Aplicación para el reporte de incidencia luego del despliegue de la Distribución Cubana GNU/Linux Nova

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

Autor:

Julio César Gallardo Concepción.

Tutor(es):

Ing. Abel Ochoa Izquierdo.

MSc. Nurileidis Almeida Cintra.

Facultad 1

La Habana, diciembre de 2022.

Año 64 de la Revolución.

DECLARACIÓN DE AUTORÍA

El autor del trabajo de diploma con título "**Aplicación móvil para el reporte de incidencias luego del despliegue de la Distribución Cubana de GNU/Linux Nova**", concede a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la investigación, con carácter exclusivo. De forma similar se declara como único autor de su contenido. Para que así conste firma la presente a los **1** días del mes de **diciembre** del año **2022.**

Julio César Gallardo Concepción

_____

Firma del Autor

Ing. Abel Ochoa Izquierdo

_____

Firma del Tutor

MSc. Nurileidis Almeida Cintra

_____

Firma del Tutor

DATOS DE CONTACTO

Autor:

Julio César Gallardo Concepción

Universidad de las Ciencias Informáticas (UCI)

e-mail: jcgallardo@estudiantes.uci.cu

Tutor(es):

MSc. Nurileidis Almeida Cintra.

Universidad de las Ciencias Informáticas (UCI)

e-mail: nurileidisac@uci.cu

Ing. Abel Ochoa Izquierdo.

Universidad de las Ciencias Informáticas (UCI)

e-mail: aoizquierdo@uci.cu

Agradecimientos:

A mis padres que me han dado tanto apoyo, consejos y han hecho posible que pueda cumplir mis metas.

A mi hermano Lucho y a mi primo José por ser un gran el ejemplo a seguir

A mi familia en general por la oportunidad de compartir todos mis logros.

A mi Emilio gracias por la compañía, el amor y la comprensión por apoyarme en todo y ser capaz de sacarme una sonrisa en el peor de los momentos.

A mis tutores y profesores que han sido capaz de no dejarme caer y ayudarme a superar este proceso tan grande.

A mis Amigos, Heidy Horta Silveira, Fernando Hernández López, Javier Anias Santos, Tania Figueredo Guzmán Yesica Rodríguez Izquierdo, Jairo llano Tejera, Claudia Fonseca y todos aquello que han hecho posible que no me rindiese en estos años y fuesen un impulso a seguir.

A los amigos de siempre, a los amigos UCI, a los amigos de la vida, que siguieron página a página esta investigación desde un café, una terraza, una fiesta, una llamada o una reunión improvisada.

¡A todos Gracias por ser parte de mi familia UCI!

Dedicatoria:

Dedicados a todos esos sueños que en ocasiones dejamos escapar.

Gracias Mama, Papa y Lucho por permitir que pueda alcanzar los míos.

A ustedes y a mi tita Elena que me mira desde el cielo, es este logro, este sueño, esta meta llamada:

Ingeniero en Ciencias Informáticas.

RESUMEN

Cuba promueve, como parte de su programa de informatización, no apostar por sistemas operativos privativos como un camino viable para el desarrollo tecnológico, sino apostar por sistemas que a la vez conduzcan a la independencia en este mismo ámbito. Una de las formas de alcanzar la soberanía tecnológica es mediante la migración a software libre. Para lograr el éxito en este proceso se utiliza Nova, sistema operativo basado en GNU/LINUX desarrollado por cubanos y para los cubanos. Luego de finalizado el proceso de migración, se detecta la ineficiencia de los reportes de incidencias y errores de los usuarios a la hora de hacer uso del sistema, es de gran importancia recopilar esta información para próximas mejoras del sistema operativo. La presente investigación tiene como objetivo desarrollar una aplicación para dispositivos móviles con sistema operativo Android, cuyo objetivo es registrar y llevar el control de los problemas detectados por los usuarios, luego del despliegue del sistema operativo Nova. Durante el proceso de desarrollo de la propuesta de solución se utilizó como metodología de desarrollo de software AUP en su variante UCI, el lenguaje de programación Java y como entorno de desarrollo integrado Android Studio. Como resultado se obtuvo una aplicación móvil titulada: “Nova Incidencias” que permite al usuario manifestar su experiencia a través de reportes, logrando establecer una comunicación directa con el equipo de desarrollo, lo que permite contar con la experiencia de usuario para futuras mejoras en las versiones del sistema operativo Nova.

PALABRAS CLAVE

Android, Aplicación Móvil, Incidencia, Migración, Nova

ABSTRACT

Cuba promotes, as part of its informatization program, not to bet on proprietary operating systems as a viable path for technological development, but to bet on systems that at the same time lead to independence in this same field. One of the ways to achieve technological sovereignty is through migration to free software. To achieve success in this process, Nova is used, an operating system based on GNU/LINUX developed by Cubans and for Cubans. After completing the migration process, the inefficiency of the reports of incidents and errors of users when using the system is detected, it is of great importance to collect this information for future improvements of the operating system. The objective of this research is to develop an application for mobile devices with Android operating system, which aims to record and keep track of the problems detected by users, after the deployment of the Nova operating system. During the development process of the proposed solution, the software development methodology used was AUP in its UCI variant, the Java programming language and Android Studio as the integrated development environment. As a result, a mobile application entitled: "Nova Incidences" was obtained, which allows the user to express their experience through reports, establishing direct communication with the development team, which allows having the user experience for future improvements in the versions of the Nova operating system.

KEYWORDS

Android, Mobile Apps, Incidence, Migration, Nova.

TABLA DE CONTENIDOS

INTRODUCCIÓN 1
CAPÍTULO I: FUNDAMENTOS TEÓRICO-METODOLÓGICOS SOBRE EL OBJETO DE ESTUDIO.....7
1.1 Conceptos asociados al dominio del problema 7
1.1.1 GNU/Linux Nova: 7
1.1.2 Migración a Software Libre GNU/Linux Nova en Cuba: 7
1.1.3 Evento no deseado o incidencia: 9
1.1.4 Reporte de incidencia: 10
1.1.5 Aplicación móvil: 11
1.1.5.1 Tipos de aplicaciones móviles:.....12
1.1.6 Sistema operativo móvil. 13
1.1.6.1Sistema operativo Android: 13
I.2 Estudio de aplicaciones referente al reporte de incidencias. 13
1.2.1 Aplicaciones homólogas: 14
1.2.1.1 Internacionales: 14
1.2.1.2 Nacionales: 15
1.3 Metodología, lenguaje, herramientas y entorno de desarrollo 17
1.3.1 Metodología de desarrollo de software AUP-UCI 17
1.3.2 Lenguaje modelado. 19
UML v2.5:.....19
Visual Paradigm v15.1: 19
1.3.3 Lenguaje de programación. 20

Java: 20
1.3.4 Entorno de desarrollo integrado.....20
Android Studio v4.1.2:.....21
SDK:.....21
1.3.5 Herramienta para el diseño de prototipo de interfaz 22
Balsamiq Mockups v3.5 22
1.3.6 Gestor de base de datos.....22
SQLite v3.6 22
1.3.7 Herramienta de pruebas. 23
JUnit v4.12 23
Conclusiones del capítulo.....24
CAPÍTULO II: ANÁLISIS Y DISEÑO DE LA PROPUESTA DE SOLUCIÓN AL PROBLEMA CIENTÍFICO 25
2.1 Modelo Conceptual - Propuesta de solución - Levantamiento de Requisitos.....25
2.1.1 Modelo Conceptual: 25
2.1.2 Propuesta de solución: 26
2.1.2 Captura de Requisito: 27
2.1.2.1 Requisitos Funcionales: 27
2.1.2.2 Requisitos no funcionales: 32
2.2 Historias de Usuario 33
2.3 Modelo del Diseño.....36
2.3.1Descripción de la arquitectura Modelo Vista - Presentador 36
2.3.2 Diagrama de clases de diseño:.....38
2.3.3 Diagramas de secuencia 39

2.3.4 Patrones de diseño: 41

2.3.2.1 Descripción de los patrones de Principios Generales para Asignar Responsabilidades “Patrones GRASP”: 41

2.3.2.2 Descripción de los patrones de diseño Gang of Four “Patrones GOF”: 42

2.3.5 Modelo de Datos: 43

Conclusiones del capítulo 44

CAPÍTULO III: VALIDACIÓN DE LA PROPUESTA DE SOLUCIÓN AL PROBLEMA CIENÍIFICO 45

3.1 Implementación 45

3.1.1 Plan de Iteración: 45

3.1.2 Diagrama de componente 47

3.1.3Diagrama de despliegue: 47

3.1.4 Estándares de codificación 48

3.2 Pruebas al Sistema 49

3.2.1 Pruebas Funcionales: 49

3.2.3 Pruebas Unitarias: 51

3.2.2 Pruebas de Aceptación: 52

3.3 Resultados de las Pruebas 55

Conclusiones del capítulo 57

CONCLUSIONES FINALES 58

RECOMENDACIONES 59

REFERENCIAS BIBLIOGRÁFICAS 60

ANEXOS: 67

Anexo 1: Entrevista realizada para identificar las deficiencias en el proceso de reporte de incidencias por parte del usuario en el uso del sistema operativo Nova 67

Anexo 2: Historias de Usuarios: 68

Anexo 3: Diagramas de Clases de diseño:.....75

Anexo 4: Diagramas de Secuencia: 77

Anexo 5: Código empleado en prueba unitaria:.....77

Anexo 6: Prueba de funcionalidad.....79

Anexo 7: Carta de aceptación 81

ÍNDICE DE ILUSTRACIONES

Ilustración 1:Actividades generales del proceso de migración (Villazón, 2015).....9

Ilustración 2:Modelo conceptual (Elaboración propia) 25

Ilustración 3: Componentes de la Historia de Usuario(Menzinsky et al. 2022) 34

Ilustración 4: Diagrama de paquete de la arquitectura Modelo-Vista-Presentador (Sánchez 2015).....38

Ilustración 5:Diagrama de clases de diseño: Adicionar Incidencia (elaboración propia).....39

Ilustración 6:Diagrama de secuencia: Añadir incidencia (elaboración propia) 40

Ilustración 7:Diagrama entidad relación (elaboración propia) 43

Ilustración 8: Diagrama de componente del sistema (elaboración propia).....47

Ilustración 9:Diagrama de despliegue.....48

Ilustración 10:Resultado de la prueba JUnit de la clase RegistrarUsuariofragment.java (elaboración propia).....52

Ilustración 11: Resultado de la prueba JUnit de la clase AdicionarIncidenciafragment.java (elaboración propia).....52

Ilustración 12: Grafo No conformidad (elaboración propia).....57

Ilustración 13:Diagrama de clases de diseño: Estadísticas Incidencias (elaboración propia).75

Ilustración 14:Diagrama de clases de diseño: Buscar Incidencia (elaboración propia.....75

Ilustración 15:Diagrama de clases de diseño: Registrar usuario (elaboración propia) 76

Ilustración 16:Diagrama de clases de diseño: Global (elaboración propia) 76

Ilustración 17: DS Registrar Usuario.....77

ÍNDICE DE TABLAS

Tabla 1: Aplicaciones homologas al sistema 17

Tabla 2: Requisitos Funcionales. 32

Tabla 3: Requisitos no funcionales 33

Tabla 4: HU Adicionar Incidencias 36

Tabla 5: Plan de iteración (elaboración propia) 46

Tabla 6:Caso de prueba de funcionalidad a la historia de usuario añadir incidencia (elaboración propia) 51

Tabla 7: Caso de prueba de aceptación, historia de usuario eliminar incidencia (elaboración propia) 54

Tabla 8:Caso de prueba de aceptación, historia de usuario listar incidencia (elaboración propia) 55

Tabla 9: HU Registrar Usuario 69

Tabla 10: HU Autenticar Usuario 70

Tabla 11: Mostrar detalle de la incidencia 72

Tabla 12:Listar incidencias 73

Tabla 13:Eliminar Incidencia 74

Tabla 14: Caso de prueba de funcionalidad a la historia de usuario cambiar estado de incidencia (elaboración propia) 79

Tabla 15:Caso de prueba de funcionalidad a la historia de usuario eliminar incidencia (elaboración propia) 80

INTRODUCCIÓN

Las Tecnologías de la Información y las Comunicaciones (TIC) son el resultado de la interrelación de muchos componentes informáticos, uno de ellos es el software que es controlado por monopolios empresariales que obtienen todos los años millones de dólares por concepto de pagos de licencias de uso. Sus clientes están obligados a depender de ellos porque restringen el conocimiento de su funcionamiento y no venden un producto sino el derecho a utilizarlo. Por este motivo la independencia tecnológica es una preocupación actual de muchos gobiernos y organizaciones que quieren mantener el control sobre las bases tecnológicas en las que se asientan las TIC (Delgado, 2014).

Cuba promueve, como parte de su programa de informatización, no apostar por sistemas operativos privativos como un camino viable para el desarrollo tecnológico, sino apostar por sistemas que a la vez conduzcan a la independencia en este mismo ámbito. Una de las formas de alcanzar la soberanía tecnológica es mediante la migración a software libre, la cual es una vía factible para Cuba por ser un país bloqueado y del tercer mundo. En abril de 2004 el Consejo de Ministros adoptó el acuerdo 084/2004 donde indica al Ministerio de la Informática y las Comunicaciones (MIC) ordenar el proceso de migración de Cuba a software libre paulatinamente(Montano, 2015). Siendo este, uno de los aspectos que se tuvo en cuenta en los lineamientos aprobados en el VI y VII Congreso del Partido Comunista de Cuba (PCC), donde se destacó la importancia de este proceso para la informática, la electrónica y las comunicaciones en nuestro país. (Lineamientos, 2011).

Por otro lado, está la Universidad de las Ciencias Informáticas (UCI), casa de altos estudios especializados, que posee varios centros de producción y desarrollo que cuentan con excelente prestigio tanto a nivel nacional como internacional. CESOL “Centro de Software Libre” es uno de los centros integrados a la actividad de desarrollo de la universidad, que conduce los procesos de migración a aplicaciones de código abierto. El producto estrella resultado de este centro es Nova, una distribución de GNU/LINUX hecha por cubanos y para los cubanos y utilizado hoy en el proceso de migración a software libre. (CESOL, 2022)

La misma es una variante orientada a estaciones de trabajo con medias o altas prestaciones, enfocadas en aumentar la productividad del usuario proponiendo un entorno de escritorio amigable y usable. Nova entre sus variantes comprende a Nova Escritorio, Nova Ligero, Nova Servidores, Novadroid. Este producto es desarrollado con el objetivo de reemplazar los softwares de pago que son utilizados en las empresas estatales del país y de esta forma estar un peldaño más arriba de la soberanía tecnológica. (Nova, 2022)

En la entrevista realizada a Yileni Hernández, jefa de proyecto de Nova, expresa que el sistema operativo GNU/Linux Nova debe superar el proceso de control de calidad antes de lanzar una versión oficial. Donde el equipo de desarrollo y trabajadores miembros de CESOL llevan a cabo el proceso de realización de las pruebas de validación de software. En esta colección de pruebas se recopilan un conjunto de información con el objetivo de detectar fallos o errores que puede generar el sistema. Luego de superado este procedimiento, y lanzada la nueva versión, los encargados de gestión de la calidad, evalúan y brindan una lista de chequeo, con la cual validan todas las funcionalidades requeridas y futuras mejoras para próximas versiones. Sin embargo, no se tiene en cuenta la experiencia del usuario final en la toma de decisiones, en cuanto a la calidad del producto, por no contar con los registros oficiales de los eventos ocurridos durante la interacción estos con el sistema operativo Nova.

En el libro “Buenas prácticas para la migración a código abierto”, el cual se basa en las experiencias adquiridas y resultados satisfactorios durante la migración de varias organizaciones, tanto en Cuba como en el extranjero, (Villazón,2014) se definen 3 etapas en el proceso de migración: preparación, ejecución y consolidación. En la etapa de preparación se realiza el diagnóstico de los elementos involucrados, el análisis de la información y confección del plan de migración a seguir; en la ejecución se migran servicios telemáticos y escritorios de usuarios de la institución y por último en la etapa de consolidación se realiza el servicio de soporte y asistencia técnica a usuarios y sistemas, dicha prestación hoy la brinda el Centro de Soporte de la UCI con los servicios de asistencia técnica a las aplicaciones informáticas desarrolladas por la institución.

Una vez concluido el proceso de migración, se detecta que los nuevos usuarios del sistema operativo Nova muestran resistencia al cambio, los mismos se encuentran más familiarizados con la plataforma privativa Windows. Lo anterior se abordó en el VIII taller de software libre de la Conferencia Científica de la UCI 2017, como una de las dificultades que actualmente enfrentaba el proceso de migración. Otra de la deficiencia que se manifiesta es el problema en la recopilación de errores, eventos o incidencia en el manejo del usuario, lo que dificulta la retroalimentación del equipo de desarrollo de nova mediante la opinión de los usuarios finales. (Prieto Carmona, Y., & Fuentes Bauta, J. 2019)

La información coleccionada sobre estas dificultades es de relevancia para aumentar la experiencia de usuario; al no existir una vía de comunicación directa que permita la retroalimentación entre el equipo de desarrollo y los usuarios, se dificulta el trabajo de recopilación de errores, evento o incidencia durante el manejo del sistema operativo Nova. Esto se debe a que el Centro de Soporte trabaja con 3 niveles de incidencia y solo se contacta al equipo de desarrollo cuando ocurre una incidencia del tercer nivel, por ser incidencias que solo estos pueden solucionar. Además, las pruebas realizadas por CESOL y el departamento de calidad son llevadas a cabo por expertos en el área de la informática, proporcionando informes técnicos que no tienen en cuenta los resultados de las interacciones de los usuarios finales con Nova. Entre los tipos de incidencias que puede presentar el sistema operativo Nova se encuentran las incidencias relacionadas con la conectividad, por lo que una solución de retroalimentación web que corra sobre el mismo sistema operativo puede ser un inconveniente a la hora de realizar un reporte de error. Al mismo tiempo se necesita agilidad y movilidad en dicho proceso.

Teniendo en cuenta que existen grandes limitaciones en el equipo de desarrollo de Nova al no tener retroalimentación por parte de los usuarios finales se plantea como **problema de investigación:** ¿Cómo automatizar el proceso retroalimentación mediante el reporte de incidencias luego del despliegue del GNU/Linux Nova en las instituciones?

Siendo el **objeto de estudio**: el proceso de reporte de errores, eventos o incidencias asociado a problemas en un sistema operativo, se define como **campo de acción**: aplicaciones móviles para el reporte de errores, eventos o incidencias asociado a problemas en un sistema

operativo. Para dar solución al problema se define como **objetivo genera**l: desarrollar una aplicación móvil para el reporte de incidencias luego el despliegue GNU/Linux Nova en instituciones, que facilite la retroalimentación de errores entre el equipo de desarrollo y los usuarios y así aumentar el nivel de aceptación del sistema operativo en próximas versiones.

Para dar cumplimiento al objetivo general se proponen las siguientes **tareas de Investigación:**

- Elaboración del marco teórico referencial sobre el reporte de incidencia luego del despliegue de GNU/Linux Nova y el desarrollo de aplicaciones móviles.
- Análisis de las aplicaciones desarrolladas para sistemas de reportes de incidencias en diferentes escenarios de ejecución.
- Selección de la metodología, las herramientas y tecnologías para el desarrollo de la aplicación Android como propuesta de solución.
- Generación de los artefactos que corresponden a las etapas ingenieriles definidas por la metodología AUIP-UCI.
- Implementación de una aplicación Android para el reporte de incidencia luego del despliegue de GNU/Linux Nova.
- Validación de los resultados obtenidos a partir de los métodos definidos en la investigación.

Para el desarrollo de estas **tareas de investigación**, se hizo uso de los siguientes métodos de investigación:

Método teórico:

Métodos utilizados en la investigación que permiten estudiar características del problema de investigación que no son observables directamente:

- **Analítico - sintético**: se hace uso del método para el análisis, evaluación y selección de las técnicas a emplear en el desarrollo de la aplicación. De igual forma se emplea para sintetizar la información que se obtuvo mediante el intercambio con los expertos para que pueda ser utilizada en la construcción del sistema, además, en la identificación de los elementos del marco teórico de la investigación.

- **Modelación:** Se utiliza para realizar una representación del proceso estudiado que sirva de guía en el desarrollo del sistema, y mediante esta representación, identificar las características y relaciones fundamentales. También se utilizó en el proceso de construcción de la aplicación para confeccionar los diagramas correspondientes a la representación de la propuesta de solución, al permitir la definición y descripción de las funcionalidades que conforman la aplicación.

Método empírico:

Este método tiene como característica fundamental la estrecha relación entre el cliente, quien orienta la investigación y quien la desarrollará. Entre sus métodos se emplearán la entrevista y el análisis documental.

- **Entrevista**: permitió recopilar información a través de los jefes de proyecto y miembro del equipo de desarrollo de nova (ver anexo) para definir el levantamiento de requisitos y la determinación de las restricciones de software e implementación que deben tenerse en cuenta en el proceso de desarrollo.
- **Observación:** para realizar el estudio de las características y comportamientos de las herramientas que presenten soluciones similares.
- **Análisis documental**: se utilizó en las consultas de las literaturas especializadas publicadas a nivel nacional e internacional para extraer la información necesaria que permitió diseñar la solución.

La investigación está estructurada por:

Capítulo 1: Fundamentos teórico-metodológicos sobre el objeto de estudio. Incluye un análisis y revisión del estado del arte de aplicaciones móviles para el reporte de incidencia. Se hace una descripción de las tendencias, tecnologías y herramientas utilizadas para el desarrollo de la aplicación.

Capítulo 2: Análisis y diseño de la propuesta de solución al problema científico. En el capítulo se describe la propuesta de solución y se explican los artefactos generados durante los procesos de análisis y diseño, de acuerdo con la metodología empleada.

Capítulo 3: Validación de la propuesta de solución al problema científico. Se muestran los artefactos generados durante la implementación de la solución y los resultados de las pruebas realizadas.

Para finalizar se presentan las conclusiones, las recomendaciones, las referencias bibliográficas y el conjunto de anexos para un mejor entendimiento de lo expuesto a lo largo de este trabajo.

CAPÍTULO I: FUNDAMENTOS TEÓRICO-METODOLÓGICOS SOBRE EL OBJETO DE ESTUDIO

En este capítulo se abordan los conceptos relacionados con la investigación, los cuales constituyen el conocimiento sobre los procesos y tecnologías asociadas al desarrollo de la solución requerida. Se exponen y explican las metodologías, herramientas y tecnologías a utilizar en el desarrollo de la aplicación, justificando el motivo de su selección.

1.1 Conceptos asociados al dominio del problema

1.1.1 GNU/Linux Nova:

En 2009 fue presentado oficialmente el sistema operativo libre GNU/Linux Nova, Nova surge como una distribución de GNU/Linux orientada a escritorio desarrollada en la Universidad de las Ciencias Informáticas (UCI) a través del proyecto del mismo nombre. Su desarrollo, uso y aplicación está basando en los principios de Seguridad, Soberanía Tecnológica, Socio-adaptabilidad y Sostenibilidad, definidos como las 4S, los cuales “pretenden materializar las razones que llevan a nuestro país a ejecutar el necesario proceso de migración tecnológica (Puente Fuentes et al. 2012). A partir de entonces fue propuesto como uno de los sistemas operativos a aplicar durante el proceso de migración hacia plataformas de software libre y código abierto en aras de alcanzar la soberanía tecnológica(Castro y Jimenez 2020).

1.1.2 Migración a Software Libre GNU/Linux Nova en Cuba:

La migración a software libre es el proceso por el cual un entorno informático basado en sistemas y software privativo se adapta y traslada al uso de software libre. En el mismo, se deben buscar alternativas a las herramientas privativas, asegurar una adecuada transición entre las mismas, y garantizar en todo momento la continuidad de la información y el trabajo realizado con anterioridad por los usuarios. Para poder realizar este cambio con seguridad, se deben seguir una serie de etapas que garanticen la correcta gestión del cambio de modelo desde el entorno privativo al nuevo paradigma libre.

GNU/LINUX Nova es la distribución recomendada para ser utilizada como tecnología base de despliegue y desarrollo de aplicaciones informáticas en los organismos de la administración central del estado, con el objetivo de satisfacer las necesidades de la migración a plataformas de código abierto como parte del proceso de informatización de la sociedad. Ofrece a los

usuarios la posibilidad de personalización, a la medida de sus necesidades, teniendo en cuenta las características tecnológicas de nuestro país para computadoras de alta y bajas prestaciones.

Para lograr el éxito en la ejecución de la migración se hace necesario pasar a través de varias etapas, descritas en la metodología cubana de migración a plataformas de código abierto, donde se plantean sus principales procesos. Dicha metodología se encuentra contenida en el libro “Buenas Prácticas para la migración a Código Abierto”, el cual se basa en las experiencias adquiridas y resultados satisfactorios durante la migración de varias organizaciones, tanto en Cuba como en el extranjero (Villazón,2014).

Las etapas del proceso de migración a software libre son las siguientes:

- Preparación: donde se ejecutarán todas las tareas de diagnóstico de los procesos, personas y tecnología de la entidad, se realizarán tareas de análisis de factibilidad de la migración y se emitirá el plan de migración institucional.
- Ejecución: comprende las actividades necesarias para la migración definitiva de usuarios y tecnologías de la institución. Incluye la migración de los servicios telemáticos y los escritorios de los usuarios.
- Consolidación: etapa que comprende las tareas destinadas a garantizar el soporte técnico a los usuarios e infraestructura luego de ser detectados el conjunto de fallos e incidencias a la hora de hacer uso de la nueva tecnología. Y es a partir de esta etapa en que es indispensable una correcta gestión de incidencia.

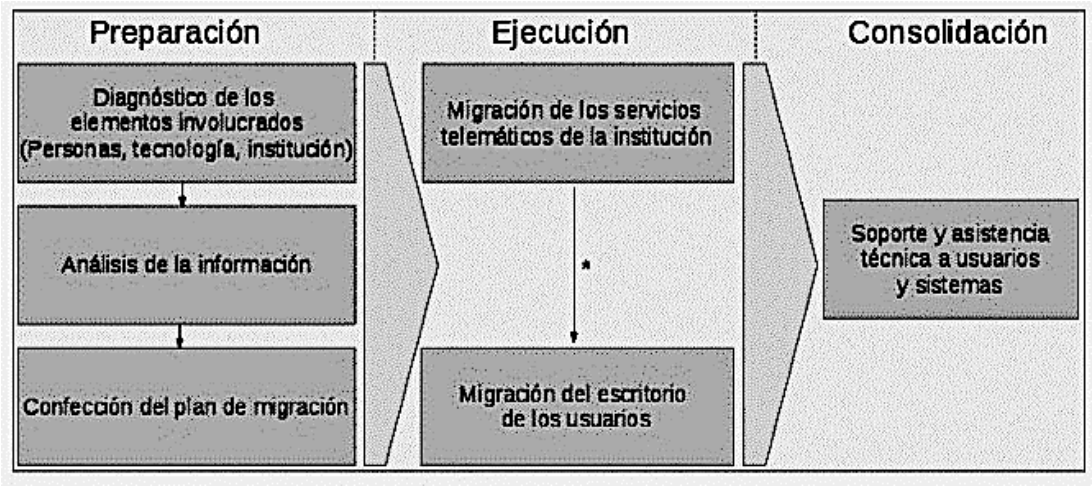


Ilustración 1:Actividades generales del proceso de migración (Villazón, 2015)

En la etapa de consolidación han sido llevados al Software Libre los servidores y las estaciones de trabajo de las computadoras presentes en la empresa. Por lo que cada una de las guías analizadas lleva a cabo un proceso de consolidación y evaluación de la migración, para corregir y perfeccionar los errores o déficit presentes en la institución, causados en las etapas anteriores, principalmente en la etapa de migración.(Hernandez Rodriguez. 2010)

Se propone hacer énfasis durante esta etapa en:

- Seguimiento, control y evaluación de procesos según los planes de implantación progresiva del software libre desarrollado con estándares abiertos.
- Continuar el monitoreo (estadísticas de uso) y mantenimiento del portal, para trabajar en función de estos resultados.
- Continuar el apoyo técnico de los grupos expertos, para lograr apoyo adicional en las instituciones vinculadas al proceso migratorio.

1.1.3 Evento no deseado o incidencia:

Se define una incidencia como “una interrupción no planificada de un servicio o reducción en la calidad del mismo” Las incidencias pueden afectar el proceso de desarrollo de un usuario, por ello la importancia de disponer de un sistema para poder minimizar las consecuencias. Se debe monitorear y analizar los eventos o incidencias que se encuentran en un proceso, debido

a su capacidad de afectar negativamente en la operación. (ISO 27001,2020)

Un error de software, evento no deseado (también conocido por el término en inglés *bug*) es un problema en un programa de computador o sistema de software que desencadena un resultado indeseado.

Entre los fallos más comunes que detectan los usuarios en uso del sistema operativo Nova se encuentran:

- Mala actualización de software.
- Problemas de incompatibilidad.
- Problemas en el manejo de la interfaz de usuario.
- Problemas con las aplicaciones ofimáticas o básicas del sistema.

Por lo que se concluye que el mal manejo de un evento no deseado o fallo puede traer consigo un costoso efecto sobre los clientes ya que se produce altos niveles de insatisfacción, además no hacer una correcta gestión trae un costo de tiempo y recursos para la empresa pudiendo desencadenar en problemas de productividad, también hay que tener en cuenta la mala recopilación de información puede costar el éxito de futuras versiones.

1.1.4 Reporte de incidencia:

El principal objetivo de un reporte de incidencia es permitir a el equipo de desarrollo, agrupar una serie de infracciones, logrando un conjunto de datos generales por categoría permitiendo obtener una vista más global según los conjuntos de incidencias reportados. (IBM, 2021)

En el proceso de detección y ratificación de errores en la distribución de GNU/Linux Nova,(Prieto Carmona, Y., & Fuentes Bauta, J. . 2019) plantea los factores que se estipulan a continuación:

- Los errores humanos cometidos por los especialistas.
- La poca documentación de las condiciones iniciales de la entidad que afecta la planificación de una acertada estrategia de migración a la medida.
- La carencia de sistemas informáticos soportados en plataformas Linux capaces de cubrir la necesidad de automatización de sistemas contables.

Luego de realizado un análisis de estos factores se puede observar que es necesario que en el momento de generar una incidencia por parte del usuario contenga los siguientes atributos:

- Identificador de incidencia.
- Nombre de la incidencia.
- Identificador de usuario que realiza el reporte.
- Categoría.
- Estado de gravedad.
- Estado de la incidencia.

Esto es necesario para lograr un correcto manejo del uso de la información ya que es de importancia el proceso de controlar, almacenar y recuperar la información adquirida por una entidad a través de diferentes fuentes, con el fin de usar la recopilación de información adquirida en futura mejoras del producto. De igual modo pone en uso los recursos de información de la organización, de origen tanto interno como externo para operar, aprender y adaptarse a los cambios del ambiente.

De esta forma se logra tener un control de las incidencias de mayor prioridad a menor, optimiza el tiempo de detección y que el equipo de desarrollo acote la deficiencia detectada, dando paso a: (IBM,2021)

- Crear un proceso de estudio que sea consistente y medible.
- Hacer la comunicación y la coordinación una prioridad para la organización.
- Lograr un enlace directo entre equipo de desarrollador - usuario.

Por lo que se concluye que tener un proceso de reporte de incidencias bien definido puede servir para reducir el tiempo de detección y aumentar el nivel de aceptación de los usuarios, puesto que se crea una comunicación durante los incidentes que permite aumentar el aprendizaje continuo del equipo de desarrollo dando paso a trabajar con incidencias vigentes.

1.1.5 Aplicación móvil:

El concepto de aplicación móvil puede ser bastante intuitivo. La compañía telefónica AT&T la define en su glosario como “*un programa que se ejecuta en un dispositivo móvil*” (AT&T,2021).

IBM por su parte ofrece un concepto similar en su glosario de términos: *"es una aplicación que ha sido diseñada para una plataforma móvil y ofrece funciones más allá que solo mostrar información estática*" (IBM,2021). En el libro *"Mobile Learning: Nuevas realidades en el aula"* de Raúl Santiago se define como: *"toda aplicación informática diseñada para ser ejecutada en teléfonos inteligentes, tablets y otros dispositivos móviles. Por lo general se encuentran disponibles a través de plataformas de distribución"*(Santiago et al. 2015). El autor de la presente investigación considera esta última definición como la más acertada. A partir de la recopilación de estos conceptos se puede definir una aplicación móvil como: un programa informático diseñado para ser ejecutado en dispositivos móviles como tabletas y celulares que ofrece una solución a un problema o necesidad.

1.1.5.1 Tipos de aplicaciones móviles:

Las aplicaciones móviles no todas tienen las mismas características, ni son del mismo tipo. Los tipos de aplicaciones móviles que se conocen son: **nativas, web e híbridas**(Salina 2015)**.**

Nativas: Son aquellas desarrolladas bajo un lenguaje y entorno de desarrollo especifico, lo cual permite que su funcionamiento sea muy fluido y estable para el sistema operativo que fue creada. Las mismas no necesitan conexión a internet para su funcionamiento, pueden hacer uso de las notificaciones del sistema operativo para mostrar avisos importantes al usuario, aun cuando no se esté usando la aplicación. Ofrecen una experiencia de uso fluida integrándose a las características de hardware del terminal, como cámara y sensores.

Web: Se puede definir las mismas como aplicaciones para brindar accesibilidad a la información desde cualquier dispositivo, sin importar el sistema operativo, solo se necesita un navegador para acceder a la misma. Estas aplicaciones se ejecutan dentro del propio navegador web del dispositivo a través de una URL, el contenido se adapta a la pantalla adquiriendo un aspecto de navegación, se adapta a cualquier sistema operativo y al tratarse sobre aplicaciones que funciona sobre la web, no es necesario que el usuario reciba actualizaciones ya que siempre va a estar viendo la última versión.

Híbridas: Se catalogan de esta forma porque son aplicaciones que combinan aspectos de las aplicaciones móviles nativas y de las webs. La facilidad que brinda este tipo de desarrollo es que no hay un entorno especifico el cual hay que utilizar para su desarrollo. Las mismas dan

la posibilidad de acceder a gran parte de las características del hardware del dispositivo, también permite acceder usando las librerías, a las capacidades del teléfono y cuentan con un diseño visual que no se identifica en gran medida con el del sistema operativo.

Por tanto, se puede caracterizar que las aplicaciones de Incidencias suelen conectarse a través de una API para acceder a la información del servidor, no necesariamente deben tener conexión con una web para su uso o su correcto funcionamiento, ofrecen una experiencia de uso fluida integrándose a las características de hardware del terminal y a todas las funciones del mismo. Se concluye así que este tipo de aplicaciones son nativas por lo que la aplicación a desarrollar contará con estas características.

1.1.6 Sistema operativo móvil.

Un sistema operativo móvil o SO móvil es un conjunto de programas que permite la abstracción de las peculiaridades específicas del teléfono móvil y, provee servicios a las aplicaciones móviles, que se ejecutan sobre él orientados hacia la conectividad inalámbrica. Los más populares son Android y el de Apple, que se conoce como iOS. (Polanco y Taibo, 2011)

1.1.6.1Sistema operativo Android:

Android es una plataforma de desarrollo libre basada en Linux y de código abierto, no ha sido diseñada exclusivamente para su uso en teléfonos y Tablet, se puede encontrar en otros dispositivos inteligentes como relojes, cámaras, tv, sistemas para automóviles y electrodomésticos. Es desarrollado por la *Open Handset Alliance* que es un conglomerado de fabricantes y desarrolladores de hardware, software y operadores de servicio, la cual es liderada por Google. Fue desarrollado inicialmente por Android Inc., una firma comprada por Google en 2005. Google liberó la mayoría del código de Android bajo la licencia Apache,una licencia libre y de código abierto. (Gironés 2019), (Montero, 2021)

I.2 Estudio de aplicaciones referente al reporte de incidencias.

Las aplicaciones de reporte de incidencia son una herramienta útil y resolutiva si de la recopilación de eventos o errores se trata, las mismas facilitan la gestión de información referentes a proyectos, actividades u organismo, lo que posibilita la retroalimentación a la hora de solucionar problemas o eventos no deseados. Se han popularizados en los últimos años,

tanto en el ámbito internacional, (IBM, 2021) como nacional por ser notable los resultados positivos que permiten.

Se podría evaluar como objetivos imprescindibles a cumplir en una aplicación de reporte de Incidencia:

- Clasificación de la incidencia según los rangos requeridos.
- Poder crear, modificar y eliminar una incidencia.
- Asociar una evaluación.
- Mostrar los detalles de la incidencia.
- Especificar el área dentro del sistema donde ocurre la incidencia.
- Mostrar el estado en la que se encuentra la misma.

1.2.1 Aplicaciones homólogas:

1.2.1.1 Internacionales:

- **Full Audits - Auditorías y Gestión de Incidencias:**

Full Audits es una aplicación privativa creada para ejecutar auditorías de calidad, identificar amenazas y oportunidades de mejora en diferentes industrias y negocios en todo el mundo. Con un sólido proceso de gestión de incidentes, la herramienta ayuda a identificar incidencias y gestionarlas hasta su resolución. (Fullaudits, 2022)

- **UUUPSApp - Gestión de Incidencias:**

Es una aplicación privativa creada para que, de manera fácil y sencilla, los usuarios pueden dar de alta los partes de incidencia en su teléfono móvil y dirigirlos al área o departamento encargado de su resolución. Se trata de una herramienta de fácil usabilidad, muy sencilla e intuitiva que permite un ahorro de tiempo en la resolución de los problemas muy significativos y a la vez permite un análisis muy exhaustivo de las incidencias acontecidas y registradas. (UUUPSAPP!, 2022)

- **Kizeo Forms – Aplicación móvil para el reporte de incidencia:**

Kizeo Forms es una herramienta para la recogida de información de procesos y digitalización de los mismos. Al crear una cuenta, tienes acceso a numerosas funcionalidades entre la que

se encuentra creación de formularios aún sin acceso a Internet, Asignación de categoría o área y generar al instante informe permitiendo el envío automático a los implicados. Es 100% personalizable y no es necesario altos conocimientos en informática para su uso.(Kizeo, 2022)

- **Servicio y comentarios en XIAOMI:**

Mediante esta aplicación móvil se puede reportar fallos que se presenten en un dispositivo móvil Xiaomi o puedes brindar sugerencias para mejorar su sistema MIUI. Se puede reportar cualquier tipo de fallo que tenga el dispositivo, tanto a nivel de sistema como de funciones o cualquier otro conflicto de aplicaciones de tercero. Para realizar un reporte, se debe elegir la categoría, describir en detalle el problema y si es posible enviar alguna captura de pantalla y un registro. Se debe tener una cuenta Xiaomi para poder hacer uso de la aplicación.(Xiaomi, 2022)

1.2.1.2 Nacionales:

- **Incidencias UCI (incidencias.uci.cu) - Gestión de incidencias en el campus universitario:**

La Universidad de las Ciencias Informáticas, es un área que, desde el punto de vista administrativo, posee una infraestructura con un alto número de recursos materiales y constructivos. Cuenta con una aplicación web diseñada dada la necesidad de automatizar el proceso de incidencias en el campus universitario y de esa forma gestionar los reportes de mantenimiento generados en la Universidad más rápida y eficiente, para que sean atendidos de manera interna y así lograr realizar reparaciones a equipos e instalaciones tanto a docente, oficinas o en el edificio donde radican los estudiantes y trabajadores de la UCI. Disponible en: https://incidencias.uci.cu

- **Herramienta Versige – Empresas de Aplicaciones Informáticas (Desoft)**

Herramienta de trabajo que facilita la gestión de la información de errores, recopilada por diferentes entidades nacionales, la misma prevé disminuir los fallos humanos además de agilizar la recepción de datos para conformación de estados financieros y procesos estadísticos.(Desoft, 2022)

Partiendo de lo antes expuesto, se muestra a continuación una tabla comparativa, exponiendo los parámetros deseados del producto final junto con las aplicaciones ya existentes.

No.	Nombre	Privativa	Objetivos	Android	Funcionalidades
1.	Full Audits	Si	Cumple con todas las características deseadas, pero está orientada a negocio empresarial.	Si	1.Crear Incidencia 2.Auditar proceso
2.	UUUPSApp	Si	Es interactiva, sus funciones están bien definidas, es de carácter empresarial.	Si	1.Gestionar Incidencias 2. Documentar la respuesta de los administradores
3.	Kizeo Forms	si	Básica y de fácil uso para el usuario promedio. Dedicada a reporte de incidencias de servicios básicos.	Si	1.Gestionar la incidencia 2.Localizar un Administrador
4.	Servicio y comentarios en XIAOMI:	si	Cumple con los objetivos, su estructura es creada para ayudar el proceso de desarrollo del sistema MIUI Xiaomi.	Si	1.Gestionar un sistema de reporte 2. Aportar sugerencias
5.	Incidencias UCI	No	Cuenta con un diseño claro, contiene herramientas y fácil uso para el usuario.	No	1.Crear la Incidencia 2. Visualiza su estado y proceso

					3. Brinda Respuesta a la incidencia.
6.	Versige	si	No cumple con todos los objetivos, la misma se especializa más en detectar errores de un proceso contable	no	1.Introducir Datos 2.Detección de Incidencia

Tabla 1: Aplicaciones homologas al sistema

Al realizar el análisis de las soluciones similares existentes para el reporte de incidencia se ha podido constatar que ninguna es aplicable para dar solución al problema de investigación. En primera instancia a nivel internacional se tiene que las aplicaciones Full Audits, UUUPSApp, Kizeo Forms Servicio y comentarios en XIAOMI son de uso privativo y no todas cubren los objetivos que se desean con la aplicación debido a que poseen funcionalidades extras como geolocalización, generación de informes contables o solo garantizan el reporte de incidencia. A nivel nacional se cuenta con aplicaciones que se pueden aplicar para la resolución del problema a investigar, pero no tienen compatibilidad con el sistema operativo Android o trabajan para garantizar la resolución de la incidencia. Con la creación de la aplicación de la propuesta de solución lo que se busca es contar con una aplicación que permita reportar incidencias con el objetivo de captar una vista global de los problemas existente en el uso del sistema operativo Nova, por tanto se crearía una herramienta totalmente adaptada a las funcionalidades deseada para la solución de la problemática existente siendo su uso totalmente gratuito y libre.

1.3 Metodología, lenguaje, herramientas y entorno de desarrollo

A continuación, se describe la metodología, el lenguaje de programación, las herramientas y el entorno de desarrollo utilizados en la investigación y puesta en práctica.

1.3.1 Metodología de desarrollo de software AUP-UCI

Existen diferentes modelos y metodologías que han sido en los últimos años herramientas de apoyo para el desarrollo del software. Sommerville (2011) define que un método de ingeniería de software es un enfoque estructurado para el desarrollo de software cuyo propósito es

facilitar la producción de software de alta calidad de una forma costeable. Para usar este enfoque se debe manejar conceptos fundamentales como: procesos, métodos, tareas, procedimientos, técnicas, herramientas, productos, entre otros.

En la presente investigación se selecciona como metodología de desarrollo de software AUP-UCI, que es una adaptación del Proceso Unificado Ágil de Scott Ambler o *Agile Unified Process* (AUP*)* quien a su vez es una versión simplificada del Proceso Unificado Racional (RUP) adaptada a la UCI. Dicha metodología describe de una manera simple y fácil de entender, la forma de desarrollar aplicaciones de software de negocio usando técnicas ágiles y manteniendo como principios básicos la simplicidad, agilidad, centrarse en actividades de alto valor y libertades en el momento de elegir las herramientas más adecuadas para el trabajo y adaptarla a las necesidades. De las 4 fases que propone AUP (Inicio, Elaboración, Construcción, Transición) se decide para el ciclo de vida de los proyectos de la UCI mantener la fase de Inicio, pero modificando el objetivo de la misma, se unifican las restantes 3 fases de AUP en una sola, llamada Ejecución y se agrega la fase de Cierre esto se planteó con el objetivo lograr estandarizar el proceso de desarrollo de software en la actividad productiva de la UCI se hizo esta adaptación, sus fases, roles y escenarios (Rodríguez, 2015)

- **Inicio:** Durante el inicio del proyecto se llevan a cabo las actividades relacionadas con la planeacióndel proyecto. En esta fase se realiza un estudio inicial de la organización cliente que permite obtener información fundamental acerca del alcance del proyecto, realizar estimaciones detiempo, esfuerzo y costo, así como decidir si se ejecuta o no el proyecto.
- **Ejecución**: En esta fase se ejecutan las actividades requeridas para desarrollar el software, incluye el ajuste de los planes del proyecto considera los requisitos y la arquitectura. Durante el desarrollo se modela el negocio, se obtienen los requisitos, se establecen la arquitectura y el diseño, se implementa y se libera el producto.
- **Cierre:** En esta fase se analizan los resultados del proyecto y se realizan las actividades formales de cierre del proyecto.

En la presente investigación se utiliza el escenario numero 4 descrito dentro de la Metodología AUP-UCI. Se utiliza este escenario ya que estipula que el cliente estará siempre acompañando al equipo de desarrollo para convenir los detalles de los requisitos y así poder implementarlos,

probarlos y validarlos además que la misma se recomienda en proyectos no muy extensos, ya que una HU no debe poseer demasiada información. El escenario 4 recoge las características del proyecto en desarrollo.

1.3.2 Lenguaje modelado.

El lenguaje de modelado permite dar apoyo a la mayoría de los procesos de desarrollo de software, estos son desarrollados con el esfuerzo de simplificar y consolidar el gran número de métodos de desarrollo que han surgido. Poseen como objetivo principal visualizar de manera gráfica el sistema que se desarrollará, como ayuda técnica a los ingenieros implicados (*Unified Modeling Language*, 2022).

UML v2.5:

UML es un lenguaje de modelado de sistemas de software. Abarca la estructura de las aplicaciones, su comportamiento y arquitectura, procesos de negocio y datos. Incluye todo el proceso de desarrollo de software. Estandariza la forma de crear diagramas, el significado preciso de los mismos y las relaciones existentes entre ellos. Utilizado para especificar, construir, visualizar y documentar los artefactos de un sistema de software orientado a objeto.(Burgués 2018)

Visual Paradigm v15.1:

Visual Paradigm es una herramienta profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Soporta, entre otros,el lenguaje de modelado UML. El software de modelado UML mejora la rapidez en la construcción de aplicaciones de calidad y a un menor costo. Está disponible para Windows y GNU/Linux. Permite dibujar diagramas de clases, código inverso, generar código desde diagramas y generar documentación, así como un diseño centrado en casos de uso y enfocado al negocio que genera un software de mayor calidad (Visual Paradigm, 2016).

Se hace uso de este programa ya que permite soportar el ciclo de vida completo del proceso de desarrollo del software a través de la representación de todo tipo de diagramas.

1.3.3 Lenguaje de programación.

Un lenguaje de programación es un lenguaje artificial que puede utilizarse para definir una secuencia de instrucciones para su procesamiento por un ordenador o computadora. Pueden usarse para crear programas que controlen el comportamiento físico y lógico de una máquina, para expresar algoritmos con precisión o como modo de comunicación humana (Alonso, 2014).

Java:

Java es un lenguaje originalmente desarrollado por un grupo de ingenieros de Sun, utilizado por Netscape posteriormente como base para JavaScript. Si bien su uso se destaca en la web, permite crear todo tipo de aplicaciones locales, intranet o internet (Oracle Corporation, 2016).

Algunas características notables son:

- Orientado a objetos.
- Independiente de la plataforma.
- Dinámico.
- Interpretado.
- Robusto.

Se hace uso de este lenguaje por su extrema versatilidad, la hace compatible con todas las plataformas, es un lenguaje de programación de código abierto por lo que se puede aprovechar fuentes existentes, además de que posee una biblioteca de código abierto. Haciendo uso del mismo se reduce la curva de aprendizaje, el desarrollador cuenta con el dominio sobre el lenguaje de programación.

1.3.4 Entorno de desarrollo integrado.

El entorno de desarrollo integrado (IDE, por sus siglas en inglés, *Integrated Development Environment*) es un programa informático compuesto por herramientas de programación que pueden ser partes de aplicaciones existentes. Estos sistemas pueden manejar uno o varios lenguajes de programación ejemplo: Java, C# y C++. Han sido empaquetados como programas de aplicaciones, editores de código, depuradores, compiladores o constructores

de interfaces gráficas (Suárez, 2015).

Android Studio v4.1.2:

Android Studio es un IDE compatible para el desarrollo de aplicaciones en Android y con su uso recibe un nuevo sistema de construcción unificado, que está totalmente integrado para permitir la máxima flexibilidad en su proceso de desarrollo (Grant, 2014). Entre sus características principales se encuentra la renderización en tiempo real; la incorporación de Gradle[1] como compilador, ofreciendo múltiples variantes para la construcción y generación de APKs; la existencia de plantillas predeterminadas para crear diseños comunes de Android (Android Developers, 2020).

Se hace uso del IDE Android Studio ya que permite ejecutar las compilaciones de forma rápida y poder comprobar los fallos existentes, ejecuta la aplicación en tiempo real desde el propio dispositivo móvil además de que permite simular diferentes dispositivos y tabletas. Se pueden crear elementos gráficos para la interfaz de la aplicación y es el IDE oficial de Android.

SDK:

SDK (por sus siglas en inglés, *Software Development Kit*) de Android contiene las herramientas para desarrollar aplicaciones desde la línea de comandos, así como otras herramientas que ayudaran a encontrar y diagnosticar problemas, y optimizar las aplicaciones (Grant, 2014).

El SDK de Android incluye:

- Librerías necesarias.
- Entorno de depuración.
- Emulador de dispositivos móviles.
- Documentación relevante para las *APIs* de Android.
- Código fuente de ejemplo.
- Tutoriales para el sistema operativo Android.

[1] Herramienta que permite la automatización de compilación de código abierto. Es el **sistema de compilación oficial para Android**(*kousen,2016*)

1.3.5 Herramienta para el diseño de prototipo de interfaz

Las herramientas de prototipo de interfaz de usuario permiten el diseño de computadoras, aplicaciones, máquinas, dispositivos de comunicación móvil, aplicaciones de software y sitios web enfocado en la experiencia de usuario y la interacción. Normalmente es una actividad multidisciplinaria que involucra a varias ramas del diseño y el conocimiento como el diseño gráfico, industrial y de software. Está implicado en un amplio rango de proyectos, desde sistemas para computadoras, vehículos hasta aviones comerciales(Granollers, 2014).

Balsamiq Mockups v3.5

Balsamiq Mockups es una herramienta que reproduce la experiencia de realizar bocetos en una pizarra a través de un ordenador, por lo que es posible instalarlo en Windows, Mac y GNU/Linux. Permite crear de un modo dinámico, sencillo y práctico los bocetos de interfaces de usuario, no solo para páginas web, sino también para aplicaciones de escritorio o dispositivos móviles. Posee gran cantidad de componentes visuales para la construcción de interfaces de usuario, íconos y elementos reutilizables tales como plantillas.(Balsamiq, 2016).

Se hace uso del mismo ya que permite de forma fácil, rápida e intuitiva la creación de bocetos a utilizar para el desarrollo de la interfaz de la aplicación.

1.3.6 Gestor de base de datos

Un gestor de base de datos, también conocido RDBMS (*Relational DataBase Management Systems,* Sistemas de Gestión de Base de Datos Relacionales) en caso de bases de datos relacionales, es un tipo de software de servidor que permiten la organización de la información mediante el uso de tablas, índices y registros**.** A nivel de hardware, es un equipo informático especializado en servir consultas a clientes remotos o locales que solicitan información o realizan modificaciones a los registros y tablas que existen dentro de las bases de datos del sistema**,** en muchos casos desde un servidor web o de aplicaciones (Borges 2019).

SQLite v3.6

Es un sistema gestor de bases de datos relacional compatible con Atomicidad, Consistencia, Aislamiento yDurabilidad (ACID, por sus siglas en inglés *Atomicity, Consistency, Isolation and Durability*), comprendida en una biblioteca relativamente pequeña. Es una librería escrita en

C que implementa un motor de bases de datos para SQL92 (Sqlite, 2016).

Se hace uso de este gestor ya que dentro de sus ventajas de encuentra: No requiere un proceso o sistema de servidor independiente para operar, no se necesita configuración ni administración, compatible con la mayoría de las características de lenguaje de consulta que se encuentran en el estándar SQL92 (SQL2) y está disponible en UNIX (Linux, Mac OS-X, Android, iOS) y Windows (Win32, Win64, WinCE, WinRT).

1.3.7 Herramienta de pruebas.

El control de calidad de software lleva consigo herramientas que permiten realizar pruebas autónomas y masivas permitiendo así la verificación desde el punto de vista estático y de caja blanca. Son pruebas donde se analiza el software sin ejecutarlo mediante el código fuente del mismo (Barrientos, 2014).

JUnit v4.12

Es un marco de trabajo creado por Erich Gamma y Kent Becket para realizar pruebas unitarias a aplicaciones que empleen el lenguaje de programación Java. Incluye formas de ver los resultados de las pruebas ya sea en forma de texto, gráfico o como tarea. Incluido a través de plugins en los principales IDEs como son NetBeans, Eclipse y Android Studio (JUnit, 2016)

Se uso de esta funcionalidad ya que JUnit se trata de un Framework Open Source para la automatización de las pruebas, provee de herramientas, clases y métodos que le facilitan la tarea de realizar pruebas en la aplicación a desarrollar y así asegurar su consistencia y funcionalidad.

Conclusiones del capítulo

Luego de ser consultada la bibliografía especializada y obtener un marco teórico y estado del arte para la investigación, se concluye lo siguiente:

- El análisis del estado del arte concluyó que no existe ninguna solución aplicable al problema de la investigación, sin embargo, las aplicaciones de gestión de incidencias tratadas poseen funcionalidades comunes que serán tomadas en cuenta para la construcción de la aplicación.
- El estudio de los diferentes tipos de aplicaciones móviles permitió identificar el desarrollo de una aplicación nativa, como la mejor opción para lograr un buen diseño de la propuesta de solución.
- El estudio de las metodologías de desarrollo de software demostró que el uso de la metodología AUP-UCI resulta ideal para guiar el proceso de desarrollo de la aplicación.
- El análisis de las herramientas y lenguajes utilizados para el desarrollo de aplicaciones móviles evidenció la factibilidad del IDE Android Studio y el lenguaje Java para el desarrollo de la propuesta de solución.

CAPÍTULO II: ANÁLISIS Y DISEÑO DE LA PROPUESTA DE SOLUCIÓN AL PROBLEMA CIENTÍFICO

En el presente capítulo se describen las principales características de la propuesta de solución. También se realiza la especificación de los requisitos funcionales y no funcionales, se conforman las historias de usuario correspondientes a los requisitos y los prototipos de diseño de interfaz de usuario. Se describe la arquitectura de la aplicación, así como los patrones de diseño, y se representa el diagrama de clases del diseño.

2.1 Modelo Conceptual - Propuesta de solución - Levantamiento de Requisitos

2.1.1 Modelo Conceptual:

Un modelo conceptual es un artefacto de la disciplina de análisis, construido con las reglas UML. Tiene como objetivo comprender y describir las clases más importantes, así como, identificar y explicar los conceptos significativos en el dominio del problema, identificando los atributos y las asociaciones existentes entre ellos. (Luciana, Vegetti y Marciszack 2018)

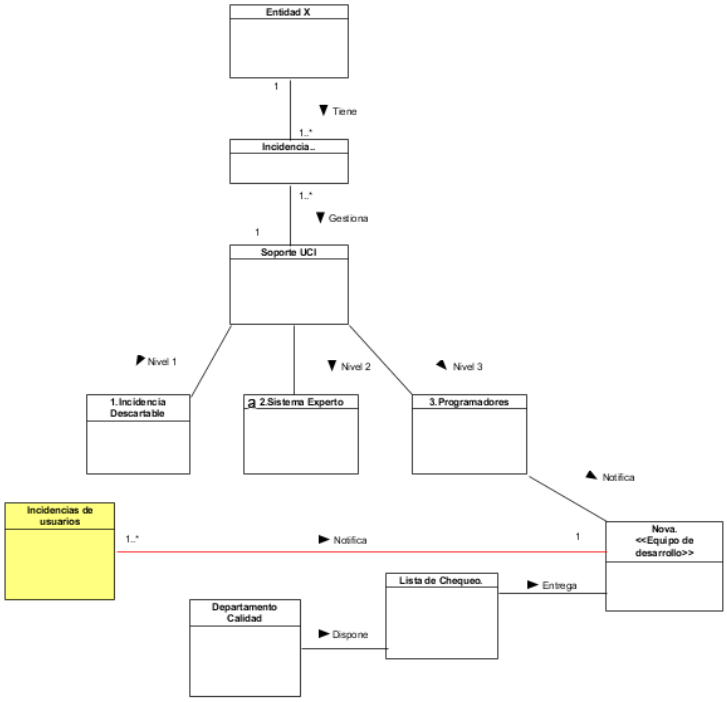


Ilustración 2:Modelo conceptual (Elaboración propia)

Se definió en el diagrama:

- **Entidad X:** Entidad donde se desplegó el sistema GNU/Linux Nova.
- **Incidencia:** Evento, fallo o incidencia detectada en el uso del sistema.
- **Soporte UCI:** Centro especializado en gestión de incidencias.
- **Incidencia descartable (Nivel 1):** Incidencia que no cumple con los patrones de Nova.
- **Sistema experto (Nivel 2):** Incidencia tratada por una base de conocimiento con información clave para su resolución.
- **Programadores (Nivel 3):** El desarrollador o el equipo de desarrollo.
- **Nova<<Equipo de desarrollo>>:** Equipo de desarrollo del sistema operativo Nova.
- **Lista de chequeo:** Documento que hace entrega el departamento de calidad, donde se verifica el cumplimiento de las funcionalidades de Nova y recomendaciones de futuras mejoras a implementar.
- **Departamento de calidad:** Centro encargado de evaluar los productos resultantes de los departamentos de la Universidad de la Ciencias Informáticas.
- **Incidencias de usuario:** Incidencias generadas por los usuarios en el uso paulatino del sistema operativo Nova.

Por lo que se concluye que el proceso de reporte de incidencias de usuario, no está definido. La información que hoy se recopila para las mejoras de las próximas versiones de Nova, surgen del departamento de calidad y de la información recopilada en el nivel 3 de soporte de incidencia.

2.1.2 Propuesta de solución:

El presente trabajo propone una aplicación para dispositivos móviles con sistema operativo Android de apoyo detección de errores, evento o incidencia luego del despliegue de Nova. El jefe del equipo de emigración junto con el equipo de desarrollo de GNU/Linux Nova debe dar a conocer, a los usuarios de la nueva institución donde se realizó el despliegue del sistema operativo, que existe una aplicación titulada “Nova Incidencia” disponible en la plataforma web www.apklis.cu. El objetivo de esta aplicación es facilitar una vía de reporte de incidencias encontradas en el uso paulatino del sistema operativo Nova. A través de la misma se adicionan

incidencias, donde por medio de una API[2] queda registrada en base de datos. Los administradores mediante una lista visualizan todos los episodios de incidencia encontradas de los usuarios de todas las instituciones donde fue desplegado el sistema operativo Nova, permitiéndoles visualizar los detalles de incidencia, cambiar el estado por la que transitan o eliminar las que consideren descartables; de forma parecida los usuarios visualizan un listado de sus incidencias, y las opciones de editar o eliminar las que deseen siempre que no se encuentren en estado de recibido, procesando o listo. La aplicación permite además a los administradores mostrar las estadísticas actuales según los datos recogidos de las incidencias en todas las instituciones, destacando los parámetros donde más recurren los usuarios. Se propone que sea a través de una aplicación móvil en Android, ya que se desea crear una herramienta simple y cómoda en su uso, para lograr que el usuario se sienta en un ambiente más favorable con el uso de su dispositivo móvil, previendo dificultar el proceso de reporte de incidencia. De esta forma obtener un avance más óptimo, tanto en tiempo como en resultado.

2.1.2 Captura de Requisito:

La captura de requisitos es uno de los pasos fundamentales en el desarrollo de software. Esta tarea está encaminada a identificar los requerimientos del cliente, analizar las necesidades y especificar los requisitos de la propuesta de solución. En la presente investigación se identificaron los requisitos funcionales y no funcionales de la solución a partir de entrevistas con el cliente.

2.1.2.1 Requisitos Funcionales:

El propósito del levantamiento de las especificaciones de los requisitos debe ser visto como un contrato entre usuario y desarrollador, que define el comportamiento funcional deseado. El proceso de reunión de requisitos funcionales se centra en el software, es fundamental que dentro del proceso de análisis se comprendan por parte del desarrollador la naturaleza de los métodos que deben construirse para desarrollar la aplicación. Tener una completa y plena

[2] No es más que un conjunto de definiciones y protocolos que se utiliza para desarrollar e integrar el software de las aplicaciones, permitiendo la comunicación entre dos aplicaciones a través de un conjunto de reglas; sirviendo para enviar y recibir datos entre apps de una manera estandarizada(Ofoeda, Boateng y Effah 2019)

comprensión tanto del diseño como del conjunto de requerimientos es de suma importancia antes de comenzar a codificar (Pressman 2010)

Para el correcto funcionamiento de la propuesta de solución se identificaron los siguientes requisitos funcionales:

No	Requisito Funcional	Descripción	Complejidad	Prioridad
1	Registrar usuario	La aplicación debe permitir a el usuario registrarse para poder hacer uso de los servicios. Procede según los siguientes parámetros: • Nombre • Apellidos • Usuario • Entidad • Contraseña • Correo.	Alta	Alta
2	Autenticar usuario	La aplicación debe permitirle a el usuario autenticarse, por su id de usuario y contraseña. Ambos predefinido en el proceso de registro.	Media	Alta
3	Adicionar una incidencia	La aplicación debe permitir solo al usuario realizar un reporte de incidencia encontrada luego de manejar GNU/Linux Nova, según los siguientes parámetros:	Alta	Alta

		• Nombre de la incidencia • Categoría de la incidencia (listado predefinido) • Gravedad de la incidencia (baja, media, alta) • Descripción de la incidencia		
4	Mostrar detalles de la incidencia	La aplicación debe mostrar al usuario los detalles registrados de la incidencia. Se muestran los siguientes datos: • Estado de la incidencia (por defecto inicial) • Nombre de la incidencia • ID de la incidencia • Categoría de la incidencia • Gravedad de la incidencia • Descripción • Editar Incidencia • Eliminar incidencia La aplicación debe mostrar al administrador los detalles registrados de la incidencia realizada por el usuario. Se muestran los siguientes datos: • Estado de la incidencia • Id de la incidencia • Usuario que realizo la incidencia • Entidad correspondiente • Categoría de la incidencia • Gravedad de la incidencia • Descripción	Media	Alta

		• Fecha • Cambiar el Estado de la incidencia • Eliminar Incidencia		
5	Editar Incidencia	La aplicación debe permitir al usuario modificar los siguientes parámetros: • Nombre de la incidencia • Categoría de la incidencia (listado predefinido) • Gravedad de la incidencia (baja, media, alta) • Descripción de la incidencia Solo se permite modificar si la incidencia está en estado de enviado	media	media
6	Eliminar Incidencia	La aplicación debe permitir al usuario eliminar la incidencia seleccionada de la lista, siempre que su estado no sea Recibido, procesando o Listo. La aplicación debe permitir al administrador eliminar todas las incidencias de la lista.	media	media
7	Cambiar de estado	La aplicación de permitir al administrador cambiar la situación del estado definido en fases: • Enviado: etapa en la que el usuario envía la incidencia detectada • Recibido: etapa donde el equipo de desarrollo visualiza la	media	media

		incidencia • Procesando: Incidencia recibida por el equipo de desarrollo lista para su análisis • Listo: Momento en que se la incidencia a sido considerada para la próxima versiona de Nova		
8	Listar Incidencia	La aplicación debe mostrar al usuario un listado de las incidencias creada por el mismo, donde se muestra una breve información de cada incidencia con los siguientes datos: • Nombre de la incidencia • Gravedad • Estado de la incidencia La aplicación debe mostrar al administrador un listado de las incidencias creada por todos los usuarios, donde se muestra una breve información de cada incidencia con los siguientes datos: • Nombre de la incidencia • Usuario • Gravedad • Estado de la incidencia	Alta	Alta
9	Buscar incidencia	La aplicación debe permitir realizar una búsqueda por el listado de las incidencias realizadas, según los filtros y criterios de búsquedas especificados. La aplicación debe permitir filtrar el	Alta	Alta

		listado de las incidencias, según los siguientes parámetros: • Usuario • Nombre • Gravedad • Categoría		
10	Actualizar	La aplicación debe permitir actualizar, lo que permitirá cargar datos nuevos de la Base de datos	Media	media
11	Mostrar estadísticas de las incidencias	La aplicación debe permitir al administrador mostrar el total de incidencias dividas por total, gravedad, estado y categorías.	media	Alta

Tabla 2: Requisitos Funcionales.

2.1.2.2 Requisitos no funcionales:

Los requisitos no funcionales son propiedades o cualidades que el producto debe tener; son las características que lo hacen atractivo, usable, rápido y confiable (Sommerville, 2011). Las restricciones impuestas al software y restricciones en la libertad del diseño, relacionándose con los requerimientos funcionales. Para lograr la satisfacción del cliente, y una buena calidad en el sistema, se definieron los siguientes requisitos no funcionales, basados en lo establecido por las normas ISO 25000 Calidad del Producto de Software, específicamente la ISO/IEC 25010, la cual define las características de calidad que se tienen en cuenta al evaluar las propiedades de un producto de software.

Se definen como requisitos no funcionales:

Clasificación	Descripción

RNF de software	**RNF_1:** Es necesario un sistema operativo Android con versión 5.0 o superior.
RNF de usabilidad	**RNF_2:** La aplicación debe poseer una interfaz intuitiva y amigable para asegurar la fácil navegación del usuario por la misma.
	RNF_3: Debe seguir los patrones del diseño material, creado y recomendado por Google, para el diseño de colores, íconos y componentes de la interfaz.
RNF de hardware	**RNF_4:** Se necesita un dispositivo móvil inteligente. **RNF_5:** La aplicación necesita al menos 30 megabytes de almacenamiento para ser instalada y 121 megabytes de memoria RAM.
RNF de seguridad	**RNF_6:** La comunicación externa entre la Api y el servidor se realiza a través del protocolo de Transferencia de Hiper-Texto seguro HTTPS **RNF_7:** Las contraseñas guardadas en la base de datos, se cifrarán por el hashing MD5

Tabla 3: Requisitos no funcionales

2.2 Historias de Usuario

Dentro de los métodos ágiles, las historias de usuario se utilizan principalmente como artefactos de requisitos principales y unidades de funcionalidad del proyecto, en el contexto de la ingeniería de requisitos ágil, como una herramienta de comunicación que combina las fortalezas de ambos medios: escrito y verbal. (Wautelet et al. 2013) Es expresar los requisitos sobre una base de baja abstracción utilizando lenguaje natural, describiendo una funcionalidad de software desde el punto de vista del usuario y que necesidades o problema soluciona lo que se va a construir. La mayoría de ellos están centrados exclusivamente en el usuario final como único actor. A lo largo de los años, algunos modelos han sido propuestos por profesionales o académicos de métodos ágiles para guiar la recopilación de requisitos. (Menzinsky et al. 2022)

Las historias de usuario son una herramienta que agiliza la administración de requisitos, reduciendo la cantidad de documentos formales y tiempo necesarios. Forman parte de la fórmula de captura de funcionalidades definida en 2001 por Ron Jeffries de las tres Cs:

- ***(Tarjeta) Card*:** cada historia de usuario se reduce hasta hacerla fácil de memorizar y de sintetizar en una tarjeta. La tarjeta sirve como recordatorio y promesa de una conversación posterior.
- ***(Conversación) Conversation*:** el equipo de desarrollo y el propietario del producto añaden criterios de aceptación a cada historia poco antes de su implementación. Los cambios son bienvenidos en agilidad, por lo que no tiene sentido profundizar en estos detalles antes. La situación puede variar mucho desde el momento en el que se sintetiza la funcionalidad en la tarjeta hasta que se implementa.
- ***(Confirmación) Confirmation*:** el propietario del producto o usuario de negocio confirma que el equipo de desarrollo ha entendido y recogido correctamente sus requisitos revisando los criterios de aceptación. A veces se pueden presentar transformados en escenarios de pruebas.

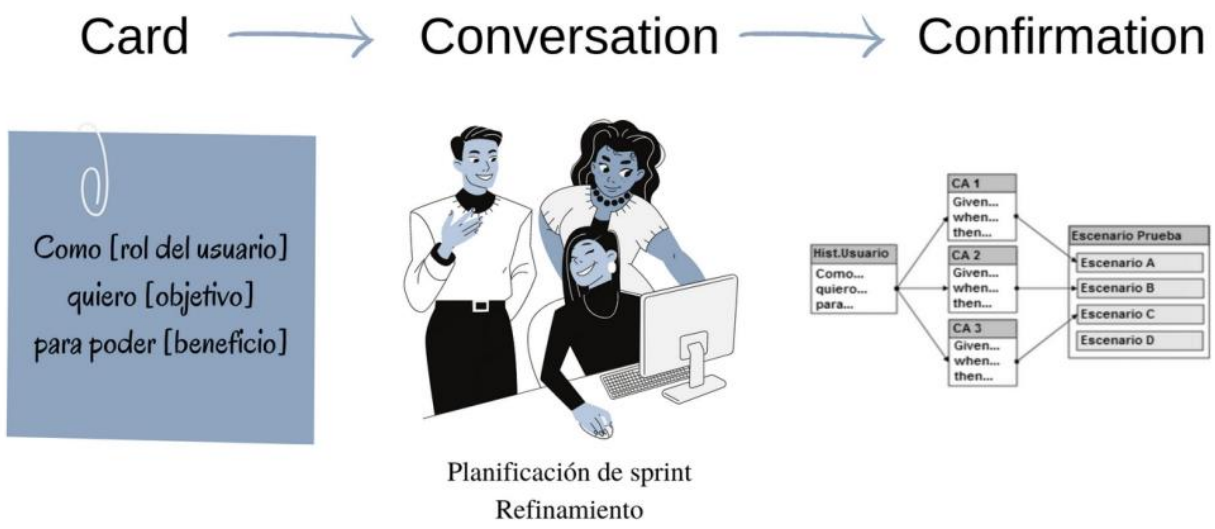


Ilustración 3: Componentes de la Historia de Usuario(Menzinsky et al. 2022)

Para la descripción de los requisitos funcionales de la propuesta de solución se define una historia de usuario para cada uno, para un total de 15 historias de usuario. A continuación, se

muestran la HU “Adicionar una incidencia” por ser la que describe la funcionalidad que tienen mayor prioridad para el cliente. El resto de las historias de usuario se definieron en el apartado de anexo.

Historia de Usuario	
Número: 3	**Nombre del Requisito:** Adicionar una incidencia
Programador: Julio C. Gallardo Concepción	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 1 semana
Riesgo en Desarrollo: Alto	**Tiempo Real:** 5 días
Descripción: La aplicación debe permitir solo al usuario realizar un reporte de incidencia encontrada luego de manejar GNU/Linux Nova, según los siguientes parámetros: • Nombre de la incidencia • Categoría de la incidencia (listado predefinido) • Gravedad de la incidencia (baja, media, alta) • Descripción de la incidencia	
Observaciones: todos los parámetros de la incidencia deben estar rellenado por el usuario, sin esto no es posible registrar la incidencia.	
Prototipo de interfaz:	

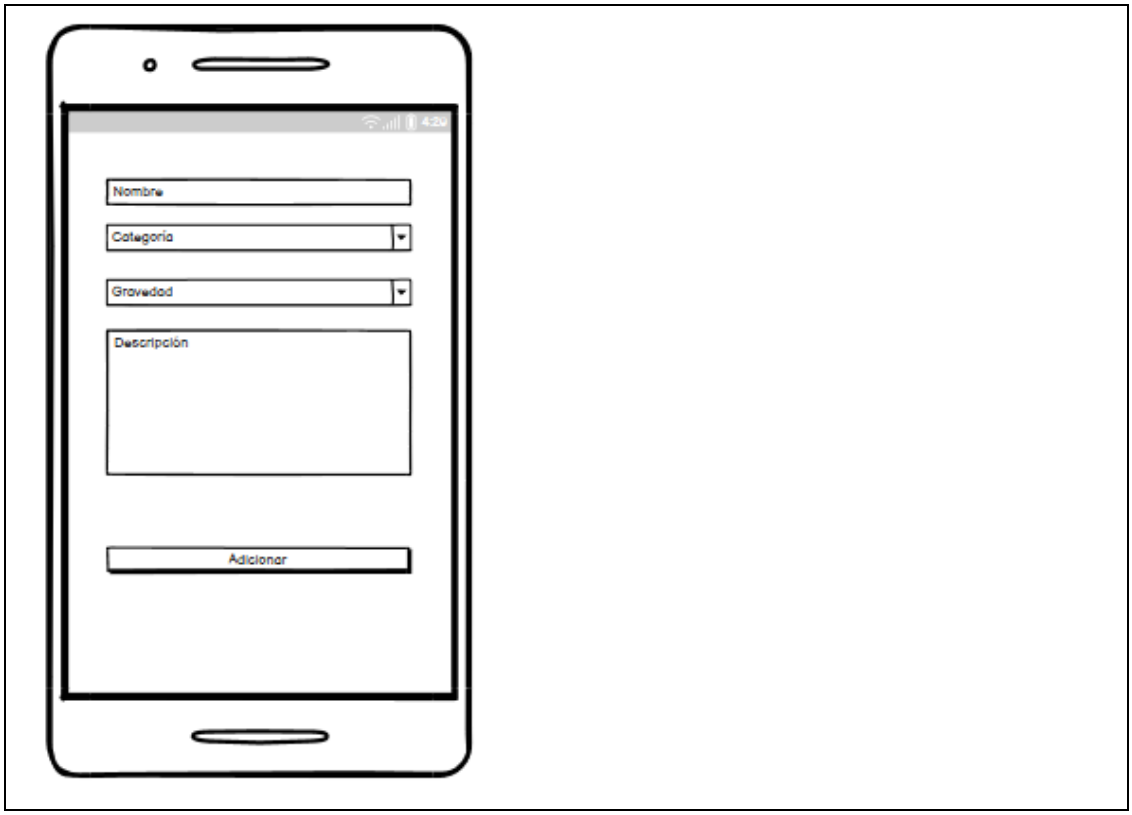


Tabla 4: HU Adicionar Incidencias

2.3 Modelo del Diseño

El diseño crea una representación o modelo de software, que a diferencia del modelo de análisis (que se enfoca en la descripción de los datos, las funciones y el comportamiento requeridos), el modelo de diseño proporciona detalles acerca de las estructuras de los datos, las arquitecturas, las interfaces y los componentes del software que son necesarios para implementar el sistema (Pressman, 2010).

2.3.1Descripción de la arquitectura Modelo Vista - Presentador

Según Pressman (2010) en su forma más simple, la arquitectura del software es la estructura u organización de los componentes del programa, es la manera en que estos interactúan y como se estructura los datos que utilizan. El uso correcto de la arquitectura del software es un factor relevante debido a que el rendimiento de la aplicación de Android es un factor que debe tenerse en cuenta, un dispositivo Android tiene energía, memoria y recursos limitados siendo

un desafío para los desarrolladores a la hora de mejorar el rendimiento de las aplicaciones y que puedan ser ejecutadas de la manera más óptima posible.

Para la aplicación se utiliza el estilo llamada y retorno. Este estilo permite obtener una estructura de programa relativamente fácil de modificar y escalar. Dentro de este se seleccionó el patrón de arquitectura de software Modelo-Vista-Presentador (MVP).

Este patrón es una variante del Modelo Vista Controlador (MVC). MVP tiene la finalidad de separar la lógica de la presentación, de tal forma que todo lo relacionado con cómo funciona la interfaz queda separado del cómo representarlo en pantalla, debido a que en Android las actividades, fragmentos y demás elementos que componen a la vista están íntimamente acopladas tanto con la interfaz como con las mecánicas de acceso de datos. Esto a nivel estructural no es una buena práctica, ya que, al no separarse estas responsabilidades; el acoplamiento, la legibilidad y otros principios de calidad del software no se ven respetados. El MVP independiza la vista de la forma de conseguir esos datos y divide la aplicación en tres capas distintas, facilitando la realización de las pruebas unitarias. Estas capas son:(Sánchez 2015)

- **Presentador**: es el encargado de actuar de intermediario entre la vista y el modelo. Recupera los datos del modelo y se los devuelve formateados a la vista. Pero a diferencia del MVC, también decide qué ocurre cuando se interactúa con la vista y el presentador no tiene conciencia de ella.

- **Vista**: está compuesta por actividades, fragmentos, adaptadores y demás elementos que heredan de la clase View. La vista contendrá una referencia al presentador y es la encargada de crear el objeto presentador. Lo único que hará la vista será llamar a un método del presentador cada vez que se realice una acción sobre la interfaz, normalmente el pulsado de un botón, un elemento de una lista, etc. Esta no se relaciona con el modelo.

- **Modelo**: es la representación de los objetos del negocio que le provee los datos que necesita el presentador y los métodos para poder modificarlos. El modelo nunca interactúa con la vista

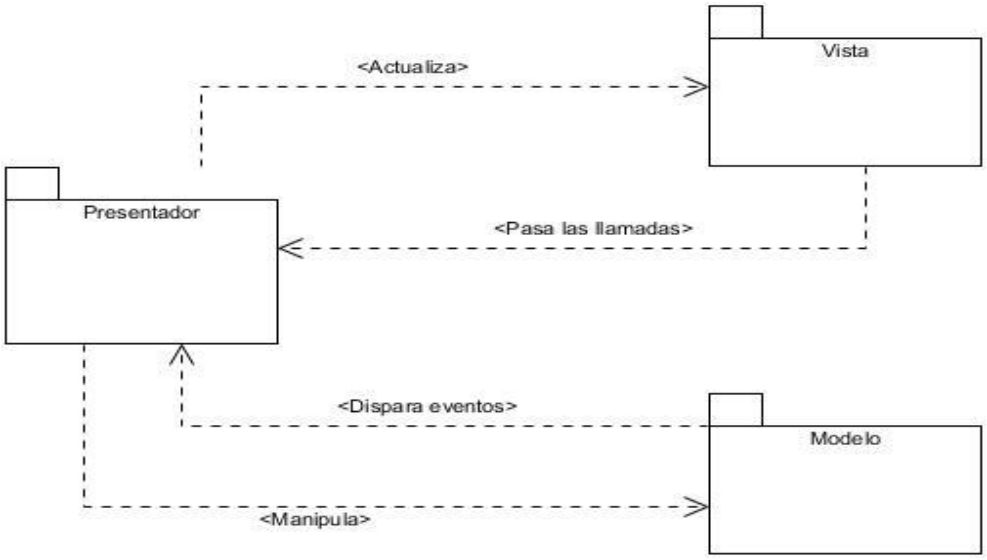


Ilustración 4: Diagrama de paquete de la arquitectura Modelo-Vista-Presentador (Sánchez 2015)

2.3.2 Diagrama de clases de diseño:

Un diagrama de clases de diseño representa las especificaciones de las clases e interfaces de software en una aplicación. Entre la información general se encuentran:(Bula 2022)

- Clases, asociaciones y atributos.
- Interfaces.
- Métodos.
- Navegabilidad.

A continuación, se muestra el diagrama de clase de diseño: Adicionar incidencia. En el apartado anexo puede visualizar los restantes diagramas de clases diseño.

Se define las clases de la siguiente forma:

Clases .java: Modelos y Presentador.

Clases. XML: Interfaz de cara al usuario.

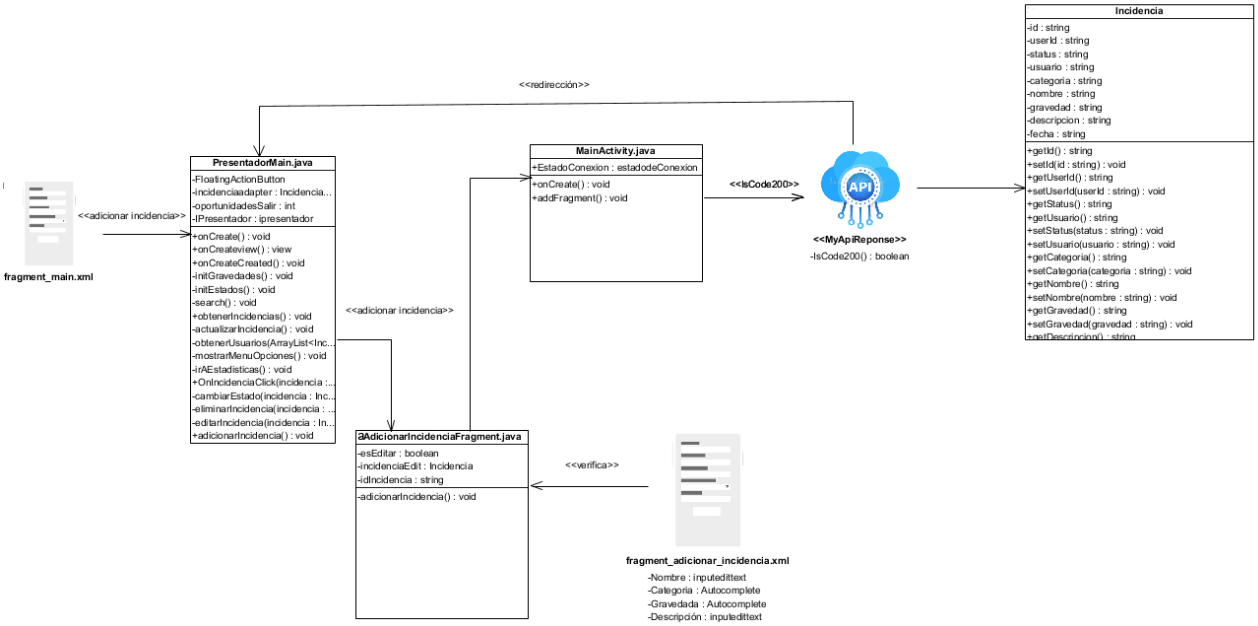


Ilustración 5:Diagrama de clases de diseño: Adicionar Incidencia (elaboración propia)

2.3.3 Diagramas de secuencia

Los diagramas de secuencia no son más que una sociedad de clases y otros elementos que colaborarán para realizar el comportamiento expresado, en este caso en las historias de usuario. Se realizan mediante las interacciones entre objetos, creando enlaces entre ellos y comunicando a través de mensajes. A continuación, se representan los diagramas de secuencias de la historias de usuario Adicionar incidencia, incorporados a la solución propuesta (Muñoz Ariza 2020). En el apartado anexo puede visualizar los restantes diagramas de secuencia.

Se define las clases de la siguiente forma:

Clases .java: Modelos y Presentador.

Clases. XML: Interfaz de cara al usuario.

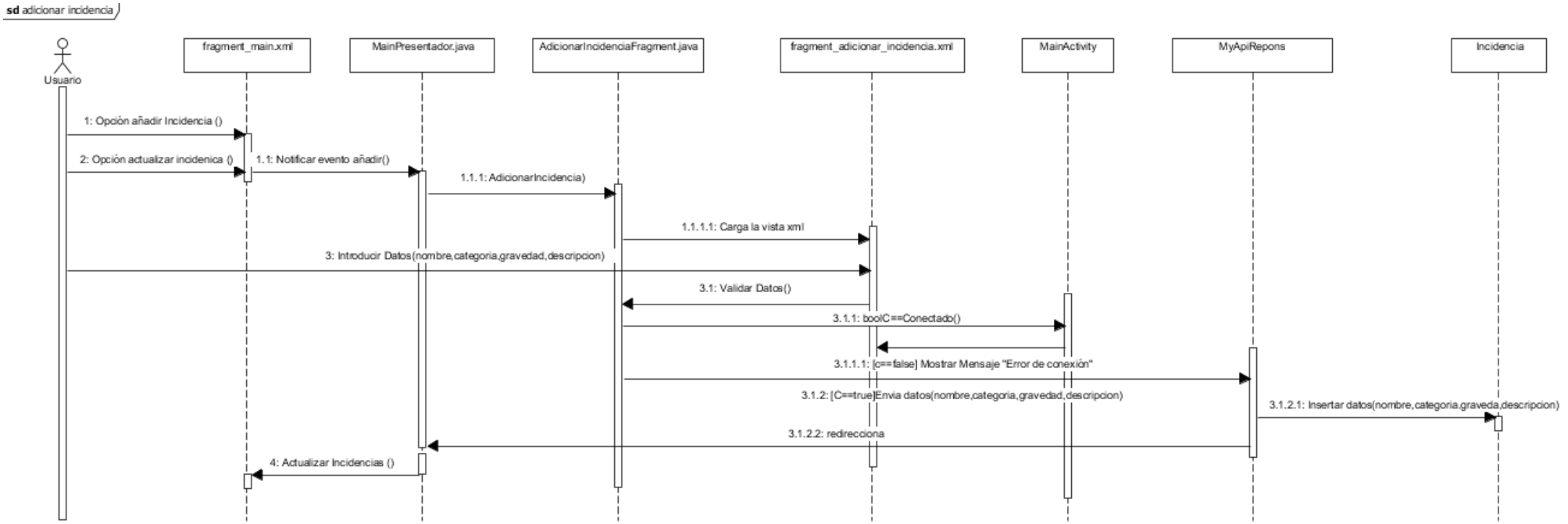


Ilustración 6:Diagrama de secuencia: Añadir incidencia (elaboración propia)

2.3.4 Patrones de diseño:

Un patrón de diseño es una descripción de la comunicación entre objetos y clases, personalizada para resolver un problema de diseño general en un contexto particular. Identifica clases, instancias, roles, colaboraciones y la distribución de responsabilidades (Larman, 2004).

Los patrones de diseño se caracterizan por:

- Representar soluciones técnicas a problemas concretos.
- Propiciar la reutilización.
- Representar problemas frecuentes

2.3.2.1 Descripción de los patrones de Principios Generales para Asignar Responsabilidades “Patrones GRASP”:

Del inglés *General Responsibility Assignment Software Patterns* (GRASP), estos patrones brindan principiosgenerales para asignar responsabilidades y en la realización de diagramas de interacción (Larman, 2004).

- **Experto:**

Básico para la asignación de responsabilidades. Consiste en la asignación de una responsabilidad a la clase que cuenta con la información necesaria para llevarla a cabo. La clase encargada de crear un objeto es la que conoce todos sus métodos y atributos. De este modo se obtiene mayor cohesión, seguridad y encapsulamiento. El cumplimiento de una responsabilidad requiere a menudo información distribuida en varias clases de objetos. Este patrón se evidencia en la clase Incidencia.java en el método Search() que es el encargado de a través de los parámetros recibidos verificar si la incidencia contiene la información deseada.

- **Bajo acoplamiento:**

El bajo acoplamiento soporta el diseño de clases más independientes y reutilizables, lo cual reduce el impacto de los cambios y acrecienta la oportunidad de una mayor productividad (Larman, 2004). Esto genera la menor afectación posible en la estructura del sistema, una clase con bajo o débil acoplamiento no depende de muchas otras. Este patrón se evidencia en

la case Usuario.java donde los modificadores de acceso son *private* y solo permite modifica o consultar el valor promedio de *setter* y *getters*.

- **Alta cohesión:**

Se caracteriza a las clases con responsabilidades estrechamente relacionadas que no realicen un trabajo enorme, clases con responsabilidades moderadas en un área funcional que colaboran con las otras parallevar a cabo las tareas. La información almacenada por cada clase debe ser la correspondiente a esta. Este patrón se evidencia RegistrarUsuarioFragment.java, pues lo que se desea es que la clase se especialice en una funcionalidad.

2.3.2.2 Descripción de los patrones de diseño Gang of Four "Patrones GOF":

Los patrones de diseño Gang of Four (GoF) son un conjunto de 23 patrones de diseño que están divididosen 3 categorías denominadas creacionales, estructurales y de comportamiento. Estos reducen la complejidad del sistema nombrando y definiendo abstracciones utilizando clases e instancias, constituyen una base reusable de experiencia para la construcción de diseño y proporcionan una reorganización de clases mediante jerarquías. En esta investigación se utilizan los siguientes(Pressman 2010):

- **Singleton:**

El objetivo de este patrón es asegurarse que una clase solo tiene una instancia y ofrecer un punto de acceso a ella. En la clase MyApiAdapter.java se utiliza para crear una única instancia de conexión con el API

- **Fachada:**

Permite proveer una interfaz unificada sencilla que haga de intermediaria entre un cliente y un subsistema o grupo de subsistema más complejo. Se utiliza ampliamente para hacer más fácil y entendible el trabajo con bibliotecas de software. Este patrón se evidencia en la clase interfaz IIncidenciaAdapter.java. permite el manejo de información con las clases incidencia Adapter e incidencia para realizar el método Search()

- **Adapter:**

Permite que objetos que se encuentran implementados puedan ser adaptados para ser reutilizados sin quese necesiten cambios en ellos, simplemente envolviéndolos con una capa alrededor que permitirá adaptarlos a la nueva interfaz deseado. Este patrón se evidencia en la clase IncidenciaAdapter.java para manejar la lista de Incidencias a utilizar en el desempeño del sistema.

2.3.5 Modelo de Datos:

Los modelos de datos aportan la base conceptual para diseñar aplicaciones que hacen un uso intensivo de datos. Un modelo de datos describe las representaciones lógicas y físicas de datos persistentes utilizados por la aplicación. Es usado para describir la representación lógica y física de la información persistente manejada por el sistema. (Rodríguez 2019)

A continuación, se presentan las entidades y campos fundamentales del modelo de datos propuesto:

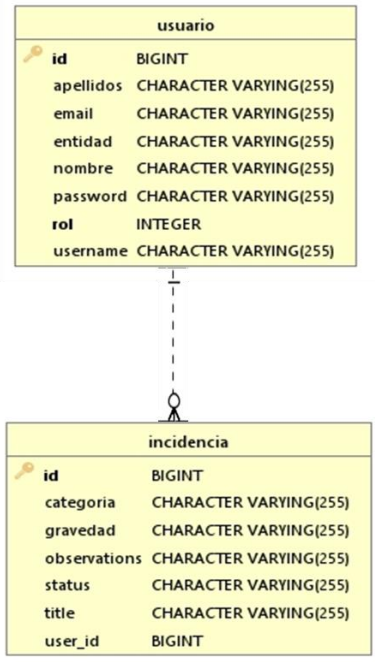


Ilustración 7:Diagrama entidad relación (elaboración propia)

Conclusiones del capítulo

Luego de transitar por las etapas de exploración e inicialización del proyecto descritas en la metodología AUP-UCI para el escenario 4 se concluye lo siguiente:

- La representación del modelo de dominio sirvió de base para la comprensión de las diferentes clases que integran el sistema.
- La definición de las historias de usuario junto a los prototipos de interfaz, permitió la comprensión de los requisitos funcionales.
- La definición de la arquitectura basada en Modelo-Vista-Presentador (MVP) junto al uso de patrones de diseño sentó las bases para la construcción de un software donde se apliquen buenas prácticas y las mejores soluciones.
- El tránsito por las fases iniciales de la metodología AUP-UCI y la generación de los artefactos correspondientes establecerá el punto de partida para la implementación de la propuesta de solución en las fases posteriores.

CAPÍTULO III: VALIDACIÓN DE LA PROPUESTA DE SOLUCIÓN AL PROBLEMA CIENÍIFICO

En el presente capítulo se exponen las especificaciones asociadas a la implementación de la aplicación. Se describen estándares de codificación utilizados, además la aplicación es sometida a un proceso de pruebas con el objetivo de verificar el cumplimiento de los requerimientos especificados anteriormente.

3.1 Implementación

3.1.1 Plan de Iteración:

El plan de iteración se divide en una serie de periodos de desarrollo de duración fija, que generalmente van de dos a seis semanas, llamados iteraciones. Planificar implica desglosar el trabajo que se va a realizar durante una iteración La planificación de la iteración se produce en el contexto de un área de proyecto. A continuación, se expone el plan de iteraciones. En este se muestra la cantidad de iteraciones en la que será desarrollada la aplicación Nova Incidencia, así como el tiempo asignado a cada iteración y las Historias de Usuario que serán implementadas en ellas. (IBM, 2021)

Iteración	Orden de las HU a implementar	Duración total	Fecha inicio	Fecha fin
1	HU1.Registrar Usuario HU2.Autenticar Usuario HU3.Adicionar Incidencia HU8.Listar Incidencia Usuario HU9.Listar Incidencia del Administrador	4 semanas	29/08//2022	19/09/2022
2	HU4. Mostrar detalles de la incidencia al usuario HU5. Mostrar detalles de la incidencia al administrador HU6.Editar Incidencia HU7.Cambiar Estado	4 semanas	19/09/2022	11/10/2022

	HU12.Actualizar			
3	HU10.Buscar Incidencia HU11.Filtrar Incidencia HU13.Mostrar estadística de las incidencias HU14.Eliminar Incidencia. HU15 Eliminar Múltiples Incidencias	2.5 semanas	12/10/2022	28/10/2022

Tabla 5: Plan de iteración (elaboración propia)

Iteración 1: Esta iteración tiene como objetivo la implementación de las historias de usuarios 1,2,3,8,9, esta son las bases para la implantación de las demás iteraciones. Se cuenta con 3 semanas para el cumplimento de estas tareas. Se obtiene como resultado una primera versión en la cual los usuarios se pueden registrar en el sistema, acceder al mismo, añadir incidencias y visualizar el listado de las mismas.

Iteración 2: Esta iteración tiene como objetivo la implementación de las historias de usuarios 4,5,6,7,12, las cuales hacen alusión a la realización de cambios en la incidencia. Se cuenta con 5 semanas para la implementación de esta iteración. Al finalizar, se obtiene una segunda versión juntando la iteración anterior con la presente.

Iteración 3: Esta iteración tiene como objetivo la implementación de las historias de usuario 10,11,13,14,15, las cuales hacen alusión al manejo de la información de una incidencia, así como eliminar las misma. Se cuenta con 2 semanas y medias para la realización de esta iteración, obteniendo como resultado una tercera versión que encapsule todas las iteraciones desarrolladas, dando paso a la realización de pruebas y cambios pertinentes.

3.1.2 Diagrama de componente

El diagrama de componentes ilustra la relación que existe entre componentes de software, así como la ubicación de cada uno de ellos dentro del módulo, como se implementan las clases en término de componentes. Describe también como se organizan los componentes de acuerdo con los mecanismos de estructuración y modularizarían el entorno de implementación y en el lenguaje de programación utilizado. Además, muestra las dependencias entre componentes (González Rodríguez 2017).

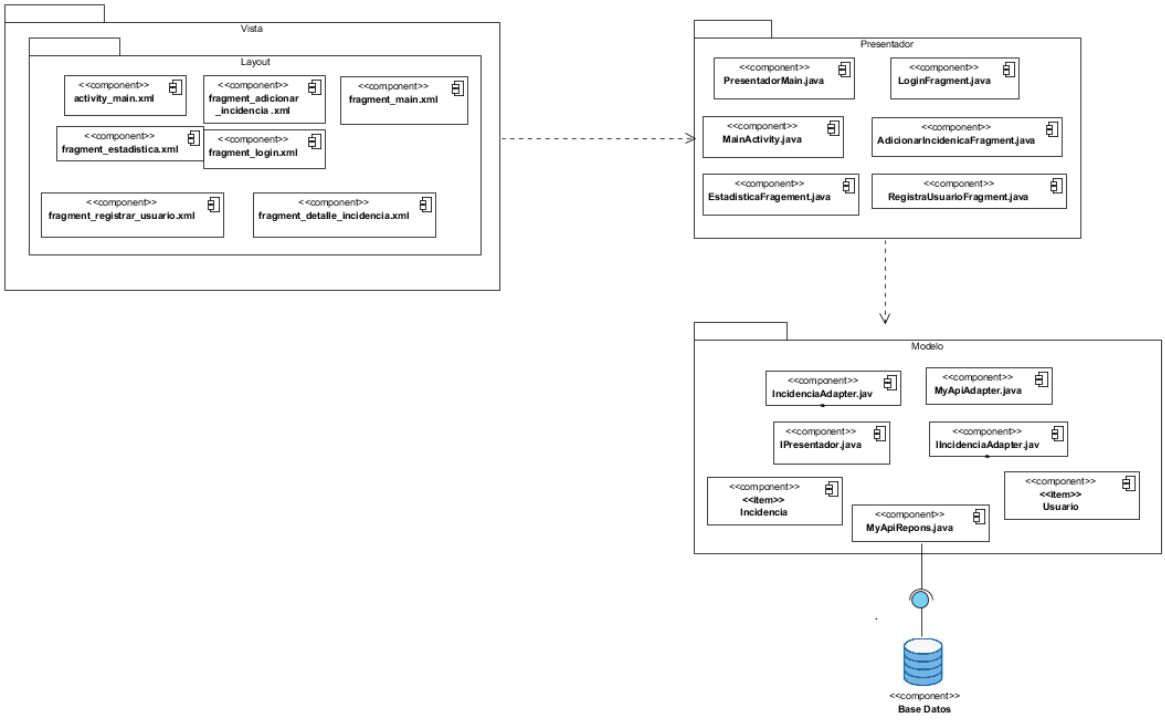


Ilustración 8: Diagrama de componente del sistema (elaboración propia)

3.1.3Diagrama de despliegue:

El diagrama de despliegue modela la topología del hardware sobre el que se ejecuta un sistema, el cual muestra la configuración de los nodos que participan en la ejecución de los componentes que residen en ellos. Además, representa el despliegue físico de un componente (Sarmiento, 2013). La aplicación es desplegada en un dispositivo móvil con sistema operativo Android que se conecta a través de una API para acceder a la base de datos del sistema.

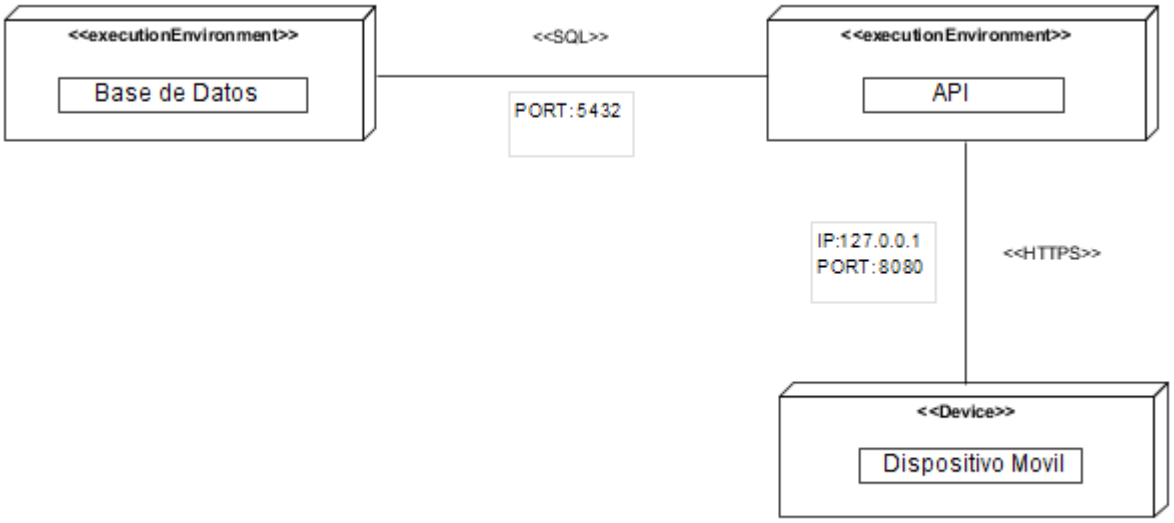


Ilustración 9:Diagrama de despliegue

3.1.4 Estándares de codificación

Los estándares de codificación son un conjunto de directrices, normas y reglamentos enfocados a la especificación de cómo debe escribirse el código fuente de la aplicación. Estos incluyen pautas sobre la nomenclatura de las variables, clases y paquetes, la correcta implementación del código, cómo escribir estructuras de control, entre otros aspectos.

El uso de estándares de codificación requiere que todos los desarrolladores escriban y mantengan el código en un formato común y consistente. Las reglas que gobiernen este formato facilitan la comunicación efectiva entre desarrolladores al proveer una base común a través de la cual entender varias unidades de código. (Maldonado,2015)

A continuación, se describen los estándares de codificación que regirán la implementación de las funcionalidades: (Arias Calleja, 2015)

- **Paquetes:** el nombre de los paquetes debe comenzar con letra minúscula. En caso de existir un cambio de palabra se utilizará el carácter guion bajo.
- **Clases:** los nombres de las clases deben ser sustantivos, se deben evitar el uso de abreviaturas o acrónimos. Es obligatorio empezar el nombre con letra mayúscula, de estar compuesto por más de una palabra se alternarán mayúsculas y minúsculas.

- **Métodos:** los métodos deben ser verbos. Es obligatorio empezar el nombre con letra minúscula, de estar compuesto por más de una palabra se alternarán mayúsculas y minúsculas.
- **Variables:** todas las variables deben comenzar con letra minúscula, al cambiar la palabra se usará letra mayúscula. Los nombres de las variables no deben empezar por el carácter guion bajo o el signo de dólar, aunque ambos son permitidos en el lenguaje Java.
- **Sentencias:** se proporcionará una sentencia por línea de código. Todo bloque de sentencia debe ser colocado entre llaves, aunque sea solamente una sentencia. Si una línea de código es demasiado larga por poseer expresiones complejas se debe dividir en varias líneas después de una coma u operador.
- **Comentarios:** es obligatorio proporcionar comentarios por cada clase o método creado, este comentario debe consistir en la descripción de la clase o método y su responsabilidad. Se deben proporcionar datos adicionales como parámetros, tipos de retorno y el número de la tarea de ingeniería para la que fue creado.

3.2 Pruebas al Sistema

La prueba de software es el proceso de evaluación y verificación de un producto o aplicación de software para comprobar que cumple con los requerimientos del usuario. Los beneficios de las pruebas incluyen la prevención de errores, la reducción de los costos de desarrollo y la mejora del rendimiento. (IMB, 2021)

3.2.1 Pruebas Funcionales:

El objetivo de las pruebas funcionales es validar si el comportamiento observado del software cumple o no con sus especificaciones por tanto se toma el punto de vista del usuario para probar las funciones con el ingreso de entradas y el examen de las salidas. La estructura interna del programa raramente es considerada para realizar estas pruebas funcionales, la especificación si se analiza para derivar los casos de prueba. Este tipo de pruebas deben ser ejecutadas idealmente por un equipo de pruebas ajeno al equipo de desarrollo. Los casos de

prueba a diseñar en este nivel de pruebas, deben cubrir los aspectos funcionales y no funcionales del sistema.(Guerrero 2019)

A continuación, se presentan el caso de prueba para la Historia de Usuario: Añadir Incidencia utilizando el método de caja negra bajo la técnica de partición de equivalencia[3]. El resto de los casos de pruebas podrán ser encontrados en los Anexos.

Escenario	Descripción	Nombre	Categoría	Gravedad	Descripción	Respuesta de la aplicación	Flujo central
		V	V	V	V		
EC1 Adicionar Incidencia.	Se añade la incidencia correctamente.	Error al abrir menú	Menú de inicio	Media	Cuando se pulsa el botón de menú no se ejecuta	Muestra el siguiente mensaje luego de registrada la incidencia “Incidencia añadida con éxito ".	La aplicación registra la incidencia, queda listada para el administrador y usuario.
EC1.2 Registrar incidencia con datos vacíos	Se insertan los datos y la aplicación verifica que no existan campos incorrectos.	V	I	V	V	La aplicación muestra un mensaje comunicando: “Campos vacíos”.	Introducir al menos un campo vacío.
		Error 2		Alta	Texto descriptivo		
		V	V	I	V		
		Gestor de archivo	Gestor de Archivo		Texto descriptivo		
		V	V	v	I		

[3] Partición de equivalencia es la porción del dominio de una entrada o una salida para la cual se asume que el comportamiento de un componente o sistema(Fuentes Ramírez, 2014)

		Office Libre	Ofimática	Media			

Tabla 6:Caso de prueba de funcionalidad a la historia de usuario añadir incidencia (elaboración propia)

3.2.3 Pruebas Unitarias:

Las pruebas unitarias tienen como objetivo, permitirle verificar al desarrollador que los componentes unitarios estén codificados bajo condiciones de robustez, soportando el ingreso de datos erróneos o inesperados y demostrando así la capacidad de tratar errores de manera controlada. Para que una prueba de unidad sea correcta debe ser independiente, completa, reutilizable y automatizable donde no se requiera de una intervención manual. (Mañas,2016)

Para la implementación de esta prueba se utilizará la herramienta JUnit 4 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 porciento de satisfacción de los casos de prueba.

A continuación, se ejemplifican las pruebas realizadas a las clases **AdicionarIncidenciaFragment** e **ResgitrarUsuarioFragment** mediante las clases **AdicionarIncidenciaFragmentTest** e **RegitrarUsuarioFragmentTest**, (Ver Anexo) donde se obtienen resultados satisfactorios:

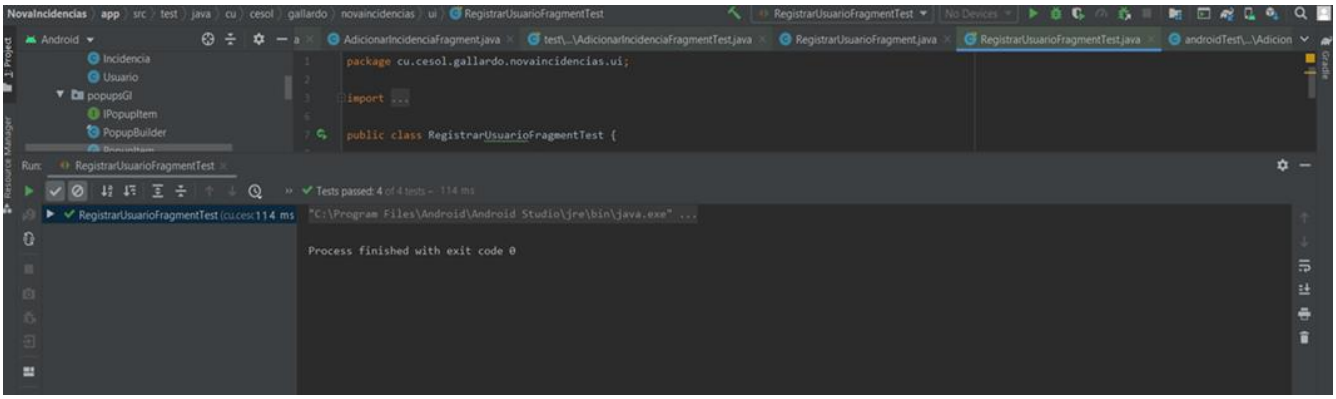


Ilustración 10:Resultado de la prueba JUnit de la clase RegistrarUsuariofragment.java (elaboración propia)

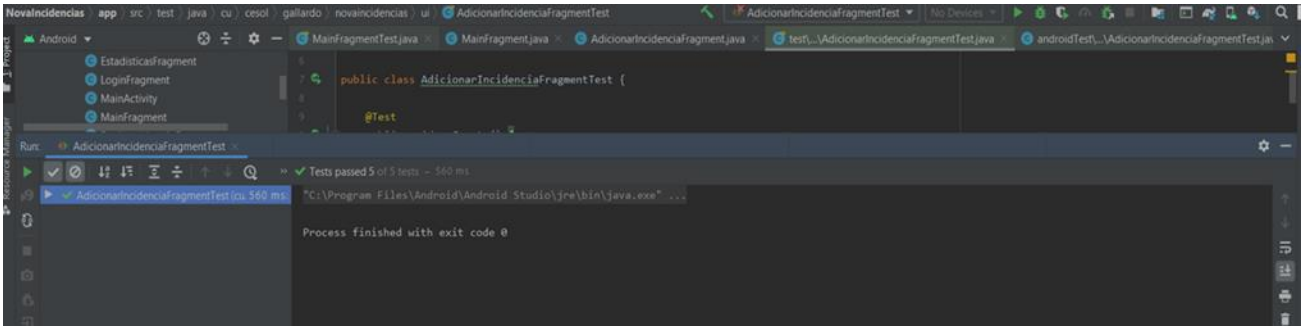


Ilustración 11: Resultado de la prueba JUnit de la clase AdicionarIncidenciafragment.java (elaboración propia)

Determinando así que todos sus componentes individualmente funcionan correctamente. Se procede a realizar las pruebas de aceptación con el cliente.

3.2.2 Pruebas de Aceptación:

Los desarrolladores escriben las pruebas unitarias para determinar si el código está funcionando correctamente. Los clientes escriben las pruebas de aceptación para determinar si el sistema está cumpliendo con sus propósitos. Las pruebas de aceptación representan los intereses del cliente y le brindan confianza que la aplicación cuenta con las funcionalidades requeridas y se comportan correctamente. Si la entrega es lo suficientemente buena, el cliente puede entonces aceptarla para su uso (Sommerville ,2005)

Las pruebas de aceptación cumplen tres funciones para el equipo de desarrollo de software:

1. Capturan los requerimientos del cliente de forma directa y verificable, además, miden que también el sistema cumple con esos requerimientos.
2. Exponen problemas que las pruebas unitarias pasan por alto.
3. Proveen una definición confeccionada de cómo hacer las actividades.

Las pruebas de aceptación son consideradas como pruebas de *caja negra*, debido a las características de la aplicación, es necesario realizarlas. Estas son creadas en base a las historias de usuario, en cada ciclo de la iteración del desarrollo, por lo que una historia de usuario no se puede considerar terminada hasta tanto pase correctamente todas las pruebas de aceptación.

Existen 2 tipos de pruebas de aceptación; las pruebas alfa y las pruebas beta que se encargan de descubrir errores que sólo el usuario final es capaz de detectar. Se desarrollaron pruebas alfa por los desarrolladores, en un grupo del docente 3, con un grupo de usuarios de Nova, del centro CESOL.

Durante la etapa de prueba se definen dos tareas fundamentales: la recolección de las pruebas presentes en la documentación y su ejecución para obtener posibles errores. Cada caso de prueba fue representado en forma de tabla con los siguientes campos: identificador, número y nombre de la historia a la que pertenece, fecha en que fue creado, fecha en que se ejecutó, si la prueba fue exitosa, entradas y resultados esperados. A continuación, se presenta 2 de las pruebas de aceptación generadas para la historia de usuario “Listar Incidencias” y “Adicionar Incidencia”. La confirmación por parte del cliente mediante la carta de aceptación podrá ser encontrada en los Anexo.

CASO DE PRUEBA DE ACEPTACIÓN	
CÓDIGO: CPA_6.1	**HISTORIA DE USUARIO:** HU_3
DESCRIPCIÓN: Caso de prueba para la funcionalidad de adicionar incidencias.	
CONDICIONES DE EJECUCIÓN: - El usuario debe estar autenticado en el sistema.	

ENTRADA/PASOS DE EJECUCIÓN: Una vez autenticado el usuario selecciona el botón flotante “+” y procede a rellenar los campos requeridos.
RESULTADO ESPERADO: Incidencia añadida al sistema Al completar la acción se muestra un mensaje “Incidencias Registrada con éxito”.
EVALUACIÓN DE LA PRUEBA: SATISFACTORIA.

Tabla 7: Caso de prueba de aceptación, historia de usuario eliminar incidencia (elaboración propia)

CASO DE PRUEBA DE ACEPTACIÓN	
CÓDIGO: CPA_6.1	**HISTORIA DE USUARIO:** HU_8
DESCRIPCIÓN: Caso de prueba para la funcionalidad de listar incidencias.	
CONDICIONES DE EJECUCIÓN: - Estar autenticado en el sistema. - El usuario debe poseer rol administrador. - Debe existir en el sistema al menos una incidencia registrada por un usuario.	
ENTRADA/PASOS DE EJECUCIÓN: Una vez autenticado el administrador visualiza el listado de Incidencias realizadas por todos los usuarios del sistema.	

RESULTADO ESPERADO: Aparece un listado con todas las solicitudes de incidencias. En el listado se muestran los siguientes detalles: - Nombre de la incidencia. - ID. - Estado. - Usuario. - Entidad.
EVALUACIÓN DE LA PRUEBA: SATISFACTORIA.

Tabla 8:Caso de prueba de aceptación, historia de usuario listar incidencia (elaboración propia)

3.3 Resultados de las Pruebas

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, ejecutándose al término de cada una de ellas pruebas de regresión, con el objetivo de asegurar que, al resolverse las no conformidades detectadas, estas no introdujeran nuevos errores en la solución.

En la primera iteración se detectaron 5 NCS y 3 NCNS. Las mismas fueron resueltas satisfactoriamente en la misma iteración.

No Conformidades Significativas:

1. La aplicación mostró la información de una incidencia distinta a la seleccionada en el listado correspondiente.
2. La aplicación no mostró la información acerca de la aplicación y sus desarrolladores al seleccionarse la opción “Acerca de”.

3. La aplicación no mostró ninguna información de fecha o entidad del usuario al ofrecer los detalles de una incidencia.
4. La aplicación se detuvo al intentar registrar una incidencia en el momento en que el dispositivo perdió la conexión.
5. El usuario puede eliminar incidencia aun cuando esta está siendo procesada.

No Conformidades No Significativas:

1. La aplicación tiene faltas de ortografía en el texto mostrado en la vista “Cerrar Cesión”.
2. En el momento que realizas una incidencia, para añadir la misma, debes ocultar el teclado móvil para poder acceder al botón “Adicionar”.
3. La aplicación tiene faltas de ortografía en los textos de selección para registrar Usuario.

En la segunda iteración se detectaron 2 NCS y 1 NCNS, siendo solucionadas de la misma forma que las anteriores.

No Conformidades Significativas:

1. La aplicación permite a el administrador volver a colocar el estado “Enviado” luego de que este fuese cambiado a “Procesado” o “Listo”.
2. La aplicación no mostró ningún resultado al ejecutar una búsqueda por usuario en el listado de incidencia registradas, conociéndose que existen evaluaciones que coinciden con el criterio de búsqueda introducido.

No Conformidades No Significativas:

1. La aplicación tiene faltas de ortografía en los textos de selección de la interfaz para añadir incidencia.

En la tercera iteración no se detectaron **NCS** ni **NCNS**, por lo que se demostró que la aplicación cumple con los requisitos funcionales establecidos y fue considerada concluida. A continuación, se muestra un gráfico con un resumen de los resultados obtenidos tras la realización de las pruebas:

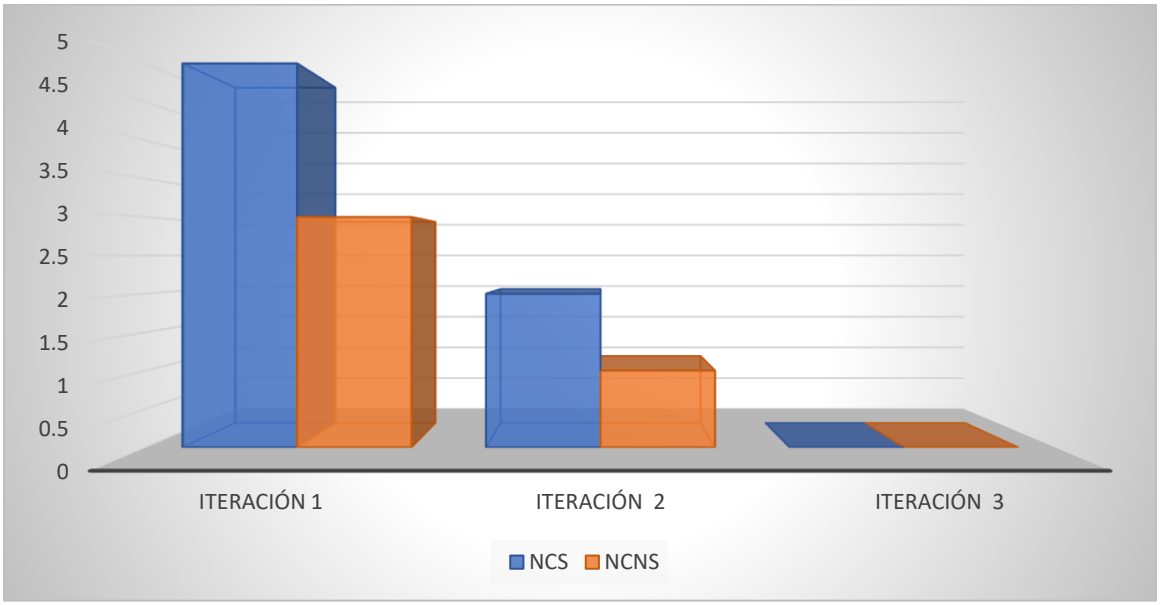


Ilustración 12: Grafo No conformidad (elaboración propia)

Conclusiones del capítulo

Luego de transitar por las fases de producto, estabilización y prueba-corrección definidas en la metodología AUP-UCI se concluye lo siguiente:

- El desglose de las historias de usuario en tareas de ingeniería permitió describir la manera en que serán implementados los requisitos funcionales del sistema.
- El uso de estándares de codificación permitió guiar el proceso de implementación, obteniendo un código más estructurado y entendible.
- El uso en conjunto de las pruebas de funcionalidad, unitarias y aceptación permitió la identificación de no conformidades en la propuesta de solución. Las deficiencias encontradas mediante la ejecución de dichas pruebas fueron corregidas satisfactoriamente.

CONCLUSIONES FINALES

Atendiendo a tareas de investigación propuestas en la investigación se concluye lo siguiente:

- El estudio de las aplicaciones de reporte de Incidencias evidenció la ausencia de soluciones aplicables al problema de la investigación, sin embargo, este análisis sirvió de base para identificar las posibles funcionalidades a implementar en la propuesta de solución.
- El estudio de las herramientas y tecnologías permitió definir la propuesta de solución: una aplicación móvil nativa para dispositivos con sistema operativo Android que servirá para el reporte de incidencias detectadas en el uso del sistema operativo Nova.
- Se implementó la aplicación móvil que permite el reporte de incidencias detectadas en el uso del sistema operativo Nova desde la plataforma móvil Android, permitiendo la retroalimentación entre el equipo de desarrollo y los usuarios de Nova.
- Las pruebas realizadas al sistema permitieron detectar un total de 11 no conformidades, donde 7 fueron no conformidades significativas y 4 no conformidades no significativas; todas las no conformidades fueron resueltas en tres iteraciones.

RECOMENDACIONES

El objetivo general trazado en el presente trabajo de diploma fue alcanzado; no obstante, se recomienda:

- Desarrollar versiones de la aplicación “Nova Incidencia” para otros sistemas operativos móviles como iOS.
- Integrar la aplicación con una plataforma web que permita el aumento de funcionalidades para posteriores evoluciones del sistema.

REFERENCIAS BIBLIOGRÁFICAS

ALVAREZ, O.D.G., LAYEDRA, N. y RAMOS, V., 2022. Análisis comparativo de Patrones de Diseño de Software. *Polo del Conocimiento*, vol. 7, no. 7, pp. 2146. ISSN 2550-682X.

ARIAS CALLEJA, M., 2015. Carmen. Estándares de codificación. [en línea]. [Consulta: 24 noviembre 2022]. Disponible en: https://docplayer.es/11247774-Carmen-estandares-de-codificacion-manuel-arias-calleja.html.

BORGES, E., 2019. Servidor Base de Datos: ¿Qué es? Funciones, Tipos y Ejemplos. *Infranetworking* [en línea]. [Consulta: 23 mayo 2022]. Disponible en: https://blog.infranetworking.com/servidor-base-de-datos/.

BULA, C., 2022. DIAGRAMA DE CLASES EN UML. [en línea], Disponible en: https://d1wqtxts1xzle7.cloudfront.net/37472390/31096724-Diagrama-de-Clases-en-UML-with-cover-page-v2.pdf?Expires=1667436001&Signature=SznoUd1a8dtO7ki57BcwXmfaEFb4gMfhVH5nCvbfmdl7VTrgxmCjitc92oOLe4T-SHrCJhKLH1hb9NliibQEQIhqunuXAcM1XMYolKtJZm11ZeR162cc7DxJ5psj0CuMRjcXxjFBwnufW4BVXYEX50EBDxegIrTRv0xaxIFxekckhOJ6yg-eVVVrUEOfNNWQKoxx~RPNWLklSL7DBqREEzn7C7xdwYY68jYFO05WzJxH8U5BbyB~ULEnDY07nmHnHZVQ292Rj6tNoT7-~9vHG1dZBM-l1J9c3p8omjaW7Vuyjt0mpetWp3fq9L3Zy2RWME2MqS0MJD1NdqqW5nTwaQ__&Key-Pair-Id=APKAJLOHF5GGSLRBV4ZA.

BURGUÉS, J.E.G., 2018. *Aprende a Modelar Aplicaciones con UML- Tercera Edición*. S.l.: IT Campus Academy. ISBN 978-1-985133-43-3.

CASTRO, M.M.A. y JIMENEZ, A.R., 2020b. La importancia de la calidad de la distribución GNU/Linux Nova para un desarrollo económico y social sostenible del país. *Serie Científica de la Universidad de las Ciencias Informáticas*, vol. 13, no. 2, pp. 108-118. ISSN 2306-2495.

Definiciones de evento, incidencia o no conformidad en ISO 27001. *Escuela Europea de Excelencia* [en línea], 2020b. [Consulta: 15 junio 2022]. Disponible en: https://www.escuelaeuropeaexcelencia.com/2020/04/definiciones-de-evento-incidencia-o-no-conformidad-en-iso-27001/.

DESOFT, 2022. Presenta Desoft nueva aplicación para gestión de la información. [en línea]. [Consulta: 1 noviembre 2022]. Disponible en: https://www.radiohc.cu/noticias/ciencias/155731-presenta-desoft-nueva-aplicacion-para-gestion-de-la-informacion.

Entrar - Sistema de Gestión de Incidencias. [en línea], [sin fecha]. [Consulta: 21 octubre 2022]. Disponible en: https://incidencias.uci.cu/otrs/customer.pl.

FUENTES RAMIREZ, 2014. *Sistema de gestión de Pruebas de Caja Negra mediante la Técnica de particiones equivalentes* [en línea]. 2014. S.l.: s.n. Disponible en: http://roa.ult.edu.cu/jspui/handle/123456789/2537.

FULLAUDITS, 2022. Checklist Software - Dashboarding & Auditorias | Full Audits. [en línea]. [Consulta: 15 septiembre 2022]. Disponible en: https://fullaudits.com/.

GIRONÉS, J.T., 2019b. *El gran libro de Android*. S.l.: Alpha Editorial. ISBN 978-958-778-545-6.

GONZÁLEZ RODRÍGUEZ, A., 2017. *Subsistema de gestión de perfil de usuario para el buscador Orión.* [en línea]. S.l.: s.n. Disponible en: https://repositorio.uci.cu/jspui/handle/123456789/7963.

GROUSSARD, T., 2012. *JAVA 7: Los fundamentos del lenguaje Java*. S.l.: Ediciones ENI. ISBN 978-2-7460-7318-0.

GUERRERO, H., 2019. *Gestión de las Pruebas Funcionales*. 2019. S.l.: s.n.

HERNANDEZ RODRIGUEZ., 2010. *GUÍA PARA LA MIGRACIÓN A SOFTWARE LIBRE EN LA EMPRESA DIVEP USANDO LA DISTRIBUCIÓN NOVA*. S.l.: s.n.

IBM, 2022. ¿Qué es la respuesta a incidentes? | IBM. [en línea]. [Consulta: 14 septiembre 2022]. Disponible en: https://www.ibm.com/cl-es/topics/incident-response.

IBM Documentation. [en línea], 2021a. [Consulta: 14 septiembre 2022]. Disponible en: https://prod.ibmdocs-production-dal-6099123ce774e592a519d7c33db8265e-0000.us-south.containers.appdomain.cloud/docs/es/guardium/11.1?topic=protect-incident-management.

IBM Documentation. [en línea], 2021b. [Consulta: 6 noviembre 2022]. Disponible en: https://prod.ibmdocs-production-dal-6099123ce774e592a519d7c33db8265e-0000.us-south.containers.appdomain.cloud/docs/es/elm/6.0.4?topic=development-planning-iteration.

IBM Documentation. [en línea], 2021c. [Consulta: 14 septiembre 2022]. Disponible en: https://prod.ibmdocs-production-dal-6099123ce774e592a519d7c33db8265e-0000.us-south.containers.appdomain.cloud/docs/es/guardium/11.1?topic=protect-incident-management.

IBM Documentation. [en línea], 2021d. [Consulta: 15 septiembre 2022]. Disponible en: https://prod.ibmdocs-production-dal-6099123ce774e592a519d7c33db8265e-0000.us-south.containers.appdomain.cloud/docs/es/mhs-and-em/7.6.1?topic=application-working-incidents.

IBM Documentation. [en línea], 2021e. [Consulta: 15 septiembre 2022]. Disponible en: https://prod.ibmdocs-production-dal-6099123ce774e592a519d7c33db8265e-0000.us-south.containers.appdomain.cloud/docs/es/mhs-and-em/7.6.1?topic=module-incidents-application.

IBM Documentation. [en línea], 2022a. [Consulta: 15 septiembre 2022]. Disponible en: https://prod.ibmdocs-production-dal-6099123ce774e592a519d7c33db8265e-0000.us-south.containers.appdomain.cloud/docs/es/control-desk/7.6.1.1?topic=incidents-resolving-incident.

IMB, [sin fecha]. ¿Qué son las pruebas de software y cómo funcionan? | IBM. [en línea]. [Consulta: 6 noviembre 2022]. Disponible en: https://www.ibm.com/cl-es/topics/software-testing.

Informática Básica: Sistemas operativos para dispositivos móviles. *GCFGlobal.org* [en línea], 2022. [Consulta: 4 octubre 2022]. Disponible en: https://edu.gcfglobal.org/es/informatica-basica/sistemas-operativos-para-dispositivos-moviles/1/.

Ingenieria del Software 7ma. Ed. - Ian Sommerville.pdf, [sin fecha]. S.l.: s.n.

KIZEO, 2022. Kizeo Forms. [en línea]. Disponible en: https://www.kizeo-forms.com.

KOUSEN, 2016. *Gradle Recipes for Android* [en línea]. S.l.: s.n. Disponible en: https://books.google.es/books?hl=es&lr=&id=10pODAAAQBAJ&oi=fnd&pg=PR2&dq=gradle+android&ots=qMJ6nEEKdG&sig=Z9FWw5UbxYQjoc1p71qe356dVfk#v=onepage&q=gradle%20android&f=false.

KUBESSI PÉREZ, M., 2015. Análisis de competencias transversales referido al modelo educativo de ingeniería areonaútica en la universidad politécnica de Valencia. ,

LARMAN, C., 2004. *Applying UML and patterns An introduction to object-oriented analysis and design and the unified process* [en línea]. S.l.: s.n. Disponible en: https://personal.utdallas.edu/~chung/SP/applying-uml-and-patterns.pdf.

LOU, T., 2016. *A comparison of Android Native App Architecture MVC, MVP and MVVM* [en línea]. S.l.: Eindhoven University of Technology. Disponible en: https://pure.tue.nl/ws/portalfiles/portal/48628529/Lou_2016.pdf.

LUCIANA, VEGETTI y MARCISZACK, 2018. Un Modelo Conceptual para la Especificación y Trazabilidad de Requerimientos Funcionales. [en línea], Disponible en: https://www.profesores.frc.utn.edu.ar/sistemas/ssl/Marciszack/Documentos/2017/Actas-CONAIISI-2017_Marciszack_1%20Proyecto%20Integrador.pdf.

LUIS E, M., 2021b. *APLICACIÓN MÓVIL PARA LA GESTIÓN DE INCIDENCIAS PARA ESTUDIANTES Y PROFESORES DE LA UNIVERSIDAD APEC, AGOSTO 2021* [en línea]. S.l.: s.n. Disponible en: https://bibliotecaunapec.blob.core.windows.net/tesis/TESIS_CI_ISC_06_2021_ET220006.pdf.

MALDONADO, R.I.Z., [sin fecha]. COMPARATIVA SINTÁCTICA ENTRE LOS LENGUAJES DE PROGRAMACIÓN JAVA Y GROOVY. , pp. 149.

MAÑAS, J.A., 2016. *Pruebas unitarias* [en línea]. 2016. S.l.: s.n. Disponible en: http://www.dit.upm.es/~pepe/doc/adsw/base/junit.pdf.

MENZINSKY, A., LÓPEZ, G., PALACIO, J. y SOBRINO, M., 2022b. *Historias de Usuario Ingeniería de Requisitos Ágil Versión 3.01 – Agosto 2022* [en línea]. S.l.: s.n. Disponible en: https://scrummanager.net/files/scrum_manager_historias_usuario.pdf.

Metodologia De Roger Pressman. *calameo.com* [en línea], 2022. [Consulta: 17 septiembre 2022]. Disponible en: https://www.calameo.com/read/00570389414c8ee17fd43.

MUÑOZ ARIZA, 2020. Simulación de Diagramas de Secuencia con la herramienta de modelado USE. [en línea], Disponible en: https://hdl.handle.net/10630/19207.

NOVA, 2022. Nova | Portal web. [en línea]. [Consulta: 4 mayo 2022]. Disponible en: https://www.nova.cu/.

OFOEDA, J., BOATENG, R. y EFFAH, J., 2019. Application Programming Interface (API) Research: A Review of the Past to Inform the Future. *International Journal of Enterprise Information Systems (IJEIS)*, vol. 15, no. 3, pp. 76-95. ISSN 1548-1115. DOI 10.4018/IJEIS.2019070105.

POLANCO, K.M. y TAIBO, J.L.B., 2011. "Android" el sistema operativo de Google para dispositivos móviles. *Negotium: revista de ciencias gerenciales*, vol. 7, no. 19, pp. 79-96. ISSN 1856-1810.

PRESSMAN, R.S., 2010b. *Ingenieria del Software. Un Enfoque Practico 7ma edición* [en línea]. septima. S.l.: McGRAW-HILL INTERAMERICANA EDITORES, S.A. DE C.V. 978-607-15-0314–5. Disponible en: http://cotana.informatica.edu.bo/downloads/ld-Ingenieria.de.software.enfoque.practico.7ed.Pressman.PDF.

PRIETO CARMONA, Y., & FUENTES BAUTA, J. ., 2019b. Métricas de usabilidad para la detección de errores en el diseño y funcionamiento del entorno de escritorio de la Distribución de GNU/LINUX NOVA. Serie Científica De La Universidad De Las Ciencias Informáticas, 11(1), 45-55. Recuperado a partir de https://publicaciones.uci.cu/index.php/serie/article/view/177. ,

PUENTE FUENTES, E., FUENTES, E.P., BAHR, D.H., FEDOROVICH, M.H. y SÁNCHEZ, J.E.T., 2012. Nova como sistema operativo embebido para hardware cubano. *Revista Cubana de Ciencias Informáticas* [en línea], vol. 6, no. 1. ISSN 1994-1536. Disponible en: https://rcci.uci.cu/?journal=rcci&page=article&op=view&path[]=173.

RODRIGUEZ, toledo, 2019. Módulo para la gestión de copias de seguridad en Nova 360. [en línea], Disponible en: https://repositorio.uci.cu/jspui/handle/123456789/10098.

RODRÍGUEZ, T., [sin fecha]. *Metodología de desarrollo para la Actividad productiva de la UCI*. S.l.: s.n.

SALINA, I., 2015. *USOS Y TIPOS DE APLICACIONES MÓVILES* [en línea]. S.l.: s.n. Disponible en: https://www.academia.edu/13777638/USOS_Y_TIPOS_DE_APLICACIONES_M%C3%93VILES.

SÁNCHEZ, D., 2015. Patrón Modelo - Vista - Presentador (MVP). [en línea], Disponible en: https://dialnet.unirioja.es/servlet/articulo?codigo=5297785.

SARMIENTO, F., Johana, 2013. Visión General de los Diagramas de Despliegue. *UML* [en línea]. [Consulta: 3 noviembre 2022]. Disponible en: http://umldiagramadespliegue.blogspot.com/.

SOMMERVILLE I, 2011. *Ingeniería de Software 7ma edición.* [en línea]. S.l.: s.n. Disponible en: http://zeus.inf.ucv.cl/~bcrawford/Modelado%20UML/Ingenieria%20del%20Software%207ma.%20Ed.%2 0-%20Ian%20Sommerville.pdf.

TEAM, A., 2022. Metodología ITIL: gestión de incidencias y objetivos. [en línea]. [Consulta: 15 junio 2022]. Disponible en: https://www.ambit-bst.com/blog/metodología-itil-gestión-de-incidencias-y-objetivos.

T.LOU, 2016. A comparison of Android Native App Architecture. *Eindhoven University of Technology research portal* [en línea]. [Consulta: 20 septiembre 2022]. Disponible en: https://research.tue.nl/en/studentTheses/a-comparison-of-android-native-app-architecture.

Todo lo que necesitas saber sobre el diagrama de caso de uso. *Venngage Blog* [en línea], 2022. [Consulta: 21 septiembre 2022]. Disponible en: https://es.venngage.com/blog/diagrama-de-caso-de-uso/.

tuexpertomovil.com | todo sobre móviles. [en línea], [sin fecha]. [Consulta: 1 noviembre 2022]. Disponible en: https://www.tuexpertomovil.com/.

UCI y CESOL, 2022. Centro de Software Libre (CESOL) | Universidad de las Ciencias Informáticas. [en línea]. [Consulta: 4 mayo 2022]. Disponible en: https://www.uci.cu/investigacion-y-desarrollo/centros-de-desarrollo/centro-de-software-libre-cesol.

UCM-Oficina de Software Libre. [en línea], 2022. [Consulta: 13 junio 2022]. Disponible en: https://www.ucm.es/oficina-de-software-libre/software-libre.

UUUPSAPP!, 2022. UUUPSAPP! APP para la gestión de incidencias. [en línea]. [Consulta: 15 junio 2022]. Disponible en: http://www.uuupsapp.com/post/13/uuupsapp-nueva-app-para-la-gestion-de-incidencias/.

VALLEJO BERMEO, J.A., 2014. Estudio comparativo de los patrones para interfaces de usuario MVVM y MVC aplicado el desarrollo del sitio de gestión de ventas para Vidrialum. En: Accepted: 2014-12-01T17:41:30Z [en línea], [Consulta: 17 septiembre 2022]. Disponible en: http://dspace.espoch.edu.ec/handle/123456789/3559.

VARIOS | 0 |, P. por M.S.| 29 marzo 2021 |, [sin fecha]. Cómo quitar la notificación Servicios y comentarios en Xiaomi. [en línea]. [Consulta: 1 noviembre 2022]. Disponible en: https://www.tuexpertomovil.com/2021/03/29/como-quitar-la-notificacion-servicios-y-comentarios-en-xiaomi/.

WAUTELET, Y., HENG, S., KOLP, M. y MIRBEL, I., 2013. *Unifying and Extending User Story Models* [en línea]. S.l.: LNISA. Disponible en: https://link.springer.com/chapter/10.1007/978-3-319-07881-6_15.

WISNUADHI, B., MUNAWAR, G. y UJANG, W., 2020. Performance Comparison of Native Android Application on MVP and MVVM. *Atlantis press*,

XETID, 2022. Portal Web de la XETID. *Portal Web de la XETID* [en línea]. [Consulta: 1 noviembre 2022]. Disponible en: https://www.xetid.cu/es/novedades/software-cubano-facilita-gestion-de-negocios-empresas-y-al-sector-privado.

XIAOMI, 2022. Xiaomi Privacy. [en línea]. [Consulta: 24 noviembre 2022]. Disponible en: https://privacy.mi.com/services-feedback/en_US/.

[En línea]. [Consulta: 15 de octubre 2022]. Disponible en: https://www.java.com/es/about/whatis_java.jsp.

Android developers, 2016. Download Android Studio and SDK Tools | Android Studio. [En línea]. [Consulta: 8 octubre 2022]. Disponible en: https://developer.android.com/studio/index.html?hl=es.

Barrientos, Pablo Andrés (25 de abril de 2014). Enfoque para pruebas de unidad basado en la generación aleatoria de objetos. [En línea]. [Consulta: 8 octubre 2022]. Disponible en: http://sedici.unlp.edu.ar/handle/10915/34969

Grant Kevin, Haseman Chris, 2014. Beginning Android Programming. Develop and Design. Printed and bound in the United States of America. ISBN-13: 978-0-321-95656-9

Borges, E. (2019, marzo 17). Servidor Base de Datos: ¿Qué es? Funciones, Tipos y Ejemplos.

JUnit, 2016. JUnit - About. [En línea]. [Consulta: 27 octubre 2022]. Disponible en: http://junit.org/junit4/.

Suárez Y., 2015. Entorno de Desarrollo Integrado. Qué es un IDE? http://deprogramacion.cubava.cu/2016/02/01/que-es-un-ide/.

Microsoft, 2016. Microsoft. MSDN. Revisiones de código y estándares de codificación. [En línea]. [Consulta:28 junio 2022]. Disponible en: http://msdn.microsoft.com/eses/library/aa291591(v=vs.71).aspx.

Feal Delgado W., Alvarez Acosta H., Miguel Canosa Reyes R., 2014. Estrategia de migración al software libre en la Universidad de Cienfuegos. Universidad y Sociedad. [En línea] [Consulta: 15 junio 2022]. Disponible en: http://rus.ucf.edu.cu

Montano José L., 2015. La migración hacia software libre en Cuba: complejo conjunto de factores sociales y tecnológicos en el camino de la soberanía nacional [En línea] [Consulta: 27 junio 2022].

Carbonell F.L, 2016. Resolución la Implementación de los Lineamientos de la Política Económica y Social aprobados en el 6to. Congreso y su actualización para el período 2016. [En línea]. [Consulta: 27 junio 2022]. Disponible en: www.granma.cu/ séptimo-congreso-del-pcc

Lineamientos de la Política Económica y Social del Partido y la Revolución, 2011.

Villazón Y.P & otros, 2014. Buenas Prácticas para la Migración a Código Abierto. La Habana, Universidadde las Ciencias Informáticas.

Samón R.P, Villazón Y.P, Abad A.M. ,2009. Guía Cubana de Migración a Software Libre

Santiago, Raúl, Susana Trabaldo, Mercedes Kamijo, y Álvaro Fernández. 2015. *Mobile learning: nuevas realidades en el aula*. Editorial Oceano. https://books.google.com/books?hl=es&lr=&id=AULhBgAAQBAJ&oi=fnd&pg=PT113&dq=Raul+et+al.+(2015).+Mobile+learning:+nuevas+realidades+en+el+aula.&ots=k9nk78oF2M&sig=1u9b6KVLeibV9eEAsK7xCGtgCqA.

ANEXOS:

Anexo 1: Entrevista realizada para identificar las deficiencias en el proceso de reporte de incidencias por parte del usuario en el uso del sistema operativo Nova.

I. Entrevista realizada a el cliente, el mismo forma parte del equipo de desarrollo de GNU/Linux Nova: Usted ha sido seleccionado para esta entrevista por identificar la necesidad de un proceso de reporte de incidencia luego del despliegue de GNU/Linux Nova.
Pregunta 1: ¿Cuáles son las vías existentes para captar la información proveniente del reporte de incidencia o experiencia de usuario? ¿Por qué es importante este proceso?
Pregunta 2: ¿Cuáles son las características o categorías del sistema operativo, en donde los usuarios presentan más dificultades en su uso paulatino? ¿Cómo categorizaría las incidencias detectadas en Nova?
Pregunta 3: ¿Por qué es importante contar con la información de la experiencia de usuario? ¿Cuál es el objetivo general de captar y obtener una base de datos con todos los reportes incidencias del usuario?
Pregunta 4: ¿Le brindaría una evaluación de gravedad a la incidencia? ¿Por qué?
Pregunta 5: ¿Por qué desarrollar una aplicación móvil y no una plataforma web? ¿Cuál es el objetivo final de la aplicación móvil?
Pregunta 6: ¿Es importante filtrar la información de reporte y convertirla en estadística visible por el equipo de desarrollo? ¿Por qué?
II. Entrevista realizada a trabajadores del departamento CESOL, vinculados al equipo de desarrollo y proceso de pruebas al sistema.
Pregunta 1: ¿Cómo se realizan las pruebas al sistema operativo Nova? ¿Cómo se evalúa la calidad del sistema operativo nova?
Pregunta 2: ¿Cuándo se obtiene la lista de chequeo de los encargados de calidad? ¿Qué patrones

se evalúan en esa lista?
Pregunta 3: ¿Cómo se realiza el proceso de despliegue del sistema operativo nova en una institución? ¿Cómo se maneja la etapa de consolidación?
Pregunta 4: ¿Cómo se maneja el proceso de soporte vinculado a la etapa de consolidación? ¿Por qué solo se contacta a el equipo de desarrollo en el nivel 3?

Anexo 2: Historias de Usuarios:

Historia de Usuario	
Número: 1	**Nombre del Requisito:** Registrar Usuario
Programador: Julio C. Gallardo Concepción	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 1 semana
Riesgo en Desarrollo: Alto	**Tiempo Real:** 8 días
Descripción: La aplicación debe permitirle a el usuario autenticarse, por su id de usuario y contraseña. Ambos predefinido en el proceso de registro.	
Observaciones: El usuario debe rellenar todos los campos con los datos requeridos, el campo de correo debe estar rellenado con el correo de la institución donde se está realizando el despliegue	
Prototipo de interfaz:	

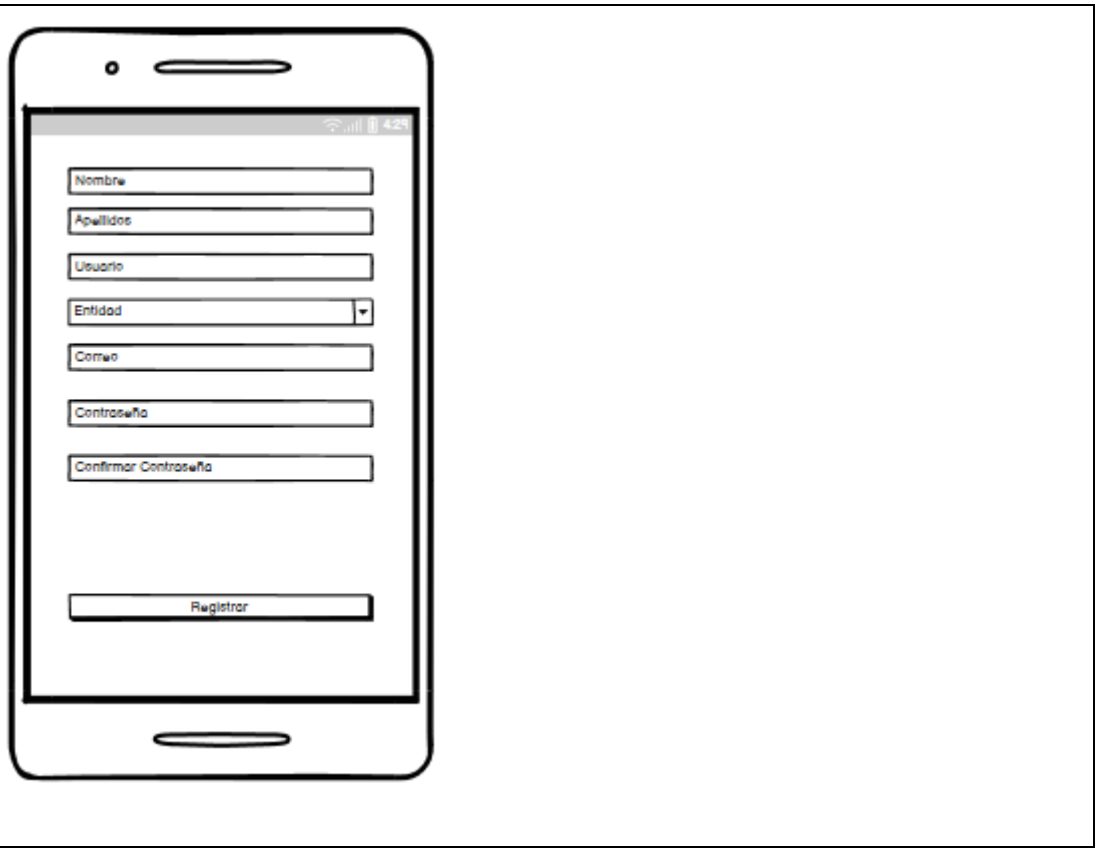


Tabla 9: HU Registrar Usuario

Historia de Usuario	
Número: 2	**Nombre del Requisito:** Autenticar Usuario
Programador: Julio C. Gallardo Concepción	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 4 días
Riesgo en Desarrollo: Medio	**Tiempo Real:** 2 días

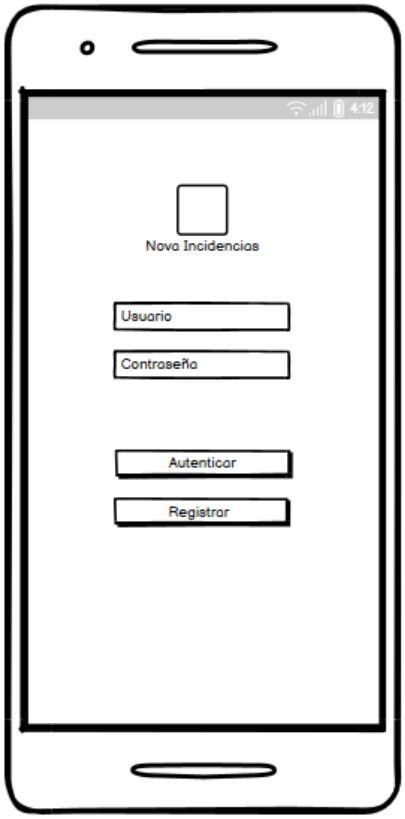
Descripción: La aplicación debe permitirle a el usuario autenticarse, por su id de usuario y contraseña. Ambos predefinido en el proceso de registro.
Observaciones: El usuario debe realizar correctamente el proceso de registro para poder acceder a las funcionalidades de la aplicación
Prototipo de interfaz: 

Tabla 10: HU Autenticar Usuario

Historia de Usuario	
Número: 4	**Nombre del Requisito:** Mostrar detalle de la incidencia

Programador: Julio C. Gallardo Concepción	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 1 semana
Riesgo en Desarrollo: Medio	**Tiempo Real:** 2 días

La aplicación debe mostrar al administrador los detalles registrados de la incidencia realizada por el usuario.

Se muestran los siguientes datos:

- Estado de la incidencia
- Id de la incidencia
- Usuario que realizo la incidencia
- Entidad correspondiente
- Categoría de la incidencia
- Gravedad de la incidencia
- Descripción
- fecha

Observaciones:

Dentro de la misma interfaz se muestran los botones que permitirán la acción de:

- Cambiar de Estado de la incidencia
- Eliminar Incidencia

Prototipo de interfaz:

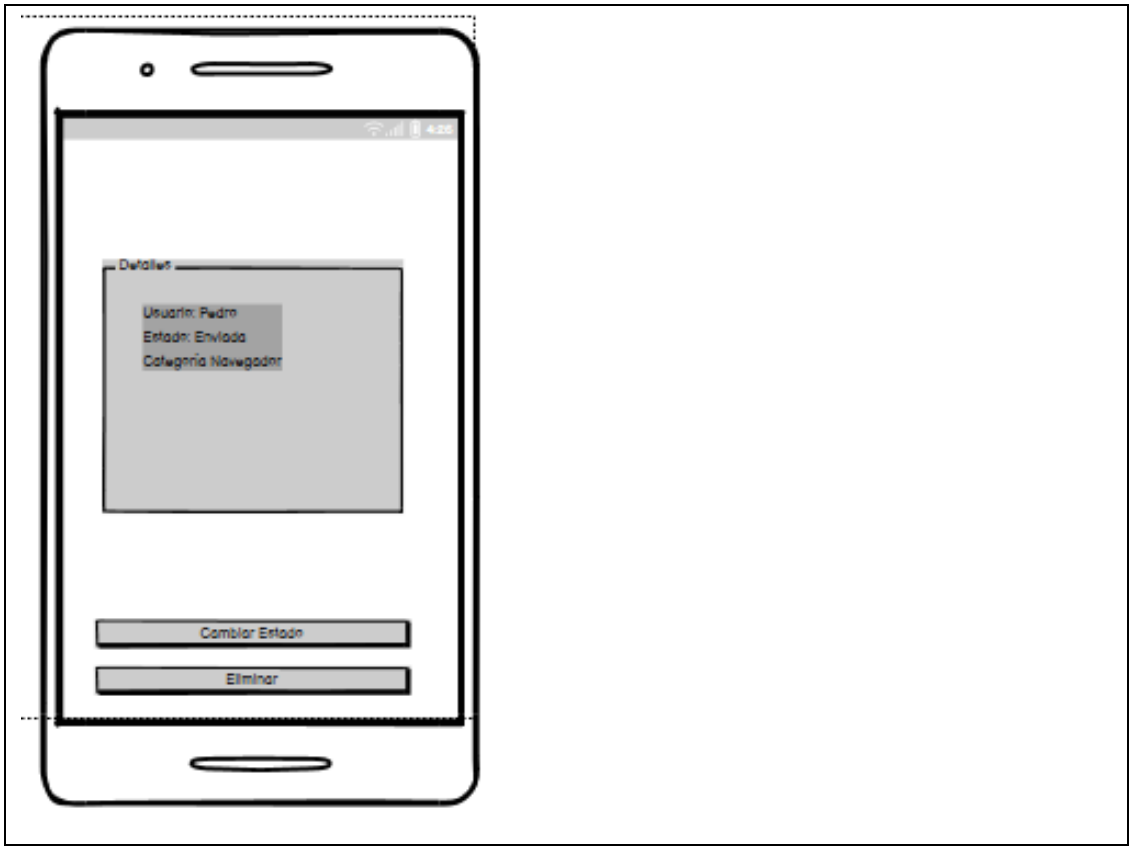


Tabla 11: Mostrar detalle de la incidencia

Historia de Usuario	
Número: 8	**Nombre del Requisito:** Listar incidencias
Programador: Julio C. Gallardo Concepción	**Iteración Asiganada:**1
Prioridad: Alta	**Tiempo Estimado:** 1 semana
Riesgo en Desarrollo: Medio	**Tiempo Real:** 4 días
La aplicación debe permitir al administrador obtener un listado de incidencias de todos los usuarios.	

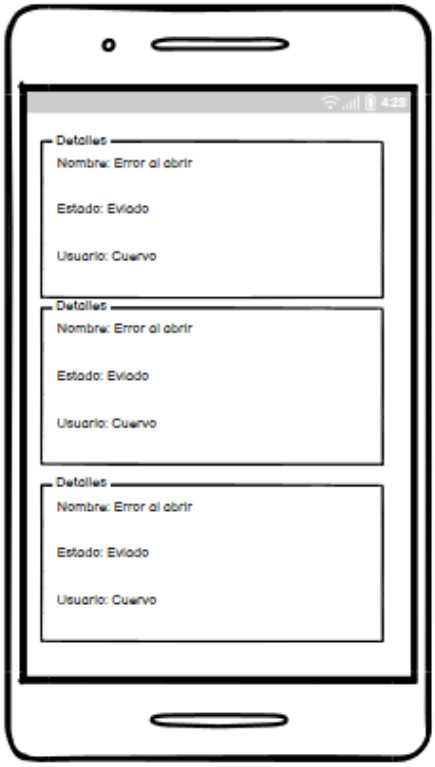
Observaciones: El administrador desde el listado de incidencia puede obtener los detalles de la misma y cambiar el estado.
Prototipo de interfaz: 

Tabla 12:Listar incidencias

Historia de Usuario	
Número: 6	**Nombre del Requisito:** Eliminar Incidencia
Programador: Julio C. Gallardo Concepción	**Iteración Asiganada:**1
Prioridad: media	**Tiempo Estimado:** 1 semana

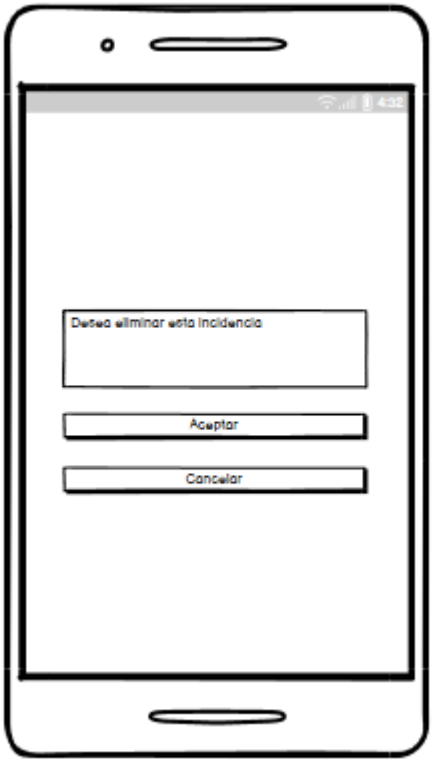
Riesgo en Desarrollo: Medio	**Tiempo Real:** 5 días
La aplicación debe permitir tanto al usuario como el administrador eliminar las incidencias que se encuentran registradas.	
Observaciones: Si una incidencia cambia de estado de Enviado a Procesado o Listo, no es posible realizar la función Eliminar.	
Prototipo de interfaz: 	

Tabla 13:Eliminar Incidencia

Anexo 3: Diagramas de Clases de diseño:

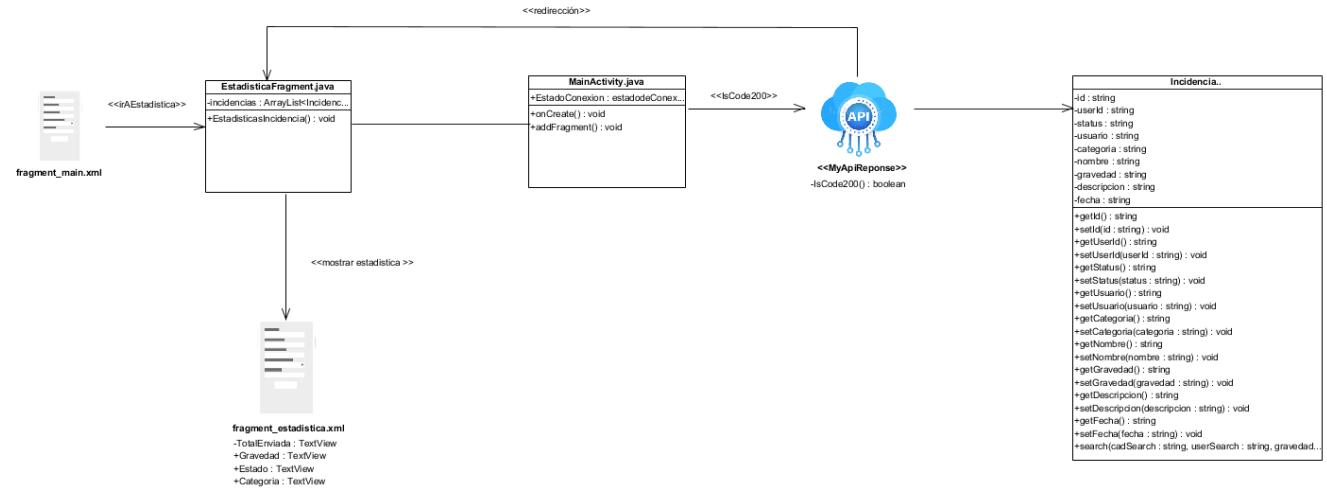


Ilustración 13:Diagrama de clases de diseño: Estadísticas Incidencias (elaboración propia)

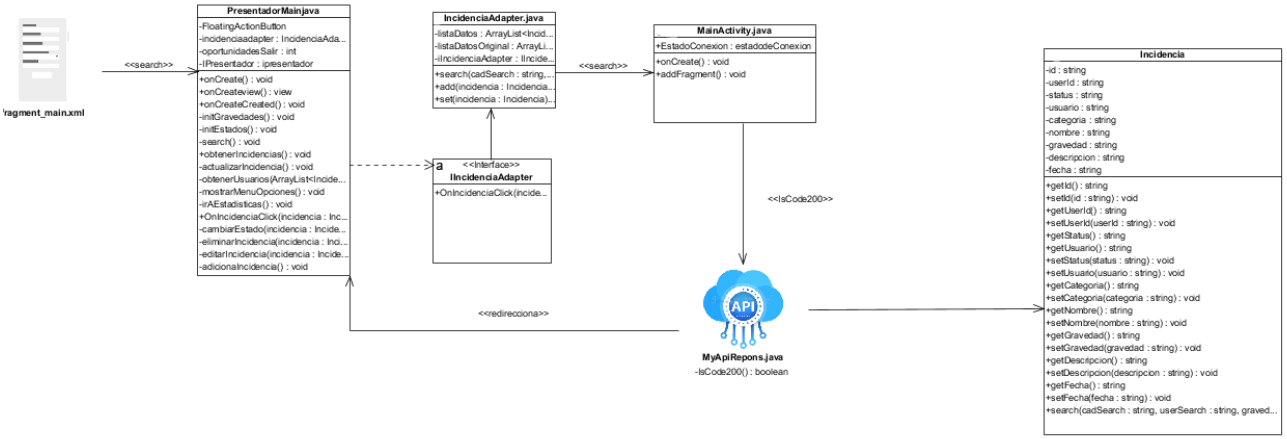


Ilustración 14:Diagrama de clases de diseño: Buscar Incidencia (elaboración propia

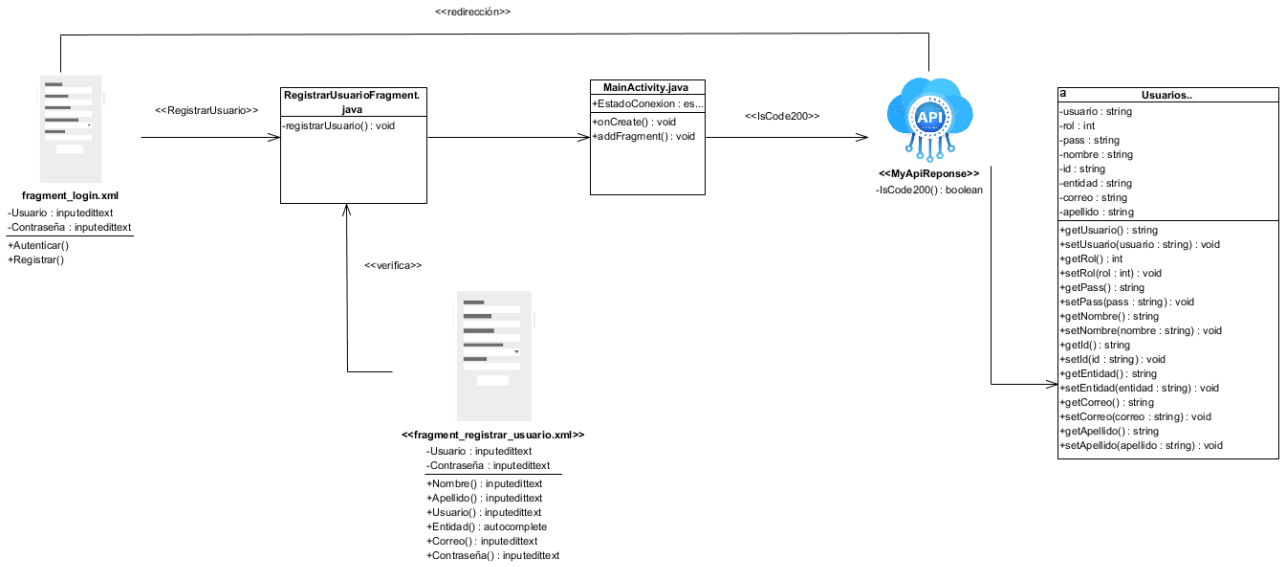


Ilustración 15:Diagrama de clases de diseño: Registrar usuario (elaboración propia)

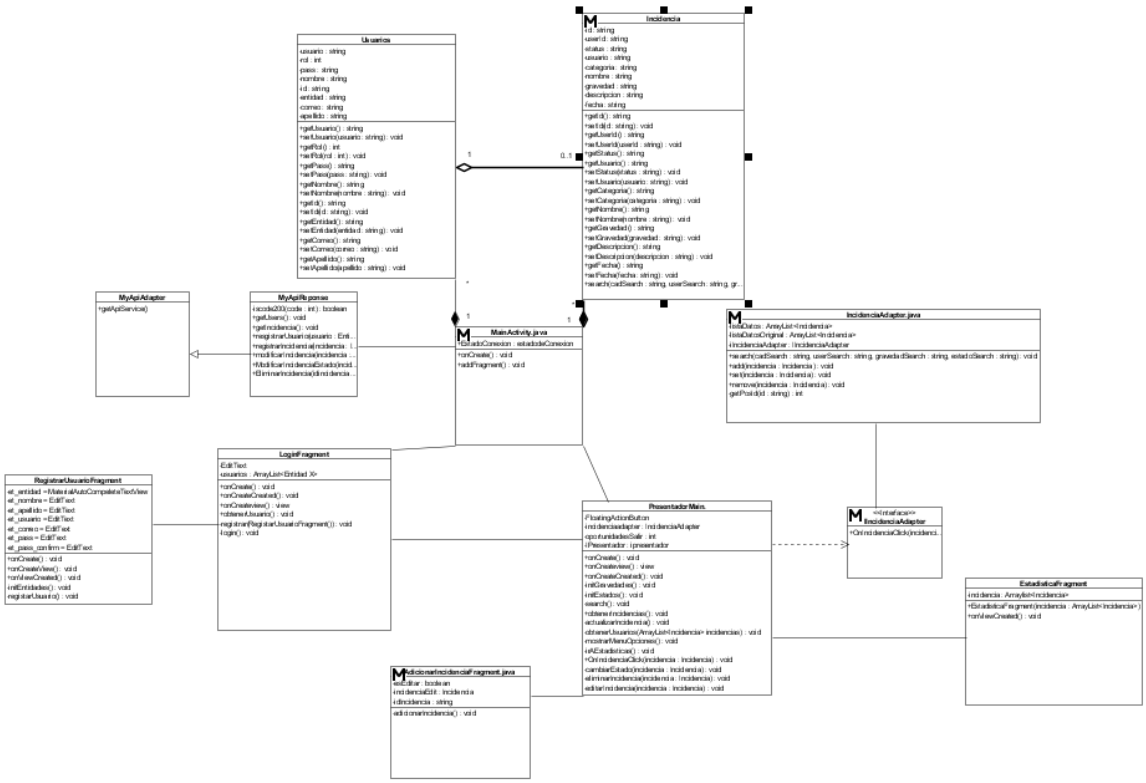


Ilustración 16:Diagrama de clases de diseño: Global (elaboración propia)

Anexo 4: Diagramas de Secuencia:

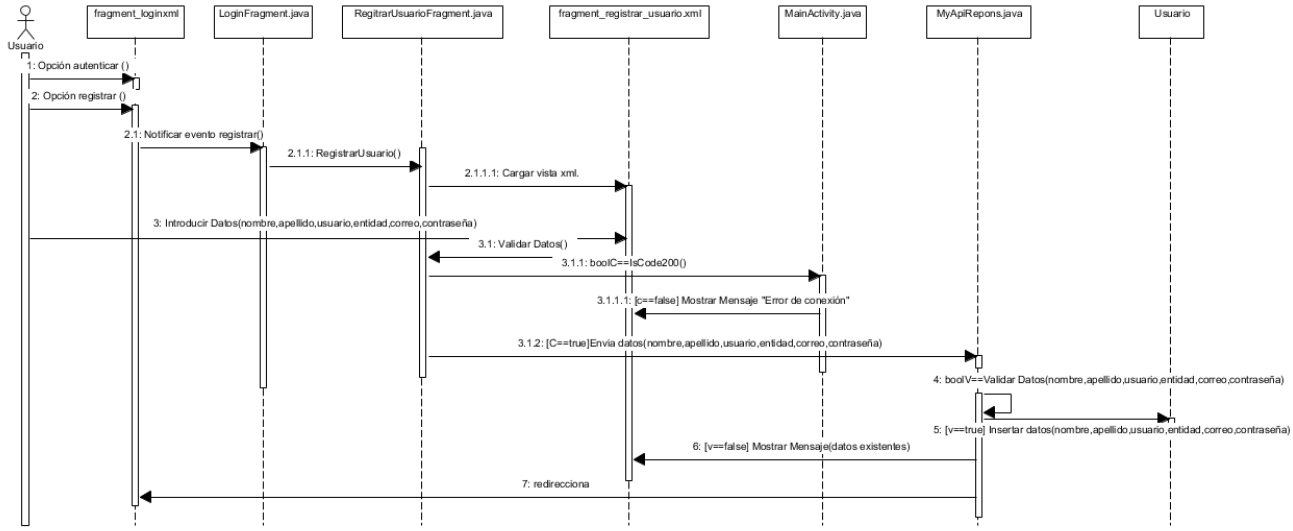


Ilustración 17: DS Registrar Usuario

Anexo 5: Código empleado en prueba unitaria:

AdicionarIncidenciaFragmentTest.java

```
package cu.cesol.gallardo.novaincidencias.ui;

import org.junit.Assert;
import org.junit.Test;
import org.junit.runner.RunWith;
import org.junit.runners.JUnit4;
import java.util.ArrayList;
import cu.uci.isw.mshare.Entities.Evaluacion;
import static org.mockito.Mockito.mock;
import static org.mockito.Mockito.verify;

@RunWith(JUnit4.class)
public class AdicionarIncidenciaFragmentTest {

    @Test
    public void TestCreateInstance(){
        Incidencia testIncidencia=mock(Incidencia.class);
        Assert.assertNotNull(testIncidencia);
```

```
    }
    @Test
    public void TestAddIncidencia(){
        try {
            Crear_Incidencia testCrearIncidencia=mock(Crear_Incidencia.class);
            testCrearIncidencia.addIncidencia();
            Assert.fail("Expected return true");
        }catch (Exception e){
            String expectedMessage = "Datos incompletos";
            Assert.assertTrue("error message should contain message " + e,
                    e.getMessage().contains(expectedMessage));
        }
    }

    @Test
    public void TestModIncidencia(){
        try {
            Crear_Incidencia testCrearIncidencia=mock(Crear_Incidencia.class);
            testCrearIncidencia.modIncidencia();
            Assert.fail("Expected return true");
        }catch (Exception e){
            String expectedMessage = "Datos incompletos";
            Assert.assertTrue("error message should contain message " + e,
                    e.getMessage().contains(expectedMessage));
        }
    }
    @Test
    public void TestinitCategorias(){
        Buscar testBuscar.=mock(Buscar.class);
        ArrayList<String>initCategoria =testBuscar.Utils.getCategorias();
        Assert.assertEquals(initCategoria,null);
    }
@Test
    public void TestinitGravedades(){
        Buscar testBuscar.=mock(Buscar.class);
        ArrayList<String>initGravedades =testBuscar.Utils.getGravedades ();
        Assert.assertEquals(Gravedades,null);
```

```
        }
}
```

Anexo 6: Prueba de funcionalidad

Escenario	Descripción	Seleccionar estado	Opciones	Respuesta de la aplicación	Flujo central
		V	V		
EC1 Cambiar Estado de la incidencia	El administrador selecciona la incidencia deseada, pulsa el botón “Cambiar Estado” y selecciona entre: “Enviada, Procesando, Listo”	Enviada Procesando Listo	Aceptar Cancelar	El estado es cambiado y se visualiza en los detalles de la incidencia	1- La aplicación luego de ejecutado el cambio de la incidencia, restringe el botón “modificar” y “eliminar” al usuario
EC1.2 Modificar Estado de la incidencia luego de eliminada	El administrador procede a realizar el cambio de estado de la incidencia en el momento que el usuario elimina la misma.	V	V	La aplicación muestra el siguiente mensaje:” Error incidencia Eliminada”	La aplicación muestra un mensaje de Error al administrador.
		Enviada Procesando Listo	Aceptar Cancelar		

Tabla 14: Caso de prueba de funcionalidad a la historia de usuario cambiar estado de incidencia (elaboración propia)

Escenario	Descripción	Estado de la incidencia	Opciones	Respuesta de la aplicación	Flujo central
		V	V		
EC1 Eliminar incidencia en estado enviado	El usuario selecciona la incidencia deseada, pulsa el botón “eliminar incidencia” y confirma la eliminación de la incidencia.	Enviado	Aceptar Cancelar	Incidencia eliminada y vuelve a el listado de incidencias del usuario	1- La aplicación luego de ejecutado la eliminación de la incidencia, también es eliminada para el administrador
EC1.2 Eliminar la incidencia luego de cambio de estado.	El usuario selecciona la incidencia deseada y no le muestra la opción eliminar.	V	V	La aplicación no le muestra a el usuario la opción de eliminar la incidencia.	La aplicación sigue mostrando la incidencia a el administrador.
		Recibido Procesando Listo			

Tabla 15:Caso de prueba de funcionalidad a la historia de usuario eliminar incidencia (elaboración propia)

Anexo 7: Carta de aceptación

FACULTAD 1

Centro de Software Libre "CESOL"

CARTA DE ACEPTACIÓN

La Habana, 18 de noviembre del 2022

DE UNA PARTE: El colectivo de trabajadores perteneciente al equipo de desarrollo del GNU/Linux Nova de la Facultad 1 en la Universidad de la Ciencias Informática, representado en este acto por el Ing. Lexys Manuel Díaz Alonso

DE OTRA PARTE: El desarrollador del software, representado en este acto por: Julio César Gallardo Concepción

CONSIDERANDO: Que se cumplen las funcionalidades definidas para el producto.

CONSIDERANDO: Que la aplicación móvil desarrollada tiene un alto valor para el reporte de incidencias del sistema operativo Nova. Definiendo el proceso de retroalimentación entre equipo de desarrollo y usuarios finales

CONSIDERANDO: Que su uso se hace extensivo a todas las entidades que cuentan hoy con el sistema operativo Nova. Garantizando la portabilidad desde el diseño Android.

POR TANTO: La parte cliente acuerda formalizar mediante la presente acta, la aceptación del producto: Nova Incidencia: Aplicación móvil para el reporte de incidencias luego del despliegue de GNU/Linux Nova.

Para que así conste, se expide la siguiente acta en dos ejemplares.

Recibe: Ing. Lexys Manuel Díaz Alonso