UCLV
Universidad Central
"Marta Abreu" de Las Villas

MFC
Facultad de Matemática
Física y Computación

Departamento

TRABAJO DE DIPLOMA

Título del trabajo: Aplicación móvil para la visualización de los eventos registrados en la Plataforma de Gestión de Eventos de la UCLV.

Autores del trabajo: Diacny Ramos Arocha.

Tutores del trabajo: Frank Reyes García, José Antonio López Arencibia.

Santa Clara, junio,2018

Pensamiento: Más que inteligencia, necesitamos bondad y gentileza.

Charles Chaplin

Agradecimientos.

A **mi familia** por creer en mí y apoyarme siempre, por tantos sacrificios, por hacer de mi la persona que soy, todo lo que soy y seré se lo debo a ustedes.

A mi amiga **Chabeli** por demostrarme que la distancia separa cuerpos no corazones. Por seguir siendo mi confidente aun en la distancia, por los consejos, el apoyo, y el cariño que de ti siempre recibí.

A mi amiga **Yadira** por poder contar con ella siempre, y enseñarme que los grandes amigos nos son aquellos que se ven o hablan todos los días sino aquellos que sin importar el tiempo que pase sin hablarse o verse siempre van a estar a tu lado cuando los necesitas.

A mis amigas **Gabriela** y **Ana** por tanto tiempo dedicado, por tantos regaños y tantas exigencias, por tantos consejos y críticas constructivas que quizás en su momento no asimile pero que sí surtieron efecto, por la ayuda y el apoyo no solo en el plano docente sino también en lo personal.

A mis amigas **Gretter**, **Rachel** y **Elizabeth** por brindarme su casa, por las cenas en familia y por permitirme dejar la luz encendida hasta tarde.

A mis amigos **Yanier**, **Rainer**, **Geovel**, **Marlon**, **Yaisel** por apoyarme siempre sin importar cuales fueran las circunstancias, por las noches de mejunje, por esas conversaciones que me ayudaron a crecer, a querer ser mejor, a creer en mí, a cultivar valores y a soñar.

A todas aquellas personas que de una forma u otra contribuyeron a que este sueño fuese hoy una realidad.

Muchas gracias.

Dedicatoria.

A mis padres por la confianza que siempre pusieron en mí y que en muchas ocasiones ha sido lo que me ha motivado a seguir adelante, por apoyarme siempre en todas mis decisiones por descabelladas que fuesen, por ser mi guía, mi refugio y mi ejemplo.

Resumen

Los dispositivos móviles en la actualidad son mucho más que un terminal con servicios de telefonía. Proporcionan multitud de servicios y funciones que pueden ser explotadas de diversas maneras a través de las aplicaciones. Debido a su portabilidad y el gran número de funcionalidades que ofrecen se han convertido en uno de los avances tecnológicos más utilizados de este siglo. En el ámbito universitario una de sus utilidades más comunes es su uso para acceder a los servicios de la red de la UCLV. Teniendo esto en cuenta se pretende desarrollar una aplicación para dispositivos móviles con Sistema Operativo Android con el objetivo de acceder a la información de los eventos disponibles en la Plataforma para la gestión de eventos de la UCLV, dicho acceso se hará a través de un servicio web con arquitectura REST el cual permitirá la conexión con la Base de Datos de la Plataforma y la extracción de información de la misma, haciéndola persistente en el dispositivo posibilitando así su consulta incluso estando offline.

Abstract

Mobile devices today are much more than a terminal with telephony services. They provide a multitude of services and functions that can be exploited in various ways through applications. Due to their portability and the large number of functionalities they offer, they have become one of the most used technological advances of this century. In the university environment one of its most common uses is its use to access the services of the UCLV network. Taking this into account, we intend to develop an application for mobile devices with Android Operating System with the objective of accessing the information of the events available in the Platform for the management of UCLV events, said access will be done through a web service with REST architecture which will allow the connection with the Database of the Platform and the extraction of information from it, making it persistent in the device thus enabling its consultation even when offline.

Tabla de Contenidos

INTRODUCCIÓN

El Acceso abierto (Open Access) es un movimiento que promueve el acceso libre y gratuito a la literatura científica, fomentando su libre disponibilidad en Internet y permitiendo a cualquier usuario su lectura, descarga, copia, impresión, distribución o cualquier otro uso legal de la misma, sin ninguna barrera financiera, técnica o de cualquier tipo. La única restricción sobre la distribución y reproducción es dar al autor el control sobre la integridad de su trabajo y el derecho a ser adecuadamente reconocido y citado. El principal objetivo del acceso abierto es aumentar el impacto de la investigación al incrementar el acceso a la misma (Parker, 2013).

En el más amplio sentido de la expresión y con respecto a las publicaciones científicas, el OA está vinculado a las iniciativas o proyectos que favorezcan y promuevan el acceso abierto, libre y sin restricciones a los trabajos publicados por la comunidad científica.

Una de las iniciativas enmarcadas en el Movimiento de Acceso Abierto es el *PublicKnowledge Project* (PKP), un proyecto fundado en 1998 por el Dr. John Willinsky en el Departamento de Lenguaje y Educación Literaria de la Facultad de la Educación en la Universidad de British Columbia (Vancouver, Canadá) (Téllez Alarcia, 2010)
Entre los softwares creados por este proyecto está el Open Conference Systems, un sistema abierto de eventos por Internet fácil de usar para todas las necesidades de un evento. Los eventos o simposios son acontecimientos donde se reúne importante conocimiento y se recogen significativos resultados. En los eventos académicos en especial se obtiene importante producción científica y se actualizan los investigadores de lo que, en un campo en particular, sucede a nivel nacional e internacional.
“Es una solución de código abierto del proyecto PKP para la administración de conferencias en varios idiomas. Está desarrolla completamente en php y puede utilizar base de datos en MySQL o PostgreSQL. Actualmente, cuenta con más 1400 conferencias que utilizan esta plataforma como soporte tecnológico para la realización del evento” (Ramírez Vega, 2011) .

Los dispositivos móviles constituyen uno de los principales medios para el intercambio y consulta de información en la actualidad debido a las facilidades que estos brindan.

Un dispositivo móvil, lo podemos definir, como un aparato de pequeño tamaño, el cual posee un sinfín de funciones, entre las cuales podemos mencionar, el procesamiento e intercambio de información, la conexión a alguna red, todo esto a través de una memoria interna y limitada (Luna, Vaca and Vásquez, 2017).

Como todas las tecnologías de la información y la comunicación, los dispositivos móviles se han instaurado en nuestras vidas. Hoy en día prácticamente todos llevamos en nuestro bolsillo un terminal móvil la gran mayoría con sistema operativo Android. En la UCLV se tratan de aprovechar al máximo estos recursos poniendo al alcance de los estudiantes diversas aplicaciones que faciliten, la consulta y acceso a la información disponible en los diferentes sitios de la red de la UCLV, pero a pesar de ello aún no se cuenta con una herramienta que permita la extracción de información de la Plataforma de Gestión de Eventos de la UCLV.

La situación antes descrita permite definir el siguiente problema de investigación:

Problema de investigación:

La poca accesibilidad a la información disponible en la plataforma de gestión de eventos de la UCLV.

Para dar respuesta al problema de investigación se propone como **objetivo general:** el diseño e implementación de una aplicación que permita la consulta de la información disponible en la Plataforma para la gestión de eventos de la UCLV.

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

1. Identificar los contenidos que se desean mostrar en la aplicación.
2. Determinar mecanismo para la extracción de la información.
3. Determinar forma de almacenar la información para hacerla persistente en el dispositivo.
4. Evaluar el desempeño del software diseñado a través de las pruebas de usabilidad y funcionamiento de la aplicación.

Las **preguntas de investigación** planteadas son las siguientes:

1. ¿Cuáles son las políticas de interacción definidas?
2. ¿Cuál es el mecanismo de extracción de información escogido?
3. ¿Cómo se almacenará la información para hacerla persistente en el dispositivo?
4. ¿Cuáles pruebas serán aplicadas para comprobar el correcto funcionamiento de la aplicación?

El **valor práctico** del trabajo está dado en que la aplicación **OCS-UCLV** permite a los usuarios consultar la información de los eventos activos en la Plataforma de Gestión de Eventos de la Universidad Central "Marta Abreu" de Las Villas, ofreciendo utilidades extras al usuario, como es el caso de la selección de eventos de su interés y la propia aplicación se encargara de notificarle en la fecha correspondiente

El **valor metodológico** del trabajo está dado por la sistematización de un conjunto de métodos científicos y procedimientos técnicos que permitieron el desarrollo de una aplicación móvil en función de promover la difusión de la información de los eventos realizados en la UCLV. Demuestra la viabilidad del uso de las tecnologías de la información y la comunicación como medio de difusión masiva de la información en la comunidad universitaria.

La presente investigación está estructurada en tres capítulos. En el **primer capítulo** se expone los fundamentos teóricos que respaldan la investigación relacionados con los repositorios digitales, en específico el repositorio de ponencias y eventos científicos de la UCLV. Se investiga los modelos usados para el desarrollo del software, así como sus diferentes fases. En el **segundo capítulo** se realiza un estudio de las principales arquitecturas, lenguajes y tecnologías utilizadas para el diseño e implementación del servicio web. También se valida el servicio implementado realizando las pruebas pertinentes. Además, se hace un análisis de los principales requerimientos de la aplicación. Se describen los principales diagramas y algunos de los algoritmos y técnicas empleadas. En el **tercer capítulo** se realizan las pruebas pertinentes a la aplicación para asegurarse de su correcto funcionamiento.

Capítulo 1: Fundamentación teórica.

El punto inicial del desarrollo de cualquier aplicación móvil es identificar el problema que se desea resolver. Algo común que sucede cuando no se tiene experiencia en el desarrollo de aplicaciones es comenzar pensando en implementar varias características que se desean tener en la aplicación, sin ver qué hay que centrarse en una característica principal de la aplicación, Es decir lo que usualmente se conoce como un Producto Mínimo Viable (MVP de *Mínimum Viable Product*) teniendo como eje la solución de este elemento inicial (Ramos Martínez, 2013).

Las aplicaciones móviles son aquellas que fueron desarrolladas para ejecutarse en dispositivos móviles. El término móvil se refiere a poder acceder a los datos, las aplicaciones y los dispositivos desde cualquier lugar. Para desarrollar software de este tipo se tiene que tener en cuenta ciertas restricciones que tiene el hardware de estos dispositivos, como por ejemplo que son de dimensiones reducidas, tienen bajo poder de cómputo, escasa capacidad de almacenamiento, ancho de banda limitado, etc. Algunos ejemplos de aplicaciones móviles son: mapas y navegación, búsqueda, juegos, mensajería, aplicaciones empresariales (Enriquez and Casas, 2014).

1.1 Definición de un Repositorio Digital.

Un repositorio digital es un conjunto de servicios prestados por una institución o centro al servicio de la comunidad para recopilar, gestionar, difundir y preservar la producción documental de dicho centro a través de la creación de una colección digital organizada, abierta e interoperable para garantizar mayor impacto y visibilidad.

(Lynch, 2003) define un repositorio digital como un sistema informático que integra varios servicios que permiten incorporar, reunir, preservar, consultar y gestionar los recursos digitales creados por un centro investigativo, así como hacerlos accesibles al resto de la comunidad, a través de una interfaz o portal web y mediante una adecuada clasificación de recursos sobre la base de metadatos.

1.2 Sistema de Repositorios Digitales de la UCLV

En el año 2016, como trabajo de diploma, Yoandy Hernández Pérez, estudiante de la FACULTAD DE INGENIERÍA ELÉCTRICA implementó en la UCLV un Sistema de

Repositorios Digitales (SDR) con el objetivo de facilitar la socialización dentro y fuera del campus, de los resultados científicos y académicos generados en cada una de las áreas, además de ampliar la visibilidad y accesibilidad de la información.

1.2.1 Estructura del Sistema de Repositorios Digitales de la UCLV.

En (Rivero, Hidalgo and Cañizares, 2016) se describe cómo está estructurado el Sistema de Repositorios Digitales de la UCLV:

1. Repositorio para la producción científica soportado por la plataforma DSpace, uno de los softwares libres más empleados a nivel mundial. Dentro de este sistema se archivan las tesis de maestría y doctorados, trabajos de diploma, artículos publicados en revistas nacionales e internacionales, informes de investigación, así como objetos docentes de la institución.
2. Repositorio de revistas científicas, para el cual se usa *Open Journal Systems* (OJS), incluye las 4 revistas científicas editadas en la UCLV (Centro Azúcar, Centro Agrícola, Islas y Biotecnología Vegetal). También se archivan las publicaciones seriadas de la Facultad de Cultura Física "Manuel Fajardo" y de la Sede "Félix Varela".
3. Repositorio de Monografías sobre *Open Monograph Press* que aloja las monografías editadas por la editorial universitaria "Feijóo". Empleado para administrar los procesos vinculados a la edición en línea de las monografías.
4. Repositorio de la Biblioteca Digital soportado también sobre DSpace donde se depositan los libros en formato electrónico adquiridos por la Universidad. También contiene los documentos reproducidos por la UCLV con fines educativos, amparado por la Ley del Derecho de Autor.
5. Repositorio de ponencias y eventos científicos. Implementado sobre la plataforma *Open Conference Systems*, es utilizado como sistema de publicación de ponencias y eventos en línea.
6. Sistema de recolección de metadatos. Utiliza el sistema *Open Harvester Systems* como proveedor de servicios.

1.2.2 Repositorio de ponencias y eventos científicos.

Como se ha mencionado anteriormente el Repositorio de ponencias y eventos científicos esta implementado sobre la plataforma Open Conference Systems la cual constituye una solución de fuente abierta creada por Public Knowledge Project (PKP) para manejar y publicar congresos programados en línea. OCS es un sistema de gestión altamente flexible basado en PHP / MySQL, así como un sistema de publicación, que puede ser descargado gratuitamente e instalado en un servidor Web local. Ha sido diseñado para reducir el tiempo y energía dedicados a las tareas de oficina y de gestión asociadas con la gestión de un congreso, mientras se trabaja sobre la archivación y la eficiencia del proceso editorial. Busca mejorar la calidad de publicación y la parte académica de la publicación de un congreso a través de un número de innovaciones, desde hacer más transparentes las políticas a aumentar la indexación. Se ajusta perfectamente a las necesidades de una conferencia o congreso para cualquier institución.

El OCS que forma parte del Sistema de Repositorios Digitales (SRD) de la UCLV, ha sido instalado en un servidor virtual, alojado en el data center principal de la institución, con sistema operativo Debian, en su versión 7.0 wheezy con todas las actualizaciones disponibles en los repositorios de software. Cuenta con 20 GB de almacenamiento con posibilidades de expansión cuando el sistema lo requiera y 2 GB de RAM. También se han instalado las aplicaciones necesarias como Apache, MySQL, PHP y demás. Dicho sitio puede ser accedido a través de la dirección http://ocs.cdict.uclv.edu.cu (Hernández Pérez, 2016).

En la Figura 1.1 se muestra la página principal de la Plataforma de Gestión de eventos de la UCLV.

Figura 1.1: *Plataforma para la gestión de eventos de la UCLV*

1.3 Modelos de desarrollo de aplicaciones móviles.

Un modelo de procesos del software es la descripción simplificada de un proceso del software que presenta una visión simplificada de ese proceso. Estos modelos pueden incluir actividades que son parte de los procesos y productos de software y el papel de las personas involucradas en la ingeniería del software (Sommerville, 2005).

1.3.1 Modelo waterfall

El modelo waterfall es el modelo más estático y predictivo. Es aplicable en proyectos en los que los requisitos están fijados y no van a cambiar durante el ciclo de vida del desarrollo. Esta aproximación divide el proyecto en fases totalmente secuenciales. En este modelo, el desarrollo se interpreta como el agua que va cayendo de un estanque al siguiente. Se le da mucho énfasis a la planificación, a los tiempos, a las fechas límite y al presupuesto (Vique, 2012a)

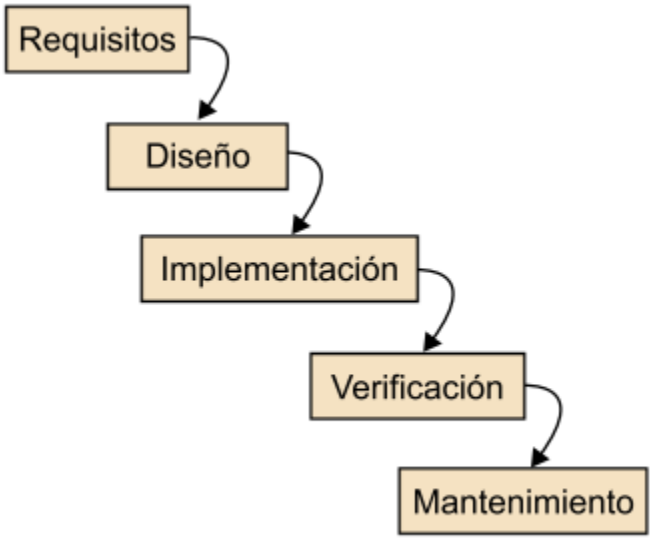

Figura 1.2: *Imagen del modelo waterfall*

1.3.2. Desarrollo ágil

Las metodologías ágiles resuelven los problemas surgidos, posteriormente, a la masificación del uso del computador personal, dado que las expectativas y necesidades por parte de los usuarios se hicieron más urgentes y frecuentes. Fue así como a comienzo de los 90 surgieron propuestas metodológicas para lograr resultados más rápidos en el desarrollo de software sin disminuir su calidad (Herrera Uribe and Valencia Ayala, 2007).

Manifiesto ágil: El manifiesto ágil fue publicado en el 2001 por diecisiete desarrolladores de software, quienes representaban entonces los métodos de desarrollo más populares, que pasarían a conocerse como ágiles (Extreme Programming, Crystal Clear, DSDM o ASD, entre otros). El manifiesto define doce principios y cuatro valores éticos para los desarrolladores. Puede ser consultado en http://agilemanifesto.org

El desarrollo ágil se basa en los principios del manifiesto ágil y sus valores éticos, que tratan de dar más valor a algunos conceptos, pero sin dejar de lado los demás. A continuación, se enumeran los cuatro valores éticos expresados en el Manifiesto Ágil:

1) Dar más valor a los individuos y a sus interacciones que a los procesos y herramientas.

2) Dar más valor al software que funciona que a la documentación exhaustiva.

3) Dar más valor a la colaboración con el cliente que a la negociación contractual.

4) Dar más valor a la respuesta al cambio que al seguimiento de un plan.

Con estos valores se intenta conseguir, entre otras cosas, entregar algo lo más pronto posible y evitar problemas originados por cambios de requisitos. Esto es muy apropiado para proyectos cambiantes, ya sean grandes o pequeños, ya que mediante estos valores se pueden mitigar los riesgos. Para conseguir proyectos que puedan cambiar fácilmente, se pone especial atención en la calidad de los productos conseguidos, cosa que es realmente importante en proyectos de software para dispositivos móviles.

1.3.3. Mobile-D

El método Mobile-D se desarrolló junto con un proyecto finlandés en el 2004. Fue realizado, principalmente, por investigadores de la VTT (Instituto de Investigación Finlandés) y, a pesar de que es un método antiguo, sigue en vigor (se está utilizando en proyectos de éxito y está basado en técnicas que funcionan).

El objetivo es conseguir ciclos de desarrollos muy rápidos en equipos muy pequeños (de no más de diez desarrolladores) trabajando en un mismo espacio físico. Según este método, trabajando de esa manera se deben conseguir productos totalmente funcionales en menos de diez semanas (Abrahamsson *et al.*, 2004).

Se trata de método basado en soluciones conocidas y consolidadas: Extreme Programming (XP), Crystal Methodologies y Rational Unified Process (RUP), XP para las prácticas de desarrollo, Crystal para escalar los métodos y RUP como base en el diseño del ciclo de vida.

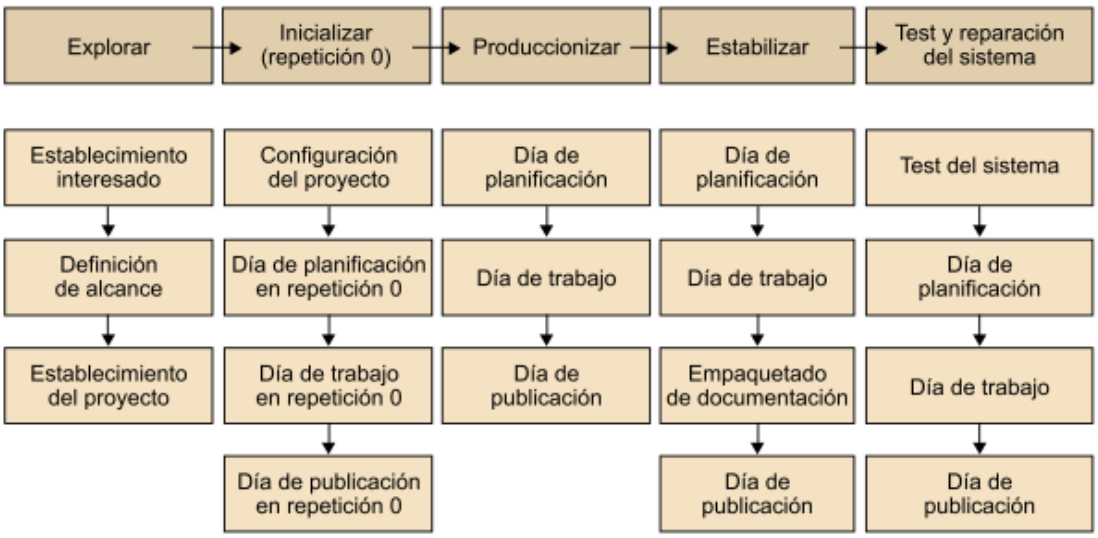

Figura 1.3: *Imagen del modelo Mobile-D*

Un ciclo de proyecto con la metodología **Mobile-D** está compuesto por cinco fases:

- Exploración. Se dedica a la planificación y a los conceptos básicos del proyecto. Es diferente del resto de fases.
- Inicialización. Se preparan e identifican todos los recursos necesarios. Se establece el entorno técnico.
- Productización o fase de producto. Se repiten iterativamente las subfases, con un día de planificación, uno de trabajo y uno de entrega. Aquí se intentan utilizar técnicas como la del test driven development para conseguir la mayor calidad.
- Fase de estabilización. Se llevan a cabo las acciones de integración para asegurar que el sistema completo funciona correctamente.
- Fase de pruebas y reparación. Tiene como meta la disponibilidad de una versión estable y plenamente funcional del sistema según los requisitos del cliente

1.6 Sistemas Operativos.

Existen diferentes sistemas operativos móviles, pero sin duda el más utilizado universalmente es Android.

Android es un sistema operativo y una plataforma software, basado en Linux para teléfonos móviles. Además, también usan este sistema operativo (aunque no es muy habitual), tablets, netbooks, reproductores de música e incluso PC's. Android permite programar en un entorno de trabajo (framework) de Java, aplicaciones sobre una máquina virtual Dalvik (una variación de la máquina de Java con compilación en tiempo de ejecución) (Báez *et al.*, 2012).

Como podemos apreciar en la Figura 1.4 Android es el Sistema Operativo que esta dominando el mercado de *Smartphone* en la actualidad.

Operating System	2017 Units	2017 Market Share (%)	2016 Units	2016 Market Share (%)
Android	1,320,118.1	85.9	1,268,562.7	84.8
iOS	214,924.4	14.0	216,064.0	14.4
Other OS	1,493.0	0.1	11,332.2	0.8
Total	**1,536,535.5**	**100.0**	**1,495,959.0**	**100.0**

Source: Gartner (February 2018)

Figura 1.4: Cuota de mercado de los Sistemas Operativos en la actualidad

La figura muestra el estado actual de los sistemas operativos móviles según un estudio realizado por Gartner[1] en febrero de 2018. Se puede apreciar que Android supone casi un 86% del mercado mientras que los demás sistemas operativos conformarían un 14%. Este predominio de Android sobre otros sistemas operativos móviles se basa en el hecho de que Android se ejecuta en muchos más móviles que la competencia ya que es un sistema de

[1] Gartner Inc. es una empresa consultora y de investigación de las tecnologías de la información con sede en Stamford, Connecticut, Estados Unidos.

código abierto, libre de implementar por cualquier compañía de telefonía. iOS, sin embargo, es privativo y solo se producen móviles con este software desde Apple.

1.6.1 Arquitectura de Android.

La arquitectura interna de la plataforma Android, está básicamente formada por 4 componentes: aplicaciones, armazón de aplicaciones, librerías y kernel/Linux, como se muestra en la siguiente gráfica (Rivera, 2012):

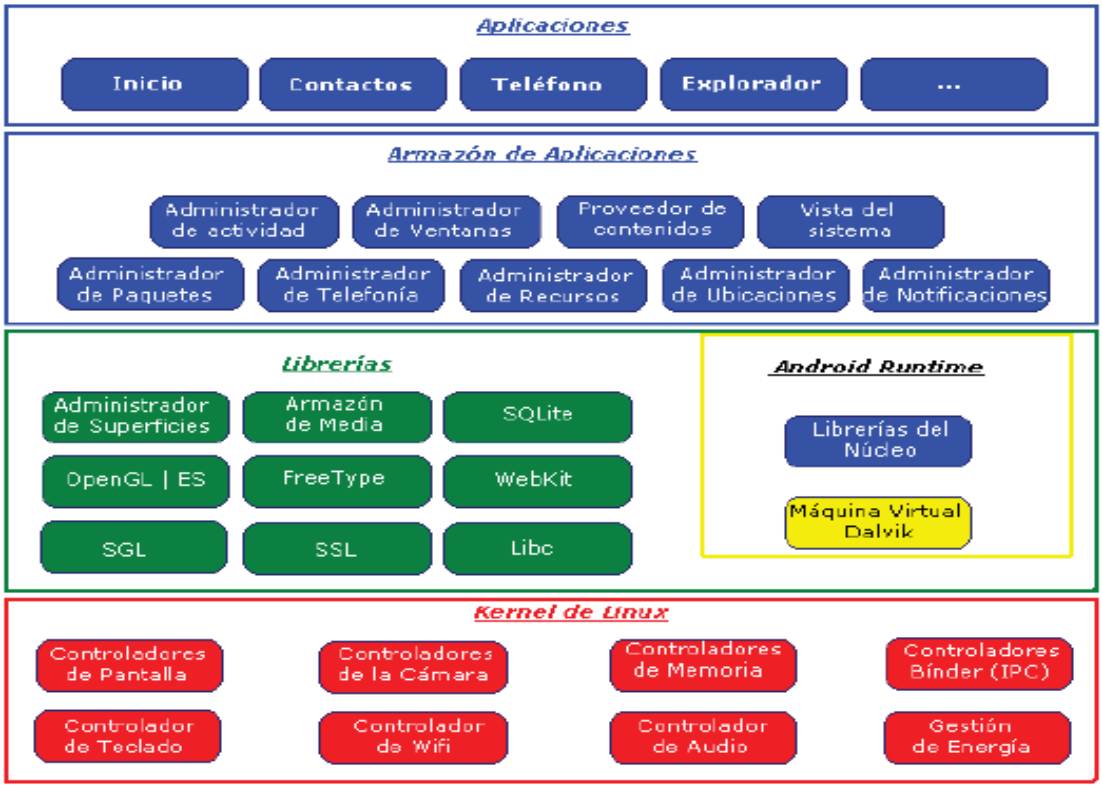

Figura 1.5: Arquitectura del Sistema Operativo Android.

Si se parte de una vista ascendente, lo primero que se encuentra es el núcleo de Android. Basado en el kernel del sistema operativo Linux. Según (Vanegas, 2012) esta es una capa de abstracción del hardware (hardware abstraction layer), que permite que las aplicaciones accedan a través de controladores (drivers) asumiendo la administración de los recursos del teléfono y del sistema operativo.

A continuación, se localiza el entorno de ejecución (*Android Runtime*), basado en la máquina virtual de Java. Dadas las limitaciones de los dispositivos Google decidió crear la máquina

virtual Dalvik, la cual respondió mucho mejor a dichas limitaciones. Entre sus características destacan, la optimización de recursos debido a la ejecución de ficheros Dalvik ejecutables (.dex) y la delegación al kernel de Linux procesos como el *threading* y el manejo de memoria a bajo nivel.

Al mismo nivel se ubican las librerías nativas, escritas en C/C++ y compiladas en código nativo del procesador. Muchas librerías utilizan código abierto.

El entorno de aplicación (*Application Framework*) ofrece una plataforma de desarrollo libre para aplicaciones, donde su principal riqueza reside en la reutilización de componentes. Componentes desarrollados por Google, o por usuarios.

Por último, se encuentra la capa de aplicaciones (*Applications*) formada por el conjunto de aplicaciones del dispositivo, ya sean nativas o instaladas por el usuario.

1.6.2 Ventajas

Entre las ventajas más importantes de desarrollar una aplicación Android, se tiene las siguientes:

- Android es compatible con varios dispositivos y actualmente es el Sistema Operativo más utilizado por dispositivos móviles, por lo tanto, la app se podría instalar en la mayoría de dispositivos móviles.
- Android es un software libre gracias a la licencia Apache, lo cual le convierte en un sistema operativo totalmente abierto, donde cualquier desarrollador pueda crear, modificar y mejorar el código de la app.
- El desarrollo de una app android no requiere de permisos a terceros, existe total libertad.
- Existe una amplia comunidad de desarrolladores en la red.

1.6.3 IDE de programación

Un IDE es una herramienta que ayuda a desarrollar de una manera amigable las aplicaciones, brindando ayudas visuales en la sintaxis, plantillas, wizards, plugins,etc.

Android Studio es el entorno de desarrollo específico de Android, basado en IntelliJ IDEA (entorno de desarrollo conocido en la comunidad de desarrolladores Java).Aporta muchas novedades respecto a Eclipse (Benbourahla, 2015) .

Entre las características principales de Android Studio, se tiene las siguientes:

- Renderizado en tiempo real
- Consola de desarrollador: consejos de optimización, ayuda para la traducción, estadísticas de uso.
- Soporte para construcción basada en Gradle.
- Refactorización específica de Android y arreglos rápidos.
- Un editor de diseño enriquecido que permite a los usuarios arrastrar y soltar componentes de la interfaz de usuario.
- Herramientas Lint para detectar problemas de rendimiento, usabilidad, compatibilidad de versiones y otros problemas.
- Plantillas para crear diseños comunes de Android y otros componentes.
- Soporte para programar aplicaciones para Android Wear.
- Un dispositivo virtual de Android que se utiliza para ejecutar y probar aplicaciones.

1.7 Tipos de Aplicaciones según su desarrollo.

Según su desarrollo las aplicaciones puden ser clasificadas de tres formas: aplicaciones web, aplicaciones nativas y aplicaciones híbridas. Hay varios factores a analizar a la hora de elegir entre los diferentes tipos de aplicaciones: costo, actualizaciones, experiencia de usuario, mantenimiento, etc.

Dadas las características de cada una de las aplicaciones, decidirse por una u otra estará determinado por un conjunto de factores fundamentales y por la forma en que afectan finalmente la experiencia de uso.

1.7.1 Aplicaciones Web

Las aplicaciones web para móviles son diseñadas para ser ejecutadas en el navegador del dispositivo móvil. Estas aplicaciones son desarrolladas utilizando HTML, CSS y JavaScript, es decir, la misma tecnología que la utilizada para crear sitios web.

Una de las ventajas de este enfoque es que los dispositivos no necesitan la instalación de ningún componente en particular, ni la aprobación de algún fabricante para que las aplicaciones sean publicadas y utilizadas. Solo se requiere acceso a internet. Además, las

actualizaciones de la aplicación son visualizadas directamente en el dispositivo, ya que los cambios son aplicados sobre el servidor y están disponibles de inmediato. En resumen, es rápido y fácil de poner en marcha. La principal ventaja de este tipo de aplicación es su independencia de la plataforma. No necesita adecuarse a ningún entorno operativo. Solo es necesario un navegador. Por contrapartida, esto disminuye la velocidad de ejecución y podrían llegar a ser menos atractivas que las aplicaciones nativas. Además, podrían tener baja performance por problemas de conectividad. Finalmente, este tipo de aplicaciones no pueden utilizar todos los elementos de hardware del dispositivo, por ejemplo, cámara, GPS, entre otros.

1.7.2 Aplicaciones Nativas

Las aplicaciones nativas son aquellas que se conciben para ejecutarse en una plataforma específica, es decir, se debe considerar el tipo de dispositivo, el sistema operativo a utilizar y su versión. El código fuente se compila para obtener código ejecutable, proceso similar que el utilizado para las tradicionales aplicaciones de escritorio. Cuando la aplicación está lista para ser distribuida debe ser transferida a las App stores (tiendas de aplicaciones) específicas de cada sistema operativo. Estas tienen un proceso de auditoría para evaluar si la aplicación se adecúa a los requerimientos de la plataforma a operar. Cumplido este paso, la aplicación se pone a disposición de los usuarios. La principal ventaja de este tipo de aplicaciones es la posibilidad de interactuar con todas las capacidades del dispositivo (cámara, GPS, acelerómetro, agenda, entre otras). Además, no es estrictamente necesario poseer acceso a internet. Su ejecución es rápida, puede ejecutarse en modo background y notificar al usuario cuando ocurra un evento que necesite su atención. Claramente estas ventajas se pagan con un mayor costo de desarrollo, pues se debe utilizar un lenguaje de programación diferente según la plataforma. Por ende, si se desea cubrir varias plataformas, se deberá generar una aplicación para cada una de ellas. Esto conlleva a mayores costos de actualización y distribución de nuevas versiones.

1.7.3 Aplicaciones Híbridas

Las aplicaciones híbridas combinan lo mejor de los dos tipos de aplicaciones anteriores. Se utilizan tecnologías multiplataforma como HTML, Javascript y CSS, pero se puede acceder a buena parte de las capacidades específicas de los dispositivos. En resumen, son

desarrolladas utilizando tecnología web y son ejecutadas dentro de un contenedor web sobre el dispositivo móvil.

Entre las principales ventajas de esta metodología se pueden mencionar la posibilidad de distribución de la aplicación a través de las tiendas de aplicaciones, la reutilización de código para múltiples plataformas y la posibilidad de utilizar las características de hardware del dispositivo. Una de las desventajas es que, al utilizar la misma interfaz para todas las plataformas, la apariencia de la aplicación no será como la de una aplicación nativa. Finalmente, la ejecución será más lenta que la ejecución en una aplicación nativa.

1.8 El estilo de las interfaces.

La interfaz de una aplicación es como la ropa que viste. Es también la capa que hay entre el usuario y el corazón funcional de la app, el lugar donde nacen las interacciones. En mayor medida está compuesta por botones, gráficos, íconos y fondos, que tienen una apariencia visual diferente en cada uno de los sistemas operativos, porque Android, iOS y Windows Phone tienen su propia forma de entender el diseño. El trabajo del diseñador consiste en interpretar la personalidad de cada sistema operativo, aportando su propia visión y estilo de diseño, para conseguir aplicaciones que, además de ser fáciles de usar, sean distintas a las demás y tengan coherencia visual con la plataforma que las acoge (Farman, 2013).

1.8.1 Interfaces nativas o personalizadas.

Las **interfaces nativas** se basan en elementos propios de cada plataforma como botones, listas y encabezados. Estas tienen un aspecto ya definido en cuanto a las características básicas de su apariencia como color, tamaño o tipo de fuente, que pueden ajustarse en mayor o menor medida para que se correspondan con la estética buscada.

El inconveniente de las interfaces nativas es que limitan la personalidad del diseño y, en algunos casos, es necesario ir un paso más allá. En situaciones como esta, todos o algunos elementos de la interfaz pueden ser personalizados, lo cual se logra creándolos de nuevo como imágenes.

Diseñar una **interfaz personalizada** tiene que planearse de antemano porque representa una mayor complejidad y tiempo de desarrollo. De la misma manera, no siempre los diseños de este tipo de interfaces se trasladan a la aplicación funcional de forma fidedigna, pues su correcta implementación queda en manos de la pericia del desarrollador.

Además del objetivo o del tipo de aplicación, hay otras variables para considerar. Las interfaces nativas tienen un punto a favor y es que están constituidas por elementos que el usuario ya conoce y a los que está habituado, por lo tanto, no representan un nuevo aprendizaje. Esto puede incidir favorablemente en la usabilidad de la app.

Por otro lado, las interfaces personalizadas pueden ofrecer una apariencia más acabada, pero al trabajar con ellas hay que considerar la compatibilidad con múltiples dispositivos

1.9 Estado del Arte de los Servicios Web.

Un servicio web ***(en inglés, Web Service o Web services)*** es una tecnología que utiliza un conjunto de protocolos y estándares que sirven para intercambiar datos entre aplicaciones. Distintas aplicaciones de software desarrolladas en lenguajes de programación diferentes, y ejecutadas sobre cualquier plataforma, pueden utilizar los servicios web para intercambiar datos en redes de ordenadores como Internet. La interoperabilidad se consigue mediante la adopción de estándares abiertos.

1.9.1 Arquitecturas.

Para el desarrollo del web service se tendrán en cuenta para su estudio las siguientes técnicas de arquitectura:

> **SOAP (Simple Object Access Protocol):** Es un protocolo estándar que define como dos objetos en diferentes procesos pueden comunicarse por medio de intercambios de datos XML, el punto identificativo de SOAP es que las operaciones son definidas como puertos WSDL(Web Services Description Language).Es por esto que será aconsejable utilizar este protocolo en entornos donde se establecerá un contrato formal y donde se describirán todas las funciones de la interfaz así como los tipos de datos utilizados tanto de entrada como de salida. El lenguaje WSDL permitirá definir claramente cualquier detalle de las funciones del web service (Curbera *et al.*, 2002).

REST (Representational State Transfer) es un estilo de arquitectura de software para sistemas hipermedias distribuidos tales como la Web (Navarro Marset, 2006).

A diferencia de SOAP, se centra en el uso de los estándares HTTP y XML para la transmisión de datos sin la necesidad de contar con una capa adicional. Las operaciones o funciones se solicitarán mediante los métodos GET, POST, PUT y DELETE, por lo que no requiere de implementaciones especiales para consumir estos servicios. Además, se podrá utilizar JSON en vez de XML como contenedor de la información, por lo que será aconsejable utilizar este protocolo cuando se busque mejorar el rendimiento, o cuando se disponga de escasos recursos, como sería el caso de los dispositivos móviles.

1.9.2 Ventajas de la arquitectura REST.

Entre las ventajas principales investigadas acerca de una arquitectura REST, se tiene las siguientes:

- Es muy ligero, las respuestas del web service enviadas a un cliente contienen exactamente la información que se necesita (datos planos), pues utiliza pocos recursos.
- Es una arquitectura sin estado, lo cual significa que cada petición al servidor es tratada de manera totalmente independiente.
- Utiliza mucho más la cache en el cliente que en el servidor, pero proporcionan una buena infraestructura de almacenamiento en caché a través de HTTP método GET (para la mayoría de los servidores).
- Es sencillo de desarrollar y no se necesita mucho código extra.
- Es flexible para sus respuestas, estas pueden estar en formato XML o JSON.
- Es fácil y simple de interpretar, no hay herramientas costosas que se requiera para utilizar con los servicios web.
- Es fácil de integrar con los sitios web existentes.

1.9.3 Lenguajes de Programación.

Uno de los aspectos que se debe tener en cuenta cuando se desea desarrollar un servicio web es el lenguaje para su implementación. Existen variedad de lenguajes para la implementación de servicios web, a continuación, se expondrán los principales lenguajes con sus características y ventajas.

PHP

PHP (acrónimo de "PHP: Hypertext Preprocessor") es un lenguaje "*open source*" interpretado de alto nivel embebido en páginas HTML y ejecutado en el servidor (Bakken *et al.*, 1997).

PHP es uno de los lenguajes más populares de programación web, es conocido como un lenguaje basado en servidores. Esto es porque el PHP no se ejecuta en el cliente, sino en el lado del servidor. Las peticiones la realizan los clientes, luego la solicitud es procesada en el servidor y finalmente se envía la respuesta al cliente.

Entre las características principales se tiene las siguientes:

Código abierto: esto significa que está disponible completamente gratis.

Multiplataforma: es decir permite operar en varios sistemas operativos tales como LINUX, UNIX y Windows.

Soporte: al ser un programa muy popular, existe una comunidad muy grande, con referencias, guías, foros, entre otros disponibles en la web.

Altos retornos: Permite crear páginas web dinámicas. Esto asegura mayor participación de los visitantes y por lo tanto mayores retornos.

Extensiones: tiene múltiples extensiones y es extremadamente escalable.

Confiable y Seguro: El código fuente escrito en PHP es invisible al navegador y al cliente ya que es el servidor el que se encarga de ejecutar el código y enviar su resultado HTML al navegador.

PHP se puede ejecutar en el conocido servidor web Apache, en el cual se tienen que instalar las dependencias correspondientes.

Python

Python es un lenguaje de programación poderoso y fácil de aprender. Cuenta con estructuras de datos eficientes y de alto nivel y un enfoque simple pero efectivo a la programación orientada a objetos. La elegante sintaxis de Python y su tipado dinámico, junto con su naturaleza interpretada, hacen de éste un lenguaje ideal para scripting y desarrollo rápido de aplicaciones en diversas áreas y sobre la mayoría de las plataformas (Van Rossum and Drake Jr, 2009).

Entre sus características principales, se puede nombrar las siguientes:

Interpretado: Todo lo escrito en python será ejecutado por medio de un intérprete, a diferencia de los lenguajes compilados (C, C++, C#, Visual Basic, Delphi, etc.) que son ejecutados por medio de un compilador.

Multiplataforma: Se puede ejecutar los scripts en cualquier plataforma, tales como GNU/Linux, Microsoft Windows o Mac OSX.

Interactivo: Se puede ejecutar sentencias desde la línea de comandos y ver qué responde el intérprete (es de gran ayuda para comprender mejor el lenguaje y, principalmente, al momento de programar, permitiendo probar segmentos específicos del código en desarrollo, en busca de errores).

Extensiones: Implementa una gran cantidad de bibliotecas disponibles en la web.

Python se puede ejecutar al igual que PHP, en un servidor web Apache, para ello se instalan las dependencias correspondientes.

JavaScript

JavaScript es un lenguaje de programación que se utiliza principalmente para crear páginas web dinámicas. Una página web dinámica es aquella que incorpora efectos como texto que aparece y desaparece, animaciones, acciones que se activan al pulsar botones y ventanas con mensajes de aviso al usuario. Técnicamente, JavaScript es un lenguaje de programación interpretado, por lo que no es necesario compilar los programas para ejecutarlos. En otras

palabras, los programas escritos con JavaScript se pueden probar directamente en cualquier navegador sin necesidad de procesos intermedios (Pérez, 2009).

Cabe resaltar que JavaScript inicialmente se ejecutaba en el cliente, sin embargo, gracias al proyecto Node.js es posible ejecutarlo en el servidor.

A continuación, se puede observar las ventajas más destacadas de JavaScript:

- Es un lenguaje muy sencillo.
- Es rápido, por lo tanto, tiende a ejecutar las funciones inmediatamente.
- Cuenta con múltiples opciones de efectos visuales.
- Es soportado por los navegadores más populares y es compatible con los más modernos, incluyendo iPhone, móviles y PS3.
- Es muy versátil, puesto que es muy útil para desarrollar páginas dinámicas y aplicaciones web.
- Es una buena solución para poner en práctica la validación de datos en un formulario.
- Es multiplataforma, puede ser ejecutado de manera híbrida en cualquier sistema operativo móvil.

JavaScript ha logrado un gran crecimiento desde que es posible trabajar del lado del servidor gracias al proyecto Node.js creado en el año 2009 por Ryan Dahl.

Node.js un intérprete JavaScript del lado del servidor que cambia la noción de cómo debería trabajar un servidor. Su meta es permitir a un programador construir aplicaciones altamente escalables y escribir código que maneje decenas de miles de conexiones simultáneas en una sólo una máquina física (Abernethy, 2016).

Utilizando un servidor web tradicional, como pueda ser Apache, cada vez que se solicita un recurso web, se crea un hilo separado, o invoca un nuevo proceso, para atender dicha solicitud, a pesar de que Apache responde rápidamente a las solicitudes, y limpia el proceso una vez que ha terminado, este sistema puede necesitar varios recursos, por decir una aplicación web con muchas visitas, puede tener serios problemas de rendimiento. Node.js, por el contrario, utiliza un sistema de E/S asíncrono y basado en eventos y callbacks de funciones. Es decir, todo lo realiza en un único hilo: se queda escuchando a ciertos eventos que ocurran en la aplicación, y cuando ocurren, responde en consecuencia, de modo

asíncrono, es decir, un evento no bloquea a otro, dicho de otro modo, crea miles de subprocesos y trata a sus outputs como streams, esto supone una mayor velocidad de respuesta.

1.10 Conclusiones parciales.

En consecuencia, con las temáticas investigadas, se opta por desarrollar un servicio web con arquitectura REST, escrito en JavaScript y como intérprete del mismo se utilizará Node.js. Se ha optado por desarrollar una aplicación nativa para dispositivos móviles con sistema operativo Android ya que en este caso todo depende de las funcionalidades del dispositivo, además se necesita que la aplicación siga funcionando, aunque no se tenga conexión a la red. Se utilizará Mobile-D como metodología de desarrollo y Android Studio como IDE para su programación.

Capítulo 2: Análisis, diseño e implementación.

Durante este capítulo realizará un análisis de los principales requerimientos con los que deben cumplir el servicio web y la aplicación que se van a desarrollar para posteriormente pasar a su diseño e implementación.

2.1 Estudio, diseño y desarrollo del *web service*.

Para el desarrollo del servicio web será necesario condicionar la PC de desarrollo con los softwares necesarios para comenzar con la implementación, a continuación, se detallan los softwares que fueron instalados, con que finalidad y los pasos para su instalación.

Instalación del Open Conference Systems.

Como se ha mencionado anteriormente este sistema ya está instalado en un servidor en el data center de la UCLV, sin embargo se hace necesario instalarlo en la PC de desarrollo, con el fin de realizar todas las pruebas a nivel local y sobre todo que no sea necesario estar conectado a la red de la UCLV.

A continuación, se detallan los requerimientos y pasos para su instalación:

Se instalará Open Conference System en su versión 2.3.6 descargada desde la página oficial de PKP http://pkp.sfu.ca/ocs_download .

Requisitos del sistema

PHP >= 4.2.x (incluido PHP 5.x)

MySQL >= 3.23.23 (incluido MySQL 4.x) o PostgreSQL >= 7.1 (incluido PostgreSQL 8.x)

Apache >= 1.3.2x o >= 2.0.4x o Microsoft IIS 6

Sistema operativo: cualquiera que soporte el software anterior, incluyendo Linux, BSD, Solaris, Mac OS X, Windows.

El Open Conference System será instalado en una máquina con sistema operativo Windows, en la cual se ha instalado el xampp en su versión **5.6.20** descargado desde https://sourceforge.net/projects/xampp/files/XAMPPWindows/5.6.20 dicho paquete instala PHP en su versión 5.6.20, MariaDB[2] en su versión 10.1.13 y Apache en su versión 2.4.17.

Una vez cumplidos los requisitos del sistema se procede a la instalación del Open Conference System en el mismo.

Se descomprime dentro de la carpeta htdocs del xampp el fichero ocs-2.3.6.tar.gz descargado, en el caso de la PC de instalación dicha ruta es C:\xampp\htdocs.

Luego se deben iniciar los servidores Apache y MySQL, para ello se ejecuta el archivo xampp-control.exe que se encuentra en la carpeta de instalación del xampp, una vez hecho esto se visualizara la siguiente ventana donde se pueden iniciar los servidores.

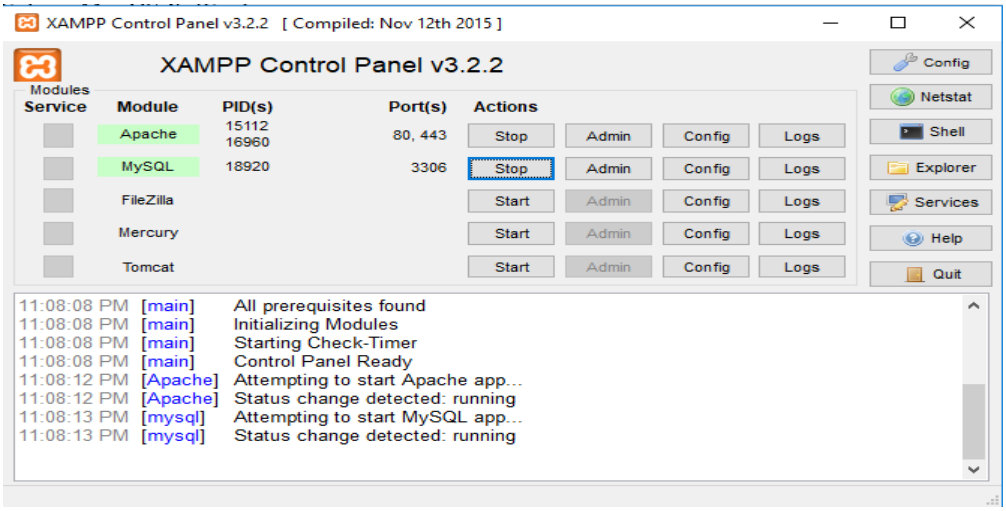

Figura 2.1: *Panel de Control del XAMPP.*

Pasos para la Pre-Instalación

[2] MariaDB es un sistema de base de datos que proviene de MySQL, pero con licencia GPL, desarrollado por Michael Widenius, fundador de MySQL y la comunidad de desarrolladores de software libre.

1. Los siguientes archivos y directorios (y sus contenidos) deben tener permisos de escritura:

- config.inc.php permiso de escritura (opcional): NO
- public/ permiso de escritura: Sí
- cache/ permiso de escritura Sí
- cache/t_cache/ permiso de escritura: Sí
- cache/t_compile/ permiso de escritura: Sí
- cache/_db permiso de escritura: Sí

El siguiente paso es abrir el navegador y colocar la URL donde se aloja el sistema, en este caso localhost/ocs-2.3.6/ de esta manera se tendrá acceso al formulario de configuración del proceso de instalación. Una vez escogidas las configuraciones necesarias, se hace click en el botón Install Open Conference System y si todo ha ido bien se mostrará la página de confirmación como se muestra en la Figura 2.2 : *Página de Confirmación de instalación del Open Conference Systems*.

Figura 2.2 : *Página de Confirmación de instalación del Open Conference Systems*.

2.1.2 Instalación de Node.js

Para comenzar con la creación del servicio web se hace necesaria la instalación de node.js ya que como se ha mencionado anteriormente será el encargado de interpretar el código del mismo. La versión de node a instalar será la 8.9.0 descargada desde https://nodejs.org/es/download/releases/

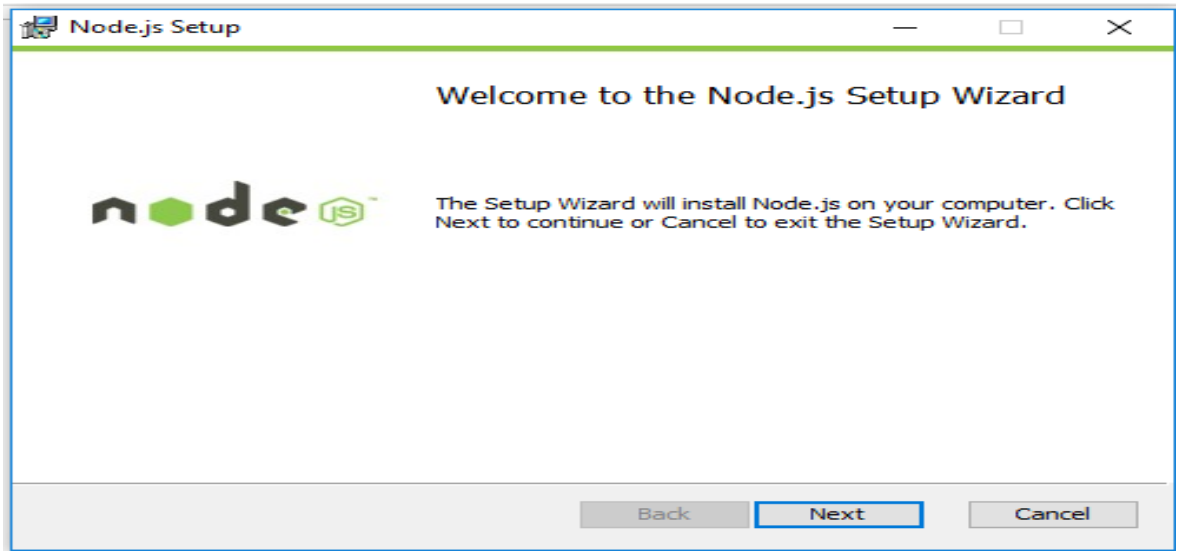

Figura 2.3: *Ventana inicial de instalación de Node.js.*

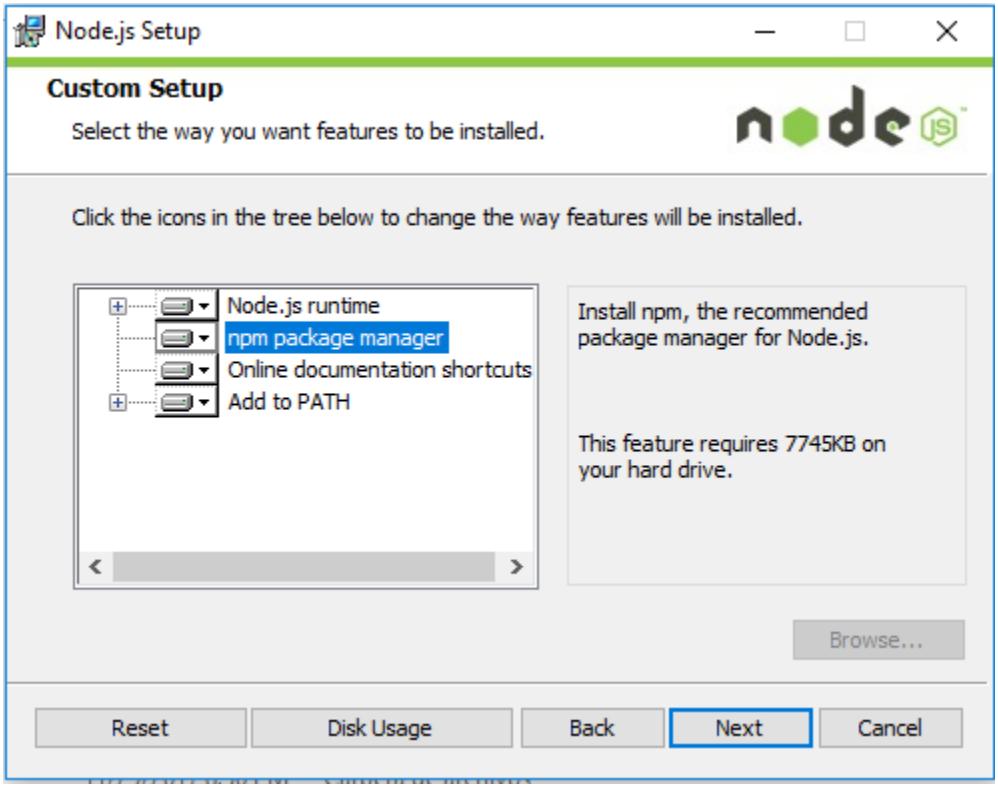

Figura 2.4: *Ventana de instalación de paquetes de Node.js.*

Es necesario destacar que node.js se instala conjuntamente con el gestor de paquetes “npm” el cual servirá para agregar módulos adicionales a node.js.

Una vez culminada la instalación haciendo uso del comando node –v se puede comprobar la versión de node que se ha instalado en el sistema, que en este caso ha sido la 8.9.0 como se muestra en la Figura 2.5: *Comprobación de versión de Node.js instalada.*.

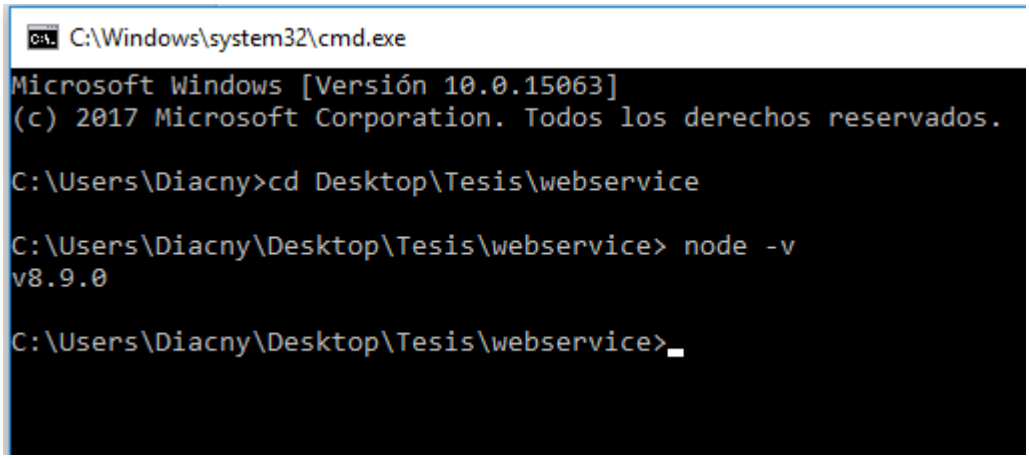

Figura 2.5: *Comprobación de versión de Node.js instalada.*

2.2 Creación del proyecto.

Para la creación del proyecto, se hace necesaria una carpeta principal en la que se guardarán todos los archivos correspondientes al código fuente del web service, dicha carpeta se llamará webservice y se ubicará en la ruta C:\Users\Diacny\Desktop\Tesis\

Luego se crea un archivo llamado server con extensión js que será el ejecutado para iniciar el servicio web.

Abrimos la ventana de comandos y ejecutamos el comando init para crear el archivo package.json.

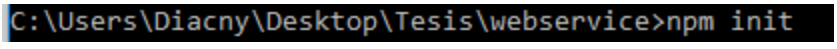

Figura 2.6: *Ejecución del comando init.*

Una vez completadas todas las preguntas se creará el archivo package.json con la siguiente estructura:

```
{
  "name": "web_service",
  "version": "1.0.0",
  "description": "intranet",
  "main": "app.js",

  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1",
    "start": "node server.js"
  },
  "author": "diacny",
  "license": "ISC"
}
```

Figura 2.7: *Estructura del archivo package.json* **.**

En el servidor de aplicaciones Node.js, se puede agregar una diversidad de módulos disponibles en internet y que se ajusten a cada una de las necesidades respectivas; por lo tanto, en el desarrollo del servidor se hará uso del módulo "express" para crear el web service en base a la arquitectura REST; y el módulo "mysql" para la conexión a la base de datos del Open Conference Systems.

Para su instalación se abre una terminal y se ubica en la ruta correspondiente al web service y mediante npm que es el gestor de paquetes de node.js se instalan las dependencias como se muestra en la Figura 1.14.

```
C:\Users\Diacny\Desktop\Tesis\webservice>npm install
```

Figura 2.8: *Ejecución del comando install.*

Luego de instalar las dependencias, en la carpeta principal del proyecto, se crea una carpeta llamada "node_modules", dentro de ella se encuentra una carpeta de nombre express la cual contiene el módulo que permite crear el web service con la arquitectura Rest, y otra carpeta de nombre mysql que contiene el módulo que permite la conexión del servicio web con la base de datos MySQL.

Conexión al Servicio Web y enlace a la base de datos.

Para la ejecución del servicio web es necesario definir una ip y un puerto de operación, por lo tanto, al estar en un entorno local la ip será la del localhost que en este caso es 127.0.0.1 y el puerto de operación será el 3002. En cuanto al enlace de la base de datos, recordando que se instaló un servidor local, se tiene la ip configurada como "localhost", el usuario por defecto "root" y la base de datos "ocs". Una vez definido los parámetros de configuración, se crean los archivos que contendrán las mismas. En la **¡Error! No se encuentra el origen de la referencia.** se muestra como queda conformado el arhivo db-mysqlconf.json para la conexión con la base de datos.

Figura 2.9: *Archivo db-mysqlconf.json para la conexión con la base de datos.*

2.3 Implementación del Web Service.

Para el desarrollo del código fuente se crea el archivo app.js, donde se incluyen las dependencias a los módulos express y mysql, que posibilitan la creación del servidor rest y la conexión a la base de datos, como se había mencionado anteriormente.

```
'use strict';

var express = require('express'),
    restFul = require('express-method-override')('_method'),

    app = express();

app
    //configurando app
    .use( restFul )

module.exports = app;

```

Figura 2.10: *Archivo app.json.*

Para las consultas a la base de datos se crea un archivo con el nombre webservice-model.js que es el que contiene las principales consultas realizadas a la misma. A continuación, se muestra dicho archivo con las consultas de los datos generales del evento y las fechas de inicio y fin.

```
'use strict';
var driverinstall = require('../../../config'),
    WebserviceModel = () => {};
var conn = require('../../../mysqlconnection');
WebserviceModel.getAll = (cb) => {
    let sql = `
    -- General Settings
    SELECT * FROM conference_settings
    WHERE setting_name IN ('title', 'description', 'contactEmail', 'homeHeaderLogoImage')
    ORDER BY conference_id, setting_name;

    -- Fechas inicio
    SELECT
        c.conference_id as conference_id, s.setting_value as start_date
    FROM
        sched_conf_settings s
        JOIN sched_confs c ON s.sched_conf_id = c.sched_conf_id
                WHERE s.setting_name = 'startDate';

    -- Fechas fin
    SELECT
        c.conference_id as conference_id, s.setting_value as end_date
    FROM
        sched_conf_settings s
        JOIN sched_confs c ON s.sched_conf_id = c.sched_conf_id
                WHERE s.setting_name = 'endDate';
        `;

    conn.query(sql, cb);
};
module.exports = WebserviceModel;
```

Figura 2.11: *Archivo webservice-model.js.*

Por último, se crea el archivo webservice-Controller.js que contiene las principales funciones para el procesamiento de los datos devueltos de las consultas anteriores.

```
145  function getFileFromSettings(baseUrl, searchString, settings) {
146      const pos = settings.search(searchString);
147      const endPos = settings.substring(pos).search(`"`)
148      const filename = settings.substring(pos, pos + endPos);
149      return baseUrl + filename;
150  };
151
```

Figura 2.12: *Función getFileFromSettings.*

La función getFileFromSettings () devuelve la ruta donde se encuentra el cronograma de actividades del evento.

2.4 Pruebas al Web Service.

Una vez implementado el web service se realizaron pruebas de inicialización y consultas para comprobar que el mismo funciona correctamente, conectándose a la base de datos y obteniendo los datos necesarios de la misma.

- La primera prueba realizada fue la de iniciar el web service, sabiendo de antemano que Node.js es el intérprete del código desarrollado para el web service, se abre una terminal en la ruta del mismo y se inicia la aplicación con el comando “node server.js”; donde se obtuvo lo siguiente:

```
C:\Users\Diacny\Desktop\Tesis\webservice>node server.js
Iniciando express en el puerto 3002
Conexión establecida con MySQL No: 6
```

Figura 2.13: *Iniciación del web service*

- Una vez iniciado el web service, se realiza la consulta a la base de datos, debido a que aún no se tiene la aplicación android desarrollada, se hará uso del navegador Mozilla Firefox para mostrar los resultados.

Figura 2.14: *Archivo json con los resultados obtenidos.*

2.5 Análisis, diseño e implementación de la aplicación móvil.

El proceso de desarrollo de una aplicación móvil puede ser segmentado a grandes rasgos en 4 fases: Análisis, Diseño, Implementación y Pruebas, a continuación, se detallará como se han llevado a cabo las 3 primeras fases en el desarrollo de la aplicación OCS-UCLV.

2.5.1 Análisis.

La fase de análisis es sumamente importante en el proceso de desarrollo de cualquier proyecto, es la que sirve como base para comprender y concretar el sistema y conocer los elementos que conforman el contexto del problema.

Los productos del análisis a desarrollar serán los siguientes:

1. Requisitos de la aplicación.
    - Requisitos funcionales.
    - Requisitos no funcionales.
2. Casos de uso.

Requisitos funcionales.

- La aplicación debe consultar la información disponible en la Plataforma para la gestión de eventos de la UCLV.

Requisitos no funcionales

- Smartphone o Tablet con sistema operativo Android 4.0.3 (Ice Cream Sandwich) o superior.
- **Rendimiento:** La aplicación debe desempeñar su función de manera fluida, buscando una experiencia de uso agradable al usuario.
- **Accesibilidad:** La aplicación debe de ser legible y tiene que seguir los patrones de accesibilidad de Google.
- **Disponibilidad:** La aplicación debe de estar disponible en el sitio Android UCLV.
- **Usabilidad:** La aplicación debe permitir el cambio de idioma, para que pueda ser usada por estudiantes extranjeros sin ningún tipo de restricción.
- **Estabilidad:** La aplicación debe ser capaz de manejar los errores ocurridos durante la ejecución de la misma, avisando al usuario la naturaleza del error.
- **Interfaz:** Clara y concisa, para no dar lugar a la confusión del usuario y debe seguir los estándares de diseño de interfaces de Google.
- **Integración:** Debe integrarse con todo el sistema operativo de Android. Hacer uso de aplicaciones nativas si se necesita y mantener un diseño acorde al sistema.
- **Optimización:** El consumo de batería y datos debe ser adecuado, y nunca dejar procesos sueltos que consuman memoria o batería. El tiempo de ejecución debe ser mínimo, para mejorar los tiempos de respuesta y la experiencia del usuario.

2.5.2 Casos de uso

Caso de Uso: Un caso de uso expresa todas las formas de usar un sistema para alcanzar una meta particular para un usuario. En conjunto, los casos de uso le proporcionan todos los caminos útiles de usar el sistema e ilustran el valor que este provee (Jacobson *et al.*, 2013).

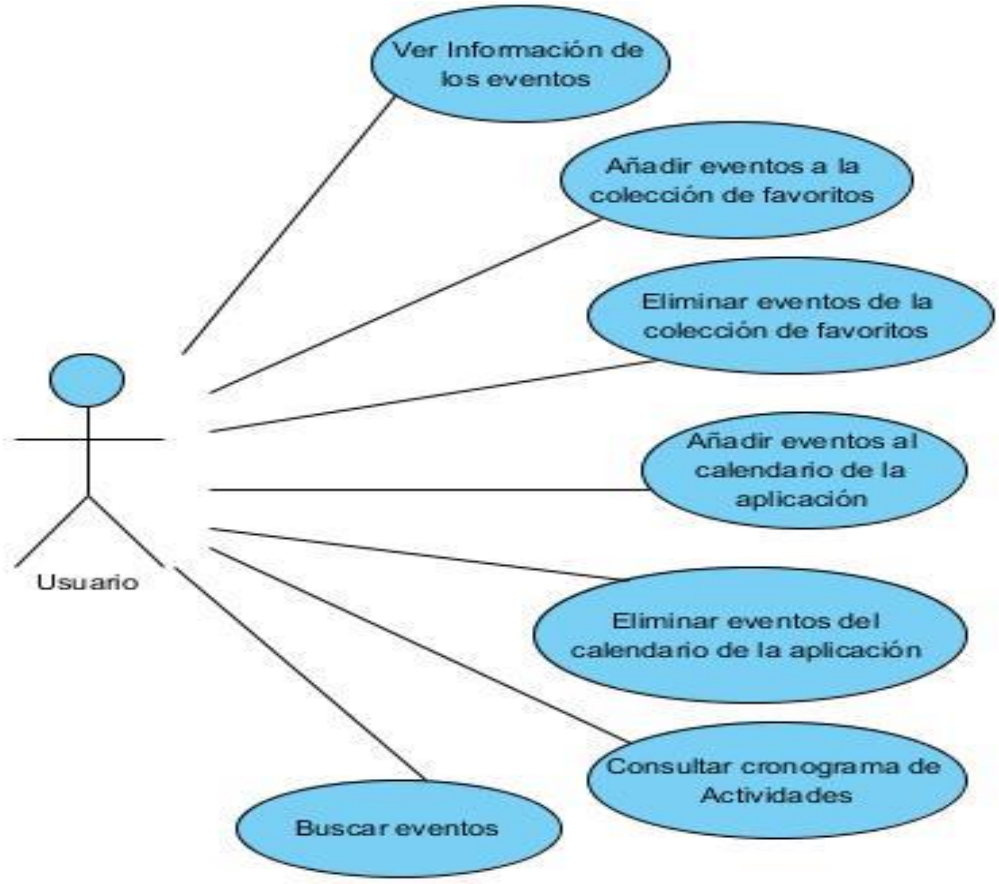

Figura 2.15: *Diagrama de casos de uso y actores del sistema.*

En el diagrama anterior Figura 2.15: *Diagrama de casos de uso y actores del sistema.* se ilustran las diferentes funcionalidades que permite la aplicación.

2.6 Diseño e implementación

Una vez determinadas las funcionalidades de la aplicación, se pasa a la fase de diseño. En esta fase, lo que se hace es crear toda la interfaz que compondrá la aplicación. Básicamente se realizará todo lo visual para que la app sea atractiva al usuario.

En la fase de implementación se dará vida a los diseños y se creará la estructura sobre la cual se apoyará el funcionamiento de la aplicación.

2.6.1 Diagrama de Actividades.

Los diagramas de actividades sirven para representar el comportamiento dinámico de un sistema haciendo hincapié en la secuencia de actividades que se llevan a cabo y las condiciones que guardan o disparan esas actividades (Falgueras, 2003).

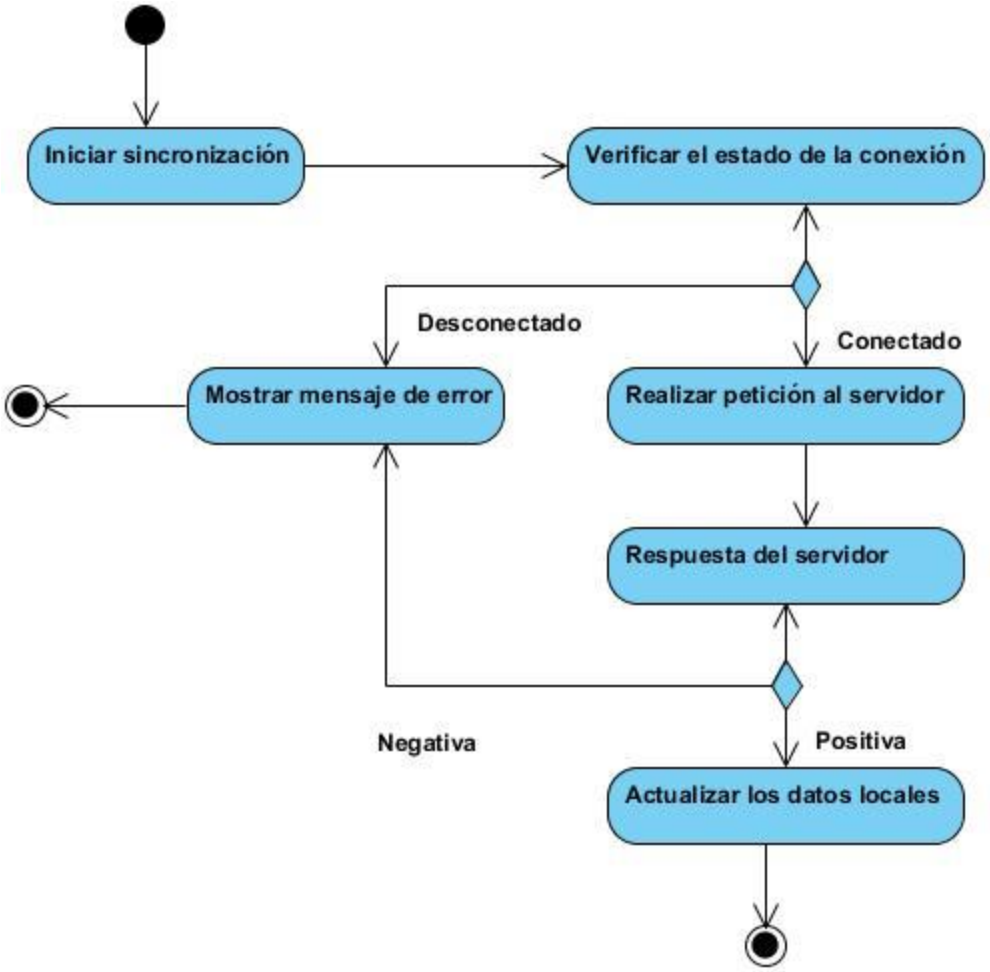

Figura 2.16: *Diagrama de actividades.*

El caso de uso **iniciar sincronización** se inicia automáticamente al iniciar la aplicación o cuando el usuario pulsa sobre el ícono de sincronización que se encuentra en la *toolbar* Primeramente, el sistema verifica el estado de la conexión, en caso de estar conectado realiza una petición al servicio web y espera una respuesta, si la respuesta es positiva dichos datos son procesados y almacenados en el dispositivo para lograr la persistencia de la información, en caso negativo se emite un mensaje de error informando al usuario que no se ha podido realizar la sincronización.

2.6.2 Diagrama de Secuencia.

Los Diagramas de Secuencias muestran la forma en que un grupo de objetos se comunican (interactúan) entre sí a lo largo del tiempo (Booch *et al.*, 1999).

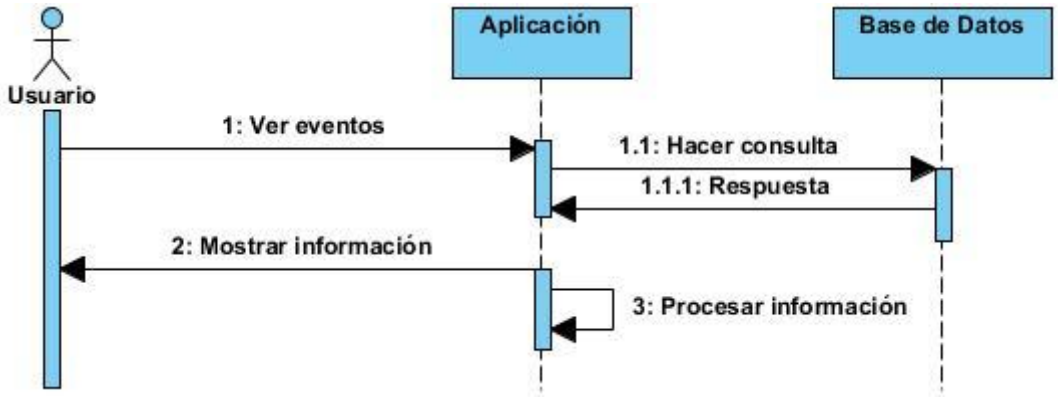

Figura 2.17: *Diagrama de secuencia.*

El diagrama describe el proceso de extracción de la información de la base de datos local para mostrarla posteriormente al usuario.

2.6.3 El diseño de la interfaz.

Con el objetivo de facilitar la usabilidad de la aplicación se ha diseñado una interfaz nativa, haciendo uso de los elementos preestablecidos en la plataforma.

Íconos interiores

Los íconos interiores tienen un papel menos estelar y más funcional que los de lanzamiento. Su uso suele estar asociado a tres escenarios:

1. Como ayuda visual para reforzar información, por ejemplo, en un cuadro de diálogo con una alerta.

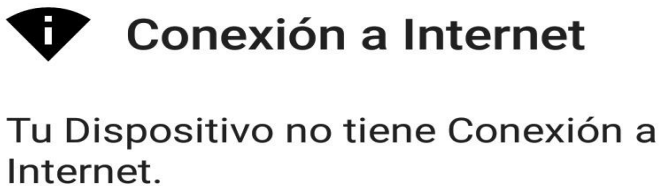

Figura 2.18: *Diálogo que se muestra al intentar sincronizar con el servidor cuando el disposito no tiene conexión.*

2. Mejorar la utilización del espacio, en este caso, el ícono resume visualmente algo que en forma de texto sería muy extenso o complejo de entender.

Figura 2.19: *Íconos de la Toolbar*.

Las acciones de sincronización y búsqueda han sido representadas con íconos con la intención de aprovechar el espacio disponible en la *Toolbar*, dichos íconos son conocidos por los usuarios ya que se asocian con estas acciones en la mayoría de las aplicaciones, por lo que facilitan la usabilidad.

Tipografía

El objetivo de la tipografía es conseguir que el texto se lea con claridad. Esto se logra no solo con una adecuada elección de la fuente, sino también gestionando su tamaño, separación entre líneas, ancho de columnas y contraste visual con el fondo.

La tipografía usada en la aplicación OCS-UCLV ha sido sans-serif con tamaños que van desde 14sp para textos secundarios hasta 22sp para los títulos, dicha tipografía ha sido jerarquizada, haciendo uso del color, utilizando para información relevante como es el caso de los títulos de los eventos colores con tonos más oscuros y menos opacidad y para el caso de información secundaria como las descripciones se usaron colores más claros y con mayor opacidad.

2.6.4 Implementación.

La implementación de aplicaciones para dispositivos móviles se asemeja mucho a la del resto de aplicaciones, aunque, generalmente, se trata de aplicaciones más pequeñas, o bien que tienen ciclos de desarrollo más cortos que las aplicaciones tradicionales. Esto se debe tanto a la propia naturaleza de la aplicación como a las necesidades del mercado que demanda conseguir prototipos o pruebas de concepto rápidas.(Vique, 2012b)

Se ha utilizado Java como lenguaje de programación y Android Studio como entorno de desarrollo debido a que es el actual IDE respaldado por Google y el que dispone de mejores herramientas para desarrollar aplicaciones para la plataforma Android.

Las herramientas necesarias para desarrollar utilizadas en esta aplicación son:

- Android Studio 2.2.3.
- Google Nexus 5 como dispositivo de prueba.

Para el desarrollo de la aplicación se ha tenido en cuenta que debe ser una aplicación intuitiva y fácil de usar. Para este objetivo se ha divido la aplicación en bloques de información:

- Ver Eventos
- Eventos de hoy
- Mi Calendario
- OCS-UCLV
- Cambiar Idioma
- Acerca de

Con el fin de conseguir una aplicación intuitiva se ha implementado el patrón de diseño con *Navigation Drawer*,un panel lateral que aconseja Google tener en todas las aplicaciones con muchos bloques de información.

El N*avigation Drawer* es un View o vista que actúa como menú para cambiar entre las distintos *Fragments* o pantallas de la aplicación. El *drawer* se oculta automáticamente cuando el usuario este usando la aplicación, por lo que permite a los *Fragments* ocupar el mayor tamaño de pantalla posible.

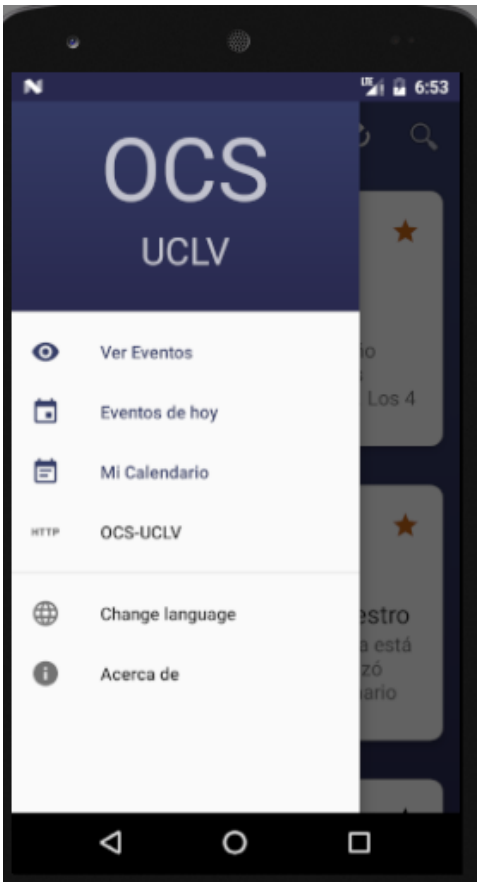

Figura 2.20: *Navigation Drawer de la aplicación*

Como se ha mencionado anteriormente, la aplicación se basa en el patrón de diseño *Drawer + Fragments.*

Para comprender el comportamiento principal de la aplicación es necesaria la definición de *Fragment.*

Un *Fragment* no puede considerarse ni una Actividad, ni una Vista, es algo parecido a un contenedor. Un *Fragment* es un pedazo de la interfaz de usuario que se puede añadir o eliminar de la interfaz global de usuario de forma independiente al resto de elementos de la Actividad. Esto permite que pueda reutilizarse en otras actividades (Robledo Fernández, 2014).

Por tanto, todo *fragment* necesita una actividad y una vista donde implementarse (Contenedor *fragment*). Este contenedor solo puede mostrar un *fragment* a la vez, pudiendo cambiar en tiempo de ejecución el *fragment* que se muestra.

Al hacer clic en un ítem del *drawer*, el *fragment* principal de la aplicación se cambia por el correspondiente *fragment* creado para cada uno de los ítems del *drawer*.

Sincronización.

El proceso de sincronización se inicia automáticamente al iniciar la aplicación a través de una llamada al método inicializarSyncAdapter () de la clase SyncAdapter o manualmente al pulsar en el ícono de sincronización que se encuentra en la *Toolbar*.

Para poder realizar la sincronización es necesario verificar si hay conexión disponible en el dispositivo esto se realiza utilizando *ConnectivityManager*, una biblioteca de Android que permite acceder a los servicios de red del dispositivo, obtener información de estos y saber si el dispositivo se encuentra conectado o desconectado. Si la respuesta es afirmativa se inicia entonces el proceso de sincronización. En la **¡Error! No se encuentra el origen de la referencia.** se muestra como ocurre dicho

```
public void realizarSincronizacionLocal(final SyncResult syncResult) {
    Log.i(TAG, "Actualizando el cliente.");
    VolleySingleton.getInstance(getContext()).addToRequestQueue(
            new StringRequest(
                    Request.Method.GET,
                    Constantes.GET_URL,
                    new Response.Listener<String>() {
                        @Override
                        public void onResponse(String response) {
                            try {
                                JSONObject jsonObject = new JSONObject(response);
                                procesarRespuestaGet(jsonObject, syncResult);
                            } catch (JSONException e) {
                                e.printStackTrace();
                            }

                        }
                    },
                    new Response.ErrorListener() {
                        @Override
                        public void onErrorResponse(VolleyError error) {
                            showAlertDialog(getContext(), "Error", "No se puede acceder a " + Constantes.GET_URL);
                        }
                    }
            )
    );
}
```

Figura 2.21: *Método realizarSincronizacionLocal*

La petición http se realiza utilizando Volley la cual es una biblioteca para hacer peticiones HTTP que está desarrollada por Google. Entre sus ventajas se destacan la programación

automática de peticiones, las conexiones de red concurrentes múltiples, el API de cancelación de peticiones y la posibilidad de personalización para programar reintentos (Fernández Riolobos and others, 2017).

```
public void actualizarDatosLocales(JSONObject response, SyncResult syncResult) {
    JSONArray eventos = null;
    try {
        // Obtener array "eventos"
        eventos = response.getJSONArray("data");
    } catch (JSONException e) {
        e.printStackTrace();
    }
    // Parsear con Gson
    Eventos[] res = gson.fromJson(eventos != null ? eventos.toString() : null, Eventos[].class);
    List<Eventos> newBD = Arrays.asList(res);

    ConferenceDao dao = new ConferenceDao();
    List<Conference> oldBD = dao.allConferences();

    // la nueva lista ya actualizada
    List<Conference> merge = new ArrayList<>();

    for (Eventos evento : newBD) {
        Conference conference = getConference(evento);
        merge.add(conference);
    }
    new ConferenceDao().deleteAllConferences();
    List<Conference> con = new ConferenceDao().allConferences();
    Log.e("count", con.size() + "");

    for (Conference conference : merge) {
        dao.insertConference(conference);
    }
}
```

Figura 2.22: *Método actualizarDatosLocales***.**

En el método actualizarDatosLocales() es donde se parsean los resultados obtenidos,los cuales son un a arreglo en formato json, a un arreglo de eventos haciendo uso de la biblioteca Gson, posteriormente estos datos son almacenados en una base de datos sqlite haciendo uso de la biblioteca requery.

Una de las funcionalidades que brinda la aplicación es la activación de alarmas para ser notificado el día del evento, para la creación de las mismas se usó la clase AlarmManager, la cual permite acceder al servicio de alarmas del sistema, cuya principal funcionalidad es poder programar operaciones, en un tiempo establecido**,** fuera del ciclo de vida de la aplicación.

En la Figura 2.23: *Método que cancela una alarma establecida.* se muestra el uso de esta clase para la cancelación de una alarma.

```
public void cancelAlarm(Conference conference) {
    AlarmManager alarmManager = (AlarmManager) context.getSystemService(Context.ALARM_SERVICE);
    Intent intent = new Intent(context, AlarmReceiver.class);
    String idPrefijo = conference.getId();
    int id = Integer.parseInt(idPrefijo);
    PendingIntent pendingIntent = PendingIntent.getBroadcast(context, id, intent, PendingIntent.FLAG_ONE_SHOT);
    alarmManager.cancel(pendingIntent);
}
```

Figura 2.23: *Método que cancela una alarma establecida.*

Capítulo 3: Pruebas al software.

Para evitar la insatisfacción del cliente y brindar un producto de calidad se deben de realizar pruebas de software, las cuales se definen como una actividad que ejecuta circunstancias previamente especificadas, donde los resultados se observan y registran para realizar una evaluación (Araya Solís, Méndez Marín and Jiménez Segura, 2014).

Una vez generado el código fuente, el software debe ser probado para descubrir (y corregir) el máximo de errores posibles antes de su entrega al cliente. Se deben diseñar una serie de casos de prueba que tengan una alto probabilidad de encontrar errores. Para conseguirlo se aplican las técnicas de pruebas del software. Estas técnicas facilitan una guía sistemática para diseñar pruebas que:

1. Comprueben la lógica interna de los componentes software.
2. Verifiquen los dominios de entrada y salida del programa para descubrir errores en la funcionalidad, el comportamiento y rendimiento

El principal objetivo del diseño de casos de prueba es obtener un conjunto de pruebas que tengan la mayor probabilidad de descubrir los defectos del software. Para llevar a cabo este objetivo, se usan dos categorías diferentes de técnicas de diseño de casos de prueba: prueba de caja blanca y prueba de caja negra (Pressman, 2010) .

La prueba de caja blanca es un método de diseño de casos de prueba que usa la estructura de control del diseño procedimental para obtener los casos de prueba. Mediante los métodos de prueba de caja blanca, se pueden obtener casos de prueba que:

1. Garanticen que se ejercita por lo menos una vez todos los caminos independientes de cada módulo.
2. Ejerciten todas las decisiones 1ógicas en sus vertientes verdadera y falsa.
3. Ejecuten todos los bucles en sus límites y con sus límites operacionales.
4. Ejerciten las estructuras internas de datos para asegurar su validez.

Las pruebas de caja blanca se centran en la estructura de control del programa. Se obtienen casos de prueba que aseguren que durante la prueba se han ejecutado, por lo menos una vez, todas las sentencias del programa y que se ejercitan todas las condiciones 1ógicas.

Las pruebas de caja negra, se centran en los requisitos funcionales del software. 0 sea, la prueba de caja negra permite obtener conjuntos de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa. La prueba de caja negra no es una alternativa a las técnicas de prueba de caja blanca, más bien se trata de un enfoque complementario que intenta descubrir diferentes tipos de errores que los métodos de caja blanca. La prueba de caja negra intenta encontrar errores de las siguientes categorías: (Pressman, 2010)

1. Funciones incorrectas o ausentes.
2. Errores de interfaz.
3. Errores en estructuras de datos o en accesos a bases de datos externas.
4. Errores de rendimiento.
5. Errores de inicialización y de terminación.

Las pruebas de caja negra amplían el enfoque. Son diseñadas para validar los requisitos funcionales sin fijarse en el funcionamiento interno de un programa. Las técnicas de prueba de caja negra se centran en el ámbito de información de un programa, de forma que se proporcione una cobertura completa de prueba.

3.1 Pruebas 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 realizaron pruebas de integración en las diferentes pantallas de la aplicación para comprobar que cada uno de los componentes cumple con la función esperada, como se describe en las Figura 3.1: *Prueba de integración a la pantalla principal.* y Figura 3.2: *Prueba de integración a la pantalla de detalles del evento.*

Figura 3.1: *Prueba de integración a la pantalla principal.*

Figura 3.2: *Prueba de integración a la pantalla de detalles del evento.*

3.2 Pruebas de validación.

La validación es un conjunto diferente de tareas que aseguran que el software que se construye sigue los requerimientos del cliente (Pressman, 2010).

La aplicación tiene como objetivo extraer la información de los eventos disponibles en la Plataforma de Gestión de Eventos de la UCLV y mostrarla en el dispositivo, además de hacerla persistente en el mismo para poder visualizarla en situaciones en las que no se cuente con conexión a la red. OCS-UCLV cumple con dichos requerimientos ya que la misma realiza peticiones http al servicio web encargado de posibilitar la extracción de la información de la Plataforma, una vez extraída la información es devuelta a la aplicación en un archivo con formato json, el cual es procesado en la aplicación y dicho contenido es mostrado al usuario de una forma diferente a la Plataforma haciéndola más atractiva, además permite la creación de alarmas que se activan al marcar un evento como favorito.

3.3 Pruebas de usabilidad:

Medida en que un producto puede ser usado por determinados usuarios para conseguir objetivos específicos con efectividad, eficiencia y satisfacción en un contexto de uso específico (Standard, 1998).

Para un mejor entendimiento de dicho concepto veamos las definiciones formales de estos conceptos según (Frekjmr, 2000)

Efectividad: Está relacionada con la precisión y completitud con la que los usuarios utilizan la aplicación para alcanzar objetivos específicos. La calidad de la solución y la tasa de errores son indicadores de efectividad.

Eficiencia: Es la relación entre efectividad y el esfuerzo o los recursos empleados para lograr esta. Indicadores de eficiencia incluyen el tiempo de finalización de tareas y tiempo de aprendizaje. A menor cantidad de esfuerzo o recursos, mayor eficiencia.

Satisfacción: Es el grado con que el usuario se siente satisfecho, con actitudes positivas, al utilizar la aplicación para alcanzar objetivos específicos. La satisfacción es un atributo subjetivo, puede ser medido utilizando escalas de calificación de actitud.

Existen diferentes opiniones sobre que atributos pueden ser considerados y la manera de combinarlos para componer la usabilidad. Los siguientes son algunos de los atributos utilizados para medir el grado de usabilidad de una aplicación de software (Folmer and Bosch, 2004):

Facilidad de Aprendizaje: La facilidad con la que los usuarios alcanzan objetivos específicos la primera vez que utilizan la aplicación. La primera experiencia que tiene los usuarios con un nuevo sistema es la de aprender a usarlo.

Memorabilidad: La facilidad para memorizar la forma de utilizar la aplicación y alcanzar objetivos específicos, y la facilidad con que vuelven a utilizar la aplicación después de un tiempo. La curva de aprendizaje debe ser significativamente menor para un usuario que ya utilizó el sistema, que para uno que es la primera vez que lo va a utilizar.

Errores: Los errores que comete el usuario al utilizar la aplicación y la gravedad de los mismos. La aplicación debe producir la menor cantidad de errores posibles. Si se producen, es importante que se den a conocer al usuario de forma rápida y clara, además de ofrecer

algún mecanismo para recuperarse de ese error. Contenido: Aspectos relacionados a la distribución del contenido y de los formatos utilizados para mostrar información al usuario.

Accesibilidad: Consideraciones tenidas en cuenta por posibles limitaciones físicas, visuales, auditivas o de otra índole de los usuarios.

Seguridad: Capacidad para alcanzar niveles aceptables de riesgo. Disponibilidad de mecanismos que controlan y protegen la aplicación y los datos almacenados.

Portabilidad: Capacidad de la aplicación de ser transferida de un entorno a otro (diferentes plataformas).

Contexto: Relacionado a los factores o variables del entorno de uso de la aplicación.

3.4 Pruebas de satisfacción de usuario.

Otro factor a analizar para saber si la aplicación cumple con los requisitos necesarios de cara al cliente es realizar pruebas de satisfacción. La población utilizada para estas pruebas constaba de un equipo de 15 personas, quienes instalaron la aplicación en sus teléfonos e hicieron uso de todas sus funcionalidades.

Una vez probada la aplicación se realizó una encuesta a los usuarios, la cual arrojó los siguientes resultados:

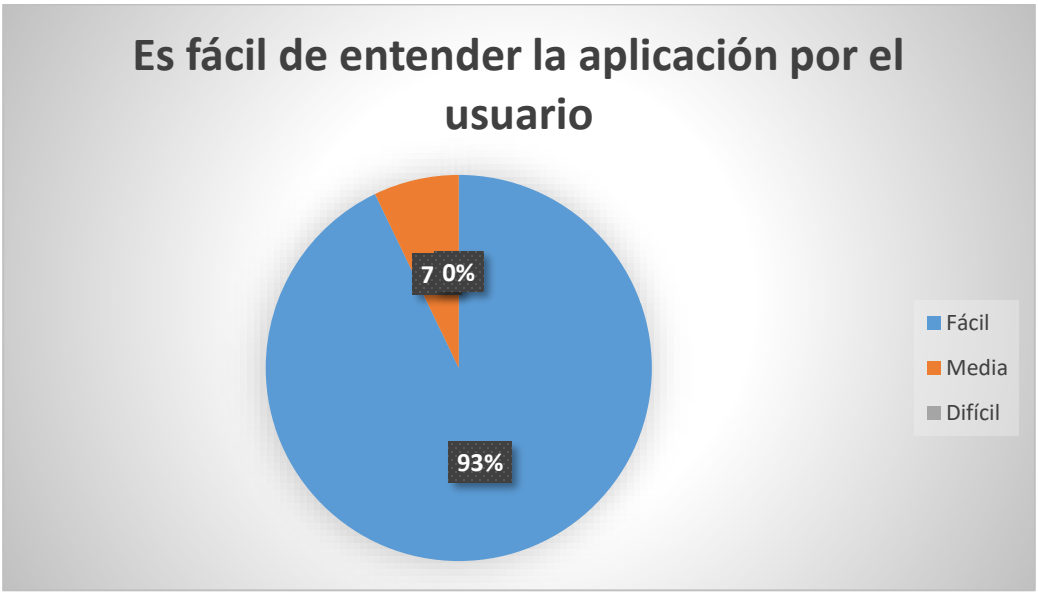

Figura 3.3: *Gráfica que muestra los resultados de aplicar la primera pregunta de la encuesta.*

Cabe destacar que los usuarios que realizaron las pruebas no tenían un conocimiento previo de la aplicación, ni se les proporcionó manual de usuario, aun así, los resultados obtenidos fueron satisfactorios siendo calificada de fácil por el 93% de los encuestados, medianamente fácil por un 7% y ningún usuario la calificó de difícil.

Otro de los aspectos a analizar en la encuesta fue el diseño de la interfaz, la cual fue de las mejores puntuadas ya que el 100% de los usuarios la calificaron de atractiva.

Figura 3.4: *Gráfica que muestra los resultado de aplicar la segunda pregunta de la encuesta.*

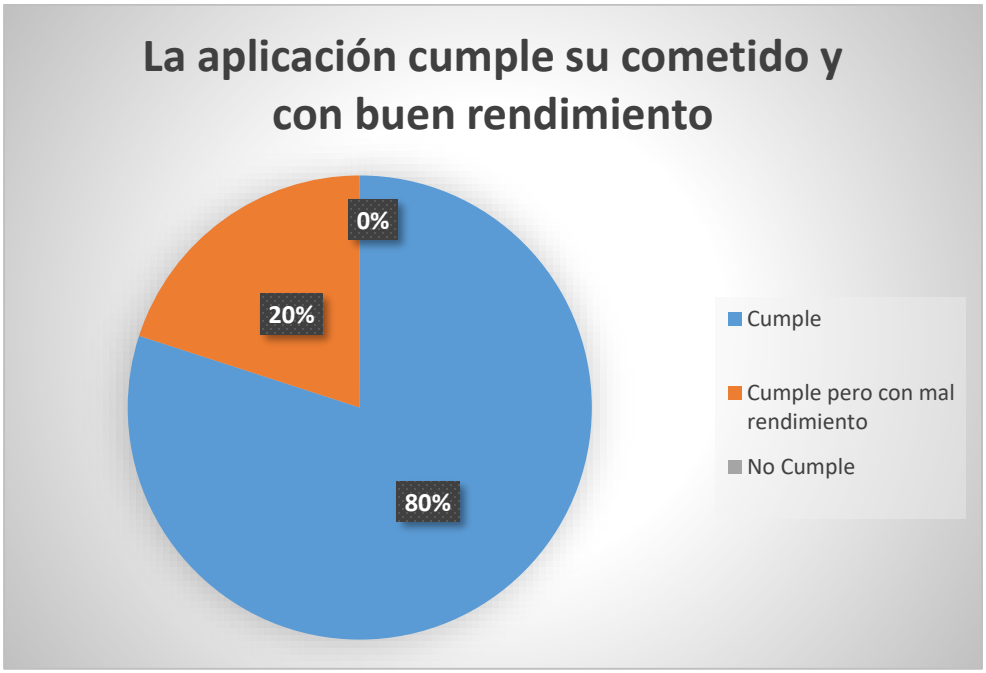

Figura 3.5: *Gráfica que muestra los resultados de aplicar la tercera pregunta de la encuesta.*

Con esta pregunta se pretendía evaluar si la aplicación cumple con los requisitos funcionales y no funcionales para los que fue creada. En la figura se pude apreciar que de los usuarios encuestados un 80% considera que cumple con dichos requerimientos y solo un 20% considera que cumple pero con mal rendimiento, dicha respuesta puede estar condicionada por las características del dispositivo donde fue instalada la aplicación, o la velocidad de los tiempos de respuesta debido a la calidad de la señal de la red inalámbrica (Wi-Fi).

3.5 Pruebas de compatibilidad.

La aplicación está diseñada para funcionar en cualquier *Smartphone* o *Tablet* con sistema operativo Android 4.0.3 (*Ice Cream Sandwich*) o superior, por lo que fue probada en varios dispositivos con diferentes características de *hardware* y *software* como se muestra en la siguiente tabla:

	- Google Nexus 5 - Android :4.4 - CPU: Qualcomm MSM8974 Snapdragon 800 - Memoria RAM: 2 GB - Almacenamiento Interno:32 GB - Pantalla táctil de 5" 1920 x 1080 píxeles.
	- Galaxy Express 3 SM-J120A - Android 6.0.1 Marshmallow - CPU 1.3Ghz Quad-Core ARM Cortex-A - Memoria RAM: 1GB - Almacenamiento interno : 8GB - Pantalla táctil 4.5" pulgadas 480 x 800 píxeles
	- Blu Dash J - Android 4.4 - KitKat - CPU MediaTek Dual-Core 1.3GHz + ARM Mail 400 GPU - Memoria RAM de 256MB - Almacenamiento interno 512MB. - Pantalla táctil 4.0" 480x800, 4:3

	- bq Aquaris M10 - Android 5.1 - CPU MediaTek Quad Core MT8163B - Memoria RAM:2 GB - Almacenamiento Interno:16 GB - Pantalla táctil: 10,1" 1280 x 800
	- Samsung P4 - Android 5.1.1 - Lollipop - CPU ARM Cortex-A9 Dual-Core 1.6ghz + Nvidia Tegra GPU - Memoria RAM De 768MB - Almacenamiento Interno 16GB. - Pantalla Táctil 10.0" 800x1280 HD 16:9

En los *Smartphones* probados se pudo apreciar como la aplicación se ajusta de manera automática a las variaciones de tamaño de pantalla y densidad de pixeles, además al ser instalada en distintas versiones del sistema operativo la aplicación respondió como se esperaba ya que al instalarla en dispositivos con sistema operativo Android inferior a la versión 5.0 – *Lollipop*, se distingue ligeramente el cambio en la apariencia de los componentes, aunque no afecta directamente el diseño visual de la *app*.

En los *Tablets* la aplicación se comporta de manera debida ajustándose los componentes al tamaño de pantalla d estos dispositivos. También se probó en versiones distintas del sistema operativo Android para evaluar su comportamiento obteniéndose resultados satisfactorios.

3.6 Conclusiones parciales.

La fase de pruebas es sumamente importante en el proceso de desarrollo de una aplicación, para poder entregar al cliente un producto de calidad, con la menor cantidad de deficiencias posibles. La aplicación fue sometida a diversas pruebas donde se evaluaron cuestiones como la usabilidad de la misma, su compatibilidad con los diferentes dispositivos, el nivel de satisfacción de los usuarios, entre otras. En la mayoría de los casos los resultados fueron satisfactorios, aunque se detectaron aspectos mejorables en el caso del nivel de satisfacción de usuarios, cuestiones que hubiesen pasado desapercibidas de no haber aplicado dichas pruebas.

Conclusiones

Como resultado de la investigación se han arribado a las siguientes conclusiones:

- Se realizó un proceso de selección de información para identificar las temáticas de mayor interés para los usuarios, las cuales fueron mostradas en la aplicación.
- Se hizo un estudio de las principales arquitecturas, lenguajes y tecnologías utilizadas para el desarrollo de servicios web, determinado como mecanismo de extracción de información un servicio web con arquitectura REST, escrito en JavaScript y como intérprete del mismo se utilizó Node.js.
- Se realizó un estudio sobre los principales tipos de medios de persistencia en Android, escogiendo para utilizar en la aplicación bases de datos sqlite ya que es el que más se ajusta a las características de la información extraída y a los requerimientos de la aplicación.
- Se evaluó el desempeño de la aplicación mediante pruebas de integración, usabilidad, y compatibilidad comprobando el correcto funcionamiento de la misma.

Referencias

Abernethy, M. (2016) '¿Simplemente qué es NODE. JS?', *Visitado: Febrero*.

Abrahamsson, P. *et al.* (2004) 'Mobile-D: an agile approach for mobile application development', in *Companion to the 19th annual ACM SIGPLAN conference on Object-oriented programming systems, languages, and applications*, pp. 174–175.

Araya Solís, G., Méndez Marín, G. and Jiménez Segura, R. (2014) 'Pruebas de software para dispositivos móviles android'. UNAN-Managua.

Báez, M. *et al.* (2012) 'Introducción a android', *EME Madrid, España*, p. 121.

Bakken, S. S. *et al.* (1997) 'Manual de PHP', *Este manual es{\copyright} Copyright*.

Benbourahla, N. (2015) *Android 5: Principios del desarrollo de aplicaciones Java*. Ediciones Eni.

Booch, G. *et al.* (1999) *El lenguaje unificado de modelado*. Addison Wesley Madrid.

Curbera, F. *et al.* (2002) 'Unraveling the Web services web: an introduction to SOAP, WSDL, and UDDI', *IEEE Internet computing*. IEEE, 6(2), pp. 86–93.

Enriquez, J. G. and Casas, S. I. (2014) 'Usabilidad en aplicaciones móviles', *Informes Científicos-Técnicos UNPA*, 5(2), pp. 25–47.

Falgueras, B. C. (2003) *Ingeniería del software*. Editorial UOC.

Farman, J. (2013) *Mobile interface theory: Embodied space and locative media*. Routledge.

Fernández Riolobos, J. and others (2017) *Desarrollo de una aplicación Android que unifique eventos y actividades en la UAM (J-Event)*.

Folmer, E. and Bosch, J. (2004) 'Architecting for usability: a survey', *Journal of systems and software*. Elsevier, 70(1–2), pp. 61–78.

Frekjmr, E. (2000) 'Measuring Usability : Are Effectiveness , Efficiency , and Satisfaction Really Correlated ?', (February 2016).

Hernández Pérez, Y. (2016) 'Implementación del Sistema de Repositorios Digitales Institucionales en la UCLV mediante la integración de las plataformas en uso.'

Herrera Uribe, E. and Valencia Ayala, L. E. (2007) 'Del manifiesto ágil sus valores y principios', *Scientia Et Technica*. Universidad Tecnológica de Pereira, 13(34).

Jacobson, I. *et al.* (2013) 'Casos de uso 2.0', *La guía para ser exitoso con los casos de uso*.

Luna, R. M. C., Vaca, K. H. A. and Vásquez, D. A. P. (2017) 'Observaciones acerca de los dispositivos móviles', *Dominio de las Ciencias*. Polo de Capacitación, Investigación y Publicación (POCAIP), 3(4), pp. 89–103.

Lynch, C. A. (2003) 'Institutional repositories: essential infrastructure for scholarship in the digital age', *portal: Libraries and the Academy*. The Johns Hopkins University Press, 3(2), pp. 327–336.

Navarro Marset, R. (2006) 'Modelado, Diseño e Implementación de Servicios Web'.

Parker, M. (2013) 'The ethics of open access publishing', *BioMed Central*.

Pérez, J. E. (2009) 'introduccion a JavaScript', *Librosweb. es (editorial digital)*.

Pressman, R. S. (2010) 'Ingeniería del Software Un enfoque práctico, Septima edición ed'. McGraw-Hill.

Ramírez Vega, A. (2011) 'El uso de la plataforma Open Conferences Systems en la organización de eventos académicos: la experiencia de la XIII CIAEM', *Cuadernos de Investigación y Formación en Educación Matemática*, (11), pp. 383–401.

Ramos Martínez, M. D. (2013) 'Metodología de desarrollo aplicaciones móviles'.

Rivera, Y. J. M. (2012) *Sistema operativo Android: características y funcionalidad para dispositivos móviles*. Universidad Tecnológica de Pereira. Facultad de Ingenierías Eléctrica, Electrónica, Física y Ciencias de la Computación. Ingeniería de Sistemas y Computación.

Rivero, M. O. M., Hidalgo, R. C. R. and Cañizares, D. J. R. (2016) 'Implementación de un sistema de repositorios digitales en la Universidad Central Marta Abreu de Las Villas', in *Congreso Universidad*.

Robledo Fernández, D. (2014) *Desarrollo de aplicaciones para Android II*. Ministerio de Educación.

Van Rossum, G. and Drake Jr, F. L. (2009) ‘Tutorial Python’, *Fred L. Drake, Jr*.

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

Standard, I. (1998) ‘Iso 9241-11’, 1998.

Téllez Alarcia, D. (2010) ‘Tiempos Modernos. Pasado, presente y futuro de una revista electrónica de Historia Moderna’, *Nuevo Mundo Mundos Nuevos. Nouveaux mondes mondes nouveaux-Novo Mundo Mundos Novos-New world New worlds*. EHESS.

Vanegas, C. A. (2012) ‘Desarrollo de aplicaciones sobre Android.’, *Revista Vínculos Vol. 9 número 2, julio de 2012, pp 129-145*, pp. 129–145.

Vique, R. R. (2012a) ‘Métodos para el desarrollo de aplicaciones móviles’, *PID_00176755*.

Vique, R. R. (2012b) ‘Métodos para el desarrollo de aplicaciones móviles’.

Yagüe, A. and Garbajosa, J. (2009) ‘Comparativa práctica de las pruebas en entornos tradicionales y ágiles’, *REICIS. Revista Española de Innovación, Calidad e Ingenier{’\i}a del Software*. Asociación de Técnicos de Informática, 5(4).