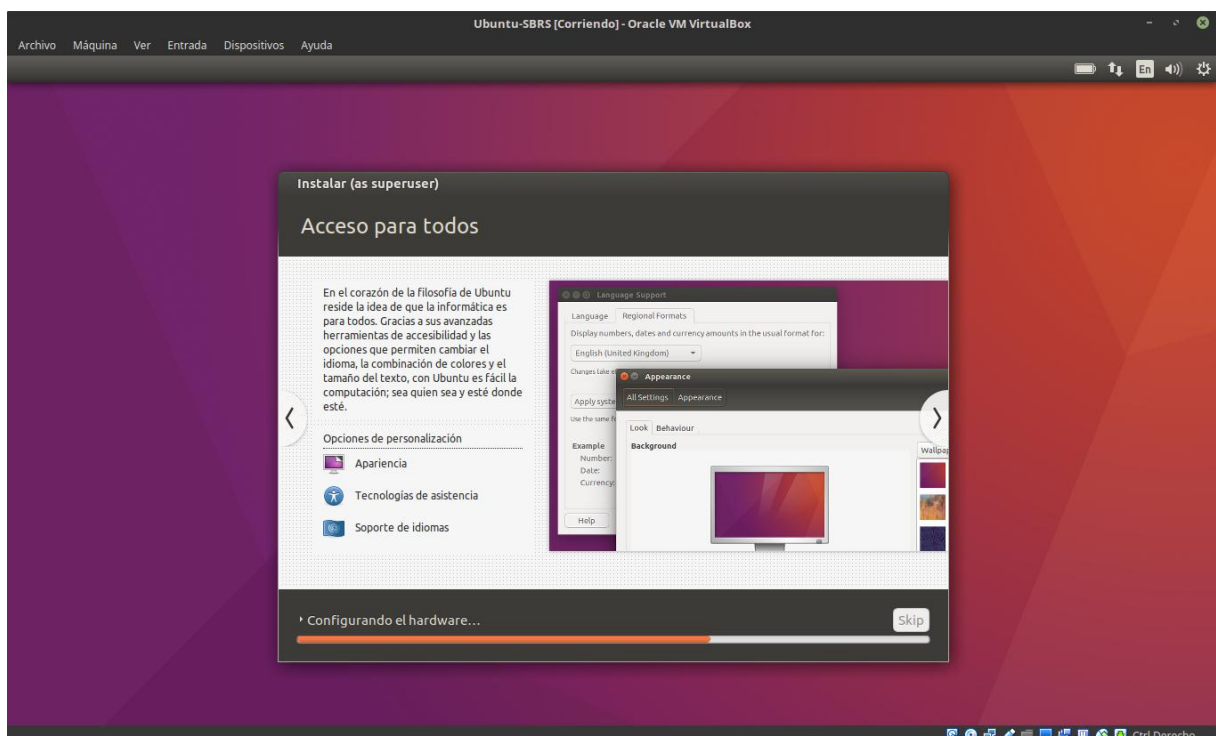
Seleccionar **“Borrar disco e instalar Ubuntu”** y continuar.



Elegir una ubicación y continuar.

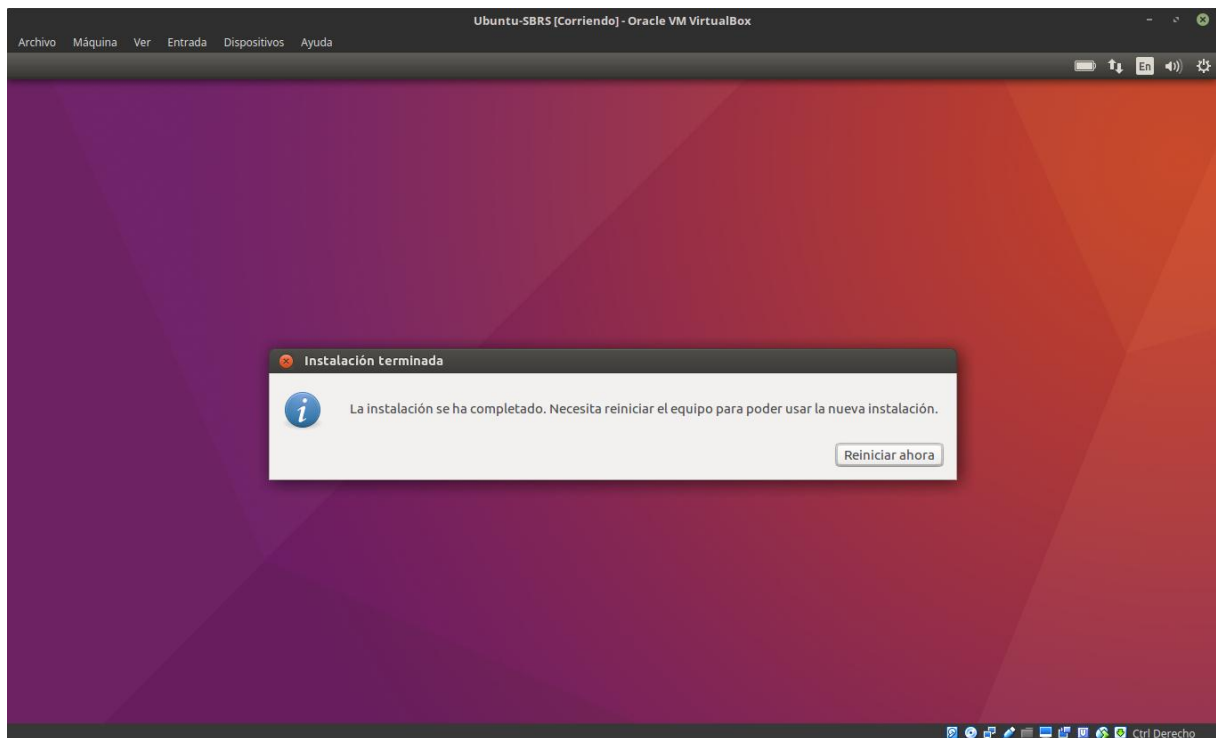


Completar el formulario y continuar.



Ubuntu comenzará a instalarse. Esto demorará unos minutos.





Una vez que se haya instalado, reiniciar.

## Instalaciones generales

Abrir una terminal y ejecutar los siguientes comandos:

Actualizar el repositorio de Ubuntu

```
$ sudo apt-get update
```

Instalar Apache

```
$ sudo apt-get install apache2 -y
```

Instalar MySQL

```
$ sudo apt-get install mysql-server mysql-client -y
```

Instalar PHP y los módulos necesarios

```
$ sudo apt-get install php7.0 libapache2-mod-php7.0 -y  
$ sudo apt-get install php7.0-mysql php7.0-curl php7.0-gd php7.0-intl php-pear php-imagick php7.0-imap php7.0-  
mcrypt php-memcache php7.0-pspell php7.0-recode php7.0-sqlite3 php7.0-tidy php7.0-xmlrpc php7.0-xsl  
php7.0-mbstring php-gettext -y
```

Habilitar el *mod rewrite* de Apache

```
$ sudo a2enmod rewrite
```

Editar el archivo de configuración de Apache ejecutando

```
$ sudo nano /etc/apache2/sites-available/000-default.conf
```

Colocar las siguientes líneas debajo de la línea “*Document Root*”:

```
<Directory "/var/www/html">  
    AllowOverride All  
</Directory>
```

Reiniciar Apache

```
$ sudo service apache2 restart
```

Instalar git

```
$ sudo apt-get install git -y
```

Instalar Java

```
$ sudo apt-get install default-jre
```

## Instalación de Solr

La versión de Solr que se ha utilizado es la 6.1.0 y se puede descargar desde la página oficial<sup>24</sup>. Se debe descargar el archivo .tgz y descomprimirlo en la carpeta home. Luego ejecutar en la terminal:

```
$ solr-6.1.0/bin/solr start -e cloud -noprompt
```

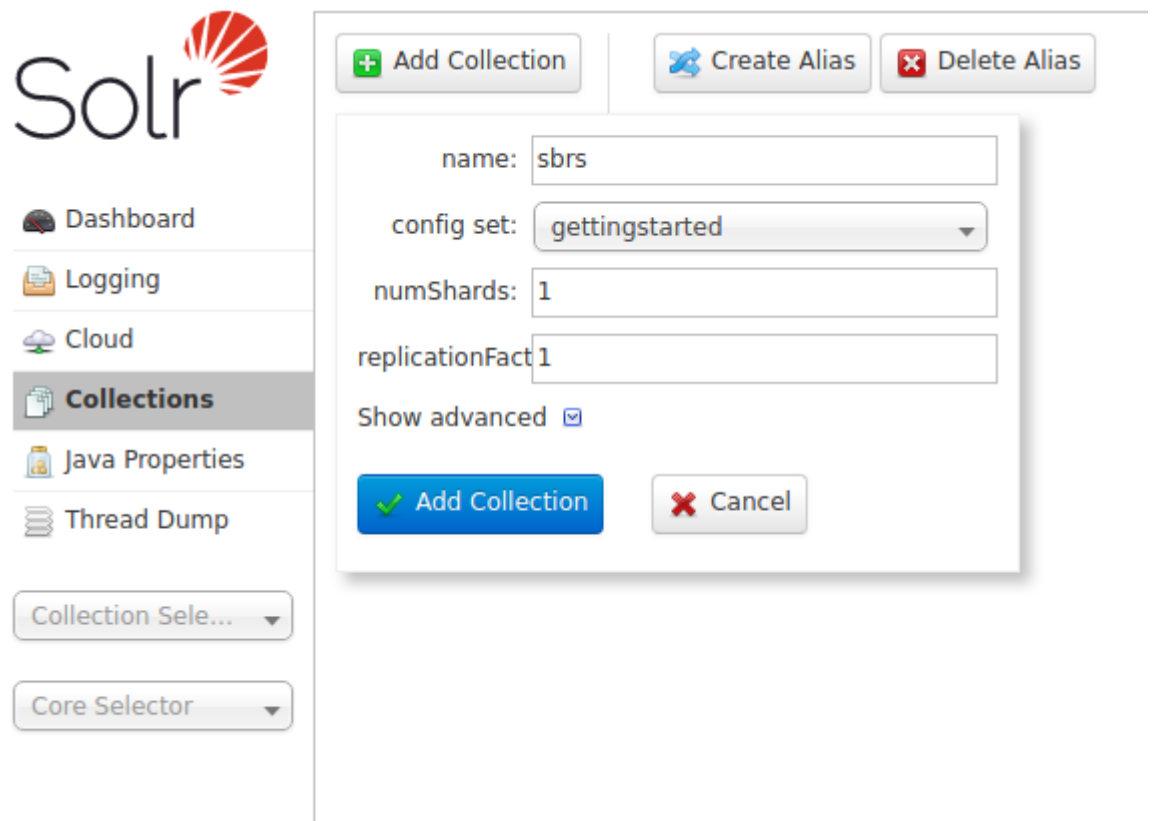
## Configuración

Ingresa a <http://localhost:8983/>.

Dirigirse a “Collections” (en el menú izquierdo), hacer clic en “Add Collection” y completar el formulario:

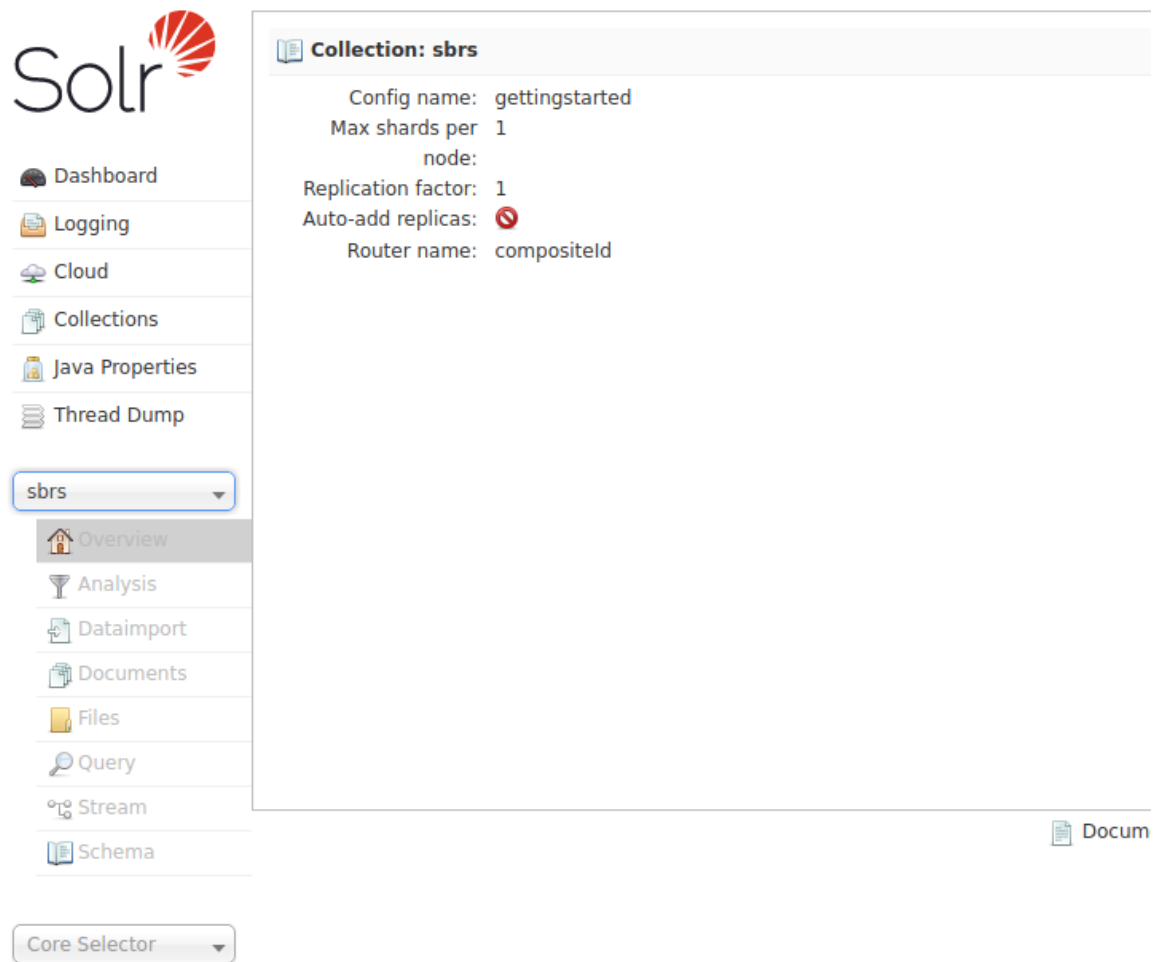
---

<sup>24</sup> <http://archive.apache.org/dist/lucene/solr/6.1.0/>



The image shows the Solr Admin interface. On the left is a sidebar with navigation links: Dashboard, Logging, Cloud, Collections (highlighted), Java Properties, and Thread Dump. Below these are two dropdown menus: 'Collection Sele...' and 'Core Selector'. On the right, a modal dialog titled 'Add Collection' is open. It contains the following fields: 'name' with the value 'sbrs', 'config set' with a dropdown menu showing 'gettingstarted', 'numShards' with the value '1', and 'replicationFactor' with the value '1'. There is a 'Show advanced' checkbox which is checked. At the bottom of the dialog are two buttons: 'Add Collection' (blue with a green checkmark) and 'Cancel' (grey with a red X).

Una vez que se haya creado, en el menú izquierdo elegir la colección que se creó:



The screenshot displays the Solr Admin interface. On the left is a sidebar with the Solr logo and a menu of navigation options: Dashboard, Logging, Cloud, Collections, Java Properties, Thread Dump, a dropdown menu currently showing 'sbrs', and a list of sub-views: Overview (selected), Analysis, Dataimport, Documents, Files, Query, Stream, and Schema. Below the menu is a 'Core Selector' dropdown. The main content area is titled 'Collection: sbrs' and lists configuration parameters: 'Config name: gettingstarted', 'Max shards per node: 1', 'Replication factor: 1', 'Auto-add replicas: [disabled icon]', and 'Router name: compositeId'. A 'Docum' link is visible in the bottom right corner of the main area.

Dirigirse a “Schema” y agregar los siguientes campos haciendo clic en “Add Field”

Add Field
Add Dynamic Field
Add Copy Field

name:

field type:

default:

☒ stored
☒ indexed
☐ docValues
☐ multiValued
☐ required

Show omit options ☒

Show term vector options ☒

Show sort options ☒

Nombre	Tipo	Opciones marcadas
arg1	text_es	stored, indexed
rel	text_es	stored, indexed
arg2	text_es	stored, indexed
url	text_es	stored, indexed
page	text_es	stored, indexed
title	text_es	stored, indexed
subtitle	text_es	stored, indexed
sentence	text_es	stored, indexed
nps	text_es	stored, indexed, multiValued

ns	text_es	stored, indexed, multiValued
synonymous	text_es	stored, indexed, multiValued
relInf	text_es	stored, indexed
site	text_es	stored, indexed

## Instalación de Freeling

Instalar las librerías necesarias:

```
$ sudo apt-get install libboost-program-options1.58.0 libboost-regex1.58.0
```

Descargar FreeLing 4.0 para Ubuntu 16.04 desde el repositorio de GitHub<sup>25</sup> e instalarlo haciendo doble clic sobre el archivo y siguiendo los pasos del instalador.

## Instalación de Laravel

Instalar composer:

```
$ sudo apt-get install curl  
$ curl -sS https://getcomposer.org/installer | php  
$ sudo mv composer.phar /usr/local/bin/composer
```

Clonar el repositorio *sbrs-api*<sup>26</sup>, ingresamos al directorio y ejecutamos:

```
$ composer install
```

Crear la base de datos:

```
$ mysql -u root -p  
mysql> create database sbrs;  
mysql> use sbrs;  
mysql> source sbrs.sql  
mysql> exit
```

Copiar el archivo entorno .env.example a .env

```
$ cp .env.example .env
```

Exportar los Documentos a Solr:

```
$ php artisan export
```

Iniciar el servidor:

```
$ php artisan serve
```

---

<sup>25</sup> <https://github.com/TALP-UPC/FreeLing/releases/download/4.0/freeling-4.0-xenial-i386.deb>

<sup>26</sup> <https://github.com/enzonotario/sbrs-api>

En otra terminal clonar *sbrs-client*<sup>27</sup> y mover el directorio *dist* a */var/www/html*:

```
$ sudo mv sbrs-client/dist/* /var/www/html
```

Clonar el repositorio *sbrs-freeling*<sup>28</sup>, ingresar a la carpeta y ejecutar:

```
$ sh freeling.sh
```


Abrir el navegador, ingresar a <http://localhost/> y utilizar **SBRS**.

---

<sup>27</sup> <https://github.com/enzonotario/sbrs-client>

<sup>28</sup> <https://github.com/enzonotario/sbrs-freeling>

## Anexo 6 – Resolución Rectoral N° 839/13



**RESOLUCIÓN RECTORAL N° 839/13**

En el Campo Castañares, sito en la ciudad de Salta, Capital de la Provincia del mismo nombre, República Argentina, sede de la Universidad Católica de Salta, a tres días del mes de octubre del año dos mil trece:

**VISTO:** La presentación efectuada por el Consejo de Investigaciones; y

**CONSIDERANDO:** Que se trata del Proyecto de Investigación "Minería de textos: búsqueda automática de respuestas" de la Facultad de Ingeniería;  
Que el objetivo general del Proyecto es de continuar la línea de investigación de los trabajos realizados en el marco de las Convocatorias 2007 y 2011 del Consejo de Investigaciones, en los que se desarrollaron y aplicaron técnicas y métodos para apoyar la categorización automática de datos según su contenido; enfocándose en este período la búsqueda automática de respuestas en un corpus de documentos de texto a preguntas formuladas en lenguaje natural;  
Que el mismo esta bajo la dirección de la Dra. Alicia Pérez Abelleira e integra el Equipo de Investigación la Lic. Alejandra Carolina Cardoso;  
Que el tema fue expuesto en reunión de Consejo Académico de fecha 25 de septiembre del corriente año, habiendo recibido dictamen favorable;  
Que es necesario emitir la Resolución Rectoral correspondiente;

**POR ELLO;**

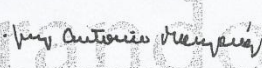
**EL RECTOR DE LA UNIVERSIDAD CATÓLICA DE SALTA**


**RESUELVE:**

**Artículo 1°.- APROBAR** el Proyecto de Investigación "**MINERÍA DE TEXTOS: BÚSQUEDA AUTOMÁTICA DE RESPUESTAS**", cuyo Presupuesto y Proyecto se adjuntan como Anexo I de la presente Resolución.-

**Artículo 2°.- Comunicar** a: Vicerrectorado Académico, Vicerrectorado Administrativo, Vicerrectorado de Formación, Vicerrectorado de Investigación y Desarrollo, Unidades Académicas y Administrativas correspondientes, a los efectos a que hubiere lugar.-

**Artículo 3°.- Registrar, reservar el original y archivar.**

  
JORGE ANTONIO MANZARÁZ  
Rector  
Universidad Católica de Salta





## Anexo 7 – Resolución Rectoral N° 421/15

UNIVERSIDAD CATÓLICA DE SALTA  
SECRETARÍA GENERAL  
18 DIC 2015  
MESA GRAL. DE ENTRADAS  
Correspondencia Gral. N° 13024

UCASAL  
UNIVERSIDAD CATÓLICA DE SALTA  
Instituto de Estudios de Salta - Dirección N° 491/002/09/1992

**RESOLUCION N° 421/15**

En la Facultad de Ingeniería de la Universidad Católica de Salta, a los catorce días del mes de diciembre del año dos mil quince.

**VISTO:** La solicitud efectuada por el Director del Proyecto de Investigación "Minería de textos: Búsqueda automática de respuestas", solicitando la incorporación de un alumno de la carrera de Ingeniería en Informática; y

**CONSIDERANDO:** Que el Proyecto de Investigación citado, ha sido aprobado mediante Resolución Rectoral N° 839/13;

Que la colaboración de alumnos de la carrera de Ingeniería en informática permitirá iniciar al alumno en su formación en investigación, como así también podrá participar en proyectos de especial interés, y que los trabajos realizados durante el desarrollo del Proyecto podrán utilizarse en el fortalecimiento de su formación;

Que la incorporación del alumno se realizará en calidad Ad HONOREM;

Que en el equipo de trabajo del Proyecto de Investigación se prevé la participación de alumnos en la actividad de investigación-desarrollo;

Que corresponde emitir el instrumento legal pertinente;

Por ello,

**EL DECANO DE LA FACULTAD DE INGENIERIA  
RESUELVE:**

**ARTÍCULO 1°.-** DISPONER la incorporación del alumno ENZO NOTARIO – D.N.I. N° 36.358.002 de la carrera Ingeniería en Informática, al Proyecto de Investigación "Minería de Textos: Búsqueda automática de respuestas", aprobado mediante R. R. N° 839/13, en calidad Ad Honorem.-

**ARTÍCULO 2°.-** ESTABLECER que las actividades a realizar por el alumno serán establecidas por el Director del Proyecto mediante un Plan de Trabajo.-

**ARTÍCULO 3°.-** DESIGNAR como Tutores del alumno a la Dra. María Alicia Pérez Abelleira, Director del Proyecto y al Dr. Javier Alberto Moya, Jefe del Departamento de Investigación de la Facultad.-

**ARTÍCULO 4°.-** COMUNICAR a: Secretaría General, Dirección de Alumnos, Jefe Departamento de Ingeniería en Informática, Jefe de Investigación de la Facultad, al Director del Proyecto y al alumno, registrar y archivar.-

Mg. NESTOR EUGENIO LESSER  
Secretario Académico  
Facultad de Ingeniería  
Universidad Católica de Salta

UNIVERSIDAD CATÓLICA  
FACULTAD DE INGENIERIA

Mg. ENZO CARLOS GERARDO SAID  
Decano  
Facultad de Ingeniería  
Universidad Católica de Salta

SEDE CENTRAL: Campo Castañares | Tel.: +54 387 426 8800 | Fax: +54 387 426 8509 | CPA: A4400EDD | Salta, Argentina  
ANEXO CENTRO: Pellegrini 790 | Tel.: +54 387 426 8800 | Fax: +54 387 426 8805 | CPA: A4402FYP | Salta, Argentina  
SUBSEDE BUENOS AIRES: Rep. Dominicana 3586 - B° Palermo | Tel.: +54 11 4897 7444 / 45 | CPA: C1425BMV | Ciudad Autónoma de Bs. As., Arg.