|  |  |
| --- | --- |
|  | UNIVERSIDAD VERACRUZANA  FACULTAD DE ESTADÍSTICA E INFORMÁTICA |

TRABAJO RECEPCIONAL:

PLATAFORMA DE SOPORTE PARA LABORATORIOS VIRTUALES UTILIZANDO UN ESTÁNDAR PARA EL MANEJO DE OBJETOS DE APRENDIZAJE Y UN LMS.

MODALIDAD:

TRABAJO PRÁCTICO-TÉCNICO

COMO REQUISITO PARCIAL PARA OBTENER EL GRADO DE:

LICENCIADO EN INGENIERÍA DE SOFTWARE

PRESENTA:

VÍCTOR JAVIER GARCÍA MASCAREÑAS

DIRECTORES:

M.C.C. GERARDO CONTRERAS VEGA

M.C.C. JUAN CARLOS PÉREZ ARRIAGA

XALAPA, VER. OCTUBRE 2019

Índice

[Definición del problema. 6](#_Toc22223146)

[Objetivos. 6](#_Toc22223147)

[Justificación. 6](#_Toc22223148)

[Alcances y limitaciones. 6](#_Toc22223149)

[Capítulo 1: Marco teórico. 8](#_Toc22223150)

[1.1. Sistemas de gestión de cursos. 9](#_Toc22223151)

[1.1.1. Características de los sistemas de gestión de cursos. 11](#_Toc22223152)

[1.1.2. Empleo de sistemas de gestión de cursos. 12](#_Toc22223153)

[1.1.3. Comparativa de sistemas de gestión de cursos 12](#_Toc22223154)

[1.2. Laboratorios virtuales. 15](#_Toc22223155)

[1.2.1. Realidad virtual. 17](#_Toc22223156)

[1.2.2. Consideraciones para el diseño de laboratorios virtuales. 18](#_Toc22223157)

[1.2.3. Características y empleo de laboratorios virtuales. 18](#_Toc22223158)

[1.3. Estándares para el empaquetado, manejo y distribución de objetos de aprendizaje. 22](#_Toc22223159)

[1.3.1. SCORM 22](#_Toc22223160)

[1.3.2. Experience API. 28](#_Toc22223161)

[1.4. ICONIX: Metodología de desarrollo de software. 30](#_Toc22223162)

[Capítulo 2: Requerimientos de software. 33](#_Toc22223163)

[2.1. Introducción 34](#_Toc22223164)

[2.1.1. Propósito 34](#_Toc22223165)

[2.1.2. Ámbito del sistema 34](#_Toc22223166)

[2.2. Descripción general 35](#_Toc22223167)

[2.2.1. Perspectiva del producto 35](#_Toc22223168)

[2.2.2. Funciones del producto 36](#_Toc22223169)

[2.2.3. Características de los usuarios 37](#_Toc22223170)

[2.2.4. Restricciones 37](#_Toc22223171)

[2.3. Requisitos específicos 37](#_Toc22223172)

[2.3.1. Interfaces externas 37](#_Toc22223173)

[2.3.2. Funciones 38](#_Toc22223174)

[Capítulo 3: Análisis de requerimientos. 41](#_Toc22223175)

[3.1. Prototipos de la interfaz de usuario. 42](#_Toc22223176)

[3.2. Modelos del análisis de requerimientos. 45](#_Toc22223177)

[Capítulo 4: Diseño preliminar. 48](#_Toc22223178)

[Capítulo 5: Diseño detallado. 53](#_Toc22223179)

[5.1. Diagrama de clases. 54](#_Toc22223180)

[5.2. Diagramas de secuencia. 54](#_Toc22223181)

[Capítulo 6: Despliegue. 63](#_Toc22223182)

[6.1. Diagrama de despliegue. 64](#_Toc22223183)

[6.2. Pruebas. 64](#_Toc22223184)

[6.3. Codificación. 72](#_Toc22223185)

[6.4. Resultados de la ejecución de pruebas. 72](#_Toc22223186)

[Conclusiones y trabajo futuro. 73](#_Toc22223187)

[Bibliografía. 74](#_Toc22223188)

Índice de ilustraciones

Ilustración 1: Diagrama conceptual del paquete de contenido IMS. 24

Ilustración 2: Transiciones de estado del adaptador de API. 26

Ilustración 3: Estructura de ICONIX. 31

Ilustración 4: Diagrama de contexto. 36

Ilustración 5: Pantalla de consulta de laboratorios por parte del administrador. 42

Ilustración 6: Pantalla de consulta de laboratorios por parte de un profesor. 43

Ilustración 7: Pantalla de consulta de estadísticas de grupo. 43

Ilustración 8: Pantalla de consulta de estadísticas de un alumno por parte de un profesor. 44

Ilustración 9: Pantalla de consulta de estadísticas personales por parte de un alumno. 45

Ilustración 10: Modelo de dominio. 46

Ilustración 11: Modelo de casos de uso. 47

Ilustración 12: Diagrama de robustez CU 01 – Consultar laboratorios propios. 49

Ilustración 13: Diagrama de robustez CU 02 - Eliminar laboratorio. 50

Ilustración 14: Diagrama de robustez CU 03 - Consultar avances por alumno. 50

Ilustración 15: Diagrama de robustez CU 04 - Consultar avances por grupo. 51

Ilustración 16: Diagrama de robustez CU 05 - Consultar laboratorios. 51

Ilustración 17: Diagrama de robustez CU 06 - Consultar avances personales. 52

Ilustración 18: Diagrama de clases. 54

Ilustración 19: Diagrama de secuencia CU 01 - Consultar laboratorios propios. 55

Ilustración 20: Diagrama de secuencia CU 02 - Eliminar laboratorio. 56

Ilustración 21: Diagrama de secuencia CU 03 - Consultar avances por alumno. 57

Ilustración 22: Diagrama de secuencia CU 04 - Consultar avances por grupo. 59

Ilustración 23: Diagrama de secuencia CU 05 - Consultar laboratorios. 60

Ilustración 24: Diagrama de secuencia CU 06 - Consultar avances personales. 61

Ilustración 25: Diagrama de despliegue. 64

Índice de tablas

Tabla 1: Comparación de LMS Open source. 12

Tabla 2: Resultados de la evaluación de la calidad de LMS. 13

Tabla 3: Categorías de metadatos del modelo de agregación de contenido. 22

Tabla 4: Funciones del API de SCORM. 24

Tabla 5: Elementos del modelo de datos de SCORM (cmi.core) para evaluación. 25

Tabla 6: Elementos del modelo de datos de SCORM (cmi.comments) para evaluación. 26

Tabla 7: Elementos del modelo de datos de SCORM (cmi.objectives) para evaluación. 26

Tabla 8: Elementos del modelo de datos de SCORM (cmi.student\_data) para evaluación. 27

Tabla 9: Elementos del modelo de datos de SCORM (cmi.interactions) para evaluación. 27

Tabla 10: Elementos de la declaración de actividad de xAPI. 28

Tabla 11: CU 01 - Consultar laboratorios propios. 37

Tabla 12: CU 04 - Consultar avances por grupo. 37

Tabla 13: CU 03 - Consultar avances por alumno. 38

Tabla 14: CU 02 - Eliminar laboratorio. 38

Tabla 15: CU 05 - Consultar laboratorios. 39

Tabla 16: CU 06 - Consultar avances personales. 39

Tabla 17: Definición de casos de prueba unitaria. 63

Tabla 18: Caso de prueba unitaria CPU001. 66

Tabla 19: Caso de prueba unitaria CPU002. 66

Tabla 20: Caso de prueba unitaria CPU003. 66

Tabla 21: Caso de prueba unitaria CPU004. 67

Tabla 22: Caso de prueba unitaria CPU005. 67

Tabla 23: Caso de prueba unitaria CPU006. 67

Tabla 24: Caso de prueba unitaria CPU007. 68

Tabla 25: Caso de prueba unitaria CPU008. 68

Tabla 26: Caso de prueba unitaria CPU009. 68

Tabla 27: Caso de prueba unitaria CPU010. 68

Tabla 28: Caso de prueba unitaria CPU011. 69

Tabla 29: Caso de prueba unitaria CPU012. 69

Tabla 30: Caso de prueba unitaria CPU013. 69

Tabla 31: Caso de prueba unitaria CPU014. 69

Tabla 32: Caso de prueba unitaria CPU015. 70

Tabla 33: Caso de prueba unitaria CPU016. 70

Tabla 34: Caso de prueba unitaria CPU017. 70

Tabla 35: Caso de prueba unitaria CPU018. 71

Tabla 36: Caso de prueba unitaria CPU019. 71

Tabla 37: Caso de prueba unitaria CPU020. 71

# Definición del problema.

En la licenciatura de redes y servicios de cómputo los estudiantes deben realizar prácticas de cableado estructurado, particularmente en la materia de arquitecturas en red, pero existe el problema de la carencia de materiales y equipo, algunos elementos necesarios son caros por lo que la universidad no cuenta con ellos, entonces los estudiantes deben gastar o limitarse a la parte teórica. Aunado a esto, otro problema es la dependencia entre las prácticas y los docentes, pues según el profesor, algunas prácticas simplemente no se realizan.

Para el estudiante, quedarse solo con la parte teórica impacta negativamente en su aprendizaje y desarrollo de competencias, puesto que lo aprendido en clase no es llevado a la práctica a pesar de ser primordial según el perfil de egreso. Los estudiantes egresados deberán ser capaces de “determinar las necesidades de red y servicios de cómputo de una organización, diseñar y crear redes de cómputo eficientes, instalar, actualizar y reparar hardware y software de red” (Universidad Veracruzana, 2018), lo cual hace imperativo aprender y reforzar los conocimientos teóricos mediante prácticas.

# Objetivos.

Extender la funcionalidad de un LMS para permitir la publicación y el uso de laboratorios virtuales dirigidos a los estudiantes de la Licenciatura en Redes y Servicios de Cómputo, donde aprendan practiquen y refuercen sus conocimientos teóricos.

Objetivos específicos:

1. Diseñar para poder integrar nuevos escenarios de prácticas de forma sencilla.
2. Mantener un registro de ingreso y uso.
3. Realizar seguimiento y dar retroalimentación a las actividades del estudiante.
4. Mantener un registro de la mejora en la realización de las actividades.
5. Acceder desde cualquier lugar y en cualquier momento.

# Justificación.

La realización de la plataforma virtual significará un apoyo académico tanto para la Universidad Veracruzana como para los estudiantes de la Licenciatura en Redes y Servicios de Cómputo: la realización y evaluación de prácticas de cableado estructurado independientes del espacio, los materiales, equipo y docentes que imparten la experiencia educativa de arquitecturas en red, de forma interactiva.

Idealmente, los estudiantes deben realizar prácticas reales de cableado estructurado, sin embargo, en algunos casos esto resulta muy costoso, por lo que el docente podrá enseñar algunos temas de cableado estructurado que son difíciles de llevar a la práctica y dar la plataforma a sus alumnos como un apoyo educativo, donde estos identifiquen los elementos expuestos en clase, posteriormente realicen prácticas reales con el equipo y los materiales que la universidad les proporciona, y mediante la plataforma practiquen con los demás elementos que no pueden tener en físico, demostrando y reforzando así lo aprendido. Al final el docente podrá retroalimentar y evaluar a los estudiantes.

# Alcances y limitaciones.

Dada la amplitud del trabajo, se pretende alcanzar únicamente un prototipo de la plataforma donde se integren un escenario de prueba manteniéndose en un nivel de complejidad básico, dejando así trabajo y documentación para un seguimiento futuro por parte de otros estudiantes.

También, debido a la duración estimada del trabajo, no se podrá alcanzar a evaluar la plataforma con los estudiantes de la experiencia educativa de arquitecturas en red.

Como limitantes se tienen los siguientes puntos:

1. Primera vez que se realiza una aplicación de realidad aumentada y/o virtual.
2. Tiempo, ya que no alcanza para desarrollar una plataforma con todos los escenarios que se necesitan.

# Capítulo 1: Marco teórico.

En este primer capítulo se presentan los fundamentos teóricos y de contexto para el desarrollo del proyecto, que van desde la instrucción por computadora, pasando por la aparición de las herramientas de aprendizaje en línea “e-learning” como los sistemas de gestión de cursos (LMS), sus características, los laboratorios virtuales como contenido de aprendizaje que estos soportan, hasta llegar a estándares para el manejo de objetos de aprendizaje y empaquetado del contenido a añadir a un LMS, como SCORM, IMS y Experience API. Así mismo, se incluye información relacionada con la metodología de desarrollo de software empleada.

Producto del aumento en el uso de Internet y las mejoras en las TIC, han surgido nuevas tecnologías que representan grandes oportunidades para distintos campos como el instruccional o educativo, para el cual pueden crearse ambientes de aprendizaje como alternativas para los puntos de vista tanto de las instituciones, como los profesores y los alumnos.

Las TIC son capaces de soportar procesos cognitivos y aspectos psicológicos, estas pueden emplearse como herramientas de mejoramiento o apoyo para el aprendizaje presencial, e incluso para crear ambientes de enseñanza cuya interacción sea gestionada por un sistema informático. Sin embargo, a pesar de las aplicaciones tecnológicas en este ámbito, es importante tomar en cuenta que los profesores son y serán elementos esenciales, es importante que, aunque su rol cambie siga existiendo una comunicación clara entre ellos y los alumnos.

La historia dice que cuando las computadoras habían sido recién inventadas, grupos de educadores y psicólogos notaron oportunidades instruccionales, por lo que de manera relativamente rápida surgió el enfoque de instrucción basada en computadora, CBI (Computed Based Instruction) por sus siglas en inglés. CBI buscó la automatización de nociones simples de aprendizaje, llegando a desarrollar métodos instruccionales efectivos (Advanced Distribute Learning Initiative, 2001).

Fue en la década de 1960 que los investigadores en paralelo con ingenieros de CBI comenzaron a explorar el potencial de un nuevo enfoque de información orientada a estructuras (Information Structure - Oriented) que permitiría representar el aprendizaje humano. Posteriormente, gracias a estudios en inteligencia artificial relativos a cómo aprendemos, se llegó a desarrollar un nuevo enfoque llamado Sistemas Tutores Inteligentes (ITS), cuya función fue proporcionar información instruccional bajo demanda y permitir un diálogo de iniciativa entre la tecnología y estudiantes (Advanced Distribute Learning Initiative, 2001).

Lamentablemente la tecnología ha avanzado rápido, y cada innovación tecnológica trae consigo un nuevo mundo de oportunidades que pueden barrer a tecnologías que parecían prometedoras. Tal es el caso de CBI e ITS, pues con el nacimiento y crecimiento de Internet y la *World Wide Web* (WWW), el desarrollo de estos enfoques se vio interrumpido. Mientras que el contenido de CBI se almacenaba y ejecutaba en un ambiente local, el internet proporcionó una plataforma de comunicaciones neutral creada sobre estándares, la cual permitía la administración de contenido remoto e información fácilmente accesible.

El desarrollo de Internet propició la creación de ambientes distribuidos en la web, cuyos beneficios fueron vistos por la comunidad de CBI. Los sistemas rápidamente comenzaron a pasarse hacia la separación de contenido y control de los sistemas cliente – servidor. Finalmente, el campo de la enseñanza tomó ventaja del Internet para proveer educación en línea o *e-learning* y soporte adicional para los estudiantes.

## Sistemas de gestión de cursos.

Un problema recurrente en instituciones educativas es la limitante de acceso estudiantil debido a la falta de infraestructura física, por lo que algunas instituciones optan por proporcionar nuevas opciones educativas haciendo uso de la tecnología. Los ambientes de enseñanza pueden desarrollarse como plataformas aprendizaje en línea o *e-learning*, que tienen la capacidad para resolver o apoyar problemas como el de acceso, debido a la falta de utilidad de espacios físicos, involucrando tiempos y profesores.

Con el uso de plataformas de aprendizaje en línea las instituciones educativas y los profesores pueden proporcionar información y prácticas individuales o colaborativas a sus alumnos, recursos que los ayuden a desarrollar habilidades requeridas e incrementar su conocimiento. Además, Paulsen (2002) menciona que incluso el tiempo de aprendizaje puede disminuir a un 50% y la taza de retención aumenta con este modelo de aprendizaje.

Las tecnologías de aprendizaje en línea también son una opción flexible de estudio, ya que pueden ser más adaptables que las clases tradicionales. Las plataformas de *e-learning* han incorporado ciertas necesidades de adaptabilidad para los usuarios, por ejemplo, configuraciones según el nivel de experiencia, y otras relacionadas con la accesibilidad, tales como configuraciones de colores, sonidos, tamaños de letra, etc.

Existen sistemas de aprendizaje en línea que han nacido y evolucionado rápidamente llamados Sistemas de Gestión de Cursos, LMS por sus siglas en inglés, que se sabe tienen efectos positivos en la enseñanza y aprendizaje. Los LMS (Learning Management System), CMS (Course Management System) o VLE (Virtual Learning Environment) son sistemas que, según Coates, James, & Baldwin (2005), ofrecen la entrega de programas de aprendizaje a gran escala y aumentan la eficiencia del proceso de enseñanza. En la literatura se encuentran varias definiciones de LMS:

“Un sistema de administración de aprendizaje (LMS) es un software para la administración, documentación, rastreo, reporte y entrega de tecnología educacional, cursos o programas de entrenamiento”. (Tretinjak, 2018).

“Un LMS es un paquete de software que habilita el manejo y entrega de contenido de aprendizaje y recursos para estudiantes”. (Awang & Darus, 2012).

“Un LMS se define como un software que ha sido utilizado en una presentación de contenido de aprendizaje, mismo que tiene un rol significativo y complejidad en el ambiente de *e-learning”*. (Aydin & Tirkes, 2010).

“Los LMS son sistemas escalables que pueden emplearse para soportar enteramente los programas de enseñanza y aprendizaje de las universidades”. (Coates, James, & Baldwin, 2005).

Por lo tanto, se puede definir a un LMS como un sistema de software web que permite la gestión de programas educativos, desde el punto de vista administrativo, de enseñanza y aprendizaje mediante contenido de aprendizaje con un control y rastreo de las actividades de los estudiantes.

Estos sistemas proporcionan simulaciones de ambientes reales de aprendizaje con el potencial de afectar al negocio central de la enseñanza y responder a las demandas masivas de acceso a la educación. Su conjunto de herramientas de software soporta los procesos de enseñanza y aprendizaje, así como habilita a los estudiantes para estudiar en cualquier momento, en cualquier lugar, entregándoles el contenido y gestionando su progreso.

La idea de utilizar un LMS en la educación superior resulta atractiva ya que es una forma de reforzar la suite pedagógica con la que se cuenta, y porque promete enriquecer el aprendizaje de los estudiantes con flexibilidad de cursos, comunicación, apoyo y evaluaciones adaptables, entre otras cosas. Hoy en día los LMS significan un cambio de cultura en el ámbito educativo que debe contemplarse.

Cuando una institución educativa se plantea extender sus capacidades mediante soluciones tecnológicas, un LMS se convierte en una necesidad puesto que la población estudiantil es muy diversa, existen múltiples medios de acceso, formas y modalidades de aprendizaje, así como niveles de experiencia tecnológica. La flexibilidad y la naturaleza web de los LMS permite que creen espacios de colaboración donde se promueve la participación de distintos estudiantes para lograr metas de aprendizaje.

Un LMS es un sistema complejo de aprendizaje en línea, puede monitorear el rendimiento de los estudiantes y determinar secuencias de interacción, pero todo es basado en contenido de aprendizaje que se entrega a los estudiantes, el cual que debe crearse. A pesar de ser un esfuerzo extra para las instituciones y profesores, el contenido de los cursos para LMS es fácilmente desarrollado, modificado y reutilizado.

### Características de los sistemas de gestión de cursos.

Los LMS implican una plataforma de aprendizaje en línea que entregue contenido de enseñanza, así como combine un rango de herramientas administrativas y pedagógicas para gestionar cursos, usuarios y contenido multimedia. Las herramientas provistas permiten desarrollar y estructurar el contenido de aprendizaje según la presentación deseada, permiten el desarrollo de actividades, evaluación de éstas, y una comunicación para con los profesores y compañeros donde compartir información y recursos. Es importante mencionar que es posible modificar los procesos y la apariencia de los LMS para adecuarlos a las necesidades.

Como mínimo, todo sistema de aprendizaje en línea considerado LMS debe contar con administración de usuarios, catálogos de cursos, almacenamiento de información de los estudiantes y reportes. Los LMS se encargan de rastrear a las personas (usuarios) y seguir sus acciones; generalmente se cuenta con los roles de estudiante y profesor o tutor cuyos privilegios son gestionados. Para el rol de estudiante, los LMS realizan un seguimiento de su progreso con las actividades que estos realizan, mientras que, para el rol de tutor, permiten que estos agreguen contenido, controlen su flujo y revisen la respuesta de los estudiantes a las actividades y evaluaciones, información que puede observarse por medio de reportes de rendimiento.

Con respecto al contenido de aprendizaje, los tutores pueden adaptar los materiales de clase para presentarlos dentro de un LMS, pueden crear actividades con una descripción, documentos relacionados y contenido multimedia variado, tal como instrucciones textuales o en video, imágenes, enlaces web y demás. Los LMS también permiten la creación de exámenes o cuestionarios con los que los tutores pueden evaluar el cumplimiento objetivos de aprendizaje, o recibir retroalimentación de los estudiantes. En general, las evaluaciones que permiten los LMS pueden contener preguntas abiertas, de opción múltiple, etc. Así como incluso tener límites para el tiempo de respuesta.

El contenido necesita estar disponible para los estudiantes, sin embargo, como ya se mencionó anteriormente en el apartado 1.2, los tutores o profesores son elementos importantes para el proceso instructivo, por lo cual los LMS deben proporcionar herramientas orientadas a la comunicación y retroalimentación. Algunas herramientas de comunicación proporcionadas por LMS incluyen chat, correo electrónico, foros, video conferencias y cuestionarios, mismas que, dependiendo del LMS con que se trabaje, pueden tener usos distintos.

Tanto los chats como el correo electrónico permiten que los estudiantes compartan información entre ellos o con sus tutores, con la posibilidad de incluir archivos y/o multimedia en los mensajes, los cuestionarios, como ya se mencionaron, fungen como herramientas de comunicación al permitir a los tutores recibir retroalimentación de parte de los estudiantes, y las video conferencias son utilizadas para reuniones en línea.

Los foros se consideran espacios sociales con múltiples usos, son útiles para mantener viva la comunicación dentro de un ambiente de aprendizaje. Dentro de los foros, los estudiantes pueden conocerse, discutir temas y consultar dudas entre ellos, donde también pueden participar los tutores, además, pueden utilizarse como centros de ayuda y como espacios para consultar información y anuncios relativos al curso.

Los LMS gestionan contenido de aprendizaje, pero deben permitir crearlo, distribuirlo, integrarlo y reutilizarlo rápidamente, deben contar con herramientas que permitan su creación y proporcionar extensibilidad. Es posible que sea necesario crear contenido de aprendizaje que pueda ser utilizado por distintos LMS, para lo cual, existe el requisito de compatibilidad con estándares de contenido, un LMS debe aceptar la integración de contenido en base a un estándar como ADL SCORM, mismo que se trata más adelante en este capítulo.

### Empleo de sistemas de gestión de cursos.

Los LMS son capaces de soportar las formas de estudio de tiempo completo, parcial y a distancia, las instituciones educativas pueden emplearlos para desarrollar carreras y cursos completamente en línea (aprendizaje a distancia), o solo proporcionar contenido adicional a los estudiantes que evaluar mediante tareas y exámenes. En muchas ocasiones los profesores y estudiantes no tienen la capacidad de participar físicamente en el proceso enseñanza – aprendizaje en un mismo espacio, por lo cual, el aprendizaje a distancia se convierte en una alternativa atractiva, que mediante el uso de LMS proporciona los recursos necesarios.

Con los LMS, los profesores pueden colaborar con otros profesores de la comunidad, presentar ambientes de aprendizaje dinámicos y ofrecer apoyo para que los estudiantes enfoquen su aprendizaje mejorando sus objetivos. Los profesores pueden ofrecer una guía personal para que los estudiantes interactúen con los materiales, actividades y tópicos del curso que normalmente son presentados en clases presenciales. Todos los contenidos de un curso pueden adaptarse para su presentación dentro de un LMS, además, pueden planificarse lecciones y secuencias para lograr objetivos de aprendizaje con las herramientas es que estos incluyen.

Existe una técnica de enseñanza favorecida por los LMS llamada enseñanza volteada, se trata de una técnica de aprendizaje en apoyo de clases tradicionales donde los profesores motivan a los estudiantes a revisar el contenido de los cursos antes de trabajar cara a cara con ellos, o realizar investigaciones. El objetivo de la enseñanza volteada es permitir la exploración personal del contenido de un curso, para que los estudiantes puedan alcanzar sus objetivos y lograr resultados a su propio ritmo donde sea que se encuentren.

Con el paso del tiempo las universidades han adoptado los LMS a pesar de sus costos y riesgos debido a algunas ventajas observadas al momento de utilizarlos. Los LMS presentan oportunidades para reducir costos generales y la demanda de espacios físicos, mejorar el acceso a la información y la calidad de los procedimientos, lo cual es atractivo en los ámbitos de administración e instrucción. Los cursos diseñados y almacenados en un LMS son de mucho valor, pues capturan el conocimiento y presentan utilidad en el presente y a futuro.

* + 1. Comparativa de sistemas de gestión de cursos.

En el mercado se aprecia una gran variedad de LMS, algunos comerciales como Topclass, Firstclass, NextEd, WebCT Vista, Blackboard, etc. Y otros *Open source* como Moodle, ATutor, Ilias, Claroline, Dokeos, etc. Que ofrecen servicios y cuentan con requisitos y precios distintos. Generalmente los LMS *Open source* no tienen costo, y suelen ser atractivos ya que llegan a ofrecer las mismas funcionalidades de LMS comerciales, sin costo. Los LMS más populares son Schoology, Edmodo y Moodle, sin embargo, Ruiz, N. et al. (2005) mencionan que, aunque ninguno alcanza una completa adaptabilidad, ATutor, Ilias y Moolde han aparecido como los mejores LMS al momento.

Los LMS y en general el software llamado *Open source*, hace referencia que el código es accesible para cualquier persona interesada, “*Open source* (OS) es el código fuente de un software que está disponible al público para su extensión y modificación según la necesidad del usuario”. (Cavus & Zabadi, 2014). Cuando se tiene la necesidad de utilizar un LMS, este debe elegirse correctamente, puesto que algunos no son claros en sus términos de uso o simplemente no son muy eficientes. A continuación, se presenta una comparativa de LMS *Open source* en base a la revisión de la literatura.

Tabla 1: Comparación de LMS Open source.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **LMS** | **Moodle** | **ATutor** | **Dokeos** | **Claroline** | **Ilias** | **Docebo** |
| **SCORM** | Importar / exportar.  SCORM 1.2.  IMS 2.0.  Conformidad certificada. | Crear / Importar / exportar.  SCORM 1.2.  IMS 1.1.3.  Conformidad certificada. | SCORM 1.2 / 2004.  Conformidad. | SCORM 1.2.  Conformidad mínima. | SCORM 1.2 / 2004.  IMS.  Conformidad certificada. | SCORM 1.2 / 2004.  Conformidad certificada.  Premiado. |
| **Seguimiento** | Enlaces visitados.  Uso de contenido.  Actividades realizadas. | | | Actividades realizadas. |  | Actividad del estudiante. |
| **Comunicación** | Skype.  Foros.  Chat. | Foros.  Correo electrónico.  Chat. | Video conferencias.  Foro.  Chat. | Foros. | Foros.  Correo electrónico.  Chat. | Video conferencias.  Foros.  Chat. |
| **Evaluación** | Cuestionarios personalizados.  Preguntas con límite de tiempo. |  | Exámenes de preguntas múltiples. |  |  |  |
| **Reportes** | Actividades realizadas.  Participación.  Estadísticas. |  | Puntuaciones.  Respuestas.  Retroalimentación.  Progreso individual. | Estadísticas de actividades. | Cursos.  Usuarios.  Exámenes. | Reportes personalizados. |

La tabla 1 representa una comparación básica de seis LMS Open source según su compatibilidad con SCORM, herramientas de comunicación incluidas, evaluaciones, tipos de reportes y el seguimiento del estudiante. Si bien no se encontró información sobre las evaluaciones, reportes y el seguimiento para algunos LMS, se sabe que por definición los LMS manejan el seguimiento del estudiante, así como también contienen formas de evaluación y de reportes.

La mantenibilidad y modularidad son aspectos importantes para trabajar con LMS, la mantenibilidad es un factor de calidad de software, es una conveniencia donde el software puede modificarse para aumentar su capacidad. Anggrainingsih et al. (2016) realizaron el cálculo del índice de mantenibilidad de ATutor, Ilias y Moodle. El índice de mantenibilidad permite estimar la posibilidad de modificar y mantener el código fuente, mientras más alto sea el índice, más fácil es mantener el software.

El cálculo se realizó según las métricas de PHP, las cuales pueden dar valores entre 0 y 118 para el índice de mantenibilidad. Como resultado, se obtuvo un índice de mantenibilidad de 73, 87 y 90 para Atutor, Ilias y Moodle respectivamente, donde Moodle es el más alto de los tres; esto significa que el código fuente de Moodle es más fácilmente mantenible y modificable en comparación con ATutor e Ilias.

Anggrainingsih et al. (2016) también mencionan en su trabajo que la modularidad, determinada por la cohesión y el acoplamiento, así como la simplicidad, son elementos muy importantes en el desarrollo de software, “la modularidad es un sub factor de la flexibilidad determinado por la cohesión y el acoplamiento. La cohesión muestra enlaces entre las funciones de un módulo, mientras que el acoplamiento muestra dependencias entre un módulo y otros”. (Anggrainingsih et al. 2016).

Los cálculos de cohesión, acoplamiento y complejidad también se realizaron en base a métricas de PHP, la simplicidad es medida según el número de líneas de código (LOC), y está ligada a la complejidad, que, a su vez, es medida según la complejidad ciclomática del software. Los resultados en cohesión y acoplamiento indican que Moodle sobresale, pues mantiene una alta cohesión y bajo acoplamiento en comparación con los otros dos LMS, sin embargo, la medición de complejidad dio los valores de 16.8, 18.2 y 17.3 para ATutor, Ilias y Moodle respectivamente, donde ATutor muestra el nivel más bajo de complejidad de los tres.

Al igual que la mantenibilidad, existen otros factores de calidad tales como la seguridad y la usabilidad. El estándar ISO/IEC 25010 define seguridad y usabilidad respectivamente como el grado en que un sistema protege la información para que las personas o sistemas tengan el grado de acceso a la información de acuerdo con sus niveles de autorización, y, el grado en el que un sistema puede utilizarse por usuarios específicos para alcanzar metas específicas con eficiencia, efectividad y satisfacción en un contexto de uso determinado.

Ouadoud, Chkouri, & Nejjari (2018) realizaron una evaluación de la calidad de plataformas de aprendizaje en línea incluyendo factores de calidad como la mantenibilidad, usabilidad y seguridad. La métrica de evaluación de la calidad empleada fue el índice de conformidad del estándar ISO 9126, con el cual, las plataformas obtienen una calificación representada como un porcentaje dividido en cuatro rangos: bajo el estándar (0 - 40), promedio (41 - 60), bueno (61 - 80) y muy bueno (81 - 100). Los resultados obtenidos en la evaluación se muestran en la siguiente tabla:

Tabla 2: Resultados de la evaluación de la calidad de LMS.

|  |  |  |  |
| --- | --- | --- | --- |
| **LMS** | **Mantenibilidad** | **Seguridad** | **Usabilidad** |
| Moodle | 100% | | |
| ATutor | 60% | 100% | 81% |
| Ilias | 80% | 88% | 75% |
| Claroline | 20% | 88% | |
| **Nota.** Fuente: LeaderTICE: A Platforms Recommendation System Based on a Comparative and Evaluative Study of Free E-learning Platforms. | | | |

En conclusión, se encuentra en la literatura que en general los LMS Open source ofrecen lo suficiente para un uso en organizaciones educativas, son ricos para el soporte de aprendizaje y mejoramiento de la conducción de la enseñanza, aunque algunos ofrecen mejores herramientas y sobresalen del resto. Varios autores coinciden en que Moodle es el mejor LMS para utilizar, Moodle cuenta con la base de datos de usuarios más grande, se ha ganado la preferencia de muchos de ellos alrededor del mundo. Muchas instituciones educativas emplean Moodle para presentar sus cursos en línea (Saydam et al. 2013).

Sabine and Bate (2005) evaluaron características de adaptabilidad, resultando Moodle como el más alto, y Al-Ajlan (2012) realizó un análisis extenso concluyendo que Moodle es el mejor de todos los LMS. Una característica de Moodle mencionada por varios autores es que éste tiene la capacidad de escalar y llegar a despliegues para cientos de miles de usuarios y comunidades sin ningún problema.

En las características particulares, la tabla 1 muestra que los seis LMS comparados admiten contenido obediente a SCORM, Moodle y ATutor cuentan con las mejores herramientas de comunicación, las mejores interfaces de usuario (GUI) y con información muy accesible; en contraste, Claroline es mucho más simple, contiene menos herramientas y sus páginas tienen una complejidad significativa que dificulta el acceso a la información.

## Laboratorios virtuales.

Las prácticas de laboratorio son muy importantes para la instrucción técnica en general, Ye & Ho (2016) indican que el aprendizaje experimental es sumamente importante en el proceso de enseñanza – aprendizaje en diversas áreas como la bioquímica, ya que pueden facilitar la consolidación de conceptos. Así mismo, otros autores como Wiseman et al. (2018) mencionan que los laboratorios “son una experiencia importante en la informática y la ingeniería, permiten a los estudiantes aplicar el conocimiento de clase en software y sistemas reales, y ayudan a que se familiaricen con el estado del arte de la tecnología y los estándares de la industria”.

Desafortunadamente, la literatura demuestra que muchas instituciones educativas carecen de ciertos elementos para que sus estudiantes realicen prácticas de laboratorio, o bien, presentan carencias que les restan efectividad a las prácticas. Tal como menciona Stark et al. (2018) no siempre es posible experimentar con dispositivos reales ya sea por altos costos o seguridad, la necesidad de infraestructura y recursos humanos vuelven oneroso a un laboratorio físico (Voss et al. 2013). Altos costos, preparación, recursos, tiempos y profesores, entre otros factores, llegan a imposibilitar la realización de prácticas en laboratorios físicos, sin embargo, una mala instrucción puede hacer a los estudiantes propensos al peligro.

Durante los últimos años, varios sectores sociales se han visto afectados por los avances tecnológicos. Con la implementación de las TIC pueden generarse servicios que soporten procesos y actividades de índole educativo, su avance actual permite hacer uso de herramientas digitales para crear, o como materiales de aprendizaje, así como implementar elementos multimedia en las aplicaciones. Así mismo, las tecnologías multimedia han avanzado de tal forma que han permitido progresar el modo de entrega por computadora, mismo que ha hecho posible el diseño de software educativo interactivo.

En décadas pasadas el internet no era la tecnología ampliamente utilizada que vemos hoy en día, el sector educativo necesitaba forzosamente de la experimentación en laboratorios físicos tomando en cuenta todas las implicaciones de éstos, tales como regulaciones de seguridad para casos específicos. Tal como menciona Luengas et al. (2009), el internet hace posible que educadores y aprendices puedan capacitarse sin estar sujetos a especio y tiempo. Gracias a él pueden realizarse experimentos con el uso de recursos de software e información que no se tiene en los laboratorios reales. El progreso tecnológico nos presenta un enfoque para resolver retos mediante laboratorios virtuales específicos para simulaciones.

Gracias al internet y al modo de entrega por computadora puede ofrecerse la educación virtual, una de las áreas más impactadas por el desarrollo tecnológico. La educación virtual provee servicios especializados de soporte para asuntos administrativos y cursos virtuales (Luengas et al. 2009). El sector educativo puede aprovechar estas tecnologías a su favor mediante laboratorios de enseñanza virtuales como software educativo, los cuales “permiten el contacto directo y la observación de objetos y eventos, así como descubrir aspectos científicos” (Voss et al. 2013).

El término software educativo hace referencia a “aquel programa informático que se emplea como recurso educativo, que ha sido concebido y desarrollado bajo claros objetivos didácticos para la generación de ambientes que favorezcan la enseñanza y el aprendizaje” (Luengas et al. 2009). Un laboratorio virtual como herramienta educativa debe facilitar el proceso de enseñanza – aprendizaje, pues un problema recurrente es que los alumnos reciben conocimientos teóricos, pero pueden no cubrir el contenido o competencias suficientes para su formación. Luengas et al. (2009) menciona que diseñar y desarrollar laboratorios virtuales educativos debe incorporarse a los modelos de enseñanza, pues motivan a realizar prácticas de laboratorio.

Actualmente se cuenta con una serie de herramientas educativas adaptadas en distintas áreas como las ciencias sociales, humanidades, artes, ingenierías, etc. Un ejemplo son los cursos en línea que pueden ofrecer videos instruccionales, aunque no llegan a resolver el problema. Es necesario utilizar y desarrollar herramientas didácticas que soporten el proceso de enseñanza – aprendizaje, “se requiere material que capture la atención de los estudiantes y los estimule mediante escenarios interactivos, como en el caso de los laboratorios virtuales”, desarrollos innovadores donde la comunicación tenga un papel muy importante (Luengas et al. 2009).

La utilización de software interactivo como herramienta puede ser beneficiosa para la instrucción, según Hashemi et al. (2005) el software interactivo podría abordar algunas deficiencias y mejorar tanto la experiencia de aprendizaje como el rendimiento de los estudiantes, el software puede proporcionar una retroalimentación inmediata al estudiante sobre la exactitud de un enfoque o solución en el que trabaja o propone. Un laboratorio virtual puede motivar a un estudiante a la experimentación presentando un ambiente interactivo atractivo, así como apoyar en su aprendizaje mostrando información según el progreso del mismo.

Los laboratorios virtuales emergen como un reemplazo potencial para las prácticas de laboratorio, son convenientes para la educación debido a varios factores, éstos son capaces de imitar prácticas reales enfatizando técnicas de experimentación y aplicaciones destinadas a un seguimiento continuo de las actividades de los estudiantes, además, “son una alternativa para las prácticas de laboratorio costosas o peligrosas” (Luengas et al. 2009), en un laboratorio virtual todos los componentes son simulados por completo, no existe equipamiento físico susceptible a daños y se aleja a los estudiantes de los riesgos posibles de trabajar con un laboratorio real.

Cuando los estudiantes trabajan con laboratorios virtuales el acceso es de forma remota, según Wiseman et al. (2008) los estudiantes responden bien a estos ambientes. A pesar de no interactuar cara a cara con los profesores, los estudiantes no están solos; tal como se menciona anteriormente, Luengas et al. (2009) Indica que existe un seguimiento continuo de las actividades de los estudiantes, los laboratorios virtuales son herramientas que permiten contar con un acompañamiento y seguimiento, los estudiantes son responsables de participar en actividades con los materiales educativos presentes tales como prácticas de laboratorio, actividades que pueden complementarse en clases presenciales.

Budai & Kuczmann (2018) describen a los laboratorios virtuales como sistemas de comunicación utilizados por profesores para crear experimentos virtuales y materiales de aprendizaje, que son usados para obtener conocimiento mediante las experiencias virtuales. Un laboratorio virtual es un programa de computadora, una representación de software que “imita las características físicas de objetos reales” (Luengas et al. 2009) con los que los estudiantes interactúan mediante una computadora conectada a internet (Stark et al. 2018).

Como se menciona anteriormente, los laboratorios virtuales son un reemplazo potencial para las prácticas de laboratorio, buscan simular las mismas actividades que involucra un laboratorio real, además, éstos pretenden según Wiseman el al. (2008) atender a una gran cantidad de alumnos y reducir la carga financiera de un laboratorio físico. El objetivo de los laboratorios virtuales es “introducir a los estudiantes en la experimentación, resolución de problemas, deducción de resultados e interpretación científica a través de sistemas 3D con componentes conformantes de un laboratorio virtual” (Luengas et al. 2009).

### Realidad virtual.

Por definición, los laboratorios virtuales utilizan interfaces gráficas basadas en realidad virtual, se trata de una relativamente nueva tecnología de computación electrónica de procesamiento gráfico digital, multimedia y de sensores. La realidad virtual “replica un ambiente que simula una presencia física en lugares del mundo real o imaginados, permitiendo al usuario interactuar en ese mundo” (Stark et al. 2018), construye un ambiente fiel a los sentidos humanos como la vista y oído (Guo et al. 2012), para lo cual, utiliza sensores y dispositivos de interacción con los que reaccionar a los eventos provocados por las acciones del usuario, ya sea de forma gráfica o auditiva mediante elementos multimedia.

Mediante la realidad virtual puede desarrollarse un laboratorio virtual donde los estudiantes realicen experimentos de simulación en internet y en cualquier momento, y esta, siendo utilizada de forma efectiva, después de un análisis del diseño, puede ir más allá de la inmersión provocando una sensación de presencia que asegure una experiencia de usuario irresistible (Callaghan et al. 2015), dónde el estudiante se sienta motivado para realizar prácticas con el software.

Stark et al. (2018) menciona que es imposible reemplazar por completo la realización de prácticas de laboratorios reales con un sistema informático, sin embargo, realizarlas en realidad virtual es mejor que no hacerlo. Los modelos 3D dentro del ambiente virtual son convenientes, su interactividad permite la manipulación de elementos virtuales y la observación de un ambiente, la realidad virtual permite incluso visualizar, identificar y controlar sistemas complejos, que, según Stark et al. (2018) “permite a los estudiantes obtener un mejor y más rápido entendimiento de los objetos estudiados”.

Construir una aplicación de realidad virtual no es una tarea muy complicada, a pesar de que en el caso de la instrucción existen elementos pedagógicos que deben ser tomados en cuenta, existen ya dispositivos, estándares y una variedad de *frameworks* que nos permiten crear ambientes artificiales, así como incluir, organizar y programar la operación de elementos 3D dentro del mismo.

La realidad virtual no está limitada a ningún dispositivo en específico, un ambiente virtual puede manejarse desde una PC o gafas especializadas tales como HTC Vive y Oculus Rift, la única variación que puede encontrarse está en los dispositivos de interacción, pues mientras el manejo de una aplicación en una PC se realiza mediante dispositivos conocidos como teclados o ratones, para el caso de las gafas son necesarios otros tipos de controles, pues estas solo se encargan de la visualización.

Los usuarios pueden recibir distintas experiencias con aplicaciones de realidad virtual gracias a la variedad de dispositivos del mercado, que se presume serán adquiridos cada vez más con mayor facilidad, según Callaghan et al. (2015) La generación actual de dispositivos de realidad virtual tiene el potencial de convertirse en dispositivos *mainstream* de consumo.

### Consideraciones para el diseño de laboratorios virtuales.

Los laboratorios virtuales han sido añadidos al catalogó de herramientas tecnológicas para la instrucción en los últimos años, pero antes de tomar la decisión de emplearlos es necesaria una evaluación por parte de la institución que lo desea, puesto que existen varios factores a ser tomados en cuenta, que van desde la implementación (dentro de la institución) hasta el diseño del laboratorio incluyendo aspectos pedagógicos y de interfaz de usuario.

En lo que a implementación se refiere, la institución debe considerar las herramientas de hardware y software, es decir, las tecnologías que va a necesitar, los costos de desarrollo, operación y/o mantenimiento, la infraestructura de red, la dificultad de instalación del producto, y, a un nivel pedagógico, los temas que sean más convenientes a tratar mediante laboratorios virtuales (Budai & Kuczmann, 2018). También, ya entrando en el diseño, se debe considerar que para satisfacer las necesidades del desarrollo y de los laboratorios en sí, en necesario reunir a un grupo de ingenieros eléctricos, profesores, desarrolladores de software y diseñadores que trabajen en conjunto para llegar a laboratorios adecuados y funcionales (Luengas et al. 2009).

Debe considerarse cómo las tecnologías ayudarán a conseguir los objetivos de aprendizaje, asegurando que sea la pedagogía la que conduzca el proceso. Al diseñar, el personal encargado de la parte pedagógica debe pensar en formas de presentar el contenido para asegurar el aprendizaje de los estudiantes, según Hashemi et al. (2005) En un laboratorio virtual deben estar presentes los objetivos, guías, así como estímulos, formas de obtener y proporcionar retroalimentación, evaluar el rendimiento y mejorar la retención de los estudiantes.

El autor menciona que los estilos de aprendizaje son un elemento muy importante, pues impactan en la motivación de los estudiantes dentro de su ambiente de aprendizaje. Los estilos de aprendizaje deben considerarse en el laboratorio de forma que éste presente distintas preferencias con las cuales cada estudiante pueda sentirse cómodo a su manera, pues cuando se presentan materiales de aprendizaje en distintas modalidades “hay mayor posibilidad de mejorar y mantener la atención de los estudiantes” (Hashemi et al. 2005).

Ahora bien, con respecto al diseño de la interfaz de realidad virtual, tal como toda interfaz de usuario está debe ser amigable, pero en dos sentidos. En el primero, la interfaz debe agradar al usuario visualmente utilizando elementos gráficos con los que ya se esté familiarizado (según otras interfaces de usuario), y en el segundo, con respecto a los modelos 3D específicos de un laboratorio, se refiere a que la descripción de los modelos del hardware y su imagen deben empatar con lo que el estudiante ya conoce respecto a su experiencia (Dias et al. 2014).

Callaghan et al. Menciona otros elementos que el diseñador debe tomar en cuenta al momento de trabajar con la realidad virtual. El diseñador debe buscar mantener el flujo y una sensación de presencia aprovechando las características psicológicas de los espacios virtuales, sin embargo, debe prever y limitar algunas sensaciones posibles tales como el vértigo y la claustrofobia llevando a una buena y fácil experiencia. Además, el autor menciona una regla cardinal de la realidad virtual que debe respetarse: “no tomar el control de la cámara a menos que sea absolutamente inevitable” (Callaghan et al. 2015).

### Características y empleo de laboratorios virtuales.

Ya se mencionó que los laboratorios virtuales son herramientas instruccionales que imitan características reales, a este respecto, Guo et al. (2012) establece que “pueden simular escenas vívidas de experimentos y proporcionar experiencia práctica similar a un experimento real”. Los laboratorios virtuales son utilizados en la preparación de profesionistas para que apliquen sus conocimientos en el mundo real, se caracterizan principalmente por dar conocimiento técnico experimental que complemente sus clases, pues la idea de un curso con un laboratorio virtual es fungir como complemento de la parte teórica (Budai & Kuczmann, 2018).

Por definición, se sabe que las prácticas o experimentos que se llevan a cabo mediante laboratorios virtuales no son reales, lo cual es una característica implícita dada por el uso de realidad virtual; los ambientes y todos los elementos 3D que componen a un laboratorio virtual son representaciones digitales, artificiales, con la posibilidad de ser tan complejos como se desee, pudiendo conseguir una alta fidelidad que motive a los estudiantes y los mantenga seguros en un ambiente virtual controlado.

Una característica importante es la ausencia del peligro: “un laboratorio virtual es una gran herramienta que funciona como alternativa para las prácticas de laboratorio costosas o peligrosas” (Luengas et al. 2009), pues tanto los componentes representados en él no necesitan ser adquiridos físicamente por la institución, como se aleja a los estudiantes del peligro. Muchas veces al experimentar dentro de un laboratorio físico los estudiantes deben ponerse una bata, utilizar gafas, guantes y acatar reglas del laboratorio con la finalidad de evitar eventos desafortunados, sin embargo, una ventaja de utilizar laboratorios virtuales es que tales eventos no existen.

Los laboratorios virtuales son excelentes complementos, pues permiten realizar variaciones de un mismo experimento sin exponerse a riesgos antes de intentar con equipamiento real (Stark et al. 2018), el que los estudiantes puedan intentar una y otra vez un experimento hasta alcanzar los objetivos sin miedo a fallar o sufrir daño alguno, es otra característica importante. También, según Ye & Ho (2016), los laboratorios virtuales “pueden reducir el riesgo de operaciones inapropiadas de los instrumentos”, es decir, que sin importar la manipulación y uso que se les dé, al ser virtuales no pueden ser dañados.

Ahora bien, a pesar de la posibilidad de evitar riesgos tanto de salud como económicos, un problema es la falta de seriedad en las prácticas. Al estar los estudiantes completamente seguros y con toda libertad, pueden tender a darle poca importancia a las consideraciones de seguridad y de empleo de los elementos, lo cual puede llevar a una mala instrucción que, en vez de alejarlos, los acerque al peligro en situaciones reales. Una forma de atacar este punto es aprovechar los elementos multimedia para presentar información de contexto.

Según Voss et al. (2013), la información de contexto puede servir para analizar problemas que rodean a los usuarios, tales como la caracterización de estilos cognitivos, análisis del aprendizaje, competencia y, además, es posible establecer reglas y definiciones para asegurar la calidad de la instrucción. Un laboratorio virtual tiene la característica de condicionar el avance de acuerdo con ciertas acciones que demuestran la competencia del estudiante para trabajar con otros elementos, así como de presentar información que éste debe conocer, sin embargo, tal como ya se mencionó, es preferible realizar prácticas reales posteriores, pues no pueden ser totalmente reemplazadas.

Así como reducen los riesgos, los laboratorios virtuales también permiten que tenga lugar la educación a distancia efectiva, lo cual trae consigo dos características benéficas para los estudiantes y las instituciones educativas. Para las instituciones, el uso de laboratorios virtuales es una forma de proporcionar instrucción especializada cuando no se cuenta con los recursos económicos o infraestructura suficiente para contar con laboratorios físicos, así como “hacen ejercicios disponibles a una mayor audiencia” (Budai & Kuczmann, 2018). Para los estudiantes, se rompe la barrera física, pues al trabajar en su propia computadora, ya no se limitan al tiempo y espacio de una institución, pueden hacerlo de forma personal y siempre accesible.

Los estudiantes pueden incluso trabajar en equipo de forma remota, sin la necesidad de materiales físicos ni residuos de componentes desde la comodidad de sus casas o cualquier lugar en el que se encuentren, mientras exista conexión a internet, lo cual es una ventaja (Dias et al. 2014). Una herramienta de laboratorios virtuales no se caracteriza solo por reducir las limitantes de tiempo y espacio, sino también la dependencia de los alumnos con sus profesores, “permitiendo al instructor contribuir de una forma más significativa en el proceso de aprendizaje” (Hashemi et al. 2005), pues ahora puede dar seguimiento a todo el proceso que lleva el estudiante.

Es importante considerar la retroalimentación cuando se emplean laboratorios virtuales, los estudiantes deben ser capaces observar la forma en que realizan los laboratorios, donde puedan conocer lo que hacen bien o mal, cómo lo hacen, cómo debe ser, así como recomendaciones por parte del instructor e información teórica de ser necesario, e incluso autoevaluaciones y evaluaciones del propio laboratorio virtual, donde el profesor pueda determinar si se cumplen los objetivos, si los estudiantes están adquiriendo las competencias y aprendiendo como deben. Según Budai & Kuczmann (2018), es posible rastrear el comportamiento de los estudiantes y generar estadísticas que ayuden a obtener una buena retroalimentación.

Habiendo mencionado la reducción de riesgos y la educación a distancia, queda la interacción de los estudiantes. Como ya se mencionó en el apartado de realidad virtual, los modelos 3D dentro del ambiente virtual son convenientes, Guo et al. (2012) menciona que la interacción humano - máquina permite a los espectadores explorar libremente el ambiente e influenciarlo con su propio comportamiento; utilizando estas aplicaciones “los estudiantes pueden entender el funcionamiento de dispositivos, ver el interior y destacar sus componentes en detalle” (Stark et al. 2018). En su trabajo, Ye & Ho (2016) afirma que después de utilizar un laboratorio virtual, comprobó que los estudiantes estaban más familiarizados antes de realizar prácticas reales.

Al utilizar una interfaz de realidad virtual adecuada los estudiantes pueden controlar un experimento y repetirlo cuantas veces quieran con un solo clic sin ningún contratiempo (Ye & Ho 2012), y si se trata de un ambiente virtual de mayor complejidad donde la interacción se de mediante un dispositivo móvil, es posible incluso utilizar controles propios de las pantallas táctiles tales como joystick virtual, toques y deslices con los cuales manipular el ambiente y sus elementos (Callaghan et al. 2015).

En cuanto al empleo de laboratorios virtuales, Guo et al. (2012) menciona en su trabajo cómo los estudiantes utilizan un laboratorio virtual, estos navegan en el sitio y en el momento en que se acercan a aparatos experimentales despliegan su nombre, descripción y un botón para iniciarse. Una vez iniciado el experimento, el estudiante puede seleccionar cada elemento del apartado utilizando el ratón, pudiendo observar la función, composición y parámetros del mismo; los eventos en la escena virtual se disparan con la actividad del estudiante (pg. Haciendo clic sobre los elementos) y se dirigen a los nodos pertinentes para lanzar animaciones.

Por su parte, Hashemi et al. (2005) presenta en su trabajo una visión más amplia de la forma en que los estudiantes pueden emplear un laboratorio virtual, la organización del contenido de aprendizaje, las opciones del mismo y otros conceptos que fueron utilizados en el diseño del laboratorio, como menús de selección, diagramas de flujo y árboles de decisión. Estas características incorporadas al software acercan al estudiante lo más posible a un ambiente real de laboratorio, permitiendo la toma de decisiones y observación de resultados en un proceso de prueba y error.

Primero, el autor indica que el estudiante obtiene una introducción de los objetivos del experimento que va a realizar, lo cual “influencia positivamente a los estudiantes que necesitan razones para aprender” (Hashemi et al. 2005), para posteriormente elegir materiales desde un menú, que muestra la estructura y características de los mismos, y comenzar el experimento, Los pasos que pueden tomarse durante el experimento se encuentran modelados dentro de un diagrama de flujo, que es utilizado para enfatizar conceptos importantes, identificar aspectos que requieren mayor esfuerzo, así como determinar cálculos y valores a reportar.

Todo el proceso del experimento se presenta paso a paso, en la primera etapa del experimento los estudiantes visualizan un video que detalla el paso que debe realizarse. Una vez concluido el video explicativo, el estudiante procede con la toma de decisiones, que realiza desde un menú de opciones para cada paso del experimento gracias a una estructura en forma de árbol de decisión, el estudiante observa el resultado de sus elecciones y aprende conceptos con la retroalimentación que el software le presenta. Según Hashemi et al. (2005), el concepto de árbol de decisión “puede mejorar la experiencia de los estudiantes con el software y llevarlos a un paso más cercano a una experiencia de laboratorio real”.

Durante la realización del experimento, la interfaz permite a los estudiantes ver imágenes magnificadas de los resultados de los procesos que se siguen con los materiales, y ya en la etapa final del experimento, los estudiantes responden una evaluación con preguntas que los ponen a prueba y sirven para que el profesor verifique el conocimiento que el estudiante obtuvo una vez realizado el experimento.

Hashemi et al. (2005) afirma que el software desarrollado puede emplearse para preparación en programas de ingeniería que requieren materiales de laboratorio, donde los estudiantes se familiarizan con los procedimientos, uso de equipo y errores comunes dentro de una experiencia real, además, pueden fungir como experiencias que reemplacen a la experimentación real para los casos en los que éstas no estén disponibles en la institución.

Por otro lado, generalmente el acceso a un laboratorio virtual no es directo, es decir, existe algún sistema de gestión donde el usuario sigue el flujo de un curso, o bien, simplemente selecciona uno o más laboratorios a realizar. Para que esto sea posible se necesitan algunos elementos, un sistema de laboratorios virtuales se conforma por ciertos componentes; Luengas et al. (2009) identifica tres componentes básicos: uno o más dispositivos de interacción, un dispositivo para la transmisión de la información y un software de aplicación, para los cuales Stark et al. (2018) agrega otros dos: uno o más laboratorios virtuales y un servidor de laboratorio.

Partiendo del punto de vista del usuario, un ejemplo típico de lo anterior sería una aplicación web (pg. un LMS) como software de aplicación, una interfaz web que permita realizar laboratorios virtuales, el teclado, ratón, bocinas y display de una PC como dispositivos de interacción enlazados directamente a la máquina y un par de servidores remotos, los cuales podrían estar dedicados uno al sistema de gestión y el otro meramente al control de los laboratorios virtuales incluidos y utilizados por los estudiantes.

Los laboratorios virtuales como contenido de aprendizaje podrían ser integrados dentro de un LMS, el cual puede emplearse para “autenticar y autorizar estudiantes y profesores, así como conectar lógicamente con los ejercicios del laboratorio virtual” (Budai & Kuczmann, 2018), pues realizar un laboratorio virtual podría ser un requisito para aprobar un curso o actividad determinada, y quien administre el sistema ya sea un administrador o un profesor, podría recolectar las acciones de los estudiantes con motivos de análisis y evaluación.

Hashemi et al. (2005) sugiere que una herramienta que permita a estudiantes entrar y completar un experimento, seguir un procedimiento, y obtener, analizar y evaluar datos y hallazgos, “permite un proceso de aprendizaje que puede mejorar significativamente la experiencia de aprendizaje en comparación con enfoques de laboratorio tradicionales”. Ya sea con un LMS o cualquier otra plataforma para la gestión de laboratorios virtuales, es importante tomar en cuenta que los laboratorios virtuales son complementos para la teoría, por lo cual pueden agregarse videos e información de contexto que suplementen el conocimiento del estudiante.

## Estándares para el empaquetado, manejo y distribución de objetos de aprendizaje.

Los sistemas de aprendizaje en línea (e-learning) se han vuelto muy populares y los educadores deben diseñar contenido de aprendizaje para ellos, sin embargo, compartir estos materiales entre plataformas se vuelve complicado y costoso debido a la falta de formatos uniformes para realizarlo; a pesar de ser un aspecto muy importante, los estándares actuales no soportan completamente la personalización del contenido de aprendizaje, lo cual es una complicación si se requiere compartirlo.

A lo largo de la historia, diversas organizaciones se han dado cuenta de la problemática del e-learning y se han propuesto el desarrollo de una serie de formatos que puedan atender las necesidades del contenido de aprendizaje, llevando a la creación de estándares tales como SCORM, IMS, LOM y AICC, con los cuales, los creadores de contenido puedan tener una base sobre la cual diseñar los materiales ignorando un LMS específico para su publicación, logrando así la posibilidad de compartir y reutilizar contenido entre plataformas.

* + 1. SCORM.

SCORM (Sharable content object reference model) se refiere a un conjunto de especificaciones de organizaciones internacionales como IEEE, LTSC, IMS, AICC e ARIADNE, fue propuesto por ADL (Defense’s advanced distributing learning) en 1997, desarrollado en el año 2000 y se ha convertido en el estándar más popular de entre todos los estándares para el manejo de materiales de aprendizaje (Su et al. 2006), así como el más aceptado.

Gracias a SCORM la experiencia instruccional de los profesores puede compartirse, así como puede desarrollarse la secuencia de actividades del contenido de aprendizaje de una forma más inteligente. De acuerdo con RUSTICI software, SCORM es una herramienta muy potente para cualquier persona que se involucre en el aprendizaje en línea, es necesario hacer que cualquier plataforma de e-learning sea obediente a SCORM para permitir la reutilización del contenido.

SCORM se basa en el concepto de objeto de aprendizaje y utiliza metadatos para especificar la estructura del mismo mediante el lenguaje XML, donde se define la secuencia y navegación, es decir, el comportamiento del curso dentro de una estructura jerárquica llamada árbol de actividades. Su et al. (2006) menciona que es posible proporcionar a los creadores de contenido una herramienta gráfica con la cual se modifique la estructura de un curso en base a ciertas reglas, y esta proporcione un archivo obediente a SCORM de forma automática.

La secuencia y navegación de SCORM controla la secuencia de operación seleccionando y entregando actividades a los estudiantes (Su et al. 2006), Para SCORM, una actividad o evento instruccional es “una unidad estructurada de instrucción” (Rey-López et al. 2006) que en conjunto con otras describe los recursos del contenido de aprendizaje que las contiene, mismo que representa a un único objeto de aprendizaje llamado SCO (sharable content object), la pieza de instrucción básica de SCORM.

Un SCO “representa un solo objeto de aprendizaje lanzable que puede comunicarse con un LMS” (Rey-López et al. 2006) y debe cumplir con ciertos requerimientos de adaptabilidad para con los LMS. Según RUSTICI software, un SCO en un LMS es un ítem rastreable de forma independiente, que contiene un marcador, puntuación y estatus de finalización propios; debe tener varias opciones de comportamiento y reglas de adopción que un LMS pueda decidir dependiendo de las preferencias del usuario y su entorno (Rey-López et al. 2006).

De acuerdo con Su et al. (2006), SCORM “es un conjunto de especificaciones para desarrollar, empaquetar y entregar educación y materiales de entrenamiento de alta calidad, donde sea y cando sea que sean necesarios”, representa tanto el contenido de aprendizaje a distancia como el comportamiento de este (Wang et al. 2005); Rey-López et al. (2006) establece que con la información de secuencia y sus reglas, un LMS decide la organización adecuada para el usuario así como el orden de presentación de las actividades. RUSTICI software establece que, en esencia, el objetivo de SCORM es conseguir que todo el contenido conforme a él peda trabajar con cualquier LMS obediente a SCORM.

Ahora bien, la documentación técnica de SCORM describe los cuatro requisitos de alto nivel para la adaptabilidad: accesibilidad, interoperabilidad, durabilidad y reusabilidad. La accesibilidad hace referencia a la habilidad de localizar y acceder al contenido de forma remota y entregarlos a otros lugares, la interoperabilidad es la habilidad de tomar componentes instruccionales diseñados para una plataforma y utilizarlos en otra. La durabilidad es la capacidad de resistir los cambios tecnológicos, y, por último, la reusabilidad se refiere a la flexibilidad de incorporar componentes en distintos contextos. Para cumplir con los requisitos, SCORM define un modelo de agregación de contenido y un ambiente en tiempo de ejecución.

#### Modelo de agregación de contenido.

El modelo de agregación de contenido de SCORM define cómo identificar, describir y agregar el contenido de aprendizaje, así como métodos técnicos para cumplir los procesos, incluyendo especificaciones para la agregación de contenido y definición de metadatos. Representa los medios pedagógicos para que diseñadores y desarrolladores agreguen recursos de aprendizaje, es decir, información utilizada en una experiencia de aprendizaje como cursos, capítulos y módulos, con una estructura de navegación y secuencia que un LMS pueda interpretar y controlar en tiempo de ejecución.

La documentación del modelo de agregación de contenido de SCORM indica que el propósito de aplicar metadatos es proporcionar una nomenclatura común, una definición con la cual describir los recursos de aprendizaje, es decir, hacer el contenido accesible facilitando su reutilización. Dicha definición es aplicada a un SCO proporcionando información descriptiva de su contenido. El modelo de agregación de contenido agrupa los metadatos en 9 categorías:

Tabla 3: Categorías de metadatos del modelo de agregación de contenido.

|  |  |
| --- | --- |
| **Categoría** | **Descripción** |
| General | Agrupa información general que toma en cuanta al recurso como un todo. |
| Lifecicle | Agrupa características relacionadas con la historia y estado actual de un recurso, así como otros que lo han afectado durante su evolución. |
| Meta-metadata | Agrupa información sobre los metadatos en sí. |
| Technical | Agrupa requerimientos técnicos y características del recurso. |
| Educational | Agrupa las características pedagógicas de un recurso. |
| Rights | Agrupa derechos intelectuales y las condiciones para el uso del recurso. |
| Relation | Agrupa características que definen la relación del recurso con otros recursos. |
| Annotation | Proporciona comentarios para el uso del recurso. |
| Clasification | Describe su el recurso se encuentra dentro de un sistema de clasificación en particular. |

Dentro de un documento llamado “SCORM XML Meta-data document” se representan los metadatos utilizando elementos y atributos en XML. Un elemento “<lom>” corresponde al elemento raíz del documento indicando el inicio del registro, y posteriormente, anidados dentro de éste, se encuentran elementos que corresponden directamente a cada una de las categorías de metadatos de la tabla 3, mismos que no deben de establecerse más de una vez dentro del elemento “<lom>” y se recomienda que se encuentren en orden. Por último, en el nivel inferior a las categorías de metadatos se encuentran anidados los elementos del modelo de información de metadatos de SCORM según cada categoría.

Por otro lado, como ya se mencionó, SCORM también presenta una forma de empaquetado para el contenido, que, de acuerdo con la documentación técnica, proporciona un estándar para el intercambio de los recursos de aprendizaje a lo largo de distintas plataformas, adherido completamente a la especificación de empaquetado de contenido IMS. El empaquetado de contenido IMS define un archivo “Manifest” que describe el paquete en sí, contiene metadatos (archivo XML), una sección de la organización que define la estructura y comportamiento, y una lista de referencias a los recursos del paquete. IMS también define una forma de crear el archivo, así como directivas para empaquetarlo junto con todos los archivos físicos.

La documentación técnica del modelo de agregación de contenido de SCORM presenta un diagrama conceptual que ilustra al archivo Manifest con sus secciones, organizado junto a los archivos físicos, que corresponden al contenido de aprendizaje en sí y cualquier otro archivo necesario.

Ilustración 1: Diagrama conceptual del paquete de contenido IMS.

#### Ambiente en tiempo de ejecución.

De acuerdo con Wang et al. (2005), “el propósito del ambiente en tiempo de ejecución de SCORM es establecer un protocolo estándar para que los cursos hablen a su LMS”. La documentación técnica de SCORM establece que debe existir una forma común para iniciar el contenido y comunicarlo con un LMS durante la ejecución con elementos de datos predefinidos; el propósito es proporcionar medios para cumplir con el requerimiento de primer nivel: interoperabilidad,

El ambiente en tiempo de ejecución de SCORM está conformado por 3 aspectos: Launch, mecanismo de lanzado que define la forma común para iniciar los recursos de aprendizaje en la web, un API (interfaz de programación de aplicaciones) como mecanismo de comunicación entre un SCO y un LMS, y un modelo de datos con elementos estándar que definen la información de estado del recurso de aprendizaje que se comunica mediante el API.

El mecanismo de lanzado define los procedimientos para establecer comunicación entre los recursos de aprendizaje y un LMS, dirige la entrega de los mismos en forma de SCO’s y Asset’s (cualquier otro contenido de aprendizaje que no necesita de la comunicación con el LMS) en una experiencia de aprendizaje, logrando una consistencia en la entrega de recursos independientemente de la implementación de un LMS.

SCORM requiere que el mecanismo lance únicamente un SCO y que solo este se mantenga activo, es decir, no es posible que más de un SCO sea lanzado al mismo tiempo. Una vez activo el SCO, este debe buscar por un adaptador de API, que implementa el API de SCORM y es proporcionado y expuesto por el LMS mediante el modelo DOM de la página web, con el cual podrá comunicarse con el LMS. Para el caso de un Asset, como este no requiere de comunicación, tampoco debe buscar el adaptador.

Ahora bien, el API satisface los requerimientos de alto nivel de SCORM: interoperabilidad y reutilización, pues proporciona una forma de comunicación estándar, incluye un conjunto de funciones predefinidas para los SCO´s disponibles mediante el adaptador de API expuesto por el LMS. Para este proceso de comunicación el LMS en cuestión funge como servidor, mientras que los SCO’s son los clientes. Las funciones del API de SCORM se muestran por categoría en la tabla 4.

Tabla 4: Funciones del API de SCORM.

|  |  |  |
| --- | --- | --- |
| **Categoría** | **Función** | **Descripción** |
| Ejecución de estado. | LMSInitialize(). | Indica al adaptador que el SCO va a comunicarse con el LMS. Permite al LMS inicializar el objeto. Esta función debe llamarse antes que cualquier otra. Recibe una cadena vacía como parámetro y regresa un booleano indicando si la inicialización fue exitosa. |
| LMSFinish(). | Indica que el SCO ya no necesita comunicarse con el LMS. Recibe una cadena vacía como parámetro y regresa un valor booleano indicando si la comunicación terminó. |
| Manejo de estado. | LMSGetLastError(). | Regresa códigos de error resultado de la llamada anterior. |
| LMSGetErrorString(number). | Recibe como parámetro un código de error y regresa una cadena que representa la descripción del mismo. |
| LMSGetDiagnostic(param). | Proporciona detalles de error adicionales. Recibe un código de error como parámetro y regresa una cadena que representa el detalle del mismo. |
| Transferencia de datos. | LMSGetValue(element). | Permite al SCO obtener información del LMS. Recibe como parámetro el nombre de un elemento de datos y regresa una cadena con el valor del mismo. |
| LMSSetValue(element). | Permite al SCO enviar información al LMS. Recibe como parámetros el nombre del elemento de datos y el valor del mismo. |
| LMSCommit(). | Fuerza la persistencia de los valores para el caso en que la implementación de la llamada LMSSetValue() almacene la información en caché. Recibe como parámetro una cadena vacía. |



Ilustración 2: Transiciones de estado del adaptador de API.

La ilustración 2 presente en la documentación técnica de SCORM (ADL, 2001), ilustra los estados por los que pasa un SCO desde que es lanzado, mostrando el proceso de comunicación que sigue un SCO y el orden en el que se realizan las llamadas a las funciones del API:

Primeramente, el LMS lanza un SCO en un estado no inicializado, posteriormente, el SCO encuentra el adaptador de API e indica el inicio de la comunicación mediante la llamada a la función LMSInitialize(), quedando el SCO en un estado inicializado. Ya estando inicializado el SCO, entonces puede llamar a todas las demás funciones, como LMSGetValue(), LMSSetValue(), LMSGetLastError(), etc. Al final del proceso, el SCO indica que la comunicación con el LMS ha terminado mediante la llamada a la función LMSFinish(), quedando el SCO en un estado terminado, sin la posibilidad de llamar a ninguna función.

El API está diseñado para que un SCO obtenga y establezca valores de datos que se encuentran definidos dentro de un amplio modelo de datos externo. A continuación, se muestra una relación de elementos del modelo de datos de SCORM organizados por apartado, que pueden ayudar a cumplir con los requerimientos de evaluación (ver capítulo 2):

1. “cmi.core”. Este apartado del modelo de datos hace referencia a toda la información que debe ser suministrada por todos los LMS.

Tabla 5: Elementos del modelo de datos de SCORM (cmi.core) para evaluación.

|  |  |
| --- | --- |
| **Elemento** | **Descripción** |
| cmi.core.student\_id | Ubica el código alfanumérico único, como identificador de un solo usuario de LMS. |
| cmi.core.lesson\_location | Corresponde al punto de salida de un SCO pasado al LMS la última vez que se experimentó con el SCO. Proporciona un mecanismo para que un estudiante regrese a un SCO en el mismo lugar que lo dejó. |
| cmi.core.credit | Indica si el estudiante está siendo calificado por el LMS en base a su rendimiento en el SCO, donde al ser el caso, el LMS indica al SCO que debe enviarle datos para poder calificarlo. |
| cmi.core.lesson\_status | Corresponde al estatus del estudiante determinado por el LMS, donde el SCO puede haber sido no intentado, solo mostrado en pantalla, no completado, fallado, completado o pasado. |
| cmi.core.entry | Indica si el estudiante ha experimentado anteriormente el SCO (continuación o ya completado) o se trata de su primera vez. |
| cmi.core.score.raw | Indica el rendimiento del estudiante en su último intento con el SCO, el cual puede ser representado como un porcentaje de objetivos cumplidos, así como el número de respuestas correctas. |
| cmi.core.score.max | Corresponde al máximo puntaje que el estudiante puede alcanzar entre 0 y 100. |
| cmi.core.score.min | Corresponde al mínimo puntaje que el estudiante puede alcanzar entre 0 y 100. |
| cmi.core.total\_time | Utilizado para mantener un seguimiento del tiempo total empleado en cada sesión del SCO, indica el tiempo acumulado de todas las sesiones del SCO. |
| cmi.core.session\_time | Utilizado para mantener un seguimiento del tiempo empleado en una sesión con el SCO, indica la cantidad de tiempo que el estudiante pasó en el SCO hasta dejarlo. |

1. “cmi.comments”. El modelo de datos de SCORM presenta este apartado como un mecanismo para la obtención y distribución de comentarios para un SCO.

Tabla 6: Elementos del modelo de datos de SCORM (cmi.comments) para evaluación.

|  |  |
| --- | --- |
| cmi.comments | Representa la retroalimentación por parte del estudiante, comentarios que se envían al LMS sobre el SCO. |
| cmi.comments\_from\_lms | Representa los comentarios sobre el SCO originados en el LMS. |

1. “cmi.objectives”. El apartado de objetivos identifica la forma en que el estudiante realiza objetivos individuales contemplados dentro de un SCO.

Tabla 7: Elementos del modelo de datos de SCORM (cmi.objectives) para evaluación.

|  |  |
| --- | --- |
| cmi.objectives.\_count | Utilizado para determinar la cantidad de objetivos del SCO que el estudiante debería alcanzar. |
| cmi.objectives.n.score.raw | Indica la representación numérica del rendimiento del estudiante posterior a cada intento en el objetivo “n”, la cual es un valor entre 0 y 100. |
| cmi.objectives.n.score.max | Corresponde al puntaje máximo que el estudiante puede alcanzar en el objetivo entre 0 y 100. |
| cmi.objectives.n.score.min | Corresponde al puntaje mínimo que el estudiante puede alcanzar en el objetivo entre 0 y 100. |
| cmi.objectives.n.status | Utilizado para mantener un seguimiento del estatus del estudiante para un objetivo dado, representa el estatus del objetivo del SCO obtenido en cada intento, donde éste puede haber sido no intentado, solo mostrado en pantalla, no completado, fallado, completado o pasado. |

1. “cmi.student\_data”. Este apartado del modelo de datos incluye la información necesaria para soportar la personalización de un SCO en base al rendimiento del estudiante.

Tabla 8: Elementos del modelo de datos de SCORM (cmi.student\_data) para evaluación.

|  |  |
| --- | --- |
| cmi.student\_data.mastery\_score | Indica el puntaje de aprobación del SCO. Si el puntaje obtenido es mayor o igual a este valor, se considera al estudiante como probado. |
| cmi.student\_data.max\_time\_allowed | Representa la cantidad de tiempo permitida para el estudiante en cada intento con el SCO. |
| cmi.student\_data.time\_limit\_action | Indica lo que el SCO debe hacer cuando se sobrepasa el máximo tiempo permitido, como continuar o no con el SCO y la muestra de mensajes. |

1. “cmi.interactions”. Todos sus elementos se relacionan con entradas de datos del estudiante reconocidas y recordables.

Tabla 9: Elementos del modelo de datos de SCORM (cmi.interactions) para evaluación.

|  |  |
| --- | --- |
| cmi.interactions.n.objectives.\_count | Utilizado para determinar la cantidad de objetivos de la interacción “n” que el estudiante debería alcanzar. |
| cmi.interactions.n.time | Corresponde al momento en el cual la interacción “n” del estudiante se completó, en horas. |
| cmi.interactions.n.type | Indica la categoría de interacción “n” que se almacena, donde ésta puede ser una pregunta de verdadero – falso, de opción, textual, de relacionar, opción múltiple, de secuencia, o numérica. |
| cmi.interactions.n.correct\_responses  .\_count | Utilizado para determinar la cantidad de respuestas correctas almacenadas para la interacción “n”. |
| cmi.interactions.n.weighting | Representa la importancia relativa de una interacción “n” frente a las demás, en la forma de un valor numérico. Dependiendo de este valor, la interacción puede ser o no tomada en cuenta para el puntaje final. |
| cmi.interactions.n.student\_response | Indica la respuesta del estudiante a una interacción “n”. |
| cmi.interactions.n.result | Corresponde al resultado de las respuestas del estudiante a una interacción “n”, la cual puede ser inesperada, incorrecta, neutral o correcta. |

### Experience API.

Experience API, xAPI o Tin Can API es un estándar de ADL (al igual que SCORM) que habilita a componentes de software para comunicarse con otros y compartir información (Kelly & Thorn, 2013). Rustici indica que xAPI “es una nueva especificación para tecnologías de aprendizaje que hace posible recolectar datos sobre un amplio rango de experiencias que tiene una persona”, es simple, flexible e interoperable.

A menudo se considera a xAPI como la evolución de SCORM, un SCORM 2.0, sin embargo, es por si misma una nueva definición con un nuevo proceso para el manejo de información, que permite la recolección, distribución y recuperación de información relativa a las experiencias de los usuarios (Zapata-Rivera & Petrie, 2018), según Rustici, xAPI captura información de las actividades de una persona o grupo proveniente de varias tecnologías de forma consistente. xAPI ha evolucionado rápidamente durante los últimos años siendo implementado en muchos sistemas incluso fuera del contexto de la instrucción (Zapata-Rivera & Petrie, 2018).

Kelly & Thorn (2013) menciona que algunos diseñadores consideran que SCORM no es más que un formato de archivo necesario para que un LMS acepte el contenido de aprendizaje, pues los datos de rastreo que este dicta tienen algunas limitaciones. xAPI proporciona un repositorio de información que permite la integración de aplicaciones mediante el acceso al mismo (Zapata-Rivera & Petrie, 2018), según Rustici, las aplicaciones se comunican de forma segura capturando y compartiendo flujos de actividades hacia el repositorio, mediante un vocabulario común. xAPI puede rastrear prácticamente todo (Kelly & Thorn, 2013).

Para xAPI, las actividades de aprendizaje generan declaraciones de actividad, mismas que resultan en flujos de actividad almacenados en el sistema de almacenamiento o repositorio llamado LRS (almacén de registros de aprendizaje), el cual puede recoger datos desde cualquier ambiente de aprendizaje, y a diferencia de SCORM, ser independiente o parte de un LMS (Taamallah & Khemaja, 2014). Manzo Vazquez et al. (2015) establece que xAPI “tiene un vocabulario simple capaz de capturar y compartir el flujo de actividades”, el cual se basa en declaraciones en la forma de ‘sustantivo, verbo y objeto’ que son almacenadas en el LRS.

Rustici indica que cuando las actividades del usuario necesitan ser almacenadas, la aplicación envía las declaraciones de forma segura al LRS de la forma antes mencionada, en formato JSON (notación de objeto de javascript). Una declaración puede enviarse con solo tres partes codificadas: sustantivo, verbo y objeto, sin embargo, es el caso más simple (Manzo Vazquez et al. 2015), la tabla 10 muestra las partes que puede contener una declaración.

Tabla 10: Elementos de la declaración de actividad de xAPI.

|  |  |
| --- | --- |
| **Nombre** | **Descripción** |
| UUID | Identificación generada. |
| Actor | Quien realiza la acción. Puede ser un agente o grupo. Contiene el nombre y algo adicional que identifique al actor. |
| Verbo (Verb) | Identifica la acción. Contiene un identificador (URL) que termina con el verbo y el verbo (se recomienda una versión para cada lenguaje). |
| Objeto (Object) | El objeto es lo que el actor experimenta. Puede ser una actividad que contiene un tipo, una descripción y un URL. |
| Contexto (Context) | Objeto que permite definir el contexto de la acción. Todos sus elementos son opcionales. |
| Result (Resultado) | Definición de resultados (si aplica). Contiene campos como completion, success, duration, score y extensions. |
| Timestamp | Almacena la fecha y el tiempo en el que se creó la declaración. |
| Autoridad, almacenado y versión (Authority, stored and version) | Tres objetos definidos por el LRS. Quien afirma la declaración (autoridad), si es un agente o grupo (almacenado) y la versión del API (versión). |
| Extensiones (Extensions) | Si existe una definición de contexto y resultado, se definen propiedades abiertas con URL’s. |
| Adjuntos (Attachments) | Permite definir una lista de archivos adjuntos a la declaración |
| **Nota:** Fuente: Development of a xAPI Application Profile for Self-Regulated Learning. | |

Zapata-Rivera & Petrie (2018) menciona que mediante el uso de xAPI y un LRS, es posible almacenar casi cualquier cosa con la forma de ‘sustantivo, verbo y objeto’, que existiendo este repositorio común los usuarios pueden compartir sus experiencias a otras organizaciones o instituciones, y que cualquier dispositivo es capaz de enviar declaraciones de xAPI teniendo o no una conexión a internet. Las librerías de xAPI se encuentran disponibles para varias tecnologías de cliente y servidor. Los paquetes más utilizados son los de JavaScript y PHP.

## ICONIX: Metodología de desarrollo de software.

ICONIX es una metodología de desarrollo de software que ofrece un uso simplificado de UML (unified modeling language), pues incluye solo un pequeño conjunto de diagramas para el análisis y el diseño (Mukandla & Dwolatzky, 2004), es una metodología conducida por casos de uso (Thakare et al. 2012) que permite llevarlos a código rápida y eficientemente (Mukandla & Dwolatzky, 2004). Ofrece también una muy buena trazabilidad haciendo posible regresar a los requerimientos en todo momento pasando por una serie de modelos, nunca permite alejarse demasiado de las necesidades (Mukandla & Dwolatzky, 2004).

A pesar de ser una metodología anterior a otras como XP (extreme programing) y al desarrollo de software ágil en sí (Thakare et al. 2012), Topper & Horner (2013) establece que ICONIX “es un proceso ligero que soporta el desarrollo ágil”, presenta un enfoque iterativo e incremental donde las iteraciones se establecen por el grupo de casos de uso que serán desarrollados (Mukandla & Dwolatzky, 2004), proporciona flexibilidad y respuesta al cambio (Topper & Horner, 2013).

Los diagramas UML incluidos ayudan en la consistencia y organización del proceso (Topper & Horner, 2013), así como en el análisis de requerimientos y la documentación de diseño (Thakare et al. 2012). ICONIX es un proceso de rigor que permite la iteración y soporta el desarrollo en amplitud, pues hace mucho énfasis en entender el dominio del problema y no permite avanzar con otros modelos hasta haber asegurado el entendimiento por parte del equipo (Topper & Horner, 2013).

Como se observa en la ilustración 3 (Rosenberg & Stephens, 2013), el proceso inicia con prototipos de la interfaz gráfica (GUI) del sistema a desarrollar, posteriormente se encuentra el modelo dinámico, conformado por el modelo de casos de uso, diagramas de robustez y diagramas de secuencia, y el modelo estático, formado por el modelo de dominio y modelo de clases. Al final, después de los modelos dinámico y estático, la metodología termina con planes de prueba, pruebas unitarias y la implementación del diseño, es decir, el código de la aplicación.



Ilustración 3: Estructura de ICONIX.

Ahora bien, ICONIX se compone de cuatro etapas secuenciales, cada una revisada y actualizada al iniciar la siguiente: análisis de requerimientos, diseño preliminar, diseño detallado y despliegue (Thakare et al. 2012). Para la primera etapa, análisis de requerimientos, la metodología propone iniciar con prototipos para la GUI y el modelado del dominio (que identifica los objetos del mundo real relacionados con el dominio del problema), elementos que permitirán identificar los casos de uso (Mukandla & Dwolatzky, 2004) para el modelo de casos de uso, último elemento del análisis de requerimientos.

Topper & Horner (2013) explica que el modelo de dominio tiene el propósito de capturar el dominio del problema y mostrar las relaciones entre los elementos que lo conforman, este lleva a los involucrados a un entendimiento común al exponer los elementos del ambiente en amplitud y especificar sus relaciones. Una vez realizado el modelado del dominio, el diseñador continúa con el modelo de casos de uso, donde se listan las funciones del sistema y sus actores (usuarios) utilizando nombres descriptivos (Topper & Horner, 2013). Thakare et al. (2012) señala que los casos de uso son más fáciles de diseñar y probar en el proceso de ICONIX.

Los casos de uso capturan los requerimientos funcionales mediante texto simple haciéndolos entendibles para todos (Mukandla & Dwolatzky, 2004), sin embargo, una vez identificados, se procede a formalizar explícitamente las funcionalidades en descripciones de casos de uso que detallan cada una de ellas en la forma de actividades en el sistema (Topper & Horner, 2013). En este punto del proceso los modelos dinámico y estático se relacionan por primera vez, según Topper & Horner (2013), el modelo de dominio y el modelo de casos de uso son dependientes el uno del otro, pues al modelar los casos de uso se descubrirán y se agregarán nuevas entidades del dominio.

Para la segunda etapa: diseño preliminar, la distinción de ICONIX es que hace uso del análisis de robustez (diagramas de robustez), con el cual se construye un puente entre el análisis y el diseño (Thakare et al. 2012), se trata de elementos que traducen las descripciones de casos de uso a representaciones gráficas con entidades correspondientes al modelo de dominio y procesos entre estas. El diseño preliminar busca el descubrimiento de entidades (Rosenberg & Stephens, 2013), según Topper & Horner (2013), en este punto se actualiza el modelo de dominio, se enfoca al problema de análisis eliminando las entidades que no conciernen e identificando nuevas. El modelo estático se va refinando mediante las iteraciones del modelo dinámico (Mukandla & Dwolatzky, 2004).

Una vez realizado y revisado el análisis de robustez sigue la etapa de diseño detallado, para la cual las descripciones de casos de uso deben estar completas, correctas, detalladas y explícitas, los casos de uso deberán estar en un estado a partir del cual pueda realizarse un diseño detallado (Rosenberg & Stephens, 2013). De acuerdo con Rosenberg & Stephens (2013), el diseño detallado trata sobre alojar el comportamiento, se alojan las funciones de software identificadas (procesos de los diagramas de robustez) en las entidades de dominio, al igual que se agregan atributos (Topper & Horner, 2013), pasando de un modelo de dominio a un modelo de clases.

El diseño detallado es conducido por diagramas de secuencia que añaden detalle al diseño preliminar, aumentan la precisión. Los diagramas de secuencia tienen 3 metas en ICONIX: alojar el comportamiento, detallar la interacción de las clases en la vida del caso de uso y finalizar la distribución de operaciones entre las clases (Rosenberg & Stephens, 2013). Un diagrama de secuencia muestra la interacción de las clases en todo el curso de un caso de uso, se incluyen las clases del diagrama de robustez asociado al caso de uso y sus procesos (funciones de las clases) se convierten en mensajes (Rosenberg & Stephens, 2013). Al igual que en el diseño preliminar, cualquier nuevo comportamiento identificado se agregará al modelo de clases.

En la cuarta y última etapa, despliegue, se realiza primero el diagrama de despliegue de la aplicación, modelo que según la OMG (2019) especifica la arquitectura de ejecución de los sistemas, capturando los elementos físicos y/o lógicos del sistema modelado, las relaciones entre dichos elementos y los artefactos de software e información asignados a estos. Posteriormente, se escriben casos de prueba unitaria, y con base en los diagramas de clases y secuencia, finalmente se escribe el código de la aplicación (Thakare et al. 2012).

En conclusión, ICONIX es una metodología de desarrollo de software fuerte en cuanto a diseño y análisis se refiere, proporciona un alto nivel de orientación a objetos enfocado a diseño pudiendo ser aplicada en el desarrollo de cualquier sistema (incluso crítico) en el que se utilice la orientación a objetos, a pasear de ser esta su única limitación, pues antes de emplear la metodología, se debe instruir al equipo de desarrollo en la tecnología de orientación a objetos (Mukandla & Dwolatzky, 2004).

# Capítulo 2: Requerimientos de software.

En este capítulo se presenta una especificación de requerimientos de software con base en el estándar IEEE Std. 830 – 1998, la cual funge como un apoyo en el análisis de requerimientos de la metodología de desarrollo de software empleada: ICONIX. Se ofrece una perspectiva general, se especifican las funciones del producto según sus usuarios, se listan los elementos de evaluación requeridos y se establecen las características de los usuarios, así como las restricciones identificadas para el desarrollo del proyecto.

* 1. Introducción.

En este capítulo se presentan los requerimientos del sistema en base al formato de especificación de requerimientos de software del estándar IEEE Std. 830-1998. Cabe mencionar que, considerando el proyecto e importancia del contexto de su desarrollo, fueron seleccionados algunos elementos de dicho estándar que permiten delimitar sus requerimientos y contexto de forma clara, la especificación de requerimientos de software aquí presentada no es una fiel implementación del estándar ya mencionado.

* + 1. Propósito.

Una especificación de requerimientos de software (ERS) busca un acuerdo entre la parte que tiene una necesidad de software y aquella de se encargará de diseñar y desarrollar el producto como una solución, “la ERS establece las funciones y capacidades que un sistema de software debe proporcionar, sus características, y las restricciones que este debe respetar” (Wiegers & Beatty, 2013).

Las funciones de un producto de software se documentan primeramente para fungir como fuente de consulta durante las siguientes etapas de la ingeniería de software, la ERS puede entregarse a los diseñadores, desarrolladores, *testers*, clientes y demás interesados con distintos fines. Para el caso específico, se presenta una ERS para establecer debidamente los requerimientos del sistema y hacerlos disponibles para que otros alumnos, profesores, etc. Puedan revisar y extender o mejorar el trabajo realizado en un futuro.

* + 1. Ámbito del sistema.

En el programa educativo de Redes y Servicios de Cómputo de la Facultad de Estadística e Informática en la Universidad Veracruzana, necesitan de un sistema de cómputo que soporte la administración y presentación de laboratorios virtuales, así como permita la consulta de información estadística relativa al uso de los laboratorios por parte de los alumnos, ya que existen algunas dificultades para realizar prácticas de laboratorio reales.

En términos generales, se necesita que el sistema de soporte presente una manera sencilla de subir laboratorios virtuales específicos, y permita que éstos sean visualizados y experimentados posteriormente por los alumnos. Un profesor o administrador deberá tener la capacidad de integrar nuevos laboratorios y probarlos, para que después los alumnos puedan seleccionarlos y experimentarlos. Al final, tanto los profesores como los administradores deberán acceder a información estadística de uso y seguimiento de avance por alumno, correspondiente a los laboratorios que éstos realicen.

También se contempla que los laboratorios añadidos ya sea por un administrador o un profesor puedan ser reemplazados, modificados. Dicha acción podrá realizarse siempre y cuando ningún alumno haya experimentado el laboratorio, puesto que, de lo contrario, la información almacenada previa a la modificación no será congruente con la información de nuevas interacciones.

La información estadística que se requiere corresponde a dos aspectos: primero, al uso de la plataforma. Es necesario conocer quién y cuándo experimenta con los laboratorios, es decir, saber la fecha y la hora en que un usuario determinado ingresa a la plataforma y trabaja en él, pues se debe verificar que los alumnos se encuentren realizando el trabajo necesario para adquirir las competencias requeridas en su formación.

El segundo aspecto hace referencia al seguimiento del avance del alumno. Cada laboratorio dentro tendrá una función específica, los alumnos interactuarán con ellos de acuerdo con su función realizando una serie de acciones y actividades; los profesores necesitan conocer el avance de sus estudiantes por lo cual se requiere conocer información de tiempo y respuesta, misma que se presentará como estadísticas de avance.

Se contempla la posibilidad de que la información de seguimiento pueda brindar pautas a los profesores para ubicar temas de interés, temas complicados para los alumnos, estos puedan dar retroalimentación y reforzar el conocimiento necesario para cumplir los objetivos incluidos en los escenarios plasmados en los laboratorios.

* 1. Descripción general.
     1. Perspectiva del producto.

El sistema se piensa como una extensión a un LMS, un complemento que permitirá realizar un seguimiento específico al avance de los alumnos y los grupos, mismo que será posible mediante el uso del estándar de SCORM. Según la documentación técnica de SCORM, los LMS se encargan de rastrear el progreso y evaluar el dominio del alumno, sin embargo, la información general proporcionada por los LMS puede estar limitada a calificaciones y actividades, por lo que el complemento presentará información de seguimiento específica extraída del modelo de datos de SCORM.

A este respecto, el complemento se encargará de obtener, procesar y mostrar la información de seguimiento de los alumnos. La información en cuestión ingresará al LMS mediante SCORM, cada laboratorio virtual particular enviará datos al LMS utilizando el API de SCORM, y una vez que estos se encuentren almacenados en la base de datos del LMS, el complemento los buscará para poder presentarlos en las estadísticas de uso y seguimiento de avance. El siguiente diagrama de contexto ilustra este proceso.



Ilustración 4: Diagrama de contexto.

El diagrama de contexto en la ilustración 5 engloba el complemento y el paquete de SCORM (laboratorio virtual) dentro del LMS (nombrado LMS extendido), puesto que toda la interacción de los usuarios será mediante el LMS en sí, ya que solo es una extensión al mismo. Los alumnos acceden a los laboratorios desde el LMS, así como los profesores y el administrador consultan los laboratorios y estadísticas de avance en éste.

Pensando en la diversidad de laboratorios y sus particularidades, los elementos de uso y seguimiento que deberá manejar el complemento son los siguientes:

1. Entrada y salida de la plataforma.
2. Laboratorios realizados.
3. Repeticiones de los laboratorios
4. Tiempo de experimentación de laboratorios.
5. Porcentaje de avance.
6. Calificación del alumno.
   * 1. Funciones del producto.

El complemento proporcionará funcionalidades para administrar los laboratorios virtuales y consultar estadísticas de avance de los alumnos, que se listan a continuación.

1. El complemento soportará la eliminación de laboratorios virtuales.
2. El complemento facilitará la consulta de laboratorios virtuales propios del profesor.
3. El complemento soportará la consulta de avances por cada alumno y por grupo.
4. El complemento facilitará la consulta de todos los laboratorios virtuales incluidos.
5. El complemento soportará la consulta de avances de todos los alumnos y grupos registrados.
6. El complemento soportará la consulta de avances personales.
   * 1. Características de los usuarios.

El complemento será empleado por profesores y alumnos de la Licenciatura en Redes y Servicios de cómputo de la Facultad de Estadística e Informática en la Universidad Veracruzana.

* + - 1. Profesores.

1. Son catedráticos en el área de redes con estudios superiores y de posgrado.
2. Están familiarizados con el empleo de computadoras y diversos sistemas operativos.
3. Están familiarizados con el uso de tecnologías web tales como LMS.
   * + 1. Administrador.
4. Es catedrático en el área de redes con estudios superiores y de posgrado.
5. Está familiarizado con el empleo de computadoras y diversos sistemas operativos.
6. Está familiarizado con el uso de tecnologías web tales como LMS.
7. Es coordinador de academia.
   * + 1. Alumnos.
8. Estudian la licenciatura de Redes y Servicios de cómputo.
9. Están familiarizados con el empleo de computadoras y diversos sistemas operativos.
10. Están familiarizados con el uso de tecnologías web tales como LMS.
    * 1. Restricciones.

A continuación, se presentan las limitaciones sobre el desarrollo del complemento.

1. El LMS seleccionado junto al complemento, serán desplegados en él o los servidores dispuestos por la Facultad de Estadística e Informática.
2. Al tratarse de una extensión a un LMS, el complemento se desarrollará siguiendo las reglas de desarrollo, estándares, bases de datos y API’s proporcionadas o utilizadas por Moodle.
3. El complemento se desarrollará en el o los lenguajes de programación en los que el LMS seleccionado esté desarrollado.
   1. Requisitos específicos.
      1. Interfaces externas.

Para ir en acuerdo con el LMS seleccionado, la interfaz gráfica de usuario (GUI) del complemento seguirá los estándares de este para el diseño de su GUI.

Con respecto a las interfaces de comunicaciones, como se menciona en el apartado perspectiva del producto (2.2.1) de este capítulo, un requerimiento es el manejo de información de uso y seguimiento del avance de los alumnos, misma que se obtendrá del LMS seleccionado, el cual, a su vez, la obtendrá mediante el estándar de SCORM. Esta información se limitará únicamente a los elementos del modelo de datos de SCORM que la manejen, puesto que, los laboratorios virtuales al igual solo enviarán la información soportada por SCORM.

* + 1. Funciones.

A continuación, se presentan las descripciones de casos de uso correspondientes al modelo mostrado en el apartado funciones del producto (2.2.2) de este capítulo, organizadas por tipo de usuario.

* + - 1. Profesor.

Tabla 11: CU 01 - Consultar laboratorios propios.

|  |  |
| --- | --- |
| **Nombre del caso de uso.** | CU 01 – Consultar laboratorios propios. |
| **Actor.** | Profesor. |
| **Descripción.** | En este caso de uso, el profesor accede a los laboratorios existentes de sus cursos. |
| **Precondiciones.** | 1. Debe existir al menos un laboratorio registrado del profesor. |
| **Postcondiciones.** | N/A. |
| **Flujo normal.** | 1. El profesor selecciona ver laboratorios. 2. El sistema obtiene los laboratorios asociados al profesor, los carga y muestra una página con el listado de laboratorios. |
| **Flujos alternos.** | No existen laboratorios registrados.  2.1. El sistema muestra un mensaje indicando que no existen laboratorios. |
| **Excepciones.** | No hay conexión con el servidor (en paso 2).   1. El sistema muestra un mensaje indicando los problemas de conexión con el servidor.   Error al obtener los datos (en paso 2).   1. El sistema muestra un mensaje indicando el error encontrado. |

Tabla 12: CU 04 - Consultar avances por grupo.

|  |  |
| --- | --- |
| **Nombre del caso de uso.** | CU 04 – Consultar avances por grupo. |
| **Actor.** | Profesor. |
| **Descripción.** | En este caso de uso, el profesor visualiza los avances de los alumnos de uno de sus grupos. |
| **Precondiciones.** | 1. Debe haber al menos 1 alumno registrado en el grupo. |
| **Postcondiciones.** | N/A. |
| **Flujo normal.** | 1. El profesor selecciona las opciones de un grupo y selecciona avances. 2. El sistema consulta si existen alumnos registrados en el grupo, obtiene los laboratorios asociados al grupo, obtiene la información de uso y seguimiento correspondiente al porcentaje de realización, tiempo de experimentación, porcentaje de avance y calificación para cada laboratorio, carga la información y muestra la página de estadísticas de grupo. |
| **Flujos alternos.** | No existen alumnos registrados.  2.1. El sistema muestra un mensaje indicando que no hay alumnos registrados en el grupo. |
| **Excepciones.** | No hay conexión con el servidor (en paso 2).   1. El sistema muestra un mensaje indicando los problemas de conexión con el servidor.   Error al obtener los datos (en paso 2).   1. El sistema muestra un mensaje indicando el error encontrado. |

Tabla 13: CU 03 - Consultar avances por alumno.

|  |  |
| --- | --- |
| **Nombre del caso de uso.** | CU 03 – Consultar avances por alumno. |
| **Actor.** | Profesor. |
| **Descripción.** | En este caso de uso, el profesor visualiza los avances de uno de sus alumnos. |
| **Precondiciones.** | 1. Deben existir registros de al menos un alumno. |
| **Postcondiciones.** | N/A. |
| **Flujo normal.** | 1. El profesor selecciona las opciones de un alumno y selecciona avances. 2. El sistema obtiene los laboratorios asociados al alumno, consulta si han sido realizados, obtiene la información de uso y seguimiento correspondiente al porcentaje de avance, calificación, tiempo de experimentación y repeticiones, para cada laboratorio, carga la información y muestra la página de estadísticas de alumno. |
| **Flujos alternos.** | No hay registros.  2.1. El sistema muestra un mensaje indicando que el alumno no tiene registros de avance en ningún laboratorio. |
| **Excepciones.** | No hay conexión con el servidor (en paso 2).   1. El sistema muestra un mensaje indicando los problemas de conexión con el servidor.   Error al obtener los datos (en paso 2).   1. El sistema muestra un mensaje indicando el error encontrado. |

Tabla 14: CU 02 - Eliminar laboratorio.

|  |  |
| --- | --- |
| **Nombre del caso de uso.** | CU 02 – Eliminar laboratorio. |
| **Actor.** | Profesor. |
| **Descripción.** | En este caso de uso, el profesor elimina un laboratorio específico de sus cursos. |
| **Precondiciones.** | 1. Debe existir al menos un laboratorio registrado del profesor. |
| **Postcondiciones.** | 1. Se actualiza el catálogo de laboratorios para el administrador, el profesor y los alumnos. |
| **Flujo normal.** | 1. El profesor selecciona las opciones de un laboratorio y selecciona eliminar. 2. El sistema solicita confirmación de la baja. 3. El profesor selecciona sí. 4. El sistema solicita la baja del laboratorio y notifica al profesor el resultado de la operación. |
| **Flujos alternos.** | No desea eliminar.  3.1. El profesor selecciona no. |
| **Excepciones.** | No hay conexión con el servidor (en paso 4).   1. El sistema muestra un mensaje indicando los problemas de conexión con el servidor.   Error al eliminar (en paso 4).   1. El sistema muestra un mensaje indicando el error encontrado. |

* + - 1. Administrador.

Tabla 15: CU 05 - Consultar laboratorios.

|  |  |
| --- | --- |
| **Nombre del caso de uso.** | CU 05 – Consultar laboratorios. |
| **Actor.** | Administrador. |
| **Descripción.** | En este caso de uso, el administrador accede a los laboratorios existentes. |
| **Precondiciones.** | 1. Debe existir al menos un laboratorio registrado. |
| **Postcondiciones.** | N/A. |
| **Flujo normal.** | 1. El administrador selecciona ver laboratorios. 2. El sistema obtiene todos los profesores registrados, obtiene los laboratorios asociados a cada uno, los carga y muestra una página con el listado de laboratorios por profesor. |
| **Flujos alternos.** | No existen laboratorios registrados.  2.1. El sistema muestra un mensaje indicando que no existen laboratorios registrados. |
| **Excepciones.** | No hay conexión con el servidor (en paso 2).   1. El sistema muestra un mensaje indicando los problemas de conexión con el servidor.   Error al obtener los datos (en paso 2).   1. El sistema muestra un mensaje indicando el error encontrado. |

* + - 1. Alumno.

Tabla 16: CU 06 - Consultar avances personales.

|  |  |
| --- | --- |
| **Nombre del caso de uso.** | CU 06 – Consultar avances personales. |
| **Actor.** | Alumno. |
| **Descripción.** | En este caso de uso, el alumno visualiza sus avances. |
| **Precondiciones.** | 1. Deben existir registros de al menos un laboratorio. |
| **Postcondiciones.** | N/A. |
| **Flujo normal.** | 1. El alumno selecciona avances. 2. El sistema obtiene los laboratorios asociados al alumno, consulta si han sido realizados, obtiene la información de uso y seguimiento correspondiente al porcentaje de avance, calificación, tiempo de experimentación y repeticiones, para cada laboratorio, carga la información y muestra la página de estadísticas de alumno. |
| **Flujos alternos.** | No hay registros.  2.1. El sistema muestra un mensaje indicando que no existen registros de avance. |
| **Excepciones.** | No hay conexión con el servidor (en paso 2).   1. El sistema muestra un mensaje indicando los problemas de conexión con el servidor.   Error al obtener los datos (en paso 2).   1. El sistema muestra un mensaje indicando el error encontrado. |

# Capítulo 3: Análisis de requerimientos.

En este capítulo se hace referencia a la primera etapa de la metodología de desarrollo de software ICONIX. Se presentan los artefactos de análisis propuestos por la metodología tales como los prototipos de interfaz de usuario, el modelo de dominio y el modelo de casos de uso.

## Prototipos de la interfaz de usuario.

Tal como lo marca la etapa de análisis de requerimientos de ICONIX, se realizaron prototipos de la interfaz de usuario, mismos que representan un punto de partida para la realización de modelos posteriores como el modelo de dominio y de casos de uso. Los prototipos del complemento se muestran a continuación.

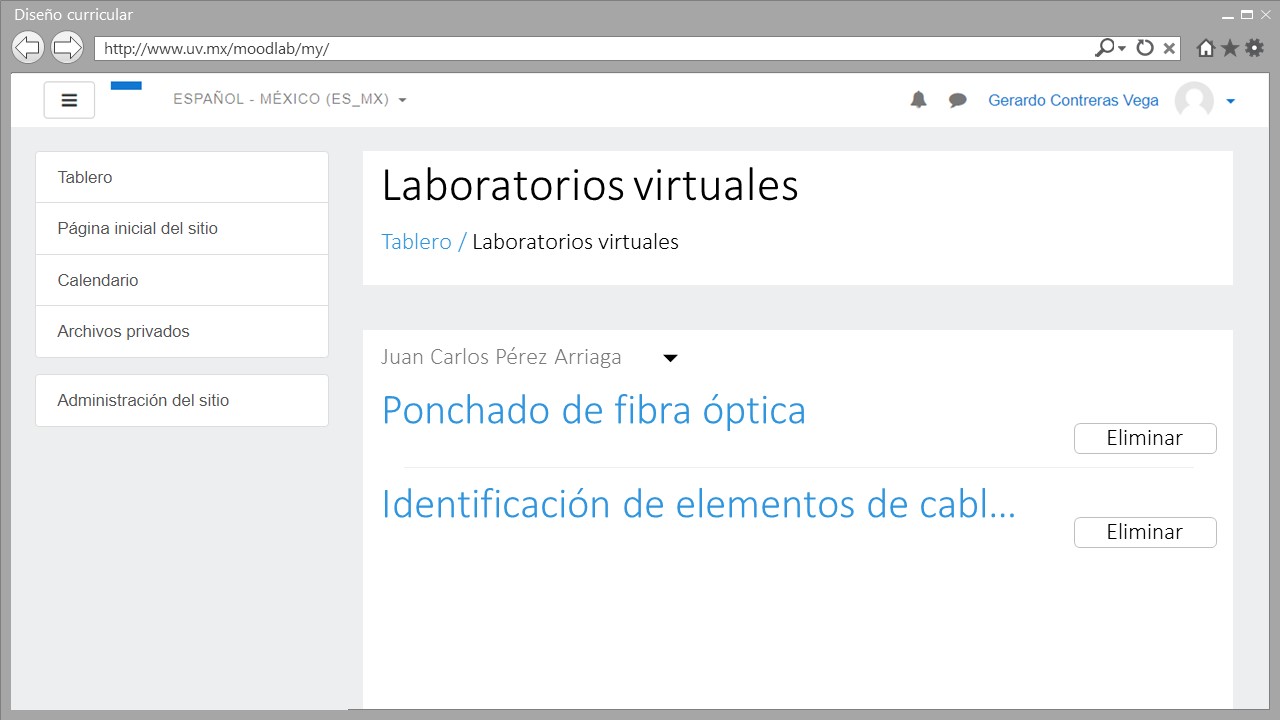


Ilustración 5: Pantalla de consulta de laboratorios por parte del administrador.

En la ilustración 5 puede apreciarse una vista de la página principal de Moodle (tablero), más una sección nombrada “Laboratorios virtuales”, donde se observa un listado de laboratorios virtuales con la opción de “eliminar” en cada uno, organizados según el profesor al cual están asociados pudiendo seleccionar a otros profesores mediante el combo.

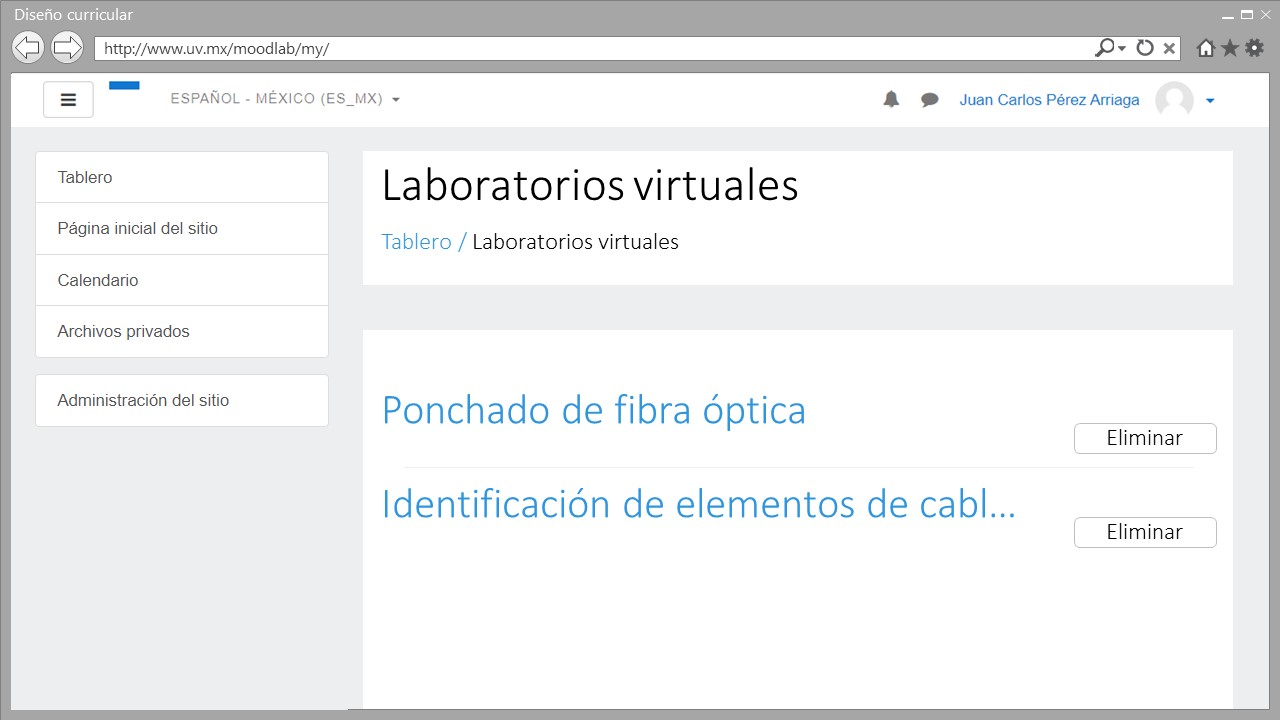


Ilustración 6: Pantalla de consulta de laboratorios por parte de un profesor.

Al igual que en la ilustración 5, la ilustración 6 muestra un listado de laboratorios virtuales con una opción para “eliminar” en cada uno, sin embargo, la diferencia aquí es que no existe el combo de selección de profesores, pues esta pantalla presenta los laboratorios virtuales propios del profesor que hace la consulta.

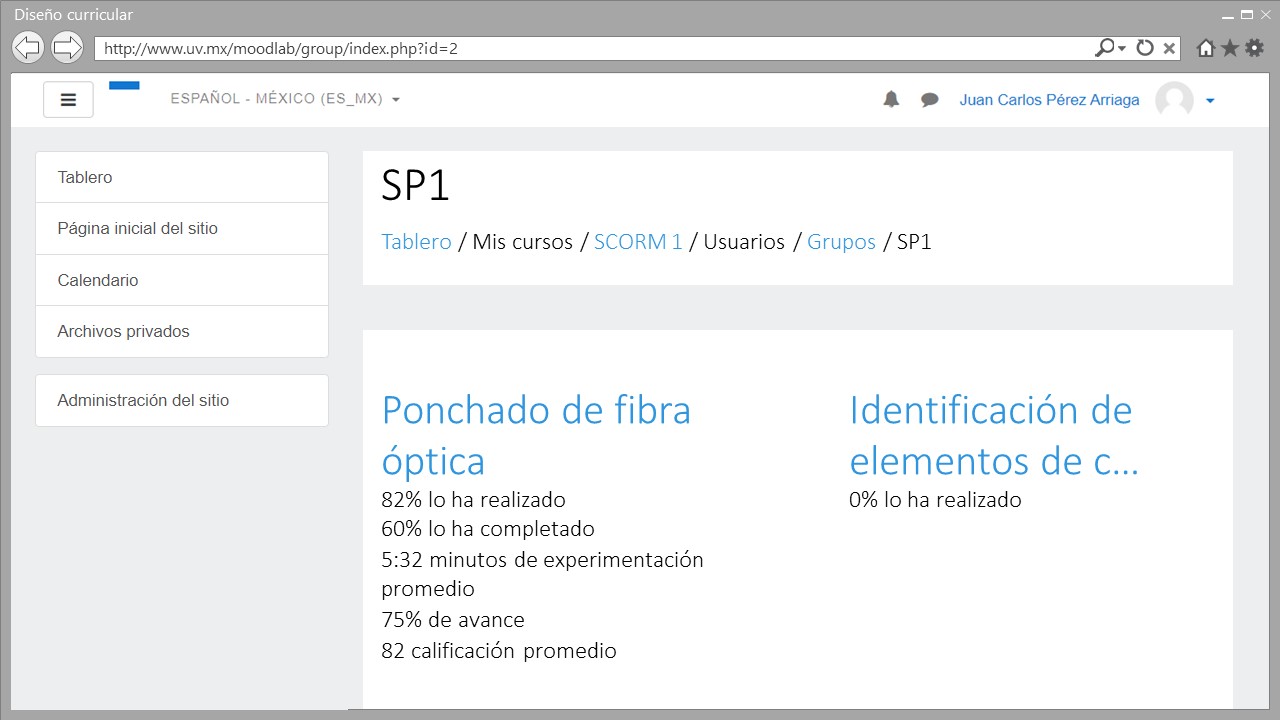


Ilustración 7: Pantalla de consulta de estadísticas de grupo.

Dejando atrás la consulta de laboratorios virtuales, en la ilustración 7 puede observarse una sección nombrada “SP1” que incluye dos laboratorios virtuales, así como información estadística relacionada con la realización del laboratorio, el tiempo de experimentación promedio, el avance y la calificación promedio de cada laboratorio correspondiente a un grupo de alumnos en específico.



Ilustración 8: Pantalla de consulta de estadísticas de un alumno por parte de un profesor.

En la ilustración 8 se aprecia una sección nombrada “Estadísticas”, misma que contiene dos laboratorios virtuales y su información estadística relativa al porcentaje de avance del laboratorio, la calificación, el tiempo de experimentación promedio y el número de repeticiones realizadas en cada uno de los laboratorios, correspondientes a un alumno en específico.

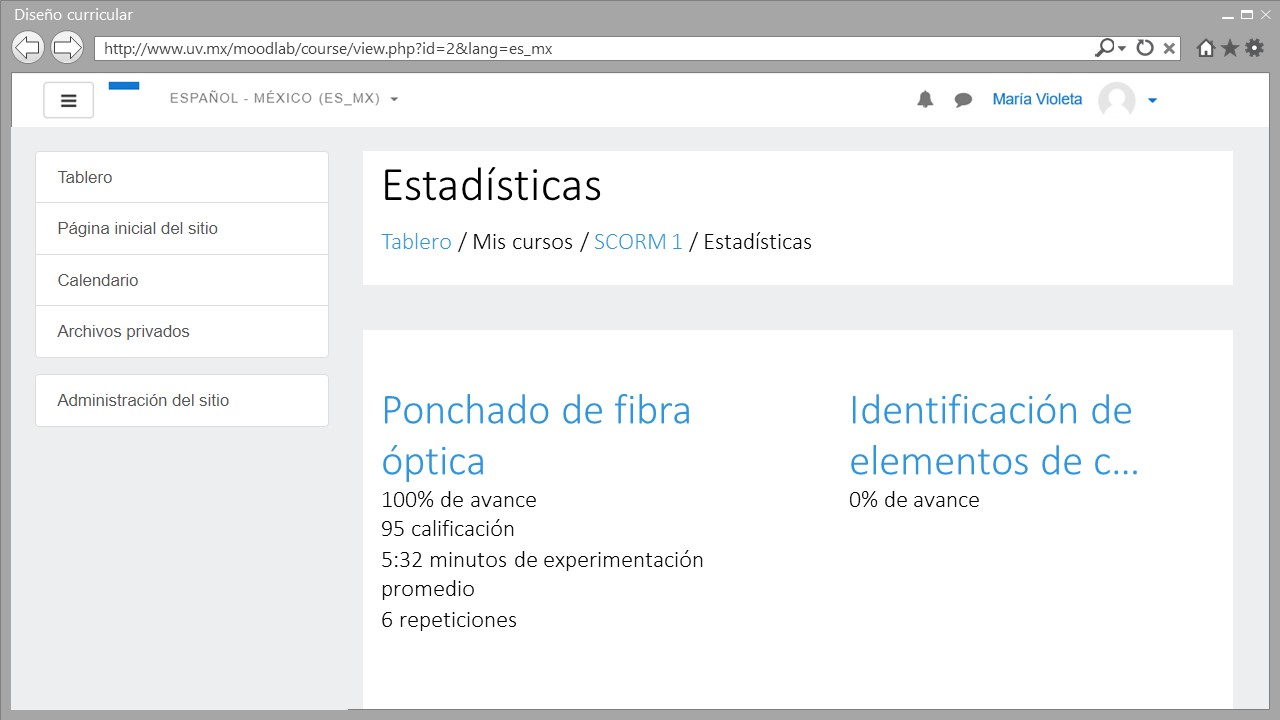


Ilustración 9: Pantalla de consulta de estadísticas personales por parte de un alumno.

La ilustración 9 presenta dos laboratorios virtuales con la misma información estadística de la ilustración 8, con la diferencia que aquí se muestran los laboratorios virtuales propios del alumno que realiza la consulta.

## Modelos del análisis de requerimientos.

A continuación, se presentan los modelos posteriores al diseño de prototipos para la interfaz de usuario: el modelo de dominio y el modelo de casos de uso.



Ilustración 10: Modelo de dominio.

En este punto de la etapa de análisis de requerimientos se procede con el modelo de dominio, con el cual se representan los objetos (entidades) del mundo real relacionados al dominio del problema de análisis, en el modelo de la ilustración 10 se aprecian las entidades de profesor, administrador, alumno, laboratorio, curso, grupo, adaptadorApi y LMS. De igual manera se observan las relaciones entre las entidades:

Los profesores se encargan de impartir cursos a los alumnos asignados a estos mediante grupos, así como de subir a un curso los laboratorios virtuales que deberán realizar los alumnos. De igual forma se muestra que los cursos pertenecen a un LMS, al cual los laboratorios consultan información del modelo de datos de SCORM mediante un adaptador de API al que tienen acceso.



Ilustración 11: Modelo de casos de uso.

Representando a los actores (usuarios) y las acciones que tendrán dentro del complemento, el modelo de casos de uso en la ilustración 11 muestra a 3 actores: profesor, administrador y alumno, y seis casos de uso, cuatro correspondientes al profesor, uno para el administrador y uno para el alumno.

El profesor consultara y eliminará sus laboratorios virtuales, al igual que consultara el avance de cada grupo y cada alumno que tenga asociado. Puede apreciarse que el administrador se piensa como un profesor, con la particularidad de poder consultar todos los laboratorios registrados, a diferencia del profesor que únicamente tendrá acceso a los suyos. Por otro lado, el alumno solo podrá consultar sus propios avances.

# Capítulo 4: Diseño preliminar.

El presente capítulo, correspondiente a la segunda etapa de la metodología de desarrollo de software ICONIX, muestra el análisis de robustez realizado que incluye un diagrama de robustez para cada caso de uso, acompañado de una descripción de caso de uso.

En el disño preliminar, tomando como base el modelo de casos de uso y las descripciones de casos de uso, los diagramas de robustez presentan la misma descripción mediante un modelo de análisis previo a el diseño detallado, en el cual se muestran los flujos del caso de uso utilizando actores, fronteras, entidades y controles, que hacen referencia a usuarios, pantallas, entidades de dominio y operaciones respectivamente.

En un diagrama de robustez, el actor puede exclusivamente comunicarse con fronteras, mismas que pueden solo comunicarse con controles, que a su vez pueden comunicarse con fronteras, otros controles y entidades, las cuales pueden comunicarse únicamente con controles. Además, se recomienda tener la descripción del caso de uso a la izquierda para facilitar la lectura del diagrama.

A continuación, se muestran los diagramas de robustez realizados para cada caso de uso. Cabe destacar que el flujo normal se denota dentro del modelo utilizando conectores en color negro, y rojo para los flujos alternos.



Ilustración 12: Diagrama de robustez CU 01 – Consultar laboratorios propios.

En la ilustración 12 se aprecia como el diagrama inicia con el profesor seleccionando “ver laboratorios” en la página principal del sitio, acto seguido, pasa a un control “mostrar laboratorios” que los obtiene de la entidad profesor, para despues cargarlos y desplegar una página con el listado de laboratorios. Para el flujo alterno, el control que pregunta si existen laboratorios continúa a desplegar un mensaje de notificación.



Ilustración 13: Diagrama de robustez CU 02 - Eliminar laboratorio.

El diagrama de eliminar laboratorio en la ilustración 13 inicia con el profesor seleccionando “eliminar” dentro de la página de visualización de laboratorios, se pasa al control de “confirmar baja” el cual despliega un mensaje de confirmación, el profesor confirma y el control continúa a “eliminar laboratorio”, que termina el flujo eliminando el laboratorio con la entidad “laboratorio”, o bien, regresa a la pagina de laboratorios si el profesor no confirma la baja en el flujo alterno.



Ilustración 14: Diagrama de robustez CU 03 - Consultar avances por alumno.

Para el diagrama de consultar avances por alumno en la ilustración 14, el profesor selecciona estadísticas de un alumno en la página de participantes de un curso, el flujo continúa al control de “mostrar estadísticas de alumno”, que obtiene los laboratorios del alumno con la entidad “profesor” y las estadísticas de cada laboratorio con la entidad “estadística”. Si hay registros de avance el flujo continúa al control que carga las estadísticas y despliega la página, de lo contrario, continúa al control que despliega un mensaje de notificación para el flujo alterno.



Ilustración 15: Diagrama de robustez CU 04 - Consultar avances por grupo.

En el diagrama de consultar avances por grupo en la ilustración 15, se aprecia como el profesor selecciona las estadísticas de un grupo en la página de grupos, posteriormente, el control “mostrar estadísticas de grupo” consulta si existen alumnos asociados al grupo, obtiene los laboratorios asociados de la entidad “profesor”, obtiene las estadísticas de cada laboratorio con la entidad “estadística”, las carga y despliega la página de estadísticas de grupo. Para el flujo alterno, si no hay alumnos asociados finaliza el flujo desplegando un mensaje de notificación.



Ilustración 16: Diagrama de robustez CU 05 - Consultar laboratorios.

En la ilustración 16 se observa como el diagrama de consultar laboratorios inicia con el administrador seleccionando los laboratorios en la página principal del sitio, después el flujo continúa hacia un control que obtiene los profesores registrados y los laboratorios de cada uno con la entidad “administrador”. Finalmente, el flujo sigue hacia otro control que carga los laboratorios y despliega una página con el listado, o continúa el flujo hacia desplegar un mensaje de notificación en caso de que no existan laboratorios registrados para el flujo alterno.



Ilustración 17: Diagrama de robustez CU 06 - Consultar avances personales.

Para el diagrama de consultar avances personales en la ilustración 17, el alumno selecciona estadísticas en la página principal del sitio, el flujo continúa al control de “mostrar estadísticas personales”, que obtiene los laboratorios del alumno con la entidad “profesor” y las estadísticas de cada laboratorio con la entidad “estadística”. Si hay registros de avance el flujo continúa al control que carga las estadísticas y despliega la página, de lo contrario, continúa al control que despliega un mensaje de notificación para el flujo alterno.

# Capítulo 5: Diseño detallado.

Este capítulo presenta los modelos de diseño propuestos por la metodología de desarrollo de software ICONIX, correspondientes a la tercera etapa de la metodología: diseño detallado. Se incluyen el diagrama de clases y un diagrama de secuencia para cada caso de uso, acompañado de una descripción de caso de uso.

## Diagrama de clases.



Ilustración 18: Diagrama de clases.

Para esta etapa de ICONIX el análisis de robustez ya ayudó a pulir el dominio del problema, descubriendo cuáles son las nuevas entidades a considerar y cuáles son las que no forman parte del dominio. Por lo tanto, en el diseño detallado el modelo de dominio se convierte en un diagrama de clases, con las entidades resultantes del proceso anterior y un comportamiento ya alojado en forma de operaciones de las clases.

En el diagrama de clases de la ilustración 18, puede apreciarse que las entidades que permanecen son: profesor, administrador, laboratorio y estadística como una clase abstracta implementada por “EstadisticaBasica”, pensada así para el caso en el cual se quiera procesar la información estadística de una forma distinta. Se agrega la entidad “ElementoScorm” para obtener los datos relacionados con elementos del modelo de datos de SCORM, y adicionalmente, se añaden cuatro interfaces: IProfesor, ILaboratorio, IAdministrador e IScorm para el acceso a datos, que estarían relacionadas con la clase “moodle\_database” de Moodle para recuperar los datos de la base de datos de Moodle.

## Diagramas de secuencia.

En el análisis detallado se utilizan diagramas de secuencia para representar cómo es la interacción entre las clases que intervienen en el flujo de un caso de uso. El diagrama cuenta con clases y actores (objetos) del caso de uso correspondiente, establecidos de forma horizontal y de izquierda a derecha iniciando por el actor, donde cada objeto tiene una línea de vida vertical y la interacción entre cada uno se muestra mediante mensajes secuenciales. Contando ya con el análisis de robustez y tomando como base el diagrama de clases, se realizan los diagramas de secuencia para cada caso de uso, siendo las operaciones establecidas en las clases los mensajes del diagrama de secuencia.

Al igual que en los diagramas de robustez, los flujos alternos resaltan por su color rojo en comparación con el color negro de los mensajes pertenecientes al flujo normal. Además, como una constante, cada que se necesita la manipulación de datos almacenados, las clases de profesor, administrador, laboratorio y estadística y elementoScorm, se comunicarán con una de las interfaces de acceso a datos que a su vez se comunicará con moodle\_database para acceder a la base de datos; también, al ser un sistema web se toma al actor como el navegador web, pues es quien se encarga de enviar las peticiones al servidor. A continuación, se presentan los diagramas de secuencia desarrollados por caso de uso.



Ilustración 19: Diagrama de secuencia CU 01 - Consultar laboratorios propios.

En el diagrama de la ilustración 19 se aprecia como primeramente el navegador solicita la página de visualización de laboratorios, después se crea una entidad “Profesor” que obtiene los laboratorios mediante las clases de IProfesor y “moodle\_database”, y al final se construye una página con los laboratorios que es regresada al navegador. Para el flujo alterno, también se construye una página que se envía al navegador, pero esta vez conteniendo un mensaje de notificación.



Ilustración 20: Diagrama de secuencia CU 02 - Eliminar laboratorio.

Para el diagrama de eliminar laboratorio en la ilustración 20, el navegador envía una petición para borrar un laboratorio, luego se invoca la operación de eliminar en la clase de “Laboratorio”, se elimina el laboratorio con las clases de “ILaboratorio” y “moodle\_database” y al final se redirige a la página de visualización de laboratorios.



Ilustración 21: Diagrama de secuencia CU 03 - Consultar avances por alumno.

En el diagrama correspondiente al caso de uso “consultar avances por alumno” en la ilustración 21, se observa que una vez solicitada la página de estadísticas de alumno por el navegador, se procede a obtener los laboratorios asociados al alumno mediante la clase “Profesor”, para después, recorrer el listado, conocer si el laboratorio ya fue realizado por el alumno (llamada a “fueRealizado” en la entidad “Laboratorio”), y en el caso correcto continuar con la obtención de los datos estadísticos con la clase “Estadistica”.

Primero, se obtienen las repeticiones del laboratorio mediante la clase “Laboratorio”, para después obtener los datos referentes a SCORM utilizando la clase “ElementoSCORM”. Al final, se construye una página con la información de avance obtenida o un mensaje de notificación para el flujo alterno y se regresa al navegador.



Ilustración 22: Diagrama de secuencia CU 04 - Consultar avances por grupo.

El diagrama de consulta de avances por grupo en la ilustración 22 inicia con la petición de la página de avances y la consulta de alumnos registrados al grupo con la clase “Profesor”, después continúa con la obtención de los laboratorios asociados al grupo, o bien, construye una página con un mensaje de notificación y la regresa al navegador en el flujo alterno.

Una vez contando con los laboratorios se recorre el listado, y para cada laboratorio se obtienen las estadísticas de grupo mediante la clase “Estadistica”, la cual obtiene los datos relativos a SCORM con la clase “ElementoSCORM” y el porcentaje de realización del laboratorio usando la clase “Laboratorio”. Finalmente se construye una página con la información de avance y se regresa al navegador.



Ilustración 23: Diagrama de secuencia CU 05 - Consultar laboratorios.

Para el diagrama de consultar laboratorio en la ilustración 23, se desarrolla la misma secuencia que para el caso de uso “CU 01 – Consultar laboratorios propios”, sin embargo, antes de obtener los laboratorios de un profesor, en este caso se obtienen los profesores registrados con la clase “Administrador” (cabe recalcar que es una especialización de Profesor), para posteriormente recorrer el listado y adquirir los laboratorios asociados a cada uno de ellos. La secuencia termina construyendo una página con los laboratorios por profesor o un mensaje de notificación y regresándola al navegador.



Ilustración 24: Diagrama de secuencia CU 06 - Consultar avances personales.

En el diagrama correspondiente al caso de uso “consultar avances personales” en la ilustración 24, se observa que, una vez solicitada la página de estadísticas personales por el navegador, se procede a obtener los laboratorios asociados al alumno mediante la clase “Profesor”, para después, recorrer el listado, conocer si el laboratorio ya fue realizado por el alumno, y en el caso correcto continuar con la obtención de los datos estadísticos con la clase “Estadistica”.

Primeramente, se obtienen las repeticiones del laboratorio mediante la clase “Laboratorio”, para después obtener los datos referentes a SCORM utilizando la clase “ElementoSCORM”. Al final, se construye una página con la información de avance obtenida o un mensaje de notificación para el flujo alterno y se regresa al navegador.

# Capítulo 6: Despliegue.

En este capítulo se relaciona la última etapa de la metodología de desarrollo de software ICONIX: despliegue, donde se incluye el diagrama de despliegue de la aplicación, el plan de pruebas, y se detalla el trabajo relativo a la codificación, es decir, la realización en sí del desarrollo de la plataforma, elementos propuestos por la metodología en su última etapa.

## Diagrama de despliegue.

El diagrama de despliegue cumple la función de representar gráficamente los componentes físicos y lógicos de la aplicación a desarrollar, así como las relaciones entre éstos utilizando UML. En ICONIX, el diagrama de despliegue forma parte de su etapa final y se realiza como actividad previa al desarrollo o codificación de la aplicación.



Ilustración 25: Diagrama de despliegue.

Tal como se observa en la ilustración 25, el complemento estaría compuesto por un servidor y varios clientes. El cliente consta de un navegador web que proporcione acceso al sitio y permita la comunicación con el mismo mediante el protocolo HTTP; por otro lado, el servidor físico constaría de un servidor web (Apache Tomcat) donde se encuentre el LMS (Moodle) extendido, y el manejador de base de datos MySQL. Moodle es la aplicación contenedora del API de SCORM, así como de los laboratorios virtuales.

## Pruebas.

Tabla 17: Definición de casos de prueba unitaria.

|  |  |  |  |
| --- | --- | --- | --- |
| **Número del caso de prueba** | **Clase** | **Método** | **Descripción** |
| CPU001 | Profesor | obtenerLaboratorios(): List<Laboratorio> | Se prueba la obtención de laboratorios asociados a un profesor, donde la lista no es nula. |
| CPU002 | Profesor | obtenerLaboratorios(): List<Laboratorio> | Se prueba la obtención de laboratorios asociados a un profesor, donde la longitud de la lista es igual al número de laboratorios registrados. |
| CPU003 | Profesor | obtenerLaboratorios(): List<Laboratorio> | Se prueba la obtención de laboratorios asociados a un profesor, donde la lista esté vacía. |
| CPU004 | Profesor | construirLista(): List<Laboratorios> | Se prueba la conversión de un arreglo a una lista, donde la lista no debe ser nula. |
| CPU005 | Profesor | construirLista(): List<Laboratorios> | Se prueba la conversión de un arreglo a una lista, donde la longitud de la lista debe ser igual a la longitud del arreglo. |
| CPU006 | Profesor | alumnosRegistrados(): boolean | Se prueba la existencia de alumnos asignados a un curso dado, donde las entradas son válidas y existen alumnos registrados en el curso. |
| CPU007 | Profesor | alumnosRegistrados(): boolean | Se prueba la existencia de alumnos asignados a un curso dado, donde las entradas son válidas y no existen alumnos registrados en el curso. |
| CPU008 | Profesor | alumnosRegistrados(): boolean | Se prueba la existencia de alumnos asignados a un curso dado, donde las entradas no son válidas. |
| CPU009 | Profesor | obtenerLaboratoriosGrupo(): List<Laboratorio> | Se prueba la obtención de laboratorios asociados a un grupo, donde las entradas son válidas y la lista no es nula. |
| CPU010 | Profesor | obtenerLaboratoriosGrupo(): List<Laboratorio> | Se prueba la obtención de laboratorios asociados a un grupo, donde las entradas son válidas y la longitud de la lista es igual al número de laboratorios registrados. |
| CPU011 | Profesor | obtenerLaboratoriosGrupo(): List<Laboratorio> | Se prueba la obtención de laboratorios asociados a un grupo, donde las entradas son válidas y la lista esté vacía. |
| CPU012 | Profesor | obtenerLaboratoriosGrupo(): List<Laboratorio> | Se prueba la obtención de laboratorios asociados a un grupo, donde las entradas no son válidas. |
| CPU013 | Profesor | obtenerLaboratoriosAlumno(): List<Laboratorio> | Se prueba la obtención de laboratorios asociados a un alumno, donde las entradas son válidas y la lista no es nula. |
| CPU014 | Profesor | obtenerLaboratoriosAlumno(): List<Laboratorio> | Se prueba la obtención de laboratorios asociados a un alumno, donde las entradas son válidas y la longitud de la lista es igual al número de laboratorios registrados. |
| CPU015 | Profesor | obtenerLaboratoriosAlumno(): List<Laboratorio> | Se prueba la obtención de laboratorios asociados a un alumno, donde las entradas son válidas y la lista esté vacía. |
| CPU016 | Profesor | obtenerLaboratoriosAlumno(): List<Laboratorio> | Se prueba la obtención de laboratorios asociados a un alumno, donde las entradas no son válidas. |
| CPU017 | Administrador | obtenerProfesores(): List<Profesor> | Se prueba la obtención de profesores, donde la lista no es nula. |
| CPU018 | Administrador | obtenerProfesores(): List<Profesor> | Se prueba la obtención de profesores, donde la longitud de la lista es igual al número de profesores registrados. |
| CPU019 | Administrador | obtenerProfesores(): List<Profesor> | Se prueba la obtención de profesores, donde la lista esté vacía. |
| CPU020 | Laboratorio | eliminar(): boolean | Se prueba la baja de un laboratorio. |
| CPU21 | Laboratorio | obtenerRepeticiones(): int | Se prueba la obtención del número de repeticiones de un laboratorio por parte de un alumno, donde las entradas son válidas y existen repeticiones registradas. |
| CPU022 | Laboratorio | obtenerRepeticiones(): int | Se prueba la obtención del número de repeticiones de un laboratorio por parte de un alumno, donde las entradas son válidas y no existen repeticiones registradas. |
| CPU023 | Laboratorio | obtenerRepeticiones(): int | Se prueba la obtención del número de repeticiones de un laboratorio por parte de un alumno, donde las entradas no son válidas. |
| CPU024 | Laboratorio | obtenerPorcentajeRealizacion(): double | Se prueba la obtención del porcentaje de realización de un laboratorio por parte de un grupo, donde las entradas son válidas y el laboratorio no ha sido realizado. |
| CPU025 | Laboratorio | obtenerPorcentajeRealizacion(): double | Se prueba la obtención del porcentaje de realización de un laboratorio por parte de un grupo, donde las entradas son válidas y el laboratorio ha sido realizado. |
| CPU026 | Laboratorio | obtenerPorcentajeRealización(): double | Se prueba la obtención del porcentaje de realización de un laboratorio por parte de un grupo, donde las entradas no son válidas. |
| CPU027 | laboratorio | fueRealizado(): boolean | Se prueba si un laboratorio fue realizado por un alumno, donde las entradas son válidas y el laboratorio ha sido realizado. |
| CPU028 | Laboratorio | fueRealizado(): boolean | Se prueba si un laboratorio fue realizado por un alumno, donde las entradas son válidas y el laboratorio no ha sido realizado. |
| CPU029 | Laboratorio | fueRealizado(): boolean | Se prueba si un laboratorio fue realizado por un alumno, donde las entradas no son válidas. |
|  |  |  |  |
|  |  |  |  |

Tabla 18: Caso de prueba unitaria CPU001.

|  |  |
| --- | --- |
| **Caso de prueba** | CPU001. |
| **Descripción** | En este caso de prueba para el método “obtenerLaboratorios” se verifica la obtención de la lista de laboratorios asociados a un profesor dado, identificado por el atributo “id” de la instancia de la clase. La lista no deberá ser nula. |
| **Salida esperada** | List<Laboratorio>. |
| **Salida obtenida** |  |
| **Observaciones** |  |

Tabla 19: Caso de prueba unitaria CPU002.

|  |  |
| --- | --- |
| **Caso de prueba** | CPU002. |
| **Descripción** | En este caso de prueba para el método “obtenerLaboratorios” se verifica la obtención de la lista de laboratorios asociados a un profesor dado, identificado por el atributo “id” de la instancia de la clase. La longitud de la lista deberá ser igual al número de laboratorios registrados asociados al profesor. |
| **Salida esperada** | List<Laboratorio>. |
| **Resultado esperado** | Longitud de lista = número de laboratorios registrados asociados al profesor. |
| **Salida obtenida** |  |
| **Resultado obtenido** |  |
| **Observaciones** |  |

Tabla 20: Caso de prueba unitaria CPU003.

|  |  |
| --- | --- |
| **Caso de prueba** | CPU003. |
| **Descripción** | En este caso de prueba para el método “obtenerLaboratorios” se verifica la obtención de la lista de laboratorios asociados a un profesor dado, identificado por el atributo “id” de la instancia de la clase. La lista deberá estar vacía, pues no habrá laboratorios registrados asociados al profesor. |
| **Salida esperada** | List<Laboratorio>. |
| **Resultado esperado** | Longitud de lista = 0. |
| **Salida obtenida** |  |
| **Resultado obtenido** |  |
| **Observaciones** |  |

Tabla 21: Caso de prueba unitaria CPU004.

|  |  |
| --- | --- |
| **Caso de prueba** | CPU004. |
| **Descripción** | En este caso de prueba para el método “construirLista” se verifica la conversión correcta de un arreglo a una lista de laboratorios. La lista no deberá ser nula. |
| **Datos de entrada** | |  | | --- | | laboratorios | | Arreglo con laboratorios asociados | |
| **Salida esperada** | List<Laboratorio>. |
| **Salida obtenida** |  |
| **Observaciones** |  |

Tabla 22: Caso de prueba unitaria CPU005.

|  |  |
| --- | --- |
| **Caso de prueba** | CPU005. |
| **Descripción** | En este caso de prueba para el método “construirLista” se verifica la conversión correcta de un arreglo a una lista de laboratorios. La longitud de la lista deberá ser igual a la longitud del arreglo. |
| **Datos de entrada** | |  | | --- | | laboratorios | | Arreglo con laboratorios asociados | |
| **Salida esperada** | List<Laboratorio>. |
| **Resultado esperado** | Longitud de lista = longitud de laboratorios. |
| **Salida obtenida** |  |
| **Resultado obtenido** |  |
| **Observaciones** |  |

Tabla 23: Caso de prueba unitaria CPU006.

|  |  |
| --- | --- |
| **Caso de prueba** | CPU006. |
| **Descripción** | En este caso de prueba para el método “alumnosRegistrados” se verifica si existen alumnos registrados a un curso dado. El curso existe y hay alumnos registrados a él. |
| **Datos de entrada** | |  | | --- | | idGrupo | | Identificador de grupo existente. | |
| **Salida esperada** | True. |
| **Salida obtenida** |  |
| **Observaciones** |  |

Tabla 24: Caso de prueba unitaria CPU007.

|  |  |
| --- | --- |
| **Caso de prueba** | CPU007. |
| **Descripción** | En este caso de prueba para el método “alumnosRegistrados” se verifica si existen alumnos registrados a un curso dado. El curso existe y no hay alumnos registrados a él. |
| **Datos de entrada** | |  | | --- | | idGrupo | | Identificador de grupo existente. | |
| **Salida esperada** | False. |
| **Salida obtenida** |  |
| **Observaciones** |  |

Tabla 25: Caso de prueba unitaria CPU008.

|  |  |
| --- | --- |
| **Caso de prueba** | CPU008. |
| **Descripción** | En este caso de prueba para el método “alumnosRegistrados” se verifica si existen alumnos registrados a un curso dado. El curso no existe. |
| **Datos de entrada** | |  | | --- | | idGrupo | | Identificador de grupo no existente. | |
| **Salida esperada** | False. |
| **Salida obtenida** |  |
| **Observaciones** |  |

Tabla 26: Caso de prueba unitaria CPU009.

|  |  |
| --- | --- |
| **Caso de prueba** | CPU009. |
| **Descripción** | En este caso de prueba para el método “obtenerLaboratoriosGrupo” se verifica la obtención de la lista de laboratorios asociados a un grupo dado. La lista no deberá ser nula. |
| **Datos de entrada** | |  | | --- | | idGrupo | | Identificador de grupo existente. | |
| **Salida esperada** | List<Laboratorio>. |
| **Salida obtenida** |  |
| **Observaciones** |  |

Tabla 27: Caso de prueba unitaria CPU010.

|  |  |
| --- | --- |
| **Caso de prueba** | CPU010. |
| **Descripción** | En este caso de prueba para el método “obtenerLaboratoriosGrupo” se verifica la obtención de la lista de laboratorios asociados a un grupo dado. La longitud de la lista deberá ser igual al número de laboratorios registrados asociados al grupo. |
| **Datos de entrada** | |  | | --- | | idGrupo | | Identificador de grupo existente. | |
| **Salida esperada** | List<Laboratorio>. |
| **Resultado esperado** | Longitud de lista = número de laboratorios registrados asociados al grupo. |
| **Salida obtenida** |  |
| **Resultado obtenido** |  |
| **Observaciones** |  |

Tabla 28: Caso de prueba unitaria CPU011.

|  |  |
| --- | --- |
| **Caso de prueba** | CPU011. |
| **Descripción** | En este caso de prueba para el método “obtenerLaboratoriosGrupo” se verifica la obtención de la lista de laboratorios asociados a un grupo dado. La lista deberá estar vacía, pues no habrá laboratorios registrados asociados al grupo. |
| **Datos de entrada** | |  | | --- | | idGrupo | | Identificador de grupo existente. | |
| **Salida esperada** | List<Laboratorio>. |
| **Resultado esperado** | Longitud de lista = 0. |
| **Salida obtenida** |  |
| **Resultado obtenido** |  |
| **Observaciones** |  |

Tabla 29: Caso de prueba unitaria CPU012.

|  |  |
| --- | --- |
| **Caso de prueba** | CPU012. |
| **Descripción** | En este caso de prueba para el método “obtenerLaboratoriosGrupo” se verifica la obtención de la lista de laboratorios asociados a un grupo dado. El grupo no existe. |
| **Datos de entrada** | |  | | --- | | idGrupo | | Identificador de grupo no existente. | |
| **Salida esperada** | List<Laboratorio>. |
| **Resultado esperado** | Longitud de lista = 0. |
| **Salida obtenida** |  |
| **Resultado obtenido** |  |
| **Observaciones** |  |

Tabla 30: Caso de prueba unitaria CPU013.

|  |  |
| --- | --- |
| **Caso de prueba** | CPU013. |
| **Descripción** | En este caso de prueba para el método “obtenerLaboratoriosAlumno” se verifica la obtención de la lista de laboratorios asociados a un alumno dado. La lista no deberá ser nula. |
| **Datos de entrada** | |  | | --- | | idAlumno | | Identificador de alumno existente. | |
| **Salida esperada** | List<Laboratorio>. |
| **Salida obtenida** |  |
| **Observaciones** |  |

Tabla 31: Caso de prueba unitaria CPU014.

|  |  |
| --- | --- |
| **Caso de prueba** | CPU014. |
| **Descripción** | En este caso de prueba para el método “obtenerLaboratoriosAlumno” se verifica la obtención de la lista de laboratorios asociados a un alumno dado. La longitud de la lista deberá ser igual al número de laboratorios registrados asociados al alumno. |
| **Datos de entrada** | |  | | --- | | idAlumno | | Identificador de alumno existente. | |
| **Salida esperada** | List<Laboratorio>. |
| **Resultado esperado** | Longitud de lista = número de laboratorios registrados asociados al alumno. |
| **Salida obtenida** |  |
| **Resultado obtenido** |  |
| **Observaciones** |  |

Tabla 32: Caso de prueba unitaria CPU015.

|  |  |
| --- | --- |
| **Caso de prueba** | CPU015. |
| **Descripción** | En este caso de prueba para el método “obtenerLaboratoriosAlumno” se verifica la obtención de la lista de laboratorios asociados a un alumno dado. La lista deberá estar vacía, pues no habrá laboratorios registrados asociados al alumno. |
| **Datos de entrada** | |  | | --- | | idAlumno | | Identificador de alumno existente. | |
| **Salida esperada** | List<Laboratorio>. |
| **Resultado esperado** | Longitud de lista = 0. |
| **Salida obtenida** |  |
| **Resultado obtenido** |  |
| **Observaciones** |  |

Tabla 33: Caso de prueba unitaria CPU016.

|  |  |
| --- | --- |
| **Caso de prueba** | CPU016. |
| **Descripción** | En este caso de prueba para el método “obtenerLaboratoriosAlumno” se verifica la obtención de la lista de laboratorios asociados a un alumno dado. El alumno no existe. |
| **Datos de entrada** | |  | | --- | | idAlumno | | Identificador de alumno no existente. | |
| **Salida esperada** | List<Laboratorio>. |
| **Resultado esperado** | Longitud de lista = 0. |
| **Salida obtenida** |  |
| **Resultado obtenido** |  |
| **Observaciones** |  |

Tabla 34: Caso de prueba unitaria CPU017.

|  |  |
| --- | --- |
| **Caso de prueba** | CPU017. |
| **Descripción** | En este caso de prueba para el método “obtenerProfesores” se verifica la obtención de la lista de profesores registrados. La lista no deberá ser nula. |
| **Salida esperada** | List<Profesor>. |
| **Salida obtenida** |  |
| **Observaciones** |  |

Tabla 35: Caso de prueba unitaria CPU018.

|  |  |
| --- | --- |
| **Caso de prueba** | CPU018. |
| **Descripción** | En este caso de prueba para el método “obtenerProfesores” se verifica la obtención de la lista de profesores. La longitud de la lista deberá ser igual al número de profesores registrados. |
| **Salida esperada** | List<Profesor>. |
| **Resultado esperado** | Longitud de lista = número de profesores registrados. |
| **Salida obtenida** |  |
| **Resultado obtenido** |  |
| **Observaciones** |  |

Tabla 36: Caso de prueba unitaria CPU019.

|  |  |
| --- | --- |
| **Caso de prueba** | CPU019. |
| **Descripción** | En este caso de prueba para el método “obtenerProfesores” se verifica la obtención de la lista de profesores. La lista deberá estar vacía, pues no habrá profesores registrados. |
| **Salida esperada** | List<Profesor>. |
| **Resultado esperado** | Longitud de lista = 0. |
| **Salida obtenida** |  |
| **Resultado obtenido** |  |
| **Observaciones** |  |

Tabla 37: Caso de prueba unitaria CPU020.

|  |  |
| --- | --- |
| **Caso de prueba** | CPU020. |
| **Descripción** | En este caso de prueba para el método “eliminar” se verifica la baja de un laboratorio dado, identificado por el atributo “id” de la clase. |
| **Salida esperada** | True. |
| **Salida obtenida** |  |
| **Observaciones** |  |

## Codificación.

## Resultados de la ejecución de pruebas.

# Conclusiones y trabajo futuro.

# Bibliografía.

Advaced distributed leraning Initiative (2001). Sharable Content Reference Model, The SCORM content aggregation model. ADL.

Advaced distributed leraning Initiative (2001). Sharable Content Reference Model, The SCORM overview. ADL.

Advaced distributed leraning Initiative (2001). Sharable Content Reference Model, The SCORM runtime - environment. ADL.

Anggrainingsih, R., Johannanda, B. O. P., Kuswara, A. P., Wahyuningsih, D., & Rejekiningsih, T. (2016). Comparison of maintainability and flexibility on open source LMS. 2016 International Seminar on Application for Technology of Information and Communication (ISemantic), 273-277. https://doi.org/10.1109/ISEMANTIC.2016.7873850.

Awang, N. B., & Darus, M. Y. B. (2012). Evaluation of an Open Source Learning Management System: Claroline. Procedia - Social and Behavioral Sciences, 67, 416-426. https://doi.org/10.1016/j.sbspro.2012.11.346.

Aydin, C. C., & Tirkes, G. (2010). Open source learning management systems in e-learning and Moodle. IEEE EDUCON 2010 Conference, 593-600. https://doi.org/10.1109/EDUCON.2010.5492522.

Budai, T., & Kuczmann, M. (2018). Towards a Modern, Integrated Virtual Laboratory System. Acta Polytechnica Hungarica, 15(3), 14.

Callaghan, Mj., Gomez Eguiluz, A., McLaughlin, G., & McShane, N. (2015). Opportunities and challenges in virtual reality for remote and virtual laboratories. Proceedings of 2015 12th International Conference on Remote Engineering and Virtual Instrumentation (REV), 235-237. https://doi.org/10.1109/REV.2015.7087298.

Caminero, A. C., Hernandez, R., Ros, S., Robles-Gomez, A., & Tobarra, L. (2013). Choosing the right LMS: A performance evaluation of three open-source LMS. 2013 IEEE Global Engineering Education Conference (EDUCON), 287-294. https://doi.org/10.1109/EduCon.2013.6530119.

Cavus, N., & Zabadi, T. (2014). A Comparison of Open Source Learning Management Systems. Procedia - Social and Behavioral Sciences, 143, 521-526. https://doi.org/10.1016/j.sbspro.2014.07.430.

Coates, H., James, R., & Baldwin, G. (s. f.). A CRITICAL EXAMINATION OF THE EFFECTS OF LEARNING MANAGEMENT SYSTEMS ON UNIVERSITY TEACHING AND LEARNING, 18.

Dias, F., Miguens Matutino, P., & Barata, M. (2014). Virtual laboratory for educational environments. 2014 11th International Conference on Remote Engineering and Virtual Instrumentation (REV), 191-194. https://doi.org/10.1109/REV.2014.6784252.

Guo, W., Gao, Y., & Wang, Y. (2012). Design and realization of the interactive virtual laboratory based on VRML. 2012 2nd International Conference on Consumer Electronics, Communications and Networks (CECNet), 2510-2513. https://doi.org/10.1109/CECNet.2012.6201755.

Hashemi, J., Austin-Stalcup, K. A., Anderson, E. E., Chandrashekar, N., & Majkowski, A. (2005). Elements of a Realistic Virtual Laboratory Experience in Materials Science: Development and Evaluation. 13.

ISO/IEC 25010, 2011. Systems and software engineering -- Systems and software Quality Requirements and Evaluation (SQuaRE) – System and software quality models.

Kelly, D., & Thorn, K. (2013). elearn Magazine: Should Instructional Designers care about the Tin Can API? Recuperado 18 de septiembre de 2019, de ELearn Magazine, an ACM Publication website: https://elearnmag.acm.org/archive.cfm?aid=2446579.

Kulshrestha, T., & Kant, A. R. (2013). Benefits of Learning Management System (LMS) in Indian Education. Engineering Technology, 4(08), 12.

Lengyel, P., Herdon, M., & Szilágyi, R. (2006). Comparison of Moodle and ATutor LMSs. 8.

Luengas, L. A., Guevara, J. C., & Sánchez, G. (2009). ¿Cómo desarrollar un laboratorio virtual? Metodología de diseño. 6.

Manso Vazquez, M., Caeiro Rodriguez, M., & Llamas Nistal, M. (2015). Development of a xAPI application profile for self-regulated learning requirements for capturing SRL related data. 2015 IEEE Global Engineering Education Conference (EDUCON), 358-365. https://doi.org/10.1109/EDUCON.2015.7095997.

Mukandla, E. & Dwolatzky, B. (2004). A survey of agile methodologies. The transactions of the SA Institute of Electrical Engineers, 236 - 247.

OMG. 2019. OMG Unified Modeling Language, Version 2.5. OMG Document Number formal/2015-03-01. http://omg.org/spec/UML/2.5.

Ouadoud, M., Chkouri, M. Y., & Nejjari, A. (2018). LeaderTICE: A Platforms Recommendation System Based on a Comparative and Evaluative Study of Free E-learning Platforms. International Journal of Online Engineering (IJOE), 14(01), 132. https://doi.org/10.3991/ijoe.v14i01.7865.

Paulsen, M. F. (2002). Online education systems in Scandinavian and Australian Universities: A Comparative Study. The International Review of Research in Open and Distance Learning. Volume 3 (2), 152 – 167.

Poulova, P., Simonova, I., & Manenova, M. (2015). Which One, or Another? Comparative Analysis of Selected LMS. Procedia - Social and Behavioral Sciences, 186, 1302-1308. https://doi.org/10.1016/j.sbspro.2015.04.052.

Rosenberg, D., & Stephens, M. (2013). Use case driven object modeling with UML: Theory and practice. Berkeley, CA: Apress.

Ruiz Reyes, N., Vera Candeas, P., Garcia Galan, S., Viciana, R., Canadas, F., & Reche, P. J. (2009). Comparing open-source e-learning platforms from adaptivity point of view. 2009 EAEEIE Annual Conference, 1-6. https://doi.org/10.1109/EAEEIE.2009.5335482.

RUSTICI Software, One minute SCORM overview for anyone. www.scorm.com.

Saydam, S., Timms, W., Raval, S., & Daly, C. (2013). Using Moodle &#x2014; An open source leaning management system in Australian Mining Engineering Education. 2013 IEEE International Conference in MOOC, Innovation and Technology in Education (MITE), 386-389. https://doi.org/10.1109/MITE.2013.6756373.

Stark, Erich & Kucera, Erik & Bisták, Pavol & Haffner, Oto. (2018). Experiment with JavaScript on Client and Server Side of the Virtual Laboratory and Visualised in Mixed Reality Using Microsoft HoloLens. Journal of Automation, Mobile Robotics & Intelligent Systems. 12. 10.14313/JAMRIS\_1-2018/2.

Su, J.-M., Tseng, S.-S., Chen, C.-Y., Weng, J.-F., & Tsai, W.-N. (2006). Constructing SCORM compliant course based on High-Level Petri Nets. Computer Standards & Interfaces, 28(3), 336-355. https://doi.org/10.1016/j.csi.2005.04.001.

Taamallah, A., & Khemaja, M. (2014). Designing and eXperiencing smart objects based learning scenarios: An approach combining IMS LD, XAPI and IoT. Proceedings of the Second International Conference on Technological Ecosystems for Enhancing Multiculturality - TEEM ’14, 373-379. https://doi.org/10.1145/2669711.2669926.

Thakare, B., Bhokse, B., & Thakare, L. (2012). Deriving Best Practices from Development Methodology Base (Part 2). International Journal of Engineering Research, 1(6), 8.

Topper, J. S., & Horner, N. C. (2013). Model-Based Systems Engineering in Support of Complex Systems Development. JOHNS HOPKINS APL TECHNICAL DIGEST, 32(1), 14.

Tretinjak, M. F. (2018). Moving teaching from blackboard to the learning management system — Helping absent students learn from home. En 2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO) (pp. 0500-0502). Opatija: IEEE. https://doi.org/10.23919/MIPRO.2018.8400095.

Voss, G. B., Nunes, F. B., Muhlbeier, A. R. K., & Medina, R. D. (2013). Context-Aware Virtual Laboratory for Teaching Computer Networks: A Proposal in the 3D OpenSim Environment. 2013 XV Symposium on Virtual and Augmented Reality, 252-255. https://doi.org/10.1109/SVR.2013.46.

Wang, T.-H., Chang, H.-P., Sie, Y.-L., Chan, K.-H., Tzou, M.-T., & Shih, T. K. (2005). Reading SCORM compliant multimedia courses using heterogeneous pervasive devices. Proceedings of the 13th Annual ACM International Conference on Multimedia - MULTIMEDIA ’05, 806. https://doi.org/10.1145/1101149.1101325

Wiegers, K. E., & Beatty, J. (2013). *Software requirements* (Third edition). Redmond, Washington: Microsoft Press, s division of Microsoft Corporation.

Wiseman, C., Wong, K., Wolf, T., & Gorinsky, S. (2008). Operational Experience With a Virtual Networking Laboratory. 5.

Zapata-Rivera, L. F., & Petrie, M. M. L. (2018). XAPI-Based Model for Tracking On-line Laboratory Applications. 2018 IEEE Frontiers in Education Conference (FIE), 1-9. https://doi.org/10.1109/FIE.2018.8658869.