Universidad de las Ciencias Informáticas

Facultad 3

Aplicación para dispositivos móviles del módulo de reporte en el SIPACDroid

Trabajo de diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autor: Karla Solanch Suárez Mato

Tutor: Ing. Samuel Ojeda Pereira

Co-tutora: Ing. Dorisley Morales Suárez

La Habana, 2020

Año 61 de la Revolución

DECLARACIÓN DE AUTORÍA

Declaro que soy la única autora del presente trabajo y autorizo a la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

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

_____	_____
Karla Solanch Suarez Mato	**Ing. Samuel Ojeda Pereira**
Autor	**Tutor**

_____

Dorisley Morales Suarez

Co-tutora

DATOS DE CONTACTO

Ing. Samuel Ojeda Pereira: Ingeniero en Ciencias Informáticas.

Ing. Dorisley Morales Suárez: Ingeniera en Ciencias Informáticas.

AGRADECIMIENTOS

AGRADECIMIENTOS

Les agradezco a mi mama y a mi papa por siempre estar junto a mí apoyándome en todo y confiar en mí sin dudar ni un solo momento que yo lo lograría. Y a pesar de mi carácter para mí ustedes son los seres más importantes de mi vida los quiero y los amo muchísimo, siempre los cuidaré, por ustedes soy quien soy y si me he convertido en ingeniera ha sido por ustedes y para que siempre estén orgullosos de mí.

Le agradezco especialmente a mi tío Juan Carlos y a mi tía Ana por ayudarme tanto sin ustedes no hubiera podido tener las laptops que tuve a lo largo de la carrera siempre estuvieron al tanto de lo que necesitaba y sé que están orgullosos de mí yo también los quiero mucho. Ambos sé que tuvieron mucha paciencia conmigo pues sé que lo que han hecho por mí requiere mucho sacrificio para ustedes siempre estaré en deuda. Los dos siempre están al tanto de mis padres eso es muy valioso para mí pues mis padres son lo más grande que yo tengo, mil gracias.

Les agradezco a todos aquellos amigos en especial a Milena y Ailyn que conocí en esta universidad se convirtieron en mi familia y ellos saben más que nadie que más que ayudarme y apoyarme fueron para mí como hermanos y hasta como padres, pues estuvieron en los momentos más duros que pase a lo largo de estos 5 años dándome fuerza y consejos muy valiosos nunca los olvidare.

Le agradezco muchísimo a Claudia Bravo, profe me ayudaste en un momento difícil de mi carrera y de mi vida personal con tus consejos y apoyo. Y especialmente en estos últimos meses que has sido un pilar importante en la realización de mi tesis a pesar de estar lejos, gracias nuevamente pues nunca me abandonaste. Sin ti no hubiera sido posible lograr esto.

A mi tutor Samuel por haberme ayudado y apoyado en todo lo que necesité y por haberme aconsejado en todo lo que estuvo a su alcance.

A las profesoras Yisel y Dorisley que se preocuparon mucho y me ayudaron con sus consejos y sugerencias para lograr un producto final con calidad.

Les agradezco a todos aquellos profesores que participaron en mi formación como ingeniera y que me ayudaron muchísimo e hicieron posible que yo estuviera ahora mismo donde estoy.

Le agradezco también a mi tía Neyi y mi hermana Lisandra, ambas de una forma u otra estuvieron ahí apoyándome siempre.

AGRADECIMIENTOS

Sin olvidar a mi tío Jorge Luis pues me ayudó muchísimo gracias tío por reparar mi laptop en un momento tan importante sin ti no hubiera podido terminar mi tesis.

Y de último, no por ser menos importante todo lo contrario, le agradezco a mi choli por quererme tanto. Yo te quiero mucho mi vida eres muy importante para mí, me siento orgullosa de que ella este cerca de mí a pesar de ser tan pequeña eres una súper niña el regalo más grande que dios me ha dado.

DEDICATORIA

A *mis padres por ser para mí, el mayor ejemplo de constancia, dedicación y amor, por ser mi fortaleza e inspiración diaria, ha sido y es, un privilegio ser su hija, gracias por darme siempre lo mejor que pudieron, gracias mamá, gracias papá, ahora me toca a mí.*

A *mi pequeña sobrina que es mi gran tesoro.*

Resumen

La Instrucción No.1 del Presidente de los Consejos de Estado y de Ministros para la Planificación de los Objetivos y Actividades en los Órganos, Organismos de la Administración Central del Estado (OACE), entidades nacionales y administraciones locales del poder popular, constituye el documento rector de la planificación en Cuba. El Sistema de Planificación de Actividades (SIPAC) se basa en la instrucción referida anteriormente. La cual, tiene integrado el módulo de Reporte que muestra una vista previa del plan de trabajo según el criterio de búsqueda introducido, sea por el periodo de creado el plan o por la clasificación en individual, mensual y anual.

SIPACDroid es un sistema para tecnologías móviles que permite brindar y consumir servicios REST del SIPAC. La presente investigación propone una mejora de la usabilidad en SIPACDroid. En cuanto a los distintos tamaños y resoluciones de pantallas en los diferentes dispositivos móviles que no permiten visualizar todas las actividades del calendario. La salida del sistema con la información estadística de las actividades programadas en los planes de trabajo que no presenta un diseño, formato y análisis organizado que pueda ser usado en la toma de decisiones de las instituciones. Para lograr lo anteriormente planteado, se desarrolla una aplicación para generar los reportes de los planes que se encuentran registrados en SIPAC. Se utilizan los métodos teóricos y empíricos de la investigación, se propone la metodología de desarrollo para la actividad productiva de la UCI y se ejecutan las pruebas para validar el que producto se encuentra correctamente funcional.

Palabras Claves: dispositivos móviles, plan, reportes, servicios REST, SIPAC, usabilidad.

ÍNDICE

ÍNDICE

ÍNDICE

ÍNDICE DE FIGURA

ÍNDICE DE TABLA

INTRODUCCIÓN

La Instrucción No.1 del Presidente de los Consejos de Estado y de Ministros para la Planificación de los Objetivos y Actividades en los Órganos, Organismos de la Administración Central del Estado (OACE), entidades nacionales y administraciones locales del poder popular, constituye el documento rector de la planificación en Cuba. La misma tiene como objetivo establecer el procedimiento para llevar a cabo el proceso de planificación del gobierno, que permita dar cumplimiento a los acuerdos y resoluciones aprobadas en el VI Congreso del Partido Comunista de Cuba, las decisiones de la Asamblea Nacional del Poder Popular, el Consejo de Ministros y la actualización de los planes de la economía. (Consejo de ministro, y otros, 2012)

El Sistema de Planificación de Actividades (SIPAC) se basa en la instrucción referida anteriormente. Forma parte del paquete de soluciones integrales de gestión Xedro para las entidades presupuestadas y empresariales, el cual está basado en los principios de independencia tecnológica y posee funcionalidades generales de los procesos y las particularidades de la economía cubana. Sus características funcionales fueron diseñadas de acuerdo con las políticas emitidas por los organismos rectores del Estado Cubano (UCI, 2017).

Las actividades planificadas en la organización tributan a lograr el cumplimiento de los objetivos trazados. SIPAC permite interrelacionar objetivos de trabajo y actividades en tiempo real; garantizando el seguimiento del desarrollo y cumplimiento de los objetivos y tareas principales en las entidades. Es una solución que informatiza los procesos de Concepción y Ejecución para la planificación a largo plazo (planificación estratégica), así como para la planificación a mediano y corto plazo (planificación operativa y personal). Esto posibilita una mayor coincidencia entre lo que aspira la dirección y lo que debe proponerse cada miembro de la organización, garantizando que todo el personal se encuentre identificado con las tareas a desempeñar, las metas a alcanzar y las prioridades establecidas.

SIPAC tiene varios módulos que permiten y garantizan la planificación de sus actividades y los niveles de acceso por cada uno de los usuarios, entre los que se encuentran: el módulo de Plan donde se crean los planes estratégicos de la entidad, se controlan y evaluan la ejecución de las actividades planificadas y las variaciones del plan inicial de acuerdo a las actividades suspendidas o canceladas y la introducción de otras nuevas. Los planes contienen todas las actividades planificadas en la organización para lograr el cumplimiento de los objetivos trazados. Además, se clasifican según las categorías en plan de trabajo individual, mensual y anual.

INTRODUCCIÓN

El plan de trabajo individual es creado por cada una de las personas de la entidad y se le asocia sus actividades diarias con una programación previa a la fecha que se debe realizar. El plan de trabajo mensual y anual es realizado por la entidad, donde se le asocian las actividades generales que debe cumplir el personal durante el periodo de un mes para el plan de trabajo mensual o durante el periodo de un año para el anual. En el módulo de Reporte del Plan se muestra una vista previa del plan de trabajo según el criterio de búsqueda introducido, sea por el periodo de creado el plan o por la clasificación en individual, mensual y anual. Luego, se genera un documento en formato pdf con toda la información asociada al plan seleccionado. Lo que contribuye a la toma de decisiones de las diferentes instituciones

La norma ISO/IEC 25010: 25010 mejora la gestión de información y el proceso de evaluación de la calidad del software. Define la usabilidad como la eficiencia y satisfacción con la que un producto permite alcanzar objetivos específicos a usuarios específicos en un contexto de uso específico. (Grisales,2019).

El Sistema de Planificación de Actividades SIPAC en su versión 3.0 tiene la capacidad de adaptarse a entornos específicos como Android, con la aplicación SIPACDroid. Actualmente, los distintos tamaños y resoluciones de pantallas en los diferentes dispositivos móviles no permiten visualizar todas las actividades del calendario y la cantidad de información que se puede presentar a la vez. La baja resolución de la pantalla degrada la calidad de la información mostrada. La salida del sistema con la información estadística de las actividades programadas en los planes de trabajo no presenta un diseño, formato y análisis organizado que pueda ser usado en la toma de decisiones de las instituciones. Dichos aspectos, están relacionados con la usabilidad.

Los atributos de usabilidad propuestos por Nielsen 1993, y su aplicación en el campo de los dispositivos móviles, hacen referencia a la satisfacción subjetiva cuando un usuario deberá estar satisfecho y cuando usa un sistema. Debe tener una experiencia agradable de uso, lo cual redundará en una opinión positiva hacia el mismo. La satisfacción del usuario depende principalmente de la correcta funcionalidad del equipo y de que la aplicación cumpla con las necesidades del usuario (ISO/IEC, 2014).

A partir de la situación antes descrita se ha identificado el siguiente **problema a resolver:** ¿cómo aumentar la usabilidad de la información disponible de los planes de trabajo en SIPACDroid?

Estableciendo como **objeto de estudio:** Aplicaciones para dispositivos móviles, enmarcado en el **campo de acción**: Aplicación para hacer reportes, en dispositivos móviles.

INTRODUCCIÓN

Para darle solución al problema planteado se define como **objetivo general:** Desarrollar una aplicación móvil que permita generar reportes de los planes de trabajo, consumiendo los servicios REST del SIPAC para aumentar la usabilidad de la información disponible.

Para guiar la investigación se define como **objetivos específicos**:

1. Construir el marco teórico de la investigación a partir de la búsqueda y revisión bibliográfica de las aplicaciones de reportes para dispositivos móviles.
2. Realizar el análisis y el diseño de una aplicación móvil que permita generar reportes de los planes de trabajo, consumiendo los servicios REST del SIPAC.
3. Implementar una aplicación móvil que permita generar reportes de los planes de trabajo, consumiendo los servicios REST del SIPAC.
4. Validar la aplicación móvil que permita aumentar la usabilidad de la información disponible de los planes de trabajo en SIPACDroid.

Se plantea como **idea a defender:** Si se desarrolla una aplicación móvil que permita generar reportes de los planes de trabajo, entonces se aumentará la usabilidad de la información disponible.

Con la misión de obtener conocimientos necesarios que hagan posible la materialización del objetivo general, se han utilizado diferentes tipos de métodos de investigación teóricos y empíricos, los cuales se describen a continuación:

Métodos teóricos:

- **Histórico-lógico**: a través de este método se realizó un estudio sobre las aplicaciones móviles que generan reportes, se realizó un análisis para entender todos los procesos del módulo de Reportes del SIPAC y se analizaron los diferentes servicios REST que se implementan en dicho módulo.
- **Método sistémico:** su aplicación permitió determinar los componentes de las actividades en el SIPAC permitiendo identificar la interrelación entre ellos y con los demás elementos que conforman el sistema, posibilitando a su vez a llegar a una mejor comprensión del negocio a informatizar.
- **Analítico-sintético:** ha sido utilizado en el análisis de la documentación identificada, permitiendo extraer los elementos que propicien la solución a la problemática planteada, así como la síntesis de los elementos necesarios para la selección de las tecnologías y metodologías adecuadas para el desarrollo de la aplicación Android propuesta.

Métodos empíricos:

- **Observación**: la observación científica como método consiste en la percepción directa del objeto de investigación. Dicho método ha sido aplicado en la observación de la apariencia de las páginas web y el tamaño de la pantalla de los diferentes dispositivos móviles permitiendo compararlas y establecer relaciones entre ellas.
- **Entrevista no estructurada o abierta:** se ejecutaron entrevistas abiertas con la finalidad de conocer los detalles de la situación problemática descrita. Teniendo en cuenta, elementos de resolución de la pantalla en las vistas de los calendarios de diferentes dispositivos móviles.

CAPÍTULO 1: MARCO TEÓRICO REFERENCIAL DE LA APLICACIÓN PARA DISPOSITIVOS MÓVILES DEL MÓDULO DE REPORTES EN EL SIPACDROID

En este capítulo se va a investigar las aplicaciones móviles que generan reportes para ello se realiza la elaboración del marco teórico donde se exponen los conceptos asociados a la solución de la problemática y las diferentes soluciones existentes a nivel mundial. También se describen y caracterizan la metodología, herramientas y tecnologías a utilizar para el desarrollo de la aplicación SIPACDroid.

1.1 Conceptos asociados al dominio del problema

1.1.1 Reportes

Los reportes informáticos son una excelente manera de organizar y presentar información que se encuentra contenida en una base de datos. Su función principal es aplicar un formato definido y mostrar los datos mediante un diseño atractivo, que sea fácil de interpretar por los usuarios. De esta forma se les proporcionan a los datos una mayor utilidad. Es fundamental señalar que los resultados obtenidos a través de un reporte pueden ser manipulados, pero sin alterar la fuente, ya que depende de otros aspectos o áreas del sistema. De esta forma, los reportes reflejan si se han cumplido los objetivos y metas de una empresa y en base a ello poder tomar las mejores decisiones que garanticen el crecimiento de las organizaciones (Ojeda, 2018).

A continuación, se muestran las plantillas que se deben utilizar para generar los reportes con cada una de las tablas y elementos específicos. Las plantillas son los formatos que aparecen en la Instrucción No.1 del Presidente de los Consejos de Estado y de Ministros para la Planificación de los Objetivos y Actividades en los Órganos, Organismos de la Administración Central del Estado (OACE), entidades nacionales y administraciones locales del poder popular, constituye el documento rector de la planificación en Cuba.

Reporte del plan de trabajo individual: El reporte debe contener: el nombre de la persona que aprueba el plan que puede ser el directivo de la entidad o el jefe inmediato. Una descripción del documento con la cantidad de hojas, la fecha que corresponde al mes y año, y las tareas principales que son tareas globales que tienen otras tareas asociadas. Los días de la semana en correspondencia con la numeración desde el primer día hasta el último, mostrando en cada campo la información de la programación de la actividad asociada al plan. Se puede visualizar un ejemplo en el Anexo 1.

Reporte del plan de trabajo mensual: El reporte debe contener: el nombre de la persona que aprueba el plan que puede ser el directivo de la entidad o el jefe inmediato. Una descripción del

documento con la cantidad de hojas, la fecha que corresponde al mes y año, y las tareas principales que son tareas globales que tienen otras tareas asociadas. Un listado con el nombre de la actividad, la hora y lugar, la fecha, la persona que dirige, los participantes y las observaciones que sería una breve descripción de organización de la actividad. Se puede visualizar un ejemplo en el Anexo 2.

Reporte del resumen de cumplimiento: El reporte debe contener: el nombre de la entidad. Una descripción del documento con la clasificación del mismo, la fecha que corresponde al mes y año. Un resumen cuantitativo que evidencie: el total de tareas planificadas, de las cuales: las cumplidas incumplidas y modificadas, y las extra planes que son las tareas que aparecieron después de haber sido aprobada la planificación por el directivo de la entidad. Además, el listado de las tareas incumplidas y las respectivas causa de su incumplimiento. El listado de las tareas extra planes, el autor que la origina y el motivo. Por último, un resumen cualitativo con los lineamientos que se rige la institución. Se puede visualizar un ejemplo en el Anexo 3.

Reporte del resumen de cumplimiento: El reporte debe contener: el nombre y cargo de la persona que aprueba el plan de actividades, el título de la coordinación de las actividades para el año, con la fecha del mismo. Un calendario que muestra todos los meses del año con una trazabilidad de los días que tienen actividades programadas, deben estar marcadas con una (X). Se puede visualizar un ejemplo en el Anexo 4.

1.1.2 Usabilidad

La usabilidad se compone de dos tipos de atributos: Atributos cuantificables de forma objetiva, como son la eficacia o número de errores cometidos por el usuario durante la realización de una tarea, y eficiencia o tiempo empleado por el usuario para la consecución de una tarea.
Atributos cuantificables de forma subjetiva, como es la satisfacción de uso, medible a través de la interrogación al usuario, y que tiene una estrecha relación con el concepto de usabilidad percibida (Hassan Montero, 2004).

Shackel

Uno de los primeros autores en el campo para reconocer la importancia de la ingeniería de usabilidad y la relatividad del concepto de usabilidad es (Shackel, 1991). Shackel define un modelo donde la aceptación del producto es el más alto concepto. La usabilidad se define como:
"la usabilidad de un sistema es la capacidad humana en términos funcionales que deban utilizarse con facilidad y eficacia por el rango especificado de usuarios, recibir una formación específica de apoyo, para cumplir con el rango especificado de tareas, dentro del rango especificado de escenarios (Shackel, 1991).

La ISO 9241-11 proporciona la definición de la usabilidad que se utiliza más a menudo en las normas ergonómicas. La usabilidad se define como: "el grado en que un producto puede ser utilizado por determinados usuarios para conseguir objetivos específicos con efectividad, que es el grado en que las metas previstas por el usuario se logran, la eficiencia, relacionada con los recursos que tienen que ser invertidos para lograr los objetivos y la satisfacción; como la medida en que el usuario encuentra el uso del producto aceptable, en un contexto de uso específico (Macleod, 1994).

La norma ISO 25010: 25010, describe la usabilidad cómo el grado con el que un producto puede ser usado por usuarios específicos para alcanzar objetivos específicos con efectividad, eficiencia y satisfacción, en un contexto de uso específico. La norma define como especificar y medir la usabilidad de productos y aquellos factores que tienen un efecto en la misma; también destaca que la usabilidad en terminales con pantalla de visualización es dependiente del contexto de uso y que el nivel de usabilidad alcanzado dependerá de las circunstancias específicas en las que se utiliza el producto. El contexto de uso lo forman los usuarios, las tareas a realizar, el equipamiento (hardware, software y materiales), así como también los entornos físicos y sociales que pueden influir en la facilidad de uso de un producto. De la definición anterior se puede observar que la usabilidad está relacionada con los atributos de una aplicación o sistema, así como también de su contexto; se entiende por atributo la característica o propiedad de una aplicación de software. En la norma mencionada anteriormente los atributos considerados son los siguientes (Grisales.2019):

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 (Grisales, 2019).

1.1.3 Servicios REST

REST es una interfaz entre sistemas que usa HTTP para obtener datos o generar operaciones sobre esos datos en todos los formatos posibles, como XML y JSON. Es una alternativa en auge a otros protocolos estándar de intercambio de datos.

El formato más usado en la actualidad es el formato JSON, ya que es más ligero y legible en comparación al formato XML. Elegir uno será cuestión de la lógica y necesidades de cada proyecto (Django REST framework, 2018).

Características de REST

Protocolo cliente/servidor sin estado: cada petición HTTP contiene toda la información necesaria para ejecutarla, lo que permite que ni cliente ni servidor necesiten recordar ningún estado previo para satisfacerla. Aunque esto es así, algunas aplicaciones HTTP incorporan memoria caché. Se configura lo que se conoce como **protocolo cliente-caché-servidor sin estado**: existe la posibilidad de definir algunas respuestas a peticiones HTTP concretas como cacheables, con el objetivo de que el cliente pueda ejecutar en un futuro la misma respuesta para peticiones idénticas. De todas formas, que exista la posibilidad no significa que sea lo más recomendable (Django REST framework, 2018).

Los objetos en REST siempre se manipulan a partir de la URI. Es la URL y ningún otro elemento el identificador único de cada recurso de ese sistema REST. La URL facilita acceder a la información para su modificación o borrado, o, por ejemplo, para compartir su ubicación exacta con terceros (Django REST framework, 2018).

Interfaz uniforme: para la transferencia de datos en un sistema REST, este aplica acciones concretas (POST, GET, PUT y DELETE) sobre los recursos, siempre y cuando estén identificados con una URL. Esto facilita la existencia de una interfaz uniforme que sistematiza el proceso con la información (Django REST framework, 2018).

Sistema de capas: arquitectura jerárquica entre los componentes. Cada una de estas capas lleva a cabo una funcionalidad dentro del sistema REST (Django REST framework, 2018).

Uso de hipermedios: Ese concepto llevado al desarrollo de páginas web es lo que permite que el usuario puede navegar por el conjunto de objetos a través de enlaces HTML. En el caso de una API REST, el concepto de hipermedia explica la capacidad de una interfaz de desarrollo de aplicaciones de proporcionar al cliente y al usuario los enlaces adecuados para ejecutar acciones concretas sobre los datos (Django REST framework, 2018).

Las operaciones más importantes relacionadas con los datos en cualquier sistema REST y la especificación HTTP son cuatro: crear (POST), leer y consultar (GET), editar (PUT) y eliminar (DELETE) de aquí surge una alternativa a SOAP.

SOAP es una arquitectura dividida por niveles que se utilizaba para hacer un servicio, es más complejo de montar como de gestionar y solo trabajaba con XML.

Ahora bien, REST llega a solucionar esa complejidad que añadía SOAP, haciendo mucho más fácil el desarrollo de una API REST (Django REST framework, 2018).

1.2 Análisis de aplicaciones móviles que generan reportes

Las aplicaciones se generan en un entorno dinámico e incierto. En su mayoría se trata de aplicaciones pequeñas, no críticas, destinadas a un gran número de usuarios finales que son liberadas en versiones rápidas para poder satisfacer las demandas del mercado. En otros casos las aplicaciones son de mayor tamaño, a veces para brindar movilidad a una parte de la funcionalidad de un sistema más grande, mientras que otras veces son el único punto de interacción del sistema (Granada, 2014).

Raken

Es la aplicación para el manejo en el campo de la construcción. Es la más rápida y sencilla para generar reportes diarios, tarjetas de tiempo, y fotos, ahorrándole tiempo y dinero en el proceso.

Los usuarios agregan sus datos en reportes personalizados creando, asignando, y compartiendo un reporte en formato PDF en tiempo real. Además de notificaciones en tiempo real y estadísticas que mejoran el flujo de información desde la obra a la oficina. Los registros diarios de subcontratistas están compilados automáticamente en un grupo, proporcionando a los contratistas generales un solo registro de los eventos del día.

Evernote

Es una aplicación móvil para la gestión de una agenda de trabajo. Se presenta como un espacio para “capturar todas las experiencias” y acceder a ellas desde cualquier lugar, ya que todas las anotaciones se actualizarán al instante en todos los dispositivos en los que se tenga instalados la aplicación. Se puede anotar todas las ideas y tareas pendientes a través de texto. Lo más interesante como herramienta es:

- Está disponible para dispositivos iOS y Android.
- Puede sincronizar todo con varios dispositivos (con los que se puede trabajar offline)

- Tiene una interfaz muy agradable y fácil de usar.
- Dispone de aplicación de escritorio, por lo que permite vincular todos tus dispositivos.
- Es una herramienta muy versátil.

Después de efectuado el análisis de las aplicaciones que generan reportes anteriormente mencionados, se decide seleccionar Evernote para aplicar su diseño de las interfaces de usuarios por ser sencillas y fáciles de usar. Además, de las sincronizaciones con las aplicaciones de escritorio.

1.3 Tecnologías y herramientas

1.3.1 Sistema operativo Android

Un sistema operativo móvil o SO móvil es un conjunto de programas de bajo nivel que permiten la abstracción de las peculiaridades del hardware específico del teléfono móvil y provee servicios a las aplicaciones móviles, que se ejecutan sobre él. Los sistemas operativos móviles son simples y están orientados a la conectividad inalámbrica, los formatos multimedia para móviles y las diferentes maneras de introducir información en ellos (Cruz, 2017).

Android es un sistema operativo de código abierto para dispositivos móviles, se programa principalmente en Java, y su núcleo está basado en Linux para el manejo de memoria, procesos y hardware. Tanto el sistema operativo, como la plataforma de desarrollo están liberados bajo la licencia de Apache. Esta licencia permite a los fabricantes añadir sus propias extensiones propietarias, sin tener que ponerlas en manos de la comunidad de software libre. Al ser de código abierto, Android hace posible: una comunidad de desarrollo, gracias a sus completas APIs y documentación ofrecida y desarrollo desde cualquier plataforma (Linux, Mac, Windows, etc). Se trata de un rama, de manera que las mejoras introducidas no se incorporan en el desarrollo del núcleo de GNU/Linux contiene (Granada, 2014):

- Bibliotecas de código abierto para el desarrollo de aplicaciones, incluyendo SQLite, WebKit, OpenGL y manejador de medios.
- Entorno de ejecución para las aplicaciones Android.
- La máquina virtual Dalvik y las bibliotecas específicas dan a las aplicaciones funcionalidades específicas de Android.
- Un marco de trabajo pone a disposición de las aplicaciones los servicios del sistema como el manejador de ventanas, de localización, proveedores de contenidos, sensores y telefonía. SDK

(kit de desarrollo de software) que incluye herramientas, plug-in para Eclipse, emulador, ejemplos y documentación.

- Interfaz de usuario útil para pantallas táctiles y otros tipos de dispositivos de entrada, como, por ejemplo, teclado y trackball.
- Aplicaciones preinstaladas que hacen que el sistema operativo sea útil para el usuario desde el primer momento.

El SKD de Android incluye numerosas y completas API's para facilitar el desarrollo. Algunas de las características más relevantes son: licencias, distribución y desarrollo gratuitos. Acceso al hardware de WiFi, GPS, Bluetooth y telefonía, permitiendo tanto realizar como recibir llamadas y SMS. Control completo de multimedia, incluyendo la cámara y el micrófono. Ademas de APIs para los sensores: acelerómetros y brújula. Mensajes entre procesos (IPC). Almacenes de datos compartidos, SQLite, acceso a SD Card. Aplicaciones y procesos en segundo plano. Widgets para la pantalla de inicio (escritorio). Integración de los resultados de búsqueda de la aplicación con los del sistema. Uso de mapas y sus controles desde las aplicaciones. Y aceleración gráfica por hardware, incluyendo OpenGL ES 2.0 para los 3D (Granada, 2014).

Muchas de las características antes expuestas ya están, de una manera o de otra, para los SDK de otras plataformas de desarrollo móvil. Las que diferencian a Android del resto son: controles de Google Maps en las aplicaciones, procesos y servicios en segundo plano. Proveedores de contenidos compartidos y comunicación entre procesos. No diferencia entre aplicaciones nativas y de terceros, todas se crean igual, con el mismo aspecto, y con las mismas posibilidades de usar el harware y las APIs. Widgets de escritorio (Granada, 2014).

1.3.2 Kotlin

Kotlin es un lenguaje de programación desarrollado por Jetbrains. Se ejecuta principalmente en la máquina virtual de Java (JVM), pero también brinda la capacidad de ejecutar JavaScript, lo que permite su uso para navegadores y desarrollo de lado del servidor. Es perfectamente adaptable para el desarrollo de aplicaciones de Android, ya que brinda todas las ventajas de un lenguaje moderno a la plataforma de Android sin introducir nuevas restricciones (Eldhuset).

A pesar de las desventajas de una compilación inicial lenta. En la presente investigacion se decide utilizar el lenguaje de programación Kotlin pues este admite una compilación incremental eficiente, por lo que, si bien hay una sobrecarga adicional para las compilaciones limpias, las compilaciones incrementales suelen ser tan rápidas o más rápidas que con Java. Es totalmente compatible con JDK 6, lo que garantiza que las aplicaciones de Kotlin puedan ejecutarse en dispositivos Android

antiguos sin problemas. Sus herramientas son totalmente compatibles con Android Studio y con el sistema de compilación de Android. Ademas es 100% interoperable con Java, lo que permite utilizar todas las bibliotecas de Android existentes en una aplicación Kotlin.

1.3.3 Gradle

Gradle es una herramienta de automatización de compilación de código abierto centrada en la flexibilidad y el rendimiento. Los scripts de compilación de Gradle se escriben utilizando un Groovy o Kotlin DSL.

Se decide utilizar esta herramienta pues es modelada de una manera altamente personalizable y extensible.Ademas completa las tareas rápidamente, reutilizando los resultados de ejecuciones anteriores, procesando solo las entradas que han cambiado y ejecutando tareas en paralelo.Y destacar que es una herramienta de compilación oficial para Android potente y es compatible con muchos lenguajes y tecnologías populares (Gradle Inc., 2018).

1.3.4 XML

XML (Extensible Markup Language o Lenguaje de Marcado Extensible), se utiliza para representar información estructurada en la web, de modo que esta información pueda ser almacenada, transmitida, procesada, visualizada e impresa, por muy diversos tipos de aplicaciones y dispositivos. Se clasifica como un lenguaje extensible porque permite a sus usuarios definir sus propios elementos. Su objetivo principal es ayudar a los sistemas de información a compartir datos estructurados, particularmente a través de Internet, y se usa tanto para codificar documentos como para serializar datos (XML| Object Management Group, 2018).

Se decidió seleccionar este lenguaje especificamente pues permite simplificar, compartir e intercambiar datos. Además, puede simplificar también tanto el transporte de la información como los cambios de plataforma.

1.3.5 SQLite

SQLite es un gestor de bases de datos relacional y de código abierto, que cumple con los estándares, y además es extremadamente ligero. Se decide seleccionar SQLite pues este guarda toda la base de datos en un único fichero. Es útil en aplicaciones pequeñas para que no requieran la instalación adicional de un gestor de bases de datos, así como para dispositivos embebidos con recursos limitados. Además, de que Android incluye soporte a SQLite. Otras de las razones por la cual se seleccionó este gestor de base de datos es que separa el código que accede a la base de datos del resto del código extrayéndolo mediante un patrón adaptador que permita abrir la base de

datos, leer, escribir, borrar, y otras operaciones que el programa pueda requerir. Así, accedemos a métodos de este adaptador, y no se tiene que introducir código SQL en el resto del programa, haciendo además que el mantenimiento de las aplicaciones sea más sencillo.

Para ilustrar el uso de SQLite en Android se representa una clase adaptador de ejemplo. En esta clase (DataHelper) se abre la base de datos de una manera estándar: mediante un patrón SQLiteOpenHelper. Esta clase abstracta obliga a implementar su propio Helper de apertura de la base de datos, de una manera estándar. Básicamente sirve para que el código esté obligado a saber qué hacer en caso de que la base de datos no exista (normalmente crearla), y cómo portar la base de datos en caso de detectarse un cambio de versión. La versión de la base de datos se indica (About SQLite, 2018).

1.3.6 DroidText 0.4.jar librería para generar PDF

Actualmente existen diversas librerías que permiten generar documentos en formato pdf. DroidText 0.4.jar es una de ellas. La cual, permite crear un documento con formato pdf desde la parte del cliente Android.

La finalidad del documento es que se muestre en un archivo los reportes con los estándares de los modelos establecidos en la Instrucción No.1 del Presidente de los Consejos de Estado y de Ministros. El documento se guarda en una ruta local en el dispositivo del usuario, no en la parte del servidor.

1.3.7 Herramienta CASE

La herramienta CASE (Computer Aided Software Engineering por sus siglas en inglés) es una herramienta del software que automatiza una parte del ciclo de desarrollo de software. Constituyen diversas aplicaciones o programas informáticos destinados a aumentar el balance en el desarrollo de software reduciendo el costo de las mismas en términos de tiempo y de dinero. Estas herramientas sirven de ayuda en todos los aspectos del ciclo de vida de desarrollo del software en tareas como el diseño de proyectos, cálculo de costos, implementación de parte del código automáticamente con el diseño dado, compilación automática, documentación o detección de errores entre otras (Agile,Disciplined, 2018).

Visual Paradigm v8.0

Constituye una herramienta CASE profesional, que utiliza Lenguaje Unificado de Modelado (UML en sus siglas en inglés) y soporta el ciclo de vida completo del desarrollo de software, además de tener la ventaja de ser multiplataforma. La UCI tiene licencia para su uso y cumple con las políticas

de migración a software libre en Cuba (Gaines B. G., 2017).

Se decide seleccionar esta herramienta de modelado pues independientemente de la facilidad de uso que posee, la misma soporta todos los diagramas de entidad-relación. Puede generar Java, C#, C++, PHP y lenguaje de definición de datos (DDL) para todas las bases de datos populares. Permite diseñar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. El mismo proporciona abundantes tutoriales del Lenguaje de Modelado, demostraciones interactivas de UML y proyectos UML. Y además se integra con herramientas Java, como son: Eclipse/IBM, NetBeansIDE, entre otras.

Lenguaje de modelado UML v2.0

UML se usa para especificar, visualizar, construir y documentar artefactos de un sistema de software, no define un proceso de desarrollo específico, tan solo se trata de una notación. Se utiliza para detallar los artefactos en el sistema y definir un sistema de software.

El análisis de este lenguaje se realiza debido a que UML es la notación utilizada por la herramienta CASE que se emplea para la creación de los componentes. Además, es el lenguaje estándar de modelado para software que emplea la metodología AUP, la cual rige el proceso de desarrollo de software de la solución (Pressman, 2010).

1.4 Metodologías usadas actualmente para el desarrollo de aplicaciones móviles

Tabla 1 Comparación de las metodologías usadas actualmente para el desarrollo de aplicaciones móviles

Área	AUP-UCI	Mobile-D	HMD	Mobile Development Process Spiral
Desarrollo	Muchos colaborativos y están orientados a planificaciones con una mezcla de habilidades con experticia en proyectos similares.	Pocos en número y colaborativos.	Muchos colaborativos y están orientados a planificaciones con una mezcla de habilidades con experticia en proyectos similares.	Pocos en número y colaborativos.

Requerimiento	Conocibles tempranamente y sujeto a cambios.	Emergentes y con rápidos cambios	Conocibles tempranamente y sujeto a cambios.	Emergentes y con rápidos cambios
Refactorización	Económicamente costosa	Económicamente costosa por requerir gestión de cambio.	Medianamente costosa por experiencia en proyectos servidores que apoyan a la aplicación móvil.	Económicamente costosa por ser un modelo de reducción de riesgos
Tamaño	Producto Grande y equipo medianamente grande	Productos y equipos pequeños	Productos y equipos medianamente grandes	Productos y equipos grandes
Etapas con las que cuenta la metodología	Inicio Ejecución Cierre	Análisis Diseño Desarrollo Pruebas de funcionamiento	Análisis Diseño Aplicación del enfoque evolutivo Desarrollo del método Hibrido ágil para software Aplicación del método a un caso real	Determinación de requisitos Diseño Prueba
Tiempos de desarrollo	Tiempo mínimo de 10 semanas y un máximo de 16 de acuerdo a la complejidad del proceso.	Tiempo estimado en 8 y 12 semanas de acuerdo a la complejidad de los proceso.	Tiempo mínimo de 10 semanas y un máximo de 14 de acuerdo al número de participantes, ya que está metodología conforma tres tipos distintos de equipos.	Tiempo estimado en 8 y 10 semanas de acuerdo a la complejidad de los proceso.
Documentación	Para aplicaciones	Al tratarse de	N/A (para	Al tratarse de

(producto final)	móviles, se orienta a aplicaciones de escritorio y web	aplicativos móviles no existe manuales.	aplicaciones móviles, se orienta a aplicaciones de escritorio y web	aplicativos móviles no existe manuales.
Tipos de proyectos	Aplicativos de desarrollo tradicional o conocidos como de escritorio Desarrollo de aplicativos web Desarrollo de aplicativos móviles Desarrollo hibrido de aplicativos para solucionar distintos problemas tanto en tradicional como en web	Desarrollo móvil en distintas plataformas sin considerar el apoyo de aplicativos servidor.	Aplicativos de desarrollo tradicional o conocidos como de escritorio Desarrollo de aplicativos web Desarrollo de aplicativos móviles Desarrollo hibrido de aplicativos para solucionar distintos problemas tanto en tradicional como en web	Desarrollo móvil en distintas plataformas sin considerar el apoyo de aplicativos servidor.
Complejidad de desarrollo	El objetivo de la metodología es mantener un desarrollo sostenido, rápido y constante del producto de software, teniendo en consideración la mayoría de los aspectos que influyen en el manifiesto ágil, estos son también los aspectos a considerar al crear el enfoque.	Esté método alcanza casi el máximo de la puntuación en la complejidad del desarrollo de aplicativos móviles, debido a que se ha creado tomando en consideración la mayoría de los aspectos que influyen en el manifiesto ágil, estos son también los aspectos a considerar al crear	El objetivo de la metodología es mantener un desarrollo sostenido, rápido y constante del producto de software. Para lograr este objetivo, el equipo de desarrollo requiere de la asistencia del equipo de control de cambio y del equipo de comprobación en las tareas y de orientación del software al equipo de	Este método prioriza la participación del usuario en todos los procesos del ciclo de vida de diseño, con el fin de garantizar un diseño centrado en el usuario incorporando procesos de evaluación de la usabilidad.

		el enfoque.	comprobación	

Luego de apreciar las caracteristicas de todas las metodologías que se utilizan actualmente para el desarrollo de aplicaciones móviles, se decide utilizar la metodología de desarrollo para la actividad productiva de la UCI, para el desarrollo de la aplicación SIPACDroid, debido a que esta es la que más se ajusta a las necesidades del proyecto por proveer mejores ventajas al mismo. La cual, se ajusta al enfoque de la aplicación móvil con la variante Descripción de Requisitos por Procesos (DPR) para encapsular requisitos de software, teniendo en cuenta la cantidad de procesos complejos e interrelacionados entre sí definidos, debido a la planificación de las actividades de las instituciones en Cuba y los estándares que se deben seguir los reportes de los documentos de la Instrucción No.1 del Presidente de los Consejos de Estado y de Ministros. Independientes de las personas que los manejan y necesitados de objetividad, solidez, y continuidad. Teniendo que representar una gran cantidad de niveles de detalles y la relaciones se decide utilizar el escenario, en el cual se modela el negocio mediante DPN y MC de manera que permita encapsular los requisitos del negocio como DRP.

Se tiene conocimientos previos de los productos de trabajos que se generan durante el desarrollo y documentación de la aplicación de la metodología en las otras tesis desarrolladas sobre el tema. Se van a referenciar elementos del análisis de las metodologías descritas, claves durante la disciplina de análisis, implementación y pruebas, las cuales se centran en ciclos de desarrollos cortos con procesos iterativos e incrementales.

1.4.1 Metodología de desarrollo para la actividad productiva de la UCI

La metodología de desarrollo a emplearse en los proyectos productivos de la Universidad de las Ciencias Informáticas (UCI) es Variación AUP-UCI. Dicha definición se basa en una variación de la metodología “Proceso Unificado Ágil (AUP por sus siglas en inglés) en unión con el modelo CMMI-DEV v1.3 que significa Integración de Modelos de Madurez de las Capacidades o Capacibility Maturity Model Integration (Sánchez, 2015).

De las cuatro fases que propone AUP (Inicio, Elaboración, Construcción, Transición) esta variación mantiene la fase de Inicio, pero modifica su objetivo, y las restantes tres fases de AUP quedan unificadas en una sola fase denominada Ejecución, esta metodología además agrega una fase a las que denomina Cierre. Para una mayor comprensión y explicación de lo anterior se muestra a continuación la Tabla 2 con los objetivos específicos de cada fase (Sánchez, 2015):

Tabla 2 Descripción de las Fases Variación AUP-UCI.

Fases AUP	Fases Variación AUP – UCI	Objetivos de las fases (Variación AUP - UCI)
Inicio	Inicio	Durante el inicio del proyecto se llevan a cabo las actividades relacionadas con la planeación del proyecto. En esta fase se realiza un estudio inicial de la organización cliente que permite obtener información fundamental acerca del alcance del proyecto, realizar estimaciones de tiempo, esfuerzo y costo y decidir si se ejecuta o no el proyecto.
Elaboración	Ejecución	En esta fase se ejecutan las actividades requeridas para desarrollar el software, incluyendo el ajuste de los planes del proyecto considerando los requisitos y la arquitectura. Durante el desarrollo se modela el negocio, obtienen los requisitos, se elaboran la arquitectura y el diseño, se implementa y se libera el producto.
Ejecución		
Transición		
	Cierre	En esta fase se analizan tanto los resultados del proyecto como su ejecución y se realizan las actividades formales de cierre del proyecto.

Al igual que AUP esta metodología propone 7 disciplinas, pero las agrupa de forma más atómica. Los flujos de trabajos: Modelado de negocio, Requisitos y Análisis y diseño en AUP se unen a la disciplina Modelo, en la variación para la UCI se consideran a cada uno de ellos disciplinas diferentes. La disciplina Implementación se mantiene y en el caso de Prueba se desagrega en tres disciplinas: Pruebas Internas, de Liberación y Aceptación. (Sánchez, 2015)

Escenario para la disciplina de requisito

Luego de definir la mmetodología de desarrollo para la actividad productiva de la UCI, a partir de que el Modelado de negocio propone tres variantes a utilizar en los proyectos: Casos de Uso del Negocio (CUN), Descripción de Proceso de Negocio (DPN) y Modelo Conceptual (MC) y existen tres formas de encapsular los requisitos: Casos de Uso del Sistema (CUS), Historias de usuario

(HU) y Descripción de requisitos por proceso (DRP), surgen cuatro escenarios para modelar el sistema en los proyectos.

Conclusiones parciales

El presente capítulo permitió realizar un análisis de los principales conceptos quedando definido los reportes como una forma de representar la información contenida en las bases de datos, que permitirá que se encuentre organizada y ajustada a los estándares de los documentos establecidos en la planificación. El análisis de las tecnologías existentes permitió definir las características comunes que pueden ser utilizadas para el desarrollo de la solución propuesta. Se describen las herramientas y lenguajes que permitirá la construcción de la aplicación en consecuencia con la arquitectura de Android y justificando la más idónea en este sentido. Se decide utilizar la metodología de desarrollo para la actividad productiva de la UCI para el desarrollo de la aplicación Reportes en SIPACDroid, debido a que esta es la que más se ajusta a las necesidades del proyecto, con la variante Descripción de Requisitos por Procesos (DPR) para encapsular requisitos de software, teniendo en cuenta la cantidad de procesos complejos e interrelacionados entre sí definidos.

CAPÍTULO 2: PROPUESTA DE SOLUCIÓN

Introducción

En el presente capítulo se realiza una descripción de la solución propuesta, a partir de las disciplinas requisitos, análisis, diseño e implementación definidos por la metodología y el escenario seleccionado. Se describen los procedimientos llevados a cabo para el desarrollo de la aplicación, haciendo uso de la metodología de desarrollo de software seleccionada. Se recopilan los requisitos funcionales y no funcionales que el sistema debe cumplir y se define el comportamiento del mismo, así como las restricciones del diseño, concebidas para la construcción de la aplicación.

2.1 Modelo conceptual

Un modelo conceptual es la representación de conceptos del mundo real, no de componentes de software. Permite aumentar la comprensión del problema y contribuir a esclarecer la terminología o nomenclatura del dominio. Modelo que comunica a los interesados cuáles son los términos importantes y cómo se relacionan entre sí (Pressman, 2010).

En el modelo conceptual que a continuación se muestra se representan los conceptos de actividad, organismo, categoría de la actividad, tipo de actividad, alarma, plan, objetivo y configuración de Reportes.

1. **El concepto de actividad** está representado por una operación o tarea, propia de una persona o entidad, destinadas para cumplir determinado(s) objetivo(s).
2. **El concepto de alarma** viene dado por una notificación o alerta que se genera u ocurre a una hora determinada donde se informa de un acontecimiento, evento o acción.
3. **El concepto de categoría de la actividad** es una clasificación para englobar las actividades que pertenecen a un dominio delimitado por la entidad.
4. **El concepto del tipo de actividad** es una descripción de la acción de la actividad.
5. **El concepto de plan** está representado por el conjunto de actividades que se realizan en un periodo delimitado que están asociados a una persona o la entidad general.
6. **El concepto de objetivo** está marcado o por los resultados, situaciones y/o estados que un organismo intenta alcanzar o a los que pretende llegar, en un período de tiempo determinado y a través del uso eficiente de recursos con los que dispone o prevé disponer.
7. **El concepto de organismo** se representa como estructuras organizativas creadas para lograr metas u objetivos por medio de la gestión de los recursos que posee.

8. **El concepto de configuración de reporte** está representado por datos constantes que se encuentran ubicados en la parte superior del documento generado.

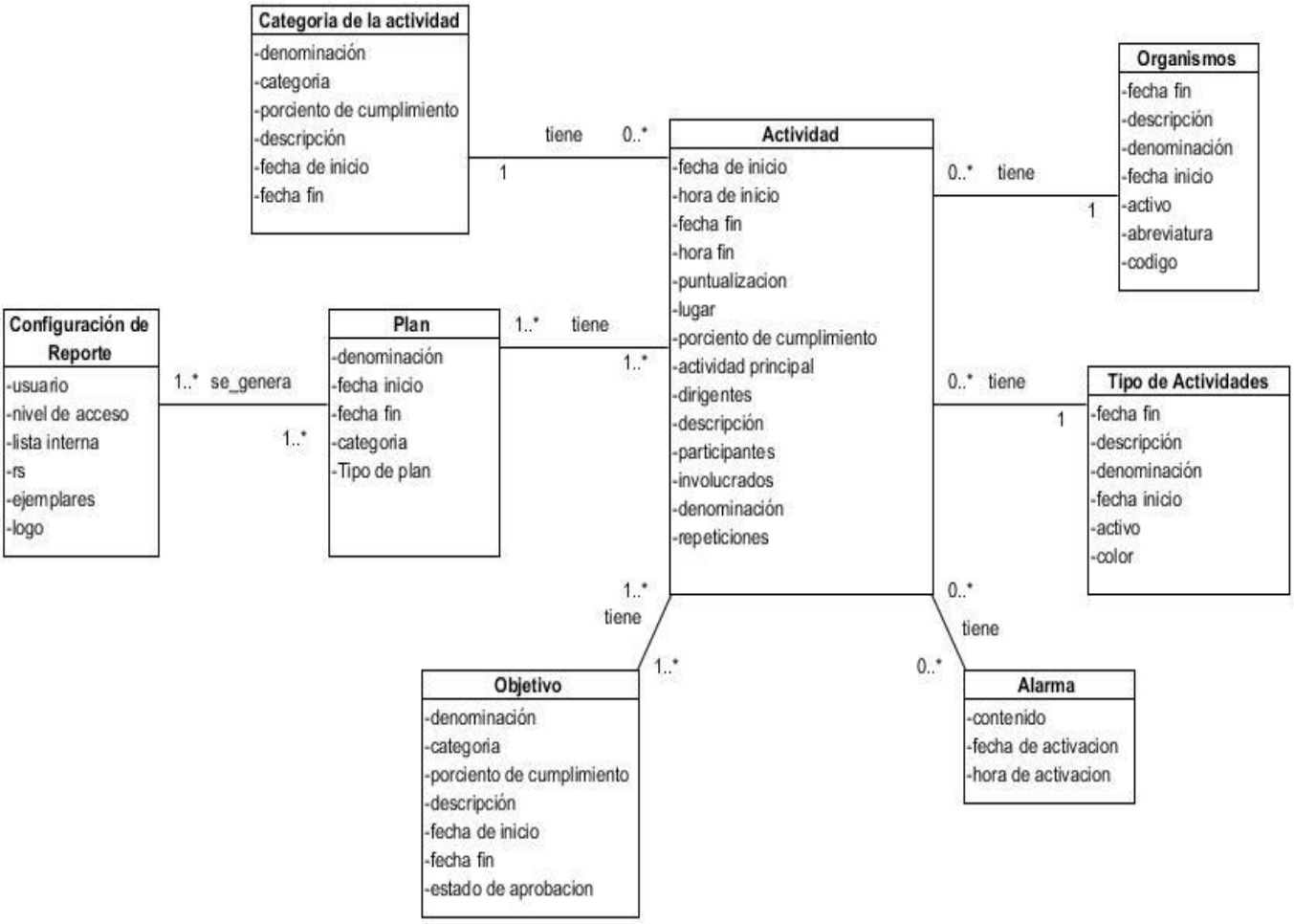


figura 1 Modelo conceptual de la aplicación SIPACDroid.

2.2 Requisitos de software

Un requisito de software es una declaración abstracta de alto nivel de un servicio que debe proporcionar el sistema o una restricción de este. Definición formal de una función del sistema (Sommerville, 2005).

Un requisito de software es una condición o capacidad que tiene que ser alcanzada o poseída por un sistema, producto o servicio para satisfacer un contrato, estándar, u otro documento impuesto formalmente (SWEBOK. CAPITULO 2, 2004).

2.2.1 Técnicas de recopilación de información e identificación de requisitos.

Para definir los requisitos que forman parte de la solución propuesta se utilizaron las siguientes técnicas de recopilación de la información:

- **Entrevistas**: requiere de una buena selección de los entrevistados para obtener información de calidad en el menor tiempo posible; es de gran utilidad para obtener información cualitativa como opiniones, o descripciones subjetivas de actividades. Esta técnica fue aplicada al cliente Orlando Martínez Obeso segundo jefe de la planificación de la secretaria de los Consejos de Estado y ministros. Además, a los analistas del SIPAC.
- **Análisis de la documentación**: requiere de varios tipos de documentación, como manuales y reportes, que propician información valiosa con respecto al SIPAC y a sus operaciones. Se analizaron las descripciones de requisitos por proceso, documentadas para la versión 3.0 de la aplicación web del SIPAC.
- **Modelo Conceptual**: requiere el análisis de un modelo conceptual para obtener información de las principales entidades y sus relaciones en el dominio del problema. Se analizó el artefacto Modelo conceptual del módulo de actividades (Figura 1).

Como resultado de aplicar las técnicas de capturas de requisitos antes descritas, se obtuvieron una serie de requisitos funcionales (RF) y no funcionales (RNF) que se listan en el siguiente subepígrafe.

2.2.2 Requisitos funcionales

Los requisitos funcionales son declaraciones de los servicios que debe proporcionar el sistema, de la manera que éste debe reaccionar a entradas particulares y de cómo se debe comportar en situaciones específicas. Capacidades o funciones que el sistema debe cumplir (SWEBOK. CAPITULO 2, 2004).

A continuación, la siguiente tabla muestra el resultado del levantamiento de los requisitos:

Tabla 3 Requisitos Funcionales de la Aplicación SIPACDroid.

No.	Nombre	Descripción	Prioridad	Complejidad
RF 1	Buscar planes de trabajo individual.	Permite realizar una búsqueda de los planes de trabajo individuales registrados en el sistema por los criterios de	Media	Media

		búsqueda: año, mes, usuario.		
RF2	Listar planes de trabajo individual	Permite listar los planes de trabajo individual disponibles que se corresponden con el filtro de búsqueda realizado. Mostrando la denominación del plan de trabajo, como un identificador único.	Media	Media
RF3	Generar reporte del plan de trabajo individual	Permite generar un reporte en formato .pdf del plan de trabajo seleccionado. El documento debe contener el nombre y cargo de la persona que aprueba el plan, el título del plan con la fecha del mismo, las tareas principales, mostrando en un calendario los días de la semana con las actividades correspondientes. El nombre y cargo del autor del plan.	Alta	Alta
RE4	Buscar resúmenes de cumplimiento del plan.	Permite realizar una búsqueda de los resúmenes de cumplimiento del plan registrados en el sistema por los criterios de búsqueda: año, mes, usuario.	Media	Media
RF5	Listar resúmenes de cumplimiento del plan.	Permite listar los planes disponibles que se corresponden con el filtro de búsqueda realizado. Mostrando la denominación del plan de trabajo, como un identificador único, con el objetivo de visualizar el resumen de cumplimiento del plan.	Media	Media
RF6	Generar reporte del resumen de cumplimiento del plan.	Permite generar un reporte en formato .pdf del resumen de cumplimiento del plan seleccionado. El documento debe contener el título del Resumen de cumplimiento del plan de trabajo con la fecha del mismo, un resumen cuantitativo	Alta	Alta

		con el total de tareas planificadas, de ellas: la cantidad que son cumplidas, incumplidas, modificadas y extra planes. En el caso, de las tereas incumplidas las causas de su incumplimiento y de las extra planes quién las originó y motivo. Además, de un breve resumen cualitativo.		
RF7	Buscar plan de trabajo mensual	Permite realizar una búsqueda de los planes de trabajo mensuales registrados en el sistema por los criterios de búsqueda: año, mes, usuario.	Media	Media
RF8	Listar planes de trabajo mensual	Permite listar los planes mensuales disponibles que se corresponden con el filtro de búsqueda realizado. Mostrando la denominación del plan de trabajo, como un identificador único.	Media	Media
RF9	Generar reporte del plan de trabajo mensual	Permite generar un reporte en formato .pdf del plan de trabajo seleccionado. El documento debe contener el nombre y cargo de la persona que aprueba el plan, el título del plan mensual con la fecha del mismo, las tareas principales, mostrando el número de la actividad, actividad hora y lugar, fecha, dirige, participantes y observaciones. El nombre y cargo del autor del plan.	Alta	Alta
RF10	Buscar plan de actividades generales por año.	Permite realizar una búsqueda de los planes de actividades generales por año registrados en el sistema por los criterios de búsqueda: año, categoría del plan, usuario.	Media	Media

RF11	Generar reporte de las actividades generales por año.	Permite generar un reporte en formato .pdf del plan de las actividades generales para el año. El documento debe contener el nombre y cargo de la persona que aprueba el plan de actividades, el título de la coordinación de las actividades para el año, con la fecha del mismo. Un calendario que muestra todos los meses del año con una trazabilidad de los días que tienen actividades programadas, deben estar marcadas con una (X).	Alta	Alta

2.2.3 Descripción del requisito Generar reporte del plan de trabajo individual

Tabla 4 Descripción del requisito Generar reportes del plan de trabajo individual.

Precondiciones	El usuario se ha autenticado en el sistema y cuenta con los permisos necesarios para ejecutar esta acción. El usuario selecciona en el menú lateral la opción Reporte. El usuario selecciona el Filtro Plan de Trabajo Individual.
Flujo de eventos	
Flujo básico Generar reporte del plan de trabajo individual	
1.	Se muestra la ventana Filtro de Plan de Trabajo Individual con los siguientes campos para que el usuario introduzca los datos (Ver validación 1): Año (campo de texto para filtrar la búsqueda de los planes de trabajo registrados de un año). Mes (campo de selección para filtrar la búsqueda de los planes de trabajo registrados de un mes). Usuario (campo de selección para filtrar la búsqueda de los planes de trabajo de los usuarios subordinados).
2.	El sistema muestra un panel con un listado de los planes de trabajo individual disponibles que se corresponden con el filtro de búsqueda realizado.
3.	El usuario selecciona un plan de trabajo individual de los listados.
4.	El usuario selecciona el botón **Generar reporte**.
5.	El sistema muestra un documento en formato pdf con los datos del plan de trabajo

	individual.
6.	Concluye el requisito.
Postcondiciones	
1.	Se generó un documento en formato pdf con los datos del plan de trabajo individual.
Flujos alternativos	
Flujo alternativo 2.a Campos vacíos	
1.	El sistema no lista ningún plan porque no existe un plan de trabajo individual que se corresponda con el criterio de búsqueda especificado".
2.	Volver al paso 1 del flujo básico.
Postcondiciones	
1.	N/A
2.	Volver al paso 1 del flujo básico.
Flujo alternativo 3.a El usuario no selecciona un plan de trabajo individual de los listados.	
1.	El sistema no activa el botón **Generar reporte.**
Postcondiciones	
1.	No se genera el documento en formato pdf con los datos del plan de trabajo individual.
Validaciones	
1.	Se validan los datos según lo establecido en el Modelo conceptual.

Concepto	Año	Visibles en la interfaz: Año
	Mes	Visibles en la interfaz: Mes
	Usuario	Visibles en la interfaz: Usuario
	Denominación del plan de trabajo individual	Visibles en la interfaz: Denominación
	Aprobado	Visibles en la interfaz: Aprobado
	Título	Visibles en la interfaz: Plan de Trabajo Individual para el mes de (fecha del criterio de búsqueda)

	Tareas Principales	Visibles en la interfaz: Tareas Principales
	Calendario	Visibles en la interfaz: Muestra los días de la semana y la actividad registrada en el día previsto.
	Autor	Visibles en la interfaz: Nombre de la persona que crea el plan de trabajo individual.
	Cargo	Visibles en la interfaz: Cargo de la persona que crea el plan de trabajo individual.
Prototipo de Interfaz de usuario		

Especialista

APROBADO:

Xipac Super LastName

PLAN DE TRABAJO INDIVIDUAL PARA EL MES DE MAYO DE 2019

Tareas Principales

Lunes	Martes	Miércoles	Jueves	Viernes	Sábado	Domingo
		1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31		

Especialista

Xipac Super LastName

2.2.3 Requisitos no funcionales

Los requisitos no funcionales son restricciones de los servicios o funciones ofrecidas por el sistema. Incluyen restricciones de tiempo, sobre el proceso de desarrollo y estándares. Normalmente se aplican a características o servicios individuales del sistema. Propiedades o cualidades que el producto debe tener (Pressman, 2010).

Requisitos organizacional: se derivan de políticas y procedimientos existentes en la organización del cliente y en la del desarrollador (Sommerville, 2005).

Requisitos de implementación.

RN1. La aplicación requiere un sistema operativo Android con versión 4.4.1 o superior.

RN2. Se requiere una PC servidora con el Sistema de Planificación de Actividades en su versión 3.0 instalado, el cual requiere el lenguaje de Programación Python 3, el marco de trabajo Django 1.11 y el gestor de bases de datos Postgres 9, para que la aplicación SIPACDroid pueda consumir los servicios REST de SIPAC.

RN3. La aplicación requiere la publicación de los servicios REST del módulo de Planificación, en el Sistema de Planificación de Actividades SIPAC.

RN4. La aplicación requiere que el protocolo de comunicación que se utilice es HTTPS.

RN5. La aplicación se implementará utilizando Android Studio como Entorno de Desarrollo Integrado (IDE), sistema de gestión de bases de datos: SQLite y lenguaje de programación: Kotlin y XML.

Requisitos del producto: especifican el comportamiento del producto (Sommerville, 2005).

Requisitos de usabilidad

RN 6. El diseño visual de la aplicación deberá ser adaptable o adaptativo, de manera que la apariencia de los desarrollos, se adapten a los dispositivos que se estén utilizando para visualizarlos. Se debe tener en cuenta, el reducido tamaño de las pantallas de los dispositivos móviles y su variedad (generalmente entre 4 y 10 pulgadas de diagonal.

RN7. La aplicación requiere un menú lateral izquierdo que permita acceder al módulo de Planificación.

Requisitos de seguridad

RN8: La aplicación garantizará la autenticidad cuando el usuario se autentique mediante el usuario y la contraseña.

RN10. La aplicación concederá acceso a cada usuario autenticado solo a las funcionalidades que le estén permitidas de acuerdo al rol especificado en el sistema.

RN11. Ante el fallo de una funcionalidad del sistema, en medio de una transacción el sistema mostrará un mensaje de fallo en la transacción.

RN12. En cada inicio de la aplicación es necesario que el usuario se autentique en el sistema.

2.3 Disciplina Análisis y diseño

En esta disciplina los requisitos pueden ser refinados y estructurados para conseguir una comprensión más precisa de estos, y una descripción que sea fácil de mantener y ayude a la estructuración del sistema (incluyendo la arquitectura). Además, en esta disciplina se modela el sistema y su forma para que soporte todos los requisitos, incluyendo los requisitos no funcionales (Sánchez, 2015).

2.3.1 Diagrama de clases del diseño.

Un diagrama de clases del diseño describe gráficamente las especificaciones de las clases de software y de las interfaces en una aplicación. Muestra las relaciones existentes entre las clases del sistema. Son utilizados durante el proceso de análisis y diseño de los sistemas, donde se crea el diseño conceptual de la información que se manejará en el sistema (Pressman, 2010).

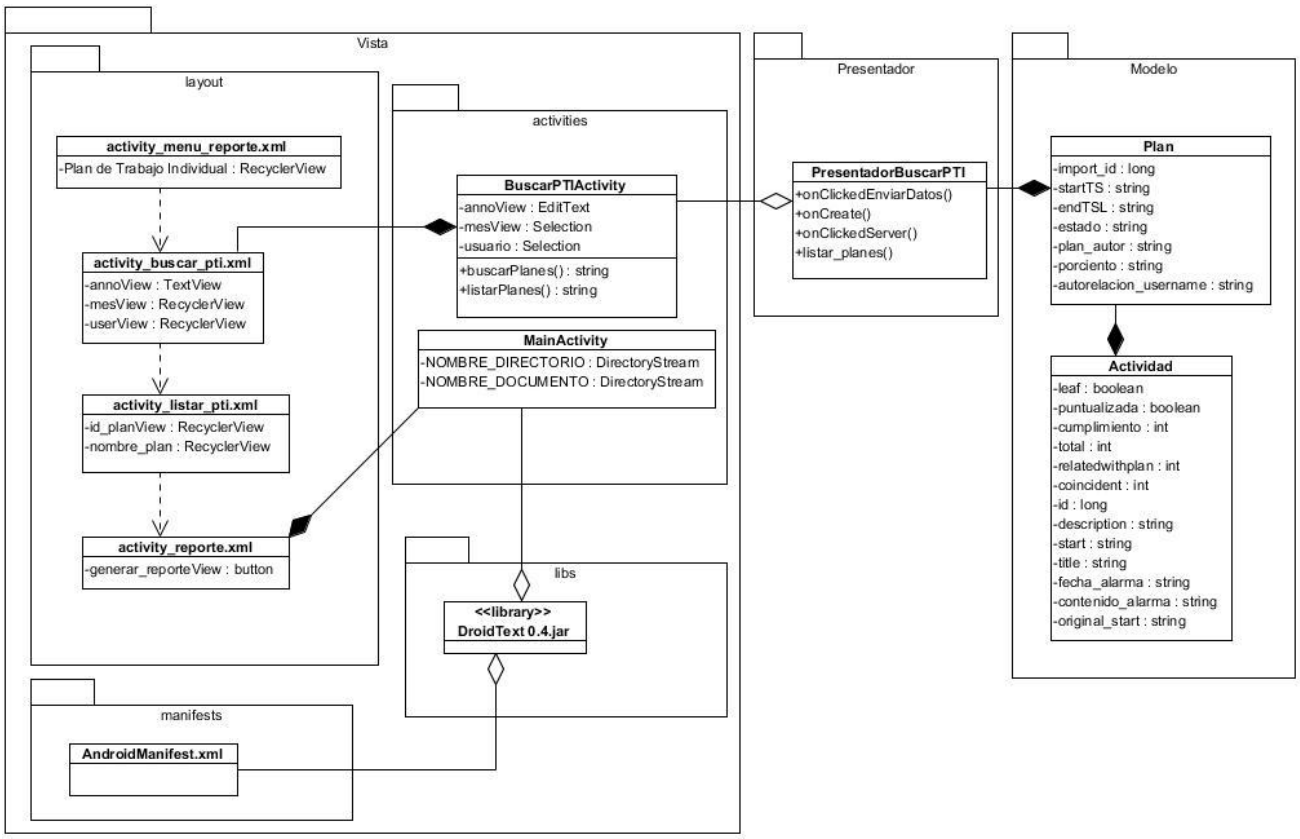


figura 2 Diagrama de clases del módulo reporte en Android.

El paquete Vista comprende las clases y elementos relacionados con la interfaz de usuario. La clase activity_menu_reporte es la encargada de decidir la forma en que se muestran los datos, la

cual implementa la interfaz BuscarPTIActivity que contiene las operaciones y construye el archivo activity_buscar_pti.xml que incluye y define los atributos de cada elemento visual de la vista.

El paquete Presentador es el encargado de poseer las clases que interactúan con el paquete Modelo y Vista. La clase BuscarPTIActivity que se encuentra en el paquete Vista construye la clase PresentadorBuscarPTI responsable de recuperar los datos del paquete modelo, devolverlos procesados a la vista, decidir qué ocurre cuando se interactúa con la vista.

El paquete Modelo contiene todas las clases relacionadas con los datos de la lógica del negocio. La clase PresentadorBuscarPTI del paquete Presentador crea la clase Plan del modelo. La clase Plan contiene todas las operaciones y atributos necesarios de los Planes de trabajo y la clase Actividad contiene las actividades asociadas a los planes por medio del identificador del plan. Todo este mecanismo se realiza por medio de la librería EasyRest que se utiliza para el consumo de servicios REST, hace uso de la clase Parametros y crea la clase DisableSSL que es la responsable de desactivar la verificación del certificado a la hora de realizar peticiones sobre el protocolo HTTPS (el servidor SIPAC utiliza este protocolo por defecto y hace uso de un certificado auto firmado no validado por ninguna entidad certificadora).

2.3.2 Patrones Generales de Software para Asignar Responsabilidades (GRASP).

Un patrón describe un problema que ocurre varias veces, así como el núcleo de la solución al inconveniente, de forma que puede utilizarse en ilimitadas ocasiones sin tener que hacer dos veces lo mismo (Larman, 2004).

En la solución del sistema se aplicaron principalmente los siguientes patrones de asignación de responsabilidades:

Controlador*:* es utilizado por todas las clases implicadas en la capa de presentación de la lógica del negocio. Poseen la responsabilidad de controlar el flujo de eventos mediante las actividades correspondientes. En la arquitectura de la aplicación se definió como clase controladora: PresentadorBuscarPTI.kt que es la encargada de controlar la lógica entre la vista y el modelo en el módulo de reporte de SIPACDroid.

Creador: ayuda a identificar quién debe ser el responsable de la creación (o instanciación) de nuevos objetos o clases. Este patrón se evidencia en la clase PresentadorMain.*kt* que crea objetos de la clase como WeeklyFragment.kt, para acceder a los elementos de esta clase.

Experto*:* indica que la clase que cuenta con la información necesaria para cumplir la responsabilidad es la responsable de manejar la información. En la aplicación SIPACDroid,

específicamente en la clase MainActivity.kt se evidencia este patrón ya que es la única que tiene la información necesaria para crear la vista activity_main.xml.

Alta Cohesión*:* plantea que la información que almacena una clase debe de ser coherente y debe estar (en la medida de lo posible) relacionada con la clase. El uso de este patrón queda evidenciado en la clase Event.kt la cual posee la estructura necesaria para guardar la información de forma coherente y de acuerdo a su responsabilidad.

Bajo acoplamiento*:* es utilizado en la creación de clases independientes para la lógica de los diferentes tipos de clases, actividades y entidades. Esto trae como ventaja que solo se realicen acciones sobre el tipo de entidad que se solicite o formulario correspondiente y no sobre todo el conjunto. Estas clases se encargan de la representación de los elementos reales de cada uno respectivamente y las actividades, de manejar los componentes visuales de cada funcionalidad.

2.4 Patrones de diseño de software

La arquitectura de software es un conjunto de patrones que proporcionan un marco de referencia necesario para guiar la construcción de un software, permitiendo a los programadores, analistas y todo el conjunto de desarrolladores del software compartir una misma línea de trabajo y cubrir todos los objetivos y restricciones de la aplicación. Es considerada el nivel más alto en el diseño de la arquitectura de un sistema, puesto que establecen la estructura, funcionamiento e interacción entre las partes del software (Gonzalez M. , 2014).

2.4.1 Patrón arquitectónico Modelo Vista Presentador

El patrón arquitectónico Modelo Vista Presentador (MVP por sus siglas en inglés) representado en la Figura 3, es un patrón de diseño que surge para realizar pruebas automáticas de la interfaz gráfica, para ello la idea es codificar la interfaz de usuario lo más simple posible. En su lugar, toda la lógica de la interfaz de usuario, se hace en una clase separada (que se conoce como Presentador) que no dependa en absoluto de los componentes de la interfaz gráfica y que, por tanto, es más fácil de realizar pruebas. Idealmente el patrón MVP permitiría conseguir que una misma lógica pudiera tener vistas totalmente diferentes e intercambiables.

Básicamente este patrón consiste en tres componentes:

- La vista. Compuesta de las ventanas y controles que forman la interfaz de usuario de la aplicación.
- El modelo. Que es donde se lleva a cabo toda la lógica de negocio.

- El presentador. Es una capa intermediaria entre la Vista (la interfaz gráfica de usuario) y el modelo de datos. Recupera los datos del modelo y se los devuelve a la vista formateada. Pero a diferencia del MVC típico, también decide qué ocurre cuando se interactúa con la vista (Olavide, 2015).

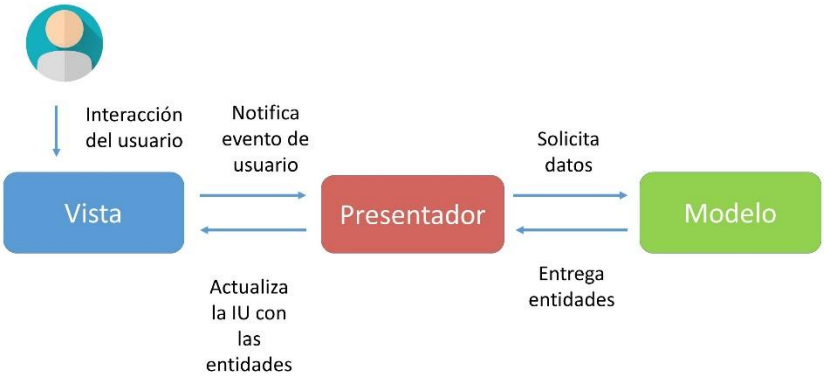


figura 3 Esquema de modelo vista controlador (MVC).

2.4.2 Generar los reportes

Para generar los reportes primeramente deben sincronizarse los datos guardados en el gestor de base de datos de la aplicación SIPACDroid con los del servidor para actualizar cualquier cambio en caso de estar conectado y luego a través de la librería DroidText 0.4.jar se generan en formato pdf.

2.4.3 Librería DroidText 0.4.jar

Se utiliza para generar un documento en formato pdf para ello:

1. **Se importa una librería en Android Studio**

La librería se incluye dentro de la carpeta lib, como se muestra en la Figura 4:

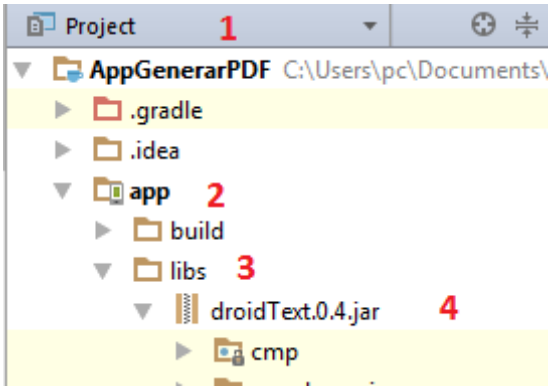


figura 4 Ruta de integración de la librería droidText0.4.

2. **Se incluye en el fichero build.gradle dentro del código de dependencias.**

```
dependencies {
        compile files('libs/droidText.0.4.jar')
}
```

4. **Se utiliza el código fuente dentro del fichero MainActivity**

La primera línea da el nombre de la carpeta donde se guardará el archivo pdf

La segunda línea da el nombre al archivo pdf

```
private final static String NOMBRE_DIRECTORIO = "MiPdf";
private final static String NOMBRE_DOCUMENTO = "prueba.pdf";
```

5. **Se incluyen los permisos dentro del fichero AndroidManifest**

```
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
<uses-permission   android:name="android.permission.WRITE_EXTERNAL_STORAGE"
/>
```

2.4.3 Servicios REST de SIPAC.

El marco de trabajo Django REST es un conjunto de herramientas potentes y flexible para crear API web, permite crear un API REST sobre Django de forma sencilla ofreciendo una alta gama de métodos y funciones para el manejo, definición y control de los recursos, y es una tecnología utilizada y reconocida por empresas tan importantes como Mozilla y Red Hat (Django REST framework, 2018).

El servidor SIPAC el cual es necesario para la sincronización de las actividades, planes, reportes y para a autenticación se hace uso del marco de trabajo de Python Django REST framework. El cual, permite la implementación de una API REST e interactúa con la base de datos de SIPAC. Para ello se identifican las tablas a interactuar y se registran como recursos en la API, creándose los servicios web que permiten la modificación de los datos. De esta forma, al iniciarse el servidor de SIPAC se inicia automáticamente la aplicación Django REST y se publican los servicios definidos, proporcionando elementos de seguridad para el consumo de los mismos. Para ello deberá implementar un consumidor de servicios, que utilizando los métodos GET, POST, PUT y DELETE definidos por el protocolo HTTPS realice las peticiones, que permitan listar, insertar, modificar y eliminar información en el servidor de SIPAC. Como resultado de estas peticiones recibirá una respuesta de la API REST en formato json, y con estos ambos sistemas podrán interoperar compartiendo información entre ellos, como se muestra en la Figura 5.

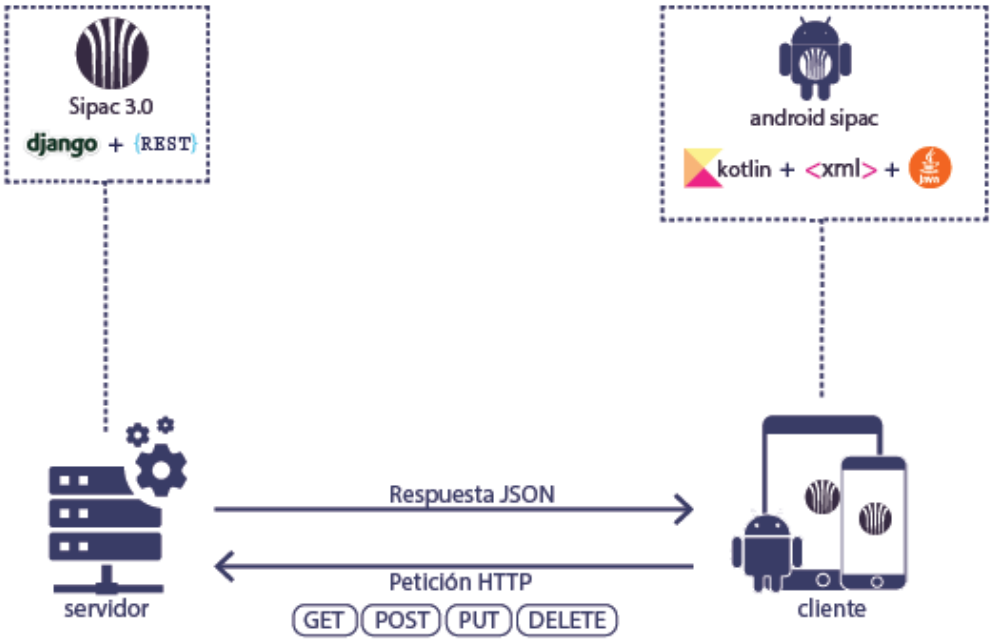


figura 5 Servicios REST de SIPAC.

Conclusiones parciales

Con la culminación del presente capítulo se elaboraron los artefactos correspondientes al escenario número tres de metodología AUP-UCI, creando así la documentación necesaria de la investigación que sirvió de guía a los desarrolladores para la implementación de la solución. Se definieron los requisitos necesarios para el correcto funcionamiento de la aplicación propuesta del módulo de Planificación de SIPAC. La utilización de los patrones arquitectónicos y de diseño seleccionados proporcionaron un correcto diseño de la aplicación propuesta del módulo de Planificación de SIPAC.

CAPÍTULO 3: IMPLEMENTACIÓN, PRUEBA Y VALIDACIÓN

Introducción del capítulo

En el presente capítulo se describen los estándares de codificación empleados durante la implementación de la solución propuesta. Se muestran los resultados de las pruebas efectuadas al software y se presenta la estrategia que se sigue para la validación de la investigación.

3.1 Modelo de bases de datos de la aplicación Android

El modelo entidad-relación representado en la Figura 6, define un modelo parcial del sistema identificando las entidades y las relaciones entre ellas. Es un modelo independiente del procesamiento que se requiere para generar o utilizar la información. Por lo tanto, es una herramienta ideal para el trabajo de modelado abstracto requerido dentro de la fase de requisitos del sistema. (Dick, et al., 2017).

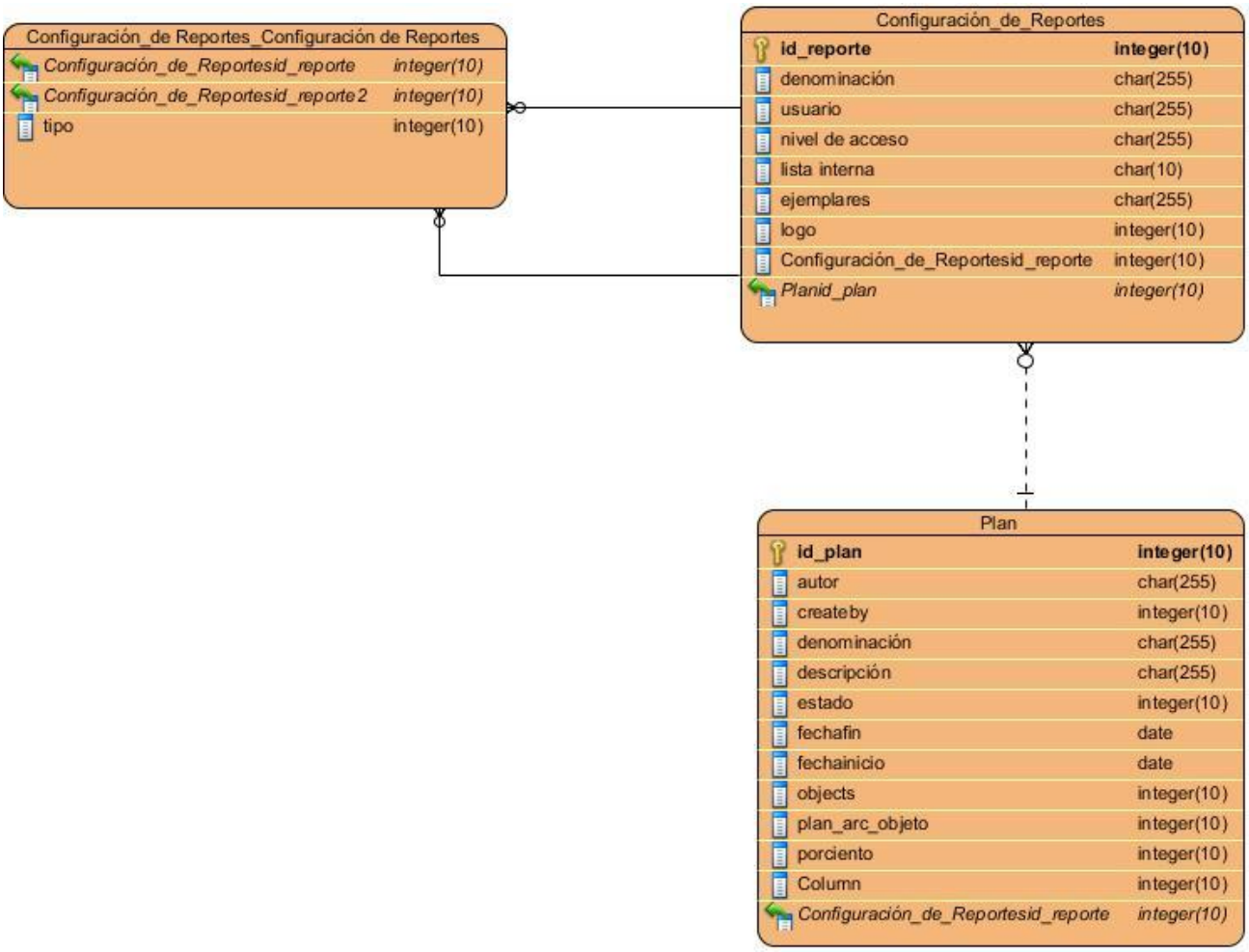


figura 6 Modelo entidad relación de la aplicación SIPACDroid.

El diagrama entidad relación de la solución descrita, está caracterizado por la entidad Configuración de Reportes que refleja la información asociada a una persona o entidad, la cual posee los atributos como: denominación, usuario, nivel de acceso, lista interna, ejemplares. La cual, se encuentra relacionada con la entidad Plan que contiene la información de los planes creados en el sistema.

3.2 Modelo de componentes de la aplicación

Un diagrama de componentes representa cómo un sistema de software es dividido en componentes y muestra las dependencias entre los mismo. Los componentes físicos incluyen archivos, módulos, ejecutables, paquetes, entre otros. Los diagramas de componentes prevalecen en el campo de la arquitectura de software, pero pueden ser usados para modelar y documentar cualquier arquitectura de sistema. (Sommerville, 2005).

El diagrama de componentes de la aplicación de Reportes para SIPACDroid incluye el componente principal SIPAC, que representa la aplicación SIPAC que incluye los componentes principales de Planificación, Calendario, Seguridad, Nomencladores, Modelos y las Vistas, y las relaciones entre los mismos. El componente SIPAC brinda servicios REST mediante el mecanismo que proporciona el componente Django REST framework al componente SIPACDroid que contiene los componentes Calendario, Autenticación y Reporte. Mediante este mecanismo la aplicación de Reporte de SIPACDroid obtiene la información contenida en la base de datos de SIPAC para generar los reportes de los planes creados en la aplicación SIPAC.

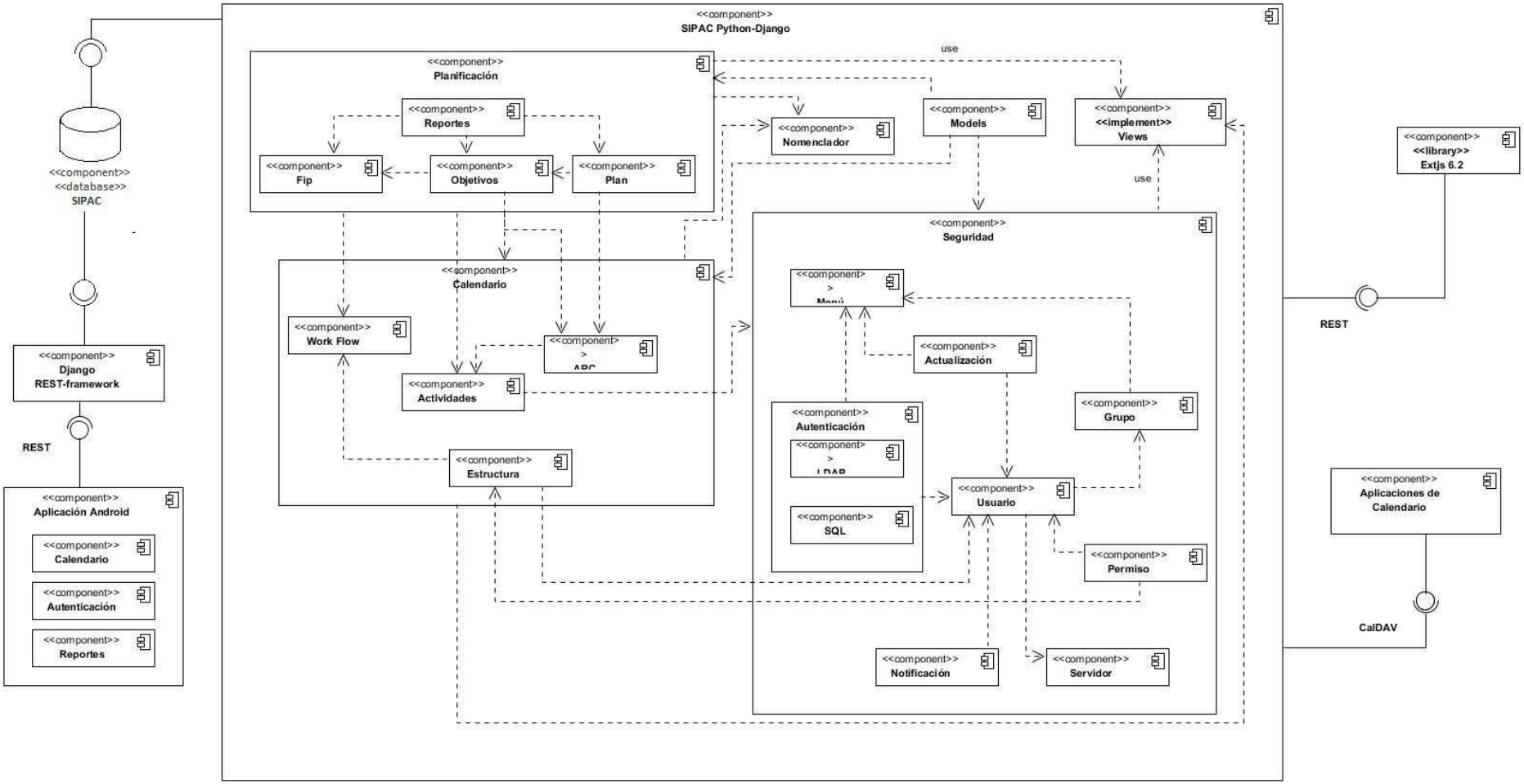


figura 7 Modelo de componente de SIPACDroid.

3.3 Estándares de codificación

Para un proyecto de desarrollo de software se definen estándares de codificación debido a que un estilo de programación homogéneo permite que todos sus participantes puedan entenderlo en menos tiempo. Esto trae como consecuencia que el código sea legible y mantenible (Eguiluz, 2014).

3.4.1 Convenciones para variables Kotlin

En Kotlin cada variable debe ser declarada y cualquier intento de usar una variable no declarada incurre en un error de sintaxis. La declaración también decide el tipo de datos que se puede almacenar en la variable, pero también es posible introducir el tipo específico de variable a declarar.

Las variables locales normalmente se declaran y se inicializan al mismo tiempo, en cuyo caso se infiere que el tipo de la variable es el tipo de la expresión con la que se inicializa:

var numero = 42

var mensaje: String = "Hello"

var edad: Int

edad = 5

Sin embargo, no es posible cambiar el tipo de una variable: edad solo puede hacer referencia a los valores enteros, y mensaje solo puede hacer referencia a los valores de String o de cadena de caracteres, por lo que numero = "Test" como mensaje= 3 son incorrectos y generarán errores de sintaxis.

También existen dos tipos de variables: las mutables (var) es decir aquellas cuyo valor es modificable y las de solo lectura o de asignación única (val) las cuales su valor solo puede ser asignado una sola vez (Eldhuset, Aasmund).

3.4.2 Estructura de paquetes

En proyectos de lenguaje mixto, los archivos de origen de Kotlin deben residir en la misma raíz de origen que los archivos de origen de Java y seguir la misma estructura de directorios (cada archivo debe almacenarse en el directorio correspondiente a cada declaración del paquete).

En proyectos que solo utilizan Kotlin, la estructura de directorio recomendada es seguir la estructura del paquete con el paquete raíz común omitido (por ejemplo, si todo el código del proyecto está en el paquete "org.example. kotlin" y sus subpaquetes, los archivos con la "org

.example. kotlin" el paquete debe colocarse directamente debajo de la raíz de origen, y los archivos en" org.example.kotlin.foo.bar "deben estar en el subdirectorio" foo/bar"de la raíz de origen) (JetBrains).

3.4.3 Nombres de archivos origen

Si un archivo en Kotlin contiene una sola clase (potencialmente con declaraciones de nivel superior relacionadas), su nombre debe ser el mismo que el nombre de la clase, con la extensión. kt adjunta. Si un archivo contiene varias clases, o solo declaraciones de nivel superior, se debe elegir un nombre que describa lo que contiene el archivo. En este caso se deben utilizar jorobas de camello con la primera letra en mayúscula (por ejemplo, MainActivity.kt).

El nombre del archivo debe describir lo que hace el código en el archivo. Por lo tanto, debe evitar el uso de palabras sin sentido como "Util" en los nombres de archivos (JetBrains).

3.4.4 Reglas de nombres

Los nombres de los paquetes siempre están en minúsculas y no usan guiones bajos (org.example.myproject). El uso de nombres de varias palabras generalmente no se recomienda, pero si fuese necesario, simplemente se concatenan o se utilizan jorobas de camello (org.example.myProject).

Los nombres de las clases y los objetos comienzan con una letra mayúscula y se deben utilizar jorobas de camello:

open class DeclarationProcessor { ... }

object EmptyDeclarationProcessor : DeclarationProcessor() { ... } (JetBrains)

3.4.5 Nombres de las funciones

Los nombres de las funciones, propiedades y variables locales comienzan con una letra minúscula y usan jorobas de camello y sin guiones bajos:

fun processDeclarations() { ... }

var declarationCount = ...

Excepción: las funciones de fábrica utilizadas para crear instancias de clases pueden tener el mismo nombre que la clase que se está creando:

abstract class Foo { ... }

class FooImpl : Foo { ... }

fun Foo(): Foo { return FooImpl(...) } (JetBrains)

3.4.6 Declaración de clases

Generalmente, los contenidos de una clase se ordenan de la siguiente forma:

- Declaraciones de propiedades y bloques de inicialización.
- Constructores secundarios.
- Declaraciones de métodos.
- Objeto comparación o acompañante.

No es recomendable ordenar las declaraciones de métodos alfabéticamente ni por visibilidad ni separar los métodos regulares de los métodos de extensión. En su lugar, es mejor unificarlas relacionadas para que sea más fácil seguir la lógica de lo que está sucediendo.

Al usar vocabulario XML de Android, puedes crear rápidamente diseños de interfaz de usuario y de los elementos de pantalla que contienen, con una serie de elementos anidados.

Cada archivo de diseño debe contener exactamente un elemento raíz, que debe ser un objeto *View* o *ViewGroup*. Una vez que se haya definido el elemento raíz, se pueden agregar widgets u objetos de diseño adicionales como elementos secundarios para crear gradualmente una jerarquía de vistas ase, se deben colocar al final, después de su objeto Companion (JetBrains).

3.4.7 Nomenclatura XML

que defina tu diseño.

Ejemplo elemento raíz:

```
<?xml version="1.0" encoding="utf-8"?>
<com.simplemobiletools.calendar.views.MyScrollView
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/content_main "
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools">

</com.simplemobiletools.calendar.views.MyScrollView>
```

Ejemplo de *widgets* u otros objetos de diseño:

```
<TextView
    android:layout_width="463dp"
    android:layout_height="wrap_content"
    android:layout_weight="3"
    android:text="Locación"
    android:textAppearance="@style/AppTheme3"
    android:textSize="@dimen/day_text_size" />
```

Después de declarar el diseño en XML, se guarda el archivo con la extensión `.xml` en el directorio *`res/layout/`* del proyecto de Android para que pueda compilarse correctamente.
Cuando se compila la aplicación, cada archivo de diseño XML se compila en un recurso `View`. Se debe cargar el recurso de diseño desde el código de la aplicación, en la implementación de callback *`Activity.onCreate()`*. Para hacerlo, llama a *`setContentView()`*, substituyendo la referencia del recurso de diseño en forma de: *`R.layout.`*layout_file_name.

Ejemplo:

```
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_event)
```

Cualquier objeto View puede tener un identificador con número entero asociado a él, para identificar de forma exclusiva, por lo que para poder manejar cualquier vista o widget se le debe asignar un id único.
Ejemplo: `android:id="@+id/start_ts "`

Luego, crear una instancia del objeto View y capturarlo desde el diseño (generalmente en el método `onCreate()`):
Ejemplo: **val** inicioText = findViewById<EditText>(R.id.*startts*) (Android Developers, 2018).

3.4 Modelo de despliegue de la aplicación

Un diagrama de despliegue muestra cómo se configuran las instancias de los componentes y los procesos para la ejecución en tiempo de las instancias de los nodos de proceso. Representa los procesos y componentes sobre los nodos de proceso, y la configuración de la red física entre los nodos (Larman, 2004).

El modelo de despliegue provee la base para la comprensión de la distribución física de un sistema a través de nodos, indicando de qué forma se sitúa el software en el hardware que lo contiene. A continuación, se presenta el modelo de despliegue de la aplicación Android:

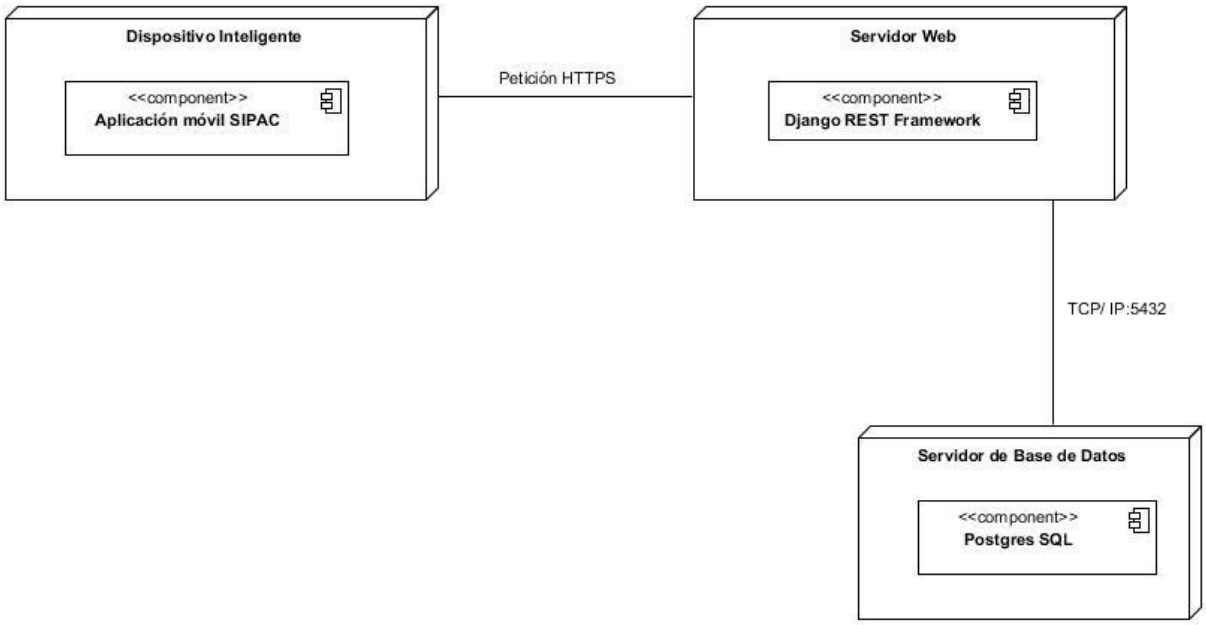


figura 8 Diagrama de despliegue.

3.3.1 Descripción de componentes

Nodo Dispositivo Inteligente: en este nodo (Figura 9) se instala la aplicación desde la cual se consumirán y se brindarán los servicios REST a partir de los recursos que posee la API Django REST Framework.

figura 9 Nodo Dispositivo Inteligente.

Nodo Servidor Web: en este nodo (Figura 10) se guardarán todas las direcciones de los recursos a manejar, mediante los servicios REST.

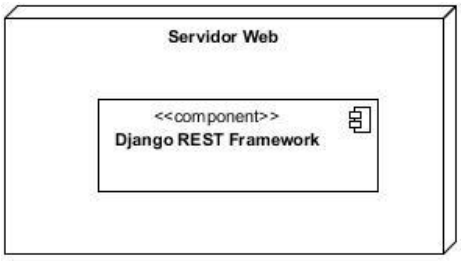


figura 10 Nodo Servidor Web.

Nodo Servidor de Base de Datos SIPAC: en este nodo (Figura 11) se almacenan todos los datos referentes al módulo de autenticación y actividades.

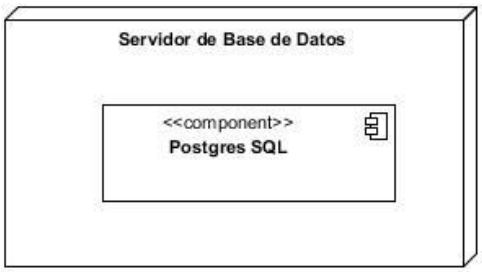


figura 11 Nodo Servidor de Base de Datos SIPAC.

3.5 Pruebas de software

Las pruebas de software son un elemento crítico para la garantía de la calidad de la aplicación y representan una revisión final de las especificaciones, del diseño y de la codificación. Las mismas son realizadas con el objetivo de detectar errores en el sistema, por lo que se llevan a cabo durante todo el ciclo de vida del producto. Estas tienen como objetivo, además de descubrir errores, medir el grado en que el software cumple con los requerimientos definidos(Pressman, 2010).

Según la norma ISO/IEC/IEEE 25010: 25010 para su correcta realización se debe tener en cuenta lo siguiente:

Verificación: Proceso de evaluación de un sistema o componente para determinar si un producto de una determinada fase de desarrollo satisface las condiciones impuestas al inicio de la fase.

Validación: Proceso de evaluación de un sistema o componente durante o al final del proceso de desarrollo para determinar cuándo se satisfacen los requerimientos especificados (ISO/IEC/IEEE24765: 2017- 2019).

Como resultado final de estas pruebas se puede obtener una determinada Predicción de Fiabilidad, o un cierto nivel de confianza en el software probado. En la siguiente Figura 12, se muestra el contexto en que se realiza la prueba de software:

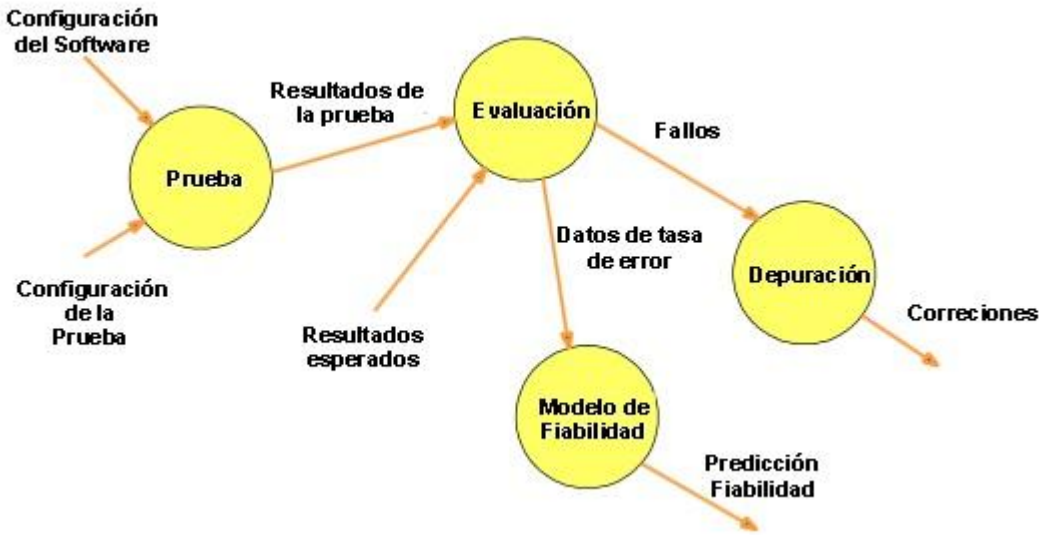


figura 12 Contexto de la prueba de software.

3.5.1 Pruebas unitarias automatizadas

3.5.2 Definición de los procedimientos de las pruebas unitarias

Especifica las operaciones a llevar a cabo durante el proceso de pruebas y se encarga a su vez de definirlas y planificarlas.

Pasos para ejecutar las pruebas unitarias en Android Studio 3.4.1

En Android Studio al crear un proyecto nuevo ya viene implementada la configuración por defecto para realizar pruebas unitarias, para esto es necesario realizar los siguientes pasos:

1. En el archivo build.gradle dentro del directorio app:
    1.1. Agregar las dependencias necesarias para ejecutar las pruebas utilizando el marco de desarrollo Junit 4:

```
testImplementation 'junit:junit:4.12'
androidTestImplementation 'com.android.support:support-annotations:28.0.0'
androidTestImplementation 'com.android.support.test:runner:1.0.2'
```

1.2. Definir el identificador de las pruebas a ejecutar:

```
testApplicationId "com.simplemobiletools.calendar.pro.test"
```

1.3. Especificar la clase encargada de ejecutar las pruebas:

```
testInstrumentationRunner "androidx.test.runner.testInstrumentationRunner"
```

1.4. Establecer los directorios donde se almacenarán los reportes y los resultados:

```
testOptions {
    reportDir = "$project.buildDir/results/report"
    resultsDir = "$project.buildDir/results"
}
```

2. En app dentro del directorio java debe existir un paquete con el mismo nombre que se especificó en el paso 1.4 (si no se encuentra es necesario crearlo) crear las clases para realizar las pruebas, el nombre de estas siempre debe contener Test al final.
3. Para programar las pruebas solo es necesario que los métodos a testear contengan la anotación @Test y que comiencen con el nombre test:

```
 class LoginActivityTest {
    @Test
    fun testGetSharedPreferences() {
        assertNull(null)
    }
}
```

4. Para ejecutar y compilar las pruebas solo es necesario seleccionar entre las configuraciones de ejecución la del test como se muestra en la Figura 19 y ejecutar la prueba presionando Run (▶) .

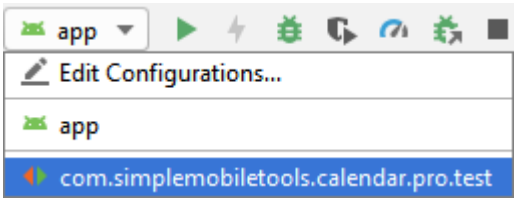


figura 13 Configuración a seleccionar para ejecutar Pruebas.

3.5.4 Pruebas funcionales

Las pruebas funcionales son pruebas diseñadas tomando como referencia las especificaciones funcionales de un componente o sistema (lo que se va a testear, el software o una parte de él). Se realizan para comprobar si el software cumple las funciones esperadas (Pressman, 2010).

3.5.4.1 Pruebas de caja negra

Las pruebas de caja negra hacen referencia a pruebas que se llevan a cabo sobre la interfaz del software sin tener en cuenta el código de la aplicación. Los casos de prueba pretenden demostrar que las funciones del software son operativas, que la entrada se acepta de forma adecuada y que se produce una salida correcta. Según (Pressman, 2010) estas pruebas tienen como propósito detectar: funciones incorrectas o ausentes, errores de interfaz, errores en de estructuras datos o en accesos a bases de datos externas, errores de rendimiento y errores de inicialización y de terminación.

Las pruebas de caja negra comprenden un conjunto de diferentes técnicas como, por ejemplo: Partición de Equivalencia, Análisis de Valores Límites y Grafos de Causa-Efecto. De estas técnicas se selecciona para ser utilizada la Partición de Equivalencia, la cual divide el campo de entrada en clases de datos que tienden a ejercitar determinadas funciones del software. Para la aplicación de la misma se realizaron once diseños de casos de prueba (DCP) uno por cada requisito funcional definido, los cuales se basan en una evaluación de las clases de equivalencia para una condición de entrada. Una clase de equivalencia representa un conjunto de estados válidos o inválidos para condiciones de entrada. Regularmente, una condición de entrada es un valor numérico específico, un rango de valores, un conjunto de valores relacionados o una condición lógica (Pressman, 2010).

Como resultado de la aplicación de las pruebas de caja negra fueron identificadas un total de 21 No Conformidades (NC), las cuales se fueron corrigiendo en tres iteraciones. La relación de NC por componente puede ser representada de la siguiente forma:

Técnica de prueba: Partición equivalente

Esta es una técnica de prueba de Caja Negra que divide el dominio de entrada de un programa en clases de datos de los que se pueden derivar casos de prueba. El diseño de estos casos de prueba se basa en la evaluación de las clases de equivalencia para una condición de entrada, así mismo, una condición de entrada es un valor numérico, un rango de valores, un conjunto de valores relacionados o una condición lógica. Una clase de equivalencia representa un conjunto de estados válidos y no válidos para las condiciones de entrada (Pressman, 2010).

Para aplicar esta técnica, se deben primeramente realizar el Diseños de casos prueba (DCP) con el objetivo de obtener un conjunto de pruebas que tengan la mayor probabilidad de descubrir los defectos del software.

Un DCP es, en ingeniería del software, un conjunto de condiciones o variables bajo las cuales un analista determinará si una aplicación o una característica de éstos es parcial o completamente satisfactoria (Pressman, 2010).

Fueron creados un total de 11 diseños de casos de prueba, los cuales encapsulan los 11 requisitos funcionales definidos y se realizaron un total de tres iteraciones de pruebas de caja negra. En la tabla 5 se muestra un ejemplo del diseño de caso de prueba: Generar reporte del plan de trabajo mensual. Obteniéndose los siguientes resultados:

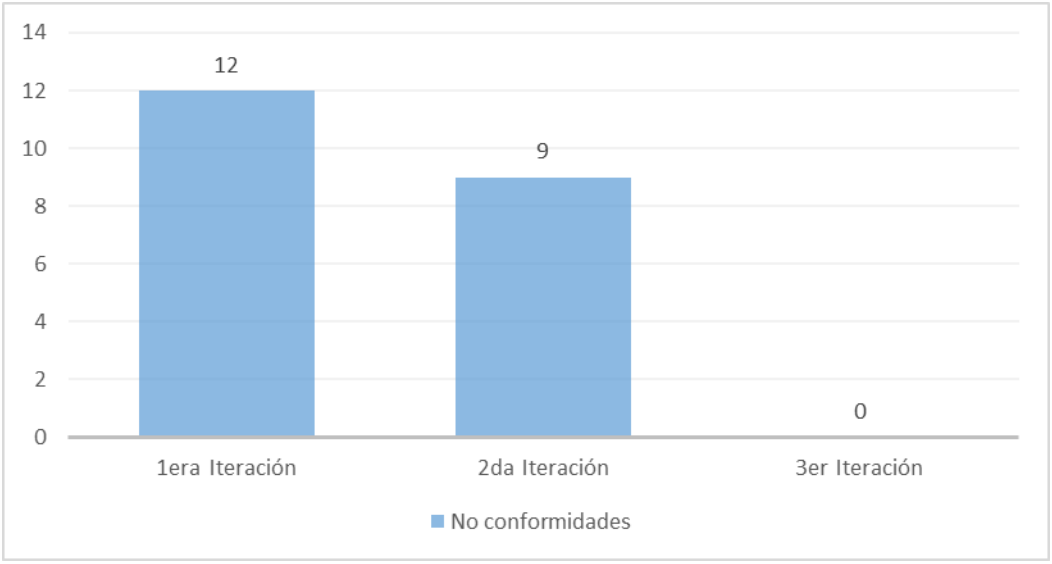


figura 14 Gráfico de no conformidades detectadas por iteración

En la primera iteración de pruebas ejecutadas se detectaron 12 No conformidades, las cuales, relacionadas principalmente con elementos de los filtros de búsquedas, errores ortográficos en la aplicación y en elementos que faltaban en los archivos que se generaban. Luego, de una segunda iteración de pruebas fueron corregidos nuevos errores introducidos para un total de 9 No conformidades y pasar a una tercera iteración sin NC conformidades detectadas. Obteniendo como resultado, una aplicación correctamente funcional, es decir, a datos correctos de entrada en las búsquedas datos correctos de salida.

Tabla 5 Diseño de Caso de Prueba Generar reporte del plan mensual

Escenario	Descripción	Año	Mes	Usuario	Respuesta del sistema	Flujo central
EC 1.1 Generar reporte seleccionando el plan.	En caso de que se desee generar el reporte de un plan de trabajo mensual, se filtra según los criterios de búsqueda Año, Mes, y Usuario.	V 2020	V Julio	NA	1.1 El sistema muestra la ventana con las opciones de Abrir y Guardar el archivo. 1.2 El usuario selecciona la opción Seleccione una carpeta. 1.3 El usuario presiona en el botón Aceptar.	1.1 El usuario da presiona el título Plan de Trabajo Mensual para expandir el panel y que el usuario pueda buscar el plan que desea. 1.2 El sistema despliega los campos Año, Mes y Usuario. 1.3 El usuario completa estos campos. 1.4 El sistema muestra al usuario la lista con los planes de trabajo individuales que coinciden con el criterio de búsqueda. 1.5 El usuario selecciona un plan de la lista. 1.6 El usuario presiona en el botón Generar reporte que se encuentra en la parte superior de la interfaz.
EC 1.4 Generar reporte sin seleccionar un plan.	En caso que se desee generar un reporte sin seleccionar previamente un plan de trabajo individual	NA	NA	NA	1.1 No se activa el botón generar reporte.	1.1 El usuario no selecciona un plan de los que se listan en la interfaz.
EC 1.5 Abortar la operación de generar reporte.	En caso que se desee abortar la operación de generar un reporte presionando en el botón Cancelar	NA	NA	NA	1.1 El sistema muestra la ventana con las opciones de Abrir y Guardar el archivo. 1.2 El usuario selecciona la opción Seleccione una carpeta. 1.3 El usuario presiona en el botón Cancelar.	1.1 El usuario da presiona el título Plan de Trabajo Mensual para expandir el panel y que el usuario pueda buscar el plan que desea. 1.2 El sistema despliega los campos Año, Mes y Usuario. 1.3 El usuario completa estos campos. 1.4 El sistema muestra al usuario la lista con los planes de trabajo individuales que coinciden con el criterio de búsqueda. 1.5 El usuario selecciona un plan de la lista. 1.6 El usuario presiona en el botón Generar reporte.

CAPÍTULO 3: IMPLEMENTACIÓN, PRUEBA Y VALIDACIÓN

3.6 Validación de la investigación

El uso de cuestionarios en la evaluación de la usabilidad permite obtener información sobre las opiniones, deseos y expectativas de los usuarios potenciales. Los cuestionarios tendrán que ser rellenados por los usuarios y enviarlos de vuelta. Los cuestionarios son útiles e informativos en todas las fases de diseño y desarrollo de la aplicación, pero requieren un número adecuado de usuarios de prueba para poder encontrar las preferencias subjetivas del usuario.

Esta técnica permite describir conductas pasadas, expectativas del usuario, actitudes y opiniones hacia el sistema. El tipo de cuestionarios está definido en función al tipo de preguntas, que puedan incluirse en él, siendo éstas:

1. Preguntas generales, usadas para establecer referencias del usuario y la localización de sujetos en la población: edad, sexo, ocupación, experiencia previa con ordenadores.

2. Preguntas abiertas/cerradas, usadas para permitir al usuario expresar su opinión con completa libertad y en sus propias palabras. Por otro lado, las cerradas restringen al usuario a seleccionar una de un conjunto de alternativas fijas, o una respuesta directa (si/no). Las preguntas cerradas requerirán establecer una escala de valoración que permitan resultados de gran precisión. Las instrucciones de uso deben ser explícitas para permitir conclusiones útiles. Dependiendo del nivel de medición deseado, las escalas pueden ser:

Escala nominal o de categoría, basadas en la selección del nivel subjetivo del atributo de un conjunto limitado de alternativas, marcadas con palabras que indican los diferentes grados de subjetividad: A: Excelente, B: Bueno, C: Justo.

Escala de valoración numérica discreta, conocida también como escala tipo Likert. Basada en el uso de números (1-5, 1-7, 1-9) que representan divisiones de escala (intervalos de igual magnitud de medida). Esta escala permite hacer conclusiones sobre el ordenamiento y las diferencias cuantitativas entre condiciones. Las respuestas obtenidas pueden ser analizadas rápidamente y permiten obtener recomendaciones para el diseño: Acuerdo o Desacuerdo.

Escalas de comparación, basada en la comparación entre dos tareas o condiciones de acuerdo a un atributo, realizada por los sujetos. Es similar al método de valoración conocido como “pares de comparación” en el cual un número de condiciones es comparado en todas las combinaciones posibles: Mucho mejor, mejor, ligeramente mejor, mucho peor.

Escala gráfica (o de valoración continua), consiste en una línea sin divisiones, representando puntos continuos o solo un punto medio entre ellos. Cada punto es definido por una etiqueta característica de definición o adjetivo.

1. Preguntas de multi-selección, usadas cuando se quiere ofrecer un conjunto cerrado de opciones en la que el usuario debe marcar su preferencia. El nivel de detalle es determinado por el investigador de acuerdo a sus intereses de investigación. Permite obtener información sobre la experiencia del usuario: Si/No; Verdadera/Falso., o diariamente, semanalmente, una vez al año.
2. Preguntas de escalamiento, se pide al usuario clasificar el orden de los ítems de una lista, forzando a la selección. Es útil para capturar preferencias del usuario: 1- Más preferido, 2- próximo).

Como parte de la estrategia de validación de la solución propuesta se ha realizado un diseño pre-experimental midiendo la variable dependiente usabilidad. Para la realización de un pre-experimento se definen diseños para pre-prueba y post-prueba. El diseño de pre-prueba permite obtener una referencia inicial y conocer el nivel de la variable dependiente antes de ser estimulada. El diseño post-prueba permite establecer una comparación con los datos obtenidos anteriormente, y el comportamiento de la variable dependiente luego de recibir el estímulo. (Guido Elmer, 2014) Para la aplicación del diseño experimental se utilizó la Lista de Chequeo de Usabilidad propuesta en el Departamento de Calidad de la Universidad de las Ciencias Informáticas como instrumento de medición adaptada a entornos móviles, la cual está compuesta por un conjunto de 48 preguntas enfocadas a medir indicadores de usabilidad. El resultado de aplicar la post-prueba muestra el siguiente comportamiento para la variable usabilidad, a partir del resultado obtenido por la lista de chequeo de usabilidad ajustada a tecnologias Android como me muestra en la tabla 6. En la Figura 15, se muestra el prociento que ha mejorado la usabilidad del sistema para un 96 % con respecto a una usabilidad inicial de 0 porciento, calculado a partir de un total de 48 preguntas con respuesta de Bien y dos con respuesta de Mal. Lo que queda evidenciado, que la aplicación para generar reporte en el sistema SIPACDroid cumple con la mejora de la usabilidad en un 96 %.

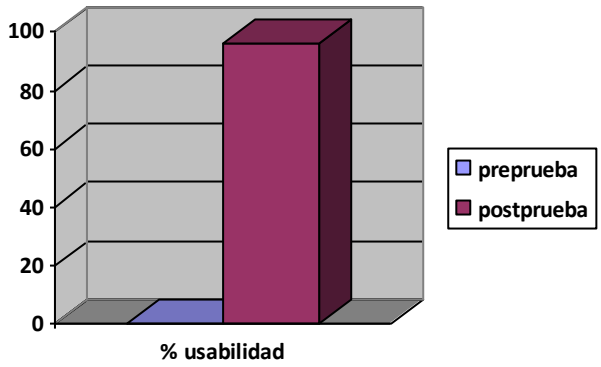


figura 15 Porciento de mejora de la usabilidad.

Tabla 6 Lista de chequeo de usabilidad ajustada a tecnologías Android.

No	Indicador a evaluar	Evaluación	No procede	Observación
1	¿La aplicación refleja la identidad de la empresa logos, compañía...)?	B		En las plantillas de los repotes se puede visualizar el logo de la entidad.
2	¿Cada pantalla empieza con un título que describe su contenido?	B		
3	¿Cuándo se selecciona un icono se diferencia de los no seleccionados?	B		
4	¿Los enlaces del menú se resaltan cuando se seleccionan?	B		
5	¿Los iconos que aparecen se identifican claramente con lo que representan?	B		
6	¿El menú de navegación aparece en un lugar destacado	B		

7	¿No utiliza más de siete opciones principales en el menú de navegación?	B		
8	¿Si la respuesta a una acción se retrasa, aparece un mensaje o indicio como que el sistema está procesando la acción?	B		
9	¿Las imágenes se muestran con buena resolución?	B		
10	¿No se muestran errores ortográficos?	B		
11	¿No hay ninguna imagen con información relevante?	B		
12	¿Si la respuesta a una acción se retrasa, aparece un mensaje o indicio como que el sistema está procesando la acción?	B		
13	¿Cuándo una tarea involucra documentos fuente, la interfaz es compatible con las características del documento fuente?	B		
Lenguaje común entre sistema y usuario				
14	¿El lenguaje es simple, con un tono adecuado?	B		
15	¿La información que se presenta en la aplicación es fácil de entender y memorizar?	B		
16	¿Utiliza los conceptos establecidos para las funciones estándar? (“buscar" para las búsquedas, etc.)	B		
17	¿Se utiliza siempre la misma nomenclatura para las mismas funciones?	B		
18	¿Los acrónimos y abreviaturas son definidos al ser	B		

	usados por primera vez?			
19	¿No hace uso de términos extranjeros?	B		
20	¿Utiliza un texto específico y descriptivo en los vínculos?	B		
21	¿La información es de rápida lectura, y con una disposición asequible?	B		
22	¿Los vínculos basados en nombres de la gente, conducen a las biografías cortas o a sus propios blogs, no a un correo electrónico?	B		
23	¿El lenguaje es simple, con un tono adecuado?	B		
24	¿La información que se presenta en la aplicación es fácil de entender y memorizar?	B		
Libertad y control por parte del usuario				
25	¿Existe una manera lógica de acceder a páginas relacionadas o a otras secciones?	B		
26	¿Tras una acción relevante hay una opción de vuelta atrás?	M		Cuando se genera el reporte no hay vuelta atrás
27	¿En las páginas internas hay un acceso a la página de inicio en una zona visible y reconocible?	B		
28	¿La aplicación cuenta con un mapa o buscador que facilite el acceso directo a los contenidos?	M		

Estética y diseño minimalista de los reportes				
29	¿Cumple el sitio con el principio de usabilidad de realizar las operaciones con un máximo de tres selecciones?	B		
30	¿Existe suficiente contraste entre el color del fondo y el del texto?	B		
31	¿Los tipos y tamaños de letra son legibles y distinguibles?	B		
32	¿Añade color de fondo a los div que llevan imagen de fondo? (Para los usuarios que desactivan las imágenes, desaparece el contraste entre texto y fondo, convirtiéndose en texto ilegible.)	B		
33	¿El uso de los colores es moderado?	B		
34	¿Se usan los estilos (negritas, cursivas...) con moderación? Si todo está resaltado con negrita o cursiva, el cerebro se acostumbra y deja de parecerle destacado.	B		
35	¿Utiliza la misma tipografía los mismos aspectos, ejemplo: mantener una tipografía estándar por elementos?	B		
36	¿Utiliza un interlineado adecuado para una buena lectura?	B		
37	¿Se usan frases breves y concisas: que resuman los puntos clave y vayan al grano?	B		
38	¿Los párrafos son cortos?	B		

39	¿Los textos están corregidos?	B		
40	¿Resalta en negrita los conceptos principales de los textos densos?	B		
41	¿Utiliza listas de boliche y numeradas?	B		
42	¿El sitio evita el uso excesivo del texto en mayúsculas?	B		
43	¿El texto dentro del sitio no debe estar justificado?	B		
44	¿No se utiliza el texto subrayado en el cuerpo del texto para resaltar a menos que sea un hipervínculo?	B		
Flexibilidad y eficiencia de uso				
45	Se definen de manera correcta los gráficos y las tablas utilizando atributos (leyendas, unidades de medida, etc.) en los reportes.	B		
46	¿Las páginas no requieren volver a escribir la información solicitada en páginas anteriores?	B		
47	¿Se implementan validaciones antes de que el usuario envíe información?	B		
48	Los reportes cumplen con las normas establecidas por la organización.	B		

Conclusiones parciales

Luego de la implementación y validación de la aplicación propuesta se arriba a las siguientes conclusiones:

Mediante el modelo de componentes se representó una vista estática de la aplicación, mostrando la organización y dependencias que existe entre los componentes físicos que se necesitan para ejecutar

la aplicación propuesta. El modelo de base de datos elaborado, permitió representar la abstracción, percepción y conocimiento en un sistema de información formado por un conjunto de objetos denominados entidades y relaciones. El uso de estándares de codificación y estilos de programación, permitió el entendimiento del código por otros programadores que no sean del equipo de desarrollo, para el mantenimiento de la aplicación propuesta. El modelo de despliegue permitió comprender la distribución física. La ejecución de las pruebas utilizando la herramienta JUnit, permitió demostrar que las funciones de la aplicación propuesta son operativas, mediante los casos pruebas ejecutados. La validación de la variable dependiente usabilidad mediante la lista de chequeo de Calidad UCI demuestra que se mejora la usabilidad en un 96 porciento con el desarrollo de la aplicación de Reporte para SIPACDroid.

CONCLUSIONES GENERALES

Con la investigación, se logró dar cumplimiento a los objetivos planteados inicialmente obteniendo como resultado una aplicación que cumple con todos los requisitos y la calidad requerida, lo cual permite arribar a las siguientes conclusiones:

- Se analizaron los principales conceptos relacionados con los reportes, estableciendo los elementos fundamentales regidos en la Instrucción no.1 del Presidente de los Consejos de Estado y de Ministros, para los diferentes tipos de reportes: anual, mensual e individual. Se identificaron los servicios REST, que permitirá la comunicación entre SIPAC y la aplicación propuesta de reporte en SIPACDroid. Se selecciona Android, como sistema operativo para el desarrollo, considerando sus características, según el estudio realizado de la cuota de mercado del SO móvil. Se analizaron las diferentes tecnologías para generar reportes y se utiliza la librería DroidText 0.4.js que permite generar el reporte y configurar la ruta donde será almacenado. Se analizaron diferentes metodologías ágiles por las características que presenta este tipo de desarrollo de software, seleccionando Variación AUP para la UCI, en correspondencia con los lineamientos de desarrollo de software de la universidad. Se identificaron las herramientas y lenguajes de programación que permitieron la construcción de la aplicación.
- El modelo conceptual elaborado, permitió relacionar el concepto de reporte con los planes y las actividades. Los requisitos funcionales identificados, a través del análisis de la documentación existente y las entrevistas realizadas, permitió definir lo que el sistema debe hacer, así como las características requeridas del software y de la organización que se desarrolla. El diseño de la aplicación, permitió representar las clases con sus atributos y las relaciones entre las mismas, utilizadas en la implementación de la aplicación propuesta. La arquitectura definida, permitió describir el esqueleto interno con sus componentes y los servicios REST utilizando los métodos GET, POST, PUT y DELETE definidos por el protocolo HTTP.
- La implementación de la aplicación para dispositivos móviles desarrollada, permitió obtener un producto para brindar y consumir los servicios REST del módulo de Reporte en el SIPAC. La ejecución de las pruebas unitarias con la herramienta JUnit, con resultados satisfactorios, permitió corroborar que los requisitos funcionales están correctamente implementados, para un conjunto de entradas, condiciones de ejecución y resultados esperados.

- La validación de la variable dependiente usabilidad mediante la lista de chequeo de Calidad UCI demuestra que se mejora la usabilidad en un 96 porciento con el desarrollo de la aplicación de Reporte para SIPACDroid.

RECOMENDACIONES

Se recomienda que se desarrolle un componente que permita actualizar los modelos de los reportes de los planes cuando se realicen actualizaciones en el documento que rige la planificación estratégica en Cuba.

BIBLIOGRAFÍA

About SQLite. (2018). Recuperado el 1 de diciembre de 2019, de About SQLite: http://www.sqlite.org/about.html

Agile, D. (2018). *Simple Tools for Software Modeling -OR- It's "Use the Simplest Tool" not "Use Simple Tools"*. (Disciplined Agile) Recuperado el 10 de Junio de 2019, de http://www.agilemodeling.com/essays/simpleTools.htm#SelectingCASE

Agile,Disciplined. (2018). Obtenido de Simple Tools for Software Modeling: http://agilemodeling.com/essays/simpleTools.htm#SelectingCASE

Aguilera López, P. (16 de Octubre de 2016). *Seguridad informática.* Obtenido de https://books.google.es/books?hl=es&lr=&id=Mgvm3AYIT64C&oi=fnd&pg=PA1&dq=seguridad+inform%C3%A1tica&ots=PpqoODDCR1&sig=_da7QHG_8yjuzjH3wsS5SnMNOn0#v=onepage&q=seguridad%20inform%C3%A1tica&f=false

Anderson, R. (2001). *Security Engineering: A Guide to Building Dependable Distributed Systems.* Jonh Wiley & Sons, Inc.

Android Developers. (2018). *Android Developers*. Recuperado el 12 de Noviembre de 2018, de https://developer.android.com/

Asteasuain, F., & Schmidt, L. A. (2005). Aplicación de la Programación Orientada a Aspectos como Solución a los Problemas de la Seguridad en el Software.

Board, I. S. (2018). *International Software Testing Qualifications Board.* Recuperado el 2019 de Abril de 19, de https://www.istqb.org/downloads/category/2-foundation-level-documents.html

Brito, H. R. (20 de 03 de 2017). *Behique Digital*. Obtenido de Behique Digital: https://henryraul.wordpress.com/2016/10/11/owasp-top-10-proactive-controls-2016/

Camarero, P. J., Fumero, A., Blanco, P., Werterski, A., & Rodríguez, P. (2009). *Metodología de desarrollo ágil para sistemas móviles.Introducción al desarrollo con Android e Iphone.* Madrid, España: Universidad Politécnica de Madrid.

Carrera, O. (3 de 2012). *olgacarrera.blogspot.com.*

BIBLIOGRAFÍA

CDI. (24 de Enero de 2017). *Centro de Delitos Informáticos*. Obtenido de Centro de Delitos Informáticos: https://centrodelitosinformaticos.com/seguridad-informatica/

Consejo de ministro, y otros. (2012).

Cruz, M. H. (2017). *Sistemas Operativos Moviles.* Chilpancingo.

David G. Rosado, C. B. (2009). *La Seguridad como una asignatura indispensable para un Ingeniero del Software.* La Mancha.

Dick, J., Hull, E., & Jackson, K. (2017). En *Requirements Engineering.* Suiza: Springer.

Django REST framework. (2018). Recuperado el 6 de noviembre de 2019, de http://www.django-rest-franework.org/.

Eldhuset, A. (s.f.). *Kotlinland.* Recuperado el 19 de octubre de 2019, de jetBrains: http://kotlinlang.org/docs

Gaines, B. G. (2017). *Visual Paradigm Online.* Obtenido de http://online.visual-paradigm.com/es/features

Gaines, J., Boyd, G., & Copley, D. (2017). *Visual Paradigm Online*. Obtenido de https://online.visual-paradigm.com/es/features/

Garfinkel, S. (1999). *Seguridad y Comercio en la Web.* McGraw Hill/Interamericana de España.

Gonzalez, M. (2014). *SlideShare*. Recuperado el 2018, de https://es.slideshare.net/MonicaGlez1/ingeniera-de-software-30069503

Gradle Inc. (2018). *Gradle Docs*. Recuperado el 10 de mayo de 2018, de https://docs.gradle.org/current/userguide/userguide.htm

Granada, D. d. (2014). *Desarrollo de aplicaciones para Android.*

Grisales, V. B. (s.f.). *modelo ISO/IEC 25010 en el proceso de evaluación de la calidad del software.*

Huebe, M. d. (2005). *Ingeniería de Requerimientos.* Pachuca, México.

ISO/IEC. (2014). *SQuaRE (System and Software Quality Requirements and Evaluation).*

BIBLIOGRAFÍA

Jacobson. (2000). *El Proceso unificado de desarrollo de software. Capítulo 6.*

JetBrains. (s.f.). *Kotlinlang*. (JetBrains) Recuperado el 19 de Abril de 2019, de https://kotlinlang.org/docs/reference/coding-conventions.html?hl=es_419

Larman, C. (2004). *UML y Patrones.Introducción al análisis y diseño orientado a objetos.* La Habana: Félix Varela.

Mentor, A. (12 de marzo de 2017). *Aula Mentor*. Obtenido de Aula Mentor. Seguridad informática. Normas ISO sobre gestión de seguridad de la información: http://descargas.pntic.mec.es/mentor/visitas/demoSeguridadInformatica/normas_iso_sobre_gestin_de_seguridad_de_la_informacin.html

Ministro, P. d. (2012). *modelo No.1.* proceso de planificación de Gobierno.

Ministro, p. d. (2012). *modelo No.1(instrucción No.1).*

Ministro, P. d. (2012). *modelo No.9.* Proceso de planificación del Gobierno.

Olavide, M. r. (2015). Patrón Modelo Vista Presentador(MVP).

OWASP. (21 de 03 de 2017). *OWASP*. Obtenido de OWASP: https://www.owasp.org/index.php/Main_Page

Pressman, R. S. (2010). *Ingeniería de software un enfoque práctico Séptima edición.*

Rajagopal, D., & Thilakavalli, K. (2017). A Study: UML for OOA and OOD.

Roger S. Pressman, B. R. (2015). *Software Engineering. A practitioner´s Aproach. 8th Edition.* New York.

Samuel Ojeda Pereira, L. G. (2018). *Proceso para la elaboración y recuperación de información de la Planificación de Actividades del Sistema de Planificacion de Actividades.*

Sánchez, T. R. (2015). *Metodología de desarrollo para la Actividad productiva de la UCI.*

Sommerville, I. (2005). *Ingeniería del software. 7ma Edición.* United Kingdom: Pearson Education.

StartCounter. (1999-2020). *http://qs.statcounter.com/os-market-share/mobile/cubq*.

(2019). *StatCounter.* Recuperado el 25 de octubre de 2019, de http://gs.statcounter.com/os-market-share/mobile/cuba

BIBLIOGRAFÍA

UCI. (26 de Enero de 2017). *Universidad de las Ciencias Informáticas*. Obtenido de Universidad de las Ciencias Informáticas: http://www.uci.cu

Visual Paradigm . (2018). *Visual Paradigm - Leading UML, BPMN, EA, Agile and Project Management Software.* Recuperado el 2018, de https://www.visual-paradigm.com/

XML| Object Management Group. (2018). Recuperado el 1 de noviembre de 2019, de http://www.omg.org/technology/readingroom/XML.htm

Yamila Estrada, W. A. (2012). Fundamentos para implementar y certificar un Sistema de Gestión de la Seguridad Informática bajo la Norma ISO/IEC 27001. *Serie Científica de la Universidad de las Ciencias Informáticas*, 10.

ANEXOS

ANEXOS

Anexo 1: Reporte del plan de trabajo individual

_ORDINARIO
Ejemplar único
Cantidad de hojas: 1

APROBADO:

PLAN DE TRABAJO INDIVIDUAL PARA EL MES DE SEPTIEMBRE DE 2017

TAREAS PRINCIPALES:

1.
2.
3.
4.
5.

Lunes	Martes	Miércoles	Jueves	Viernes	Sábado	Domingo
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17

ANEXOS

Anexo 2: Reporte del resumen de cumplimiento

_ORDINARIO
Ejemplar único
Cantidad de hojas: 3

APROBADO:

PLAN DE TRABAJO PARA EL MES DE SEPTIEMBRE 2016 DE CENTRO DE INFORMATIZACIÓN DE ENTIDADES

TAREAS PRINCIPALES:

1.
2.
3.
4.
5.
6.
7.
8.
9.

No.	Actividad Hora y Lugar	Fecha	Dirige	Participantes	Observaciones
1				_ _	
2					
3					
4					
5					

ANEXOS

Anexo 3: Reporte del resumen de cumplimiento

_ORDINARIO
Ejemplar único

UNIVERSIDAD DE LAS CIENCIAS INFORMATICAS

RESUMEN DEL CUMPLIMIENTO DEL PLAN DE TRABAJO DEL MES SEPTIEMBRE DE 2017

RESUMEN CUANTITATIVO

TOTAL DE TAREAS PLANIFICADAS	DE ELLAS			EXTRA PLANES
	CUMPLIDAS	INCUMPLIDAS	MODIFICADAS	

TAREAS INCUMPLIDAS Y EXTRA PLANES	
INCUMPLIDAS	CAUSAS
EXTRA PLANES	QUIÉN LAS ORIGINÓ Y MOTIVO

ANEXOS

Anexo 4: Reporte del resumen de cumplimiento

Jefe de Departamento

APROBADO:

CORDINACIÓN DE LAS ACTIVIDADES PARA EL AÑO 2019

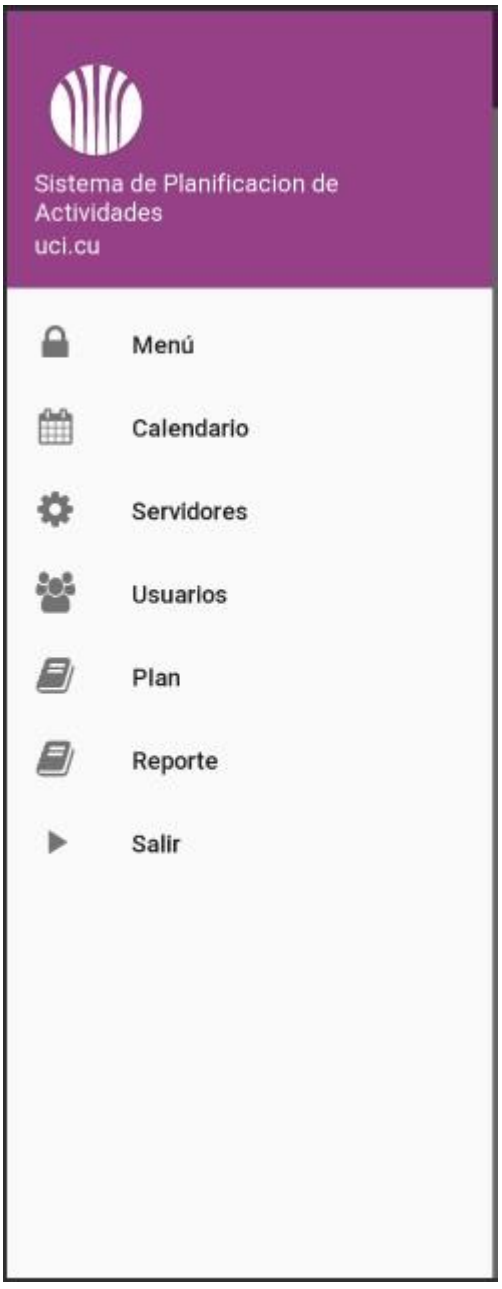
Mes	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Enero																															
Febrero																															
Marzo																															
Abril																															
Mayo																															
Junio	X		X	X		X	X			X	X	X	X	X			X		X		X					X		X			
Julio																															
Agosto																															
Septiembre																															
Octubre																															
Noviembre																															
Diciembre																															

ANEXOS

Anexo 5: Autenticación de usuario en SIPACDroid

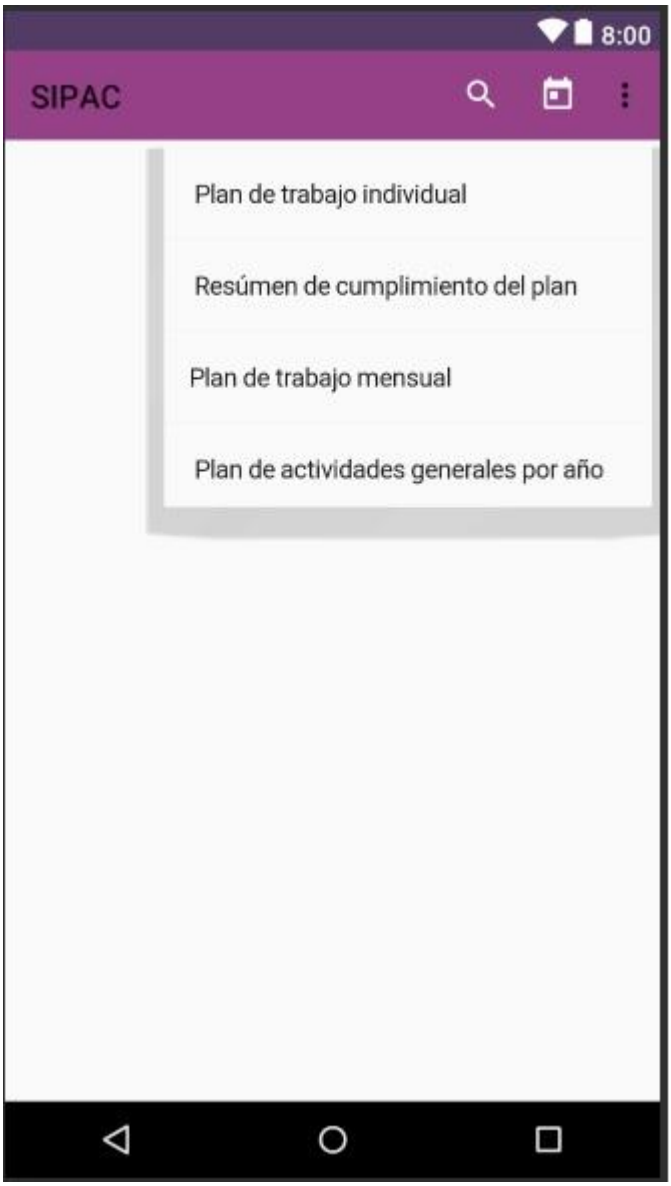
ANEXOS

Anexo 6: Menú principal



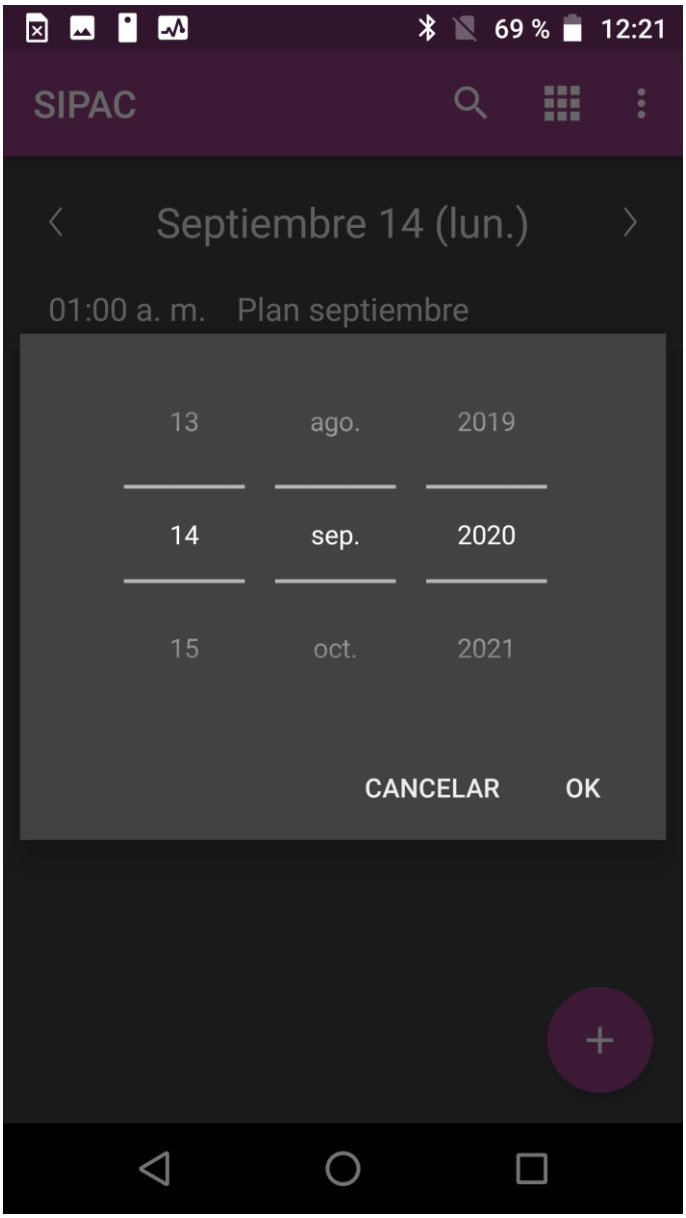
ANEXOS

Anexo 7: Menú de reportes



ANEXOS

Anexo 8: Filtros de días, mes y años

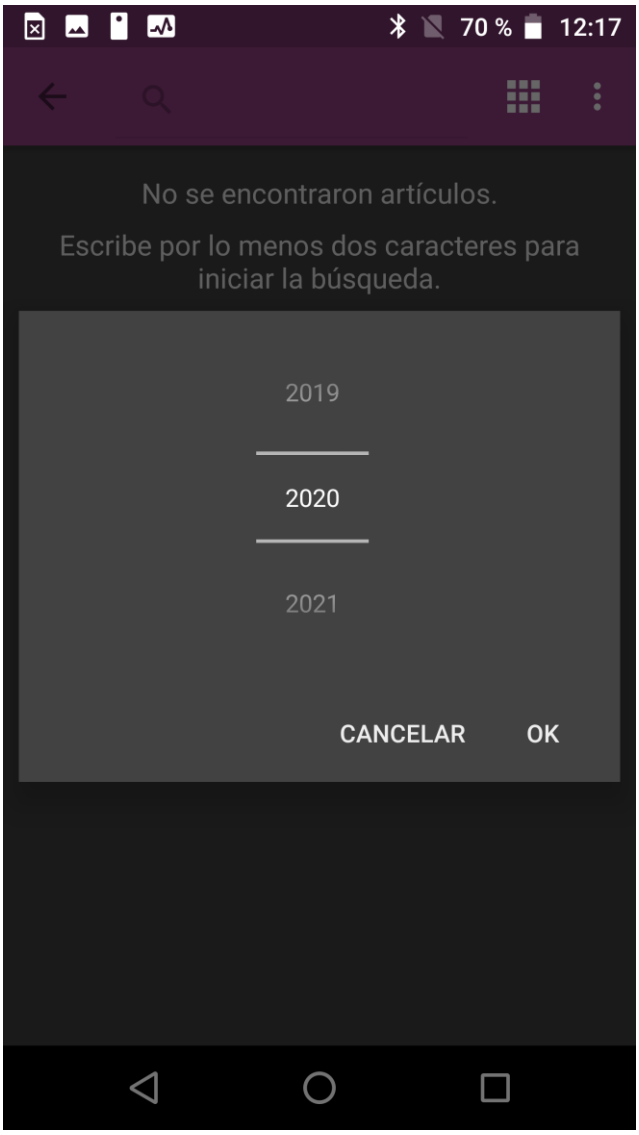


ANEXOS

Anexo 9: Lista de los planes mensuales

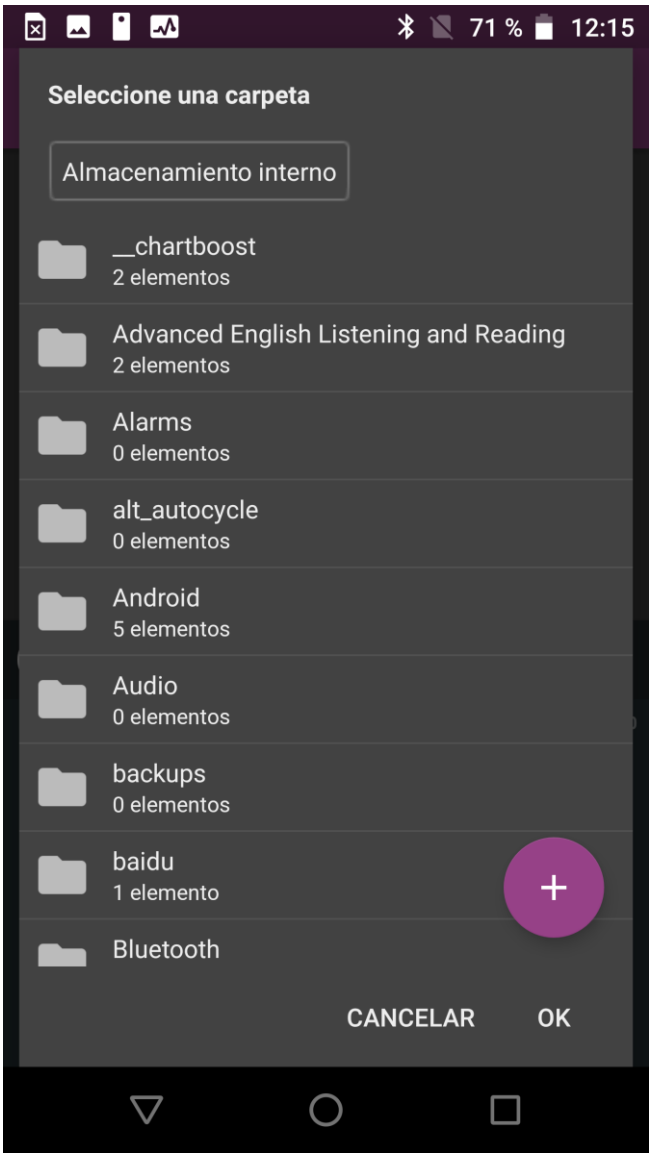
ANEXOS

Anexo 10: Filtro para el plan anual



ANEXOS

Anexo 11: Importar reporte del plan en el dispositivo



GLOSARIO DE TÉRMINOS

GLOSARIO DE TÉRMINOS

Actividades: conjunto de operaciones o tareas, propias de una persona o entidad, destinadas para cumplir determinado(s) objetivo(s).

Anexo: fichero que se anexa a un elemento de la planificación, el mismo puede contener cualquier objeto digitalizado de los siguientes: documentos (PDF, EXCEL, DOC) o imágenes.

Categoría de la actividad: elemento de clasificación para el ordenamiento de las actividades con el fin de mejorar su organización y control.

Criterio de medida: condición que debe cumplir una o varias medidas para que el indicador al que tributan sea evaluado con un resultado determinado.

Entidad: colectividad considerada como unidad, especialmente, cualquier corporación, compañía, institución, etc. En el campo de la planificación se aborda genéricamente como organización; referida a los disímiles grupos humanos conformados para cumplir sus correspondientes objetivos explícitos e implícitos.

Factores que Intervienen en la Planificación (FIP): documento que contiene elementos que desencadenan cambios en los planes, el mismo es emitido por el nivel superior o propio.

Indicador: puntos de referencia, que brindan información cualitativa o cuantitativa, conformada por una o varias medidas, que permiten seguir el desenvolvimiento de un proceso y su evaluación, y que deben guardar relación con el mismo.

Plan: modelo sistemático que se elabora antes de realizar una acción, con el propósito de dirigirla y encausarla. En este sentido, un plan también es un documento que precisa los detalles necesarios para realizar una misión.

Planificación: acciones llevadas a cabo para realizar planes y proyectos de diferente índole. La planificación ejecuta los planes desde su concepción y si es el caso, se encarga de la operación en los diferentes niveles y amplitudes de la planeación.

GLOSARIO DE TÉRMINOS

Planificar: acción de crear o proyectar metas donde se interrelacionan los factores tiempo, espacio, recursos y personas.

Principio de consecución: condición que debe cumplir uno o varios indicadores para que el objetivo al que tributan sea evaluado con un resultado determinado.