Universidad Zaragoza
1542

Trabajo Fin de Grado

Geolake Search: Una nueva forma de buscar datos georreferenciados
Geolake Search: A new way of searching georreferenced data

Autor/es

Sergio Martín Segura

Director/es

Francisco Javier López Pellicer

Escuela de Ingeniería y Arquitectura

2018

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. Sergio Martín Segura,

con nº de DNI 73017399N 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) Geolake Search: Una nueva forma de buscar datos georreferenciados

Geolake Search: A new way of searching georreferenced data

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

Zaragoza, viernes 2 de febrero de 2018

Fdo: Sergio Martín Segura

Agradecimientos

En primer lugar, quiero agradecer el trabajo de mis profesores y profesoras que supieron convertir a un proyecto de diseñador en un informático apasionado. No es sencillo inspirar una vocación como ellos lo han hecho.

Me gustaría agradecer en concreto a dos profesores el papel que han supuesto en mi formación. En primer lugar, a José Manuel Colom y a su lucha por reivindicar la ingeniería. Por no dejarnos olvidar que somos algo más que programadores y que tenemos que hacernos valer. En segundo lugar, a Rubén Béjar, del cual podré decir que me enseñó casi todo lo que sé sobre la ingeniería del software. Cuatro asignaturas juntos nos dieron para mucho.

También me gustaría agradecer a Francisco Javier López Pellicer su apoyo y orientación a lo largo de este trabajo. No es nada fácil emprender un proyecto así en soledad, tras cuatro años de trabajos en grupo, pero Javier ha sabido mantener mi motivación para seguir superándome en cada prueba hasta llegar a encontrar el investigador que había en mí. No hay nada comparable al hormigueo que se siente al saber que eres la primera persona en el mundo en hacer algo así.

Mi siguiente agradecimiento va para esos compañeros a los que he echado en falta estos meses. Al capitán Rubén Moreno, por enrolarnos a Iñigo y a mí en su tripulación y enseñarnos lo que es asumir el liderazgo; a Santi Gil por ser el primero que me ofreció su ayuda en el momento que más la necesitaba; y a mis compañeros electrónicos, Fernando y Octavio, por haberme acompañado en más proyectos de los que podría reconocer aquí.

Durante estos cuatro años, la frase que más veces he oído es “Es que tú estás metido en muchas cosas” y es precisamente a esa otra media vida, a la que agradezco haberme enseñado lo que la Universidad no me podía enseñar. A mis camaradas del colectivo, Adrián y Dani, por haber luchado a mi lado por cambiar el mundo. A mis compañeros y compañeras de la sectorial RITSI, donde aprendí más sobre nuestra profesión que en mi propia carrera y donde conocí gente espectacular en todos los rincones de España. Y también a mis compañeros y compañeras del Consejo de Estudiantes que, tras años de inactividad, comienza ahora a caminar con fuerza.

Finalmente, quiero agradecer a Helena todo lo que ha supuesto para mí éste último año. Su capacidad para insuflarme su fuerza de voluntad cuando la mía flaqueaba. Su capacidad para animarme cuando todo lo veía negro. Haberme dado un proyecto y un propósito.

Finalmente, agradezco a mi familia todo el apoyo que me han dado y, más importante aún, la confianza que depositaron en mí cada vez que les dije “Está todo controlado” y el trabajo que han hecho desde que nací para abrirme todas las puertas que pudieran.

Resumen ejecutivo

En la era de la información, los datos espaciales suponen una parte importante de ese gran océano de datos. Los geoportales son los buscadores responsables de hacer esos datos accesibles a los usuarios, pero algo pasa con ellos. En los últimos veinte años apenas se ha visto una evolución en sus modelos y siguen tratando los datos como entonces. Esto ha llevado a que su utilidad no logre equipararse a la de otras herramientas similares que sí han logrado evolucionar en sus respectivos ámbitos.

¿Por qué no son tan útiles?

Uno de los objetivos de este trabajo ha sido escudriñar y analizar el comportamiento de los principales geoportales que se usan hoy en día, investigar su funcionamiento y la razón de que funcionen así, y conocer su historia e identificar sus aciertos y sus errores.

Lo que tienen en común todos catálogos y geoportales es que no han sabido profundizar en las colecciones de datos que almacenan, olvidando la importancia del dato en sí. Una idea sencilla que no ha brotado hasta que la tecnología de *big data* la ha hecho factible.

¿Y ahora qué?

El primer paso para solventar un problema es detectar sus causas, el segundo es idear una solución, el tercero es ponerla a prueba. En este trabajo se han implementado diversas pruebas de concepto para estudiar la viabilidad de una idea clave: ir más allá de las colecciones de datos. Así cada prueba ha tratado un aspecto diferente del mismo problema: cómo descomponer las colecciones, hasta ahora indivisibles desde el punto de vista del catálogo tradicional, y extraer de ellas los datos para realizar una búsqueda de la colección basada en su contenido espacial real y no en descripciones más o menos acertadas de la colección.

Se ha trabajado con tecnologías de *big data* como Hadoop y Elasticseach; con tecnologías de desarrollo de aplicaciones empresariales como Spring y Spring Data; y con tecnologías de programación web *front-end* como el framework React.js. No todas están presentes en la solución final, pero todas aportan conocimiento y experiencia sobre un problema que se ha ido desvelando como un camino tras una montaña.

¿Entonces, funciona?

El método tecnológico tiene un cuarto paso, analizar los resultados y proponer modificaciones. Tras el desarrollo de la prueba de concepto final, se han estudiado los resultados de ésta para comprobar así que, efectivamente, no sólo es una idea viable, sino que tiene un potencial que va más allá. La tecnología Elasticsearch permite librarse de la limitación del volumen de datos a la vez que se desbloquea un abanico de funcionalidades nunca vistas en un geoportal tradicional.

Sin lugar a duda, los geoportales necesitan evolucionar y Geolake es una solución que apuesta firmemente por ello.

Índice

Lista de figuras

Lista de tablas

Glosario

Para entender el mundo de la información geográfica es necesario tener unas nociones mínimas del modelo y la terminología. Esencialmente podemos descomponer el problema en el siguiente diagrama:

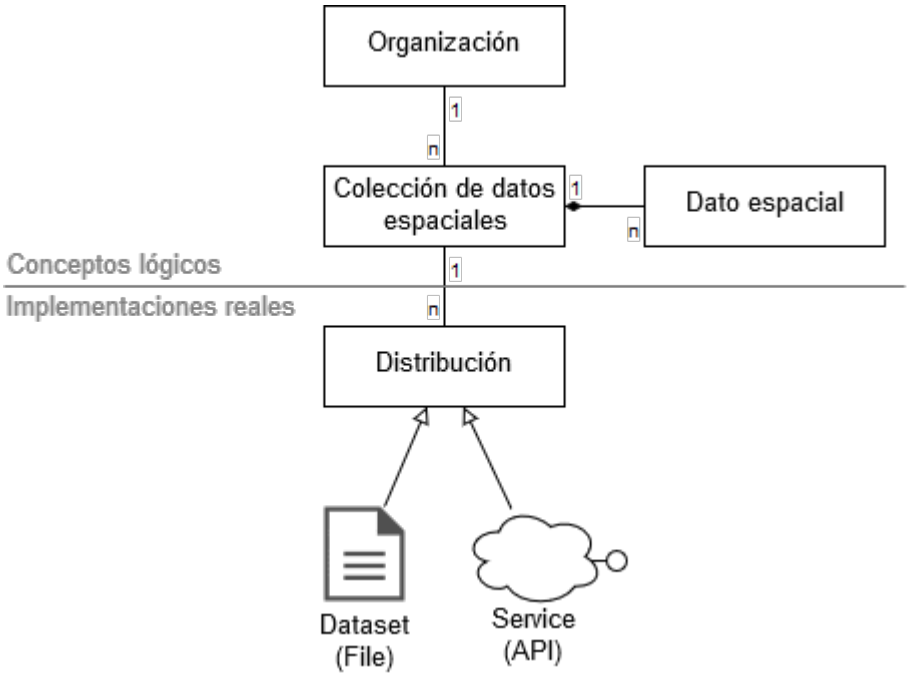


Figura 1. Conceptos básicos en la distribución de información geográfica

Colección de datos: También conocido como colección de datos espaciales. A menudo es usado indistintamente como "*dataset*". Para una redacción clara y sin ambigüedades se reservará el término "*colección de datos*" para el concepto lógico que representa la agrupación de datos espaciales lógicos.

Dato espacial: También conocido como objeto espacial, dato georreferenciado o dato geográfico. Es la unidad mínima de información espacial. Representa un objeto con una geometría y un lugar en el espacio concretos. Puede contener información adicional en forma de atributos. Ej.: si el objeto espacial es una línea que describe el trazo de un río uno de sus atributos podría ser su nombre.

Distribución: Implementación digital de los datos lógicos. Una misma colección de datos puede tener varias distribuciones. Las distribuciones pueden ser de dos tipos:

- **Datasets:** Ficheros físicos que almacenan los datos.
- **Servicios:** API que ofrecen los datos explotables programáticamente.

Organización: Es la propietaria de la colección de datos. Existen diversos matices que podrían requerir diferenciar entre productor, editor, dueño, etc. pero a efectos académicos podemos restringirnos a la simplicidad de las organizaciones.

Además de la taxonomía básica existen otros conceptos de interés como:

Datos: Conviene recordar que los datos son valores concretos para atributos sin ningún otro significado. Carecen de valor por sí mismos si no se les dota de un contexto y una interpretación. Ej.: *temperatura = 45ºC*

Faceta: Es una forma de llamar a un atributo por el cual se puede filtrar un resultado. Ej.: de un *coche* son facetas su *marca*, su *color* o su *antigüedad*.

Filtro: El acto de filtrar por un atributo y un valor implica que, de un conjunto de resultados, se descartan aquellos que para ese atributo no posean ese valor.

Geoportal: Portal de búsqueda que permite localizar datos espaciales, a menudo en forma de colecciones.

Información: Conocimiento que se extrae de unos datos en un contexto y con unas relaciones conocidas entre ellos. Ej.: *una temperatura de 45ºC en Zaragoza supone una temperatura potencialmente peligrosa debido a que sobrepasa el umbral de riesgo de golpe de calor.*

Query: Es el contenido de una búsqueda. En él se define el término por el que se busca y las condiciones de esta.

Recurso: Se entiende como recurso todo objeto lógico o físico interesante para el usuario. En este contexto se acotará su uso a colecciones de datos, distribuciones y datos espaciales.

Servicio: La información espacial no sólo está disponible en forma de ficheros, sino que con los años han aparecido diversos protocolos que ofrecen la información de forma programática mediante una API. Estos servicios pueden implementar estándares OGC como WMS (Web Map Service) o WFS (Web Feature Service).

1. Algo pasa con los geoportales

Desde los mapas de Anaximandro de Mileto en el Siglo VI a.C., pasando por las mediciones de Eratóstenes, los descubrimientos de Magallanes o los ojos de Yuri Gagarin, el ser humano se ha fascinado por la geografía de la tierra que habita. La geografía y la cartografía se han ido introduciendo en otras disciplinas como la epidemiología, la ingeniería energética, la arquitectura, la historia, el turismo, etc. Esto es debido a un hecho muy significativo:

> *Toda realidad material tiene una posición y una forma en el espacio y la forma en la que se relaciona con otras realidades materiales depende de esa posición y esa forma.*

Por extensión, toda información que represente realidades materiales podrá incluir en sí misma la posición y forma de la realidad que representa. De ese modo se podrá estudiar su relación con otras realidades adyacentes. Uno de los ejemplos más recientes del poder de la información espacial lo acabamos de comprobar con el reciente escándalo sobre la publicación de un mapa de calor que una famosa aplicación deportiva para móviles hizo como campaña publicitaria ya que a partir de ella se ha podido *deducir* la localización de instalaciones militares secretas en zonas de conflicto (Hern, 2018).

La agrupación de esa información con datos espaciales se hace en lo que hoy conocemos como Infraestructura de Datos espaciales (IDE) (Williamson, Rajabifard, & Feeney, 2004). Los datos pueden ser desde conjuntos de puntos con información adjunta hasta fotos aéreas, mapas manuscritos, etc.

Actualmente, los geoportales constituyen un elemento clave para las IDE (Magure & Longey, 2005). Entre otras funcionalidades, los geoportales permiten localizar las colecciones de datos espaciales que necesitamos mediante un catálogo. Sin ellos, el uso de información espacial sería mucho más heterogénea y deberíamos conocer de antemano el origen de cada fuente para solicitarle el recurso que buscamos.

Es intuitivo pensar que, del mismo modo que los buscadores de páginas web nos permiten encontrar páginas que no conocemos, los geoportales nos permiten encontrar colecciones de datos que no conocemos. No obstante, los buscadores han evolucionado hasta convertirse en enormes sistemas llenos de funcionalidades interconectadas con avanzados algoritmos, pero …

> *¿Por qué los geoportales no han evolucionado de igual forma?*
>
> *¿Siguen siendo útiles hoy para encontrar datos espaciales?*
>
> *¿Ofrecen todo tipo de funcionalidades como cabría esperar en la era del big data?*

Este Trabajo de Fin de Grado tiene como objetivo identificar los defectos que hoy arrastran los geoportales y proponer una solución que puede ayudar a arrojar luz sobre lo que debería ser su futuro.

1.1. Los portales de datos

Un portal de datos es una aplicación perteneciente normalmente a la familia de los buscadores y catálogos que ofrece una ventana de acceso y búsqueda a un gran repositorio de datos archivados virtualmente. Los primeros portales de datos (por ejemplo, NOMADS (Rutledge, Alpert, Stouffer, & Lawrence, 2003)) nacen influenciados por las bibliotecas digitales (Arms, 2000). Uno de los aspectos clave de las bibliotecas digitales fue que, debido a la imposibilidad tecnológica de digitalizar todos los documentos, se adoptó la idea con las que las bibliotecas llevaban trabajando desde el siglo XIX, el catálogo de fichas. Estas fichas originalmente eran creadas por la biblioteca y describían la información básica de los recursos originales tales como libros, mapas o gráficos y eran fácilmente manejables. Estas fichas sí eran fácilmente digitalizables debido a su escaso contenido en comparación con un libro entero, por lo que fueron elegidas como objeto de búsqueda para los portales.

Las aplicaciones no tardaron en abrirse a la web y adaptar su paradigma de búsqueda a nuevo entorno como muestra el trabajo de Andrei Border, investigador de IBM y uno de los responsables de AltaVista en *"A taxonomy of web search"* (Broder, 2002), cuya adaptación a la Web del modelo clásico de recuperación de información se muestra en la Figura 2.

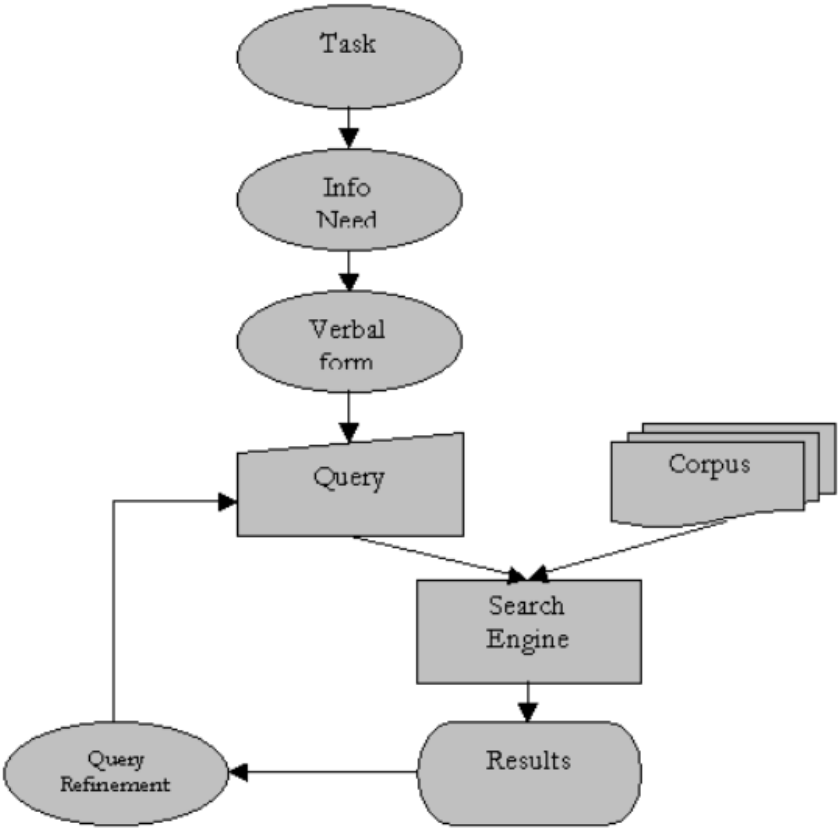


Figura 2. El modelo clásico para recuperación de información aumentado para la Web según Broder

En esta publicación ya se definen los comportamientos de los usuarios ante los distintos tipos de búsqueda, pero serán dos responsables de Yahoo! en "*Understanding User Goals in Web Search*" (Rose & Levinson, 2004) quienes refinarán la idea y defenderán que toda búsqueda puede ser clasificada en una de las siguientes categorías:

- *Búsqueda navegacional* donde el usuario conoce el recurso o lugar al que quiere acceder, pero no recuerda su dirección exacta por lo que recurre al buscador para encontrarla. Ejemplo: *"página web de unizar"*
- *Búsqueda informacional* donde la intención del usuario es obtener información sobre algo, sin saber exactamente qué recurso desea encontrar. Dentro de esta categoría se encuentran las *búsquedas directas* donde el usuario sabe qué información desea sobre un tema. Ejemplo: *"fechas de apertura de matrícula de la universidad"*. En contraposición a la búsqueda directa está la *búsqueda indirecta* en la que el usuario busca cualquier tipo de información sobre algo. Ejemplo: *"trabajos de fin de grado"*. Finalmente se incluyen en esta categoría la *búsqueda de consejo*, la *búsqueda de una dirección física* y la *búsqueda de una lista* las cuales tienen nombres bastante autoexplicativos.
- *Búsqueda de recurso* en la cual lo que se busca como fin no es obtener información sino un recurso. En esta categoría estaría la clásica *descarga,* pero también se incluiría el consumo de *entretenimiento* como imágenes o vídeos; la *interacción* con herramientas web como conversores o calculadoras o finalmente la *obtención* de un recurso que no requeriría de un ordenador para ser consumido como una imagen que se va a imprimir.

1.2. El origen los geoportales

De forma análoga al problema que resolvieron las bibliotecas en su momento, los institutos de investigación y las agencias geográficas tuvieron que afrontar el reto de ofrecer una forma sencilla de encontrar sus recursos espaciales. La idea de geoportal resultó al inicio extraña dentro del concepto de IDE, pero pronto se adoptó a nivel nacional (Béjar, y otros, 2004) y europeo (Bernard, Kanellopoulos, Annoni, & Smits, 2005).

Dichos geoportales necesitaban de interfaces de búsqueda estándar para los catálogos. Dicha tarea fue abordada por la Open Geospatial Consortium (OGC) que en 1999 publicaría la primera versión de, una interfaz de uso común para los catálogos geoespaciales (CAT) de modo que, independientemente del cómo estuvieran implementadas, pudieran interoperar entre ellas y ofrecer una forma de explotación unificada (OGC, 1999). La especificación CAT de 1999 describía un modelo de datos comunes para todos los catálogos caracterizado por pocos atributos, un modelo abstracto de interoperabilidad entre los catálogos, lo que permitía ejecutar una búsqueda en un catálogo y que éste propagara la consulta a otros catálogos conocidos, y la arquitectura lógica del servicio en su conjunto (Figura 3). La especificación CAT estaba se adaptaba perfectamente al mundo de la información geográfica y tenía visión de convertirse en un sistema de recuperación de información. Sin embargo, en la versión de 2007 (Figura 4), pretendieron abrir su compatibilidad y adaptarse a los estándares e ideas de las bibliotecas digitales, por ejemplo, utilizando como modelo de referencia Dublin Core (OGC, 2007).

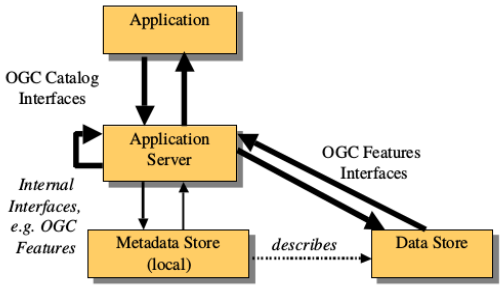


Figura 3. Arquitectura de referencia de la especificación CAT (1999)

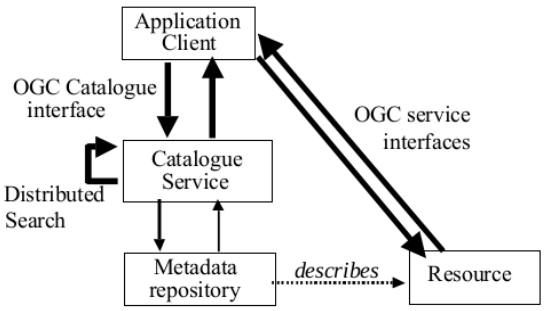


Figura 4. Arquitectura de referencia de la especificación CAT (2007)

Como se ha mencionado ya, las bibliotecas digitales en los años 90 no ofrecían servicios de catálogo que permitieran buscar por el contenido de los libros ya que no estaban digitalizados en su mayoría. Sin embargo, sí que era posible analizar las colecciones de datos ya que éstas estaban digitalizadas. Sin embargo, el acercamiento de los geoportales al modelo de bibliotecas digitales hizo que heredaran su forma de tratar las colecciones de datos como recursos monolíticos alejándose de la capacidad de hacer búsquedas en profundidad y otorgando más relevancia al metadato que describe el recurso que a la capacidad autodescriptiva del propio dato en sí.

Además, la información geográfica cuenta con una peculiaridad que comparte, en cierta forma, con los libros: su localización espacial. Al no analizarse los datos dentro de una colección, se tuvo que idear una manera resumida y simplificada de describir el espacio que los mismos abarcan. Una vez más, las bibliotecas digitales tenían la solución (Hill, 2004): Optar por la forma geométrica más sencilla de describir y buscar en un sistema de dos coordenadas, el rectángulo. A este rectángulo se le da el nombre de *bounding box* y permitirá que indicando tan sólo las coordenadas de sus esquinas opuestas se puedan filtrar rápidamente las colecciones de datos cuyos *bounding box* estén dentro o en contacto con el *bounding box* de búsqueda (Figura 5 – En Naranja: la búsqueda. En azul: *bounding box* del resultado).

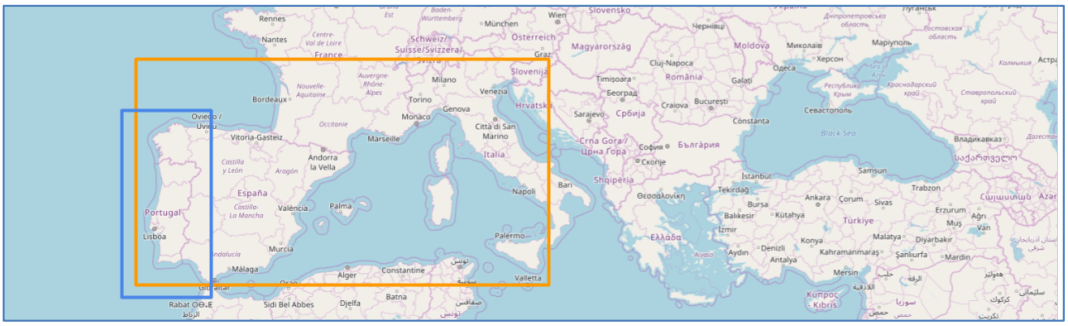


Figura 5. Mapa con dos bounding boxes

En cuanto al comportamiento de los usuarios frente a estos portales, los tipos de búsqueda que Rose y Levinson identificaron son perfectamente aplicables a las búsquedas en ellos. Ejemplos prácticos de búsqueda serían (Tabla 1):

Tabla 1. Búsquedas navegacionales, informacionales y de recursos en geoportales

Tipo	Subtipo	Ejemplo
Navegacional	Directa	*"Listado de carriles bici por el Ayto de Zaragoza 2017"*
Informacional	Directa	*"Carriles bici de zaragoza"*
	Indirecta	*"Transporte en zaragoza"*
	Consejo	*-- no aplicable a geoportales --*
	Localización	*-- implícita en todas las búsquedas --*
	Lista	*-- implícita en todas las búsquedas --*
De recurso	Descarga	*"Carriles bici de zaragoza en GeoJSON"*
	Entretenimiento	*-- no aplicable a geoportales --*
	Interacción	*-- el propio geoportal ofrecería la interacción --*
	Obtención	*-- Imprimir -- "Carriles bici de zaragoza"*

1.3. ¿Qué hace un geoportal?

Para abordar la pregunta que da título a este apartado se han seleccionado algunos de los geoportales más representativos. Cada uno de ellos representa cómo afrontan un mismo problema distintas administraciones o empresas de distintos países. Desde la directiva INSPIRE, un proyecto de la Unión Europea, pasando por la implementación de éste en España con la directiva IDEE, la administración del Gobierno de los Estados Unidos, la Universidad de Stanford, la administración de Noruega o incluso la perspectiva de una empresa privada como ESRI, dedicada a la consultoría de software para infraestructuras de datos geográficos.

Todos ellos serán analizados siguiendo una metodología específica diseñada para este propósito. Sus detalles se encuentran en el Anexo A.

1.3.1. INSPIRE Geoportal (Comisión Europea, Unión Europea)

INSPIRE Geoportal[1], desarrollado por la Unión Europea tiene como el objetivo de ser el centro neurálgico de la información espacial de la Comunidad Europea. Su existencia no se puede entender sin comprender antes el proyecto que lo apadrina y le da nombre, la Directiva 2007/02/CE por la que se establece una infraestructura de información espacial en la Comunidad Europea, más conocida como Directiva INSPIRE. Dicha directiva propone una estandarización en la información espacial que generan las diversas agencias de los diversos países miembros. De esta forma se establece una normativa de obligado cumplimiento que define una serie de requisitos que tiene que tener todo dato compartido por una agencia que cumpla unas características concretas.

La tecnología usada es un desarrollo propio fruto de la licitación del proyecto en el año 2010 con un presupuesto de casi 650,000€. Se compone de una aplicación web *front-end* pura que envía las consultas directamente a la API HTTP de Solr (Solr Documentation).

En el momento de la publicación de este trabajo, ofrece metadatos de 65.079 recursos y es el portal de referencia para publicar colecciones y servicios de datos tanto para las propias agencias europeas como para agregar los datos y servicios publicados en los diferentes geoportales nacionales de los países miembros de la Unión Europea (Figura 6).

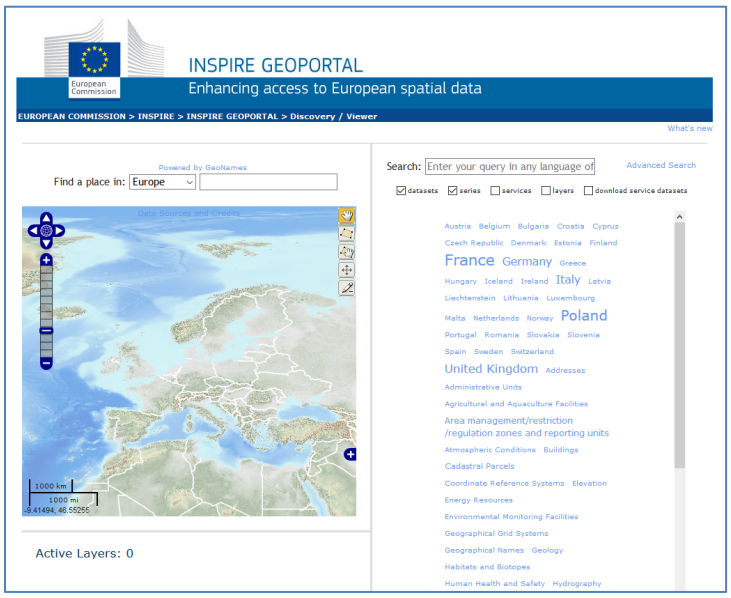


Figura 6. Geoportal INSPIRE

Tras realizar el análisis (Anexo B.1) nos encontramos con uno de los portales que claramente han envejecido peor. Una interfaz desfasada, unas funciones poco intuitivas y una representación de los datos que hace prácticamente inútil la búsqueda exploratoria ya que la única forma de encontrar información es saber previamente dónde debe estar.

[1] http://inspire-geoportal.ec.europa.eu/discovery/#

1.3.2. Geospatial Platform (FGDC, EEUU)

Geospatial Platform[2] es un proyecto lanzado desde el Gobierno de los Estados Unidos de América para ofrecer la información espacial de sus diversas instituciones gubernamentales y asociadas de una forma centralizada y accesible para todos.

La tecnología que usa es CKAN, el proyecto *open source* líder mundial en soluciones para portales de datos abiertos que además cuenta soporte para datos espaciales. Uno de los motivos de su éxito es su arquitectura modular que permite adaptar y extender las funcionalidades de forma rápida y sencilla para cubrir las necesidades de cada portal.

En el momento de publicación de este trabajo, ofrece 150.992 recursos y es el portal de referencia de todas las administraciones gubernamentales estadounidenses tanto a nivel federal como a nivel estatal (Figura 7).

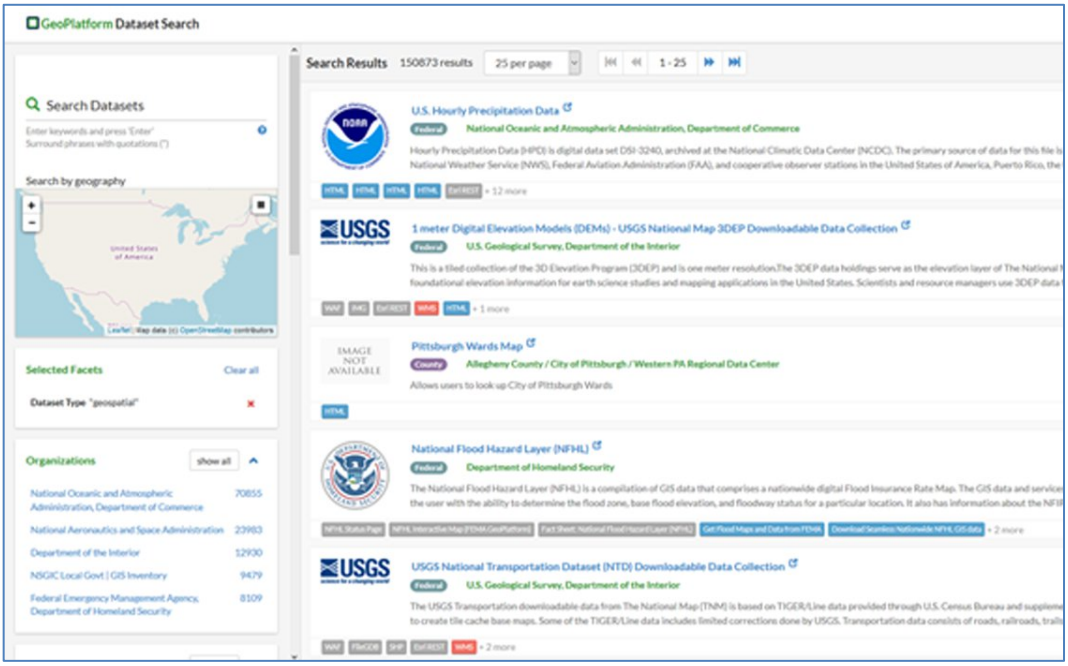


Figura 7. Geoportal GeoPlatform

Tras realizar el análisis (Anexo B.2) encontramos uno de los mejores portales en términos de diseño y usabilidad. No cuenta con el mejor inspector de datos, pero cumple su función.

[2] https://ckan.geoplatform.gov/

1.3.3. EarthWorks (Universidad Stanford, EEUU)

EarthWorks[3] nace en la Universidad de Stanford, concretamente a cargo de la institución encargada de la gestión de sus bibliotecas, Stanford Libraries. El objetivo es ofrecer las herramientas para buscar y acceder a los recursos espaciales que la propia biblioteca posee, así como datos de otras instituciones asociadas (Reed, 2015). Dichos recursos provienen de equipos de investigación, tanto de la propia universidad como de universidades colaboradoras. No todos son públicos por lo que ofrece la posibilidad de autenticarse en el sistema para acceder a ellos.

La tecnología usada es GeoBacklight, una "plataforma de descubrimiento de recursos geoespaciales"[4] que ofrece las herramientas necesarias para crear un geoportal. En su desarrollo contribuyen el MIT y las universidades de Stanford, Princeton y Nueva York. Sus puntos fuertes son la capacidad de inspeccionar los datos desde el propio portal antes de ser descargados.

En el momento de publicación de este trabajo, ofrece 26.523 recursos y es el portal de referencia utilizado por todos los departamentos de la Universidad de Stanford relacionados de forma directa o indirecta con recursos geoespaciales (Figura 8).

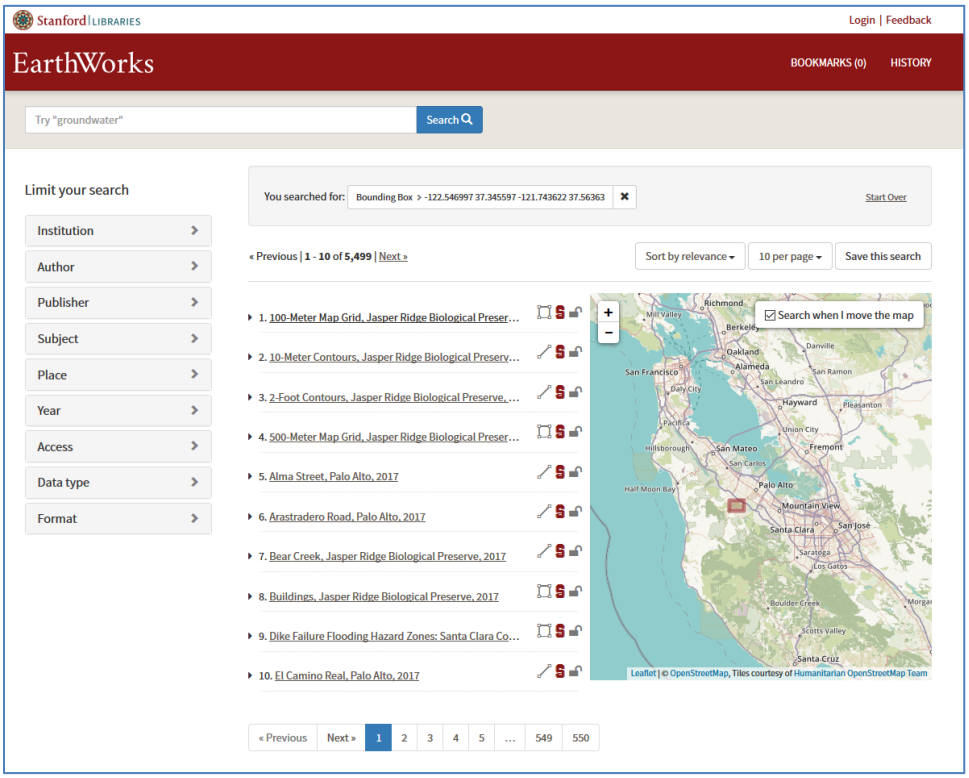


Figura 8. Geoportal EarthWorks

Tras realizar el análisis (Anexo B.3) nos encontramos con un portal sólido, que se aprovecha de las restricciones que impone a la hora de subir los datos para ofrecer una experiencia homogénea, por ejemplo, en los tipos de capas en función de su geometría donde limita los posibles valores a puntos, líneas, polígonos, rasters y mixtos.

[3] https://earthworks.stanford.edu/
[4] http://geoblacklight.org/

1.3.4. GeoNorge (Kartverket, Noruega)

GeoNorge[5], el geoportal noruego desarrollado por el Kartverket, la autoridad noruega responsable del mantenimiento de la información geográfica[6] propone un geoportal que ofrezca información espacial de las diversas agencias noruegas.

La tecnología que usa es GeoNorge Kartkatalog, un geoportal de código libre desarrollado por la propia agencia[7].

En el momento de publicación de este trabajo, ofrece 4.378 recursos y es el portal de referencia para publicar información geoespacial por parte de las entidades públicas noruegas (Figura 9).

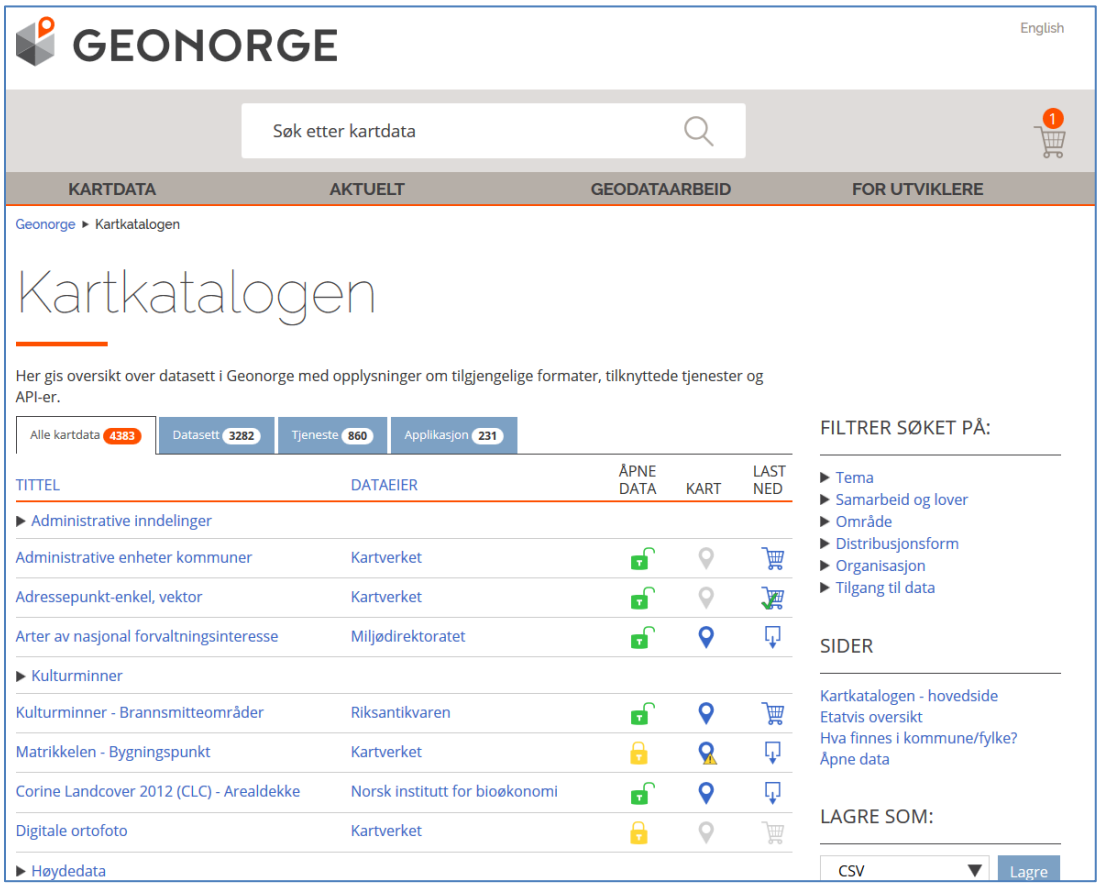


Figura 9. Geoportal GeoNorge

Tras realizar el análisis (Anexo B.4) encontramos un portal con ideas frescas y un estilo diferente. Prescinde totalmente del mapa para ofrecer una búsqueda centrada en el contenido y donde brilla especialmente es en sus herramientas de post-procesado para la descarga, las cuales, si bien no son muy intuitivas de usar, ofrecen gran potencia a quien las sepa explotar.

[5] https://kartkatalog.geonorge.no
[6] https://www.kartverket.no/en/About-The-Norwegian-Mapping-Authority/
[7] https://github.com/kartverket/Geonorge.Kartkatalog

1.3.5. IDEE (Ministerio de Fomento, España)

Coordinada y dirigida por el Consejo Superior Geográfico, la Infraestructura de Datos Espaciales de España (IDEE) nace en 2002 con la intención de integrar bajo una directiva común todos los datos espaciales de las distintas administraciones públicas estatales, autonómicas y locales. Implementa además la directiva europea INSPIRE por lo que no sólo establece criterios compartidos a nivel nacional, sino que cumple aquellos requeridos a nivel europeo. La directiva IDEE indica que dichos datos espaciales han de ser ofrecidos a través de un geoportal por lo que en 2004 se lanza su primera versión (Rodríguez Pascual, López Romero, Abad Power, & Sánchez Maganto, 2005). No será hasta 2014 cuando el sistema adopte un proyecto como GeoNetwork (Rodríguez Pascual, 2014).

Actualmente cuenta con dos herramientas. *Visualizador Básico IDEE* donde los usuarios pueden consultar los datos sobre un mapa y *Catálogo de Metadatos IDEE*[8] donde pueden buscar y descargar los metadatos de un amplio catálogo. El objeto de análisis será el catálogo debido a que la herramienta de visualización no puede ser considerada un portal de búsqueda.

La tecnología antes mencionada, GeoNetwork, es un proyecto *open source* que ofrece las herramientas necesarias para crear un geoportal. De ese modo, el Instituto Geográfico Nacional, encargado de desarrollar y mantener el catálogo, adaptó el código para que se adecuara a las necesidades específicas de la directiva. En el momento de publicación de este trabajo, el *Catálogo de Metadatos IDEE* ofrece metadatos de tan sólo 6.031 recursos y es el portal de referencia de todas las administraciones públicas españolas que producen datos geoespaciales (Figura 10).

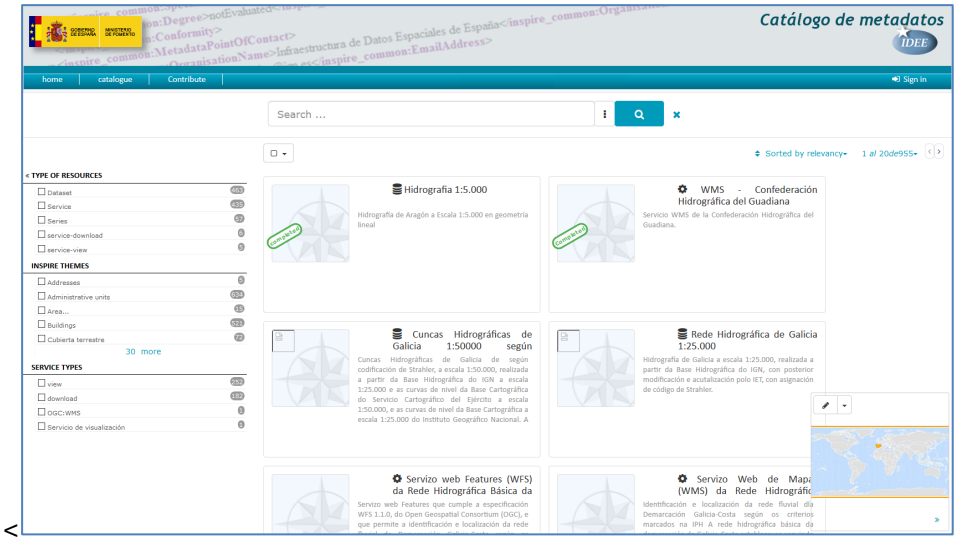


Figura 10. Geoportal IDEE

Tras realizar el análisis (Anexo B.5) se concluye que los principales defectos del portal, además de su falta de actualizaciones, se derivan de su foco en ofrecer facilidades para descargar metadatos de colecciones de datos, series y servicios sin dar un soporte que pueda ser considerado realmente equivalente para obtener o acceder a dichos recursos.

[8] http://www.idee.es/csw-inspire-idee/srv/eng/catalog.search#/home

1.3.6. ArcGIS Hub (Esri, EEUU)

ArcGIS es el buque insignia del gigante de los sistemas de información geográfica, ESRI. Se trata de su producto estrella, un software que promete una potencia y facilidad de uso por encima de la competencia. Es importante destacar en este punto que, aunque se trate de la herramienta más privativa y menos abierta de las analizadas en este trabajo, no se le va a aplicar ningún sesgo relativo a ello ni va a influir en la valoración de sus funcionalidades. Todo lo contrario, el motivo de incluirla es poder contar con una perspectiva diferente de lo que pueden ofrecer los geoportales.

Siendo ArcGIS el nombre de la *suite* completa, la tecnología a analizar va a ser el geoportal ArcGIS Hub[9] dado que es el que más se aproxima a los ya analizados. Concretamente una versión alojada en su web a modo de demo donde los futuros clientes pueden ver el potencial de esta. En el momento de la publicación de este trabajo, ofrece 97.006 recursos de ejemplo provenientes de agencias públicas, institutos y demás organizaciones (Figura 11).

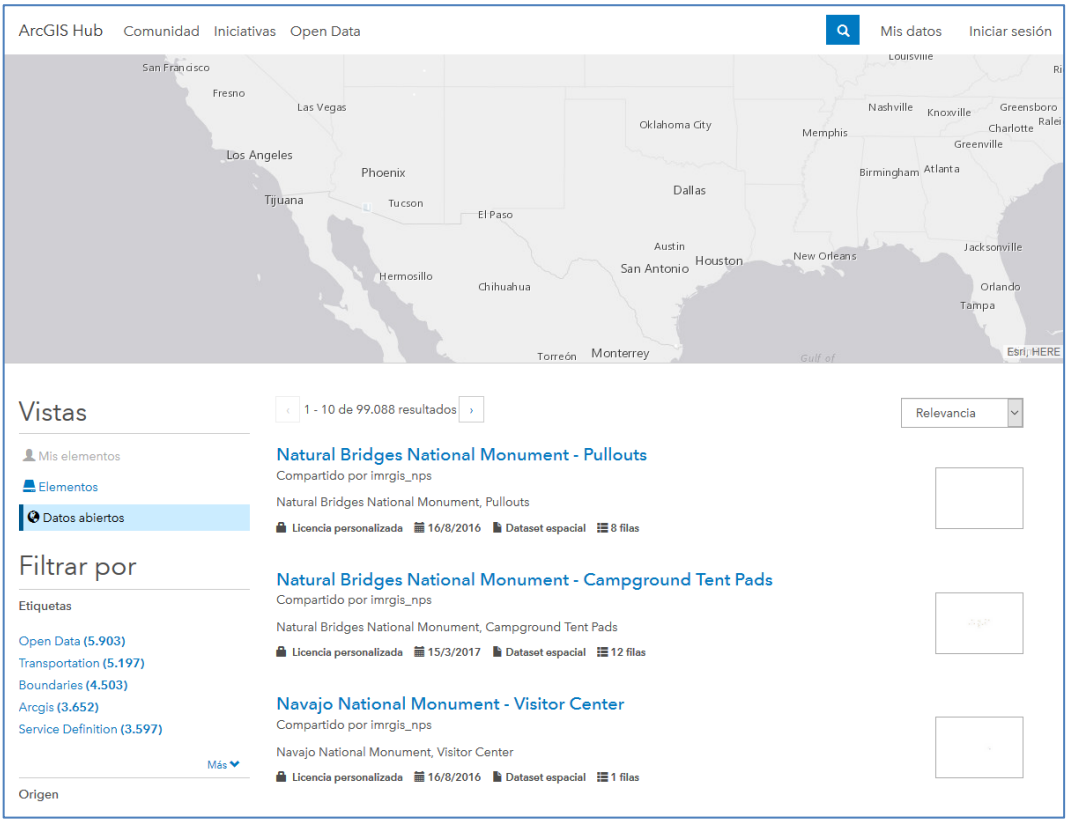


Figura 11. Geoportal ArcGIS Hub

Tras realizar el análisis (Anexo B.6) encontramos con una herramienta a la altura de las expectativas y el presupuesto. Un portal sólido, con gran atención al detalle y fruto de constantes revisiones e incorporaciones de ideas nuevas. Su visor de datos es el mejor de entre los analizados debido a su potencia y opciones permitiendo visualizar cada detalle de los datos, así como crear vistas generales en base a sus atributos.

[9] http://hub.arcgis.com/datasets

1.3.7. Comparativa general

Tras analizar cada uno de los portales podemos extraer sus funcionalidades comunes, entendiéndolas como algo imprescindible para tener en cuenta (Anexo B.7). Esto no quiere decir que deba ser implementado en este proyecto, sino que forma parte de las funcionalidades esenciales de un geoportal de cara al usuario que habitualmente lo utiliza.

Además, podemos contrastar el volumen de datos que manejan (Figura 12) y así hacernos una idea de la dimensión y el volumen del problema en cuanto a capacidad de carga.

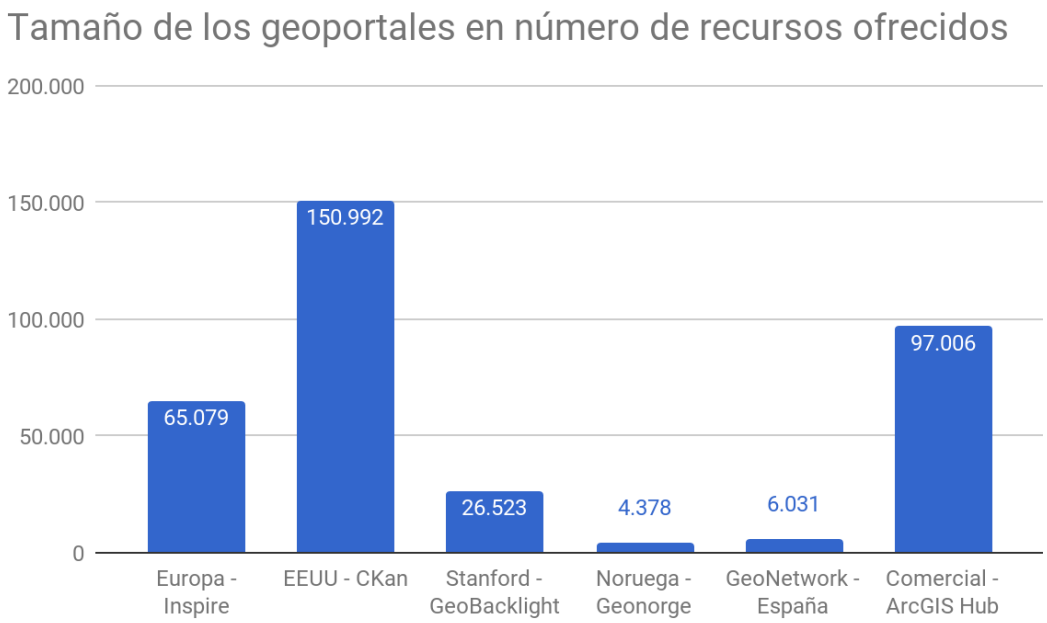


Figura 12. Comparativa del tamaño de los catálogos de los geoportales

1.4. ¿Qué hace *mal* un geoportal? La aguja en el pajar

Cualquiera que los use regularmente compartirá la opinión de que no son pocas las veces que un geoportal arroja tal cantidad de información irrelevante que hace imposible localizar un recurso en la primera búsqueda. Esto es debido a varios defectos:

- El tratamiento de las colecciones de datos como conjuntos monolíticos y opacos.
- El uso de *bounding boxes* como único indicador de la cobertura espacial de una colección.
- El uso de la cobertura espacial como filtro binario y no como factor de relevancia.

Como ya se ha mencionado, los catálogos de los geoportales se inspiraron en las bibliotecas digitales y en cómo estas trataban los recursos de forma monolítica, describiéndolos como un todo mediante una ficha o registro de metadatos, normalmente redactada por la propia biblioteca.

Ya en 1999 Max J. Egenhofer y Werner Kuhn (Egenhofer & Kuhn, 1999) señalaban este mal uso de los metadatos:

> *Emerging metadata standards, however, focus more on what the data producers have to say than on what the data users need to know*

Los metadatos perdían en parte su utilidad ya que eran los autores de estos quienes decidían qué información era relevante para la colección de datos sin tener en cuenta qué información era útil para el consumidor que buscara en un catálogo.

El segundo defecto está en la técnica de filtrado geoespacial del rectángulo o *bounding box* la cual plantea un serio problema: el distanciamiento entre la realidad de los datos y su simplificación en los metadatos adjuntos. De este problema se podían manifestar diversos escenarios:

- Groenlandia está más al norte, más al sur, más al este y más al oeste que Islandia. Toda búsqueda sobre el área de Islandia arrojaría también los resultados de Groenlandia (Figura 13).
- Zaragoza se encuentra en el interior de la Península Ibérica. Una búsqueda sobre el área de Zaragoza podría arrojar una colección de datos que recogiera información de las costas de la Península Ibérica a pesar de que Zaragoza no tenga playa y ningún dato de esa colección se encontrara en esa zona (Figura 14).

Figura 13. Bounding box de Groenlandia en una búsqueda sobre Islandia

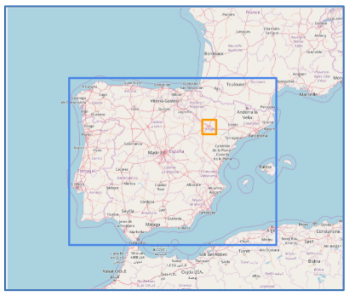


Figura 14. Bounding box de la Península Ibérica en una búsqueda sobre Zaragoza

Los sistemas no eran capaces de averiguar si un resultado que estaban ofreciendo era realmente relevante en esa área.

La última de las claves radica en que se entendió el filtrado por *bounding box* como un filtro binario, del ámbito de la recuperación de datos, y no como un término más de una función de puntuación de un sistema de recuperación de información. De ese modo se perdió la oportunidad de dar resultados relevantes para cada área, lo que hacía que fuera perfectamente posible que una búsqueda sobre un área de la extensión de Europa arrojara como primer resultado un dataset que recogiera datos de un pequeño pueblo de Albania (Figura 15).

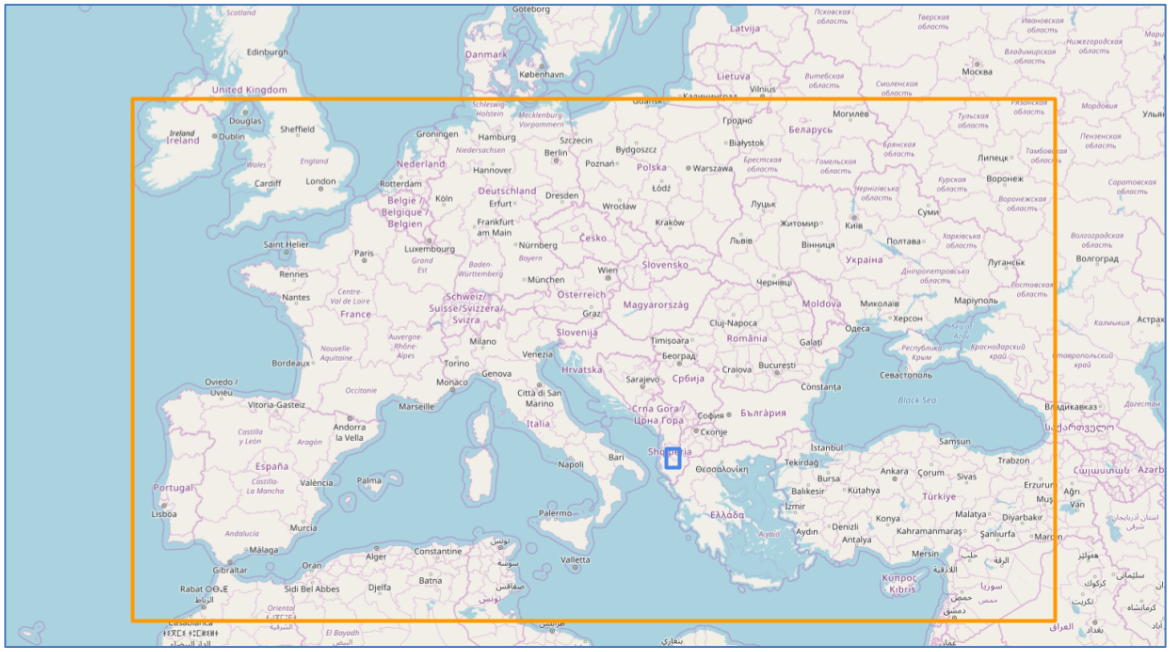


Figura 15. Bounding box de Europa en una búsqueda sobre Albania

Es cierto que no es la primera vez que se intentan solventar estos problemas. En 2015 la tesis doctoral de Walter Rentería Agualimpia, realizada también en la Universidad de Zaragoza proponía un sistema que midiera cuán cercanas eran las geometrías de los *bounding box* del dataset de búsqueda al de cada una de las respuestas (Rentería Agualimpia, 2015). De este modo se obtenía una métrica no binaria y se introducía el concepto de relevancia. No obstante, el problema de ésta medía seguía siendo que dependía de la descripción que el productor hubiera dado de sus datos estando limitados además a un rectángulo.

2. Geolake: una nueva forma de buscar datos georreferenciados

Geolake propone una solución innovadora a los problemas vistos anteriormente. Como ya se ha indicado, las malas decisiones venían de limitaciones tecnológicas y de tener los referentes equivocados. Por ello se ha echado la vista atrás y se ha reanalizado el problema desde su origen, detectando sus cualidades intrínsecas y revisando las influencias externas. La conclusión sido rotunda:

> *Los geoportales han muerto, la web de los buscadores los ha matado.*

Los buscadores de música como Spotify permiten encontrar sus canciones, aunque éstas se encuentren en álbumes. Los buscadores de vídeos como Youtube permiten encontrar los propios vídeos, aunque éstos pertenezcan a una cuenta o canal. Incluso Google con su plataforma Google Books está cambiando el paradigma de las bibliotecas analizando dentro de los propios libros.

El mundo de la información geográfica no puede permitirse quedarse atrás. Por eso es necesario profundizar en las, antiguamente inexpugnables, colecciones de datos y encontrar en ellas la solución integral al problema, porque buscamos datos y los datos son la respuesta. Es decir:

> *Los geoportales no deben publicar colecciones de datos, deben ser buscadores de datos.*

2.1. Casos de uso

En base a lo aprendido en el análisis de los geoportales, podemos entender mejor las necesidades de sus usuarios. Detectar qué puntos son un acierto y cuales son un error en términos de *UX* (Experiencia de Usuario). De este modo procedemos a analizar los casos de uso de la aplicación.

En primer lugar, es importante explicar que se ha optado por una representación multivista debido a que esto permite abstraer casos de uso en una vista para posteriormente refinarlos en las siguientes. Es por esto por lo que se ven relaciones de herencia, las cuales, si uno no está habituado a utilizar casos de uso complejos, pueden resultar extrañas.

No obstante, según Martin Fowler (Fowler, 2003):

> *The key value of use cases lies in the text which is not standardized in UML.*

Por ello la descripción textual de los casos de uso se puede consultar en el Anexo C.

En primer lugar, tenemos una vista de alto nivel del escenario del usuario (Figura 16), en el que se muestran los casos de uso que representan su finalidad última: “Descargar un dataset”, ”Acceder a un servicio”, “Consultar un dataset”, “Descargar el resultado de una búsqueda como dataset” y “Consultar una organización”. Estos son los principales fines que busca un usuario y el resto son medios para alcanzarlos.

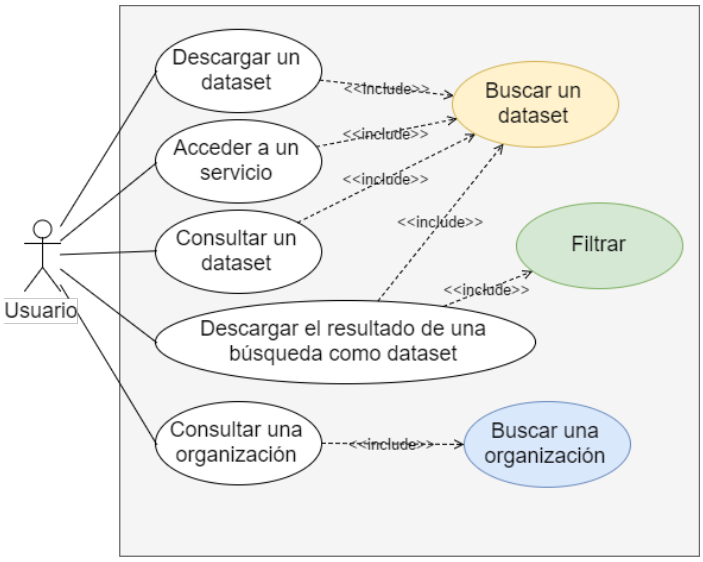


Figura 16. Casos de uso de usuario de alto nivel

En segundo lugar, tenemos la vista de casos de uso relacionados con “Buscar un dataset” (Figura 17). Evidentemente es uno de los casos de uso clave en un buscador. De él heredan dos casos de uso más específicos que tienen sus raíces en el comportamiento se usuario. Por eso se especifican dos subtipos de usuario. La diferencia entre ambos casos es que uno de ellos incluye el filtrado mientras que en el otro es opcional mediante una extensión.

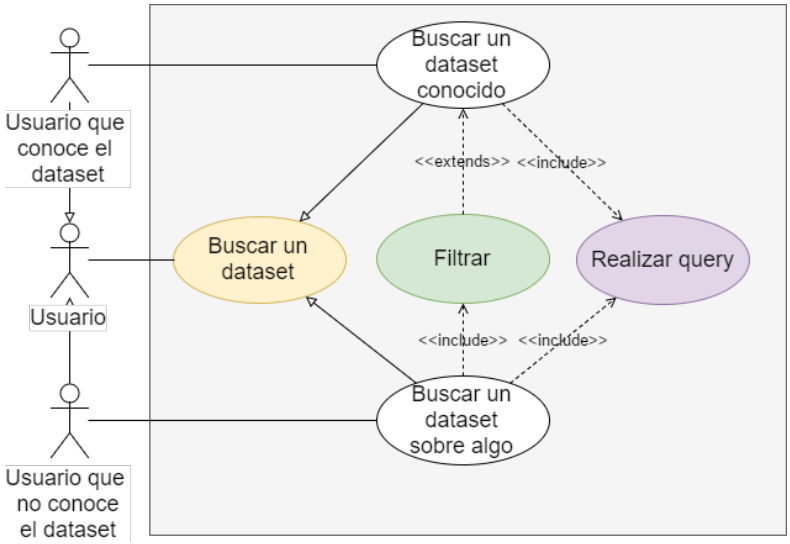


Figura 17. Casos de uso bajo Buscar un dataset

En tercer lugar, tenemos la vista de casos de uso relacionados con “Buscar una organización” (Figura 18). Al igual que en la vista anterior, existen unos casos de uso específicos que heredan del caso de uso genérico para concretarlo.

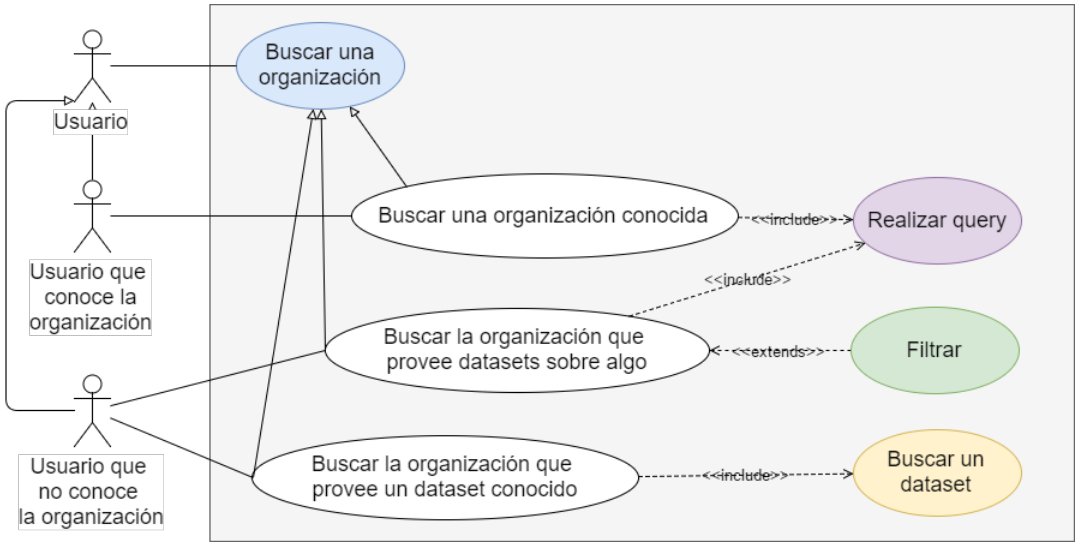


Figura 18. Casos de uso bajo Buscar una organización

En cuarto lugar, tenemos la vista de los casos de uso relacionados con “Filtrar”. A diferencia de las dos anteriores, en este caso no se ha utilizado la relación *generalización/concreción* porque, a pesar de que se pueden entender bajo una relación de herencia, el caso de uso “Filtrar” puede incluir a ambas, algo que no contempla la herencia. Una de las notas UML indica que mínimo ha de extenderse una de las dos para que el caso de uso tenga sentido.

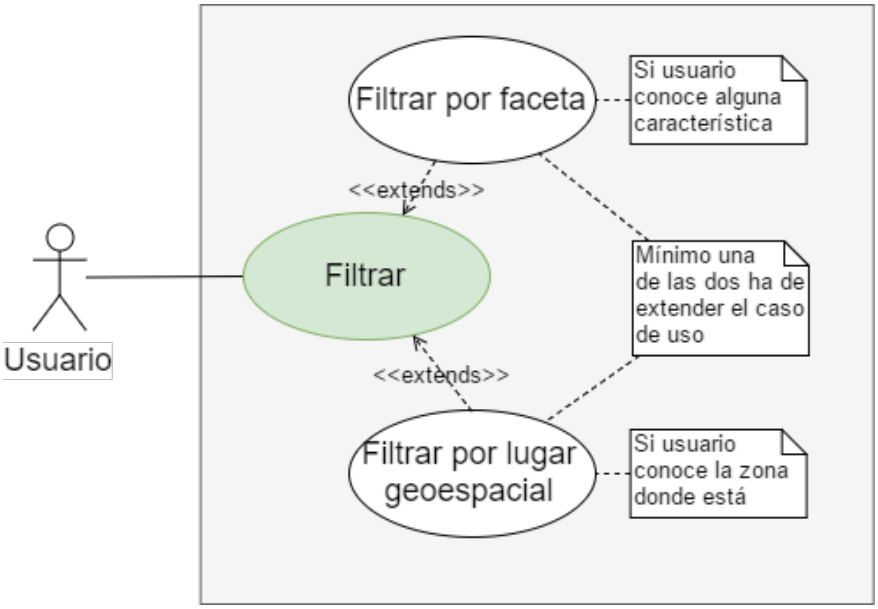


Figura 19. Casos de uso bajo Filtrar

Finalmente, al margen de la vista (Figura 16) tenemos los casos de uso del administrador (Figura 20). Son escuetos debido a que este proyecto se ha enfocado hacia la perspectiva del usuario que consume la aplicación y hacia quien la gestiona.

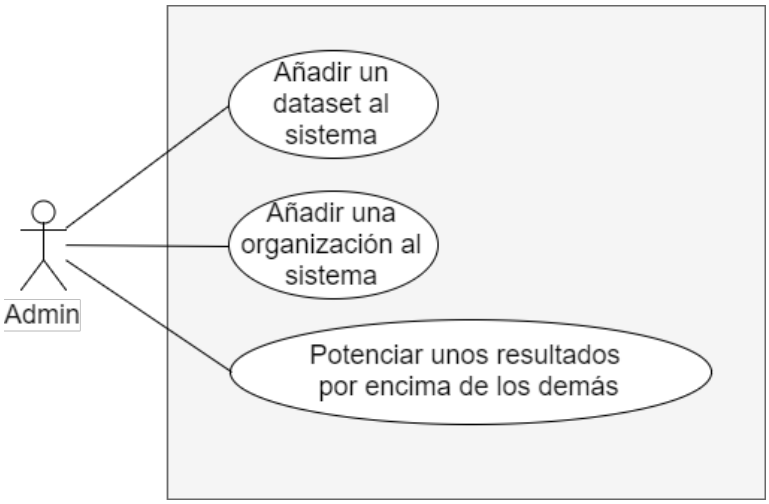
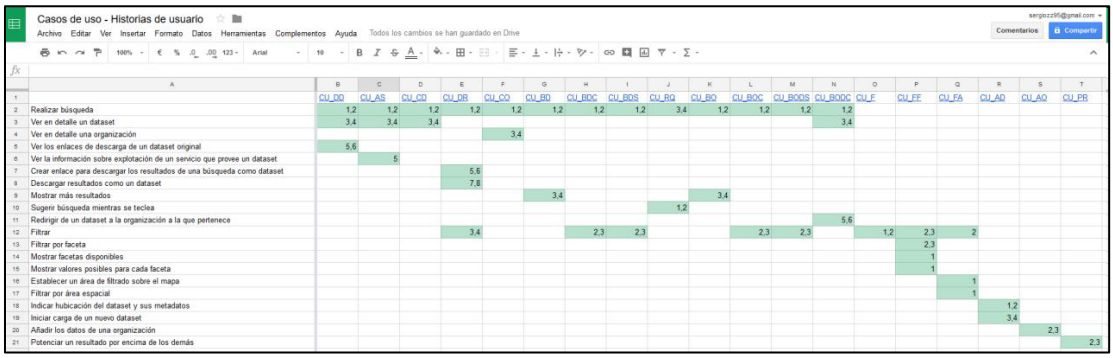


Figura 20. Casos de uso del Administrador

2.2. Historias de usuario

El formato escogido para esto es el de Historias de Usuario debido a las ventajas que ofrecen en flexibilidad, así como la posibilidad de posponer decisiones concretas. Algo que en un sistema de requisitos tradicional no sería posible ya que tienen su origen en las planificaciones en cascada donde se cierran todos los detalles al inicio del proyecto.

Para esta labor se ha creado una tabla (Figura 21) donde se alinean en columnas los casos de uso y se redactan en las filas las funcionalidades necesarias para ese caso. Las celdas con valor representan el número del paso o pasos en los que esa funcionalidad tiene lugar. Esto permite seguir una traza de las funcionalidades hacia los escenarios que motivaron su creación.



	CU_DD	CU_AS	CU_CD	CU_DR	CU_CO	CU_BD	CU_BDC	CU_BDS	CU_RQ	CU_BO	CU_BOC	CU_BODS	CU_BODC	CU_F	CU_FF	CU_FA	CU_AD	CU_AO	CU_PR
Realizar búsqueda	1,2	1,2	1,2	1,2	1,2	1,2	1,2	1,2	3,4	1,2	1,2	1,2	1,2						
Ver en detalle un dataset	3,4	3,4	3,4										3,4						
Ver en detalle una organización					3,4														
Ver los enlaces de descarga de un dataset original	5,6																		
Ver la información sobre explotación de un servicio que provee un dataset		5																	
Crear enlace para descargar los resultados de una búsqueda como dataset				5,6															
Descargar resultados como un dataset				7,8															
Mostrar más resultados						3,4				3,4									
Sugerir búsqueda mientras se teclea									1,2										
Redirigir de un dataset a la organización a la que pertenece													5,6						
Filtrar				3,4			2,3	2,3			2,3	2,3		1,2	2,3	2			
Filtrar por faceta															2,3				
Mostrar facetas disponibles															1				
Mostrar valores posibles para cada faceta															1				
Establecer un área de filtrado sobre el mapa																1			
Filtrar por área espacial																1			
Indicar ubicación del dataset y sus metadatos																	1,2		
Iniciar carga de un nuevo dataset																	3,4		
Añadir los datos de una organización																		2,3	
Potenciar un resultado por encima de los demás																			2,3

Figura 21. Tabla de Casos de Uso - Historias de Usuario

Las historias de usuario resultantes de este proceso pueden consultarse en el Anexo D.

2.3. Priorización

Técnica empleada. El método MosSCoW, una metodología desarrollada por Dai Clegg y Richard Barker (Barker & Clegg, 2004) que propone que las funcionalidades de una aplicación pueden ser categorizadas en cuatro sencillas prioridades (*must have*, *should have*, *could have*, y *won't have*), abandonando así sistemas de puntuaciones más complejos y dotando además de una semántica clara a cada categoría (Wikipedia Contributors, 2018).

Must have (es necesario)

- Realizar una búsqueda textual difusa.
- Ver la cobertura de los resultados que sean colecciones de datos sobre un mapa.
- Desactivar todos los filtros de una vez.
- Filtrar los resultados por sus facetas.
- Ver las facetas disponibles.
- Ver los posibles valores para cada faceta.
- Filtrar por área espacial.
- Indicar la ubicación de un dataset y sus metadatos al sistema.
- Iniciar carga de un nuevo dataset.

Should have (es recomendable)

- Ver los objetos espaciales de una colección de datos sobre un mapa.
- Consultar toda la información de una organización.
- Ver más resultados en una búsqueda.
- Desactivar un filtro.
- Ser redirigido a un sitio externo para descargar la colección de datos original en un formato específico cuando ésta no puede ser descargada desde el sistema.
- Encontrar la organización que provee una colección de datos que he encontrado en una búsqueda, accediendo a ella desde la información de la colección.
- Añadir los datos de una organización.

Could have (podría implementarse)

- Ver todos los filtros activos agrupados en un lugar.
- Ver cuántos resultados disponibles hay para cada valor de cada faceta.
- Ver una descripción del servicio remoto que provee una determinada colección de datos.
- Recibir sugerencias de búsqueda mientras tecleo.
- Establecer un área compleja de filtrado sobre el mapa.
- Potenciar un resultado por encima de los demás.

Won't have (no se implementará)

- Descargar la colección de datos original en uno de los formatos específicos en los que fue originalmente distribuido.
- Crear una URI parametrizada que me permita descargar los resultados de una búsqueda como si fuera una colección de datos y en el formato que yo seleccione.
- Descargar los resultados de una búsqueda como si fuera una colección de datos y en el formato que yo seleccione.
- Recibir un informe detallado si un proceso del sistema falla indicando el motivo del fallo.

2.4. Análisis de viabilidad: tecnologías y alternativas

Teniendo clara la idea y habiendo extraído las funcionalidades necesarias, necesitamos saber si existe una tecnología que nos permita conseguir lo que queremos, ofrecer búsquedas espaciales sobre datos y no sobre el resumen de sus colecciones.

Para ello echaremos la vista a uno de los paradigmas que más ha crecido en los últimos años, el *big data*. Su capacidad para tratar enormes volúmenes de datos de una forma rápida y distribuida lo hace perfecto para nuestras necesidades. Estudiaremos por ello las alternativas más comunes:

2.4.1. Elastic Stack

El stack de Elastic es un conjunto de aplicaciones que trabajan en torno a Elasticsearch, dedicadas a buscar y obtener información de enormes volúmenes de datos (Figura 22). Está compuesto por:

- **Beats**: Es una aplicación ligera que monitorea y envía *logfiles* en tiempo real a Logstash o Elasticsearch. Nace debido a la incomodidad de instalar y configurar Logstash en cada una de las máquinas que generan *logfiles* así como no afectar a su rendimiento.
- **Logstash**: Puede ser entendido como un filtro o un preprocesador de la información que llegar a Elasticsearch. Permite programar transformaciones de datos.
- **Elasticsearch**: El centro neurálgico del *stack*. En él reside el verdadero potencial de la plataforma, que es su capacidad de búsqueda, análisis y aprendizaje. La inyección de datos se hace a través de su API HTTP o mediante su cliente java nativo. Tiene un lenguaje de *query* extremadamente potente y permite hacer agregaciones de datos de forma rápida y distribuida.
- **Kibana**: Una aplicación web puramente *front-end* que permite explotar todas las capacidades de Elasticsearch a través de una interfaz amigable. Cabe puntualizar que sólo permite consumir información, la inyección corre a cargo de las capas inferiores.
- **X-Pack**: Es un software de seguridad y monitoreo interesante para producción, pero que no aporta nada en una fase de desarrollo. Por ello no va a ser usado en este proyecto.

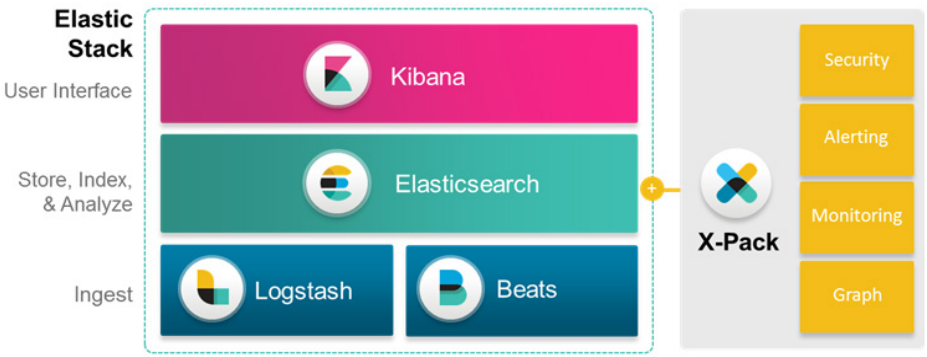


Figura 22. El stack tecnológico de Elasticsearch

2.4.2. Hadoop Framework

De las manos de Apache tenemos el *framework* de Hadoop, uno de los principales responsables de la popularización del *big data*. Sus aplicaciones trabajan en torno al motor MapReduce, un software especializado en trabajos de map/reduce en paralelo y distribuidos (Figura 23). Estas aplicaciones son:

- **HDFS**: Si trabajas grandes cantidades de datos necesitas una forma de almacenarlos. *Hadoop Distributed File System* se encarga precisamente de eso. De ofrecer un sistema de ficheros distribuido de forma transparente al usuario.
- **HBase**: Es una base de datos no relacional de tipo clave-valor que se asienta sobre HDFS y ofrece funcionalidades limitadas, pero de baja latencia. Sirven como entrada o salida de trabajos de Hadoop.
- **HCatalog**: Es un software de gestión de Tablas y esquemas. Permite abstraer ciertos trabajos de map/reduce. No se ha profundizado en su uso ya que no era necesario para el proyecto.
- **MapReduce**: El motor principal del *framework*. Un sistema de *job tracker* y *workers* que ejecutan trabajos en paralelo de forma distribuida.
- **Hive**: Se trata de una base de datos relacional. A diferencia de HBase, no trabaja sobre HDFS sino sobre rutinas map/reduce de Hadoop. Interpreta las *queries* en HQL (pseudo SQL) y crea las rutinas necesarias para responder con los resultados adecuados. Al tratarse de trabajos en lote sobre grandes cantidades de ficheros los resultados pueden tardar minutos o incluso horas. Una de las ventajas es que permite integración con JDBC lo que la hace extremadamente interoperable.
- **Pig**: Es el motor que ejecuta el lenguaje de alto nivel Pig Latin, que sirve para programar tareas map/reduce en un lenguaje más amigable.

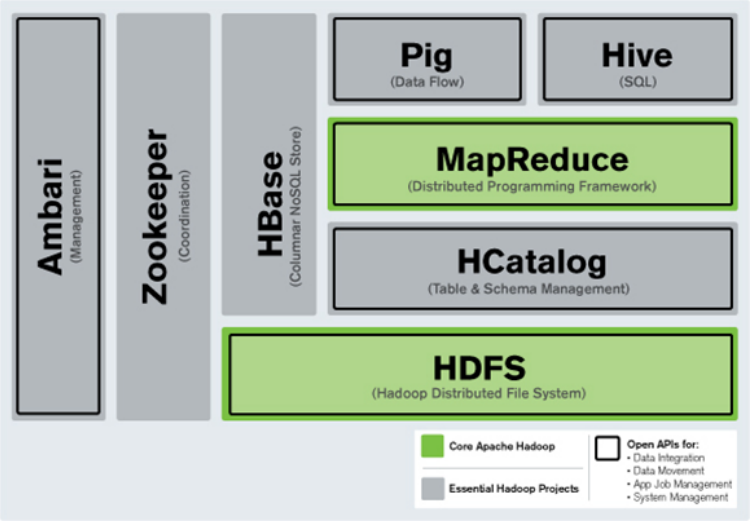


Figura 23. El stack tecnológico de Apache Hadoop

3. Diseño

3.1. Modelo de información

El primer punto a la hora de diseñar el sistema es definir el modelo de la información. Para ello nos basaremos en el conocimiento extraído del análisis de los otros portales. En ellos hemos identificado los distintos tipos de entidades que participan en el ecosistema GIS (Figura 24). El resultado de este análisis es el que se ha podido ver en la sección del Glosario que introducía este trabajo.

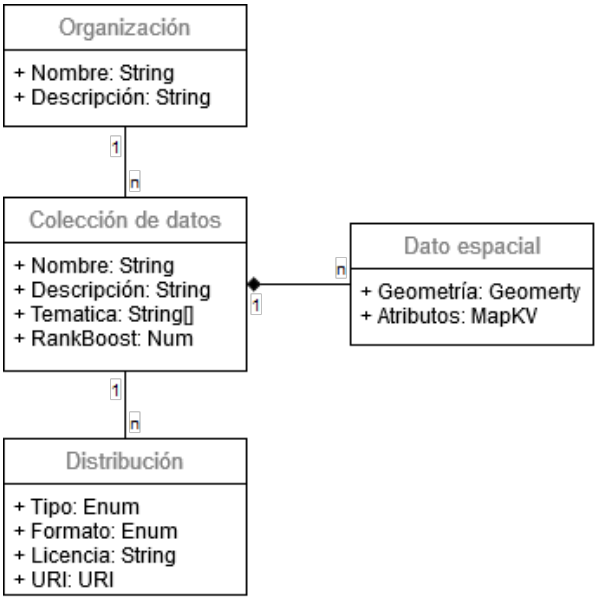


Figura 24. Modelo de información

Habiendo definido la estructura de datos básica, lo siguiente es especificar qué información compone cada una de las entidades. Para ello podríamos inspirarnos en las interfaces del OGC, pero se ha preferido mantener la complejidad al mínimo para simplificar las pruebas de concepto.

Cabe avisar de que estas representaciones, en un nivel de diseño, prescinden de los identificadores y los campos que las relacionan.

3.1.1. Organizaciones

Como entidades responsables de las colecciones de datos, las organizaciones cuentan con la información básica de una corporación.

- **Nombre**: El nombre a mostrar de la organización.
- **Descripción**: Descripción dada por la propia organización directa o indirectamente.

3.1.2. Colecciones de datos espaciales

Son los conjuntos de datos agrupados bajo un criterio. Habitualmente, por no decir siempre, los datos que contienen comparten los mismos tipos de atributos, aunque difieran en sus valores. Esto se deriva de su tradicional estructuración en tablas.

- **Nombre**: El nombre a mostrar de la colección de datos.
- **Descripción**: Descripción dada por el proveedor de la colección de datos.
- **Temática/etiquetas**: Conjunto de palabras clave, acotadas o no, que orientan la búsqueda de la colección.

3.1.3. Datos espaciales

Es la unidad mínima de información espacial. Representa un objeto con una geometría y un lugar en el espacio concretos. Puede contener información adicional en forma de atributos.

- **Geometría**: Define tanto su forma como su posición en el espacio
- **Atributos**: Conjunto de atributos y sus valores

3.1.4. Distribuciones

Es la implementación digital y real de los datos lógicos:

- **Tipo**: Define si se trata de un fichero o un servicio web.
- **Formato**: Define el formato o formatos en el que está disponible ese fichero para esa distribución.
- **Licencia**: Especifica los derechos que se tienen sobre ese recurso.
- **URI**: Es la dirección de descarga o explotación original de esa distribución del recurso.

4. Pruebas de concepto: El *sandbox* como estrategia de investigación

Enfrentarse a unas tecnologías desconocidas en un dominio desconocido requiere de numerosas pruebas de concepto y prototipos rápidos en los que familiarizarse con los pormenores de cada tecnología. Por ello el proceso de desarrollo ha sido una constante iteración de prototipos basados en distintas propuestas de arquitecturas y tecnologías.

4.1. Hadoop, Hive y Elasticsearch

La primera idea fue utilizar el *framework* de Hadoop para almacenar y exponer los datos originales y Elaticsearch para crear los índices de búsqueda en base a esos datos. De ese modo:

- **HDFS** almacena los datos de forma distribuida
- **Hive** mapea los datos almacenados en HDFS a vistas de tablas relacionales y uso del plugin elasticsearch-hadoop.jar se comunica con Elasticsearch y le inyecta los datos
- **Elasticsearch** recibe los datos, los indexa y los expone para su búsqueda

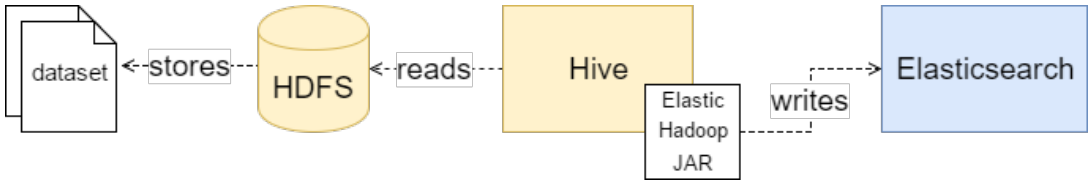


Figura 25. Esquema de la idea detrás del primer prototipo

Tras desarrollar esta primera aproximación se detecta un gran inconveniente: La necesidad de actualizar manualmente las tablas relacionales para insertar los nuevos elementos. No es un mecanismo sencillo y no parece ofrecer ventajas más allá de un sistema de persistencia distribuido con HDFS. No obstante, todavía no se va a desechar la opción. En el siguiente prototipo se explorará una alternativa a este sistema.

Los detalles de la implementación están en el Anexo E.1.

4.2. Hadoop, Hive y Elasticsearch con JDBC

En base al prototipo anterior y sus conclusiones, se busca trasladar la responsabilidad de la actualización de los datos a Elasticsearch con la esperanza de encontrar alguna ventaja. Ésta prueba es breve ya que sólo implica cambiar la comunicación entre Hive y Elasticsearch.

La alternativa escogida es explotar la base de datos de Hive a través del protocolo JDBC desde el lado de Elasticsearch. Elasticsearch por sí mismo no soporta esta función ya que es un elemento pasivo a la espera de que le lleguen los datos. Por esto se introduce Logstash en la cadena. Como preprocesador y filtro tiene la opción de leer los datos vía JDBC, procesarlos e insertarlos en Elasticsearch.

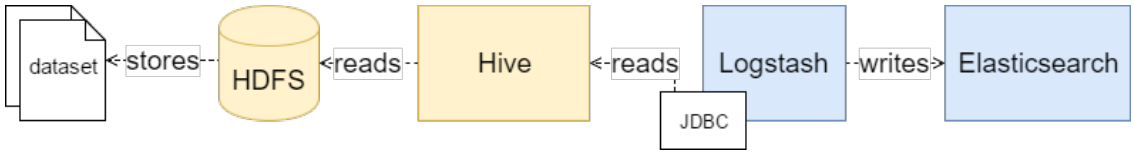


Figura 26. Esquema de la idea detrás del segundo prototipo

Un segundo intento ha sido más que suficiente para descubrir que Hive no se adapta a las necesidades del problema. No permite una actualización en caliente de los datos, sino que fuerza un proceso consulta síncrona nada conveniente. Además, la comunicación entre componentes no es todo lo integrada que uno desearía y se tiene la sensación de no estar usando las herramientas para lo que fueron diseñadas.

Se decide entonces prescindir del Hadoop Framework y buscar una solución más integrada, que permita inyección de datos en caliente conforme éstos van siendo generados.

Los detalles de la implementación están en el Anexo E.2.

4.3. Elastic Stack

Una de las mayores ventajas de Elastic es las facilidades que ofrece a la hora de desplegar sus sistemas por primera vez. Desde su repositorio de imágenes de Docker propio puedes tener un clúster con el stack entero funcionando en cuestión de minutos. Si la solución pasa por ceñirnos a un sólo *stack*, el de Elastic hace tentador adoptar todos sus elementos.

La idea detrás de haber abortado el prototipo anterior era que se buscaba obtener un flujo de datos similar de forma fluida y no con actualizaciones en bloque. El *stack* de Elastic está destinado precisamente a eso. Sin reinventar nada, sin forzar la comunicación de componentes que no fueron pensados para ello. Tan sólo desplegando y configurando cada parte de la cadena:

- **Beats** monitorea un directorio o un fichero a la espera de ver cambios en él. Si se añade un fichero nuevo, procede a su lectura hasta el final del documento y mantiene un puntero de lectura al final de este. Si el cambio viene por una modificación del fichero, asume que ha sido una adición de contenido y procede a continuar su lectura hasta el nuevo final del documento. Esta asunción viene de que su rol principal es monitorear logfiles. Cada línea leída es enviada a Logstash como un mensaje diferente.

- **Logstash** recibe los datos enviados desde Beats, los procesa según le indique la receta y los envía a Elasticsearch.

- **Elasticsearch** recibe los datos, los indexa y los expone para su búsqueda.

- **Kibana** ofrece un panel de control que permite crear todo tipo de visualizaciones de los datos. Trabaja contra la API de búsqueda de Elasticsearch.

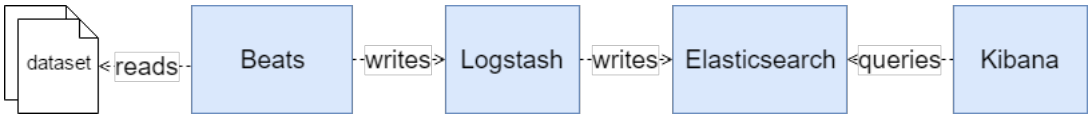


Figura 27. Esquema de la idea detrás del tercer prototipo

Por primera vez podemos visualizar los datos y trabajar con ellos desde una interfaz web. Kibana demuestra perfectamente el potencial de esta idea. No obstante, está muy lejos de ser la solución que necesitamos. No soporta visualización de geometrías ya que está centrado en la agregación de datos en vistas como mapas de calor que indiquen la densidad de los puntos (Figura 47).

Aunque este prototipo se considera el arranque real debido a los grandes progresos que reporta, es necesario reemplazar tres componentes.

- El primero, ya mencionado, es Kibana que, como era de esperar, no ofrece el soporte para búsquedas que necesitamos ni las visualizaciones o la flexibilidad. Por ello se deberá encontrar y adaptar o desarrollar una aplicación web que sí ofrezca esas funcionalidades.
- Los otros dos son Beats y Logstash que, si bien han cumplido su función, no permiten un fácil control de la lógica y las transformaciones. Esto hace que los cambios estén limitados a lo que el scripting de la herramienta permita, lo cual es inadmisible en un entorno de investigación como éste.

Los detalles de la implementación están en el Anexo E.3.

4.4. Spring y Elasticsearch con Elastic Java API

Para este prototipo es irrelevante el front-end ya que su objetivo es el desarrollo de un programa propio que, por la línea de comandos, lea los ficheros originales, los procese e inserte sus datos en Elasticsearch. Esto permitirá realizar transformaciones complejas, así como tener un control de la lógica de negocio. La mejor elección es usar el *framework* de Spring por dos motivos: el primero es la comodidad de desarrollar en un lenguaje conocido como es Java y en un *framework* conocido como es Spring, el cual ofrece inyección de dependencias y otra serie de funcionalidades muy interesantes además de ser un producto pensado para ser competente en entornos de producción; el segundo es que, al estar Elasticsearch desarrollado enteramente en Java, su librería/API tienen una integración perfecta ya que son las que ellos mismos usan internamente. De hecho, la propia librería permite ejecutar un nodo de Elasticsearch dentro del programa que la usa.

La otra decisión para este prototipo es prescindir de Docker ya que con este despliegue comienza a ofrecer más impedimentos que facilidades.

Figura 28. Esquema de la idea detrás del cuarto prototipo

Si bien la Elastic Java API y su documentación son relativamente complejas, se considera un buen punto de partida, a falta de otras librerías o abstracciones que ofrezcan funciones de más alto nivel.

Los resultados de este prototipo se valoran altamente positivos. Se ha logrado con sencillez crear un sistema robusto, en un lenguaje conocido y sin sacrificar las funcionalidades que ya se tenían. Además, abre la puerta a introducir todo tipo de lógica y procesamientos complejos.

No obstante, tiene un problema de prestaciones ya que las inserciones se hacen por elemento. Existe una API dentro de la librería para inserciones en lote, pero no se ha profundizado en ella al haber encontrado antes una alternativa que será explorada en el siguiente prototipo.

Los detalles de la implementación están en el Anexo E.4.

4.5. Spring y Elasticsearch con Spring Data

El prototipo anterior era ideal salvo por dos inconvenientes:

- El primero de ellos era la Elasticsearch Java API de bajo nivel, que exigía largas búsquedas en la documentación interna para ofrecer las funcionalidades más sencillas
- El segundo era la lentitud en la inserción de los datos

Existe una solución que solventa ambos y además utiliza una tecnología ya conocida. Se trata de una librería creada por la comunidad de Spring y soportada oficialmente por Pivotal que ofrece comunicación con Elasticsearch para el framework Spring Data.

Spring Data es un framework de mapeo objeto-relacional que permite abstraer la lógica de persistencia de su implementación lo cual la hace a la aplicación *database agnostic*.

De ese modo *Spring Data Elasticsearch* ofrece una abstracción de muy alto nivel del repositorio de datos ofreciendo una interfaz con sencillas operaciones de inserción y búsqueda que puede ser extendida con funciones nuevas.

Figura 29. Esquema de la idea detrás del quinto prototipo

Ahora sí tenemos un sistema de inserción solvente, con posibilidad de expandirlo a otros formatos de entrada; con posibilidad de añadirle *fases* en la cadena de procesado; y con un rendimiento excelente pudiendo procesar e insertar volúmenes del orden de 100.000 datos en cuestión de un minuto.

Los detalles de la implementación están en el Anexo E.5.

4.6. Spring, Elasticsearch y Searchkit

Teniendo un sistema de inserción rápido y robusto, toca centrarse en una aplicación web a la altura. Para ello, tras una larga búsqueda de alternativas e ideas se encuentra un proyecto que tal vez sea de gran ayuda. Su nombre es Searchkit y ofrece un módulo para React.js con los componentes necesarios para montar un portal de búsqueda que trabaje directamente contra la API HTTP de Elasticsearch.

Como viene siendo habitual en este proyecto, React.js es otra de esas tecnologías con las que nunca se ha trabajado. Por ello este prototipo está enfocado en familiarizarse con dicho *framework* y averiguar de qué es posible el proyecto Searchkit.

Se sabe de antemano que la aplicación no posee soporte de mapa de modo que se deberá implementar una solución provisional. Para ello habrá que estudiar el comportamiento de los componentes de Searchkit y tratar de replicarlos de la forma adecuada para crear un componente propio.

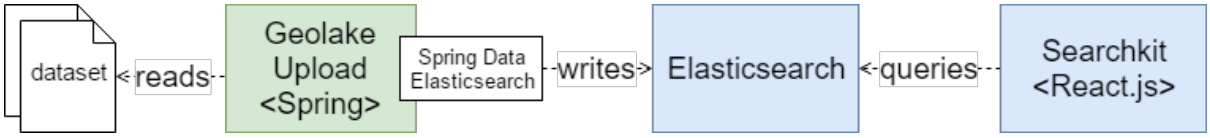


Figura 30.Esquema de la idea detrás del sexto prototipo

La herramienta parece ser todo lo que necesitamos para desarrollar una prueba de concepto. Ofrece unos tiempos de respuesta del orden de 100 ms y muestra los resultados con rapidez. El lado negativo es que el uso del framework, si bien demuestra la viabilidad de la idea, limita su personalización y adaptación a otras lógicas que se salgan del mismo. No obstante, como espacio de trabajo y prototipado es perfecto.

El siguiente paso es diseñar una interfaz desde cero con los requerimientos de nuestra aplicación como punto de partida y posteriormente incluir y adaptar los componentes de Searchkit. Además, el componente de mapa creado deberá refinarse y pulir su comportamiento.

Los detalles de la implementación están en el Anexo E.6.

5. Prototipo final: Geolake Search

El séptimo y último prototipo. Una revisión de todo lo aprendido. Una actualización de modelos, arquitecturas, ideas y soluciones. Una reimplementación integral de la aplicación.

El principio fundamental de un prototipo es que sirve para probar algo y que, una vez probado, se ha de destruir por completo. Los prototipos previos han aportado cada uno su perspectiva a un mismo problema. Ahora es el momento de tomar todo ello e implementar una aplicación que demuestre todos los principios.

5.1. Arquitectura

Si podemos entender la aplicación como la coordinación de dos acciones: la carga y la búsqueda, entonces podemos analizar por separado los componentes arquitectónicos que componen cada una. Por la parte de la carga, la arquitectura sigue un patrón tradicional de *batch processing* con la inclusión de un módulo que le permite la inserción en *Elasticsearch*. En cambio, en la parte de la búsqueda, la arquitectura sigue un patrón *cliente/servidor* con la particularidad de que el servidor no es un desarrollo propio, sino que es una herramienta completa como lo es *Elasticsearch*.

5.1.1. Vista de módulos

Lo primero que destaca de la aplicación de inserción ha sido diseñada en una arquitectura en tres capas:

- La capa de infraestructura es la que comunica la aplicación con el exterior: sistemas de ficheros, puertos HTTP, conexiones con bases de datos, etc.
- La capa de aplicación implementa casos de uso o funcionalidades atómicas. Permite controlar acciones transaccionales y orquestar los elementos de la capa de modelo. En este caso sólo cuenta con una clase ya que sólo se ha definido una transformación: de CSV a Elasticsearch y finalmente a un fichero JSON para revisar los resultados. Otras transformaciones podrían ser *GeoJSONtoES* para procesar la entrada en formato GeoJSON o *SHPtoES* para procesar la entrada en formato ESRI Shapefile.
- La capa de modelo implementa y representa la lógica de negocio en forma de objetos. En este caso, el flujo de control forma parte del modelo ya que es parte intrínseca de la razón de ser de la aplicación, las transformaciones en cadena.

Aquellos que estén familiarizados con las arquitecturas en tres capas, les llamará la atención ver la capa de infraestructura encima de la capa de aplicación. Esto es un principio de la arquitectura hexagonal que permite que modelo y aplicación no dependan de la infraestructura y por ende sean más reutilizables. Para lograrlo hay que aplicar inversión de dependencias.

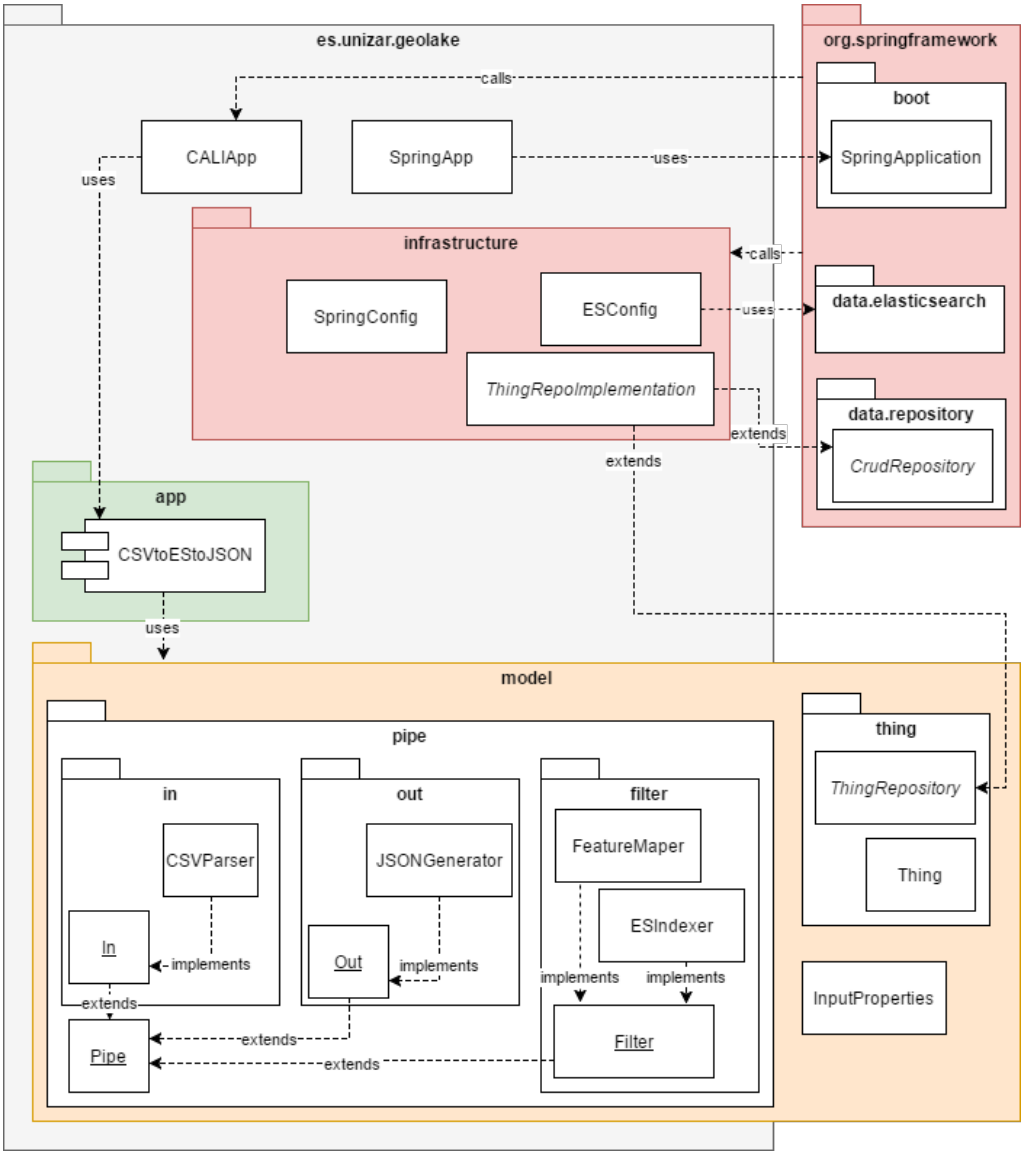


Figura 31. Vista de módulos (back-end)

En la Figura 31 podemos apreciar el ordenamiento de capas mencionado anteriormente, el cual ha sido destacado en tres colores para hacerlo aún más legible. Cabe destacar que *CLIApp* y *SpringApp* pertenecen a la capa de arquitectura, pero Spring Framework exige que dichas clases estén en la raíz del proyecto y no en un subpaquete.

Se puede comprobar así que se han respetado las capas al no haber dependencias en sentido ascendiente, sólo de arriba a abajo o dentro de la misma capa.

Existe un elemento que exige una explicación más detalla, y éste es la del trinomio *ThingRepository*, *ThingRepoImplementation*, *CrudRepository*. Las cursivas representan interfaces y en este caso estamos viendo la forma elegante de mantener inversión de dependencias cuando se usa la función de Spring Data que autogenera los repositorios en base a las interfaces definidas por el usuario. Por ello *ThingRepository* existe para que las clases de su capa la usen y *ThingRepoImplementation* existe para extender *CrudRepository* y que el framework Spring Data detecte la interfaz y genere el repositorio adecuado. En este caso, uno de Elasticsearch, configurado en la clase *ESConfig*.

Por otro lado, tenemos en la capa de modelo cuatro clases abstractas que definen el esquema general del flujo de control, permitiendo a las clases que las completen definir cómodamente la funcionalidad sin preocuparse de implementar flujo de control alguno:

- **Pipe**: como clase general fija aspectos comunes a todas las etapas. De ella heredan las tres siguientes.
- **In**: como clase de entrada al flujo, define su función de inicio con un *InputStream* para la entrada y una *BlockingQueue* para la salida.
- **FIlter**: como clase de procesado intermedio, define su función de inicio con una *BlockingQueue* tanto para la entrada como para la salida.
- **Out**: como clase de entrada al flujo, define su función de inicio con una *BlockingQueue* para la entrada y un *OutputStream* para la salida.

De este modo, sus implementaciones tan sólo tienen que implementar las funciones abstractas que son llamadas por la propia clase abstracta. Es una forma de delegar el control y el paso de la información a un solo elemento, la clase abstracta, en lugar de reimplementarlo en cada etapa.

En el lado del *front-end* (Figura 32) en cambio, se ha seguido una arquitectura de dos capas, pero sin la inversión de dependencias con la arquitectura. Esto es debido a que no existen soluciones populares para la inyección de dependencias y tampoco es un tema crucial debido al escaso tamaño del código y a las limitaciones impuestas por el módulo Searchkit. Por ello el código propio dependerá y conocerá los detalles de las aplicaciones externas subyacentes.

Si bien en 2017 ya se empieza a hablar de *arquitecturas de capas* en el *front-end*, es un tema aún muy inmaduro y poco documentado que requiere conocer frameworks que permitan el uso de clases y realizar una correcta división de responsabilidades. El, framework usado, React.js, permitiría esto y se consideró interesante tratar de acercarse a ese planteamiento lo cual ha resultado en un código más ordenado. Sin embargo, el caso ideal sería extraer la lógica de la capa de la vista e implementarla en una capa intermedia, en *JavaScript vanilla*, ajena a los componentes de React de modo que el *framework* fuera usado solamente para el aspecto visual.

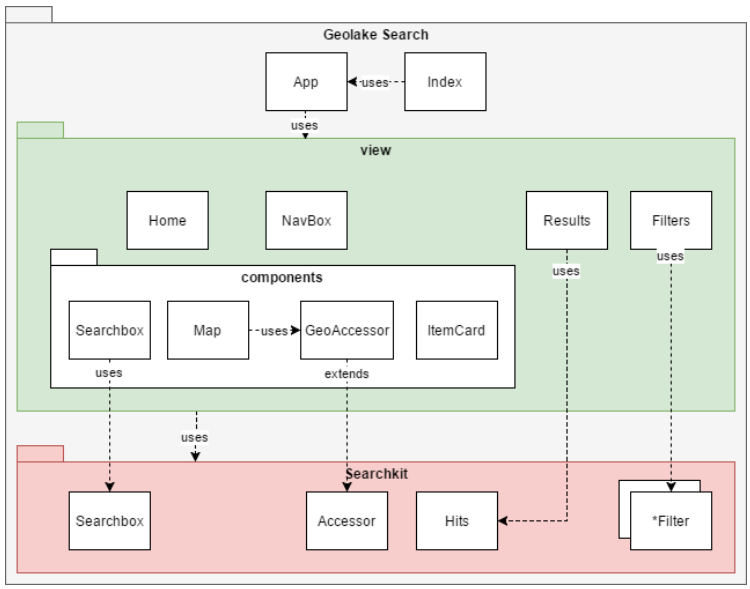


Figura 32. Vista de módulos (front-end)

5.1.2. Vista de componentes y conectores

Como se aprecia en la vista resumen de *componentes y conectores* (Figura 33), principalmente tenemos los 3 componentes que ya conocemos (Geolake, Elasticsearch y Geolake Search)

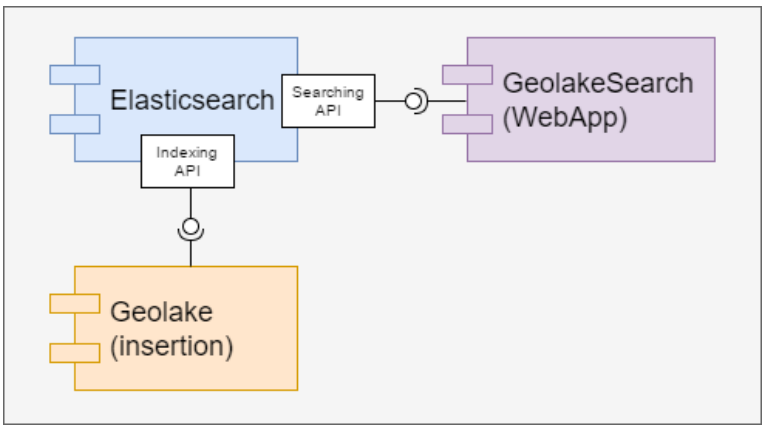


Figura 33. Vista de componentes y conectores

Lo interesante de esta vista, como en casi todas las vistas de *componentes y conectores*, son sus interfaces, ambas de Elasticsearch ya que, como se ha mencionado anteriormente, se usa como elemento pasivo que ofrece su API, pero no inicia comunicación alguna. La API se puede entender como una sola, pero resulta más interesante si la entendemos como dos distintas con responsabilidades distintas.

Por un lado, la API de indexado (Elastic, 2018) que es la que ofrece los *endpoints* para insertar datos. Cabe mencionar que, en el caso de la inserción, al estar usándose la API de Java nativa tras la librería Spring Data, no se está accediendo a la API HTTP sino a la API programática la cual tiene una relación de equivalente con la API HTTP por lo que no profundizaremos en ello

Por otro lado, la API de búsqueda (Elastic, 2018), la cual tiene responsabilidades como búsqueda y agregación posee un rico *Query Domain Specific Languaje* que permite refinar las búsquedas con complejas lógicas, así como controlar los *scoring* de los resultados.

Si nos vamos a la vista específica de la aplicación de inserción de datos encontraremos unas relaciones mucho más interesantes (Figura 34).

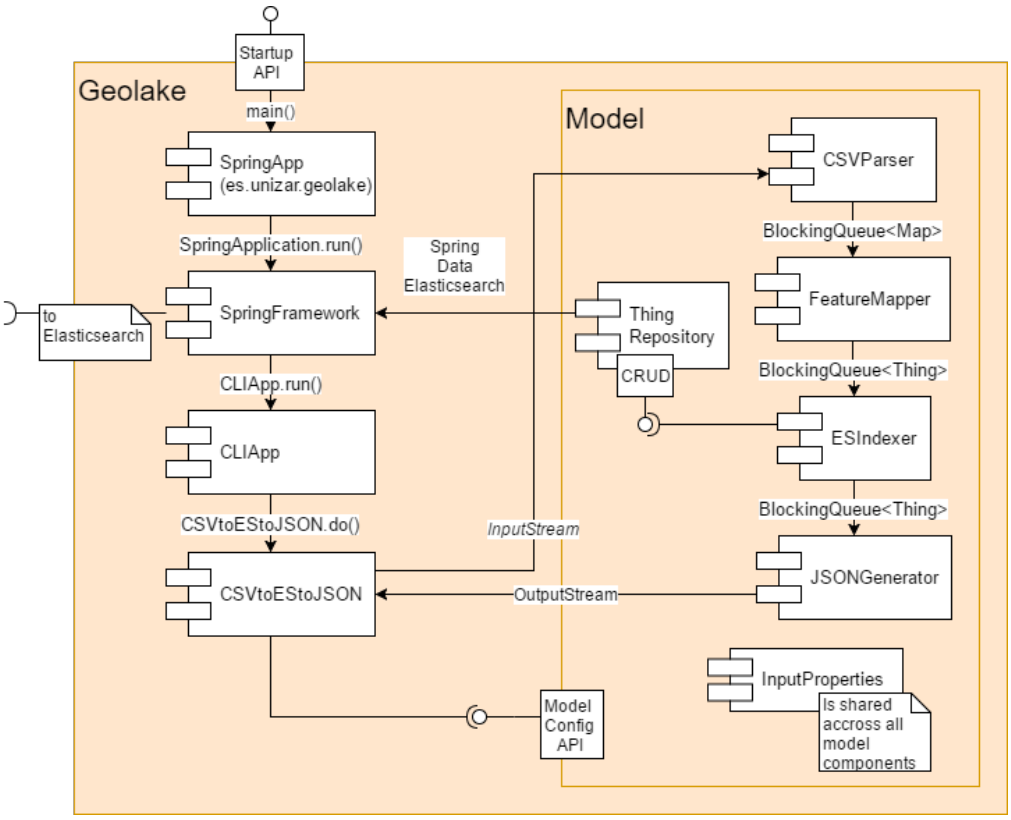


Figura 34. Vista específica de la Geolake

Si observamos la columna de componentes alineados a la izquierda representa la cadena de llamadas que pone en marcha el proceso. Son conexiones sencillas y sin parámetros interesantes hasta que llegamos a *CSVtoEStoJSON.do()*. En este punto es cuando se pasan 3 elementos: un InputStream, un OutputStream y un InputProperties

Ésta es la forma que tiene la aplicación de pasar a CSVtoEStoJSON todos los parámetros necesarios para ejecutar su proceso manteniendo el agnosticismo de la capa arquitectural. De ese modo, independientemente de si el *stream* viene de un sistema de ficheros o de un canal TCP, si el formato es el adecuado, la implementación es reutilizable.

Como clase orquestadora del modelo, utiliza la API de configuración de este, que básicamente permite asignarles los canales, en este caso *Blocking Queue*, e iniciar el proceso.

Finalmente está, el filtro interesante, ESIndexer, que utiliza la API de ThingRepository para operar indirectamente contra Elasticsearch a través de Spring Data Elasticsearch.

5.1.3. Vista de despliegue

En la vista de despliegue se muestran los elementos desplegados y las comunicaciones entre ellos (Figura 35).

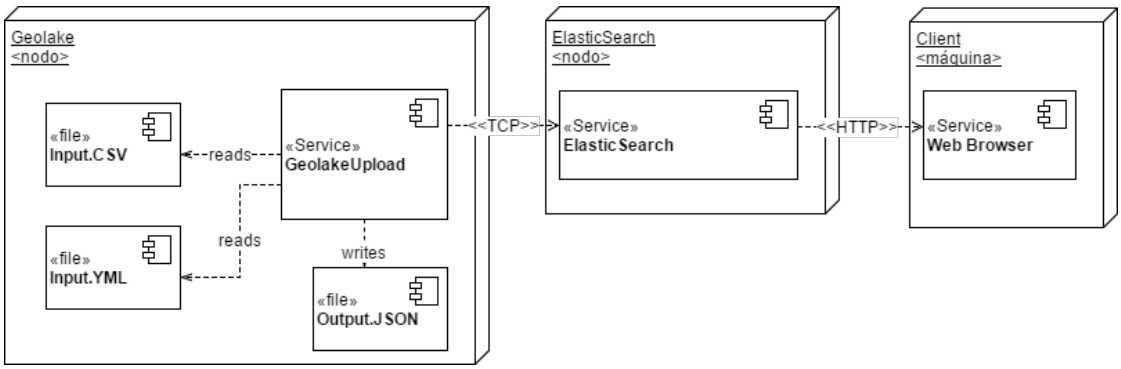


Figura 35. Vista de despliegue

Si bien es cierto que sería posible incluir *GeolakeUpload* y *Elasticsearch* en un mismo nodo, se ha optado por mostrar la representación más desacoplada y escalable. Cabe destacar que los nodos y máquinas del diagrama son replicables de modo que la capacidad de carga del sistema se puede adaptar escalando horizontalmente. Esto se traduciría en:

- Varios nodos con *GeolakeUpload* comunicándose mediante la *Elasticsearch Java API* con el clúster de *Elasticsearch*.
- Varios nodos de *Elasticsearch* en clúster recogiendo la información y autocoordinándose para ofrecer respuestas de baja latencia.
- Varios clientes con *Geolake Search* lanzando preguntas contra el clúster de *Elasticsearch*. Este punto es evidente debido a que en todo sistema cliente servidor se han de suponer varios clientes concurrentes.

5.2. Tecnologías

Al inicio de este proyecto se analizaron diversas tecnologías para poner en práctica la idea. Tras los seis prototipos previos, esta selección ha cambiado notablemente, prescindiendo de algunas de las iniciales para dejar paso a otras más interesantes. Para analizarlas las anidaremos en las tecnologías de mayor rango.

5.2.1. Elasticsearch

Para unos, un potente motor de búsqueda; para otros, una base de datos documental y distribuida con grandes herramientas de recuperación. Elasticsearch es una herramienta que ha cambiado la forma en la que entendemos el Big Data. La capacidad de almacenar inmensas cantidades de datos en índices invertidos y ofrecer respuestas a complejas búsquedas ordenadas por relevancia es sólo la punta del iceberg de todo lo que puede hacer.

Actualmente, los desarrolladores se están interesando por sus capacidades de agregación ya que permiten obtener información que antes era impensable de una forma cómoda, muy lejos de lo que suponía programar rutinas *map/reduce*.

En el caso de nuestro problema, la capacidad de agregación es imprescindible, pero todavía es más importante el filtrado espacial. Haciendo uso del algoritmo de *geohash*, que codifica una localización en un hash de precisión variable, Elasticsearch consigue construir un árbol espacial en el que indexar las localizaciones de sus elementos. De ese modo, consigue hacer filtrados espaciales a gran velocidad independientemente de la forma de la geometría de filtrado o de la geometría de los resultados.

5.2.2. Spring Framework

El framework Spring es el elegido para una gran cantidad desarrollos en el ámbito profesional debido a su potencia y estabilidad. Respaldado por su cuidada arquitectura, Spring permite extender sus funcionalidades mediante los componentes que el desarrollador desee y de ese modo empaquetar una solución *production ready* de forma sencilla.

Dentro de este framework, se harán uso principalmente de dos tecnologías ya mencionadas:

- La tecnología de inyección de dependencias Spring Beans, la cual forma parte del núcleo del *framework*. Nos permitirá inyectar instancias de clases concretas en objetos que sólo conozcan la interfaz que esa clase concreta implementa. Es imprescindible para aplicar inversión de dependencias.
- La librería de la comunidad Spring Data Elasticsearch, que implementa una versión de Spring Data para Elasticsearch, ofrece una abstracción de alto nivel en la comunicación con Elasticsearch a través de interfaces predefinidas por Spring Data y comunes a todas sus implementaciones. Sus implicaciones en el diseño ya han sido expuestas en las vistas arquitecturales.

5.2.3. React.js

El framework de *JavaScript* desarrollado por Facebook propone una solución nunca vista en otros *frameworks*. Generar una representación virtual del DOM (Documen Object Model) que es almacenada en memoria, de modo que los cambios se realizan sobre esa versión virtual y no sobre el DOM real lo que sería muy costoso. Finalizado ese proceso, se comparan los cambios entre el DOM real y el Virtual DOM y se actualizan sólo las partes nuevas.

Este cambio de paradigma, sumado a las posibilidades de modularizar el código en componentes agnósticos han hecho que la popularidad del *framework* en proyectos grandes crezca y sea adoptado por empresas como Netflix.

Además, posee una comunidad activa de desarrolladores que crean o adaptan librerías JavaScript de todo tipo, algunos de los cuales veremos la aplicación como:

- React-leaflet es un wrapper de Leaflet.js para React.
- Leaflet.js es una librería que incluye todo lo necesario para visualizar mapas y datos sobre ellos.
- Searchkit es una librería que empaqueta lógica de búsqueda contra Elasticsearch y visualización de los resultados y componentes de control.

6. Pruebas: encontrando la aguja en el pajar

El objetivo que nos poníamos al inicio de este proyecto era arreglar lo que nadie había arreglado. Encontrar una forma de que los geoportales devolvieran información relevante y útil para el usuario utilizando un sistema de filtrado espacial que realmente filtrara sobre los propios datos y no sobre descripciones simplificadas de sus geometrías.

Tras un arduo proceso de investigación y prototipado procedemos a valorar los resultados; a verificar las historias de usuario; y a mostrar en acción algunas de las funcionalidades clave.

Para ello agruparemos todas las historias de usuario según su temática y expondremos varias capturas de su funcionamiento en la aplicación real. Como la que se muestra en (Figura 36) donde se visualiza el estado de la aplicación al acceder a ella por primera vez.

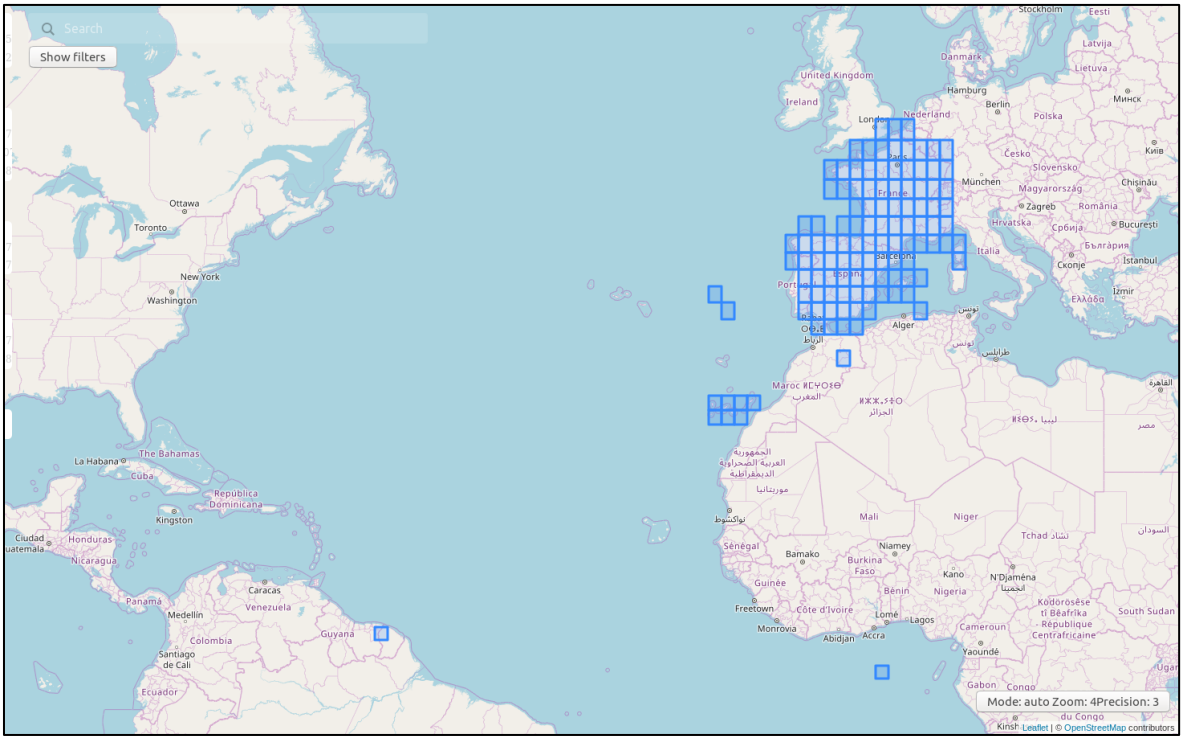


Figura 36. Página inicial

6.1. Funcionalidades de búsqueda

Las funcionalidades de búsqueda a verificar son:

- Realizar una búsqueda textual difusa
- Ver la cobertura de los resultados que sean colecciones de datos sobre un mapa
- Consultar toda la información de una colección de datos
- Ver los objetos espaciales de una colección de datos sobre un mapa
- Consultar toda la información de una organización
- Ver más resultados en una búsqueda
- Recibir sugerencias de búsqueda mientras tecleo
- Encontrar la organización que provee una colección de datos que he encontrado en una búsqueda, accediendo a ella desde la información de la colección

En la imagen (Figura 37) se muestra un escenario en el que se ha tecleado Zaragoza y se han obtenido todos los resultados relacionados con dicho término, no sólo aquellos que están en la ciudad sino también nombres de calles o pueblos incluidos en el dataset de Spain de Geonames.

En el mapa se pueden ver las coberturas de los datos. Si no se muestran los datos reales es porque debido a la gran cantidad de resultados, superior a 50, se decide mostrar un resumen en forma de área de coberturas. Este umbral es arbitrario y podría ser elevado a un número mayor, pero para garantizar rendimiento en el prototipo se ha fijado en 50.

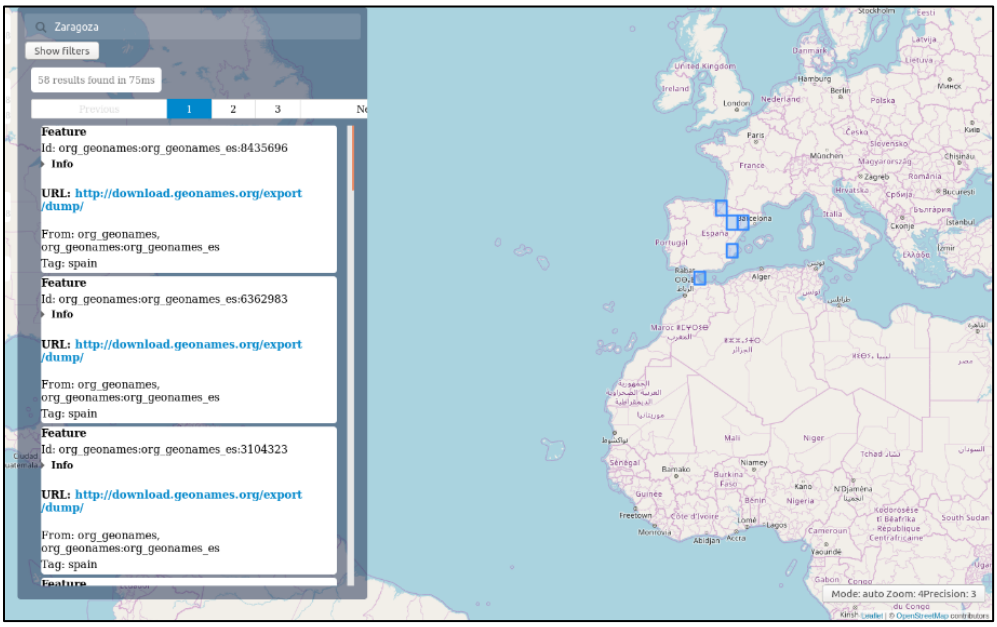


Figura 37. Búsqueda de "Zaragoza"

En la sección de resultados, se muestran los resultados ordenados por relevancia, con la opción de consultar toda su información (Figura 38), ya sean datos espaciales, colecciones u organizaciones. También se muestran las opciones de página para ver más resultados. Además, se puede acceder a la colección de datos o a la organización que pertenece un dataset.

Si hacemos zoom como se muestra en la misma figura, al haber un conjunto de datos inferior a 50, se muestran sus geometrías, en este caso, puntos.

La única funcionalidad de esta categoría que no está cubierta es la de ofrecer sugerencias de búsqueda ya que el componente visual del *searchbox* pertenece a la librería Searchkit y no facilita esa opción.

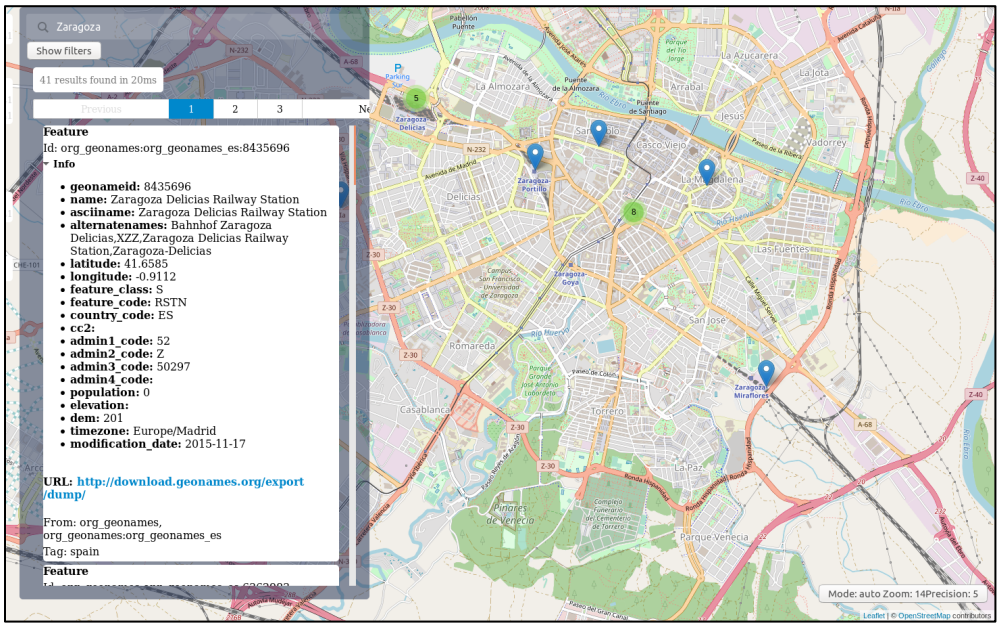


Figura 38. Búsqueda de "Zaragoza" ampliada

6.2. Funcionalidades de descarga y consumo de servicio

Las funcionalidades de descarga y consumo de servicio a verificar son:

- Descargar la colección de datos original en uno de los formatos específicos en los que fue originalmente distribuido.
- Ser redirigido a un sitio externo para descargar la colección de datos original en un formato específico cuando ésta no puede ser descargada desde el sistema.
- Ver una descripción del servicio remoto que provee una determinada colección de datos
- Crear una URI parametrizada que me permita descargar los resultados de una búsqueda como si fuera una colección de datos y en el formato que yo seleccione.
- Descargar los resultados de una búsqueda como si fuera una colección de datos y en el formato que yo seleccione.

Si observamos la figura superior (Figura 38) encontraremos el enlace de descarga externo del recurso original. Ésta es la única de las funcionalidades de descarga implementadas debido a que las demás no eran tan prioritarias y no eran fáciles de implementar.

6.3. Funcionalidades de filtrado

Las funcionalidades de filtrado a verificar son:

- Ver todos los filtros activos agrupados en un lugar
- Desactivar un filtro
- Desactivar todos los filtros de una vez
- Filtrar los resultados por sus facetas
- Ver las facetas disponibles
- Ver los posibles valores para cada faceta
- Ver cuántos resultados disponibles hay para cada valor de cada faceta
- Filtrar por área espacial
- Establecer un área compleja de filtrado sobre el mapa

El escenario representado muestra los resultados antes (Figura 39) y después (Figura 40) de aplicar un filtro, en este caso, para excluir los resultados que no pertenezcan al dataset de Francia.

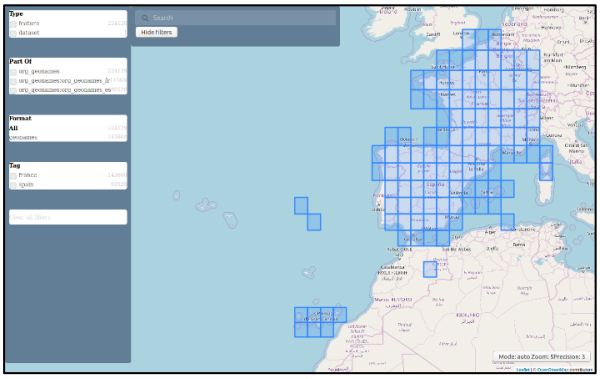


Figura 39. Resultados sin filtrar

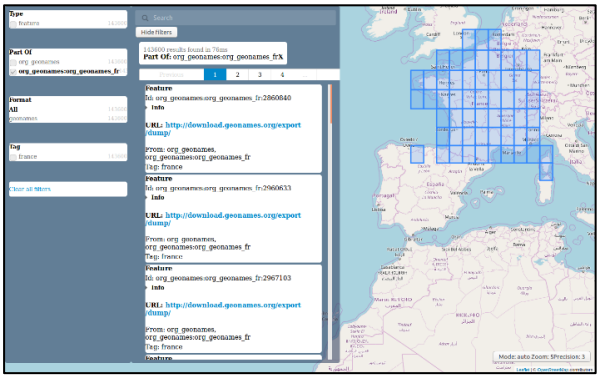


Figura 40. Resultados filtrando por dataset "Francia"

Se puede apreciar como claramente el área de los resultados cambia, pero se aprecian también más cosas. En el área de resultados se muestra el filtro activo encima del selector de página, desde el cual se puede desactivar un filtro o todos. Además, se observan en la sección de filtros las distintas facetas por las que se puede filtrar, así como sus valores disponibles y la cantidad de elementos que comparten cada valor.

Por último, queda el filtrado espacial, el cual ya se ha visto en acción en la sección anterior (Figura 38) al ampliar el área sobre Zaragoza de modo que los resultados fuera de visión desaparecían.

El motivo para no incluir las formas personalizadas es que, el acto de dibujar sobre un mapa y capturar esa entrada para convertirla al formato de búsqueda, GeoJSON, resultaba demasiado complejo para el escaso valor extra que añadía. Dado que el objetivo era probar el filtrado sobre los datos, servía el área de visión sobre el mapa.

6.4. Funcionalidades de administrador

Las funcionalidades de administrador a verificar son:

- Indicar la ubicación de un dataset y sus metadatos al sistema.
- Iniciar carga de un nuevo dataset.
- Añadir los datos de una organización.
- Potenciar un resultado por encima de los demás.
- Recibir un informe detallado si un proceso del sistema falla indicando el motivo del fallo.

En cuanto a la inserción, podemos observar la localización de los datos a insertar (Figura 41), en este caso el dataset *spain.csv* y su fichero adjunto *spain.yml* (Figura 42).

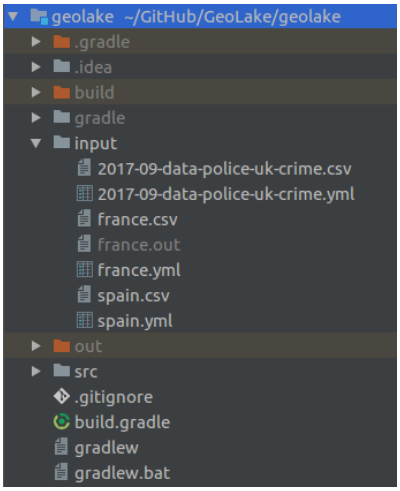


Figura 41. Ficheros a insertar

Figura 42. Fichero de configuración YML

Si lanzamos el programa con la dirección de los datos como argumento (Figura 45) podremos observar cómo los datos son insertados en lotes de 1000 (Figura 43 y Figura 44).

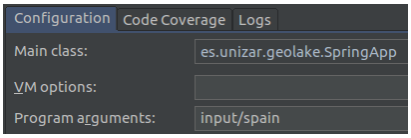


Figura 45. Argumentos del programa

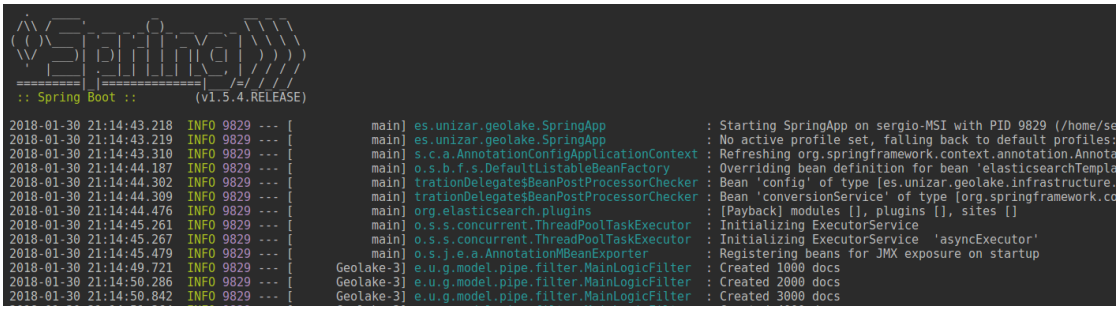


Figura 43. Inicio de la carga

Figura 44. Fin de la carga

6.5. Resultado global

Como se puede ver la Tabla 2. Todas las funcionalidades necesarias y recomendables han sido implementadas, así como varias de las opcionales de baja prioridad. Esto supone un acierto en cuanto a los objetivos asumidos y constata que el proyecto ha sido un éxito.

Tabla 2. Resultado de las pruebas

Actor	Historia	MoSCoW	Hecha
Usuario	Realizar una búsqueda textual difusa	Must	Si
	Ver la cobertura de los resultados que sean colecciones de datos sobre un mapa	Must	Si
	Desactivar todos los filtros de una vez	Must	Si
	Filtrar los resultados por sus facetas	Must	Si
	Ver las facetas disponibles	Must	Si
	Ver los posibles valores para cada faceta	Must	Si
	Filtrar por área espacial	Must	Si
	Consultar toda la información de una colección de datos	Should	Si
	Ver los objetos espaciales de una colección de datos sobre un mapa	Should	Si
	Consultar toda la información de una organización	Should	Si
	Ver más resultados en una búsqueda	Should	Si
	Desactivar un filtro	Should	Si
	Ser redirigido a un sitio externo para descargar la colección de datos original en un formato específico cuando ésta no puede ser descargada desde el sistema	Should	Si
	Encontrar la organización que provee una colección de datos que he encontrado en una búsqueda, accediendo a ella desde la información de la colección	Should	Si

	Ver todos los filtros activos agrupados en un lugar	Could	Si
	Ver cuántos resultados disponibles hay para cada valor de cada faceta	Could	Si
	Ver una descripción del servicio remoto que provee una determinada colección de datos	Could	No
	Recibir sugerencias de búsqueda mientras tecleo	Could	No
	Establecer un área compleja de filtrado sobre el mapa	Could	No
	Descargar la colección de datos original en uno de los formatos específicos en los que fue originalmente distribuido	Wont	No
	Crear una URI parametrizada que me permita descargar los resultados de una búsqueda como si fuera una colección de datos y en el formato que yo seleccione	Wont	No
	Descargar los resultados de una búsqueda como si fuera una colección de datos y en el formato que yo seleccione	Wont	No
Admin	Indicar la ubicación de un dataset y sus metadatos al sistema	Must	Si
	Iniciar carga de un nuevo dataset	Must	Si
	Añadir los datos de una organización	Sould	Si
	Potenciar un resultado por encima de los demás	Could	No
	Recibir un informe detallado si un proceso del sistema falla indicando el motivo del fallo	Wont	No

7. El futuro: Geolake Search como producto

Lo que hemos visto es sólo un atisbo de lo que esta aplicación puede ser. Con algunos cambios en su diseño y la reimplementación de algunas de sus partes clave, podríamos estar sin duda ante el mejor geoportal del mundo. Estos cambios serían:

Reimplementar la aplicación web bajo unos principios arquitecturales claros. Utilizando los *stores* de React.js para controlar el flujo de información y delegando toda la lógica de búsqueda al lado del servidor. Rediseñar además el estilo y unificarlo bajo un *sistema de diseño* (Fanguy, 2017).

Incluir un servidor intermedio entre Elasticsearch y la aplicación web de modo que encapsule la lógica de búsqueda en una sencilla API. Además, esto permitiría ajustar los pesos de las búsquedas sin tener que tocar la aplicación del *front-end*. Por otro lado, este servidor intermedio sería el encargado de traducir la respuesta de Elasticsearch al formato solicitado de modo que se estaría implementando la funcionalidad de tener URI parametrizadas para la descarga de resultados como un dataset.

Aprovechar la capacidad de agruparse en *clústeres* de Elasticsearch para crear una red de catálogos. De ese modo, cada institución podría cargar sus propios datos y ser responsable de ellos a la vez que se conecta a un *clúster* con las demás unificando de ese modo la búsqueda bajo todas ellas. Es una idea que ya se introdujo en los catálogos del OGC por lo que no se perdería esa capacidad.

Reemplazar el programa de inserción por una herramienta ETL como Talend que permita configurar gráficamente diversos flujos de datos de entrada hasta Elasticsearch. De ese modo la inserción sería mucho más sencilla de cara a un trabajador no versado en las aplicaciones de línea de comandos. Posteriormente se podría incluso crear un panel de control en el que se pudiera visualizar y editar cada elemento.

8. Gestión del proyecto

Dada la naturaleza exploratoria del problema, la gestión se ha visto irrevocablemente enfocada hacia las metodologías ágiles. Si bien las mismas ya suponen hoy en día una gran alternativa a las metodologías tradicionales, incluso en los entornos profesionales, son todavía más interesantes en proyectos con tanta incertidumbre que ni siquiera se conocen las tecnologías que formarán parte de la solución final.

Por ello la metodología escogida ha sido un sistema iterativo inspirado en Scrum, pero adaptado a un equipo unipersonal donde se es *Scrum Master* y *Product Owner* al mismo tiempo. También se ha considerado la figura del *Tutor* como una suerte de *Cliente* que se reúne con el *Product Owner* para priorizar las historias de usuario, las cuales no ha propuesto él, sino que han sido fruto del proceso de investigación previo y la definición de los casos de uso.

Cada una de las iteraciones se corresponde con cada uno de los prototipos de modo que el prototipo final suponga una evidencia de la viabilidad del concepto.

8.1. Planificación

Uno de los aspectos que más han determinado la planificación de este proyecto ha sido la disponibilidad para el desarrollo. Con un *bus factor 1* (Coplien & Harrison, 2004), cualquier limitación o contratiempo bloquean el proyecto por completo. Esto, sumado al resto de responsabilidades con las que este trabajo se compagina, ha llevado a que la planificación no se ha realizado en términos de fechas sino en términos de alcance, sabiendo que se cuenta con unos plazos flexibles. Los hitos que delimitaron dichos términos fueron finalizar el análisis de los geoportales, la captura de requisitos, el prototipado, el análisis de los prototipos, el diseño e implementación de la solución final y la finalización de la elaboración de esta memoria.

8.2. Esfuerzos

Medir los esfuerzos de un trabajo de estas características es complicado debido a que gran parte del trabajo es reflexión e investigación la cual no siempre es fácilmente medible. No obstante, podemos hacer una estimación bastante aproximada basándonos en los esfuerzos semanales, teniendo en cuenta que siempre estaremos subestimando el tiempo real ya que se trata de un proyecto que ha llevado 8 meses de duración y que ha ocupado la mente más horas de las que serían estimables en un análisis de esfuerzos.

Como muestran los gráficos (Figura *46*), los dos últimos prototipos son los que más tiempo han llevado debido a que, al ser los más avanzados, son los que más tiempo de programación y refinado han requerido para acercarse a la solución final.

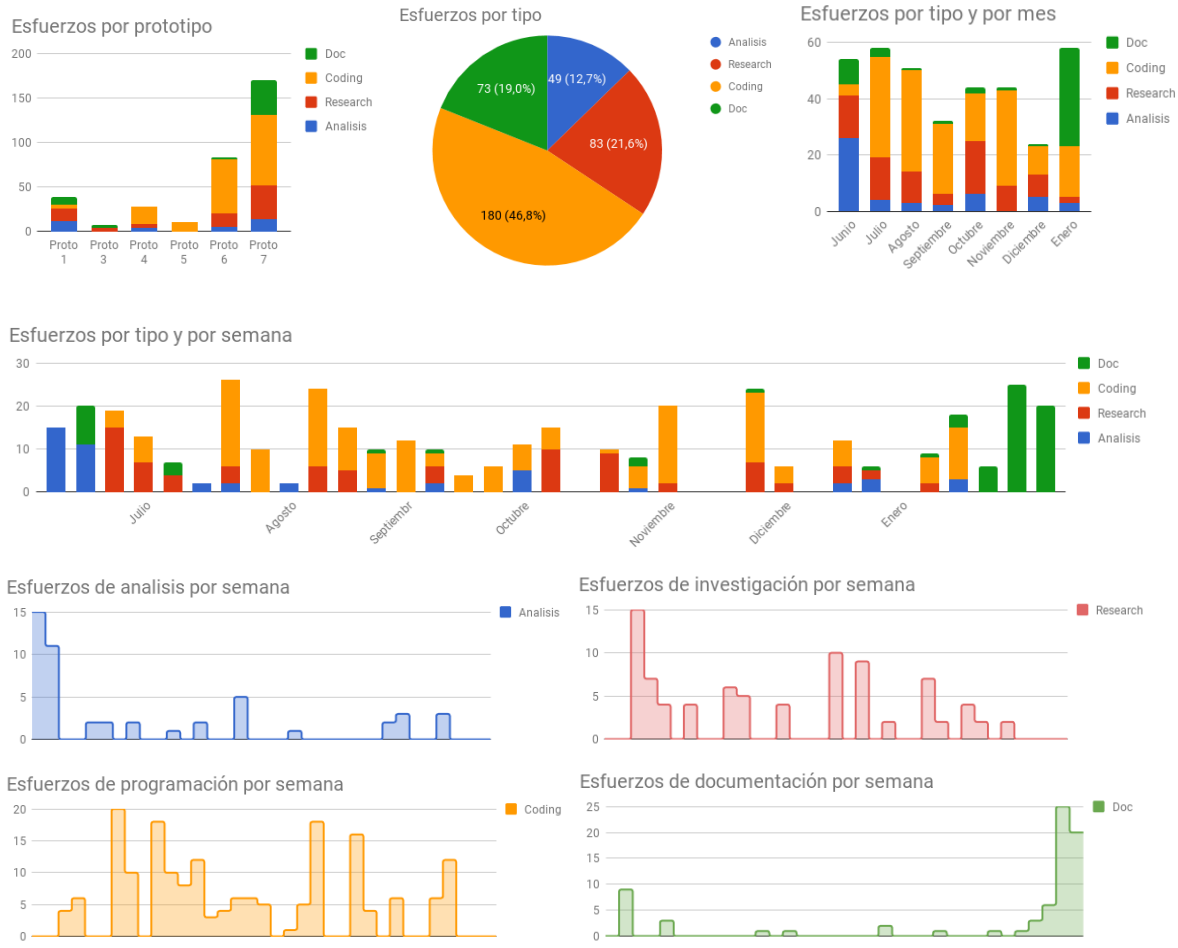


Figura 46. Diferentes vistas del esfuerzo dedicado

El total de horas es 385 sumando 49 de Análisis, 83 de investigación, 180 de programación y 73 de documentación.

8.3. Presupuesto

Estimar el presupuesto de un proyecto así no es sencillo. En primer lugar, porque en ningún momento ha sido concebido como un producto sino como un proyecto en aras del progreso. En segundo lugar, es un proyecto extendido en el tiempo, lo que obliga a considerar no sólo el salario en base a las horas imputadas sino también el coste de oportunidad de los 8 meses que se han dedicado a ello. Finalmente, hay que considerar el valor de la propiedad intelectual adquirida durante el desarrollo, la cual, en el caso de transferirse el proyecto, ha de rentabilizarse económicamente.

Si consideramos:

- 1100 euros como salario limpio adecuado para una jornada completa de un recién titulado en materia de desarrollo e investigación
- 40 horas semanales
- 385 horas totales
- 8 meses de duración real

$$\frac{385\ horas}{40\ horas\ por\ semana} = 9{,}65\ semanas \qquad \frac{9{,}65\ semanas}{4{,}3\ semanas\ por\ \ mes} = \ 2{,}24\ meses$$

$$2{,}24\ meses \times \ 1100\ euros\ al\ mes = 2464\ €$$

Para que el trabajador percibiese un salario de 1100 euros proporcional a sus horas trabajadas deberían imputarse un *mínimo de 2464 euros* en la venta del proyecto. Todo ello sin sumar amortizaciones e impuestos lo cual haría subir más aún ese mínimo.

Por otro lado, si considerando el coste de oportunidad, en base a los mismos términos sabremos que el trabajador, de haber optado por otro empleo con unas condiciones similares, pero durante el tiempo:

$$1100\ euros\ al\ mes\ \times 8\ meses\ de\ duración\ real = 8800€$$

Esto no quiere decir que se deba imputar ese coste íntegro al proyecto, principalmente porque no es equiparable una jornada completa a una jornada de 40 horas semanales, pero sí ha de tenerse en cuenta como un factor más.

El valor de la propiedad intelectual es difícilmente calculable con la poca perspectiva temporal que tenemos. Hace un año este proyecto tan sólo era una idea germinando en una cabeza, ahora constituye un prototipo que podría dar paso a un nuevo tipo de geoportales.

Con todos estos elementos en cuenta podemos concluir que las dos vías más razonables serían:

- Transferir el proyecto por una cuantía muy superior al límite inferior de los 2464€, de modo que el valor de la propiedad intelectual fuera también retribuido.
- Mantener el proyecto y desarrollarlo por cuenta propia hasta poder ofrecer una solución explotable comercialmente. Bien fuera ofreciendo el servicio o bien vendiendo la infraestructura.

9. Conclusiones

Hemos visto cómo la idea funciona y no sólo es realizable, sino que abre una nueva vía a cómo entendemos la búsqueda de datos espaciales. Los resultados de este proyecto arrojan luz sobre un paradigma caduco que no ha sabido adaptarse. Ésta idea es una nueva especie en el ecosistema que obligará a los geoportales antiguos adaptarse o morir.

Aún queda mucho camino por delante. Tan sólo se ha arañado la superficie del problema. Por delante quedan numerosos retos:

- La verdadera integración entre catálogos, que permitiría a cada productor de datos espaciales poseer su propio servicio donde publicara los suyos y que éstos fueran automáticamente compartidos por los demás.
- Consolidar la aplicación Geolake Upload como un software de calidad empresarial; con capacidad para convertir, procesar e insertar datos de múltiples formatos y capaz de interpretar un fichero de metadatos original.
- Resolver el problema del control de versiones de los datos, ya que un dataset puede ser insertado más veces si sufre modificaciones.

Bibliografía

Arms, W. Y. (2000). Automated Digital Libraries . *D-Lib Magazine, 6*(7/8).

Barker, R., & Clegg, D. (2004). *Case Method Fast-Track: A RAD Approach.* Addison-Wesley. Recuperado el 31 de 1 de 2018

Béjar, R., Gallardo, P., Gould, M., Muro-Medrano, P. R., Nogueras-Iso, J., & Zarazaga-Soria, F. J. (2004). A High level architecture for national SDI: The Spanish Case. *10th EC-GI & GIS Workshop.* Varsovia.

Bernard, L., Kanellopoulos, I., Annoni, A., & Smits, P. (2005). The European geoportal––one step towards the establishment of a European Spatial Data Infrastructure. *Computers, Environment and Urban Systems, 29*(1), 15-31.

Broder, A. (2002). A Taxonomy of Web Searc. *SIGIR Forum, 36*(2), 3-10.

Coplien, J., & Harrison, N. (2004). *Organizational patterns of agile software development.* Wiley.

Cron Sintaxis. (s.f.). Obtenido de Wikipedia: https://es.wikipedia.org/wiki/Cron_(Unix)#Sintaxis

Documen Object Model. (s.f.). Obtenido de Wikipedia: https://es.wikipedia.org/wiki/Document_Object_Model

Dowload Site. (s.f.). Obtenido de Geonames: http://download.geonames.org/export/dump/

Egenhofer, M., & Kuhn, W. (1999). Interacting with GIS. En *Geographical Information Systems: Principles, techniques, applications, and management* (págs. 401-412).

Elastic. (2018). *Elasticsearch Reference, Document APIs, Index API.* Obtenido de https://www.elastic.co/guide/en/elasticsearch/reference/current/docs-index_.html

Elastic. (2018). *Elasticsearch Reference, Search API.* Obtenido de https://www.elastic.co/guide/en/elasticsearch/reference/current/search.html

Experiencia de Usuario. (s.f.). Obtenido de Wikipedia: https://es.wikipedia.org/wiki/Experiencia_de_usuario

Fanguy, W. (1 de 12 de 2017). *Guide to design systems.* Obtenido de InvisionApp.com: https://www.invisionapp.com/blog/guide-to-design-systems/

Fowler, M. (2003). *Use Case.* Obtenido de MartinFowler.com: https://martinfowler.com/bliki/UseCase.html

Hern, A. (2018). Fitness tracking app gives away location of secret us army bases. *The Guardian*, 1. Obtenido de https://www.theguardian.com/world/2018/jan/28/fitness-tracking-app-gives-away-location-of-secret-us-army-bases

Hill, L. L. (2004). Georeferencing in Digital Libraries. *D-Lib Magazine*.

Magure, D. J., & Longey, P. A. (2005). The emergence of geoportals and their role in spatial data infrastructures. *Computers, environment and urban systems, 29*(1), 3-14.

OGC. (1999). *OpenGIS - Catalog Interface Implementation Specification (Version 1.0).* Obtenido de http://portal.opengeospatial.org/files/?artifact_id=831

OGC. (2007). *OpenGIS® Catalogue Services Specification (Version 2.0.2).* Obtenido de http://portal.opengeospatial.org/files/?artifact_id=20555

Reed, J. (2015). *Introducing EarthWorks, Stanford's new GIS data discovery application*. Obtenido de Stanford Libraries: http://library.stanford.edu/blogs/digital-library-blog/2015/04/introducing-earthworks-stanfords-new-gis-data-discovery

Rentería Agualimpia, W. (2015). *Quality Assessment for Semantically Close Geographical Properties.* Universidad de Zaragoza.

Rodríguez Pascual, A. F. (2014). Latest Developments and activities in the Spanish NSDI. *Proceedings of the AGILE'2014 International Conference on Geographic Information Science.* Castellón.

Rodríguez Pascual, A. F., López Romero, E., Abad Power, P., & Sánchez Maganto, A. (2005). The IDEE (Spanish SDI) Geoportal: technical aspects of an emerging reality. *22nd International Cartographic Conference.* A Coruña.

Rose, D. E., & Levinson, D. (2004). Understanding User Goals in Web Search. *13th International Conference on World Wide Web.* New York.

Rutledge, G. K., Alpert, J., Stouffer, R., & Lawrence, B. (2003). The NOAA Operational Model Archive and Distribution System (NOMADS). *Proceedings of the Tenth ECMWF Workshop on the Use of High Performance Computers in Meteorology* , (págs. 106-109). Reading, UK.

Solr Documentation. (s.f.). Obtenido de Apache.org: https://lucene.apache.org/solr/guide/7_2/solr-tutorial.html

The Apache Software Foundation. (s.f.). *Solr*. Obtenido de https://lucene.apache.org/solr/

Web Feature Service. (s.f.). Obtenido de Wikipedia: https://es.wikipedia.org/wiki/Web_Feature_Service

Web Map Service. (s.f.). Obtenido de Wikipedia: https://es.wikipedia.org/wiki/Web_Map_Service

Wikipedia Contributors. (31 de 1 de 2018). *MoSCoW method.* Obtenido de Wikipedia, The Free Encyclopedia: https://en.wikipedia.org/wiki/MoSCoW_method

Williamson, I. P., Rajabifard, A., & Feeney, M.-E. F. (2004). *Developing spatial data infrastructures: from concept to reality.* CRC Press.

Anexo A. Metodología de análisis de geoportales

La metodología para la exploración de los portales ha sido:

1. En base a la ficha realizada, hacer una primera exploración interactuando con los portales a fin de encontrar en ellos comportamientos interesantes. En primera instancia no buscamos nada en concreto, sino que recogemos lo que hace especial a cada uno.
2. Tras acabar la primera versión de las fichas, se amplían las mismas con una segunda revisión de los sitios. Esta vez comprobando si ofrecen las características descubiertas en los otros sistemas.
3. En base a las fichas terminadas y con el apoyo del portal original, se crea una tabla de funcionalidades con todas y cada una de las funciones vistas en los portales asegurándose de no duplicarlas bajo distintos nombres.
4. Finalmente se rellenan dichas tablas comparativas marcando para cada portal cuales de ellas cumplen.

Tabla 3. Plantilla de tabla descriptiva de geoportal

	Aspecto y opciones	Funcionalidad/Comportamiento
Página principal		
Página de resultados		
Filtro por texto		
Filtro por facetas		
Filtro geográfico		
Detalle individual		
Descarga		

Anexo B. Resultados del análisis de geoportales

Aquí se plasman las tablas resultantes de los análisis de cada geoportal. Son unos datos imprescindibles para conocer a fondo las alternativas y tener claro el punto de partida.

B.1. INSPIRE Geoportal (Comisión Europea, Unión Europea)

	Aspecto y opciones	Funcionalidad/Comportamiento
Página principal	La página de resultados con una query vacía y sin filtros activos. En el lugar de los resultados muestra una nube de palabras.	Puedes iniciar una búsqueda tecleando una query o activando el filtrado del mapa.
Página de resultados	Muestra media área principal con un cuadro de búsqueda, un mapa. La otra media contiene un cuadro de búsqueda, con un pequeño filtro según tipo de recurso y los resultados. Un botón "búsqueda avanzada" revela los filtros por faceta, así como los filtros y queries activas.	Si la query es vacía, la nube de palabras que muestra son posibles valores para los filtros, sean de la faceta que sean. Cada resultado tiene dos iconos que permiten mostrar sobre el mapa su bounding box o sus datos (si tiene capas disponibles) sin entrar en la vista detalle.
Filtro por texto	Muestra sugerencias *on typing*. No tiene fuzzy search.	Si se efectúa la query: *hydrography*. los resultados son satisfactorios. *hydrogrephy*: el servicio no arroja ningún resultado. Las queries son acumulativas y de tipo "and".
Filtro por facetas	Por tipo de Dato. Por Origen (lugar). Por Idioma de los metadatos. Por Temática. Por Categoría de la Temática. Por Tipo de servicio.	Filtrado tipo "and". El filtro "Tipo de dato" sólo es editable cuando la query está vacía.
Filtro geográfico	Muestra un mapa	Permite seleccionar bounding box para acotar la búsqueda. Los resultados serán aquellos recursos que tengan parte de su bounding box dentro del bounding box de búsqueda.
Detalle individual	En la misma página, en la sección de resultados, muestra una ficha mal formateada con los datos básicos: Título, idioma, una visualización de la bounding box en otro mapa con otra tecnología distinta al mapa principal, las proyecciones en las que está disponible.	

	En el mapa principal se muestra, además, el bounding box del dataset.	
Descarga	Apenas se podría decir que el portal ofrece la descarga de los datasets. Sólo en alguna ocasión muestra un enlace en el detalle del dataset que conduce a la fuente original.	

B.2. Geospatial Platform (FGDC, EEUU)

	Aspecto y opciones	Funcionalidad/Comportamiento
Página principal	La página de resultados con una query vacía y con el filtro de la faceta "Type" activado para el valor "geospatial".	
Página de resultados	Muestra los resultados en un área principal Muestra un cuadro de búsqueda, un mapa pequeño, los filtros activos y los propios filtros en una barra lateral.	
Filtro por texto	No tiene fuzzy search.	Si se efectúa la query: *hydrography*. los resultados son satisfactorios. *hydrogrephy*: el servicio no arroja ningún resultado.
Filtro por facetas	Por Organización. Por Etiqueta. Por Tipo de organización. Por Formato. Por Temática. Por Tipo de dataset. Por Estado. Por Categoría de temática.	Filtrado tipo "and".
Filtro geográfico	Muestra un mapa	Permite seleccionar bounding box para acotar la búsqueda. Los resultados serán aquellos recursos que tengan parte de su bounding box dentro del bounding box de búsqueda.
Detalle individual	Lanza a una página independiente Una sección principal muestra título, descripción, enlaces de descarga/consumo del recurso original en sus formatos disponibles, información del propio recurso. Una barra lateral muestra una descripción de la organización, información de contacto, funciones sociales y el bounding box del recurso sobre un pequeño mapa.	En el caso de los servicios existe una opción llamada "View Map" que permite visualizar los datos sobre un mapa.
Descarga	Las opciones se limitan a ofrecer el recurso original para su descarga si es un dataset o para su consumo si es un servicio.	

B.3. EarthWorks (Universidad de Stanford, EEUU)

	Aspecto y opciones	Funcionalidad/Comportamiento
Página principal	Muestra un área principal con un cuadro de búsqueda, una sección con 4 facetas y sus posibles valores y un mapa.	Puedes iniciar una búsqueda tecleando una query, activando uno de los filtros o activando el filtrado del mapa.
Página de resultados	Muestra la caja de búsqueda en una barra superior. Muestra los resultados, los filtros activos y un mapa en un área principal. Muestra unos propios filtros en una barra lateral.	En el mapa se muestran los bounding box de los resultados al pasar el puntero por encima.
Filtro por texto	Muestra sugerencias *on typing*. No tiene fuzzy search.	Si se efectúa la query: *hydrography*. los resultados son satisfactorios. *hydrogrephy*: el servicio no arroja ningún resultado.
Filtro por facetas	Por Institución (Organización). Por Autor. Por Publisher. Por Temática. Por Lugar (textual). Por Año (un *slider* define mín. y máx.). Por Disponibilidad. Por Tipo de dato. Por Formato.	Filtrado tipo “and”.
Filtro geográfico	Muestra un mapa	Permite activar y desactivar el filtrado por mapa. Cuando está activo, cambiar la posición del mapa hace cambiar el filtro.
Detalle individual	Lanza a una página independiente Una sección principal muestra título, descripción, y otros metadatos del dataset así como un mapa que muestra los propios datos sobre él con la posibilidad de leer los valores de sus atributos. Un discreto cuadro lateral muestra enlaces para guardar el dataset en marcadores, enviarlo por email, acceder a los servicios que lo ofrecen, descargar sus metadatos, descargar el dataset en varios formatos, visualizar en Carto (requiere tener cuenta activa).	Al seleccionar los objetos geográficos que se muestran en el mapa, los atributos y valores del objeto seleccionado se muestran en una tabla a su lado.
Descarga	Las opciones se limitan a ofrecer el recurso original para su descarga si es un dataset o para su consumo si es un servicio.	

B.4. GeoNorge (Kartverket, Noruega)

	Aspecto y opciones	Funcionalidad/Comportamiento
Página principal	La página de resultados con una query vacía y sin filtros activos.	Puedes iniciar una búsqueda tecleando una query o activando uno de los filtros.
Página de resultados	Muestra la caja de búsqueda y un carrito en una barra superior. Muestra los resultados en un área con las pestañas: all, dataset, service, app. Muestra el resto de filtros, páginas relacionadas y "guardar cómo" en una barra lateral.	Las pestañas son un filtro para el tipo de recurso. Los resultados pueden tener subresultados. Se pueden añadir resultados al carrito. "Guardar como" permite descargar el carrito.
Filtro por texto	Muestra sugerencias *on typing* clasificadas por tipo de recurso.. No tiene fuzzy search.	Si se efectúa la query: *hydrography*. los resultados son satisfactorios. *hydrogrephy*: el servicio sólo arroja 2 resultados que seguramente contengan dicho término. No permite borrar la búsqueda. Si se borra el texto del cuadro de búsqueda, la query se mantiene. La única manera de limpiar el campo es borrándolo de la URL.
Filtro por facetas	Por Temática. Por Cooperación y Normativa. Por Área. Por Forma de distribución. Por Organización. Por Accesibilidad.	Filtrado tipo "and". No permite visualizar los filtros activos en un sólo lugar ni desactivarlos todos a la vez.
Filtro geográfico	No se muestra mapa.	
Detalle individual	Lanza a una página independiente Muestra el título, un conjunto de acciones comunes que están activas o desactivadas para cada dataset en concreto tales como "Mostrar en mapa", descargar, mostrar cobertura en mapa (bounding box en una web externa), contacto, página relacionada y visualizado de "tabla del producto" y "especificación del producto" y "cartografía" que redirigen a 3 páginas con sus respectivos PDF.	
Descarga	La página a la que se accede seleccionando el carrito recoge toda la potencia a la hora de descargar los datasets. Muestra en columna los datasets añadidos con desplegables que muestran las opciones de postprocesado que ofrece y la	Los datasets añadidos al carrito son combinados generando un único documento que es descargado. Suele generar problemas de incompatibilidades que pueden tratar de ser solventadas con los post-procesados. Se pueden aplicar 3 postprocesados a

	posibilidad de quitarlos de carrito.	cada dataset o al dataset final: Delimitar el área de la cual queremos sus datos, Seleccionar a proyección y el formato en los que están codificados de entre los que tengan disponibles, no permite su conversión.

B.5. IDEE (Ministerio de Fomento, España)

	Aspecto y opciones	Funcionalidad/Comportamiento
Página principal	Un espacio principal donde se listan las temáticas disponibles. Una barra lateral con dos buscadores y filtro para buscar datasets, servicios o series. No muestra ningún mapa.	Seleccionar una de las temáticas redirige a la página de resultados con query varía y facetado por la temática seleccionada.
Página de resultados	Muestra los resultados de la búsqueda. Muestra un diminuto mapa que con las bounding box de los resultados en pantalla. Cuenta con un botón para limpiar la query y las facetas establecidas. No muestra la query y facetas establecidas. Los servicios muestran un menú desplegable cuyas opciones no tienen ningún efecto al ser activadas.	En el mapa se resaltan los bounding box de los resultados al pasar el puntero por encima.
Filtro por texto	Sugerencias *on typing.* No tiene fuzzy search.	Tecleando ”presa” se sugiere “Presas” e “Inventario de Presas”. Si se efectúa la query: *Presa*. los resultados no son satisfactorios. *Inventario de Presas*: el servicio no arroja ningún resultado. *Presas*: el resultado es satisfactorio y además, el título del primer resultado es “Inventario de Presas”.
Filtro por facetas	Por Tipo de Recurso. Por Temática. Por Tipo de Servicio.	Tipo “and” PERO no permite seleccionar más de un campo por cada faceta Ej.: No permite seleccionar simultáneamente "Servicio" y "Dataset" aunque existan Servicios y Datasets que comparten propiedades, como es el título.
Filtro geográfico		Permite seleccionar bounding box para acotar la búsqueda. Los resultados serán aquellos recursos que tengan parte de su bounding box dentro del bounding box de búsqueda.
Detalle individual	Lanza a una página independiente Muestra título, descripción, enlaces	En el caso de los servicios existe una opción llamada “Add to map” que al ser

	de descarga/consumo del recurso original en sus formatos disponibles, recursos relacionados e información del propio recurso como su temática, su identificador y sus restricciones de uso así como información de la identidad que provee los datos.	seleccionada redirige a una página en blanco en tanto en Firefox 57 como en Chrome 63. Se presume que la intención de esta función era mostrar los datos sobre un mapa ya que la vista del detalle individual no muestra ninguno.
Descarga	Las opciones se limitan a ofrecer el recurso original para su descarga si es un dataset o para su consumo si es un servicio.	

B.6. ArcGIS HUB (ESRI, EEUU)

	Aspecto y opciones	Funcionalidad/Comportamiento
Página principal	La página de resultados con una query vacía y sin filtros activos.	
Página de resultados	Muestra un cuadro de búsqueda y un mapa en una fila superior que ocupa todo el ancho de la pantalla. Muestra los resultados en un área principal. Muestra los filtros en una barra lateral.	Al hacer una búsqueda centra el mapa superior para mostrar todos los bounding box de los resultados sobre el mapa. Algunos datasets no tienen bounding box
Filtro por texto	Muestra sugerencias *on typing.* Permite cierto grado de fuzzy search.	Si se efectúa la query: *hydrography*. se obtienen 1910 resultados. *hydrogrephy o hydrogrnphy:* se obtienen 952 resultados. No permite el borrado de la query de texto si no es borrándola de la URL
Filtro por facetas	Por Organización. Por Etiqueta. Por Tipo de dataset.	Filtrado tipo “and”.
Filtro geográfico	Muestra un mapa	No permite seleccionar bounding box para acotar la búsqueda. Tan sólo se puede seleccionar una opción de “Cerca de ...” en el cuadro de búsqueda.
Detalle individual	Se mantiene la misma página con el mapa superior donde se muestran los propios datos sobre él. Una sección de superior muestra 3 pestañas y un desplegable para descargar los datos o acceder a su API. En la pestaña Introducción se muestran título, descripción, y una herramienta interactiva de exploración estadística de los atributos. En la pestaña Datos se muestran los datos en crudo en forma de tabla. En la pestaña explorador de API se	En el caso de los servicios existe una opción llamada “View Map” que permite visualizar los datos sobre un mapa.

	muestra una herramienta interactiva que permite hacer peticiones contra la API original estableciendo de forma cómoda los valores de cada atributo.	
Descarga	Las opciones se limitan a ofrecer el recurso original para su descarga y el enlace a su API para su consumo.	

B.7. Resumen de características

Este es el resultado de agregar todas las funcionalidades encontradas e indicar qué geoportales las cumplen:

Página principal	**A**	**B**	**C**	**D**	**E**	**F**
Obtener sugerencias de las búsquedas habituales	0	0	0	0	0	0
Obtener sugerencias de los resultados más habituales	0	0	0	0	0	0
Obtener sugerencias de los facetados más habituales	0	0	1	0	1	0
Obtener sugerencias de los valores de facetados más habituales	1	0	1	0	0	0

Página de resultados	**A**	**B**	**C**	**D**	**E**	**F**
Ver los resultados de la búsqueda sobre un mapa	1	0	0	0	0	0
Ver los bounding box de los resultados sobre el mapa	0	0	0	0	1	1
Resaltar la bounding box de cada resultado selectivamente	1	0	1	0	1	1
Mezclar los resultados de sucesivas búsquedas vía unión o intersección.	0	0	0	0	0	0
Mostrar la accesibilidad de cada resultado en la vista de resultados	0	0	1	1	0	1
Mostrar los formatos en los que está disponible cada resultado en la vista de resultados	0	1	0	0	0	0

Filtro por texto	**A**	**B**	**C**	**D**	**E**	**F**
Buscar resultados relacionados con una query textual	1	1	1	1	1	1
Obtener sugerencias mientras se teclea la query	1	0	0	1	1	1
Buscar resultados con una query textual aunque la query contenta errores tipográficos (*fuzzy search*)	0	0	0	0	0	1

Descarga	**A**	**B**	**C**	**D**	**E**	**F**
Descargar el recurso original	1	1	1	1	1	1
Descargar una versión sintética del recurso original	0	0	1	1	0	0
Descargar una colección de resultados como recurso sintético	0	0	0	1	0	0

Filtro geográfico	**A**	**B**	**C**	**D**	**E**	**F**
Filtrar por bounding box con la operación "*within*"	0	0	0	0	1	0
Filtrar por bounding box con la operación "*insertects with*"	1	1	1	0	1	0
Elegir el tipo de operación que se aplica al filtrar por bounding box	0	0	0	0	1	0
Dibujar la bounding box de filtrado sobre el mapa	1	1	0	0	1	0
Dibujar un área de búsqueda distinta a una bounding box.	0	0	0	0	0	0

Filtro por facetas	A	B	C	D	E	F
Refinar los resultados de una búsqueda vía facetas	1	1	1	1	1	1
Desactivar filtros selectivamente	1	1	1	1	1	1
Desactivar todos los filtros a la vez	1	1	1	0	1	0
Mostrar todos los filtros activos agrupados en un lugar	1	1	1	0	0	0
Agregan los filtros con el operador OR	0	0	0	1	0	0
Agregan los filtros con el operador AND	1	1	1	0	1	1
Seleccionar el tipo de operador para agregar los filtros	0	0	0	0	0	0
Facetado por Tipo de Recurso (Dataset, Servicio, etc.)	1	0	0	1	1	1
Facetado por Temática	1	1	1	1	1	0
Facetado por Categoría de la Temática	1	1	0	0	0	0
Facetado por Tipo de servicio	0	0	0	1	1	0
Facetado por Organización	0	1	1	1	0	1
Facetado por Tipo de Organización	0	1	0	0	0	0
Facetado por Autor	0	0	1	0	0	0
Facetado por Etiqueta	0	1	0	0	0	1
Facetado por Formato	0	1	1	0	0	0
Facetado por Estado	0	1	0	0	0	0
Facetado por Lugar	1	0	1	1	0	0
Facetado por Año	0	0	1	0	0	0
Facetado por Normativa	0	0	0	1	0	0
Facetado por Accesibilidad	0	0	1	1	0	0
Facetado por Idioma	1	0	0	0	0	0

Detalle individual	A	B	C	D	E	F
Ver el detalle individual en la misma página de resultados	1	0	0	0	0	1
Ver el bounding box del resultado sobre un mapa	1	1	0	0	0	0
Ver los datos del resultado sobre un mapa	1	0	1	1	0	1
Ver los datos del resultado en forma de tabla	0	0	1	1	0	1
Ver Título	1	1	1	1	1	1
Ver Descripción	1	1	1	1	1	1
Ver Enlaces de descarga/consumo	1	1	1	1	1	1
Ver Recursos relacionados	0	0	0	0	1	1
Ver Accesibilidad	1	1	1	1	1	1
Ver Temática	1	1	1	1	1	0
Ver Organización/Autor	1	1	1	1	1	1
Ver Estado	0	0	0	0	1	0

Anexo C. Casos de uso

Aquí es donde se describen textualmente todos los casos de uso recogidos en el sistema. Sus descripciones son detalladas y muestran los pasos que ejecutaría un usuario para realizar ese escenario, así como los pasos que se darían en variaciones alternativas del mismo.

C.1. CU_DD - Descargar un dataset

Principal escenario de éxito

1. El Usuario (CU_BD - Buscar un dataset) concreto
2. El Sistema muestra, entre otros, el resultado que espera el usuario
3. El Usuario (CU_CD - Consultar la descripción de un dataset) que estaba buscando
4. El Sistema muestra, entre otras cosas, los enlaces a diferentes distribuciones del dataset caracterizadas por su formato
5. El Usuario pulsa sobre el enlace de descarga de la distribución que desea
6. El Sistema responde enviando como respuesta un fichero que contiene el dataset en el formato indicado en la distribución

Escenario alternativo: Enlace de descarga externo

6. b. El Sistema responde redirigiendo al usuario a un servicio externo que debería dar como respuesta un fichero que contiene el dataset en el formato indicado en la distribución

C.2. CU_AS - Acceder a un servicio

Principal escenario de éxito

1. El Usuario (CU_BD - Buscar un dataset)
2. El Sistema muestra, entre otros, el resultado que espera el usuario
3. El Usuario (CU_CD - Consultar la descripción de un dataset) que esperaba
4. El Sistema muestra, entre otras cosas, los enlaces a las distribuciones del dataset que son servicios web caracterizados por el tipo de servicio
5. El Usuario copia el enlace del servicio de distribución que le interese en ese momento

C.3. CU_CD - Consultar la descripción de un dataset

Principal escenario de éxito

1. El Usuario (CU_BD - Buscar un dataset)
2. El Sistema muestra, entre otros, el dataset que espera el usuario
3. El Usuario selecciona el dataset que esperaba
4. El Sistema muestra la toda la información del dataset incluidas sus features sobre el mapa

C.4. CU_DR - Descargar el resultado de una búsqueda como dataset

Principal escenario de éxito

1. El Usuario (CU_BD - Buscar un dataset)
2. El Sistema muestra, entre otros, el resultado que espera el usuario
3. El Usuario (CU_F - Filtrar) los resultados hasta obtener sólo aquellos que desea
4. El Sistema muestra los resultados refinados que el usuario desea combinar
5. El Usuario selecciona la opción de descargarlos como un solo dataset
6. El Sistema genera un enlace de descarga que ofrece el resultado de la búsqueda como un dataset
7. El Usuario selecciona el enlace
8. El Sistema descarga el recurso en el equipo

C.5. CU_CO - Consultar la descripción de una organización

Principal escenario de éxito

1. El Usuario (CU_BO - Buscar una organización)
2. El Sistema muestra, entre otras, la organización que espera el usuario
3. El Usuario selecciona la organización que esperaba
4. El Sistema muestra la toda la información de la organización

C.6. CU_BD - Buscar un dataset

Principal escenario de éxito

1. El Usuario (CU_RQ - Realizar una query) con el texto por el que quiere buscar
2. El Sistema muestra los resultados de búsqueda para esa query, así como el área de cobertura de estos sobre el mapa. Entre éstos se encuentra el dataset que espera el usuario

Escenario alternativo: Mostrar siguiente grupo de resultados

2. b. El Sistema muestra los resultados de búsqueda para esa query, así como el área de cobertura de estos sobre el mapa. Entre éstos no se encuentra el dataset que espera el usuario
3. b. El Usuario solicita mostrar el siguiente grupo de resultados
4. b. El Sistema muestra un nuevo grupo de resultados. Entre estos se encuentra el dataset que espera el usuario

C.7. CU_BDC - Buscar un dataset conocido

Principal escenario de éxito

1. El Usuario (CU_RQ - Realizar una query) con el nombre del dataset conocido
2. El Sistema muestra los resultados de búsqueda para esa query entre los cuales está el dataset que espera el usuario

Extensión: Usar filtrado

2. b. El Sistema muestra los resultados de búsqueda para esa query entre los cuales no está el dataset que espera el usuario
3. b. El Usuario (CU_F - Filtrar) bajo el criterio deseado
4. b. El Sistema mostrará un nuevo grupo de resultados entre los cuales está el dataset que espera el usuario

C.8. CU_BDS - Buscar un dataset sobre algo

Principal escenario de éxito

1. El Usuario (CU_RQ - Realizar una query) con aquello con lo que está relacionado el dataset que busca
2. El Sistema muestra los resultados de búsqueda para esa query entre los cuales hay algunos datasets interesantes para el usuario
3. El Usuario (CU_F - Filtrar) por faceta o lugar geográfico para refinar los resultados obtenidos
4. El Sistema muestra un nuevo grupo de resultados entre los cuales hay muchos más datasets interesantes para el usuario siendo uno de ellos lo que el usuario esperaba

C.9. CU_RQ - Realizar una query

Principal escenario de éxito

1. El Usuario introduce el texto por el que quiere buscar
2. El Sistema muestra sugerencias de nombres mientras el texto es introducido
3. El Usuario selecciona la opción de buscar
4. El Sistema muestra los resultados de búsqueda para esa query

Escenario alternativo: Aceptar la sugerencia

3. b. El Usuario selecciona una de las sugerencias

C.10. CU_BO - Buscar una organización

Principal escenario de éxito

1. El Usuario (CU_RQ - Realizar una query) con el texto por el que quiere buscar
2. El Sistema muestra los resultados de búsqueda para esa query entre los cuales está la organización que espera el usuario

Escenario alternativo: Mostrar siguiente grupo de resultados

1. El Usuario (CU_RQ - Realizar una query) con el mensaje que quiere buscar
2. El Sistema muestra los resultados de búsqueda para esa query entre los cuales no está la organización que espera el usuario
3. El Usuario solicita mostrar el siguiente grupo de resultados
4. El Sistema muestra un nuevo grupo de resultados entre los cuales está la organización que espera el usuario

C.11. CU_BOC - Buscar una organización conocida

Principal escenario de éxito

1. El Usuario (CU_RQ - Realizar una query) con el nombre de la organización conocida
2. El Sistema muestra los resultados de búsqueda para esa query entre los cuales está la organización que espera el usuario

Extensión: Usar filtrado

2. b. El Sistema muestra los resultados de búsqueda para esa query entre los cuales no está el dataset que espera el usuario
3. b. El Usuario (CU_F - Filtrar) bajo el criterio deseado
4. b. El Sistema mostrará un nuevo grupo de resultados entre los cuales está el dataset que espera el usuario

C.12. CU_BODS - Buscar la organización que provee datasets sobre algo

Principal escenario de éxito

1. El Usuario (CU_RQ - Realizar una query) con aquello con lo que está relacionado el dataset que busca
2. El Sistema muestra los resultados de búsqueda para esa query entre los cuales hay algunos datasets interesantes para el usuario
3. El Usuario (CU_F - Filtrar) por faceta o lugar geográfico para refinar los resultados obtenidos
4. El Sistema muestra un nuevo grupo de resultados entre los cuales hay muchos más datasets interesantes para el usuario siendo uno de ellos lo que el usuario esperaba

C.13. CU_BODC - Buscar la organización que provee un dataset conocido

Principal escenario de éxito

1. El Usuario (CU_BD - Buscar un dataset)
2. El Sistema muestra, entre otros, el dataset que espera el usuario
3. El Usuario (CU_CD - Consultar la descripción de un dataset)
4. El Sistema muestra la toda la información del dataset, incluida la organización a la que pertenece
5. El Usuario selecciona la organización
6. El Sistema redirige al usuario a la organización que esperaba

C.14. CU_F - Filtrar

Principal escenario de éxito

1. El Sistema muestra un conjunto de resultados, ya sea el inicial o uno procedente una búsqueda realizada por el Usuario
2. El Usuario aplica un filtro
3. El Sistema muestra un nuevo conjunto de resultados filtrando los resultados del conjunto del paso 1 con el filtro establecido en el paso 2. También añade el filtro activo en la sección de filtros activos
4. El Usuario aplica un filtro adicional
5. El Sistema muestra un nuevo conjunto de resultados aplicando al conjunto de los resultados ya filtrados el nuevo filtro. También añade el filtro activo en la sección de filtros activos. A continuación, el usuario vuelve al paso 4 o finaliza el escenario

Escenario alternativo: Eliminar un filtro activo

4. b. El Usuario desactiva un filtro activo
5. b. El Sistema muestra un nuevo resultado añadiendo a los resultados ya filtrados los resultados que fueron eliminados por dicho filtro que cumplen los filtros que siguen activos. También retira el filtro activo en la sección de filtros activos. A continuación, el usuario vuelve al paso 4 o finaliza el escenario

Escenario alternativo: Eliminar todos los filtros

6. c. El Usuario selecciona la opción de eliminar todos los filtros activos
7. c. El Sistema muestra el mismo grupo de resultados que mostró antes de aplicar los filtros, así como hace desaparecer todos los filtros de la sección de filtros activos

C.15. CU_FF - Filtrar por faceta

Principal escenario de éxito

1. El Sistema muestra las facetas por las que se puede filtrar y sus respectivos valores disponibles, así como la cantidad de resultados que poseen ese valor para esa faceta
2. El Usuario selecciona uno de los valores disponibles para una de las facetas
3. El Sistema muestra un nuevo grupo de resultados con aquellos que poseen ese valor para esa faceta, así como el filtro activo en la sección de filtros activos y actualiza los valores disponibles para el resto de las facetas

C.16. CU_FA - Filtrar por área espacial

Principal escenario de éxito

1. El Usuario establece sobre el mapa un área sobre la cual quiere buscar
2. El Sistema representa dicha área de forma visual y muestra un nuevo grupo de resultados con aquellos que se encuentran ubicados en el área establecida, así como el filtro activo en la sección de filtros activos

C.17. CU_AD - Añadir un dataset datos al sistema

Principal escenario de éxito:

1. El Administrador entra en el Sistema y selecciona añadir un dataset
2. El Administrador indica al Sistema dónde está el dataset y el documento con los metadatos básicos que lo describen
3. El Administrador solicita al Sistema que se añada el dataset
4. El Sistema importa ambos recursos, los procesa y responde con un mensaje de éxito al Administrador

Escenario alternativo: El Sistema falla al importar los recursos

4. b. El Sistema informa del fallo de importación al Administrador respondiendo con un mensaje de error describiendo el problema
5. b. El Administrador puede volver a indicar qué quiere importar (paso 2) o puede cancelar el proceso

C.18. CU_AO - Añadir una organización al sistema

Escenario principal de éxito

1. El Administrador entra en el sistema y selecciona añadir una organización
2. El Administrador provee todos los datos relacionados con la organización
3. El Sistema procesa la petición, añade la organización y responde con un mensaje de éxito Al administrador

Escenario alternativo: El Sistema falla al crear la organización

3. b. El Sistema informa del fallo al Administrador respondiendo con un mensaje de error describiendo el problema
4. b. El Administrador puede volver a indicar los datos de la organización (paso 2) o puede cancelar el proceso

C.19. CU_PR - Potenciar unos resultados por encima de los demás

Flujo de éxito

1. El Administrador entra en el sistema y selecciona potenciar un resultado
2. El Administrador provee el resultado que quiere alterar y la cantidad de ventaja que desea
3. El Sistema procesa la petición, modifica dicho valor y responde con un mensaje de éxito

Escenario alternativo: El Sistema falla al crear la organización

3. b. El Sistema informa del fallo al Administrador respondiendo con un mensaje de error describiendo el problema
4. b. El Administrador puede volver a indicar los datos de la organización (paso 2) o puede cancelar el proceso

Anexo D. Historias de usuario

Las historias de usuario describen en lenguaje común las funcionalidades del sistema desde el punto de vista del usuario que las utiliza.

- Como **usuario**
  - quiero poder **realizar una búsqueda textual difusa**
  - para **encontrar recursos cuyo título o contenido contengan un texto similar al buscado**

- Como **usuario**
  - quiero poder **ver la cobertura de los resultados que sean colecciones de datos sobre un mapa**
  - para **tener una idea visual de qué área abarca cada uno**

- Como **usuario**
  - quiero poder **consultar toda la información de una colección de datos**
  - para **satisfacer alguna duda sobre él, acceder a otro elemento referenciado en él, como podría ser la organización a la que pertenece o asegurarme de que es la colección de datos que busco**

- Como **usuario**
  - quiero **poder ver los objetos espaciales de una colección de datos sobre un mapa**
  - para **asegurarme de que es la colección de datos que busco**

- Como **usuario** quiero
  - poder **consultar toda la información de una organización**
  - para **satisfacer alguna duda o para usarla como filtro en una futura búsqueda**

- Como **usuario** quiero
  - poder **descargar la colección de datos original en uno de los formatos específicos en los que fue originalmente distribuido**
  - para **tener acceso a él localmente y usarlo en otros sistemas**

- Como **usuario** quiero
  - poder **ser redirigido a un sitio externo para descargar la colección de datos original en un formato específico cuando ésta no puede ser descargada desde el sistema**
  - para **tener acceso a él localmente y usarlo en otros sistemas**

- Como **usuario** quiero
  - poder **ver una descripción del servicio remoto que provee una determinada colección de datos**
  - para **poder explotar dicho servicio desde otros sistemas**

- Como **usuario** quiero
  - poder **crear una URI parametrizada que me permita descargar los resultados de una búsqueda como si fuera una colección de datos y en el formato que yo seleccione**
  - para **explotarla como un servicio**

- Como **usuario** quiero
  - poder **descargar los resultados de una búsqueda como si fuera una colección de datos y en el formato que yo seleccione**
  - para **descargarla localmente como un dataset virtual**

- Como **usuario** quiero
  - poder **ver más resultados en una búsqueda**
  - para **continuar buscando el resultado adecuado, aunque no aparezca entre los primeros**

- Como **usuario** quiero
  - poder **recibir sugerencias de búsqueda mientras tecleo**
  - para **ahorrarme tiempo escribiendo o recibir ideas nuevas**

- Como **usuario** quiero
  - poder **encontrar la organización que provee una colección de datos que he encontrado en una búsqueda, accediendo a ella desde la información de la colección**
  - para **encontrar esa organización, aunque no conozca su nombre ni su temática**

- Como **usuario** quiero
  - poder **ver todos los filtros activos agrupados en un lugar**
  - para **conocer en todo momento qué cómo se está filtrando mi búsqueda**

- Como **usuario** quiero
  - poder **desactivar un filtro**
  - para **rectificar una decisión de filtrado previa**

- Como **usuario** quiero
  - poder **desactivar todos los filtros de una vez**
  - para **ver el resultado sin filtrar rápidamente**

- Como **usuario** quiero
  - poder **filtrar los resultados por sus facetas**
  - para **ver sólo aquellos cuyo valor para esa faceta coincida con el del filtro**

- Como **usuario** quiero
  - poder **ver las facetas disponibles**
  - para **saber por qué criterios puedo filtrar**

- Como **usuario** quiero
  - poder **ver los posibles valores para cada faceta**
  - para **saber qué valores puedo seleccionar en ese filtro**

- Como **usuario** quiero
  - poder **ver cuántos resultados disponibles hay para cada valor de cada faceta**
  - para **estimar la popularidad de cada valor y valorar su utilidad**

- Como **usuario** quiero
  - poder **filtrar por área espacial**
  - para **obtener sólo resultados que contengan datos relevantes en esa área**

- Como **usuario** quiero
  - poder **establecer un área compleja de filtrado sobre el mapa**
  - para **poder adaptar dicha área a la superficie que deseo sin estar limitado a geometrías simples**

- Como **administrador** quiero
  - poder **indicar la ubicación de un dataset y sus metadatos al sistema**
  - para **que éste los procese y añada desde la ubicación seleccionada**

- Como **administrador** quiero
  - poder **iniciar carga de un nuevo dataset**
  - para **aumentar la cantidad de dataset disponibles en el sistema**

- Como **administrador** quiero
  - poder **añadir los datos de una organización**
  - para **que dicha organización tenga una formación más útil o actualizada**

- Como **administrador** quiero
  - poder **potenciar un resultado por encima de los demás**
  - para **beneficiar unos resultados sobre todos en las búsquedas según mi criterio**

- Como **administrador** quiero
  - poder **recibir un informe detallado si un proceso del sistema falla indicando el motivo del fallo**
  - para **poder solventar el problema rápidamente**

Anexo E. Pruebas de concepto

Las pruebas de consepto son prototipos realizados con el propósito de comprobar o demostrar algo en relación con una idea de diseño o una tecnología.

E.1. Hadoop, Hive y Elasticsearch

Al ser el inicio de un desarrollo y la primera toma de contacto con varias de las herramientas implicadas, no suponía un esfuerzo mucho mayor introducir una más si esto ayudaba a simplificar su uso. Por ello se introdujo Docker como gestor de contenedores, para facilitar el despliegue y comunicación de los distintos elementos de la cadena.

De ese modo se desplegaba un contenedor con una imagen que contuviera HDFS, Hadoop y Hive en la misma máquina y con el plugin disponible a través de un volumen montado en el directorio de Hive; y otro con una imagen que contuviera Elasticsearch listo para recibir datos e indexarlos.

El primer paso fue definir la estructura que tendría y los datos que se usarían. Para una primera aproximación se decidió utilizar los *datasets* de Geonames (Dowload Site), los cuales estaban en un conveniente formato CSV y ofrecían la geometría más sencilla, el punto.

En el directorio vinculado al volumen se añadieron el fichero CSV con los datos a cargar y un nuevo documento que contenía el script HQL que se había programado para la creación de la tabla:

```
DROP TABLE IF EXISTS lake;

CREATE EXTERNAL TABLE lake (
  --- Columnas que se corresponden con los datos del fichero CSV ---
)
COMMENT 'Lake table'
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '\t'
STORED AS TEXTFILE
LOCATION '/user/geolake/lake';
```

Este fichero define una tabla externa que monta sus datos desde en el directorio de HDFS '/user/geolake/lake/'. Si el directorio no existe lo creará. Es por eso que el movimiento de los datos a HDFS lo hacemos después y no antes:

```
beeline -u jdbc:hive2://localhost:10000 -f /opt/geolake/HQL.sql
hdfs dfs -copyFromLocal /opt/geolake/spain.txt
/user/geolake/plaintext/spain.txt
```

Podremos comprobar que la tabla se ha generado correctamente al realizar una query sobre ella:

```
beeline -u jdbc:hive2://localhost:10000
select * from lake limit 10;
```

Para crear la conexión con Elasticsearch es necesario crear otra tabla externa que en lugar de estar soportada sobre HDFS, lo esté sobre Elasticsearch. Creamos un nuevo script HQL en el que se añada el plugin, se cree la tabla y mueva los datos de una tabla a otra:

```
ADD jar /opt/geolake/elasticsearch-hadoop.jar;
DROP TABLE IF EXISTS es_lake;

CREATE EXTERNAL TABLE es_lake (
  --- Columnas que se corresponden con los datos del fichero CSV ---
)
STORED BY 'org.elasticsearch.hadoop.hive.EsStorageHandler'
TBLPROPERTIES(
  'es.nodes' = '---Elasticsearch IP---',
  'es.resource' = 'geo/lake',
  'es.index.auto.create' = true,
);

INSERT OVERWRITE TABLE es_lake
SELECT
  --- Columnas que se corresponden con los datos del fichero CSV ---
FROM lake;
```

Con esto los datos estarían en Elasticsearch listos para ser buscados.

E.2. Hadoop, Hive y Elasticsearch con JDBC

El primer paso es añadir un tercer contenedor que contenga una imagen de Logstash, también con un volumen montado en algún punto del sistema de ficheros.

Repetimos el proceso anterior en Hive, eliminando el paso del uso del plugin y la creación de la segunda tabla. Con ello nos quedamos con una sola tabla en Hive exponiendo nuestros datos por JDBC.

En el directorio vinculado al volumen de Logstash, creamos la *receta*, que es como llama Logstash a sus ficheros de configuración:

```
input{
    jdbc_driver_library => "hive-jdbc.jar"
    jdbc_driver_class => "org.apache.hadoop.hive.jdbc.HiveDriver"
    jdbc_connection_string => "jdbc:hive2://hive"
    schedule => "* * * * *"
    statement => "select * from lake limit 10;"
}
output{
    elasticsearch{
       id => "es_geolake"
       hosts => ["elasticsearch:9200"]
       index => "geo"
       document_type => "lake"
       document_id => "%{id}"
       template_name => "geolake"
    }
    stdout{
       id => "stdout"
       codec => rubydebug
    }
}
```

El plugin de entrada de JDBC es configurado con los datos necesarios sobre la librería a usar y es programado mediante una sintaxis *cron-like* (Cron Sintaxis). Esto nos confirma, lo que en un principio se sospechaba con este prototipo, que el proceso ha de ser periódico y no puede dispararse manualmente. Al menos con esta aproximación.

Ya en este punto del desarrollo es fácil darse cuenta de que el rumbo que se ha tomado no va a llevar a la solución esperada de modo que se decide abortar la prueba y extraer las conclusiones.

E.3. Elastic Stack

Como se ha comentado, Elastic ofrece sus productos dockerizados desde su propio repositorio por lo que la composición se hace sencilla. Un contenedor para cada imagen y un volumen montado en cada uno de ellos y vinculado a un directorio local. De este modo cada aplicación tiene acceso a sus ficheros de configuración.

Beats, como responsable de la lectura, tendrá además otro volumen que estará vinculado al directorio donde se alojan. Como tiene que ser configurado para leer ese directorio, editamos el fichero *filebeat.yml* y establecemos los siguientes parámetros:

```
############ Filebeat Configuration Example ##################
#================ Filebeat prospectors ======================

filebeat.prospectors:
- input_type: log
  paths:
      - /opt/geolake/*

#====================== Outputs ==============================
#------------------- Logstash output ----------------------
output.logstash:
  hosts: ["logstash:5044"]
```

Hemos definido la ruta como "/opt/geolake/*" y el puerto de salida con la dirección de Logstash.

Para la configuración de Logstash modificaremos el fichero que ya creamos en el prototipo anterior sustituyendo el plugin de entrada de JDBC por el de Beats, configurando el puerto de escucha al establecido en la configuración de Beats. Además, introducimos un filtro que aplique una transformación sencilla para probar el potencial de filtrado:

```
input{
    beats{
       port => 5044
       codec => "plain"
    }
}
filter {
  json {
      source => "message"
       remove_field => ["message"]
  }
    ruby {
       code =>
"event.remove('beat');event.remove('offset');event.remove('input_type');
event.remove('type');event.remove('tags');"
    }
}
output{
    elasticsearch{
       id => "es_geolake"
       hosts => ["elasticsearch:9200"]
       index => "geo"
       document_type => "lake"
       document_id => "%{id}"
       template => "/usr/share/logstash/template/template.json"
       template_name => "geolake"
    }
    stdout{
       id => "stdout"
       codec => rubydebug
    }
}
```

Nótese que en el plugin de salida de Elasticsearch había configurada una plantilla en un directorio local que apunta a donde el volumen de Docker ha sido montado para este contenedor. Es ahí donde debemos crear el fichero de *mapping* para Elasticsearch. Éste define la estructura de datos que tendrá cada elemento al ser indexado. Los campos que no estén definidos pueden ser autogenerados:

```
{
    "template": "geo*",
    "settings": {
       "number_of_shards": 1
    },
    "mappings": {
       "lake": {
             "properties": {
                   "id": {
                         "type": "keyword"
                   },
                   "version": {
                         "type": "long"
                   },
                   "dataset": {
                         "type": "keyword"
                   },
                   "location": {
                         "type": "geo_point"
                   },
                   "info": {
                         "type": "object"
                   }
             }
       }
    }
}
```

Realizada la configuración, se puede lanzar el proceso de docker-compose que inicialice los contenedores. Beats espera a Logstash para transmitirle la información. Logstash la transforma y envía a Elasticsearch y ésta es visualizada en Kibana, el cual no necesita de ficheros de configuración (Figura 47 y Figura 48).

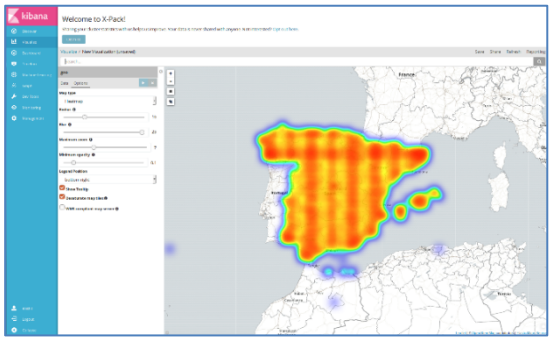


Figura 47. Panel geográfico de Kibana

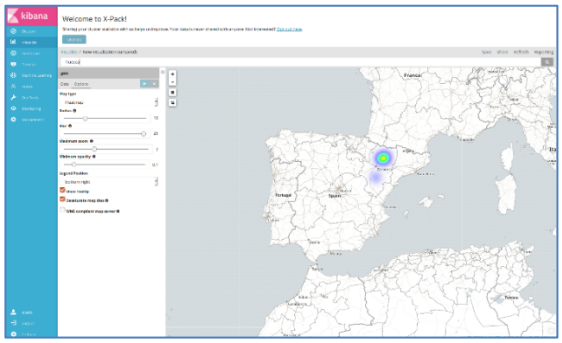


Figura 48. Panel geográfico de Kibana buscando por Huesca

E.4. Spring y Elasticsearch con Elastic Java API

Habitualmente un proyecto en Spring se empieza con un código base que sirve de plantilla para cada tipo de proyecto, el cual está disponible en los repositorios de Pivotal, y a partir de él se programan las funcionalidades. En este caso, al ser un programa de línea de comandos, la interfaz que implementamos es *CommandLineRunner*. De ese modo el comportamiento de Spring es el de una aplicación de CLI con todas las ventajas que ofrece el framework. Una de ellas y la más explotada es la inversión de dependencias, una función clave para aplicar arquitectura hexagonal.

¿Por qué arquitectura hexagonal? La arquitectura hexagonal permite abstraer el modelo de la implementación elevando la capa de infraestructura por encima de la capa de aplicación y rebautizándola como capa de adaptadores. Así tanto la aplicación como el modelo desconocen las implementaciones de la infraestructura lo que permite cambiarla y adaptarla fácilmente. En nuestro caso, por ejemplo, nos permite abstraer la lectura del fichero CSV en el modelo y programar toda la lógica para objetos del dominio. De ese modo para cada formato de origen existirá un adaptador en la capa de adaptadores, pero todos harán uso del mismo modelo.

Debido a la naturaleza de "trabajo en lote" que implica el problema, se opta por implementar un patrón de *pipes* productoras-consumidoras con colas bloqueantes (Figura 49). De ese modo cada elemento de la cadena consume un tipo de dato, lo procesa y emite un tipo igual o distinto al consumidor.

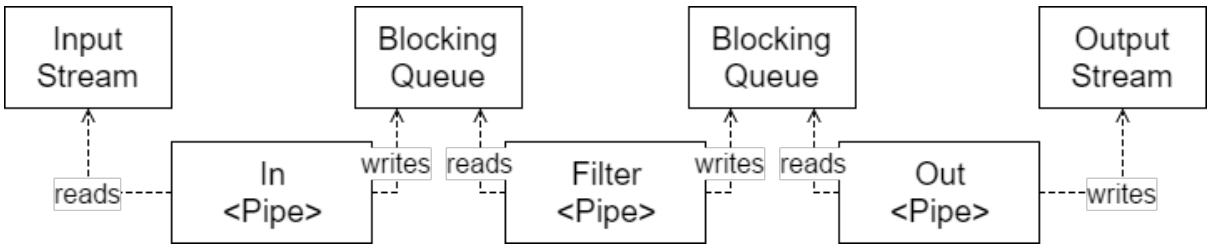


Figura 49. Esquema del funcionamiento del sistema de pipes

El funcionamiento es correcto, pero la API de Elasticsearch es el foco de la mayoría de los problemas. Si bien su uso es intuitivo para alguien habituado a trabajar con la herramienta, para un desarrollador que se enfrenta al sistema por primera vez resulta confusa debido al bajo nivel que ofrece/exige.

E.5. Spring y Elasticsearch con Spring Data

Este es el punto donde las decisiones de diseño brillan y se justifica el uso de una arquitectura flexible y unos principios robustos de programación orientada a objetos. El paso de usar la API de Elasticsearch a usar Spring Data es tan sencillo como:

- Añadir las dependencias de la nueva librería al proyecto
- Anotar las clases *Entidad* debidamente para indicar al framework cosas como el tipo de datos al que deben ser mapeadas
- Sustituir las clases que implementan los repositorios por interfaces que heredan de la clase *CrudRepository* y refactorizar los nombres de los métodos si estos no cumplieran con las directivas de Spring Data. Como ya se había trabajado con el framework anteriormente, los nombres usados para la implementación del prototipo anterior coincidían perfectamente.

Reemplazada la API pura por su abstracción, sólo queda retocar la lógica para que en lugar enviar los datos de uno en uno, los almacene en lotes en memoria y posteriormente los envíe en lote.

E.6. Spring, Elasticsearch y Searchkit

La forma más sencilla de iniciar un proyecto para React.js es con la herramienta *create-react-app*. Teniendo el Node Package Manager instalado es tan sencillo como ejecutar:

```
npm install -g create-react-app
create-react-app geolakeGUI
```

Esto creará un proyecto base de *Hello World* y nos permitirá empezar a programar rápidamente. Si bien en el repositorio de GitHub de Searchkit ofrecen un código de ejemplo, éste sólo ha sido utilizado para aprender sobre la herramienta y poder así decidir qué elementos incluir y cómo configurarlos.

Los elementos son esencialmente:

- Una barra buscadora
- Un espacio para los resultados
- Varios tipos de filtros distintos que actúan sobre las facetas
- Elementos que aportan funcionalidades como control de página o estadísticas
- Elementos visuales para organizar el espacio

Sin ni siquiera conocer cómo funcionan internamente, no es difícil configurarlos y hacer que encajen a la perfección con nuestro sistema. Solamente existe un fallo, no hay mapa.

El siguiente paso es el que realmente entraña complejidad. Comprender el funcionamiento interno de Searchkit para lograr crear un componente propio que interactúe correctamente con todo el sistema de búsqueda.

El principio de búsqueda básico de Searchkit es el de la *Inmutable Query*. Una cadena de *Query* básicas que juntas forman "la pregunta" que se está haciendo a Elasticsearch.

La *Immutable Query* es compartida por todos los *Accessor* de modo que cada uno puede crear parte de esa pregunta (Figura 50). Por ejemplo, un componente filtro tiene un *Accessor* propio que puede añadir o quitar de la *Immutable Query* el filtro para el que está configurado. De igual forma, la caja de búsqueda tiene un *Accessor* que añade la query textual cuando recibe un mensaje de texto.

En el lado opuesto a la realización de la consulta está el consumo del resultado para el cual existe una función *.getResults()* que devuelve los resultados de la última búsqueda. Para poder acceder a esa función es necesario extender al componente *SearchkitComponent*.

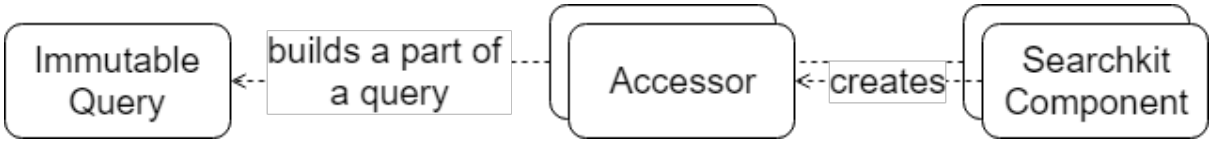


Figura 50. Relación entre la Immutable Query, los Accessor y los Component

Por lo tanto, sabemos que nuestro componente debe cumplir dos condiciones

- Para hacer un filtrado geoespacial: Tener un *Accessor* que genere su parte de la Query en la cual defina el área a filtrar.
- Para visualizar los resultados: Extender al componente *SearchkitComponent* para tener acceso a estos.

Como no existe tampoco ningún Accessor que ofrezca geofiltrado hay tenemos que implementar uno basándonos en uno ya existente. La Query que incluirá será:

```
filter:{
    geo_bounding_box:{
        location:this.area
    }
}
```

Finalmente, el desarrollo del componente *Map*, el cual extiende a *SearchkitComponent*, debe instanciar el *Accessor* en su método *.defineAccessor()* y mostrar un mapa en su método *render()*.

Para el propio mapa se ha decidido utilizar la versión para React.js de la herramienta Leaflet, una librería JavaScript de código abierto que permite mostrar mapas interactivos. Es una de las más populares y ofrece una gran cantidad de complementos que extienden su funcionamiento.

Habiendo añadido al componente *Map* todo lo que necesita, sólo queda escribir la lógica de tratamiento de los resultados y mostrarlos sobre el mapa usando las herramientas que Leaflet ofrece, así como algún *plug-in* visual como *react-leaflet-markercluster* que al cual se le puede delegar una nube de puntos y éste la representa agrupando los puntos que están muy próximos.