Universidad del Bío-Bío
Facultad de Ciencias Empresariales
Departamento de sistemas de información

EXTRACCIÓN, MODELADO Y PROPUESTA DE ARQUITECTURA PARA LA GESTIÓN DE RECURSOS DE APRENDIZAJE Y OPINIONES RECUPERADOS DESDE EL REPOSITORIO KHAN ACADEMY

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL EN INFORMÁTICA

AUTORES: ELGUETA MORALES, JORGE A.
NEIRA SAN MARTÍN, ALEJANDRO E.

Profesor Guía: Vidal Castro, Christian L.

Concepción 2014

Resumen

Un repositorio es un sitio en el cual se almacenan objetos digitales, como pueden ser videos, animaciones, imágenes, documentos, entre otras cosas. Cuando estos objetos digitales adoptan una orientación académica, y además poseen una estructura de información detallada sobre su contenido (metadatos), se les denomina objetos de aprendizaje.

Al contar con esta información detallada de los objetos de aprendizaje, estos pueden ser localizados de mejor manera para ser utilizados en sistemas de e-learning (educación a distancia), o para realizar diversos procesos de análisis de información (análisis de sentimientos, minería de datos, entre otros). Además de contar con información que describe al recurso, para realizar estos procesos de análisis, es necesario contar con comentarios que permitan conocer la percepción de los usuarios sobre estos recursos, por lo que dentro de los repositorios existentes que se mencionan en el proyecto, se elige uno en particular que cumple con todas las características mencionadas: Khan Academy[1].

El objetivo de este proyecto es extraer los metadatos y comentarios correspondientes a los recursos educativos del repositorio Khan Academy, para generar un repositorio local basado en estándares existentes de organización de datos de repositorios, que facilite la realización de procesos de análisis de información contenida en objetos de aprendizaje.

Para cumplir con este objetivo, es necesario plantear la arquitectura que debe tener el repositorio. Una arquitectura es la estructura del sistema, que comprende sus componentes, las propiedades de estos componentes y las relaciones entre ellos. Un componente importante dentro de la arquitectura es la estructura que tendrán los datos en el repositorio. Para ello, se estudian los distintos tipos de modelado de repositorios existentes, donde se encuentran los estructurados (utilizan SQL), no estructurados (utilizan XML) y los semánticos (utilizan RDF/OWL). Tras este análisis, se propone una arquitectura híbrida donde coexistan los dos primeros bajo el estándar de LOM, dado que se representa a los recursos de mejor manera y además, facilita la búsqueda y actualización de contenido en el repositorio local. Dentro de los diseños de arquitectura más utilizados actualmente se utiliza el de arquitectura de tres capas (presentación, negocio y datos). Esta propuesta se orienta en la capa de datos, puesto que es la más

[1] Khan Academy: Repositorio educativo de objetos de aprendizaje en formato video. URL de acceso: https://www.khanacademy.org

relacionada a los recursos en el repositorio local.

Tras contar con la arquitectura, se hace necesario validarla con datos reales que puedan dar vida a esta nueva estructura. Para ello, se mejoró y adaptó un recolector de objetos de aprendizaje existente, haciendo que este nuevo recolector obtenga metadatos y comentarios de un recurso disponible en el repositorio Khan Academy, e inserte estos datos en las dos representaciones de modelo existente en el repositorio local bajo la arquitectura propuesta.

En términos generales, el principal aporte de este proyecto es dar el primer paso en procesos de análisis de información, contenida en objetos de aprendizaje, generando un repositorio local que incluya datos y comentarios de objetos de aprendizaje.

Agradecimientos

Después de un largo camino recorrido para poder lograr un gran paso en mi vida, se hace necesario poder agradecer a quienes ayudaron a hacer posible este logro.

Primeramente, quiero agradecer a Dios, pues sé que siempre me acompaña en mi camino y me ayuda a avanzar en todas las decisiones importantes de mi vida.

A mi madre, pues ha sido un apoyo incondicional, por su preocupación, sacrificio de muchos años para lograr que yo fuese un gran hombre, un hombre de bien, un profesional. Sé que velaste por mí desde donde estás y sé que nunca me dejarás solo en lo que me queda enfrentar hacia adelante. Sé que todo esto te será recompensado, porque eres una gran madre, que se merece lo mejor. Te Quiero Mucho.

A mi padre, quien sé me observa desde el cielo. Cuando partiste te prometí que lograría el gran sueño que tenías, verme como un profesional. Aquí estoy, cruzando a la próxima vereda. Te quiero mucho y extraño.

Cynthia, mi amor. Has estado junto mi durante todo este proceso, me has apoyado incondicionalmente en todo y orientado en muchas decisiones importantes, me has visto agotado y me has levantado. Me han enseñado a crecer y mirar la vida con otros ojos. Desde que te conocí supe que no me equivocaría contigo y lo que más quiero es poder compartir junto a ti este gran triunfo que nos abrirá el camino a nuestros sueños Te Amo Mucho.

A mis compañeros: Alejandro, gracias por tu esfuerzo y dedicación en este proyecto. También a Joel, Cristian, Renato, Paola y Héber, que siempre nos apoyamos independiente lo que pasara. Los estimo demasiado amigos.

A mi profesor guía Christian Vidal y a mi profesora co-guía Alejandra Segura, por su dedicación en este proyecto, su ayuda, su orientación y su simpatía. Es difícil encontrar profesores como ustedes.

Por último a mi Universidad, a la facultad de ciencias empresariales y al Departamento de Sistemas de Información, por facilitar la realización de este proyecto y de muchas actividades relacionadas a mi carrera.

Jorge Elgueta Morales

Agradecimientos

En este proceso muchas personas me ayudaron y apoyaron. Agradezco de todo corazón a mis padres, Jacqueline y Omar, quienes han sido fundamentales, reconocer su paciencia, sacrificio, apoyo que me brindaron y por haberme enseñado el valor de la perseverancia, la honestidad, respeto y ante todo la humildad. A mi hermana Andrea, por las alegrías y motivos para seguir adelante, estando siempre conmigo en las buenas y en las malas. Además sin dejar de lado a mi abuela, tíos Juan y Marcia; y mis primas/os quienes estuvieron conmigo apoyándome en todo momento. Y sin dejar de lado a personas que estuvieron conmigo en este proceso y ahora están apoyándome desde el cielo, como son mi tata José, y abuelita Raquel.

Quiero agradecer también, la ayuda y comprensión brindada por nuestro profesor guía Christian Vidal y nuestra Co-guía Alejandra Segura, donde ambos dedicaron tiempo y esfuerzo, para guiarnos y mostrarnos el camino que debíamos seguir para llevar esto adelante.

Tengo la fortuna de contar con buenos amigos, los cuales andábamos siempre los siete para todos lados, Jorge, Renato, Paola, Héber, Joel y Cristian, gracias por su amistad, compañerismo, fueron un sostén en el camino que llevamos recorrido. Además agradecer a mis otros compañeros que estuvieron conmigo durante estos años (Jonathan, Luis entre otros)

Del mismo modo quiero agradecer a la Universidad del Biobío, a la Facultad de Ciencias Empresariales y al Departamento de Sistemas de Información, que a través de sus directivos siempre me han brindado el apoyo necesario para cumplir esta primera meta.

Alejandro Neira San Martín

Tabla de contenido

Tabla de figuras

1 Introducción/Perfil

El ser humano se encuentra inmerso en una era en la que el conocimiento y la información implican el desarrollo social y la globalización. La información es el nuevo recurso en este desarrollo, a la cual hay que proporcionar la capacidad de acceso oportuno. Esta cantidad de información que caracteriza a la sociedad moderna impone más que nunca la necesidad de aprender; sin embargo, un volumen importante de información a aprender, la diversidad de contenidos, las distintas locaciones en las que se puede acceder y la velocidad requerida para hacerlo, dificulta la labor de quien requiere recuperar información.

Tradicionalmente, los textos impresos como libros, revistas, periódicos, etc. han sido los que han entregado la información requerida, es decir, para poder nutrirse intelectualmente era necesario asistir a una biblioteca o comprar textos en una librería. Con la llegada de los ordenadores, la información pasó a tener un formato digital y cuando se empezaron a construir las primeras redes de ordenadores, esta información empezó a ser compartida por distintas personas que utilizaban la misma red. Con el paso del tiempo, la innovación tecnológica ha conseguido que la difusión mundial del conocimiento y de la información sea algo natural.

Tanta es la información existente que puede difundirse en todo el mundo a través de la web, que para poder almacenarla y clasificarla de manera eficiente, nace la alternativa de utilizar repositorios digitales.

Los repositorios digitales, en el sentido más amplio de la definición, se emplean para almacenar cualquier tipo de material digital. La utilización de repositorios permite a los usuarios poder acceder a diversos contenidos, ya sea con fines educativos, entretención, laborales, etc., de una manera rápida y eficaz.

Esta innovación tecnológica que influye en la forma en que se difunde el conocimiento y la información a través del mundo, hace que la educación deba adaptarse a los requerimientos actuales. Es por ello que nacen los Sistemas de Gestión del Aprendizaje (Learning Management System - LMS), constituyendo una categoría de software que automatiza la administración de acciones de formación académica: gestión de usuarios, gestión y control de cursos, gestión de los servicios de comunicación, etc. Estos sistemas, además, gestionan los contenidos almacenados generalmente en

repositorios con orientación académica, contenidos que hoy en día son una pieza importante en la construcción del material docente.

Un repositorio o almacén digital de recursos educativos es una colección de recursos (objetos o unidades de aprendizaje) que son accesibles a través de una red de comunicaciones (IMS, 2003). El empleo de objetos o unidades de aprendizaje permite reutilizar los contenidos creados de una determinada experiencia educativa en contextos de aprendizaje diferentes. Por lo tanto, el objetivo de un repositorio educativo es facilitar la reutilización de objetos de aprendizaje, facilitando el acceso a los recursos almacenados en el mismo. Estas facilidades están relacionadas con la búsqueda de objetos de aprendizaje y con el acceso a los objetos de aprendizaje localizados.

Un usuario puede enfrentarse a diversos tipos de recursos en un repositorio educativo, siendo estos de diversa calidad, pudiendo encontrar material que realmente cumple con sus expectativas de búsqueda o incluso encontrar material que no tiene ninguna relación al respecto. Generalmente, para poder obtener información de los recursos respecto a su contenido, se debe realizar un proceso complejo. Por ejemplo, para saber si el contenido de una imagen es el que realmente se necesita, es necesario abrir el archivo y revisar si las formas, colores o trazos, corresponden con los criterios de búsqueda. Este proceso se hace más complejo si se desea revisar el contenido de un vídeo, pues es posible estar horas frente a él para darse cuenta que realmente no contiene lo que se busca. Hoy en día este proceso es mucho más fácil, debido a que existe información asociada a los recursos, y utilizando las herramientas correspondientes, se puede verificar ciertos datos de un recurso sin la necesidad de abrir y revisar el contenido. Esta información asociada se llama Metadatos.

Si bien, los metadatos entregan bastante información sobre el recurso (según el estándar en el cual se encuentra establecido), puede resultar compleja la búsqueda de contenido dado que los metadatos generalmente aportan datos técnicos del recurso. Es por la razón anterior, y a la vez por la evolución a redes sociales que se maneja en la actualidad, que los repositorios están comenzando a agregar en sus sitios la posibilidad de realizar comentarios acerca de los contenidos de los recursos. Esto, con el fin de poder ayudar a los usuarios finales a poder filtrar de alguna manera lo expuesto en el recurso.

Dado que la estructura de comentarios en los repositorios es relativamente nueva, es que se hace necesaria la incorporación de ellos a los estándares existentes, para

poder hacer mucho más fácil la navegación en los metadatos del recurso y así, facilitar la búsqueda del contenido solicitado.

Este proyecto establece la elaboración de un repositorio local que permite acceder a los comentarios de recursos educativos (además de otro tipo de información contenida en los metadatos) por medio de un archivo final accesible al usuario.

Para ello, es necesario modelar la estructura de la organización de los datos del repositorio, y a la vez, proponer una arquitectura que permita en un futuro la utilización de este repositorio en otros proyectos relacionados al análisis de contenidos.

1.1 Objetivos

A continuación, se exponen los objetivos tanto generales como específicos que busca cumplir el proyecto dentro de su periodo de desarrollo:

- Objetivo General.
  - Extraer los metadatos y comentarios correspondientes a los recursos educativos del repositorio Khan Academy, para generar un repositorio local basado en estándares existentes de organización de datos de repositorios, que facilite la realización de procesos de análisis de información contenida en objetos de aprendizaje.
- Objetivos Específicos.
  - Proponer el uso de uno o más modelos de organización de datos de repositorios, haciendo que el repositorio cumpla con albergar objetos de aprendizaje y facilitando en él la clasificación, interoperabilidad y búsqueda de contenido a través de los metadatos asociados.
  - Proponer una arquitectura de repositorio local que permita una posterior utilización de estos recursos, permitiendo la aplicación de técnicas avanzadas de análisis, clasificación y recuperación de información.
  - Extraer del repositorio de Khan Academy recursos educativos, específicamente el recurso en sí mismo (videos), sus respectivos metadatos (información de los recursos en formato XML) y los comentarios emitidos por los usuarios, con el fin de disponer de un repositorio local de recursos que permita validar la arquitectura propuesta.

1.2 Metodología utilizada

Las actividades y el método de trabajo a seguir para conseguir los objetivos mencionados con anterioridad son los siguientes:

- Etapa 1: En la primera fase, se define la problemática que aborda esta investigación, considerando la necesidad de disponer de un repositorio local que permita acceder a comentarios para su posterior tratamiento.
- Etapa 2: En esta fase, se estudia todo lo relacionado a la teoría de recursos educativos (objetos de aprendizaje) y su entorno de localización como lo son los repositorios. Aquí se menciona el repositorio a utilizar y su respectiva justificación, y se dan a conocer algunos antecedentes sobre él. Además, se echa un vistazo a aspectos importantes de la tecnología a utilizar a lo largo del proyecto.
- Etapa 3: Para establecer la propuesta de arquitectura, se revisan las alternativas de modelado de datos de repositorios existente, para luego seleccionar la o las que mejor cumplen con los objetivos del proyecto.
- Etapa 4: En esta fase estudia la teoría correspondiente a arquitectura de sistema. Se propone una arquitectura de acuerdo a las alternativas de modelado de datos de repositorios seleccionadas en la etapa anterior. Como el objetivo es proponer una arquitectura donde se representen los recursos de aprendizaje, la propuesta se enfoca en el detalle de la capa de datos.
- Etapa 5: Esta fase pretende realizar la aplicación de la recuperación de recursos desde el repositorio. Está compuesta de 3 sub etapas:
    - En la primera etapa, se estudia el prototipo de un recolector de objetos de aprendizaje existente, que tiene parte de la funcionalidad necesaria. Se analizan los problemas que enfrenta el prototipo, al cual se hace necesario adaptar y realizar mejoras para poder recuperar los recursos desde el repositorio Khan Academy.
    - En la segunda etapa, se detallan todas las mejoras y adaptaciones realizadas, además de los módulos nuevos. Además se detalla una interfaz de usuario para hacer más amigable la experiencia de uso.
    - Finalmente, se presenta un caso de estudio en el cual se puede apreciar la ejecución de la recuperación de videos desde el repositorio, siendo estos almacenados en el repositorio local y creando el archivo XML disponible para el usuario.

- Etapa 6: Finalmente se dan a conocer los proyectos futuros en los cuales es posible utilizar la aplicación y los datos generados por este proyecto, para luego dar las conclusiones finales del proyecto.

1.3 Problemática

Para poder realizar procesos de análisis de información (análisis de sentimientos, minería de datos, entre otros) contenida en repositorios educativos, a un nivel de percepción del usuario, es necesario contar con comentarios. Lamentablemente, muchos repositorios no almacenan los comentarios de usuario en la estructura de metadatos de los recursos. La gran mayoría de los repositorios educativos presenta sus recursos de la forma: Archivo – Metadatos, sin considerar la percepción de los usuarios respecto a estos.

Es por ello dentro de los repositorios educativos más importantes (analizados a fondo en el apartado 2.2.1) es que se ha seleccionado el repositorio de Khan Academy para realizar este proyecto; debido a que además de presentar la información de una forma adecuada para la aplicación de las técnicas mencionadas, es de los pocos que mantiene un panel de comentarios relacionados al contenido del recurso.

Ahora bien, para obtener la representación de los recursos de la forma adecuada: Archivo – Metadato – Comentario, se hace necesario ubicar el recurso educativo, extraer sus metadatos y acceder al panel de comentarios para individualizar cada uno de ellos y asociarlos al recurso educativo. Tener toda esta información sin un orden correspondiente puede generar inconsistencia de datos y pérdida de información, es por ello que se propone la creación de un repositorio local que pueda albergar toda esta información de manera ordenada y que a la vez gestione los objetos de aprendizaje de forma rápida y útil.

De acuerdo a la forma como se organizan los datos dentro de un repositorio, muchas veces que se busca un objeto de aprendizaje, el repositorio que lo contiene se enlaza con la página del autor, donde el material puede cambiar de enlace y volverse ilocalizable. Para ello, es necesario que la arquitectura que se propone para el repositorio local garantice el acceso al material educativo en todo momento, independiente del medio físico y la tecnología de almacenamiento que posea el repositorio. Además, el repositorio debe proporcionar apoyo en la búsqueda de nuevos recursos de forma efectiva, mediante el uso de categorías, como por ejemplo tema, área, autor, etc. Los filtros de las búsquedas deberán cumplir con entregar los objetos de aprendizaje que tengan mayor coincidencia con los filtros y es ahí donde se hace necesario un modelado

acorde con la arquitectura propuesta.

El tema de los repositorios es un tema emergente en proyectos de título en la Universidad del Biobío y que no ha sido trabajado ampliamente en la región. Además, existen pocos trabajos que se relacionen con la extracción de recursos educativos y sus correspondientes comentarios, por lo que este proyecto logra aportar nuevo conocimiento con respecto a la extracción de material desde repositorios educativos y a la creación de un repositorio local basado en estándares de e-learning.

2 Estado del arte

En este capítulo se presenta una visión actual del estado de las investigaciones y trabajos relacionados a la extracción, modelado y propuesta de arquitectura para un repositorio local, de recursos alojados en Khan Academy.

La presentación de estos aspectos se centra en los aspectos importantes para el cumplimiento de los objetivos del proyecto y que servirán además para entender el contexto general y específico de la tesis.

2.1 Objetos de aprendizaje

Según el estándar LOM de la IEEE [IEEE, 2002], un Objeto de Aprendizaje es cualquier entidad digital o no digital, que pueda ser usada para aprender, educar o enseñar. Según esta amplia definición, textos impresos de estudio, documentos digitales, herramientas generadoras de test, presentaciones y videos entre otros podrían ser consideradas como Objetos de Aprendizaje. Una definición más concreta es la de McGreal [McGreal, 2004] que los define como “cualquier recurso digital reusable que tiene encapsulado una lección o ensamblado un grupo de lecciones en unidades, módulos, cursos e incluso programas”.

Un objeto de aprendizaje es almacenado generalmente en un repositorio de objetos de aprendizaje a modo de colecciones [IMS, 2003], donde es posible facilitar su localización y utilización. Para realizar búsquedas de objetos de aprendizaje, basados en su contenido, se utilizan los metadatos asociados.

2.2 Repositorio

Un repositorio, depósito o archivo es un sitio web centralizado donde se almacena y mantiene información digital, habitualmente bases de datos o archivos informáticos. Pueden contener los archivos en su servidor o referenciar desde su web al alojamiento originario.

Formalmente, un repositorio o almacén digital de recursos educativos es una colección de recursos (objetos o unidades de aprendizaje) que son accesibles a través de una red de comunicaciones. La estructura de la colección es invisible para el usuario, quien solo se limita a esperar un resultado. El repositorio puede contener los propios recursos o únicamente los metadatos que los describen, junto con una referencia para su localización [IMS, 2003].

Un repositorio puede ser de acceso público, o pueden estar protegidos y necesitar de una autentificación previa. Los repositorios de acceso público se pueden clasificar en:

- Institucionales: Documentos administrativos, colecciones especiales, trabajos de investigación, entre otros, que son de pertenencia de la institución pero disponibles para todo público. Ejemplos: Repositorio de la escuela politécnica militar, Repositorio de datos del gobierno de Chile, entre otros.
- Temáticos: Acceso a contenidos de una disciplina o área temática. Dirigidos por instituciones académicas o de investigación. Ejemplos: Merlot, Laclo, Khan Academy, entre otros.
- Agregadores: Recogen su contenido en repositorios institucionales y lo dejan disponible para un fácil acceso. Ejemplo: Google Académico.
- De datos científicos: Almacenan y conservan datos científicos generados en investigaciones (y que pueden dar lugar o no a publicaciones científicas). Ejemplo: E-ciencia.
- Huérfanos: Contienen datos que no están albergados en repositorios mencionados anteriormente. Ejemplo: Webs personales de los autores.

Los repositorios más conocidos son los de carácter académico e institucional y tienen por objetivo organizar, archivar, preservar y difundir la producción intelectual resultante de la actividad investigadora. Algunos de ellos son: Merlot, Laclo, Agora, Khan Academy, entre otros.

2.2.1 Repositorio educativo

Un repositorio educativo contiene documentos, videos, audios, imágenes, etc. sobre distintos temas o sobre algunos en específico, según la orientación educativa o de investigación que mantiene la entidad a cargo.

Sin contar los repositorios gubernamentales o de bibliotecas públicas con contenido on-line, los repositorios, en su mayoría son utilizados para difundir el contenido educativo de asignaturas en específico, departamento o facultad de alguna institución. Generalmente están protegidos o son de exclusivo acceso de personas autorizadas.

Ahora bien, existen proyectos de repositorios temáticos abiertos a todo público, que poseen de todo tipo de material, y que se han hecho famosos y son bastante utilizados por personas de todo el mundo dada la calidad del contenido y la constante actualización que brindan los propios usuarios. Todos ellos varían en la forma de presentar el material o en el tipo de contenidos que albergan. Por ejemplo, algunos solo contienen documentos científicos, otros contienen solo videos educativos y otros contienen enlaces a imágenes, videos o audios.

2.2.2 Tecnologías en un repositorio educativo

Un repositorio permite la utilización de diversas herramientas para poder compartir la información. Por ejemplo, un repositorio puede almacenar meta-metadatos como formatos de datos permitidos para las definiciones de registros y campos, los metadatos como registro específico y definiciones de campo, y los datos en sí, y estos pueden ser compartidos por diversas herramientas: compiladores y depuradores, procesadores de consultas de bases de datos, gestores de formularios y redactores de informes. Para que la representación de la estructura del repositorio (modelo de datos del repositorio) se ajuste a un formato de datos admisibles, y a la vez se conserve la integridad de la información, las herramientas pueden compartir datos y metadatos sin estar bien informados sobre el funcionamiento interno de otras herramientas. En este sentido, un repositorio es como un diccionario de datos.

El manejo de los datos en un repositorio está a cargo del administrador del repositorio. Este administrador utiliza una API (Application Programming Interface) que permite:

- Incorporar las operaciones de tipo específico que encapsulan las herramientas que es posible utilizar. (Por ejemplo, editar un documento, aprobar un diseño, revisar y extraer datos del archivo, etc.).
- Manejar una jerarquía de herencia para compartir información de datos. (Por ejemplo, si a partir un recurso en video que dura diez minutos se genera un video con el mismo contenido, pero más corto, se modificarán solamente los campos referidos a la duración y fecha de creación/modificación, pero los campos que hacen referencia al contenido se mantendrán iguales, por ende se dice que se heredan esos datos).

- Construir nuevos tipos de objetos, sujetos a ciertos controles de integridad y creación.

Por otro lado, la información en el repositorio está sujeta a servicios de control de manera frecuente, lo que hace más fácil (en algunos casos) la utilización de dichas herramientas. La base de datos de un repositorio está sujeta a los controles típicos de base de datos, tales como la integridad, la concurrencia y el control de acceso. Además, un sistema de repositorio proporciona chequeo de salida de información, versiones y control de la configuración, la notificación, la gestión del contexto y de flujo de trabajo.

2.2.3 Repositorios educativos más importantes

Es muy importante diferenciar un repositorio educativo de un portal. La principal característica que poseen los repositorios educativos es que los recursos que se albergan allí son objetos de aprendizaje, esto quiere decir, que están descritos por metadatos asociados. En un portal, es posible encontrar un recurso requerido, pero no contiene mayor información al respecto más que el contenido de este.

A continuación, se mencionan los repositorios educativos más importantes, dado el nivel de visitas a nivel mundial y la calidad de los contenidos de los recursos.

- MERLOT (Multimedia Educational Resource for Learning and Online Teaching): Repositorio de la universidad de California en los Estados Unidos centrado en recursos universitarios. Es un repositorio libre y gratuito diseñado principalmente para estudiantes de educación superior, donde pueden dejar y encontrar recursos de aprendizaje “on-line. En él se puede inspeccionar su catálogo de contenidos relacionados con campos como negocios, arte, humanidades, ciencia y tecnología, ciencias sociales. También se pueden realizar búsquedas sobre el mismo; para ello, los contenidos suelen llevar una pequeña descripción mediante la utilización de algunos metadatos como la descripción, autor, audiencia, idioma, coste, derechos de autor, etc.

- LACLO: Esta es una comunidad abierta, integrada por personas e instituciones interesadas en la investigación, desarrollo y aplicación de las tecnologías relacionadas con Objetos

de Aprendizaje en el sector educativo Latinoamericano. Su principal misión es ayudar a la articulación de los diferentes esfuerzos en la Región para diseminar los avances y beneficios de esta tecnología, a fin de que Latinoamérica pueda hacer frente al gran reto educativo de este siglo: poder ofrecer recursos educativos personalizados y de calidad a cualquier persona, en cualquier momento y en cualquier lugar.

- Texas A&M Repository: La Universidad de Texas A&M proporciona acceso a las colecciones digitales y servicios académicos en apoyo de la investigación, la enseñanza y el aprendizaje. Estos servicios se prestan en colaboración con la Biblioteca de Texas Digital (TDL), un consorcio de varias universidades de los cuales Texas A & M es miembro fundador. El repositorio provee de acceso a colecciones de imágenes históricas, videos, mapas, entre otros.

- ARIADNE: Se trata de una red europea de recursos educativos distribuidos, alrededor de la cual se han creado una serie de herramientas que ayudan a la compartición y reutilización del material educativo. La principal ventaja que presenta ARIADNE es la posibilidad de hacer búsquedas en otros repositorios externos, lo que también se llama búsquedas federadas.

- AGORA: abreviatura de “Ayuda a la Gestión de Objetos Reutilizables de Aprendizaje”, es una plataforma donde las actividades y tareas relacionadas con la gestión de los recursos educativos y los objetos de aprendizaje están integradas, ofrece además un espacio para intercambiar recursos educativos. Este tiene como finalidad asistir al profesor en el proceso de búsqueda y construcción de recursos de aprendizaje, conforme a sus necesidades de diseño instruccional, a partir de recursos digitales y utilizando las tecnologías más actuales. En este proyecto hubo participacion de profesionales españoles, mexicanos y chilenos (los

cuales pertenecen a la Universidad del Bío Bío). Esta plataforma consta de cuatro subsistemas, las cuales son:

- Gestión de objetos de aprendizaje.
- Modelos de conocimiento para diseño instruccional.
- Meta buscador de Objetos de Aprendizaje.
- Sistema de recomendación.

- Khan Academy: Con la misión de "proporcionar una educación de nivel mundial para cualquier persona, en cualquier lugar", es una organización de aprendizaje electrónico en línea gratuita con más de 4 500 vídeos dirigidos a escolares de enseñanza primaria y secundaria sobre matemáticas, biología, química, física, e incluso de humanidades como finanzas o historia.

2.3 Khan Academy

A continuación se deja en claro por qué se ha seleccionado a Khan Academy como repositorio de estudio y se presentan antecedentes sobre la historia y funcionamiento del mismo.

2.3.1 Elección del repositorio Khan Academy

Como el objetivo de este proyecto es la recuperación de recursos educativos junto a sus respectivos comentarios, se ha seleccionado el repositorio Khan Academy, ya que de los repositorios más importantes vistos anteriormente, es el que cumple con presentar los recursos junto a sus respectivos comentarios. Estos comentarios además son relevantes al contenido del recurso, mientras que otros repositorios que manejan comentarios, estos solo son referidos a los enlaces de descarga de contenidos y no se asegura su relación al contenido del recurso como sí lo hace Khan Academy.

La Academia Khan (en inglés Khan Academy) es una organización educativa sin ánimo de lucro y un sitio web creado en 2006 por el educador estadounidense Salman Khan, un graduado del MIT y de la Universidad Harvard.

2.3.2 Historia

El fundador de la organización, Salman Khan, hijo de inmigrantes de India y Bangladesh, trabajaba en Boston como analista de finanzas cuando recibió la visita de sus tíos y primos, aún residentes en su Louisiana natal. A finales de 2004, Khan comenzó ayudando a su prima Nadia con una tutoría en matemáticas utilizando una herramienta de Yahoo!. Cuando otros niños en la familia pidieron una ayuda similar y Khan comenzó a dedicar cada vez más tiempo a sus alumnos, un amigo le sugirió grabar las clases y colocarlas en YouTube.

Para sorpresa de Khan comenzaron a llegar mensajes de agradecimiento de usuarios inesperados, desde la madre de un niño autista que progresaba con las lecciones online, hasta adultos que ingresaron a la universidad después de abandonar sus estudios y una niña en un orfanato de Mongolia en el que ingenieros voluntarios habían instalado computadoras.

En 2009, Khan decidió abandonar su trabajo para dedicarse a las lecciones. Con escasos fondos, el joven recibió la visita de una filántropa estadounidense, Ann Doerr, quien donó al proyecto los primeros US$100.000. Poco después Bill Gates mencionó en el Festival de Ideas de 2010 en Aspen que usaba los cursos de Khan Academy para enseñar a sus hijos. La Fundación Bill y Melinda Gates acabó convirtiéndose en uno de los principales patrocinadores del proyecto.

Hoy en día, Khan Academy es uno de los repositorios educativos más grandes e importantes a nivel mundial, con más de 4500 videos en inglés y muchos de ellos están siendo traducidos a otros idiomas, entre ellos el español.

2.3.3 Funcionalidad proporcionada

Un usuario cualquiera puede conectarse directamente a la página de Khan Academy (http://www.khanacademy.org) y luego buscar videos por nombre o temática en la opción “Learn” (aprender).

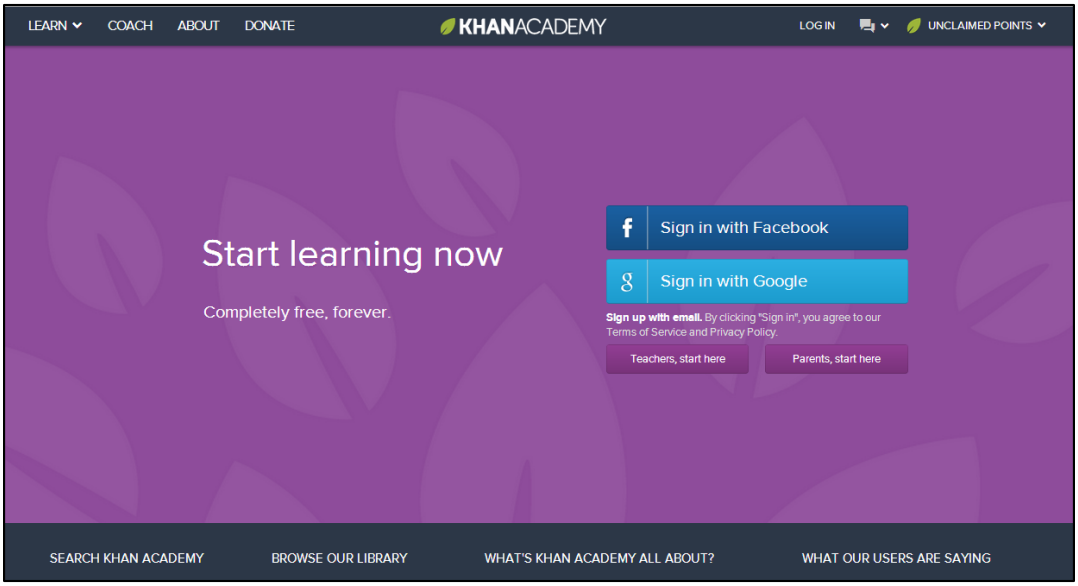
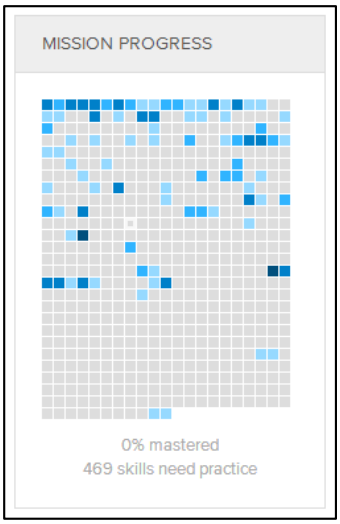


Ilustración 1: Sitio oficial de Khan Academy.

Un usuario registrado cuenta con la misma forma de buscar videos por nombre o temática, pero además tiene asociado un perfil en el cual se van ganando puntos haciendo ejercicios (matemáticos, alternativas, verdadero y falso, entre otros).



Si el usuario desconoce cierta temática, se le presenta el video asociado a ese contenido para reforzarlo y poder realizar nuevamente el ejercicio. Las lecciones cubren materias básicas como matemáticas, física, biología y arte. Estas materias se revisan en micro lecciones que permiten a cada estudiante avanzar a su ritmo y es el propio alumno el que va "escalando andamios", pasando al nivel siguiente sólo cuando domina el anterior.

Esta metodología permite que muchos profesores puedan ayudarse con el sitio web, haciendo que los alumnos estudien las lecciones online en casa y durante clase hacen ejercicios. Los maestros monitorean en una planilla electrónica cómo avanza cada alumno en su computadora y ve qué alumno "se tranca" y requiere más atención individual del profesor.

2.3.4 Estructura del sitio

La organización de los contenidos en el sitio es muy simple. Para usuarios no registrados, se muestra una barra superior en la cual se encuentran todos los links correspondientes para encontrar el contenido (ordenado por áreas del conocimiento). Para usuarios libres se muestras opciones de ingreso, registro e información sobre el sitio. Para usuarios registrados, se despliega la información de la cuenta y el avance de conocimientos en los contenidos. En la barra izquierda, se encuentra una lista con todos los videos relacionados al contenido solicitado, presentados por orden de conocimiento y nivel de dificultad.

El área principal contiene el recurso educativo: El video. En la parte inferior, inmediatamente después del video, se encuentra un panel de preguntas, tips y agradecimientos, donde se muestran las interacciones que han registrado los usuarios. (No es posible realizar comentarios ni votaciones, a menos que se encuentre conectado con una cuenta de usuario o Facebook).

Finalmente, la parte inferior contiene información acerca del sitio e información de contacto.

La siguiente ilustración muestra lo señalado con anterioridad:

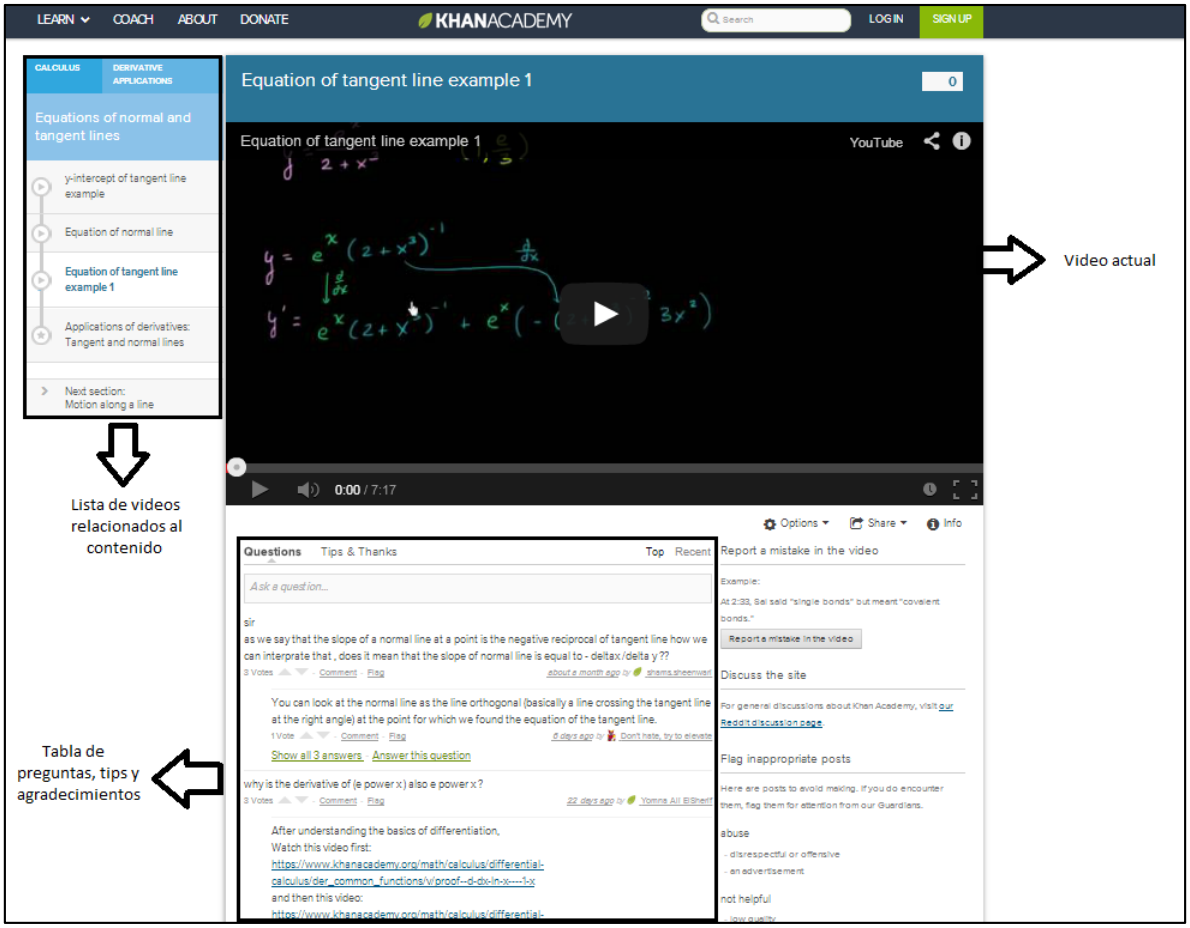


Ilustración 2: Disposición de los elementos en el sitio de Khan Academy.

2.4 Tecnologías necesarias para el desarrollo del proyecto

Dentro del proyecto se hace necesario utilizar muchas tecnologías que permitirán cumplir con los objetivos propuestos. Dentro de las más importantes, destacan las siguientes:

2.4.1 XML

XML viene de la sigla e**X**tensible **M**arkup **L**anguage, que significa "lenguaje de marcas extensible". Este fue desarrollado por el World Wide Web Consortium

(W3C). XML es un lenguaje de marcado para documentos que contienen información estructurada. XML da soporte a bases de datos, siendo útil cuando varias aplicaciones se deben comunicar entre sí o integrar información.

Con este lenguaje se transforman datos en información, ya que al estar estructurado, se le añade un significado concreto y se le asocia a un contexto. XML consta de elementos, y estos a su vez pueden tener otras partes, por lo cual se tiene una especie de árbol de trozos de información. Estas partes tienen la siguiente estructura: <Nombre> donde Nombre es el nombre que se le quiera colocar. De forma formal un documento XML consta con el prólogo, el cual no es obligatorio, donde se describe la versión XML, el tipo de documento etc. Luego de esto se tiene el cuerpo, el cual debe estar contenido en solo un elemento raíz; para poder mantener el documento bien formado. Los elementos XML pueden contener más elementos, ya sea un elemento propiamente tal, caracteres o ambos. Además se le puede incorporar características o propiedades a los elementos, y a esto se le llama atributos. Otros elementos de un documento XML son las entidades predefinidas, para representar los caracteres especiales, que no son interpretados como marcado en el procesador XML. Secciones CDATA esto sirve para especificar datos utilizando cualquier carácter sin que se interprete como marcado XML. Y por último otra parte que siempre sirve son los comentarios, el cual se denota como <!---Comentario--->.

Existen diversos estándares que permiten describir un objeto de aprendizaje, como por ejemplo LOM, LOM ES, CANCORE, DUBLIN CORE, CROSSREF, entre otros. XML es utilizado ampliamente para apoyar la representación de estos.

2.5 Conclusiones del estado del arte.

- Existen muchos recursos disponibles en repositorios. Estos repositorios pueden ser locales o web, y además se pueden clasificar según la orientación de su contenido. Dentro de estas clasificaciones, los repositorios más importantes son los de carácter educativo.
- Se usará el repositorio Khan Academy ya que, entre los repositorios más comunes en el ámbito educativo, es uno que facilita la recopilación de sus recursos. Esta facilidad está dada gracias al acceso público por medio de una API para conectarse a su servidor y poder recuperar los metadatos. Además, consta con un

apartado donde se puede realizar comentarios sobre el contenido del recurso, a diferencia de otros repositorios donde se utiliza el apartado de comentarios para mencionar problemas con el link de conexión o problemas con la ubicación del contenido.

- XML es una tecnología altamente relacionada con los objetos de aprendizaje, ya que los metadatos asociados a estos recursos son identificados de una manera más rápida y eficiente gracias a los identificadores utilizados por esta tecnología.

3 Modelado de un repositorio

Antes de proponer una arquitectura para la creación del repositorio local, es necesario realizar un análisis a las alternativas de modelado de repositorios existentes. Esto, pues la arquitectura es la estructura del sistema, en este caso el repositorio, que consta de componentes que la componen, las propiedades de sus componentes y cómo interactúan entre ellos. Dentro de los componentes de una arquitectura, el más importante a considerar en este proyecto es el modelado que tendrán los datos, ya que un modelo permite comprender la organización de los datos asociados a los recursos, pues se muestra la relación que existirá entre los objetos que se describirán en él. En este caso particular, es necesario para comprender la forma en que los recursos educativos estarán almacenados en el repositorio local y así facilitar la ubicación de la información existente según las categorías que se puedan presentar.

La decisión de adoptar una (o más de una) de las alternativas de modelado de repositorios, debe estar enfocada en el cumplimiento de los objetivos del proyecto, pensando que la orientación con la cual se va a crear el repositorio local será el trabajo con objetos de aprendizaje y sus correspondientes comentarios para facilitar la realización de procesos de análisis de información contenida en recursos educativos.

Entre las alternativas de modelado para repositorios, podemos encontrar:

- Repositorios estructurados.
- Repositorios no estructurados.
- Repositorios semánticos.

3.1. Alternativas de modelado para repositorios

En este apartado se analizaran las ventajas y desventajas de las alternativas de modelado existentes para repositorios. Entre las alternativas se encuentran: estructurado, semiestructurado o semántico.

3.1.1. Repositorios estructurados o relacionales

Los repositorios estructurados, ocupan una base de datos relacional, que facilita la flexibilidad en su estructura de datos. Las características principales de este tipo de repositorio son:

- Los motores de gestión son demasiado pesados, además de que de acuerdo a la plataforma a utilizar, se debe seleccionar los que son compatible con estos.
- El lenguaje de consulta es SQL, el cual está bastante maduro, tiene una base matemática que respalda su funcionamiento. Además de ser unos de los lenguajes más utilizados mundialmente.
- Facilita el acceso a partes de la base de datos, debido a su estructura que permite agrupar los datos similares, por lo tanto pueden ser relacionados entre sí, en caso de que sea necesario.
- Tiene flexibilidad en el acceso a las partes de la base de datos, la cual permite al usuario extraer en el formato más apropiado para ser mostrado, de acuerdo al dato exacto que él quisiera.
- Evita la replicación de los datos, debido a que se guarda la información correspondiente en tablas, en las cuales son muy simple la utilización de restricciones de integridad.
- Al guardar toda la información en una base de datos relacional, por lo cual no es necesario guardar el recurso o metadatos originales en otro lugar diferente.
- Al estar estructurado disminuye la complejidad de obtener información de ella.
- Puede ser actualizada por diversas personas, pero esto no perjudica la seguridad, debido a que los gestores soportan los permisos de acceso.
- Una problemática que se tiene al basarse en este tipo, es que las base de datos relaciones no tiene la capacidad para manejar aplicaciones graficas o tipos de datos especiales, asimismo con aplicaciones que manejen datos complejos interrelacionados.
- Al momento de trabajar con archivos XML estos deberán ser divididos para su posterior almacenaje con objetos SQL (tablas, columnas, tipos, etc.) lo cual si no se tiene bien definido la estructura en ambos sentidos puede llevar a inconsistencias de los datos.

3.1.2. Repositorios semiestructurados o XML

Un repositorio XML permite trabajar con grandes grupos de documentos de XML, en la cual será necesario un gestor de bases de datos con manejo especializado de este tipo de archivos. Las características principales de este tipo de repositorios son:

- Los lenguajes que más se utilizan para realizar las consultas son XQuery o XPath pero estos son mucho más complejo de manejar, en comparación con SQL.
- La manera de operar es las consultas es a través de una secuencia de datos en XML y lo que devuelve como resultado es otra secuencia en XML.
- A los documentos XML que están guardado se puede realizar a través de XMLParse o con alguna otra función propia del lenguaje consultas.
- El XQuery está basado en XPath y de él fundamente la selección de información y la iteración que realiza a través del conjunto de datos, este mismo lenguaje ofrece una alternativa para realizar transformación de XML a su representación en HTML o PDF en caso de que sea necesario.
- Se puede cambiar la estructura completa, cuando sea necesaria una forma de organización de la información diferente.
- Aquí se mantiene todo el contenido de un documento en un lugar fácil de administrar, lo que realiza que su búsqueda sea fácil y no es necesario de preocuparse por grupo de archivos o por la estructura que tendrá, ya que todo se encuentra en la base de datos y solo se hace necesario extraer la información del XML que tiene la base de datos.
- Al contar con este tipo se tiene como ventaja que puede ser visto de diferentes formas, la cual varía de acuerdo a la necesidad del usuario.
- Cuando las consultas están bien diseñada e implementada es mucho más rápido la relación de la consulta, en comparación a los otros tipos de base de datos y esto se debe a que puede realizar cualquier método de indexación para que trabaje de una manera más rápida, es decir si se busca un documento en especial se busca el identificador en la tabla y ya se tiene el documento, en cambio sí es en XML puro debe buscar en cada parte de los elementos hasta encontrarlo.
- Una gran ventaja es que al momento de almacenar un documento este es pre analizado, por lo cual aumenta el desempeño posteriormente, ya que no se revisar cada documento, ni crea un modelo que represente los documentos que se consulten.
- Se puede mantener un gran volumen de datos, ya que se tiene una característica de respaldo de documentos, además se puede incorporar herramientas que realicen respaldos cuando se llegue a un determinado tamaño, para no bajar el rendimiento.

- Es de fácil acceso al documento original (las cuales tienen asociado un identificador por lo que son reconocido dentro del repositorio), en cambio sí un XML se guarda en una base de datos relacional, esta debe ser dividida y al no realizarlo bien, puede crear inconsistencia, y de esta misma forma en ese tipo para extraer el original se debe re ensamblar todas las partes y puede que se obtenga uno totalmente diferente, ya que dentro del etiquetado se puede tener elementos repetidos.

3.1.3. Repositorio semántico

Este tipo de repositorios de objetos de aprendizaje, permite incluir expresiones semánticas capaces de ser procesadas por una máquina dentro de los campos de los registros de metadatos de los objetos de aprendizaje. Este tipo de repositorios se basa en la identificación de objetos por medio de ontologías, lo que permite relacionar nuevo conocimiento para poder utilizar en las búsquedas o en la asociación de información a los recursos según se vaya añadiendo más información al repositorio. Dentro de las características principales se encuentran las siguientes:

- Los repositorios semánticos, tienen la ventaja de que no utilizan una base de datos relacional con SQL, sino ocupan RDF, el cual es un modelo de datos para la Web Semántica, donde se representa la información en la Web de manera entendible por los computadores.
- RDF es la representación principal. Para ampliar su poder expresivo y representar de la forma más potente posible el conocimiento, se pueden utilizar otros lenguajes como OWL (Lenguaje de etiquetado semántico para publicar y compartir ontologías en la Web, es una extensión del lenguaje RDF y emplea sus tripletas, aunque es un lenguaje con más poder expresivo que éste).
- De igual manera que los anteriores es necesario un motor de consultas, por ejemple cuando se utiliza el framework Jena, podría ocuparse ARP como motor, el cual permite tipo de consultas a esta representación.
- El lenguaje más común para realizar consultas es SPARQL y los resultados de estas pueden tratarse como ResultSets (set de resultado) o como gráficos de RDF.

- Las ontologías juegan un importante rol para el soporte de un modelo semántico sólido que cumpla con los nuevos requisitos que la flexibilidad que sugiere la nueva generación.
- Una de las grandes ventajas de este tipo es que al tratar con ontologías se puede generar un nuevo espectro de posibilidades de inferencia sobre los registros que contienen la información de metadatos de los objetos de aprendizaje, ya que existe la posibilidad de definir las relaciones semánticas dentro de la información descrita.
- El tamaño necesario es menor, en comparación a la base de datos relacional, pero es demasiado costoso realizar búsquedas en él.

Hoy en día se ha investigado bastante sobre este tipo de repositorios, siendo los trabajos realizados por el profesor Jesús Toro Carrión, de la Universidad de Salamanca, España los que mayor aportes han entregado.

3.2. Elección de la mejor alternativa de modelado

Al analizar las tres alternativas existentes, se concluye que la mejor alternativa para representar de mejor manera un repositorio, es a través de un modelo híbrido en el cual coexistan el modelo estructurado y el semiestructurado. Esto es debido a que se adoptarían las ventajas de ambos modelos.

En cuanto al modelo estructurado, se elige debido a la madurez del lenguaje de consulta que se puede realizar en él, lo cual facilita las búsquedas aproximadas, a pesar que también se adoptan las desventajas de este tipo de modelo. Para solucionar las carencias que conlleva el uso del modelado estructurado, se propone su utilización en conjunto con la representación semiestructurada.

En cuanto al modelado semiestructurado, se tiene como gran ventaja la facilidad de traspaso de un modelo a otro, como por ejemplo ir desde un XML a una representación en RDF/OWL. Además, trabajar con este modelo, facilita las búsquedas exactas al momento de consultar en el XML. Al trabajar con esta tecnología, se facilita la transferencia de información, la cual se realiza de manera segura, fácil y fiable, lo cual es debido a la compatibilidad entre los sistemas con esta.

Mantener un repositorio que albergue dos representaciones distintas brinda las ventajas de ambas como se menciona en el párrafo anterior, pero a la vez, se requiere de un gran esfuerzo para mantener siempre actualizados y coherentes ambos modelos que coexisten.

3.3 Elección del estándar para describir objetos de aprendizaje en el repositorio.

Existen diversos estándares que permiten describir un recurso educativo u objeto de aprendizaje, como por ejemplo LOM, LOM ES, CANCORE, DUBLIN CORE, CROSSREF, entre otros. La gran mayoría de estos tienen en común que trabajan con XML como lenguaje de marcado para identificar la información del recurso por intermedio de metadatos. Las variantes existentes entre ellos tienen que ver con la cantidad de categorías que contienen y la información a la cual hacen referencia.

Dentro de los estándares mencionados, el más utilizado a nivel mundial es LOM (Learning Object Metadata), ya que es el estándar más completo y que mejor describe a los objetos de aprendizaje. Además, es uno de los pocos que se encuentra con el respaldo de una sociedad internación de estandarización.

La elección del estándar LOM para representar los objetos de aprendizaje en el repositorio local está dada por la cantidad de usuarios que utilizan el estándar. Esto ya que permite que la representación del repositorio sirva ampliamente a cualquier usuario y puedan reutilizarse objetos de aprendizaje ya sea desde el repositorio local hacia el exterior y viceversa. Además, al ser el estándar con mayor completitud de categorías, se hace más eficiente el acceso a la información requerida.

3.3.1 LOM

Proveniente de la sigla Learning Object Metadata, este es un estándar (IEEE 1484.12.1:2002), el cual define los atributos necesarios para describir completamente y adecuadamente un objeto de aprendizaje a través de un esquema de metadatos.

Su propósito es ayudar a la reutilización de objetos de aprendizaje y facilitar su interoperabilidad, usualmente en el contexto de sistemas de aprendizaje on-line. La utilidad de un esquema de metadatos radica en su acepción por una comunidad suficientemente amplia de productores y consumidores de material educativo.

El esquema de metadatos que establece el estándar se da a conocer en la siguiente ilustración:

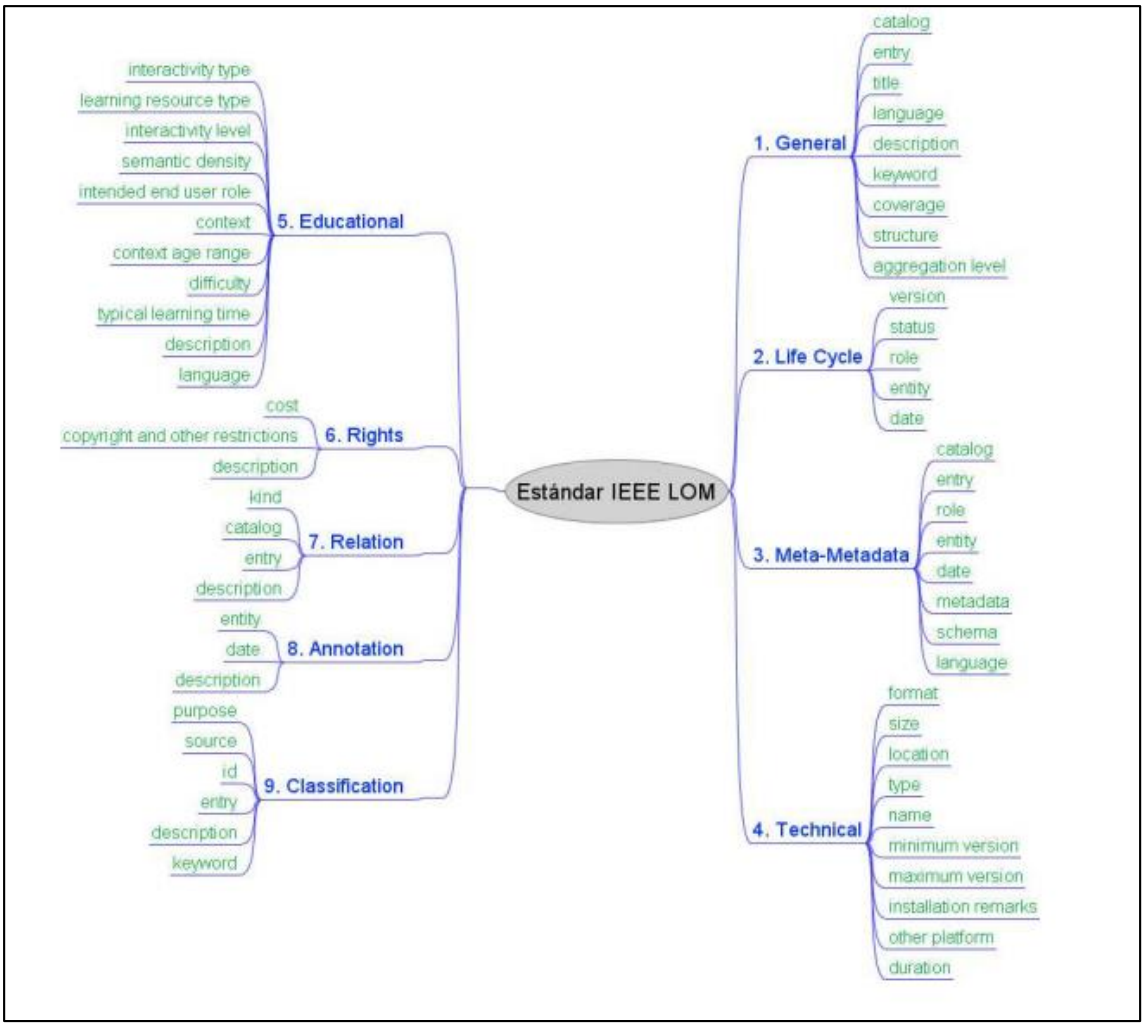


Ilustración 3: Estructura de LOM.

La estructura de LOM está compuesta por 9 categorías:

- La primera de ella es “General” donde se especifica los atributos frecuentes, como un identificador, lenguaje, descripción entre otros.
- La segunda categoría es “Life Cycle” donde se describe tanto la versión que es, por quien fue contribuido. Etc.
- Siguiendo con las categorías se encuentra con “Meta-Metadata”, donde se toma en cuenta los metadatos del recurso, aquí se tiene un identificador, la contribución, etc.

- En la categoría “Technical” se tienen las características y requisitos técnicos, como el formato en que se encuentra, requerimientos, etc.
- Con relación a “Educational” tenemos los usos educativos del mismo, donde podemos encontrar atributos como son el contexto, la densidad semántica, entre otros.
- En donde se agrupan los derechos de propiedad e intelectuales del recurso es en la categoría “Rights”.
- En la categoría “Relation” se ocupa para mostrar la relación que tiene con otros recursos, por lo tanto aquí se tiene el identificador del recurso con el cual se relación, además del tipo.
- En la penúltima categoría “Annotations” se representa los comentarios y anotaciones sobre el recurso, donde se tiene el comentario mismo, además de la fecha, etc.
- La última categoría “Classification” tiene propósito clasificar el recurso, debido a que en un repositorio de agrupa por medio de taxonomías y es aquí donde se especifica estas entidades.

4 Propuesta de arquitectura

Partiendo de la base que un repositorio es una colección de recursos (objetos o unidades de aprendizaje) que son accesibles a través de una red de comunicaciones [IMS, 2003], se debe asociar la creación de un repositorio local a la creación de un sistema, que permita consultar y almacenar la información correspondiente al contenido de los objetos de aprendizaje.

La arquitectura de un sistema es la estructura de este, que comprende sus componentes, las propiedades de estos componentes y las relaciones entre ellos [Bass et all, 1997]. Según esta definición, es que la creación de un repositorio local está directamente relacionada a definir la arquitectura necesaria para establecer la funcionalidad correspondiente del repositorio.

Existen 4 tipos de estilos arquitectónicos que permiten definir un sistema de repositorio local (y cualquier otro tipo de sistema): arquitectura orientada a servicios, arquitectura orientada a objetos, arquitectura basada en componentes y arquitectura basada en recursos. Todos estos estilos arquitectónicos tienen un factor común: utilizan la arquitectura en capas.

4.1 Arquitectura en capas

Lo que se conoce como arquitectura en capas es en realidad un estilo de programación donde el objetivo principal es separar los diferentes aspectos del desarrollo, tales como las cuestiones de presentación, lógica de negocio, mecanismos de almacenamiento, etc. Un ejemplo básico de esto consiste en separar la capa de datos de la capa de presentación al usuario.

La ventaja principal de este estilo es que el desarrollo se puede llevar a cabo en varios niveles y, en caso de que sobrevenga algún cambio, sólo se ataca al nivel requerido sin tener que revisar entre código mezclado. Un buen ejemplo de este método de programación sería el modelo de interconexión de sistemas abiertos.

El diseño más utilizado actualmente es el diseño en tres niveles (o en tres capas). A continuación, se deja un detalle de cada una de las capas existentes en una arquitectura:

- Capa de presentación: es la que ve el usuario (también se la denomina "capa de usuario"), presenta el sistema al usuario, le comunica la información y captura la información del usuario en un mínimo de proceso (realiza un filtrado previo para comprobar que no hay errores de formato). También es conocida como interfaz gráfica y debe tener la característica de ser "amigable" (entendible y fácil de usar) para el usuario. Esta capa se comunica únicamente con la capa de negocio.
- Capa de negocio: es donde residen los programas que se ejecutan, se reciben las peticiones del usuario y se envían las respuestas tras el proceso. Se denomina capa de negocio (e incluso de lógica del negocio) porque es aquí donde se establecen todas las reglas que deben cumplirse. Esta capa se comunica con la capa de presentación, para recibir las solicitudes y presentar los resultados, y con la capa de datos, para solicitar al gestor de base de datos almacenar o recuperar datos de él. También se consideran aquí los programas de aplicación.
- Capa de datos: es donde residen los datos y es la encargada de acceder a los mismos. Está formada por uno o más gestores de bases de datos que realizan todo el almacenamiento de datos, reciben solicitudes de almacenamiento o recuperación de información desde la capa de negocio.

Estas capas se aprecian claramente en la siguiente ilustración:

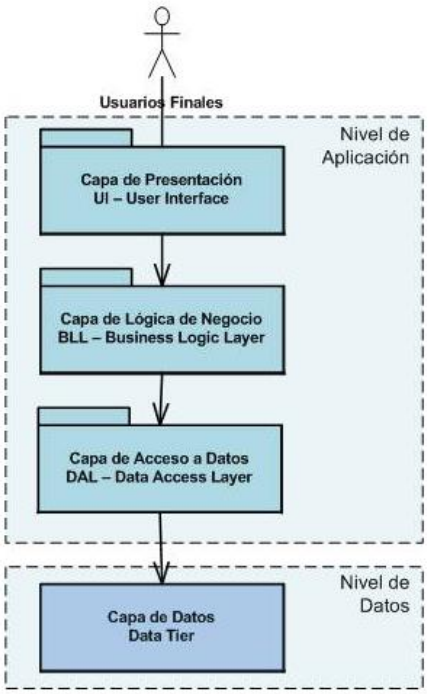


Ilustración 4: Arquitectura de capas de un sistema

Dentro de los objetivos del proyecto, se establece la creación de un repositorio local donde la principal orientación será proveer de objetos de aprendizaje y sus correspondientes comentarios para facilitar la realización de procesos de análisis de información contenida en recursos educativos. Ya que se desconoce la forma en que se van a llevar a cabo estos procesos, la forma en que se van a utilizar estos recursos y la accesibilidad que se va a brindar a la información, es que la propuesta de arquitectura se enfoca en establecer los componentes necesarios para dejar operativa la capa de datos, que es donde se almacenará la información correspondiente a los metadatos y comentarios de los objetos de aprendizaje. Los demás servicios correspondientes a las capas superiores deberán ser descritos en trabajos futuros que deseen trabajar con el repositorio local de datos.

4.2 Propuesta de arquitectura para el repositorio local

La propuesta de arquitectura contempla establecer el tratamiento de la información existente a nivel de la capa de datos. A continuación, se deja una ilustración con la arquitectura propuesta, además de la explicación de cada uno de los elementos:

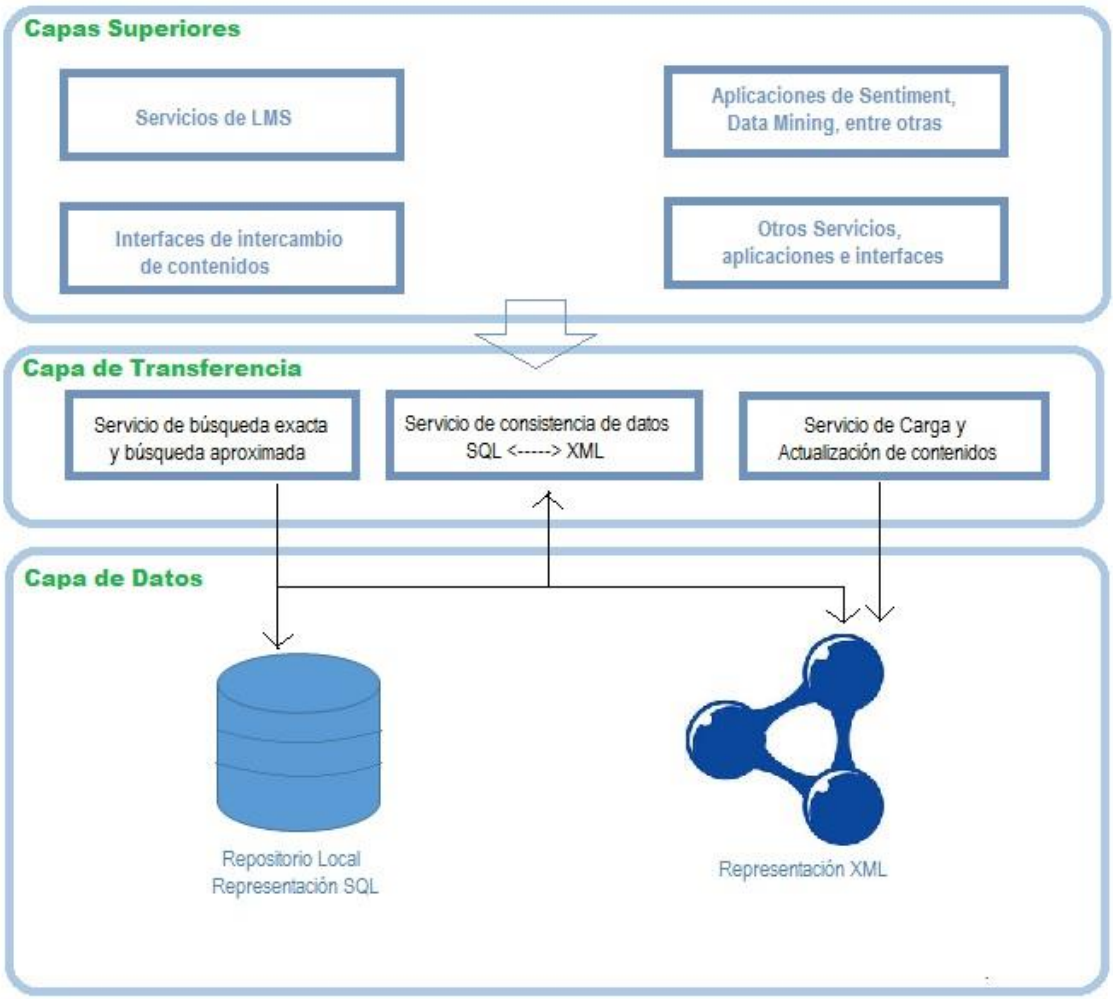


Ilustración 5: Arquitectura propuesta.

Esta propuesta está enfocada en la capa de datos, la que será implementada en su totalidad. Además, se propone una capa de transferencia, que busca mantener consistencia entre la representación híbrida de datos (propuesta en el capítulo 3.2). Con respecto a las capas superiores, solo se implementará un servicio de petición de recursos a Khan Academy. Otros servicios podrían ser implementados en trabajos futuros.

Capa de datos: De acuerdo a la selección de alternativas de modelado realizada en el capítulo 3.2, se propone que en la capa de datos exista una representación híbrida de almacenamiento donde coexiste un repositorio local con representación relacional y una representación de los datos del repositorio en XML. Esto permite contar con las ventajas que entregan ambas representaciones y así optimizar las operaciones que puedan realizar las capas superiores utilizando la capa de datos, tanto en tiempo de

respuesta como en consistencia de datos. Esto es posible pues en una representación relacional se utiliza SQL, lenguaje estándar para bases de datos, por ende universal y con amplio conocimiento por parte de usuarios, que además tiene como ventaja la efectividad al momento de realizar búsquedas aproximadas. Por otro lado, tener la representación XML hace eficientes las búsquedas exactas y, pensando que el repositorio local cumplirá una función educativa, es posible que muchas de las aplicaciones y servicios de capas superiores interactúen con la representación XML, siendo útil cuando varias aplicaciones se deben comunicar entre sí o deban integrar información de manera rápida y consistente. Cabe destacar que mantener una representación en XML hace más fácil la opción que servicios y aplicaciones de conversión de archivos XML puedan representar los datos en otros formatos como RDF/OWL, dada su universalidad.

Tanto la representación relacional como la representación en XML, están estructuradas bajo el estándar de LOM, según la elección realizada en el capítulo 3.3. LOM contiene un número de categorías con las cuales se representa de manera eficiente a un recurso de aprendizaje, y además, como es el estándar más utilizado para representar objetos de aprendizaje, permite que la representación del repositorio sirva ampliamente a cualquier usuario.

A continuación se presenta el modelo del repositorio bajo el estándar de LOM, tanto para la representación relacional como para la representación en XML:

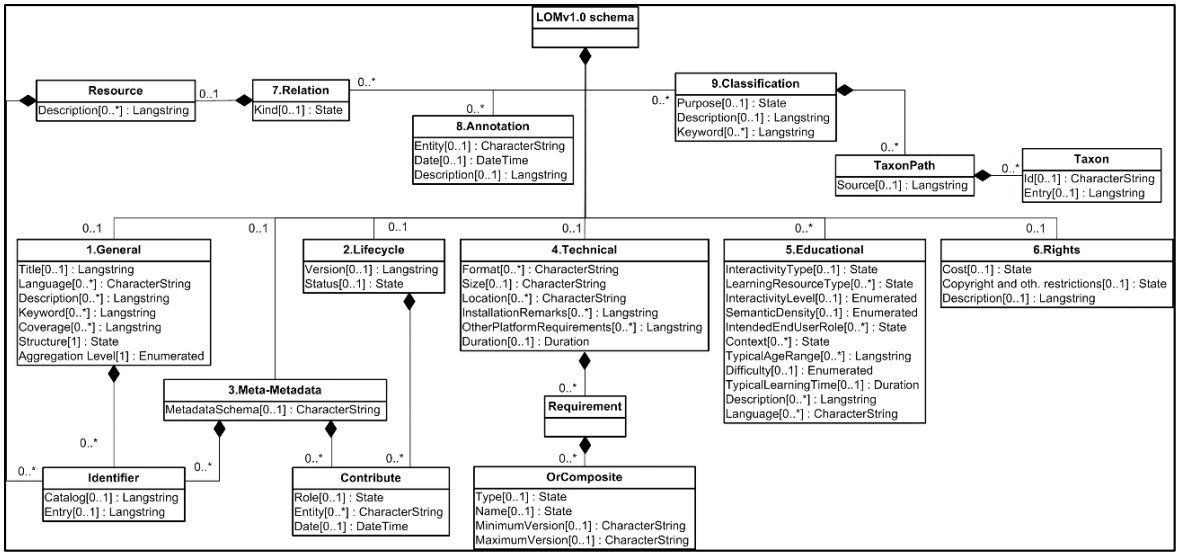


Ilustración 6: Representación relacional bajo el estándar LOM [EleGall, 2011]

En la ilustración anterior es posible ver la representación relacional bajo el estándar LOM que debe tener el repositorio local. Las características del estándar están expuestas en el apartado 3.3.1.

Cada una de las tablas, corresponde a las distintas categorías existentes en el estándar LOM y como se relacionan con la categoría principal (Recurso), donde los atributos de cada una de ellas almacenan la información relacionada al recurso. Además, el esquema muestra las cardinalidades que pueden existir entre las diferentes categorías.

Junto con la representación relacional, coexiste la representación XML, la cual también debe estar estructurada con el estándar LOM. En ella, es posible identificar por intermedio de etiquetas cada uno de los recursos correspondientes. Cada sub-etiqueta representa un mayor nivel de detalle en la información que representa al objeto de aprendizaje.

Ilustración 7: Representación XML bajo el estándar LOM

La ventaja de que ambas representaciones estén bajo un mismo estándar es que es mucho más fácil mantener la consistencia de los datos tras realizar operaciones de carga y actualización, puesto que en ambas representaciones existirán las mismas categorías, que en la representación relacional serían tablas y los atributos de cada categoría se identifican de la misma forma en ambas representaciones.

Con respecto al modelo físico que contempla la arquitectura, queda a disposición del usuario final su distribución (Discos duros, particiones, tamaños, etc), ya que esta va a variar dependiendo de la cantidad de datos y el uso que vaya a tener el repositorio, además de la interoperabilidad de los servicios y aplicaciones que trabajen con la capa de datos, por ejemplo si se establece el repositorio como un servidor local o un servidor web.

Capa de transferencia: Dado que en la capa de datos se establecen dos representaciones de los datos en distintos formatos, se hace necesario contar con una capa superior que le brinde soporte, por ende existe una interoperabilidad entre ambas capas. Para ello, se proponen servicios que se detallan a continuación:

- Servicio de búsqueda exacta y búsqueda aproximada: Este servicio provee el acceso a la información contenida tanto en el repositorio local relacional como a la representación XML.

  Generalmente, se desea acceder al repositorio para realizar búsquedas aproximadas, por lo que se hace necesario consultar el repositorio relacional. Esta búsqueda aproximada basada en palabras clave se realiza como una búsqueda libre tanto en el contenido textual como en los metadatos, implementada usando motores de recuperación de información tradicionales.

  Si se desea acceder al repositorio para realizar búsquedas exactas, es posible acceder directamente a la representación XML, utilizando motores de búsqueda XML. Además, es posible realizar las búsquedas en la representación XML utilizando expresiones en lenguaje SQL (más familiar a un usuario dado que es estándar y universal) por medio de una herramienta propuesta por Cartujano et al., (2002) llamada SQL2XQuery que retorna un documento XML.
- Servicio de carga y actualización de contenidos: Para poder insertar nuevos datos al repositorio o actualizar los existentes, es necesario contar con este servicio. Lo que hace básicamente es modificar o insertar datos solamente en la representación XML, pues como ambas representaciones están realizadas bajo el

alero del estándar de LOM, se apoya del servicio de consistencia de datos para simplemente ubicar la categoría y el atributo correspondiente a cargar o actualizar.

- Servicio de consistencia de datos SQL/XML: Dada la existencia de 2 formas de representación de los mismos datos, se propone la existencia de un servicio que maneje la consistencia de los datos al momento de realizar acciones de carga y actualización de datos. Incluso, puede realizar comparaciones de datos sin la necesidad de ejecutarse al momento de la carga o actualización, simplemente para ver que los datos se mantengan correctamente en ambas representaciones. Lo ideal es que se mantenga en funcionamiento al momento de ejecutarse el servicio mencionado en el ítem anterior, dado que a la vez puede ejecutarse el servicio de búsqueda exacta y búsqueda aproximada y requerir datos actualizados.

Capas superiores: Sobre la capa de datos y la capa de transferencia, existen todos los demás servicios y aplicaciones que de una u otra manera utilizan los servicios expuestos de la capa de datos para realizar las acciones mencionadas. Estas capas pretenden dar funcionalidad sobre los datos a los usuarios finales. Dentro de estos servicios o aplicaciones que puedan implementarse existen los servicios de LMS, aplicaciones de Análisis de Sentimientos y Minería de Datos, entre otras. Otros servicios podrían ser implementados en trabajos futuros.

Cabe destacar que para validar la arquitectura propuesta, en este proyecto se implementa una aplicación que se conecta al repositorio Khan Academy para extraer los metadatos y comentarios de los objetos de aprendizaje existentes y los entrega a la capa de datos para almacenarlos según la estructura propuesta. Esta aplicación será expuesta en el siguiente capítulo.

5 Recuperación de recursos desde el repositorio Khan Academy

Considerando la arquitectura propuesta en el capítulo anterior, se requiere crear un repositorio local el cual pueda almacenar datos reales de objetos de aprendizaje en ambas representaciones propuestas. Para ello, se considera el desarrollo de una aplicación que pueda recolectar información de los recursos de aprendizaje disponibles en el repositorio Khan Academy, y así poder almacenarlos en el repositorio local según la estructura mencionada en la propuesta.

En esta sección se describe el acceso ofrecido por los desarrolladores de Khan Academy para utilizar la API del repositorio y así poder recuperar metadatos de sus recursos. Además, se presentan los antecedentes de un Recolector (en inglés: Gather) que brindaba una posible solución a una parte de los objetivos del proyecto, el cual fue necesario adaptar y mejorar para que cumpliese con los objetivos propuestos en este proyecto. Junto con ello, se describe la tecnología usada en la construcción de una aplicación que permitirá recuperar la información, los metadatos y los comentarios de cada recurso educativo, para luego almacenarlos en el repositorio local bajo la representación relacional y la representación XML, ejemplificando con un caso de estudio.

5.1 Aplicación: Recolector de Objetos de Aprendizaje

La recuperación de comentarios a realizar en este proyecto consiste en poder conectarse al repositorio Khan Academy mediante un Recolector que localiza el recurso y sus respectivos comentarios. Este recolector tiene la funcionalidad de poder navegar automáticamente por todos los recursos existentes en el repositorio, permitiendo obtener datos relevantes de estos como lo son los metadatos y los comentarios.

Para cumplir con este objetivo, se ha propuesto una aplicación en la cual sea posible conectarse al repositorio Khan Academy por intermedio de la API. Los metadatos y comentarios recuperados, deben ser almacenados en una base de datos temporal para poder ser clasificados de manera correcta, y así, se realice la carga de estos datos de manera efectiva al repositorio local. Recordemos que el repositorio local, en su capa de datos, estará compuesto por dos representaciones (representación relacional y representación XML), ambas bajo el estándar de etiquetado LOM.

El procedimiento mencionado, que será explicado en el transcurso de este capítulo, está ejemplificado en la siguiente ilustración:

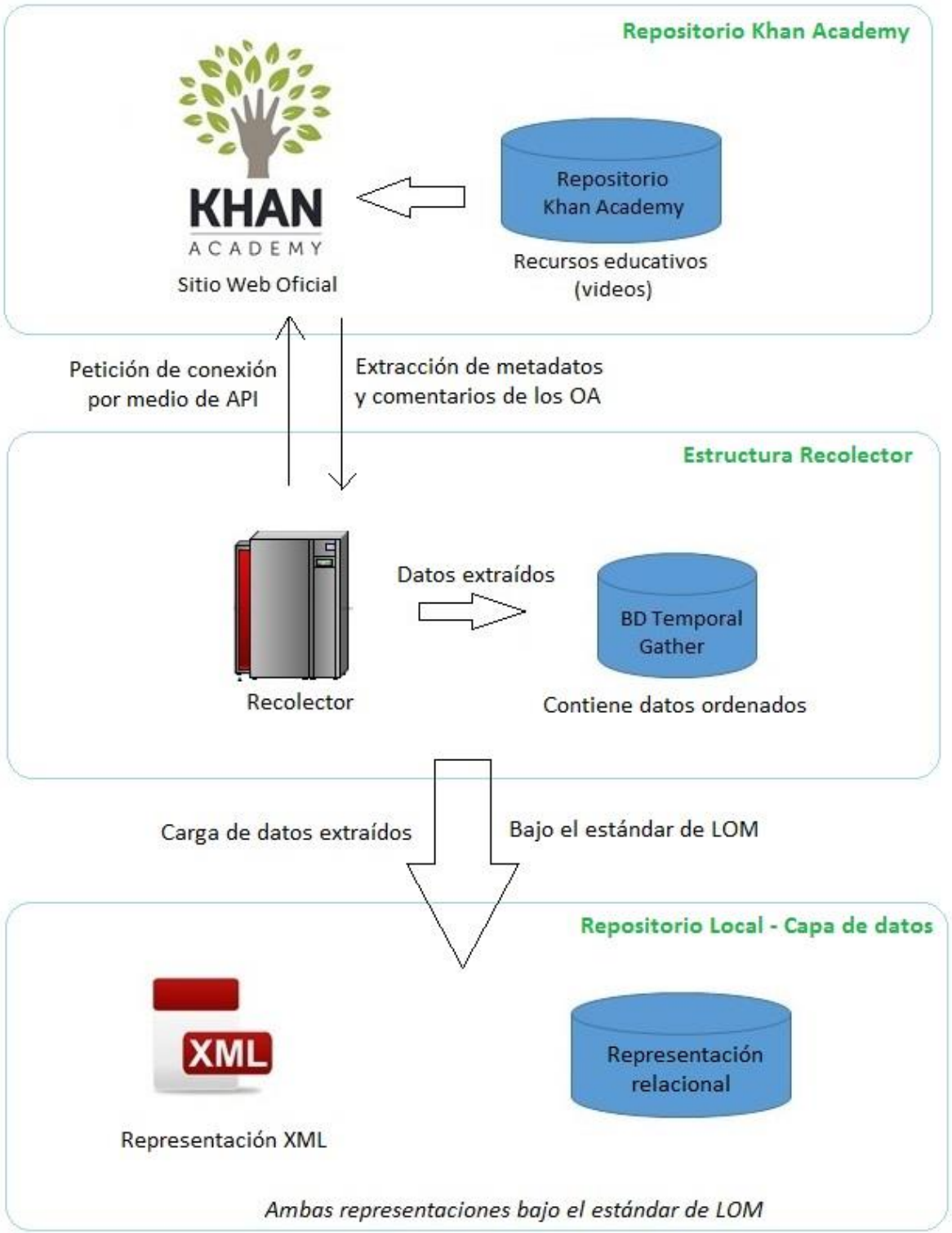


Ilustración 8: Procedimiento extracción de datos

5.2 Perfil de usuario

En este apartado se describe de forma breve el tipo de usuario al cual está dirigido el uso de este recolector.

El recolector es un sistema de recuperación de información desde el repositorio Khan Academy, el cual permite la recuperación de videos y comentarios, generando una representación local. Esta representación del repositorio local está enfocada en la investigación, orientada en la realización de futuros procesos de análisis de información (ver apartado 6.2).

El uso de este recolector está enfocado en investigadores que requieran realizar procesos sobre los metadatos y comentarios de los objetos de aprendizaje, en particular del repositorio del cual se extraen esos recursos. Además, gracias a lo intuitiva de la interfaz gráfica de trabajo, es posible que un usuario con conocimientos básicos, pueda realizar la recuperación de datos de manera sencilla, desconociendo la utilización que se le pueda dar a los datos.

5.3 Antecedentes: Recolector Original

A partir de los trabajos en desarrollo realizados por el profesor Miguel Ángel Sicilia y el grupo de investigación IERU de la Universidad de Alcalá, España (www.ieru.org), se hizo la revisión a un recolector de su autoría (de ahora en adelante, Recolector Original), desarrollado en lenguaje de programación Python a fines del año 2012; el cual podía recorrer los recursos existentes en el repositorio Khan Academy y almacenar algunos datos relevantes. Cabe destacar que el objetivo del Recolector Original está enfocado en la realización de Análisis de Sentimiento, con lo que existen algunas funcionalidades que están orientadas a cumplir con otras expectativas no vistas en este proyecto.

El código fuente correspondiente al Recolector Original está disponible para su revisión en el anexo 1.

En términos simples, lo que buscaba realizar el recolector original era lo siguiente:

1. Crear una base de datos donde se almacenará la información asociada a los objetos de aprendizaje. Los atributos de las tablas existentes corresponden a los datos que entrega la API de Khan Academy.
2. Conectarse al repositorio Khan Academy, por intermedio de la API del sitio.

3. Según la estructura del repositorio, se almacenan en la base de datos, primero datos referentes a playlist que contienen videos, y luego, se almacenan los datos referentes a videos asociados a las correspondientes playlist.
4. Finalmente, se recorre cada una de las URL correspondientes a cada uno de los videos recuperados en el paso anterior para rescatar los comentarios asociados a ellos.
5. La información y comentarios de cada uno de los videos deben ser escritos en un archivo con formato compatible con aplicaciones de realización de Análisis de Sentimientos.

El proceso descrito anteriormente, podría llegar a cumplir con parte de los objetivos propuestos en este proyecto, pero se aleja de la creación de un repositorio local bajo un estándar de almacenamiento de datos de objetos de aprendizaje. Además, este recolector original no está ajeno a problemas para poder ser ejecutado (como podrá revisarse a fondo en el apartado 5.3), por lo que se hace necesario adaptar el proceso para cumplir con los objetivos planteados y además, solventar los problemas existentes. Para ello, es necesario entender el funcionamiento del repositorio y luego analizar a fondo el recolector original.

5.3.1 Estructura de los recursos dentro del repositorio

Antes de explicar el código utilizado en el Recolector original, es necesario explicar la forma de almacenamiento de recursos utilizada en el repositorio Khan Academy. Es necesario saber que el repositorio Khan Academy retorna la información de los recursos en formato JSON. JSON (acrónimo de *JavaScript Object Notation*) es un formato ligero para el intercambio de datos, que generalmente es utilizado en sitios web, dado que la gran mayoría emplea JavaScript. La ventaja de JSON frente a otros formatos de intercambio de datos es que mantiene el intercambio cuando el tamaño del flujo de datos entre el cliente y el servidor es de vital importancia y la fuente de datos es explícitamente de fiar, como lo es un repositorio.

Khan Academy, como muchos repositorios, mantiene una estructura definida para poder conectarse a sus recursos. Esta información está disponible en la API de Khan Academy (de libre acceso: http://api-explorer.khanacademy.org/). Se hace necesario poder acceder por medio de esta API ya que lo que se requiere es automatizar la búsqueda de recursos, y además, tener acceso garantizado, dada la constante

actualización que se realiza en el repositorio Khan Academy, tanto de carga de nuevo contenido como de actualización y eliminación.

Cabe destacar que los trabajos realizados con el Recolector Original fueron desarrollados a fines del año 2012, en donde la API de Khan Academy mantenía una estructura distinta a la actual. Para comprender el funcionamiento del Recolector Original, a continuación se detalla la forma en la cual se encontraba estructurada la conexión a los recursos en ese entonces. (En el punto 5.3.2 se muestran los cambios realizados a la API de Khan Academy que permitirán entender las modificaciones realizadas para el correcto funcionamiento del recolector propuesto en el proyecto).

La API establece que para poder identificar un recurso, este tiene un ID (identificador) y una URL asociada con el cual poder acceder al sitio web que aloja al recurso. Este identificador, por temas de seguridad, no se encuentra disponible directamente. En ese entonces, el acceso a los videos se realizaba identificando el tópico al cual pertenecen los videos (por ejemplo: álgebra, cálculo, física mecánica, etc.). Según la Api de ese entonces, la lista de tópicos podía ser identificada utilizando la siguiente URI:

/api/v1/topic

Cada tópico debiese contener muchas playlist de videos relacionados a ese tópico, por ejemplo, el tópico álgebra puede contener playlist de álgebra vectorial, álgebra matricial, álgebra y espacios vectoriales, etc. Con la siguiente URI es posible acceder a los videos pertenecientes a las distintas temáticas de un tópico:

/api/v1/topic/topic_slug/videos

Gracias a este acceso, es posible obtener un texto temporal de formato JSON con los datos de todos los videos pertenecientes a esa temática. De esta forma, se almacena el ID, la URL relativa y otros datos relevantes de todos los videos del repositorio y así poder hacer posible la carga automática de las direcciones en el navegador. Luego, para poder acceder al recurso, se carga en el navegador el antecedente http://www.khanacademy.org/, se copia la URL relativa del recurso y se pega justo después de este antecedente. Esto se realizaría de la siguiente forma:

http://www.khanacademy.org/URL_relativa

Tras cargar el recurso de la forma mencionada, Khan Academy nos redirige (agregando algunas variables en la barra de direcciones) a la página donde se encuentra almacenado el recurso. Este re direccionamiento ocurre dado que Khan no almacena directamente sus recursos, pues son videos almacenados en YouTube, por lo que se hace necesario mostrar la página completa del recurso en Khan Academy. Todo esto ocurre a nivel de API, por lo que la dirección final que se obtiene es distinta a la que se muestra a un usuario común ingresando normalmente a la página.

5.3.2 Almacenamiento de los datos

Dada la estructura que entrega la API para la conexión al repositorio, se hace necesario almacenar los datos correspondientes para establecer futuras conexiones, evitar duplicar datos y además, guardar los datos esenciales de los recursos.

En el Recolector Original, se estableció guardar los datos correspondientes a los objetos de aprendizaje recuperados en una base de datos, donde las tablas utilizadas para su representación quedaban expresadas de la siguiente forma:

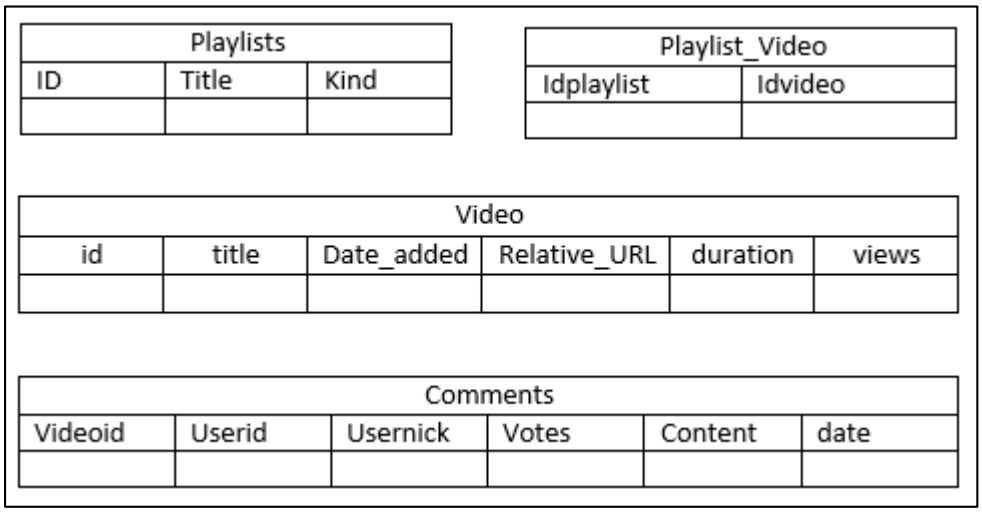


Ilustración 9: Representación tablas, Recolector original.

- La tabla playlist almacena los datos de la playlist siendo el ID y el título proporcionados por el repositorio, además incluyendo el topic en el atributo kind.
- La tabla Video almacena los datos del recurso, en este caso un video. Contiene un ID correspondiente la url_relativa sin el antecedente “video/ “, que distingue indiscutidamente a un video de otro. Además contiene el título correspondiente,

fecha de subida al repositorio, URL_relativa, duración del video y cantidad de visitas.

- La tabla playlist_video contiene los emparejamientos entre los id de la playlist y su correspondientes videos, asociados por el idvideo.
- La tabla comments contiene el id del video al cual pertenece el comentario, además el id y Nick del usuario que realizó el comentario, la cantidad de votos positivos, el contenido del comentario y la fecha en el cual fue realizado.

Cabe destacar que los atributos correspondientes a la tabla playlist, la tabla Video y la tabla playlist_video, almacenan los metadatos correspondientes a los recursos encontrados. Estos son los únicos datos que son posibles acceder por medio de la API, así que para acceder a los comentarios, es necesario conectarse a la URL correspondiente al recurso y extraerlos por medio de texto plano.

5.3.3 Funciones del Recolector Original

En este apartado se busca explicar detalladamente las funciones contenidas en el Recolector original, explicando su funcionalidad. A continuación, se presenta el análisis de cada una de ellas:

5.3.3.1 Llamado a librerías

Al igual que en todo código fuente de cualquier lenguaje de proramación, al principio se incluyen las librerías a utilizar. En este caso particular, las librerías necesarias son:

- Httplib: Librería que permite trabajar en python con un cliente http.
- Json: Librería que permite trabajar con textos de formato JSON.
- Sqlite3: Librería que permite trabajar con SQLite y su respectivo tratamiento de bases de datos.
- Selenium: paquete de librería que monta un entorno de pruebas de software para aplicaciones basadas en la web.

Luego se establece como variable global KHANACAD_URI: www.khanacademy.org.

5.3.3.2 Storetopics

La función *Storetopics* tiene por objetivo conectarse al repositorio Khan Academy por intermedio de una compuerta HTTP y almacenar los tópicos a los que se asociaban los videos (Según la API al momento de la creación del recolector original). Para ello, recibe como parámetro la base de datos a utilizar, luego establece la conexión y hace la solicitud al sitio de la raíz del tópico para después mostrarla por pantalla.

```
def storetopics(db):

    uri = "/api/v1/topic"
    # Get the root topic:
    conn = httplib.HTTPConnection(KHANACAD_URI)
    conn.request("GET", uri+"/root")
    res = conn.getresponse()
    data = res.read()
    print data
    return
```

Ilustración 10: Modulo storetopics (Recolector original).

5.3.3.3 Storevideos

La función *Storevideos* tiene por objetivo conectarse al repositorio Khan Academy por intermedio de una compuerta HTTP a la URI de las playlist y obtener un archivo JSON con los datos todas ellas. Para ello, recibe como parámetro la base de datos a utilizar, procede a conectarse al repositorio para obtener el archivo JSON, decodificándolo y almacenando sus datos por medio de la función *__storeplaylistinfo*. Una vez obtenidos estos datos, carga por intermedio de una compuerta HTTP la URI de los videos. Llama a la función *__Storevideoinfo*.

```
def storevideos(db):
    """
     Saves Khan Academy playlist&video info in the database.
     Playlists group videos.
    """
    uri = "/api/v1/playlists"
    conn = httplib.HTTPConnection(KHANACAD_URI)
    conn.request("GET", uri)
    res = conn.getresponse()
    data = res.read()
    decoded = json.loads(data)
    for playlist in decoded:
        print playlist['title']
        __storeplaylistinfo(db, playlist)

        # Get the videos of each playlist
        conn2 = httplib.HTTPConnection(KHANACAD_URI)
        uri2 = "/api/v1/playlists/"+playlist['id']+'/videos'
        conn2.request("GET", uri2)
        res2 = conn2.getresponse()
        data2 = res2.read()
        decoded2 = json.loads(data2)

        for video in decoded2:
             __storevideoinfo(db, video, playlist)

    conn.close
```

Ilustración 11: Modulo storevideos (Recolector original).

5.3.3.4 __Storevideoinfo

La función *__Storevideoinfo* tiene por objetivo almacenar en la base de datos la información de los videos y la playlist a la cual está asociado. Para ello, recibe como parámetro la base de datos a utilizar, el video (datos decodificados del archivo JSON recolectado en la función *Storevideos*) y la playlist.

```
def __storevideoinfo(db, video, playlist):
    """
    Stores info on the video in the DB, including the playlist in which it is included.
    """
    title = video["title"]
    title = title.replace("'", " ")
    id = video["readable_id"]
    id = id.replace("'", " ")
    date_added = video["date_added"]
    date_added = date_added.replace("'", " ")
    duration = video["duration"]
    views = video["views"]
    relative_url = video["relative_url"]

    db.execute("INSERT INTO video VALUES ('"+ id +"','"+title+"','"+date_added+"','"+
               relative_url + "','" + str(duration)+"','" + str(views)+"')")
    db.execute("INSERT INTO playlist_video VALUES ('"+ playlist["id"] +"','"+id+"'"+")")
    db.commit()
    return
```

Ilustración 12: Modulo__ storevideos (Recolector original).

5.3.3.5 __Storeplaylistinfo

La función __Storeplaylist info tiene por objetivo almacenar la información relacionada a la playlist. Una playlist contiene una lista de videos asociada, lo que hace que tengan características distintas entre ellas. Para funcionar, recibe como parámetro la base de datos a utilizar y la playlist (con datos decodificados del archivo JSON en la función *Storevideos*).

```
def __storeplaylistinfo(db, playlist):
    """
    Stores the information of a playlist in the db
    """
    title = playlist["title"]
    title = title.replace("'", " ")
    id = playlist["id"]
    id = id.replace("'", " ")
    kind = playlist["kind"]
    kind = kind.replace("'", " ")


    db.execute("INSERT INTO playlist VALUES ('"+ id +"','"+title+"','"+kind+"')")
    db.commit()
    return
```

Ilustración 13: Modulo __storeplaylistinfo (Recolector original).

5.3.3.6 Createdb

La función Createdb tiene por objetivo crear las tablas de la base de datos (en sqlite3), con sus respectivos atributos, para ser utilizada en las demás funciones. Si la

base de datos ya se encuentra creada, simplemente omite la creación. Recibe como parámetro el nombre que se quiere asociar a la nueva base de datos.

```
def createdb(dbname):
    """
    Creates an in-memory sqlite3 database with a schema for Cosm feeds.
    """
    # For in-memory database:
    #     conn = sqlite3.connect(':memory:')
    conn = sqlite3.connect(dbname)

    conn.execute('''CREATE TABLE playlist
                    (id text, title text, kind text)''')
    conn.execute(("CREATE TABLE playlist_video (idplaylist text, idvideo text)"))
    conn.execute('''CREATE TABLE video
                    (id text, title text,  date_added text, relative_url text,
                    duration text, views text)''')
    conn.execute('''CREATE TABLE comments
                    (videoid text, userid text, usernick text, votes integer,
                    content text, date text)''')
    return conn
```

Ilustración 14: Modulo createdb (Recolector original).

5.3.3.7 Opendb

La función *Opendb* tiene por objetivo realizar la conexión a la base de datos para dejarla utilizable por las demás funciones.

```
def opendb(dbname):
    conn = sqlite3.connect(dbname)
    return conn
```

Ilustración 15: Modulo openbd (Recolector original).

5.3.3.8 Scrap_videos

La función Scrap_videos tiene por objetivo inicializar las aplicaciones necesarias para realizar la extracción de comentarios desde Khan Academy. Para ello, recibe como parámetro el nombre de la base de datos donde se almacenaran los datos. Luego, abre una sesión del navegador Mozilla Firefox para así poder recorrer los sitios donde se encuentran almacenados los recursos y sus correspondientes comentarios. Pasa por parámetro URI la dirección de todos los videos para poder recorrer todos los videos y capturar los comentarios por medio de la función *__Scrap_video (apartado 5.2.3.9)*.

```
def scrap_videos(db):
    """

    """
    res = db.execute("SELECT id, relative_url FROM video")
    browser = webdriver.Firefox() # Get local session of firefox
    for row in res:
        ___scrap_video(db, "http://"+KHANACAD_URI+row[1], browser)

        break
    return
    browser.close()
```

Ilustración 16: Modulo scrap_videos (Recolector original).

5.3.3.9 __Scrap_video

Esta función no se encuentra implementada. Lo que pretendía hacer es: al cargar la página del video, hacer clic en el botón "tips & comments" y almacenar los comentarios. Solo hace el clic e imprime el largo del texto del botón.

```
def ___scrap_video(db, uri, browser):
    """
    NOT WORKING: See Java implementation.
    """
    print uri +"\n"
    browser.get(uri) # Load page
    # Move to "Tips & comments"
    tipscommentslink = browser.find_element_by_xpath("//a[contains(text(),'Tips & Comments')]")
    tipscommentslink.click()

    # Find the two list: first will be questions, second will be tips & comments
    tabs = browser.find_elements_by_xpath("//div[@class='discussion-list-content']")
    #contents = comments.find_elements_by_xpath("/div[@class='comment  discussion-item']")
    print len(tabs)

    return
```

Ilustración 17: Modulo __scrap_video (Recolector original).

Nota: La página de Khan Academy maneja en la actualidad 2 botones: "Questions" y "Tips & Thanks". Lo más probable es que en el momento en que se desarrolló el Recolector Original, el repositorio solo contenía el apartado "Tips & Comments".

5.3.3.10 Dump_playlists_per_video y dump_comments_users_to_pajek

Ambas funciones reciben como parámetro la base de datos y el nombre que se quiere poner al archivo de salida. Estas funciones realizan el acomodo de los datos para crear un archivo TXT que pueda ser leído por el software Pajek. Ambas funciones no

tienen que ver con el propósito de la investigación, por lo que no se consideran en este apartado.

Nota: Pajek es un software desarrollado por profesores de la Universidad de Ljubljana (Slovenia) que realiza análisis de redes sociales. Este software utiliza como entrada archivos de texto con la información a analizar.

5.3.3.11 Dump_video_time_series y dump_video_contrib_frequency

Ambas funciones reciben como parámetro la base de datos y el nombre que se quiere poner al archivo de salida. La primera muestra la frecuencia de contribuciones de videos y la segunda muestra la cantidad de videos subidos al repositorio por día. Ambas funciones no tienen que ver con el propósito de la investigación, por lo que no se consideran en este apartado.

5.4 Problemática del Recolector original

Tras realizar un análisis al Recolector original y tratar de ejecutar las distintas funciones, pocas de ellas son las que se encuentran funcionando en la actualidad. Esto se debe a que el Recolector original estaba destinado a realizar otras funcionalidades, y además, ha sido afectado por el paso del tiempo. A pesar de esto, sí se puede ocupar como base para un nuevo Recolector, realizando adaptaciones que sirvan para cumplir con los objetivos propuestos en este proyecto.

Dentro de las funciones que operan sin problemas se encuentran:

- Crear base de datos
- Conexión a la base de datos

Si bien, estas funciones se ejecutan sin problemas, la creación de la base de datos ha de ser modificada debido la nueva estructura de almacenamiento de datos en el repositorio local.

Con respecto a las demás funciones que posee el recolector original, es posible utilizarlas de manera independiente, pero se hace necesario adaptar funciones existentes y crear algunas nuevas para así cumplir con los objetivos de la investigación. Esto, ya que

al ejecutar las funciones mencionadas anteriormente, solo se cuenta con una base de datos con tablas vacías de la cual no es posible rescatar nada.

La problemática de las demás funciones y de la ejecución del recolector original en sí se detalla a continuación.

5.4.1 Ambiente de trabajo:

Para poder ejecutar el Recolector, ya sea en un equipo con sistema operativo Windows o Linux, es necesario contar con una versión de Python en el equipo. Las librerías que se ejecutan dentro del Recolector pueden ser fáciles de conseguir dentro de un sistema operativo u otro, pero varían según la versión de Python a utilizar, como lo es Selenium, por lo que se hace necesario corroborar la versión de Python, que pueda contener todas las librerías necesarias, y que además, pueda ser ejecutada en el sistema operativo que se posea.

Cabe destacar que para poder realizar este proyecto, y después de realizar innumerables pruebas para hacer funcionar el recolector, la configuración utilizada fue la siguiente:

- Windows 7 Ultimate 64 bits.
- Python en su versión 2.7.
- Sqlite Database Browser. Versión 3.1.
- Equipo con hardware compatible con el software mencionado anteriormente.

5.4.2 Cambios en la API del repositorio:

Como se ha mencionado anteriormente, los trabajos realizados por el profesor Miguel Ángel Sicilia fueron realizados a fines del 2012, donde Khan Academy manejaba un tratamiento distinto de los recursos según la API. El acceso a los videos se realizaba por tópico, por lo que cada playlist manejaba ciertos tópicos en los cuales estaban almacenados los videos. La estructura era la siguiente:

/api/v1/topic

Khan Academy modificó este apartado en su API, dejando sin funcionamiento este acceso, y dejando el acceso al repositorio por medio de las playlist. Este nuevo acceso queda definido por medio de la siguiente estructura de la URI:

/api/v1/playlist

Las playlist son listas que contienen videos relacionadas a una temática específica, y es aquí donde recién obtenemos el identificador de los videos para poder acceder a ellos. La estructura de acceso al video sería la siguiente:

/api/v1/playlist/playlist_title/videos

Gracias a este nuevo acceso, es posible obtener un texto temporal de formato JSON con los datos de todos los videos pertenecientes a esa playlist. De esta forma, se puede almacenar el ID, la URL relativa y otros datos relevantes de todos los videos del repositorio y así poder hacer posible la carga automática de las direcciones en el navegador como se mencionó en el apartado de estructura del repositorio.

Si bien, en la API se deja establecido que el acceso mediante el tópico o la playlist cumplen la misma función, al quedar obsoleto el acceso mediante tópico, se hace necesario cambiar en el Recolector las URI de conexión.

5.4.3 Tratamiento de URL

El archivo JSON que posee los datos de los videos, contiene datos en formato texto. Dentro de estos datos, se utiliza la URL relativa para poder conectarse al sitio donde se encuentra el recurso.

Muchas de estas URL no se encuentran en un formato en el cual se pueda copiar directamente a la barra de direcciones del navegador, con lo que se hace necesario realizar un análisis para idear un tratamiento de casos especiales.

Algunos de los casos mencionados son:

- Ć : Introduction to l'Hôpital's rule
- ™: Hour of Code™

5.4.4 Inexistencia de una función que rescate comentarios

En el apartado 5.3.4 es posible apreciar que la función que pretende rescatar comentarios no se encuentra implementada. El Recolector original no contaba con una implementación adecuada para recuperar comentarios, debido a que este estaba desarrollado con otro objetivo, pero se contaba con una breve descripción de cómo se podía realizar, donde mencionaba que podría ser implementado a través de Java.

Además, revisando los datos de la API, no se cuenta con ninguna función que pueda recuperar comentarios directamente desde el repositorio, por lo cual fue necesario indagar al respecto, donde se encontró la solución para esta dificultad.

5.5 Mejoras realizadas al Recolector original

En este apartado se busca explicar las mejoras implementadas al Recolector original, además de las nuevas funcionalidades incorporadas.

5.5.1 Creación de estructuras: Bases de datos

Teniendo en cuenta la propuesta de aplicación de recuperación realizada en el apartado 4.1, lo primero que se debe realizar es establecer la forma en que se crearán las estructuras de almacenamiento de datos del repositorio local. Para ello, la función *Createdb* ha sido modificada para trabajar de la siguiente forma:

- Crear una base de datos donde se almacenará la representación relacional del repositorio. Para ello, se crean todas las tablas correspondientes al estándar LOM según el modelo presentado en el apartado 3.3.1.
- Dentro de la misma base de datos, crear unas tablas temporales que servirán para organizar los datos extraídos de cada video y su posterior traspaso a la representación relacional (acomodar en las tablas correspondientes) y a su representación XML. Este procedimiento se hace necesario dada la estructura que maneja el repositorio Khan Academy, que es distinta a la propuesta en el repositorio local.

Nota: El motivo de realizar esta creación de tablas temporales dentro de la misma base de datos utilizada para la representación relacional del repositorio, tiene que ver con que Python no permite realizar dos conexiones en una misma ejecución a dos bases de datos distintas. Dado que estas tablas temporales no pertenecen a la representación relacional,

tras haber llenado todos los campos correspondientes a los recursos recuperados, se procede a eliminar cada una de estas tablas temporales, dejando en el repositorio solo la base de datos que corresponde a la representación relacional del repositorio.

En relación a las tablas temporales, y considerando el almacenamiento de los datos del apartado 5.2.2, se propone una mejora a la forma en que se almacenan los datos en la base de datos temporal.

Es posible formalizar un modelo de datos que permite representar de una manera comprensible los elementos que intervienen en el sistema y sus relaciones. Para ello, se presenta el modelo entidad-relación, describiendo cada una de las entidades identificadas:

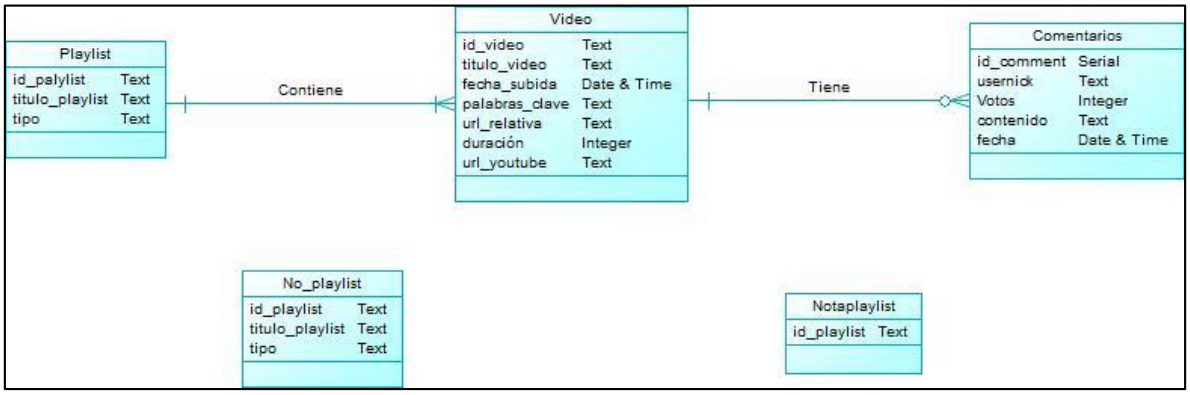


Ilustración 18: Modelo entidad-relación.

En él, es posible identificar tres entidades activas relacionadas entre sí: Playlist, Video y Comentarios, además de dos entidades pasivas: No_playlist y Notaplaylist, que sirven para poder identificar elementos que no participan directamente pero que pueden servir para tener algún registro.

Principalmente se aprecia que una playlist contiene uno o muchos videos y un video está asociado a una sola playlist. Además, un video puede no tener comentarios como tener muchos y un comentario pertenece solo a un video.

El diseño físico de la base de datos muestra la representación de la organización y estructura de la base de datos que será utilizada. Para ello se especifica con el siguiente Modelo Relacional, consistente con el modelo entidad-relación presentado anteriormente, en el cual cada tabla representa un almacén de datos de la base de datos con los atributos correspondientes:

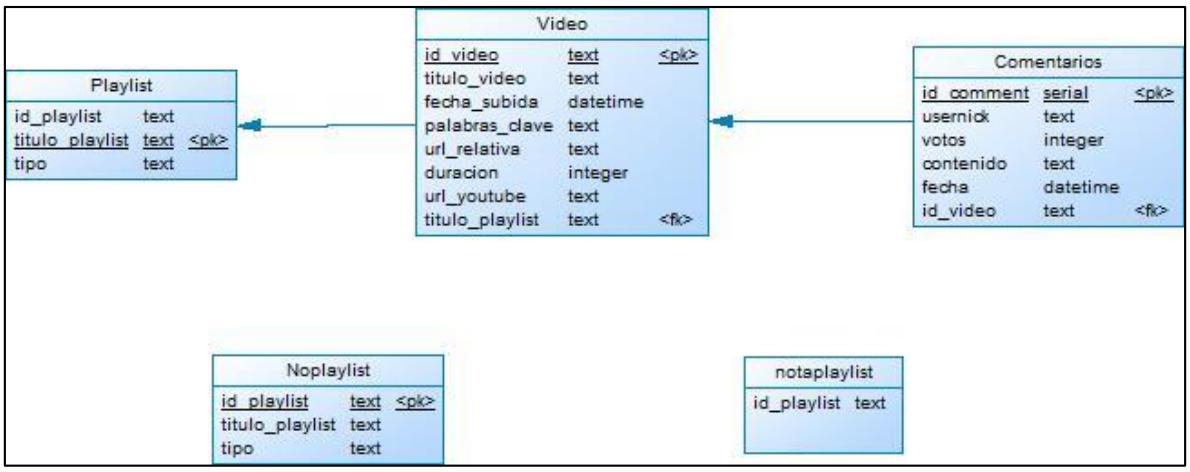


Ilustración 19: Modelo Relacional: tablas temporales

El modelo relacional está compuesto por cinco tablas, tres de las cuales se encuentran relacionadas según el modelo entidad-relación. Cada una de las tablas contiene los atributos necesarios para identificar correctamente las interacciones de los actores con el sistema. Estas se explican a continuación:

- Playlist: Contiene los atributos de ID, título y tipo. Se ha establecido como clave primaria el título dado que es el que da el acceso a los videos y es único, para así no obtener repetidos.
- Video: Contiene los atributos de ID, título, fecha de subida, palabras clave, url relativa, duración (del video), url de YouTube y el título de la playlist. Se establece como clave primaria el ID del video, el cual da el acceso directo al video. Además se establece como clave foránea el título de la playlist, que indica a qué playlist se encuentra asociado el video.
- Comentarios: Contiene los atributos ID, Nick de usuario, cantidad de votos, contenido del comentario, fecha de publicación y el id del video. La clave primaria es el id del comentario y la clave foránea corresponde al id del video en el cual está realizado.

Las entidades pasivas pretenden representar datos de los cuales puede ser necesario tener un registro, pero que no intervienen directamente en el modelo. Estas tablas generadas se explican en detalle a continuación:

- No_playlist: Esta tabla pretende almacenar los datos de las playlist que se encuentran repetidas, aunque poseen distinto ID. Si bien, la clave primaria es el

título, se almacenan dado que en algún momento el repositorio puede asignar está ID a otros títulos y así, no queden fuera de la recopilación.

- Notaplaylist: Esta tabla pretende almacenar el id de las playlist que no retornan un archivo JSON o que no se encontró el sitio dentro del repositorio.

5.5.2 Mejora a la forma de acceder y realizar solicitudes a Khan Academy

Teniendo en conocimiento el código del Recolector Original y su funcionalidad, además del modo de operar por parte del repositorio (Khan Academy), se propuso modificar la función storevideos, debido a los problemas de respuesta que se tenían al momento de conectarse al servidor. Para esto se dividió en dos módulos.

La primera **storeplaylist**, que realizaba prácticamente lo mismo que storevideos, pero solo hasta el guardado de la información de la playlist, sin conectarse a Khan para rescatar al mismo tiempo el video.

Al momento de recuperar la información del video, era necesario conectarse a Khan Academy por medio de la API. El recolector original tenía configurada la identificación de los videos por su ID, pero al haberse realizados cambios en la API de Khan Academy, el acceso ahora se debe realizar por medio del título del video. Para ello, la función **storevideos** ha sido modificada, en la cual se cambió al identificador por el título, dejándolo consistente con lo que menciona la API.

Otra mejora al código original fue el uso de la función try: y except: (incorporadas en python) para solventar problemas con caracteres especiales. Esto, ya que cuando se tenían caracteres en un título como ć, ç, u otros, no es posible conectarse a través de HTTP. Al ocurrir esto, el recolector original mostraba por pantalla que no era posible recuperar el video y se detenía su ejecución, pero con las mejoras realizadas es posible mostrar dejar un registro que no pudo acceder al video y continuar la ejecución normal del procedimiento.

5.5.3 Mejoras a la forma de almacenar los datos

Una mejora implementada fue la solución para recuperar videos que no se podían decodificar, ya fuera porque no devolvían un archivo JSON, o porque en el servidor retiraron dicho recurso. Para esto, se ocupó la misma estrategia del apartado anterior

(función try: y except:), pero los que no se podían recuperar se agregaron en la tabla llamada Notaplaylist, guardando el título de la playlist a buscar y conservando estos registros.

En cuanto a la función __**storeplaylistinfo**, se mejoró la forma de ingresar los datos a la tabla, ya que es posible encontrar playlist duplicadas. Esto ocurre dado que el repositorio tenía playlist con distinto id pero mismo título, y como la forma de acceso al recurso es por título, se accedía dos veces a un mismo. Esto generaba problema al insertar la tupla, pues esta arrojaba errores. Esto se mitigó al usar la función try:, pues al momento de no poder ingresar una playlist ya existente, se almacenará en una tabla llamada Noplaylist, donde estarán las playlist de distinto id, pero que tiene el mismo título.

En cuanto a la función __**storevideoinfo** se consideró que recuperara más información relacionada a la url de YouTube, la descripción del video y la cantidad de vistas que tuvo el video, aunque finalmente el texto JSON no retornaba dicho dato (cambios en las API, en los datos que retornaba). Como este Recolector puede ser ocupado todas las veces que sea necesario para agregar videos o playlist nuevas que se ingresen a Khan, podría traer problemas al momento de insertar la tupla, a causa de la duplicidad de clave primaria, ya que se tratara de ingresar el mismo video, cuando se utilice ese modulo, es por esto que se usó el try: para que cuando no se pudiera insertar este imprimiera por pantalla de que el video ya existe almacenado y no lo agregue a la base de datos.

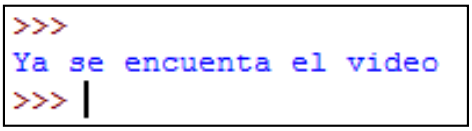


Ilustración 20: Resultado visible cuando se encontraba el video en la tabla.

5.5.4 Implementación de una función que recupere los comentarios de videos accesibles de Khan Academy

Dado que el recolector original no tenía una función que pudiese recolectar los comentarios desde Khan Academy, fue necesario implementar una función que recorriera las URL de los videos existentes para cargarlas en un navegador y poder extraer los comentarios realizados.

Nota: Recordar que los comentarios no son accesibles por medio de la API del repositorio Khan Academy, por lo cual, para poder recuperarlos, hay que acceder al panel de comentarios de cada video a través de la carga del sitio web en el navegador.

Para ello, **scrap_videos** quedo prácticamente igual, solo se suprimió el break, ya que como __scrap_video no estaba 100% creado, rompía el ciclo para trabajar solo con un video. Es por esto que donde se realizaron mejoras importantes fue en el módulo **__scrap_video**. Este módulo llegaba hasta el recurso, hacía clic al campo “Tips & Thanks” y recuperaba en una variable los comentarios. Lo que se debía realizar era agregar dicha información a la base de datos, y para esto se utilizó una variable, donde se debía recorrer la información recopilada y se agregaba el texto a la variable creada. Luego de esto era necesario diferenciar un comentario de otro, ya que el contenido se encontraba en texto plano como se muestra en la siguiente figura.

```
this video is not even 10 minits but it helps so much.
3 Votes   • Comment • Flag 2 years ago by   Christine
Khan Academy, please make an exercise set for absolute value inequalities. The videos are great (as always), but the application of it is
necessary for long term memory. Thanks!
2 Votes   • Comment • Flag 6 months ago by   Mo
cool Sal uses cursive. my english teacher doesn't like that
2 Votes   • Comment • Flag about a year ago by   marko
```

Ilustración 21: Ejemplo de comentarios recopilados (3 comentarios).

Para identificar cada uno de los comentarios, fue necesario incorporar un buscador de texto implementado en python, donde se buscan palabras estratégicas para diferenciar un comentario de otro. Por ejemplo, cada comentario tenia incluido el término” Comment”: en caso de que no se encuentre, significaba que el video no tenía comentarios, o que ya se llegó al final del texto plano. En caso contrario, se recorre de forma inversa desde la posición de la palabra encontrada, hasta encontrar el salto de línea, y por lo tanto, hasta este carácter seria el comentario.

Siguiendo en la misma línea, es posible rescatar la cantidad de votos que posee el video. Para ello, se busca desde la última posición de ubicación de comentarios hasta encontrar la letra “V”. Una vez localizada, se almacena el valor asociado. Luego de esto, se busca la palabra “by”. Con ello es posible recuperar el usuario que realizo el comentario. El nombre del usuario se identifica hasta encontrar el salto de línea. Luego se reiniciaba las variables de inicio, para partir desde el segundo comentario, luego se llama a la función que guarda la información del video a la base de datos temporal.

El procedimiento explicado anteriormente queda más claro con el ejemplo que se presenta a continuación:

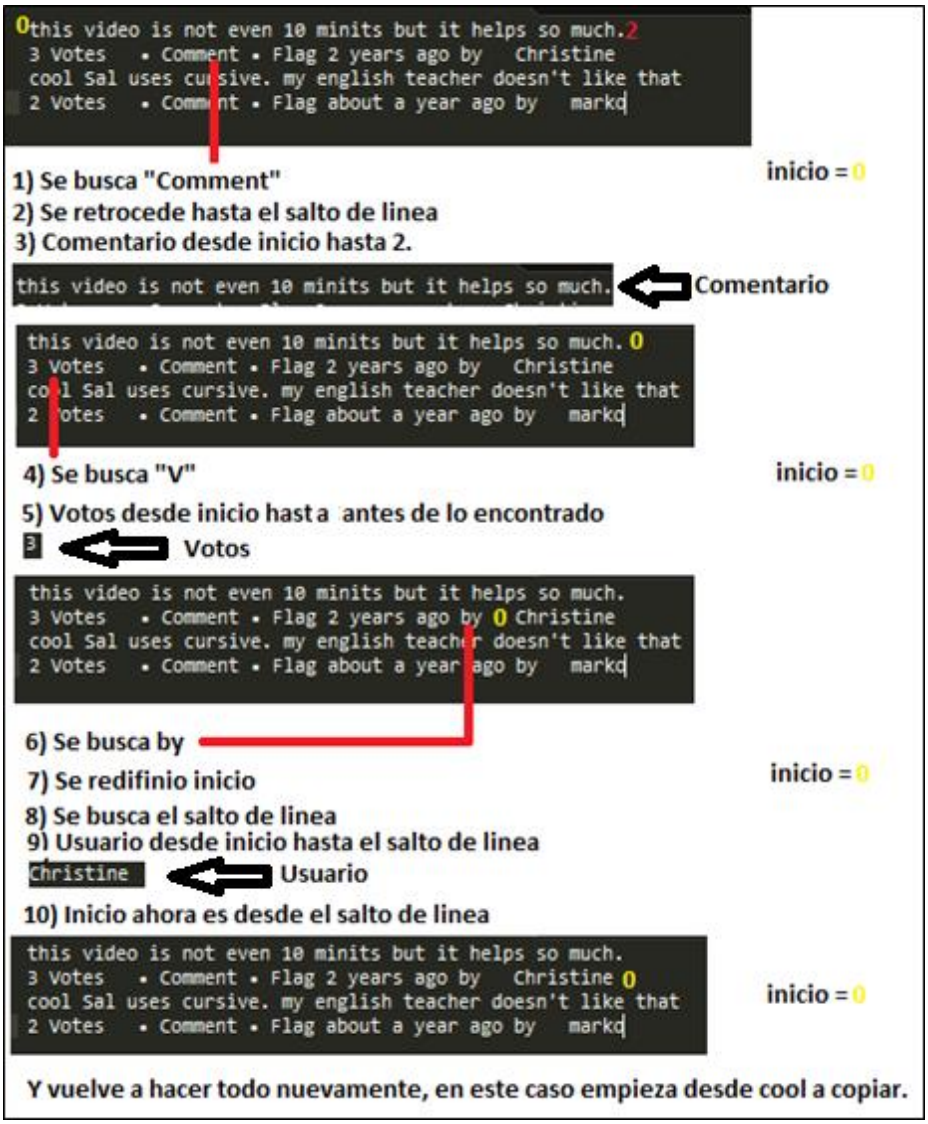


Ilustración 22: Ejemplo del proceso de división de los comentarios.

Otra mejora realizada al Recolector fue la creación de **__storecommentinfo**, donde se inserta la información del video a la base de datos. Para ello, se reemplazan las comillas simples por espacios, luego se consulta si el comentario ya existe en la base de datos, pero asociado al video y usuario que lo realizó. Esto, ya que pueden haber comentarios que digan lo mismo, pero asociados a distintos videos o a distintos usuarios. En caso de que ya esté en la base de datos, en el mismo video y con el mismo usuario,

solo se actualiza la cantidad de votos, en caso contrario, se guarda la tupla de la información del video completa.

5.5.5 Implementación de módulo de creación de archivo XML y posterior llenado.

Primero que todo, fue necesario suprimir por completo las instrucciones dump del recolector original y reemplazarlas por nuevos módulos. El primero de ellos es **createtxt**, donde se creara un texto plano, con toda la información de los recursos educativos (se explica más detalladamente en el apartado 5.5.5).

Lo siguiente que se añadió fue el módulo **LOM**, donde se realizaba prácticamente lo mismo que en createtxt, pero no se realizaba en un texto plano, más bien se creaba en un XML, que está de acuerdo a un estándar de recursos, como es el LOM. Para ello, es necesario realizar inserción de las etiquetas correspondientes a las categorías de LOM para luego llenar cada una de ellas con la información correspondiente a los recursos recuperados desde Khan Academy.

5.6 Entorno grafico

En este apartado se presentan las diversas funciones y ventanas del entorno grafico presentes para los diversos módulos creados. La importancia de este entorno gráfico es hacer más amigable e intuitiva la experiencia con el usuario que desee realizar la recuperación de objetos de aprendizaje desde el repositorio Khan Academy.

Al ejecutar el recolector, se recibe al usuario con una ventana de bienvenida, donde se explica de forma breve la función de dicho Recolector.

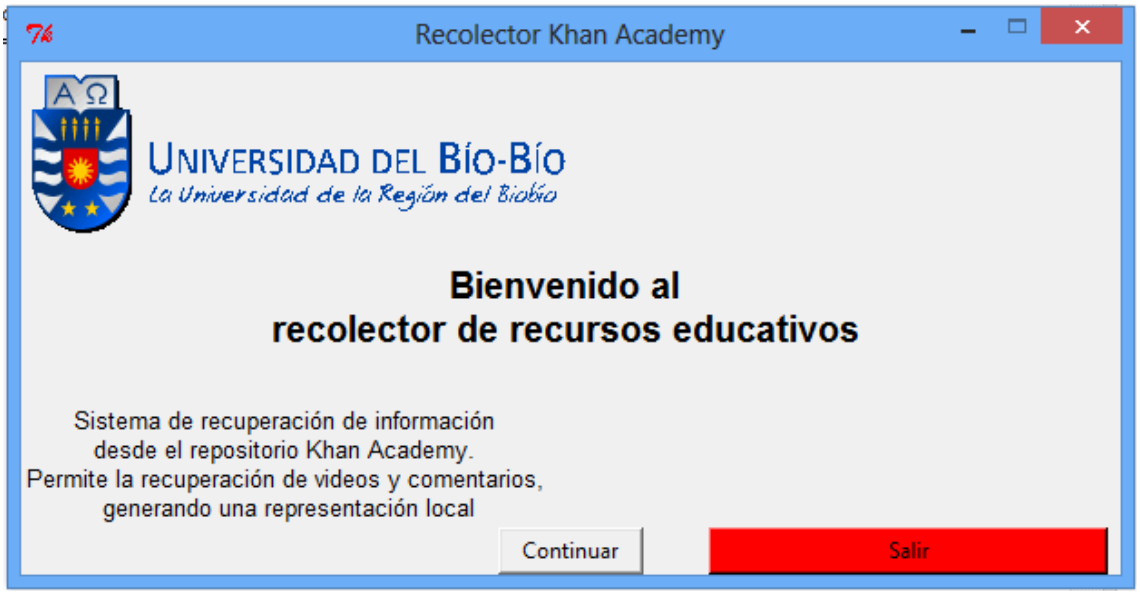


Ilustración 23: Ventana de Bienvenida al Recolector.

La ilustración muestra que dicha ventana de bienvenida da opción a dos caminos: uno es hacer clic a Continuar y el otro es hacer clic a Salir del Recolector. Al oprimir el botón Continuar, se abre el primer módulo que se sin él no se puede pasar a los siguientes.

5.6.1 Recuperar playlist.

Al hacer clic en el botón “continuar” de la ventana de bienvenida, nos redirige a la ventana de recuperación de playlist. En ella encontramos el primer módulo que nos permite recuperar todas las playlist desde el repositorio Khan Academy. La primera vez que se ejecuta el módulo, no habrá ninguna playlist almacenada, por lo que recomienda realizar una recuperación para poder continuar. Si ya se cuentan con playlist recuperadas, es posible continuar al módulo de recuperación de videos o actualizar la lista existente. Para iniciar la recuperación o actualizar la lista existente, se debe hacer clic en el botón “actualizar”. Para continuar, se debe hacer clic en el botón “continuar”.

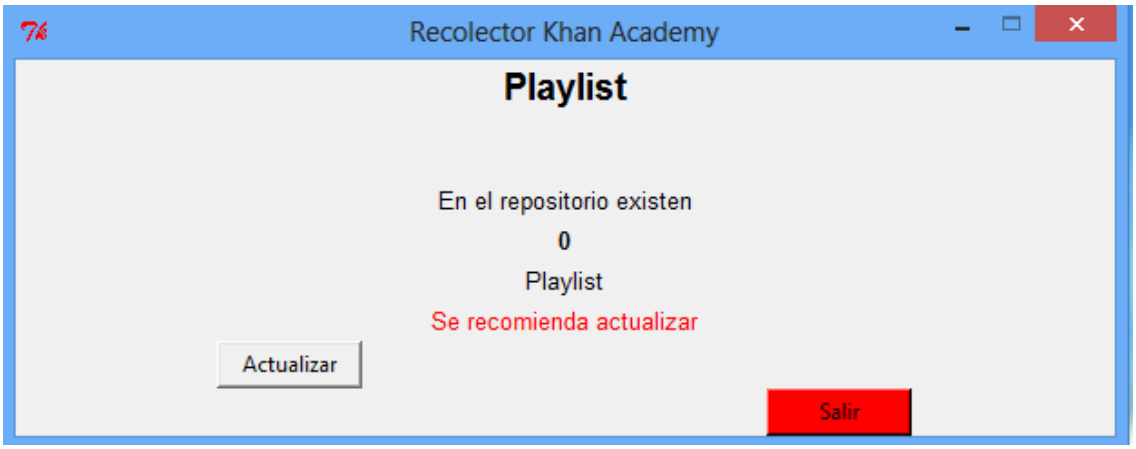


Ilustración 24: Ventana del módulo playlist.

Al actualizar, se despliega una ventana de confirmación donde se tiene la opción de confirmar el inicio de la recuperación o actualización de la lista existente (según sea la instancia), o volver atrás (ilustración 25). Cuando se decide recuperar, este módulo estará trabajando por unos minutos, ya que internamente se debe conectar al servidor del repositorio Khan Academy. Una vez realizado este proceso, se muestra una ventana donde se confirma el término del proceso de recopilación, donde se tiene dos opciones: una donde muestre el total de playlist nuevas agregadas (ilustración 26), o de que terminó el proceso sin encontrar nuevas playlist (ilustración 27).

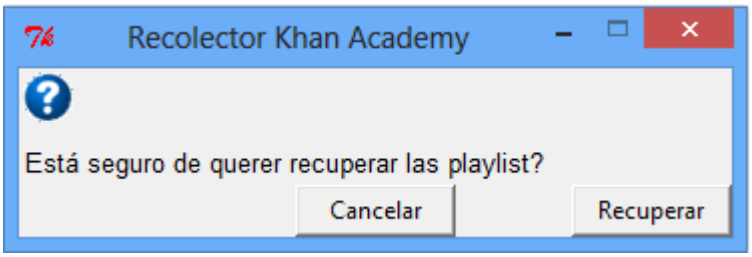


Ilustración 25: Ventana de decisión para recuperar las playlists.

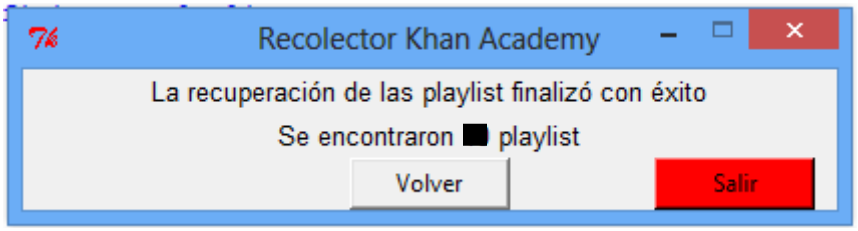


Ilustración 26: Ventana de finalización de recuperación de las playlists (opción 1).

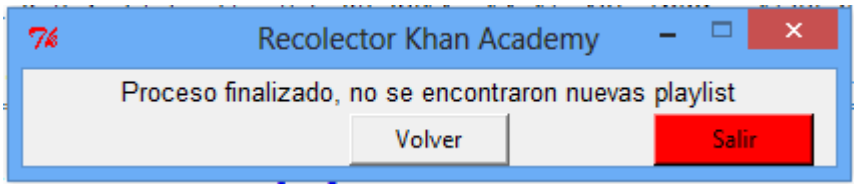


Ilustración 27: Ventana de finalización de recuperación de las playlists (opción 2).

5.6.2 Recuperar videos

Al hacer clic en el botón “continuar” en el módulo de recuperación de playlist, nos redirige a la ventana de recuperación de videos. En ella podremos recuperar todos los videos asociados a las playlist recuperadas anteriormente desde el repositorio Khan Academy. La primera vez que se ejecuta el módulo, no habrá ningún video almacenado, por lo que recomienda realizar una recuperación para poder continuar. Si ya se cuentan con videos recuperados, es posible continuar al módulo de recuperación de comentarios o actualizar la lista existente. Para iniciar la recuperación o actualizar la lista existente, se debe hacer clic en el botón “actualizar”. Para continuar, se debe hacer clic en el botón “continuar”.

Ilustración 28: Ventana del módulo videos.

Al actualizar, se despliega una ventana de confirmación donde se tiene la opción de confirmar el inicio de la recuperación o actualización de la lista existente (según sea la instancia), o volver atrás (ilustración 29). Cuando se decide recuperar, este módulo estará trabajando por unos minutos, ya que internamente se debe conectar al servidor del repositorio Khan Academy. Una vez realizado este proceso, se muestra una ventana

donde se confirma el término del proceso de recopilación, donde se tiene dos opciones: una donde muestre el total de videos nuevos agregados (ilustración 30), o de que terminó el proceso sin encontrar nuevos videos (ilustración 31).

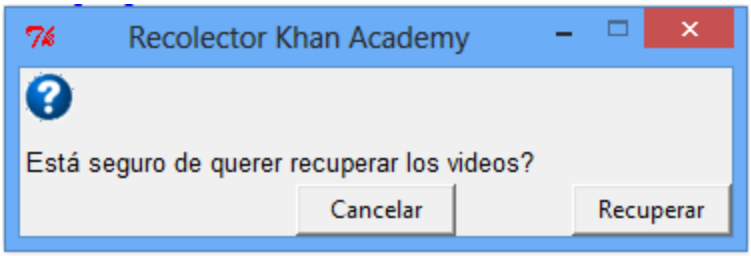


Ilustración 29: Ventana de decisión para recuperar los videos.

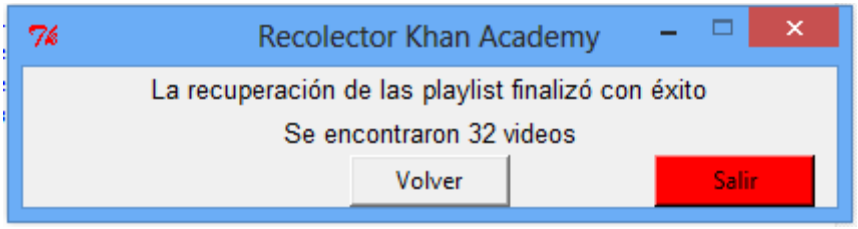


Ilustración 30: Ventana de finalización de recuperación de los videos (opción 1).

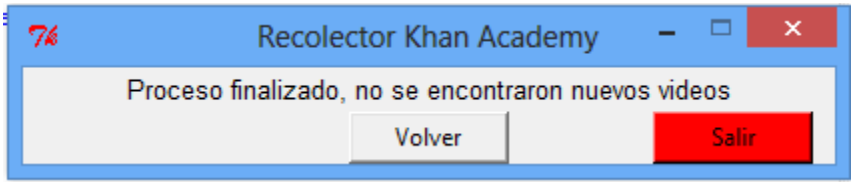


Ilustración 311: Ventana de finalización de recuperación de los videos (opción 2).

5.6.3 Recuperar comentarios

Al hacer clic en el botón “continuar” en el módulo de recuperación de videos, nos redirige a la ventana de recuperación de comentarios. En ella podremos recuperar todos los comentarios asociados a los videos recuperados anteriormente desde el repositorio Khan Academy. La primera vez que se ejecuta el módulo, no habrá ningún comentario almacenado, por lo que recomienda realizar una recuperación para poder continuar. Si ya se cuentan con comentarios recuperados, es posible continuar al módulo de actualización del repositorio local o actualizar la lista existente. Para iniciar la recuperación o actualizar

la lista existente, se debe hacer clic en el botón “actualizar”. Para continuar, se debe hacer clic en el botón “continuar”.

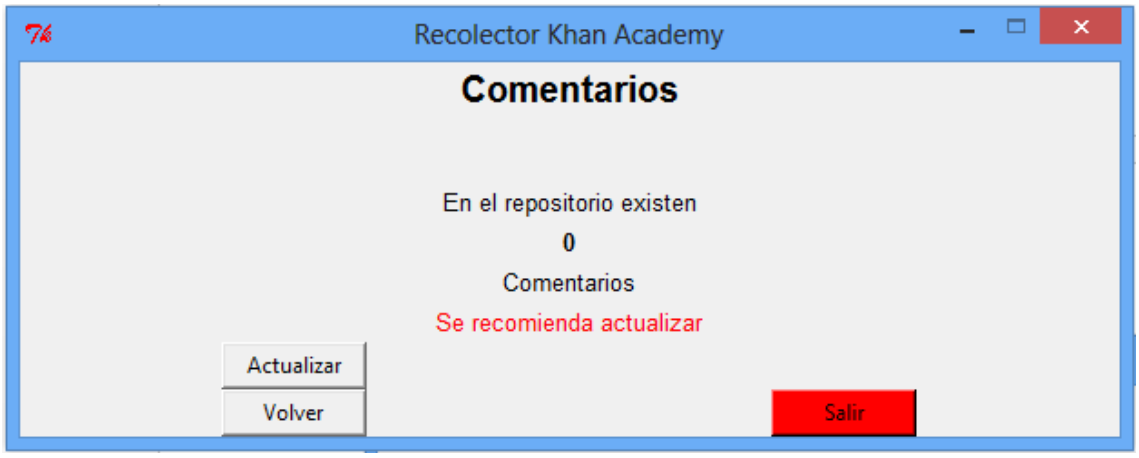


Ilustración 32: Ventana del módulo comments.

Al actualizar, se despliega una ventana de confirmación donde se tiene la opción de confirmar el inicio de la recuperación o actualización de la lista existente (según sea la instancia) o volver atrás (ilustración 33). La única diferencia con los módulos anteriores se aprecia al momento de confirmar la recuperación, pues se abrirá una pestaña de navegador. Es estos momentos, se procede video tras video para recuperar los comentarios de cada uno. Este proceso puede tardar horas, debido al gran volumen de videos que debe visitar para recuperar sus comentarios y todo dependerá de la velocidad de carga de la página. Una vez realizado este proceso, se muestra una ventana donde se confirma el término del proceso de recopilación, donde se tiene dos opciones: una donde muestre el total de comentarios nuevos agregados (ilustración 34), o de que terminó el proceso sin encontrar nuevos comentarios (ilustración 35).

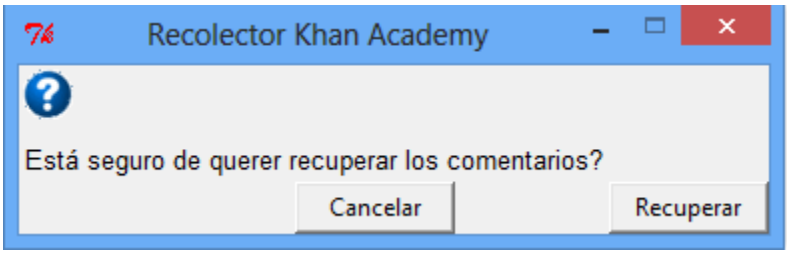


Ilustración 33: Ventana de decisión para recuperar los comentarios de todos los videos.

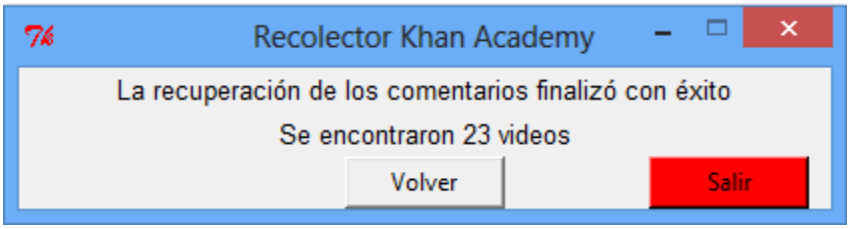


Ilustración 34: Ventana de finalización de recuperación de los comentarios de los videos (opción 1).

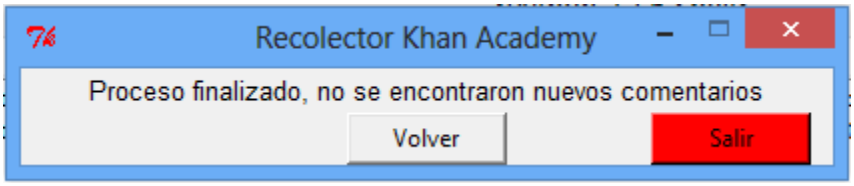


Ilustración 35: Ventana de finalización de recuperación de los comentarios de los videos (opción 2).

5.6.4 Crear archivo

Al hacer clic en el botón “continuar” en el módulo de recuperación de comentarios, nos redirige a la ventana de creación del repositorio. Aquí encontramos las opciones de crear un archivo TXT o crear la representación del repositorio, donde creara la representación estructurada (base de datos con formato LOM) y la representación semi estructurada (XML con formato LOM).

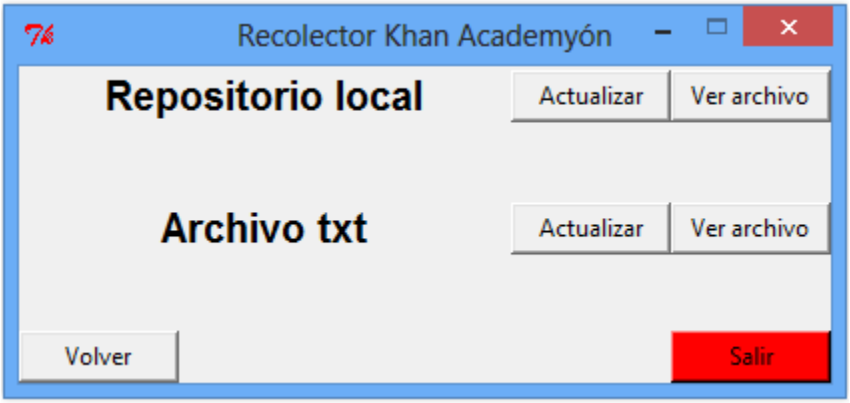


Ilustración 36: Ventana del módulo crear archivos.

5.6.4.1 Crear archivo TXT

Para crear un archivo TXT donde se muestre información de los objetos de aprendizaje almacenados, se debe dirigir al botón “actualizar” de la sección “Archivo txt”.

Al presionar “actualizar”, se despliega una ventana donde es necesario confirmar la creación del archivo (ilustración 37). Para continuar, se presiona el botón “crear” o si se requiere volver al Menú principal de este módulo, se presiona el botón “volver”. Cuando se confirme la creación, el programa trabajará por unos momentos, mientras llena el archivo con toda la información. Una vez finalizado el proceso, se despliega una ventana donde se confirma el término de creación con éxito (ilustración 38), luego se puede volver o si se quiere, se puede ver el archivo creado.

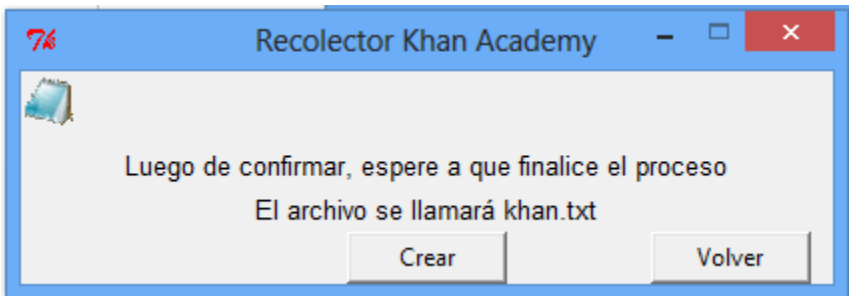


Ilustración 37: Ventana de confirmación para crear el archivo de texto.

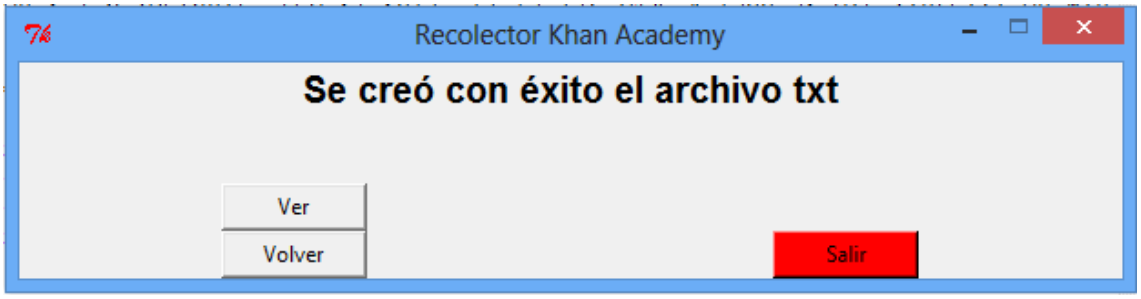


Ilustración 38: Ventana de finalización del archivo de texto creado.

5.6.4.2 Crear Representación del repositorio

Cuando se quiere crear la representación del repositorio (bases de datos relacional y archivo XML, ambos bajo el estándar LOM), se debe dirigir al botón “actualizar” de la sección Repositorio local. Tras presionar el botón “actualizar”, se despliega una ventana donde es necesario confirmar la creación de la representación del repositorio (ilustración 39). Para continuar, se presiona el botón “crear” o si se requiere volver al Menú principal de este módulo, se presiona el botón “volver”. Cuando se confirme la creación, el programa trabajará por unos momentos, ya que estará creando el archivo e insertando los datos correspondientes. Una vez finalizado el proceso, se despliega una ventana con el

mensaje de finalización (ilustración 40). Luego, se puede volver al Menú principal de este módulo o si se requiere, revisar ambos archivos. Tras presionar el botón “ver”, se abrirá automáticamente el archivo XML y a la vez, se desplegará una ventana donde se muestra la ruta de ambos archivos en el equipo en el cual se encuentra trabajando.

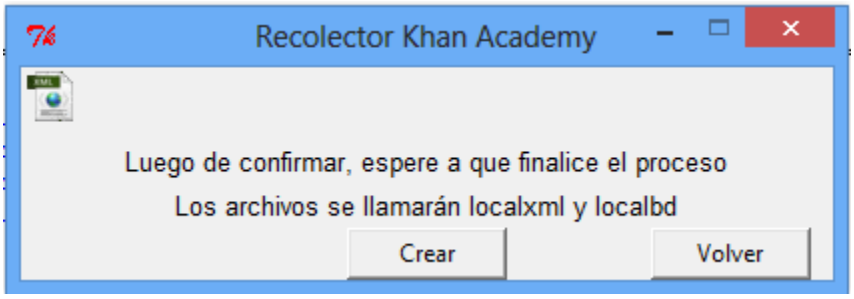


Ilustración 39: Ventana de opción de nombre para el archivo XML que se creara.

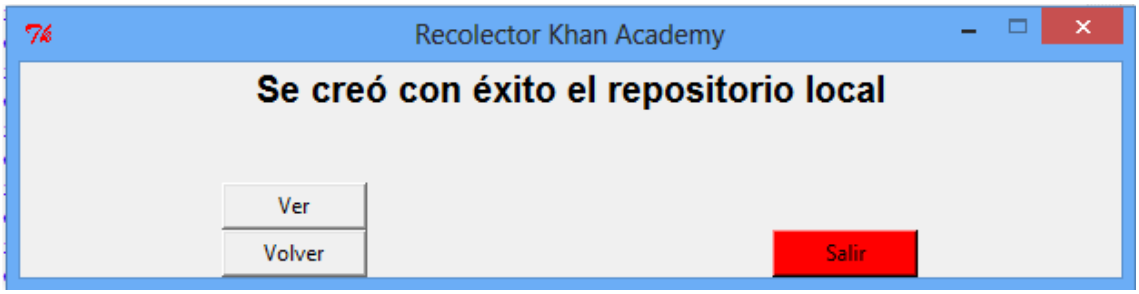


Ilustración 40: Ventana de finalización de creación de la representación.

5.6.5 Salir

La mayoría de las ventanas tienen un botón en rojo con el texto Salir. Al momento de presionar este botón, se despliega una ventana con una pregunta si realmente se quiere salir del Recolector. En caso de presionar en Salir se terminará la ejecución y se cerrarán todas las ventanas y procesos. En caso de cancelar, volverá a la ventana principal del módulo playlist, para seguir trabajando según la preferencia del usuario (ilustración 41).

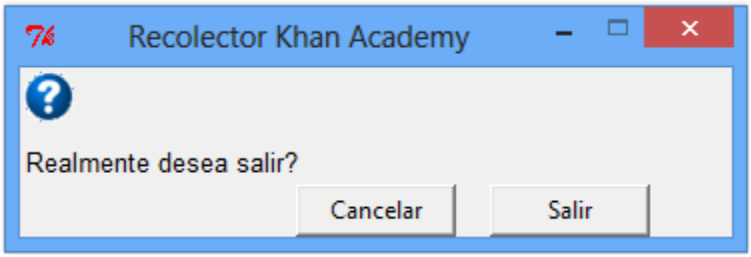


Ilustración 41: Ventana de decisión de querer salir o volver al Menú principal.

5.5 Tabla comparativa de los objetivos cumplidos entre Recolector original y la versión mejorada

La siguiente tabla, muestra una comparativa entre los objetivos alcanzados por el recolector original versus los objetivos alcanzados por el recolector propuesto en el proyecto. Se aprecia claramente que el recolector propuesto en el proyecto obtiene ventajas gracias a las mejoras propuestas en el apartado 5.4 y el desarrollo de un entorno gráfico propuesto en el apartado 5.5.

Función	**Recolector Original**	**Recolector propuesto en el proyecto**
CREAR BASE DE DATOS	Si.	Si.
ABRIR BASE DE DATOS	Si.	Si.
RECOPILAR PLAYLISTS	No, debido a los cambios realizados por parte de Khan Academy en la API del sitio.	Si, y como en la API ahora se busca a través del título de la playlist, se dividieron en dos tablas distintas, las que tienen mismo título y distinto id.
RECOPILAR VIDEOS	No, pues el Recolector se caía al momento de recibir respuesta por parte del servidor.	Si. Se accede a la lista completa de videos. En caso de haber repetidos o con caracteres especiales, continúa con el video siguiente.
RECOPILAR COMENTARIOS	No, solo se tenía una idea de que podía ser recopilado	Sí, pero en los videos más comentados se recuperan

	a través de una implementación realizada en Java.	los 20 mejores, de acuerdo a los votos de los mismos usuarios.
TRATADO DE ALTERNATIVA SOBRE LOS CARACTERES ESPECIALES	No, ya que cuando se encontraba con un carácter especial, ya sea en el título, id o en la url, al tratar de ir a ese video se caía el Recolector.	Si, ahora al encontrar un carácter especial, los omite, siguiendo con el siguiente video, y se menciona que en aquel video no se pudo recuperar los comentarios.
CREAR TXT CON DATOS RECOPILADOS	Sí, pero sin formato.	Si, con una estructura que muestra la información de forma ordenada para cada uno de los recursos recolectados.
CREAR XML	No.	Si, se puede crear un XML, el cual viene con el formato del estándar LOM.
IMPLEMENTACIÓN CON INTERFAZ GRAFICA	No.	Si, se desarrolló una implementación básica de una interfaz gráfica, para hacer más agradable la experiencia del usuario.

5.8 Caso de estudio.

En este apartado se describe cómo funciona el Recolector en su versión final, donde se explica de forma breve lo que realiza internamente y los hechos perceptibles por

los usuarios, cuando se inicia la ejecución completa de una recolección de datos desde el repositorio Khan Academy.

Se presenta una ejecución general del recolector, donde se recolectan datos de más de 5000 videos y un caso particular para una playlist. Esto, debido a que la cantidad de datos hacen muy extenso la presentación en este informe.

5.8.1 Aplicación de módulos

A partir de las modificaciones ya realizadas, se pudo contar con el Recolector en su versión final, para poder realizar la extracción completa de la información necesaria. Esto no fue realizado como un programa único, sino más bien fue dividido en módulos, que podían trabajar en forma independiente. Esto quiere decir que no es necesario ejecutar todos los módulos de una vez y hace posible reanudar la recuperación desde el módulo requerido. Por ejemplo, cuando ya se tienen los datos correspondientes a las playlist, los videos y comentarios y solo se necesite generar el archivo XML, No es necesario realizar este proceso nuevamente, por lo que solo se realiza el proceso de generar el archivo XML.

En esta división realizada al recolector, se pudieron definir siete módulos diferentes, aunque uno de ellos le presta servicio a otros cinco.

A continuación, se presenta el caso de estudio general, donde se muestra la ejecución para la recolección de datos de las playlist y videos de todo el repositorio. Una vez finalizada esta etapa, se muestra el caso particular de una playlist donde es posible apreciar la representación relacional y XML de los datos recuperados del caso particular.

5.8.1.1 Módulo createdb

Un usuario no ejecuta este módulo de forma manual, más bien, éste es llamado al principio de cada ejecución del recolector. El recolector llama al módulo *createdb*, el cual ha quedado estructurado de la siguiente forma:

```
def createdb(dbname,ventana):
    """
    Conecion a la base de datos, que primeramente no tiene nada.
    """
    conn = sqlite3.connect(dbname)

    """
    Creacion de las tablas necesarias.
    """
    try:
        conn.execute('''CREATE TABLE playlist
                        (id text, title text primary key, kind text)''')
        conn.execute('''CREATE TABLE Noplaylist
                        (id text, title text, kind text)''')
        conn.execute(("CREATE TABLE playlist_video (idplaylist text, idvideo text)"))
        conn.execute('''CREATE TABLE video
                        (id text primary key, title text,  date_added text, keywords text, relative_url text, duration text,
                           views text, url text, description text)''')
        conn.execute('''CREATE TABLE comments
                        (videoid text, userid text, usernick text, votes integer, content text, date text)''')
        conn.execute('''CREATE TABLE Notaplaylist
                        (title text)''')
        """
        Creacion de las tablas necesarias para bd local del repositorio.
        """
        conn.execute('''CREATE TABLE General
                        (Correl text primary key, Title text, Language text, Description text, Keyword text, Coverage text,
                           Structure text, AggregationLevel text)''')
        conn.execute('''CREATE TABLE Identifier
                        (Correl text,Catalog text, Entry text primary key)''')
        conn.execute('''CREATE TABLE IdentifierYoutube
                        (Correl text primary key, Catalog text, Entry text)''')
        conn.execute('''CREATE TABLE Resource
                        (Correl text primary key, Description text)''')
        conn.execute('''CREATE TABLE Lifecycle
                        (Correl text primary key, Version text, Status text)''')
        conn.execute('''CREATE TABLE MetaMetadata
                        (Correl text primary key, MetadataSchema text, Languaje text)''')
        conn.execute('''CREATE TABLE Technical
                        (Correl text primary key, Format text, Size text, Location text, InstallationRemarks text,
                           OtherPlataformRequirements text, Duration text)''')
        conn.execute('''CREATE TABLE Requeriment
                        (Correl text)''')
        conn.execute('''CREATE TABLE OrComposite
                        (Correl text primary key, Type text, Name text, MinimunVersion text, MaximunVersion text)''')
        conn.execute('''CREATE TABLE Educational
                        (Correl text primary key, InteractivityType text, LearningResourceType text, InteractivityLevel text,
                           SemanticDensity text, IntendedEndUserRole text, Context text, TypicalAgeRange text, Difficulty text,
                           TypicalLearningTime text, Description text, Language text)''')
        conn.execute('''CREATE TABLE Rights
                        (Correl text primary key, Cost text, CopyrightAndOthRestrictions text, Description text)''')
        conn.execute('''CREATE TABLE Relation
                        (Correl text primary key, Kind text, Correl2 text)''')
        conn.execute('''CREATE TABLE Clasification
                        (Correl text primary key, Purpose text, Description text, Keyword text)''')
        conn.execute('''CREATE TABLE TaxonPath
                        (Correl text primary key, Source text)''')
        conn.execute('''CREATE TABLE Taxon
                        (Correl text primary key, Id text, Entry text)''')
        conn.execute('''CREATE TABLE Annotation
                        (ID integer primary key, Correl text, Entity text, Date text, Description text, Votes text)''')
        conn.execute('''CREATE TABLE Contribute
                        (Correl text primary key, Rol text, Entity text, Date text)''')
    except:
        print "Ya estan creadas las tablas"
    return conn
```

Ilustración 42: Modulo createdb.

Lo que se realiza en este módulo es: Primero, en la línea seis se establece la conexión de la base de datos (a través de sqlite3), para luego poder hacer operaciones que se requieran. Desde la línea doce hasta la veintitrés se crean las estructuras de la Base de datos temporal, y desde la línea veintisiete hasta la sesenta y cuatro se crea la estructura de la base de datos de acuerdo al estándar LOM (estas dos estructuras son creadas a través de sentencias SQL).

La base de datos temporal alberga los datos para luego llenar ambas representaciones finales del repositorio. La primera tabla creada fue playlist (en la línea

doce y trece), la cual tiene tres atributos (id, title y kind). La segunda tabla fue Noplaylist (línea catorce y quince), que tiene los mismos atributos de playlist, es decir id, title y kind. Luego crea playlist_video (línea dieciséis), la cual tiene como atributos idvideo e idplaylist. En la línea diecisiete hasta la dieciocho se crea la tabla video, la cual guarda la información relevante a esta y tiene como atributo id, title, date_added, keywords, relative_url, duration y views; aunque este último no está siendo ocupada, ya que como cambiaron las API de Khan ya no puede ser recuperada la información sobre la cantidad de vistas que tienen los videos. La tabla comments, es creada con la sentencia de la veinte y veintiuno, donde se guarda tanto el videoid, userid, usernick, votes, content y date, pero no se puede recuperar ni el id del usuario, ya que no es entregado y tampoco se puede recuperar la fecha en que fue realizado, ya que solo muestra el tiempo atrás que se realizó el comentario, y esto no nos da una fecha exacta, sino más bien se puede encontrar una fecha aproximada aunque puede variar muchísimo cuando se trata de X años atrás. También se crea la tabla Notaplaylist (en la línea veintidós y vientres), que tiene como único atributo title (el título de la playlist). Aparte de las seis tablas mencionadas con anterioridad, se crean otras diecisiete tablas, correspondientes a las tablas que contiene la representación relacional bajo el estándar de LOM, las cuales no son utilizadas para crear la representación del repositorio.

En la línea sesenta y seis, se imprime un mensaje de que estaban creadas las tablas, esto ocurre cuando se trata de crear la estructura del repositorio y estas ya están creadas.

5.8.1.2 Módulo opendb

El módulo *opendb* presta servicio a los demás módulos ejecutables.

```
1  def opendb(dbname):
2      conn = sqlite3.connect(dbname)
3      return conn
```

Ilustración 43: Módulo opendb.

La función de este módulo, es simplemente la acción de la línea número dos, donde se realiza la conexión a la base de datos, conectando a través de sqlite3, y así dejar disponible la base de datos a los demás módulos. Este módulo se ejecuta al momento de iniciar las operaciones de Recuperar playlist, Recuperar videos, Recuperar

comentarios, Crear archivo txt y Crear representación del repositorio, iniciadas desde el menú inicio.

5.8.1.3 Módulo storeplaylists

Un usuario, al ejecutar la operación “Recuperar playlist” en el menú de inicio, el recolector llama a la función storeplaylists.

```
1  def storeplaylists(db,ventana):
2      """
3      Recopilación de las playlist
4      """
5      cuenta = db.execute("SELECT Count(*) FROM playlist")
6      for veri in cuenta:
7          total = veri[0]
8      uri = "/api/v1/playlists"
9      conn = httplib.HTTPConnection(KHANACAD_URI)
10     conn.request("GET", uri)
11     res = conn.getresponse()
12     data = res.read()
13     decoded = json.loads(data)
14     for playlist in decoded:
15         __storeplaylistinfo(db, playlist)
16     print "Se terminó la recuperación de las playlist"
17     cuenta = db.execute("SELECT Count(*) FROM playlist")
18     for veri in cuenta:
19         total2 = veri[0]
20     if total == total2:
21         ventana.destroy()
22         conn2 = opendb("temporal.db")
23         v0=Tk()
24         v0.resizable(0,0)
25         v0.title("Recolector Khan Academy")
26         Label(v0,font="Arial 10",text="Proceso finalizado, no se encontraron nuevas playlist",width=50).grid(row=0,column=0,columnspan=4)
27         Button(v0,text="Volver",width=10,command=lambda: uno(v0,conn2)).grid(row=12,column=0,columnspan=4)
28         Button(v0,text="Salir",width=10,bg='red',command=lambda: salir(v0)).grid(row=12,column=3,columnspan=4)
29     else:
30         ventana.destroy()
31         conn2 = opendb("temporal.db")
32         v0=Tk()
33         v0.resizable(0,0)
34         v0.title("Recolector Khan Academy")
35         total2 = total2 - total
36         Label(v0,font="Arial 10",text="La recuperación de las playlist finalizó con éxito",width=50).grid(row=0,column=0,columnspan=4)
37         Label(v0,font="Arial 10",text="Se encontraron " +str(total2)+ " playlist",width=50).grid(row=1,column=0,columnspan=4)
38         Button(v0,text="Volver",width=10,command=lambda: uno(v0,conn2)).grid(row=12,column=0,columnspan=4)
39         Button(v0,text="Salir",width=10,bg='red',command=lambda: salir(v0)).grid(row=12,column=3,columnspan=4)
40     db.close()
41     v0.mainloop()
42     conn.close()
```

Ilustración 42: Modulo soreplaylists.

Lo que se realiza en este módulo es ocupar la librería httplib para tratar de conectarse al repositorio Khan por medio de la API (esto se puede ver en la línea nueve, para que en la siguiente se envié formalmente la solicitud al servidor, a través del método GET). En la línea doce, se recibe la respuesta del repositorio, la cual es codificada en la línea trece, debido a que el servidor envía la información en un texto de tipo JSON. Ahora que se tienen los archivos en formato JSON, en la línea catorse se empieza a recorrer cada uno de ellos, a través de un ciclo for, para poder hacer llamado a la función __storeplaylistinfo, la cual se detalla a continuación:

```
1   def __storeplaylistinfo(db, playlist):
2       """
3       Stores the information of a playlist in the db
4       """
5       title = playlist["title"]
6       title = title.replace("'", " ")
7       id = playlist["id"]
8       id = id.replace("'", " ")
9       kind = playlist["kind"]
10      kind = kind.replace("'", " ")
11      try:
12          db.execute("INSERT INTO playlist VALUES ('"+ id +"','"+title+"','"+kind+"')")
13          db.commit()
14      except:
15          db.execute("INSERT INTO Noplaylist VALUES ('"+ id +"','"+title+"','"+kind+"')")
16          db.commit()
17      return
```

Ilustración 43: Función __storeplaylistinfo

La función __storeplaylistinfo guarda la información de las playlists recuperadas. Para ello, en la línea cinco recibe el título de la playlist, para que en la siguiente acción reemplace las comillas simples por espacios. De igual manera, lo realiza con el id y el kind de la playlist, para que se pueda ingresar la información de forma correcta. A continuación se tienen dos opciones: la primera es la línea doce y trece, donde se guarda en la base de datos la información importante, como es el id, el título y el tipo; pero si esto no se puede realizar, lo anula y realiza la segunda opción: guardar la misma información, pero en otra tabla, debido a que ese título ya se encontraba guardado.

Volviendo a la función storeplaylist, y tras haber recorrido todo el archivo JSON y de guardar la información correspondiente, se imprime un texto por pantalla para saber que finalizó con éxito (línea trece del código), como se muestra en la siguiente figura.

```
>>>
Se termino la recuperacion de las playlists
>>>
```

Ilustración 44: Resultado visible por pantalla (storeplaylists).

Volviendo desde la línea cinco hasta la siete, se cuenta la cantidad de playlist que existen en hasta ese momento guardada. Tomando el camino anterior desde la línea diecisiete hasta la diecinueve consulta por la cantidad de playlist guardadas, después de haber realizado el guardado, luego en la línea veinte se hace la consulta si los dos

números son iguales, en caso de serlo se crea una ventana (desde la línea veintiuno hasta la veintiocho), la cual tiene el mensaje de éxito, pero que no se encontraron nuevas playlist, además de dos botones uno de salida y otro para volver al menú, los cuales se definen en la línea veintisiete y veintiocho respectivamente. Si los dos números eran distintos se crea una ventana (desde la línea treinta hasta la treinta y nueve), la cual tiene el mensaje de éxito y el número de playlist nuevas encontradas, además de dos botones uno de salida y otro para volver al menú, los cuales se definen en la línea treinta y ocho y treinta y nueve respectivamente.

Otra forma de corroborar su ejecución es la a través de un programa que permita la visualización de la base de datos (como por ejemplo SQLite Database Browser 2.0 b1), y comprobar que ahora ya tienen datos la tabla playlist y Noplaylist (en caso de repetidos).

Tras realizar una ejecución del módulo, fue posible encontrar 1052 playlist disponibles en el repositorio de Khan Academy. Además, se encontraron 111 playlist repetidas, esto quiere decir con distinto ID pero mismo título. Las ilustraciones 44 y 45 muestran el estado de las tablas Noplaylist y Playlist respectivamente luego de haber realizado la recuperación.

Table: playlist

	id	title	kind
1	xdea303c8	"The One World Sch	Topic
2	xdfe2a1cb	1.0 Welcome to the	Topic
3	x749b95a7	1.1 What is big histo	Topic
4	xc6e336fa	1.2 Complexity and	Topic
5	x57bacc4d	1.4 Origin stories	Topic
6	xaa369a76	1.5 Ways of knowin	Topic
7	x493f9d7a	10.0 What does the	Topic
8	x5d0c82a0	10.1 Big history s m	Topic
9	xea0ccf23	1400s quiz	Topic
10	xd7054342	1913 Centennial Cel	Topic
11	xcc25eb22	2D divergence theor	Topic
12	x938795a0	2.0 The big bang	Topic
13	x71f502a6	2.1 How did our viev	Topic
14	x7c5efc82	2.2 What emerged f	Topic
15	xe3c2c35e	2.3 Ways of knowin	Topic
16	x595e11fb	2003 AIME	Topic
17	x0b27fe22	2008 bank bailout	Topic

< 1 - 1000 of 1052 >

Ilustración 46: Tabla playlist

Table: Noplaylist

	id	title	kind
1	xc1404ea3	Absolute value	Topic
2	x828b0a04	Absolute value	Topic
3	x167c02e5	Acceleration	Topic
4	xc1c09c3c	Adding and subtract	Topic
5	x534e3a4f	Adding and subtract	Topic
6	x8ff43f34	Adding and subtract	Topic
7	x5f6abeb2	Adding and subtract	Topic
8	x1898834a	Adding and subtract	Topic
9	xc2ab8f63	Adding and subtract	Topic
10	x9d287e4b	Adding and subtract	Topic
11	x4d56a2fa	Area	Topic
12	xea43dd8d	Flanders	Topic
13	x12007035	Florence	Topic
14	x5b36944f	Venice	Topic
15	x5ff9185d	Basic probability	Topic
16	x944c39f8	Basic probability	Topic
17	xeb6fb8e4	Circles	Topic

< 1 - 111 of 111 >

Ilustración 45: Tabla Noplaylist.

5.8.1.4 Módulo storevideos

Un usuario, al ejecutar la operación “Recuperar videos” luego de haber apretado continuar en el módulo playlist, el recolector llama a la función *storevideos*, la cual se detalla a continuación:

```
1  def storevideos(db,ventana):
2      """
3      Selecionar los titulos para buscar los videos de cada playlist
4      """
5      cuenta = db.execute("SELECT Count(*) FROM video")
6      for veri in cuenta:
7          total = veri[0]
8      res = db.execute("SELECT id, title FROM playlist")
9      for row in res:
10         buscar = row[1]
11         buscar = buscar.replace(" ", "%20")
12         conn = httplib.HTTPConnection(KHANACAD_URI)
13         uri = "/api/v1/playlists/"+buscar+'/videos'
14         try:
15             conn.request("GET", uri)
16             res = conn.getresponse()
17         except:
18             print uri
19         try:
20             data = res.read()
21             decoded = json.loads(data)
22             for video in decoded:
23                 __storevideoinfo(db, video, row[0])
24         except:
25             db.execute("INSERT INTO Notaplaylist VALUES ('"+buscar+"')")
26             db.commit()
27     print "Se terminó la recuperación de los videos"
28     cuenta = db.execute("SELECT Count(*) FROM video")
29     for veri in cuenta:
30         total2 = veri[0]
31     if total == total2:
32         ventana.destroy()
33         conn2 = opendb("temporal.db")
34         v0=Tk()
35         v0.resizable(0,0)
36         v0.title("Recolector Khan Academy")
37         Label(v0,font="Arial 10",text="Proceso finalizado, no se encontraron nuevos videos",width=50).grid(row=0,column=0,columnspan=4)
38         Button(v0,text="Volver",width=10,command=lambda: dos(v0,conn2)).grid(row=12,column=0,columnspan=4)
39         Button(v0,text="Salir",width=10,bg='red',command=lambda: salir(v0)).grid(row=12,column=3,columnspan=4)
40     else:
41         ventana.destroy()
42         conn2 = opendb("temporal.db")
43         v0=Tk()
44         v0.resizable(0,0)
45         v0.title("Recolector Khan Academy")
46         total2 = total2 - total
47         Label(v0,font="Arial 10",text="La recuperación de las playlist finalizó con éxito",width=50).grid(row=0,column=0,columnspan=4)
48         Label(v0,font="Arial 10",text="Se encontraron " +str(total2)+ " videos",width=50).grid(row=1,column=0,columnspan=4)
49         Button(v0,text="Volver",width=10,command=lambda: dos(v0,conn2)).grid(row=12,column=0,columnspan=4)
50         Button(v0,text="Salir",width=10,bg='red',command=lambda: salir(v0)).grid(row=12,column=3,columnspan=4)
51     db.close()
52     v0.mainloop()
```

Ilustración 47: Módulo storevideos.

En este módulo, la primera acción se realiza en la línea ocho, donde se recuperan desde la base de datos los títulos de los videos. En la siguiente instrucción, con un ciclo for, se empieza a realizar el recorrido de los resultados de dicha consulta. En la línea once se cambia los espacio por %20, en las dos siguientes se establece la url, donde se tendrá que conectar para poder recuperar datos. Ahora bien, existen dos posibilidades: la primera es poder realizar la acción de la línea quince y dieciséis, donde se solicita al servidor los videos de la playlist con la que se está trabajando. La segunda posibilidad lo

que se muestra en la línea dieciocho, donde simplemente se imprime en pantalla la url, debido a que no se encontraba el recurso en dicho servidor.

Realizado el procedimiento anterior, nuevamente se encuentran 2 posibles caminos a seguir: el primero de ellos, desde la línea veinte hasta la veintitrés, donde se lee el dato recibido por parte del servidor y se codifica, por el hecho de que viene en un texto de tipo JSON, pero como por una playlist son varios videos, se crea un ciclo for (línea diecinueve), donde se recorre la playlist para poder llamar a la función *__storevideoinfo*. En caso contrario, el segundo camino ingresa la url a la tabla Notaplaylist, ya sea porque no se recuperó un texto de tipo JSON o porque simplemente era nulo.

En la función anterior, se hace el llamado a la función *__storevideoinfo*, para poder almacenar en la base de datos temporal la información asociada a cada uno de los recursos. Esta función se detalla en el siguiente código:

```
1  def __storevideoinfo(db, video, playlist):
2      """
3      Una cosa que hay que tener en cuenta es que las API de Khan
4      ya no retornan el dato de view es decir las veces vista
5      """
6      title = video["title"]
7      title = title.replace("'", " ")
8      id = video["readable_id"]
9      id = id.replace("'", " ")
10     date_added = video["date_added"]
11     date_added = date_added.replace("'", " ")
12     duration = video["duration"]
13     keywords= video["keywords"]
14     keywords= keywords.replace("'", " ")
15     views = 0
16     relative_url = video["relative_url"]
17     relative_url = relative_url.replace("'", " ")
18     url = video["url"]
19     url = url.replace("'", " ")
20     description = video["description"]
21     description = description.replace("'", " ")
22     """
23     Insercion de los datos a cada tabla
24     """
25     try:
26         db.execute("INSERT INTO video (id, title, date_added, keywords, relative_url, duration, views, url, description)VALUES ('"
27             + id +"','" + title+"','" + date_added+"','" + keywords+"','" + relative_url + "','" +
28             str(duration)+"','"+ str(views)+"','"+ url +"','" + description +"')")
29         db.execute("INSERT INTO playlist_video VALUES ('"+ playlist +"','"+id+"'"+")")
30         db.commit()
31     except:
32         print "Ya se encuenta el video"
33     return
```

Ilustración 48: Función __storevideoinfo.

De forma similar que en *__storeplaylistinfo*, lo que realiza *__storevideoinfo*, desde la línea seis hasta la veintiuno, es cambiar algunos caracteres como son las comillas simples por espacios de los atributos title, id, date_added, keywords, relative_url y description. Luego de esto, se tienen dos opciones: la primera de ellas es realizar los instrucciones de la línea veintiséis hasta la treinta, donde se insertan los datos en la tabla video y playlist_video, pero en la línea veintiocho, además de ingresarla se cambia el

formato de tipo texto a numérico, con la función str. La segunda opción es lo que se procede a realizar en la línea treinta y dos, donde solamente se muestra un mensaje, en el cual queda estipulado que se ya encontraba el video.

Luego de ingresar el primer video, vuelve a la función __storevideos y sigue con los siguientes videos de esa playlist. Posteriormente, se continúa con los videos de las demás playlists.

Una vez recuperados los videos asociados a las playlist, se imprime por pantalla un mensaje de éxito (puede haber otros tipo de mensajes donde se mencione que el video ya existe en la base de datos).

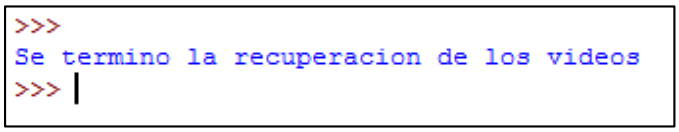
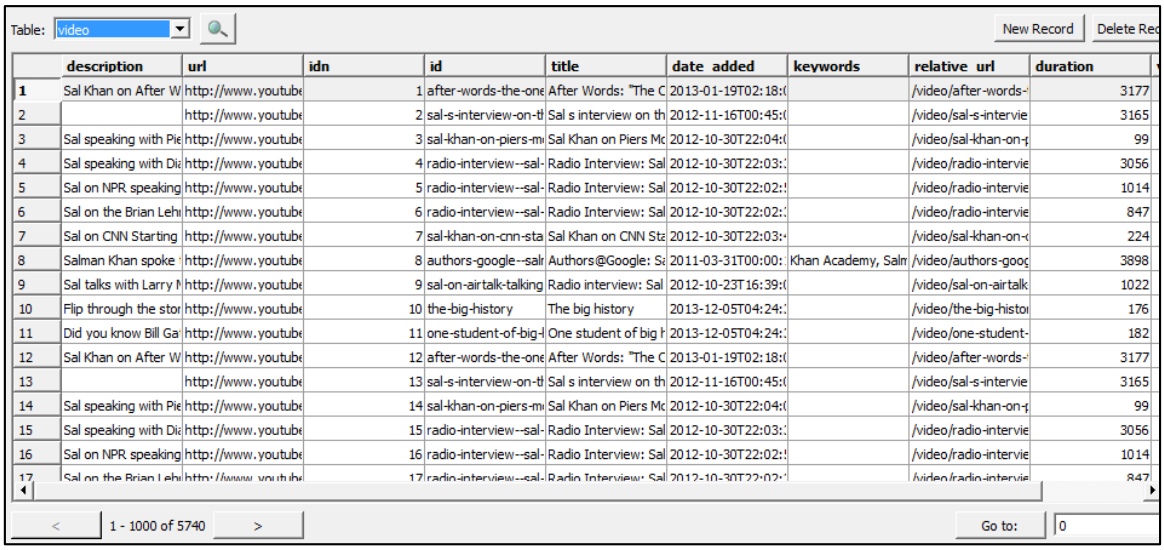


Ilustración 49: Resultado visible por pantalla (storevideos).

Volviendo a la línea cinco hasta la siete, se hace consulta de la cantidad de videos guardados hasta ese momento. Luego de haber realizado todo el proceso anterior se vuelve a realizar la misma consulta (línea veintiocho hasta la treinta), en la treinta y uno se consulta si ambas cantidades son iguales, en caso de serlo, desde la línea treinta y dos hasta la treinta y nueve, se crea una ventana, la cual tiene el mensaje de éxito, pero que no se encontraron nuevos videos, además de dos botones: uno de salida y otro para volver al menú del módulo, los cuales se definen en la línea treinta y ocho y treinta y nueve respectivamente. En caso de que sean distintos se crea una ventana (desde la línea cuarenta y uno hasta la cincuenta), la cual tiene el mensaje de éxito y la cantidad de videos nuevos encontrados, además de dos botones: uno de salida y otro para volver al menú del módulo, los cuales se definen en la línea cuarenta y nueve y cincuenta respectivamente.

Tras realizar una ejecución del módulo, fue posible encontrar 5740 videos, asociados a las 1052 playlist disponibles en el repositorio de Khan Academy. En la siguiente ilustración se observa un ejemplo de la estructura de la tabla videos después de haber recuperado la información de los objetos de aprendizaje desde el repositorio Khan Academy:



	description	url	idn	id	title	date_added	keywords	relative_url	duration
1	Sal Khan on After W	http://www.youtube	1	after-words-the-one	After Words: "The C	2013-01-19T02:18:(		/video/after-words-	3177
2		http://www.youtube	2	sal-s-interview-on-tl	Sal s interview on th	2012-11-16T00:45:(		/video/sal-s-intervie	3165
3	Sal speaking with Pie	http://www.youtube	3	sal-khan-on-piers-m	Sal Khan on Piers Mc	2012-10-30T22:04:(		/video/sal-khan-on-p	99
4	Sal speaking with Dia	http://www.youtube	4	radio-interview--sal-	Radio Interview: Sal	2012-10-30T22:03::		/video/radio-intervie	3056
5	Sal on NPR speaking	http://www.youtube	5	radio-interview--sal-	Radio Interview: Sal	2012-10-30T22:02:5		/video/radio-intervie	1014
6	Sal on the Brian Lehi	http://www.youtube	6	radio-interview--sal-	Radio Interview: Sal	2012-10-30T22:02::		/video/radio-intervie	847
7	Sal on CNN Starting	http://www.youtube	7	sal-khan-on-cnn-sta	Sal Khan on CNN Sta	2012-10-30T22:03:•		/video/sal-khan-on-c	224
8	Salman Khan spoke	http://www.youtube	8	authors-google--salr	Authors@Google: Sa	2011-03-31T00:00:	Khan Academy, Salm	/video/authors-goog	3898
9	Sal talks with Larry M	http://www.youtube	9	sal-on-airtalk-talking	Radio interview: Sal	2012-10-23T16:39:(		/video/sal-on-airtalk	1022
10	Flip through the stor	http://www.youtube	10	the-big-history	The big history	2013-12-05T04:24::		/video/the-big-histor	176
11	Did you know Bill Ga	http://www.youtube	11	one-student-of-big-h	One student of big h	2013-12-05T04:24::		/video/one-student-	182
12	Sal Khan on After W	http://www.youtube	12	after-words-the-one	After Words: "The C	2013-01-19T02:18:(		/video/after-words-	3177
13		http://www.youtube	13	sal-s-interview-on-tl	Sal s interview on th	2012-11-16T00:45:(		/video/sal-s-intervie	3165
14	Sal speaking with Pie	http://www.youtube	14	sal-khan-on-piers-m	Sal Khan on Piers Mc	2012-10-30T22:04:(		/video/sal-khan-on-p	99
15	Sal speaking with Dia	http://www.youtube	15	radio-interview--sal-	Radio Interview: Sal	2012-10-30T22:03::		/video/radio-intervie	3056
16	Sal on NPR speaking	http://www.youtube	16	radio-interview--sal-	Radio Interview: Sal	2012-10-30T22:02:5		/video/radio-intervie	1014
17	Sal on the Brian Lehi	http://www.youtube	17	radio-interview--sal-	Radio Interview: Sal	2012-10-30T22:02::		/video/radio-intervie	847

Ilustración 50: Tabla video.

Además, es posible ver en la tabla Notaplylist que existen datos asociados a playlists en las que no se pudieron recuperar videos, ya sea porque no retornaba un archivo JSON o porque no tenía ningún video asociado. La cantidad de playlist a las cuales no fue posible acceder a recopilar información de videos tras la ejecución del módulo asciende a 367. Esto es posible de apreciar en la siguiente ilustración:

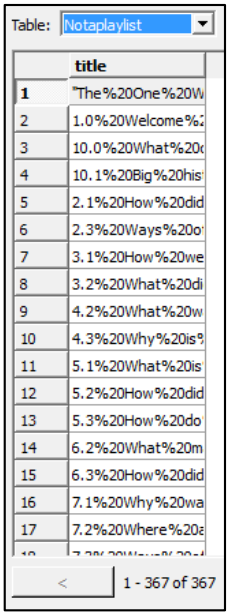


Ilustración 51: Tabla Notaplaylist.

5.8.1.5 Módulo scrap_videos

Un usuario, al ejecutar la operación “Recuperar comentarios” en el menú de inicio, el recolector llama a la función scrap_videos, la cual se detalla a continuación:

```
def scrap_videos(db,ventana):
    cuenta = db.execute("SELECT Count(*) FROM comments")
    for veri in cuenta:
        total = veri[0]
    res = db.execute("SELECT id, relative_url FROM video")
    browser = webdriver.Firefox() # Get local session of firefox
    for row in res:
        ___scrap_video(db, "http://"+KHANACAD_URI+row[1], browser, row[0])
    print "Se terminó la recuperación de los comentarios"
    cuenta = db.execute("SELECT Count(*) FROM comments")
    for veri in cuenta:
        total2 = veri[0]
    if total == total2:
        ventana.destroy()
        conn2 = opendb("temporal.db")
        v0=Tk()
        v0.resizable(0,0)
        v0.title("Recolector Khan Academy")
        Label(v0,font="Arial 10",text="Proceso finalizado, no se encontraron nuevos comentarios",width=50).grid(row=0,column=0,columnspan=4)
        Button(v0,text="Volver",width=10,command=lambda: tres(v0,conn2)).grid(row=12,column=0,columnspan=4)
        Button(v0,text="Salir",width=10,bg='red',command=lambda: salir(v0)).grid(row=12,column=3,columnspan=4)
    else:
        ventana.destroy()
        conn2 = opendb("temporal.db")
        v0=Tk()
        v0.resizable(0,0)
        v0.title("Recolector Khan Academy")
        total2 = total2 - total
        Label(v0,font="Arial 10",text="La recuperación de los comentarios finalizó con éxito",width=50).grid(row=0,column=0,columnspan=4)
        Label(v0,font="Arial 10",text="Se encontraron " +str(total2)+ " videos",width=50).grid(row=1,column=0,columnspan=4)
        Button(v0,text="Volver",width=10,command=lambda: tres(v0,conn2)).grid(row=12,column=0,columnspan=4)
        Button(v0,text="Salir",width=10,bg='red',command=lambda: salir(v0)).grid(row=12,column=3,columnspan=4)
    db.close()
    browser.close()
```

Ilustración 52: Módulo scrap_videos.

Este módulo busca recuperar los comentarios de los videos. Para ello, en la línea cinco, selecciona el id y la url relativa de todos los videos, a través de una consulta en SQL a la base de datos. Luego de esto, en la línea seis, se abre un navegador, en este caso Firefox (se necesita que esté instalado en el equipo donde se ejecute el recolector). Como se necesita recuperar los comentarios de todos los videos existentes, en la sesta instrucción, por medio de un ciclo for, se empiezan a recorrer todos los videos, para luego llamar a la función *__scrap_video*.

```
1  def ___scrap_video(db, uri, browser, idvideo):
2      try:
3          browser.get(uri)
4          tipscommentslink = browser.find_element_by_xpath("//a[contains(text(),'Tips & Thanks')]")
5          tipscommentslink.click()
6          tabs = browser.find_elements_by_class_name("discussion-list-content")
7          cadena = ''
8          for elem in tabs:
9              cadena = cadena + elem.text
10         inicio = 0
11         ban = 0
12         while (inicio < len(cadena)) and (ban != 1):
13             pos = cadena.find (" Comment", inicio)
14             if pos == -1:
15                 ban = 1
16             else:
17                 ban2=0
18                 while ban2 == 0:
19                     if cadena[pos-1] != '\n':
20                         pos = pos -1
21                     else:
22                         ban2 = 1
23                 pos = pos - 1
24                 comentario=cadena[inicio:pos]
25                 inicio = pos +1
26                 pos = cadena.find("V", inicio)
27                 likes = cadena[inicio:pos]
28                 inicio = cadena.find("by", pos)
29                 inicio = inicio + 3
30                 pos = cadena.find("\n", inicio)
31                 if pos == -1:
32                     usuario = cadena[inicio:]
33                     ban = 0
34
35                 else:
36                     usuario = cadena[inicio:pos]
37                     inicio = pos + 1
38                 __storecommentinfo(db, idvideo, comentario, likes, usuario)
39     except:
40         print 'No se pudo ir a esta pagina'
41         print uri
42     return
```

Ilustración 53: Código de la Función __scrap_video.

Lo que realiza __scrap_video es recibir como parámetro un navegador, el cual fue abierto de forma automática por la función scrap_video (ilustración 56). Se tienen dos caminos a seguir: el primero comienza desde la línea tres en la cual se carga la URL del video en la barra de direcciones del navegador conformada por URI + url_relativa. Luego, por medio de las instrucciones de la línea cuatro y cinco, el Recolector hace clic en el apartado “Tips & Thanks” (ubicado en el panel de preguntas, tips y agradecimientos), donde se alojan los comentarios realizados al contenido del video (ilustración 57 y 58).

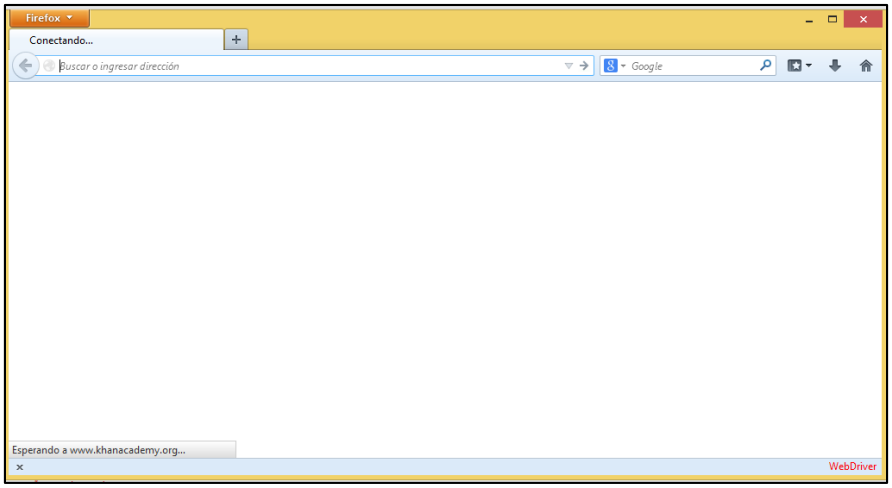


Ilustración 54: Se abre el navegador automáticamente (nótese Web Driver parte inferior derecha).

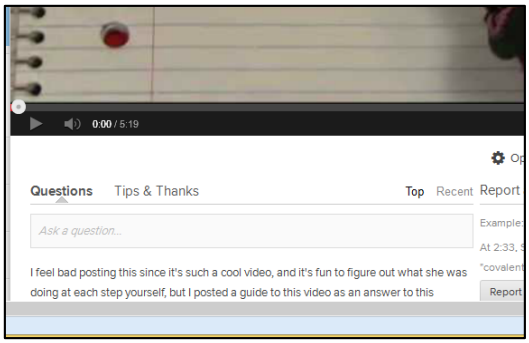


Ilustración 56: Primera visualización del video.

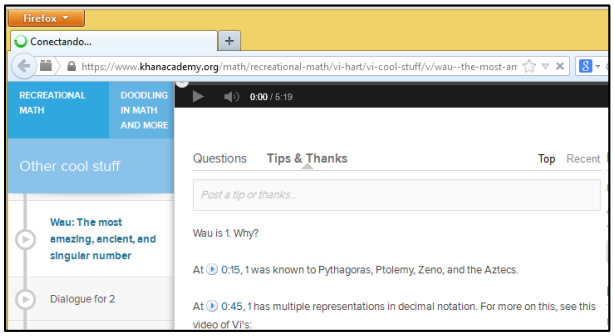


Ilustración 55: El Recolector le hace clic a Tips & Thanks.

En las próximas líneas, desde la siete hasta la treinta y ocho, se identifica y divide el texto recolectado desde la página del video, es decir, reconoce el usuario, comentario y los votos. Luego, por cada comentario encontrado se llama a la función *__storecommentinfo*.

```
1  def __storecommentinfo(db, idvideo, comentario, likes, usuario):
2      comentario = comentario.replace("'", " ")
3      usuario= usuario.replace("'", " ")
4      res = db.execute("SELECT count(*) FROM comments Where videoid='"+idvideo+"' and usernick = '"
5                       +usuario+"' and content = '"+comentario+"'")
6      if (res == 0) or (res is None):
7          db.execute("UPDATE comments Set votes = '" + str(likes)+"' Where videoid='"+idvideo+"' and usernick = '"
8              +usuario+"' and content = '"+comentario+"'")
9      else:
10         db.execute("INSERT INTO comments (videoid, usernick, votes, content) VALUES ('"+idvideo+"','"+usuario+"','"
11             + str(likes)+"','"+comentario+"')")
12     db.commit()
13     return
```

Ilustración 57: Código de la Función __storecommentinfo.

En la función *__storecommentinfo*, en la línea dos y tres, se reemplaza las comillas simples por espacios en blanco, esto para evitar problemas al momento de almacenar los datos en la base de datos. Luego, en la línea cuatro, se hace una consulta a la base de datos, en donde se trata de ver si dicho comentario está almacenado: Si no se encuentra almacenado, lo ingrese con la instrucción de la línea diez y once, y en caso de ya estar almacenado, solo se actualizará el registro, como se puede ver en la línea siete y ocho.

Retomando la función _scrap_video, el segundo camino está constituido por las línea cuarenta y cuarenta y uno, se imprime un mensaje de que no se pudo ir a la página del video, y además se imprime la url del mismo. Esto es necesario ya que en el repositorio Khan Academy pueden haber sacado el recurso del servidor o por fallo de la conexión a internet.

Luego de haber terminado de recuperar todos los comentarios posibles, el módulo scrap_videos en la línea nueve realiza la impresión de un mensaje (ilustración 60).

```
>>>
Se termino la recuperacion de los comentarios
>>>
```

Ilustración 58: Resultado visible por pantalla (scrap_videos).

Volviendo a la línea dos hasta la cuatro, se consulta por la cantidad de comentarios guardados antes de empezar el proceso, luego desde la línea diez hasta la doce se vuelve a consultar, pero ya con el proceso realizado, para finalizar con este módulo, en la línea trece se consulta si ambas cantidades son iguales, en caso de serlo se crea una ventana (desde la línea catorce hasta la veintiuno), la cual tiene el mensaje de éxito, pero que no se encontraron nuevos comentarios, además, cuenta con botones de salida y para volver al menú, los cuales se definen en la línea veinte y veintiuno respectivamente. En caso de que ambos números sean distintos se crea una ventana (desde la línea veintitrés hasta la treinta y dos), la cual tiene el mensaje de éxito, además de la cantidad de comentarios nuevos agregados a la base de datos, además cuenta con botones de salida y para volver al menú, los cuales se definen en la línea treinta y uno y treinta y dos respectivamente

Revisando los datos obtenidos, fue posible recolectar 32770 comentarios, asociados a los 5740 videos recopilados anteriormente. En la siguiente ilustración se observa un ejemplo de la estructura de la tabla comentarios después de haberlos

recuperado recorriendo cada uno de los videos, por medio del navegador web, desde el repositorio Khan Academy:

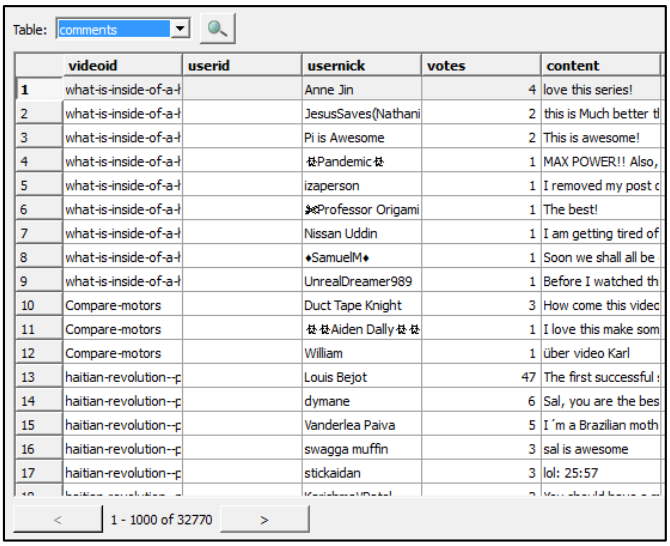


Table: comments

	videoid	userid	usernick	votes	content
1	what-is-inside-of-a-l		Anne Jin	4	love this series!
2	what-is-inside-of-a-l		JesusSaves(Nathani	2	this is Much better tl
3	what-is-inside-of-a-l		Pi is Awesome	2	This is awesome!
4	what-is-inside-of-a-l		☢Pandemic☢	1	MAX POWER!! Also,
5	what-is-inside-of-a-l		izaperson	1	I removed my post c
6	what-is-inside-of-a-l		✄Professor Origami	1	The best!
7	what-is-inside-of-a-l		Nissan Uddin	1	I am getting tired of
8	what-is-inside-of-a-l		♦SamuelM♦	1	Soon we shall all be
9	what-is-inside-of-a-l		UnrealDreamer989	1	Before I watched th
10	Compare-motors		Duct Tape Knight	3	How come this videc
11	Compare-motors		☢☢Aiden Dally☢☢	1	I love this make som
12	Compare-motors		William	1	über video Karl
13	haitian-revolution--p		Louis Bejot	47	The first successful s
14	haitian-revolution--p		dymane	6	Sal, you are the bes
15	haitian-revolution--p		Vanderlea Paiva	5	I´m a Brazilian moth
16	haitian-revolution--p		swagga muffin	3	sal is awesome
17	haitian-revolution--p		stickaidan	3	lol: 25:57

< 1 - 1000 of 32770 >

Ilustración 59: Tabla comments.

5.8.1.6 Módulo createtxt

Un usuario, al ejecutar la operación “Crear archivo TXT” en el menú de archivos, el recolector llama a la función createtxt. Con este módulo, es posible generar un archivo de texto plano que permita visualizar de otra forma la información del repositorio local. Este módulo es opcional y no afecta al estado final del repositorio.

```
1   def createtxt(db,filename,ventana):
2       """
3       creacion de un txt con toda la informacion recopilada
4       """
5       res = db.execute("SELECT id, title, date_added, keywords, relative_url, duration, description, url FROM video")
6       f = open(filename, 'w')
7       for row in res:
8           f.write('Video\n')
9           idn = row[0]
10          f.write('   id:'+idn.encode('utf-8')+'\n')
11          titulo = row[1]
12          f.write('   Titulo:'+titulo.encode('utf-8')+'\n')
13          fecha = row[2]
14          f.write('   Fecha agregado:'+fecha.encode('utf-8')+'\n')
15          claves = row[3]
16          f.write('   Palabras claves:'+claves.encode('utf-8')+'\n')
17          url = row[4]
18          f.write('   URL:'+url.encode('utf-8')+'\n')
19          duracion = row[5]
20          f.write('   Duracion:'+str(duracion)+'\n')
21          description = row[6]
22          f.write('   Descripcion:'+description.encode('utf-8')+'\n')
23          url_youtube = row[7]
24          f.write('   URL_Youtube:'+url_youtube.encode('utf-8')+'\n')
25          cont = db.execute("SELECT usernick, votes, content FROM comments where videoid='"+idn+"'")
26          for lin in cont:
27              f.write('       Comentario:')
28              usuario=lin[0]
29              f.write('           Nick del usuario:'+usuario.encode('utf-8')+'\n')
30              votos=lin[1]
31              f.write('           Votos:'+str(votos)+'\n')
32              contenido=lin[2]
33              f.write('           Contenido:'+contenido.encode('utf-8')+'\n')
34              f.write('\n\n')
35      ventana.destroy()
36      db.close()
37      conn = opendb("temporal.db")
38      v0=Tk()
39      v0.resizable(0,0)
40      v0.title("Recolector Khan Academy")
41      Label(v0,font="Arial 15 bold",text="Se creó con éxito el archivo txt",width=50).grid(row=0,column=0,columnspan=4)
42      Label(v0,text="\n",width=50).grid(row=1,column=0,columnspan=4)
43      Button(v0,text="Ver",width=10,command=lambda: siete(v0,conn)).grid(row=6,column=1,sticky=W)
44      #Button(v0,text="Continuar",width=10,command=lambda: cuatro(v0,conn)).grid(row=6,column=3,sticky=W)
45      Button(v0,text="Volver",width=10,command=lambda: cuatro(v0,conn)).grid(row=7,column=1,sticky=W)
46      Button(v0,text="Salir",width=10,bg='red',command=lambda: salir(v0)).grid(row=7,column=3,sticky=W)
47      v0.mainloop()
```

Ilustración 60: Módulo createtxt

Lo que realiza este módulo, en la línea cinco, es recuperar desde la base de datos temporal toda la información del video. Luego, abre un archivo de texto y lo deja disponible para la escritura de los datos (en caso de que no esté creado lo crea, y de estar creado lo sobrescribe). Como se tienen muchos videos, la escritura se realiza a través de un ciclo for, donde se empiezan a recorrer los resultados traídos desde la base de datos, pero se le colocaron cabeceras para distinguir cada dato (las cabeceras están en las líneas ocho, doce, catorce, dieciséis, dieciocho, veinte, veintidós y veinticuatro; las cabeceras del comentario están en la línea veintinueve, treinta y uno y en la treinta y tres), luego de que se escribe los datos relevantes del video, en la línea veinticinco se consulta a la base de datos por los comentarios del video que se está escribiendo. Luego, se recorren los resultados para escribirlos en el texto de igual forma que los datos relevantes del video. La ejecución de este módulo no es perceptible en pantalla por el usuario y el término del proceso se identifica con un mensaje por pantalla (ilustración 60), la cual se implementa con la instrucción de la línea treinta y cinco.

```
>>>
Se termino la escritura del txt
>>>
```

Ilustración 61: Resultado visible por pantalla (createtxt).

Para finalizar, desde la línea treinta y cinto hasta la cuarenta y seis, se crea la ventana del entorno gráfico, la cual tiene el mensaje de éxito, además de tres botones: uno de salida, otro para volver al menú y por ultimo un botón para poder ver el archivo creado, los cuales se definen en la línea cuarenta y tres, cuarenta y cinco y cuarenta y seis respectivamente.

Al término de este proceso, es posible verificar la información en el archivo creado en la carpeta del programa. La estructura del documento está descrita de la siguiente forma:

```
Video
   id:
   Titulo:
   Fecha agregado:
   Palabras claves:
   URL:
   Duracion:
   Descripcion:
   URL_Youtube:
       Comentario:          Nick del usuario:
           Votos:
           Contenido:
```

Ilustración 62: Formato del archivo txt.

5.8.1.7 Módulo LOM

Representación XML: Un usuario, al ejecutar la operación “Crear representación del repositorio” en el menú de inicio, el recolector llama a la función *LOM*. Este módulo, al igual que el módulo createtxt, es posible ejecutar al momento de tener toda la información de videos y comentarios recopilada. Este proceso es realizado con todos los videos almacenados en la base de datos temporal y sus correspondientes comentarios.

Lo que realiza este módulo es seleccionar toda la información de los videos para luego recorrerlos uno por uno. A continuación, los datos de la tupla los codifica a Unicode, puesto que para poder generar un archivo XML con lenguaje Python, solo permite este formato. Además, se agrega a la Url relativa el siguiente texto al comienzo:

http://www.khanacademy.org. Lo siguiente que hace internamente es llenar cada uno de los campos de la estructura XML bajo el estándar LOM, o sea, la categoría inferior, de acuerdo a los datos contenidos en la base de datos temporal. Se agrega otro elemento en la categoría Annotations: Votes y en la categoría Meta-Metadata: Identifier_Youtube. Lo anterior, se realiza solo para un video y su respectiva información y comentarios. Este proceso se debe realizar para todos los videos existentes, para que una vez finalizado, se asocien a la categoría superior, la cual será escrita en el archivo final.

Finalizado este proceso, para todos los videos y comentarios, es posible verificar la información en el archivo XML del repositorio. Para ello, se debe abrir dicho archivo, ya sea en un navegador, un bock de notas o en algún visualizador de XML, etc. Este archivo tendrá una estructura como la siguiente.

Representación Relacional: En conjunto con la escritura de la representación en XML, al momento de estar llenando las categorías inferiores de LOM del archivo XML, se agregan los datos correspondientes en las tablas de LOM. Recordar que la base de datos, desde un principio, está compuesta por seis tablas para almacenar datos temporales y diecisiete tablas del estándar LOM.

Una vez finalizado el proceso de llenado de datos en las tablas de LOM, con toda la información correspondiente a los videos y sus respectivos comentarios, se cierra la conexión a la base de datos para realizar una copia completa del archivo de base de datos, la que llevará el nombre “local”. Se procede a abrir la conexión a esta nueva base de datos y se procede a eliminar las 6 tablas de datos temporales, quedando en esta base de datos solo las tablas que cumplen con el estándar LOM. Este proceso es necesario ya que Python no permite tener 2 conexiones a bases de datos en una misma ejecución, si es que se quisiera tener ambas bases de datos de forma independiente.

La ejecución del módulo completo de LOM no es perceptible en pantalla por el usuario y el término del proceso se identifica con un mensaje por pantalla (ilustración 62).

```
>>>
Se termino la escritura del XML con formato LOM
>>>
```

Ilustración 63: Resultado visible por pantalla (createtxt).

5.8.2 Caso de estudio específico

A continuación, se presenta un caso de estudio específico, donde se muestra la recuperación solo de una playlist y sus respectivos videos y comentarios.

Para este caso de estudio, se asume que las estructuras necesarias no contienen datos, y a la vez la conexión se encuentra realizada.

El primer paso es recuperar la playlist a través del recolector. Realizado este proceso según 5.7.1.3, se aprecia en la base de datos temporal la playlist con título: **"The One World Schoolhouse" book** (ilustración 66)

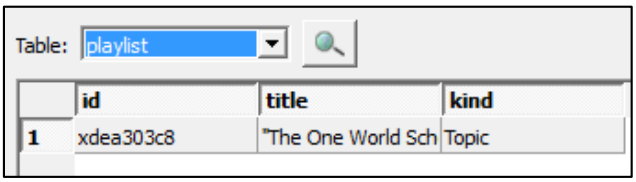


Table: playlist

	id	title	kind
1	xdea303c8	"The One World Sch	Topic

Ilustración 64: Playlist recuperada.

El siguiente paso es rescatar los videos asociados a la playlist ya recuperada. Para ello se ejecuta el módulo de recuperación de videos storevideos descrito en 5.7.1.4. Tras finalizar este proceso, se aprecia la tabla video con los siguientes datos:

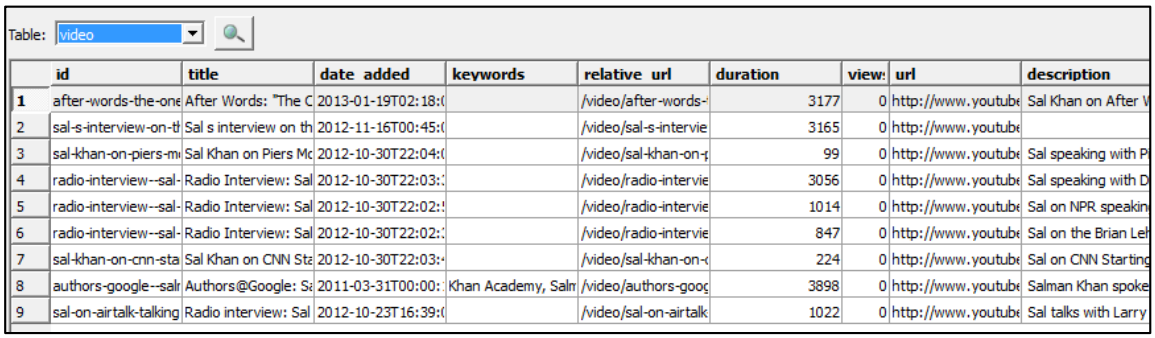


Table: video

	id	title	date_added	keywords	relative_url	duration	view:	url	description
1	after-words-the-one	After Words: "The C	2013-01-19T02:18:(		/video/after-words-	3177	0	http://www.youtube	Sal Khan on After V
2	sal-s-interview-on-tl	Sal s interview on th	2012-11-16T00:45:(		/video/sal-s-intervie	3165	0	http://www.youtube	
3	sal-khan-on-piers-m	Sal Khan on Piers Mc	2012-10-30T22:04:(		/video/sal-khan-on-p	99	0	http://www.youtube	Sal speaking with P
4	radio-interview--sal-	Radio Interview: Sal	2012-10-30T22:03::		/video/radio-intervie	3056	0	http://www.youtube	Sal speaking with D
5	radio-interview--sal-	Radio Interview: Sal	2012-10-30T22:02::		/video/radio-intervie	1014	0	http://www.youtube	Sal on NPR speakin
6	radio-interview--sal-	Radio Interview: Sal	2012-10-30T22:02::		/video/radio-intervie	847	0	http://www.youtube	Sal on the Brian Lel
7	sal-khan-on-cnn-sta	Sal Khan on CNN Sta	2012-10-30T22:03:-		/video/sal-khan-on-(	224	0	http://www.youtube	Sal on CNN Starting
8	authors-google--salr	Authors@Google: Sa	2011-03-31T00:00:	Khan Academy, Salm	/video/authors-goog	3898	0	http://www.youtube	Salman Khan spoke
9	sal-on-airtalk-talking	Radio interview: Sal	2012-10-23T16:39:(		/video/sal-on-airtalk	1022	0	http://www.youtube	Sal talks with Larry

Ilustración 65: Videos de la playlist recuperada.

Como se aprecia en la ilustración 67, la tabla video contiene 9 videos. Esto significa que han sido encontrados 9 videos relacionados con el título de la playlist rescatada.

A continuación, se deben rescatar los comentarios asociados a cada uno de los videos de la playlist. Para ello, se ejecuta el módulo scrap_video descrito en 5.7.1.5. En este caso de estudio, solo se muestran los comentarios asociados al primer video de la lista. Esto, debido a la cantidad de comentarios que se pueden encontrar en los videos hace muy extenso el documento.

Table: comments

	videoid	userid	usernick	votes	content	date
1	after-words-the-one		Greg	7	Sal is such an interes	
2	after-words-the-one		Kenneth Epley	5	At 12:20 and all thro	
3	after-words-the-one		Taylor Swift	5	I love Khan Academ	
4	after-words-the-one		undefined.brilliance	3	Your view s are truly	
5	after-words-the-one		Ryan Lynn Wikert	2	Thank you Khan for	
6	after-words-the-one		Nissan Uddin	2	Wow. I read the pre	
7	after-words-the-one		Petrie (Peter S. Asia	2	Without a doubt Sal	
8	after-words-the-one		福龍丸 (Lucky Drago	2	At around 33:00: I v	
9	after-words-the-one		Mark Greco	2	Sal I love you! I ve a	
10	after-words-the-one		Nerd Herd	1	can you make this sl	
11	after-words-the-one		Unmil Parghi	1	Mr. Sal Khan is the b	
12	after-words-the-one		Shmuel Chaim	1	thanks	
13	after-words-the-one		Unmil Parghi	1	Mr. Sal Khan is the b	
14	after-words-the-one		Uday Parmar	0	Sal must be famous.	
15	after-words-the-one		chenesther011	0	Post a tip or thanks.	

Ilustración 66: Comentarios asociados al video 1.

Para corroborar que los comentarios rescatados son efectivamente los asociados al primer video, se chequea en la página del video (visible a todo usuario). Recordar que la dirección está compuesta por la URI (definida como http://khanacademy.com) más la url relativa asociada al video. Por lo tanto, para este caso, la dirección a visitar es http://khanacademy.com/video/after-words-the-one-world-schoolhouse-education-reimagined.

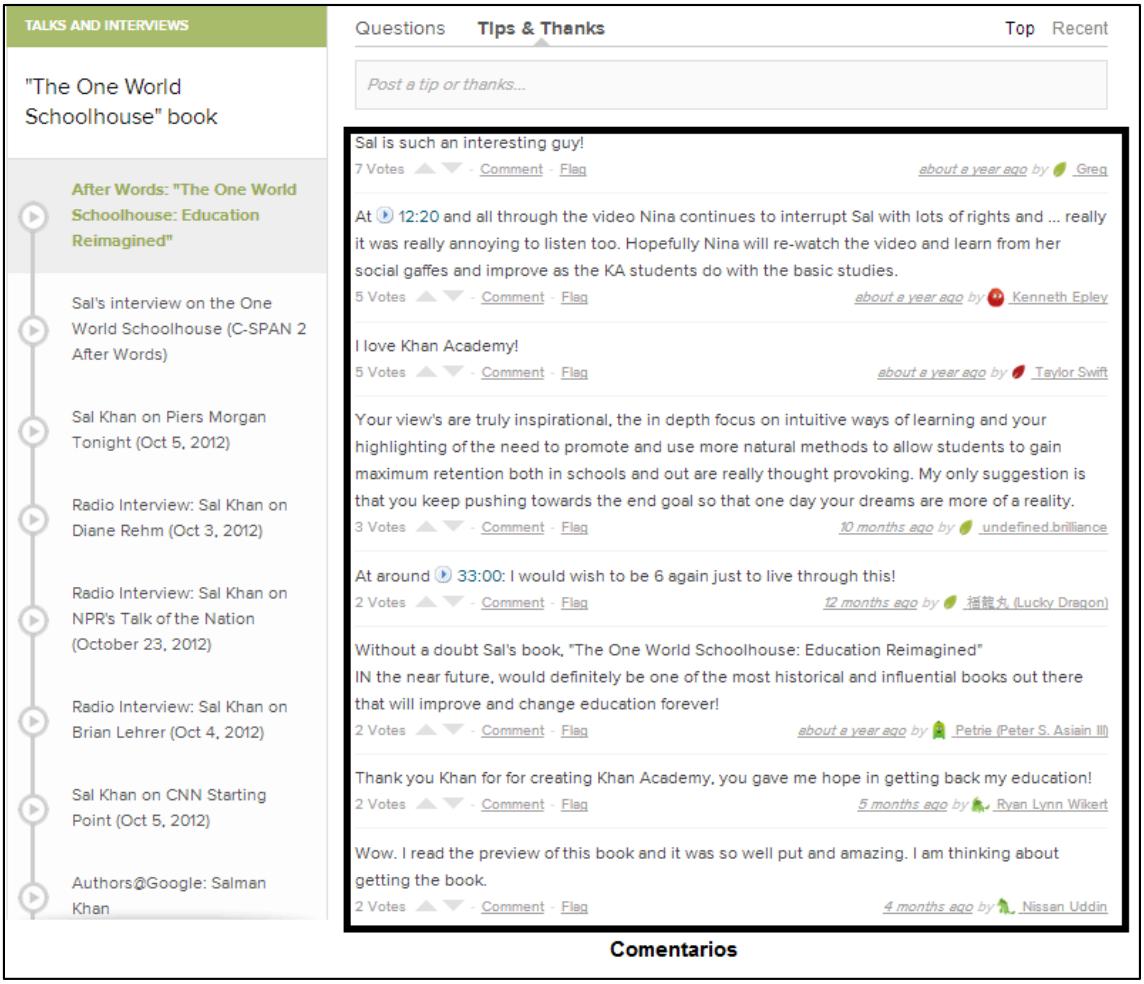


Ilustración 67: Comentarios del video en el sitio Khan Academy.

Al visitar dicha url, se puede ver el video y verificar que los comentarios existentes en la base de datos temporal son los que realmente están en el repositorio Khan Academy, los cuales están ordenados de la misma manera que en el video, es decir por cantidad de votos.

Una vez recuperados los comentarios, se crea la representación del repositorio, es decir, se crea un archivo XML (representación no estructurada) y la base de datos relacional (representación estructurada), ambas bajo el estándar de LOM. Para ello, se ejecuta el módulo LOM descrito en 5.7.1.7.

Detalle de la representación no estructurada: La representación no estructurada consta de un archivo en formato XML, donde las etiquetas que representan a los recursos recuperados están estructuradas según el estándar LOM. Para este caso de estudio, la información correspondiente al video se encuentra asociada en las diferentes categorías. Habrán varias etiquetas que estarán vacías, ya que los metadatos asociados en el repositorio Khan Academy son muy pocos en comparación a la cantidad de metadatos que se representan con LOM. El ejemplo correspondiente al caso de estudio se deja a disposición en el anexo 2, dada la extensión de visualización del archivo XML.

Detalle de la representación estructurada: A continuación, en la representación de la base de datos, se podrán encontrar las diecisiete tablas del modelo LOM. Para este caso se mostrara principalmente dos tablas “General” y “Annotation”, las cuales son correspondiente a los comentarios y lo principal del video.

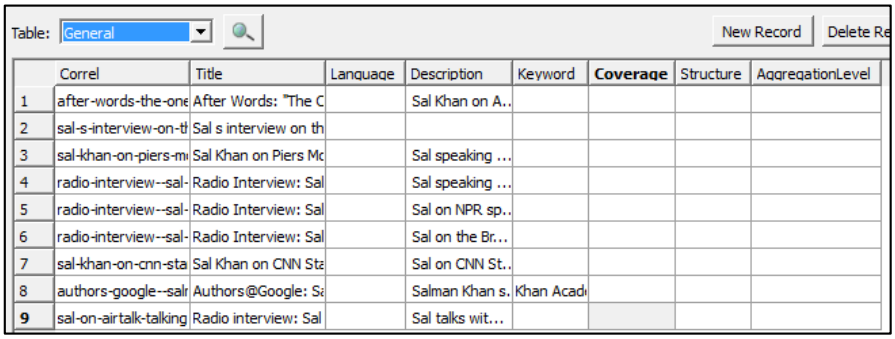


Table: General — New Record — Delete Re

	Correl	Title	Language	Description	Keyword	Coverage	Structure	AggregationLevel
1	after-words-the-one	After Words: "The C		Sal Khan on A...				
2	sal-s-interview-on-th	Sal s interview on th						
3	sal-khan-on-piers-m	Sal Khan on Piers Mc		Sal speaking ...				
4	radio-interview--sal-	Radio Interview: Sal		Sal speaking ...				
5	radio-interview--sal-	Radio Interview: Sal		Sal on NPR sp...				
6	radio-interview--sal-	Radio Interview: Sal		Sal on the Br...				
7	sal-khan-on-cnn-sta	Sal Khan on CNN Sta		Sal on CNN St...				
8	authors-google--salr	Authors@Google: Sa		Salman Khan s...	Khan Acad			
9	sal-on-airtalk-talking	Radio interview: Sal		Sal talks wit...				

Ilustración 68: Tabla general para los 9 videos rescatados.

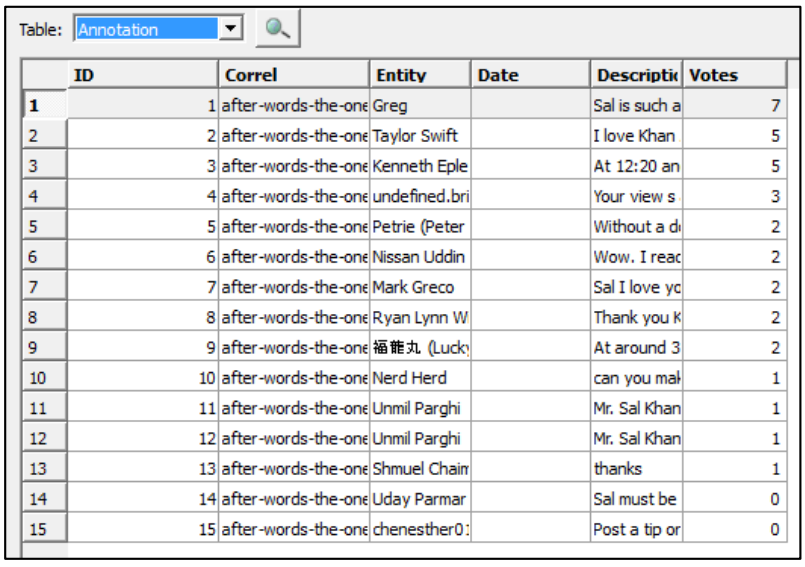


Table: Annotation

	ID	Correl	Entity	Date	Descripti	Votes
1	1	after-words-the-one	Greg		Sal is such a	7
2	2	after-words-the-one	Taylor Swift		I love Khan	5
3	3	after-words-the-one	Kenneth Eple		At 12:20 an	5
4	4	after-words-the-one	undefined.bri		Your view s	3
5	5	after-words-the-one	Petrie (Peter		Without a d	2
6	6	after-words-the-one	Nissan Uddin		Wow. I reac	2
7	7	after-words-the-one	Mark Greco		Sal I love yc	2
8	8	after-words-the-one	Ryan Lynn W		Thank you K	2
9	9	after-words-the-one	福龍丸 (Lucky		At around 3	2
10	10	after-words-the-one	Nerd Herd		can you mak	1
11	11	after-words-the-one	Unmil Parghi		Mr. Sal Khan	1
12	12	after-words-the-one	Unmil Parghi		Mr. Sal Khan	1
13	13	after-words-the-one	Shmuel Chain		thanks	1
14	14	after-words-the-one	Uday Parmar		Sal must be	0
15	15	after-words-the-one	chenesther0:		Post a tip or	0

Ilustración 69: Tabla Annotation - Muestra los comentarios del primer video.

Con este caso de estudio, es posible mostrar que los datos correspondientes a un video son almacenados en ambas representaciones.

Cabe mencionar que este proceso puede realizarse más de una vez, por lo que el recolector verifica que los datos extraídos no se repitan en ambas representaciones, por ende actualiza solamente con datos nuevos.

6 Proyectos futuros

Tras cumplir con los objetivos del proyecto, se podrá facilitar diversos proyectos futuros. Entre las cosas que se le abren las puertas para realizar, son primeramente.

6.1 Repositorio local de fines académicos

A partir del repositorio creado en este proyecto, es posible trabajar en proyectos que desarrollen las capas superiores de la arquitectura propuesta. Estos desarrollos pueden estar enfocados en habilitar un sistema de búsquedas en el repositorio con el despliegue correspondiente de la información solicitada. También puede habilitarse una plataforma de carga de contenidos educativos, con sus correspondientes metadatos y comentarios, extraídos desde cualquier otro repositorio que pueda existir o habilitarse en un futuro, para así tener una base de datos más amplia, dependiendo el dominio de trabajo que se quiera lograr.

6.2 Tratamiento de comentarios

Ya que a estas alturas se cuenta con un repositorio de objetos de aprendizaje que contiene metadatos y comentarios asociados, se cumple con el objetivo de facilitar la realización de procesos de análisis de información contenida en objetos de aprendizaje. Estos procesos de análisis de información pueden realizarse aplicando herramientas ya existentes o desarrollando nuevas aplicaciones.

Algunos de los procesos de análisis de información que pueden aplicarse gracias al proyecto desarrollado son los siguientes:

6.2.1 Análisis de Sentimiento

Se conoce también como “opinion mining”, se refiere al uso de procesamiento del lenguaje natural, análisis de textos y la lingüística computacional para identificar y extraer la información subjetiva de los materiales de base.

El Análisis de Sentimiento tiene como objetivo determinar la actitud de un orador o un escritor con respecto a algún tema o la polaridad del contexto general de un

documento. La actitud puede ser a su juicio o evaluación, el estado afectivo, o la comunicación emocional previsto.

El Análisis de Sentimiento es una de las áreas de investigación que ha recibido bastante atención por parte de los investigadores en la informática. Más de 7000 artículos se han escrito sobre el tema. Cientos de startups están desarrollando soluciones de Análisis de Sentimiento, y de los principales paquetes estadísticos, como SAS y SPSS, incluyen módulos de análisis de emociones específicas. Hay una enorme explosión hoy de “Sentiment” disponible en las redes sociales como Twitter, Facebook, foros, blogs, y foros de usuarios. Estos fragmentos de texto son una mina de oro para las empresas e individuos que quieren controlar su reputación y obtener retroalimentación oportuna sobre sus productos y acciones. El análisis de sentimientos ofrece a estas organizaciones la posibilidad de supervisar los diferentes sitios de medios sociales en tiempo real y actuar en consecuencia.

Es común para clasificar las penas en dos clases principales con respecto a la subjetividad: oraciones objetivas que contienen información sobre los hechos y frases subjetivas que contienen opiniones explícitas, creencias y puntos de vista sobre las entidades específicas. Aquí, en su mayoría se centran en el análisis de frases subjetivas.

6.2.2 Minería de Datos

Según Gregory Piatetsky-Shapiro, dijo que Minería de Datos era “Descubrimiento de conocimiento en bases de datos”. La Minería de Datos (minería de datos), es el conjunto de técnicas y tecnologías que permiten explorar grandes bases de datos, de manera automática o semiautomática, con el objetivo de encontrar patrones repetitivos, tendencias o reglas que expliquen el comportamiento de los datos en un determinado contexto.

Básicamente, la Minería de Datos surge para intentar ayudar a comprender el contenido de un repositorio de datos. Con este fin, hace uso de prácticas estadísticas y, en algunos casos, de algoritmos de búsqueda próximos a la Inteligencia Artificial y a las redes neuronales.

De forma general, los datos son la materia prima bruta. En el momento que el usuario les atribuye algún significado especial pasan a convertirse en información. Cuando los especialistas elaboran o encuentran un modelo, haciendo que la interpretación que surge entre la información y ese modelo represente un valor agregado, entonces nos referimos al conocimiento.

Las técnicas más comúnmente usadas en Minería de Datos son:

- Redes neuronales artificiales: Modelos predecibles no-lineales que aprenden a través del entrenamiento y semejan la estructura de una red neuronal biológica.
- Arboles de decisión: Estructuras de forma de árbol que representan conjuntos de decisiones. Estas decisiones generan reglas para la clasificación de un conjunto de datos. Métodos específicos de árboles de decisión incluyen Arboles de Clasificación y Regresión (CART: Classification And Regression Tree) y Detección de Interacción Automática de Chi Cuadrado (CHAI: Chi Square Automatic Interaction Detection).
- Algoritmos genéticos: Técnicas de optimización que usan procesos tales como combinaciones genéticas, mutaciones y selección natural en un diseño basado en los conceptos de evolución.
- Método del vecino más cercano: Una técnica que clasifica cada registro en un conjunto de datos basado en una combinación de las clases del/de los k registro (s) más similar/es a él en un conjunto de datos históricos. Algunas veces se llama la técnica del vecino k-más cercano.
- Regla de inducción: La extracción de reglas if-then de datos basados en significado estadístico.

7 Conclusión

En la actualidad, el uso de repositorios educativos se hace cada vez más frecuente, ya que los avances tecnológicos hacen que los profesores e instituciones vayan buscando nuevas herramientas para poder fortalecer el contenido académico. Esto permite poder encontrar un gran número de recursos de aprendizaje, de los cuales se hace necesario mantener información referente al respecto. Para poder realizar búsquedas e intercambiar información acerca de los recursos de aprendizaje, se hace necesaria la creación de un repositorio local que permita poder realizar estas acciones de manera rápida y eficiente, brindando al usuario la información requerida de forma clara y ordenada.

En base a los objetivos planteados, el proyecto culmina exitosamente: Se ha propuesto la arquitectura y se han implementado los principales aspectos de un repositorio local. Esta arquitectura se estructuró en la elección de la mejor alternativa de modelos existentes para representar repositorios educativos; demostrando la validez de la propuesta por medio de la elaboración de un prototipo que permite recolectar recursos de aprendizaje desde el repositorio Khan Academy y almacenar su información en el formato requerido (metadatos que incorporan comentarios) para futuras investigaciones y proyectos. Esto quiere decir que se cumple con la totalidad de los objetivos propuestos.

Dentro de ello se infiere:

- Con el fin de poder estudiar la información contenida en los recursos de aprendizaje, es necesaria la creación de un repositorio local que cumpla con almacenar metadatos y comentarios.
- LOM es el estándar de etiquetado de recursos de aprendizaje más utilizado para el intercambio de información, además el que la representa de mejor manera debido a amplitud de sus categorías.
- Para modelar un repositorio de objetos de aprendizaje existen diferentes alternativas, como lo son la representación estructurada, semiestructurada y semántica. Cada una de ellas posee ventajas y desventajas y su utilización depende de las características del repositorio
- La arquitectura propuesta en la cual coexisten 2 formas de representación de datos, es una de las más efectivas, ya que facilita la búsqueda aproximada y

exacta. Solo basta con proveer un servicio que mantenga la integridad de los datos en ambas representaciones.

- El repositorio Khan Academy es uno de los pocos en los cuales se puede recolectar recursos de aprendizaje donde se asocian comentarios al contenido del recurso. Es por ello que pasó a ser el repositorio elegido para realizar la extracción y posterior poblado del repositorio local de estudio.

Este proyecto no estuvo ajeno a dificultades, tras enfrentarse a un lenguaje de programación nuevo en la formación académica de los autores de este documento, pero que brinda una amplia gama de posibilidades de trabajo que muchos otros lenguajes pueden solventar pero de manera más costosa.

Este proyecto deja la puerta abierta a investigadores del área para profundizar en los temas tratados y además para completar las etapas posteriores con las que fue ideada la arquitectura de trabajo, como lo es utilizar los datos del repositorio local para la realización de Análisis de Sentimiento.

8 Bibliografía y Referencias

- [Bass et al., 1997] Bass, L., Clements, P., Kazman, R., "Software Architecture in Practice". Boston: Addison-Wesley. 1997.

- [Cartujano et al., 2002] Cartujano J., Espinosa E., Lozano R. "*SQL2XQuery"*. Memorias del Congreso Internacional en Ciencias computacionales e Informáticas, Durango, México, Marzo 2002.

- [IEEE, 1990] IEEE Std 610.12-1990. "*IEEE standard glossary of software engineering terminology*". New York. 1990.

- [IEEE, 2000] IEEE Std 1471-2000. "*IEEE Recommended Practice for Architectural Description of Software-Intensive Systems*". New York. 2000.

- [IMS, 2003] IMS-DRI. IMS "*Digital Repositories Interoperability - Core Functions Information Model*", *Version 1.0 Final Specification*: IMS Global Learning Consortium, 2003.

- [IMS, 2014] IMS-SS. IMS *Simple Sequencing Specification. Global Liaoning Consortium LD 2003*. Recuperado desde http://www.imsglobal.org/simplesequencing/, en Febrero de 2014

- [Jazayeri et al., 2000] Jazayeri, M., Ran, A., Linden, F., "*Software Architecture for Product Families: Principles and Practice*". Addison Wesley. 2000.

- [IEEE, 2002] LTSC. IEEE Learning Object Metadata (LOM), Draft Standard 1484.12.1, Learning Technology Standards Committee, (2002).

- [McGreal, 2004] McGreal, R. "*Learning Objects: A practical Definition"*. International Journal of InstructionalTechnology and Distance Learning. *Vol. 1*, pp 21 -32. 2004

- [Pozo, 2002] Pozo I., "*Aprendices y maestros: la nueva cultura del aprendizaje*.". Madrid: Alianza Editorial. 2002.

- [Shaw & Garlan, 1996] Shaw, M., Garlan, D., "*Software Architecture: Perspectives on an Emerging Discipline*". Prentice Hall, 1996.

- [Metz et al., 2004] Metz, M., Kumaresan, P., Gavinlertvatana, N., Keow, K and Ramachandran, P. *Object oriented databas. 2004*. Recuperado el 27 de febrero de 2014, desde: Hhttp://www.ieor.berkeley.edu/~goldberg/courses/F04/215/215-OODB.ppt#7

- [EleGall, 2011]. *Imagen: Estructura modelado relacional LOM* Recuperado el 27 de febrero del 2014, desde http://commons.wikimedia.org/wiki/File:LOM_base_schema.svg

9 Anexos

Anexo 1: Recolector original desarrollado por el Profesor Miguel Ángel Sicilia

Nota: Las líneas rojas cumplen la función de separar las distintas funciones del código. Los textos en rojo se encuentran comentados y corresponden a explicaciones de la funcionalidad y líneas de código de prueba realizadas por el profesor Miguel Ángel Sicilia.

```
import httplib
import json
import sqlite3
from selenium import webdriver
from selenium import selenium
from  selenium.webdriver.common.action_chains import ActionChains
#_____________________________________________________________________

# The URI of Khan Academy API
KHANACAD_URI= 'www.khanacademy.org'
#_____________________________________________________________________

def storetopics(db):

    uri = "/api/v1/topic"
    # Get the root topic:
    conn = httplib.HTTPConnection(KHANACAD_URI)
    conn.request("GET", uri+"/root")
    res = conn.getresponse()
    data = res.read()
    print data

    return
#_____________________________________________________________________

def storevideos(db):
    """
     Saves Khan Academy playlist&video info in the database.
     Playlists group videos.
    """
    uri = "/api/v1/playlists"
    conn = httplib.HTTPConnection(KHANACAD_URI)
    conn.request("GET", uri)
    res = conn.getresponse()
    data = res.read()
    decoded = json.loads(data)
    for playlist in decoded:
        print playlist['title']
        __storeplaylistinfo(db, playlist)

        # Get the videos of each playlist
        conn2 = httplib.HTTPConnection(KHANACAD_URI)
        uri2 = "/api/v1/playlists/"+playlist['id']+'/videos'
        #print uri2
        conn2.request("GET", uri2)
        res2 = conn2.getresponse()
        #print res2.status, res2.reason
        data2 = res2.read()
        decoded2 = json.loads(data2)

        for video in decoded2:
             __storevideoinfo(db, video, playlist)

    conn.close
#_____________________________________________________________________
```

```
def __storevideoinfo(db, video, playlist):
    """
    Stores info on the video in the DB, including the playlist in which it is included.
    """
    title = video["title"]
    title = title.replace("'", " ")
    id = video["readable_id"]
    id = id.replace("'", " ")
    date_added = video["date_added"]
    date_added = date_added.replace("'", " ")
    duration = video["duration"]
    views = video["views"]
    relative_url = video["relative_url"]

    db.execute("INSERT INTO video VALUES ('"+ id +"','"+title+"','"+date_added+"','"+
               relative_url + "','" + str(duration)+"','" + str(views)+"')")
    db.execute("INSERT INTO playlist_video VALUES ('"+ playlist["id"] +"','"+id+"'"+")")
    db.commit()
    return
#_____________________________________________________________________

def __storeplaylistinfo(db, playlist):
    """
    Stores the information of a playlist in the db
    """
    title = playlist["title"]
    title = title.replace("'", " ")
    id = playlist["id"]
    id = id.replace("'", " ")
    kind = playlist["kind"]
    kind = kind.replace("'", " ")


    db.execute("INSERT INTO playlist VALUES ('"+ id +"','"+title+"','"+kind+"')")
    db.commit()
    return
#_____________________________________________________________________

def createdb(dbname):
    """
    Creates an in-memory sqlite3 database with a schema for Cosm feeds.
    """
    # For in-memory database:
    #     conn = sqlite3.connect(':memory:')
    conn = sqlite3.connect(dbname)

    conn.execute('''CREATE TABLE playlist
                    (id text, title text, kind text)''')
    conn.execute(("CREATE TABLE playlist_video (idplaylist text, idvideo text)"))
    conn.execute('''CREATE TABLE video
                    (id text, title text,  date_added text, relative_url text,
                    duration text, views text)''')
    conn.execute('''CREATE TABLE comments
                    (videoid text, userid text, usernick text, votes integer,
                    content text, date text)''')
    return conn
#_____________________________________________________________________

def opendb(dbname):
    conn = sqlite3.connect(dbname)
    return conn
#_____________________________________________________________________
```

```
def dump_playlists_per_video(db, filename):
    """
    Dumps the playlist-video network to a Pajek File.
    """

    # Dump the number of nodes:
    res = db.execute("SELECT COUNT(*) FROM video")

    f = open(filename, 'w')
    numvideos = res.fetchone()[0]
    res2 = db.execute("SELECT COUNT(*) FROM playlist")
    numplaylists = res2.fetchone()[0]
    f.write("*Vertices " + str(numvideos + numplaylists)  + " "+ str(numvideos)+"\n")

    # Dump the nodes:
    nodenumber = 1
    res = db.execute("SELECT id FROM video order by id")
    for row in res:
        f.write(str(nodenumber)+' "'+row[0]+'"'+"\n")
        nodenumber += 1

    res2 = db.execute("SELECT id FROM playlist order by id")
    dict = {} # To keep track of the nodenumbers assigned to the playlists

    for row in res2:
        f.write(str(nodenumber)+' "'+row[0]+'"'+"\n")
        dict[row[0]] = nodenumber
        nodenumber += 1

    # Dump the edges:

    res = db.execute("SELECT * FROM video order by id")
    f.write("*Arcs " +"\n")
    f.write("*Edges " +"\n")
    nodenumber = 1
    for row in res:
        print row
        videoid = row[0]
        print videoid
        res2 = db.execute("SELECT idplaylist from playlist_video WHERE

                          idvideo='"+videoid+"'")
        nodenumber +=1
        for row2 in res2:
             f.write(str(nodenumber)+"  "+ str(dict[row2[0]]) +"\n")
             print "-->"+row2[0]
    return
#_____________________________________________________________________
```

```
def dump_video_time_series(db, filename):

    """
    Dumps a text data file with the timestamps and basic information on videos.
    """
    f = open(filename, 'w')
    res = db.execute("SELECT id, date_added FROM video order by date_added")
    for row in res:
        f.write(str(row[0]) + "," + str(row[1]) + "\n")
    return
#-------------------------------------------------------------------------------

def dump_video_contrib_frequency(db, filename):

    """
    Dumps a text data file with the count of videos contributed per day.
    """
    f = open(filename, 'w')
    res = db.execute("SELECT count(id), strftime('%Y-%m-%d', date_added) AS
                     totalvideos "+" FROM video group by
                     strftime('%Y-%m-%d', date_added) order by
                     strftime('%Y-%m-%d', date_added) desc")
    for row in res:
        f.write(str(row[0]) + "," + str(row[1]) + "\n")
    return
#-------------------------------------------------------------------------------

def ___scrap_video(db, uri, browser):
    """
    NOT WORKING: See Java implementation.
    """
    print uri +"\n"
    browser.get(uri) # Load page
    # Move to "Tips & comments"
    tipscommentslink = browser.find_element_by_xpath("//a[contains(text(),
                                                       'Tips & Comments')]")
    tipscommentslink.click()

    # Find the two list: first will be questions, second will be tips & comments
    tabs = browser.find_elements_by_xpath("//div[@class='discussion-list-content']")
    #contents = comments.find_elements_by_xpath("/div[@class='commentdiscussion-item']")
    print len(tabs)

    return
#-------------------------------------------------------------------------------

def scrap_videos(db):
    """

    """
    res = db.execute("SELECT id, relative_url FROM video")
    browser = webdriver.Firefox() # Get local session of firefox
    for row in res:
        ___scrap_video(db, "http://"+KHANACAD_URI+row[1], browser)

        break
    return
    browser.close()
#-------------------------------------------------------------------------------
```

```
def dump_comments_users_to_pajek(db, filename):

    """
    Dumps the video-user bipartite graph to Pajek.
    Arcs are weighted with number of votes of the comment.
    """

    f = open(filename, 'w')

    # Get the number of courses:
    res = db.execute("SELECT count(*) FROM video")
    for row in res:
        numvideos = row[0]
    # Get the number of users:
    res = db.execute("SELECT count(distinct(userid)) FROM comments")
    for row in res:
        numusers = row[0]

    print numusers

    res = db.execute("SELECT count(*) FROM comments")
    for row in res:
        numcomments = row[0]
    print numcomments

    f.write("*Vertices " + str(numvideos+numusers)+ " "+ str(numvideos) +"\n")

    count = 1
    pajekid_of = {}
    # Dump videos and their number of views:
    res = db.execute("SELECT id, views FROM video")
    for row in res:
         f.write(str(count) + " "+ str(row[0]) +"\n")
         pajekid_of[str(row[0])] = count
         count+=1
    # Dump users:

    res = db.execute("SELECT DISTINCT userid FROM comments")
    # Get the counts in a dictionary
    for row in res:

        f.write(str(count) + " "+ str(row[0]) +"\n")
        pajekid_of[str(row[0])] = count
        count+=1
        #print pajekid_of[str(row[0])]

    f.write("*Edgeslist " +"\n")

    # Get the videos:
    res = db.execute("SELECT id FROM video")
    for row in res:  # Get the users commenting this video
        videoid = row[0]
        res2 = db.execute("SELECT DISTINCT userid FROM comments WHERE
                          videoid = '"+ str(videoid)+"'")
        row2 = res2.fetchone()
        if  not (row2 is None):
           f.write(str(pajekid_of[str(videoid)]))
           f.write(" " + str(pajekid_of[row2[0]]))
           for row2 in res2:
               f.write(" " + str(pajekid_of[row2[0]]))
           f.write("\n")
           print videoid
    return
#_____________________________________________________________________
```

```
def main():
    #conn = createdb("ka.db")
    conn = opendb("ka.db")
    storevideos(conn)
    #dump_playlists_per_video(conn, 'khan.txt')
    #dump_video_time_series(conn, "contrib-ka.txt")
    #scrap_videos(conn)
    dump_comments_users_to_pajek(conn, "videos_users.txt")
    return
#_____________________________________________________________________

if __name__ == "__main__":
    main()
#_____________________________________________________________________
```

Anexo 2: **Detalle de la representación No Estructurada**: para el caso de estudio presentado, se muestra el detalle de la representación XML. Se muestra principalmente el título del recurso y sus correspondientes comentarios en el apartado “annotations”. Esta representación es posible encontrarla en el repositorio local.

```
▼<Repository>
  ▼<Resource>
    ▼<General>
      ▶<Identifier>...</Identifier>
      ▼<Title>
         After Words: "The One World Schoolhouse: Education Reimagined"
       </Title>
       <Languaje/>
      ▶<Description>...</Description>
       <Keywords/>
       <Coverage/>
       <Struct/>
       <Agregation_Level/>
     </General>
    ▶<Life_Cycle>...</Life_Cycle>
    ▶<Meta-Metadata>...</Meta-Metadata>
    ▶<Technical>...</Technical>
    ▶<Educational>...</Educational>
    ▶<Rights>...</Rights>
    ▶<Relation>...</Relation>
    ▼<Annotations>
       <Entity>Greg</Entity>
       <Date/>
       <Description>Sal is such an interesting guy!</Description>
       <Votes>7</Votes>
     </Annotations>
    ▼<Annotations>
       <Entity>Taylor Swift</Entity>
       <Date/>
       <Description>I love Khan Academy!</Description>
       <Votes>5</Votes>
     </Annotations>
```

```
▼<Annotations>
   <Entity>Kenneth Epley</Entity>
   <Date/>
  ▼<Description>
     At 12:20 and all through the video Nina continues to interrupt Sal with lots of
     rights and ... really it was really annoying to listen too. Hopefully Nina will
     re-watch the video and learn from her social gaffes and improve as the KA
     students do with the basic studies.
   </Description>
   <Votes>5</Votes>
 </Annotations>
▼<Annotations>
   <Entity>undefined.brilliance</Entity>
   <Date/>
  ▼<Description>
     Your view s are truly inspirational, the in depth focus on intuitive ways of
     learning and your highlighting of the need to promote and use more natural
     methods to allow students to gain maximum retention both in schools and out are
     really thought provoking. My only suggestion is that you keep pushing towards
     the end goal so that one day your dreams are more of a reality.
   </Description>
   <Votes>3</Votes>
 </Annotations>
▼<Annotations>
   <Entity>Petrie (Peter S. Asiain III)</Entity>
   <Date/>
  ▼<Description>
     Without a doubt Sal s book, "The One World Schoolhouse: Education Reimagined" IN
     the near future, would definitely be one of the most historical and influential
     books out there that will improve and change education forever!
   </Description>
   <Votes>2</Votes>
 </Annotations>

▼<Annotations>
   <Entity>Nissan Uddin</Entity>
   <Date/>
  ▼<Description>
     Wow. I read the preview of this book and it was so well put and amazing. I am
     thinking about getting the book.
   </Description>
   <Votes>2</Votes>
 </Annotations>
```

```
▼<Annotations>
   <Entity>Mark Greco</Entity>
   <Date/>
  ▼<Description>
     Sal I love you! I ve always been pretty bright, and I owe a lot of that to my
     older brother, similarly to how you credit your sister. My brother would
     introduce me to topics like exponents and square roots when I was in 1st grade,
     and I loved learning. He s 10 years older than me, so when I started junior
     high, he was busy with medical school. I still did very well in class but I felt
     like I wasn t learning anymore, and certainly less enthusiastic, when he wasn t
     around to discuss new topics… (more)
   </Description>
   <Votes>2</Votes>
 </Annotations>
▼<Annotations>
   <Entity>Ryan Lynn Wikert</Entity>
   <Date/>
  ▼<Description>
     Thank you Khan for for creating Khan Academy, you gave me hope in getting back
     my education!
   </Description>
   <Votes>2</Votes>
 </Annotations>
▼<Annotations>
   <Entity>福龍丸 (Lucky Dragon)</Entity>
   <Date/>
  ▼<Description>
     At around 33:00: I would wish to be 6 again just to live through this!
   </Description>
   <Votes>2</Votes>
 </Annotations>

▼<Annotations>
   <Entity>Nerd Herd</Entity>
   <Date/>
   <Description>can you make this shorter i lasted 46 seconds!</Description>
   <Votes>1</Votes>
 </Annotations>
▼<Annotations>
   <Entity>Unmil Parghi</Entity>
   <Date/>
   <Description>Mr. Sal Khan is the best!</Description>
   <Votes>1</Votes>
 </Annotations>
▼<Annotations>
   <Entity>Unmil Parghi</Entity>
   <Date/>
   <Description>Mr. Sal Khan is the best!</Description>
   <Votes>1</Votes>
 </Annotations>
▼<Annotations>
   <Entity>Shmuel Chaim</Entity>
   <Date/>
   <Description>thanks</Description>
   <Votes>1</Votes>
 </Annotations>
▼<Annotations>
   <Entity>Uday Parmar</Entity>
   <Date/>
   <Description>Sal must be famous.Vote up if you agree.</Description>
   <Votes>0</Votes>
 </Annotations>
```

```
▼<Annotations>
   <Entity>chenesther011</Entity>
   <Date/>
  ▼<Description>
     Post a tip or
     thanks...njbhjvkgccccccccccccvhbkhvgcvfzsedfgrtszsedsaasesrdgcfgdfcdygfugygAyrge
     fgfyrgyytyrtyr
   </Description>
   <Votes>0</Votes>
 </Annotations>
▶<Classification>...</Classification>
</Resource>
▼<Resource>
 ▼<General>
  ▶<Identifier>...</Identifier>
  ▼<Title>
     Sal s interview on the One World Schoolhouse (C-SPAN 2 After Words)
   </Title>
   <Languaje/>
   <Description/>
   <Keywords/>
   <Coverage/>
   <Struct/>
   <Agregation_Level/>
 </General>
▶<Life_Cycle>...</Life_Cycle>
▶<Meta-Metadata>...</Meta-Metadata>
▶<Technical>...</Technical>
▶<Educational>...</Educational>
▶<Rights>...</Rights>
▶<Relation>...</Relation>
▶<Classification>...</Classification>
</Resource>

▼<Resource>
 ▼<General>
  ▶<Identifier>...</Identifier>
   <Title>Sal Khan on Piers Morgan Tonight (Oct 5, 2012)</Title>
   <Languaje/>
  ▶<Description>...</Description>
   <Keywords/>
   <Coverage/>
   <Struct/>
   <Agregation_Level/>
 </General>
▶<Life_Cycle>...</Life_Cycle>
▶<Meta-Metadata>...</Meta-Metadata>
▶<Technical>...</Technical>
▶<Educational>...</Educational>
▶<Rights>...</Rights>
▶<Relation>...</Relation>
▶<Classification>...</Classification>
</Resource>
```

```
▼<Resource>
  ▼<General>
    ▶<Identifier>...</Identifier>
    ▼<Title>
       Radio Interview: Sal Khan on Diane Rehm (Oct 3, 2012)
     </Title>
     <Languaje/>
    ▶<Description>...</Description>
     <Keywords/>
     <Coverage/>
     <Struct/>
     <Agregation_Level/>
   </General>
  ▶<Life_Cycle>...</Life_Cycle>
  ▶<Meta-Metadata>...</Meta-Metadata>
  ▶<Technical>...</Technical>
  ▶<Educational>...</Educational>
  ▶<Rights>...</Rights>
  ▶<Relation>...</Relation>
  ▶<Classification>...</Classification>
 </Resource>
▼<Resource>
  ▼<General>
    ▶<Identifier>...</Identifier>
    ▼<Title>
       Radio Interview: Sal Khan on NPR s Talk of the Nation (October 23, 2012)
     </Title>
     <Languaje/>
    ▶<Description>...</Description>
     <Keywords/>
     <Coverage/>
     <Struct/>
     <Agregation_Level/>
   </General>
  ▶<Life_Cycle>...</Life_Cycle>
  ▶<Meta-Metadata>...</Meta-Metadata>
  ▶<Technical>...</Technical>
  ▶<Educational>...</Educational>
  ▶<Rights>...</Rights>
  ▶<Relation>...</Relation>
  ▶<Classification>...</Classification>
 </Resource>
▼<Resource>
  ▼<General>
    ▶<Identifier>...</Identifier>
    ▼<Title>
       Radio Interview: Sal Khan on Brian Lehrer (Oct 4, 2012)
     </Title>
     <Languaje/>
    ▶<Description>...</Description>
     <Keywords/>
     <Coverage/>
     <Struct/>
     <Agregation_Level/>
   </General>
  ▶<Life_Cycle>...</Life_Cycle>
  ▶<Meta-Metadata>...</Meta-Metadata>
  ▶<Technical>...</Technical>
  ▶<Educational>...</Educational>
  ▶<Rights>...</Rights>
  ▶<Relation>...</Relation>
  ▶<Classification>...</Classification>
 </Resource>
```

```
▼<Resource>
  ▼<General>
    ▶<Identifier>...</Identifier>
     <Title>Sal Khan on CNN Starting Point (Oct 5, 2012)</Title>
     <Languaje/>
    ▶<Description>...</Description>
     <Keywords/>
     <Coverage/>
     <Struct/>
     <Agregation_Level/>
   </General>
  ▶<Life_Cycle>...</Life_Cycle>
  ▶<Meta-Metadata>...</Meta-Metadata>
  ▶<Technical>...</Technical>
  ▶<Educational>...</Educational>
  ▶<Rights>...</Rights>
  ▶<Relation>...</Relation>
  ▶<Classification>...</Classification>
 </Resource>

▼<Resource>
  ▼<General>
    ▶<Identifier>...</Identifier>
     <Title>Authors@Google: Salman Khan</Title>
     <Languaje/>
    ▶<Description>...</Description>
    ▶<Keywords>...</Keywords>
     <Coverage/>
     <Struct/>
     <Agregation_Level/>
   </General>
  ▶<Life_Cycle>...</Life_Cycle>
  ▶<Meta-Metadata>...</Meta-Metadata>
  ▶<Technical>...</Technical>
  ▶<Educational>...</Educational>
  ▶<Rights>...</Rights>
  ▶<Relation>...</Relation>
  ▶<Classification>...</Classification>
 </Resource>
```

```
▼<Repository>
  ▶<Resource>...</Resource>
  ▶<Resource>...</Resource>
  ▶<Resource>...</Resource>
  ▶<Resource>...</Resource>
  ▶<Resource>...</Resource>
  ▶<Resource>...</Resource>
  ▶<Resource>...</Resource>
  ▶<Resource>...</Resource>
  ▼<Resource>
    ▼<General>
      ▶<Identifier>...</Identifier>
      ▼<Title>
         Radio interview: Sal on AirTalk talking about his new book
       </Title>
       <Languaje/>
      ▶<Description>...</Description>
       <Keywords/>
       <Coverage/>
       <Struct/>
       <Agregation_Level/>
     </General>
    ▶<Life_Cycle>...</Life_Cycle>
    ▶<Meta-Metadata>...</Meta-Metadata>
    ▶<Technical>...</Technical>
    ▶<Educational>...</Educational>
    ▶<Rights>...</Rights>
    ▶<Relation>...</Relation>
    ▶<Classification>...</Classification>
   </Resource>
 </Repository>
```