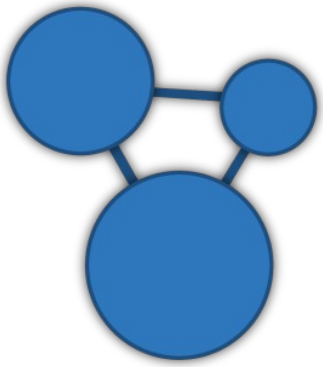
PROYECTO DE GRADO

Plataforma Web colaborativa para anotar recursos multimedia basada en Semantic Web



OPENFING

DICIEMBRE 2013

Matías Parodi
TUTOR: Ms. Ing. Fernando Carpani

RESUMEN

Cada vez son más las universidades que ofrecen abiertamente sus cursos en Internet como parte de políticas internas o bien por la adopción de metodologías de dictado alternativas a las tradicionales.

En ese sentido, el proyecto OpenFING ha dado un paso importante, grabando y publicando en Internet clases de varios cursos de la Facultad de Ingeniería, lo que se espera que permita mitigar parte de los problemas que los estudiantes enfrentan en la actualidad y sirva como una herramienta de apoyo al aprendizaje.

El propósito de este trabajo es contribuir al proyecto OpenFING mediante la construcción de una plataforma Web colaborativa que permita, entre otras cosas, la creación de fragmentos y anotaciones que describan el contenido de los videos.

Para esto se adoptaron las tecnologías de la Web Semántica y se diseñó una ontología de dominio con la cual se modeló la información del sistema. A partir de esta ontología, se usa un motor de inferencia y búsqueda que permite encontrar fragmentos relevantes sobre un tema y cruzar el contenido de los cursos.

Como resultado del proyecto se obtuvo un prototipo funcional que valida la aplicabilidad de la ontología diseñada y la viabilidad de la adopción de las tecnologías y herramientas usadas para el desarrollo de una plataforma con las características requeridas.

La solución propuesta permite la creación de fragmentos de videos y anotaciones con metadatos, la sugerencia de fragmentos relacionados con el video que se está mirando y la búsqueda de contenido a partir de consultas textuales. Además, gracias a la utilización de Semantic Web el prototipo provee mecanismos simples de integración de la información con otros sistemas.

Palabras Claves: Web colaborativa, Web Semántica, ontología, metadata, recursos multimedia, OpenFING

ÍNDICE GENERAL

CAPÍTULO 1: INTRODUCCIÓN

1.1. Motivación

Todos los años ingresan a la Facultad de Ingeniería de la Universidad de la República miles de estudiantes, lo cual termina implicando clases despersonalizadas y muchas veces en horarios inconvenientes, lo que fuerza a muchos de ellos a abandonar sus estudios en la facultad.

Por otro lado, cada vez son más las universidades que ofrecen abiertamente sus cursos en Internet [1] como parte de una política que incentiva la publicación del conocimiento impartido en dicha institución, o bien por la adopción de metodologías de dictado a distancia o de otros estilos conocidos como *hybrid learning* o *blended learning* [2].

El costo, en horas docentes, que conllevaría la introducción de metodologías de enseñanza y aprendizaje a distancia es muy alto debido al cambio en la forma en que se deben pensar las actividades en los cursos. Este factor sumado a la sobrecarga de trabajo que tienen habitualmente hace que sea difícil que lleven adelante este tipo de soluciones.

Es bastante claro también, que los estudiantes se enfrentan con algunos problemas en los cursos presenciales, entre otros: la superpoblación de clases, dificultad temática e inmediatez del dictado y poca disponibilidad horaria debido a limitaciones laborales en muchos casos.

Este análisis hace pensar que disponer de videos de las clases puede ser una herramienta que aporte en dos sentidos: desde el punto de vista del estudiante, ya que le permitiría ir a otro ritmo que el que impone la clase, y desde el punto de vista del docente, ya que es poco costoso dado que solo tiene que permitir la filmación de sus clases.

En ese sentido, el proyecto OpenFING[1] Apéndice A, propuesto e impulsado por estudiantes desde el 2012, trabaja en la digitalización y publicación en Internet de cursos de la Facultad, lo que se espera que permita mitigar parte de los problemas mencionados anteriormente y que sea una herramienta de apoyo al aprendizaje para los estudiantes de la institución.

Dado el volumen y características de la información publicada resulta difícil encontrar fragmentos específicos asociados a determinados temas, más aún, resulta imposible cruzar el contenido de los cursos de manera de estudiar un mismo tema desde distintas perspectiva. Por otro lado, sería difícil comprometer a los profesores a que dediquen horas de su tiempo en anotar los videos o agregar descripciones detalladas del contenido de los mismos.

1 http://open.fing.edu.uy/

1.2. Objetivos

De acuerdo a lo planteado anteriormente, el propósito de este trabajo es contribuir al proyecto OpenFING mediante el desarrollo del prototipo de una plataforma Web colaborativa, para lo cual se plantean los siguientes objetivos:

- Diseñar una ontología de dominio que permita modelar recursos multimedia y metadatos asociados a fragmentos de video de manera de poder relacionar estos últimos entre sí.

- Crear un prototipo que permita evaluar la viabilidad del desarrollo de una plataforma Web colaborativa de anotación de videos basada en la ontología diseñada.

- Implementar un mecanismo simple que permita a los usuarios, debidamente identificados en el sistema, contribuir con el proyecto, creando las estructuras conceptuales que reflejan las relaciones entre los contenidos de los cursos publicados y entre éstos y recursos externos al sistema.

1.3. Resultados Esperados

A partir de los objetivos planteados se definen los siguientes resultados esperados para el proyecto:

- Relevar información sobre otros proyectos relevantes y las principales características de las herramientas relacionadas a las tecnologías de la Web Semántica y al uso de estándares para este dominio.

- Crear una ontología de dominio que permita modelar los datos generados por el proyecto OpenFING y problemas de similares características.

- Validar el diseño y usabilidad de la ontología en un caso de estudio real a través del desarrollo de un prototipo de la plataforma Web colaborativa que permita fácilmente anotar fragmentos de clases utilizando la ontología diseñada.

1.4. Organización del Resto del Documento

En el capítulo 2 se describe el problema planteado y se detallan algunos de las dificultades que presenta, junto al relevamiento de los principales conceptos existentes en la realidad y la lista de los requerimientos que serán tenidos en cuenta en el trabajo.

En el capítulo 3 se presenta una introducción a los vocabularios de modelado que se usarán en en todo el trabajo, y se realiza un estudio de los proyectos y estándares relacionados más relevantes al proyecto.

En el capítulo 4 se estudian las dos ontologías de dominio diseñadas como parte de la solución al problema, y se detalla como éstas se relacionan con las presentadas en el capítulo anterior. Además, se presentan algunas consideraciones importantes que se desprenden del diseño propuesto, junto a un caso de estudio que demuestra el uso de las mismas en un problema real.

En el capítulo 5 se presenta una descripción del prototipo implementado para validar las ontologías propuestas, se estudia además los principales componentes del mismo y algunas de sus limitaciones.

En el capítulo 6 se consideran tres casos de uso típicos y se analiza de que manera la solución presentada en este trabajo resuelve cada uno de ellos.

En el capítulo 7 se presentan las conclusiones y trabajo a futuro.

El detalle de temas específicos se expone en los apéndices correspondientes, seguidamente se presenta una lista de bibliografía consultada.

CAPÍTULO 2: DESCRIPCIÓN DEL PROBLEMA

2.1. Ejemplo de Uso

En esta sección se describe un caso de uso en donde un estudiante de la Facultad interactúa con el sistema. Se espera que a partir del mismo se desprendan los principales requerimientos de la plataforma, sin embargo no todas las funcionalidades descritas serán implementadas en esta primer versión.

El estudiante en cuestión llega a la página principal del sitio Web, en la cual encuentra información acerca del proyecto, de que se trata, formas de contacto, etc. Seguidamente se identifica en el sistema utilizando sus credenciales de la universidad (cédula de identidad y contraseña de bedelías), lo que le habilitará nuevas funcionalidades en el futuro.

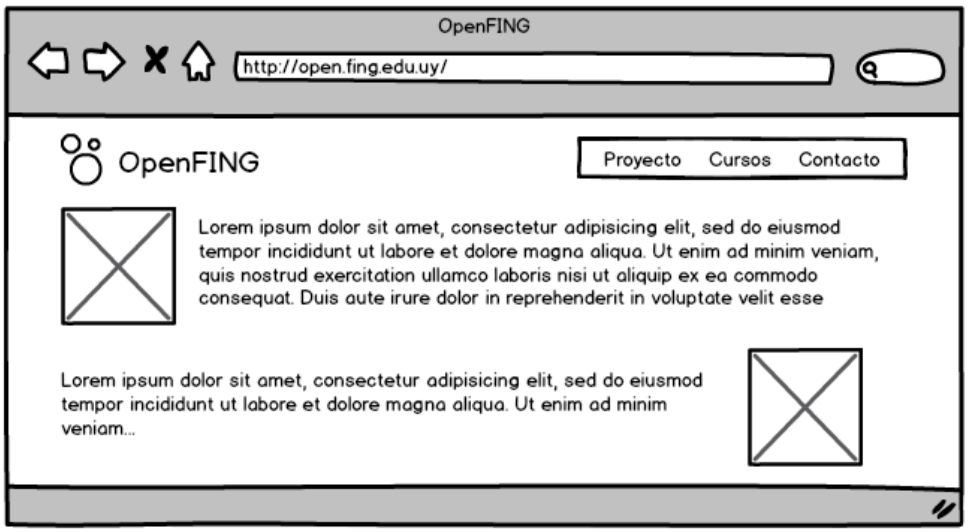


Figura 1: Sección principal con información del proyecto.

Una vez identificado navega entre las distintas secciones y lee información detallada acerca de los cursos ofrecidos y los profesores que lo dictan. Como resultado, encuentra el curso de Cálculo 1, en el que se encuentra inscrito, y decide ver un video para experimentar con el sistema.

En la sección en donde se ve el video puede encontrar el título de la clase, y una breve descripción acerca del contenido presentado en la misma, a partir de esto logra hacerse una idea de los temas tratados. A la derecha del video el sistema le ofrece una lista con fragmentos de otros videos que fueron previamente etiquetados por otros usuarios. En esta lista reconoce uno de sus temas favoritos, inducción, y decide hacer click en él, lo que ocasiona que el reproductor salte a ese punto en la clase.

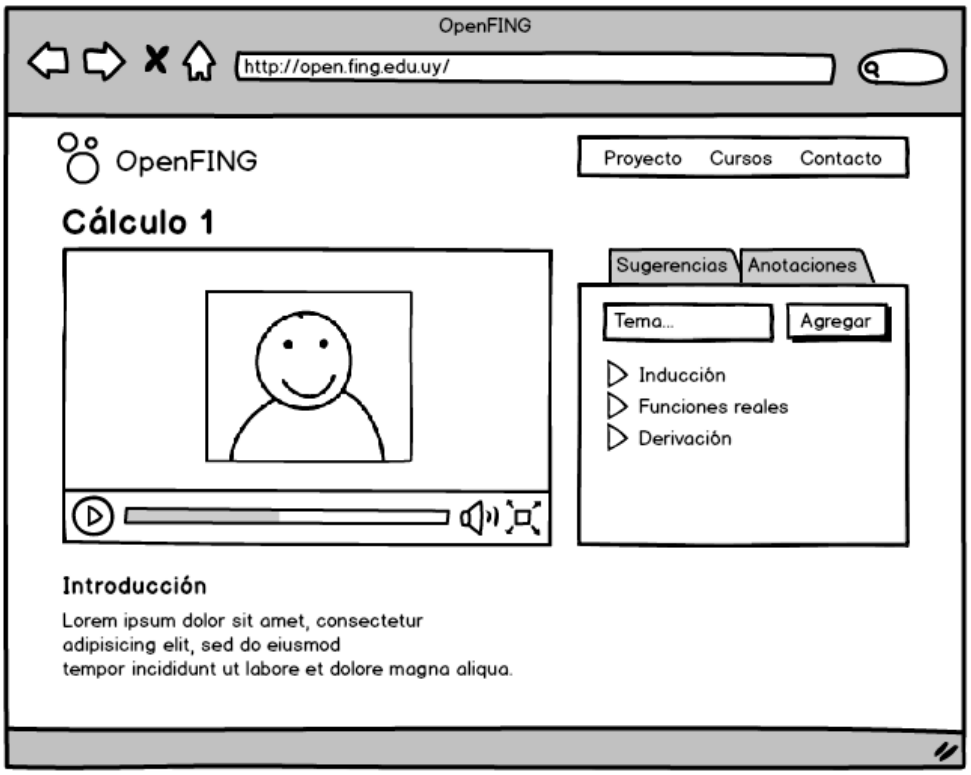


Figura 2: Vista de un video de una clase

Rápidamente encuentra la utilidad de contar con estas etiquetas, ya que le ahorra mucho tiempo en buscar temas específicos que le gustaría repasar, por lo que decide crear algunas etiquetas más. Para esto, revisa el contenido de otro fragmento de la clase y crea un nuevo tema, "conjuntos inductivos".

Más adelante en el video el profesor empieza a explicar los distintos tipos de funciones con dominio real, a lo que el sistema le ofrece ver un fragmento del curso de Física 1 que fue etiquetado como "función de energía". El estudiante sigue este enlace con el objetivo de ver ejemplos reales en este otro curso, al que también está anotado.

A medida que el estudiante pasa de un fragmento a otro, el sistema registra la secuencia de URLs visitadas junto a la anotación que lo llevó ahí. Esto es usado para ordenar las sugerencias de acuerdo a la cantidad de usuarios que siguieron ese enlace, y para estudiar la conducta de los mismos con el fin de encontrar otro tipo de relaciones entre los videos.

Satisfecho por lo que vio, decide ver un poco más, y para su sorpresa nota que otro intervalo, en donde se habla de rozamiento dinámico, está etiquetado como "lógica de predicados", tema que claramente está desvinculado del contenido real presentado. Para arreglar este problema decide votar negativamente la anotación, que ya había recibido 4 votos negativos de otros usuarios, por lo que la misma desaparece de la lista.

Desde esta clase pasa a la sección principal del curso de Física, en la que encuentra un temario completo del curso. Para su sorpresa entre los puntos presentados encuentra su propia anotación, “rozamiento dinámico”, lo que le hace pensar que el sistema genera estas listas dinámicamente a partir de los aportes de sus usuarios.

Seguidamente el estudiante decide volver a la clase de Cálculo 1 que estaba viendo originalmente. Unos pocos minutos después de donde había quedado el profesor hace un paréntesis para comentar acerca de algunas aplicaciones del análisis de funciones en distintas ramas de la ingeniería, y entre las cosas que menciona se encuentra el estudio del desempeño de algoritmos, lo que le da la idea de buscar videos relacionados con “programación”, tema que maneja pero le gustaría saber más, por lo que ingresa esta búsqueda en el cuadro de búsqueda de la plataforma.

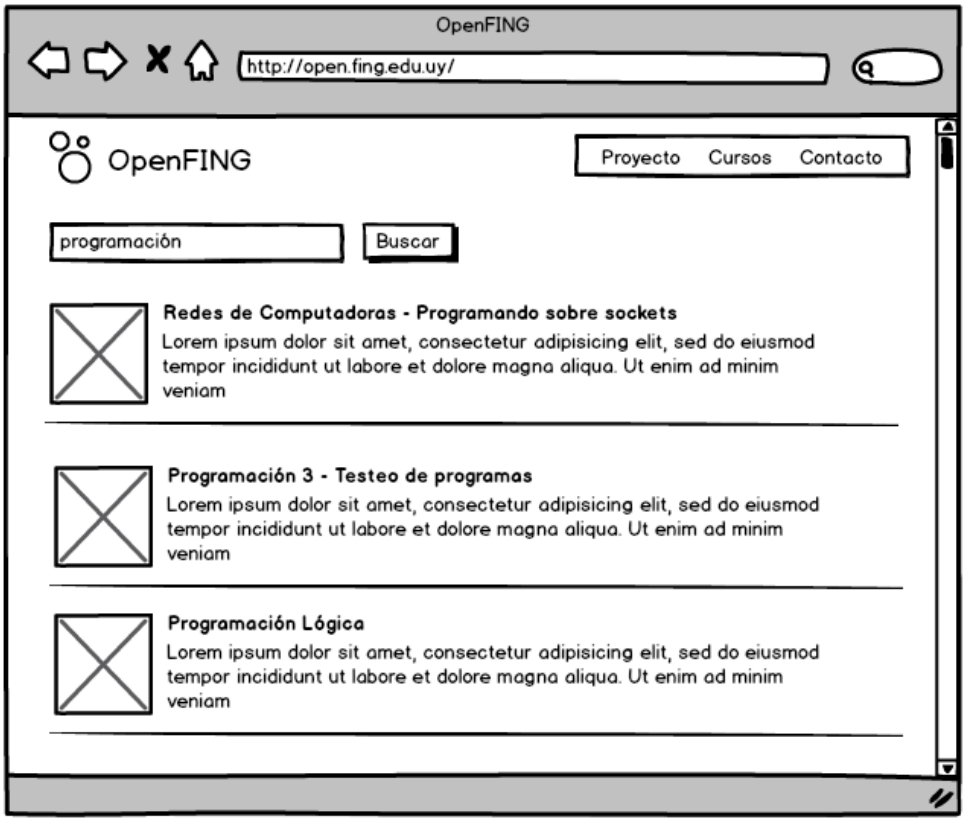


Figura 3: Resultados de una búsqueda

Para esta consulta el sistema devuelve distintos resultados, entre ellos un fragmento introductorio de programación que fue etiquetado como “programando sobre sockets”, etiqueta que está semánticamente relacionado a la consulta ingresada por el estudiante.

El contenido del fragmento le gusta bastante, por lo que decide compartirlo con sus compañeros de Programación 1. Para esto copia la URL del video, que hace referencia a ese intervalo en específico, y lo envía en un mensaje a EVA.

2.2. Visión General

Tradicionalmente un sistemas de estas características se desarrollaría persistiendo los datos en una base de datos relacional [3]. Existen muchas razones bien fundamentadas que sostienen esta decisión, sin embargo también existen otras razones por las cuales no conviene hacerlo, como se verá a continuación.

Entre las principales ventajas asociadas al uso de bases de datos relacionales se encuentra la variedad de productos estables, que se han desarrollado durante décadas y que resuelven el problema de manera consistente y en base a los estándares de SQL [4]. Además, debido a la misma popularidad de los manejadores de bases de datos relacionales (RDBMS) existe una infinidad de bibliotecas que facilitan la comunicación y manipulación de datos, en particular se han desarrollado muchos mapeadores objeto-relacional (ORM), de los que se comentará más adelante.

Si bien parece razonable adoptar este enfoque, debido a la necesidad de relacionar entidades que no están explícitamente relacionadas, prima la necesidad de contar con un mecanismo flexible y potente, que permita realizar inferencias y trabajar a nivel semántico con los datos.

En los últimos años ha aparecido una nueva tendencia en la Web llamada Semantic Web [5], que plantea cambiar la forma en la que se publica información en Internet.

La Web Semántica es una extensión de la World Wide Web (WWW) que permite a las personas compartir contenido más allá de los límites de las aplicaciones y sitios Webs. El World Wide Web Consortium (W3C) [6], comunidad internacional encargada de estandarizar la tecnología sobre las que se fundamenta la Web, la define como “la Web de datos” y propone como su objetivo último el permitir que las computadoras hagan un mejor trabajo, mediante la publicación de información legible tanto para humanos como para computadoras a través de vocabularios bien definidos y con una semántica clara.

Asociado a esta tendencia, impulsada por Tim Berners-Lee, quien es considerado el padre de la Web tal cual la conocemos hoy, la W3C ha estandarizado diferentes formatos y lenguajes de modelado y consulta [7].

Durante los últimos años estas tecnología han ganado el apoyo de muchas instituciones, tanto públicas como privadas [8]–[11], y de empresas que se dedican al desarrollo de herramientas que implementan los estándares. Junto con estos productos aparecieron bibliotecas para distintos lenguajes de programación y se han propuesto formas de mapear el esquema de bases de datos relacionales a la Web Semántica [12].

Existen algunas ventajas claras en esta tecnología. En primer lugar, debido al propio diseño de la Web Semántica no se impone un esquema rígido de datos, lo cual implica una flexibilidad mayor que en una base de datos relacional. De este modo si en el futuro se desea agregar un

nuevo tipo de datos, o una especialización de uno ya existente, en una base de datos relacional se requeriría potencialmente una reestructuración completa, mientras que en una base de datos que implementa estos principios de diseño el costo es muy bajo.

Esta adaptabilidad es siempre algo deseable en cualquier sistema, principalmente en uno como el que se intenta desarrollar aquí en donde no se tienen antecedentes y por tanto es de esperar que los requerimientos de modelado cambien en un futuro.

Por las razones mencionadas y por el hecho de que la complejidad no está en la persistencia de los datos sino en la forma en la que se relacionan los mismos, se decidió adoptar las tecnologías y estándares propuestos por Semantic Web.

El procedimiento del desarrollo de cualquier sistema que adopte esta tecnología se puede resumir en 3 grandes pasos. En primer lugar, se debe hacer un relevamiento de las entidades relevantes para el sistema. A partir de esto, en una segunda etapa se debe diseñar una ontología de dominio, es decir, describir formalmente la semántica de las clases, relaciones y propiedades que se usarán para definir instancias concretas de datos. Por último, se crea una interfaz, en este caso un sitio Web, que permita consumir y modificar el contenido de la base de datos de forma amigable para el usuario final.

2.3. Problemas a Enfrentar

En primer lugar se debe tener en cuenta que muchas de las tecnologías utilizadas son relativamente nuevas y por ende no son ampliamente usadas en la industria, en relación a la penetración que han tenido las bases de datos relacionales en las últimas décadas. Esto implica que existan menos bibliotecas y herramientas desarrolladas, lo que tiene asociado un gran riesgo desde el punto de vista técnico.

Otro problema a destacar consiste en la integración de todos los componentes relevantes en la solución planteada. Si bien existen productos que son relativamente conocidos en este ámbito, como Virtuoso[2], Jena[3], Stardog[4] y Pellet[5], al parecer los mismos no disponen de mecanismos confiables que permitan integrarse unos con otros de forma de cubrir todas las necesidades del sistema.

Al mismo tiempo, poco se conoce respecto a la performance de dichos productos en contextos reales y de estas características, ya que esto depende fuertemente de las reglas de inferencia aplicadas por el motor de inferencia y la infraestructura con la que se cuente, entre otras cosas.

2 http://virtuoso.openlinksw.com/
3 http://jena.apache.org/
4 http://stardog.com/
5 http://clarkparsia.com/pellet/

Adicionalmente, se detecta un problema relativo al diseño de la ontología de dominio. Se pretende que la misma pueda ser utilizada por otros sistemas en donde sea necesario modelar la metadata asociada a fragmentos de recursos multimedia, sin embargo, esta realidad depende fuertemente del lugar en donde se implemente la solución, por lo que la ontología deberá ser suficientemente general como para que pueda ser utilizada en contextos similares y extendida de ser necesario, ante cambios de requerimientos.

Existen otros problemas relacionados a la seguridad de la información, que no serán tratados exhaustivamente en este trabajo ya que escapa al alcance, pero no se puede negar que en futuro aparecerá la necesidad asegurar cierto grado de confidencialidad de la información sensible de usuarios que interactúan con el sistema.

2.4. Requerimientos

A pesar de que el prototipo a construir se plantea como un mecanismo de validación del diseño y el uso de la ontología aplicada al proyecto OpenFING, existen muchos requerimientos a cubrir para alcanzar los resultados esperados. Algunos de ellos se desprenden fácilmente del caso de uso presentado, otros están relacionados con las características de la Web Semántica y no serán necesariamente visibles para el usuario final del sistema.

- Diseñar de una ontología de dominio que modele la realidad planteada utilizando estándares de la W3C de manera que pueda ser utilizada para resolver este y otros problemas similares.

- Deberá permitirse, a los usuarios identificados, crear anotaciones y relacionarlas a fragmentos de videos.

- Deberá asociarse el contenido de distintos cursos a través de las anotaciones creadas por los usuarios del sistema, recomendando fragmentos estrechamente relacionados.

- Deberá ofrecerse un mecanismo de búsqueda a nivel conceptual, mediante el análisis de las consultas ingresadas por los usuarios.

- Deberá permitirse la navegación entre las distintas entidades presentes en el sistema de forma de ver todo el contenido relacionado a ellas.

- Deberá ser capaz de integrarse con otras aplicaciones a través de una interfaz pública de consulta que permita la extracción de datos estructurados y fácilmente procesables por una computadora.

CAPÍTULO 3: TRABAJOS RELACIONADOS Y ESTÁNDARES

Existen muchos proyectos y estándares que se relacionan de una u otra manera con el presente trabajo. Por esta razón, se dividirá el estudio en secciones, comenzando con proyectos de publicación de clases en video, siguiendo con los estándares de modelado propuestos por la W3C, luego se verán algunas ontologías de dominio relevantes y usadas por la industria y finalmente analizaremos como se ha usado la Web Semántica en aplicaciones reales.

3.1. Trabajos Relacionados

En todo el mundo existen proyectos similares a OpenFING, muchos de los cuales nacieron como experimentos llevados a cabo por Universidades de renombre internacional y hoy son empresas establecidas, como es el caso de Coursera[6] y Udacity, ambas provenientes de la Universidad de Stanford[7]. Otros proyectos como edX, fueron fundados en acuerdos hechos entre más de una Universidad, en este caso Harvard y MIT, y aún son mantenidos por estas universidades.

Todos estas iniciativas permiten a grandes rasgos lo mismo, disponiendo de decenas y hasta centenas de cursos de distintas Universidades, destacándose Coursera con más de 500 cursos de instituciones educativas de todo el mundo.

A pesar de esto, pocos han sido los esfuerzos para ofrecer un mecanismo de anotación de fragmentos que permita a los estudiantes buscar por un tema en concreto dentro de un video. Esto puede deberse a distintas razones, en particular en Coursera las clases son divididas manualmente en módulos de 15 minutos aproximadamente, y a cada uno de estos módulos se les asocia un título que describe el contenido del video. De alguna manera el texto asociado a cada módulo parece ser suficiente para el caso de uso que implementan, sin embargo, de esta forma no se puede inferir nada respecto a la distribución de los temas dentro de cada módulo, impidiendo realizar una búsqueda precisa que devuelva como resultado un fragmento de unos pocos minutos si corresponde.

Existe otro proyecto llamado Annotating Academic Video Tool [13], desarrollado por distintas Universidades Suizas, que pretende brindar un servicio de anotación capaz de integrarse a cualquier reproductor de video online. Este servicio solo consiste en el front-end y no ofrece un mecanismo de persistencia, para cubrir esta necesidad se limitaron a desarrollar adaptadores para dos manejadores de video: SWITCHcast[8] y Opencast Matterhorn. Si bien el proyecto es prometedor y está relativamente activo, aún no se cuenta con una versión estable y madura. Más importante aún, solo resuelve el problema relacionado al mecanismo de anotación sobre el reproductor, dejando afuera gran parte del trabajo como la sugerencia de fragmentos a través de las anotaciones, modelado de distintos tipos de metadatos, etc.

6 https://www.coursera.org/ https://www.udacity.com/ https://www.edx.org/
7 http://www.stanford.edu/ http://www.harvard.edu/ http://mit.edu/
8 https://cast.switch.ch http://opencast.org/matterhorn/

Otro emprendimiento parecido al presentado en este trabajo es Unova[9]. Sus autores proponen un mecanismo de anotación de videos [14] similar al presentado aquí, lo que esperan que ayude a mejorar la enseñanza en México, país en el que se está desarrollando el sistema. A pesar de que el proyecto tiene dos años de existencia aún no se ha completado, por lo que poco se conoce del funcionamiento del mismo.

3.2. Estándares de Modelado

3.2.1. Resource Description Framework (RDF)

RDF es una familia de especificaciones estandarizadas por la W3C [15] originalmente diseñado como un modelo de datos. Actualmente es usado como un método general para la descripción conceptual de información a través de sus numerosos formatos y sintaxis como Notation3 [16], Turtle [17], RDF/XML, etc.

Se podría decir que RDF es similar a otros enfoques de modelado conceptual de datos como el modelo entidad-relación o los diagramas de clases, y se basa en la afirmación de *oraciones* de la forma sujeto-predicado-objeto. El sujeto es el objeto *descrito*, del que se dice que tiene relación con el *objeto*, a través del *predicado* usado.

Estos tres componentes conforman lo que se conoce como *ternas* o *ternas* y se pueden ver como una arista de un grafo, desde este punto de vista los triplestores no son más que manejadores de bases de datos que persisten grafos. Los grafos a su vez pueden tener un nombre, lo que da lugar a *quads*.

Si bien esto es cierto, en la práctica los datos conviven con vocabularios, que definen con precisión cuál es la semántica de cada una de las entidades que aparecen: clases, predicados, propiedades, etc. De esta forma al realizar una consulta, como se verá más adelante, no solo se estará usando las afirmaciones realizadas explícitamente a través de las ternas persistidas, sino que existen ternas que deberán ser inferidas a partir de los vocabularios usados.

Como un ejemplo introductorio se considera el siguiente conjunto de datos serializados utilizando la sintaxis conocida como Turtle.

```
data:MyCourse rdf:type edu:Course.
data:MyCourse edu:name “Course Example”.
data:MyCourse edu:hasProfessor data:John.
data:MyCourse edu:hasProfessor data:Anna.
```

Esta es sola una de las formas propuestas por Turtle para serializar dichas ternas, en la práctica hay formas más compactas, aunque equivalentes, como la siguiente.

9 http://unova.mx/

```
data:MyCourse
  rdf:type          edu:Course;
  edu:name          “Course Example”;
  edu:hasProfessor data:John, data:Anna
.
```

Si bien este es un ejemplo sencillo, presenta sutilezas que resulta conveniente notar.

En particular, *rdf*, *edu y data* son prefijos o atajos que permiten mayor legibilidad al escribir las ternas, pero en la práctica son expandidos a URIs. Por ejemplo, el predicado *rdf:type* es en realidad un atajo de *<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>*. Dicho esto, es importante considerar que en RDF todo es una URI (a excepción de los literales, como “Course Example”).

Dado que Semantic Web está diseñado de manera que cualquiera pueda decir lo que sea acerca de cualquier cosa [18], es necesario definir los prefijos que se usan junto a los datos, o bien especificar la URI completa como se mostró antes.

Es importante resaltar que si bien rdf:type podría hacer referencia a cualquier predicado con una semántica arbitraria, por convención se asume que se hace referencia a la establecida por la W3C. De este modo los razonadores que operan sobre estos datos pueden interpretarlos y hacer inferencia sobre los mismos.

El alcance del vocabulario RDF es limitado ya que no permite describir muchas cosas más allá de la información concreta que conforma el sistema, para el resto se han definidos otros vocabularios como RDFS y OWL.

3.2.2. Resource Description Framework Schema (RDFS)

RDFS es otro de los estándares producidos por la W3C [19] y consiste en un vocabulario que extiende RDF, proveyendo elementos básicos para la descripción de ontologías (vocabularios RDF).

Entre las cosas que agrega se encuentran distintas clases como rdfs:Class (la clase de las clases), predicados para especificar el dominio y rango de un predicado (rdfs:domain y rdfs:range), los predicados de subclase y subpropiedad (rdfs:subClassOf y rdfs:subPropertyOf), entre otros.

3.2.3. Web Ontology Language (OWL)

Al igual que RDF y RDFS, OWL es una familia de estándares definidos por la W3C [20, p. 2] y extiende a RDFS aún más, incorporando nuevas clases y predicados.

Existen distintos *perfiles* de especificaciones que se diferencian en su expresividad y por tanto en el tipo de inferencias que se pueden realizar, a saber: EL, QL, RL y DL [21].

Entre las cosas importantes que agregan se encuentra una subclase muy usada de rdfs:Class (llamada owl:Class) entre otros muchos otros predicados particularmente útiles para el diseño de cualquier ontología (owl:equivalentClass, owl:disjointWith, owl:inverseOf, etc).

Es particularmente interesante mencionar que en este vocabulario se definen las clases de tres de los componentes más importantes en RDF. Como vimos en el ejemplo, hay URIs que representan individuos concretos, pertenecientes a owl:NamedIndividual, por otro lado hay predicados que relacionan objetos, pertenecientes a owl:ObjectProperty y finalmente hay predicados entre objetos y literales, pertenecientes a owl:DatatypeProperty.

Además, cabe resaltar que tanto RDFS como OWL son vocabularios definidos sobre RDF, por lo que su definición también está escrita en en forma de ternas.

3.2.4. SPARQL Protocol And RDF Query Language (SPARQL)

Como se ha visto anteriormente, existen equivalencias claras entre los conceptos encontrados en sistemas con bases de datos relacionales y sistemas que usan tecnologías de la Web Semántica. Por ejemplo, cada una de las celdas de una tabla en un RDBMS se corresponde con una terna RDF, el esquema de dicha tabla se corresponde con un vocabulario escrito en RDFS y OWL, los manejadores de bases de datos relacionales (RDBMS) tienen su equivalente en los triplestores y SQL su contraparte en lo que se conoce como SPARQL [22].

SPARQL es un lenguaje de consulta diseñado por la W3C para obtener y manipular instancias en formato RDF.

Una consulta típica en este lenguaje se muestra a continuación.

```
SELECT ?professor
WHERE {
  ?course rdf:type         edu:Course.
  ?course edu:hasProfessor ?professor.
}
LIMIT 10
```

Para el ejemplo presentado anteriormente, el resultado de la consulta debería ser data:John y data:Anna.

Como se puede apreciar, las variables son denotadas con un signo de pregunta delante (*?course* y *?professor*). De esta forma se puede apreciar como la clausula WHERE está compuesta por un patrón de grafo, potencialmente incompleto (en el sentido de que puede contener variables que deberán ser unificadas a objetos y predicados concretos).

Dada una consulta como ésta, el triplestore deberá encontrar en su base de datos subgrafos isomorfos al patrón presentado. Esto deberá realizarlo en muchos casos con ayuda de un razonador, ya que para resolver la consulta es necesario usar los datos declarados explícitamente y los que se deben inferir a partir de la semántica establecida en las odontologías correspondientes.

De más está aclarar que los prefijos utilizados deberán ser declarados mediante la clausula PREFIX de SPARQL, por simplicidad del ejemplo se omitió en este caso.

3.3. Ontologías de Dominio Relacionadas

La parte crucial del sistema descansa en un buen modelo de los datos, que sea suficientemente genérico como para poder adaptarse a las necesidades y extensiones futuras, pero que a la vez sea bastante expresivo, de manera que se pueda utilizar razonadores para realizar inferencia sobre él.

El modelo de datos en este contexto es una *ontología de dominio*, y consiste en la definición del vocabulario, y la semántica asociada al mismo, que se utiliza para describir la información persistida en la base de datos.

Dicho modelo no solo debe cumplir con los estándares definidos por la W3C, sino que además es una práctica habitual reutilizar ontologías existentes, por lo que será necesario estudiar la posibilidad de incorporar parte del vocabulario de estándares ya establecidos.

A continuación se describe y evalúa la posible utilidad que podría tener seis de estas ontologías en este problema en particular.

3.3.1. Ontology for Media Resources (MA)

MA es un estándar de la W3C [23] propuesto como un vocabulario para describir recursos multimedia, abarcando desde la organización que los publican hasta los subtítulos del mismo en caso de tratarse de un video, pero no se limita a eso, también define el concepto de imagen, audio, fragmento, audiencia, etc..

Si bien en un principio esta ontología parecía ser exactamente lo que se necesitaba, luego de estudiar la documentación publicada se encontró diferencias importantes en las definiciones de las clases y predicados declarados en la misma y el uso que se le daría en este sistema.

La diferencia principal está en el nivel con la que se modelan los datos. Para nuestro sistema los recursos multimedia son atómicos y no divisibles en "pistas" como se propone en MA. Si bien se busca crear fragmentos de videos, estos fragmentos serán divisiones lógicas de un archivo concreto persistido en disco y no fragmentos físicos de un mismo archivo multimedia.

Esta diferencia sutil en la semántica de los conceptos modelados tiene implicancia en los predicados definidos y por tanto en las relaciones entre ellos. Por ejemplo, se encontró que las diferentes pistas de videos de un mismo recurso multimedia, instancias de la clase *VideoTrack*, son además instancias de *MediaFragment*, que puede tener asociado una pista de audio, es decir, una instancia de *AudioTrack*, que a su vez puede tener asociada una pista de video, y así sucesivamente. Estas relaciones resultan ser, además de extrañas, demasiado específicas y describen a los recursos multimedia a un nivel de detalle completamente innecesario para el caso de uso con el que se trabaja en este proyecto.

Más allá de las diferencias encontradas, hay una cosa importantes a destacar. Se encontró que MA está alineada con muchas otras ontologías menos conocidas, lo que permite declarar datos en MA y utilizar, directa o indirectamente, alguna de estas otras ontologías para hacer consultas. Lo mismo es cierto en el otro sentido, dado datos declarados utilizando alguna de las ontologías alineadas, es posible consultar utilizando el vocabulario definido por MA.

El proceso de alineación entre dos ontologías consiste en declarar que una clase dada en una de ellas es subclase, equivalente o superclase de una clase declarada en la otra, análogamente se puede establecer una jerarquía entre los predicados definidos.

Esto puede parecer poco relevante, sin embargo es la base para un paradigma de diseño también muy extendido, que consiste en definir el vocabulario necesario de acuerdo a los requerimientos específicos de la aplicación que se está desarrollando, y alinear la misma a estándares establecidos de manera que pueda ser utilizada a través de otra ontología. En efecto, si se tiene que la clase C1 en la ontología 1 es subclase de la clase C2 en la ontología 2, y asumiendo la existencia de un objeto X de tipo C1 (y por tanto de tipo C2), si se consulta por todos los objetos de C2 utilizando únicamente el vocabulario de la ontología 2, X deberá aparecer como parte de los resultados obtenidos.

Como será el caso con el resto de los vocabularios estudiados, en general los estándares definen conceptos que son bastante genéricos como para no limitar su aplicabilidad en distintos contextos, sin embargo en la práctica se trabaja con conceptos más específicos.

3.3.2. Friend Of A Friend (FOAF)

FOAF [24] es posiblemente uno de los vocabularios más conocidos y ampliamente usados. Su foco está en la definición de la semántica de las relaciones entre personas, organizaciones y su interacción con redes sociales. Define conceptos como Agente, Documento, OnlineAccount, OnlineChatAccount, etc y predicados como accountName, age, geekcode, gender, logo, name, openid, thumbnail, etc.

Si bien la plataforma que se está desarrollando presenta características de redes sociales, no se pretende, en un principio al menos, integrar las cuentas de los usuarios con otras redes sociales ni publicar sus datos y credenciales en Internet. Por estas razones de FOAF solo fue posible sacar algunos de los conceptos claves.

3.3.3. Simple Knowledge Organization System (SKOS)

SKOS es una ontología también estandarizada por la W3C [25] diseñada para representar tesauros, esquemas de clasificación, taxonomías o cualquier otro tipo de conocimiento estructurado.

Está definida de manera muy genérica, presentando solo unas pocas clases abstractas (skos:Collection, skos:OrderedCollection, skos:Concept, skos:ConceptScheme y skos:List) y más de una docena de predicados que permiten relacionar las instancias de datos (skos:hasMember, skos:hasTopConcept, skos:hasBroader, skos:hasNarrower, skos:hasRelated, etc).

La ontología que se pretende diseñar para este sistema deberá modelar, de alguna forma, el concepto de “tema”, ya que parte sustancial del trabajo consiste en anotar los videos mediante este tipo de metadata. Parece razonable entonces considerar el uso de parte de este vocabulario, posiblemente a través de subclases y subpropiedades que permitan reducir a un nivel aceptable el alcance de los conceptos usados.

3.3.4. Dublin Core (DC)

El vocabulario DC [26], estandarizado por el grupo Dublin Core Metadata Initiative (DCMI), pretende asistir al modelado de datos relacionados a publicaciones en general.

Entre las clases que se definieron se encuentran dc:Agent (clase de entidades como personas y organizaciones) y dc:PhysicalResource (definido como "algo material"), entre otros. Para los mismos se definen muchos predicados, a modo de ejemplo se puede relacionar los autores, en general personas, con sus obras y estas con una audiencia y agentes que hayan contribuido (dc:Creator, dc:Audience y dc:Contributor).

En este contexto se puede ver a los recursos multimedia como un objeto material, no en el sentido físico de la palabra, sino como algo concreto y no abstracto. Asimismo, los agentes involucrados, en este caso profesores, se pueden considerar los creadores de dichos recursos, y las personas que aparecen en él exponiendo algún tema serían los contribuyentes.

3.3.5. Event

La ontología Event [27] no es tan conocida como las mencionadas anteriormente, sin embargo el vocabulario definido en ella resulta en gran medida muy útil para el caso de uso con el que se trabaja aquí.

Como es habitual en muchas ontologías se define el concepto de Agente, desde luego también aparece el concepto de Evento (definido como una clasificación arbitraria de un intervalo de tiempo y lugar), el Producto producido en dicho evento, etc. Sobre estos conceptos se definen muchos predicados que permiten definir con precisión, entre otras cosas, el momento y lugar exacto en el que tiene lugar el evento y quién lo organiza.

Dada la forma en la que está definida esta ontología se pueden establecer muchos puntos en común con el vocabulario que se intenta modelar. En particular los cursos y clases pueden verse como eventos, y los videos de sus clases como el producto de dicho evento.

3.3.6. Academic Institution Internal Structure (AIISO)

AIISO [28] es, junto con Event, una ontología poco conocida pero no por eso menos relevante.

Su especificación es muy simple, sencillamente define de manera muy genérica conceptos relacionados al ámbito académico, como aiiso:Institute, aiiso:College, aiiso:Department, aiiso:Course, aiiso:Program, etc. y relaciones igual de abstractas entre ellos como aiiso:partOf, aiiso:responsibilityOf, entre otras.

De esta ontología se pueden extraer los conceptos relacionados a la estructura jerárquica con la que se organizan las Universidades, comenzando por la misma institución de estudio y llegando a los cursos, existiendo un mapeo directo en cada nivel.

3.4. Otros Trabajos

Existen relativamente pocos sistemas orientados al usuario final no técnico, de los cuales se sepa que están implementados enteramente utilizando herramientas y estándares de la Web Semántica. La W3C mantiene un repositorio de estos proyectos [29], contando con alrededor de 50 sitios, ninguno de los cuales está orientado a la anotación de fragmentos de videos. A pesar de esto, resulta importante evaluar el uso que le han dado los mismos a la Web Semántica y los resultados que se han obtenido.

Quizás los casos de éxito más famosos sean los múltiples proyectos llevados a cabo por la BBC[10]. En particular, BBC Programs[11] es una iniciativa lanzada en 2007 y pretende proveer un identificador Web accesible como HTML y feeds RDF/XML, JSON y XML, para todos los programas producidos por la compañía en sus 70 canales de broadcasting.

Por otro lado, el sitio BBC Music[12], funciona como un agregador de noticias de artistas, contenido que se extrae directamente desde Musicbrainz[13] y Wikipedia, entre otros recursos externos. Esto es posible en gran medida gracias a la capacidad de algunos triplestores de incorporar recursos externos de Internet que están publicados en RDF.

De igual manera, el equipo BBC Search Team ha utilizado satisfactoriamente esta tecnología para mejorar la experiencia de los usuarios al realizar búsquedas, en lo que se conoce como Search+. Estas páginas de resultados integran el mejor contenido referente al tema seleccionado tanto de repositorios internos de información como de recursos externos como DBPedia.

En resumen, un patrón recurrente tanto en los muchos proyectos de esta compañía como en el resto de los proyectos relevados por la W3C consiste en publicar la información de manera estructurada y procesable por terceros, incentivando la aparición de aplicaciones externas que utilizan el contenido del sitio de diversas maneras, logrando extender su alcance. Por otro lado, se busca mejorar el contenido ofrecido a través de la incorporación de información externa y también relevante al tema, lo que permite mejorar el servicio de búsqueda en sus catálogos y en especial los resultados ofrecidos, logrando asi mejorar la experiencia de los usuarios.

Para cada uno de estos proyectos la BBC ha diseñado ontologías de dominio [30] como Programmes Ontology, Wildlife Ontology, Sport Ontology, etc. las cuales le permiten modelar la información publicada de manera consistente y establecer relaciones semánticas entre los

10 http://www.bbc.co.uk/
11 http://www.bbc.co.uk/programmes
12 http://www.bbc.co.uk/music
13 http://musicbrainz.org/

recursos ofrecidos. Para esto, se utilizan vocabularios estandarizados y ampliamente utilizados en el diseño de ontologías, como los que se presentan a continuación.

CAPÍTULO 4: DISEÑO DE LA ONTOLOGÍA

4.1. Análisis

A partir de lo descrito en la sección 2.1 es posible extraer varios de los conceptos clave para el sistema. Otros, como se verá enseguida, son agregados con el fin de aumentar la expresividad del vocabulario, ya que si bien el presente trabajo se enmarca en un proyecto más amplio y en donde la información que se requiere modelar es conocida, resulta interesante diseñar una ontología de dominio suficientemente general como para que pueda ser aplicada en otros problemas de similares características. De este modo, se decide buscar solución al problema tratando de ser lo más generales posibles sin sacrificar por ello la usabilidad del mecanismo de modelado.

Entre los conceptos que interesa modelar se encuentran las entidades de la jerarquía académica en la realidad de la Facultad de Ingeniería: Universidad, Facultad e Instituto. Estos tres conceptos son parte de un concepto más general, el de "organización", que a la vez es tan solo uno de los dos tipos de agentes relevantes, siendo el otro el concepto de "persona", del cual se desprenden "profesores" y "estudiantes".

Por otro lado, es evidente la necesidad de modelar "videos", los cuales se agrupan conformando "clases" y "cursos", dos términos que pueden agruparse bajo el concepto de "colección". Del mismo modo, los videos son solo uno de los tipos de "recursos multimedia" que se pueden generar a partir de una clase, por lo que se agrega también el concepto de "audio" e "imagen", y dos conceptos relevantes que a futuro posiblemente resulten útiles: "subtítulos" y "audiencia".

En este punto cabe hacer una distinción importante que será de relevancia a lo largo de todo el documento. Los videos son considerados entidades atómicas en algún sentido, como archivos persistidos en un disco duro, que si bien se podrían descomponer en pistas, audio, frames, etc, no es algo que resulte relevante modelar en este contexto.

No se debe olvidar que uno de los objetivos principales de la ontología consiste en permitir anotar los recursos multimedia mediante "metadatos", y que estos deben estar asociados a "fragmentos" de video. Se espera que en futuro existan otros tipos de metadatos, pero en principio serán necesarios dos tipos: "tema" y "recurso". El primero de estos metadatos permitirá asociar un tema, como "inducción, a un fragmento de video, mientras que el segundo permitirá asociarle un recurso externo como la página de Wikipedia[14] referente a inducción.

Otra aclaración importante, aunque sutil, en cierta medida está relacionada a la anterior respecto a la atomicidad de los videos. Por este mismo hecho, un fragmento será simplemente una división lógica de un archivo físico, por lo que será indispensable que tenga propiedades que permitan delimitar el intervalo relevante al fragmento.

14 http://www.wikipedia.org/

Por ser la plataforma colaborativa parece razonable suponer que los usuarios se verían beneficiados si pudiesen agregar texto libre. El concepto de "comentario" será entonces un tipo adicional de metadato, mediante el cual los usuarios del sistema podrán agregar aclaraciones, correcciones, hacer preguntas, etc.

En resumen, existen muchos conceptos que tiene sentido modelar, algunos de los cuales serán abstractos, mientras que otros tendrán una correspondencia muy clara con la realidad. Las relaciones expresadas anteriormente deberán poder ser establecidas entre dos conceptos siempre que esto tenga sentido, al mismo tiempo deberá existir un gran número de propiedades, como *nombre*, que aplicarán a entidades concretas y permitirá a los humanos distinguir unas de otras.

4.2. Enfoque

Como se discutió en el capítulo anterior, son varias las ontologías que se podrían usar en conjunto para cubrir todas las necesidades requeridas en este sistema. Eso implicaría la adopción de muchos prefijos distintos, y por lo tanto seis semánticas diseñadas de forma independiente, lo cual podría traer una serie de inconvenientes que se discuten a continuación.

Hacer esto no solo influye negativamente en la simplicidad que se busca al modelar datos, sino que además nada garantiza que las semánticas de los conceptos definidos en estos vocabularios son compatibles, pudiendo ocasionar potencialmente contradicciones que harían que no sea posible inferir nueva información a partir de los datos persistidos usando estos prefijos.

Adicionalmente, ninguna de las ontologías cubre todo el vocabulario que se necesita, en el mejor de los casos se podría usar varias construcciones de MA, pero quedarían afuera tanto conceptos no definidos (como el de "tema") como conceptos que no son de interés (como el de "pista"), por lo que de todas maneras habría que diseñar una extensión propia para incorporar los elementos que falten, agregando un nuevo prefijo al sistema.

El hecho de definir un vocabulario a medida simplificará el modelado ya que todos los conceptos y relaciones entre ellos están definidos bajo un mismo prefijo y su semántica es la adecuada, asegurando que la ontología está libre de contradicciones y el significado de sus elementos es el correcto para el caso de uso trabajado.

Por las razones antes mencionadas el enfoque de diseño adoptado por MA parece ser el más adecuado en este caso, es por eso que se decide diseñar una ontología a medida y posteriormente alinearla a los seis vocabularios vistos anteriormente.

El diseño de la ontología del sistema se describe a continuación y, como se verá, está dividido en dos. Por un lado se tendrá un vocabulario base genérico (MMC), sobre el que se definirá un vocabulario más específico (OFM).

4.3. Metadata for Media Collections (MMC)

MMC, adjunta a este informe, es la ontología principal con la que se trabajará y cubre gran parte del vocabulario necesario, centrándose en la descripción de colecciones de recursos multimedia y su relación con la metadata asociada a fragmentos de los mismos.

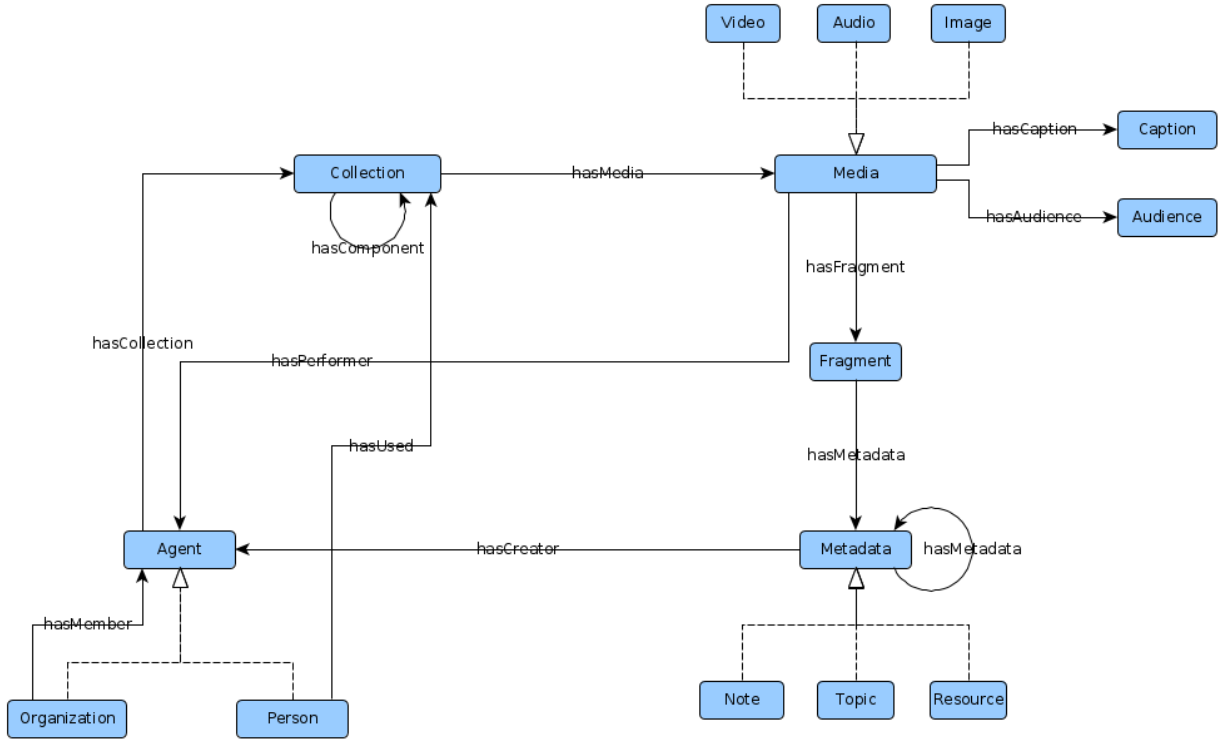


Figura 4: Simplificación del diseño de la ontología MMC

4.3.1. Conceptos

Como se puede apreciar en la Figura 4, se definieron numerosas clases relevantes para el dominio tratado, algunas de ellas no están destinadas a ser utilizadas directamente, como Metadata, sino a través de una de sus subclases más específicas.

CLASE	SUPERCLASE	DESCRIPCIÓN
Collection	-	Conjunto de colecciones o recursos multimedia.
Media	-	Recurso multimedia representando un archivo
Video	Media	Archivo de video como una película o videoclip.
Audio	Media	Archivo de audio como una canción o la grabación de una conferencia.
Image	Media	Archivo de imagen como una foto o captura de pantalla.

Fragment	-	División lógica de un recurso multimedia que define un intervalo contenido en el mismo.
Metadata	-	Datos asociados a un recurso multimedia completo o alguno de sus fragmentos.
Note	Metadata	Entidad que representa una nota consistente en texto libre asociada a un recurso.
Topic	Metadata	Entidad que representa un texto corto que describe el contenido de un recurso.
Resource	Metadata	Entidad que representa un documento Web que complementa un recurso.
Agent	-	Una organización o persona que es dueña o usa recursos multimedia.
Organization	Agent	Una organización compuesta de agentes.
Person	Agent	Una persona física.
Caption	-	Archivo de subtítulos de videos o transcripciones de audio.
Audience	-	Entidad que representa una audiencia objetivo para la cual una colección o recurso multimedia está dirigido.

4.3.2. Relaciones

Para las clases definidas anteriormente se agregan varias relaciones que permitirá realizar inferencia sobre los datos como se describirá en el caso de estudio.

PROPIEDAD	DOMINIO	RANGO
:hasMember	:Organization	:Agent
:hasCollection	:Agent	:Collection
:hasComponent	:Collection	:Collection
:hasMedia	:Collection	:Media
:hasVideo	:Collection	:Video
:hasAudio	:Collection	:Audio
:hasImage	:Collection	:Image
:hasPerformer	:Organization :Media :Collection	:Agent
:hasFavorite	:Person	:Media :Collection
:hasSaved	:Person	:Media :Collection

:hasUsed	:Person	:Media :Collection
:hasWatched	:Person	:Video
:hasListened	:Person	:Audio
:hasSeen	:Person	:Image
:hasAudience	:Media :Collection	:Audience
:hasCaption	:Media	:Caption
:hasFragment	:Media :Collection	:Fragment
:hasMetadata	:Metadata :Fragment :Media :Collection	:Metadata
:hasNote	:Metadata :Fragment :Media :Collection	:Note
:hasTopic	:Metadata :Fragment :Media :Collection	:Topic
:hasResource	:Metadata :Fragment :Media :Collection	:Resource
:hasCreator	:Metadata	:Person

Si bien en el diagrama aparecen estas relaciones desde un concepto a otro, en realidad muchas de ellas están definidas desde la unión de conceptos hacia otro como se puede ver en la tabla. Esto es necesario ya que, por ejemplo, si un fragmento tiene asociado un metadato, es razonable pensar que el recurso multimedia al cual el fragmento hace referencia también tiene asociada ese metadato, lo mismo es válido para la colección a la que pertenece ese recurso, y así sucesivamente.

Este tipo de inferencia, en donde un predicado en algún sentido en un nivel inferior de la estructura, es elevado a un nivel superior, se realiza a través de lo que se conoce como *property chain* o *composición de relaciones*, y es usado extensivamente en esta ontología.

Ejemplos de relaciones definidas de esta manera incluyen: mmc:hasMetadata (compuesta con mmc:hasMedia y mmc:hasFragment), mmc:hasFragment (compuesta con mmc:hasMedia), mmc:hasComponent (compuesta con mmc:hasCollection), etc.

Por otro lado existe un caso particular de composición de predicados, las relaciones transitivas, y en la ontología se definen algunas: mmc:hasComponent, mmc:hasMetadata y mmc:hasMember.

Como se verá más adelante, una de las ventajas de dotar a las relaciones con una semántica como esta, además de definir su dominio y rango, es que permite relacionar objetos que en principio solo están relacionados a través de objetos intermediarios, lo que simplifica significativamente las consultas.

4.3.3. Propiedades

Para cada clase se definen propiedades específicas que describen las características más relevantes de las instancias de dato.

En general estas propiedades tienen como dominio alguna de las clases de la ontología o la unión de varias de ellas, y como rango un literal definido en XML Schema [31], [32] como *xsd:string*.

En la siguiente tabla se presentan algunas de estas propiedades.

PROPIEDAD	DOMINIO	RANGO
:name	-	xsd:string
:updated	-	xsd:date
:file	:Video :Audio :Image	xsd:string
:width	:Video :Image	xsd:positiveInteger
:index	:Topic	xsd:string
...	...	...

4.4. OpenFING MMC (OFM)

MMC cubre todas las necesidades de modelado que se presentan en el proyecto, pero dado que no se quería limitar su uso a la realidad de la Universidad de la República, fue diseñada de manera que pueda ser aplicada a cualquier contexto en donde se tengan colecciones de objetos multimedia, fragmentos y metadatos asociados a ellos.

OFM, adjunta al informe, es una extensión de MMC que ofrece conceptos y predicados más específicos, construidos en base a dicha ontología, en donde aparecen entidades como University, School, College, Department, Course, etc. Estas entidades son propias de la realidad en la que está inmersa el proyecto OpenFING, y por tanto la estructura o jerarquía establecida podría ser compartida o no por otras instituciones de enseñanza. Por estas razones se decidió crear una extensión de MMC en lugar de definir en la ontología base todas las posibles clases que podrían existir en otra realidad, como por ejemplo: Escuela, Seminario, Conferencia, etc.

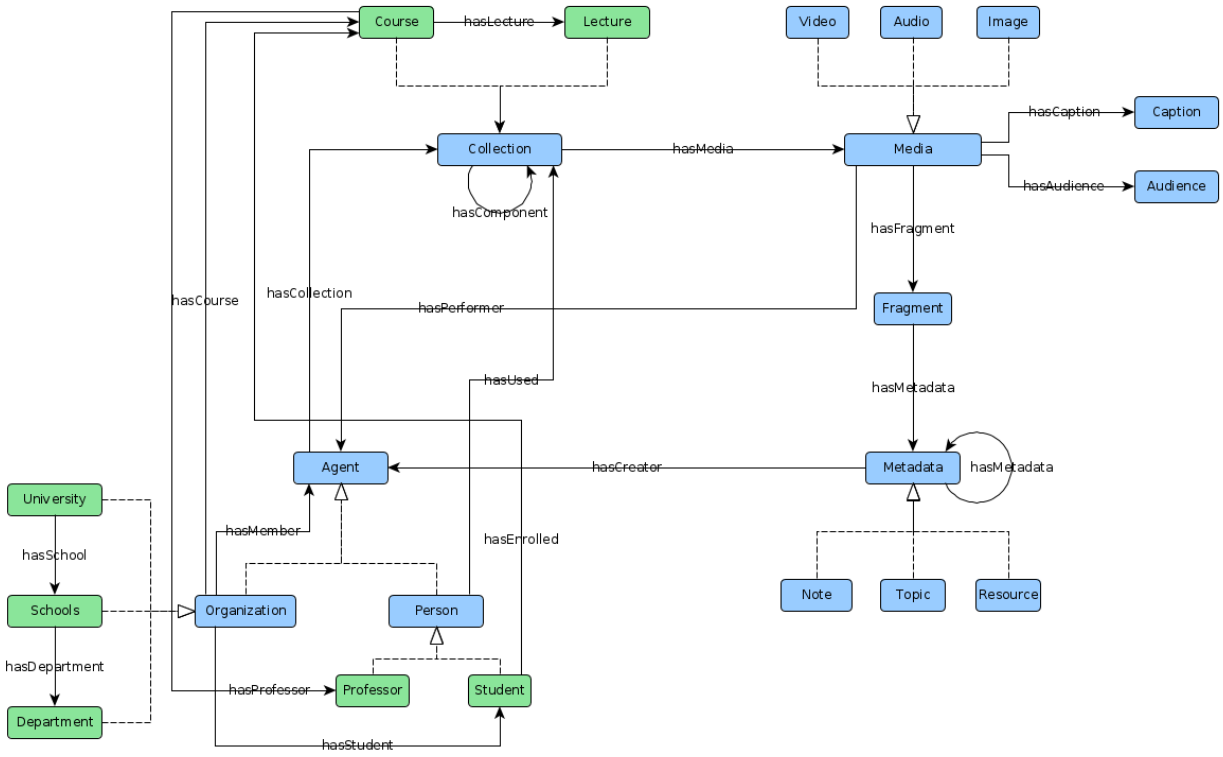


Figura 5: Simplificación del diseño de la ontología OFM como una extensión de MMC

4.4.1. Conceptos

Como es posible ver en la Figura 5, en esta ontología se agregan nuevas clases que son subclases de las definidas en MMC.

CLASE	SUPERCLASE	DESCRIPCIÓN
:University	mmc:Organization	Institución de educación superior denominada por sus miembros “universidad”.
:College	mmc:Organization	Institución de educación superior o parte de la misma denominada por sus miembros “facultad”.
:Department	mmc:Organization	Una división de una organización dedicada a una disciplina particular, denominada por sus miembros “instituto”.
:Course	mmc:Collection	Unidad de enseñanza que típicamente dura un semestre y está compuesta por muchas clases.
:Lecture	mmc:Collection	Presentación oral con la intención de exponer información y enseñar a las personas acerca de un tema particular.
:Professor	mmc:Person	Persona física que enseña en una organización.
:Student	mmc:Person	Persona física que estudia en una organización.

4.4.2. Relaciones

El hecho de que ahora las clases sean más específicas y su semántica menos abstracta permite definir nuevas relaciones como se detalla a continuación.

PROPIEDAD	DOMINIO	RANGO
:hasCollege	:University	:College
:hasDepartment	:University :College	:Department
:hasProfessor	:Organization :Media :Collection	:Professor
:hasStudent	mmc:Organization	:Student
:hasCourse	mmc:Organization	:Course
:hasLecture	:Course	:Lecture
:hasEnrolled	:Student	:Course

4.4.3. Propiedades

Además de nuevas relaciones entre conceptos aparecen propiedades de datos que son específicas para las clases creadas en esta extensión de MMC. Algunas de ellas se presentan en la siguiente tabla.

PROPIEDAD	DOMINIO	RANGO
:id	-	xsd:string
:semester	:Course	xsd:positiveInteger
:username	:Person	xsd:string
:password	:Person	xsd:string
:signin	:Person	xsd:date
...	...	...

4.5. Alineación

En la sección anterior se discutió la posibilidad de utilizar en conjunto seis ontologías conocidas, más extensiones propias que las complementen, sin embargo se adoptó un enfoque alternativo consistente en el diseño de una ontología de dominio a medida que es en una segunda etapa alineada a los estándares existentes según corresponda.

Como se adelantó, el proceso de alineación consiste en definir para cada elemento presente en la ontología que se está alineando, como se relaciona con los elementos de la otra. Estos elementos pueden ser de básicamente tres tipos, también ya presentados: clases, relaciones entre conceptos y propiedades que relacionan conceptos con literales. Para los elementos que corresponda se debe definir entonces si son subclases, clases equivalentes o superclases del elemento correspondiente en la otra ontología (análogamente para las relaciones).

A continuación se presenta parte de la alineación creada para MMC y OFM, la misma está hecha en base a la descripción y semántica de los elementos presentes en cada ontología. Aquí solo se presentará parte de las correspondencias encontradas, por una versión detallada dirigirse al Apéndice B.

ELEMENTO	RELACIÓN	CORRESPONDIENTE
mmc:Agent	rdfs:subClassOf	ma:Agent
mmc:Organization	rdfs:subClassOf	ma:Organisation
mmc:Metadata	rdfs:subClassOf	skos:Collection

mmc:Topic	rdfs:subClassOf	skos:Concept
of:University	rdfs:subClassOf	aiiso:Institution
of:College	rdfs:subClassOf	aiiso:College
of:Department	rdfs:subClassOf	aiiso:Department
mmc:hasPerformer	rdfs:subPropertyOf	ma:features
mmc:hasCollection	rdfs:subPropertyOf	ma:hasCreator
mmc:hasMetadata	rdfs:subPropertyOf	skos:broaderTransitive
mmc:hasCollection	rdfs:subPropertyOf	dc:Creator
mmc:hasPerformer	rdfs:subPropertyOf	dc:Contributor
mmc:hasMedia	rdfs:subPropertyOf	event:product
mmc:hasComponent	rdfs:subPropertyOf	event:sub_event
mmc:hasMember	rdfs:subPropertyOf	aiiso:partOf

A partir de esto es posible realizar consultas a la base de datos utilizando cualquiera de los conceptos alineados.

A modo de ejemplo, se podrían consultar todos los agentes y sus tipos como se muestra a continuación, obteniéndose los mismos resultados que de usarse mmc:Agent, ya que cualquier elemento en esta última clase también pertenece a la primera.

```
SELECT ?agent ?type
WHERE {
  ?agent rdf:type ma:Agent.
}
```

4.6. Consideraciones

Durante el transcurso del diseño de la ontología se discutió y analizó el uso y significado de cada término y la semántica asociada a cada relación y propiedad. A pesar de haber conseguido un diseño que cumple con los requerimientos establecidos y modela muy bien la realidad, existen problemas que no pudieron ser atacados debido a las limitaciones dadas por los estándares de los vocabularios usados y relaciones que no fueron modeladas ya que escapaban a la realidad modelada.

4.6.1. Relaciones Regulares

Si nos concentramos en la semántica de las relaciones mmc:hasMetadata y sus subrelaciones mmc:hasNote, mmc:hasTopic y mmc:hasResource, veremos que existe una limitación en la capacidad de inferencia, como se describe a continuación.

Para estudiar este problema se considera el siguiente ejemplo, consistente en cuatro entidades: una instancia de fragmento que tiene asociada una instancia de tema y nota a través de una relación con un tema.

```
:MyFragment
  rdf:type     mmc:Fragment;
  mmc:hasTopic :MyTopic1;
.

:MyTopic1
  mmc:hasTopic :MyTopic2;
  mmc:hasNote  :MyNote;
.
```

```
:MyTopic2   rdf:type        mmc:Topic.
:MyNote     rdf:type        mmc:Note.

:MyFragment mmc:hasTopic    :MyTopic2.
:MyFragment mmc:hasMetadata :MyNote.
```

Las ternas inferidas son resueltas a través de la transitividad de las relaciones usadas, sin embargo, no se infirió que :MyFragment se relaciona con :MyNote a través de mmc:hasNote, sino a través de mmc:hasMetadata. Para inferir esto se necesitaría una composición en mmc:hasNote (análogamente en mmc:hasTopic y mmc:hasResource) como se muestra a continuación, pero esto no es posible.

```
:hasNote owl:propertyChainAxiom (:hasTopic :hasNote);
```

Este hecho representa en cierta medida pérdida de información desde el punto de vista de los fragmentos, y cualquier entidad por encima de ellos en la estructura definida por la ontología, y se debe a la inexistencia de composiciones de tipos distintos de subrelaciones de mmc.hasMetadata.

Las implicancias que trae esta decisión de diseño son importantes, por ejemplo, es la responsable de que la siguiente consulta tenga un resultado vacío, a pesar de que se podría pensar que debería haber devuelto :MyNote.

```
SELECT ?note
WHERE {
   :MyFragment mmc:hasNote ?note.
}
```

```
-
```

A pesar de esto, es posible obtener todos los comentarios relacionados al fragmento, pero para ello se requiere una consulta un poco más sofisticada, como se muestra a continuación.

```
SELECT ?note
WHERE {
   :MyFragment mmc:hasMetadata ?note.
   ?note a mmc:Note.
}
```

```
:MyNote
```

Este hecho, lejos de ser un error de diseño, es una limitación dada por la propia especificación de OWL en la que restringe el uso de owl:propertyChainAxiom de forma de evitar problemas de decidibilidad [33].

Las restricciones dadas por el estándar prohíben el uso de composición de relaciones que resulten en relaciones *no regulares*, y para esto, se establece que siempre debe existir un orden estricto entre todas las relaciones definidas.

De este modo, basta analizar lo que sucede con nuestras relaciones mmc:hasTopic y mmc:hasNote y para esto se consideran dos de las composiciones faltantes en términos de inclusión.

$$\text{mmc:hasTopic} \circ \text{mmc:hasNote} \subseteq \text{mmc:hasNote}$$
$$\text{mmc:hasNote} \circ \text{mmc:hasTopic} \subseteq \text{mmc:hasTopic}$$

Como se puede apreciar, si se declararan estas composiciones dejaría de haber un orden entre las relaciones, por lo que la ontología definida de esta forma podría ocasionar resultados indecidibles, razón por la cual no es posible hacerlo.

4.6.2. Especificidad de las Relaciones

Otro problema existente es la incapacidad de relacionar metadatos de manera más específica que con mmc:hasMetadata. En efecto, solo se definieron las relaciones mínimas que permiten crear un grafo de metadatos, pero existen otras muchas relaciones que no fueron incluidas en la ontología. Por ejemplo, no se definió una relación permita explicitar qué un metadato es más específico que otro.

Esta decisión de diseño se basa en la necesidad de mantener la ontología simple. Incluirlas de manera exhaustiva hubiera significado agregar demasiadas relaciones nuevas, que a su vez hubiera resultado en un diseño más complejo y con un vocabulario mucho más amplio del necesario.

Si bien existe esta limitación, se debe aclarar que esto no tiene implicancias significativas ya que de ser necesario establecer relaciones más específicas entre metadatos en un futuro, existen dos posibilidades. En primer lugar, se puede extender la ontología agregando subrelaciones de mmc:hasMetadata, lo cual tiene un costo muy bajo. En segundo lugar, se puede utilizar tanto MMC como OFM en conjunto con SKOS debido a que estas ontologías fueron alineadas como se mostró anteriormente.

4.6.3. Integridad

La Web Semántica fue diseñada con tres principios muy importantes en mente que reflejan los problemas principales que se enfrentan al modelar información proveniente de Internet. En promer lugar, cualquiera puede decir cualquier cosa acerca de cualquier tema (*Anyone can say Anything about Any topic*, AAA), además no se asume que se cuenta con toda la información acerca de ningún recurso (open world assumption [34]) y se supone que URIs distintas pueden referenciar a las mismas entidades (non-unique name assumption).

Si bien esto era necesario para facilitar que esta tecnología pueda ser usada en la escala de Internet, trae importantes consecuencias prácticas y limita la capacidad de los triplestores para, entre otras cosas, mantener o forzar la integridad referencial o imponer un esquema rígido como se tiene en las bases de datos relacionales.

Los vocabularios de modelado RDFS y OWL fueron diseñados siguiendo estos lineamientos, de manera que el sistema sea robusto a información faltante o contradictoria. Cualquier otra restricción de integridad que se quiera imponer sobre los datos, salvo las pocas restricciones provistas por estos vocabularios, deben ser implementadas por el desarrollador.

Recientemente han aparecido vocabularios que permiten definir [35] ciertas restricciones adicionales, pero dado que aún no son recomendaciones de la W3C no ha sido implementado por muchos triplestores. Por otro lado, algunos triplestores particulares ofrecen mecanismos propios con los cuales se pueden definir algunas restricciones, pero esto trae otros problemas como el acoplamiento de la aplicación con un endpoint en particular, lo cual no es deseable como se comentará más adelante.

4.6.4. URIs

Además de no suponerse que URIs distintas necesariamente referencian a objetos distintos existen otras consideraciones importantes a tener en cuenta respecto a los identificadores de los distintos tipos de entidades que se quieren persistir.

Si bien las URIs funcionan simplemente como identificadores y en principio podrían ser cualquier cosa siempre que se respete el formato, es deseable que la entidad referenciada se vea reflejada, en algún sentido, de forma que incluso un humano pueda fácilmente reconocerla. Por lo tanto, consideraciones adicionales se deben tener en cuenta a la hora de diseñar las URIs que se usarán en el sistema. Por ejemplo, no se debería usar explícitamente el tipo de la entidad, dado que muchas veces pertenece a más de una clase al mismo tiempo.

En este trabajo se adoptó la siguiente convención, en donde cada sufijo es concatenado a “http://open.fing.edu.uy/data/” y cada variable (precedida por “$” por claridad) es sustituida por una propiedad de la entidad a la que se está referenciando.

ENTIDAD	SUFIJO
Universidad	$universidad
Facultad	$universidad/$facultad
Instituto	$universidad/$facultad/$instituto
Curso	$universidad/$facultad/$instituto/$curso
Clase	$universidad/$facultad/$instituto/$curso/$fecha
Video	$universidad/$facultad/$instituto/$curso/$fecha/$formato
Fragmento	$universidad/$facultad/$instituto/$curso/$fecha#t=$comienzo,$fin
Metadata	$universidad/$facultad/$instituto/$curso/$fecha#t=$comienzo,$fin&m=$timestamp
Usuario	user/md5($email)

Para las clases Universidad, Facultad, Instituto y Curso se tomará el valor de la propiedad *ofm:id*, la fecha de las clases tendrá el formato YYYYMMDD y el formato del video será la extensión del archivo. Es importante señalar que la URI del Fragmento, y por tanto de las

entidades de Metadata, son congruentes con la recomendación Media Fragments URI [36] de la W3C.

A continuación se muestra un ejemplo concreto de posibles URIs.

ENTIDAD	EJEMPLO
Universidad	http://open.fing.edu.uy/data/udelar
Facultad	http://open.fing.edu.uy/data/udelar/fing
Instituto	http://open.fing.edu.uy/data/udelar/fing/inco
Curso	http://open.fing.edu.uy/data/udelar/fing/inco/logica
Clase	http://open.fing.edu.uy/data/udelar/fing/inco/logica/20130630
Video	http://open.fing.edu.uy/data/udelar/fing/inco/logica/20130630/webm
Fragmento	http://open.fing.edu.uy/data/udelar/fing/inco/logica/20130630#t=534-1385
Metadata	http://open.fing.edu.uy/data/udelar/fing/inco/logica/20130630#t=534-1385&m=644742600
Usuario	http://open.fing.edu.uy/data/user/406612d2fe11eba6f91daa8a1b72604e

4.7. Ejemplo de Aplicación de la Ontología

En esta sección presenta un ejemplo construido utilizando MMC y OFA que describe una realidad concreta de un curso, una clase y fragmentos con metadatas relacionados a la misma.

Sin ánimo de cubrir todas las posibles formas en las que se pueden usar las dos ontologías diseñadas, este ejemplo es solo un vistazo que intentará presentar algunos tipos de inferencia básicos que se pueden dar y de este modo se busca validar el vocabulario para describir el dominio del problema. Por una versión más completa y exhaustiva del uso de los distintos elementos definidos en las ontologías y las posibles inferencias que se pueden realizar, referirse al Apéndice C.

En los recuadros blancos aparecen las ternas declaradas, y en los recuadros grises la información que es posible inferir a partir de la semántica de las ontologías usadas.

```
:LogicCourse
  rdf:type          owm:Course;
  ofm:hasLecture    :Introduction;
  ofm:hasProfessor  :John;
.
```

```
:LogiCourse    rdf:type          mmc:Collection.
```

```
:Introduction  rdf:type          ofm:Lecture.
:Introduction  rdf:type          mmc:Collection.

:John          rdf:type          ofm:Professor.
:John          rdf:type          mmc:Person.
:John          rdf:type          mmc:Agent.

:Introduction  ofm:isLectureOf   :LogicCourse.
:John          ofm:isProfessorOf :John.
```

En general las inferencias de tipo son debidas a que la instancia es declarada como de un tipo que es subclase de la clase inferida, o bien se usa un predicado que tiene a la clase inferida como su dominio o rango, según corresponda.

En el caso anterior aparece un ejemplo de estos dos tipos de inferencia. Por un lado :LogicCourse fue declarada como de tipo ofm:Course, la cual es sublcase de mmc:Collection, por lo que se infiere la primer terna. En segundo lugar se dice que el curso ofm:hasLecture :Introduction, y dado que esta relación tiene rango rango ofm:Lecture, se infiere la segunda terna, y a partir de esta la tercera como fue explicado al comienzo.

Por último, como es el caso para todos los predicados definidos en ambas ontologías, existen predicados inversos que son inferidos automáticamente. En el ejemplo anterior :LogicCourse se relaciona con :Introduction a través de ofm:hasLecture, por lo que también es válida la relación inversa utilizando ofm:isLectureOf.

```
:Introduction
  mmc:description "Introduction to the course";
  mmc:hasVideo    :Video01;
  mmc:image       "misc/video1.png";
.
```

```
:Video01      rdf:type      mmc:Video.
:Video01      rdf:type      mmc:Media.

:Video01      mmc:isVideoOf :Introduction.

:LogicCourse  mmc:hasVideo  :Video01.
:Video01      mmc:isVideoOf :LogicCourse.
```

En este caso se tiene un ejemplo de composición de relaciones. En la ontología, mmc:hasMedia y todos sus subrelacionas fueron declarados usando owl:propertyChainAxiom, lo cual permite especificar que es posible realizar ciertas composiciones para inferir una nueva relación de este tipo.

En particular, se tiene que mmc:hasVideo compuesto con mmc:hasComponent es una subrelación de mmc:hasVideo. Observando que mmc:hasLecture es a su vez subrelación de mmc:hasComponent se puede apreciar como cuando un curso, como :LogicCourose, se relaciona con una clase a través de mmc:hasLecture, como :Introduction, y dicha clase tiene un video, como :Video01, entonces es válido afirmar que el curso tiene ese video también.

```
:Syllabus
  rdf:type          mmc:Fragment;
  mmc:isFragmentOf  :Video01;
  mmc:hasNote       :CourseResources;
  mmc:start         85;
  mmc:finish        560;
.

:FirstTopic
  rdf:type          mmc:Fragment;
  mmc:isFragmentOf  :Video01;
  mmc:hasTopic      :FirstOrderLogic;
  mmc:hasResource   :Induction;
  mmc:start         683;
.
```

```
:Video01          mmc:hasFragment :Syllabus;
:Introduction     mmc:hasFragment :Syllabus;

:Video01          mmc:hasFragment :FirstTopic;
:Introduction     mmc:hasFragment :FirstTopic;

:CourseResources rdf:type         mmc:Note.
:CourseResources rdf:type         mmc:Metadata.

:FirstOrderLogic rdf:type         mmc:Note.
:FirstOrderLogic rdf:type         mmc:Metadata.

:Induction        rdf:type        mmc:Resource.
:Induction        rdf:type        mmc:Metadata.

:Video01          mmc:hasResource :Induction.
:Introduction     mmc:hasResource :Induction.
:LogicCourse      mmc:hasResource :Induction.

:Video01          mmc:hasTopic    :FirstOrderLogic.
:Introduction     mmc:hasTopic    :FirstOrderLogic.
:LogicCourse      mmc:hasTopic    :FirstOrderLogic.

:Video01          mmc:hasNote     :CourseResources.
```

```
:Introduction    mmc:hasNote     :CourseResources.
:LogicCourse     mmc:hasNote     :CourseResources.

:Video01         mmc:hasMetadata :Induction.
:Introduction    mmc:hasMetadata :Induction.
:LogicCourse     mmc:hasMetadata :Induction.

:Video01         mmc:hasMetadata :FirstOrderLogic.
:Introduction    mmc:hasMetadata :FirstOrderLogic.
:LogicCourse     mmc:hasMetadata :FirstOrderLogic.

:Video01         mmc:hasMetadata :CourseResources.
:Introduction    mmc:hasMetadata :CourseResources.
:LogicCourse     mmc:hasMetadata :CourseResources.
```

Si bien en este último caso se infiere mucha información, ninguna de estas inferencias escapa a los casos de ya mencionados anteriormente (inferencias de tipo por dominios y rangos y composición de relaciones). Sin embargo, era importante la creación de estos fragmentos, ya que estas instancias hacen el nexo entre dos de los componentes más importantes de la ontología: los recursos multimedia y los metadatos.

A modo de ejemplo, las siguientes ternas muestran instancias concretas de metadatos creados por usuarios, como se infirió anteriormente, las instancias corresponden a un tema, un comentario y un recurso, respectivamente.

```
:FirstOrderLogic
  mmc:text       "First order logic";
  mmc:created    "2013-01-20"^^xsd:date;
.

:CourseResources
  mmc:text       "Check the course's book for more details";
  mmc:created    "2013-02-21"^^xsd:date;
.

:Induction
  mmc:text       "http://logic.com/induction";
  mmc:created    "2013-03-22"^^xsd:date;
.
```

De esta forma se ha creado el curso :LogicCourse, se le ha asociado la clase :Introduction y a ella el video :Video01. Estos datos comprenden lo que será la información básica a publicar para cada clase, y sobre ello los usuarios agregarán fragmentos como :Syllabus y :Induction, a los que asociarán metadatos como :CourseResources, :FirstOrderLogic e :Induction.

A partir de este caso de uso se podrían realizar consultas SPARQL que serían resueltas utilizando las las ternas declaradas explícitamente y toda la información inferida a partir de la semántica de los vocabularios usados.

Por ejemplo, se pueden obtener todos los profesores existentes en el sistema junto a los cursos que dictan.

```
SELECT ?professor ?course
WHERE {
  ?course a ofm:Professor.
  OPTIONAL {?professor ofm:isProfessorOf ?course.}
}
```

```
:John :LogicCourse
```

Mediante la siguiente consulta se pueden obtener todas las clases y las posiciones de comienzo y fin de sus fragmentos, junto al texto del metadato asociado.

```
SELECT ?lecture ?start ?finish ?text
WHERE {
  ?lecture a ofm:Lecture.
  ?lecture mmc:hasFragment ?fragment.
  ?fragment mmc:hasMetadata ?metadata.
  ?metadata mmc:text ?text.
  OPTIONAL { ?fragment mmc:start ?start }
  OPTIONAL { ?fragment mmc:finish ?finish }
}
```

```
:Indroduction  85 560 "Check the course's book for more details"
:Indroduction 683  -  "First order logic"
:Indroduction 683  -  "http://logic.com/induction"
```

CAPÍTULO 5: IMPLEMENTACIÓN DEL PROTOTIPO

En este capítulo se describirán los aspectos más importantes del prototipo desarrollado. Se comenzará por la descripción de una arquitectura teórica de la plataforma y fundamentos relacionados a la elección de las tecnologías usadas en cada uno de los módulos e la misma, para luego pasar a aspectos más técnicos de la implementación.

5.1. Arquitectura

La arquitectura del sistema en gran medida fue diseñada para ser simple y a la vez genérica, y sigue algunos de los componentes básicos descritos en [37]. Por esta razón no solo describe los componentes principales del sistema que se está desarrollado, sino también la estructura de cualquier aplicación Web que adopte de alguna manera un modelado de datos a nivel semántico.

Cabe señalar que el diseño realizado es una guía que deberá ser considerada en la versión final de la plataforma a construir. Dado el alcance del prototipo desarrollado en este trabajo, solo se considerarán los componentes relevantes.

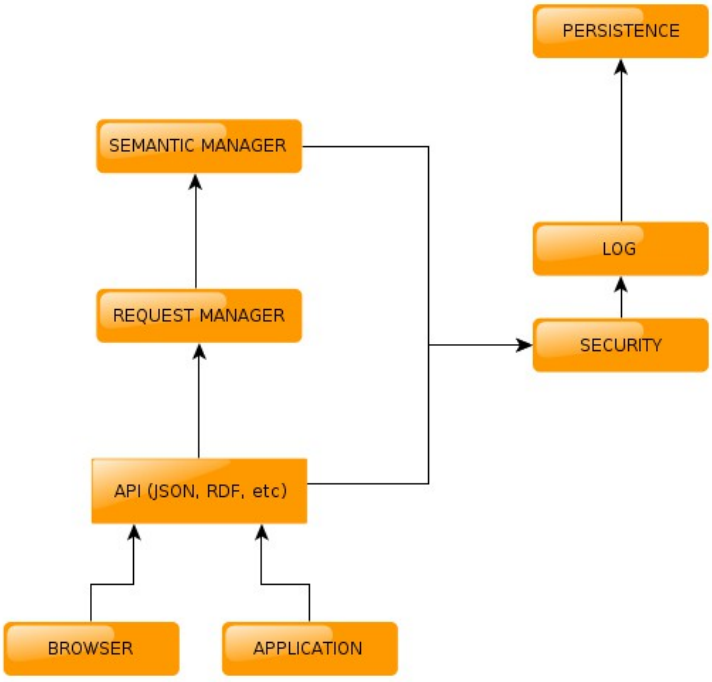


Figura 6: Arquitectura de la plataforma

A continuación se describe cada uno de los módulos que integran la arquitectura.

- BROWSER : Navegador Web usado por usuarios finales para interactuar con el sistema mediante una interfaz amigable, a través de una API RESTful sobre HTTP.

- APPLICATION: Esta entidad representa a cualquier aplicación, desarrollada como parte del sistema o bien por terceros, que se comunica con la plataforma a través de alguna de las interfaces ofrecidas, posiblemente intercambiando strings JSON con el servidor haciendo uso de la API RESTful sobre HTTP o bien comunicándose directamente con la base de datos a través de un protocolo de comunicación.

- API: Interfaces brindadas por el sistema para que usuarios y aplicaciones interactúen con él. Si bien la API va a ser una sola, se espera que soporte varios formatos de consulta y salida como HTML, JSON, RDF, etc.

- REQUEST MANAGER: Módulo del sistema responsable de recibir y atender las consultas realizadas por los usuarios. Su responsabilidad consiste básicamente en ser un puente entre una consulta simple a la API y el manejador semántico. Una vez recibida una petición, deberá validar la entrada y llevar a cabo un primer nivel de procesamiento que permita transformar el request en algo más elaborado y estructurado, para luego desencadenar el conjunto de llamadas necesarias para responder el pedido.

- SEMANTIC MANAGER: Entidad responsable de la comunicación con la base de datos. Deberá ofrecer una interfaz al request manager que oculte todos los detalles del modelado de datos, logrando así centralizar las consultas y updates realizados.

- SECURITY: Presta un servicio de seguridad, encargado tanto de la autenticación de los usuarios y la propia aplicación, como la autorización de los mismos para realizar ciertas operaciones.

- LOG: Módulo responsable de registrar el tráfico desde la aplicación hacia la base de datos, con el objetivo de obtener estadísticas de uso del sistema asociadas a los distintos usuarios.

- PERSISTENCE: Representa la persistencia del sistema. Contendrá por tanto la base de datos y archivos estáticos como imágenes y videos que serán usados por la interfaz Web.

5.2. Tecnología y Herramientas

5.2.1. Back-end

Existen decenas de lenguajes de programación que podrían ser usados del lado del servidor en el desarrollo del prototipo, entre muchos otros se destacan PHP[15], Python[16] y Ruby[17] ya que son posiblemente los tres más conocidos en el contexto del desarrollo Web.

Si bien estos tres lenguajes ofrecen básicamente las mismas características desde muchos puntos de vista, se diferencian principalmente en los paradigmas que representan, la comunidad que los sostiene y da soporte y las herramientas que fueron creadas para los mismos.

En este sentido, los tres lenguajes son similares, quizás sobresaliendo particularmente Python y Ruby por su enfoque híbrido: orientación a objetos y programación funcional, lo cual trae ventajas desde el punto de vista de legibilidad de código.

En los últimos años Ruby ha ganado la aceptación de muchos desarrolladores gracias al framework Ruby on Rails[18] (ROR), sobre el cual se han inspirado muchos otros frameworks de distintos lenguajes. Este framework, que sigue la arquitectura de desarrollo MVC [38], presenta muchas características innovadoras y un paradigma de *convención sobre configuración,* además de una amplia comunidad que da soporte a sus usuarios mediante mailing lists, grupos de desarrolladores regionales e incluso salas de soporte técnico en tiempo real.

Por otro lado, existen repositorios como Ruby Toolbox[19] con más de 100.000 bibliotecas, denominadas *gemas*, desarrolladas para este lenguaje, por lo que resulta difícil no encontrar alguna que se adapte a lo que buscamos, cualquiera sean los requerimientos.

Por estas razones se decide utilizar Ruby y Ruby on Rails en el servidor, ya que como se discutirá más adelante, en una búsqueda muy rápida acerca de herramientas desarrolladas en Ruby para la Web Semántica se encontraron varias que resultaron ser muy buenas.

5.2.2. Front-end

A diferencia del back-end, existe consenso en cuáles deben ser los lenguajes y estándares utilizados para el front-end, entre ellos se destaca HTML5, CSS3 y JavaScript como tecnologías aceptadas por la industria e implementadas por todos los navegadores. Sin más razones, serán estos los lenguajes usados en el front-end.

15 http://php.net/
16 http://www.python.org/
17 https://www.ruby-lang.org/
18 http://rubyonrails.org/
19 https://www.ruby-toolbox.com/

5.2.3. Persistencia

Si bien la Web Semántica es una tendencia relativamente joven, en la actualidad existen muchos manejadores de bases de datos RDF, denominados *triplestores*, cada uno de ellos con características y prestaciones.

Entre los más conocidos se encuentra TDB[20], Stardog y Virtuoso. El primero de ellos, TDB, es un proyecto de código abierto por lo que su utilización es gratuita, mientras que los últimos dos son desarrollos cerrados aunque para ambos casos existen versiones comunitarias gratuitas cuyas limitaciones están, de momento, muy por encima a los requerimientos de capacidad de almacenamiento que se tienen en este trabajo.

Si analizamos nuevamente los resultados esperados y en particular los requerimientos relevados, resulta evidente la necesidad de que el triplestore se pueda integrar con un razonador que explote la semántica de la ontología, y con un motor de búsqueda de texto que implemente técnicas denominadas comúnmente como *full-text-search [39]*.

El primero de estos requerimientos es soportado por TDB, ya que es posible utilizarlo a través del framework Jena, el cual a su vez se puede integrar con Pellet, uno de los razonadores más avanzados y completos en la industria. Del mismo modo, Stardog trae integrado dos motores de inferencia, uno de los cuales es Pellet ya que ambos productos son desarrollados en la misma empresa. Por otro lado, Virtuoso si bien es completo en muchos otros aspectos, no ofrece una solución al problema del razonamiento, viéndose incluso limitada la posibilidad de integrarlo con razonadores externos como Pellet ya que los drivers necesarios no fueron mantenidos en mucho tiempo, razón por la cual fue descartado.

En referencia al segundo requerimiento, existe un proyecto llamado Jena-Text[21] que, como su nombre lo indica, es un módulo de Jena que agrega soporte a full text search utilizando internamente el motor de búsqueda de texto Lucene[22], uno de los más conocidos para estas tareas. A pesar de esto la versión de Jena que es compatible con Pellet no lo es con Jena-Text por lo que se descarta esta solución también. Afortunadamente Stardog también soporta nativamente full-text-search a través de Solr[23], un servidor de búsqueda basado en Lucene.

Por las razones mencionadas anteriormente, de las soluciones consideradas el único triplestore que cumple con las necesidades básicas del sistema es Stardog, por lo que se opta por su utilización.

Antes de concluir este análisis se debe aclarar que en un primer momento y durante gran parte del desarrollo del prototipo se utilizó el servidor SPARQL Fuseki[24], el cual permite integrar

20 http://jena.apache.org/documentation/tdb/
21 http://jena.apache.org/documentation/query/text-query.html
22 http://lucene.apache.org/
23 http://lucene.apache.org/solr/
24 http://jena.apache.org/documentation/serving_data/

fácilmente Jena, TDB y Pellet. Avanzado el desarrollo se vio la necesidad de agregar el motor de búsqueda, y fue ahí en donde se encontró la incompatibilidad de versiones en los productos, por lo que se decidió rápidamente migrar a Stardog. Este cambio de tecnología ocasionó retrasos en el cronograma pero debido a una decisión de diseño que se discute a continuación, fue posible adoptar el desarrollo sin demasiadas complicaciones.

5.3. Capa de Modelos

Debido a la cantidad de triplestores existentes y dado que sus características y capacidades eran poco conocidas, se decidió desde un primer momento adoptar una estrategia preventiva que permitiera cambiar la tecnología subyacente sin tener que realizar muchos cambios a la aplicación.

Para esto se optó por la utilización de Tripod[25], un ORM para Ruby que abstrae la comunicación con la base de datos utilizando un protocolo de comunicación HTTP con el servidor SPARQL, lo que disminuye el acoplamiento entre la aplicación Web y la base de datos.

Debido a esta decisión, como fue mencionado antes, fue posible pasar de usar Fuseki a usar Stardog con relativamente pocos cambios en el código ya implementado. Sin embargo, la migración implicó la modificación de Tripod ya que la misma no soportaba que autentificación contra el servidor SPARQL, lo cual es obligatorio en Stardog. Adicionalmente, fue necesario modificar levemente las consultas que se generan automáticamente en la biblioteca, ya que a partir de SPARQL 1.1 es necesario que las variables estén tipadas, de lo contrario el resultado es impredecible.

Este bajo acoplamiento de la aplicación con la base de datos se logra gracias a una abstracción de los conceptos definidos en la ontología en lo que se conoce como *modelos* en el paradigma MVC. Esta capa es particularmente importante ya que de ella depende todo el flujo de información desde y hacia la base de datos.

A modo de ejemplo a continuación se muestra el modelo que abstrae la clase mmc:Video.

```
class Video

  rdf_type type(mmc: "Video")

  field :filename, uri(mmc: "filename"), datatype: RDF::XSD.string
  field :format,   uri(mmc: "format"),   datatype: RDF::XSD.string
  field :size,     uri(mmc: "size"),     datatype: RDF::XSD.integer
  field :created,  uri(mmc: "created"),  datatype: RDF::XSD.date
  field :width,    uri(mmc: "width"),    datatype: RDF::XSD.integer
  field :height,   uri(mmc: "height"),   datatype: RDF::XSD.integer
```

25 https://github.com/Swirrl/tripod

```
  field :duration, uri(mmc: "duration"), datatype: RDF::XSD.integer

  linked_to :lecture, uri(of: "isVideoOf"), class_name: 'Lecture'

  linked_to :persons, uri(of: "wasWatchedBy"), multivalued: true,
class_name: 'Person'

end
```

En una breve descripción se puede decir que los modelos en RoR con Tripod son clases instanciables de Ruby que tienen asociado un tipo, correspondiente a rdf:type, propiedades definidas a través de la función *field*, para las cuales se especifica además su rango, y finalmente ciertas relaciones con otras clases, multivaluadas o no, establecidas a través de *linked_to* y *linked_from*. Tanto para las propiedades como para las relaciones es indispensable especificar la URI que fue usada en la ontología, de forma de que al crear una nueva instancia de la relación se pueda construir la terna correcta.

La forma de uso de estos modelos son una prueba clara del nivel de abstracción que se logra de esta forma, para apreciar esto seguidamente se muestra un ejemplo mediante el cual se crea una clase y se le asocia un video.

```
lecture = Lecture.new "http://example.com/data#lecture01"
lecture.name = "Lecture 01"
lecture.save

video = Video.new "http://example.com/data#video01"
video.filename = "videos/video01.mp4"
video.created  = Time.now
video.duration = 4980
video.lecture  = lecture
video.save
```

5.4. Implementación

5.4.1. Secciones

El prototipo construido cuenta con varias secciones que son presentadas en el Apéndice D. Las mismas ofrecen información de distintos tipos de entidades, permitiendo además navegar entre ellas de acuerdo a las relaciones establecidas en la ontología.

Entre las secciones implementadas se encuentran, además de la página principal, una sección que describe el proyecto OpenFING, la página principal de los institutos, profesores y cursos, una sección en donde se puede ver una clase y agregar anotaciones y la página de contacto.

5.4.2. Busqueda

El módulo de búsqueda es el encargado de brindar un mecanismo que permite obtener cualquier entidad relacionada directa o indirectamente con una consulta dada en lenguaje natural. Esto presenta distintas dificultades, quizás la más evidente sea la brecha existente entre el nivel de abstracción de la consulta ingresada por el usuario y lo que es posible expresar mediante una consulta SPARQL.

Sin embargo, existen mecanismos alternativos que explotan el uso de keywords, como el presentado en [40], pero en última instancia todos ellos presentan las mismas limitaciones. En efecto, SPARQL solo ofrece búsqueda exacta de texto o a partir de expresiones regulares. Esto no resulta ser suficientemente expresivo y sería muy difícil resolver cualquier consulta interesante. Por ejemplo, ante “inducción completa” se esperaría encontrar también resultados acerca de “inductividad” y “conjuntos inductivos”. Hacer esto a través de búsqueda exacta no es posible, y crear una expresión regular para tal cometido resulta impracticable ya que se deberían tener en cuenta todas las permutaciones de los términos ingresados, aún si no se consideran variaciones ocasionadas por errores de tipeo.

El mecanismo más adecuado para este caso de uso parece comprender un conjunto de técnicas conocidas como full-text-search. Como se mencionó anteriormente, una de las razones por la cual se eligió Stardog es por contar con soporte nativo para esto, facilitando enormemente la tarea.

Stardog soporta el predicado *textMatch* provisto por Jena para este tipo de búsqueda, pudiendo a partir de él especificar cualquier criterio utilizando la sintaxis de Lucene. La única construcción no soportada por Stardog son los campos (fields), pero como se verá a continuación esto no será necesario.

A modo de ejemplo se considera la siguiente consulta SPARQL, en donde se obtienen todos los sujetos *?s* relacionados a través de una propiedad *?p* con el texto *?text.* Dicho literal debe

aparear con el criterio “inducción~ completa~”. Además, se especificó un threshold de 0.5 para eliminar cualquier resultado espurio y un límite de 50 resultados para el motor de búsqueda.

```
SELECT DISTINCT ?s ?text
WHERE {
  ?s ?p ?text.
  ( ?text ?score ) jena:textMatch ( "inducción~" 0.5 50 ).
}
```

Los literales que cumplen con el criterio dado contienen el término “inducción” o uno similar con distancia de edición menor o igual a dos.

En la práctica la consulta generada es más elaborada y se parece a la siguiente, en donde se dejaron dos parámetros, #type y #criteria, que son sustituidos por Ruby antes de ejecutarla en el endpoint SPARQL.

```
SELECT DISTINCT ?graph ?uri ?type
WHERE {
  GRAPH ?graph {
    #type.
    ( ?index ?score ) jena:textMatch ( '#criteria' 0.1 ).
    {
      ?uri mmc:index ?index.
      BIND(0 AS ?distance).
    } UNION {
      ?uri mmc:hasFragment ?f.
      ?f   mmc:hasTopic    ?m.
      ?m   mmc:index       ?index.
      BIND(1 AS ?distance).
    } UNION {
      ?uri mmc:hasTopic ?m.
      ?m   mmc:index    ?index.
      BIND(2 AS ?distance).
    }
  }
}
ORDER BY DESC(?score) ASC(?distance)
```

Se puede apreciar como la consulta está dividida en dos uniones. En primer lugar se obtienen las entidades que tienen relación directa con la consulta ingresada. En segundo lugar se obtienen las entidades que se relacionan a través de un fragmento. En tercer lugar se obtienen las entidades que se relacionan a través de un tema, pudiendo éste estar separado por varios niveles de entidades, es decir, no estar relacionado directamente a través de un fragmento, sino a través de una curso que tiene una clase que tiene un fragmento, por ejemplo.

Es importante notar el uso de *mmc:index*. Esta propiedad es creada por el sistema para distintos tipos de instancias: mmc:Organization, ofm:Professor, ofm:Course, ofm:Lecture y mmc:Topic. El contenido de la misma depende de la entidad sobre la que se la use, por ejemplo para anotaciones de tipo *tema* mmc:index contiene el texto ingresado por el usuario preprocesado, para los demás tipos de instancias se usa en su lugar su nombre o descripción. El preprocesamiento que se realiza es idéntico al descrito en la sección 5.4.2.2. para las consultas ingresadas en el buscador, sin embargo para esta propiedad no se utiliza la sintaxis de Lucene, simplemente se usa una secuencia de términos y raíces.

Los resultados de las tres uniones son ordenados primero de acuerdo a la similitud del texto apareado con el criterio de búsqueda y en segundo lugar de acuerdo a la distancia entre la entidad devuelta y la entidad que tiene dicho texto.

La variable *#criteria* es sustituida por la consulta ingresada por el usuario, a la cual previamente se le aplica un preprocesador como se describe más adelante. La variable #type es sustituida por un filtro que permite especificar los tipos de entidades en los que se está interesado.

En la próxima sección se describe el lenguaje de consulta con todas las construcciones soportadas por el motor de búsqueda implementado.

5.4.2.1. Lenguaje de Consulta

Dado que el motor de búsqueda de la plataforma es uno de los componentes principales del sistema se optó por permitir al usuario usar algunas construcciones propias de Lucene, lo cual se espera que le ofrezca mayor flexibilidad y expresividad.

La consulta se puede ver como un prefijo, su tipo, seguido de un criterio de búsqueda. A su vez, el criterio ingresado estará compuesto por términos o frases que deberán aparecer preferiblemente en el texto apareado pero esto no será indispensable al menos que se indique lo contrario mediante un operador. De este modo los espacios funcionan de alguna manera como OR lógicos.

Por defecto las búsquedas devuelven entidades de los siguientes tipos: universidad, facultad, instituto, profesor, curso, clase y fragmento, pero este comportamiento se puede alterar tipando los resultados. Esto se logra incluyendo un prefijo en la consulta que debe estar formado por el nombres del tipo esperado (en plural o singular) seguido por ":". Por ejemplo, la consulta "fragmento: conjuntos" es equivalente a la consulta "conjuntos" en donde las únicas entidades devueltas son de tipo fragmento.

Entre los símbolos permitidos en el criterio de búsqueda se encuentran los caracteres alfanuméricos, cualquiera de los diferentes tipos de caracteres espaciadores, dos tipos de comodines y dos operadores unarios.

La semántica de cada uno de estos comodines y operadores se detalla a continuación.

- Comodín “?”: este comodín representa un único caracter. Por ejemplo, para buscar “computadora” y “computadoras” se puede usar “computadora?”.

- Comodín “*”: este comodín representa 0 o más caracteres. Por ejemplo, para buscar por “algoritmo”, “algoritmos” y “algoritmia” se puede usar “algoritm*”.

- Operador “+”: este operador indica que el término o frase siguiente debe aparecer en el texto apareado. Por ejemplo, para buscar resultados en donde aparezca preferiblemente “programación” pero necesariamente deba aparecer “lógica”, se puede usar “programación +logica”.

- Operador “-”: este operador indica que el término o frase siguiente no debe aparecer en el texto apareado. Por ejemplo, para buscar resultados en donde aparezca preferiblemente “programación” pero en donde no deba aparecer “lógica”, se puede usar “programación -lógica”.

Los operadores descritos actúan sobre términos simples o sobre conjuntos de términos, denominados frases. Para especificar que un conjunto de términos es una frase basta con envolverlos entre comillas dobles. Por ejemplo, “ +"recta numérica" ” indica que debe necesariamente aparecer el string “recta numérica” en el texto apareado.

Existe otro uso para las comillas dobles y es el de demarcar frases que no deben ser expandidas. Para entender que significa esto es necesario primero mencionar que por defecto todo substring de caracteres alfanuméricos es expandido utilizando el preprocesador descrito en la próxima sección, para evitar este comportamiento se deben utilizar comillas dobles.

5.4.2.2. Expansión de Términos

La expansión consiste en aplicar una transformación a los términos (no frases) del criterio de búsqueda, similares a las propuestas en [41], que incorpore las reíces de las mismas y permita así aumentar las posibilidades de encontrar entidades relevante. De acuerdo al lenguaje de consulta descrito en la sección anterior, solo se expanden substrings alfanuméricas que no aparezcan en la consulta entre comillas dobles.

Para esto, se desarrolló un servicio que consume una oración en español y mediante técnicas de procesamiento de lenguaje natural devuelve como resultado un criterio de acuerdo a la sintaxis de Lucene.

El proceso de transformación se detalla a continuación.

1. En primer lugar se encuentra la etiqueta gramatical (sustantivo, adjetivo, verbo, artículo, etc) de cada palabra a partir de un tagger, conocido como Brill Tagger [42], que internamente considera bigramas (pares consecutivos de palabras).

2. Se elimina cualquier palabra demasiado frecuente, conocidas como *stop-words*, aquellas cuyo largo no supere un threshold dado (por defecto se asignó en cuatro letras) o que no tengan una etiqueta gramatical de adjetivo, sustantivo o verbo.

3. Para cada término no filtrado de la consulta, se obtiene su raíz mediante un *stemmer*, conocido como Snowball [43].

4. Se combinan todas estas palabras mediante ORs lógicos, para cada una se utiliza el operador de búsqueda difusa "~", y se le asigna un peso a través del operador de *boosting* "^".

El siguiente es el resultado devuelto para la consulta "conjuntos definidos inductivamente".

```
(conjuntos~^10 OR conjunt~^5) OR (definidos~^10 OR defin~^5) OR (inductivamente~^10 OR induct~^5)
```

Cabe mencionar que esto fue implementado usando Python, ya que muchas de las herramientas utilizadas provienen de NLTK[26], una plataforma de procesamiento del lenguaje natural escrita en dicho lenguaje.

Si bien este servicio de procesamiento y transformación de la consulta se encuentra lejos de ser óptimo, la comunicación con el resto del proyecto se da a través de un socket TCP. Esto permite mantener un nivel mínimo de acoplamiento, pudiéndose eventualmente cambiar el servicio por uno más elaborado en el futuro sin impactar de ninguna manera en el resto de la aplicación.

5.4.2.3. Limitaciones

En un primer momento se intentó dar un paso más en el procesamiento de la consulta, e incorporar sinónimos e hiperónimos para cada término ingresado a través de una versión offline de Wordnet [44].

Si bien esto resulta ser una buena idea, en la práctica se agrega mucho ruido que termina degradando la calidad de los resultados obtenidos. Por ejemplo, para la consulta "base de datos" se ingresaba el término "escolta".

26 http://nltk.org/

Esto se podría mitigar agregando un desambiguador de significado que permita decidir cuáles sinónimos se deben agregar a la consulta y cuales no. Desafortunadamente este es un problema abierto y no existe este recurso para español en NLTK.

Las implicaciones de este problema podrían ser notorias. Por ejemplo, el sistema no sería capaz de encontrar el tema "reconocimiento de patrones" a partir de un término como "detección", a menos de que éste venga acompañado también de "patrones".

5.4.3. Relacionamiento de Contenido

A medida que un usuario mira un video, el navegador consulta al servidor en busca de clases y fragmentos relacionados con la anotación más relevante en ese momento, si ésta es de tipo *tema*. Por su parte el servidor consulta la base de datos, y devuelve los fragmentos más relevantes, que serán ofrecidos al usuario como recursos relacionados.

Este proceso de búsqueda en gran medida se asemeja al proceso de búsqueda ya presentado, por lo que la consulta generada para obtener dichos recursos también es muy parecida, presentando solo algunas diferencias que son mencionadas a continuación.

```
SELECT DISTINCT ?graph ?uri ?type
WHERE {
  GRAPH ?graph {
    ?uri stardog:directType ?type.
    ( ?index ?score ) jena:textMatch ( '#criteria' 0.1 ).
    {
      ?l   mmc:hasFragment ?uri.
      ?uri rdf:type        mmc:Fragment.
      ?uri mmc:hasTopic    ?t.
      ?t   mmc:index       ?index.
      FILTER (?l != #filter)
      BIND(0 AS ?distance).
    } UNION {
      ?uri rdf:type  mmc:Lecture.
      ?uri mmc:index ?index.
      FILTER (?uri != #filter)
      BIND(1 AS ?distance).
    }
  }
}
ORDER BY DESC(?score) ASC(?distance)
```

A diferencia del caso anterior, en este caso la consulta consiste en una sola unión de dos patrones de grafo. Mediante el primer patrón se obtienen todos los fragmentos que tienen un

tema que satisface el criterio de búsqueda y a partir del segundo patrón se obtienen todas las clases relacionadas a través de una de sus propiedades. En ambos casos, se filtran los resultados de manera que no se devuelvan aquellos que son o pertenecen a la clase que el usuario está mirando.

En referencia al criterio de búsqueda, en primera instancia se utiliza la consulta para encontrar recursos relacionados a partir de la expansión de términos del nombre y la descripción de la clase correspondiente que se va a mostrar. Este conjunto de recursos es utilizado por defecto, es decir, en caso de que no haya ninguna otra clase o fragmento más relevante a la posición en la que se encuentra el video.

A medida que avanza el video, se elige la anotación de tipo *tema* más relevante en esa posición y se consulta al servidor para obtener recursos relacionados a la misma. En este caso el criterio de búsqueda es la expansión de términos del texto asociado a la anotación en cuestión.

CAPÍTULO 6: CASO DE ESTUDIO

En este capítulo se presentan tres casos de estudio de usuarios que interactúan con el sistema de diferentes formas. En primer lugar, dos estudiantes que lo hacen a través de la interfaz Web creada y pretende estudiar o repasar conceptos relacionados con "inducción" de una clase de Cálculo 1. En segundo lugar, un usuario utiliza la información publicada directamente a través del endpoind SPARQL para extraer información útil para una aplicación externa que se encuentra desarrollando. Por último, se mostrará como es posible importar la información generada en esta plataforma desde triplestores externos.

6.1. Usuario Final

Este caso de uso será el más frecuente, y consiste en un estudiante que, ayudado por los videos publicados, se encuentra preparando un parcial o examen de un curso, como Lógica.

En primer lugar el estudiante ingresa al sistema, utilizando para esto sus credenciales en la Universidad de la República, las cuales son verificadas en el servidor de EVA de la Facultad de Ingeniería.

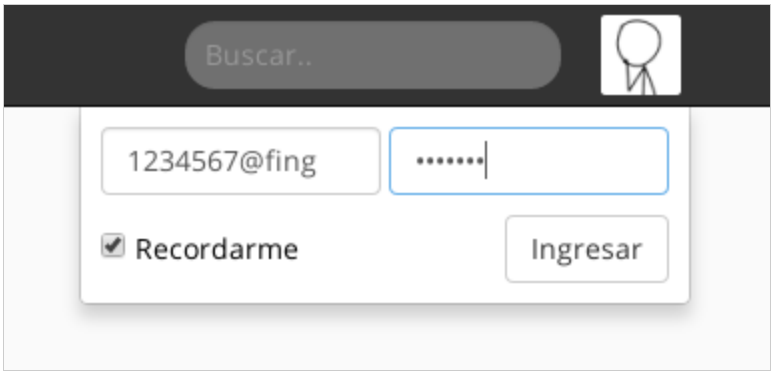


Figura 7: El estudiante ingresa al sistema

Seguidamente utilizando el buscador ingresa una consulta de su interés, como "inductividad", obteniendo como resultado una clase entera de Lógica, ya que en su nombre y descripción aparece un concepto relacionado: "conjuntos definidos inductivamente".

Figura 8: Resultado de la búsqueda "inductividad"

De este modo el estudiante comienza a mirar la clase, buscando el fragmento exacto en el que se trata ese tema.

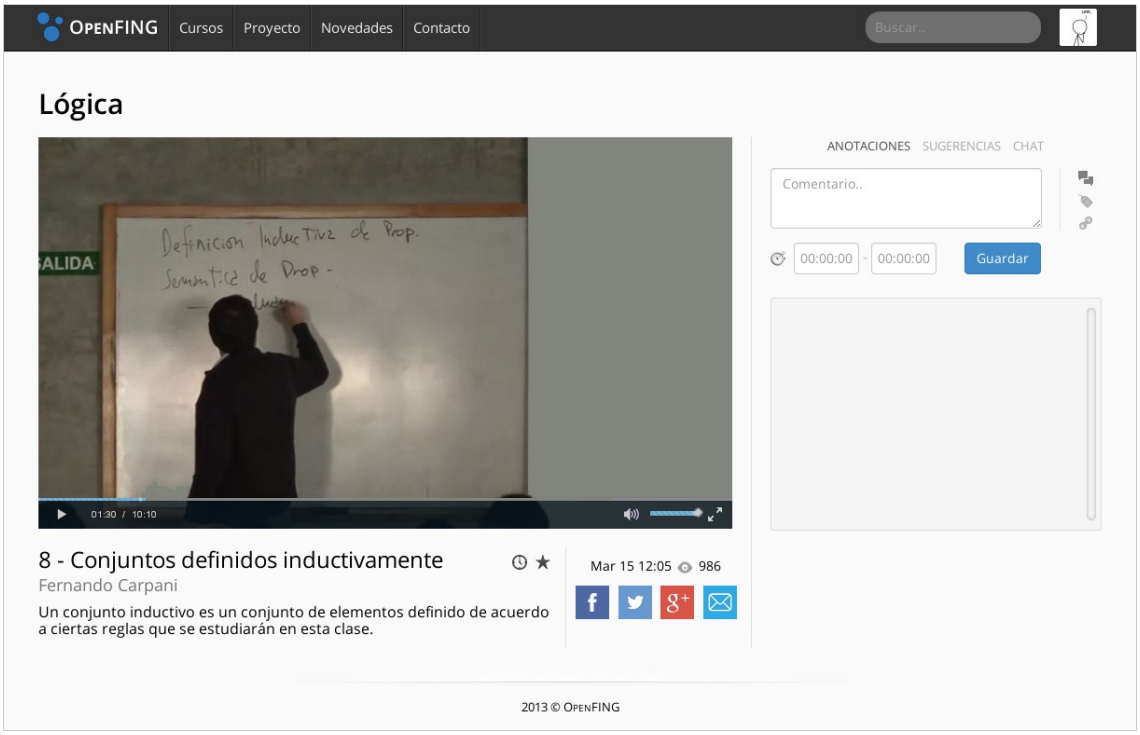


Figura 9: Vista de una clase

Una vez encuentra el fragmento que buscaba, decide anotarlo para que él mismo en otra ocasión, u otro estudiante, no tenga que invertir tiempo en mirar todo el video si solo está interesado en estudiar inducción. Para esto, crea un tema asociado a un intervalo de tiempo de pocos minutos y le asigna el texto “Conjuntos inductivos”.

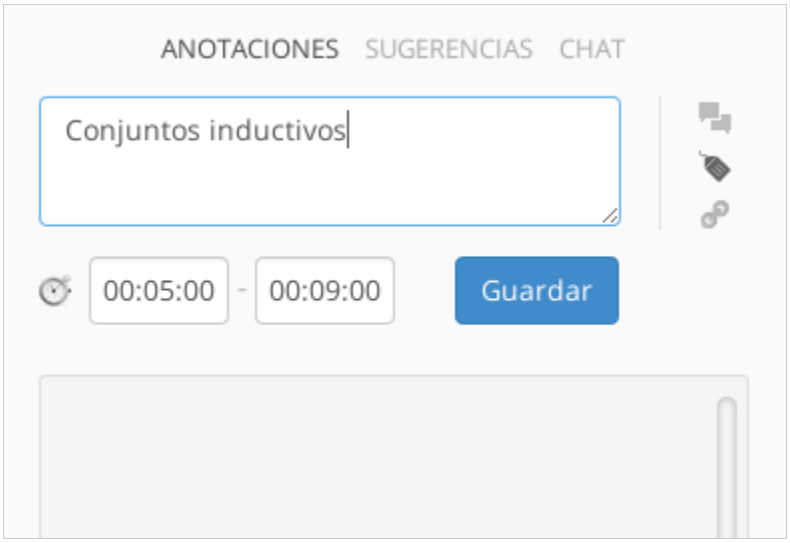


Figura 10: Creación de un fragmento

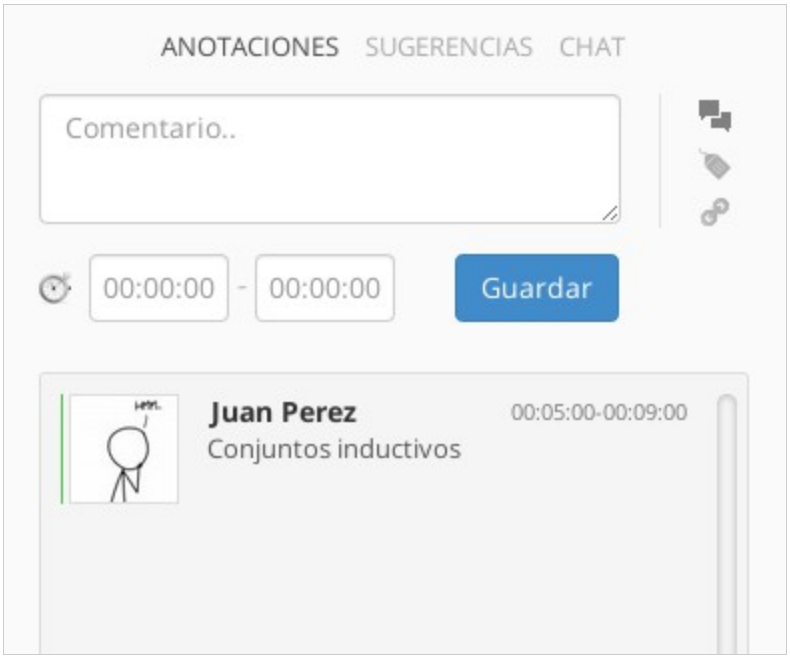


Figura 11: Fragmento creado

Dado que luego de mirar el contenido del mismo le queda una duda respecto a este tema, crea un comentario en el que consulta a otros usuarios del sistema. Para aumentar las posibilidades de que alguien responda su pregunta, utiliza la URL del fragmento y hace referencia a ella desde EVA.

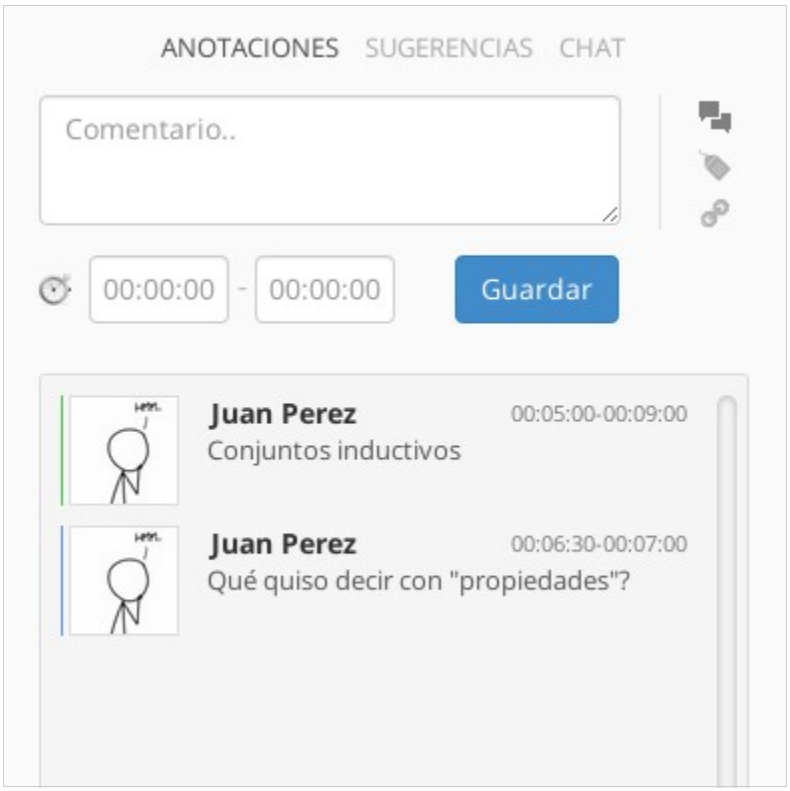


Figura 12: Creación de un comentario

Figura 13: Referencia a un fragmento

Por otro lado, un segundo estudiante también preparando el parcial o examen del mismo curso ingresa al sistema en busca del mismo tema.

Una vez más, la plataforma recibe la consulta “inductividad”, pero en esta ocasión además de los resultados devueltos anteriormente, el sistema le ofrece un fragmento de pocos minutos a partir de la anotación creada.

Figura 14: Resultados de la búsqueda "inductividad"

Mientras mira este fragmento, se le sugiere otro de Cálculo 1, anotado previamente como "Ejemplos de conjuntos definidos inductivamente" en una clase titulada "Inducción primitiva".

Figura 15: Fragmento sugerido

6.2. Protocolo HTTP

En este caso de uso el usuario es un desarrollador que se encuentra creando una versión móvil de la plataforma.

Dicha aplicación puede verse simplemente como otra interfaz de comunicación con la base de datos, y por tanto debería limitarse a consumir los datos ofrecidos por la misma a través del endpoint SPARQL.

Esta comunicación tiene lugar mediante consultas HTTP tradicionales como se muestra en la Figura 16, en la que se hace uso de cURL[27] para extraer los cursos existentes en el sistema.

```
mparodi# curl -G "http://mparodi.wildgeeks.com:5820/openfing/query" \
>   -H "SD-Connection-String: reasoning=SL" \
>   -H "Accept: application/sparql-results+json" \
>   --data-urlencode query="
dquote>     SELECT ?course
dquote>     WHERE {
dquote>       GRAPH ?g {
dquote>         ?course a <http://open.fing.edu.uy/ofm#Course>
dquote>       }
dquote>     }
dquote>     LIMIT 10
dquote>   "
{
  "head" : {
    "vars" : [ "course" ]
  },
  "results" : {
    "bindings" : [ {
      "course" : {
        "type" : "uri",
        "value" : "http://open.fing.edu.uy/data/imerl/c1"
      }
    }, {
      "course" : {
        "type" : "uri",
        "value" : "http://open.fing.edu.uy/data/inco/logica"
      }
    }, {
      "course" : {
        "type" : "uri",
        "value" : "http://open.fing.edu.uy/data/inco/redes"
      }
    } ]
  }
}#
mparodi#
```

Figura 16: Uso de cURL para obtener información del triplestore

27 http://curl.haxx.se/

6.3. Consultas Federadas

En este último caso de estudio se pretende realizar una consulta desde un triplestore directamente a la base de datos de la plataforma. Esto podría ser de utilidad en sistemas desarrollados en estas tecnologías que deseen incorporar información relativa a las entidades presentes en el sistema, como cursos o clases, de forma de enriquecer la experiencia e información presentada a sus propios usuarios.

Virtuoso ofrece esta posibilidad a través de una consulta SPARQL en donde se especifica mediante la clausula SERVICE la dirección de un endpoint externo que se encargará de proveer parte de la información necesaria.

En la Figura 17 se presenta la misma consulta que en el caso de estudio anterior, pero esta vez se hace uso de la palabra reservada SERVICE por lo que parte de la misma es ejecutada remotamente. Los resultados obtenidos se encuentran en la 18.

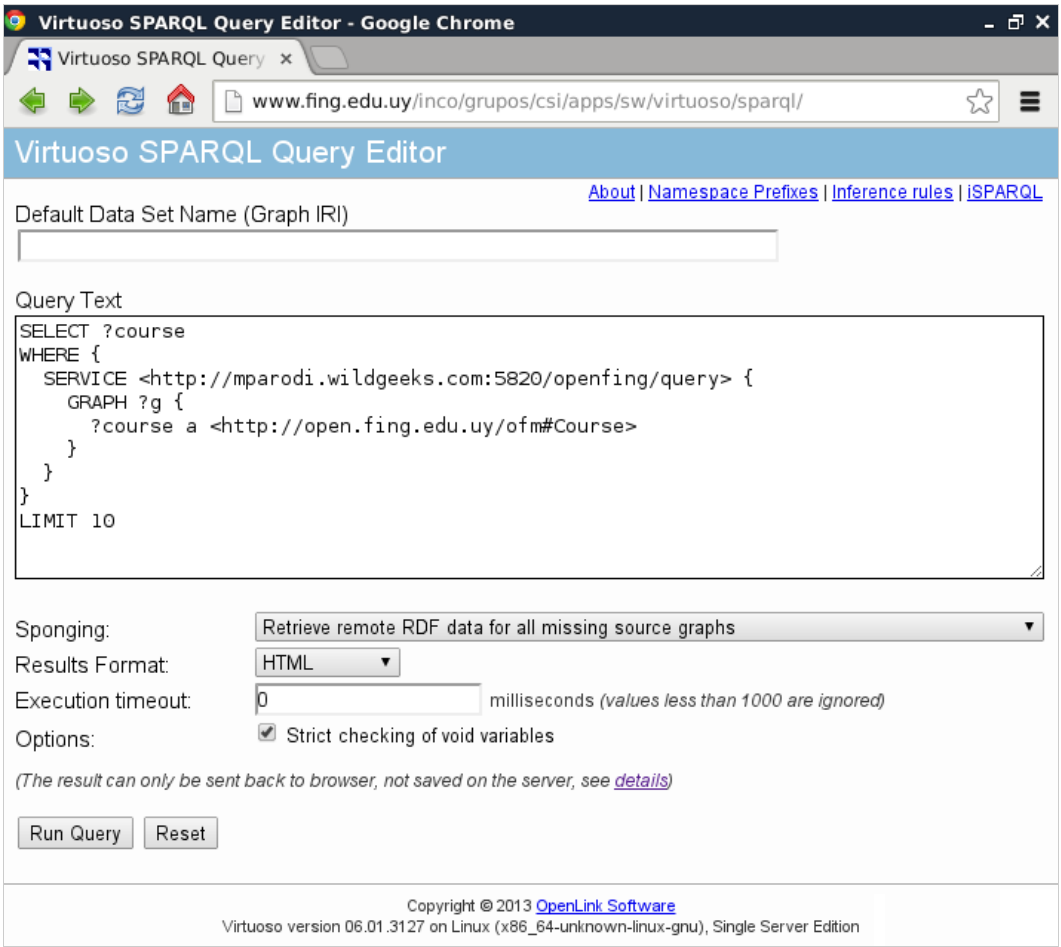


Figura 17: Consulta SPARQL utilizando SERVICE

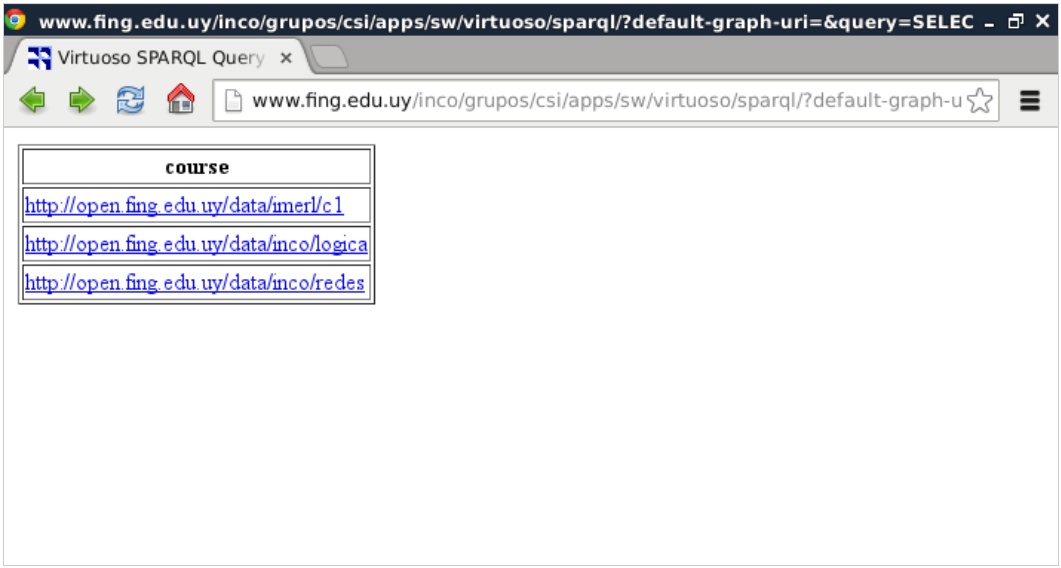


Figura 18: Resultados de una consulta SPARQL utilizando SERVICE

CAPÍTULO 7: CONCLUSIONES Y TRABAJO FUTURO

7.1. Conclusiones

Este trabajo estuvo motivado por la necesidad de contribuir al proyecto OpenFING con el desarrollo de un prototipo de una plataforma Web colaborativa que permita a sus usuarios hacer un mayor y mejor uso de los videos publicados.

Para esto, se realizó un relevamiento de proyectos similares y estándares vinculados, lo que derivó en la creación de una ontología de dominio genérica (MMC) y una extensión con vocabulario específico para este trabajo (OFM).

A partir de estas ontologías se creó un prototipo funcional utilizando tecnologías de la Web Semántica con el objetivo de validar el uso de los vocabularios diseñados en un problema real.

Esta plataforma posibilita a los usuarios, debidamente identificados en el sistema, contribuir con OpenFING creando anotaciones y asociando las mismas a fragmentos de video, lo cual permite relacionar cursos que en un principio se encontraban desvinculados.

Se logró construir un mecanismo de anotación y búsqueda de fragmentos basados enteramente en el motor de inferencia provisto por el triplestore usado y un mecanismo de expansión de consultas implementado para este proyecto, logrando cumplir los objetivos establecidos.

Cada fragmento es identificado por una URI, lo que permite a otros desarrolladores crear índices independientes a la plataforma y hacer referencia a los mismos desde otros sistemas. Esto expande las posibilidades de uso de la información generada y posibilita la creación de nuevas herramientas que exploten de otra forma los datos publicados.

Respecto al desarrollo y uso de Semantic Web como una alternativa a las bases de datos relacionales, se puede afirmar que es posible crear aplicaciones operativas en estas tecnologías, sin embargo es esperable que los problemas enfrentados hubieran sido más fáciles de resolver en una base de datos tradicional, pero no se hubiera tenido los beneficios de la flexibilidad del esquema de datos y el motor de inferencia.

Esta plataforma habilita a los docentes a adoptar nuevos mecanismos de enseñanza como *blended learning* y proporciona a los estudiantes un mecanismo adicional de aprendizaje, en el que pueden participar y contribuir activamente.

Finalmente se concluye que el trabajo realizado cumple con los objetivos planteados al comienzo del proyecto y satisface los resultados esperados en el mismo.

7.2. Trabajo a Futuro

7.2.1. Requerimientos

Existen varios requerimientos que no fueron incluidos en esta primera versión, entre ellos se encuentra implementar un mecanismo que permita a los usuarios moderar las anotaciones ingresadas, y a partir de ellas generar temarios de cursos y clases dinámicamente.

Además de la automoderación llevada a cabo por los usuarios, se podría agregar un módulo que filtre contenido mal intencionado o spam antes de que este llegue a la base de datos. Esto aumentaría sustancialmente la calidad de la información generada y la relevancia real de las anotaciones que se crean.

Otro requerimiento importante que beneficiaría a los usuarios es agregar ciertos videos a favoritos o para mirar más tarde, además se podría permitir suscribirse a cursos para recibir notificaciones cuando se publique una nueva clase. Asimismo, contar con salas de chat aumentaría en gran medida la utilidad del sistema y fomentaría su uso.

7.2.2. Búsqueda

El preprocesamiento de las anotaciones creadas y consultas ingresadas en el buscador es vital para el buen funcionamiento del módulo de búsqueda. En un futuro se deberá considerar la adopción de procesos más elaborados de análisis que permitan determinar con mayor exactitud la relación entre el texto ingresado y las entidades persistidas en el sistema.

7.2.3. Performance y Seguridad

Existen requerimientos claros relacionados a la performance de la plataforma y la seguridad de los datos, en un futuro se deberá estudiar la carga que soporta el sistema y se deberá hacer un análisis exhaustivo de los puntos débiles que presenta para mitigar cualquier riesgo asociado a la privacidad de la información sensible.

7.2.4. Vistas

Si bien se considera que las vistas son amigables con el usuario, se debe estudiar más detalladamente como mejorarlas, en particular la apariencia en general y la posición de los distintos componentes de las secciones. Eventualmente se podría permitir que los usuarios la personalicen en base a módulos independientes que se activan o desactivan de acuerdo a sus preferencias, lo cual permitiría a su vez que distintos equipos de desarrollo sean responsables por su funcionamiento.

7.2.5. Alta de Entidades

Algo importante a tener en cuenta es que no se desarrolló un mecanismo de alta de cursos y clases. Si bien esto no es estrictamente indispensable, ya que las ternas necesarias se pueden subir directamente a la base de datos, en una versión en producción sería deseable para facilitar el proceso de publicación.

7.2.6. Recursos Externos

En [45] se discute la posibilidad realizar análisis lingüístico sobre fuentes complementarias de información, en el contexto de este trabajo estas fuentes son, por ejemplo, el texto que aparece en las diapositivas de cada clase o las notas del curso. En etapas futuras se deberá considerar explotar las anotaciones de tipo recurso, las cuales de alguna manera se podrían ver como fuentes de información complementaria a los fragmentos a los que están asociadas.

7.2.7. Dereferenciación

En el futuro se deberá configurar el servidor Web sobre el que se ejecuta la plataforma de manera que sea posible dereferenciar las URIs de las entidades persistidas, según la recomendación de [46]. De esta forma mediante la negociación de contenido, una aplicación externa podrá obtener información estructurada respecto a un recurso, sin necesidad de ejecutar consultas SPARQL.

7.2.8. Extracción de Información de Videos

Algunos autores proponen extraer conceptos directamente de los frames clave de videos [47] a través de algoritmos de aprendizaje automático. En esta plataforma sería imposible realizar una análisis similar sobre los videos, ya que éstos en general muestran un pizarrón escrito y un profesor hablando. A pesar de eso, a partir de un proceso similar al que proponen podría ser posible realizar un análisis del audio de los videos. Evidentemente la metadata generada por este mecanismo serían de inferior calidad a la ingresada por usuarios, pero para aquellos videos para los que no exista anotación alguna podría ser de utilidad.

BIBLIOGRAFÍA

[1] "OpenCourseWare Members." [Online]. Available: http://www.ocwconsortium.org/members/.

[2] D. Ifenthaler, "Blended Learning," in *Encyclopedia of the Sciences of Learning*, N. Seel, Ed. Springer US, 2012, pp. 463–465.

[3] E. F. Codd, "A Relational Model of Data for Large Shared Data Banks," *Commun ACM*, vol. 13, no. 6, pp. 377–387, Jun. 1970.

[4] C. J. Date and H. Darwen, *A Guide to the SQL Standard*. Addison-Wesley, 1997.

[5] T. Berners-Lee, J. Hendler, and O. Lassila, "The Semantic Web," *Sci. Am.*, vol. 284, no. 5, pp. 35–43, May 2001.

[6] "World Wide Web Consortium (W3C)." [Online]. Available: http://www.w3.org/.

[7] "W3C Ontologies." [Online]. Available: http://www.w3.org/standards/semanticweb/ontology#specifications.

[8] J. Hendler, J. Holm, C. Musialek, and G. Thomas, "US Government Linked Open Data: Semantic.Data.Gov," *IEEE Intell. Syst.*, vol. 27, no. 3, pp. 25–31, May 2012.

[9] N. Shadbolt, K. O'Hara, T. Berners-Lee, N. Gibbins, H. Glaser, W. Hall, and m. c. schraefel, "Linked Open Government Data: Lessons from Data.Gov.Uk," *IEEE Intell. Syst.*, vol. 27, no. 3, pp. 16–24, May 2012.

[10] G. Kobilarov, T. Scott, Y. Raimond, S. Oliver, C. Sizemore, M. Smethurst, C. Bizer, and R. Lee, "Media Meets Semantic Web — How the BBC Uses DBpedia and Linked Data to Make Connections," in *Proceedings of the 6th European Semantic Web Conference on The Semantic Web: Research and Applications*, Berlin, Heidelberg, 2009, pp. 723–737.

[11] J. G. Breslin, D. O'Sullivan, A. Passant, and L. Vasiliu, "Semantic Web Computing in Industry," *Comput Ind*, vol. 61, no. 8, pp. 729–741, Oct. 2010.

[12] D. V. Levshin, "Mapping Relational Databases to the Semantic Web with Original Meaning," in *Proceedings of the 3rd International Conference on Knowledge Science, Engineering and Management*, Berlin, Heidelberg, 2009, pp. 5–16.

[13] "Annotating Academic Video Tool." [Online]. Available: http://entwinemedia.com/2013/annotations-tool/.

[14] A. L. Franzoni, C. P. Ceballos, and E. Rubio, "Interactive Video Enhanced Learning-Teaching Process for Digital Native Students," in *Advanced Learning Technologies (ICALT), 2013 IEEE 13th International Conference on*, 2013, pp. 270–271.

[15] R. Cyganiak, D. Wood, and M. Lanthaler, "RDF 1.1 Concepts and Abstract Syntax," Nov. 2013.

[16] "Notation3 (N3): A readable RDF syntax." [Online]. Available: http://www.w3.org/TeamSubmission/n3/.

[17] G. Carothers and E. Prud'hommeaux, "Turtle Syntax," W3C, Candidate Recommendation, Feb. 2013.

[18] "RDF: Anyone Can Make Statements About Any Resource." [Online]. Available: http://www.w3.org/TR/rdf-concepts/#section-anyone.

[19] D. Brickley and R. Guha, "RDF Vocabulary Description Language 1.0: RDF Schema," W3C, W3C Recommendation, Feb. 2004.

[20] P. Hitzler, M. Krötzsch, B. Parsia, P. F. Patel-Schneider, and S. Rudolph, Eds., *OWL 2 Web Ontology Language: Primer*. W3C Recommendation, 2009.

[21] A. Fokoue, Z. Wu, B. Motik, I. Horrocks, and B. C. Grau, "OWL 2 Web Ontology Language Profiles (Second Edition)," W3C, W3C Recommendation, Dec. 2012.

[22] S. Harris and A. Seaborne, “SPARQL 1.1 Query Language,” W3C, W3C Recommendation, Mar. 2013.
[23] J.-P. EVAIN, T. Bürger, T. Michel, F. Stegmaier, W. Bailer, J. Strassner, F. Sasaki, P.-A. Champin, V. Malaisé, J. Söderberg, and W. Lee, “Ontology for Media Resources 1.0,” W3C, W3C Recommendation, Feb. 2012.
[24] D. Brickley and L. Miller, *FOAF Vocabulary Specification*. 2007.
[25] A. Miles and J. R. Pérez-Agüera, “SKOS: Simple Knowledge Organisation for the Web,” *Cat. Classif. Q.*, vol. 43, no. 3, pp. 69–83, 2007.
[26] D. U. Board, *DCMI Metadata Terms*. 2006.
[27] Y. Raimond and S. Abdallah, *The Event Ontology*. 2007.
[28] R. Styles and N. Shabir, “Academic Institution Internal Structure Ontology (AIISO).” [Online]. Available: http://vocab.org/aiiso/schema.
[29] “Semantic Web Case Studies and Use Cases.” [Online]. Available: http://www.w3.org/2001/sw/sweo/public/UseCases/.
[30] “BBC - Ontologies.” [Online]. Available: http://www.bbc.co.uk/ontologies/.
[31] M. Maloney, N. Mendelsohn, H. Thompson, D. Beech, S. Gao, and M. Sperberg-McQueen, “W3C XML Schema Definition Language (XSD) 1.1 Part 1: Structures,” W3C, W3C Recommendation, Apr. 2012.
[32] A. Malhotra, M. Sperberg-McQueen, S. Gao, H. Thompson, D. Peterson, and P. V. Biron, “W3C XML Schema Definition Language (XSD) 1.1 Part 2: Datatypes,” W3C, W3C Recommendation, Apr. 2012.
[33] H. J. ter Horst, “Completeness, Decidability and Complexity of Entailment for RDF Schema and a Semantic Extension Involving the OWL Vocabulary,” *Web Semant*, vol. 3, no. 2–3, pp. 79–115, Oct. 2005.
[34] E. Sirin, M. Smith, and E. Wallace, *Opening, Closing Worlds — On Integrity Constraints*. 2008.
[35] B. Glimm, M. Horridge, B. Parsia, and P. F. Patel-schneider, *SWRL: A Syntax for Rules in OWL 2*. .
[36] D. V. Deursen, S. Pfeiffer, R. Troncy, and E. Mannens, “Media Fragments URI 1.0,” W3C, W3C Recommendation, Sep. 2012.
[37] B. Heitmann, S. Kinsella, C. Hayes, and S. Decker, *Implementing Semantic Web applications: reference architecture and challenges*. .
[38] A. Leff and J. T. Rayfield, “Web-Application Development Using the Model/View/Controller Design Pattern,” in *Proceedings of the 5th IEEE International Conference on Enterprise Distributed Object Computing*, Washington, DC, USA, 2001, p. 118–.
[39] H. Bast, F. Bäurle, B. Buchhold, and E. Haussmann, “A Case for Semantic Full-Text Search,” in *Proceedings of the 1st Joint International Workshop on Entity-Oriented and Semantic Search*, Portland, Oregon, 2012.
[40] J. Waitelonis and H. Sack, “Towards exploratory video search using linked data,” *Multimed. Tools Appl.*, vol. 59, no. 2, pp. 645–672, 2012.
[41] H. Q. Yu, C. Pedrinaci, S. Dietze, and J. Domingue, “Using Linked Data to Annotate and Search Educational Video Resources for Supporting Distance Learning,” *IEEE Trans Learn Technol*, vol. 5, no. 2, pp. 130–142, Jan. 2012.
[42] E. Brill, “Brill Tagger: A Simple Rule-Based Part of Speech Tagger,” in *Proceedings of the Third Conference on Applied Natural Language Processing*, Trento, Italy, 1992, pp. 152–155.
[43] M. F. Porter, “Snowball: An Algorithm for Suffix Stripping,” *Program*, pp. 130–137, 1980.
[44] C. Fellbaum, Ed., *WordNet, an Electronic Lexical Database*. MIT Press, 1998.

[45] T. Declerck, P. Buitelaar, M. Alcantara, M. Labský, and V. Svátek, “Adding Semantic Metadata to Audio-Video Material by Automatic Analysis of Complementary Sources,” in *VIE 2006: Proceedings of the International Conference on Visual Information Engineering, Bangalore, India*, 2006, pp. 261–266.
[46] T. Heath and C. Bizer, *Linked Data: Evolving the Web into a Global Data Space*. Morgan & Claypool Publishers, 2011.
[47] H.-S. Min, Y. B. Lee, W. De Neve, and Y. M. Ro, “Home Video Management System with Semantic Information,” in *IEEE International Conference on Consumer Electronics*, 2009, pp. 1–2.

APÉNDICE A: OpenFING

A comienzos del año 2012 propuse comenzar el trabajo de digitalización y publicación en Internet de cursos de la Facultad de Ingeniería, como un mecanismo para mitigar los principales problemas a los que se enfrenta la misma y como una herramienta de apoyo al aprendizaje para los estudiantes de la facultad.

De este modo OpenFING comenzó siendo un proyecto estudiantil, que hoy en día se mantiene gracias al trabajo voluntario de más de 15 estudiantes de varias carreras que han hecho posible la grabación y publicación de más de 250 clases de 11 cursos en tan solo tres semestres.

Debido al trabajo realizado, OpenFING ganó el apoyo del decanato, de muchos profesores de distintos institutos y de cientos de estudiantes que usan los videos diariamente como parte de su rutina de estudio o para preparar parciales y exámenes.

Esto permitió que a mediados del 2013 se plantee un módulo de taller, mediante el cual se les dió créditos a los estudiantes que participaron en OpenFING durante el segundo semestre del mismo año.

Además, como una propuesta planteada por las seis facultades del Área de Tecnologías y Ciencias de la Naturaleza y el Hábitat de la Universidad de la República se está comenzando a trabajar en extender el proyecto a otras facultades con el objetivo de crear un gran repositorio multimedia del conocimiento impartido en las distintas instituciones, posiblemente centralizando las publicaciones en esta plataforma en la Facultad de Ingeniería.

APÉNDICE B: Alineación de las Ontologías

ELEMENTO	RELACIÓN	CORRESPONDIENTE
mmc:Agent	rdfs:subClassOf	ma:Agent
mmc:Organization	rdfs:subClassOf	ma:Organisation
mmc:Person	rdfs:subClassOf	ma:Person
mmc:Audience	rdfs:subClassOf	ma:TargetAudience
mmc:Collection	rdfs:subClassOf	ma:Collection
mmc:Media	rdfs:subClassOf	ma:MediaResource
mmc:Fragment	rdfs:subClassOf	ma:MediaFragment
mmc:hasPerformer	rdfs:subPropertyOf	ma:features
mmc:hasCollection	rdfs:subPropertyOf	ma:hasCreator
mmc:hasFragment	rdfs:subPropertyOf	ma:hasNamedFragment
mmc:hasCaptioning	rdfs:subPropertyOf	ma:hasCaptioning
mmc:hasTopic	rdfs:subPropertyOf	ma:hasKeyword
mmc:hasMember	rdfs:subPropertyOf	ma:hasMember
mmc:hasAudience	owl:sameAs	ma:hasTargetAudience

ELEMENTO	RELACIÓN	CORRESPONDIENTE
mmc:Agent	rdfs:subClassOf	foaf:Agent
mmc:Person	rdfs:subClassOf	foaf:Person
mmc:Organization	rdfs:subClassOf	foaf:Organization

ELEMENTO	RELACIÓN	CORRESPONDIENTE
mmc:Metadata	rdfs:subClassOf	skos:Collection
mmc:Topic	rdfs:subClassOf	skos:Concept
mmc:hasMetadata	rdfs:subPropertyOf	skos:broaderTransitive

ELEMENTO	RELACIÓN	CORRESPONDIENTE
mmc:Agent	rdfs:subClassOf	dc:Agent

mmc:Media	rdfs:subClassOf	dc:PhysicalResource
mmc:hasCollection	rdfs:subPropertyOf	dc:Creator
mmc:hasPerformer	rdfs:subPropertyOf	dc:Contributor
mmc:hasAudience	rdfs:subPropertyOf	dc:Audience

ELEMENTO	RELACIÓN	CORRESPONDIENTE
mmc:Collection	rdfs:subClassOf	event:Event
mmc:Media	rdfs:subClassOf	event:Product
mmc:Agent	rdfs:subClassOf	event:Agent
mmc:hasPerformer	rdfs:subPropertyOf	event:agent
mmc:hasMedia	rdfs:subPropertyOf	event:product
mmc:hasComponent	rdfs:subPropertyOf	event:sub_event

ELEMENTO	CORRESPONDENCIA	CORRESPONDIENTE
mmc:Organization	rdfs:subClassOf	aiiso:Organization
mmc:Agent	rdfs:subClassOf	aiiso:Agent
mmc:hasMember	rdfs:subPropertyOf	aiiso:partOf
of:University	rdfs:subClassOf	aiiso:Institution
of:College	rdfs:subClassOf	aiiso:College
of:Department	rdfs:subClassOf	aiiso:Department
of:Collection	rdfs:subClassOf	aiiso:KnowledgeGrouping
of:Course	rdfs:subClassOf	aiiso:Course

APÉNDICE C: Caso de Estudio Extendido

Antes de empezar es indispensable definir los prefijos que se usarán para declarar los datos, tanto para el prefijo vacío ":" (en general usado para los datos que se están declarando en el archivo en cuestión) como las URIs del resto de los vocabularios involucrados: ofm, mmc, rdf, rdfs, owl y xsd.

```
@prefix :     <http://open.fing.edu.uy/data/>.
@prefix ofm:  <http://open.fing.edu.uy/ofm#>.
@prefix mmc:  <http://open.fing.edu.uy/mmc#>.
@prefix rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>.
@prefix owl:  <http://www.w3.org/2002/07/owl#>.
@prefix xsd:  <http://www.w3.org/2001/XMLSchema#>.
```

En los recuadros blancos aparecen las ternas declaradas, y en los recuadros grises la información que es posible inferir a partir de la semántica de las ontologías usadas.

```
:UniversityOfLocalhost
  ofm:id          "uol";
  rdf:type        ofm:University;
  mmc:name        "University of Localhost";
  mmc:description "University description.";
  mmc:website     "http://uol.edu/";
  mmc:image       "misc/university.jpg";
  mmc:thumbnail   "misc/university_thumbnail.png";
  ofm:hasCollege  :EngineeringCollege;
.
```

```
:UniversityOfLocalhost rdf:type mmc:Organization.
:EngineeringCollege    rdf:type ofm:College.
:EngineeringCollege    rdf:type mmc:Organization.
:EngeneeringCollege    ofm:isCollegeOf :UniversityOfLocalhost.
```

En general las inferencias de tipo son debidas a que la instancia es declarada como de un tipo que es subclase de la clase inferida, o bien se usa un predicado que tiene a la clase inferida como su dominio o rango, según corresponda.

En el caso anterior aparece un ejemplo de estos dos tipos de inferencia. Por un lado :UniversityOfLocalhost fue declarada como de tipo ofm:University, la cual es sublcase de mmc:Organization, por lo que se infiere la primer terna. En segundo lugar se dice que la Universidad of:hasCollege :EngineeringCollege, y dado que este predicado tiene rango ofm:College, se infiere la segunda terna, y a partir de esta la tercera como fue explicado al

comienzo. Estos dos tipos de inferencia son extremadamente frecuentes, y sencillas, por lo que se dará por entendida y se las omitirá para el resto del caso de uso.

Por último, como es el caso para todos los predicados definidos en ambas ontologías, existen predicados inversos que son inferidos automáticamente. En el ejemplo anterior :UniversityOfLocalhost se relaciona con :EngineeringCollege a través de ofm:hasCollege, por lo que también es válida la relación inversa utilizando ofm:isCollegeOf. El mismo hecho sucede con el resto de las relaciones por lo que también se omitirá la explicación para el resto de las ternas.

```
:EngineeringCollege
  ofm:id            "ec";
  mmc:name          "Engineering College";
  mmc:description   "College description";
  mmc:website       "http://uol.edu/ec";
  mmc:image         "misc/college.jpg";
  mmc:thumbnail     "misc/college_thumbnail.png";
  ofm:hasDepartment :ComputerScienceDepartment;
.
```

```
:ComputerScienceDepartment rdf:type ofm:Department.
:ComputerScienceDepartment rdf:type mmc:Organization.
```

En este caso, de igual manera que para ofm:hasCollege, se usa el rango de ofm:hasDepartment para establecer el tipo de :ComputerScienceDepartment.

```
:ComputerScienceDepartment
  ofm:id          "csd";
  mmc:name        "Computer Science Department";
  mmc:description "Department description.";
  mmc:website     "http://uol.edu/ec/csd";
  mmc:image       "misc/department.jpg";
  mmc:thumbnail   "misc/department_thumbnail.jpg";
  ofm:hasCourse   :LogicCourse;
.
```

```
:LogicCourse rdf:type ofm:Course.
```

```
:LogicCourse
  ofm:id          "logica";
  mmc:name        "Lógica";
  mmc:description "Logic description.";
  mmc:website     "http://uol.edu/ec/csd/logic";
  mmc:image       "misc/course.png";
```

```
  mmc:thumbnail   "misc/course_thumbnail.png";
.
```

```
:LogicCourse  rdf:type mmc:Collection.
```

```
:LogicCourse
  ofm:hasProfessor :John;
  ofm:hasProfessor :Anna;
.
```

```
:John rdf:type ofm:Professor.
:Anna rdf:type ofm:Professor.
:Anna rdf:type ofm:Professor.
```

```
:LogicCourse
  ofm:hasLecture   :FirstPart;
  ofm:hasLecture   :SecondPart;
  ofm:hasProfessor :John;
  ofm:hasProfessor :Anna;
.
```

```
:FirstPart  rdf:type ofm:Lecture.
:SecondPart rdf:type ofm:Lecture.
:John       rdf:type ofm:Professor.
:Anna       rdf:type ofm:Professor.
```

```
:FirstPart
  ofm:id          "1";
  mmc:description "Introduction to the course";
  mmc:hasVideo    :Class1;
  mmc:hasAudio    :Introduction;
  mmc:hasImage    :Professors;
  mmc:image       "misc/video1.png";
  mmc:thumbnail   "misc/video1_thumbnail.png";
.
```

```
:Class1       rdf:type mmc:Video.
:Introduction rdf:type mmc:Audio.
:Professors   rdf:type mmc:Image.
:LogicCourse  mmc:hasVideo :Class1.
:LogicCourse  mmc:hasAudio :Introduction.
:LogicCourse  mmc:hasImage :Professors.
```

En el caso anterior se tiene un ejemplo de composición de predicados. En la ontología, mmc:hasMedia y todos sus subpredicados fueron declarados usando owl:propertyChainAxiom,

lo cual permite especificar que es posible componer dos o más predicados para inferir una nueva relación.

En particular, mmc:hasVideo compuesto con mmc:hasComponent es una subpropiedad de de mmc:hasVideo. Observando que mmc:hasLecture es a su vez subpropiedad de mmc:hasComponent se puede apreciar como cuando un curso (:LogicCourose) se relaciona con una clase a través de mmc:hasLecture (:FirstPart) y dicha clase tiene un video (:Class1), entonces es válido afirmar que el curso tiene ese video también. Análogamente, el curso tiene los audios e imágenes que tienen sus clases.

```
:SecondPart
  ofm:id          "2";
  mmc:description "First order logic";
  mmc:hasVideo    :Class2;
  mmc:image       "misc/video2.png";
  mmc:thumbnail   "misc/video2_thumbnail.png";
.
```

```
:Class2 rdf:type mmc:Video.
```

```
:Class1
  mmc:duration    5640;
  mmc:hasFragment :Review;
  mmc:hasFragment :Syllabus;
.
```

```
:Class2
  mmc:duration    4254;
.
```

```
:Review   rdf:type mmc:Fragment.
:Syllabus rdf:type mmc:Fragment.
```

```
:Review
  mmc:hasTopic :FirstOrderLogic;
.
```

```
:FirstOrderLogic rdf:type mmc:Topic.
:Class1          mmc:hasTopic :FirstOrderLogic.
:FirstPart       mmc:hasTopic :FirstOrderLogic.
:LogicCourse     mmc:hasTopic :FirstOrderLogic.
```

De igual manera que ocurría con mmc:hasMedia y sus subpropiedades, mmc:hasMetadata y sus subpropiedades se elevan en la jerarquía a través de tres predicados distintos,

mmc:hasFragment, mmc:hasMedia y mmc:hasComponent. Esto explica las últimas tres ternas inferidas.

```
:Syllabus
  mmc:hasTopic    :FirstOrderLogic;
  mmc:hasNote     :CourseResources;
  mmc:hasResource :Induction;
  mmc:start       7;
  mmc:finish      15;
.
```

```
:FirstOrderLogic rdf:type     mmc:Topic.
:CourseResources rdf:type     mmc:Note.
:Induction       rdf:type     mmc:Resource.
:FirstOrderLogic rdf:type     mmc:Metadata.
:CourseResources rdf:type     mmc:Metadata.
:Induction       rdf:type     mmc:Metadata.
:Class1          mmc:hasTopic :FirstOrderLogic.
:FirstPart       mmc:hasTopic :FirstOrderLogic.
:LogicCourse     mmc:hasTopic :FirstOrderLogic.
:Class1          mmc:hasNote  :CourseResources.
:FirstPart       mmc:hasNote  :CourseResources.
:LogicCourse     mmc:hasNote  :CourseResources.
:Class1          mmc:hasNote  :Induction.
:FirstPart       mmc:hasNote  :Induction.
:LogicCourse     mmc:hasNote  :Induction.
```

```
:FirstOrderLogic
  mmc:text       "First order logic";
  mmc:created    "2013-01-20"^^xsd:date;
  mmc:hasCreator :Anna;
.

:CourseResources
  mmc:text       "Check the course's book for more details";
  mmc:hasCreator :Anna;
  mmc:created    "2013-02-21"^^xsd:date;
.

:Induction
  mmc:text       "http://logic.com/induction";
  mmc:created    "2013-03-22"^^xsd:date;
  mmc:hasCreator :John;
.
```

```
:John
  ofm:id             "jsmith";
  mmc:name           "John Smith";
  mmc:image          "misc/professor1.png";
  mmc:website        "http://johnsmith.com/";
  mmc:description    "John description.";
  mmc:hasCollection :LogicCourse;
.
```

```
:John mmc:hasCollection :FirstPart.
```

mmc:hasCollection también se compone con mmc:hasComponent, esto explica la terna inferida ya que :John está relacionado con :LogicCourse a través de mmc:hasCollection, y este curso tiene un componente, :FirstPart, que es de tipo mmc:Collection.

```
:Anna
  ofm:id             "asmith";
  mmc:name           "Anna Smith";
  mmc:image          "misc/professor2.png";
  mmc:website        "http://annasmith.com/";
  mmc:description    "Anna description.";
  mmc:hasCollection :FirstPart;
.
```

```
:Anna mmc:hasCollection :FirstPart.
```

```
:Peter
  ofm:id           "pgold";
  mmc:name         "Peter Gold";
  rdf:type         ofm:Student;
  ofm:hasEnrolled :LogicCourse;
  mmc:hasWatched   :Class1;
.
```

```
:Peter mmc:hasUsed :FirstPart.
:Peter mmc:hasUsed :Logic.
```

Es posible inferir las dos ternas del fragmento anterior debido a que mmc:hasUsed, y por tanto sus subpropiedades como mmc:hasWatched, se pueden componer con mmc:isMediaOf y mmc:isComponentOf. Intuitivamente esto quiere decir que si una persona ha usado (mirado, escuchado o visto) un recurso multimedia, también ha usado la colección en la que aparece, las colecciones en las que esta está contenida y así sucesivamente.

```
:Sam
  ofm:id   "sdulk";
  mmc:name "Sam Dulck";
  rdf:type ofm:Student;
.
```

```
:Class1
  mmc:hasFragment :Correction;
.
```

```
:Correction rdf:type mmc:Fragment.
```

```
:Correction
  mmc:hasNote :Proof;
  mmc:start   0;
  mmc:finish  5;
.
```

```
:Class1      mmc:hasMetadata :Proof.
:FirstPart   mmc:hasMetadata :Proof.
:LogicCourse mmc:hasMetadata :Proof.
```

```
:Proof
  mmc:text       "It was supposed to be x^2";
  mmc:created    "2013-01-20"^^xsd:date;
  mmc:hasCreator :Sam;
.
```

APÉNDICE D: Interfaces del Prototipo

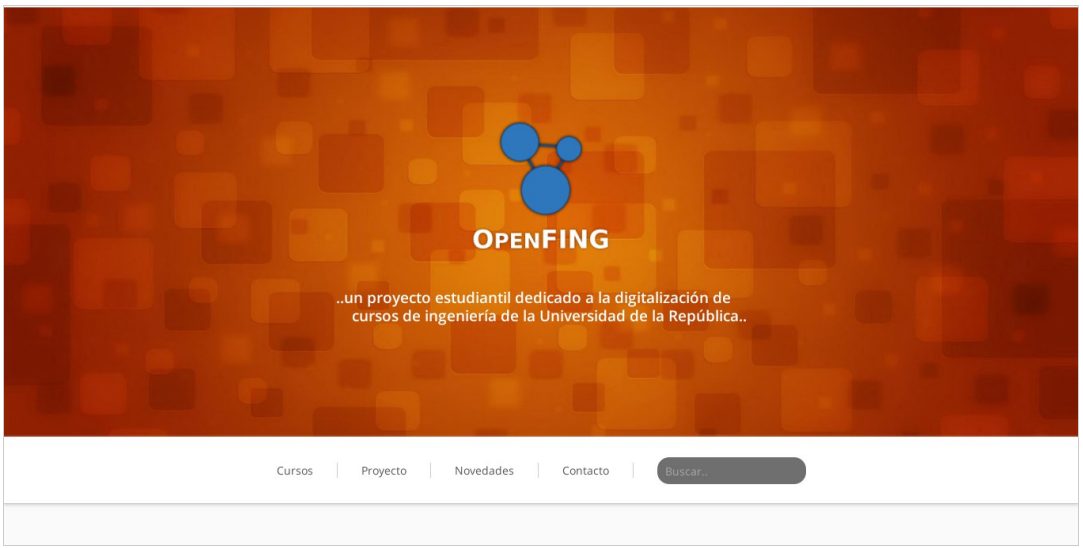


Figura 19: Sección de inicio

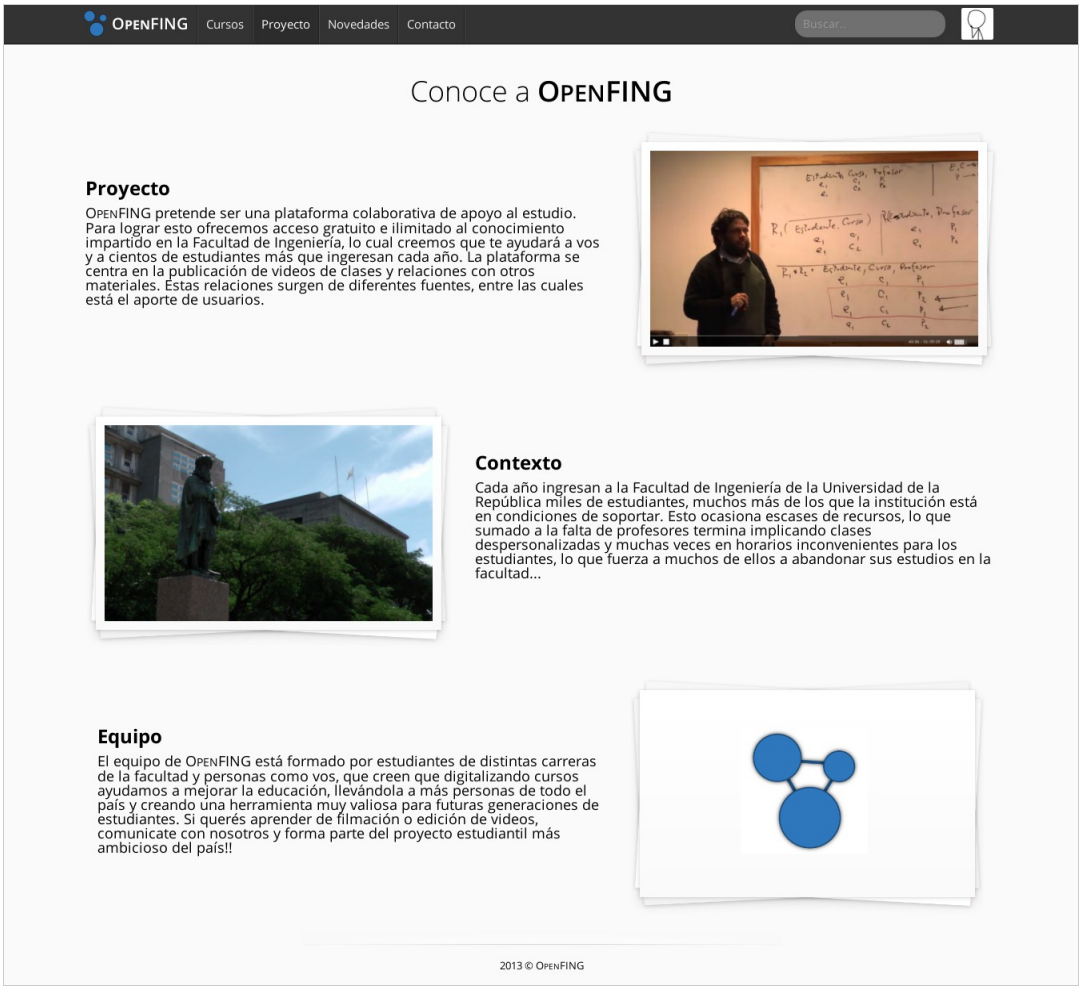


Figura 20: Información del proyecto

Figura 21: Sección de contacto

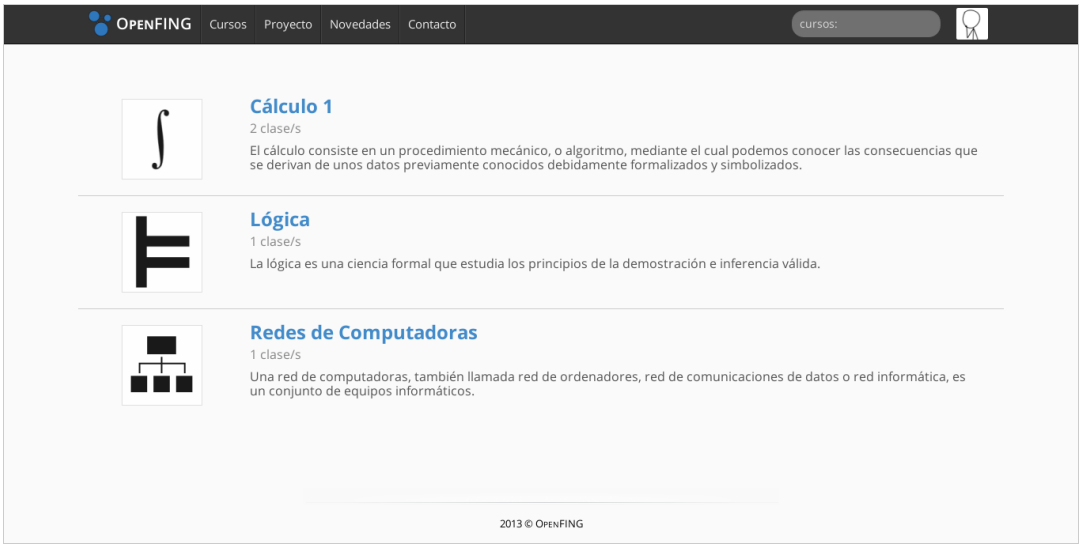


Figura 22: Lista de cursos

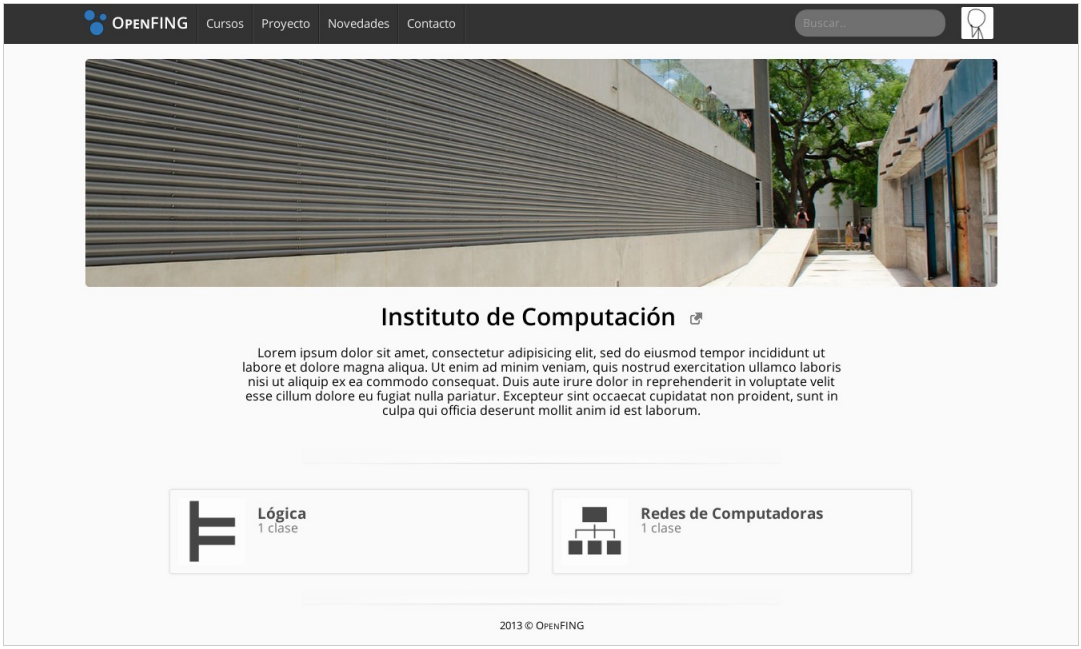


Figura 23: Página de un instituto

Figura 24: Página de un curso

Figura 25: Página de un profesor

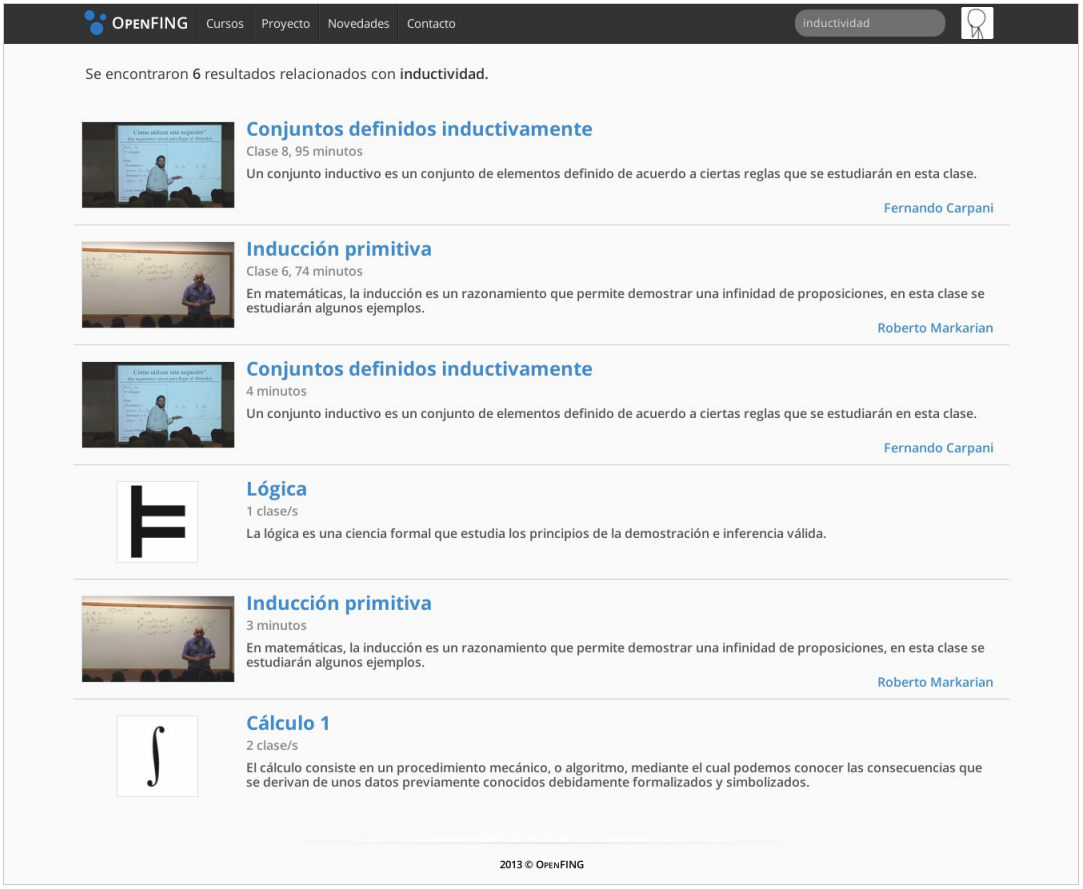


Figura 26: Resultados de una búsqueda

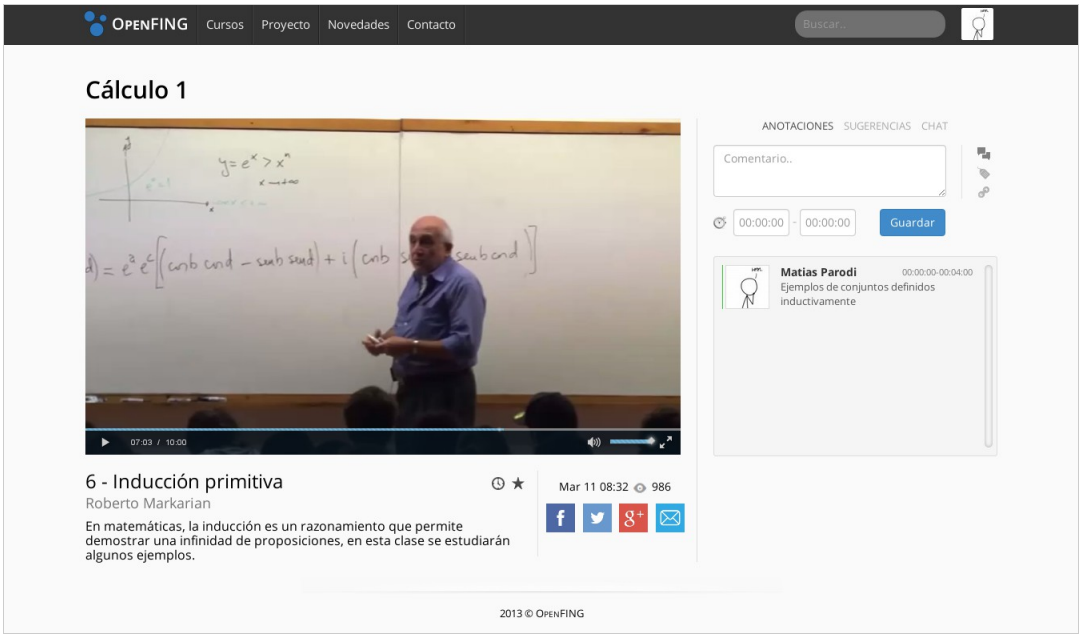


Figura 27: Vista del video de una clase