Centro de Informatización de la Gestión Documental (CIGED)

Facultad 2

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

Aplicación móvil para el control de inventario en los locales del centro de desarrollo CIGED

Autor:

Johan Alejandro Falcón Martínez

Tutores:

Ing. Roberto Soriano Sifontes

Ing. Liomar Rodríguez Guerra

La Habana Cuba, septiembre de 2023

Año 65 de la Revolución

Declaración de autoría

Declaro por este medio que yo Johan Alejandro Falcón Martínez, con carné de identidad 99101102401 soy el autor principal del trabajo titulado *"Aplicación móvil para el control de inventario en los locales del centro CIGED"* y autorizo a la Universidad de las Ciencias Informáticas (UCI) a hacer uso de la misma en su beneficio, así como los derechos patrimoniales con carácter exclusivo.

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

_____

Firma del autor

Johan Alejandro Falcón Martínez

_____

Firma del tutor

Ing. Roberto Soriano Sifontes

_____

Firma del tutor

Ing. Liomar Rodríguez Guerra

Datos de contacto

Datos del Autor

Nombre y Apellidos: Johan Alejandro Falcón Martínez

Correo electrónico: johanafm@estudiantes.uci.cu

Situación laboral: Estudiante

Institución a la que pertenece: Universidad de las Ciencias Informáticas.

Dirección: Carretera a San Antonio de los Baños, Km 2 ½, reparto Torrens, municipio Boyeros, La Habana, Cuba. CP: 19370

Datos de los Tutores

Nombre y apellidos: Roberto Soriano Sifontes

Correo electrónico: rsoriano@uci.cu

Situación laboral: Director Centro CIGED

Años de graduado: 8 Años

Especialidad de graduación: Ingeniero en Ciencias Informáticas.

Institución a la que pertenece: Universidad de las Ciencias Informáticas.

Dirección: Carretera a San Antonio de los Baños, Km 2 ½, reparto Torrens, municipio Boyeros, La Habana, Cuba. CP: 19370

Nombre y apellidos: Liomar Rodríguez Guerra

Correo electrónico: liomar@uci.cu

Situación laboral: Desarrollador Centro CIGED

Años de graduado: 6 Años

Especialidad de graduación: Ingeniero en Ciencias Informáticas.

Institución a la que pertenece: Universidad de las Ciencias Informáticas.

Dirección: Carretera a San Antonio de los Baños, Km 2 ½, reparto Torrens, municipio Boyeros, La Habana, Cuba. CP: 19370

Dedicatoria

Quiero dedicar este trabajo de diploma a mi familia en primer lugar por nunca dejar de creer en mi incluso en los peores momentos. Por ser mi motor impulsor en cada una de mis batallas, por el cariño y amor incondicional. Le dedico este gran logro especialmente a mi abuelo con mucho cariño donde quiera que estés espero estes muy orgulloso de tu nieto.

Quiero además dedicarle mi investigación a todos los que de una manera u otra formaron parte de este sueño y a todos los que hicieron posible su culminación de forma exitosa.

A todas las personas que como yo lucharon hasta el último minuto sin rendirse.

A la Universidad de las Ciencias Informáticas y en especial a mi querida facultad, a todos los escorpiones de corazón.

Agradecimientos

Quiero agradecer de manera muy especial a todas y cada una de las personas especiales que hicieron posible este sueño. En primer lugar, a mi familia por darme la oportunidad de contar con ellos en cada momento. A mi mamá por no dejar de confiar en mi y ser mi fortaleza. A mi abuela por ser la que nos guía y siempre nos enseña a luchar y no rendirnos. A mi abuelo, que, aunque no esté jamás dejará de estar en nuestros corazones. A mi papá que por darme la fortaleza y las ganas. A mi padrastro, mi otro padre, por acompañarnos en este largo camino. A mi madrastra por ser mi otra madre y siempre tenerme presente. A mi hermana por quererme tanto y joderme en los momentos más difíciles. A mis abuelos paternos y al resto de la familia, que todos son muy importantes para mí.

A mis profesores, por su entrega diaria. Desde los primeros hasta los últimos, todos pusieron su granito de arena para construir este gran logro.

A mis tutores por apoyarme en el proceso y confiar en mí como estudiante y amigo. Por su conocimiento y atención en cada momento.

A mi tribunal por cada uno de los consejos para el mejoramiento y el aprendizaje como ingeniero y como persona.

A mi oponente por sus consejos y ayuda para la culminación exitosa de la presente investigación.

A mis compañeros que hoy son mis hermanos de vida por cada momento especial que vivimos a lo largo de esta bella etapa.

En general a todas y cada una de las personas que tengo en mi vida.

A la Universidad de las Ciencias Informáticas por acogerme como un hijo más en este camino lleno de momentos lindos que se quedarán por siempre en mi corazón.

Muchas Gracias.

Resumen

Las nuevas tecnologías de la información y las comunicaciones han incidido cada vez más en el desarrollo de aplicaciones para teléfonos inteligentes con el fin de ayudar en la realización de actividades. De ahí que el objetivo de la presente investigación sea desarrollar una aplicación móvil con Sistema Operativo (SO) Android que sea capaz de manejar de manera eficiente el control de inventario en las áreas asociadas al Centro de Informatización de la Gestión Documental (CIGED) de la Facultad 2. El desarrollo de la aplicación se basa en el uso de la Metodología de Desarrollo de Software XP. Finalmente, se lleva a cabo la validación de la propuesta a través de los métodos definidos en la investigación.

Palabras clave

Aplicación, Android, control, denominación, inventario, medios, sistema.

Abstract

New information and communications technologies have increasingly influenced the development of applications for smartphones to help carry out activities. Hence, the objective of this research is to develop a mobile application with Android Operating System (OS) that is capable of efficiently managing inventory control in the areas associated with the Document Management Computerization Center (CIGED) of the Faculty 2. The development of the application is based on the use of the XP Software Development Methodology. Finally, the validation of the proposal is carried out through the methods defined in the research.

KEYWORDS

Application, Android, control, naming, inventory, media, system.

Índice general

Índice de tablas

Índice de figuras

Introducción

En la actualidad, las Tecnologías de la Información y Comunicación (TIC) se han convertido en una parte esencial de nuestras vidas. Desde los teléfonos inteligentes y las redes sociales hasta los sistemas en la nube y la inteligencia artificial, las TIC están cambiando la forma en que nos comunicamos, trabajamos y nos relacionamos con el mundo.

Uno de los aspectos fundamentales de las TIC en el ámbito empresarial es la gestión de inventario de medios. Este proceso básico es vital para las empresas que dependen de equipos, dispositivos y otros recursos tecnológicos para llevar a cabo sus operaciones diarias, además de medios no tecnológicos utilizados en sus principales locales.

La autora Carmen Arenal Laza define en su libro Gestión de inventarios. UF0476 el término inventario como la relación de bienes de los cuales se dispone, clasificados según se categoría(de Guevara, 2020). La misma expresa la necesidad de que las empresas realicen de forma eficaz sus controles asociados al tema para garantizar el funcionamiento adecuado de cada institución.

El control de inventario en una empresa es importante porque con él es posible mantener un balance correcto sobre la mercancía y la distribución de un almacén, lo que ayuda a reducir costos, prevenir el fraude, acelerar los procesos logísticos y realizar informes sobre el estado financiero del negocio (Dsipatchtrack, 2023). Los negocios de cualquier tipo y tamaño necesitan llevar un control detallado de sus activos y productos para poder operar eficientemente. Ese registro se realiza a través del inventario, que es una herramienta fundamental en la gestión y toma de decisiones empresariales. (HubSpot, 2023)

El proceso de inventario de medios implica el seguimiento y la administración de todos los activos de una organización, como computadoras, servidores, impresoras, mesas, sillas y demás medios de un local. El objetivo principal es tener un registro preciso y actualizado de los medios disponibles, su ubicación, estado y otra información relevante.

La Universidad de las Ciencias Informáticas (UCI) no se encuentra exenta de este tipo de procesos ya que cuenta con múltiples instalaciones las cuales

albergan un gran número de medios y dispositivos informáticos a su disposición. En el caso del Centro de Informatización de la Gestión Documental (CIGED) de la Facultad 2, se lleva a cabo regularmente un proceso de inventario, el cual se utiliza para tener un control fiable de los medios básicos en los locales asociados al mismo. A día de hoy este proceso presenta deficiencias en la recopilación de datos y el control adecuado de los medios, por llevarse a cabo de manera manual, además de una lenta gestión en este tipo de procesos tan importantes.
A continuación, se detallan los problemas que surgen al llevar a cabo este proceso de manera manual:

- Dificultades en la recopilación de datos: Al realizar el inventario de forma manual, se depende de la intervención humana para recopilar y registrar la información. Esto da lugar a errores humanos, como omisiones, duplicaciones o datos incorrectos. La recopilación manual también puede ser un proceso lento y propenso a retrasos, lo que afecta la eficiencia del inventario.
- Falta de control adecuado de los medios: La realización manual del inventario dificulta tener un control preciso sobre los medios y dispositivos informáticos presentes en los locales del CIGED. La falta de precisión en la información recopilada puede llevar a discrepancias entre los registros y la realidad, lo que afecta la exactitud de los datos y la toma de decisiones basada en ellos.
- Mal control en los locales: La falta de una herramienta eficiente para el inventario manual conlleva un riesgo de un control deficiente en los locales asociados al centro CIGED. Existen dificultades para identificar rápidamente los medios disponibles, determinar su ubicación exacta y registrar los cambios o movimientos de los mismos. Esto puede resultar en pérdida de tiempo y recursos al buscar o reemplazar medios extraviados o mal gestionados.
- Lenta gestión de procesos: La realización manual del inventario implica un proceso más lento en comparación con el uso de herramientas informáticas especializadas. La necesidad de revisar físicamente cada dispositivo, registrar manualmente los detalles y actualizar los registros de inventario

consume tiempo y recursos significativos. Esta lentitud en la gestión puede repercutir en la eficiencia general del centro CIGED y afectar la capacidad de responder rápidamente a las necesidades y cambios en los medios informáticos.

Informatizar el proceso de control de inventario brinda una serie de beneficios significativos. En primer lugar, mejora la eficiencia y precisión al reducir la dependencia de la intervención humana y minimizar los errores asociados con la recopilación y el registro manual de datos. Al automatizar el proceso, se acelera el tiempo necesario para realizar inventarios y se minimizan los errores humanos, como omisiones o registros incorrectos.

Del análisis de la situación anteriormente descrita, se identifica como **problema a resolver:** ¿Cómo mejorar el proceso de control de inventario que se lleva a cabo en los locales del centro de desarrollo CIGED de la Facultad 2 en la UCI?

A partir del problema a resolver se define como **objeto de estudio**: Control de inventario en los locales del centro de desarrollo CIGED.

Para darle solución al problema a resolver se define como **objetivo general**: Desarrollar una Aplicación móvil para el control de inventario en los locales del centro de desarrollo CIGED.

Se reconoce como **campo de acción**: Aplicación móvil para el control de inventario de los locales del centro de desarrollo CIGED.

Para darle cumplimiento al objetivo general se definieron los siguientes **objetivos específicos**:

1. Establecer los fundamentes y referentes teórico metodológicos para el control de inventario.
2. Elaborar el diseño de la propuesta de solución teniendo en cuenta las características y necesidades del centro de desarrollo CIGED.
3. Implementar las funcionalidades necesarias para el correcto funcionamiento de la aplicación.
4. Validar la aplicación a partir de una estrategia de prueba definida.

Diseño metodológico de la investigación:

El diseño metodológico de la investigación se refiere al plan o estructura que se establece para llevar a cabo un estudio o investigación de manera sistemática y rigurosa. Es la estrategia que se utiliza para recopilar datos, analizarlos y obtener conclusiones válidas y confiables.

Métodos Teóricos:

- **Histórico-Lógico:** para determinar los antecedentes, tendencias, irregularidades asociadas al proceso del control de inventario.
- **Analítico-Sintético:** para realizar el estudio teórico de la investigación facilitando el análisis de documentos y la extracción de los elementos más importantes acerca del proceso de inventario.
- **Inductivo-deductivo:** Se utiliza para inducir las características particulares de los sistemas de control de inventario.
- **Modelación:** Se utiliza para la generación de los diagramas necesarios para lograr una correcta implementación de la propuesta de solución.

Métodos Empíricos:

- **Observación:** permite la realización de análisis asociados a la recopilación de datos en el proceso de inventario.
- **Entrevista:** consiste en llevar a cabo una entrevista con los principales implicados en el proceso de inventario a través de una comunicación interpersonal que se lleva a cabo por medio de una conversación estructurada donde es muy importante la forma en que se plantea la conversación y la relación que se establece en la entrevista.

El presente trabajo de diploma se ha estructurado de la siguiente manera:

Capítulo 1. Fundamentos Teóricos: En este capítulo se exponen los principales conceptos que contribuyen al mejor entendimiento del problema en cuestión. Se especifican detalladamente todos los argumentos que esclarecen el objeto de estudio, a partir del análisis del estado del arte de los sistemas homólogos, enfocados en los procesos de inventario utilizando el SO Android. Además, se realiza la caracterización de la metodología, herramientas y tecnologías que serán utilizadas en el desarrollo de la solución.

Capítulo 2. Planificación y Diseño: Este capítulo contiene un análisis asociado al proceso de inventario en el centro de desarrollo, detallándose como se realiza el diseño del sistema a desarrollar. Se elaborarán las tablas y artefactos pertinentes para un mejor entendimiento del sistema.

Capítulo 3. Implementación y Prueba: Este capítulo contiene una descripción de las distintas pruebas realizadas a la propuesta de solución, con el fin de comprobar su funcionamiento, robustez y validar su uso para el centro de desarrollo.

Capítulo 1. Fundamentos Teóricos

Introducción al capítulo

Para lograr la comprensión del tema en cuestión se realiza en la investigación un acercamiento a los conceptos fundamentales asociados al control de inventario en las áreas descritas, así como una descripción detallada del entorno de trabajo donde se desarrolla el producto que le da solución a la problemática descrita. El control de inventario de medios básicos es fundamental para una gestión eficiente de cualquier empresa. Este capítulo explora los fundamentos teóricos que respaldan este proceso. También se analizan los objetivos y beneficios del control de inventario, así como las herramientas y tecnologías utilizadas en este proceso. Este capítulo sienta las bases para comprender el objetivo final de esta investigación.

1.1. Conceptos fundamentales

El proceso de control de inventario en los locales del centro CIGED se lleva a cabo de manera periódica, y los datos se almacenan en un archivo Excel. El control de inventario es fundamental para garantizar una gestión eficiente de los medios básicos en los locales de dicho centro, permitiendo un seguimiento preciso de las existencias.

Cada medio controlado se registra manualmente en una hoja de inventario, que suele estar dividida en columnas que contienen información como el Rótulo, la Subclasificación y el Local al cual pertenece cada medio. Esta información se transfiere posteriormente a un archivo Excel, donde se almacena y se utiliza para llevar un seguimiento más detallado.

Aunque el proceso de control de inventario manual tiene ventajas, como la simplicidad y la accesibilidad, también presenta desafíos. La posibilidad de errores humanos en los registros y la falta de actualización en tiempo real son algunas de las limitaciones que pueden afectar la precisión y eficiencia del sistema.

A continuación, se presentan algunos de los conceptos fundamentales asociados al tema de la investigación.

Inventario

Un inventario es un documento donde se registran todos los bienes tangibles y en existencia de una empresa, que pueden utilizarse para su alquiler, uso, transformación, consumo o venta. Debe ser una relación detallada en la que se incluyan, además de los tangibles, los derechos y deudas de una empresa. (HubSpot, 2023)

Control

Control puede ser el dominio sobre algo o alguien, una forma de fiscalización, un mecanismo para regular algo manual o sistémicamente o un examen para probar los conocimientos de los alumnos sobre alguna materia. (Significados.Com, 2023)

Control de inventario

El control de inventarios es un sistema que permite que una empresa gestione las existencias que almacena. De esta forma, además de saber qué tiene, identifica cuáles productos debe mover más rápido, cuáles son los que escasean, cómo es su rotación y en cuáles invierte más recursos para su correcto almacenaje. (HubSpot, 2023)

Código QR

Los códigos QR (Quick Response) son códigos de barras, capaces de almacenar determinado tipo de información, como una URL, SMS, EMail, Texto, etc. El código QR “Quick Response”, es un código de respuesta rápida. Es la evolución del código de barras y permite, al ser escaneado, ver la información que contiene. (CEPAL, 2023)

Rótulo

Letrero o inscripción con que se indica o da a conocer el contenido, objeto o destino de algo, o la dirección a que se envía. (Real Academia Española, 2023)

Escáner

El término inglés *scanner* llegó al castellano como escáner. Se trata de un dispositivo que se utiliza para la exploración y el registro de una imagen. El resultado de esta actividad es traducido por el escáner en señales eléctricas que pueden procesarse. (Definición.De, 2023)

1.2. Análisis de las soluciones existentes

Teniendo en cuenta las características exigidas por el cliente se decide enfocar la investigación al área de las aplicaciones móviles asociadas al proceso de control de inventario. A continuación, se detallan algunos puntos clave que respaldan esta elección:

- Acceso y movilidad: Una aplicación móvil permite realizar control de inventario en cualquier momento y desde cualquier lugar. Los dispositivos móviles, como teléfonos inteligentes o tabletas, son ampliamente utilizados y están siempre al alcance de la mano. Esto facilita el seguimiento y la actualización del inventario en tiempo real. La movilidad proporcionada por una aplicación APK mejora la eficiencia y la capacidad de respuesta del proceso de control de inventario.
- Escaneo de códigos QR: Las aplicaciones móviles suelen estar equipadas con cámaras que permiten escanear códigos QR de manera rápida y sencilla. Los códigos QR se utilizan ampliamente en la gestión de inventario para identificar medios y ubicaciones específicas. Al utilizar una aplicación móvil, el usuario puede escanear códigos QR en tiempo real para acceder a su información. Esto agiliza el proceso de control de inventario y reduce los errores asociados con la entrada manual de datos.
- Interfaz intuitiva y amigable: Las aplicaciones móviles suelen tener interfaces intuitivas y amigables para el usuario, diseñadas específicamente para dispositivos móviles. Esto facilita el uso y la adopción por parte del personal, incluso para aquellos con poca experiencia técnica. La interfaz de una aplicación móvil puede presentar información de manera clara y concisa, permitiendo una navegación fluida y un acceso rápido a las funciones necesarias para el control de inventario. Esto reduce la curva de aprendizaje y aumenta la eficiencia en el manejo de las tareas relacionadas con el inventario.
- Sincronización de datos en tiempo real: Una aplicación móvil puede estar conectada a una base de datos que almacena y sincroniza los datos en tiempo real. Esto significa que cualquier cambio realizado en la aplicación

móvil se reflejará instantáneamente en el sistema principal y viceversa. La sincronización de datos en tiempo real garantiza que los registros de inventario estén siempre actualizados. Esto evita discrepancias y mejora la precisión de la información del inventario.

- Actualizaciones y mejoras ágiles: Una alternativa móvil en forma de una aplicación APK, es más fácil implementar actualizaciones y mejoras en el sistema de control de inventario. Las actualizaciones se pueden enviar directamente a los dispositivos móviles de los usuarios a través de tiendas de aplicaciones o actualizaciones automáticas. Esto permite corregir errores, agregar nuevas funcionalidades y adaptarse rápidamente a los cambios en los requisitos o procesos comerciales. La agilidad en las actualizaciones asegura que la solución de control de inventario se mantenga al día y se ajuste a las necesidades cambiantes de la organización.

Actualmente existen a nivel mundial aplicaciones dedicadas al control de inventario las cuales son utilizadas por las corporaciones para llevar a cabo este proceso en sus instalaciones. Sin embargo, asociadas al control de inventario en locales tales como centros de desarrollo de software no se encontró ninguna alternativa. Las aplicaciones que se presenta a continuación cumplen indistintamente con algunos de los requisitos necesarios.

- **Inventario Móvil:** Estas aplicaciones están diseñadas para ser utilizadas en dispositivos móviles, lo que permite a los usuarios acceder y actualizar la información del inventario de cualquier lugar y en cualquier momento. (Bino Solutions, 2023)
- **Generador de código QR:** es una herramienta simple y conveniente que te ayuda a crear una imagen de código QR que se muestra en la pantalla. Se admiten varios tipos de contenido, incluidos Texto, Url, Correo electrónico, Número de teléfono, Contacto, Geolocalización y SMS. (YKART, 2023)
- **Sistema de Inventario Inteligente:** La aplicación permite administrar su inventario desde su computadora u otros sistemas operativos móviles mediante el uso de la misma. (NonZeroApps, 2023)

En Cuba también existen alternativas utilizadas para llevar a cabo el control de inventario. A continuación, se muestra un ejemplo encontrado en el sitio oficial (Tecnomática, 2023).

- **(tGestor)** Sistema de Control de Inventario de medios informáticos y comunicaciones.

El sistema, (con el nombre comercial provisional tGestor), está compuesto por los siguientes módulos:

- Gestor.
- Agentes de PC y móvil.
- Módulo General.
- Módulo de Seguridad.
- Módulo de Contabilidad
- Módulo de Medios.
- Módulo de PC
- Módulo de Gestión de Incidencias.
- Módulo de Telefonía móvil.

El sistema permite almacenar el historial de inventarios realizados. Permite realizar búsquedas avanzadas y da la posibilidad de crear reportes de incidencias en las empresas donde se utiliza.

Estas alternativas son muy utilizadas por diferentes empresas principalmente las que cuentan con almacenes de productos. En la investigación no se encuentran alternativas dedicadas exclusivamente al control de medios básicos en locales. A continuación, se muestra la tabla de análisis asociada a las alternativas estudiadas y los principales aspectos que se tienen en cuenta para este análisis. Se toman en cuenta los siguientes aspectos por ser los principales requisitos a tener en cuenta por el centro CIGED.

	Escanea códigos QR	**Genera códigos QR**	**Realiza inventario**	**Genera reportes**	**Versión gratuita**
Inventario Móvil			X	X	

Generador de código QR		X			X
Sistema de Inventario Inteligente	X		X		
(tGestor)			X		X

Tabla 1: Análisis de alternativas existentes. Elaboración propia.

Las soluciones de control de inventario disponibles presentar limitaciones en términos de funcionalidad, adaptabilidad o escalabilidad. Estas limitaciones pueden dificultar su implementación y uso en entornos específicos o en situaciones que requieren características específicas como las que exige el centro de desarrollo CIGED.

Estos sistemas son estándar y no permitir la personalización según las necesidades específicas del centro. Cada empresa tiene sus propios procesos, flujos de trabajo y requisitos únicos. Al desarrollar una alternativa propia, se puede adaptar la aplicación de control de inventario para que se ajuste perfectamente a los requisitos y características específicas del negocio, lo que resultará en una mayor eficiencia y efectividad en la gestión del inventario.

Algunas de las soluciones presentadas pueden tener altos costos asociados, como tarifas de licencia, mantenimiento o actualizaciones. Estos costos pueden ser prohibitivos para expandir su uso.

Al analizar las alternativas existentes y comprender sus limitaciones, se puede identificar la necesidad de desarrollar una solución propia que se ajuste mejor a los requisitos y particularidades del centro de desarrollo CIGED.

1.3. Metodología a utilizar

El concepto de metodología de desarrollo según Pressman en su 7ª edición se refiere a un enfoque sistemático y estructurado para el desarrollo de software. Es un conjunto de principios, prácticas y procedimientos que guían a los equipos de desarrollo en la planificación, diseño, implementación, pruebas y mantenimiento de un sistema de software (Pressman, 2010).

Una metodología de desarrollo de software es un marco de trabajo que se usa para estructurar, planificar y controlar el proceso de desarrollo de sistemas de información. Una gran variedad de estos marcos de trabajo ha evolucionado durante los años, cada uno con sus propias fortalezas y debilidades(Maida & Pacienzia, 2015).

El desarrollo de software no es una tarea fácil. Prueba de ello es que existen numerosas propuestas metodológicas que inciden en distintas dimensiones del proceso de desarrollo. Por una parte, tenemos aquellas propuestas más tradicionales que se centran especialmente en el control del proceso, estableciendo rigurosamente las actividades involucradas, los artefactos que se deben producir, y las herramientas y notaciones que se usarán. Estas propuestas han demostrado ser efectivas y necesarias en un gran número de proyectos, pero también han presentado problemas en otros muchos. Una posible mejora es incluir en los procesos de desarrollo más actividades, más artefactos y más restricciones, basándose en los puntos débiles detectados. Sin embargo, el resultado final sería un proceso de desarrollo más complejo que puede incluso limitar la propia habilidad del equipo para llevar a cabo el proyecto(Letelier, 2006).

Otra aproximación es centrarse en otras dimensiones, como por ejemplo el factor humano o el producto software. Esta es la filosofía de las metodologías ágiles, las cuales dan mayor valor al individuo, a la colaboración con el cliente y al desarrollo incremental del software con iteraciones muy cortas. Este enfoque está mostrando su efectividad en proyectos con requisitos muy cambiantes y cuando se exige reducir drásticamente los tiempos de desarrollo, pero manteniendo una alta calidad. Las metodologías ágiles están revolucionando la manera de producir software y, a la vez, generando un amplio debate entre sus seguidores y quienes, por escepticismo o convencimiento, no las ven como una alternativa para las metodologías tradicionales(Letelier, 2006).

Existen varias metodologías ágiles actualmente, a continuación, se presentan algunas de las más utilizadas.

- **Metodología Scrum**

Scrum es una metodología ágil de desarrollo de software que se basa en el trabajo colaborativo y en la entrega iterativa e incremental de productos. Se enfoca en la flexibilidad, adaptabilidad y satisfacción del cliente, al tiempo que garantiza la transparencia y la comunicación efectiva dentro del equipo. (Pressman, 2010)

Figura 1: Metodología Scrum. Fuente (Linkedin, 2023)

- **Metodología Kanban:**

  Kanban es una metodología ágil que se centra en la visualización y el control del flujo de trabajo. Se basa en una serie de tarjetas o notas (conocidas como "kanban") que representan las tareas o elementos de trabajo. Estas tarjetas se mueven a través de un tablero, que muestra claramente el estado y progreso de cada tarea. (Pressman, 2010)

Figura 2: Metodología Kanban. Fuente (Tecnosoluciones, 2023)

- **Metodología XP:**

  XP (Extreme Programming) es una metodología ágil que enfatiza la comunicación efectiva, el trabajo en equipo y la entrega temprana y frecuente de software funcional. Se basa en una serie de prácticas y valores fundamentales para promover la calidad, la adaptabilidad y la satisfacción tanto del cliente como del equipo de desarrollo. (Pressman, 2010)

  Las características clave de XP incluyen:

- Comunicación constante: La comunicación fluida y constante entre el cliente y el equipo de desarrollo es fundamental en XP. Esto se logra a través de la interacción directa, reuniones frecuentes y colaboración estrecha.
- Desarrollo orientado a pruebas (TDD): En XP, se enfatiza la escritura de pruebas antes de desarrollar el código. Esto ayuda a garantizar la calidad, permite la detección temprana de errores y facilita el mantenimiento.
- Integración continua: XP promueve la integración frecuente y continua del código fuente para detectar problemas de forma temprana y asegurar una base sólida.
- Diseño simple: XP promueve la simplicidad en el diseño del software, evitando la complejidad innecesaria y facilitando el mantenimiento y las modificaciones futuras.

- Entrega frecuente: En XP, se busca entregar software funcional de forma temprana y frecuente. Esto permite al cliente obtener valor rápidamente y brinda la oportunidad de recibir retroalimentación y realizar ajustes en etapas tempranas del desarrollo.

Figura 3: Metodología XP. Fuente (Openwebinars, 2023)

Teniendo en cuenta lo requisitos necesarios para la entrega de una solución eficaz, el corto periodo de tiempo dedicado al desarrollo del producto final y otros factores como los cambios constantes en el centro, el autor opta por escoger la metodología de desarrollo de software XP para darle cumplimiento a la tarea.

1.4. Entorno de trabajo

Para lograr cumplir los objetivos de la investigación en cuestión y la entrega de un producto final se hace necesario contar con los siguientes requisitos para el entorno de trabajo:

1.4.1 Lenguaje de programación

Dart – versión 3.0

- Dart es un lenguaje *open source* desarrollado en Google con el objetivo de permitir a los desarrolladores utilizar un lenguaje orientado a objetos y con análisis estático de tipo. Desde la primera versión estable en 2011, Dart ha cambiado bastante, tanto en el lenguaje en sí como en sus objetivos principales. Con la versión 2.0, el sistema de tipo de Dart pasó de opcional a estático, y desde su llegada, Flutter (explicamos qué es más

adelante) se ha convertido en el principal objetivo del lenguaje. (UPC, 2023)

1.4.2 Herramienta de desarrollo integrado (IDE)

Visual Studio Code – versión 1.82, (7 de septiembre de 2023)

- Visual Studio Code es un editor de código fuente ligero pero eficaz que se ejecuta en el escritorio y está disponible para Windows, macOS y Linux. Incluye compatibilidad integrada con JavaScript, TypeScript y Node.js, y cuenta con un amplio ecosistema de extensiones para otros lenguajes (como C++, C#, Java, Python, Go, .NET). (Microsoft, 2023)

1.4.3 Framework

Flutter SDK – versión 3.13.5 (20 de septiembre de 2023).

- Flutter es un framework de interfaz de usuario móvil gratuito y de código abierto creado por Google y lanzado en mayo de 2017. En pocas palabras, te permite crear una aplicación móvil nativa con una sola base de código. Esto significa que puedes usar un lenguaje de programación y una base de código para crear dos aplicaciones diferentes (para iOS y Android). (FreeCodeCamp, 2023)

1.4.4 Gestión de paquetes

Pub Dev

- El repositorio de paquetes oficial para las aplicaciones Dart y Flutter (Pub Dev, 2023).

1.4.5 Control de versiones

Git – Versión 2.42.0

- Es una herramienta muy utilizada por los programadores para llevar a cabo un control eficiente de las versiones que van generando en sus proyectos (Git, 2023).

1.4.6 Base de Datos

SQLite – Versión 2.3.0

- SQLite es una biblioteca de software que proporciona un motor de base de datos relacional de código abierto. A diferencia de las bases de datos tradicionales, SQLite no se ejecuta en un servidor separado, sino que se integra directamente en la aplicación que lo utiliza. (Android Developers, 2023)

Conclusiones del capítulo

La realización del análisis de los principales conceptos asociados al tema de la investigación, así como las alternativas informáticas existentes actualmente y la demostración del no cumplimiento de las mismas con las necesidades del centro permiten llegar a la conclusión de la necesidad de desarrollar una alternativa propia. Por otra parte, el análisis de diferentes herramientas y tecnologías permitió alcanzar los conocimientos necesarios para seleccionar las adecuadas para el desarrollo de la solución. Por lo que, el autor considera la utilización de Visual Studio Code como (IDE). Como metodología a utilizar XP, y como lenguaje de programación Dart para la implementación de la propuesta de solución.

Capítulo 2. Planificación y Diseño

Introducción del capítulo

En este capítulo se explican los principales aspectos de diseño de la aplicación móvil *Control de Inventario* para dispositivos con Sistema Operativo Android. Su contenido incluye la descripción del modelo de propuesta de solución, así como las fases de Planeación y Diseño correspondientes a la metodología de desarrollo de software seleccionada. También se define la arquitectura de software a utilizar y sus principales características. Expone además los ciclos de desarrollo, así como los planes de entrega del producto al cliente.

2.1 Propuesta de solución

El autor presenta como propuesta de solución de la investigación el desarrollo de la aplicación *Control de Inventario* para dispositivos con Sistema Operativo Android con versión igual o superior a 4.0. La aplicación permitirá realizar un control de inventario en los locales asociados al centro de desarrollo CIGED ubicado en la Faculta 2 de la Universidad de las Ciencias Informáticas. Además, contará con funciones dedicadas a ofrecer estadísticas, reportes del control de inventario y de movimientos mediante el escáner de códigos QR asociados a los medios básicos. La aplicación también contará con funciones asociadas a la generación de dichos códigos QR para los nuevos medios.

A continuación, se presenta el diagrama asociado a la propuesta de solución.

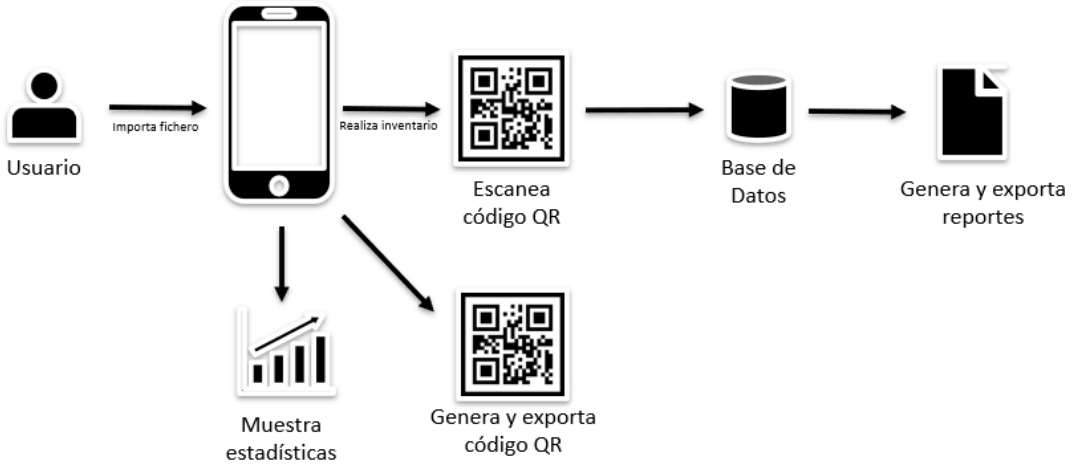


Figura 4: Diagrama propuesta de solución. Elaboración propia

Realizar el proceso asociado al control de inventario en un teléfono inteligente trae grandes ventajas, a continuación, se presentan las más significativas:

1. Movilidad: Con una aplicación móvil para el control de inventario, puedes acceder a la información del inventario desde cualquier lugar y en cualquier momento.
2. Ahorro de tiempo: Una aplicación móvil de control de inventario puede eliminar la necesidad de registrar información manualmente en papel. Puedes ingresar la información directamente en la aplicación desde tu dispositivo móvil. Esto ahorra tiempo y reduce los errores de registro.
3. Actualización en tiempo real: Con una aplicación móvil, los cambios en la información del inventario se pueden actualizar en tiempo real. Esto significa que puedes ver las últimas cifras de inventario, incluso si se están actualizando en este momento.
4. Análisis de datos: Las aplicaciones móviles de control de inventario también pueden proporcionar informes y análisis de datos detallados. Esto permite analizar más fácilmente las tendencias del inventario y tomar mejores decisiones de gestión de inventario.

En el caso específico de la aplicación desarrollada como parte de la propuesta de solución se aboga por el uso de códigos QR para el escaneo de la información de cada medio básico. El uso de códigos QR para este fin trae varias ventajas algunas de las cuales son:

1. Acceso rápido a información: Los códigos QR pueden contener una gran cantidad de información, como enlaces a sitios web, detalles de contacto, textos, etc. Los usuarios pueden escanear el código con su teléfono y acceder a la información directamente en su dispositivo.
2. Ahorro de tiempo y errores: Al utilizar un código QR para proporcionar información, se evita el proceso manual de ingreso de datos, lo que ahorra tiempo y reduce los errores. Los usuarios pueden escanear el código y recibir la información exacta sin tener que escribir o buscar.

3. Interactividad: Los códigos QR brindan una experiencia interactiva a los usuarios. Esto puede aumentar el compromiso del usuario y mejorar la experiencia del cliente.
4. Seguimiento y análisis: Los códigos QR se pueden utilizar para rastrear y analizar los medios. Además, puedes utilizar herramientas de análisis para ver cuántas veces se ha escaneado un código. Esto es útil para evaluar ciertos parámetros.
5. Versatilidad: Los códigos QR se pueden imprimir y colocar en diversos tipos de medios básicos. Esto los convierte en una herramienta versátil y accesible para aplicaciones en diferentes entornos.

2.2 Planeación

En la etapa de Planeación comienza la interacción con el cliente para identificar los requerimientos del sistema y se identifican las iteraciones y ajustes necesarios a la metodología según las características de la solución. En la misma se describen tres elementos fundamentales, los cuales son: Historia de Usuario (HU), Tiempo de Ejecución del Proyecto y Plan de Entrega, en ellos se recogen las características principales del sistema a construir y se concreta el tiempo que se necesitará para implementarlas.

2.3 Historias de Usuario

Las Historias de Usuario representan una breve descripción del comportamiento del sistema, se realizan por cada característica principal del sistema y son utilizadas para cumplir estimaciones de tiempo y el plan de lanzamientos, así mismo reemplazan un gran documento de requisitos y presiden la creación de las pruebas de aceptación. (UNIVERSIDAD NACIONAL AUTONOMA DE NICARAGUA, 2016)

2.3.1 Leyenda de las Historias de Usuario

Número: número de identificación para las HU, comenzando en 1 aumenta de forma incremental.

Nombre historia de usuario: nombre asignado a la HU, sirve para identificarla fácilmente tanto para los desarrolladores como para los clientes.

Modificación de historia de usuario: cantidad de modificaciones que se le ha realizado a la HU (de no tener modificaciones se pone ninguna).
Usuario: nombre del programador encargado de implementar la HU.
Prioridad del negocio: qué tan importante es para el cliente, se clasifica en Alta, Media y Baja.

- **Alta:** Son aquellas HU que constituyen funcionalidades fundamentales en el desarrollo el sistema, a las que el cliente define como principales para el control integral del sistema.
- **Media:** Son las funcionalidades a tener en cuenta por el cliente, sin que estas tengan una afectación sobre el sistema que se esté desarrollando.
- **Baja:** Es otorgada a las HU que son funcionalidades que sirven de ayuda al control de elementos asociados al equipo de desarrollo.

Iteración asignada: iteración en la que se desarrollará la HU.
Puntos estimados: tiempo en semanas que se le asignará, donde un punto de estimación corresponde a una semana de 5 días con 8 horas de trabajo diarias.
Descripción: es una breve descripción de la historia, detallando las operaciones del usuario y las respuestas del sistema.
Observaciones: informaciones de interés, como glosarios, detalles del usuario, entre otros.
Prototipo elemental de interfaz gráfica de usuario: se muestra una imagen de cómo quedaría la interfaz gráfica del requisito asociado a la HU.

2.4 Historias de Usuario del sistema

A continuación, se presentan las Historias de Usuario principales asociadas al desarrollo de la aplicación *Control de Inventario*, las restantes se encuentran en los anexos.

Historia de usuario	
Número: HU-2	**Nombre de la Historia de usuario:** Mostrar formulario para recopilar datos del código QR a generar.
Modificación de la historia de usuario:	

Usuario: Johan Alejandro Falcón Martínez	**Iteración asignada:** 1
Prioridad en el negocio: Alta	**Puntos estimados:** 0.5
Descripción: El sistema muestra un formulario para recopilar los datos necesarios que se utilizarán para generar un código QR los cuales son: Área de responsabilidad material, Rótulo y Subclasificación.	
Observaciones:	
Prototipo elemental de interfaz gráfica de usuario: 	

Tabla 2: HU-2 Mostrar formulario para recopilar datos del código QR a generar. Elaboración propia.

Historia de usuario	
Número: HU-3	**Nombre de la Historia de usuario:** Generar y exportar código QR.
Modificación de la historia de usuario:	
Usuario: Johan Alejandro Falcón Martínez	**Iteración asignada:** 1
Prioridad en el negocio: Alta	**Puntos estimados:** 1
Descripción: El sistema genera el código QR asociado a los datos recopilados para el nuevo medio básico y lo exporta como imagen (.jpg) a una carpeta en el dispositivo móvil.	
Observaciones:	

Prototipo elemental de interfaz gráfica de usuario:


Tabla 3: HU-3 Generar y exportar código QR. Elaboración propia.

Historia de usuario	
Número: HU-4	**Nombre de la Historia de usuario:** Escaneo de código QR.
Modificación de la historia de usuario:	
Usuario: Johan Alejandro Falcón Martínez	**Iteración asignada:** 2
Prioridad en el negocio: Alta	**Puntos estimados:** 1
Descripción: El sistema permite escanear un código QR asociado a un medio básico el cual contiene los siguientes datos: Área de responsabilidad material, Rótulo y Subclasificación. Muestra si el medio básico está ubicado en el Área de responsabilidad material asignada o no.	
Observaciones:	
Prototipo elemental de interfaz gráfica de usuario:	
	

Tabla 4: HU-4 Escaneo de código QR. Elaboración propia.

Historia de usuario	
Número: HU-5	**Nombre de la Historia de usuario:** Realizar inventario general.

Modificación de la historia de usuario:	
Usuario: Johan Alejandro Falcón Martínez	**Iteración asignada:** 2
Prioridad en el negocio: Alta	**Puntos estimados:** 2
Descripción: El sistema permite hacer un inventario general abarcando una o varias Áreas de responsabilidad material escaneando los códigos QR de entre un 10 % y un 100 % de los medios básicos mostrando una barra de progreso asociada al % de medios inventariados. Al finalizar muestra y genera un reporte en formato (.pdf) el cual exporta a una carpeta específica en el dispositivo móvil.	
Observaciones:	
Prototipo elemental de interfaz gráfica de usuario: 	

Tabla 5: HU-5 Realizar inventario general. Elaboración propia.

Historia de usuario	
Número: HU-7	**Nombre de la Historia de usuario:** Generar y exportar reporte movimientos de medios.
Modificación de la historia de usuario:	
Usuario: Johan Alejandro Falcón Martínez	**Iteración asignada:** 3
Prioridad en el negocio: Alta	**Puntos estimados:** 2
Descripción: El sistema realiza el movimiento de acuerdo al tipo seleccionado y genera un reporte en formato (.pdf) el cual se guarda en una carpeta específica del dispositivo móvil.	
Observaciones:	

Prototipos elementales de interfaces gráficas de usuario:


Tabla 6: HU-7 Generar y exportar reporte movimientos de medios. Elaboración propia.

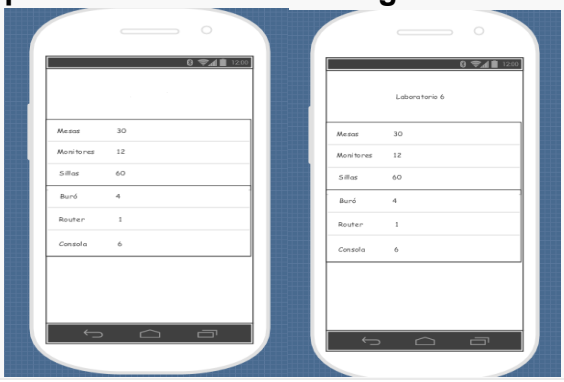
Historia de usuario	
Número: HU-9	**Nombre de la Historia de usuario:** Mostrar estadísticas.
Modificación de la historia de usuario:	
Usuario: Johan Alejandro Falcón Martínez	**Iteración asignada:** 4
Prioridad en el negocio: Media	**Puntos estimados:** 1
Descripción: El sistema muestra la cantidad de medios básicos en total agrupados por subclasificación según la información obtenida de la base de datos. Además, muestra dicha información por Área de responsabilidad material.	
Observaciones:	
Prototipo elemental de interfaz gráfica de usuario: 	

Tabla 7: HU-9 Mostrar estadísticas. Elaboración propia.

2.5 Requisitos funcionales

La declaración de los requisitos funcionales de un sistema es un paso fundamental en el proceso de ingeniería que se debe realizar previo al desarrollo

de cualquier sistema informático. Los Requisitos Funcionales (RF) son un pilar indispensable en la implementación del sistema y sirven para enfocar el desarrollo a las necesidades reales del cliente.

Según (Visuresolutions, 2023) Un requisito funcional es una declaración de cómo debe comportarse un sistema. Define lo que el sistema debe hacer para satisfacer las necesidades o expectativas del usuario. Los requisitos funcionales se pueden considerar como características que el usuario detecta.

Ventajas de los Requisitos Funcionales:

1. Claridad en la funcionalidad requerida: Los requisitos funcionales permiten definir claramente las funcionalidades que se esperan de un sistema o software en particular. Esto proporciona una guía precisa para el desarrollo y ayuda a evitar malentendidos entre los desarrolladores y los clientes.
2. Orientación hacia los objetivos del usuario: Al identificar y definir los requisitos funcionales, se puede tener en cuenta la perspectiva y los objetivos del usuario final. Esto ayuda a asegurar que el sistema cumpla con las necesidades y expectativas de los usuarios.
3. Evaluación de la calidad del sistema: Los requisitos funcionales permiten evaluar la calidad del sistema antes de su implementación. Al definir claramente las funcionalidades y los comportamientos esperados, se pueden realizar pruebas y revisiones para asegurarse de que el sistema cumpla con los requisitos establecidos.

Desventajas de los Requisitos Funcionales:

1. Dificultad para capturar todos los requisitos necesarios: Identificar y documentar todos los requisitos funcionales puede ser un desafío. Puede haber funcionalidades subyacentes o necesidades adicionales que los usuarios no mencionan explícitamente, lo que puede llevar a funcionalidades incompletas o incorrectas en el sistema.
2. Requisitos en constante cambio: Los requisitos funcionales pueden cambiar con el tiempo debido a las necesidades y demandas cambiantes de los usuarios o las actualizaciones tecnológicas. Esto puede requerir

ajustes y cambios frecuentes en el sistema, lo que puede ser costoso y llevar a retrasos en el desarrollo.

3. Restricciones de tiempo y recursos: Algunas funcionalidades pueden requerir mucho tiempo, esfuerzo y recursos para desarrollarse y mantenerse. Esto puede limitar la capacidad de los desarrolladores para cumplir con todos los requisitos funcionales, lo que puede dar lugar a compromisos en la calidad o la funcionalidad del sistema.

2.5.1 Requisitos Funcionales del sistema

A continuación, se presentan los Requisitos Funcionales (RF) asociados a la propuesta de solución de este informe.

RF.1 Cargar Fichero

Descripción: El sistema permite importar un archivo en formato (csv) para utilizar como base de datos el cual contiene los siguientes campos: Área de responsabilidad material, Rótulo y Subclasificación.

RF.2 Mostrar formulario para recopilar datos del código QR a generar.

Descripción: El sistema muestra un formulario para recopilar los datos necesarios que se utilizarán para generar un código QR los cuales son: Área de responsabilidad material, Rótulo y Subclasificación.

RF.3 Generar código QR.

Descripción: El sistema permite genera un código QR que contiene los datos recopilados: Área de responsabilidad material, Rótulo y Subclasificación.

RF.4 Generar imagen.

Descripción: El sistema permite generar una imagen en formato (jpg) que contiene el código QR antes generado acompañado de un texto legible con la información asociada a la subclasificación del medio y una imagen del logo de la Universidad de las Ciencias Informáticas.

RF.5 Exportar imagen.

Descripción: El sistema permite exportar la imagen generada en formato (jpg) a una carpeta en el dispositivo móvil.

RF.6 Escaneo de código QR

Descripción: El sistema permite escanear un código QR asociado a un medio básico el cual contiene los siguientes datos: Área de responsabilidad material, Rótulo y Subclasificación.

RF.7 Mostrar Información del escaneo

Descripción: El sistema muestra la información obtenida del escaneo del código QR que contiene los siguientes datos: Área de responsabilidad material, Rótulo y Subclasificación.

RF.8 Mostrar estado del medio.

Descripción: El sistema muestra si el medio básico está en el Área de responsabilidad material asignada según la información obtenida mediante el escaneo haciendo una comprobación con la información de la base de datos.

RF.9 Mostrar mensaje asociado a la ubicación del medio.

Descripción: El sistema muestra un mensaje asociado a la ubicación del medio básico, si se encuentra en el Área de responsabilidad material asignada se muestra un mensaje donde dice que el medio está en su lugar, en caso contrario se muestra un mensaje donde expresa el área donde pertenece según la información de la base de datos.

RF.10 Realizar inventario general.

Descripción: El sistema permite hacer un inventario general abarcando todas las Áreas de responsabilidad material escaneando los códigos QR de entre un 10 % y un 100 % de los medios básicos mostrando una barra de progreso asociada al % de medios inventariados.

RF.11 Realizar inventario por Área de responsabilidad material.

Descripción: El sistema permite seleccionar un Área de responsabilidad material para hacer un inventario de los medios asociados a la misma mediante el escaneo de sus códigos QR.

RF.12 Generar reporte de medios básicos fuera de Área de responsabilidad material.

Descripción: El sistema permite generar un reporte de los medios básicos fuera de su Área de responsabilidad material con los siguientes datos: Fecha y Hora

del reporte, Cantidad de medios básicos fuera de lugar. De cada medio básico fuera de lugar: Área de responsabilidad material donde se encuentra, Área de responsabilidad material donde se pertenece, Rótulo y Subclasificación.

RF.13 Exportar reporte de medios básicos fuera de Área de responsabilidad material.

Descripción: El sistema permite exportar un archivo en formato (pdf) con la información generada en el reporte de medios básicos fuera de Área de responsabilidad material a una carpeta en el dispositivo móvil.

RF.14 Mostrar estadísticas generales.

Descripción: El sistema muestra la cantidad de medios básicos en total agrupados por subclasificación según la información obtenida de la base de datos.

RF.15 Mostrar estadísticas por Área de responsabilidad material.

Descripción: El sistema muestra la cantidad de medios básicos asociados al Área de responsabilidad material seleccionada agrupados por su subclasificación según la información obtenida de la base de datos.

RF.16 Mostrar formulario para recopilar los datos y el tipo de movimiento.

Descripción: El sistema muestra un formulario para recopilar los siguientes datos asociados a los movimientos de medios básicos: Entidad, Centro de costo, Área de responsabilidad, Descripción, Inventario No, Nombre, Cargo y Rótulo del medio básico. También muestra la opción de seleccionar el tipo de movimiento a realizar que se debe escoger entre: Movimiento simple interno, Movimiento simple externo, Movimiento masivo interno y Movimiento masivo externo.

RF.17 Generar reporte de movimiento simple interno.

Descripción: El sistema permite generar un reporte de movimiento simple interno con los datos siguientes del medio básico que se va a mover: Entidad, Centro de costo, Área de responsabilidad, Descripción, Inventario No, Nombre, Cargo y Rótulo del medio básico.

RF.18 Generar reporte de movimiento simple externo.

Descripción: El sistema permite generar un reporte de movimiento simple externo con los datos siguientes del medio básico que se va a mover: Entidad, Centro de costo, Área de responsabilidad, Descripción, Inventario No, Nombre, Cargo y Rótulo del medio básico.

RF.19 Generar reporte de movimiento masivo interno.

Descripción: El sistema permite generar un reporte de movimiento masivo interno con los datos siguientes de los medios básico que se van a mover: Entidad, Centro de costo, Área de responsabilidad, Descripción, Inventario No, Nombre, Cargo y Rótulo de los medios al dorso.

RF.20 Generar reporte de movimiento masivo externo.

Descripción: El sistema permite generar un reporte de movimiento masivo externo con los datos siguientes de los medios básico que se van a mover: Entidad, Centro de costo, Área de responsabilidad, Descripción, Inventario No, Nombre, Cargo y Rótulo de los medios al dorso.

RF.21 Exportar reporte de movimiento

Descripción: El sistema permite exportar un archivo en formato (pdf) con los datos del tipo de movimiento generado a una carpeta del dispositivo móvil.

RF.22 Mostrar tutorial inicial.

Descripción: El sistema muestra un tutorial inicial asociado a las principales funciones de la aplicación. Este tutorial se muestra únicamente al abrir por primera vez la aplicación.

2.6 Requisitos no Funcionales

Las características no funcionales o requisitos no funcionales del sistema suponen una parte necesaria a especificar en cada proceso de desarrollo que se lleve a cabo.

Los requisitos no funcionales (NFR) son las restricciones o los requisitos impuestos al sistema. Especifican el atributo de calidad del software. Los requisitos no funcionales se ocupan de problemas como la escalabilidad, la mantenibilidad, el rendimiento, la portabilidad, la seguridad, la confiabilidad y muchos más. (Visuresolutions, 2023)

Ventajas de los Requisitos no Funcionales:

1. Proporcionan una comprensión integral del sistema: Los requisitos no funcionales permiten una comprensión completa del sistema, incluyendo sus características de rendimiento, seguridad, escalabilidad y experiencia del usuario. Esto es fundamental para la optimización del sistema y para satisfacer las necesidades de los usuarios.
2. Mejoran la calidad del sistema: Los requisitos no funcionales están relacionados con la calidad, fiabilidad y seguridad del sistema, lo que puede mejorar la calidad del sistema y aumentar su aceptación por parte de los usuarios. Además, estos requisitos pueden ayudar a detectar y corregir posibles vulnerabilidades y defectos en el sistema.
3. Permiten la evaluación y comparación de sistemas: Los requisitos no funcionales permiten una comparación objetiva de diferentes sistemas y su evaluación en términos de calidad, eficiencia y otros aspectos importantes para el usuario.

Desventajas de los Requisitos no Funcionales:

1. Definición de requisitos imprecisa: A menudo los requisitos no funcionales pueden ser abstractos y difíciles de definir, lo que puede llevar a una especificación imprecisa o ambigua que dificulta la comprensión de los equipos de desarrollo.
2. La falta de claridad en cuanto a cómo cumplir con los requisitos: Los requisitos no funcionales pueden ser vagos sobre cómo se deben cumplir, lo que puede crear problemas de comunicación y dificultades adicionales en el proceso de desarrollo.
3. Limitaciones en la capacidad de respuesta: Los requisitos no funcionales pueden limitar la capacidad de respuesta de un sistema o software, ya que estos requisitos enfatizan el rendimiento, la seguridad y otros aspectos importantes que pueden ralentizar la capacidad del sistema para responder a las solicitudes de los usuarios.

2.6.1 Requisitos no funcionales del sistema

Versión de Android: el dispositivo donde se instalará la aplicación debe tener instalado una versión de Android igual o posterior a la versión 4.0. Esta es la versión mínima con soporte completo en el SDK utilizado por el equipo de desarrollo.

Interfaces: las interfaces gráficas implementadas deben ser intuitivas y con diseño acorde a las necesidades del cliente.

Seguridad: El sistema debe estar protegido contra el acceso no autorizado.

Mantenimiento: El sistema debe ser fácil de mantener y actualizar.

Usabilidad: El sistema debe ser fácil de usar y comprender.

Tipo de dispositivo: no es preciso que sea un teléfono móvil, también se pueden emplear tabletas o usar una computadora haciendo uso de un emulador de Android.

Hardware: para un correcto funcionamiento de la aplicación el dispositivo donde se encuentre instalada debe contar con los requerimientos de hardware que se describen a continuación. Procesador ARMv6, ARMv7, ARM o ARM64 a 800 MHz o superior, 512 Mb mínimos de memoria RAM, contar con mínimo 512 Mb de memoria ROM y 80 Mb de espacio libre de almacenamiento.

2.7 Tiempo de ejecución del proyecto

La duración del proyecto es un factor fundamental para que el equipo de desarrollo logre completar las historias de usuario en cada iteración. Esta duración se determina sumando los puntos estimados de las historias de usuario que se desarrollarán en cada iteración. Es importante mencionar que cada punto estimado representa una semana ideal de trabajo, con ocho horas diarias durante cinco días consecutivos y sin interrupciones.

La planificación del proyecto se basa en el tiempo necesario para implementar cada historia de usuario. Es por eso que el tiempo de ejecución se utiliza para determinar cuántas historias de usuario se pueden implementar antes de una fecha específica, o cuánto tiempo tomará implementar un conjunto determinado de historias de usuario.

En la siguiente tabla se muestra una estimación del tiempo requerido, basada en las especificaciones de las historias de usuario. Esta estimación brinda una idea de la duración del desarrollo de la solución propuesta en este informe.

Número	Nombre de la historia de usuario	Estimación
HU-1	Cargar fichero	0.5
HU-2	Mostrar formulario para recopilar datos del código QR a generar	0.5
HU-3	Generar y exportar código QR	1
HU-4	Escaneo de código QR	1
HU-5	Realizar inventario	2
HU-6	Mostrar formulario para recopilar datos de movimiento	1
HU-7	Generar y exportar reporte movimientos de medios	2
HU-8	Mostrar tutorial inicial	1
HU-9	Mostrar estadísticas	1

Tabla 8: Estimación del costo en tiempo por historia de usuario. Elaboración propia.

2.8 Iteraciones

El proyecto se divide en iteraciones, donde se implementa un conjunto específico de historias de usuario y se entrega el resultado al cliente al final de cada iteración. Cada iteración tiene una duración máxima de tres semanas. Al final de la última iteración, el sistema estará listo para su lanzamiento. En la tabla a continuación se muestra la distribución de las historias de usuario en cada iteración del proceso de desarrollo de la solución propuesta en este informe.

Iteración	Título de la historia de usuario	Duración total
1	1. Cargar fichero. 2. Mostrar formulario para recopilar datos del código QR a generar 3. Generar y exportar código QR	2
2	4. Escaneo de código QR. 5. Realizar inventario	3

3	6. Mostrar formulario para recopilar datos de movimiento 7. Generar y exportar reporte movimientos de medios	3
4	8. Mostrar tutorial inicial 9. Mostrar estadísticas	2

Tabla 9: Plan estimado de duración de las iteraciones. Elaboración propia.

Los resultados anteriores permiten estimar la duración de cada iteración, utilizando para ello los puntos estimados de las HU correspondientes. Esto permite elaborar un plan de entrega de funcionalidades implementadas al final de cada iteración.

2.9 Plan de entrega

El plan de entrega consiste en establecer, de manera conjunta entre el equipo de desarrollo y el cliente, la duración y fecha de entrega de cada iteración hasta lograr el producto final. En este plan se detalla la fecha fin de cada iteración, mostrando una versión desarrollada del producto en ese momento, hasta lograr el producto final en la fecha establecida. En la siguiente tabla presenta el plan de entrega para las funcionalidades implementadas en cada iteración del proceso de desarrollo de la aplicación Control de Inventario.

Iteraciones	Fecha de entrega
1	8 de septiembre de 2023
2	29 de septiembre de 2023
3	20 de octubre de 2023
4	3 de noviembre de 2023

Tabla 10: Plan de entrega por iteraciones. Elaboración propia.

2.10 Arquitectura de software

La arquitectura de software es un aspecto fundamental en el desarrollo de aplicaciones, ya que determina la estructura y organización del código. Proporciona al equipo de desarrollo una visibilidad estandarizada para guiar el proceso y ayuda en gran medida al aspecto de trabajo grupal. Actualmente

existen varias arquitecturas de software muy conocidas, a continuación, se presentan algunas de ellas.

1. **MVC (Modelo-Vista-Controlador):** Es uno de los patrones de arquitectura más antiguos y populares. En MVC, el Modelo representa los datos y la lógica de negocio, la Vista se encarga de mostrar la interfaz de usuario y el Controlador maneja las interacciones del usuario y coordina la comunicación entre el Modelo y la Vista.

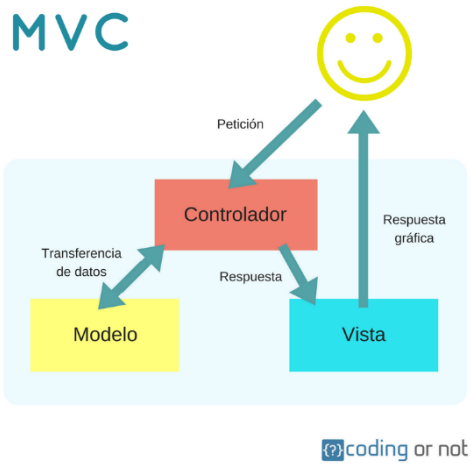


Figura 5: Arquitectura MVC. Fuente (Codingornot, 2023).

2. **MVP (Modelo-Vista-Presentador):** Similar a MVC, pero con una diferencia clave: el Presentador actúa como intermediario entre el Modelo y la Vista, en lugar de ser controlado por el Controlador. El Presentador maneja la lógica de negocio y actualiza la Vista según los cambios en el Modelo.

Figura 6: Arquitectura MVP. Fuente (Apiumhub, 2023).

3. **Clean Architecture:** Es un enfoque moderno que promueve la independencia de las capas de una aplicación. Se basa en el principio de inversión de dependencias y utiliza capas bien definidas, como la capa de presentación, la capa de dominio y la capa de infraestructura, para separar las responsabilidades y facilitar el mantenimiento y la escalabilidad.

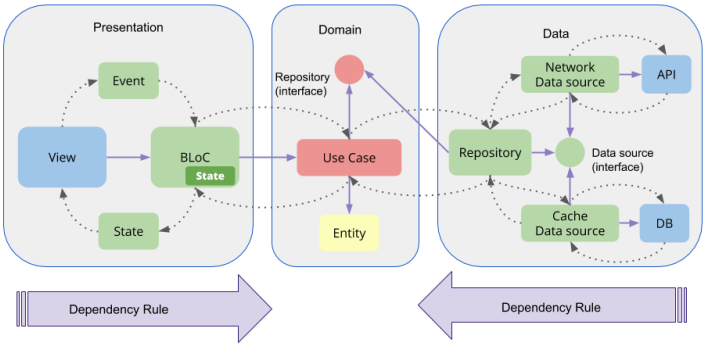


Figura 7: Clean Architecture. Fuente (Xurxodev, 2023).

4. **Microservicios:** En lugar de tener una única aplicación monolítica, los microservicios dividen la aplicación en componentes más pequeños y autónomos, cada uno con su propia lógica de negocio y base de datos. Esto permite una mayor flexibilidad, escalabilidad y desacoplamiento entre los diferentes servicios de una aplicación.

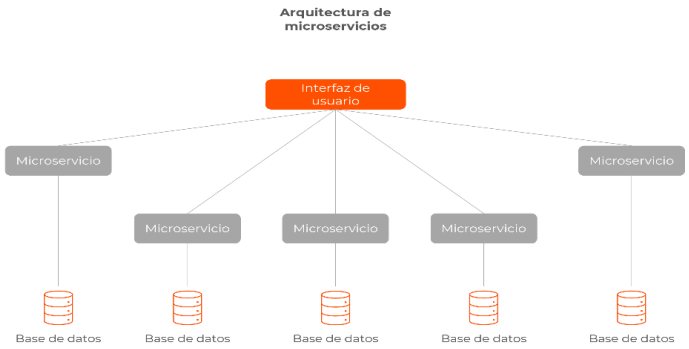


Figura 8: Arquitectura Microservicios. Fuente (Medium.com, 2023).

3 **MVVM:** Es una arquitectura que se ha vuelto muy popular en los últimos años debido a sus ventajas y beneficios. Una de las principales razones por las que es recomendable utilizar MVVM es su capacidad para separar de manera clara y concisa las responsabilidades dentro de una aplicación.

Figura 9: Arquitectura MVVM. Fuente (Techtarget, 2023).

2.9.1 Arquitectura de software seleccionada

Teniendo en cuenta los requisitos necesarios para dar respuesta a la propuesta de solución asociada a este informe, el autor decide utilizar la arquitectura MVVM (Modelo-Vista-ViewModel) para llevar a cabo el desarrollo de la aplicación Control de Inventario por las siguientes razones.

Algunas de las ventajas de utilizar MVVM (Modelo-Vista-ViewModel) como arquitectura para el desarrollo de aplicaciones son:

1. Separación de responsabilidades: MVVM ayuda a mantener una separación clara de las responsabilidades entre los distintos componentes de la aplicación. El modelo se encarga de la lógica y los datos, la vista se encarga de la interfaz de usuario y el ViewModel se encarga de actuar como intermediario entre el modelo y la vista.
2. Reutilización de código: La arquitectura MVVM fomenta la reutilización de código, ya que separa claramente la lógica del negocio y los datos del diseño de la interfaz de usuario. Esto permite que los componentes se puedan reutilizar en diferentes partes de la aplicación o incluso en diferentes aplicaciones, lo que simplifica el desarrollo y mejora la eficiencia.
3. Facilita la colaboración en equipo: Al tener una separación clara de responsabilidades, diferentes desarrolladores pueden trabajar simultáneamente en los diferentes componentes de la aplicación sin afectar el trabajo de los demás. Esto permite una mejor colaboración y una mayor eficiencia en el trabajo en equipo.

4. Soporte de enlace de datos: MVVM se integra muy bien con el enlace de datos, lo que permite una sincronización automática de los datos entre la vista y el ViewModel. Esto simplifica el código y reduce la cantidad de código repetitivo necesario para mantener actualizada la interfaz de usuario.
5. Mantenibilidad y escalabilidad: La arquitectura MVVM ofrece una estructura clara y organizada, lo que facilita la mantenibilidad y escalabilidad de la aplicación. El código se vuelve más modular y fácil de comprender, lo que facilita las futuras actualizaciones y mejoras en la aplicación.

MVVM establece los principios fundamentales para la organización y estructuración de los componentes de una aplicación. Al igual que otras arquitecturas de software, MVVM especifica los roles y responsabilidades de cada componente y cómo interactúan entre sí para lograr los objetivos de la aplicación.

2.10 Patrones de diseño utilizados

Utilizar patrones de diseño en el desarrollo de aplicaciones Android es de suma importancia para lograr un código limpio, modular y de alta calidad. Los patrones de diseño son soluciones probadas y documentadas para problemas comunes en el desarrollo de software, y permiten estandarizar y estructurar el diseño de una aplicación de manera eficiente.

GRASP (General Responsibility Assignment Software Patterns) es un conjunto de patrones de diseño que ayudan en la asignación de responsabilidades a las clases y objetos en un sistema de software. Estos patrones son útiles para diseñar software que sea fácil de entender, mantener y modificar(Gamma Erich et al., 2003).

Utilizar los patrones GRASP en el diseño de software tiene varias ventajas:

1- Mejora el diseño: Los patrones GRASP proporcionan un marco de diseño que ayuda a mejorar la estructura y organización del código. Al seguir los principios y pautas proporcionadas por los patrones GRASP, se obtiene un diseño más claro, modular y mantenible.

2- Aumenta la flexibilidad y la reutilización: Los patrones GRASP promueven un diseño flexible y modular, lo que facilita la reutilización de componentes y módulos en diferentes partes del sistema. Esto permite agregar, modificar o eliminar funcionalidades fácilmente, sin afectar el resto del sistema.
3- Ayuda en la toma de decisiones: Los patrones GRASP proporcionan un conjunto de lineamientos y principios que ayudan a tomar decisiones de diseño de manera más informada. Al seguir estos patrones, se pueden evaluar diferentes opciones y elegir la solución más adecuada para cada caso.
4- Promueve el bajo acoplamiento y la alta cohesión: Los patrones GRASP enfatizan la importancia de tener un bajo acoplamiento entre los objetos y una alta cohesión entre las responsabilidades. Esto ayuda a reducir las dependencias entre los objetos y mejora el modularidad y la escalabilidad del sistema.
5- Facilita la comunicación y comprensión del diseño: Los patrones GRASP proporcionan un vocabulario común y una forma de comunicar y documentar el diseño. Esto facilita la comunicación entre los miembros del equipo de desarrollo y mejora la comprensión y el mantenimiento del código a lo largo del tiempo.

A continuación, se presentan los patrones GRASP utilizados en el desarrollo de la propuesta de solución:

- **Experto**: asigna la responsabilidad a la clase o al objeto que tiene la información o los conocimientos necesarios para realizar una tarea en particular.
- **Creador:** asigna la responsabilidad de crear objetos a la clase o al objeto que tiene la información necesaria para configurar correctamente el nuevo objeto.
- **Controlador:** asigna la responsabilidad de manejar y coordinar las interacciones entre objetos a una clase llamada "Controller". Esta clase es responsable de recibir las solicitudes, tomar decisiones y comunicarse con otros objetos según sea necesario.

- **Bajo Acoplamiento:** busca minimizar las dependencias entre objetos para que los cambios en un objeto no afecten a otros. Esto se logra mediante el diseño de interfaces claras y cohesivas entre objetos y evitando acoplamientos fuertes.
- **Alta Cohesión:** busca asegurar que una clase tenga una única y bien definida responsabilidad. Una alta cohesión significa que las clases y los métodos son más fáciles de entender, mantener y reutilizar.

En general, utilizar los patrones GRASP en el diseño de software ayuda a crear sistemas más robustos, flexibles y fáciles de mantener, lo que conduce a un desarrollo de software más eficiente y exitoso.

2.11 Mapa de navegación de la aplicación

Un mapa de navegación de aplicaciones Android es una herramienta visual que muestra la estructura y la jerarquía de las diferentes pantallas o vistas de una aplicación específica de Android. Es una representación gráfica que permite visualizar cómo los usuarios pueden navegar por las diferentes secciones y funcionalidades de la aplicación.

Las ventajas de utilizar un mapa de navegación para aplicaciones Android incluyen:

1. Claridad en la estructura: Al mapear todas las pantallas y vistas en un solo lugar, el mapa de navegación proporciona una visión clara y organizada de la estructura de la aplicación. Esto facilita la comprensión de cómo los usuarios pueden moverse a través de las diferentes secciones y ayuda a identificar cualquier confusión o falta de coherencia en el diseño.

2. Mejora de la experiencia del usuario: Al definir y visualizar las rutas de navegación, el mapa de navegación ayuda a garantizar que los usuarios puedan acceder fácilmente a todas las funcionalidades de la aplicación. Esto mejora la experiencia del usuario al proporcionar una navegación intuitiva y eficiente, evitando que los usuarios se pierdan o se sientan frustrados al no encontrar lo que buscan.

3. Identificación de problemas de usabilidad: Al revisar el mapa de navegación de la aplicación, es posible detectar y corregir cualquier problema de usabilidad, como la falta de fluidez en la navegación, el desorden en el diseño, o la falta de consistencia en los estilos y características de la interfaz.

4. Reducción del tiempo de desarrollo: El mapa de navegación ayuda a definir la estructura de la aplicación antes de comenzar el proceso de desarrollo. De esta manera, al tener un esquema bien definido, el equipo puede enfocarse en la creación de cada pantalla o vista de forma más detallada, sin perder tiempo tratando de descubrir la estructura básica de la aplicación.

5. Mejora de la comunicación entre el equipo: El mapa de navegación es una herramienta visual que puede ser compartida y discutida entre los miembros del equipo de desarrollo y diseñadores. Esto fomenta la colaboración, ayuda a alinear las expectativas y permite a todos los involucrados tener una comprensión común de la estructura y las funcionalidades de la aplicación.

A continuación, se muestra el mapa de navegación asociado a la propuesta de solución planteada.

Figura 10: Mapa de Navegación - Aplicación Control de Inventario. Elaboración Propia.

2.12 Tarjetas Clase-Responsabilidad-Colaborador

Scott Ambler (Ambler, 2002), define las tarjetas Clase-Responsabilidad-Colaborador (CRC) como tarjetas de índice estándar que han sido divididas en tres secciones, una que indica el nombre de la clase que la tarjeta representa,

un inventario de las responsabilidades de la clase, y la tercera con los nombres de las otras clases con las que ésta colabora para cumplir sus responsabilidades. En el Manifiesto para el Desarrollo Ágil de Software (Beck et al., 2001), se recomienda el uso de este artefacto para inventariar las clases que son relevantes para el incremento actual del software. A continuación, se muestran las tarjetas CRC elaboradas para el modelo de la solución propuesta en el presente trabajo.

Clase: GeneradorQR	
Responsabilidad: Generar códigos QR para nuevos medios básicos.	**Colaborador:** • MedioBasico

Tabla 11: Tarjeta CRC-1 Generar código QR. Elaboración propia.

Clase: EscannerQR	
Responsabilidad: Escanea códigos QR de los medios básicos.	**Colaborador:** • MedioBasico

Tabla 12: Tarjeta CRC-2 - Escanear código QR. Elaboración propia.

Clase: MostrarEstadisticas	
Responsabilidad: Muestra estadísticas asociadas a los medios básicos.	**Colaborador:** • FicheroDeDatos

Tabla 13: Tarjeta CRC-3 - Mostrar estadísticas. Elaboración propia.

Clase: Inventario	
Responsabilidad: Generar inventario de medios básicos.	**Colaborador:** • MedioBasico • QRCode • FicheroDeDatos

Tabla 14: Tarjeta CRC-4 - Inventario. Elaboración propia.

Clase: GenerarReportes

Responsabilidad: Generar reportes de inventario.	**Colaborador:** • QRCode • MedioBasico

Tabla 15: Tarjeta CRC-5 - Generar reportes. Elaboración propia.

Clase: GenerarMovimientos	
Responsabilidad: Generar movimiento de acuerdo a su tipo.	**Colaborador:** • QRCode • MedioBasico

Tabla 16: Tarjeta CRC-6 - Generar Movimientos. Elaboración propia.

2.13 Modelo de datos

El modelo de datos utilizado en la aplicación Control de Inventario se basa en una base de datos creada con SQLite. Esta base de datos contiene información de los medios básicos, específicamente los datos de rótulo, subclasificación y área, todos almacenados como campos de tipo varchar. La aplicación obtiene esta información directamente del fichero (csv) que se carga durante el inicio de la aplicación. Así, la información se encuentra disponible para que se usuario administre y mantenga el inventario de medios de manera más eficiente y organizada. A continuación, se muestra el modelado de datos en la Figura 10.

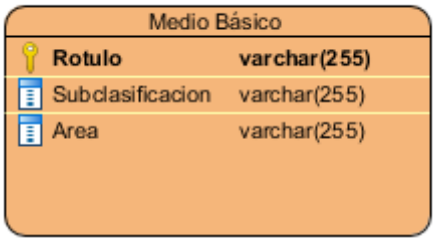


Figura 11: Modelo de datos. Elaboración propia

Conclusiones del capítulo

En este capítulo se abordaron temas con un alto impacto en el desarrollo asociado a la propuesta de solución. Se generaron los diagramas y artefactos que establece la metodología de desarrollo de software utilizada en la investigación. Además, se profundizó en la presentación de conceptos asociados al tema para lograr un correcto entendimiento y evolución en el desarrollo.

Capítulo 3. Implementación y Prueba

Introducción al capítulo

En el presente capítulo se exponen las principales características de las dos últimas fases del ciclo de vida de la metodología XP: desarrollo y pruebas. Se procede a desarrollar las tareas de la ingeniería que responden a las historias de usuario abordadas en cada iteración, luego mediante las pruebas de aceptación se verifica que el producto resultante cumpla con los requerimientos definidos.

3.1 Tareas de ingeniería

Las tareas de ingeniería de software son las actividades o acciones específicas que se llevan a cabo durante el desarrollo de software, con el objetivo de lograr los objetivos del proyecto. Estas tareas se enfocan en distintas áreas de conocimiento y se realizan de manera iterativa e incremental, adaptándose al ciclo de vida del software y a las necesidades específicas del proyecto (Pressman, 2010).

A continuación, se presentan las tareas de ingeniería agrupadas por cada historia de usuario asociada a la propuesta de solución.

Historia de Usuario	Tareas de ingeniería
Cargar Fichero	1. Desarrollar una interfaz asociada a la función de cargar fichero de datos. 2. Desarrollar una función que permita cargar fichero de datos. 3. Desarrollar una base de datos para guardar los datos del fichero.
Mostrar formulario para recopilar datos del código QR a generar	1. Desarrollar una interfaz visual asociada a la función de recopilación de datos. 2. Desarrollar una función que permita recopilar los datos.
Generar y exportar código QR	1. Desarrollar una función que permita generar el código QR. 2. Desarrollar una función que permita exportar el código QR.
Escaneo de código QR	1. Desarrollar una interfaz visual asociada a la función de escaneo de códigos QR.

	2. Desarrollar una función que permita escanear códigos QR. 3. Desarrollar una función que permita consultar el estado del medio.
Realizar inventario	1. Desarrollar una interfaz visual asociada a la función de mostrar información del inventario. 2. Desarrollar una función que permita realizar un inventario de medios básicos. 3. Desarrollar una función que permita hacerle consultas a la base de datos para realizar el inventario. 4. Desarrollar función asociada a la generación y exportación de los datos recopilados del inventario.
Mostrar formulario para recopilar datos de movimiento	1. Desarrollar una interfaz visual asociada a la función de recopilación de datos. 2. Desarrollar una función que permita recopilar los datos.
Generar y exportar reporte movimientos de medios	1. Desarrollar una función que permita generar los reportes de movimientos. 2. Desarrollar una función que permita exportar los reportes de movimientos.
Mostrar tutorial inicial	1. Desarrollar una interfaz visual con animaciones que muestre un tutorial inicial. 2. Desarrollar una función que permita mostrar el tutorial una única vez al abrir por primera vez la aplicación.
Mostrar estadísticas	1. Desarrollar una interfaz visual que muestre estadísticas relacionadas a los medios básicos. 2. Desarrollar una función asociada a la consulta realizada a la base de datos para mostrar estadísticas.

Tabla 17: Tares de Ingeniería agrupadas por HU. Elaboración propia.

A continuación, se evidencian las principales tareas ingeniería o programación en las que fueron desglosadas las HU mencionadas anteriormente, las restantes se encuentran en los anexos. Para la confección de cada una de las tareas se utilizó la tabla que cuenta con los siguientes campos:

- **No. de tarea**: Numeración continua que identifica a la tarea.
- **No. de HU**: Número de la HU a la cual pertenece.
- **Nombre de la tarea**: Identificación literal de la tarea.
- **Tipo de tarea**: Clasificación de la tarea.
- **Puntos estimados**: Representación en porciento de la cantidad de tiempo estimada de una semana, que se utilizara para su realización.
- **Fecha inicio**: Fecha estimada de inicio de realización.

Tarea de ingeniería	
No. de tarea: 3	**No. de UH:** 4
Nombre de la tarea: Desarrollar una función que permita consultar el estado del medio.	
Tipo de tarea: Desarrollo	**Responsable:** Johan Alejandro Falcón Martínez
Fecha inicio: 13/09/2023	**Fecha fin:** 15/09/2023
Descripción: Se desarrollan las funcionalidades asociadas a la consulta del estado del medio.	

Tabla 18: Tarea de Ingeniería No.3 HU_4. Elaboración propia

Tarea de ingeniería	
No. de tarea: 2	**No. de tarea:** 5
Nombre de la tarea: Desarrollar una función que permita realizar un inventario de medios básicos.	
Tipo de tarea: Desarrollo	**Responsable:** Johan Alejandro Falcón Martínez
Fecha inicio: 20/09/2023	**Fecha inicio:** 22/09/2023

Descripción: Se desarrollan las funcionalidades necesarias para la realización del inventario.

Tabla 19: Tarea de Ingeniería No.2 HU_5. Elaboración

3.2 Estándares de codificación

En el desarrollo de aplicaciones para Android, es recomendable seguir los estándares y convenciones establecidos por Google en su guía oficial para desarrolladores de Android (Android Developers, 2023). En el caso de la propuesta de solución se utilizó el lenguaje de programación Dart creado por dicha empresa. A continuación, se presentan algunos de los principales estándares y convenciones de codificación para Dart:

1. Nomenclatura de variables y funciones: Utiliza nombres descriptivos y significativos para las variables, funciones y métodos. Sigue la convención de nomenclatura camelCase para nombres de variables y funciones (por ejemplo, miVariable, calcularResultado()). Para los nombres de clases, utiliza UpperCamelCase (por ejemplo, MiClase).
2. Estilo de código: Utiliza sangría de 2 espacios para el código y evita el uso de tabulaciones. Asegura que el código esté bien formateado y sea legible. Además, utiliza espacios en blanco de manera consistente para mejorar la claridad del código.
3. Comentarios y documentación: Utiliza comentarios para explicar el propósito y el funcionamiento del código. Documenta las clases, métodos, parámetros y variables utilizando comentarios de estilo de documentación de Dart. Esto ayuda a otros desarrolladores a comprender tu código y facilita futuras modificaciones o colaboraciones.
4. Organización del código: Organiza el código en archivos y carpetas de manera lógica. Separa diferentes funcionalidades en archivos individuales y utiliza carpetas para agrupar archivos relacionados. Mantiene una estructura de proyecto coherente y fácil de seguir.
5. Importaciones: Organiza importaciones de manera ordenada y coherente. Agrupa las importaciones en tres secciones: paquetes de Dart, paquetes

de terceros y archivos locales. Utiliza rutas relativas para los archivos locales y evita las importaciones no utilizadas.

6. Convenciones de estilo específicas de Dart: Sigue las convenciones específicas de Dart, como utilizar llaves en nuevas líneas para estructuras de control y declaraciones de funciones. Utiliza el operador => para funciones de una sola línea. Evita el uso de punto y coma al final de las declaraciones.
7. Pruebas unitarias: Escribe pruebas unitarias utilizando el framework de pruebas integrado de Dart, llamado test. Mantiene las pruebas cerca del código fuente correspondiente y sigue las mejores prácticas para asegurar una cobertura adecuada y una validación efectiva del código.

3.3 Interfaces de usuario de la aplicación

En el desarrollo de aplicaciones Android, una interfaz de usuario (UI, por sus siglas en inglés) se refiere a la manera en que los usuarios interactúan con la aplicación a través de elementos visuales, como botones, campos de texto, menús y otros componentes gráficos. La UI es responsable de presentar la información y proporcionar los controles necesarios para que los usuarios puedan utilizar la aplicación de manera intuitiva y eficiente. A continuación, se presentan las principales UI de la aplicación *Control de Inventario*.

Figura 12: Interfaces - Tutorial Inicial. Elaboración propia.

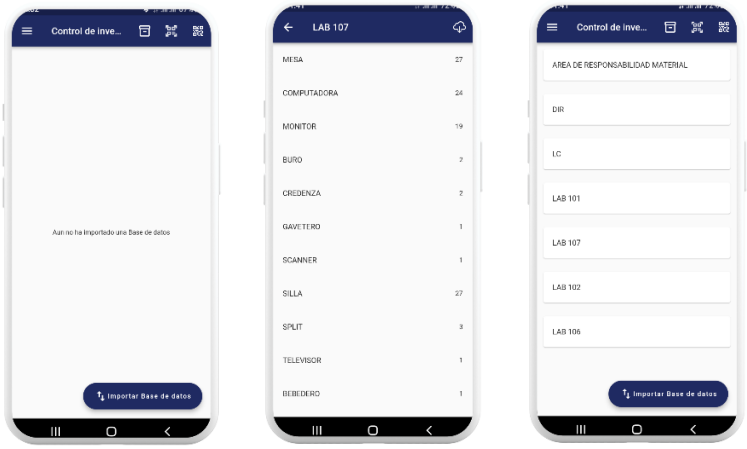


Figura 13: Interfaces - Cargar fichero y muestra de estadísticas. Elaboración propia.

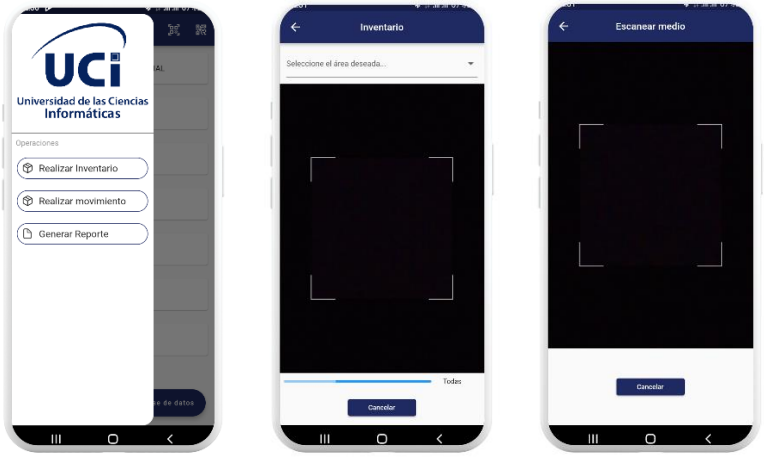


Figura 14: Interfaces - Realizar inventario. Elaboración propia.

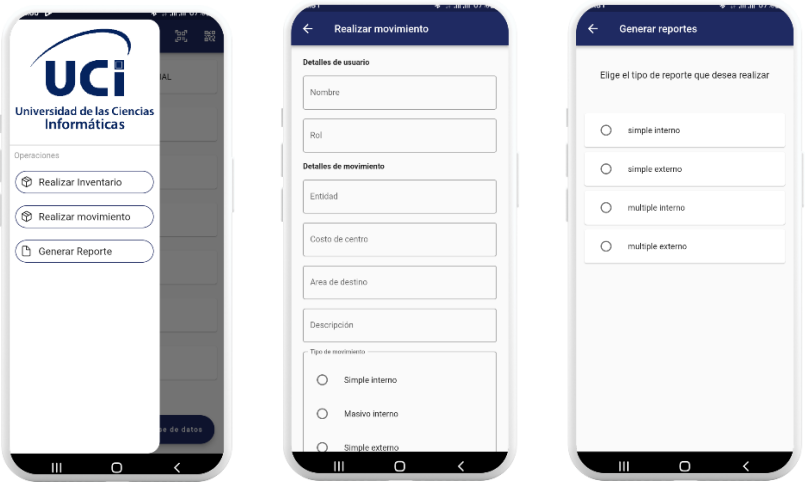


Figura 15: Interfaces - Realizar movimiento. Elaboración propia.

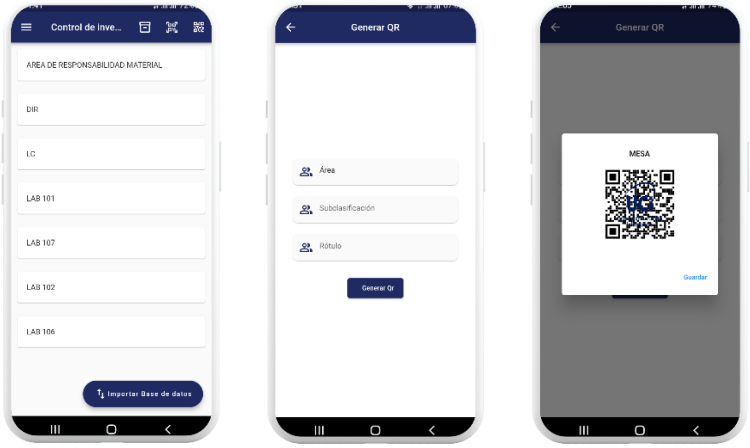


Figura 16: Interfaces - Generar código QR. Elaboración propia.

3.4 Pruebas de Software

Según la séptima edición del libro "Ingeniería del software: un enfoque práctico" de Roger S. Pressman, las pruebas de software se definen como actividades realizadas para evaluar un sistema o componente de software con el propósito de descubrir errores, verificar si el software satisface los requerimientos y validar que el sistema cumple con su propósito previsto. El concepto de pruebas de software implica la realización de actividades planificadas y sistemáticas que involucran la ejecución de programas o componentes de software y el análisis de los resultados obtenidos. El objetivo principal de las pruebas de software es identificar defectos o errores en el software antes de su liberación, y así asegurar la calidad y confiabilidad del sistema (Pressman, 2010).

Las pruebas de software son una parte fundamental del proceso de desarrollo y se realizan con el propósito de:

1 Descubrir defectos: Las pruebas de software buscan identificar errores o defectos en el software, tanto en la implementación de los requerimientos como en el comportamiento del sistema.

2 Validar el cumplimiento de los requerimientos: Las pruebas se utilizan para verificar que el software cumple con los requerimientos establecidos, asegurando que el sistema funcione correctamente y cumpla con su propósito previsto.

3 Evaluar la calidad del software: Las pruebas permiten evaluar la calidad del software en términos de funcionalidad, rendimiento, usabilidad, seguridad y otros atributos de calidad.
4 Mitigar riesgos: Las pruebas ayudan a identificar y mitigar riesgos asociados con la operación del software, como errores que pudieran causar daños o pérdidas.
5 Mejorar y validar el sistema: Las pruebas proporcionan información valiosa sobre la calidad del software y permiten tomar acciones correctivas para mejorar y validar el sistema antes de su liberación.

3.5 Estrategia de pruebas

La estrategia de prueba abarca distintos aspectos clave del proceso de desarrollo de software. Con el objetivo de garantizar la calidad del producto final, se ha decidido implementar pruebas unitarias, pruebas de rendimiento y pruebas de aceptación. A continuación, se describe brevemente cada una de estas estrategias de prueba:

Pruebas Unitarias

Las pruebas unitarias se centran en verificar el correcto funcionamiento de las unidades individuales de código. Para ello, se han desarrollado casos de prueba que cubren diferentes escenarios y condiciones posibles. Estas pruebas se ejecutan de forma automatizada y se integran en el proceso de desarrollo, lo que permite detectar errores a medida que se escriben y modifican las unidades de código. Además, las pruebas unitarias sirven como una forma efectiva de documentar y comprender el comportamiento esperado de cada componente del sistema.

Pruebas de Rendimiento

Con el objetivo de evaluar la capacidad de respuesta y eficiencia del sistema en diferentes condiciones de carga y estrés se realizan las pruebas de rendimiento tomando como muestra una selección de dispositivos móviles. Estas pruebas permiten optimizar el rendimiento y garantizar que el sistema sea capaz de manejar la carga esperada. Se miden aspectos esenciales como la utilización de

recursos, el impacto en el consumo de batería y otros indicadores de rendimiento relevantes.

Pruebas de Aceptación:

Las pruebas de aceptación son fundamentales para validar que el software cumpla con los requisitos y expectativas del cliente. Se trabaja en estrecha colaboración con los usuarios finales para identificar los escenarios de prueba más relevantes. Estos casos de prueba se basan en situaciones reales de uso y se centran en verificar que el sistema funcione correctamente, cumpla con los criterios de aceptación y proporcione una experiencia satisfactoria para los usuarios.

3.6 Pruebas Unitarias Caja Blanca

Las pruebas unitarias de caja blanca, también conocidas como pruebas de caja clara o pruebas estructurales, son un tipo de prueba de software en el que se examinan los componentes internos de un programa de manera detallada. Se centran en probar el funcionamiento interno del código, evaluando la lógica individual de las unidades de software, como funciones, métodos o clases. Este tipo de pruebas permite verificar que cada componente funcione correctamente, evaluar la cobertura del código, comprobar si se cumplen las condiciones de flujo de control (Pressman, 2010).

Para el caso de la propuesta de solución se selecciona el tipo de pruebas unitarias de camino básico el cual se utilizada para probar el flujo lógico de un programa. Estas pruebas se centran en probar todas las posibles rutas o caminos a través del código. El objetivo de las pruebas unitarias de camino básico es asegurarse de que todas las rutas posibles del código hayan sido probadas al menos una vez. Para esto, se identifican los caminos posibles a través del código.

El proceso para realizar las pruebas unitarias de camino básico en la propuesta de solución comprende los siguientes pasos.

1. Identificar los caminos o rutas posibles del código: Esto implica analizar el código y determinar todas las posibles combinaciones de caminos que pueden ser tomados durante la ejecución.

2. Diseñar casos de prueba para cubrir cada camino: Para cada uno de los caminos identificados, se deben diseñar casos de prueba que permitan probar ese camino específico.
3. Ejecutar las pruebas: Una vez que se tienen los casos de prueba diseñados, se ejecutan para verificar si el programa se comporta correctamente en cada uno de los caminos.
4. Analizar los resultados: Se analizan los resultados de las pruebas para determinar si el programa pasó o falló en alguno de los caminos. En caso de falla, se deben identificar y corregir los errores.

Es importante tener en cuenta que las pruebas unitarias de camino básico no cubren necesariamente todos los posibles escenarios de uso del programa. Estas pruebas se enfocan en probar el flujo lógico interno del código, pero no necesariamente prueban su interoperabilidad con otros componentes o sistemas.

En el caso de la propuesta de solución se utilizaron herramientas propias proporcionadas por el Framework Flutter dedicadas a la realización de este tipo de pruebas.

1. Flutter Test Framework: Framework de prueba incorporado que permite realizar pruebas unitarias en aplicaciones Flutter. Proporciona una variedad de clases y funciones para crear y ejecutar casos de prueba.
2. WidgetTester: Permite realizar pruebas de widget para probar el comportamiento y la apariencia de los widgets en el árbol de widgets de Flutter.

```dart
import 'package:flutter_test/flutter_test.dart';
import 'package:qr_code_scanner/qr_code_scanner.dart';
import 'package:my_project/inventory_screen.dart';

void main() {
  group('Inventory Screen tests', () {
    QRViewController controller;

    setUp(() async {
      // Configura un controlador QR para las pruebas
      controller = MockQRViewController();
    });

    test('Barcode scanning test', () {
      // Configura una instancia de InventoryScreen
      final inventoryScreen = InventoryScreen(controller: controller);

      // Genera un código QR de prueba
      final barcodeData = 'ABC123';

      // Simula el escaneo de un código QR
      inventoryScreen.onScannedBarcode(barcodeData);

      // Verifica que el código escaneado se agregue a la lista de inventario de la pantalla
      expect(inventoryScreen.inventory.contains(barcodeData), isTrue);
    });
  });
}

class MockQRViewController implements QRViewController {
  // Implementa los métodos necesarios de QRViewController para las pruebas
}
```

Figura 17: Realización de pruebas unitarias a la aplicación Control de Inventario

Realización de pruebas unitarias en la pantalla de inventario (InventoryScreen) de la aplicación, que escanea códigos QR utilizando el paquete qr_code_scanner. Primero, se una clase de controlador QR de prueba (MockQRViewController) que implementa los métodos necesarios de QRViewController para las pruebas. En el método setUp(), se inicializa un controlador QR de prueba para usar en las pruebas. Luego, se define una función que prueba el escaneo de un código QR simulado utilizando InventoryScreen.onScannedBarcode(). Se verifica que el código escaneado haya agregado a la lista de inventario de la pantalla usando expect().

```
$ flutter test test/inventario_qr_test.dart
00:00 +0: Inventario QR Prueba de escaneo de QR
Prueba de escaneo de QR satisfactoria
00:00 +1: Prueba Inventario
Prueba Inventario satisfactoria
00:00 +2: Prueba de Consulta de Inventario
Prueba de Consulta de inventario satisfactoria
00:00 +3: Prueba de Generación de Reportes
Prueba de Generación de Reportes satisfactoria
00:00 +4: Prueba de Generación de Movimientos
Prueba de Generación de Movimientos satisfactoria
00:00 +5: All tests passed!
```

Figura 18: Resultados de pruebas unitarias a la aplicación Control de Inventario

3.7 Pruebas de rendimiento

De acuerdo a las características de la aplicación se decide realizar pruebas de rendimiento para comprobar el desempeño de la solución bajo distintos entornos de trabajo. Para llevar a cabo estas pruebas se utilizaron los siguientes dispositivos móviles:

Dispositivo	Versión de Android	CPU	RAM	Pantalla
Alcatel One Touch 4030A	4.1.1	1 *core @ 1.0 GHz*	512 MB	3.5”, 165 dpi
Samsung Galaxy S4 i337	4.4.2	4 *cores @ 1.9 GHz*	2 GB	4.7”, 420 dpi
Samsung Galaxy S5 G900A	6.0.1	4 *cores @ 2.4 GHz*	2 GB	5.2“, 480 dpi
Huawei Ascend XT2 H1711	6.0.1	8 *cores @ 1.4 GHz*	2 GB	5.5“, 420 dpi
BLU Studio 5.0 II	4.2.2	2 *cores @ 1.3 GHz*	512 MB	5.0“, 220 dpi
Samsung Galaxy J3 2016	5.1.1	4 *cores @ 1.5 GHz*	1.5 GB	5.0“, 370 dpi
Samsung Galaxy Tab SIII	4.1.2	2 *cores @ 1.2 GHz*	2 GB	7.0“, 170 dpi

Tabla 20: Muestra de dispositivos escogidos para las pruebas de rendimiento. Elaboración propia.

En la comparación se tuvo en cuenta el uso de los siguientes recursos: tiempo de uso, batería, microprocesador, memoria RAM.

3.7.1 Resultados de las pruebas de rendimiento

En esta etapa de las pruebas se pudo analizar el comportamiento de los dispositivos móviles en la ejecución de la aplicación. A continuación, se muestran los resultados obtenidos:

	Alcatel One Touch	***Samsung Galaxy S4***	***Samsung Galaxy S5***	***Huawei Ascend XT2***	***BLU Studio 5.0 II***	***Samsung Galaxy J3 2016***	***Samsung Galaxy Tab SIII***
Tiempo de uso	35 minutos	30 minutos	1 hora y 12 minutos	17 minutos	40 minutos	11 minutos	1 hora 17 minutos
Consumo de batería (%)	12%	9%	7%	3%	11%	6%	22%
Microprocesador (%)	19,13%	2,33 %	0,98 %	2,98%	9,35%	6,12%	11,13%
Memoria RAM (en MB)	49,33 MB	19,90 MB	11,21 MB	21,12 MB	40,84 MB	23,32 MB	42,45 MB

Tabla 21: Resultados de las pruebas de rendimiento. Elaboración propia.

Después de hacer pruebas de rendimiento en los dispositivos tomados como muestra, se comprobó que la solución es factible para dispositivos tanto de altas prestaciones, como para dispositivos con pocos recursos de hardware y con un software desactualizado. Se comprobó que los elementos en la pantalla se distribuyen de manera uniforme, independientemente del tamaño y la resolución de pantalla del dispositivo usado. El consumo de memoria RAM, CPU y batería es el adecuado para una aplicación de este tipo, permitiendo validar el uso de la propuesta de solución.

3.8 Pruebas de aceptación

En el contexto de eXtreme Programming (XP), las pruebas de aceptación se refieren a un enfoque específico para validar si una funcionalidad o característica desarrollada cumple con los requisitos y expectativas del cliente o usuario final. Estas pruebas se realizan en colaboración con los clientes o usuarios y se

centran en verificar que el software cumpla con sus necesidades y sea aceptable para su uso. Las pruebas de aceptación en XP se enfocan en la creación de escenarios realistas que simulan situaciones reales en las cuales el software será utilizado. Estas pruebas se definen en conjunto con los clientes o usuarios y en las Historias de Usuario antes descritas y criterios de aceptación. El objetivo principal es demostrar que el software cumple con los criterios acordados y que es capaz de satisfacer las necesidades del cliente. En XP, las pruebas de aceptación se consideran parte integral del proceso de desarrollo y se realizan de manera continua a lo largo del proyecto. Estas pruebas se ejecutan desde la perspectiva del usuario, y pueden incluir casos de prueba, pruebas exploratorias y pruebas de usabilidad(Beck, 2000).

A continuación, se muestran algunos de los casos de prueba de aceptación correspondientes a una historia de usuario de cada iteración, el resto se encuentran en los anexos del presente documento.

Caso de prueba de aceptación	
Código: HU1_CP1	**Historia de usuario:** Cargar Fichero.
Nombre: Cargar Fichero.	
Descripción: Se prueba la funcionalidad de cargar el fichero de datos proporcionado por el centro para cargar en la base de datos de la aplicación.	
Condiciones de ejecución: El usuario debe tener en el almacenamiento del dispositivo el archivo con formato (csv) proporcionado por el centro.	
Entrada/Pasos de ejecución: • Se selecciona el archivo en la ruta existente. • Se importa el archivo.	
Resultado esperado: • Se muestra una interfaz con los datos obtenidos de la consulta a la base de datos.	
Evaluación: Satisfactoria.	

Tabla 22: HU1_CP1 Cargar Fichero. Elaboración propia.

Caso de prueba de aceptación

Código: HU4_CP4	**Historia de usuario:** Escaneo de código QR.
Nombre: Escaneo de código QR válido.	
Descripción: Se prueba la funcionalidad asociada al escaneo de códigos QR y su estado escaneando un código válido.	
Condiciones de ejecución: El usuario debe escanear un código QR valido.	
Entrada/Pasos de ejecución: • Se escanea un código QR asociado a un medio básico.	
Resultado esperado: • Se escanea correctamente el código QR y se muestra un mensaje sobre el estado del medio básico.	
Evaluación: Satisfactoria.	

Tabla 23: HU4_CP4 Escaneo de código QR. Elaboración propia.

Caso de prueba de aceptación	
Código: HU5_CP6	**Historia de usuario:** Realizar inventario.
Nombre: Realizar inventario	
Descripción: Se prueba la funcionalidad asociada a la realización del inventario.	
Condiciones de ejecución: El usuario debe escanear códigos QR válidos.	
Entrada/Pasos de ejecución: • Se escanean los códigos QR asociado a los medios básicos. • Se realiza un inventario.	
Resultado esperado: • Se realiza un inventario asociado a los medios básicos escaneados.	
Evaluación: Satisfactoria.	

Tabla 24: HU5_CP6 Realizar inventario. Elaboración propia.

Caso de prueba de aceptación	
Código: HU7_CP9	**Historia de usuario:** Generar y exportar reporte movimientos de medios.
Nombre: Generar y exportar reporte movimientos de medios.	

Descripción: Se prueba la funcionalidad asociada a la generación y exportación de los reportes asociados al tipo de movimiento seleccionado.
Condiciones de ejecución: El usuario ingresa datos válidos.
Entrada/Pasos de ejecución: • Datos asociados a los movimientos. • Tipo de movimiento.
Resultado esperado: • Se genera un reporte asociado al movimiento seleccionado. • Se exporta un reporte asociado al movimiento seleccionado.
Evaluación: Satisfactoria.

Tabla 25: HU7_CP9 Generar y exportar reporte movimientos de medios. Elaboración propia.

Caso de prueba de aceptación	
Código: HU9_CP11	**Historia de usuario:** Mostrar estadísticas.
Nombre: Mostrar estadísticas.	
Descripción: Se prueba la funcionalidad asociada a la muestra de estadísticas.	
Condiciones de ejecución: El usuario debe haber ingresado previamente el fichero de datos.	
Entrada/Pasos de ejecución: • Importar fichero de datos	
Resultado esperado: • Se muestra una interfaz con las estadísticas asociadas a los medios básicos existentes.	
Evaluación: Satisfactoria.	

Tabla 26: HU9_CP11 Mostrar estadísticas. Elaboración propia.

3.8.1 Resultados de las pruebas de aceptación

Las pruebas aplicadas contribuyeron a mejorar la calidad y el funcionamiento del sistema. A continuación, se muestra una gráfica resumen con la cantidad de errores detectados por cada iteración clasificados en Errores de Interfaz, Errores de Funcionalidades, Errores Menores y Recomendaciones por parte del cliente. Ver resumen en la Figura 19.

Errores de Interfaz: Son errores de tipo visual o de diseño.

Errores de Funcionalidades: Son los errores asociados a funcionalidades de la aplicación.

Errores Menores: Son los errores que no afectan el funcionamiento de la aplicación tales como errores ortográficos, coherencia de diálogos, etc.

Recomendaciones: Notas y sugerencias por parte del cliente.

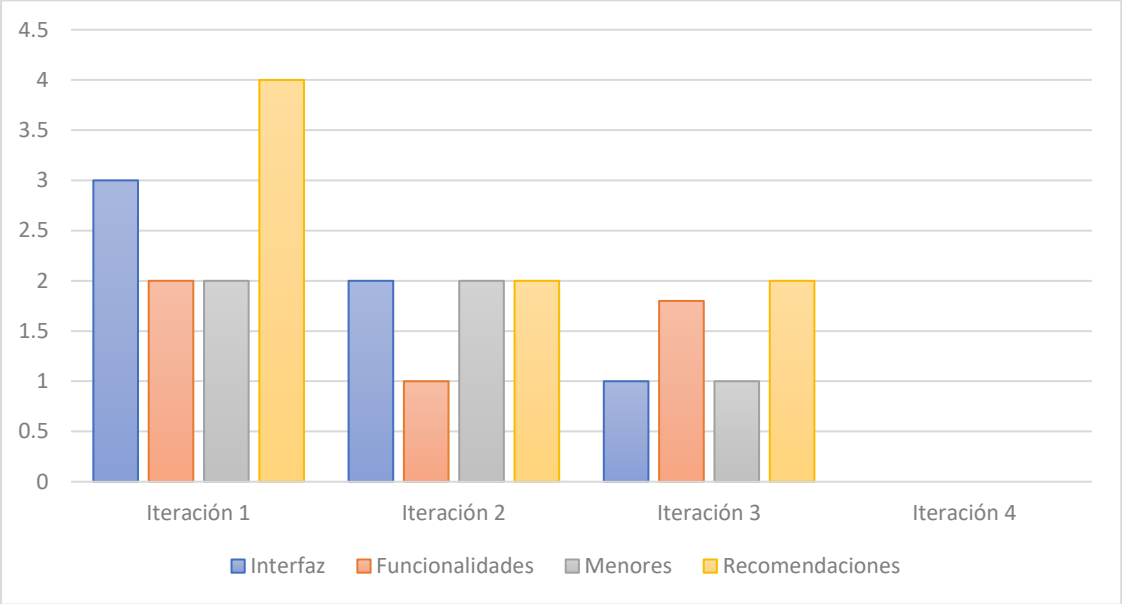


Figura 19: Gráfica de no conformidades. Elaboración propia

Tras obtener en la cuarta iteración los resultados requeridos se aprueba por parte del cliente y el desarrollador el uso de la aplicación móvil *Control de Inventario* para los procesos de esta índole en el centro de desarrollo CIGED.

Conclusiones del capítulo

Tras definir las características de implementación y pruebas de la propuesta de solución, se concluye que se generaron los artefactos pertinentes a las etapas de la metodología seleccionada asociadas al capítulo. Además, se diseñaron los casos de prueba para rectificar a tiempo defectos que pudiera presentar el sistema. Se validó la propuesta de solución a través de la estrategia de pruebas planteada lo que permite el cumplimiento de los requisitos demandados por el cliente y la liberación final del producto listo para su uso.

Conclusiones Generales

Con la culminación de presente Trabajo de Diploma se da cumplimiento al objetivo propuesto al inicio de la investigación, además se comprobó que:

- Se establecieron los fundamentos teóricos asociados al tema de la investigación.
- Se desarrolló la aplicación Control de Inventario cumpliendo con los requisitos del cliente.
- Se validó la propuesta de solución por medio de la estrategia de pruebas establecida para su uso y liberación por parte del desarrollador.

Recomendaciones

Con el fin de expandir las posibilidades y funciones de la propuesta de solución se recomienda:

- Implementar funcionalidades asociadas a la conexión de la aplicación con servicios en línea.
- Implementar un sistema de notificaciones y recordatorios para realizar el inventario.
- Expandir el uso de la aplicación a otras áreas de la Universidad de las Ciencias Informáticas.

Anexos

Acta de aceptación

Universidad de las Ciencias Informáticas

Acta de aceptación

En cumplimiento de la ejecución de la asignatura **Proyecto de Investigación y Desarrollo** en el **Centro de Informatización de la Gestión Documental (CIGED)** y en función de la ejecución del proyecto **Aplicación móvil para el control de inventario en los locales del centro de desarrollo CIGED**, se hace entrega de los siguientes productos:

- Especificación de requisitos de software.
- Aplicación móvil (con manual de usuario incluido).

La Parte Cliente, luego de haber revisado los productos de trabajo determina que el producto cumple con los requisitos de software planteados fueron desarrollados según lo descrito.

Entrega	Recibe
Nombre y apellidos:	**Nombre y apellidos:**
Johan Alejandro Falcón Martínez	Ing. Roberto Soriano Sifontes
Cargo:	**Cargo:**
Desarrollador de aplicaciones	Director de Centro

CIGED
UCi Universidad de las Ciencias Informáticas
CENTRO DE INFORMATIZACIÓN DE LA GESTIÓN DOCUMENTAL

Fecha: 13/11/2023

Historias de Usuario

Historia de usuario	
Número: HU-1	**Nombre de la Historia de usuario:** Cargar Fichero.
Modificación de la historia de usuario:	
Usuario: Johan Alejandro Falcón Martínez	**Iteración asignada:** 1
Prioridad en el negocio: Alta	**Puntos estimados:** 0.5
Descripción: El sistema permite importar un archivo en formato (csv) para utilizar como base de datos el cual contiene los siguientes campos: Área de responsabilidad material, Rótulo y Subclasificación.	
Observaciones:	
Prototipo elemental de interfaz gráfica de usuario: 	

Tabla 27: HU-1: Cargar fichero. Elaboración propia.

Historia de usuario	
Número: HU-6	**Nombre de la Historia de usuario:** Mostrar formulario para recopilar datos de movimiento.
Modificación de la historia de usuario:	
Usuario: Johan Alejandro Falcón Martínez	**Iteración asignada:** 3
Prioridad en el negocio: Alta	**Puntos estimados:** 1
Descripción: El sistema muestra un formulario para recopilar los siguientes datos asociados a los movimientos de medios básicos: Entidad, Centro de costo, Área de responsabilidad, Descripción, Inventario No, Nombre, Cargo y Rótulo del medio básico. También muestra la opción de seleccionar el tipo de movimiento a realizar que se debe escoger entre: Movimiento simple interno, Movimiento simple externo, Movimiento masivo interno y Movimiento masivo externo.	
Observaciones:	

Prototipos elementales de interfaces gráficas de usuario:

Tabla 28: HU-6: Mostrar formulario para recopilar datos del movimiento. Elaboración propia.

Historia de usuario	
Número: HU-8	**Nombre de la Historia de usuario:** Mostrar tutorial inicial.
Modificación de la historia de usuario:	
Usuario: Johan Alejandro Falcón Martínez	**Iteración asignada:** 4
Prioridad en el negocio: Media	**Puntos estimados:** 1
Descripción: El sistema muestra un tutorial inicial asociado a las principales funciones de la aplicación. Este tutorial se muestra únicamente al abrir por primera vez la aplicación.	
Observaciones:	
Prototipo elemental de interfaz gráfica de usuario:	

Tabla 29: HU-8: Mostrar tutorial inicial. Elaboración propia.

Casos de prueba de aceptación

Caso de prueba de aceptación	
Código: HU2_CP2	**Historia de usuario:** Mostrar formulario para recopilar datos del código QR a generar.
Nombre: Mostrar formulario para recopilar datos del código QR a generar	
Descripción: Se prueba la funcionalidad de mostrar la interfaz de formulario para recopilar los datos asociados a la generación de un nuevo código QR.	
Condiciones de ejecución: El usuario debe ingresar datos válidos para generar los nuevos códigos QR asociados a los medios básicos.	
Entrada/Pasos de ejecución: • Se ingresan los datos asociados a los nuevos medios básicos.	
Resultado esperado: • Se muestra una interfaz de formulario para recopilar los datos asociados a la generación de un nuevo código QR.	
Evaluación: Satisfactoria.	

Tabla 30: HU2_CP2 - Mostrar formulario para recopilar datos del código QR a generar. Elaboración propia.

Caso de prueba de aceptación	
Código: HU3_CP3	**Historia de usuario:** Generar y exportar código QR.
Nombre: Generar y exportar código QR	
Descripción: Se prueba la funcionalidad asociada a la generación y exportación en formato (jpg) del nuevo código QR	
Condiciones de ejecución: El usuario debe haber ingresado datos válidos para generar los nuevos códigos QR asociados a los medios básicos.	
Entrada/Pasos de ejecución: • Se genera un nuevo código QR asociado a un medio básico. • Se exporta un nuevo código QR asociado a un medio básico.	
Resultado esperado: • Se genera y se guarda en el dispositivo una imagen en formato jpg asociada al código QR generado.	
Evaluación: Satisfactoria.	

Tabla 31: HU3_CP3 - Generar y exportar código QR. Elaboración propia.

Caso de prueba de aceptación	
Código: HU6_CP7	**Historia de usuario:** Mostrar formulario para recopilar datos de movimiento.
Nombre: Mostrar formulario para recopilar datos de movimiento.	
Descripción: Se prueba la funcionalidad de mostrar la interfaz de formulario para recopilar los datos asociados al tipo de movimiento seleccionado.	
Condiciones de ejecución: El usuario ingresa datos válidos.	
Entrada/Pasos de ejecución: • Se ingresan los datos. • Se selecciona el tipo de movimiento.	
Resultado esperado: • Se muestra una interfaz de formulario para recopilar los datos asociados al tipo de movimiento seleccionado.	
Evaluación: Satisfactoria.	

Tabla 32: HU6_CP7 - Mostrar formulario para recopilar datos de movimiento. Elaboración propia.

Caso de prueba de aceptación	
Código: HU8_CP10	**Historia de usuario:** Mostrar tutorial inicial.
Nombre: Mostrar tutorial inicial.	
Descripción: Se prueba la funcionalidad asociada a mostrar un tutorial inicial de la aplicación.	
Condiciones de ejecución: El usuario ingresa por primera vez a la aplicación.	
Entrada/Pasos de ejecución: • Iniciar aplicación	
Resultado esperado: • Se muestra un tutorial inicial	
Evaluación: Satisfactoria.	

Tabla 33: HU8_CP10 - Mostrar tutorial inicial. Elaboración propia.

Caso de prueba de aceptación

Código: HU4_CP5	**Historia de usuario:** Escaneo de código QR.
Nombre: Escaneo de código QR inválido.	
Descripción: Se prueba la funcionalidad asociada al escaneo de códigos QR y su estado escaneando un código inválido.	
Condiciones de ejecución: El usuario debe escanear un código QR inválido.	
Entrada/Pasos de ejecución: • Se escanea un código QR asociado a un medio básico.	
Resultado esperado: • Se escanea correctamente el código QR y se muestra un mensaje de error de color rojo.	
Evaluación: Satisfactoria.	

Tabla 34: HU4_CP5 - Escaneo de código QR. Elaboración propia.

Caso de prueba de aceptación	
Código: HU6_CP8	**Historia de usuario:** Mostrar formulario para recopilar datos de movimiento.
Nombre: Mostrar formulario para recopilar datos de movimiento.	
Descripción: Se prueba la funcionalidad de mostrar la interfaz de formulario para recopilar los datos asociados al tipo de movimiento seleccionado.	
Condiciones de ejecución: El usuario ingresa datos inválidos.	
Entrada/Pasos de ejecución: • Se ingresan los datos. • Se selecciona el tipo de movimiento.	
Resultado esperado: • Se muestra una interfaz de formulario para recopilar los datos asociados al tipo de movimiento seleccionado. El sistema muestra un mensaje de error en color rojo.	
Evaluación: Satisfactoria.	

Tabla 35: HU6_CP8 - Mostrar formulario para recopilar datos de movimiento. Elaboración propia.

Tareas de ingeniería

Tarea de ingeniería	
No. de tarea: 1	**No. de UH:** 1
Nombre de la tarea: Desarrollar una interfaz asociada a la función de cargar fichero de datos.	
Tipo de tarea: Desarrollo	**Responsable:** Johan Alejandro Falcón Martínez
Fecha inicio: 28/08/2023	**Fecha fin:** 29/09/2023
Descripción: Se desarrolla una interfaz de usuario asociada a la funcionalidad de cargar fichero.	

Tabla 36: Tarea de Ingeniería No. HU_1. Elaboración propia.

Tarea de ingeniería	
No. de tarea: 2	**No. de UH:** 1
Nombre de la tarea: Desarrollar una función que permita cargar fichero de datos.	
Tipo de tarea: Desarrollo	**Responsable:** Johan Alejandro Falcón Martínez
Fecha inicio: 29/08/2023	**Fecha fin:** 30/08/2023
Descripción: Se desarrollan las funcionalidades necesarias para la importación del fichero de datos.	

Tabla 37: Tarea de Ingeniería No.2 HU_1. Elaboración propia.

Tarea de ingeniería	
No. de tarea: 3	**No. de UH:** 1
Nombre de la tarea: Desarrollar una base de datos para guardar los datos del fichero.	
Tipo de tarea: Desarrollo	**Responsable:** Johan Alejandro Falcón Martínez
Fecha inicio: 30/08/2023	**Fecha fin:** 31/08/2023

Descripción: Se desarrollan las funcionalidades necesarias para crear una base de datos para el sistema.

Tabla 38: Tarea de Ingeniería No.3 HU_. Elaboración propia.

Tarea de ingeniería	
No. de tarea: 1	**No. de UH:** 2
Nombre de la tarea: Desarrollar una interfaz visual asociada a la función de recopilación de datos.	
Tipo de tarea: Desarrollo	**Responsable:** Johan Alejandro Falcón Martínez
Fecha inicio: 30/08/2023	**Fecha fin:** 31/08/2023
Descripción: Se desarrolla una interfaz de usuario asociada a la función de recopilación de datos.	

Tabla 39: Tarea de Ingeniería No.1 HU_2. Elaboración propia.

Tarea de ingeniería	
No. de tarea: 2	**No. de UH:** 2
Nombre de la tarea: Desarrollar una función que permita recopilar los datos.	
Tipo de tarea: Desarrollo	**Responsable:** Johan Alejandro Falcón Martínez
Fecha inicio: 31/08/2023	**Fecha fin:** 1/09/2023
Descripción: Se desarrollan las funcionalidades necesarias para la recolección de datos.	

Tabla 40: Tarea de Ingeniería No.2 HU_2. Elaboración propia.

Tarea de ingeniería	
No. de tarea: 1	**No. de UH:** 3
Nombre de la tarea: Desarrollar una función que permita generar el código QR.	
Tipo de tarea: Desarrollo	**Responsable:** Johan Alejandro Falcón Martínez
Fecha inicio: 4/09/2023	**Fecha fin:** 6/09/2023

Descripción: Se desarrollan las funcionalidades asociadas a generar los nuevos códigos QR.

Tabla 41: Tarea de Ingeniería No.1 HU_3. Elaboración propia.

	Tarea de ingeniería
No. de tarea: 2	**No. de UH:** 3
Nombre de la tarea: Desarrollar una función que permita exportar el código QR.	
Tipo de tarea: Desarrollo	**Responsable:** Johan Alejandro Falcón Martínez
Fecha inicio: 6/09/2023	**Fecha fin:** 8/09/2023
Descripción: Se desarrollan las funcionalidades asociadas a guardar y exportar los nuevos códigos QR.	

Tabla 42: Tarea de Ingeniería No.2 HU_3. Elaboración propia.

	Tarea de ingeniería
No. de tarea: 1	**No. de UH:** 4
Nombre de la tarea: Desarrollar una interfaz visual asociada a la función de escaneo de códigos QR.	
Tipo de tarea: Desarrollo	**Responsable:** Johan Alejandro Falcón Martínez
Fecha inicio: 11/09/2023	**Fecha fin:** 12/09/2023
Descripción: Se desarrolla una interfaz de usuario asociada a la funcionalidad de escaneo de códigos QR.	

Tabla 43: Tarea de Ingeniería No.1 HU_4. Elaboración propia.

	Tarea de ingeniería
No. de tarea: 2	**No. de UH:** 4
Nombre de la tarea: Desarrollar una función que permita escanear códigos QR.	
Tipo de tarea: Desarrollo	**Responsable:** Johan Alejandro Falcón Martínez

Fecha inicio: 12/09/2023	**Fecha fin:** 13/09/2023
Descripción: Se desarrollan las funcionalidades asociadas al escaneo de códigos QR.	

Tabla 44: Tarea de Ingeniería No.2 HU_4. Elaboración propia.

Tarea de ingeniería	
No. de tarea: 1	**No. de tarea:** 5
Nombre de la tarea: Desarrollar una interfaz visual asociada a la función de mostrar información del inventario.	
Tipo de tarea: Desarrollo	**Responsable:** Johan Alejandro Falcón Martínez
Fecha inicio: 18/09/2023	**Fecha inicio:** 20/09/2023
Descripción: Se desarrollan las funciones necesarias para la implementación de una interfaz de usuario que muestre la información del inventario.	

Tabla 45: Tarea de Ingeniería No.1 HU_5. Elaboración propia.

Tarea de ingeniería	
No. de tarea: 3	**No. de tarea:** 5
Nombre de la tarea: Desarrollar una función que permita hacerle consultas a la base de datos para realizar el inventario.	
Tipo de tarea: Desarrollo	**Tipo de tarea:** Desarrollo
Fecha inicio: 25/09/2023	**Fecha inicio:** 27/09/2023
Descripción: Se desarrollan las funcionalidades necesarias para realizar consultas a la base de datos.	

Tabla 46: Tarea de Ingeniería No.3 HU_5. Elaboración propia.

Tarea de ingeniería	
No. de tarea: 4	**No. de tarea:** 5
Nombre de la tarea: Desarrollar función asociada a la generación y exportación de los datos recopilados del inventario.	

Tipo de tarea: Desarrollo	**Responsable:** Johan Alejandro Falcón Martínez
Fecha inicio: 27/09/2023	**Fecha inicio:** 29/09/2023
Descripción: Se desarrollan las funcionalidades asociadas a la generación y la funcionalidad de exportar los datos del inventario.	

Tabla 47: Tarea de Ingeniería No.4 HU_5. Elaboración propia.

Tarea de ingeniería	
No. de tarea: 1	**No. de tarea:** 6
Nombre de la tarea: Desarrollar una interfaz visual asociada a la función de recopilación de datos.	
Tipo de tarea: Desarrollo	**Responsable:** Johan Alejandro Falcón Martínez
Fecha inicio: 2/10/2023	**Fecha inicio:** 4/10/2023
Descripción: Se desarrolla una interfaz de usuario asociada a la función de recolección de datos.	

Tabla 48: Tarea de Ingeniería No.1 HU_6. Elaboración propia.

Tarea de ingeniería	
No. de tarea: 2	**No. de tarea:** 6
Nombre de la tarea: Desarrollar una función que permita recopilar los datos.	
Tipo de tarea: Desarrollo	**Responsable:** Johan Alejandro Falcón Martínez
Fecha inicio: 4/10/2023	**Fecha inicio:** 6/10/2023
Descripción: Se desarrollan las funciones necesarias para la recopilación de datos.	

Tabla 49: Tarea de Ingeniería No.2 HU_6. Elaboración propia.

Tarea de ingeniería	
No. de tarea: 1	**No. de tarea:** 7

Nombre de la tarea: Desarrollar una función que permita generar los reportes de movimientos.	
Tipo de tarea: Desarrollo	**Responsable:** Johan Alejandro Falcón Martínez
Fecha inicio: 9/10/2023	**Fecha inicio:** 13/10/2023
Descripción: Se desarrollan las funciones necesarias para permitir generar los reportes de movimientos de medios.	

Tabla 50: Tarea de Ingeniería No.1 HU_7. Elaboración propia.

	Tarea de ingeniería
No. de tarea: 2	**No. de tarea:** 7
Nombre de la tarea: Desarrollar una función que permita exportar los reportes de movimientos.	
Tipo de tarea: Desarrollo	**Responsable:** Johan Alejandro Falcón Martínez
Fecha inicio: 16/10/2023	**Fecha inicio:** 20/10/2023
Descripción: Se desarrollan las funciones necesarias para permitir exportar los reportes de movimientos de medios.	

Tabla 51: Tarea de Ingeniería No.2 HU_7. Elaboración propia.

	Tarea de ingeniería
No. de tarea: 1	**No. de tarea:** 8
Nombre de la tarea: Desarrollar una interfaz visual con animaciones que muestre un tutorial inicial.	
Tipo de tarea: Desarrollo	**Responsable:** Johan Alejandro Falcón Martínez
Fecha inicio: 23/10/2023	**Fecha inicio: 25**/10/2023
Descripción: Se desarrolla una interfaz asociada a la muestra de un tutorial con las principales funciones de la aplicación.	

Tabla 52: Tarea de Ingeniería No.1 HU_8. Elaboración propia.

Tarea de ingeniería

No. de tarea: 2	**No. de tarea:** 8
Nombre de la tarea: Desarrollar una función que permita mostrar el tutorial una única vez al abrir por primera vez la aplicación.	
Tipo de tarea: Desarrollo	**Responsable:** Johan Alejandro Falcón Martínez
Fecha inicio: 25/10/2023	**Fecha inicio:** 27/10/2023
Descripción: Se desarrollan las funciones asociadas a la muestra de un tutorial con las principales funciones de la aplicación.	

Tabla 53: Tarea de Ingeniería No.2 HU_8. Elaboración propia.

	Tarea de ingeniería
No. de tarea: 1	**No. de tarea:** 9
Nombre de la tarea: Desarrollar una interfaz visual que muestre estadísticas relacionadas a los medios básicos.	
Tipo de tarea: Desarrollo	**Responsable:** Johan Alejandro Falcón Martínez
Fecha inicio: 30/10/2023	**Fecha inicio:** 1/11/2023
Descripción: Se desarrolla una interfaz de usuario para la muestra de estadísticas asociadas a la cantidad de medios básicos.	

Tabla 54: Tarea de Ingeniería No.1 HU_9. Elaboración propia.

	Tarea de ingeniería
No. de tarea: 2	**No. de tarea:** 9
Nombre de la tarea: Desarrollar una función asociada a la consulta realizada a la base de datos para mostrar estadísticas.	
Tipo de tarea: Desarrollo	**Responsable:** Johan Alejandro Falcón Martínez
Fecha inicio: 1/11/2023	**Fecha inicio:** 3/11/2023
Descripción: Se desarrollan las funcionalidades necesarias para acceder y trabajar sobre la base de datos de la aplicación.	

Tabla 55: Tarea de Ingeniería No.2 HU_9. Elaboración propia.

Bibliografía

Android Developers. (2023). Obtenido de https://developer.android.com/jetpack/androidx/releases/sqlite?hl=es-419

Apiumhub. (11 de 10 de 2023). *Apiumhub.com*. Obtenido de Apiumhub.com: https://apiumhub.com/es/tech-blog-barcelona/arquitectura-android-repensando-mvp-en-android/

Asana. (23 de 9 de 2023). Obtenido de https://asana.com/es/resources/what-is-kanban

Asana. (23 de 9 de 2023). Obtenido de https://asana.com/es/resources/extreme-programming-xp

Biblioteca digital de la Universidad Católica de Argentina. (25 de 9 de 2023). Obtenido de https://repositorio.uca.edu.ar/bitstream/123456789/522/1/metodologias-desarrollo-software.pdf

Bino Solutions. (2023). *Play Sotre*. Obtenido de https://play.google.com/store/apps/details?id=ro.bino.inventory&hl=es&gl=US

CEPAL. (23 de 9 de 2023). *Biblioguías - Biblioteca de la CEPAL*. Obtenido de https://biblioguias.cepal.org/QR

Codingornot. (11 de 10 de 2023). *Codingornot.com*. Obtenido de Codingornot.com: https://codingornot.com/mvc-modelo-vista-controlador-que-es-y-para-que-sirve

Definición.De. (23 de 9 de 2023). *Definición.De*. Obtenido de Definición.De: https://definicion.de/

Dialnet. (23 de 9 de 2023). *Dialnet*. Obtenido de Dialnet: https://dialnet.unirioja.es/servlet/articulo?codigo=1983605

Dsipatchtrack. (20 de 9 de 2023). *Dsipatchtrack*. Obtenido de Dsipatchtrack: https://www.beetrack.com/es/blog/control-de-inventario

FreeCodeCamp. (1 de 5 de 2023). Obtenido de https://www.freecodecamp.org/espanol/news/que-es-flutter-y-porque-deberias-aprenderlo-en-2020/

Git. (10 de 10 de 2023). *git-scm.com*. Obtenido de git-scm.com: https://git-scm.com/

Google. (12 de 6 de 2023). *Android Developers*. Obtenido de Android Developers: https://developer.android.com/

GS1 Mexico. (23 de 9 de 2023). Obtenido de https://blog.gs1mexico.org/apps-para-el-control-del-inventario-en-tu-pyme

HubSpot. (20 de 9 de 2023). *HubSpot*. Obtenido de https://blog.hubspot.es/sales/que-es-inventario

Linkedin. (2023). Obtenido de https://es.linkedin.com/posts/eduardorubiom_linkedin-linkedin-liderazgo-activity-7043921698848473089-MvLb

Medium.com. (11 de 10 de 2023). *Medium.com*. Obtenido de Medium.com: https://medium.com/@goodrebels/microservicios-ventajas-y-contras-de-la-arquitectura-descentralizada-a3b7fc814422

Microsoft. (2023 de 9 de 2023). Obtenido de https://visualstudio.microsoft.com/es/#vscode-section

NonZeroApps. (2023). *Play Store*. Obtenido de https://play.google.com/store/apps/details?id=com.nonzeroapps.android.smartinventory&hl=es&gl=US

Openwebinars. (2023). Obtenido de https://openwebinars.net/blog/conoce-las-3-metodologias-agiles-mas-usadas/

Pressman, R. S. (2010). *Ingeniería del Software: Un enfoque práctico (7ª edición).* México D.F.: McGraw-Hill.

Proyectos Agiles. (23 de 9 de 2023). *ProyectosAgiles.org*. Obtenido de ProyectosAgiles.org: https://proyectosagiles.org/que-es-scrum/

Pub Dev. (23 de 9 de 2023). Obtenido de https://pub.dev/

Real Academia Española. (23 de 9 de 2023). *Real Academia Española*. Obtenido de Real Academia Española: https://dle.rae.es/

Significados.Com. (23 de 9 de 2023). *Significados.Com*. Obtenido de https://www.significados.com/control/

Techtarget. (11 de 10 de 2023). *Techtarget.com*. Obtenido de Techtarget.com: https://www.techtarget.com/whatis/definition/Model-View-ViewModel

Tecnomática. (1 de 10 de 2023). *tecnomatica.cupet.cu*. Obtenido de tecnomatica.cupet.cu: https://tecnomatica.cupet.cu/es/print/productos/sistema-de-control-de-inventario-de-medios-informaticos-y-comunicaciones-tgestor

Tecnosoluciones. (2023). Obtenido de https://tecnosoluciones.com/10-razones-para-usar-la-metodologia-kanban-en-tu-organizacion/

UNIVERSIDAD NACIONAL AUTONOMA DE NICARAGUA, M. (2016). *METODOLOGIA ÁGIL DE DESARROLLO DE SOFTWARE PROGRAMACION.* MANAGUA, NICARAGUA.

UPC. (23 de 9 de 2023). Obtenido de https://inlab.fib.upc.edu/es/blog/que-es-el-lenguaje-de-programacion-dart

Visuresolutions. (3 de 10 de 2023). *visuresolutions.com*. Obtenido de visuresolutions.com: https://visuresolutions.com/es/blog/non-functional-requirements/

Xurxodev. (11 de 10 de 2023). *Xurxodev.com*. Obtenido de Xurxodev.com: https://xurxodev.com/frontend-clean_architecture/

YKART. (2023). *Google Play*. Obtenido de https://play.google.com/store/apps/details?id=com.ykart.tool.qrcodegen&hl=es&gl=US&pli=1

Ambler, S. (2002). Agile modeling: Effective practices for extreme programming and the unified process. John Wiley & Sons.

Beck, K. (2000). *Extreme programming explained: Embrace change*. addison-wesley professional.

Beck, K., Beedle, M., Van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., & Jeffries, R. (2001). *Manifesto for agile software development*.

de Guevara, M. Á. L. (2020). *Gestión de inventarios. UF0476.* Tutor formación.

Fowler, M. (2018). *UML distilled: A brief guide to the standard object modeling language*. Addison-Wesley Professional.

Gamma Erich, R. H., Ralph, J., & John, Vl. (2003). *Patrones de Diseño. Elementos de software orientado a objetos reutilizables*. España.

Letelier, P. (2006). *Metodologías ágiles para el desarrollo de software: eXtreme Programming (XP)*.

Maida, E. G., & Pacienzia, J. (2015). *Metodologías de desarrollo de software*.