Título: Herramienta de sincronización entre los sistemas operativos GNU/Linux Nova y Android

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

Autor: Ronny Montano Martínez

Tutores: Ing. Juan Manuel Fuentes Rodríguez

Ing. Héctor Pérez Baranda

La Habana, junio 2016

DECLARACIÓN DE AUTORÍA

Declaro ser autor del presente trabajo de diploma y concedo a la Universidad de las Ciencias Informáticas los derechos patrimoniales del mismo, con carácter exclusivo. Para que así conste firmamos la presente a los ___ días del mes de _____ del año _____.

_____

Ronny Montano Martínez

Autor

_____ _____

Ing. Héctor Pérez Baranda

Tutor

Juan M. Fuentes Rodríguez

Tutor

Agradecimientos

A mi madre por saber comprenderme en los momentos más difíciles y darme todo el amor del mundo, GRACIAS

A mi padre que siempre me apoyó y confió en mí, GRACIAS

A mi hermano Onay, por estar siempre ahí para mí y ser mi compañero y amigo GRACIAS

A mi hermanita Karo que ojalá un día decida seguir mis pasos, a Ricci que ha sido como una madre para mí, al igual que su familia ha sido como la mía, GRACIAS

A Rodo, que ha sido como un padre para mí, GRACIAS

A mis abuelos Enrique y José, que aunque no estén sé que están orgullosos de mí, GRACIAS

A mis abuelas que espero se sientan orgullosas de mí, GRACIAS

A mis tías, tíos, primas y primos que me ayudaron siempre que necesité, GRACIAS

A Jessica no solo por ser mi novia sino gran parte de mi vida, por cada día robarme un pedacito de mi corazón y hacerlo todo suyo, y por tantas cosas que me serían imposibles expresarlo en estas cortas líneas, GRACIAS

A la familia de Jessica, a sus padres, abuela, tías y primos por dejarme formar parte de su mundo y darme siempre mucho cariño, GRACIAS

A mis tutores Juan y Héctor que han sido como unos compañeros de tesis, sin ellos no hubiera sido posible realizar esta investigación, GRACIAS

A mis amigos del barrio: Yosme, Yenni, Ale, Victor, Diego, Andy, Range, Yendry, por hacer mis cortas estancias allá divertidas y entretenidas, GRACIAS

A mi gente del 2505, por aguantar día a día las "locuras" que yo hablaba en el aula, GRACIAS

A mi gente del 2503, no es mi grupo de plantilla pero si de corazón, GRACIAS

A mi gente de la UCI y del fútbol que han formado parte de mi vida en esta etapa: Braiman, Manuel, Stephany, Rere, Say, Arain, Miguelón, Osciel, Josué, GRACIAS

A mi tía adjunta Mayte, GRACIAS

A todas las personas que de una forma u otra han aportado a mi formación como profesional y como persona,

A todos… muchas gracias

Ronny

Dedicatoria

*Este trabajo lo dedico a **mi madre**, que hizo hasta lo imposible para que llegara a este punto, por quererme y apoyarme tanto, a **mi familia** en conjunto, tanto la de Jessica como la mía y a todos **mis amigos**, que no son ni muchos ni pocos pero son los SUFICIENTES.*

Ronny

RESUMEN

La sincronización de datos entre dispositivos móviles inteligentes y computadoras es una acción muy realizada en la actualidad. Dicha tarea alcanza mayor auge debido al gran porciento de usuarios que contienen terminales de este tipo. Uno de los problemas actuales es que los usuarios de la plataforma GNU/Linux no pueden realizar la sincronización de datos de sus dispositivos móviles con sistema operativo Android y el uso de una herramienta de este tipo se evidencia en el proceso de migración a software con estándares abiertos propuesto por el país, tomando como plataforma para la migración la distribución cubana GNU/Linux Nova, creada en la Universidad de las Ciencias Informáticas. En esta investigación se tiene como objetivo el desarrollo de una solución informática para la sincronización de datos compatible con la distribución GNU/Linux Nova. Para el logro del mismo se realiza un estudio de los antecedentes de los sistemas informáticos de sincronización de datos. Se define que la solución informática a desarrollar basará su funcionamiento en una arquitectura cliente-servidor, especificándose Java y Python como lenguajes de desarrollo; Android Studio y Geany como entornos de desarrollo para cada lenguaje respectivamente; que ayudarán al desarrollo de la solución informática. Luego de creadas las bases, se realiza la implementación de las funcionalidades obteniendo como resultados: la sincronización de los contactos, eventos del calendario y de los archivos multimedia.

Palabras claves: dispositivo móvil inteligente, estándares abiertos, informática, plataforma, sincronización de datos.

ÍNDICE

Índice de ilustraciones

Índice de tablas

INTRODUCCIÓN

Los dispositivos móviles inteligentes han cambiado la forma en que la sociedad percibe y disfruta de la tecnología, pues brindan la posibilidad de tener un equipo de cómputo que en ocasiones posee prestaciones similares a los de escritorio, teniendo la ventaja de ser más portátiles debido al tamaño y número de funcionalidades. El desarrollo en los terminales móviles ha estado marcado en gran parte por el amplio progreso de los sistemas operativos; que van desde Java ME, Windows Phone, BlackBerry hasta iOS y Android, siendo estos últimos los más usados en la actualidad. Este último se ha destacado grandemente dentro de esta gran evolución, siendo una plataforma de código abierto destinada a terminales móviles basada en una variante del núcleo Linux y patrocinada por un consorcio de empresas llamada Open Handset Alliance[1] (OHA) (1). Android es un sistema operativo que ha expandido su uso a alrededor de un mil millones de dispositivos que van desde relojes inteligentes, teléfonos, tabletas, equipos de cómputos, televisores, autos y equipos de robótica (1). Es el sistema operativo más utilizado en la tecnología móvil (2), y ocupa más de la mitad del mercado de dispositivos móviles en el mundo (*ver Ilustración 1*). Parte de su éxito se debe en gran medida a que posee la mayor tienda en línea de aplicaciones oficiales, que brindan un mayor número de funcionalidades al sistema (3).

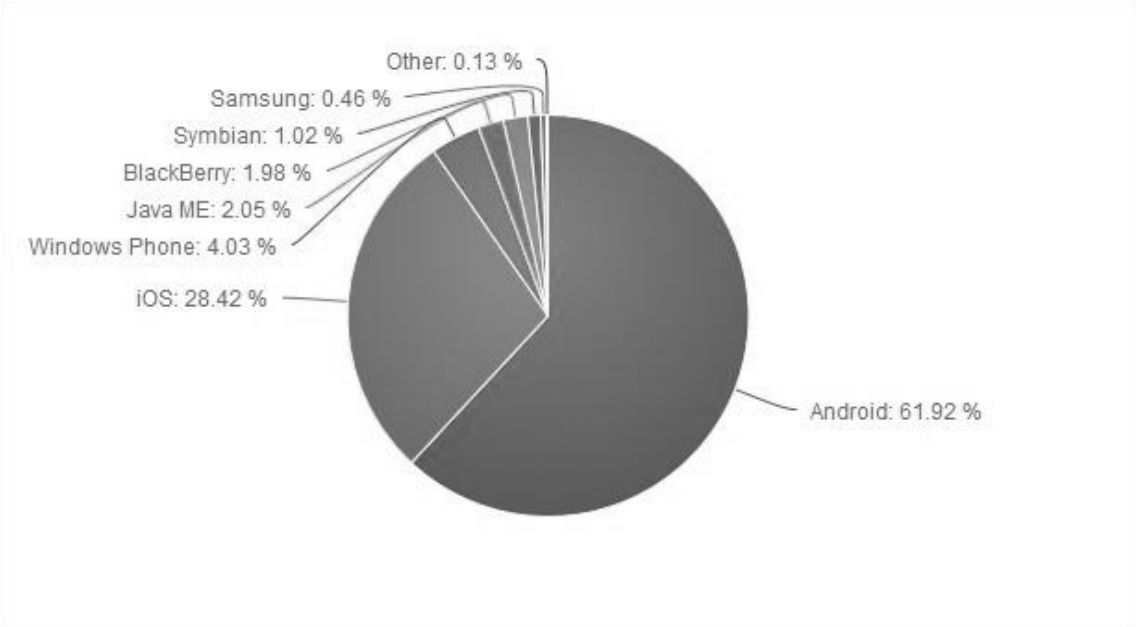


Ilustración 1: Uso de los sistemas operativos en dispositivos móviles en el mundo

Android ha tenido un marcado auge en Cuba, pues domina grandemente en el área de los dispositivos móviles. Este suceso se debe considerablemente al bajo precio de los

[1] Open Handset Alliance es una alianza comercial de 84 compañías que se dedica a desarrollar estándares abiertos para dispositivos móviles.

dispositivos de gama media y baja. Según el Vicerrector de Tecnología de la Universidad de las Ciencias Informáticas (UCI) Denys Buedo Hidalgo todos los teléfonos inteligentes entregados por la universidad a sus trabajadores y directivos, tienen sistema operativo Android. Un estudio realizado en dicha universidad, tomando como muestra el tercer año de la Facultad 2 de dicho centro de altos estudios, revela que el 88,7 % de los estudiantes poseen teléfonos inteligentes con Android *(ver anexo 1),* obteniendo de esta manera la gran repercusión de dicho sistema operativo en la comunidad universitaria.

Conociendo el nivel de popularidad de Android, la Universidad de las Ciencias Informáticas está desarrollando un sistema operativo para teléfonos móviles, este es NovaDroid una personalización de Android para su uso en el país. Este tiene como objetivo realizar una asimilación tecnológica y adaptación a las condiciones del país debido las características de su parque tecnológico y a la incapacidad de acceder a varios servicios por la política de los Estados Unidos de América. Simultáneamente la universidad y nuestro país están comprometidos con la migración a sistemas operativos con estándares abiertos en su infraestructura informática para elevar los niveles de seguridad, soberanía tecnológica, socio-adaptabilidad y sostenibilidad; proceso en el que la universidad ya ha cumplido con el 75% de sus equipos de cómputo (4) extendiendo el uso de GNU/Linux Nova, un sistema operativo desarrollado en la UCI. Dicho *software* está basado en GNU/Linux, y tiene como objetivo apoyar la migración del país a tecnologías de software libre y código abierto. Durante el proceso de migración se evidenció la necesidad de intercomunicar los dispositivos móviles y las computadoras con los sistemas operativos Android y GNU/Linux Nova respectivamente, debido al gran auge que posee el uso de dispositivos móviles.

A pesar de la existencia de muchas herramientas que logran la sincronización de datos, no pueden ser usadas para Android y GNU/Linux Nova, siendo este el sistema operativo usado en el proceso de migración, pues son creadas exclusivamente para Windows y para el uso específico en dispositivos creados por grandes empresas.

Existe un conjunto de aplicaciones que al no ser de código abierto atentan contra la soberanía y la seguridad tecnológica propuestas por el país, siendo estas aplicaciones no compatibles con el proceso de migración. Ejemplo de ellas son las aplicaciones introducidas por las empresas Wondershare, Samsung, HTC y LG desarrolladas para lograr una mayor aceptación en el mercado de sus productos. Existen otras herramientas para la sincronización de datos entre un teléfono inteligente y una computadora pero su funcionamiento se basa en una conexión inalámbrica, siendo este un periférico no disponible en la mayoría de las computadoras de las entidades del país.

Teniendo en cuenta la situación problemática planteada se tiene como **problema científico de la investigación:**

¿Cómo lograr la correcta sincronización de datos del usuario entre los dispositivos móviles con sistema operativo Android y la distribución GNU/Linux Nova?
Luego de planteado el problema científico de la investigación, se define como **objeto de estudio**, el proceso de sincronización entre los sistemas operativos de teléfonos inteligentes y los de computadoras, delimitando como **campo de acción** los procesos de sincronización entre el sistema operativo Android y GNU/Linux Nova.
Para dar solución al problema se plantea como **objetivo general:** desarrollar una solución informática que permita la sincronización de datos entre los teléfonos inteligentes con sistema operativo Android y las computadoras con la distribución de GNU/Linux Nova, determinando como **objetivos específicos:**

- Analizar el proceso de sincronización de datos entre los sistemas operativos de dispositivos inteligentes y de computadoras mediante el estudio del estado del arte de aplicaciones que realizan dicha tarea.
- Analizar y diseñar una aplicación para la sincronización entre los sistemas operativos Android y GNU/Linux Nova.
- Implementar la propuesta de solución para la sincronización entre dichos sistemas operativos.
- Validar la solución propuesta mediante las pruebas de *software*.

Se plantea como **idea a defender:** el desarrollo de una solución informática que realice la sincronización de datos entre los sistemas operativos Android y GNU/Linux Nova permitirá una mayor interoperabilidad entre los mismos.

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

- Elaboración del marco teórico de la investigación para garantizar el basamento y fundamento científico de la misma.
- Análisis y comparación de los sistemas actuales que realizan la sincronización entre los dispositivos móviles y las computadoras convencionales.
- Fundamentación de la selección de la metodología, las herramientas y las tecnologías a utilizar en el desarrollo de la solución propuesta.
- Especificación de los requisitos funcionales y no funcionales de la solución propuesta para establecer lo que el *software* debe hacer y bajo qué circunstancias debe hacerlo.
- Implementación de la solución propuesta para dar cumplimiento a los requisitos definidos

- Realización de las pruebas de software para validar el correcto funcionamiento de la propuesta de solución.

En la realización de la investigación fue necesario el uso de varios métodos para la obtención y extensión del conocimiento, entre los métodos teóricos están:

- **Análisis Histórico – Lógico:** para estudiar la evolución y desarrollo de herramientas para la sincronización de datos entre teléfonos inteligentes y computadoras personales, además de realizar un estudio del estado del arte de las distintas aplicaciones con esa funcionalidad.
- **Análisis y síntesis:** para concretar los elementos más importantes relacionados con el objeto de estudio. Se realizó el análisis de un gran volumen de documentación, que permitió sintetizar el contenido que da soporte a la propuesta del trabajo a desarrollar.

Entre los **métodos empíricos** utilizados en la investigación está el **método de observación** para conocer el funcionamiento de las herramientas de sincronización de datos entre dispositivos móviles y computadoras personales. Se utilizó el **método de la encuesta** para obtener el porcentaje de uso de los sistemas operativos, tomando como muestra el 3er año de la facultad 2.

La investigación está estructurada de forma organizada, como se muestra a continuación:

Capítulo 1: Fundamentación teórica se realiza un estudio de estado del arte referente a las herramientas de sincronización de datos existentes. Se describe y caracteriza la metodología, las herramientas y tecnologías empleadas en el desarrollo del sistema, justificando además la selección y uso de estas.

Capítulo 2: Análisis y diseño de la solución propuesta, se abordará con mayor detalle los aspectos relacionados con los elementos de análisis y diseño de la propuesta de solución, tomando como base los resultados del estudio realizado en el estado del arte. Se realiza el diseño de la solución propuesta guiado por la metodología de desarrollo a utilizar, identificación de funcionalidades, descripción de historias de usuario y consideraciones del diseño del sistema. En el **Capítulo 3: Implementación y pruebas** se explican las tareas de implementación, los resultados obtenidos, las funcionalidades alcanzadas en el período de implementación, se realizan y documentan las pruebas para lograr la validación del producto.

Capítulo 1: Fundamentación teórica

1.1 Introducción

El objetivo de este capítulo es plantear algunos aspectos teóricos que servirán de soporte para la elaboración del sistema. Se exponen conceptos para lograr una mejor comprensión sobre los términos tratados en el proceso de sincronización de datos entre los dispositivos móviles inteligentes y las computadoras convencionales, así como un estudio de estado del arte sobre aplicaciones existentes para la sincronización de datos. Por último, se presentan tanto la metodología como las herramientas y tecnologías a utilizar en la implementación del sistema.

1.2 Conceptos fundamentales

Sincronización de datos: proceso en el que se relacionan dos partes, para lograr una comunicación entre ellos con el fin de compartir archivos (5).

Sistema operativo: capa de software encargada de proporcionar a los programas de usuario un modelo de sistema mejor, más simple y pulcro, así como encargarse de la administración de todos los recursos de la computadora (6).

Teléfono inteligente: es un teléfono celular con pantalla táctil, que permite al usuario conectarse a internet, gestionar cuentas de correo electrónico e instalar otras aplicaciones y recursos a modo de pequeño computador (7).

1.3 Estudio de homólogos

Actualmente existen aplicaciones que realizan la sincronización de los datos de usuario, de las cuales algunas de ellas se analizan a continuación debido a que fueron desarrolladas por las grandes empresas para lograr una mayor aceptación de sus productos, otras que han sido debates en foros y blogs debido a su gran propagación por parte de los usuarios de los dispositivos con Android.

1.3.1 AirDroid

Es una herramienta que permite a los teléfonos inteligentes compartir con la computadora los archivos multimedia, contactos personales, leer los mensajes de textos recibidos y documentos que no son clasificados como multimedia. Contiene una interfaz amigable para el usuario y no se necesitan conocimientos avanzados para su uso. Existen muchas versiones de esta aplicación pero todas tienen en común que se utilizan en sistemas operativos Windows, y la principal desventaja es que el modo de conexión

es por Wifi (8), periférico no disponible en las computadoras de escritorio, siendo estas las más comunes en las instituciones del país.

1.3.2 Samsung Kies

Es un programa diseñado para uso exclusivo de los teléfonos inteligentes creados por la empresa Samsung, para lograr la sincronización de datos. Es capaz de transferir música, fotos, vídeos. Permite realizar actualizaciones del *firmware* del dispositivo. Cuenta con dos aplicaciones, una para la computadora y la otra instalada en el dispositivo. Permite conectar el dispositivo móvil con Android al computador utilizando un cable USB, o a través de Bluetooth o Wifi, lo que no sería una desventaja en este caso. La principal desventaja de esta aplicación es exclusividad para teléfonos marca Samsung (9).

1.3.3 MobileGo

Es una aplicación que puede ser utilizada para realizar transferencia de aplicaciones, contactos, mensajes de textos, archivos multimedia y otros tipos de archivos. Permite realizar la recuperación de datos eliminados y copias de seguridad del contenido del dispositivo. Contiene la opción de eliminar archivos duplicados y además brinda toda la información referente al dispositivo. Permite el envío de mensajes de textos, conversión de formatos de audio y video, y la importación de listas de reproducción de audio. También otorga permisos de administrador del dispositivo y cuenta con una opción de controlar el terminal en tiempo real. La principal desventaja de esta aplicación es que no es compatible con ninguna distribución de GNU/Linux, siendo solo de uso para el sistema operativo Windows (10).

1.3.4 QtADB

Es una herramienta de código abierto compatible con distribuciones GNU/Linux y con Windows. Gestiona los archivos y directorios del dispositivo móvil. Permite la instalación y eliminación de aplicaciones, además brinda la posibilidad de realizar copias de respaldo a dichas aplicaciones. Gestiona mensajes de textos, permitiendo el envío y recibo de dichos textos. Contiene la opción de escribir comandos mediante el Shell de Android, la captura de pantalla es otra de las ventajas de esta aplicación. Además brinda al usuario la posibilidad de detección automática del dispositivo. La principal desventaja de este software es que no cuenta con la sincronización de la mayoría de los datos que contiene el terminal (11).

Conclusiones del estado del arte

Luego del análisis realizado sobre las aplicaciones que realizan la sincronización de datos de usuario entre los teléfonos inteligentes y las computadoras se arriba a las siguientes conclusiones:

- La herramienta AirDroid cuenta con un modo de conexión no disponible en las computadoras convencionales, su diseño de interfaz es muy usable, sus funcionalidades se tendrán en cuenta.
- La aplicación MobileGo a pesar de contar con muchas funcionalidades, no es compatible con ninguna de las distribuciones GNU/Linux.
- La herramienta QtADB a pesar de ser multiplataforma, solo se tuvo en cuenta funcionalidades de este, debido a que el autor evaluó que la inclusión de la biblioteca Qt requeriría más espacio de almacenamiento. Además se estimó que el tiempo para realizar ingeniería inversa sería mayor que si se realizaba una nueva propuesta de solución al problema.
- Samsung Kies es una aplicación que brinda funcionalidades de sincronización de datos, pero es solo para uso en dispositivos de marca Samsung.

A continuación se muestra una tabla comparativa de las herramientas estudiadas que realizan la sincronización, los rasgos de comparación se definieron de acuerdo a características que presentan dichas aplicaciones y de acuerdo a las necesidades de la investigación (*ver Tabla 1*).

Modos de conexión.	**Aplicaciones para sincronización de datos entre dispositivos móviles con Android y la computadora.**			
	AirDroid	Samsung Kies	QtADB	MobileGo
Bluetooth		X		
Wifi	X	X		
USB		X	X	X
Elementos que sincronizan.				
Imágenes	X	X	X	X
Videos	X	X	X	X
Música	X	X	X	X
Contactos	X			X

Mensajes de textos	X			X
Documentos	X	X		X
Aplicaciones	X	X		X
Sistema Operativo en el que se ejecuta				
Mac	X	X	X	X
Linux	X		X	
Windows	X	X	X	X

Tabla 1: Comparación de las aplicaciones analizadas

Al efectuar la comparación entre las herramientas de sincronización de datos entre los dispositivos inteligentes y las computadoras se decidió desarrollar una nueva solución para lograr la sincronización entre Android y GNU/Linux Nova

A continuación se muestran los elementos referentes a las herramientas y la metodología a utilizar en el desarrollo de la propuesta de solución.

1.4 Herramientas y lenguajes de desarrollo

Para la realización de la propuesta de solución se realizó un estudio, para la selección de aquellas respondieran a las necesidades del sistema a implementar. A continuación se mostrará el resultado del estudio realizado.

1.4.1 Lenguajes de desarrollo

Java

Es un lenguaje multiplataforma debido a que posee una máquina virtual de java o JMV por sus siglas en inglés y cuenta con la característica de ser uno de los lenguajes de programación más utilizados en el mundo (12). Este contiene una sintaxis sencilla, orientada a objetos que permite optimizar el tiempo y ciclo de desarrollo (compilación y ejecución). Además es independiente de las arquitecturas debido a que el compilador no produce un código específico para un tipo de arquitectura (13). Se utiliza para la implementación de las aplicaciones en el sistema operativo Android.

Python 2.7

Es un lenguaje de programación de alto nivel creado por Guido van Rossum en el año 1991. Es interpretado o de script, lo que permite realizar cambios en el código mientras se está ejecutando la aplicación, además posee la ventaja de ser multiplataforma. Una ventaja fundamental es que su intérprete es de código abierto. Python contiene una gran cantidad de bibliotecas, tipos de datos y funciones incorporadas en el propio lenguaje, que ayudan a realizar muchas tareas comunes sin necesidad de tener que programarlas desde cero. Posee un gran soporte e integración con otros lenguajes y herramientas (14). Python se utiliza para el desarrollo de la aplicación cliente, haciendo uso de bibliotecas contenidas en él.

1.4.2 Entorno de desarrollo

Geany

Es un pequeño y ligero entorno de desarrollo integrado. Fue desarrollado para proporcionar un pequeño y rápido entorno de desarrollo integrado (IDE), que tiene sólo unas pocas dependencias de otros paquetes. Presenta algunas de las siguientes características (15):

- El resaltado de sintaxis.
- Navegación en el código.
- Auto-cierre de etiquetas XML y HTML.

Es un entorno de desarrollo integrado multiplataforma, que se apoya en las librerías GTK. Una de las principales ventajas es que código está disponible bajo los términos de la Licencia Pública General de GNU (16).

Android Studio 1.3.2

Es el entorno de desarrollo integrado oficial para el desarrollo de aplicaciones Android, basado en IntelliJ IDEA, desarrollado por Google y fue anunciado en mayo de 2013. Es un IDE distribuido de forma libre bajo la licencia de Apache License 2.0. Tiene integrado una herramienta llamada Gradle para la construcción y compilación de la aplicación. Contiene plantillas para ayudar a construir aplicaciones con características comunes. Presenta la ventaja de que permite crear dispositivos virtuales para emular las aplicaciones, debido a que cuenta con un paquete de desarrollo de software, SDK por sus siglas en inglés (17).

Android SDK

El paquete de desarrollo de software incluye un conjunto de herramientas para ayudar al programador, tales como bibliotecas, un manejador de emuladores, documentación,

códigos ejemplos y tutoriales. Es un paquete multiplataforma. Permite la integración con Eclipse usando el Android Development Tools (ADT) y también con Android Studio (17). Se usa en el desarrollo de la solución debido a que es imprescindible para el entorno de desarrollo integrado seleccionado.

ADB

Android Debug Bridge (ADB) es una herramienta de línea de comandos versátil que permite comunicarse con una instancia de emulador o dispositivo con Android conectado. Es un programa cliente-servidor que incluye tres componentes (18):

- Un cliente, que se ejecuta en la máquina de desarrollo. Puede invocar un cliente desde un shell mediante la emisión de un comando adb.
- Un servidor, que se ejecuta como un proceso en segundo plano en el equipo de desarrollo.
- Un *demon*, que se ejecuta como un proceso en segundo plano en cada emulador o dispositivo instancia.

JSON

Es un formato ligero de intercambio de datos. Su escritura y lectura es simple para los humanos, mientras que para las máquinas es simple interpretarlo y generarlo. Es completamente independiente del lenguaje pero utiliza convenciones que son ampliamente conocidos por los programadores de la familia de lenguajes C, incluyendo C, C++, C#, Java, JavaScript, Perl, Python, y muchos otros. JSON está constituido por dos estructuras (19):

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

En la aplicación desarrollada fue necesario su uso para la transferencia de la información de los elementos entre la aplicación cliente y el servidor, debido a las ventajas que tiene, algunas de las cuales vemos a continuación:

- Escribir pares del tipo "nombre: valor" y asignarlos a un objeto.
- Fácil de entender, porque si está bien escrito forma una estructura que se auto explica.

1.4.3 Bibliotecas para el desarrollo

GTK

Es un kit de herramientas multiplataforma para crear interfaces gráficas de usuario. Ofreciendo un conjunto completo de los *widgets*[2], GTK es adecuado para proyectos que van desde pequeñas herramientas únicas para completar suites de aplicaciones. Está escrito en C, pero ha sido diseñado desde cero para apoyar una amplia gama de lenguajes, como Perl y Python proporciona un método eficaz para el desarrollo rápido de aplicaciones. Su característica de ser un software libre y parte del proyecto GNU es una de las principales, algunas de las otras se muestran a continuación (20):

- Contiene una amplia colección de widgets básicos e interfaces para el uso en su aplicación.
- Disponible en muchos otros lenguajes de programación gracias a los enlaces de lenguaje disponibles. Esto hace que sea GTK un conjunto de herramientas muy atractivo para el desarrollo de aplicaciones.
- Puede ser usado en Windows, y en Linux.

Gson

Es una biblioteca de código abierto para el lenguaje de programación Java que permite la serialización y deserialización entre objetos Java y su representación en notación JSON. Presenta características que definen su uso, algunas de ellas se muestran a continuación (21):

- Proporciona un fácil mecanismo de utilizar para convertir objetos de Java a JSON y viceversa.
- Genera un fichero JSON contacto y legible.

1.4.4 Herramienta de modelado

Visual Paradigm:

Es una herramienta de Ingeniería de Software Asistida por Computadora CASE que permite la representación de todo tipo de diagramas (Procesos de Negocio, Decisión, Actor de negocio, Documento). Fue diseñado para una amplia gama de usuarios interesados en la construcción de sistemas de software de forma fiable a través de la utilización de un enfoque Orientado a Objetos. Se encuentra disponible para varias

[2] Widgets es una pequeña aplicación o programa, usualmente presentado en archivos o ficheros pequeños.

plataformas y cuenta con múltiples versiones, con diferentes especificaciones. Entre los elementos que ofrece Visual Paradigm se encuentran (22):

- Navegación intuitiva entre la escritura del código y su visualización.
- Potente generador de informes en formato PDF/HTML.
- Ambiente visualmente superior de modelado.
- Sofisticado diagramador de *Layout*[3].

1.4.5 Lenguaje de modelado UML

Lenguaje de modelado unificado (UML) es un lenguaje de modelado visual que se emplea para graficar, generar y documentar artefactos en un sistema de software. Contiene un conjunto de notaciones específicas para la programación orientada a objetos las cuales se combinan con simbologías para lograr los distintos diagramas que representan las etapas del desarrollo de software en un proyecto.

UML se conforma de varios tipos de diagramas de forma estándar que le permite al analista crear un anteproyecto de varias facetas que sean comprensibles para los clientes, desarrolladores y todo los que se encuentran inmersos en el proceso de desarrollo. Todos esos diagramas son necesarios puesto que cada uno se dirige a una persona con un rol específico en el sistema. Un modelo UML indica qué es lo que debe hacerse pero no explica el cómo se hará (23).

Ventajas de UML

- Es un lenguaje de modelado de propósito general que pueden usar todos los modeladores, no es un método de desarrollo. No explica cómo pasar del análisis al diseño, ni del diseño a la implementación.
- Es independiente del ciclo de desarrollo que se vaya a seguir, soporta metodologías de desarrollo de software, pero no especifica que metodología usar.
- No tiene propietario y está basado en el común acuerdo de gran parte de la comunidad informática.
- Pretende abordar los problemas actuales del desarrollo del software y en cualquier plataforma tecnológica donde se utilizará.

1.5 Metodología de desarrollo de desarrollo de software

Es un conjunto de métodos, estas se elaboran a partir del marco definido por uno o varios modelos del ciclo de vida. Una metodología de desarrollo consiste en un conjunto

[3] Suele utilizarse para nombrar al esquema de distribución de los elementos dentro un diseño.

de procedimientos, técnicas, herramientas y soporte documental, que deben seguirse para el desarrollo de un software. Estas especifican (24):

- Cómo se debe dividir un proyecto en etapas.
- Qué tareas hay que realizar en cada etapa.
- Qué salidas se producen y cuándo.
- Qué herramientas se utilizan.
- Cómo se gestiona y controla un proyecto.

1.5.1 Metodologías de desarrollo de software ágiles

Los métodos ágiles son formas de desarrollo iterativo que se centran en la especificación, diseño e implementación del sistema de forma incremental, implican directamente a los usuarios en el proceso de desarrollo del software. Lograr reducir la carga en cuanto al esfuerzo de desarrollo puede hacer posible un desarrollo de software más rápido (24).

1.5.2 Metodología de desarrollo AUP-UCI

Proceso Unificado Ágil (AUP) en su variante UCI, es un acercamiento aerodinámico a desarrollo del software basado en el Proceso Unificado Rational de IBM (RUP), basado en disciplinas y entregables incrementales con el tiempo. El ciclo de vida en proyectos grandes es serial mientras que en los pequeños es iterativo. Promover la colaboración para alinear los intereses comunes y difundir el conocimiento (25). La metodología AUP-UCI cuenta con tres fases que son Inicio, Ejecución y Cierre, dividido en varias disciplinas que se muestran a continuación:

- **Modelado de negocio:** Disciplina destinada a comprender los procesos de negocio de la organización.
- **Requisitos:** Disciplina que comprende la administración y gestión de los requisitos funcionales y no funcionales del producto.
- **Análisis y diseño:** En esta disciplina se modela el sistema y su forma para que pueda soportar todos los requisitos, incluyendo los requisitos no funcionales.
- **Implementación:** En la implementación, a partir de los resultados del Análisis y Diseño se construye el sistema.
- **Pruebas interna:** Se verifica el resultado de la implementación probando cada construcción, incluyendo tanto las construcciones internas como intermedias, así como las versiones finales a ser liberadas.
- **Pruebas de liberación:** Pruebas diseñadas y ejecutadas por una entidad certificadora de la calidad externa.

- **Pruebas de aceptación:** La prueba final antes del despliegue del sistema. Su objetivo es verificar que el software está listo y que puede ser usado por usuarios.

La solución propuesta será desarrollada basándose en el escenario 4 que propone la metodología AUP-UCI que plantea (26):

Escenario No 4: "*Aplica a los proyectos que hayan evaluado el negocio a informatizar y como resultado obtengan un negocio muy bien definido. El cliente estará siempre acompañando al equipo de desarrollo para convenir los detalles de los requisitos y así poder implementarlos, probarlos y validarlos. Se recomienda en proyectos no muy extensos, ya que una HU no debe poseer demasiada información*".

La elección de la dicha metodología para regir el proceso de desarrollo de software se debe en gran parte a lograr una mayor homogeneidad en los procesos de desarrollo de software de la Universidad de las Ciencias Informáticas.

1.6 Conclusiones del capítulo

Luego de la sistematización realizada a las herramientas existentes y concluir que estas no resuelven el problema planteado se decidió la implementación de una nueva propuesta de solución. Debido a que las herramientas estudiadas a pesar de contar con características similares al sistema deseado no resuelven la situación problemática, sin embargo pueden ser utilizadas como base a la implementación a realizar. Basándose en que la solución propuesta debe ser funcional para Android y GNU/Linux Nova hay que seguir ciertas normas de software libre por eso se seleccionaron un conjunto de herramientas de desarrollo que no se requiere licencias para su uso siguiendo así las normas del proceso de migración. Siendo el equipo de desarrollo y el tiempo de duración del proyecto pequeños se decidió el uso de una metodología de desarrollo ágil, que ayudará a su vez a un ajuste con al proceso de desarrollo de la universidad. A partir de este momento están creadas las bases teóricas para proceder al análisis y diseño de la solución propuesta, elementos que serán tratados en el próximo capítulo.

Capítulo 2: Análisis y diseño de la solución propuesta

2.1 Introducción

Este capítulo tiene como meta plantear la vía a seguir para la realización de la solución propuesta, siguiendo la metodología AUP-UCI para llegar a tener un producto con buena calidad teniendo en cuenta las prácticas y los principios que contribuyan a agilizar el proceso.

2.2 Modelo de dominio

Es una representación visual de clases conceptuales o de objetos reales en un dominio de interés. No constituyen clases de software, son representaciones de los conceptos involucrados para la comprensión de la solución y las asociaciones establecidas describen las relaciones entre los conceptos (27). La Ilustración 2 muestra una representación estática del entorno real del proyecto mediante el modelo de dominio.

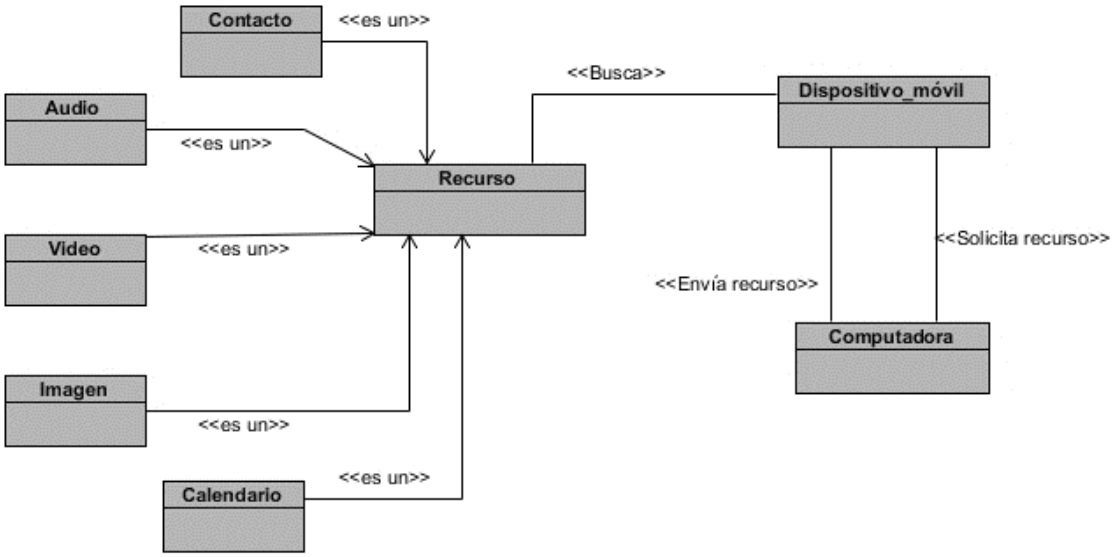


Ilustración 2: Diagrama de clases del modelo de dominio

A continuación se presenta una descripción de las entidades involucradas en el modelo de dominio:

Audio: es una entidad que hace referencia a los archivos de audio almacenados en el dispositivo móvil.

Contacto: es una entidad que hace referencia a los contactos almacenados en la aplicación Contactos del dispositivo móvil con sistema operativo Android, encargado de almacenar la información de las personas conocidas

Video: es una entidad que hace referencia a los archivos de video almacenados en el dispositivo móvil.

Imagen: es una entidad que hace referencia a los archivos de imagen almacenados en el dispositivo móvil.

Calendario: es una entidad que hace referencia a los eventos almacenados en la aplicación Calendario del dispositivo móvil con sistema operativo Android, encargado de almacenar la información referente a los eventos registrados en dicha aplicación en ella.

Recurso: es una entidad que su tipo puede ser una contacto, calendario, video, imagen o música.

Dispositivo móvil: es la entidad que hace referencia al dispositivo móvil que hace función de servidor en la implementación de la solución propuesta.

Computadora: es la entidad que hace referencia a la computadora, que hace función de cliente en la implementación de la solución propuesta.

2.3 Modelado del sistema

A continuación se describe en detalles la construcción de la aplicación de sincronización de datos de usuarios. Se realiza la propuesta del sistema y se analizan los requerimientos funcionales y no funcionales.

2.3.1 Solución propuesta

Como se ha abordado hasta este punto de la investigación, para darle solución al problema tratado se propone el desarrollo de la aplicación *DroydConnect* que permita la sincronización de los datos de usuarios en los teléfonos móviles inteligentes con sistema operativo Android y las computadoras convencionales con la distribución GNU/Linux Nova.

La solución propuesta permitirá la sincronización de los contactos, eventos del calendario y archivos multimedia. Las aplicaciones cliente y servidor de la solución informática están escritas en Python y Java respectivamente. El modo de conexión entre los dispositivos es mediante una conexión USB.

La solución propuesta se desarrolla en una arquitectura cliente-servidor. El dispositivo móvil será el encargado de crear el servidor para responder a las peticiones del cliente. El usuario al conectar el dispositivo móvil a la computadora, y ejecutar la aplicación cliente, esta redirecciona el puerto hacia una conexión TCP, mediante el uso de ADB.

La aplicación cliente realizará peticiones de los datos que desea obtener; que pueden ser contactos, eventos del calendario o archivos multimedia seleccionándolos mediante la interfaz de usuario. La solicitud del cliente será enviada utilizando las funcionalidades de la biblioteca urllib2. El servidor realizará una búsqueda del contenido solicitado por el cliente para su posterior envío, haciendo uso de los proveedores de contenidos de Android. Se creará una lista de los elementos solicitados, que luego serán convertidos a un formato de texto JSON, mediante el uso de la biblioteca Gson. Luego de realizada la búsqueda y la conversión de los datos solicitados, el servidor envía la respuesta de la solicitud realizada. La aplicación cliente listará el contenido recibido para luego mostrárselo al usuario, que seleccionará cuales son los elementos que desea sincronizar.

Los contactos o eventos del calendario seleccionados para la sincronización serán integrados a la aplicación de Contactos y Calendario respectivamente previamente instalada por defecto en la distribución GNU/Linux Nova. La sincronización de los archivos multimedia se realiza mediante el uso de comandos brindados por ADB. Luego de seleccionado el elemento multimedia que se desea sincronizar, la aplicación cliente ejecuta un comando **adb pull [dir source] [dir destino]** donde **[dir source]** es la localización del archivo en el dispositivo móvil obtenida mediante el archivo JSON antes enviado y [**dir destino]** la dirección destino en la computadora, que varía en dependencia del archivo multimedia a sincronizar (*ver Figura 3*).

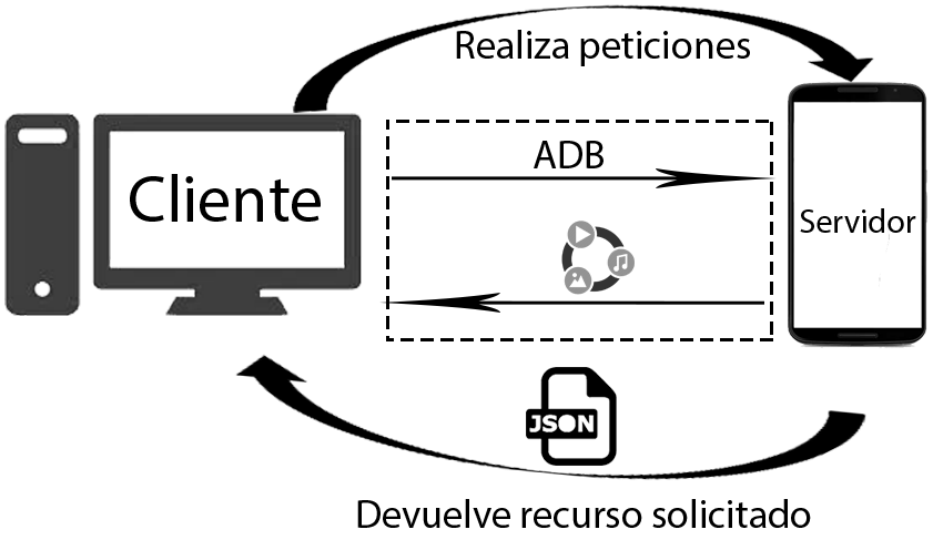


Ilustración 3: Diagrama de la propuesta de solución

2.4 Listado de requerimientos

Los requerimientos para un sistema son la descripción de los servicios proporcionados por el sistema y sus restricciones operativas (24).

2.4.1 Requisitos funcionales

Para el desarrollo de la solución propuesta se evidenció la necesidad de implementar un conjunto de restricciones que ayudarán a que el sistema cumpla con los requisitos del cliente.

A continuación se muestran los requisitos funcionales de la aplicación para la sincronización de datos entre la distribución GNU/Linux Nova y Android (*ver Tabla 2*):

ID del requisito	Nombre del requisito	Descripción
RF1	Listar contactos	Realiza una búsqueda de los contactos en la aplicación de “Contactos” del dispositivo móvil y los muestra en al usuario.

RF2	Listar eventos del calendario	Realiza una búsqueda de los eventos del calendario en la aplicación de “Calendario” del dispositivo móvil y los muestra en al usuario.
RF3	Listar archivos de imagen	Realiza una búsqueda de las imágenes contenidas en dispositivo móvil y las muestra al usuario.
RF4	Listar archivos de video	Realiza una búsqueda de los videos contenidos en dispositivo móvil y las muestra al usuario.
RF5	Listar archivos de audio	Realiza una búsqueda de los archivos de audio contenidos en dispositivo móvil y los muestra al usuario.
RF6	Sincronizar contactos	Realiza la sincronización de los contactos con la aplicación de Contactos de la distribución GNU/Linux Nova.
RF7	Sincronizar eventos del calendario	Realiza la sincronización de los eventos del calendario con la aplicación Calendario de la distribución GNU/Linux Nova.
RF8	Sincronizar archivos de imagen	Realiza la sincronización de los archivos en la carpeta Imágenes de la distribución GNU/Linux Nova.

RF9	Sincronizar archivos de videos	Realiza la sincronización de los archivos en la carpeta Videos de la distribución GNU/Linux Nova.
RF10	Sincronizar archivos de audio	Realiza la sincronización de los archivos en la carpeta Música de la distribución GNU/Linux Nova.
RF11	Enviar peticiones	Envía las peticiones de datos realizadas
RF12	Recibir peticiones	Recibe las peticiones de datos realizada

Tabla 2: Requisitos funcionales de la propuesta de solución

2.4.2 Requisitos no funcionales

La propuesta de solución debe presentar un conjunto de características que ayuden a un mejor rendimiento y uso de la misma.

A continuación se listan los requisitos no funcionales.

RNF1. Requerimiento de usabilidad

La aplicación debe tener una interfaz usable, de manera que permita al usuario tener un fácil acceso a todas las operaciones del sistema.

RNF2. Requerimiento de mantenibilidad

El software deberá permitir posteriores modificaciones y actualizaciones por el administrador y se debe garantizar el mantenimiento de la aplicación.

RNF3. Requerimiento de funcionalidad

La solución informática debe cumplir las funcionalidades definidas anteriormente, realizando tareas de sincronización de contactos, calendario y archivos multimedia.

2.5 Historias de usuario (HU)

Las Historias de Usuario representan la técnica utilizada para especificar los requisitos del software, realizándose una por cada característica principal del sistema; tienen el mismo propósito que los casos de uso en las metodologías de desarrollo de software

pesadas, aunque no son lo mismo ya que son escritas por los propios clientes desde su perspectiva del sistema, por lo que serán descripciones cortas y escritas en el lenguaje del usuario.

A continuación, se muestran las historias de usuario definidas para la elaboración de la solución propuesta. En las HU los tiempos estimados se escriben en semanas, mientras los tiempos reales se escriben días/días de la semana. El anexo 2 contiene las HU que no se muestran a continuación.

Número: 1	**Nombre del requisito:** Listar archivos de audio
Programador: Ronny Montano Martínez	**Iteración Asignada:** 1
Prioridad: Media	**Tiempo Estimado:** 1
Riesgo en Desarrollo: Medio	**Tiempo Real:** 2/5
Descripción: Muestra todos los archivos de audio contenidos en el dispositivo móvil.	
Observaciones:	
Prototipo de interfaz: 	

Tabla 3: Historia de usuario listar archivos de audio

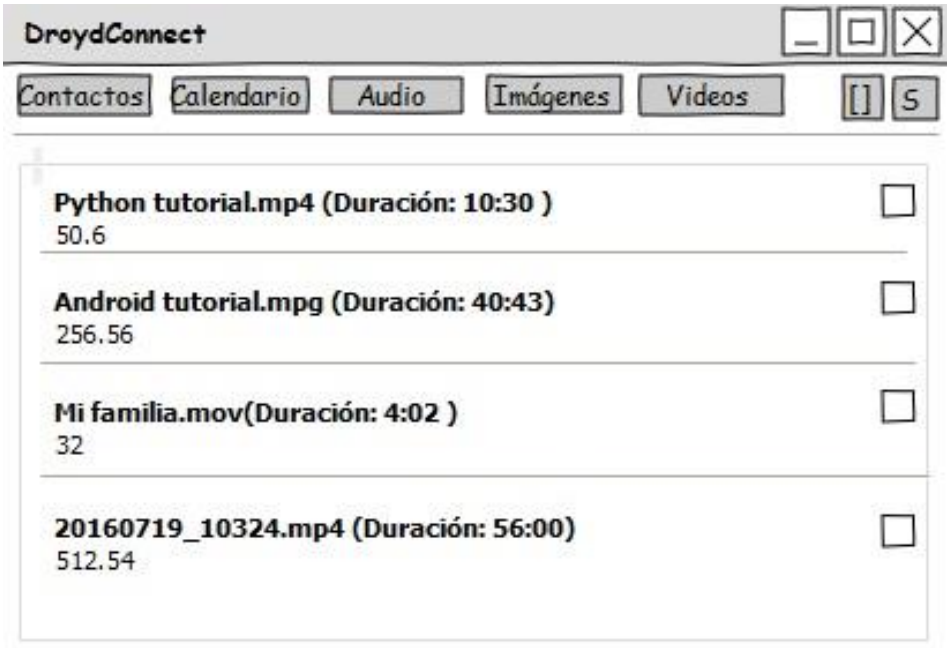
Número: 4	**Nombre del requisito:** Listar archivos de video
Programador: Ronny Montano Martínez	**Iteración Asignada:** 1
Prioridad: Media	**Tiempo Estimado:** 1
Riesgo en Desarrollo: Medio	**Tiempo Real:** 2/5
Descripción: Muestra todos los archivos de video contenidos en el dispositivo móvil.	
Observaciones:	
Prototipo de interfaz: 	

Tabla 4: Historia de usuario listar archivos de video

Número: 8	**Nombre del requisito:** Sincronizar archivos de imagen
Programador: Ronny Montano Martínez	**Iteración Asignada:** 2
Prioridad: Alta	**Tiempo Estimado:** 1
Riesgo en Desarrollo: Alto	**Tiempo Real:** 4/5

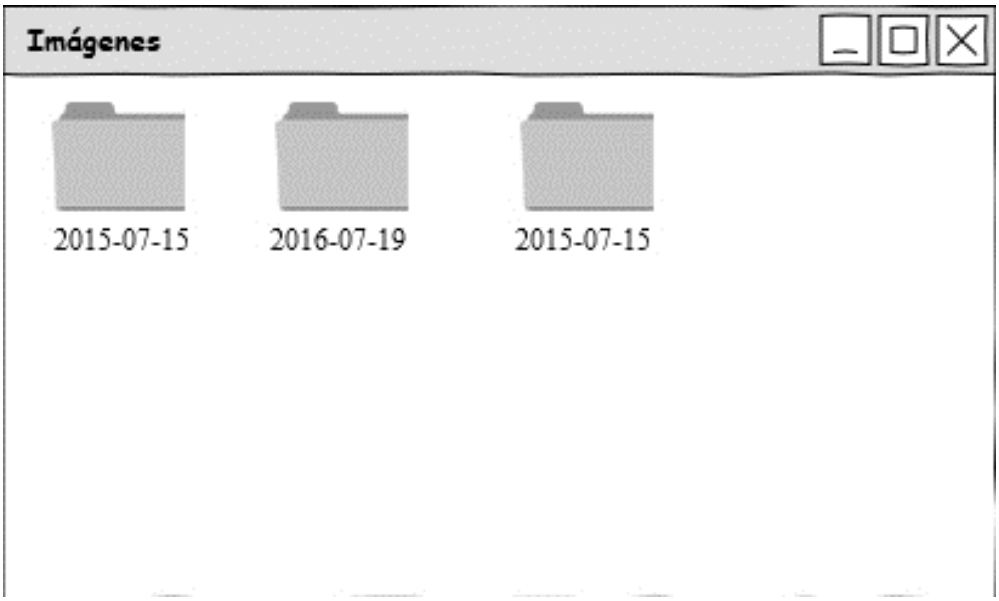
Descripción: Sincroniza las imágenes seleccionadas, organizándolas por fecha en la carpeta “Imágenes” de GNU/Linux Nova.
Observaciones:
Prototipo de interfaz: 

Tabla 5: Historia de usuario sincronizar archivos de imagen

Número: 10	**Nombre del requisito:** Sincronizar archivos de audio
Programador: Ronny Montano Martínez	**Iteración Asignada:** 2
Prioridad: Media	**Tiempo Estimado:** 1
Riesgo en Desarrollo: Medio	**Tiempo Real:** 3/5
Descripción: Sincroniza los archivos de sonido seleccionados, organizándolos por autor en la carpeta “Música ” de GNU/Linux Nova	
Observaciones:	
Prototipo de interfaz:	

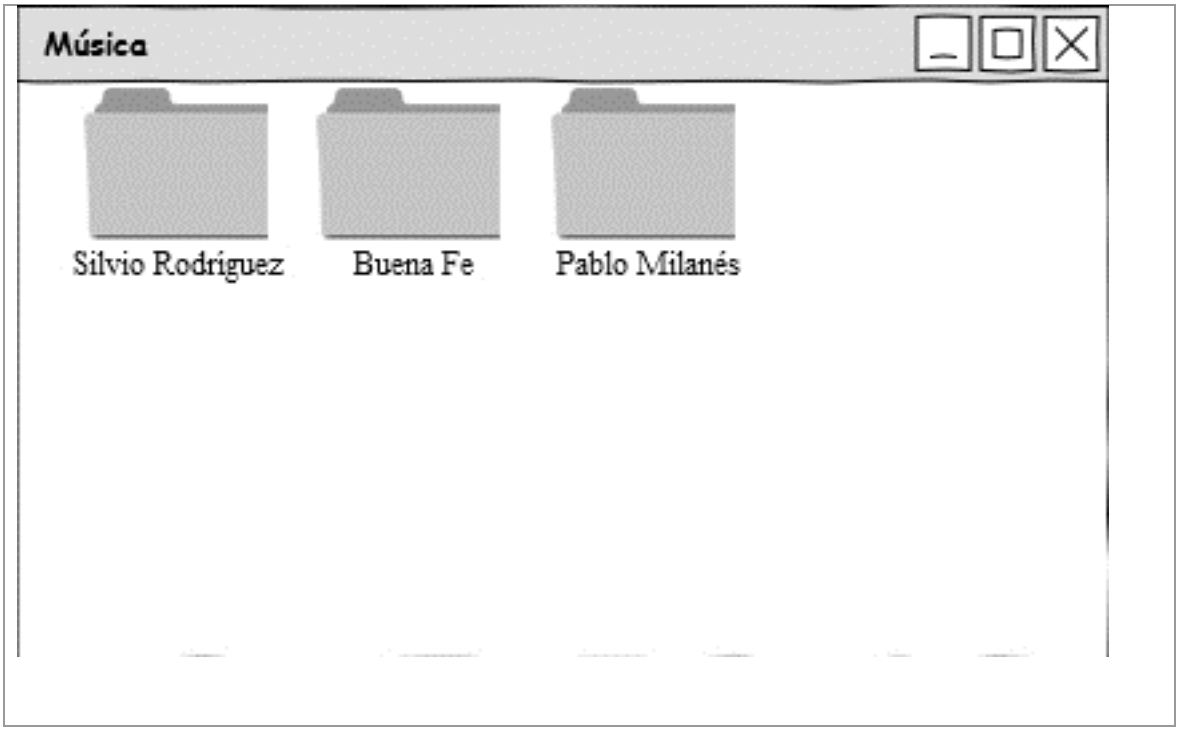


Tabla 6: Historia de usuario sincronizar archivos de audio

Número: 9	**Nombre del requisito:** Sincronizar archivos de video
Programador: Ronny Montano Martínez	**Iteración Asignada:** 2
Prioridad: Alta	**Tiempo Estimado:** 1
Riesgo en Desarrollo: Alto	**Tiempo Real:** 2/5
Descripción: Sincroniza los videos seleccionados en la carpeta “Videos” de GNU/Linux Nova.	
Observaciones:	

Prototipo de interfaz:

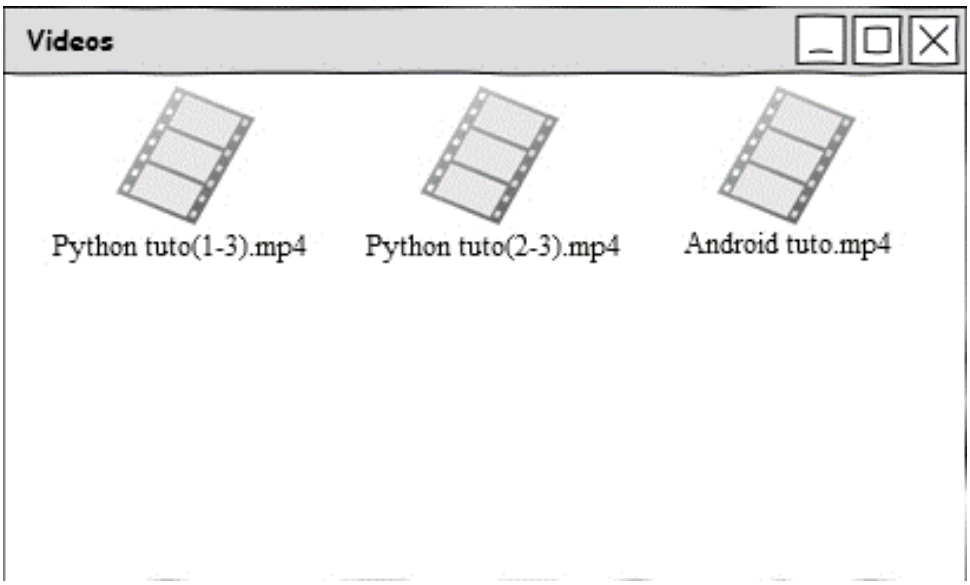


Tabla 7: Historia de usuario sincronizar archivos de video

2.6 Diagrama de paquetes

A continuación se presenta el diagrama de paquetes de la aplicación servidor (*ver Ilustración 4*). La aplicación cliente cuenta con un conjunto de clases y paquetes (*ver Anexo 4*)

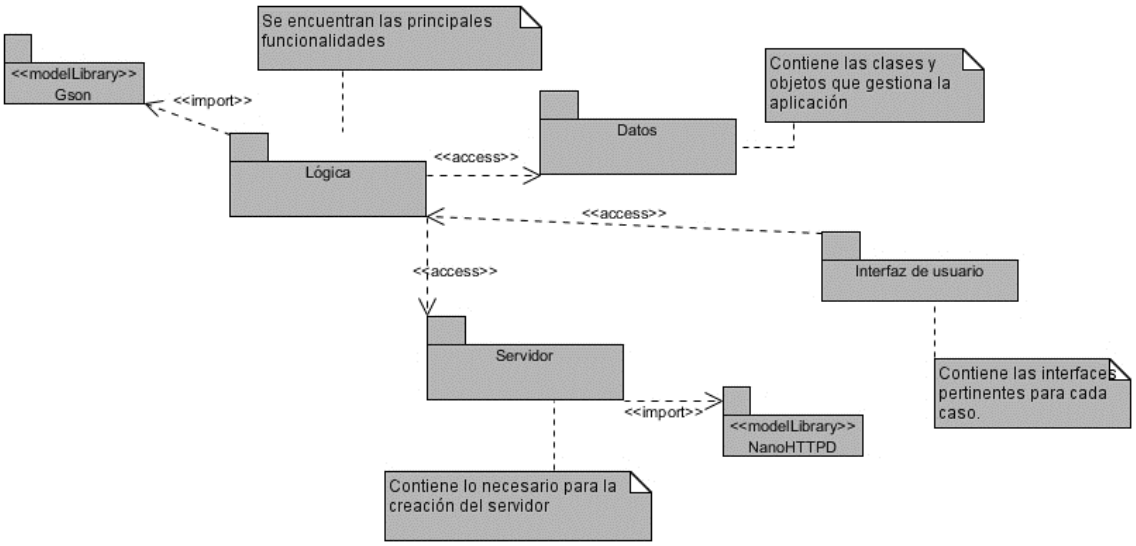


Ilustración 4: Diagrama de paquetes de la aplicación servidor

2.7 Patrones de diseño

Los patrones de diseño describen un problema que ocurre reiteradas veces y el núcleo de la solución al problema, de forma que pueden utilizarse en ilimitadas ocasiones sin

tener que hacer dos veces lo mismo. Un patrón de diseño no es más que una solución estándar para un problema común de programación. Una técnica para flexibilizar el código haciéndolo satisfacer ciertos criterios (28).

2.7.1 Patrones Grasp

Es un acrónimo de *General Responsibility Assigment Software Patterns* (patrones generales de software para asignar responsabilidades). Son utilizados para describir los principios fundamentales del diseño y la asignación de responsabilidades (28).

- **Experto**

  Consiste en asignar la responsabilidad a la clase que tiene la información necesaria para realizarla.

  Por ejemplo, la clase *SyncController* es la encargada de crear los métodos correspondientes para el recibo del elemento solicitado (*ver Ilustración 5*).

```
class SyncController:

    def __init__(self):
        self._redirect()

        self._port = 6789
        self._url = "http://localhost"

        self._contacts_url = "%s:%d/contactos" % (self._url, self._port)
        self._audio_url = "%s:%d/musica" % (self._url, self._port)
        self._images_url = "%s:%d/imagenes" % (self._url, self._port)
        self._video_url = "%s:%d/videos" % (self._url, self._port)
        self._calendar_url = "%s:%d/calendario" % (self._url, self._port)

    def _redirect(self):
        argu = ['adb', 'forward', 'tcp:6789', 'tcp:6789']
        subprocess.call(argu)

    def get_calendar_events(self):
        calendario = urllib2.urlopen(self._calendar_url)
        calendario_l = json.loads(calendario.read())
        return calendario_l

    def get_contacts(self):
        contactos = urllib2.urlopen(self._contacts_url)
        contactos_l = json.loads(contactos.read())
        return contactos_l
```

Ilustración 5: Ejemplo patrón Experto en la propuesta de solución

- **Creador**

La creación de instancias es una de las actividades más comunes en un sistema orientado a objetos. En consecuencia, es útil contar con un principio general para la asignación de las responsabilidades de creación. Si se asignan bien, el diseño puede soportar un bajo acoplamiento, mayor claridad y reutilización (28).

Ejemplo de ello, es en la clase *Server* que se crea una instancia de la clase Functionalities, para luego invocar las funcionalidades de dicha clase (*ver Ilustración 6*).

```
Functionalities fun;
func = new Functionalities(this.resolver);
```

Ilustración 6: Ejemplo patrón Creador en la propuesta de solución

- **Alta cohesión**

  La cohesión es la medida de la fuerza que une a las responsabilidades de una clase. Una clase con baja cohesión hace muchas cosas no relacionadas, o hace demasiado trabajo. Tales clases no son convenientes ya que son difíciles de mantener, de reutilizar y de entender (28).

  Por ejemplo la clase *GenericDownloader* que no hace mucho trabajo, solo el de crear una clase para la obtención de los audios contenidos en el dispositivo servidor (*ver Ilustración 7*).

```
class GenericDownloader:
    def __init__(self):
        self.object_controller = SyncController()

    def get_list_resources(self):
        pass

    def get_list_show(self):
        pass

    def sync(self, elements):
        pass
```

Ilustración 7: Ejemplo patrón Alta cohesión en la propuesta de solución

- **Bajo acoplamiento**

  El acoplamiento es una medida de la fuerza con que un elemento está conectado a otro, un elemento con bajo acoplamiento no depende demasiado de otros elementos. Una clase con alto acoplamiento confía en muchas otras clases, tales clases podrían no ser deseables ya que son muy complicadas de entender, son

difíciles de reutilizar y los cambios realizados en las clases relacionadas fuerzan cambios locales (28).

Este patrón se evidencia dentro de la aplicación en la clase *AudioDownloader* siendo de esta manera más independiente. Hay poca dependencia entre esas clases lo que permite una mayor reutilización (*ver Ilustración 8*).

```
class AudioDowloader(GenericDownloader):
    def __init__(self):
        GenericDownloader.__init__(self)

    def get_list_resources(self):
        return self.object_controller.get_songs()

    def set_elements(self, elements):
        self.elements = elements

    def get_list_show(self):
        list_elements_show = []
        self.list_elements = self.get_list_resources()

        for i in self.list_elements:
            parameter = [i['autor'], i['album'], i['title']]

            n = AudioElement(parameter)
            if n.exist():
                pass
            else:
                list_elements_show.append(i)
        return list_elements_show
```

Ilustración 8: Ejemplo patrón Bajo acoplamiento en la propuesta de solución

2.8 Conclusiones del capítulo

En el capítulo se describió la arquitectura de la solución a desarrollar, esta propuesta se basó en las características no funcionales requeridas, además fueron identificadas las principales funcionalidades a desarrollar, las cuales quedaron plasmadas en las historias de usuario para lograr un mayor entendimiento de la propuesta de solución. Además se explicaron los patrones de diseño utilizados y se concluye que, los artefactos generados por la metodología de desarrollo en esta etapa permiten tener una visión más clara de los elementos del sistema, permitiendo una mayor comprensión del mismo y guiarán de forma efectiva el capítulo de implementación y pruebas.

Capítulo 3: Implementación y pruebas

3.1 Introducción

En el presente capítulo se desarrolla lo referente a la fase de Implementación y Pruebas, los cuales son determinantes en el proceso de desarrollo de software. En la primera parte de esta fase se generan los documentos relacionados con la planificación de las iteraciones y las tareas a realizar durante la implementación. Además, se genera el código fuente en la etapa de implementación y luego los documentos relacionados con las pruebas de software.

3.2 Diagrama de componentes

A continuación se muestra el diagrama de componentes de la aplicación servidor, que expresa la relación existente entre todos los elementos que componen dicha aplicación (*ver Ilustración 9*). Para consulta del diagrama de la aplicación cliente (*ver Anexo 3*).

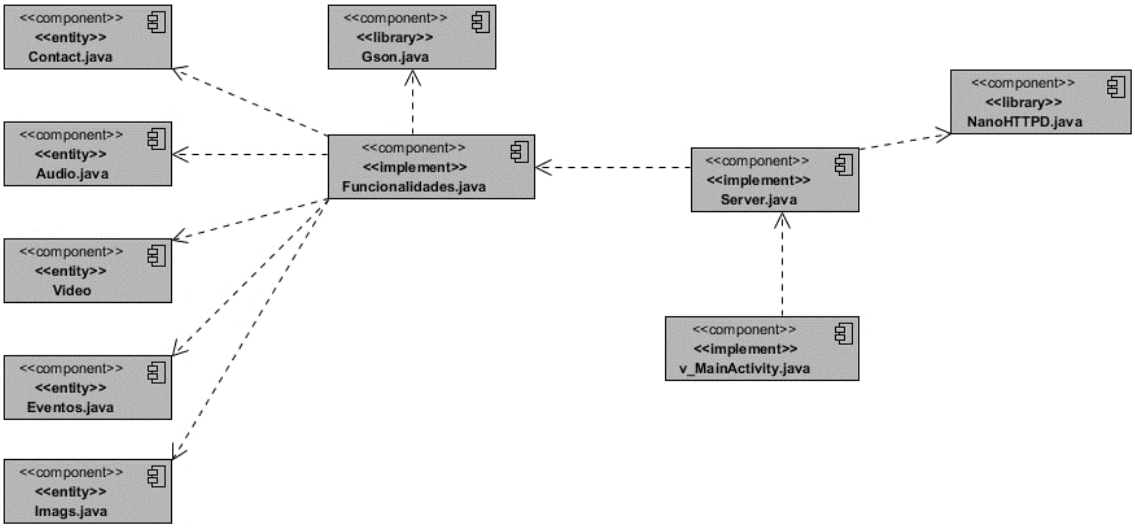


Ilustración 9: Diagrama de componentes de la aplicación servidor

3.3 Estándares de codificación

Se definen estándares de codificación porque un estilo de programación homogéneo en un proyecto permite que todos los participantes lo puedan entender en menos tiempo y que el código en consecuencia sea mantenible (29). La codificación de las aplicaciones fue realizada en inglés y siguiendo los estándares que se explican a continuación.

3.3.1 Estándares de nomenclatura

- **Nomenclatura de las clases desarrolladas en la implementación**

Se determina que los nombres de las clases comienzan con la primera letra en mayúscula y el resto en minúscula, en caso de que sea un nombre compuesto se empleará notación *CamelCase*, lo que posibilita que al leer el nombre de la clase se reconozca el propósito de la misma.
Ejemplo: SyncController.

- **Nomenclatura de las variables**

Los nombres de variables deberán estar escritos en letras minúsculas, en el caso de que sea una palabra compuesta será separada mediante guiones bajos según sea necesario para mejorar la legibilidad.
Ejemplo: parameters.

- **Nomenclatura de las funciones**

Los nombres de funciones deberán estar escritos en letras minúsculas, en el caso de que sea una palabra compuesta será separada mediante guiones bajos según sea necesario para mejorar la legibilidad.
Ejemplo: get_contacts

- **Brazas o llaves**

En la declaración de clases, métodos y bloques la apertura de llaves se hace en la misma línea.

Ejemplo: *ver Ilustración 10*

```
public Funcionalidades(ContentResolver contentResolver) {
    this.contentResolver = contentResolver;
}
```

Ilustración 10: Ejemplo del uso de las llaves

3.4 Resultados obtenidos

Para obtener la herramienta deseada se implementaron los siguientes métodos en la aplicación Android que realizará función de **servidor**:

- **list_contacts():** para realizar la búsqueda de todos los contactos en el dispositivo móvil.
- **list_calendar_events()**: para realizar la búsqueda de los eventos programados en el calendario del dispositivo móvil.
- **list_songs()**: para realizar la búsqueda de todos los archivos de sonido almacenados en el dispositivo móvil.

- **list_videos()**: para la búsqueda de todos los videos almacenados en el dispositivo móvil.
- **list_images()**: para la búsqueda de todas las imágenes que se encuentran almacenadas en el dispositivo móvil.
- **handle_uri():** recibe la URI[4] con la petición realizada por el cliente y ejecuta la acción para buscarla.

list_contacts()

Este método se implementa para la recolección de todos los contactos existentes en la aplicación Contactos del dispositivo móvil. En él se realizan consultas al proveedor de contenidos de Android, buscando el nombre, el número de teléfono y el correo electrónico de los contactos almacenados. Luego se crea una instancia de la clase Contact, que es adicionada a una lista de contactos. Después de realizada la búsqueda para todos los contactos, la lista creada es convertida a un archivo JSON, haciendo uso de la biblioteca **gson**, que es esta la encargada de convertir la lista obtenida a una cadena de tipo **String** para luego devolverla.

list_calendar_events()

Este método se implementa para realizar la búsqueda de todos los eventos almacenados en la aplicación Calendario del dispositivo móvil. En él se realizan consultas al proveedor de contenidos de Android, buscando el título y la fecha de los eventos existentes. Luego se crea una instancia de la clase Eventos, que es adicionada a una lista de eventos. Después de realizada la búsqueda, la lista creada es convertida a un archivo JSON, haciendo uso de la biblioteca **gson**, que es esta la encargada de convertir la lista obtenida a una cadena de tipo **String** para luego devolverla.

list_songs()

Este método se encarga de buscar todos los archivos de sonido almacenados en el dispositivo móvil. En él se realizan consultas al proveedor de contenidos de Android, buscando el autor, título del archivo, tamaño en MB[5], duración en minutos y la dirección de donde se encuentra almacenado en el terminal. Luego se crea una instancia de la clase Audio, que es adicionada a una lista de audios. Después de realizada la búsqueda, la lista creada es convertida a un archivo JSON, haciendo uso de la biblioteca **gson**,

[4] URI: es una cadena de caracteres que identifica los elementos de una red de forma unívoca.
[5] MB: (megabytes) unidad que sirve para medir cantidad datos informáticos.

que es esta la encargada de convertir la lista obtenida a una cadena de tipo **String** para luego devolverla.

list_videos()

Este método se encarga de buscar todos los archivos de video almacenados en el dispositivo móvil realizando consultas al proveedor de contenidos de Android, solicitando el nombre del video, tamaño en MB, duración en minutos y la dirección de donde se encuentra almacenado en el terminal. Luego se crea una instancia de la clase Video, que es adicionada a una lista de videos. Después de realizada la búsqueda, la lista creada es convertida a un archivo JSON, haciendo uso de la biblioteca **gson**, que es esta la encargada de convertir la lista obtenida a una cadena de tipo **String** para luego devolverla.

list_images()

Este método es utilizado para buscar todas las imágenes almacenadas en el dispositivo móvil realizando consultas al proveedor de contenidos de Android, solicitando el nombre de la imagen, dimensiones, tamaño en MB y la dirección de donde se encuentra almacenada en el terminal. Luego se crea una instancia de la clase Video, que es adicionada a una lista de videos. Después de realizada la búsqueda, la lista creada es convertida a un archivo JSON, haciendo uso de la biblioteca **gson**, que es esta la encargada de convertir la lista obtenida a una cadena de tipo **String** para luego devolverla.

handle_uri()

Este método es el encargado de manipular la petición realizada por el usuario. Recibe como parámetro una petición de tipo URI, que indica lo que debe hacer. Las opciones que tiene se muestran a continuación:

- Si el usuario realiza la petición de *"CONTACTOS"*, este método ejecuta la funcionalidad list_contacts().

```
case "contactos":
    return newFixedLengthResponse(func. list_contacts ());
```

- Si el usuario realiza la petición de *"CALENDAR"*, este método ejecuta la funcionalidad list_calendar_events().

```
case "calendario":
  return newFixedLengthResponse(func.list_calendar_events());
```

- Si el usuario realiza la petición de *“MUSICA”*, este método ejecuta la funcionalidad list_songs().

```
case "musica":
    return newFixedLengthResponse(func.list_songs());
```

- Si el usuario realiza la petición de *“VIDEOS”*, este método ejecuta la funcionalidad list_videos().

```
case "videos":
    return newFixedLengthResponse(func.list_videos());
```

- Si el usuario realiza la petición de *“IMÁGENES”,* este método ejecuta la funcionalidad list_images().

```
case "imagenes":
    return newFixedLengthResponse(func.list_images());
```

En la aplicación **cliente** escrita en Python fueron necesarias las implementaciones de los métodos mostrados a continuación:

- **_init__():** de la clase SyncController que es el encargado de realizar la conexión con el servidor creando las direcciones para su posterior uso
- **get_contacts():** tiene como finalidad realizar la petición de los contactos a la aplicación servidor haciendo uso de la biblioteca urllib2, enviando como mensaje una URI, que incluye la palabra “contactos”. Luego de realizada la petición este espera la respuesta del servidor, que es recibida usando métodos de dicha biblioteca. La información recibida es analizada usando el método **json.loads()**, que retorna la información recibida preparada para mostrar al usuario.
- **get_calendar_events():** tiene como finalidad realizar la petición de los contactos a la aplicación servidor haciendo uso de la biblioteca urllib2, enviando como mensaje una URI, que incluye la palabra “calendario”. Luego de realizada la petición este espera la respuesta del servidor, que es recibida usando métodos de dicha biblioteca. La información recibida es analizada usando el método **json.loads()**, que retorna la información recibida preparada para mostrar al usuario.
- **get_songs():** tiene como finalidad realizar la petición de los contactos a la aplicación servidor haciendo uso de la biblioteca urllib2, enviando como mensaje una URI, que incluye la palabra “música”. Luego de realizada la petición este espera la respuesta del servidor, que es recibida usando métodos de dicha biblioteca. La información recibida es analizada usando el método **json.loads()**, que retorna la información recibida preparada para mostrar al usuario.

- **get_images():** tiene como finalidad realizar la petición de los contactos a la aplicación servidor haciendo uso de la biblioteca urllib2, enviando como mensaje una URI, que incluye la palabra “contactos”. Luego de realizada la petición este espera la respuesta del servidor, que es recibida usando métodos de dicha biblioteca. La información recibida es analizada usando el método **json.loads()**, que retorna la información recibida preparada para mostrar al usuario.
- **get_video():** tiene como finalidad realizar la petición de los contactos a la aplicación servidor haciendo uso de la biblioteca urllib2, enviando como mensaje una URI, que incluye la palabra “contactos”. Luego de realizada la petición este espera la respuesta del servidor, que es recibida usando métodos de dicha biblioteca. La información recibida es analizada usando el método **json.loads()**, que retorna la información recibida preparada para mostrar al usuario.

Por cada tipo de elemento a sincronizar fue necesario la creación de una clase *Downloader* encargada de realizar la sincronización de los datos una vez seleccionado por el usuario, a continuación se muestra la descripción de dichas funcionalidades.

ContactDownloader, funcionalidad **sync(),** utilizada para la sincronización de los contactos**:** recibe una lista de los contactos seleccionados por el usuario para ser sincronizados, y los añade a la aplicación Contactos de GNU/Linux Nova.

CalendarDownloader, funcionalidad **sync(),** utilizada para la sincronización de los eventos del calendario**:** recibe una lista de los eventos seleccionados por el usuario para ser sincronizados, y los añade a la aplicación Calendario de GNU/Linux Nova.

AudioDownloader, funcionalidad **sync(),** utilizada para la sincronización de los archivos de audio**:** recibe una lista de los archivos de audio seleccionados por el usuario para ser sincronizados, y los añade a la carpeta Música de GNU/Linux Nova, organizándola de la siguiente forma: autor/álbum/titulo.

ImagenDownloader, funcionalidad **sync(),** utilizada para la sincronización de los archivos de imagen**:** recibe una lista de los archivos de imagen seleccionados por el usuario para ser sincronizados, y los añade a la carpeta Imágenes de GNU/Linux Nova, organizándola de la siguiente forma: fecha_captura/nombre_de_la_foto.

VideoDownloader, funcionalidad **sync(),** utilizada para la sincronización de los archivos de video**:** recibe una lista de los archivos de video seleccionados por el usuario para ser sincronizados, y los añade a la carpeta Videos de GNU/Linux Nova.

3.5 Diagrama de despliegue

A continuación se muestra el diagrama de despliegue del sistema, en el cual se aprecia (*ver Ilustración 11*):

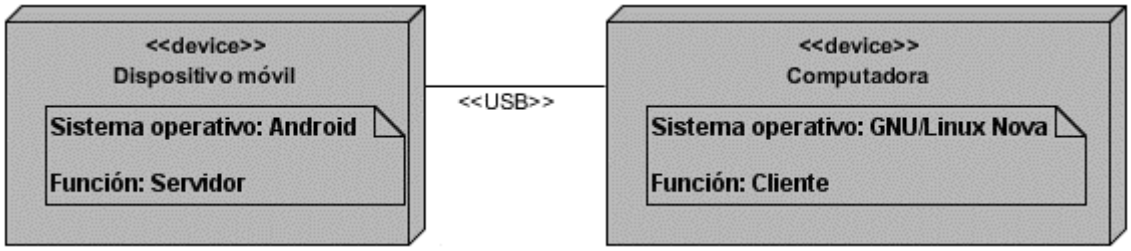


Ilustración 11: Diagrama despliegue de la aplicación

Computadora: en ella estará instalada la aplicación cliente, que representa la estación principal donde se llevaran a cabo las solicitudes de datos al servidor. Será la encargada también de mostrar los resultados de las peticiones. La distribución usada en dicha computadora es GNU/Linux Nova.

Dispositivo móvil: es quien crea el servidor, cuenta con la particularidad de que el sistema operativo instalado en él tiene que ser Android.

3.6 Verificación funcional

La calidad del software debe implementarse a lo largo de todo el ciclo de vida de un software, debe correr paralela desde la planificación del producto hasta la fase de producción de este como actividad de protección. El aspecto a considerar en el Control de la Calidad del Software es la “Prueba de Software”.

3.6.1. Pruebas de Software

Las pruebas de software es un concepto que a menudo, es conocido como verificación y validación del *software*. Son un elemento crítico para la garantía de calidad del software y representan una revisión final de las especificaciones, del diseño y de la codificación (30). Entre los de los métodos que se llevan a cabo para el proceso de prueba se encuentran los de caja negra y caja blanca.

Para un mejor acabado del producto que da respuesta a la solución propuesta se le decidió hacer pruebas en el nivel de sistema haciendo uso del método de caja negra, mediante la técnica de partición equivalente, derivándose en casos de pruebas que se verán más adelante.

El método de caja negra se centra en los requisitos funcionales del *software*. Comprueban que la funcionalidad del programa o sistema es completamente operativa.

Verifican que las entradas se aceptan de forma adecuada y la salida es correcta. Buscan errores tales como (31):

- Funciones incorrecta o ausente
- Errores de interfaces
- Errores de inicialización o finalización

A continuación se muestran algunos de los casos de prueba creados para iniciar las iteraciones de pruebas que se realizarán a la herramienta de sincronización de datos de usuario entre Android y GNU/Linux Nova. Para ver los otros casos de pruebas *ver anexo 5*.

Escenario	**Descripción**	**Respuesta del sistema**	**Flujo central**
EC 1.1 [Selecciona un contacto]	Sincroniza los Contactos	Sincroniza el contacto seleccionado a la aplicación Contactos de GNU/Linux Nova	1. Selecciona el botón Contactos del menú de opciones 2. Seleccionar utilizando el checkbox el contacto deseado 3. Seleccionar el botón sincronizar
EC 1.2 [Selecciona todos los contactos]	Sincroniza los Contactos	Sincroniza todos los contactos seleccionados a la aplicación Contactos de GNU/Linux Nova	1. Selecciona el botón Contactos del menú de opciones 2. Selecciona el botón "Opciones" 3. Selecciona el botón "Seleccionar todos" 4. Seleccionar el botón sincronizar

Tabla 8: Caso de prueba para requisito sincronizar contactos

Escenario	**Descripción**	**Respuesta del sistema**	**Flujo central**
EC 1.1 Selecciona un evento	Sincroniza los eventos del calendario	Sincroniza el evento seleccionado a la aplicación	1. Selecciona el botón Calendario del menú de opciones 2. Seleccionar utilizando el checkbox el evento

		Calendario de GNU/Linux Nova	deseado 3. Seleccionar el botón sincronizar
EC 1.2 Selecciona todos los eventos	Sincroniza los eventos del calendario	Sincroniza todos los eventos seleccionados a la aplicación Calendario de GNU/Linux Nova	1. Selecciona el botón Calendario del menú de opciones 2. Selecciona el botón "Opciones" 3. Selecciona el botón "Seleccionar todos" 4. Seleccionar el botón sincronizar

Tabla 9: Caso de prueba para requisito sincronizar calendario

Escenario	Descripción	Respuesta del sistema	Flujo central
EC 1.1 Selecciona un archivo de audio	Sincroniza los archivos de audio	Sincroniza el archivo de audio seleccionado a la carpeta Música de GNU/Linux Nova	1. Selecciona el botón Música del menú de opciones 2. Seleccionar utilizando el checkbox el archivo de audio deseado 3. Seleccionar el botón sincronizar
EC 1.2 Selecciona todos los archivos de audio	Sincroniza los archivos de audio	Sincroniza todos los archivos de audio seleccionados a la carpeta Música de GNU/Linux Nova	1. Selecciona el botón Música del menú de opciones 2. Selecciona el botón "Opciones" 3. Seleccionar el botón "Seleccionar todos" 4. Seleccionar el botón sincronizar

Tabla 10: Caso de prueba para requisito sincronizar música

Escenario	Descripción	Respuesta del sistema	Flujo central
EC 1.1 Listar contactos	El usuario selecciona la opción "Contactos" del menú de opciones	Muestra una lista con los contactos registrados en la aplicación Contactos del dispositivo móvil	1. Seleccionar el botón Contactos

Tabla 11: Caso de prueba para requisito listar contactos

Escenario	Descripción	Respuesta del sistema	Flujo central
EC 1.1 Listar archivos de imagen	El usuario selecciona la opción "Imágenes" del menú de opciones	Muestra una lista con los archivos de imagen almacenados en el dispositivo móvil	1. Seleccionar el botón Imágenes

Tabla 12: Caso de prueba para requisito listar archivos de imagen

Escenario	Descripción	Respuesta del sistema	Flujo central
EC 1.1 Listar eventos del calendario	El usuario selecciona la opción "Calendario" del menú de opciones	Muestra una lista con los eventos registrados en la aplicación Calendario del dispositivo móvil	1. Seleccionar el botón Calendario

Tabla 13: Caso de prueba para requisito listar eventos del calendario

Se realizaron un total de tres iteraciones de pruebas encontrándose un total de 12 No Conformidades, de ellas una fue un error que cuando la aplicación cliente intentaba mostrar los datos de los contactos o los calendarios pero existían algunos objetos que no contenían determinado dato. Otro de los errores fue cuando la aplicación cliente se iniciaba sin estar conectado el dispositivo móvil, 8 fueron errores del código fuente corrigiéndose los mismos en el momento en que ocurrieron y el resto fueron errores de la interfaz de consola, corrigiéndose después por el autor.

Ilustración 12: Estadísitica del número de no conformidades encontradas

3.7 Conclusiones del capítulo

A lo largo de este capítulo se diseñó el diagrama de componentes que ayuda a un mejor entendimiento de los componentes y relaciones de la solución propuesta. Se mostraron los resultados obtenidos durante la etapa de implementación ayudando a dar cumplimiento a los requisitos funcionales de la solución propuesta. El diseño del diagrama de despliegue de la propuesta de solución fue otra de las tareas de este capítulo, ayudando a lograr un mayor razonamiento del entorno de desarrollo de la solución propuesta. Se realizó la descripción de los casos de prueba basados en requisitos utilizados durante las iteraciones de pruebas la herramienta de sincronización de datos entre los sistemas operativos Android y GNU/Linux Nova, se mostraron además los resultados de las pruebas realizadas, validando las funcionalidades del sistema, haciendo uso de la técnica de caja negra.

Conclusiones

Luego de la investigación realizada se concluye que:

- El análisis realizado en el estudio de estado del arte permitió conocer herramientas existentes que realizan la tarea de sincronización pero que no resuelven el problema expuesto, aunque fueron tomadas como parte del punto de partida de la investigación.
- El análisis y diseño realizado de la solución propuesta permitió obtener un mejor entendimiento y organización del funcionamiento del sistema, que ayudó a su vez a una mayor comprensión de las tareas a desarrollar por el equipo de desarrollo.
- Se evidenció la necesidad de la creación de una solución informática que cumpliera con las especificaciones del cliente y usando funcionalidades que se analizaron en las otras aplicaciones estudiadas se realizó la implementación de la solución propuesta.
- Se necesitó la realización de pruebas a la solución informática en general que dieron como resultado una aplicación más refinada y menos propensa al mal funcionamiento.

Por lo expuesto anteriormente, se puede concluir que se desarrolló satisfactoriamente una solución informática que permite la sincronización de datos entre los dispositivos móviles inteligentes con Android y las computadoras con GNU/Linux Nova.

Recomendaciones

A continuación se muestran las recomendaciones para lograr una solución más completa:

- Agregar funcionalidades para realizar la sincronización de datos de forma inversa, es decir, introducirle al dispositivo móvil datos desde la computadora.

Referencias

1. **Google.** Android. [En línea] DigitalGlobe, 2014. [Citado el: 15 de Septiembre de 2015.] http://www.android.com/.

2. **NetMarketShare.** Market share for mobile, browsers, operating systems and search engines | NetMarketShare. [En línea] 2016. [Citado el: 28 de Abril de 2016.] https://www.netmarketshare.com/operating-system-market-share.aspx?qprid=8&qpcustomd=1&qptimeframe=M&qpsp=207&qpct=3&qpmr=10.

3. **The Statistics Portal.** Number of apps available in leading app stores as of July 2015. [En línea] Julio de 2015. [Citado el: 30 de Septiembre de 2015.] http://www.statista.com/statistics/276623/number-of-apps-available-in-leading-app-stores/..

4. **Universidad de las Ciencias Infomaticas.** Proceso de Migración de Software Libre. [En línea] 2014. [Citado el: 21 de Noviembre de 2015.] https://migracion.uci.cu/.

5. **Indiana University.** Knowledge Base. [En línea] Complaints, 2015. [Citado el: 12 de Enero de 2016.] https://kb.iu.edu/d/alkc.

6. **Tanenbaum, Andrew S.** *Sistemas operativos modernos.* México : PEARSON EDUCACIÓN, 2009. ISBN: 978-607-442-046-3.

7. **Oxford University Press.** Oxford Dictionaries. [En línea] 2016. [Citado el: 27 de Mayo de 2016.] www.oxforddictionaries.com/definition/english/smartphone.

8. **Sand Studio.** AirDroid. [En línea] Tongbu Networks, 2016. [Citado el: 23 de Marzo de 2016.] https://www.airdroid.com/es-es/.

9. **SAMSUNG.** Samsung Kies. [En línea] Samsung, 2016. [Citado el: 23 de Marzo de 2016.] http://www.samsung.com/es/support/usefulsoftware/KIES/.

10. **Wondershare.** Mobilego. [En línea] 2016. [Citado el: 23 de Marzo de 2016.] http://www.wondershare.es/mobilego/.

11. **The INove Theme. .** QtADB Your android manager. [En línea] 2016. [Citado el: 23 de Marzo de 2016.] https://qtadb.wordpress.com/features/.

12. **Oracle.** Java, cree el futuro. [En línea] Oracle. [Citado el: 14 de Enero de 2016.] http://www.oracle.com/es/technologies/java/features/index.html.

13. **GROUSSARD, Thierry.** *JAVA 7: Los fundamentos del lenguaje Java.* s.l. : Ediciones Eni, 2012.

14. **González Duque, Raúl.** *Python para todos.* Espana : Creative Commons Reconocimiento, 2008.

15. **Tröger, Enrico , y otros.** Geany. [En línea] PmWiki, 2016. [Citado el: 23 de Marzo de 2016.] http://www.geany.org/.

16. **Tröger, Enrico , y otros.** Geany. [En línea] PmWiki. [Citado el: 23 de Enero de 2016.] http://www.geany.org/.

17. **Google.** Android Developers. [En línea] Creative Commons Attribution. [Citado el: 14 de Enero de 2016.] http://developer.android.com/intl/es/tools/studio/studio-features.html.

18. **Android Developers. [En línea] Creative Commons Attribution, 2016. [Citado el: 6 de Junio de 2016.] https://developer.android.com/studio/command-line/adb.html.**

19. Introducción a JSON. [En línea] 2016. [Citado el: 23 de Marzo de 2016.] http://www.json.org/.

20. The GTK+ Team. The GTK+ Proyect. [En línea] 2015. [Citado el: 20 de Noviembre de 2015.] http://www.gtk.org/features.php.

21. Patel, Sandeep Kumar. *Instant Gson.* s.l. : Packt Publishing Ltd, 2013.

22. Targetware Informática S.A.C. Visual Parading para UML. [En línea] Targetware, 2015. [Citado el: 22 de Noviembre de 2015.] http://www.software.com.ar/p/visual-paradigm-para-uml.

23. Object Management Group. Unified Modeling Language. [En línea] 2015. [Citado el: 23 de Noviembre de 2015.] http://www.uml.org/.

24. Sommerville, Ian. *Ingeniería de Software.* Madrid : Pearson Addison Wesley, 2005. ISBN:84-7829-074-5.

25. Figueroa, Robert G. y Solis, Camilo J. *Metodologías Tradicionales vs. Metodologías Ágiles.* s.l. : Universidad Técnica Particular de Loja, Escuela de Ciencias en Computación, 2008.

26. Rodríguez Sánchez, Tamara. *Metodología de desarrollo para la Actividad productiva de la UCI.* [Documento] La Habana : Universidad de las Ciencias Informáticas, 2015.

27. Carolina Martínez, Ivette. *Modelo Conceptual/Modelo de Dominio.* [Documento] Caracas : Universidad Simón Bolívar, 2010.

28. Larman, Craig. *UML y patrones.* Madrid : LAVEL, S.A, 2003.

29. Arias Calleja, Manuel. *Carmen.Estándares de codificación.* [Documento] 2013.

30. Pressman, Roger S. *Ingeniería del Software: Un enfoque práctico.* Quinta. Madrid : McGraw-Hill, 2002.

31. Blanco Bueno, Carlos. *Ingeniería de software II, Construcción y pruebas de software.* [Documento] s.l. : Universidad de Cantabria.

Anexos

Anexo 1: Uso de Android en muestra tomada

A continuación se muestra el uso del sistema operativo Android en una muestra tomada en la Universidad de las Ciencias Informáticas (*ver Ilustración 18*).

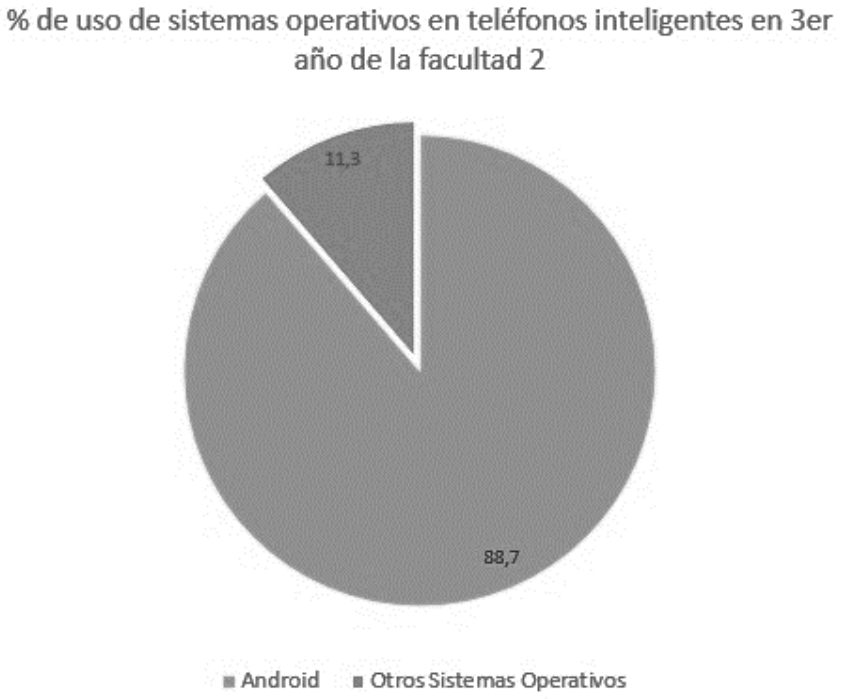


Ilustración 13: Uso de los sistemas operativos en dispositivos inteligentes

Anexo 2: Historias de usuarios

Número: 2	**Nombre del requisito:** Listar eventos del calendario
Programador: Ronny Montano Martínez	**Iteración Asignada:** 1
Prioridad: Media	**Tiempo Estimado:** 1
Riesgo en Desarrollo: Medio	**Tiempo Real:** 3/5
Descripción: Muestra al usuario los eventos registrados en la aplicación “Calendario” del dispositivo móvil.	

Observaciones:

Tabla 14: Historia de usuario listar eventos del calendario

Número: 3	**Nombre del requisito:** Listar archivos de imagen
Programador: Ronny Montano Martínez	**Iteración Asignada:** 1
Prioridad: Media	**Tiempo Estimado:** 1
Riesgo en Desarrollo: Medio	**Tiempo Real:** 3/5
Descripción: Muestra al usuario las imágenes almacenadas en el dispositivo móvil.	
Observaciones:	

Tabla 15: Historia de usuario listar archivos de imagen

Número: 6	**Nombre del requisito:** Sincronizar contactos
Programador: Ronny Montano Martínez	**Iteración Asignada:** 3
Prioridad: Alta	**Tiempo Estimado:** 1
Riesgo en Desarrollo: Alto	**Tiempo Real:** 1
Descripción: Realiza la sincronización de los contactos seleccionados con la aplicación de “Contactos” de GNU/Linux Nova	
Observaciones:	

Tabla 16: Historia de usuario sincronizar contactos

Número: 6	**Nombre del requisito:** Sincronizar eventos del calendario
Programador: Ronny Montano Martínez	**Iteración Asignada:** 3
Prioridad: Alta	**Tiempo Estimado:** 1

Riesgo en Desarrollo: Alto	**Tiempo Real:** 1
Descripción: Realiza la sincronización de los eventos del calendario seleccionados con la aplicación de “Calendario” de GNU/Linux Nova	
Observaciones:	

Tabla 17: Historia de usuario sincronizar eventos del calendario

Número: 11	**Nombre del requisito:** Enviar petición
Programador: Ronny Montano Martínez	**Iteración Asignada:** 1
Prioridad: Media	**Tiempo Estimado:** 1
Riesgo en Desarrollo: Medio	**Tiempo Real:** 1/5
Descripción: Realiza la petición de los datos solicitados por el cliente a la aplicación servidor	
Observaciones:	

Tabla 18: Historia de usuario enviar petición

Número: 12	**Nombre del requisito:** Recibir peticiones
Programador: Ronny Montano Martínez	**Iteración Asignada:** 1
Prioridad: Media	**Tiempo Estimado:** 1
Riesgo en Desarrollo: Medio	**Tiempo Real:** 1/5
Descripción: Recibe la respuesta del servidor a la solicitud realizada por el cliente.	
Observaciones:	

Tabla 19: Historia de usuario recibir petición

Anexo 3: Diagrama de componentes de la aplicación cliente

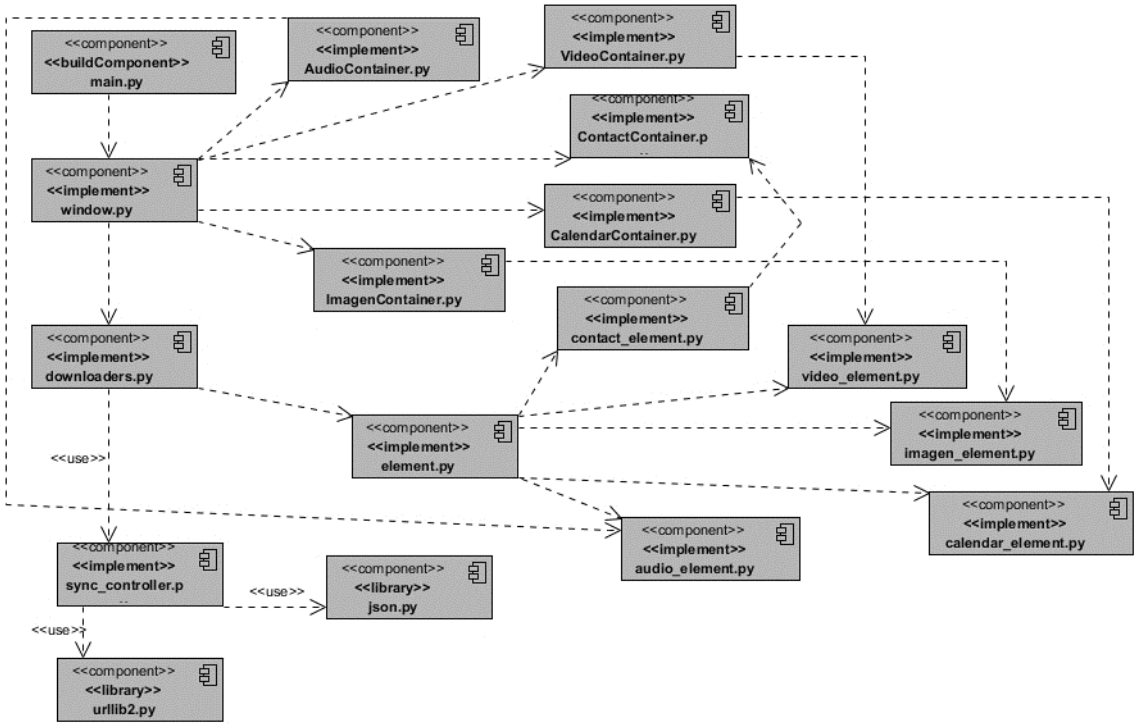


Ilustración 14: Diagrama de componentes de la aplicación cliente

Anexo 4: Diagrama de paquetes de la aplicación cliente

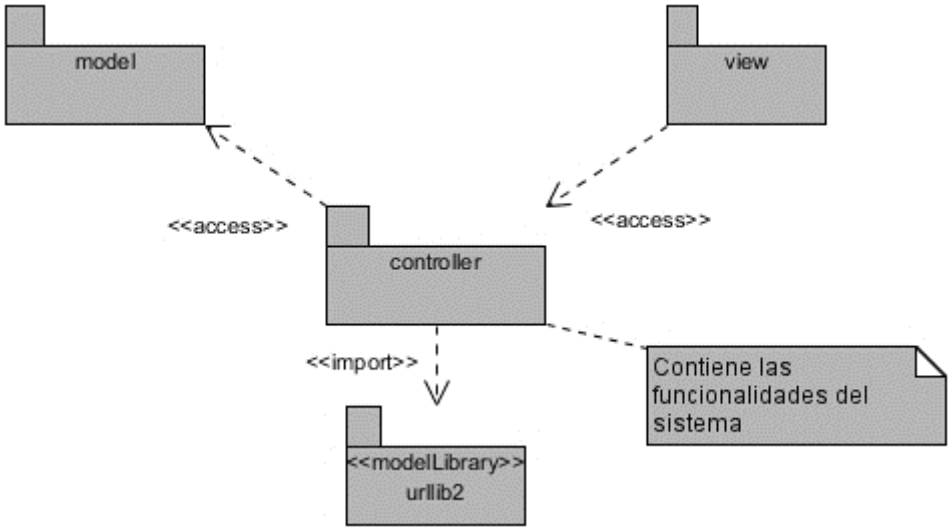


Ilustración 15: Diagrama de paquetes de la aplicación cliente

Anexo 5: Caso de pruebas

Escenario	Descripción	Respuesta del sistema	Flujo central

EC 1.1 Selecciona un archivo de imagen	Sincroniza los archivos de imagen	Sincroniza el archivo de imagen seleccionado a la carpeta Imágenes de GNU/Linux Nova	1. Selecciona el botón Imágenes del menú de opciones 2. Seleccionar utilizando el checkbox el archivo de imagen deseado 3. Seleccionar el botón sincronizar
EC 1.2 Selecciona todos los archivos de imagen	Sincroniza los archivos de imagen	Sincroniza todos los archivos de imagen seleccionados a la carpeta Imágenes de GNU/Linux Nova	1. Selecciona el botón "Imágenes" del menú de opciones 2. Selecciona el botón "Opciones" 3. Seleccionar el botón "Seleccionar todos" 4. Seleccionar el botón sincronizar

Tabla 20: Caso de prueba para requisito sincronizar imágenes

Escenario	**Descripción**	**Respuesta del sistema**	**Flujo central**
EC 1.1 Selecciona un archivo de video	Sincroniza los archivos de video	Sincroniza el archivo de video seleccionado a la carpeta Videos de GNU/Linux Nova	1. Selecciona el botón Videos del menú de opciones 2. Seleccionar utilizando el checkbox el archivo de video deseado 3. Seleccionar el botón sincronizar
EC 1.2 Selecciona todos los archivos de video	Sincroniza los archivos de video	Sincroniza todos los archivos de video seleccionados a la carpeta Videos de GNU/Linux Nova	1. Selecciona el botón Videos del menú de opciones 2. Selecciona el botón "Opciones" 3. Seleccionar el botón "Seleccionar todos" 4. Seleccionar el botón sincronizar

Tabla 21: Caso de prueba para requisito sincronizar videos

Escenario	**Descripción**	**Respuesta del sistema**	**Flujo central**
EC 1.1 Listar archivos de audio	El usuario selecciona la	Muestra una lista con los archivos de	1. Seleccionar el botón Música

	opción "Música" del menú de opciones	audio almacenados en el dispositivo móvil	

Tabla 22: Caso de prueba para requisito listar archivos de audio

Escenario	**Descripción**	**Respuesta del sistema**	**Flujo central**
EC 1.1 [Listar archivos de video]	El usuario selecciona la opción "Videos" del menú de opciones	Muestra una lista con los archivos de video almacenados en el dispositivo móvil	1. Seleccionar el botón Videos

Tabla 23: Caso de prueba para requisito listar archivos de video

Glosario de términos

Estándares abiertos: es un formato o protocolo que está sujeto a una evaluación pública completa, se puede usar sin restricciones y esté disponible por igual para todas las partes.

Firmware: es el software operativo disponible en un dispositivo Android, y está disponible en diferentes versiones diseñadas por diferentes fabricantes. Básicamente se trata de la parte específica del dispositivo del software.

Socio-adaptabilidad: Las bases tecnológicas para la informatización de Cuba, deben ser hechas por cubanos y para los cubanos, logrando inigualable adaptabilidad a las condiciones de nuestro país.

Terminal: es un dispositivo electrónico o electromecánico que se utiliza para interactuar con una computadora.