Aplicación para dispositivos móviles de los módulos de autenticación, notificaciones y calendario del SIPACDroid.

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

Autor: Angel Enrique Herrera Crespo

Tutoras:

Ing. Claudia Bravo Batista

Ing. Liset Gómez Chávez

DECLARACIÓN DE AUTORÍA

Declaramos que somos los únicos autores de este trabajo y autorizamos a la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio. Para que así conste firmamos la presente a los _____ días del mes de _____ del año _____.

_____

Angel Enrique Herrera Crespo

_____ _____

Ing. Claudia Bravo Batista Ing. Liset Gómez Chávez

AGRADECIMIENTOS

A Dios primeramente por ser la razón por la que vivo, todo esto es gracias a Él.

A mi mamá Milagros por aguantarme y sustentarme toda mi vida, aunque nunca te lo diga. Gracias por todo. Te amo.

A mi papá, que pese a estar lejos físicamente, siempre ha estado pendiente de mí.

A mi padrastro Nelson que ha sido el apoyo y sustento de mi madre y por consecuente mío. Gracias de veras.

A mi tía Mayelin que fue la primera persona en tratarme como un adulto y como hombre antes de que yo mismo lo supiera.

A mi familia en general por siempre estar ahí, en las buenas y en las malas.

Para mis amigos de la UCI, mi otra familia, que saben más de mí que nadie, no los menciono a todos porque no quiero olvidar a ninguno.

A mi Mota por aguantarme mi mal carácter, y abrazarme aun cuando no lo necesitaba. Te quiero.

A mis tutoras, por siempre apoyarme y ayudarme en todo lo que necesitaba.

A Dariela por sus consejos y apoyo en todo sobre todo en las cosas que solo los dos sabemos. De verdad te agradezco mucho.

DEDICATORIA

A mis abuelos, los cuales nunca llegue a conocer, pero estoy seguro que estarían orgullosos de mí. En donde quiera que estén esto es para ustedes.

A mis padres por su apoyo incondicional por ser mi fortaleza e inspiración diaria, ha sido y es, un privilegio ser su hijo, gracias por darme siempre lo mejor que pudieron.

RESUMEN

El Sistema de Planificación de Actividades (SIPAC) 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 con 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. Incluye el módulo de Planificación que permite ejecutar la planificación de actividades basadas en reglas de la compartimentación de la información, permitiendo que la información planificada sea accedida por la persona autorizada, en el momento indicado. La presente investigación tiene como objetivo el desarrollo de una aplicación para dispositivos móviles que permita brindar y consumir los servicios REST (por sus siglas en inglés Representational State Transfer) de los módulos de calendario y autenticación SIPAC, para mejorar la portabilidad del sistema.

Para guiar el proceso de desarrollo de la aplicación se utilizó como metodología AUP-UCI, Kotlin y XML como lenguajes de programación y Android Studio 3.4.1 como Entorno de Desarrollo Integrado (IDE en sus siglas en inglés). Se definieron los requisitos de la aplicación, se realizó el análisis y diseño garantizando la correcta implementación. Se ejecutaron las pruebas unitarias utilizando la herramienta JUnit. Se validó la investigación utilizando la norma ISO/IEC 25023 que define específicamente las métricas para realizar la medición de la calidad de productos y sistemas software.

Palabras Claves: autenticación, dispositivos móviles, calendario, portabilidad, servicios REST, SIPAC

ABSTRACT

The Activity Planning System (SIPAC) is part of the Xedro comprehensive management solutions package for budgeted and corporate entities, which is based on the principles of technological independence and with general functionalities of the processes and the characteristics of the Cuban economy. It´s functional characteristics were designed in accordance with the policies issued by the governing bodies of the Cuban State. It includes the Planning module that allows executing the planning of activities based on rules for the compartmentalization of information, allowing the planned information to be accessed by the authorized person, at the indicated time. The objective of this research is the development of an Application for mobile devices that allows to provide and consume the REST services (for its acronym in English of Representational State Transfer) of the Planning module of SIPAC, to improve the portability of the system.

To guide the development process, the application was used as methodology AUP-UCI, Kotlin and XML as programming languages and Android Studio 3.4.1 as Integrated Development Environment (IDE in its acronym in English). The requirements of the application were defined, the analysis and design were carried out guaranteeing the correct implementation. The unit tests were executed using the JUnit tool. The research was validated using the ISO / IEC 25023 standard that specifically defines the metrics for measuring the quality of software products and systems.

Keywords: mobile devices, planning, portability, REST services, Android, SIPAC

ÍNDICE DE CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

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.

En la Instrucción No 1. se refiere que las actividades tienen como objetivo principal establecer las acciones que se ejecutarán durante el proceso de planificación y, deben tener una correspondencia directa y responder a los objetivos de trabajo y los recursos de las entidades. Se caracterizan por poseer un carácter flexible, es decir, deben ser totalmente gestionables al producirse algún cambio de escenario o al surgir nuevas necesidades dentro de las instituciones. Sobre estas tareas o actividades debe existir un registro y control periódico por parte de la institución mediante un sistema de registro, comprobación y monitoreo, que permita asegurar el cumplimiento de los planes en todos sus niveles. (Consejo de Ministros, 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.

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), lo que 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.

Dicho sistema contiene 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 seguridad que garantiza la autenticación y autorización en el sistema entre otros aspectos. El

módulo de calendario o actividades que permite la gestión de los planes y las actividades de los usuarios, así como la integración con otros calendarios, a través de la reutilización y extensión del módulo Radicale[1] que actúa como servidor CALDAV (Calendaring and Scheduling Extensions to WebDAV)[2]. Este último, permite crear las actividades de la entidad, mostrándolas de diferentes formas en correspondencia a los ajustes del tema seleccionado por el usuario, en vista anual, mensual, diaria, semanal, como una lista de elementos y en forma de gráfico. El módulo de notificaciones que notifica en tiempo real y por correo las ocurrencias de actividades. Además, maneja una bandeja de notificaciones.

Según la Norma ISO/IECE 9126-1:2000, la portabilidad es una de las características que categorizan los atributos de calidad de un producto y es definida como la capacidad que tiene un producto de software para ser transferido de un ambiente a otro, incluyendo entornos organizacionales, de hardware o de software. Esta característica está compuesta por las siguientes subcaracterísticas: adaptabilidad, instalabilidad, coexistencia, reemplazabilidad y cumplimiento con la portabilidad. Para cada característica y subcaracterística, la capacidad del software es determinada por un conjunto de atributos internos que pueden ser medidos externamente por la capacidad provista por el sistema que contiene el software.

En la actualidad solamente se encuentra desarrollado para dispositivos móviles[3] específicamente para aquellos que poseen el Sistema Operativo(SO) Android el módulo de Planificación del SIPAC, que implementa las operaciones a realizar con los documentos de la planificación, entiéndase: planes, objetivos, actividades, áreas de resultados clave (ARC) y los factores que intervienen en la planificación (FIP). Por tanto, los múltiples usuarios que hacen uso del sistema solo pueden acceder a las diferentes funcionalidades de los restantes módulos cuando tengan disponible un dispositivo móvil y/o computadora (PC) con comunicación directa a la aplicación ya sea utilizando el protocolo CalDAV o mediante un navegador web.

La utilización del protocolo CalDAV permite sincronizar las actividades asociadas al usuario anteriormente autenticado con los calendarios offlines y/o onlines que se encuentran en estos dispositivos y lo soportan. Una de las desventajas de este protocolo es la posibilidad de solo sincronizar los campos predefinidos o estándares en todos los calendarios, es decir los campos concretos o diferentes que posee el SIPAC entiéndase: organismos, dirigentes, tipo de actividad, entre otros; son imposibles de sincronizar, por lo que la personalización de estos términos no es soportada por este. Por otra parte, no garantiza la seguridad de la información, una vez

[1] Servidor CalDAV libre y de código abierto desarrollado por la Compañía de Software Francesa Kozea.
[2] Protocolo que permite la comunicación entre calendarios y permitiendo gestionar los eventos de estos.
[3] Refiere a cualquier dispositivo o terminal con algunas capacidades de procesamiento de datos que pueda transportarse con facilidad y que sea de reducido tamaño con capacidad para conectarse a una red inalámbrica. (Delía, y otros, 2013)

sincronizadas las actividades de los calendarios del dispositivo móvil, las personas que puedan acceder el dispositivo, pueden visualizar, agregar, modificar y/o eliminar las actividades que están planificadas en el calendario. Lo antes descrito, tiene como implicación que la información de la planificación del país que es totalmente confidencial quede expuesta, afectando la confidencialidad y autenticidad de la información.

El acceso desde cualquier navegador web a la ruta donde se encuentra disponible la aplicación en el servidor web mediante un método de conexión como la Wi-Fi, implica que la conectividad sea estable y el SIPAC se encuentre accesible en el espacio, lugar de acceso y momento. Sin embargo, el sistema no se encuentra diseñado, para gestionar la pérdida de la conexión.

Además, el sistema solo posee la capacidad de ser instalado en los sistemas operativos: Windows (versión 7 o superior siempre a 64 bits), Mac (desde la versión de MacOS El Capitán 10.11 en adelante) y Linux (a 64 bits y con la versión 3.10 o superior del kernel de Linux). Conjuntamente a esto, debe ser adaptado a diversos entornos de hardware que tengan como prestaciones mínimas una PC con características de Procesador Intel(R) Core (TM) i3-2120 o algún equivalente, 4GB de RAM. Debido a estas características tecnológicas, no es posible instalar en dispositivos móviles los módulos de seguridad (autenticación y autorización) y calendario, viéndose afectada las subcaracterísticas de la portabilidad: adaptabilidad e instalabilidad.

A partir de la situación antes descrita se ha identificado el siguiente **problema a resolver:** ¿Cómo aumentar la portabilidad de la planificación de actividades en SIPAC?

Estableciendo como **objeto estudio**: las aplicaciones para dispositivos móviles, enmarcado en el **campo de acción**: las aplicaciones para dispositivos móviles de calendarios, autenticación y notificaciones.

Para darle solución al problema planteado se define como **objetivo general:** Desarrollar una aplicación móvil que permita brindar y consumir los servicios REST del Sistema de Planificación de Actividades para aumentar la portabilidad de la planificación de las actividades.

Para guiar la investigación se especifican los siguientes **objetivos específicos:**

- Construir el marco teórico de la investigación a partir de la búsqueda y revisión bibliográfica de las aplicaciones de calendario para dispositivos móviles.
- Diseñar una aplicación móvil que permita brindar y consumir los servicios REST de los módulos de autenticación, notificaciones y calendario del SIPAC.
- Implementar una aplicación móvil que permita brindar y consumir los servicios REST de los módulos de autenticación, notificaciones y calendario del SIPAC.

- Validar la aplicación móvil que permita brindar y consumir los servicios REST de los módulos de autenticación, notificaciones y calendario del SIPAC.

Se plantea como **idea a defender:** Si se desarrolla una aplicación móvil que permita brindar y consumir los servicios REST de los módulos de autenticación, notificaciones y calendario del SIPAC, entonces se aumentará la portabilidad de la planificación de las actividades.

Los posibles resultados están centrados en alcanzar con el desarrollo de la investigación, una aplicación móvil que permita brindar y consumir los servicios REST de los módulos de autenticación, notificaciones y calendario del SIPAC. Además, de la documentación técnica de la investigación.
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 análisis en las diferentes fuentes bibliográficas sobre las aplicaciones móviles de calendarios, los tipos de notificaciones en Android y servicios REST de los módulos de autenticación y calendario del SIPAC.
- **Método sistémico:** su aplicación permitió determinar los elementos de las actividades en el SIPAC identificando la interrelación entre los mismos, posibilitando llegar a una mejor comprensión del negocio a informatizar.
- **Analítico-sintético:** permitió sintetizar la documentación identificada, extraer los elementos asociados a la problemática planteada, así como la selección de las tecnologías y metodologías adecuadas para el desarrollo de la solución propuesta.

Métodos empíricos:

- **Observación**: la observación científica como método consiste en la percepción directa del objeto de investigación. El cual, 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, establecer relaciones entre ellas y diseñar las vistas que se adapten dinámicamente a su contenido, logrando así reducir tiempo de producción.
- **Entrevista no estructurada o abierta:** obteniendo información para la construcción simultánea de la aplicación a partir de las respuestas del entrevistado, necesitando el entrevistador una gran cantidad de documentación y preparación.

INTRODUCCIÓN

El presente trabajo de investigación está compuesto por tres capítulos, estructurados de la siguiente manera:

Capítulo 1. Fundamentación teórica de la investigación

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 sobre los calendarios. También se describen y caracterizan la metodología, herramientas y tecnologías a utilizar para el desarrollo de la solución propuesta.

Capítulo 2. Descripción de solución propuesta

Se realiza el modelo conceptual para identificar los principales conceptos del mundo real. Se elaboran los productos de trabajo generados por la metodología seleccionada. Se modelan y detallan los diagramas que representan las funcionalidades del sistema que guiarán la posterior implementación, a partir de los patrones de diseño identificados.

Capítulo 3. Implementación, prueba y validación de la investigación

Se muestra la distribución física de los distintos componentes lógicos desarrollados, a través del modelo de despliegue y la organización del sistema mediante el modelo de componentes. Se explican los estilos de programación y estándares de codificación empleados y por último se prueba que la aplicación funciona correctamente y la investigación responde a la problemática planteada.

CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA DE LA INVESTIGACIÓN

Introducción

En el presente capítulo son abordados los principales conceptos utilizados en la investigación. Se describen las principales tecnologías, lenguajes de programación y herramientas definidas para el desarrollo de la solución, así como la metodología de desarrollo y los patrones empleados.

1.1 Principales conceptos

Atendiendo a las características de la planificación en Cuba y el objetivo social descrito anteriormente se definen un conjunto de conceptos significativos para el posterior entendimiento del proyecto.

1.1.1 Actividades

Dentro de la planificación estratégica en la que se encuentra inmersa nuestro país, al mencionar el término “actividades” se hace referencia a las grandes decisiones de largo y mediano plazo dentro de una institución, al establecimiento de los objetivos de estratégicos que permiten materializar su misión y visión; estableciendo su foco de atención en los aspectos del ambiente externo de esta: Los usuarios finales a quienes se entregan los productos principales o estratégicos y los resultados finales o los impactos de la intervención institucional (Armijo, 2011)

Durante el proceso de planificación, las actividades y objetivos de trabajo deben cumplir los siguientes principios fundamentales:

a. **El carácter rector de los objetivos de trabajo**, por constituir la categoría rectora de la planificación de actividades y determinar el desarrollo, desempeño y funcionamiento de las organizaciones, para alcanzar resultados concretos y tangibles, en cumplimiento de su misión y de las funciones estatales asignadas.
b. **Correspondencia entre objetivos de trabajo, actividades y recursos**, concebidos como un sistema, asegurado a través del Plan de Actividades y su alcance se regula por el Plan de la Economía Nacional, teniendo en cuenta lo aprobado en el VI Congreso del Partido Comunista de Cuba.
c. **Participación directa y activa de los jefes, por el cumplimiento del proceso de planificación**, orientando las políticas de trabajo y adoptando las decisiones correspondientes.
d. **Participación y compromiso del colectivo**, en las partes que les concierne, resultando importante este proceso, para una mejor ejecución de los planes.
e. **Carácter flexible**, para permitir introducir, quitar, modificar o trasladar, objetivos de trabajo o actividades, producto de cambios en los escenarios o nuevas necesidades.

f. **Eficacia y eficiencia**, para hacer lo que corresponda en cada etapa de trabajo con el uso racional y eficiente de los recursos disponibles y del tiempo, en función del cumplimiento de los objetivos de trabajo.
g. **Disciplina y coordinación**, para exigir el cumplimiento de lo planificado, así como una estrecha coordinación y cooperación entre los ejecutores en los diferentes niveles de dirección.
h. **Registro y control sobre la marcha del cumplimiento de los planes**, mediante un sistema de registro, comprobación y monitoreo, que permitan asegurar el cumplimiento de los planes.

Según el Anexo No.2 de la Instrucción No.1 las tareas o actividades que pueden reflejarse en los planes de trabajo individuales se clasifican en:

1. De dirección (planificación, coordinación entre áreas y control).
2. De control del cumplimiento de acuerdos, indicaciones u otros documentos que provienen del nivel superior o de los propios que genera su sistema, como, por ejemplo: cumplimiento de acuerdos de la Reunión Conjunta del Buró Político y el Comité Ejecutivo del Consejo de Ministros, del Consejo de Dirección (Administración), de las reuniones y despachos con subordinados u otros jefes, etc.
3. Principales tareas a cumplir en el mes, que tributan a los objetivos de trabajo de la organización.
4. Sistemáticas y periódicas de autocontrol, de control a los subordinados y de aseguramiento de tareas en cumplimiento de sus deberes funcionales, trabajo en comisiones y grupos gubernamentales, entre otras.
5. Visitas a entidades, empresas y territorios.
6. Acciones de superación.
7. Preparación para reuniones y otras actividades.
8. Comprobación, procesamiento, tramitación y entrega de la información.
9. Tareas cuyo cumplimiento sobrepasan la fecha del mes anterior.
10. Tareas sociopolíticas.

La forma de representación de estas actividades de este tipo de plan se ilustra según el formato que se encuentra definido en el Modelo No. 3 de esta Instrucción:

Modelo No. 3

Cargo del que lo aprueba

Aprobado:

Nombre (s) y apellidos del que lo aprueba

PLAN DE TRABAJO INDIVIDUAL PARA EL MES DE.....

Tareas Principales

1.-
2.-
3.-

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

Cargo del que lo presenta

Nombre (s) y apellidos

Figura 1. Modelo No.3 de la Instrucción No.1.

En el Plan Anual de Actividades se recogen las principales tareas contenidas en los planes del Partido y de las Asambleas del Poder Popular (Nacional, Provincial y Municipal) que tienen incidencia en el Gobierno; las que aseguran: el cumplimiento de los acuerdos y resoluciones del VI Congreso del Partido Comunista de Cuba y los objetivos de trabajo en cumplimiento de las funciones estatales, ejecutivas y administrativas de cada nivel de dirección, tales como visitas de trabajo, actividades de preparación, productivas y de servicios, cursos, eventos, salidas al exterior, temas de los consejos de dirección o de la administración, reuniones, despachos, entre otros.

El contenido de los capítulos del Plan Anual de Actividades y la clasificación de las actividades en este aparecen en el Anexo No.1 de esta Instrucción como se muestra a continuación:

I. **Trabajo político - ideológico y de organización del Partido:** Incluye las actividades de la Central de Trabajadores de Cuba, la Unión de Jóvenes Comunistas y de las organizaciones de masa que tienen incidencia en el Gobierno, las de carácter político e ideológico que tienen relación con el funcionamiento de los órganos, organismos de la Administración Central del Estado, entidades nacionales y las administraciones locales del Poder Popular, así como actos de contenido político ideológico.

II. **Funcionamiento y control del Estado**: Incluye las actividades de la Asamblea Nacional del Poder Popular y de los Órganos del Estado, que tienen incidencia en el Gobierno, así como las acciones de control que ejercen.

III. **Funcionamiento y control del Gobierno**: Incluye las actividades del Comité Ejecutivo del Consejo de Ministros, las acciones de control que ejercen los órganos auxiliares que forman parte de este, así como las actividades de la Comisión de Implementación y Desarrollo de los Lineamientos aprobados en el VI Congreso del Partido Comunista de Cuba.

IV. **Funciones y encargo estatal en los OACE Y de las administraciones locales del Poder Popular. Funciones empresariales:** Incluye las actividades por orden jerárquico, relacionadas con:

a. Resultados que proponen alcanzar cada organización, en interés del cumplimiento de los acuerdos del VI Congreso del Partido Comunista de Cuba.
b. Objetivos del organismo aprobados para el año.
c. Lineamientos y enmarcamientos fijados por el Gobierno para el cumplimiento de sus funciones estatales, el crecimiento y desarrollo que propone alcanzar la organizaci6n, aseguramiento a la producci6n de bienes y servicios y a la eficiencia, eficacia y calidad de todas sus actividades.
d. Control externo que realizan los Órganos y Organismos de la Administración Central del Estado en cumplimiento de sus funciones estatales.
e. Actividades de las organizaciones no gubernamentales con incidencia en el Gobierno.

V. **Funcionamiento Interno:** Incluye actividades propias del organismo para su sistema de dirección, acciones de control interno, atención al hombre, aplicación de la política el sistema de trabajo con los cuadros y sus reservas y el cumplimiento de las disposiciones vigentes, entre otros.

VI. **Defensa, Orden Interior y Defensa Civil:** Incluye las actividades de los niveles superiores y las propias que generan estas direcciones de trabajo, sobre la base de lo establecido por los organismos rectores, así como las referidas a la reducción de riesgos.

La forma de representación de estas actividades del plan anual se elabora según el formato del acápite II contenido en el Modelo No. 1 del este documento rector de la Planificación en Cuba como se muestra a continuación:

II. PLAN DE ACTIVIDADES

No	Actividades, hora y lugar	Meses												Dirige	Participantes	Observaciones sobre el cumplimiento
		E	F	M	A	M	J	J	A	S	O	N	D			
I.- Trabajo político–ideológico y de organización del Partido.																
II.- Funcionamiento y control del Estado.																
III.- Funcionamiento y control del Gobierno.																
IV.- Funciones y encargo estatal en los OACE y de las administraciones locales del Poder Popular. Funciones empresariales.																
V.- Funcionamiento Interno.																
VI.- Defensa, Orden Interior y Defensa Civil.																

Figura 2. Acápite II del Modelo No.1 de la Instrucción No.1.

Durante el proceso de planificación de las actividades y, para facilitar la distribución temporal de las tareas distribuidas es posible emplear el Gráfico de Coordinación (Modelo No. 4) en el cual se reflejan las actividades a realizar en todo un año como se evidencia a continuación:

Modelo No. 4

VARIANTE DEL GRÁFICO DE COORDINACIÓN DE LAS ACTIVIDADES PARA EL AÑO

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
E																															
F																															
M																															
A																															
M																															
J																															
J																															
A																															
S																															
O																															
N																															
D																															

Figura 3. Modelo No.4 de la Instrucción No.1

(Consejo de Ministros, y otros, 2012)

1.1.2 Portabilidad

La portabilidad, según la Norma ISO/IECE 9126-1:2000 es, dentro del modelo para la calidad interna y externa una de las seis características que categorizan los atributos de calidad de un producto y se define como la capacidad que tiene un producto de software para ser transferido o migrado de un entorno a otro, incluyendo entornos organizacionales, de hardware o de software. Esta característica está compuesta por las siguientes subcaracterísticas:

Adaptabilidad: Capacidad del producto software para ser adaptado a diferentes entornos, sin aplicar acciones o mecanismos distintos de aquellos proporcionados para este propósito por el propio software.

Instabilidad: Capacidad del producto software para ser instalado en un cierto entorno.

Coexistencia: Capacidad del producto software para coexistir con otro software independiente, en un entorno común, compartiendo recursos comunes.

Capacidad para reemplazar: Capacidad del producto software para ser usado en lugar de otro producto software, para el mismo propósito, en el mismo entorno.

Cumplimiento de la portabilidad: Capacidad del producto software para adherirse a normas o convenciones relacionadas con la portabilidad. (Norma ISO/IECE 9126, 2000).

Sin embargo, la Norma ISO 25000 del 2004 la define como la capacidad del producto o componente de ser transferido de forma efectiva y eficiente de un entorno hardware, software, operacional o de utilización a otro. Esta característica se subdivide a su vez en las siguientes subcaracterísticas:

Adaptabilidad: Capacidad del producto que le permite ser adaptado de forma efectiva y eficiente a diferentes entornos determinados de hardware, software, operacionales o de uso.
Capacidad para ser instalado: Facilidad con la que el producto se puede instalar o desinstalar de forma exitosa en un determinado entorno.
Capacidad para ser reemplazado: Capacidad del producto para ser utilizado en lugar de otro producto de software determinado con el mismo propósito y el mismo entorno (Norma ISO 25000, 2004).
Por otro lado, según Paul Clements, Rick Kazman y Mark Klein la portabilidad es "la habilidad del sistema para ser ejecutado en diferentes ambientes de computación. Estos ambientes pueden ser hardware, software o una combinación de los dos" (Clements, y otros, 2001)

Basándose en los conceptos antes citados se define la portabilidad como un atributo fundamental para la validación de la calidad de un producto de software y más específicamente como la capacidad que poseen estos de ser utilizados, transferidos, instalados y/o migrados de un entorno o ambiente computacional (ya sea de hardware o de software), organizacional, operacional o de uso a otros, de forma eficiente y efectiva logrando una adaptación eficaz y mantenible en el tiempo.

1.1.3 Servicios REST

El término REST (Representational State Transfer) se originó en el año 2000, descrito en la tesis de Roy Fielding, padre de la especificación HTTP. Un servicio REST no es una arquitectura software, sino un conjunto de restricciones con las que podemos crear un estilo de arquitectura de software, la cual podremos usar para crear aplicaciones web respetando HTTP.
Según Fielding las restricciones que definen a un sistema RESTful serían:

- **Cliente-servidor**: 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.
- **Interfaz uniforme**: Para la transferencia de datos en un sistema REST, este aplica acciones concretas [POST (crear), GET (leer y consultar), PUT (editar) y DELETE (eliminar)] sobre los recursos, siempre y cuando estén identificados con una URI. Esto facilita la existencia de una interfaz uniforme que sistematiza el proceso con la información.
- **Sistema de capas**: Arquitectura jerárquica entre los componentes. Cada una de estas capas lleva a cabo una funcionalidad dentro del sistema REST. Esto ayuda a mejorar la escalabilidad, el rendimiento y la seguridad.

En la actualidad la mayoría de las empresas (Facebook (Facebook, 2019), Google (Android Developers, 2018), WordPress (Wordpress, 2019), entre otras) utilizan API REST para crear servicios. Esto se debe a que es un estándar lógico y eficiente para la creación de servicios web. Estos sistemas están caracterizados por el uso de hipermedios (término que en el ámbito de las páginas web define el conjunto de procedimientos para crear contenidos que contengan texto, imagen, video, audio y otros métodos de información) para permitir al usuario navegar por los distintos recursos de una API REST a través de enlaces HTML. Uno de los aspectos más importantes a tener en cuenta al crear un servicio o API REST no es el lenguaje en el que es implementado, sino que las respuestas de sus peticiones se hagan en XML o JSON, ya que son los lenguajes de intercambio más utilizados.

Diseñar un servicio Web basado en REST

1. Identificar todas las entidades conceptuales que se desean exponer como servicio.
2. Crear una URL para cada recurso. Los recursos deberían ser nombres no verbos (acciones).
3. Categorizar los recursos si los clientes pueden obtener una representación del recurso o si pueden modificarlo. Para el primero, debemos hacer los recursos accesibles utilizando un HTTP GET. Para el último, se debe hacer los recursos accesibles mediante HTTP POST, PUT y/o DELETE.
4. Todos los recursos accesibles mediante GET no deben tener efectos secundarios. Es decir, los recursos deben devolver la representación del recurso. Por tanto, invocar al recurso no debe ser resultado de modificarlo.
5. Ninguna representación debe estar aislada. Es decir, es recomendable poner hipervínculos dentro de la representación de un recurso para permitir a los clientes obtener más información.
6. Especificar el formato de los datos de respuesta mediante un esquema (DTD, W3C Schema). Para los servicios que requieren un POST o un PUT se recomienda proporcionar un esquema para especificar el formato de la respuesta.
7. Describir como el servicio va a ser invocado, mediante un documento WSDL/WADL o simplemente HTML. (Silega Martínez, y otros, 2017)

1.2 Sistemas operativos móviles

Los dispositivos móviles en los últimos años han tenido un gran auge, esto conlleva a una amplia repercusión en casi todos los aspectos de nuestra vida: relaciones personales y profesionales, ocio y consumo, y como no, las actividades económicas y empresariales. Los dispositivos en sí, y las tecnologías que los dotan de funcionalidades de todo tipo, evolucionan a vertiginosa velocidad, y generan en la sociedad nuevas tendencias y modelos de comportamiento que cada vez se hacen más importantes para todo tipo de empresas, pues pueden lograr o perder grandes oportunidades de negocio si pueden o no adaptarse a ellos. Dentro de los Sistemas Operativos (SO) más utilizados

en dispositivos móviles se encuentran Android, iOS y Windows Phone. A continuación, se muestra en la gráfica siguiente un estudio de la cuota de mercado del SO móvil en Cuba:

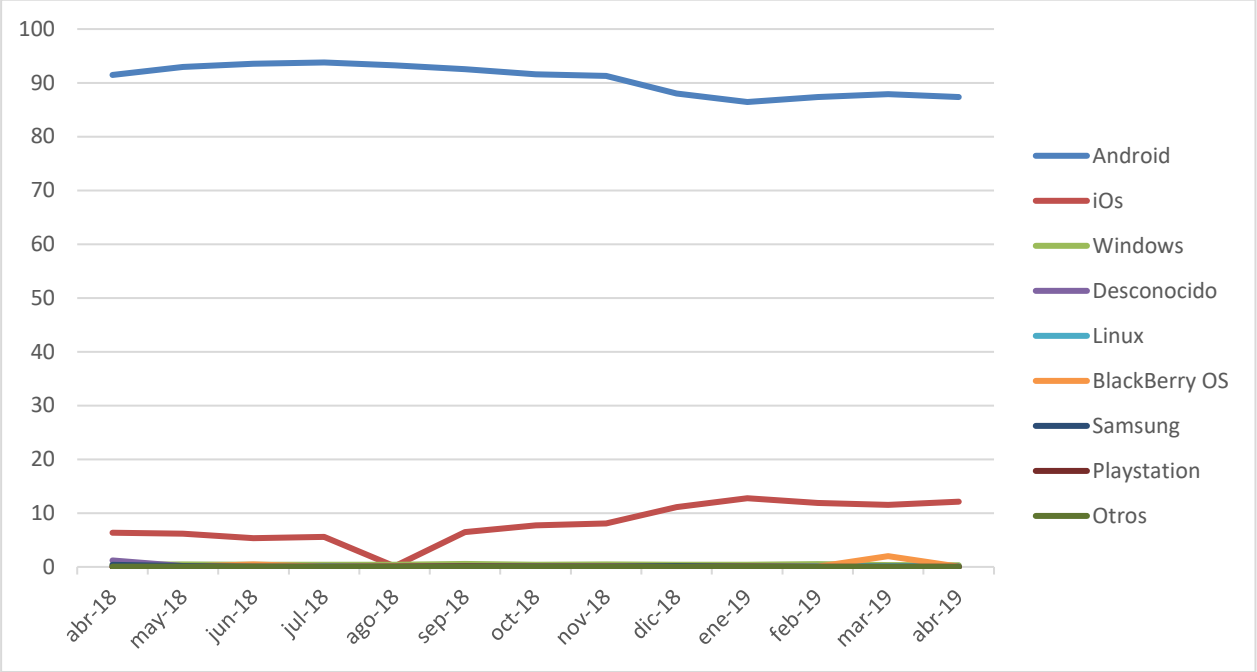

Figura 4. Cuota del mercado de Android (StatCounter, 2019)

Se selecciona Android como sistema operativo para el desarrollo de la solución propuesta considerando que es el más utilizado en Cuba según el estudio realizado de la cuota de mercado del SO móvil, como se ilustra en la Figura 4. Además, es una plataforma de código abierto y las herramientas y tecnologías oficiales que se utilizan para el desarrollo son libres.

1.3 Notificaciones en Android

Una notificación en Android es un mensaje que se muestra fuera de la interfaz de usuario común de las aplicaciones que proporciona al usuario de recordatorios, comunicaciones de otras personas u otro tipo de información de las aplicaciones. Los usuarios pueden interactuar con la notificación para abrir su aplicación o realizar una acción directamente desde la notificación. Las notificaciones se muestran en diferentes ubicaciones y formatos, como un ícono en la barra de estado, una entrada más detallada en el buzón de notificaciones, como una insignia en el ícono de la aplicación y en elementos portátiles emparejados automáticamente. (Android Developers, 2018)

Notificaciones en la barra de estado y cajón de notificaciones.

Cuando se emiten, primero aparece como un icono en la barra de estado. Los usuarios pueden deslizarse hacia abajo en la barra de estado para abrir el cajón de notificaciones, donde pueden ver más detalles y realizar acciones con la notificación, una vez allí es posible arrastrar hacia abajo la notificación en el cajón para revelar la vista expandida, que muestra contenido adicional y botones

de acción, si se proporcionan. Estas permanecen visibles en el cajón de notificaciones hasta una aplicación o el usuario la rechace. (Android Developers, 2018)

Notificaciones de aviso o head-ups

A partir de Android 5.0, las notificaciones pueden aparecer brevemente en una ventana flotante llamada notificación de heads-up o de aviso. Este comportamiento es normalmente para notificaciones importantes que el usuario debe conocer de inmediato, y aparece solo si el dispositivo está desbloqueado. La notificación de aviso aparece en el momento en que una aplicación la emite y desaparece después de un momento, pero permanece visible en el buzón de notificaciones. (Android Developers, 2018)

Notificaciones de pantalla bloqueada

A partir de Android 5.0, las notificaciones pueden aparecer en la pantalla de bloqueo. Donde programáticamente se puede establecer el nivel de detalle visible en las notificaciones publicadas en una pantalla de bloqueo segura, o incluso si la notificación se mostrará por completo en la pantalla de bloqueo. Los usuarios pueden usar la configuración del sistema para elegir el nivel de detalle visible en las notificaciones de la pantalla de bloqueo, incluida la opción para desactivar todas las notificaciones de la pantalla de bloqueo. A partir de Android 8.0, los usuarios pueden elegir deshabilitar o habilitar las notificaciones de bloqueo de pantalla para cada canal de notificación. (Android Developers, 2018)

Notificaciones como insignias en el icono de la aplicación

En los launchers[4] soportados en dispositivos que ejecutan Android 8.0 y superior, los iconos de la aplicación indican nuevas notificaciones con una "insignia" de color (también conocida como "punto de notificación") en el ícono del iniciador de la aplicación correspondiente. Los usuarios pueden presionar prolongadamente el ícono de una aplicación para ver las notificaciones de esa aplicación. Luego, pueden descartar o actuar sobre las notificaciones de ese menú, similar al cajón de notificaciones. (Android Developers, 2018)

La aplicación que responde a la problemática planteada anteriormente maneja información que es sensible y de vital importancia para las empresas y para los individuos que las conforman y estas están en constante cambio, actualización y monitoreo. Es con esta premisa que se decide utilizar todos los tipos de notificaciones antes descritas, para que el usuario pueda accederlas y/o visualizarlas en cualquier modo en que se encuentre su dispositivo móvil, y reaccionar ante ellas en tiempo real logrando de esta manera una mejor experiencia de usuario.

[4] Parte de la interfaz de usuario de Android que permite a los usuarios personalizar la pantalla de inicio,las aplicaciones móviles, hacer llamadas telefónicas y realizar otras tareas.

1.4 Aplicaciones de calendario existentes

Aplicaciones de Calendarios

En el mundo existen aplicaciones móviles para la gestión de actividades o eventos en un calendario con el objetivo de organizar mejor el trabajo y mejorar la productividad. A continuación, se exponen algunas de las aplicaciones que se analizaron para obtener elementos que ayuden a guiar el diseño de la interfaz visual, así como la distribución de la información y las funcionalidades principales de una aplicación de este tipo. Para su selección se utilizaron las estadísticas de calificaciones y bifurcaciones tanto de GitHub[5] como de Google Play[6].

1.4.1 Google Calendar

Es la herramienta móvil de calendario oficial de Google, permite anotar cualquier cita o compromiso en un calendario virtual, para recibir un aviso cuando se aproxime la fecha elegida. Lo más interesante como herramienta es:

- Posee varias vistas: mes, semana o día
- Permite la sincronización automática con tus eventos de Gmail[7].
- Posibilidad de crear tareas con sus recordatorios
- Permite añadir objetivos y te mostrará cómo y cuándo cumplirlos.
- Se sincroniza con todos los calendarios que se posean en el dispositivo.

1.4.2 Etar Calendar

Aplicación de código abierto que está disponible en la tienda de Google. Está diseñado como un calendario de código abierto, con algunas características clave que se incluyen a continuación:

- Posee una interfaz Material Design[8], compatible con Android 4.1 en adelante.
- Permite la sincronización CalDav con otros calendarios.
- No contiene anuncios.
- Es completamente libre y de código abierto.

1.4.3 Simple Calendar

Es un simple calendario para dispositivos con Android con sincronización opcional con Google y otros calendarios, en donde se pueden crear fácilmente eventos que se repitan y configurar recordatorios o que te muestre los días de la semana.

[5] Mayor servicio de alojamiento a nivel global que ofrece a los desarrolladores repositorios de software usando el sistema de control de versiones Git.
[6] Plataforma oficial de distribución digital de aplicaciones móviles para los dispositivos con sistema operativo Android.
[7] Cliente de correo desarrollado por Google.
[8] Lenguaje visual y patrón de diseño desarrollado por Google que sintetiza los principios clásicos del diseño. Es adaptado por Android desde la versión 5.0.

- Contiene un widget 4x4 redimensionable donde puedes personalizar el color del texto, así como la transparencia y el color del fondo.
- No contiene anuncios ni permisos innecesarios. Es completamente de código abierto y permite personalizar los colores.
- El permiso de almacenamiento sólo se necesita para exportar o importar eventos desde archivos con extensión .ics.
- El permiso de contactos se usa sólo para importar los cumpleaños de los contactos.

Selección del Calendario a utilizar en la propuesta de Calendario

Se realizó un análisis de los calendarios existentes a nivel nacional e internacional en correspondencia con sus características y estructura, creados para aplicaciones Android, donde se propone para darle solución a la problemática planteada, personalizar un calendario ya desarrollado que permita gestionar las actividades que implementa el SIPAC, para ahorrar tiempo de producción. Con este objetivo se tuvieron en cuenta los siguientes criterios para seleccionar el más adecuado e integrarlo a la solución:

- **El código fuente debe ser libre:** Es decir que se posible accederlo, modificarlo y/o eliminarlo sin ningún tipo de restricción de licencia.
- **Fácil de mantener en el tiempo:** El calendario debe tener una vasta documentación sobre su uso y puesta en práctica para un mejor aprendizaje de su funcionamiento, puesta en práctica y corrección de errores.
- **Permitir trabajar tanto offline como online:** La aplicación debe poder funcionar normalmente tanto como si el dispositivo se encuentra conectado a una red como si esta desconectado, permitiendo la sincronización de las actividades al cambiar de estado.
- **Poseer una interfaz amigable preferiblemente Material Design**: Este patrón de diseño, desarrollado por Google permite disminuir la curva de aprendizaje en el uso de aplicaciones Android.
- **Permitir agregar campos personalizados**: Debe permitir modificar y/o eliminar los campos que son estándar en los eventos de los calendarios sustituyéndolos o adaptándolos según los requisitos del sistema y también añadir nuevos.

Después de efectuado el estudio de todas las aplicaciones de calendario anteriormente mencionados, se decide utilizar en la implementación de la solución el calendario Simple Calendar, siendo este el único que cumple con todos los requisitos expuestos.

1.5 Tecnologías y herramientas

1.5.1 Sistema Operativo Android

Android es una solución completa de software de código libre para teléfonos y dispositivos móviles. Es un paquete que engloba un sistema operativo, un "runtime" de ejecución basado en Java, un conjunto de librerías de bajo y medio nivel y un conjunto inicial de aplicaciones destinadas al usuario

final (todas ellas desarrolladas en Java). Android se distribuye bajo una licencia libre permisiva (Apache) que permite la integración con soluciones de código propietario.

Surge como resultado de la Open Handset Alliance (OHA), alianza comercial de 84 compañías que se dedica a desarrollar estándares abiertos para dispositivos. Actualmente, el desarrollo viene avalado principalmente por Google (tras la compra de Android Inc. en 2005) y entre otras compañías de software (Ebay, LivingImage), operadores (Telefónica, Vodafone, T-Mobile), fabricantes de móviles (Motorola, Samsumg, Acer, LG, HTC) y de Hardware (nVidia, Intel o Texas Instruments).(Camarero, y otros, 2009)

Arquitectura de Android

Android es una pila de software de código abierto basado en Linux creada para una variedad amplia de dispositivos y factores de forma. En el siguiente diagrama se muestran los componentes principales de la plataforma Android.

Figura 5. Arquitectura del Sistema Operativo Android (Imagen de elaboración propia).

Kernel de Linux: La base de la plataforma Android es el kernel de Linux. El uso de este kernel permite que Android aproveche funciones de seguridad claves y, al mismo tiempo, permite a los fabricantes de dispositivos desarrollar controladores de hardware para un kernel conocido.

Capa de abstracción de hardware (HAL): Brinda interfaces estándares que exponen las capacidades de hardware del dispositivo al framework de Java API de nivel más alto. La HAL consiste en varios módulos de biblioteca y cada uno de estos implementa una interfaz para un tipo específico de componente de hardware, como el módulo de la cámara o de Bluetooth. Cuando el framework de una API realiza una llamada para acceder al hardware del dispositivo, el sistema Android carga el módulo de biblioteca para el componente de hardware en cuestión.

Runtime de Android: Para los dispositivos con Android 5.0 o versiones posteriores, cada app ejecuta sus propios procesos con sus propias instancias del tiempo de ejecución de Android (ART). El ART está escrito para ejecutar varias máquinas virtuales en dispositivos de memoria baja

ejecutando archivos DEX, un formato de código de bytes diseñado especialmente para Android y optimizado para ocupar un espacio de memoria mínimo. Crea cadenas de herramientas, como Jack, y compila fuentes de Java en código de bytes DEX que se pueden ejecutar en la plataforma Android.

Antes de Android 5.0, Dalvik era el tiempo de ejecución del sistema operativo. Si tu app se ejecuta bien en el ART, también debe funcionar en Dalvik, pero es posible que no suceda lo contrario. En Android también se incluye un conjunto de bibliotecas de tiempo de ejecución centrales que proporcionan la mayor parte de la funcionalidad del lenguaje de programación Java; se incluyen algunas funciones del lenguaje Java en su versión 8.0, que el framework de la API de Java usa.

Bibliotecas C/C++ nativas: Muchos componentes y servicios centrales del sistema Android, como el ART y la HAL, se basan en código nativo que requiere bibliotecas nativas escritas en C y C++. La plataforma Android proporciona la API del framework de Java para exponer la funcionalidad de algunas de estas bibliotecas nativas a las aplicaciones (apps). Si se desarrolla una app que requiere C o C++, puedes usar el Native Development Kit (NDK) de Android para acceder a algunas de estas bibliotecas de plataformas nativas directamente desde tu código nativo.

Framework de la API de Java: Todo el conjunto de funciones del SO Android está disponible mediante API escritas en el lenguaje Java. Estas API son los cimientos que se necesitan para crear apps de Android simplificando la reutilización de componentes del sistema y servicios centrales y modulares, como los siguientes:

- Un sistema de vista enriquecido y extensible que puedes usar para compilar la interfaz de usuario de una app; se incluyen listas, cuadrículas, cuadros de texto, botones e incluso un navegador web integrable.
- Un administrador de recursos que te brinda acceso a recursos sin código, como strings localizadas, gráficos y archivos de diseño.
- Un administrador de notificaciones que permite que todas las apps muestren alertas personalizadas en la barra de estado.
- Un administrador de actividad que administra el ciclo de vida de las apps y proporciona una pila de retroceso de navegación común.
- Proveedores de contenido que permiten que las apps accedan a datos desde otras apps, como la app de contactos, o compartan sus propios datos.

Los desarrolladores tienen acceso total a las mismas APIs del framework que usan las apps del sistema Android.

Aplicaciones del sistema: En Android se incluye un conjunto de apps centrales para correo electrónico, mensajería SMS, calendarios, navegación en Internet y contactos, entre otros elementos. Las apps incluidas en la plataforma no tienen un estado especial entre las apps que el usuario elije instalar; por ello, una app externa se puede convertir en el navegador web, el sistema

de mensajería SMS o, incluso, el teclado predeterminado del usuario (existen algunas excepciones, como la app Configuración del sistema). Las apps del sistema funcionan como apps para los usuarios y brindan capacidades claves a las cuales los desarrolladores pueden acceder desde sus propias apps. Por ejemplo, si en tu app se intenta entregar un mensaje SMS, no es necesario que compiles esa funcionalidad tú mismo; como alternativa, puedes invocar la app de SMS que ya está instalada para entregar un mensaje al receptor que especifiques. (Android Developers, 2018).

1.5.2 Android Studio

Android Studio es el entorno de desarrollo integrado (IDE) oficial para el desarrollo de aplicaciones para Android y se basa en IntelliJ IDEA. Además del potente editor de códigos y las herramientas para desarrolladores de IntelliJ, Android Studio ofrece más funciones que aumentan la productividad durante la compilación de apps para Android, como las siguientes:

- Un sistema de compilación basado en Gradle flexible.
- Un emulador rápido con varias funciones.
- Un entorno unificado en el que puedes realizar desarrollos para todos los dispositivos Android.
- Instant Run para aplicar cambios mientras tu app se ejecuta sin la necesidad de compilar un nuevo APK.
- Integración de plantillas de código y GitHub para ayudarte a compilar funciones comunes de las apps e importar ejemplos de código.
- Gran cantidad de herramientas y frameworks de prueba.
- Herramientas Lint para detectar problemas de rendimiento, usabilidad, compatibilidad de versión, etc.
- Compatibilidad con C++ y NDK.
- Soporte incorporado para Google Cloud Platform, lo que facilita la integración de Google Cloud Messaging y App Engine (Android Developers, 2018)

1.5.3 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:

Compatibilidad: Totalmente compatible con JDK 6, lo que garantiza que las aplicaciones de Kotlin puedan ejecutarse en dispositivos Android antiguos sin problemas. Las herramientas de Kotlin son totalmente compatibles con Android Studio y son compatibles con el sistema de compilación de Android.

Rendimiento: una aplicación Kotlin se ejecuta tan rápido como una Java equivalente, gracias a una estructura de código de byte muy similar. Con el soporte de Kotlin para las funciones en línea, el código que utiliza lambdas a menudo se ejecuta incluso más rápido que el mismo código escrito en Java.

Interoperabilidad: Es 100% interoperable con Java, lo que permite utilizar todas las bibliotecas de Android existentes en una aplicación Kotlin. Esto incluye el procesamiento de anotaciones, por lo que el enlace de datos y Dagger también funcionan.

Perfil: Posee una biblioteca de tiempo de ejecución muy compacta, que se puede reducir aún más mediante el uso de ProGuard.

Tiempo de compilación: Kotlin 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.

Curva de aprendizaje: para un desarrollador de Java, comenzar con este lenguaje es muy fácil. El convertidor automatizado de Java a Kotlin incluido en el complemento Kotlin ayuda con los primeros pasos.

1.5.4 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. Entre sus principales características se encuentran:

Altamente personalizable: Es modelado de una manera que es personalizable y extensible de la manera más fundamental.

Rápido: Completa las tareas rápidamente, reutilizando los resultados de ejecuciones anteriores, procesando solo las entradas que han cambiado y ejecutando tareas en paralelo.

Potente: Herramienta de compilación oficial para Android y es compatible con muchos lenguajes y tecnologías populares. (Gradle Inc., 2018)

1.5.5 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).

1.5.6 SQLite

SQLite es una biblioteca en proceso que implementa un motor de base de datos SQL transaccional independiente, sin servidor y de configuración cero. El código para SQLite es de dominio público y, por lo tanto, es gratuito para cualquier uso, comercial o privado. SQLite es la base de datos más implementada del mundo, incluyendo varios proyectos de alto perfil. SQLite es un motor de base de datos SQL incorporado. A diferencia de la mayoría de las otras bases de datos SQL, SQLite no tiene un proceso de servidor por separado. SQLite lee y escribe directamente en archivos de disco ordinarios. Una base de datos SQL completa con múltiples tablas, índices, disparadores y vistas, está contenida en un solo archivo de disco. El formato de archivo de la base de datos es multiplataforma: puede copiar libremente una base de datos entre sistemas de 32 bits y de 64 bits o entre arquitecturas big-endian y little-endian. Estas características hacen que SQLite sea una opción popular como formato de archivo de aplicación (About SQLite, 2018).

1.5.7 Librería ROOM

La librería ROOM lanzada por Google en 2017 proporciona una capa de abstracción o arquitectura más segura y robusta que permite el acceso sin problemas a las bases de datos aprovechando la librería SQLite. ROOM posee 3 componentes principales:

- Base de datos: Comprende el contenedor de la base de datos y sirve como punto de acceso principal para la conexión subyacente a los datos relacionales persistentes en las aplicaciones.
- Entidad: representa una tabla dentro de la base de datos.
- DAO: contiene los métodos utilizados para acceder a la base de datos. (Android Developers, 2018)

1.5.8 Herramienta CASE (Computer Aided Software Engineering por sus siglas en inglés)

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, 2018).

Visual Paradigm v15.1

Constituye una herramienta CASE profesional, que utiliza ¨UML¨ cómo lenguaje de modelado y que 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, Boyd, & Copley, 2017).

Características:

- Soporta todos los diagramas de entidad-relación.
- Genera 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.
- Se integra con herramientas Java, como son: Eclipse/IBM, NetBeansIDE, entre otras. (Visual Paradigm , 2018)

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. (Pressman, 2010)

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.

1.6 Metodología de desarrollo

Las metodologías ágiles son una excelente alternativa para guiar proyectos de desarrollo de aplicaciones para dispositivos móviles, gracias a la gran facilidad de adaptación que poseen, ya que algunos de los factores que caracterizan el entorno móvil son la rapidez con la que se actualizan o cambian cambiando las versiones de los sistemas operativos móviles, la aparición de nuevas prestaciones de hardware, la reñida competencia de las empresas y comunidades de desarrollo por apropiarse del mercado, lo que conduce a la tendencia de desarrollar aplicaciones en cortos lapsos de tiempo y en la mayoría de casos sacrificando la calidad del producto, precisamente por no seguir una metodología o técnicas de desarrollo apropiadas y es por eso que en la mayoría de los casos, sus esfuerzos terminan por no dar los frutos esperados. Dentro del mundo metodologías diseñadas específicamente para aplicaciones móviles se destacan Mobile-D una de las pocas metodologías con certificación CMMI que fue desarrollada por científicos del Instituto de Investigación Finlandés (VTT) y Proceso de desarrollo móvil en espiral desarrollado por Vahid Rahimian y Raman Ramsin. A continuación, se refieren algunas de las propiedades que presentan estas metodologías que son aplicables al desarrollo móvil:

Tabla 1. Tabla comparativa entre las características básicas o bases (home ground) ágiles con el desarrollo de software para dispositivos móviles (Abrahamsson, 2005)

Características ágiles	**Motivación lógica**	**Aplicable a tecnologías móviles**

Alta volatilidad del entorno	Debido a la frecuencia en el cambio que sufren los requisitos, se tienen menos necesidad de diseño y planificación inicial y mayor necesidad de desarrollos incrementables e iterativos.	Alta incertidumbre, entornos dinámicos, cientos de nuevos terminales cada año.
Equipos de desarrollos pequeños.	Capacidad de reacción más rápida, trabajo basado en la compartición de la información menos documentos.	La mayor parte de los proyectos de desarrollo móvil son de equipos pequeños.
Cliente identificable	Conocimiento de los intereses de los clientes.	Potencialmente, hay un número ilimitado de usuarios finales, pero los clientes son fáciles de identificar.
Entornos de desarrollo orientados a objetos	Mayoría de las herramientas de desarrollo ágil existen bajo plataformas orientadas a objetos.	Ejemplo, Java y C++ se usan, en algunos problemas en herramientas como refactorizaciones o primeros test.
Sistemas pequeños	Menos necesidad de diseño inicial.	Las aplicaciones, aunque variables en tamaño, no suelen superar las 10.000 líneas de código.
Ciclos de desarrollo cortos	Propósito de realimentación rápida.	Periodos de desarrollo de 1 a 6 meses.

Se decidió utilizar metodologías ágiles en el desarrollo de la aplicación propuesta para realizar las operaciones sobre los módulos de Autenticación y Calendario de SIPAC, por las características que presentan este tipo de desarrollo de software antes expuestas en el desarrollo de aplicaciones móviles. Se presentan tres propuestas en el desarrollo de la tesis, analizando los elementos como:

1. Ciclo de desarrollo.
2. Fases e iteraciones.
3. Productos de trabajo generados por cada una de las fases.
4. Cantidad de iteraciones de cada una de las fases.

Seleccionando la Variación AUP para la UCI, para guiar el proceso de desarrollo de software de la propuesta solución, por la solicitud realizada por el Centro de Informatización de Entidades (CEIGE), en correspondencia con los lineamientos de desarrollo de software que posee la universidad.

1.6.1 Modelo híbrido para el desarrollo ágil

La metodología propuesta por Vahid Rahimian y Raman Ramsin en el documento Designing an Agile Methodology for Mobile Software Development: A Hybrid Method Engineering Approach, se apoya en una combinación del desarrollo adaptativo de software (Adaptive Software Development, ASD) y el diseño de nuevos productos (New Product Development, NPD). En la primera iteración, se divide la fase de análisis con la intención de mitigar riesgos de desarrollo, de la misma forma, el diseño también se segmenta para introducir algo de diseño basado en arquitectura. La implementación y las pruebas sin embargo fusionan introduciendo conceptos de desarrollo orientado a pruebas (Test-Driven Development, TDD). Aparece además una fase de comercialización, incidiendo en el sesgo hacia el desarrollo de producto que se imponen en el escenario del desarrollo de aplicaciones para plataformas móviles. Desde el punto de vista metodológico, los autores afirman haberse apoyado en metamodelos como SPEN (Software Processes Engineering Metamodel, soportado por el entorno de desarrollo de Eclipse, por ejemplo) y OPF, (Open Processes Framework), así como conceptos genéricos de ciclos de vida orientados a objetos como OPSP (Object Oriented Software Processes)

La segunda iteración, realiza una integración de ciertas partes de los modelos NPD, añadiendo la generación de ideas en el inicio del ciclo y una prueba de mercado antes de lanzar la fase de comercialización.

La tercera iteración, integra directamente el “motor de desarrollo” de los métodos de desarrollo adaptativo (ASD) muy orientado al aseguramiento de la calidad de los procesos de desarrollo.

Pensando en aquella propiedad “ideal” de disponer de la arquitectura física en una fase temprana del proceso, en la cuarta iteración se añaden elementos de prototipado; se refina, además, la fase de iniciación del proyecto, sobre la base del mismo elemento de los procesos adaptativos. (Camarero, y otros, 2009)

1.6.2 Mobile –D

Es un modelo propuesto por Pekka Abrahamsson y su equipo del VTT (Valtion Teknillinen Tutkimuskeskus), en inglés Technical Research Centre of Finland) en Filandia. El ciclo del proyecto se divide en cinco fases: exploración, inicialización, producción, estabilización y prueba del sistema, como se muestra en la Figura 6. En general, todas las fases con la excepción de la primera fase exploratoria contienen tres días de desarrollo distintos: planificación, trabajo y liberación.

Figura 6. Ciclo de Desarrollo Mobile-D

La fase de exploración, siendo ligeramente diferente del resto del proceso de producción, se dedica al establecimiento de un plan de proyecto y los conceptos básicos.

Durante la fase de inicialización, los desarrolladores preparan e identifican todos los recursos necesarios. Se preparan los planes para las siguientes fases y se establece el entorno técnico (incluyendo el entrenamiento al desarrollo ágil se centra fundamentalmente en esta fase, en la investigación de la línea arquitectónica). Los desarrolladores analizan el conocimiento y los patrones arquitectónicos utilizados en la empresa y los relacionan con el proyecto actual. Se agregan observaciones, se identifican similitudes y se extraen soluciones viables para su aplicación en el proyecto.

En la fase de producción se repite la programación de tres días (planificación-trabajo-liberación) se repite iterativamente hasta implementar todas las funcionalidades. Primero se planifica la iteración de trabajo en términos de requisitos y tareas a realizar. Se preparan las pruebas de iteración. Las tareas se llevan a cabo durante el día de trabajo, desarrollando e integrando el código con los repositorios existentes. Durante el último día se realiza la integración del sistema.

En la fase de estabilización, se llevan a cabo las últimas acciones de integración para asegurar que el sistema completo funciona correctamente. Esta será la fase más importante en los proyectos multiequipo con diferentes subsistemas desarrollados por equipos distintos. En esta fase, los desarrolladores realizarán tareas similares a las que debían desarrollar en la fase de “producción”, aunque en este caso todo el esfuerzo se dirige a la integración del sistema.

La última fase (prueba y reparación del sistema) tiene como objetivo la disponibilidad de una versión estable y plenamente funcional del sistema. El producto terminado e integrado se prueba con los requisitos de cliente y se eliminan todos los defectos encontrados (Camarero, y otros, 2009).

1.6.3 Variación de AUP para 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"

De las 4 fases que propone AUP (Inicio, Elaboración, Construcción, Transición) esta variación mantiene la fase de Inicio, pero modifica su objetivo, y las restantes 3 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 una tabla con los objetivos específicos de cada fase: (Sánchez, 2015)

Tabla 2. Descripción de las Fases Variación AUP-UCI. (Sánchez, 2015)

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 Transición	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.
	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 3 disciplinas:

Pruebas Internas, de Liberación y Aceptación. Las restantes 3 disciplinas de AUP asociadas a la parte de gestión se cubren con las áreas de procesos que define CMMIDEV v1.3 para el nivel 2, serían CM (Gestión de la configuración), PP (Planeación de proyecto) y PMC (Monitoreo y control de proyecto). (Sánchez, 2015)

1.6.4 Escenario para la disciplina de requisito

Luego de definir como metodología de desarrollo la Variación AUP-UCI y, 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.

Después de evaluar el negocio se obtuvo como resultado procesos muy complejos e interrelacionados entre sí, 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 número tres en el cual se modelan el negocio mediante DPN y MC de manera que permita encapsular los requisitos del negocio como DRP. (Sánchez, 2015)

1.7 Conclusiones parciales

Al culminar este capítulo se arribaron a las siguientes conclusiones lográndose la construcción del marco teórico conceptual de la investigación:

- Se exponen los principales conceptos de la investigación, principios y clasificación de las actividades que permitió un mejor entendimiento del negocio.
- Se abordó sobre las características y restricciones de los servicios REST, las tecnologías, lenguajes de programación y herramientas que se utilizaron para la implementación de la solución. Además de la metodología de desarrollo, elementos que se definieron para la solución de la investigación.

CAPÍTULO 2. DESCRIPCIÓN DE LA SOLUCIÓN PROPUESTA

Introducción

En el presente capítulo se realiza una descripción de la solución propuesta, a partir de las disciplinas requisitos, análisis, diseño e implementación definidas por la metodología y escenario seleccionados. 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

Modelado de Negocio

El Modelado del Negocio es la disciplina destinada a comprender los procesos de negocio de una organización. Se comprende cómo funciona el negocio que se desea informatizar para tener garantías de que el software desarrollado va a cumplir su propósito (Sánchez, 2015). Para modelar el negocio de la solución se propone la variante de modelo conceptual según lo que plantea el escenario seleccionado de la metodología AUP-UCI.

2.1 Modelo conceptual

Un modelo conceptual explica (a sus creadores) los conceptos significativos en un dominio del problema; es el artefacto más importante a crear durante el análisis orientado a objetos. Es una representación de conceptos en un dominio del problema (Larman, 2004).

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, observación y anexo.

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 observación está representado por denominar una nota, comentario o una indicación en un escrito para de esta manera explicar o precisar un punto dudoso o ambiguo.

8. 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.
9. El concepto de anexo está representado por un fichero que se anexa a un elemento de la planificación, puede contener cualquier objeto digitalizado de los siguientes: documentos (PDF, EXCEL, DOC) o imágenes.

En la figura que se muestra a continuación se presentan los principales conceptos antes descritos y las relaciones que existen entre ellos.

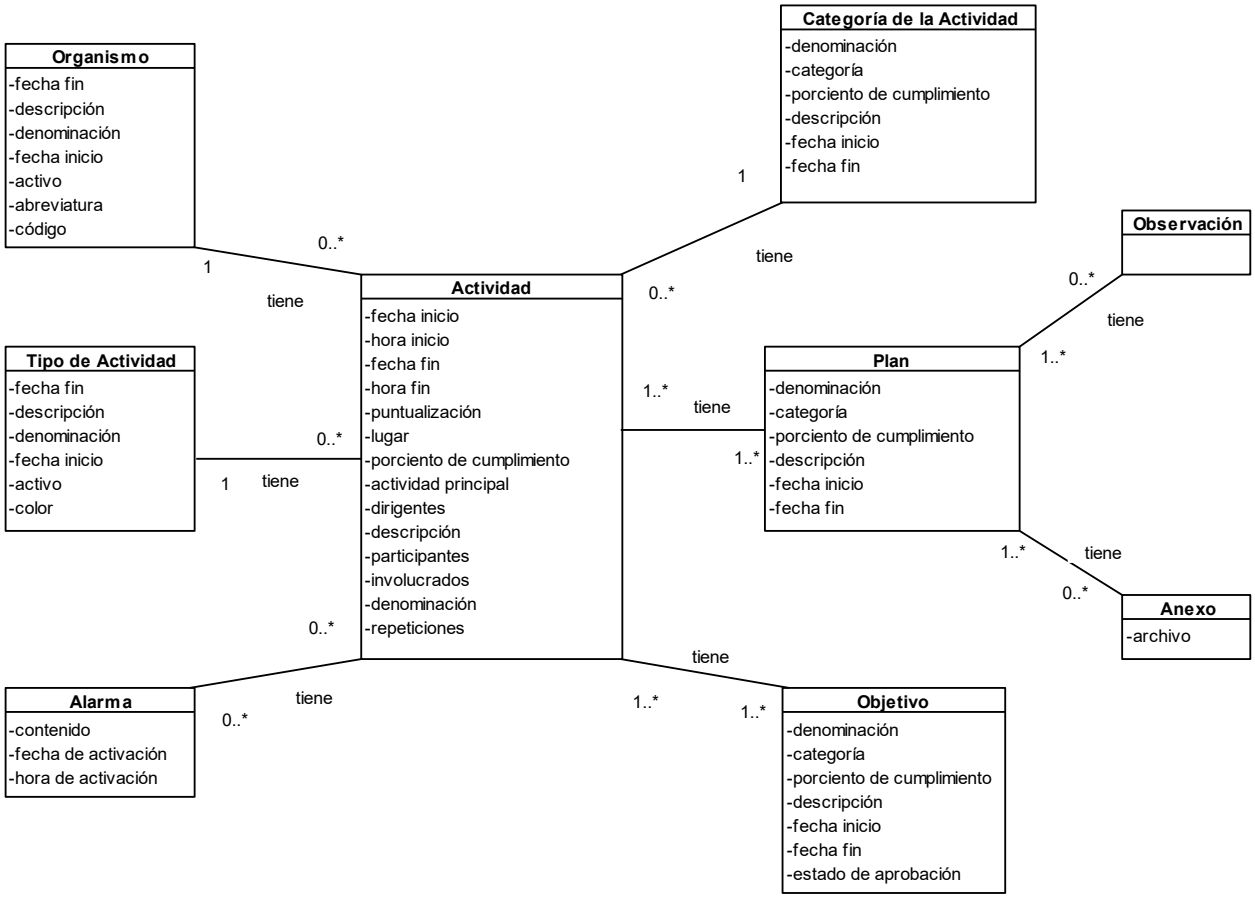

Figura 7. Modelo conceptual del módulo de actividades.

2.2 Disciplina Requisitos

El esfuerzo principal en la disciplina Requisitos es desarrollar un modelo del sistema que se va a construir y comprende la administración de los requisitos funcionales y no funcionales del producto (Sánchez, 2015).

Un requisito de software no es más que 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. (IEEE, 2014)

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 7).

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 Especificación de requisitos

Los requisitos funcionales (RF) de un sistema, son aquellos que especifican o definen una función que un sistema o componente debe cumplir. (IEEE, 2014)

2.2.2.1 Requisitos funcionales

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

Tabla 3. Requisitos funcionales del sistema.

No.	Nombre	Descripción	Prioridad	Complejidad
RF1	Autenticar usuario	Permite asociar una solicitud entrante con un conjunto de credenciales de identificación como son el usuario y la contraseña.	Alta	Baja
RF2	Adicionar actividad	Registra una actividad en el sistema.	Alta	Alta
RF3	Modificar actividad	Modifica una actividad especificada con la información necesaria.	Alta	Media
RF4	Eliminar actividad	Elimina una actividad en el sistema.	Alta	Baja

RF5	Listar actividades	Muestra todas las actividades existentes.	Alta	Alta
RF6	Buscar actividad	Muestra las actividades que coincidan con el criterio de búsqueda.	Media	Baja
RF7	Ver datos de la actividad	Muestra los datos de una actividad seleccionada por el usuario.	Baja	Media
RF8	Asignar involucrado	Permite asignar a una actividad los usuarios involucrados.	Media	Baja
RF9	Adicionar relación en actividades	Permite relacionar las actividades con los planes, objetivos, plan-ARC y los FIP .	Alta	Media
RF10	Eliminar relación de actividades	Permite eliminar la relación de las actividades con los planes, objetivos, plan-ARC y los FIP.	Alta	Alta
RF11	Listar relaciones	Muestra las relaciones registradas de las actividades.	Media	Baja
RF12	Insertar repetición periódica	Permite insertar una actividad de forma periódica.	Alta	Media
RF13	Modificar repetición periódica	Permite modificar una actividad de forma periódica.	Alta	Baja
RF14	Insertar repetición aleatoria	Permite insertar una actividad de forma aleatoria.	Alta	Media
RF15	Modificar repetición aleatoria	Permite modificar una actividad de forma aleatoria.	Alta	Media
RF16	Eliminar repetición	Permite eliminar las repeticiones de la actividad periódicas y aleatorias.	Alta	Alta
RF17	Listar repeticiones	Muestra las actividades que se registraron de forma periódica y aleatoria en las diferentes vistas del calendario.	Baja	Baja

En la tabla 3 anteriormente descrita la columna Número (No.) se refiere el identificador del requisito, Ejemplo: RF1 corresponde al requisito funcional número uno; la columna Nombre se corresponde con el calificativo del requisito. La prioridad representa la importancia y/o relevancia que este posee para el desarrollo y completamiento de la solución y la complejidad permite estimar el esfuerzo de implementación del requisito y contribuye a la decisión sobre la inclusión en las etapas de desarrollo

del software (Saria Preval, 2014),para su medición se utilizó el artefacto Evaluación de Requisitos (Saria Preval, 2014).

2.2.2.1.1 Descripción de requisitos por proceso

Tabla 4. Descripción textual del requisito Autenticar usuario.

Precondiciones	El usuario debe estar registrado en el sistema.
Flujo de eventos	
Flujo básico Autenticar usuario	
1.	Se muestra la ventana Autenticar usuario con los siguientes campos para que el usuario introduzca los datos: **Usuario** (Nombre que recibe el usuario en el sistema que, al ser adicionado, puede ser una combinación de letras y dígitos.) **Contraseña** (El usuario especifica una clave para la autenticación en el sistema, la clave es única para el usuario, debe tener más de 8 dígitos y no deben coincidir los caracteres con los datos: usuario, nombre, apellidos o correo)
2.	Se pulsa el botón **Autenticar**.
3.	Se verifica la autenticidad del usuario (Ver validación 1).
4.	Si los datos introducidos son correctos el usuario accede a los servicios del sistema.
5.	Se muestra un mensaje de información indicando que la acción se ha realizado satisfactoriamente.
6.	Concluye el requisito.
Postcondiciones	
1.	Se autenticó el usuario y puede acceder a los servicios del sistema que cumplan con el rol y función que tiene asignado.
Flujos alternativos	
Flujo alternativo 4.a Campos vacíos	
1.	El sistema no activa el botón Autenticar
2.	Volver al paso 2 del flujo básico.
Postcondiciones	
1.	N/A
Flujo alternativo 4.b Campos con valores incorrectos	
1.	El sistema muestra un mensaje de error “El nombre de usuario y la contraseña son incorrectos”

2.	Volver al paso 2 del flujo básico.	
Validaciones		
1.	Se validan los datos según lo establecido en el Modelo conceptual.	
Conceptos	**Identificador**	**Utilizados internamente**: Identificador
	Usuario	**Visibles en la interfaz**: Usuario
	Contraseña	**Visibles en la interfaz**: Contraseña
Requisitos especiales	N/A.	
Asuntos pendientes	N/A.	

Tabla 5. Descripción textual del requisito Adicionar actividad.

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 Calendario. El usuario selecciona un ícono en forma del signo (+) que tiene un tooltip Adicionar actividad.
Flujo de eventos	
Flujo básico Adicionar actividad	
1.	Se muestra la ventana Adicionar actividad con los siguientes campos para que el usuario introduzca los datos (Ver validación 1): **Denominación** (Nombre que recibe la actividad al ser adicionada que, puede ser una combinación de letras, dígitos y caracteres especiales) **Fecha inicio** (El usuario selecciona la fecha en que se inicia la actividad). **Fecha fin** (El usuario selecciona la fecha en la que termina la actividad). **Hora de inicio** (Permite seleccionar la hora de inicio de la actividad creada). **Hora de fin** (Permite seleccionar la hora de fin de la actividad creada). **Categoría de la actividad** (Permite seleccionar la categoría de la actividad que desea adicionar). **Tipo de actividad** (Permite seleccionar el tipo de actividad que desea adicionar) **Por puntualización** (Se selecciona si la actividad es Puntualizada o es Extraplan) **Organismo (**Se selecciona en el caso que la actividad venga de un organismo externo, si es interna el sistema por defecto toma el nombre del organismo interno)

	Lugar (Describe el lugar donde se va a efectuar dicha actividad) **Porciento de cumplimiento** (Describe hasta que porciento se ha cumplido el plan, este porciento puede estar comprendido entre 0 y 100, por defecto se crea la actividad con porciento de cumplimiento 0) **Actividad principal** (Se selecciona en caso que la actividad sea principal) **Contenido de la alarma** (Define el contenido de la alarma) **Fecha de activación de la alarma** (Permite seleccionar la fecha de activación de la alarma) **Hora de activación de la alarma** (Permite seleccionar la hora de activación de la alarma) **Involucrados** (Se seleccionan los involucrados a los cuales les debe llegar la actividad) **Participantes** (Se definen los participantes de la actividad) **Dirigentes** (Se definen los dirigentes de la actividad) **Descripción** (El usuario especifica las características que tendrá la actividad)
2.	Se presiona en el botón **Aceptar**.
3.	Se validan los datos introducidos (Ver validación 1 y 2).
4.	Si los datos introducidos son correctos se registran los datos de la actividad.
5.	Se muestra un mensaje de información indicando que la acción se ha realizado satisfactoriamente.
6.	Concluye el requisito.
Postcondiciones	
1.	Se adicionó en el sistema la actividad con el estado de creado.
2.	Se muestra la actividad en las diferentes vistas que tiene el módulo Actividades: Vista mensual, anual, diaria, semanal.
Flujos alternativos	
Flujo alternativo 4.a Campos vacíos	
1.	El sistema señala de color rojo los campos obligatorios mostrando un tooltip “Este campo es obligatorio”.
2.	Volver al paso 1 del flujo básico.
Postcondiciones	
1.	Volver al paso 1 del flujo básico.
Flujo alternativo *.a El usuario cancela la acción	
1.	Concluye el requisito.
Postcondiciones	
1.	No se registran los datos.
Validaciones	

1.	Se validan los datos según lo establecido en el Modelo conceptual.
2.	El sistema valida que la fecha de fin sea igual o mayor que la fecha de inicio.
Flujo alternativo 4.a Datos incorrectos	
1.	El sistema señala de color rojo los campos obligatorios mostrando un tooltip “Existen campos con entradas inválidas.”.
2.	Volver al paso 1 del flujo básico.
Flujo alternativo *. a En caso que seleccione la opción Aceptar en el mensaje sobre actividades coincidentes.	
1.	EL usuario presiona en el botón Aceptar si desea adicionar la actividad a pesar de la coincidencia.
Postcondiciones	
1.	Se inserta la actividad a pesar de ser una actividad coincidente.
Flujo alternativo *.a En caso que seleccione la opción Cancelar en el mensaje sobre actividades coincidentes.	
1.	El usuario presiona en el botón Cancelar para abortar la operación de Adicionar actividad.
Postcondiciones	
1.	Se cierra la interfaz Adicionar actividad.

Concepto	**Identificador**	**Utilizados internamente**: Identificador
	Denominación	**Visibles en la interfaz:** Denominación
	Fecha inicio	**Visibles en la interfaz:** Fecha inicio
	Fecha fin	**Visibles en la interfaz**: Fecha fin
	Hora de inicio	**Visibles en la interfaz**: Hora de inicio
	Hora de fin	**Visibles en la interfaz**: Hora de fin
	Categoría de la actividad	**Visibles en la interfaz**: Categoría de la actividad
	Por puntualización	**Visibles en la interfaz**: Por puntualización
	Tipo de actividad	**Visibles en la interfaz**: Tipo de actividad
	Organismo	**Visibles en la interfaz**:

		Organismo
	Lugar	**Visibles en la interfaz**: Lugar
	Porciento de cumplimiento	**Visibles en la interfaz**: Porciento de cumplimiento
	Actividad principal	**Visibles en la interfaz**: Actividad principal
	Contenido de la alarma	**Visibles en la interfaz**: Contenido de la alarma
	Fecha de activación de la alarma	**Visibles en la interfaz**: Fecha de activación de la alarma
	Hora de activación de la alarma	**Visibles en la interfaz**: Hora de activación de la alarma
	Involucrados	**Visibles en la interfaz**: Involucrados
	Participantes	**Visibles en la interfaz**: Participantes
	Dirigentes	**Visibles en la interfaz**: Dirigentes
	Descripción	**Visibles en la interfaz**: Descripción
Requisitos especiales	N/A.	
Asuntos pendientes	N/A.	

2.2.2.2 Requisitos no funcionales

Los requisitos no funcionales representan características generales y restricciones de la aplicación o sistema que se esté desarrollando. Suelen presentar dificultades en su definición dado que su conformidad o no conformidad podría ser sujeto de libre interpretación, por lo cual es recomendable acompañar su definición con criterios de aceptación que se puedan medir (Pressman, 2010).

Requisitos organizaciones: 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. La aplicación requiere una PC que funcione como servidor y tenga instalado el Sistema de Planificación de Actividades en su versión 3.0.

RN3. La aplicación requiere la publicación de los servicios REST del módulo de Calendario y de Seguridad (Autenticar Usuario), en el Sistema de Planificación de Actividades SIPAC.
RN4. La aplicación requiere que el protocolo de comunicación que se utilice sea 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 se ajusten 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.
RN8. En la ampliación se deben visualizar todos los mensajes en idioma español. La tipografía debe ser uniforme, de un tamaño adecuado.

Requisitos externos: se derivan de los factores externos del sistema y de su proceso de desarrollo (Sommerville, 2005).
Requisitos de seguridad
RN9. La aplicación garantizará la autenticidad cuando se realice una solicitud entrante con un conjunto de credenciales de identificación como son el usuario, la contraseña y dirección del servidor.
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 clase 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). A continuación, la Figura 8 representa el diagrama de clases del diseño del Requisito Autenticar usuario.

El paquete Vista comprende las clases y elementos relacionados con la interfaz de usuario. La clase LoginActivity es la encargada de decidir la forma en que se muestran los datos, la cual implementa la interfaz IManView que contiene las operaciones y construye el archivo activity_login.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 LoginActivity que se encuentra en el paquete Vista construye la clase PresentadorLogin responsable de recuperar los datos del paquete modelo, devolverlos procesados a la vista, decidir qué ocurre cuando se interactúa con la vista e implementa la interfaz ImanPresenter que contiene las operaciones.

El paquete Modelo contiene todas las clases relacionadas con los datos de la lógica del negocio. La clase PresentadorLogin del paquete Presentador crea Parametros, ViewDialog y UserLoginTask. La clase Parametros es la responsable de los parámetros y/o headers de las peticiones REST que las necesitan. La clase ViewDialog es la clase genérica que define el formato de todos los ViewDialogs de la aplicación. La clase UserLoginTask contiene todas las operaciones y atributos necesarios para realizar la autenticación en un segundo plano de la aplicación y crea 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).

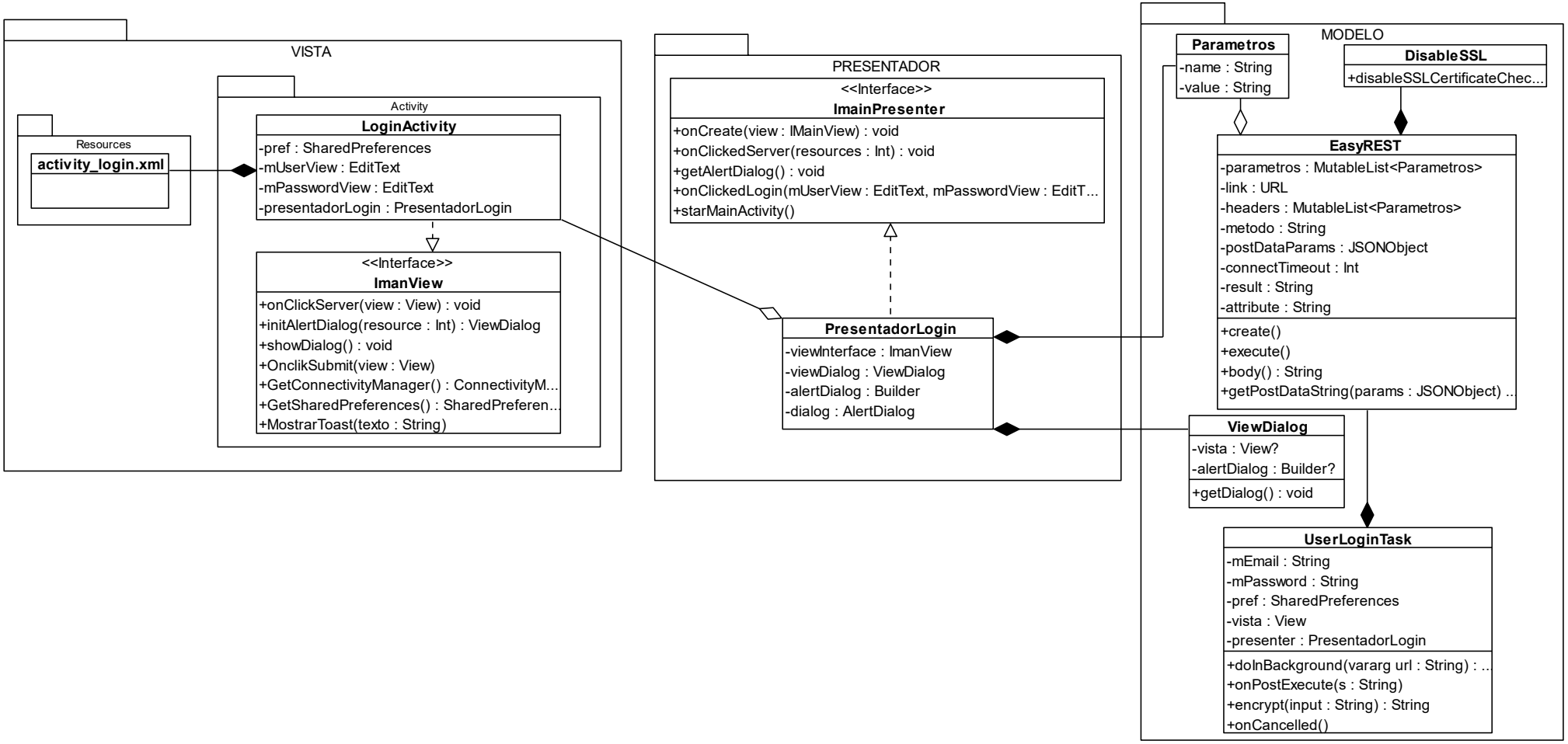

Figura 8. Diagrama de clases del diseño autenticar usuario.

2.3.1.1 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 clases controladoras: *PresentadorLogin.kt* que es la encargada de controlar la lógica de la Autenticación en el SIPACDroid, la clase *PresentadorMain.kt* y *PresentadorEvent.kt*, administradora de los procesos de la Gestión de la Vista y de la Gestión de las Actividades respectivamente.

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 el Sistema de Planificación, 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 dicho 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, 2014).

2.4.2 Patrón arquitectónico Modelo Vista Presentador

El patrón arquitectónico Modelo Vista Presentador (MVP) representado en la Figura 9 es una derivación del patrón arquitectónico Modelo–Vista–Controlador (MVC), que tiene como objetivo separar la interfaz de usuario de la lógica de las aplicaciones.

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 formateados. Pero a diferencia del MVC típico, también decide qué ocurre cuando se interactúa con la vista (MoleQla: revista de Ciencias de la Universidad Pablo de Olavide, Patrón Modelo-Vista-Presentador (MVP), 2015).

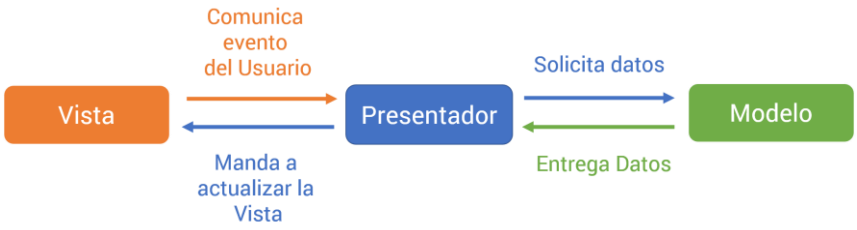

Figura 9. Patrón arquitectónico Modelo-Vista-Presentador.

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).

En servidor SIPAC el cual es necesario para la sincronización de las actividades 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 10.

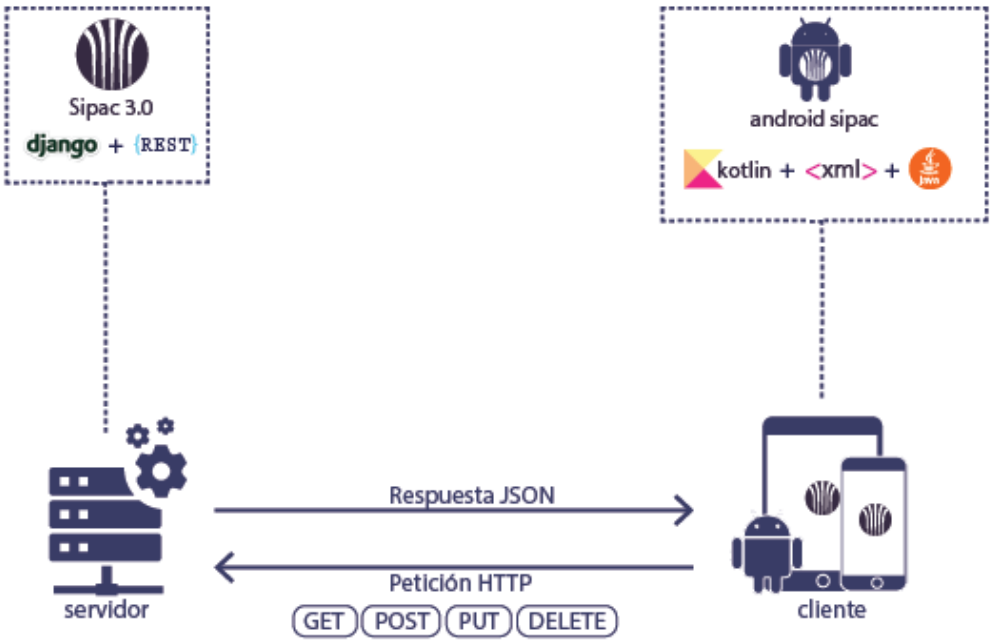

Figura 10. Servicio REST de SIPAC

En el Django REST framework del servidor del SIPAC la ruta para la autenticación en el sistema es /api-token-auth/ y para las actividades /api/actividad/). En la Figura 11, se muestra un ejemplo del consumo mediante el POST del servicio REST de autenticación. En la Figura 12 se muestra un ejemplo del consumo mediante el GET del servicio REST de Actividad mostrándose los elementos de las actividades según se evidencia en el modelo conceptual (Figura 7


```
{
  "token": ""
  "permissions": "super",
  "userId": 2
}
```


Figura 11. Resultado de realizar una petición POST válida en la ruta de autenticación.

```
{"results": [{"start": "2019-06-17T04:00:00Z",
              "event": {"title": "hoy como ayter",
                  "dirigentes": "",
                  "extra_plan": false,
                  "tipo_actividad": null,
                  "categoria": null,
                  "participantes": "",
                  "end": "2019-06-17T04:00:00Z",
                  "es_actividad_principal": false,
                  "color_event": "",
                  "puntualizada": false,
                  "description": "",
                  "organismo": null },
              "creador": "xipacsuper",
              "tipo_concepto": 1,
              "fecha_alarma": null,
              "contenido_alarma": null,
              "id": 53,
              "description": "",
              "cancelled": false,
              "id_padre": null,
              "relatedwithplan": 0,
              "cumplimiento": 0,
              "concepto": 5,
              "tiporepeticion": false,
              "end": "2019-06-17T04:00:00Z",
              "observaciones": 0,
              "puntualizada_x_mod": false,
                "original_start": "2019-06-17T04:00:00Z",
              "estado": "Creado",
              "lugar": null }] }
```


Figura 12. Resultado de realizar una petición GET válida en la ruta de las actividades.

2.5 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 abordaron los principales conceptos del dominio del problema lo que permitió un mejor entendimiento del módulo de Planificación de SIPAC. 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, PRUEBAS Y VALIDACIÓN DE LA INVESTIGACIÓN

Introducción

En el presente capítulo se tendrán en cuenta todos los aspectos del diseño del sistema con la finalidad de ejecutar las tareas de implementación y prueba. Se abordan los estándares de codificación empleados durante el desarrollo del sistema para garantizar un buen entendimiento y legibilidad del código. Se describen las pruebas efectuadas al software que tienen como objetivo: detectar y corregir el máximo de errores en el sistema, antes de su entrega al cliente y, se valida la investigación propuesta, midiendo la variable dependiente portabilidad mediante la norma ISO/IEC 25023.

3.1 Modelo de bases de datos de la aplicación Android

El modelo entidad-relación representado en la Figura 13, 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)

El diagrama entidad relación de la solución descrita (Figura 13), está caracterizado por la entidad Actividad que responde a acciones o tareas, propias de una persona o entidad, destinadas para cumplir determinado(s) objetivo(s) la cual posee los atributos como: start_ts, end_ts, title, location, description, import_id, flags, event_type, last_updated, source, cumplimiento, principal y dirigentes. Dicha entidad se encuentra relacionada con la entidad Objetivo que describe los estados que un organismo intenta alcanzar con una cardinalidad de uno a muchos. También posee relaciones con una cardinalidad de uno a uno con las entidades Organismo, TipoActividad y CategoríaActividad, así como una relación auto referenciada de cardinalidad de muchos a muchos que representa las repeticiones que poseen las actividades.

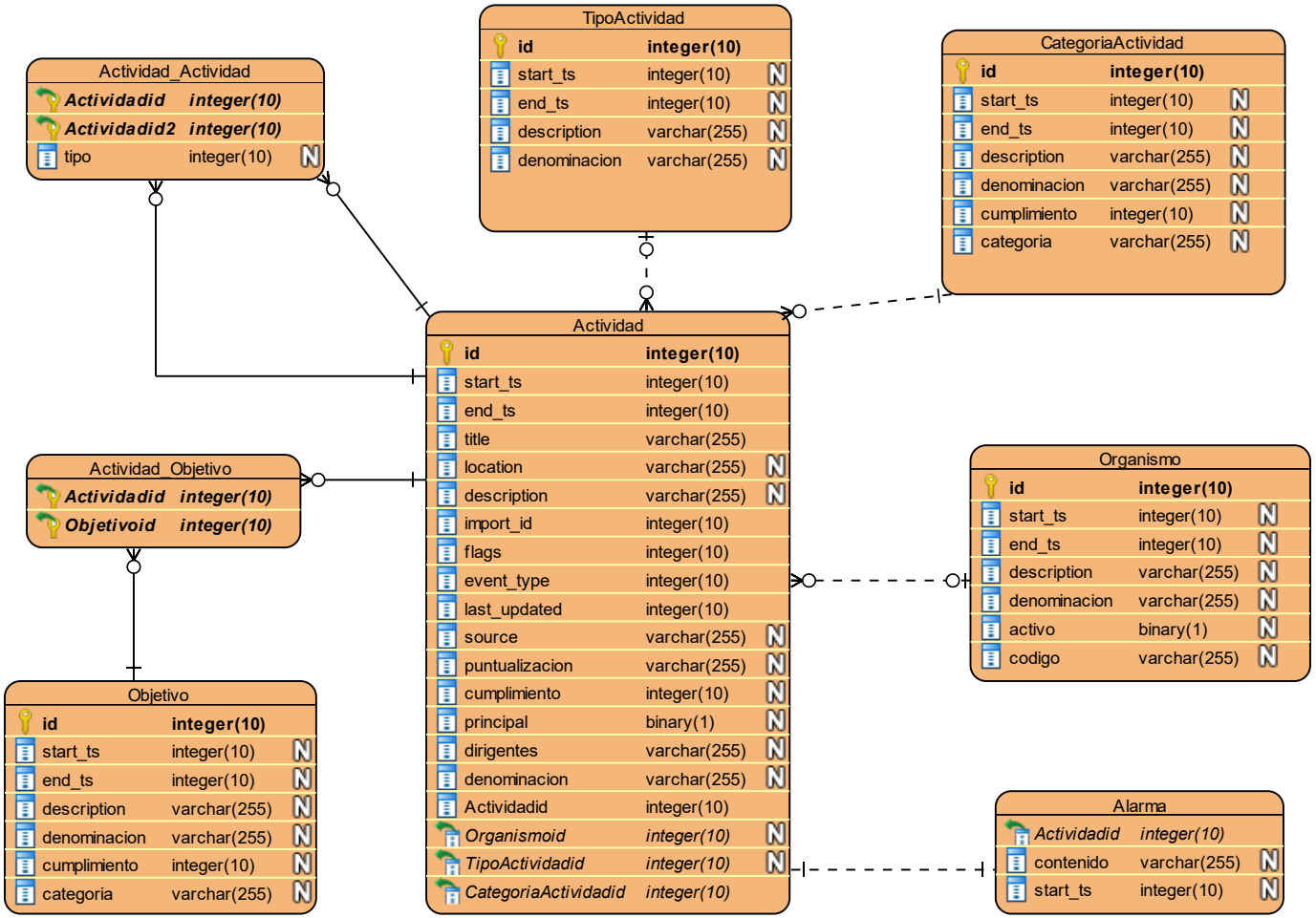

Figura 13. Modelo entidad relación del módulo de actividades[9]

[9] En este diagrama existen términos en inglés que se corresponden con los campos predefinidos utilizados por el calendario seleccionado.

3.2 Modelo de componentes de la aplicación

El diagrama de componentes representado en la Figura 14, describe los componentes usados para desarrollar las funcionalidades del sistema, estos son representados por librerías, paquetes, documentos y otros elementos que conforman y describen a la organización que permite a su vez la construcción de ejecutables utilizando ingeniería directa e inversa. (Rajagopal, et al., 2017)

En la Figura 14, el componente Sipac Python-Django responde a los elementos que componen el servidor del SIPAC, así como las estructuras internas necesarias para crear una api REST que permita realizar el consumo de servicios con componentes externos como aplicaciones de calendario que utilicen el protocolo CalDav y EXTJS 6.2 modelo encargado de soportar las operaciones de la aplicación web de SIPAC en su versión 3.0.Este gestiona una base de datos denominada SIPAC con el gestor PostgresSQL que, haciendo uso del Django REST-Framework permite el consumo de servicios de la aplicación Android que contiene los componentes Autenticación y Calendario.

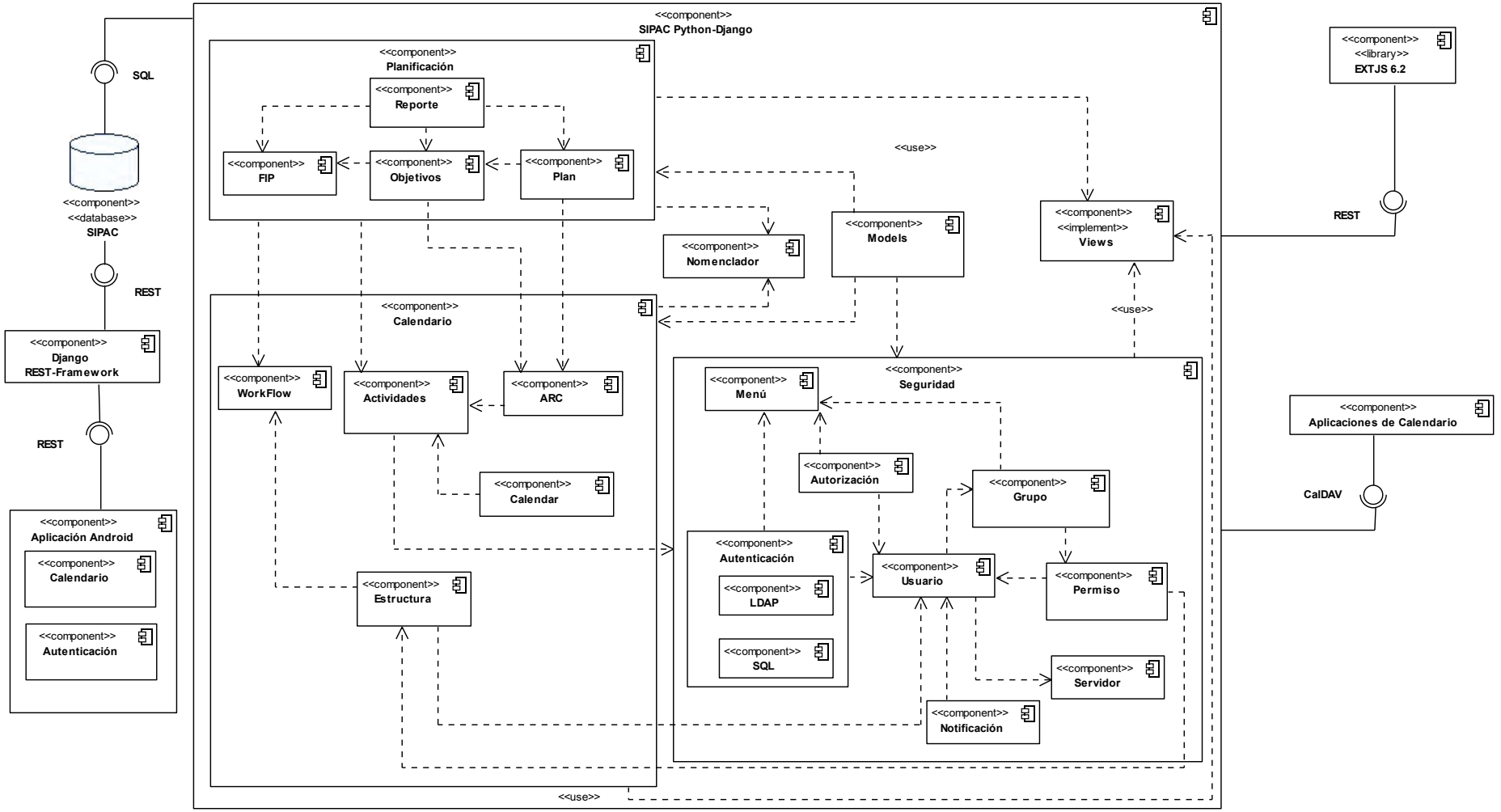

Figura 14. Modelo de componentes de la aplicación.

3.3 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 run-time en 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:

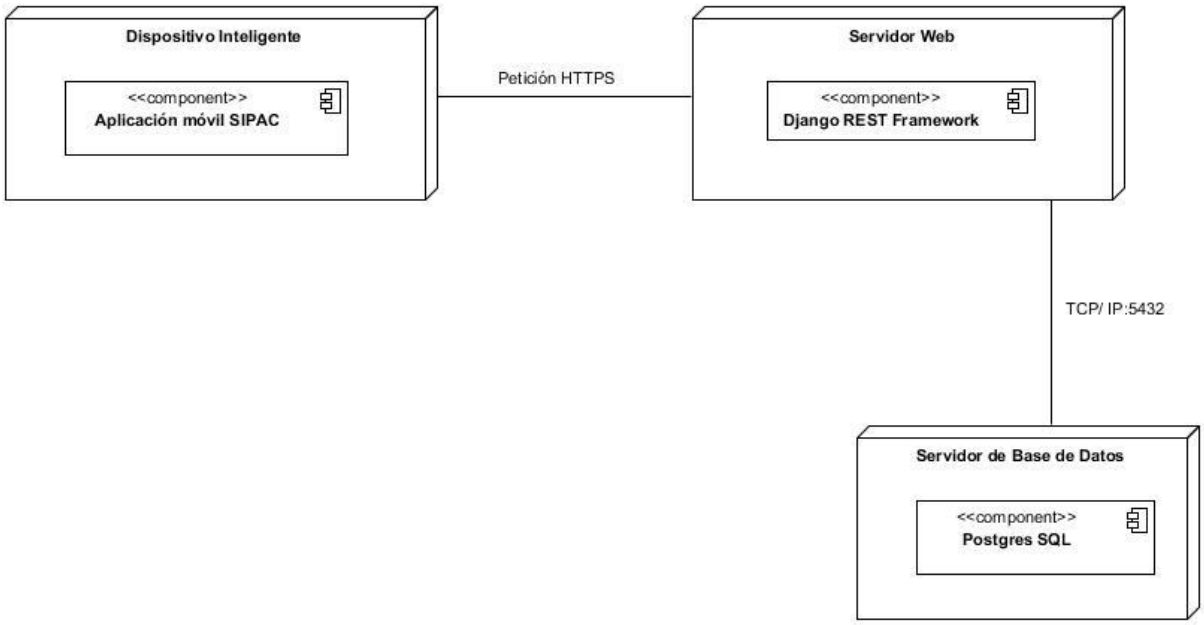

Figura 15. Diagrama de despliegue.

3.3.1 Descripción de componentes

Nodo Dispositivo Inteligente: en este nodo (Figura 16) 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 16. Nodo Dispositivo Inteligente.

Nodo Servidor Web: en este nodo (Figura 17) se guardarán todas las direcciones de los recursos a manejar, mediante los servicios REST.

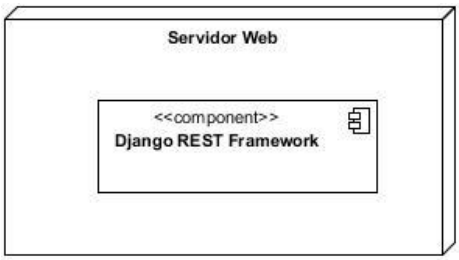

Figura 17. Nodo Servidor Web.

Nodo Servidor de Base de Datos SIPAC: en este nodo (Figura 18) se almacenan todos los datos referentes al módulo de autenticación y actividades.

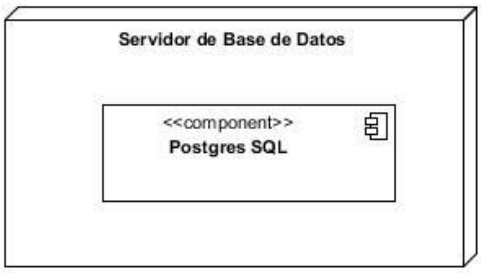

Figura 18. Nodo Servidor de Base de Datos SIPAC.

3.4 Estándares de codificación de la aplicación Android

El estándar de codificación o Code Standard en inglés comprende los aspectos a tener en consideración para la generación de código, permite la legibilidad del código fuente y repercute directamente en la capacidad de comprender uno o varios sistemas. Todo lo anterior viene marcado por el mantenimiento del código, facilidad con que el sistema de software puede modificarse para añadirle nuevas características, modificar las ya existentes, depurar errores, o mejorar el rendimiento. (Board, 2018)

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 “Canel Humps” o 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 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 Companion 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.

Las clases anidadas deben ser colocadas junto al código que la utilizas. Si las clases están destinadas a ser utilizadas externamente y no se hace referencia dentro de la clase, se deben colocar al final, después de su objeto Companion. (JetBrains)

3.4.7 Nomenclatura XML

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 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.start_ts)` (Android Developers, 2018).

3.5 Pruebas de la aplicación Android

La prueba de software es la verificación dinámica del comportamiento de un programa contra el comportamiento esperado, usando un conjunto finito de casos de prueba, seleccionados de manera adecuada (Board, 2018).

Según la norma ISO/IEC/IEEE 24765: 2017- 2019 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/IEEE, 2017)

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 19, se muestra el contexto en que se realiza la prueba de software:

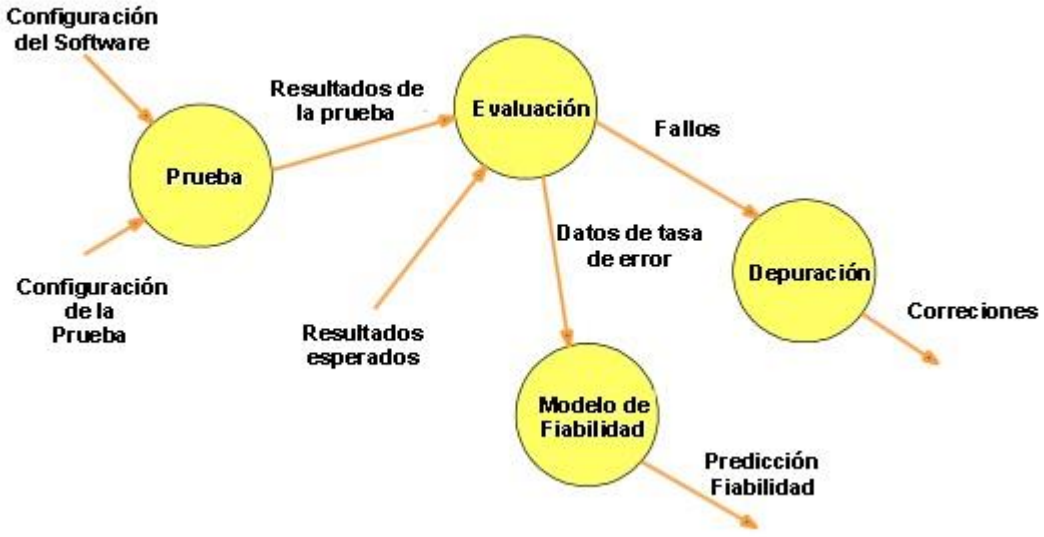

Figura 19. Contexto de la prueba de software

3.5.1 Diseño de las pruebas de la aplicación Android

Diseños de las pruebas: es la identificación de la técnica o técnicas de pruebas que se utilizarán para probar el software. Distintas técnicas de prueba ejercitan diferentes criterios como guías para realizar las pruebas.

Se selecciona para realizar las pruebas a la aplicación Android desarrollada para ejecutar las operaciones de Planificación, la estrategia de pruebas unitarias, funcionales, de seguridad y de rendimiento. Además, la librería JUnit desarrollada para probar el funcionamiento de las clases y métodos que componen la aplicación.

La prueba de unidad es la primera fase de las pruebas dinámicas y se realizan sobre cada módulo del software de manera independiente. El objetivo es comprobar que el módulo, entendido como unidad funcional de un programa independiente, está correctamente codificado. En general, un módulo se entiende como un componente software que cumple las siguientes características: debe ser un bloque básico de construcción de programas, debe implementar una función independiente simple, no deberá tener más de 500 líneas de código.

En los últimos años se han desarrollado un conjunto de herramientas que facilitan la elaboración de pruebas unitarias en diferentes lenguajes. Dicho conjunto se denomina XUnit y JUnit es la herramienta utilizada para realizar pruebas unitarias en Kotlin. El concepto fundamental de estas herramientas es el caso de prueba (test case), y la suite de prueba (test suite). Los casos de prueba son clases o módulos que disponen de métodos para probar los métodos de una clase o módulo concreto. Mediante las suites se organizan los casos de prueba, de forma que cada suite agrupa los casos de prueba de los módulos que están funcionalmente relacionados.

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 20 y ejecutar la prueba presionando Run ( ) .

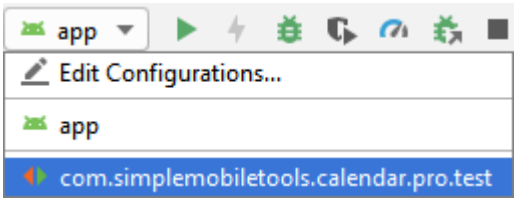

Figura 20. Configuración a seleccionar para ejecutar Pruebas.

Aplicando los casos de pruebas generados previamente e identificando los posibles fallos producidos al comparar los resultados esperados con los obtenidos. Es necesario ejecutar un total de 17 casos de pruebas para cada uno de los requisitos funcionales definidos en la aplicación propuesta. Ejemplo, el test LoginActivityTest.kt, como se muestra en la Figura 21. Los resultados, en una primera iteración fueron satisfactorios, corroborando la correcta implementación de los métodos.

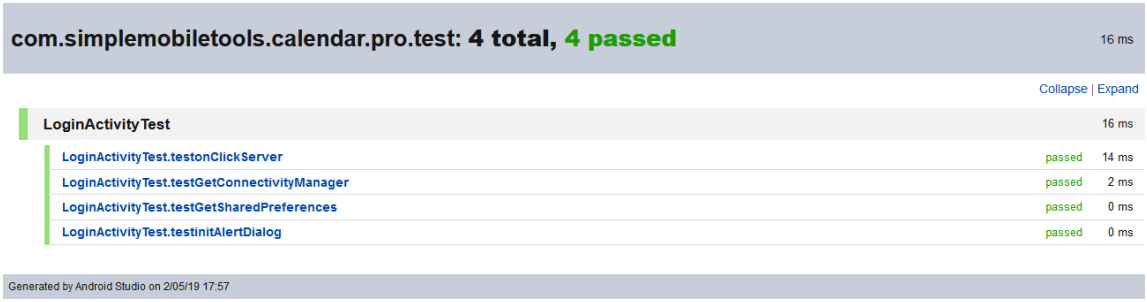

Figura 21. Resultado de la prueba unitaria en LoginActivityTest.kt.

3.5.3 Pruebas de Seguridad

La prueba de seguridad es aquella que intenta verificar que los mecanismos de protección que se construyen en un sistema en realidad lo protegerán de cualquier penetración impropia. (Pressman, 2010)

Para la aplicación de estas pruebas se definió la utilización de los indicadores de seguridad en dispositivos móviles definidas por el Proyecto de Seguridad de Aplicaciones Web Abiertas o OWASP por sus siglas en inglés:

1. Uso inadecuado de la Plataforma: Esta categoría abarca el uso indebido de una función de la plataforma o la falta de uso de los controles de seguridad de esta.
2. Almacenamiento inseguro de datos: Cubre el almacenamiento de datos inseguros y la fuga de datos no intencionales.
3. Comunicación Insegura: Abarca el mal establecimiento de la conexión, las versiones de SSL incorrectas, la negociación débil, la comunicación en texto claro de activos sensibles, etc.
4. Autenticación Insegura: Esta categoría encapsula las nociones de autenticación del usuario final o la mala gestión de la sesión. Esto puede incluir:
    a. No identificación del usuario cuando debería ser requerido.
    b. No mantener la identidad del usuario cuando es requerido.
    c. Debilidades en la gestión de sesiones.
5. Criptografía insuficiente: Aplica a la criptografía a un activo de información sensible.
6. Autorización insegura: Captura cualquier falla en la autorización (por ejemplo, decisiones de autorización en el lado del cliente, navegación forzada, etc.).
7. Calidad del Código del lado del cliente: Captura todos los problemas de implementación del lado del cliente móvil.
8. Manipulación del código: Abarca la aplicación de parches binarios, la modificación de recursos locales, el enlace de métodos, el método swizzling y la modificación de memoria dinámica.
9. Ingeniería inversa: Incluye el análisis del binario central final para determinar su código fuente, bibliotecas, algoritmos y otros activos.

10. Funcionalidad extraña: Funcionalidad de puerta trasera oculta u otros controles de seguridad de desarrollo internos que no están destinados a ser liberados en un entorno de producción. (OWASP, 2017)

Para verificación del cumplimiento de estos indicadores en la solución se utilizó la herramienta automatizada online ImmuniWeb[10] la cual posee una certificación ISO/IEC 27001.Durante la primera iteración se identificaron un total de 7 tipos de riesgos como se evidencia en la siguiente figura:

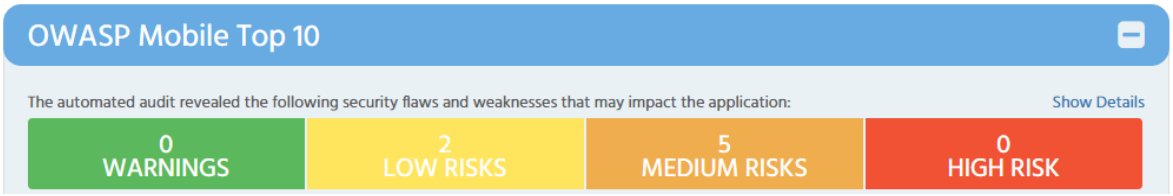

Figura 22. Resultado de la ejecución de las pruebas a los indicadores de OWASP.

Dentro de los riesgos de nivel Medio se identificaron:

1. Sensibilidad de datos codificados.
2. Datos externos en consultas SQL sin procesar.
3. Usos del protocolo no encriptado HTTP.
4. Generador de números aleatorios predictibles.
5. Utilización de base de datos sin encriptación.

De nivel bajo se reconocieron:

1. Sensibilidad de datos.
2. Falta de protección de pulsaciones.

De los riesgos identificados solamente corresponden a la solución propuesta 4 no conformidades, las restantes pertenecen a declaraciones y clases propias del sistema operativo, las cuales no son editables. Durante una segunda iteración de la prueba todas las no conformidades referentes a la aplicación fueron resueltas.

3.5.4 Pruebas de Rendimiento

Las pruebas de rendimiento son un tipo de prueba realizadas para evaluar el grado en que un elemento de prueba cumple con sus funciones designadas dentro de las restricciones de tiempo y otros recursos dados. (ISO/IEC/IEEE, 2017)

Según la Norma ISO/IEC 25010:2011 una de las propiedades en las que se descompone la calidad del producto es la eficiencia del rendimiento y esta no es más que el rendimiento relativo a la cantidad de recursos utilizados en las condiciones establecidas. (ISO/IEC, 2014)

[10] Proveedor global de pruebas de seguridad y clasificaciones de seguridad para aplicaciones web y móviles utilizando como bases la Inteligencia Artificial y el Aprendizaje Automático o Machine Learning.

Para la medición del rendimiento se realizaron dos iteraciones de 25 minutos y se midió la utilización de recursos en el tiempo, utilizando las siguientes herramientas automatizadas del IDE Android Studio en su versión 3.4.1:

- Perfil de Memoria: permite identificar fugas y migraciones de memoria que puedan generar interrupciones, congelamiento e incluso bloqueo en la app (Android Developers, 2018).
- Perfil de la Unidad Central de Procesamiento o CPU según sus siglas en inglés: posibilita la inspección del uso de CPU y la actividad de subprocesos de las apk en tiempo real, y registrar seguimientos a los métodos (Android Developers, 2018).
- Perfil de uso de red: muestra la actividad de red en tiempo real en una línea de tiempo; incluyendo datos enviados y recibidos, y la cantidad de conexiones (Android Developers, 2018).

Como resultado de la primera iteración se identificaron ocho problemas que incurrían en un uso excesivo de memoria RAM, dos problemas en el uso del CPU y cinco relacionados con la utilización de recursos de red. A continuación, se muestra un gráfico que ilustra el uso de estos recursos durante la primera iteración:

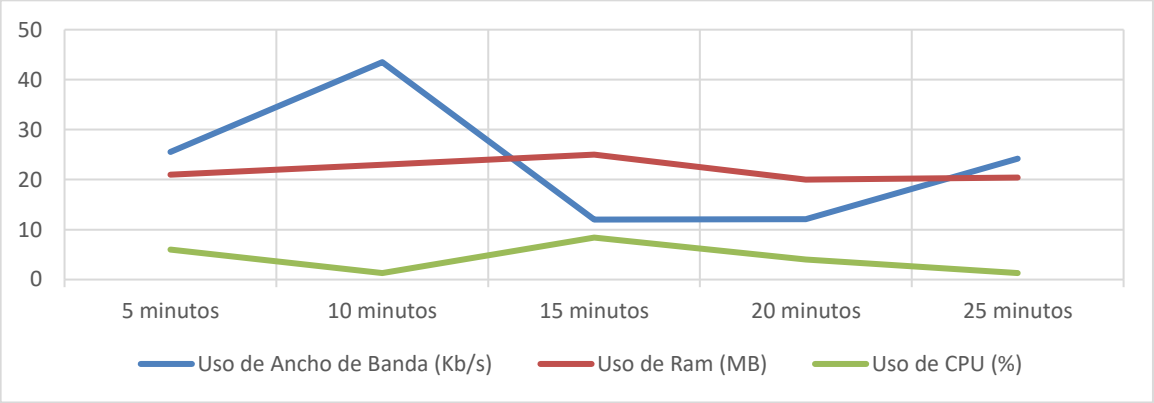

Figura 23.Resultado de las pruebas de la primera iteración.

La realización de la segunda iteración de las pruebas de rendimiento, resultó exitosa al solucionarse los problemas identificados anteriormente incurriendo en un aumento del rendimiento en el uso de los recursos de red, de memoria y de procesamiento en un 30.08, 54.9 y 25 por ciento respectivamente, equivaliendo a un total de aumento del rendimiento de la aplicación Android en un 36.39 % como se evidencia en la siguiente gráfica:

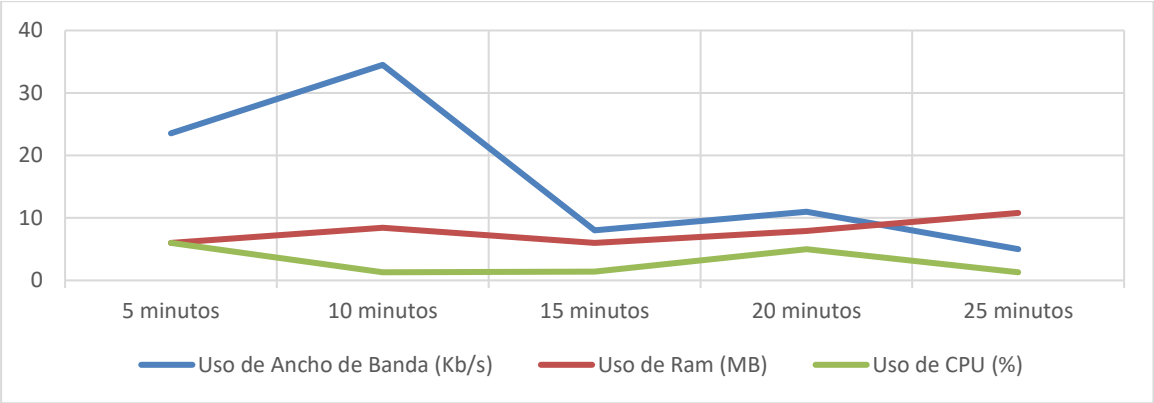

Figura 24.Resultados de la segunda iteración de las pruebas de rendimiento.

3.5.5 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)

3.5.5.1 Método de Caja Negra

Las pruebas de caja negra se centran en los requisitos funcionales del software, es decir, la prueba de caja negra permite obtener conjuntos de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa. La prueba de caja negra intenta encontrar errores de las siguientes categorías. (Pressman, 2010)

- Funciones incorrectas o ausentes.
- Errores de interfaz.
- Errores en estructuras de datos o en accesos a las bases de datos externas.
- Errores de rendimiento.
- Errores de inicialización y de terminación.

Se utiliza la técnica de partición equivalente para llevar a cabo el método de caja negra, a continuación, se describe.

3.5.5.1.1 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 22 diseños de casos de prueba, los cuales encapsulan los 17 requisitos funcionales definidos en la tabla 3 y con el objetivo de comprobar que las funcionalidades del sistema se realizaron un total de 3 iteraciones de pruebas de caja negra, obteniéndose los siguientes resultados:

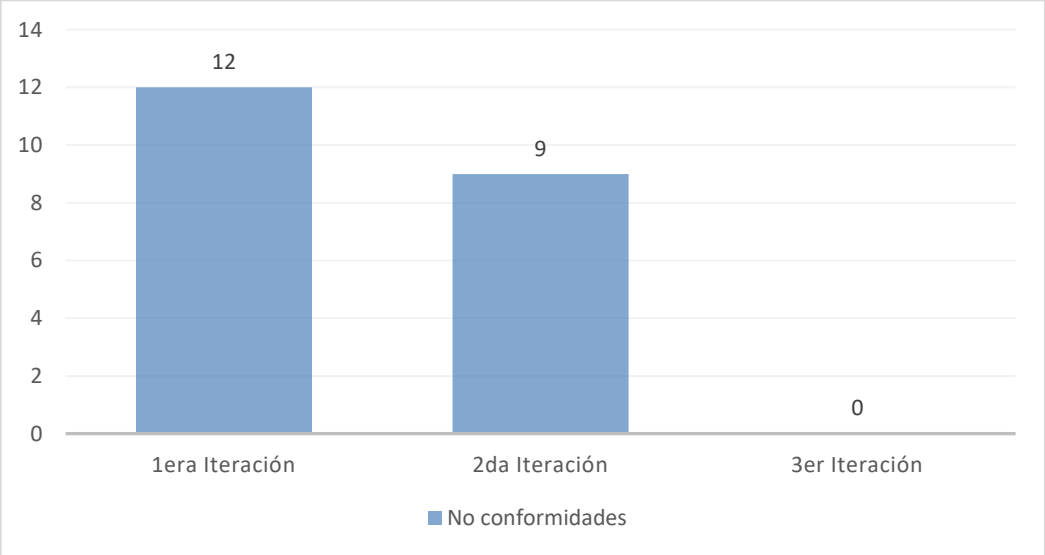

Figura 25. Gráfico de no conformidades.

En la primera iteración se ejecutaron las pruebas internas realizadas por las analistas y probadores del departamento Desarrollo de Componentes del centro CEIGE, en la cual, se detectaron 9 NC de la aplicación. En la segunda iteración se ejecutaron las pruebas de liberación por la Dirección de Calidad de la UCI, en la cual, se detectaron 12 NC. En la tercera iteración realizada se obtuvieron resultados satisfactorios al no detectarse no conformidades.

3.6 Validación de la investigación de la aplicación Android

La portabilidad según la ISO 9126-1 es la capacidad de transferencia de un producto de software de un entorno a otro. El entorno puede ser de tipo organizacional, hardware o software. Diferente software, diferente plataforma, código portátil o componentes cuyo reemplazo no afecta al resto de la aplicación.

Para evaluar la variable dependiente portabilidad, se utiliza el estándar internacional para la evaluación del software ISO/IEC 25023. La norma ISO/IEC 25023 - Measurement of system and

software product quality: define específicamente las métricas para realizar la medición de la calidad de productos y sistemas software.

3.6.1 Medidas de portabilidad

Las medidas de portabilidad se usan para evaluar el grado de efectividad y eficiencia con el cual un sistema, producto o componente se puede transferir de un hardware, software u otro entorno operacional o de uso a otro.

3.6.2 Medidas de adaptabilidad

Las medidas de adaptabilidad se utilizan para evaluar el grado en que un producto o sistema puede adaptarse eficaz y eficientemente a un hardware, software u otros entornos operativos de uso diferentes o en evolución.

Tabla 6. Medidas de adaptabilidad.

ID	Nombre	Descripción	Función de medición
PAd-1-G	**Adaptabilidad ambiental del hardware**	¿El software o el sistema es lo suficientemente capaz para adaptarse a diferentes entornos de hardware?	X = 1 - A / B A = Número de funciones que no se completaron o resultados insuficientes para cumplir con los requisitos durante las pruebas B = Número de funciones que han sido probadas en diferentes entornos de hardware
PAd-2-G	**Adaptabilidad ambiental del software del sistema**	¿El software o el sistema es lo suficientemente capaz para adaptarse a diferentes entornos de software del sistema?	X = 1 - A / B A = Número de funciones que no se completaron o resultados insuficientes para cumplir con los requisitos durante las pruebas B = Número de funciones que han sido probadas en diferentes entornos de software del sistema
NOTA 1: Cuando un usuario tiene que aplicar un procedimiento de adaptación que no haya sido previamente proporcionado por el software para una necesidad de adaptación específica, el esfuerzo del usuario requerido para adaptarse debe ser medido. NOTA 2: El software del sistema puede incluir sistemas operativos, middleware, sistema de gestión de bases de datos, compilador, sistema de gestión de red, etc.			

PAd-3-S	**Adaptabilidad del entorno operativo**	¿Con qué facilidad puede realizarse la prueba de funcionamiento desde el punto de mantenimiento?	X = 1 - A / B A = Número de funciones que no se completaron o resultados insuficientes para cumplir con los requisitos durante las pruebas operativas con el entorno del usuario B = Número de funciones que han sido probadas en diferentes entornos operacionales

3.6.3 Evaluación de las medidas de adaptabilidad

Tabla 7. Evaluación de las medidas de adaptabilidad.

Aplicación Android para realizar las operaciones sobre los elementos de la planificación Total de requisitos probados: 17.				
No	**Ambiente del software**	**Ambiente del hardware**	**Adaptabilidad del entorno operativo**	**Resultado**
1	Versión de Android 6.0.1	Modelo: SM-J710GN Marca: Samsung Galaxy J7 (2016)	Compilar la aplicación con los cambios realizados.	Se adapta el software y hardware, se puede compilar la aplicación.
2	Versión de Android 8.1.0	Modelo: JKM-LX3 Marca: Huawei Y9 2019	Compilar la aplicación con los cambios realizados.	Se adapta el software y hardware, se puede compilar la aplicación.
3	Versión de Android 5.1.1	Modelo: SCL-AL00 Marca: Huawei Honor	Compilar la aplicación con los cambios realizados.	Se adapta el software y hardware, se puede compilar la aplicación.
4	Versión de Android 7.0	Modelo:SM-G925T Marca: Samsung Galaxy S6 Edge	Compilar la aplicación con los cambios realizados.	Se adapta el software y hardware, se puede compilar la aplicación.
5	Versión de Android 7.1	Modelo: LG M150 Marca: LG Phoenix 3	Compilar la aplicación con los cambios realizados.	Se adapta el software y hardware, se puede compilar la aplicación.
	Función de medición X = 1 - A / B	Función de medición X = 1 - A / B	Función de medición X = 1 - A / B X = 1 - 0/ 85 X = 1	Ambiente del software = 1 Ambiente del hardware = 1

	X = 1 - 0/ 85 X = 1	X = 1 - 0/ 85 X = 1		Adaptabilidad del entorno operativo = 1

3.6.4 Medidas de instalabilidad

Las medidas de instalabilidad se usan para evaluar el grado de efectividad y eficiencia con que un producto o sistema puede ser instalado y/o desinstalado con éxito en un entorno específico.

Tabla 8. Medidas de instalabilidad

ID	Nombre	Descripción	Función de medición
PIn-1-G	**Eficiencia del tiempo de instalación**	¿Qué tan eficiente es el tiempo real de instalación en comparación con el tiempo esperado?	X=$\sum_{i=1}^{n}(A_i/B_i)/ n$ A_i = Tiempo de trabajo total dedicado a la instalación i B_i = Tiempo esperado para realizar una instalación i *n*= Número de instalaciones medidas
NOTA 1: X mayor que 1 representa una instalación ineficiente, y X menos de 1 representa una instalación muy eficiente. NOTA 2: El tiempo esperado para realizar una instalación puede basarse en datos históricos o promedios de la industria.			
PIn-2-G	**Facilidad de instalación**	¿Pueden los usuarios o encargados de mantenimiento personalizar el procedimiento de instalación para su conveniencia?	X = A / B A = Número de casos en los que un usuario logra personalizar el procedimiento de instalación B = Número de casos en los que un usuario intentó personalizar el procedimiento de instalación para la conveniencia del usuario
NOTA Estos cambios de procedimiento de instalación pueden ser reconocidos como personalización de la instalación por el usuario.			

3.6.5 Medidas de reemplazabilidad

Las medidas de reemplazabilidad se usan para evaluar el grado en que un producto puede reemplazar a otro producto de software especificado para el mismo propósito en el mismo ambiente.

Tabla 9. Medidas de reemplazabilidad.

ID	Nombre	Descripción	Función de medición
PRe-1-G	**Similitud de uso**	¿Qué proporción de funciones de usuario del producto reemplazado se puede realizar sin ningún aprendizaje adicional o solución?	X = A/B A = Número de funciones de usuario que se pueden realizar sin ningún aprendizaje adicional o solución alternativa B = Número de funciones de usuario en el producto de software reemplazado
NOTA Las funciones de usuario son aquellas que el usuario puede llamar y utilizar para realizar las tareas previstas, incluidas las interfaces de usuario.			
PRe-2-S	**Equivalencia de la calidad del producto**	¿Qué proporción de las medidas de calidad se satisface después de reemplazar el producto de software anterior por éste?	X = A / B A = Número de medidas de calidad del nuevo producto que son mejores o iguales al producto sustituido B = Número de medidas de calidad del producto de software sustituido que son relevantes
NOTA Algunas de las cualidades de los productos críticos correspondientes a la intercambiabilidad son la interoperabilidad, la seguridad y la eficiencia del rendimiento.			
PRe-3-S	**Inclusión funcional**	¿Pueden las funciones similares ser usadas fácilmente después de substituir el producto de software anterior por éste?	X = A/B A = Número de funciones que producen resultados similares a los anteriores B = Número de funciones que deben utilizarse en el producto de software sustituido
PRe-4-S	**Capacidad de reutilización / importación de datos**	¿Pueden usarse los mismos datos después de reemplazar el producto de software anterior por éste?	X = A / B A = Número de datos que se pueden utilizar continuamente como antes

			B = Número de datos que se utilizarán continuamente en el producto de software sustituido

3.6.6 Evaluación de las medidas de instalabilidad

Tabla 10. Evaluación de las medidas de instalabilidad.

Aplicación Android para realizar las operaciones sobre los elementos de la planificación Total de requisitos probados: 17.		
Prestaciones Hardware	**Eficiencia del tiempo de instalación**	**Facilidad de instalación**
Marca Huawei Y9 2019 Procesador: Hisilicon Kirin 710 Memoria: 3 GB	X=$\sum_{i=1}^{n}(A_i/B_i)$/ n X = ((5.2s/10s) /4 X= 0,13 Resultado: muy eficiente	X = A / B X= 4/4 X=1 Resultado: eficiente
Marca LG Phoenix 3 Memoria: 1.5 GB Procesador: Qualcomm Snapdragon 210	X=$\sum_{i=1}^{n}(A_i/B_i)$/ *n* X = (11.76 s/10s)/2 X= 0.588 Resultado: eficiente	X = A / B X= 2/2 X=1 Resultado: eficiente
Marca Galaxy S6 Edge Procesador: Samsung Exynos 7 Octa(7420) Memoria RAM: 3 GB	X=$\sum_{i=1}^{n}(A_i/B_i)$/ *n* X = (4.26s/10s)/4 X= 0.1065 Resultado: muy eficiente	X = A / B X= 4/4 X=1 Resultado: eficiente

3.6.7 Evaluación de las medidas de reemplazabilidad

Tabla 11. Evaluación de las medidas de reemplazabilidad.

Aplicación Android para realizar las operaciones sobre los elementos de la planificación.			
Similitud de uso	**Equivalencia de la calidad del producto**	**Inclusión funcional**	**Capacidad de reutilización/importación de datos**
X = A/B X= 14/14 X= 1	X = A/B X= 14/4 X= 3,5	X = A/B X= 14/14 X=1	X = A/B X= 14/7 X=2

	Relevantes: Autenticar Usuario Adicionar Actividad		Nota: En el sistema Android no se carga toda la información contenida en el sistema web, solo la seleccionada por el usuario.

3.6.8 Evaluación de la portabilidad aplicando la norma ISO/IEC 25023

Tabla 12. Evaluación de la portabilidad aplicando la norma ISO/IEC 25023.

Características	**Ponderación**	**Resultado**
Adaptabilidad	Ambiente del software = 1 Ambiente del hardware = 1 Adaptabilidad del entorno operativo = 1	**Aceptable**: los resultados de la variable x siempre fue 1.
Instalabilidad	**Eficiencia del tiempo de instalación =** Eficiente, muy eficiente, muy eficiente. **Facilidad de instalación =** Muy Eficiente	**Bueno**: la eficiencia del tiempo siempre fue menor que 1 y la facilidad de la instalación 1. Con resultados de eficiente para la instalación. **Recomendación** Fortalecer algunas de las características de la aplicación.
Reemplazabilidad	**Similitud de uso**= 1 **Equivalencia de la calidad del producto** = 3,5 **Inclusión funcional** =1 **Capacidad de reutilización/importación de datos** = 2	**Poco aceptable**: Es poco aceptable porque en la aplicación web se muestra toda la información mientras que en la aplicación Android por las características solamente se muestran las seleccionadas por el usuario.

3.6.9 Conclusiones de la evaluación de la portabilidad aplicando la norma ISO/IEC 25023

Para obtener los valores de las características de portabilidad, después de la aplicación de las medidas de evaluación, se procede adaptar la fórmula para el cálculo de las subcaracterísticas: adaptabilidad, instalabilidad y reemplazabilidad, y la característica de portabilidad; propuestas en la norma ABNT NBR ISO/IEC 14598-6 Anexo C (ISO, 2001) **Vc = ∑ Vsc/nsc y Vsc = ∑ m/ (n-nd).** En donde: Vc es el valor medio de la característica; Vsc es el valor medido de la subcaracterística; nsc es el número de subcaracterísticas; m es 1, si la respuesta es positiva, en caso contrario es 0; n es el número total de medidas; nd es el número de preguntas descartadas.

Tabla 13. Resultados de la aplicación de las medidas de la portabilidad.

Subcaracterística	Medidas	m	n	nd	m/ (n-nd)

Adaptabilidad	Adaptabilidad ambiental del hardware	1	3	0	1/3
	Adaptabilidad ambiental del software del sistema	1	3	0	1/3
	Adaptabilidad del entorno operativo	1	3	0	1/3
	Valor medido de la subcaracterística				1
Instalabilidad	Eficiencia del tiempo de instalación	1	2	0	1/2
	Facilidad de instalación	1	2	0	1/2
	Valor medido de la subcaracterística				1
Reemplazabilidad	Similitud de uso	1	4	0	1/4
	Equivalencia de la calidad del producto	0	4	10	0
	Inclusión funcional	1	4	0	1/4
	Capacidad de reutilización/importación de datos	0	4	7	0
	Valor medido de la subcaracterística				1/2
Valor de la característica					0.955

El resultado obtenido basado en la evaluación para la subcaracterística propuesta en la norma 25000, aplicada a la característica portabilidad, es de un valor Vc=0,955, representando un 95.5%. El valor significa que la portabilidad del sistema SIPAC, en dispositivos móviles con sistema operativo Android en versiones 4.4.1 o superior, está representada por un 95.5%. A partir de la situación antes descrita, se evidencia que se ha mejorado la portabilidad del sistema SIPAC, contribuyendo en un 95.5 % en portabilidad, para dispositivos móviles con sistema operativo Android.

3.7 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 mediante los casos pruebas ejecutados permitió demostrar que las funciones de la aplicación propuesta son operativas. La aplicación de las medidas de las subcaracterísticas; adaptabilidad, instalabilidad y reemplazabilidad de la norma ISO/IEC 25023, para la evaluación de la portabilidad en dispositivos móviles, obtuvo que la aplicación propuesta mejora la portabilidad del sistema SIPAC.

Conclusiones Generales

Con la investigación propuesta se logró dar cumplimiento a los objetivos específicos planteados inicialmente, lo cual permite arribar a las siguientes conclusiones:

- Se analizaron los diferentes conceptos asociados al dominio del problema de la investigación como las subcaracterísticas: adaptabilidad, instalabilidad y reemplazabilidad de la norma ISO/IEC 25023 referente a la portabilidad que se utilizó en la validación de la investigación. Se definieron los conceptos de actividades reflejados en el documento de la Instrucción No.1 del Presidente de los Consejos de Estado y de Ministros. Además, se identificaron y analizaron diferentes calendarios, seleccionando el más adecuado para el desarrollo de la solución propuesta, permitiendo una personalización con los atributos distintivos de la planificación en Cuba.
- Se analizaron las tecnologías, herramientas, lenguajes de programación y metodologías de desarrollo de software que propiciaron la construcción de los módulos de Autenticación, Notificaciones y Calendario en la aplicación SIPACDroid.
- Se elaboró el modelo conceptual de la aplicación permitiendo identificar los conceptos del mundo real asociados al problema de la investigación. Además, se identificaron los requisitos funcionales y no funcionales que facilitó un mejor entendimiento de las condiciones y capacidades que el sistema debe cumplir. Se realizó el diagrama de clases del diseño reflejando las clases con sus respectivos atributos, métodos y relaciones, que se definieron en el desarrollo de la solución propuesta.
- Se realizaron los modelos de datos reflejando la relación entre las entidades; de componentes representando las interacciones entre los componentes y las interfaces que se implementan donde se brindan y consumen servicios entre la aplicación web y la aplicación Android; y de despliegue mostrando la estructura física de la solución propuesta. Obteniendo un producto que brinda y consume los servicios REST de los módulos de Autenticación, Calendario y Notificaciones en el SIPAC.
- La ejecución de las pruebas con resultados satisfactorios, permitió corroborar que los requisitos se encuentran correctamente implementados, para un conjunto de entradas, condiciones de ejecución y resultados esperados. El resultado obtenido basado en la evaluación de la característica portabilidad utilizando la norma ISO/IEC 25023, evidencia que se ha mejorado la portabilidad del sistema SIPAC en un 95,5%.

Bibliografía

Saria Preval, Ing. Katia. 2014. *Evaluación de Requisitos.* 2014.

About SQLite. 2018. About SQLite. SQLite. [En línea] 2018. [Citado el: 20 de Noviembre de 2018.] https://www.sqlite.org/about.html.

Agile, Disciplined. 2018. Simple Tools for Software Modeling -OR- It's "Use the Simplest Tool" not "Use Simple Tools". [En línea] Disciplined Agile, 2018. [Citado el: 10 de Junio de 2019.] http://www.agilemodeling.com/essays/simpleTools.htm#SelectingCASE.

Android Developers. 2018. Android Developers. [En línea] 2018. [Citado el: 12 de Noviembre de 2018.] https://developer.android.com/.

Armijo, Marianela. 2011. *Planificación Estratégica e Indicadores .* s.l. : Instituto Latinoamericano y del Caribe de Planificación Económica y Social (ILPES), 2011.

Board, International Software Testing Qualifications. 2018. International Software Testing Qualifications Board. [En línea] 2018. [Citado el: 2019 de Abril de 19.] https://www.istqb.org/downloads/category/2-foundation-level-documents.html.

Camarero, Puras, Julio, y otros. 2009. *Metodología de desarrollo ágil para sistemas móviles.Introducción al desarrollo con Android e Iphone.* Madrid : Universidad Politécnica de Madrid, 2009.

Clements, Mark, Kazman, Rick y Klein, Mark. 2001. *Evaluation the Software Architecture.* 2001.

Consejo de Ministros , Partido Comunista de Cuba y Asamblea Nacional del Poder Popular. 2012. *Instrucción No1 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 de Estado,Entidades Nacionales y Administraciones Locales del Poder Popular.* 2012.

Delía, Lisandro, y otros. 2013. *Un Análisis Experimental de Tipo de Aplicaciones para .* s.l. : Instituto de Investigación en Informática LIDI. Facultad de Informática. , 2013.

Dick, Jeremy, Hull, Elizabeth y Jackson, Ken. 2017. *Requirements Engineering.* Suiza : Springer, 2017.

Django REST framework. 2018. Django REST framework. [En línea] 2018. [Citado el: 6 de enero de 2018.] http://www.django-rest-framework.org/.

Eldhuset, Aasmund. Kotlinlang. [En línea] JetBrains. [Citado el: 19 de Abril de 2019.] https://kotlinlang.org/docs/tutorials/kotlin-for-py/declaring-variables.html.

Facebook. 2019. Facebook for developers. [En línea] 25 de 5 de 2019. https://developers.facebook.com/docs/graph-api/.

Gaines, Jeff, Boyd, Geraldine y Copley, Della. 2017. Visual Paradigm Online. [En línea] 2017. https://online.visual-paradigm.com/es/features/.

Gonzalez, Mónica. 2014. SlideShare. [En línea] 2014. https://es.slideshare.net/MonicaGlez1/ingeniera-de-software-30069503.

Gradle Inc. 2018. Gradle Docs. [En línea] 2018. [Citado el: 10 de mayo de 2018.] https://docs.gradle.org/current/userguide/userguide.htm.

BIBLIOGRAFÍA

IEEE. 2014. *Standard for Software Quality Assurance Processes.* 2014.

ISO. 2001. *ISO/IEC 14598-6,Software engineering — Product evaluation — Part 6.* 2001.

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

ISO/IEC/IEEE. 2017. *Systems and software engineering ISO/IEC/IEEE 24765:2017.* 2017.

JetBrains. Kotlinlang. [En línea] JetBrains. [Citado el: 19 de Abril de 2019.] https://kotlinlang.org/docs/reference/coding-conventions.html?hl=es_419.

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

MoleQla: revista de Ciencias de la Universidad Pablo de Olavide, Patrón Modelo-Vista-Presentador (MVP). **Diéguez, D.S. 2015.** 20, 2015.

Norma ISO/IECE 9126. 2000. *Norma ISO/IECE 9126.* 2000.

OWASP, The Open Web Application Security Project. 2017. Mobile Top 10 2016-Top 10. [En línea] 13 de Febrero de 2017. [Citado el: 12 de Junio de 2019.] https://www.owasp.org/index.php/Mobile_Top_10_2016-Top_10.

Pressman, Roger S. 2010. *Ingeniería de software un enfoque práctico Séptima edición.* 2010.

Rajagopal, D. y Thilakavalli, K. 2017. A Study: UML for OOA and OOD. 2017.

Sánchez, Tamara Rodríguez. 2015. *Metodología de desarrollo para la Actividad productiva de la UCI.* 2015.

Saria Preval, Ing. Katia . 2014. Evaluación de Requisitos.Proyecto Arquitectura de referencia para PHP [marco de trabajo Bosón]. 21 de 11 de 2014.

Silega Martínez, Nemurys, Bravo Batista, Claudia y Treto Portal, Janier. 2017. *Solución para lograr la interoperabilidad de Odoo con otros sistemas.* La Habana,Cuba : s.n., 2017.

Sommerville, Ian. 2005. *Ingenieria del Software Séptima Edición.* 2005.

StatCounter. 2019. GlobalStats. [En línea] 25 de Mayo de 2019. http://gs.statcounter.com/os-market-share/mobile/cuba.

Visual Paradigm . 2018. Visual Paradigm - Leading UML, BPMN, EA, Agile and Project Management Software. [En línea] 2018. https://www.visual-paradigm.com/.

Wordpress. 2019. Wordpress. [En línea] 25 de 5 de 2019. https://developer.wordpress.org/rest-api/.

XML | Object Management Group. 2018. XML | Object Management Group. [En línea] 2018. [Citado el: 1 de abril de 2018.] http://www.omg.org/technology/readingroom/XML.htm.