CHAMILOMOBILE – APLICACIÓN MÓVIL ANDROID PARA LA INTERACCIÓN Y OBTENCIÓN DE INDICADORES DE SEGUIMIENTO, SINCRONIZADA CON EL LMS CHAMILO DE LA FACULTAD DE INGENIERÍA DE LA UNIVERSIDAD DE NARIÑO

JORGE ALBEIRO RIVERA ROSERO
FELIPE ROA NARVÁEZ

UNIVERSIDAD DE NARIÑO
FACULTAD DE INGENIERÍA
DEPARTAMENTO DE INGENIERÍA DE SISTEMAS
SAN JUAN DE PASTO
2016

Universidad de Nariño

CHAMILOMOBILE – APLICACIÓN MÓVIL ANDROID PARA LA INTERACCIÓN Y OBTENCIÓN DE INDICADORES DE SEGUIMIENTO, SINCRONIZADA CON EL LMS CHAMILO DE LA FACULTAD DE INGENIERÍA DE LA UNIVERSIDAD DE NARIÑO

JORGE ALBEIRO RIVERA ROSERO
FELIPE ROA NARVÁEZ

Trabajo de grado presentado como requisito parcial para optar al título de Ingenieros de Sistemas

Esp. HECTOR ANDRES MORA PAZ
Director

M. Sc. JUAN CARLOS CASTILLO
Codirector

UNIVERSIDAD DE NARIÑO
FACULTAD DE INGENIERÍA
DEPARTAMENTO DE INGENIERÍA DE SISTEMAS
SAN JUAN DE PASTO
2016

NOTA DE RESPONSABILIDAD

“La Universidad de Nariño no se hace responsable de las opiniones o resultados obtenidos en el presente trabajo y para su publicación priman las normas sobre el derecho de autor”.

Artículo 13, Acuerdo N. 005 de 2010, emanado del honorable Consejo Académico.

Artículo 1, Acuerdo No. 324 de octubre 11 de 1966 emanado del honorable Consejo Directivo de la Universidad de Nariño.

NOTA DE ACEPTACIÓN

_____

_____

_____

_____

_____
Firma del presidente del jurado

_____
Firma del jurado

_____
Firma del jurado

_____
Firma del Director de proyecto

San Juan de Pasto, Mayo de 2016

Con toda gratitud a todas las personas que siempre creyeron en nuestra capacidad, es grato saber la fuerza y determinación que poseemos cuando queremos alcanzar algo.

A mis padres por entregarme las pautas para leer las notas resonantes del tiempo, guiarme con entonación de su experiencia y enseñarme que siempre es posible corregir las desafinadas pisadas de la vida.

A mi compañero de estudio y amigo Felipe Roa, gracias por tu apoyo y por seguir en la lucha por lograr lo que siempre anhelamos en nuestro futuro académico y laborar.

A mis hermanos por su apoyo condicional y por guiarme para poder alcanzar grandes triunfos.

A mi Lu, la inspiración más grande de ingenio y ternura. Por ti también lucho y prometo ser tu escudero toda la vida.

Jorge Rivera

A mi madre Lucia la mejor madre de mundo a la cual amo con todas las fuerzas de mi alma, por su protección constante, su amor verdadero y su lucha continúa. Por hacer de nosotros los mejores hijos. Gracias por estar siempre ahí.

A mis hermanas por su apoyo y compañía incondicional y porque sé que guían mi conducta a través de mi actuar.

A Daisy por convertirse en mi compañera de vida y por construir juntos un futuro mejor.

A mi Juli por inspirar las melodías de mi corazón, las tonadas de mis ideas, las partituras de mis metas y por interpretar a dúo esta sinfonía interpretada seguramente hasta la eternidad.

Felipe Roa

AGRADECIMIENTOS

Al ingeniero Hector Mora, por manifestarnos su interés en dirigir nuestro trabajo de grado, por su colaboración, apoyo, revisiones puntuales y aportes precisos en los temas tratados en este proyecto.

Al ingeniero Juan Carlos Castillo, por su valioso apoyo e interés en la construcción del proyecto.

A la ingeniera Dalila Pachajoa, por su apoyo incondicional y su gran interés porque todo lo propuesto hoy tenga un gran fin.

A nuestra Alma Mater, la Universidad de Nariño por darnos tantos espacios y oportunidades de adquirir conocimiento.

A nuestros docentes de la Universidad de Nariño que compartieron sus conocimientos, dentro y fuera de clase, haciendo posible que nuestra formación profesional se resuma en satisfacciones académicas.

A nuestros compañeros de estudio con los cuales compartimos y construimos buenos momentos.

RESUMEN

CHAMILOMOBILE, es un sistema que permite a los usuarios del LMS Chamilo acceder a contenido relevante de esta plataforma, de manera simple, rápida, segura y confiable, siendo acorde a las tecnologías de última generación, las cuales cada vez toman mayor protagonismo en el entorno.

Con la herramienta CHAMILOMOBILE los usuarios del LMS Chamilo pueden acceder desde un Smartphone con sistema operativo Android a funcionales como: cursos, enviar mensajes a amigos, visualizar agenda, visualizar datos básicos de amigos, editar información básica del perfil y visualizar indicadores de uso, los cuales hacen referencia a datos estadísticos que evidencian la frecuencia con la que es usada la aplicación CHAMILOMOBILE y algunas de sus funcionalidades.

La interaccion entre CHAMILOMOBILE y el LMS Chamilo se realiza utilizando un middleware que consiste en una API que gestiona las peticiones y la seguridad en las mismas, realizando consultas directamente a la base de datos del LMS Chamilo ofreciendo además de una interfaz para la aplicación móvil, un recurso reutilizable para poder ser consumido desde otras plataformas.

Finalmente, las pruebas de funcionamiento de CHAMILOMOBILE se realizaron mediante la construcción de un prototipo de aplicación móvil ANDROID desarrollada en JAVA. La cual concluye exitosamente.

Palabras Claves: servicio Web, middleware, arquitectura, aplicación móvil, LMS.

ABSTRACT

CHAMILOMOBILE, is a system that allows users of LMS Chamilo to access to relevant content of this platform, in a simple, fast secure and reliable way, being according to the latest technologies, which increase their role in the environment.

With CHAMILOMOBILE tool, users of LMS Chamilo can access from a smartphone with Android Operating System to functionalities such as: courses, send messages to friends, view calendar, view basic data of friends, edit profile basic information and view indicators of use, which refer to statistic data that show the frequency with which is used CHAMILOMOBILE application and some of its functionalities.

The interaction between CHAMILOMOBILE and the LMS Chamilo is made using a middleware consisting of an API that manages the requests and the security in them, performing queries directly from the LMS Chamilo database, offering an interface for app and also a reusable resource to be consumed from other platforms.

Finally, the function tests of CHAMILOMOBILE were made with the construction of an ANDROID mobile application prototype developed in JAVA. Which concluded successfully.

Key words: Web service, middleware, architecture, app, LMS.

CONTENIDO

Pág.

1. MARCO TEÓRICO....20
1.1 INTRODUCCIÓN....20
1.2 ANTECEDENTES....20
1.3 APLICACIÓN MÓVIL....21
1.3.1 Generalidades.....21
1.3.1.1 Que es una aplicación móvil....21
1.3.1.2 El proceso de diseño y desarrollo de una aplicación....21
1.3.1.3 Tipos de aplicaciones según su desarrollo....22
1.3.2 Arquitecturas diseñadas para una aplicación móvil. 24
1.3.2.1 Componentes de los SO móviles.. 25
1.4 ANDROID....26
1.4.1 Código abierto.....27
1.4.2 Características 28
1.5 INTERACCIÓN 28
1.5.1 Servicios Web 28
1.5.1.1 SOAP. 32
1.5.1.2 REST....33
1.6 SINCRONIZACIÓN 37
1.7 LMS....38
1.7.1 Ventajas 38
1.7.2 Chamilo LMS....40
1.8 INGENIERÍA DE SOFTWARE. 40
1.8.1 RUP (Proceso Unificado de Rational) 42
1.9 HERRAMIENTAS TECNOLÓGICAS PARA LA CONSTRUCCIÓN DE CHAMILOMOBILE 45
1.9.1 Middleware.....45
1.9.2.1 JSON....46
1.9.2.2 XML.....47
1.9.3 Ingeniería de software basada en componentes – ISBC 47
1.9.3.1 Ventajas de la ISBC 48

1.9.4 Lenguajes de programación 49
1.9.4.1 JAVA 49
1.9.4.2 PHP 50
1.9.5 Tecnologías para prototipos de prueba. 51
1.9.5.1 Sistemas operativos 51
1.9.5.2 Entornos de desarrollo integrado (IDE). 52
1.9.5.3 Máquinas virtuales 54
1.9.5.4 Servidores Web 55
2. CHAMILOMOBILE: APLICACIÓN MÓVIL ANDROID PARA LA INTERACCIÓN Y OBTENCIÓN DE INDICADORES DE SEGUIMIENTO, SINCRONIZADA CON EL LMS CHAMILO DE LA FACULTAD DE INGENIERÍA DE LA UNIVERSIDAD DE NARIÑO 20
2.1 DESARROLLO CHAMILOMOBILE 20
2.2 PLAN ITERATIVO RUP SEGUIDO PARA CHAMILOMOBILE 20
2.3 ENTIDADES QUE INTERACTÚAN CON CHAMILOMOBILE 20
2.3.1 Actor Estudiante. Para los módulos de 20
2.4 INTERACCIONES DE LOS ACTORES CON EL SISTEMA 22
2.4.1 Diagrama de casos de uso general 22
2.4.2 Diagrama de casos de uso gestionar cursos 23
2.4.3 Diagrama de casos de uso gestionar mensajes 23
2.4.4 Diagrama de casos de uso gestionar perfil 24
2.5 ARQUITECTURA DEL SISTEMA 24
2.5.1 Diagrama de bloques 24
2.5.2 Diagrama de arquitectura del sistema 25
2.5.3 Diagrama de clases 26
2.6 COMPORTAMIENTO DEL SISTEMA 32
2.6.1 Diagramas de actividades 32
2.7 ESPECIFICACIONES DE IMPLEMENTACIÓN DE CHAMILOMOBILE 35
3. PROTOTIPO DE APLICACIÓN PARA EL DESARROLLO DE CHAMILOMOBILE 36
3.1 FUNCIONAMIENTO DE CHAMILOMOBILE EN EL PROTOTIPO DE APLICACIÓN 36
3.1.1 Indicadores 51
4 CONCLUSIONES 52

5 RECOMENDACIONES 53
REFERENCIAS BIBLIOGRÁFICAS.....54

LISTA DE FIGURAS

Pág.

Figura 1. Esquema RUP 24
Figura 2. Arquitectura Android 27
Figura 3. Proceso de servicios Web 29
Figura 4. Esquema básico de funcionamiento del servicio Web 30
Figura 5. Arquitectura básica de un sistema construido sobre SOAP 33
Figura 6. Datos actuales del LMS Chamilo 40
Figura 7. Plataforma Java 50
Figura 8. Iteraciones de fase por proceso 20
Figura 9. Diagrama de casos de uso general 22
Figura 10. Diagrama de caso de uso de gestionar cursos 23
Figura 11. Diagrama de casos de uso gestionar mensajes 23
Figura 12. Diagrama de casos de uso gestionar perfil 24
Figura 13. Diagrama de despliegue del sistema 24
Figura 14. Diagrama de bloques del sistema 25
Figura 15. Diagrama de clases de manejadores de interfaz 27
Figura 16. Diagrama de clases de representación de entidades 28
Figura 17. Diagrama de clases para la representación gráfica de datos para el usuario final 29
Figura 18. Diagrama de clases relacional para la representación gráfica de datos para el usuario final 30
Figura 19. Diagrama de clases del API (Middleware) 31
Figura 20. Diagrama de actividades de procesamiento de peticiones 32
Figura 21. Diagrama de actividades de cambios de estados en sincronización 33
Figura 22. Diagrama de actividades de sincronización 34
Figura 23. Diagrama de actividad de sincronización de conjunto de datos 34
Figura 24. Splash Screen 37
Figura 25. Solicitud de permisos 37
Figura 26. Formulario de logueo 38
Figura 27. Configuraciones de URL del API 38
Figura 28. Listado de cursos 39
Figura 29. Menú 39
Figura 30. Perfil del usuario 40
Figura 31. Menú de cursos 40
Figura 32. Descripción de curso 41
Figura 33. Documentos del curso 41
Figura 34. Enlaces del curso 42
Figura 35. Evaluaciones del curso 42
Figura 36. Glosario del curso 43
Figura 37. Informes del curso 43
Figura 38. Usuarios del curso 44

Figura 39. Tareas del curso 44
Figura 40. Archivos compartidos del curso 45
Figura 41. Notas personales del curso 45
Figura 42. Ejercicios del curso 46
Figura 43. Conversaciones del usuario 46
Figura 44. Mensajes del usuario 47
Figura 45. Agenda del usuario 47
Figura 46. Amigos del usuario 48
Figura 47. Perfil del amigo del usuario 48
Figura 48. Indicador accesos en este día 49
Figura 49. Indicador tareas consultadas en este día 49
Figura 50. Indicador descargas de documentos en este día 50
Figura 51. Acerca de 50

LISTA DE TABLAS

Pág.

Tabla 1. Especificación de módulos de CHAMILOMOBILE 20
Tabla 2. Descripción actor estudiante 21
Tabla 3. Descripción actor docente 21
Tabla 4. Especificaciones de plataforma de desarrollo y ejecución para el prototipo. 36
Tabla 5. Especificaciones adicionales de ejecución del servicio en el prototipo 36
Tabla 6. Tabla de resultados del prototipo de aplicación móvil 51

MARCAS REGISTRADAS

PHP es una marca registrada de The PHP Group © 2001-2015.

JAVA es una marca registrada de ORACLE Corporation.

ANDROID OPEN SOURCE PROJECT, es una marca registrada de Google Inc. ©2015.

CHAMILO es una marca registrada de The Chamilo Association ©2014.

APACHE es una marca registrada de The Apache Software Foundation © 2015.

DREAMWEAVER® es una marca registrada de Adobe Systems.

RUP es una marca registrada de International Business Machines Corporation.

UML es una marca registrada de OMG, consorcio dedicado al cuidado y el establecimiento de diversos estándares de tecnologías orientadas a objetos.

WINDOWS 8® es una línea de sistemas operativos registrada por Microsoft Corp.

WINDOWS SERVER™ es una línea de productos para servidores de Microsoft Corp.

GLOSARIO

- **API:** Interfaz de programación de aplicaciones, es el conjunto de funciones y procedimientos (o métodos, en la programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción. Son usadas generalmente en las bibliotecas.

- **Componente software**: es un elemento de un sistema software que ofrece un conjunto de servicios, o funcionalidades, a través de interfaces definidas. Un componente de software debe poseer las siguientes características: ser reutilizable, ser intercambiable, poseer interfaces definidas y ser cohesivos.

- **Escalabilidad**: es la propiedad deseable de un software que indica su habilidad para extender el margen de operaciones sin perder calidad, o bien manejar el crecimiento continuo de trabajo de manera fluida, o bien para estar preparado para crecer sin perder calidad en los servicios ofrecidos.

- **Extensibilidad**: es un principio de diseño de sistema en el que la aplicación tiene en cuenta el crecimiento futuro. Es una medida sistémica de la capacidad de extender un sistema y el nivel de esfuerzo que se requiere para poner en práctica la extensión. Las extensiones pueden ser a través de la adición de nuevas funcionalidades o mediante la modificación de las funcionalidades existentes.

- **Framework**: La palabra inglesa "framework" (marco de trabajo) define, en términos generales, un conjunto estandarizado de conceptos, prácticas y criterios para enfocar un tipo de problemática particular que sirve como referencia, para enfrentar y resolver nuevos problemas de índole similar.

- **Integración**: consiste en conectar distintas aplicaciones informáticas con la intención de que la información sea compartida entre ellas. El objetivo de la integración de sistemas es doble.

- **Interoperabilidad**: habilidad de dos o más sistemas o componentes para intercambiar información y utilizar la información intercambiada.

- **Bibliotecas**: es un conjunto de implementaciones de comportamiento, escritas para un lenguaje de programación, que tienen una interfaz bien definida para el comportamiento que se invoca.

- **Middleware**: es un software que asiste a una aplicación para interactuar o comunicarse con otras aplicaciones, software, redes, hardware y/o sistemas operativos. Éste simplifica el trabajo de los programadores en la compleja tarea de generar las conexiones que son necesarias en los sistemas distribuidos. De esta forma se provee una solución que mejora la calidad de servicio, seguridad, envío de mensajes, directorio de servicio, etc.

- **Portabilidad**: es la capacidad de uso del mismo software en diferentes entornos. Cuando el software con la misma funcionalidad es producido para varias plataformas de computación, la portabilidad es la cuestión clave para la reducción de los costes de desarrollo.

- **Protocolos**: es el conjunto de reglas y estándares que controlan la secuencia de mensajes que ocurren durante una comunicación entre entidades que forman una red, como teléfonos o computadoras.

- **Proxy**: es un programa o dispositivo que realiza una acción en representación de otro, esto es, si una hipotética máquina A solicita un recurso a una C, lo hará mediante una petición a B; C entonces no sabrá que la petición procedió originalmente de A. Esta situación estratégica de punto intermedio suele ser aprovechada para soportar una serie de funcionalidades: proporcionar caché, control de acceso, registro del tráfico, prohibir cierto tipo de tráfico, etc.

- **Recurrente**: un método usual de simplificación de un problema complejo es la división de este en subproblemas del mismo tipo. Esta técnica de programación se conoce como divide y vencerás y es el núcleo en el diseño de numerosos algoritmos de gran importancia, así como también es parte fundamental de la programación dinámica.

- **Reutilización de software**: la reutilización de software aparece como una alternativa para desarrollar aplicaciones y sistemas SW de una manera más eficiente, productiva y rápida. La idea es reutilizar elementos y componentes SW en lugar de tener que desarrollarlos desde un principio.

- **Robustez**: capacidad y proceso de reacción apropiada ante condiciones que se encuentren fuera del alcance del software, estas condiciones son excepcionales.

- **URI**: identificador uniforme de recursos es una cadena de caracteres corta que identifica inequívocamente un recurso (servicio, página, documento, dirección de correo electrónico, enciclopedia, etc.). Normalmente estos recursos son accesibles en una red o sistema. Los URI pueden ser localizadores uniformes de recursos (URL).

- **URL**: localizador uniforme de recursos, es una secuencia de caracteres, de acuerdo a un formato modélico y estándar, que se usa para nombrar recursos en Internet para su localización o identificación, como por ejemplo documentos textuales, imágenes, vídeos, presentaciones digitales, etc.

INTRODUCCIÓN

La venta de dispositivos móviles continúa imparable. Cada vez son más los empresarios que buscan desarrolladores con la capacidad de transformar sus ideas en aplicaciones móviles. Crear una aplicación móvil empieza a convertirse en algo imprescindible para que las herramientas web ya existentes se adapten a lo que demanda la sociedad, y puedan dar respuestas inmediatas a sus usuarios. Para el éxito en este campo, hace falta más que una gran idea para crear una aplicación que de beneficios. De hecho, muchas ideas que podrían haberse convertido en grandes aplicaciones, se han quedado solo en propuestas.

Dentro de las herramientas web de e-learning más usadas en la Facultad de Ingeniería de la Universidad de Nariño se encuentra Chamilo, con la cual se pretende responder de un modo sencillo las necesidades de los usuarios y tome distintas formas para adaptarse al flujo de trabajo de las actividades que se desarrollan en el campo educativo. Chamilo, es un sistema web que organiza procesos de enseñanza y aprendizaje a través de contenidos instruccionales e interacciones colaborativas.

Por ello, se empieza a trazar una estrategia para sacar ventajas de esta herramienta se utilizan los dispositivos móviles. La herramienta móvil puede llegar a convertirse en un buen canal de comunicación con los usuarios de Chamilo web. De manera instantánea y desde cualquier lugar, los usuarios pueden acceder a las funcionalidades consumidas por la aplicación móvil, y sincronizarse con la plataforma web al contar con conexión a internet.

Debido a lo anterior, desarrollar CHAMILOMOBILE como un instrumento indispensable que debe formar parte de las estrategias educativas para conseguir que los usuarios aprovechen de mejor manera las herramientas que están a su disposición, subestimar la presencia móvil en este campo supone perder un importante canal de comunicación y difusión, y con él un gran porcentaje de público objetivo.

En el presente documento se muestra la propuesta para el desarrollo de la aplicación CHAMILOMOBILE y se organiza de la siguiente manera: en principio se muestra la necesidad de integrar la herramientas de aprendizaje web a entornos móviles, seguido del planteamiento del problema y sistematización, objetivos, justificación, antecedentes relacionados con la construcción de aplicaciones para utilizar la plataforma Chamilo en entornos móviles, la metodología de desarrollo, resultados esperados y recursos a utilizar a lo largo del trabajo.

PLANTEAMIENTO DEL PROBLEMA

La computación en la nube y las aplicaciones móviles van de la mano, hoy son prácticamente inseparables. En el momento, la Facultad de Ingeniería de la Universidad de Nariño cuenta con la aplicación Web Chamilo que está a disposición de la comunidad académica como herramienta de aprendizaje y colaboración en las diferentes actividades, su actual presentación estable es Chamilo LMS 1.8.8.4. Si bien es una herramienta muy usada y requerida por la comunidad académica, se evidencia la necesidad de ampliar las opciones de acceso desde entornos móviles, con el objetivo de disminuir las limitaciones que se presentan en cuanto a tiempo, espacio y facilidad de acceso.

El dispositivo móvil ha evolucionado hasta el punto de ser un recurso indispensable en cualquier ámbito. Este cambio se ha dado en un corto espacio de tiempo, en el que han pasado de utilizarse como teléfonos portátiles a usarlos para cualquier tarea imaginable. Posiblemente en los próximos años, este crecimiento va a seguir siendo tan amplio como lo es ahora.

Las tecnologías que en este momento están en desarrollo y se empiezan a aplicar, tienen relevancia en este trabajo, ya que es necesario su estudio para implementar servicios eficientes, robustos y acordes al entorno de desarrollo. Por esta razón el primer paso para integrar las aplicaciones móviles a la nube genera grandes desafíos.

La habilitación móvil por ejemplo, implica acceder a los datos y a la lógica de negocio de las aplicaciones web, pero desafortunadamente pocas brindan los API o los servicios para desarrollar las nuevas interfaces que el usuario móvil requiere. Para reducir la carga de trabajo que significa alterar las aplicaciones existentes, una de las soluciones es aprovechar las capacidades de los dispositivos móviles al utilizar una interfaz de conexión entre el dispositivo y la aplicación web.

La lógica de negocio de la aplicación no debería codificarse en el dispositivo móvil, esto implica tener que administrar códigos inmanejables y duplicados de los mismos procesos de la aplicación web en múltiples aplicaciones móviles. Tener esta lógica de negocio concentrada en el lado del servidor hace que la arquitectura sea más simple y la aplicación más eficiente. La creación de una capa de lógica de negocio en el lado del servidor que pueda combinar, orquestar y computar los datos desde muchas fuentes es una idea inteligente.

El desarrollo de una aplicación para el sistema operativo Android, junto a otras tecnologías, permitirá acceder a las bases de datos de la aplicación web Chamilo, para ofrecer a los usuarios amplios beneficios en el desarrollo de sus actividades académicas.

Formulación del problema

¿Cómo extender a un entorno móvil funcionalidades de la plataforma Chamilo LMS usada por la comunidad académica de la Facultad de Ingeniería de la Universidad de Nariño?

Sistematización del problema

- ¿Cuáles son las funcionalidades más importantes para los usuarios de la comunidad académica actual?
- ¿Cómo optimizar y simplificar la conexión entre un dispositivo móvil y la aplicación web Chamilo LMS?
- ¿Cómo aportar en las necesidades de seguimiento en el uso de estas herramientas?

Alcance y delimitaciones

La aplicación móvil contendrá funcionalidades escalables que permitan obtener una interfaz de usuario conectada a un API REST, para poder acceder a la base de datos de la aplicación web Chamilo LMS de la Facultad de Ingeniería de la Universidad de Nariño, y así mostrar algunos módulos teniendo en cuenta lo siguiente:

- El Servicio API REST y la aplicación móvil se desarrollarán en los lenguajes de programación PHP y Java respectivamente.
- CHAMILOMOBILE, ofrece funcionalidades que permitirán la interacción de los usuarios de Chamilo LMS por medio de un dispositivo móvil con sistema operativo Android.
- Los módulos que se visualizarán desde CHAMILOMOBILE son: Ejercicios y Mi Agenda, tanto para profesores como para estudiantes.
- Los módulos que generan transacciones desde CHAMILOMOBILE son: Descripción, Documentos, Enlaces, Anuncios, Evaluaciones, Glosario, Informes, Lista de Usuarios, Tareas, Compartir documentos, Notas personales, Mensajes y Perfil, tanto para profesores como para estudiantes.
- Para la interacción con el aplicativo web Chamilo LMS se implementará una interfaz con arquitectura orientada al servicio para la conexión a la base de datos.
- Se desarrollará un módulo de indicadores de uso del LMS Chamilo.

Líneas de investigación

Las líneas de investigación en las que se enmarca este trabajo de grado son las correspondientes a: Sistemas Computacionales y Software y manejo de Información

Modalidad

Este trabajo de grado se enmarca en la modalidad de Trabajo de Aplicación.

OBJETIVOS

Objetivo general

Desarrollar una aplicación móvil integrada a la plataforma Chamilo LMS de la Facultad de Ingeniería de la Universidad de Nariño, para extender su usabilidad y funcionalidad, permitiendo al usuario conectarse de forma simple y eficiente, incluyendo indicadores de uso.

Objetivos específicos

- Identificar las funcionalidades relevantes para el usuario de Chamilo LMSweb para implementar los módulos de la aplicación móvil.

- Establecer una interfaz de conexión que permita la interacción entre la aplicación móvil y la aplicación Chamilo LMS web

- Dotar de una herramienta de medición para que el usuario pueda medir la frecuencia de uso de algunas funcionalidades de Chamilo Mobile.

JUSTIFICACIÓN

La tendencia actual sitúa el mercado de las aplicaciones móviles como un fuerte aliciente para que los desarrolladores se inclinen por implementar aplicaciones con tecnologías nuevas asumiendo el esfuerzo que conlleva el aprendizaje de ellas y en ocasiones cerradas a unos pocos dispositivos.

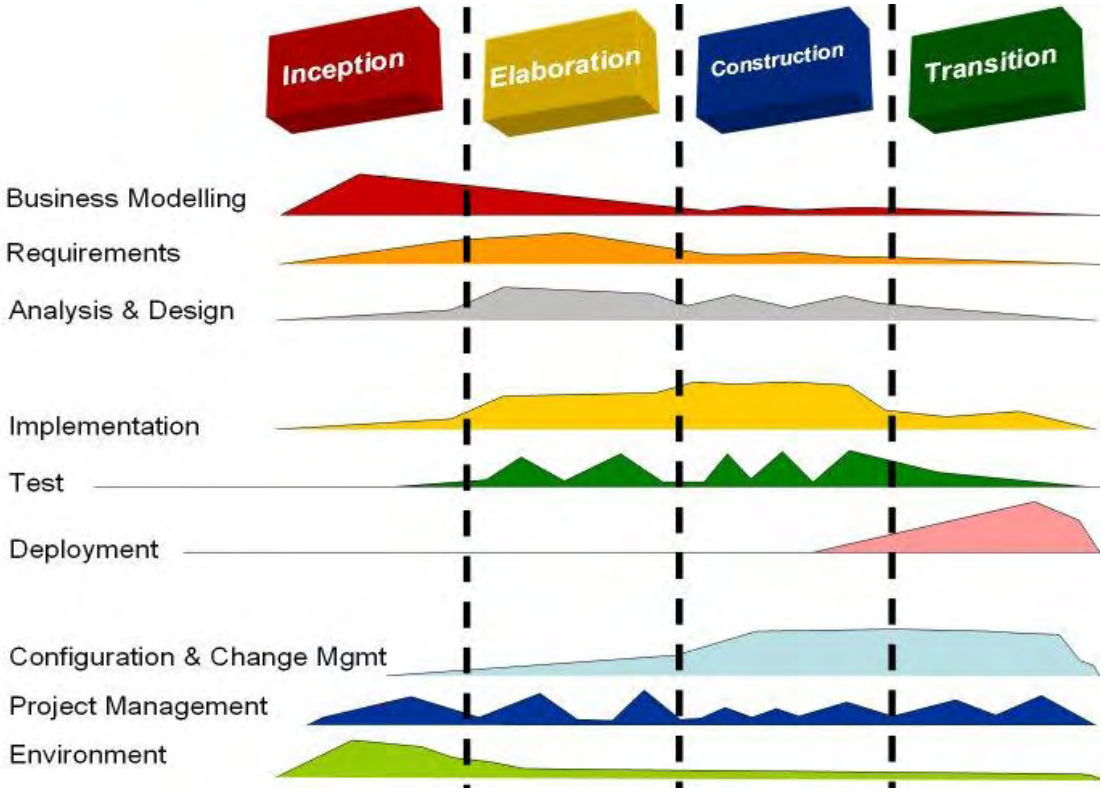
Muchos de los desarrolladores de Android vienen del mundo de Java dada la ligera curva de aprendizaje. La tendencia actual indica que el número de desarrolladores Android crecerá, dado el aumento de dispositivos que llevan el sistema operativo Android y la libertad de creación marcada desde un principio por Google. Si bien las aplicaciones web muestras grandes ventajas, se debe pensar que las funcionalidades ya disponibles puedan mostrarse en diferentes dispositivos, para lo cual se justifica el desarrollo de aplicaciones móviles que contribuyan a la comodidad y eficiencia de los servicios que se prestan a los usuarios.

Para la construcción de una aplicación móvil, se tuvo en cuenta las diferentes necesidades de los usuarios que actualmente utilizan la herramienta Chamilo LMS Web para establecer las condiciones de diseño de las etapas en la cual se escribe el código fuente necesario para dar funcionamiento a la pieza de software móvil.

METODOLOGÍA

Para el desarrollo de CHAMILOMÓVIL, se ha escogido al proceso de software RUP (Proceso Unificado Rational) el cual propone el desarrollo de software basado en un conjunto de fases, cada una de las cuales se desarrolla por medio de una o varias iteraciones donde por cada iteración se desarrollan un conjunto de actividades llamadas flujo de trabajo y flujo de soporte, a continuación se muestra un esquema para RUP en la figura 1.

Figura 1. Esquema RUP



En la figura 1, el eje horizontal representa el tiempo y muestra los aspectos del ciclo de vida del proceso. El eje vertical representa las disciplinas que agrupan actividades por su naturaleza.

RUP, está centrado en la arquitectura, dirigido por Casos de Uso, iterativo e incremental. RUP se dice que es centrado en arquitectura por que el trabajo se centra en: el desarrollo de software, identificación de componentes software y definición de funcionalidad de módulos, gracias a que se definen por medio de requisitos.

Para lo cual en este estudio se escogió el proceso RUP y las fases a seguir se contemplan en el cronograma a seguir. Como la metodología RUP se destaca por ser iterativa e incremental se sacara por cada iteración un producto que siga la línea base del ciclo de vida del software así:

Para el modelado del negocio se hará estudio sobre las entidades y procesos dentro de la plataforma Chamilo LMS en su versión web.

Los requerimientos se desprendieron de los objetivos y alcances de este proyecto en paralelo al punto antes descrito.

El proceso de análisis y diseño se llevó de manera continua a lo largo de todo el desarrollo y se apoyó en bitácoras de avance del proyecto.

La codificación se realizó en el lenguaje nativo del sistema operativo Android, basándose en el análisis y diseño descrito en el punto anterior.

En la etapa de pruebas se aplicaron únicamente pruebas de unidad.

El despliegue se hizo inmediatamente después de cada iteración que arroje un producto funcional.

El resto de etapas serán transversales a los puntos anteriores y serán apoyadas mediante bitácoras de avance.

1. MARCO TEÓRICO

1.1 INTRODUCCIÓN

Aunque los primeros teléfonos inteligentes denominados Smartphones surgieron a finales de la década de los noventa y ya contaban con aplicaciones precargadas en su sistema, éstas eran de diseño simple y realizaban funciones básicas muy alejadas de las capacidades que alcanzan hoy en día. Estos primeros Smartphones fueron diseñados para proporcionar acciones propias de una agenda electrónica de bolsillo muy extendida en esa década pero permitiendo además las llamadas de voz y el envío de mensajes SMS, por lo que las aplicaciones que llevaban preinstaladas se limitaban a gestionar contactos, llevar una agenda y algunos juegos. Por otra parte, estos teléfonos estaban restringidos a desarrolladores externos limitados por sus sistemas operativos, realizados por los propios fabricantes.

Con la mejora de las capacidades para la conexión a internet de los teléfonos móviles a principios de la década del 2000 y tras la presentación del primer Iphone en 2007 la importancia de las aplicaciones móviles y su despegue como negocio ha sido imparable.

En 2008, Apple presentaba una tienda virtual de aplicaciones llamada App Store, no sólo poniendo a disposición de los compradores de Iphone las aplicaciones desarrolladas por la compañía, sino abriendo a terceros la posibilidad de crear software para dispositivos móviles por su cuenta que pueden subir a la tienda y del que obtienen el 70% de los beneficios que se generan. Este nuevo modelo de mercado en el mundo del desarrollo mediante el cual la compañía de Cupertino ingresa el 30% de las ventas de las aplicaciones reservándose además el derecho a la revisión de las mismas para controlar su calidad, ha sido adoptado posteriormente por otras compañías como Google para sus dispositivos móviles con sistema operativo Android, o Microsoft, uno de los últimos en incorporarse a esta tendencia pero que además la ha introducido también a sus versiones del sistema operativo para computadores portátiles y de escritorio.

El gran auge de las aplicaciones no se ha aprovechado solamente por desarrolladores para buscar negocio donde se aportan funcionalidades a los usuarios, sino que también las empresas, las instituciones y diferentes tipos de organismos han visto en ellas un modo de vender sus productos, promocionar sus servicios y comunicar su información de un modo rápido, directo y cómodo para el consumidor. Es en este último caso en el que se incluye la situación de la aplicación objeto de este trabajo fin de grado.

El LMS Chamilo Web que usa actualmente la Universidad de Nariño, genera grandes beneficios a la comunidad educativa y para buscar mejorar estos beneficios se planteó la posibilidad de crear una aplicación móvil mediante la cual los usuarios puedan interactuar con esta plataforma desde sus dispositivos móviles.

El desarrollo de CHAMILOMOBILE se soportó en tecnologías de última generación, que permiten que el sistema sea robusto y configurable, además con la implementación y uso de las herramientas teóricas se tiene como finalidad dar una solución óptima en el uso de la plataforma Web.

A continuación, se realiza una descripción de los temas principales que se abordan en CHAMILOWEB.

1.2 ANTECEDENTES

En la actualidad se encuentran diversas propuestas para Chamilo mobile. En esta sección se referencia algunas de estas propuestas con el propósito de acoger experiencia y conocimiento que sirvan como base para esta investigación.

En el ámbito internacional.

- CHAMILO MOBILE ROADMAP, es una propuesta que pretendía adaptar la aplicación web a un entorno móvil, presentado por Jose Arturo Mora y Roberto Gonzales, España, 2013. Aunque solo quedo en propuesta permite confirmar la necesidad de desarrollar una herramienta que beneficie a los usuarios de Chamilo LMS Web.
- CHAMILO MOBILE FOR STUDENTS: propuesta de desarrollo de una aplicación móvil para utilizar Chamilo LMS como alumno. Presentada por Paulo Cesar Salazar en el año 2013 y dirigida por Jose Arturo Mora. Esto ratifica que existe interés por el desarrollo de una aplicación móvil para Chamilo LMS pero que solo han quedado en propuestas, lo que motiva a llegar hasta su implementación.

En el ámbito nacional.

En Colombia se realizó una investigación la cual arrojó un resultado negativo en cuanto al desarrollo e implementación de aplicaciones móvil basadas en Chamilo LMS Web.

En el ámbito regional.

En la región también se obtienen resultados no favorables en cuanto al desarrollo de aplicaciones relacionadas con el estudio actual.

1.3 APLICACIÓN MÓVIL

1.3.1 Generalidades. Las aplicaciones móviles están presentes en los teléfonos desde hace tiempo; de hecho, ya estaban incluidas en los sistemas operativos años atrás. En esencia, una aplicación no deja de ser un software. Para entender un poco mejor el concepto, se puede decir que las aplicaciones son para los móviles lo que los programas son para los ordenadores de escritorio. Actualmente hay aplicaciones de todo tipo, forma y color, pero en los primeros teléfonos, estaban enfocadas en mejorar la productividad personal: se trataba de alarmas, calendarios, calculadoras y clientes de correo. Hubo un cambio grande con el ingreso de nuevos productos al mercado, ya que se generaron nuevos modelos de negocio que hicieron que las aplicaciones sean más rentables, tanto para desarrolladores como para los mercados de aplicaciones, como App Store, Google Play y Windows Phone Store. Al mismo tiempo, también mejoraron las herramientas de las que disponían diseñadores y programadores para desarrollar aplicaciones, se facilita la tarea de producir una aplicación y lanzarla al mercado, incluso por cuenta propia.

1.3.1.1 Que es una aplicación móvil. Es posible que cuando llegue la hora de diseñar una aplicación ya exista una web como antecedente. En esos casos, la aplicación tiene que tomar las funciones y contenidos que se han pensado para la web y adaptarlos para que tengan sentido, de acuerdo al tamaño de pantalla y a la forma de interacción de un móvil. En otros casos, el diseño comienza desde cero, cuando todavía no hay ni web ni aplicación, y hay que decidirse por cuál de ellas empezar. Aquí es donde adquiere más trascendencia el concepto de mobile first, que implica plantear el proceso de diseño teniendo en cuenta el móvil en primer lugar.

Una vez que la aplicación está diseñada, puede preguntarse cuál es la mejor forma de llevar lo hecho para el teléfono a una pantalla de computador o a otros dispositivos, extendiendo y escalando el contenido y repensando la diagramación. Todos los dispositivos tienen usos diferentes, y en el momento de adaptar el diseño, hay que tener en cuenta las características particulares de cada uno de ellos.

1.3.1.2 El proceso de diseño y desarrollo de una aplicación. El proceso de diseño y desarrollo de una aplicación móvil, abarca desde la concepción de la idea hasta el análisis posterior a su publicación en las tiendas.

Cada una de las etapas de desarrollo se explica con detalle los procesos y metodologías para avanzar entre ellas.

- **Conceptualización**. El resultado de esta etapa es una idea de aplicación, que tiene en cuenta las necesidades y problemas de los usuarios. La idea responde

a una investigación preliminar y a la posterior comprobación de la viabilidad del concepto.

- **Definición**. En este paso del proceso se describe con detalle a los usuarios para quienes se diseñará la aplicación. También aquí se sientan las bases de la funcionalidad, lo cual determinará el alcance del proyecto y la complejidad de diseño y programación de la aplicación.

- **Diseño**. En la etapa de diseño se llevan a un plano tangible los conceptos y definiciones anteriores, y posteriormente, en un diseño visual acabado que será provisto al desarrollador, en forma de archivos separados y pantallas modelo, para la programación del código.

- **Desarrollo**. El programador se encarga de dar vida a los diseños y crear la estructura sobre la cual se apoyará el funcionamiento de la aplicación. Una vez que existe la versión inicial, hay que dedicarse gran parte del tiempo a corregir errores funcionales para asegurar el correcto desempeño de la aplicación y se prepara para su aprobación en las tiendas.

- **Publicación**. La aplicación es finalmente puesta a disposición de los usuarios en las tiendas. Luego de este paso trascendental se realiza un seguimiento a través de analíticas, estadísticas y comentarios de usuarios, para evaluar el comportamiento y desempeño de la aplicación, corregir errores, realizar mejoras y actualizarla en futuras versiones.

1.3.1.3 Tipos de aplicaciones según su desarrollo. A nivel de programación, existen varias formas de desarrollar una aplicación. Cada una de ellas tiene diferentes características y limitaciones, especialmente desde el punto de vista técnico.

- **Aplicaciones nativas**. Las aplicaciones nativas son aquellas que han sido desarrolladas con el software que ofrece cada sistema operativo a los programadores, llamado genéricamente Software Development Kit o SDK. Así, Android, iOS y Windows Phone tienen uno diferente y las aplicaciones nativas se diseñan y programan específicamente para cada plataforma, en el lenguaje utilizado por el SDK.

  Las aplicaciones nativas se actualizan frecuentemente y en esos casos, el usuario debe volver a descargarlas para obtener la última versión, que a veces corrige errores o añade mejoras.

  Una característica generalmente menospreciada de las aplicaciones nativas, es que pueden hacer uso de las notificaciones del sistema operativo para mostrar

avisos importantes al usuario, aun cuando no se esté usando la aplicación, como los mensajes de Whatsapp, por ejemplo.

Además, no requieren Internet para funcionar, porque ofrecen una experiencia de uso más fluida y están realmente integradas al teléfono, lo cual les permite utilizar todas las características de hardware del terminal, como la cámara y los sensores (GPS, acelerómetro, giróscopo, entre otros).

A nivel de diseño, esta clase de aplicaciones tiene una interfaz basada en las guías de cada sistema operativo, donde se logra mayor coherencia y consistencia con el resto de aplicaciones y con el propio SO. Esto favorece la usabilidad y beneficia directamente al usuario que encuentra interfaces familiares.

- **Aplicaciones web**. La base de programación de las aplicaciones web —también llamadas webapps es el HTML, conjuntamente con JavaScript y CSS. En este caso no se emplea un SDK, lo cual permite programar de forma independiente al sistema operativo en el cual se usará la aplicación. Por eso, estas aplicaciones pueden ser fácilmente utilizadas en diferentes plataformas sin mayores inconvenientes y sin necesidad de desarrollar un código diferente para cada caso particular.

  Las aplicaciones web no necesitan instalarse, ya que se visualizan a través navegador del teléfono como un sitio web normal. Por esta misma razón, no se distribuyen en una tienda de aplicaciones, sino que se comercializan y promocionan de forma independiente.

  Al tratarse de aplicaciones que funcionan sobre la web, no es necesario que el usuario reciba actualizaciones, ya que siempre va a estar viendo la última versión. Pero, a diferencia de las aplicaciones nativas, requieren de una conexión a Internet para funcionar correctamente.

  Adicionalmente, tienen algunas restricciones e inconvenientes en factores importantes como gestión de memoria y no permiten aprovechar al máximo la potencia de los diferentes componentes de hardware del teléfono.

  Las aplicaciones web suelen tener una interfaz más genérica e independiente de la apariencia del sistema operativo, por lo que la experiencia de identificación del usuario con los elementos de navegación e interacción, suele ser menor que en el caso de las nativas.

- **Aplicaciones híbridas**. Este tipo de aplicaciones es una especie de combinación entre las dos anteriores. La forma de desarrollarlas es parecida a la de una aplicación web con el uso de HTML, CSS y JavaScript, y una vez que

la aplicación está terminada, se compila o empaqueta de forma tal, que el resultado final es como si se tratara de una aplicación nativa.

Esto permite casi con un mismo código obtener diferentes aplicaciones, por ejemplo, para Android y iOS, y distribuirlas en cada una de sus tiendas.

A diferencia de las aplicaciones web, estas permiten acceder, con el uso de librerías, a las capacidades del teléfono, tal como lo haría una aplicación nativa.

Las aplicaciones híbridas, también tienen un diseño visual que no se identifica en gran medida con el del sistema operativo. Sin embargo, hay formas de usar controles y botones nativos de cada plataforma para apegarse más a la estética propia de cada una.

Existen algunas herramientas para desarrollar este tipo de aplicaciones. Apache Cordova es una de las más populares, pero hay otras que tienen la misma finalidad.

1.3.2 Arquitecturas diseñadas para una aplicación móvil. Los sistemas operativos móviles son mucho más simples y están más orientados a la conectividad inalámbrica, los formatos multimedia para móviles y las diferentes maneras de introducir información en ellos.

Aplicaciones de un Cliente Delgado. En el modelo cliente inteligente, los datos son almacenados localmente y por lo tanto se pueden realizar las consultas y operaciones de la BD en el dispositivo móvil.

Sin embargo, los datos deben ser actualizados y sincronizados con los del servidor de Data Server. Las actualizaciones se hacen en los dos sentidos:

- Cliente-servidor: los datos capturados en el cliente son enviados al servidor de datos.
- Servidor-cliente: los cambios realizados en el servidor son enviados al cliente para actualizar los datos locales.

El tipo de conexión afecta la sincronización de los datos entre el cliente y el servidor.

Las arquitecturas implementadas en el desarrollo de aplicaciones móviles, se deben al tipo de información el cual se va a impartir; hay que tener en cuenta la escalabilidad y todas sus ventajas. Las aplicaciones móviles según la información que imparten se dividen en dos grupos, aplicaciones móviles autocontenidas y las aplicaciones con conexión a Internet.

- **Aplicaciones móviles autocontenidas**. En este tipo de aplicaciones el contenido es estático, sus imágenes, su información, sus menús, casi nunca cambia o rara vez lo hace. Esto es debido a que todo su contenido se encuentra dentro de la misma aplicación, no necesita valerse de una herramienta o un servicio para funcionar. Este tipo de aplicaciones son del tipo nativo, debido a que brinda una serie de herramientas y posibilidades a la hora de desarrollar la aplicación móvil de este tipo, un ejemplo de este tipo de aplicaciones seria por ejemplo una calculadora desarrolla para Android, iOS, Windows Phone, entre otros; la información puede ser suministrada a través de un manual en alguna función u opción del menú, y si necesita brindar herramientas como tema puede predeterminar colores tamaños y demás cosas al respecto.

- **Aplicaciones móviles con conexión a Internet**. En este tipo de aplicaciones la información está alojada en un back-end, el cual cuenta con una base de datos, que accedida a través de un API de servicios web, como SOAP o REST, los cuales se comunican a la base de datos con el front-end de la aplicación, que es donde se puede ver la información solicitada.

  Un ejemplo claro son las aplicaciones de las redes sociales, como facebook y twitter, aplicaciones de mensajería instantánea como line, viber, whatsapp; necesitan el servicio de Internet para poder acceder a información específica o realizar alguna acción. Dentro de este tipo de aplicaciones, se encuentran dos tipos, los cuales son con descarga de información estática y con información dinámica.

  Aplicaciones móviles con descarga de información estática. Cuenta con un archivo en formato XML, que se modifica de forma manual cuando se requiera cambiar información.

  Aplicaciones móviles con información dinámica. Este tipo de aplicaciones se llama así, debido a que la información esta almacenada en una base de datos la cual está alojada en un servidor y que se accede a esta a través de servicios API los cuales se comunican a la base con la interfaz de la aplicación.

1.3.2.1 Componentes de los SO móviles. Un sistema operativo móvil se encuentra compuesto por varias capas.

El núcleo o Kernel, es la capa de software que permite el acceso a los diferentes elementos de hardware que conforman el dispositivo móvil. Encargado de brindar diferentes servicios a las capas superiores como los controladores de hardware,

gestión de procesos, sistemas de archivos, además del acceso y administración de la memoria del sistema.
Los sistemas operativos para móviles pueden basarse en núcleos Linux, tal como lo hace Android, o hasta inclusive IOS, el SO del iPhone utiliza un kernel heredado de Unix.

En el caso de Android los sistemas operativos presentan la particularidad de contar con un motor Java en el desarrollo de sus núcleos.

Middleware. Conjunto de módulos que permite que las aplicaciones diseñadas y escritas para tales plataformas puedan ser ejecutadas. Su funcionamiento es totalmente trasparente para el usuario, no debiendo realizar ninguna acción ni configurar alguna para su correcto desenvolvimiento.

Entre los servicios que presta esta capa se pueden citar los motores de comunicaciones y mensajería, funciones de seguridad, servicios para la gestión de diferentes aspectos del móvil, ofrece servicios claves como el motor de mensajera y comunicaciones, codecs multimedia, intérpretes de páginas Web y servicios WAP, además de soporte para una gran variedad de servicios concernientes al apartado multimedia que es capaz de ejecutar el móvil.

1.4 ANDROID

Android es un sistema operativo diseñado por Google para teléfonos móviles o smartphones basado en el SO Linux.

La mayoría de las aplicaciones Android se encuentran disponibles en Google Play, antes conocida como Android Market, aunque también existen desarrolladores que crean aplicaciones que puedes encontrar en sitios web independientes. En este último caso, es recomendable estar seguro de que el sitio web en cuestión es confiable para evitar la instalación de virus o malware en general en los dispositivos Android.

Gran parte de los móviles Android disponen de una aplicación llamada Play Store, esta aplicación te permite conectarte directamente a Google Play para navegar entre las aplicaciones disponibles e instalar las que te interesen.

En la actualidad existe un gran número de desarrolladores que escriben aplicaciones específicas para la plataforma Android hasta el punto de que el número de aplicaciones que puedes encontrar actualmente supera las 400.000, gran parte de las cuales son de uso gratuito. Aquellas aplicaciones de pago reparten su costo entre Google y el desarrollador con un porcentaje del 30% y 70% respectivamente.

Las aplicaciones se desarrollan habitualmente en el lenguaje Java con Android Software Development Kit (Android SDK)[***]. Pero están disponibles otras herramientas de desarrollo, incluyendo un Kit de Desarrollo Nativo para aplicaciones o extensiones en C o C++, Google App Inventor y un entorno visual para programadores novatos. El desarrollo de aplicaciones para Android no requiere aprender lenguajes complejos de programación. Todo lo que se necesita es un conocimiento aceptable de Java y estar en posesión del kit de desarrollo de software o SDK.

Arquitectura de Android. En el caso de Android está formada por varias capas que facilitan al desarrollador la creación de aplicaciones. Además, esta distribución permite acceder a las capas más bajas mediante el uso de bibliotecas para que así el desarrollador no tenga que programar a bajo nivel las funcionalidades necesarias para que una aplicación haga uso de los componentes de hardware de los teléfonos. En la figura 2, se representa la Arquitectura.

Figura 2. Arquitectura Android

1.4.1 Código abierto. Android está basado en Open Source. Esto significa que el código del sistema y el software en general que lo compone está disponible para ser consultado y modificado libremente por cualquiera que lo desee.

La ventaja inmediata es que un programador que tenga interés en hacerlo puede adaptar Android para que funcione en cualquier equipo informático o dispositivo que desee. Este es uno de los puntos clave para que Android haya extendido tanto su uso a lo largo de los últimos años.

[***]Android SDK: es el proceso por el cual se crean nuevas aplicaciones para el sistema operativo Android.

1.4.2 Características

- Los móviles Android son baratos. Este punto es relativo y, seguramente, polémico. El hecho de que sea un sistema operativo de libre uso significa que se puede encontrarlo en dispositivos de todo tipo y precio.
- Personalización. Android puede personalizarse ampliamente, eso implica que cada fabricante puede adaptarlo como mejor le parezca y que los programadores tienen bastante libertad de movimiento a la hora de personalizarlo.
- Gran número de aplicaciones. Existen más de 400.000 aplicaciones diferentes para Android.
- Android es fácilmente transportable. Se puede encontrar dispositivos Android de todo tipo, desde smartphones y tablets hasta reproductores multimedia. Un gran número de fabricantes lanzan diferentes productos basados en Android, así que la oferta que existe es realmente amplia lo cual lleva a que sea relativamente fácil encontrar justo aquello que se necesita.

Características técnicas de Android

- Núcleo basado en Linux.
- Máquina virtual basada en Java para la ejecución de aplicaciones.
- Navegador integrado basado en WebKit.
- Soporte para formatos multimedia, audio y vídeo: WebM, H.263, H.264, MPEG-4 SP, AMR, AMR-WB, AAC, HE-AAC, MP3, MIDI, Ogg Vorbis, WAV, JPEG, PNG, GIF y BMP.
- Gráficos optimizados 2D/3D.
- Soporte para Bluetooth, Wi-Fi, 3G y EDGE.
- Soporte para GPS, brújulas y acelerómetros.

1.5 INTERACCIÓN

El desarrollo actual de la tecnología favorece la creación y el enriquecimiento de las propuestas de educación, en tanto que permiten abordar de manera ágil muchos tratamientos de temas, así como generar nuevas formas de encuentros entre docentes y alumnos, y por ende de alumnos entre sí.

Existes varias técnicas para lograr establecer comunicación entre plataformas, entre las cuales están los servicios web.

1.5.1 Servicios Web. La alta conectividad entre computadores ha sido una meta desde que comenzó la informática personal. Con el auge de las redes internas dentro de las empresas llega el deseo de unir máquinas de forma programática.

Es decir, un programa en una máquina debería poder llamar a métodos del programa en otra máquina sin la necesidad de intervención humana, para lo que se utilizaron diferentes tecnologías. Si se intenta centrar el estado actual del desarrollo de aplicaciones basadas en Web, se puede encontrar una gran cantidad de tecnologías, muchas de ellas incompatibles entre sí.

Se encuentra también que como la Internet se ha convertido en una herramienta de trabajo habitual, actualmente no es más que una fuente de datos y no de servicios dirigidos a facilitar el trabajo del usuario.

En este sentido, los servicios que ofrecen las nuevas tecnologías deberían cooperar para beneficio de los usuarios. Los sitios Web aislados y los diferentes dispositivos deberían trabajar juntos para ofrecer soluciones mucho más valiosas. Se trata de ofrecer a través de Internet no sólo datos, sino también software y servicios que puedan ser fácilmente accesibles, servicios que integren y busquen la información que se necesita, pudiendo acceder a esta información en cualquier momento y desde cualquier dispositivo.

Figura 3. Proceso de servicios Web

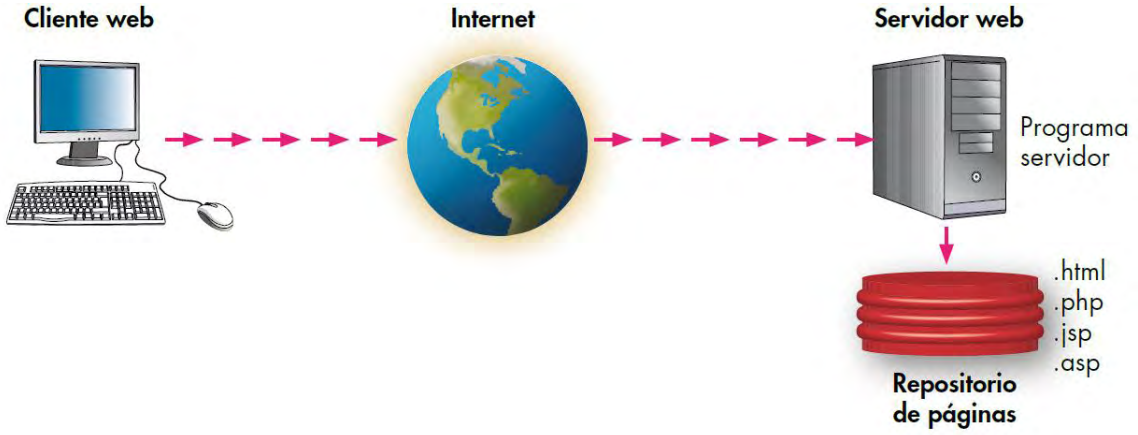
¿Qué es un servicio Web?

Un servicio Web es un componente de software independiente de plataforma e implementación, que lleva a cabo un servicio concreto y que puede integrarse con otros servicios Web para dar un servicio diferente. Los servicios Web se auto-describen y auto-exponen, permitiendo a los consumidores (aplicaciones clientes) localizarlos en Internet, para ser invocados y escuchados sobre protocolos estándar.

Los servicios Web son aplicaciones que utilizan estándares para el transporte, codificación y protocolo de intercambio de información que permiten la intercomunicación entre sistemas de cualquier plataforma y se utilizan en una gran variedad de escenarios de integración.

Normalmente los servicios Web no son para uso público o no están fácilmente accesibles. Son las aplicaciones las que acceden internamente para transmitir datos. Es por ello que la seguridad no suele ser muy buena ya que el programador piensa que sólo su aplicación va a consumir los servicios, pero realmente el servicio Web se encuentra en Internet esperando a que cualquiera conecte con él para solicitarle datos. En la figura 4, se muestra el funcionamiento básico de servicios Web.

Figura 4. Esquema básico de funcionamiento del servicio Web



El concepto de servicio Web se apoya en los estándares HTML y XML. El desarrollador puede crear programas accesibles desde cualquier dispositivo que soporte estos estándares, aprovechando la conectividad de Internet. Se pueden crear servicios accesibles desde Internet que realmente proporcionen una utilidad real.

Ventajas de los servicios Web. A continuación se muestran las ventajas que conlleva el uso de los servicios Web en el desarrollo de las aplicaciones.

- **Interoperabilidad**. Nuevas relaciones pueden ser construidas dinámicamente y automáticamente ya que los servicios Web aseguran una interoperabilidad completa entre sistemas. Cualquier servicio Web puede interactuar con cualquier otro servicio Web o cliente, gracias a que la comunicación entre ambos se lleva a cabo en XML vía Internet (HTTP). Un servicio Web podrá estar escrito en cualquier plataforma o lenguaje que soporte estos estándares, lo cual no importará para su utilización o integración.

- **Fácil implementación**. Los conceptos en los que se basan los servicios Web son fácilmente entendibles y actualmente existen herramientas que permiten desarrollar y crear un servicio Web prácticamente teniendo sólo algunas nociones de programación.

- **Accesibilidad**. Los servicios Web pueden ser completamente descentralizados y distribuidos sobre Internet y accedidos a través de una gran variedad de dispositivos.

- **Especificaciones universalmente aceptadas**. Los servicios Web se basan en especificaciones estándar para el intercambio de datos, mensajería, búsqueda, descripción de la interfaz y coordinación de los procesos.

- **Integración**. Los servicios Web proporcionan mayor agilidad y flexibilidad debido a una mejor integración con los sistemas existentes.

Operaciones de Servicios Web:

- **Publicar/Cancelar**. Los proveedores de servicios publican la disponibilidad de su servicio a uno o más registros de servicios, o cancelan la publicación de su servicio.

- **Búsqueda**. Los solicitantes de servicios interactúan con uno o más registros de servicios para descubrir un conjunto de servicios con los que pueden interactuar para encontrar una solución.

- **Ligar, Unir**. Los solicitantes de servicios negocian con los proveedores de servicios para acceder e invocar los servicios.

Protocolos de comunicación. Los protocolos de comunicación son una serie de normas que usan los computadores para gestionar sus diálogos en los intercambios de información. Dos equipos diferentes de marcas diferentes se pueden comunicar sin problemas en el caso en que usen el mismo protocolo de comunicaciones.

Se define al protocolo de comunicaciones como un conjunto de reglas y normas que permiten que dos o más entidades de un sistema de comunicación se comuniquen entre ellos para transmitir información por medio de cualquier tipo de

variación de una magnitud física[1]. Los protocolos pueden ser implementados por hardware, software, o una combinación de ambos.

1.5.1.1 SOAP. (Protocolo de Acceso de Objeto Simple) es un protocolo estándar que define cómo dos objetos en diferentes procesos pueden comunicarse por medio de intercambio de datos XML. Este protocolo deriva de un protocolo creado por David Winer en 1998, llamado XML-RPC. SOAP fue creado por Microsoft, IBM y otros, y está actualmente bajo el auspicio de la W3C. Es uno de los protocolos más utilizados en los servicios Web.

Características de SOAP. SOAP ofrece un framework de mensajería básica en la cual los servicios Web se puedan construir. Este protocolo basado en XML consiste de tres partes: un sobre, el cual define qué hay en el mensaje y cómo procesarlo; un conjunto de reglas de codificación para expresar instancias de tipos de datos; y una convención para representar llamadas a procedimientos y respuestas. El protocolo SOAP tiene tres características principales:

- **Extensibilidad**: la seguridad es una extensión aplicada en el desarrollo.
- **Neutralidad**: puede ser utilizado sobre cualquier protocolo de transporte.
- **Independencia**: SOAP permite cualquier modelo de programación.

Mensajes de SOAP. La especificación de SOAP define el formato de los mensajes para comunicar aplicaciones, normalmente dichos mensajes son una forma de comunicación de una única vía entre un emisor y un receptor (mensajes asincrónicos), sin embargo pueden ser combinados de manera de implementar patterns de requerimiento/respuesta (mensajes sincrónicos).

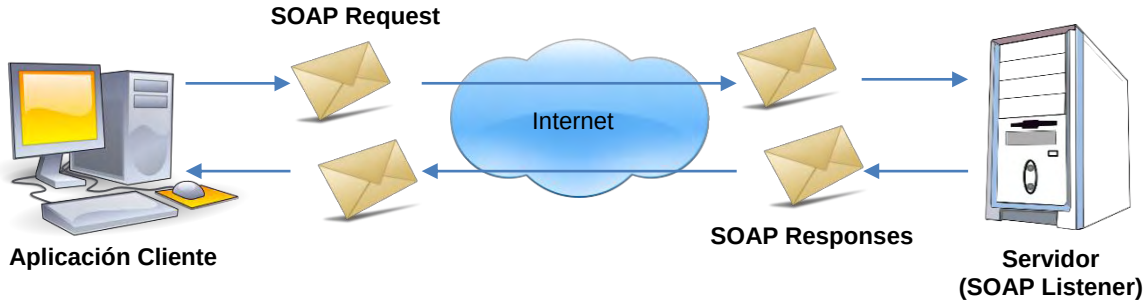
En la versión 1.0 de la especificación de SOAP se establecía como mandatorio el uso de HTTP como protocolo de transporte, sin embargo a partir de la versión 1.1 se desacopla SOAP del uso de HTTP, lo cual permite una mayor variedad de protocolos de transporte. En definitiva la especificación no establece un protocolo de transporte particular pero si específica cómo se realiza el transporte en caso de usar HTTP.

Arquitectura básica de SOAP. Desde el punto de vista de la presente investigación interesa usar SOAP para comunicar aplicaciones, realizando invocaciones RPC e implementando el pattern requerimiento/respuesta, por lo cual el comportamiento es similar al de una arquitectura cliente-servidor, donde el proceso es iniciado por el cliente y el servidor responde al requerimiento del mismo. Este tipo de comunicación se basa en un sistema de mensajes SOAP sincrónicos, codificados en XML que son transportados por HTTP.

[1] RODRÍGUEZ - ARAGÓN, Licesio J., "T-, D- and c- Optimum Designs for BET and GAB Adsorption Isotherms", Chemometrics and Intelligent Laboratory Systems. 2007, Volumen: 89.

En la figura 5, se observa la arquitectura básica de un sistema, análogo al descripto, construido sobre SOAP y los mensajes que definen la interacción entre la aplicación cliente y la aplicación servidor. Generalmente, la aplicación cliente envía un mensaje (Request vía HTTP), el cual al ser recibido por la aplicación servidor genera una respuesta (Response) que es enviada a la aplicación cliente vía HTTP.

Figura 5. Arquitectura básica de un sistema construido sobre SOAP



En definitiva en el entorno de sistemas distribuidos funcionando sobre Internet, las tecnologías existentes presentan serias limitaciones debido a la heterogeneidad del entorno, la presencia de firewalls y al uso de formatos propietarios para representar datos. Bajo estas condiciones SOAP es una alternativa sumamente útil para comunicar aplicaciones heterogéneas (interoperabilidad), fundamentalmente por el uso de XML que es un estándar basado en texto e independiente de plataformas y lenguajes y por el uso de HTTP como transporte, el cual normalmente atraviesa los firewalls dado que estos no bloquean el puerto HTTP. Además al no ser un protocolo sofisticado y al no definir modelos de programación permite que las aplicaciones tengan un bajo acoplamiento, lo cual es ideal en el entorno de Internet.

De todas maneras puede verse que SOAP no es una tecnología que reemplace a las existentes sino una tecnología que permite la interoperabilidad de aplicaciones heterogéneas.

1.5.1.2 REST. (Transferencia de Estado Representacional) es un estilo de arquitectura software especialmente pensada para sistemas distribuidos, como la WWW. En los últimos años se ha popularizado un estilo de arquitectura software conocido como REST. Este nuevo estilo ha supuesto una nueva opción de estilo de uso de los servicios Web. Se basa en que el cliente que accede a una aplicación Web va cambiando su estado en función de los enlaces que se van eligiendo. Este enfoque plantea una aplicación Web como una máquina de estados virtual, dónde las transiciones entre los mismos son hiperenlaces.

Entre otras definiciones para REST, la refieren estrictamente a una colección de principios para el diseño de arquitecturas en red. Estos principios resumen como

los recursos son definidos y divididos. El término frecuentemente es utilizado en el sentido de describir a cualquier interfaz que transmite datos específicos de un domino sobre HTTP sin una capa adicional, como hace SOAP.

Cabe destacar que REST no es un estándar, ya que es tan solo un estilo de arquitectura. Pero aunque REST no es un estándar, está basado en los siguientes estándares:

- HTTP
- URL
- Representación de los recursos: XML/HTML/GIF/JPEG
- Tipos MIME: text/xml, text/html

REST es un estilo de arquitectura para el diseño de aplicaciones en red. La idea es que, en lugar de utilizar los mecanismos complejos, tales como RPC o SOAP para la conexión entre máquinas, se utiliza HTTP para hacer llamadas entre las máquinas. En muchos sentidos, la WWW en sí, basado en HTTP, se puede ver como una arquitectura basada en REST.

Principios de REST

El estilo de arquitectura subyacente a la Web es el modelo REST. Los objetivos de este estilo de arquitectura se listan a continuación:

- **Escalabilidad de la interacción con los componentes**. La Web ha crecido exponencialmente; una prueba de ellos es la variedad de clientes que pueden acceder a través de ella tales como: estaciones de trabajo, sistemas industriales, dispositivos móviles, etc.

- **Generalidad de interfaces**. Gracias al protocolo HTTP, cualquier cliente puede interactuar con cualquier servidor HTTP sin ninguna configuración especial. Esto no es del todo cierto para otras alternativas, como SOAP para los servicios Web.

- **Puesta en funcionamiento independiente**. Este hecho es una realidad que debe tratarse cuando se trabaja en Internet. Los clientes y servidores pueden ser puestas en funcionamiento durante años. Por tanto, los servidores antiguos deben ser capaces de entenderse con clientes actuales y viceversa. Diseñar un protocolo que permita este tipo de características resulta muy complicado. HTTP permite la extensibilidad mediante el uso de las cabeceras, a través de las URIs[*], a través de la habilidad para crear nuevos métodos y tipos de contenido.

[*]URI: identificador uniforme de recursos, que identifica un recurso (servicio, página, documento, dirección de correo electrónico, enciclopedia, etc.)

- **Compatibilidad con componentes intermedios.** Los más populares intermediaros son varios tipos de proxys para la Web. Algunos de ellos, la cache, se utilizan para mejorar el rendimiento. Otros permiten reforzar las políticas de seguridad: firewalls. Y por último, otro tipo importante de intermediarios, Gateway, permiten encapsular sistemas no propiamente Web. Por tanto, la compatibilidad con intermediarios que permite reducir la latencia de interacción, reforzar la seguridad y encapsular otros sistemas.

REST logra satisfacer estos objetivos aplicando las siguientes restricciones:

- **Identificación de recursos y manipulación de ellos a través de representaciones**. Esto se consigue mediante el uso de URIs. HTTP es un protocolo centrado en URIs. Los recursos son los objetos lógicos a los que se le envían mensajes. Los recursos no pueden ser directamente accedidos o modificados, más bien se trabaja con representaciones de ellos. Internamente el estado del recurso puede ser cualquier cosa desde una base de datos relacional a un fichero de texto.

- **Mensajes auto descriptivos**. REST establece que los mensajes HTTP deberían ser tan descriptivos como sea posible. Esto hace posible que los intermediarios interpreten los mensajes y ejecuten servicios en nombre del usuario. Uno de los modos que HTTP logra esto por medio del uso de varios métodos estándares, muchos encabezamientos y un mecanismo de direccionamiento.

- **Hipermedia como un mecanismo del estado de la aplicación**. El estado actual de una aplicación Web debería ser capturada en uno o más documentos de hipertexto, residiendo tanto en el cliente como en el servidor. El servidor conoce sobre el estado de sus recursos, aunque no intenta seguirle la pista a las sesiones individuales de los clientes. Esta es la misión del navegador, saber cómo navegar de recurso a recurso, recogiendo información que el necesita o cambiar el estado que el necesita cambiar.

En la actualidad muchos sitios Web comprometen uno más de estos principios, como por ejemplo, seguir la pista de los usuarios moviéndose a través de un sitio. Esto es posible dentro de la infraestructura de la Web, pero daña la escalabilidad, volviendo un medio sin conexión en todo lo contrario.

1.5.2 Indicadores de uso. Analizar qué pasa dentro de la aplicación y cómo la están utilizando los usuarios es clave para preparar las nuevas versiones y funcionalidades de la misma. Es por ello que hay que ser consciente de que cada aplicación tendrá sus indicadores específicos. Las métricas que todas las aplicaciones deben considerar son las siguientes:

- **Usuarios Activos**. Descargar una aplicación es la parte más fácil, conseguir que el usuario vuelva a entrar ya cuesta un poco más. Es por esta razón se debe considerar tanto los usuarios activos mensuales (MAU) como los diarios (DAU). Conociéndolo todo sobre ellos, el uso que hacen de la aplicación, de dónde son, etc permite segmentarlos y definir acciones personalizadas a cada uno con el objetivo de fidelizarlos. Los usuarios activos son importantes pero también nuestra capacidad de atraer nuevos usuarios. El porcentaje de nuevos usuarios es otra métrica para aplicaciones que no se debe olvidar: cada vez que disminuye este porcentaje debe sonar una alarma.
- **Uso de la aplicación**. Una de las preguntas claves es: ¿en qué pantalla pierdes la mayoría de tus usuarios?, conocer el flujo de navegación dentro de la aplicación es muy importante ya permite saber por qué pantallas han ido los usuarios y, sobre todo, en qué pantallas se han quedado. En un juego este dato puede informar sobre un nivel demasiado difícil. En una aplicación sobre productividad, informará de algún error de planteamiento.
- **Largada de la sesión**. ¿Cuántos minutos tiempo utilizan la aplicación? Los que entran y salen, seguramente se tiene una aplicación que no es lo que buscaban.
- **Retención**. Se puede medir la retención como el porcentaje de usuarios que vuelven a la apliación después de su primera visita. Y no solamente es importante saber la retención de los usuarios sino que también la frecuencia con la que vuelven a ella. Como más comprometidos y fidelizados sean los usuarios, mejores estrategias de monetización se pueden desarrollar.
- **Coste de adquisición del cliente**. Saber cuánto cuesta adquirir un usuario es importante ya que en función de este dato se sabe hasta cuánto se puede invertir en publicidad. Este indicador se calcula sumando todos los gastos empleados en conseguir el nuevo cliente (marketing, comercial, infraestructura) y se divide por el número de clientes conseguidos en este mismo periodo.
- **Ingresos medios por cliente o average revenue per user (ARPU**). Es muy importante que los usuarios utilicen la aplicación, que estén comprometidos y vuelvan a ella. Tan importante como esto es saber rentabilizarlos. Para ello se utiliza el ARPU que se calcula sumando todos los ingresos por usuario (precio de la aplicación, in-app purchases, anuncios, etc) y se divide por el número de usuarios.
- **Vida útil o customer lifetime value (LTV).** Para saber cuál es el valor del usuario. Se tiene que es el resultado de multiplicar el ARPU por la vida media que se espera de un cliente. Teniendo este dato se puede hacer previsiones sobre lanzamientos y necesidades de capital.

Otra cuestión a tener en cuenta es la navegación dentro de cada pantalla para saber si realmente los usuarios están siguiendo el proceso que se había pensado. Para ellos los mapas pueden ayudar.

1.6 SINCRONIZACIÓN

Sincronizar hace referencia a la coordinación de procesos que se ejecutan simultáneamente para completar una tarea, con el fin de obtener un orden de ejecución correcto y evitar así estados inesperados.

La sincronización de los datos se refiere al proceso de propagación de los cambios en los datos y el esquema entre el publicador y los suscriptores después de haber aplicado la instantánea inicial en el suscriptor. La sincronización puede producirse:

- De forma continua, lo que es típico de la replicación transaccional.
- A petición, lo que es típico de la replicación de mezcla.
- Según una programación, lo que es típico de la replicación de instantáneas.

Cuando se sincroniza una suscripción, se producen diferentes procesos según el tipo de replicación que se utilice:

- **Replicación de instantáneas**. La sincronización significa que el Agente de distribución vuelve a aplicar la instantánea en el suscriptor, de modo que los datos y el esquema de la base de datos de suscripciones sean coherentes con la base de datos de publicación. Si se han realizado modificaciones de los datos o del esquema en el publicador, es necesario generar una nueva instantánea para poder propagarlas al suscriptor.
- **Replicación transaccional**. La sincronización significa que el Agente de distribución transfiere las actualizaciones, las inserciones, las eliminaciones y otros cambios de la base de datos de distribución al suscriptor.
- **Replicación de mezcla**. La sincronización significa que el Agente de mezcla carga los cambios del suscriptor en el publicador y, después, descarga los cambios del publicador en el suscriptor. Si hubiera conflictos, se detectan y se resuelven. Los datos convergen y, al final, el publicador y todos los suscriptores acaban por tener los mismos valores. Si se detectan conflictos y se resuelven, el trabajo confirmado por algunos usuarios se modifica para solucionar el conflicto según las directrices definidas.

Las publicaciones de instantáneas actualizan completamente el esquema en el suscriptor cada vez que se produce una sincronización, así que todos los cambios de esquema se aplican en el suscriptor. La replicación transaccional y la replicación de mezcla también admiten los cambios de esquema más comunes. Para obtener más información, consulte hacer cambios de esquema en bases de datos de publicación.

Uno de los puntos fuertes que tiene Android es su gestión a la hora de sincronización de servicios y cuentas. Desde la pestaña de cuentas y sincronización, se puede administrar todos los servicios que se sincronizan automáticamente y así optimizar el consumo de batería y la información que recibimos a través de estos servicios.

1.7 LMS

Un LMS (Learning Management System) es un sistema de gestión de aprendizaje online, que permite administrar, distribuir, monitorear, evaluar y apoyar las diferentes actividades previamente diseñadas y programadas dentro de un proceso de formación completamente virtual, o de formación semi-presencial.

Su conceptualización está orientada a que éstos sean fácilmente accesibles, amigables, intuitivos y flexibles, permitiendo ser utilizados tanto por los administradores, coordinadores y formadores, como por los estudiantes de un determinado curso, en cualquier momento y lugar, mientras se disponga de conexión a Internet. Por otro lado, también potencian de forma destacable la interacción online entre todos los agentes implicados dentro de un proceso de aprendizaje con componente online.

Los LMS son cada vez más utilizados, tanto por empresas que desean proporcionar capacitación para sus empleados, como también por instituciones educativas y centros escolares.

Se puede decir que un sistema de gestión del aprendizaje efectivo, tiene el potencial necesario para optimizar los sistemas de formación de una organización y sus procesos ya que, en gran medida, se puede adaptar a las necesidades de cualquier organización.

Algunas de las herramientas que permiten crear ambientes de aprendizaje efectivos a nivel online, son:

- Sistema de registro.
- Catálogo de cursos.
- Bibliotecas digitales.
- Seguimiento del desempeño de los estudiantes.
- Mecanismos de autoevaluación.
- Estadísticas e información de cursos y estudiantes.
- Apoyo a comunidades de aprendizaje.

1.7.1 Ventajas. Un LMS eficiente y correctamente utilizado presentará una serie de ventajas, como son:

- **Organización**. En los casos en los que se gestionen grandes volúmenes de usuarios, un LMS permite tener bajo control gran parte del trabajo administrativo necesario que se debe llevar a cabo. Un buen sistema permitirá, en cada punto del proceso online, realizar las tareas de organización necesarias, de forma centralizada: gestión de altas y bajas de alumnos, creación de grupos de trabajo, organización de aulas, establecer calendarios y recordatorios para las tareas y los plazos de entrega de cada curso, realizar la recepción de las pruebas de forma online, e incluso, en algunos casos, validar dichas pruebas de forma automática, según el tipo de evaluación estipulada para cada ejercicio a entregar por los alumnos.

- **Control**. Los administradores de un LMS poseen control total sobre el formato de su aula virtual. Algunos sistemas de gestión han llegado a ser altamente personalizables. Ciertos LMS incluso permiten a los estudiantes poder personalizarse sus opciones a la hora de visualizar su entorno de aprendizaje dentro de cada curso.
- **Seguimiento**. Un LMS permite realizar un seguimiento de las acciones realizadas por los diferentes agentes que intervienen en una acción formativa o entorno virtual de aprendizaje. Esto puede ser muy útil en la medición de los resultados de los estudiantes y su evolución. Mediante el seguimiento de su progreso, se pueden detectar las áreas que necesitan ser reforzadas para mejorar. Cuando esta información puede ser fácilmente accesible, el estudiante siente que tiene un mayor control de su aprendizaje y puede inspirarse a seguir mejorando. Los sistemas de seguimiento y presentación de informes en este tipo de sistemas han ido mejorando con el paso de los años.
- **Evaluación continua**. Muchos usuarios puedan ser evaluados antes de comenzar un curso, durante su aprendizaje y tras la finalización de la acción formativa. Esta información que proporciona el LMS también puede ser útil, no sólo para ver el progreso de cada alumno, sino también para evaluar la eficacia de los programas de formación que la empresa u organización educativa ofrece.
- **Flexibilidad**. En la mayoría de LMS los módulos formativos se pueden adaptar u ordenar para satisfacer diferentes necesidades de la organización o entidad que ofrece los cursos. Por otro lado, para el estudiante, dicha flexibilidad le permite poder llevar su propio ritmo en la evolución de su aprendizaje.
- **Efectividad**. Con toda la información del curso al alcance de los estudiantes, un LMS hace que el hecho de aprender pueda resultar más efectivo, a la vez que pautado. Tener acceso a los calendarios y recordatorios fechados es sumamente útil para los estudiantes.
- **Obligaciones legales**. La mayoría de organizaciones están obligadas a cumplir con ciertos requisitos legales y reglamentarios a la hora de llevar a cabo sus formaciones. Un LMS puede ayudar en ello, ya que puede ser usado para rastrear eficazmente los resultados y los tiempos necesarios para los

requisitos que se deben actualizar o presentar a los organismos que lo requieran.

1.7.2 Chamilo LMS. Chamilo LMS, es un software con todo lo necesario para poner en funcionamiento un aula virtual y gestionar un curso de e-learning.

Objetivos fundamentales

- Mejorar la educación a nivel mundial desarrollando un software sencillo y accesible a nivel técnico.
- Garantizar que el software Chamilo LMS permanezca siempre como un producto 100% de código libre.
- Respetar el trabajo de los usuarios de la comunidad Chamilo LMS que contribuyen al crecimiento del proyecto.

En la actualidad Chamilo LMS presenta los siguientes datos:

Figura 6. Datos actuales del LMS Chamilo

Aspectos técnicos

- Modelo de base de datos relacional. Database_schema_1.8.5.
- Hojas de referencia rápida para los nuevos desarrolladores.
- Revisión una vez al año por un cracker de seguridad para mejorar la seguridad contra los ataques a la web.
- Los equipos de desarrollo de Chamilo LMS son muy abiertos acerca de sus procesos, y siempre se puede encontrar información actualizada.
- Código abierto, bajo la licencia GNU/GPLv3, lo que permite desarrollar extensiones o mejoras, y compartir o no con la comunidad.

1.8 INGENIERÍA DE SOFTWARE.

La Ingeniería del Software es una disciplina de la ingeniería que comprende todos los aspectos de la producción de software desde las etapas iniciales de la especificación del sistema, hasta el mantenimiento de éste después de que se utiliza. En esta definición, existen dos frases clave:

1. **Disciplina de la ingeniería**. Los ingenieros hacen que las cosas funcionen. Aplican teorías, métodos y herramientas donde sean convenientes, pero las utilizan de forma selectiva y siempre tratando de descubrir soluciones a los problemas, aun cuando no existan teorías y métodos aplicables para resolverlos. Los ingenieros también saben que deben trabajar con restricciones financieras y organizacionales, por lo que buscan soluciones tomando en cuenta estas restricciones.
2. **Todos los aspectos de producción de software**. La ingeniería del software no sólo comprende los procesos técnicos del desarrollo de software, sino también, con actividades tales como la gestión de proyectos de software y el desarrollo de herramientas, métodos y teorías de apoyo a la producción de software.

En general, los ingenieros de software adoptan un enfoque sistemático y organizado en su trabajo, ya que es la forma más efectiva de producir software de alta calidad. Sin embargo, aunque la ingeniería consiste en seleccionar el método más apropiado para un conjunto de circunstancias, un enfoque más informal y creativo de desarrollo podría ser efectivo en algunas circunstancias. El desarrollo informal es apropiado para el desarrollo de sistemas basados en Web, los cuales requieren una mezcla de técnicas de software y de diseño gráfico.

Proceso de software. Un proceso del software es un conjunto de actividades y resultados asociados que producen un producto de software. Existen cuatro actividades fundamentales de procesos que son comunes para todos los procesos del software. Estas actividades son:

1. Especificación del software donde los clientes e ingenieros definen el software a producir y las restricciones sobre su operación.
2. Desarrollo del software donde el software se diseña y programa.
3. Validación del software donde el software se válida para asegurar que es lo que el cliente requiere.
4. Evolución del software donde el software se modifica para adaptarlo a los cambios requeridos por el cliente y el mercado.

El proceso general de Ingeniería de Software es seguido en la mayoría de los desarrollo de software, incluso esta investigación sigue este proceso que tiene las siguientes fase que guían el proceso de desarrollo del sistema de un manera organizada y estructurada.

1 **Ingeniería de requisitos**: por lo general los clientes piensan que ellos saben lo que el software tiene que hacer, pero lo cierto es que se necesita de un proceso de descubrimiento, refinamiento, modelado y especificación, donde se depuran en detalle los requisitos del sistema y el papel asignado al software, para esta investigación se toman requisitos iniciales y básicos para que den solución a lo propuesto y puedan evolucionar de tal manera que se ajusten a las necesidades de los usuarios finales.

2 **Análisis**: se comprende de forma detallada cual es el problema a resolver, obteniendo así toda la información necesaria y suficiente para afrontar la solución propuesta por el sistema.

3 **Diseño**: ya con la información suficiente del problema a solucionar, se determina la estrategia que está enfocada en el uso de los sistemas orientados a los servicios Web y en la reutilización para así la resolver el problema.

4 **Implementación**: mediante el uso de las diferentes herramientas y el conocimiento de los temas necesarios se procede al desarrollo del sistema el cual da solución a los problemas planteados y se verifica su funcionalidad en diferentes plataformas.

5 **Pruebas**: para garantizar el correcto funcionamiento del sistema se realizan las pruebas bajo el mayor número de situaciones posibles a las que se pueda enfrentar tanto el servicio Web como las aplicaciones que dan resolución a los problemas de transportes.

6 **Documentación**: dentro de esta investigación se tomaron muchos temas que son de gran impacto y evolución en la actualidad, por lo cual es importante documentar cada proceso del desarrollo del sistema, para que además pueda ser modificado y sus funcionalidades también evolucionen y den solución a todos los problemas relacionados con él.

Un proceso de Ingeniería de Software requiere herramientas que apoyen todas las actividades del ciclo de vida de los sistemas. Como respuesta a ello Rational Software Corp. puso a disposición herramientas que están diseñadas para hacer más eficiente la aplicación de metodologías como el RUP. De esa forma se permite la creación de un ambiente de desarrollo óptimo.

Además, junto con UML constituyen la metodología estándar para el análisis, diseño, implementación y documentación de sistemas orientados a objetos.

1.8.1 RUP (Proceso Unificado de Rational). El Proceso Unificado de Rational (RUP) es un ejemplo de un modelo de proceso moderno que proviene del trabajo en el UML y el asociado Proceso Unificado de Desarrollo de Software. RUP fue

utilizado en este trabajo dentro de la metodología para llevar a cabo el desarrollo de CHAMILO MOBILE.

El RUP reconoce que los modelos de procesos genéricos presentan un solo enfoque del proceso. En contraste, el RUP se describe normalmente desde tres perspectivas:

- Una perspectiva dinámica que muestra las fases del modelo sobre el tiempo.
- Una perspectiva estática que muestra las actividades del proceso que se representan.
- Una perspectiva práctica que sugiere buenas prácticas a utilizar durante el proceso.

La mayor parte de las descripciones del RUP intentan combinar las perspectivas estática y dinámica en un único diagrama (Krutchen). Esto hace el proceso más difícil de entender, por lo que aquí se utilizan descripciones separadas de cada una de estas perspectivas.

El RUP es un modelo en fases, que identifica cuatro fases diferentes en el proceso del software. Estas fases están mucho más relacionadas con asuntos de negocio más que técnicos. Las fases en el RUP son:

1. **Inicio**. El objetivo de la fase de inicio es el de establecer un caso de negocio para el sistema. Se deben identificar todas las entidades externas que interactuarán con el sistema y definir estas interacciones. Si esta aportación es de poca importancia, se puede cancelar el proyecto después de esta fase.

2. **Elaboración**. Los objetivos de la fase de elaboración son desarrollar una comprensión del dominio del problema, establecer un marco de trabajo arquitectónico para el sistema, desarrollar el plan del proyecto e identificar los riesgos clave del proyecto. Al terminar esta fase, se debe tener un modelo de los requerimientos del sistema, una descripción arquitectónica y un plan de desarrollo del software.

3. **Construcción**. La fase de construcción fundamentalmente comprende el diseño del sistema, la programación y las pruebas. Durante esta fase se desarrollan e integran las partes del sistema. Al terminar esta fase, debe tener un sistema software operativo y la documentación correspondiente lista para entregarla a los usuarios.

4. **Transición**. La fase final del RUP se ocupa de mover el sistema desde la comunidad de desarrollo a la comunidad del usuario y hacerlo trabajar en un entorno real. Esto se deja de lado en la mayor parte de los modelos de procesos del software ya que es, en realidad, una actividad de alto costo y a

veces problemática. Al terminar esta fase, se debe tener un sistema software documentado que funciona correctamente en su entorno operativo.

El Proceso Unificado de Rational fue escogido en esta investigación porque es una metodología que permite construir software adaptable al contexto y a las necesidades expuestas.

Además, el conjunto de herramientas de Rational está basado en seis principios fundamentales para el desarrollo de software que son: administración de los requerimientos, desarrollar en forma iterativa, modelar visualmente, pruebas continuas, arquitecturas basadas en componentes y control de cambios.

Principios que se adaptan al manejo del trabajo que se lleva a cabo.

1. **Administrar los requerimientos**: el desafío de administrar los requerimientos de una aplicación móvil es que son dinámicos, es decir, pueden cambiar durante la vida de un proyecto. Por tanto identificar los verdaderos requerimientos de un sistema debe hacerse de tal manera que logre satisfacer tanto objetivos económicos como técnicos en un proceso continuo.

2. **Desarrollar software iterativamente**: un enfoque iterativo facilita la implementación de nuevos cambios tácticos de requerimientos y características del cronograma. Con este enfoque se fortalece la identificación temprana de los riesgos del proyecto.

3. **Modelar software visualmente**: hacer modelos es importante, ya que permite visualizar, especificar, construir y documentar la estructura y comportamiento de la arquitectura de un sistema, si además se utiliza un lenguaje de modelamiento estándar se mejora la capacidad del equipo de trabajo para administrar la complejidad del software.

4. **Pruebas continuas**: la revisión continua permite encontrar las fallas antes de la puesta en producción de un sistema. Si además se desarrolla el software iterativamente, se está revisando la versión en cada iteración, logrando así un proceso de evaluación continuo y cuantitativo.

5. **Utilizar arquitecturas basadas en componentes**: el desarrollo basado en componentes permite la reutilización o adaptación de componentes existentes y disponibles.

6. **Control de cambios**: la coordinación de las actividades y los productos que se van generando implica administrar los cambios al software, este control permite una mejor asignación de los recursos.

1.9 HERRAMIENTAS TECNOLÓGICAS PARA LA CONSTRUCCIÓN DE CHAMILOMOBILE

La construcción de CHAMILOMOBILE se hizo posible gracias a la investigación de nuevas tecnologías. Su desarrollo se implementa en dispositivos móviles con sistema operativo Android, y se considera un sistema escalable, seguro, confiable y estable.

A continuación se da una descripción general de las tecnologías usadas para el desarrollo de CHAMILOMOBILE y que hacen que sea un sistema de gran apoyo para el aprendizaje desde Chamilo LMS Web, ya que cuenta con diferentes opciones sincronizadas y es muy fácil de manejar.

1.9.1 Middleware. La construcción de un middleware tiene como objetivo servir como puente entre la base de datos de la plataforma de Chamilo LMS Web con CHAMILOMOBILE y fue desarrollado bajo PHP.

Un Middleware es en esencia un conjunto de abstracciones de programación que facilitan el desarrollo de sistemas distribuidos complejos. Para comprender una plataforma middleware sólo se precisa conocer su modelo de programación, a partir del modelo de programación se puede determinar cuál serán las limitaciones del middleware. El modelo de programación subyacente determinará cómo evolucionará la plataforma cuando las nuevas tecnologías evolucionen.

Middleware es parte integral en la tecnología moderna para manejo de información basada en XML, SOAP, servicios web y arquitectura orientada a servicios.

Los programas de middleware proporcionan servicios de mensajería de modo que diferentes aplicaciones pueden comunicarse. Por ejemplo, hay una serie de productos de middleware que vinculan a un sistema de base de datos a un servidor Web. Esto permite a los usuarios solicitar información almacenada en la base de datos mediante formularios residentes en un navegador Web, con los datos recibidos el servidor Web puede devolver páginas web dinámicas basadas en las peticiones y perfiles de los usuarios.

Características

- Oculta la complejidad de las aplicaciones distribuidas.
- Oculta la heterogeneidad de hardware, sistemas operativos y protocolos.
- Proporciona interfaces uniformes y de alto nivel utilizados para hacer aplicaciones interoperables y reutilizables y portátiles.
- Proporciona un conjunto de servicios comunes que reduce al mínimo la duplicación de esfuerzos y mejora la colaboración entre aplicaciones.

1.9.2 Estándares de representación de datos. Los estándares son de vital importancia en las comunicaciones. Establecer un lenguaje común permite que múltiples sistemas desarrollados independientemente por distintos fabricantes puedan interoperar.

Obviamente, no es bueno que existan muchas normas para un mismo propósito, ya que la multiplicidad de estándares lleva a la incompatibilidad de los sistemas, a la fragmentación del mercado y a una mayor complejidad técnica en equipos con soporte multiestándar. En el caso de CHAMILOMOBILE entre los estándares de representación de datos utilizados están JSON.

1.9.2.1 JSON. Es un formato alternativo de envío y recepción de datos, es decir remplaza a XML o el envío de texto plano. Este formato de datos es más liviano que XML. Hace el código más sencillo ya que utiliza el código JavaScript como modelo de datos.

Una de las ventajas de JSON sobre XML como formato de intercambio de datos en este contexto es que es mucho más sencillo escribir un analizador sintáctico (parser) de JSON.

Si bien es frecuente ver JSON posicionado contra XML, también es frecuente el uso de JSON y XML en la misma aplicación. Por ejemplo, una aplicación de cliente que integra datos de Google Maps con datos meteorológicos en SOAP hacen necesario soportar ambos formatos.

JSON se basa en dos estructuras:

1. Una colección de pares nombre/valor. En varios idiomas esto se realiza como un objeto, registro, estructura, diccionario, o una matriz asociativa.
2. Una lista ordenada de valores. En la mayoría de los idiomas, esto se realiza en forma de matriz, vector, lista o secuencia.

Se trata de estructuras de datos universales. Prácticamente todos los lenguajes de programación modernos las acogen de una forma u otra. Tiene sentido que un formato de datos que es intercambiable con los lenguajes de programación también se basa en estas estructuras.

Para concluir, JSON es empleado con frecuencia en entornos Web donde el flujo de los datos es alto entre el cliente y el servidor, y donde los tiempos de procesamiento de los datos son de vital importancia. JSON optimiza los tiempos de respuesta debido a su leve peso. Y aunque es frecuente ver a JSON enfrentado contra XML, no deben ser excluyentes ya que incluso pueden ser complementarios el uso de JSON y XML en la misma aplicación.

1.9.2.2 XML. Es un Lenguaje de Etiquetado Extensible, muy simple, pero estricto que juega un papel fundamental en el intercambio de una gran variedad de datos. Es un lenguaje muy similar a HTML pero su función principal es describir datos y no mostrarlos como es el caso de HTML. XML es un formato que permite la lectura de datos a través de diferentes aplicaciones.

En CHAMILOMOBILE se utiliza para la presentación de clases y vistas de la interfaz gráfica. Las tecnologías XML son un conjunto de módulos que ofrecen servicios útiles a las demandas más frecuentes por parte de los usuarios. XML sirve para estructurar, almacenar e intercambiar información. Entre las tecnologías XML disponibles se pueden destacar:

1. **XSL**: Lenguaje Extensible de Hojas de Estilo, cuyo objetivo principal es mostrar cómo debería estar estructurado el contenido, cómo debería ser diseñado el contenido de origen y cómo debería ser paginado en un medio de presentación como puede ser una ventana de un navegador Web o un dispositivo móvil.

2. **XPath**: Lenguaje de Rutas XML, es un lenguaje para acceder a partes de un documento XML.

3. **XLink**: Lenguaje de Enlace XML, es un lenguaje que permite insertar elementos en documentos XML para crear enlaces entre recursos XML.

4. **XPointer**: Lenguaje de Direccionamiento XML, que permite el acceso a la estructura interna de un documento XML, a elementos, atributos y contenido.

5. **XQL**: Lenguaje de Consulta XML, es un lenguaje que facilita la extracción de datos desde documentos XML. Ofrece la posibilidad de realizar consultas flexibles para extraer datos de documentos XML en la Web.

1.9.3 Ingeniería de software basada en componentes – ISBC. El desarrollo de software basado en componentes permite reutilizar piezas de código pre elaborado que permiten realizar diversas tareas, conllevando a diversos beneficios como las mejoras a la calidad, la reducción del ciclo de desarrollo y el mayor retorno sobre la inversión.

La ISBC es un acercamiento basado en la reutilización para definir, implementar, y componer, componentes débilmente acoplados en sistemas.

Los fundamentos de la ingeniería del software basada en componentes son:

1. **Componentes independientes**, que son completamente especificados por sus interfaces. Debería haber una clara separación entre la interfaz de los

componentes y su implementación para que una implementación de un componente pueda reemplazar por otro sin cambiar el sistema.

2. **Estándares de componentes**, que facilitan la integración de los componentes. Estos estándares se incluyen en un modelo de componentes y definen, en el nivel más bajo cómo las interfaces de componentes deberían especificarse y cómo se comunican los componentes. Si los componentes cumplen con los estándares, entonces su funcionamiento es independiente de su lenguaje de programación.
3. **El middleware**, que proporciona soportes software para la integración de componentes. Para conseguir que componentes independientes y distribuidos trabajen juntos, necesita un soporte middleware que maneje las comunicaciones de los componente. Un producto middleware utilizado para implementar un modelo de componentes puede proporcionar soporte para asignación de recursos, gestión de transacciones, seguridad y concurrencia.

4. **Un proceso de desarrollo**, que se adapta a la ingeniería del software basada en componentes. Si se intenta añadir una aproximación basada en componentes a un proceso de desarrollo que está adaptado a la producción de software original.

El desarrollo basado en componentes se está adoptando cada vez más como una aproximación fundamental a la ingeniería del software incluso aunque los componentes reutilizables no estén disponibles. La ISBC está asentada sobre unos principios de diseño sólidos que soportan la construcción de software comprensible y mantenible. Los componentes son independientes para que no interfieran en su funcionamiento unos con otros. Los detalles de implementación se ocultan, por lo que la implementación de los componentes puede cambiarse sin afectar al resto del sistema.

El paradigma de ensamblar componentes y escribir código para hacer que estos componentes funcionen se conoce como Desarrollo de Software Basado en Componentes.

1.9.3.1 Ventajas de la ISBC. El uso de este paradigma posee algunas ventajas:

- **Reutilización del software**, el desarrollo de software basado en componentes es un acercamiento basado en la reutilización para definir, implementar, y componer, componentes débilmente acoplados en los sistemas.

- **Simplifica las pruebas**, las pruebas son ejecutadas probando cada uno de los componentes antes de probar el conjunto completo de componentes.

- **Simplifica el mantenimiento del sistema**, el desarrollador es libre de actualizar o agregar componentes según sea necesario sin afectar otras partes

del sistema, esto es posible cuando existe un débil acoplamiento entre componentes.

- **Mayor calidad**: la calidad de una aplicación basada en componentes mejora con el paso del tiempo, dado que un componente puede ser construido y mejorado continuamente.

1.9.4 Lenguajes de programación

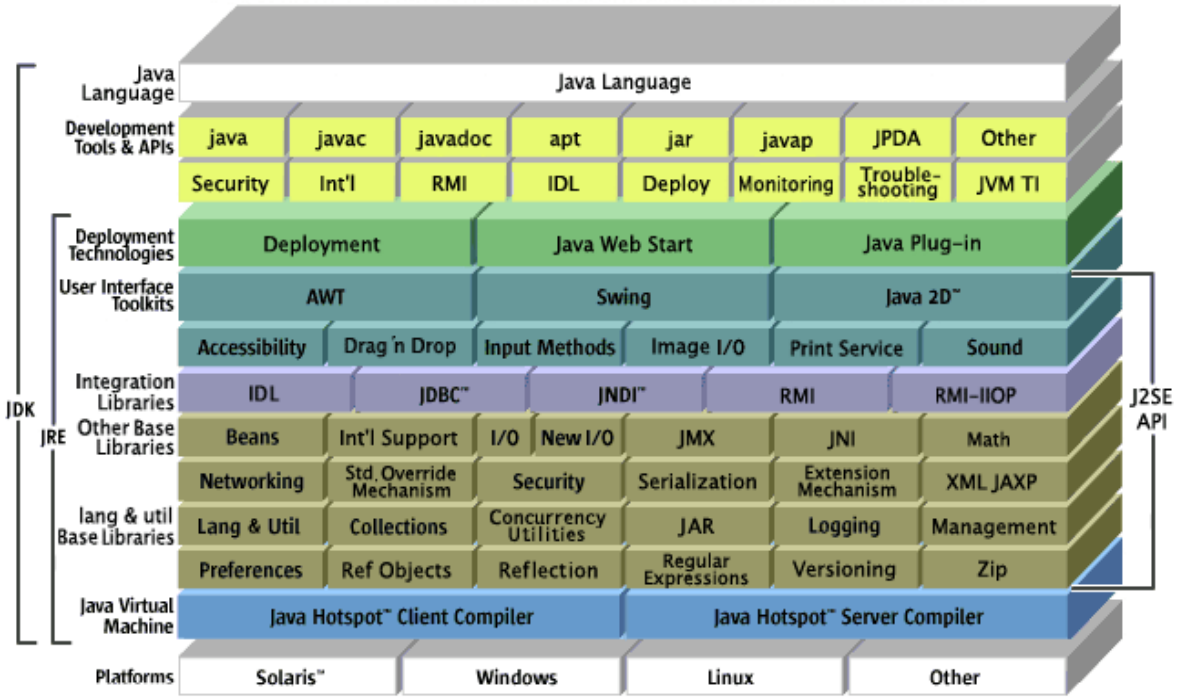
En CHAMILOMOBILE para el desarrollo de las capas lógica y de datos se trabajó con JAVA.

1.9.4.1 JAVA. Es un lenguaje de programación donde su sintaxis deriva mucho de C y C++, pero tiene menos facilidades de bajo nivel que cualquiera de ellos.

La creación de este lenguaje y plataforma se inspiró en las funcionalidades interesantes propuestas por otros lenguajes tales como C++, Eiffel, SmallTalk, Objective C, Cedar/Mesa, Ada Perl. El resultado es una plataforma y un lenguaje idóneo para el desarrollo de aplicaciones seguras, distribuidas y portables en numerosos periféricos y sistemas transportables interconectadas en red pero también en Internet (clientes ligeros) y en estaciones de trabajo (clientes pesados).

La plataforma Java. La mayoría de las plataformas actuales son la combinación de una máquina y de un sistema operativo (ej.: PC + Windows). La plataforma Java se distingue por el hecho de que solo se compone de una parte de software que se ejecuta en numerosas plataformas físicas y diferentes sistemas operativos. La siguiente figura, sobre el lenguaje Java, muestra los diferentes componentes que conforman a la plataforma.

Figura 7. Plataforma Java



Como muestra la figura 7, se compone de los elementos siguientes:

- La máquina virtual. Java (JVM)
- La interfaz de programación de aplicación Java (API Java), repartida en tres categorías (APIs básicas, APIs de acceso a los daros y de integración con lo existente, APIs de gestión de la interfaz de las aplicaciones con el usuario).
- Las herramientas de despliegue de las aplicaciones.
- Las herramientas de ayuda al desarrollo.

En conclusión Java es una plataforma, que permitir ejecutar programas sin tener relativamente en cuenta el hardware final, sin volver a reescribir todo el código del programa y consiste en tres grandes bloques, el lenguaje Java, una máquina virtual y un programa de aplicación de interfaz o API.
Para el despliegue de la aplicación CHAMILOMOBILE se utilizó Android Studio.

1.9.4.2 PHP. Usa una mezcla entre interpretación y compilación para intentar ofrecer a los programadores la mejor mezcla entre rendimiento y flexibilidad.

PHP compila para el código propio en una serie de instrucciones siempre que estas son accedidas. Estas instrucciones son entonces ejecutadas una por una hasta que el script termina. PHP es recompilado cada vez que se solicita un script.

Una ventaja importante de interpretar el código es que toda la memoria usada por tu código es manejada por PHP, y el lenguaje automáticamente vacía esta memoria cuando el script finaliza. Esto significa que no hay que preocuparse de las conexiones a la base de datos, porque PHP lo hace por uno.

Es un lenguaje multiplataforma; los programas funcionan igual sobre diferentes plataformas, trabajando sobre la mayoría de servidores Web.

1.9.5 Tecnologías para prototipos de prueba. Para los prototipos de prueba de CHAMILOMOBILE, se utilizaron diferentes tecnologías que permitieron que el desarrollo de los prototipos sea óptimo. .

A continuación, se describe dichas tecnologías dentro de las siguientes secciones:

- Sistemas Operativos
- Entornos de Desarrollo Integrados (IDE)
- Máquinas Virtuales
- Servidores Web

1.9.5.1 Sistemas operativos

Todos los sistemas operativos poseen ciertas características fundamentales en común: deben administrar la memoria, la capacidad de procesamiento, los dispositivos y periféricos, archivos y redes.

A continuación se procede a especificar los sistemas operativos utilizados en los prototipos de prueba.

- **Windows Server**. Es una marca que abarca una línea de productos servidor de Microsoft Corporation, consiste en un sistema operativo diseñado para servidores de Microsoft y una gama de productos dirigidos al mercado más amplio de negocios. Windows Server ofrece más control sobre la infraestructura de servidores y red, mejor hosting, protección del sistema operativo y el entorno de red, herramientas administrativas intuitivas, facilidad de consolidación, virtualización de servidores y aplicaciones.

Se describe como “listo para el centro de datos”[2], así mismo como “el movimiento hacia una mayor modularidad, automatización más fuerte y una mejor virtualización tienen mucho sentido en un mundo de nubes públicas y privadas”, pero remarcó que “dicho esto, la capacidad de Windows para suministrar errores

[2] BISSON, Simon. En: ZDNet. <http://WWW.zdnet.com/> [22 de julio de 2013].

oscuros y el tiempo que consumen esos errores no han cambiado", y concluye que "no obstante, se trata de una fuerte mejora en general".

- **Linux.** El sistema operativo Linux, nació en 1991, gracias a un estudiante de la universidad de Helsinki[3]. El éxito de Linux se basa en una idea ingeniosa de su creador, L. Torvalds: en inscribir su proyecto bajo términos de la licencia GPL y proponer a todos los programadores y otros hackers de Internet que le ayudaran.

El impulso de Linux proviene en gran parte del hueco que lleno en términos de núcleo en el proyecto GNU[**]. En conclusión Linux es un sistema libre completo.

- **Android.** Es un sistema operativo basado en Linux diseñado principalmente para dispositivos móviles con pantalla táctil, como teléfonos inteligentes o Tablet.

El lanzamiento de Android como nueva plataforma para el desarrollo de aplicaciones móviles ha causado grandes expectativas y está teniendo una importante aceptación tanto por parte de los usuarios como de la industria.

Las aplicaciones se desarrollan habitualmente en el lenguaje Java con Android Software Development Kit (Android SDK)[***]. Pero están disponibles otras herramientas de desarrollo, incluyendo un Kit de Desarrollo Nativo para aplicaciones o extensiones en C o C++, Google App Inventor y un entorno visual para programadores novatos. El desarrollo de aplicaciones para Android no requiere aprender lenguajes complejos de programación. Todo lo que se necesita es un conocimiento aceptable de Java y estar en posesión del kit de desarrollo de software o SDK.

1.9.5.2 Entornos de desarrollo integrado (IDE). Un entorno de desarrollo integrado, es un entorno de programación que ha sido empaquetado como un programa de aplicación, es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica (GUI).

Los IDE proveen un marco de trabajo amigable para la mayoría de los lenguajes de programación tales como C++, PHP, Java, C#, Visual Basic, Python, Delphi, etc. En algunos lenguajes, un IDE puede funcionar como un sistema en tiempo de ejecución, en donde se permite utilizar el lenguaje de programación en forma interactiva, sin necesidad de trabajo orientado a archivos de texto. Los entornos integrados desarrollo (IDE) utilizados en esta investigación son los siguientes.

[3] Ibid. p. 499.
[**] GNU: es un proyecto que consiste en reescribir completamente un sistema operativo libre.
[***] Android SDK: es el proceso por el cual se crean nuevas aplicaciones para el sistema operativo Android.

Android Studio. Es un entorno de desarrollo integrado (IDE), y utiliza una licencia de software libre Apache 2.0, está programado en Java y es multiplataforma.

Se desarrolló con el objetivo de crear un entorno dedicado en exclusiva a la programación de aplicaciones para dispositivos Android, proporcionando a Google un mayor control sobre el proceso de producción.

Características

- Soporte para programar aplicaciones para Android Wear (sistema operativo para dispositivos corporales como por ejemplo un reloj).
- Herramientas Lint, detecta código no compatible entre arquitecturas diferentes o código confuso que no es capaz de controlar el compilador, para detectar problemas de rendimiento, usabilidad y compatibilidad de versiones.
- Utiliza ProGuard para optimizar y reducir el código del proyecto al exportar a APK, muy útil para dispositivos de gama baja con limitaciones de memoria interna.
- Integración de la herramienta Gradle encargada de gestionar y automatizar la construcción de proyectos, como pueden ser las tareas de testing, compilación o empaquetado.
- Nuevo diseño del editor con soporte para la edición de temas.
- Nueva interfaz específica para el desarrollo en Android.
- Permite la importación de proyectos realizados en el entorno Eclipse.
- Posibilita el control de versiones accediendo a un repositorio.
- Alertas en tiempo real de errores sintácticos, compatibilidad o rendimiento antes de compilar la aplicación.
- Vista previa en diferentes dispositivos y resoluciones.
- Integración con Google Cloud Platform, para el acceso a los diferentes servicios que proporciona Google en la nube.
- Editor de diseño que muestra una vista previa de los cambios realizados directamente en el archivo xml.

Dreamweaver. Es el programa incluido en el suite de Adobe, destinado a la creación y la gestión de sitios Web.

La gran ventaja de este editor es su gran poder de ampliación y personalización del mismo, puesto que en este programa, sus rutinas (como la de insertar un hipervínculo, una imagen o añadir un comportamiento) están hechas en JavaScript-C, lo que le ofrece una gran flexibilidad en estas materias. Esto hace que los archivos del programa no sean instrucciones de C++ sino rutinas de JavaScript[*] que hace que sea un programa muy fluido, que todo ello hace, que

[*]JavaScript: es un lenguaje de programación interpretado. Se define como orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico.

programadores y editores Web hagan extensiones para su programa y lo ponga a su gusto.

Ventajas de los IDEs.

- Es más ágil y óptimo para usuarios que no son expertos en manejo de consola.
- Formateo de código.
- Funciones para renombrar variables, funciones.
- Warnings y errores de sintaxis en pantalla de algo que no va a funcionar al interpretar o compilar.
- Poder crear proyectos para poder visualizar los archivos de manera gráfica.
- Herramientas de refactoring como por ejemplo seria extraer una
- porción de código a un método nuevo.
- No es recomendado pero posee un navegador Web interno por si se quiere probar las cosas dentro de la IDE.

De acuerdo con todo esto cabe resaltar que algunos IDEs no son gratuitos por el mismo motivo que el software son utilizados para trabajos mejorados, por tal motivo en los IDEs se puede implementar líneas de código donde se pueda resolver algún problema con base al compilador, este es el que permitirá modificar o corregir errores del programa.

1.9.5.3 Máquinas virtuales. Una máquina virtual es un software que simula a una computadora y puede ejecutar programas como si fuese una computadora real. Para las pruebas pertinentes del prototipo de aplicación de CHAMILOMOBILE, se tomó la siguiente máquina virtual.

- **Dalvik.** Es la máquina virtual que utiliza la plataforma para dispositivos móviles Android. Ejecuta archivos en el formato Dalvik Executable (*.dex), un formato optimizado para el almacenamiento eficiente y ejecución mapeable en memoria. Su objetivo fundamental es el mismo que cualquier máquina virtual, permite que el código sea compilado a un bytecode independiente de la máquina en la que se va a ejecutar, y la máquina virtual interpreta este bytecode a la hora de ejecutar el programa.

Dalvik está basada en registros y puede ejecutar clases compiladas por un compilador Java y que posteriormente han sido convertidas al formato nativo usando la herramienta “dx”. El hecho de que corra sobre un kernel Linux le permite delegar las tareas relacionadas con la gestión de hilos y memoria a bajo nivel. El uso de Dalvik para los prototipos de aplicación del proyecto, permite reducir bastante el tamaño del programa buscando información duplicada en las diversas clases y reutilizándola.

1.9.5.4 Servidores Web. La principal función de un servidor Web es almacenar los archivos de un sitio y emitirlos por Internet para poder ser visitado por los usuarios. Básicamente, un servidor Web es una gran computadora que guarda y transmite datos vía Internet. Cuando un usuario entra en una página de Internet su navegador se comunica con el servidor enviando y recibiendo datos que determinan qué es lo que ve en la pantalla. Por eso se dice que los servidores Web están para almacenar y transmitir datos de un sitio según lo que pida el navegador de un visitante.

Es importante contar con un servidor estable para alojar las aplicaciones, para lo cual en esta investigación se contó con uno que permitió alojar el prototipo de aplicación móvil.

Apache. Es el servidor de páginas Web, que permite acceder a páginas Web alojadas en un ordenador. Es el más utilizado seguido de Microsoft Information Services. En este proyecto se emplea el servidor Apache, por múltiples razones como disponibilidad, facilidad de instalación, pocos recursos necesarios, precio, disponibilidad del código fuente.

Ventajas: Modular, Código abierto, Multi-plataforma, Extensible y Popular.

Además, Apache permite configurar un Hosting Virtual basado en IPs o en nombres, es decir, tener varios sitios Web en un mismo equipo (por ejemplo: nombreWeb1.com, nombreWeb2.com,….).

Características de Apache

- Soporte para los lenguajes perl, python, tcl y PHP.
- Módulos de autenticación: mod_access, mod_auth y mod_digest.
- Soporte para SSL y TLS.
- Permite la configuración de mensajes de errores personalizados y negociación de contenido.
- Permite autenticación de base de datos basada en SGBD.

La arquitectura característica de un servidor que funciona con Apache es modular. Esto quiere decir que está compuesto por partes o módulos que se utiliza de acuerdo con las necesidades que se presentan.

2. CHAMILOMOBILE: APLICACIÓN MÓVIL ANDROID PARA LA INTERACCIÓN Y OBTENCIÓN DE INDICADORES DE SEGUIMIENTO, SINCRONIZADA CON EL LMS CHAMILO DE LA FACULTAD DE INGENIERÍA DE LA UNIVERSIDAD DE NARIÑO

CHAMILOMOBILE, nace de la necesidad de desarrollar un aplicativo móvil para incrementar la accesibilidad y disponibilidad de la plataforma Chamilo LMS Web, ya que se notó que dicha plataforma es usada actualmente en la Universidad de Nariño. Si bien, los estudiantes cuentan con el acceso a la plataforma vía web, en muchos casos es más probable que un individuo cargue un dispositivo móvil que un computador, de esta manera CHAMILOMOBILE propone.

2.1 DESARROLLO CHAMILOMOBILE

CHAMILOMOBILE es un aplicación 100% nativa que utiliza las últimas tecnologías que combina estabilidad, integridad y robustez, la cual consta de un entorno amigable y de fácil acceso.

Tabla 1. Especificación de módulos de CHAMILOMOBILE

Módulo	Submódulos	Objetivo Principal
Mis cursos	• Lista de cursos • Descripción del curso • Documentos • Enlaces • Anuncios • Evaluaciones • Glosarios • Informes • Usuarios • Tareas • Documentos compartidos • Notas • Ejercicios	Obtener la información básica para un curso.
Mensajes	• Lista de mensajes • Envió de mensajes	Mantener una comunicación con los usuarios de la plataforma.
Agenda	• Calendario	Visualización de eventos.
Mis amigos	• Información de amigos • Interacción por mensajes	Conocer la información de las personas que son tus amigos e interactuar con ellos.
Indicadores	• Tabla de indicadores	Poder observar el impacto de la plataforma de manera sustentable.

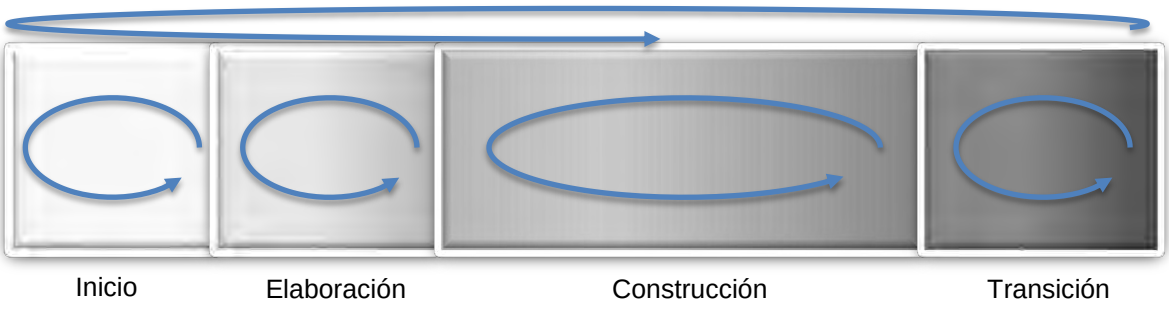
En la tabla 1, la columna submódulos hace referencia a las funcionalidades que están inmersas dentro de cada módulo.

Debido a que RUP es una metodología iterativa incremental por fase y actividades se presenta la última versión, y de ella únicamente los artefactos software más relevantes. En principio se muestra el plan iterativo de CHAMILOMOBILE, las entidades que interactúan con CHAMILOMOBILE, las interacciones de los actores con el sistema, la arquitectura del sistema, el comportamiento del sistema y algunas especificaciones de implementación.

2.2 PLAN ITERATIVO RUP SEGUIDO PARA CHAMILOMOBILE

CHAMILOMOBILE se desarrolla en dos iteraciones de proceso*

Figura 8. Iteraciones de fase por proceso



2.3 ENTIDADES QUE INTERACTÚAN CON CHAMILOMOBILE

Los usuarios comunes son los estudiantes y docentes, quienes interactúan con Chamilo LMS Web y por lo tanto podrán acceder a CHAMILOMOBILE para ampliar su experiencia de uso.

A continuación se presentan un detalle más específico de los actores que interactúan con SISNOVA.

2.3.1 Actor Estudiante. Para los módulos de

* Una iteración de proceso se refiere a la culminación de un objetivo propuesto para un sistema, en donde se ha seguido un respectivo orden de fase. Si aparece nuevos requerimientos al final de la fase del objetivo i se plantea un objetivo i+1 y se repite el proceso por fase. En la figura 8, la flecha circular de mayor tamaño encima de todas las fases, representa las iteraciones de proceso.

En la tabla 2, se muestra la descripción del actor estudiante, siendo este el actor principal del sistema.

Tabla 2. Descripción actor estudiante

Nombre:	Estudiante
Descripción	
Es el actor principal, debe contar con conocimientos sobre la estructura básica de Chamilo LMS Web. Este actor además de los conocimientos genéricos enunciados en el anterior párrafo, debe tener inscrito un curso. Para el uso de los módulos este actor está limitado a la habilitación por parte del docente	
Objetivos	
– Utilizar los módulos desarrollados. – Mantener contacto directo con Chamilo LMS Web.	

En la tabla siguiente, se describe al actor docente.

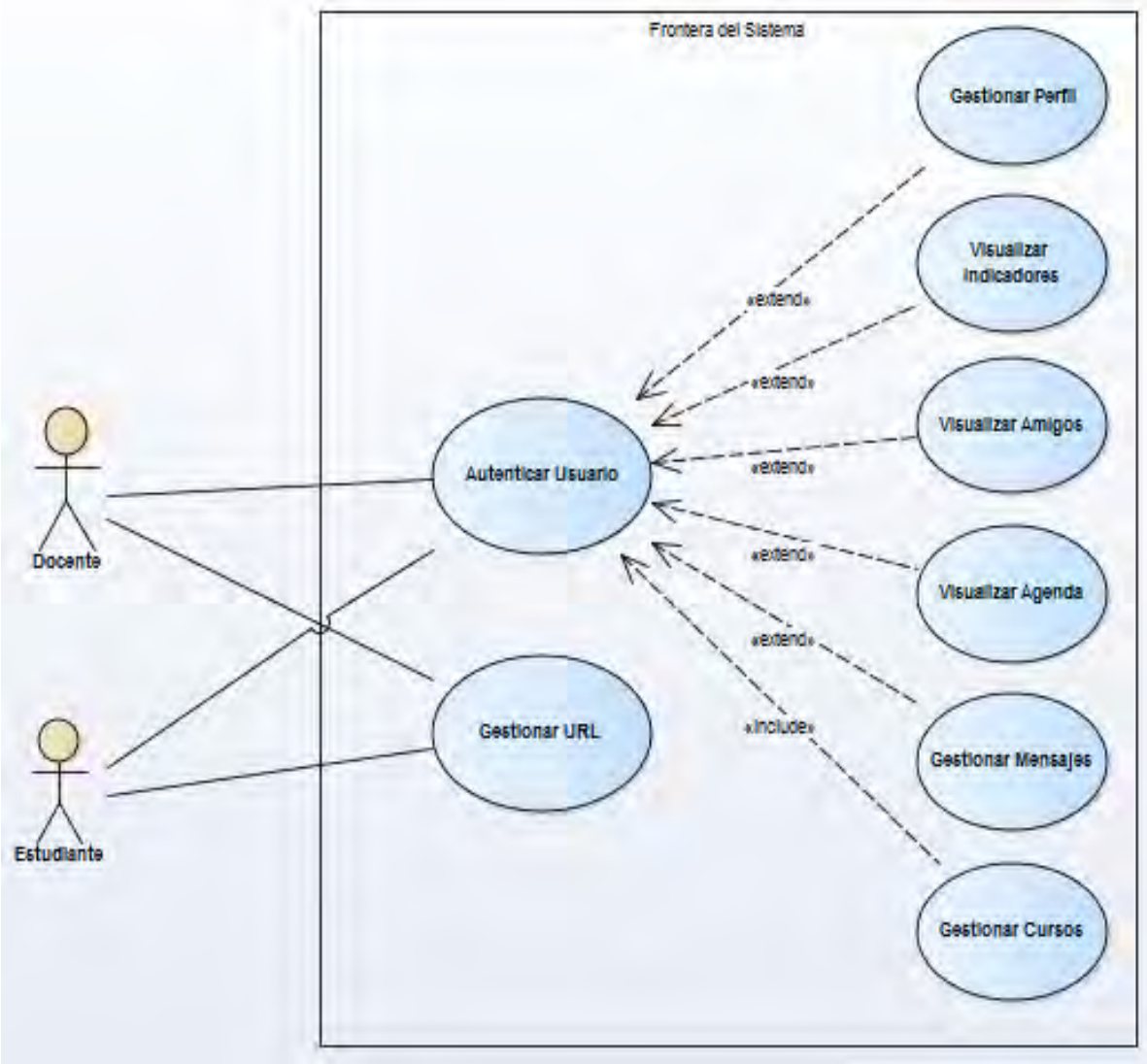
Tabla 3. Descripción actor docente

Nombre:	Docente
Descripción	
Este actor es quien crea las descripciones de los cursos.	
Objetivos	
– Descripción de los cursos. – Cargue de documentos – Mantener contacto directo con Chamilo LMS Web.	

2.4 INTERACCIONES DE LOS ACTORES CON EL SISTEMA

2.4.1 Diagrama de casos de uso general

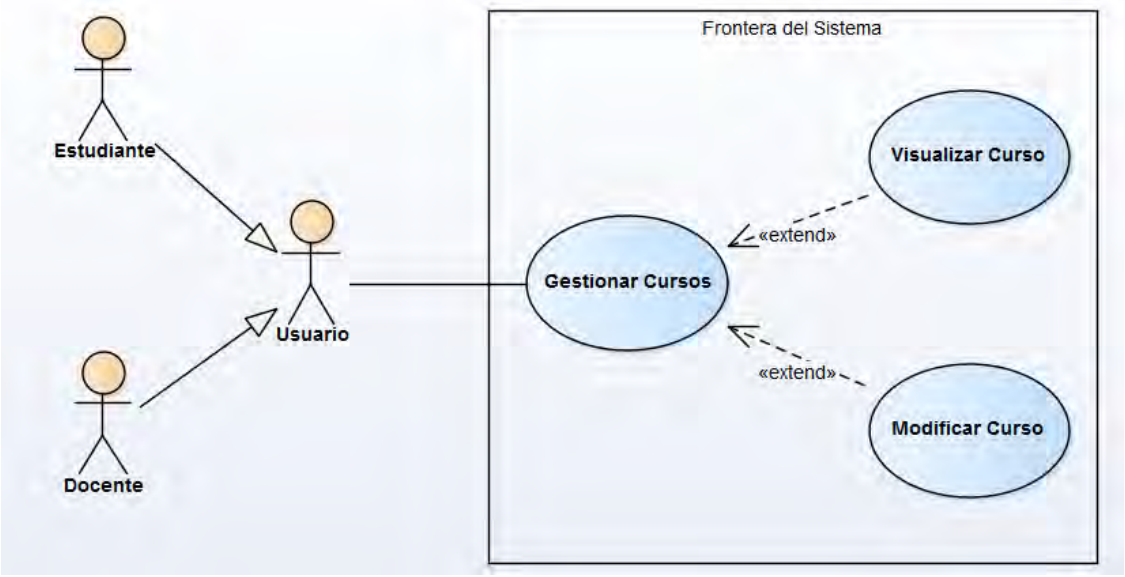
Figura 9. Diagrama de casos de uso general



En el diagrama anterior, se aprecia las interacciones que los actores Docente y Estudiante tienen con el sistema, limitado en el recuadro de borde azul, se puede observar que los dos actores pueden autenticarse como usuarios permitiendo visualizar y gestiona los módulos desarrollados.
El caso de uso "Gestionar URL" los actores pueden gestionarlo sin necesidad de autenticarse.

2.4.2 Diagrama de casos de uso gestionar cursos

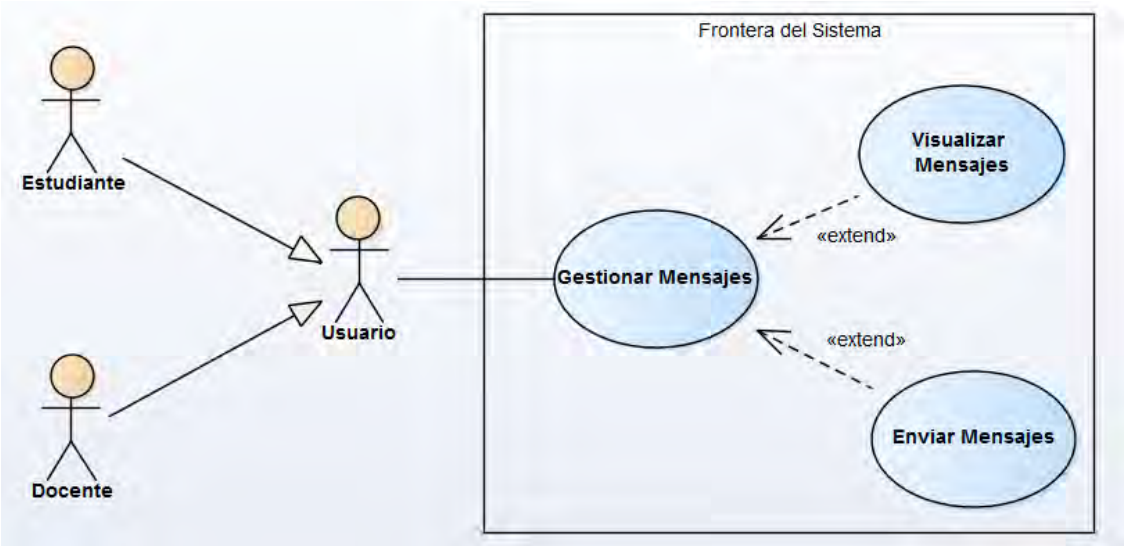
Figura 10. Diagrama de caso de uso de gestionar cursos



En el figura 10, los actores Docente y Estudiante interactúan con el usuario

2.4.3 Diagrama de casos de uso gestionar mensajes

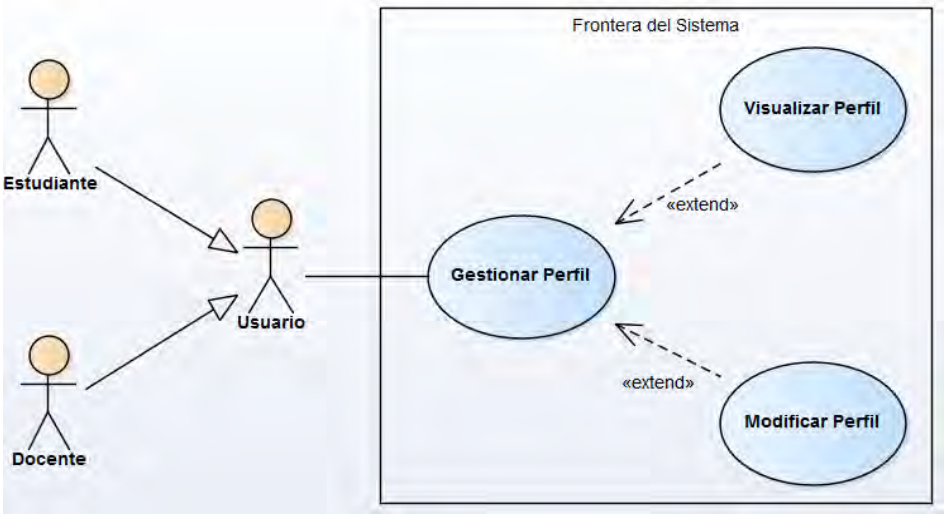
Figura 11. Diagrama de casos de uso gestionar mensajes



En la Figura 11, los actores del sistema pueden visualizar los mensajes intercambiados con otros usuarios, y además enviar mensajes a los mismos.

2.4.4 Diagrama de casos de uso gestionar perfil

Figura 12. Diagrama de casos de uso gestionar perfil

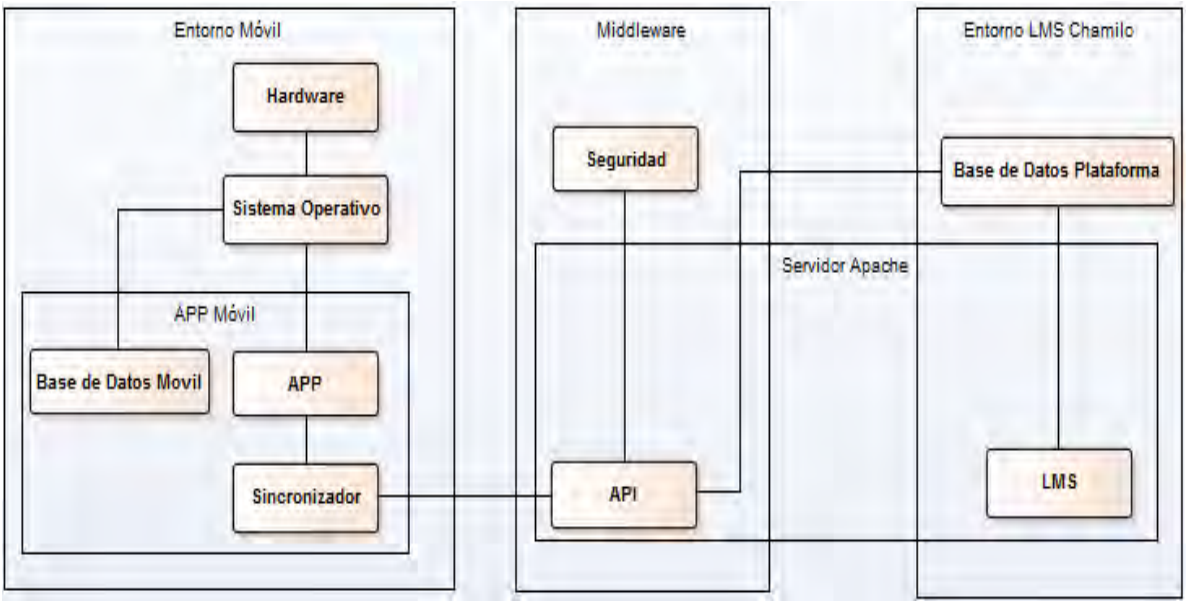


En la figura 12, se observa la interactividad de los actores con el sistema, para visualizar y modificar datos básicos de su perfil.

2.5 ARQUITECTURA DEL SISTEMA

2.5.1 Diagrama de bloques

Figura 13. Diagrama de despliegue del sistema

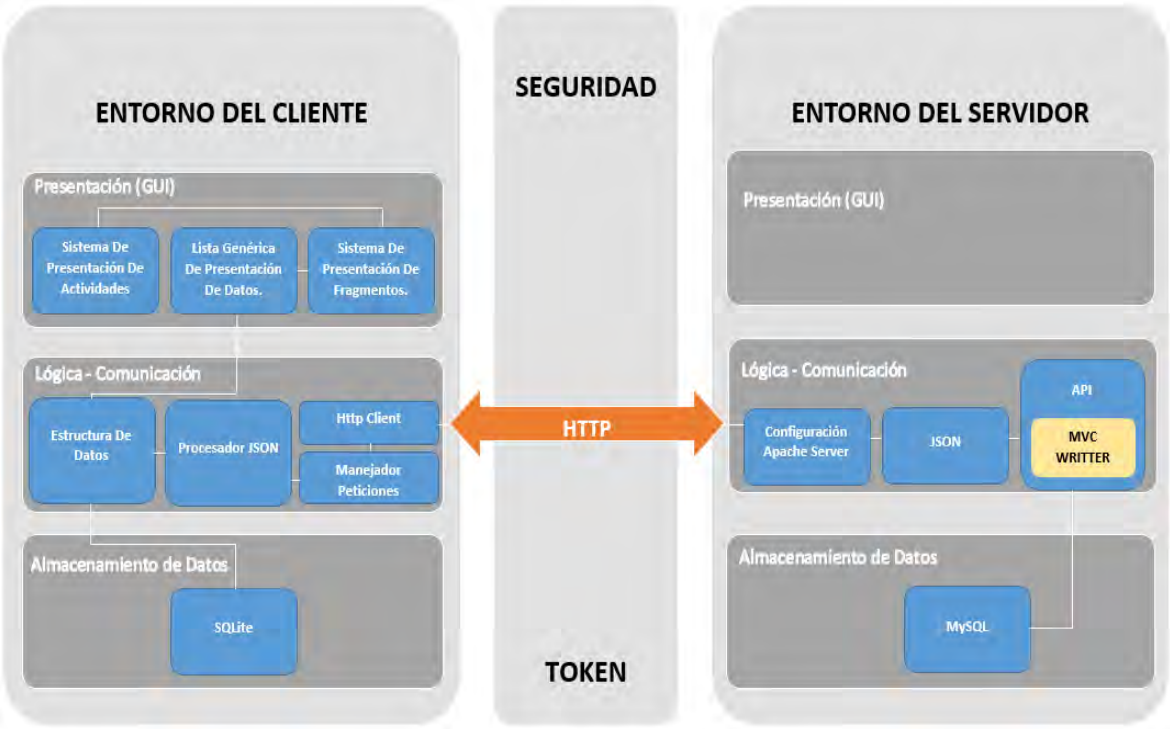


En la Figura 13 se observan tres elementos los cuales son: el entorno móvil, que está constituido por el hardware donde funciona el Sistema operativo, dando soporte a las APPs y almacenando las bases de datos de las mismas.

La APP CHAMILOMOBILE cuenta con un componente sincronizador que se comunica con el middleware. El middleware está conformado por el API funcionando en el servidor web Apache implementando un componente de seguridad, dicha API se comunica con el entorno del LMS Chamilo LMS interactuando de manera directa con la base de datos del mismo el cual esta soportado en el servidor Web Apache al igual que el API.

2.5.2 Diagrama de arquitectura del sistema

Figura 14. Diagrama de bloques del sistema



En la figura 14, se ve la arquitectura usada para el desarrollo de CHAMILOMOBILE, la línea continua de color blanco indica una interacción directa entre los elementos del sistema.

En el entorno del cliente se observa un diseño de tres capas donde la capa de presentación contiene tres elementos generales que son: el sistema de presentación de actividades que permite mostrar al usuario la información con

marcos constantes por ejemplo: menús, títulos, etc. Este elemento interactúa directamente con el sistema de presentación de fragmentos que muestran al usuario fina información en una ubicación que después puede ser ocupada por otro fragmento, estos a la ves interactúan con las listas de presentación de datos que sirven para mostrar de forma amigable y ordenada la información al usuario final.

Las listas de datos interactúan con la capa lógica y de comunicación, de tal forma que ocupan estructuras de datos que son las que contienen la información a mostrar, y dichas estructuras de datos son cargadas gracias a un elemento de tipo JSON procesados, y almacenados en una base de datos residente en el dispositivo móvil (SQLite). Los objetos de tipo JSON requieren de un manejador de peticiones el cual se soporta en la estructura del HTTP Client para establecer comunicación por medio del protocolo HTTP.

La información viaja desde el entorno móvil hasta el entorno del servidor a través del método POST. Además, existe un token que eleva el nivel de seguridad en las peticiones que lo requieran, puesto que el token es validado por el API, que interpreta la petición realizada por el cliente del entorno móvil, aplicando las operaciones necesarias directamente sobre el motor de base de datos del LMS Chamilo, y retornando el resultado de dichas operaciones como objetos JSON.

2.5.3 Diagrama de clases. Los diagramas siguientes muestran en un principio una vista general donde se pueden observar las clases más relevantes de cada componente y las relaciones entre ellas.

En la figura 16 que se muestra más adelante está representado el mapa de entidades que intervienen en la capa lógica y en la capa de datos, además se muestra la clase que maneja la itneraccion con el motor SQLite, junto con las clases encargadas de la comunicación con el API y la interfaz que permite que los elementos de interfaz gráfica puedan implementar dicha comunicación.

Figura 15. Diagrama de clases de manejadores de interfaz

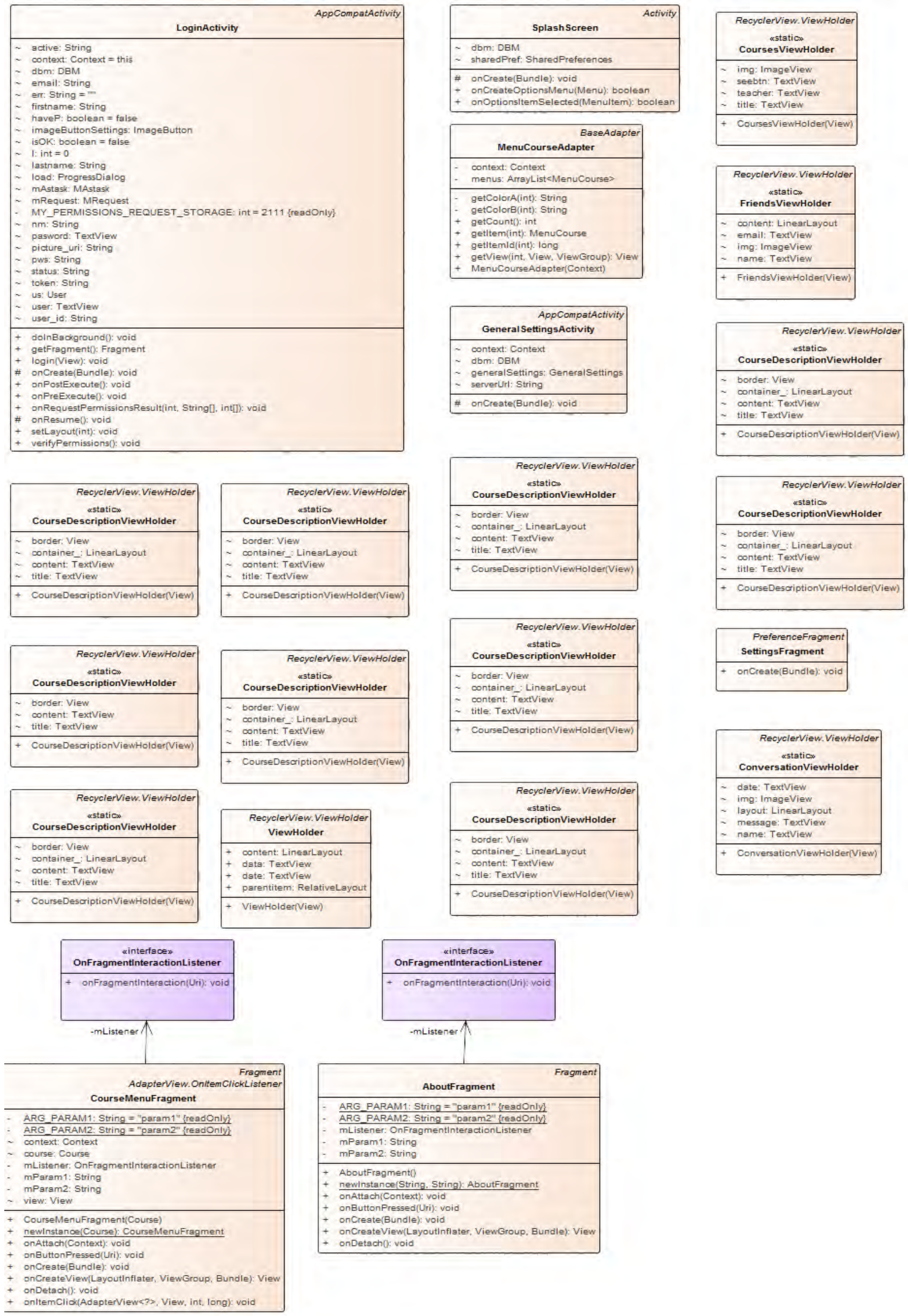


Figura 16. Diagrama de clases de representación de entidades

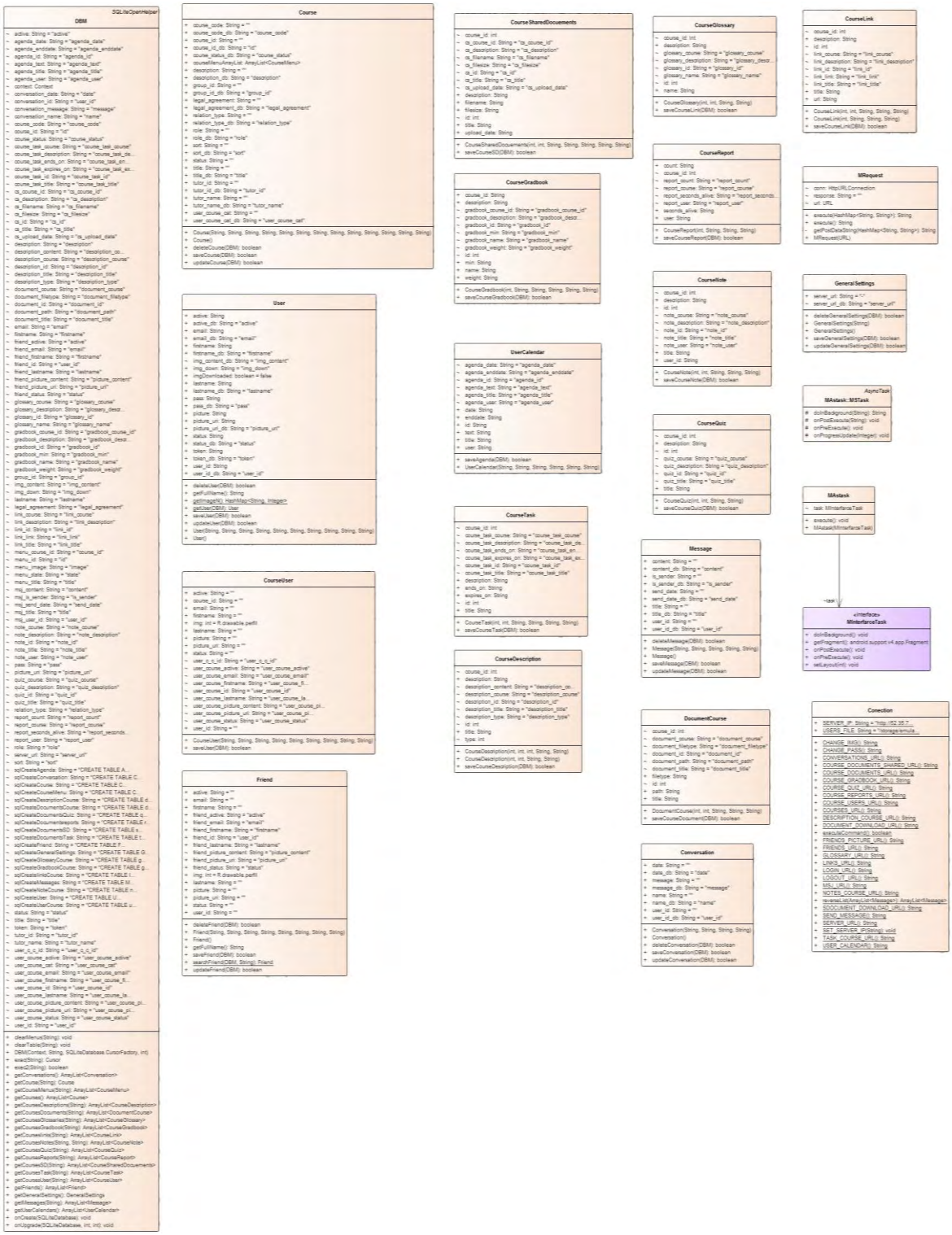


Figura 17. Diagrama de clases para la representación gráfica de datos para el usuario final

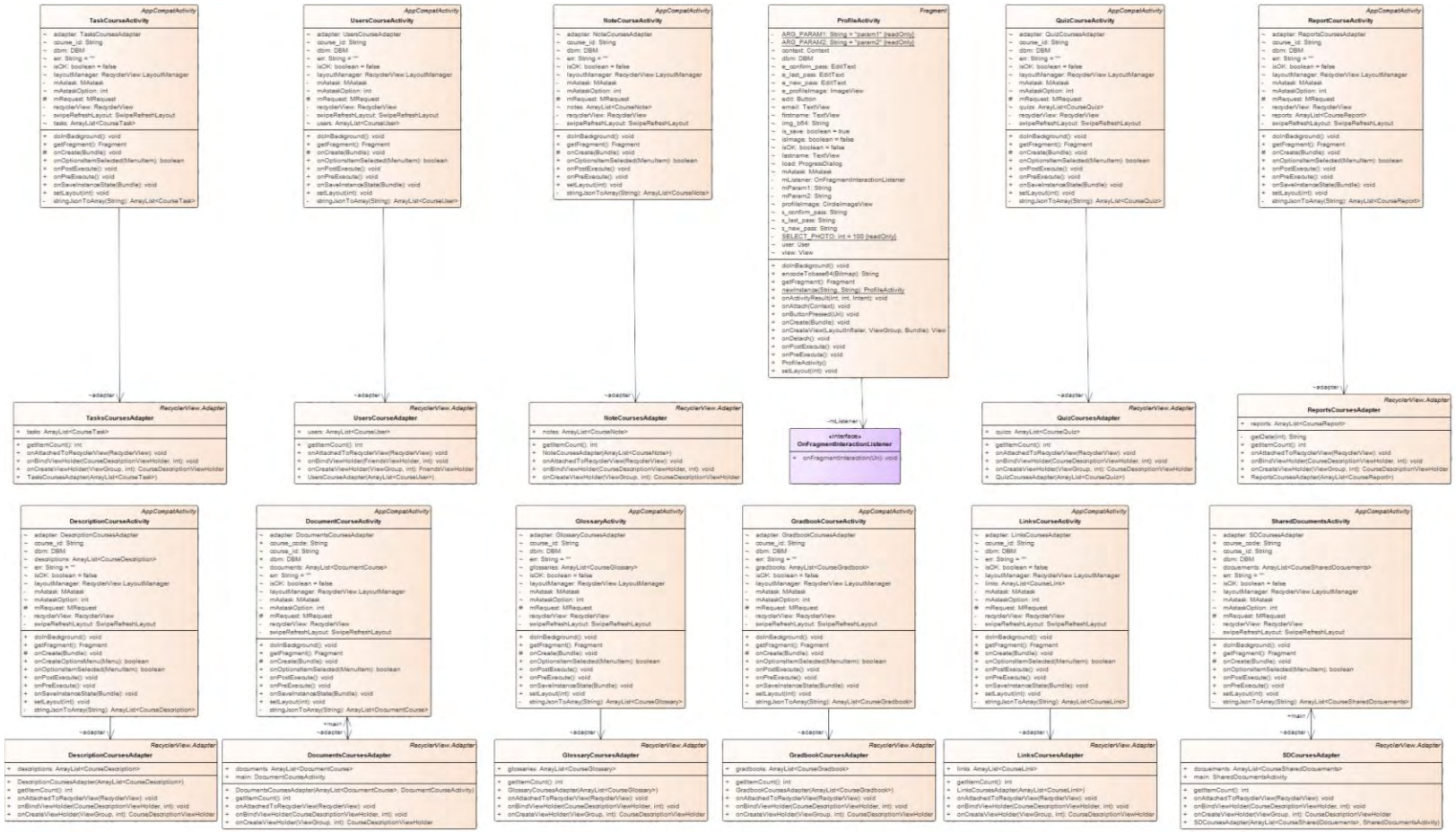


Figura 18. Diagrama de clases relacional para la representación gráfica de datos para el usuario final

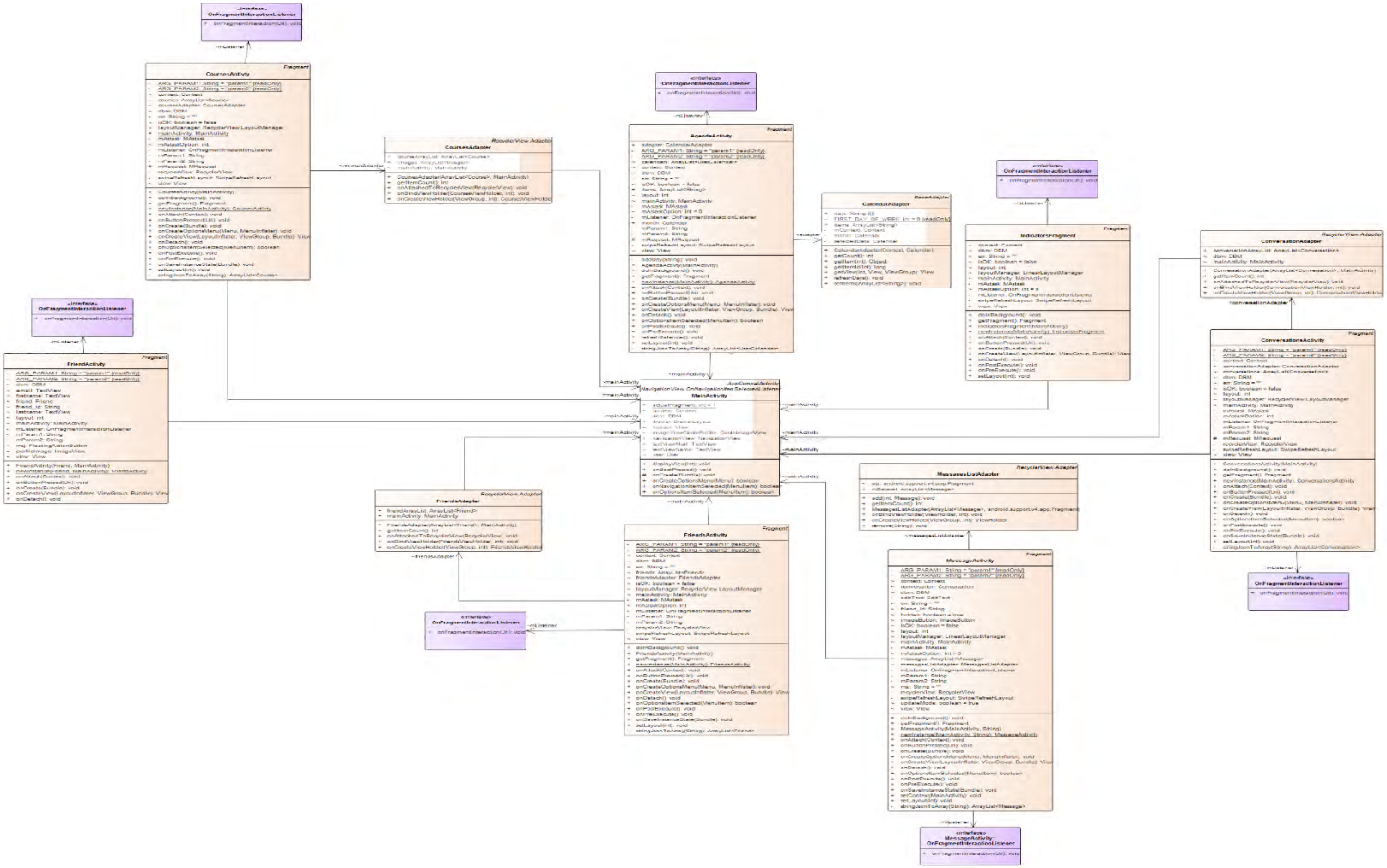
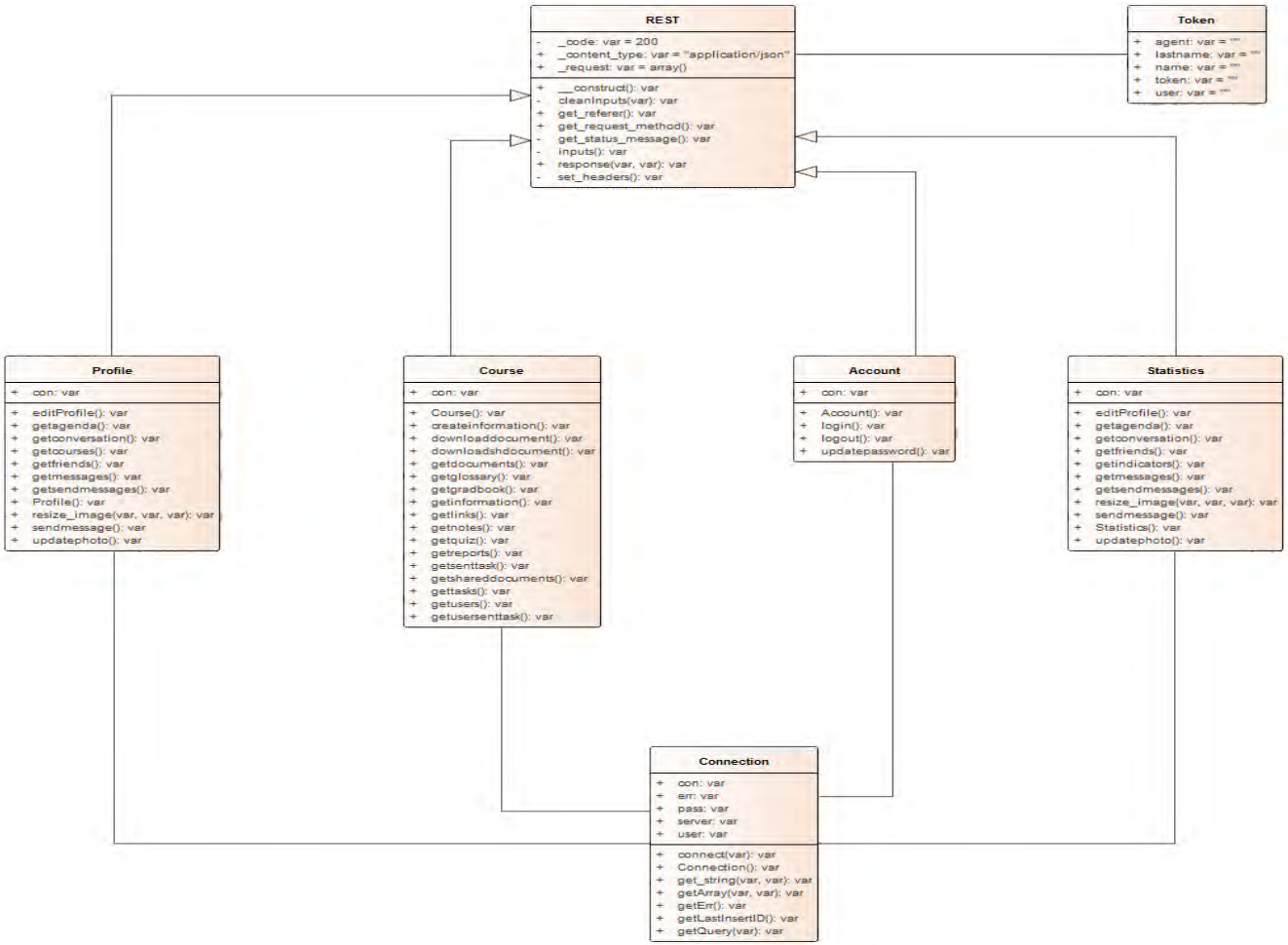


Figura 19. Diagrama de clases del API (Middleware)

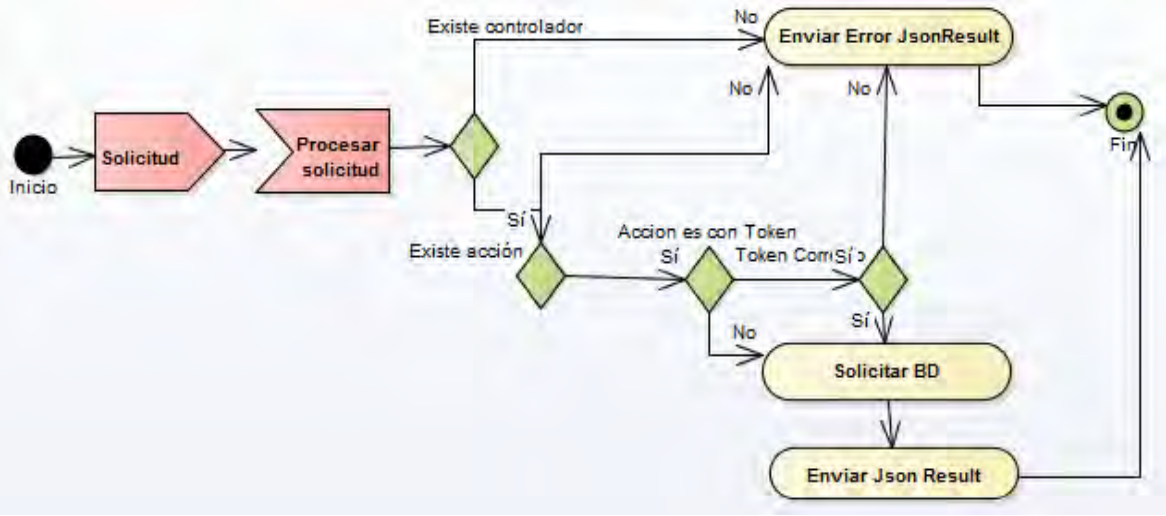


2.6 COMPORTAMIENTO DEL SISTEMA

A continuación, se presentan los diagramas de actividades y de secuencia para CHAMILOMOBILE. En el presente proyecto los diagramas de actividades se han orientado al orden de visión general sobre las actividades de los actores principales de cada módulo y los diagramas de secuencia se han orientado a dar una visión de detalle sobre la secuencia de ejecución de funcionalidades de manera que se dé una idea de los pasos que sigue cada funcionalidad y la colaboración entre objetos de las clases presentadas en la sección arquitectura del sistema del presente documento.

2.6.1 Diagramas de actividades

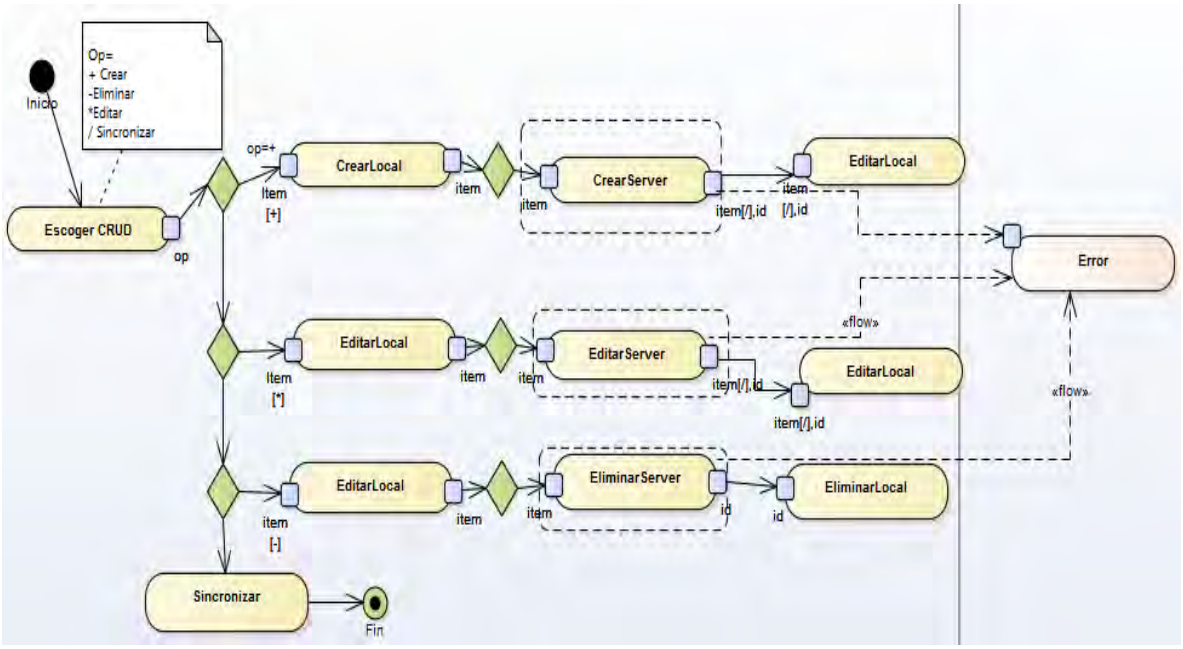
Figura 20. Diagrama de actividades de procesamiento de peticiones



En la figura 20, se muestra la forma en que es procesada una petición en el API.

Una vez la solicitud llega al servidor, el API verifica que el controlador solicitado exista, si el controlador no existe el API responde con un JsonResult notificando el error, si el controlador existe, se verifica que la acción solicitada sobre dicho controlador exista, si la acción no existe el API responde con un JsonResult notificando el error, si la acción existe, se verifica si dicha acción requiere token de seguridad, si la acción no requiere token de seguridad se procede a hacer la solicitud a la base de datos y retornar el resultado de la consulta con JsonResult, si la acción requiere token entonces se verifica la valides del token, si este no es válido el API responde con un JsonResult notificando el error, si el token es válido se hace la consulta a la base de datos y se retorna el resultado como JsonResult.

Figura 21. Diagrama de actividades de cambios de estados en sincronización



En la figura 21, se muestra la forma como se sincroniza la información entre las bases de datos local (móvil) y servidor (LMS Chamilo).

En este proceso inicialmente se debe establecer cuál es la acción a ejecutar, simbolizadas así: crear (+), Eliminar (-), Editar (*) y sincronizar (/).

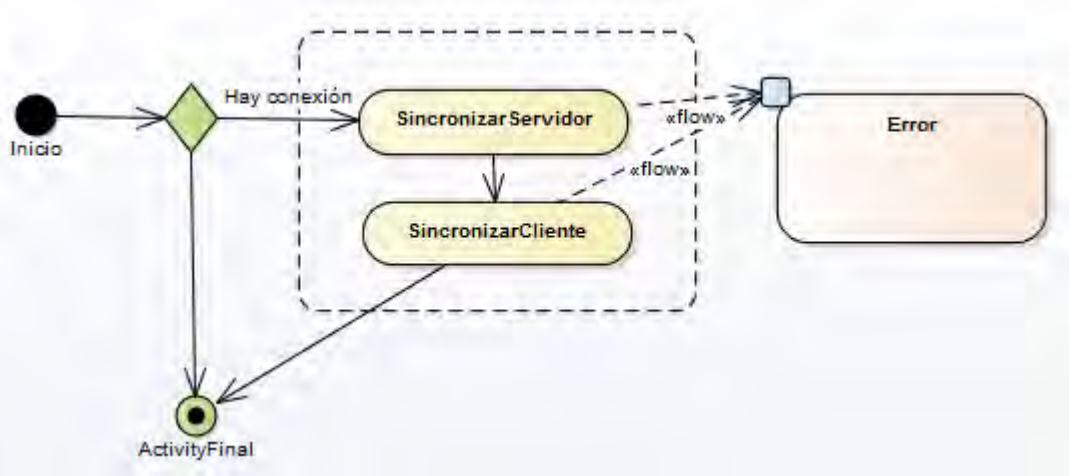
Si la opción es de creación (+), se crea el registro de manera local, luego se crea el registro en el servidor en donde pueden desprenderse excepciones, si no ocurre alguna excepción se realiza una edición sobre el registro creado localmente para modificar su estado y evidenciar que dicho registro ya está sincronizado.

Si la opción es de edición (*), se modifica el registro de manera local, luego se modifica el registro en el servidor en donde pueden desprenderse excepciones, si no ocurre alguna excepción se realiza una edición sobre el registro editado localmente para modificar su estado y evidenciar que dicho registro ya está sincronizado.

Si la opción es de eliminación (-), se modifica el registro de manera local, luego se elimina el registro en el servidor en donde pueden desprenderse excepciones, si no ocurre alguna excepción se realiza la eliminación local del registro.

Si la opción es de sincronización (/) se ejecuta el tal proceso descrito en el diagrama de la figura 23.

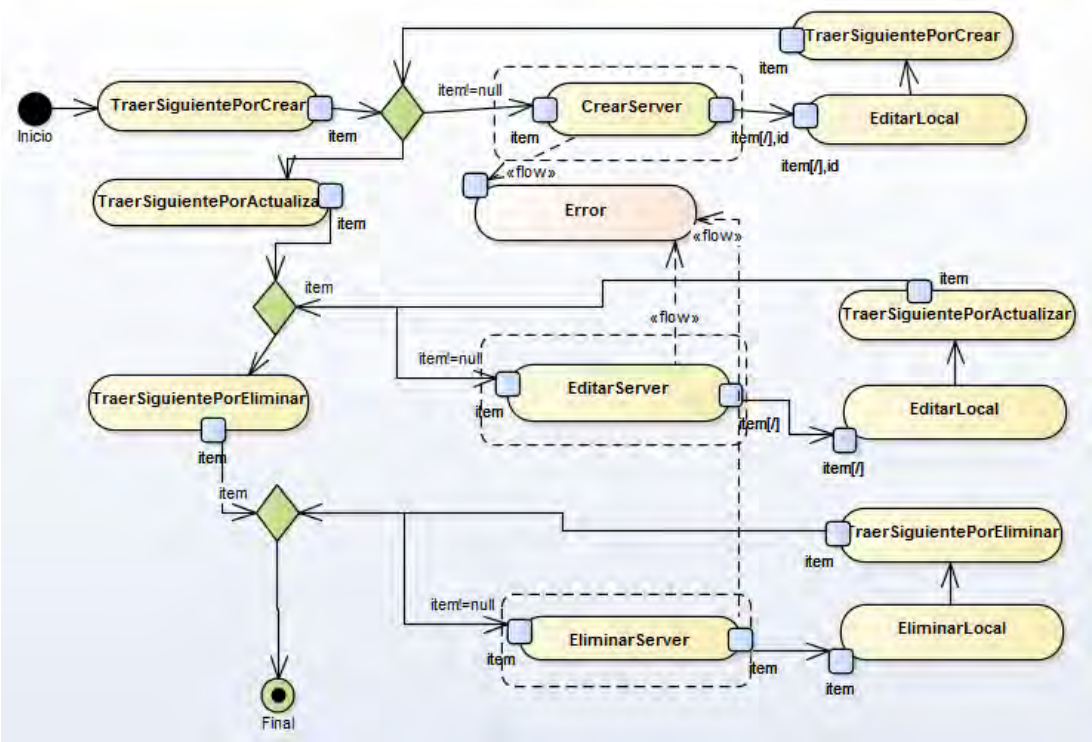
Figura 22. Diagrama de actividades de sincronización



En la figura 22, se muestra como se realiza la sincronización de los aplicativos.

Se verifica el estado de la conexión entre el cliente y el servidor, si hay conexión se ejecuta el proceso de sincronización del servidor y posteriormente el proceso de sincronizar el cliente, en ambos procesos se pueden presentar excepciones.

Figura 23. Diagrama de actividad de sincronización de conjunto de datos



En la figura 23, se muestra como se realizan las operaciones CRUD en el proceso de sincronización.

Si el tipo de operación es de creación, se trae el registro a crear, se valida que el elemento a insertar no sea nulo, si el elemento no es nulo entonces se realiza la creación en el servidor proceso en el cual pueden presentarse excepciones, si no se presentan excepciones s realiza la edición local para evidenciar que el elemento fue sincronizado, y se repite el proceso con el siguiente elemento a crear hasta que no hayan más elementos por crear.

Si el tipo de operación es de actualización, se trae el registro por actualizar, se valida que el elemento por actualizar no sea nulo, si el elemento no es nulo entonces se realiza la edición en el servidor, proceso en el cual se pueden presentar excepciones, si no se presentan excepciones se realiza la edición local para evidenciar que el elemento fue sincronizado, y se repite el proceso con el siguiente elemento por actualizar hasta que no hayan más elementos por actualizar.

Si el tipo de operación es de eliminación, se trae el registro por eliminar, se valida que el elemento por eliminar no sea nulo, si el elemento no es nulo entonces se realiza la eliminación del elemento en el servidor, proceso en el cual se pueden presentar excepciones, si no se presentan excepciones se realiza la eliminación local, y se repite el proceso con el siguiente elemento por eliminar hasta que no hayan más elementos por eliminar.

2.7 ESPECIFICACIONES DE IMPLEMENTACIÓN DE CHAMILOMOBILE

Para el uso del aplicativo móvil se debe seguir los siguientes pasos:

1. Descargar la aplicación desde el Play Store.
2. Configuración de la URL a la cual se desea conectar.
3. Ingreso y uso de los módulos desarrollados.

3. PROTOTIPO DE APLICACIÓN PARA EL DESARROLLO DE CHAMILOMOBILE

En esta sección, se expone le prototipo de aplicación utilizado para las pruebas de CHAMILOMOBILE. A continuación, se muestra las especificaciones que se tuvieron en cuenta para este prototipo.

Tabla 4. Especificaciones de plataforma de desarrollo y ejecución para el prototipo.

Prototipo de Aplicación	Hardware	Sistema Operativo	IDE	Lenguaje	Máquina Virtual	Servidor Web
Móvil	Celular Tablet	Android	Android Studio	Java	Dalvik y ART	Apache

Tabla 5. Especificaciones adicionales de ejecución del servicio en el prototipo

Prototipo de Aplicación	Acceso al servicio	Tipo de llamadas a funcionalidades
Móvil	REST	Llamadas asíncronas

La aplicación móvil Android se desarrolla en lenguaje JAVA, para probar el funcionamiento bajo peticiones de recursos REST asíncronas.

3.1 FUNCIONAMIENTO DE CHAMILOMOBILE EN EL PROTOTIPO DE APLICACIÓN

En esta sección se presenta el prototipo de aplicación móvil de prueba construida para ejecutarse en dispositivos con sistema operativo Android y codificado mediante el lenguaje Java.

La prueba de este prototipo se realizó en un dispositivo móvil y en una Tablet, que previamente tenían conexión a Internet, así entonces en estos entornos se demostró la funcionalidad de la aplicación.

A continuación, se muestran pantallazos del prototipo de prueba de aplicación en un dispositivo móvil.

Figura 24. Splash Screen

Figura 25. Solicitud de permisos

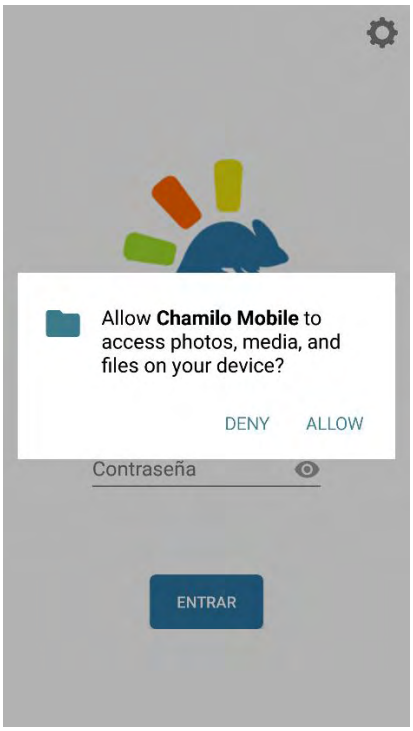


Figura 26. Formulario de logueo

Figura 27. Configuraciones de URL del API

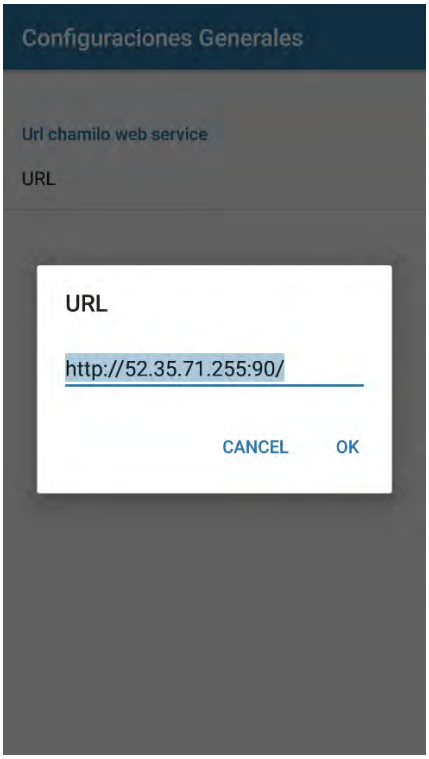


Figura 28. Listado de cursos

Figura 29. Menú

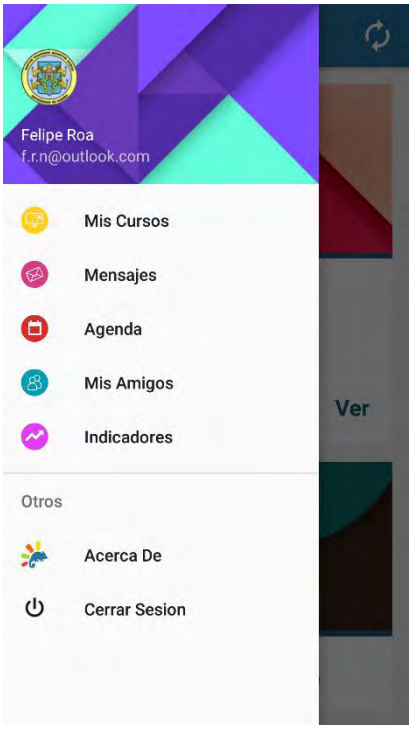


Figura 30. Perfil del usuario

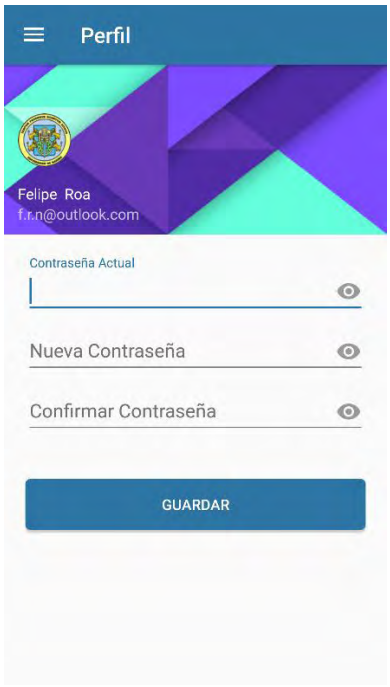


Figura 31. Menú de cursos

Figura 32. Descripción de curso

Figura 33. Documentos del curso

Figura 34. Enlaces del curso

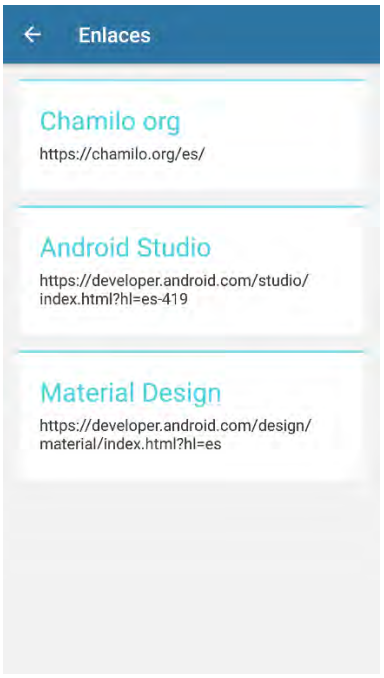


Figura 35. Evaluaciones del curso

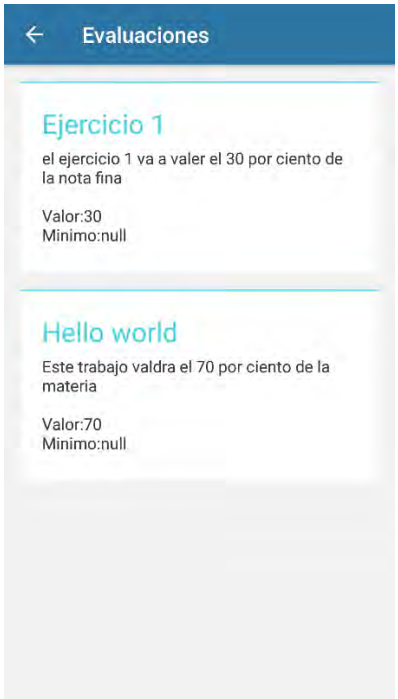


Figura 36. Glosario del curso

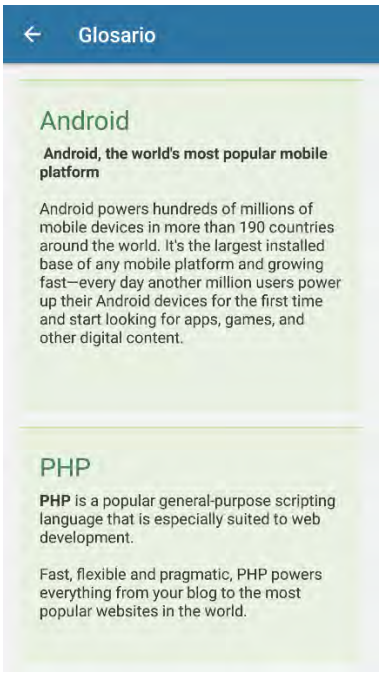


Figura 37. Informes del curso

Figura 38. Usuarios del curso

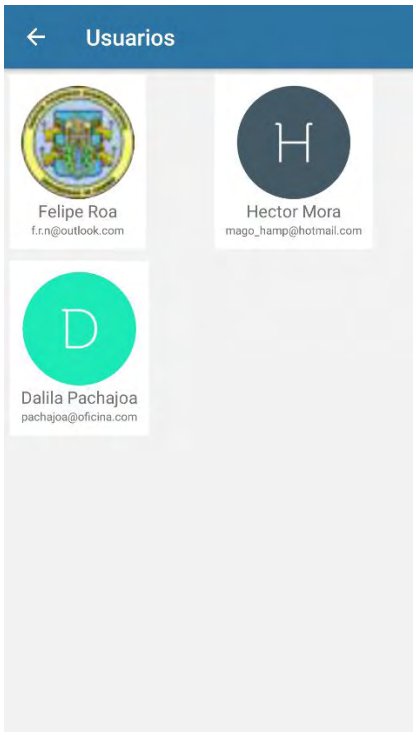


Figura 39. Tareas del curso

Figura 40. Archivos compartidos del curso

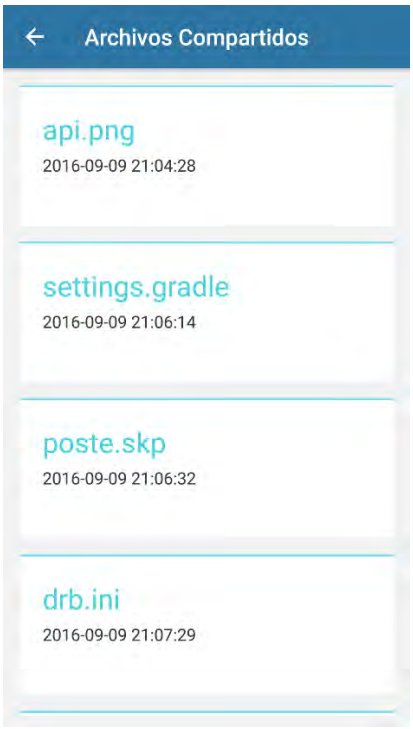


Figura 41. Notas personales del curso

Figura 42. Ejercicios del curso

Figura 43. Conversaciones del usuario

Figura 44. Mensajes del usuario

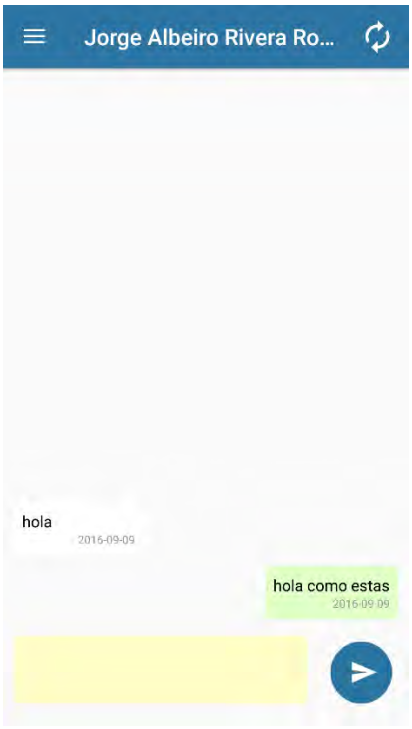


Figura 45. Agenda del usuario

Figura 46. Amigos del usuario

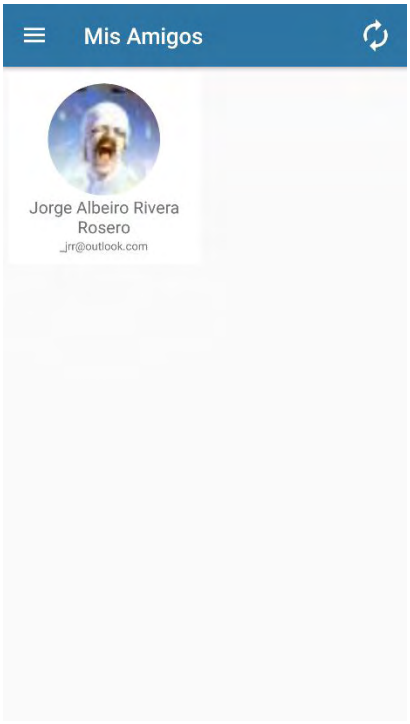


Figura 47. Perfil del amigo del usuario

Figura 48. Indicador accesos en este día

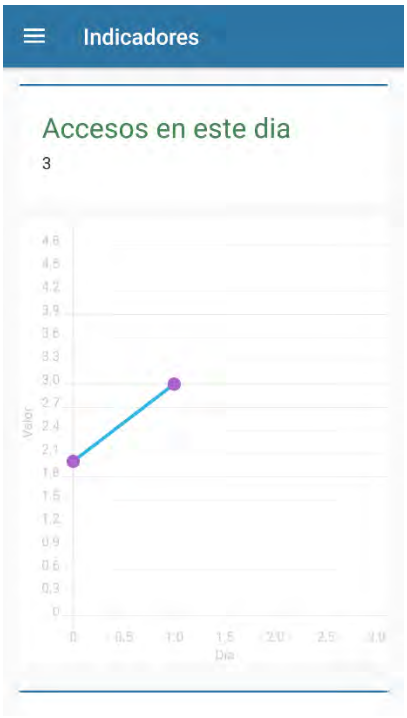


Figura 49. Indicador tareas consultadas en este día

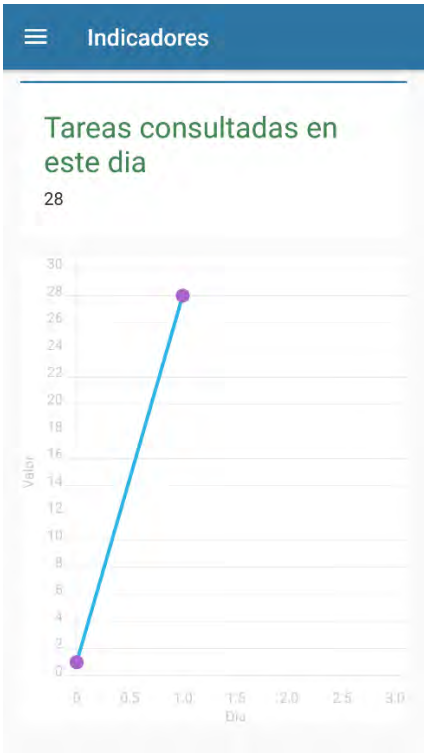


Figura 50. Indicador descargas de documentos en este día

Figura 51. Acerca de

3.1.1 Indicadores

Tabla 6. Tabla de resultados del prototipo de aplicación móvil

SOLICITUD	TAMAÑO DE DESCARGA EN LA PETICIÓN		DURACIÓN DE LA PETICIÓN	
	CHAMILO MOBILE	CHAMILO WEB	CHAMILO MOBILE	CHAMILO LMS WEB
Obtener formulario de logueo	0 bytes	33.6 KB	0 seg	2.39 seg
Inicio de sesión	213 bytes	48.7 KB	182 Ms	2.08 seg
Obtener lista de cursos	450 bytes	36.5 KB	334 Ms	1.17 seg
Seleccionar curso	0 bytes	54.7 KB	0 Ms	1.93 seg
Obtener lista de tareas	1.07 bytes	44.8 KB	315 Ms	2.16 seg

4 CONCLUSIONES

El acceso móvil es la forma como la gente interactúa hoy en día y, por ello, los usuarios tienen expectativas al respecto, como por ejemplo poder realizar las mismas actividades que se hacen en los computadores de la oficina a través de un teléfono inteligente.

La arquitectura de software juega un papel fundamental en el desarrollo de aplicaciones móviles y se cree que al darle un énfasis mayor al actual podría aportar grandes beneficios.

La evolución del sistema operativo Android es constante debido a que se observo que al iniciar el proyecto estaba en la versión 4.0 y ahora nos enfrentamos a la versión 7.0, lo que indica el mejoramiento continuo y que la arquitectura de este sistema operativo está tan bien diseñada que le permite a las aplicaciones funcionar en cualquiera de sus versiones independientemente de en cuál de ellas se produjo.

De acuerdo con los indicadores mostrados se evidencia que es más económico desde los parámetros de consumo de datos y tiempo acceder desde un entorno móvil consumiendo la API desarrollada en este proyecto que acceder directamente desde la Web.

El desarrollo en lenguajes nativos permite hacer un uso más óptimo de los recursos hardware y software de los dispositivos.

5 RECOMENDACIONES

Usar para la transferencia de datos entre diferentes sistemas formatos estándares como JSON en lugar de otros estándares más complejos como XML.

Realizar una separación de capas similar a la propuesta por CHAMILOMOBILE para el desarrollo de futuras soluciones, para aprovechar los insumos de la lógica de negocio de cada solución y de esta manera evitar la reconstrucción de funcionalidades iguales.

Implementar estándares de diseño en el desarrollo de aplicaciones móviles como Material Designer.

Implementar el patrón de desarrollo de MVC para proyectos de desarrollo, así como trabajar en lenguajes nativos para lograr soluciones más óptimas.

Reutilizar artefactos software para la construcción de aplicaciones siempre que sea posible y cuando no lo sea crear nuevos productos de manera que se cree una cultura de reutilización de software y una evolución en la generación de conocimiento.

REFERENCIAS BIBLIOGRÁFICAS

BOULIER, Anaël. Administrator guide Chamilo 1.9. 15 de enero de 2015 Disponible en Internet: http://cdn-chamilo.cblue.be/docs/en/chamilo-admin-guide-1.9-en.pdf.

BOULIER, Anaël. Chamilo 1.9 Teacher's Guide. 10 de abril de 2013 Disponible en http://cdn-chamilo.cblue.be/docs/en/chamilo-teacher-guide-1.9-en.pdf.

ESLAVA, Vicente J. M. El nuevo PHP. Conceptos avanzados. España. 2013. P 146. (ISBN 846864434X).

GIRONA, Jordi, et al. Redes de comunicaciones: De la telefonía móvil a Internet. Universidad Politécnica de Catalunya, 2010. 1 ed. (ISBN 8476539258)

GIRONÉS, Jesus Tomas. El gran libro de Android, Barcelona, 2013. 3 ed.

GROUSSARD, Thierry. Java 7 Los fundamentos de Java. Ediciones ENI. 2012.

MEIER, Reto. Professional Android 4 Application Development. John Wiley & Sons, Inc. Indianapolis, Indiana. 2012. (ISBN 9781118237229)

MEIER, Reto. Professional Android Application Development. Wiley India Pvt. Limited, 2008.

SANTIAGO, R., Trabaldo, Kamijo, Fernández. Mobile Learning: Nuevas realidades en el aula. Editorial Océano. Universidad Cardenal Herrera CEU, Elche Alicante. 2015.