Trabajo Fin de Grado
Grado en Ingeniería Informática

Desarrollo de un tutor de aprendizaje basado en tecnologías Cloud y semánticas

A Web-accessible tutoring system based on semantic and cloud computing technologies

Autor

David Vergara Manrique

Directores

Sandra Baldassarri
Pedro Álvarez Pérez-Aradros

Universidad de Zaragoza
Escuela de Ingeniería y Arquitectura

Septiembre 2016

TRABAJOS DE FIN DE GRADO / FIN DE MÁSTER

Escuela de Ingeniería y Arquitectura Universidad Zaragoza

DECLARACIÓN DE AUTORÍA Y ORIGINALIDAD

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

D./Dª. David Vergara Manrique,

con nº de DNI 73132646Y en aplicación de lo dispuesto en el art. 14 (Derechos de autor) del Acuerdo de 11 de septiembre de 2014, del Consejo de Gobierno, por el que se aprueba el Reglamento de los TFG y TFM de la Universidad de Zaragoza,

Declaro que el presente Trabajo de Fin de (Grado/Máster) Grado, (Título del Trabajo) Desarrollo de un tutor de aprendizaje basado en tecnologías Cloud y semánticas,

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

Zaragoza, a 26 de Agosto de 2016

Fdo: David Vergara Manrique

Desarrollo de un tutor de aprendizaje basado en tecnologías Cloud y semánticas

Resumen

Este trabajo de fin de grado comprende el análisis, diseño e implementación de un sistema software, accesible vía Web, que permita a los estudiantes acceder y utilizar objetos de aprendizaje, así como evaluar el conocimiento adquirido a partir de los mismos. Esta evaluación se lleva a cabo mediante la creación de estructuras semánticas que representen el contenido del objeto de aprendizaje accedido.

Funcionalmente, el sistema permite al profesor almacenar estos objetos de aprendizaje, en formato de vídeo, junto con un *Topic Map*(estructura semántica) que describa lo que espera que el estudiante aprenda a través del objeto de aprendizaje. Los estudiantes, son capaces de buscar y acceder a estos objetos, bien a través de las asignaturas en las que están matriculados, o haciendo uso de una barra de búsqueda mediante palabras clave. Tras la visualización del vídeo, el estudiante puede introducir un *Topic Map* que represente el conocimiento adquirido a través del objeto de aprendizaje. Internamente, el sistema analizará semánticamente en qué medida el *Topic Map* elaborado por el estudiante se corresponde con el *Topic Map* provisto por el profesor, utilizando para ello técnicas predefinidas y contrastadas en base a distintas métricas de similaridad, junto con algoritmos del teorema de grafos. A partir de este análisis, se generan una serie de indicadores que son mostrados al estudiante para que éste observe el grado de comprensión alcanzado del objeto de aprendizaje.

El sistema desarrollado posee una arquitectura desplegada en la nube, orientada a servicios, que implementa un modelo Cliente-Servidor estructurado en capas. La parte servidor ofrece un API RESTful y es la encargada de llevar a cabo el análisis semántico, durante el cual se hará uso de servicios web externos. La parte cliente presenta una interfaz gráfica a los profesores y estudiantes mediante la cual poder interaccionar con la aplicación. Adicionalmente, el sistema cuenta con un sistema de acceso seguro basado en cuentas de usuario, y un sistema de registro de eventos, que genera ficheros de *log* con las interacciones de red y acciones de los usuarios en el sistema para su posterior análisis.

En relación a las tecnologías, se puede diferenciar claramente entre tecnologías utilizadas en la parte cliente, servidor y las tecnologías de almacenamiento de información. Por un lado, la interfaz del usuario se ha implementado haciendo uso de AngularJS junto a Boostrap, dos frameworks que permiten la creación de interfaces dinámicas y responsivas (para su utilización en smartphones). En lo relativo al servidor, éste hace uso de NodeJS junto con su *framework* ExpressJS, tecnologías con la que se define el API RESTful mediante el cual se puede interaccionar con el sistema vía llamadas HTTP. Además, se hace uso de Java con el fin de llevar a cabo el análisis de equivalencia entre *Topic Maps*. Por último, para el almacenamiento de información del sistema, se ha hecho uso de MongoDB como base de datos. Esta base de datos es de tipo NoSQL orientada a documentos; la interacción con la misma se realizará desde el servidor haciendo uso de la herramienta Mongoose.

Agradecimientos

Quisiera empezar, agradeciendo tanto a Pedro como a Sandra, los directores de mi proyecto, la ayuda, implicación y el interés mostrado a lo largo de todo el proyecto, así como haberme dado la oportunidad de llevarlo a cabo.

También me gustaría agradecer a Javier Fabra por los consejos y la ayuda que nos ha prestado durante el desarrollo del proyecto.

A mis padres por sus ánimos, paciencia, consejos y haberme apoyado durante toda mi vida, sin su ayuda no habría conseguido llegar hasta aquí.

Finalmente quisiera agradecer a todos mis amigos que se han preocupado y me han ayudado durante todos estos años y, especialmente, a mis amigos y compañeros de clase por estos cuatro años, a ratos estresantes, pero sobre todo cargados de buenos momentos. Sin ellos hubiera sido mucho más duro el camino hasta llegar aquí.

Índice

Índice de figuras

Índice de tablas

1. Introducción

Este primer capítulo describe el contexto en el que se desarrolla el trabajo, así como la motivación del mismo. A continuación se hace una breve introducción a los objetivos de alto nivel del trabajo y a las tecnologías utilizadas, finalizando con la explicación de la estructura que presenta el resto de la memoria.

1.1. Contexto y motivación del trabajo

En muchas asignaturas, los profesores elaboran material docente u objetos de aprendizaje para sus estudiantes y publican estos objetos para que sean utilizados en el proceso de aprendizaje. La duda es en qué medida los estudiantes aprenden con estos objetos. Una opción es que los estudiantes trabajen los objetos y después hacerles un test o un examen donde tengan que demostrar lo que aprendieron. Lo ideal sería que el estudiante recibiera un *feedback* automático sobre cómo está desarrollando su aprendizaje (uno de los objetivos del sistema), pero para ello debería ser capaz de expresar de alguna forma lo que ha aprendido (razón de usar estructuras semánticas denominadas *Topic Map*). Por otro lado, también es interesante que el profesor reciba información continua sobre el proceso de aprendizaje de sus estudiantes para poder reaccionar y mejorar sus materiales (otro objetivo del sistema).

El sistema desarrollado pretende ser una herramienta que facilite este contexto de aprendizaje. Desde el punto de vista del profesor, resulta interesante disponer de una plataforma, accesible vía web, en la que poder almacenar objetos de aprendizaje que sirvan como material educativo, que la misma plataforma permita a los estudiantes representar y almacenar lo aprendido a través de estructuras semánticas, *Topic Maps*, y que el proceso de análisis de este aprendizaje se desarrolle de forma automática. Desde el punto de vista de los estudiantes, resulta muy útil poder acceder a todo el material educativo a través de un navegador, poder representar y almacenar los conocimientos adquiridos en cualquier momento y obtener los resultados del análisis de su aprendizaje al instante.

Con lo definido anteriormente este sistema queda enmarcado en el ámbito educativo. Con él se espera agilizar y automatizar el proceso de estudio por parte de los estudiantes y de evaluación del conocimiento por parte de los profesores. Por último, permite obtener información detallada y personalizada de y para cada estudiante, lo que genera un retroalimentación muy interesante.

1.2. Objetivos de alto nivel

El proyecto propone una solución al problema de automatizar el análisis del aprendizaje que un estudiante adquiere a través de un objeto de aprendizaje, desarrollando para ello un sistema que ha de ser accesible vía web. Para la realización del proyecto han sido definidos varios objetivos generales, los cuales se pueden diferenciar en objetivos funcionales y arquitecturales.

Los objetivos funcionales se corresponden con:

- Permitir el almacenamiento de objetos de aprendizaje en el sistema y proporcionar facilidad de acceso y reproducción a los vídeos que contienen.
- Permitir el almacenamiento de *Topic Maps*, estructuras semánticas que contienen la representación del contenido de un objeto de aprendizaje, en el sistema de una forma fácil e intuitiva.
- Permitir llevar a cabo un análisis de equivalencia de conceptos entre *Topic Maps*, desarrollados por estudiantes y profesores para un mismo objeto de aprendizaje, que aporte una serie de indicadores de aprendizaje que ayuden tanto a estudiantes a comprender sus fallos y sus progresos, como a profesores a medir la calidad de los materiales proporcionados y el aprendizaje de sus estudiantes. Para ello es necesaria la implementación de algoritmos de análisis.

Respecto a los objetivos arquitecturales, estos tienen como finalidad:

- La realización de un estudio de las tecnologías a utilizar para el desarrollo de la aplicación web y llevar a cabo la implementación de las mismas teniendo en cuenta que el sistema puede sufrir un aumento de usuarios, por lo que se deberán elegir tecnologías que faciliten la escalabilidad del sistema.
- La implementación de una arquitectura modular en el sistema durante su desarrollo con el fin de que en un futuro éste pueda ser fácilmente distribuido entre varias máquinas.
- La elección del lenguaje y las estructuras de programación más adecuadas para modelar la evaluación entre *Topic Maps*.

Por último, hay que realizar una serie de tests y evaluaciones del sistema para comprobar el correcto funcionamiento del mismo. Con este fin se compararán los resultados obtenidos del análisis automático del aprendizaje de los estudiantes, con los resultados obtenidos en un proceso manual llevado a cabo por el profesor.

Categoría	Tecnologías y Herramientas
Implementación interfaz usuario	AngularJS [2] Boostrap [3] HTML [13] CSS [14] JavaScript [15]
Implementación BBDD	MongoDB [10] MongooseJS [12]
Implementación lógica del sistema	NodeJS [4] ExpressJS [5] JavaScript [15]
Procesado de las estructuras semánticas	Java EE [9] Wordnik [16]
Despliegue del sistema	Azure Cloud Computing [17]
Herramientas de desarrollo	Eclipse IDE [18] WebStorm IDE [19] GitHub [20]
Ofimática y documentación	LibreOffice [21] LaTeX [22] Notepad ++ [23] Creately [24]

Tabla 1: Tecnologías y herramientas utilizadas en el proyecto

1.3. Tecnologías utilizadas

Para llevar a cabo la implementación de los objetivos citados anteriormente, es necesario el uso de varias herramientas y tecnologías. El objetivo de esta sección es dar una primera visión general de qué recursos se van a emplear en el proyecto. En la tabla 1 de esta sección aparecen enumeradas las diferentes herramientas y tecnologías a utilizar en el desarrollo del sistema. La tabla se encuentra agrupada por categorías, mostrándose en primer lugar las tecnologías relacionadas con la implementación del proyecto, tanto las destinadas a la parte de interfaz de usuario, como las enfocadas a la implementación de la base de datos y las orientadas a la parte lógica. A continuación se especifican las tecnologías que hay que aplicar en el procesado de las estructuras semánticas, tecnologías de gran importancia en el sistema. Son seguidas por los proveedores de los que se hará uso para el despliegue del sistema y, por último, aparecen las herramientas de desarrollo y herramientas de ofimática y documentación que sirven de apoyo para la realización del proyecto. Más adelante, en la sección 2.5, se puede encontrar un análisis detallado de por qué se ha elegido cada tecnología.

1.4. Estructura de la memoria

La memoria del trabajo se compone de seis capítulos y cinco anexos que se especifican a continuación:

- **Capítulo 1**: Introducción. Se hace una introducción general del proyecto, mostrando el contexto y la motivación del mismo, los objetivos de alto nivel definidos, diferenciando entre objetivos funcionales y arquitecturales y, por último, las tecnologías a utilizar.
- **Capítulo 2**: Análisis del sistema. Se profundiza en la definición y representación de estructuras del conocimiento, así como en los objetivos del sistema, definiendo para estos requisitos funcionales y no funcionales. Al final del capítulo se lleva a cabo una justificación de las distintas tecnologías seleccionadas para el desarrollo del proyecto.
- **Capítulo 3**: Diseño e Implementación. Se muestra el entorno de implantación del sistema, la arquitectura general del mismo, desglosando a continuación las diferentes capas que la forman, y el proceso de diseño e implementación que se ha llevado a cabo en cada una de ellas.
- **Capítulo 4**: Evaluación de los algoritmos de análisis. En este capítulo queda reflejada la evaluación realizada de los algoritmos de análisis del sistema. Esta evaluación se ha centrado tanto en términos de rendimiento a la hora de evaluar un *Topic Map*, como en términos de la corrección de los resultados obtenidos.
- **Capítulo 5**: Gestión del proyecto. Se define la metodología utilizada en el desarrollo del proyecto así como la planificación y esfuerzos invertidos en el mismo.
- **Capítulo 6**: Conclusiones y trabajo futuro. Para finalizar la memoria se muestran las conclusiones obtenidas de la realización del proyecto, el trabajo que se puede llevar a cabo para mejorar el sistema y la opinión personal del estudiante.

- **Anexo A**: Lógica matemática de los indicadores de aprendizaje. Se detalla desde un punto de vista matemático los cálculos y fórmulas necesarias para la obtención de los indicadores de aprendizaje.
- **Anexo B**: Manual de instalación del sistema. Se muestra una explicación paso a paso de cómo realizar el despliegue del sistema.
- **Anexo C**: Estructura de directorios y ficheros del proyecto. Se incluye una explicación para programadores de cual es la estructura de directorios y componentes que presenta el proyecto.
- **Anexo D**: Especificación de las APIs. Se detallan las distintas APIs del sistema, explicando la estructura de las llamadas y el tipo de respuestas que se generan.

- **Anexo E**: Manual de usuario. Se detalla cómo llevar a cabo las funciones básicas del sistema, tanto para un usuario estudiante, como para un usuario administrador.

2. Análisis del sistema

Tras una introducción general del proyecto, este segundo capítulo se centra en exponer el trabajo de análisis llevado a cabo en el mismo. Comienza explicando la representación y análisis de estructuras de conocimiento introducidas en el capítulo anterior, a continuación se plantean los objetivos generales del sistema, seguidos por la definición de los requisitos funcionales y no funcionales del proyecto, y, por último, se incluye una justificación de las tecnologías seleccionadas para desarrollar el sistema.

2.1. Representación y análisis de estructuras del conocimiento

Los conceptos claves en la representación de las estructuras de conocimiento de la aplicación son los objetos de aprendizaje, los mapas semánticos (*Topic Maps*), y el análisis e indicadores de aprendizaje; a continuación se detalla y explica cada uno de ellos.

2.1.1. Objeto de aprendizaje

En el sistema, un objeto de aprendizaje se corresponde con un vídeo de corta duración de un tema de estudio. Estos vídeos pueden haber sido desarrollados por un estudiante o profesor, o bien haber sido obtenidos de internet. El profesor es el encargado de introducirlo en el sistema y de asociarle una serie de palabras claves que representen el contenido del mismo, las cuales servirán posteriormente para que los estudiantes puedan buscar estos objetos de aprendizaje en la aplicación, bien a través de asignaturas en las que están matriculados, o a través de un buscador mediante palabras clave.

Estos vídeos suelen resumir y exponer los aspectos más importantes del tema de estudio a tratar, por lo que son de gran utilidad en el aprendizaje de los estudiantes. Estos estudiantes, tras acceder a estos objetos de aprendizaje y visualizarlos, deben representar el aprendizaje adquirido del mismo a través de un mapa semántico (*Topic Map*).

2.1.2. Mapa semántico

Un *Topic Map* es una estructura semántica, en forma de grafo dirigido, que permite representar el contenido de un vídeo. Al introducir un objeto de aprendizaje en el sistema, los profesores incluyen un *Topic Map* que representa los contenidos y conceptos a adquirir a través de ese vídeo. Por otro lado, los estudiantes deben introducir un *Topic Map* por cada objeto de aprendizaje visualizado en el sistema, con el fin de expresar el conocimiento adquirido. Internamente se lleva a cabo un análisis de equivalencia de conceptos con respecto al *Topic Map* desarrollado por el profesor. Este análisis genera una serie de indicadores de aprendizaje que permitirán que el estudiante pueda conocer el grado de aprendizaje obtenido, y el profesor el desarrollo de sus estudiantes.

La estructura de un *Topic Map* se corresponde con un grafo dirigido. Este grafo se compone de nodos etiquetados, arcos etiquetados y dirigidos entre nodos, y, por último, de un valor de importancia comprendido entre 1 y 3 que se le asigna a cada nodo. Un nodo se corresponde con un concepto importante, la etiqueta del nodo define este concepto, y, el valor numérico, indica el nivel de importancia (siendo 3 el valor que representa mayor importancia). Por otro lado, una arco orientado entre dos nodos representa una relación entre los conceptos de cada nodo. Este arco se encuentra etiquetado por un verbo que define la relación.

En la figura 1 se puede observar un ejemplo de *Topic Map* que representa el contenido de un vídeo del tema de estudio "Diseño centrado en niños". Se puede observar que se han identificado seis conceptos relevantes del vídeo, siendo los más importantes los conceptos de "Niño" y "Interacción del usuario". Además, aparecen ocho relaciones dirigidas entre estos conceptos, cada una de ellas con un verbo que la define.

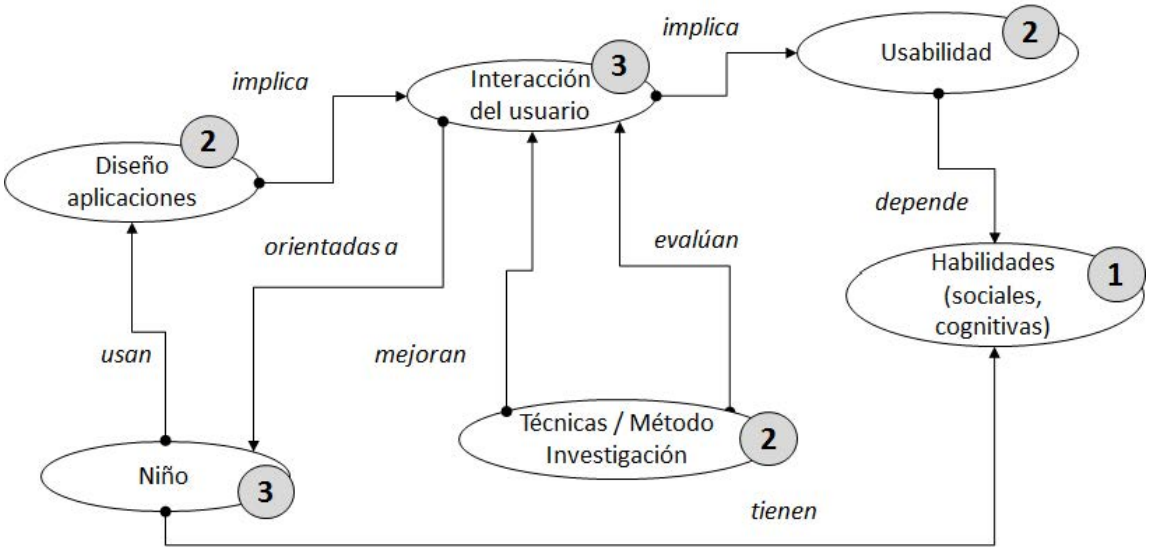


Figura 1: *Topic Map* que representa los contenidos de un vídeo del tema "Diseño centrado en niños"

2.1.3. Análisis e indicadores de aprendizaje

El análisis del aprendizaje permite conocer lo que ha aprendido un estudiante a través de un objeto de aprendizaje. Para ello se lleva a cabo un análisis de equivalencia de conceptos entre los *Topic Maps* del estudiante y del profesor de un objeto de aprendizaje concreto. En este análisis se comparan los nodos, relaciones y valores entre los dos *Topic Maps*, con el fin de conocer el grado de similaridad que presentan ambas estructuras semánticas. Además, se hace uso de servicios externos de equivalencia de términos que proporcionen información acerca de si dos términos se pueden considerar equivalentes o no. La implementación detallada de esta comparación entre *Topic Maps* puede encontrarse en el capítulo 3.4.5 de la memoria.

Este análisis genera una serie de indicadores de aprendizaje que son de utilidad tanto para profesores como para estudiantes, ya que miden el desarrollo del aprendizaje de los estudiantes y se

pueden extraer distintas conclusiones a través de ellos, como la calidad de los recursos proporcionados por el profesor. A continuación se presentan los indicadores de aprendizaje obtenidos a través de este análisis:

- **Similitud entre conceptos**: Relaciona los nodos introducidos en un *Topic Map* con respecto a los que aparecen en el *Topic Map* de referencia. Para ello se tiene en cuenta el número de nodos que presentan una equivalencia semántica (etiquetados por términos semánticamente equivalentes) entre ambas estructuras y la diferencia existente entre los valores de importancia (valor numérico comprendido entre 1 y 3) con los que están etiquetados cada par de nodos equivalentes. El indicador genera un resultado numérico comprendido entre 0 y 1, siendo mayor la similitud entre conceptos cuanto más próximo es este valor a 1.

- **Similitud en adyacencia**: Establece una correspondencia entre las relaciones introducidas en un *Topic Map* con respecto a las que aparecen en el *Topic Map* de referencia. Para explicar el indicador se supone que en TM (*Topic Map* a evaluar) existen dos nodos A y B unidos por un arco desde el nodo A hasta el nodo B y en TM_Ref (*Topic Map* de referencia) existen dos conceptos C y D unidos por un arco desde el nodo C hasta el nodo D. Si los nodos del TM A y B son equivalentes semánticamente a los nodos del TM_Ref C y D respectivamente, y los arcos de ambas relaciones están etiquetados por conceptos NO equivalentes semánticamente, entonces se dice que existe una relación en equivalencia en estructura entre las dos relaciones (arcos). El indicador cuenta cuántas relaciones en equivalencia en estructura existen entre los dos *Topic Maps* y las relaciona con el número total de relaciones que posee el *Topic Map* de referencia. Se genera de este modo un resultado numérico comprendido entre 0 y 1, siendo mayor la similitud en adyacencia cuanto más próximo es este valor a 1.

- **Similitud en relación**: Establece una correspondencia entre las relaciones introducidas en un *Topic Map* con respecto a las que aparecen en el *Topic Map* de referencia. Para explicar el indicador se supone que en TM (*Topic Map* a evaluar) existen dos nodos A y B unidos por un arco desde el nodo A hasta el nodo B y en TM_Ref (*Topic Map* de referencia) existen dos conceptos C y D unidos por un arco desde el nodo C hasta el nodo D. Si los nodos del TM A y B son equivalentes semánticamente a los nodos del TM_Ref C y D respectivamente, y los arcos de ambas relaciones están etiquetados por conceptos equivalentes semánticamente, entonces se dice que existe una relación en equivalencia semántica entre las dos relaciones (arcos). El indicador cuenta cuántas relaciones en equivalencia semántica existen entre los dos *Topic Maps* y las relaciona con el número total de relaciones que posee el *Topic Map* de referencia. Se genera de este modo un resultado numérico comprendido entre 0 y 1, siendo mayor la similitud en relación cuanto más próximo es este valor a 1.

- **Similitud de *Topic Map***: Establece una correspondencia general entre dos *Topic Maps*, uno que se desea evaluar y otro que es el de referencia. Para ello se tienen en cuenta los indicadores

de aprendizaje definidos anteriormente: “Similitud entre conceptos”, “Similitud en relación” y “Similitud en adyacencia”. A cada uno de ellos se le aplica un peso de importancia con el fin de poder relacionarlos entre sí. Tras llevar a cabo cálculos con los resultados de los indicadores explicados anteriormente, el indicador genera un resultado numérico comprendido entre 0 y 1, siendo mayor la similitud general entre los dos *Topic Maps* cuanto más próximo es este valor a 1.

- **Conceptos no detectados**: Muestra los conceptos que aparecen en el *Topic Map* de referencia pero no en el *Topic Map* introducido para ser analizado.

En el anexo A ”Lógica matemática de los indicadores de aprendizaje”puede encontrarse una explicación detallada de las fórmulas matemáticas, en las que se aplican los conceptos definidos previamente, utilizadas para la obtención de cada indicador de aprendizaje.

2.2. Descripción de los objetivos

Una vez definida la representación de las estructuras de conocimiento de la aplicación, se pueden describir los objetivos generales de la misma. Estos objetivos se pueden agrupar en objetivos de los usuarios de la aplicación, distinguiendo entre profesores y estudiantes, en objetivos de análisis automático de estructuras del conocimiento y en objetivos tecnológicos.

Se definen los siguientes objetivos orientados a profesores:

- Podrán gestionar objetos de aprendizaje en el sistema.
- Podrán gestionar cuentas de estudiante.
- Podrán gestionar asignaturas en el sistema.

En cuanto a los estudiantes, aparecen los siguientes objetivos:

- Podrán acceder y reproducir objetos de aprendizaje en el sistema.
- Se permitirá la introducción a los estudiantes de un *Topic Map* en el sistema que represente el aprendizaje obtenido por los mimos a través de un objeto de aprendizaje.
- Se permitirá la evaluación de los *Topic Maps* introducidos en el sistema, obteniendo indicadores de aprendizaje.
- Podrán acceder a un histórico de los indicadores de aprendizaje obtenidos.

En relación al análisis automático de estructuras del conocimiento se definen los siguientes objetivos:

- Se podrá llevar a cabo el análisis de dos *Topic Maps* en el sistema obteniendo para los mismos una serie de indicadores de aprendizaje.
- Hay que realizar la implementación interna del análisis entre dos *Topic Maps*.
- Se debe llevar a cabo la implementación de un servicio web de equivalencia entre términos que será usado en la implementación del análisis citado anteriormente.
- Se debe llevar a cabo de una evaluación para la implementación realizada.

Por último, existen una serie de objetivos tecnológicos directamente relacionados con los objetivos descritos anteriormente:

- Hay que llevar a cabo un análisis de tecnologías a utilizar en el desarrollo de la plataforma web que permitan una sencilla escalabilidad y distribución de los componentes del sistema.
- Hay que llevar a cabo la implementación de la aplicación web.
- Se deben analizar las tecnologías a utilizar en la implementación del análisis de equivalencia entre *Topic Maps* del sistema.

2.3. Requisitos funcionales

A partir de la descripción de objetivos llevada a cabo se pueden definir una serie de requisitos funcionales enmarcados en tres categorías. A continuación se muestra esta serie de requisitos funcionales:

Requisitos funcionales desde el punto de vista del profesor

RF-P1: Un profesor puede acceder al sistema utilizando una cuenta de administrador.

RF-P2: Un profesor puede introducir y eliminar objetos de aprendizaje en el sistema.

RF-P3: Un profesor puede introducir nuevos usuarios en el sistema, generando una clave-contraseña para ellos.

RF-P4: Un profesor puede introducir asignaturas en el sistema.

Requisitos funcionales desde el punto de vista del estudiante

RF-E1: Un estudiante puede acceder al sistema utilizando un identificador de usuario y contraseña que le habrá sido proporcionado previamente por un profesor.

RF-E2: Un estudiante puede consultar la lista de asignaturas en las que está matriculado.

RF-E3: Un estudiante puede buscar objetos de aprendizaje en el sistema bien por un buscador introduciendo palabras clave, o a través de una asignatura en la que se encuentra matriculado.

RF-E4: Un estudiante puede visualizar objetos de aprendizaje.

RF-E5: Un estudiante puede almacenar un *Topic Map* (que represente el conocimiento adquirido a través de un objeto de aprendizaje) para cada objeto de aprendizaje existente en el sistema.

RF-E6: Un estudiante puede almacenar en el sistema una imagen que represente gráficamente un *Topic Map* a la vez que introduce el *Topic Map* en el sistema.

RF-E7: Un estudiante puede grabar un nuevo *Topic Map* para un objeto de aprendizaje, siendo borrado automáticamente el anterior *Topic Map* asociado a ese objeto de aprendizaje en el momento que es subido el nuevo.

RF-E8: Un estudiante puede evaluar el *Topic Map* introducido previamente en el sistema para un objeto de aprendizaje reproducido.

RF-E9: Un estudiante puede consultar una lista con todas las evaluaciones de *Topic Map* que ha realizado, y acceder a cada una de ellas.

RF-E10: Un estudiante puede descargar cada *Topic Map* en formato de texto y cada imagen introducidas por él en el sistema.

Requisitos funcionales de estructuras del conocimiento y funcionalidad del sistema

RF-C1: Una vez solicitada la evaluación de un *Topic Map* por parte de un estudiante, este sistema hará uso de un servicio de equivalencia de términos para el futuro análisis del *Topic Map*.

RF-C2: Tras consultar el servicio de equivalencia de términos, el sistema llevará a cabo una evaluación semántica de equivalencia entre el *Topic Map* introducido por el estudiante para un objeto de aprendizaje, y el *Topic Map* de referencia introducido por el profesor para ese objeto de aprendizaje.

RF-C3: La evaluación semántica llevada ha cabo por el sistema ha de generar un indicador de "Similitud entre conceptos", que muestre la similitud entre los conceptos de los dos *Topic Maps* en forma de un valor numérico comprendido entre 0 y 1, siendo mayor la similitud entre conceptos cuanto más próximo es este valor a 1.

RF-C4: La evaluación semántica llevada ha cabo por el sistema ha de generar un indicador de "Similitud en relación", que muestre la similitud entre las relacione de los dos *Topic Maps* en forma de un valor numérico comprendido entre 0 y 1, siendo mayor la similitud entre relaciones cuanto más próximo es este valor a 1.

RF-C5: La evaluación semántica llevada ha cabo por el sistema ha de generar un indicador de "Similitud en adyacencia", que muestre la similitud entre la estructura de los dos *Topic Maps* en forma de un valor numérico comprendido entre 0 y 1, siendo mayor la similitud entre las estructuras cuanto más próximo es este valor a 1.

RF-C6: La evaluación semántica llevada ha cabo por el sistema ha de generar un indicador de "Similitud de *Topic Map*", que muestre la similitud general entre los dos *Topic Maps* en forma de un valor numérico comprendido entre 0 y 1, siendo mayor la similitud entre ellos cuanto más próximo es este valor a 1.

RF-C7: La evaluación semántica llevada ha cabo por el sistema ha de generar un indicador de "Conceptos no detectados", que muestre los conceptos presentes en el *Topic Map* de referencia introducido por el profesor para el objeto de aprendizaje, y que no aparecen en el *Topic Map* introducido por el estudiante para ese objeto de aprendizaje.

RF-C8: El sistema llevará a cabo una evaluación de los algoritmos de análisis implementados.

RF-C9: El sistema comprobará si un estudiante ya tiene introducido en el sistema un *Topic Map* para un objeto de aprendizaje concreto, y en caso de que el estudiante quiera introducir un nuevo *Topic Map* para ese objeto de aprendizaje, se le advertirá de que su *Topic Map* anterior junto con los indicadores obtenidos para el mismo serán eliminados.

RF-C10: El sistema almacenará ficheros de *log* donde quedarán reflejadas todas las operaciones realizadas en el sistema. Tanto los accesos y llamadas de red, como las acciones de los usuarios. El formato de este fichero debe incluir un identificador único para cada sesión de cada usuario y debe ser apropiado para poder realizar futuros análisis.

RF-C11: El sistema almacenará la cuenta de administrador y las cuentas de estudiante.

RF-C12: El sistema almacenará las asignaturas introducidas por los profesores.

RF-C13: El sistema almacenará los objetos de aprendizaje introducidos por los profesores.

RF-C14: El sistema almacenará los *Topic Maps* introducidos por estudiantes y profesores.

RF-C15: El sistema almacenará los indicadores de aprendizaje generados de la evaluación de un *Topic Map* introducido por un estudiante.

RF-C16: El sistema incluirá una página de ayuda on-line para la ayuda a la interacción y evaluación de *Topic Maps* con el mismo.

2.4. Requisitos no funcionales

Aparte de requisitos funcionales se han definido requisitos no funcionales, aquellos que hacen referencia a tecnologías, despliegue del sistema, instalación, etc. Son los siguientes:

RNF1: El sistema desarrollado será accesible vía red.

RNF2: El sistema desarrollado será accesible a través de diferentes navegadores y dispositivos.

RNF3: El sistema desarrollado será auto desplegable en instancias remotas.

RNF4: El sistema deberá comportarte del mismo modo en todos los navegadores web.

RNF5: El sistema deberá ser capaz de soportar múltiples peticiones concurrentes de distintos usuarios.

RNF6: El sistema deberá permitir una sencilla escalabilidad y distribución de los componentes del sistema.

2.5. Justificación de tecnologías seleccionadas

Para la resolución del proyecto se han utilizado variedad de tecnologías en los distintos componentes del sistema. Es necesario realizar una justificación de por qué se ha seleccionado cada tecnología y qué ventajas tiene. Se pueden dividir las tecnologías a justificar en las siguientes categorías: plataforma y *framework* para el desarrollo del servidor web, tecnología para el desarrollo de la capa lógica de análisis de *Topic Maps*, tecnología para el almacenamiento y persistencia de datos, tecnología para el desarrollo de la interfaz Web y plataforma y *framework* para el desarrollo del servidor web.

2.5.1. Plataforma y framework para el desarrollo del servidor web

Se había de elegir una plataforma que permita el desarrollo de aplicaciones Web y muestre una buena escalabilidad y concurrencia. Por otro lado, la elección de un *framework* facilita el desarrollo del software y el mantenimiento de una estructura modular del mismo.

Como plataforma se ha elegido NodeJS [4], un entorno de ejecución dirigido por eventos, por lo tanto asíncrono, que se ejecuta sobre un intérprete en JavaScript [15]. Los motivos de la elección son que es una tecnología que permite desarrollar de una manera rápida y fácil un servidor web, a la vez que dota al mismo de una gran escalabilidad y posibilidad de establecer un número de conexiones muy elevado, en comparación con otros lenguajes de programación de servidores como Java [9] o PHP [25]. Estos beneficios los logra gracias al motor V8 [26] (desarrollado por Google) sobre el que se ejecuta, el cual proporciona una compilación y ejecución de código JavaScript a altas velocidades; al bucle de eventos que gestiona internamente creando llamadas de entrada/salida asíncronas; y al uso de un único hilo de ejecución, el cual evita crear un nuevo hilo de sistema operativo para cada conexión, lo que significa un ahorro de memoria física del sistema.

En referencia al *framework*, se ha elegido para el desarrollo ExpressJS [5], ya que es el más utilizado en NodeJS, y facilita la programación del servidor web mediante la gran cantidad de *plugins* que tiene disponibles, los cuales permiten implementar diferentes tipos de servicios.

2.5.2. Tecnología para el desarrollo de la capa lógica de análisis de *Topic Maps*

Se ha tenido que estudiar la tecnología a utilizar en el análisis semántico de *Topic Maps*, teniendo en cuenta que será necesario mantener distintas estructuras complejas como matrices de adyacencia, tablas hash o listas para poder llevar a cabo el análisis, y que será necesario realizar iteraciones sobre estas estructuras.

La capa lógica de análisis de *Topic Maps* ha sido desarrollada utilizando Java EE, plataforma de programación para desarrollar software en el lenguaje de programación Java. Se ha elegido ya que es una tecnología fácilmente integrable con NodeJS; orientada a objetos, lo que permite la creación de objetos y estructuras complejas, como las nombradas anteriormente, para el análisis de *Topic Maps*; por otra parte se aprovecha la experiencia del estudiante con el lenguaje, punto clave en este apartado ya que es necesario la realización de cálculos de cierta complejidad en la programación de los análisis.

2.5.3. Tecnología para el almacenamiento y persistencia de datos

Ha habido que seleccionar una base de datos y una tecnología de acceso asociada a la misma que resuelvan el problema de almacenamiento de datos de la aplicación.

Se ha decidido hacer uso de una base de datos NoSQL [11] orientada a documentos, en concreto MongoDB [10]. MongoDB no guarda los datos en registros como hacen las bases de datos relacionales, sino que los guarda en documentos [27]. La estructura de estos documentos es almacenada en BSON [28], una representación binaria de JSON, lo cual permite guardar documentos con estructura JSON. Esto que será de gran utilidad para almacenar las estructuras de los *Topic Maps* ya que contienen varias subestructuras complejas como son los nodos y relaciones, fácilmente definibles a través de estrcturas JSON.

Otra razón para la elección de MongoDB es la buena integración que posee junto a NodeJS, para la que se hace uso de Mongoose [12], una herramienta de modelado de objetos para MongoDB diseñada con el fin de trabajar en un entorno asíncrono. Mongoose permite generar esquemas de datos, lo que facilita el almacenamiento de los documentos en la base de datos de una forma más estructurada, a la vez que proporciona un sistema de consultas y una interacción con la base de datos muy sencilla desde NodeJS.

2.5.4. Tecnología para el desarrollo de la interfaz Web

Es necesario el uso de una tecnología soportada por la mayoría de navegadores web con el fin de que el sistema sea lo más accesible posible.

Para el desarrollo de la interfaz Web se ha hecho uso de AngularJS [2], *framework* para el desarrollo "Front end"de una página Web basado en el patrón de diseño MVC (modelo, vista, controlador) [30]. Junto a AngularJS se hace uso del *framework* Boostrap [3], el cual proporciona

plantillas de diseño, formularios, botones, menús de navegación etc. basados en HTML [13] y CSS [14], que hacen que el diseño de una página Web sea más sencillo de llevar a cabo.

El punto más interesante de AngularJS en este proyecto es su facilidad de integración junto a NodeJS y MongoDB. Estas tres tecnologías junto a ExpressJS (framework de desarrollo utilizado en el "Back end") definen el conjunto de subsistemas de software para desarrollo de aplicaciones y páginas web dinámicas, basado en el lenguaje JavaScript, MEAN Stack(MongoDB, ExpressJS, AngularJS, NodeJS) [31]; lo que permite que todos los componentes del proyecto gocen de una excelente interoperabilidad entre sí.

2.5.5. Plataforma proveedora de servicios de ejecución en la nube

Se ha tenido que seleccionar un proveedor de servicios que permita la ejecución del sistema desarrollado y su acceso remoto desde cualquier ordenador.

Microsoft Azure [17] ha sido la plataforma elegida para ejecutar el sistema sobre su infraestructura de Cloud Computing [32]. La elección de esta tecnología viene marcada por la disponibilidad de cuentas que posee la Universidad de Zaragoza en esta plataforma. La infraestructura Cloud Computing de Microsoft Azure permite reducir la necesidad de mantener un hardware que de soporte al sistema en la Universidad, así como minimizar los gastos en la administración de los sistemas. Además, permite un fácil despliegue de aplicaciones desarrolladas en NodeJS.

3. Diseño e implementación

En este capítulo se explica el diseño y la implementación de la aplicación. Se comienza haciendo una introducción al entorno del sistema, a continuación se presenta la arquitectura general del sistema, diferenciando las distintas capas y luego se detalla cada una de ellas.

3.1. Entorno del Sistema

La figura 2, muestra una primera aproximación a los diferentes actores y componentes que interactúan entre sí en el diseño realizado para la aplicación del tutor de aprendizaje. Esta aplicación implementa un modelo Cliente-Servidor [7] y se corresponde con un servicio web [6], ya que los distintos componentes están distribuidos en la red.

Los actores que interactúan con el sistema son estudiantes y profesores, punto uno (1) de la figura. Esta interacción se lleva a cabo a través de un navegador web (correspondiente al "Front end" del sistema). Gracias a esta capa, los profesores son capaces de introducir en el sistema objetos de aprendizaje y *Topic Maps* de referencia. Del mismo modo, los estudiantes pueden acceder a los objetos de aprendizaje almacenados previamente por los profesores en la aplicación, observar los distintos vídeos educativos que ofrecen y, posteriormente, almacenar un *Topic Map* que represente los conceptos y relaciones más importantes del vídeo que acaba de visualizar.

Al almacenar el estudiante un *Topic Map* en el sistema, la parte lógica de la aplicación, "Back end"(2), realiza una evaluación del mismo, comparándolo con el *Topic Map* que un profesor había creado previamente como referencia para ese objeto de aprendizaje concreto. Antes de comenzar la evaluación de los *Topic Maps* estudiante y profesor, es necesario obtener una serie de sinónimos para los conceptos del *Topic Map* creado por el estudiante con el fin de realizar una posterior evaluación semántica entre ellos. La obtención de estos sinónimos se realiza mediante la llamada a un servicio web de equivalencia de términos mediante el protocolo HTTP (3). Una vez se tienen los sinónimos, se comienza la evaluación. En ésta, se aplican una serie de métodos predefinidos en la aplicación, posteriormente, el estudiante recibe los resultados de la misma en forma de indicadores de aprendizaje, lo que le proporciona una retroalimentación muy importante para su proceso de estudio.

Por último, el sistema integra una base de datos (4) NoSQL con el fin de dar persistencia a los datos que se utilizan en la aplicación, tanto los introducidos por estudiantes (*Topic Maps*, indicadores generados como resultados), como los que han debido de ser introducidos previamente por los profesores para que el estudiante pueda trabajar con la aplicación (Objetos de aprendizaje, asignaturas, cuentas de estudiante).

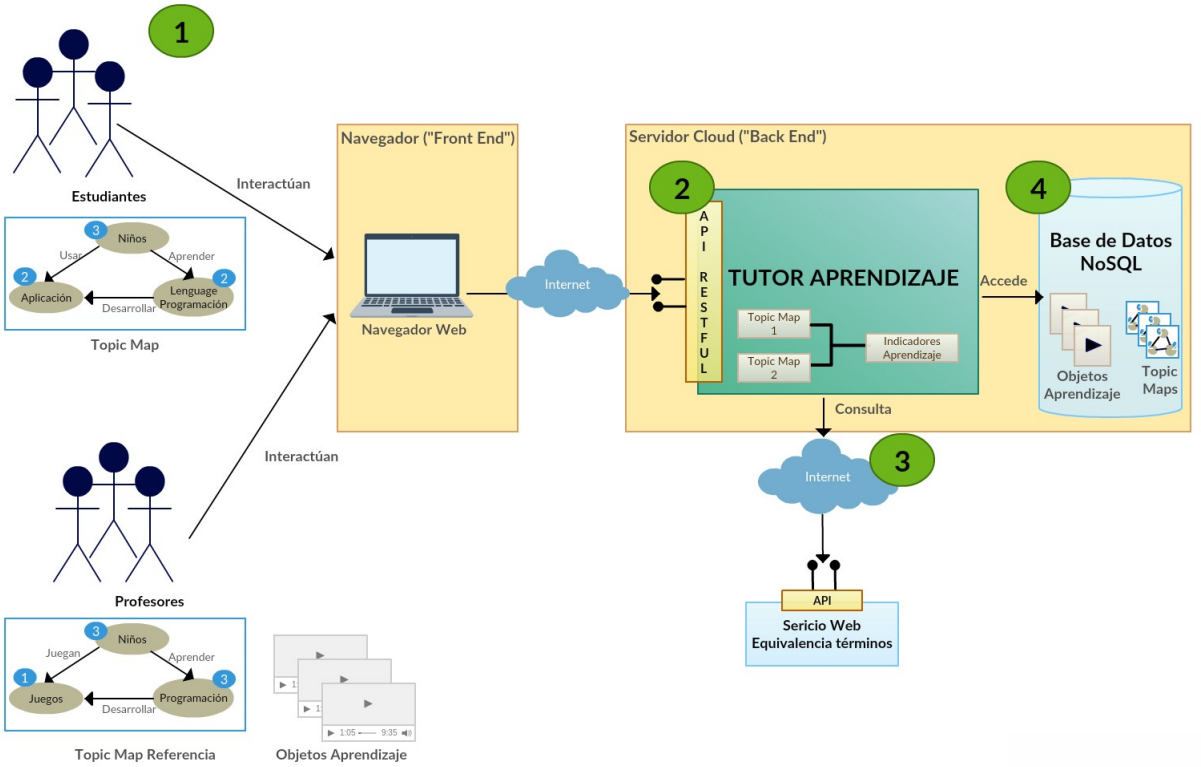


Figura 2: Interacción de actores y componentes

3.2. Arquitectura General

El sistema se basa en una arquitectura multicapa [33]. Se ha optado por la realización de un diseño multicapa ya que permite abstraer cada capa del resto, consiguiendo de este modo que la arquitectura sea escalable. Cada una de ellas posee una misión simple, por lo que es sencillo entender la finalidad de cada capa y a la vez aportar independencia entre entre sí, pudiendo modificar el comportamiento de una de ellas sin necesidad de modificar el resto del sistema. A continuación se va a explicar la distribución de estas capas y, posteriormente, cómo interactúan entre sí.

3.2.1. Capas/componentes del sistema

La figura 3 muestra la arquitectura multicapa generada en el sistema, la cual está compuesta por tres capas comunes a la mayoría de servicios Web: capa de presentación, capa lógica de negocio y capa de persistencia y acceso a datos.

- La capa de presentación se corresponde con el "Front end" del sistema, es la encargada de tomar datos de los usuarios con el fin de enviarlos a la capa lógica de negocio, y de obtener datos de esta capa para mostrárselos al usuario a través de una interfaz. Esta interacción se realiza haciendo uso del protocolo HTTP. AngularJS y Boostrap son las tecnologías usadas en el sistema para la implementación de esta capa. Mientras Boostrap es utilizado como

framework que proporciona plantillas y menús personalizables, AngularJS es el encargado de gestionar el intercambio de información con el "Back end", la interacción del usuario con el sistema. Esta capa sigue un patrón de diseño MVC (modelo, vista, controlador) [30], el cual separa los datos y la lógica de negocio de la interfaz de usuario.

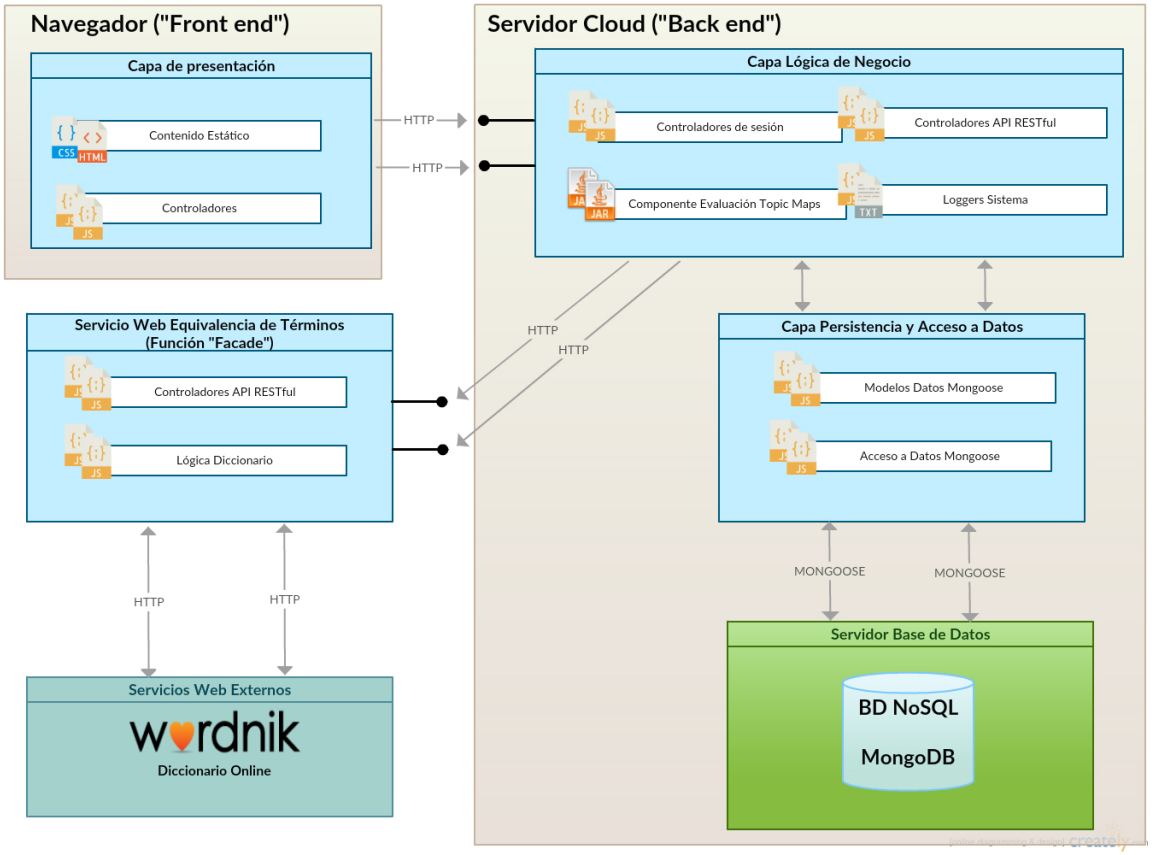


Figura 3: Arquitectura multicapa del sistema

- La capa lógica de negocio realiza las funciones de atender las peticiones realizadas desde la capa de presentación, procesarlas y generar una respuesta. El procesamiento que realiza va desde una simple llamada/escritura a la base de datos, hasta el análisis de un *Topic Map*. Para poder realizar estos procesamientos hace uso de la capa de persistencia y acceso a datos, y de un servicio de equivalencia de términos para obtener los sinónimos necesarios para el análisis de los *Topic Maps*. Esta capa presenta un API RESTful [1] a través del cual se puede interaccionar con la misma. Este API ha sido desarrollado utilizando la tecnología NodeJS junto al framework ExpressJS. Además, presenta la tecnología Java con el fin de llevar a cabo el análisis de equivalencia entre *Topic Maps*.

- La capa de persistencia y acceso a datos es la que ataca directamente a la base de datos (MongoDB). Esta capa se encarga de obtener y procesar las peticiones generadas por la capa lógica que afectan a la base de datos (escrituras, lecturas, consultas complejas), así como de definir el esquema que han de seguir los documentos almacenados en la misma. El sistema

utiliza la herramienta Mongoose en esta capa, encargada de establecer las conexiones con MongoDB y definir los esquemas de datos. Además, gracias a Mongoose y su interfaz HTTP, la base de datos no tiene que estar situada en la misma máquina que el servidor web.

- Aparte de estas tres capas, ha surgido la necesidad de definir un servicio web propio encargado de realizar invocaciones a servicios externos de equivalencia de términos; quedando integrado dentro del servidor web pero siendo fácilmente migrable a otra máquina. Este servicio implementa un patrón "Facade"[34], utilizado con el fin de abstraer la invocación a estos servicios web externos, de modo que, si se desea invocar a un servicio externo diferente, únicamente hay que modificar el servicio web implementado, dejando intactas el resto de capas. La función de este servicio consiste en aceptar peticiones de la capa lógica, llamar a servicios web externos para generar una respuesta, y proporcionar esta respuesta a la capa lógica. Actualmente se utiliza el diccionario online Wordnik [16] como servicio externo de equivalencia de términos, pero este servicio web ha sido diseñado con la intención de que sea sencilla la adhesión de uno o más servicios externos, o la sustitución de éste. Al igual que capa lógica, las tecnologías empleadas en esta capa son NodeJS junto al framework ExpressJS, implementado en el servicio web una interfaz API RESTful.

Cómo llevar a cabo el despliegue de esta arquitectura se puede encontrar en el anexo B "Manual de instalación del sistema".

3.2.2. Estructura dinámica del sistema

Una vez se tiene la arquitectura general y antes de detallar cada capa es interesante entender cómo interaccionan y qué tipo de interacciones se producen entre las mismas. La interfaz API RESTful definida en la capa lógica de negocio ("Back end"), genera dos tipos de modelos de operaciones lógicas en función del recurso invocado. Cada modelo corresponde con un patrón de interacción entre las capas de presentación, capa lógica, capa de persistencia y base de datos. Toda la funcionalidad del sistema ha sido implementada en base a estos patrones.

Por un lado, se puede definir el patrón de interacción más común del sistema, que se corresponde con una petición de acceso directo a los datos del repositorio. En este tipo de petición la capa lógica se limita a procesar la llamada de la capa de presentación y hacer uso de la capa de persistencia para obtener una respuesta, o almacenar información. Un ejemplo de este tipo de interacción es la obtención de una lista de objetos de aprendizaje a través del buscador del sistema, la obtención de información de un objeto de aprendizaje concreto, o el almacenamiento de un objeto de aprendizaje.

La figura 4 muestra un diagrama de caso de uso del primer patrón de interacción, mediante el cual un estudiante obtiene una serie de objetos de aprendizaje introduciendo en el buscador una "keyword" o un conjunto de "keywords". Se puede observar como el estudiante interactúa con el "Front end" del sistema, capa de presentación (1); a continuación el "Front end" hace una llamada de API al "Back end" (2), capa lógica del sistema; y, por último, esta capa lógica hace una llamada

a la base de datos utilizando para ello la herramienta Mongoose (3), capa de persistencia y acceso a datos. Una vez se obtiene la respuesta de la base de datos MongoDB (4), cada capa va propagando esta respuesta a la capa por la que ha sido invocada anteriormente, hasta llegar al estudiante.

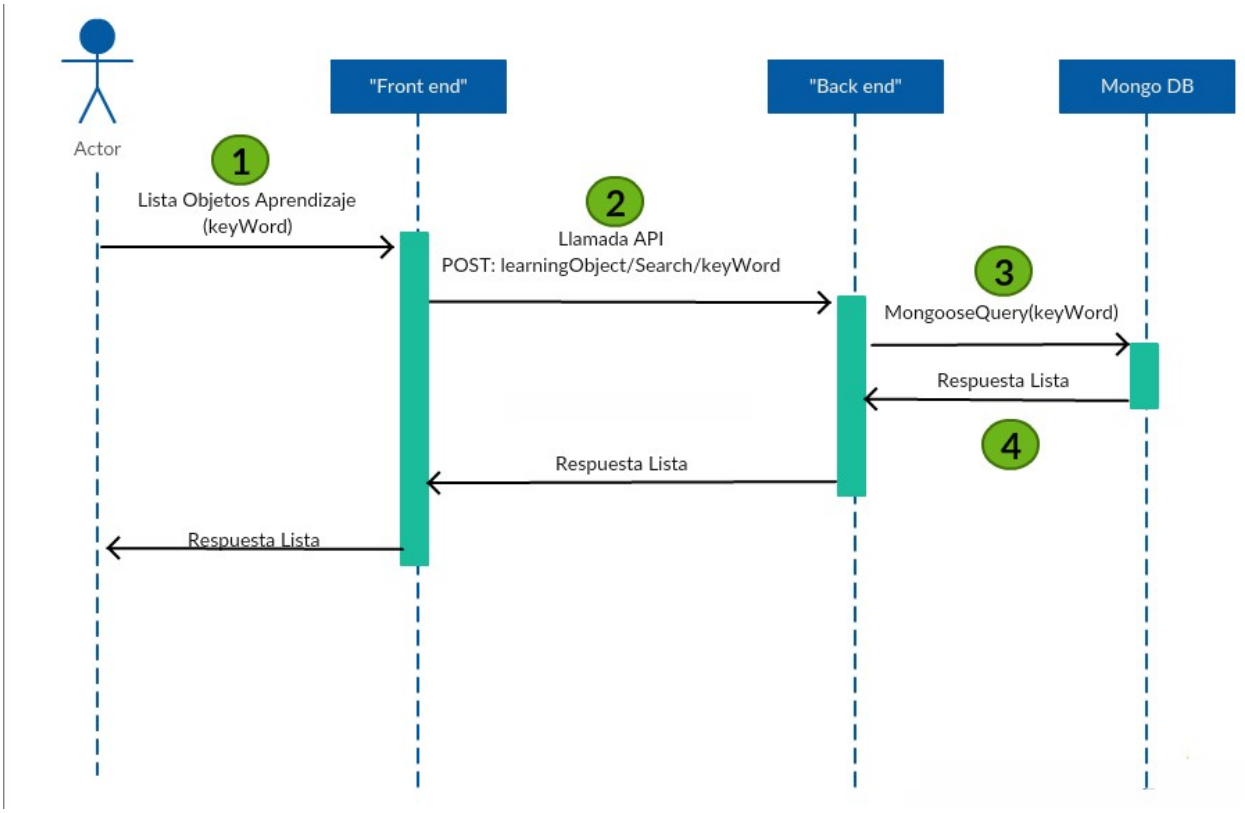


Figura 4: Diagrama de caso de uso de una consulta simple al sistema

Por otro lado aparece un tipo de patrón de interacción más elaborado, que necesita de un preprocesado de datos antes de acceder al repositorio del sistema. Se correspondiente con el análisis de un *Topic Map*, interacción que genera un workflow de ejecución en la capa lógica de negocio. La capa lógica hace uso de llamadas al servicio web de equivalencia de términos para obtener sinónimos, a instancias Java con el fin de llevar a cabo el análisis del *Topic Mapy* a la base de datos, en primer lugar para obtener estructuras necesarias para llevar a acabo el análisis y posteriormente para guardar los resultados generados del mismo.

La figura 5 muestra un diagrama de caso de uso del segundo tipo de interacción, en el que se pueden observar las llamadas que se producen entre diferentes componentes del sistema y el workflow generado en la capa lógica del sistema ("Back end") a la hora de llevar a cabo el análisis de un *Topic Map*. En primer lugar el estudiante realiza una petición al "Front end" con un *Topic Map* (1), y ésta es trasladada hasta el "Back end"(2) generando el workflow. Una vez recibe la llamada la capa lógica, lo primero que hace es realizar dos invocaciones a la capa de persistencia con el fin de obtener una estructura "Learning Indicaror" (3) (donde se guardarán los resultados del análisis) y una estructura "Learning Object" (4) (objeto de aprendizaje al que hace referencia el *Topic Map* introducido por el estudiante) de la base de datos. Ahora es necesario obtener sinónimos

para cada uno de los conceptos y relaciones definidos por el estudiante en el *Topic Map*, por lo que se hará una llamada al servicio web de equivalencia de términos con estos conceptos (5), que a su vez llamará al diccionario online "Wordnik" para generar una respuesta (6). Una vez se tienen los sinónimos se puede proceder al análisis del *Topic Map*, para ello se crea una instancia Java (7), encargada de realizar las operaciones necesarias con los *Topic Map* estudiante y profesor, y se invoca con el *Topic Map* introducido por el estudiante, *Topic Map* de referencia del profesor (obtenido a través de la estructura "Learning Object") y lista de sinónimos generada por el servicio Web diccionario(8). La instancia Java se encargará de llevar a cabo el análisis y devolverá una serie de indicadores al acabarlo(9). Por último, se destruye la instancia Java y se almacenan estos indicadores en un documento de tipo "Learning Indicator" en la base de datos por medio de la capa de persistencia(10).

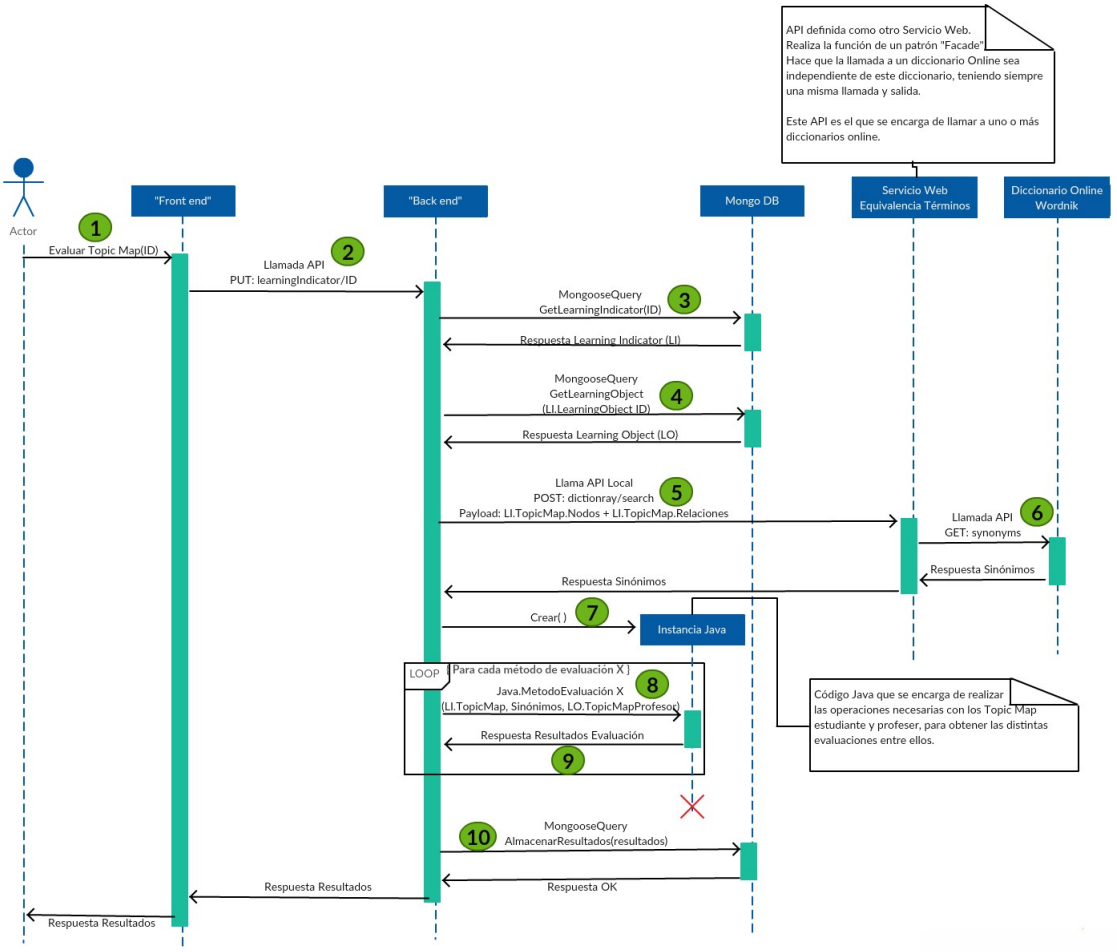


Figura 5: Diagrama de caso de uso de una consulta de análisis de *Topic Map*

3.3. Capa de persistencia y acceso a datos

Es la capa inferior del sistema, encargada de llevar a cabo las interacciones con la base de datos y mantener el modelo de datos de la misma. A continuación se procede a describir el modelo de

datos de la base de datos, el acceso a la base de datos y el modelo de dominio de la misma.

3.3.1. Modelo de datos

Como se ha explicado en la sección 2.5.3, se ha decidido hacer uso de la base de datos MongoDB [10], base de datos NoSQL orientada a documentos [27] en la no se define ningún esquema entidad-relación [35]. Tras llevar a cabo un estudio de qué información es necesaria hacerla persistente en el sistema, se ha definido un modelo de datos, utilizando para ello la herramienta Mongoose. Este modelo de datos consiste en definir la estructura que va a tener cada documento almacenado en la base de datos y qué tipos de datos primitivos van a formar ese documento.

Los datos que necesitan ser almacenados en la base de datos están formados por las cuentas de estudiantes, los objetos de aprendizaje introducidos por los profesores en el sistema, las asignaturas creadas por los profesores, los indicadores de aprendizaje generados de la evaluación de dos *Topic Maps* estudiante-profesor y los propoios *Topic Maps*. Los *Topic Map* creados por los profesores se almacenan junto al objeto de aprendizaje al que hacen referencia, en un mismo documento. Por otro lado, los *Topic Maps* creados por los estudiantes se almacenarán en el mismo documento que se almacenan los indicadores de aprendizaje generados para la evaluación de ese *Topic Map*. De este modo se almacenan cuatro tipo de documentos en la base de datos, se especifican a continuación:

- **Estudiantes**: este documento representa la información necesaria de un estudiante que utiliza el sistema. Se almacena nombre, apellidos, dirección, correo electrónico, lista de asignaturas en las que se encuentra matriculado, nip de acceso y contraseña en formato hash [36].

- **Objetos de aprendizaje**: es la estructura fundamental del sistema. Un objeto de aprendizaje se compone de una serie de atributos que representan información técnica del mismo: identificador, título, descripción, lenguaje, keywords, versión, estatus, contribuidores, formato, tamaño del vídeo, localización del vídeo, duración del vídeo, contexto, dificultad y copyright.

  Posee dos estructuras en las que se almacenan el *Topic Map* del creador del objeto de aprendizaje, y el *Topic Map* propuesto por el profesor como conocimiento que un estudiante debería adquirir a través del objeto de aprendizaje. Este segundo *Topic Map* se utilizará para comparar el *Topic Map* introducido por un estudiante en el sistema referenciando este objeto de aprendizaje. La forma de almacenar los *Topic Maps* es mediante dos atributos que indican el número de nodos y de relaciones que poseen, y dos estructuras, una para los nodos, en la que se almacena el identificador del nodo, nombre del nodo, y valor del nodo; y otra para las relaciones, en la que se almacena el iddentificador de la relación, el nodo del que parte la relación, el nodo hasta el que llega la relación, y el nombre de la relación. En la figura 6 se puede observar un ejemplo de la estructura que presenta un *Topic Map* sencillo en el formato JSON de almacenamiento.

- **Asignaturas**: Esta entidad consta de un código único, un título, una descripción, un curso

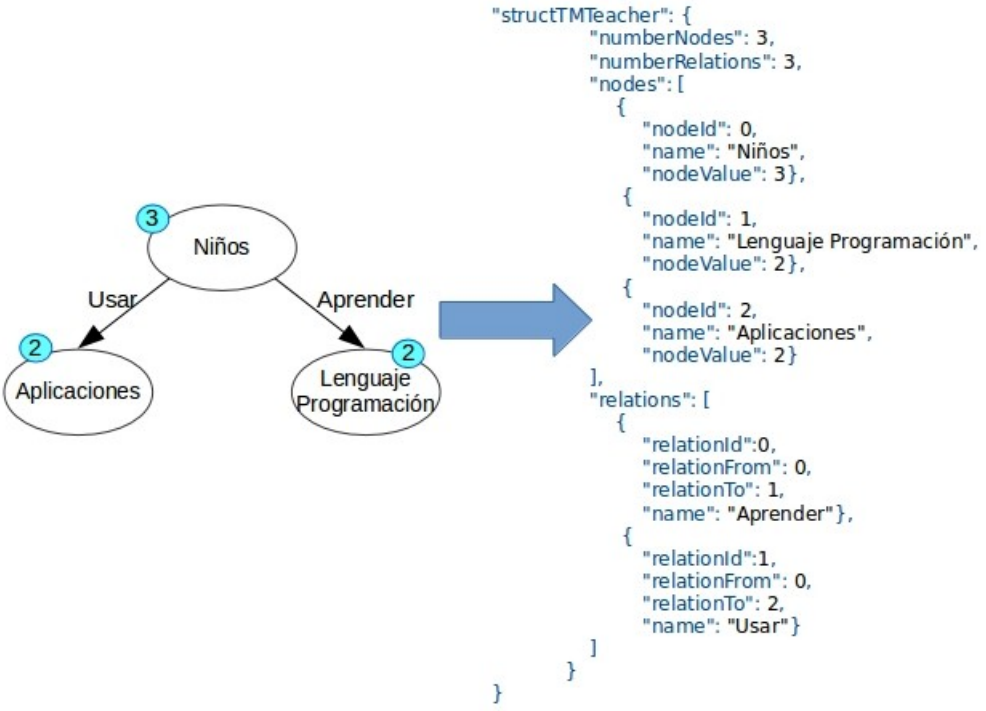


Figura 6: Ejemplo de almacenamiento de *Topic Map*

y una lista de palabras clave. Esta lista de palabras clave es usada para relacionar objetos de aprendizaje con la asignatura dentro de la aplicación.

- **Indicador de aprendizaje**: esta estructura es creada cuando un estudiante almacena un *Topic Map* en el sistema. Posee atributos de nip del estudiante, nombre del objeto de aprendizaje al que hace referencia el Topic Map, identificador único, estructura del *Topic Map* almacenado con el mismo formato que en los documentos de objetos de aprendizaje, ruta de la imagen que describe el *Topic Map*, y estructura de indicadores del *Topic Map*. Al almacenar el *Topic Map* al sistema, la estructura de indicadores se genera vacía, y, una vez que el estudiante evalúa este *Topic Map*, la estructura se rellena con los indicadores generados para el mismo.

Además de los datos almacenados en la base de datos, existen dos carpetas en el servidor en las que se almacenan los vídeos de los objetos de aprendizaje y las imágenes que describen los *Topic Maps*. Para localizar dónde se encuentran estas carpetas se puede observar el Anexo C .ᴱˢtructura de directorios y ficheros del proyecto".

3.3.2. Acceso a la base de datos y modelo de dominio

El acceso a la base de datos se realiza a través de NodeJS, haciendo uso de la herramienta Mongoose. Al poner en marcha el servidor Web, Mongoose establece una conexión con la base de datos que se utilizará en cada una de las lecturas/escrituras de la misma. En esta conexión es

posible especificar la localización de la máquina de la base de datos, por lo que se puede desacoplarla totalmente del servidor y realizar la ejecución de la base de datos en una máquina completamente diferente.

Para realizar la interacción con la base de datos, Mongoose hace uso de schemas y modelos. Un schema define la estructura de los documentos de la base de datos, especificando los tipos primitivos por los que se encuentra formado (int, String, Array, etc.). Por lo tanto, a pesar de que los documentos de MongoDB son por defecto desestructurados, se les puede aplicar una estructura por medio de Mongoose, lo cual permite tener un control de los datos que se introducen en la base de datos, haciendo que cada documento siempre tenga una estructura consistente.

Por otro lado, los modelos actúan sobre los schemas definidos previamente y permiten trabajar con éstos como si fueran objetos, puediendo acceder a cada uno de sus atributos. Cuando se a almacena un cierto documento en la base datos se crea una instancia de un modelo, modelo que a su vez lleva asociada el schema que define la estructura de ese documento. Un modelo posee una interfaz para llevar a cabo las operaciones de persistencia CRUD [37] (Create, Read, Update y Delete), por lo tanto se pueden realizar todo tipo de operaciones necesarias con la base de datos a través de un modelo.

3.4. Capa lógica de negocio

Es la capa más importante del sistema, presenta un API RESTful con una interfaz exterior mediante la cual interaccionar con el mismo, generando un *WorkFlow* cuando se invoca este API. En este *WorkFlow* se encuentran involucrados diferentes servicios y procesos de análisis del sistema. La figura 7 muestra un esquema de los componentes existentes y los procesos que se llevan a cabo en esta capa tras la petición de evaluación de un *Topic Map* por parte de un estudiante, estructura de interacción dinámica más compleja del sistema. El API RESTful permite la interacción con el sistema a través del "Front end", parte del sistema con la que interacciona directamente el estudiante, y cómo una petición desencadena un *WorkFlow* en esta capa. Este *WorkFlow* interacciona con dos servicios, uno de sesión y otro de registro de eventos del sistema. El primero de ellos es el encargado de comprobar que el estudiante está autentificado en el sistema y puede acceder al recurso solicitado, mientras que el segundo graba un registro de cada interacción producida en la aplicación. Además, son llevados a cabo una serie de procesos por este *WorkFlow*; la recuperación del objeto de aprendizaje, y *Topic Map* referencia del mismo, con el que se quiere analizar el *Topic Map* introducido por el estudiante; la llamada al servicio web de equivalencia de términos con el fin de obtener estructuras semánticas equivalentes a la introducida por el estudiante; un proceso de transformación del *Topic Map* y estas estructuras semánticas a formato JSON [29] para el posterior análisis semántico del *Topic Map*; un proceso que realiza la evaluación citada haciendo uso de la tecnología Java, obteniendo de la misma los indicadores de aprendizaje; y, por último, un proceso de almacenamiento de estos indicadores en la base de datos del sistema. A continuación se explica la implementación de estos servicios y procesos.

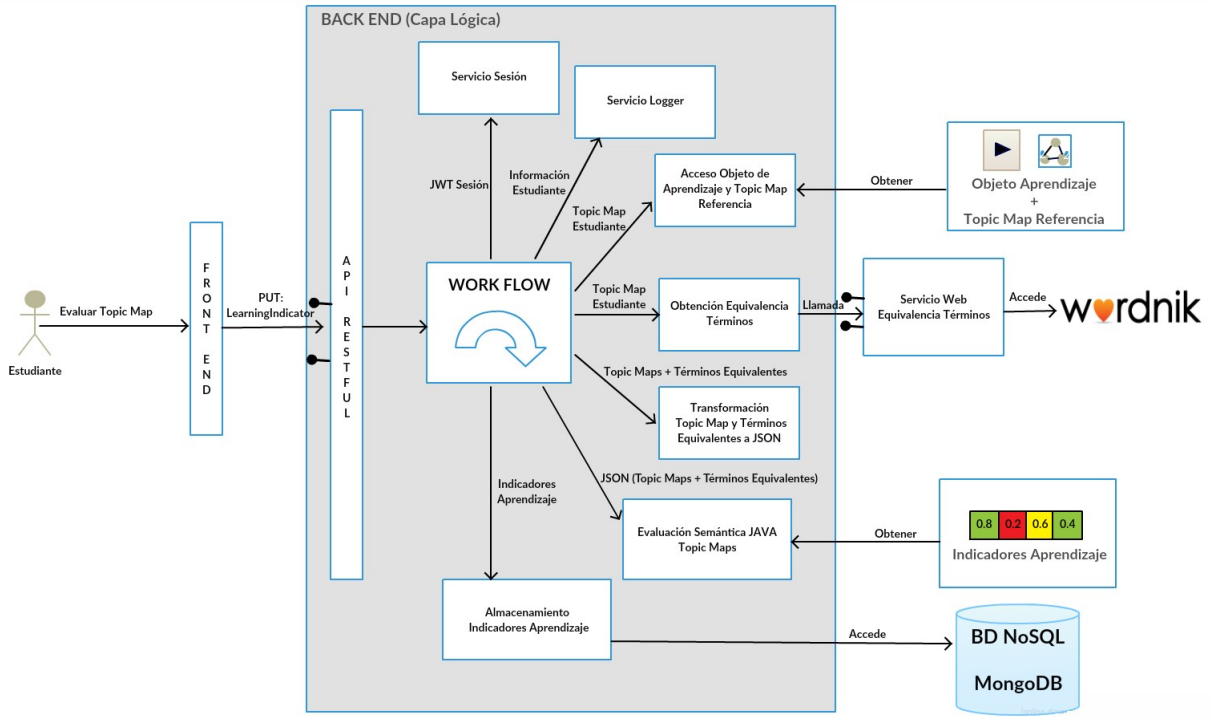


Figura 7: Servicios y procesos llevados a cabo en la capa lógica del sistema

3.4.1. API RESTful

Un API RESTful [1] representa un tipo de arquitectura de desarrollo web apoyada en el estándar HTTP. Permite la creación de servicios y aplicaciones que puedan ser utilizados a través de una conexión HTTP, en este caso concreto se utiliza para realizar las interacciones entre la capa de presentación del sistema y la capa lógica del mismo. Para ello presenta una interfaz de acceso al sistema web basada en recursos. Un recurso es la entidad a la que se desea acceder del sistema. Al ser un API RESTful quedan definidas las operaciones CRUD [37] (Create, Read, Update y Delete) sobre cada recurso. Estas operaciones son implementadas a través del protocolo HTTP mediante los métodos: GET (consultar un recurso), POST (creación de un recurso), PUT (edición de un recurso) y DELETE (borrado de un recurso). Además este API devuelve un código de estado en función del resultado de la invocación a un recurso determinado: 200 (OK), 404 (Recurso no encontrado), 401(Acceso no autorizado) y 500 (Error interno del servidor). Por último, para poder considerarla como RESTful, el API devuelve un enlace en cada respuesta, este enlace sirve para enlazar este recurso con otro con el que se encuentre directamente relacionado.

Los recursos definidos en el sistema se corresponden con el modelo de datos explicado en el apartado anterior. Por lo tanto, existen los recursos: estudiante, objeto de aprendizaje, asignatura e indicador de aprendizaje, cada uno de ellos con las cuatro operaciones CRUD explicadas anteriormente que permiten insertar, eliminar, modificar o actualizar. Por razones de seguridad, ciertas operaciones sólo están permitidas tras el acceso a la aplicación con una cuenta de administración, como son las operaciones de eliminar estudiantes o asignaturas del sistema por ejemplo. En el

anexo D “Especificación de las APIs”se especifica detalladamente este API; quién puede acceder a cada operación, una explicación concreta de qué hace cada una de ellas, así como la sintaxis JSON utilizada para realizar una llamada a cada recurso del sistema y el formato en el que se devuelve cada respuesta.

3.4.2. Servicio Sesión

Este servicio es invocado cada vez que se realiza una petición al API del sistema, la funcionalidad del mismo es detectar si un usuario autentificado en el sistema es el que está intentando llevar a cabo la petición, y, en caso de que lo fuera, comprobar si tiene permiso para realizarla. En función de la comprobación de este servicio se permite al usuario acceder al recurso o es devuelto un código de error 401(Acceso no autorizado).

La implementación del mismo ha sido llevada a cabo haciendo uso de JSON Web-Tokens [38]. Un JSON Web-Token es un estándar abierto de creación de tokens, utilizados para llevar a cabo un intercambio de información cifrada e interacción sin estado entre la parte cliente y servidor de una aplicación. Este token es generado cuando un usuario consigue loguearse correctamente en la aplicación, estableciendo un tiempo de expiración del mismo de 30 minutos. Durante estos 30 minutos el usuario puede acceder a la aplicación gracias al token, siendo este servicio de sesión el encargado de comprobar que el token con el que accede el usuario existe, es válido, no ha caducado y que este usuario concreto está autorizado a acceder al recurso solicitado.

3.4.3. Servicio de registro de eventos del sistema

La principal funcionalidad del servicio es guardar un registro de toda las interacciones relevantes que han tenido lugar en el sistema. Se pueden distinguir dos tipos de interacciones, las llevadas a cabo a través de la red, como son todas las peticiones HTTP al API; y las producidas entre componentes internos del sistema.

Para la implementación del servicio se han utilizado dos herramientas de log diferentes. Por un lado se ha hecho uso de Morgan [39], un middleware encargado de trazar todas las peticiones HTTP que tienen lugar contra la aplicación, proporcionando información útil de las mismas como la IP desde la que se ha realizado la petición o el navegador utilizado. Por otro lado se ha integrado en el sistema Winston [40], una librería asíncrona de NodeJS que permite introducir *logs* personalizados para la ejecución cada componente del sistema. Es posible poner en común las trazas de *log* obtenidas a partir de cada herramienta ya que se introduce un identificador único para cada usuario y sesión en cada traza, tanto en las generadas por Morgan como por Winston.

3.4.4. Obtención equivalencia términos

Este proceso tiene lugar cuando ha sido solicitada la evaluación de un *Topic Map* por parte de un estudiante, y tras haber recuperado de la base de datos el objeto de aprendizaje y el *Topic Map* de referencia elaborado por el profesor. La finalidad del mismo es la extracción de todos los términos existentes en el *Topic Map* introducido por el estudiante, tanto los que etiquetan los nodos del mismo, como los que etiquetan las relaciones. Una vez se obtienen estos términos, se agrupan y se hace una llamada al servicio web de equivalencia de términos con el fin de obtener una lista términos equivalentes para cada uno de ellos. Todos ellos serán usados posteriormente en la evaluación semántica del *Topic Map*, con el fin de detectar estructuras semánticas equivalentes entre los dos *Topic Maps*. La implementación y modo de invocación concreta de este servicio de equivalencia se puede encontrar en la sección 3.5.

3.4.5. Evaluación semántica mediante el uso de Java

Una vez se obtiene la equivalencia de términos se produce la evaluación semántica del *Topic Map* introducido por el estudiante. Esta evaluación tiene lugar en la capa lógica, utilizando la tecnología Java a través de la librería NPM [41] integrada en NodeJS, que permite la invocación de un proyecto desarrollado en Java. Antes de poder hacer uso de esta librería, es necesario transformar el formato de los *Topic Maps* estudiante y referencia profesor, así como los términos equivalentes obtenidos previamente, para que estos sean procesables por Java. Esta transformación se consigue haciendo uso de la función JSON.stringify [43] de JavaScript, con la que se transforman los valores disponibles en JavaScript a cadenas de texto con formato JSON, procesables por Java.

Tras obtener los *Topic Maps* del estudiante, profesor y los términos equivalentes, se puede instanciar la función del proyecto Java "similarityEvaluation", figura 9, con el fin de comenzar la evaluación semántica. La figura 8 muestra la secuencia de los distintos procesos llevados a cabo en esta evaluación. Se puede observar como en primer lugar, número uno (1) de la figura 8, se realiza la transformación de las cadenas con formato JSON recibidas a objetos Java. De este modo se crearán dos objetos "LabelledTopicMap", que dispondrán de métodos para acceder con facilidad a los distintos nodos y relaciones de cada *Topic Map*. A continuación se procede a calcular la lista de definiciones básicas de equivalencia y los indicadores de aprendizaje definidos en el capítulo 2.1.3 de esta memoria. Con este fin se ha definido una clase "SimilarityImplementation", figura 9, encargada de realizar el cálculo de todas estas definiciones de equivalencia cuando es instanciada con dos objetos "LabelledTopicMap" como parámetro. Además, esta clase ofrece métodos para una vez instanciada y calculadas dichas definiciones, poder obtener los indiciadores de aprendizaje haciendo uso de las mismas.

La primera definición procesada en la instanciación es el cálculo de conceptos equivalentes, número (2), para ello se itera sobre los nodos del *Topic Map* profesor, comparándolos con los nodos y las equivalencias de los nodos estudiante. En caso de encontrar una equivalencia semántica se incrementa el número de conceptos equivalentes encontrados, se almacenan en una estructura ArrayList [44]

los índices de los nodos profesor que poseen equivalencia, y, además, en una estructura HashMap [45] se guarda el índice del nodo profesor como clave y el índice del nodo estudiante equivalente como valor para poder acceder a ellos más rápidamente en posteriores cálculos. Tras la obtención de esta primera definición se generan dos matrices (3) de adyacencia [49], una para cada *Topic Map*. Las matrices se componen de un número de filas y columnas igual a número de nodos del Topic Map, estableciendo a 0 aquellas filas y columnas que corresponden a nodos sin equivalencia, y, por tanto, quedando a 1 únicamente aquellas celdas que marcan una relación entre dos nodos con equivalencia en el segundo *Topic Map*. Estas matrices serán de gran utilidad a la hora de calcular las relaciones en estructura entre los *Topic Maps* por su capacidad de representar las relaciones como la intersección de dos índices.

En este punto se procede a calcular la definición de desviación de pesos, punto (4). Para ello se itera sobre el ArrayList de nodos profesor con equivalencia definido anteriormente y, para cada nodo profesor, se accede al nodo estudiante equivalente haciendo uso de la estructura HashMap. De este modo se puede acceder fácilmente al peso de los dos nodos equivalentes y calcular la desviación de pesos.

Posteriormente se lleva a cabo el cálculo de la relación de equivalencia en estructura(5). En este cálculo se usan las matrices de adyacencia definidas anteriormente y la estructura HashMap. Este cálculo se lleva a cabo iterando sobre la matriz de adyacencia profesor, y, para cada relación almacenada en la misma, se comprueba si los dos nodos equivalentes a los pertenecientes a esta relación también presentan una relación en la matriz de adyacencia estudiante (un 1 entre estos dos nodos).

A continuación se procesa la última definición de equivalencia, el cálculo de relación en equivalencia semántica(6). Este cálculo sigue el mismo proceso que el citado en el párrafo anterior, a diferencia de que aparte de comprobar si existe un 1 en la matriz de adyacencia estudiante, hay que comprobar si el término que etiqueta las dos relaciones es equivalente, utilizando para ellos los términos equivalentes obtenidos del servicio web.

Una vez se han procesados todas las definiciones, hay que calcular los indicadores de aprendizaje similitud entre conceptos (7), similitud en relación (8), similitud en adyacencia (9) y similitud de *Topic Map* (10). Este proceso de cálculo es secuencial ya que cada indicador necesita ciertos resultados generados en el cálculo de otros indicadores. La implementación es simple, ya que consiste en aplicar las fórmulas especificadas en el Anexo A "Lógica matemática de los indicadores de aprendizaje".

El último indicador de aprendizaje que falta por calcular se corresponde con los conceptos no detectados (11). Se utilizan el ArrayList con los índices de nodos profesor con equivalencia y el objeto "LabelledTopicMap" que contiene el *Topic Map* profesor. Iterando sobre los índices de los nodos del objeto y los del ArrayList se pueden obtener fácilmente cuáles son los nodos que no ha introducido el estudiante, ya que no poseen aparecerán en el ArrayList de equivalencia de nodos.

Por último, una vez se han calculado todos los indicadores, éstos se almacenan en un array (12)

y son devueltos, con el fin de que NodeJS pueda almacenarlos en la base de datos.

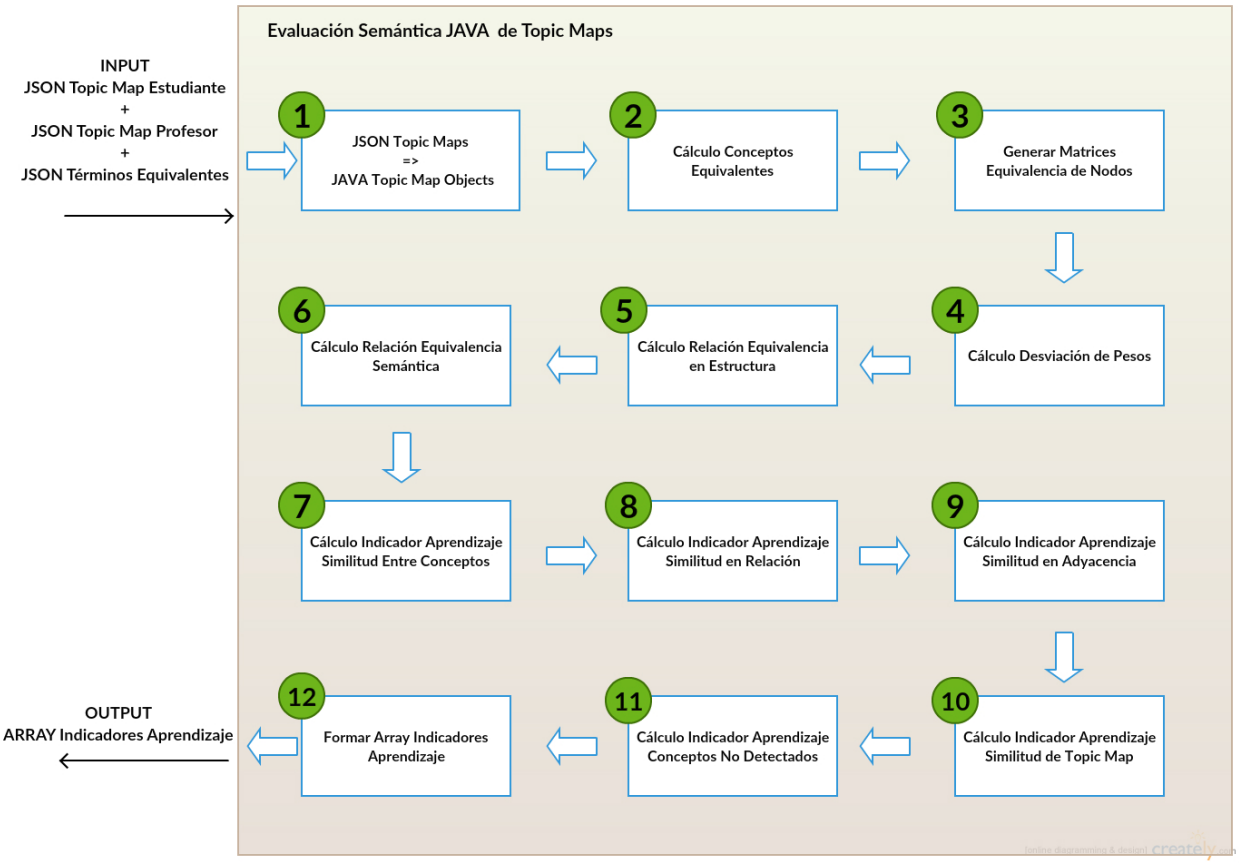


Figura 8: Procesos Java llevados a cabo en la evaluación de un *Topic Map*

3.5. Servicio Web de equivalencia de términos

La obtención de términos equivalentes para los conceptos y relaciones de los *Topic Map* es un punto crítico del sistema. En la actualidad, la generación de estos términos se lleva a cabo mediante una llamada al diccionario online Wordnik, pero el servicio se ha implementado escribiendo un código reutilizable y no dependiente de un único servicio web externo al sistema. Esto explica la implementación de un servicio web propio que se encargue de realizar estas llamada a otros servicios de equivalencia de términos externos.

Este servicio web de equivalencia de términos implementado realiza la función de un patrón de diseño "Facade" [34], haciendo que la obtención de términos equivalentes sea independiente del servicio externo. Cuando se quiere obtener esta lista de términos, se hace una llamada al servicio web de equivalencia propio, el cual siempre va a tener una misma llamada y respuesta; internamente, se encargará de realizar las llamadas concretas a cada servicio web externo. De este modo se consigue que el servicio web externo de obtención de términos pueda ser modificado sin tener que cambiar el código del resto de las capas del sistema.

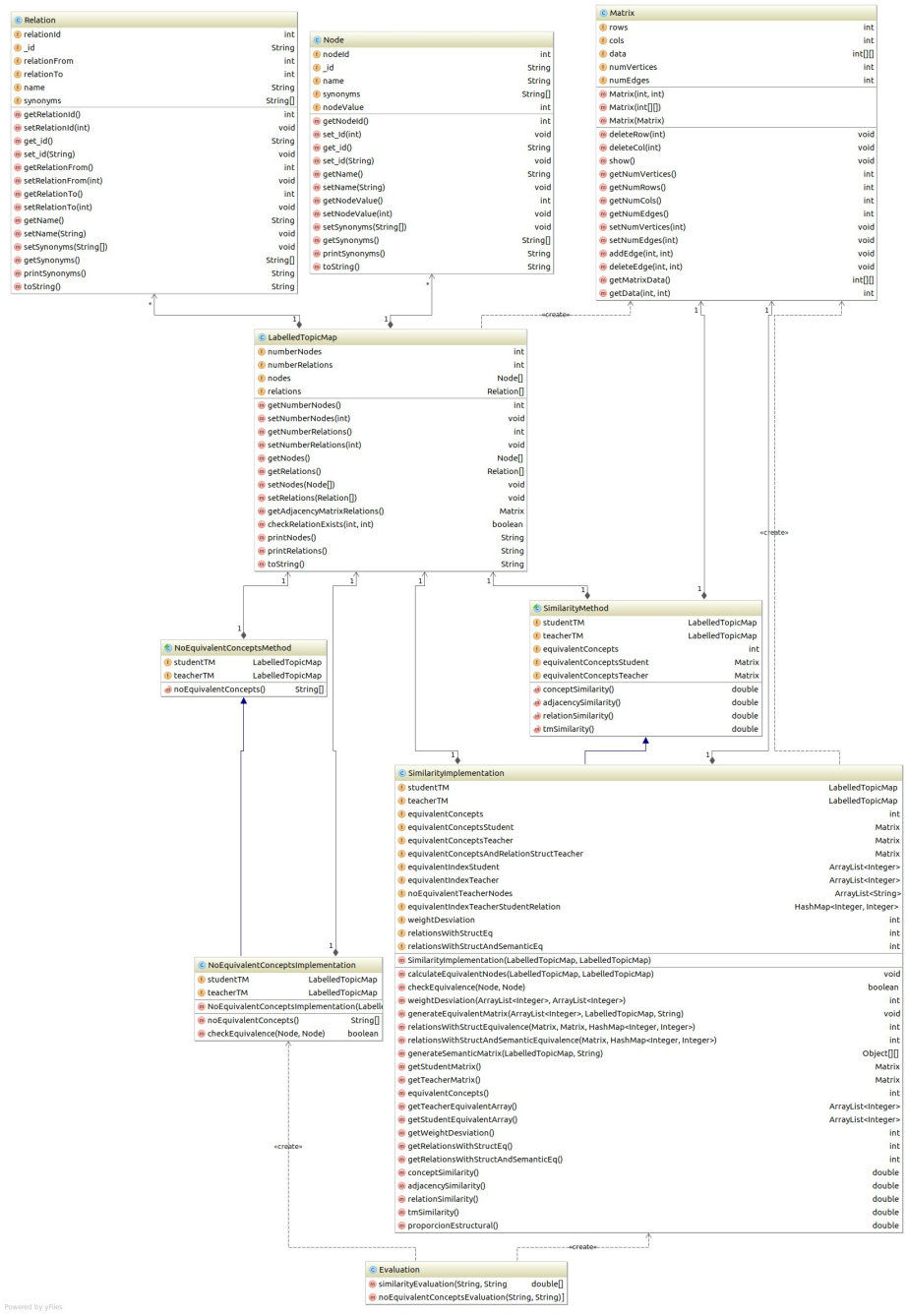


Figura 9: Diagrama de clases del proyecto Java

La implementación de este servicio Web se muestra en la figura 10. Se puede observar cómo se ha definido un API RESTful, del mismo modo que en la capa lógica, que en este caso cuenta con un único recurso "diccionario" accesible vía HTTP. Tanto la llamada como la salida del servicio Web presentan siempre una misma estructura. Por un lado, la llamada consiste en una llamada HTTP de tipo POST, en la que se introduce un vector con todos los términos para los que se desea obtener equivalencias. Por otro lado, la salida se corresponde con una estructura JSON en la que primero se indica si se ha producido algún tipo de error o no, y a continuación se muestra una cadena

"message" compuesta de pares "input", "synonyms"; donde "input" corresponde con una palabra introducida en el Array de entrada, en el mismo orden, y "synonyms" con la lista de términos equivalentes generados para esa palabra. Como se puede ver en la figura, el servicio se encarga de transformar la lista de palabras que recibe como entrada al formato concreto del servicio web externo con el que se vaya a comunicar, en este caso Wordnik, y, posteriormente, de transformar la respuesta de este servicio externo al formato JSON de salida especificado previamente.

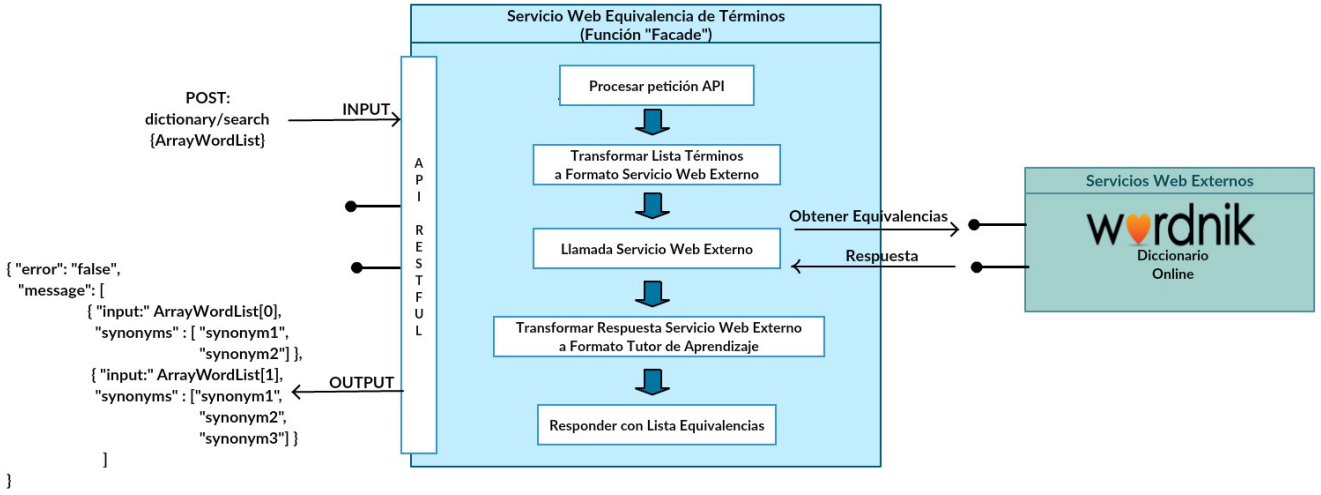


Figura 10: Implementación servicio web equivalencia de términos

3.6. Capa de presentación

Capa superior del sistema, proporciona un interfaz cliente a los usuarios. En la figura 11 se observa el mapa de navegación de la aplicación. Las pantallas "Pantalla Licencia Aplicación", "Pantalla Listado Indicadores Aprendizaje", "Pantalla Manual de Usuario" y "Pantalla Asignaturas – Búsqueda Objeto Aprendizaje" son alcanzables desde cualquier pantalla del sistema por encontrarse situadas en la barra de navegación, y, del mismo modo, la pantalla "Pantalla Inicio Sesión", ya que es posible iniciar cerrar sesión desde cualquier pantalla del sistema. En el Anexo E "Manual de usuario"puede encontrarse una descripción detallada de cada pantalla, así como un proceso guiado de cómo utilizar la aplicación.

Como se ha explicado anteriormente la implementación se basa en del patrón de diseño MVC (Modelo, Vista, Controlador), el cual es muy útil para desarrollar un código claro y estructurado en el "Front end" de la aplicación. Se basa en separar la visualización de la lógica de negocio, por lo que se tienen por un lado los ficheros HTML y CSS correspondientes a la interfaz visual de la aplicación y por otro los ficheros JavaScript que se encargan de obtener y enviar datos a la capa lógica, los controladores. De este modo se evita tener en un fichero HTML el código HTML junto con funciones JavaScript, lo que suele acabar desembocando en un código poco o nada legible.

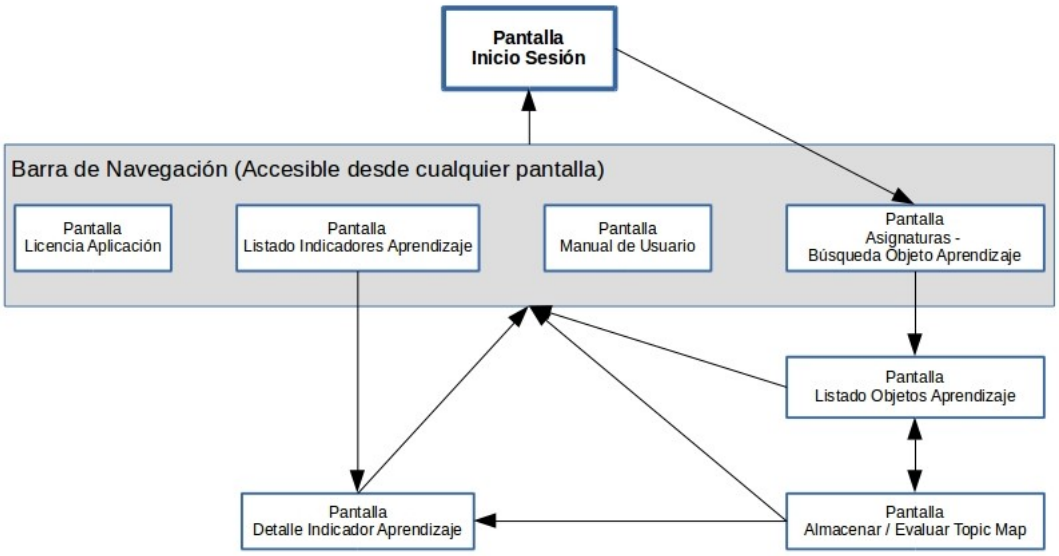


Figura 11: Mapa de navegación de la aplicación

Siguiendo este patrón, MVC, cada pantalla de la barra de navegación se compone de estos tres ficheros, HTML, CSS y JavaScript; además, existen otros dos ficheros JavaScript en esta capa, "sessionService.js" y "app.js". El primero de los últimos dos es el encargado de manejar los JSON Web-Token que definen la sesión del usuario en la parte cliente. El segundo, se encarga de invocar al controlador JavaScript de la página correspondiente en función de lo tecleado en el navegador por el usuario.

Boostrap por su parte se limita a modificar el aspecto visual del sistema, proporcionando botones, formularios, plantillas o barras de navegación visualmente muy agradables y fácilmente personalizables. En las figuras 12 y 13 se muestran las dos pantallas más representativas la interfaz definitiva del sistema desarrollado, se puede observar que sus contenidos están escritos en inglés, pues es el idioma principal de la aplicación.. La figura 12 se corresponde con la pantalla "Pantalla Almacenar / Evaluar Topic Map" del mapa de navegación. En ella el estudiante es capaz de visualizar el vídeo del objeto de aprendizaje accedido y, a continuación, de introducir un *Topic Map* para este objeto de aprendizaje. Para la introducción del mismo, en primer lugar, el estudiante define el número de nodos y relaciones que posee a través los campos de entrada que aparecen en la esquina superior derecha de la pantalla, generándose de este modo la estructura "*Topic Map* information struct", justo debajo, en la que debe incluir todos los detalles de nodos y relaciones. Haciendo uso del botón "Choose a file", que se encuentra en la parte inferior derecha de la pantalla, es capaz de introducir una imagen de este Topic Map. Por último, una vez completados todos los campos se desbloquea el botón "Submit Topic Map", botón más inferior, mediante el cual se almacena el *Topic Map* en el sistema. Tras almacenarse, aparece un nuevo botón en la pantalla que permitirá al estudiante la evaluación del mismo.

Por otro lado, la figura 13 se corresponde con la pantalla "Pantalla Detalle Indicador Aprendizaje"

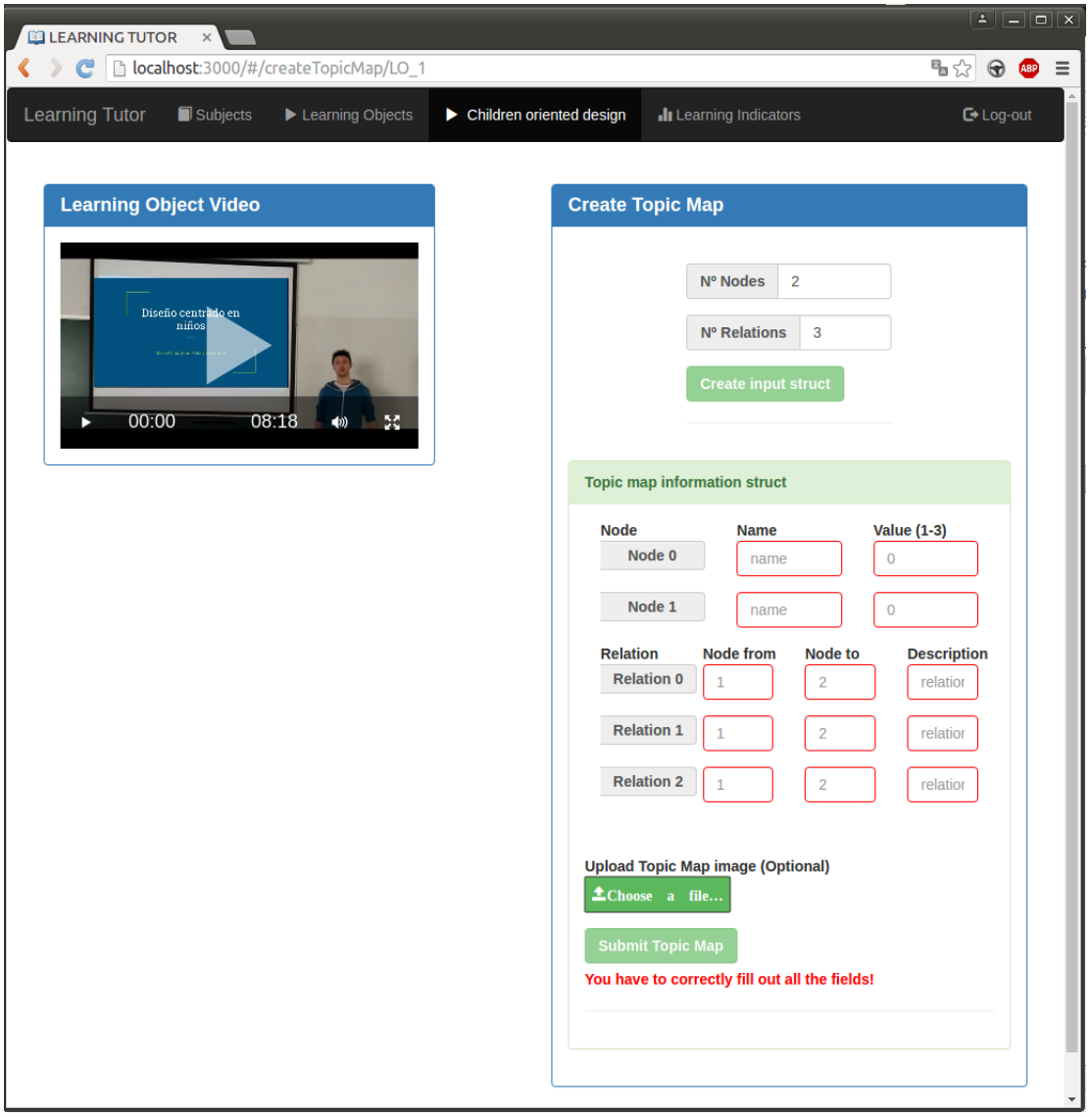


Figura 12: Pantalla de introducción y evaluación de un *Topic Map* en el sistema

del mapa de navegación. En la parte inferior de la pantalla, el estudiante es capaz de observar los indicadores obtenidos para la evaluación de uno de sus *Topic Maps*. Por un lado, a la izquierda, aparecen los cuatro indicadores de similitud, y por otro puede ver los conceptos que aparecían en el *Topic Map* del profesor de referencia que no ha sabido identificar. Además, en la parte superior, se muestra la imagen que adjuntó a la hora de almacenar el *Topic Map*, y es capaz de descargar esta imagen y la estructura del *Topic Map* introducida mediante los links "Download Image" y "Download TM" respectivamente.

Por último, para la implementación de la interfaz se han tenido en cuenta la accesibilidad en cuanto a los colores de la aplicación y los contrastes generados entre ellos. Con esto se busca que cualquier persona con deficiencias en la visión de colores sean capaces de visualizar la página web correctamente. Para ello se ha utilizado la herramienta Checkmycolours [46], la cual se encarga de

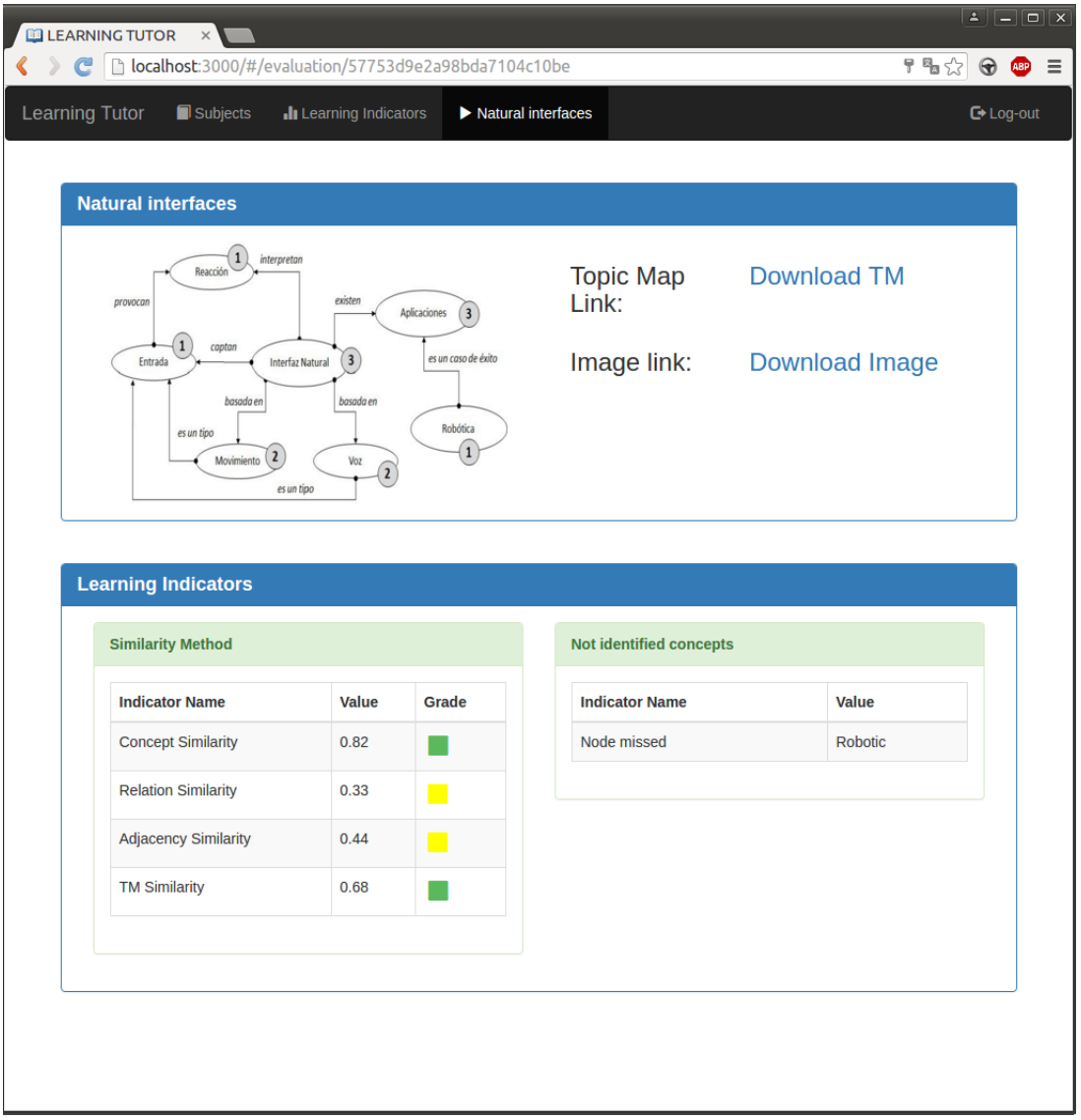


Figura 13: Pantalla de detalle de los indicadores de aprendizaje

revisar todos los colores e indicar si entre algunos de ellos no se presenta el suficiente contraste. Otro punto que se ha tenido en cuenta han sido las jerarquías de tamaños, es decir, el tamaño de la letra en los distintos títulos y secciones de la aplicación. Se mantiene una jerarquía uniforme en todo el sistema, utilizando el mismo tipo de letra y tamaño para títulos principales, y decrementando este tamaño para subsecciones. Con esto se consigue hacer mucho más accesible el sistema a los usuarios, ya que estos serán capaces de identificar los conceptos importantes más rápidamente.

4. Evaluación de los algoritmos de análisis

Este capítulo tiene como objetivo presentar los tests y evaluaciones llevados a cabo en el proyecto. En primer lugar se muestran los test realizados para los indicadores de similaridad implementados en el sistema, en los que se comparan los resultados obtenidos en la evaluación automática de una serie de *Topic Maps* por el sistema, con los obtenidos por un profesor manualmente en base a las fórmulas especificadas en el Anexo A "Lógica matemática de los indicadores de aprendizaje".. A continuación muestra una evaluación del tiempo empleado en realizar el análisis semántico entre dos *Topic Maps*.

4.1. Evaluación de los indicadores de similaridad

El principal problema a la hora de aplicar las fórmulas de similaridad es que estas hacen uso de equivalencias semánticas entre conceptos. Detectar una equivalencia semántica puede ser subjetivo en un proceso manual, ya que depende del criterio del profesor que evalue decir si los términos son equivalentes o no. Sin embargo, en el proceso automático, se hace uso de servicios de equivalencia de términos externos, por lo que los resultados de aplicar estas fórmulas de similaridad podrían no coincidir con los resultados obtenidos manualmente. Por ejemplo, dos términos que un profesor puede considerar como equivalentes semánticamente, un servicio web de equivalencia los podría considerar como no equivalentes entre sí, generándose de esta forma valores de indicadores de aprendizaje diferentes.

Para estudiar el grado de similitud que presentan los resultados proporcionados por la evaluación automática de aplicación desarrollada, con respecto a los resultados obtenidos a través de la evaluación manual llevada a cabo por profesores, se han realizado tests sobre los indicadores de aprendizaje. En concreto se han estudiado los resultados obtenidos para el indicador de aprendizaje "similitud de Topic Map", indicador de aprendizaje más representativo del sistema ya que mide la similitud entre un *Topic Map* desarrollado por el estudiante para un objeto de aprendizaje concreto y el desarrollado por el profesor para ese mismo objeto de aprendizaje, haciendo uso de los resultados generados por el resto de indicadores de aprenizaje. Este método devuelve un valor numérico entre 0.0, y 1.0, indicando una mayor similitud entre *Topic Maps* cuanto mayor sea este valor.

Para la realización de los tests se han empleado cuatro objetos de aprendizaje diferentes: "Diseño centrado en niños (DCN)", "Interfaces naturales (IN)", "Computación afectiva (CA)" e "Interfaz cerebro ordenador (ICO)". Cada uno de estos objetos se compone de un vídeo desarrollado por estudiantes de la asignatura "Diseño centrado en el usuario. Diseño para la multimedia", impartida en el curso 2015-2016 en la Universidad de Zaragoza. Catorce estudiantes de la asignatura han desarrollado *Topic Maps* mostrando los conocimientos adquiridos tras visualizar el vídeo de cada uno de los objetos de aprendizaje (a excepción de los objetos de aprendizaje desarrollados por ellos mismos). Posteriormente, estos *Topic Maps* han sido analizados y comparados con el *Topic Map*

de referencia generado por el profesor para cada objeto de aprendizaje, tanto con la aplicación de forma automática, como por un profesor de forma manual. Los resultados obtenidos de estos tests se pueden observar en la tabla 2. Cada par de columnas representa los resultados obtenidos para un objeto de aprendizaje concreto, comparando la evaluación automática llevada a cabo por la aplicación (APP) con la evaluación manual del profesor (MANUAL). Las filas representan los resultados obtenidos por cada estudiante. Que los estudiantes no dispongan de datos para ciertos objetos se debe a que esos objetos de aprendizaje han sido desarrollado por el grupo al que pertenece el estudiante.

	DCN		IN		CA		ICO	
	APP	MANUAL	APP	MANUAL	APP	MANUAL	APP	MANUAL
E 1	0,47	0,52	Sin datos	Sin datos	Sin datos	Sin datos	0,35	0,40
E 2	Sin datos	Sin datos	Sin datos	Sin datos	0,38	0,35	0,72	0,81
E 3	Sin datos	Sin datos	0,40	0,46	0,32	0,32	0,27	0,41
E 4	0,41	0,41	0,27	0,30	0,24	0,24	0,25	0,35
E 5	0,17	0,24	0,33	0,33	0,23	0,23	0,51	0,53
E 6	0,48	0,53	0,39	0,46	0,35	0,39	Sin datos	Sin datos
E 7	0,42	0,53	0,28	0,32	0,35	0,30	Sin datos	Sin datos
E 8	0,31	0,31	Sin datos	Sin datos	0,24	0,24	Sin datos	Sin datos
E 9	0,32	0,41	Sin datos	Sin datos	0,27	0,33	0,39	0,51
E 10	0,37	0,40	Sin datos	Sin datos	0,19	0,26	0,35	0,37
E 11	0,32	0,44	Sin datos	Sin datos	0,37	0,37	0,41	0,54
E 12	Sin datos	Sin datos	0,35	0,47	0,29	0,29	0,35	0,32
E 13	0,41	0,49	0,38	0,42	Sin datos	Sin datos	0,18	0,27
E 14	0,35	0,47	0,40	0,41	0,37	0,40	0,29	0,40
Media	0,37	0,43	0,35	0,40	0,30	0,31	0,37	0,45

Tabla 2: Evaluación de la precisión del indicador de similitud de *Topic Map*

En general los resultados obtenidos entre la evaluación llevada a cabo con la aplicación y la evaluación llevada a cabo por el profesor son similares. El análisis del objeto de aprendizaje "Computación afectiva" ha obtenido una diferencia media de 0.01 puntos, lo que supone una diferencia prácticamente despreciable. Por otro lado, las diferencias medias presentes en los análisis del resto objetos de aprendizaje son algo mayores, 0.0463 para "Interfaces naturales", 0.0654 para "Diseño centrado en niños" y 0.0764 para "Interfaz cerebro ordenador". A pesar de que éstas sean algo mayores, haciendo la media de las diferencias de todos los objetos de aprendizaje se obtiene una similitud del 87.5 % entre los resultados obtenidos con el análisis manual y los resultados obtenidos automáticamente por el sistema, lo que implica que se están obtenido una equivalencia de términos de forma automática muy similar a la que se obtiene manualmente por el criterio del profesor. Este porcentaje de similitud se puede considerar como muy satisfactorio teniendo en cuenta lo explicado al principio del capítulo, que el proceso manual de análisis es subjetivo a la hora de determinar equivalencias de términos y por tanto no siempren coinciden las equivalencias detectadas por los profesores con las detectadas por un servicio web de equivalencia de términos, haciendo que estos

indicadores de aprendizaje no muestren resultados idénticos.

Aparte de la similitud numérica entre resultados, hay que tener en cuenta también el color que representa cada resultado. El color rojo implica que el estudiante ha obtenido un nivel de aprendizaje bajo a través del objeto de aprendizaje (resultado 0 – 0.30), el amarillo que el aprendizaje ha sido medio (0.31 – 0.49) y el verde que el aprendizaje ha sido alto(mayor que 0.49). Por lo tanto, es interesante que a pesar de las pequeñas diferencias numéricas entre resultados, la aplicación genere resultados para los indicadores de aprendizaje que se encuentren en el mismo rango de color que los generados manualmente. Observando los resultados se puede apreciar que de los 42 *Topic Maps* analizados, 32 de ellos mantienen los resultados de aprendizaje en el mismo rango tras el análisis automático, lo que supone el 76.19 % de éstos. En cuanto a los 10 restantes, se puede observar que 5 de ellos presentan una diferencia de 0.06 puntos o menos con respecto al resultado manual, y, que por tanto, han cambiado de rango por encontrarse el resultado manual en la frontera entre dos rangos diferentes. Los 5 resultados restantes presentan una diferencia aproximada de 0.1 puntos con respecto al resultado manual, es una diferencia algo mayor, pero en términos generales éstos se pueden considerar como unos resultados bastante buenos teniendo en cuenta las limitaciones a la hora de determinar conceptos equivalentes nombradas anteriormente.

4.2. Evaluación del rendimiento del análisis semántico

Otro tipo de evaluación que es interesante llevar a cabo es una evaluación del tiempo de análisis de un *Topic Map*.

Para llevar a cabo la evaluación se han vuelto a utilizar los cuatro objetos de aprendizajeden el apartado anterior: “Diseño centrado en niños (DCN)”, “Interfaces naturales (IN)”, “Computación afectiva (CA)” e “Interfaz cerebro ordenador (ICO)”. En esta ocasión se ha medido el tiempo empleado por el sistema para llevar a cabo el análisis automático de cinco *Topic Maps* desarrollados para cada objeto de aprendizaje. Este análisis ha sido ejecutado en un ordenador que cuenta con las siguientes especificaciones: procesador Intel Core i7-4700MQ CPU @ 2.40GHz x 4 y 16GB de RAM.

Teniendo en cuenta el diagrama de caso de uso del análisis de un *Topic Map* mostrado en la figura 5, se pueden definir dos puntos críticos en dicho análisis. Por un lado, la llamada al servicio web externo de equivalencia de términos; por otro lado, el tiempo de análisis empleado por la instancia Java para llevar a cabo el análisis semántico del *Topic Map*. En la tabla 3 se muestran los tiempos utilizados en el análisis total del *Topic Map* (T evaluación) (desde que llega la petición al “Back end”, hasta que se devuelven los resultados al estudiante), en las llamadas realizadas al servicio Web externo (API), y en los cálculos realizados por la instancia Java para cada uno de los 20 *Topic Maps*. En la última columna, se muestra la suma del número de conceptos y relaciones que presenta el *Topic Map* a analizar. Finalmente, en las últimas dos filas, se han incluido las medias del tiempo empleado en cada parte de la evaluación, y las desviaciones estándares, con el fin de poder analizar mejor los resultados obtenidos.

Topic Map	T evaluación(ms)	T llamadas API(ms)	T cálculo Java(ms)	Nº Conceptos + Relaciones
DCN TM 1	876	844	7	16
DCN TM 2	927	884	7	13
DCN TM 3	739	718	6	5
DCN TM 4	803	779	4	15
DCN TM 5	931	900	5	13
IN TM 1	964	942	5	7
IN TM 2	732	679	5	7
IN TM 3	998	965	7	11
IN TM 4	733	691	7	11
IN TM 5	871	835	6	15
CA TM 1	810	771	4	16
CA TM 2	791	765	4	15
CA TM 3	713	684	4	7
CA TM 4	642	617	4	6
CA TM 5	736	698	3	11
ICO TM 1	809	775	7	13
ICO TM 2	614	590	5	13
ICO TM 3	665	642	4	16
ICO TM 4	614	593	4	11
ICO TM 5	621	592	3	11
Media	779,45	748,2	5,05	
Desviación	119,41	117,61	1,39	

Tabla 3: Evaluación de rendimiento del sistema

Observando los resultados se puede comprobar que el punto del análisis en el que más tiempo se emplea es la llamada al API del servicio web externo. Independientemente del número de conceptos y relaciones que presente el *Topic Map* a analizar (se realiza una llamada al servicio web externo por término), el tiempo de llamada al servicio web externo se mantiene muy similar. Esto es debido a que todas estas llamadas son lanzadas al mismo tiempo de forma concurrente, por lo tanto, el tiempo de llamada total viene marcado por la última llamada en ser respondida, y, a su vez, el tiempo crítico de cada llamada lo define la latencia de red, ya que el tiempo empleado entre el lanzamiento de una llamada y otra es despreciable al lado de ésta. Por lo tanto, es la latencia de red lo que marca el tiempo total de evaluación, que en general suele ser constante. Esto se traduce en unos tiempo de evaluación totales muy similares, como se puede ver observando la desviación estándar. El tiempo medio de una evaluación de *Topic Map* se encuentra por debajo de un segundo, lo que se puede considerar un tiempo muy bajo de espera, ya que es prácticamente imperceptible para el usuario.

5. Gestión del proyecto

El objetivo de este capítulo es describir la metodología organizativa utilizada durante el desarrollo del proyecto así como mostrar los esfuerzos realizados en la elaboración del mismo.

5.1. Metodología organizativa

La implementación del proyecto se ha basado en el uso de tecnologías ágiles [47]. Este tipo de tecnologías se basan en el desarrollo de proyectos de manera iterativa e incremental, donde la solución va evolucionando constantemente. El uso de este tipo de tecnología ha hecho necesaria la planificación de reuniones semanales entre el estudiante y los directores de proyecto. En estas reuniones se estudia el trabajo realizado hasta la fecha, se proponen mejoras a este trabajo y, a su vez, se definen nuevas tareas y fechas de entrega para estas, consiguiendo de este modo avanzar iterativamente a la vez que se solucionan diversos problemas que van apareciendo.

El principal beneficio de utilizar este tipo de metodología reside en la posibilidad que tienen los directores de proyecto (cliente final en este caso) de acceder a una aplicación usable desde un primer momento, pudiendo interaccionar con ella y viendo como va evolucionando el desarrollo de la misma. De esta forma es más fácil proponer cambios, mejoras y detectar problemas en la implementación de esta aplicación.

En la figura 14 se puede observar un diagrama Gantt [48] que marca las fechas de inicio y fin de cada fase llevada a cabo en el proyecto. Al haber utilizado tecnologías ágiles, se puede observar con facilidad como coexisten varias fases a la vez en el desarrollo del proyecto, como son las fases de análisis, diseño e implementación del sistema. Se van produciendo iteraciones sobre las mismas y avanzan todas a la vez. El proyecto ha sido desarrollado en 5 meses, en los 3 primeros (Abril – Junio) se han realizado esfuerzos continuados, estableciendo una reunión semanal, pero no a tiempo completo, ya que a la vez que se compaginaba con el resto de asignaturas del grado. Los dos últimos meses (Julio – Agosto) han servido para pulir detalles de la implementación (parte visual), llevar a cabo la realización de la memoria y realizar el despliegue final del sistema en un servidor *cloud*.

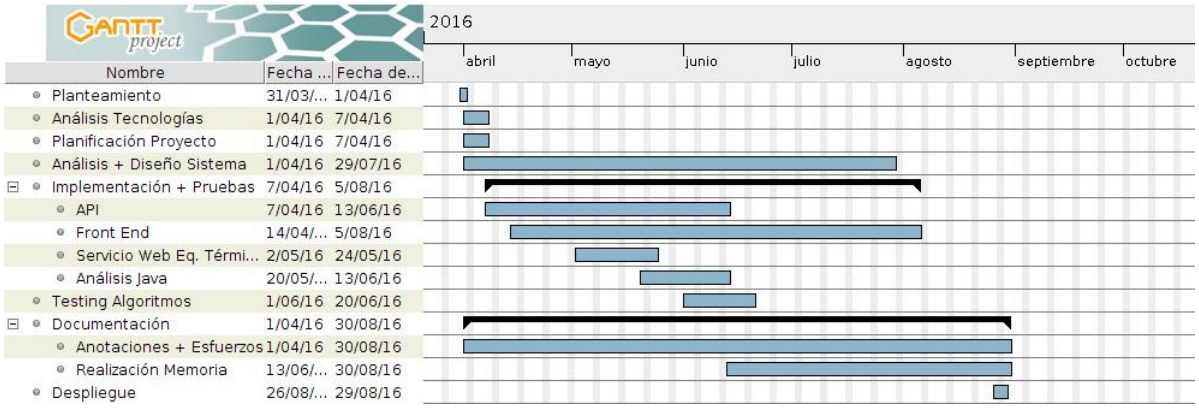


Figura 14: Diagrama Gantt de planificación del proyecto

5.2. Esfuerzos

Durante los 5 meses de duración del proyecto se han invertido en el mismo 350.5 horas. En la tabla 4 se muestra la distribución de las horas invertidas en cada una de las tareas definidas en el diagrama Gantt del apartado anterior. Puede observarse como la fase de implementación (147.5) ha sido las que más esfuerzos ha necesitado. Además, se incluyen las horas invertidas en reuniones con los directores del proyecto, ya que su número es considerable y han sido muy importantes en este desarrollo basado en tecnologías ágiles.

Fase	**Horas**	**Agrupado**
Planteamiento	4	
Análisis Tecnologías	10	
Planificación Proyecto	6	
Análisis + Diseño Sistema	47.5	
Análisis + Planificación		**67.5**
Implementación API	46.5	
Implementación Front end	38	
Implementación Servicio Web	26.5	
Implementación Análisis Java	36.5	
Implementación		**147.5**
Testing Algoritmos	29	
Despliegue	8	
Realización Memoria	77.5	
Reuniones	21	
TOTAL	350.5	

Tabla 4: Distribución de horas invertidas en el proyecto

6. Conclusiones y trabajo futuro

Este último capítulo muestras las conclusiones obtenidas de la realización del proyecto, el trabajo futuro con el que continuar desarrollando el sistema y la opinión personal del estudiante que lo ha llevado a cabo.

6.1. Conclusiones

El objetivo principal del proyecto, el desarrollo de una aplicación accesible vía web que mejore el contexto de aprendizaje en el ámbito académico se ha conseguido realizar con éxito. Desde el punto de vista del profesor se ha conseguido permitir el almacenamiento de objetos de aprendizaje que sirven como material educativo en el sistema, que sus estudiantes puedan representar y almacenar los conocimientos adquiridos por medio de estructuras semánticas, y, que el análisis de las mismas, se realice de forma automática. Desde el punto de vista de los estudiantes se ha conseguido que puedan obtener recursos de aprendizaje fácilmente accesibles, que sean capaces de representar su conocimiento y éste sea evaluado automáticamente, y, que tras esta evaluación, se les proporcione una retroalimentación en forma de indicadores de aprendizaje a cerca de hasta que punto han comprendido los recursos de aprendizaje que han utilizado.

Para ello se han utilizado tecnologías de desarrollo web punteras y en un gran auge como son NodeJS y AngularJS, y, del mismo modo, se ha utilizado una base de datos novedosa, NoSQL, y que se adapta a la perfección a estas tecnologías como es MongoDB. El uso de estas tecnologías en constante evolución, con una gran comunidad detrás de ellas, hace que el proyecto sea fácilmente actualizable por una gran cantidad de desarrolladores y pueda evolucionar satisfactoriamente en los próximos años.

Durante la realización del mismo han surgido varios problemas y retos, como la integración de servicios web externos manteniendo la escalabilidad del sistema e independencia entre capas, o la representación y almacenamiento de las estructuras semánticas *Topic Map* en la aplicación. Gracias a la realización de un proceso iterativo, la planificación de distintas reuniones y una buena metodología de análisis y diseño se consiguió abordar todos ellos de manera satisfactoria.

Como resultado se ha conseguido dejar implantado un sistema completamente operativo, extensible, fácilmente escalable, distribuible y que presenta una interfaz de usuario dinámica y accesible.

6.2. Trabajo futuro

La aplicación desarrollada abarca distintos puntos de vista (profesores y estudiantes), consta de una parte lógica con cálculos complejos e interacciona con servicios web externos, es por esto que el trabajo futuro puede ser desarrollado en diversas líneas.

Por un lado, la aplicación puede evolucionar y ampliar desde el punto de vista de profesor. Para

ello sería interesante incluir las siguientes funcionalidades: 1. Asociar una serie de estudiantes a un profesor modificando el modelo de datos de la aplicación. 2. Generar estadísticas globales de los estudiantes mediante el uso de los indicadores de aprendizaje, pueden servir al profesor para estudiar el desarrollo general de sus estudiantes. 3. Implementar un sistema de alertas que envíe un mensaje al profesor cuando detecte que muchos estudiantes están obtenido malos resultados para un objeto de aprendizaje concreto, de este modo se puede estudiar si existe algún problema con ese recurso de aprendizaje.

Desde un punto de vista del estudiante sería muy interesante desarrollar una interfaz gráfica que permita introducir *Topic Maps* en el sistema mediante un editor de grafos, pudiendo añadir nodos, dibujar arcos entre nodos y permitiendo el desplazamiento de estos objetos por la pantalla para ajustar su estructura. Se estudiaron librerías JavaScript con las que llevar a cabo esta implementación como son Vis.js [50] y Sigma.js [51], pero hubo que descartar este desarrollo por estar fuera del alcance del proyecto.

En relación a la parte lógica de evaluación de *Topic Maps*, se podrían desarrollar nuevos indicadores de aprendizaje que muestren más datos al profesor y al estudiante de los conocimientos adquiridos. Otro punto interesante sería mejorar la obtención de términos equivalentes haciendo uso de más de un servicio de equivalencia a la vez, pudiendo incrementar de este modo la similitud de los resultados obtenidos automáticamente al tener gran cantidad de conceptos de equivalencia provenientes de varias fuentes, en comparación con los resultados obtenidos manualmente que únicamente obtienen conceptos de equivalencia de una fuente, el criterio del profesor.

Por último, en la actualidad la aplicación depende de servicios web externos para la obtención de términos de equivalencia. La implementación de una librería local con listas de términos equivalentes actualizables por los profesores, a medida que introducen nuevos objetos de aprendizaje, supondría dejar de depender de servicios externos a la vez que mejoraría la calidad de los términos equivalentes proporcionados.

6.3. Opinión personal

La realización de este proyecto me ha generado una opinión personal muy positiva, tanto por el proceso de desarrollo llevado a cabo, así como por el resultado final obtenido. Considero que este proyecto me ha permitido aplicar los distintos conocimientos adquiridos a lo largo del grado y adquirir experiencia en la gestión e implementación de proyectos en el ámbito de la ingeniería informática.

Desde un punto de vista tecnológico y del desarrollo web, he conseguido aprender tecnologías novedosas que se encuentran en auge, como son NodeJS, AngularJS y MongoDB, aumentar mis conocimientos en tecnologías ya conocidas como es Java, hacer uso de metodologías ágiles para gestionar la metodología de trabajo y mejorar en la redacción de documentos formales haciendo uso del editor de textos Latex, herramienta profesional de elaboración de documentos. Del mismo modo, este trabajo me ha servido para mejorar mis capacidades de adaptación a nuevas tecnologías,

de búsqueda de información y para mejorar en la capacidad de administración de varias tecnologías y sistemas diferentes dentro en un mismo proyecto.

Desde un punto de vista de desarrollo personal, este proyecto me ha permitido centrarme en un área de la ingeniería informática de interés para mí, como lo es el desarrollo web, pudiéndola conocer en mayor profundidad y adquiriendo nuevos conocimientos más allá de los impartidos en el grado, así como descubriendo las dificultades que presenta enfrentarse a un proyecto de esta índole, lo cual considero muy importante y enriquecedor para mi futuro como ingeniero en informática.

En mi opinión, el trabajo de fin de grado es una parte muy importante en el desarrollo de todo ingeniero, y, en este caso, no lo ha sido menos; la experiencia en la interacción y el trabajo con los directores de proyecto ha sido muy buena, del mismo modo que estoy muy satisfecho con los conocimientos adquiridos a raíz de esta interacción y del propio trabajo realizado. Espero que la aplicación desarrollada pueda servir de utilidad dentro de la Universidad de Zaragoza y, si se da el caso, en otros centros educativos.

A. Lógica matemática de los indicadores de aprendizaje

Este anexo tiene como objetivo la definición y explicación de las fórmulas matemáticas aplicadas en la obtención de los indicadores de aprendizaje, que tiene lugar durante el proceso de evaluación semántica entre dos *Topic Maps*.

A.1. Similitud entre conceptos

Este indicador relaciona los nodos introducidos en un *Topic Map* TM con respecto a los que aparecen en el *Topic Map* de referencia TM_Ref. La fórmula aplicada en su obtención es la siguiente:

$$ConceptSimilarity = (\frac{\sharp ConceptosEq.TM}{\sharp ConceptosTotalesTM_REF}) \cdot cte_1 + (1 - \frac{\sum DesviacionPesos}{2 \cdot \sharp ConceptosEq.TM}) \cdot cte_2$$

Hace uso de las siguientes definiciones básicas:

- $\sharp ConceptosEq.TM$: representa el número de conceptos semánticamente equivalentes que existen entre los dos *Topic Maps*. Se dice que un nodo A perteneciente a TM y un nodo B perteneciente a TM_Ref presentan una equivalencia semántica cuando éstos están etiquetados por términos semánticamente equivalentes.
- $ConceptosTotalesTM_Ref$: representa el número de conceptos que posee el *Topic Map* de referencia.
- $\sum DesviacionPesos$: dados dos *Topic Maps* TM y TM_Ref, la desviación de pesos se corresponde con el sumatorio de la diferencia de pesos (valor numérico de 1 a 3 que etiqueta cada nodo) entre cada par de conceptos equivalentes.

Dadas estas definiciones, el primer término de la fórmula divide el número de conceptos equivalentes que muestran los dos *Topic Maps*, entre el número de conceptos totales del TM_Ref, siendo multiplicado el resultado por una constante $cte1$ con valor igual a 0.8. Esto implica que el número de conceptos equivalentes entre los dos *Topic Maps* posee un 80 % del peso del indicador.

El segundo término resta a uno la división del sumatorio de la desviación de pesos de los nodos entre dos veces el número de conceptos equivalentes. Todo ello va multiplicado por una constante $cte2$ con valor igual a 0.2. Esto implica que la desviación de peso posee un 20 % del peso del indicador.

Sumando ambos términos se obtiene el valor final del indicador, que estará comprendido entre 1 y 0, siendo mayor la similitud entre conceptos cuanto más próximo es este valor a 1.

A.2. Similitud en adyacencia

Este indicador establece una correspondencia entre las relaciones introducidas en un *Topic Map* TM con respecto a las que aparecen en el *Topic Map* de referencia TM_Ref, sin tener en cuenta el verbo que etiqueta los arcos de las relaciones. La fórmula aplicada en su obtención es la siguiente:

$$AdjacencySimilarity = \frac{\sharp RelacionesEq.Estructura}{\sharp RelacionesTotalesTM_REF}$$

Hace uso de las siguientes definiciones básicas:

- $\sharp RelacionesEq.Estructura$: representa el número de relaciones con equivalencia en estructura entre los dos *Topic Maps* TM y TM_Ref. Para explicar este concepto se supone que en TM (*Topic Map* a evaluar) existen dos nodos A y B unidos por un arco desde el nodo A hasta el nodo B y en TM_Ref (*Topic Map* de referencia) existen dos conceptos C y D unidos por un arco desde el nodo C hasta el nodo D. Si los nodos del TM A y B son equivalentes semánticamente a los nodos del TM_Ref C y D respectivamente, y los arcos de ambas relaciones están etiquetados por conceptos NO equivalentes semánticamente, entonces se dice que existe una relación en equivalencia en estructura entre las dos relaciones (arcos).
- $\sharp RelacionesTotalesTM_Ref$: representa el número de relaciones (arcos) que posee el *Topic Map* de referencia.

Dadas estas definiciones, la fórmula se compone de un único término en el que se realiza la división entre el número de relaciones en equivalencia en estructura existentes entre los dos *Topic Maps* y el número de relaciones que posee el *Topic Map* de referencia. Se genera de este modo un resultado numérico comprendido entre 0 y 1, siendo mayor la similitud en adyacencia cuanto más próximo es este valor a 1.

A.3. Similitud en relación

Este indicador establece una correspondencia entre las relaciones introducidas en un *Topic Map* TM con respecto a las que aparecen en el *Topic Map* de referencia TM_Ref, teniendo en cuenta el verbo que etiqueta los arcos de las relaciones. La fórmula aplicada en su obtención es la siguiente:

$$RelationSimilarity = \frac{\sharp RelacionesEq.Semantica}{\sharp RelacionesTotalesTM_Ref}$$

Hace uso de las siguientes definiciones básicas:

- $\sharp RelacionesEq.Semantica$: representa el número de relaciones con equivalencia semántica entre los dos *Topic Maps* TM y TM_Ref. Para explicar este concepto se supone que en TM (*Topic Map* a evaluar) existen dos nodos A y B unidos por un arco desde el nodo A hasta el nodo B y en TM_Ref (*Topic Map* de referencia) existen dos conceptos C y D unidos por un arco desde el nodo C hasta el nodo D. Si los nodos del TM A y B son equivalentes semánticamente a los nodos del TM_Ref C y D respectivamente, y los arcos de ambas relaciones están etiquetados por conceptos equivalentes semánticamente, entonces se dice que existe una relación en equivalencia semántica entre las dos relaciones (arcos).
- $\sharp RelacionesTotalesTM_Ref$: representa el número de relaciones (arcos) que posee el *Topic Map* de referencia.

Dadas estas definiciones, la fórmula se compone de un único término en el que se realiza la división entre el número de relaciones en equivalencia semántica existentes entre los dos *Topic Maps* y el número de relaciones que posee el *Topic Map* de referencia. Se genera de este modo un resultado numérico comprendido entre 0 y 1, siendo mayor la similitud en relación cuanto más próximo es este valor a 1.

A.4. Similitud de *Topic Map*

Este indicador establece una correspondencia general entre dos *Topic Maps*, uno que se desea evaluar TM y otro que es el de referencia TM_Ref. La fórmula aplicada en su obtención es la siguiente:

$TM\_Similarity = (Concep\_Sim. \cdot ct_3 + (Relation\_Sim \cdot ct_4 + Adjacency\_Sim \cdot ct_5) \cdot ct_7) \cdot (1 - Prop.Estructural \cdot ct_6)$

Hace uso de las siguientes definiciones básicas:

- $Concep\_Sim.$: representa el resultado obtenido de aplicar el indicador de aprendizaje "Similitud entre conceptos"*a* los *Topic Maps* TM y TM_Ref.
- $Relation\_Sim$: representa el resultado obtenido de aplicar el indicador de aprendizaje "Similitud en relación"*a* los *Topic Maps* TM y TM_Ref.
- $Adjacency\_Sim$: representa el resultado obtenido de aplicar el indicador de aprendizaje "Similitud en adyacencia"*a* los *Topic Maps* TM y TM_Ref.
- $Prop.Estructural$: representa la similitud en estructura (suma de conceptos y relaciones) entre dos *Topic Maps*. Dados dos *Topic Maps* TM y TM_Ref, se obtiene la proporción estructural de los mismos calculando la diferencia en valor absoluto entre la suma de los conceptos y relaciones del *Topic Map* TM y la suma de los conceptos y relaciones del *Topic Map* TM_Ref. A continuación este resultado se divide entre una constante denominada $Tam.GrafoDefecto$, con valor por defecto igual a 17. Cuanto más próximo es el resultado a 0, mayor es la similitud estructural entre los dos *Topic Maps*. A continuación se muestra la fórmula que se acaba de explicar, mediante la cual se obtiene la proporción estructural:
  $Prop.Estructural = \frac{ABS((ConceptosTM + RelacionesTM) - (ConceptosTM\_REF + RelacionesTM\_REF))}{Tam.GrafoDefecto}$

Dadas estas definiciones, en el primer término de la fórmula se calcula la suma del resultado obtenido a través del indicador de aprendizaje "Similitud en relación", multiplicado por una constante ct_4 con valor igual a 0.8 (lo cual le da un peso de importancia del 80 % a este indicador en la suma), con el resultado obtenido a través del indicador de aprendizaje "Similitud en adyacencia", multiplicado por una constante ct_5 con valor igual a 0.2 (20 % del peso de la suma). A continuación, se realiza la suma de este resultado, multiplicado por una constante ct_7 con valor igual a 0.3 (le da a este resultado un peso igual al 30 % en la suma) con el resultado obtenido a través del indicador de aprendizaje "Similitud entre conceptos", multiplicado por una constante ct_3 con valor igual a 0.7 (le da a este indicador un peso de importancia del 70 % en la suma).

El segundo término resta a uno el resultado obtenido de aplicar la fórmula de "Proporción estructural"*a* los dos *Topic Maps* TM y TM_Ref, multiplicado por una constante ct_6 con valor igual a 0.5 (reduce a la mitad el impacto de la proporción estructural en la fórmula).

Por último, multiplicando ambos términos se obtiene el valor final del indicador, que estará comprendido entre 1 y 0, siendo mayor la similitud entre *Topic Maps* cuanto más próximo es este valor a 1.

A.5. Conceptos no detectados

Este indicador muestra los conceptos que aparecen en el *Topic Map* de referencia pero no en el *Topic Map* introducido para ser analizado. No se requiere del uso de ninguna fórmula matemática para su obtención, únicamente se itera sobre los conceptos de las dos estructuras semánticas.

B. Manual de instalación del sistema

Este anexo tiene como finalidad explicar como realizar un despliegue del sistema, para ello, en primer lugar, se detalla el fichero de configuración del proyecto, y, a continuación, se procede a definir el proceso de despliegue del sistema.

B.1. Fichero de configuración

El fichero de configuración del sistema se encuentra en la raíz del proyecto con nombre"config.js". El contenido del mismo se puede observar en la figura 15. A continuación se explica cada uno de los componentes de este fichero, siendo todos ellos modificables por el usuario según sus preferencias a la hora de realizar el despliegue.

- **db**: se encarga de definir la URL en la que se encuentra la instancia de la base de datos MongoDB para producción, desarrollo y tests. Por defecto se utiliza la base de datos del modo de desarrollo que está situada en local, en la misma máquina en la que se lanza el servidor web.
- **port**: define el puerto en el que se va a ejecutar el API del sistema, por defecto lo hace en el puerto 3000.
- **portDictionary**: define el puerto en el que se va a ejecutar el API del servicio de equivalencia de términos desarrollado. Por defecto tiene definido el puerto 3100.
- **deploy**: efine los atributos de la cuenta de administrador de sistema que se creará a la hora de desplegar el sistema. Por defecto todos sus campos contienen "admin". Es importante no modificar el contenido del campo "nip", pues dejaría de ser una cuenta de administrador.
- **JwtSignature**: muestra la clave usada para firmar los JSON Web Token que se transmiten en la aplicación. Se puede modificar por el usuario.
- **WORDNIK_API**: define la clave que se utiliza para acceder al API del servicio externo de equivalencia de términos. El usuario puede acceder a este servicio web y obtener una nueva clave si lo desea.

```
module.exports = {
    /* Servidor base de datos MongoDB */
    db:
    {
        production: "mongodb://localhost/tutordb-prod",
        development: "mongodb://localhost/tutordb-dev",
        test: "mongodb://localhost/tutordb-test"
    },
    /* Puerto API sistema */
    port:{
        production: 3000,
        development: 3000,
        test: 3000
    },
    /* Puerto API servicio Facade equivalencia términos */
    portDictionary:{
        production: 3100,
        development: 3100,
        test: 3100
    },
    /* Cuenta de administrador creada en el despliegue */
    deploy:{
        admin:{
            nip: "admin",
            email: "admin@gmail.com",
            password: "admin",
            name: "Admin",
            lastname1: "Admin",
            lastname2:"Admin",
            address: "Admin"
        }
    },
    /* Firma de los JSON Web Token */
    jwtSignature: "jwtsignaturecomplex",

    /* Clave del API del servicio web de equivalencia de términos Wordnik */
    WORDNIK_API: "90fe73eb690d010ebd0080f662d0e4df6ae3b1bc98cf9e057"
};
```

Figura 15: Fichero de configuración del sistema

B.2. Despliegue del sistema

A continuación se especifica el proceso a seguir para llevar a cabo el despliegue del sistema en un servidor. Cabe destacar que este servidor tendrá que tener instalada una versión de NodeJS 5.0 o superior. Además, será necesario poseer una instancia de la base de datos MonoDB en ejecución, en su versión 3.2.4 o superior, bien en el servidor local o en remoto. Los pasos a seguir son los siguientes:

1. Acceder a la carpeta raíz del código fuente y ejecutar "npm install". Con ello se descargarán todas las dependencias de la aplicación, tanto del "Front end" como del "Back end".

2. Modificar el fichero de configuración nombrado anteriormente “config.js”. En él se deberá especificar la URL en la que se está ejecutando la instancia de MongoDB (línea “db.development”), los puertos en los que se quieren ejecutar las APIs (líneas “port.development” y “portDictionary.evelopment”) y, por último, el objeto “deploy.admin” si se desea modificar algún atributo de la cuenta de administrador del sistema que se creará.

3. Ejecutar el comando “node deploy.js” en el mismo directorio con el fin de inicializar la base de datos. En este punto se creará el usuario administrador del sistema especificado anteriormente.

4. En la misma carpeta, ejecutar “node app.js”. Esto inicia el servidor.

5. Ya se puede acceder a la aplicación a través del puerto definido en el fichero de configuración, por defecto 3000, introduciendo la cuenta de administrador en la pantalla de “login”. El API del servicio web de equivalencia de términos también estará disponible por defecto en el puerto 3100.

En las futuras ejecuciones del sistema únicamente será necesario ejecutar el comando número cuatro: “node app.js”.

C. Estructura de directorios y ficheros del proyecto

Este anexo tiene como finalidad explicar las estructuras de directorios y ficheros que componen el sistema desarrollado. Se puede distinguir la estructura del servidor web, formada por un "Back end" desarrollado en NodeJS y un "Front end" desarrollado en AngularJS, y, por otro lado, la estructura de un proyecto JAVA, encargado de llevar a cabo la evaluación semántica de un Topic Map.

C.1. Estructura general del servidor web

En la figura 16 se puede observar la estructura del proyecto correspondiente al servidor Web. En este proyecto se encuentra tanto el "Back end" como el "Front end" de la aplicación, correspondiendo al "Front end" la carpeta llamada "frontend" y al "Back end" el resto de ficheros y directorios. El archivo principal del proyecto es "app.js", que, al ser ejecutado con "node app.js", se encarga de lanzar la apliación NodeJS y de servir el contenido estático del "Front end" en la carpeta "frontend/app".

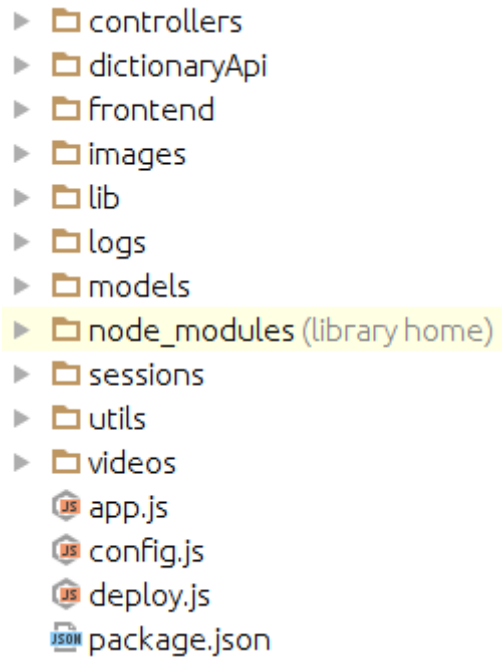


Figura 16: Estructura general del proyecto que implementa el servidor Web

C.1.1. Estructura del Back end

La estructura del "Back end" ha sido definida de manera que quedan agrupado los distintos componentes en carpetas, facilitando de este modo la ampliación del proyecto. En la figura 17 se puede observar esta distribución de componentes en directorios.

A continuación de explica la funcionalidad de cada directorio:

- **Controllers**: este directorio contiene un fichero por cada endpoint definido en la API del sistema. Cada uno de estos ficheros implementa las distintas operaciones del API y la funcionalidad explicada en el anexo D. El fichero "index.js" no define un endpoint, sino que es el encargado de asociar cada enpoint a una ruta del API.
- **DictionaryApi**: este directorio tiene la misma funcionalidad que el directorio "controllers" pero para el API del servicio web de equivalencia de términos. Se observa como únicamente posee un fichero "dictionary.js" que se corresponde con el único endpoint de este API.
- **Images**: directorio en el que se almacenan las imágenes que se adjuntan en cada Topic Map introducido por un estudiante en el sistema.
- **Lib**: este directorio se encarga de almacenar los ficheros "jar" que definen el proyecto JAVA que lleva a cabo la evaluación semántica de un Topic Map. Estos ficheros serán instanciados desde el fichero "learningIndicator.js", situado en la carpeta "controllers", cuando un estudiante solicite una evaluación de un Topic Map introducido en el sistema.
- **Logs**: directorio encargado de almacenar los ficheros de log generados durante la ejecución del sistema. Se distingue entre los logs de red, y los logs del sistema.
- **Models**: en este directorio existe un fichero por cada modelo de Mongoose definido en el sitema. Es decir, define la estructura de las colecciones que se almacenan en la base de datos. El fichero "index.js" es el encargado de exportar las declaraciones de todos los modelos.
- **Node_modules**: contiene todas las librerías necesarias para el buen funcionamiento del proyecto. Se genera durante el despliegue del sistema.

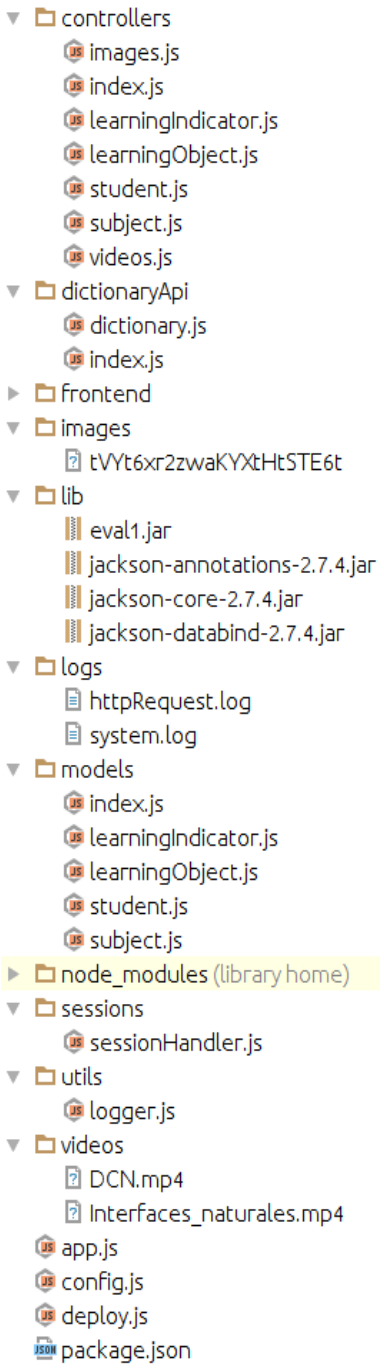


Figura 17: Estructura del Back end.

- **Sessions**: este directorio contiene un único fichero que es el encargado de definir el Middleware que gestiona la autenticación en el sistema mediante el uso de JSON Wen Token.
- **Utils**: define la configuración de servicios de utilidad en el sistema. En concreto el fichero "logger.js" define la configuración de los loggers del sistema.
- **Videos**: directorio en el que se almacenan los vídeos que posee cada objeto de aprendizaje introducido por un profesor en el sistema.
- **Directorio raíz**: Contiene el fichero de configuración "config.js", de inicialización de la base de datos MongoDB "deploy.js", un fichero "package.json" encargado de declarar las dependencias del proyecto, y, por último, el fichero "app.js" mediante el cual se lanza el servidor.

C.1.2. Estructura del Front end

Del mismo modo que ocurre en el "Back end", la estructura del "Front end" (figura 18) también ha sido definida agrupando componentes, ficheros pertenecientes a cada pantalla del sistema, con el fin de darle una mayor organización y posibilidad de extensión.

A continuación se detalla la estructura del "Front end":

- Dentro del directorio "app" puede observarse gran cantidad de subdirectorios, cada uno de ellos define una pantalla de la interfaz del sistema, a excepción de las carpetas "common" y "services". Todos estos subdirectorios contienen una misma estructura, basada en el patrón de diseño modelo vista controlador de AngularJS. Tomando como ejemplo el directorio "createTopicMap", el cual define la pantalla mediante la cual el estudiante es capaz de almacenar un Topic Map en el sistema, se puede observar como existen tres ficheros en su interior, uno con extensión ".js", el cual define el controlador de la pantalla, otro con extensión ".html", el cual define la vista, y, por último, un fichero con extensión ".css", el cual define el estilo de la pantalla.

- El directorio "common" tiene como finalidad almacenar archivos comunes a muchas pantallas, en este caso almacena las imágenes que aparecen en distinta partes de la interfaz de la aplicación como puede ser el "Background".

- El directorio "services" almacena el servicio de sesión del "Front end", encargado de manejar los JSON Web Token que definen la sesión de los usuarios.

- En la raíz del directorio "app" se pueden distinguir tres ficheros: "favicon.ico", corresponde a la pequeña imagen que se muestra en la pestaña del navegador web, "index.html", encargado de definir los scripts y hojas de estilo que se han de utilizar en la aplicación , y, por último, "app.js", fichero principal de AngularJS encargado de asociar rutas de la aplicación a cada vista y de interceptar posibles errores en las llamadas a las rutas.

- La carpeta "node_modules", al igual que ocurre en el "Back end", contiene todas las librearías necesarias para el funcionamiento del "Front end" y es generada en el despliegeu del sistema.

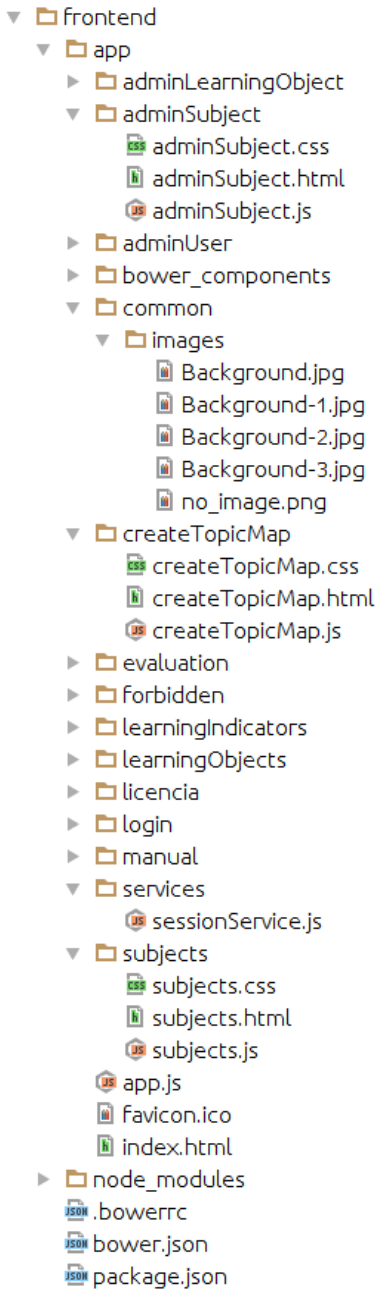


Figura 18: Estructura del Front end.

- Por último los ficheros “bower.json” y “package.json” definen las dependencias utilizadas por el “Front end”.

C.2. Estructura del proyecto JAVA

El proyecto JAVA es el encargado de llevar a cabo el análisis semántico de Topic Map en el sistema. Este proyecto se comprime en archivos “.jar”, que se añaden a la estructura “Back end”, para después ser instanciados y ejecutados por el controlador “learningIndicator.js” cuando un estudiante solicita el análisis de un Topic Map. En la figura 19 se puede observar la estructura que presenta este proyecto, a continuación se define la funcionalidad de cada una de las clases contenidas en el mismo.

- **Evaluation**: es la clase que se instancia desde el “Back end” del servidor. Se encarga de parsear los Topic Maps recibidos desde el “Back end” en formato cadena de texto JSON a objetos Java “LabelledTopicMap”. A continuación lleva a cabo la evaluación del Topic Map estudiante haciendo uso del resto de clases, y, por último devuelve los indicadores de aprendizaje al “Back end”.
- **LabelledTopicMap**: define un objeto en el que almacenar la estructura de un Topic Map. Hace uso de las clases “Node” y “Relation” para almacenar las listas de los nodos y las relaciones del Topic Map.
- **Node**: define un objeto que permite almacenar la estrcutura de un nodo de un Topic Map.
- **Relation**: define un objeto que permite almacenar la estructura de una relación de un Topic Map.
- **Main**: clase definida para la realización de pruebas desde el proyecto JAVA, simula la interacción con el “Back end” del servidor.
- **Matrix**: define un objeto matriz de adyacencia que será utilizado en los en los cálculos de análisis del Topic Map.
- **SimilarityMethod**: define una clase abstracta que representa una abstracción de un método de similaritud. Define los indicadores de aprendizaje que ha de proporcionar este método de similaritud, así como operaciones básicas y atributos que ha de tener, pero queda abierta la posibilidad de realizar distintos tipos de implementaciones.
- **SimilarityImplementation**: extendiendo la clase “SimilarityMethod”, implementa el cálculo de los indicadores de aprendizaje “similaridad ente conceptos”, “similaridad en adyacencia”, “similaridad en relacion” y “similaridad de Topic Map”.
- **NoEquivalentConceptsMethod**: define una clase abstracta que representa una abstracción de un método de cálculo de conceptos no equivalentes entre dos Topic Maps. Define los

indicadores de aprendizaje que ha de proporcionar este método, así como operaciones básicas y atributos que ha de tener, pero queda abierta la posibilidad de realizar distintos tipos de implementaciones.

- **NoEquivalentConceptsImplementation**: extendiendo la clase "NoEquivalentConcepts-Method", implementa el cálculo del indicador de aprendizaje "conceptos no detectados".

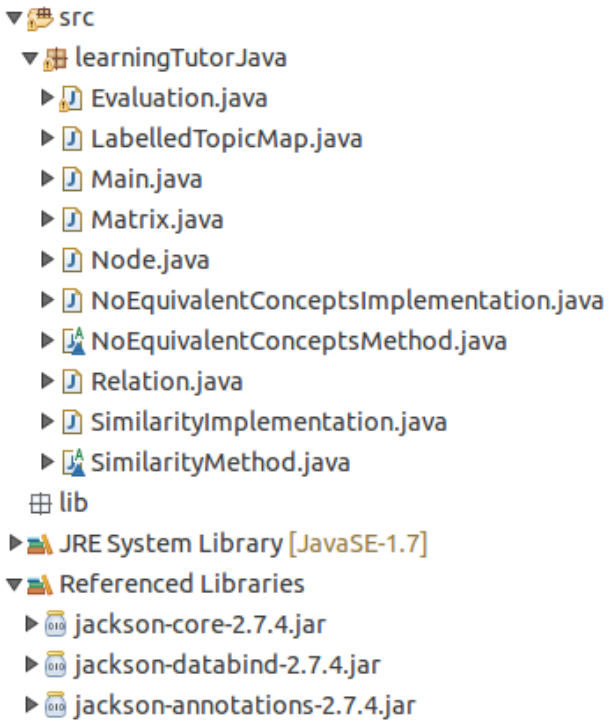


Figura 19: Estructura del proyecto Java

D. Especificación de las APIs

Este anexo tiene como finalidad la especificación del API RESTful definida en la aplicación. Para ello se detalla en un primer lugar la estructura de los mensajes de respuesta a peticiones, a continuación se explica cómo realizar una autenticación de usuario contra la misma y, por último, se enumera detalladamente cada uno de los endpoints que ofrece junto a sus operaciones.

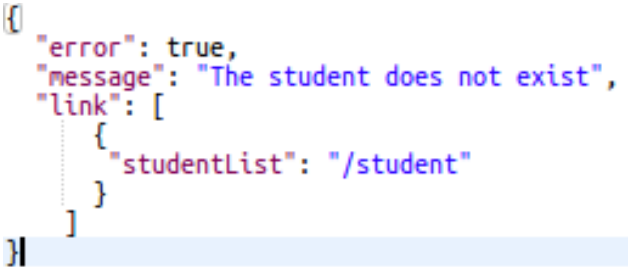
D.1. Estructura de mensajes

Mientras que los mensajes de llamada al API son específicos de cada endpoint, pudiendo tener estos payload o siendo únicamente una invocación a una ruta concreta, las respuestas que ofrece el API siguen siempre una misma estructura. Esta estructura se corresponde con un objeto JSON formado por tres campos: "error", "message" y "link".

- **"Error"**: Este campo indica si la operación solicitada ha sido completada con éxito, siendo su valor igual a "true" en caso afirmativo e igual a "false" en caso contrario.
- **"Message"**: Campo que contiene el payload de la respuesta, es decir, los datos. El contenido de este campo varía en función del endpoint llamado y el éxito con el que se haya completado la operación. En caso de error en la operación, este campo siempre contiene una cadena de error. En caso de éxito, este campo puede contener un objeto JSON o un Array dependiendo del endpoint.
- **"Link"**: Campo encargado de devolver links para la navegación en la aplicación. Estructuralmente"Student successfully deleted"se corresponde con un Array, en el que cada elemento contiene una clave y un valor. La clave describe el enlace de navegación devuelto mientras que el valor es el propio enlace.

En la figura 20 puede observarse un ejemplo de una respuesta devuelta por el API. A la izquierda puede verse una respuesta sin error, mientras que en la derecha de la misma se observa el mensaje devuelto en caso de no tener éxito la operación.

(a) Respuesta sin error



(b) Respuesta con error

Figura 20: Ejemplo de respuesta sin error y con error proporcionada por el API

D.2. Gestión de autenticación

Como se muestra a continuación en el apartado D.3, ciertas operaciones del API requieren de una autenticación de usuario antes de su invocación por motivos de seguridad. Los tipos de usuarios existentes en la aplicación se corresponden con cuentas de estudiantes o de administrador, utilizada por profesores, pudiendo acceder de este modo los profesores a ciertas operaciones exclusivas como puede ser el borrado de cuentas.

Esta autenticación se lleva a cabo mediante el uso de JSON Web Token, unas cadenas que guardan información sobre el usuario. De este modo, si se desea acceder a una operación protegida, es necesario en primer lugar obtener el token, y, a continuación, incluirlo en cada petición realizada.

La obtención del token se hace realizando una llamada al recurso "login" del endpoint Estudiantes, introduciendo como cuerpo del mensaje las credenciales (nip, password). Estas credenciales se verifican en el "Back end" del sistema, y, si son correctas, se devuelve al usuario un JSON Web Token con información que lo identifica firmada y un tiempo de expiración del mismo (30 minutos).

Una vez se posee este token, para hacer uso de él, hay que añadirlo en cada petición. Esto se hace añadiendo en cada petición el Header HTTP "Authorization" con contenido "Bearer tokenDevuelto", donde "tokenDevuelto" se corresponde con el token proporcionado en la llamada al recurso "login". Una vez se agote el tiempo de expiración habrá que repetir el proceso para obtener un nuevo token.

D.3. Endpoint de gestión de usuarios

Este endpoint se encarga de gestionar los usuarios, tanto los estudiantes como la cuenta de administrador. La tabla 5 muestra un listado con todas las operaciones disponibles, las cuales permiten obtener, crear, modificar y eliminar cuentas de usuario. Esta tabla se apoya en las figuras 21, 22 y 23, en lsa que se muestran los distintos objetos mencionados en la tabla. Además, en este endpoint, se incluye el recurso "login", mediante el cual un usuario puede identificarse en el sistema.. Se puede observar cómo la ruta del endpoint es definida con el nombre "student", esto es debido a que el sistema consta principalmente de cuenta de estudiantes. Un caso especial es la cuenta de administrador, que se identifica por tener un nip igual a "admin". A continuación se describe cada una de las operaciones definidas en la tabla.

- La operación POST /student se encarga de añadir un nuevo usuario al sistema, a partir de un objeto de inserción de usuario 21. Devuelve un objeto de información completa del usuario 22 en caso de éxito.
- La operación GET /student es la encargada de obtener un listado de todos los usuarios del sistema. No es necesario incluir ningún tipo de dato en la llamada.
- La operación GET /student/:id tiene como objetivo obtener la información de un usuario en concreto. Se introduce en la URL el identificador de ese usuario y no es necesaria la inserción de ningún tipo de dato más en la llamada.
- La operación PUT /student/:id es utilizada para actualizar la información de un usuario concreto especificado en la URL. Como "Input" se ha de introducir un objeto de inserción de usuario 21 (sin nip), del que se pueden eliminar los campos que no se desean modificar. Devuelve un objeto de información completa del usuario 22 en caso de éxito.
- La operación DELETE /student/:id se encarga de eliminar un usuario del sistema. Este usuario se especifica en la URL y no es necesaria la inserción de ningún tipo de dato más en la llamada. Devuelve un mensaje de éxito en caso de producirse el borrado.
- La operación POST /student/login es la encargada de autenticar un usuario en el sistema. Se ha de introducir un objeto credenciales 23 en la llamada. Devuelve un JSON Web Token en caso de éxito.

Método	Ruta	Input	Output	Autentic.	Link
POST	/student	Objeto de inserción de usuario	Objeto de información completa del usuario	Admin	Lista usuarios
GET	/student	-	Lista de objetos de información completa de todos los usuarios del sistema	Admin, Estudiante	Link a cada usuario
GET	/student/:id	-	Objeto de información completa del usuario	Admin, Estudiante	Lista usuarios
PUT	/student/:id	Objeto de inserción de usuario sin nip	Objeto de información completa del usuario	Admin	Lista usuarios
DELETE	/student/:id	-	Mensaje: "Student successfully deleted"	Admin	Lista usuarios
POST	/student/login	Objeto de credenciales	JSON Web Token	No	-

Tabla 5: Operaciones del endpoint de gestión de usuarios


```
{
"nip":"565656",
"email":"565656@unizar.es",
"password":"565656",
"name":"Juan",
"lastname1":"Mateo",
"lastname2":"Rodriguez",
"address":"Avenida de Madrid, 120, 2A, Zaragoza",
"subject":["Matematicas","Fisica","Interfaces"]
}
```


Figura 21: Objeto de inserción usuario


```
{
  "error": false,
  "message": {
    "nip": "565656",
    "email": "565656@unizar.es",
    "name": "Juan",
    "lastname1": "Mateo",
    "lastname2": "Rodriguez",
    "address": "Avenida de Madrid, 120, 2A, Zaragoza",
    "subject": [
      "Matematicas",
      "Fisica",
      "Interfaces"
    ],
    "href": "/student/565656"
  },
  "link": [
    {
      "studentList": "/student"
    }
  ]
}
```


Figura 22: Objeto de información completa usuario


```
{"nip":"565656",
"password":"565656"}
```


Figura 23: Objeto de inserción de credenciales

D.4. Endpoint de objetos de aprendizaje

Endpoint encargado de gestionar los objetos de aprendizaje de la aplicación. Mediante el mismo se permite la obtención, creación, actualización y eliminación de objetos de aprendizaje, como muestra la tabla 6. La figuras 24 y 25 sirven de apoyo a la tabla 6, mostrando los objetos nombrados en esta última. Una descripción de cada una de las operaciones de la tabla se muestra a continuación.

- La operación POST /learningObject se encarga de añadir un nuevo objeto de aprendizaje al sistema, a partir de un objeto de inserción de objeto de aprendizaje 24. Devuelve un objeto de información completa de objeto de aprendizaje 25 en caso de éxito.
- La operación GET /learningObject es la encargada de obtener un listado de todos los objetos de aprendizaje del sistema. No es necesario incluir ningún tipo de dato en la llamada. Se devuelve únicamente el identificador y el título para cada uno de ellos.
- La operación GET /learningObject/:id tiene como objetivo obtener la información de un objeto de aprendizaje en concreto. Se introduce en la URL el identificador de ese objeto de aprendizaje y no es necesaria la inserción de ningún tipo de dato más en la llamada.
- La operación PUT /learningObject/:id es utilizada para actualizar la información de un objeto de aprendizaje concreto especificado en la URL. Como "Input" se ha de introducir un objeto de inserción de objeto de aprendizaje 24 (sin el identificador), del que se pueden eliminar los campos que no se desean modificar. Devuelve un objeto de información completa de objeto de aprendizaje 25 en caso de éxito.
- La operación DELETE /learningObject/:id se encarga de eliminar un objeto de aprendizaje del sistema. Este objeto de aprendizaje se especifica en la URL y no es necesaria la inserción de ningún tipo de dato más en la llamada. Devuelve un mensaje de éxito en caso de producirse el borrado.
- La operación POST /learningObject/search es la encargada de buscar objetos de aprendizaje en el sistema a través de palabras clave. Se ha de introducir una lista de palabras clave en la llamada. Devuelve un listado de todos los objetos de aprendizaje que contienen esas palabras claves.

Método	Ruta	Input	Output	Autentic.	Link
POST	/learningObject	Objeto de inserción de objeto de aprendizaje	Objeto de información completa de objeto de aprendizaje	Admin	Lista objetos aprendizaje
GET	/learningObject	-	Lista de objetos de aprendizaje del sistema. Se incluye el identificador y el título de cada uno en la lista	Admin	Link a cada objeto aprendizaje
GET	/learningObject/:id	-	Objeto de información completa de objeto de aprendizaje	Admin Estudiante	Lista objetos aprendizaje
PUT	/learningObject/:id	Objeto de inserción de objeto de aprendizaje sin identificador.	Objeto de información completa de objeto de aprendizaje.	Admin	Lista objetos aprendizaje
DELETE	/learningObject/:id	-	Mensaje: "LearningObject successfully deleted"	Admin	Lista objetos aprendizaje
POST	/learningObject/search	Lista de palabras clave	Lista de objetos de aprendizaje que contienen las palabras clave en el sistema. Se incluye el identificador y el título de cada uno en la lista	Admin Estudiante	Link a cada objeto aprendizaje

Tabla 6: Operaciones del endpoint objetos de aprendizaje

```
{       "identifier" : "LO_10",
        "title" : "Natural Interfaces",
        "description" : "This learning object explains the concept of natural interfaces",
        "language" : "Spanish",
        "keywords" : ["interfaces", "gestures", "voice", "applications"],
        "aggregationlevel" : "1",
        "version" : "1",
        "status" : "final",
        "contribute" : [613429 , 612360 , 628279],
        "format" : "mp4",
        "size" : "65.9 MB",
        "location" : "videos/Interfaces_naturales.mp4",
        "requirement": ["browser", "any"],
        "duration" : "06:15",
        "learningsourcetype" : "Narrative text",
        "context" : "University second cycle",
        "dificulty" : "Medium",
        "copyrightandotherrestrictions" :   "yes",
            "annotatorCreator": "Topic map creador",
            "dateCreator": "2013-12-25",
            "textCreator": "none",
        "structTMCreator": {
                        "numberNodes": 1,
                        "numberRelations": 1,
                        "nodes": [{
                            "nodeId": 0,
                            "name": "Input",
                            "nodeValue": 1}],
                        "relations": [{
                            "relationId":0,
                            "relationFrom": 0,
                            "relationTo": 0,
                            "name": "cause"}]
                    },
                "annotatorTeacher": "Profesor",
                "dateTeacher": "2013-04-26",
                "textTeacher": "none",
        "structTMTeacher": {
                        "numberNodes": 1,
                        "numberRelations": 1,
                        "nodes": [{
                            "nodeId": 0,
                            "name": "Interface",
                            "nodeValue": 1}],
                        "relations": [{
                            "relationId":0,
                            "relationFrom": 0,
                            "relationTo": 0,
                            "name": "use"}]
}               }
```


Figura 24: Objeto de inserción de objeto de aprendizaje

Figura 25: Objeto de información completa de objeto de aprendizaje devuelto por el API

D.5. Endpoint de asignaturas

Este endpoint está destinado a la administración y gestión de las asignaturas del sistema. Al igual que los endpoints anteriores implementa las operaciones de obtención, creación, actualización y eliminación de asignaturas. Estas operaciones se pueden observar en la tabla 7, la cual hace uso de la figuras 26 y 27 como apoyo para definir los objetos mencionados en la misma. A continuación se muestra una descripción de cada una de las operaciones definidas en la tabla.

- La operación POST /subject se encarga de añadir una nueva asignatura al sistema, a partir de un objeto de inserción de asignatura 26. Devuelve un objeto de información completa de asignatura 27 en caso de éxito.
- La operación GET /subject es la encargada de obtener un listado de todas las asignaturas del sistema. No es necesario incluir ningún tipo de dato en la llamada. Se devuelve únicamente el código identificador, título y curso para cada una de ellas.
- La operación GET /subject/:id tiene como objetivo obtener la información de una asignatura en concreto. Se introduce en la URL el identificador de esa asignatura y no es necesaria la inserción de ningún tipo de dato más en la llamada.
- La operación PUT /subject/:id es utilizada para actualizar la información de una asignatura concreta especificada en la URL. Como "Input" se ha de introducir un objeto de inserción de asignatura 26 (sin el código identificador), del que se pueden eliminar los campos que no se desean modificar. Devuelve un objeto de información completa de asignatura 27 en caso de éxito.
- La operación DELETE /subject/:id se encarga de eliminar una asignatura del sistema. Esta asignatura se especifica en la URL y no es necesaria la inserción de ningún tipo de dato más en la llamada. Devuelve un mensaje de éxito en caso de producirse el borrado.
- La operación POST /subject/search es la encargada de buscar asignaturas en el sistema a través de un nip de un usaurio. Se ha de introducir el nip de un usuario válido en la llamada. Devuelve un listado de todas las asignaturas en las que se está "matriculado" este usuario. claves.

Método	Ruta	Input	Output	Autentic.	Link
POST	/subject	Objeto de inserción de asignaturas	Objeto de información completa de asignatura	Admin	Lista asignaturas
GET	/subject	-	Lista de asignaturas del sistema. Se incluye el código identificador, título y curso para cada una en la lista	Admin	Link a cada asignatura
GET	/subject/:id	-	Objeto de información completa de asignatura	Admin Estudiante	Lista asignaturas
PUT	/subject/:id	Objeto de inserción de asignaturas sin código identificador.	Objeto de información completa de asignatura	Admin	Lista asignaturas
DELETE	/subject/:id	-	Mensaje: "Subject successfully deleted"	Admin	Lista asignaturas
POST	/subject/search	Nip de usuario	Lista de objetos de información completa de asignaturas en las que está "matriculado" un usuario.	Admin Estudiante	Link a cada asignatura

Tabla 7: Operaciones del endpoint asignaturas

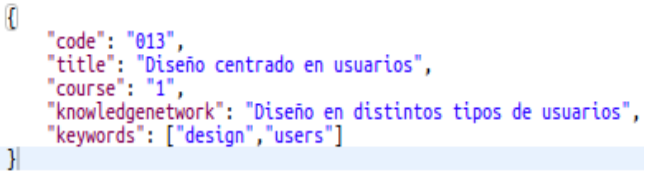


Figura 26: Objeto inserción de asignatura

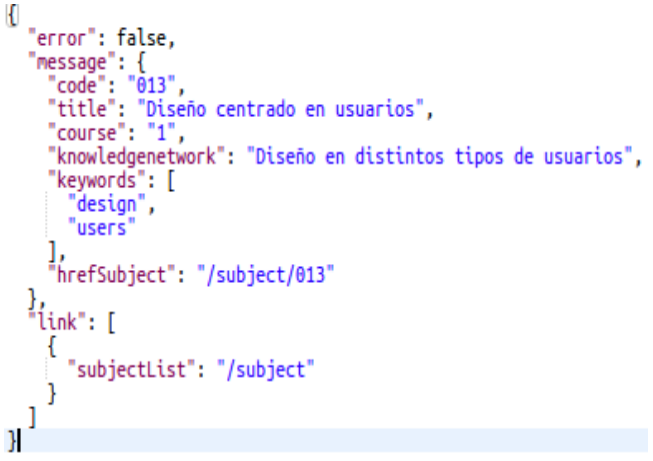


Figura 27: Objeto información completa de asignatura devuelto por el API

D.6. Endpoint de indicadores de aprendizaje

Este endpoint es el encargado de la gestión de estructuras "indicador de aprendizaje". Estas estructuras almacenan un Topic Map introducido por un estudiante en el sistema junto a la lista de indicadores de aprendizaje generados tras su evaluación. Debido a esto, en el momento de su creación mediante el método "POST", la estructura únicamente contendrá el Topic Map estudiante, sin indicadores de aprendizaje, y, una vez que se invoque el método "PUT", se desencadenará la evaluación semántica del Topic Map generando los indicadores de aprendizaje asociados al mismo. La tabla 8 especifica el comportamiento de las operaciones de este endpoint. Junto a la tabla 8 se adjuntan las figuras 28 y 29, la cuales sirven de apoyo para definir los objetos utilizados en la primera. A continuación se muestra la descripción de cada una de las operaciones definidas en la tabla.

- La operación POST /learningIndicator se encarga de crear una nueva estructura indicador de aprendizaje en el sistema, a partir de un objeto de inserción de estructura indicador de aprendizaje 28. Es el método mediante el cual un estudiante almacena un Topic Map en el sistema. Devuelve un objeto de información completa de estructura indicador de aprendizaje 29 en caso de éxito.
- La operación GET /learningIndicator se encarga de obtener un listado de todas las estructuras indicador de aprendizaje del sistema. No es necesario incluir ningún tipo de dato en la llamada. Para cada estructura devuelta se excluye el Topic Map por ocupar demasiado espacio.
- La operación GET /learningIndicator/:id tiene como objetivo obtener la información de una estructura indicador de aprendizaje en concreto. Se introduce en la URL el identificador de esa estructura y no es necesaria la inserción de ningún tipo de dato más en la llamada.Devuelve un objeto de información completa de estructura indicador de aprendizaje 29 en caso de éxito.
- La operación GET /learningIndicator/:id/TM tiene como objetivo obtener un archivo de texto que contenga la información del Topic Map de una estructura indicador de aprendizaje en concreto. Se introduce en la URL el identificador de esa estructura indicador de aprendizaje y no es necesaria la inserción de ningún tipo de dato más en la llamada.
- La operación PUT /learningIndicator/:id es utilizada con el fin de calcular los indicadores e aprendizaje de un Topic Map almacenado previamente en el sistema en una estructura indicador de aprendizaje especificada en la URL. No es necesaria la inserción de ningún tipo de dato más en la llamada. Realiza el cálculo de los indicadores de aprendizaje para ese Topic Map, y, a continuación, devuelve un objeto de información completa de estructura indicador de aprendizaje 29 en caso de éxito.
- La operación DELETE /learningIndicator/:id se encarga de eliminar una estructura indicador de aprendizaje del sistema. Esta estrcutura se especifica en la URL y no es necesaria la

inserción de ningún tipo de dato más en la llamada. Devuelve un mensaje de éxito en caso de producirse el borrado.

- La operación POST /learningIndicator/search es la encargada de buscar estructuras indicador de aprendizaje en el sistema mediante la introducción en la llamada de un identificador de objeto de aprendizaje o de un nip de usuario válidos. Devuelve un listado de todas las estructuras indicador de aprendizaje que posee un estudiante, o que están relacionadas con un objeto de aprendizaje.


```
{
    "nip" : "569181",
    "identifier" : "LO_1",
    "annotation" : "{{2,1}};{(0,\"application\",2),(1\"interaction\",3)};{(0,0,1,\"imply\")};}",
    "LO_name" : "Children oriented design",
    "structAnnotation": {
                    "numberNodes": 2,
                    "numberRelations": 1,
                    "nodes": [{
                        "nodeId": 0,
                        "name": "application",
                        "nodeValue": 2},
                        {
                        "nodeId": 1,
                        "name": "interaction",
                        "nodeValue": 3}
                        ],
                    "relations": [{
                        "relationId":0,
                        "relationFrom": 0,
                        "relationTo": 1,
                        "name": "imply"}
                        ]
    }
}
```


Figura 28: Objeto de inserción de estructura indicador de aprendizaje

```
{
  "error": false,
  "message": {
    "_id": "57c4626f066bf0f61eb47ec5",
    "nip": "569181",
    "LO_name": "Children oriented design",
    "identifier": "LO_1",
    "annotation": "{{2,1};{(0,\"application\",2),
    (1,\"interaction\",3)};{(0,0,1,\"imply\")};}",
    "imagePath": "",
    "indicators": [
      {
        "method": "Similarity Method",
        "showColour": true,
        "_id": "57c46271066bf0f61eb47ed0",
        "methodArray": [
          {
            "indicatorName": "Concept Similarity",
            "indicatorValue": "0.47",
            "_id": "57c46271066bf0f61eb47ed4"
          },
          {
            "indicatorName": "Relation Similarity",
            "indicatorValue": "0.13",
            "_id": "57c46271066bf0f61eb47ed3"
          },
          {
            "indicatorName": "Adjacency Similarity",
            "indicatorValue": "0.13",
            "_id": "57c46271066bf0f61eb47ed2"
          },
          {
            "indicatorName": "TM Similarity",
            "indicatorValue": "0.25",
            "_id": "57c46271066bf0f61eb47ed1"
          }
        ]
      },
      {
        "method": "Not identified concepts",
        "showColour": false,
        "_id": "57c46271066bf0f61eb47ecb",
        "methodArray": [
          {
            "indicatorName": "Node missed",
            "indicatorValue": "usability",
            "_id": "57c46271066bf0f61eb47ecf"
          },
          {
            "indicatorName": "Node missed",
            "indicatorValue": "children",
            "_id": "57c46271066bf0f61eb47ece"
          },
          {
            "indicatorName": "Node missed",
            "indicatorValue": "Investigation",
            "_id": "57c46271066bf0f61eb47ecd"
          },
          {
            "indicatorName": "Node missed",
            "indicatorValue": "cognitive skills",
            "_id": "57c46271066bf0f61eb47ecc"
          }
        ]
      }
    ],
    "structAnnotation": {
      "numberNodes": 2,
      "numberRelations": 1,
      "relations": [
        {
          "relationId": 0,
          "relationFrom": 0,
          "relationTo": 1,
          "name": "imply",
          "_id": "57c4626f066bf0f61eb47ec8"
        }
      ],
      "nodes": [
        {
          "nodeId": 0,
          "name": "application",
          "nodeValue": 2,
          "_id": "57c4626f066bf0f61eb47eca"
        },
        {
          "nodeId": 1,
          "name": "interaction",
          "nodeValue": 3,
          "_id": "57c4626f066bf0f61eb47ec9"
        }
      ]
    }
  },
  "link": [
    {
      "learningIndicatorList": "/learningIndicator"
    }
  ]
}
```


Figura 29: Objeto de información completa de estructura indicador de aprendizaje

Método	Ruta	Input	Output	Autentic.	Link
POST	/learningIndicator	Objeto de inserción de estructura indicador de aprendizaje	Objeto de información completa de estructura indicador de aprendizaje	Admin Estudiante	Lista de estructuras indicador aprendizaje
GET	/learningIndicator	-	Lista de estructuras indicador de aprendizaje del sistema. Se excluye el Topic Map de cada estructura en esta lista.	Admin	Link a cada estructura indicador aprendizaje
GET	/learningIndicator/:id	-	Objeto de información completa de estructura indicador de aprendizaje	Admin Estudiante	Lista de estructuras indicador aprendizaje
GET	/learningIndicator/:id/TM	-	Archivo de texto que contiene el Topic Map de una estructura indicador de aprendizaje	Admin Estudiante	-
PUT	/learningIndicator/:id	-	Objeto de información completa de estructura indicador de aprendizaje tras el cálculo de los indicadores de aprendizaje	Admin Estudiante	Lista de estructuras indicador aprendizaje
DELETE	/learningIndicator/:id	-	Mensaje: "LearningIndicator successfully deleted"	Admin Estudiante	Lista de estructuras indicador aprendizaje
POST	/learningIndicator/search	Nip de usuario ó Identificador objeto aprendizaje	Lista de objetos de información completa de estructuras indicador de aprendizaje correspondientes a un usuario o a un objeto de aprendizaje.	Admin Estudiante	Link a cada estructura indicador aprendizaje

Tabla 8: Operaciones del endpoint indicadores de aprendizaje

E. Manual de usuario

Este capítulo tiene como objetivo definir un manual de uso del sistema tanto para un usuario de tipo estudiante, como para el usuario administrador (profesor). En primer lugar se detallan al estudiante una serie de pasos a seguir para almacenar y evaluar en el sistema un Topic Map. A continuación se explica a un administrador del sistema cómo gestionar los usuarios, asignaturas y objetos de aprendizaje del mismo.

E.1. Manual de uso para estudiantes

Esta primera sección está destinada a explicar a los estudiantes que vayan a hacer uso del sistema cómo interactuar con el mismo. Se define como objetivo conseguir almacenar y evaluar un Topic Map en el sistema. Para ello hay que seguir los siguientes pasos:

1. A través de la pantalla de inicio de sesión, figura 30, se han de introducir las credenciales proporcionadas por el profesor. Al pulsar sobre el botón "Log in" se accederá al sistema si estas credenciales son correctas.

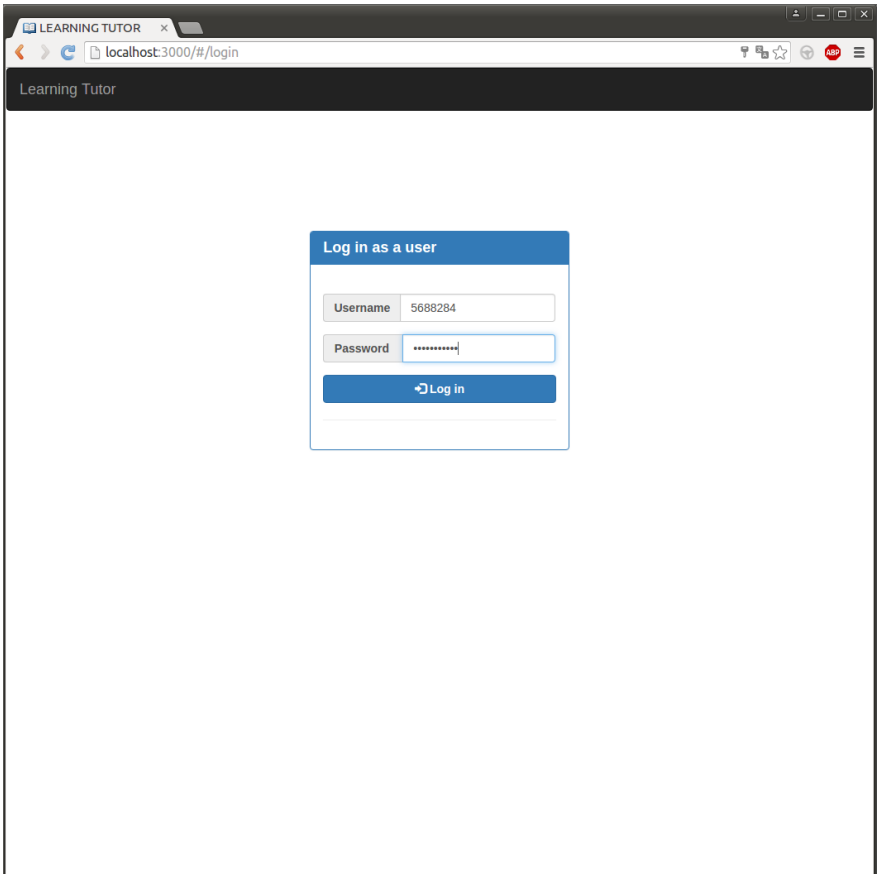


Figura 30: Pantalla de inicio de sesión del sistema

2. Tras ingresar en el sistema se muestra la pantalla “Subjects”, figura 31. Esta pantalla contiene el listado de las asignaturas en las que el estudiante ha sido matriculado por el profesor. Al pulsar sobre el icono “Search” de una asignatura concreta, se mostrará una lista de objetos de aprendizaje relacionados con el contenido de la asignatura. Otra forma de buscar objetos de aprendizaje es haciendo uso del buscador situado en la parte superior de la pantalla. Se introduce una lista de palabras clave que definan el contenido del objeto de aprendizaje al que se desea acceder y se pulsa sobre la “lupa” para realizar la búsqueda.

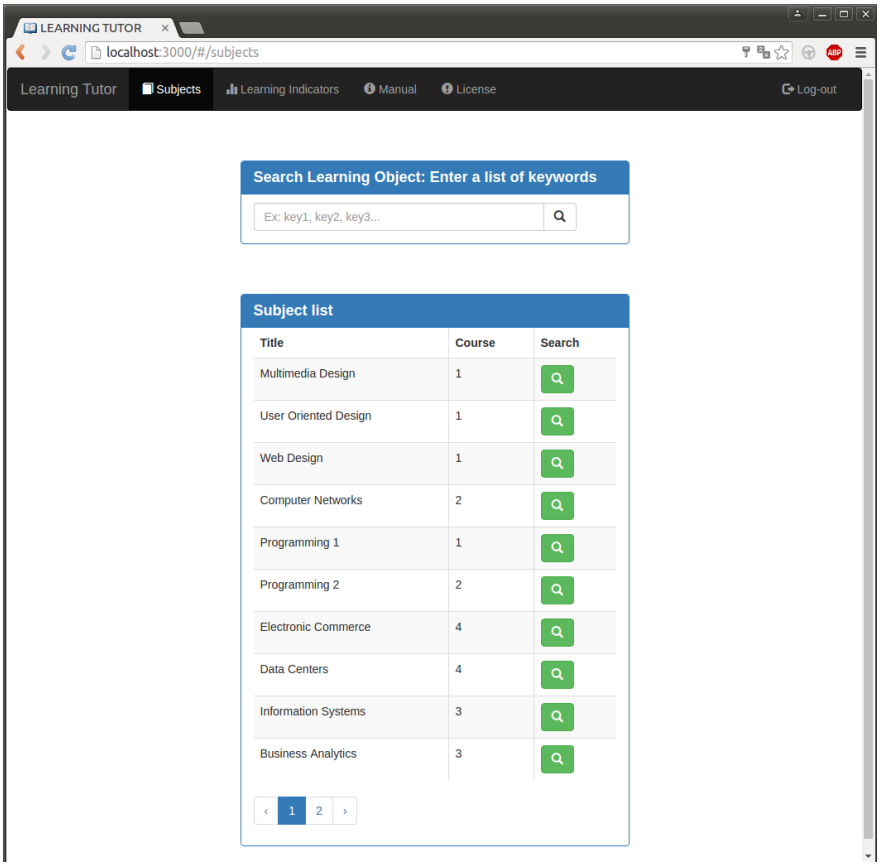


Figura 31: Pantalla Subjects del sistema

3. Tras haber realizado una búsqueda de objetos de aprendizaje, bien por palabra clave, o a través de una asignatura, aparecerá la pantalla “Learning Objects” correspondiente a la figura 32. En esta pantalla se muestra la lista de objetos de aprendizaje generados a raíz de la búsqueda anterior. Utilizando el botón “Link” del objeto de aprendizaje deseado se puede accede al contenido del mismo.

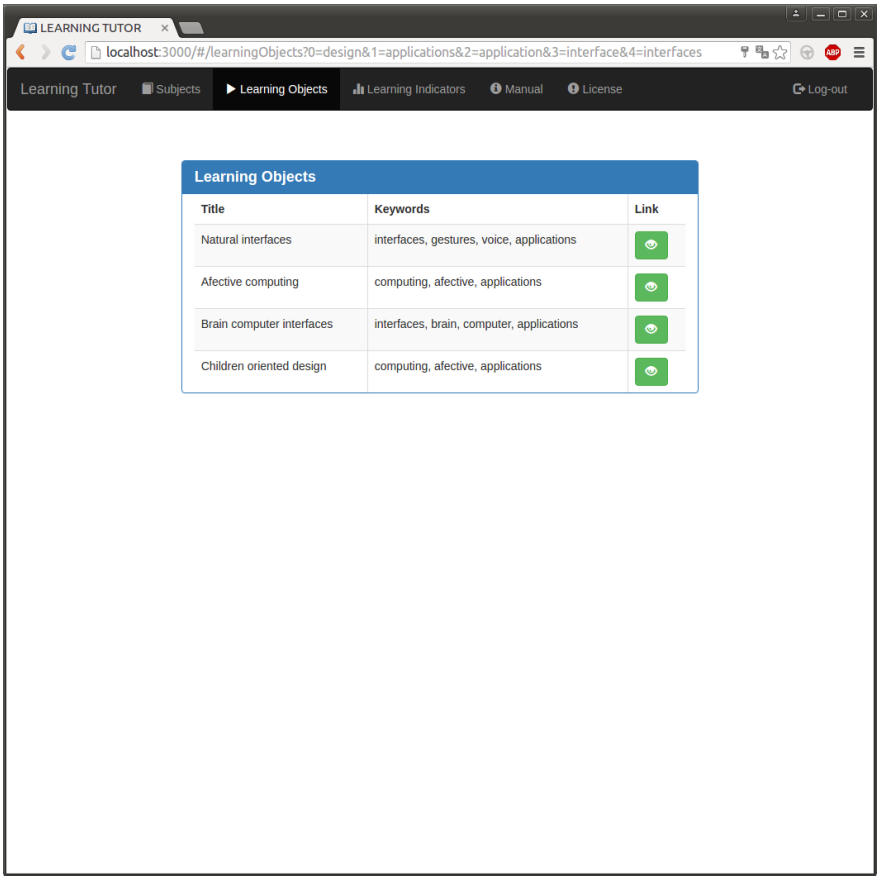


Figura 32: Pantalla Learning Objects del sistema

4. Una vez seleccionado un objeto de aprendizaje, el sistema mostrará el contenido del mismo, figura 33. En primer lugar se puede observar un mensaje de advertencia, mensaje que aparecerá únicamente si el estudiante ya ha almacenado un Topic Map en el sistema para este objeto de aprendizaje. En caso de introducir uno nuevo, el Topic Map almacenado será eliminado junto con sus indicadores de aprendizaje. A la izquierda de la pantalla se puede ver el reproductor de vídeo con el contenido del objeto de aprendizaje. Tras visualizar el vídeo, se ha de introducir un Topic Map en el sistema que describa el contenido del vídeo.

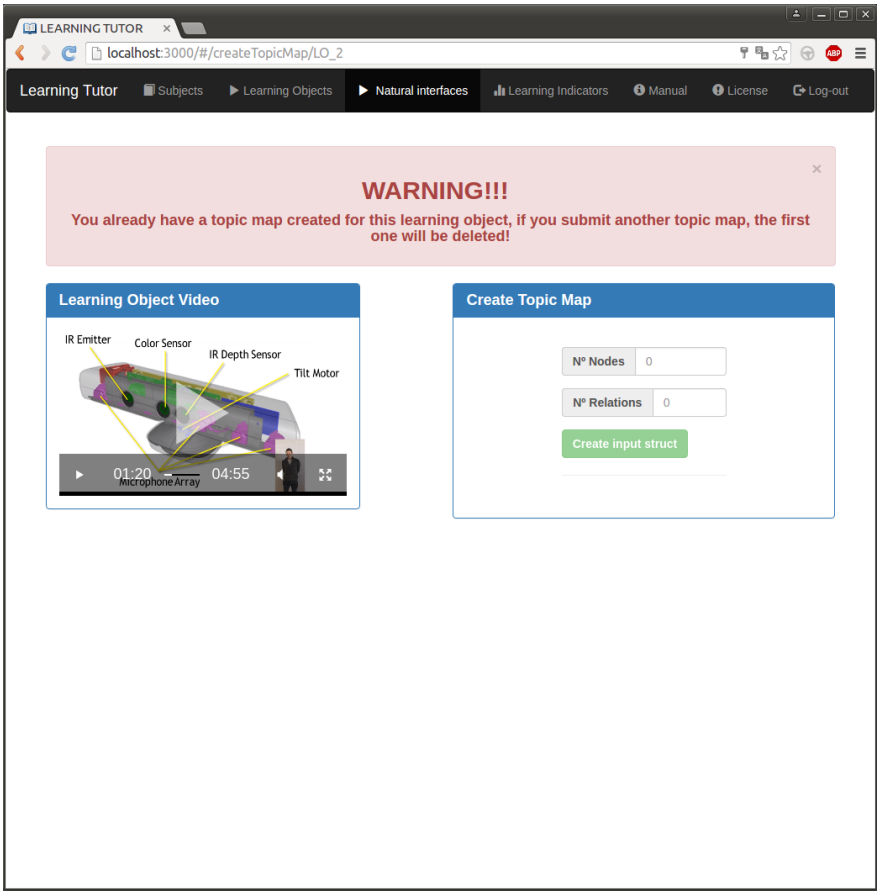


Figura 33: Pantalla contenido de un Learning Object del sistema

5. La introducción del Topic Map en el sistema se lleva a cabo haciendo uso del formulario situado en la parte derecha de la pantalla. En primer lugar hay que introducir el número de nodos y de relaciones que va a tener el Topic Map y pulsar sobre “Create input struct”. Una vez hecho esto, se genera una estructura que permite la inserción de cada nodo y relación del Topic Map, figura 34. Cada nodo ha de estar formado por un nombre y un valor numérico (1 – 3), siendo 3 un valor de importancia mayor. Cada relación se compone de un índice nodo origen, un índice nodo destino y una descripción de la relación (verbo). Aparecen en color rojo los campos aún sin rellenar o aquellos que presentar errores en su contenido. Opcionalmente, se puede introducir una imagen que represente gráficamente el Topic Map a través del botón “Choose a file”, figura 35. Una vez completos todos los campos, se puede presionar el botón “Submit Topic Map”, ésto almacenará el Topic Map en el sistema.

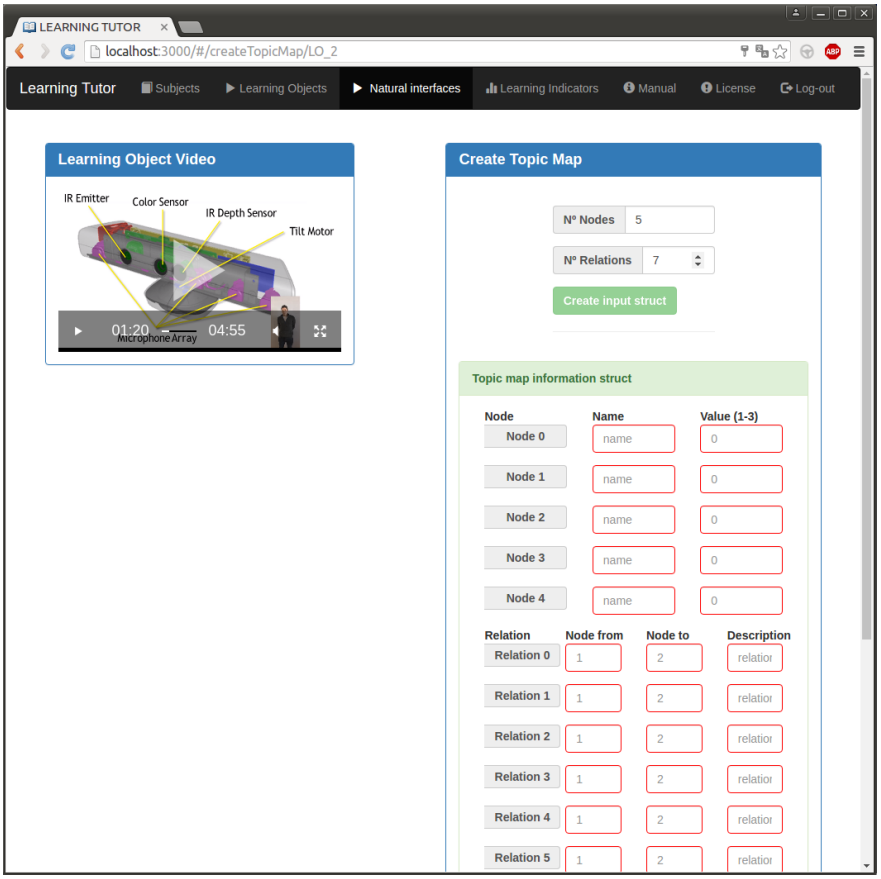


Figura 34: Estructura de introducción de un Topic Map en el sistema

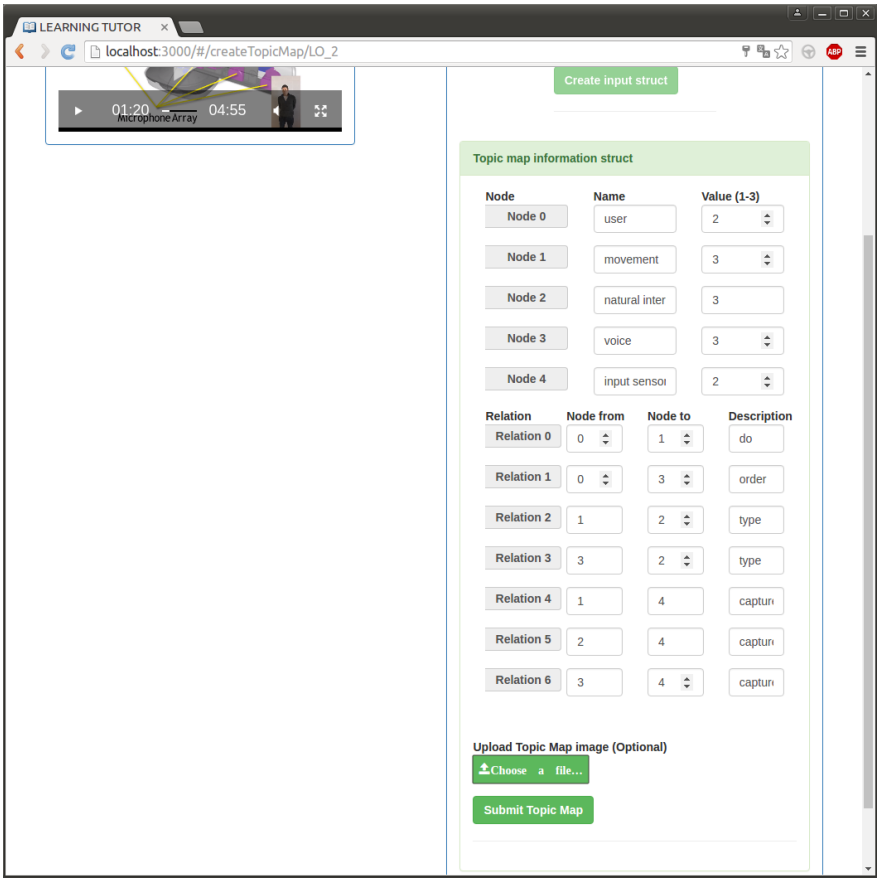


Figura 35: Estructura de introducción de un Topic Map en el sistema 2

6. Una vez almacenado el Topic Map, se mostrará en la pantalla un nuevo botón “Evaluate Topic Map”, figura 36. Al pulsar sobre él comenzará la evaluación semántica del Topic Map introducido en el sistema.

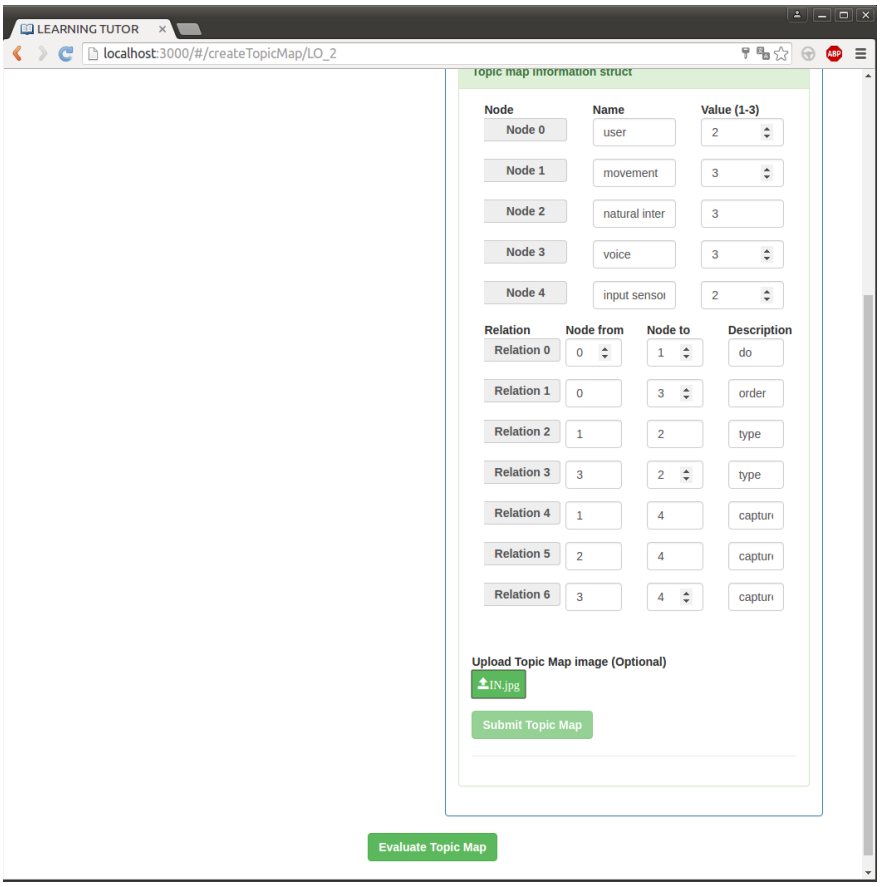


Figura 36: Botón de evaluación de Topic Map

7. Tras la evaluación se mostrará una pantalla con los indicadores de aprendizaje obtenidos, figura 37. En la parte superior se puede observar la imagen almacenada en el sistema junto a dos links que permiten la descarga de la propia imagen y del Topic Map introducido en formato de texto. En la parte inferior aparecen los indicadores de aprendizaje. A la izquierda se muestran los cuatro indicadores de similaridad junto a un cuadro de color. Este cuadro indica el grado de conocimiento adquirido para cada indicador, siendo verde el más alto y rojo el más bajo. A la derecha aparecen los conceptos introducidos por el profesor en su Topic Map de referencia que no han sido identificados por el estudiante en el suyo.

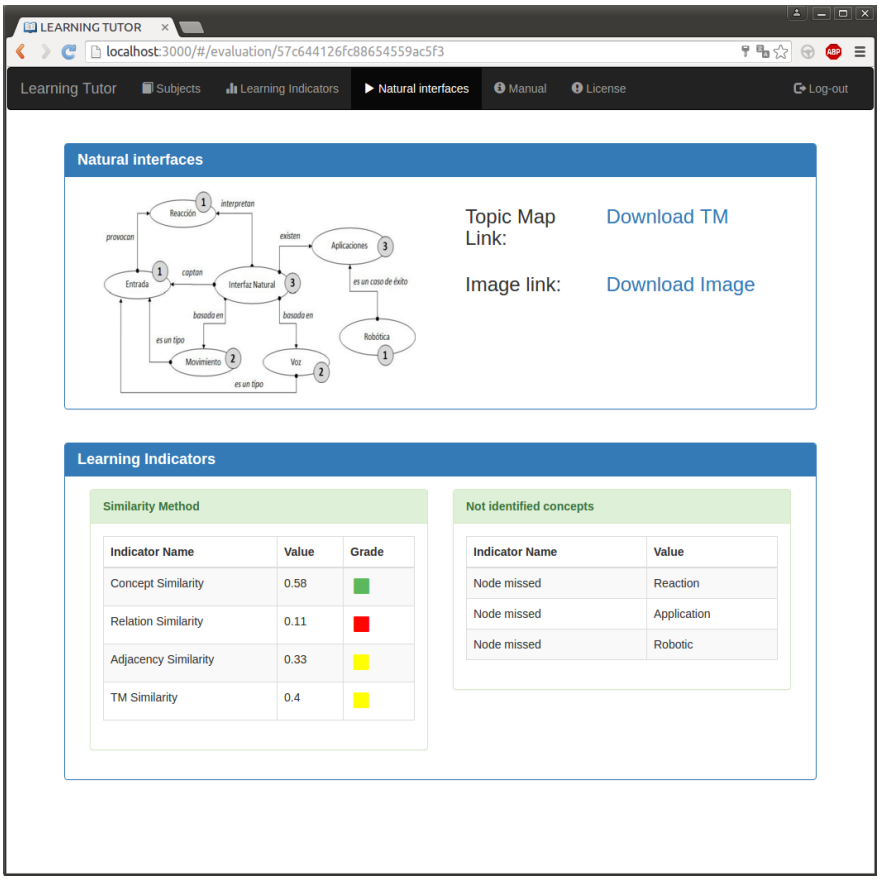


Figura 37: Pantalla de inidicadores de aprendizaje de un Topic Map concreto del sistema

8. Si una vez abandonada la pantalla anterior se desea volver a visualizar estos indicadores de aprendizaje, puede hacerse a través de la pantalla “Learning indicators”, figura 38. Esta pantalla es accesible haciendo uso de la barra de navegación. Muestra un listado de todos los objetos de aprendizaje para los que se ha almacenado y evaluado un Topic Map. Pulsando sobre el botón “Link” de uno de ellos se accederá a una pantalla que muestre los indicadores de aprendizaje obtenidos, equivalente a la explicada en el punto 7.

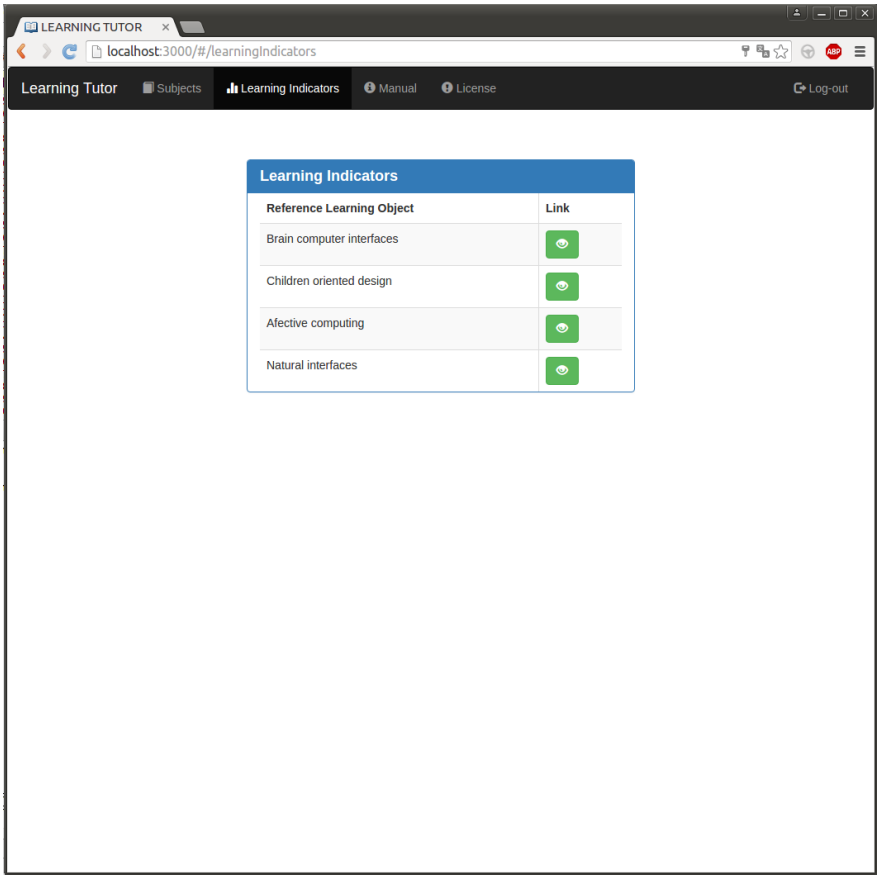


Figura 38: Pantalla Learning Indicators del aprendizaje del sistema

E.2. Manual de uso para el administrador del sistema

El objetivo de esta sección es explicar a un profesor cómo hacer uso de la cuenta de administrador con el fin de gestionar usuarios, asignaturas y objetos de aprendizaje. Se detallan una serie de pasos a seguir:

1. A través de la pantalla de inicio de sesión, figura 39, introducir las credenciales de la cuenta de administrador definida en el despliegue del sistema.

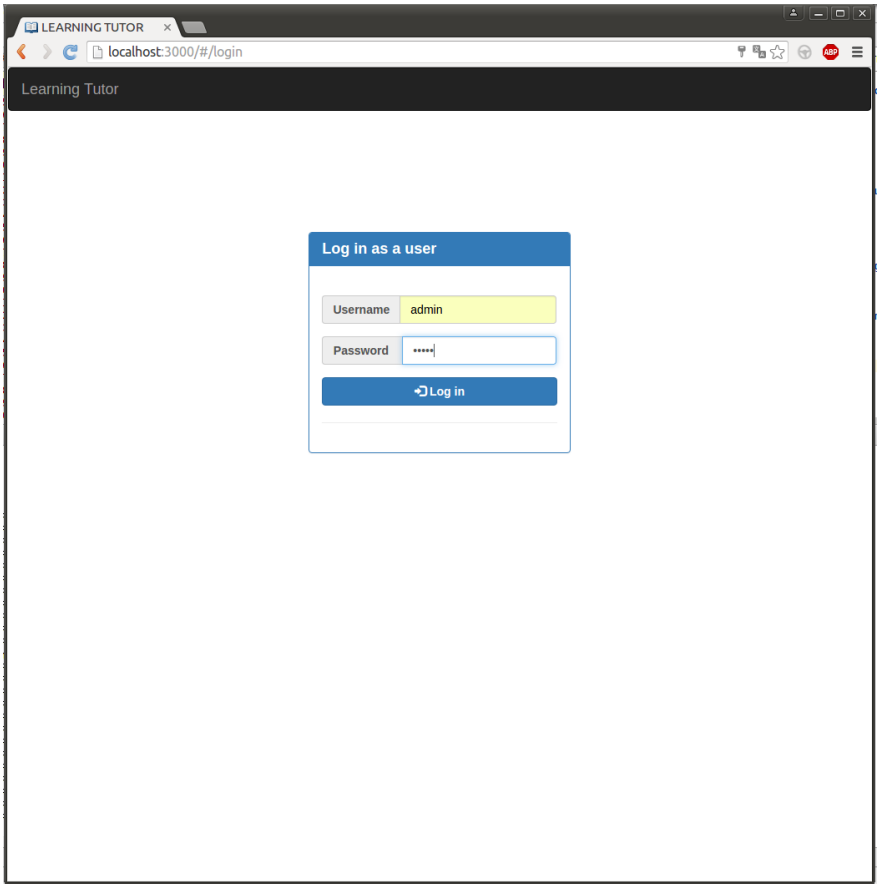


Figura 39: Pantalla de inicio de sesión del sistema con cuenta Admin

2. Tras iniciar sesión se muestra la pantalla “Manage users”, figura 40. Esta pantalla permite la visualización, creación, edición y eliminación de cuentas estudiante. En primer lugar se puede observar un listado con todos las cuentas de usuario almacenadas en el sistema. Sobre cada cuenta, a excepción de la de administración, se pueden realizar tres acciones: visualización, edición y borrado. Al pulsar sobre el primer botón se mostrarán los detalles de la cuenta seleccionada, figura 41. El segundo botón abre una nueva pantalla en la que se pueden editar los atributos de ese usuario concreto, figura 42. El tercer botón sirve para eliminar esa cuenta del sistema. En cuanto a la creación de un nuevo usuario, puede hacerse rellenando el formulario que aparece en el margen derecho de la pantalla.

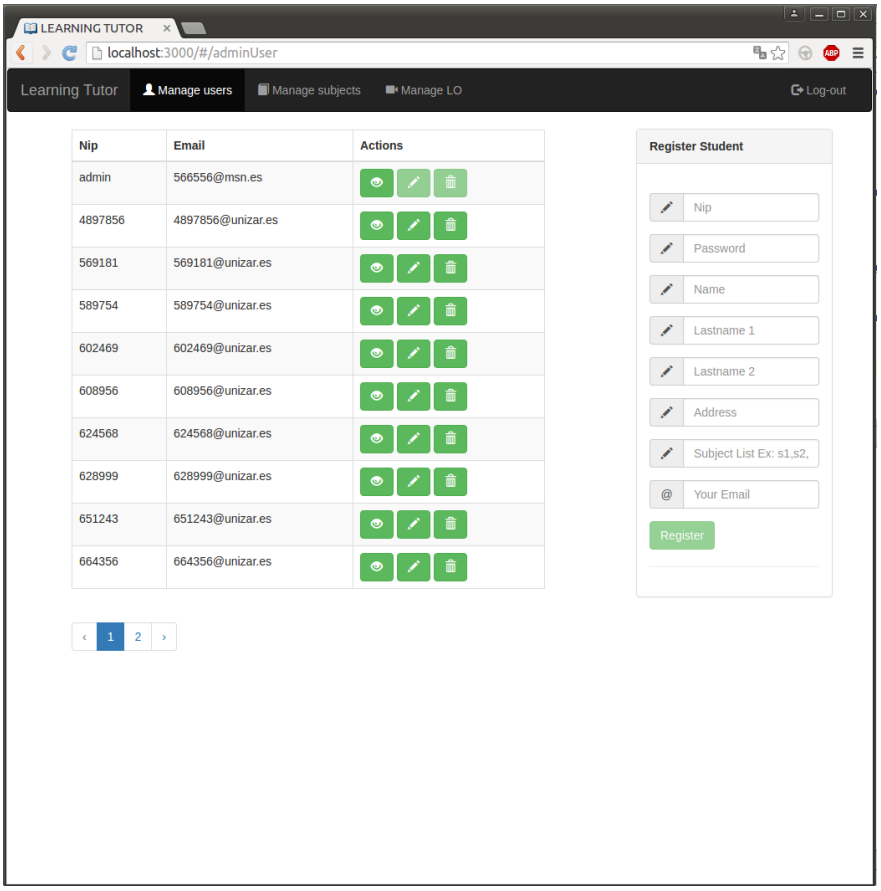


Figura 40: Pantalla Manage Users del sistema

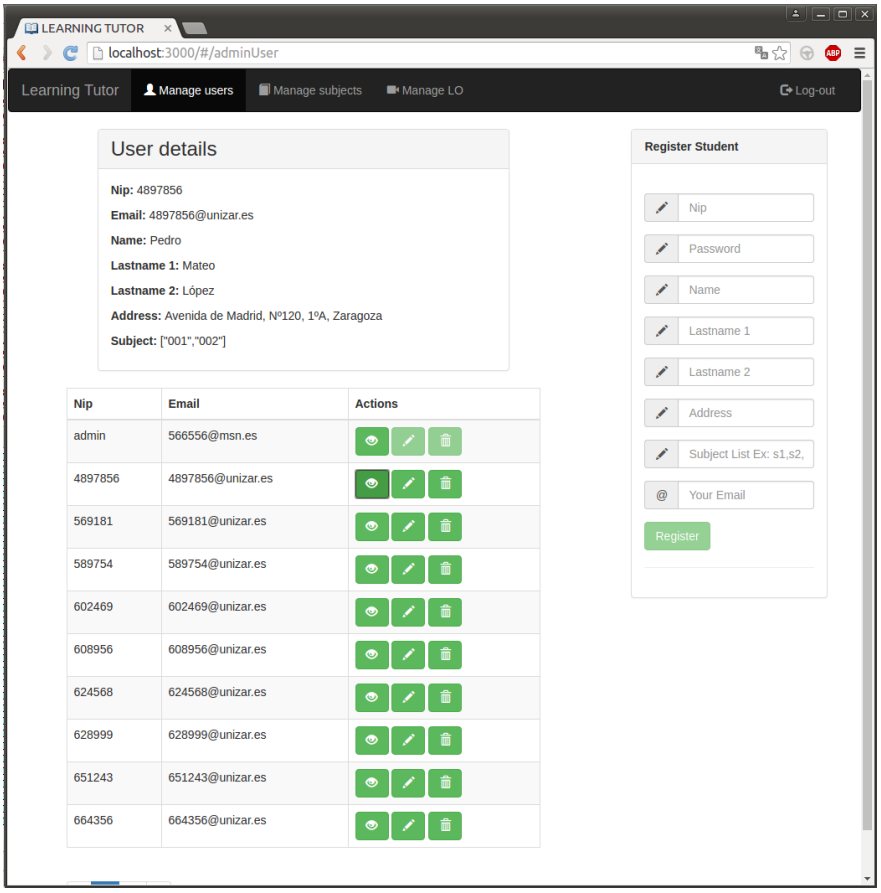


Figura 41: Detalles de una cuenta de usuario del sistema

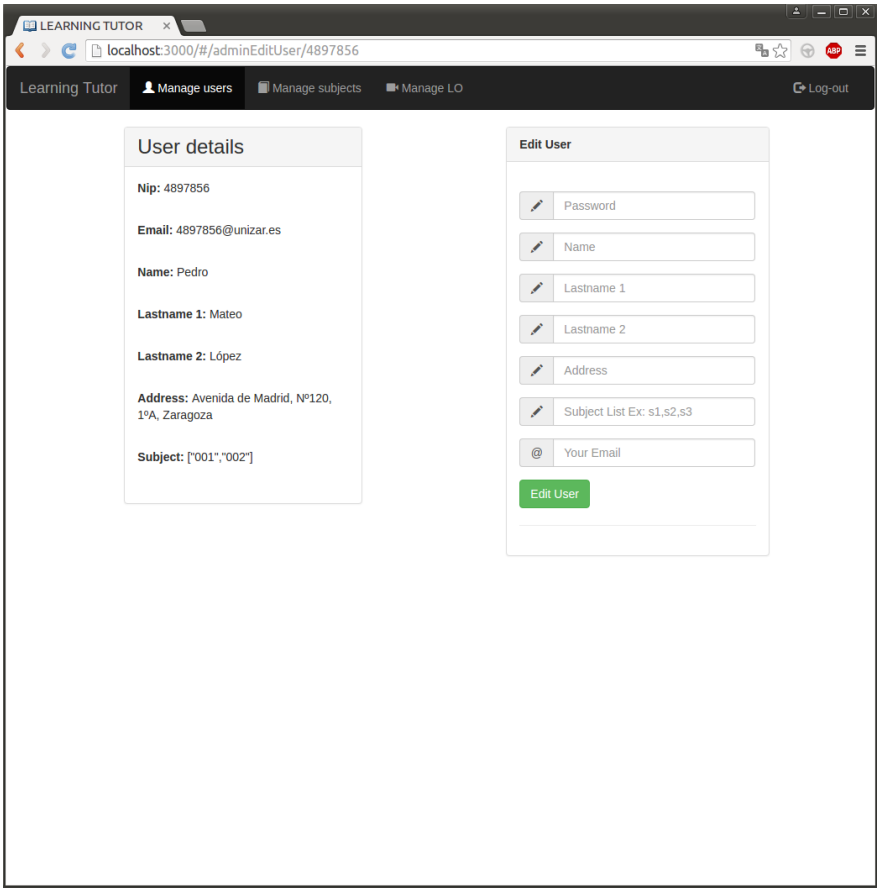


Figura 42: Pantalla de edición de una cuenta de usuario del sistema

3. Haciendo uso de la barra de navegación se puede acceder a la pantalla “Manage subjects”. Como se puede observar en la figura 43, muestra la misma estructura que la pantalla “Manage users”. Se pueden llevar a cabo las mismas acciones, visualización, edición, borrado e inserción de una nueva asignatura a través del formulario mostrado en la derecha de la pantalla. Un ejemplo de visualización de una asignatura puede observarse en la figura 44.

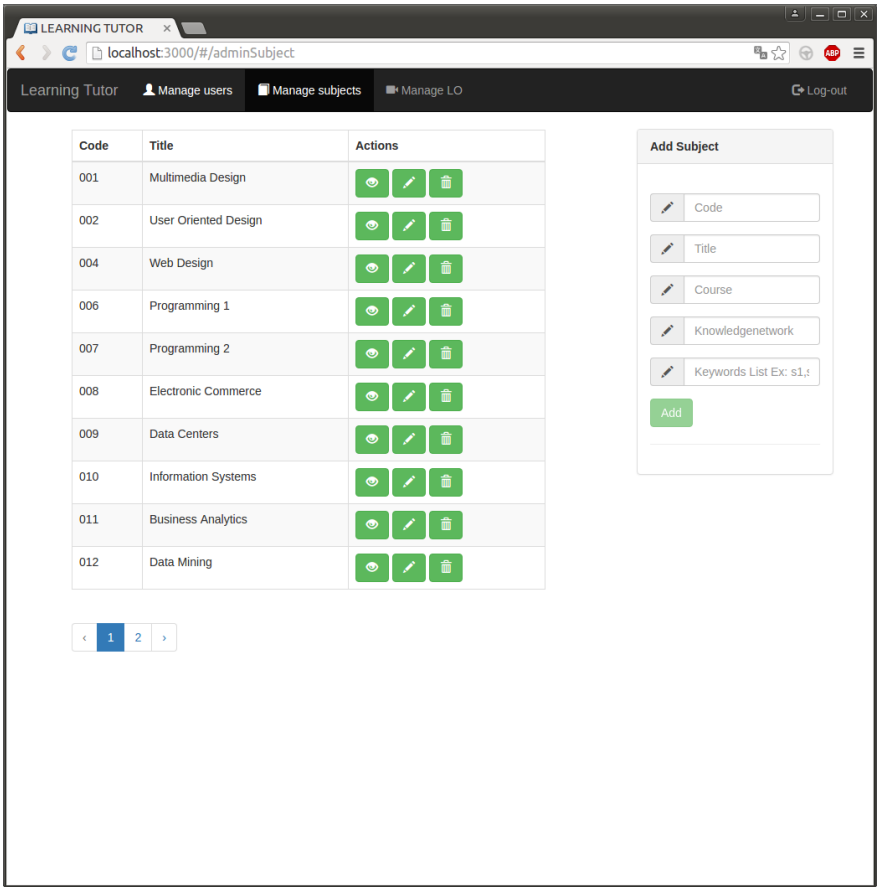


Figura 43: Pantalla Manage Subjects del sistema

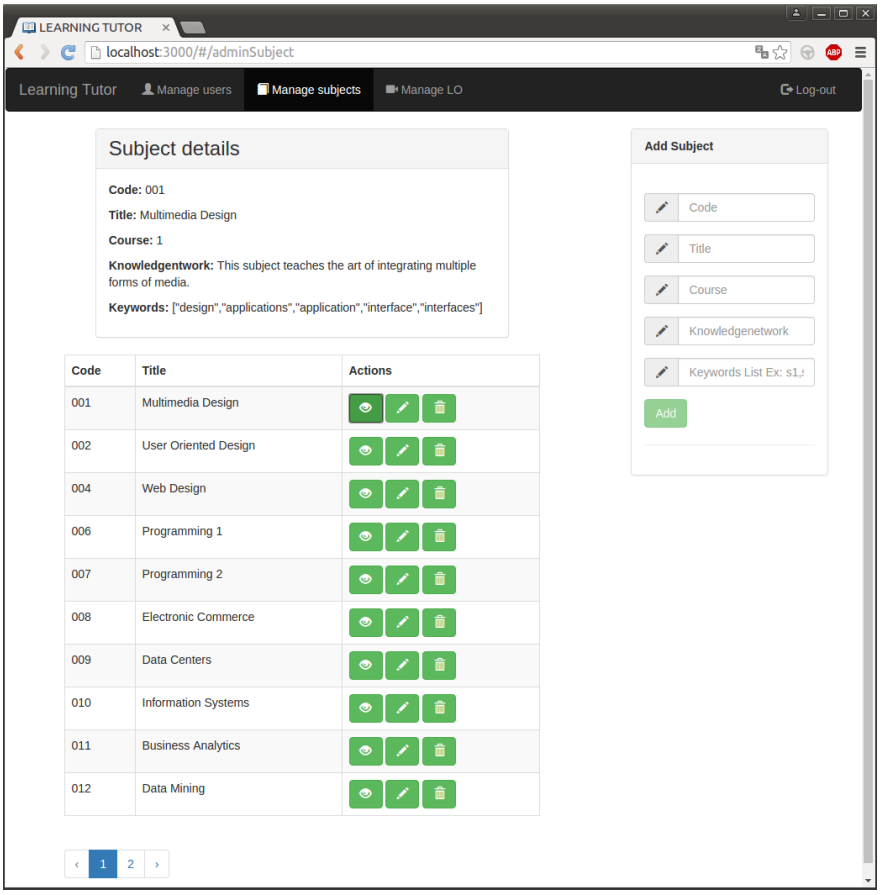


Figura 44: Detalle de una asignatura del sistema

4. Volviendo a hacer uso de la barra de navegación puede accederse a la pantalla “Manage LO”, figura 45. Esta pantalla tiene como finalidad la administración de objetos de aprendizaje. Al igual que en los casos anteriores, puede observarse una lista con todos los objetos de aprendizaje almacenados en el sistema. En la figura 46 puede apreciarse un ejemplo de visualización de los detalles de un objeto de aprendizaje concreto. La inserción de un objeto de aprendizaje se realiza mediante el uso de la estructura presente a la derecha de la pantalla. En primer lugar se ha de introducir el número de nodos y de conceptos que va a tener el Topic Map de referencia para este objeto de aprendizaje, pulsar tras ello sobre el botón “Create input struct”. Ésto genera una nueva estructura, figura 47, en la que hay que introducir en primer lugar los nodos y relaciones de este Topic Map, y, a continuación, el resto de información que describe el objeto de aprendizaje. Nótese que algunos campos están marcados como opcionales. Una vez completa la estructura se pulsa sobre el botón “Add learning object” con el fin de almacenar el objeto de aprendizaje en el sistema.

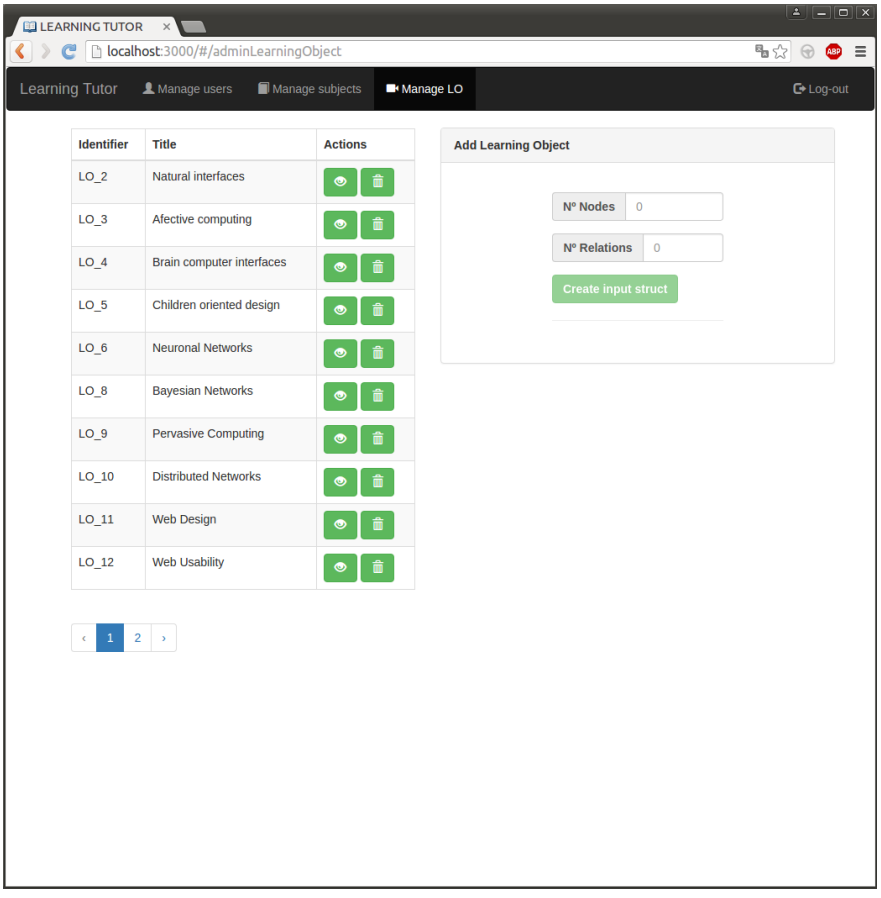


Figura 45: Pantalla Manage LO del sistema

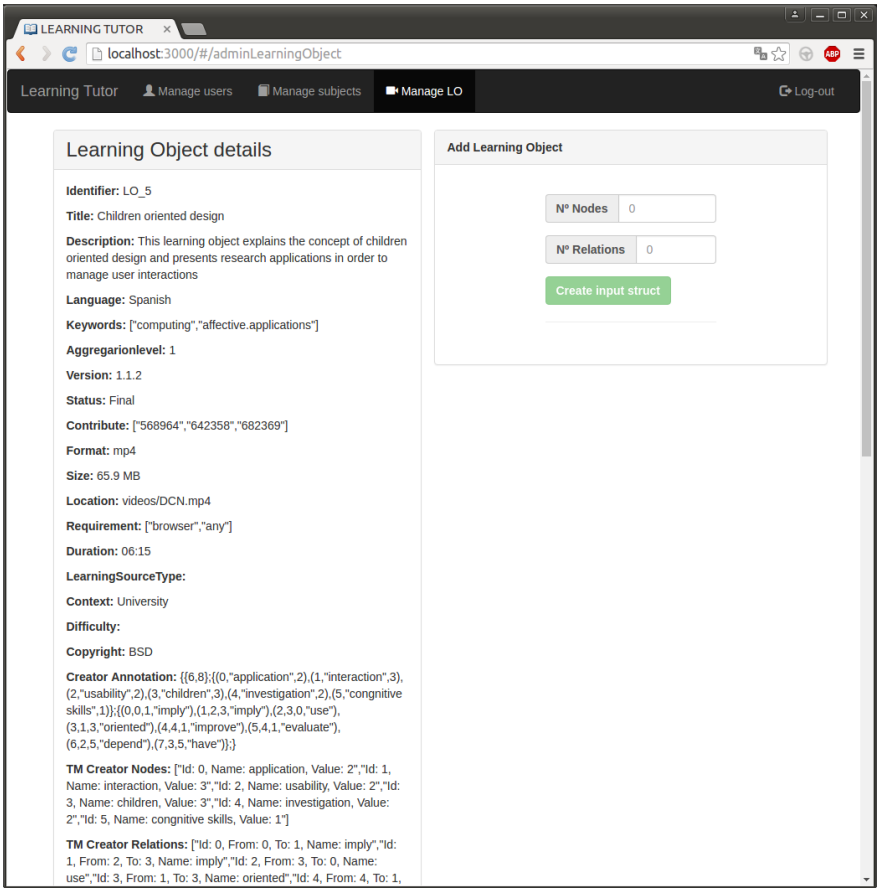


Figura 46: Detalle de un objeto de aprendizaje del sistema

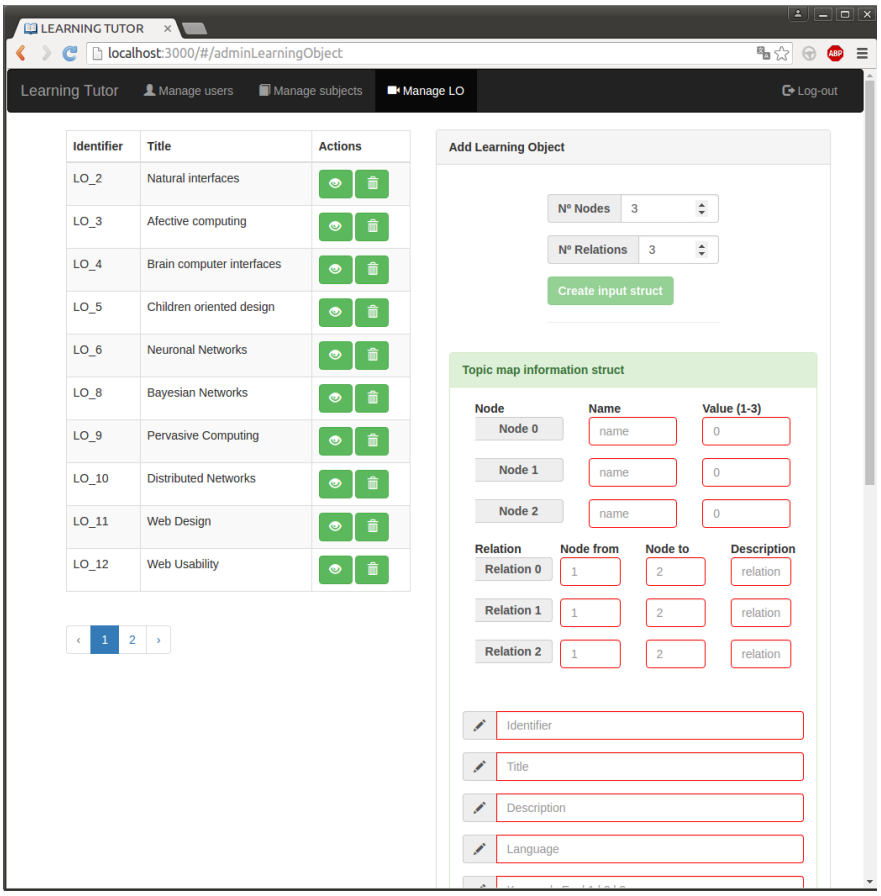


Figura 47: Estructura de inserción de un objeto de aprendizaje en el sistema

F. Referencias

[1] Leonard Richardson, & Mike Amundsen (2007). *RESTful Web Services*.

[2] AngularJS. *AngularJS by Google*. https://angularjs.org/

[3] Boostrap. *Boostrap 3*. http://getbootstrap.com/

[4] Linux Foundation Collaborative Projects. *NodeJS*. https://nodejs.org/en/

[5] Fundación NodeJS. *ExpressJS*. http://expressjs.com/es/

[6] World Wide Web Consortium (W3C). *Web Services Architecture*. https://www.w3.org/TR/ws-arch/

[7] Leon Shklar, & Rich Rosen (2009). *Web Application Architecture: Principles, Protocols and Practices* The client server paradigm (pp. 14-16).

[8] World Wide Web Consortium (W3C). *Hypertext Transfer Protocol* . https://www.w3.org/Protocols/rfc2616/rfc2616.html

[9] Oracle. *Java EE Technical Documentation*. http://docs.oracle.com/javaee/7/index.html

[10] MongoDB. *MongoDB Documentation*. https://docs.mongodb.com

[11] World Wide Web Consortium. (W3C) *NoSQL*. http://www.w3resource.com/mongodb/nosql.php

[12] Mongoose. *Mongoose*. http://mongoosejs.com/

[13] World Wide Web Consortium (W3C). *HTML*. https://www.w3.org/standards/webdesign/htmlcss

[14] World Wide Web Consortium (W3C). *CSS*. https://www.w3.org/standards/webdesign/htmlcss

[15] World Wide Web Consortium (W3C). *JavaScript Web Apis*. https://www.w3.org/standards/webdesign/script

[16] Wordnik. *Developer Wordnik*. http://developer.wordnik.com/docs.html

[17] Microsoft. *Microsoft Azure*. https://azure.microsoft.com/es-es/

[18] Oracle. *Oracle Eclipse*. https://eclipse.org/

[19] Jetbrains. *WebStorm The smartest JavaScript IDE*. https://www.jetbrains.com/webstorm/

[20] GitHub, Inc. *GitHub*. https://github.com/

[21] The Document Foundation. *LibreOffice*. https://es.libreoffice.org/

[22] The LaTeX Project. *LaTeX - A document preparation system.* https://www.latex-project.org/

[23] Notepad++. *Notepad++.* https://notepad-plus-plus.org/

[24] Creately. *Creately.* http://creately.com/

[25] The PHP Group. *PHP.* https://secure.php.net/

[26] Google. *Chrome V8 - Google's high performance, open source, JavaScript engine.* https://developers.google.com/v8/

[27] MongoDB. *Document Databases.* https://www.mongodb.com/document-databases

[28] Bsonspec. *BSON.* http://bsonspec.org/

[29] JSON. *JavaScript Object Notation (JSON).* http://www.json.org/

[30] Shyam Seshadri, & Brad Green (2014). *AngularJS: Up and Running* What is MVC (Model View Controller) (pp. 2-4).

[31] Mean.io. *The Friendly & Fun JavaScript Fullstack.* http://mean.io

[32] World Wide Web Consortium (W3C). *Cloud Computing.* https://www.w3.org/wiki/CloudComputing

[33] Leon Shklar, & Rich Rosen (2009). *Web Application Architecture: Principles, Protocols and Practices* Multi tier web applications (pp. 201-203).

[34] Erich Gamma, & Richard Helm, & Ralph Johnson, & John Vlissides (1994). *Design Patterns: Elements of Reusable Object-Oriented Software* Facade (pp. 208-218).

[35] Eduardo Mena, & Santiago Velilla, & Javier Lacasta. *Nivel conceptural de una Base de Datos. El modelo ER.* http://webdiis.unizar.es/asignaturas/BD/transparenciasBD/PDFs_1x1/leccion_3.pdf

[36] Vrije Universiteit Amsterdam. *Hash Functions and Hash Tables.* http://www.cs.vu.nl/~tcs/ds/lecture6.pdf

[37] Leonard Richardson, & Mike Amundsen (2007). *RESTful Web Services* The Protocol Semantics of HTTP (pp. 33-45)

[38] AuthO. *JSON Web Tokens.* https://jwt.io/

[39] GitHub. *Expressjs - Morgan.* https://github.com/expressjs/morgan

[40] GitHub. *WinstonJS.* https://github.com/winstonjs/winston

[41] NPM. *Java - Bridge API to connect with existing Java APIs.* https://www.npmjs.com/package/java

[42] Oracle. *Packaging Programs in JAR Files.* https://docs.oracle.com/javase/tutorial/deployment/jar/

[43] Mozilla.org. *JSON.stringify().* https://developer.mozilla.org/es/docs/Web/JavaScript/Referencia/Objetos_globales/JSON/stringify

[44] Oracle. *Class ArrayList.* https://docs.oracle.com/javase/7/docs/api/java/util/ArrayList.html

[45] Oracle. *Class HashMap.* https://docs.oracle.com/javase/7/docs/api/java/util/HashMap.html

[46] CheckMyColours. *Analyze the colors of any webpage to verify optimal contrasts for better accessibility..* http://www.checkmycolours.com/

[47] Agilemethodology. *The Agile Movement.* http://agilemethodology.org/

[48] W. Durfee (2008). *University of Minnesota: Project Planning and Gantt Charts.* http://www.me.umn.edu/courses/me2011/handouts/proj_planning.pdf

[49] Gary Chartrand (1984). *Introductory Graph Theory* (p. 218)

[50] VisJS. *A dynamic, browser based visualization library..* http://visjs.org/index.html

[51] SigmaJS. *JavaScript library dedicated to graph drawing..* http://sigmajs.org/

Las Web han sido accedidas por última vez el 29 de Agosto de 2016.