UCLV
Universidad Central
"Marta Abreu" de Las Villas

MFC
Facultad de Matemática
Física y Computación

Departamento

Desarrollo de Aplicaciones para Dispositivos Móviles.

TRABAJO DE DIPLOMA

Título del trabajo: Aplicación móvil para el control de asistencia de profesores a clases.

Autores del trabajo: José Angel Sánchez Enríquez

Tutores del trabajo: Dr. Carlos E. García González
Lic. Geidy Pino Acosta

Santa Clara , Junio 2019

UCLV
Universidad Central
"Marta Abreu" de Las Villas

MFC
Facultad de Matemática
Física y Computación

Academic Departament:

Development Applications for Mobile Devices.

DIPLOMA THESIS

Title: Mobile Application for the attendance control of teachers to classes

Author: José Angel Sánchez Enríquez

Thesis Director: Dr. Carlos E. García González
Lic. Geidy Pino Acosta

Santa Clara , June, 2019

Pensamiento

"Las oportunidades grandes nacen

de haber sabido aprovechar las pequeñas".

Bill Gates.

Dedicatoria

A toda mi familia, por darme el apoyo necesario,

en especial a mis padres (Yolanda Enríquez y José Ramón Sánchez), por el interés demostrado en mi formación he logrado culminar mis estudios superiores,

A ese grupo de informática donde recibí tanto apoyo de todos.

Agradecimiento

A toda aquella persona que ha contribuido a mi formación como profesional.

Al claustro de profesores de la Facultad MFC por su apoyo y profesionalidad.

A mis tutores Carlos García y Geidy Pino por su apoyo en cada momento.

A José Antonio López, gracias por estar siempre en los momentos donde tuve que necesitar tu ayuda y tenerte como mi mejor amigo, para no decir mi hermano.

A mis compañeros de aula, gracias por el apoyo que ustedes me brindaron.

En fin, a todas las personas que de alguna forma u otra estuvieron involucrados y me apoyaron en la realización de este trabajo.

Resumen

En la Universidad Central “Marta Abreu” de las Villas (UCLV), el control de la asistencia de los profesores es una de las labores que se realiza para controlar el cumplimiento del horario docente. Este proceso es realizado de forma manual hasta el momento, lo cual hace que sea demasiado tedioso y sin ningún tipo de seguridad a la hora de recopilar esos datos. En el siguiente trabajo se propone como objetivo la creación de una aplicación móvil que permita controlar la asistencia de los profesores a clases, poder consultar información acerca del horario docente, así como los turnos en los cuales los profesores debieran estar impartiendo sus clases, todo esto pudiera consultarse estando offline. La principal fuente de alimentación es una base de datos, de esta forma se garantiza mayor facilidad y seguridad a la hora de controlar la asistencia.

Abstract

In the Central University "Marta Abreu" of Las Villas (UCLV), the control of the attendance of the professors is one of the tasks that is carried out to control the fulfillment of the teaching schedule. This process is done manually so far which makes it too tedious and without any security at the time of collecting that data. In the following work is proposed as the goal of creating a mobile application to control the attendance of teachers to classes, to see information about the teaching schedule, as well as the shifts in which teachers should be teaching their classes, all this could be consulted while offline. The main source of power is a database, thus ensuring greater ease and security when controlling attendance.

Tabla de Contenidos

Introducción

En la Universidad Central “Marta Abreu” de las Villas (UCLV), el control de la asistencia de los profesores es una de las acciones que realiza la institución para supervisar el cumplimiento del horario docente. Este proceso es realizado de forma manual, lo cual hace que sea demasiado tedioso y sin ningún tipo de seguridad a la hora de recopilar esos datos.

El proceso de control del cumplimiento del horario docente en la facultad de Matemática, Física y Computación (FMFC) de la UCLV, es ejecutado por una persona que recorre cada aula del edificio docente en cada uno de los turnos de clases de la sesión de la mañana y de la tarde. El proceso da como resultado un informe escrito en Microsoft Word, donde se especifican los turnos de clases en los cuales los profesores no asistieron; se elabora semanalmente y se envía por correo electrónico a los jefes de departamento para que estos realicen el análisis con sus profesores. Este informe no cuenta con una estructura bien definida y hace difícil su procesamiento.

Por esta razón se hace necesario cambiar este proceso, de manera que el procedimiento manual descrito anteriormente sea sustituido por una aplicación móvil que pueda ofrecer reportes fiables y útiles para los directivos de la facultad.

Por lo antes expuesto se plantea el siguiente **problema de investigación**: ¿Cómo facilitar el control de asistencia de los profesores en la facultad MFC?

Para resolver esta problemática, se propone como **objetivo general**: desarrollar una aplicación móvil que permita a los directivos de la facultad de Matemática, Física y Computación llevar el control de la asistencia de los profesores a clases.

Para darle cumplimiento al objetivo general, se proponen los siguientes **objetivos específicos**:

1. Realizar un estudio del proceso manual de control del cumplimiento del horario docente en la facultad de Matemática, Física y Computación (FMFC) de la UCLV.

2. Implementar una aplicación móvil que registre el cumplimiento del horario docente.
3. Generar reportes fiables y seguros que le permitan a los directivos de la facultad supervisar el cumplimiento del horario docente.
4. Diseñar una base de datos que almacene toda la información relacionada con el horario docente.
5. Evaluar el desempeño de la aplicación para comprobar su correcto funcionamiento.

El **valor práctico** del trabajo está dado en que la aplicación móvil dispone de medios para facilitar el uso del control de asistencia de los profesores de forma automática, inmediata y eficaz, lo cual eliminaría la forma manual en la que se está llevando este proceso.

El trabajo de diploma está estructurado en tres capítulos:

Capítulo 1. “Marco teórico referencial”. En esta sección se abordará la descripción del problema, el proceso del control de la asistencia de los profesores en la facultad, así como su análisis crítico del control actual, generalidades en el mundo moderno con respecto a este tema y sus tendencias y tecnologías actuales.

Capítulo 2. “Análisis, diseño e implementación de la aplicación móvil”, se describe los principales diagramas de la aplicación y algunos de los algoritmos y técnicas empleadas.

Capítulo 3. Presenta las diferentes pruebas de usabilidad y funcionamiento a las que se sometió la aplicación y los resultados que se obtuvieron de las mismas.

El documento termina con las conclusiones, recomendaciones y referencias bibliográficas.

Capítulo 1: Marco teórico referencial

Este capítulo es el respaldo teórico de los temas tratados en el informe, necesario para el correcto entendimiento de la solución planteada; se describen los conceptos fundamentales asociados al dominio del problema y el objeto de estudio, se presenta la fundamentación de las tecnologías utilizadas para el diseño de la aplicación y las propuestas para su implementación y desarrollo.

1.1 Descripción del problema

En la actualidad las tecnologías de la información están presente en casi todas las ramas del saber. Se ha adoptado una cultura que depende en gran parte, por no decir totalmente, del funcionamiento que estas ofrecen. Como consecuencia de esto, las instituciones deben evolucionar para mantenerse a la par de los constantes cambios que las Tecnologías de la Información y el Conocimiento (TIC) generan en el entorno, automatizando procesos industriales y requiriendo una infraestructura sólida para poder funcionar, así como personal capacitado que pueda prevenir riesgos y diseñar productos o servicios basados en las oportunidades que brinda la tecnología (Flores 2016).

Los sistemas de control de asistencia modernos se basan en tecnologías de identificación automáticas, utilizando, por ejemplo: códigos de barras, bandas magnéticas, tarjetas de proximidad por radio frecuencia (RFID) e incluso sistemas biométricos de huella digital. Siendo todos estos solamente una parte de la solución debido a que el componente principal es, fundamentalmente, el software de control de asistencia debido a que los datos capturados con los distintos modelos de lectores necesitan ser procesados para recién entonces llegar a convertirse en información (tardanzas, inasistencias, horas extras, etc.) (ROMERO 2011).

En la Facultad de Matemática, Física y Computación (FMFC) de la Universidad Central “Marta Abreu” de Las Villas, no se cuenta con un sistema automatizado de control de asistencia del personal. Actualmente se hace el control del cumplimiento del

horario docente utilizando un proceso manual, en el cual una persona recorre cada aula del edificio docente en cada uno de los turnos de clases de la sesión de la mañana y de la tarde.

Dicho informe debe ser enviado a la vicedecana docente de la facultad, la cual elabora un informe más completo. Este informe (ver Fig. 1.1), además de ser un mecanismo de control para la facultad, se envía por correo electrónico a los jefes de departamento para que estos realicen análisis con sus profesores.

Carrera: Ingeniería Informática

Semana de clase	Turnos de clases programados	Turnos de clases perdidos	De estos cuantos por falta de fluido eléctrica	Turnos de clases recuperados	De estos cuantos de facultad	De estos cuantos servicios
Semana 10 3ro II	14	1	0	0	1	0
Semana 10 4to II	16	1	0	0	1	0
total						

Incidencias Fundamentales:
Un turno de Android en 3ro II
Un turno de Optimización en 4to II
Optimización – Juan Pérez

Fig. 1.1 Reporte de asistencia mediante una tabla en Word

Entre las dificultades que se presentan al contar con un proceso manual de este tipo se pueden mencionar:

1. El informe no cuenta con una estructura bien definida, lo cual hace difícil su procesamiento.
2. Las horas trabajadas no coinciden con las horas de clases.

3. Todo este proceso manual termina convirtiéndose en una tarea tediosa y lenta a la hora de realizar consultas y obtener reportes de asistencia.

1.1.1 Análisis crítico del control actual de asistencia de profesores

En la actualidad el control de asistencia en la universidad es uno de los temas pendiente a solucionar, ya que puede darse el caso de que un profesor exprese que hoy no puede ir a clases, porque tiene un acto o algo relacionado con su investigación; cuando esto pasa se le termina dando la asistencia y por consiguiente esos alumnos pierden sus clases. Otro caso es que pueden decir que tienen cualquier evento, sin embargo, abusan de su asistencia para priorizar otras actividades sobre sus obligaciones con la enseñanza. Ante estas situaciones puntuales, se reconoce que una hoja para dar la asistencia no es suficiente para vigilar la presencia de esos profesores en las aulas. La mayoría de las universidades carecen de sistemas objetivos de control del horario que midan con precisión la entrada y salida de quienes imparten clases.

En varias ocasiones se ha planteado la necesidad de sustituir los procedimientos actuales**,** basado en la rúbrica en un papel por otros más avanzados, que permitan escrutar el absentismo docente con exactitud y evitar excesos de confianza como los denunciados por algunos universitarios, aunque la mayoría de los profesores son cumplidores de sus deberes.

Esta dificultad se determinó como resultado de una encuesta anónima aplicada a diferentes estudiantes de la facultad de Ciencias Sociales (CS).

1.2 Generalidades de los sistemas de control de asistencia modernos

Básicamente, un sistema de control de asistencia universitario es todo aquel con el que se supervisa mediante un horario docente a un profesor, con el objetivo de evaluar

las incidencias que se puedan producir: la puntualidad, la inasistencia a los turnos de clases, etc.

En el momento de elegir un sistema de control de asistencia, hay que tener en cuenta todas las posibilidades disponibles hoy día en el mercado, sus pros y sus contras, siempre teniendo en cuenta que los sistemas modernos de control de asistencia ofrecen beneficios adicionales sobre los sistemas manuales tradicionales (MARTÍNEZ 2017):

1. Un procesamiento más rápido de los empleados ya que la asistencia puede grabarse con sólo un toque o una verificación rápida.
2. Prevención de fraudes mediante la eliminación de registros duplicados y falsos.
3. Ahorra tiempo, ya que la asistencia puede integrarse directamente con un sistema de nómina o puede producir un informe que se puede descargar o imprimir.
4. Mejora la puntualidad y reduce las pausas largas y el absentismo.

Entre los sistemas de control de asistencia más utilizados se encuentran los siguientes, (MARTÍNEZ 2017):

1. Sistemas de control mediante tarjeta: estos sistemas son seguramente los más utilizados y su característica común es la presencia de bandas magnéticas, códigos de barras, etc. La información se recoge a través de un lector habilitado para ello en el lugar de acceso a la empresa o al puesto.
2. Tarjetas con banda magnética: su funcionamiento es simple, cada empleado deberá pasar su tarjeta por el lector tanto a la entrada como a la salida. La tarjeta cuenta con una banda magnética, que puede ser leída por un dispositivo electrónico y es ahí donde está registrada la información del empleado en cada caso.
3. Tarjetas de proximidad: este tipo de tarjetas utiliza la tecnología RFID (siglas de Radio Frequency IDentification, en español, identificación por radiofrecuencia). Una tarjeta de proximidad es una tarjeta plástica que lleva incrustada en su

núcleo un circuito integrado y una antena de comunicación. Cada ejemplar contiene un número de serie que es leído por frecuencia de radio, típicamente de 125kHz. Las tarjetas de proximidad son capaces de transmitir cuando un código único, de más de 1 billón de combinaciones posibles, es acercado al lector adecuado. Su durabilidad es superior a las de banda ya que la lectura se produce sin contacto físico, por lo que el deterioro es mínimo.

4. Sistemas de control biométricos: son sistemas muy sofisticados y muy interesantes porque eliminan los fraudes más comunes. Son el resultado de la aplicación de técnicas matemáticas y estadísticas sobre los rasgos físicos o de conducta de una persona para verificar identidades o identificar a personas. Como ejemplo de características físicas estarían las huellas dactilares, los patrones faciales, las retinas, el iris o la geometría de la palma de la mano. Su principal desventaja es su alto precio de implantación, debido al coste de la tecnología que utilizan.

5. Identificación por huellas dactilares: funciona por reconocimiento de la huella digital de alguno de los dedos de la mano. El sistema de identificación automatizada de huellas dactilares, tiene un índice de seguridad del 99.9%.

6. Identificación por biometría vascular: lee el patrón de las venas de las manos. La tecnología más fiable y segura dentro del campo de la biometría de la mano es la que consta de un escáner que captura una imagen del tramado de las venas de la palma de la mano a través del reflejo de ondas de frecuencia corta (muy similares a los infrarrojos).

7. Identificación por biometría facial: el reconocimiento por biometría facial es una tecnología que requiere de unas condiciones muy concretas, sobre todo de luz, por lo que su rendimiento puede verse afectado por circunstancias ajenas a las personas.

8. Identificación por escáner de iris u ojos: este sistema es tecnológicamente muy seguro, pero actualmente es también una técnica bastante cara. Suele utilizarse para proteger recintos que requieren de un elevado índice de seguridad.

9. Biometría de perfil de mano: funciona por reconocimiento de la morfología de la mano.

10. Otros sistemas de control: otra alternativa para controlar y supervisar el horario, consiste en recurrir a sistemas, no tan avanzados tecnológicamente, pero sí más al alcance de los bolsillos, como el siguiente:

11. Time trackers: son programas informáticos que miden cuando arrancamos y apagamos nuestro ordenador. Obviamente, este sistema solamente sería válido para puestos que requieran del uso de un ordenador. Estos time trackers permiten llevar un seguimiento de cuánto tiempo se ha invertido en cada tarea y facilita mucho más el día a día. No todos los sistemas pueden proporcionar tan alto nivel de detalle. Workmeter, por ejemplo, sí que ofrece todos estos datos, mientras que otras alternativas solo pueden proporcionar los horarios de entrada y salida.

1.3 Tendencias y tecnologías actuales

Las aplicaciones móviles en la actualidad tienen un gran uso en la sociedad, no solo las que ofrecen servicios o productos; también están incorporadas a los procesos internos de las instituciones y empresas sin importar su tamaño o sector por la inmediatez que proporcionan las aplicaciones y el gran impacto de esta tecnología en la sociedad.

En el mundo actual las medianas empresas y las grandes corporaciones iniciaron una transformación basada en las nuevas innovaciones tecnológicas. El primer protagonista de este cambio fue Internet, y unas décadas después las Aplicaciones Móviles o Apps.

Los dispositivos móviles forman parte de la vida cotidiana y cada vez se vuelven más sofisticados. Su poder de cómputo genera posibilidades que unos años atrás eran impensables (Cuello and Vittone 2013).

Una aplicación móvil o app es una aplicación informática diseñada para ser ejecutada en teléfonos inteligentes (*smartphones*), tabletas u otros dispositivos móviles, que permiten al usuario efectuar una tarea concreta de cualquier tipo: profesional, ocio, educativa, de acceso a servicios, etc., facilitando las gestiones o actividades a desarrollar (Santiago, Trabaldo et al. 2015).

Por lo general se encuentran disponibles a través de plataformas de distribución, operadas por las compañías propietarias de los sistemas operativos móviles como Android, iOS, BlackBerry OS, Windows Mobile, entre otros. Existen aplicaciones móviles gratuitas y otras de pago (Wolfram and Schilling 2015).

En la actualidad se encuentran todo tipo de aplicaciones de diferentes formas o color, funcionamiento o productividad; su funcionamiento y recursos se encaminan en una serie de ventajas tales como: un acceso más rápido y sencillo a la información necesaria, un almacenamiento de datos personales que es de manera segura, una gran versatilidad en cuanto a su utilización o aplicación práctica; la atribución de funcionalidades específicas, así como mejoras en la capacidad de conectividad y disponibilidad de servicios y productos. Simultáneamente se mejoran las herramientas que tenían los diseñadores y programadores para desarrollar apps, facilitando la tarea de realizar una aplicación y llevarla al mercado.

Al ser aplicaciones que se vinculan en los dispositivos, estas están escritas en algún lenguaje de programación compilado. Es importante destacar que una app no es una aplicación web, tampoco es un sistema operativo, ni un servicio de alojamiento informático o web (Tracy 2012).

Los servicios WEB en el mundo actual juegan un rol importante dentro de la programación, ya que no son más que un conjunto de aplicaciones o de tecnologías con capacidad para interactuar en la Web. Estas intercambian datos entre sí con el objetivo de ofrecer servicios. Los proveedores ofrecen sus servicios como procedimientos remotos y los usuarios los solicitan llamando a estos procedimientos a través de la Web y a su vez proporcionan mecanismos de comunicación estándares entre diferentes aplicaciones, que interactúan entre sí para presentar información dinámica al usuario. Para proporcionar interoperabilidad y extensibilidad entre estas

aplicaciones, y que al mismo tiempo sea posible su combinación para realizar operaciones complejas, es necesaria una arquitectura de referencia estándar.

Los Servicios Web pueden ser muy útiles en ciertos casos concretos de programación. Los expertos de marketing de empresas de software como Microsoft anuncian a bombo y platillo una revolución debido a la aparición de esta tecnología.

La única revolución que se puede ver en relación con los Servicios Web es la de cómo los servidores de Internet hablan entre ellos, cada día son más independientes del programador y eso es algo que va a pasar completamente sin cuidado para el resto de los usuarios de Internet. Si se logra que existan Servicios Web de utilidad, gratuitos y sencillos, este nuevo esquema de comunicación y programación tendrá futuro.

1.3.1 Fundamentación de la metodología utilizada

Las Metodologías de Desarrollo de Software surgen ante la necesidad de utilizar una serie de procedimientos, técnicas, herramientas y soporte documental a la hora de desarrollar un producto de software.

Dichas metodologías pretenden guiar a los desarrolladores al crear un nuevo software, pero los requisitos de un software a otro son tan variados y cambiantes, que ha dado lugar a que exista una gran variedad de metodologías para la creación del software (Araujo 2011).

Para el desarrollo de la aplicación se analizaron diversas metodologías ágiles, como Scrum, eXreme programming y Mobile-D.Esta última fue la escogida para la realización de este trabajo, por ser la que más se adecua a los requisitos para el desarrollo de la herramienta.

Mobile-D fue especialmente creada para el desarrollo de dispositivos móviles. Su estructura se apoya en otras metodologías ya existentes, como son eXtreme Programming (XP), Crystal Methodologies y Rational Unified Process (RUP) (Blanco 2009).

En el desarrollo con Mobile-D el ciclo de vida del proyecto consta de cinco fases:

Exploración, inicialización, producción, estabilización y prueba del sistema. Todas las fases con excepción de la primera cuentan con tres días de desarrollo distintos: planificación, trabajo y liberación.

Fig. 1.2 Fases de Mobile D

1.3.2 Fundamentación del entorno de desarrollo y tecnologías utilizadas

Los sistemas operativos móviles han pasado por un largo camino de cambios, altibajos, aprobación y desaprobación, amplio alcance y muchas historias que afectaron su desarrollo. De entre todas las opciones, Android ha logrado llegar a la mayor cantidad de activaciones de dispositivos que utilizan este sistema operativo. Por esta razón se escogió esta plataforma para que llegue la aplicación de Control de Asistencia a Profesores (**CAP**) a la mayor cantidad de dispositivos móviles para su uso.

Hoy en día existen varios softwares donde podemos programar para Android, el recomendado por Google que es el creador de Android es Android Studio, este teniendo en cuenta que tiene un gran repositorio y mejoras realmente considerables y necesarias, nos da la posibilidad de ver cómo se vería nuestra aplicación en diferentes resoluciones automáticamente, desde el mismo IDE y sin necesidad de compilar y

probar en dispositivos múltiples, sencillamente con el marcado de XML y los datos de muestra.

Para poder darle un buen desarrollo a la aplicación se utilizan varias herramientas, las cuales nos proporcionan un mejor rendimiento y funcionamiento de ella. Como plataforma de desarrollo se escoge Android Studio acompañado de Java, para la base de datos se escoge MySQL y SQLite, este último ligado al ORM *ActiveAndroid* para facilitar el trabajo con la Base de Datos local en SQLite. También se utilizan servicios WEB mediante una API, lo cual permite el enlace entre ambas bases de datos.

Se está trabajando en Android 4.4 KitKat, que es la plataforma mínima hoy en día para programar con el objetivo de que la app pueda funcionar en la mayoría de los dispositivos. Se está usando la versión 3.1.0 de ActiveAndroid con APIs 19, que es uno de los ORM más usado en Android KitKat.

Android. Sistema Operativo para dispositivos móviles

Android es un paquete de software para dispositivos móviles que incluyen un sistema operativo, *middleware* y aplicaciones. Los componentes de Android están diseñados en forma de paquete o *stack*, las aplicaciones conforman la capa superior del paquete, mientras que un *kernel* de Linux conforman las demás capas. Las aplicaciones que conforman el sistema operativo son para realizar las operaciones básicas como revisar el correo electrónico, enviar SMS, mapas, navegador web, contactos, entre otras. Todas las aplicaciones están escritas en el lenguaje de programación Java, por tanto es en el que se desarrollan todas las aplicaciones para este sistema.

El *framework* de desarrollo de aplicaciones tiene el mismo acceso al mismo *framework* con que se desarrollan las aplicaciones por defecto del sistema, su arquitectura está diseñada para simplificar y reutilizar componentes; las capacidades de cualquier aplicación pueden ser publicadas para ser utilizadas por cualquier otra aplicación (sujeto a requerimientos de seguridad establecidos por el *framework*). Este mismo mecanismo permite a los componentes ser reemplazados por el usuario. Por ejemplo, si se tiene una pequeña aplicación para almacenar notas y deseas buscar una

ubicación específica cuya localización acabas de escribir, puedes utilizar las funcionalidades de la aplicación de mapas dentro de la misma aplicación en vez de considerar cambiar a la aplicación de mapas y buscar la ubicación manualmente. Android incluye un set de bibliotecas **C/C++** que son utilizadas por varios componentes del sistema operativo. Estas funcionalidades están a disposición a través del *framework* de desarrollo de aplicaciones (Developers 2011).

Android Studio

Android se ha convertido en los últimos años no solo en el sistema operativo con mayor número de usuarios, sino también el que tiene más aplicaciones y una mayor cantidad de dispositivos; cifra que, además, va en aumento (Developers 2011).

Android Studio es la herramienta que ofrece Google a los desarrolladores para programar aplicaciones para su popular sistema operativo. Una herramienta que nos permite la implementación de elementos fundamentales como el acceso a APIs, la posibilidad de agregar dependencias, de gestionar y administrar archivos

Java

Java Development Kit (**JDK**) es el kit de desarrollo oficial del lenguaje de programación Java. Además de la máquina virtual de Java, indispensable para ejecutar los programas, cuenta con diversas herramientas: como *javac*, el compilador de bytecode de Java, *javap*, el desensamblador de clases, y *jdb*, el depurador de bugs. Estas herramientas se encuentran en el subdirectorio *bin* del **JDK**. Además de estas herramientas, el **JDK** contiene ejemplos y demostraciones (Guilermo 2011).

Java sirve para crear aplicaciones y procesos en una gran diversidad de dispositivos. Se basa en la programación orientada a objetos, permite ejecutar un mismo programa en diversos sistemas operativos y ejecutar el código en sistemas remotos de manera segura.

Su ámbito de aplicación es tan amplio que se utiliza tanto en aplicaciones de escritorio, móviles e incluso en equipos electrodomésticos. Muchos programadores

también utilizan este lenguaje para crear pequeñas aplicaciones que se insertan en el código HTML de una página para que pueda ser ejecutada desde un navegador.

MySQL

MySQL es un sistema de administración de bases de datos relacional (RDBMS). Se trata de un programa capaz de almacenar una enorme cantidad de datos de gran variedad y de distribuirlos para cubrir las necesidades de cualquier tipo de organización, desde pequeños establecimientos comerciales a grandes empresas y organismos administrativos. MySQL compite con sistemas RDBMS propietarios conocidos, como Oracle, SQL Server y DB2.

MySQL incluye todos los primeros elementos necesarios para instalar el programa, preparar diferentes niveles de acceso de usuario, administrar el sistema y proteger y hacer volcados de datos. Puede desarrollar sus propias aplicaciones de base de datos en la mayor parte de los primeros lenguajes de programación utilizados en la actualidad y ejecutarlos en casi todos los primeros sistemas operativos, incluyendo algunos de primeros que probablemente no ha oído nunca hablar. MySQL utiliza el lenguaje de consulta estructurado SQL. Se trata del lenguaje utilizado por todas las bases relacionales. Este lenguaje permite crear bases de datos, así como agregar, manipular y recuperar datos en función de criterios específicos (Gilfillan 2011).

SQLite

SQLite es una librería compacta y auto contenida de código abierto y distribuida bajo dominio público que implementa un gestor de bases de datos SQL embebido, sin configuración y transaccional (Montiel 2015).

Los usuarios más conocidos que la utilizan actualmente en sus aplicaciones son: Android, Adobe, Mozilla, Google, McAfee, Microsoft, Philips y Toshiba, entre otros.

El uso de bases de datos en Android se basa en el motor libre SQLite.

ActiveAndroid

ActiveAndroid es un ORM de estilo de registro activo (mapeado relacional de objetos).

ActiveAndroid te permite guardar y recuperar registros de bases de datos SQLite sin tener que escribir una sola instrucción SQL. Cada registro de la base de datos se ajusta perfectamente en una clase con métodos como save() y delete().

Sin embargo, ActiveAndroid hace mucho más que esto. Acceder a la base de datos es una molestia, por decir lo menos en Android, ActiveAndroid se encarga de todas las tareas de configuración y desordenadas, y todo con solo unos simples pasos de configuración (Schmidt 2012).

APIs

Una API detalla solamente la forma de llamar a cada función y la tarea que esta desempeña, sin importar cómo se lleva a cabo dicha tarea. Puede combinar recuperación de errores, traducción de datos, seguridad, manejo de colas y nomenclatura, capaz de comprender acciones y comandos simples, pero con muchas opciones. Para invocarla, el programa debe llamar a una función tipo *"POST"* o *"GET"*, especificando parámetros para el nombre de destino, indicadores de datos y opciones de confirmación, esta toma la información y hace que todo el trabajo específico de comunicación sea transparente para la aplicación. Las APIs pueden desarrollarse para cualquier plataforma y sistema operativo o para todos estos sistemas al mismo tiempo. También desde hace unos años se vienen desarrollando como forma de acceder a servicios web.

1.3.3 Algoritmo de seguridad para la aplicación móvil

HMACSHA1 es un tipo de algoritmo de hash con clave que se construye a partir de la función de hash SHA1 y se utiliza como un HMAC, o un código de autenticación de mensaje basado en hash. El proceso HMAC mezcla una clave secreta con los datos del mensaje, hace un hash del resultado con la función hash, mezcla de nuevo ese

valor hash con la clave secreta, y luego aplica la función hash por segunda vez. El hash de salida es de 160 bits de longitud.

La fuerza del algoritmo de hash es importante, pero no es tan importante en las funciones de derivación clave. Es poco probable que incluso si SHA-1 se rompe, eso influiría en la seguridad de PBKDF2. Es mejor usar SHA-1 y aumentar el recuento de iteraciones hasta un nivel que se ha ajustado para su configuración específica (nextcloud.com 2017).

Existen en la actualidad varios tipos de algoritmos de seguridad para aplicaciones móviles, entre los cuales se encuentran **MD5, SHA1, SHA2.** Se escoge SHA-1, porque este tiene un bloque de 160 bits en lugar de 128, bits lo que permite que la función sea más compleja y el número de pasos son de 80 con respecto a los otros tipos de seguridad, es decir, tiene mayor longitud, lo cual posibilita que su función de compresión sea también más compleja En general SHA-1 es más robusto y seguro. SHA1 es mejor que MD5, pero más malo que SH2. Sin embargo, implementar SH2 es computacionalmente más costoso que SH1 y no es adecuado para los dispositivos móviles, pues no todos pueden tener buenas propiedades de hardware para garantizar la fluidez de la aplicación y evitar congelamientos por procesamiento excesivo.

1.4 Conclusiones parciales

En este primer capítulo se ofreció una visión general relacionado con la asistencia de los profesores en la universidad, su análisis crítico y los tipos de tecnologías usados en la actualidad para el control de la asistencia. También se abordaron algunas características de las herramientas utilizadas, teniendo en cuenta que el desarrollo de las tecnologías y los sistemas que las emplean se perfeccionan cada día; se decidió implementar la aplicación móvil con el uso de Android Studio como ambiente o herramienta de desarrollo y Java como lenguaje de programación, para almacenar la información, se escoge SQLite porque es el estándar de BD local para Android y MySQL como BD remota por ser una herramienta libre muy utilizada a nivel mundial entre los desarrolladores, estas dos utilizando los servicios WEB para la sincronización entre ellas.

Capítulo 2: Análisis, diseño e implementación de la aplicación móvil

En el siguiente capítulo se explican un conjunto de fases para poder realizar el diseño y desarrollo de la herramienta, lo cual va a ser la solución a nuestra problemática, se describen las diferentes funcionalidades del software desarrollado, presentando algunos diagramas creados para entender el funcionamiento de algunas de las acciones ejecutadas por la app.

2.1 Requisitos y análisis

En esta fase se analizan las peticiones o requerimientos de las personas o entidad para la cual se desarrolla el servicio móvil “Cliente”. El propósito de los mismos es definir las características del mundo o entorno de la aplicación. Se realizan tres tareas: obtener requerimientos, clasificar los requerimientos y personalizar el servicio (Mantilla, Ariza et al. 2014).

Para que se manifieste los síntomas del problema o necesidades que se pretenden solucionar con las tecnologías móviles, o simplemente, para que señale las características que debe tener la aplicación, se sugiere hacer una cadena de entrevistas al cliente para obtener requerimientos. Una vez obtenidos estos requerimientos se procede a su clasificación los cuales son: entorno, mundo, funcionales y no funcionales; estos dos últimos son los que se utilizarán como medio de desarrollo.

2.1.1 Requisitos funcionales

Los requisitos funcionales definen el comportamiento interno del software, son declaraciones de los servicios que debe proporcionar el sistema, especifican la manera en que éste debe reaccionar a determinadas entradas y cómo debe comportarse ante

situaciones particulares, pueden declarar explícitamente lo que el sistema no debe hacer.

Rf 1. Registrarse en la app.

Rf 2. Establecer los roles de trabajo.

Rf 3. Gestionar (agregar o modificar) el contenido.

Rf 4. Gestionar el horario docente por carreras.

Rf 5. Consultar el horario docente por carreras.

Rf 6. Realizar control de asistencia a turnos de clases.

Rf 7. Generar un reporte de incidencias por semana.

2.1.2 Requisitos no funcionales

Rnf 1. Smartphone o Tablet con SO Android 4.4 (KitKat) o superior.

Rnf 2. 10 MB de almacenamiento interno libre como mínimo.

Rnf 3. La aplicación debe estar conectado a la red para algunas funcionalidades.

Rnf 4. La aplicación debe ser fácil de instalar.

Rnf 5. La aplicación debe mantener los datos seguros y protegidos.

2.1.3 Diagrama de casos de uso

En la aplicación existen tres roles diferentes de usuarios: se encuentra el **administrador**, el cual establecerá los roles de trabajo, podrá gestionar contenido, gestionar y consultar horario, generar reporte; el **jefe de año** podrá gestionar horario y contenido, consultar el propio horario, generar un reporte para los directivos; mientras que el **chequeador** solo podrá consultar el horario y realizar el control de asistencia hacia los profesores, estos tres roles deberán registrarse en la aplicación.

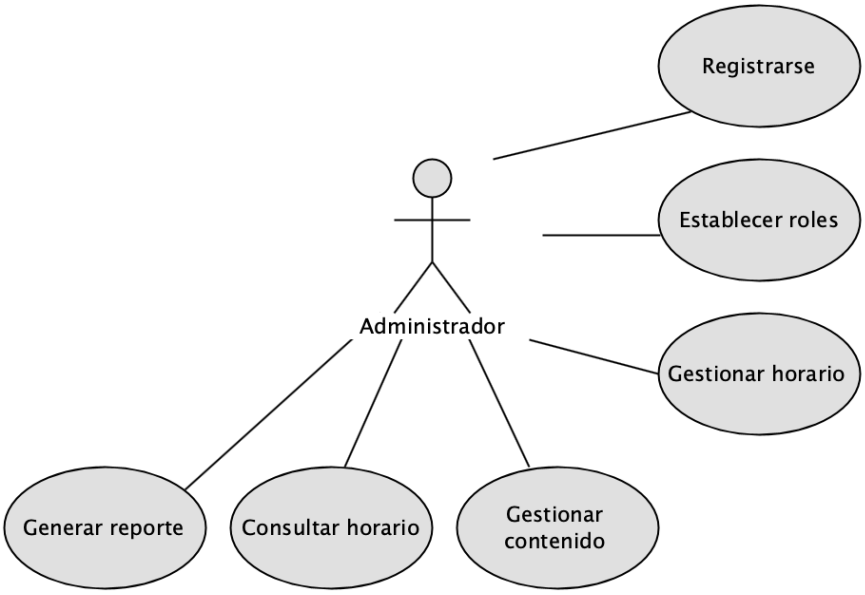


Fig. 2.1 Diagrama de Casos de uso Administrador

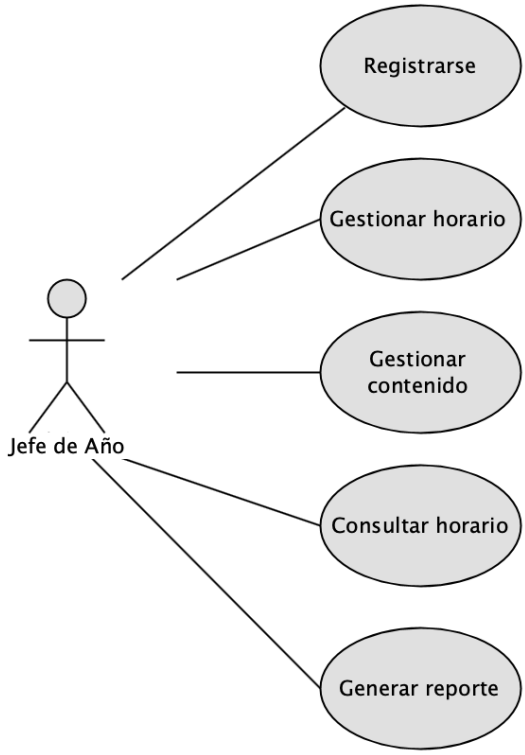


Fig. 2.2 Diagrama de Casos de uso Jefe de Año

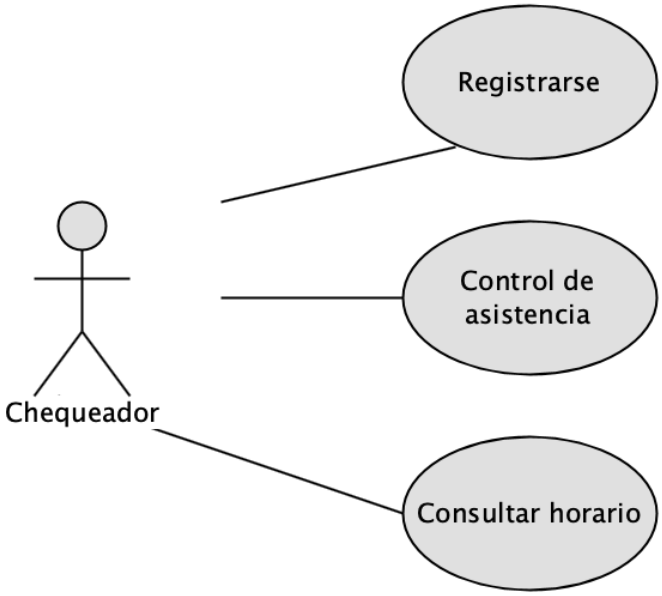


Fig. 2.3 Diagrama de Casos de uso Chequeador

2.1.4 Diagrama de actividades

Los diagramas de actividades, permiten una representación del orden en que se ejecutan las actividades correspondientes a diversos elementos productivos (personas, máquinas, entre otros). El estudio de este tipo de diagramas, podrá dirigirnos hacia una programación más adecuada de todos los elementos, y con ello lograr que se reduzcan los tiempos muertos o, que se puedan eliminar operaciones redundantes (Quesada, Alonso et al. 1996).

En las siguientes figuras (Fig. 2.4, Fig. 2.5 y Fig. 2.6) se muestran cómo se desarrollan los procesos de actividades de login, chequear asistencia y gestionar contenido.

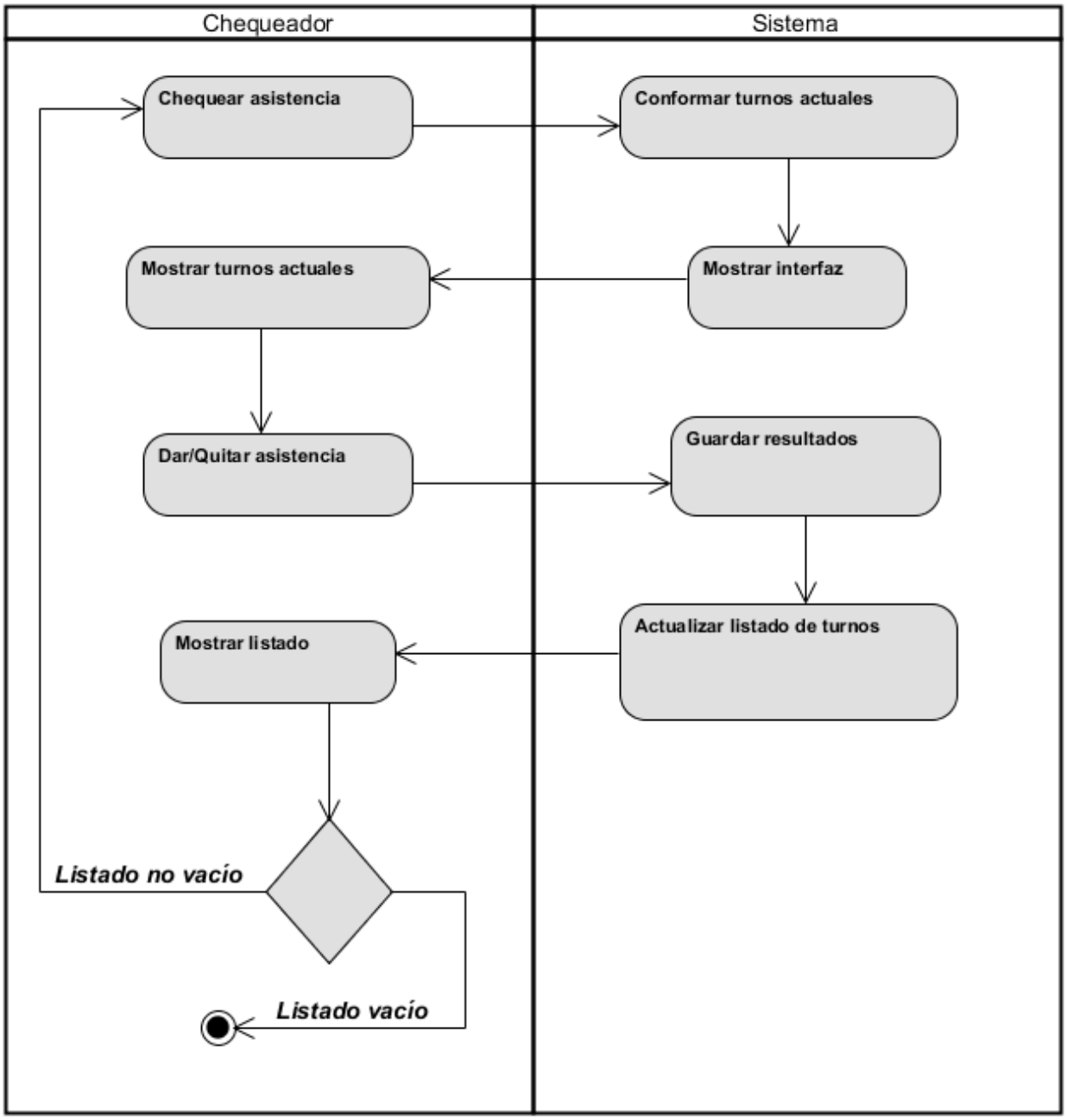


Fig. 2.4 Diagrama de actividad Chequear asistencia

El chequeador inicia el proceso chequear asistencia, el sistema verifica si hay turnos conformados y mediante una interfaz muestra al chequeador los turnos conformados actuales. El chequeador pasa y tiene la opción de dar o quitar asistencia, el sistema guarda los resultados, actualiza el listado de turnos y muestra nuevamente ese listado. Si ese listado ya está vacío se finaliza el proceso, si no se encuentra vacío vuelve a iniciarse el proceso.

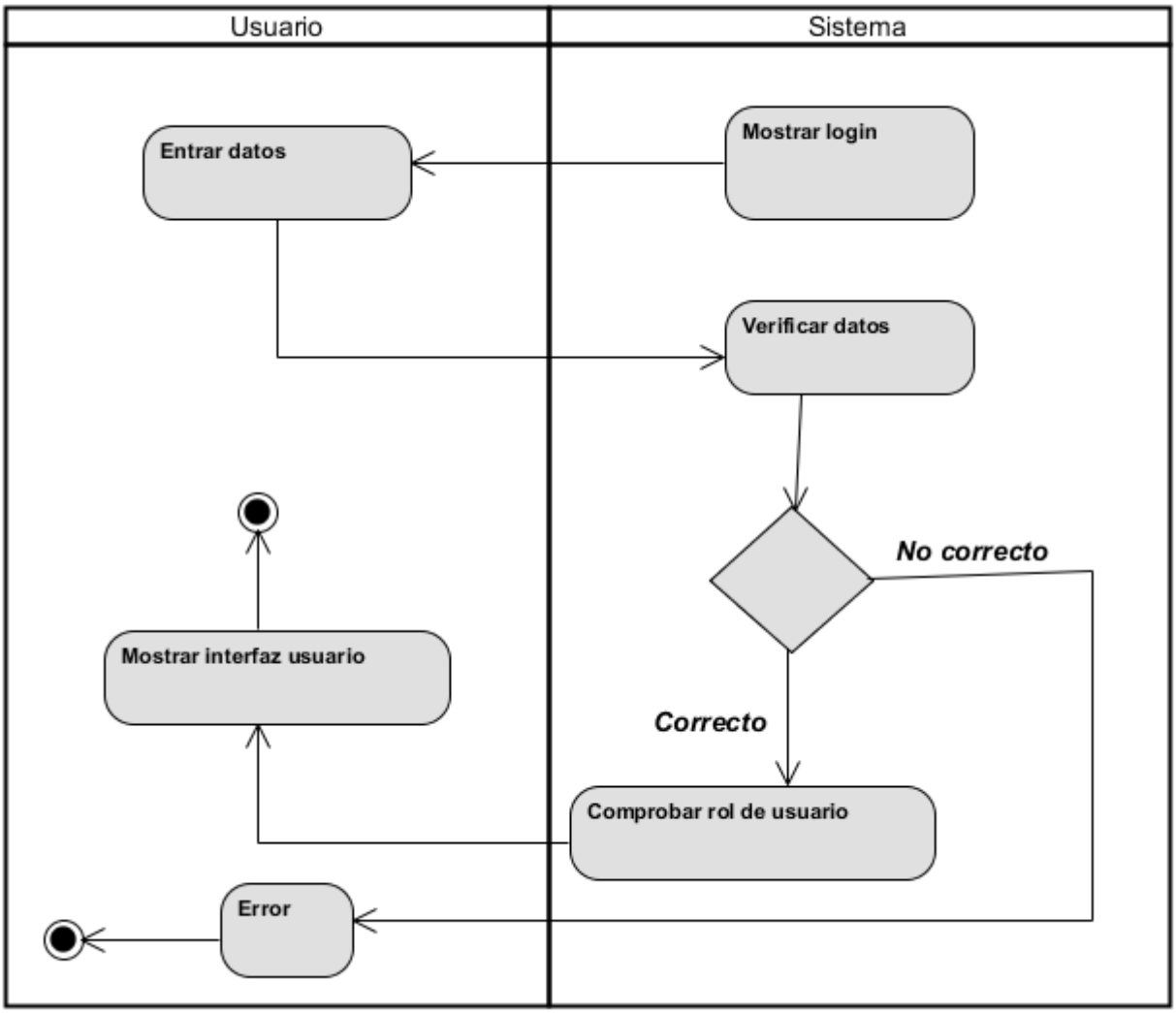


Fig. 2.5 Diagrama de actividad Login

El diagrama de actividad Login comienza cuando el sistema le muestra una interfaz de autentificación al usuario; este entra los datos y el sistema tiene la obligación de verificar esos datos, si los datos son correctos comprueba el rol de usuario y muestra el tipo de interfaz de usuario en el caso de ser administrador, chequeador o jefe de año; si los datos pasados por el usuario son incorrectos se muestra un cartel de usuario o contraseña incorrectos.

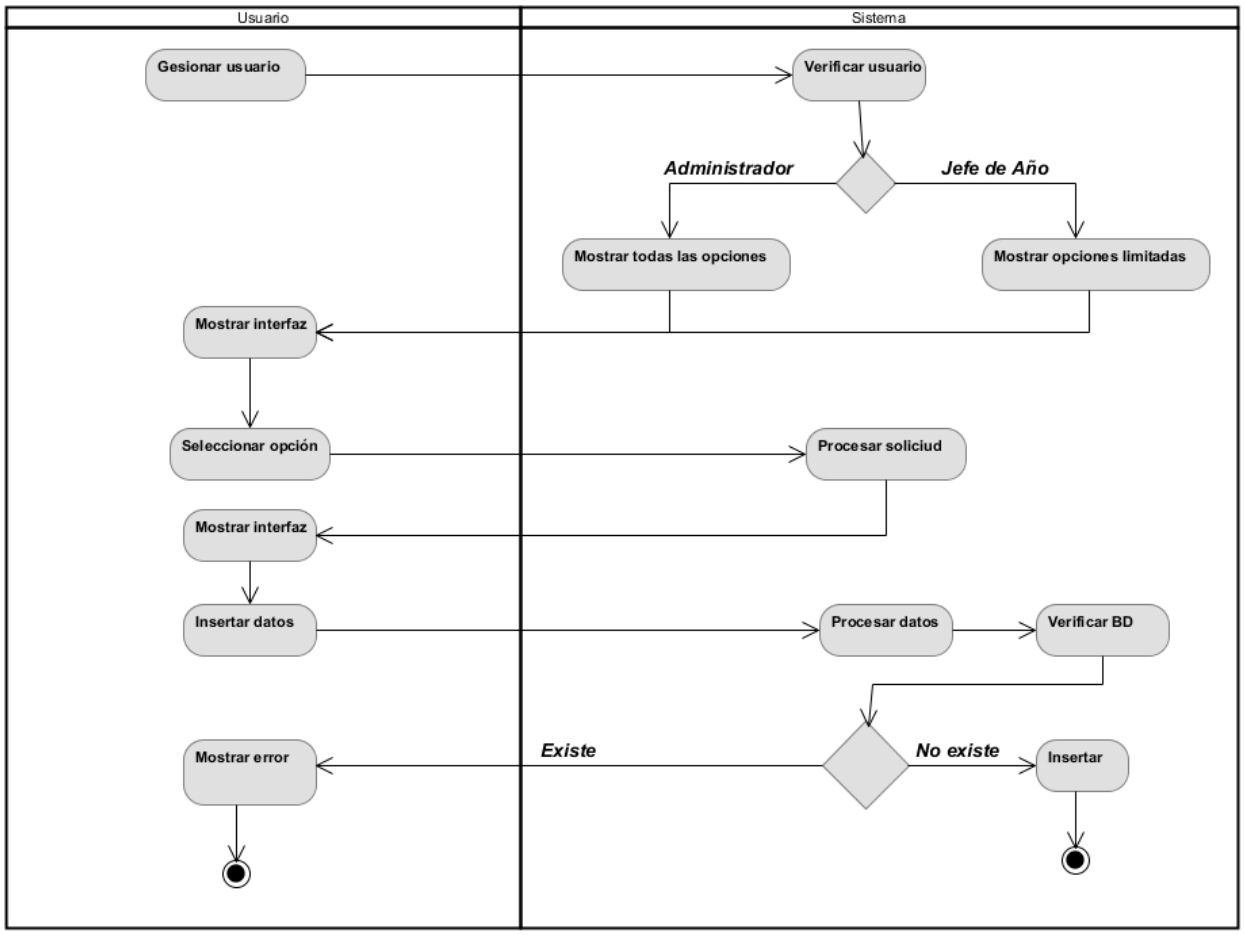


Fig. 2.6 Diagrama de actividad Gestionar contenido

Para poder gestionar contenido el usuario mediante la aplicación móvil selecciona esta opción, el sistema verifica el tipo de rol de usuario, si es Administrador muestra todas las opciones y si es Jefe de año muestra las opciones predestinadas a él, mediante una interfaz el usuario selecciona la opción mientras que el sistema procesa la solicitud enviada, este vuelve a mostrar una interfaz para poder insertar los datos, estos datos son procesados y verificados en la base de datos, si esos datos existen se le muestra al usuario un error, de lo contrario se insertan los datos enviados.

2.1.5 Diagrama de paquetes

Un diagrama de paquetes muestra cómo un sistema está dividido en agrupaciones lógicas mostrando las dependencias entre esas agrupaciones. Se puede decir que

existe dependencia entre varios paquetes cuando un elemento de un paquete requiere de otro elemento que pertenece a un paquete distinto (Sommerville 2005).

En la siguiente figura (Fig. 2.7) se muestran los paquetes desarrollados en la aplicación.

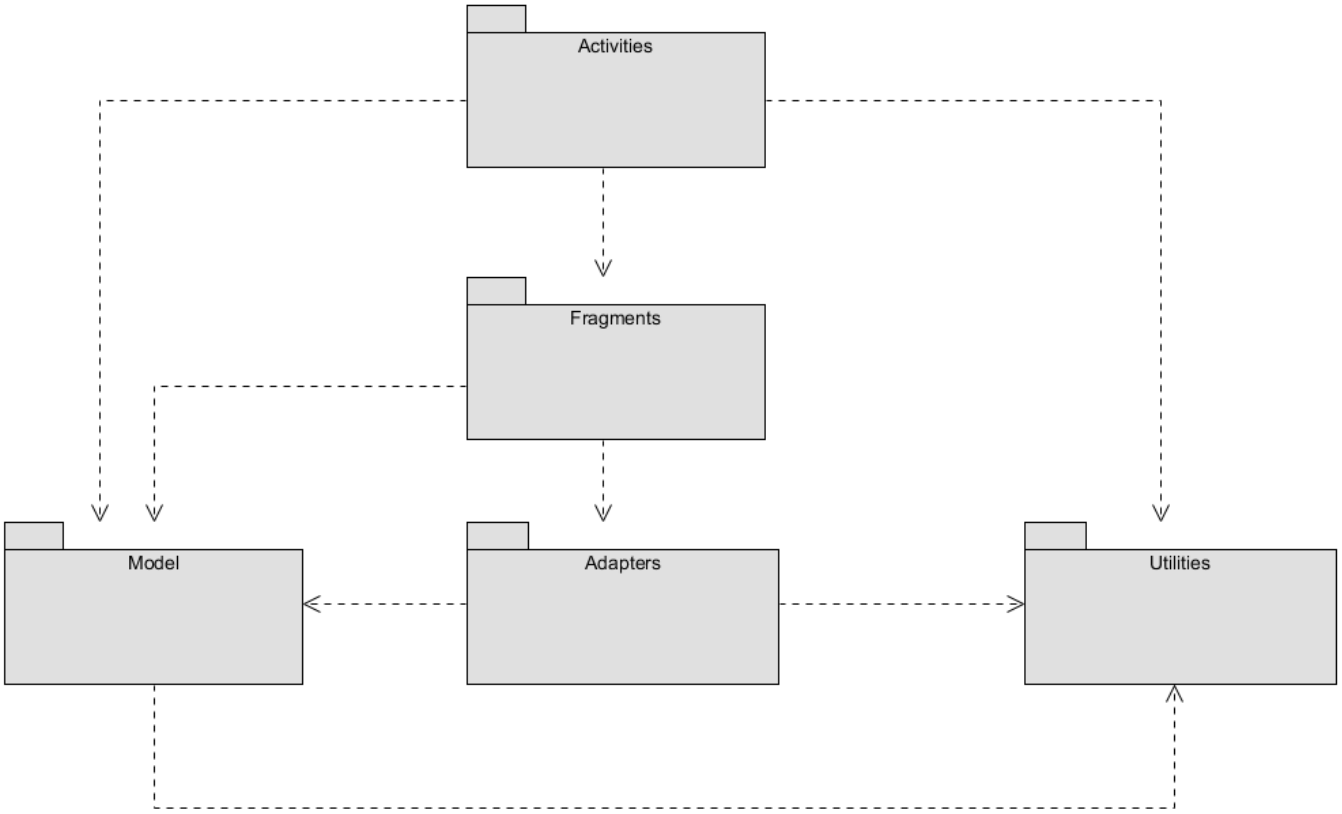


Fig. 2.7 Diagrama de Paquetes

2.1.6 Diagrama de despliegue

Un diagrama de despliegue modela la arquitectura en tiempo de ejecución de un sistema, esto muestra la configuración de los elementos de hardware (nodos) y muestra cómo los elementos y artefactos del software se trazan en esos nodos. Un nodo es un elemento de hardware o software.

En la figura (Fig. 2.8) se aprecia como un usuario mediante un interfaz móvil puede acceder a una base de datos local, esta acompañada de un manejador de base de datos establece una conexión con el servidor mediante un servicio Web para poder

tener acceso a la base de datos montada en el servidor para guardar los datos insertados por la interfaz móvil y a su vez poder también descargar los últimos datos en caso de que la base de datos en el móvil sea nueva.

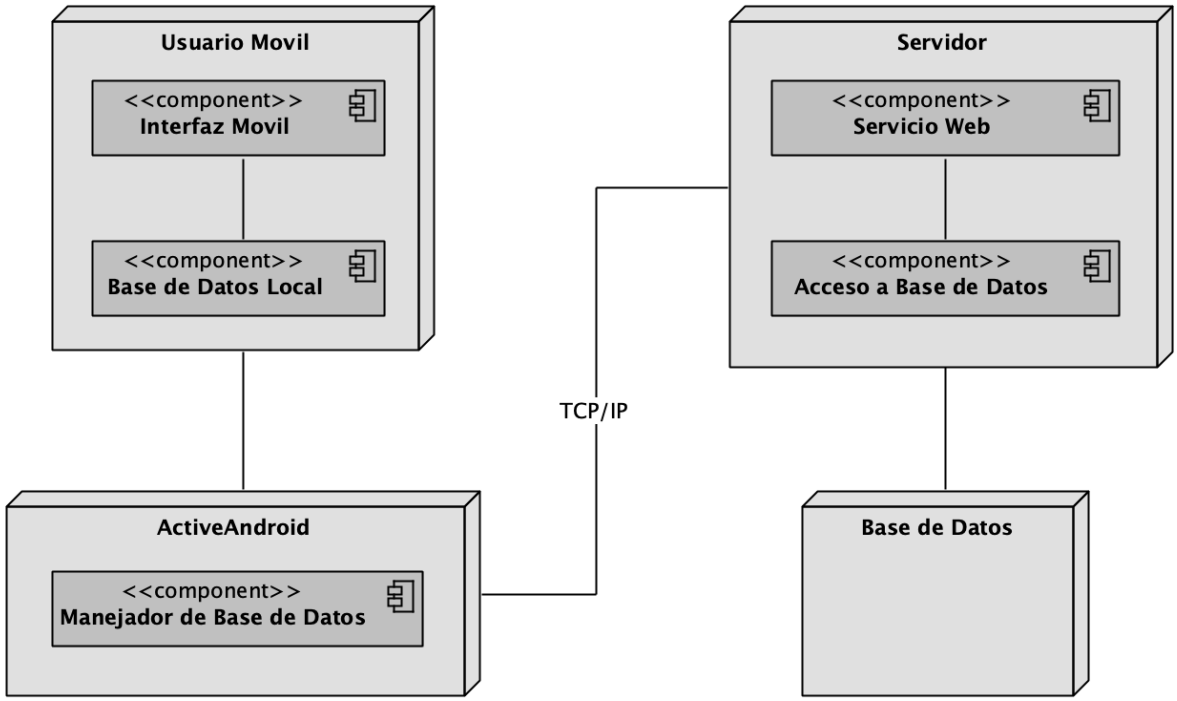


Fig. 2.8 Diagrama de Despliegue

2.2 Diseño e implementación

Para desarrollar una aplicación en Android se utiliza el patrón Modelo Vista Controlador (MVC), este modelo surge con el objetivo de reducir el esfuerzo de programación necesario en la implementación de sistemas múltiples y sincronizados de los mismos datos, a partir de estandarizar el diseño de las aplicaciones. El patrón MVC es un paradigma que divide las partes que conforman una aplicación en el Modelo, las Vistas y los Controladores, permitiendo la implementación por separado de cada elemento, garantizando así la actualización y mantenimiento del software de forma sencilla y en un reducido espacio de tiempo. A partir del uso de *frameworks* basados en

el patrón MVC se puede lograr una mejor organización del trabajo y mayor especialización de los desarrolladores y diseñadores (González and Romero 2012).

En la actualidad, en cualquier lugar del mundo los que desarrollan aplicaciones informáticas centran su atención en dos aspectos fundamentales: cómo lograr construir mejores aplicaciones en menos tiempo, y cómo utilizar mayor cantidad de estándares en el diseño de las aplicaciones que permitan mayor reutilización del código y mejores mantenimientos de los sistemas desarrollados. Teniendo en cuenta el creciente uso de la programación orientada a objeto en la concepción e implementación de este tipo de aplicaciones, y la gran actualidad que tiene el uso de patrones internacionalmente aceptados (González and Romero 2012).

2.2.1 Arquitectura del sistema

El diseño modular permite a los diseñadores y a los desarrolladores trabajar conjuntamente y hacer cambios en una parte de la aplicación sin que las demás se vean afectadas. Desde el punto de vista del usuario, todo comienza en el controlador.

Según Catalani, este modelo de arquitectura presenta varias ventajas (Catalani 2007):

1. Separación clara entre los componentes de un programa; lo cual permite su implementación por separado.
2. Interfaz de Programación de Aplicaciones API (*Application Programming Interface*) muy bien definida; cualquiera que use el API podrá reemplazar el Modelo, la Vista o el Controlador, sin aparente dificultad.
3. Conexión entre el Modelo y sus Vistas dinámicas; se produce en tiempo de ejecución.

Al desarrollar una aplicación para dispositivos móviles utilizando Android Studio, el Modelo queda conformado por el conjunto de clases que se utilizan para el correcto funcionamiento de ActiveAndroid para interactuar con la BD. La Vista está integrada por las clases que representan las pantallas que muestran información al usuario, en el

caso de programación para Android estas están definidas por archivos XML que son los encargados de generar una interfaz gráfica. Como Controlador están las clases en Java asociadas con las vistas XML, son las que realizan la parte funcional de la aplicación, se manipulan todos los eventos que ocurren en la misma y se controlan todos los componentes visuales para que se comporten de manera adecuada. Pueden definirse e implementarse otras clases controladoras con el fin de realizar otros procedimientos.

2.2.2 Diagrama de clases

Por otra parte, el diagrama de clases de un diseño incluye información como: clases, asociaciones, atributos, operaciones, constantes, métodos, navegabilidad y dependencias. Según Jacobson, estos diagramas describen la realización física de los casos de uso, sirven de abstracción en la implementación del sistema y son utilizados como una entrada fundamental de las actividades de implementación (Jacobson, Booch et al. 1997).

En la figura (Fig. 2.9) se muestra de forma general como está compuesto el diagrama de clases de la aplicación en los distintos paquetes que la conforman.

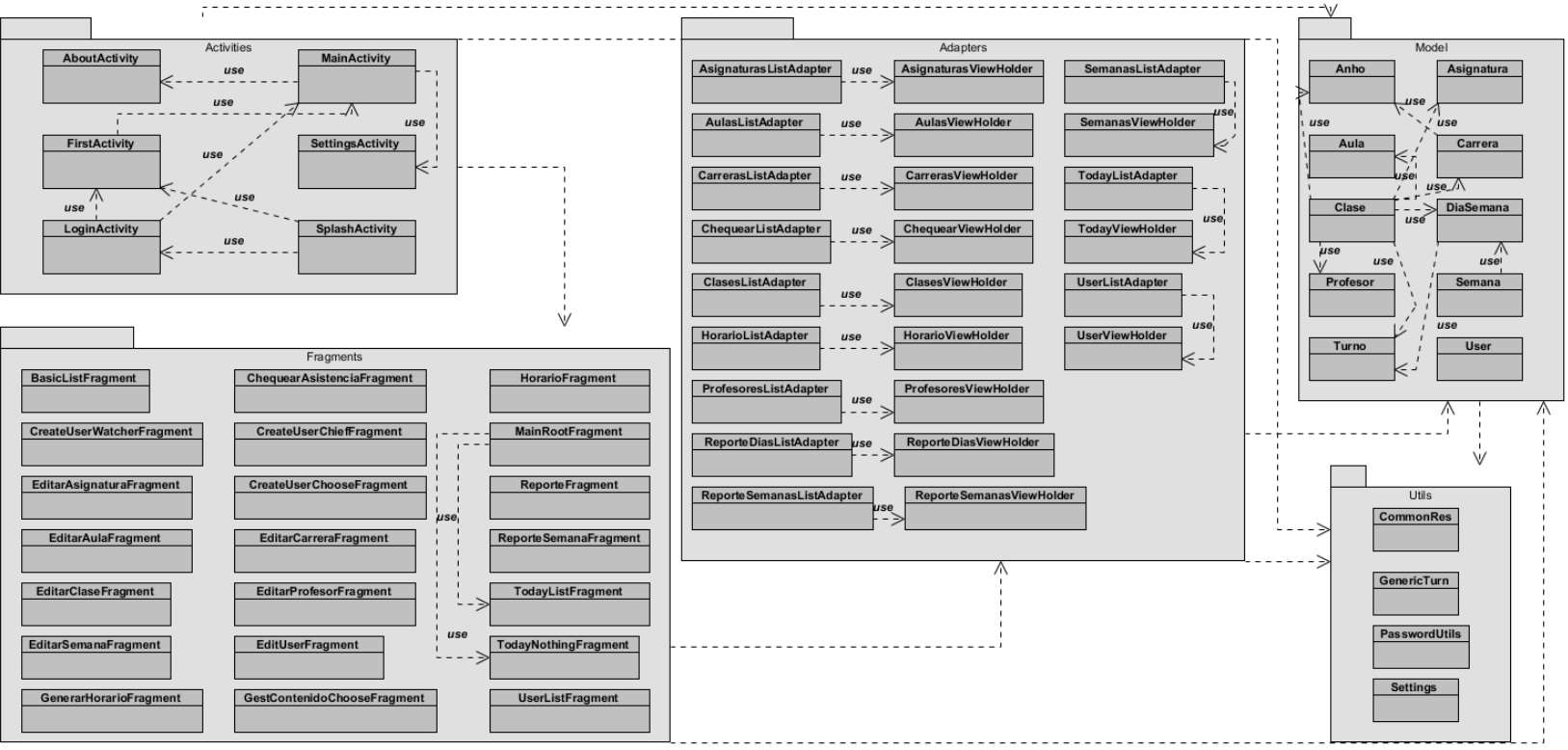


Fig. 2.9 Diagrama de Clases

En el paquete **Model** se encuentran las clases encargadas del funcionamiento de ActiveAndroid para el manejo de la base de datos. El paquete **Utils** contiene clases auxiliares que en su mayoría poseen atributos y métodos estáticos que son utilizados por otras clases en la aplicación. El paquete **Adapters** contiene los adaptadores y los *ViewHolder*, que son los encargados de mostrar la información en las listas de forma personalizada mejorando la experiencia del usuario. En el paquete **Activities** se encuentran las actividades principales de la aplicación, son las encargadas de navegar a otras vistas o la navegación entre fragmentos. El paquete **Fragments** contiene los distintos fragmentos que serán cargados en las actividades para ejecutar las diferentes funcionalidades.

2.2.3 Diagrama de secuencia

Los diagramas de secuencia forman una parte esencial de la investigación para interpretar un software. Estos muestran gráficamente los eventos que fluyen de los actores de la herramienta (Larman 2003).

En la figura (Fig. 2.10) se muestra el orden secuencial de los eventos que ocurren al ejecutar la aplicación por primera vez y si existen usuarios registrados en el servidor.

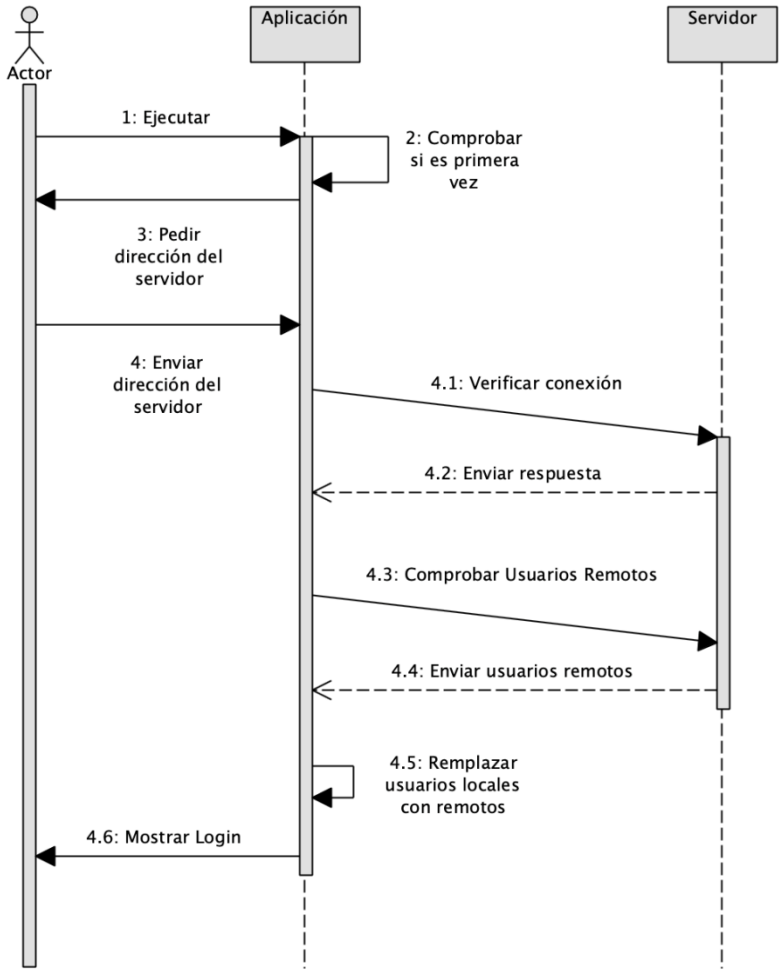


Fig. 2.10 Diagrama de secuencia

2.2.4 Mapa de navegación

El mapa de navegación muestra cada una de las pantallas o *Activities* de la aplicación y desde que lugar se puede acceder a cada una de estas.

Fig. 2.11 Mapa de Navegación

2.2.5 Diagrama de componentes

El diagrama de componentes muestra las organizaciones y dependencias lógicas entre componentes software, sean estos componentes de código fuente, binarios o ejecutables. Desde el punto de vista del diagrama de componentes se tienen en consideración los requisitos relacionados con la facilidad de desarrollo, la gestión del

software, la reutilización, y las restricciones impuestas por los lenguajes de programación y las herramientas utilizadas en el desarrollo. Los elementos de modelado dentro de un diagrama de componentes serán componentes y paquetes (Sommerville 2005).

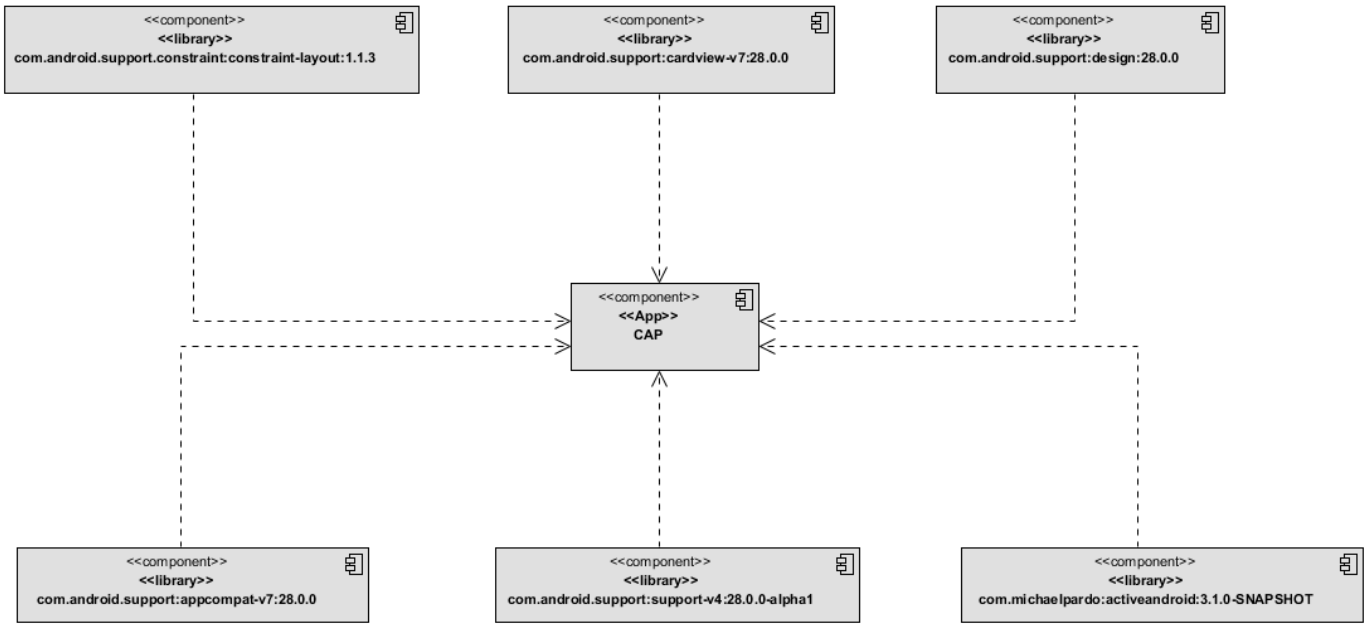


Fig. 2.12 Diagrama de Componentes

2.2.6 Trabajo con ActiveAndroid

Para poder trabajar con el ORM ActiveAndroid lo primero que debemos entender para poder facilitar su proceso es: ¿qué es el mapeo objeto-relacional (ORM)?

En un nivel alto, el concepto de mapeo objeto-relacional es muy sencillo, los programadores de Java, están preparados para pensar en términos orientados a objetos (OO). Una definición básica de ORM es que es "una técnica para convertir entre sistemas de tipos incompatibles en un lenguaje de programación orientado a objeto". En nuestro caso, estamos tratando de convertir entre los tipos de objetos que usamos para modelar nuestra aplicación y los tipos básicos permitidos en SQLite (INTEGER, TEXT, REAL, NONE).(Carson 2016). En otras palabras, el ORM traduce el objeto Java en una tabla SQL.

Ahí es donde entra en juego **ActiveAndroid**. ActiveAndroid traduce su modelo de dominio de Objeto Java de alto nivel en una base de datos SQLite usando anotaciones simples. Permite que los gráficos de objetos complejos se guarden y consulten fácilmente, y no tiene que escribir una sola instrucción SQL. Algunos conocimientos de SQL pueden ser útiles para adaptar las consultas y realizar migraciones de esquemas.

Una advertencia final: de ninguna manera es ActiveAndroid la única herramienta ORM para la base de datos disponible en Android, y en algunos casos ni siquiera es la mejor para usar. Es simplemente una herramienta que está bastante bien documentada, es fácil de aprender y fácil de usar. Otras herramientas incluyen **Room**, **Realm** y **OrmLite**. Realm es una biblioteca interesante porque está escrita en C++ y construida específicamente para aplicaciones móviles. Esto significa que evita algunos de los gastos generales que SQLite trae a la imagen. OrmLite es una herramienta ORM de Java (no específica de Android), que utiliza anotaciones de la misma manera que lo hace ActiveAndroid, se escoge el ORM ActiveAndroid porque trabaja en Android 4.4 KitKat, lo cual nos facilita la posibilidad de que en la mayoría de los dispositivos Android pueda ejecutarse la aplicación.

A continuación, se muestra cómo funciona ActiveAndroid y como se relacionan dos tablas de la base de datos SQLite. En este caso es una relación de uno a muchos en **Carrera** y **Anho** respectivamente.

```
@Table(name = "Anhos")
public class Anho extends Model {

    @Column(name = "anho")
    public String anho;

    @Column(name = "carrera",
            onDelete = Column.ForeignKeyAction.CASCADE,
            onUpdate = Column.ForeignKeyAction.CASCADE)
    public Carrera carrera;

    public Anho() {
        super();
    }

    public Anho(String anho, Carrera carrera) {
        super();
        this.anho = anho;
        this.carrera = carrera;
        save();
    }

    public static List<Anho> getAll() {
```

```
            return new Select()
                    .from(Anho.class)
                    .orderBy("carrera")
                    .execute();
        }

        public static Anho getAnho(String anho, Carrera carrera) {
            return new Select()
                    .from(Anho.class)
                    .where("anho = ?", anho)
                    .where("carrera = ?", carrera.getId())
                    .executeSingle();
        }

        public static String[] getStringArray() {
            String[] result = {"1", "2", "3", "4", "5"};
            return result;
        }

        public String getLiteral() {
            String result = anho;
            if (result.equals("1")) {
                result = result.concat("ro");
            } else if (result.equals("2")) {
                result = result.concat("do");
            } else if (result.equals("3")) {
                result = result.concat("ro");
            } else
                result = result.concat("to");
            return result;
        }
    }


    @Table(name = "Carreras")
    public class Carrera extends Model {

        @Column(name = "nombre")
        public String nombre;

        @Column(name = "tipo")
        public String tipo;

        public static final String NOMBRE = "nombre";
        public static final String TIPO = "tipo";

        public Carrera() {
            super();
        }

        public Carrera(String nombre, String tipo, String anhos) {
            super();
            this.nombre = nombre;
            this.tipo = tipo;
            int a = Integer.parseInt(anhos);
            save();
            for (int i = 0; i < a; i++) {
                new Anho(String.format("%d", i + 1), this);
            }
        }

        public Carrera(String nombre, String tipo) {
            super();
            this.nombre = nombre;
            this.tipo = tipo;
            int a = 5;
            save();
            for (int i = 0; i < a; i++) {
                new Anho(String.format("%d", i + 1), this);
            }
        }

        public static Carrera getExactly(String nombre, String tipo) {
            return new Select()
                    .from(Carrera.class)
```

```
                .where("nombre = ?", nombre)
                .and("tipo = ?", tipo)
                .executeSingle();
    }


    public static List<Carrera> getAll() {
        return new Select()
                .from(Carrera.class)
                .orderBy("nombre")
                .execute();
    }

    public static Carrera getCarrera(String carrera) {
        return new Select()
                .from(Carrera.class)
                .where("nombre = ?", carrera)
                .executeSingle();
    }

    public List<Anho> anhos() {
        return getMany(Anho.class, "carrera");
    }

    public static String[] getStringArray() {
        List<Carrera> all = getAll();
        String[] result = new String[all.size()];
        for (int i = 0; i < all.size(); i++) {
            result[i] = all.get(i).nombre;
        }
        return result;
    }

    public String[] getAnhosStringArray() {
        List<Anho> anhos = anhos();
        String[] array = new String[anhos.size()];
        for (int i = 1; i <= anhos.size(); i++) {
            array[i] = i + "";
        }
        return array;
    }
}
```

2.3 Conclusiones parciales

En este capítulo se realizó un estudio detallado de la aplicación móvil, además de las funcionalidades y requerimientos que debe cumplir dicha aplicación. Se describió la estructura lógica del sistema para una mejor comprensión de su funcionamiento y se determinaron los casos de uso donde entra cada rol de usuario. Se realizaron un conjunto de diagramas especificando los más significativos, ofreciendo una visión general del problema y de lo que será el producto, estableciendo la base fundamental para la realización del análisis y el diseño.

Capítulo 3: Descripción de las pruebas realizadas

Una vez terminado el software, es de gran importancia verificar y evaluar su funcionamiento y calidad. Debe asesorarse si el software cumple con los requisitos definidos y generen los resultados esperados por los usuarios. Para cumplir el objetivo es necesario ejecutar la aplicación en diversos dispositivos y con diferentes usuarios. También se deben realizar un conjunto de pruebas funcionales para la satisfacción total del usuario. Para realizar la entrega del producto terminado antes deben llevarse a cabo una serie de pruebas que ayudarán al proceso de corrección de errores y así mejorar la interacción con el usuario final.

3.1 Prueba unitaria

Cuando se construye una aplicación móvil, es de vital importancia comprobar que el código fuente escrito funcione correctamente y se adapte al funcionamiento permanente del programa. Una prueba unitaria es una forma de comprobar el correcto funcionamiento de una unidad de código, esto sirve para asegurar que cada unidad funcione correctamente y eficientemente por separado. Además de verificar que el código hace lo que debería hacer, verificamos que sea correcto los tipos de los parámetros y el tipo de lo que se devuelve. Si el estado inicial es válido, entonces el estado final es válido también.

En la aplicación las pruebas unitarias se realizaron en el paquete **utils**, a la clase **CommonRes**. En esta clase se realizaron pruebas unitarias a dos operaciones en específico **getFullMonth** y **isValidEmail**.

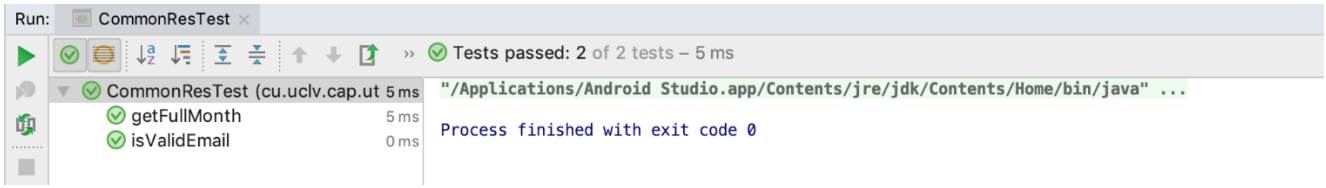
El método *getFullMonth* recibe un **int** como parámetro y devuelve el correspondiente mes en formato literal (**String**), teniendo en cuenta que los meses se encuentran en un rango de 0 a 11.

```java
@Test
public void getFullMonth() {
    int numero = 0;
    String mes = "Enero";

    assertEquals(
            String.format("P. Unitaria: verificar que el mes %s se corresponda a %s", numero, mes),
            mes,
            CommonRes.getFullMonth(numero)
    );

    numero = 1;
    mes = "Febrero";
    assertEquals(
            String.format("P. Unitaria: verificar que el mes %s se corresponda a %s", numero, mes),
            mes,
            CommonRes.getFullMonth(numero)
    );
```

El método *isValidEmail* recibe como parámetro una cadena de caracteres o String y verifica que sea una dirección de correo válida para la UCLV. Para ello el correo debe estar terminado en uclv.cu o uclv.edu.cu después de la @. En este caso solo verificaremos si es válida, no si existe la dirección, ya que para comprobar la existencia de la dirección necesitamos funcionalidades que están en un entorno distinto al de las pruebas por tanto no es posible comprobar con pruebas unitarias.

```java
@Test
public void isValidEmail() {
    String correo = "frank26@uclv.cu";
    assertTrue(
            String.format("P. Unitaria: Verificar si el correo (%s) es una dirección válida de la uclv", correo),
            CommonRes.isValidEmail(correo)
    );

    correo = "ernesto@uclv.com";
    assertFalse(
            String.format("P. Unitaria: Verificar si el correo (%s) es una dirección válida de la uclv", correo),
            CommonRes.isValidEmail(correo)
    );
```

Los resultados obtenidos de las pruebas unitarias fueron satisfactorios, ejecutando todas las pruebas con tiempos entre 0 y 30 milisegundos. Donde una prueba cuenta con 5 casos y la otra con 12.



3.2 Prueba de validación

Este proceso tiene como objetivo determinar si la aplicación móvil cumple los objetivos para los que se construyó el producto (Pressman and Troya 1988).

La aplicación tiene como objetivo controlar la asistencia de los profesores a las aulas y generar un reporte de incidencias. El chequeador pasa por las aulas y marca si el profesor se encuentra impartiendo clases, este contenido es guardado en la base de datos del móvil y una vez conectado sube esos datos para que el jefe de año o el administrador puedan generar un reporte de incidencias y poder observar si esa persona asistió o no al aula, también se podrá gestionar cierto contenido para generar un horario por semana. Se conecta a una API que se encarga de subir y descargar los datos desde el servidor para sincronizar las bases de datos local y remota. Los datos se pueden consultar de forma offline aunque no garantiza que sean los más actualizados

3.3 Prueba de compatibilidad

Aseguran que las aplicaciones móviles funcionan como se pretende, con los dispositivos seleccionados, con diferentes tamaños de pantalla, resolución y versión de sistema operativo. Pueden ser de inspección manual o automatizada mediante la captura de pantallas que son posteriormente revisadas por los *testers(Díaz 2013).*

La aplicación está diseñada para funcionar en cualquier *Smartphone* o *Tablet* con sistema operativo SO Android 4.4 (KitKat) o superior, por tanto, se realizaron pruebas en distintos dispositivos, estos dispositivos se muestran en la siguiente tabla.

	**Nombre del dispositivo:** Xiaomi Mi A2 lite **Fecha de lanzamiento:** julio 2018 **CPU:** Octa-core 2.0 GHz Cortex-A53 **RAM:** 3Gb **Almacenamiento interno:** 32Gb **Android:** Android 8.1(Oreo) MIUI 10
	**Nombre del dispositivo:** Samsung Express 2 **Fecha de lanzamiento:** octubre 2013 **CPU:** Dual-core 1.7 GHz Krait **RAM:** 1.5Gb **Almacenamiento interno:** 8Gb **Android:** Android 4.2.2 (Jelly Bean)
	**Nombre del dispositivo:** LG G5 **Fecha de lanzamiento:** abril 2016 **CPU:** Quad-core (2x2.15 GHz Kryo & 2x1.6 GHz Kryo) **RAM:** 4Gb **Almacenamiento interno:** 32Gb **Android:** Android 6.0.1 (Marshmallow)

	**Nombre del dispositivo:** Samsung Galaxy J7 **Fecha de lanzamiento:** junio 2015 **CPU:** Octa-core 1.5 GHz Cortex-A53 **RAM:** 1.5Gb **Almacenamiento interno:** 16Gb **Android:** Android 5.1 (Lollipop)
	**Nombre del dispositivo:** ZTE Blade Z Max **Fecha de lanzamiento:** agosto 2017 **CPU:** Octa-core 1.4 GHz Cortex-A53 **RAM:** 2Gb **Almacenamiento interno:** 32Gb **Android:** Android 7.1.1 (Nougat)
	**Nombre del dispositivo:** Motorola Moto E4 **Fecha de lanzamiento:** junio 2017 **CPU:** Quad-core 1.3 GHz Cortex-A53 **RAM:** 2Gb **Almacenamiento interno:** 16Gb **Android:** Android 7.1.1 (Nougat)

En los *Smartphones* probados se pudo valorar como la *app* se acomoda de manera automática a las variaciones de tamaño de pantalla y densidad de pixeles, además al

instalarse en los diferentes sistemas operativos la aplicación respondió exitosamente ya que en dispositivos con sistema operativo Android inferior a la versión 5.0 – *Lollipop*, se distingue un cambio liviano en el aspecto de los componentes, aunque no perturba directamente la estética de la *app*.

La aplicación se manifiesta de manera exitosa variando la forma en que se visualiza el contenido, para ajustarse a pantallas con mayor tamaño se utilizaron *layouts* diferentes con el objetivo de rendir las ventajas de tener más área para mostrar información en dispositivos más grandes.

3.4 Prueba de campo

La aplicación debe alcanzar la información mediante la base de datos creada en MySQL que se localiza en los servidores de la UCLV, la conectividad es uno de los retos con lo que se debe afrontar. Se realizaron pruebas de este tipo en distintos dispositivos en zonas diferentes de la facultad de FMFC, cubriendo áreas donde cobrase buena señal de la red inalámbrica (Wi-Fi) donde el trabajo de la app alcanzó los mejores resultados, el contenido era descargado de forma instantánea con tiempos de 1 a 3 segundos para descargar todo el contenido necesario. Sin embargo, en áreas con baja cobertura se pudo determinar que la información se tardaba más en ir apareciendo en las vistas correspondientes con tiempos de 6 a 10 segundos. En zonas de cobertura media los valores obtenidos fueron de 1 a 6 segundos aproximadamente.

3.5 Prueba de usabilidad

Las pruebas de usabilidad son un servicio de aseguramiento de calidad que reside en invitar a profesionales, cuyo perfil se adapta al de su público objetivo, probar el producto y proporcionar comentarios valiosos sobre su habilidad de uso y eficiencia. El objetivo principal de las pruebas de usabilidad es asemejar los problemas de usabilidad, cosechar comentarios oportunos y mejorar la satisfacción de sus usuarios.

Es beneficioso realizar pruebas de usabilidad al término del desarrollo de su producto. Idealmente, conviene no esperar demasiado antes de realizar una prueba de

usabilidad porque mientras más espera, más costoso y difícil será el tratamiento de los defectos identificados durante la sesión de *test*. A la misma vez, realizar pruebas de usabilidad demasiado pronto no tiene sentido porque los defectos y funcionalidades incompletas podrían molestar la experiencia del usuario. De hecho, se recomienda realizar pruebas funcionales (integración, validación, compatibilidad entre otras) antes de ejecutar pruebas de usabilidad.

Una de las pruebas de usabilidad más frecuentemente realizada, son las *heurísticas de Nielsen (Nielsen 1995)*

Nielsen, tras analizar varios artículos y libros plantea diez heurísticas que permiten evaluar la usabilidad de un producto *(Nielsen 1995)*

1. **Visibilidad del estado del sistema. El sistema siempre debe mantener a los usuarios informados sobre lo que está pasando, a través de una retroalimentación adecuada dentro de un tiempo razonable.**

En la aplicación se cumple ya que se muestra el estado de conexión en la esquina superior derecha de la pantalla principal, así como un indicador cuando la información se está actualizando.

2. **Correspondencia entre el sistema y el mundo real. El sistema debe hablar el idioma de los usuarios, con palabras, frases y conceptos familiares para el usuario, en lugar de términos orientados al sistema. Haciendo que la información aparezca en un orden natural y lógico.**

Los mensajes de error surgen en el momento en que el usuario comete algún error de entrada o si la aplicación produce algún error de conectividad, estos son breves y no utilizan palabras técnicas o confusas que puedan enredar al usuario del por qué el error.

3. **Control del usuario y libertad. Los usuarios suelen elegir funciones del sistema por error y necesitarán una "salida de emergencia" claramente marcada para dejar el estado no deseado sin tener que pasar por un diálogo extendido. La posibilidad de deshacer y rehacer acciones.**

En la app no es necesaria una “salida de emergencia” ya que los procesos que pueden tardar en terminar están implementados en hilos diferentes al de ejecución, por lo que el usuario puede navegar libremente por la aplicación sin interrupción.

4. **Consistencia y estándares. Los usuarios no deberían preguntarse si diferentes palabras, situaciones o acciones significan lo mismo. Siga las convenciones de la plataforma.**

Uno de los principales aspectos de la aplicación es que utiliza componentes estándares a los que los usuarios ya están acostumbrados a interactuar con ellos y a su vez estos responden de manera automática a posibles interacciones. Las palabras manejadas en botones, etiquetas, títulos y otros componentes guardan relación con la acción asociada.

5. **Prevención de errores. Incluso mejor que los buenos mensajes de error, es un diseño cuidadoso que impide que se produzca un problema en primer lugar. Elimine las condiciones propensas a errores o compruébelo y presente a los usuarios una opción de confirmación antes de comprometerse con la acción.**

En cada lugar donde podría incurrir un error por parte del usuario a la hora de escribir datos vitales o que deben corresponderse exactamente con valores de la BD se sustituyeron por componentes que en vez de escribir se selecciona las opciones posibles. De esta forma se reduce el error humano producido por el usuario al poder escribir datos incorrectos

6. **Reconocimiento en lugar de recordar. Minimice la carga de memoria del usuario haciendo visibles los objetos, las acciones y las opciones. El usuario no debe tener que recordar la información de una parte del diálogo a otra. Las instrucciones de uso del sistema deben ser visibles o fácilmente recuperables cuando sea apropiado.**

La aplicación cuenta con varios componentes interactivos, todos están visibles y situados de forma que minimice el riesgo de que el usuario pueda realizar acciones

equívocas accidentalmente. En todo momento se indica dónde se encuentra el usuario y la información necesaria en dependencia a la vista en la que se encuentre.

7. **Flexibilidad y eficiencia de uso. Los aceleradores no vistos por el usuario principiante pueden a menudo acelerar la interacción para el usuario experto, de manera que el sistema pueda atender a usuarios inexpertos y experimentados. Permite a los usuarios adaptar acciones frecuentes.**

Este parámetro no es evaluado ya que la aplicación no hace uso de aceleradores, esto se debe a que los dispositivos para los que fue diseñada no incluyen teclados que puedan facilitar su uso mediante acceso directo de por combinaciones de teclas. Además, los objetos con los que se interactúa no contienen opciones adicionales.

8. **Diseño estético y minimalista. Los diálogos no deben contener información irrelevante o raramente necesaria. Cada unidad extra de información compite con las unidades relevantes de información y disminuye la visibilidad relativa.**

El diseño de la aplicación está establecido en los principios de *Material Design*, de esta forma se certifica que el diseño sea minimalista y afín con la plataforma, en cuanto a la estética se tuvo en cuanto a los tamaños de las fuentes manejadas en correspondencia con la función que el texto desempeña, los controles alineados y uniformes que se adaptan de igual forma para cualquier dispositivo sin importar tamaño o versión del sistema operativo.

9. **Ayuda a los usuarios a reconocer, diagnosticar y recuperar errores. Los mensajes de error deben expresarse en lenguaje sencillo (sin códigos), indicar con precisión el problema y sugerir constructivamente una solución.**

Los mensajes de error son precisos sin llegar a una formalidad que desinterese a los usuarios, pero tampoco con una informalidad que los incomode; e indicando qué hacer para solucionar el problema.

10. **Ayuda y documentación. A pesar de que es mejor si el sistema puede utilizarse sin documentación, puede ser necesario proporcionar ayuda y**

documentación. Cualquier información de este tipo debe ser fácil de buscar, centrada en la tarea del usuario, enumerar los pasos concretos que se deben llevar a cabo, y no ser demasiado grande.

En caso de esta app no es necesario porque los usuarios ya están familiarizados con el proceso, por tanto, la app solo les facilitaría el trabajo que siempre han hecho.

3.6 Prueba de integración

Se debe probar no solo el funcionamiento individual de cada componente del software móvil, sino también su integración con los demás componentes de la aplicación, especialmente con el servidor de datos con el cual se realiza el proceso de sincronización de la información utilizada en el dispositivo (Yagüe and Garbajosa 2009).

Se realizó una prueba absoluta en cada una de las posibles pantallas, verificando el uso correcto del funcionamiento de cada uno de los componentes interactivos o no, y en el caso de los que presentan interacción se comprobó que cada uno de estos realizaba la función esperada, como se describe en las siguientes figuras.

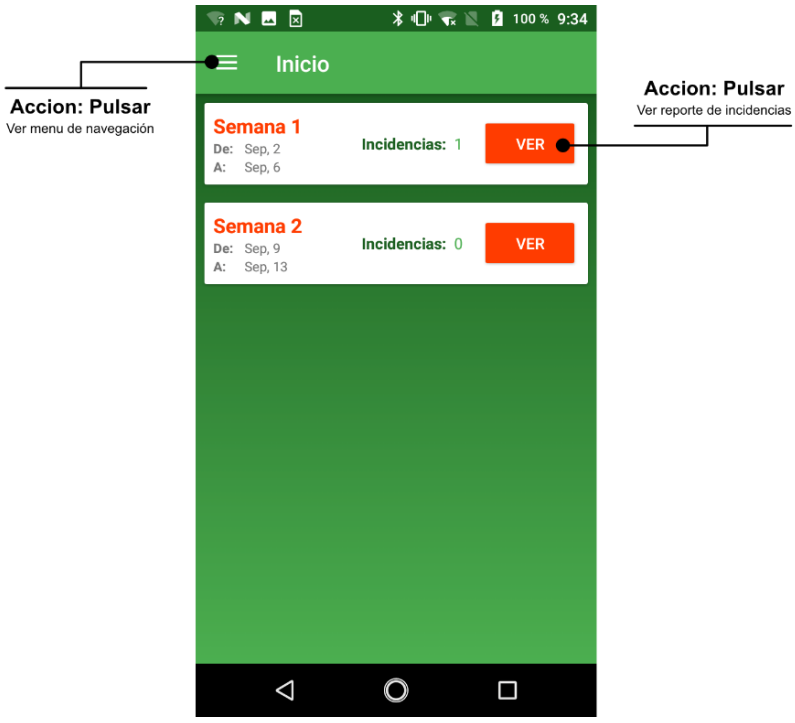


Fig. 3.1 Prueba de integración a la vista Ver reportes de Incidencia

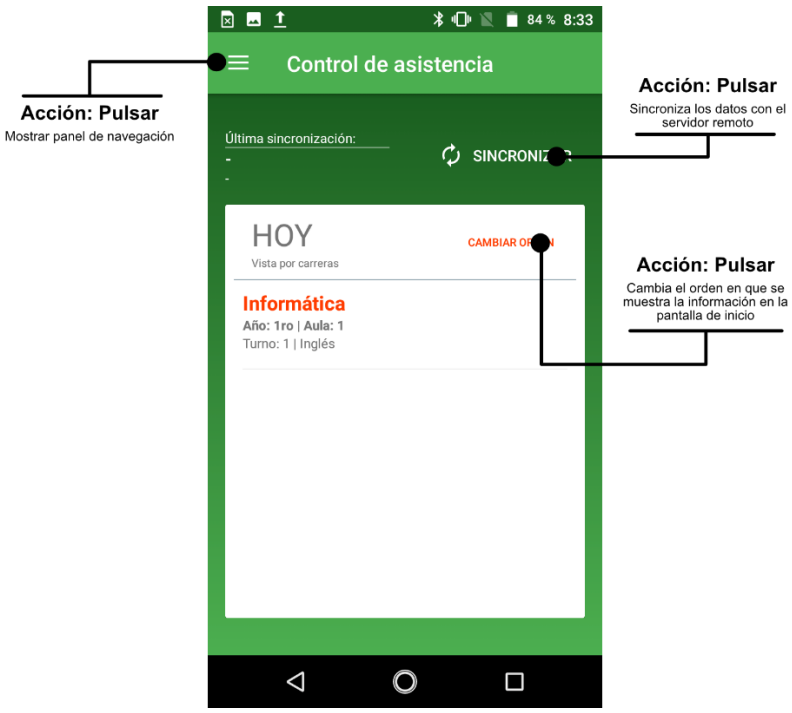


Fig. 3.2 Prueba de integración a la Vista Principal

Fig. 3.3 Prueba de integración a la Vista Chequear asistencia

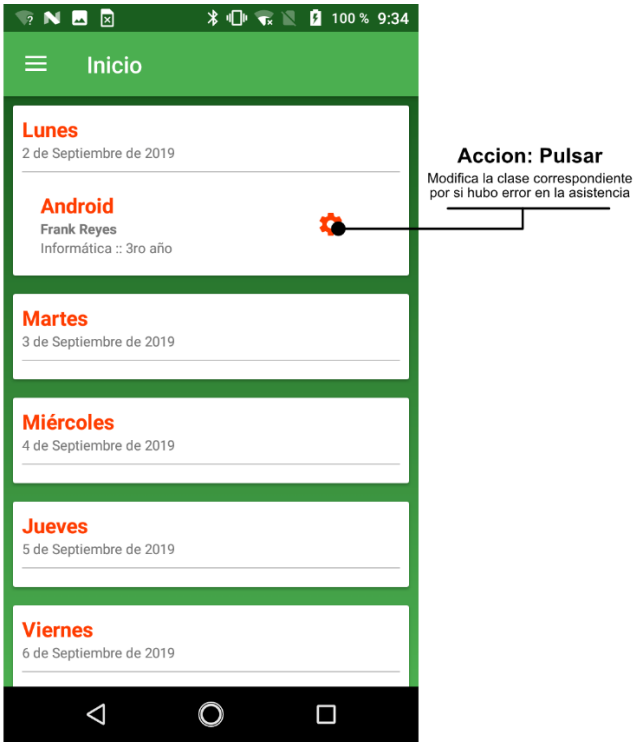


Fig. 3.4 Prueba de integración a la Vista Modificar asistencia

3.7 Conclusiones parciales

La aplicación pasó por un conjunto de pruebas tanto de funcionamiento como pruebas de usabilidad, unitarias entre otras; y los resultados obtenido fueron satisfactorios en cada caso que fue evaluado.

Conclusiones

Como resultado de esta investigación, se ha llegado a las siguientes conclusiones:

1. Se realizó un estudio del proceso manual de control del cumplimiento del horario docente en la facultad de Matemática, Física y Computación (FMFC) de la UCLV llegando a entender que una solución manual no es la más adecuada, ni efectiva para controlar la asistencia de los profesores a sus clases.
2. El diseño de una aplicación móvil para el sistema operativo Android permitió automatizar todo el proceso de control de asistencia de profesores a la facultad.
3. La aplicación logró generar reportes fiables y seguros para supervisar el cumplimiento del horario docente.
4. El diseño de la base de datos permitió el almacenamiento de toda la información relacionada con el horario docente.
5. Se evaluó el funcionamiento de la aplicación mediante pruebas unitarias, validación, usabilidad y pruebas de campo llegando a obtener resultados satisfactorios.

Recomendaciones

A partir del estudio realizado se recomienda:

1. Extender el uso de la aplicación para todas las facultades de la Universidad Central ‘‘Marta Abreu’’ de Las Villas.
2. Desarrollar un sitio web y vincularlo a la aplicación con el fin de mejorar el control del cumplimiento del horario docente en la FMFC.
3. Realizar una investigación más profunda para determinar nuevas funcionalidades que se puedan agregar a la aplicación.

Referencias Bibliográficas

Araujo, Y. (2011). "Metodologia RUP."

Blanco (2009). Mobile D.

Carson, R. S. (2016). "La guía definitiva para ORM en Android con ActiveAndroid: Parte 1."

Catalani, E. (2007). "Arquitectura Modelo/Vista/Controlador." Extraído El **30**.

Cuello, J. and J. Vittone (2013). Diseñando apps para móviles, José Vittone—Javier Cuello.

Developers, A. (2011). What is android, Android Developers, http://developer. android. com/guide/basics/what-is-android. html, accessed May.

Díaz, S. (2013). "Mejores prácticas en las pruebas de aplicaciones móviles."

Flores, F. C. (2016). "Sistema de control de asistencia de personal de la Universidad del Bío-Bío."

Fowler, M. and K. Scott (1999). UML gota a gota, Pearson Educación.

Gilfillan, I. (2011). La Biblia MySQL.

González, Y. D. and Y. F. Romero (2012). "Patrón Modelo-Vista-Controlador." Revista Telem@ tica **11**(1): 47-57.

Guilermo, J. M. (2011). Introduccion a Java.

Jacobson, I., et al. (1997). "The Objectory Software Development Process." ISBN: 0-201-57169-2, Addison Wesley.

Larman, C. (2003). UML Y PATRONES. Una introducción al análisis y diseño orientado a objetos y al proceso unificado. Aragón DF, Madrid: Pearson Educación. SA.

Mantilla, M. C. G., et al. (2014). "Metodología para el desarrollo de aplicaciones móviles." Revista Tecnura **18**(40): 20-35.

MARTÍNEZ, D. I. M. (2017). DESARROLLO DE PROTOTIPO MÓVIL PARA EL

CONTROL Y REGISTRO DE ASISTENCIA DE

ALUMNOS, Pontificia Universidad Católica De Valparaíso.

Montiel, D. P. (2015). "Introduccion a SQLite version moderna."

nextcloud.com (2017). End-to-End Encryption Design.

Nielsen, J. (1995). "10 usability heuristics for user interface design." Nielsen Norman Group **1**(1).

Pressman, R. S. and J. M. Troya (1988). "Ingeniería del software."

Quesada, I. F., et al. (1996). Diseño y medición de trabajos, Universidad de Oviedo, Servicio de Publicaciones.

ROMERO, M. N. U. (2011). SISTEMA INFORMÁTICO PARA EL CONTROL

DE ASISTENCIA DEL PERSONAL DOCENTE

DEL CENTRO DE EDUCACIÓN BÁSICA, Granma.

Santiago, R., et al. (2015). Mobile learning: nuevas realidades en el aula, Editorial Oceano.

Schmidt, D. C. (2012). "Systems Programming for Android."

Sommerville, I. (2005). Ingeniería del software, Pearson Educación.

Tracy, K. W. (2012). "Mobile application development experiences on Apple's iOS and Android OS." Ieee Potentials **31**(4): 30-34.

Wolfram, W. and N. Schilling (2015). American English: dialects and variation, John Wiley & Sons.

Yagüe, A. and J. Garbajosa (2009). "Las pruebas en metodologías ágiles y convencionales: papeles diferentes." Actas de los Talleres de las Jornadas de Ingeniería del Software y Bases de Datos **3**(4).