Universidad de las Ciencias Informáticas

Facultad 5

GECMABOT: Aplicación para la gestión del conocimiento de la Matemática Discreta en sistema operativo Android

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

Autores:

Alejandro Gómez Rivas

Ernesto Alfonso Caballero

Tutores:

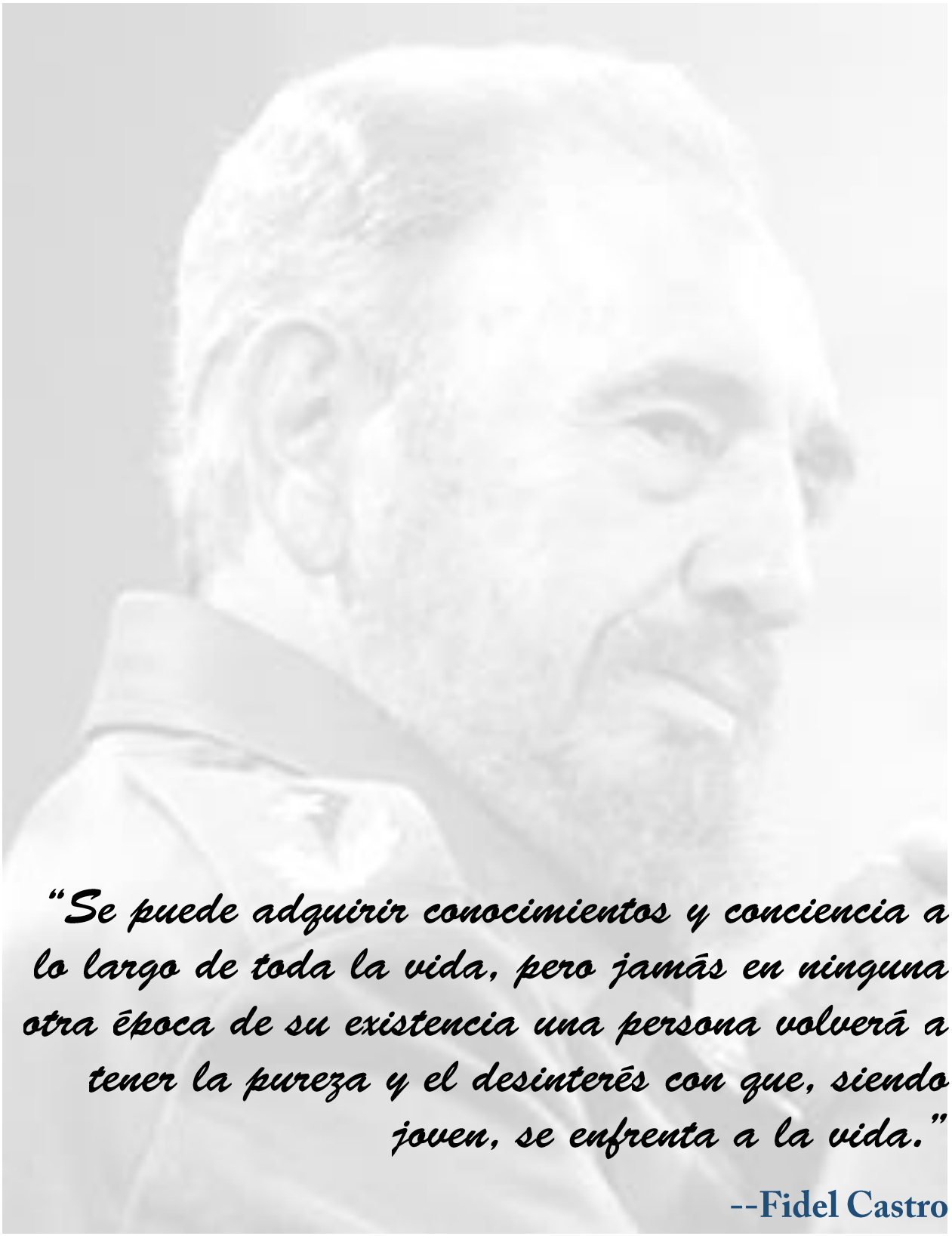
Msc. Yidian Y. Castellanos Sabarí

Drc. Antonio Cedeño Pozo

"Año 58 de la Revolución"

La Habana, Cuba

Junio 2016



"Se puede adquirir conocimientos y conciencia a lo largo de toda la vida, pero jamás en ninguna otra época de su existencia una persona volverá a tener la pureza y el desinterés con que, siendo joven, se enfrenta a la vida."

--Fidel Castro

Declaración de autoría

Declaro ser autor de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

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

_____

Firma del Autor: Ernesto Alfonso Caballero

_____

Firma del Tutor: Msc. Yidian Yosbel Castellanos Sabarí

_____

Firma del Tutor: DrC. Antonio Cedeño Pozo

Declaración de autoría

Declaro ser autor de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

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

_____

Firma del Autor: Alejandro Gómez Rivas

_____

Firma del Tutor: Msc. Yidian Yosbel Castellanos Sabarí

_____

Firma del Tutor: DrC. Antonio Cedeño Pozo

Datos de contacto

Tutor: Msc. Yidian Yosbel Castellanos Sabarí.

Edad: 29

Ciudadanía: cubana

Institución: Universidad de las Ciencias Informáticas (UCI)

Título: Master en Ciencias Matemáticas en el 2015

Categoría Docente: Asistente

E-mail: yycastellanos@uci.cu

Tutor: Dr.C. Antonio Cedeño Pozo.

Edad: 31

Ciudadanía: cubana

Institución: Universidad de las Ciencias Informáticas (UCI)

Título: Doctor en Ciencias Técnicas (Informáticas) en el 2015

Categoría Docente: Instructor

E-mail: acedeno@uci.cu

De Ernesto:

A mis padres, que fueron la mejor escuela, los mejores amigos.

A mi Abuelo, que fue mi cómplice, mi mejor amigo, mi modelo a seguir.

A nuestros tutores, en especial al Msc. Yidian, que fue (y será) más que un tutor un amigo.

A mi tío Arturo, que fue el motivo por el cual me incliné a esta carrera y fue otro tutor más de este trabajo.

A mi hermano Ale, que fue el mejor compañero de tesis.

A mi novia Yeni, que llegó con los Rollin' en el momento oportuno.

A mis hermanos de estos 5 años en la UCI, del grupo 5502, de los apartamentos 66102 y 91102.

A mis amigos en general, los que ayudaron un poco y los que compartieron noches de trova y rock n' roll en el Café.

Al tribunal que fue testigo de la evolución de este Trabajo de Diploma e hizo oportunas correcciones y recomendaciones.

De Alejandro:

A mis padres, especialmente a mi madre por ser la persona más importante en mi vida.

A mi primo por ser mi guía y siempre estar ahí cuando más lo necesito.

A toda mi familia por su apoyo incondicional.

A nuestros tutores por su ayuda y comprensión.

A mi buen amigo y compañero de tesis.

De Ernesto:

Quisiera agradecer en primer lugar a todos los que hicieron posible esta tesis. A Jorge “El Hacker”, mi tío Arturo y a Alberto Marturelo que me brindaron su ayuda en momentos críticos, me ayudaron siempre que lo necesité y sus consejos fueron muy importantes. A mi tutor Yidian que a pesar de nuestras pequeñas peleas fue un gran tutor y un gran amigo, espero no haberte defraudado y sabes que puedes contar conmigo para siempre. A mi familia, en especial a mis padres por su esfuerzo en estos 5 años, por ser un ejemplo de unidad y por brindarme siempre un hogar lleno de felicidad y armonía. A los amigos de estos 5 años en la uci, que más que amigos son parte de la familia. A los compañeros de apartamento, por aguantar mi reguero, mi bulla en las mañanas y por ayudarme siempre con sus comentarios y críticas. A los compañeros de aula por elegirme en su momento como presidente y por cumplir conmigo aun cuando en ocasiones no cumplí con ellos. A todos los amigos del café, los que no están y los que quedan aún. A mis hermanos de la vida Jorge Alfonso Fuentes (el de sangre), Javier Incera, Roger Bárbaro y Noslen los 3 mosqueteros. Por ultimo al amor de mi vida, a Yenisey, por llegar en el momento justo y demostrarme que el amor existe y que se puede ser tan feliz y tan dichoso solo con amar.

De Alejandro:

Quisiera agradecer primer lugar a mi mamá por ser la persona más importante en mi vida, por apoyarme en todas mis locuras, confiar en mi cuando nadie lo haría. A mi primo por ser más que un hermano, un padre, ayudarme y guiarme en las decisiones difíciles de la vida, por ser mi orgullo, mi inspiración y ejemplo a seguir, sin él no hubiera tenido el valor de empezar esta larga carrera por convertirme en Ingeniero. A mi abuela, por empeño y dedicación todos estos años. A mi segunda madre por sus enseñanzas y paciencia desde muy pequeño, A mis primos, a mis tíos, a mi padre, en fin, a toda mi familia, sencillamente sin ellos nada de esto hubiese sido posible. Agradecer a mi tutor y amigo Yidian por toda su ayuda y comprensión. A mis amigos de la Bati-cueva, a Dimitri de la Luz del Topo (Pin), Abel, el Fiera, Haniel, el Sugar, Hector, Juan, por sus regaños que me hicieron esforzarme cada día más. A todos mis amigos, los del barrio y los de aquí, por siempre confiar en mí.

Resumen

Los profesores con un adecuado seguimiento del estado docente de sus estudiantes, pueden tomar acciones en aras de obtener buenos resultados. Con este propósito fue desarrollada GECMA, una solución web para el control docente en la asignatura Matemática Discreta I, con el uso de la Gestión del Conocimiento. GECMA presenta limitaciones debido a que no se dispone todo el tiempo ni en todos los lugares de conexión a la red. Es por ello que esta investigación pretende solventar el problema de cómo contribuir a perfeccionar el control docente que se realiza a través de GECMA, mediante la gestión de datos desde ambientes desconectados.

Plantean los autores como objetivo el desarrollo de una solución informática, para el sistema operativo Android, que permita el empleo de las funcionalidades de GECMA en un ambiente desconectado, denominada GECMABOT. El desarrollo de la solución fue guiado por la metodología ágil OpenUP y se utilizaron para su confección las herramientas Android Studio versión 1.5 mediante el empleo de la Android SDK versión 23. Para la validación de la solución se realizaron pruebas de aceptación y de rendimiento, obteniendo con estas, resultados satisfactorios para este tipo de aplicación.

Palabras claves: Android, ambiente desconectado, control docente, gestión del conocimiento, replicación de datos.

Abstract

Teachers with appropriate monitoring of the teaching status of their students, may take action in order to obtain good results. For this purpose was developed GECMA, a web solution for teaching control in the subject Discrete Mathematics I, which makes use of Knowledge Management. GECMA has limitations because it is not available all the time and everywhere connection to the network. That is why this research aims to solve the problem of how to help improve the teaching control is performed through GECMA, by managing data from disconnected environments.

The authors propose as objective the development of a software solution for the Android operating system, which allows the use of the functionalities of GECMA in a disconnected environment, called GECMABOT. The development of the solution was guided by the OpenUP agile methodology and used as tools for build Android Studio version 1.5 using the Android SDK version 23. For the solution validation were performed acceptance testing and performance testing, obtaining with these, satisfactory results for this type of application.

Keywords: Android OS, data replication, knowledge management, offline environment, scholar control.

ÍNDICE

Introducción

En las instituciones educativas, el control docente es una de las fuentes principales con la que se cuenta para conocer el estado de los estudiantes. Con el desarrollo de las Tecnologías de la Información y las Comunicaciones (TIC), el mismo, ha pasado a realizarse de forma informatizada en la mayoría de los centros educacionales de todo el mundo (Islas, 2008). Para ello se han desarrollado diversas aplicaciones con el propósito de recoger la información de cada uno de los estudiantes en las diferentes materias (asistencias/inasistencias, notas en los exámenes, las bonificaciones y premios). En algunas de estas instituciones la información obtenida es utilizada para realizar análisis históricos-estadísticos de estos datos, con vistas de obtener predicciones y comportamientos futuros. Esto permitirá reajustar los planes y modelos de estudio y favorecer al Proceso de Enseñanza Aprendizaje (PEA) (Kanter y otros, 2015).

La Universidad de las Ciencias Informáticas (UCI) es un ejemplo de las instituciones que han impulsado el buen uso de las TIC en favor de potenciar la gestión de la información académica como parte del PEA. En este ámbito se han desarrollado aplicaciones tales como el Sistema de Gestión Universitaria: Akademos, el cual es un sistema informático cuya principal misión es el control de todos los procesos que intervienen en la gestión académica en la UCI. Surge como respuesta a la necesidad de sustentar y dar soporte a toda la labor del personal de secretaría, con la visión de obtener un producto genérico, capaz de ser aplicable y adaptable en cualquier centro que implemente el control docente universitario (Suárez Hernández, 2009). Luego de un estudio de la aplicación se pudo constatar que Akademos es una herramienta para el control docente, pero la misma no gestiona el conocimiento almacenado de forma automática, característica de las aplicaciones insertadas en la Sociedad del Conocimiento. El hecho de que Akademos sea una aplicación web constituye una limitante, ya que en la universidad (específicamente en los salones de clases) no siempre se cuenta con la conexión a la red (imprescindible para el trabajo con este tipo de aplicaciones), lo cual conlleva a que, al no ofrecer un ambiente de trabajo en modo desconectado, se deban registrar los datos en un medio intermediario hasta que se cuente con la conexión necesaria. Esto ha traído consigo que los profesores realicen el control docente desde registros en soporte papel y en ocasiones con el uso de algunas herramientas informáticas construidas de forma empírica o la utilización de hojas de cálculo con el uso de las herramientas ofimáticas.

Teniendo en cuenta lo anterior, como iniciativa de uno de los profesores de la asignatura Matemática Discreta 1 (MD1), el pasado curso se desarrolla GECMA, aplicación informática para el control docente que hace uso de la Gestión del Conocimiento (GC) en el PEA de las Matemáticas Discretas 1 en la UCI. Creada a partir de la necesidad de perfeccionar el modelo de enseñanza de la asignatura, pues constituye una piedra angular en el proceso de formación de los futuros ingenieros. Sin embargo, los resultados docentes que muestran los alumnos al ser evaluados, evidencian el grado de dificultad con que estos se enfrentan a ella. Los autores parten del presupuesto de que el profesor desconoce en qué momento el alumno disminuye la aprehensión del conocimiento y por tanto se torna difícil realizar una atención diferenciada en el momento y con los recursos adecuados. Es por esto que entre las funcionalidades desplegadas en GECMA se logra un nivel más detallado de las evaluaciones y la asistencia, donde el educador puede desglosar una evaluación en objetivos vencidos, generar cortes evaluativos automatizados de los grupos, entre otras funcionalidades, las cuales hoy no son cubiertas por Akademos.

En encuestas desarrolladas a los profesores de la asignatura Matemática Discreta 1 (MD1) se pudo constatar que se han presentado dificultades para el uso de GECMA. Algunas motivadas por estar desarrollada para ser desplegada en ambientes web. Se pudo corroborar que la mayoría de estos profesores, que poseen dispositivos móviles inteligentes y en algunos casos tabletas con sistema operativo Android, consideran factible realizar estos registros desde este tipo de dispositivos.

Si se considera lo antes descrito, para los profesores, el seguimiento sistemático del estado de los estudiantes les permite tomar acciones a tiempo, en aras de la obtención de buenos resultados. La UCI ha apostado por la construcción de herramientas que beneficien este propósito, pero las mismas aún presentan limitaciones debido a que en nuestro país aún no se ha logrado alcanzar un consumo continuo de la información, a pesar de la existencia de gran cantidad de dispositivos móviles. Las posibilidades del uso de este tipo de dispositivos y la ventaja de contar con el sistema operativo Android (que es de código abierto), no ha sido explotada debidamente para brindar aplicaciones que apoyen a las soluciones web existentes en la universidad.

A partir de la situación anterior se plantea el siguiente **problema a resolver:** ¿Cómo contribuir a perfeccionar el control docente que se realiza a través de GECMA?

Para resolver el problema planteado se propone como **objetivo general** de la investigación: Desarrollar una solución informática para el sistema operativo Android que permita el empleo de las actuales funcionalidades de GECMA en la Universidad de las Ciencias Informáticas.

Lo antes expuesto permitió definir como **objeto de estudio:** El proceso de control docente con el empleo de herramientas informáticas. Enmarcado en el **campo de acción** de la investigación: El control docente en la asignatura Matemática Discreta 1 con el empleo de herramientas informáticas para ambientes desconectados.

Para dar cumplimiento al objetivo general se plantean las siguientes **tareas de investigación**:

- Revisión de la bibliografía asociada al desarrollo de sistemas para ambientes desconectados, la gestión del conocimiento y las aplicaciones para plataformas Android.
- Definición de las herramientas a utilizar para la implementación de la solución.
- Levantamiento de requisitos para el diseño e implementación de la solución.
- Implementación de una aplicación para el sistema operativo Android que permita la utilización de las funcionalidades actuales de GECMA.
- Implementación de un Servicio Web que permita el intercambio de datos entre GECMA y la aplicación Android confeccionada.
- Implementación de los mecanismos de sincronización de datos para garantizar la integridad de los datos de GECMA.
- Realización de pruebas a la solución.
- Validación de la solución a partir de la norma ISO/IEC 9126.

Con el cumplimiento de las tareas antes planteadas, se consideran como **posibles resultados**:

- La entrega de una aplicación informática, para el sistema operativo Android, que garantice el empleo de las funcionalidades actuales de GECMA en teléfonos móviles o tabletas.

- La entrega de un servicio web que permita la comunicación y el flujo seguro y confiable de datos entre la aplicación desarrollada y GECMA.
- La elaboración de una base teórica y conceptual, que a partir de diversos artefactos permita la comprensión de la aplicación desarrollada.

A continuación, se detallan los **métodos de investigación científica** empleados en la investigación:

Métodos Teóricos:

Analítico – Sintético: Se realiza un análisis del PEA de la asignatura MD1 y de la aplicación GECMA para determinar las principales funcionalidades que deben ser implementadas en la aplicación móvil. Se analiza además la información disponible en Internet sobre las buenas prácticas para el diseño e implementación de aplicaciones sobre la tecnología Android.

Histórico – Lógico: Se realiza el estudio de las tendencias históricas y actuales en cuanto al PEA de la asignatura MD1, así como de la Gestión del Conocimiento y las tecnologías móviles de sistema Android.

Métodos Empíricos:

Entrevista: Se obtiene, mediante su aplicación, la información necesaria para la obtención de los requisitos de la aplicación, así como, datos de interés sobre las funcionalidades de GECMA que serán desplegadas.

Pruebas: Se detectan y corrigen el máximo número de errores posibles en el producto antes de su entrega al cliente (Pressman, 2010). Consisten en el diseño de casos de prueba, con el empleo de las técnicas de pruebas de software, que arrojen la mayor cantidad de errores en la aplicación.

Búsqueda bibliográfica: Se obtiene la base teórica de la presente investigación mediante el análisis y la lectura de la bibliografía disponible, la cual es referenciada en el presente trabajo de diploma.

La investigación está estructurada de la siguiente forma:

- Introducción.
- Tres Capítulos.
- Conclusiones.
- Bibliografía.
- Anexos.

Capítulo 1 Fundamentación Teórica: en este capítulo se define la base teórica de la presente investigación, haciendo énfasis fundamentalmente las soluciones desarrolladas para ambientes desconectados, la Gestión del Conocimiento en la MD1 y las tecnologías móviles con sistema operativo Android.

Capítulo 2 Características y Diseño del sistema: en este capítulo se realiza una descripción detallada de la solución, donde se definen los requisitos funcionales y se realizan actividades de análisis y diseño.

Capítulo 3 Implementación y Pruebas: en este capítulo se describe la implementación de la solución, se ejecutan los casos de pruebas y se corrigen las no conformidades encontradas para garantizar el buen funcionamiento de la solución.

Capítulo 1. Fundamentación teórica

Introducción

En este capítulo se detalla la base teórica de la investigación desarrollada, a partir de las bondades del empleo de la GC y las tecnologías móviles como herramientas para el perfeccionamiento del control docente que se realiza desde la aplicación GECMA. Explicita un estudio de herramientas que puedan dar solución al objetivo propuesto. Como conclusión se definen las herramientas, espacios de trabajo, lenguajes de programación y metodología a utilizar con el fin de construir la solución informática.

Como antecedente se realizó, por parte de los autores, un estudio de la investigación elaborada por Yandry Gutiérrez (Gutiérrez, 2015) en la cual se exponen criterios sobre la importancia de la asignatura MD1 para los ingenieros en Ciencias Informáticas. Se hace un análisis de algunas situaciones reales que evidencian los bajos resultados obtenidos por los estudiantes de primer año en la UCI. Se detalla el diseño y la implementación de GECMA, la cual da cumplimiento al objetivo propuesto por el investigador, y es la aplicación en la cual se apoyará para gestionar los datos la herramienta resultante de la presente investigación.

1.2 La Gestión del Conocimiento y la Matemática Discreta

Luego de un análisis de diferentes conceptos asociados (Davenport, y otros, 2001) y (López, 2014) se puede concluir que existen dos formas de obtención de conocimiento, comúnmente conocidas como tipos de conocimiento. Por una parte, se encuentra el conocimiento explícito o codificado, que es comúnmente conocido por ser trasmisible y estar determinado por la forma simple de obtenerlo, expresarlo y estructurarlo (López, 2014). En contraposición, el conocimiento tácito tiene un carácter personal, que lo hace difícil de formalizar y comunicar. Este se obtiene de manera inconsciente como resultado de la experiencia y la observación. Los autores Nonaka & Takeuchi definen la GC como la transformación de conocimiento explícito en conocimiento tácito y viceversa, a partir de prácticas de combinación (de explícito a explícito), interiorización (de explícito a tácito), socialización (tácito para tácito) y exteriorización (tácito a explícito) (Nonaka, y otros, 1995).

De acuerdo con el modelo anterior, el objetivo principal de la GC consiste en identificar, capturar, analizar, compartir y enriquecer el conocimiento, tanto explícito como tácito, teniendo en cuenta su contexto, de modo que sea posible su reutilización en la generación de nuevos

conocimientos. Con el auge de las redes sociales, se ha comenzado a reconocer el rol de las personas en la producción de conocimiento. Esto ha provocado un giro en la concepción del conocimiento como objeto, para enfocarlo como un proceso colaborativo en el que intervienen activamente las personas (Katrib Mora y otros, 2013) En el mismo se centraliza en tres elementos, el primero dirigido al desarrollo y aprendizaje continuo de las personas, que tiene en cuenta la medición de capital intelectual, el segundo a la gestión, almacenamiento de la información, el conocimiento y como último elemento la interrelación con las tecnologías.

1.2.1 La Gestión del Conocimiento aplicada al proceso docente en la MD1: GECMA

La competencia matemática es algo complejo, difícil de definir, que exige del estudiante su dominio sobre conocimientos y destrezas, y también, por lo menos, sobre algunas capacidades medibles, a las que Niss denomina «competencias específicas» (Niss, 2003) (Oecd Pisa, 2009). Si se asigna un valor medible al desempeño de estas capacidades se generan datos que constituyen un indicador de aprendizaje para cada estudiante. Luego de un análisis, Gutiérrez, en su investigación, considera que “los profesores de esta universidad (UCI) se apoyan básicamente en el registro de asistencia y evaluaciones para gestionar estos indicadores, siendo los mismos de vital importancia para comprender la situación de los discentes ya que los malos resultados que se obtengan en la asignatura pueden estar influenciados por problemas de asistencia” (Gutiérrez, 2015). En entrevista realizada a los profesores de MD1 estos concuerdan y mantienen que, de existir una detección a tiempo de problemas de aprendizaje y asistencia en un estudiante, el profesor podría tomar una serie de medidas preventivas para lograr la aprensión del conocimiento en el estudiante.

En este marco la GC, insertada dentro de GECMA brinda a los profesores un estudio de referentes anteriores que permiten llegar a una mejor detección de estos estudiantes con problemas y brinda además una serie de remediales. Los datos con los cuales se opera en esta herramienta lo constituyen las puntuaciones de las evaluaciones, los datos identificativos de cada alumno y la asistencia, entre otros. La información que se puede extraer de estos datos depende de la creatividad e inteligencia de cada uno de los actores que interactúen con ella. Por su parte el conocimiento será extraído de un conjunto de informaciones propias de los datos relacionados con cada uno de los estudiantes, sus evaluaciones y los remediales que a cada uno de ellos se le apliquen. De ahí que la GC se hace evidente desde el momento en que

se necesite, en cursos posteriores, ante situaciones similares emitir una respuesta adecuada (Gutiérrez, 2015).

Teniendo en cuenta los bajos resultados académicos que obtienen los estudiantes en la asignatura MD1, los cuales persisten hoy día, se ha podido constatar que aún se requiere una atención diferenciada de los profesores a los estudiantes con problemas, para corregir a tiempo estos resultados. Es por ello que la obtención de los datos es vital para el correcto funcionamiento de la GC en GECMA. Y es que como en cualquier sistema de este tipo, el acceso a información actualizada, fiable y abundante debe ser garantizado en todo momento por los actores del mismo.

1.3 Soluciones informáticas para ambientes desconectados

En ocasiones no es posible contar con conexión a Internet al estar dentro de un avión, en transportes públicos sin señal o situados de forma temporal en espacios abiertos sin cobertura. Esto en si constituye un entorno desconectado.

Un entorno desconectado es aquel en el que un usuario o una aplicación no están conectados constantemente a una fuente de datos. Para solucionar este inconveniente, algunas de las principales empresas que prestan servicio web cuentan con características *offline*. Estas permiten a los usuarios no interrumpir el uso de sus principales aplicaciones por el hecho de estar desconectados, e incluso, acceder a sus últimos documentos sin necesidad de contar con un canal de datos que les comunique con el servidor. En este caso luego de que se abre la conexión, se recuperan los datos y la conexión se cierra. El usuario trabaja con los datos en el navegador y la conexión vuelve a abrirse para actualizar u otras peticiones (Softwarelibre, 2010).

La mayoría de los dispositivos informáticos de hoy en día permiten el trabajo en entornos desconectados. A las tradicionales computadoras, *laptops* y *notebooks* se han sumado otras como los dispositivos móviles, PDA (Por las siglas de *personal digital assistant*) y los sistemas de Posicionamiento Global (GPS por sus siglas en Ingles). En estos entornos, los usuarios móviles son también los usuarios principales.

El entorno desconectado proporciona las siguientes ventajas (Softwarelibre, 2010):

- Las conexiones se utilizan durante el menor tiempo posible, permitiendo que menos conexiones den servicio a más usuarios.
- Mejoran la escalabilidad y el rendimiento de las aplicaciones, maximizando la disponibilidad de conexiones.

Por otra parte, tiene los siguientes inconvenientes (Softwarelibre, 2010):

- Los datos no siempre están actualizados.
- Pueden producirse conflictos de cambios que deben solucionarse.

Es por ello que las tecnologías móviles vienen a resolver el problema de acceso constante a fuentes institucionales de datos en nuestro país. Al existir una conexión intermitente a la red, los mismos pueden llevarse un subconjunto de datos en un equipo desconectado y posteriormente fusionar los cambios con el almacén de datos central.

1.4 Las tecnologías móviles, su inserción y repercusión dentro de la GC

Las tecnologías móviles son un medio de comunicación que ha superado a la telefonía fija, esto se debe a que las redes de telefonía móvil son más fáciles y baratas de desplegar. El uso de las tecnologías móviles entre los habitantes de una población, ayuda a disminuir la brecha digital existente entre cada lugar, ya que muchos usuarios utilizan este medio tecnológico para el desarrollo de sus actividades y por eso se reduce el conjunto de personas que no las utilizan (García Carranco, y otros, 2014).

Los primeros dispositivos móviles disponían simplemente de las funcionalidades básicas de telefonía y mensajes de texto. Poco a poco se han ido añadiendo pantallas de colores, cámaras de fotos, entre otros. En 2004 llegó la posibilidad de videoconferencias. En el año 2005, los teléfonos fueron capaces de reproducir archivos de audio, también, sistemas operativos y conexión a internet, destacando los BlackBerry de la empresa *Research in Motion*. De esta manera, los usuarios empezaron a entender el móvil como una prolongación de sus computadoras en movimiento, así nació el concepto de dispositivo móvil inteligente (Ramirez Acosta, 2010).

Un dispositivo móvil inteligente es un dispositivo electrónico, por lo general conectado a otros dispositivos o redes a través de diferentes protocolos como Bluetooth, NFC, Wi-Fi, 3G, X10,

entre otros, que puede funcionar hasta cierto punto de forma interactiva y autónoma. Los más conocidos y reconocidos internacionalmente son los *smartphones* o teléfonos inteligentes o la mayoría de los dispositivos con sistema operativo integrado, como los *tablets* o tabletas. El campo se ha expandido además a otros equipos como televisores, relojes y automóviles. Este tipo de dispositivos además incorporan funciones como juegos, reproducción de música MP3 y otros formatos, correo electrónico, Servicio de Mensajes Cortos (SMS por sus siglas en ingles), agenda electrónica PDA, fotografía digital y video digital, video llamada, entre otras(Salazar, 2014).

Para el uso de estos dispositivos se han desarrollado varias plataformas, o sistemas operativos móviles. Entre las más usadas se encuentran Android (de Google), iOS (de Apple), Symbian (de Nokia), BlackBerry OS (de BlackBerry) y *Windows Phone* (de Microsoft) (Hyers, 2014).

En la sociedad del conocimiento los avances tecnológicos dan respuesta a las necesidades que plantea esta sociedad. Así, en una sociedad en movimiento surgen las tecnologías móviles para dar respuesta a las necesidades constantes de acceso a la información y de comunicación (Cantillo Valero, 2012). Con la evolución de la Internet, la Web de conjunto con la aparición y el desarrollo de las tecnologías móviles inteligentes, el usuario pasa a tomar un rol fundamental como creador, proveedor y consumidor de la información. Esta es la razón por la cual los autores de la presente investigación se han decantado por desarrollar para este tipo de dispositivos, ya que los mismos pueden sustituir al intermediario entre el aula y GECMA, mediante una aplicación que sea capaz de almacenar datos en el dispositivo y luego que estos datos sean enviados al sistema GECMA.

1.5 El Sistema operativo Android

A nivel internacional Android es un Sistema Operativo (SO) basado en Linux, el cual es de código abierto y provee además de una capa de librerías escritas en C y C++ aparejado a un marco de trabajo para el desarrollo de aplicaciones. Originalmente este marco de trabajo estaba solo basado en Java y posteriormente se liberó para aplicaciones nativas en C, aunque no es recomendable a menos que se necesite utilizar de manera excesiva al microprocesador. Incluye además una suite de aplicaciones iniciales (Firtman, 2013).

Google es su principal impulsor, pues es el que lo mantiene, pero en realidad para hacerlo más abierto y que no sea un sistema operativo manejado por una sola empresa, creó una

organización sin fines de lucro denominada la Alianza Abierta de Dispositivos (*Open Handset Alliance*). Incluye a más de 65 empresas del sector de fabricación de dispositivos móviles, entre los que se encuentran a Google, fabricantes de equipos Motorola, HTC, LG, Samsung, Alcatel; operadores móviles como Telefónica, Movistar, T-Mobile, Sprint, China Mobile y fabricantes de microprocesadores y placas de video Intel, nVidia, Qualcomm, Texas Intruments, Huawei, entre otros. Estas empresas son unas de las pocas que forman la alianza, responsables de mantener a Android como sistema operativo (Firtman, 2013).

El Anexo 1 representa la tabla que contiene la cantidad de unidades vendidas (*Global Smartphone Operating System Shipments*) de los principales Sistemas Operativos (SO) de teléfonos inteligentes como son Android, *Apple iOS*, *Microsoft*, *Blackberry* y de otros, estableciendo una comparativa entre el segundo trimestre del año 2013 (Q2´13) y el segundo trimestre del año 2014 (Q2´14). La segunda parte muestra cómo se encuentran estos SO en el mercado (*Global Smartphone Operating System Marketshare*) para los mismos períodos de tiempo. En la parte final se muestra expresado en por ciento el crecimiento anual de los trimestres antes mencionados (*Total Growth Year over Year*) (Hyers, 2014). Se evidencia del análisis de la tabla anterior que Android domina las ventas del mercado con un 80, 2% en el segundo trimestre del 2013 y con un 84, 6% en el segundo trimestre del 2014. Esto se debe a que se desarrolla de forma abierta, permitiendo acceder tanto al código fuente como a la lista de incidencias, donde se pueden ver problemas aún no resueltos y reportar problemas nuevos, además Android como *software* libre permite la distribución de las mejoras hechas en su código fuente de forma gratuita, con lo que se logra que todos los usuarios que lo utilicen salgan beneficiados (Hyers, 2014).

1.6 La sincronización como parte vital del trabajo en modo desconectado

Una de las características que más gusta a los usuarios de Android es la enorme capacidad de sincronización de sus dispositivos. Gracias a este servicio se pueden administrar y gestionar los perfiles y cuentas de varios servicios además de las aplicaciones, sin necesidad de ir una a una. La sincronización de datos o *data synchronization* es el proceso del establecimiento de consistencia entre datos en fuentes remotas y de la continua armonización de los datos en el tiempo (Alegsa, 2008).

La sincronización sin conexión tiene varias ventajas:

- Mejora la capacidad de respuesta de las aplicaciones almacenando en caché datos de servidor de forma local en el dispositivo.
- Hace que las aplicaciones sean resistentes ante una conectividad de red intermitente.
- Permite crear y modificar los datos incluso con poca o ninguna conectividad.
- Sincroniza datos en varios dispositivos.
- Detecta los conflictos cuando dos dispositivos modifican el mismo registro.

De ahí que una de las condiciones principales para el trabajo en modo desconectado en dispositivos móviles es garantizar una buena y correcta sincronización de datos. Los objetivos a cumplir para realizar la sincronización de datos en sistema de dispositivos móviles son los siguientes (Buchman, 2002):

- Mantener cada sistema con los datos actualizados: Si se piensa en una red donde sus datos están distribuidos, es indispensable que cada sistema cuente con datos actualizados.
- Reducir la carga sobre la red: El flujo de datos de la red puede ser reducido considerablemente si en lugar de consultar un servidor central, se accede a datos locales sincronizados.
- Datos consistentes: A pesar de que los clientes móviles contienen sus propios datos locales sincronizados, no siempre están conectados a la red. En esos momentos que no se posee conexión, el cliente móvil debe ser capaz de seguir operando sus sistemas con los datos locales obtenidos en la última sincronización.
- Resolver los conflictos surgidos: Cuando en una red existen varios clientes que utilizan los mismos sistemas sobre el mismo conjunto de datos, es posible que existan conflictos al querer sincronizar sus datos con los demás.
- Entregar servicio de calidad: Aplicaciones como voz y video, necesitan ir al mismo tiempo con cada entrega de información. Si las aplicaciones no pueden entregar el mínimo servicio de calidad requerido, las aplicaciones no podrán operar en forma óptima.

Basados en cómo será la sincronización, se puede clasificar en dos categorías (Buchman, 2002): Sincronización Lenta (Slow sync) y Sincronización Rápida (Fast sync).

- Sincronización lenta: Durante la sincronización lenta, también conocida como sincronización completa, todas las entradas de un dispositivo son enviadas hacia el otro dispositivo. Una vez enviadas todas las entradas, se descartan todas las entradas duplicadas y se detectan las nuevas o modificadas para finalmente actualizar los restantes dispositivos. Este es un proceso largo y costoso en recursos, pero a veces necesario según el dispositivo.

- Sincronización rápida: La sincronización rápida involucra el envío de los datos que sólo han sido modificados. Las entradas modificadas sobre el servidor pueden ser rastreadas en intervalos de tiempo. Similarmente en el cliente, se puede utilizar un indicador para saber si la entrada ha sido modificada. Mientras se realiza la sincronización, todas las entradas en el cliente son enviadas y una vez terminada la sincronización exitosa, se eliminan esos indicadores de las entradas. Este mecanismo puede provocar la existencia de conflictos durante la sincronización. Algunos conflictos pueden ser resueltos automáticamente combinando ambas entradas y otros requieren que sus conflictos sean resueltos manualmente.

La sincronización de datos sobre dispositivos móviles no es tarea sencilla. Al encontrarse sus datos dispersos sobre los dispositivos móviles y la pérdida de conexión que sufren los mismos, es probable que se produzcan colisiones y conflictos entre los datos a sincronizar.

Hay varios tipos de conflicto de datos (Borda, 2012):

- Conflictos de inserción: Cuando se inserta un mismo dato en dos aplicaciones locales. Cuando la primera aplicación intenta confirmar la inserción en la aplicación central, lo realiza sin inconvenientes. El problema lo posee la segunda aplicación al intentar sincronizar sus datos, ya que el dato insertado también fue insertado por la otra aplicación y genera problemas para determinar que dato es el válido.

- Conflictos de actualización: Es similar al conflicto de inserción. Dos sistemas modifican el mismo dato en sus aplicaciones locales y al intentar sincronizar sus datos con la aplicación central se producen los conflictos.

- Conflictos de actualización vs. eliminación: Un problema frecuente en aplicaciones sobre dispositivos móviles es resolver las modificaciones de un dato que ha sido eliminada por una aplicación, pero modificada por otro.

En la Figura 1 se muestra un ejemplo del surgimiento de un conflicto de actualización entre los datos de las aplicaciones. En un inicio, las aplicaciones A y B obtienen el mismo dato desde la componente central. Luego, ambas modifican ese dato y la aplicación B lo confirma a la componente central. Al momento de que la aplicación A obtiene los datos modificados en la componente central, se encuentra que el mismo dato modificado localmente fue confirmado por la aplicación B antes.

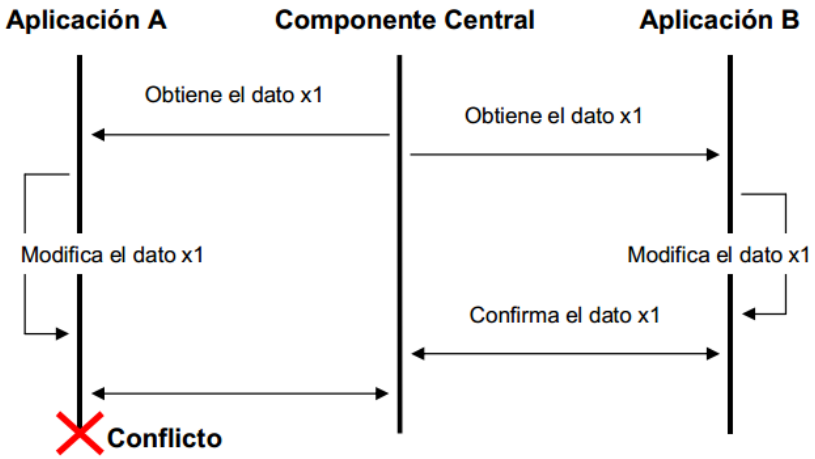


Figura 1: Conflicto en los datos (Borda, 2012)

De existir estos conflictos en las aplicaciones locales, el motor de sincronización cliente debe proveer una forma de resolver estos conflictos surgidos. Para mayor seguridad y a fin de evitar datos corruptos, el motor de sincronización cliente no debe permitir confirmar datos al servidor que estén en conflictos. Todos los conflictos se deben solucionar en la aplicación local antes de confirmar los datos a la componente central.

1.7 Estudio de aplicaciones análogas

Luego de un estudio bibliográfico para determinar aplicaciones que pudieran dar solución al problema científico, se encontraron las siguientes aplicaciones análogas:

Cuaderno del profesor Lite (CDP): Es una aplicación diseñada para dispositivos móviles como smartphones, note-tablet y tablet, que usen el sistema operativo Android. El objeto de la

aplicación es contribuir en las tareas habituales del profesorado, agilizándolas y optimizándolas. La aplicación está diseñada para sustituir a los tradicionales "cuadernos del profesor" en formato papel, que se emplean para gestionar los datos de cursos y alumnado. CDP fue publicada en septiembre de 2012 bajo la supervisión de un equipo de docentes de Andalucía, Castilla la Mancha y Extremadura (España). En su posterior rediseño y evolución se han tenido en cuenta las necesidades y peticiones que los usuarios han ido remitiendo a lo largo de los meses.

La aplicación dispone de una versión “lite” gratuita 100% funcional, limitada a la creación y gestión de 2 grupos, y una versión “profesional” con capacidad de gestionar grupos ilimitados. La licencia profesional es por tiempo ilimitado, permite la instalación simultánea en varios terminales de un mismo usuario y tiene un precio de 4€ (Labedroid, 2015).

Attendance: Con esta aplicación se puede realizar un seguimiento de la asistencia para cualquier tipo de evento. Permite:

-Hacer un seguimiento de la asistencia de las sesiones de entrenamiento deportivo.

-Seguimiento de los estudiantes (asistencias, evaluaciones).

-Seguimiento de la participación en las reuniones regulares de trabajo.

También se pueden definir eventos, asignar a los participantes al evento (de sus contactos de la cuenta de Google) y hace el seguimiento correspondiente. Exporta los datos Excel (CSV e), hojas de cálculo de Google y LibreOffice.

Teacher Pal: Es una aplicación sencilla, fácil de usar y rica en funciones que permite la administración de clases. Los profesores pueden organizar clases y administrar estudiantes fácilmente. La misma está disponible en iOS y Windows 8, es una aplicación de software propietario y solamente se encuentra soportada para los idiomas inglés y árabe.

GradeBook: Es una herramienta potente e intuitiva para el manejo de la clase. Permite al profesor realizar evaluaciones, pasar asistencia y comprobar el desempeño de los estudiantes. Es una aplicación que fue diseñada con recomendaciones de educadores profesionales La aplicación requiere iOS 8.0 o posterior. Compatible con iPhone, iPad e iPod touch y como software propietario, tiene un valor de $349.00 (Lombardo, 2015).

Luego del estudio de las aplicaciones análogas, se puede comprobar que ninguna de ellas satisface el problema planteado ya que en ningún caso se permite el envío de datos a terceros. En la esta investigación esto es primordial, ya que la misma se trata de realizar una mejora a la aplicación GECMA, dotando a esta de una aplicación que solucione el problema del trabajo en modo desconectado y la gestión confiable de datos. Aunque se reconoce a la aplicación Cuaderno del Profesor como la más completa, la misma presenta deficiencias debido a que, para gestionar más de dos grupos, se debe pagar una licencia. El resto de las aplicaciones es de código privativo también con altos precios de licencia.

Es por esto que, dada la importancia que tiene GECMA, se plantea la creación de un nuevo sistema para dispositivos móviles con sistema operativo Android que permita el trabajo con la aplicación GECMA en un ambiente desconectado. A través de esta se debe posibilitar el envío de datos de manera segura y confiable.

1.8 Tecnologías utilizadas

Como parte del diseño de la arquitectura de software se definen herramientas y tecnologías que son utilizadas para garantizar el óptimo resultado por parte del equipo de desarrollo. A continuación, se definen el conjunto de tecnologías y herramientas necesarias.

1.8.1 Metodología de desarrollo

Uno de los principales aspectos de un proyecto de software es el proceso de desarrollo que se va seguir para la implementación de un producto. Dentro de este proceso es necesario especificar las actividades del ciclo de vida del software, esto viene dado por la metodología que sea usada en el mismo. Para el correcto desarrollo del proyecto de construcción de la solución informática es necesario el uso de una metodología que sea capaz de conducir al equipo de trabajo, con el objetivo de asegurar la calidad y eficacia durante el proceso de concepción del producto (Pinzón y otros, 2006).

1.8.1.1 Metodologías Ágiles vs Metodologías Tradicionales

Las metodologías de desarrollo de software se clasifican en:

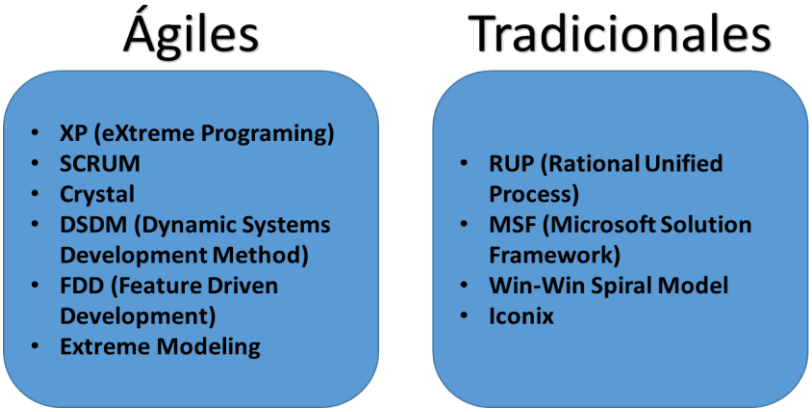


Figura 2. Metodologías Ágiles y Tradicionales

La elección respecto a la utilización o no de una determinada metodología depende principalmente del grado de predictibilidad que se desee tener en el desarrollo. En el desarrollo de la presente solución informática se selecciona el uso de una metodología de desarrollo ágil, debido a que se cuenta con un equipo formado por dos desarrolladores para realizar las tareas. Se tiene en cuenta de que estas permiten simplicidad de sus reglas y prácticas, son flexibles ante los cambios que puedan surgir en el trascurso de la implementación. Esto se evidencia en la Figura 3, donde luego de realizar una comparación de ambos modelos se puede observar que las metodologías ágiles se adaptan mejor a equipos con poco personal.

Metodología Ágil	Metodología No Ágil (Tradicional)
Pocos artefactos	Más artefactos
Pocos roles	Más roles
No existe un contrato tradicional o al menos es bastante flexible	Existe un contrato prefijado
El cliente es parte del equipo de desarrollo (además in-situ)	El cliente interactúa con el equipo de desarrollo mediante reuniones
Grupos pequeños (< 10 integrantes) y trabajando en el mismo sitio	Grupos grandes
Menos énfasis en la arquitectura	La arquitectura es esencial

Figura 3. Comparación entre metodologías Ágiles y Tradicionales.

1.8.1.2 OpenUp

En la UCI, desde hace algunos años se utiliza en algunos departamentos OpenUp para el desarrollo de software, teniendo en cuenta que la misma está dirigida a la gestión y el desarrollo de proyectos de software basados en un desarrollo iterativo, ágil e incremental. Esta metodología además permite la entrega de artefactos comúnmente utilizados en metodologías tradicionales, que logran una mejor comprensión de la aplicación y que pueden servir como documentación para el proceso de mantenimiento de la misma. Es por esto que luego de un consenso entre los autores y el cliente se define esta metodología como la más idónea para el desarrollo de la solución propuesta. La metodología OpenUP define cuatro fases (OpenUp, 2012):

1. Inicio: En esta fase, las necesidades de cada participante del proyecto son tomadas en cuenta y son plasmadas en los objetivos del proyecto. Se identifican y refinan los requisitos, se define el límite del proyecto y se realiza una estimación inicial del coste, los riesgos y las actividades asociadas al proyecto.

2. Elaboración: En esta fase se realizan tareas de análisis del dominio y definición de la arquitectura del sistema. Se debe elaborar un plan de proyecto, estableciendo unos requisitos y arquitectura estables. Al final de la fase se debe tener una definición clara y precisa de los casos de uso, actores, la arquitectura del sistema y un prototipo ejecutable.

3. Construcción: En esta fase se desarrollan de manera incremental todos componentes y funcionalidades del sistema que falten por implementar. Los mismos son probados e integrados a la solución final. Al concluir el producto el mismo debe estar listo para su entrega al cliente.

4. Transición: Esta fase corresponde a la introducción del producto en la comunidad de usuarios. En la misma se llevan a cabo las sub-fases de pruebas beta, pilotaje y capacitación. En función a la respuesta obtenida por los usuarios, puede ser necesario realizar cambios en las entregas finales o implementar alguna funcionalidad más solicitada por la mayoría, siempre que no se aleje de los costes y requisitos definidos al inicio del proyecto.

En la Figura 4 se puede observar cada una de estas fases que componen el proceso de entrega, en la misma se puede observar que cada fase corresponde en si un proceso iterativo en la cual se debe dar cumplimiento a los hitos previamente definidos los cuales pueden ser almacenados, valorados y utilizados como futuras referencias en proyectos similares.

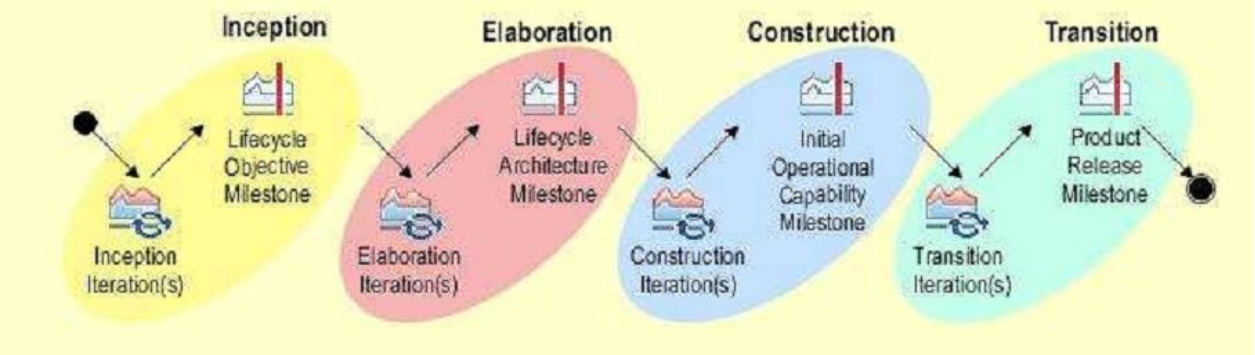


Figura 4: El proceso de entrega de OpenUP

1.8.2 Entorno de Desarrollo Integrado

Un Entorno de Desarrollo Integrado (IDE del inglés *Integrated Development Environment)*, es un programa informático compuesto por un conjunto de herramientas de programación. Los IDE proveen un marco de trabajo para los lenguajes de programación (Gonzáles Duque, 2006).

Android *Studio* 1.5

Es un IDE para la plataforma Android. Se encuentra disponible para desarrolladores de forma gratuita. Es diseñado específicamente para desarrollar soluciones para Android y su uso es recomendado por Google para el desarrollo de este tipo de aplicaciones. (Android, 2014).

Android Studio presenta las siguientes características (Android, 2014):

- Renderización en tiempo real.
- Consola de desarrollador: consejos de optimización, ayuda para la traducción y estadísticas de uso.
- Refactorización específica de Android y arreglos rápidos.
- Herramientas para detectar problemas de rendimiento, usabilidad, compatibilidad de versiones y otros problemas.
- Plantillas para crear diseños comunes de Android y otros componentes.

Estas características constituyen la razón por la cual es la herramienta seleccionada para el desarrollo de la solución propuesta.

1.8.3 Lenguaje Unificado de Modelado

Lenguaje Unificado de Modelado (UML, por sus siglas en inglés *Unified Modeling Language*) es el lenguaje de modelado de sistemas de software, utilizado para visualizar, especificar, construir y documentar un sistema. Es aplicable en el proceso de construcción de un software para dar soporte a una metodología de desarrollo de software, ayudando a especificar, visualizar y documentar esquemas de sistemas de software, incluyendo su estructura y diseño, de una manera que cumpla con todos los requisitos del mismo. El uso de herramientas basadas en UML permite analizar los requisitos y diseñar una solución que sea satisfactoria (UML, 2014).

1.8.4 Herramienta CASE

Las herramientas de Ingeniería de Software Asistida por Computadora (CASE, acrónimo de Computer Aided Software Enfineering) son diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software, reduciendo el costo de las mismas en términos de tiempo y dinero. Estas herramientas pueden ayudar en todos los aspectos del ciclo de vida del desarrollo del software, en tareas como el proceso de realizar un diseño del proyecto, cálculo de costos, implementación de parte del código automáticamente con el diseño dado, compilación automática, documentación o detección de errores, entre otras (López, 2014).

Visual Paradigm v8.0

Visual Paradigm es una herramienta CASE. La misma propicia un conjunto de ayudas para el desarrollo de programas informáticos, desde la planificación, el análisis y el diseño, hasta la generación del código fuente de los programas y la documentación. Ha sido concebida para soportar el ciclo de vida completo del proceso de desarrollo del software a través de la representación de varios tipos de diagramas (López, 2014).

1.8.5 Lenguajes de Programación

Un lenguaje de programación es un lenguaje formal diseñado para expresar procesos que pueden ser llevados a cabo por computadoras. Son utilizados para crear programas que controlen el comportamiento físico y lógico de una máquina, para expresar algoritmos con precisión, o como modo de comunicación humana. Está formado por un conjunto de símbolos y reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones. Al proceso por el cual se escribe, se prueba, se depura, se compila (de ser

necesario) y se mantiene el código fuente de un programa informático, se le llama programación (Sebesta, y otros, 2012).

En Android existen dos formas de manejar la implementación, ellas son mediante Código Dalvik o código nativo.

Código Dalvik

Dalvik es la máquina virtual que utiliza la plataforma para dispositivos móviles Android. La Máquina Virtual Dalvik (DVM, por sus siglas en inglés *Dalvik Virtual Machine*) sacrifica la portabilidad que caracteriza a Java para poder crear aplicaciones con un mejor rendimiento y menor consumo de energía. Estas dos características son extremadamente importantes en dispositivos móviles, debido a que la capacidad de las baterías en estos dispositivos es limitada. Esta máquina virtual se ejecuta por encima de un kernel Linux, el cual le permite, entre otras cosas, delegar las tareas relacionadas con la gestión de hilos y memoria a bajo nivel (Android (2), 2014).

DVM permite ejecutar aplicaciones programadas en Java, siendo este un lenguaje de programación de propósito general, concurrente, orientado a objetos que fue diseñado específicamente para tener tan pocas dependencias de implementación como fuera posible. Su intención es permitir que los desarrolladores de aplicaciones escriban el programa una vez y lo ejecuten en cualquier dispositivo. Las aplicaciones de Java son generalmente compiladas a *bytecode* (clase Java) que puede ejecutarse en cualquier Máquina Virtual Java (JVM, por sus siglas en inglés *Java Virtual Machine*) sin importar la arquitectura de la computadora subyacente (Byous, 2003).

Código Nativo

Es una forma de código de la programación de computadora que se configura para funcionar con el uso de un procesador especificado. La estructura exacta del código nativo se usa para responder a las instrucciones que son enviadas por el procesador (Sheng, 2000).

Siguiendo este concepto se encuentra Interfaz nativa de Java (JNI por sus siglas en ingles), siendo este un framework de programación que permite que un programa escrito en Java ejecutado en la JVM pueda interactuar con programas escritos en otros lenguajes como C, C++ y Ensamblador. El JNI se usa para escribir métodos nativos que permitan solventar situaciones

en las que una aplicación no puede ser enteramente escrita en Java, como por ejemplo, en el caso de que la biblioteca estándar de clases no proporcione soporte para funcionalidades dependientes de la plataforma (Sheng, 2000).

Teniendo en cuenta los conceptos anteriores, los autores del presente trabajo de investigación hacen uso para el desarrollo de la solución del Código Dalvik mediante el trabajo con DVM, teniendo en cuenta que se encuentra optimizada para necesitar poca memoria, aprovechando así el mejor rendimiento de los móviles, además DVM se encuentra distribuida como software libre.

1.8.6 Android SDK v23

El Kit de Desarrollo de Software (SDK, siglas en inglés de *Software Development Kit*): Android SDK es un conjunto de herramientas de desarrollo de software que le permite al programador crear aplicaciones para un sistema concreto, en este caso para Android. Este paquete o kit de desarrollo incluye las API (siglas en inglés de *Application Programing Interface,* Interfaz de Programación de Aplicaciones) y herramientas necesarias para desarrollar las aplicaciones, mediante el uso de Java como lenguaje de programación (Android, 2014).

1.8.7 Gestor de Bases de Datos

Es un conjunto de programas que permiten crear y mantener una BBDD (BBDD), los cuales aseguran su integridad, confidencialidad y seguridad. La plataforma Android proporciona tres herramientas principales para el almacenamiento y consulta de datos estructurados: Bases de Datos, Content *Providers y ORMs*.

SQLite:

SQLite es un motor de bases de datos muy popular en la actualidad por ofrecer características tan interesantes como su pequeño tamaño, no necesitar servidor, precisar poca configuración, ser transaccional y por supuesto ser de código libre. SQLite presenta las siguientes características distintivas (SQLite, 2016):

- No requiere el soporte de un servidor, no ejecuta un proceso para administrar la información, si no que implementa un conjunto de librerías encargadas de la gestión.

- No necesita configuración, libera al programador de todo tipo de configuraciones de puertos, tamaños, ubicaciones, entre otros.
- Es de Código Abierto, está disponible al dominio público de los desarrolladores al igual que sus archivos de compilación e instrucciones de escalabilidad.

Android incorpora de serie todas las herramientas necesarias para la creación y gestión de bases de datos SQLite, y entre ellas una completa API para llevar a cabo de manera sencilla todas las tareas necesarias.

Realm:

Para el trabajo con bases de datos en Android, las vías tradicionales o estándares resultan en ocasiones incómodas. Trabajar con SQLite en ocasiones hace el código difícil de entender y la escritura de consultas SQL puede tomar mucho tiempo. El uso de los ORM (Siglas en ingles de Modelo de Objetos Relacionales) parcialmente resuelve este problema, pero una mejor alternativa que se presenta hoy en día es el uso del modelo de datos Realm.

Realm es un modelo de base de datos persistentes (al igual que SQLite) que fue construido para ganar en simplicidad y velocidad en las transacciones en aplicaciones móviles. Debido a su modelo de construcción "*Zero-copy"* Realm es mucho más rápido que otros modelos como ORM y SQLite (ver Figura 5). Está disponible para IOs, OS X y Android permitiendo el intercambio de datos entre estos sistemas con el uso del mismo nivel en los modelos para Java, Objetive-C y Swift. También Realm presenta otras características avanzadas como la construcción de bases de datos encriptadas a través del algoritmo de encriptación estándar AES-256, además de permitir la realización de consultas gráficas y la migración fácil de bases de datos. (Realm, 2014).

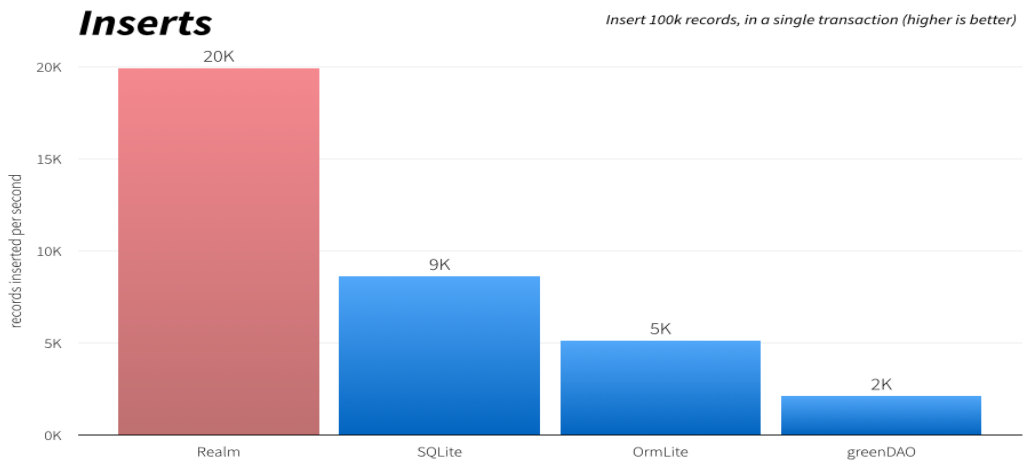


Figura 5: Cantidad de transacciones por segundo de diferentes modelos de bases de datos para dispositivos móviles (Stackoverflow, 2014)

Como ventajas del uso de Realm se puede destacar, además de las antes mencionadas:

- Es una forma rápida de ejecutar consultas y transacciones de manera asíncrona.
- Todas las transacciones son rápidas.
- Ofrece una forma fácil de construir consultas (sólo se invoca un método).
- Permite ejecutar la BBDD desde la memoria RAM del dispositivo.
- Permite la encriptación mediante AES-256 de la BBDD.

Entre las desventajas que se pueden citar:

- Se puede perder el archivo de la BBDD al ser borrado del dispositivo.
- No se pueden crear ID's auto-incrementales.
- Las clases del modelo solamente pueden tener métodos *get* y *set*. No permiten incluir métodos polimórficos como el *equals()* o el *toString()*.

Luego de un estudio se pudo observar cómo una sencilla aplicación desarrollada con Realm, aventaja en velocidad y reduce el tamaño de la BBDD de una aplicación que utiliza SQLite. Se decide la utilización de Realm para la construcción de la BBDD, obteniendo de esta forma

velocidad y ahorro de espacio, así como seguridad, ya que Realm permite crear una BBDD encriptadas sin pérdida de velocidad de transacción.

1.8.8 Patrones de diseño para aplicaciones Android

Modelo N-Capas o por Capas

El patrón arquitectónico N-Capas, es recomendable para los entornos de ejecución de las aplicaciones para el SO Android (Android, 2014). A partir de esta definición aparecen dos conceptos importantes: las capas (*Layers*) que se encargan de la distribución lógica de los componentes, y los niveles (*Tiers*) que se refieren a la colocación física de los recursos. La característica principal de este patrón es que cada capa oculta las capas inferiores de las siguientes superioras a esta.

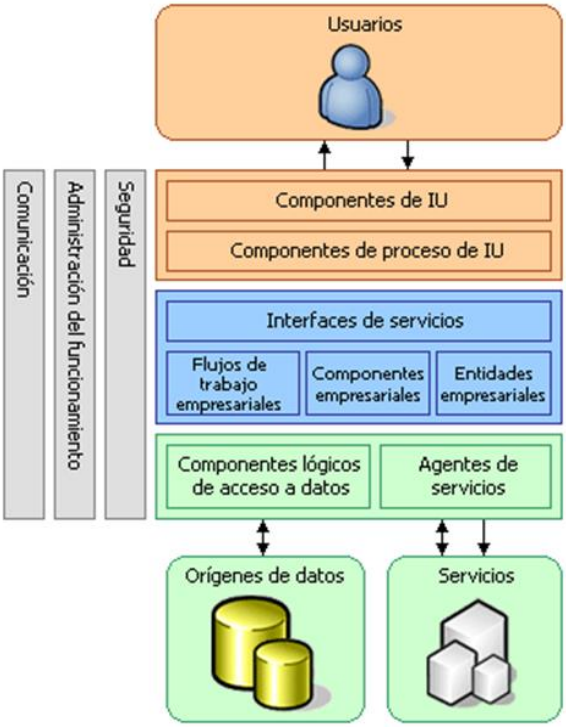


Figura 6. Ejemplo del patrón N-Capas (Peláez, 2009).

Algunas de las ventajas de utilizar este patrón son (Peláez, 2009):

- Mejoras en las posibilidades de mantenimiento, debido a que cada capa es independiente de la otra los cambios o actualizaciones pueden ser realizados sin afectar la aplicación como un todo.
- Escalabilidad, ya que las capas están basadas en diferentes máquinas, el escalamiento de la aplicación hacia afuera es razonablemente sencillo.
- Flexibilidad, como cada capa puede ser manejada y escalada de forma independiente, la flexibilidad se incrementa.
- Disponibilidad, las aplicaciones pueden aprovechar la arquitectura modular de los sistemas habilitados mediante el uso de componentes que escalan fácilmente lo que incrementa la disponibilidad.

Modelo Vista Controlador (MVC)

El MVC es una especificación del patrón N-Capas aplicado solamente a 3 capas. Es un patrón de arquitectura de las aplicaciones software que separa la lógica de negocio de la interfaz de usuario. El mismo separa la lógica de negocio de la interfaz de usuario lo cual genera un incremento de la flexibilidad y la reutilización en las aplicaciones. Este fue descrito por primera vez en 1979 para Smalltalk (Pabón Maestras, 2008).

Flujo de control

1. El usuario realiza una acción en la interfaz.

2. El controlador trata el evento de entrada.

    - Previamente se ha registrado.

3. El controlador notifica al modelo la acción del usuario, lo que puede implicar un cambio del estado del modelo (si no es una mera consulta).

4. Se genera una nueva vista. La vista toma los datos del modelo.

    - El modelo no tiene conocimiento directo de la vista.

5. La interfaz de usuario espera otra interacción del usuario, que comenzará otro nuevo ciclo.

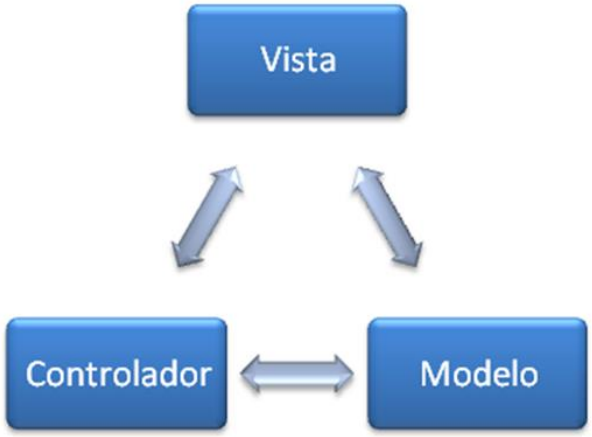


Figura 7. Patrón Modelo-Vista-Controlador ((Pabón Maestras, 2008)

Las ventajas de utilizar el MVC son (Imaginanet, 2015):

- La separación del Modelo de la Vista.
- Es mucho más sencillo agregar múltiples representaciones de los mismos datos o información.
- Facilita agregar nuevos tipos de datos según sea requerido por la aplicación ya que son independientes del funcionamiento de las otras capas.
- Crea independencia de funcionamiento.
- Facilita el mantenimiento en caso de errores.
- Ofrece maneras más sencillas para probar el correcto funcionamiento del sistema.
- Permite el escalamiento de la aplicación en caso de ser requerido.

Modelo Vista Presentador (MVP)

El MVP es una especificación o personalización del MVC. El principal objetivo de aplicar este patrón, es desacoplar la vista, del modelo. Es el patrón utilizado actualmente por las aplicaciones Android ya que en este tipo de aplicaciones las vistas (formadas por las Actividades y los Fragmentos) se comunican a través del presentador, o la capa de lógica de Negocio, donde son interpretados, dado que las vistas son archivos xml y las clases son

generalmente en algún lenguaje orientado a objetos (casi siempre Java) (Sánchez Dieguez, 2015).

- Modelos: Agrupa la información de la aplicación, es decir, los datos.
- Vista: Para Android, las vistas son los *activities*, *fragments* y *adapters*. Estos se encargan de mostrar la información y manejar los eventos de entrada y salida del usuario.
- Presentador: Es una clase que se encarga de manejar la comunicación entre los modelos y la vista

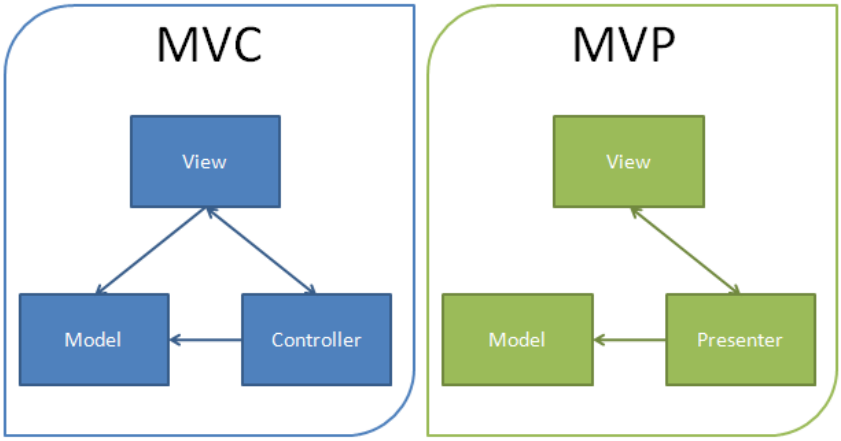


Figura 8. Diferencias entre los patrones MVC y MVP (Sánchez Dieguez, 2015).

Se puede decir que el patrón MVP es una mejora del patrón MVC basado en tres características (Imaginanet, 2015):

- La vista no conoce el modelo.
- El presentador es independiente de la tecnología de interfaz de usuario.
- La vista y el presentador son testeables puesto que está basada en un contrato.

1.8.9 Herramientas para la construcción de la API de Sincronización

Para la construcción de la API que permite la sincronización de los datos entre la aplicación para dispositivos móviles y el sistema web GECMA se utilizan las siguientes herramientas y lenguajes.

1.8.9.1 Framework para la implementación de la API

Un *framework* web provee una infraestructura de programación para aplicaciones, para que el programador pueda concentrarse en escribir código limpio y de fácil mantenimiento sin tener que reinventar la rueda. En resumidas cuentas, eso es lo que hace Django (Holovaty, y otros, 2009).

Django es un *framework* Web que fue desarrollado por un equipo de programadores web de Lawrence, Kansas en el año 2003. Fue liberado en julio de 2005 (Holovaty, y otros, 2009).

Se escoge este *framework* ya que el mismo permite de manera fácil y rápida la creación de una sencilla API mediante la utilización de una API REST y Django como *framework* para su implementación.

1.8.9.2 Lenguaje de programación para la implementación de la API

El *framework* Django utiliza Python como lenguaje de programación. Python es un lenguaje de programación similar a Perl, pero con una sintaxis muy limpia y que favorece un código legible. Se trata de un lenguaje interpretado o de script, con tipado dinámico, fuertemente tipado, multiplataforma y orientado a objetos (Gonzáles Duque, 2006).

Un lenguaje interpretado o de script es aquel que se ejecuta a través de un programa intermedio llamado intérprete, en lugar de compilar el código a lenguaje máquina que pueda comprender y ejecutar directamente una computadora (lenguajes compilados). La ventaja de los lenguajes compilados es que su ejecución es más rápida. Sin embargo, los lenguajes interpretados son más flexibles y más portables.

1.8.9.3 Entorno de desarrollo integrado

PyCharm es un IDE desarrollado por la compañía Jetbrains, está basado en IntelliJ IDEA, el IDE de la misma compañía, pero enfocado hacia Java y la base de Android Studio. Pycharm tiene cientos de funciones que lo puede ver como una herramienta muy pesada, pero que valen la pena ya que ayudan al desarrollo de aplicaciones.

Entre algunas de las ventajas de PyCharm se pueden enunciar:

- El autocompletado, resaltador de sintaxis, herramienta de análisis y refactorización.

- La integración con *frameworks* web como: Django, Flask, Pyramid, Web2Py.
- La inclusión de *Frameworks javascripts*: jQuery, AngularJS.
- Integración con lenguajes de plantillas: Mako, Jinja2, Django Template.

Por sus características, los autores del presente trabajo seleccionan el IDE PyCharm en su versión 5.0.4 como la herramienta a utilizar para el desarrollo de la API de Sincronización.

1.10 Conclusiones parciales

En este capítulo, tras realizar por parte de los autores un estudio, derivaron las siguientes conclusiones parciales:

- La herramienta GECMA es una solución propia de la UCI que integra la gestión de conocimientos al proceso de control docente. Aun así, encuentra limitaciones en cuanto a la adquisición de datos, debido a que esta aplicación está desarrollada sobre la tecnología Web.
- Las tecnologías móviles vienen a resolver el problema de trabajo en un entorno desconectado. Luego del estudio de aplicaciones, para este tipo de tecnologías, que puedan resolver esta problemática se pudo constatar que ninguna soluciona el problema para el entorno donde se encuentra desplegada GECMA.
- Se propone la construcción de una solución propia mediante la utilización de Android Studio 1.5 como IDE de desarrollo, el empleo de Android SDK 23 como kit de desarrollo, el lenguaje de programación Java y el modelo de gestión de datos Realm. El Framework Django en su versión 1.7.8, con el empleo del lenguaje de programación Python en su versión 2.7.11 y el IDE PyCharm en su versión 5.0.4 son utilizados para la construcción de un mecanismo que garantice la sincronización segura de datos. Este proceso es guiado a través de la metodología ágil OpenUP. Para ello se emplea como lenguaje de modelado el UML y se utiliza como herramienta de modelado el Visual Paradigm en su versión 8.

CAPITULO 2: Análisis y diseño de la solución propuesta

Introducción

En el presente capítulo se analiza y diseña la propuesta de solución del sistema a implementar, guiada por la metodología de desarrollo OpenUP. Además, se describen las características que debe cumplir el sistema a través del diseño de los diferentes casos de uso basados en requisitos funcionales y no funcionales.

2.1 Propuesta de solución

Se propone la creación de un sistema que funciona en dispositivos inteligentes con el SO Android a partir de la versión 3.0 y es denominado GECMABOT. Entre las funcionalidades que brinda se encuentra: permitir al profesor gestionar el proceso docente de los grupos de estudiantes a los que imparte clases en su dispositivo móvil y soportar además la sincronización posterior y automatizada de los datos de la aplicación a GECMA. Al ser GECMA una aplicación Web terminada, que no presenta un servicio para la entrada y la obtención de datos, se hace necesaria la comunicación a través de una API de sincronización. Mediante su utilización se inserta la información en la BBDD de GECMA y se proveen los mecanismos necesarios para garantizar la seguridad y confiabilidad del proceso de sincronización.

Para la solución se propone un modelo de arquitectura basado en el uso del patrón arquitectónico N-Capas. La descomposición en capas que se propone es:

1. Capa de presentación: Es la única que interactúa directamente con el usuario presentándole el sistema. Permite el intercambio de información entre ambos. Contiene una interfaz gráfica que posibilita capturar los datos insertados por el cliente, además de mostrarle los resultados de sus peticiones y los estados de la aplicación. Esta capa solo interactúa con la capa de negocio, que es su inferior inmediata.

2. Capa de negocio: Conocida como lógica de negocio. Es la que recibe las peticiones de la capa de presentación y le envía las respuestas tras el proceso. Aquí se realiza la mayor parte del procesamiento de la información del dominio de la aplicación, se ejecutan cálculos sobre los datos de entrada o almacenados. Otra de sus funciones es la de validar los contenidos provenientes de la capa superior y ejecutar los algoritmos específicos del programa. Esta capa interactúa con la capa de presentación, para recibir sus solicitudes y presentarle los resultados. Intercambia con la capa de datos, para

solicitar al Sistema Gestor de Base de Datos (SGBD) almacenar u obtener datos de él; y con la Capa de Comunicación para realizar las peticiones de sincronización y acceso a servicios.

3. Capa de comunicación y herramientas: Es una capa que contiene algunas herramientas que son utilizadas tanto en la capa de negocio como en la capa de acceso a datos. Agrupa las clases que permiten la comunicación con otros sistemas. Contiene un componente propio del sistema operativo Android cuya función principal es manejar el consumo de servicios web y peticiones-respuestas HTTP.

4. Capa de acceso a datos: Es la encargada de intercambiar con la BBDD. Esta capa interactúa con la capa de negocio, recibe sus solicitudes de almacenamiento o recuperación de información, y envía la respuesta a las peticiones.

Con el objetivo de exponer claramente el funcionamiento de la propuesta arquitectónica se presentan los siguientes diagramas que muestran la estructura de la arquitectura, los elementos que la componen y la forma en que interactúan ellos. En la Figura 9 se puede observar el diagrama general de la propuesta arquitectónica. En el Anexo 2 se encuentra el diagrama orientado al dominio de la propuesta arquitectónica.

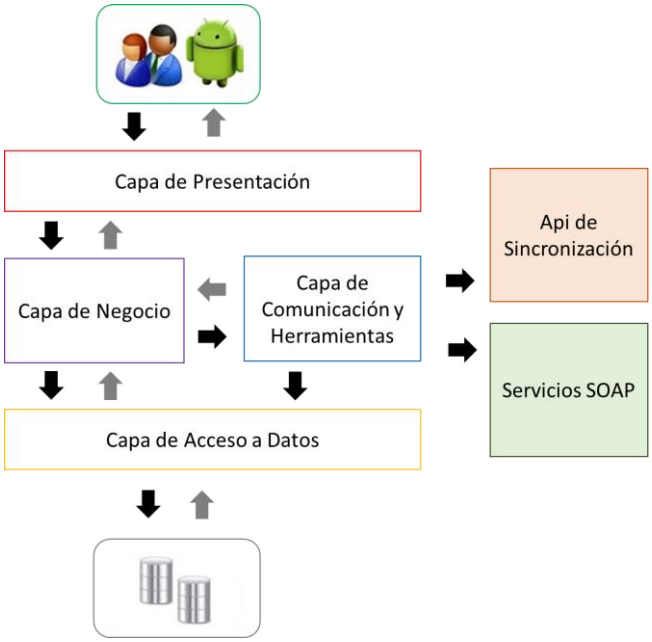


Figura 9: Diagrama General de la propuesta arquitectónica

A continuación, se describen los elementos de cada capa:

1) Capa de presentación:

    a) Componentes de interfaz de usuario

        i) Elementos de formularios: tienen diseños nativos del sistema operativo Android, incluyen botones, etiquetas, selectores, barras de progreso y de puntuación, etc.

        ii) Scripts de Estilos: posibilitan la creación de estilos personalizados para los elementos visuales de la aplicación.

        iii) Elementos multimedia: permiten reproducir audio y visualizar vídeos e imágenes en el dispositivo móvil.

        iv) Elementos de navegación: posibilitan embeber sitios web como parte de las aplicaciones.

        v) Organizadores visuales: son una especie de cajas contenedoras que distribuyen en la interfaz gráfica los restantes componentes de presentación al usuario.

    b) Componentes de proceso de interfaz de usuario:

        i) Actividades Android: representan la capa de presentación de toda aplicación Android, por ejemplo, una pantalla que el usuario ve. Una aplicación para Android puede tener varias actividades y se puede cambiar entre ellas en tiempo de ejecución de la aplicación.

        ii) Fragmentos de Android: Los fragmentos son porciones dentro de una Actividad que pueden ser cambiadas sin necesidad de cambiar de Actividad. Fueron introducidos a partir de la API 21 de Android.

        iii) Notificador Android: engloba los servicios para notificar al usuario cuando algo requiera su atención mostrando alertas en la barra de estado.

2) Capa de negocio:

    a) Componentes lógicos de gestión de vistas

i) Controladores de Actividades y Fragmentos: se encargan de ejecutar las tareas propias de la aplicación, en ellas se realizan cálculos, se validan y convierten al formato correcto los datos provenientes de la capa de presentación.

ii) Adaptadores: Permiten personalizar un elemento visual de la aplicación para que funcione según comportamientos definidos por el programador.

iii) Controladores de diálogos: Se encarga de ejecutar y controlar el comportamiento de los mensajes de diálogos en la aplicación.

b) Componentes de presentación de datos:

i) Presentadores de datos: Implementan la interfaz de presentación de datos, permitiendo establecer la comunicación con la capa de acceso a datos de la aplicación. Contiene los escuchadores de eventos de Realm.

3) Capa de acceso a datos:

a) Componentes de control y gestión de datos:

i) Repositorio de interfaces: Contiene las interfaces que posibilitan la comunicación entre esta capa con la BBDD de la aplicación.

ii) Repositorios: Consisten en la lógica de la gestión de datos. En ellos se implementan las consultas y transacciones con la BBDD.

iii) Módulos: Son una abstracción del modelo de datos.

iv) Tablas: Agrupa las clases del modelo en tablas, son usadas generalmente para las consultas gráficas.

4) Otros elementos serían:

a) Herramientas:

i) Consumidor de servicios web para Android (KSoap): encapsula los datos de la petición para consumir determinado servicio y obtiene la respuesta de la fachada para luego ser procesada por la aplicación.

ii) Controladores de sincronización: permiten la sincronización de la aplicación con GECMA a través de la utilización de la API de sincronización.

b) API de Sincronización:

i) Interfaz de adquisición: es la encargada de la adquisición de los datos. Es la encargada además de validar que el sistema que está haciendo la petición de sincronización es auténtico. La misma utiliza un sistema de peticiones HTTP 1.1 vasado en peticiones de tipo POST y devuelve los datos a través de tramas sobre el mismo protocolo.

ii) Mecanismos de sincronización y transformación de datos: en la API se implementan todos los mecanismos necesarios para garantizar de que no se inserten datos duplicados, de que se respeten las llaves y de que un usuario no autorizado no pueda insertar datos.

iii) Interfaz de comunicación: es la encargada del intercambio de datos entre la API y la BBDD de GECMA.

2.2 Desglose de responsabilidades

Como parte de la metodología OpenUP se encuentra la definición de responsabilidades para cada participante en el proyecto, definiéndose los diferentes roles a desempeñar, como se muestra en la tabla.

Rol	Responsabilidad	Nombre
Analista	Representa al cliente y las preocupaciones de los usuarios finales mediante la recopilación de información a los interesados, para entender el problema a resolver, por la captura y fijación de prioridades para las necesidades.	Alejandro Gómez Rivas Ernesto Alfonso Caballero
Arquitecto	Responsable de definir la arquitectura de software, que incluye tomar las principales decisiones técnicas que limitan el diseño y la	Alejandro Gómez Rivas Ernesto Alfonso Caballero

	implementación del sistema.	
Programador	Responsable del desarrollo de una parte del sistema, que incluye el ajuste del diseño a la arquitectura. Construye los prototipos de la interfaz de usuario, de la aplicación y de la unidad de pruebas.	Alejandro Gómez Rivas Ernesto Alfonso Caballero
Líder de proyecto	Responsable de la planificación del proyecto, coordina las interacciones con las partes interesadas y mantiene al equipo de proyecto centrados en el cumplimiento de los objetivos del mismo.	Alejandro Gómez Rivas Ernesto Alfonso Caballero
Stakeholder	Representa los grupos de interés cuyas necesidades deben ser satisfechas por el proyecto.	Drc. Antonio Cedeño Pozo Msc. Yidian Y. Castellanos Sabarí
Probador	Responsable de las actividades básicas de la prueba de esfuerzo. Esas actividades incluyen la identificación, definición, implementación y realización de las pruebas necesarias, así como registrar los resultados de las pruebas y análisis de los resultados.	Alejandro Gómez Rivas Ernesto Alfonso Caballero

Tabla 1: Desglose de responsabilidades por miembro

2.3 Listado de requisitos funcionales y no funcionales

La administración de requisitos es una de las actividades iniciales de OpenUP para la posterior confección de la especificación de requisitos, donde se definen de manera priorizada, los requisitos funcionales y no funcionales de la aplicación.

Requisitos Funcionales (RF) de GECMABOT:

Un requisito funcional define servicios o funciones que el cliente requiere de un sistema (Sommerville, 2005).

Prioridad Alta:

1. Registrar usuario: posibilitar que cada usuario cree una cuenta personalizada para cada dispositivo.
2. Autenticar usuario: proveer una interfaz para que el usuario se autentique antes de acceder a la aplicación.
3. Gestionar grupo: proveer un mecanismo para que el usuario pueda obtener los listados de los estudiantes de su(s) grupo(s), así como la posibilidad de actualizar estos listados y eliminar un grupo.
4. Gestionar estudiantes: posibilitar que el usuario modifique datos de un estudiante, así como acceder a la información del mismo o eliminarlo de la BBDD.
5. Gestionar actividad: posibilitar que el usuario pueda crear, actualizar o eliminar actividades.
6. Establecer asistencia: posibilitar que el profesor establezca la asistencia de un estudiante en una actividad existente.
7. Establecer evaluación: posibilitar que el profesor evalúe a un estudiante en una actividad.
8. Sincronizar: permitir el intercambio seguro de datos entre la aplicación y GECMA además de garantizar la integridad de datos en ambas aplicaciones.
9. Generar registro docente: posibilitar la obtención de un registro con todas las asistencias y evaluaciones de los estudiantes de un grupo.

Prioridad media:

10. Realizar corte evaluativo: posibilitar la realización de un corte evaluativo para un estudiante con los datos de asistencias y evaluaciones.

11. Generar reportes: posibilitar que se muestren gráficas con reportes de asistencias y evaluaciones por estudiante y grupo.

Requisitos funcionales de la API de sincronización:

Prioridad alta:

12. Autenticar usuario: posibilitar la comprobación de que el usuario tiene permisos para escribir datos en GECMA.
13. Insertar datos: posibilitar la inserción datos en la BBDD de GECMA.
14. Actualizar datos: posibilitar la modificación de datos existentes en GECMA.

Requisitos no funcionales (RNF):

Software:

1. La solución debe ser funcional para el SO Android a partir de la versión 3.0.
2. Se requiere un dispositivo con capacidad para conectarse a redes Wifi.
3. Se requiere tener instalado en el servidor Python en su versión 2.7.11 y Django en su versión 1.5 o superior.

Confiabilidad:

4. La solución cuenta con los mecanismos de sincronización necesarios para que el intercambio de datos entre aplicaciones sea confiable.
5. La solución tiene un usuario o contraseña para evitar que personas no deseadas modifiquen datos en la misma.

Ambiente:

6. La sincronización y la gestión de grupos solo pueden realizarse en la Universidad de las Ciencias Informáticas, ya que se precisa de conexión a la intranet institucional de la misma.

Soporte:

7. La aplicación se realiza con el empleo de Java y Android SDK 23.
8. La API Web se realiza con el empleo de Django Rest Framework 3 mediante la utilización del IDE PyCharm 5.0.4.
9. La aplicación se realiza en el IDE Android Studio 1.5.

2.4 Análisis del dominio

Como parte de la fase de elaboración vista en la metodología OpenUP, se realiza un análisis del dominio y definición de la arquitectura del sistema. Se brinda al equipo de proyecto un prototipo funcional inicial de la solución, el cual se irá modificando según los requisitos que busque el cliente, hasta llegar a la propuesta final.

2.4.1 Diagrama de clases del dominio

El modelo de dominio es una representación visual de los conceptos u objetos del mundo real significativos para un problema o área de interés. Representa clases conceptuales del dominio del problema, conceptos del mundo real, no de componentes de software (Larman, 2003).

La Figura 10 muestra el modelo de dominio realizado, teniendo en cuenta lo planteado en la descripción del problema.

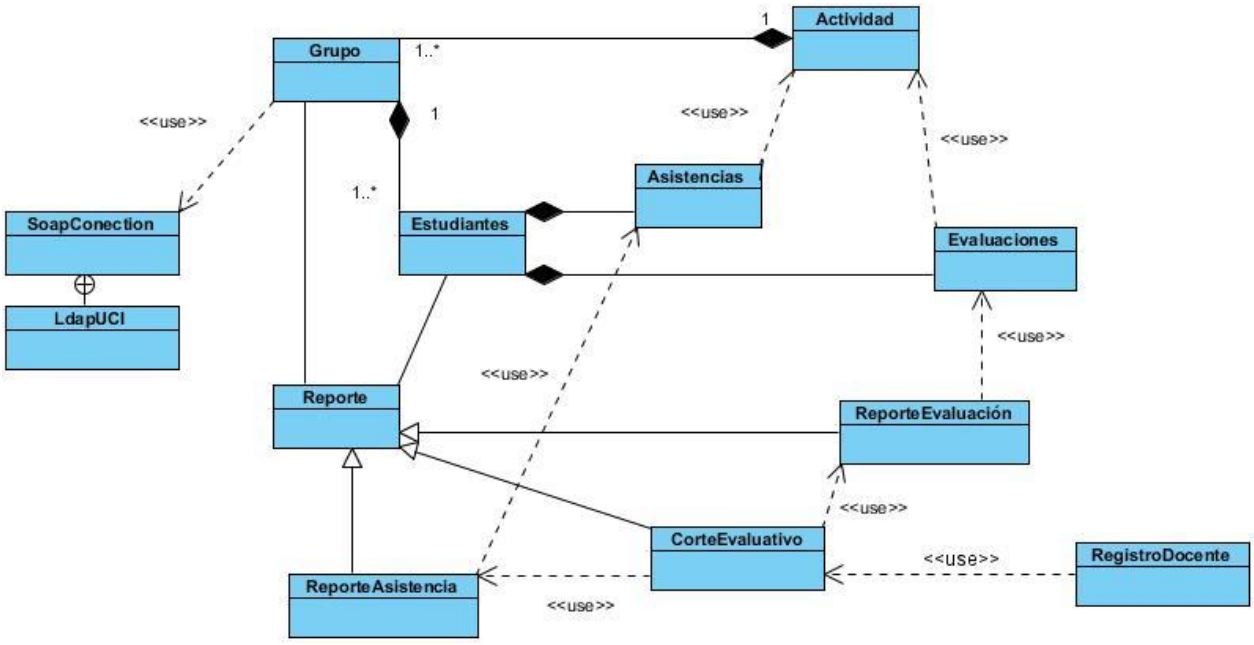


Figura 10. Diagrama de clases del dominio.

2.5 Patrones utilizados

Un patrón es un conjunto de información que proporciona respuesta a un conjunto de problemas similares. Para ello se aíslan los aspectos comunes y su solución, luego se añade cuantos comentarios y ejemplos sean oportunos. Los patrones ayudan a capturar conocimiento y a crear un vocabulario técnico, hacen el diseño orientado a objetos más flexible, elegante y en algunos casos reusable.

2.5.1 Patrones GRASP

Patrones de Asignación de Responsabilidad (GRASP, por sus siglas en inglés General *Responsibility Assignment Software Patterns*) representan los principios básicos de la asignación de responsabilidades a objetos, expresados en forma de patrones.

Polimorfismo: Este patrón se evidencia cuando se tienen varios objetos que presentan comportamientos similares y se asigna el mismo nombre a una función con la que se va a obtener la misma información, aunque su forma de obtenerla sea diferente.

```
10      public Estudiante() {
11      }
12
13      public Estudiante(String nombre, String apellidos) {
14          this.nombre = nombre;
15          this.apellidos = apellidos;
16      //    this.imgUrl = imgUrl;
17      }
18
```

Figura 11: Ejemplo de uso del patrón Polimorfismo

Bajo Acoplamiento: El bajo acoplamiento es un principio que se debe tener en cuenta durante las decisiones de diseño. Soporta el diseño de clases más independientes, que reducen el impacto de los cambios, y también más reutilizables, que acrecienten la oportunidad de una mayor productividad. En el diseño de clases de la solución propuesta las clases se encuentran lo menos ligadas entre sí que se pueda de tal forma que en caso de producirse una modificación en alguna de ellas, se tenga la mínima repercusión posible en el resto de clases, potenciando la re-utilización y disminuyendo la dependencia entre las clases (Larman, 2003). En la aplicación se evidencia en la mayoría de las clases controladoras, pues no depende del

funcionamiento de las otras, de manera que modificar una clase afectaría muy poco o casi nada al resto de la aplicación.

Creador: Ayuda a identificar quién debe ser el responsable de la creación (o instanciación) de nuevos objetos o clases (Larman, 2003). En el desarrollo de la solución planteada se trabaja con clases que realizan esta función, como por ejemplo la clase Grupo, la cual es encargada de crear o instanciar a los objetos Estudiantes.

Controlador: Se basa en asignar una responsabilidad de recibir o manejar un mensaje de evento del sistema a una clase que representa una de las opciones siguientes: Representa el sistema global, dispositivo, subsistema o representa un caso de uso en el que tiene lugar el evento del sistema a menudo denominado. En la aplicación se tienen controladores que representan los casos de uso de la aplicación.

2.5.2 Patrones GoF

Los patrones “Banda de los Cuatro” (GoF, por sus iniciales en inglés *Gang of Four*), describen las formas comunes en que diferentes tipos de objetos pueden ser organizados para trabajar unos con otros. Tratan la relación entre clases, la combinación de clases y la formación de estructuras de mayor complejidad. Existen tres tipos de patrones: los de creación que abstraen el proceso de creación de instancias, estructurales que se ocupan de cómo las clases y objetos son utilizados para componer estructuras de mayor tamaño y de comportamiento que se refieren a los algoritmos y a la asignación de responsabilidades entre objetos.

A continuación, se identifican los patrones utilizados en el desarrollo de la aplicación, de acuerdo a su clasificación.

***Abstract Factory* (asociado al grupo Creación):** Dado un conjunto de clases abstractas relacionadas, el patrón *Abstract Factory* permite el modo de crear instancias de estas clases abstractas desde el correspondiente conjunto de subclases concretas. Proporciona una interfaz para crear familias de objetos relacionados o dependientes sin especificar su clase concreta. El patrón *Abstract Factory* puede ser muy útil para permitir a un programa trabajar con una variedad compleja de entidades externas, tales como diferentes sistemas de ventanas con una funcionalidad similar.

En la solución planteada este patrón es utilizado con el objeto *Intent*, permitiendo cuando se instancia especificar el tipo de clase que va a crear.

```
webservicePG.setVisibility(View.INVISIBLE);
Intent intObj = new Intent(LoginActivity.this, HomeActivity.class);
//Comprueba que no existan errores
if (!errored) {
    //Ejecuta una accion basado en si el usuario fue autenticado correctamente
    if (loginStatus) {
        //Cambia hacia la Actividad "Home"
        startActivity(intObj);
```

Figura 12. Ejemplo del uso del patrón Abstract Factory

***Interface* (asociado al grupo Creación):** Una interfaz se define como una "Conexión física y funcional entre dos aparatos o sistemas independientes" [Rae 2013], esta describe las operaciones (la cara al mundo) que una entidad puede realizar, de igual manera establece los limites, niveles de acceso y la manera en que se desarrolla la comunicación entre dos entidades, para este caso las diferentes capas de la aplicación. Por lo general, se refiere a una abstracción que proporciona un activo de sí mismo a la parte exterior.

***Decorator* (asociado al grupo Estructura):** El patrón *Decorator* responde a la necesidad de añadir dinámicamente funcionalidad a un Objeto. Esto permite eliminar la creación de sucesivas clases que hereden de la primera incorporando una nueva funcionalidad, sino otras que la implementan y se asocian a la primera**.** En la aplicación se tienen diferentes Presentadores, que no son más que objetos que escuchan y añaden funcionalidades a los controladores de interfaz (Lazyloading (2), 2008).

***Observer* (asociado al grupo Comportamiento):** *Observer* es un patrón de diseño que define una dependencia del tipo uno-a-muchos entre objetos, de manera que cuando uno de los objetos cambia su estado, notifica este cambio a todos los dependientes**.** Las clases Presentadores igualmente incluyen este patrón, que implementa un sistema de observadores llamados *callbacks* encargados de escuchar los cambios en la BBDD y hacer modificaciones en los controladores de interfaz (Lazyloading, 2008).

```
onAddActividadCallback = new IActividadRepository.onAddActividadCallback(){

    @Override
    public void onSuccess() {
        view.showMessage("Actividad adicionada satisfactoriamente");
        getActividades();
    }

    @Override
    public void onError(String mensaje) { view.showMessage(mensaje); }
};
```

Figura 13: Ejemplo del uso de los patrones *Decorator* y *Observer* en la aplicación

***State* (asociado al grupo Comportamiento):** Este patrón consiste en permitir que un objeto modifique su comportamiento cada vez que cambie su estado interno. Esto se puede observar en las diferentes clases Asíncronas que realizan operaciones en la aplicación. Las mismas tienen varios comportamientos en dependencia del estado en que se encuentra el hilo de ejecución.

```
protected void onPreExecute() {
    webservicePG.setVisibility(View.VISIBLE);
}

@Override
protected Void doInBackground(String... params) {
    //Invoca el SSL Web Method
   AuthRequest auth = new AuthRequest(NAMESPACE,HOST, WS_OPS, SOAP_ACTION, METHOD_NAME);
   loginStatus = auth.makeAuthRequestLogin(editTextUsername, editTextPassword);
   return null;
}

@Override
//Obtiene la respuesta de ejecucion del metodo doInBackground
protected void onPostExecute(Void result) {
    //Hace la barra de progreso invisible
    webservicePG.setVisibility(View.INVISIBLE);
    Intent intObj = new Intent(LoginActivity.this, HomeActivity.class);
    ...

// Este metodo se ejecuta cuando la barra de progreso se va actualizando
@Override
protected void onProgressUpdate(Void... values) {
}
```

Figura 14: Ejemplo del patrón State aplicado en una clase Asíncrona

2.6 Descripción de los Actores del sistema

En el Lenguaje Unificado de Modelado (UML), un actor "especifica un rol jugado por un usuario o cualquier otro sistema que interactúa con el sujeto" (UML, 2014).

La descripción de los actores que interactúan con el sistema que corresponde a la solución propuesta, se muestra a continuación:

Actores	Justificación
Usuario Nuevo	Es el usuario que ejecuta por primera vez la aplicación. A este se le da la posibilidad de crear una cuenta para poder acceder a la aplicación.
Usuario Desconocido	Es el usuario que aún no está autenticado en la aplicación y por lo tanto no puede hacer uso de la misma.
Usuario Autenticado	Es el usuario con todos los privilegios para poder usar la aplicación y modificar los datos de la misma.

Tabla 2. Actores del sistema.

2.7 Diagrama de Casos de Uso del sistema

En UML, un diagrama de casos de uso es una forma de diagrama de comportamiento UML, la descripción escrita del comportamiento del sistema al afrontar una tarea de negocio o un requisito de negocio (UML. 2014).

Siguiendo lo antes planteado, se muestra en la Figura 15 el diagrama del modelo de casos de uso para la solución propuesta.

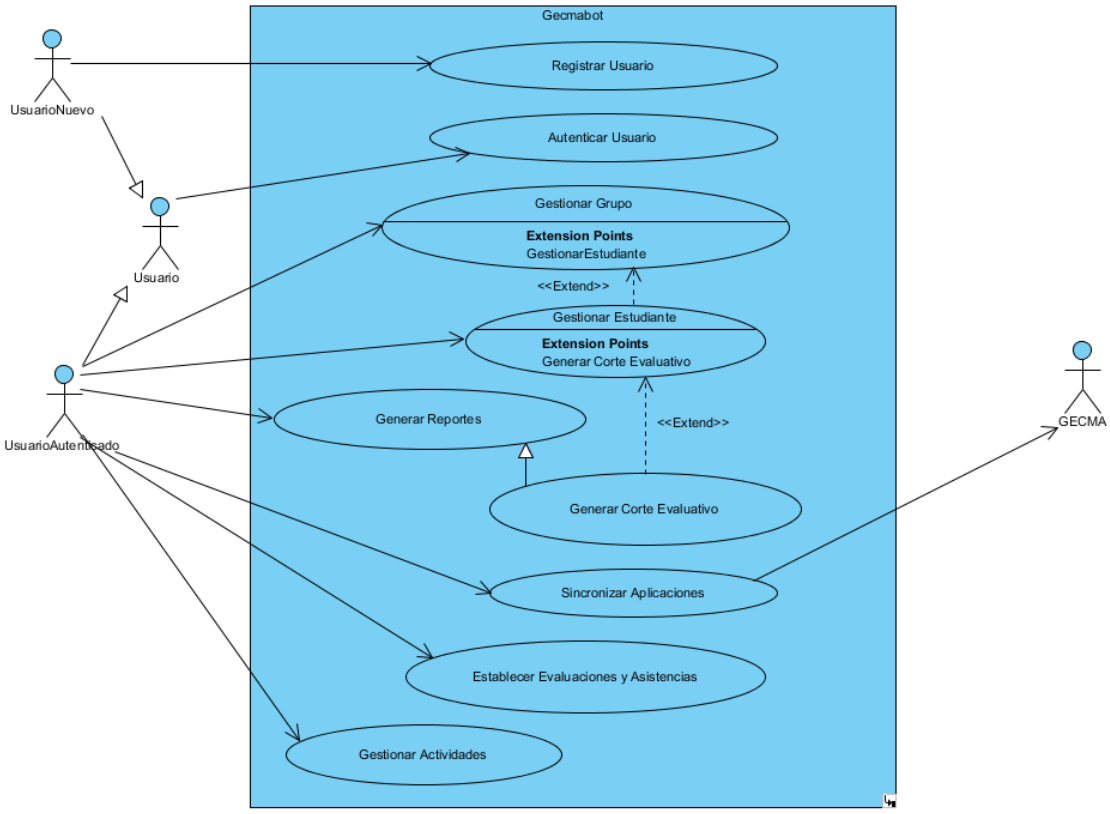


Figura 15: Modelo de casos de uso del sistema

2.7.1 Patrones de casos de uso utilizados

Son comportamientos que deben existir en el sistema, ayudan a describir qué es lo que el sistema debe hacer, es decir, describen el uso del sistema y cómo este interactúa con los usuarios. Estos patrones son utilizados generalmente como plantillas que describen cómo debería ser estructurados y organizados los casos de uso. Son patrones que capturan mejores prácticas para modelar casos de uso (Larman, 2003).

Patrón CRUD: el patrón CRUD Completo consiste en un caso de uso para administrar la información, permite modelar las diferentes operaciones para administrar una entidad de información, tales como crear, leer, cambiar y eliminar o dar de baja (Larman, 2003). En la aplicación existen los dos tipos de patrones CRUD (Completo y Parcial). El CRUD Completo está presente en el caso de uso Gestionar Grupo donde se pueden realizar todas las operaciones sobre el modelo Grupo. El CRUD Parcial se puede ver en el caso de uso Gestionar

Estudiantes, donde los estudiantes no son creados en el mismo caso de uso, sino que son obtenidos cuando se gestiona un grupo.

Múltiples actores con roles diferentes: la herencia de actores va a ser la distribución a lo largo de una jerarquía de roles, de las actividades a realizar, representadas estas como casos de uso. La herencia nace como una forma de organizar los enlaces entre actores y casos de uso a fin de simplificar los diagramas y reducir la necesidad de presentar información repetida (Larman, 2003). Esto se puede apreciar en la Generalización de Usuario.

Extensión Concreta: un caso de uso extiende a otro, cuando sin alterar a este, se incorpora su funcionalidad como parte integral del primero. Se denota con una relación que apunta del caso extendido al caso base y la conexión se hace al principio del flujo de eventos del caso base o en alguno de los puntos de extensión definidos (Larman, 2003). Se puede apreciar en las relaciones entre Generar Corte Evaluativo y Gestionar Estudiante, así como Gestionar Estudiante y Gestionar Grupo.

Generalización: un caso de uso (sub-caso) hereda el comportamiento y significado de otro, las relaciones de comunicación, inclusión y extensión del súper-caso de uso. En muchas ocasiones este súper-caso de uso es abstracto y corresponde a un comportamiento parcial completado en el sub-caso de uso. Los casos de uso “hijo” son una especialización del caso de uso “padre” (Larman, 2003). Es el caso de Generar Corte Evaluativo, el cual es una especialización del caso de uso Generar Reportes.

2.8 Descripción de los casos de uso

La descripción de los casos de uso brinda una información detallada del caso de uso mediante la cual se logra una mejor comprensión del mismo. A continuación, se muestra el resumen de la descripción de los casos de uso. Para una profundización consultar el Anexo 2: Descripción de los Casos de Uso.

Registrar Usuario:

Identificador del caso de Uso	Registrar Usuario
Criticidad	Alta
Complejidad	Baja
Descripción	El Caso de Uso inicia cuando el Usuario accede al sistema por

	primera vez. El Sistema le muestra un Formulario de Registro el cual llena y si los datos están completos son almacenados en la BBDD creando así el nuevo usuario de la aplicación.

Gestionar Grupo:

Identificador del caso de Uso	Gestionar Grupo
Criticidad	Alta
Complejidad	Alta
Descripción	El Caso de Uso inicia cuando el usuario decide Gestionar un Grupo en la aplicación, el mismo introduce el Identificador de un grupo en el campo indicado, luego el Sistema busca que el grupo exista en el servidor LDAP de la universidad y descarga los datos de los estudiantes asociados a ese grupo a la BBDD.

Gestionar Actividades:

Identificador del caso de Uso	Gestionar Actividades
Criticidad	Alta
Complejidad	Alta
Descripción	El Caso de Uso inicia cuando el Usuario Autenticado decide Gestionar las Actividades. El mismo puede escoger entre las opciones "Adicionar Actividad", "Modificar Actividad" o "Eliminar Actividad" y realizar las acciones pertinentes en cada una de ellas. Al terminar los cambios son guardados o actualizados en la BBDD.

Gestionar Estudiantes:

Identificador del caso de Uso	Gestionar Estudiantes
Criticidad	Alta
Complejidad	Alta
Descripción	El Caso de uso inicia cuando el Usuario Autenticado desea modificar o visualizar a un estudiante. Para ello accede al apartado de grupos, donde se muestra una lista de los grupos. Al seleccionar un grupo se muestra un listado de los estudiantes de ese grupo, y brinda al Usuario la posibilidad de Modificar o eliminar los datos de un estudiante, así como visualizar su ficha.

Generar Reportes:

Identificador del caso de Uso	Generar Reportes
Criticidad	Media
Complejidad	Alta
Descripción	El Caso de uso inicia cuando el usuario selecciona la vista detallada de un estudiante en la pantalla del grupo. El sistema muestra una lista con las actividades existentes y el estado del estudiante en la misma, así como una vista grafica del estado de la asistencia y las evaluaciones del estudiante.

Generar Corte Evaluativo:

Identificador del caso de Uso	Generar Corte Evaluativo
Criticidad	Media
Complejidad	Alta
Descripción	El Caso de Uso inicia cuando el usuario decide generar un corte evaluativo, tanto para un estudiante como para un grupo completo. El Sistema calcula mediante una formula y luego nominaliza el valor obtenido en los posibles resultados mostrándolo en pantalla.

Sincronizar Aplicaciones:

Identificador del caso de Uso	Sincronizar Aplicaciones
Criticidad	Alta
Complejidad	Alta
Descripción	El Caso de Uso inicia cuando el usuario autenticado escoge la opción del menú "Sincronizar". A partir de aquí el Sistema muestra un formulario de autenticación. El usuario introduce sus credenciales de GECMA y si la autenticación es correcta el sistema envía los datos a la API de Sincronización la cual transforma estos datos y los envía al sistema GECMA para que sean guardados en la BBDD del mismo.

Establecer Evaluaciones y Asistencia:

Identificador del caso de Uso	Establecer Evaluaciones y Asistencias
Criticidad	Alta
Complejidad	Media
Descripción	El Caso de Uso inicia cuando el Usuario Autenticado, accede a una actividad previamente creada. En la misma se le permite pasar asistencia a los estudiantes de sus grupos, así como registrar las evaluaciones a una actividad. Las mismas quedan guardadas para cada estudiante en la BBDD.

Autenticar Usuario:

Identificador del caso de Uso	Autenticar Usuario
Criticidad	Alta
Complejidad	Baja
Descripción	El Caso de Uso inicia cuando un usuario ejecuta la aplicación. El sistema muestra el formulario de autenticación, donde el usuario introduce sus datos que, de ser correctos, le permiten acceder al sistema.

2.9 Modelo de datos de la aplicación

Un modelo de datos es una estructura abstracta que documenta y organiza la información para la comunicación. En la informática, se centra en el planeamiento del desarrollo de aplicaciones y la decisión de cómo se almacenan los datos y cómo se accede a ellos (Alegsa, 2008). El modelo relacional se caracteriza a muy grandes rasgos por disponer que toda la información debe estar contenida en tablas, y las relaciones entre datos deben ser representadas explícitamente en esos mismos datos (ver Figura 16).

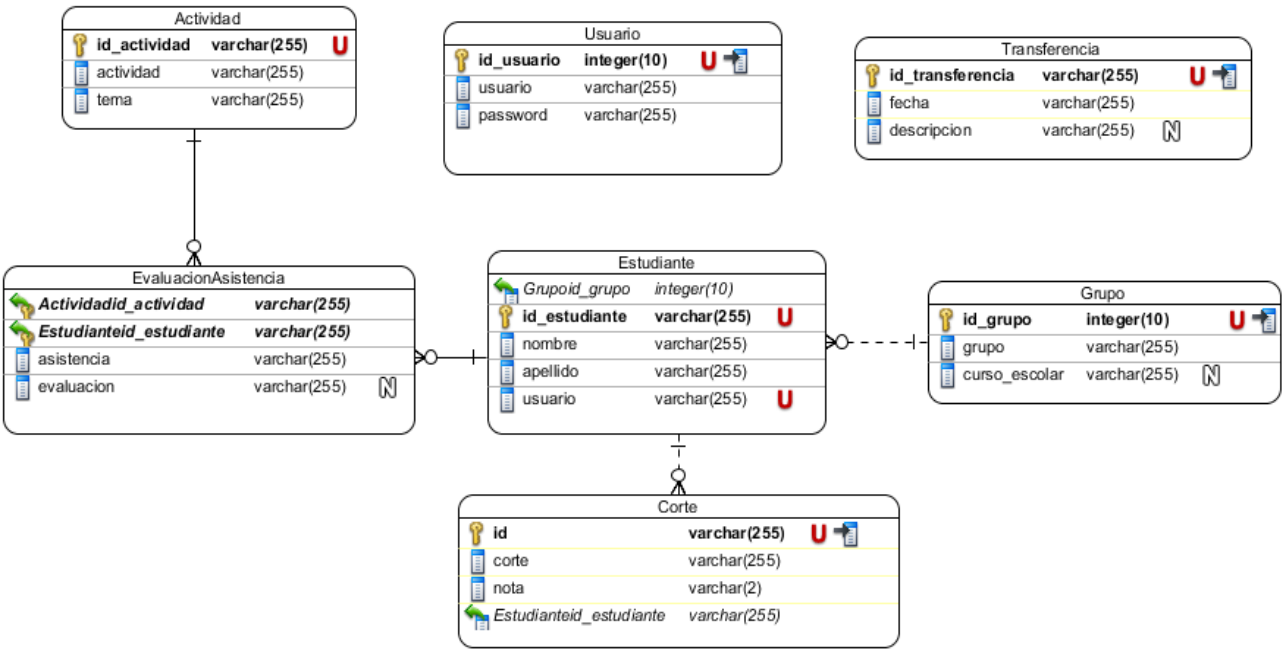


Figura 16. Modelo de datos de la aplicación

2.10 API de Sincronización

Como parte del diseño de la API de sincronización se tuvieron en cuenta varios aspectos con el propósito de garantizar la confiabilidad, seguridad e integridad de los datos:

- **Diferencias en los modelos de datos:**

  Uno de los primeros problemas con los que se lidió fue con la diferencia existente en los modelos de datos. La aplicación GECMA proponía un modelo diferente al de GECMABOT (ver Figura 17). Al ser una aplicación diseñada para un trabajo menos diario, en la misma no se trabaja con Actividades, sino que para cada estudiante se tiene un registro mensual de las asistencias y un registro de las evaluaciones por tipo de actividad. Otro de los problemas encontrados radica en que, en la tabla estudiante, así como en la tabla evaluación existen columnas con información que no está contemplada en GECMABOT.

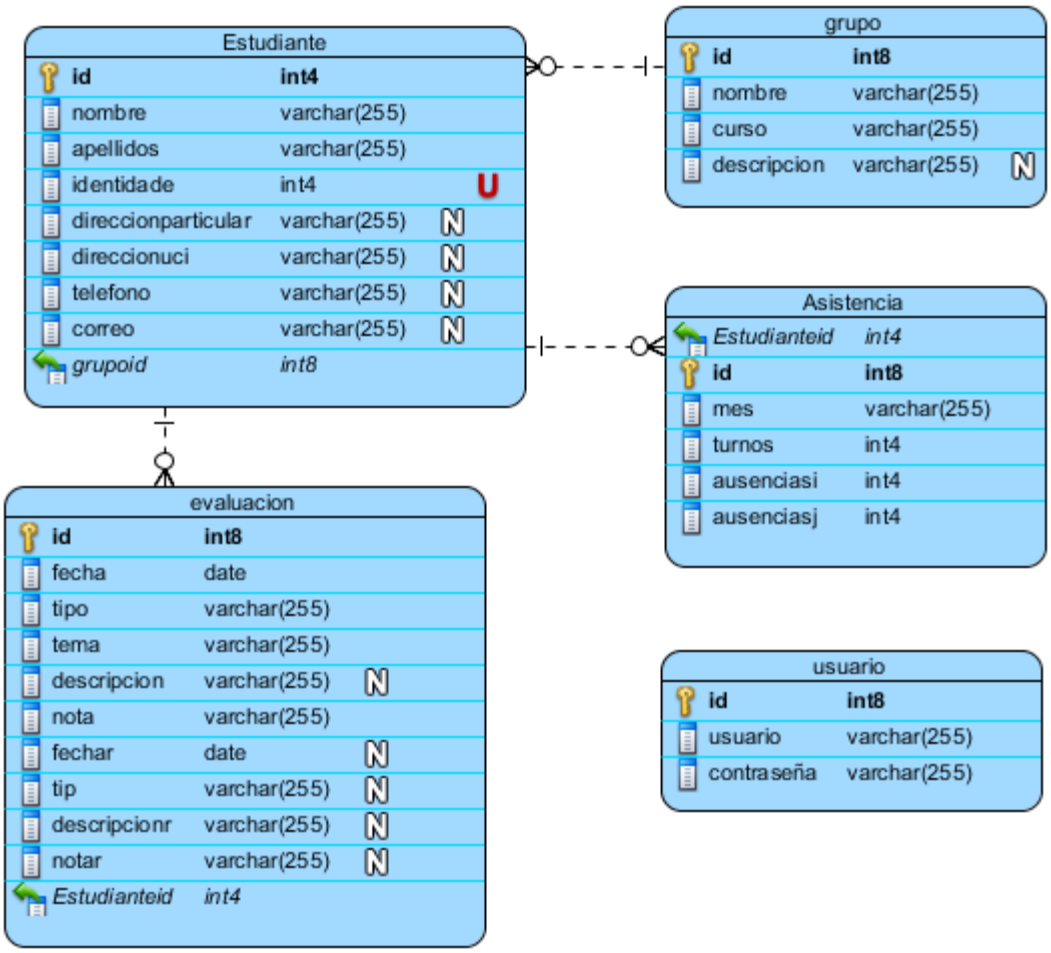


Figura 17. Diagrama de Entidad Relación de a aplicación GECMA según el estudio realizado (fabricación propia)

- **Índices numéricos:**

  Otra de las diferencias en los modelos de ambas aplicaciones radica en que en GECMA se utilizan índices numéricos de tipo largo (*long* o *int8*) mientras que en GECMABOT, el modelo Realm, al no poseer llaves auto-incrementales, plantea la utilización de llaves de cadena (*String* o *varchar(255)*) para las llaves.

Para dar solución a estos inconvenientes y mantener la integridad de los datos se acordaron las siguientes resoluciones para el diseño de la API de sincronización:

- Para los campos diferentes (Ejemplo: estudiante.direccionparticular) que, según el modelo de GECMA pueden ser nulos, se toma la decisión que estos son dejados nulos o vacíos.

- Para las llaves se toman las siguientes resoluciones:

    - Para la tabla Estudiante se construye una llave de tipo *int8* a partir del ID del estudiante en la aplicación GECMABOT, que coincide con el número de expediente del sistema Akademos 2.0. (Ejemplo: Para el ID = EH15239 se genera un nuevo ID = 15239)

    - Para la tabla Evaluación se construye una llave de tipo *int8* a partir de la suma del CI del estudiante y la fecha (F) de la actividad (Ejemplo: Para CI = 92091004665 y F= 6-5-2016 se genera un ID = 920910006692).

    - Para la tabla Asistencia se construye una llave de tipo *int8* a partir de la suma del CI del estudiante y el mes (M) de las evaluaciones (Ejemplo: Para CI = 92091004665 y M= 5 se genera un ID = 92091004670).

    - Para la tabla Grupo se utilizará una modificación hacia *int8* del formato de id de grupo de la aplicación GECMABOT, el cual es obtenido del servicio web de postgrado del sistema Akademos 2.0 (Ejemplo: Para el ID: 100-581 se genera el nuevo ID = 100581).

- A partir de las llaves generadas, en caso de existir colisiones se realiza una actualización de las columnas diferentes (*update*) y se registra la colisión en la tabla Transacciones de la aplicación GECMABOT.

Teniendo en cuenta que en la UCI existe el Sistema de Gestión Universitaria, con una BBDD actualizada y con mayor grado de completamiento, se decide que la aplicación GECMABOT descargará los datos sobre estudiantes y grupos de estudio de este sistema. De esta forma se contribuye a mejorar la calidad de los datos actuales almacenados en GECMA. A partir de ahí se logra la compatibilización con el registro de estudiantes del Sistema de Gestión Universitaria, que puede resultar interesante para un posible intercambio entre ambas aplicaciones.

2.11 Conclusiones Parciales

En este capítulo se identifica una visión general de lo que es la solución del sistema. Se utiliza para ello como metodología de desarrollo OpenUP. Del mismo se obtienen las siguientes conclusiones:

- GECMABOT es un sistema que funciona en dispositivos inteligentes que cuenten con el SO Android a partir de la versión 3.0.
- Para la solución se definen once requisitos funcionales y tres requisitos funcionales para la API de sincronización, los cuales están agrupados en nueve casos de uso.
- Mediante el uso de patrones de diseño para la construcción de la aplicación se facilita una mayor comprensión del código y se lograr un diseño orientado a objetos más flexible y reutilizable para construcción de aplicaciones similares.

Capítulo 3: Implementación y prueba del sistema

En este capítulo se aborda la fase de implementación y prueba de la solución propuesta, guiada por el análisis y diseño de la misma, abordados en el Capítulo 2. Análisis y diseño de la solución propuesta. Una vez finalizada la fase de implementación se realiza el diseño y estudio de los diferentes casos de prueba a aplicar a la solución con el fin de asegurar la calidad y el buen funcionamiento de la misma.

3.1 Implementación

La implementación de la solución desarrollada se corresponde con la Fase de Construcción definida por la metodología OpenUP. Para la obtención del resultado esperado son necesarias cuatro iteraciones. Cada iteración termina con un demo que se presenta al cliente en la que se determinan las no conformidades y se planifica la siguiente iteración. El resultado de cada iteración y las modificaciones realizadas en cada una de ellas se detallan a continuación.

Iteración	Descripción	Duración
1	• Aplicación funcional que permite la gestión del usuario, la gestión de estudiantes, grupos y actividades	8 semanas
2	• Se agregan los reportes • Se añade la pantalla de *splash* al inicio • Se realizan mejoras al diseño • Se corrigen errores de la iteración anterior • Se entrega la versión Beta funcional de la aplicación	4 semanas
3	• Se realizan mejoras al diseño • Se implementa la Api de sincronización y se adiciona esta funcionalidad a la aplicación • Se adiciona la funcionalidad de cortes evaluativos • Se establece la compatibilidad con los diferentes dispositivos y tamaños de pantalla (*Responsive*)	6 semanas
4	• Se corrigen las no conformidades encontradas • Se refina la interfaz según el criterio de los *stakeholders.* • Se entrega la versión 1.0 estable de la solución	3 semanas

Tabla 3. Modificaciones de la fase de implementación.

3.1.1 Estándar de codificación

Una de las buenas prácticas utilizadas en la construcción de la solución planteada lo constituye la adopción del estándar de codificación ***CamelCase***, asegurando que el código exprese claramente el propósito del mismo y agilice el proceso de refactorización, que sea legible, entendible y refleje un estilo armonioso.

3.1.2 Diagrama de componentes

Un diagrama de componentes representa cómo un sistema de software es dividido en componentes y muestra las dependencias entre estos. Los diagramas de componentes prevalecen en el campo de la arquitectura de software, pero pueden ser usados para modelar y documentar cualquier arquitectura de sistema. Estos son utilizados para modelar la vista estática y dinámica de un sistema. Muestran la organización y las dependencias entre un conjunto de componentes (Pressman, 2010).

El Diagrama de componentes de la solución propuesta se representa en el Anexo 3.

3.1.3 Diagrama de despliegue

Es un tipo de diagrama del UML que se utiliza para modelar la disposición física de los artefactos del software en nodos (UML, 2014).

Figura 18. Diagrama de despliegue.

3.2 Validación de la solución propuesta

Durante y después del proceso de implementación, el programa que debe ser comprobado para asegurar que satisface los requerimientos del cliente (Sommerville, 2005). Los autores del presente trabajo de investigación hacen uso de las técnicas de evaluación estáticas en las fases de análisis y diseño de la solución. Se realizan en forma de revisiones, de carácter informal, los diferentes artefactos que son necesarios generar para la posterior implementación. La metodología OpenUP define las pruebas funcionales basadas en casos de prueba, razón por la cual una vez finalizadas las diferentes fases de implementación se llevan a cabo técnicas de

evaluación dinámicas, tras seguir el método de caja negra en la generación de casos de prueba. Los tipos de prueba que se realizan en esta fase fueron tipos de prueba de funcionalidad. Estas se realizan con el objetivo de validar si el comportamiento observado del software cumple o no con sus especificaciones, como lo constituyen las **pruebas de aceptación**.

3.2.1 Pruebas de aceptación

Cuando se realizan este tipo de pruebas, el producto está listo para implantarse en el entorno del cliente. Están enfocadas a probar los requisitos y los criterios de aceptación, si no se consigue demostrar el incumplimiento de tales requisitos el cliente deberá aceptar el producto. En caso de un producto de interés general, se realizan pruebas con varios usuarios que reportarán sus valoraciones sobre el producto. Para la generación de casos de prueba de aceptación se utilizan técnicas de caja negra.

3.2.1.1 Casos de prueba de aceptación

En correspondencia con los casos de usos más importantes, se diseñaron los casos de prueba que son basados en escenarios, como se observa en la Tabla 4.

Id del Escenario	Escenario	Descripción	Respuesta del Sistema	Resultado de la prueba.
EC1	Escenario 1: Ejecutar la aplicación por primera vez.	El usuario ejecuta el sistema por primera vez y no existe, por tanto, ningún usuario en la aplicación.	El Sistema muestra el formulario de registro.	Satisfactoria.
EC2	Escenario 2: Campos vacíos al registrarse.	El usuario deja campos vacíos en el formulario de registro.	El Sistema muestra un mensaje indicándole que no puede dejar campos vacíos y permanece en la	Satisfactoria.

			pantalla del formulario de registro.	
EC3	Escenario 3: Campos vacíos al autenticarse.	El usuario deja campos vacíos en el formulario de autenticación.	El Sistema muestra un mensaje indicándole que no puede dejar campos vacíos y permanece en la pantalla del formulario de autenticación.	Satisfactoria.
EC4	Escenario 4: Usuario o contraseña incorrectos.	El usuario introduce un usuario y/o una contraseña inválidos en el formulario de autenticación.	El Sistema muestra un mensaje indicándole que el usuario o la contraseña son incorrectos y permanece en la pantalla del formulario de autenticación.	Satisfactoria.
EC5	Escenario 5: Crear actividades sin tener grupos registrados.	El usuario va a crear actividades sin que exista al menos un grupo en la aplicación.	El Sistema re direcciona al usuario a la sección de grupos indicándole que no hay grupos en	Satisfactoria.

			la aplicación.	
EC6	Escenario 6: Desconexión de la Wifi UCI al buscar un grupo o realizar la sincronización.	El usuario no está conectado a la wifi, o se encuentra fuera del entorno de la UCI al querer buscar y descargar los datos de un grupo o al realizar la sincronización.	El sistema muestra el mensaje: "Error de conexión. Compruebe su conexión a la red interna de la UCI"	Satisfactoria.
EC7	Escenario 7: Introducir un grupo incorrecto o dejar el campo vacío.	El usuario introduce los datos de un grupo que no existe o deja el campo vacío	El sistema muestra el mensaje: "Grupo incorrecto"	Satisfactoria.
EC8	Escenario 8: Campos sin seleccionar al crear o editar una actividad.	El usuario deja campos sin seleccionar al querer crear o editar una actividad.	El sistema muestra el mensaje "Debe seleccionar todas las opciones" y permanece en el formulario de creación/edición	Satisfactoria.
EC9	Escenario 9: Eliminar una actividad.	El usuario decide eliminar una actividad.	El sistema muestra un diálogo donde el usuario debe confirmar. De ser así procede a eliminar todos	Satisfactoria.

			los datos de asistencias y evaluaciones asociados a esa actividad.	
EC10	Escenario 10: Campos vacíos al editar un estudiante.	El usuario deja campos vacíos al querer editar un estudiante.	El sistema muestra el mensaje “Debe rellenar todos los campos” y permanece en el formulario de edición	Satisfactoria.
EC11	Escenario 11: Eliminar un estudiante.	El usuario decide eliminar un estudiante.	El sistema muestra un diálogo donde el usuario debe confirmar. De ser así procede a eliminar todos los datos de asistencias y evaluaciones asociados a ese estudiante.	Satisfactoria.
EC12	Escenario 12: Eliminar un grupo.	El usuario decide eliminar un grupo.	El Sistema muestra un diálogo donde el usuario debe confirmar. De ser así procede a	Satisfactoria.

			realizar el proceso de eliminación un estudiante para cada uno de los estudiantes de un grupo.	
EC13	Escenario 13: Campos faltantes al Sincronizar datos.	Cuando el usuario inicia la sincronización, dado que las bases de datos no son iguales, se envían datos que no corresponden con los datos de la BBDD de GECMA	El sistema establece un valor predeterminado para estos datos (Ej. null, 0) de forma tal que pueda ser realizada la sincronización con GECMA.	Satisfactoria.
EC14	Escenario 14: Usuario sin permisos sobre un grupo.	El usuario no tiene permisos en GECMA para registrar datos de un grupo determinado.	El sistema deja de sincronizar los datos de este grupo, mostrando una notificación al usuario.	Satisfactoria.
EC15	Escenario 15: Campos vacíos al cambiar la contraseña	El usuario deja campos vacíos al querer modificar la contraseña.	El sistema muestra el mensaje “Debe rellenar todos los campos” y permanece en el	Satisfactoria.

			formulario de edición	
EC16	Escenario 16: Contraseña vieja incorrecta.	El usuario introduce una contraseña no válida en el campo de contraseña vieja, en el formulario de modificar contraseña.	El sistema muestra un mensaje de "Contraseña incorrecta. Verifique la contraseña vieja" y permanece en el formulario de edición.	Satisfactoria.
EC17	Escenario 17: Corte evaluativo de un estudiante sin datos.	EL usuario selecciona la opción de generar el corte evaluativo sin que exista ninguna actividad o no estén registrados los datos de asistencia o evaluaciones para un grupo o un estudiante.	El sistema muestra un error diciendo que no hay datos suficientes para realizar el corte.	Satisfactoria.

Tabla 4. Escenarios de prueba de aceptación

Las pruebas de aceptación fueron realizadas al final de cada iteración, a partir de que cada iteración termina con un prototipo funcional de la solución. Al finalizar la primera iteración se obtuvo como resultado un total siete No Conformidades (NC), de ellas tres fueron clasificadas de Significativas (S), dos de No Significativas (NS) y dos Recomendaciones (R). En la segunda iteración se encontraron dos NC, de ellas una de tipo S y una de tipo R. La tercera iteración finalizó con cuatro NC, de ellas dos de tipo S, una de tipo NS y una de tipo R. En la iteración final no se obtuvo ninguna NC. La explicación anterior se puede ver reflejada en la Figura 19.

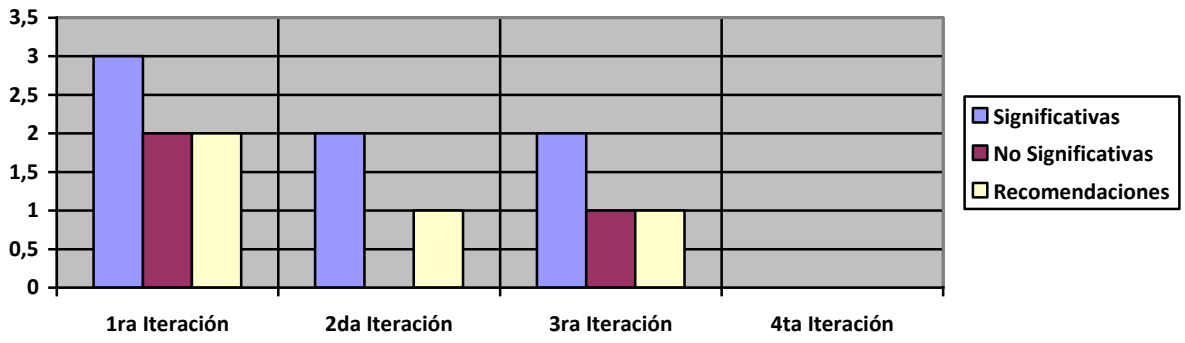


Figura 19: No conformidades encontradas según la iteración de prueba (elaboración propia)

NC de carácter S detectadas en la primera iteración

- El EC5 permite a los usuarios crear Actividades sin que existan grupos en la BBDD.
- El EC6 muestra un cartel que dice “Grupo Incorrecto” en vez de mostrar el cartel de error de conexión a la WIFI.
- El EC7 para determinados datos de nombres de grupos (Ej. 1000) crea el grupo sin estudiantes.

NC de carácter S detectadas en la segunda iteración

- En el EC9 cuando se elimina una actividad no se están eliminando correctamente los registros de asistencia y evaluaciones a los estudiantes.
- La acción de editar un estudiante elimina todos los registros de asistencia y evaluaciones de ese estudiante.

NC de carácter S detectadas en la tercera iteración

- En el EC14 la aplicación permite a profesores crear u modificar grupos a los que no están asignados.
- En la pantalla de grupo, al girar el teléfono, se lanza una excepción y se detiene la aplicación.

3.2.2 Pruebas de rendimiento

Con el objetivo de comprobar el tiempo de respuesta de la solución para las acciones permitidas en la misma, se realizan pruebas de rendimiento, utilizando para ello el IDE seleccionado y el emulador proporcionado por el ADT del marco de trabajo de Android. Los resultados obtenidos en estas pruebas demuestran que, en dependencia de las características del dispositivo móvil, así como de los reportes tratados correspondientemente; el tiempo de respuesta cumple con las características propuestas de rendimiento.

Para ver los resultados de estas pruebas y la respuesta de la herramienta. Ver Anexo 5.

3.3 Conclusiones Parciales

En este capítulo se describien las cuatro iteraciones que fue necesario realizar hasta llegar a la solución óptima. En las pruebas de aceptación en total se detectan trece NC, de ellas siete de tipo S, tres de tipo NS y cuatro de tipo R, siendo todas resueltas. Esto garantiza que la versión implementada en la iteración final cumple con las necesidades del cliente. Además, se puede verificar el rendimiento que tiene la aplicación en diferentes tipos de dispositivos y se certifica la satisfacción del cliente con la misma.

Después de logrado un 100% en la implementación de la solución, se puede concluir por parte de los autores y desarrolladores de la misma que:

- La sincronización de datos es un proceso complejo, en gran medida, porque al desarrollar una aplicación se debe tener en cuenta el correcto diseño de la BBDD.
- Las pruebas de software, al utilizar técnicas de evaluación dinámicas, aportan seguridad y fiabilidad a las aplicaciones y no deben ser tomadas a la ligera. Estas contribuyen a la obtención de no conformidades que ayudan a refinar y perfeccionar el producto de software que se entrega.

Conclusiones

El resultado de la investigación permite concluir de forma general que:

- A partir del estudio realizado en la etapa inicial de la investigación se puede afirmar que las tecnologías móviles constituyen una solución al problema de la gestión de datos en ambientes desconectados y que las soluciones existentes no satisfacen la problemática abordada.

- Mediante el uso de las tecnologías seleccionadas se obtiene un producto con un diseño simple e intuitivo que responde a las necesidades planteadas y mejora el control docente que se realiza a través de GECMA.

- El empleo de un proceso de desarrollo ágil y la realización de entregas de forma incremental posibilita el cumplimiento adecuado del cronograma de ejecución hasta arribar a la validación de la solución por parte del cliente.

Recomendaciones

A partir de los resultados alcanzados, dada las posibilidades de la introducción de las tecnologías móviles y la solución aportada para su empleo en escenarios desconectados se recomienda:

- Hacer extensible la aplicación al sistema Akademos 2.0, permitiendo la sincronización a la BBDD del mismo.
- Ajustar la aplicación para que pueda ser utilizada en todas las asignaturas de la universidad.

Bibliografía referenciada

Alegsa. 2008. Diccionario de informática y tecnología. *alegsa.com.ar.* [En línea] 2008. [Citado el: 23 de mayo de 2016.] http://www.alegsa.com.ar/Dic/sincronizar%20datos.php.

Android. 2014. Android Studio Overview. *Android Developers.* [En línea] Google, 2014. [Citado el: 16 de noviembre de 2015.] http://developer.android.com/tools/studio/index.html.

Android (2). 2014. What is Android? *Android Developer.* [En línea] Google, 2014. [Citado el: 15 de noviembre de 2015.] http://developer.android.com.

Borda, Fabricio D. 2012. *Una Herramienta para la Sincronización de Datos en Redes Móviles.* San Luis, Argentina : SeDICI, Universidad Nacional de San Luis, 2012.

Buchman, David. 2002. *SyncML (Synchronization Markup Language) and its Java Implementation sync4j - Diploma Thesis in Informatics.* Friburgo, Suiza : Universidad de Friburgo, 2002.

Byous, Jon. 2003. Java technology: The early years. *java.sum.com.* [En línea] 12 de abril de 2003. [Citado el: 3 de diciembre de 2015.] http://java.sun.com/features/1998/05/birthday.html.

Cantillo Valero, Carmen; Roura Redondo, Margarita; Sánchez Palacín, Ana;. 2012. *Tendencias actuales en el uso de dispositivos móviles en educación.* Buenos Aires, Argentina : s.n., 2012, La Educación digital magazine, Vol. 147, págs. 1-21. ISSN 0013-1059.

Davenport, Thomas H. y Prusak, Lawrence. 2001. *Conocimiento en acción. Cómo las organizaciones manejan lo que saben.* s.l. : Prentice Hall, 2001.

Firtman, Maximiliano. 2013. *Curso Básico Android.* Video2Brain, 2013.

García Carranco, Sergio Miguel y Contreras Mayén, Gabriel Rubén. 2014. *Tecnologías Móviles.* Salamanca : Ediciones Universidad de Salamanca, 2014.

Gonzáles Duque, Raúl. 2006. *Python para todos.* s.l. : Creative Commons Reconocimiento, 2006. Vol. 2.

Gutiérrez, Yandri. 2015. *GECMA: Aplicación para la gestión y apoyo al control del Proceso de Enseñanza Aprendizaje de la Matemática discreta.* Universidad de las Ciencias Informáticas. La Habana, Cuba : Repositorio Institucional, 2015. Tesis de grado.

Holovaty, Adrian y Kaplan-Moss, Jacob. 2009. *The Definitive Guide to Django: Web Development Done Right.* EEUU : Apress, 2009.

Hyers, Kent. 2014. *Global smartphone shipments reach a record 990 million units in 2013.* EEUU : Strategy Analytics, 2014, Vol. 27.

Imaginanet. 2015. Patron MVP. *imaginanet.com.* [En línea] diciembre de 2015. [Citado el: 12 de junio de 2016.] https://www.imaginanet.com/blog/patrón-mvp.html.

Islas, Claudia y Martinez, Evelio. 2008. *El uso de las TIC como apoyo a las actividades docentes.* Murcia : Revista RED, 2008, págs. 209-212.

Kanter, James Max y Veeramachaneni, Kalyan. 2015. *Deep Feature Synthesis: Towards Automating Data Science Endeavors.* Paris : IEEE DSAA, 2015. IEEE conference on Data Science and Advanced Analytics (DSAA). págs. 1-10.

Katrib Mora, Miguel y Sánchez, Lester. 2013. *Un primer análisis de los resultados de la aplicación de un enfoque colaborativo en la gestión del conocimiento en el proceso de enseñanza y aprendizaje.* La Habana : Universidad de la Habana, 2013, Compumat 2013.

Labedroid. 2015. *Manual de usuario de la aplicación Android Cuaderno del Profesor v. 2.3.* Madrid : Laboratorios LabeDroid, 2015.

Larman, Craig. 2003. *UML y Patrones. Una introducción al análisis y diseño orientado a objetos y al proceso unificado.* 2da Edición. Aragón : Prentice Hall, 2003.

Lazyloading. 2008. Patrones Observer. *WeboLogs.* [En línea] Lazyloading, 2008. [Citado el: 2 de febrero de 2016.] http://mgaravaglia.com.ar/blog/?p=32.

Lazyloading (2). 2008. Patrones Decorator. *WeboLog.* [En línea] Lazyloading, 2008. [Citado el: 2 de febrero de 2016.] http://mgaravaglia.com.ar/?p=30..

Lombardo, Enmanuela. 2015. GradeBook Pro - Grade, Attendance, and Behavior Tracking. *iTunes.* [En línea] Apple, 2015. [Citado el: 22 de noviembre de 2015.] https://itunes.apple.com/mx/app/gradebook-pro-grade-attendance/id393777614?mt=8.

López, Pedro. 2014. *Herramienta CASE Visual Paradigm.* Cantabria : Universidad de Cantabria, 2014.

Moreno, Ana Maria, Juristo, N. y Vegas, Sira. 2006. Técnicas de evaluación de software. *Ingenieria de software.* [En línea] 17 de octubre de 2006. [Citado el: 13 de marzo de 2016.] http://is.ls.fi.upm.es/udis/docencia/erdsi/Documentacion_Evaluacion?7.pdf.

Nonaka, Ikujiro y Takeuchi, Hirotaka. 1995. *The Knowledge-creating Company. How Japanese companies create the dynamics of innovations.* Nueva York : Oxford University Press, 1995.

Niss, Mogens. 2003. *Mathematical competencies and the learning of mathematics: The Danish KoMproject.* Atenas : Hellenic Mathematical Society, 2003. 3rd Mediterranean Conference on Mathematical Education. págs. 115-124.

Oecd Pisa. 2009. Assessment Framework - Key Competencies in Reading, Mathematics and Science. *OECD.org.* [En línea] noviembre de 2009. [Citado el: 13 de noviembre de 2015.] http://www.oecd.org/dataoecd/11/40/44455820.pdf.

OpenUp. 2012. OpenUp. *OpenUP.* [En línea] Eclipse.org, 2012. [Citado el: 2 de diciembre de 2015.] http://epf.eclipse.org.

Pabón Maestras, Juan. 2008. *Estructura de las Aplicaciones Orientadas a Objetos. El patrón Modelo-Vista-Controlador (MVC).* Universidad de Complutense, Madrid : Ediciones Complutense, 2008.

Peláez, Juan. 2009. *Arquitectura basada en capas.* Silicon Valley, San Fransisco : Microsoft Technologies, 2009, Vol. 29.

Pressman, Roger S. 2010. *Ingeniería de Software. Un enfoque práctico.* 7ma Edición. Madrid : McGraw-Hill, 2010.

Pinzón, Sonia y Guevara Bolaños, Juan Carlos. 2006. *La gestión, los procesos y las metodologías de desarrollo de software.* 4, Bogotá : e-Revistas, 2006, Vol. 2, págs. 82-100. ISSN 1794- 211X.

Ramirez Acosta, Elvira Patricia. 2010. Tecnologías Móviles. *TICSMIELCA.* [En línea] UNAM, Mexico D.F., 2010. [Citado el: 3 de diciembre de 2015.] https://sites.google.com/site/ticsmielca/home.

Realm. 2014. About us. *Realm.io.* [En línea] Realm, 2014. [Citado el: 13 de marzo de 2016.] https://realm.io.

Salazar, Alejandro. 2014. Tecnologías Móviles. *MIS.204.* [En línea] 2014. http://www.i.edu.mx/aportaciones/trabajo%20final_11.pdf.

Sánchez Diegues, Diego. 2015. *Modelo Vista Presentador (MVP).* 20, Sevila : Editorial de Ciencias de la Universidad Pablo de Olavide, 2015, MoleQla, pág. 7.

Sebesta, Robert W. y Mukherjee, Soumen. 2012. *Concepts of Programming Languages.* Nueva Jersey : Addison-Wesley, 2012.

Sheng, Liang. 2000. *Java Native Interface: Programmer's Guide and Specification.* Nueva Jersey : Addison-Wesley, 2000.

Softwarelibre. 2010. Entorno desconectado en VB.NET 2005. *Softwarelibre+Programación.* [En línea] 2010. [Citado el: 2 de febrero de 2016.] https://skrdz.wordpress.com/2010/06/25/entorno-desconectado-en-vb-net-2005/.

Sommerville, Ian. 2005. *Ingeniería de Software.* 7ma Edición. Nueva York : Pearson Education, 2005.

SQLite. 2016. About SQLite. *sqlite.org.* [En línea] sqlite.org, 2016. [Citado el: 2 de febrero de 2016.] https://www.sqlite.org/about.html.

Stackoverflow. 2014. Using the recyclerview with a database. *StackOverflow.* [En línea] Stack Exchange, 22 de Octubre de 2014. [Citado el: 13 de marzo de 2016.] http://stackoverflow.com/questions/26517855/using-the-recyclerview-with-a-database .

Suárez Hernández, Andry. 2009. *Arquitectura para Akademos 2.0.* Universidad de las Ciencias Informáticas. La Habana : Repositorio Institucional, 2009. Tesis de Grado.

UML. 2014. Unified Modeling Language (UML). *UML.* [En línea] 2014. [Citado el: 2 de diciembre de 2015.] http://UML.html.