



Incorporación de un sistema de almacenamiento de experiencias a un ecosistema educativo para permitir el uso de técnicas de análisis de datos (Educational Analytics)

Rubén Darío Espinosa Roldán

Universidad EAFIT
Escuela de Ingeniería
Medellín, Colombia
2015

Incorporación de un sistema de almacenamiento de experiencias a un ecosistema educativo para permitir el uso de técnicas de análisis de datos (Educational Analytics)

Rubén Darío Espinosa Roldán

Tesis de grado presentada como requisito parcial para optar al título de:
Magíster en Ingeniería

Director(a):
Doctor Juan Guillermo Lalinde Pulido

Línea de Investigación:
Tecnologías de Información para la Educación
Grupo de Investigación:
Desarrollo e Innovación en TIC (GIDITIC)
Departamento de Informática y Sistemas

Universidad EAFIT
Escuela de Ingeniería
Medellín, Colombia
2015

Declaración de autoría

Yo, Rubén Darío Espinosa Roldán, me permito afirmar que he realizado la presente tesis de manera autónoma y con la única ayuda de los medios permitidos y no diferentes a los mencionados en la propia tesis. Todos los pasajes que se han tomado de manera textual o figurativa de textos publicados y no publicados, los he reconocido en el presente trabajo. Ninguna parte del presente trabajo se ha empleado en ningún otro tipo de tesis.

Medellín, Antioquia. 10 de Agosto, 2015.

Rubén Espinosa

Rubén Darío Espinosa Roldán

Agradecimientos

A la línea de investigación en informática educativa del GIDITIC por su apoyo en el desarrollo de esta tesis.

A mi asesor, sin su ayuda y dedicación no hubiera podido alcanzar las metas propuestas para este trabajo.

A mis padres, por su búsqueda insaciable de hacer de mí una gran persona.

Resumen

La Universidad EAFIT, en los últimos años, por medio de la realización de varias investigaciones, ha estado desarrollado una propuesta con la cual se busca definir los componentes tecnológicos que deben componer un ecosistema de aplicaciones educativas, con el fin de apalancar la adopción del modelo de ubicuidad en las instituciones de educación superior.

Por medio del grupo de investigación desarrollo e innovación en Tecnologías de la Información y las Comunicaciones (GIDITIC) ha realizado la selección de los primeros componentes del ecosistema en trabajos de tesis de grado de anteriores investigaciones[1, 2]. Adicionalmente, algunos trabajos realizados por el gobierno local de la Alcaldía de Medellín en su proyecto de Medellín Ciudad Inteligente[3], también realizó una selección de algunos componentes que son necesarios para la implementación del portal. Ambas iniciativas coinciden en la inclusión de un componente de registro de actividades, conocido como “Sistema de almacenamiento de experiencias” (LRS).

Dados estos antecedentes, se pretende realizar una implementación de un LRS que cumpla con los objetivos buscados en el proyecto de la Universidad, siguiendo estándares que permitan asegurar la interoperabilidad con los otros componentes del ecosistema de aplicaciones educativas.

Contenido

Agradecimientos	VII
Resumen	IX
Abreviaturas	XII
Lista de Figuras	XIII
Lista de Tablas	XIV
1. Introducción	1
2. Planteamiento del Problema	3
2.1. Objetivo General	3
2.1.1. Objetivos Específicos	3
2.2. Justificación	4
2.3. Pregunta de Investigación	4
3. Estado del Arte	5
3.1. Internacional	5
3.2. Nacional	9
4. Marco de Referencia Conceptual	11
4.1. Plataformas de gestión del aprendizaje	11
4.2. Repositorio de registro del aprendizaje	12
4.3. Estándares de interoperabilidad	12

5. Propuesta de Implementación	15
5.1. Arquitectura	15
5.1.1. Vista Conceptual	15
5.1.2. Vista Lógica	21
5.1.3. Vista Física	25
5.2. Prototipo	30
5.2.1. Documentación	31
5.2.2. Aplicaciones	34
5.2.3. Sentencias	35
5.2.4. Estadísticas	36
5.3. Ejemplo de un cliente para el API	39
6. Conclusiones y Trabajo a Futuro	43
6.1. Conclusiones	43
6.2. Trabajo a Futuro	44
Bibliografía	46

Abreviaturas

Abreviatura	Término
<i>ADL</i>	Advanced Distributed Learning
<i>API</i>	Application Programming Interface
<i>DDD</i>	Domain Driven Design
<i>GIDITIC</i>	Grupo de Investigación, Desarrollo e Innovación en TIC
<i>IMS GLC</i>	Instructional Management Systems Global
<i>IEEE</i>	Institute of Electrical and Electronics Engineers
<i>JSON</i>	JavaScript Object Notation
<i>LRS</i>	Learning Record Storage
<i>Modelo TAG</i>	Tecnología Aprendizaje Gestión
<i>MVC</i>	Modelo Vista Controlador
<i>ORM</i>	Object Relational Mapping
<i>REST</i>	REpresentational State Transfer
<i>SCO</i>	Sharable Courseware Object
<i>SOA</i>	Sevice Oriented Architecture
<i>SCORM</i>	Sharable Content Object Reference Model
<i>TIC</i>	Tecnologías de la Información y la Comunicación
<i>VLE</i>	Virtual Learning Environments

Lista de Figuras

3-1.	Componentes del estándar SCORM. [4].	6
3-2.	Flujo sencillo del estándar LTI. [5].	8
3-3.	Módulos de la arquitectura del portal Medellín Ciudad Inteligente [6].	10
4-1.	Ecosistema de aplicaciones de un portal educativo [7].	13
5-1.	Diagrama de paquetes.	16
5-2.	Actores del sistema.	17
5-3.	Diagrama de caso de uso - Crear Usuario.	18
5-4.	Diagrama de caso de uso - Registrar Evento.	19
5-5.	Diagrama de caso de uso - Publicar Sentencia.	19
5-6.	Diagrama de caso de uso - Consultar Sentencia.	20
5-7.	Flujo MVC.	22
5-8.	Diagrama de clases.	23
5-9.	Diagrama de Paquetes.	24
5-10.	Vista Física.	25
27figure.5.11		
5-12.	Vista lógica.	28
5-13.	Pantalla de inicio.	30
5-14.	Documentación del API.	31
5-15.	Administración de aplicaciones.	35
5-16.	Listado de sentencias.	36
5-17.	Estadísticas	37
5-18.	Gráfico de sentencias publicadas por día.	38

Lista de Tablas

4-1. Plataformas de Gestión del Aprendizaje (LMS).	12
4-2. Organizaciones dedicadas al establecimiento de estándares de interoperabilidad.[8]	13
4-3. Estándares de interoperabilidad en educación más relevantes.[9]	14
5-1. Actores del sistema.	17

1 Introducción

Desde finales de los años 90, cuando surgieron las primeras plataformas para ofrecer cursos virtuales, con el fin de llevar la educación por fuera de el aula de clase, han surgido nuevas plataformas y estándares con el fin de lograr este objetivo. Con el fin de realizar un seguimiento al progreso y al comportamiento de los estudiantes en las plataformas educativas LMS, se desarrollo en el año 2003 el estándar SCORM, el cual permitió el avance en la integración de múltiples plataformas WEB con los LMS, sin embargo, con el auge en el desarrollo de plataformas móviles, simulaciones y juegos con fines educativos, el protocolo SCORM fue una limitante, ya que estaba dirigido solamente a plataformas WEB.

Con estos antecedentes, la Universidad EAFIT con su grupo de investigación Proyecto 50, crearon una linea de investigación con el fin de mejorar la calidad educativa apalancados en las soluciones tecnológicas, de esta iniciativa nace el proyecto en el cual se desarrolla el Modelo TAG (Tecnología Aprendizaje, Gestión) para la valoración de la ubicuidad en instituciones de educación superior, en la cual se plantean algunos componentes que deben conformar el ecosistema de aplicaciones educativas, las cuales deben apoyar el proceso formativo. Uno de los componentes importantes es el repositorio de experiencias.

En esta tesis se presenta la selección de un repositorio de almacenamiento de experiencias basado en la especificación xAPI dada por la organización ADL en el proyecto Tin Can, además de esto, se realiza una propuesta de implementación, en la cual se especifica la arquitectura de software, la cual esta compuesta por las capas conceptual, lógica y física. Con base a la ar-

arquitectura propuesta, finalmente, se realiza un prototipo de implementación software, en la cual se refleja todos los componentes tecnológicos especificados en la arquitectura propuesta.

2 Planteamiento del Problema

Este capítulo presenta la justificación para la realización de esta tesis de grado, presentando en el los objetivos que se busca alcanzar al concluir este trabajo.

2.1. Objetivo General

Diseñar un sistema de almacenamiento de experiencias a un ecosistema educativo, basado en la propuesta realizada por el proyecto TinCan con el repositorio de experiencias, el cual cumpla con condiciones de flexibilidad, interoperabilidad y desacoplamiento, que a su vez permita a las diferentes plataformas que componen el ecosistema educativo, registrar los eventos resultantes de la interacción con los diferentes actores (profesores, estudiantes, administradores), y que adicionalmente permita el uso de técnicas de análisis de datos por medio de la integración de aplicaciones de terceros.

2.1.1. Objetivos Específicos

- Diseñar un sistema de almacenamiento basado en la especificación realizada por el proyecto TinCan de la organización ADL.
- Construir el estado del arte en la integración de plataformas educativas con sistemas de registro de actividades.
- Describir los elementos que componen una sentencia valida de un LRS, basado en la especificación como Activity Streams.
- Implementar un prototipo de la solución de software basado en el patrón de arquitectura para diseño de aplicaciones distribuidas (SOA) por medio del protocolo REST.

2.2. Justificación

La Universidad EAFIT (Medellín-Colombia), en el marco de su aniversario número 50, dio inicio a una propuesta “con el propósito de potenciar las competencias de los docentes a través de la innovación en los procesos de enseñanza, aprendizaje e investigación creativa con el uso de TIC” [10], es así como nace de la mano Proyecto 50 y el Grupo de Investigación, Desarrollo e Innovación sobre TIC de la Universidad EAFIT (GIDITIC) el proyecto “Modelo TAG (Tecnología Aprendizaje, Gestión) para la valoración de la ubicuidad en instituciones de educación superior” [1].

Una de las dimensiones evaluadas en el modelo TAG, es la dimensión tecnológica, esta comprende en su diseño la inclusión de arquitecturas de referencia (Infraestructura y patrones de diseño) que permiten evaluar el nivel de ubicuidad de las instituciones de educación superior. Es en este punto, donde este proyecto pretende dar continuidad al objetivo de la Universidad en la labor investigativa de la Universidad.

2.3. Pregunta de Investigación

¿Basado en la propuesta realizada por la organización ADL en el repositorio de experiencias xAPI, cuales son los patrones arquitectónicos y herramientas tecnológicas que permiten diseñar e implementar un sistema de almacenamiento de experiencias LRS, basado en los principios de aplicaciones distribuidas, escalables y desacopladas?

3 Estado del Arte

Este capítulo realiza una revisión a las iniciativas internacionales y nacionales, las cuales fueron realizadas con diferentes objetivos y con su realización permiten dar bases teóricas y técnicas en el desarrollo este trabajo, lo cual permitirá realizar una propuesta de implementación de un sistema de almacenamiento de experiencias (LRS).

3.1. Internacional

En el ámbito internacional, el desarrollo de los estándares para compartir información entre portales educativos ha estado liderado por Estados Unidos, es así como en el año 1999, en el durante el gobierno de William Jefferson Clinton, se delegó al departamento de defensa la creación de la organización ADL (Advanced Distributed Learning). Esta organización tenía como fin que los empleados federales pudieran “sacar el máximo provecho de los avances tecnológicos con el fin para adquirir las habilidades y el aprendizaje necesario para tener éxito en un lugar de trabajo” [11]. La conformación de la organización ADL buscaba poder realizar un “establecimiento de directrices sobre el uso de normas y proporcionar un mecanismo para ayudar en el desarrollo a gran escala, la implementación y la evaluación de los sistemas de aprendizaje interoperables y reutilizables”. [12].

Uno de los primeros logros obtenidos por ADL en términos de interoperabilidad, fue el desarrollo del estándar SCORM (Sharable Courseware Object Reference Model). Los objetivos planteados que buscaba ADL con el desa-

rollo de SCORM[13] eran:

- Permitir el intercambio de cursos a través de sistemas LMS por medio de un formato de empaquetamiento.
- Permitir la comunicación entre el contenido y los sistemas LMS.
- Promover la reutilización de contenido a través de diferentes cursos.
- Permitir la secuenciación de contenidos de adaptados a la persona.

El estándar de SCORM, en su versión más actualizada (2004) se compone de un *entorno de ejecución*, un *modelo de agregación de contenidos* y una especificación de *secuenciación y navegación* (Ver figura 3-1). El modelo de agregación y navegación es el encargado de definir y compartir el contenido, el contenido obtiene el nombre de SCO (Sharable Courseware Object)[14]. El entorno de ejecución es el componente encargado de establecer la comunicación entre el LMS y el SCO[15]. Adicionalmente, el módulo de secuenciación y navegación es el encargado de definir el orden que los recursos serán presentados al usuario[16].

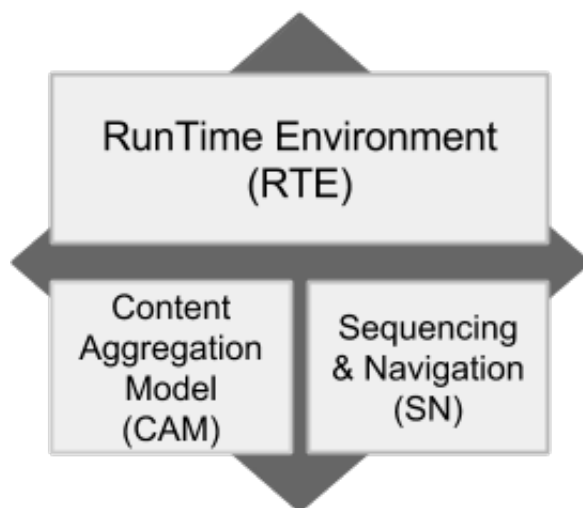


Figura 3-1: Componentes del estándar SCORM. [4].

La organización IMS (Instructional Management Systems), la cual es una organización sin ánimo de lucro fundada en el año 1995, con el propósito

de permitir el crecimiento y el impacto de las tecnologías de aprendizaje en los sectores educativos y corporativos en todo el mundo. IMS proporciona liderazgo en la formación y el crecimiento de la industria del aprendizaje a través del desarrollo de estándares de interoperabilidad y en herramientas de aprendizaje y tecnología educativa.[17]

Dadas las nuevas tendencias que se estaban presentando en el desarrollo de contenido educativo y la necesidad de incluirlo en los cursos desarrollados en los sistemas LMS, la organización IMS comienza con la tarea de permitir la integración de herramientas desarrolladas por terceros en el LMS, es así como en el año 2010[18] desarrollan la primer especificación del estándar “LTI” (Learning Tools Interoperability). “El concepto principal de LTI es del de establecer una forma estándar de integrar aplicaciones ricas de aprendizaje (en algunos casos alojados remotamente y proveídos a través de servicios de terceros) con plataformas de aprendizaje tipo LMS, portales u otras plataformas del entorno educativo.”.[5] El estandar LTI se compone de tres elementos. *Tool Provider*, *Tool* y *Tool Consumer*.

La especificación de LTI permite que aplicaciones sencillas como un foro, o aplicaciones mas avanzadas como simulaciones matemáticas puedan ser adicionadas a las plataformas que los usuarios acceden, evitando así que los usuarios tengan la necesidad de registrarse en otras plataformas y salir del LMS. Adicionalmente la herramienta (Tool) informara al LMS (Comsumer) sobre el resultado obtenido por el usuario en la herramienta que fue lanzada, este flujo se puede ver en la figura **3-2**.

Sin embargo, los estándares de SCOMR y LTI en su especificación solo incluyen el seguimiento al usuario por medio del reporte del resultado, y deja por un lado todo el aprendizaje significativo que pudo obtener el usuario durante el desarrollo de la actividad, adicional a esto, con el desarrollo de contenido educativo por fuera de los sistemas LMS, estos protocolos comenzaron a ser un obstáculo en el desarrollo de nuevas herramientas que permitieran reali-

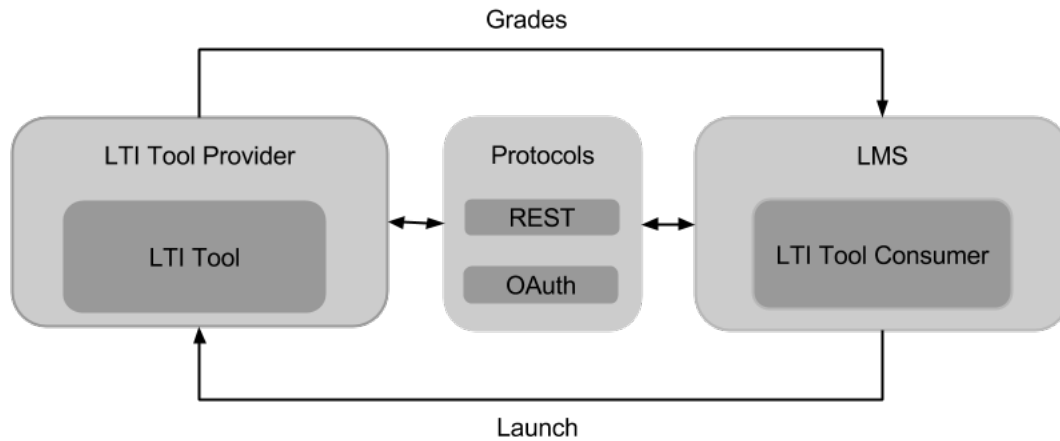


Figura 3-2: Flujo sencillo del estándar LTI. [5].

zar el seguimiento del aprendizaje en otros sistemas y entornos. Este hecho se empieza a evidenciar aún más con el auge en el desarrollo de herramientas de simulación, aplicaciones para teléfonos móviles y tabletas.

Teniendo en cuenta los retos mencionados anteriormente, la organización ADL en el año 2013 anuncio el proyecto TIN CAN[19], el cual fue adoptado como el sucesor del estándar SCORM, en su diseño adopta algunos de los protocolos y estándares que la comunidad de desarrolladores de software conocían a profundidad, es el caso de OAuth (administración de permisos de usuarios) y la especificación de *Activity Streams*. Esta última, fue desarrollada con el fin de proveer una “especificación técnica que permita expresar metadatos relacionada con actividades en un formato rico, amigable para los humanos pero al mismo de una manera que sea extensible y procesable por las maquinas” [20], por esto el formato acogido tiene la forma Actor-Verbo-Objeto, la cual permite registrar casi cualquier actividad realizada por un usuario en una plataforma en un lenguaje natural.

El proyecto TIN CAN logró solventar algunas de las limitantes de SCORM y LTI. Algunas de las características mas importantes son:

- Llevar el registro de aprendizaje por fuera de los navegadores web.

- Un modelo sólido modelo de seguridad usando OAuth.
- Habilidad de registrar juegos y simulaciones.
- Independencia del LMS y otros aplicativos del ecosistema educativo.
- Permitir al usuario intercambiar plataformas (Ej. Pasar del móvil al computador sin perder el progreso).

3.2. Nacional

El ámbito nacional, la propuesta realizada por la ciudad de Medellín en el proyecto “Portal Medellín Ciudad Inteligente” [3], el cual es un “programa de la Alcaldía de Medellín, que lidera la transformación de Medellín en una ciudad al servicio de la calidad de vida de sus ciudadanos a través del buen uso de las Tecnologías de la Información y la Comunicación, TIC, y el empoderamiento de los ciudadanos de su propio entorno.” [21]. La universidad EAFIT, en su rol de diseñador de la arquitectura de software para el portal, partió del principio de que “su arquitectura debe ser flexible, ajustada a estándares, segura y robusta” [22]. Por esta razón, la arquitectura propuesta se compone de 6 módulos (Ver figura **3-3**), los cuales permiten garantizar un alto nivel de flexibilidad para integrar nuevos componentes.

Uno de los componentes más importantes en la propuesta de la arquitectura del portal, es el componente de Experiencia (LRS), dado que este es el que permite agregar el valor de inteligencia¹ al portal, posibilitando así, que se pueda ofrecer sugerencias y personalización a la interfaz de usuario. “Con el fin de ilustrar la importancia de este módulo en el contexto de Ciudad Inteligente, supóngase que todo ciudadano tiene una identidad digital que no sólo le permite ingresar al portal sino que le da acceso a las bibliotecas, al metro, etc. Los sistemas de estas instituciones podrían alimentar el LRS con el registro de las experiencias del usuario en cada uno de esos contextos,

¹Un sistema inteligente es aquel que con base en la información que recibe de su entorno puede actuar y tomar decisiones.



Figura 3-3: Módulos de la arquitectura del portal Medellín Ciudad Inteligente [6].

de manera que dicha información pueda ser tenida en cuenta para adaptar el portal a cada usuario. Esto quiere decir que el LRS es el responsable de preservar toda la información que sirve de base para el análisis del perfil del usuario del portal y poder responder de manera personalizada a cada uno de ellos” [23]

4 Marco de Referencia Conceptual

En este capítulo se presentan los conceptos importantes para el desarrollo de este trabajo: Plataformas de aprendizaje, Repositorios de registro de aprendizaje y Estándares de interoperabilidad.

4.1. Plataformas de gestión del aprendizaje

Las plataformas de aprendizaje (LMS), son sistemas de que tienen como objetivo principal, la administración de recursos electrónicos que permitan el desarrollo de cursos a distancia, semi-presenciales y presenciales. Por lo general, los LMS incluyen las siguientes funcionalidades:

- Administración de roles (Profesores, estudiantes y administradores).
- Realización de evaluaciones.
- Desarrollo de actividades curriculares.
- Calendario de actividades de los cursos.
- Publicación de contenido.
- Foros de discusión.

Uno de las soluciones software más usado en cuanto LMS se refiere es moodle, la cual fue desarrollada a principio del año 2000 por Martin Dougiamas[24], como parte del artículo "Improving the Effectiveness of online Learning"[25]. Algunos ejemplos de plataformas de aprendizaje: ver tabla 4-1

Nombre	Sitio	Tipo de licencia
Moodle	http://moodle.org	Open Source
EAFIT Interactiva	http://interactiva.eafit.edu.co/ei	Propietario
Canvas	http://www.canvaslms.com/	Open Source
Black Board	http://anz.blackboard.com/	Comercial

Tabla 4-1: Plataformas de Gestión del Aprendizaje (LMS).

4.2. Repositorio de registro del aprendizaje

Un repositorio de registro del aprendizaje es un sistema que permite almacenar datos resultantes de la interacción de un actor (Estudiante, Profesor, Administrador) con algún componente del portal educativo, por ejemplo:

- El resultado de la solución de un examen.
- Hora de inicio de una actividad del curso.
- Pasos que realizó un estudiante durante la ejecución de una simulación.

Este sistema cumple un papel muy importante en el diseño que propone ADL¹ para la infraestructura de las plataformas educativas del futuro (Ver figura 4-1.). En su propuesta, se realiza un acercamiento a un ecosistema de aplicaciones que son independientes e interoperables entre sí, lo que permite que el repositorio de registro del aprendizaje pueda recopilar información de todos los sistemas incluidos en el portal, lo cual lo hace un componente vital al momento de realizar análisis de datos y comportamiento de los actores del sistema.

4.3. Estándares de interoperabilidad

Los estándares de interoperabilidad son normas y especificaciones que facilitan el desarrollo tecnológico de los sistemas y su integración con otros, la gestión de los recursos; que repercutan en el almacenamiento, intercambio y

¹ADL es una iniciativa del gobierno de los Estados Unidos, el cual fue creado con el fin de investigar, realizar prototipos e implementar las nuevas generaciones en ambientes de aprendizaje[11]

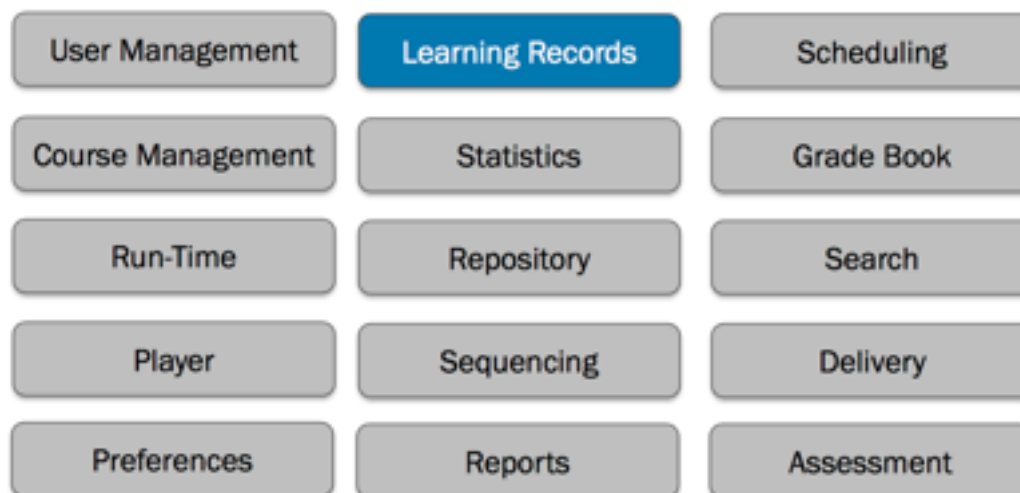


Figura 4-1: Ecosistema de aplicaciones de un portal educativo [7].

búsqueda de los contenidos. Es así como la estandarización de las tecnologías, permite la reutilización de recursos y la interoperabilidad entre sistemas y software heterogéneo.[8]

Existen varias organizaciones dedicadas al establecimiento de metodologías y protocolos de estandarización en temas de aplicaciones educativas, algunas de ellas son: (ver tabla 4-2)

Sigla	Nombre	Sitio
IMS	IMS Global Learning Consortium, Inc	http://www.imsglobal.org
W3C	World Wide Web Consortium	http://www.w3.org
ADL	Advancing Distributed Learning	http://www.adlnet.org
IEEE	Institute of Electrical and Electronics Engineers	https://www.ieee.org

Tabla 4-2: Organizaciones dedicadas al establecimiento de estándares de interoperabilidad.[8]

Para el objeto de estudio de este trabajo, hay algunos estándares que son más relevantes(ver tabla 4-3), estos han sido desarrollados a lo largo de la última década, permitiendo que los sistemas que componen el ecosistema de aplicaciones del portal educativo, puedan compartir objetos de aprendizaje²

²Cualquier entidad, digital o no digital, que puede ser utilizada, para el aprendizaje, la educación o el entrenamiento.[26]

Nombre	Descripción
LTI	El principal objetivo de “Learning Tools Interoperability” (LTI) es el de establecer una integración de aplicaciones ricas en aprendizaje (incluidas las alojadas remotamente y las proveídas por medio de servicios de terceros) con plataformas como LMS, portales, o otras del sector educativo.[18]
SCORM	Conjunto de estándares técnicos para software dedicado al e-learning. Su objetivo es el de habilitar la interoperabilidad de contenido de aprendizaje entre sistemas LMS, dando una especificación de como se da la comunicación entre los sistemas.[27]
TIN CAN API	También conocida como API de experiencia, es una nueva especificación para las tecnologías de aprendizaje, que hace posible recolectar datos de un amplio rango de experiencias que una persona tiene (online and offline). Su gran ventaja esta dado en la habilidad de capturar y compartir los datos de las actividades usando un vocabulario simple (Sujeto Verbo Objeto), el cual es compartido por medio de su conversión a mensajes JSON. A diferencia de SCORM, Tin Can API puede recolectar datos de simulaciones, móviles, realidad virtual, juegos serios, actividades de la vida real, aprendizaje experiencial y aprendizaje colaborativo.[28]

Tabla 4-3: Estándares de interoperabilidad en educación más relevantes.[9]

5 Propuesta de Implementación

Este capítulo presenta los detalles técnicos de la implementación de la plataforma LRS para la universidad EAFIT, se presenta la información del diseño de la arquitectura, apoyado en las practicas de ingeniería de software, adicionalmente se realiza una descripción de las tecnologías seleccionadas para el desarrollo del software.

5.1. Arquitectura

El propósito de este sección es exponer el diseño de la arquitectura de software de la plataforma LRS. Tomando como base los casos de uso del sistema, para esto, se construyen 4 vistas arquitectónicas para representar las principales características del sistema:

- **Vista conceptual:** Presenta los principales conceptos del sistema respecto a las necesidades expresadas por el cliente.
- **Vista lógica:** presenta la estructura lógica y de negocio importante para el proceso de desarrollo de la aplicación
- **Vista física:** presenta la distribución física que tendrán los diferentes componentes de la aplicación.
- **Vista de implementación:** presenta la interacción entre los diferentes componentes del sistema y su distribución.

5.1.1. Vista Conceptual

La vista conceptual permite definir la visión que los usuarios tienen de la aplicación. Esta vista muestra los subsistemas y módulos en los que se divi-

de la aplicación y la funcionalidad que brinda dentro de cada uno de ellos[29].

La figura **5-1** representa los módulos o subsistemas que hacen parte del sistema de registro del aprendizaje LRS.

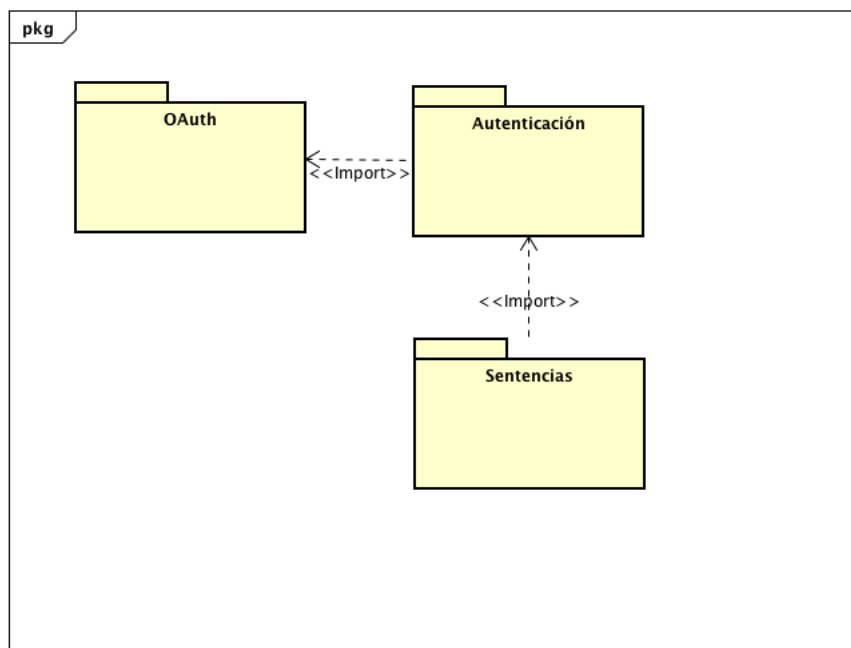


Figura 5-1: Diagrama de paquetes.

A continuación se presentan los casos de uso más relevantes con una corta descripción y la definición de los actores del sistema y su interacción con los módulos.

En la tabla **5-1**, se describen los actores que se encuentran involucrados con el sistema LRS.

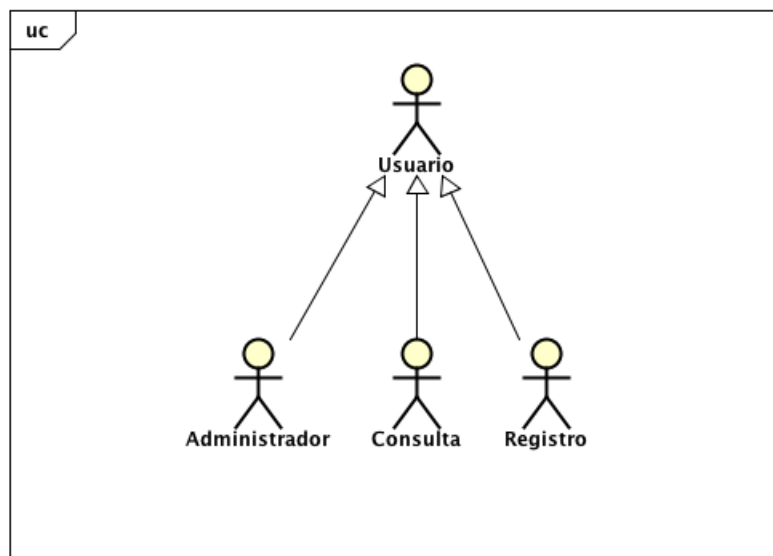


Figura 5-2: Actores del sistema.

Nombre del Actor	Descripción del actor y responsabilidades dentro del sistema
Administrador	Encargado de realizar la parametrización que requiere el sistema para su funcionamiento, es decir, es el encargado de crear usuarios de consulta y de registro.
Consulta	Usuario encargado de realizar consultas al API mediante el protocolo REST para leer <i>STATEMENTS</i> .
Registro	Usuario encargado de registrar <i>STATEMENTS</i> en el sistema mediante el protocolo REST.

Tabla 5-1: Actores del sistema.

Paquete de OAUTH: Este modulo representa la funcionalidad de la administración de usuarios y el manejo de credenciales para el acceso a la aplicación por medio del cliente WEB y las llamadas por medio del API.

- CU-01 - Crear Usuario: Responsable de la creación de los usuarios en la aplicación y la asignación de credenciales según el protocolo de OAUTH. (Ver figura 5-3)
- CU-02 - Notificar Usuario: Responsable de informar a los nuevos usuarios

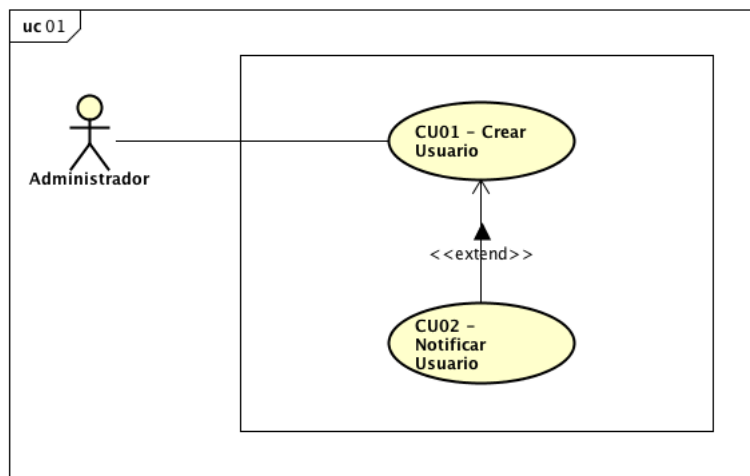


Figura 5-3: Diagrama de caso de uso - Crear Usuario.

los datos de acceso por medio de un email.

- CU-06 - Administrar Aplicación: Responsable de la administración de aplicaciones que pueden registrar y consultar sentencias, adicionalmente cada aplicación puede tener diferente nivel de permisos.

Paquete de Autenticación: Este modulo es el encargado de la administración de la autenticación en el sistema, dado que es necesario identificar los usuarios y sus permisos en la plataforma. Adicionalmente, este paquete ofrece dos tipos de autenticación, uno por el cliente WEB y por medio del API para aplicaciones de terceros que van a acceder a las sentencias almacenadas en la plataforma.

- CU-03 Validar Usuario: Responsable de garantizar el acceso a los usuarios de la plataforma. (Ver figura 5-4)
- CU-07 Registrar Evento: Responsable de registrar los eventos de autenticación fallidos y exitosos en la plataforma.

Paquete de Sentencias: Este paquete representa la funcionalidad de la gestión de las sentencias o statementens. Este es el encargado de permitir la creación y consulta de las mismas teniendo en cuenta los permisos de los usuarios.

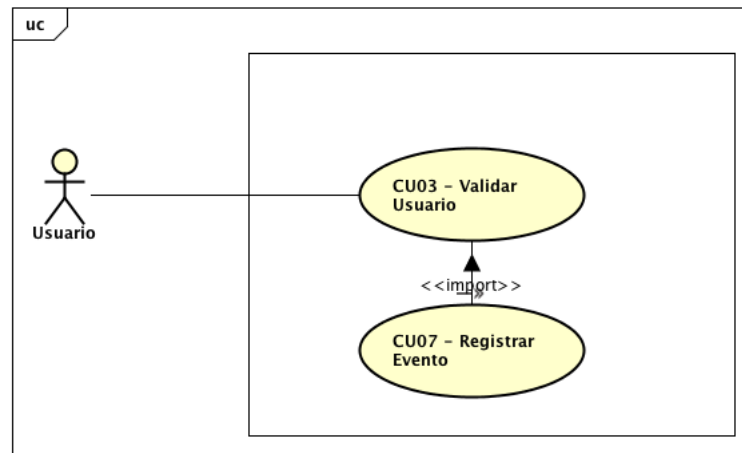


Figura 5-4: Diagrama de caso de uso - Registrar Evento.

- CU-04 Publicar Sentencia: Encargado de la creación de las sentencias en la plataforma, previamente valida que el usuario debe cumplir con el permiso de lectura. (Ver figura 5-5)

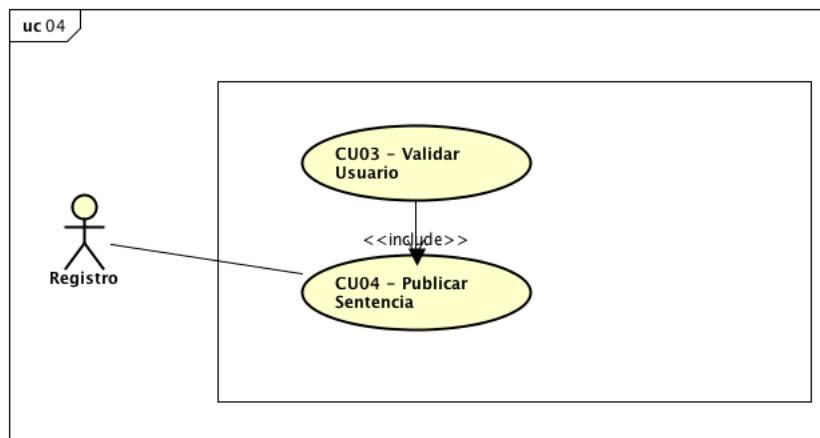


Figura 5-5: Diagrama de caso de uso - Publicar Sentencia.

- CU-05 Consultar Sentencia: Encargado de la creación de las sentencias en la plataforma, previamente valida que el usuario debe cumplir con el permiso de escritura. (Ver figura 5-6)

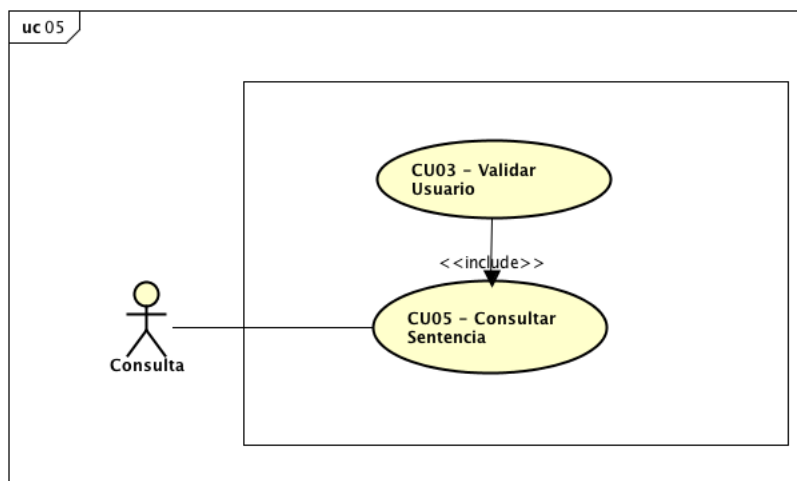


Figura 5-6: Diagrama de caso de uso - Consultar Sentencia.

5.1.2. Vista Lógica

La vista lógica representa una especificación técnica, la cual fue tomada en cuenta como base para el desarrollo de la plataforma, en esta se define en alto nivel los diseños arquitectónicos. En este diseño se tomó como base la propuesta definida en la arquitectura a N Capas dada en el diseño orientado por el dominio (Domain Driven Design), este tiene como ventaja principal, que el sistema es definido al rededor de la lógica del negocio, desarrollando así capas colaborativas y reusables que permitan ejecutar la lógica principal de la plataforma.

Adicionalmente, la plataforma está diseñada para ser desarrollada con el patrón de arquitectura MVC (Modelo Vista Controlador), el cual es un patrón que permite separar el modelo del dominio, la presentación y las acciones basadas en la interacción con el usuario en tres entidades separadas[30]:

- *El modelo* es el encargado de manejar la lógica de la aplicación, eventualmente, el modelo es el encargado de obtener y guardar información en una base de datos.
- *La vista* es la parte de la aplicación encargada de desplegar la información al usuario, por lo general, los datos son entregados por datos obtenidos en los modelos.
- *El controlador* es el encargado de manejar la interacción con el usuario, generalmente, los controladores obtienen datos de la vista, valida los datos y los envía a un modelo.

Siguiendo la propuesta arquitectónica DDD y teniendo en cuenta la estructura de una aplicación web, se identifican 2 capas principales:

- **Cliente:** La aplicación contará con un cliente HTML sencillo, el cual permitirá administrar los usuarios y las aplicaciones que accederán los datos por medio del API REST. El cliente realizará llamadas a los controladores alojados en la capa del servidor.

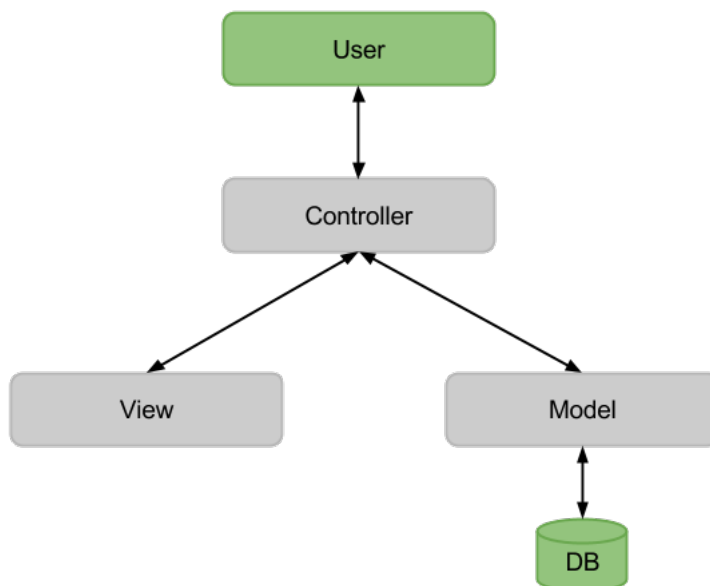


Figura 5-7: Flujo MVC.

- **Servidor:** A nivel del servidor se definen los servicios distribuidos y demás funcionalidades necesarias para garantizar el acceso a la fuente de datos asociada la plataforma; además proveen las interfaces necesarias para garantizar la conexión entre las capas del Cliente y los servicios publicados en el servidor.

A cada una de las Tiers identificadas para este sistema han sido subdivididas en un conjunto de capas y paquetes siguiendo las recomendaciones de la arquitectura por referencia DDD, las cuales se describen a continuación y se pueden observar gráficamente sus relaciones en la figura 5-9:

- **Cliente:**
 - **Presentación:** Esta capa contiene todas las pantallas necesarias para poner a disposición del usuario las diferentes funcionalidades del sistema.
 - **Vistas UI:** Paquete que contiene los recursos HTML, CSS y Javascript correspondientes a la vista de la plataforma.
- **Servidor:**

- **Servicios Distribuidos:** A esta capa pertenece todas las interfaces, clases y operaciones necesarias para garantizar el acceso a datos de forma remota desde el cliente. Las clases de esta capa se conectarán directamente con las clases del paquete Capa de Dominio :: Entidades de Dominio, para acceder a las funcionalidades específicas.
 - **Vistas:** Este paquete es el encargado de generar las vistas de la aplicación e incluir los datos suministrados por el controlador.
 - **Controladores:** Este paquete es el encargado de recibir los datos del cliente y generar una respuesta, interactúa con las Entidades del dominio y con las Vistas.
- **Dominio:** La capa de Dominio contiene la estructura de negocio del sistema acorde a las funcionalidades identificadas e implementa el patrón Active Record para gestionar el acceso a los datos cuando sea necesario (Ver figura 5-8).

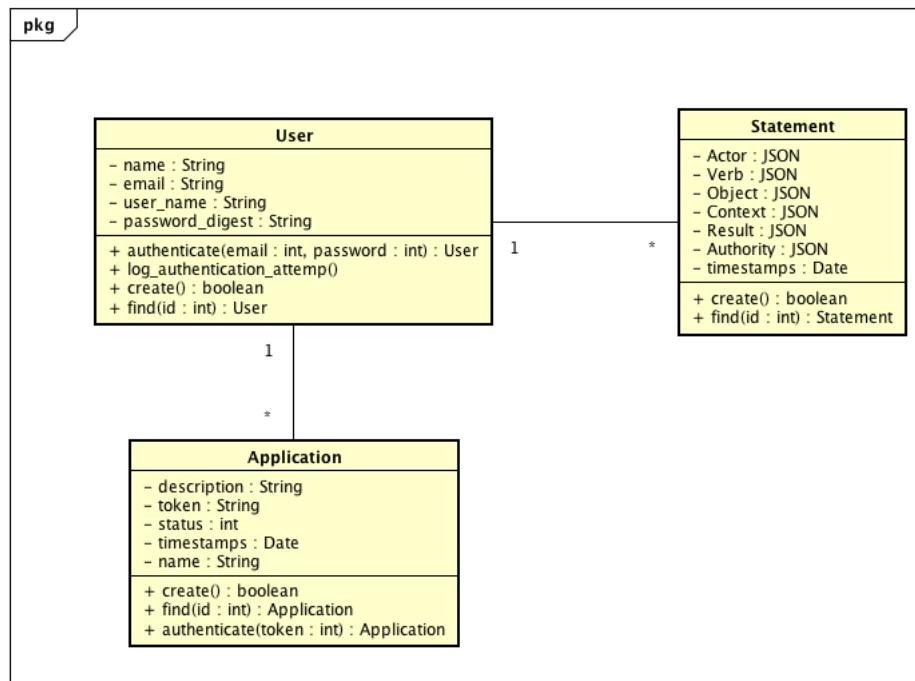


Figura 5-8: Diagrama de clases.

- **Infraestructura:** La capa de infraestructura de persistencia de datos contiene la estructura y funcionalidades necesarias para acceso

a datos.

- **Acceso a Datos:** Se usa del patrón Active Record, el cual es “un patrón que envuelve un registro de base de datos, una tabla o una vista, encapsula el acceso a la base de datos y añade lógica a esos datos” [31]

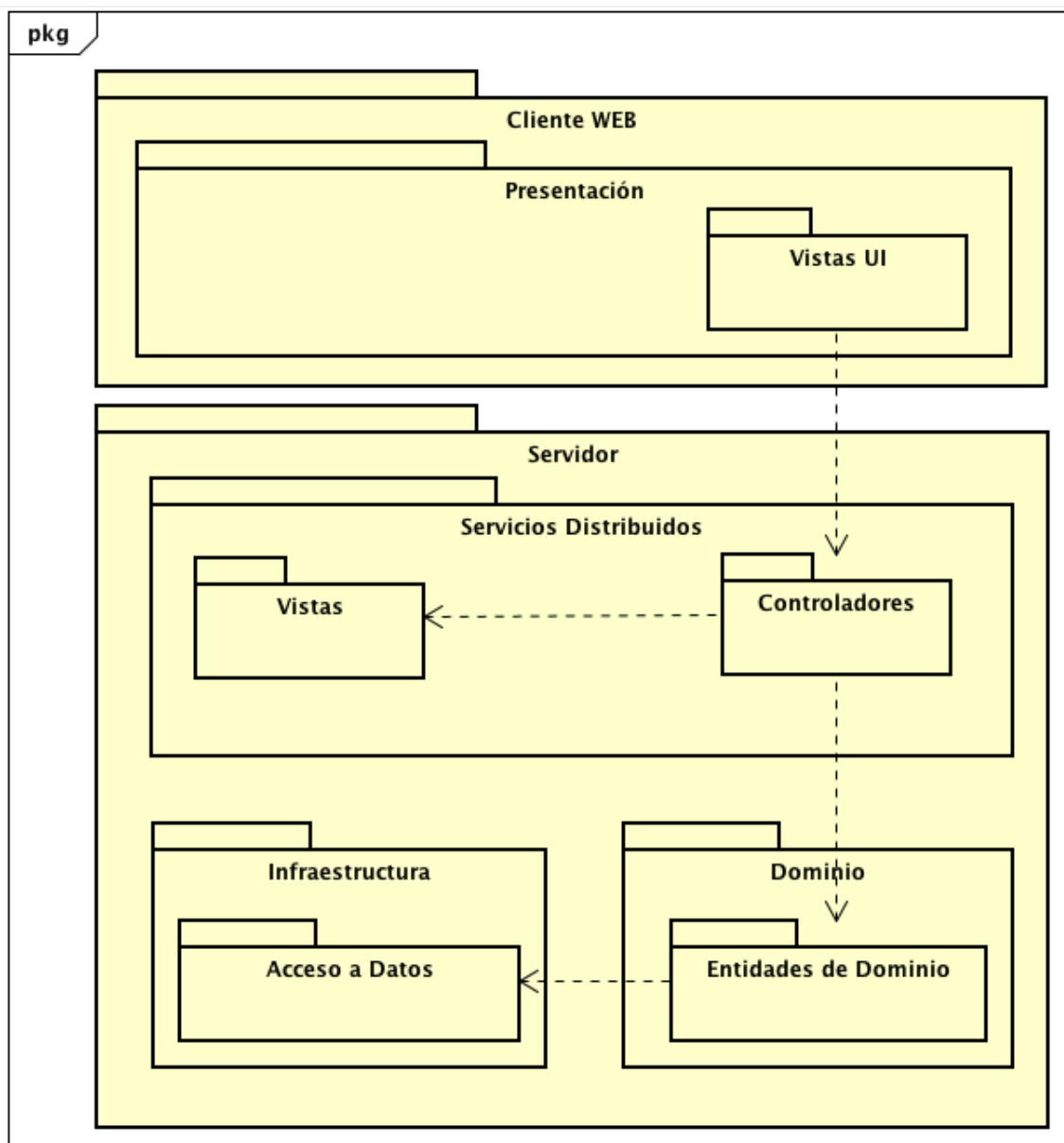


Figura 5-9: Diagrama de Paquetes.

5.1.3. Vista Física

En esta vista se ilustra la distribución física de los componentes del sistema LRS, transformando la vista lógica presentada en la sección anterior a componentes de software y de hardware que interactúan para presentar al usuario las funcionalidades esperadas.

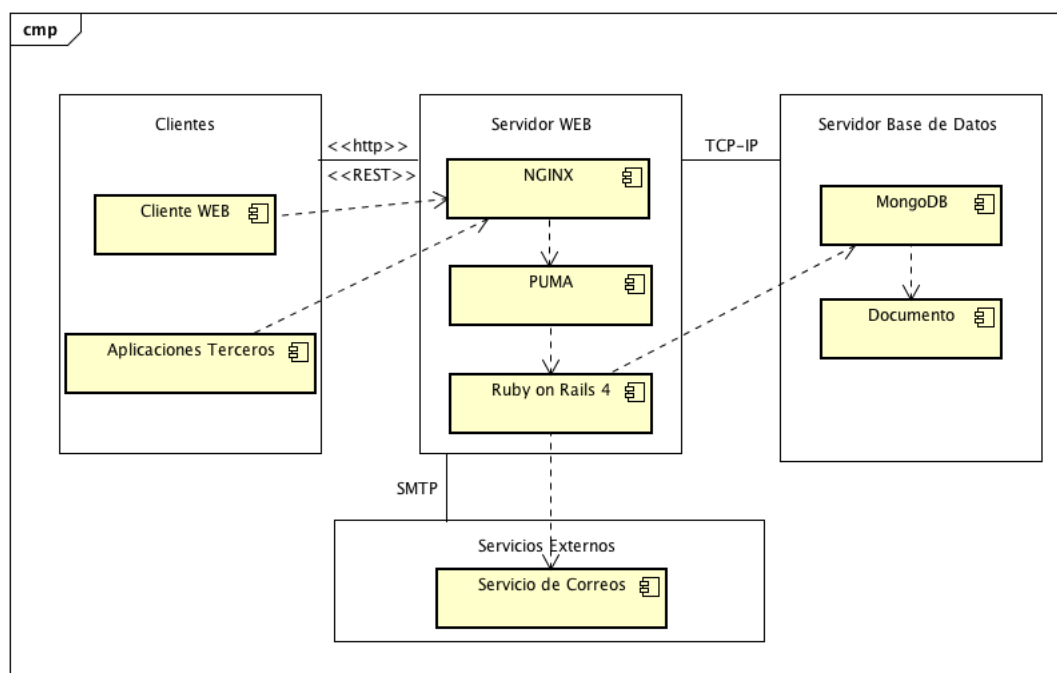


Figura 5-10: Vista Física.

Linux OS

Los servidores (Web y Base de Datos) tendrán instalados un sistema operativo basados en linux, por efectos de documentación y soporte, se recomienda utilizar la distribución Ubuntu Server en su versión más reciente.

Adicionalmente, Ubuntu Server¹ ofrece las siguientes ventajas:

- **Licenciamiento:** Dado a que Ubuntu esta licenciado bajo permisos de distribución libre, el costo de uso es nulo, por lo que nos da una gran ventaja en reducción de costo. [32]

¹<http://www.ubuntu.com/server>

- Soporte: Ubuntu es una de las distribuciones de linux que mayor soporte ofrece, gracias a su ciclo de lanzamientos cada 10 meses, permite obtener las últimas versiones de las librerías, junto con las actualizaciones de seguridad. Adicionalmente la comunidad de usuarios² es una de las más amplias, por lo que se puede encontrar solución a problemas rápidamente.
- Recursos hardware: Las distribuciones de linux son conocidas por su óptimo uso de los recursos de hardware, por lo que en general, pueden ejecutarse en maquinas más pequeñas³ que un servidor con Microsoft Windows⁴.
- Soporte de herramientas: Las herramientas utilizadas en el desarrollo del LRS, tienen soporte de ejecución en linux, y de igual manera, la comunidad de desarrolladores a adoptado Ubuntu como sistema operativo, por lo que se encuentra fácilmente documentación de configuración y solución de errores.

NGINX

Se selecciono NGINX⁵ el cual “es un servidor WEB de código abierto y de libre uso, ha tenido un gran crecimiento en los últimos años gracias a su gran rendimiento, estabilidad, simple configuración y bajo consumo de recursos de la maquina” [33]. NGINX es usado por el 24.9 % de aplicaciones de internet (Dato al 23 de Julio de 2015) y es el segundo más usado en sitios de alto tráfico (Ver figura 5-11).

Puma

Puma⁷ es un servidor de aplicaciones, el cual es diseñado para soportar aplicaciones escritas en el lenguaje Ruby⁸, esta “librería provee una implemen-

²<http://community.ubuntu.com>

³https://help.ubuntu.com/community/Installation/SystemRequirements#Ubuntu_Server_.28CLI.29_Installation

⁴<https://technet.microsoft.com/en-us/library/jj200132.aspx>

⁵<http://wiki.nginx.org/Main>

⁷<http://puma.io>

⁸<https://www.ruby-lang.org>

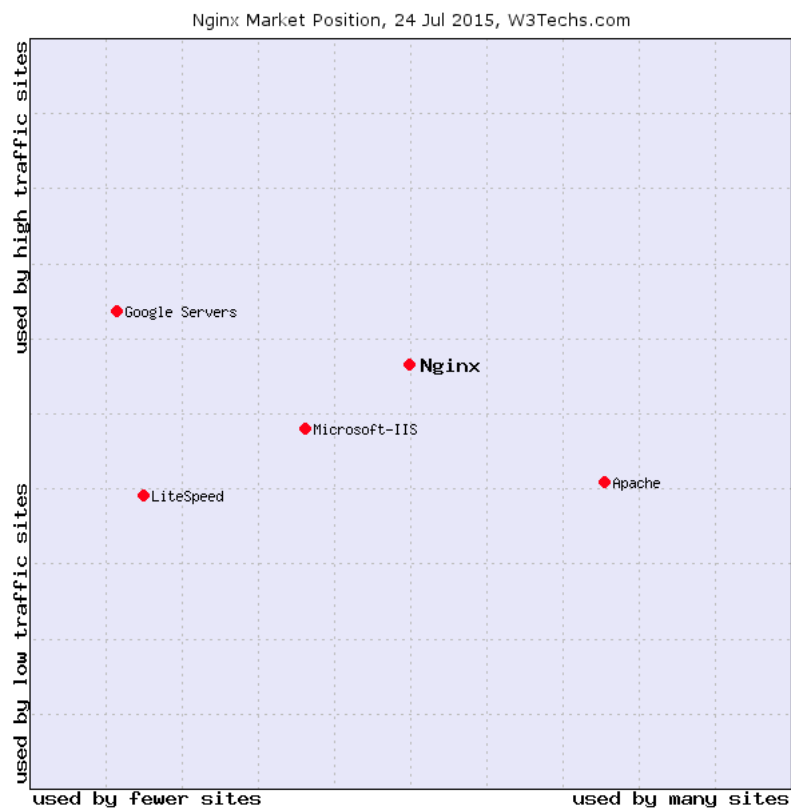


Figura 5-11: Gráfico de uso de NGINX.⁶

tación muy veloz y un servidor HTTP que atiende conexiones de manera concurrente” [34] con un bajo consumo de memoria del servidor.

Ruby on Rails

Ruby on Rails⁹ es un framework para desarrollar aplicaciones Web, el cual ofrece de manera nativa, solución a la mayor parte de los requerimientos plasmados en la arquitectura propuesta para la implementación del LRS en las secciones de vista conceptual y lógica. Rails, en su arquitectura (Ver figura ?? implementa los siguientes patrones:

- Modelo Vista Controlador
- Active Record + ORM (Object Relational Mapping)
- RESTFUL

⁹<http://rubyonrails.org>

- JSON API
- Envío de Correos

Ruby On Rails fue lanzado por primera vez en el año 2006, y desde entonces ha ganado una gran aceptación en la comunidad de desarrolladores, gracias a que el diseño adoptado por el framework de convención sobre configuración, el cual permite al desarrollador seguir algunas pautas, con el fin de automatizar gran parte del desarrollo del código, esto permite lograr desarrollar productos en un menor tiempo (comparado con otros frameworks). Aplicaciones en conocidas como Twitter ¹⁰, Groupon¹¹, GitHub¹², entre muchas otras, están desarrolladas con Ruby On Rails. Adicionalmente, hay muy buena cantidad de documentación y librerías que permiten agilizar el proceso de desarrollo de software.

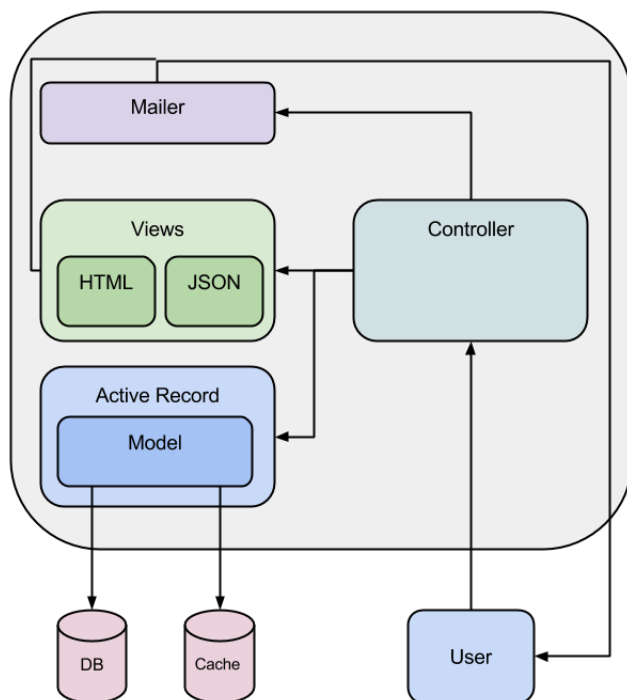


Figura 5-12: Vista lógica.

¹⁰<https://twitter.com>

¹¹<http://groupon.com/co/>

¹²<https://github.com>

MongoDB

MogoDB¹³ “es una base de datos no relacional de documentos, que provee un alto rendimiento, alta disponibilidad y escalamiento automático”[35].

Las bases de datos NoSQL o no relacionales fueron creadas para “manejar datos no estructurados como documentos, multimedia de manera eficiente. Las características de las bases de datos no relacional son:”[36]

- Alta escalabilidad.
- Alta confiabilidad.
- Modelo de datos muy simple.
- Lenguaje de consulta muy simple.

Se selecciono una base de datos NoSQL, ya que el modelo de datos que define el LRS es bastante simple, adicionalmente con el fin de soportar operaciones de inserción y consultas de datos de forma masiva, se requiere que la capa de acceso a datos tenga un tiempo de respuesta muy alto, y las comparativas en relación con las bases de datos relaciones muestran que MongoDB tiene un rendimiento mucho mayor, como fue evidenciado en el articulo “Comparing NoSQL MongoDB to an SQL DB”[37].

Adicionalmente, MongoDB es una base de datos de código abierto, la cual puede ser usada sin adquirir ningún tipo de licencia, lo cual permite obtener un ahorro en costos de desarrollo. La comunidad de desarrolladores ha implementado librerías para soportar la integración con múltiples lenguajes, y Ruby es uno de ellos, por lo que el uso con Ruby on Rails es transparente para el desarrollador.

¹³<https://www.mongodb.org>

5.2. Prototipo

Como parte de la tesis, se desarrollo un prototipo de un sistema LRS (Ver figura 5-13), basado en la arquitectura de software definida en la sección anterior y en las especificaciones de Tin Can API¹⁴, JSON API¹⁵, Activity Stream¹⁶, las cuales fueron detalladas en el marco teorico.

El prototipo esta publicado en el servicio Heroku, el cual es un servicio gratuito para hospedar aplicaciones realizadas con Ruby on Rails, para acceder a la plataforma se debe ingresar a la dirección web <http://eafit-lrs.herokuapp.com> con los siguientes datos:

- Correo: prueba@lrs.com
- Clave: 12345678

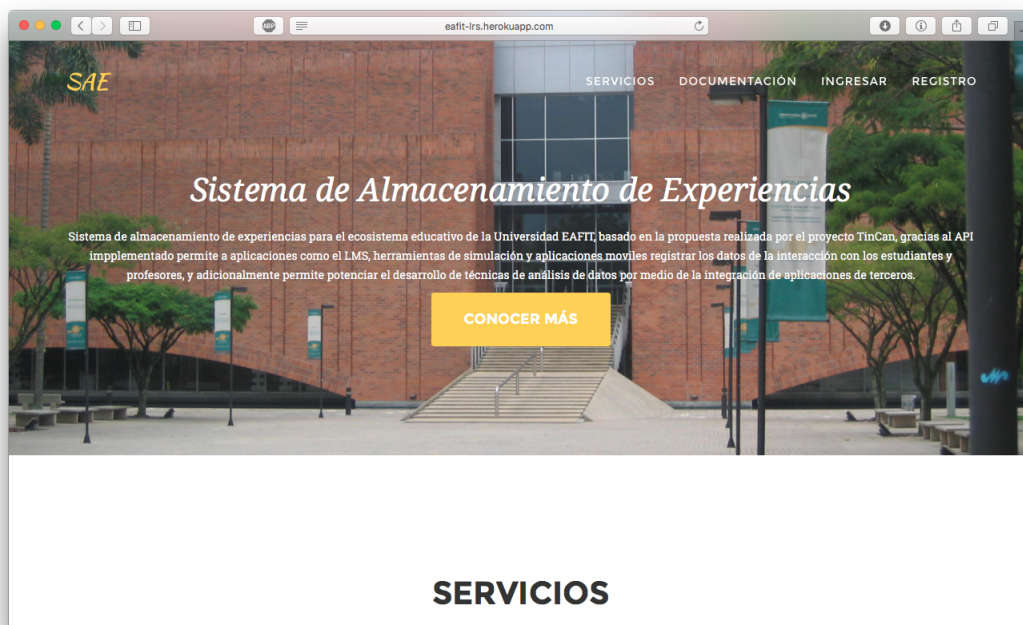


Figura 5-13: Pantalla de inicio.

¹⁴<http://tincanapi.com/>

¹⁵<http://jsonapi.org>

¹⁶<http://activitystrea.ms/>

5.2.1. Documentación

En la plataforma se tiene especificada la documentación (Ver figura 5-14)¹⁷ del API, esta incluye la descripción de cada endpoint (Punto de acceso a datos), el modelo de autenticación y algunos ejemplos de conexión al API en algunos lenguajes, sin embargo la integración con el API no está limitado a un lenguaje de programación en especial, dado a que la integración es dada por medio de peticiones HTTP.

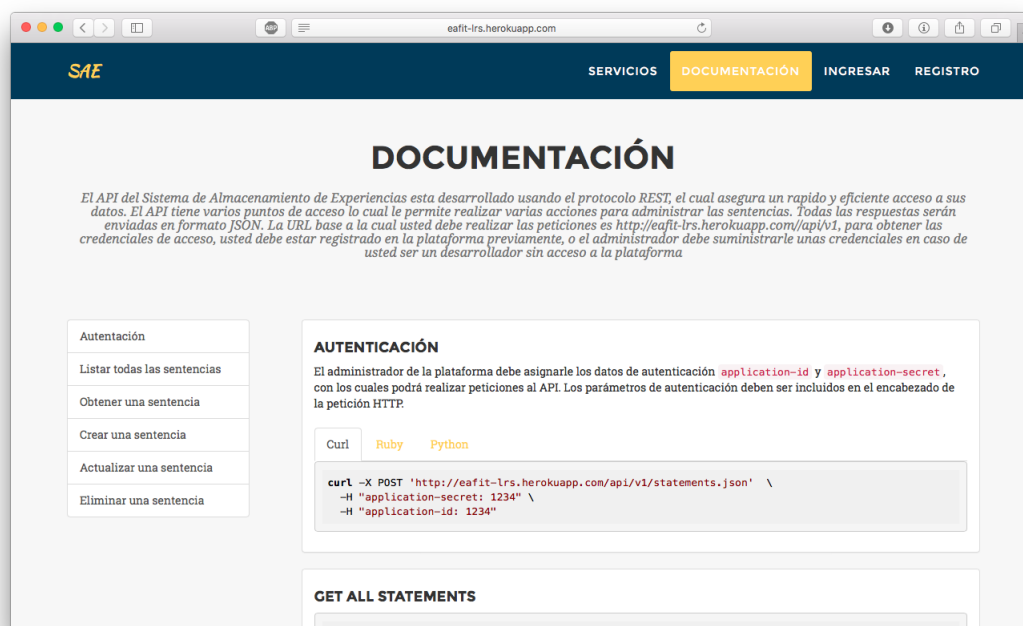


Figura 5-14: Documentación del API.

Endpoints

Los Endpoints son las URL que las aplicaciones externas deben llamar para poder realizar la integración con el API del LRS, estas le permitirán realizar todas las operaciones CRUD (Create, Read, Update and Delete) relacionadas con las sentencias, el siguiente listado detalla el verbo HTTP, la URL y una respuesta de ejemplo:

¹⁷<http://eafit-lrs.herokuapp.com/documentation>

- Listar todas las sentencias

```

1 GET http://eafit-lrs.herokuapp.com/api/v1/statements.json
2 {
3   "statements": [
4     {
5       "id": "55c1803a5275627958020000",
6       "url": "http://localhost:3000/api/v1/statements",
7       "actor": {
8         "mbox": "respinos@eafit.edu.co",
9         "name": "Ruben Espinosa",
10        "object_type": "Actor"
11      },
12      "verb": {
13        "uri": "http://api.verbs/done",
14        "display": {
15          "en": "done",
16          "es": "terminar"
17        }
18      },
19      "result": null,
20      "context": null,
21      "updated_at": "2015-08-05T03:17:14.270Z"
22    },
23    {
24      "id": "55c180ad5275627958050000",
25      "url": "http://localhost:3000/api/v1/statements",
26      "actor": {
27        "mbox": "jlalinde@eafit.edu.co",
28        "name": "Juan Lalinde",
29        "object_type": "Actor"
30      },
31      "verb": {
32        "uri": "http://api.verbs/start",
33        "display": {
34          "en": "start",
35          "es": "comenzar"
36        }
37      },
38      "result": null,
39      "context": null,
40      "updated_at": "2015-08-05T03:19:09.208Z"
41    }
42  ]
43 }

```

Listing 5.1: Listar todas las sentencias

- Detalle de una sentencia.

```
1 GET http://eafit-lrs.herokuapp.com/api/v1/statements/:id.json
2 {
3   "id": "55c1803a5275627958020000",
4   "actor": {
5     "_id": {
6       "$oid": "55c1803a5275627958000000"
7     },
8     "mbox": "respinos@eafit.edu.co",
9     "name": "Ruben Espinosa",
10    "object_type": "Actor"
11  },
12  "verb": {
13    "_id": {
14      "$oid": "55c1803a5275627958010000"
15    },
16    "display": {
17      "en": "done",
18      "es": "terminar"
19    },
20    "uri": "http://api.verbs/done"
21  },
22  "result": null,
23  "context": null,
24  "authority": null,
25  "created_at": "2015-08-05T03:17:14.270Z",
26  "updated_at": "2015-08-05T03:17:14.270Z"
27 }
```

Listing 5.2: Detalle de una sentencia.

■ Crear una sentencia.

```
1 POST http://eafit-lrs.herokuapp.com/api/v1/statements.json
2 {
3   "statement": {
4     "actor": {
5       "name": "Juan Lalinde",
6       "mbox": "jlalinde@eafit.edu.co",
7       "object_type": "Actor"
8     },
9     "verb": {
10      "uri": "http://api.verbs/start",
11      "display": {
12        "en": "start",
13        "es": "comenzar"
14      }
15    }
16  }
```

```
17 }
```

Listing 5.3: Crear una sentencia.

- Actualizar una sentencia.

```
1 PUT/PATCH http://eafit-lrs.herokuapp.com/api/v1/statements/:id.json
2 {
3   "statement": {
4     "result": "5.0"
5   }
6 }
```

Listing 5.4: Actualizar una sentencia.

- Eliminar una sentencia.

```
1 DELETE http://eafit-lrs.herokuapp.com/api/v1/statements/:id.json
```

Listing 5.5: Eliminar una sentencia.

5.2.2. Aplicaciones

El modulo de aplicaciones (Ver figura **5-15**), le permite a cada usuario obtener los datos para poder realizar la integración del API con sus aplicaciones privadas, una vez las aplicaciones son registradas, la plataforma otorgará dos parámetros, llamados el ***Application ID*** y ***Application Secret*** los cuales son requeridos en cada petición HTTP con el fin de relacionar cada sentencia con una aplicación.

Las aplicaciones son administradas según el rol de cada usuario basado la siguiente descripción:

- Administrador
 - Crear, actualizar, listar, eliminar y ver las credenciales sus propias aplicaciones.
 - Habilitar, deshabilitar, listar y eliminar las aplicaciones de los otros usuarios sin permiso de ver las credenciales.
- Usuario Normal

- Crear, actualizar, listar, eliminar y ver las credenciales sus propias aplicaciones.

Las aplicaciones tienen 2 estados, aprobado y declinado, con el fin de poder garantizar la integración con el API, la aplicación deberá tener al momento de las peticiones HTTP el estado aprobado.

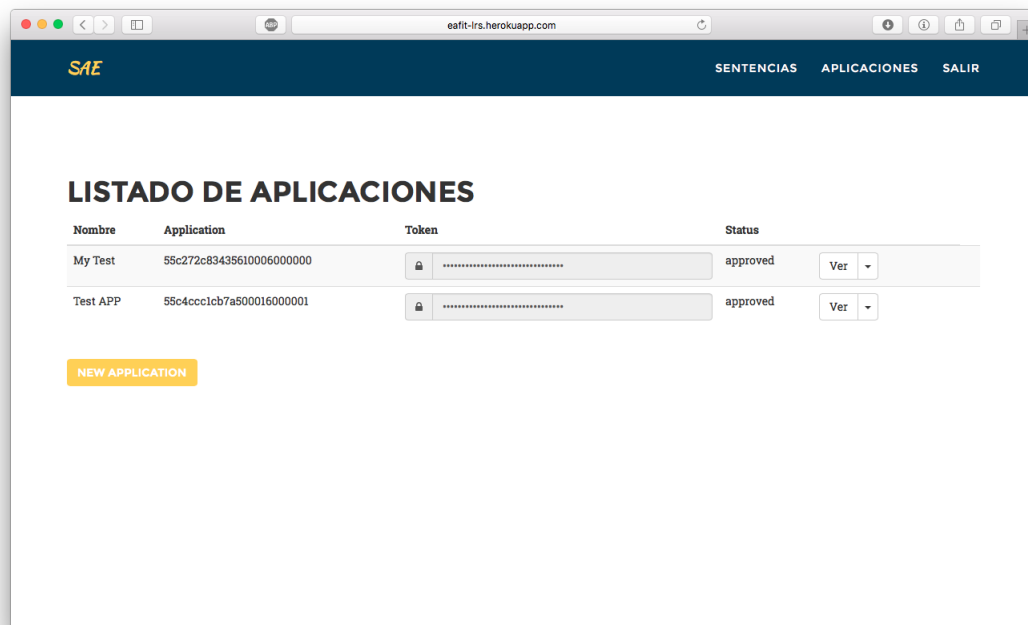


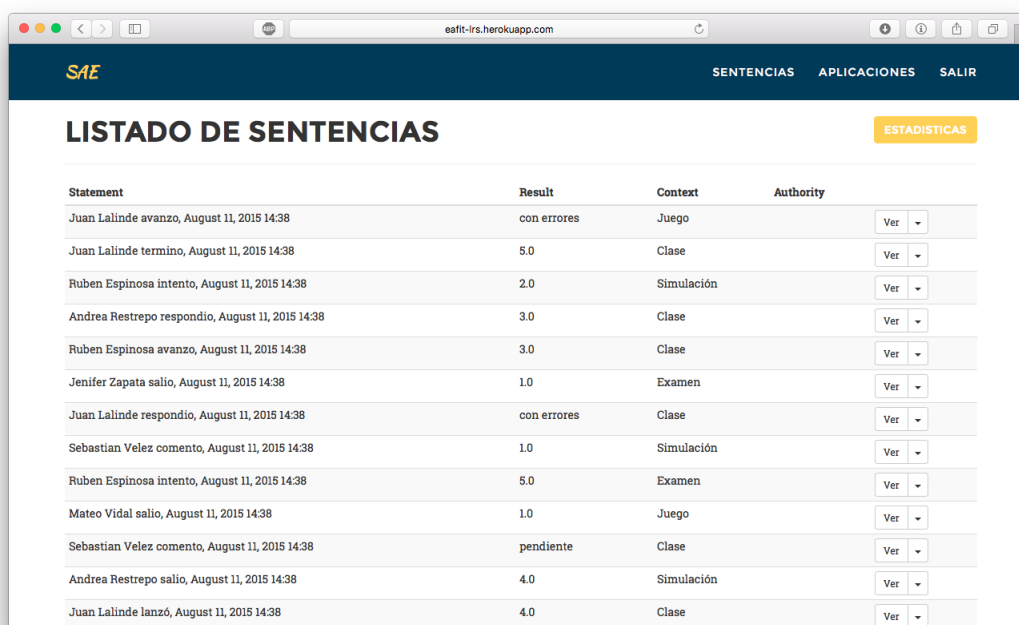
Figura 5-15: Administración de aplicaciones.

5.2.3. Sentencias

El modulo de sentencias (Ver figura 5-16) le permite a cada usuario ver las sentencias publicadas en el LRS por medio del API con sus aplicaciones privadas. Este modulo tiene implementado las funciones de eliminar y ver el detalle de una sentencia en el formato JSON.

Las sentencias son desplegadas según el rol de cada usuario basado la siguiente descripción:

- Administrador



Statement	Result	Context	Authority
Juan Lalinde avanzo, August 11, 2015 14:38	con errores	Juego	Ver
Juan Lalinde termino, August 11, 2015 14:38	5.0	Clase	Ver
Ruben Espinosa intento, August 11, 2015 14:38	2.0	Simulación	Ver
Andrea Restrepo respondio, August 11, 2015 14:38	3.0	Clase	Ver
Ruben Espinosa avanzo, August 11, 2015 14:38	3.0	Clase	Ver
Jenifer Zapata salio, August 11, 2015 14:38	1.0	Examen	Ver
Juan Lalinde respondio, August 11, 2015 14:38	con errores	Clase	Ver
Sebastian Velez comento, August 11, 2015 14:38	1.0	Simulación	Ver
Ruben Espinosa intento, August 11, 2015 14:38	5.0	Examen	Ver
Mateo Vidal salio, August 11, 2015 14:38	1.0	Juego	Ver
Sebastian Velez comento, August 11, 2015 14:38	pendiente	Clase	Ver
Andrea Restrepo salio, August 11, 2015 14:38	4.0	Simulación	Ver
Juan Lalinde lanzo, August 11, 2015 14:38	4.0	Clase	Ver

Figura 5-16: Listado de sentencias.

- Listar, eliminar y ver el detalle de las sentencias publicadas por aplicaciones propias.
 - Listar y ver el detalle de las sentencias publicadas por aplicaciones de otros usuarios.
- Usuario Normal
 - Listar, eliminar y ver el detalle de las sentencias publicadas por aplicaciones propias.

5.2.4. Estadísticas

El modulo de estadísticas (Ver figura 5-17) le permite a cada usuario obtener algunas estadísticas básicas relacionadas con el uso de la plataforma, las cuales son presentadas de manera amigable al usuario por medio de gráficos interactivos.

Las gráficas son implementadas con la librería javascript HighCharts.js¹⁸, la

¹⁸<http://highcharts.com/>

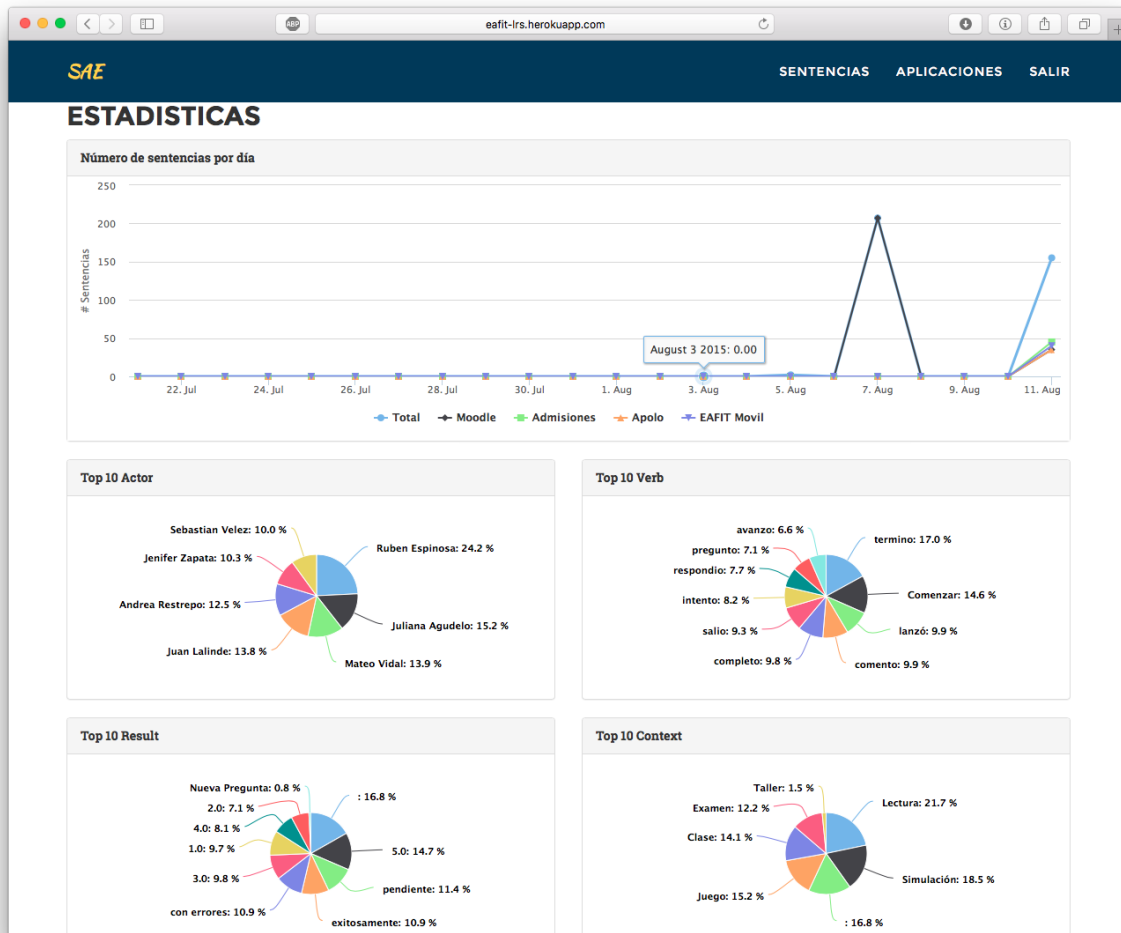


Figura 5-17: Estadísticas

cual es de uso gratuito para proyectos sin animo de lucro. Los datos que alimentan las estadísticas provienen de dos cálculos principalmente

1. Número de sentencias por día: Este obtiene un calculo simple, donde se cuentan el número de sentencias publicadas en la plataforma de manera global en la serie de datos llamada *Total*, el gráfico incluye también el conteo desagregado de sentencias publicadas por aplicación.
2. Los gráficos de tendencias, los cuales están reflejados en 4 gráficos, los cuales representan el top 10 de tendencias en los datos de actores, verbos, resultados y contextos, son construidos basado en la librería Popular

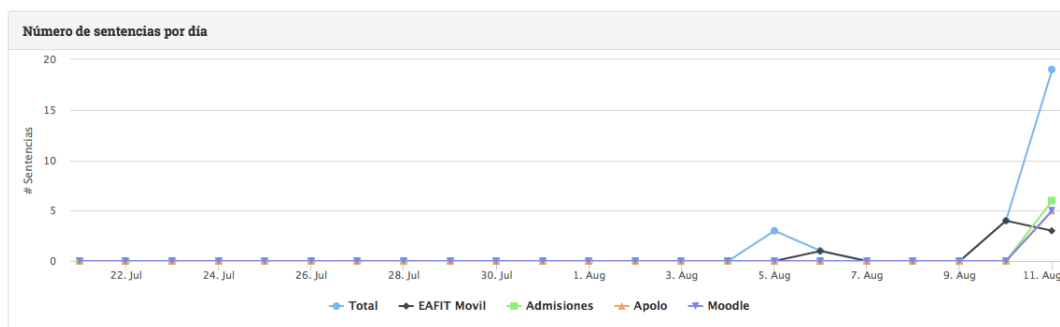


Figura 5-18: Gráfico de sentencias publicadas por día.

Streams¹⁹. Esta librería permite resolver el problema de saber que es tendencia en este momento, “dado que el problema de las tendencias es que decaen con el tiempo. Esto significa que algo que era popular hace una semana con 10 votos es menos popular que algo que es votado 6 veces el día de hoy” [38] Gracias a esta librería se obtendrán estadísticas de tendencia al día de consulta, teniendo en cuenta el peso de aparición dado cada día.

Las gráficas son desplegadas según el rol de cada usuario basado la siguiente descripción:

- Administrador
 - Número de sentencias publicadas totales y por aplicación, de las aplicaciones propias.
 - Número de sentencias publicadas totales y por aplicación, de las aplicaciones registradas en toda la plataforma.
 - Top 10 de actores, verbos, resultados y contextos en el momento de consulta, de las aplicaciones propias.
 - Top 10 de actores, verbos, resultados y contextos en el momento de consulta, de las aplicaciones registradas en toda la plataforma.

- Usuario Normal

¹⁹https://github.com/nhocki/popular_streams

- Número de sentencias publicadas totales y por aplicación, de las aplicaciones propias.
- Top 10 de actores, verbos, resultados y contextos en el momento de consulta, de las aplicaciones propias.

5.3. Ejemplo de un cliente para el API

```

1 require 'httparty'
2 class Publish
3
4   # Realiza la petici n HTTP por medio del metodo POST, obtiene los datos
5   # de autenticaci n de manera aleatoria del listado de aplicaciones
6   # Ejemplo
7   # p = Publish.new.post
8   def post
9     url = 'http://eafit-lrs.herokuapp.com/api/v1/statements'
10    HTTParty.post(url, headers: applications.sample, body: build_statement)
11  end
12
13  # Construye la sentencias tomando datos aleatorios de actor, verbo,
14  # y contexto
15  # Ejemplo
16  # Publish.new.build_statement
17  # => {
18  #   :statement=>{
19  #     :actor=> {
20  #       :name=>"Sebastian Velez",
21  #       :mbox=>"jvelezpo@eafit.edu.co",
22  #       :object_type=>"Actor"
23  #     },
24  #     :verb=>{
25  #       :uri=>"http://adlnet.gov/expapi/verbs/progressed",
26  #       :display=>{
27  #         :en=>"progressed",
28  #         :es=>"avanzo"
29  #       }
30  #     },
31  #     :result=>"con errores",
32  #     :context=>"Clase"
33  #   }
34  # }
35  def build_statement
36    {
37      statement:

```

```
38         {actor: actors.sample,
39           verb: verbs.sample,
40           result: results.sample,
41           context: contexts.sample
42         }
43     }
44 end
45
46 # Listado de posibles verbos con sus respectivos atributos de uri y display
47 def verbs
48     [
49         {uri: 'http://adlnet.gov/expapi/verbs/launched', display: { en: '
50 launched', es: 'lanz ' }},
51         {uri: 'http://adlnet.gov/expapi/verbs/attempted', display: { en: '
52 attempted', es: 'intento' }},
53         {uri: 'http://adlnet.gov/expapi/verbs/completed', display: { en: '
54 completed', es: 'completo' }},
55         {uri: 'http://adlnet.gov/expapi/verbs/finished', display: { en: '
56 finished', es: 'termino' }},
57         {uri: 'http://adlnet.gov/expapi/verbs/commented', display: { en: '
58 commented', es: 'comento' }},
59         {uri: 'http://adlnet.gov/expapi/verbs/exited', display: { en: 'exited'
60 , es: 'salio' }},
61         {uri: 'http://adlnet.gov/expapi/verbs/progressed', display: { en: '
62 progressed', es: 'avanzo' }},
63         {uri: 'http://adlnet.gov/expapi/verbs/answered', display: { en: '
64 answered', es: 'respondio' }},
65         {uri: 'http://adlnet.gov/expapi/verbs/asked', display: { en: 'asked',
66 es: 'pregunto' }}
67     ]
68 end
69
70 # Listado de posibles resultados
71 def results
72     [
73         'exitosamente',
74         'con errores',
75         '1.0',
76         '5.0',
77         'pendiente',
78         '3.0',
79         '2.0',
80         '4.0'
81     ]
82 end
83
84 # Listado de posibles contextos
85 def contexts
```

```
77     [
78         'Examen',
79         'Simulaci n',
80         'Clase',
81         'Juego',
82         'Lectura'
83     ]
84 end
85
86 # Listado de posibles actores con sus respectivos atributos de name, mbox y
87   object_type
88 def actors
89     [
90         {name: 'Ruben Espinosa', mbox: 'respinos@eafit.edu.co', object_type: '
91         Actor'},
92         {name: 'Mateo Vidal', mbox: 'jvidalba@eafit.edu.co', object_type: '
93         Actor'},
94         {name: 'Sebastian Velez', mbox: 'jvelezpo@eafit.edu.co', object_type:
95         'Actor'},
96         {name: 'Juan Lalinde', mbox: 'jlalinde@eafit.edu.co', object_type: '
97         Actor'},
98         {name: 'Jenifer Zapata', mbox: 'jzapata@eafit.edu.co', object_type: '
99         Actor'},
100        {name: 'Andrea Restrepo', mbox: 'arestrepo@eafit.edu.co', object_type:
101        'Actor'},
102        {name: 'Juliana Agudelo', mbox: 'jagudelo@eafit.edu.co', object_type:
103        'Actor'}],
104     ]
105 end
106
107 # Listado de posibles aplicaciones, estos datos corresponden a las
108 # credenciales obtenidas en la plataforma de registro de experiencias LRS
109 def applications
110     [
111         {'application_id' => '55c272c8343561000600000', 'application_secret'
112         => '82df40caee8844d6ba184d90e931d43e'},
113         {'application_id' => '55c4ccc1cb7a500016000001', 'application_secret'
114         => '956638da73d64f70be5b8b8857a2e3e3'},
115         {'application_id' => '55ca062421ba410016000000', 'application_secret'
116         => '6f74241c8b8d4b08b10b1808e3a9fb2a'},
117         {'application_id' => '55ca068721ba410016000001', 'application_secret'
118         => 'f5a1dbbd2ed9418db85cb8ca0cb7068b'}
119     ]
120 end
121 end
```

Listing 5.6: Ejemplo de un cliente.

6 Conclusiones y Trabajo a Futuro

En este capítulo se da una presentación de las conclusiones a las que se llega luego de la realización de esta tesis, en la cual se realiza la inclusión de un sistema de almacenamiento de experiencias LRS, mediante la propuesta desde el modelo teórico, el planteamiento desde la arquitectura de software y el desarrollo del prototipo software. Adicional a esto, se dan algunas posibilidades en líneas de trabajo que pueden ser exploradas luego del planteamiento realizado en esta tesis.

6.1. Conclusiones

La adopción del estándar xAPI desarrollado en el proyecto Tin Can por la organización ADL, permite hacer un seguimiento detallado de como los actores del ecosistema educativo están interactuando con el mismo. Además de esto, el formato definido para las sentencias, el cual esta dado uso de lenguaje sencillo Actor Verbo Objeto, permite que sea fácilmente entendible por humanos y maquinas, facilitando así el intercambio de mensajes entre plataformas del ecosistema educativo, y permite potenciar el desarrollo de plataformas nuevas en el futuro cercano. Algunas conclusiones a las que se llega con la realización de este trabajo son:

- El sistema de almacenamiento de experiencias puede ser usando en otros entornos diferentes al educativo, dado su diseño iteroperable y desacoplado, puede capturar información de cualquier aplicación que este en capacidad de integrarse con el.
- El diseño de la plataforma, enfocado en el estándar planteado en el planteamiento arquitectónico SOA, permite realizar la integración de

sistemas heterogéneos.

- La tendencia en el desarrollo de software en el sector educativo desde finales de los años 90, en el cual se ha buscado la estandarización de protocolos con el fin de compartir información, ha permitido que al día de hoy, cuando emergen nuevas tendencias y tecnologías, estas puedan ser adaptadas al modelo educativo con facilidad, y así poder hacer un mejor seguimiento a la interacción entre los actores, con el fin de mejorar la calidad en la educación.
- El uso de arquitecturas basadas en SOA, permite desarrollar módulos que no dependen tecnológicamente de los otros, por lo que permite a las organizaciones el uso de tecnologías emergentes, como es el caso de las bases de datos no relaciones, y el software open source, los cuales fueron usados ampliamente en el desarrollo de esta tesis.

6.2. Trabajo a Futuro

Dado este trabajo propone la inclusión de un componente del ecosistema de aplicaciones educativas, es importante que los otros componentes del ecosistema puedan interactuar con el repositorio de experiencias LRS, por este motivo se pueden derivar trabajos de investigación y de implementación en los siguientes temas:

- **Definición de verbos:** La organización ADL propone la inclusión de unos verbos para la definición de unos verbos¹, sin embargo, es un tema importante a desarrollar, dado la naturaleza del ecosistema educativo de la Universidad EAFIT, una definición de los verbos que aplican para el contexto en particular. Adicionalmente se debe proveer acceso a estos verbos por medio de un API, con la cual se pueda hacer referencia por medio de una URL única.
- **Definición de actividades:** De igual manera que en los verbos, la orga-

¹<http://adlnet.gov/expapi/verbs>

nización ADL propone una definición de actividades² validas que puede desarrollar un actor en el ecosistema de las aplicaciones educativas, sin embargo es pertinente evaluar que tipo de actividades aplican y cuales nuevas de deben incluir. Adicionalmente se debe proveer acceso a estas actividades por medio de un API, con la cual se pueda hacer referencia por medio de una URL única.

- **Cliente y plugin para captura de datos:** El desarrollo de esta tesis, solo se compone del repositorio de experiencias, por lo tanto otras plataformas el ecosistema de aplicaciones educativas deben encargarse de publicar las sentencias en el repositorio de experiencias. Con este fin, se debe proponer e implementar plugins para las plataformas LMS entre otros.
- **Plataforma de análisis de datos:** Uno de los objetivos que se logra con el desarrollo de esta tesis es el de permitir técnicas de análisis de datos, específicamente, el análisis de datos educativos, esto se logra a través de la publicación de los datos a través del API REST, ahora, se debe desarrollar una plataforma que haga parte del ecosistema de aplicaciones educativas, la cual debe encargarse de obtener los datos y realizar análisis de los mismos con el fin de hacer predicciones y presentar tendencias, las cuales permitan mejorar la calidad de los contenidos, las técnicas de enseñanza, entre otros aspectos.

²<http://adlnet.gov/expapi/activityTypes/index.html>

Bibliografía

- [1] C. M. Zea, M. del Rosario Atuesta, J. G. Lalinde-Pulido, and C. Vieira, ““modelo tag (tecnología aprendizaje gestión) para la valoración de ubicuidad en instituciones de educación superior,” Master’s thesis, Universidad EAFIT, 2013.
- [2] E. A. Ceballos, ““patrón arquitectónico de software para portales educativos con atributos de interoperabilidad,” Master’s thesis, Universidad EAFIT, 2015.
- [3] Alcaldía de Medellín, “Medellín ciudad inteligente.” <http://www.mdeinteligente.co/>, 2015. Accedido: 13/05/2015.
- [4] J. Haag, “Experience api (xapi) and the future of scorm.” <http://www.slideshare.net/jhaag75/experience-api-xapi-and-the-future-of-scorm>, 2015. Accedido: 10/05/2015.
- [5] IMS GLB, “Introducing learning tools interoperability.” <http://www.imslobal.org/toolsinteroperability2.cfm>, 2015. Accedido: 13/05/2015.
- [6] J. G. Lalinde-Pulido, C. M. Zea, L. F. Londoño, N. Hock, and S. A. Monsalve, “Portal inteligente medellín documentación de la arquitectura de software - introducción,” p. 21, 2013. Version 1.1.
- [7] Advanced Distributed Learning (ADL), “The training and learning architecture: Infrastructure for the future of learning.” <http://es.slideshare.net/damonregan/>

- regan-tla-infrastructureforfutureoflearningsintice2013, 2013. Accedido: 13/01/2015.
- [8] L. M. C. de León, “Interoperabilidad; estándares,” *Revista Digital Universitaria*, vol. 5, no. 10, p. 4, 2004.
- [9] L. M. C. de León, “Interoperabilidad; estándares,” *Revista Digital Universitaria*, vol. 5, no. 10, p. 5, 2004.
- [10] Universidad EAFIT, “Proyecto 50 presentación.” <http://www.eafit.edu.co/proyecto50/p50/Paginas/presentacion.aspx>, 2015. Accedido: 13/01/2015.
- [11] Advanced Distributed Learning (ADL), “Overview.” <http://www.adlnet.gov/overview.html>, 2015. Accedido: 13/01/2015.
- [12] Advanced Distributed Learning (ADL), “History.” <http://www.adlnet.gov/overview.html>, 2015. Accedido: 13/01/2015.
- [13] D. R. Branon, J. Nelson, J. Poltrack, and Problem Solutions, “Scorm update.” http://www.adlnet.gov/wp-content/uploads/2011/08/poltrack_branon_nelson_SCORM_update_iFest2011.pdf, 2011. Accedido: 13/01/2015.
- [14] Advanced Distributed Learning (ADL), “Sharable content object reference model (scorm®) 2004 4th edition content aggregation model (cam),” tech. rep., 2009. Version 1.1.
- [15] Advanced Distributed Learning (ADL), “Sharable content object reference model (scorm®) 2004 4th edition run-time environment (rte),” tech. rep., 2009. Version 1.1.
- [16] Advanced Distributed Learning (ADL), “Sharable content object reference model (scorm®) 2004 4th edition sequencing and navigation (sn),” tech. rep., 2009. Version 1.1.
- [17] IMS GLB, “About ims global learning consortium.” <http://www.imslobal.org/background.html>, 2015. Accedido: 15/05/2015.

-
- [18] IMS., “Learning tools interoperability.” <http://www.imsglobal.org/toolsinteroperability2.cfm>, 2015. Accedido: 13/01/2015.
- [19] M. Rustici, “Project tin can.” <http://www.slideshare.net/mikerustici/project-tin-can>, 2015. Accedido: 13/05/2015.
- [20] W3C® (MIT, ERCIM, Keio, Beihang), “Activity streams 2.0 - introduction,” tech. rep., 2015. W3C Working Draft.
- [21] Alcaldía de Medellín, “Medellín ciudad inteligente - preguntas frecuentes.” <http://www.mdeinteligente.co/estrategia/faq-3/>, 2015. Accedido: 13/05/2015.
- [22] J. G. Lalinde-Pulido, C. M. Zea, L. F. Londoño, N. Hock, and S. A. Monsalve, “Portal inteligente medellín documentación de la arquitectura de software - resumen,” p. 2, 2013. Version 1.1.
- [23] J. G. Lalinde-Pulido, C. M. Zea, L. F. Londoño, N. Hock, and S. A. Monsalve, “Portal inteligente medellín documentación de la arquitectura de software - repositorio de experiencias (tin can api),” p. 34, 2013. Version 1.1.
- [24] Moodle Org., “Moodle history.” <https://docs.moodle.org/28/en/History>, 2015. Accedido: 13/01/2015.
- [25] M. Dougiamas, “Improving the effectiveness of tools for internet based education,” in *In A. Herrmann and M.M. Kulski (Eds), Flexible Futures in Tertiary Teaching. Proceedings of the 9th Annual Teaching Learning Forum, 2-4 February 2000. Perth: Curtin University of Technology., 2000.*
- [26] IEEE, “Learning technology standards committee. draft standard for learning object metadata,” tech. rep., IEEE, 2002.
- [27] SCORM., “Scorm explained.” <http://scorm.com/scorm-explained/>, 2015. Accedido: 13/01/2015.

-
- [28] Advanced Distributed Learning (ADL), “What is the tin can api?” <http://tincanapi.com/overview/>, 2015. Accedido: 13/01/2015.
- [29] A. Lasso, “Arquitectura de software.” http://eisc.univalle.edu.co/materias/Material_Desarrollo_Software/Arquitectura%20de%20Software.htm. Accedido: 13/07/2015.
- [30] Microsoft, “Model-view-controller.” <https://msdn.microsoft.com/en-us/library/ff649643.aspx>. Accedido: 13/07/2015.
- [31] M. Fowler, *Patterns of enterprise application architecture*. Addison-Wesley.
- [32] Ubuntu, “Licensing.” <http://www.ubuntu.com/about/about-ubuntu/licensing>. Accedido: 13/07/2015.
- [33] NGINX, “Overview.” <http://wiki.nginx.org/Main>. Accedido: 13/07/2015.
- [34] Puma.io, “Overview.” <http://puma.io>. Accedido: 13/07/2015.
- [35] Mongo.org, “The mongodb 3.0 manual.” <http://docs.mongodb.org/manual/>. Accedido: 13/07/2015.
- [36] C. J. M. Tauro, A. S, and S. A.B, “Comparative study of the new generation, agile, scalable, high performance nosql databases,” *International Journal of Computer Applications*, vol. 48, no. 20, p. 4, 2012.
- [37] Z. Parker, S. Poe, and S. V. Vrbsky, “Comparing nosql mongodb to an sql db,” p. 6.
- [38] N. Hock, “Trending content with ruby and redis.” <http://blog.nhocki.com/2014/12/02/trending-content-with-ruby-and-redis/>. Accedido: 13/07/2015.