



**Facultad de Ciencias y Tecnologías Computacionales**

# **Aplicación móvil para la gestión de inventario de Activos Fijos Tangibles en la Residencia Estudiantil #2 de la Universidad de las Ciencias Informáticas**

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

**Autor:** Andy Jesús Santana Sánchez

**Tutor(es):** Ing. Leonel Eduardo Perdomo Roig

La Habana, noviembre de 2022

Año 64 de la Revolución



## DECLARACIÓN DE AUTORÍA

Yo, Andy Jesús Santana Sánchez con Carnet de Identidad: 97101406360 declaro ser el único autor del presente trabajo de diploma titulado “ **Aplicación móvil para la gestión de inventario de Activos Fijos Tangibles en la Residencia Estudiantil #2 de la Universidad de las Ciencias Informáticas** ” y concedo a la Universidad de Ciencias Informáticas (UCI) los derechos Patrimoniales de la misma en uso de su beneficio. Para que así conste firma(n) la presente a los <día> días del mes de <mes> del año <año>.

<nombre del autor>

---

Firma del Autor

<nombre del tutor>

---

Firma del Tutor

<nombre del autor>

---

Firma del Autor

<nombre del tutor>

---

Firma del Tutor

## DATOS DE CONTACTO

### **Autores:**

Andy Jesús Santana Sánchez

Universidad de la Ciencias Informáticas (UCI)

e-mail: [ajsantana@estudiantes.uci.cu](mailto:ajsantana@estudiantes.uci.cu)

### **Tutor:**

Leonel Eduardo Perdomo Roig

Universidad de la Ciencias Informáticas (UCI)

e-mail: [leo@uci.cu](mailto:leo@uci.cu)

## AGRADECIMIENTOS

*Mis agradecimiento a mis padres por inculcar en mi la necesidad de superarse, de no quedar siendo un mediocre.*

*Agradezco a la profesora de Calculo 1 Dianelys por abrirnos los ojos desde el primer encuentro  
Agradezco Infinitamente en el paso por la universidad haber conocido a hermanos como Daniel Rodríguez, Reniel Rodríguez, Eliany Nodarse, Danaysi la profe de cálculo, al Migue, Carlos el Suave, El Figaro, Verenis.*

*Agradezco a Carlos Figueroa mi amigo y profesor de Matemática Discreta.*

*Agradezco a mis hermanos Dariel Lazo, Roger Enrico, Alain Stable cuando me dio la perreta de irme ahí estuvieron para evitarlo pero más que eso sacamos a flote el titanic.*

*A mi Tutor Leo, y sus repasos de Programación 2.*

*Agradezco a mis hermanos de la Facultad 4 Dayan, El Bronw y Héctor.*

*Agradezco a la gente de mi grupo la gente del FI33 y FI34, Andyto el Bonachon, Darien, Ramon, Nataly, El Calvo, Robey, Karla, Raciél, Isabelita, Amaury.*

*Agradezco a la familia del 105205, por los momentos vividos, el sufrimiento con cada examen, a Luis, Andro, Ridel.*

*A mi mejor Amigo Wilder a mi hermano Alejandro por ser incondicionales a nuestra amistad.*

*A mi familia de Los pinos.*

*A mis amistades del barrio, del pre que siempre estuvieron presente, profesores de la vieja guardia que siempre apoyaron mi carrera, a mis vecinos los que siempre preguntan cómo me va.*

*A mis primas, mis tíos, mi abuela Fina, mi primo Alexander, mi abuela Isabel, mi abuelo Jesús.*

*A mis amigos Emilio y Kevin sin sus enseñanzas no hubiera llegado hasta aquí.*

*Por último y no menos importante a mi novia por soportarme, apoyarme, levantarse todos los días con más ánimos que el día anterior, por ser mi equipo por ser mi todo.*

## DEDICATORIA

*Dedicado a mis padres, a mi familia, a mis abuelos sobre todo, a todos mis amigos que apoyaron mi carrera, a mis hermanos adoptivos de la universidad, a mi tutor, dedicado a todos mis amigos que aportaron sus ideas a esta investigación. Dedicado a los que quedaron en el camino, dedicado aquellos que se mostraron negativos también.*

## **RESUMEN**

Desde sus inicios el ser humano siempre busco la manera de tener un control sobre sus medios. Como parte de las normas del sistema de control interno cubano, se establece el 10% de los Activos Fijos Tangibles (AFT) sean auditados mensualmente. La Universidad de las Ciencias Informáticas dentro de su campus posee una amplia variedad de recursos ya sean tangibles o intangibles. En el caso de los tangibles constituyen un gran volumen, por lo que el proceso de documentar la existencia de los mismos suele ser bastante complicado, saber qué cantidad existe, a qué lugar corresponden entre otros aspectos, que se resumen en una auditoria. En la residencia #2 se realizan las acciones antes descritas de manera manual, contra un inventario previamente impreso, lo que genera un proceso engorroso e ineficiente. La investigación que le sigue a continuación consiste en el análisis, diseño, implementación y evaluación de una aplicación móvil que busca darle solución de una manera ágil a los procesos antes descritos. El desarrollo de la misma se establece a partir de uso de herramientas de código abierto, siguiendo la metodología Programación Extrema, Visual Paradigm UML 8.0 como herramienta de modelado, como lenguaje de programación Java 15.0.2, Android Studio 4.1 como IDE de desarrollo, Wamp Server como servidor Apache, usando JDBC como API y SQL para las consultas a la Base de Datos. Se obtiene como resultado una aplicación móvil con una arquitectura sólida implementada sobre los patrones GRASP y GOF, asegurando su desempeño con la ejecución de pruebas unitarias, de sistema y aceptación las cuales arrojaron resultados positivos.

## **PALABRAS CLAVE**

Activo Fijo Tangible, Control, Aplicación Móvil, residencia #2

## **ABSTRACT**

*Since its inception, human beings have always sought ways to have control over their means. As part of the rules of the Cuban internal control system, it is established that 10% of the Tangible Fixed Assets (AFT) are audited monthly. The University of Informatics Sciences within its campus has a wide variety of resources, whether tangible or intangible. In the case of tangibles, they constitute a large volume, so the process of documenting their existence is usually quite complicated, knowing how much exists, where they belong, among other aspects, which are summarized in an audit. In residence #2, the actions described above are carried out manually, against a previously printed inventory, which generates a cumbersome and inefficient process. The investigation that follows consists of the analysis, design, implementation and evaluation of a mobile application that seeks to provide an agile solution to the processes described above. Its development is established from the use of open source tools, following the Extreme Programming methodology, Visual Paradigm UML 8.0 as a modeling tool, Java 15.0.2 as a programming language, Android Studio 4.1 as a development IDE, Wamp Server as Apache server, using JDBC as API and SQL for queries to the Database. As a result, a mobile application with a solid architecture implemented on the GRASP and GOF patterns is obtained, ensuring its performance with the execution of unit, system and acceptance tests, which yielded positive results.*

### **KEYWORDS**

*Tangible Fixed Assets, Control, Mobile Application, residence #2*



## TABLA DE CONTENIDOS

### Contenido

CAPÍTULO 1: Fundamentos y referentes teórico-metodológicos del proceso de gestión de inventario de Activos Fijos Tangibles en la Residencia.....	16
1.1 Conceptos Asociados a la Investigación.....	16
1.2 Estudio del Estado del Arte.....	18
Conclusiones sobre el estudio de soluciones similares.....	21
1.3 Herramientas.....	22
1.3.1 Herramienta CASE.....	22
1.3.2 Herramientas del lado del Servidor.....	22
1.3.3 Herramientas del lado del cliente.....	23
1.3.4 Lenguaje de Programación.....	23
1.4 Metodología.....	24
1.4.1 ¿Qué es una Metodología de desarrollo de software?.....	24
1.4.2 Metodologías ágiles.....	24
1.4.3 Metodología Programación Extrema (XP).....	25
Conclusiones del capítulo.....	28
CAPÍTULO II: DISEÑO DE LA SOLUCIÓN PROPUESTA.....	29
Introducción.....	29
2.1 Descripción del Proceso.....	29
2.1.2 Modelo de Dominio.....	30
2.1.3 Descripción de los Conceptos del Modelo de Dominio.....	31
2.2 Requisitos de Software.....	31
2.2.1 Requisitos Funcionales.....	32
2.2.2 Requisitos no Funcionales.....	34
2.2.3 Historias de Usuario.....	36
2.3 Diseño de la APK ResideM.....	40
2.3.1 Diseño Arquitectónico.....	40
2.3.2 Mapa de Navegabilidad.....	41
2.3.3 Tarjeta Clase Responsabilidad Colaborador CRC.....	42
2.3.4 Patrones de Diseño.....	43
2.2.3 Conclusiones del capítulo.....	45
CAPÍTULO III: VALIDACIÓN DE LA SOLUCIÓN PROPUESTA.....	46
Introducción.....	46
3.1 Plan de Iteraciones.....	46
3.2 Interfaz de comunicación.....	47
3.3 Diagrama de Componentes.....	48
3.4 Pruebas de Software.....	48
3.4.1 Pruebas de Caja Negra.....	49
Conclusiones del capítulo.....	55
CONCLUSIONES FINALES.....	56
RECOMENDACIONES.....	57
REFERENCIAS BIBLIOGRÁFICAS.....	58
ANEXOS.....	61

## ÍNDICE DE FIGURAS

Figura 1. Solución propuesta. Fuente: elaboración propia.....	30
Figura 2. Modelo Conceptual. Fuente: elaboración propia. ....	31
Figura 3. Diagrama de paquetes. Fuente: Elaboración propia.....	41
Figura 4. Mapa de navegabilidad. Fuente: elaboración propia. ....	42
Figura 5. Diagrama de Componentes. Fuente: Elaboracion propia .....	48
Figura 6. Resultado de la aplicación de la prueba unitaria .....	51
Figura 7. Resultado de la aplicación de la prueba unitaria .....	52
Figura 8. No conformidades. Fuente: Elaboracion propia.....	55

## ÍNDICE DE TABLAS

Tabla 1. Sistemas homólogos estudiados, tabla comparativa de elaboración propia. ....	21
Tabla 2. Historia de usuario requisito funcional Insertar Edificio .....	38
Tabla 3. Historia de usuario requisito funcional Insertar Apartamento .....	39
Tabla 4. Tarjeta CRC Insert_building_activity. Fuente elaboración propia .....	42
Tabla 5. Tarjeta CRC Insert_apartment_Activity. Fuente Elaboracion Propia .....	43
Tabla 6. Plan de Iteraciones. Fuente de Elaboracion Propia. ....	46
Tabla 7. Caso de Prueba del Requisito Insertar Medio Básico.....	50

**OPINIÓN DEL(OS) TUTOR(ES)**

<Contenido de la opinión de los tutores>

## **AVAL DEL CLIENTE**

<Contenido del aval del cliente sobre la solución desarrollada>

## INTRODUCCIÓN

Desde sus inicios, el ser humano, controla las actividades que lo rodean, por lo que es posible decir que el control es inherente al mismo. Debido al descubrimiento de prácticas delictivas, malversaciones y pagos ilegales, en los negocios, se le presta gran atención a la necesidad de elevar las exigencias en los controles internos y se comenzaron a ejecutar una serie de acciones en diversos países desarrollados con el fin de dar respuestas a inquietudes sobre la diversidad de conceptos e interpretaciones que existían sobre el control interno en el ámbito internacional.

En la actualidad, a nivel mundial las organizaciones se mueven mediante procesos y de ahí la necesidad de controlar cada proceso para que este se desarrolle de manera eficiente y eficaz. Por lo que es significativo el control interno aplicado a manuales de procedimientos, los que al mismo tiempo son la mejor herramienta, idónea para plasmar todas las actividades específicas dentro de una organización y establecer responsabilidades de los funcionarios, para el cumplimiento de los objetivos [1].

Las Tecnologías de la Información y las Comunicaciones(TIC) derivadas del Internet han generado una transformación significativa que va más allá de las transacciones que efectúan las entidades económicas al desempeñar actividades empresariales, es decir, han tenido un impacto en la vida de los individuos, al grado de modificar sus actividades cotidianas, esto se evidencia en la manera en que el ser humano puede adquirir un bien o servicio en la actualidad, los medios utilizados para establecer relaciones interpersonales y las herramientas que se emplean para llevar a cabo investigaciones.

Las TIC son una herramienta fundamental hoy en día para cualquier negocio sin importar el lugar del mundo en el que se ubique. Se han vuelto muy importantes para cualquier entidad económica en este siglo, ya que son necesarias para poder ejecutar la operación del negocio[2].

Cuba es un país que se encuentra bajo un proceso de informatización de la sociedad, conformando así un nuevo modelo socio-técnico que es impulsado por el desarrollo de las TIC con el objetivo de simplificar complejos procesos en las distintas esferas de las ciencias. Esto trae como resultado una evolución para la realización de determinadas tareas en las esferas

de la sociedad haciendo natural el uso de técnicas, herramientas y métodos vinculados a las TIC.

Dentro de los sectores en los que se hace imprescindible el uso de las TIC tenemos al empresarial, donde su uso evidencia el perfeccionamiento e innovación en sus procesos, tanto negocios, como administrativos. Las soluciones informáticas llegaron para establecer un estrecho vínculo entre proveedores y clientes permitiendo una mejor toma de decisiones, dándole mayor seguridad y credibilidad a la información.

En cuanto al sector empresarial son diversas las cuestiones a informatizar, entre ello hoy se enfoca la perspectiva al control de los medios existentes. El control no es más que un plan donde los actores económicos prefieren precaver antes de lamentar pérdidas con el fin de cumplir sus objetivos, de ahí entonces se establece que el control interno no es más que un plan de políticas y procedimientos que persigue una entidad para salvaguardar sus recursos. La efectividad de esta idea se centra en establecer el control como único modo de proteger los recursos contra pérdida, fraude o ineficiencia.

Las herramientas de gestión y Control de Activos Fijos Tangibles (AFT) ayudan a las empresas a supervisar, saber con cuántos y cuáles recursos realmente cuenta. Esto especialmente porque toda compañía se enfrenta a muchos requisitos tecnológicos obligatorios para satisfacer las exigencias del mercado actual. Desde luego, mientras más grande sea nuestra empresa, mayor será el volumen de AFT que se debe controlar y conocer.

Las empresas, organizaciones y entidades cubanas, como base y sostén de la economía del país, son las encargadas de llevar el control de sus AFT. En estos momentos la gestión de los AFT de la residencia estudiantil #2 de la Universidad de Ciencias Informáticas (UCI) se realiza de forma manual por parte del personal encargado de manejar la información, almacenando los datos en formato duro, lo que resulta muy engorroso a la hora de manipular los datos, ya sea para realizar una búsqueda, modificar algún dato, obtener reportes, realizar revisiones, afectando así la calidad y rapidez del proceso. Todo esto conlleva a que se produzcan una serie de irregularidades entre las que se incluyen, en el peor de los casos, pérdidas de los medios y bienes debido a la mala planificación y asignación de los mismos.

Todo esto puede traer consigo:

1. La búsqueda y recuperación de datos para la generación de reportes es lenta.

2. Gastos innecesarios o pérdidas de los medios y bienes debido a la mala planificación y asignación de los mismos.
3. Dificultad para llevar el control de las entradas y salidas de los medios en el almacén.
4. Pérdida de información asociada a la gestión de inventario de los activo fijo tangible debido al deterioro de los documentos archivados.
5. El ineficiente control del nivel de acceso a los documentos perjudica la confidencialidad e integridad de los datos.
6. Demora e ineficiencia en el registro de los movimientos de activo fijo tangible por la incapacidad de realizarlo en tiempo real.

Lo antes expuesto, dificulta y demora la gestión de inventario de los activos fijo tangible a los trabajadores de la residencia lo cual conlleva a la búsqueda de una solución al siguiente **problema de investigación**: ¿Cómo agilizar los procesos de gestión de inventario de activo fijo tangible en la residencia estudiantil #2 de la Universidad de las Ciencias Informáticas?

Partiendo del problema expuesto el **objeto de estudio**: El proceso de gestión de activos fijos tangibles mediante la incorporación de dispositivos móviles.

Para resolver el problema planteado se establece como **objetivo general**: Desarrollar una aplicación móvil para la gestión de inventario de activo fijo tangible en la residencia estudiantil #2 que permita agilizar este proceso.

Enmarcándose en el **campo de acción**: Aplicaciones móviles para la gestión de inventario de activos fijos tangibles.

Partiendo del análisis del objetivo general se derivan los siguientes **objetivos específicos**:

1. Elaborar el marco teórico de la investigación.
2. Diseñar las funcionalidades de la aplicación móvil de gestión de inventario de activo fijo tangible.
3. Implementar las funcionalidades del sistema de gestión de inventario de activo fijo tangible.
4. Probar las funcionalidades del sistema de gestión de inventario de activo fijo tangible.

Con el propósito de dar cumplimiento a los objetivos planteados se definen las siguientes **tareas de investigación**:

1. Elaborar la fundamentación teórica de la investigación.
2. Estudio diagnóstico de sistemas informáticos para la gestión de inventario.

3. Análisis de las herramientas, tecnologías y metodología a emplear durante el desarrollo del sistema.
4. Modelado del dominio.
5. Análisis de los requisitos funcionales y no funcionales identificados del sistema.
6. Diseño de la aplicación móvil para gestionar activos fijos tangibles.
7. Implementación de la aplicación móvil para gestionar activos fijos tangibles.
8. Validación de la aplicación móvil para gestionar activos fijos tangibles.
9. Pruebas de la aplicación móvil para gestionar activos fijos tangibles.

La investigación se basa en la siguiente idea a defender: El desarrollo una aplicación móvil permitirá agilizar los procesos de gestión de inventario de activo fijo tangible que tienen lugar en la residencia estudiantil #2 de la Universidad de las Ciencias Informáticas.

**Palabras claves:** activo fijo tangible, inventario, sistema de gestión

Para el desarrollo de esta investigación se proponen los siguientes métodos de investigación:

**Métodos Teóricos:**

**Histórico-Lógico:** Mediante este método se analiza la trayectoria real de los fenómenos, se comprenden los antecedentes, las tendencias, su evolución y desarrollo. Se identifican procesos relacionados a la gestión de los activo fijo tangible, además de realizar una investigación sobre sistemas homólogos al que se desea implementar.

**Analítico-Sintético:** Es un proceso mediante el cual se relacionan hechos aparentemente aislados y se formula una teoría que unifica los diversos elementos. Consiste en la reunión racional de varios elementos dispersos en una nueva totalidad, este se presenta más en el planteamiento de la hipótesis. El investigador sintetiza las superaciones en la imaginación para establecer una explicación tentativa que someterá a prueba [3]. Se analiza de la información referente a los activo fijo tangible en cuestión, tomar conceptos fundamentales, se realiza una síntesis de los elementos más relevantes, se organiza y se le da una estructura.

**Modelación:** Es justamente el método mediante el cual se crean abstracciones con vistas a explicar la realidad. El modelo como sustituto del objeto de investigación. El modelo revela la unidad de lo objetivo y lo subjetivo. La modelación es el método que opera en forma práctica o teórica con un objeto, no en forma directa, sino utilizando cierto sistema intermedio, auxiliar,



natural o artificial [3]. Se utiliza para crear el proceso de diseño y modelar los diagramas asociados al objeto.

### **Métodos Empíricos:**

**Entrevista Semiestructurada:** La entrevista es una técnica de recopilación de información mediante una conversación profesional, con la que además de adquirirse información acerca de lo que se investiga, tiene importancia desde el punto de vista educativo; los resultados a lograr en la misión dependen en gran medida del nivel de comunicación entre el investigador y los participantes en la misma [3]. (Ver Anexo 1)

**Observación:** La observación científica como método consiste en la percepción directa del objeto de investigación. La observación investigativa es el instrumento universal del científico. La observación permite conocer la realidad mediante la percepción directa de los objetos y fenómenos. La observación, como procedimiento, puede utilizarse en distintos momentos de una investigación más compleja: en su etapa inicial se usa en el diagnóstico del problema a investigar y es de gran utilidad en el diseño de la investigación [3].

Se utiliza este método para apreciar de manera común como funciona el control de los activos fijo tangible, de esta manera se obtiene la información que permite conocer la actual situación en la residencia, el comportamiento de los fenómenos cuestionados, contribuyendo en la búsqueda de una solución factible.

### **Estructura de la investigación:**

**Capítulo1:** Este capítulo plantea todos los referentes teóricos, conceptos, definiciones acordes con la investigación, se realiza con el objetivo de estudiar sistemas informáticos similares al que se desea implementar con el fin de obtener de ellos rasgos que sean adaptables, que permitan hacer una aplicación competente a la realidad del mercado. Se describen las principales tecnologías, herramientas y metodología de desarrollo.

**Capítulo2:** En este capítulo se modela el dominio como resultado del primer contacto con el cliente, se especifica los requisitos funcionales y no funcionales, se describen los artefactos generados en correspondencia con la metodología seleccionada, se lleva a cabo el estudio de los patrones de diseño que conforman la aplicación en cuestión.

**Capítulo3:** En este capítulo se incluye la realización de pruebas de validación, tratamiento de errores, se verifica la calidad del sistema implementado con las pruebas correspondientes.

## **CAPÍTULO 1: Fundamentos y referentes teórico-metodológicos del proceso de gestión de inventario de Activos Fijos Tangibles en la Residencia.**

En este capítulo se trata sobre los principales conceptos implicados en la investigación, asociados al proceso de control de AFT como objeto de estudio. Con dicha investigación se determina, se valora, se compara, se critican posibles caminos que ofrecen soluciones al problema en cuestión. El análisis también obliga a tomar decisiones argumentadas sobre las principales herramientas, tecnologías, metodologías y/o lenguajes que se utilizan para el desarrollo del proyecto.

### **1.1 Conceptos Asociados a la Investigación**

#### **Sistema de Gestión**

El sistema de gestión es la herramienta que permite controlar los efectos económicos y no económicos de la actividad de la empresa. El control, en este caso, se define como aquella situación en que se dispone de conocimientos ciertos y reales de lo que está pasando en la empresa, tanto internamente como en su entorno y permite planificar, en cierta manera, lo que pasará en el futuro. Mide el aprovechamiento eficaz y permanente de los recursos que posee la empresa para el logro de sus objetivos [4].

#### **Sistema Informático**

El sistema informático de una empresa es un subsistema dentro del sistema de información de la misma, y está formado por todos los recursos necesarios para dar respuesta a un tratamiento automático de la información y aquellos otros que posibiliten la comunicación de la misma [5].

#### **Activo Fijo Tangible**

Incluye todos aquellos bienes y materiales tangibles, es decir, se pueden tocar. En función de las características del negocio los activos fijos podrán variar de manera notoria. Según la clasificación establecida por el Plan General Contable, algunos de los bienes tangibles de los que pueden disfrutar las empresas serían los siguientes:

Terrenos y bienes naturales: Aquellos terrenos y solares que posea la empresa, ya sean urbanos o no.

Construcciones: Hace referencia a todo tipo de inmuebles en general y que son propiedad de la organización, como edificios, naves, pisos o locales.

Instalaciones técnicas: Este concepto hace alusión a todos aquellos elementos que, en conjunto, constituyen una unidad de uso especializada necesaria para la actividad de la empresa. Se trata de montajes en cadena y otro tipo de construcciones similares.

Maquinaria: Dentro de este apartado se incluyen todas aquellas máquinas, vehículos industriales y herramientas necesarias para la actividad cotidiana.

Mobiliario: Todas las estanterías, mesas, sillas, mostradores y demás muebles que posee la empresa.

Equipos para procesos informáticos: Compuesto por ordenadores, impresoras, escáner y demás aparatos electrónicos.

Elementos de transporte: Dentro de esta categoría se encuentran todos los medios de transporte que forman parte de los bienes de la compañía, como coches, camiones, motos, barcos, etc., utilizados para el transporte de personas, mercancías, materiales o animales.

Otros: Aquellos bienes que no se puedan incluir dentro de ninguna de las categorías anteriores.

### **Inventario**

Es la verificación y control de los materiales o bienes patrimoniales de la empresa, que se realiza para regularizar la cuenta de existencias contables y calcular pérdidas o beneficios [6].

### **Medio Básico**

Se consideran como medios básicos al conjunto de medios de trabajo destinados a la producción material o a la prestación de servicios, que conservan su vida útil por más de un año y que transfieren gradualmente su valor al producto o servicio que prestan según su naturaleza y ubicación en la esfera productiva e improductiva.

Los medios básicos se registran por su valor inicial, el que se conforma por el precio de adquisición o de construcción y los gastos de transportación y montaje, manteniéndose inalterable durante toda su vida útil [7].

### **Aplicación móvil**

Las aplicaciones móviles son programas diseñados para ser ejecutados en teléfonos, Tablet y otros dispositivos móviles, que permitan al usuario realizar actividades profesionales, acceder a servicios, mantenerse informado, entre otro universo de posibilidades .

## **Android**

Android es un sistema operativo, inicialmente diseñado para teléfonos móviles como los sistemas operativos iOS (Apple), FireFoxOS (Mozilla) y Blackberry OS. En la actualidad, este sistema operativo se instala no sólo en móviles, sino también en múltiples dispositivos, como tabletas, GPS, televisores, discos duros multimedia, mini ordenadores, incluso se ha instalado en microondas y lavadoras [8].

Android es un sistema operativo móvil que se basa en una versión modificada de Linux. Fue desarrollado originalmente por una startup del mismo nombre, Android, Inc. en 2005, como parte de su estrategia para ingresar al espacio móvil, Google compró Android, Inc. y se hizo cargo de su trabajo de desarrollo (así como de su equipo de desarrollo). Google quería que el sistema operativo Android fuera abierto y gratuito, por lo que la mayor parte del código de Android se publicó bajo la licencia Apache de código abierto [9].

### **1.2 Estudio del Estado del Arte**

El estudio del arte se hace con el objetivo de investigar soluciones antes desarrolladas, aprovechando de estas sus mejores ideas e innovaciones, además de atribuirle al sistema en desarrollo ideas que son reutilizables, aportándole cualidades comunes, necesarias ya planteadas por otros desarrolladores las cuales constituyen patrones siempre y cuando se puedan adaptar a esta nueva solución. De esta manera los sistemas que se estudian en la presente investigación forman parte del apoyo teórico para el objetivo de la solución propuesta.

## **Distra**

Distra es un sistema de gestión empresarial, que permite integrar y optimizar los procesos empresariales, manejar con eficacia la información y disminuir los costos de las operaciones. Se puede trabajar, de forma centralizada, la información relevante de las estructuras organizativas e integrar todos los procesos de trabajo. Está compuesto por diversos subsistemas y módulos, que cubren las principales áreas de gestión de cualquier entidad. Es una oferta muy útil para las medianas y grandes empresas, que poseen varias entidades subordinadas y para aplicar el teletrabajo, en cualquiera de las áreas de gestión de la entidad.

## **AUDITA**

Es un sistema capaz de agilizar el proceso de auditoría y control de los AFT. Se basa en un sistema de escritorio multiplataforma con comunicación mediante servicios al SGU. La

aplicación es capaz de, ante la selección de un área determinada a auditar, obtener el listado de medios y servirse de un dispositivo o scanner de barras para apoyar el testeado de los medios, ofreciendo de manera automática los resultados de la auditoria a manera de informe, con gráficos y otros elementos que apoyen la toma de decisiones [10].

### **VERSAT**

Su utilización, que comenzó en julio de 2001 en el central George Washington, del municipio villaclareño de Santo Domingo, permite la planificación y control de todos los recursos humanos, materiales y financieros de cualquier tipo de entidad, tanto del sistema empresarial como presupuestado.

Miguel Pascual Cabrera González, autor principal de esta herramienta informática, destacó que fue creada por un grupo villaclareño de especialistas económicos e informáticos pertenecientes a la Empresa de Servicios Técnicos Industriales del Grupo Azucarero Nacional. Subrayó que el VERSAT Sarasola consta de 12 módulos o subsistemas de configuración, contabilidad y costos; finanzas, inventarios, activos fijos; nóminas, planificación, facturación, mensajería, auditoría y generador de reportes [11].

### **Software Advance Assets**

El Sistema de Gestión Integral (ASSETS) actualmente es usado en la UCI, por la Dirección de economía. ASSETS NS es un sistema de Gestión Integral estándar y parametrizado que Sistema informático para la gestión de la información de AFT, los Útiles y Herramientas de la Residencia de la UCI permite el control de los procesos de Compras, Ventas, Producción, Taller, Inventario, Finanzas, Contabilidad, Presupuesto, Activos Fijos, Útiles y Herramientas y Recursos Humanos. Como sistema Integral todos sus módulos trabajan en estrecha relación, generando, automáticamente, al módulo de contabilidad los comprobantes de operaciones por cada una de las transacciones efectuadas, esto permite que se pueda trabajar bajo el principio de contabilidad al día. Este sistema multiusuario está montado en una plataforma de servidores SQL, dividido en módulos económicos que trabajan en conjuntos para el control de las actividades económicas, financieras y contables sobre los medios materiales y financieros. Al estar en plataforma SQL, garantiza mayor seguridad y consistencia en los datos, se obliga que sea ilimitado el número de usuarios conectados y hace posible la utilización de servidores

remotos. Facilita, además, el ingreso al inventario y también la administración de los códigos asignados y todo el manejo internamente requerido [12].

### **SoftExpert Activo**

SoftExpert Activo es un software corporativo proyectado para ayudar en la gestión de todos los activos de la empresa – desde máquinas importantes a computadoras y vehículos. El software garantiza que se seleccione el equipamiento correcto de acuerdo con las necesidades de cada tarea, se mantenga en funcionamiento por el mayor tiempo posible y se sustituya de manera organizada, evitando fallas e interrupciones en los procesos de negocio.

También mantiene registros detallados de los cronogramas de servicio, localización, verificación, mantenimiento y provee recursos de búsqueda avanzados que permitan listar y organizar los activos por categoría, estatus, condición de uso, entre otros. SoftExpert Activo permite un equilibrio estratégico entre disponibilidad y uso de activos, además de integrarse a otros sistemas corporativos.

### **Asset Panda**

Es un software de gestión y seguimiento de activos que nunca permite perder ningún registro o pista al adaptarse al trabajo. Puede rastrear cualquier cosa desde una computadora portátil y configurarse en la forma en que administra un equipo actualmente. Con un mejor seguimiento de los activos, puede tener un mejor rendimiento comercial, más ahorros y menos pérdidas de activos. El software consolidado de seguimiento de activos de Asset Panda puede mostrar información de garantía, historiales de acciones, fotos y manuales de usuario, todo en un solo lugar. También puede compartir los datos de un equipo en la sala de juntas o en el almacén de una organización para ponerlos en la misma página. Las pistas de auditoría completas permiten mejorar la previsión y la rendición de cuentas de cada activo.

Todos los interesados obtendrán información sobre la mejora de la planificación y la eficiencia de forma automática. Además, obtendrá un acceso más fácil a los datos vitales, un mejor autoservicio de los empleados y una identificación de activos más rápida. Puede utilizar el escáner de código de barras integrado, compatible con muchos dispositivos para evitar equipos de escaneo voluminosos y costosos.

<b>SOFTWARES</b>	<b>GESTIONAN AFT</b>	<b>PLATAFORMA</b>	<b>SOFTWARE LIBRE</b>
Audita	SI	WEB	SI
Versat	SI	WEB	NO
SoftExpert Activo	SI	WEB	NO
Distra	SI	WEB	NO
Asset Panda	SI	WEB , Android	NO
Software Advance Assets	Si	WEB	NO

*Tabla 1. Sistemas homólogos estudiados, tabla comparativa de elaboración propia.*

### **Conclusiones sobre el estudio de soluciones similares**

Después del análisis de los sistemas homólogos, se puede apreciar que Distra nos trae una solución compuesta por diversos módulos o subsistemas que conforman una estructura robusta aplicable a grandes empresas. Audita plantea una solución estática, su objetivo hacia el control de AFT es factible pero, el testeado de medios se puede implementar dentro de una misma aplicación móvil, sin la necesidad de terceros. Versat constituye uno de los softwares de desarrollo cubano más antiguos, sus métodos de agrupación son importantes una característica ya común en otros sistemas dicho software necesita ser renovado. Software Advance Asset es un sistema ideal en cuanto a su alcance pero no fue concebido en la UCI, no es cubano, aunque tiene licencia, no es gratuita y se debe pagar regularmente por lo tanto es de vital importancia acudir al desarrollo de una aplicación autónoma de software libre. SoftExpert Activo es un software menos complicado, basado en la creación de módulos, pero no tiene la facilidad de que los técnicos se desplacen de un lugar a otro dentro de la residencia. Asset Panda ofrece acceso rápido, mantiene un seguimiento de las acciones realizadas, va guardando cada acción que el usuario realiza, teniendo en cuenta las características se

concluye que los mismos no cumplen con las necesidades descritas en la problemática, viendo en ellos como principal problema que son diseñados para la web, por lo que precisan de un equipo de cómputo físico, dicho proyecto necesita de la facilidad de moverse por todas las áreas de la residencia, esto incluye la facilidad de actualizar el estado de los AFT al instante de haberse producido algún cambio durante los chequeos de las áreas.

## **1.3 Herramientas**

### **1.3.1 Herramienta CASE**

Para el modelado con UML se utilizará Visual Paradigm for UML 8.0 por ser una herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Permite modelar todos los tipos de diagramas de clases, generar código desde diagramas y generar documentación. La herramienta agiliza la construcción de aplicaciones con calidad y a un menor coste. Posibilita la generación de bases de datos, transformación de diagramas de Entidad-Relación en tablas de base de datos, así como obtener ingeniería inversa de bases de datos [13].

### **1.3.2 Herramientas del lado del Servidor**

Para la creación de la base de datos se utiliza un paquete de instalación con las librerías de las aplicaciones ya compiladas para Windows 10 en arquitectura 64 bits llamado WampServer que incluye PHP 5.4.3, Apache 2.4.2, MySQL 5.5.24 y el gestor phpMyAdmin que serán las herramientas que utilizaremos para crear y gestionar la base de datos[14].

**Apache** es un servidor web de protocolo HTTP que permite la transmisión de información en la World Wide Web, es de código abierto y nos permitirá la utilización de phpMyAdmin a través del navegador web[14] .

**phpMyAdmin** es una herramienta escrita en el lenguaje PHP al que se accede mediante páginas web que garantiza el control de las bases de datos con una interfaz sencilla e intuitiva a la vez que potente. También es completamente gratuito y que ofrece una vasta cantidad de características y opciones para manejar la base de datos [14].

**WampServer**, es una aplicación gratuita que permite la creación de sitios web considerando el uso de la base de datos[15].



**Hypertext Preprocessor** (PHP) se utiliza solamente para aplicaciones presentes y que actúan en el lado del servidor capaz de generar contenido dinámico en la World Wide Web. Figura entre los primeros lenguajes posibles para la inserción de documentos HTML, dispensando en muchos casos el uso de archivos externos para eventuales procesamientos de datos. El código es interpretado en el lado del servidor por el módulo PHP, que también genera la página web para ser visualizada en el lado del cliente. El lenguaje evolucionó, pasó a ofrecer funcionalidades en la línea de comandos, y además, ganó características adicionales, que posibilitaron usos adicionales del PHP [16].

**Java Database Connectivity** (JDBC) es un derivado inspirado en el mismo, una interfaz de programación de aplicaciones que permite la ejecución de operaciones sobre bases de datos desde el lenguaje de programación Java independientemente del sistema operativo donde se ejecute o de la base de datos a la cual se accede utilizando el dialecto SQL del modelo de base de datos que se utilice. Una dificultad enfrentada por los desarrolladores de JDBC fue que existen muchos proveedores de bases de datos cada uno usando su propio protocolo. Es así como se acordó el desarrollo de una API Java para SQL, la cual accede la base de datos vía un administrador de drivers de terceros los cuales se conectan a bases de datos específicas. Cada proveedor de bases de datos debía generar su propio driver conectable al administrador de drivers[17].

### 1.3.3 Herramientas del lado del cliente

#### **Android Studio v4.1**

Este IDE fue desarrollado por google compañía propietaria de Android, es nombrado oficial como el IDE oficial para el desarrollo de aplicaciones Android. Está basado en IntelliJ, cuenta con diferencias notables en comparación con Eclipse, cambian algunas cosas como la compilación, el uso de librerías o paquetes externos, pero el código de una aplicación en su mayoría funciona de la misma forma sin importar el IDE que se esté utilizando[18].

### 1.3.4 Lenguaje de Programación

#### **Java**

Es un lenguaje de programación orientado a objetos el cual es uno de los tres más utilizados en la actualidad. Gracias al amplio soporte con el que cuenta, así como también con la gran variedad de clases y colecciones que tiene; Java es uno de los lenguajes más robustos y utilizados en el mundo del desarrollo de software multiplataforma.

Hoy día en la era de los teléfonos inteligentes, el uso de Java en la construcción de aplicaciones para móviles sigue presente; ya que la programación en Android está basada en Java e incluso se puede hacer uso de código de este lenguaje para controlar algunas de las funciones en la aplicación, lo cual da muestra del alcance y poder que tiene Java dentro del mundo de las aplicaciones móviles[19].

## **1.4 Metodología**

### **1.4.1 ¿Qué es una Metodología de desarrollo de software?**

“Conjunto de métodos coherentes y relacionados por unos principios comunes”. El concepto de desarrollo, está vinculado a la acción de desarrollar o a las consecuencias de este accionar, por lo tanto es necesario, rastrear el significado del verbo desarrollar: se trata de incrementar, agrandar, extender, ampliar o aumentar alguna característica de algo físico (concreto) o intelectual (abstracto). Por lo anterior, se concluye que metodología de desarrollo es: el estudio y determinación de cuál es el método más adecuado para dar incremento a algo en este caso al software [20].

### **1.4.2 Metodologías ágiles**

Las metodologías ágiles son flexibles, pueden ser modificadas para que se ajusten a la realidad de cada equipo y proyecto. Los proyectos ágiles se subdividen en proyectos más pequeños mediante una lista ordenada de características. Cada proyecto es tratado de manera independiente y desarrolla un subconjunto de características durante un período de tiempo corto, entre dos y seis semanas. La comunicación con el cliente es constante al punto de requerir un representante de él durante el desarrollo. Los proyectos son altamente colaborativos y se adaptan mejor a los cambios; de hecho, el cambio en los requerimientos es una característica esperada y deseada, al igual que las entregas constantes al cliente y la retroalimentación por parte de él. Tanto el producto como el proceso son mejorados frecuentemente[21].

### 1.4.3 Metodología Programación Extrema (XP)

La Programación Extrema se basa en una serie de reglas y principios que se han ido gestando a lo largo de toda la historia de la ingeniería del software. Usadas conjuntamente proporcionan una nueva metodología de desarrollo software que se puede englobar dentro de las metodologías ligeras, que son aquéllas en la que se da prioridad a las tareas que dan resultados directos y que reducen la burocracia que hay alrededor tanto como sea posible[22].

Es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo. XP se basa en realimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios. XP se define como especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico.

Las historias de usuario son la técnica utilizada en XP para especificar los requisitos del software. Se trata de tarjetas de papel en las cuales el cliente describe brevemente las características que el sistema debe poseer, sean requisitos funcionales o no funcionales. El tratamiento de las historias de usuario es muy dinámico y flexible, en cualquier momento historias de usuario pueden romperse, reemplazarse por otras más específicas o generales, añadirse nuevas o ser modificadas. Cada historia de usuario es lo suficientemente comprensible y delimitada para que los programadores puedan implementarla en unas semanas.

Un proyecto XP tiene éxito cuando el cliente selecciona el valor de negocio a implementar basado en la habilidad del equipo para medir la funcionalidad que puede entregar a través del tiempo. El ciclo de desarrollo consiste (a grandes rasgos) en los siguientes pasos.

El cliente define el valor de negocio a implementar.

El programador estima el esfuerzo necesario para su implementación.

El cliente selecciona qué construir, de acuerdo con sus prioridades y las restricciones de tiempo.

El programador construye ese valor de negocio.

Vuelve al paso 1.

En todas las iteraciones de este ciclo tanto el cliente como el programador aprenden. No se debe presionar al programador a realizar más trabajo que el estimado, ya que se perderá calidad en el software o no se cumplirán los plazos. De la misma forma el cliente tiene la obligación de manejar el ámbito de entrega del producto, para asegurarse que el sistema tenga el mayor valor de negocio posible con cada iteración.

El ciclo de vida ideal de XP consiste de seis fases: Exploración, Planificación de la Entrega (Release), Iteraciones, Producción, Mantenimiento y Muerte del Proyecto[23].

### **Fase I: Exploración**

En esta fase, los clientes plantean a grandes rasgos las historias de usuario que son de interés para la primera entrega del producto. Al mismo tiempo el equipo de desarrollo se familiariza con las herramientas, tecnologías y prácticas que se utilizarán en el proyecto. Se prueba la tecnología y se exploran las posibilidades de la arquitectura del sistema construyendo un prototipo. La fase de exploración toma de pocas semanas a pocos meses, dependiendo del tamaño y familiaridad que tengan los programadores con la tecnología[23].

### **Fase II: Planificación de la Entrega**

En esta fase el cliente establece la prioridad de cada historia de usuario, y correspondientemente, los programadores realizan una estimación del esfuerzo necesario de cada una de ellas. Se toman acuerdos sobre el contenido de la primera entrega y se determina un cronograma en conjunto con el cliente. Una entrega debería obtenerse en no más de tres meses. Esta fase dura unos pocos días.

Las estimaciones de esfuerzo asociado a la implementación de las historias la establecen los programadores utilizando como medida el punto. Un punto, equivale a una semana ideal de programación. Las historias generalmente valen de 1 a 3 puntos. Por otra parte, el equipo de desarrollo mantiene un registro de la "velocidad" de desarrollo, establecida en puntos por iteración, basándose principalmente en la suma de puntos correspondientes a las historias de usuario que fueron terminadas en la última iteración.

La planificación se puede realizar basándose en el tiempo o el alcance. La velocidad del proyecto es utilizada para establecer cuántas historias se pueden implementar antes de una fecha determinada o cuánto tiempo tomará implementar un conjunto de historias. Al planificar por tiempo, se multiplica el número de iteraciones por la velocidad del proyecto, determinándose

cuántos puntos se pueden completar. Al planificar según alcance del sistema, se divide la suma de puntos de las historias de usuario seleccionadas entre la velocidad del proyecto, obteniendo el número de iteraciones necesarias para su implementación[23].

### **Fase III: Iteraciones**

Esta fase incluye varias iteraciones sobre el sistema antes de ser entregado. El Plan de Entrega está compuesto por iteraciones de no más de tres semanas. En la primera iteración se puede intentar establecer una arquitectura del sistema que pueda ser utilizada durante el resto del proyecto. Esto se logra escogiendo las historias que fueren la creación de esta arquitectura, sin embargo, esto no siempre es posible ya que es el cliente quien decide qué historias se implementarán en cada iteración (para maximizar el valor de negocio). Al final de la última iteración el sistema estará listo para entrar en producción.

Los elementos que deben tomarse en cuenta durante la elaboración del Plan de la Iteración son: historias de usuario no abordadas, velocidad del proyecto, pruebas de aceptación no superadas en la iteración anterior y tareas no terminadas en la iteración anterior. Todo el trabajo de la iteración es expresado en tareas de programación, cada una de ellas es asignada a un programador como responsable, pero llevadas a cabo por parejas de programadores. Wake en proporciona algunas guías útiles para realizar la planificación de la entrega y de cada iteración[23].

### **Fase IV: Producción**

La fase de producción requiere de pruebas adicionales y revisiones de rendimiento antes de que el sistema sea trasladado al entorno del cliente. Al mismo tiempo, se deben tomar decisiones sobre la inclusión de nuevas características a la versión actual, debido a cambios durante esta fase.

Es posible que se rebaje el tiempo que toma cada iteración, de tres a una semana. Las ideas que han sido propuestas y las sugerencias son documentadas para su posterior implementación (por ejemplo, durante la fase de mantenimiento) [23].

### **Fase V: Mantenimiento**

Mientras la primera versión se encuentra en producción, el proyecto XP debe mantener el sistema en funcionamiento al mismo tiempo que desarrolla nuevas iteraciones. Para realizar esto se requiere de tareas de soporte para el cliente. De esta forma, la velocidad de desarrollo puede bajar después de la puesta del sistema en producción. La fase de mantenimiento puede requerir nuevo personal dentro del equipo y cambios en su estructura [23].

### **Fase VI: Muerte del Proyecto**

Es cuando el cliente no tiene más historias para ser incluidas en el sistema. Esto requiere que se satisfagan las necesidades del cliente en otros aspectos como rendimiento y confiabilidad del sistema. Se genera la documentación final del sistema y no se realizan más cambios en la arquitectura. La muerte del proyecto también ocurre cuando el sistema no genera los beneficios esperados por el cliente o cuando no hay presupuesto para mantenerlo [23].

### **Conclusiones del capítulo**

En el presente capítulo se realizó el estudio perteneciente a los referentes teóricos y principales conceptos que guían la investigación lo cual posibilita una mayor comprensión del problema a resolver. El estudio de sistemas homólogos arroja como conclusión que ninguno de los sistemas antes planteados satisfacen el problema que dio origen a esta investigación, pero tienen características comunes las cuales aportan al desarrollo del software. Se selecciona las herramientas y las tecnologías, estableciendo la relación de las mismas con la solución propuesta determinando así que dichas herramientas son las adecuadas para el desarrollo de la aplicación aportando rapidez y eficacia durante el proceso de implementación. Dichas herramientas apoyan los proyectos de soberanía tecnológicos impulsados por la universidad. Se selecciona XP como metodología de desarrollo para guiar la investigación.

## CAPÍTULO II: DISEÑO DE LA SOLUCIÓN PROPUESTA

### Introducción

El presente capítulo aborda los procesos que serán objeto de informatización, las principales características del sistema, se expresa una idea de conceptos principales que corresponden a la solución a través de un modelo conceptual, se obtienen los requisitos funcionales y no funcionales, en correspondencia a la metodología seleccionada se realizan ejemplos de historias de usuario, se exponen las responsabilidades de las clases con las tarjetas de clase responsabilidad colaborador. Se describe la arquitectura seleccionada y los patrones de diseño utilizados. Se utiliza el mapa de navegabilidad como herramienta de representación gráfica de los contenidos agrupados en las distintas interfaces disponibles para los usuarios y de los enlaces existentes entre los mismos.

### 2.1 Descripción del Proceso

**Proceso de Gestión de Activos Fijos Tangibles:** Se propone desarrollar un sistema que permita tomar un control exacto de los medios existentes en la residencia además de que permita insertar, modificar, eliminar, chequear de manera ágil y efectiva los mismos. La solución se lleva a cabo a través de una aplicación cliente para dispositivos móviles con sistema operativo Android. En dicho entorno según el rol del usuario se almacena la información perteneciente a todos los AFT que incluye la residencia seleccionada, con la opción de filtrar la búsqueda según el edificio que solicita, el apartamento, el inmueble por su nombre o por su numeración. El sistema abastece también al usuario no solo de búsqueda sino también de la cantidad total del AFT en cuestión según el área seleccionada.

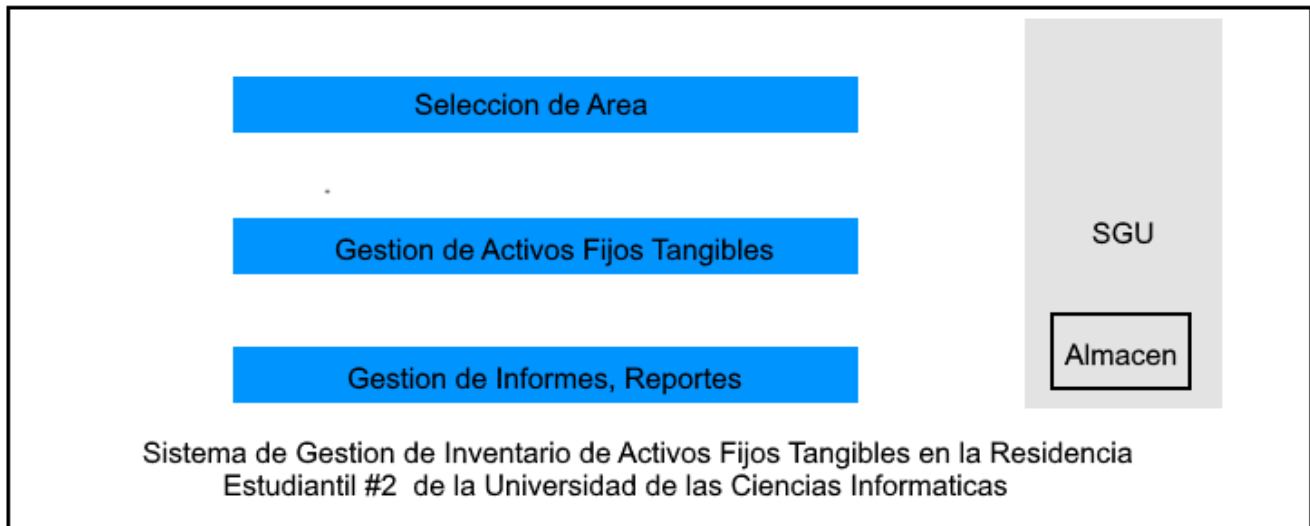


Figura 1. Solución propuesta. Fuente: elaboración propia

### 2.1.2 Modelo de Dominio

Un modelo de dominio es una representación de las clases conceptuales del mundo real, no de componentes software. No se trata de un conjunto de diagramas que describen clases software, u objetos software con responsabilidades. Su utilidad radica en ser una forma de “inspiración” para el diseño de los objetos software. Es entrada para muchos de los artefactos que se construyen en un proceso software. Un modelo de dominio muestra las clases conceptuales significativas en un dominio del problema. Se centra en las abstracciones relevantes, vocabulario del dominio e información del dominio. Es el artefacto clave del análisis orientado a objetos [24].

En UML se utilizan los diagramas de clases para representar los modelos de dominio:



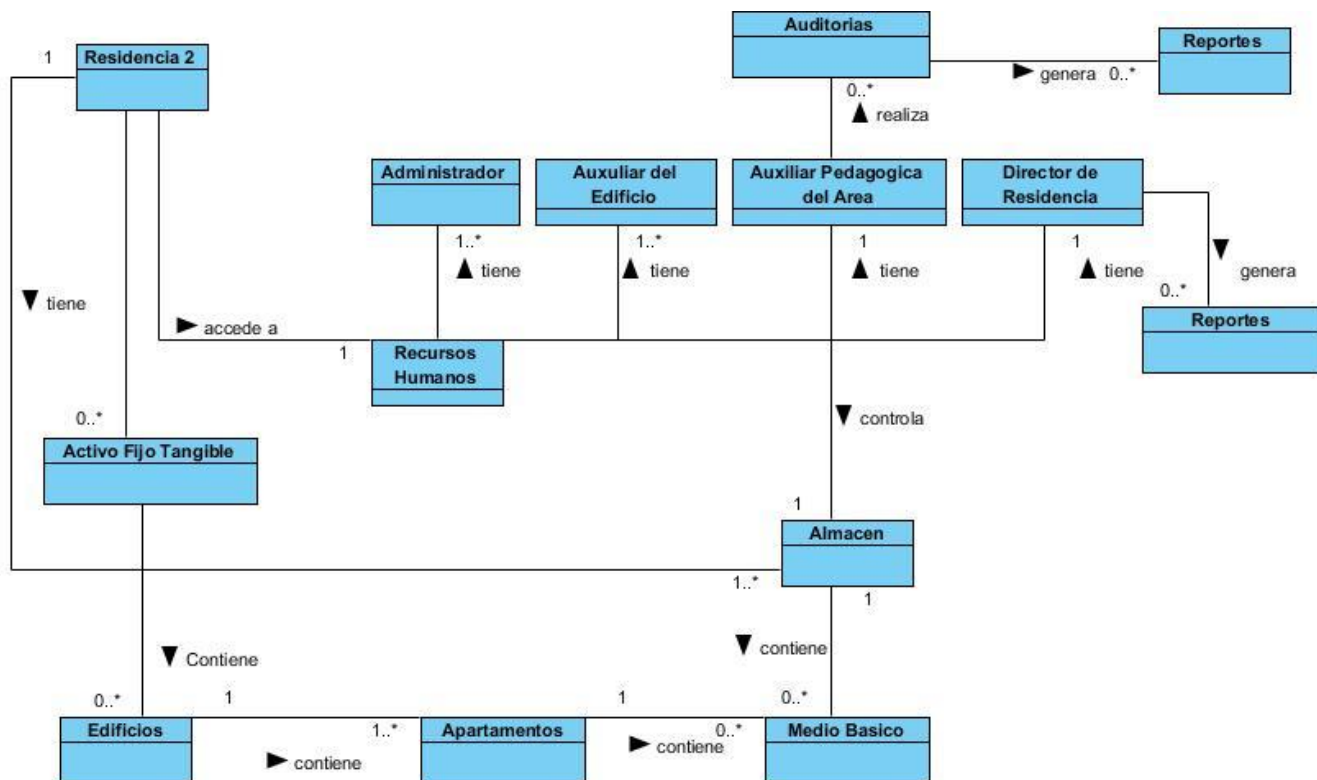


Figura 2. Modelo Conceptual. Fuente: elaboración propia.

### 2.1.3 Descripción de los Conceptos del Modelo de Dominio

**Residencia:** Las Residencias muestran todo lo contenido en ellas, además presentan un apartado de autenticación para el personal de Recursos Humanos el cual tiene los permisos de modificar, insertar o eliminar la información guardada.

**Edificios:** Almacena toda la información referente a los apartamentos.

**Apartamentos:** Almacenan los Activos Fijos existentes en ellos.

**Medios Básicos:** Almacenan su nombre e identificación.

**Recursos Humanos:** Se manejan las opciones que tratan de manera sensible sobre los activos fijos tangibles ya sea control, administración, gestión, contabilidad de los mismos.

**Activos Fijos Tangibles:** Espacio real donde van a estar almacenados todo el mobiliario real de las residencias con sus principales especificación e identificación.

### 2.2 Requisitos de Software

En el proceso de desarrollo de un sistema, el equipo de desarrollo se enfrenta al problema de la identificación de requisitos. La definición de las necesidades del sistema es un proceso

complejo pues en él hay que identificar los requisitos que el sistema debe cumplir para satisfacer las necesidades de los usuarios finales y de los clientes. Para realizar este proceso, no existe una única técnica estandarizada y estructurada que ofrezca un marco de desarrollo que garantice la calidad del resultado. Existe en cambio un conjunto de técnicas, cuyo uso proponen las diferentes metodologías para el desarrollo de aplicaciones móviles. Se debe tener en cuenta que la selección de las técnicas y el éxito de los resultados que se obtengan dependen en gran medida tanto del equipo de análisis y desarrollo como de los propios clientes o usuarios que en ella participen [25].

### 2.2.1 Requisitos Funcionales

No	Nombre	Complejidad
RF1	Autenticar Usuario	Media
RF2	Insertar Usuario	Alta
RF3	Modificar Usuario	Media
RF4	Eliminar Usuario	Media
RF5	Mostrar Usuario	Baja
RF6	Asignar Roles a Usuarios	Media
RF7	Insertar Residencia	Alta
RF8	Modificar Residencia	Media
RF9	Eliminar Residencia	Media
RF10	Mostrar Residencia	Baja
RF11	Insertar Edificios	Alta
RF12	Modificar Edificios	Media
RF13	Eliminar Edificios	Media
RF14	Mostrar Edificios	Baja
RF15	Listar Edificios	Media
RF16	Buscar Edificios	Media
RF17	Insertar Apartamento	Alta
RF18	Modificar Apartamento	Media
RF19	Eliminar Apartamento	Baja

RF20	Mostrar Apartamento	Baja
RF21	Listar Apartamento	Media
RF22	Buscar Apartamento	Media
RF23	Insertar Medio Básico	Alta
RF24	Modificar Medio Básico	Media
RF25	Eliminar Medio Básico	Baja
RF26	Mostrar Medio Básico	Baja
RF27	Listar Medio Básico	Alta
RF28	Buscar Medio Básico	Alta
RF29	Mover Medio Básico	Alta
RF30	Crear modelo de movimiento de medios	Baja
RF31	Visualizar modelo de movimiento de medios	Baja
RF32	Aprobar Movimiento de un Medio Básico	Baja
RF33	Visualizar el listado de movimientos de medios pendientes a aprobación	Baja
RF34	Exportar a formato PDF el listado de movimientos de medios pendientes a aprobación	Baja
RF35	Visualizar las trazas del movimiento de un medio	Baja
RF36	Exportar a formato PDF las trazas del movimiento de un medio	Baja
RF37	Visualizar el listado de movimientos realizados	Baja
RF38	Exportar a formato PDF el listado de movimientos realizados	Baja
RF39	Asignar Medio Básico	Alta
RF40	Insertar AFT	Alta
RF41	Modificar AFT	Medio
RF42	Eliminar AFT	Medio
RF43	Mostrar AFT	Bajo
RF44	Listar AFT	Medio
RF45	Generar informe de auditoria	Bajo
RF46	Exportar en formato PDF auditoria	Bajo

RF47	Crear Reporte	Alto
RF48	Enviar Reporte	Bajo
RF49	Importar los AFT	Medio
RF50	Exportar en formato PDF reporte de activos	Bajo
RF51	Visualizar acta de responsabilidad material	Bajo
RF52	Gestionar Recursos Humanos	Alta
RF53	Exportar en formato PDF acta de responsabilidad material	Bajo
RF54	Visualizar submayor de AFT	Bajo
RF55	Exportar a formato PDF el submayor de AFT	Bajo

### 2.2.2 Requisitos no Funcionales

Los requisitos no funcionales son propiedades o cualidades que el producto debe tener. Forman una parte significativa de la especificación. Son importantes para que clientes y usuarios puedan valorar las características no funcionales del producto, pues si se conoce que el mismo cumple con todas las funcionalidades requeridas, las propiedades no funcionales, como cuán usable, seguro, conveniente y agradable, pueden marcar la diferencia entre un producto bien aceptado y uno con poca aceptación [25].

#### ISO/IEC 25010

El modelo de calidad representa la piedra angular en torno a la cual se establece el sistema para la evaluación de la calidad del producto. En este modelo se determinan las características de calidad que se van a tener en cuenta a la hora de evaluar las propiedades de un producto software determinado.

La calidad del producto software se puede interpretar como el grado en que dicho producto satisface los requisitos de sus usuarios aportando de esta manera un valor. Son precisamente estos requisitos (funcionalidad, rendimiento, seguridad, mantenibilidad, etc.) los que se encuentran representados en el modelo de calidad, el cual categoriza la calidad del producto en características y subcaracterísticas.

### **Adecuación Funcional**

RNF1 La aplicación móvil debe tener la capacidad de proporcionar funciones que satisfacen las necesidades declaradas e implícitas, cuando el producto se usa en las condiciones especificadas.

RNF2 Completitud funcional: Grado en el cual el conjunto de funcionalidades de la aplicación móvil cubren todas las tareas y los objetivos de los usuarios especificados.

RNF3 Corrección funcional: Capacidad de la aplicación móvil para proveer resultados correctos con el nivel de precisión requerido.

### **Eficiencia de desempeño**

Esta característica representa el desempeño relativo a la cantidad de recursos utilizados bajo determinadas condiciones.

RNF4 Comportamiento temporal: Los tiempos de respuesta y procesamiento y los ratios de (*throughput*) de la aplicación móvil cuando lleva a cabo sus funciones bajo condiciones determinadas en relación con un banco de pruebas (*benchmark*) establecido.

RNF5 Utilización de recursos: Las cantidades y tipos de recursos utilizados cuando la aplicación móvil lleva a cabo su función bajo condiciones determinadas.

RNF6 Capacidad: Grado en que los límites máximos de un parámetro de la aplicación móvil cumplen con los requisitos.

### **Usabilidad**

RNF7 Capacidad para reconocer su adecuación: Capacidad de la aplicación móvil que permite al usuario entender si el software es adecuado para sus necesidades.

RNF8 Capacidad de aprendizaje: Capacidad del producto que permite al usuario aprender su aplicación.

RNF 9Capacidad para ser usado: Capacidad de la aplicación móvil que permite al usuario operarlo y controlarlo con facilidad.

RNF10 Protección contra errores de usuario: Capacidad de la aplicación móvil para proteger a los usuarios de hacer errores.

RNF11 Estética de la interfaz de usuario: Capacidad de la interfaz de usuario de agradar y satisfacer la interacción con el usuario.

## **Fiabilidad**

RNF12 Madurez: Capacidad de la aplicación móvil para satisfacer las necesidades de fiabilidad en condiciones normales.

RNF13 Tolerancia a fallos: Capacidad de la aplicación móvil para operar según lo previsto en presencia de fallos hardware o software.

RNF14 Capacidad de recuperación: Capacidad de la aplicación móvil para recuperar los datos directamente afectados y reestablecer el estado deseado del sistema en caso de interrupción o fallo.

## **Seguridad**

RNF15 Capacidad de protección de la información y los datos de manera que personas o sistemas no autorizados no puedan leerlos o modificarlos asegurando así la confidencialidad e integridad de la aplicación móvil

RNF16 Autenticidad: Capacidad de demostrar la identidad de un sujeto o un recurso.

## **Mantenibilidad**

RNF17 La aplicación móvil tiene la capacidad de ser modificada efectiva y eficientemente, debido a necesidades evolutivas, correctivas o perfectivas garantizando así modularidad y reusabilidad.

RNF18 Analizabilidad: Facilidad con la que se puede evaluar el impacto de un determinado cambio sobre el resto del software, diagnosticar las deficiencias o causas de fallos en el software, o identificar las partes a modificar.

## **Portabilidad**

RNF19 La aplicación móvil tiene la capacidad de ser transferido de forma efectiva y eficiente de un entorno hardware, software, operacional o de utilización a otro.

RNF20 Adaptabilidad: La aplicación móvil permite ser adaptada de forma efectiva y eficiente a diferentes entornos determinados de hardware, software, operacionales o de uso.

### **2.2.3 Historias de Usuario**

Las historias de usuario se usan, en el contexto de la ingeniería de requisitos ágil, como una herramienta de comunicación que combina las fortalezas de ambos medios: escrito y verbal. Describen, en una o dos frases, una funcionalidad de software desde el punto de vista del usuario, con el lenguaje que éste emplearía. El foco está puesto en qué necesidades o

problemas soluciona lo que se va a construir. Su origen viene de la metodología XP donde las historias de usuario deben ser escritas por los clientes [23].

<b>Numero:</b> 11	<b>Nombre de la Historia de Usuario:</b> Insertar Edificio	
<b>Cantidad de modificaciones a la Historia de Usuario:</b> 0		
<b>Usuario:</b> Auxiliar Responsable del Área	<b>Iteración asignada:</b> 1	
<b>Prioridad en negocio:</b> Alta	<b>Tiempo estimado:</b> 24h	
<b>Riesgo en desarrollo:</b> Bajo	<b>Tiempo real:</b> 24h	
<p><b>Descripción:</b> Los usuarios que tengan rol Administrador o Auxiliares se les permite insertar cualquier edificio teniendo en cuenta los siguientes campos: cantidad de apartamentos (Obligatorio campo de texto, admite solo los caracteres 0-9 no guiones y no espacios); nombre (Obligatorio campo de texto, admite solo los caracteres 0-9 no guiones y no espacios).los mismos se guardan en la base de datos relacionada con el sistema.</p>		
<p><b>Observaciones:</b> 1. Si el usuario introduce la información correcta el sistema emite una notificación que ha insertado correctamente el AFT</p> <p>2. Si el usuario introduce un edificio ya existente se emite una notificación que ya existe el edificio</p>		
<b>Prototipo de interfaces:</b>		



Tabla 2. Historia de usuario requisito funcional Insertar Edificio

<b>Numero:</b> 17	<b>Nombre de la Historia de Usuario:</b> Insertar Apartamento	
<b>Cantidad de modificaciones a la Historia de Usuario:</b> 0		
<b>Usuario:</b> Auxiliar Responsable del Área, Administrador, Auxiliar Pedagógica, Director de Residencia.	<b>Iteración asignada:</b> 1	
<b>Prioridad en negocio:</b> Alta	<b>Tiempo estimado:</b> 16h	




<p><b>Riesgo en desarrollo:</b> Bajo</p>	<p><b>Tiempo real:</b> 16h</p>
<p><b>Descripción:</b> Los usuarios que tengan rol administrador o Auxiliares se les permite insertar cualquier apartamento teniendo en cuenta los siguientes campos: número de apartamento (Obligatorio campo de texto, admite solo los caracteres 0-9 no guiones y no espacios); cantidad de habitaciones (Obligatorio campo de texto, admite solo los caracteres 0-9 no guiones y no espacios); Estado( campo que define si es o no habitable dicho apartamento); el mismo se guarda en la base de datos relacionada con el sistema.</p>	
<p><b>Observaciones:</b></p> <ol style="list-style-type: none"> <li>1. No debe quedar ningún campo vacío</li> </ol>	
<p><b>Prototipo de interfaz:</b></p> 	

Tabla 3. Historia de usuario requisito funcional Insertar Apartamento

## **2.3 Diseño de la APK ResideM**

El diseño arquitectónico permite definir cómo debe organizarse un sistema y cómo tiene que diseñarse la estructura global de éste. Es el enlace entre el diseño y la ingeniería de requerimientos, que identifica los principales componentes estructurales en un sistema y la relación entre ellos. La salida del proceso de diseño arquitectónico consiste en un modelo arquitectónico que describe la forma en que se organiza el sistema como un conjunto de componentes en comunicación[26].

### **2.3.1 Diseño Arquitectónico**

#### **Modelo Vista Controlador**

El Modelo es el objeto que representa los datos del programa. Maneja los datos y controla todas sus transformaciones. El Modelo no tiene conocimiento específico de los Controladores o de las Vistas, ni siquiera contiene referencias a ellos. Es el propio sistema el que tiene encomendada la responsabilidad de mantener enlaces entre el Modelo y sus Vistas, y notificar a las Vistas cuando cambia el Modelo.

La Vista es el objeto que maneja la presentación visual de los datos representados por el Modelo. Genera una representación visual del Modelo y muestra los datos al usuario. Interactúa preferentemente con el Controlador, pero es posible que trate directamente con el Modelo a través de una referencia al propio Modelo.

El Controlador es el objeto que proporciona significado a las órdenes del usuario, actuando sobre los datos representados por el Modelo, centra toda la interacción entre la Vista y el Modelo. Cuando se realiza algún cambio, entra en acción, bien sea por cambios en la información del Modelo o por alteraciones de la Vista. Interactúa con el Modelo a través de una referencia al propio Modelo [27].

IVista: es la interfaz con la que el controlador se comunica con la vista.

Vista: vista que implementa la interfaz IVista y se encarga de manejar los aspectos visuales.

Mantiene una referencia al controlador, al cual le delega la responsabilidad del manejo de los eventos.

Controlador: contiene la lógica para responder a los eventos y manipula el estado de la vista mediante una referencia a la interfaz IVista. El Controlador utiliza el modelo para saber cómo

responder a los eventos y es responsable de establecer y administrar el estado de una vista. Modelo: está compuesto por los objetos que conocen y manejan los datos dentro de la aplicación.

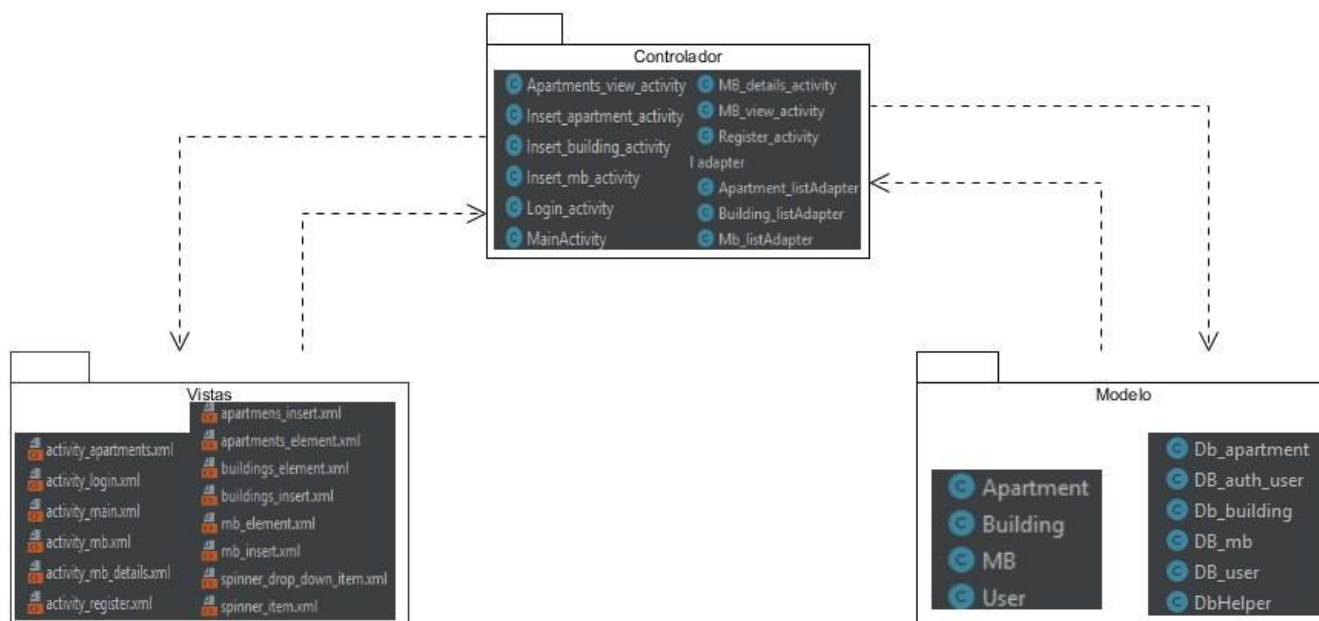


Figura 3. Diagrama de paquetes. Fuente: Elaboración propia

### 2.3.2 Mapa de Navegabilidad

Los sistemas de navegación que son estructuras básicas de un producto multimedia los cuales permiten que el contenido sea interpretado y distribuido adecuadamente, cumpliendo los conceptos de navegabilidad e interactividad usuario/producto. Expresa todas las relaciones de jerarquía y secuencia y permite elaborar escenarios de comportamiento de los usuarios. El principal valor de un mapa de navegación es que permite anticipar errores de organización de la información, de modo de corregirlos cuando aún no se ha invertido tiempo y dinero en la construcción del producto.

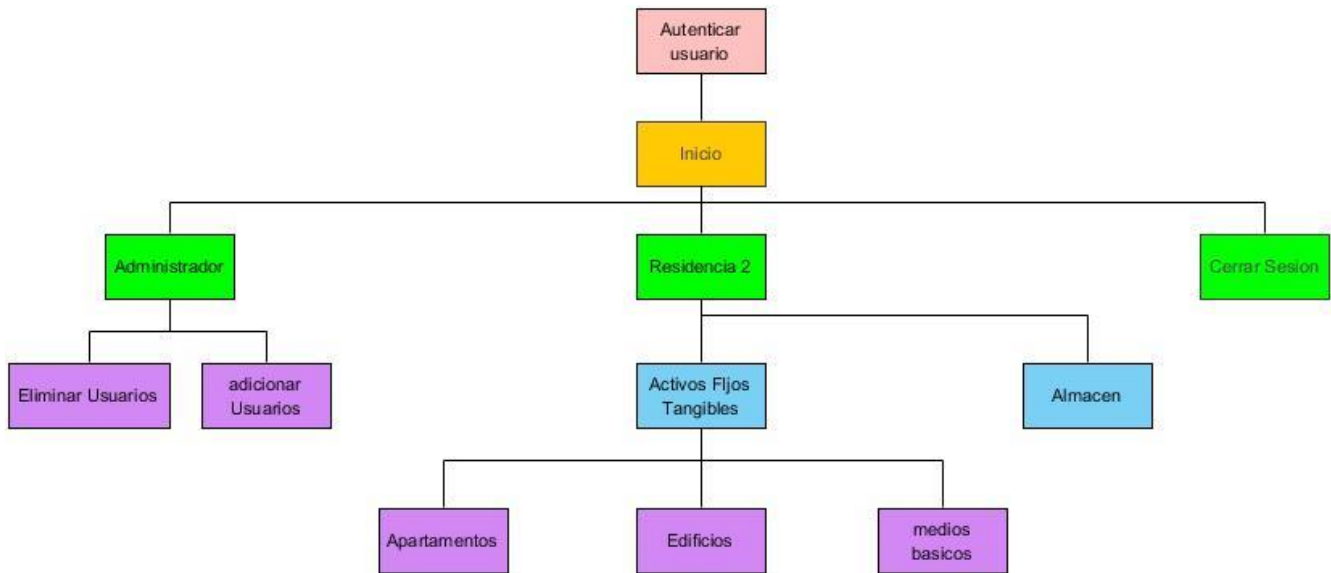


Figura 4. Mapa de navegabilidad. Fuente: elaboración propia.

### 2.3.3 Tarjeta Clase Responsabilidad Colaborador CRC

Emplear las tarjetas CRC (clase-responsabilidad-colaborador) para identificar y organizar las clases orientadas a objetos que son relevantes para el incremento actual de la aplicación móvil. El modelo CRC se compone de tarjetas índice CRC, cada tarjeta CRC menciona el nombre de la clase, sus responsabilidades (operaciones) y sus colaboradores (otras clases a las que envían mensajes y de las que depende para lograr sus responsabilidades). Las colaboraciones implican una serie de relaciones (es decir, conexiones) entre clases del sistema.

Clase: Insert_building_activity	
Responsabilidad: onCreate() fillFields() buttonaddlisterner() onClick()	Colaboración: bAddBuilding editTextBuildingName db_building

Tabla 4. Tarjeta CRC Insert\_building\_activity. Fuente elaboración propia

Clase: Insert_apartment_Activity	
<b>Responsabilidad:</b> onCreate() fillFields() init() addbuttonlistener() onClick() FillCombo()	<b>Colaboración:</b> Combo State Button_add_apartment db_apartment

*Tabla 5. Tarjeta CRC Insert\_apartment\_Activity. Fuente Elaboracion Propia*

### 2.3.4 Patrones de Diseño

#### Patrones GRASP

Lo esencial de un diseño de objetos lo constituye el diseño de las interacciones de objetos y la asignación de responsabilidades. Las decisiones que se tomen pueden influir profundamente en la extensibilidad, claridad y mantenimiento del sistema de software de objetos, además en el grado y calidad de los componentes reutilizables, por esta razón, durante el diseño se deben realizar los casos de usos con objetos basado en los patrones GRASP.

**Alta cohesión:** Es un principio evaluativo que aplica un diseñador mientras evalúa todas las decisiones de diseño. Indica la relación que existe entre los elementos de un mismo módulo. Es la medida de la relación funcional de los elementos de un módulo. El objetivo es organizar estos elementos de manera que los que tengan una mayor relación a la hora de realizar una tarea pertenezcan al mismo módulo, y los elementos no relacionados, se encuentren en módulos separados.

**Bajo acoplamiento:** Impulsa la asignación de responsabilidades de manera que su localización no incremente el acoplamiento hasta un nivel que nos lleve a los resultados negativos que

puede producir un acoplamiento alto. Es el grado de interdependencia entre los módulos. Un buen diseño se caracteriza por un acoplamiento mínimo, es decir, unos módulos tan independientes los unos de los otros como sea posible.

Controlador: Proporciona guías acerca de las opciones generalmente aceptadas y adecuadas para manejar eventos. Se evidencia en el uso de clases implicadas en la capa de control, poseen la capacidad de controlar el flujo de eventos mediante las actividades correspondientes, dentro de los ejemplos tenemos: MainActivity.java, Mb\_details\_activity.java, Login\_Activity.java.

Es conveniente utilizar la misma clase controlador para todos los eventos del sistema de un caso de uso, de manera que es posible manejar la información acerca del estado del caso de uso en el controlador.

Creador: Guía la asignación de responsabilidades relacionadas con la creación de objetos, una tarea muy común. La intención básica del patrón es encontrar un creador que necesite conectarse al objeto creado en alguna situación. Se evidencia su utilización en la clase Dbhelper

```
Public class DbHelper extends SQLiteOpenHelper {  
  
    private static final int DATABASE_VERSION = 1;  
    private static final String DATABASE_NAME = "ResideM.db";  
    private static final String TABLE_BUILDING = "t_building";  
    private static final String TABLE_APARTMENT = "t_apartment";  
}
```

Experto: Se utiliza con frecuencia en la asignación de responsabilidades; es un principio de guía básico que se utiliza continuamente en el diseño de objetos. Se evidencia en la clase Insert\_Building\_Activity, la cual realiza una inserción a la base de datos un determinado edificio. Expresa la intuición común de que los objetos hacen las cosas relacionadas con la información que tienen [28].

## Patrones GOF

Los patrones GOF los cuales se clasifican en tres grupos fundamentales: los patrones Creacionales que es donde se inicializan y configuran los objetos, los patrones Estructurales donde se separa la interfaz de la implementación y los patrones de Comportamiento los cuales describen el comportamiento entre las clases y los objetos.

**Adapter (Estructural):** Adapta una interfaz, facilitando la implementación para que pueda ser utilizada por una clase, que, de otro modo, no podría utilizarla. Esto se evidencia en la clase `login_activity.java`, interfaz creada para poder acceder al entorno de trabajo de la aplicación.

**Prototype:** Permite crear nuevos objetos por duplicación de objetos existentes llamados prototipos que disponen de la capacidad de clonación. Esto se evidencia en la clase `MB_details_activity.java`, donde se asignan dinámicamente los activos fijos tangibles a cada apartamento.

### 2.2.3 Conclusiones del capítulo

Se realizó un estudio de cómo llevar a cabo el proceso de control de los AFT, dicho proceso concluye en acciones que no están definidas en el marco actual de ejecución, por lo que fue necesario plantear un modelo de negocio. Mediante la descripción del modelado de Negocio se obtuvieron las principales características y conceptos que debe tener el sistema para controlar los activos fijos tangibles de la residencia #2. Las entrevistas realizadas con el cliente permitieron la captura de los requisitos funcionales y los no funcionales estos últimos apoyados en la norma de calidad ISO 25010, estos a su vez fueron agrupados en historias de usuario, obteniendo así de manera general las funcionalidades que debe cumplir la aplicación ResideM. Se utiliza como patrón arquitectónico modelo vista controlador posibilitando la construcción de un sistema legible, reutilizable y de fácil mantenimiento, brindando a su vez una organización básica para la implementación al equipo de desarrollo. El uso de las tarjetas de clase responsabilidad colaboración (CRC) contribuyen a la elaboración del software en un contexto orientado a objetos, permitiendo identificar y organizar las clases que satisfacen el cumplimiento de los requisitos identificados. Los patrones de diseño seleccionados proporcionan un correcto diseño de la aplicación propuesta. Con este capítulo se crearon las bases necesarias para la posterior implementación del sistema.

## CAPÍTULO III: VALIDACIÓN DE LA SOLUCIÓN PROPUESTA

### Introducción

En el presente capítulo se reflejan artefactos como el plan de iteraciones que recoge la implementación de las historias de usuario diagrama de componentes y las interfaces de comunicación. Se reflejan además las pruebas realizadas al software desarrollado, con el objetivo de comprobar las funcionalidades en los diferentes escenarios, para medir la calidad del producto y el grado en que cumple con los requisitos previamente identificados.

### 3.1 Plan de Iteraciones

Las Iteraciones son la fase principal en el ciclo de desarrollo de XP. Las funcionalidades son desarrolladas en esta fase, generando al final de cada una un entregable funcional que implementa las historias de usuario asignadas a la iteración [29]. En este se muestra la cantidad de iteraciones en que será desarrollada la aplicación ResideM, así como el tiempo asignado a cada iteración y las historias de usuario que serán implementadas en ellas.

Iteración	Orden de las HU a implementar	Duración total	Fecha inicio	Fecha fin
1	HU1. Autenticar Usuario HU7.Insertar residencia HU8.Modificar Residencia HU11.Insertar Edificios HU12.Modificar Edificios	5	15/8/22	19/9/22
2	HU17. Insertar apartamento HU18. Modificar apartamento	2.5	20/9/22	5/10/22
3	HU23. Insertar medio básico HU24. Modificar Medio básico	3	6/10/22	27/10/22

Tabla 6. Plan de Iteraciones. Fuente de Elaboracion Propia.



### Descripción del Plan de iteraciones

Iteración 1: Esta iteración se establece para implementar las Historias de usuarios 1, 7, 8, 11,12, estas servirán de entrada para implementar las iteraciones siguientes. Se cuenta con 5 semanas para el cumplimiento de estas tareas. Se obtiene un primer resultado donde se le permite al cliente autenticarse según su rol, crear un primer bloque residencial e insertar los primeros AFT que son los Edificios, este avance será mostrado posteriormente al cliente.

Iteración 2: Esta iteración tiene como objetivo la implementación de las Historias de Usuarios 17,18, las cuales incluyen todo lo referente a los apartamentos, según estos estén disponibles o no. Se cuenta con 2 semanas y media para la implementación de esta iteración. Al concluir, se obtiene la segunda entrega al cliente, una versión que contiene a la iteración anterior con la presente.

Iteración 3: Esta iteración tiene como objetivo la implementación de las Historias de Usuarios 23, 24, las cuales permiten trabajar todo lo referente a medios básicos según el tipo , la cantidad, lugar, estado. Se obtiene como resultado una tercera versión en conjunto con las anteriores, esto posibilita realizar pruebas, luego realizar cambios según sea necesario

### **3.2 Interfaz de comunicación**

El modelo TCP/IP (Protocolo para el Control de Transmisión/ Protocolo de Internet), está compuesto por cuatro capas, en la que cada una se encarga de determinados aspectos en la comunicación y a su vez cada una brinda un servicio específico a la capa superior. En la capa de aplicación del protocolo TCP/IP, se manejan protocolos de alto nivel que permiten la representación de los datos, codificación y control de dialogo. En la capa de transporte se establece una conexión lógica entre el host transmisor y el host receptor. Los protocolos de transporte segmentan los datos en el host origen para que las capas inferiores realicen el envío y una vez que estos llegan a su destino, son ensamblados para recuperar el mensaje original, brindando de esta manera un transporte de extremo a extremo. La capa de Internet tiene como finalidad seleccionar la mejor ruta para transmitir los paquetes por la red, de tal manera que cada paquete atraviese la menor cantidad de routers en el menor tiempo posible. El protocolo principal que opera en la capa es el protocolo de internet (IP). La capa de acceso a la red o “capa de host de red”, en ella se manejan todos los aspectos que un paquete IP requiere para

efectuar un enlace físico real con los medios de la red. En ella se incluyen los detalles de la tecnología LAN Y WAN y todos los detalles de la capa física y de enlace de datos del modelo OSI[30].

### 3.3 Diagrama de Componentes

Un componente puede ser tanto un código en lenguaje de programación como un código ejecutable ya compilado. En un sistema desarrollado en Java, cada archivo .java o .class es un componente del sistema, y se mostrará en el diagrama de componentes que los utiliza[31].

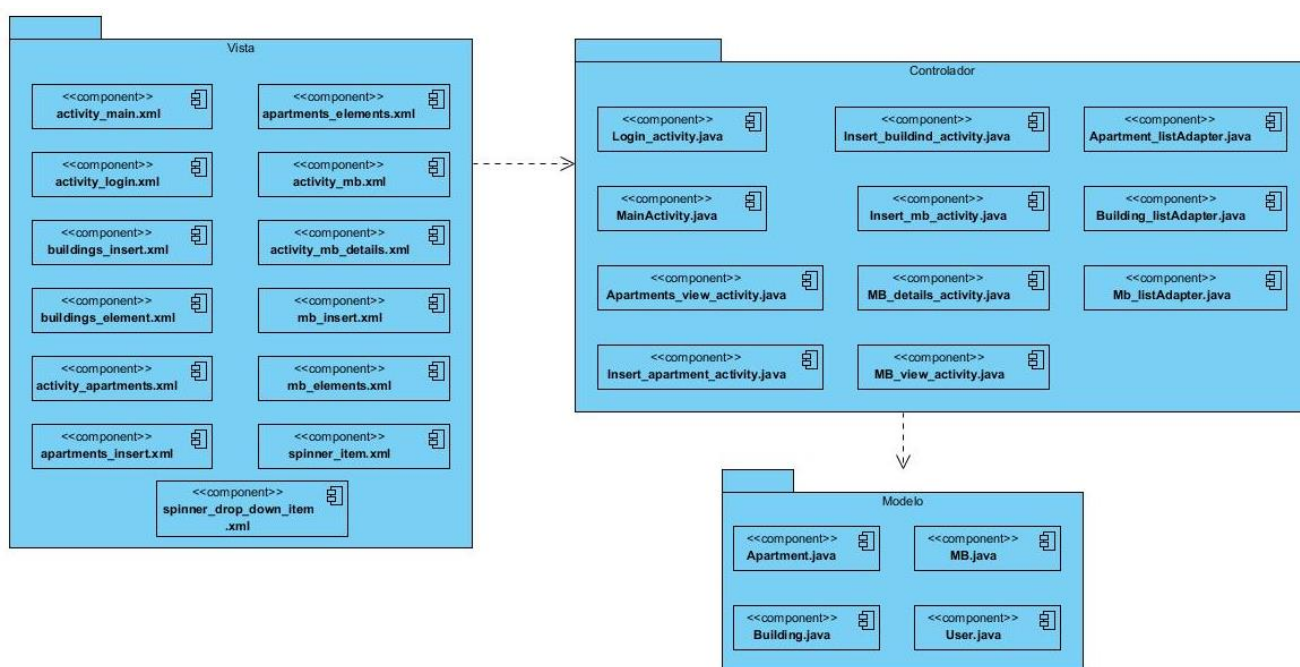


Figura 5. Diagrama de Componentes. Fuente: Elaboración propia

### 3.4 Pruebas de Software

Las pruebas de software permiten valorar la calidad de un programa, es decir, descubrir sus errores, sin embargo no deben verse como una red de seguridad. La calidad se incorpora a lo largo de todo el proceso de ingeniería de software y las pruebas deben confirmar que se ha logrado un buen programa, a través de la adecuada aplicación de métodos y herramientas.

Las pruebas de software tienen un doble objetivo: la verificación y la validación. La verificación persigue comprobar que se ha construido el producto correctamente. La validación comprueba que el producto se ha construido de acuerdo a los requerimientos del cliente [32].

### 3.4.1 Pruebas de Caja Negra

La estrategia de caja negra (black box) ignora por completo la estructura interna del programa y se basa exclusivamente en la comprobación de la especificación entrada-salida del software. Se trata de verificar que todos los requisitos impuestos al programa se cumplen. Esta es la única estrategia que puede adoptar el cliente o cualquier persona ajena al desarrollo del programa [32].

Escenario	Descripción	ID	Tipo	Cantidad	Respuesta del sistema	Flujo Central
EC23.1 Insertar Medio Básico	La aplicación inserta un medio básico de forma correcta	V	V	V	La aplicación añade a la base de datos el nuevo medio básico, y se muestra el medio básico añadido	El técnico accede a un edificio determinado luego a un apartamento dentro de este, selecciona insertar medios, se despliega una barra donde encuentra los medios disponibles, selecciona la cantidad que va añadir, y este queda insertado de forma exitosa.
EC23.2 insertar medio	La aplicación inserta un	V	V	I	La aplicación no añade a	El técnico accede a un edificio determinado luego a un apartamento dentro de

básico con campos vacíos	medio básico de forma incorrecta dejando campos vacíos				la base de datos el nuevo medio básico, y se muestra un mensaje "indicando que faltan campos por llenar "	este, selecciona insertar medios, se despliega una barra donde encuentra los medios disponibles, no selecciona la cantidad que precisa utilizar, el sistema arroja un mensaje de error, indicando que faltan campos por llenar
-----------------------------------	--------------------------------------------------------------------------	--	--	--	-----------------------------------------------------------------------------------------------------------------------------------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

*Tabla 7. Caso de Prueba del Requisito Insertar Medio Básico.*

### 3.4.2 Pruebas Unitarias Caja Blanca

Este tipo de técnicas son las realizadas ejecutando la aplicación y son las utilizadas para el diseño de los casos de prueba. La mayoría del software puede probarse de dos maneras diferentes. Conociendo el funcionamiento interno, podemos probar que todos los módulos encajan unos con otros, es decir, desde una visión interna. Estas pruebas son las pruebas de caja blanca. Al conocer las funciones específicas del producto se pueden llevar a cabo pruebas que demuestren que estas funciones son operativas y la búsqueda de errores en dichas funciones [33].

#### **JUnit**

Para la implementación de esta prueba se utilizará la herramienta JUnit 3 integrada con el IDE Android Studio que se utilizó para desarrollar la aplicación, en la cual se selecciona la clase a probar, esta herramienta automáticamente crea el paquete de prueba y dentro de este genera la clase de prueba (Test). La clase Test tiene métodos homólogos a la clase probada que contienen los valores establecidos y una función "fail ()" que lanza un fallo en el método. Si se ejecuta la clase Test esta muestra los errores en el código y el porcentaje de satisfacción de los casos de prueba. A continuación, se ejemplifican las pruebas realizadas a la clase DB\_Building mediante la clase DB\_BuildingTest, (Ver Anexo 3).

```
19:24 Gradle sync finished in 21 m 10 s 107 ms
19:29 Executing tasks: [:app:assembleDebug, :app:assembleDebugAndroidTest] in
19:34 Gradle build finished in 5 m 3 s 438 ms
19:34 Success: Operation succeeded
19:34 Tests passed: 1
19:34 Tests passed: 1
19:35 Executing tasks: [:app:assembleDebug, :app:assembleDebugAndroidTest] in
19:35 Gradle build finished in 4 s 405 ms
19:35 Success: Operation succeeded
19:35 Tests passed: 1
19:35 Tests passed: 1
```

Figura 6. Resultado de la aplicación de la prueba unitaria

```
45 ▶▶ public class Db_buildingTest extends TestCase {
46     private Db_building db_building;
47     Context context = new Context() {...};
572
573     @Before
574     public void SetUp() {
575         db_building = new Db_building(context);
576     }
577
578     @Test
579 ▶▶ public void testDeleteBuilding() {
580         boolean valor_esperado = true ;
581         boolean valor_real = db_building.deleteBuilding( id: 2);
582         assertEquals(valor_esperado, valor_real);
583     }
584
585
586     @Test
587 ▶▶ public void testsearchBuilding(){
588         Building buildingesperado = new Building( id: 1, name: "90", apartments: 0);
589         assertEquals(buildingesperado, db_building.searchBuilding( id: 1));
590     }
591
592     @Test
593 ▶▶ public void InsertBuilding () {
594         boolean valor_esperado = true;
595         assertEquals(valor_esperado, actual: true);
```

Figura 7. Resultado de la aplicación de la prueba unitaria

## Pruebas de aceptación

Las pruebas de aceptación, conocidas también como técnicas de evaluación dinámica, son un elemento crítico para la garantía de la calidad del sistema. Representan una revisión final de las especificaciones del diseño y de la implementación. Su principal objetivo es diseñar pruebas que, sistemáticamente, saquen a la luz diferentes clases de errores, haciéndolo con la menor cantidad de tiempo y esfuerzo [34].

Pruebas Alfa: La prueba alfa se lleva a cabo en el sitio del desarrollador por un grupo representativo de usuarios finales. El software se usa en un escenario natural con el

desarrollador “mirando sobre el hombro” de los usuarios y registrando los errores y problemas de uso. Las pruebas alfa se realizan en un ambiente controlado[34].

Pruebas Beta: La prueba beta se realiza en uno o más sitios del usuario final. A diferencia de la prueba alfa, por lo general el desarrollador no está presente. Por tanto, la prueba beta es una aplicación “en vivo” del software en un ambiente que no puede controlar el desarrollador. El cliente registra todos los problemas (reales o imaginarios) que se encuentran durante la prueba beta y los reporta al desarrollador periódicamente. Como resultado de los problemas reportados durante las pruebas beta, es posible hacer modificaciones y luego preparar la liberación del producto de software a toda la base de clientes[34].

Debido al uso de las Historias de Usuario se decide aplicar la prueba de aceptación a la aplicación móvil ResideM. Se desarrollaron pruebas alfas por un grupo de desarrolladores docentes de la facultad de Ciencias y Tecnologías Computacionales, en la asignatura de Ingeniería de Software II. Durante las pruebas beta la aplicación móvil fue sometida a un proceso de adaptación por parte de los técnicos de residencia y auxiliares pertenecientes a la residencia #2. Obteniendo que:

Se cumplen los Requisitos Funcionales y no Funcionales.

La aplicación móvil resultante es de gran valor y aprecio por los usuarios finales producto a su eficacia, agilidad, movilidad por las áreas.

La aplicación aparte de informatizar el proceso de gestión de activos fijos tangibles en la residencia número #2 de la Universidad de Ciencias Informáticas, hace más humano el proceso.

### **3.4.3 Resultados de las pruebas aplicadas a la aplicación móvil ResideM**

Los problemas detectados en el período de pruebas de validación y aceptación se clasificaron en: No conformidades significativas (NCS) y en No conformidades no significativas (NCNS). A continuación, se describen los aspectos que se tuvieron en cuenta en cada clasificación.

NCS: son las no conformidades referentes a las funcionalidades de la aplicación: validaciones incorrectas o respuestas de la aplicación diferentes a lo descrito previamente en las historias de usuario.

NCNS: son las no conformidades en cuanto al diseño de la propuesta de solución y errores ortográficos.

Fueron realizadas 3 iteraciones de pruebas, en las que se encontraron 11 no conformidades significativas y 4 no conformidades en la primera iteración, luego de haber corregido las no conformidades anteriores se obtiene en la segunda iteración 5 no conformidades significativas y 2 no conformidades no significativas, quedando resueltas para la tercera iteración donde no se encontraron no conformidades.

No conformidades encontradas:

**Significativas:**

Error de validación en el número de cada edificio.

Error al insertar un apartamento en la base de datos.

Error al introducir la cantidad de habitaciones que puede tener un apartamento.

Error al introducir la cantidad de medios básicos de un mismo tipo dentro de un apartamento

Roles no definidos.

No existencia de una base de datos remota, pero si local.

Cualquier usuario tendría una versión propia de su base de datos y de sus datos introducidos.

Desde la vista de gestión de medios básicos el software vuelve al menú principal y no al menú anterior.

Error en la sintaxis de las consultas a la base de datos.

Algunos mensajes de error muestran los nombres de atributos.

Introducción de caracteres no validos en campos de texto.

**No Significativas:**

Errores de ubicación de textos en los cuadros de información, en las vistas de registrar usuario e insertar edificio.

Error de diseño al establecer si un apartamento está cerrado o abierto, el diseño corresponde al cerrado cuando está abierto y viceversa.

Los iconos que se despliegan junto a las opciones para gestionar medios básicos se muestran fuera de lugar.

El menú de gestionar se muestra por detrás de los demás cuadros informativos.



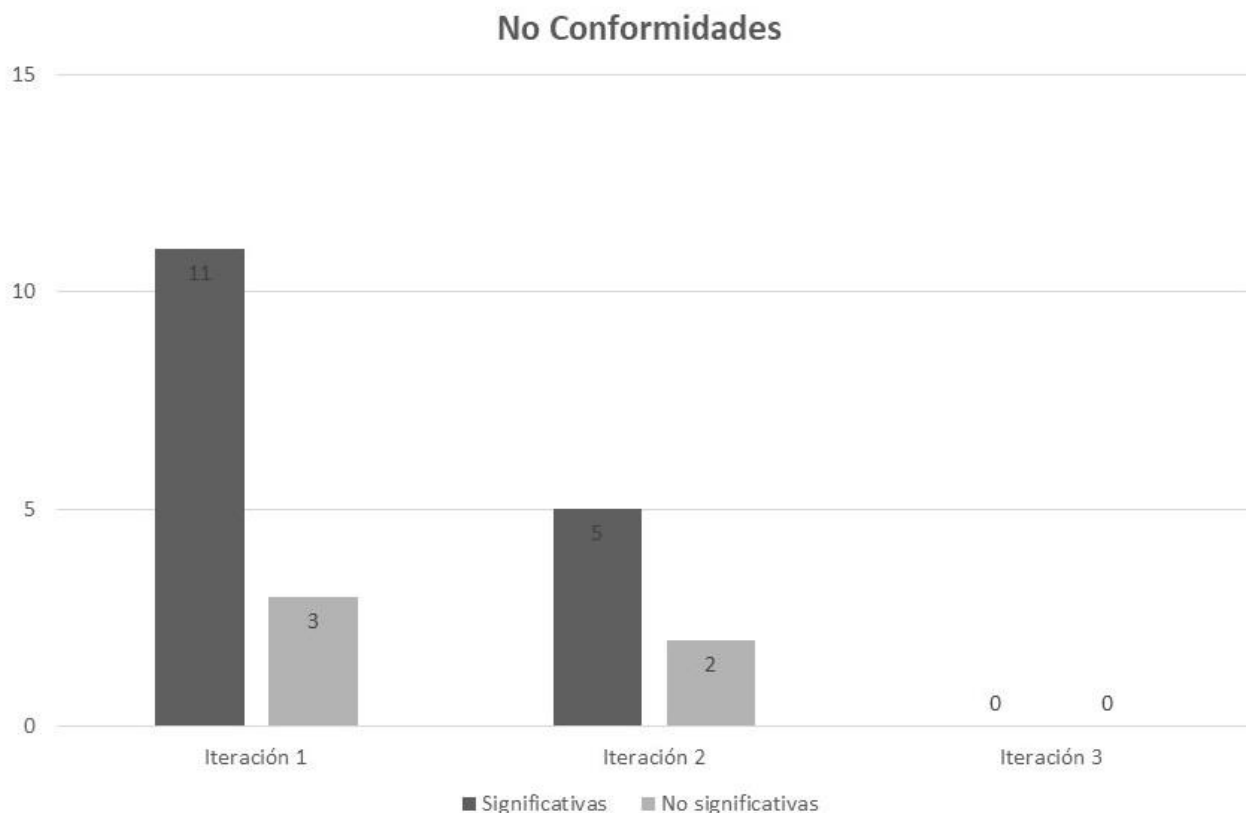


Figura 8. No conformidades. Fuente: Elaboración propia

### Conclusiones del capítulo

La elaboración del plan de iteraciones permitió la planificación de cada una de las iteraciones, dejando organizado el tiempo de trabajo, distribuyendo cargas según la complejidad para completar el desarrollo de la aplicación. Cada iteración arroja un conjunto de componentes, los cuales son asociados más tarde al diagrama de componentes, diagrama que permite una mayor comprensión de los elementos estructurales y de implementación de la aplicación. Las pruebas permitieron comprobar el correcto funcionamiento del código de la aplicación, validar la completitud de los requisitos y determinar la aceptación por parte del cliente, respectivamente.

## **CONCLUSIONES FINALES**

Una vez culminada la investigación se desarrolló una aplicación que permitiera el control y la gestión de todos los activos fijos tangibles de la residencia estudiantil, concluyendo lo siguiente:

Durante el desarrollo del presente trabajo se realizó el diseño del marco teórico de la investigación donde se planteó la situación problemática, el problema a resolver, el objeto de estudio, el campo de acción, la idea a defender, el objetivo general y los objetivos específicos, quedando con esto definidas las acciones a ejecutar en cada fase del trabajo para llevar a buen término la investigación. El estudio del proceso de control de activos fijos tangibles, incorporando la opción de desarrollar una aplicación móvil permitió comprender los elementos teóricos y conceptos para el desarrollo de una aplicación con tal fin. Los sistemas existentes tanto nacional como internacional que se investigaron no contenían una solución factible, pero contribuyeron de forma significativa para elaborar una idea general de cómo llevar a cabo la representación de dicho sistema. El modelo del dominio establece el contexto donde será usada la aplicación, además permitió en conjunto con el cliente la elaboración de los requisitos funcionales y los no funcionales, los cuales serían agrupados posteriormente en historias de usuario ayudando a concretar la elaboración del sistema en cuestión. En la ejecución de pruebas a la aplicación se detectó un conjunto no conformidades que fueron resueltas, logrando con esto una aplicación libre de errores y lista para satisfacer las necesidades de los clientes.

## **RECOMENDACIONES**

Se recomienda la puesta en práctica, realizar un pequeño despliegue de esta versión de software en la residencia #2 y la posibilidad de ampliar su uso a las residencias vecinas.

## REFERENCIAS BIBLIOGRÁFICAS

1. <https://anepsa.com.mx/software-control-activo-fijo/>
1. Fuentes Díaz, D. and D.J.O.d.l.E.L. Hernández González, *Rediseño del manual de procedimientos de los activos fijos tangibles en la empresa eléctrica Cienfuegos*. 2019(octubre).
2. Gutiérrez Dávila, C., A. Hernández Campos, and J. Rositas Martínez, *Influencia de las TIC en la creación de valor de las empresas que cotizan en la Bolsa Mexicana de Valores: sector financiero e industrial*. 2020.
3. Chagoya, E.R., *Métodos y técnicas de investigación*. Obtenido de Gestipolis: <https://www.gestipolis.com/metodos-y-tecnicas-de-investigacion>, 2008.
4. Segura, F.O., *Sistema de gestión: Una guía práctica*. 2005: Ediciones Díaz de Santos.
5. de Pablos Heredero, C., *Informática y comunicaciones en la empresa*. 2004: ESIC Editorial.
6. MEANA COALLA, P.P., *Gestión de inventarios*. 2017: Ediciones Paraninfo, SA.
7. Pedraza Alvarez, I., *Sistema de medios básicos de la biofábrica de las plantas de Villa Clara*. 2005, Universidad Central “Marta Abreu” de Las Villas.
8. Robledo, D., *Desarrollo de aplicaciones para Android I*. 2016: Ministerio de Educación, Cultura y Deporte.
9. DiMarzio, J., *Beginning Android Programming with Android Studio*. 2016: John Wiley & Sons.
10. Hernández González, B., T. Ramírez Ramírez, and O.J.R.U.y.S. Mar Cornelio, *Sistema para la auditoría y control de los activos fijos tangibles*. 2019. **11**(1): p. 128-134.
11. Herrera, D.I.A., S.G.M. Triana, and I.M.J.T.r.a.d.i. Nieblas, *Actividades para favorecer el aprendizaje del subsistema activo fijo tangible en el Versat-Sarasola*. 2020. **11**(35): p. 1-11.
12. Padrón Mojena, Y. and L.D. González Díaz, *Sistema informático para la gestión de la información de los Activos Fijos Tangibles, los Útiles y Herramientas de la residencia de la Universidad de las Ciencias Informáticas*. 2016, Universidad de las Ciencias Informáticas. Facultad 4.
13. Peña, D.M. and L.J.U.I.C.I. Baquero, *Extensión de la herramienta Visual Paradigm for UML para la evaluación y corrección de Diagramas de Casos de Uso*. 2016.
14. Lozano Banqueri, J.M., *Creación y gestión de una base de datos con MySQL y phpMyAdmin*. 2018.
15. Salas-Rueda, R.-A.J.E.-L. and D. Media, *Impact of the WampServer application in Blended learning considering data science, machine learning, and neural networks*. 2020. **17**(3): p. 199-217.
16. Arias, M.Á., *Aprende Programacion Web con PHP y MySQL: 2ª Edicion*. 2017: IT campus Academy.
17. Bernal, A.I.E. and I.A. Palma, *Taller De Bases De Datos*.
18. Castillo, J.D.L., *Desarrollo de aplicaciones Android con Android Studio: Conoce android studio*. 2019: José Dimas Luján Castillo.
19. Beltrán, L.G.M.J.X.B.C.d.l.E.S.d.T., *JAVA como lenguaje universal de programación*. 2016. **4**(8).
20. Rivas, C.I., et al., *Metodologías actuales de desarrollo de software*. 2015. **2**(5): p. 980-986.

21. Cadavid, A.N., J.D.F. Martínez, and J.M.J.P. Vélez, *Revisión de metodologías ágiles para el desarrollo de software*. 2013. **11**(2): p. 30-39.
22. Robles, G. and J.J.U.P.d.M.E. Ferrer, *Programación eXtrema y Software Libre*. 2002.
23. Letelier, P. and M.C. Penadés, *Métodologías ágiles para el desarrollo de software: eXtreme Programming (XP)*. 2006.
24. Vázquez-Ingelmo, A. and F.J. García-Peñalvo, *Modelo de Dominio*. 2019.
25. Escalona, M.J. and N. Koch, *Ingeniería de Requisitos en Aplicaciones para la Web—Un estudio comparativo*. Universidad de Sevilla, 2002.
26. Bosch, J., *Design and use of software architectures: adopting and evolving a product-line approach*. 2000: Pearson Education.
27. Fernández Romero, Y. and Y.J.T.t. Díaz González, *Patrón Modelo-Vista-Controlador*. 2012. **11**(1): p. 47-57.
28. Villa, M.C., *Título: " Modelo para la ayuda a la toma de decisiones en la selección de patrones de desarrollo de software*. 2007, Universidad de las Ciencias Informáticas.
29. Joskowicz, J.J.U.d.V., *Reglas y prácticas en eXtreme Programming*. 2008. **22**.
30. Hernández, E.A., et al., *Comparación de los modelos OSI y TCP/IP*. 2017. **5**(10).
31. Burgués, J.E.G., *Aprende a Modelar Aplicaciones con UML-Tercera Edición*. 2018: IT campus academy.
32. Palomo, S.R.G. and E.M. Gil, *Aproximación a la ingeniería del software*. 2020: Editorial Centro de Estudios Ramon Areces SA.
33. Sanchez Peño, J.M., *Pruebas de software. fundamentos y técnicas*. 2015.
34. Aliaga, A.M.C., Á.A.A. Ruiz, and J.C.E. Pérez, *Sistema para la gestion de activos multimedia para la Dirección de Extensión Universitaria System for multimedia asset management for the management of University Extension*.
35. Rimawi, D. and S. Zein. *A model based approach for Android design patterns detection*. in *2019 3rd International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT)*. 2019. IEEE.
36. Peters, E., G.K.J.A.i.S. Aggrey, Technology, and E.S. Journal, *An ISO 25010 based quality model for ERP systems*. 2020. **5**: p. 578-583.
37. Dominguez Vargas, A.A. and E.M. Hinojosa León, *Desarrollo e Implementacion de Aplicación Web y Aplicación Movil para el Registro y el Control de los Activos Fijos de la Universidad de Guayaquil*. 2018, Universidad de Guayaquil. Facultad de Ciencias Matemáticas y Físicas ....
38. Yupanqui Palomino, L.C.A., *Propuesta para implementar un sistema de gestión de activos fijos tangibles para la IE N° 20123—cañete; 2020*. 2020.
39. Aguirre Ayo, F.P., *SISTEMA DE LEVANTAMIENTO DE ACTIVOS FIJOS EN ANDROID PARA LA EMPRESA CAYMAN SYSTEMS*. 2018, Quito.
40. Valerio, C.L. and J.C.M.J.R.I.d.S.e.T.d.I. Cruz, *Guía para la Ingeniería de Requerimientos bajo un enfoque ágil integrando técnicas de usabilidad*. 2020(E32): p. 546-558.
41. Sangama Oñate, A.F., *Metodologías ágiles Scrum, XP, SLeSS, Scrumban, HME, Mobile-D y MASAN empleadas en la industria de dispositivos móviles: Un contraste en favor de la industria del desarrollo móvil*. 2020.
42. García Cuenca, M.V., *Construcción de mapas conceptuales navegables y comprensión lectora: análisis de procesos y diseño de instrucción*. 2019.
43. Montero, B.M., H.V. Cevallos, and J.D.J.E.r.m.d.i. Cuesta, *Metodologías ágiles frente a las tradicionales en el proceso de desarrollo de software*. 2018. **2**(17): p. 114-121.

44. Sohaib, O., et al., *Integrating design thinking into extreme programming*. 2019. **10**(6): p. 2485-2492.
45. Bai, Y., *JDBC API and JDBC Drivers*, in *Oracle Database Programming with Java*. Auerbach Publications. p. 67-86.
46. Ganesh, S., H. Kiran, and T. Sharma, *Building Database Applications with JDBC*, in *Oracle Certified Professional Java SE 8 Programmer Exam 1Z0-809*. 2016, Springer. p. 359-387.

#### Bibliografías consultadas

[35], [36],[37],[38],[39],[40],[41],[42],[43],[44],[45],[46]

## ANEXOS

Anexo 1 Entrevista realizada para identificar las deficiencias del proceso de gestión de activos fijos tangibles para la residencia #2 de la universidad de ciencias informáticas

I.	Entrevista realizada a las Auxiliares para identificar las deficiencias en el proceso de gestión de activos fijos tangibles en la residencia #2 de la universidad de las ciencias informáticas. Es necesario que responda las siguientes preguntas basándose en su experiencia práctica como auxiliar del área
Pregunta 1	¿Cómo se registran actualmente los activos fijos tangibles de la residencia #2?
Pregunta 2	¿Cómo se determina que edificios o apartamentos son no habitables?
Pregunta 3	¿Cómo se lleva a cabo el proceso de asignar medios básicos a un apartamento?
Pregunta 4	¿Cuáles son los medios básicos que debe tener un apartamento?
Pregunta 5	¿Cada que tiempo se realiza el control de activos fijos tangibles?
Pregunta 6	¿Qué sucede en caso de pérdidas o Robos de activos fijos tangibles?
Pregunta 7	¿Cómo se administran los recursos existentes en el almacén?
Pregunta 8	¿Qué datos le resultaría imprescindible para el control de activos fijos tangibles?
Pregunta 9	¿Que facilidades les brinda el uso de una aplicación móvil para su desempeño laboral?

Anexo 2: Tarjetas CRC

Clase: Apartment_view_Activity	
Responsabilidad: onCreate() fillFields() SetButtonslistener() ButtonEditListener() onClick() ButtonDeleteListener() onClick() ButtonAddNewListener() onClick() ButtonVisibilityToggleListener() SetButtonVisibilityOff() SetButtonVisibilityOn() Init() fillExtras()	Colaboración: List<Apartment> Textview FloatingActionButton Bundle

Clase: Apartment_view_Activity	
Responsabilidad: onCreate() fillFields() AddButtonListener() onClick() Init() fillCombo()	Colaboración: Bundle Spinner Button Number Picker ArrayAdapter<CharSequence> DB_mb

Clase: Login_Activity	
Responsabilidad: onCreate()	Colaboración: BundlesavedInstancedState

Clase: Main_Activity	
Responsabilidad: onCreate() addListenerToButtons() onClick() initDB()	Colaboración: FloatingActionButton TextView List<Building> NumberPicker Boolean isAllButtonVisible

Clase: MB_details_activity	
Responsabilidad: onCreate() setButtonListener() ButtonEditListener() ButtonDeleteListener() onClick() init()	Colaboración: Bundle FloatingActionButton TextView

Clase: MB_view_activity	
Responsabilidad: onCreate() setButtonListener() ButtonVisibilityToggleListener() SetButtonVisibilityOn() SetButtonVisibilityOff() ButtonAddNewListener() onClick()	Colaboración: Bundle FloatingActionButton TextView ArrayList<MB> Boolean isAllButtonVisible



ButtonDeleteListener() onClick() ButtonEditListener() onClick() init() fillExtras()	
----------------------------------------------------------------------------------------------------	--

Clase: apartment_listAdapter	
Responsabilidad: onCreateViewHolder() onBindviewHolder() getItemCount() setItems() ViewHolder() BindData() OnClick()	Colaboración: LayoutInflater List<Apartment> Context

Clase: Building_listAdapter	
Responsabilidad: onCreateViewHolder() onBindviewHolder() getItemCount() setItems() ViewHolder() BindData() OnClick()	Colaboración: LayoutInflater List<Building> Context

Clase: MB_listAdapter	
Responsabilidad: onCreateViewHolder() onBindviewHolder() getItemCount() setItems() ViewHolder() BindData() OnClick()	Colaboración: LayoutInflater List<MB> Context

## Anexo 3

## Implementación de la prueba Unitaria :

```
package com.example.controluci.db;

import android.content.BroadcastReceiver;
import android.content.ComponentName;
import android.content.ContentResolver;
import android.content.Context;
import android.content.Intent;
import android.content.IntentFilter;
import android.content.IntentSender;
import android.content.ServiceConnection;
import android.content.SharedPreferences;
import android.content.pm.ApplicationInfo;
import android.content.pm.PackageManager;
import android.content.res.AssetManager;
import android.content.res.Configuration;
import android.content.res.Resources;
import android.database.DatabaseErrorHandler;
import android.database.sqlite.SQLiteDatabase;
import android.graphics.Bitmap;
import android.graphics.drawable.Drawable;
import android.net.Uri;
import android.os.Bundle;
import android.os.Handler;
import android.os.Looper;
import android.os.UserHandle;
import android.view.Display;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;

import com.example.controluci.model.Building;

import junit.framework.TestCase;

import org.junit.Before;
import org.junit.Test;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;

public class Db_buildingTest extends TestCase {
    private Db_building db_building;
    Context context = new Context() {
```

```
@Override
public AssetManager getAssets() {
    return null;
}

@Override
public Resources getResources() {
    return null;
}

@Override
public PackageManager getPackageManager() {
    return null;
}

@Override
public ContentResolver getContentResolver() {
    return null;
}

@Override
public Looper getMainLooper() {
    return null;
}

@Override
public Context getApplicationContext() {
    return null;
}

@Override
public void setTheme(int i) {
}

@Override
public Resources.Theme getTheme() {
    return null;
}

@Override
public ClassLoader getClassLoader() {
    return null;
}

@Override
public String getPackageName() {
    return null;
}

@Override
public ApplicationInfo getApplicationInfo() {
    return null;
}
}
```

```

@Override
public String getPackageResourcePath() {
    return null;
}

@Override
public String getPackageCodePath() {
    return null;
}

@Override
public SharedPreferences getSharedPreferences(String s, int i) {
    return null;
}

@Override
public boolean moveSharedPreferencesFrom(Context context, String s) {
    return false;
}

@Override
public boolean deleteSharedPreferences(String s) {
    return false;
}

@Override
public FileInputStream openFileInput(String s) throws FileNotFoundException {
    return null;
}

@Override
public FileOutputStream openFileOutput(String s, int i) throws FileNotFoundExcep-
tion {
    return null;
}

@Override
public boolean deleteFile(String s) {
    return false;
}

@Override
public File getFileStreamPath(String s) {
    return null;
}

@Override
public File getDataDir() {
    return null;
}

@Override
public File getFilesDir() {

```

```
        return null;
    }

    @Override
    public File getNoBackupFilesDir() {
        return null;
    }

    @Nullable
    @Override
    public File getExternalFilesDir(@Nullable String s) {
        return null;
    }

    @Override
    public File[] getExternalFilesDirs(String s) {
        return new File[0];
    }

    @Override
    public File getObbDir() {
        return null;
    }

    @Override
    public File[] getObbDirs() {
        return new File[0];
    }

    @Override
    public File getCacheDir() {
        return null;
    }

    @Override
    public File getCodeCacheDir() {
        return null;
    }

    @Nullable
    @Override
    public File getExternalCacheDir() {
        return null;
    }

    @Override
    public File[] getExternalCacheDirs() {
        return new File[0];
    }

    @Override
    public File[] getExternalMediaDirs() {
        return new File[0];
    }
}
```

```

@Override
public String[] fileList() {
    return new String[0];
}

@Override
public File getDir(String s, int i) {
    return null;
}

@Override
public SQLiteDatabase openOrCreateDatabase(String s, int i, SQLiteDatabase.Cursor-
Factory cursorFactory) {
    return null;
}

@Override
public SQLiteDatabase openOrCreateDatabase(String s, int i, SQLiteDatabase.Cursor-
Factory cursorFactory, @Nullable DatabaseErrorHandler databaseErrorHandler) {
    return null;
}

@Override
public boolean moveDatabaseFrom(Context context, String s) {
    return false;
}

@Override
public boolean deleteDatabase(String s) {
    return false;
}

@Override
public File getDatabasePath(String s) {
    return null;
}

@Override
public String[] databaseList() {
    return new String[0];
}

@Override
public Drawable getWallpaper() {
    return null;
}

@Override
public Drawable peekWallpaper() {
    return null;
}

@Override

```

```
public int getWallpaperDesiredMinimumWidth() {
    return 0;
}

@Override
public int getWallpaperDesiredMinimumHeight() {
    return 0;
}

@Override
public void setWallpaper(Bitmap bitmap) throws IOException {

}

@Override
public void setWallpaper(InputStream inputStream) throws IOException {

}

@Override
public void clearWallpaper() throws IOException {

}

@Override
public void startActivity(Intent intent) {

}

@Override
public void startActivity(Intent intent, @Nullable Bundle bundle) {

}

@Override
public void startActivities(Intent[] intents) {

}

@Override
public void startActivities(Intent[] intents, Bundle bundle) {

}

@Override
public void startActivitySender(IntentSender intentSender, @Nullable Intent intent,
int i, int i1, int i2) throws IntentSender.SendIntentException {

}

@Override
public void startActivitySender(IntentSender intentSender, @Nullable Intent intent,
int i, int i1, int i2, @Nullable Bundle bundle) throws IntentSender.SendIntentException {
```

```

    }

    @Override
    public void sendBroadcast(Intent intent) {

    }

    @Override
    public void sendBroadcast(Intent intent, @Nullable String s) {

    }

    @Override
    public void sendOrderedBroadcast(Intent intent, @Nullable String s) {

    }

    @Override
    public void sendOrderedBroadcast(@NonNull Intent intent, @Nullable String s, @Nul-
lable BroadcastReceiver broadcastReceiver, @Nullable Handler handler, int i, @Nullable
String s1, @Nullable Bundle bundle) {

    }

    @Override
    public void sendBroadcastAsUser(Intent intent, UserHandle userHandle) {

    }

    @Override
    public void sendBroadcastAsUser(Intent intent, UserHandle userHandle, @Nullable
String s) {

    }

    @Override
    public void sendOrderedBroadcastAsUser(Intent intent, UserHandle userHandle, @Nul-
lable String s, BroadcastReceiver broadcastReceiver, @Nullable Handler handler, int i,
@Nullable String s1, @Nullable Bundle bundle) {

    }

    @Override
    public void sendStickyBroadcast(Intent intent) {

    }

    @Override
    public void sendStickyOrderedBroadcast(Intent intent, BroadcastReceiver broad-
castReceiver, @Nullable Handler handler, int i, @Nullable String s, @Nullable Bundle bun-
dle) {

    }

```



```

@Override
public void removeStickyBroadcast(Intent intent) {

}

@Override
public void sendStickyBroadcastAsUser(Intent intent, UserHandle userHandle) {

}

@Override
public void sendStickyOrderedBroadcastAsUser(Intent intent, UserHandle userHandle,
BroadcastReceiver broadcastReceiver, @Nullable Handler handler, int i, @Nullable String s,
@Nullable Bundle bundle) {

}

@Override
public void removeStickyBroadcastAsUser(Intent intent, UserHandle userHandle) {

}

@Nullable
@Override
public Intent registerReceiver(@Nullable BroadcastReceiver broadcastReceiver, In-
tentFilter intentFilter) {
    return null;
}

@Nullable
@Override
public Intent registerReceiver(@Nullable BroadcastReceiver broadcastReceiver, In-
tentFilter intentFilter, int i) {
    return null;
}

@Nullable
@Override
public Intent registerReceiver(BroadcastReceiver broadcastReceiver, IntentFilter
intentFilter, @Nullable String s, @Nullable Handler handler) {
    return null;
}

@Nullable
@Override
public Intent registerReceiver(BroadcastReceiver broadcastReceiver, IntentFilter
intentFilter, @Nullable String s, @Nullable Handler handler, int i) {
    return null;
}

@Override
public void unregisterReceiver(BroadcastReceiver broadcastReceiver) {

}

```

```

@Nullable
@Override
public ComponentName startService(Intent intent) {
    return null;
}

@Nullable
@Override
public ComponentName startForegroundService(Intent intent) {
    return null;
}

@Override
public boolean stopService(Intent intent) {
    return false;
}

@Override
public boolean bindService(Intent intent, @NonNull ServiceConnection serviceCon-
nection, int i) {
    return false;
}

@Override
public void unbindService(@NonNull ServiceConnection serviceConnection) {
}

@Override
public boolean startInstrumentation(@NonNull ComponentName componentName, @Nulla-
ble String s, @Nullable Bundle bundle) {
    return false;
}

@Override
public Object getSystemService(@NonNull String s) {
    return null;
}

@Nullable
@Override
public String getSystemServiceName(@NonNull Class<?> aClass) {
    return null;
}

@Override
public int checkPermission(@NonNull String s, int i, int i1) {
    return 0;
}

@Override
public int checkCallingPermission(@NonNull String s) {
    return 0;
}

```

```

    }

    @Override
    public int checkCallingOrSelfPermission(@NonNull String s) {
        return 0;
    }

    @Override
    public int checkSelfPermission(@NonNull String s) {
        return 0;
    }

    @Override
    public void enforcePermission(@NonNull String s, int i, int i1, @Nullable String
s1) {

    }

    @Override
    public void enforceCallingPermission(@NonNull String s, @Nullable String s1) {

    }

    @Override
    public void enforceCallingOrSelfPermission(@NonNull String s, @Nullable String s1)
{

    }

    @Override
    public void grantUriPermission(String s, Uri uri, int i) {

    }

    @Override
    public void revokeUriPermission(Uri uri, int i) {

    }

    @Override
    public void revokeUriPermission(String s, Uri uri, int i) {

    }

    @Override
    public int checkUriPermission(Uri uri, int i, int i1, int i2) {
        return 0;
    }

    @Override
    public int checkCallingUriPermission(Uri uri, int i) {
        return 0;
    }
}

```

```

@Override
public int checkCallingOrSelfUriPermission(Uri uri, int i) {
    return 0;
}

@Override
public int checkUriPermission(@Nullable Uri uri, @Nullable String s, @Nullable
String s1, int i, int i1, int i2) {
    return 0;
}

@Override
public void enforceUriPermission(Uri uri, int i, int i1, int i2, String s) {
}

@Override
public void enforceCallingUriPermission(Uri uri, int i, String s) {
}

@Override
public void enforceCallingOrSelfUriPermission(Uri uri, int i, String s) {
}

@Override
public void enforceUriPermission(@Nullable Uri uri, @Nullable String s, @Nullable
String s1, int i, int i1, int i2, @Nullable String s2) {
}

@Override
public Context createPackageContext(String s, int i) throws Pack-
ageManager.NameNotFoundException {
    return null;
}

@Override
public Context createContextForSplit(String s) throws PackageManager.NameNot-
FoundException {
    return null;
}

@Override
public Context createConfigurationContext(@NonNull Configuration configuration) {
    return null;
}

@Override
public Context createDisplayContext(@NonNull Display display) {
    return null;
}

```

```
        @Override
        public Context createDeviceProtectedStorageContext() {
            return null;
        }

        @Override
        public boolean isDeviceProtectedStorage() {
            return false;
        }
    };

    @Before
    public void SetUp() {
        db_building = new Db_building(context);
    }

    @Test
    public void testDeleteBuilding() {
        boolean valor_esperado = true ;
        boolean valor_real = db_building.deleteBuilding(2);
        assertEquals(valor_esperado, valor_real);
    }

    @Test
    public void testsearchBuilding(){
        Building buildingesperado = new Building(1,"90", 0);
        assertEquals(buildingesperado, db_building.searchBuilding(1));
    }

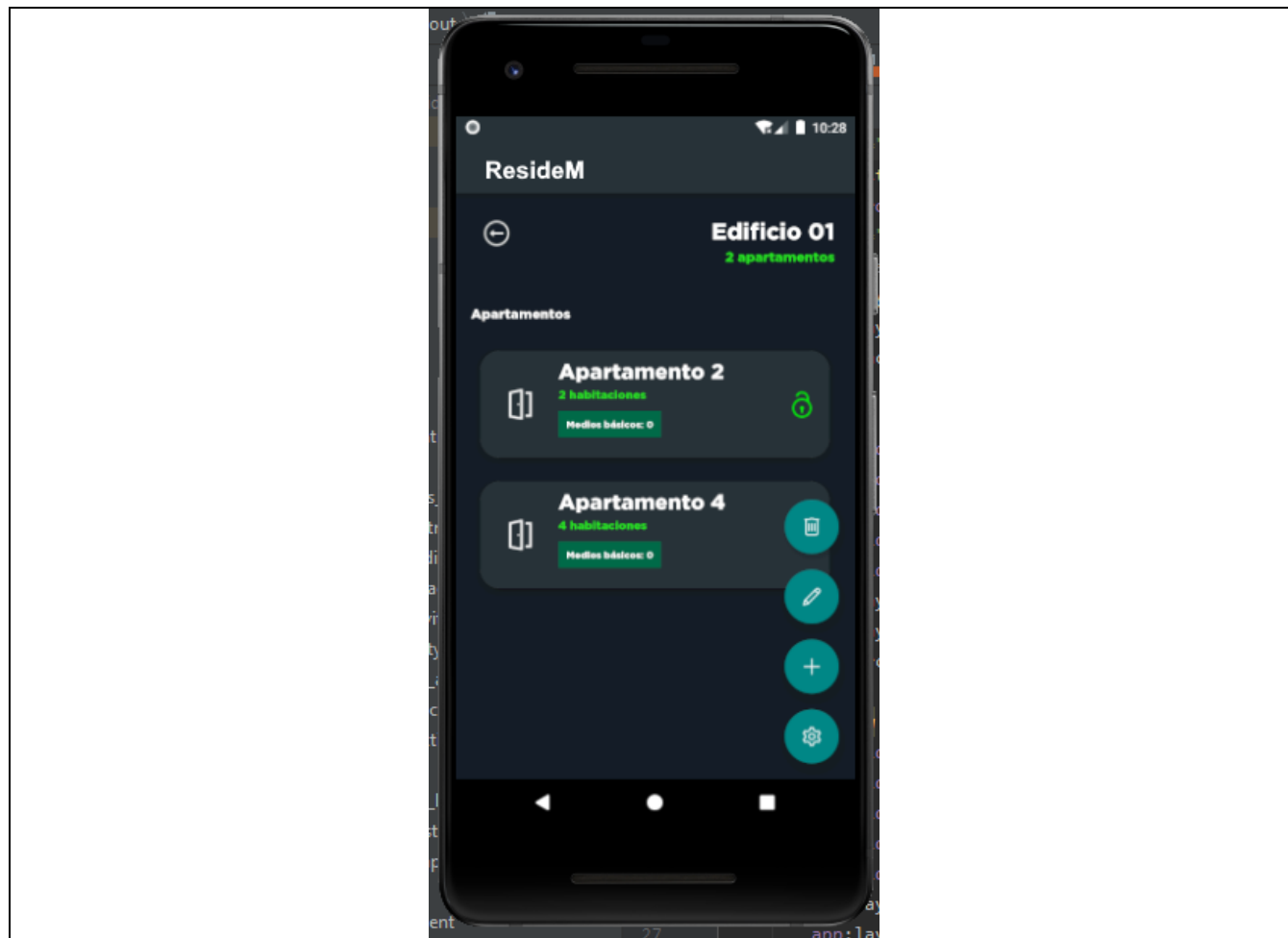
    @Test
    public void InsertBuilding () {
        boolean valor_esperado = true;
        assertEquals(valor_esperado, true);
    }
}
```

## Anexo 4 Historias de Usuario

<b>Número:</b> 1	<b>Nombre de la Historia de Usuario:</b> Autenticar Usuario	
<b>Cantidad de modificaciones a la Historia de Usuario:</b> 0		
<b>Usuario:</b> Auxiliar Responsable del Área, Administrador, Auxiliar Pedagógica, Director de Residencia.	<b>Iteración asignada:</b> 1 <b>Tiempo estimado:</b> 10h	
<b>Prioridad en negocio:</b> Alta		
<b>Riesgo en desarrollo:</b> Bajo	<b>Tiempo real:</b> 10h	
<b>Descripción:</b> Todos los usuario deben registrarse		
<b>Observaciones:</b> 1. No debe quedar ningún campo vacío		
<b>Prototipo de interfaz:</b>		



<b>Numero:</b> 12	<b>Nombre de la Historia de Usuario:</b> modificar edificio	
<b>Cantidad de modificaciones a la Historia de Usuario:</b> 0		
<b>Usuario:</b> Auxiliar Responsable del Área, Administrador, Auxiliar Pedagógica, Director de Residencia.	<b>Iteración asignada:</b> 1 <b>Tiempo estimado:</b> 12h	
<b>Prioridad en negocio:</b> Alta		
<b>Riesgo en desarrollo:</b> Bajo	<b>Tiempo real:</b> 12h	
<b>Descripción:</b> El usuario define cuales son los apartamentos disponibles del edificio en cuestión		
<b>Observaciones:</b>		
<b>Prototipo de interfaz:</b>		



<b>Numero:</b> 19	<b>Nombre de la Historia de Usuario:</b> Eliminar apartamento
<b>Cantidad de modificaciones a la Historia de Usuario:</b> 0	
<b>Usuario:</b> Auxiliar Responsable del Área, Administrador, Auxiliar Pedagógica, Director de Residencia.	<b>Iteración asignada:</b> 1 <b>Tiempo estimado:</b> 2h
<b>Prioridad en negocio:</b> Alta	
<b>Riesgo en desarrollo:</b> Bajo	<b>Tiempo real:</b> 2h
<b>Descripción:</b> El usuario selecciona en el menú desplegable si desea eliminar el apartamento.	
<b>Observaciones:</b>	



Prototipo de interfaz:

