UNIVERSIDAD AUTÓNOMA DE GUADALAJARA

Prototipo de Plataforma Estructurada para el Acceso a Sistemas de Transparencia Gubernamental

presentado por

Esperanza Citlali García Medina

Para obtener el título de
Maestría en Ciencias Computacionales

en la
Facultad de Electrónica y Ciencias Computacionales

21 de abril de 2015

"La educación es el arma más poderosa que puedes usar para cambiar el mundo."

Nelson Mandela

RESUMEN

En este trabajo se realizó una síntesis de los principales conceptos relacionados con Conocimiento Abierto, tales como Datos Abiertos y Contenido Abierto, a su vez se repasó la historia de como ha evolucionado el movimiento Datos Abiertos a través de los años. Los Gobiernos ahora están liberando información que solía ser privada con el objetivo de ser mas transparentes y eficientes, y para hacerlo, muchos de ellos están aprovechando las tecnologías ya existentes tales como la plataforma CKAN, mantenida por la Fundación Conocimiento Abierto. Sin embargo, el proceso de consulta y análisis de la información es aún muy compleja. Dentro de este documento se discuten algunos de los problemas relacionados con datos abiertos y se sugiere un enfoque diferente a la búsqueda de información no estructurada a través de la implementación de una aplicación de búsqueda basada en Apache Lucene.

Palabras Clave:

Conocimiento Abierto, Datos Abiertos, Datos de Gobierno, Indexado, Motor de Búsqueda.

ABSTRACT

In this work, a summary has been done of the main concepts related to Open Knowledge such as Open Data and Open Content, also an overview of the history about how the Open Data movement has evolved through the years has been covered. Governments are now releasing data so as to be more transparent and efficient, and in order to do so, many of them are taking advantage of the technologies that already exist such as the CKAN platform maintained by the Open Knowledge Foundation. However the query and analysis processes are still very complex. Within this document some of the issues related to open data are discussed and a different approach to the search of unstructured data is suggested through the implementation of a search application based on Apache Lucene.

Keywords:

Open Knowledge, Open Data, Government Data, Indexing, Search Engine.

Agradecimientos

A mi asesor de tesis, el Dr. Rogelio Dávila por sus consejos en el desarrollo de este proyecto.

A CONACYT por la beca que hizo posible mi estudio de posgrado.

A mis maestros por todas sus enseñanzas.

A mi marido, Romeo por todo su apoyo y comprensión en mis noches de desvelo y estrés.

A mi familia, por estar ahí siempre.

Índice general

Resumen II

Agradecimientos III

Índice de Figuras VI

Índice de Cuadros VIII

Abreviaciones IX

1. Introducción 1
1.1. Datos Abiertos de Gobierno . . . 3

2. Descripción del Problema 5

3. Definición del Problema 8

4. Pregunta de Investigación 9

5. Objetivos de la Investigación 10
5.1. Objetivo General . . . 10
5.2. Objetivos Particulares . . . 10

6. Resultados Esperados 11

7. Delimitación de la Investigación 12

8. Justificación 13

9. Bases Teóricas 15
9.1. Marco Historico y Contextual . . . 15
9.2. Marco Referencial . . . 17
9.3. Términos Básicos . . . 18

10. Propuesta de Solución 19
10.1. Selección de un Conjunto de Datos . . . 19
10.2. Recuperación de datos usando la API de CKAN . . . 21
10.3. Arquitectura de la Aplicación . . . 27

11. Implementación de la Solución 29
11.1. Configuración de Ambiente . . . 29

11.1.1. Instalación de Apache Lucene 29
11.1.2. Instalación del módulo de Python para acceder la API de CKAN . 30
11.1.3. Instalación de Apache Tika 31
11.2. Detalles Técnicos 32
11.2.1. Adquirir Contenido 34
11.2.2. Extraer Texto 39
11.2.3. Construir Documentos 40
11.2.4. Analizar los Documentos 41
11.2.5. Indexar los Documentos 42
11.2.6. Construir Consulta 43
11.2.7. Ejecutar Consulta 44
11.2.8. Mostrar Resultados 45
11.3. Instrucciones para compilar el código fuente 46

12.Resultados Obtenidos **47**
12.1. Resultados de la ejecución del proceso de Indexado 47
12.2. Resultados de la ejecución del proceso de Búsqueda 49

13.Conclusiones y Trabajo a Futuro **51**

A. Anexo - Resumen de formatos de archivos más comunes **52**

B. Anexo - Calendario de Actividades **53**

C. Anexo - Aplicación de Búsqueda (Código Fuente) **54**

D. Anexo - Ejemplos de implementaciones CKAN en el mundo **60**

Bibliografía **70**

Índice de figuras

8.1. Los datos abiertos pueden ayudar a generar mas de 3 billones de dólares al año en el mundo. 13

9.1. Catálogo de Datos Abiertos del Gobierno de México. 16

10.1. Ejemplo de consulta CKAN para obtener el listado de paquetes 21
10.2. Ejemplo de consulta CKAN para obtener el listado de etiquetas 22
10.3. Ejemplo de consulta CKAN para obtener el listado de organizaciones . . . 23
10.4. Ejemplo de consulta CKAN para obtener los detalles de una organización 24
10.5. Ejemplo de consulta CKAN para obtener los detalles de un conjunto de datos 25
10.6. Ejemplo de consulta CKAN para obtener los detalles de un recurso 26
10.7. Arquitectura Típica de una Aplicación de Búsqueda. 27

11.1. Ejemplo de solicitud remota utilizando la librería ckanapi en Python . . . 30
11.2. Ejemplo de respuesta a solicitud remota utilizando la librería ckanapi en Python 30
11.3. Interfaz Gráfica de la herramienta Apache Tika. 31
11.4. Clases del proceso de Indexado 32
11.5. Clases del proceso de Búsqueda 32
11.6. Arquitectura Prototipo de Plataforma Estructurada para el Acceso a Sistemas de Transparencia Gubernamental. 33
11.7. Código fuente de la clase CkanDatosAbiertosMX 35
11.8. Código fuente de el módulo test_ckandgm.py (Parte 1) 36
11.9. Código fuente de el módulo test_ckandgm.py (Parte 2) 37
11.10Directorio de Documentos Abiertos extraídos de una plataforma CKAN . 38
11.11Firma del método parse en la librería Tika 39
11.12Bloque de código: Extraer Texto 39
11.13Bloque de código: Construir un Documento 40
11.14Bloque de código: Analizar un Documento 41
11.15Bloque de código: Indexar un Documento 42
11.16Bloque de código: Construir Consulta 43
11.17Bloque de código: Ejecutar Consulta 44
11.18Bloque de código: Mostrar Resultados 45
11.19Compilar las clases de Java 46

12.1. Ejecutar la clase TikaIndexer.java 47
12.2. Crear índices de los archivos existentes en un directorio. 48
12.3. Ejecutar la clase Searcher.java 49

12.4. Resultados de Búsqueda. Query = pemex 49
12.5. Ejemplo de Búsqueda. Query = 'indicadores+pobreza modified:[1/1/15 TO 12/31/15] -multidimensional': 50

C.1. Código Fuente para clase Indexer.java (Parte 1) 54
C.2. Código Fuente para clase Indexer.java (Parte 2) 55
C.3. Código Fuente para clase TikaIndexer.java (Parte 1) 56
C.4. Código Fuente para clase TikaIndexer.java (Parte 2) 57
C.5. Código Fuente para clase Searcher.java (Parte 1) 58
C.6. Código Fuente para clase Searcher.java (Parte 2) 59

D.1. Datos Abiertos del Gobierno de Amsterdam. 60
D.2. Datos Abiertos del Gobierno de África. 61
D.3. Datos Abiertos del Gobierno de Australia. 62
D.4. Datos Abiertos del Gobierno de Berlin. 63
D.5. Datos Abiertos del Gobierno de Brasil. 64
D.6. Datos Abiertos del Gobierno de Buenos Aires. 65
D.7. Datos Abiertos del Gobierno de Montreal. 66
D.8. Datos Abiertos del Gobierno de Montreal. 67
D.9. Datos Abiertos del Gobierno del Reino Unido. 68
D.10. Datos Abiertos del Gobierno de Estados Unidos de América. 69

Índice de cuadros

10.1. Variables para fórmula de cálculo de el tamaño de muestra 19
10.2. Valores reales para fórmula de cálculo de el tamaño de muestra 20
10.3. Selección de tamaño de muestra por tipo de formato 20

Abreviaciones

API	**A**pplication **P**rogram **I**nterface
CKAN	**C**omprehensive **K**nowledge **A**rchive **N**etwork
ODM	**O**pen **D**ata **M**ovement
OKF	**O**pen **K**nowledge **F**oundation
URL	**U**niform **R**esource **L**ocator

Introducción

En la última década, un movimiento ha surgido gradualmente en todo el mundo, conocido como el movimiento de los Datos Abiertos. Su objetivo es la difusión del Conocimiento Abierto en su mas amplio sentido.

La Fundación Conocimiento Abierto define el conocimiento abierto como:

> «cualquier contenido, información o dato que puede ser libremente utilizado, reutilizado y redistribuido -sin restricciones legales, tecnológicas o sociales-. [...] El conocimiento abierto es en lo que se convierten los datos abiertos cuando son útiles, usables y utilizados» [1]

Los Datos Abiertos son por ende, la base de construcción en la que se fundamenta el Conocimiento Abierto.

El término de Datos Abiertos es una práctica que tiene como objetivo que determinados tipos de datos estén disponibles para todo el mundo sin restricciones de derechos de autor, patentes u otro mecanismo de control. Dicho concepto no es nuevo, sin embargo, ha ganado mayor relevancia con el aumento en la popularidad de tecnologías como el Internet y la Red Informática Mundial (WWW), así como a la acumulación masiva de datos.

La filosofía detrás de éste movimiento se centra en 3 principales aspectos:

1) Disponibilidad y Accesso: Los datos deben de estar disponibles preferentemente mediante una descarga de Internet. Los datos deben de ser presentados en un formato conveniente y que permita modificaciones fácilmente.

2) Reutilización y Redistribución: Los datos deben de estar disponibles bajo términos que permitan su reutilización y redistribución, incluida la composición con otros conjuntos de datos. Los datos deben de ser capaces de ser leídos por una computadora.

3) Participación Universal: Cualquier persona debe de tener la posibilidad de usar, reutilizar y redistribuir datos abiertos sin discriminación alguna.

Los tipos de Datos Abiertos comprenden campos de estudio muy diversos como lo son Cultural, Científico, Financiero, Estadístico, Climatológico, Ambiental, de Transporte y Gubernamental.

1.1. Datos Abiertos de Gobierno

Los Datos Abiertos de Gobierno son un proyecto de la Fundación Conocimiento Abierto enfocada exclusivamente a la publicación de datos producidos por instituciones gubernamentales. Varios países incluidos los Estados Unidos Americanos, Reino Unido, Canada, Nueva Zelanda y recientemente México han anunciado sus propias iniciativas hacia la apertura de su información. [2–7]

Ya es posible señalar una gran cantidad de áreas donde los Datos Abiertos de Gobierno están creando valor. Algunos ejemplos de ellos son: [8, 9]

- En términos de transparencia, proyectos como el Finlandés 'árbol de impuestos' o el Británico '¿A donde va mi dinero?' nos muestran como se gasta el dinero de los impuestos por el gobierno. También páginas web como la página Danes folketsting.dk permite monitorear las actividades en el parlamento y el proceso de construcción de leyes para poder saber exactamente qué esta sucediendo y que representantes están involucrados.

- Los Datos Abiertos de Gobierno también pueden ayudar a tomar mejores decisiones en la vida personal, o habilitar a las personas para tener una participación más activa en la sociedad. Una mujer en Dinamarca construyó el sitio web findtoilet.dk para mostrar todos los baños públicos y ayudar a las personas con problemas de incontinencia a salir nuevamente. En Holanda existe un servicio vervuilingsalarm.nl que te advierte con un mensaje de la calidad de el aire en tu comunidad. En Nueva York se puede encontrar fácilmente donde sacar el perro a pasear así como conocer otras personas que frecuentan el mismo parque. Servicios como 'mapumental' en Reino Unido y 'Magnificent' en Alemania te permiten encontrar un lugar para vivir teniendo en consideración la duración de el traslado al trabajo, los precios de las casas y la calidad de el area. Todos estos ejemplos utilizan datos abiertos.

- Económicamente, los datos abiertos también son de gran importancia. Varios estudios han estimado el valor de los datos abiertos es superior a los diez mil millones de Euros anualmente. Nuevos productos y compañías están utilizando esta información por ejemplo, la página web Danesa husetsweb.dk te ayuda a encontrar manera

de mejor la eficiencia energética en el hogar basado en información de los subsidios gubernamentales. Google Translate utiliza la enorme cantidad de documentos abiertos para entrenar sus algoritmos de traducción y por lo tanto mejorando la calidad de su servicio.

- Los datos abiertos son de gran valor para el gobierno mismo ya que colabora a incrementar su eficiencia y efectividad. Por ejemplo el Ministerio de Educación Alemán ha publicado toda la información relacionada con educación en linea para su reutlización. Desde entonces, el número de preguntas que ellos reciben a disminuido reduciendo la carga de trabajo y costos.

Aún cuando ya hay numerosos ejemplos de las maneras en que los datos abiertos están creando valor tanto social como económico, todavía hay un área muy grande de oportunidad para descubrir nuevos campos de investigación.

Los sitios web que forman parte de el proyecto Datos Abiertos de Gobierno están desarrollados utilizando una plataforma tecnológica conocida como CKAN, también mantenida por la Fundación Conocimiento Abierto. CKAN es un sistema para el almacenamiento y distribución de información que provee herramientas para publicar, compartir, buscar y utilizar los datos. CKAN esta enfocado a gobiernos regionales y nacionales, compañías y organizaciones que quieren hacer que sus datos sean abiertos y disponibles al público. [10]

Éste documento investiga la estructura e implementación de la plataforma CKAN en diferentes instituciones gubernamentales, particularmente en la plataforma del Gobierno Mexicano, sin embargo será aplicable para cualquier otra institución que este desarrollada utilizando las mismas tecnologías, con la finalidad de proponer una nueva herramienta capaz de extraer la información proporcionada por las instituciones gubernamentales y estructurarla de una manera uniforme utilizando en un formato genérico y permitir así la consulta y análisis de la información de una manera más simple y natural.

Descripción del Problema

Los catálogo de Datos Abiertos permiten que se creen herramientas de todo tipo para medir y estudiar lo que ocurre a partir de la información recolectada. Nuevas tecnologías permiten ahora construir servicios para responder automáticamente preguntas como: ¿Cuál es el eje carretero más largo en un país? ¿Qué porcentaje del presupuesto se destina para el alumbrado público? ¿En que región existen mejores oportunidades de trabajo? Mucha de la información necesaria para responder estas preguntas es generada por instituciones públicas, sin embargo, frecuentemente la información no esta disponible en un formato que sea fácil de utilizar.

En México, en el año 2014, como parte de la Política de Datos Abiertos, el Gobierno Mexicano ha puesto a disposición de toda la población la mayor cantidad de información posible de todo lo que ocurre en las entidades de Gobierno y con cada programa social que se desarrolla en el país a través de la plataforma llamada datos.gob.mx. De acuerdo con el comunicado oficial del Gobierno Federal, desde dicha página se puede «acceder, descargar, y utilizar libremente los datos abiertos que el Gobierno de la República genera y recolecta, con el fin de innovar, emprender, incrementar la transparencia, eficientar al gobierno y promover la innovación cívica.» [11]

Una investigación realizada en la Universidad de Waterloo en Canada en el año de 2014 analiza diferentes casos de uso de sistemas que fueron implementados utilizando datos abiertos y refleja la complejidad que implica el trabajar con ellos: [12]

- Hay situaciones en las que para poder hacer un análisis se requiere información tanto de instituciones gubernamentales como no gubernamentales. ¿Cómo aseguramos que toda la información necesaria para un proyecto se encuentra disponible?
- Aplicaciones que requieren información de ultimo momento, como una aplicación para predecir el clima, requiere de la información este lo más actualizada posible.

¿Cómo podemos asegurar que la información requerida aún es válida y que sea actualizada en un periodo aceptable de tiempo?

- Algunas veces es necesario consultar información no propia del documento pero de su publicación, por ejemplo, lugar y fecha de publicación ¿Cómo puedo acceder a herramientas que permitan consultar matadatos y palabras clave de un documento?

- Liberar datos abiertos puede tener efectos en la privacidad de los individuos afectados cuando alguna información personal pudiera ser inferida por datos abiertos. ¿Cómo puedo asegurar que los datos abiertos que han sido compartidos no implican potenciales problemas de seguridad o privacidad?

Idealmente para un usuario debería ser posible fácilmente realizar tareas tales como: [8]

- Descubir la existencia de determinado conjunto de datos.
- Acceder a los datos para su investigación y análisis.
- Encontrar la información detallada describiendo la información y su proceso de producción.
- Acceder a las fuentes de datos e instrumentos con los cuales la información fue colectada.
- Efectivamente comunicarse con las agencias involucradas en el proceso de producción, almacenamiento y distribución de la información.
- Compartir el conocimiento con otros usuarios.

Sin embargo, la realidad aún se aleja de los ideales de el Movimiento Conocimiento Abierto. No es suficiente con declarar los conjuntos de datos como 'abiertos' para que estos datos tenga un uso práctico para el ciudadano común.

Cuando estos datos son liberados en su formato crudo (raw en Inglés), sólo tienen sentido para los especialistas técnicos que saben como extraerlos, interpretarlos y utilizarlos. De nuevo, esta información sigue estando únicamente a disposición de ciertos grupos privilegiados.

Si se pretende que esta información tenga un alcance masivo y sea significativa para la ciudadanía en general, debemos de enfocarnos en solucionar los principales retos a los que se enfrenta un usuario es su búsqueda de datos abiertos los cuales son:

- La navegación en la plataforma no es intuitiva para un usuario que no este familiarizado con entornos tecnológicos.
- Los formatos en que los datos son presentados son muy diversos, varía desde archivos en formatos ZIP, CSV, XML, JSON, KMZ, etc.
- El motor de búsqueda integrado a la plataforma es poco eficiente por lo que un usuario debe saber de antemano cual es la fuente que esta generado los datos de su interés, de otra manera tiene que recurrir a la búsqueda mediante etiquetas que no siempre pueden estar disponibles.

Definición del Problema

El presente trabajo acota la investigación a un ejemplo de implementación de plataforma CKAN en particular, siendo el sitio web de Datos Abiertos del Gobierno Mexicano (datos.gob.mx) el seleccionado por ser un ejemplo práctico y de mayor beneficio para la comunidad local.

Cabe destacar que aún cuando la aplicación esta enfocada a esta plataforma, aplica prácticamente a cualquier otra que cuente con la misma estructura, por lo que este prototipo puede ser reutilizable y ajustarse a otras necesidades de implementación.

Con ésta aplicación se pretende contribuir a el desarrollo de herramientas que pongan a el alcance de todo el mundo la información pública que las instituciones gubernamentales proporcionan.

Pregunta de Investigación

¿Cómo desarrollar una aplicación que permita generar valor a partir de la consulta, uso y análisis de los datos abiertos liberados por las organizaciones gubernamentales?

Objetivos de la Investigación

5.1. Objetivo General

Desarrollar una aplicación que permita extraer los datos abiertos disponibles en una plataforma de datos abiertos de instituciones gubernamentales y que una vez obtenidos, los diferentes tipos de archivos que existen en diversos formatos sean estructurados uniformemente utilizando un formato único, de tal manera que puedan ser consultados y analizados de una manera más simple y natural.

5.2. Objetivos Particulares

1. Desarrollar un programa que inspeccione las páginas de una manera metódica y automatizada (Web Crawler en Inglés) para navegar a través del sitio de Datos Abiertos del Gobierno Mexicano (datos.gob.mx) y descargar la información requerida utilizando el API de CKAN mediante la ejecución de un script en lenguaje Python.

2. Transformar los documentos obtenidos de la plataforma CKAN en Documentos de formato único para su representación uniforme utilizando la herramienta Apache Tika.

3. Proveer las capacidades de búsqueda de texto y definir un lenguaje de consulta utilizando las capacidades que la herramienta Apache Lucene proporciona.

Resultados Esperados

Como producto del desarrollo de esta tesis y su proyecto de intervención se espera obtener una aplicación que aporte novedades en el ámbito de sistemas de búsqueda orientado a archivos utilizando un método de búsqueda avanzada basado en Apache Lucene y empleando como fuente de información los datos abiertos liberados por Instituciones Gubernamentales.
Dado que la aplicación estará desarrollada en su totalidad con herramientas de código abierto y diseñada para alimentarse con información de datos abiertos, es natural que la aplicación misma también sea distribuida libremente por lo que se espera que la implementación y el código fuente este disponible en un repositorio público al acceso de cualquier persona.
Así mismo, se espera que la aplicación pueda ser publicada en un servidor público y gratuito para que este disponible a la ciudadanía en general.
Los requerimientos básicos para la implementación de la aplicación comprenden:

- La navegación en la plataforma deberá ser sencilla e intuitiva para cualquier usuario.
- Los formatos en que los datos son procesados deberá ser uniforme.
- El motor de búsqueda deberá permitir consultas tanto por metadatos, etiquetas y al propio contenido de un documento.

Alcanzar estos objetivos contribuirá a un cambio cultural al contribuir a mejorar la capacidad de comprensión de la información, así como la habilidad de encontrar, utilizar y compartir datos y alentar la colaboración entre gobiernos, negocios y sociedad civil para combatir los principales retos de la actualidad.

Delimitación de la Investigación

A la fecha, la única manera de descargar datos de una plataforma CKAN, es a través de su propia API. No hay un servicio web que permita descargar los datos como conjunto, si no que tienes que recorrer la lista de resultados uno a uno. Hay 3 maneras en las que se puede filtrar los resultados: por paquete, por grupo y por etiquetas.

Debido a esta restricción, para los propósitos de construir un prototipo, se comenzará a trabajar con un subconjunto de datos inicial en los formatos XML, XLS, CSV, TXT y JSON que será una copia no actualizada de los datos en el sistema, por lo que eventualmente no serán sincronizados a la última versión disponible.

La investigación no comprende el proceso de implementar un sitio web basado en la plataforma CKAN, sólo comprende la tarea de cómo consumir la información de una plataforma ya existente y procesarla para mejorar el proceso de búsqueda.

Los archivos que contienen imágenes (formatos PNG, JPG and GIF) no serán incluidos como parte de éste estudio.

El proyecto de intervención es un prototipo que será mejorado de manera incremental en futuras versiones.

Justificación

La compañía McKinsey's publicó un estudio en Octubre de 2013 [13] en el que valora el potencial de los Datos Abiertos en términos económicos de siete sectores de actividad: educación, transporte, productos de consumo, energía eléctrica, petróleo, gas, salud y economía familiar.

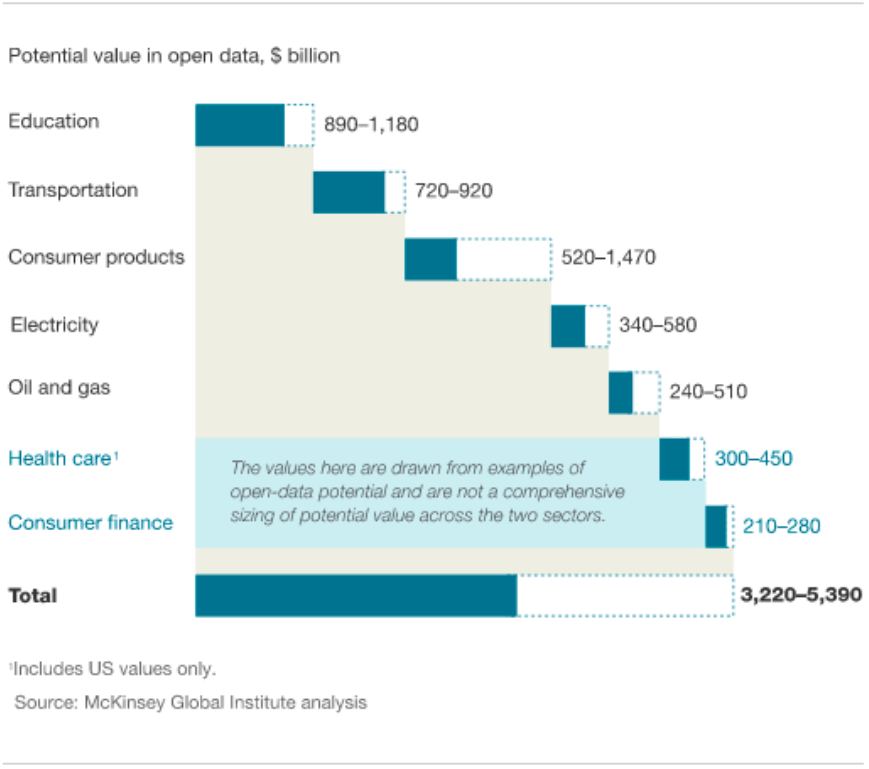


FIGURA 8.1: Los datos abiertos pueden ayudar a generar mas de 3 billones de dólares al año en el mundo.

La investigación sugiere que esos siete sectores exclusivamente pudieran generar mas de 3 billones de dólares al año de los cuales 1,1 billón corresponde a Estados Unidos, 900.000

millones a Europa y los otros 1,7 billones al resto del mundo como resultado de la utilización de datos abiertos.

Otro punto interesante en el estudio, revela cómo se ha incrementado el número de iniciativas de portales de datos abiertos así como el número de conjuntos de datos disponibles en cada portal. Por ejemplo, en 2009, el Gobierno de Estados Unidos publicó los primeros 47 conjuntos de datos y avanzó rápidamente hasta llegar a más de 90000 en Octubre de 2013. El Gobierno del Reino Unido alcanza los 10000 conjuntos de datos para esa misma fecha.

Por su parte, México se integró a esta iniciativa en Abril de 2014 y a la fecha en Abril de 2015 cuenta con un total de 358 conjuntos de datos.

Estadísticas como la anterior está dando pie para la creación de nuevos negocios y ayudando a compañías ya establecidas a explorar nuevos segmentos de mercado, definir nuevos productos y servicios y mejorar la eficiencia y efectividad de sus operaciones.

Aún cuando el fenómeno de datos abiertos aún esta en sus inicios en nuestro país, podemos observar un claro potencial para utilizar la información como un instrumento para ayudar a las instituciones gubernamentales a reemplazar la toma de decisiones tradicional por un enfoque orientado a datos. El análisis de datos abiertos también puede ayudar a descubrir preferencias de los ciudadanos, permitiendo a las instituciones mejorar sus servicios y descubrir anomalías y problemas.

Con base en esta información es que desarrollar una aplicación enfocada a facilitar el proceso de búsqueda y análisis de los datos abiertos disponibles prueba ser de gran conveniencia y viabilidad en nuestro entorno global actual.

Bases Teóricas

9.1. Marco Historico y Contextual

El término datos abiertos apareció por primera vez en 1995, en un documento de la Agencia Científica Americana. Ahí se discutía acerca de la difusión de información geofísica y ambiental y promovía una apertura de el intercambio de información científica entre diferentes países como requisito para el análisis y la comprensión de éstos fenómenos globales.

Pero fue hasta que los ideales científicos se cruzaron con los ideales del software libre, que surgió el concepto tal y como lo conocemos hoy en día.

En Diciembre de 2007, en Sebastopol, California, un grupo de 30 líderes entre ellos Tim O'Reilly and Lawrence Lessig [14] se reunieron para discutir y definir el concepto de datos públicos y abiertos así como los principios que permiten evaluarlos. Los resultados hasta la fecha han excedido las expectativas. En Enero de 2009, el Presidente Barack Obama tomo el mando de los Estados Unidos de América y firmó el Memorando para la Transparencia y el Gobierno Abierto, declarando « La información mantenida por el Gobierno Federal es un recurso nacional. Mi administración tomará las acciones apropiadas, consistentes con la ley, para difundir la información rápidamente en formatos que el público pueda rápidamente consultar y utilizar » [15]

En Mayo de ese mismo año, Data.gov iniciaba operaciones con únicamente 47 conjuntos de datos. Ahora, 6 años después la plataforma contiene más de 100,000 conjuntos de datos de 227 agencias y organizaciones locales, estatales y federales.

Por su parte en México, El 20 de septiembre de 2011, el Presidente de la República Mexicana presentó ante la Alianza para el Gobierno Abierto, el plan de acción de México. Los gobiernos que se incorporen a la Alianza para el Gobierno Abierto elaborarán un Plan de Acción que contenga compromisos concretos. En base a esos compromisos se lanzó el Datatron, una encuesta digital para conocer más sobre la demanda de Datos Abiertos y abrir el proceso a una mayor participación. El Datatron pedía a los encuestados elegir qué datos de gobierno le gustaría conocer, como resultado de dicha encuesta es que surge la plataforma de Datos Abiertos del Gobierno Mexicano (datos.gob.mx)
Es así que el estudio se desarrolla dentro de un contexto en donde el Movimiento Datos Abiertos ha alcanzado amplios niveles de popularidad dentro de los gobiernos regionales y nacionales. El proyecto de intervención comprende el total de 358 conjuntos de datos abiertos publicados desde Abril del 2014 a Abril del 2015 en la plataforma de Datos Abiertos del Gobierno Mexicano (datos.gob.mx)

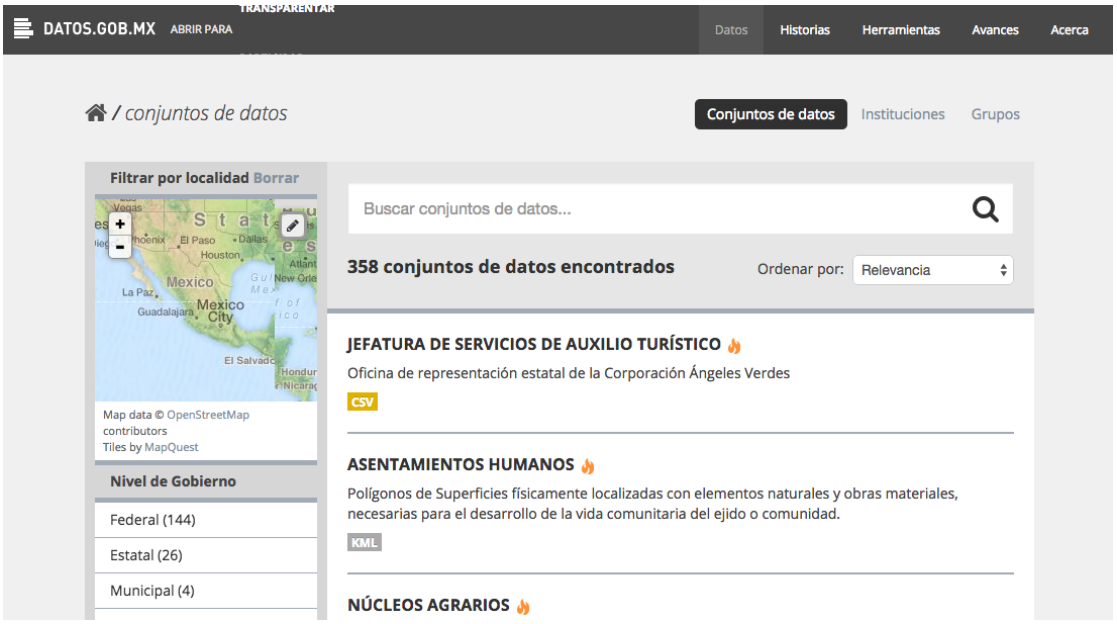


Figura 9.1: Catálogo de Datos Abiertos del Gobierno de México.

9.2. Marco Referencial

La propuesta de estudio esta fundamentada tecnológicamente en una librería de código abierto para búsqueda y recuperación de la información conocido como Lucene.
Lucene fue originalmente escrita en en lenguaje de programación Java por Doug Cutting (creador también de Hadoop) en 1997. [16] En el año 2000, Lucene se convirtió en una librería de código abierto alojada en SourceForge y rápidamente ganó adeptos por lo que en 2001 fue adoptaba por la Fundación Apache y para Febrero de 2005 ya se había convertido en uno de los principales proyectos de Apache . El número de contribuyentes creció sostenidamente, y a través de los años, Lucene ha sido traducido a múltiples lenguajes de programación, incluidos C++, C#, Perl, y Python. En la versión original de Java o en cualquiera de sus otras versiones, Lucene es ampliamente usado en el mundo. Ejemplos de sus aplicaciones incluyen [17]:

1. Bixee - Motor de Búsqueda de empleos en India.
2. Casabuscador - Motor de Búsqueda para la renta y venta de bienes raíces en España.
3. CiteSeerX - Motor de Búsqueda de documentos académicos.
4. CodeCrawler - Motor de Búsqueda de código fuente.
5. Dialo.de - Motor de Búsqueda para el directorio de servicios sección amarilla para Alemania.
6. Doctoralia - Motor de Búsqueda para profesionales de la salud.
7. Eclipse - Motor de Búsqueda para la documentación dentro de el entorno gráfico (IDE) de Eclipse.
8. Netflix - Motor de Búsqueda para la aplicación de reproducción de películas y series.
9. Keljob.com - Motor de Búsqueda de curriculum vitae.
10. Twitter Trends - Motor de Búsqueda para la herramienta que analiza las tendencias en Twitter.

9.3. Términos Básicos

Apache Lucene:

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

Apache Tika:

La herramienta Apache Tika es utilizada para extraes metadatos y texto de más de mil formatos de archivos diferentes (PPT, XLS, PDF, etc). Tika es muy util para el indexado de motores de búsqueda, análisis de contenido y traducción de documentos.

Araña Web:

También conocida como Web Crawler por su término en Inglés es un programa que inspecciona las páginas web de forma metódica y automatizada. Uno de los usos más frecuentes que se les da consiste en crear una copia de todas las páginas web visitadas para su procesado posterior por un motor de búsqueda que indexa las páginas proporcionando un sistema de búsquedas rápido.

Indexado:

Acción de agregar una o más páginas web ó documentos a las bases de datos de un buscador, para que estas aparezcan en los resultados de búsquedas de los mismos.

Metadatos:

Son datos que describen otros datos. El uso de los metadatos mencionado más frecuentemente es la refinación de consultas a buscadores. Usando información adicional los resultados son más precisos, y el usuario se ahorra filtraciones manuales complementarias.

Recuperación de la Información:

Proceso de búsqueda de documentos, información dentro de documentos o metadatos de los documentos.

Propuesta de Solución

10.1. Selección de un Conjunto de Datos

Hasta Abril de 2015 se reportan un total de 358 conjuntos de datos disponibles en el sitio web http://catalogo.datos.gob.mx/dataset

El tamaño de la muestra seleccionada para este estudio fue calculado con base en los procedimientos sugeridos en la teoría del muestreo y probabilidad. Las variables a considerar son las siguientes:

Variable	Descripción
n	Tamaño de la muestra
N	Tamaño del universo
p	Probabilidad de ocurrencia
q	Probabilidad de no ocurrencia
Me	Margen de error o precisión
Nc	Nivel de confianza o exactitud

CUADRO 10.1: Variables para fórmula de cálculo de el tamaño de muestra

$$n = \frac{Npq}{\left[\frac{Me^2}{Nc^2}(N-1)\right] + PQ}$$

Variable	Descripción
N	358
p	.5 (Mayor punto de incertidumbre 50 %)
q	1 - .5 = .5
Me	5 (Margen de error o precisión +/- 5 %)
Nc	1.96 (95 % de nivel de confianza o exactitud)

CUADRO 10.2: Valores reales para fórmula de cálculo de el tamaño de muestra

Al substituir la fórmula obtenemos un tamaño de muestra recomendado de: **186**

La distribución de los documentos de la muestra, organizados por tipo de formato e institución que lo genera, fue seleccionado de acuerdo a la disponibilidad de los mismos y se distribuye de la siguiente manera:

Cantidad	Formato	Instituciones
CSV	89	PEMEX, SAGARPA
XML	55	PEMEX
JSON	3	PROMEXICO
XLS	2	SHCP
TXT	37	SEDESOL

CUADRO 10.3: Selección de tamaño de muestra por tipo de formato

10.2. Recuperación de datos usando la API de CKAN

Desarrolladores que desean escribir código que interactue con sitios CKAN y su información deben de hacer uso de la API, cualquier operación que un usuario pueda hacer con la interfaz web es posible realizarla utilizando invocaciones a través de código. La lista completa de operaciones soportadas puede ser consultada en el sitio web de CKAN. [18]

- Para realizar una solicitud a la API de CKAN, es necesario hacer una invocación de tipo HTTP a la URL donde la información se encuentre disponible.
- Los parámetros de consulta deberán ser proporcionados es un diccionario en formato JSON en caso de existir.
- La respuesta también será regresada en un diccionario en formato JSON.

La URL en donde se realizan las consultas para obtener información de la plataforma de Datos Abiertos del Gobierno Mexicano es http://catalogo.datos.gob.mx/
A continuación se ejemplifica una serie de consultas que pueden ser realizadas sobre una plataforma CKAN a través de su API y que son las que se emplean para el desarrollo de la aplicación aquí presentada:

1. Consultar la lista de los nombres de los conjuntos de datos (paquetes).
   http://catalogo.datos.gob.mx/api/3/action/package_list

```
{
    "success": true,
    "result": [
    "actividad-hotelera-por-entidad-federativa",
    "agencias-del-ministerio-publico-del-estado-de-puebla",
    "apoyos-otorgados-en-2013-sagarpa",
    "asegurados-en-el-imss",
    "avisos-de-ciclon-tropical",  "..." ]
}
```

FIGURA 10.1: Ejemplo de consulta CKAN para obtener el listado de paquetes

2. Consultar la lista de las etiquetas del sitio.

   http://catalogo.datos.gob.mx/api/3/action/tag_list


```
{
    "success": true,
    "result": [
    "072",
    "1er sem",
    "2014",
    "2014-2018",
    "70 y mas",
    "abasto",
    "acceso",
    "accidentes",
    "accionario",
    "accionarios",
    "acciones",
    "Acciones",
    "aceptantes",
    "activa",
    "Activa.",
    "activas",
    "actividad",
    "Actividad",
    "actividades",
    "activo",
    "activos",
    "actores sociales",
    "acuacultura",
    "Acuacultura",
    "acuerdo",
    "acuícola",
    "acuícolas",
    "Administración Pública Federal",
    "administrativa",, "..." ]
}
```


FIGURA 10.2: Ejemplo de consulta CKAN para obtener el listado de etiquetas

3. Consultar la lista de las organizaciones del sitio.

   http://catalogo.datos.gob.mx/api/3/action/organization_list


```
{
    "success": true,
    "result": [
    "ayuntamiento-de-san-luis-potosi",
    "ayuntamiento-de-san-pedro-garza-garcia",
    "ayuntamiento-de-xalapa",
    "cdi",
    "cfe",
    "conagua",
    "conapo",
    "coneval",
    "estado-de-jalisco",
    "estado-de-morelos",
    "estado-de-puebla",
    "imss",
    "inegi",
    "nafin",
    "pemex",
    "pgr",
    "presidencia",
    "promexico",
    "ran",
    "sagarpa",
    "salud",
    "sct",
    "se",
    "sectur",
    "sedesol",
    "sener",
    "sep",
    "servicio-geologico-mexicano",
    "shcp" ]
}
```


FIGURA 10.3: Ejemplo de consulta CKAN para obtener el listado de organizaciones

4. Consultar los detalles de una organización en particular.

   http://catalogo.datos.gob.mx/api/3/action/organization_show?id=pemex


```
{
success: true,
result: {
    users: [...],
    display_name: "PEMEX",
    description: "Petróleos Mexicanos es la mayor empresa de México,
        el mayor contribuyente fiscal del país, así como una de las
        empresas más grandes de América Latina.
        Es de las pocas empresas petroleras del mundo
        que desarrolla toda la cadena productiva de la industria,
        desde la exploración, hasta la distribución y
        comercialización de productos finales, incluyendo
        la petroquímica.
        Durante 2012, sus ingresos totales ascendieron a
        un billón 647 mil millones de pesos,
        obtuvo un rendimiento de operación de 905 mil millones
        de pesos y su inversión ascendió a 311 mil
        millones de pesos.",
    image_display_url: "http://upload.wikimedia.org/wikipedia/
        en/thumb/a/ac/Pemex_logo.svg/196px-Pemex_logo.svg.png",
    title: "PEMEX",
    package_count: 62,
    created: "2014-06-19T16:48:30.775076",
    approval_status: "approved",
    is_organization: true,
    state: "active",
    extras: [ ],
    image_url: "http://upload.wikimedia.org/wikipedia/
        en/thumb/a/ac/Pemex_logo.svg/196px-Pemex_logo.svg.png",
    groups: [ ],
    num_followers: 0,
    revision_id: "7677a8f4-cfdc-4692-8216-e30bfc5280aa",}
    packages: [...]
    type: "organization",
    id: "bf6fceea-24d7-4e8e-a51c-8e08cdd7eef3",
    tags: [ ],
    name: "pemex"
}
```


FIGURA 10.4: Ejemplo de consulta CKAN para obtener los detalles de una organización

5. Consultar los detalles de un conjunto de datos en particular.

   http://catalogo.datos.gob.mx/api/3/action/package_show?id=4c98b21b-25ab-437b-9e73-88dde1c7e6a1&limit=1


```
{
success: true,
result: {
    license_title: null, maintainer: null,
    relationships_as_object: [ ], private: false,
    maintainer_email: null,
    revision_timestamp: "2014-12-18T05:25:26.236235",
    id: "4c98b21b-25ab-437b-9e73-88dde1c7e6a1",
    metadata_created: "2014-10-07T23:05:36.716956",
    metadata_modified: "2015-01-13T19:35:59.881787",
    author: null, author_email: null,
    state: "active", version: null,
    creator_user_id: "61464db5-c56e-483a-9510-b30d874f1a22",
    type: "dataset",
    resources: [...],
    num_resources: 1,
    tags: [...],
    tracking_summary: { total: 225, recent: 6},
    groups: [ ],
    license_id: null,
    relationships_as_subject: [ ],
    num_tags: 3,
    organization: {},
    name: "estudios-e-investigaciones",
    isopen: false,
    gov_type: [],
    url: null,
    notes: "Estudios e Investigaciones contratadas en 2013 en
        Petroleos Mexicanos y Organismos Subsidiarios",
    owner_org: "bf6fceea-24d7-4e8e-a51c-8e08cdd7eef3",
    extras: [
        { key: "dcat_modified", value: "2014-09-24T18:00:00"},
        { key: "dcat_publisher_email",
           value: "azucena.espindola@pemex.com"},
        { key: "dcat_publisher_name", value: "PEMEX"},
        { key: "guid", value: "EstudiosEInvest20132014"},
        { key: "language", value: "es"}
    ],
    title: "Estudios e investigaciones",
    revision_id: "cdef3d52-cede-4bda-aa1c-5c5f432aa2b0"
}
```


FIGURA 10.5: Ejemplo de consulta CKAN para obtener los detalles de un conjunto de datos

6. Consultar los detalles de un recurso en particular.

   http://catalogo.datos.gob.mx/api/3/action/resource_show?id=86c242a9-a7ed-4e66-9f6a-1c6f2a44e817


```
{
success: true,
result: {
    resource_group_id: "5ab78142-d6ee-4a3c-90a4-aba2412a9688",
    cache_last_updated: null,
    webstore_last_updated: null,
    datastore_active: false,
    id: "86c242a9-a7ed-4e66-9f6a-1c6f2a44e817",
    size: "36864",
    state: "active",
    last_modified: null,
    hash: "",
    description: "Estudios e Investigaciones contratadas en 2013 en
                Petroleos Mexicanos y Organismos Subsidiarios",
    format: "CSV",
    tracking_summary: {
        total: 31,
        recent: 1
    },
    mimetype_inner: null,
    url_type: null,
    mimetype: null,
    cache_url: null,
    name: "Petroleos Mexicanos y Organismos Subsidiarios",
    created: "2014-12-18T18:43:57.685996",
    url: "http://www.pemex.com/acerca/informes_publicaciones/
        Documents/Responsabilidad_Social/
        EstudiosEInvest20132014.txt",
    webstore_url: null,
    position: 0,
    revision_id: "a8a267e7-27b5-4337-8623-a6bbbe7c844f",
    resource_type: null
}
```


FIGURA 10.6: Ejemplo de consulta CKAN para obtener los detalles de un recurso

10.3. Arquitectura de la Aplicación

Los motores de búsqueda comúnmente utilizan una arquitectura similar a la que se muestra en la Figura 10.7.

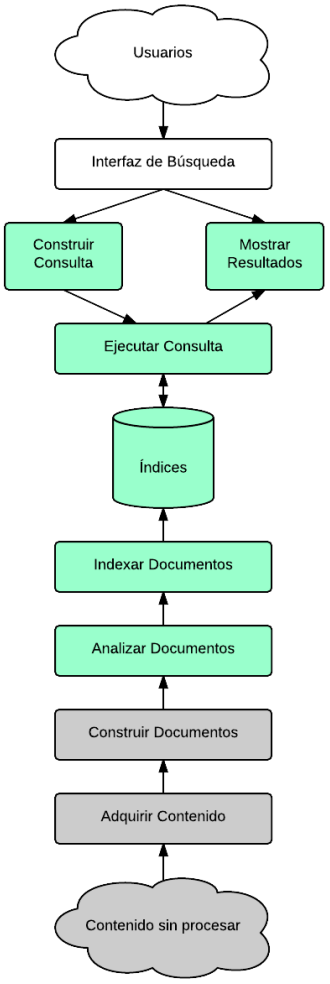


FIGURA 10.7: Arquitectura Típica de una Aplicación de Búsqueda.

Es importante aclarar que Lucene es básicamente una librería de búsqueda y el resto de los componentes de una aplicación de búsqueda (obtención de información, filtrado de documentos, interfaz de usuario, administración, etc) deberán ser implementados independientemente.

Lucene permite indexar y consultar cualquier información de la que se pueda extraer texto, no tiene restricción a cerca de el tipo de dato, formato o idioma, siempre y cuando se pueda obtener texto. Eso significa que se puede indexar y buscar información almacenada en páginas web y servidores remotos, documentos almacenados en sistemas de archivos locales, archivos de texto plano, documentos de Microsoft Word, XML, HTML, PDF u otro formato de el cual se pueda extraer información textual.

La aplicación propuesta en este documento utilizará Lucene para proveer la funcionalidad para los componentes de indexado y búsqueda que se encuentran señalados en verde, y otras herramientas tecnológicas que se describirán más adelante para proveer la funcionalidad de los componentes señalados en gris. Los componentes señalados en blanco no se incluyen como parte de la propuesta de implementación y forman parte de los objetivos para trabajo a futuro.

Implementación de la Solución

11.1. Configuración de Ambiente

Para el ambiente de desarrollo es necesario instalar las herramientas descritas a continuación.

11.1.1. Instalación de Apache Lucene

La distribución Java de Lucene consiste de múltiples librerías en formato JAR.
Para obtener la distribución binaria de Lucene es necesario seguir los siguientes pasos:

1. Descargar la versión mas reciente de la sección de descargas en el sitio web de Apache Lucene. http://lucene.apache.org/core/downloads.html . Para el propósito de este proyecto se trabajó con la versión 4.9.0
2. Extraer el archivo binario a el directorio deseado dentro del sistema.
3. Dentro de el nuevo directorio se encuentran los archivos enumerados a continuación, que son los únicos necesarios para el proyecto:

   a) **lucene-core-4.9.0.jar**.

   b) **lucene-analyzers-common-4.9.0.jar**.

   c) **lucene-queryparser-4.9.0.jar**.

   Estos archivos son los que se utilizarán para construir la aplicación. Para utilizarlos es necesario incluir la ubicación en la lista de librerías de clase cuando se compilen las clases de Java.

11.1.2. Instalación del módulo de Python para acceder la API de CKAN

La interfaz ckanapi permite acceder de manera local y remota a instancias de la plataforma CKAN para realizar operaciones masivas de datos y consultas utilizando el lenguaje de programación Python.

Esta librería fue desarrollada por el desarrollador Ian Ward de Ottawa, Canada y puede ser utilizada libremente ya que esta distribuida con licencia MIT. [19]

Como prerequisito para utilizar el módulo de Python ckanapi es necesario instalar Python en el sistema y es compatible con las versiones 2 y 3.

```
{
import ckanapi

demo = ckanapi.RemoteCKAN('http://demo.ckan.org',
    user_agent='ckanapiexample/1.0 (+http://example.com/my/website)')
groups = demo.action.group_list(id='data-explorer')
print groups
}
```

FIGURA 11.1: Ejemplo de solicitud remota utilizando la librería ckanapi en Python

```
{
 [u'data-explorer', u'example-group', u'geo-examples', u'skeenawild']
}
```

FIGURA 11.2: Ejemplo de respuesta a solicitud remota utilizando la librería ckanapi en Python

La librería ckanapi será utilizada para consultar la plataforma de Datos Abiertos del Gobierno Mexicano utilizando el API de CKAN a través de un programa escrito en lenguaje Python y a su vez descargar los documentos abiertos que las instituciones gubernamentales han hecho disponibles.

Los pasos para su instalación son:

1. Descargar la librería de el siguiente sitio https://github.com/ckan/ckanapi.
2. Ejecutar el comando *python setup.py install*

11.1.3. Instalación de Apache Tika

La herramienta Apache Tika permite la extracción de los metadatos e información textual de los documentos abiertos que se encuentran en múltiples formatos. [20].

Para propósitos de este proyecto se utilizará para extraer datos de archivos en formato XLM, XLS, JSON y CSV.

Tika es una librería que contiene múltiples analizadores (parsers en inglés) por cada tipo de documento soportado. La librería presenta la misma API para extraer texto y metadatos de un documento independientemente de el formato, e internamente encuentra el analizador apropiado, permitiendo escribir un sólo programa que uniformemente puede trabajar con diversos tipos de archivos.

La última versión disponible es la 1.7, consiste en un archivo ejecutable en formato jar llamado **tika-app-1.7.jar** y puede ser descargada de el sitio:
https://tika.apache.org/download.html.

Una vez descargada la herramienta, ésta puede ser ejecutada en modo línea de comando:

cat Document.pdf | java -jar tika-app-1.7.jar -

ó utilizando su interfaz gráfica:

java -jar tika-app-1.7.jar –gui

FIGURA 11.3: Interfaz Gráfica de la herramienta Apache Tika.

11.2. Detalles Técnicos

En las siguientes secciones se describe los detalles de implementación de cada uno de los componentes involucrados en la creación de la aplicación.

El código fuente completo puede ser consultado en el Apéndice C Aplicación de Búsqueda (Código Fuente).

También Es posible acceder al código de implementación en el repositorio:

https://github.com/citlalig/DatosAbiertosMXCrawler.

https://github.com/citlalig/DatosAbiertosMXSearcher.

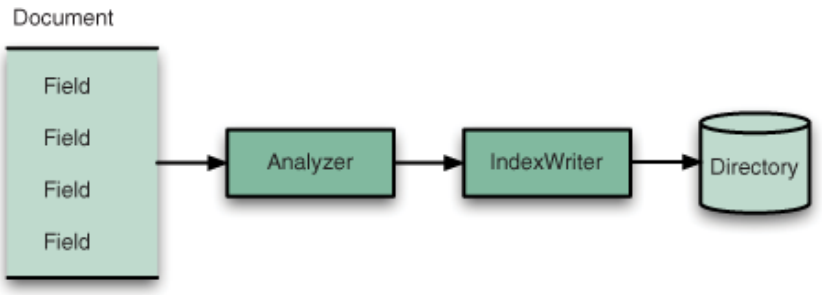


FIGURA 11.4: Clases del proceso de Indexado

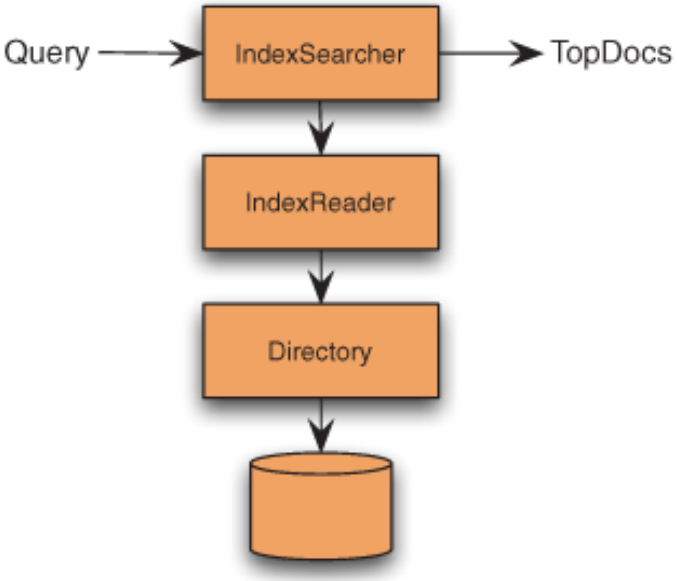


FIGURA 11.5: Clases del proceso de Búsqueda

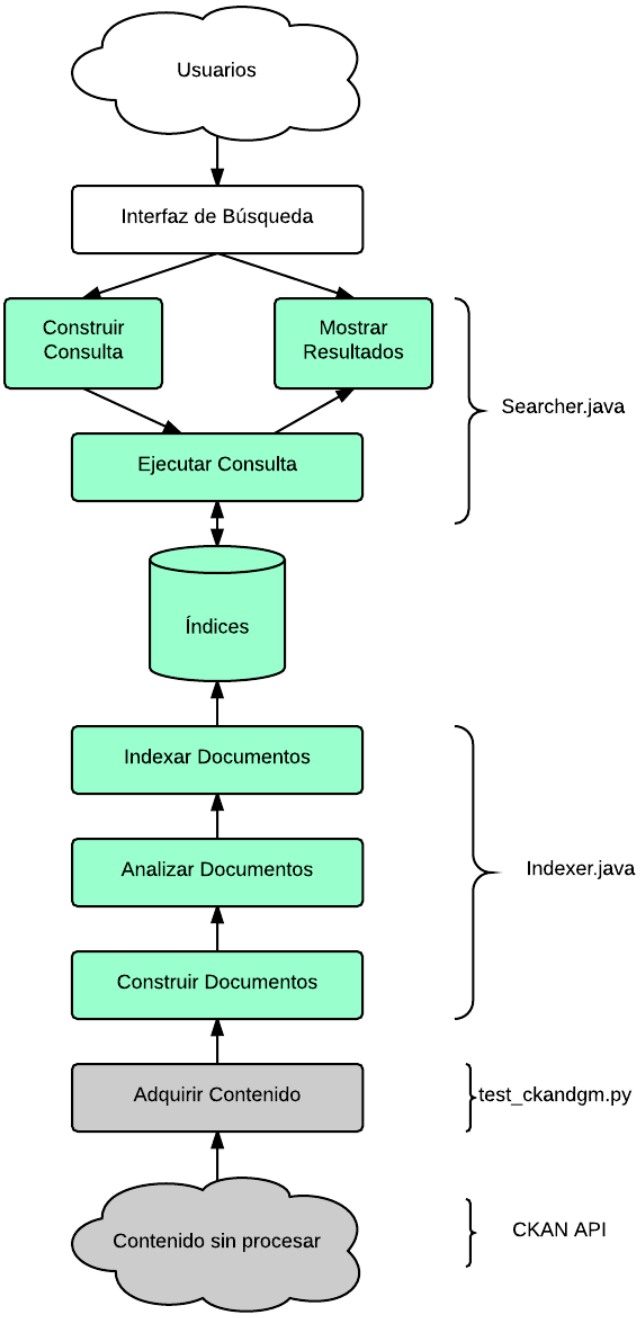


FIGURA 11.6: Arquitectura Prototipo de Plataforma Estructurada para el Acceso a Sistemas de Transparencia Gubernamental.

11.2.1. Adquirir Contenido

El primer paso es adquirir el contenido que necesita ser indexado. Este proceso require el uso de un programa que inspecciona las páginas web de forma metódica y automatizada. Dicha funcionalidad no es provista por la librería de Lucene por lo que se ha escrito un programa en lenguaje Python en conjunto con el módulo ckanapi para obtener los documentos en formato original, es decir sin procesar.
El programa esta compuesto de :

1. Una clase **CkanDatosAbiertosMX** encargada de realizar la conexión remota a el sitio especificado como parámetro así como realizar las invocaciones a la API de CKAN.

2. Un módulo llamado **test_ckandgm.py** encargado de hacer la conexión a la dirección URL de la plataforma de Datos del Gobierno Mexicano, consultar la lista de conjuntos de datos basado en una lista de organizaciones e iterar por cada paquete y recurso dentro de una organización particular para obtener así la dirección web de el recurso y descargar el documento a un repositorio local. Los documentos descargados serán almacenados en un directorio llamado *catálogo* relativo a la ubicación donde el programa fue ejecutado.

```
import ckanapi

class CkanDatosAbiertosMX:

    def __init__(self, site_name, user_agent=None):
        self.cknsite = ckanapi.RemoteCKAN(site_name,
        user_agent= user_agent)

    def getOrganizationList(self):
        # Return a list of the names of the site organizations.
        return self.cknsite.action.organization_list()

    def getOrganizationDetails(self, org):
        # Return the details of a organization
        return self.cknsite.action.organization_show(id=org)

    def getPackageList(self):
        # Return a list of the names of the site datasets (packages).
        return self.cknsite.action.package_list()

    def getPackageDetails(self, pck):
        #Return the metadata of a dataset (package) and its resources.
        return self.cknsite.action.package_show(id=pck)

    def getTagList (self):
        #Return a list of the site tags.
        return self.cknsite.action.tag_list()

    def getTagDetails(self, tag):
        #Return the details of a tag and all its datasets.
        return self.cknsite.action.show_tag(id=tag)

    def getResourceDetails(self, rsrc):
        #Return the metadata of a resource.
        return self.cknsite.action.resource_show(id=rsrc)
```

FIGURA 11.7: Código fuente de la clase CkanDatosAbiertosMX

```
# Encoding: UTF-8
__author__ = 'citlalig'

from ckandgm import CkanDatosAbiertosMX
import urllib2
import os
import httplib
import socket
import ssl
import sys

def writeFile(fileName, data):
    output = open(fileName, 'wb')
    output.write(data)
    output.close()

def downloadFile(url, path, resource_name, resource_format):
    print 'Downloading file '+ url
    try:
        response = urllib2.urlopen(url, timeout = 30)
        if resource_format == "" :
            basename = os.path.basename(url)
            resource_format = basename[basename.rfind('.')+1:]
        file_name = resource_name + '.' + resource_format.lower()
        file_name.encode("UTF-8")
        print 'Save file ' + file_name + ' to Path: ' + path
        complete_file_name = os.path.join(path, file_name)
        writeFile(complete_file_name, response.read())

    except urllib2.HTTPError, e:
        print type(e)
    except httplib.BadStatusLine, e:
        print type(e)
    except urllib2.URLError, e:
        print type(e)
    except socket.timeout as e:
        print type(e)
    except ssl.SSLError as e:
        print type(e)
    except ssl.SSLError as e:
        print type(e)
    except IOError as e:
        print type(e)

def makeDir(dirName):
    if not os.path.exists(dirName):
        os.makedirs(dirName)
```

FIGURA 11.8: Código fuente de el módulo test_ckandgm.py (Parte 1)

```
##### MAIN CODE ####

SITE_NAME = 'http://catalogo.datos.gob.mx/'
USER_AGENT = 'MXOpenDataEngine/1.0'
organizations_list = ['pemex', 'promexico',
    'sep', 'presidencia', 'sagarpa',
    'shcp', 'sedesol' ]
MAIN_FOLDER = 'catalogo'
SHOW_DETAILS_ACTION = 'api/3/action/organization_show?id='
datosabiertosmx = CkanDatosAbiertosMX(SITE_NAME, USER_AGENT)
makeDir(MAIN_FOLDER)
organizations = datosabiertosmx.getOrganizationList()
for org in organizations:
    print "Organization: ", org
    if org in organizations_list:
        print "Processing Organization: " + org
        org_det = datosabiertosmx.getOrganizationDetails(org)
        name = org_det['name']
        path = os.path.join(MAIN_FOLDER, name)
        makeDir(path)
        downloadFile(SITE_NAME + SHOW_DETAILS_ACTION
                + name, path, name, "json")
        packages = org_det['packages']
        for package in packages:
            package_name = package['name']
            print '\n\tProcessing Package: "' +  package_name
            path = os.path.join(os.path.join(MAIN_FOLDER ,
                name , package_name))
            makeDir(path)
            resources = package['resources']

            for resource in resources:
                print '\t\t Processing Resources '
                        + resource['name'] + 'in package. '
                downloadFile(resource['url'], path,
                        resource['name'], resource['format'])
```

FIGURA 11.9: Código fuente de el módulo test_ckandgm.py (Parte 2)

El programa descrito en el código fuente de las imágenes 11.8 y 11.9 deberá ser ejecutado utilizando el comando

python test_ ckandgm.py >resultados.log »resultados.log

Y producirá como resultado un directorio llamado **catalogo** en la ubicación donde el programa fue ejecutado con un subdirectorio por cada organización examinada.

```
/Users/citlalig $ tree Documents/GitHub/DatosAbiertosMXCrawler/catalogo/ -
d -L 1
Documents/GitHub/DatosAbiertosMXCrawler/catalogo/
├── pemex
├── presidencia
├── promexico
├── sagarpa
└── sedesol
├── sep
└── shcp

7 directories
```

FIGURA 11.10: Directorio de Documentos Abiertos extraídos de una plataforma CKAN

11.2.2. Extraer Texto

Un paso crítico al construir una aplicación de búsqueda es extraer texto de los documentos que se necesita indexar. En un escenario ideal, el texto debería de existir en formato plano, sin embargo la mayoría de los archivos no se encuentran en tal formato, por el contrario, se encuentran en formatos populares como Word, Excel, PowerPoint, Visio, Flash, PDF, Open Office, Rich Text Format (RTF), TAR, ZIP, y BZIP2.
Incluso formatos relativamente sencillos como XML y HTML deben ser tratados con cuidado para no incluir accidentalmente alguna etiqueta que no forma parte del texto original.
Para resolver los retos que involucra el extraer texto de formatos diversos, existe la librería Apache Tika, la cual es una API fácil de usar que permite filtrar texto.

El método principal es *parse* en la clase **org.apache.tika.parser.Parser**:

```
void parse(InputStream stream
           ContentHandler handler,
           Metadata metadata,
           ParseContext context) {}
```

FIGURA 11.11: Firma del método parse en la librería Tika

```
Metadata metadata = new Metadata();
metadata.set(Metadata.RESOURCE_NAME_KEY, f.getName());
InputStream is = new FileInputStream(f);
Parser parser = new AutoDetectParser();
ContentHandler handler = new BodyContentHandler();
ParseContext context = new ParseContext();
context.set(Parser.class, parser);
try {
    parser.parse(is, handler, metadata, new ParseContext());
} finally {
    is.close();
}
```

FIGURA 11.12: Bloque de código: Extraer Texto

11.2.3. Construir Documentos

Una vez que hemos obtenido los archivos en su formato original, necesitan ser indexados. Para ello es necesario que su contenido sea traducido en unidades usadas por el motor de búsqueda y llamados *Documentos*.

El *Documento* típicamente consiste de campos con un nombre y un valor, por ejemplo: título, cuerpo, resumen, autor, url, etc.

Lucene provee una API para construir campos a través de las clases **org.apache.lucene.document.Document** y **org.apache.lucene.document.Field**.

La clase *Document* representa una colección de objetos de tipo *Field*.

Field es la clase que contiene el contenido a ser indexado.

En la aplicación se utilizan lo campos ubicación de archivo, nombre de archivo, fecha de modificación y contenido para identificar un Documento.

```
Document doc = new Document();
doc.add(new StringField("fullpath", f.getPath(),
    Field.Store.YES));
doc.add(new StringField("filename", f.getCanonicalPath(),
    Field.Store.YES));
doc.add(new LongField("modified", f.lastModified(),
    Field.Store.NO));
doc.add(new TextField("contents", handler.toString(),
    Field.Store.NO));
```

FIGURA 11.13: Bloque de código: Construir un Documento

11.2.4. Analizar los Documentos

Durante el proceso de analizar un documento, el texto debe ser dividido en elementos llamados *tokens*. Cada *token* corresponde generalmente a una palabra.
Lucene provee un conjunto de analizadores que permiten tener control sobre este proceso. Algunos ayudan a ignorar palabras que no ayudan a distinguir un documento de otro por ejemplo la, el, en, y, etc o que convierten los *token* a minúscula para favorecer la búsqueda.
La aplicación de búsqueda descrita en este documento utiliza la clase
org.apache.lucene.analysis.standard.StandardAnalyzer

```
Directory dir = FSDirectory.open(new File(indexDir));
Analyzer analyzer = new StandardAnalyzer(Version.LUCENE_4_9);
IndexWriterConfig iwc = new IndexWriterConfig(Version.LUCENE_4_9,
        analyzer);
```

FIGURA 11.14: Bloque de código: Analizar un Documento

11.2.5. Indexar los Documentos

Durante el indexado de los documentos, el *Documento* se añade a el *Índice*.

Lucene provee las herramientas necesarias para realizar este proceso utilizando la clase **org.apache.lucene.index.IndexWriter**

IndexWriter el el componente principal en el proceso de indexado. La clase crea un índice nuevo o abre un índice existente, y agrega, actualiza o elimina documentos de él.

```
private IndexWriter writer = new IndexWriter(dir, iwc);

private void indexFile(File f) throws Exception {
    System.out.println("Indexing " + f.getCanonicalPath());
    Document doc = getDocument(f);
    writer.addDocument(doc);
}

public int index(String dataDir, FileFilter filter)
    throws Exception {
    File[] files = new File(dataDir).listFiles();
    for (File f : files) {
        if (!f.isDirectory() && !f.isHidden() && f.exists()
            && f.canRead()
            && (filter == null || filter.accept(f))) {
                indexFile(f);
        }
    }
    return writer.numDocs();
}
```

FIGURA 11.15: Bloque de código: Indexar un Documento

11.2.6. Construir Consulta

Lucene incluye varias clases concretas de la clase abstracta:

org.apache.lucene.search.Query

Algunos ejemplos de ellas son: TermQuery, BooleanQuery, PhraseQuery, PrefixQuery, PhrasePrefixQuery, TermRangeQuery, NumericRangeQuery, FilteredQuery, y SpanQuery. Cuando se solicita una consulta, la solicitud original se traduce a un objeto de tipo *Query*. Los objetos *Query* pueden ser simples o complejos. Lucene provee un paquete llamado *QueryParser* para convertir texto a objetos *Query* utilizando la clase: **org.apache.lucene.queryparser.classic.QueryParser**

```
Analyzer analyzer = new StandardAnalyzer(Version.LUCENE_4_9);
QueryParser parser = new QueryParser(Version.LUCENE_4_9, "contents",
    analyzer);
Query query = parser.parse(q);
```

FIGURA 11.16: Bloque de código: Construir Consulta

Expresión de Consulta	Coincide con documentos que ...
java	contienen la palabra java
java junit	contienen la palabra java o junit o ambas.
+java +junit	contienen ambas palabras java y junit.
title:ant	El campo title coincide con el nombre ant.
java	Contiene términos que son similares como lava.
lastmodified: [1/1/15 TO 12/31/15]	El campo lastmodified contiene valores entre las fechas.

CUADRO 11.1: Ejemplos de expresiones para realizar una consulta

11.2.7. Ejecutar Consulta

La ejecución de una consulta es el proceso de consultar el *Índice* y regresar los *Documentos* que concuerdan.

La clase para ejecutar la consulta es **org.apache.lucene.search.IndexSearcher**

IndexSearcher es el enlace principal hacia un índice y provee varios métodos de búsqueda. Es una clase que abre un índice en modo de sólo lectura y ofrece numerosos métodos de búsqueda. El resultado es un objeto de tipo TopDocs.

La clase: **org.apache.lucene.search.TopDocs** es simplemente un contenedor de punteros a los resultados de búsqueda ordenados por relevancia.

```
IndexSearcher is = new IndexSearcher(reader);
long start = System.currentTimeMillis();
TopDocs hits = is.search(query, 10);
long end = System.currentTimeMillis();

System.err.println("Found " + hits.totalHits + " document(s) (in "
+ (end - start) + " milliseconds) that matched query '" + q
+ "':");
```

FIGURA 11.17: Bloque de código: Ejecutar Consulta

11.2.8. Mostrar Resultados

Una vez que se obtienen los *Documentos* que concuerdan con una consulta, y han sido ordenados, están listos para ser mostrados de una manera que sea intuitiva para el usuario. A su vez, la interfaz deberá permitir búsquedas adicionas o filtrado de búsqueda. Lucene se encarga de administrar este proceso pero permitiendo personalizar los resultados.

```
for (ScoreDoc scoreDoc : hits.scoreDocs) {
    Document doc = is.doc(scoreDoc.doc);
    System.out.println(doc.get("fullpath"));
}
```

FIGURA 11.18: Bloque de código: Mostrar Resultados

11.3. Instrucciones para compilar el código fuente

Los comandos necesarios para compilar las clases Indexer.java, TikaIndexer.java y Searcher.java desde una linea de comandos son:

```
export LUCENE_HOME=/Applications/lucene-4.9.0
export APP_HOME=/Documents/Workspace/DatosAbiertosMXSearcher
export DATA_CATALOG=/Documents/GitHub/DatosAbiertosMXCrawler

javac -classpath $LUCENE_HOME/core/lucene-core-4.9.0.jar:\
    $LUCENE_HOME/analysis/common/lucene-analyzers-common-4.9.0.jar \
    $APP_HOME/src/datos/gob/mx/Indexer.java
javac -classpath $LUCENE_HOME/core/lucene-core-4.9.0.jar:\
    $LUCENE_HOME/analysis/common/lucene-analyzers-common-4.9.0.jar:\
    $LUCENE_HOME/core/tika-app-1.7.jar:\
    $APP_HOME/bin \
    $APP_HOME/src/datos/gob/mx/TikaIndexer.java
javac -classpath $LUCENE_HOME/core/lucene-core-4.9.0.jar:\
    $LUCENE_HOME/analysis/common/lucene-analyzers-common-4.9.0.jar:\
    $LUCENE_HOME/queryparser/lucene-queryparser-4.9.0.jar \
    $APP_HOME/src/datos/gob/mx/Searcher.java
```

FIGURA 11.19: Compilar las clases de Java

Resultados Obtenidos

En esta sección se demuestra como una vez que los archivos han sido adquiridos a través de la plataforma CKAN, es sencillo realizar el Indexado y Búsqueda de texto dentro de el contenido y metadatos de un documento mediante el programa que fue escrito para ese propósito utilizando las clases que la herramienta Apache Lucene provee.

12.1. Resultados de la ejecución del proceso de Indexado

Los comandos necesarios para ejecutar la clase TikaIndexer.java y por lo tanto iniciar el proceso de Indexado se describen a continuación.
Se ejecutará una vez por cada subdirectorio correspondiente a una institución gubernamental para mayor claridad:
Usage: java datos.gob.mx.TikaIndexer <index dir><data dir>

```
java -cp $LUCENE_HOME/core/lucene-core-4.9.0.jar:\
    $LUCENE_HOME/analysis/common/lucene-analyzers-common-4.9.0.jar:\
    $APP_HOME/bin:\
    $LUCENE_HOME/core/tika-app-1.7.jar\
    datos.gob.mx.TikaIndexer index  $DATA_CATALOG/catalogo/pemex
```

FIGURA 12.1: Ejecutar la clase TikaIndexer.java

Y el resultado de la ejecución serán los Documentos indexados en el directorio indicado.

```
Indexing pemex/pr-elaboracion-de-petroliferos/
    PR-Elaboracion de petroliferos.csv
Indexing pemex/pr-elaboracion-de-petroliferos/
    PR-Elaboracion de petroliferos.xml
Indexing pemex/pr-elaboracion-de-petroliferos-por-refineria/
    PR-Elaboracion de petroliferos por refineria.csv
Indexing pemex/pr-elaboracion-de-petroliferos-por-refineria/
    PR-Elaboracion de petroliferos por refineria.xml
Indexing pemex/pr-estructura-de-precios-de-gasolinas-y-diesel/
    PR-Estructura de precios de gasolinas y diesel.csv
Indexing pemex/pr-estructura-de-precios-de-gasolinas-y-diesel/
    PR-Estructura de precios de gasolinas y diesel.xml
Indexing pemex/pr-proceso-de-crudo/
    PR-Proceso de crudo.csv
Indexing pemex/pr-proceso-de-crudo/
    PR-Proceso de crudo.xml
Indexing pemex/pr-valor-comercio-exterior/
    PR-Valor comercio exterior.csv
Indexing pemex/pr-valor-comercio-exterior/
    PR-Valor comercio exterior.xml
Indexing pemex/pr-valor-ventas/
    PR-Valor ventas.csv
Indexing pemex/pr-valor-ventas/
    PR-Valor ventas.xml
Indexing pr-volumen-comercio-exterior/
    PR-Volumen comercio exterior.csv
Indexing pemex/pr-volumen-comercio-exterior/
    PR-Volumen comercio exterior.xml
Indexing pemex/pr-volumen-ventas/
    PR-Volumen ventas.csv
Indexing pemex/pr-volumen-ventas/
    PR-Volumen ventas.xml
Indexing pemex/quejas-oic-junio-2014/
    Petroleos Mexicanos y Organismos Subsidiarios.csv
Indexing pemex/quejas-oic-septiembre-2014/
    Petroleos Mexicanos y Organismos Subsidiarios.csv
Indexing 181 files took 3583 milliseconds
```

FIGURA 12.2: Crear índices de los archivos existentes en un directorio.

12.2. Resultados de la ejecución del proceso de Búsqueda

Los comandos necesarios para ejecutar la clase Searcher.java y por lo tanto realizar una búsqueda a partir de los índices existentes se describen a continuación:
Usage: java datos.gob.mx.Searcher <index dir><query>

```
java -cp $LUCENE_HOME/core/lucene-core-4.9.0.jar:\
    $LUCENE_HOME/analysis/common/lucene-analyzers-common-4.9.0.jar:\
    $APP_HOME/bin:\
    $LUCENE_HOME/queryparser/lucene-queryparser-4.9.0.jar\
    datos.gob.mx.Searcher index pemex
```

FIGURA 12.3: Ejecutar la clase Searcher.java

```
Found 210 document(s) (in 37 milliseconds)
    that matched query 'pemex':
catalogo/pemex/quejas-oic-junio-2014/
    Petroleos Mexicanos y Organismos Subsidiarios.csv
catalogo/pemex/quejas-oic-septiembre-2014/
    Petroleos Mexicanos y Organismos Subsidiarios.csv
catalogo/pemex/pemex.json
catalogo/pemex/pemex-estadisticas-seleccionadas/
    PEMEX Estadisticas seleccionadas.xml
```

FIGURA 12.4: Resultados de Búsqueda. Query = pemex

El resultado de la ejecución serán la lista de Documentos que satisfacen la consulta.
La consulta mostrada previamente es muy sencilla sin embargo pueden ser tan complicadas como se desee.
Un ejemplo mas elaborado de búsqueda utilizando una expresión de consulta utilizando metadatos, basado en contenido, titulo, y fecha de modificación se muestra a continuación:

```
$ java -cp $LUCENE_HOME/core/lucene-core-4.9.0.jar:\
    $LUCENE_HOME/analysis/common/lucene-analyzers-common-4.9.0.jar:\
    $APP_HOME/bin:\
    $LUCENE_HOME/queryparser/lucene-queryparser-4.9.0.jar \
    datos.gob.mx.Searcher index
    "indicadores+pobreza modified:[1/1/15 TO 12/31/15]
     -multidimensional"
Found 7 document(s) (in 88 milliseconds)
that matched query 'indicadores+pobreza
    modified:[1/1/15 TO 12/31/15] -multidimensional':
catalogo/presidencia/
    mexico-incluyente-indicadores-del-pnd-y-pmp/
    Indicadores del Plan Nacional de Desarrollo, 2013-2018 2.
    México Incluyente.xlsx
catalogo/presidencia/mexico-con-educacion-de-calidad-
    indicadores-del-pnd-y-pmp/
    Indicadores del Programa Nacional de Cultura Física
    y Deporte 2014-2018
    y su vinculación con la planeación nacional.xlsx
catalogo/presidencia/mexico-con-educacion-de-calidad-
    indicadores-del-pnd-y-pmp/
    Indicadores del Plan Nacional de Desarrollo,
    2013-2018 3.
    México con Educación de Calidad.xlsx
catalogo/presidencia/mexico-incluyente-estadisticas-nacionales/
    Indicadores de seguridad social del IMSS
    Concepto.xlsx
catalogo/presidencia/mexico-con-educacion-de-calidad-
    indicadores-del-pnd-y-pmp/
    Indicadores del Programa Sectorial de
    Educación 2013-2018
    y su vinculación con la planeación nacional
    (Ciclos escolares).xlsx
catalogo/presidencia/mexico-en-paz-indicadores-del-pnd-y-pmp/
    Indicadores del Programa Sectorial de Defensa Nacional 2013-2018
    y su vinculación con la planeación nacional.xlsx
catalogo/presidencia/mexico-incluyente-indicadores-del-pnd-y-pmp/
    Indicadores del Programa Nacional para el Desarrollo
    y la Inclusión de las Personas con Discapacidad, 2014-2018
    y su vinculación con la planeación nacional.xlsx
```

FIGURA 12.5: Ejemplo de Búsqueda. Query = 'indicadores+pobreza modified:[1/1/15 TO 12/31/15] -multidimensional':

Conclusiones y Trabajo a Futuro

Anexo - Resumen de formatos de archivos más comunes

JSON

Acrónimo de JavaScript Object Notation, es un formato ligero para el intercambio de datos. JSON es un subconjunto de la notación literal de objetos de JavaScript que no requiere el uso de XML.

XML

Siglas en inglés de eXtensible Markup Language ('lenguaje de marcas extensible'), es un lenguaje de marcas desarrollado por el World Wide Web Consortium (W3C) utilizado para almacenar datos en forma legible.

XLS

La extensión de archivo por defecto para aplicaciones de hojas de cálculo.

CSV

Tipo de documento en formato abierto sencillo para representar datos en forma de tabla, en las que las columnas se separan por comas y las filas por saltos de línea.

TXT

La extensión para archivo compuesto únicamente por texto sin formato, sólo caracteres, lo que lo hace también legible por humanos.

ZIP

Formato de compresión sin pérdida, muy utilizado para la compresión de datos como documentos, imágenes o programas.

KMZ

Es un lenguaje de marcado basado en XML para representar datos geográficos en tres dimensiones en formato comprimido.

Anexo - Calendario de Actividades

Anexo - Aplicación de Búsqueda (Código Fuente)

```
package datos.gob.mx;
import ...

/**
 * This code was originally written for Erik's Lucene intro java.net
 * article and modified by Citlali Garcia
 * to meet the application needs.
 */

public class Indexer {
    public static void main(String[] args) throws Exception {
        if (args.length != 2) {
            throw new IllegalArgumentException("Usage: java "
                    + Indexer.class.getName()
                    + " <index dir> <data dir>");
        }
        String indexDir = args[0];
        String dataDir = args[1];
        long start = System.currentTimeMillis();
        Indexer indexer = new Indexer(indexDir);
        int numIndexed;
        try {
            numIndexed =
                indexer.index(dataDir, new TextFilesFilter());
        } finally {    indexer.close();}
        long end = System.currentTimeMillis();
        System.out.println("Indexing " + numIndexed + " files took "
                + (end - start) + " milliseconds");
    }
```

FIGURA C.1: Código Fuente para clase Indexer.java (Parte 1)

```
    private IndexWriter writer;
    public Indexer(String indexDir) throws IOException {
        Directory dir = FSDirectory.open(new File(indexDir));
        Analyzer analyzer = new StandardAnalyzer(Version.LUCENE_4_9);
        IndexWriterConfig iwc =
            new IndexWriterConfig(Version.LUCENE_4_9, analyzer);
        writer = new IndexWriter(dir, iwc);
    }

    public void close() throws IOException { writer.close(); }

    public int index(String dataDir, FileFilter filter)
        throws Exception {
        File[] files = new File(dataDir).listFiles();
        for (File f : files) {
            if (!f.isDirectory() && !f.isHidden() && f.exists()
                && f.canRead()
                && (filter == null || filter.accept(f))) {
                    indexFile(f);
            }
        }
        return writer.numDocs();
    }

    private static class TextFilesFilter implements FileFilter {
        public boolean accept(File path) {
            return path.getName().toLowerCase().endsWith(".txt");
        }
    }

    protected Document getDocument(File f) throws Exception {
        Document doc = new Document();
        doc.add(new StringField("fullpath", f.getPath(),
                    Field.Store.YES));
        doc.add(new TextField("contents", new BufferedReader(
                new InputStreamReader(new FileInputStream(f),
                        StandardCharsets.UTF_8))));
        doc.add(new StringField("filename", f.getName(),
            Field.Store.YES));
        doc.add(new LongField("modified", f.lastModified(),
            Field.Store.NO));
        return doc;
    }

    private void indexFile(File f) throws Exception {
        System.out.println("Indexing " + f.getCanonicalPath());
        Document doc = getDocument(f);
        writer.addDocument(doc);
    }
}
```

FIGURA C.2: Código Fuente para clase Indexer.java (Parte 2)

```
package datos.gob.mx;
import ...
public class TikaIndexer extends Indexer {

    static Set<String> textualMetadataFields = new HashSet<String>();
    static {
        textualMetadataFields.add(
            TikaCoreProperties.TITLE.getName());
        textualMetadataFields.add(
            TikaCoreProperties.COMMENTS.getName());
        textualMetadataFields.add(
            TikaCoreProperties.KEYWORDS.getName());
        textualMetadataFields.add(
            TikaCoreProperties.DESCRIPTION.getName());
        textualMetadataFields.add(
            TikaCoreProperties.KEYWORDS.getName());
    }

    public static void main(String[] args) throws Exception {
        if (args.length != 2) {
            throw new IllegalArgumentException("Usage: java "
                    + TikaIndexer.class.getName()
                    + " <index dir> <data dir>");
        }

        TikaConfig config = TikaConfig.getDefaultConfig();
        System.out.println("Mime type parser:" + config.getParser());

        String indexDir = args[0];
        String dataDir = args[1];

        long start = new Date().getTime();
        TikaIndexer indexer = new TikaIndexer(indexDir);
        int numIndexed = indexer.index(dataDir, null);
        indexer.close();
        long end = new Date().getTime();

        System.out.println("Indexing " + numIndexed + " files took "
                + (end - start) + " milliseconds");
    }
```

FIGURA C.3: Código Fuente para clase TikaIndexer.java (Parte 1)

```
public TikaIndexer(String indexDir) throws IOException {
        super(indexDir);
    }

    protected Document getDocument(File f) throws Exception {
        Metadata metadata = new Metadata();
        metadata.set(Metadata.RESOURCE_NAME_KEY, f.getName());
        InputStream is = new FileInputStream(f);
        Parser parser = new AutoDetectParser();
        ContentHandler handler = new BodyContentHandler();
        ParseContext context = new ParseContext();
        context.set(Parser.class, parser);
        try {
            parser.parse(is, handler, metadata, new ParseContext());
        } finally {
            is.close();
        }
        Document doc = new Document();
        doc.add(new StringField("fullpath", f.getPath(),
            Field.Store.YES));
        doc.add(new StringField("filename", f.getCanonicalPath(),
                Field.Store.YES));
        doc.add(new LongField("modified", f.lastModified(),
            Field.Store.NO));
        doc.add(new TextField("contents", new BufferedReader(
                new InputStreamReader(new FileInputStream(f),
                        StandardCharsets.UTF_8))));
        doc.add(new TextField("contents", handler.toString(),
            Field.Store.NO));
        for (String name : metadata.names()) {
            String value = metadata.get(name);
            if (textualMetadataFields.contains(name)) {
                doc.add(new TextField("contents", value,
                    Field.Store.NO));
            }
            doc.add(new TextField(name, value,
                Field.Store.YES));
        }
        return doc;
    }
}
```

FIGURA C.4: Código Fuente para clase TikaIndexer.java (Parte 2)

```java
package datos.gob.mx;

import org.apache.lucene.document.Document;
import org.apache.lucene.index.DirectoryReader;
import org.apache.lucene.index.IndexReader;
import org.apache.lucene.search.IndexSearcher;
import org.apache.lucene.search.Query;
import org.apache.lucene.search.ScoreDoc;
import org.apache.lucene.search.TopDocs;
import org.apache.lucene.store.Directory;
import org.apache.lucene.store.FSDirectory;
import org.apache.lucene.queryparser.classic.QueryParser;
import org.apache.lucene.queryparser.classic.ParseException;
import org.apache.lucene.analysis.Analyzer;
import org.apache.lucene.analysis.standard.StandardAnalyzer;
import org.apache.lucene.util.Version;

import java.io.File;
import java.io.IOException;

/**
 * This code was originally written for Erik's Lucene intro java.net
 * article and modified by Citlali Garcia to meet
 * the application needs.
 */
public class Searcher {

    public static void main(String[] args) throws
        IllegalArgumentException, IOException, ParseException {
        if (args.length != 2) {
            throw new IllegalArgumentException("Usage: java "
                    + Searcher.class.getName()
                    + " <index dir> <query>");
        }

        String indexDir = args[0];
        String query = args[1];
        search(indexDir, query);
    }
```

FIGURA C.5: Código Fuente para clase Searcher.java (Parte 1)

```
  public static void search(String indexDir, String q)
      throws IOException, ParseException {

        Directory dir = FSDirectory.open(new File(indexDir));
        IndexReader reader = DirectoryReader.open(dir);
        IndexSearcher is = new IndexSearcher(reader);

        Analyzer analyzer = new StandardAnalyzer(Version.LUCENE_4_9);
        QueryParser parser = new QueryParser(Version.LUCENE_4_9,
            "contents", analyzer);

        Query query = parser.parse(q);
        long start = System.currentTimeMillis();
        TopDocs hits = is.search(query, 10);
        long end = System.currentTimeMillis();

        System.err.println("Found " + hits.totalHits
                + " document(s) (in " + (end - start)
                + " milliseconds) that matched query '" + q
                + "':");

        for (ScoreDoc scoreDoc : hits.scoreDocs) {
            Document doc = is.doc(scoreDoc.doc);
            System.out.println(doc.get("fullpath"));
        }
        reader.close();
    }
}
```

FIGURA C.6: Código Fuente para clase Searcher.java (Parte 2)

Anexo - Ejemplos de implementaciones CKAN en el mundo

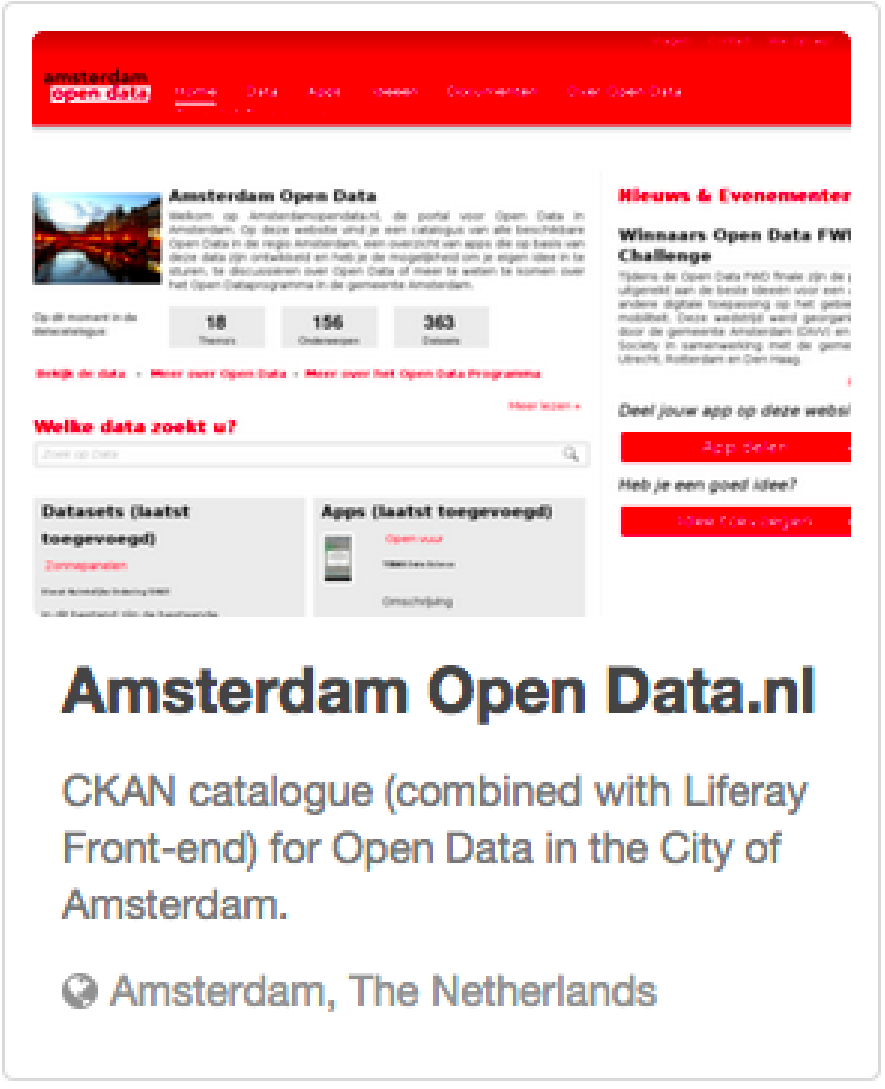
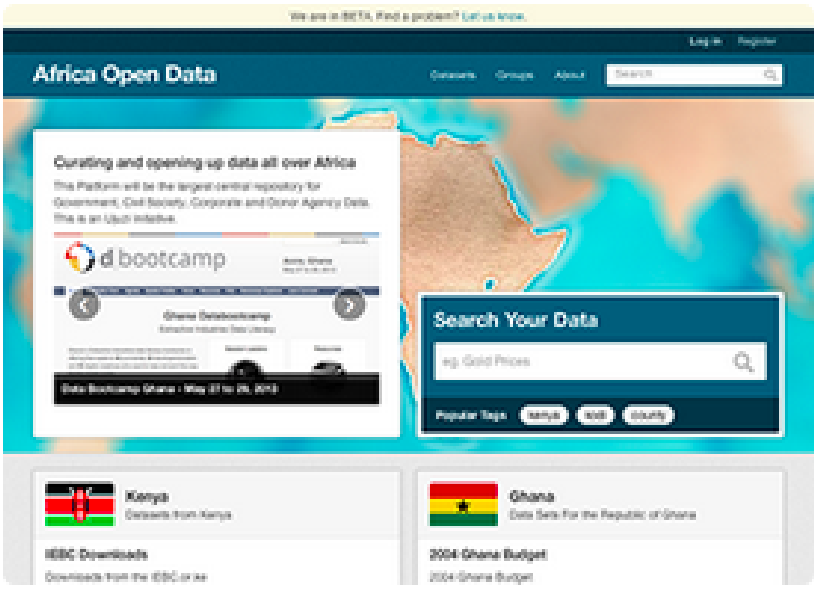


FIGURA D.1: Datos Abiertos del Gobierno de Amsterdam.



Africa Open Data

The Open Africa Platform initiative aims to be largest repository of Data on the African continent - a Library of Congress of Data for African for use by citizens, media, activists, governments and civil society. It is part of the Ujuzi Initiative which is created and funded by the Africa Media Initiative (AMI), the World Bank Institute (WBI) and Google.

Africa

FIGURA D.2: Datos Abiertos del Gobierno de África.

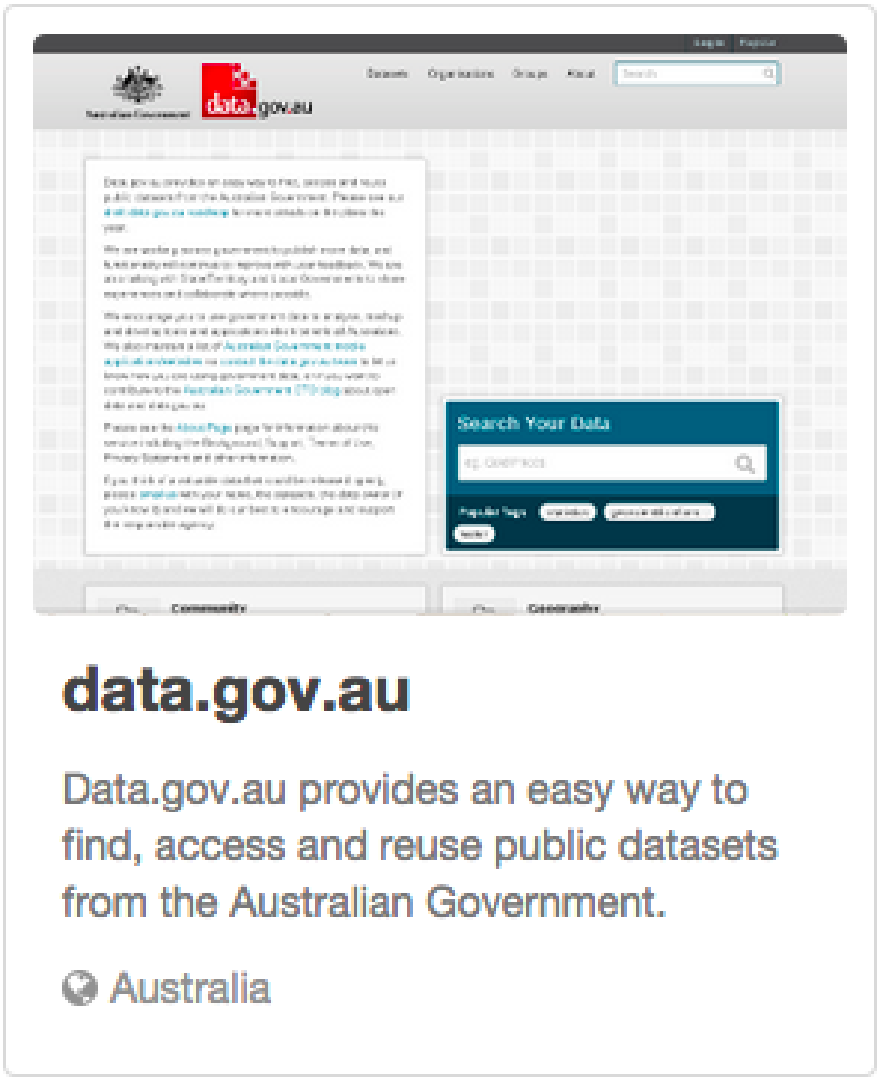


FIGURA D.3: Datos Abiertos del Gobierno de Australia.

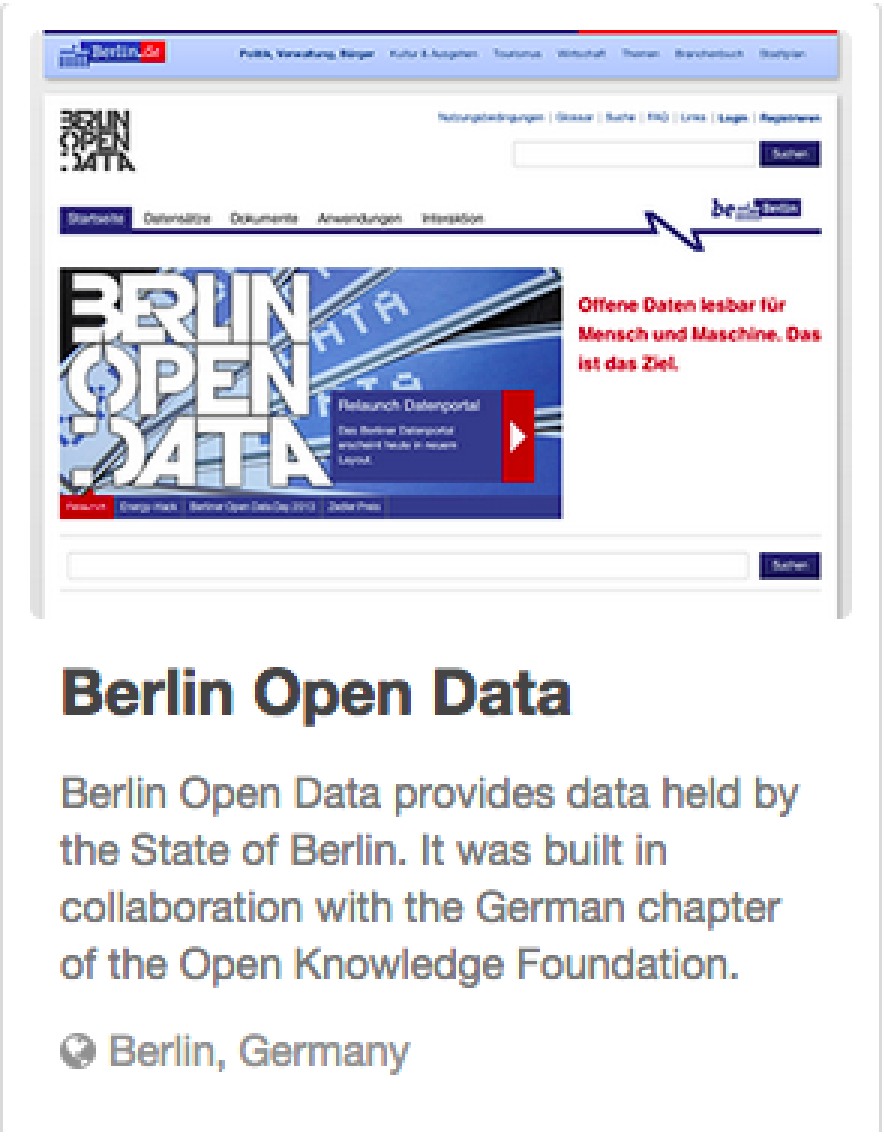


FIGURA D.4: Datos Abiertos del Gobierno de Berlin.

FIGURA D.5: Datos Abiertos del Gobierno de Brasil.

FIGURA D.6: Datos Abiertos del Gobierno de Buenos Aires.

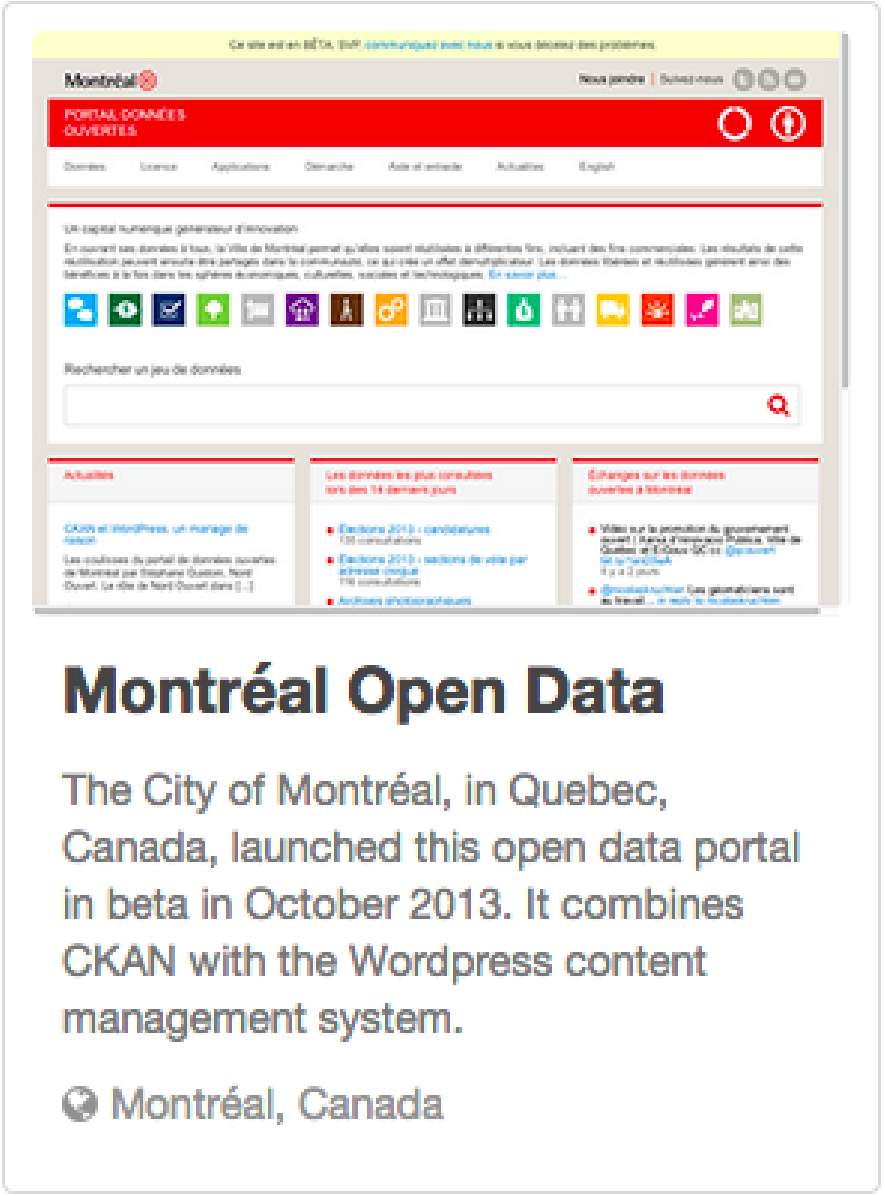


FIGURA D.7: Datos Abiertos del Gobierno de Montreal.

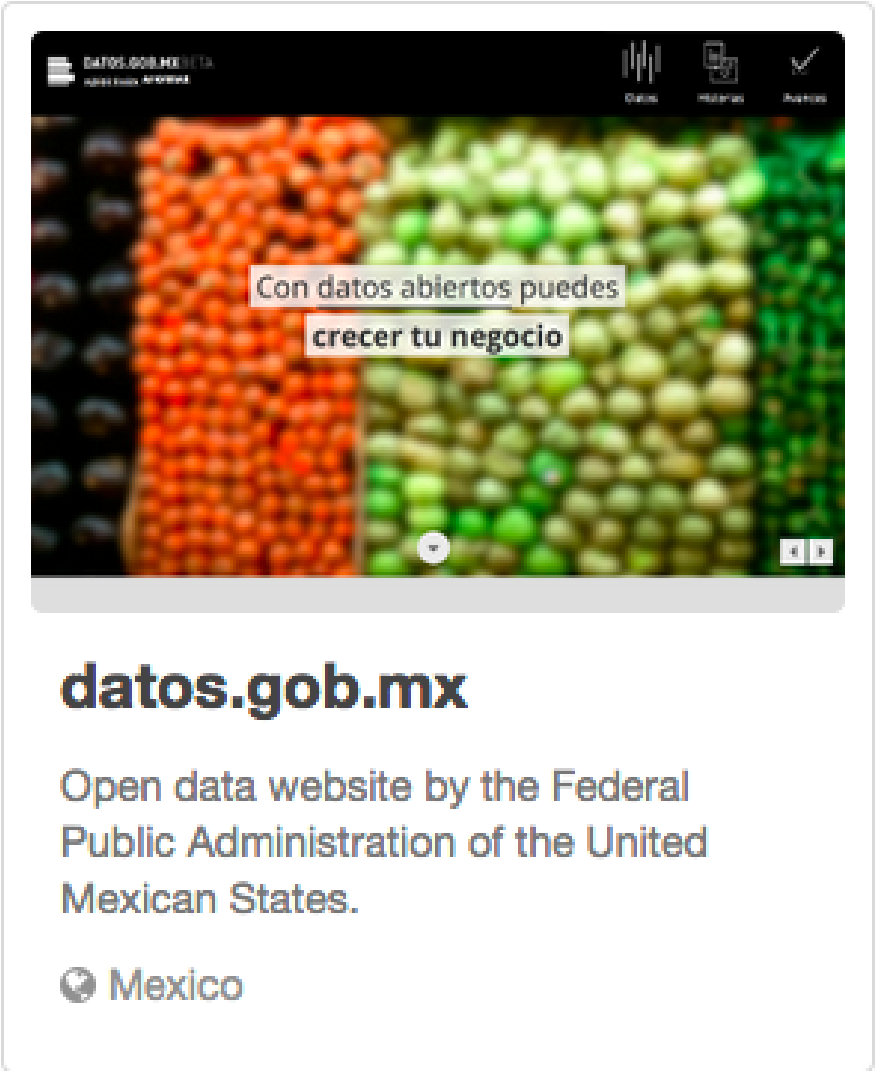
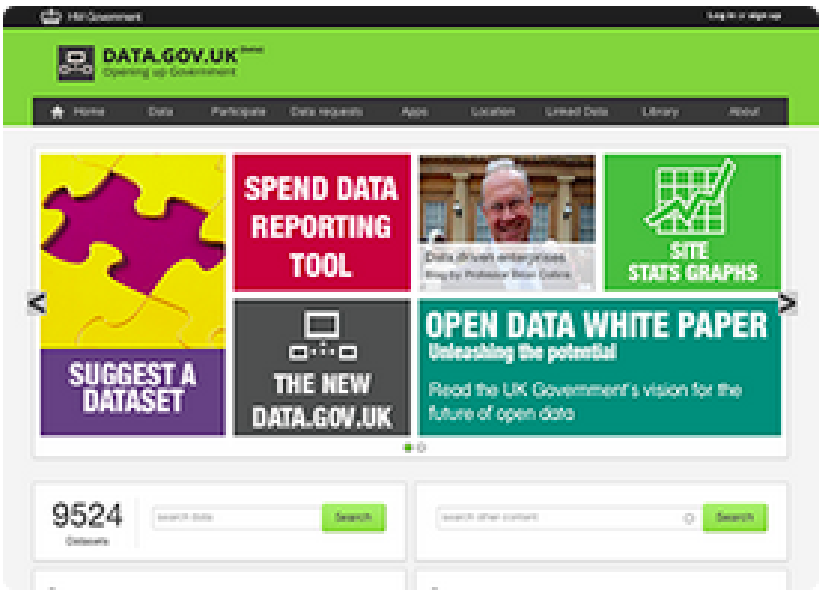


FIGURA D.8: Datos Abiertos del Gobierno de Montreal.



Data.gov.uk

The UK's http://data.gov.uk/ was one of the first government data portals on its launch in 2009, using Drupal to build the site and CKAN as a back-end. It was relaunched in June 2012 with data search pages now presented directly by CKAN.

United Kingdom

FIGURA D.9: Datos Abiertos del Gobierno del Reino Unido.

Data.gov

The US government announced in January 2013 that they were moving to CKAN for their open data catalogue, combining datasets previously published in different places. The relaunch went live in May 2013.

USA

FIGURA D.10: Datos Abiertos del Gobierno de Estados Unidos de América.

Bibliografía

[1] Open Knowledge. What is open?, Febrero 2015. URL https://okfn.org/opendata/.

[2] Open Government Data. Open government data, Diciembre 2014. URL http://opengovernmentdata.org/.

[3] Data.Gov. U.s. government open data, Enero 2015. URL http://www.data.gov/.

[4] Data.Gov. U.k. government open data, Enero 2015. URL http://data.gov.uk/.

[5] Data.Gov. Canada government open data, Enero 2015. URL http://open.canada.ca/en.

[6] Data.Gov. New zealand government open data, Enero 2015. URL https://data.govt.nz/.

[7] Data.Gov. Mexico government open data, Enero 2015. URL http://datos.gob.mx/.

[8] Open Knowledge. Open data handbook, Diciembre 2014. URL http://opendatahandbook.org/pdf/OpenDataHandbook.pdf.

[9] UK Government. Open data white paper unleashing the potential, Febrero 2015. URL https://www.gov.uk/government/uploads/system/uploads/attachmentdata/file/78946/CM8353acc.pdf.

[10] CKAN. The open source data software portal, Diciembre 2014. URL http://ckan.org/.

[11] Presidencia de la República. ¿ya conoces datos.gob.mx?, Junio 2014. URL http://www.gob.mx/presidencia/datos-abiertos/ya-conoces-datos-gob-mx/.

[12] Cowan D. ; Alencar P. ; McGarry F. Perspectives on open data: Issues and opportunities. *Software Science Technology and Engineering (SWSTE) 2014 IEEE International Conference on*, pages 24–33, June 2014. URL http://link.aip.org/link/?RSI/62/1/1.

[13] James Manyika; Michael Chui; Peter Groves; Diana Farrell; Steve Van Kuiken; Elizabeth Almasi Doshi. Open data: Unlocking innovation and performance with liquid information (2013), Enero 2015. URL http://www.mckinsey.com/insights/business_technology/open_data_unlocking_innovation_and_performance_with_liquid_information.

[14] D. Lathrop and L. Ruma. *Open Government: Collaboration, Transparency, and Participation in Practice*. O'Reilly Series. O'Reilly Media, Incorporated, 2010. ISBN 9780596804350. URL http://books.google.com.mx/books?id=9UibAgAAQBAJ.

[15] Barak Obama. Memorandum on transparency and open government, Enero 2009. URL https://www.whitehouse.gov/the-press-office/transparency-and-open-government.

[16] Michael McCandless; Erik Hatcher; Otis Gospodnetic. *Lucene in Action, Second Edition*. Manning Publications, Stamford, CT, 2010.

[17] Apache Lucene. Lucene powered by, Septiembre 2014. URL http://wiki.apache.org/lucene-java/PoweredBy.

[18] CKAN. Api guide, Diciembre 2014. URL http://docs.ckan.org/en/latest/api/index.html.

[19] Ian Ward. Python module for accessing the ckan action api, Octubre 2014. URL https://github.com/ckan/ckanapi.

[20] Apache Tika. Apache tika - a content analysis toolkit, Octubre 2014. URL https://tika.apache.org/.