Universidad de las Ciencias Informáticas

Facultad 2

BKDroid: Aplicación móvil para la gestión de los procesos de la Residencia Estudiantil en la UCI

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

Autor: Héctor Bobadilla Corbillón

Tutor: Ing. Victor Gabriel González Cardoso

La Habana, Junio de 2016
"Año 58 de la Revolución"

DECLARACIÓN DE AUTORÍA

Declaro que soy el único autor del trabajo BKDroid: Aplicación móvil para la gestión de los procesos en la residencia estudiantil de la UCI y autorizo a la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

Para que así conste firmamos la presente a los días 14 del mes de Junio del año 2016

Héctor Bobadilla Corbillón	Ing. Victor Gabriel González Cardoso
Firma del Autor	Firma del Tutor

DATOS DE CONTACTO

Tutor:

Ing. Víctor Gabriel González Cardoso: graduado en la Universidad de las Ciencias Informáticas en el año 2014. Profesor Instructor de la Facultad Introductoria en Ciencias Informáticas. Correo electrónico: victor@uci.cu

DEDICATORIA

A mis padres por darme su apoyo incondicional.

A mi hermano que es mi vida.

A mis abuelos y mi tía Maylin.

AGRADECIMIENTOS

A todos los que de una forma u otra tuvieron que ver con la realización del presente trabajo.

Especialmente para Victor, Hairon, Adonis y Ernesto muchas gracias.

RESUMEN

Con el transcurso de los años la utilidad de los dispositivos móviles ha aumentado producto a la gran variedad de servicios que pueden brindar. Las instituciones necesitan gestionar de forma eficaz sus actividades por lo que demandan nuevos modelos de sistemas de gestión. Cuba no está ajena a este fenómeno y se inserta en el desarrollo de soluciones informáticas activamente con instituciones como Universidad de las Ciencias Informáticas.

La residencia estudiantil dentro de la UCI es una de las áreas más extensas dentro del campus universitario. En ella se llevan a cabo diferentes procesos como: la cuartelería, evaluaciones integrales de cada estudiante, así como las incidencias constructivas por cada edificio y apartamento. Se puede mencionar además que las Instructoras, Especialistas y Directivos del área de residencia, que están obligados a visitar constantemente los edificios dentro del área residencial, recopilan la información con medios tradicionales para luego introducirlos en un sistema informático a través de la computadora. Esto hace que la recogida de la información sea engorrosa y propensa a errores humanos por los volúmenes de datos que se generan. Es por lo que se plantea como objetivo desarrollar una aplicación móvil que aumente la accesibilidad y disponibilidad de los datos generados en la residencia estudiantil de la Universidad de las Ciencias Informáticas.

En la presente investigación se describe el desarrollo de la aplicación móvil BKDroid, la cual fue implementada utilizando Android como Sistema Operativo.

Palabras clave: BKDroid, información, móvil, residencia.

ABSTRACT

Over the years the usefulness of mobile devices has increased product to the variety of services they can provide. Institutions need to effectively manage their activities so they demand new models of management systems. Cuba is not immune to this phenomenon and inserted in the development of solutions actively with institutions such as University of Information Science.

The student residence in the UCI is one of the largest on-campus areas. It carried out different processes such as: cuartelería, comprehensive assessments of each student as well as the constructive incidents for every building and apartment. It can also mention that instructors, specialists and managers of residential area, that are bound to constantly visit the buildings in the residential area, gather information through traditional means and then enter them into a computer system through the PC. This makes collection of information is cumbersome and prone to human error by the data volumes that are generated. It is so therefore seeks to develop a mobile application to increase the accessibility and availability of data generated in student residence of the University of Information Science.

In this research the development of the mobile application BKDroid described, which was implemented using Android operating system.

Keywords: BKDroid, information, mobile, residence.

ÍNDICE

INTRODUCCIÓN

En los últimos años se ha apreciado un creciente desarrollo de la telefonía móvil y un incremento de su uso en las actividades cotidianas. Más allá de limitarse solo a llamar o enviar mensajes de texto, esta tecnología, ha incorporado otras funcionalidades que van desde cámara de fotos, agenda electrónica, reloj despertador, calculadora, hasta acceso a la red de redes. Según la UIT, el organismo especializado de las Naciones Unidas para las Tecnologías de la Información y la Comunicación, en el año 2000 había en todo el mundo poco más de 700 millones de líneas móviles. Hoy, 16 años después, el número se sitúa en los 7.000 millones, prácticamente lo mismo que el número de habitantes del planeta (1).

Pese al desarrollo alcanzado por los dispositivos móviles no todos los servicios se pueden visualizar correctamente desde ellos. Estos presentan varias limitaciones entre las que pueden encontrarse: un teclado muy pequeño para la entrada de la información, una pantalla de reducidas dimensiones por lo que el despliegue de la información es limitado y el tiempo de respuesta de los sitios convencionales en este tipo de dispositivo es muy lento. Debido a esto es necesario desarrollar aplicaciones móviles especialmente diseñadas para adaptarse a los teléfonos celulares.

Las instituciones necesitan gestionar de forma eficaz sus actividades, demandan por ello nuevos modelos de sistemas de gestión que sean una herramienta útil para realizar estas tareas (2). Hasta el momento y siendo consecuentes con el desarrollo tecnológico alcanzado por la sociedad, algunas empresas han dado respuesta a este problema optando por la implementación de aplicaciones móviles para la gestión de sus principales procesos (3).

Cuba no está ajena a este fenómeno y se inserta en el desarrollo de soluciones informáticas activamente, tal y como quedó expresado en los Lineamientos de la Política Económica y Social del Partido y la Revolución en el VI Congreso del Partido Comunista de Cuba. Los centros educacionales llevan la vanguardia en esta área, encontrándose dentro de ellos la Universidad de las Ciencias Informáticas (UCI), casa de altos estudios líder en el desarrollo de software.

La UCI cuenta con un ecosistema de aplicaciones informáticas denominado MAYA, perteneciente a la Facultad Introductoria de Ciencias Informáticas (FICI), el cual persigue como objetivo integrar y gestionar los principales procesos de la vida universitaria. Dentro de este ecosistema se encuentra la

aplicación GERES: “Sistema para la gestión de la información asociada a la residencia estudiantil”, la que al estar implementada sobre la plataforma web no garantiza la portabilidad de la información (4).

La residencia es una de las áreas más extensas dentro del campus universitario cuenta con un aproximado de 210000 m^2. Se encuentra dividida en tres áreas residenciales: dos de ellas pertenecientes a estudiantes y la restante a profesores. En conjunto, las dos áreas de estudiantes están compuestas por 51 edificios para un total de 637 apartamentos donde conviven cerca de 2700 estudiantes (5). En ella se llevan a cabo los procesos referentes a la cuartelería y evaluaciones integrales de cada estudiante, así como las incidencias constructivas por cada edificio y apartamento, esto propicia que se generen grandes volúmenes de datos a procesar. Las instructoras, especialistas y directivos que constantemente visitan el área residencial hacen una recopilación de estos datos de forma manual utilizando hojas y lápices, lo que favorece la pérdida, deterioro y/o duplicidad de la información. De igual manera no es posible tener los datos referentes al estudiante cuando el personal de la residencia visita o se hace trabajo educativo en algún apartamento. Se puede agregar también que luego de culminada la visita se introducen los datos en el sistema GERES (https://geres.uci.cu) haciendo uso de las estaciones de trabajo que se encuentran en lugares fijos dentro del área residencial, lo que es propenso a errores humanos. A esto se le suma que en ocasiones la información recopilada no es introducida por la propia instructora lo que puede resultar en malas interpretaciones de la misma.

Es por ello que a partir de lo anteriormente descrito se plantea como **problema de la investigación:** ¿cómo aumentar la accesibilidad y disponibilidad de los datos generados en la residencia estudiantil de la Universidad de las Ciencias Informáticas?, definiéndose como **objeto de estudio** la gestión de los procesos asociados a residencias estudiantiles, enmarcado en el **campo de acción** de las aplicaciones móviles para la gestión de los procesos de la residencia estudiantil en la Universidad de Ciencias Informáticas.

Se plantea como **objetivo general** desarrollar una aplicación móvil que aumente la accesibilidad y disponibilidad de los datos generados en la residencia estudiantil de la Universidad de las Ciencias Informáticas y como **idea a defender:** si se desarrolla una aplicación móvil para la gestión de los procesos asociados a la residencia estudiantil en la UCI, entonces se aumentará la accesibilidad y disponibilidad de los datos.

Para dar cumplimiento al problema anterior se definen como **tareas de la investigación**:

1. Realización de un estudio del estado del arte de sistemas móviles para la gestión de residencias.
2. Selección de los métodos, técnicas y herramientas de desarrollo adecuadas para la implementación del software.
3. Descripción de los requisitos funcionales necesarios para dar solución al problema planteado.
4. Descripción de la integración de la aplicación móvil a la aplicación web GERES.
5. Implementación de los requisitos propuestos.
6. Validación de la solución propuesta a través de pruebas al sistema.

Para dar solución a las tareas antes mencionadas se utilizarán los siguientes **métodos** de investigación:

Métodos teóricos:

- ✓ **Analítico - sintético:** Se utilizará en el análisis de los elementos bibliográficos y definiciones sobre la gestión de procesos residenciales, con el objetivo de arribar a conclusiones que respalden la necesidad de la investigación.
- ✓ **Modelación:** Se empleará para la creación de modelos y diagramas, brindando así una reproducción ampliada de la realidad. Posibilita descubrir y estudiar nuevas relaciones y cualidades del objeto de estudio.

Métodos empíricos:

- ✓ **Observación:** Se empleará para la determinación de la problemática que da origen a la investigación.
- ✓ **Entrevista:** Permitirá contactar personas que están involucradas en el campo de acción y que puedan aportar información valiosa para la investigación. En este caso se entrevistarán a directivos de la Dirección General de la Residencia, así como instructoras y psicopedagogas de la Dirección de Residencia dos.

El presente trabajo de diploma se encuentra estructurado en tres capítulos los cuales están distribuidos de la siguiente forma:

CAPÍTULO 1. **FUNDAMENTACIÓN TEÓRICA**

Se abordan aspectos referentes a las diferentes aplicaciones móviles que existen en el mundo para la gestión de procesos residenciales. Se selecciona la plataforma de desarrollo y se mencionan características de las herramientas, tecnologías y metodología empleadas en la construcción de la misma.

CAPÍTULO 2. **DESCRIPCIÓN DE LA SOLUCIÓN PROPUESTA**

Se realiza el proceso de captura de los requerimientos funcionales y no funcionales que debe cumplir la aplicación, además de describir los procesos del negocio. Se define la representación arquitectónica de la aplicación, se hace énfasis en el patrón de arquitectura utilizado, así como los patrones de diseño que fueron empleados y se lleva a cabo el diseño del sistema.

CAPÍTULO 3. **IMPLEMENTACIÓN Y PRUEBAS**

Se fundamenta el diseño del modelo de datos físico, así como la integración que existe entre el sistema propuesto y la aplicación web GERES del ecosistema MAYA. Se describen las pruebas realizadas a las funcionalidades con el motivo de comprobar que el sistema cumpla con todos los requisitos funcionales.

CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA

En el presente capítulo se describe la aplicación web GERES del ecosistema MAYA. Se realiza un estudio de las diferentes aplicaciones móviles encargadas de la gestión de los procesos residenciales. Se selecciona la plataforma de desarrollo a utilizar y se hace énfasis en la descripción de la metodología, herramientas y tecnologías a utilizar para el desarrollo de la aplicación móvil.

1.1. GERES: Sistema para la gestión de la información asociada a la residencia estudiantil del ecosistema MAYA

En el año 2014 surge en la Facultad Introductoria de Ciencias Informáticas el proyecto MAYA: “Ecosistema de aplicaciones informáticas para la caracterización integral de los estudiantes en el contexto universitario”, el cual persigue como objetivo: integrar en una plataforma un conjunto de aplicaciones que tributen información en relación al estudiante y contribuir a mejorar el proceso de recopilación, procesamiento y persistencia de los datos que se genera entorno al estudiante. Este proyecto constituye además la base tecnológica de una estrategia pedagógica de orientación profesional con el fin de tributar al desarrollo de las potencialidades motivacionales, morales e intelectuales básicas de los estudiantes, para que actúen responsablemente en el proceso de su formación profesional (6).

Dentro de este ecosistema de aplicaciones se encuentra GERES: Sistema para la gestión de la información asociada a la residencia estudiantil en la Universidad de las Ciencias Informáticas. Aplicación web desarrollada sobre el marco de trabajo Symfony 2.7.9, utilizando postgreSQL como sistema gestor de base de datos en su versión 9.1 y Boostrap 3.0 como capa de presentación. Se encuentra estructurado en los siguientes módulos (4):

1. **Estructura y composición**: en este apartado se pueden crear tantas direcciones de residencia como se necesiten siempre asignando un director, permite mediante una búsqueda asociar recursos humanos (instructoras y técnicos) y edificios a esta.
2. **Cuartelería**: permite a las instructoras planificar las cuartelerías de los estudiantes por los edificios que estas atienden. Dispone de un generador de reportes en formato PDF con información nominal y estadística sobre las mismas.

3. **Caracterización integral:** se gestiona la información concerniente a los estudiantes como la caracterización inicial, familiares, evaluaciones integrales, frecuentes, de cuartelería y las indisciplinas cometidas.
4. **Mantenimiento constructivo**: permite gestionar las incidencias de mantenimiento permitiendo dar seguimiento a cada uno, además de realizar reportes en formato PDF de los mismos.
5. **Capacidad y matrícula**: permite la gestión de edificios, apartamentos y estudiantes.

GERES, se encuentra actualmente desplegado en su versión 1.0 y es utilizado por la Dirección General de Residencia desde Febrero 2016, brinda un conjunto de servicios que se encuentran públicos y constituyen una vía para intercambiar información con aplicaciones externas.

1.2. Sistemas informáticos existentes vinculados al campo de acción

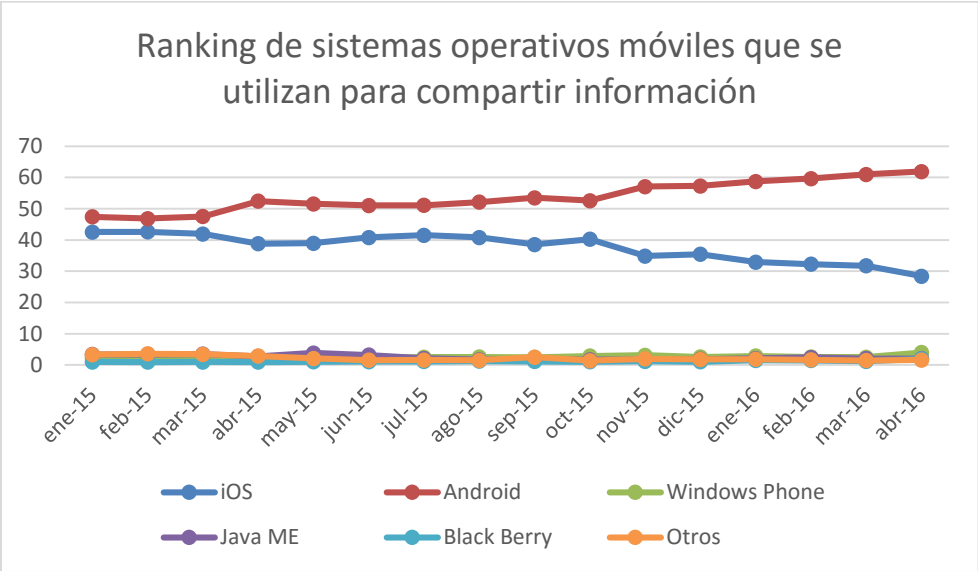
Se realizó una búsqueda de sistemas homólogos que gestionaran procesos residenciales con el objetivo de conocer los principales referentes teóricos asociados con el objeto de estudio. La misma arrojó la existencia de varias aplicaciones que manejaban información de centros residenciales, hoteles, becas universitarias y aglomeraciones municipales; pero estaban implementadas sobre entornos web. Sólo se encontró una aplicación bajo tecnologías móviles la cual se describe a continuación:

MisCondominios

Software de gestión de condominios, palabra que significa según la real academia de la lengua española *"propiedad que pertenece de manera colectiva e indivisible a un conjunto de personas sin asignación de cuotas entre ellas"* (7), el cual consta con una aplicación móvil que se ha desarrollado para el acceso de los propietarios del conjunto residencial. Esta constituye un canal de comunicación entre el administrador y los propietarios. Cada propietario del conjunto residencial tiene constancia de todo lo que sucede en su condominio, ya que puede dar de alta incidencias, visualizar su estado y responder, reservar las áreas comunes de las instalaciones, intercambiar mensajes con el administrador, revisar el estado de las cuentas del último año y votar en diferentes consultas a la comunidad. La presente aplicación no está dirigida a los administradores por lo que no es posible obtener estadísticas, asignar tareas ni dar evaluaciones. De igual manera no permite la integración hacia el ecosistema MAYA (8).

1.3. Selección de la plataforma de desarrollo

A continuación, se muestra una gráfica que muestra el comportamiento de la utilización de los sistemas operativos para la telefonía móvil en el periodo de enero de 2015 a abril de 2016 a nivel mundial.



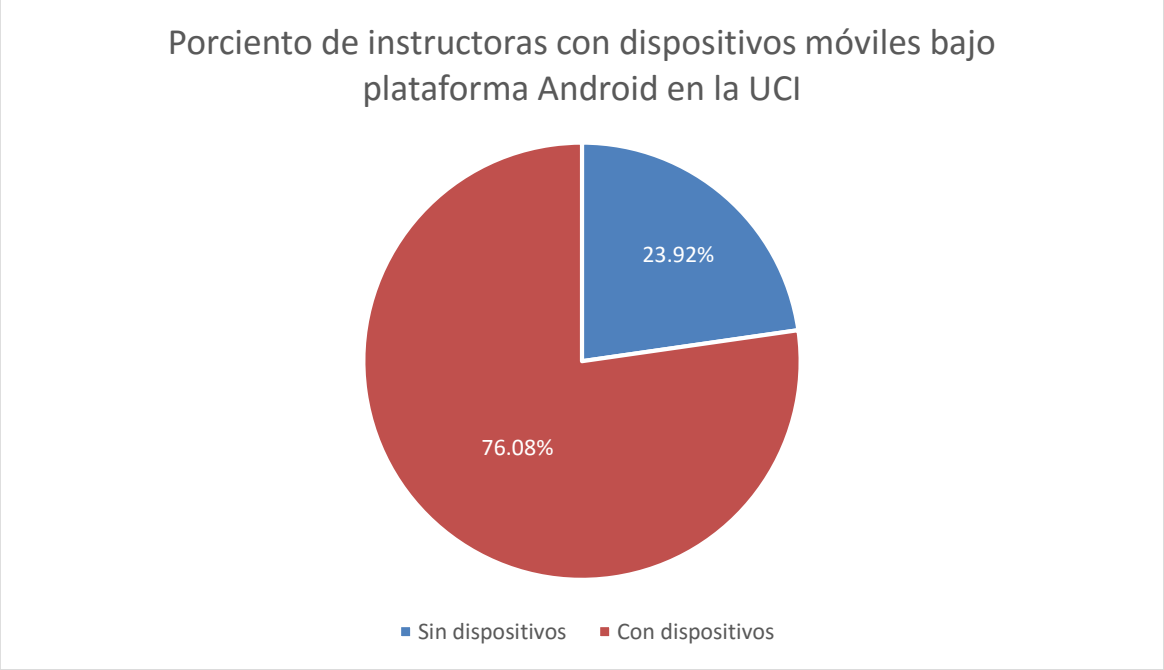
Porcentaje de uso de sistemas operativos que se utilizan para compartir información (9)

Según las tablas anteriormente mostradas, extraídas del sitio web http://netsharemarket.com se concluye que Android es el sistema operativo para tecnologías móviles más utilizado para compartir información en internet, dentro de sus principales ventajas se pueden mencionar que (10):

1. Puede instalarse prácticamente en todo tipo de dispositivos, sean móviles, portátiles e incluso microondas, siempre está presente en los terminales más potentes del mercado siendo una apuesta importante por fabricantes y operadoras por la posibilidad de que independientemente del potencial, gama o prestaciones del dispositivo, podrá adaptarse a la perfección a todo tipo de necesidades.
2. Al estar liberado con licencia Apache y código abierto lo convierte en un sistema operativo totalmente libre para que un desarrollador no solo pueda modificar su código sino también mejorarlo.
3. Da completa libertad al propietario de un terminal a instalar lo que desee, sea desde Android Market como un ejecutable aparte (*.apk) no limitando la libertad del usuario ni imponiendo software propietario para poder instalar música, archivos y documentos directamente desde el cable USB.

4. No solo cuenta con la comunidad más grande mundial de desarrolladores sino también el mayor movimiento de estos con multitud de eventos, concursos, competiciones y reuniones, así como múltiples vías de comunicación como foros y chats oficiales para fomentar la participación y la colaboración.
5. Puede ser instalado en teléfonos de cualquier fabricante o incluso en otros dispositivos, esto permite poder disfrutar de una amplia gama de terminales.
6. Es capaz de gestionar varias aplicaciones abiertas a la vez dejando en suspensión aquellas que no se utilicen y cerrarlas en caso de resultar ya inútiles para evitar un consumo de memoria.

Se realizó un estudio por el campus universitario con el objetivo de conocer el porciento de instructoras que poseen dispositivos propios, el que arrojo la existencia de 35 instructoras de un total de 46 con dispositivos inteligentes basados en la plataforma Android, lo que representa el 76,08% del total. La siguiente gráfica ilustra este comportamiento.



Se consultó a la Vicerrectoría de Tecnología de la UCI para obtener los modelos que han sido entregados por la institución en los últimos 5 años (mostrados en la siguiente tabla) correspondientes a tecnologías con Sistemas Operativos Android.

Teléfonos	Tabletas
Itelecom	THTF
Alcatel OT 4030	RCA
Alcatel OT 5030	
Alcatel OT C5	

Modelos de teléfonos entregados por la Vicerrectoría de Tecnología en la UCI.

Se puede agregar además que como parte de la política de la dirección de la Universidad se prevee la entrega de dispositivos móviles bajo la plataforma Android a las instructoras, especialistas y directivos de la residencia. Además, se ampliarán las áreas de conexión WiFi por los lugares estratégicos dentro del campus universitario. Es por todos los elementos anteriormente planteados que se selecciona como Sistema Operativo a Android para el cumplimiento del objetivo propuesto en la presente investigación.

1.4. Herramientas y tecnologías

El proyecto MAYA define un conjunto de herramientas y tecnologías para la implementación de las aplicaciones que integrarán el ecosistema. A continuación se describen las utilizadas para el desarrollo de aplicaciones móviles:

Android Studio 1.3.2

Constituye el Entorno de Desarrollo Integrado(IDE) oficial para el desarrollo de aplicaciones en Android, basado en IntelliJ IDEA (11). Es publicado de forma gratuita a través de la licencia Apache 2.0, está disponible para las plataformas Windows, Mac OS y GNU/Linux. Posibilita la renderización en tiempo real, posee soporte para la construcción basada en Gradle, incorpora plantillas para crear diseños comunes de Android y otros componentes (12).

Visual Paradigm para UML 8.0

Herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Es una herramienta multiplataforma que se integra con varios IDEs (Ambiente Integrado de Desarrollo) y soporta múltiples usuarios trabajando sobre un mismo proyecto. Además, importa y exporta diagramas en XML como imágenes (ya sea con extensiones jpg o png). Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación (13).

Java 8.77

Es un lenguaje de programación orientado a objetos donde el código fuente se compila y las clases que se generan son interpretadas por la máquina virtual de Java, siendo ésta la que mantiene el control sobre las clases que se estén ejecutando. Es un lenguaje multiplataforma; el mismo código Java que funciona en un sistema operativo, funcionará en cualquier otro que tenga instalada la máquina virtual.

La máquina virtual al ejecutar el código, realiza comprobaciones de seguridad, además el propio lenguaje carece de características inseguras, como por ejemplo los punteros. Java elimina muchas de las características de otros lenguajes como C++, para mantener reducidas las especificaciones del lenguaje y añadir características muy útiles como el *garbage collector* (reciclador de memoria dinámica). Reduce en un 50% los errores más comunes de programación en lenguajes como C y C++ al eliminar muchas de las características negativas de éstos (14).

Extensible Markup Language (XML) 1.0

Lenguaje de marcas desarrollado por el World Wide Web Consortium (W3C) utilizado para almacenar datos en forma legible. Propone un estándar para el intercambio de información estructurada entre diferentes plataformas (15).

SQLite 3

Biblioteca que tiene incorporado un motor de base de datos Structured Query Languages (SQL). A diferencia de la mayoría de bases de datos en SQL, SQLite no tiene un proceso de servidor independiente, SQLite lee y escribe directamente en archivos de disco ordinarios. Es una base de datos completa de SQL con varias tablas, índices, activadores y vistas, todo está contenido en un solo archivo de disco. El formato de archivo de base de datos es multiplataforma por lo que se extiende su uso a cualquier sistema operativo existente (16).

GreenDao 2.0

Proyecto de código abierto que ayuda a los desarrolladores de Android a trabajar con bases de datos en SQlite. Utilizarlo requiere de no solo conocimientos de SQL, greenDao hace uso de clases en Java para mapear los objetos dentro de la base de datos con lo cual insertar, eliminar, actualizar o crear relaciones sea una tarea sencilla. Minimiza el uso de memoria lo cual es favorable ya que su uso se extiende a los teléfonos mientras que las bibliotecas para su uso no exceden los 65 kilobyte de capacidad. Proporciona una capa de seguridad que evita los ataques SQL injections (inserción de código malicioso en variables incrustadas en sentencias Transact-SQL) (17).

Gradle 2.2.1

Herramienta de código abierto la cual se encarga de la automatización de la construcción de código. Dispone de una gran flexibilidad que permite trabajar utilizando otros lenguajes, no solo Java; posee un sistema de gestión de dependencias sólido. usa un Domain Specific Language(DSL) basado en Groovy para declarar las formas de construir el proyecto (18).

Lenguaje Unificado de Modelado (UML) 2.0

Serie de normas y estándares gráficos que brinda las herramientas necesarias para visualizar, especificar, construir y documentar un sistema. Se puede aplicar en el desarrollo de software gran variedad de formas para dar soporte a una metodología de desarrollo de software (19).

1.5. Metodología para el desarrollo del software

Se entiende por metodología a un marco de trabajo en el cual se estructura, planifica y controla el proceso de desarrollo de un sistema de información (20). A continuación, se describe la utilizada por el autor:

Extreme Programing (XP)

Es el más destacado de los procesos ágiles de desarrollo de software. La programación extrema se diferencia de las metodologías tradicionales principalmente en que pone más énfasis en la adaptabilidad que en la previsibilidad. Los cambios de requisitos sobre la marcha son un aspecto natural, inevitable e incluso deseable del desarrollo de proyectos. La adaptación a los cambios de requisitos en cualquier punto de la vida del

proyecto es una aproximación mejor y más realista que intentar definir todos los requisitos al comienzo del proyecto e invertir esfuerzos después en controlar los cambios en los requisitos (21).

Fases

Cuenta con cinco fases, en la fase de **exploración** los clientes escriben las historias de usuario. Al mismo tiempo el equipo del proyecto se familiariza con las herramientas, la tecnología y las prácticas que utilizarán en el proyecto. En la fase de **planeación** configura la prioridad para las historias de usuario y se realiza un contrato del contenido para la primera entrega. Los programadores primero estiman cuánto esfuerzo requieren para cada historia y se hace una programación de acuerdo a esta estimación. La fase de **iteraciones** hacia la entrega incluye varias iteraciones del sistema antes de la primera entrega. La programación que se determinó en la etapa de planeación es dividida en un número de iteraciones donde cada una tomará de una a cuatro semanas para ser implementada. La primera iteración crea la arquitectura de todo el sistema. La fase de **producción** requiere pruebas extras y chequeos de la ejecución del sistema antes de que sea entregado al cliente. En ésta fase, se pueden encontrar nuevos cambios y se toma la decisión si serán incluidos en la entrega actual. La fase de **mantenimiento** requiere también un esfuerzo para soportar las tareas de los clientes. Éste es el momento en el proceso XP cuando la documentación necesaria del sistema es finalmente escrita porque no habrá más cambios en la arquitectura, diseño o código (21).

Ventajas

XP da lugar a una programación sumamente organizada donde ocasiona eficiencias en el proceso de planificación y pruebas las cuales cuentan con una tasa de errores muy pequeña. Propicia la satisfacción del programador aumentando la comunicación entre los clientes y los desarrolladores. Facilita los cambios mientas que permite ahorrar mucho tiempo y dinero. Puede ser aplicada a cualquier lenguaje de programación (21).

Desventajas

Es recomendable emplearla solo en proyectos a corto plazo por lo que, en caso de fallar, las comisiones son muy altas. Puede no siempre ser más fácil que el desarrollo tradicional (21).

El proyecto define esta metodología ya que los equipos de desarrollo son pequeños, el cliente forma parte del proyecto y la documentación generada es de pequeña envergadura.

1.6. Conclusiones parciales

Con el estudio del estado del arte realizado se logró identificar que no se conoce una aplicación móvil que permita gestionar los procesos residenciales y resuelva las problemáticas de esta área en la UCI. La definición de un ambiente tecnológico basado en software libre permitió seleccionar las herramientas y tecnologías adecuadas para el desarrollo de la aplicación. A partir de la investigacion realizada de las tendencias de sistemas operativos para el desarrollo de tecnologías móviles y la cantidad de dispositivos con que cuenta el área residencial de la UCI se decidió implementar la solución propuesta en Android.

CAPÍTULO 2. DESCRIPCIÓN DE LA SOLUCION PROPUESTA

En este capítulo se describen los procesos asociados a la gestión de la residencia estudiantil de la UCI, se identificaron los requisitos funcionales y no funcionales a tener en cuenta para la implementación de la aplicación móvil, así como la descripción de la arquitectura propuesta y la integración de dicha aplicación con el ecosistema MAYA.

2.1. Descripción de los procesos de negocio

GERES define un conjunto de procesos para gestionar la información asociada a la residencia estudiantil, tomándolos como base, se describen a continuación los que serán utilizados en la presente investigación:

2.1.1. Planificación y evaluación de la cuartelería

El proceso de planificación comienza un mes antes de que la misma se realice, la instructora con el listado de estudiantes asigna una fecha a cada uno indicándole el día que realizarán la misma. Una vez concluida se emite una evaluación teniendo en cuenta el desempeño del estudiante en la actividad.

2.1.2. Incidencias constructivas

Cuando ocurre una incidencia constructiva el estudiante lo reporta a la instructora asociada a su edificio. Luego, la instructora informa al departamento de mantenimiento donde se archiva el reporte guardando un identificador único, el apartamento, la fecha en que se realizó y la pieza o materiales necesarios. Al solucionarse esta incidencia se le informa a la instructora su culminación la cual se encarga de informarle a los estudiantes y recoger su grado de satisfacción con el trabajo realizado.

2.1.3. Evaluaciones generales y frecuentes

Mensualmente cada becario es evaluado teniendo en cuenta su comportamiento, cuartelería realizada, limpieza y organización durante todo el periodo a evaluar. Las evaluaciones emitidas están en la escala de Bien (B), Regular (R), Mal (M). Las instructoras o psicopedagogas pueden además realizar evaluaciones frecuentes según lo estimen conveniente.

2.2. Requisitos funcionales y no funcionales

2.2.1. Técnicas de extracción de requisitos

Entrevista:

La entrevista se utilizó para obtener información cualitativa como opiniones o descripciones donde existe un moderador y un entrevistado (22). Abarcó cuatro pasos principales: identificar los entrevistados, preparar y realizar la entrevista y la documentación de los resultados. Se puede observar el guion con que se conformó la misma en el Anexo 3.

Observación:

Con la aplicación de esta técnica se pudo obtener de primera mano la forma de realización de los procesos en la residencia estudiantil, además de verificarse que estos seguían los pasos establecidos para su desarrollo. En muchos de los casos el papel no muestra lo que sucede realmente en la práctica, de ahí la importancia de su empleo (22).

Estudio de la documentación:

Tomando como base el estudio los reportes, evaluaciones proporcionados por las instructoras y el reglamento disciplinario de la residencia, se obtuvo información con respecto a la forma de organización y el flujo de los procesos en la residencia estudiantil. La documentación en ocasiones no reflejaba la forma en que realmente se desarrollan las actividades, sin embargo, fue de gran importancia para introducir a los desarrolladores el dominio de la operación y el vocabulario que se emplea en esta esfera de la vida universitaria (22).

A partir de la utilización de las técnicas anteriores se obtuvo un total de 35 requisitos funcionales y 5 no funcionales. Es importante señalar que para la definición de estos requisitos se tuvo en cuenta los definidos por la aplicación GERES, pero al estar desarrollada en un entorno web diferían de los necesarios para el desarrollo de la presente aplicación móvil.

2.2.2. Definición de requisitos funcionales

Los requisitos funcionales son capacidades o condiciones que el sistema debe poseer. Se mantienen invariables sin importar con qué propiedades o cualidades se relacionen. Describen su funcionalidad, los servicios que de él se esperan, o los que proveerá y definen las funciones que el sistema será capaz de

realizar (23). Estos fueron clasificaron teniendo en cuenta su prioridad y complejidad en Alta, Media y Baja como se muestra en la siguiente tabla.

No.	Nombre	Descripción	Prioridad	Complejidad
RF1	Reporte de Incidencias constructivas por apartamento	Generar reporte con un listado de incidencias constructivas por apartamentos	Media	Baja
RF2	Reporte de Incidencias constructivas por edificio	Generar reporte con un listado de incidencias constructivas por edificios	Media	Baja
RF3	Reporte de necesidades de recursos por edificio	Generar reporte con un listado de necesidades de recursos por edificio	Media	Baja
RF4	Listar incidencias constructivas	Listar las incidencias constructivas de un apartamento	Alta	Media
RF5	Adicionar incidencias constructivas	Adicionar incidencia constructivas a un apartamento	Alta	Media
RF6	Editar incidencias constructivas	Editar incidencia constructivas a un apartamento	Alta	Media
RF7	Eliminar incidencias constructivas	Eliminar incidencia constructivas a un apartamento	Alta	Media
RF8	Estudiantes evaluados	Listar estudiantes evaluados por apartamento	Alta	Alta
RF9	Estudiantes por evaluar	Listar estudiantes pendientes a evaluar por apartamento	Media	Alta
RF10	Adicionar evaluación	Adicionar evaluación a estudiante	Baja	Media
RF11	Eliminar evaluación	Eliminar evaluación a estudiante	Baja	Media
RF12	Editar evaluación	Editar evaluación a estudiante	Baja	Media
RF13	Reporte con listado de evaluaciones por edificio	Generar un reporte con un listado de evaluaciones por edificio	Baja	Baja
RF14	Reporte con listado de evaluaciones por apartamento	Generar un reporte con un listado de evaluaciones por apartamento	Baja	Baja

RF15	Planificados con cuartelería por apartamento	Listar estudiantes planificados con cuartelería por apartamento	Alta	Alta
RF16	Sin planificar cuartelería por apartamento	Listar estudiantes por planificar con cuartelería por apartamento	Alta	Alta
RF17	Adicionar cuartelería	Adicionar planificación de cuartelería a estudiante	Media	Media
RF18	Eliminar cuartelería	Eliminar planificación de cuartelería a estudiante	Media	Media
RF19	Editar cuartelería	Editar planificación de cuartelería a estudiante	Media	Media
RF20	Planificados con cuartelería por fecha	Listar estudiantes planificados con cuartelería por fecha	Alta	Alta
RF21	Adicionar evaluación de la cuartelería	Adicionar evaluación de la cuartelería a estudiante	Media	Media
RF22	Editar evaluación de la cuartelería	Editar evaluación de la cuartelería a estudiante	Media	Media
RF23	Eliminar evaluación de la cuartelería	Eliminar evaluación de la cuartelería a estudiante	Media	Media
RF24	Mostrar caracterización	Mostrar caracterización a estudiante	Media	Baja
RF26	Reporte de la caracterización	Generar reporte de la caracterización de estudiante	Baja	Baja
RF27	Listar edificios	Listar edificios asignados a instructora	Alta	Alta
RF28	Listar apartamentos	Listar apartamentos por edificio	Alta	Alta
RF29	Listar estudiantes	Listar estudiantes por apartamento	Alta	Alta
RF30	Reporte de estadística por apartamento	Generar un reporte de estadísticas por apartamento	Media	Baja
RF31	Reporte de estadística por edificio	Generar un reporte de estadísticas por edificio	Media	Baja

RF32	Reporte con listado de estudiantes por apartamento	Generar un reporte con listado de estudiantes por apartamento	Media	Baja
RF33	Reporte con listado de estudiantes por edificio	Generar un reporte con listado de estudiantes por edificio	Media	Baja
RF34	Reporte con listado de apartamentos por edificio	Generar reporte con el listado de apartamentos por edificio	Media	Baja
RF35	Autenticar usuario	Permitir autenticar con un nombre de usuario y una contraseña UCI	Alta	Alta

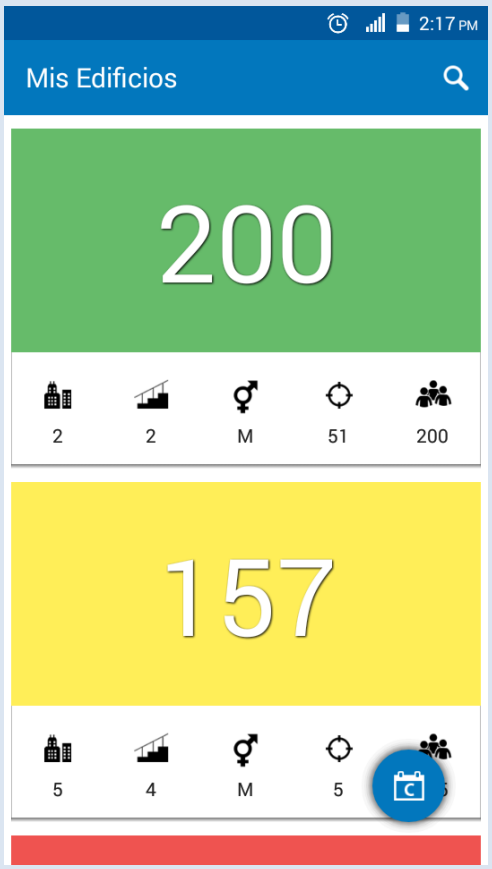
Tabla de requisitos funcionales.

2.2.3. Descripción de requisitos funcionales

Las Historias de Usuario (HU) son unos de los artefactos más importantes que genera la metodología XP, son una representación de un requisito de software escrito en una o dos frases utilizando el lenguaje común del usuario. Estas proporcionaran los detalles sobre la estimación del riesgo y cuánto tiempo conllevará la implementación de cada una. Son una forma rápida de administrar los requisitos de los usuarios sin tener que elaborar gran cantidad de documentos formales y sin requerir de mucho tiempo para administrarlos (24).

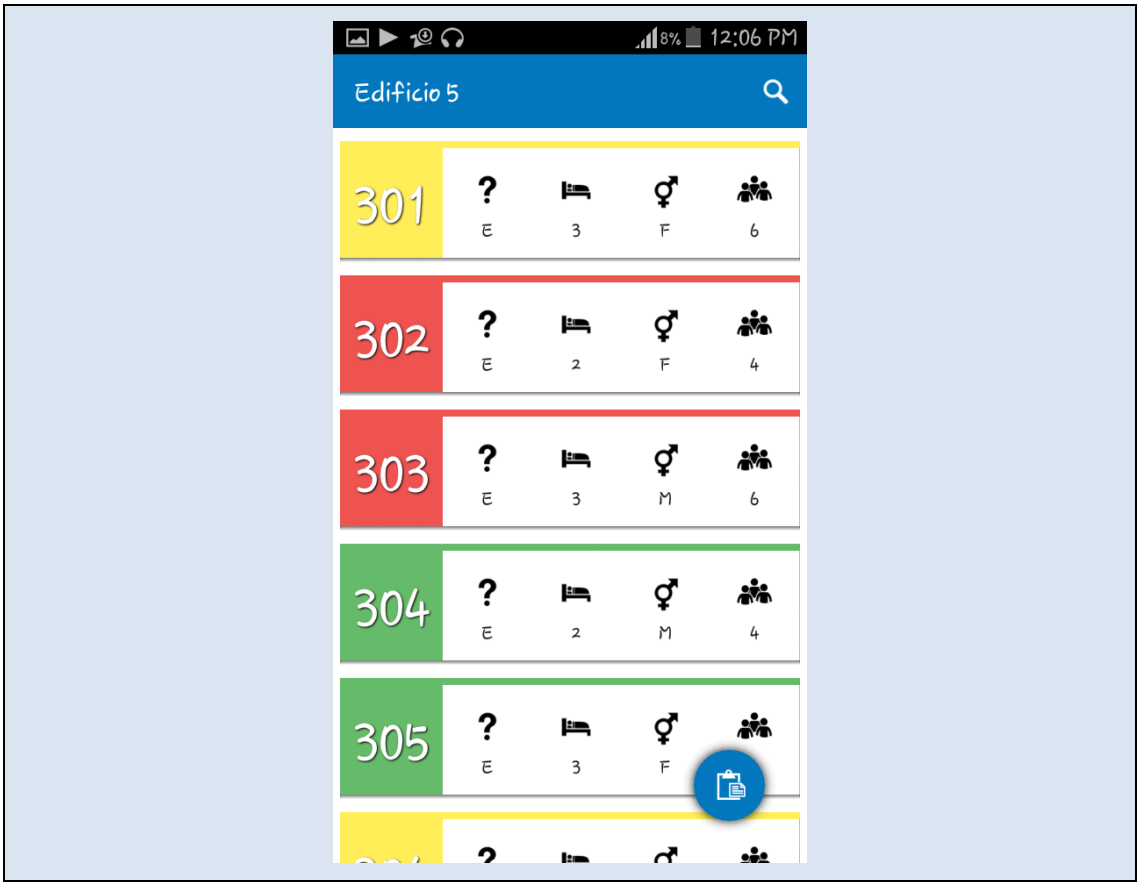
Se confeccionaron 35 HU en correspondencia con las diferentes funcionalidades identificadas. A continuación, se describen la HU correspondientes al módulo de Capacidad y matricula, el resto se encuentran en el expediente de proyecto.

Historia de usuario	
Numero: 27	**Nombre:** Listar edificios
Programador: Héctor Bobadilla Corbillón	
Prioridad en Negocio: Alta	**Riesgo en desarrollo:** Alto
Puntos estimados: 8	**Iteración asignada:** 1
Descripción: Al entrar al sistema se muestra un listado con los edificios asignados a dicho usuario. Se muestran el número, el estado constructivo dado por tres colores rojo (malo), amarillo (regular) y verde(bueno), la facultad a la que pertenece, la	

cantidad de pasos de escaleras, el sexo, el área en que se encuentran ubicados y la cantidad de personas que se encuentran habitando. En la parte superior se encuentra la barra de título, está conformada por un nombre y un botón de buscar con el que podemos filtrar por número los edificios. En la parte inferior derecha se encuentra un botón que permite acceder a otra vista, que muestra el o los cuarteleros de los edificios pertenecientes al usuario autenticado.
Observaciones: Es necesario haberse autenticado en el sistema para poder ver los edificios.
Prototipo de interfaz: 

Historia de usuario correspondiente al requisito funcional 27.

Historia de usuario	
Numero: 28	**Nombre:** Listar apartamentos
Programador: Héctor Bobadilla Corbillón	
Prioridad en Negocio: Alta	**Riesgo en desarrollo:** Alto
Puntos estimados: 6	**Iteración asignada:** 1
Descripción: Al presionar en un edificio se abre una interfaz en la que se lista los apartamentos asociados. De estos se muestra el su número, su estado constructivo que está dado por tres colores rojo(malo), amarillo(regular) y verde(bueno), la clasificación, la cantidad de cuartos, el sexo de los habitantes y cantidad de personas que viven en él. En la parte superior se encuentra la barra de título, está conformada por el número del edificio seleccionado y un botón de buscar con el que podemos filtrar por número apartamentos. En la parte inferior derecha se encuentra un botón que conduce a otra vista, esta muestra un listado con los tipos de reportes que se pueden realizar al edificio seleccionado.	
Observaciones: cuando el apartamento este inhabilitado mostrarlo de color negro y no permitir que sea seleccionado.	
Prototipo de interfaz:	



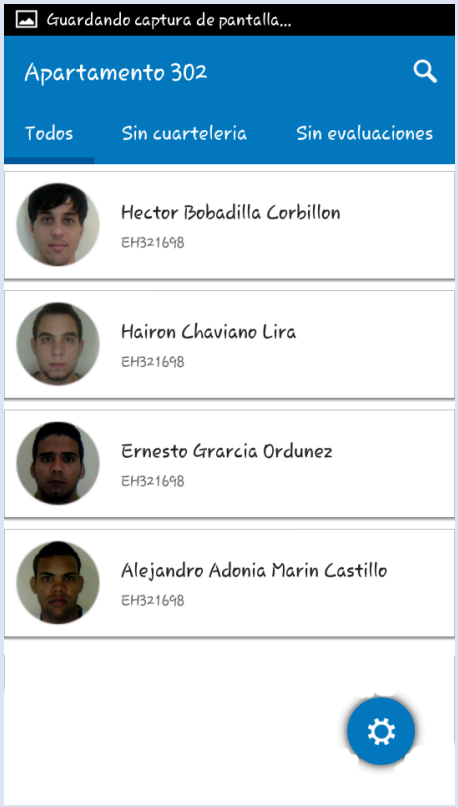
Historia de usuario correspondiente al requisito funcional 28

Historia de usuario	
Numero: 29	**Nombre:** Listar estudiantes
Programador: Héctor Bobadilla Corbillón	
Prioridad en Negocio: Alta	**Riesgo en desarrollo:** Alto
Puntos estimados: 11	**Iteración asignada:** 1
Descripción: Al presionar en un apartamento se obtiene en una nueva vista el listado de las personas residentes en él, es posible mediante tres botones que se encuentran en la parte superior de la lista aplicarle un filtro para obtener los que faltan por	

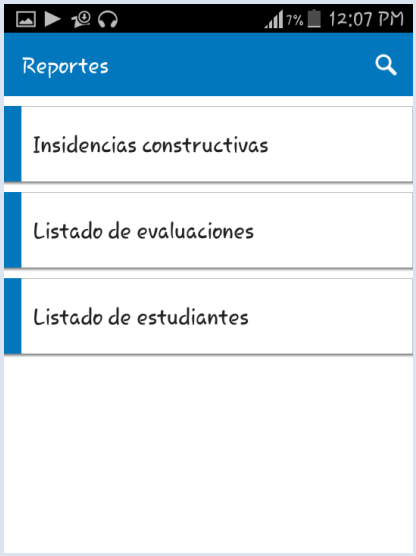
planificar cuartelería y los que aún no han sido evaluados. De los residentes se muestra el nombre, el solapín y la foto. En la parte superior se encuentra la barra de título, está conformada por el número del apartamento seleccionado y un botón de buscar con el que podemos filtrar por nombre los estudiantes. En la parte inferior derecha se encuentra un botón que al presionar sobre él se muestran otros dos botones, uno de ellos nos dirige a una vista donde es posible realizar varios tipos de reportes asociados al apartamento en cuestión, mientras que el otro nos dirige a una vista donde es posible la gestión de las incidencias constructivas del apartamento.

Observaciones:

Prototipo de interfaz:



Historia de usuario correspondiente al requisito funcional 29

Historia de usuario	
Numero: 32	**Nombre:** Reporte con listado de estudiantes por apartamento
Programador: Héctor Bobadilla Corbillón	
Prioridad en Negocio: Media	**Riesgo en desarrollo:** Bajo
Puntos estimados: 2	**Iteración asignada:** 2
Descripción: Al seleccionar el botón correspondiente a la opción de generar reportes por apartamento se muestra una vista que contiene un listado de reportes asociados a un apartamento donde se encuentra las opciones: “Incidencias constructivas”, “Listado de evaluaciones” y “Listado de estudiantes”, al presionar en una de estas se genera un documento en formato PDF con los becarios que pertenecen al apartamento. Este se guarda de forma automática en el directorio “/storage/emulated/0/BKDroid/Reportes”.	
Observaciones:	
Prototipo de interfaz: 	

Historia de usuario correspondiente al requisito funcional 31

Historia de usuario	
Numero: 33	**Nombre:** Reporte con listado de estudiantes por edificio
Programador: Héctor Bobadilla Corbillón	
Prioridad en Negocio: Media	**Riesgo en desarrollo:** Baja
Puntos estimados: 2	**Iteración asignada:** 2
Descripción: Al seleccionar el botón correspondiente a la opción de generar reportes por edificio se muestra una vista que contiene un listado de reportes asociados a un edificio donde se encuentra las opciones: “Incidencias constructivas”, “Necesidades de recursos”, “Listado de evaluaciones” y “Listado de estudiantes”, al presionar en una de estas se genera un documento en formato PDF con los becarios que pertenecen al edificio. Este se guarda de forma automática en el directorio “/storage/emulated/0/BKDroid/Reportes”.	
Observaciones:	
Prototipo de interfaz: 	

Historia de usuario correspondiente al requisito funcional 32

2.2.4. Descripción de requisitos no funcionales

Los requisitos no funcionales son propiedades o cualidades que el producto debe tener, haciéndolo atractivo, usable, rápido y confiable. Normalmente están vinculados a los requisitos funcionales y son fundamentales en el éxito del software (23).

Usabilidad

RNF 1: Interacción del usuario con la aplicación.

- La aplicación está dirigida a las instructoras, psicopedagogas y directivos de la residencia, las cuales son personas que no tienen una formación técnica avanzada por lo que es necesario que esta sea intuitiva.

Rendimiento del sistema

RNF 2: Disponibilidad.

- La aplicación debe funcionar manteniendo una copia de los datos localmente en el dispositivo móvil.

RNF 3: Seguridad.

- Los usuarios para la entrada a la aplicación deben ser autenticados, garantizando la confiabilidad y la confidencialidad de los datos. Las credenciales de seguridad que se intercambia desde y hacia la aplicación debe estar cifrada con el algoritmo MD5.

Restricciones de diseño

RNF 4: Requerimiento de hardware.

- El dispositivo debe poseer conexión con redes inalámbricas.

RNF 5: Requerimientos de software.

- Se debe poseer un dispositivo que utilice la API 14 o mayor y una versión de Android igual o superior a 4.0.

2.2.5. Validación de requisitos

La validación de los requisitos, tiene como objetivo comprobar que estos son consistentes, completos, precisos, realistas, verificables y definen lo que el usuario desea del producto final. La actividad de validación tiene como entrada el documento de requisitos y como salida se obtiene una lista de problemas y de acciones recomendadas, tal y como se muestra en la siguiente imagen (25).

La validación en el proceso de requisitos

Los métodos para validar los requisitos son (25):

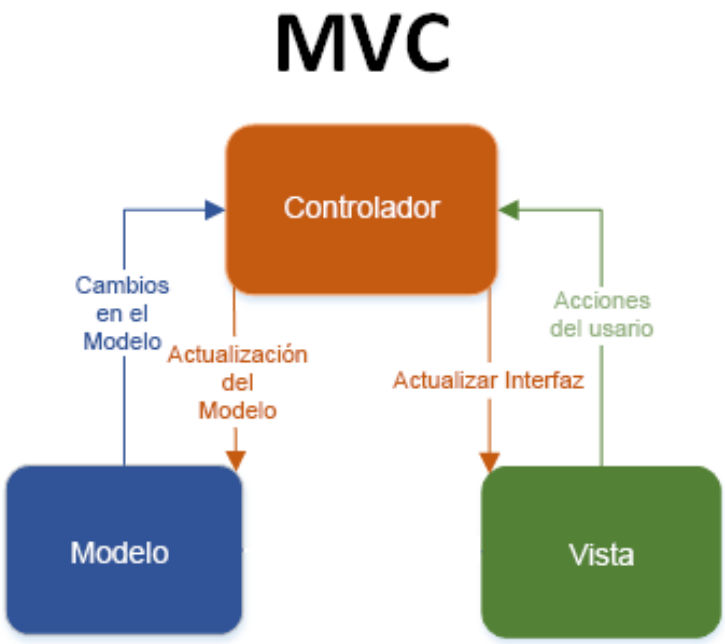
- **Revisión de requisitos:** consisten en reuniones donde un equipo de analistas intenta localizar errores en el documento de especificación.
- **Prototipado:** se construye una maqueta del fututo sistema de software a partir de los requisitos recogidos en la especificación. Esta maqueta será evaluada por el cliente y usuarios para comprobar su corrección y completitud.
- **Generación de casos de prueba (test de requisitos):** tiene como objetivo comprobar la verificabilidad de los requisitos. Consiste en la definición de casos de prueba que permitan verificar el cumplimiento de los requisitos funcionales.

El método utilizado fue el de revisión de requisitos, todos los parámetros mencionados referente a este método fueron seguidos correctamente. Las discusiones de estas reuniones están recogidas en las minutas de trabajo las cuales se archivan el expediente de proyecto. Se puede agregar además que la Dirección General de Residencia aprobó dichos requisitos en reunión efectuada con los directores de residencia y una representación de las instructoras y psicopedagogas.

2.3. Arquitectura del sistema

2.3.1. Modelo Vista Controlador (MVC)

Se utiliza como patrón arquitectónico el denominado **Modelo Vista Controlador (MVC),** recomendado por Google para el desarrollo de aplicaciones móviles. Su principal bondad consiste en separar los datos de una aplicación, la interfaz de usuario y la lógica de negocios en tres componentes distintos de manera que si uno de estos componentes se modifica el resto no sufre cambios (26).



Patrón de desarrollo Modelo Vista Controlador (MVC)

El **modelo** son las representaciones que se construyen basadas en la información con la que operará la aplicación, está compuesto por las clases y métodos encargados del mapeo de la base de datos. La **vista** es la interfaz con la que va a interactuar el usuario. En Android, las interfaces se construyen en lenguaje XML (15) y se encuentran en una carpeta llamada “layout” ubicada en el directorio “app/src/main/res”. Componen al **controlador** las clases que permiten darle funcionalidad a las interfaces y consumir información del modelo. Se definen con un nombre seguido de la palabra Activity las que se guardan en la carpeta “java” ubicada en el directorio “app/src/main/” (27).

2.3.2. Módulos

En programación, un módulo es un fragmento de un programa que se desarrolla de forma independiente. Esta independencia hace posible un mecanismo de compilación por separado que limita la complejidad del programa que se está desarrollando (28).

A partir de los requisitos funcionales definidos en el epígrafe 2.2.2, se estructura la solución de la presente investigacion en siete módulos, los cuales se describen a continuación:

Módulo	Descripción
Mantenimiento constructivo	Gestiona las incidencias constructivas en la residencia, genera reportes por apartamentos y edificios.
Evaluaciones integrales	Asigna las evaluaciones a los estudiantes. Lista estudiantes evaluados por apartamento, los pendientes a evaluar y genera reporte de evaluaciones por edificio y apartamento.
Planificación y evaluación de la cuartelería	Gestiona la planificación de la cuartelería. Lista los estudiantes planificados dada una fecha como criterio de búsqueda, los planificados y por planificar. Asigna la evaluación a los estudiantes por el desempeño de esta actividad.
Caracterización de los estudiantes	Muestra y genera reportes de la caracterización de un estudiante (datos personales, adicciones, evaluaciones frecuentes y generales).
Capacidad y matrícula	Genera reporte estadísticos y nominales de los estudiantes por apartamento y edificios.
Administración y seguridad	Garantiza la confidencialidad de la información en la aplicación. Gestiona los usuarios y roles. Administra las variables de configuración del software.
Sincronización	Mediante la arquitectura REST sincroniza la base de datos local del dispositivo con la aplicación web GERES.

Tabla con la descripción de los módulos.

2.3.3. Patrones de diseño

Lograr un diseño simple, pero a la vez robusto requiere en gran medida del empleo de buenas prácticas. Se tuvieron en cuenta al modelar el sistema los conceptos de bajo acoplamiento para evitar las dependencias excesivas, y la alta cohesión tratando de que cada clase realice labores únicas y bien relacionadas, siempre con la intención de lograr un punto de equilibrio entre ambos. Se empleó el patrón experto para la asignación de tareas, siendo la clase que cuenta con los datos involucrados la responsable de ejecutar las funciones, y el controlador para manejar los eventos del sistema. A continuación se detallan los patrones utilizados durante la etapa de desarrollo (29):

Patrones GRASP

Controlador: Se evidencia en las clases Activity, las cuales manejan todas las peticiones que se realizan por el usuario. Cada una de estas clases está vinculada con su correspondiente fichero XML el cual contiene la información visual.

Experto: Consiste en que la responsabilidad de la creación de un objeto o la implementación de un método, debe recaer sobre la clase que conoce toda la información necesaria para crearlo. De este modo se obtiene un diseño con mayor cohesión y así la información se mantiene encapsulada, es decir, disminuye el acoplamiento. En la aplicación desarrollada se aprecia este patrón en las clases entidades ya que se encargan de gestionar la información asociada a la base de datos.

Alta cohesión: Se le asigna a cada clase las responsabilidades que le corresponde. Se establecen las condiciones para que cada clase colabore con las demás en la resolución de tareas que las implican y que no son capaces de resolver por sí solas.

Creador: Guía la asignación de responsabilidades relacionadas con la creación de objetos. El propósito fundamental de este patrón es encontrar un creador que se debe conectar con el objeto producido en cualquier evento. Al escogerlo como creador, se da soporte también al bajo acoplamiento lo cual supone menos dependencias respecto al mantenimiento y mejores oportunidades de reutilización. En el caso de la aplicación desarrollada este patrón fue utilizado en las clases controladoras en las cuales se crean objetos de las entidades para su manipulación.

Bajo acoplamiento: Las clases que implementan la lógica de negocio y de acceso a datos se encuentran en el modelo, estas clases no tienen asociaciones con la vista por lo que la dependencia en este caso es baja.

En la solución propuesta se puede apreciar que las clases entidades son las más reutilizadas puesto que son requeridas para varias de las funcionalidades dentro de un mismo subproceso.

2.4. Conclusiones parciales

La utilización de las técnicas de obtención de requisitos permitió la definición de 35 funcionales y 5 no funcionales. El estudio y análisis del patrón Modelo Vista Controlador garantizó una correcta construcción de la arquitectura de la aplicación, la cual fue dividida en siete módulos: mantenimiento constructivo, planificación y evaluación de la cuartelería, administración y seguridad, sincronización, evaluaciones integrales, caracterización de los estudiantes y capacidad y matrícula.

CAPÍTULO 3. IMPLEMENTACIÓN Y PRUEBAS

Durante el presente capitulo se realiza el diseño del modelo de datos físico. Se plantea la integración que existe entre el sistema propuesto y el ecosistema MAYA. Se verifica el cumplimiento de los requisitos funcionales mediante pruebas para evaluar la calidad del producto desarrollado.

3.1. Modelo de datos físico

Es un lenguaje utilizado para la descripción de una base de datos. Permite escribir el tipo de dato que incluye y la forma en que se relacionan, así como las restricciones de integridad y las operaciones de manipulación de la información, describe la representación lógica y física de los datos persistentes (30).

Se creó una base de datos con 22 tablas de las cuales 10 son nomencladores. Se utilizó el patrón de diseño llave subrogada, encargado de almacenar un valor numérico único para cada fila de la tabla actuando como una clave sustituta, de forma totalmente independiente a los datos. La notación utilizada para nombrar las tablas fue UpperCamelCase poniendo la primera letra de cada palabra en mayúsculas. La base de datos se encuentra en 3 forma normal, al ser todos los atributos dependientes funcionalmente solo de la clave primaria y no existir dependencias transitivas. El modelo perteneciente al diseño de la base de datos se encuentra en el Anexo 1.

La base de datos creada permite almacenar en el dispositivo móvil una copia de los datos asociados a los usuarios autenticados en la aplicación, la cual se obtiene de GERES mediante el estilo arquitectónico REST. Esto posibilita que los datos queden almacenados en el cliente que realizó la petición, lográndose no depender constantemente de las conexiones a través de las redes WiFi.

3.2. REST

Representational State Transfer (REST) es un estilo de arquitectura de software que se utiliza para describir cualquier interfaz entre sistemas que utilicen directamente HTTP para obtener o indicar la ejecución de operaciones sobre los datos en múltiples formatos. Cuenta con una arquitectura cliente servidor donde cada mensaje HTTP contiene toda la información necesaria para comprender la petición. Es posible cachear las peticiones de tal forma que una vez realizada la primera el resto puedan apoyarse en la cache si fuera

necesario. Sus servicios comparten una forma de invocación y métodos uniformes utilizando los métodos GET, POST, PUT y DELETE (31).

Estas peticiones se implementan a través de la biblioteca RESTrung. La interfaz principal envía peticiones y recibe respuestas serializadas a través de la clase RESTClient. Las invocaciones son asíncronas y son devueltos y enviados los datos mediante la utilización de las clases Request encargada de las peticiones y Response la que maneja las respuestas.

3.3. Sincronización

En informática, la sincronización de datos es el proceso de asegurarse de que dos o más ubicaciones contengan las mismas versiones. Si se agrega, modifica o elimina uno de estos datos en una ubicación, el proceso de sincronización agregará, modificará o eliminará el mismo dato en las otras ubicaciones (32). Esta permite mantener la misma versión de los datos en múltiples ubicaciones, generalmente bases de datos, directorios y aplicaciones en una computadora, dispositivos de almacenamiento extraíble y/o un dispositivo móvil como un celular.

Tipos

- Una vía (one-way): también denominada espejado, los datos son copiados solo desde una ubicación fuente hacia una o más ubicaciones destino, pero ninguno es copiado en el sentido inverso. Por lo tanto, las modificaciones hechas en destino no afectan a la fuente.
- Dos vías (two-way): los datos son copiados en ambos sentidos, manteniendo ambas ubicaciones sincronizadas una con la otra. Esta es la definida para utilizar en la solución propuesta. En un punto se tiene la aplicación móvil (BKDroid) y en el otro la aplicación web (GERES).

Se definen las siguientes características para el proceso de sincronización:

- Para lograr más eficiencia, se sincronizan sólo aquellos datos que han cambiado. Cada tabla de la base de datos contiene un campo denominado updatedAt de tipo timestamp, el cual es actualizado cada vez que se produce un cambio en la tupla a la cual hace referencia. Esto garantiza que persista el dato más resiente.
- Los datos son enviados y recibidos utilizando las redes WiFi. Dentro de la universidad la velocidad de conexión de estas redes es como promedio de 64mb/s.

- Se definen funciones para la detección de conflictos, por ejemplo, cuando hay algún dato que no está sincronizado correctamente (diferentes versiones de ambos lados). El usuario puede determinar a qué fuente (BKDroid o GERES) darle mayor prioridad para la actualización de los datos.
- Posibilita previsualizar cualquier cambio antes de que se haga.

Para la primera autenticación del usuario en el sistema es necesario que este se encuentre conectado a una red WiFi que tenga acceso a los servicios que brinda la UCI. Al ocurrir la primera autenticación del usuario se gurda una copia local de sus credenciales encriptadas utilizando el algoritmo MD5, de igual manera se crea una base de datos local en el dispositivo móvil que contiene solo los datos referentes al usuario autenticado. Es posible trabajar con dicha base de datos de modo offline, las modificaciones que en ella ocurran serán guardadas para posteriormente al encontrarse online ser sincronizadas con la base de datos de GERES.

3.4. Integración con otros sistemas

BKDroid consume servicios de otras aplicaciones del ecosistema MAYA (ALMA y GERES) empleando la arquitectura REST. Además, utiliza el protocolo SOAP que permite la comunicación con el servidor LDAP-UCI para autenticar a los usuarios del sistema. A continuación, se muestra una imagen que ilustra esta integración.

.

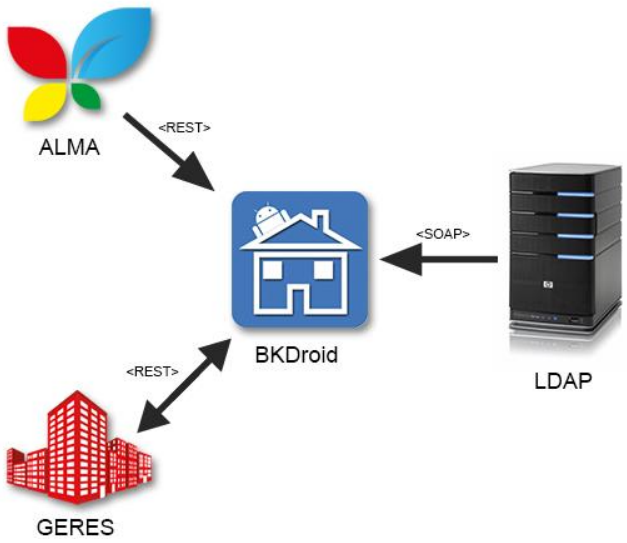


Diagrama de integración de BKDroid

La siguiente tabla muestra los servicios a consumir del API de la aplicación ALMA (https://alma.uci.cu/api/doc):

Nombre	**uri**	**Descripción**	**Parámetros Entrada**	**Salida (Retorno)**
http://alma.uci.cu/foto/{idExpediente}	Renderiza la foto, la petición es en forma de url. Esto posibilita que la carga de la información sea en paralelo.	idExpediente	La foto.	
Obtenercursoactual	/api/v1/curso/actual.json	Obtiene el curso actual que tiene activo el sistema.	-	Array con las claves: id fechaInicio fechaFin nombre
Obtenerpersonadadoidexpediente	/api/v1/personadadoidexpedientes/{idExpediente}.json	Obtiene una persona dado el idExpediente pasado por paráremo. Devuelve los datos generales de la persona, que puede existir o no en el curso actual.	Idexpediente	Array con las claves: idExpediente usuario pNombre sNombre pApellido sApellido solapin sexo direccion

				tipoPersona municipio provincia activo En el caso que sea profesor devuelve: cargo *En caso de no encontrar datos, retorna false.*
Obtenerfacultades	/api/v1/facultades.json	Obtiene todas las facultades activas	-	Array de array con las claves: id codigo nombre siglas
Obtenerapartamentos	/api/v1/apartamentos.json	Obtiene los apartamentos	-	Array de array con las claves: id numero edificio

				idEdificio *En caso de no encontrar datos, retorna false.*
Obtenerapartamentosdadoedificio	/api/v1/apartamentodadoedificios/{idEdificio}.json	Obtiene los apartamentos dado un edificio entrado por parámetro	idEdificio	Array de array con las claves: id numero edificio idEdificio *En caso de no encontrar datos, retorna false.*
Obteneredificios	/api/v1/edificios.json	Obtiene los edificios del sistema	-	Array de array con las claves: id numero cantAptos *En caso de no encontrar datos, retorna false.*

Obtenerapartamentosdadoedificio	/api/v1/apartamentosdadoedificios/{idEdificio}.json	Obtiene los apartamentos dado un Edificio entrado por parámetros	idEdificio	Array de array con las claves: id numero edificio idEdificio *En caso de no encontrar datos, retorna false.*

La siguiente tabla muestra los servicios a consumir del API de la aplicación GERES (https://geres.uci.cu/api/doc):

Nombre	uri	Descripción	Parámetros Entrada	Salida (Retorno)
ObtenerCuarteleriadadoIdExpedienteyCurso	/api/v1/cuarteleriasidexpedientes/{idExpediente}/cursos/{idCurso}.json	Obtiene las cuartelería de un estudiante dado su IdExpediente y el Curso dado por parámetro.	idExpediente idCurso	Array de array con las claves: evaluacion fecha observaciones id *Retorna false en*

				caso de no encontrar datos
ObtenerEvaluacionesFrecuentesdadoIdExpedienteyCurso	/api/v1/evaluacionesfrecuentesidexpedientes/{idExpediente}/cursos/{idCurso}.json	Obtiene todas las evaluaciones frecuentes de los estudiantes dado su idExpediente y el curso.	idExpediente idCurso	Array de array con las claves: evaluacion fecha observaciones id *Retorna false en caso de no encontrar datos*
ObtenerEvaluacionesIntegralesdadoIdExpedienteyCurso	/api/v1/evaluacionesintegralesidexpedientes/{idExpediente}/cursos/{idCurso}.json	Obtiene todas las evaluaciones integrales de los estudiantes dado su idExpediente y el curso.	idExpediente idCurso	Array de array con las claves: evaluacion fecha observaciones id *Retorna false en caso de no encontrar datos*
ObtenerIndisciplinasdadoIdExpedienteyCurso	/api/v1/indisciplinasidexpedientes/{idExpediente}/cursos/{idCurso}.json	Obtiene todas las indisciplinas los estudiantes dado su	idExpediente idCurso	Array de array con las claves:

		idExpediente y el curso.		evaluacion fecha observaciones Id *Retorna false en caso de no encontrar datos*
ObtenerEdificiosAsociadosaInstructoradadoIdExpedienteyCurso	/api/v1/edificiosasociadosinstructorasdadoidexpediente/{idExpediente}/cursos/{idCurso}.json	Obtiene los edificios asociados a una instructora dado su idExpediente y el curso entrados por parámetros.	idExpediente idCurso	<u>Array de array con las claves:</u> id edificioid numeroEdificio facultad cantidadAptos *Retorna false en caso de no encontrar datos*

3.5. Pruebas

Constituyen el instrumento adecuado para determinar el estado de la calidad de un producto de software. En este proceso se ejecutan pruebas dirigidas a componentes del software o al sistema en su totalidad, con el objetivo de medir el grado en que la aplicación cumple con los requerimientos. Son diseñados casos de pruebas especificados de forma estructurada mediante técnicas, que sistemáticamente identifican diferentes clases de errores, realizándose con la menor cantidad de tiempo y esfuerzo (33).

3.5.1. Niveles de pruebas

Pruebas unitarias

Consisten en comprobaciones (manuales o automatizadas) desarrolladas por los programadores. Las cuales se realizan para verificar que el código correspondiente a un módulo concreto se comporta de manera esperada (34). Las pruebas unitarias proporcionan beneficios tales como:

- Brindan al programador una inmediata retroalimentación de cómo está realizado su trabajo.
- El programador puede realizar cambios de forma segura respaldada por efectivos casos de pruebas.
- Permite saber si una determinada funcionalidad se puede agregar al sistema existente sin alterar el funcionamiento actual del mismo.

Pruebas de integración

Aun cuando los módulos de un programa funcionen bien por separado es necesario probarlos conjuntamente: un módulo puede tener un efecto adverso o inadvertido sobre otro módulo; las subfunciones, cuando se combinan, pueden no producir la función principal deseada; la imprecisión aceptada individuamente puede crecer hasta niveles inaceptables al combinar los módulos; los datos pueden perderse o malinterpretarse entre interfaces. Por lo tanto, es necesario probar el software ensamblando todos los módulos probados previamente (35).

Pruebas de sistema

Buscan discrepancias entre el programa y el objetivo o requerimiento, enfocándose en los errores hechos durante la transición del proceso al diseñar la especificación funcional. Esto hace a las pruebas de sistema un proceso vital de pruebas, ya que, en términos del producto, numero de errores hechos y severidad de estos errores es considera un paso en el ciclo de desarrollo generalmente propenso a la mayoría de los errores (36)

3.5.1. Tipos de pruebas

Pruebas de caja negra

Se realizan en la interfaz del sistema, obviando el comportamiento interno y la estructura del programa, pretenden demostrar que las funciones del software son operativas, que las entradas de datos se aceptan de forma correcta, así como la salida de los mismos y que la integridad de la información externa se mantiene. Los errores que pretende detectar son dados en base a funciones incorrectas, errores de interfaz, de rendimiento, de inicialización o terminación (37).

Pretenden encontrar estos tipos de errores:

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

Como resultado se realizó un caso de prueba por cada requisito (tomando los diseños basados en requisitos), en los cuales se detectaron varias no conformidades en revisiones internas realizadas por el equipo de desarrollo, las cuales fueron resueltas.

Prueba de caja blanca

Las pruebas de caja blanca (también conocidas como pruebas de caja de cristal) se centran en los detalles procedimentales del software, por lo que su diseño está fuertemente ligado al código fuente. Aunque las pruebas de caja blanca son aplicables a varios niveles (unidad, integración y sistema), habitualmente se aplican a las unidades de software. Su objetivo es comprobar los flujos de ejecución dentro de cada unidad (37).

Estas se llevan a cabo en primer lugar, sobre un módulo concreto, para luego realizar las de caja negra sobre varios subsistemas. Mediante los métodos de prueba de caja blanca, se puede obtener casos de prueba que:

- Garanticen que se ejercita por lo menos una vez todos los cambios independientemente de cada módulo. Ejerciten todas las decisiones lógicas en sus vertiente verdadera y falsa.
- Ejecuten todos los ciclos en sus límites.

- Ejerciten las estructuras internas de datos para asegurar su validez.

Las principales técnicas de diseño son:

- Pruebas de la estructura de control.
- Pruebas de caminos básicos.

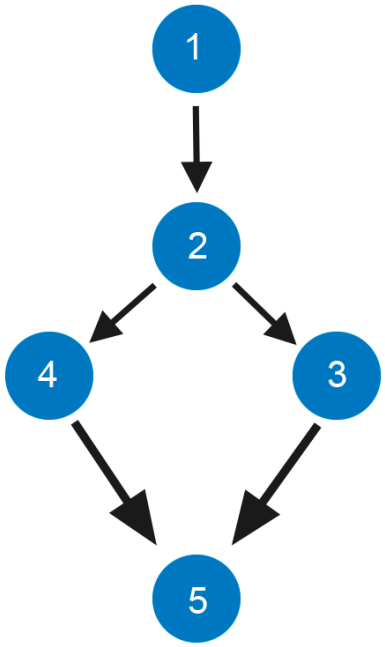
La prueba de caja blanca empleada en la solución fue la prueba del camino básico, a partir del cálculo de la complejidad ciclomática del algoritmo a ser analizado. Para realizarla se deben enumerar las sentencias de código y a partir de ahí elaborar el grafo de flujo de esta funcionalidad.

Para la misma se definieron una serie de pasos a seguir:

1) Notación del grafo de flujo: usando el código como base se realiza la representación del grafo de flujo, mediante una sencilla notación. Cada construcción estructurada tiene su correspondiente símbolo.
    - Nodo: a cada circulo denominado nodo, representa una o más sentencias procedimentales.
    - Arista: las flechas del grafo de flujo, denominadas aristas, representan el flujo de control y son análogas a las flechas del diagrama de flujo.
    - Región: las áreas delimitadas por aristas y nodos se denominan regiones.
2) Complejidad ciclomática: es una métrica que proporciona una medición cuantitativa de la complejidad lógica de un programa. El valor calculado define el número de caminos independientes del conjunto básico de un programa. Esto indica el límite superior para el número de pruebas que se deben realizar, para asegurar que se ejecuta cada sentencia al menos una vez. Se utilizó la siguiente forma: V (G), de un grafo de flujo G se define como: V (G)=A-N+2, donde A es el número de aristas del grafo de flujo y N es el número de nodos.
3) Determinar un conjunto básico de caminos linealmente independientes: el valor de V (G) es el número de caminos linealmente independientes de la estructura de control del programa.
4) Obtención de casos de prueba: se realizan los casos de pruebas que forzarán la ejecución de cada camino del conjunto básico.

La imagen del Anexo 2 muestra el código que se tomó como ejemplo para realizar el procedimiento anterior descrito para la técnica de camino básico, correspondiente al método onClick(View v) de la clase controladora EstudiantesActivity.

En la siguiente imagen se muestra el grafo de flujo asociado al requisito funcional seleccionado:



Grafo de flujo utilizando la técnica de caminos básicos

Luego se calcula la complejidad ciclomática:

V (G) = (A - N) + 2, V (G) = (5 - 5) + 2, V (G) = 2

Siendo A la cantidad total de aristas del grafo y N la cantidad de nodos.

A partir de los resultados obtenidos en cada uno, se determina que la complejidad ciclomática del código analizado es 2, que a su vez es el número de caminos posibles a circular el flujo y el límite superior de casos de prueba que se le pueden aplicar a dicho código.

En la siguiente tabla se muestra los caminos básicos por donde puede circular el flujo.

Números	Caminos básicos
1	1, 2, 3, 5
2	1, 2, 4, 5

Posibles caminos básicos

Cada camino independiente es un caso de prueba a realizar, de forma que los datos introducidos provoquen que se visiten las sentencias vinculadas a cada nodo del camino.

3.5.2. Diseños de casos de prueba

En la siguiente tabla se muestra el diseño de caso de prueba para para el módulo de Capacidad y Matrícula los requisitos que fueron descritos en el epígrafe 2.2.2. El resto de los casos de pruebas se encuentran en el expediente de proyecto.

Escenario	Descripción	Respuesta del sistema	Flujo central
RF27: Listar edificios	Muestra un listado con los edificios asociados a un usuario.	Listado de edificios	Una vez que el usuario se autentique en el sistema se muestra el listado de edificios asociados indicando el número del mismo, la facultad que pertenece, la cantidad de pasos de escaleras, el estado constructivo, el sexo y el área donde se encuentra.
RF28: Listar apartamentos	Muestra un listado con los apartamentos asociados a un edificio.	Listado de apartamentos	Al presionar sobre un edificio se muestra un listado con los apartamentos del edificio en cuestión.
RF29: Listar estudiantes	Muestra un listado con los estudiantes asociados a un apartamento.	Listado de estudiantes	Al seleccionar un apartamento se muestra el listado de los estudiantes que pertenecen a él.
RF31: Reporte de listado de	Genera un listado de todos los estudiantes residentes en el edificio.	Genera en formato PDF un listado con los estudiantes pertenecientes al edificio.	Al seleccionar un edificio conjuntamente con el listado de apartamentos se muestra

estudiantes por edificio			un botón en la parte inferior derecha que al presionarlo de abre una nueva vista con los diferentes reportes asociados a un edificio. Luego de esto se selecciona la opción “listado de estudiantes” la cual genera un reporte en formato PDF con el listado de estudiantes del edificio.
RF32: Reporte de listado de estudiantes por apartamento	Genera un listado de todos los estudiantes residentes en un apartamento	Genera en formato PDF un listado con los estudiantes pertenecientes al apartamento.	Al seleccionar un apartamento conjuntamente con el listado de estudiantes se muestra un botón en la parte inferior derecha que al presionarlo se muestran dos nuevas opciones. Luego de esto se selecciona la de la izquierda que abre una nueva vista con los diferentes reportes asociados a un apartamento donde se selecciona la opción “listado de estudiantes” que genera un reporte en formato PDF con el listado de estudiantes del apartamento.

Tabla de casos de pruebas

3.5.3. Implementación de casos de prueba

Durante el desarrollo de las funcionalidades se realizaron pruebas unitarias al código para ir comprobando el funcionamiento del software, estas fueron realizadas por el desarrollador. No se planificaron ni se registraron sus resultados, fueron haciéndose a medida que la solución se desarrollaba.

A través del método de caja negra, y apoyados en el diseño de los Casos de Pruebas explicados en el epígrafe anterior, se realizaron cuatro iteraciones de pruebas internas pertenecientes al nivel del sistema. Dichas pruebas fueron realizadas con el objetivo de detectar y corregir errores que impidieran el correcto funcionamiento de la solución. A continuación, se presentan los resultados arrojados durante las diferentes pruebas aplicadas:

- Pruebas internas: Realizadas por el equipo de desarrollo con el fin de entregar un producto con la menor cantidad de errores posibles. Se centraron en el cumplimiento de las funcionalidades descritas en el listado de requisitos y de casos de uso del sistema. Además, se revisó toda la documentación.
- Pruebas cruzadas: Realizadas también por el equipo de desarrollo al sistema con el fin de encontrar la mayor cantidad de errores posibles en términos de validaciones, pautas definidas por arquitectura de información, formato de los campos, entre otras.

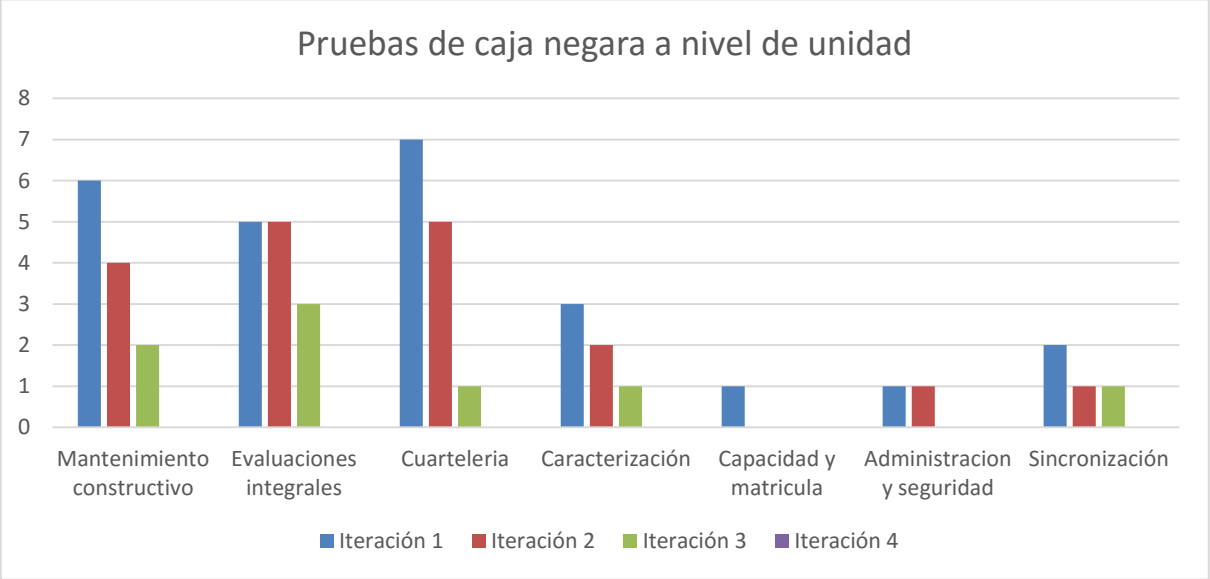


Diagrama de pruebas de caja negra a nivel de unidad

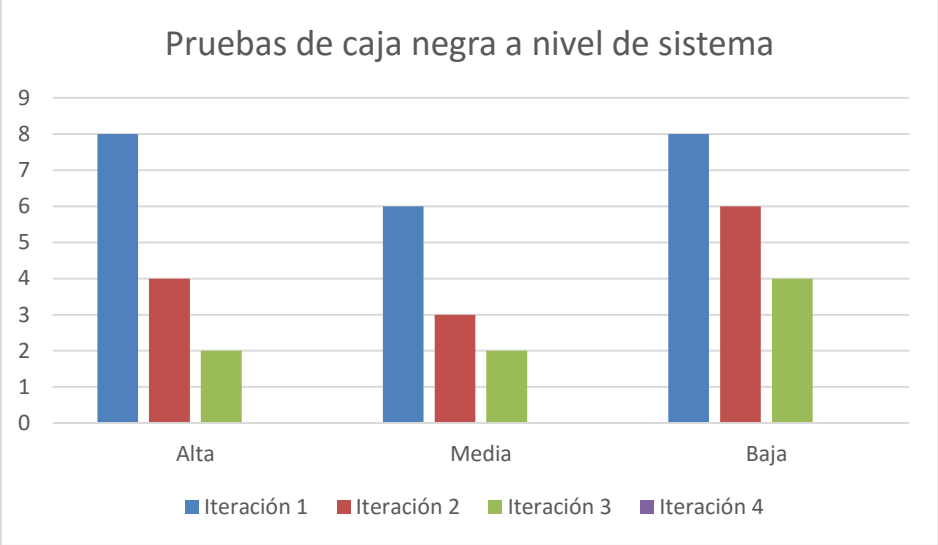


Diagrama de pruebas de caja negra a nivel de sistema

Las no conformidades más comunes encontradas estuvieron relacionadas con el mal funcionamiento de las interfaces, comportamiento de objetos de forma no deseada y mensajes de información no personalizados.

3.6. Conclusiones parciales

Mediante el diseño de la base de datos se logró el registro de los datos de la aplicación garantizándose la persistencia de los mismos. La implementación de la arquitectura REST para el intercambio de información permitió la integración de la solución propuesta con el ecosistema MAYA. Los diferentes tipos de pruebas realizadas arrojaron que el sistema funciona correctamente y de acuerdo a los requisitos funcionales establecidos por los desarrolladores.

CONCLUSIONES

Una vez finalizada la presente investigación se puede arribar a las siguientes conclusiones:

- El estudio del estado del arte realizado demostró que no se conoce un sistema basado en tecnologías móviles para la gestión de los procesos asociados a la Residencia Estudiantil que respondan al problema planteado.
- El análisis de las herramientas y tecnologías seleccionadas permitió definir el marco tecnológico para el desarrollo de la solución en apego a la soberanía tecnológica.
- La implementación de la arquitectura REST para el intercambio de información permitió la integración de la solución propuesta con el ecosistema MAYA.
- El desarrollo de la aplicación contribuyó a aumentar la disponibilidad y accesibilidad de los datos asociados a la residencia estudiantil.

RECOMENDACIONES

Teniendo en cuenta el estudio realizado durante todo el proceso de desarrollo de la presente investigación y en aras de enriquecer la solución el autor recomienda:

- Desarrollar los módulos actividades educativas e incidencias disciplinarias.
- Ampliar el alcance del sistema para las áreas residenciales de especialistas y profesores.

BIBLIOGRAFÍA

1. UIT: Comprometida para conectar el mundo. [En línea] [Citado el: 3 de Junio de 2016.] http://www.itu.int/es/Pages/default.aspx.

2. *Gestión de la Calidad y del Medio Ambiente en Instituciones de Educación Superior mediante Integración de ISO 9001 e ISO 14001.* Diego A. Tlapa, Jorge Limon y Yolanda A. Báez. 2009. ISSN 07185006.

3. *Aspectos clave de la integración de sistemas de gestión.* Puente, Jesus Abad. 2009.

4. Redondo, Yoe Reyes. *Sistema para la gestión de la información asociada a la residencia estudiantil.* La Habana, Cuba : 11na Peña Tencnológica, Universidad de las Ciencias Informáticas. Memoria de eventos., 2016.

5. SIG UCI. [En línea] [Citado el: 2 de Junio de 2016.] http://siguci.uci.cu/.

6. Cardoso, Victor Gonzalez. *Documento visión del proyecto MAYA.* La Habana : s.n., 2014.

7. Real Academia Española. Diccionariod e la Lengua Española. [En línea] [Citado el: 4 de Junio de 2016.] http://dle.rae.es/?id=ACBTb7y|ACDpxn0.

8. Administracion condominios, propiedad horizontal, conjuntos residenciales y apartamentos. [En línea] [Citado el: 25 de Mayo de 2016.] http://www.miscondominios.com/.

9. Market Share. [En línea] [Citado el: 23 de Mayo de 2016.] http://netmarketshare.com/.

10. The Android Source Code | Android Developers. [En línea] [Citado el: 25 de Mayo de 2016.] http://web.archive.org/web/20140702115900/https://source.android.com/source/index.html.

11. IntelliJ IDEA the Java IDE. *IntelliJ IDEA the Java IDE.* [En línea] [Citado el: 29 de Febrero de 2016.] https://www.jetbrains.com/idea/.

12. Android Studio Overview | Android Developers. *Android Studio Overview | Android Developers.* [En línea] [Citado el: 29 de Febrero de 2016.] http://developer.android.com/intl/es/tools/studio/index.html.

13. Software Design Tools for Agile Teams, with UML, BPMN and More. *Software Design Tools for Agile Teams, with UML, BPMN and More.* [En línea] [Citado el: 29 de Febrero de 2016.] http://www.visual-paradigm.com/.

14. Java. *Java.* [En línea] [Citado el: 18 de Abril de 2016.] https://www.java.com/es/.

15. Extensible Markup Language (XML). *Extensible Markup Language (XML).* [En línea] [Citado el: 29 de Febrero de 2016.] https://www.w3.org/XML/.

16. SQLite. *AQLite.* [En línea] [Citado el: 29 de Febrero de 2016.] https://www.sqlite.org/.

17. Open Source by greenrobot | EventBus, greenDAO, and greenrobot Essentials: Open Source by Heart. *Open Source by greenrobot | EventBus, greenDAO, and greenrobot Essentials: Open Source by Heart.* [En línea] [Citado el: 29 de Febrero de 2016.] http://greenrobot.org/greendao/.

18. Gradle. *Gradle.* [En línea] [Citado el: 8 de Marzo de 2016.] http://gradle.org/.

19. Unefied Modeling Language. *Unefied Modeling Language.* [En línea] [Citado el: 8 de Marzo de 2016.] http://www.uml.org/.

20. Pressman, Reger S. *Ingeniería de Software, un enfoque práctico. Quinta edición.* 2002.

21. Extreme Programming. *Extreme Programming.* [En línea] [Citado el: 8 de Marzo de 2016.] http://www.extremeprogramming.org/.

22. Guerra, César Arturo. SG Buzz. [En línea] Octubre de 2007. [Citado el: 2 de Junio de 2016.] https://sg.com.mx/revista/17/obtencion-requerimientos-tecnicas-y-estrategia.

23. DECSAI Departamento de Ciencias de la Computación e IA Universidad de Granada. Especificación de Requerimientos. [En línea] [Citado el: 1 de Junio de 2016.] http://elvex.ugr.es/idbis/db/docs/design/2-requirements.pdf.

24. Londoño, Jorge Hernán Abad. Scrum Community. [En línea] 23 de Agosto de 2013. [Citado el: 3 de Junio de 2016.] https://www.scrumalliance.org/community/articles/2013/august/caracteristicas-de-una-buena-historia-de-usuario.

25. Universidad Docente de Ingenieria de Software. Master de Ingeniería de Software. [En línea] [Citado el: 25 de Mayo de 2016.] http://is.ls.fi.upm.es/docencia/masterTI/ARS/docs/Manual_M2C1U11.pdf.

26. Carlos Reynos, Nicolás Kicillof. *Estilos y Patrones en la Estrategia de Arquitectura de Microsoft.* 2004.

27. MVC Android. *MVC Android.* [En línea] [Citado el: 19 de Abril de 2016.] http://androideity.com/2012/05/10/la-importancia-del-mvc-en-android/.

28. Departamento de Lenguajes y Ciencias de la Computación. Universidad de Málaga. *Apuntes para la Asignatura Informática. Tema 10. Diseño Modular.*

29. Grosso, Andrés. Patrones GRASP | Prácticas de Software. [En línea] 21 de Marzo de 2011. [Citado el: 21 de Abril de 2016.] http://www.practicasdesoftware.com.ar/2011/03/patrones-grasp/.

30. Date, C J. *Introducion a los sistemas de Bases de Datos.* 2009.

31. Caules, Cecilio Álvarez. ArquitecturaJava. *ArquitecturaJava.* [En línea] 14 de Junio de 2013. [Citado el: 8 de Marzo de 2016.] http://www.arquitecturajava.com/servicios-rest/.

32. Alegsa, Leandro. ALEGSA. [En línea] 3 de Noviembre de 2008. [Citado el: 11 de Mayo de 2016.] http://www.alegsa.com.ar/Dic/sincronizar%20archivos.php.

33. Gestión de Calidad y Pruebas de Software. Pruebas de Software. [En línea] [Citado el: 2 de Marzo de 2016.] http://www.pruebasdesoftware.com/laspruebasdesoftware.htm.

34. B., Ing. Alexander Oré. CalidadySoftware. [En línea] 2009. [Citado el: 27 de Abril de 2016.] http://www.calidadysoftware.com/testing/pruebas_unitarias1.php.

35. Rosales, Marlene. Prezi. [En línea] 27 de MAyo de 2015. [Citado el: 17 de Mayo de 2016.] https://prezi.com/ewxpjagr08xu/ciclo-de-pruebas-integrales/.

36. Prezi. [En línea] 31 de Marzo de 2014. [Citado el: 15 de Abril de 2016.] https://prezi.com/dvkiw1wwle5b/aplicacion-de-las-pruebas-del-sistema/.

37. Carabal'i, Mauricio. prezi. [En línea] 21 de Septiembre de 2013. [Citado el: 2 de Junio de 2016.] https://prezi.com/sjwfwmix7slk/pruebas-de-caja-negra-y-caja-blanca/.

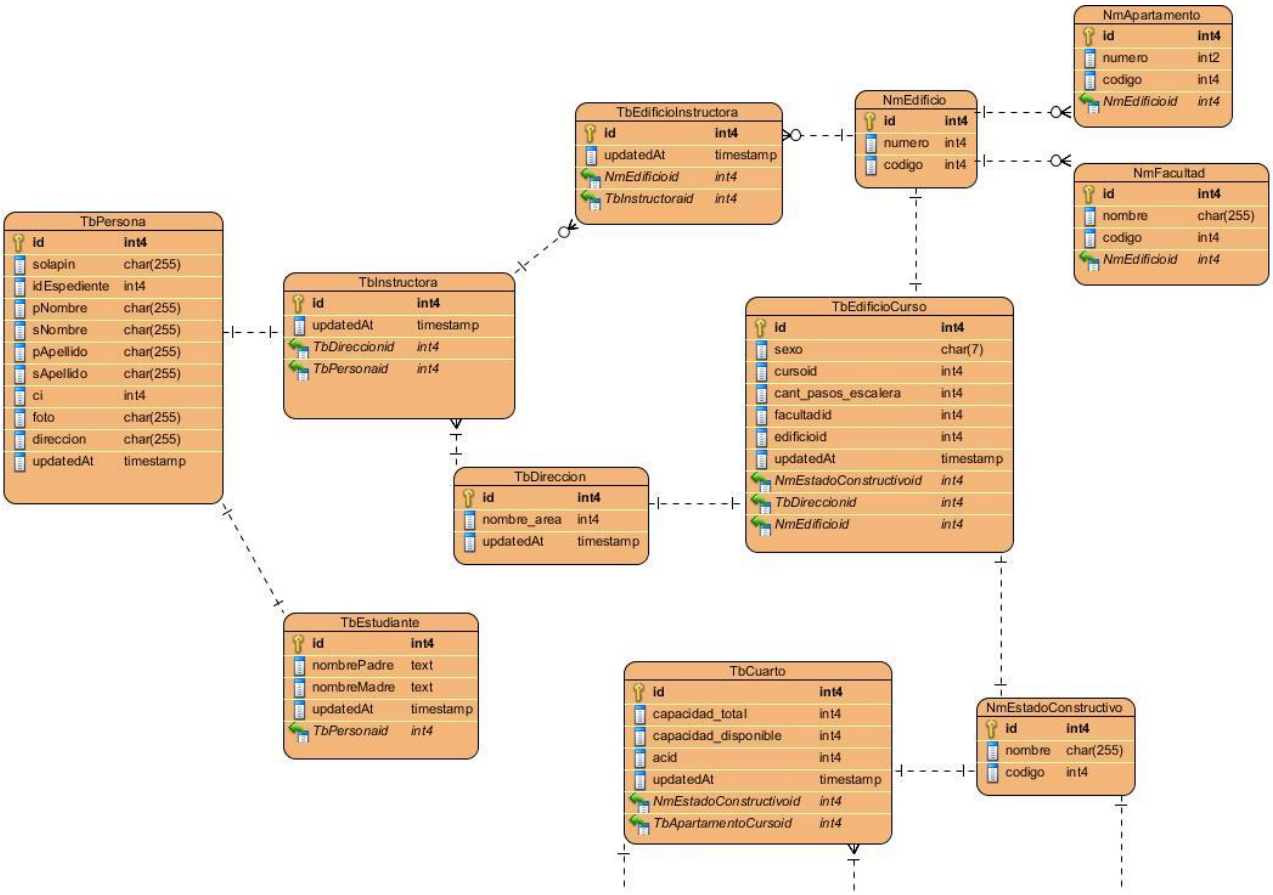
38. *Modelos de Integracion de Sistemas en Empresas Venesolanas.* Irina Titaeva, Luis E. Mendoza, María Pérez. 2012.

39. RDNS. *Informe central: Integracion de Sistemas de seguridad.* 2012.

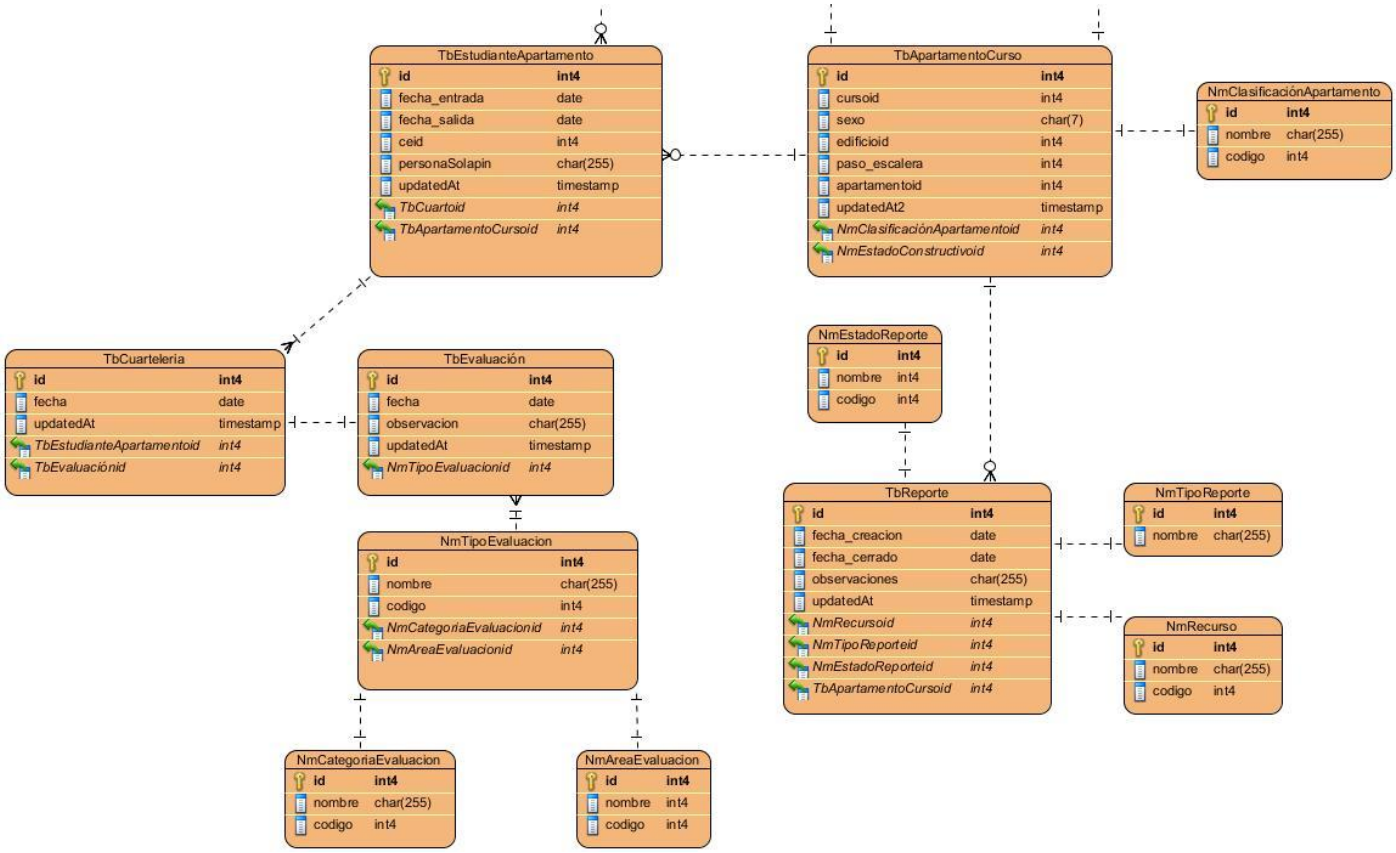
ANEXOS

Anexo 1: Tabla del modelo de datos físicos.

Fragmento 1 de 2.



Fragmento 2 de 2.



Anexo 2: Código de onClick de la clase controladora EstudiantesActivity.

```
public void onClick(View v) {

    if(boton_img){
        incidencias_constructivas.setVisibility(View.GONE);
        generar_reporte_apartamento.setVisibility(View.GONE);
        boton_img = false;

    }else{

        incidencias_constructivas.setVisibility(View.VISIBLE);
        generar_reporte_apartamento.setVisibility(View.VISIBLE);
        boton_img = true;
    }
}
```

Anexo 3: Encuesta al personal de la residencia.

Se realizó encuestas al personal de la residencia entre los días 20 y 25 de febrero de 2016 en los locales de las instructoras ubicados en los apartamentos 1403, 15304, 3102, 7104, 56302, 88102 y 84201, las preguntas realizadas fueron:

1. ¿Cómo es que se planifica la cuartelería?
2. ¿Qué ocurre cuando un estudiante quiere cambiar su cuartelería?
3. ¿Cómo es que se evalúa la cuartelería?
4. ¿Cuáles son las incidencias constructivas en la residencia?
5. ¿Cómo es que se manejan las incidencias constructivas en la residencia?
6. ¿Cómo es que ocurre el proceso de evaluación de un estudiante?
7. ¿Cuáles son las evaluaciones que se le asignan a los estudiantes?

Se entrevistó un total de 16 instructoras, un director de residencia y el director general de residencia de la UCI.