UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

Aplicación móvil de la Plataforma Educativa XAUCE ZERA 2.0 para el Sistema Operativo Android.

Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas

Autor: Gleidys Pérez Díaz.

Tutores: Ing. Agustín Castillo Cordero
Ing. Sandy Nuñez Padrón.
Ing. Manuel Alejandro Oliva Labaut

La Habana, Cuba.
Año 59 de la Revolución
Curso:2016-2017

Pensamiento

Después de escalar una montaña muy alta, descubrimos que hay muchas otras montañas por escalar.

Nelson Mandela

Declaración de autoría

Declaro ser autor del presente trabajo de diploma y autorizó a la Facultad 4 de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

Para que así conste, firmo la presente a los ____ días del mes de _____ del año _____.

Gleidys Pérez Díaz	Ing. Agustín Castillo Cordero
_____	_____
Firma del autor	Firma del tutor
Ing. Sandy Nuñez Padrón	Ing. Manuel Alejandro Oliva Labaut
_____	_____
Firma del tutor	Firma del tutor

Dedicatoria

Dedico esta gran victoria a mis padres y mi hermano por ser los pilares que me hacen ser quien soy, por enseñarme a luchar por mis sueños y ayudarme a cumplir mis metas. Por ser las personas más importantes en mi vida.

Les dedico este trabajo a toda mi familia y amigos, por el amor, dedicación y entrega que han puesto junto a mí para lograr juntos esta meta.

Agradecimientos

Agradezco a mis padres por ser mi apoyo y mi fuerza durante toda mi vida, les agradezco, sobre todo, su comprensión y el amor que me han dado, y les dedico esta victoria por la cual han luchado al igual que yo.

Agradezco a mi hermano por ser esa persona que sabe alegrarme el día y escucharme, por siempre estar para mí cuando lo necesito y darme ese abrazo cuando me hace falta.

Agradezco a mis abuelos: mami y papi, por siempre preocuparse por mí y estar siempre pendientes de mis problemas.
Agradezco a mi tío Juli por todo el apoyo y los consejos que me da, por ser esa persona que te demuestra todo el tiempo que te quiere, y que cuando lo necesitas está a tu disposición.

Le agradezco muchísimo a mi novio Daimel por ser para mí, la persona más especial del mundo y hacerme sentir de la misma manera. Le agradezco todo el amor que me da y todos los bellos momentos que hemos pasado juntos.

Le agradezco a mis grandes amigas Rosa, Lisy y Rosmeri por estar junto a mí en los malos y buenos momentos durante todos estos años, por el respeto y cariño que me dan. Sobre todo, agradecerles su amistad incondicional.

Le agradezco a mis buenos amigos Adrian y Ronnay por todo el apoyo que recibí de ellos durante todo este tiempo.

Les agradezco a mis tutores por el interés que han mostrado durante todo este proceso y por la ayuda que he recibido de ellos.

Al profesor Dosagües por apoyarme y ayudarme con el trabajo, por la preocupación que ha mostrado todo este tiempo.

Agradezco a mis compañeros de aula por todo el tiempo que hemos pasado juntos y los momentos que hemos vivido.

Agradezco en general a todas las personas que se preocuparon por mí y me ayudaron a cumplir esta meta.

Resumen

El presente trabajo de diploma se centra en la necesidad de desarrollar una aplicación nativa para dispositivos móviles con sistema operativo Android, con fines educacionales. Esta aplicación debe conectarse a la Plataforma Educativa XAUCE ZERA 2.0 a través de servicios web, para que los estudiantes puedan visualizar los cursos disponibles en la misma, sin necesidad de estar en un aula o laboratorio, permitiendo introducir la estrategia *m-learning* en la misma. Para dar solución a esta necesidad se realizó una fundamentación teórica, donde se analizaron los conceptos fundamentales relacionados con el tema, para ahondar los conocimientos sobre las aplicaciones móviles asociadas a las Plataformas Educativas. En esta etapa también se realizó un estudio de las tecnologías, herramientas y lenguajes utilizados en el desarrollo de aplicaciones móviles para el sistema operativo Android. Se efectuó un análisis de aplicaciones similares existentes en el ámbito nacional e internacional. Del estudio realizado se especificó el uso de lenguaje de programación Java, Android Studio como entorno de desarrollo integrado y REST como tecnología web para el intercambio de la información entre la plataforma y la aplicación. El proceso de desarrollo de software estuvo guiado por la metodología AUP en su versión UCI, dada la corta duración del desarrollo del producto y variabilidad de los requisitos funcionales de la misma. Se realizó el modelado y diseño de la propuesta de solución, donde se extrajeron los requisitos que debería cumplir la aplicación para cumplir el objetivo planteado. Por último, se realizó la implementación de la aplicación y las correspondientes pruebas para validar su funcionamiento.

Palabras claves: aplicación nativa, *m-learning,* servicios web, plataforma educativa.

Índice general

Contenido

Introducción

La innovación tecnológica en materia de Tecnologías de la Información y la Comunicación (TIC), ha permitido la creación de nuevos entornos comunicativos y precisos que abren la posibilidad de desarrollar nuevas experiencias pedagógicas y expresivas. Así, en la actualidad, a las tradicionales características de la enseñanza presencial y a distancia, se suma la enseñanza en línea, que usa las redes para posibilitar la conexión entre el profesorado y alumnado, propiciando las actividades de enseñanza-aprendizaje (1).

El mundo se encuentra en una época de cambios donde la sociedad, dentro del amplio espectro de las TIC, trata cada vez más de independizarse de los medios computacionales tradicionales. Lo antes mencionado se fundamenta con la utilización de los dispositivos móviles, que con el paso del tiempo han aumentado su nivel de utilidad y funcionalidad. Estos dispositivos se caracterizan por ser lo suficientemente ligeros como para ser transportados por una persona y disponen de la capacidad de batería suficiente como para poder funcionar de forma autónoma. Tienen como características, que poseen diferentes capacidades de procesamiento, conexión permanente o intermitente a una red, memoria limitada, diseños específicos para una función principal y versatilidad para el desarrollo de otras funciones, como son revisar el correo personal, brindar ayuda a través de mapas y sistemas de posicionamiento global, entre otras. Su posesión se asocia al uso individual de una persona, la cual puede configurarlo a su gusto (2).

La influencia de la utilización de estos dispositivos ha llegado hasta la educación ofreciendo varias ventajas, muchas de las cuales requieren replantear metodologías y estándares en la educación. Uno de los principales cambios se encuentra en la forma en que los profesores interactúan con sus alumnos, ya que permite la comunicación en tiempo real entre los mismos.

Para lograr este objetivo se hace uso de redes sociales y aplicaciones de mensajería instantánea entre alumnos y profesores, lo cual agiliza el proceso de comunicación y reduce tiempos. También agiliza y vuelve eficiente la distribución de contenidos y materiales, un ejemplo son las aplicaciones y servicios que permiten almacenar información en la nube, para compartirla con otras personas. Elimina la barrera geográfica en el aprendizaje de manera que no sea necesario que alumnos y profesores estén todo el tiempo en un salón de clases. Permite que el aprendizaje llegue a lugares remotos. Promueve que los estudiantes sean más activos durante el proceso de aprendizaje, ya sea al realizar investigaciones, utilizar nuevas tecnologías, crear documentos y compartirlos. (3)

Dado el auge y las múltiples ventajas del uso de los dispositivos móviles, con el paso del tiempo se ha incrementado la cantidad de aplicaciones educativas desarrolladas para estos dispositivos. De esta forma han pasado a desempeñar un rol importante en la formación y elevación del nivel cultural de los usuarios, lo cual ha posibilitado el nacimiento de un nuevo concepto: “aprendizaje móvil (*m-learning*)” (3).

El *m-learning* se basa en el aprovechamiento de las tecnologías móviles como base del proceso de aprendizaje. Por tanto, es un proceso de enseñanza- aprendizaje que tiene lugar en distintos contextos (virtuales o físicos). El término tecnología móvil se vincula al ámbito de las comunicaciones móviles y describe las capacidades de comunicación electrónica de forma no cableada o fija entre puntos remotos y en movimiento. Las tecnologías móviles propician que el estudiante no precise estar en un lugar predeterminado para aprender y constituyen un paso hacia el aprendizaje en cualquier momento y en cualquier lugar (4).

Lo anteriormente mencionado se confirma teniendo en cuenta las características tecnológicas asociadas al *m-learning*, como son la portabilidad la cual, se evidencia debido al pequeño tamaño de los dispositivos, la inmediatez y conectividad mediante redes inalámbricas. Otra característica es la ubicuidad, que libera al aprendizaje de barreras espaciales o temporales. También existe la posibilidad de incluir accesorios como teclados o lápices para facilitar su uso (4).

En la actualidad ya existen diferentes maneras de gestionar el aprendizaje haciendo uso de componentes computacionales, principalmente a través de las computadoras, de esta manera surgieron las plataformas educativas. Estas pueden ser clasificadas en cuanto a los elementos y características que posean. Como estas plataformas tienen una importante misión educacional, se hace necesario que no se queden obsoletas ante la nueva realidad tecnológica que brinda el desarrollo de los dispositivos móviles. Dada la situación antes mencionada estas poderosas herramientas están evolucionando adoptando las nuevas estrategias que van surgiendo.

El *m-learning* asociado a las plataformas educativas es una manera de construir conocimiento y transmitirlo a un bajo costo, ya que las plataformas constituyen una fuente de información constante, a través de los cursos y actividades que se publican en ellas. A su vez las plataformas permiten incluir recursos multimedia tales como: videos y audios, lo cual permite crear experiencias de aprendizaje independientes y grupales, sin estar todo el grupo en el mismo lugar. También ayuda a los estudiantes a identificar las áreas donde

necesitan ayuda y a los docentes le ofrece mantener una comunicación constante con sus estudiantes. Teniendo en cuenta lo antes mencionado, el docente puede enviar recordatorios, plazos de entrega, comentarios, sugerencias y avisos. En resumen, la utilización del *m-learning* ayuda a combatir la resistencia ante el uso de las TIC.

En Cuba, el uso de las redes inalámbricas, específicamente de la red *wifi,* ha sentado las bases para la aplicación de este método de enseñanza. A pesar de los altos costos de los dispositivos móviles para la mayoría de la población cubana, datos divulgados por la Empresa de Telecomunicaciones de Cuba (ETECSA), confirman el crecimiento en la venta de nuevas líneas de esta importante vía de comunicación (5).

Un ejemplo del uso de los dispositivos móviles en Cuba se evidencia en el uso de los teléfonos móviles. Estos dispositivos se encuentran entre los medios favoritos de los jóvenes, se caracteriza por ser la primera generación que ha crecido con este medio, y por lo tanto lo ve totalmente integrado a su propia vida. Por lo que, si el docente logra dominar alguna de estas capacidades tecnológicas, logrará guiar a los estudiantes en actividades que respondan al programa y a su vez ejerciten las habilidades con el uso de la telefonía, lo cual es muy motivador para ellos.

Por lo cual, el desarrollo de aplicaciones educacionales es una de las líneas de desarrollo por las que apuesta la Universidad de las Ciencias Informáticas (UCI). Esta línea abarca el *software* educativo, tanto para los niveles curriculares de la enseñanza inicial, media o preuniversitaria, así como tecnologías de formación a distancia y semi-presencial, utilizados mayormente en el nivel superior y de postgrado.

La Universidad es la sede de varios centros de producción de *software* y cuenta con diferentes líneas de investigación, en las cuales se ha comenzado a propiciar el uso de la tecnología móvil. Específicamente, en la Facultad cuatro radica el Centro de Tecnologías para la Formación (FORTES), en el cual se desarrolla la Plataforma Educativa XAUCE ZERA 2.0.

La Plataforma Educativa XAUCE ZERA en su versión 2.0, está destinada a apoyar el proceso de enseñanza-aprendizaje en instituciones educativas y empresas. La plataforma no cuenta con una aplicación móvil que posibilite a los estudiantes, el mantener un contacto constante con el contenido de la misma en cualquier momento del día, fomentando con ello una educación individualizada. Además, no aprovecha en su totalidad las posibilidades y ventajas que ofrece el aprendizaje móvil, como es el caso de suplir el déficit de profesores en todas las enseñanzas y fomentar el autoaprendizaje en la población. De esta manera se

daría cumplimiento a el principal objetivo de la plataforma ZERA de apoyar el proceso de enseñanza-aprendizaje en el país.

Lo anteriormente planteado denota como **problema a resolver***:* ¿Cómo desarrollar una aplicación móvil, que contribuya al aprendizaje móvil en la Plataforma Educativa XAUCE ZERAv2?

A partir del problema a resolver se define como **objeto de estudio**, el desarrollo aplicaciones para dispositivos móviles y como **campo de acción**, el desarrollo de aplicaciones móviles para plataformas educativas.

Para dar solución al problema antes descrito se plantea como **objetivo general:** Desarrollar una aplicación para dispositivos móviles con sistema operativo Android, que incluya la estrategia de aprendizaje *m-learning* en la Plataforma Educativa XAUCE ZERA 2.0.

Para dar cumplimiento al objetivo general se plantean los **objetivos específicos** siguientes:

- Elaborar el marco teórico de la investigación mediante el estudio del estado del arte acerca de las tendencias tecnológicas actuales y las estrategias utilizadas en el desarrollo de aplicaciones para el sistema operativo Android.
- Diseñar una aplicación nativa para el sistema operativo Android, que contenga las principales funcionalidades de la Plataforma Educativa ZERA.
- Implementar una aplicación para darle soporte a la Plataforma Educativa XAUCE ZERA 2.0 en dispositivos móviles con sistema operativo Android que contenga características de sistemas similares.
- Realizar pruebas a la propuesta de solución desarrollada.

Preguntas Científicas:

¿Cómo se presenta en la literatura científica el proceso de desarrollo de aplicaciones para el sistema Operativo Android que incluyan la estrategia de aprendizaje m-learning?

¿Cómo modelar una aplicación móvil que cumpla con las principales funcionalidades presentes en la Plataforma Educativa ZERA?

¿Cómo desarrollar la aplicación móvil de la Plataforma Educativa XAUCE ZERA para el sistema operativo Android?

¿Cómo se asegura la calidad de la aplicación móvil de la Plataforma Educativa XAUCE ZERA para el sistema operativo Android, que incluya la estrategia de aprendizaje m-learning en la misma?

Como **posible resultado** se obtendrá una aplicación que permita incluir la estrategia de aprendizaje *m-learning* en la Plataforma Educativa XAUCE ZERA 2.0 con su

correspondiente documentación y brindándole a los usuarios de dispositivos móviles con SO Android una mejor experiencia de usuario.

Tareas a cumplir:

- Realización de un análisis acerca de las diferentes tecnologías para el desarrollo de aplicaciones para móviles con sistema operativo Android.
- Realización de un estudio acerca de la estrategia de aprendizaje *m-learning*.
- Análisis acerca de las metodologías de desarrollo de *software* existentes.
- Identificación de los requerimientos funcionales y no funcionales de la solución propuesta.
- Realización del diseño de la aplicación propuesta.
- Generación de los principales artefactos de la metodología de desarrollo seleccionada.
- Implementación de la aplicación haciendo uso de las tecnologías y herramientas seleccionadas.
- Realización de pruebas a la aplicación.

Métodos investigativos: para llevar a cabo la investigación, se emplearon métodos de nivel teórico y empírico.

Métodos teóricos:

- **Análisis Histórico-Lógico:** en la presente investigación se utilizó este método para realizar el estudio del estado del arte, permitiendo identificar las tendencias sobre las aplicaciones móviles asociadas a las plataformas educativas. También permitió identificar las soluciones similares, que aportaron elementos importantes a la propuesta de solución, los lenguajes y metodologías de desarrollo de software existentes.
- **Analítico-sintético**: se empleó en el análisis de la documentación relacionada con las aplicaciones móviles y el sistema operativo Android, su estructura y las aplicaciones que se desarrollan para él. De igual manera se analizaron las tecnologías que se emplean para el desarrollo de aplicaciones Android y se sintetizaron las ideas con respecto a la implementación de la solución.

Métodos empíricos:

- **Observación:** permitió estudiar de cerca el objeto de la investigación, las acciones, causas y consecuencias logrando conocer la esencia del problema planteado, analizando desde varios puntos de vista la propuesta de solución y otras soluciones existentes e identificando que está hecho y que falta por hacer.

Estructura capitular

Capítulo I: Fundamentación teórica. Constituye la base teórica de la investigación, en la cual son expuestos los principales conceptos que contribuyen al entendimiento del problema en cuestión. Se describen los principales aspectos de las herramientas, tecnologías y metodología a emplear durante el desarrollo de la misma.

Capítulo II: Propuesta de solución. En este capítulo se describe el modelo de dominio y el diseño de las clases del sistema. Se hace un levantamiento de los requisitos funcionales y no funcionales necesarios para lograr que el sistema funcione correctamente. Se describe la arquitectura que se va a utilizar y los patrones de diseño que se utilizan. Se construyen los artefactos correspondientes al diseño y se modela la aplicación.

Capítulo III: Implementación y pruebas. En este capítulo se muestra el modelo de implementación como resultado del diseño anteriormente desarrollado, así como el diagrama de componentes de la aplicación. Se describen las pruebas a realizar, con el objetivo de comprobar el correcto funcionamiento de la aplicación.

Capítulo 1: Fundamentación teórica

En este capítulo se precisan un conjunto de conceptos y fundamentos que constituyen el marco teórico, relacionado con el objeto de estudio definido en la investigación. Se analizan las soluciones similares con el objetivo de conocer las principales funcionalidades que servirán como base para el desarrollo de la solución. Se exponen la metodología de desarrollo y las principales herramientas y tecnologías que serán utilizadas para el desarrollo de una solución al problema planteado.

1.1. Conceptos y definiciones asociados al problema de investigación

1.1.1. M-learning

El incremento de la población ha hecho que los espacios físicos para educar no sean suficientes. Por este motivo, distintas alternativas han aparecido para que el aula no sea necesariamente el único lugar para aprender. La educación a distancia es un ejemplo de estas nuevas alternativas. Las nuevas tecnologías han abierto nuevas aristas en el mercado educativo acortando distancias y tiempos de respuesta en el proceso de enseñanza-aprendizaje, mediante el uso de métodos alternos. El *e-learning* surgió de esta manera con la limitante de la movilidad, ahora el *m-learning* está siendo muy popular tomado en cuenta el surgimiento de los dispositivos móviles, los cuales hacen posible que este se transforme en la variante educacional más aceptada por las nuevas generaciones. (3)

Muchos son los autores que han definido el *m-learning*. Primeramente Manuel Area dice que “el *m-Learning*, con *m* de móvil, es el término utilizado para designar un espacio relativamente nuevo de investigación, producto de la confluencia entre el *e-learning*, entendido en sentido amplio, y los dispositivos móviles de comunicación: ordenadores portátiles y ultraportátiles, PDAs, teléfonos móviles con acceso a Internet, Tablet PC e incluso consolas de videojuegos” (3).

Por su parte la MSc. María J. Vidal Ledo afirma que “el aprendizaje móvil o *m-learning* como también se le conoce, es una metodología de enseñanza-aprendizaje, que se vale del uso de los teléfonos móviles u otros dispositivos móviles, como son las agendas electrónicas y las tabletas o *tablets*, entre otros, con conectividad a *Internet*” (6).

Después de analizar las definiciones dadas por estos dos autores, se puede llegar a la conclusión de que, el *m-learning* es una nueva estrategia educativa, que hace uso de los dispositivos móviles modernos, para apoyar el proceso de enseñanza-aprendizaje en la actualidad.

Ventajas y desventajas de la metodología *m-learning*.

Ventajas funcionales (7):

- Elimina la brecha geográfica.
- Contribuye a disminuir la brecha digital.
- Los dispositivos móviles permiten estar conectados la mayor parte del tiempo.
- Favorece que los alumnos puedan compartir determinadas tareas y actividades con otros compañeros.

Desventajas (7):

- La pantalla pequeña en los dispositivos móviles dificulta la lectura y trae consigo la imposibilidad de mostrar mucha información.
- Dificultades a la hora de instalar y usar determinado software.
- Los estudiantes que no sean usuarios comunes de este tipo de dispositivos, tardarán en adaptarse o se resistirán al cambio.

1.1.2. Dispositivos móviles

Un dispositivo móvil se puede definir, según un grupo de autores, como un aparato de pequeño tamaño, con algunas capacidades de procesamiento, con conexión permanente o intermitente a una red. También cuentan con memoria limitada, que ha sido diseñado específicamente para una función, pero que puede llevar a cabo otras funciones más generales. De acuerdo con esta definición existen multitud de dispositivos móviles, desde los reproductores de audio portátiles hasta los navegadores de Sistema de Posicionamiento Global, pasando por los teléfonos móviles, los PDAs[1] o los *Tablet* PCs (8).

Tipos de dispositivos móviles

Los tipos y modelos de dispositivos móviles que se pueden encontrar varían en cuanto a peso, tamaño, funcionalidades, sistemas operativos que utilizan y servicios que brindan. A continuación, se presentan algunos de ellos (8).

- Celulares

Los celulares son dispositivos electrónicos de comunicación, normalmente de diseño reducido y sugerente, que tiene la misma funcionalidad que un teléfono de línea fija. Son portables e inalámbricos, lo que significa que no necesita de una red cableada para llevar a cabo la conexión a la red telefónica.

[1]*PDAs: computadora de mano originalmente diseñada como agenda personal electrónica.*

Los celulares modernos suelen incorporar un conjunto de funciones adicionales, tales como mensajería instantánea (SMS), agenda y juegos, que aumentan la potencialidad de utilización de estos dispositivos. Su manejo suele ser sencillo, ya sea a través de la escritura sobre una pantalla táctil, o de un pequeño teclado (8).

- *Smartphone*.

Un *Smartphone* es un teléfono móvil construido sobre una plataforma informática móvil, con mayor capacidad de almacenamiento de información de decenas de gigabyte de información y mejor conectividad que un celular. El término inteligente hace referencia a la capacidad de usarse como un ordenador de bolsillo, llegando incluso a remplazar a un ordenador personal en algunos casos.

Estos *Smartphone* cuentan con cámaras, GPS (Sistema de Posicionamiento Global por sus siglas en inglés) y *bluetooth*[2]. Les dan soporte a redes *wifi*[3], 3G y 4G para el acceso a Internet. Sus pantallas generalmente alcanzan más de cinco pulgadas. A algunos dispositivos se le pueden introducir tarjetas de memorias extraíbles (micro SD). Los sistemas operativos más comunes que utilizan para su funcionamiento son: *Android*, *iOS* y *Windows Phone* (8).

- *Tablets*.

Un *tablet* (tableta en español) es un tipo de computadora portátil, de mayor tamaño de pantalla que un teléfono inteligente o un PDA, con una pantalla táctil (sencilla o multitáctil) con la que se interactúa primariamente con los dedos, sin necesidad de teclado físico.

Técnicamente los *tablets* se ven dotados de la mayoría de las características de los *smartphones*. La mayor diferencia está marcada por el tamaño, ya que sus pantallas miden desde siete pulgadas, hasta más de quince en algunos casos. A pesar de que algunos no están provistos de líneas para efectuar llamadas telefónicas, cuentan con servicios de *internet* como video llamadas y *chats* que permiten la comunicación entre varios usuarios (8).

Después de analizar los tipos de dispositivos móviles existentes, las características y capacidades de estos, se definen los *Smartphone* y los *Tablets*, como dispositivos a los que va dirigida la solución de la investigación.

[2] *Bluetooth: especificación industrial para Redes Inalámbricas de Área Personal (WPAN) que posibilita la transmisión de voz y datos entre diferentes dispositivos mediante un enlace por radiofrecuencia.*

[3] *wifi: mecanismo de conexión de dispositivos electrónicos de forma inalámbrica.*

1.1.3. Sistema operativo Android

El sistema operativo (SO en lo adelante) Android es una plataforma de código abierto que admite la construcción de aplicaciones para, de manera sencilla, manejar, configurar y personalizar el dispositivo móvil. Esto hace que se pueda disfrutar de una interfaz amigable en la que se puede priorizar necesidades como correo, *chat*, redes sociales, multimedia, mensajería, etc. Android es un sistema operativo de código abierto para móviles basado en el núcleo de Linux, el cual permite desarrollar ilimitadas aplicaciones para teléfonos inteligentes, tabletas, reproductores MP3, televisores, cámaras. También admite la distribución de los productos generados (9).

Arquitectura de Android

La arquitectura de Android está formada por cuatro capas que permiten la generación de aplicaciones. El acceso a cada capa se realiza por intermedio de bibliotecas, en las cuales se pueden utilizar los elementos de la capa inferior para realizar sus funciones. Las capas de la arquitectura Android según Carlos Alberto Vanegas (9), son:

1. *Linux kernel*: capa de abstracción del hardware, que permite que las aplicaciones accedan a través de controladores asumiendo la administración de los recursos del teléfono y del sistema operativo. En esta capa están disponibles los controladores para pantalla, teclado, cámara, conexión inalámbrica, memoria rápida, audio, cobertura y administrador de energía.
2. Librerías: bibliotecas de Android escritas en C/ C++, que se encargan de realizar la comunicación entre la capa de abstracción de hardware con la interfaz de programación de aplicaciones. Se destacan entre estas librerías la gestión gráfica, *openGL/ES* (gráficas 3D), *SGL* (gráficas 2D), las fuentes de mapas de bits y la seguridad.
3. Entorno de ejecución Android*:* no es considerada una capa, pero está formada por las siguientes librerías: máquina virtual Dalvik en versiones anteriores a Android 5.0 denominado Lollipop y ART[4] a partir de ese lanzamiento.
4. Marco de trabajo de aplicaciones: está conformado por clases y servicios que permiten las funciones básicas de los móviles y la programación de las aplicaciones. En él se encuentran los siguientes elementos: el administrador de actividades, el administrador de ventanas, el compartidor de aplicaciones, las vistas, el administrador de

[4] *ART: entorno de ejecución de aplicaciones utilizado por el sistema operativo móvil Android.*

notificaciones, el administrador de paquetes, el administrador telefónico, el administrador de recursos, el administrador de posicionamiento y las aplicaciones (9). El sistema operativo Android es el escogido para ejecutar sobre él la solución, una vez desarrollada. Se elige este sistema dada la aceptación que tiene en la población, según datos de su página oficial (10). También fue uno de los requerimientos del cliente, que la solución se implemente para este sistema operativo en específico.

1.1.4. Tipos de aplicaciones

Para lograr llevar a cabo el desarrollo de una aplicación móvil primeramente hay que tener conocimiento de cuáles son los principales tipos de aplicaciones que se desarrollan en el mundo, como es el caso de las variantes que se muestran a continuación (11):

Aspectos comparativos	Aplicaciones Web	Aplicaciones Nativas	Aplicaciones Híbridas
Entorno de ejecución	Navegador del dispositivo	Dispositivo	Dispositivo
Conectividad obligatoria	Sí	No	No
Consumo de recursos de *hardware* (RAM,CPU)	Alto	Bajo	Medio
Disponibilidad de la aplicación en el dispositivo	No	Sí	Sí
Acceso a las características del dispositivo	Parcial	Total	Parcial
Velocidad de funcionamiento	Depende de la conexión	Muy rápido	Media

Una vez analizados los tipos de aplicaciones que se desarrollan para los dispositivos móviles con sistema operativo Android, se elige como aplicación a implementar una aplicación nativa. Este tipo de aplicación tiene como principal ventaja que se desarrolla para un sistema operativo especifico, esto eleva su capacidad de procesamiento porque no hay

que utilizar la misma interfaz para todos los sistemas operativos. Además, estas aplicaciones pueden acceder a todas las capacidades de los dispositivos y adaptarse fácilmente a los tipos de pantallas. Trabajan con las notificaciones internas de cada entorno manteniendo al usuario al tanto de nuevas noticias o actualizaciones.

1.1.5. Usos en la educación

Las aplicaciones móviles poseen un papel fundamental en la implementación de la metodología *m-learning* en la educación. Un ejemplo de esto son las disímiles aplicaciones educativas creadas, como es el caso de las plataformas educativas. Diversos sectores de la sociedad han decidido incluir los dispositivos móviles en sus procesos. La educación se ha servido de los dispositivos para apoyar el proceso de enseñanza-aprendizaje. El apoyo que brinda esta estrategia se evidencia en la creación de aplicaciones educacionales para los dispositivos móviles, como es el caso de *Duolingo* la cual está enfocada en el aprendizaje de diferentes idiomas. Estas aplicaciones permiten la interacción a distancia de los estudiantes y los profesores. Con esta interacción a distancia el profesor puede supervisar las actividades prácticas de los estudiantes. La mayor de las ventajas de estas aplicaciones se encuentra en el acceso a la plataforma desde cualquier lugar a través de una conexión a *internet*. Los motivos antes mencionados dan mayor importancia al desarrollo de esta investigación.

1.2. Plataforma Educativa XAUCE ZERA 2.0

Para dar cumplimiento a los objetivos planteados se hace necesario hacer una investigación sobre la Plataforma Educativa XAUCE ZERA. Se analizarán sus principales características y ventajas, por las cuales se hace imprescindible la implementación de una aplicación móvil de la misma.

La Plataforma Educativa XAUCE ZERA 2.0 es capaz de proporcionar a los profesores y aprendices un sistema integrado en línea único, robusto, seguro y fácil de usar. Permite crear ambientes de aprendizaje personalizados que puedan soportar las necesidades tanto de clases pequeñas como de grandes organizaciones, debido a su flexibilidad y escalabilidad. También aporta tecnología educativa innovadora que ayuda a los profesores a adaptarse a las nuevas normas y personalizar el aprendizaje. La plataforma brinda la posibilidad de crear cursos agrupados por categorías. Soporta contenidos interactivos y enriquecidos con elementos psicológicos y didácticos. Cuenta con recursos multimedia a partir de plantillas previamente definidas que se entremezclan con documentos. Además,

acumula presentaciones y foros de discusión, cuestionarios, disímiles tipologías de ejercicios y actividades que pueden ser evaluados de forma automática o directamente por el profesor. La plataforma gestiona grupos académicos, asegura que no haya limitaciones lingüísticas para aprender en línea, dando soporte a varios idiomas, y da soporte a los *MOOC*[5]. Cuenta con un editor de texto enriquecido que funciona con todos los navegadores de Internet y en todos los dispositivos móviles permitiendo darle el formato al texto según el gusto y la intención del usuario. Se presenta con una estructura multi-organizativa que permite ser utilizada por diferentes instituciones educacionales. También contiene herramientas para la comunicación y colaboración entre sus usuarios, como son los foros y las notificaciones. Todo este ambiente de enseñanza-aprendizaje se desarrolla bajo conceptos como: las redes sociales y la *Web* 2.0, los que proveen una mayor interactividad y protagonismo, elementos que marchan acorde a la creciente revolución tecnológica (12). XAUCE ZERA posee 5 subsistemas estrechamente relacionados: Administración, Gestión de recursos, Gestión de Contenidos, Gestión de Cursos y Gestión de Reportes (12).

1.3. Aplicaciones similares

Siguiendo la nueva metodología *m-learning,* en el mundo se han implementado diferentes plataformas educativas. La implementación de una plataforma educativa para la administración de cursos, permite la autonomía de producción y publicación en la red de recursos y contenidos por parte de los participantes. El docente, teniendo en cuenta los objetivos educativos, autónomamente y con la posibilidad de editar el contenido, pone a disposición de sus estudiantes el programa del curso, contenidos o unidades temáticas (expuestos en textos, hipertextos, presentaciones, animaciones, videos), actividades, bibliografía y evaluación. De manera análoga, con otro nivel de autonomía, el estudiante puede acceder a los contenidos y al desarrollo de las actividades propuestas (13).

Las plataformas educativas permiten estimular la idea de cooperación y de interacción, como aspectos centrales del proceso de aprendizaje y enseñanza. De igual manera que favorecen la adquisición de contenidos significativos en los estudiantes y al mismo tiempo, afianzan en los docentes la práctica de la enseñanza mediadas por las TIC (13).

[5] *MOOC (curso online masivo abierto): son cursos en línea dirigidos a un amplio número de participantes a través de Internet según el principio de educación abierta y masiva.*

1.3.1. Plataforma Moddle

Moodle (*Modular Object-Oriented Dynamic Learning Enviroment*) o Entorno Modular Dinámico Orientado a Objetos de Aprendizaje, como su página *web* indica, es un paquete de *software* para la creación de cursos y sitios *web* basados en *Internet.* Fue creada en 1999 por Martin Dougiamas, profesor en la Universidad Australiana de Curtin y está inspirada en la pedagogía constructivista social. Idea que plantea que el conocimiento se va construyendo por el estudiante a partir de su participación activa en el proceso (14).

Moodle es la plataforma más usada, con más de treinta y siete millones de usuarios, aplicada en la construcción de más de cuarenta y ocho mil sitios registrados en doscientos doce países (14).

El funcionamiento de *Moodle* se basa en la interacción de cuatro tipos de usuarios: invitados, estudiantes, profesores y administradores. Los invitados son autorizados por el administrador(es) y por el profesor(es), es el grupo que menos privilegios tiene, por ende, su accionar es limitado. Los estudiantes, en cambio, pueden matricularse en los cursos, participar en las actividades y utilizar sus recursos, así como también formar grupos para interactuar entre ellos y con el profesor. Los administradores poseen todos los privilegios y su principal función es gestionar la información de la base de datos y controlar su acceso. Finalmente, los profesores son los que diseñan las actividades y los materiales de las asignaturas, con base en la aplicación de principios pedagógicos. *Moodle* les permite controlar y evaluar el aprendizaje de cada estudiante y realizar seguimiento de sus avances (15).

Aplicación *Moodle Mobile*

Moodle Mobile (MM) es la *app* oficial de *Moodle*. Es una aplicación la cual utiliza tecnologías *webs* comunes, entre ellas REST (Transferencia de Estado Representacional, siglas en inglés) como protocolo para obtener y enviar información a la plataforma *Moodle*. El diseño se creó usando HTML 5 y CSS 3, la interacción con el teléfono y el envasado se realiza con *PhoneGap27*[6]. Se encuentra disponible para dispositivos con los sistemas operativos *Android* y *iOS*. Esta aplicación es un reemplazo de la anterior *Moddle App* (16).

Entre sus funcionalidades se encuentran:

- Subir fotos y videos a la *web* (seleccionados de la galería o tomados instantáneamente de la cámara del móvil).
- Grabar y subir un archivo de audio.

[6] *PhoneGap27: framework para el desarrollo de aplicaciones móviles.*

- Ver la lista de participantes en los cursos en los que el usuario está inscrito.
- Ver la información de estos participantes.
- Mandar un mensaje privado a algún participante.
- Agregar una nota privada acerca de algún participante.
- Permite el trabajo *off-line* con sincronización automática.
- Agregar los participantes a la lista de contactos del teléfono.
- Acceso a la versión *web* de la plataforma.
- Acceso a los cursos de la plataforma.

1.3.2. Plataforma Blackboard

Es una plataforma que integra un ambiente sólido de enseñanza-aprendizaje en línea. Se caracteriza por administrar un conjunto de recursos que permiten desarrollar cursos virtuales, específicamente: impartir y distribuir contenidos que se encuentran presentados en diversos formatos (texto, sonido, video y animación). Permite realizar evaluaciones en línea, llevar a cabo el seguimiento académico de los alumnos, asignar tareas y desarrollar actividades en ambientes colaborativos (17).

Aplicación *Blackboard Mobile Solutions*

"Blackboard Mobile Solutions es un proyecto que brinda a estudiantes y educadores acceso a los aspectos del proceso de enseñanza-aprendizaje desde sus dispositivos móviles. Este proyecto cuenta con aplicaciones que se complementan entre sí para formar un paquete completo de funcionalidades que brindan al estudiante y al docente total acceso a los aspectos más importantes del centro de estudio al que pertenecen. *Blackboard Mobile Central* cubre los temas que respectan a la residencia escolar como mapas, direcciones, anuncios generales y rutas de ómnibus. Por otro lado, *Blackboard Mobile Learn* se encarga de todo lo referido a la gestión del aprendizaje, brindando acceso a los elementos fundamentales de los cursos a los que el estudiante y el docente se encuentran registrados" (17).

Funcionalidades:

- La realización de pruebas interactivas desde el dispositivo.
- El recibo de notificaciones de tipo *push.*
- El recibo de avisos enviados por estudiantes y docentes.
- Servicio en la nube que permite el almacenamiento de datos.
- El acceso a las evaluaciones obtenidas en clases.
- El acceso a foros y discusiones.

- El acceso a los *blogs.*
- El acceso a la información de los estudiantes que pertenecen al centro de estudio.
- El acceso a los cursos.

1.3.3. Plataforma Educativa ZERA 1.0

La Plataforma Educativa ZERA 1.0, la anterior versión de la plataforma actual, se basa en la concepción pedagógica cubana llamada Hiperentornos de aprendizaje. Estos entornos constituyen medios de enseñanza que le permiten al estudiante apropiarse del contenido de manera reflexiva y consciente. ZERA integra los principales conceptos de los hiperentornos, las mejores prácticas y elementos arquitectónicos de soluciones similares. Características que comparten las principales especificaciones y estándares educativos desarrollados y utilizados a nivel mundial en plataformas de aprendizaje (18).

Aplicación ZERApp

Según el autor de esta aplicación, José González Castellanos, la define como "una aplicación móvil desarrollada para la plataforma educativa ZERA 1.0. Esta aplicación fue realizada para el SO Android y se ejecutó nativamente en los dispositivos móviles con el antes mencionado SO. El desarrollo de esta aplicación permitió el vínculo con la Plataforma Educativa ZERA 1.0, lo que propició que los estudiantes tengan acceso a varias de las funcionalidades que brinda la plataforma y permitió el trabajo *offline* con dicho contenido."

Para la implementación de la aplicación, se utilizaron una serie de servicios *web* creados en la plataforma, que permiten la descarga de contenido a la aplicación, para de esta manera mantenerla actualizada con los últimos cambios realizados (17).

Las funcionalidades implementadas fueron:

- Autenticarse en el sistema con el rol de estudiante.
- Realizar búsqueda de estudiantes por filtros predefinidos.
- Listar notificaciones del estudiante por programa de estudio.
- Actualizar el contenido de la aplicación de forma manual y automática.
- Permitir la configuración de las preferencias de la aplicación.
- Mostrar y descargar los archivos asociados a las evidencias.
- Listar las evidencias compartidas del estudiante por Materia y programa de estudio.

Resumen comparativo

Las aplicaciones anteriormente estudiadas no satisfacen las necesidades de la investigación, ya que son hechas a la medida de la plataforma para la que fueron realizadas, no pudiéndose utilizar en otras plataformas. Otro elemento a tener en cuenta es que estas

aplicaciones son privativas por lo que no se puede tomar su código fuente como base para el desarrollo de otra solución. Sin embargo, al haber sido utilizadas en otros países, se estudió el impacto que estas tuvieron y la aceptación de las funcionalidades desarrolladas, sirviendo como guía para la actual investigación.

En cuanto a la aplicación móvil de la Plataforma Educativa ZERA 1.0, la misma fue desarrollada para esa versión de la plataforma. ZERApp no cumple con las necesidades del cliente y está desarrollada para la versión 3.0 de Android, lo cual la hace incompatible con las versiones actuales de ese SO.

Por otro lado, la versión 1.0 de la plataforma está basada en la teoría pedagógica cubana de Hiperentornos de Aprendizaje y la 2.0 tiene como base los Cursos Masivos Abiertos en línea (MOOC). Además, dichas versiones no están desarrolladas usando la misma versión del Framework de desarrollo PHP Symfony.

El análisis realizado aportó una idea general de las principales funcionalidades que debe poseer la aplicación a implementar para que dar respuesta a los objetivos planteados. También sirvió de apoyo para seleccionar el servicio web que se va a utilizar para el intercambio de datos entre la Plataforma y la aplicación.

1.4. Tecnologías empleadas en el desarrollo de aplicaciones para el sistema operativo Android

Para la implementación de aplicaciones Android se utilizan diferentes tipos de tecnologías y herramientas. Muchas de estas tienen una gran aceptación por parte de los desarrolladores que las utilizan. En este epígrafe se realiza un análisis de las tecnologías y herramientas que se utilizan en la actualidad y cuáles de ellas se eligen para el desarrollo de una solución al problema antes planteado.

1.4.1. Android SDK

Según Jesús Tomás Gironés, “El Android SDK en su versión 24.3.1 (Paquete de desarrollo de software) brinda las librerías y herramientas necesarias para construir, probar y depurar aplicaciones para Android”. A través de este, *Google* ofrece un paquete completo que incluye depurador, emulador, documentación, *demos* y el código de muchas aplicaciones. Este paquete de desarrollo no funciona por sí solo, sino que es necesario que sea importado a algún IDE (19).

1.4.2. Entornos de desarrollo

Un entorno de desarrollo integrado, llamado también IDE, es un programa informático compuesto por un conjunto de herramientas de programación. Puede dedicarse en exclusivo a un solo lenguaje o bien utilizarse para varios. Consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica (20). Ejemplo de estos IDE son:

NetBeans

NetBeans es un entorno de desarrollo integrado (IDE) que permite a los programadores escribir, compilar, depurar y ejecutar programas. Es un entorno modular y escrito en el lenguaje de programación *Java*. Está constituido por un IDE de código abierto y una plataforma de aplicación, las cuales pueden ser usadas como una estructura de soporte general (*framework*) para compilar cualquier tipo de aplicación" (20).

Este IDE vincula con Android a través del *NBAndroid plugin* disponible para las versiones 7.2 y 7.3 del *NetBeans* desde su sitio en *Internet*. Entre las funcionalidades que brinda se encuentran:

- Brinda soporte al núcleo del Android SDK.
- Permite el uso de emuladores y dispositivos reales para ejecutar los proyectos.
- Integra el visor *LogCat29,* imprescindible para saber que está pasando con el proyecto que se encuentra ejecutándose.
- Editores sofisticados para los archivos de tipo XML de *Android*.

Android Studio

Android Studio es un nuevo entorno de desarrollo integrado para el sistema operativo Android lanzado por *Google*. Fue diseñado para ofrecer nuevas herramientas para el desarrollo de aplicaciones y como alternativa al entorno *Eclipse*, hasta aquel momento el IDE más utilizado para el desarrollo de aplicaciones Android. Este entorno de desarrollo, gracias a su sistema de emulación integrado, permite ver los cambios que se realizaron en la aplicación en tiempo real, pudiendo además comprobar cómo se visualiza en diferentes dispositivos *Android,* con distintas configuraciones y resoluciones de forma simultánea (21). Este entorno de desarrollo tiene como ventajas, que permite que la compilación del proyecto se realice de forma rápida y que la ejecución de la aplicación se puede realizar directamente desde el móvil, lo que permite que el desarrollador pueda apreciar cómo se ejecuta su producto y evaluar el comportamiento del mismo en el dispositivo (22).

Para realizar la implementación de la propuesta de solución se escoge como Entorno de desarrollo integrado *Android Studio* en su versión 2.3. En esta versión esta herramienta ofrece un entorno de desarrollo claro y robusto, facilidad para probar el funcionamiento en otros tipos de dispositivos. También posee asistentes y plantillas para los elementos comunes de programación en Android, así como, un completo editor con muchas herramientas extras para agilizar el desarrollo de nuestras aplicaciones (21).

1.4.3. Servicios web

Los servicios *web* se definen como un conjunto de tecnologías con capacidad para operar en la *web* intercambiando datos entre sí con el objetivo de ofrecer servicios. Estos servicios proporcionan mecanismos de comunicación estandarizados para diferentes aplicaciones, interactuando entre sí para presentar información dinámica al usuario (23).

Los Servicios *Web* son sistemas de software diseñados para soportar una interacción máquina a máquina sobre una red. Suelen ser APIs[7] *Web* que pueden ser accedidas dentro de una red (principalmente Internet) y son ejecutados en el sistema que los aloja (24).

Muchas de las aplicaciones desarrolladas para dispositivos móviles en la actualidad, intercambian información a través de *Internet* mediante la citada tecnología, lo que se logra haciendo un correcto uso de los estándares que se proponen, a partir de las necesidades de la aplicación a desarrollar (25).

El estilo REST (*Representational State Transfer*) es una forma ligera de crear Servicios *Web*. REST es una arquitectura que define una serie de principios para el intercambio de datos o indicar la ejecución de operaciones sobre los datos. Los sistemas que siguen los principios REST se les llama RESTful El elemento principal en el que se basan estos servicios son las URLs[8]. En líneas generales se comenta que estos servicios consisten en URLs a las que se accede mediante protocolo HTTP, para conseguir información o realizar alguna operación. El formato de la información que se intercambie con estas URLs lo decide el desarrollador del servicio (24).

Este estilo de arquitectura asociado a los servicios web tiene como características (24):

- Las operaciones se definen en los mensajes.
- Una dirección única para cada instancia del proceso.
- Cada objeto soporta las operaciones estándares definidas.

[7] *APIs: aplicaciones web.*
[8] *URLs (Uniform Resource Locator): Localizador Uniforme de Recursos*

- Componentes débilmente acoplados.

Posee como ventajas (24):

- Bajo consumo de recursos.
- Las instancias del proceso son creadas explícitamente.
- El cliente no necesita información de enrutamiento a partir de la URI inicial.
- Los clientes pueden tener una interfaz “listener” (escuchadora) genérica para las notificaciones.

Para este tipo de servicio web se utilizan diferentes tipos de formatos para el intercambio de información. Para la investigación es de especial interés el formato JSON. Se trata de un lenguaje ligero de intercambio de información. Una de las mayores ventajas que tiene el uso de JSON es que puede ser leído por cualquier lenguaje de programación. Por lo tanto, puede ser usado para el intercambio de información entre distintas tecnologías (26).

Este servicio debe ser implementado en la Plataforma Educativa XAUCE ZERA 2.0, para permitir que la aplicación móvil se conecte a la plataforma, para el intercambio de datos y la extracción de la información.

1.4.4. Herramienta de modelado

Visual Paradigm

Visual Paradigm es una herramienta CASE (del inglés, *Computer Aided Software Engineering*) aplicable en todo el ciclo de vida del desarrollo de software. Soporta el lenguaje de modelado UML (del inglés, *Unified Modeling Language*), SysML (del inglés, *Systems Modeling Language*) y BPMN (del inglés, *Business Process Modeling Notation*). Permite modelar todos los tipos de diagramas de clases, generar código desde diagramas y generar documentación. También proporciona abundantes tutoriales UML, demostraciones interactivas y proyectos. Presenta licencia gratuita y comercial. Es fácil de instalar y actualizar, es compatible entre ediciones (27).

Visual Paradigm también ofrece:

- Navegación intuitiva entre la escritura del código y su visualización.
- Potente generador de informes en formato PDF/HTML.
- Documentación automática Ad-hoc.
- Ambiente visualmente superior de modelado.
- Sincronización de código fuente en tiempo real.

Por todas las características antes descritas de elije esta herramienta para realizar el proceso de modelado de la aplicación.

1.4.5. Lenguajes de desarrollo

Java es un lenguaje orientado a objetos de propósito general. Su principal característica es ser un lenguaje compilado e interpretado. Todo programa en *Java* ha de compilarse y el código que se genera es interpretado por una máquina virtual. De este modo se consigue la independencia de la máquina, el código compilado se ejecuta en máquinas virtuales que si son dependientes de la plataforma. También posee mecanismos para garantizar la seguridad durante la ejecución, comprobando, antes de ejecutar código, que este no viola ninguna restricción de seguridad del sistema donde se va a ejecutar. Cuenta con un cargador de clases, de modo que todas las clases cargadas a través de la red, tienen su propio espacio de nombres para no interferir con las clases locales. Otra característica de es que está preparado para la programación concurrente, sin necesidad de utilizar ningún tipo de biblioteca. Finalmente, posee un gestor de seguridad con el que poder restringir el acceso a los recursos del sistema (28).

XML (Extensible Markup Language)

El lenguaje de marcas extensible (XML) es un simple y flexible formato de texto, derivado del SGML (Lenguaje de Marcado de Anotaciones Generales, siglas en inglés). Originalmente diseñado para la publicación electrónica a gran escala. También desempeña un papel cada vez más importante en el intercambio de una amplia variedad de datos en la *web.* Es mucho más simple que sus parientes SGML y HTML. Presenta dos grandes ventajas como leguaje de representación de datos: es a base de texto y no presenta dependencia en cuanto a la posición de sus elementos (19).

XML es el lenguaje en el cual se etiquetan las vistas de las aplicaciones Android por lo cual será utilizado en la creación de las vistas de la aplicación.

JSON (JavaScript Object Notation - Notación de Objetos de JavaScript)

JSON es un formato ligero de intercambio de datos. Está basado en un subconjunto del Lenguaje de Programación JavaScript. JSON es un formato de texto que es completamente independiente del lenguaje, pero utiliza convenciones que son ampliamente conocidos por los programadores de la familia de lenguajes C, incluyendo C, C++, C#, *Java*, *JavaScript*, *Perl* y *Python* (29).

El formato JSON está constituido por dos estructuras (29):

- Una colección de pares de nombre/valor. En varios lenguajes esto es conocido como un objeto, registro, estructura, diccionario, tabla hash, lista de claves o un arreglo asociativo.
- Una lista ordenada de valores. En la mayoría de los lenguajes, esto se implementa como arreglos, vectores, listas o secuencias.

Estas son estructuras universales, ya que de manera virtual todos los lenguajes de programación las soportan de una forma u otra. Es razonable utilizar este formato para el intercambio de datos dado que es independiente del lenguaje de programación que se utilice (29).

1.5. Metodologías de desarrollo de software

Las metodologías de desarrollo de software surgen ante la necesidad de utilizar una serie de procedimientos, técnicas y soporte documental a la hora de desarrollar un producto de *software*. Debido a las disímiles circunstancias que van surgiendo al enfrentarse a un proyecto de este tipo, no existe una metodología universal que satisfaga todas las necesidades. Tampoco se puede afirmar que una metodología sea mejor que otra, sino que una se ajuste mejor que otra a un determinado proyecto, según las características del mismo. Por estas razones es necesario determinar las características específicas del entorno de trabajo para así poder realizar una correcta selección.

1.6. Metodologías tradicionales

Los llamados métodos pesados se caracterizan por su rigidez metodológica y la exhaustiva documentación. Uno de ellos es *Rational Unified Process (RUP),* basado en *Unified Modeling Language (UML)* para la ingeniería de sistemas y de software.

Según Eucario Parra Castrillón, para la metodología RUP el ciclo de vida de un proyecto se divide en las fases de concepción, elaboración, construcción y transición. En la concepción se define el alcance del sistema, la estimación de costos, la síntesis de la arquitectura candidata y la organización del proyecto. La fase de elaboración comprende las actividades para la refinación y validación de la arquitectura y los componentes. Dentro de la fase de construcción se gestionan los recursos, optimización y control de los procesos y se completa el desarrollo de componentes. En la fase de transición se ejecutan los planes de implementación, se terminan los manuales de usuario y técnicos, se integra el sistema y se ajusta según la validación que hace el usuario (30). Existen otras metodologías tradicionales, pero no es objetivo de la investigación hacer énfasis en ellas.

1.7. Metodologías ágiles

Los métodos ágiles se originaron en el año 2001 por la inestabilidad del entorno técnico, y porque el cliente a veces es incapaz de definir con exactitud los requisitos del proyecto de software. El término ágil se relaciona con la capacidad de adaptarse a los cambios de contexto y de especificaciones que ocurren durante el proceso de desarrollo. Estas metodologías se caracterizan por lo siguiente (30).

- Individuos e interacciones en lugar de procesos y herramientas. Las personas son el factor de éxito más importante en un proyecto de software. Lo apropiado es elegir adecuadamente el equipo de trabajo y que éste configure su entorno.
- Desarrollo de software en lugar de documentación exhaustiva. La documentación debe ser corta y centrarse solo en lo fundamental.
- Trabajo con el cliente en lugar de negociaciones contractuales. Debe existir una colaboración constante entre el cliente y el equipo de desarrollo.
- Apertura para los cambios en lugar de cumplimiento de planes poco flexibles. El éxito o el fracaso de un proyecto depende de la capacidad de adaptación a los cambios en los requisitos, la tecnología y el equipo de desarrollo.

Ventajas del uso de las metodologías ágiles

A continuación, se mencionan algunas de las ventajas del uso de las metodologías agiles en el desarrollo de *software* según Manuel Trigas Gallego (31).

- Se basa en el control empírico, en que se asume que va a haber cambios en el contexto del proyecto, por lo tanto, el control del proyecto se basará en controlar los resultados obtenidos y en función de estos, hacer las adaptaciones adecuadas.
- Las fases se plantean en función de los objetivos del producto, que suelen ser en cortos períodos de tiempo y en los que se hacen demostraciones del producto a los clientes; de esta forma es más fácil realizar los cambios.
- El proceso no necesita de tanto control.
- El cliente es parte del proyecto.
- Todo el equipo participa en todas las fases del proyecto.
- Hay menos roles.
- Se realiza retrospectiva durante todo el proyecto.

Después de analizar las características de los dos diferentes tipos de metodologías existentes, se elige utilizar el enfoque ágil en el proceso de desarrollo del software.

1.7.1. Metodología AUP UCI

AUP UCI es una metodología ágil creada en la Universidad de las Ciencias Informáticas (UCI) con el propósito de hacer converger el trabajo de todos los proyectos productivos de la universidad en cuanto a la metodología que utilizan. Esta metodología es una variación que se le realiza a el Proceso Unificado Ágil. El Proceso Unificado Ágil, de Scott Ambler, o *Agile Unified Process* (AUP) en inglés, es una versión simplificada del Proceso Unificado de *Rational* (RUP). Este describe de una manera simple y fácil de entender la forma de desarrollar aplicaciones de software de negocio usando técnicas ágiles y conceptos que aún se mantienen válidos en RUP (32). El AUP, según Tamara Rodríguez Sánchez, aplica técnicas ágiles incluyendo (32):

- Desarrollo Dirigido por Pruebas (*test driven development* -TDD en inglés)
- Modelado ágil
- Gestión de Cambios ágil
- Refactorización de Base de Datos para mejorar la productividad.

Al igual que en RUP, en AUP se establecen cuatro fases que transcurren de manera consecutiva (32).

- Inicio: El objetivo de esta fase es obtener una comprensión común cliente-equipo de desarrollo del alcance del nuevo sistema y definir una o varias arquitecturas candidatas para el mismo.
- Elaboración: el objetivo es que el equipo de desarrollo profundice en la comprensión de los requisitos del sistema y en validar la arquitectura.
- Construcción: durante la fase de construcción el sistema es desarrollado y probado al completo en el ambiente de desarrollo.
- Transición: el sistema se lleva a los entornos de preproducción donde se somete a pruebas de validación y aceptación y finalmente se despliega en los sistemas de producción.

La nueva variante de la metodología AUP surge dada la no existencia de una metodología de software universal, ya que toda metodología debe ser adaptada a las características de cada proyecto (equipo de desarrollo, recursos), exigiéndose así que el proceso sea configurable. Se decide hacer una variación de la metodología AUP, de forma tal que se adapte al ciclo de vida definido para la actividad productiva de la UCI (32).

Una metodología de desarrollo de software tiene entre sus objetivos aumentar la calidad del software que se produce, de ahí la importancia de aplicar buenas prácticas, para ello

nos apoyaremos en el Modelo CMMI-DEV v1.3. Este modelo constituye una guía para aplicar las mejores prácticas en una entidad desarrolladora. Estas prácticas se centran en el desarrollo de productos y servicios de calidad. De esta manera surgen las siguientes modificaciones, como es el caso de las fases. En la fase de Inicio se modificó el objetivo de la misma, las restantes 3 fases de AUP, antes expuestas, se unifican en una sola, a la que llamaremos Ejecución y se agrega la fase de Cierre. Otra de las modificaciones de la anteriormente mencionada metodología se realizó a las disciplinas de la misma, las cuales se decide para el ciclo de vida de los proyectos de la UCI, tener 7 disciplinas, pero a un nivel más atómico que el definido en AUP. Los flujos de trabajos: Modelado de negocio, Requisitos y Análisis y diseño en AUP están unidos en la disciplina Modelo, en la variación para la UCI se consideran a cada uno de ellos disciplinas. Se mantiene la disciplina Implementación, en el caso de Prueba se desagrega en 3 disciplinas: Pruebas Internas, de Liberación y Aceptación. Las restantes 3 disciplinas de AUP asociadas a la parte de gestión para la variación UCI se cubren con las áreas de procesos que define CMMI-DEV v1.3 para el nivel 2, serían CM (Gestión de la configuración), PP (Planeación de proyecto) y PMC (Monitoreo y control de proyecto) (32).

Para el Modelado del dominio se plantean tres formas de encapsular los requisitos Casos de uso del sistema(CUS), Historias de usuario (HU) y por último la Descripción de requisitos de procesos(DRP), de los cuales surgen cuatro escenarios para modelar el sistema en los proyectos quedando de la siguiente forma (32):

Escenario No1: proyectos que modelen el negocio con CUN solo pueden modelar el sistema con CUS.

Escenario No2: proyectos que modelen el negocio con MC solo pueden modelar el sistema con CUS.

Escenario No3: proyectos que modelen el negocio con DPN solo pueden modelar el sistema con DRP.

Escenario No4: proyectos que no modelen negocio solo pueden modelar el sistema con HU.

El autor de la presente investigación escoge el escenario No4 para realizar la descripción de los requisitos, debido a que en la investigación no se modela el negocio, solo se puede modelar el sistema. El cliente estará siempre acompañando el desarrollo del producto, para convenir los detalles de los requisitos funcionales y apoyar las fases de implementación y

pruebas. También es el escenario que se recomienda para los proyectos de poca duración, porque las historias de usuario no deben poseer demasiada información.

1.8. Conclusiones parciales

- El estudio y análisis del estado del arte permitió obtener un mejor entendimiento sobre las aplicaciones nativas que se implementan para el sistema operativo Android, las cuales se vinculan a las plataformas educativas permitiendo introducir el *m-learning* en estas.
- Del análisis de las aplicaciones similares se determinó que estas no cumplen con las necesidades del cliente y se identificaron varias funcionalidades que ofrecen las aplicaciones móviles vinculadas a plataformas educativas, como fue el caso de visualizar el contenido de los cursos.
- Se describe la metodología de desarrollo AUP en su variante UCI, elegida para regir el desarrollo de la propuesta de solución porque se ajusta al ciclo de desarrollo de la misma.
- Se analizaron las tecnologías y herramientas que se utilizan en el desarrollo de aplicaciones nativas del sistema operativo Android, eligiéndose para el desarrollo de la solución *Android Studio* en su versión 2.3 como entorno de desarrollo integrado, con SDK en su versión 24.3.1; como lenguaje de programación, Java y XML para el etiquetado; como herramienta para el modelado se elige Visual Paradigm en su versión 8.0.

Capítulo 2: Propuesta de solución

En el presente capítulo se aclaran los conceptos y las relaciones que existen entre ellos mediante la realización del modelo de dominio. Se realiza el levantamiento y especificación de los requisitos funcionales y no funcionales a implementarse. Se plantea la propuesta de solución al problema de investigación. Se describe la arquitectura a utilizar y los patrones de diseño empleados en la implementación. Además, se desarrollarán los artefactos pertinentes según la metodología de desarrollo elegida.

2.1. Modelo de dominio

El modelo de dominio ayuda a la comprensión del estado de los procesos que se desean desarrollar. Una vez investigado los principales conceptos asociados al negocio, se presenta mediante un diagrama la relación existente entre estos.

2.1.1 Conceptos del dominio

Estudiante: usuario o persona que interactúa con la aplicación.

Aplicación Android: es la aplicación con la cual interactúa directamente con el usuario y consumiendo los servicios de la plataforma, muestra al usuario los cursos que esta posee.

Plataforma Educativa XAUCE ZERA 2.0: plataforma educativa destinada a apoyar el proceso de enseñanza-aprendizaje, que les permite a los usuarios del sistema intervenir en un ambiente dinámico e interactivo.

Notificación: es el tipo de alerta que se envía la plataforma para mantener al usuario informado.

Curso: modo de presentar el conocimiento acerca de una asignatura determinada, es la forma de mostrar el conocimiento del cual se desea que se apropie el usuario.

Recurso: materiales de apoyo que utilizan los cursos para hacer llegar al usuario un conocimiento dado. Estos recursos pueden ser de diferentes tipos (imágenes, vídeos y audios).

2.1.2 Diagrama del modelo de dominio

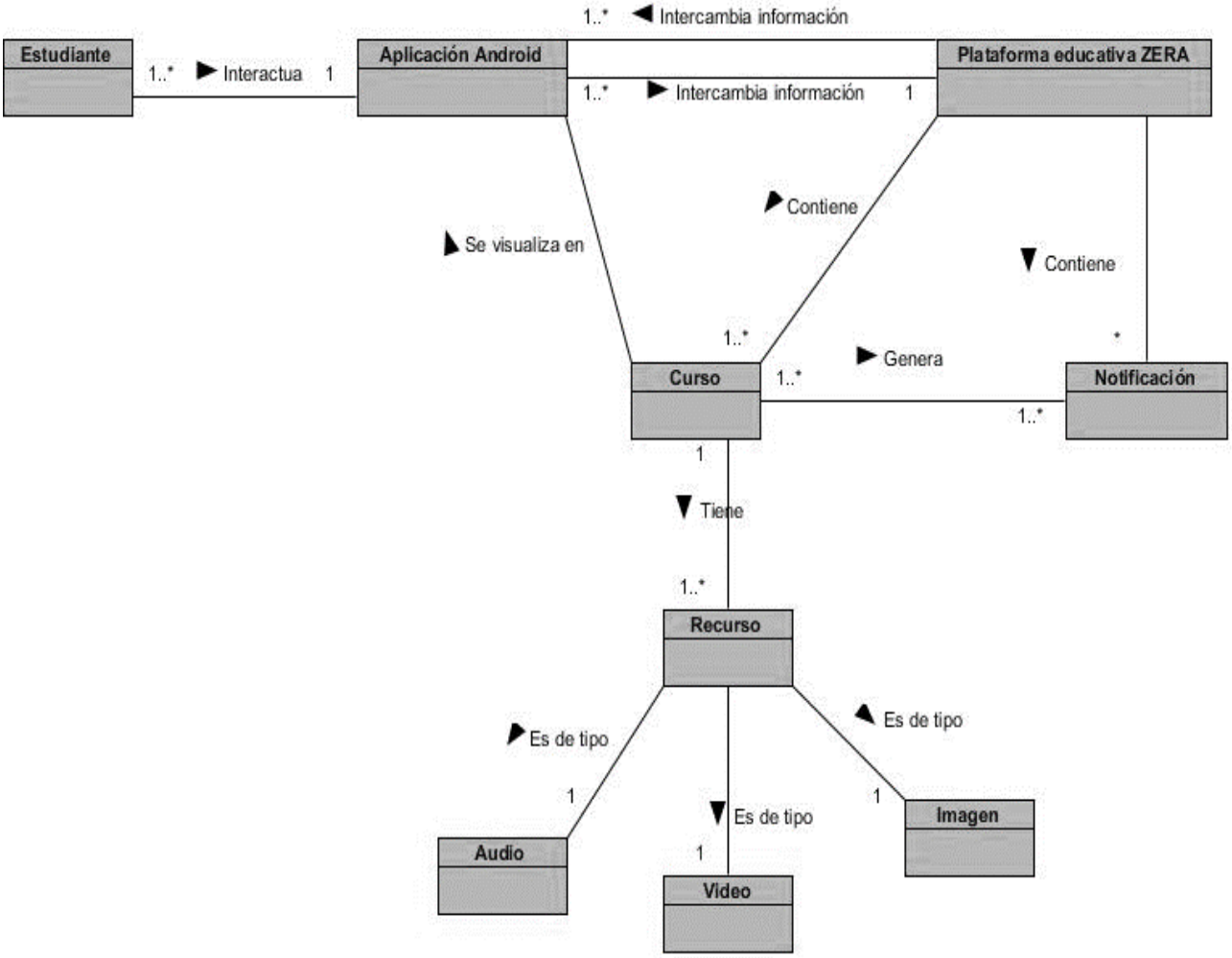


Figura #1: Diagrama de modelo de dominio

2.2. Descripción de la propuesta solución

El sistema propuesto se corresponde con una aplicación para dispositivos móviles con sistema operativo *Android*. Esta se ejecutará nativamente en dichos dispositivos y permitirá el vínculo con la Plataforma Educativa XAUCE ZERA 2.0. La aplicación posibilitará que los estudiantes tengan acceso a los cursos que brinda la plataforma en cada programa de estudio. Además, se utilizarán una serie de servicios *web* creados en la plataforma, que permiten visualizar el contenido de los cursos y que los usuarios puedan recibir notificaciones desde la plataforma. La solución propuesta se convertirá en una aplicación que mantenga al estudiante informado y pueda gestionarse su propio conocimiento como resultado del estudio de los cursos que ofrezcan, enriqueciendo así el proceso de enseñanza-aprendizaje. Los estudiantes podrán acceder a esta información desde

cualquier lugar que se encuentren y en cualquier momento, cumpliendo así con el objetivo de integrar la estrategia de aprendizaje *m-learning* en la Plataforma Educativa XAUCE ZERA 2.0.

2.3. Requerimientos del Sistema

La especificación de requisitos de *software* (ERS) es una descripción completa del comportamiento del sistema que se va a desarrollar. Incluye un conjunto de requisitos funcionales que describe todas las interacciones que tendrán los usuarios con el *software*. La ERS también contiene requisitos no funcionales (o complementarios). Los requisitos no funcionales son requisitos que imponen restricciones en el diseño o la implementación (32).

2.3.1. Requisitos funcionales del sistema

Un **requisito funcional** define una función del sistema de *software* o sus componentes. Una función es descrita como un conjunto de entradas, comportamientos y salidas. En otras palabras, los requisitos funcionales constituyen las características que debe cumplir el sistema como se debe comportar el mismo una vez terminado el desarrollo del producto (33). A continuación, se muestran los requisitos funcionales que debe cumplir la aplicación.

RF 1. Autenticar usuario. Haciendo uso de los servicios de la plataforma, se valida que el nombre del usuario y la contraseña entrada por el usuario estén registrados en la plataforma.

RF 2. Ver notificaciones. La aplicación debe permitir que el usuario reciba las notificaciones que envía la plataforma.

RF 3. Visualizar cursos. La aplicación debe mostrar al usuario todos los cursos que estén disponibles, así como aquellos en los que esté matriculado ya.

RF 4. Matricular en un curso. El usuario puede matricularse en cualquiera de los cursos que estén disponibles.

RF 5. Darse baja de un curso. El usuario debe poder darse baja del curso en el cual esté matriculado.

RF 6. Visualizar árbol de contenido de los cursos. La aplicación debe permitir al usuario que una vez matriculado en el curso pueda ver cuantos temas tiene el curso, así como la composición de cada tema.

RF 7. Visualizar contenido del curso. La aplicación debe permitir que el usuario visualice el contenido del curso.

RF 8. Ver detalles del curso. La aplicación debe permitir al usuario visualizar la descripción del curso en el que se desea matricular.

2.3.2. Requisitos no funcionales del sistema

Un **requisito no funcional** o atributo de calidad, es un requisito que especifica criterios que pueden usarse para juzgar la operación de un sistema en lugar de sus comportamientos específicos (33).

Por tanto, se refieren a todos los requisitos que no describen información a guardar, ni funciones a realizar, sino características de funcionamiento.

Usabilidad

RNF 1. La aplicación podrá ser utilizada por usuarios con conocimientos básicos en el manejo de dispositivos móviles que posean SO *Android*.

Confiabilidad

RNF 2. Sólo los usuarios con una combinación de nombre de usuario y contraseña válidos podrán acceder a la información de la plataforma. Restringir el acceso sólo para los usuarios que estén registrados en la plataforma.

Eficiencia

RNF 3. Las peticiones a la plataforma a través de *Internet*, deben realizarse en un hilo aparte del principal, permitiendo así que el usuario nunca deje de interactuar con la aplicación.

Soporte

RNF 4. La aplicación debe adaptarse a los distintos tamaños de pantalla que presentan los dispositivos móviles en los cuales se utilice.

RNF 5. La aplicación necesitará de una conexión a para el intercambio de información con la plataforma.

2.4. Historias de usuario

Las historias de usuario (HU) constituyen una de las técnicas que utiliza la metodología AUP UCI para especificar los requisitos del *software*. Son tarjetas de papel en las cuales se

describen brevemente las características que el sistema debe poseer, utilizando para ello un lenguaje no técnico. Las Historias de Usuario son un enfoque de requerimientos ágil que se focaliza en establecer conversaciones acerca de las necesidades de los clientes. Son descripciones cortas y simples de las funcionalidades del sistema (34).

Cada HU es lo suficientemente comprensible y delimitada para que pueda ser implementada entre una y tres semanas, asignándoles a cada una de las tareas de programación un número de horas de desarrollo. Son la base para las pruebas funcionales, porque a partir de ellas se realizan los casos de pruebas, que son la entrada para la realización de las pruebas funcionales, que se utilizan para verificar si la propuesta desarrollada cumple con lo que especifica la HU (34).

En las siguientes tablas, se pueden visualizar dos de las historias de usuario realizadas en la presente investigación el resto se pueden ver en el Anexo # 1.

Tabla # 1: Autenticar usuario.

Número: 1	**Nombre del requisito:** Autenticar usuario
Programador: Gleidys Pérez Díaz	**Iteración Asignada:** 1ra
Prioridad: Alta	**Tiempo Estimado:** 2 días
Riesgo en Desarrollo: N/A	**Tiempo Real:** 1 días
Descripción: **1- Objetivo:** Permitir que el usuario se autentique para realizar la matrícula en los cursos ofertados. **2- Acciones para lograr el objetivo (precondiciones y datos):** El usuario se autenticará cada vez que acceda a la aplicación para lo cual debe estar registrado anteriormente en la Plataforma Educativa XAUCE ZERA 2.0, es decir ser usuario del sistema. El usuario debe de estar conectado a una red inalámbrica para realizar estas acciones. El sistema mostrará los siguientes campos: -Usuario: campo de texto obligatorio donde se debe poner el usuario en letras minúsculas. -Contraseña: campo de texto obligatorio donde se debe poner la contraseña definida por el	

usuario cuando se registró.

3- Flujo de la acción a realizar:

La aplicación debe permitir que el usuario introduzca su nombre de usuario y la contraseña. Una vez introducido los datos correctamente y quede autenticado el usuario, el sistema muestra las siguientes opciones:

-Los cursos en los que el usuario está matriculado.

Si los datos introducidos son incorrectos o incompletos se mostrarán los campos en cuestión, dando la posibilidad de realizar la acción nuevamente.

La aplicación debe validar, haciendo uso de los servicios de la plataforma, que el usuario y la contraseña sean correctos.

Observaciones: El permiso al usuario se le da desde la plataforma.

Prototipo de interfaz:

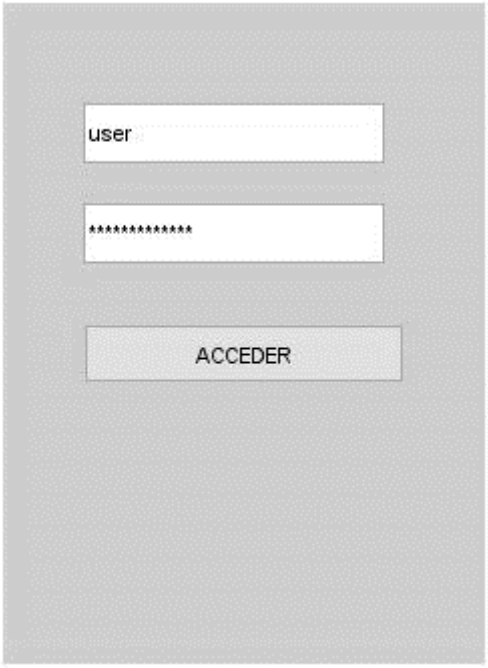


Tabla # 2: Visualizar Cursos.

Número: 3	**Nombre del requisito:** Visualizar cursos
Programador: Gleidys Pérez Díaz	**Iteración Asignada:** 1ra
Prioridad: Alta	**Tiempo Estimado:** 10 días
Riesgo en Desarrollo: N/A	**Tiempo Real:** 10 días

Descripción:

1- Objetivo:

Permitir que el usuario visualice todos los cursos de la plataforma en los cuales se puede matricular, así como aquellos en los que ya este matriculado.

2- Acciones para lograr el objetivo (precondiciones y datos):

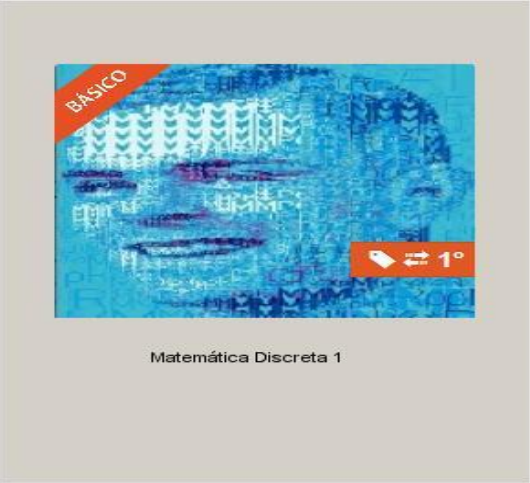
El usuario puede visualizar todos los cursos que posea la plataforma y matricularse en estos. De estos cursos se puede visualizar una foto sugerente del mismo y el nombre.

3- Flujo de la acción a realizar:

La aplicación permite que el usuario pueda ir directamente a los cursos en los que está matriculado o puede elegir la opción matricularse en los cursos para poder visualizar los cursos en los que se puede matricular.

Observaciones: El usuario debe de estar conectado a una red inalámbrica para poder realizar estas acciones.

Prototipo de interfaz:



2.5. Descripción de la arquitectura

El diseño arquitectónico representa la estructura de los datos y de los componentes del programa que se requieren para construir un sistema basado en computadora. Considera el estilo de arquitectura que adoptará el sistema, la estructura y las propiedades de los componentes que lo constituyen y las interrelaciones que ocurren entre sus componentes arquitectónicos. Resumiendo según Pressman "la arquitectura es la estructura de organización de los componentes de un programa (módulos), la forma en la que éstos interactúan y la estructura de datos que utilizan" (35).

La arquitectura del *software* por otra parte, es la estructura o estructuras del sistema, lo que comprende a los componentes del *software*, sus propiedades externas visibles y las relaciones entre ellos" (35).

La Arquitectura Modelo-Vista-Controlador (*Model View Controller*, en inglés) o MVC como se llama popularmente, es un patrón de diseño de *software* para el desarrollo de aplicaciones *web*. Un patrón *Model View Controller* se compone de las tres partes siguientes (36):

- Modelo: es el nivel más bajo del patrón responsable de mantener los datos.
- Vista: es responsable de mostrar todo o una parte de los datos al usuario.
- Controlador: es un código de *software* que controla las interacciones entre el Modelo y la Vista.

En el caso de la presente investigación se elige está arquitectura ya que es la predeterminada para las aplicaciones nativas de Android. Esta arquitectura permite un mayor rendimiento y bajo acoplamiento. También facilita la realización de las pruebas (37). La arquitectura MVC para adaptarse al diseño de las aplicaciones Android define los componentes de la arquitectura de la siguiente manera:

El modelo suele ser una clase de objeto java simple. Es responsable de la lógica del negocio. Por ejemplo, recuperar datos (imágenes, video, texto) desde un servidor remoto (37).

La vista es la combinación de archivo de recursos de diseño y Actividad o Fragmento. No es solo el recurso de diseño porque el recurso de diseño no tiene control total sobre el sistema. Por ejemplo, todos los widgets de los archivos de diseño deben crear Actividades

o fragmentos antes de que se muestren en la pantalla. Además, el archivo de disposición no puede controlar la visibilidad del widget dinámicamente (37).

El controlador es la actividad o fragmento. Es la encargada de capturar una serie de acontecimientos y enviar la respuesta o solicitudes al componente de modelo para obtener y actualizar datos. Por ejemplo, el evento click se captura en el método onClickListener () en Actividad. En Android, el componente View es el archivo XML y *Activity* o *Fragment*. El controlador es el Actividad o Fragmento. Aparentemente, los componentes View y Controller se superponen en Actividad y Fragmento (37).

Para un mejor entendimiento a continuación se explica el significado de Actividad y Fragmento.

"Una actividad es una cosa única y enfocada que el usuario puede hacer." La interacción de los usuarios con la aplicación es principalmente a través de la actividad. En resumen, una actividad es una pantalla completa mostrada en los dispositivos (37).

"Un Fragmento es una pieza de la interfaz de usuario de una aplicación, o un comportamiento que puede ser colocado en una actividad ". Fue introducido por primera vez en Android 3.0. Según su definición, el fragmento es similar a la actividad, la principal diferencia es que la actividad es la pantalla completa, mientras el fragmento forma parte de la pantalla. El fragmento es una parte de la aplicación flexible. Por ejemplo, en la tableta, la pantalla suele dividirse en dos partes: izquierda y derecha. Muy probablemente, la parte izquierda y derecha son dos fragmentos que fueron incorporados en una actividad (37).

2.6. Patrones de diseño

El diseño basado en patrones crea una aplicación nueva, encontrando un conjunto de soluciones comprobadas para un grupo de problemas delineados con claridad. Cada problema y su solución está descrito por un patrón de diseño catalogado y analizado por otros ingenieros de *software* que han encontrado y resuelto el problema mientras diseñaban otras aplicaciones. Cada patrón de diseño provee un enfoque demostrado para una parte del problema que debe resolverse (35).

Un patrón de diseño es una regla de tres partes que expresa una relación entre cierto contexto, un problema y una solución. En la presente investigación se utilizaron diversos patrones de diseño los cuales se exponen a continuación.

2.6.1. Patrones GRASP

Los autores Gloria L Giraldo y Juan F Acevedo enuncian que: “Lo esencial de un diseño de objetos lo constituyen las interacciones de objetos y la asignación de responsabilidades. Las decisiones que se tomen pueden influir profundamente en la extensibilidad, claridad y mantenimiento del sistema de software de objetos, además en el grado y calidad de los componentes reutilizables, por esta razón, durante el diseño se deben utilizar objetos basado en los patrones GRASP[9] ” (38). A continuación, se evidencian los patrones de este tipo utilizados.

Controlador: proporciona guías acerca de las opciones generalmente aceptadas y adecuadas para manejar eventos. Es conveniente utilizar la misma clase controlador para todos los eventos del sistema de un requisito, de manera que es posible manejar la información acerca del estado del caso de uso en el controlador. Se utilizó en las clases controladoras MainActivity.java, Autenticar.java y ServicesController.java.

Creador: guía la asignación de responsabilidades relacionadas con la creación de objetos, una tarea muy común. La intención básica del patrón es encontrar un creador que necesite conectarse al objeto creado en alguna situación. Se utilizó este patrón en la clase ServicesController.java para la creación de objetos.

Experto: se utiliza con frecuencia en la asignación de responsabilidades, es un principio de guía básico que se utiliza continuamente en el diseño de objetos. Expresa la intuición común de que los objetos hacen las cosas relacionadas con la información que tienen. Este patrón se utilizó en las clases LoginFragment.java, NotificationsFragment.java y CursoListFragment.java. En general se utilizó en todas las clases Fragment de la aplicación.

2.6.2. Patrones GOF

Singleton (instancia única): garantiza la existencia de una única instancia para una clase y la creación de un mecanismo de acceso global a dicha instancia. Restringe la instanciación de una clase o valor de un tipo a un solo objeto. Se utilizó este patrón en la clase MyAplication.java para crear una única instancia de la misma y llamar de ella sus métodos en cualquier otra clase del proyecto.

[9]*GRASP: Acrónimo de Patrones de Software de Asignación de Responsabilidad General.*

2.7. Modelo de diseño

El modelo de diseño, es una abstracción de la implementación del sistema, que describe la realización física de los requisitos, centrándose en cómo los requisitos funcionales y no funcionales intervienen en el desarrollo de la aplicación. En el modelo de diseño se representan todas las clases del diseño, paquetes y las relaciones entre ellos, constituyendo la entrada principal de las tares que se realizan en la fase de implementación (39).

2.7.1. Diagramas de clases del diseño

Los diagramas de clases del diseño (DCD) son una representación concreta y detallada que los diagramas de clases del análisis, aunque también representan la parte estática del sistema conteniendo las clases y sus relaciones. Son empleados para representar las relaciones que se establecen entre las clases y constituyen una fuente de apoyo durante el proceso de implementación (39).

A continuación, se hace una breve descripción de los componentes principales que conforman un diagrama de clases del diseño con el objetivo de lograr un mayor entendimiento.

Paquete Vista: contiene las clases que muestran la información al usuario como resultado de ejecutar una acción determinada. Las clases *layout* son las encargadas de mostrar la información al usuario. Como tal ellas no gestionan la información que muestran, sino que esta información se gestiona en los fragmentos y las actividades, en su totalidad forman las vistas del sistema.

Paquete Controlador: contiene las clases encargadas de relacionar la lógica del negocio con la presentación. Este paquete está conformado por el fragmento, el cual controla el orden y la lógica en que se muestra la información, esta información se obtiene a través de las actividades, las cuales acceden a los datos almacenados en el modelo.

Paquete Modelo: contiene las entidades generadas en correspondencia con las tablas de la base de datos que almacenan toda la información que maneja la aplicación. Estas clases obtienen sus valores, cuando se ejecutan los métodos que hacen las llamadas a los servicios web.

A continuación, se muestra el DCD correspondiente a las historias de usuario Autenticar Usuario y Visualizar Cursos. Para el estudio de los demás diagramas de clases del diseño ver Anexo #2.

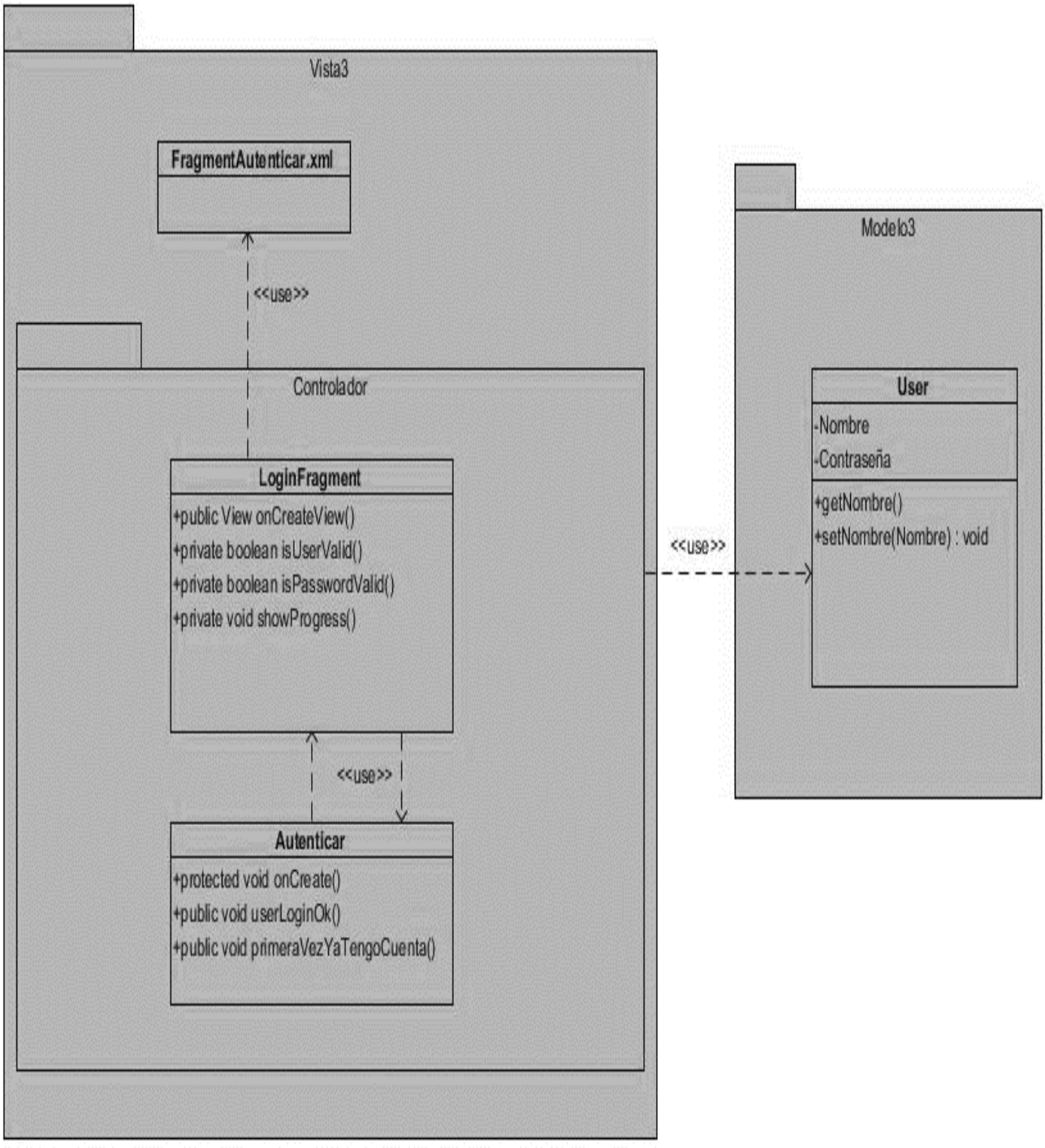


Figura # 2: Diagrama de clases del diseño Autenticar Usuario

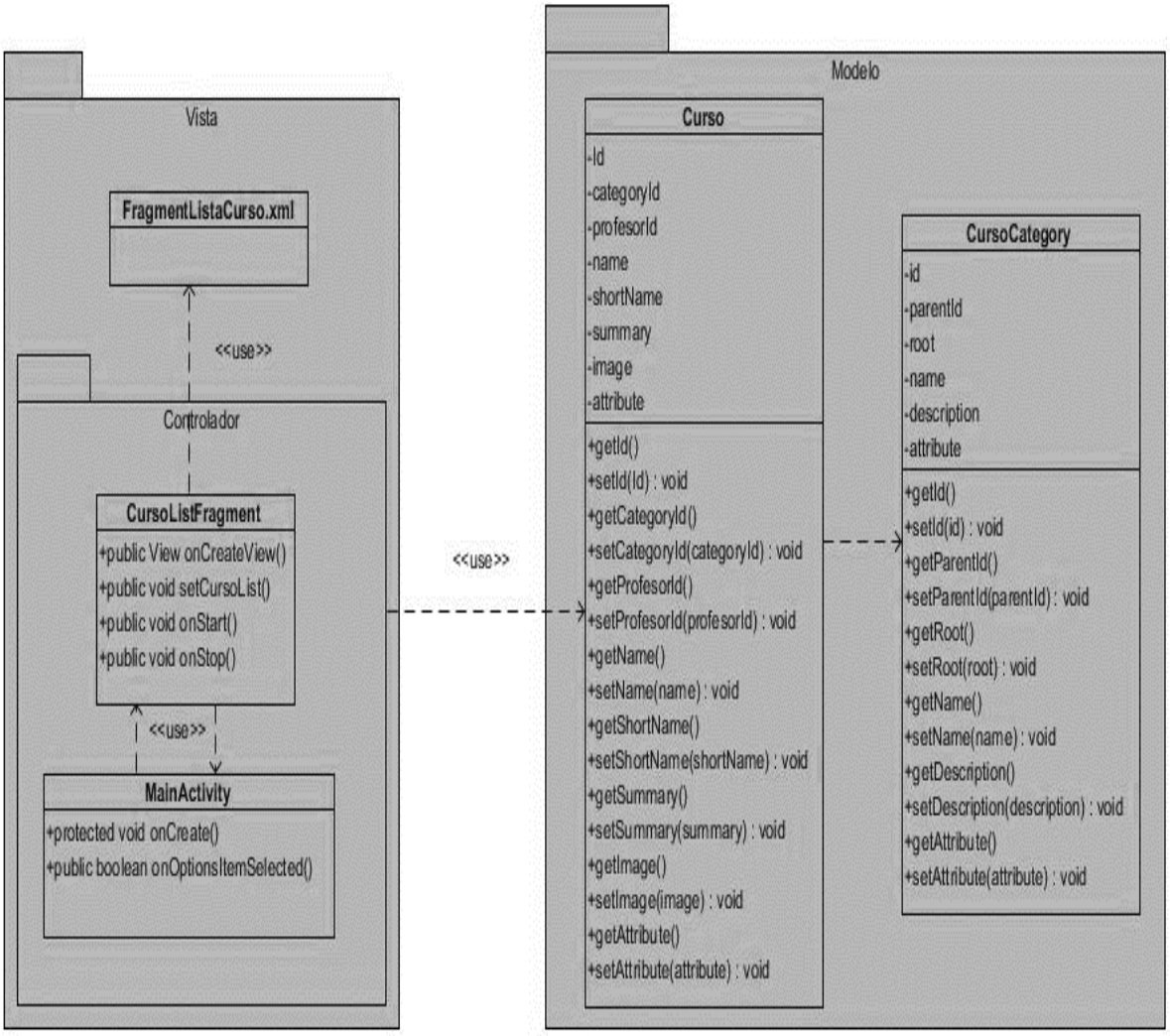


Figura # 3: Diagrama de clases del diseño Visualizar Cursos

2.7.2. Diagramas de secuencia del diseño

Los diagramas de secuencia del diseño (DSD) muestran la iteración ordenada entre los objetos y los mensajes que se intercambian según una secuencia temporal de los eventos. En resumen, el diseño de este artefacto permitirá representar la interacción de los objetos que intervienen en una funcionalidad determinada, mediante la transferencia de mensajes o métodos.

En el caso de esta investigación se realizó el correspondiente DSD para el requisito funcional Autenticar Usuario y Visualizar Cursos, para el estudio de los demás ver Anexo#3.

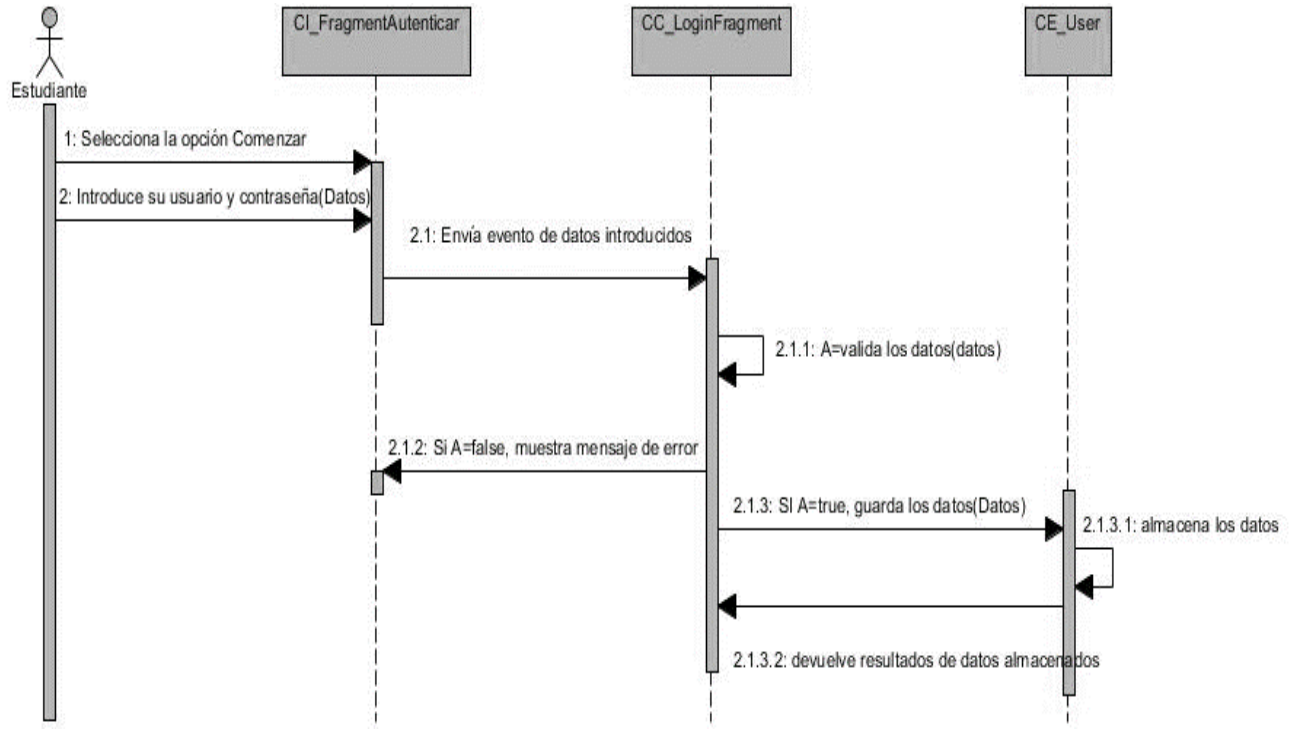


Figura # 4: Diagrama de secuencia del diseño Autenticar Usuario

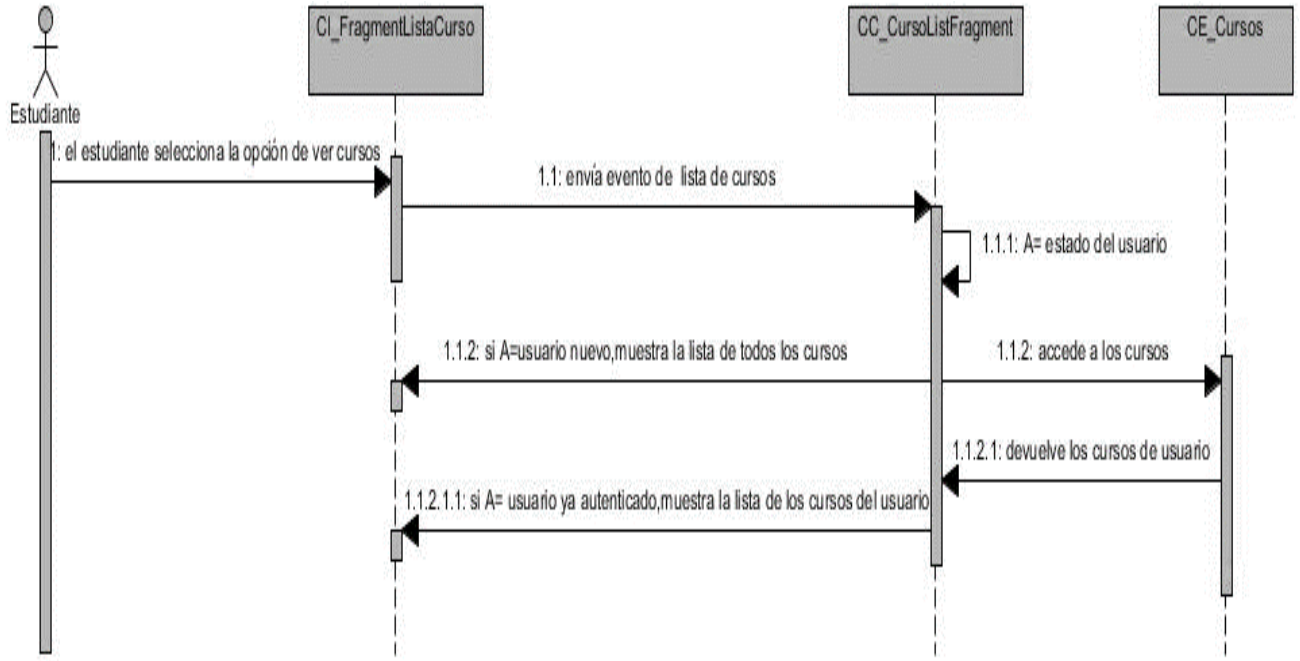


Figura # 5: Diagrama de secuencia del diseño Visualizar Cursos

2.7.3. Diagrama de despliegue

El **Diagrama de Despliegue** es un tipo de diagrama del Lenguaje Unificado de Modelado que se utiliza para modelar la disposición física de los artefactos software en nodos (40).

En otras palabras, modela la arquitectura en tiempo de ejecución de un sistema. Esto muestra la configuración de los elementos de *hardware* (nodos) y muestra cómo los elementos y artefactos del software se trazan en esos nodos (41).

Nodo: un Nodo es un elemento de hardware o software y se diagrama en forma de una caja en tres dimensiones (41).

En la Figura # 6 se muestra como se despliega la aplicación móvil una vez completada su implementación, así como los elementos de *hardware* que se necesitan para su funcionamiento.

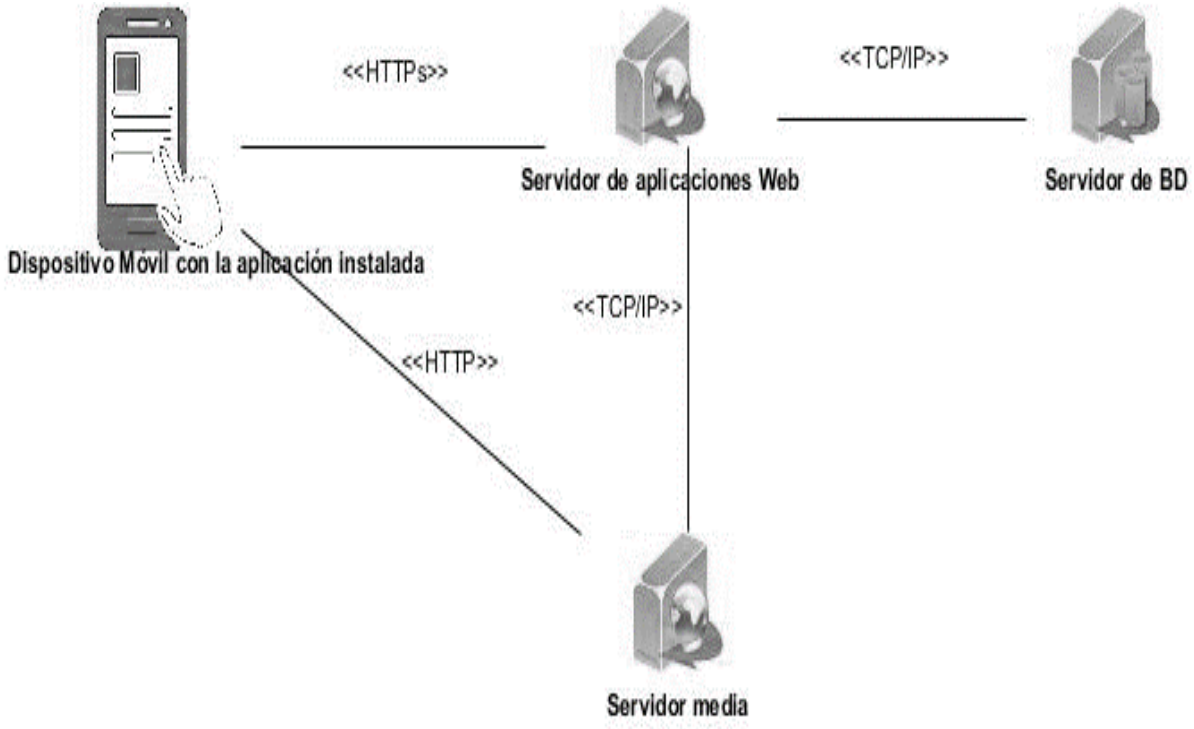


Figura # 6: Diagrama de despliegue

2.8. Conclusiones del capítulo

Después de realizado el diseño de la propuesta de solución se concluye que:

- La realización del modelo de dominio posibilitó definir los conceptos que intervienen en la implementación de la propuesta de solución.
- Se especificaron los requisitos funcionales y no funcionales, obteniéndose una idea general de las funcionalidades a desarrollar en la aplicación.
- A partir de los requisitos funcionales identificados, se realizaron las correspondientes historias de usuario, obteniendo un orden para guiar el proceso de desarrollo de los mismos, en las iteraciones planificadas.
- Se describieron los patrones de diseño y la arquitectura que sirvieron de guía para la ejecución de la solución.
- Se realizaron los diagramas de clases del diseño lo cual permitió facilitar el proceso de implementación de las funcionalidades.

Capítulo 3: Implementación y Pruebas

En este capítulo se expone el proceso de implementación de los elementos identificados durante la realización del diseño. Se evidencian los resultados de las pruebas que se realizan a la aplicación para comprobar la correcta ejecución de las funcionalidades implementadas, permitiendo de esta manera que los componentes desarrollados cumplan con los requisitos establecidos.

3.1 Modelo de implementación

El modelo de implementación tiene como objetivos transformar su modelo en código ejecutable y realizar un nivel básico de las pruebas. El propósito principal de este flujo de trabajo es desarrollar la arquitectura y el sistema como un todo. Describe también la organización de los componentes según los mecanismos de estructuración y modularización disponibles en el entorno de desarrollo, el lenguaje de programación utilizado, y la dependencia entre componentes. Como parte del modelo de implementación se obtienen los diagramas de componentes (42).

3.1.1 Diagrama de componentes

Los diagramas de componentes (DC) tienen como objetivo modelar el sistema o subsistema que se implementara tal cual es. Un diagrama de componente se estructura en varios elementos. Normalmente los diagramas de Componentes contienen: componentes, interfaces, relaciones de dependencia, generalización, asociación y realización, Paquetes o subsistemas. Un componente es una parte física de un sistema (modulo, base de datos, programa ejecutable). Un componente es la materialización de una o más clases, porque una abstracción con atributos y métodos pueden ser implementados en los componentes. En un DC, un componente se representa con un rectángulo en el que se escribe su nombre y en él se muestran dos pequeños rectángulos al lado izquierdo. Los componentes se pueden agrupar en paquetes, así como los objetos en clases, además puede haber entre ellos relaciones de dependencia como: generalización, asociación y agregación (43).

En la figura #7 se muestran todos los componentes que conforman el proyecto. Los componentes son de diferentes tipos, los componentes .xml son las vistas del sistema que se conforman haciendo uso de las clases .java que se encuentran dentro del paquete controlador. En el paquete Modelo se evidencian las clases que se encargan de la interacción con los datos. El paquete *event* almacena la clase que controla las acciones o eventos que se ejecutan dentro de la aplicación. El paquete servicios REST está compuesto

por las clases que controlan la interacción de la aplicación con los servicios web implementados en la plataforma. En el paquete *utils* se encuentran las clases que se utilizan para la generación de los métodos genéricos del SDK.

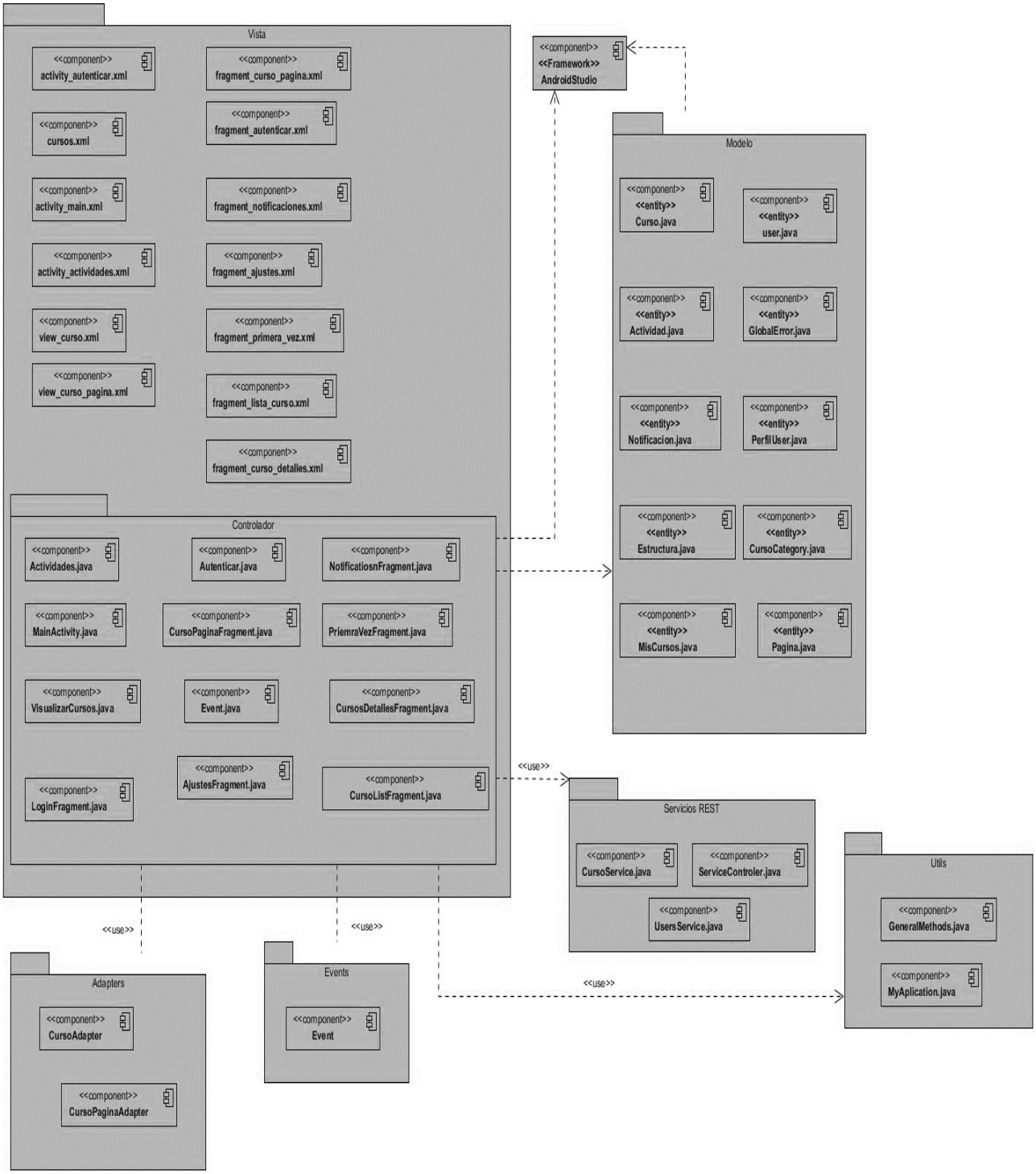


Figura # 7: Diagrama de Componentes

3.1.2 Descripción de las librerías utilizadas

La aplicación no accede a los datos desde una base de datos como lo hacen las aplicaciones web, sino que hace uso de una serie de servicios web implementados en la plataforma para el intercambio de datos con la aplicación. Se hace necesario explicar cuál fue la librería utilizada por la aplicación para hacer las peticiones a través de estos servicios.

Retrofit es un cliente REST para Android y Java, desarrollado por Square. Permite hacer peticiones GET, POST, PUT, PATCH, DELETE y HEAD y descargar datos en formato JSON o XML de manera sencilla. Una vez descargado se parsean en un POJO (representación del Json o XML) para poder tratar los datos en la aplicación (44).

Descripción de las peticiones (44).

- GET: el método GET se utiliza para recuperar los recursos. Antes de hacer la petición hay que determinar cuál es el recurso que vamos a manejar y el tipo de representación que vamos a utilizar. Se utilizó para obtener los cursos, notificaciones, tareas y recursos de la plataforma.
- POST: el método POST se utiliza para crear recursos.
- PUT: el método PUT se utiliza para actualizar (modificar) recursos, o para crearlos si el recurso no existiese previamente.
- DELETE para eliminar un recurso del servidor.

Características de Retrofit (45):

- Aísla el manejador de peticiones en una interfaz
- Usa anotaciones para definir los componentes de la API (verbo, parámetros, cuerpo, cabeceras, etc.)
- Nos permite enviar las peticiones de manera asíncrona
- Es capaz de integrar múltiples conversores para JSON y XML como Gson y Simple XML.

Retrofit se utilizó para el desarrollo de la aplicación como librería de para Android y Java que actúa como cliente HTTP. El uso de esta librería doto la implementación de un mecanismo fácil de usar para comunicar la aplicación Android con los servicios REST implementados en la plataforma. Un ejemplo del uso de esta librería se muestra en la

siguiente imagen que ilustra la forma en que la aplicación hace el llamado a los servicios y accede a los datos.

```
public interface CursoService {


    @GET("/es/api_services/categories")
    Call<CursoCategory[]> listaDeCategoriasCursos();

    @GET("pkt_course.tb_category/parent_id/{parent_id}")
    Call<CursoCategory[]> sublistaDeCategoriasCursos(@Path("parent_id") int parent_id);

    @GET("/es/api_services/courses_by_category")
    Call<Curso[]> listaCursusByCategoria(@Query("id") int category_id);

    @GET("/es/api_services/getCourses")
    Call<Curso[]> listaCursusByUser(@Query("user") String user);

    @GET("pkt_course.tb_structure/course_id/{course_id}/?by=parent_id&by=id")
    Call<Estructura[]> listaEstructuraByCurso(@Path("course_id") int course_id);

    @GET("pkt_resource.tb_page/structure_id/{structure_id}")
    Call<Pagina> listaPaginaByEstructura(@Path("structure_id") int structure_id);

    @GET("/es/api_services/getCoursePages")
    Call<Pagina[]> listaPaginasByEstructura(@Query("courseId") int courseId);


    @GET("/es/api_services/register_in_course")
    Call<Boolean> registrarEnCurso(@Query("course_id") int course_id,@Query("user_id") int user_id);

    @GET("/es/api_services/unregister_in_course")
    Call<Boolean> unregistrarEnCurso(@Query("course_id") int course_id,@Query("user_id") int user_id);


}
```

Imagen#1: Uso de la librería Retrofit.

3.2 Diseño de caso de prueba

El diseño de casos de prueba consiste en probar el sistema, incluyendo los datos de entrada y los resultados esperados. Los casos de pruebas se derivan de las historias de usuario y su objetivo fundamental es encontrar la mayor cantidad de defectos en las funcionalidades implementadas. De esta forma se demuestra que el sistema satisface las necesidades del cliente.

Para realizar un correcto diseño de los casos de pruebas se utilizan diferentes técnicas. para el diseño de los casos de prueba que se muestran a continuación se hizo uso de la técnica de Partición en clases de equivalencia. Esta técnica consiste en dividir los posibles valores en valores de entrada (“*input values*”) y valores de salida (“*output values*”) (46). El rango de valores definido se agrupa en clases de equivalencia para las cuales se aplican las siguientes reglas (46):

- Todos los valores para los cuales se espera que el programa tenga un comportamiento común se agrupan en una clase de equivalencia (CE)
- Las clases de equivalencia no pueden superponerse y no pueden presentar ningún salto (discontinuidad)
- Las clases de equivalencia pueden consistir en un rango de valores (por ejemplo, 0<x<10) o en un valor aislado (por ejemplo, V = Verdadero y F = Falso)

A continuación, se presenta el diseño del caso de prueba perteneciente a la historia de usuario Autenticar Usuario y Visualizar cursos, el resto de los casos de prueba se encuentran en el Anexo # 4.

Tabla#3: Diseño de casos de prueba. Autenticar Usuario

Descripción general: Permitir que el usuario se autentique.					
Condiciones de ejecución: el usuario debe de estar conectado a una red inalámbrica para realizar esta acción.					
Nombre de la sección: SC1 Autenticar Usuario					
Escenario	**Descripción**	**Usuario**	**Contraseña**	**Respuesta del sistema**	**Flujo central**
EC 1.1 Opción Comenzar	Selecciona la opción Comenzar en la vista inicial del sistema			El sistema mostrará los siguientes campos: -Usuario: campo de texto obligatorio donde se debe poner el usuario en letras minúsculas. -Contraseña: campo de texto obligatorio donde se debe poner la contraseña definida por el usuario.	Comenzar/ Autenticar
EC 1.2 Opción de	Selecciona la opción de Iniciar sesión	V	V	Una vez el usuario haya introducido sus datos y estos sean correctos el	Comenzar/ Autenticar/

Iniciar sesión				sistema muestra la lista de los cursos. Ver CP: ”Visualizar Cursos”.	Iniciar Sesión
EC 1.3 Datos Incorrectos	Se introducen datos incorrectos	V	I	El sistema muestra un mensaje de error ”Los datos no son correctos ”	Comenzar/ Autenticar/ Datos incorrectos
		I	V		
EC 1.4 Campos vacíos	No se completan los campos obligatorios	V	I	El sistema no realiza la autenticación y muestra mensaje de error ”Campos vacíos”	Comenzar/ Autenticar/ Campos vacíos
		I	V		

Tabla#4: Diseño de casos de prueba. Visualizar Cursos

Descripción general: Permitir que el usuario visualice todos los cursos de la plataforma en los cuales se puede matricular, así como aquellos en los que ya este matriculado.			
Condiciones de ejecución: para poder ver el listado de los cursos, el usuario tiene que estar conectado a una red inalámbrica.			
Nombre de la sección: SC1 Visualizar cursos.			
Escenarios	Descripción	Respuesta del sistema	Flujo central
EC1.1 Opción Mis cursos	El usuario selecciona en el menú inferior la opción de ver los cursos.	El sistema muestra al usuario en un menú lateral una lista de las categorías de los cursos y acceder desde esta a los cursos. De los cursos de muestra: • Nombre • Imagen del curso	MisCursos/Categoría1/cursos

3.3 Pruebas de software

Las pruebas son aquellas que, durante la ejecución de los procesos principales de la aplicación, hace uso de dos procesos de soporte para probar el producto. Entre éstos se encuentran los procesos de Validación y de Verificación (47).

El proceso de Validación tiene como objetivo determinar si los requisitos y el sistema final cumplen los objetivos para los que se construyó el producto (47).

El proceso de Verificación intenta determinar si los productos software de una actividad se ajustan a los requisitos o a las condiciones impuestas en actividades anteriores (47).

Las pruebas realizadas al producto de software permiten medir la calidad e integridad del mismo, pues posibilitan a los desarrolladores identificar, documentar y corregir un conjunto de no conformidades antes de que el producto sea liberado. Los resultados de las pruebas garantizan que el producto final funcione de acuerdo a los fines para los cuales fue diseñado.

3.3.1 Niveles de prueba

Los niveles de prueba son diferentes formas de verificar y validar un producto de software. A continuación, se distinguen los siguientes niveles de prueba empleados para la identificación de los errores en la aplicación realizada.

Pruebas de Sistema: este tipo de pruebas son ejecutadas idealmente por un equipo de pruebas ajeno al equipo de desarrollo, una buena práctica en este punto corresponde a la tercerización de esta responsabilidad. La obligación de este equipo, consiste en la ejecución de actividades de prueba en donde se debe verificar que la funcionalidad total de un sistema fue implementada de acuerdo a los documentos de especificación definidos en el proyecto. Los casos de prueba a diseñar en este nivel de pruebas, deben cubrir los aspectos funcionales y no funcionales del sistema (48).

Este nivel de prueba se realizó en tres iteraciones, en las cuales se realizaban los casos de pruebas correspondientes a cada requisito funcional del sistema que se implementaba.

Pruebas de Aceptación: las pruebas de aceptación es un proceso realizado por terceros agentes que emiten un certificado de calidad sobre el sistema objeto de prueba. Se designa un personal que sea parte de los procesos de negocio, para la ejecución de pruebas de

aceptación. Es recomendable, que los usuarios finales que participen en este proceso, sean independientes al personal que apoyó el proceso de desarrollo (48).

Este nivel de prueba se realizó de igual manera que el anterior, en tres iteraciones. Para evaluar el nivel de aceptación de la aplicación, se realizaron reuniones con el cliente para verificar que los requisitos funcionales implementados cumplieran con las exigencias del mismo. Las evidencias de estas pruebas se reúnen en el documento de aceptación final que firma el cliente una vez culminado el desarrollo de la aplicación.

3.3.2 Métodos de prueba

Los métodos de pruebas constituyen diferentes técnicas, las cuales se ejecutan en el momento más apropiado en el desarrollo de software. El primer momento es donde hay que probar su código, donde es el mismo desarrollador el que desempeña este papel, aunque puede ser asistido por un grupo de personas ajenas al proyecto. Un segundo momento es cuando en producto ya está terminado y juega un papel fundamental el probador (tester en inglés), el cual es el encargado de eliminar el conflicto de interés que surge cuando la persona que creo el producto realiza las pruebas (49).

Pruebas de Caja negra: las pruebas de caja negra son aquellas en las cuales el elemento es estudiado, desde el punto de vista de las entradas que recibe y las salidas o respuestas que produce, sin tener en cuenta su funcionamiento interno. Se da respuesta a la pregunta, ¿Qué es lo que hace?, pero sin dar importancia a cómo lo hace (50).

3.3.4 Resultados obtenidos

Se realizaron las pruebas utilizando el método de caja negra apoyándose en los diseños de casos de prueba de cada una de las funcionalidades definidas, según el nivel de prueba Sistema. En estos diseños de casos de prueba se describen los escenarios que posee el requisito y se evalúa la respuesta del sistema cuando se realiza una determinada acción en el mismo. Un ejemplo de entradas y salidas se ve representado en la acción de llenar un formulario de autenticación donde el usuario entra un conjunto de datos, el sistema evalúa si los datos son correctos y otorga los permisos al usuario o muestra un mensaje de error en caso de haber datos incorrectos o espacios vacíos.

Como resultado de las pruebas de Sistema realizadas por iteraciones, se identificaron en una primera iteración, un total de seis no conformidades de tipo Alta, correspondientes a errores en el código, cuatro no conformidades de tipo medio, correspondientes a errores en

las vistas y tres no conformidades de tipo bajo, correspondientes a errores ortográficos en los botones de las vistas. En una segunda iteración de las pruebas se detectaron un total de nueve no conformidades, cuatro de tipo alto, tres de tipo medio y dos de tipo bajo. En una tercera iteración no se detectaron no conformidades, lográndose que el sistema cumpla satisfactoriamente con los requerimientos definidos en el levantamiento de información. La relación anteriormente mencionada se evidencia en la siguiente tabla.

Tabla # 5: Resultado de las pruebas por iteraciones

Iteraciones	Cantidad de casos de pruebas	No conformidades detectadas			
		Alta	Media	Baja	Total
1	8	6	4	3	13
2	8	4	3	2	9
3	8	0	0	0	0

Las pruebas de aceptación se realizaron en tres iteraciones. Se obtuvieron en la primera iteración un total de cinco no conformidades de tipo alta, correspondientes a errores de código, cuatro de tipo media, correspondientes a errores de interfaz y dos errores de tipo bajo, correspondientes a errores de ortografía. En la segunda iteración se detectaron tres no conformidades de tipo alta, dos de tipo media y una de tipo baja. En la tercera iteración no se detectaron no conformidades lográndose la satisfacción del cliente con la solución informática. Las relaciones antes mencionadas se evidencian en la siguiente tabla.

Tabla # 6: Resultado de las pruebas de aceptación por iteraciones

Iteraciones	Cantidad de casos de pruebas	No conformidades detectadas			
		Alta	Media	Baja	Total
1	8	5	4	2	11

2	8	3	2	1	6
3	8	0	0	0	0

3.4 Conclusiones del capítulo

Una vez realizadas las fases de implementación y pruebas se concluye que:

- Se implementaron todas las funcionalidades previstas haciendo uso de las herramientas y tecnologías anteriormente seleccionadas.
- Se generó el diagrama de componentes para un mejor entendimiento de los elementos que componen el sistema.
- Se realizaron los diseños de casos de pruebas correspondientes a cada una de las historias de usuarios.
- Se realizaron las pruebas de aceptación en tres iteraciones haciendo uso de los casos de pruebas generados, verificándose el funcionamiento de los requisitos implementados.
- Se comprobó con el desarrollo de tres iteraciones de pruebas de sistema, el correcto funcionamiento de las funcionalidades implementadas, detectándose un total de 12 no conformidades a las que se dio solución.

Conclusiones Generales

- Las aplicaciones móviles existentes de las plataformas educativas estudiadas están diseñadas a la medida para sus respectivas plataformas, teniendo en cuenta los requerimientos de las misma.
- El diseño de la solución propuesta permite visualizar los cursos y los contenidos de los mismo en los dispositivos móviles con sistema operativo Android.
- La solución propuesta constituye una aplicación para los dispositivos móviles, con versiones del sistema operativo Android por encima de la versión 4.0, contribuyendo a la aplicación del *m-learning* en la Plataforma Educativa XAUCE ZERA 2.0.
- La aplicación de las pruebas de Aceptación y de Sistema permitieron asegurar la satisfacción del cliente y que el sistema cumpla satisfactoriamente con los requerimientos definidos en el levantamiento de información.

Recomendaciones

- Desarrollar una nueva versión que le permita al usuario interactuar con los cursos de la Plataforma Educativa XAUCE ZERA 2.0 de forma *off-line*.
- Desarrollar soluciones similares para otros sistemas operativos que son utilizados en los dispositivos móviles.

Referencias Bibliográficas

1. Ventajas del uso de las TIC en el proceso de enseñanza-aprendizaje. Soto, Carlos Ferro. 29, España : EDUTEC, 2009.

2. Dispositivos Móviles y Multimedia. Tardáguila, César. 2009.

3. Area, Manuel. E-Learning: enseñar y aprender en espacios. 2016.

4. Tendencias actuales en el uso de dispositivos móviles en educación. Valero, Carmen Cantillo y Redondo, Margarita Roura. 147, 2012.

5. Juventud Rebelde. Etecsa: Planes actuales y proyecciones. Juventud Rebelde. 10 de Mayo del 2017, 2017.

6. Vidal Ledo, María J, Gavilondo Mariño, Xaily y Rodríguez, Alfredo. Scielo Educ Med Super. Scielo Educ Med Super. [En línea] julio de 2015. [Citado el: 24 de noviembre de 2016.] http://scielo.sld.cu/scielo.php?pid=S0864-21412015000300024&script=sci_arttext&tlng=pt. 3.

7. Guerrero, Moreno. M-learning. 2011.

8. Baz Alonso, Arturo, Ferreira Artime, Irene y Álvarez Rodríguez, María. Dispositivos móviles. 2011.

9. Desarrollo de aplicaciones sobre Android. Vanegas, Carlos Alberto. 2, 2012, Vol. 9.

10. Google. android.com. android.com. [En línea] 2016. [Citado el: 12 de junio de 2017.] https://www.android.com/.

11. Delía, Lisandro, Galdamez, Nicoláz y Pesado, Patricia. Un Análisis Experimental de Tipo de Aplicaciones para Dispositivos Móviles. Universidad Nacional de La Plata. Argentina : s.n., 2012.

12. Padrón, Sandy Nuñez. Ficha XAUCE ZERA 2.0. La Habana : s.n., 2016.

13. Aguledo, Mónica María. Programa de Integración de Tecnologías a la Docencia. 2011.

14. www.moodle.org. www.moodle.org. [En línea] [Citado el: 27 de junio de 2017.] http://www.moodle.org.

15. Valenzuela Zambrano, B., y Pérez Villalobos. unisabana.edu.co. unisabana.edu.co. [En línea] 15 de febrero de 2013. [Citado el: 6 de diciembre de 2016.] http://educacionyeducadores.unisabana.edu.co/index.php/eye/article/view/2000/3074.

16. Montaño, Enrique Castillo. CLIENTE MODDLE PARA ANDROID. Cádiz : s.n., 2014.

17. Castellanos, José González. Desarrollo de una aplicación para móviles para la Plataforma Educativa ZERA. La habana : s.n., 2013.

18. Dianelys Pérez González, Sandy Nuñez Padrón. SCORMVIEW: VISOR DE PAQUETES SCORM PARA XAUCEMOVIL. La Habana : s.n., 2015.

19. Tomás Gironés, Jesús. El gran libro de Android. 2011.

20. Oracle. netbeans.org. netbeans.org. [En línea] Oracle, 2017. [Citado el: 15 de enero de 2017.] https://netbeans.org/index_es.html.

21. colectivo de desarrolladores. uptodown. uptodown.com. [En línea] 14 de junio de 2014. [Citado el: 12 de enero de 2017.] https://android-studio.uptodown.com/windows.

22. AS. androidstudiofaqs.com. androidstudiofaqs.com. [En línea] 8 de diciembre de 2016. [Citado el: 22 de junio de 2016.] https://androidstudiofaqs.com/conceptos/ventajas-desventajas-android-studio.

23. Sylvain Hébuterne - Sébastien Pérochon. Android: Guía de desarrollo de aplicaciones para Smartphones y Tabletas (2a .. Android: Guía de desarrollo de aplicaciones para Smartphones y Tabletas (2a .. [En línea] octubre de 2014. [Citado el: 20 de noviembre de 2016.] https://books.google.com.cu/books?hl=es&lr=&id=DnQkGjh2H3MC&oi=fnd&pg=PA15&dq=ventajas+de+las+aplicaciones+android&ots=G_1xeLYAZU&sig=mrvZ-B-VD4cOblcNGKf0W-SJsmE&redir_esc=y#v=onepage&q=ventajas%20de%20las%20aplicaciones%20android&f=false.

24. Navarro Marset, Rafael. Modelado, diseño e implementación de servicios web. 2014.

25. Sylvain Hébuterne - Sébastien Pérochon. Android: Guía de desarrollo de aplicaciones para Smartphones y Tabletas (2a .. Android: Guía de desarrollo de aplicaciones para Smartphones y Tabletas (2a .. [En línea] octubre de 2014. [Citado el: 20 de noviembre de

2016.] https://books.google.com.cu/books?hl=es&lr=&id=DnQkGjh2H3MC&oi=fnd&pg=PA15&dq=ventajas+de+las+aplicaciones+android&ots=G_1xeLYAZU&sig=mrvZ-B-VD4cOblcNGKf0W-SJsmE&redir_esc=y#v=onepage&q=ventajas%20de%20las%20aplicaciones%20android&f=false.

26. Esquiva Rodríguez, Alejandro. geekytheory.com. geekytheory.com. [En línea] 2015. [Citado el: 14 de junio de 2017.] https://geekytheory.com/json-i-que-es-y-para-que-sirve-json/.

27. Targetware Informática S.A.C. . sofware.com.ar. sofware.com.ar. [En línea] 2007-2017. [Citado el: 21 de enero de 2017.] http://www.software.com.ar/p/visual-paradigm-para-uml.

28. Fernández, Oscar Belmonte. Introducción al lenguaje de programación Java.Una guía básica. 2011.

29. json.org. www.json.org. www.json.org. [En línea] [Citado el: 27 de junio de 2017.] http://www.json.org/json-es.html.

30. Propuesta de metodología de desarrollo de software para objetos virtuales de aprendizaje. Castrillón, Eucario Parra. 34, Colombia : s.n., 2011.

31. Gallego, Manuel Trigas. GESTION DE PROYECTOS INFORMÁTICOS. 2012.

32. Sánchez, Tamara Rodríguez. Metodología de desarrollo para la Actividad productiva de la UCI. La Habana : s.n., 2015.

33. Sommerville, Ian. Ingeniería de software. Séptima edición, Ian Sommerville. Ingeniería de software. 2009.

34. Izaurralde, María Paula. Caracterización de Especificación de Requerimientos en entornos Ágiles: Historias de Usuario. Córdoba : s.n., 2013.

35. Ingeniería de software. Un enfoque práctico. [aut. libro] Roger S. Pressman. Ingeniería de software. Un enfoque práctico.

36. Tutorialspoint. Angularjs- MVC Architecture. [En línea] 2016. [Citado el: 27 de febrero de 2017.] http://www.tutorialspoint.com/angularjs/angularjs_mvc_architecture.htm.

37. Lou, T. A comparison of Android Native App Architecture. 2016.

38. Gloria L Giraldo, Juan F Acevedo. Una Ontología para la presentación de conceptos de diseño de software. 2011.

39. Arletys González Madera, Reysel Pérez Aviléz. Componente para la integracion del Juez en Línea Caribeño y la plataforma educativa ZERA 2.0. La Habana : s.n., 2016.

40. Xavier Ferré Grau, María Isabel Sánchez Segura. Desarrollo Orientado a objetos con UML. 2012.

41. Sparx Systems. Sparx Systems Pty Ltd. Sparx Systems Pty Ltd. [En línea] 17 de noviembre de 2016. [Citado el: 23 de marzo de 2017.] http://www.sparxsystems.com.ar/index.html.

42. Cortes, Olson De Jesús Henao. DISEÑO, ELABORACIÓN, PRUEBAS E IMPLEMENTACIÓN PARA DESARROLLAR APLICACIONES . 2013.

43. Cruz , Victor Favio. slideshare.net. slideshare.net. [En línea] 7 de abril de 2017. [Citado el: 4 de mayo de 2017.] https://es.slideshare.net/uitron/diagrama-de-componentes-7551535.

44. Aramburu, Juan Maria. beeva.com. beeva.com. [En línea] 2016. [Citado el: 11 de mayo de 2017.] https://www.beeva.com/beeva-view/desarrollo/retrofit-una-libreria-para-desarrollo-android-y-java/.

45. www.hermosaprogramacion.com. www.hermosaprogramacion.com. [En línea] noviembre de 2016. [Citado el: 14 de junio de 2017.] http://www.hermosaprogramacion.com/2016/10/retrofit-android-app-medica/.

46. Cusmai, Carlos R. Tecnicas de diseño de casos de pruebas. 2014.

47. Usaola, Dr. Macario Polo. Mantenimiento Avanzado de Sistemas de Información. Pruebas del Software. 2011.

48. S, Javier Zapata. wordpress.com. wordpress.com. [En línea] enero de 2013. [Citado el: 29 de marzo de 2017.] https://pruebasdelsoftware.wordpress.com/.

49. Rocha, Mario A Moreno. es.slideshare.net. es.slideshare.net. [En línea] 22 de mayo de 2015. [Citado el: 29 de marzo de 2017.] https://es.slideshare.net/sirpeto/5-mtodos-de-prueba-de-software.

50. StudentPc. es.slideshare.net. es.slideshare.net. [En línea] 10 de marzo de 2015. [Citado el: 29 de marzo de 2017.]

51. IZAQUITA, MARIA EUGENIA ROJAS. AGILIZANDO LO ÁGIL. Bogota : s.n., 2011.

52. Balaguera, Yohn Daniel Amaya. Metodologías ágiles en el desarrollo de aplicaciones para dispositivos móviles.Estado actual. 2013.

53. Pressman, Roges S. Ingeniería de Software. Un efoque práctico, sexta edición. 2011.

54. Entorno personal de aprendizaje móvil (M-PLE). Pérez, Lidia Sabater. 4, La Habana : s.n., 2016, Vol. 5.