Universidad Zaragoza
1542

Escuela de Ingeniería y Arquitectura

Trabajo Fin de Grado

Grado en Ingeniería Informática
Departamento de Informática e Ingeniería de Sistemas

Sistema de información para generar mapas temáticos a partir de textos con referencias territoriales

Information system to generate thematic maps from texts with territorial references

Autor

Ana Roig Jiménez

Director

David Portolés Rodríguez

Ponente

Raquel Trillo Lado

Escuela de
Ingeniería y Arquitectura
Universidad Zaragoza

Septiembre 2017

TRABAJOS DE FIN DE GRADO / FIN DE MÁSTER

Escuela de Ingeniería y Arquitectura Universidad Zaragoza

DECLARACIÓN DE AUTORÍA Y ORIGINALIDAD

(Este documento debe acompañar al Trabajo Fin de Grado (TFG)/Trabajo Fin de Máster (TFM) cuando sea depositado para su evaluación).

D./Dª. Ana Roig Jiménez,

con nº de DNI 78757854N en aplicación de lo dispuesto en el art. 14 (Derechos de autor) del Acuerdo de 11 de septiembre de 2014, del Consejo de Gobierno, por el que se aprueba el Reglamento de los TFG y TFM de la Universidad de Zaragoza,

Declaro que el presente Trabajo de Fin de (Grado/Máster) Grado, (Título del Trabajo) Text2Map: Sistema de información para generar mapas temáticos a partir de textos con referencias territoriales,

es de mi autoría y es original, no habiéndose utilizado fuente sin ser citada debidamente.

Zaragoza, 25 de agosto del 2017

Fdo: Ana Roig Jiménez

Resumen

La empresa Idearium Consultores dedica gran parte de su trabajo a los campos de Sistemas de información geográficos (Geographical Information Systems, *GIS*, en inglés), datos abiertos (generalmente nombrado por su término en inglés, *Open Data*) y datos enlazados (también usualmente nombrado en inglés *Linked Data*), habiendo desarrollado múltiples aplicaciones en ellos. Una de ellas es Tbl2Map, la cual genera un mapa dada una tabla (en formato CSV, XML, JSON, XLS, o GPX) con la siguiente información: regiones a considerar en el mapa a generar, sus identificadores y los valores para cada región. En este proyecto se aborda la ampliación de Tbl2Map para considerar un documento de texto no estructurado como entrada en lugar de una tabla, puesto que la mayor parte de la información generada por las administraciones públicas y entidades empresariales se encuentra en documentos de texto en diferentes formatos (PDF, DOC, etc.)

Para dar solución a esa necesidad, se ha realizado una aplicación web que recibe un documento de texto como entrada y genera un mapa con las áreas identificadas en él, dado el tipo de área (comarca o municipio). En esta aplicación, el usuario sólo debe seleccionar los valores que desea representar en el mapa (es decir, densidad de población, niveles de contaminación, etc.) entre las diferentes opciones disponibles. Además, también puede configurar el formato del mapa a su gusto (gama de colores, orientación, y formato de salida, entre otros aspectos).

En síntesis, para desarrollar la aplicación se han manejado diferentes herramientas y tecnologías en cada fase del proceso. En primer lugar, se ha realizado la identificación de regiones aparecidas en el documento, mediante Apache Solr para representarlas en el mapa. A continuación, se ha utilizado el lenguaje de consultas SPARQL para obtener desde Aragopedia los valores vinculados a las regiones a representar en el mapa. Finalmente, para integrar Text2Map con la aplicación Tbl2Map, se han programado diferentes paquetes basados en tecnologías web (HTTP, servicios web, etc.) que permiten la transmisión de información conseguida para poder crear el mapa temático

En consonancia con la aplicación brevemente descrita, otro objetivo requerido por la empresa para este proyecto fue la automatización de la generación de los mapas sin intervención del usuario final mediante la interfaz web, es decir, que se pudiese realizar procesos de generación en *batch*. Para ello, se definieron diferentes *scripts* parametrizados que toman como entrada el tipo de área deseada y la una ruta en la que se almacenan los documentos, y generan como salida un mapa por cada documento de la ruta.

La finalidad de esta memoria es comentar detalladamente todas las fases y pasos llevados a cabo para desarrollar e implantar la aplicación propuesta cumpliendo todos y cada uno de los requisitos determinados por la empresa Idearium S.L.

Índice

Índice de figuras

Índice de tablas

1. Introducción

Este proyecto ha surgido a partir de las necesidades de ampliación de los sistemas desarrollados por la empresa Idearium Consultores para las administraciones públicas. En concreto, la empresa consideró que era necesario ampliar las funcionalidades actualmente ofrecidas por el sistema o aplicación Tbl2Map. En mayor detalle, se requiere que dicho sistema, además de recibir como entrada una tabla, pueda trabajar con textos, es decir, con datos de entrada no estructurados, puesto que la mayor parte de las administraciones públicas y empresas de hoy en día publican gran volumen de textos en distintos formatos, como PDF o txt. Para ello, en este proyecto se ha abordado el desarrollo de un sistema o aplicación web para crear una solución que maneje documentos con texto en lenguaje natural de los que se desea obtener información. Esta es la tarea que se tiene como fin en este proyecto: utilizarlo a través de documentos oficiales, normalmente del BOA [1].

A continuación, se muestra en la figura 1 la arquitectura global de la aplicación desarrollada, para ilustrar los componentes principales. Se diferencian claramente tres partes:

La primera se encarga de la identificación de áreas contenidas en el documento dado un tipo de área (comarca o municipio); la segunda obtiene la información a representar en el mapa; y la última genera el mapa de acuerdo a los valores obtenidos previamente y a la configuración del mapa especificada.

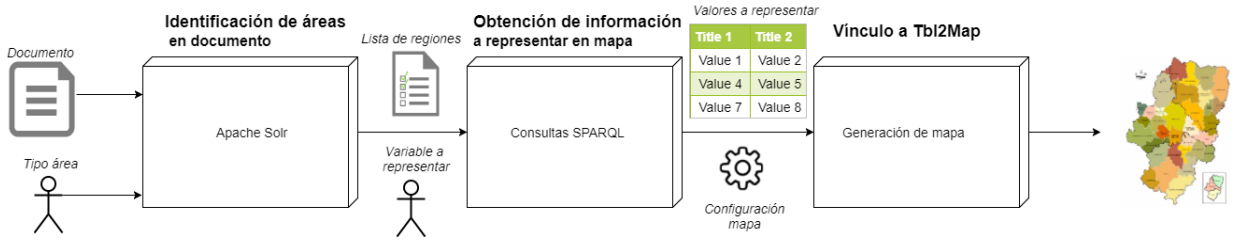

Figura 1: Arquitectura global de la aplicación desarrollada

Otro requisito establecido por la empresa fue la automatización de procesos. Para no requerir la intervención del usuario de forma interactiva, se han elaborado scripts que realizan la misma tarea que la aplicación web sin la conducta de un usuario. Estos scripts pueden recibir más de un documento para analizar y, o bien generar un mapa por cada documento o generar un sólo mapa considerando los datos contenidos en la colección de documentos.

1.1. Motivación

La empresa Idearium S.L. ha identificado la necesidad de visualizar la información y datos de carácter geográfico que contiene un documento. Es decir, considera muy útil el poder reconocer rápidamente las áreas o regiones que se referencian en un artículo y,

además, ser capaz de recoger información sobre estas áreas de una manera sencilla para plasmarla visualmente en un mapa. Por otro lado, no se han localizado aplicaciones o sistemas con el objetivo que tiene este proyecto, por lo que se aborda el desarrollo de una aplicación o sistema que permite realizarlo.

En concreto, la empresa ha requerido como resultado del proyecto una aplicación independiente que puede utilizarse de forma independiente, y también como una parte que pueda ser integrada con otras aplicaciones existentes de la propia empresa.

1.2. Trabajos previos

La aplicación desarrollada ejerce una tarea de apoyo a una aplicación ya existente de la empresa Idearium, además de añadir una funcionalidad nueva y muy útil para su plataforma. Esta aplicación existente es Tbl2Map, la cual recibe como entrada una tabla con datos, entre los que se encuentran el identificador del área, el nombre del área, y un valor asociado a ésta. La aplicación, después de verificar que los datos son correctos, pide una configuración al usuario a través de distintas pantallas, en las cuales se le requiere de indicar el título del mapa, la gama de colores, y el formato de salida (GML, GeoJSON, SHP, PDF y JPG), entre otros aspectos. Una vez concluido este contexto, se realiza una llamada a un servicio web que se encarga de generar el mapa y mostrárselo al usuario, de forma que pueda tener la opción de descargarlo.

En relación con trabajos similares, cabe comentar una aplicación existente utilizada para analizar los subtítulos de la televisión para identificar regiones territoriales y generar un mapa con ellas. No obstante, este sistema no cumple con todos los requisitos propuestos en este proyecto, puesto que solamente realiza la tarea de señalar palabras encontradas, pero no es capaz de recuperar información relacionada con ellas, como sí lo hace la aplicación desarrollada.

1.3. Objetivos

Este proyecto se ha llevado a cabo considerando los requisitos que la empresa Idearium S.L. ha propuesto. Sin embargo, la empresa ha dado libertad en aspectos relacionados con el uso de las tecnologías web y el diseño de las interfaces. Estos aspectos han sido tratados y desarrollados por la estudiante, resueltos utilizando sus conocimientos e intentando integrar lo mejor posible el proyecto con el resto de aplicaciones de la empresa.

Con los requisitos de la empresa en cuanto al proyecto y el objetivo principal que se pretendía conseguir con este trabajo se ha elaborado la tabla 1.

- Objetivo principal. *Desarrollar un sistema de información web para detectar automáticamente referencias territoriales citadas en un documento y crear un mapa que las represente según los valores de una medida.*
- Requisito 1. Identificación de las áreas en un documento.

- Requisito 2. Hacer uso del motor de búsqueda Apache Solr [2].
- Requisito 3. Obtener información estadística para poder representar el mapa.
- Requisito 4. Utilizar consultas RDF [3].
- Requisito 5. Hacer uso de la API de Aragopedia [4].
- Requisito 6. Transmitir la información a Tbl2Map.
- Requisito 7. Acceso automatizado para conseguir un lote de mapas.
- Requisito 8. Acceso automatizado para conseguir un mapa de popularidad de áreas.

Cabe decir que se incluyó un objetivo más cuando ya se habían cumplido todos los anteriores: automatizar el acceso al servicio web de Tbl2Map para facilitar la obtención de un mapa basado en la popularidad de las regiones en los documentos indicados, tratado en su sección 4.3.5. Además, en las diferentes reuniones han surgido requisitos secundarios que se han ido añadiendo dinámicamente después de analizar su dificultad y su utilidad.

- Requisito adicional a. Uso de tecnologías web para desarrollar el sistema.
- Requisito adicional b. Uso del servicio web para integrar diferentes componentes de las aplicaciones existentes.
- Requisito adicional c. Utilizar petición HTTP POST.
- Requisito adicional d. Acceso en formato Shell.
- Requisito adicional e. Manejar el modelo Jenks para tratar los colores.

Referencia	Requisito
1	Identificar referencias territoriales en un documento
2	Utilizar Apache Solr
3	Obtener información estadística para poder representar el mapa
4	Utilizar consultas RDF
5	Hacer uso de la API de Aragopedia
6	Transmitir la información a Tbl2Map
7	Acceso automatizado para conseguir un lote de mapas
8	Acceso automatizado para conseguir un mapa de popularidad de áreas
a	Uso de tecnologías web para desarrollar el sistema
b	Uso del servicio web para integrar diferentes componentes de las aplicaciones existentes
c	Utilizar petición HTTP POST
d	Acceso en formato Shell
e	Manejar el modelo Jenks para tratar los colores

Tabla 1: Requisitos del proyecto

1.4. Planificación del proyecto

Para cumplir con los objetivos y requisitos indicados en la sección anterior, se dividió el proyecto en las siguientes fases:

- **Análisis y diseño**. En esta fase se especificaron los requisitos del sistema a desarrollar (ver sección 4.2), y se realizó un primer diseño de la arquitectura del sistema, como se observa en la figura 8. Una vez se realizó una primera aproximación, los resultados de esta fase fueron validados por el director y ponente del proyecto para tener claro el trayecto a seguir. Esta fase se desarrolló de forma intensa la primera semana del proyecto. No obstante, a lo largo de todo el proyecto se revisaron y aplicaron tanto los requisitos como la arquitectura inicial.

- **Adquisición de conocimientos**. El estudio de las tecnologías a utilizar, las herramientas, y el análisis de la documentación de éstas supuso una ardua labor influyente que ocupó la mayor parte de las semanas 2 y 3 del proyecto, debido a la inexperiencia de la estudiante en el campo de Recuperación de información.

- **Implementación**. En cuanto al desarrollo técnico se refiere (ver sección 4.3), se identificaron tres grandes bloques que dividían al proyecto:

  1. Identificación, indexación y búsqueda de referencias territoriales en documentos.
  2. Recuperación de información de Aragopedia para representarlas en el mapa.
  3. Transmisión de toda la información generada a la aplicación Tbl2Map para generar el mapa.

  Esta fase ha sido la más costosa y de más relevancia del proyecto, y se ha tratado de desarrollar el código lo más reutilizable posible, dividiéndolo en diferentes componentes que se puedan integrar y reutilizar en otras aplicaciones. Su realización ha conllevado, aproximadamente, 8 semanas.

- **Implantación y pruebas**. Cabe comentar que el proceso de pruebas (ver sección 4.4), aunque al final del proyecto se realizó una fase completa de testing, se fue elaborando a través de diferentes verificaciones a lo largo de todo el desarrollo (ver diagramas de Gantt en anexo E), probando unitariamente los componentes que se iban implementando para comprobar su correcto funcionamiento.

- **Documentación técnica**. De la misma manera, la elaboración de la memoria ha sido una fase que ha estado presente desde el inicio hasta el final del proyecto. Sin embargo, ha sido en las últimas semanas del proyecto cuando más se ha incidido en ella.

1.5. Estructura de la memoria

En este capítulo se va a comentar la estructura del resto de la memoria, que es la siguiente: En el capítulo 2, se describe el contexto tecnológico en el que se desarrolla

el proyecto, se analiza la evolución de los sistemas web desde su nacimiento hasta la actualidad, y cómo ha influido ésta en el desarrollo de aplicaciones web.

A continuación, en el capítulo 3, se describen la metodología utilizada en el desarrollo del proyecto, las herramientas empleadas, el proceso llevado a cabo para alcanzar los objetivos, y las diferentes reuniones con el director y ponente del proyecto. Constituyeron el punto de partida para el desarrollo de la aplicación Text2Map.

Posteriormente, en el capítulo 4, se detalla la aplicación web construida, explicando el proceso de desarrollo. En primer lugar, se realiza un análisis y diseño. A continuación, se describen la fases de implementación y pruebas, así como los resultados obtenidos. Este apartado es el que más peso tiene del proyecto.

Finalmente, en el último capítulo, se explican las conclusiones del proyecto, incidiendo los posibles problemas encontrados, soluciones elegidas, y se realiza una valoración personal de éste. Por último, se indican posibles líneas de trabajo futuro para ampliar la aplicación desarrollada.

2. Contexto tecnológico

La evolución de los sistemas web ha traído cambios considerables en los últimos años. En general, se identifican 4 etapas o eras en la evolución de la Web desde que esta surgió a finales de la década de los 80, en el centro europeo de investigación nuclear (European Organization for Nuclear Research -CERN-, en inglés).

La era de la *Web estática*. Esta etapa se desarrolló entre los años 1989-1998 aproximadamente, y tenía como objetivo principal compartir e intercambiar documentos para facilitar el consumo de información. Esta fase también se conoce con el nombre de Web de sólo lectura, ya que los usuarios visualizaban páginas web, generalmente publicadas por grandes administraciones y empresas, a través de sus navegadores.

La era de la *Web dinámica*. Esta etapa se caracteriza por tratar de integrar la información procedente de los sistemas de gestión de las entidades con la publicada en la Web. Esto produce un cambio tecnológico, pero no de filosofía, pues se sigue considerando una web de sólo lectura. Es en esta etapa cuando surgen tecnologías que permiten ejecutar código en los servidores web, como por ejemplo JSP y PHP. En esta etapa se produce un gran incremento de sitios web disponibles para su consulta, debido a la popularización de la tecnología. Así, por ejemplo, numerosos profesionales crean su propia página personal. Debido al aumento del volumen de documentos y sitios disponibles, se desarrollan tecnologías y herramientas para facilitar la localización de las páginas de interés a los usuarios finales, como por ejemplo, los directorios y los motores de búsqueda Yahoo! y Google.

La era de la *Web social*. Con esta etapa se produce un cambio de filosofía, no sólo tecnológico como hasta entonces, puesto que la mayor parte de los usuarios se convierte en proveedores de información y datos, y no sólo en lectores/consumidores de ésta. Por ello, esta fase también se conoce con el nombre de Web de lectura y escritura. En esta fase surgen las plataformas sociales que actualmente están muy avanzadas, como Youtube, Facebook, Twitter, etc.

La era de la *Web de datos*. Esta etapa o fase es en la que nos encontramos actualmente. El objetivo principal de esta fase es crear una Web en la que agentes software consumen datos de forma automática, es decir, que la Web no sólo esté orientada a que la lean y consuman personas, sino también procesos software de forma automática. Esta etapa comenzó a principios de milenio y persigue que no sólo se referencien documentos, sino también los datos disponibles dentro de dichos documentos. Para ello, se han desarrollado numerosas tecnologías, como por ejemplo, RDF y SPARQL.

En más detalle, la gran cantidad de datos que se almacena hoy en día en la Web hace que la búsqueda de información concreta genere resultados no deseados o que no se están buscando. Por ello, uno de los objetivos de la Web de datos es que las máquinas sean capaces de entender qué se le está pidiendo y ejecuten acciones para lograr lo que se le requiere. Para ello, es necesario una semántica común entre computadores y personas, necesitando tres tecnologías clave: los vocabularios u ontologías comunes que constituyen un marco común, es decir, que nos permitan "hablar" en el mismo idioma; modelos y lenguajes que nos permitan publicar datos (por ejemplo, el estándar de RDF); y lenguajes de consulta para recuperar información de las nuevas fuentes de

datos, como por ejemplo, SPARQL para obtener fuentes RDF.

2.1. Herramientas

El proyecto basa su parte más importante en el tratamiento de documentos para permitir localizar las áreas geográficas que aparecen en ellos, por lo que la herramienta *Apache Solr* [2] tiene una importancia de enorme peso en su desarrollo. Solr se ha encargado por completo de la indexación de documentos y búsqueda de términos que interesan, y se ha intentado aprovechar al máximo su gran potencia. Es una herramienta muy rápida que permite la localización de palabras solicitadas a través de varios ajustes en la configuración de su esquema. La premisa fundamental de Solr es sencilla: el usuario le da mucha información en forma de documentos, y después puede hacerle consultas y preguntarle por partes de información que se requieren. La parte en la que se alimenta a Solr con información es llamada *indexado*, y cuando se le pregunta, se llama *consulta*.

Por otro lado, existe una gran consideración en cuanto a la parte relacionada con *SPARQL*, el lenguaje de consulta de grados RDF, mediante la cual se ha podido llevar a cabo la búsqueda de información referente a las áreas geográficas encontradas en el texto. Este proceso se ha realizado apoyándose en el API proporcionado por Open Data Aragón, una iniciativa que dispone de contenidos territorializados, ya que es una fuente oficial de información y muy generosa/rica en cuanto al contenido ofrecido.

Las aplicaciones web emergen desde finales de la década de los 90 para no requerir la instalación local de Sistemas web específicos más allá de un navegador, y están compuestas generalmente por dos grandes componentes: el *frontend* y el *backend*. En el backend se encuentra el servidor web, al que se le realiza peticiones desde el frontend para realizar operaciones, recuperar información y devolverla al frontend, para mostrarla al usuario. El conjunto formado por las tecnologías AngularJS y Bootstrap está siendo muy utilizado en la mayoría de sitios web por añadirle facilidad, dinamismo y novedad al frontend, apoyado sobre un backend basado en Java mediante el framework Spring, el cual también aporta comodidad para manejar operaciones realizadas el servidor. Estas tecnologías han sido la clave de la estructura web de la aplicación desarrollada.

3. Metodología

En este capítulo, en primer lugar se describe la metodología/enfoque aplicada en el desarrollo del proyecto y cómo se ha llevado a cabo éste. A continuación, se describe el conjunto de herramientas que han hecho posible el desarrollo de una manera más ágil y confortable.

3.1. Metodología del proyecto

El trabajo actual se ha sostenido en una metodología ágil en la cual se ha incidido en la adaptación de los avances del proceso más que en la predicción, además de ser más orientada al personal que al proceso. El uso de este tipo de metodologías permite una generación de prototipados rápidos y versiones previas a la entrega final, y es menos arriesgado el poder realizar modificaciones en el diseño, tal y como se indica en el artículo [5].

Conviene recalcar que el proyecto ha seguido un desarrollo en espiral, en el cual se producen múltiples iteraciones de cada ciclo de actividades, sin pasar al siguiente ciclo hasta que no se ha finalizado correctamente el actual. Además, en cada uno de estos ciclos se han establecido cuatro pasos esenciales que permiten adaptar las diferentes actividades o tareas:

- **Determinar los objetivos.** Se fijan las metas que se quieren lograr en ese ciclo, definiendo la situación ideal.
- **Análisis del riesgo.** Se realiza un estudio de las dificultades identificadas en el ciclo y sus posibles soluciones y estrategias para reducirlas y evitarlas. La bibliografía utilizada en esta fase fue muy extensa, analizando tanto páginas oficiales de la distribución, consumidas para aprender el funcionamiento, como foros no oficiales, empleados para resolver problemas.
- **Desarrollar y validar.** Se ejecuta el desarrollo de las tareas o actividades planificadas en los pasos anteriores.
- **Comprobar o verificar.** Por último, se valida si es posible continuar con el ciclo posterior o son necesarios cambios en el ciclo actual.

El desarrollo del proyecto se dividió en tres grandes bloques (ver figura 7), y cada uno de ellos se fragmentó en varios ciclos como los que se acaba de comentar, para llevar a cabo satisfactoriamente cada tarea tanto por separado como en conjunto. Con cada iteración alrededor de la espiral, se crean distintas versiones de la aplicación, cada vez más completas, hasta que se tiene el resultado final deseado.

El modelo en espiral adoptado permite una comunicación con el cliente final constante para poder adaptar las funcionalidades desde el principio hasta el final del trabajo. Con el avance de la aplicación, se mostraba el trabajo realizado desde el punto anterior, habiendo atendido a los aspectos considerados en reuniones precedentes. De esta manera, se recibía la retroalimentación necesario para poder, tanto mejorar lo realizado,

como avanzar en fases posteriores. Normalmente, se hacían reuniones con el ponente cada semana, si había trabajo relevante que enseñar. Estas reuniones servían para poder avanzar más rápidamente, resolver dudas y solucionar problemas que podían paralizar momentáneamente el progreso del proyecto.

Otro de los rasgos importantes es la planificación, puesto que es imprescindible conocer los recursos y el tiempo que se van a necesitar, y ser capaz de añadir más informaciones relevantes conforme progresa el proceso. Además, el estudio de los riesgos que se incluye en este método es fundamental para evitar problemas y conflictos posteriores que se podrían haber remediado.

3.2. Herramientas utilizadas en el desarrollo

En este apartado se describen las herramientas de desarrollo que se han empleado para la construcción y despliegue del proyecto, mientras que los *frameworks*, librerías y lenguajes núcleo de la solución se describen en el del capítulo 2.1.

IDE: Tanto para la parte de implementación del desarrollo web como para llevar un control de versiones del proyecto, se ha hecho uso del IDE *IntelliJ IDEA* [6] de JetBrains. Este entorno hace posible de una manera cómoda el progreso de la aplicación web, aportando ventajas y opciones para el desarrollador, como por ejemplo la depuración paso a paso para identificar posibles errores, o el autocompletado de diferentes elementos muy comunes.

Control de versiones: El IDE IntelliJ IDEA también aporta la posibilidad de realizar un control de versiones. Esta posibilidad se ha explotado para sacar el máximo partido a una gestión organizada del código. Esta gestión se ha llevado a cabo a través de *GitHub* [7], por ser la plataforma más utilizada actualmente en el ámbito de desarrollo web y por tener conocimiento pleno de su funcionamiento. La oportunidad de poder integrar el control de versiones dentro del IDE en el que se trabaja continuamente hace mucho más fácil organizar el proyecto en diferentes fases y tener cada versión del proyecto en el repositorio GitHub.

SonarQube: Para la generación de estadísticas del código, se ha hecho uso de la plataforma *SonarQube* [8]. Esta plataforma de software libre permite identificar errores, proveer sus soluciones y mejorar el código desarrollado basándose en estándares oficiales. Más concretamente, se ha aprovechado el *plugin* existente *SonarLint* para el IDE utilizado, el cual provee retroalimentación instantánea para encontrar advertencias, errores y limpieza de código, junto a las soluciones estandarizadas para solventar todo código no correcto. De esta manera, se obtiene un código limpio, con soluciones de estándares de Java, en este caso. En la figura 2 se puede contemplar una muestra del tipo de estadísticas ofrecidas por SonarQube, aunque se ha explotado la mayor parte del tiempo el plugin SonarLint, por poder disfrutarlo desde el entorno de desarrollo.

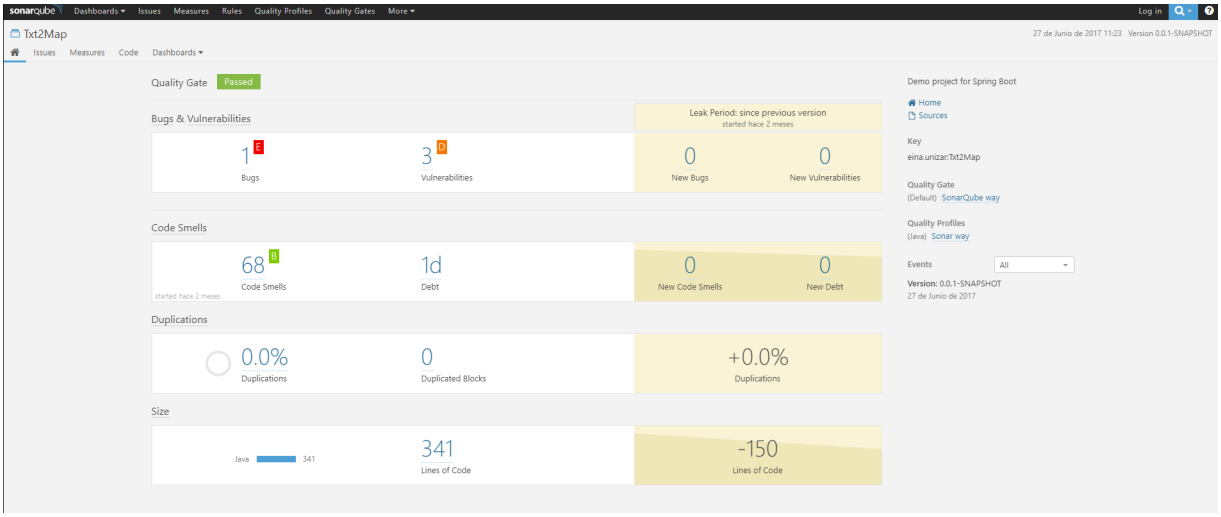

Figura 2: Estadísticas ofrecidas por SonarQube

Documentación: Por último, indicar que para la generación de la documentación técnica del código desarrollado se ha empleado la herramienta *Javadoc* [9], una herramienta de generación de documentación en formato HTML para software desarrollado en Java. En concreto, se crea una página web por cada clase, y en ella se especifican las características de la clase y sus métodos.

4. Aplicación web desarrollada

En este capítulo, se describen las diferentes fases que componen el desarrollo del proyecto, formado por el análisis y diseño, implementación y pruebas.

4.1. Descripción y funcionamiento

La aplicación desarrollada en este proyecto tiene como nombre **Text2Map**, refiriéndose a "de texto a mapa", literalmente. Su funcionamiento se basa en recibir un documento de texto con formato PDF, word, o .txt por parte del usuario, procesarlo para encontrar en él áreas geográficas de Aragón según el tipo de área que el usuario elige (comarca o municipio), y permitir al usuario elegir un parámetro de información y un período de tiempo de dicho parámetro en las áreas geográficas localizadas previamente. El usuario puede decidir si descargar la información obtenida como un fichero excel o bien reflejarlo en un mapa temático.

En resumen, el objetivo de la aplicación es poder representar de una manera rápida las áreas aragonesas existentes en un texto, y además poder obtener una información real y fiable de un determinado período para cada área de una forma sencilla y eficaz. En las figuras 3, 4, 5 y 6 se puede ver todo el proceso a seguir por parte del usuario desde su acceso a la aplicación hasta la generación del mapa, mediante un mapa de navegación.

4.2. Análisis y diseño

Al inicio del proyecto, se realizó un análisis global de la aplicación desarrollada, determinando los puntos fuertes en los que se debía intensificar el trabajo. Se describieron los requisitos funcionales que debía cumplir la aplicación, así como los no funcionales. Los requisitos del proyecto fueron determinados por el director del proyecto, a su vez jefe de la empresa para la cual se realizaba este proyecto, Idearium S.L., para que la aplicación desarrollada fuera consistente con los demás productos de la organización. No obstante, la estudiante debatió y sugirió al director nuevos requisitos y funcionalidades.

El planteamiento inicial de la aplicación se realizó a través de diagramas para tener una idea general de la arquitectura global del sistema. En la figura 7 se puede contemplar este diagrama. De la misma manera, se desarrollaron tanto diagramas de la arquitectura global como diagramas técnicos para entrar en más detalle en aspectos relacionados con la implementación de los componentes o paquetes de la arquitectura (ver anexos 5.4 para más detalle de los componentes). Estos diagramas ayudaron a comprender el debido funcionamiento de la aplicación en una fase inicial del proyecto.

En la fase de diseño, se plantearon diferentes alternativas para abordar la confirmación e implementación de cada uno de los componentes reflejados en los diagramas comentados anteriormente, considerando las principales ventajas e inconvenientes de cada una de las alternativas. Finalmente, se optó por un diseño reflejado en la figura

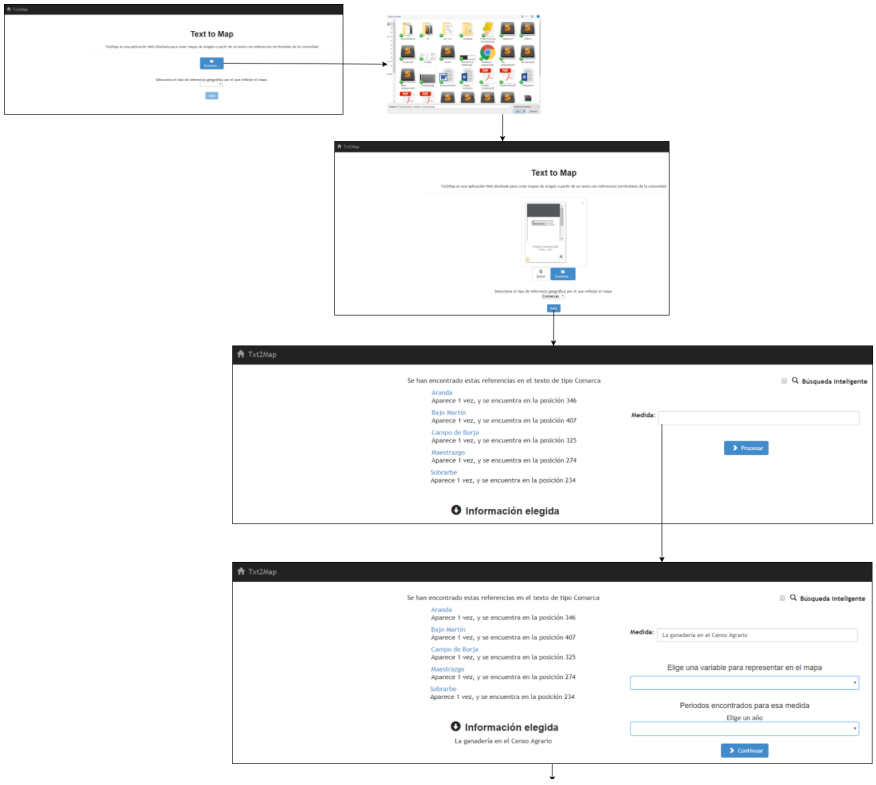

Figura 3: Mapa de navegación por parte del usuario de la primera parte del proceso de Text2Map para generar el mapa

8 donde se puede observar la arquitectura interna planteada e implementada para el backend y el frontend del sistema desarrollado.

4.3. Implementación

La parte más significativa del proyecto ha sido su implementación, la cual ha ocupado la mayor parte del tiempo. Después de efectuar todo el análisis y diseño previo de la aplicación, considerando los aspectos fundamentales, se dio paso al desarrollo técnico. Para más detalles sobre los diagramas de secuencia que reflejan los principales procesos del sistema, ver anexo Diagramas y esquemas complementarios.

4.3.1. Identificación de referencias territoriales del documento

La versión utilizada para este proyecto ha sido la más reciente en ese momento[1], Apache Solr [10] 6.5.1. Apache Solr es un motor de búsqueda de código abierto que está basado en Apache Lucene [11] para recuperar información que interesa al usuario de un texto. La característica de ser código abierto posibilita el acceso público y posibles mejoras posteriores. Es muy útil para cualquier aplicación que requiera de un proceso de indexación y búsqueda en textos, por lo tanto, es adecuado para desarrollar el

[1] Abril de 2017

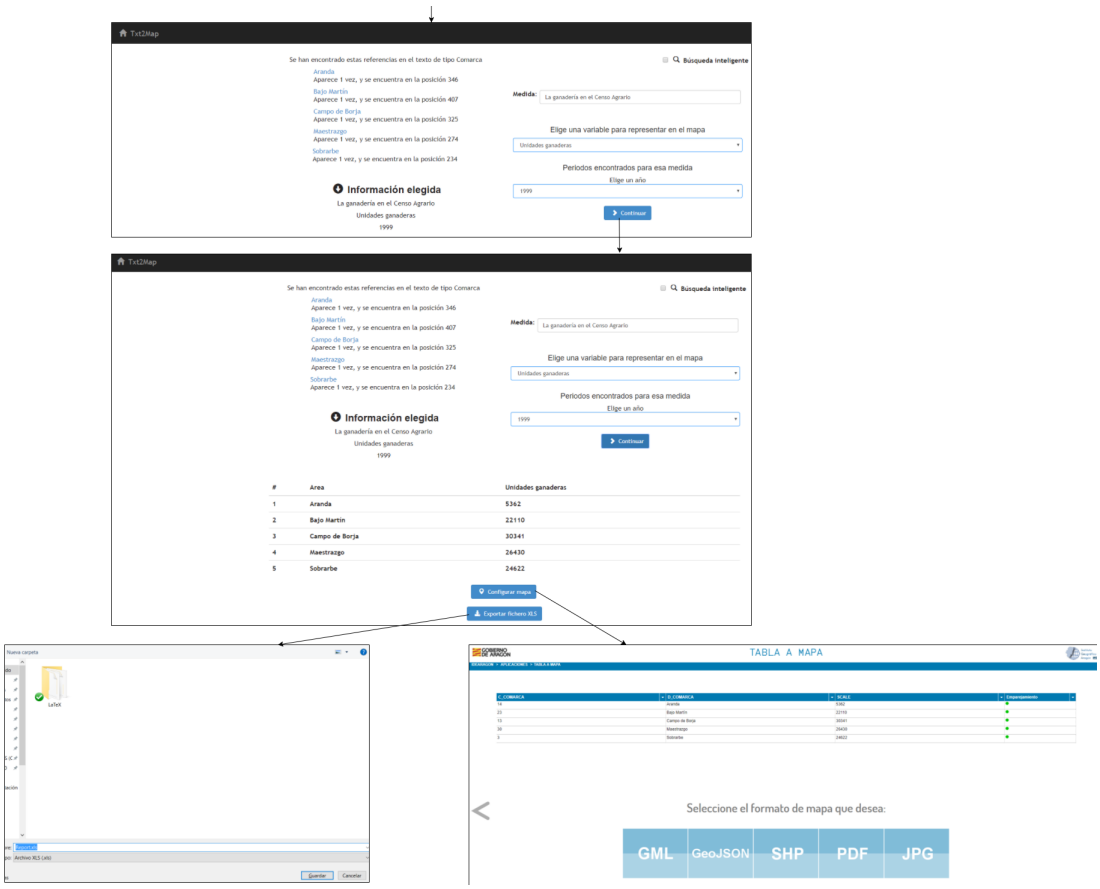

Figura 4: Mapa de navegación por parte del usuario de la segunda parte del proceso de Text2Map para generar el mapa

primer componente de este proyecto, en el cual se introduce un fichero y se deben buscar en él palabras ya conocidas previamente-áreas geográficas de Aragón en este caso.

Esta plataforma tiene diferentes formas de ser utilizada, y en este proyecto se ha hecho uso de dos de ellas:

1. A través del cliente Java *Solrj*
2. Mediante comandos *curl*

SolrJ se ha manejado desde el servidor web desarrollado en Java, proporcionando un cliente Java que puede realizar las operaciones de añadido, indexación, y consulta de documentos. Por otro lado, el comando curl ha sido empleado en la elaboración de los *scripts* para realizar las mismas operaciones comentadas. Para Solr, cada documento está compuesto por *campos*, que son piezas más específicas de información, y que pueden contener diferentes tipos de dato (por ejemplo, el campo nombre puede ser un texto, pero el campo número de pie puede ser un número). Estas clases de dato se le especifican a través de sus *tipos de dato*, para indicar cómo interpretar cada campo y cómo ser consultado.

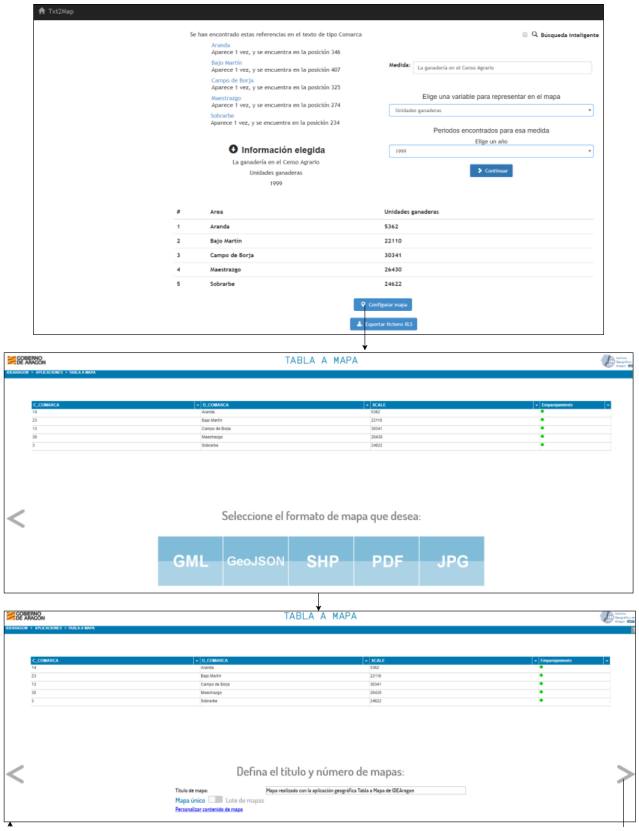

Figura 5: Mapa de navegación por parte del usuario de la primera parte del proceso de Tbl2Map para generar el mapa

Solr permite construir un índice con éstos diferentes campos durante el indexado, para posteriormente poder realizar las búsquedas sobre palabras rápidamente. De esta manera, cuando se añade un documento a Solr, éste extrae la información de los campos del documento y la añade a su índice. Cuando se le realiza una consulta, podrá buscar directamente en el índice y devolver los documentos coincidentes con la petición. La ventaja que proporciona Solr es que es una herramienta muy flexible, teniendo una cantidad muy extensa de parámetros con los que trabajar para alcanzar el objetivo final.

En el proceso de la indexación, se indica a Solr cómo y qué palabras se quiere almacenar en su índice, puesto que puede haber una gran cantidad de palabras de un documento que no interesa guardar. De forma análoga, en el proceso de consulta, también se indica cómo se pretende realizar la búsqueda, ya que se pude realizar un proceso de tratamiento de la búsqueda previo a ejecutar la consulta. Toda esta configuración se determina en el fichero *schema.xml* [12].

En el caso concreto de Text2Map, se copia todo el contenido del documento añadido en dos campos creados, uno para comarcas y otro para municipios. Al final del proceso, en cada campo quedarán las comarcas y los municipios, respectivamente, que se han encontrado en el documento, y cada área apuntará al documento en el que se ha identificado. En ambos campos, en el proceso de indexación, se realiza un tratamiento de los datos que consiste en:

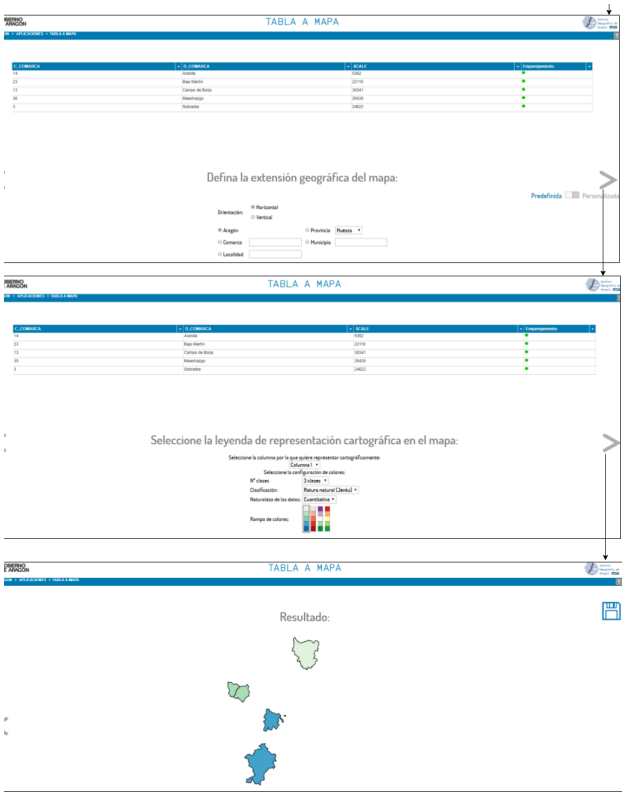

Figura 6: Mapa de navegación por parte del usuario de la segunda parte del proceso de Tbl2Map para generar el mapa

1. Dividir el contenido del documento por espacios en blanco, y asociar cada palabra a un *token*.

2. Combinar cada *token* con los siguientes n *tokens* para formar un nuevo *token* de palabras compuestas (muchos nombres de áreas son compuestos)

3. Filtrar esas combinaciones resultantes del punto anterior con la lista de los nombres de las áreas para que sólo queden áreas en el índice. Aquí, depende del tipo de índice que sea, se utilizará la lista de comarcas o de municipios.

Una vez se ha realizado el proceso de indexación, se pueden realizar consultas a ese índice. El proceso de consulta consiste en determinar si cada área a tratar aparece o no en el índice. Si el resultado devuelto por Solr indica que ese área está en el documento a tratar, entonces este área se debe considerar en la consulta SPARQL. Esta primera parte está reflejada visualmente en la figura 9 y en el diagrama de secuencia de la figura 10. Para más detalles consultar el anexo A.

4.3.2. Obtención de información estadística

La información que se puede representar como medida estadística en el mapa temático viene dada por la API proporcionada por el portal Open Data del Gobierno de Aragón [4]. Es por ello que no se almacena información en la propia aplicación en forma de base de datos, sino que se realizan consultas *online* para evitar trabajar con

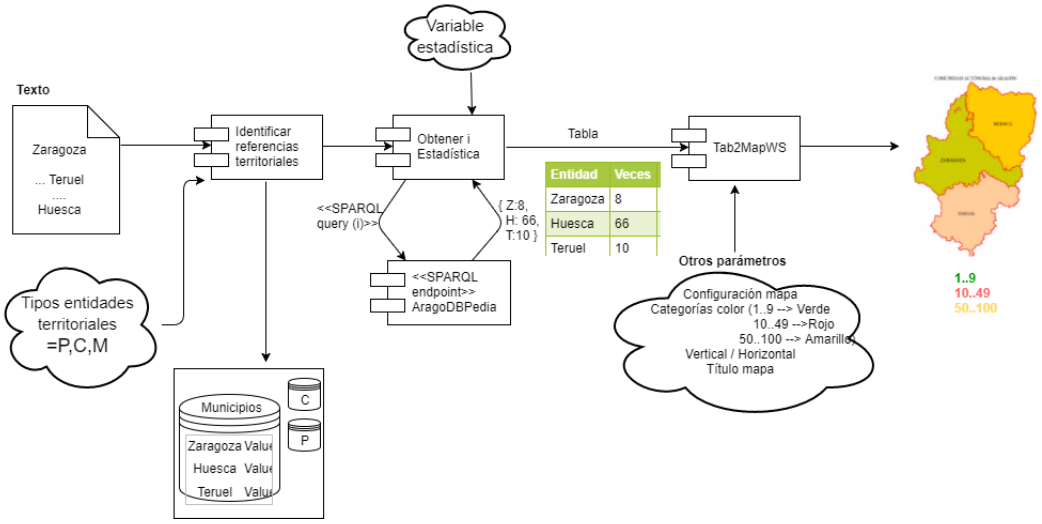

Figura 7: Visión global de los componentes de la aplicación desarrollada

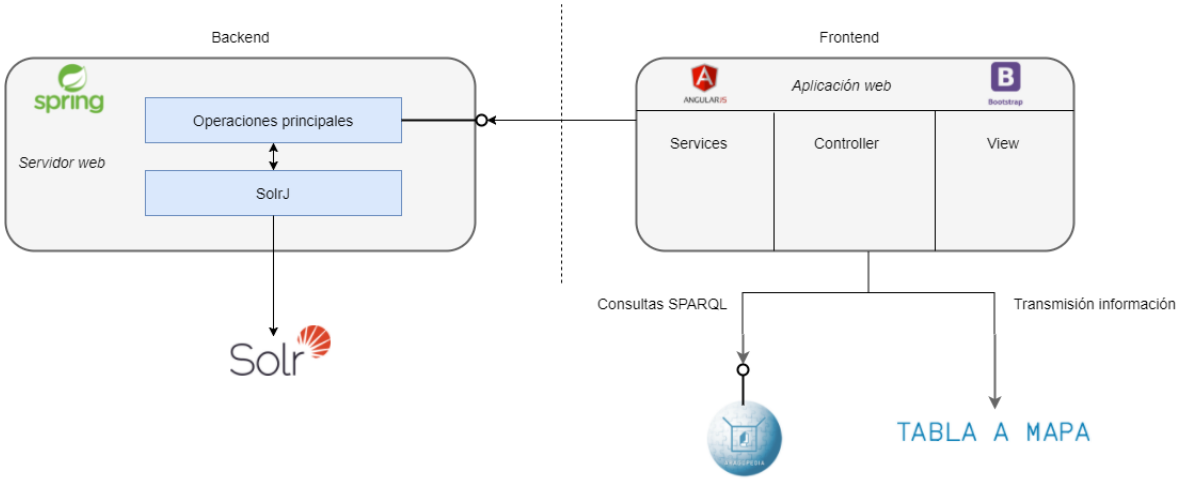

Figura 8: Arquitectura interna de la aplicación web

información obsoleta. Además, se considera que los datos proporcionados por la fuente consultada son fiables y actuales, revisados periódicamente. Así mismo, de esta manera no se tiene que hacer un control de la mantenibilidad y actualización de la información dentro de la aplicación, simplificando su arquitectura y mantenimiento.

Aragón Open Data es una iniciativa que permite obtener datos concretos de la medida que se desea mostrar, aportando, además de la medida de información a representar, las áreas geográficas y el período de tiempo del que se quiere encontrar la información. Para ello, el componente de la aplicación desarrollada ejecuta un proceso compuesto por varias peticiones con la finalidad de componer una consulta final a la API de SPARQL que provee Aragopedia [13]. En este proceso, los resultados de las peticiones se van mostrando al usuario para que él decida todos los elementos necesarios para formar la consulta (excepto las áreas geográficas, puesto que éstas ya se han obtenido previamente del documento de texto tratado).

El usuario debe indicar tanto el período de tiempo que desea de información, como el

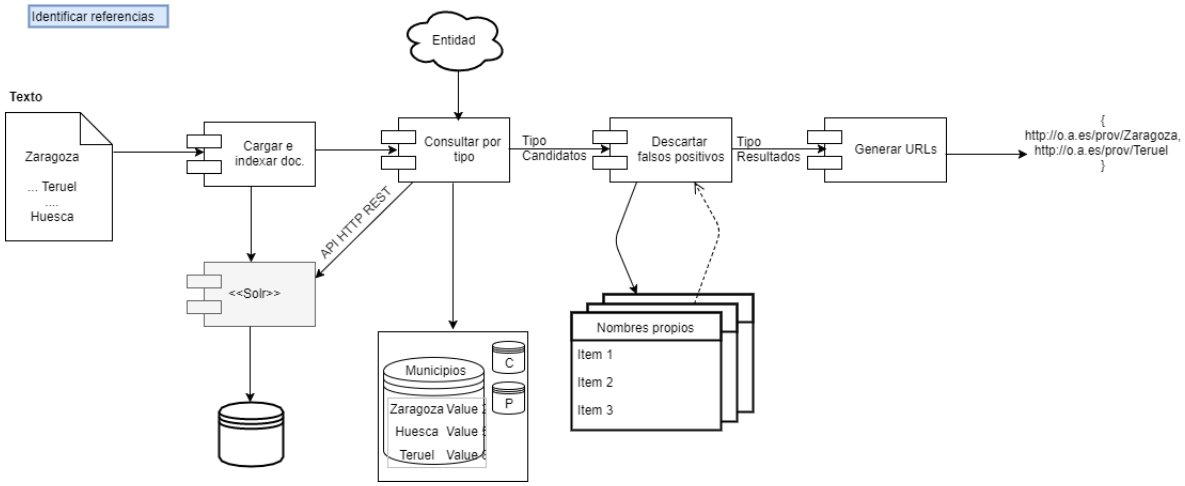

Figura 9: Diagrama técnico del proceso Solr

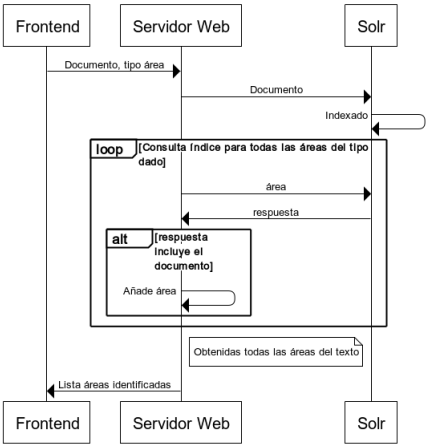

Figura 10: Proceso identificación de áreas

tipo de informe deseado, y la variable de ese informe que quiere representar finalmente en el mapa. Es posible que tenga que decidir también un valor para otras variables de tipos enumerados de ese informe. Para más información, consultar anexo A, donde se explican los cubos de dimensiones que se consultan.

En concreto, el proceso se compone de los siguientes pasos:

1. Realizar una primera petición a Aragopedia para recoger todos los informes (temas de información) existentes en su base de datos, por ejemplo, “Paro registrado”, “Censo demográfico” o “Parcelas existentes”.

2. Mostrar los posibles informes al usuario para que seleccione uno de ellos a través de una búsqueda simple donde aparecen todos.

3. Una vez elegido el informe, enviar una petición para recoger los períodos de tiempo disponibles para este informe, y las variables/atributos que lo componen; por ejemplo, para el informe “Ganadería en el censo agrario”, existen variables como “Ganado ovino”, “Ganado bovinoz “Ganado porcino”.

4. El usuario debe elegir un año, y una variable numérica de las obtenidas en el punto 1 anterior para representarla en el mapa posteriormente.

5. Además, es necesario tener en cuenta que, si el informe contiene variables no numéricas, el usuario debe fijar un valor para cada una de ellas.

6. Construir la consulta necesaria para obtener los valores correspondientes a cada área, el año propuesto de la variable elegida, y enviarla al endpoint SPARQL.

7. Construir una tabla con los datos obtenidos para mostrársela al usuario

8. Definir el estilo o configuración del mapa (colores, orientación, etc.)

El diagrama de secuencia perteneciente a este proceso se muestra en la figura 11. Cabe decir que el proceso técnico llevado a cabo en esta parte está contemplado con mayor detalle en el anexo B.

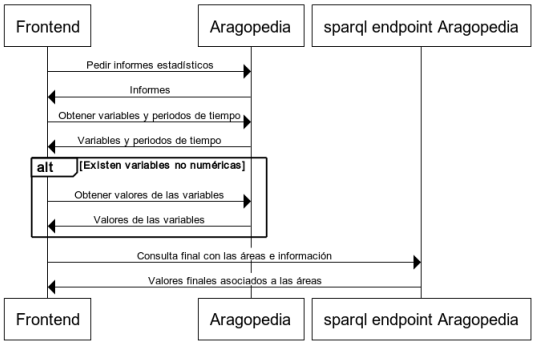

Figura 11: Diagrama de secuencia del proceso Solr

Como se ha comentado anteriormente, esta parte depende absolutamente de la información almacenada en Aragopedia. Esto quiere decir que es posible que el usuario seleccione un informe en el que no existen datos en ese momento, o que ninguna de las variables devuelva resultados. Además, Aragopedia es una iniciativa que sufre cambios habitualmente, y que deja inactivo su acceso a la información en ocasiones para producir mejoras en su servicio o actualizaciones de datos. En cualquier caso, si se dan estos casos, se avisa mediante un mensaje de alerta al usuario para que tenga consciencia de ello y pueda cambiar la información de búsqueda.

4.3.3. Generación de mapa temático

Tbl2Map es una aplicación web propia de la empresa Idearium Consultores que genera mapas temáticos de Aragón a partir de los datos proporcionados en una tabla. Por ello, esta aplicación se emplea en la construcción del tercer componente del sistema desarrollado.

La aplicación Tbl2Map original recibe un fichero que puede ser de los siguientes formatos estructurados: CSV, XML, XLS, GPX o JSON. Este fichero debe contener el código indicador del área, el nombre del área y un valor asociado, que es el que se

representa en el mapa. Como áreas posibles, se pueden representar comarcas, localidades, municipios, calles, cuadrículas UTM, parcelas, topónimos y coordenadas. Abarca un gran abanico de información, pero lo más relevante en este caso es que acepta las dos categorías de áreas admitidas en la aplicación del proyecto.

Para realizar el vínculo entre Tbl2Map y nuestra aplicación (Text2map), fue necesario realizar varias modificaciones en el código de Tbl2Map. Concretamente, se tuvo que ampliar Tbl2Map para permitir recibir información en una petición HTTP POST sin interactuar con un usuario final. Para ello, se creó una página JSP para obtener desde ahí los parámetros emitidos desde la aplicación Txt2Map y hacer con ellos los cambios necesarios que se comentan a continuación:

- No tener que introducir un fichero estructurado con la información, sino que la obtiene de los parámetros recibidos.
- Realizar el emparejamiento de las áreas encontradas con la base de datos de la empresa para comprobar que son correctas.
- Pasar de estado/página inicial a mostrar para exponer directamente la página de elección del formato de salida de la imagen al usuario.

Una vez terminadas estas modificaciones en Tbl2Map, se ha empaquetado la aplicación y desplegado en el servidor web Tomcat para poder probar su correcto funcionamiento. Cabe decir que también se ha dejado la opción inicial, en la cual la aplicación Tbl2Map no recibe ningún parámetro, sino que es un proceso entero en el que se pide introducir un documento estructurado y tratar la información contenida en él.

4.3.4. *Script* de lotes de mapas

Como resultado complementario al desarrollo de la aplicación, se tenía como objetivo proporcionar también un acceso al servicio web de Tbl2Map para poder obtener un lote de mapas a la vez, sin intervención del usuario, dado un directorio donde hay varios documentos; es decir, cada documento debe generar un mapa correspondiente a las áreas encontradas en ese documento según el tipo de área indicado.

Para lograr este objetivo, se ha decidido crear un script de *Bash Shell*, por su sencillez y por ser el medio preferido en la empresa para llevar a cabo esta tarea. En este script, se ha tratado de repetir el proceso simplificado que se ha realizado en la aplicación web; es decir, para cada documento se realiza el proceso entero, generando su mapa correspondiente con las áreas encontradas en él, antes de pasar al siguiente documento. A continuación se van a detallar los pasos seguidos para desarrollar este script:

1. Cargar e indexar los documentos existentes en un directorio. La ruta de éste es especificada por el usuario como primer parámetro del script, y el tipo de área es indicado también como segundo parámetro (en este caso, comarca o municipio). Este paso se ejecuta mediante el comando *curl*, haciendo una petición a la plataforma Solr en la que ya está configurado el schema para crear el índice tal

y como se desea. Para obtener una información detallada de este proceso ver la sección 4.3.1.

2. Realizar un conjunto de peticiones a través del comando *wget* con las diferentes áreas existentes en Aragón del tipo indicado (es decir, todas las comarcas o todos municipios de Aragón), para saber cuáles pertenecen al índice recién creado. Si el resultado de la consulta incluye el documento que recién se ha subido, el área consultada se almacena en un array, y se procede con la siguiente área, hasta que se procesan todas las áreas. De esta manera se consigue la lista de áreas encontradas en el documento actual. Por el contrario, si la respuesta de la consulta no devuelve el documento, el área no se encuentra en el índice, por lo que se elimina de la consulta SPARQL que se realizará cuando este proceso termine.

3. Construir la consulta SPARQL. La ruta del fichero que contiene la *consulta SPARQL* a realizar debe especificarse antes de comenzar este proceso de reconocimiento de áreas, para poder transmitirla a una variable y tratarla más fácilmente. En esta consulta se determina la medida estadística que dará como resultado un valor concreto para cada área. Es por ello que una vez termina este procedimiento se consigue, por un lado, la consulta SPARQL con las áreas reconocidas en el texto, preparada para ser enviada al endpoint SPARQL de *OpenData* [13], y por otro lado, las áreas reconocidas en un *array*.

4. Enviar la consulta construida al endpoint indicado previamente. Al recibir la respuesta de esta llamada, ya se tiene el valor esperado que corresponde a cada área indicada en la consulta, entre otros datos que no tienen relevancia. Para conseguir extraer solamente la información necesaria de la respuesta a la petición SPARQL, se ha creado un breve programa Java en el que se utiliza el parser SAX para obtener información concreta de documentos XML. Este parser obtiene el nombre de las áreas y el valor asociado a éstas, y lo devuelve de vuelta al script, obteniendo así la información necesaria para transmitirle al servicio web de generación de mapa.

5. Configurar el mapa. Se desarrolla el procedimiento encargado de colocar cada área identificada en su color correspondiente según el valor recogido de ésta a través de la consulta l. Para ello, se ha utilizado como modelo de clasificación el método de agrupamiento *Jenks* [14], por ser el más justo con las distancias entre valores y el preferido por la empresa. Como características fijas se han elegido 10 colores diferentes de la gama roja. Para lograr este proceso, primero hay que definir los distintos intervalos de valores en los que se dividen los colores. Para ello, se ha desarrollado un conciso programa en Python que lo resuelve.

6. Envío del contenido XML al servicio web de Tbl2Map [15] para la generación del mapa. Para llevar a cabo este cometido se ha realizado un programa de Java, al que se le transmite una lista de colores con las áreas correspondientes a cada color. En él, simplemente se construye el XML que necesita posteriormente el servicio web con la lista recibida, y devuelve el contenido generado (el mapa en formato *base64*) al script.

7. Conversión del mapa al formato png. El script recoge la respuesta del servicio web de Tbl2Map a través de un programa Java y lo convierte en un fichero con

extensión .png para poder visualizarlo, llamado "mapadocX", siendo X el número de documento procesado. En este mapa se representan las áreas encontradas en el texto con los colores correspondientes según el valor de la medida indicada en la query l.

Todo este proceso se repite para cada uno de los documentos que existen en el directorio indicado por el usuario, generando así tantos mapas como documentos presentes hay en él.

4.3.5. *Script* de mapa de popularidad

Además del script en el cual se genera un mapa por cada documento indicado, anteriormente comentado, se decidió realizar otro con una misión diferente, pero con un proceso similar. Este script se encarga de generar un solo mapa que recoge todas las áreas de los diferentes documentos, y en lugar de utilizar una consulta SPARQL para recoger valores que representar en el mapa como el otro script, se utilizaría como valor el número de documentos en los que aparece cada territorio. De esta manera, se consigue ilustrar la popularidad de cada lugar en un conjunto de documentos. Por otro lado, se crea también un pequeño informe en el que se representa cada documento con las áreas encontradas en él.

Este script recibe como primer parámetro la ruta del directorio en el que se presentan los distintos documentos a analizar, y como segundo parámetro el tipo de área a considerar (comarca o municipio), y realiza los siguientes pasos:

1. En primer lugar, se cargan e indexan en Solr todos los documentos del directorio mediante un bucle compuesto de peticiones *curl*. De esta forma, se crea el índice con todas las áreas encontradas en los documentos, divididas en el campo dedicado a las comarcas y el campo dedicado a los municipios. Cada área apunta al identificador del documento/s en el/los que aparece.

2. A continuación, se extran las palabras existentes en el índice para saber los territorios encontrados. En esta opción, debido a que hay varios documentos presentes en Solr, al realizar una consulta con un área la respuesta indica los identificadores de los documentos en los que se ha identificado dicha área, entre otros apuntes. Por lo tanto, se ha recogido para cada lugar el número de documentos en los que aparece, si aparece en alguno. Cuando acaba este procedimiento, se transmite a un fichero la información obtenida con la siguiente estructura:

```
Doc: X
Areas: aaa bbb ccc...
```

3. Por último, se realiza la generación de un único mapa de forma análoga al script anteriormente descrito. La única diferencia encontrada con el anterior script es que, en este caso, la lista de valores que recibe el programa Python se basa en el número de documentos en los que se encuentra cada lugar, en lugar de adoptar estos valores de una consulta l.

Este script se ha desarrollado de una forma que sea generalizable para poder añadir/quitar elementos de forma sencilla y permitir reutilizar código (programas Java y Python)

4.4. Pruebas

En el desarrollo de este proyecto se han realizado fundamentalmente tres tipos de pruebas:

1. Pruebas unitarias de cada uno de los componentes.
2. Pruebas de integración de los diferentes componentes.
3. Pruebas globales del sistema desarrollado.

Las pruebas unitarias y de integración se han ido ejecutando desde el inicio de la aplicación cada vez que se implementaba o depuraba un nuevo componente, analizándolas de forma exhaustiva. Por otro lado, las pruebas globales realizadas una vez se hubo desarrollado la primera versión del proyecto tuvieron como objetivo depurar los errores cometidos en el desarrollo que no fueron detectados previamente, analizar la experiencia de los usuarios, medir el rendimiento y la carga que puede soportar el sistema, etc.

En concreto, se han realizado pruebas de funcionalidad, usabilidad, compatibilidad, rendimiento y seguridad [16] que se describen a continuación.

- **Funcionalidad.** Durante el proceso de desarrollo de la aplicación web, se ejecutaban pruebas de la identificación de áreas mediante breves documentos creados por la estudiante, en los cuales se incluían referencias territoriales, entre otras informaciones, para comprobar su determinación. Una vez finalizada la aplicación, se elaboró una prueba a través de la cual se ha hecho uso de documentos oficiales del BOA[2]. En éstos, aparece una gran cantidad de áreas para poder probar tanto la identificación correcta de todas ellas, como el rendimiento con documentos más extensos que los probados hasta el momento. Gracias a esta prueba, se ha consiguieron depurar errores hasta entonces no detectados. También se analizaron todos los enlaces de la aplicación para comprobar que funcionan correctamente, las conexiones correspondientes a otras APIs, y que los formularios realizan su tarea adecuadamente, así como la muestra de errores con información conveniente.

- **Usabilidad.** La importancia de la experiencia del usuario al utilizar un proyecto web como el que en este caso nos ocupa es tan importante como su entero desarrollo, ya que está pensado para ser utilizado por todos los públicos. Considerar el comportamiento del usuario frente a la aplicación web nos permite tener en cuenta su opinión y mejorar el sistema.

[2]Boletín Oficial de Aragón

Por esta razón, se realizaron test de usabilidad con 3 usuarios a los que se les solicitó que realizasen varias tareas unitarias con el fin de crear el mapa correspondiente. En este test se ha pretendido medir tanto las dificultades o confusiones que encuentran, como el tiempo de respuesta de éstos. Para obtener unas respuestas distintas, se ha elegido un grupo heterogéneo de usuarios, entre los que se encuentran personas con poca experiencia tratando aplicaciones web, y personas con bastante habilidad para ello.

La respuesta de los dos usuarios inexpertos o no acostumbrados a este tipo de aplicaciones web ha sido similar. La aplicación les resulta sencilla en cuanto a interfaz, con elementos limitados en las pantallas de la aplicación. Uno de los cambios propuestos por un usuario ha sido modificar tanto los nombres que aparecen en algunos botones como la palabra que acompaña a la entrada de texto del informe estadístico (llamada "medida") para que se entienda mejor.

Otro de los cambios sugeridos ha sido eliminar el subrayado de las áreas identificadas en el texto y cambiarle el color azul que tiene, porque se confunde con un hipervínculo. Por último, se ha recogido la opinión de que era más razonable acceder directamente a la búsqueda en la que aparece el desplegable de todos los datos posibles en lugar de tener que escribir para que aparezcan referencias, ya que no se conoce qué tipo de información está almacenada.

Por otro lado, se ha realizado el mismo test a un usuario acostumbrado a tratar con aplicaciones web modernas, y el resultado ha sido diferente. No ha puesto aspectos en duda, y ha alcanzado el objetivo pedido con mayor rapidez.

Estas pruebas con usuarios han sido de gran ayuda para poder realizar sus cambios propuestos y contribuir a que todos los públicos puedan participar en el uso de la aplicación.

- **Compatibilidad.** La compatibilidad de un sitio web es un aspecto fundamental en la fase de testeo. Hoy en día, se utiliza toda clase de dispositivos tecnológicos para navegar por la web, y no sólo ordenadores como antiguamente. Es por ello que se ha tratado de desarrollar una aplicación *responsive* en todo momento, es decir, que se adapte al dispositivo en el que se está mostrando. Aunque es una aplicación pensada para emplearla a través del ordenador por la necesidad de subir ficheros (más habituales en este tipo de dispositivos), se ha comprobado que también se adecua a smartphones. Para ver los resultados de las diferentes pruebas en dispositivo móviles, ver figuras 19, 20, 21 y 22 en anexo F.

  Tan importante como probar la aplicación en otros dispositivos (o más, en este caso) es probarla en distintos *navegadores*, ya que algunas aplicaciones son muy dependientes de los navegadores, por sus diferentes configuraciones y herramientas. Esta aplicación ha sido testeada en Google Chrome, Firefox, Microsoft Edge, Safari y Opera, que actualmente son los más utilizados en todo el mundo. El funcionamiento va correctamente tanto en Chrome como en Firefox. Esta aplicación tiene toda la interfaz del cliente basada en Bootstrap 3.3.6, y Angular 1.4.8, las cuales son tecnologías más o menos recientes, por lo que es posible que algunas funcionalidades no funcionen en navegadores que no las soportan.

- **Rendimiento.** Las aplicaciones web deben soportar, en ocasiones, una gran carga en ellas. Para testear esta tarea, muchos usuarios tienen que acceder a una

misma página y comprobar que el sistema responde con normalidad en los picos de carga. Para realizar este proceso, se han utilizado los scripts elaborados, y se han lanzado varias veces simulando ser varios usuarios para comprobar tiempos de respuesta y comprobar cuándo se produce algún error por recibir demasiada carga. Se han realizado pruebas con 10, 20, 30 y 40 usuarios simultáneos para los dos scripts desarrollados, cuyos resultados se muestran en la tabla 2. Con 40 usuarios simultáneos, el script de lote de mapas se precipita en la creación de los diferentes png.

Usuarios	Script lotes				Script popularidad			
	10	20	30	40	10	20	30	40
$Tiempo(sg)$	19	33	51	67(falla)	14	24	37	52

Tabla 2: Rendimiento según número de usuarios y tipo de *script*

- **Seguridad.** La seguridad proporcionada por una aplicación web reside en haber comprobado todas las entradas y salidas posibles en cada proceso existente para evitar situaciones de riesgo que no han aparecido anteriormente. Uno de los aspectos más primordiales hoy en día en las aplicaciones web trata poder visualizar ciertas páginas de la aplicación sólamente si el usuario ha iniciado sesión en la aplicación. En este caso, no existe la posibilidad de login, por lo tanto, se elimina gran cantidad de comprobaciones. Para testear la seguridad, se ha comprobado en el cliente que no deje enviar los formularios si no se han introducido todos los parámetros que requiere, pero también se comprueba en el servidor si la información es apropiada (puede ser que se realice la petición desde otra herramienta, sin pasar por la interfaz). Además, se han verificado que los mensajes de error sean adecuados a cada situación para dar información al usuario de la situación.

4.5. Plan de mantenimiento

El desarrollo de una aplicación web no finaliza cuando todo funciona, sino que se requiere realizar un mantenimiento del proyecto una vez que se haya implantado. Es de gran relevancia realizar, mínimo, un *mantenimiento correctivo*, el cual consiste en corregir errores que surjan, aunque también es aconsejable realizar un *mantenimiento evolutivo*, encargado de progresar en la aplicación desarrollando nuevas funcionalidades.

Por desgracia, es habitual que la entrega de un proyecto web requiera un período de correcciones de bugs que pueden resultar pequeñas anomalías constatadas durante la fase de pruebas a las cuales no se le dieron importancia, o pueden ser nuevas situaciones que no habían aparecido anteriormente. Cabe destacar que los fallos encontrados por los usuarios deben avisarse al creador de la aplicación para tener constancia de ello y poder dar solución al problema, además de aportar apoyo en forma de información a los usuarios.

Asimismo, en relación al aspecto de seguridad hay que garantizar la prevención de bugs que puedan afectar gravemente al flujo habitual de la aplicación. El problema

principal en el desarrollo de aplicaciones web es la falta de seguimiento en los hilos de los que depende cualquier aplicación: las entradas y salidas del sistema. Es imprescindible considerar la exposición accidental de datos que puedan servir como ataque al sistema. Por ejemplo, es recomendable actuar sobre los mensajes de error enviados por el servidor, ya sea deshabilitándolos o modificándolos (por ejemplo, cuando el servidor no encuentra algún archivo pedido en particular), ya que da información al atacante sobre el sistema.

Cabe destacar también la importancia de la realización de *backups* constantes para no perder ningún dato que tenga relevancia en el sistema. Por ello, es esencial ser selectivo con la información a guardar, y sobre todo poder hacer el backup de forma automática para evitar perder tiempo y cometer errores manuales.

Por último, cabe indicar que si se recogen nuevas ideas que puedan ser desarrolladas para implementar nuevas funcionalidades, se deberían recoger los requisitos que requieren estas tareas para realizar un análisis y saber cómo tratarlas. En definitiva, seguir el proceso indicado en la metodología descrita en el capítulo 3.1. Además, se deberá comunicar al responsable de la aplicación de estas modificaciones para que tenga constancia y pueda participar en la renovación.

5. Conclusiones y resultados

En este capítulo se indican las principales conclusiones del desarrollo de este proyecto, atendiendo a diferentes aspectos como el cumplimiento de los objetivos, las competencias adquiridas y lecciones aprendidas, y posibles líneas de trabajo futuro. Además, se incluye una valoración personal de la estudiante.

5.1. Cumplimiento objetivos

Los objetivos del proyecto se definieron en la fase inicial de análisis, tanto los principales como los secundarios, tal y como se ha comentado en su sección 1.3. Durante el desarrollo del proyecto, se ha dado prioridad a los requisitos principales en cada una de las tres partes o componentes en los que se ha dividido el proyecto. La estudiante se muestra muy satisfecha con el trabajo realizado, porque considera que se han cumplido todos los requisitos establecidos y el objetivo principal. Además, se siente complacida con lo aprendido en el ámbito académico.

En la primera parte del proyecto, el objetivo principal era identificar las referencias territoriales en el documento, para lo cual se ha requerido hacer uso de la plataforma Apache Solr, y recoger el conjunto completo de comarcas y municipios. El desarrollo de este primer componente de la solución ha costado 75 horas-persona aproximadamente, y ha sido el corazón del proyecto.

En cuanto al segundo componente de la solución desarrollada, el objetivo principal era conseguir la información del portal de datos abiertos de Aragón (más concretamente, de Aragopedia) y mostrarla al usuario como diferentes opciones. Como requisito, se indicó realizarlo mediante consultas SPARQL dirigidas a su endpoint SPARQL. El desarrollo de este componente ha tenido un coste de 40 horas-persona aproximadamente. La mayor parte de las horas se han invertido en mostrar al usuario la información adecuadamente.

Por lo que se refiere a la tercera componente de la solución desarrollada, el objetivo era claro: poder transmitir las áreas identificadas junto a sus valores correspondientes a la aplicación Tbl2Map. No se impusieron requisitos al respecto, pero sí se aconsejó que el paso de los parámetros se ejecutara a través del método POST. Por ello, se tuvieron que realizar varios cambios en Tbl2Map para poder recibir estos datos. Esta parte costó finalizarla 40 horas-persona aproximadamente.

Con respecto a los scripts desarrollados para permitir procesos *batch*, el objetivo inicial propuesto era crear una manera de acceder de forma automática al servicio web de Tbl2Map y obtener varios mapas correspondientes a sus documentos, sin tener que hacer todo el proceso de la aplicación. Aunque en esta tarea tampoco hubo requisitos estrictos, se optó por un desarrollo a través de un script de Shell por su comodidad y fácil automatización. Por último, la idea del segundo script se originó al finalizar el primer script, puesto que se consideró que sería de utilidad en las tareas de la empresa. Su objetivo principal es generar un mapa, también a través del acceso automático a Tbl2Map, consumiendo varios documentos en lugar de solo uno, y visualizar el número

de documentos en los que aparece cada área encontrada en ellos. El coste de realizar ambos scripts ha sido de 50-persona horas aproximadamente.

5.2. Coste del proyecto

En este capítulo se expone el coste total que ha tenido el desarrollo de este trabajo final de grado, así como el coste desgranado en fases (ver figura 12), acompañado de diagramas de Gantt (ver apéndice E donde se aprecia esta información de forma visual).

- **Identificación de áreas en documentos a través de Solr.** 75 horas aproximadamente.
- **Extracción de información estadística a través de Open Data Aragón.** 40 horas aproximadamente.
- **Transmisión de información a Tbl2Map.** 40 horas aproximadamente.
- **Script de lotes de mapas.** 30 horas aproximadamente.
- **Script de popularidad de áreas.** 20 horas aproximadamente.
- **Elaboración y revisión de la memoria.** 70 horas aproximadamente.
- **Pruebas.** 20 horas aproximadamente.
- **Corrección de errores.** 30 horas aproximadamente.
- **Reuniones con los superiores.** 15 horas aproximadamente.

Una vez dividido por fases el coste del proyecto, se procede a realizar la suma de los costes, y se conoce que el coste total es 340 horas.

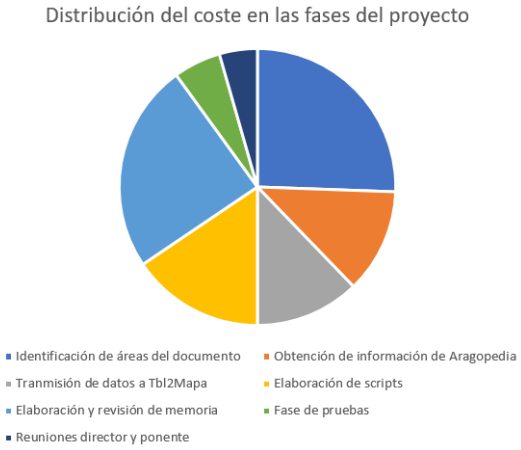

Figura 12: Gráfico circular que representa el coste de cada fase

5.3. Posibles líneas de trabajo futuro

El propósito de esta sección consiste en indicar posibles líneas de trabajo futuro o extensiones de este proyecto partiendo de él como base. Antes de comenzar a enumerar dichas líneas, cabe comentar que son intenciones o planes que puede necesitar la empresa en un determinado momento en el futuro, y que no se han abordado porque no se indicaron como objetivos para este trabajo.

En esta aplicación se ha obtenido el número de apariciones de cada área en el texto gracias a una de las operaciones proporcionadas por Solr, pero no se ha utilizado para actividades posteriores, simplemente se muestra como información. Una posible mejora del sistema consistiría en mostrar en el mapa generado fragmentos del texto del documento donde se hace referencia a dicha área geográfica, por ejemplo, destacando de alguna forma un mayor número de apariciones de las referencias territoriales en el documento. De esta manera, se podría saber a simple vista de qué áreas se está hablando más en el texto, es decir, las que tienen más relevancia.

Text2Map recoge todas las áreas de un tipo (comarca o municipio) que aparecen sobre un documento de texto, pero podría interesar también conseguir las áreas que no se manifiestan en el documento. Es decir, calcular la tarea inversa a lo que se realiza actualmente.

Actualmente, en esta aplicación se ha utilizado el sitio web de aragon.opendata.es para obtener, con su endpoint SPARQL, toda la información referente a territorios que se muestra al usuario como opción para representar en el mapa. Otra posibilidad podría ser dar opción a elegir otros enlaces del endpoint SPARQL, como la *dbpedia*, para recoger otros tipos de información y tener datos más variados.

Otro de los posibles trabajos pendientes es la implementación de un sistema seguro a través de HTTPS en lugar de utilizar HTTP, para tener una comunicación entre servidor web y navegador cifrada. Para más detalles, ver anexo Implantación de seguridad mediante HTTPS.

Por último, cabe indicar que se ha realizado un desarrollo escalable, de modo que es sencillo integrar la aplicación existente con otras aplicaciones, tal y como se ha hecho con Tbl2Map. En un futuro, el sistema desarrollado podría ser integrado con otras aplicaciones de la empresa, incluso con un gestor de contenidos (CMS) para vincular, por ejemplo, noticias.

5.4. Valoración personal

Este proyecto no sólo me ha servido para poner en práctica conocimientos adquiridos durante el grado, sino que también me ha hecho desarrollarme como ingeniera, y saber buscar las mejores soluciones a los problemas que van surgiendo sin perder el control. Gracias a ir cumpliendo con los objetivos propuestos y avanzar en cada fase no he perdido la esperanza de poder terminarlo en mi fecha de meta. Aprecio muchísimo todo lo que he aprendido, tanto lo relativo a herramientas concretas que desconocía por completo, como las mejoras que he adquirido con la base que ya tenía en otros aspectos.

De los distintos problemas que me han surgido, el obstáculo que más peso ha tenido para mí ha sido la imposibilidad de avanzar en ocasiones por cambios realizados en el sistema de Aragopedia. Se han producido a lo largo del proyecto, y al depender una parte de este sistema, hubo que adoptar otras medidas para poder progresar y no perder tiempo.

Valoro mucho la atención proporcionada por mi tutor y mi directora, los cuales han podido y sabido solucionarme las dudas, indicarme el camino a seguir, y darme ánimos cuando más se necesitaban. Por último, espero que las competencias adquiridas en el desarrollo de este proyecto me sirvan en un futuro y pueda continuar desarrollándolas en el ámbito profesional en diferentes empresas innovadoras dentro del ámbito TIC, como Idearium.

Como he comentado anteriormente, el aprendizaje obtenido durante todo el proyecto ha sido de enorme valor para mí. Se agradece haber tenido que lidiar con distintos temas, algunos tratados durante el grado y otros desconocidos totalmente hasta el momento. Se ha mejorado la habilidad de separar las partes relevantes de lo que se desea en una documentación extensa y nueva para el estudiante, saber hacer uso de ella y corregir errores lo más rápido posible.

La curva de aprendizaje que más trabajo ha conllevado ha sido la fase inicial, en la cual se trabaja con Recuperación de información, concretamente, Apache Solr. Esto se debe a que anteriormente en la carrera no se ha trabajado prácticamente en este ámbito, aunque al finalizar este proyecto se tienen bases suficientes de conocimiento para proseguir en este área. El contexto de los motores de búsqueda se vuelve cada día más importante en el mundo de la informática, por ejemplo, en el desarrollo de aplicaciones de inteligencia artificial, entre otras. Ha sido grato llegar a conocimientos más profundos en este campo.

En relación a la parte de consultas SPARQL, ha servido para mejorar los breves conocimientos adquiridos durante el grado, y poder analizar más esta materia de la web semántica, también muy importante actualmente en toda la web.

Para cumplir con el objetivo de generar los scripts, se ha tenido que hacer uso de la documentación existente de Bash e invertir mucho tiempo en comprender la mejor manera de desarrollar las tareas correspondientes. Es por ello que con esta fase se han aprendido muchos conceptos, y se ha conseguido elaborar una distinta forma de pensar la ejecución de tareas, muy útil seguramente en un futuro. Teniendo en cuenta que la

mayoría de empresas hacen un uso habitual de scripts Shell, es satisfactorio tener una base de conocimiento en este campo.

En definitiva, ha sido un proyecto que ha fortalecido los conocimientos que ya se tenían sobre algunos campos, y que ha hecho crear una fuerte base sobre otros desconocidos hasta el momento. De manera semejante, se ha aprendido a llevar a cabo una organización secuencial de las tareas y saber distribuir el tiempo de manera correcta para cada fase, comunicando en cada momento la situación actual del proyecto a director y ponente.

Bibliografía

[1] BOA. Boletín Oficial de Aragón. http://www.boa.aragon.es/#/, 2014. [Online; accedida 25-agosto-2017].

[2] Apache Solr. Apache Solr Official Site. http://lucene.apache.org/solr/, 2017. [Online; accedida 3-mayo-2017].

[3] W3C. Resource Description Framework (RDF). https://www.w3.org/RDF/, 2014. [Online; accedida 25-agosto-2017].

[4] Aragón Open Data. Aragón Open Data. http://opendata.aragon.es/, 2014. [Online; accedida 5-julio-2017].

[5] Universidad Rey Juan Carlos. Procesos ágiles para el desarrollo de aplicaciones Web. http://www.dlsi.ua.es/~jaime/webe/articulos/s112.pdf, 2010. [Online; accedida 10-agosto-2017].

[6] JetBrains. IntelliJ IDEA. Capable and Ergonomic IDE for JVM. https://www.jetbrains.com/idea/, 2000-2017. [Online; accedida 22-agosto-2017].

[7] GitHub. GitHub. http://www.github.com/, 2008. [Online; accedida 25-agosto-2017].

[8] SonarQube. Continuos Code Quality. https://www.sonarqube.org/, 2008-20017. [Online; accedida 16-agosto-2017].

[9] Oracle. How to Write Doc Comments for the Javadoc Tool. http://www.oracle.com/technetwork/articles/java/index-137868.html, 2010. [Online; accedida 13-agosto-2017].

[10] Apache Solr. Apache Solr Reference Guide. https://cwiki.apache.org/confluence/display/solr/Apache+Solr+Reference+Guide, 2017. [Online; accedida 3-mayo-2017].

[11] Apache Lucene. Apache Lucene Official. https://lucene.apache.org/, 2017. [Online; accedida 3-mayo-2017].

[12] Apache Solr. Documents, Fields and Schema Design. https://cwiki.apache.org/confluence/display/solr/Documents%2C+Fields%2C+and+Schema+Design, 2016. [Online; accedida 20-junio-2017].

[13] Aragón Open Data. Herramientas SPARQL. http://opendata.aragon.es/herramientas/sparql, 2014. [Online; accedida 18-julio-2017].

[14] Wikipedia. Jenks natural breaks optimization. https://en.wikipedia.org/wiki/Jenks_natural_breaks_optimization, 2017. [Online; accedida 8-agosto-2017].

[15] Idearium. Servicio Web Tbl2Map. http://idearagon.aragon.es/tab2mapWS/tab2mapWS, 2015. [Online; accedida 28-agosto-2017].

[16] Software Testing Help. Web Testing: A complete guide about testing web applications. http://www.softwaretestinghelp.com/web-application-testing/, 2017. [Online; accedida 9-agosto-2017].

[17] Apache Tika. Apache Tika Official. https://tika.apache.org/, 2017. [Online; accedida 4-mayo-2017].

[18] Google. Comenzar a utilizar SSL. https://support.google.com/adwords/answer/2580401?hl=es-419, 2017. [Online; accedida 17-agosto-2017].

Anexos

A. Proceso de indexación y consulta mediante Solr

La instalación inicial de un servidor de Solr se elaboró en local para hacer el desarrollo y pruebas durante todo el proyecto, en el puerto por defecto, 8983. Se creó un *core* en el que se realizaría la indexación y búsqueda de ficheros introducidos por el usuario. Para que la indexación del fichero y consulta de las palabras necesarias se ejecutara de la forma más rápida y adecuada posible, fue imprescindible la configuración del *core* a través del esquema proporcionado por Solr. Este esquema es un fichero que viene por defecto con la instalación de la plataforma con una configuración determinada; no obstante, es posible modificarlo si la estructura por defecto no es la deseada. En el esquema, se determinan los diferentes *tipos de campos* en los que se va a estructurar cada fichero introducido en Solr y los campos pertenecientes a esos tipos. Además, también se indica cómo se indexa el fichero y se consulta en el índice creado en la indexación. Por defecto, se crean varios tipos de campos y campos predefinidos, los cuales se indexan y consultan mediante las técnicas configuradas por defecto.

La configuración que se realizó en el núcleo creado de Solr se concentró en crear un índice en el que sólamente se ubicaran palabras pertenecientes a áreas geográficas del tipo de área seleccionado por el usuario que existieran en el texto dado. Es decir, si el usuario pretende encontrar en el texto las comarcas de Aragón, el índice del campo dedicado a comarcas se compone de sólo comarcas encontradas en el texto proporcionado, por lo tanto buscará en ese campo. Para ello, se tuvieron que realizar varios pasos, y comprender el funcionamiento de los *Analyzers*, *Tokenizers* y *Filters* suministrados por Solr.

Inicialmente, todo el contenido del fichero se traslada a un campo denominado "text", el cual es de tipo "string", y por lo tanto, no tiene una configuración más allá de la que tiene por defecto. Este campo se encarga de abarcar ese contenido para posteriormente trasladarlo a varios campos que se ocupan de la configuración del indexado y la consulta, ya que Solr no puede trasladar en un principio el contenido de un documento rico (PDF, Word...) a otro campo que no sea *text*.

Una vez se tiene el contenido en ese campo *text*, se crean dos campos distintos que pertenecen a dos tipos diferentes de campo, uno por cada categoría de áreas que se pueden buscar: municipios y comarcas. Esto es, se han creado los campos "campoMz "campoC", pertenecientes a los tipos creados "myTextFieldMz "myTextFieldC", respectivamente. Obligatoriamente, cada campo debe tener un tipo asociado. Es en estos tipos donde se especifica cómo se quieren crear los índices y cómo se quiere realizar consultas en ellos (creando un índice por cada campo), que es la configuración más importante de toda esta sección. A continuación, se traslada el contenido almacenado en el campo *text*, es decir, todo el fichero a los dos campos creados, a través del recurso de Solr denominado *copyField*. Esta copia se realiza antes de que se realice el análisis, lo que significa que se tienen los dos campos nuevos con el mismo contenido original. De esta manera, se puede realizar una configuración diferente para cada tipo de campo, en este caso, para cada categoría de área, que es lo que se pretende.

En los dos tipos de campo creados se utiliza una configuración prácticamente idéntica, excepto por una diferencia que se comentará a continuación. Solr se encarga de crear un índice en cada uno de los campos que pertenece a dicho tipo ("campoM", "campoC" en este caso) a partir del contenido del texto almacenado en cada campo. Para ello, Solr ofrece una multitud de maneras para gestionar los datos textuales, englobadas en tres conceptos: Analyzers, Tokenizers y Filters. Una vez que el texto pasa por los tres recursos ordenadamente, la salida final resulta ser el índice asociado a un campo.

- Los **Analyzers** se utilizan tanto al indexar el documento, como a la hora de consultar, y se encargan de examinar el texto contenido en los campos y generar un flujo de *token*s con todo el contenido del campo. Inicialmente, los estados del índice son los *tokens* generados por los Analyzers.
- Los **Tokenizers** tienen como misión recibir el flujo de texto y partirlo en *tokens* para proporcionárselo al Filter. En este caso, para los dos campos manejados se crea un tokenizer mediante la clase *WhiteSpaceTokenizerFactory*. El WhiteSpaceTokenizer tiene como función partir el texto del campo en *tokens*, tratando espacios en blanco y puntuaciones como delimitadores (excepto puntos que no vayan seguidos de un espacio en blanco). De esta manera, se generan *tokens* que pertenecen a palabras sueltas que aparecen en el texto.
- Por último, los **Filters** se encargan de recibir el flujo de *tokens* que genera el previo Tokenizer u otro Filter, y vuelven a producir un flujo de *tokens*. Puede haber varios Filters asociados a un Analyzer, en cuyo caso cada Filter recibiría como entrada la salida del Filter anterior a él.

Para la configuración del proyecto, se han utilizado los Filters creados por: PatternReplaceFilterFactory, ShingleFilterFactory y KeepWordFilterFactory.
En primer lugar, el filtro **PatternReplaceFilterFactory** simplemente reemplaza los caracteres que se indiquen por otros diferentes en cada *token*. Como parámetros, se le tiene que indicar, obligatoriamente, la cadena de patrón que se quiere reemplazar, y la cadena por la cual se quiere sustituir. En este caso, se ha utilizado para eliminar posibles puntuaciones que se hayan quedado dentro del *token*, para poder dejar únicamente el nombre del área en el índice.

El filtro creado por **ShingleFilterFactory** es capaz de combinar varios *tokens* en un único *token*, considerándolo así como una sola palabra, y no como varias. Como parámetros, es posible indicar el mínimo de *tokens* por palabra, el máximo de *tokens* por palabra, y varios más de los que no se ha tenido la necesidad de utilizar para alcanzar el objetivo. Puesto que muchas de las áreas geográficas tratadas en este proyecto son palabras compuestas (por ejemplo, Campo de Borja, Ribera Baja del Ebro, y Hoya de Huesca/Plana de Uesca), se ha aplicado este filtro para combinar cada palabra con las 4-5 siguientes palabras que le siguen para componer en un solo *token* el nombre entero del área, y poder guardarlo como tal en el índice. Se ha comprobado que con obtener palabras de hasta 5 *tokens*, se recogen todas las comarcas, y con un máximo de 6, todos los municipios.

El filtro creado por **KeepWordFilterFactory** descarta todos los *tokens* que recibe

excepto aquellos que están indicados en una lista facilitada. Como parámetros, es obligatorio indicarle la ruta del fichero que contiene la lista de palabras a considerar. En este fichero de configuración se debe incluir una palabra por línea. Además, se puede indicar a este filtro que realice comparaciones *case sensitive* o no. Aquí es donde se encuentra la diferencia que tienen los tres tipos de campo creados, puesto que cada uno tiene una lista de palabras diferentes que quiere mantener. De esta forma, el tipo de campo "myTextFieldM"tiene asociado un fichero con la lista de municipios totales que hay en Aragón y el tipo de campo "myTextFieldCüna lista de todas las comarcas de Aragón. Con este filtro se termina el proceso de indexación, y se genera el índice del campo con las palabras que proceden de este filtro, es decir, las áreas encontradas según la categoría seleccionada por el usuario.

Se ha comentado la importancia que tiene el esquema en la indexación de un documento al subirlo a la plataforma de Solr, en la cual se crea un índice para cada uno de los dos nuevos campos originados que contiene áreas geográficas de Aragón encontradas en el texto, y que servirá para comparar las consultas realizadas a estos campos. A continuación, se describe el proceso de consultas, también denominado query.

El proceso de *consulta* es más sencillo, porque una vez que se tienen las palabras que forman los índices, basta recuperar cuáles están y cuáles no a través de consultas de las áreas. Es decir, si realizando una consulta con el nombre de una comarca al "campoC"(perteneciente a comarcas) Solr devuelve como resultado el texto introducido, entonces esa comarca existe en el índice, y por tanto ha sido encontrada anteriormente en el texto.

De esta manera, se realizan tantas consultas consecutivas como áreas haya de la categoría elegida. Si el usuario ha decidido buscar municipios, se realizarán 730 consultas (tantas como municipios existen) al "campoM", y cuando Solr devuelva resultados positivos, se indica que ese municipio ha sido encontrado en el índice del campo. La lista de consultas a realizar se obtiene de los mismos ficheros creados anteriormente para la indexación que contienen las áreas geográficas según su categoría. Por ejemplo, si el usuario decide buscar comarcas, antes de realizar las consultas se realiza una lectura del fichero asociado de comarcas y se recogen todas ellas en una lista, la cual se utiliza para elaborar tantas consultas como comarcas haya, en ese caso.

Por otro lado, para que el proceso de consulta se realice correctamente, se ha tenido que modificar su configuración en el esquema también, de forma similar a la indexación ya comentada anteriormente, pero el proceso es más breve. Para cada tipo de campo creado (los dos tipos que corresponden a las dos diferentes categorías de áreas) se ha establecido un Analyzer que se encarga del proceso de indexación, en el que se han indicado tanto el Tokenizer como los Filters. Para el proceso de consulta es necesario otro Analyzer que se dedique exclusivamente a atender consultas en ese determinado tipo de campo, y para el que hay que decidir qué se quiere realizar con la consulta recibida.

En este caso, se ha decidido que la forma más rápida sería realizar la consulta con la palabra completa del área, y que Solr se dedique a obtener esa palabra completa sin elaborar ninguna proceso de partición (no como se ha hecho en la indexación). Esto es, como el proceso de indexación ya se ha encargado de la tarea de las palabras com-

puestas y ha guardado las áreas compuestas por varias palabras tal cual en el índice, simplemente es necesario realizar la comparación de la misma manera, con la palabra compuesta. Para lograr este objetivo, se ha utilizado el Tokenizer **KeywordTokenizerFactory**, el cual trata el texto de la consulta como un *token* único, pudiendo coincidir con la misma palabra si existe en el índice del campo. No es necesario ningún recurso más para el proceso de consulta. Todo este proceso descrito se representa en la figura16

Ambos procesos, tanto indexación como consulta, se han realizado a través del API que ofrece Solr para ello, pudiendo llamarlo desde el programa Java para tener la configuración integrada mediante el componente SolrJ. **SolrJ** es un cliente Java que se utiliza para añadir documentos al índice, actualizarlo, o realizar consultas en él. Oculta muchos detalles de la conexión con Solr y permite a la aplicación interactuar con la plataforma con métodos de alto nivel. Así pues, el fichero se indexa en Solr mediante la clase **ContentStreamUpdateRequest**, indicando como parámetro la url "/update/extract", y señalando el campo al que se pretende volcar el contenido del fichero.

Posteriormente, se realiza la ejecución de consultas consecutivas a través de la clase **SolrQuery** al campo que corresponda a la categoría de área elegida, recogiendo la respuesta con la clase **QueryResponse**. Si se quiere buscar en municipios, la consulta se realizará en el "campoM". De esta manera, se van seleccionando los resultados en los que Solr devuelve el fichero para indicar ese área como encontrada.

Los últimos Request Handlers de Solr se encargan de recibir la petición tramitada a Solr y manejarla de acuerdo a la configuración de indexación y consulta establecida previamente. Nativamente, soporta la indexación y búsqueda de los documentos estructurados XML, CSV y JSON. Para poder importar documentos ricos como Word y PDF, Solr utiliza Tika [17] para la extracción de texto del documento y trabaja la de este.

Una de las pruebas que se idearon en la primera parte del desarrollo en la que se estaba trabajando con la identificación de áreas con Solr fue la siguiente: Se pensaron dos distintas aproximaciones en cuanto a la estructura que debería tener el proceso de indexación en el *schema.xml*. Una de ellas, es la definitivamente elegidaA, en la cual se realiza una partición en *tokens* mediante el WhitespaceTokenizer, después se combinan diferentes n-grams originando muchos más nuevos *tokens*, y finalmente se realiza un filtrado con el fichero de las áreas para dejar sólamente las áreas que aparecen en el documento.

La otra aproximación reflexionada consistía en dividir también en *tokens* mediante el WhitespaceTokenizer, y hacer lo mismo con el fichero de texto en el cual están las áreas (una en cada línea). De esta manera, se tendría tanto el documento del usuario como la lista de áreas separados por *tokens*. En ese momento, se realiza un filtrado en los *tokens* originados del documento a través de los *tokens* originados de la lista de áreas, y del documento quedan solo palabras pertenecientes a las áreas. A continuación, se realiza un combinado de n-grams con los *tokens* que han resultado del paso anterior (es decir, con todo partes de palabras de áreas). Finalmente, se vuelve a hacer un filtrado del resultado con el fichero de las áreas. En la figura X se puede ver visualmente.

Como se puede observar, la segunda aproximación es algo más larga en pasos, aunque se elimina el tener que hacer combinación de n-grams de todo el documento, pero existen dos filtrados en lugar de uno solo. Para comprobar el rendimiento en cada una, se midió el tiempo que costaba del indexado y de las consultas, es decir, el proceso de identificar las áreas del documento. Se observó que los tiempos en ficheros pequeños era parecido en ambos casos, pero algo menor en el segundo. Sin embargo, en ficheros más grandes, la diferencia de tiempo entre ambos casos era algo mayor, y la primera aproximación tenía menor coste. Además, teniendo en cuenta que van a ser documentos extensos la mayoría que se utilizan en esta aplicación, tiene mejor rendimiento esa primera estructura.

B. Proceso de consulta de las variables y valores a representar en el mapa mediante SPARQL

El proceso empieza justo cuando Solr termina su tarea encontrando las áreas geográficas que aparecen en el texto. Se envía una petición *HTTP GET* a la plataforma que contiene Aragopedia, en el que se almacenan todos los informes que están disponibles. De esta manera, se recogen todos ellos y se muestran al usuario en forma de lista desplegable autocompletable. Cabe decir que se presentan dos maneras de elegir el informe:

1. la forma avanzada/inteligente, en la que el usuario escribe en una entrada de texto y se van mostrando las palabras acordes a lo escrito.

2. la forma por defecto, en la que se muestran todos los informes en una gran lista, sin necesidad de escribir.

Cuando el usuario elige uno de los informes representados, se realiza una nueva petición al recurso *dsd*[3]de Aragopedia para obtener las variables asociadas a ese informe. Cada informe tiene un listado de características de las cuales se elige una que se representará en el mapa. Una vez seleccionada dicha característica, también se tramita una petición para obtener el periodo de tiempo del que se desea información, ya que la información a lo largo del tiempo varía según el informe; es decir, para una temática puede haber datos desde, por ejemplo, el año 2010, pero para otra puede ser que solo existan datos más recientes (desde el año 2015).

Cuando se han especificado datos de ambos, se muestran al usuario las variables disponibles para que este seleccione la que se represente posteriormente en el mapa. Cabe destacar que se ha realizado un filtro en las variables mostradas al usuario para que solamente muestre las variables numéricas, es decir, que se pueda representar en el mapa.

Por último, si del conjunto de variables que se habían obtenido en el paso anterior existe alguna que no es numérica, sino que es de tipo *codelist* (tiene un rango de valores posible), se muestran al usuario éstas variables con su colección de valores para que fije un valor para cada variable. Para lograr esto, se ha realizado un bucle de peticiones *HTTP GET* al recurso *skos*[4], cada una con su correspondiente variable, y se han guardado los resultados de todos los valores posibles de cada una, para poder mostrarlo al usuario. Este paso es necesario para fijar valores en el cubo dimensional, y evitar filas repetidas del mismo área con diferentes valores del informe elegido.

Cuando el proceso termina, se muestra al usuario una tabla en la que se representa cada área geográfica junto a su valor correspondiente del informe elegido. En este momento, ya se tienen todos los datos necesarios que necesita el mapa para poder ser interpretado (la información a representar), pero es importante añadir una configuración previa del formato final que va a tener el mapa. Aquí es donde empieza la

[3]http://opendata.aragon.es/recurso/iaest/dsd/informe.json
[4]http://opendata.aragon.es/kos/iaest/codigo.json

siguiente y última sección del desarrollo. Todo el proceso comentado en esta sección está representado en la figura 17.

C. Transmisión de información a Tbl2Map y generación de mapa

La forma en la que se ejecuta el paso de información desde el cliente de Txt2Map ha sido resuelta a través de un formulario oculto en el que se realiza una redirección en la cual es posible adjuntar datos en el cuerpo de la petición.

En un principio, el paso de la información se realizó con una petición GET en lugar de una POST por sencillez del proceso y para tener una versión estable, y posteriormente se completó con la solución comentada por ser más completa, más difícil de atacar, y no tener restricciones de cantidad de información transmitida, además de ser oculta en la URL para el usuario.

D. Implantación de seguridad mediante HTTPS

Cada vez más sitios web implementan el *protocolo HTTPS* para poder cifrar el canal de comunicación por el que se envía la información. Aunque es altamente recomendable llevarlo a cabo en sitios web en los que se emplea información personal o financiera y en esta aplicación no se hace uso de información de ese tipo en este momento, es recomendable implementarlo. Los pasos que se tienen que realizar son los siguientes [18]:

- **Obtener un certificado SSL para el sitio web.** Un certificado SSL es un documento electrónico que verifica la identidad de la empresa y permite que el servidor web realice una encriptación segura con el navegador web.
- **Instalar el certificado SSL.** La forma de instalación varía según el servidor web y el tipo de certificado obtenido.
- **Identificar las páginas del sitio web para asegurarlas con SSL.** Lo imprescindible es encriptar las páginas en las que se transmiten datos personales o financieros, como contraseñas de acceso, números de cuenta...
- **Editar los vínculos a las páginas indicadas como seguras.** Las páginas que se han decidido implementar con el protocolo HTTPS deben modificarse en el código, y cambiar de "http" a "https" todas ellas para poder ponerlo en práctica.
- **Verificar que funcionan las páginas seguras.** Realizar pruebas después de las modificaciones es el último paso fundamental para comprobar que todo sigue funcionando con normalidad, pero con la seguridad ya implementada.

E. Diagramas de Gantt

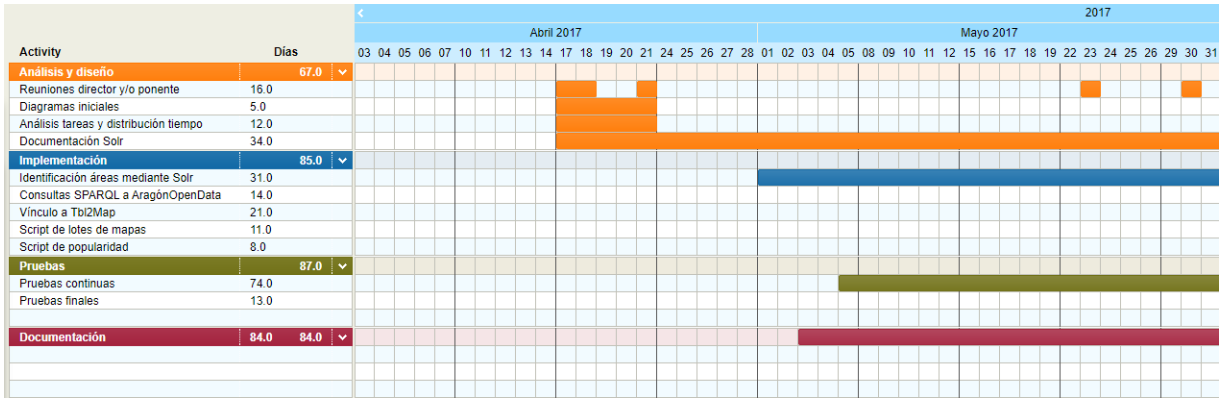

Figura 13: Diagrama de Gantt de los meses abril y mayo

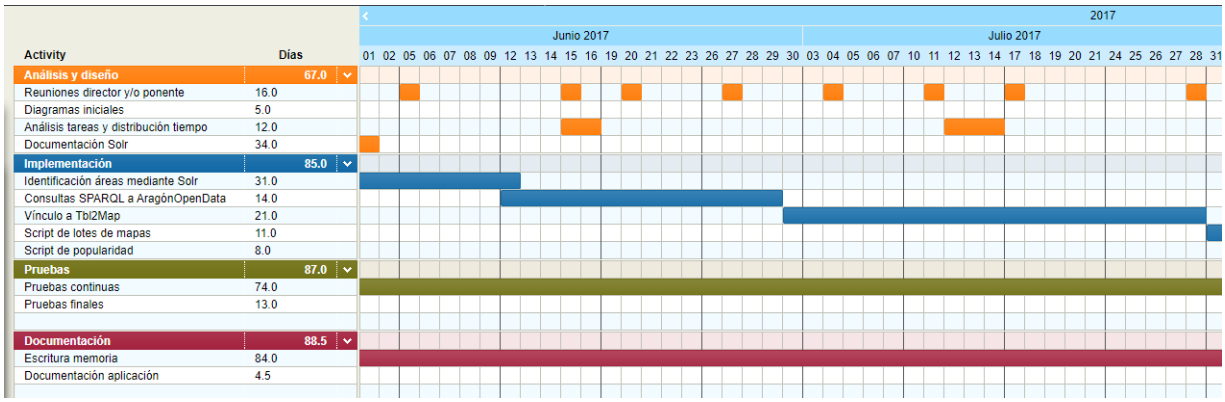

Figura 14: Diagrama de Gantt de los meses junio y julio

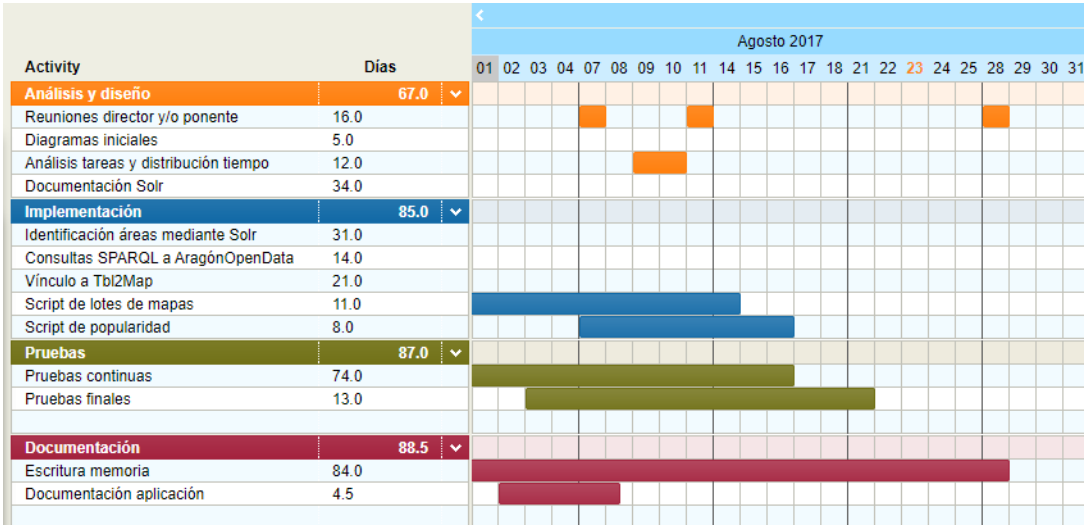

Figura 15: Diagrama de Gantt del mes agosto

F. Diagramas y esquemas complementarios

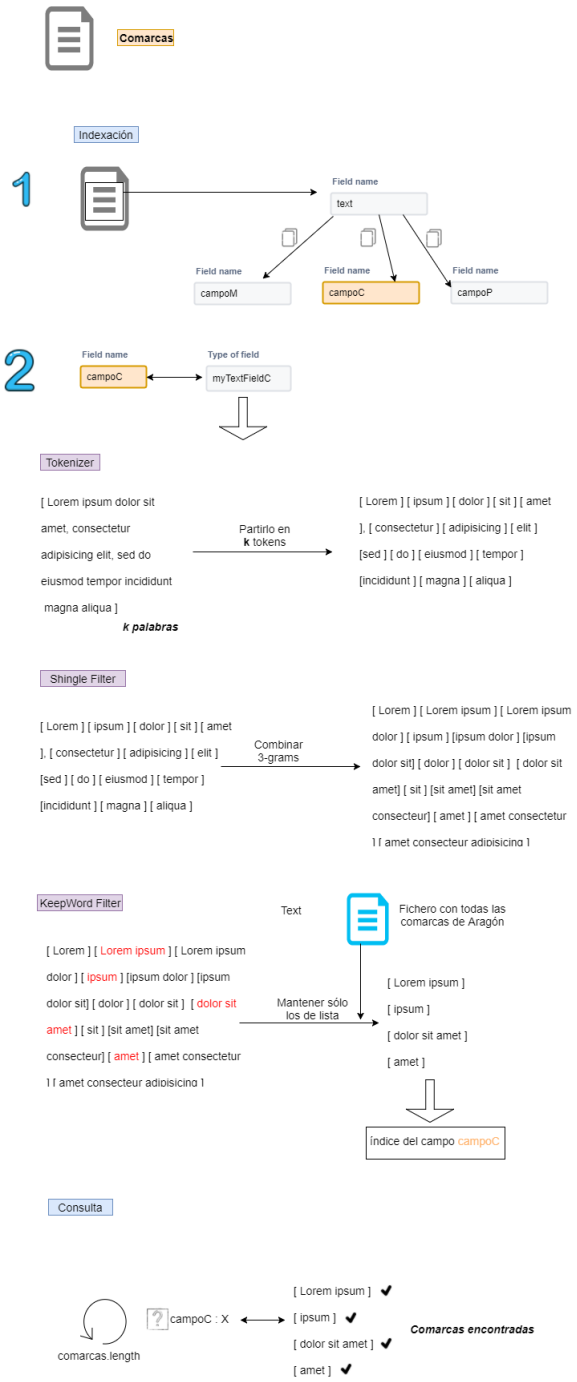

Figura 16: Proceso de búsqueda de referencias geográficas realizado por Solr

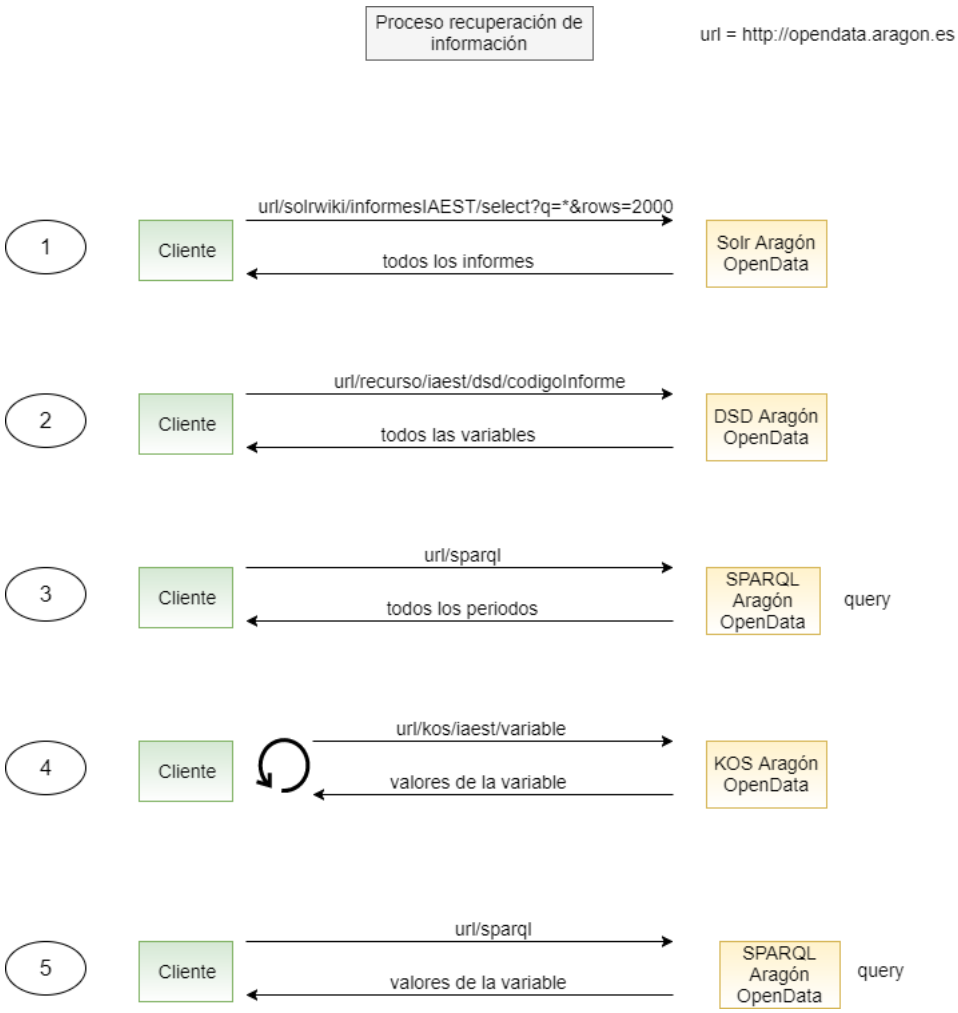

Figura 17: Proceso de obtención de las variables de interés

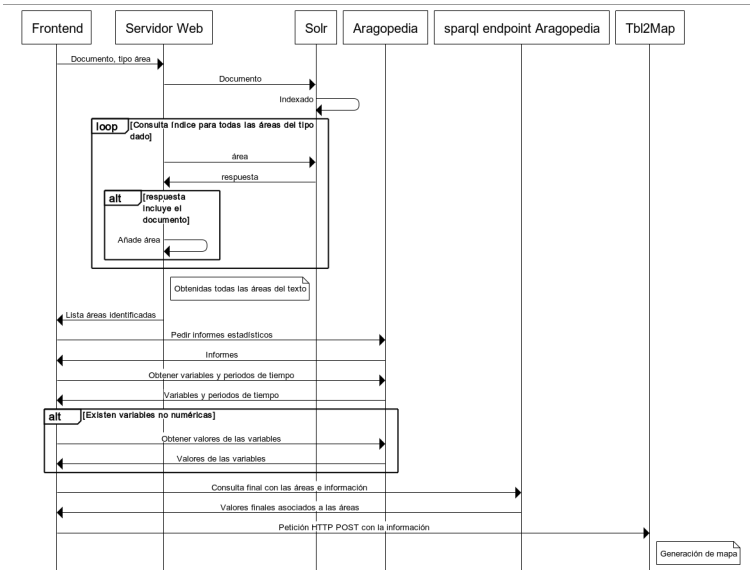

Figura 18: Diagrama de secuencia de la generación de un mapa a partir de un documento

Figura 19: Pantalla inicial desde un *smartphone*

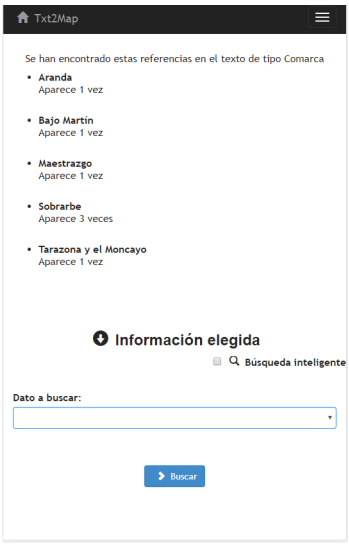

Figura 20: Búsqueda información de áreas desde un *smartphone*

Figura 21: Información recibida de áreas desde un *smartphone*

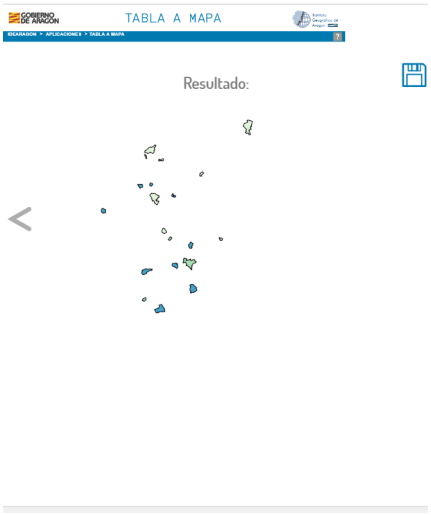

Figura 22: Mapa generado desde un *smartphone*

G. Glosario

Analyzer: Examina el texto de los campos y genera una cadena de *tokens*. Son especificados en los tipos de campo.

Case Sensitive: Representa que importa el uso de mayúsculas y minúsculas, siendo los elementos diferentes con cada aproximación.

CSV: Comma-Separated Values, es un tipo de documento en formato sencillo para representar datos en forma de tabla, en el que las columnas se separan por comas, y las filas por saltos de línea.

Core: Es un software que constituye una parte fundamental del sistema operativo.

ElasticSearch: Es un servidor de búsqueda basado en Lucene. Provee un motor de búsqueda de texto completo, distribuido y con capacidad de multi-tenencia con una interfaz web RESTful y con documentos JSON.

Filter: Reciben una entrada compuesta por diferentes *tokens* y producen una cadena de *tokens*. Normalmente, tratan cada *token* secuencialmente y decide si pasa al siguiente filtro o se descarta.

GeoJSON: Es un formato estándar abierto diseñado para representar elementos geográficos sencillos, junto con sus atributos no espaciales, en JavaScript Object Notation.

GML: Geography Markup Language, es un sublenguaje de XML descrito para el modelaje, transporte y almacenamiento de información geográfica

GPX: GPS eXchange Formato, es un esquema XML pensado para transferir datos GPS entre aplicaciones. Se suele usar para describir puntos, recorridos y rutas

IDE: Integrated Development Environment, es un editor de código fuente, herramientas de construcción automáticas y un depurador, que permiten desarrollar un proyecto de forma más visual y más cómoda.

JPG: Joint Photographic Group, es un formato de almacenamiento y la transmisión de imágenes fotográficas en la Web

JSON: JavaScript Object Notation, es un formato de texto ligero para el intercambio de datos.

Parser SAX: Analiza el lenguaje XML y definir la estructura de un documento, pudiendo obtener información deseada de una manera sencilla.

PDF: Portable Document Format, es un formato de almacenamiento para documentos digitales

Plugin: Un plug-in o complemento es una aplicación (o programa informático) que se relaciona con otra para agregarle una función nueva y generalmente muy específica. Esta aplicación adicional es ejecutada por la aplicación principal e interactúan por

medio de la interfaz de programación de aplicaciones.

SHP: Shapefile es un formato estándar de facto para el intercambio de información geográfica entre Sistemas de Información Geográfica

endpoint SPARQL: Un endpoint SPARQL permite a los usuarios consultar una base de conocimientos a través del lenguaje SPARQL. Los resultados típicamente se devuelven en uno o más formatos procesables por máquina.

Tokenizer: Divide la cadena de *tokens* que recibe en diferentes *tokens* según su categoría, donde cada *token* es una subsecuencia de caracteres.

Tomcat: Tomcat es un contenedor web con soporte de servlets y JSPs. Incluye el compilador Jasper, que compila JSPs convirtiéndolas en servlets. El motor de servlets de Tomcat a menudo se presenta en combinación con el servidor web Apache.

XLS: Formato de archivo usualmente utilizado como hoja de cálculo.

XML: eXtensible Markup Language, es un meta-lenguaje que permite definir lenguajes de marcas, desarrollado por la World Wide Web Consortium, y utilizado para almacenar datos en forma legible.