



UNIVERSIDAD CATÓLICA
DE SANTIAGO DE GUAYAQUIL

SISTEMA DE POSGRADO

MAESTRÍA EN TELECOMUNICACIONES

TEMA:

Diseño e Implementación de un sistema de inventario usando la tecnología NFC para la Unidad Educativa Particular Virgen del Cisne en la ciudad de Machala mediante una aplicación con sistema Operativo IOS

AUTOR:

Rafael Guillermo Castro Merino

**Trabajo de titulación previo a la obtención del grado de
Magister en Telecomunicaciones**

TUTOR:

MSc. Manuel Romero Paz

Guayaquil, 1 de diciembre de 2021



UNIVERSIDAD CATÓLICA
DE SANTIAGO DE GUAYAQUIL

**SISTEMA DE POSGRADO
MAESTRÍA EN TELECOMUNICACIONES**

CERTIFICACIÓN

Certificamos que el presente trabajo fue realizado en su totalidad por Rafael Guillermo Castro Merino como requerimiento parcial para la obtención del Título de Magíster en Telecomunicaciones.

TUTOR

MSc. Manuel Romero Paz

DIRECTOR DEL PROGRAMA

MSc. Manuel Romero Paz

Guayaquil, 1 de diciembre de 2021



UNIVERSIDAD CATÓLICA
DE SANTIAGO DE GUAYAQUIL

**SISTEMA DE POSGRADO
MAESTRÍA EN TELECOMUNICACIONES**

DECLARACIÓN DE RESPONSABILIDAD

YO, RAFAEL GUILLERMO CASTRO MERINO

DECLARO QUE:

El trabajo de Titulación “Diseño e implementación de un sistema de inventario usando la tecnología NFC para la Unidad Educativa Particular Virgen del Cisne en la ciudad de Machala mediante una aplicación con sistema operativo IOS” previo a la obtención del Título de **Magíster en Telecomunicaciones**, ha sido desarrollado respetando derechos intelectuales de terceros conforme las citas que constan en el documento, cuyas fuentes se incorporan en las referencias o bibliografías. Consecuentemente este trabajo es de mi total autoría.

En virtud de esta declaración, me responsabilizo del contenido, veracidad y alcance del Trabajo de Titulación referido.

Guayaquil, 1 de diciembre de 2021

EL AUTOR

Ing. Rafael Guillermo Castro Merino



UNIVERSIDAD CATÓLICA
DE SANTIAGO DE GUAYAQUIL

**SISTEMA DE POSGRADO
MAESTRÍA EN TELECOMUNICACIONES**

AUTORIZACIÓN

YO, RAFAEL GUILLERMO CASTRO MERINO

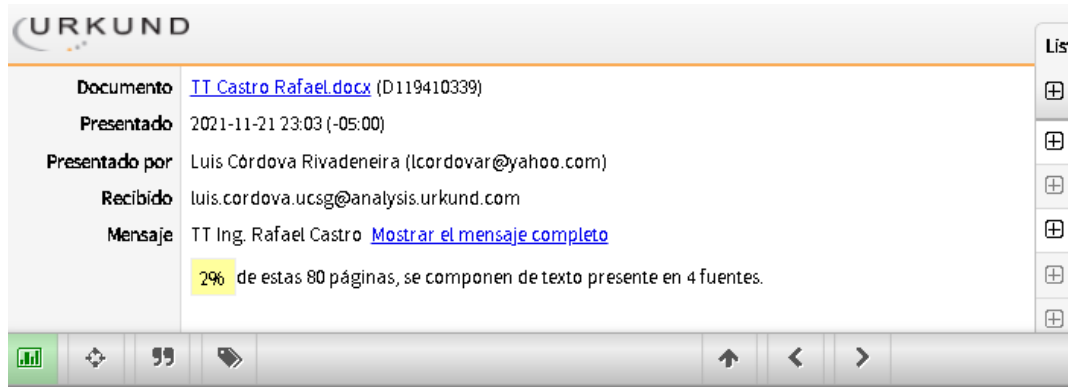
Autorizo a la Universidad Católica de Santiago de Guayaquil a la **publicación**, en la biblioteca de la institución del Trabajo de Titulación, “Diseño e implementación de un sistema de inventario usando la tecnología NFC para la Unidad Educativa Particular Virgen del Cisne en la ciudad de Machala mediante una aplicación con sistema operativo IOS”, cuyo contenido, ideas y criterios son de mi exclusiva responsabilidad y total autoría.

Guayaquil, 1 de diciembre de 2021

EL AUTOR

Ing. Rafael Guillermo Castro Merino

REPORT E JRKUND



The screenshot shows the URKUND interface with the following details:

- Documento:** [TT Castro Rafael.docx](#) (D119410339)
- Presentado:** 2021-11-21 23:03 (-05:00)
- Presentado por:** Luis Córdova Rivadeneira (lcordovar@yahoo.com)
- Recibido:** luis.cordova.ucsg@analysis.urkund.com
- Mensaje:** TT Ing. Rafael Castro [Mostrar el mensaje completo](#)
2% de estas 80 páginas, se componen de texto presente en 4 fuentes.

The interface includes a toolbar at the bottom with icons for search, zoom, and navigation, and a vertical sidebar on the right with expand/collapse buttons.

SISTEMA DE POSGRADO

MAESTRÍA EN TELECOMUNICACIONES

TEMA: Diseño e Implementación de un sistema de inventario usando la tecnología NFC para la Unidad Educativa Particular Virgen del Cisne en la ciudad de Machala mediante una aplicación con sistema Operativo IOS

AUTOR: Rafael Guillermo Castro Merino

Trabajo de titulación previo a la obtención del grado de Magister en Telecomunicaciones

TUTOR: MSc. Manuel Romero Paz

Guayaquil, a los 25 días del mes junio del año 2020

SISTEMA DE POSGRADO MAESTRÍA EN TELECOMUNICACIONES

CERTIFICACIÓN Certificamos que el presente trabajo fue realizado en su totalidad por Rafael Guillermo Castro Merino como requerimiento parcial para la obtención del Título de Magister en Telecomunicaciones.

DEDICATORIA

Dedico este trabajo a mis padres Guillermo y Teresa que siempre se enfocaron en hacerme entender la importancia del conocimiento, que nunca dudaron de mis capacidades y siempre me apoyaron en todo lo que me propusiera.

Castro Merino, Rafael Guillermo

AGRADECIMIENTOS

Agradezco eternamente a la Universidad Católica Santiago de Guayaquil por permitirme ingresar en su prestigiosa institución y proporcionarme la preparación académica necesaria para lograr mis objetivos profesionales y personales.

Le agradezco también a mi tutor el Ing. Manuel Romero que me supo comprender para ayudarme y guiarme durante el proceso del desarrollo de mi tesis, en una situación político-económica social incierta por lo acontecido en la república del Ecuador y el mundo entero al momento de escribir estas palabras.

Mi mayor agradecimiento es sobre todo a mis padres que hicieron todo lo posible por formarme y ayudarme durante mi vida y me inculcaron la idea de que la perseverancia y el esfuerzo supera al talento y que nunca dejamos de aprender.

Castro Merino, Rafael Guillermo



**UNIVERSIDAD CATÓLICA
DE SANTIAGO DE GUAYAQUIL**

**SISTEMA DE POSGRADO
MAESTRÍA EN TELECOMUNICACIONES**

TRIBUNAL DE SUSTENTACIÓN

f.

MSc. Manuel Romero Paz

TUTOR

f.

MSc. Manuel Romero Paz

DIRECTOR DEL PROGRAMA

f.

MSc. Luis Córdova Rivadeneira

REVISOR

f.

MSc. Edgar Quezada Calle

REVISOR

ÍNDICE GENERAL

ÍNDICE DE FIGURAS	XIV
ÍNDICE DE TABLAS	XVIII
CAPITULO 1 . Descripción del proyecto	2
1.1. Introducción	2
1.2. Antecedentes	3
1.3. Problema.....	3
1.4. Justificación	4
1.5. Objetivos	5
1.6. Hipótesis	5
1.7. Campo de acción	6
1.8. Metodología de investigación.....	6
Capítulo 2. Marco Teórico.....	7
2.1. ¿Qué es la tecnología NFC?.....	7
2.1.1. Foro NFC	7
2.1.2. Estandarizacion de nfc.....	8
2.1.2.1. ISO/IEC 14443.....	8
2.1.2.1.1. ISO/IEC 14443-1:2008 Características físicas	9
2.1.2.1.2. ISO/IEC 14443-2:2010 Radio frecuencia y señal de interfaz	9
2.1.2.1.2. ISO/IEC 14443-3:2011 Inicialización y anticolidión	9
2.1.2.1.3. ISO/IEC 14443-4:2008 Protocolo de trasmisión	9
2.1.2.2. ISO/IEC 18000-3.....	10
2.1.2.3. JIS 6319-4 Felica	10
2.1.3. Modos de operación	11
2.1.3.1. Modo de comunicación activa.....	11
2.1.3.2. Modo de comunicación Pasiva	12
2.1.3.3. Modo de operación emulacion de tarjeta	12
2.1.3.4. Modo de operación lectura/escritura	12
2.1.3.5. Modo de operación Punto a punto (p2p).....	13
2.1.4. Etiquetas NFC.....	14
2.1.4.1. Transferencias de poder	14

2.1.4.2. Definición de los 5 tipos de etiquetas.....	15
2.1.4.2.1. Etiquetas tipo 1	16
2.1.4.2.2. Etiquetas tipo 2	17
2.1.4.2.3. Etiquetas tipo 3	17
2.1.4.2.4. Etiquetas tipo 4	17
2.1.4.2.5. Etiquetas tipo 5	18
2.1.4.3. Proceso de lectura de una etiqueta	18
2.1.4.4. Formato de intercambio de datos (NDEF)	20
2.1.4.4.1. Sistema de registro NDEF	21
2.2. Iphone os (IOS).....	23
2.2.1. Arquitectura de IOS	23
2.2.1.1. Cocoa y Carbón	25
2.2.1.2. BSD/Match	25
2.2.1.2. El kernel Darwin	26
2.3. Xcode y Swift	26
2.3.1. XCode	27
2.3.1.1. Nuevo proyecto (new project)	27
2.3.1.2. Pantalla del proyecto	28
2.3.1.2.1. Panel de navegación.....	29
2.3.1.2.1.1. Navegador de proyecto.....	29
2.3.1.2.1.2. Navegador de control de código fuente	30
2.3.1.2.1.3. Navegador del símbolo	30
2.3.1.2.1.4. Navegador de búsqueda.....	30
2.3.1.2.1.5. Navegador de problemas.....	31
2.3.1.2.1.6. Navegador de pruebas.....	31
2.3.1.2.1.7. Navegador de depuración.....	31
2.3.1.2.1.7. Navegador de punto de quiebre.....	32
2.3.1.2.1.8. Navegador de reportes	32
2.3.2. Swift	32
2.3.2.1. Características principales	33
2.3.2.1.1. Sintaxis básica	33
2.3.2.1.2. Orientado a objetos.....	34
2.3.2.1.3. Estructura del archivo Swift.....	34

2.3.2.1.3.1. Importación de módulos.....	35
2.3.2.1.3.2. Declaración de variables.....	35
2.3.2.1.3.3. Declaración de funciones.....	35
2.3.2.1.3.4. Declaración de tipos de objetos.....	35
2.3.2.1.3.5. Alcance y tiempo de vida.....	36
2.3.2.1.3.6. Objetos miembros.....	36
2.3.2.1.3.6.1. Namespaces.....	36
2.3.2.1.3.6.2. Módulos.....	37
2.3.2.1.3.6.3. Instancias.....	37
2.3.2.1.3.6.4. Privacidad.....	37
2.3.3. Cocoa.....	38
2.3.3.1. Administración de memoria de Cocoa.....	38
2.3.3.2. Reglas de la administración de memoria.....	39
2.3.3.3. Funcionamiento de ARC.....	39
2.3.3.3.1. Referencias débiles.....	42
2.3.3.3.2. Referencias sin propietario.....	42
2.4 Bases de datos.....	42
2.4.1. Gestor de base datos.....	42
2.4.2. Tipos o modelos de base de datos.....	43
2.4.2.1. Base de datos relacionales.....	43
2.4.2.2. NO SQL.....	43
2.4.2.3. Orientadas a objetos.....	43
2.4.3. SQL (Structure Query Language).....	44
2.5. Aplicaciones web.....	44
2.6. Servidor web.....	44
2.7. PHP.....	45
2.8. API.....	45
2.8.1. Servicios Web.....	45
2.8.1. Métodos HTTP.....	46
Capítulo 3. Desarrollo del trabajo de titulación.....	47
3.1. Requerimientos del sistema.....	47
3.2. Arquitectura de la aplicación.....	48
3.3. Funcionamiento de la aplicación.....	48

3.3.1. Etiquetas NFC.....	49
3.3.2. Aplicación web	50
3.3.3. Aplicación movil	50
3.3.4. Base de Datos	50
3.3.5. Servidor web	51
3.4. Estructura de la base de datos	51
3.5. Diseño e implementación.....	52
3.5.1. Pantalla de login	53
3.5.2. Pantalla de inicio	54
3.5.3. Interfaz de inicio	55
3.5.4. Usuarios – Administración de grupos	56
3.5.5. Administración de usuarios	57
3.5.6. Agregar locación	59
3.5.7. Agregar categoría	59
3.5.8. Agregar Marca	60
3.5.9. Resumen de Bienes.....	60
3.5.10. Agregar bienes.....	61
3.5.11. Perfiles de usuario	62
3.6. Diseño de la aplicación móvil de IOS.....	64
3.6.1. Login de usuario.	65
3.6.2. Menú Principal	69
3.6.3. Escritor NFC	70
3.6.4. Lector NFC.....	74
3.6.4.1. Método de registro	76
Capítulo 4. Pruebas y Resultados.....	81
4.1. componentes de hardware y software implementados	81
4.2. Escenario de pruebas	81
4.3. Pruebas de funcionalidad	82
4.3.1. Pruebas de aplicación web	82
4.3.1.1. Añadir un nuevo grupo.....	82
4.3.1.2. Añadir un nuevo usuario	84
4.3.1.3. Añadir un nueva categoría	85
4.3.1.4. Añadir una nueva marca	85

4.3.1.5. Añadir una nueva locación.....	86
4.3.1.6. Añadir un nuevo bien	86
4.3.1.7. Editar cuenta de usuario	88
4.3.2. Pruebas de aplicación móvil	88
4.3.2.1. Login	88
4.3.2.2. Escribir etiqueta	90
4.3.2.3. Leer etiqueta	91
4.3.2.3. Registro de etiqueta.....	93
4.4. Análisis de los resultados.....	93
4.4.1. Login	93
4.4.2. Registro en la base de datos	94
4.4.3. Visualización de la información procesada	94
4.4.4. Lectura y escritura	94
4.4.5. Errores de conexión	94
Conclusiones	95
Recomendaciones	96
Bibliografía	97
ANEXOS.....	100
ANEXO A.....	100
ANEXO B.....	111
ANEXO C.....	114

ÍNDICE DE FIGURAS

Figura 2. 1 Estándares NFC	15
Figura 2. 2 Cinco tipos de etiquetas NFC	16
Figura 2. 3 Secuencia de Iniciación de etiquetas NFC	18
Figura 2. 4 Estructura de Mensajes NDEF	20
Figura 2. 5 Registros de Mensajes NDEF	21
Figura 2. 6 Arquitectura de IOS	24
Figura 2. 7 Separación de los servicios de IOS	25
Figura 2. 8 Ventana del proyecto	29
Figura 2. 9 Pantalla navegador del Proyecto	30
Figura 2. 10 Navegador de Búsqueda	31
Figura 2. 11 Navegador de depuración.....	32
Figura 2. 12 Contador de referencias para las clases Job y Person.....	41
Figura 2. 13 Referencias fuertes entre las clases Job y Person	41
Figura 2. 14 Ciclo ARC	42
Figura 2. 15 Ejemplo de PHP.....	45
Figura 3. 1 Diagrama de despliegue de la aplicación	48
Figura 3. 2 Diagrama de funcionamiento de la aplicación	49
Figura 3. 3 Estructura de categorías.....	51
Figura 3. 4 Estructura de Productos.	52
Figura 3. 5 Estructura de Usuarios.	52
Figura 3. 6 Estructura de Grupos de Usuarios.....	52
Figura 3. 7 Pantalla de inicio de la aplicación web.....	53
Figura 3. 8 Código de inicio de sesión.	54
Figura 3. 9 Diagrama de flujo de inicio de sesión.	54
Figura 3. 10 Pantalla de inicio.....	55
Figura 3. 11 Interfaz de inicio.....	55
Figura 3. 12 Interfaz de inicio.....	55
Figura 3. 13 Interfaz de Administración de grupos.....	56
Figura 3. 14 Agregar nuevos grupos de usuarios	56
Figura 3. 15 Diagrama de creación de grupo.....	56
Figura 3. 16 Código fuente de creación de grupo.	57

Figura 3. 17 Pantalla de administración de Usuarios.....	57
Figura 3. 18 Pantalla para agregar un usuario nuevo.	58
Figura 3. 19 Diagrama agregar un usuario nuevo.....	58
Figura 3. 20 Código fuente de proceso para agregar usuarios.	58
Figura 3. 21 Pantalla de locaciones.	59
Figura 3. 22 Pantalla editar locaciones.	59
Figura 3. 23 Pantalla editar categorías.	59
Figura 3. 24 Pantalla actualizar categorías.	60
Figura 3. 25 Pantalla agregar marcas.....	60
Figura 3. 26 Pantalla editar marcas	60
Figura 3. 27 Pantalla resumen de inventario.	61
Figura 3. 28 Pantalla de agregar nuevo inventario.	61
Figura 3. 29 Diagrama de agregar nuevo inventario.....	62
Figura 3. 30 Código fuente del código agregar bienes.	62
Figura 3. 31 Pantalla de edición de perfil.....	63
Figura 3. 32 Formulario de edición de perfil.....	63
Figura 3. 33 Código fuente de edición de perfil.	63
Figura 3. 34 Diagrama de casos de uso de la aplicación móvil.	64
Figura 3. 35 Diagrama de casos de uso de la aplicación móvil.	65
Figura 3. 36 Pantalla de login de la aplicación móvil.	65
Figura 3. 37 Servicio Web Gestonador.	66
Figura 3. 38 Función para obtener usuario.	66
Figura 3. 39 Función login.	67
Figura 3. 40 Función login- creación de petición.....	67
Figura 3. 41 Estructura de credencial.	67
Figura 3. 42 Función login-Decodificando data.....	68
Figura 3. 43 Diagrama de flujo Login.	68
Figura 3. 44 Menú principal aplicación web.	69
Figura 3. 45 Código fuente vista de interfaz HomeView.	69
Figura 3. 46 Código fuente selector de vistas dentro de Home.	70
Figura 3. 47 Formulario Escritura NFC.	70
Figura 3. 48 Código fuente servicio web pickers.....	71
Figura 3. 49 Estructura de categorías.....	71

Figura 3. 50 Estructura de marcas.....	71
Figura 3. 51 Estructura de locaciones.....	72
Figura 3. 52 Función cargar categoría.	72
Figura 3. 53 Decodificando JSON de categoría.	72
Figura 3.54 Métodos del protocolo NFCNDEFSessionReaderDelegate.73	
Figura 3. 55 Agregando payload en didDetect.....	73
Figura 3. 56 Encapsulado del mensaje en un payload.	74
Figura 3. 57 Leer etiqueta.....	74
Figura 3. 58 Código fuente Leer etiqueta.....	75
Figura 3. 59 Clase decodificadora de payload.....	75
Figura 3. 60 Protocolo NFCTagReaderSessionDelegate.	76
Figura 3. 61 Opción de tipo de etiqueta ISO 14443.....	76
Figura 3. 62 Método para capturar UID con formato.....	76
Figura 3.63 Verificar Existencia de ID de tarjeta NFC en el servicio web...	77
Figura 3. 64 Ingreso del registro en servicio web.....	77
Figura 3. 65 Actualización del registro en servicio web.	78
Figura 3. 66 Estructura de producto.....	78
Figura 3. 67 Estructurando los parámetros del TextEditor.....	78
Figura 3. 68 Estructurando los parámetros del TextEditor.....	79
Figura 3. 69 Envío de Petición de registro a la base de datos.	79
Figura 3. 70 Diagrama de flujo de registro.....	80
Figura 4. 1 Etiquetas NFC.	81
Figura 4. 2 Nombre de grupo Duplicado.....	82
Figura 4. 3 Nombre de grupo en blanco.	83
Figura 4. 4 Grupo Ingresado Correctamente.	83
Figura 4. 5 Listado de Grupos.....	83
Figura 4. 6 Datos de usuario en blanco.	84
Figura 4. 7 Datos de usuario creados.....	84
Figura 4. 8 Lista de usuario creados.....	84
Figura 4. 9 Lista de Categorías.....	85
Figura 4. 10 Categoría agregada.....	85

Figura 4. 11 Listado de Marcas.....	85
Figura 4. 12 Marca agregada.....	86
Figura 4. 13 Listado de locaciones.	86
Figura 4. 14 Locaciones agregadas.....	86
Figura 4. 15 Agregar bien con datos incompletos.....	87
Figura 4. 16 Proceso de nuevo registro.	87
Figura 4. 17 Mensaje de registro de bien exitoso.	87
Figura 4. 18 Listado actualizado con un nuevo bien.	88
Figura 4. 19 Edición de cuenta.	88
Figura 4. 20 Cuenta editada.	88
Figura 4. 21 Validación de campos en blanco en pantalla de Login.	89
Figura 4.22 Validación de usuario no encontrado en pantalla de Login.	89
Figura 4. 23 Menú principal con el nombre del usuario.	90
Figura 4. 24 Ingreso de campos en escritor NFC.	90
Figura 4. 25 Botón escribir escritor NFC.....	91
Figura 4. 26 Mensaje de escaneo de escritor NFC.....	91
Figura 4. 27 Lectorn NFC.	92
Figura 4. 28 Escaneo etiqueta NFC.....	92
Figura 4. 29 Etiqueta NFC escaneada.....	93
Figura 4. 30 Nuevo registro agregado.	93

ÍNDICE DE TABLAS

Tabla 3. 1 Característica de las etiquetas NTAG25	49
Tabla 4. 1 Característica de las etiquetas NTAG25	81

RESUMEN

El proyecto presentado en este documento está relacionado con el uso e implementación de la tecnología Near Field Communication, abreviada NFC, mediante una aplicación móvil, y un servidor web para almacenar y procesar la información obtenida con el fin de mantener el control sobre los ítems que componen el Inventario de la empresa. El prototipo desarrollado en este proyecto utiliza un servidor web, una base de datos, etiquetas NFC y la aplicación hecha para trabajar con el sistema operativo IOS de la empresa Apple que es la segunda marca móvil más utilizada del mercado. El capítulo uno se enfoca en el tipo de proyecto a realizar. El capítulo dos presenta los conceptos sobre tecnología NFC, bases de datos, sistema operativo IOS, lenguajes de programación y todo el software incluido. Además, los objetivos y alcance del proyecto. El capítulo tres describe la arquitectura y el desarrollo general de las aplicaciones y cómo estas interactúan entre ellas. El capítulo cuatro involucra los datos de prueba y los parámetros para el uso correcto del software. Finalmente describe las conclusiones, recomendaciones y prácticas de seguridad que se obtuvieron a través de este proyecto.

Palabras clave: NFC, NDEF, IOS, Apple, web, inventario.

ABSTRACT

The project presented in this document is related to the use and implementation of Near Field Communication technology, abbreviated (NFC) using a mobile app, and a web server to store and process the obtained information to maintain control over the items that consist of the inventory of the company. The prototype developed in this project use a web server, a database, NFC tags, and the application made for working with IOS operative system which belongs to Apple company which is the second most use mobile brand in the market. Chapter one focuses on the type of project to be carried out. Chapter two presents the concepts about NFC technology, databases, IOS operating system, programming languages and all the software included. Also, the objectives and scope of the project. Chapter three describes the architecture and general development of the applications and how these interact between them. Chapter four involves the testing data and the parameters for the correct use of the software. The final chapter describes the conclusions, recommendations, and security practices that we gained through this project.

Keywords: NFC, NDEF, IOS, Apple, web, Inventory.

CAPITULO 1 . Descripción del proyecto

En el presente capítulo se describirá el campo donde se implementará el proyecto, los problemas presentados y los diferentes conceptos y metodologías que se van a aplicar.

1.1. Introducción

Debido a los avances tecnológicos que se dan con regularidad, en donde cada vez se ve la necesidad de automatización de procesos que normalmente dependen del control humano. Por desgracia ese control puede fallar en momentos cruciales cuando se necesitan recursos de forma inmediata o para conocer su ubicación.

Las entidades, ya sean públicas o privadas, necesitan tener un sistema que les permita llevar un sistema de organización para conocer de manera automática y rápida los registros, movimientos y actualización de los bienes muebles con los que cuenta la empresa. El principal objetivo que se debe cumplir es el de tener los registros de manera optimizada para evitar las pérdidas y saber de antemano la cantidad disponible de un bien y así ahorrar gastos operativos y tiempo destinado a esa tarea.

El problema al trabajar con la administración de inventarios es que cuando los bienes son movidos de sus respectivos lugares por la persona encargada, a veces sin una notificación previa, se pierde la capacidad de controlar la ubicación de dicho bien. Otro inconveniente es que, al cambiar el personal administrativo, los nuevos encargados desconocen de las acciones tomadas por los empleados que estuvieron previamente. Al darse estos casos la empresa incurre en pérdidas tanto financieras como administrativas.

El presente proyecto tiene la finalidad de desarrollar e implementar un prototipo de un sistema de inventario que por medio de una aplicación

desarrollada para el sistema operativo IOS se pueda comunicar con una aplicación web para visualizar diferentes datos como su ubicación, cantidad, encargado, etc. Para esto se utilizarán stickers con la tecnología NFC (Near Field Communication) que serán colocados en los bienes.

1.2. Antecedentes

En los últimos años ha crecido de manera exponencial, el uso de la tecnología en todos los ámbitos de la sociedad por su practicidad y eficiencia en las necesidades de la vida diaria.

La tecnología NFC tiene ya un tiempo en el mercado y es utilizada ampliamente en tarjetas de ingreso, documentos de identificación, pagos digitales, etc. Con el beneficio de que la distancia media entre la comunicación de los dispositivos es prácticamente bastante corta de alrededor de 4 centímetros, lo que evita que la comunicación sea fácilmente intervenida por agentes externos.

NFC viene siendo utilizada en el mercado desde hace más de 17 años y está siendo adoptada como un medio RFID (Radio Frequency Identification) alternativo que viene siendo utilizado en varios aspectos de la sociedad, desde su utilización en la economía como un medio de pago, en los deportes por su capacidad de guardar información sobre un balón, incluso en plataformas de videojuegos para vender equipos que trabajan con la misma y fomentar el fortalecimiento y la expansión de diferentes productos como el caso de la compañía Nintendo y sus Amiibos que usan esta tecnología. En dispositivos Android ya era posible desarrollar aplicaciones que pudieran leer y escribir datos en las tarjetas NFC, pero para sistemas operativos IOS, la opción de escritura es completamente reciente a partir del 2020 por el estrecho recelo de la compañía Apple para darle libre acceso al hardware con el que vienen sus equipos.

1.3. Problema

La Unidad Educativa Particular Virgen del Cisne, ubicada en la provincia de El Oro, en el cantón Machala, parroquia Puerto Bolívar, no cuenta con

un sistema automatizado que ayude a verificar de manera eficaz la correcta localización y cantidad de los recursos de los cuales disponen, para el correcto control y manipulación de los bienes por parte de los administradores.

1.4. Justificación

La Unidad Educativa Virgen del Cisne tiene 29 años de experiencia en la enseñanza de la educación básica y bachillerato unificado, donde se enfoca mayormente en dar una enseñanza integral donde se promulguen los valores y el respeto necesario para educar a excelentes ciudadanos que sean actores directos en el desarrollo de la comunidad Orense y sobre todo sean el futuro de la república del Ecuador.

La institución se encuentra ubicada en una zona geográfica donde cuenta con mucha competencia de otras entidades educativas públicas y privadas, donde se ganan a la clientela con calidad y precio.

En promedio se contaba con un mínimo de 300 estudiantes por año, al mismo tiempo que se invertía en infraestructura para seguir aportando la calidad exigida por los padres de familia y sobre todo los estudiantes quienes son los principales consumidores y a su vez exponentes de la excelencia académica aportada por la misma.

A raíz de la pandemia mundial que empezó a finales del 2019, que provocó contagios masivos que continúan hasta el día de escribir estas palabras, donde por las situaciones de confinamiento y la necesidad de normas de bioseguridad, vio disminuido su nivel de ingresos por la disminución del aforo de estudiantes, por lo que se ha tenido que recurrir a tecnificar la empresa en muchos aspectos, dentro de los cuales se incluyen reformas tecnológicas que ayuden durante la duración del brote epidemiológico y que brinden cualquier estrategia que se pueda implementar a futuro, donde entre tantas necesidades se descubrió que no había una manera correcta de contabilizar los inventarios sin tener una

constatación física de los activos que se tienen y se puedan adquirir, debido a que hay que limitar las interacciones sociales, esto propició la idea de este proyecto.

Los administradores de la institución utilizan dispositivos Apple por lo que para ellos es necesario un sistema que puedan verificar desde sus teléfonos móviles

1.5. Objetivos

Objetivo General

- Diseñar e implementar un sistema de inventario de bienes muebles usando la tecnología NFC en un sistema operativo IOS mediante una aplicación móvil para la Unidad Educativa Virgen del Cisne en la ciudad de Machala.

Objetivos Específicos:

- Desarrollar e implementar el software para la aplicación web.
- Desarrollar e implementar la aplicación móvil basada en IOS para leer y escribir las etiquetas NFC y permita una visualización detallada de los artículos que conforman el inventario.
- Desarrollar e implementar la comunicación entre la aplicación móvil y la aplicación web.
- Realizar una muestra con 5 etiquetas.

1.6. Hipótesis

La implementación de un sistema que utilice tecnologías de la información, como un sitio web y una aplicación desarrollada en IOS para la gestión del inventario de la Unidad Educativa Virgen del Cisne, ubicada en la provincia de El Oro, en el cantón Machala, Parroquia Puerto Bolívar, ayudará con el mejor registro y control de los bienes muebles, así como también reducirá gastos administrativos.

1.7. Campo de acción

La investigación que se plantea es el uso de una aplicación escrita para el sistema operativo IOS perteneciente a Apple, para el manejo del inventario, mediante el uso concreto de tecnología NFC.

1.8. Metodología de investigación

En el presente trabajo se utilizará las siguientes metodologías de investigación:

Cuasi Experimental: Se utilizará esta metodología ya que solo se puede controlar algunas de las variables de este caso de estudio al ser este orientado a un sector productivo específico.

Hipotética - deductiva: Se orientará con una observación de una realidad que se ha presentado en la actualidad y que va de la mano con el problema de investigación para emitir una hipótesis, la cual es analizada en las prácticas.

Deductiva: Las conclusiones formarán parte de la premisa del problema, donde las conclusiones serán visualizadas después de aplicar el método inductivo

Capítulo 2. Marco Teórico

El marco teórico es el que brinda los fundamentos en los que se basa la investigación, por medio de los conceptos entre los cuales están incluidos los básicos, complementarios y específicos. Por medio de este se le brinda al lector una comprensión más profunda del tema investigado. Los entes ya sean públicos o privados siempre están buscando alternativas que ayuden a mejorar sus procesos por medio de la innovación tecnológica.

2.1. ¿Qué es la tecnología NFC?

NFC es una tecnología inalámbrica que empezó usando como referencia métodos de identificación ya existentes como RFID, que tiene sus orígenes en la segunda guerra mundial, pero se asocia su invención a Charles Walton, en especial por que fue la primera persona en patentarlo.

La tecnología NFC fue desarrollada en conjunto entre Sony y semiconductores Philips en 2002.

NFC se sobrepone a RFID, en donde una de sus principales características son las distancias cortas de operación para mantener la seguridad. Los estándares para la tecnología NFC fueron aprobados por la International Standards Organization (ISO) a partir del año 2003 (Robert P. Sabella, 2016).

2.1.1. Foro NFC

En el año 2004 se estableció un foro entre Sony, Nokia y Philips en la ciudad de Hannover en Alemania, donde se establecieron los usos que se planeaba dar a la tecnología NFC, entre los cuales estaban las interacciones táctiles entre los usuarios con los dispositivos electrónicos, teléfonos móviles, computadoras personales y medios de pago.

Las interacciones táctiles ayudan al usuario a utilizar de manera intuitiva contenido y servicios simplemente tocando dispositivos inteligentes o sosteniéndolos cerca. El foro se llevó a cabo para establecer la estandarización de NFC.

NFC es el producto de una combinación de varias tecnologías RFID de interconexiones. Opera dentro del rango de frecuencia de los 13.56 MHz a una distancia de unos pocos centímetros (Vanderkay, 2004).

2.1.2. Estandarización de NFC

NFC ha sido estandarizado por organismos reguladores como ISO, IEC (International Electro-Technical Commission), ETSI (European Telecommunications Standards Institute) y por último ECMA (European Association for Standardizing Information and Communication System).

Los estándares asociados a NFC son por lo general dos, entre los cuales se tiene el ISO/IEC 14443 y ISO/IEC 18000-3 que dictan las reglas para la mayoría de las acciones que se realizan a través de esta tecnología. Los productos y servicios que se utilicen deben tomar en consideración estos estándares para evitar posibles problemas de compatibilidad entre ellos.

Hay un tercer estándar elaborado por la JIS (Japanese Industrial Standards) que propone reglas para etiquetas tipo 3 usadas por aplicaciones Felicity Card (FeliCa), que son tarjetas inteligentes sin contacto que se utilizan para tarjetas electrónicas de dinero que usa la compañía SONY (Sabella, 2016).

2.1.2.1. ISO/IEC 14443

El estándar ISO/IEC14443 define los requerimientos para tarjetas de proximidad que por lo general se usan para propósitos de identificación. El protocolo ayuda a diferenciar las tarjetas de proximidad de las de acoplamiento cerrado definidas por el ISO/IEC 10536 que sirve para los dispositivos de acoplamiento que trabajan a distancias extremadamente

cortas y las tarjetas de proximidad de mayor distancia para lo cual se usa el protocolo ISO/IEC 15693 (Sabella, 2016).

2.1.2.1.1. ISO/IEC 14443-1:2008 Características físicas

Aquí se indican esencialmente como las tarjetas son puestas juntas físicamente. El estándar se ocupa de dos tipos de tarjetas: Tarjetas de identificación ISO/IEC 7810 y las tarjetas más flexibles y finas que funcionan con el estándar ISO/IEC 15457-1. El estándar también aclara que la tecnología se puede presentar en otras formas diferentes a las mencionadas (Sabella, 2016).

2.1.2.1.2. ISO/IEC 14443-2:2010 Radio frecuencia y señal de interfaz

Se especifica las características de los campos que se utilizarán para poder proporcionar comunicación bidireccional entre tarjetas de proximidad u objetos (PICC, Proximity Integrated Circuit Card) y dispositivos de acoplamiento de proximidad (PCD, Proximity Coupling Device). La especificación como tal no indica que medios se deben usar para generar los campos. Los tipos A, B, y las tarjetas FeliCA utilizan diferentes sistemas de modulación y esquemas de codificación (Sabella, 2016).

2.1.2.1.2. ISO/IEC 14443-3:2011 Inicialización y anticolidión

Define como los procesos de comunicación deben empezar y terminar. Indica como un dispositivo debe buscar una conexión disponible y empezar con los comandos para iniciar la transferencia de datos. Las tarjetas tipo A, B y FeliCa utilizan diferentes procedimientos de inicialización de protocolos (Sabella, 2016).

2.1.2.1.3. ISO/IEC 14443-4:2008 Protocolo de transmisión

Aquí se definen los protocolos de mensajes para ser utilizados en la comunicación de las tarjetas de proximidad. Se describe el formato del mensaje cuando se lea o escriba una tarjeta y el mismo se utiliza para los

tipos A y B, mientras que FeliCa utiliza un estándar diferente para su protocolo especificada en JIS 6319-4 (Sabella, 2016).

2.1.2.2. ISO/IEC 18000-3

Este estándar define los requerimientos para la identificación de radio frecuencia (RFID) para la administración de los dispositivos, la parte que se centra en NFC se enfoca en estos puntos:

- Definir las reglas de comunicación entre los dispositivos por medio de las frecuencias que estén disponibles en los diferentes países.
- Especifica el mismo conjunto de reglas para cada frecuencia que entre en el rango de utilización de tal manera que la migración de una frecuencia a otra no represente un problema técnico mayor.
- Reducir los costos de software y la implementación que se utiliza en los equipos para poder expandir su utilización.
- Crear un método común que ayude a los controles del sistema como su administración y también para los procesos de intercambio de información en su totalidad dentro de las posibilidades

El estándar define tres modos de operación que no operan ni se produce una intervención entre ellos, por lo que los modos funcionan separados por completo. Cada modo a su vez trabaja a velocidades diferentes al transferir datos ya que cada uno usa diferentes técnicas de codificación de señales que trabajan paralelamente de tal manera que el usuario no nota la diferencia (Sabella, 2016).

2.1.2.3. JIS 6319-4 Felica

Este estándar fue originalmente propuesto como ISO/IEC 14443 (Tipo C), pero fue rechazado por el comité, por lo que es regulado por el Japan IC Card System Application Council (JICSAP), aquí se indica que la frecuencia de las etiquetas NFC trabajan a 13.56 MHz a 212 Kb/s (Sabella, 2016).

2.1.3. Modos de operación

NFC tiene 3 tipos de operación que por lo general son implementados en sus equipos y debido a esto también trabaja con el estándar ISO/IEC 18092 para conexiones Peer Two Peer (P2P), este estándar funciona también con el ISO/IEC 14443 para la comunicación entre un dispositivo y una etiqueta o tarjeta inteligente.

Para poder cubrir los dispositivos populares en el mercado, el estándar ISO/IEC 18092 es insuficiente ya que se necesitan de otros estándares entre los cuales el foro NFC incluyó el ISO/IEC 14443, JIS 6319-4 y el ISO/IEC 18092 en un solo protocolo que utilice la interfaz más común que es la inalámbrica.

Las especificaciones sirven para dar indicaciones a los productores como crear dispositivos que sean funcionales con todos los protocolos que existen al mismo tiempo. Esto permite que haya una respuesta inmediata y compatibilidad entre los productos que usen NFC, que dispone de dos modos de comunicación, la cual consiste en un iniciador que sería el dispositivo que empieza a transmitir y un blanco que recibe la señal del iniciador. Los 2 modos se clasifican en activa y pasiva (Sabella, 2016).

2.1.3.1. Modo de comunicación activa

Cuando se habla de un modo de comunicación activa, se refiere a dos dispositivos que cuentan cada uno con su respectiva fuente de poder. Los dos equipos utilizan señales de transmisión activa para comunicarse entre si ya que ambos crean un campo RF y la comunicación se da mediante la modulación de la señal.

En este modo, los dos dispositivos utilizan ASK (Amplitude Shift Keying) como su esquema de modulación. Para evitar problemas de colisiones el receptor apaga su campo para que haya un solo dispositivo transmitiendo al mismo tiempo. Las ventajas de este modo es que el uso de

transferencia de datos tiende a ser mayor y abarcar distancias mayores (Sabella, 2016).

2.1.3.2. Modo de comunicación Pasiva

En el modo de comunicación pasiva el iniciador envía energía RF a su blanco para encenderlo. El blanco o receptor modula la energía recibida para enviar datos al iniciador o emisor. Otra diferencia con el modo activo es que el receptor trabaja por medio de cargas de modulación, donde se hacen cambios a la amplitud de la señal original para transmitir los datos. No crea un campo por sí mismo, sino que modifica el campo original del iniciador para transmitir los datos (Sabella, 2016).

2.1.3.3. Modo de operación emulación de tarjeta

Esta modalidad coloca los dispositivos de NFC en modo de comunicación pasivo, por lo que los dispositivos trabajan de igual manera que una tarjeta inteligente. La seguridad de las tarjetas NFC se da a través de SE (Secure Elements) o HCE (Host Card Emulation) respectivamente. Esta seguridad solo protege los tokens que son utilizados para identificar a los individuos, por lo que se debe tener una aplicación que proporcione la seguridad para los datos.

El dispositivo NFC puede emular más de una tarjeta inteligente y a su vez el soporte para la tarjeta puede incluir más de un sistema de identificación de tal manera que si se cuenta por ejemplo con una billetera electrónica, esta a su vez pueda albergar varias tarjetas de crédito y débito a la vez (Sabella, 2016).

2.1.3.4. Modo de operación lectura/escritura

La mayoría de los dispositivos que trabajan bajo esta modalidad lo hacen en modo de lectura, por lo general lo hacen en modo activo para leer el contenido de una tarjeta. Cuando hay más de una tarjeta, utiliza el algoritmo de anticolisiones para seleccionar solo la que se necesita. Los dispositivos que trabajan con NFC deben poder identificar los tipos de

etiquetas, las cuales están bajo el estándar ISO/IEC 14443 A/B y FeliCa donde deben trabajar efectivamente y el método seleccionado para trabajar con la anticolisión depende del tipo de tarjeta.

Las aplicaciones deben actuar según los datos contenidos en las tarjetas, como por ejemplo si se tiene almacenada la dirección de un sitio web en la tarjeta y una aplicación la lee, se abre el explorador y automáticamente se accede a dicho sitio.

Los productores de las tarjetas pueden proporcionar tanto un bloqueo global como específico de ciertos datos. Cuando una tarjeta permanece abierta, se pueden escribir datos en la misma con alguna aplicación que soporte esas capacidades (Sabella, 2016).

2.1.3.5. Modo de operación Punto a punto (p2p)

Dos dispositivos pueden establecer una comunicación P2P (Peer to Peer) establecido por NFC, sin embargo, el iniciador empieza con su campo de radiofrecuencia encendido y el equipo con el cual se quiera conectar empezará con su campo RF apagado, o también puede encontrarse en modo pasivo para evitar consumir energía. El estado del campo RF cambia a la vez que la dirección de las comunicaciones cambia. Por lo general los dispositivos NFC se encuentran en modo de escucha en caso de que haya otro dispositivo transmitiendo, de esa manera evita problemas de interferencia en la comunicación.

Durante el inicio de la comunicación entre los dos dispositivos, definen los parámetros con los que van a trabajar, como por ejemplo el tamaño del bloque de datos. El tamaño máximo es 256 bytes.

Por lo general este modo se usa para emparejar los dispositivos que utilizan otro medio de comunicación ya sea wifi o bluetooth. Este proceso se denomina traspaso de comunicación (Sabella, 2016).

2.1.4. Etiquetas NFC

Cuando se habla de etiquetas NFC, se debe entender que la comunicación se da entre un interrogador y una etiqueta. Sus roles dentro de la comunicación difieren debido a que el interrogador cuenta con energía mientras que la etiqueta no, por lo que el proceso funciona de la siguiente manera:

El interrogador envía una señal hacia la etiqueta, y cuando los dos dispositivos están juntos, el interrogador enciende la etiqueta.

1. Los dispositivos cuentan con bobinas débilmente acopladas que crean un campo electromagnético entre ellas. Cuando el campo es establecido los dos equipos se pueden comunicar.
2. El interrogador envía el mensaje a la etiqueta para determinar qué tipo de etiqueta se usa, ya sea A o B.
3. La etiqueta responde con la información solicitada.
4. El interrogador a veces establece una comunicación segura con el canal si es requerido.
5. El interrogador envía un comando mediante la especificación apropiada.
6. La etiqueta recibe el mensaje para poder determinar su validez.
7. Si el comando es válido la etiqueta responde con los datos solicitados, caso contrario no hace nada.
8. Los pasos 6 y 7 se repiten hasta lograr la comunicación.

El interrogador se encarga de modular la señal RF para que trabaje a 13.56 MHz. Los datos se transfieren a diferente frecuencia y dependiendo del tipo de etiqueta a 106 Kbps, 212 Kbps y 424 Kbps respectivamente. Por último, la comunicación está regulada también por el software utilizado para llevar a cabo el proceso, el cual también condiciona el tipo de datos a transmitir (Sabella, 2016).

2.1.4.1. Transferencias de poder

Las antenas NFC trabajan como transformadores de núcleos de aire y no como fuentes de radiación. En la frecuencia de los 13.56 MHz el tamaño

de la onda es de 22 metros de largo. Para que se pueda radiar la energía de manera óptima como antena, la longitud de la antena debería ser de 11 metros. No se utilizan ese tipo de antenas debido a la imposibilidad de colocarlas en un teléfono inteligente. En realidad, la radiación emitida por una antena NFC es de 0.

Lo que realmente utilizan los dispositivos NFC son bobinas de bucle, aunque sean llamadas antenas, donde el uso continuo de la corriente alterna, produce un campo magnético continuo desde el teléfono inteligente. Cuando ese dispositivo se acerca lo suficiente a una etiqueta el inductor de bloque del dispositivo trabaja con la bobina de la etiqueta NFC donde ocurre un acoplamiento de las dos antenas, creando un transformador en el aire.

La comunicación en NFC se da por medio de una transferencia Half-Duplex en la frecuencia 13.56 MHz. En caso de necesitar una comunicación Full-Duplex, sería necesario incluir un segundo canal (Sabella, 2016).

2.1.4.2. Definición de los 5 tipos de etiquetas

Las etiquetas realizan procedimientos asociados a diferente tipo de información. Cada una cuenta con un tipo diferente de funcionalidad y por el momento solo 5 tipos de ellas están estandarizadas.

	Type 1 [5]	Type 2 [6]	Type 3 [7]	Type 4 [8]	Type 5 [9]
Supported standard	ISO/IEC 14443 A	ISO/IEC 14443 A	JIS X 6319 -4 (Felica)	ISO/IEC 14443 A/B	ISO/IEC 15693 (18000-3)
Carrier Frequency			13.56 MHz ± 7 KHz		
Data rate	106 kbps	106 kbps	212/424 kbps	106/212/424 kbps	26.48 kbps
Modulation (Reader to Tag)	ASK 100 %	ASK 100 %	ASK 10%	Standard A + ASK 10%	10% or 100% ASK
Data coding (Reader to Tag)	modified Miller	modified Miller	Manchester MSB first	NRZ-L (Std B)	Pulse position mod. 1 out of 256 / 1 out of 4
Modulation (Tag to Reader)	Load modulation (ASK) sub-carrier (± 848 kHz)	ASK 10%	Load modulation with no sub-carrier	Standard A + Load mod. (BPSK) sub carrier (Std B)	Load mod.
Data coding (Tag to Reader)	Manchester	NRZ-L	Manchester	NRZ-L	OOK/FSK sub-carrier Manchester
Anti-collision	No	Yes	Yes	Yes	Yes

Figura 2. 1 Estándares NFC
Fuente: (Bhattacharyya, 2018)

NFC Forum Platform					
	Type 1 Tag NFC-A compliant	Type 2 Tag NFC-A compliant	Type 3 Tag NFC-F compliant	Type 4 Tag NFC-A, NFC-B compliant	Type 5 Tag NFC-V compliant
Compatible Products	Broadcom Topaz	NXP MIFARE Ultralight/ NXP MIFARE Ultralight C NXP NTAG 21s (F) NXP NTAG I2C	Sony FeliCa	NXP DESFire / NXP SmartMX-ICOP (MIFARE DESFire implementation) / ST Microelectronics	NXP ICODE SLI (x) / Texas Instruments Tag-It HF-1 / EM423x / ST Microelectronics
Memory Size	454 Bytes	48 / 128 / 144 / 504 / 888 / 1904 Bytes	1, 4, 9 KBytes	2 / 4 / 8 KBytes / up to 144 Kbytes / 106 Kbytes	32 / 112 / 128 / 160 / 256 Bytes
Unit Price	Low	Low	High	Medium / High	Low / Medium
Data Access	Read/Write	Read/Write or Read-only	Read/Write or Read-only	Read/Write or Read-only	Read/Write
Active Content*	-	-	-	x / 0	-
Operation Specification	ISO/IEC 14443-3 A	ISO/IEC 14443-3 A	JIS 6319-4	ISO/IEC 14443-4 A/B	ISO/IEC 15693

*Active Content refers to the ability for a developer to store and run small applications (called "variants") on the tag. Only the Type 4, non-DESFire tags have this ability, all other tags are treated as static storage device.
MIFARE Ultralight, NXP MIFARE Ultralight C, MIFARE DESFire, SmartMX, NXP ICODE SLI(x) are registered trademarks of NXP B.V.
Tag-It HF-1 is a registered trademark of Texas Instruments.
EM423(x) is a registered trademark of EM Microelectronic.

Figura 2. 2 Cinco tipos de etiquetas NFC
Fuente: (Sabella, Dummies , 2016)

En el caso de etiquetas que se basen en tecnología Java, pueden incluir un contador automático que modifique el formato de intercambio de datos denominado NFC Data Exchange Format (NDEF) contenido en la tarjeta. En cada ocasión donde se haga una lectura de la etiqueta, el contador se incrementará automáticamente, lo que permite que se conozca la cantidad de veces que dicha tarjeta fue leída. Las etiquetas versión 4 de 32 Kb son las únicas que permiten esto actualmente (Sabella, 2016).

2.1.4.2.1. Etiquetas tipo 1

Es el tipo más simple de etiqueta y también las que contienen el chip más lento. Al sacrificar su complejidad, se incrementa el tamaño de su memoria. Por lo general debido a sus pocas funcionalidades son más económicas, son utilizadas en conjunto con las siguientes aplicaciones:

- Aprovechamiento único
- Aplicaciones de solo lectura
- Tarjetas de negocios
- Dispositivos de acoplamiento mediante bluetooth
- Lectura de una etiqueta específica cuando haya más de una etiqueta presente (Sabella, 2016).

2.1.4.2.2. Etiquetas tipo 2

Son el tipo de etiquetas más populares en el mercado porque ofrecen el mayor costo/beneficio al proporcionar una amplia variedad de funcionalidades a precios razonables. Son más veloces que el tipo 1, por lo que son confiables en aplicaciones de respuestas rápidas. Son utilizadas de la siguiente manera (Sabella, 2016):

- Transacciones de bajo valor
- Pases de tránsito diarios
- Tiques para eventos
- Redirecciones de URL (Uniform Resource Locator)

2.1.4.2.3. Etiquetas tipo 3

Las etiquetas tipo 3 utilizan un tipo diferente de formato ya que utiliza la innovación japonesa de FeliCa desarrollada por Sony y es ampliamente usada en Asia. Proporciona muchas funcionalidades, pero es costosa. Por lo general se usan en Japón con estas aplicaciones (Sabella, 2016):

- Tickets de tránsito.
- Dinero electrónico.
- Identificación electrónica.
- Tarjetas de membresía.
- Tiques electrónicos.
- Dispositivos de salud.
- Aparatos electrónicos

2.1.4.2.4. Etiquetas tipo 4

La etiqueta tipo 4, es de las más flexibles de todas las utilizadas, vienen con un buen espacio de memoria. En el mercado se encuentran a precios de rango moderado a alto. Por lo general este tipo de etiqueta se utiliza por sus capacidades de seguridad ya que ofrece funcionalidades para autenticación. Esta es la única etiqueta que soporta el estándar ISO 7816 para seguridad y también permite la modificación del contenido

NDEF, por sus atributos se utiliza para aplicaciones de tiques de tránsito (Sabella, 2016).

2.1.4.2.5. Etiquetas tipo 5

Estas etiquetas proporcionan soporte para ISO/IEC 15693. Incluye el modo de comunicación activa que permite la transferencia de datos en general similar a la tecnología RF soportado por el foro NFC. La distancia de lectura son las mismas de todas las tecnologías NFC que son soportadas por el estándar. Los dispositivos que trabajan con etiquetas tipo 5 pueden leer ISO/IEC 15693, por lo general se utilizan para aplicaciones como (Sabella, 2016):

- Libros, productos y paquetería.
- Paquetería de medicación médica, tiques

2.1.4.3. Proceso de lectura de una etiqueta

Cuando un lector NFC inicia una etiqueta, tanto el lector como la etiqueta debe seguir una secuencia específica para asegurar que la operación se dé correctamente. El lector y la etiqueta deben especificar el tipo utilizado y cuál debe ser seleccionada. Ejemplo de secuencia tipo A:

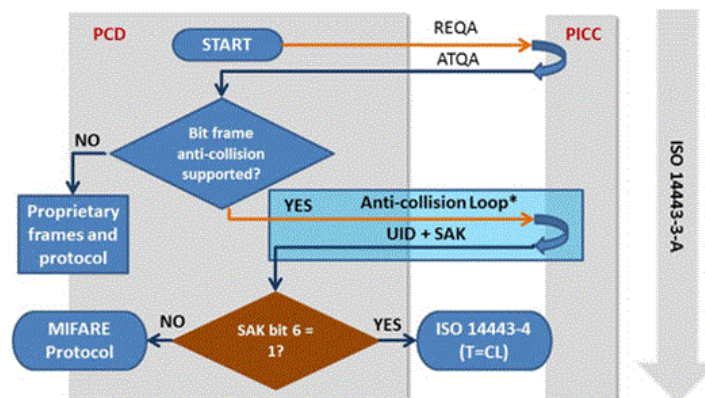


Figura 2. 3 Secuencia de Iniciación de etiquetas NFC
Fuente: (Sabella, Dummies, 2016)

Cuando se trabaja con NFC, se utiliza un dispositivo de acoplamiento de proximidad (PCD) y una tarjeta de proximidad de acoplamiento inductivo (PICC). El primero emite una señal de radio frecuencia (RF) que le otorga la energía al segundo. Pero en los casos de que haya más de un PICC

cerca de la distancia requerida para la interacción, se produce una colisión, por lo que es importante que al empezar la secuencia se defina cuál es el PICC con el que se va a conectar para establecer la comunicación y así prevenir los problemas de colisión. El proceso se lleva a cabo de la siguiente manera:

1. El PCD envía una petición (ReQA, Retrieval Question-Answering) hacia el PICC. Esta petición tiene solamente 7 bits. El PCD continúe enviando la petición hasta obtener una respuesta de la etiqueta. Los intervalos varían de acuerdo al vendedor.
2. El PICC responde con un bloque de respuesta para petición (ATQA, Answer To reQuest Type A), el cual contiene información específica como un prefijo identificador único (UID, Unique Identifier), y también le indica al PCD donde se proporciona una respuesta anticolidión por parte de la etiqueta. En casos de tecnología no propietaria, no da soporte por defecto a los estándares NFC y el prefijo no es el identificador completo, es solo una parte de este.
3. En el caso de las etiquetas que proporcionan soporte anticolidión, el PCD envía un comando SELECT donde va incluido el prefijo UID. Más de una etiqueta puede tener el mismo prefijo. Cuando se da ese caso el PCD envía otro comando SELECT con más partes del prefijo, y lo hace hasta que las respuestas de los dispositivos disminuyan y solo quede el indicado.
4. El último paso para evitar la anticolidión ocurre cuando el comando SELECT incluye el UID completo enviado por el PCD.
5. El PICC responde enviando un reconocimiento selectivo (SAK, Select Acknowledgement) hacia el PCD.
6. En este punto el PCD debe saber cómo interactuar con el PICC. Lo que ocurre cuando un SAK que contiene 6 bits, contiene a 1, se activa en el PICC el protocolo MiFARE. En caso de no tener soporte para ese protocolo se asume que se tiene soporte para ISO 14443-4.

7. El PICC continua todos estos procesos hasta que recibe un comando HALT por parte del PCD y el PICC se deshabilita (Sabella, 2016).

2.1.4.4. Formato de intercambio de datos (NDEF)

Por lo general la tecnología NFC utiliza su máximo potencial cuando es utilizada junto al sistema de intercambio de datos de punto a punto (P2P). Para que la comunicación se establezca de manera fluida, tanto el emisor del mensaje como el receptor deben establecer un protocolo que sea entendible por las dos partes.

Los mensajes NDEF proporcionan un método estandarizado para la comunicación entre el lector y el dispositivo NFC, mediante múltiples registros. El soporte de este protocolo viene dado por etiquetas estandarizadas y en el caso de los 5 tipos mencionados todos vienen con el mismo formato de mensajería NDEF.

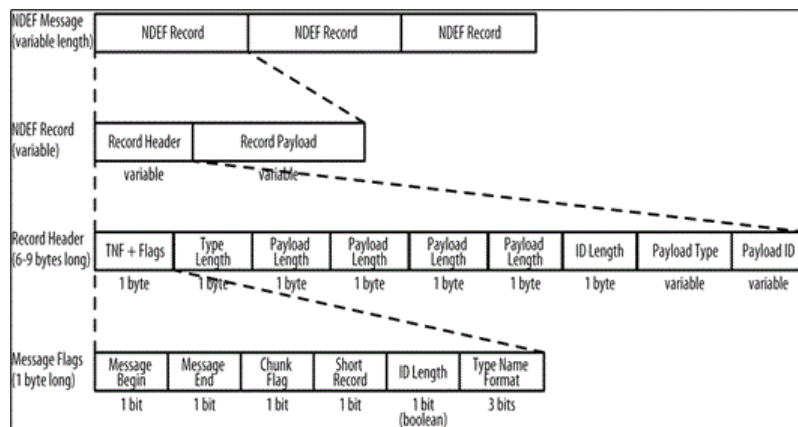


Figura 2. 4 Estructura de Mensajes NDEF
Fuente: (O'Reilly, 2021)

Cada registro contiene un encabezado (Header) y datos (Payload). El encabezado tiene información específica para el lector como el ID asociado al registro, su tamaño y tipo. El tipo define qué clase de datos son los que van a ser transmitidos por el Payload (Sabella, 2016).

2.1.4.4.1. Sistema de registro NDEF

Los registros contienen mucha información, los primeros 8 bits contienen información en forma de banderas, que sirve como esquema para interpretar el resto de los datos que están dentro del registro. De acuerdo a como estén habilitadas estas banderas, se pueden utilizar varios recursos.

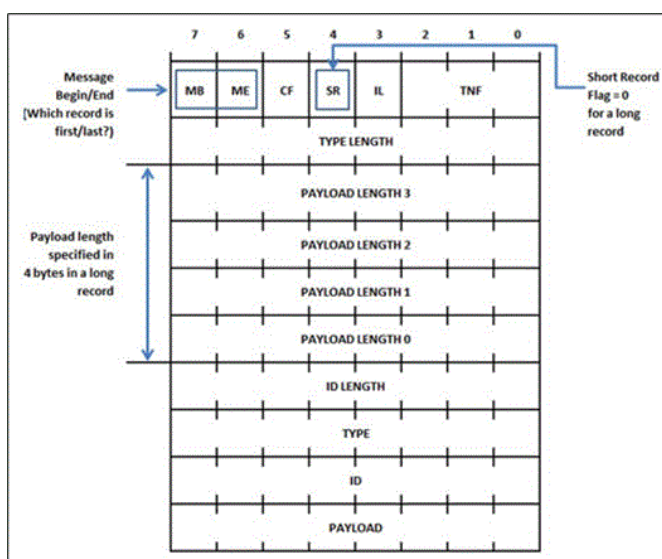


Figura 2. 5 Registros de Mensajes NDEF
Fuente: (O'Reilly, 2021)

El tipo de nombre del formato se denomina TNF (Type Name Format), este campo sirve para identificar qué tipo de contenido se encuentra dentro del registro. Los tipos de contenido que se puede encontrar dentro de un registro por lo general son los siguientes:

- **0 – Vacío(Empty).**- El registro no contiene ninguna información.
- **1- Bien-conocido (Well-Known).** - La información que se encuentra dentro del registro es definida por el Record Type Definition (RFD), que es la definición del tipo de registro. Esta es una especificación establecida por el foro NFC.
- **2- Extensiones multipropósito de correos de internet.** – Multipurpose Internet Mail Extensions (MIME), es uno de los tipos de datos comúnmente encontrados en las comunicaciones de internet definidas por RFC 2046.

- **3- Identificador absoluto uniforme de recursos.** – Absolute Uniform Resource Identifier (URI), es un puntero a un recurso que sigue la sintaxis de RFC 3986.
- **4- Externo.** – Estos son datos definidos por los usuarios, que se basan en los formatos de la especificación RTD.
- **5- Desconocido.** – Son tipos de datos desconocidos, por lo que se establece el tipo de tamaño en 0.
- **6- Sin cambios.** – Algunos datos se envían en partes debido a que son demasiado grandes para caber en un solo registro, por lo que existen varios registros que contienen partes de los datos. El TNF indica que no es el primer registro de todos los datos, es uno de la mitad o del final. El TNF no cambia el tipo de dato encontrado en el primer registro.
- **7- Reservado.** – Es un valor reservado para usos futuros.

El IL Flag comunica cuando un registro posee un campo para el tamaño del ID, a su vez no indica el tamaño específico de este, sino el tamaño disponible para su uso.

El Flag SR determina si el registro es corto, los cuales por lo general tienen un tamaño menor o igual a 255 Bytes. Un registro normal va desde esa cantidad hasta los 4 Gb. La bandera SR permite el uso de una cabecera comprimida para datos específicos en un solo byte en lugar de los 4 normalmente requeridos. La bandera CF indica cuando los datos solo son una parte de un único registro.

Un mensaje NDEF puede contener múltiples registros, donde el primero contiene un flag MB (Message Begin) para indicar el inicio del mensaje y el último tiene un flag ME (Message End) para indicar el final del mismo. Los flags son banderas o códigos booleanos donde solo se indican dos estados: verdadero y falso. Todos los mensajes intermedios entre el inicio y el fin tienen valores falsos de MB y ME, mientras que los mensajes que

si corresponden al tipo requerido de inicio y fin tienen el valor de verdadero.

El campo del tipo de tamaño especifica la cantidad del volumen de datos expresados en bytes y el tipo de datos específicos. Un ejemplo claro sería el TNF que a pesar de que es un mensaje tipo MIME, esa información por sí sola no es suficiente para procesar los datos. También se debe conocer las características de la información, puede ser texto plano, como uno de muchos tipos.

El campo del tamaño de los datos (Payload) contiene el registro en bytes de los datos que se almacenan. El campo de tamaño del ID tiene la misma funcionalidad (Sabella, 2016).

2.2. Iphone os (IOS)

La compañía Apple desarrolló dos sistemas operativos que son exclusivos para sus equipos. Mac OS para los computadores personales de escritorio y portátiles de la marca y iPhone OS (IOS). Siendo el primero disponible para arquitecturas X86-64 ya sean AMD o INTEL y el segundo para arquitecturas basadas en ARM, la que se utiliza en los equipos celulares por su bajo consumo energético, lo que lo ha expandido de manera global en el uso de dispositivos móviles de todo tipo incluyendo teléfonos celulares y tabletas.

Ambos sistemas operativos están basados en UNIX y especialmente IOS está orientado a funcionar con interfaces multi-touch. Los sistemas reciben soporte regularmente con cada nuevo modelo de teléfono celular que sale y la última versión es IOS 14 en el mercado al momento de escribir este trabajo (Honan, 2007).

2.2.1. Arquitectura de IOS

A raíz de la adquisición de NeXT por parte de Apple, se obtuvo acceso a la arquitectura de NeXTSTEP el sistema operativo de la compañía.

Después de que se utilizara componentes de la arquitectura original Mac OS y se complementara con aspectos de la arquitectura de NeXT, el desarrollo del nuevo sistema operativo que se utiliza hoy en día pasó por varias transiciones desde el uso de un sistema operativo interno llamado Rhapsody, que nunca fue mostrado al público, aunque a partir de este mismo sistema operativo se daría la evolución de lo que se conoce hoy como Mac OS X y su núcleo interno llamado Darwin. Los componentes claves de OS X son Cocoa, Mach, IOKit, la interfaz de desarrollo de XCode entre muchas otras.

No se debe confundir OS X con Darwin ya que el primero es el sistema operativo como tal, mientras que Darwin es su kernel basado en Unix que a su vez es basado en el kernel XNU (acrónimo de X is Not Unix). Darwin como tal es open source, excepto por las adaptaciones para procesadores ARM de parte de Apple iOS.

Las diferencias entre OS X e iOS aparte de la arquitectura a la que van orientados es la eficiencia de iOS en el uso de dispositivos móviles, como el manejo de la memoria, el endurecimiento o hardening del sistema que no permite que se pueda acceder fácilmente con privilegios de root y solo deja que las aplicaciones puedan acceder a sus propios directorios. La arquitectura de iOS viene dada de la siguiente manera:

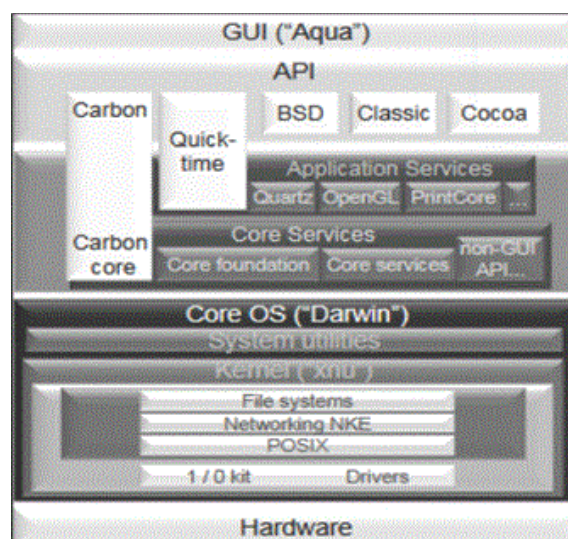


Figura 2. 6 Arquitectura de iOS
Fuente: (Gangula, 2019)

Donde se aprecia que se reduce en varias capas que son representadas por la siguiente figura y donde se puede ver la separación:

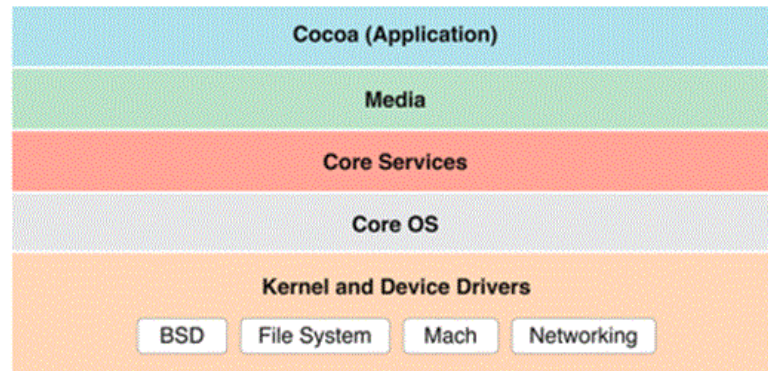


Figura 2. 7 Separación de los servicios de IOS
Fuente: (Apple, 2015)

La primera capa se ocupa de la interacción entre el usuario y el dispositivo (Apple, 2015).

2.2.1.1. Cocoa y Carbón

Carbón es el nombre que llevan las interfaces antiguas de OS 9 que ahora son declaradas obsoletas para muchas aplicaciones, a pesar de eso Apple todavía confiaba en esta interfaz y la usaba para compatibilidad de esas aplicaciones (Apple, 2011).

Por otro lado, Cocoa es el entorno de programación preferido basado en el entorno de NeXTSTEP, lo cual resulta evidente por el prefijo de muchas de sus clases base NS el cual es dado por NeXTSTEP/SUN. El lenguaje preferido para la programación de Cocoa es Objective C, a pesar de que puede ser accedido por Java y AppleScript (Singh, 2006).

2.2.1.2. BSD/Match

A pesar de que el lenguaje preferido para IOS y OS X es Objective-C, las aplicaciones nativas pueden utilizar C/C++ y pueden optar por renunciar a los frameworks, trabajando directamente con las librerías del sistema y las interfaces de bajo nivel de BSD y Mach en su lugar. Esto a su vez permite un enfoque directo de portabilidad de códigos que son basados en UNIX como es el caso de Apache, PHP, SSH, así como muchos otros productos

que son Open Source. Otras iniciativas como MacPorts y Fink permiten que las fuentes de estos paquetes puedan ser utilizadas en el modelo utilizado por las distribuciones de Linux como son RPM, APT, DEB.

La funcionalidad de POSIX de OS X permite que sea muy fácil portar aplicaciones, confiando en las librerías y llamadas al sistema estándar. El mismo caso también es para IOS donde los desarrolladores tienden a portar todo. Sin embargo, hay otro conjunto de APIs que son específicas de OS X (Wells, 1994).

2.2.1.2. El kernel Darwin

El núcleo Darwin es una implementación completamente funcional de UNIX y es parte de XNU. Es un Kernel Híbrido que utiliza OSFMK (Open Software Foundation Mach Kernel) desde OSF en conjunto con varios elementos de FreeBSD donde se encuentra modelo de procesos, pilas de red, un sistema virtual de archivos y una API orientada a objetos la cual se denomina el kit I/O. El diseño proporciona flexibilidades a nivel de microkernel y se comporta de manera monolítica (Apple, 2015).

2.3. Xcode y Swift

El lenguaje Swift nació en julio de 2010 de parte de Chris Lattner con el nombre de Shiny, después en 2014 el lenguaje fue renombrado Swift y no fue aceptado de manera inmediata por la comunidad.

Anteriormente la programación de Cocoa sobre IOS y antes de eso en Mac OS siempre fue hecha en Objective-C. El framework de Cocoa que les da las funcionalidades a las aplicaciones desarrolladas para IOS que están basadas en Objective-C.

Xcode es un entorno de desarrollo integrado o IDE desarrollado por Apple para crear software para MacOS, IOS, IpadOS, watchOS y tvOS (Neuburg, 2020).

2.3.1. XCode

Xcode es el software que se utiliza para crear aplicaciones para IOS. Los proyectos de Xcode son las fuentes de las aplicaciones y están compuestos por una colección de archivos y configuraciones utilizadas para desarrollar la aplicación.

Xcode tiene dos significados, siendo uno el nombre del programa que se utiliza para desarrollar la aplicación, así como también es un conjunto de utilidades complementarias. Los instrumentos y el simulador son parte de Xcode (Neuburg, 2020).

2.3.1.1. Nuevo proyecto (new project)

Para empezar un nuevo proyecto en Xcode se necesita abrir la aplicación y seguir los siguientes pasos (Neuburg, 2020):

- Empezar Xcode y escoger File → New → Project.
- Luego se debe escoger una plantilla (Template) la cual sirve para elegir los archivos que formarán parte de la configuración del proyecto.
- Luego se pide indicar un nombre para el proyecto. En el momento que Xcode comienza a colocar ese mismo nombre en varios lugares incluso en el nombre de la aplicación. El nombre puede ser cambiado después en las configuraciones. Se soportan espacios en los nombres del proyecto, pero no caracteres especiales.
- Luego le pide introducir el nombre de la organización.
- El nombre de la organización se coloca la primera vez con la finalidad de crear una cadena de caracteres que identifique a la organización y por lo general a este identificador las convenciones establecen que empiece por (com.) y luego debe ser seguida por una cadena que contenga multiples puntos de ser preferible donde la cadena no se confunda con una utilizada por otra compañía. Cada aplicación desarrollada en un dispositivo o subida en la App Store necesita un identificador único. El identificador del paquete

de la aplicación se muestra debajo del identificador de la organización el cual está en gris que va a ser consistente con la versión por defecto de la empresa mas la versión del nombre del proyecto. Cuando se elige un solo identificador para la organización.

- El menú emergente de la interfaz debe decir StoryBoard no SwiftUI.
- El menú emergente permite escoger entre swift y Objective-c. Esta opción no es positivamente vinculante. Inicialmente al escoger el lenguaje se define la estructura del proyecto, sin embargo, se es libre de agregar archivos de Objective-c a un proyecto de Swift y viceversa.
- Xcode también ofrece la opción de crear un repositorio Git para el proyecto que se quiera elaborar.

2.3.1.2. Pantalla del proyecto

En un proyecto Xcode se muestra mucha información relacionada a los componentes que lo conforman y como son utilizados cuando se crea la aplicación:

- El código fuente de los archivos que van a ser compilados.
- Cualquier archivo `.storyboard` o `.xib` expresando gráficamente la interfaz donde se crean las instancias a medida que se ejecuta la aplicación.
- Cualquier recurso que sea necesario para la app, como iconos, imágenes o archivos de sonido.
- Todas las configuraciones necesarias que debe cumplir la app mientras es compilada.
- Cualquier framework necesario para que la aplicación funcione.

La ventana de un proyecto de Xcode permite observar toda esta información, y que se pueda acceder fácilmente al código fuente. Tiene muchas funcionalidades descritas a continuación (Neuburg, 2020):

- 1) Panel de navegación.**
- 2) Editor de código fuente**
- 3) Panel inspector**

4) Panel de depuración.



5) **Figura 2. 8** Ventana del proyecto
Fuente: (Neuburg, 2020)

2.3.1.2.1. Panel de navegación

El panel de navegación que se muestra en el lado izquierdo del proyecto, es el mecanismo primario para poder administrar los archivos que contienen el proyecto como tal.

Al seleccionar un archivo del panel de navegación, este se muestra en el editor que se encuentra del lado derecho. El panel de navegación se puede ocultar o maximizar. Se puede cambiar el tamaño del navegador con el ratón y a partir de Xcode 12 la ventana aparece oculta y se muestra si se posiciona el ratón encima de la misma.

El panel de navegación muestra nueve tipos de información porque lo que en realidad se podría decir que hay nueve tipos de paneles (Neuburg, 2020).

2.3.1.2.1.1. Navegador de proyecto

Aquí se puede observar la navegación básica a través de los archivos que constituyen el proyecto. La carpeta ventana vacía es donde se puede ver el archivo AppDelegate.swift que al seleccionarlo se puede observar su código fuente en el editor.

En la parte superior del navegador del proyecto con el icono de color azul de Xcode, esta la ventana vacía del proyecto. Al hacer click sobre la misma se

puede observar las configuraciones asociadas al proyecto y la barra de filtros en la parte de abajo permite limitar que archivos se muestran (Neuburg, 2020).

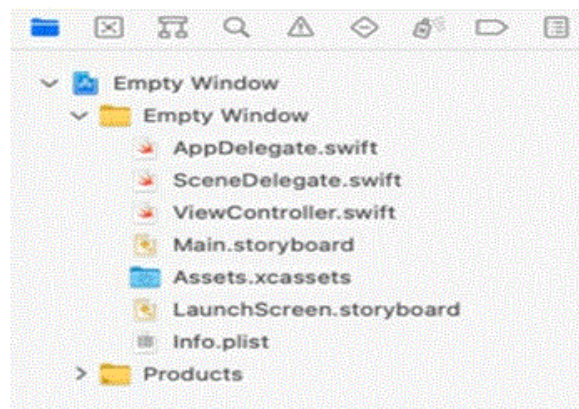


Figura 2. 9 Pantalla navegador del Proyecto
Fuente: (Neuburg, 2020)

2.3.1.2.1.2. Navegador de control de código fuente

Este navegador ayuda como los archivos del proyecto son manejados, así como su sistema de versiones, para llevar un mejor control del código fuente (Neuburg, 2020).

2.3.1.2.1.3. Navegador del símbolo

Un símbolo (Symbol) es un nombre, típicamente es de la clase o método. El navegador de símbolos muestra la lista de símbolos disponibles para el código fuente. Entre muchas de sus funciones es útil para navegar ya que resalta los iconos en la barra de filtros, permitiendo abrir listados de clases y eligiendo rápidamente las implementaciones de métodos creados (Neuburg, 2020).

2.3.1.2.1.4. Navegador de búsqueda

El navegador de búsqueda permite buscar texto de manera global en el proyecto. Las palabras en la parte superior del buscador muestran las opciones que se encuentren disponibles en ese instante.

Hay opciones de filtrado donde también se pueden buscar desde referencias hasta definiciones de un texto específico, además de

proporcionar la opción de limitar la búsqueda a carpetas específicas y tipos de archivos (Neuburg, 2020).

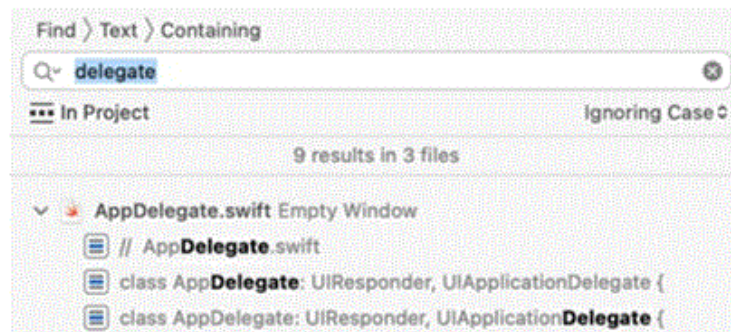


Figura 2. 10 Navegador de Búsqueda
Fuente: (Neuburg, 2020)

2.3.1.2.1.5. Navegador de problemas

Este navegador es donde se muestra cualquier error o advertencias que se den durante la ejecución del código fuente, muestra los errores asociados con este como el uso de palabras ilegales y resaltando en donde se está originando el problema (Neuburg, 2020).

2.3.1.2.1.6. Navegador de pruebas

Enumera archivos de pruebas y métodos de estas de manera individual, permitiendo ejecutar pruebas y comprobar que sean exitosas. Una prueba no es un código que forme parte de la aplicación y trabaja ejecutando parte del código de la aplicación o la interfaz de esta para ver si su comportamiento es el esperado (Neuburg, 2020).

2.3.1.2.1.7. Navegador de depuración

Este navegador se muestra de manera determinada cuando el código se pausa mientras se depura. Se deben asignar puntos de interrupción para poder ver los errores que se muestran mientras se trabaja en la depuración.

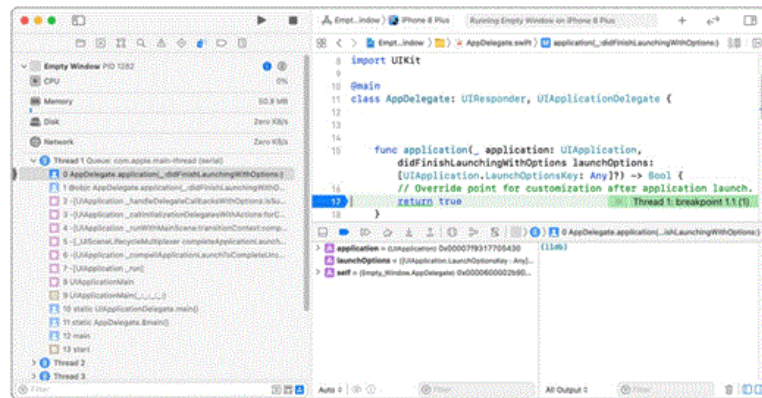


Figura 2. 11 Navegador de depuración
Fuente: (Neuburg, 2020)

Se muestran varias pantallas gráficas y numéricas con información de creación de perfiles, entre los cuales se puede ver como mínimo el CPU, la memoria, disco duro y el uso de la red. Si se hace click sobre una de las opciones mostradas se puede ver información más extensa.

Con el navegador de depuración también se muestra la pila de llamadas, con nombres de métodos anidados en donde se puedan dar los errores y se puede hacer click sobre el método para ver su código (Neuburg, 2020).

2.3.1.2.1.7. Navegador de punto de quiebre

El navegador enumera los puntos de interrupción. Aquí también se crea puntos de interrupción especiales como puntos simbólicos y es el centro de administración de puntos existentes (Neuburg, 2020).

2.3.1.2.1.8. Navegador de reportes

El navegador enumera sus acciones principales recientes. Como la construcción o ejecución del proyecto. Aquí se puede observar información que no se puede mostrar de otra manera y mensajes de la consola pasados (Neuburg, 2020).

2.3.2. Swift

Swift es un lenguaje fácil de usar que se ha popularizado hasta ser una de las principales herramientas de desarrollo en IOS por las siguientes características (Neuburg, 2020):

- Orientado a Objetos.- Es una versión moderna de la orientación a objetos en la manera mas pura. Todo es básicamente un objeto.
- Claridad.- Es fácil de leer y de escribir, con una sintaxis relativamente sencilla de aprender.
- Seguridad.- Utiliza el tipeado duro para establecer siempre cada referencia de un objeto.
- Economía.- Como lenguaje es bastante ligero, ofreciendo las funcionalidades básicas. El resto debe ser creado por el desarrollador, o librerías que se puedan utilizar como el caso de Cocoa.
- Administración de Memoria.- Hace un manejo automático de la memoria por lo que se vuelve una preocupación menor a la hora del desarrollo.
- Compatibilidad con Cocoa.- Las APIs de Cocoa estan desarrolladas en C y Objective-c. El lenguaje puede trabajar con estas APIs sin problemas.

2.3.2.1. Características principales

El lenguaje de programación Swift cuenta con algunas particularidades que lo diferencian de otros lenguajes usados anteriormente para desarrollo en IOS como es las disposiciones de su sintaxis a la hora de desarrollar el código fuente (Neuburg, 2020).

2.3.2.1.1. Sintaxis básica

Un comando de Swift es una declaración. Un archivo de texto Swift consta de varias líneas de texto donde los saltos de líneas son significativos (Neuburg, 2020).

- El diseño típico de un programa es de la siguiente manera también puede ser colocado saltándose una línea hacia abajo `print("Hola mundo")`.
- Ahora en el caso de que se desee ponerse mas de un comando en la misma línea se debe colocar un punto y coma como se muestra a continuación: `print("Hola mundo") ; print("Hola mundo")`. Se puede

colocar un punto y coma al final, aunque no sea necesario de cada línea como se hace en lenguaje C.

- Para colocar un comentario se utiliza el siguiente comando para una sola línea “//” y para comentar párrafos /*...*/.
- Las construcciones en swift usan llaves como delimitadores en general, y los contenidos suelen estar precedidos y seguidos por salto de línea por convención, aunque no es necesario que se respete la misma.

```
class Cat { func meow() { print(“miau”) }
```

2.3.2.1.2. Orientado a objetos

Swift es un lenguaje compilado por lo que significa que el código primero debe ser construido luego pasar por el compilador y luego ser convertido de texto en un lenguaje de bajo nivel que el computador pueda interpretar.

Swift es un lenguaje que se enfoca mucho en evitar los errores de programación por lo que mandara advertencias y no se ejecutara en primero lugar si encuentra fallas en el código fuente. En Swift todo es tratado como un objeto y la sintaxis para utilizar un método o mensaje es la notación del punto. Por ejemplo, el objeto Carro.Encender() donde el objeto seria Carro y Encender separado por el operador punto es el método asociado.

En Swift cada sustantivo es un objeto y cada verbo es un mensaje o método el cual puede ser creado por el mismo programador. Existen 3 tipos de objetos en el lenguaje Swift los cuales son clases(class), estructuras(struct) y enumeraciones(enum). Un entero o Int por ejemplo pertenece a los objetos tipo struct, esta es una diferencia en como Cocoa actúa con Objective-C (Neuburg, 2020).

2.3.2.1.3. Estructura del archivo Swift

Un programa elaborado en Swift consiste en uno o múltiples archivos. Un archivo tiene una estructura de la siguiente manera (Neuburg, 2020):

- Importación de módulos
- Declaración de variables
- Declaración de funciones
- Declaración de tipos de objetos

2.3.2.1.3.1. Importación de módulos

Un módulo es una unidad más grande que un archivo ya que el mismo consiste en múltiples archivos donde todos pueden verse automáticamente. Los archivos de una aplicación pertenecen a un solo módulo. Un módulo no puede tener alcance a otro modulo sin antes utilizar la declaración **import** así se comunica Cocoa en un programa de IOS y la primera línea que se agrega es: `import UIKit`.

(Neuburg, 2020)

2.3.2.1.3.2. Declaración de variables

Una variable que se declara al inicio de un archivo es de carácter global, por lo que cualquier código en cualquier archivo tendrá libre acceso a la misma (Neuburg, 2020).

2.3.2.1.3.3. Declaración de funciones

El caso de las funciones es similar al de las variables. Aquellas declaradas al inicio del archivo son de carácter global y serán vistas por todos (Neuburg, 2020).

2.3.2.1.3.4. Declaración de tipos de objetos

Es la declaración para una clase, estructura o enumeración (`class`, `struct`, `enum`), a continuación, el ejemplo básico de un archivo de Swift (Neuburg, 2020):

```
import UIKit
Var uno = 1
func cambiarUno(){
}
class Muchos {
}
struct Rafael {
```

```
}  
enum Guillermo {  
}
```

2.3.2.1.3.5. Alcance y tiempo de vida

En el lenguaje Swift los objetos, variables, etc. Poseen un alcance de donde pueden ser accedidos por un método, objeto, etc. Por lo general se pueden tener alcance a objetos que estén dentro del mismo nivel y de jerarquía superior. Tanto los módulos, archivos y llaves proporcionan su propio nivel de alcance.

El tiempo de vida también está asociado a su alcance, es decir, una variable por ejemplo vive según lo que el alcance se lo permita. Si se tiene la siguiente función:

```
func Hola () {  
    if true {  
        class suma {  
            var uno= 1  
            uno = uno+1  
        }  
    }  
}
```

Debido al alcance del ejemplo anterior se puede observar que el alcance de la variable uno empieza y acaba dentro de la clase suma. Después de que se asigna 1 y se suma otro para volverse 2, la variable desaparece y no se vuelve utilizar (Neuburg, 2020).

2.3.2.1.3.6. Objetos miembros

Dentro de los 3 tipos de objetos (class,enum, struct), lo que se declara en el nivel superior tienen nombres especiales (Neuburg, 2020).

2.3.2.1.3.6.1. Namespaces

Un namespace es una región con nombre de un programa. Los nombres con las cosas dentro del espacio del nombre no pueden ser alcanzados por las cosas fuera de el sin pasar primero por la barrera de dicho nombre de esa región (Neuburg, 2020).

2.3.2.1.3.6.2. Módulos

Los espacios de nombre o namespaces de nivel superior son llamados módulos. Una aplicación es un módulo y por lo tanto un namespace y es el nombre por defecto que lleva la aplicación. Por ejemplo, si se crea una aplicación con el siguiente nombre Miapp, se puede invocar el siguiente método perteneciente al módulo por medio de Miapp.Metodo. Esto no es necesario cuando se trabaja dentro del mismo modulo, pero si cuando se importa un módulo externo a la aplicación (Neuburg, 2020).

2.3.2.1.3.6.3. Instancias

Los tipos de objetos mencionados con anterioridad tienen una función en común, la cual es que todos pueden ser instanciados. Cuando se declara un tipo de objeto solo se declara un tipo. Para instanciar un tipo es hacer una instancia de ese tipo.

```
class Perro {  
  func ladrar () {  
    print("Ladrar")  
  }  
}
```

Se define una clase perro y ahora se crea una instancia de esa clase (Neuburg, 2020).

```
let rocky = Perro()  
rocky.ladrar()
```

2.3.2.1.3.6.4. Privacidad

Una declaración de clase define un espacio de nombres. Este espacio de nombre requiere que otros objetos usen un nivel adicional de notación de puntos para referirse a lo que está dentro del espacio de nombres, pero otros objetos aún pueden hacer referencia a lo que está dentro del espacio de nombre; el espacio de nombres no cierra, en sí mismo, ninguna puerta de visibilidad. La palabra reservada **private** permite que se cierren esas puertas (Neuburg, 2020).

2.3.3. Cocoa

Es una API (Application Programming Interface) nativa de Apple y está constituida por 3 frameworks las cuales son Foundation Kit, Application Kit y Core Data.

- **Foundation Kit** .- Es un framework de Objective-C que proporciona soporte para clases básicas como clases contenedoras y clases de estructura de datos. Es parte de la biblioteca de Cocoa y por ende de Swift
- **Application Kit** .- Es una herramienta para la interfaz gráfica de usuario. Sirven para realizar las interfaces visuales de las aplicaciones
- **Core Data** .- Es un framework de objetos y persistencia hecho por Apple para sistemas operativos Mac. Permite que los datos que se organizan en el modelo entidad relación sean serializados en XML, binarios o SQL ligero o lite.

Las aplicaciones desarrolladas en Cocoa son por medio de XCode por medio de Objective-C y Swift, aunque puede ser utilizado con otros lenguajes (Apple, 2018).

2.3.3.1. Administración de memoria de Cocoa

La administración de memoria es utilizada de similar manera tanto por Objective-C como con Swift, pero este último utiliza ARC (Automatic Reference Counter) pero incluso con este recurso se debe administrar cuidadosamente la memoria para evitar desbordes.

La razón principal por la que la memoria de tipo de referencia debe administrarse, es porque los tipos de objetos son punteros a los verdaderos objetos y a su vez estos pueden tener más de uno.

Para evitar que ocurra la pérdida de memoria y los punteros colgantes, existe una política de administración manual de memoria que se basa en un número mantenido por cada objeto de tipo de referencia que se identifica como recuento de detención. Otros objetos pueden incrementar

o disminuir el valor de retención de un objeto, mientras el recuento sea positivo, el objeto persistirá. Solo en caso de que el recuento caiga a cero, el objeto asociado es destruido (Neuburg, 2020).

2.3.3.2. Reglas de la administración de memoria

- Si un objeto 1 instancia explícitamente un objeto 2 llamando directamente a un inicializador, entonces el inicializador incrementa el conteo de retención del objeto 2.
- Si un objeto 1 hace una copia de un objeto 2, utilizando métodos como mutableCopy o que tengan copy en su nombre, entonces el método copy incrementa el recuento de retención del objeto 2 duplicado.
- Si un objeto 1 adquiere una referencia a un objeto 2 que no sea a través de una instanciación o copia explícita y necesita que el objeto 2 persista en el tiempo que sea suficiente para trabajar con el objeto 2 en el código, o el suficiente tiempo para ser valor de una instancia entonces el mismo incrementa el valor del objeto 2.
- Si un objeto 1 ha realizado algunas de las acciones mencionadas, o si el objeto ha provocado directa o indirectamente que el conteo de retención del objeto 2 se incremente, entonces cuando el mismo ya no necesite esa referencia al objeto 2, antes de soltarlo, esa referencia disminuye el recuento del objeto 2 para equilibrar todos los incrementos anteriores que el mismo ha realizado. Cuando se libera el objeto 2, el objeto 1 debe asumir que ya no existe por lo que su recuento debe ser cero (Neuburg, 2020).

2.3.3.3. Funcionamiento de ARC

NSObject implementa la retención y liberación de memoria bajo ARC, el cual esta implementado como parte del compilador. Por medio del compilador se insertan llamadas de retención y liberación.

Cuando recibe un objeto de tipo referencia o algún método, ARC lo retiene para que persista mientras este mismo código continúe

ejecutándose y luego se libera cuando llega a su fin. Cuando se crea un objeto o se copia que sea de tipo de referencia ARC sabe que su recuento se ha incrementado y lo libera al finalizar el código. Se utiliza el conteo automático de referencias para rastrear y administrar el uso de memoria de las aplicaciones.

En algunos casos ARC requiere la información que esta relacionadas en el código fuente de una aplicación específica. El recuento de las referencias solo se aplica a instancias de clases. Las estructuras y enumeraciones son tipos de valor y no de referencia por lo que no pasan por referencia y no se almacenan.

Cada vez cuando es creada una nueva instancia de clase ARC asigna una parte de esa memoria para almacenar información sobre esa instancia. En la memoria se guardará información sobre el tipo de instancia y los datos y características asociadas a la instancia en particular.

Cuando una instancia deja de ser necesitada se elimina información de memoria de esa instancia para que sea utilizada con otros fines para evitar que el espacio de memoria siga en uso. Cuando ARC elimina la asignación de una instancia que estaba en uso, esto imposibilita que se pueda acceder a la misma y la aplicación se bloquee.

Para evitar que una instancia desaparezca cuando aún está siendo utilizada ARC la mantendrá asignada mientras posea una referencia activa. Con el fin de lograr este cometido, cuando se asigne una instancia de una clase a una variable, propiedad o constante, estas últimas crean una referencia fuerte a esa instancia la cual se mantiene mientras la referencia fuerte exista (Swift, 2021).

En el siguiente ejemplo hay dos clases llamadas Job y Person. La clase Job tiene una variable llamada person con el tipo de clase llamado

Person, de la misma manera se tiene una variable Job con el tipo de clase del mismo tipo que es creada en Person. Las propiedades son creadas en el método DidLoad(). Las dos variables creadas en el método tienen su espacio en memoria y un contador de referencia de 1. Joe es una variable de tipo persona y dev es una variable perteneciente a Job.

joe?.job=dev

La variable de joe está referenciando al objeto dev, al hacer esto joe tiene un contador de referencias de dos de tal manera que dos objetos apuntan a este.

dev?.person=joe

En este momento la variable dev apunta a la clase Person por lo que también tiene un conteo de referencia de dos.

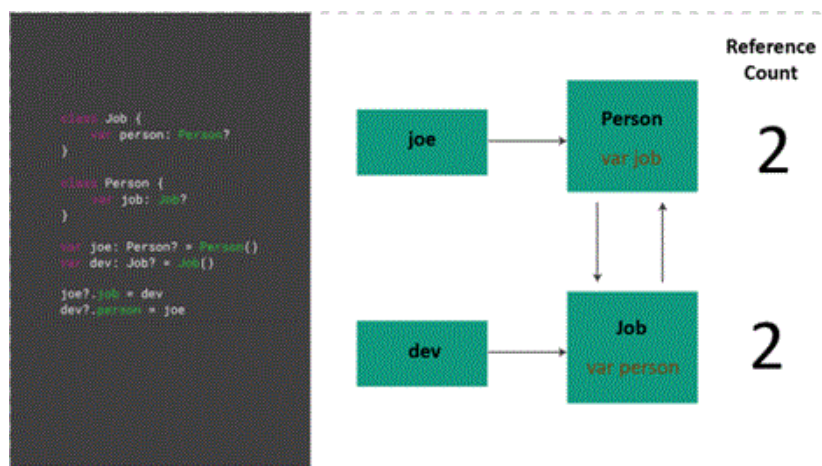


Figura 2. 12 Contador de referencias para las clases Job y Person Fuente: (Sharma, 2019)

En el caso de que las referencias sean seteadas a nil.

joe=nil
dev= nil

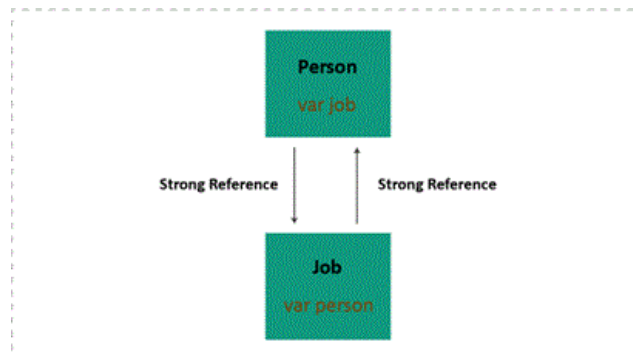


Figura 2. 13 Referencias fuertes entre las clases Job y Person Fuente: (Sharma, 2019)

En este momento no hay una eliminación de la asignación y las clases Job y Person no pueden ser removidas de la memoria, porque las variables de job y person apuntan al hacia ellos. Esto es llamado ciclo de retención. Para solucionar este problema se utiliza referencias débiles o sin propietario ya que estas no incrementan el ciclo conteo (Sharma, 2019).

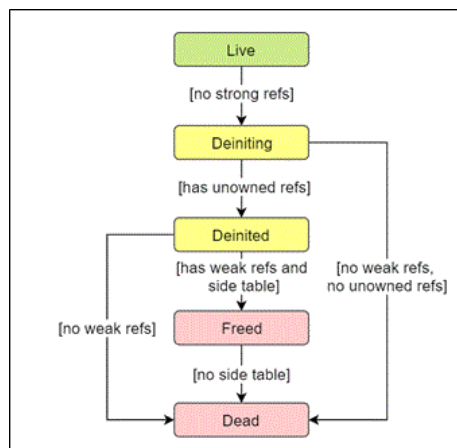


Figura 2. 14 Ciclo ARC
Fuente: (Bulavin, 2019)

2.3.3.3.1. Referencias débiles

Las referencias débiles, hijos pueden o no existir si el padre es removido de la memoria. Son opcionales y pueden ser inicializados en **nil** (Sharma, 2019).

2.3.3.3.2. Referencias sin propietario

En este caso el hijo existe en todo momento, pero es removido en el momento en que el padre es removido también, no puede ser inicializado a **nil** (Sharma, 2019).

2.4 Bases de datos

Las bases de datos son colecciones organizadas de información estructurada, o datos que normalmente se almacenan electrónicamente en un sistema informático y suelen estar controladas por un sistema de gestión de bases de datos o DBMS (Oracle, 2021).

2.4.1. Gestor de base datos

Una base de datos utiliza por lo general un software completo que permite la administración de la base de datos llamado DBMS. Un DBMS sirve de

interfaz entre programas o usuarios finales, lo que les permite a los mismos actualizar, recuperar y administrar como es organizada y optimizada la información. Los gestores de bases de datos también facilitan la supervisión y control de las bases de datos, lo que permite operaciones administrativas como seguimiento de rendimiento, ajustes, copias de seguridad y recuperación (Oracle, 2021).

2.4.2. Tipos o modelos de base de datos

Según la estructura que se utilices, existen diversos modelos de bases de datos en el mercado. Se clasifican en: Relacionales, NOSQL, En la nube, en columnas, orientadas a objetos, jerárquicas, de grafos, etc. A continuación, las principales utilizadas para el desarrollo de aplicaciones móviles (Matillion, 2020).

2.4.2.1. Base de datos relacionales

Los datos se almacenan en varias tablas que se encuentran relacionadas. Dentro de las tablas los datos se organizan en filas y columnas. El lenguaje de consultas estructurados SQL es el más utilizado por las bases de datos en general. Las bases de datos son muy confiables y cumplen con el estándar ACID (atomicidad, consistencia, aislamiento, durabilidad). Funciona bien con datos estructurados (Matillion, 2020).

2.4.2.2. NO SQL

Este tipo de base de datos no utilizan SQL como idioma principal para acceder a los datos. También son denominadas bases de datos no relacionales. A diferencia del otro tipo de base de datos, estas no deben ajustarse a un esquema específico, por lo que son útiles para empresas que deseen almacenar datos no estructurados o semiestructurados. Una de sus principales ventajas es que permite que se puedan realizar cambios sobre la marcha en la base de datos (Matillion, 2020).

2.4.2.3. Orientadas a objetos

En este tipo de base los datos son representados como objetos y clases. Los objetos son representantes de elementos del mundo real, con

atributos, mientras que las clases son grupos de objetos. Las bases de datos de objetos son de tipo relacional (Matillion, 2020).

2.4.3. SQL (Structure Query Language)

SQL es un lenguaje de programación utilizado por casi la totalidad de las bases de datos relacionales que sirve para consultar, definir, manipular datos y proporcionar control de acceso a la base. Fue desarrollado en la década de 1970 y se rige por la implementación del estándar SQL ANSI (Oracle, 2021).

2.5. Aplicaciones web

Las aplicaciones web funcionan con un esquema cliente-servidor. La comunicación empieza de lado del cliente a través de un navegador que es el encargado de visualizar e interpretar la información que es enviada por el servidor. El servidor siempre está en un proceso de escucha de peticiones de nuevos clientes para solicitar la información solicitada. Esto se realiza por medio de protocolos de comunicación (Gomez, 2016)

HTTP. – Es el protocolo que se utiliza para el intercambio de hipertexto y su definición esta especificada en RFC7230 (Request for comments) perteneciente al IETF (Internet Engineering Task Force).

HTTPS. – El protocolo HTTPS es una ampliación del protocolo HTTP que agrega la encriptación a la comunicación entre cliente y servidor. Es ideal para la seguridad y privacidad de los datos.

2.6. Servidor web

Un servidor web es aquel componente conformado por software y hardware que tiene la funcionalidad de atender peticiones tipo HTTP, por medio de la escucha de algunos puertos determinados, siendo el más común el puerto 80. Son utilizados en modelos cliente/servidor, donde el cliente origina las peticiones HTTP al utilizar un navegador mientras escribe la URL a la que se quiere dirigir, a su vez obtendrá respuesta de

algún servidor de la nube que este asociado a esa URL mediante una IP y reciba una página web (Cedeño, 2016).

2.7. PHP

PHP son las siglas para preprocesador de hipertexto, y es un lenguaje de scripting de código abierto de propósito general que es utilizado para el desarrollo web y puede ser utilizado en conjunto con HTML. En lugar de escribir muchas líneas de código HTML, el lenguaje PHP posee instrucciones especiales que facilitan la generación del HTML (PHP, 2021).

```
<!DOCTYPE html>
<html>
  <head>
    <title>Example</title>
  </head>
  <body>

    <?php
      echo "Hi, I'm a PHP script!";
    ?>

  </body>
</html>
```

Figura 2. 15 Ejemplo de PHP
Fuente: (PHP, 2021).

2.8. API

Una interfaz de programación de aplicaciones denominada API es una interfaz proporcionada por el sistema operativo o programas con licencia que permite que lenguajes de alto nivel utilicen del sistema operativo o del software licenciado en sus programas (IBM, 2013).

- Algunas de las API proporcionan las mismas funciones que el lenguaje de control (CL).
- Proporcionan funciones que los comandos del lenguaje de control no ofrecen.
- La mayoría de las API funcionan rápidamente y utilizan menos recursos del sistema.

2.8.1. Servicios Web

Son un grupo de estándares y protocolos que ayudan al intercambio de información entre aplicaciones, adicional a ello se pueden utilizar también

otras aplicaciones que hayan sido desarrolladas en lenguajes de programación alternativos y que pueden ser ejecutadas en cualquier plataforma. Debido a sus estándares abiertos se puede operar con varios servicios web y se tiene una mejor interoperabilidad gracias al WS-I (Web Services InterOperability) quienes están encargados de desarrollar varios perfiles que ayudan a determinar los estándares (Suero, 2017).

2.8.1. Métodos HTTP

El protocolo HTTP se compone de varias partes: La URL a la que va dirigida la petición, el verbo utilizado, diferentes encabezados, códigos de estado y el cuerpo de las respuestas recibidas. Los verbos HTTP denominados GET y POST permiten enviar peticiones al servidor junto con la URL para indicarle que hacer (Mitchell, 2016).

- GET: Pide alguna representación de un recurso específico.
- POST: Envía información a un recurso específico.
- PUT: Hace una sustitución de las representaciones existentes del recurso al cual va destinado por la carga útil de la petición.
- DELETE: Borra un recurso específico.
- CONNECT: Se establece un túnel a través de la red a un servidor especificado.
- OPTIONS: Describe las opciones de especificación de los recursos.

Capítulo 3. Desarrollo del trabajo de titulación

3.1. Requerimientos del sistema

El objetivo principal es implementar un sistema prototipo de administración del inventario mediante el uso de etiquetas NFC y una aplicación web basada en IOS. El proyecto cuenta con los siguientes elementos fundamentales: Un servidor web, una aplicación móvil que será la encargada de leer y escribir los datos en las etiquetas NFC, una base de datos y por último una aplicación web para poder controlar y administrar los datos.

En la base de datos serán agregados los siguientes campos que son tomados de cada bien ingresado: nombre, código NFC, categoría, serial, código de inventario, administrador, ubicación, fecha de ingreso, fecha de adquisición, fecha última revisión, fecha garantías, marca, estado, características, observaciones.

En la tarjeta NFC la cantidad de espacio es limitado, por lo que la cantidad de campos que se pueden escribir en la misma deben ser específicos. Los campos para utilizar son: Administrador, código NFC, nombre, código del inventario, última fecha de mantenimiento, serial o numeración.

El prototipo debe cumplir con las siguientes características:

- Inicio de sesión en el sitio web, así como en la aplicación móvil.
- Escritura de la etiqueta NFC por medio del teléfono móvil haciendo uso de la aplicación donde se agregarán los identificadores correspondientes.
- Lectura de las etiquetas NFC mediante la aplicación móvil que obtendrá los datos de la base de datos para ser visualizados en la aplicación.
- Consultar la totalidad del inventario tanto en la base de datos como en la aplicación.
- Administración del sistema web para facilidades en el manejo del inventario.
- Creación de registros tanto para usuarios como para el inventario.
- Uso del servidor web para obtener la información por medio del aplicativo web y móvil.
- El usuario podrá realizar operaciones tanto con el sistema web como la aplicación móvil después de haber ingresado al sistema con su respectiva contraseña. En el sistema móvil el usuario podrá escribir, leer

etiquetas y visualizar el inventario. En el sistema web se podrá consultar el inventario, así como su eliminación y edición en general.

3.2. Arquitectura de la aplicación

Para poder visualizar mejor el funcionamiento de la aplicación en la figura 3.1 se puede observar cómo se inicia el proceso de conexión desde el usuario hasta la base de datos.

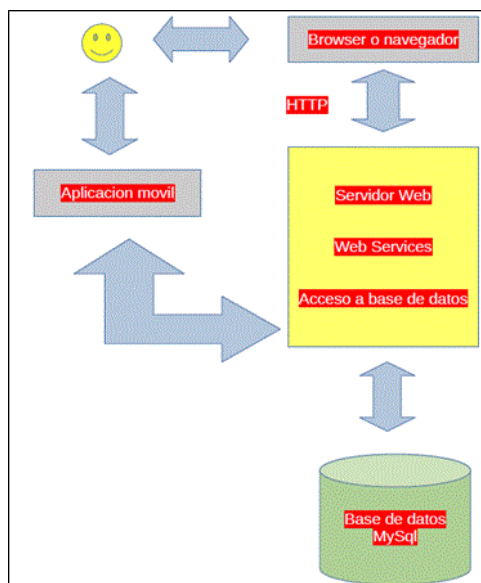


Figura 3. 1 Diagrama de despliegue de la aplicación

Fuente: Autor.

El sistema se basa en una aplicación móvil que es la encargada de leer y escribir los datos asociados a los bienes del inventario. Las etiquetas NFC son escritas en primer lugar con la aplicación donde por medio de un web service los datos son escritos en la base de datos.

El segundo sistema de visualización es el del sitio web, el cual permite una edición de los datos guardados para darles más nivel de detalle. Es necesario un usuario y contraseña para poder realizar las acciones.

3.3. Funcionamiento de la aplicación

Se detalla a continuación los protocolos y tecnologías utilizadas para el funcionamiento de la aplicación.

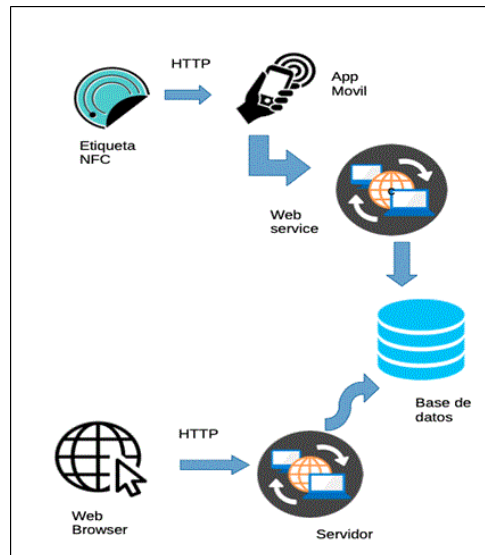


Figura 3. 2 Diagrama de funcionamiento de la aplicación

Fuente: Autor.

3.3.1. Etiquetas NFC

Para el proyecto se seleccionó las etiquetas tipo NTAG215 de la marca THONSEN. Tienen una compatibilidad universal con teléfonos móviles inteligentes y es donde será escrita la información de los bienes. Están estandarizadas con la norma ISO 14443-3A y son definidas según el foro NFC como etiquetas tipo 2, tiene una duración de 10 años, se pueden leer y escribir hasta 100000 veces y no admiten la encriptación.

Tabla 3. 1 Característica de las etiquetas NTAG25

CARACTERISTICAS	NTAG25
Memoria Total	540 Bytes
Memoria Disponible	504 Bytes
Longitud del Texto	490 Caracteres
Retención de datos	10 años
Resistencia a lectura/escritura	1000000 Ciclos
Encriptación	No
Número de serie (UID)	7 bytes Asignados
Compatibilidad	Compatible con estándar ISO 14443-3A

Fuente: Autor.

Cada característica es descrita a continuación:

Memoria total: Es la memoria total que no puede ser modificada y es dedicada para UID.

Memoria disponible: Total de memoria que si puede ser utilizada.

Longitud de texto: Caracteres máximos que pueden ser escritos en las etiquetas NFC.

Encriptación: Funciones de cifrado dentro del chip que evita la modificación de los datos.

Número de serie: Numero único asociado en las etiquetas NFC.

Compatibilidad Universal: El modelo de etiqueta utilizado fue desarrollado por NXP Semiconductores como un circuito integrado y están desarrolladas para cumplir con las especificaciones ISO/IEC14443 tipo A.

3.3.2. Aplicación web

Se utiliza para el desarrollo web la aplicación Visual Studio Code, el cual es un software de código abierto libre que permite la integración de varios lenguajes de programación y sirve como un IDE (Integrated Development Enviroment) universal. La estructura de la aplicación utilizará HTML, CSS, javascript con jquery para poder interactuar rápidamente con el usuario, además de contar con librerías que ayuden al entorno gráfico pertenecientes a Boostrap que ayuda a adaptar las interfaces visuales a las pantallas de cualquier dispositivo mediante el uso de HTML5. Para el lenguaje del servidor se utilizará PHP.

3.3.3. Aplicación móvil

En cuanto al desarrollo de la aplicación móvil se utiliza software nativo propio de Apple que permite el uso de la tecnología NFC, ya que, aunque se pueden hacer aplicaciones en editores de código de terceros, es necesario equipos de la marca Apple para poder trabajar en ellos y se requiere adicionalmente una licencia de desarrollador. Se utiliza lenguaje Swift para la programación y el IDE es Xcode.

3.3.4. Base de Datos

Para el desarrollo de la base de datos se utiliza MySQL en su variante de código abierto llamada MariaDB. El lenguaje que se utilizará en la misma es SQL (Structure Query Language) el cual es específico de estos sistemas.

3.3.5. Servidor web

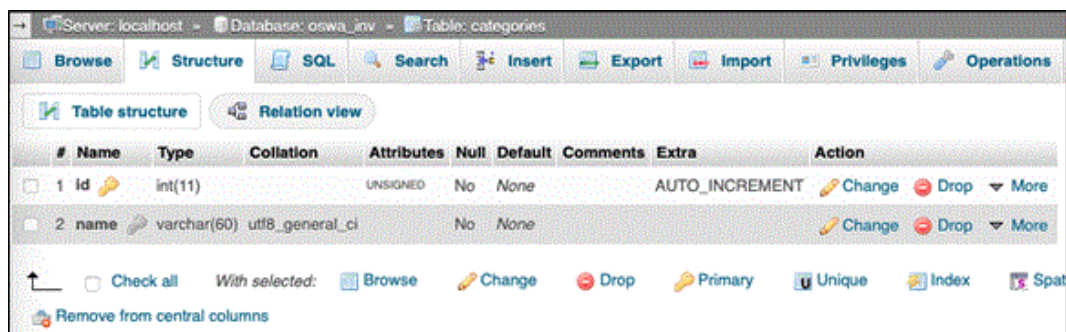
Se utiliza un servidor web de un servicio de hosting el cual proporciona la IP para acceder al aplicativo, el servidor utiliza sistema operativo Linux y una base de datos MySQL. Se trabaja con la herramienta administrativa C-Panel.

3.4. Estructura de la base de datos

La base de datos utiliza las siguientes tablas con las siguientes columnas donde almacenar los datos tanto de usuarios como de bienes muebles:

- Users: id, name, username, password, user_level, image, status, last_login.
- User_groups: id, group_name, group level, group_status.
- Categories: id, name.
- Locations: id, name.
- Brand: id, name.
- Products: id, name, quantity, buy_price, sale_price, categorie_id, media_id, date, serial_code, tag_id, administrador, purchase_date, product_status, features, last_maintenance, location_id, brand_id

La estructura de la base está representada de la siguiente manera:



#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	id	int(11)		UNSIGNED	No	None		AUTO_INCREMENT	Change Drop More
2	name	varchar(60)	utf8_general_ci		No	None			Change Drop More

Figura 3. 3 Estructura de categorías.

Fuente: Autor.

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	id	int(11)		UNSIGNED	No	None		AUTO_INCREMENT	Change Drop More
2	name	varchar(255)	utf8_general_ci		No	None			Change Drop More
3	quantity	varchar(50)	utf8_general_ci		Yes	NULL			Change Drop More
4	buy_price	decimal(25,2)			Yes	0.00			Change Drop More
5	sale_price	decimal(25,2)			No	0.00			Change Drop More
6	categorie_id	int(11)		UNSIGNED	No	None			Change Drop More
7	media_id	int(11)			Yes	0			Change Drop More
8	date	datetime			No	None			Change Drop More
9	serial_code	varchar(50)	utf8_general_ci		Yes	NULL			Change Drop More
10	tag_id	varchar(50)	utf8_general_ci		Yes	NULL			Change Drop More
11	administrador	varchar(50)	utf8_general_ci		Yes	NULL			Change Drop More
12	purchase_date	datetime			Yes	NULL			Change Drop More
13	product_status	varchar(50)	utf8_general_ci		Yes	NULL			Change Drop More
14	features	varchar(50)	utf8_general_ci		Yes	NULL			Change Drop More
15	last_maintenance	datetime			Yes	NULL			Change Drop More
16	location_id	int(11)		UNSIGNED	No	None			Change Drop More
17	brand_id	int(11)		UNSIGNED	No	None			Change Drop More

Figura 3. 4 Estructura de Productos.

Fuente: Autor.

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	id	int(11)		UNSIGNED	No	None		AUTO_INCREMENT	Change Drop More
2	name	varchar(60)	latin1_swedish_ci		No	None			Change Drop More
3	username	varchar(50)	latin1_swedish_ci		No	None			Change Drop More
4	password	varchar(255)	latin1_swedish_ci		No	None			Change Drop More
5	user_level	int(11)			No	None			Change Drop More
6	image	varchar(255)	latin1_swedish_ci		Yes	no_image.jpg			Change Drop More
7	status	int(1)			No	None			Change Drop More
8	last_login	datetime			Yes	NULL			Change Drop More

Figura 3. 5 Estructura de Usuarios.

Fuente: Autor.

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	id	int(11)			No	None		AUTO_INCREMENT	Change Drop More
2	group_name	varchar(150)	latin1_swedish_ci		No	None			Change Drop More
3	group_level	int(11)			No	None			Change Drop More
4	group_status	int(1)			No	None			Change Drop More

Figura 3. 6 Estructura de Grupos de Usuarios

Fuente: Autor.

3.5. Diseño e implementación

Para poder operar los registros guardados de los diferentes bienes, se utiliza la aplicación web que permite que el usuario final sin necesidades de conocimiento de programación y lenguaje SQL pueda consultar la

información de cada bien guardado. La ventaja de usar una aplicación web es su compatibilidad con cualquier tipo de sistema operativo ya que solo se necesitaría un navegador web para acceder al mismo.

Los archivos que conforman la aplicación web están desarrollados en PHP y para cada interfaz a la que va a tener acceso el usuario se llama al archivo load.php el cual contiene llamados a otros archivos necesarios para la funcionalidad de las interfaces. Estos son los siguientes:

- Config.php: Información de conexión a la base de datos donde se incluye usuario, password, servidor, base de datos.
- Database.php: Aquí se colocan los procedimientos para abrir y cerrar las conexiones con la base de datos.
- Sql.php: funciones asociadas a las operaciones de la base de datos como agregar, eliminar y actualizar registros.
- Functions.php: Contiene los procedimientos utilizados por la aplicación como validaciones y comportamientos del sitio web.

3.5.1. Pantalla de login

Se accede mediante el uso de usuario y contraseña:

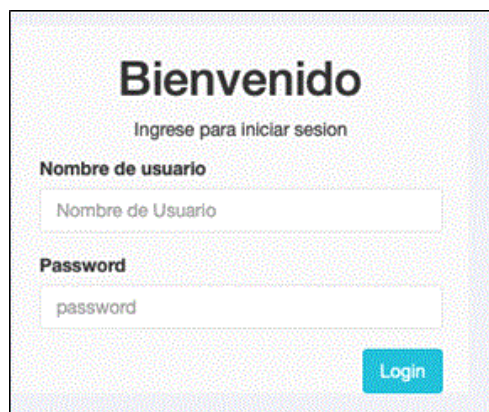
The image shows a login interface with a light blue background. At the top, the word "Bienvenido" is written in a large, bold, black font. Below it, the text "Ingrese para iniciar sesion" is centered in a smaller font. There are two input fields: the first is labeled "Nombre de usuario" and contains the text "Nombre de Usuario"; the second is labeled "Password" and contains the text "password". A blue button with the word "Login" in white text is positioned at the bottom right of the form area.

Figura 3. 7 Pantalla de inicio de la aplicación web.

Fuente: Autor.

Al iniciar la aplicación llama a la página principal que es index.php la cual contiene el método load.php donde una vez colocado el usuario y contraseña correspondiente se procede a la validación de los datos por medio del proceso de autenticación de credenciales. Mediante el método

POST se envían los datos a la URL que se incluye mediante load la cual procesa la petición con el método de la página auth.php.

```

<?php include_once('includes/load.php'); ?>
<?php
$req_fields = array('username','password' );
validate_fields($req_fields);
$username = remove_junk($_POST['username']);
$password = remove_junk($_POST['password']);

if(empty($errors)){
  $user_id = authenticate($username, $password);
  if($user_id){
    //crear sesion con id
    $session->login($user_id);
    //Actualizar logoneo
    updateLastLogIn($user_id);
    $session->msg("s", "Bienvenido al sistema de inventarios.");
    redirect('home.php',false);
  } else {
    $session->msg("d", "Password o contraseña incorrectos.");
    redirect('index.php',false);
  }
} else {
  $session->msg("d", $errors);
  redirect('index.php',false);
}
?>

```

Figura 3. 8 Código de inicio de sesión.
Fuente: Autor.

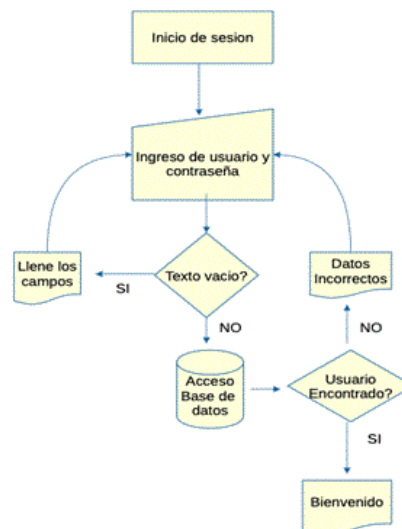


Figura 3. 9 Diagrama de flujo de inicio de sesión.
Fuente: Autor.

3.5.2. Pantalla de inicio

Después de iniciar sesión, la pantalla de inicio es donde visualizar las opciones administrativas.



Figura 3. 10 Pantalla de inicio.
Fuente: Autor.

3.5.3. Interfaz de inicio

Después de iniciar sesión, la pantalla de inicio es donde visualizar las opciones administrativas. Aquí se pueden visualizar la cantidad de usuarios y bienes añadidos de manera resumida. La función count_by es utilizada para saber la cantidad de usuarios y productos que hay ingresados.

```

54  /*-----*/
55  /* Function for Count id By table name
56  /*-----*/
57
58  function count_by_id($table){
59      global $db;
60      if(tableExists($table))
61      {
62          $sql = "SELECT COUNT(id) AS total FROM ".$db->escape($table);
63          $result = $db->query($sql);
64          return($db->fetch_assoc($result));
65      }
66  }

```

Figura 3. 11 Interfaz de inicio.
Fuente: Autor.



Figura 3. 12 Interfaz de inicio.
Fuente: Autor.

3.5.4. Usuarios – Administración de grupos

En esta opción se puede editar los permisos de los usuarios para otorgarles diferentes niveles de acceso y solo puede ser modificada por usuarios administradores.

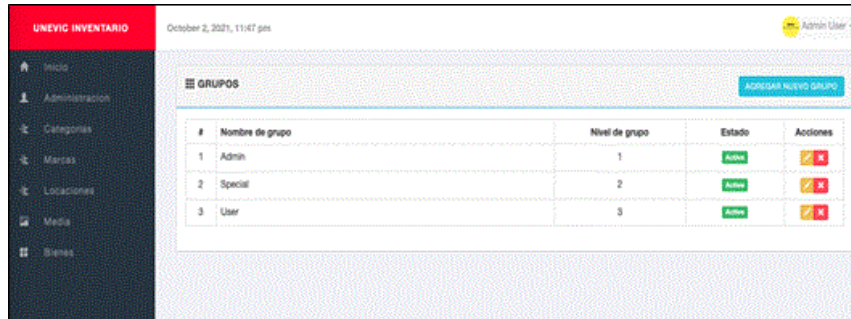


Figura 3. 13 Interfaz de Administración de grupos

Fuente: Autor.

El proceso para añadir un grupo nuevo es el siguiente:

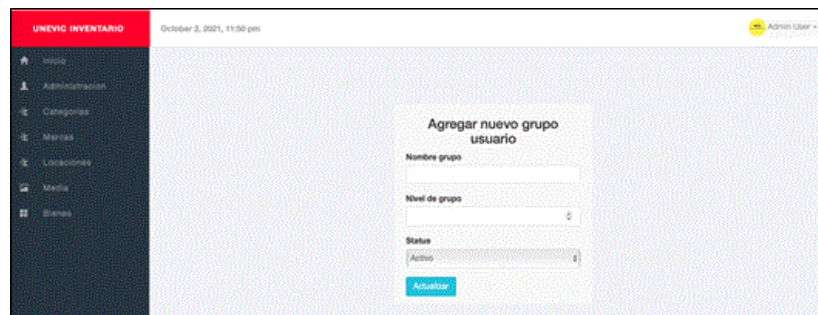


Figura 3. 14 Agregar nuevos grupos de usuarios

Fuente: Autor.

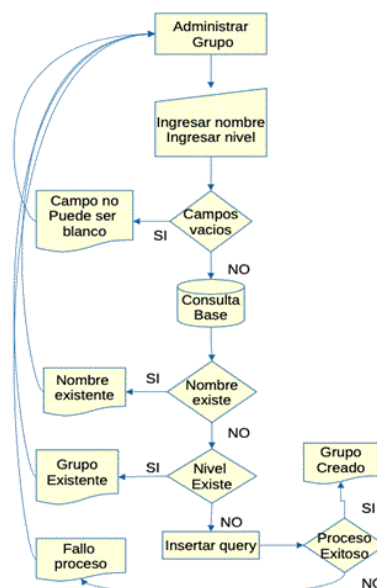


Figura 3. 15 Diagrama de creación de grupo.

Fuente: Autor.

Una vez comprobado que los datos ingresados son correctos, por medio de una función que contiene una sentencia SQL se añade el registro a la tabla correspondiente. Los campos son enviados desde la página web utilizando un elemento de tipo formulario FORM y utilizando el método POST, de la página add_group.php.

```
<?php
if(isset($_POST['add'])){

$req_fields = array('group-name','group-level');
validate_fields($req_fields);

if(find_by_groupName($_POST['group-name']) == false ){
    $session->msg('d','<b>Lo siento!</b> Nombre duplicado en la base de d
    redirect('add_group.php', false);
}elseif(find_by_groupLevel($_POST['group-level']) == false) {
    $session->msg('d','<b>Lo siento!</b> Nivel de grupo duplicado en la b
    redirect('add_group.php', false);
}
}
if(empty($errors)){
    if($name = remove_junk($db->escape($_POST['group-name']));
    $level = remove_junk($db->escape($_POST['group-level']));
    $status = remove_junk($db->escape($_POST['status']));

    $query = "INSERT INTO user_groups (";
    $query .= "group_name,group_level,group_status";
    $query .= ") VALUES (";
    $query .= " '{$name}', '{$level}','{$status}'";
    $query .= ")";
    if($db->query($query)){
        //sucess
        $session->msg('s',"Grupo ha sido creado! ");
        redirect('add_group.php', false);
    } else {
        //failed
    }
}
}
```

Figura 3. 16 Código fuente de creación de grupo.
Fuente: Autor.

3.5.5. Administración de usuarios

Aquí se edita los usuarios que tienen acceso al sistema, así como también editar o eliminar los usuarios ya creados siempre y cuando tengan permisos de administrador.

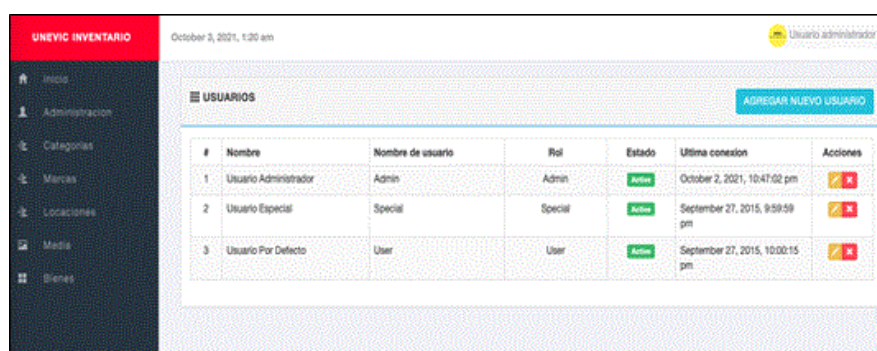


Figura 3. 17 Pantalla de administración de Usuarios.
Fuente: Autor.

Cuando se elige agregar un usuario se pasa a la siguiente página donde se pide los datos como nombre, id, contraseña, estado y rol.

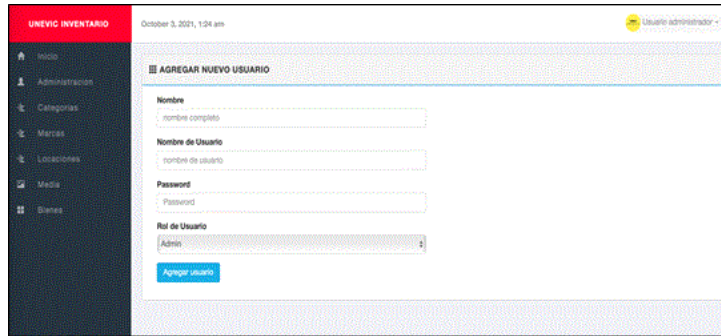


Figura 3. 18 Pantalla para agregar un usuario nuevo.

Fuente: Autor.

El funcionamiento para agregar un nuevo usuario se puede apreciar en el siguiente diagrama.

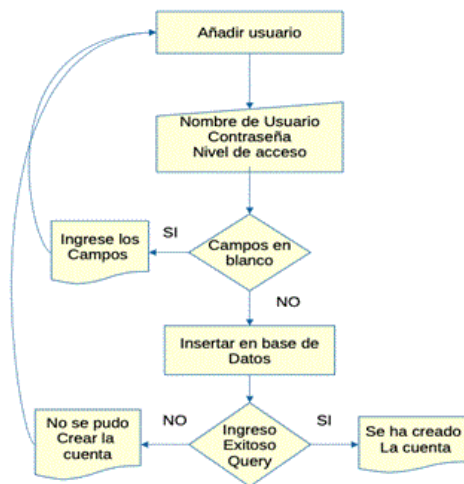


Figura 3. 19 Diagrama agregar un usuario nuevo.

Fuente: Autor.

Por medio del método POST los valores del formulario son enviados a add_user.php, donde son agregados a la base de datos.

```

?php
if(isset($_POST['add_user'])){
    $req_fields = array('full-name','username','password','level' );
    validate_fields($req_fields);

    if(empty($errors)){
        $sname = remove_junk($db->escape($_POST['full-name']));
        $susername = remove_junk($db->escape($_POST['username']));
        $spassword = remove_junk($db->escape($_POST['password']));
        $suser_level = (int)$db->escape($_POST['level']);
        $spassword = sha1($spassword);
        $squery = "INSERT INTO users (";
        $squery .= "name,username,password,user_level,status";
        $squery .= ") VALUES (";
        $squery .= "'{$sname}', '{$susername}', '{$spassword}', '{$suser_level}',";
        $squery .= ")";
        if($db->query($squery){
            //success
            $session->msg('s','Cuenta ha sido creada! ');
            redirect('add_user.php', false);
        } else {
            //failed
            $session->msg('d',' No se pudo crear la cuenta!');
            redirect('add_user.php', false);
        }
    } else {

```

Figura 3. 20 Código fuente de proceso para agregar usuarios.

Fuente: Autor.

3.5.6. Agregar locación

En esta pantalla se podrá agregar las ubicaciones físicas donde pueden ir los productos. Para su correcto almacenamiento se debe indicar las bodegas donde serán depositadas.

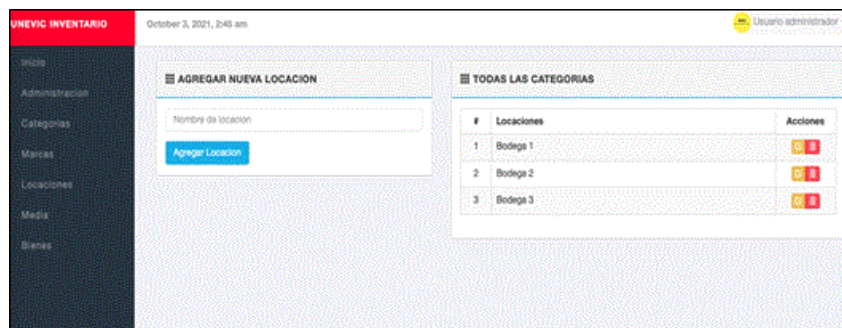


Figura 3. 21 Pantalla de locaciones.

Fuente: Autor.

En la pantalla anterior se puede agregar locaciones nuevas como editar las locaciones ya existentes. A continuación, la pantalla para modificar una locación existente.

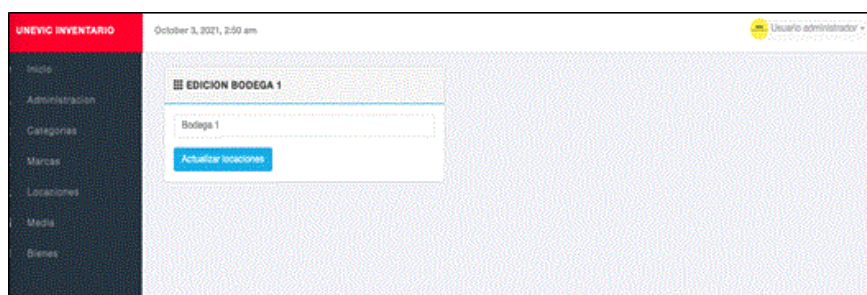


Figura 3. 22 Pantalla editar locaciones.

Fuente: Autor.

3.5.7. Agregar categoría

Aquí se agrega el tipo de categoría a la que pertenece el bien o producto que se va a almacenar en la bodega.

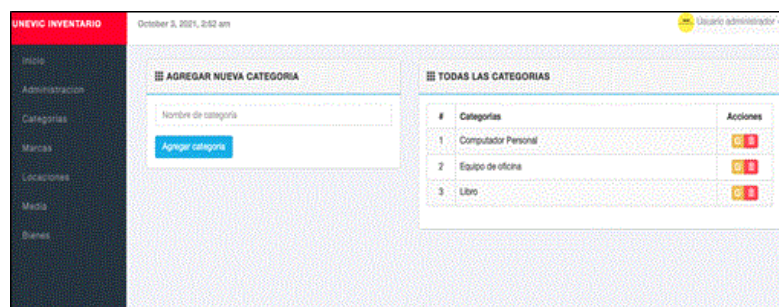


Figura 3. 23 Pantalla editar categorías.

Fuente: Autor.

Se edita la categoría en el siguiente formulario:

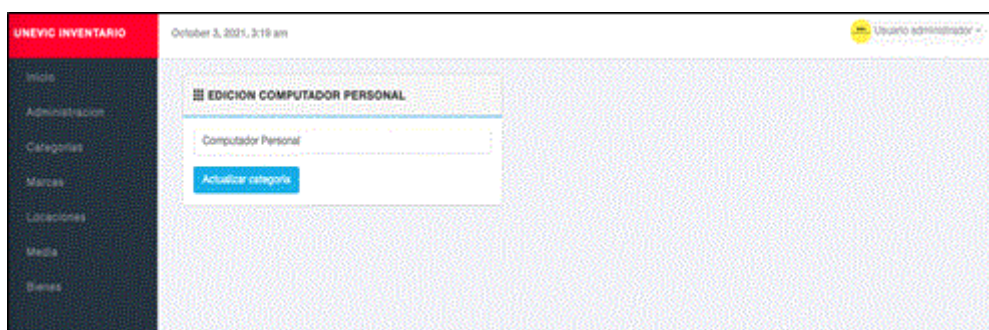


Figura 3. 24 Pantalla actualizar categorías.

Fuente: Autor.

3.5.8. Agregar Marca

Aquí se agrega el tipo de categoría a la que pertenece el bien o producto que se va a almacenar en la bodega.

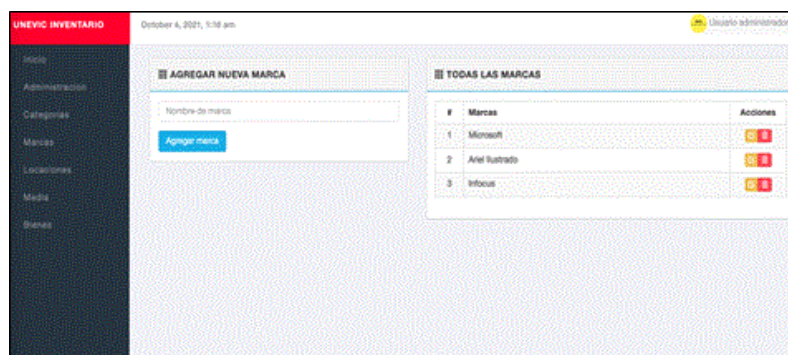


Figura 3. 25 Pantalla agregar marcas.

Fuente: Autor.

Para la edición de las marcas se tiene la siguiente pantalla.

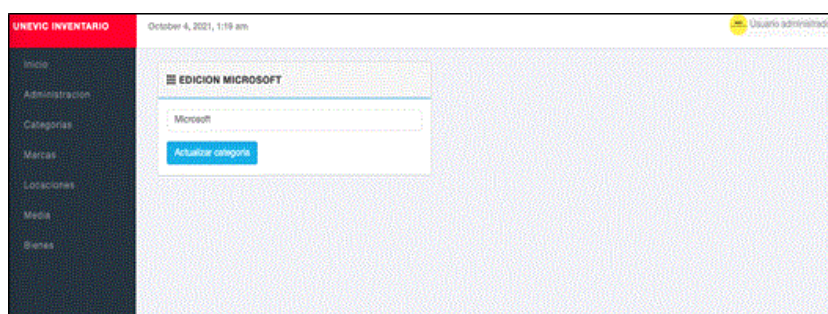


Figura 3. 26 Pantalla editar marcas

Fuente: Autor.

3.5.9. Resumen de Bienes

Aquí se agrega el tipo de categoría a la que pertenece el bien o producto que se va a almacenar en la bodega. En esta parte de la interfaz se visualizan los bienes que ya han sido agregados en la base de datos. La información compartida es de fechas de ingreso, de compra, valores, locación, marca, administrador, cantidad, etc.

#	Foto	Título de la foto	Categoría	Cantidad	Precio	Precio de venta	Bien agregado	Fecha Compra	Fecha Mantenimiento	Encargado	Serial	Locacion	Marca	Acciones
1		HP 4055	Computador Personal	1	400.00	0.00	October 1, 2021, 12:00:00 am	2021-10-01 00:00:00	2021-10-01 00:00:00	Rafael Castro	454545	Bodega 1	Microsoft	
2		Don Quijote	Libro	1	50.00	0.00	October 1, 2021, 12:00:00 am	2021-10-01 00:00:00	2021-10-01 00:00:00	Rafael Castro	45555	Bodega 1	Ariel suizado	
3		Proyector	equipo de oficina	1	900.00	0.00	October 1, 2021, 12:00:00 am	2021-10-01 00:00:00	2021-10-01 00:00:00	Rafael Castro	515155	Bodega 3	Infocus	

Figura 3. 27 Pantalla resumen de inventario.

Fuente: Autor.

3.5.10. Agregar bienes

En esta pantalla se tiene la opción de agregar nuevos bienes, agregando tanto su locación, tipo de categoría, marca, características y estado. Aquí se pueden agregar bienes que no se les puede agregar una etiqueta NFC por cualquier motivo y que deben ser tomados en cuenta en el inventario.

Figura 3. 28 Pantalla de agregar nuevo inventario.

Fuente: Autor.

Cuando se oprime el botón agregar del formulario los datos se procesan de la siguiente manera. Figura 3.29.

El código fuente para agregar inventario nuevo se muestra a continuación con el método post donde se envían los datos del formulario a la URL `add_product.php`. Figura 3.30.

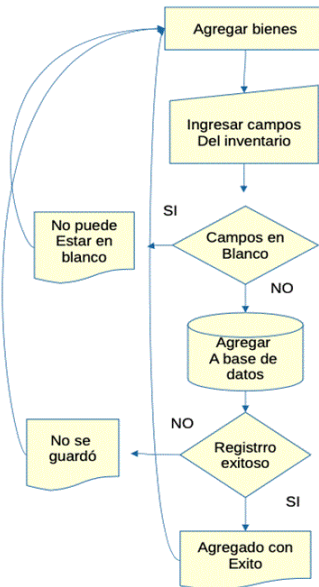


Figura 3. 29 Diagrama de agregar nuevo inventario.
Fuente: Autor.

```

if (empty($errors)) {
    $p_name = remove_junk($db->escape($_POST['product-title']));
    $p_admin = remove_junk($db->escape($_POST['product-administrator']));
    //$p_nfc= remove_junk($db->escape($_POST['product-nfccode']));
    $p_serial= remove_junk($db->escape($_POST['product-serial']));
    $p_status=remove_junk($db->escape($_POST['product-status']));
    $p_feature= remove_junk($db->escape($_POST['product-feature']));
    $p_location=remove_junk($db->escape($_POST['product-location']));
    $p_brand= remove_junk($db->escape($_POST['product-brand']));
    $p_cat = remove_junk($db->escape($_POST['product-categorie']));
    $p_qty = remove_junk($db->escape($_POST['product-quantity']));
    $p_buy = remove_junk($db->escape($_POST['buying-price']));
    $p_sale = '0';//remove_junk($db->escape($_POST['saleing-price']));
    $p_entrydate= read_date_parse(remove_junk($db->escape($_POST['entry-date']));
    $p_buyingdate= read_date_parse(remove_junk($db->escape($_POST['buying-dat]));
    $p_maintenancedate= read_date_parse(remove_junk($db->escape($_POST['maint

    if (is_null($_POST['product-photo']) || $_POST['product-photo'] == "") {
        $media_id = '0';
    } else {
        $media_id = remove_junk($db->escape($_POST['product-photo']));
    }
    $date = make_date();
    $query = "INSERT INTO products (";
    $query .= " name,quantity,buy_price,sale_price,categorie_id,media_id,date";
    $query .= ") VALUES (";
    $query .= " '{$p_name}', '{$p_qty}', '{$p_buy}', '{$p_sale}', '{$p_cat}',";
    $query .= " )";
    $query .= " ON DUPLICATE KEY UPDATE name='{$p_name}'";
    if ($db->query($query)) {

```

Figura 3. 30 Código fuente del código agregar bienes.
Fuente: Autor.

3.5.11. Perfiles de usuario

Por medio del sistema se pueden acceder a los perfiles de usuarios. Estas acciones estarán disponibles para los usuarios con permisos de administración. Por medio de la opción de perfiles se puede también salir del sistema.

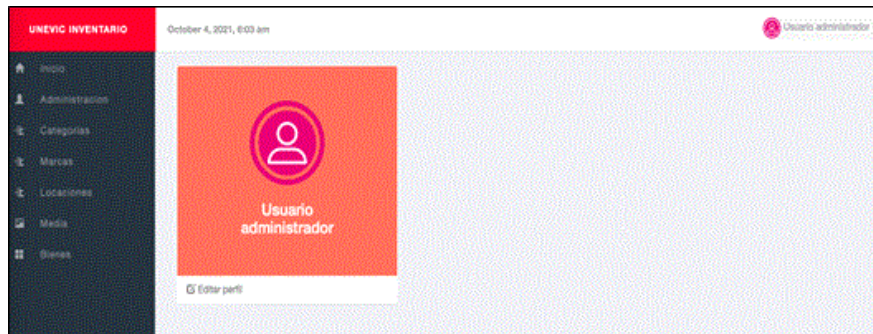


Figura 3. 31 Pantalla de edición de perfil.

Fuente: Autor.

Se puede ingresar en el proceso de configuración para editar las opciones del usuario y cambiar los datos asignados.

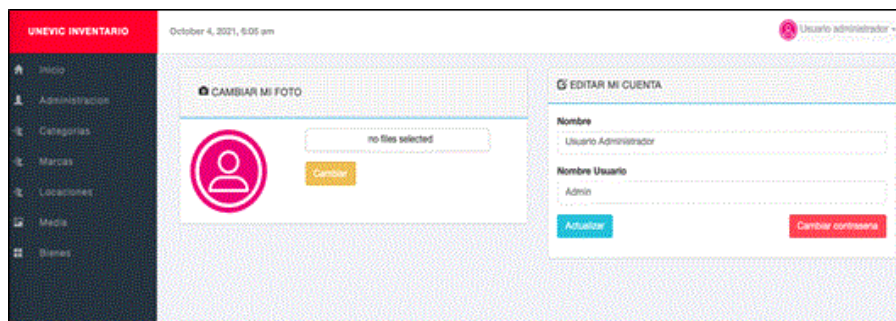


Figura 3. 32 Formulario de edición de perfil.

Fuente: Autor.

El método POST envía los datos del formulario a la página edit_account.php donde son procesados para actualizar la base de datos.

```

<?php
//update user other info
if(isset($_POST['update'])){
    $req_fields = array('name','username' );
    validate_fields($req_fields);
    if(empty($errors)){
        $id = (int)$SESSION['user_id'];
        $name = remove_junk($db->escape($_POST['name']));
        $username = remove_junk($db->escape($_POST['username']));
        $sql = "UPDATE users SET name = '{$name}', username = '{$username}'";
        $result = $db->query($sql);
        if($result && $db->affected_rows() === 1){
            $session->msg('s',"Account updated ");
            redirect('edit_account.php', false);
        } else {
            $session->msg('d',' Sorry failed to updated!');
            redirect('edit_account.php', false);
        }
    } else {
        $session->msg("d", $errors);
        redirect('edit_account.php',false);
    }
}
}

```

Figura 3. 33 Código fuente de edición de perfil.

Fuente: Autor.

3.6. Diseño de la aplicación móvil de IOS.

La aplicación móvil permite pasos sencillos para poder agregar los diferentes bienes al inventario. Consiste en un Login que permite al usuario registrado poder ingresar a la aplicación, luego permite tanto la escritura como la lectura de las etiquetas NFC y por último el almacenamiento del registro en la base de datos.

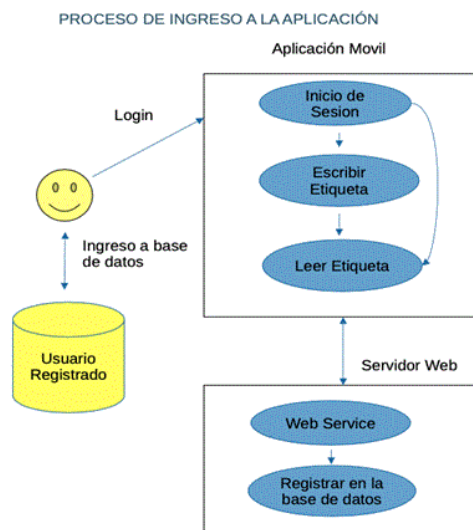


Figura 3. 34 Diagrama de casos de uso de la aplicación móvil.

Fuente: Autor.

La aplicación móvil tiene como finalidad la lectura y escritura de etiquetas NFC. Mediante la escritura se guardan los datos del bien en la etiqueta, a su vez, con la lectura se pueden obtener esos datos y guardarlos en la base de datos de productos. El proceso consta de 5 actividades que conforman la aplicación. El proyecto se encuentra estructurado de la siguiente manera en XCode, figura 3.35:

Dentro de la carpeta principal de MobileAPP, se cuenta con la sección Model que equivale a los modelos de datos que serán recibidos del servicio web. Recoge los datos que vienen codificados como JSON y permite descomprimirlos en Swift por medio del método JSONDecoder. APIS contiene todo lo relacionado a la lógica de la aplicación, que van desde conexiones hacia el servicio web, así como Codificadores y Decodificadores de los datos manipulados.

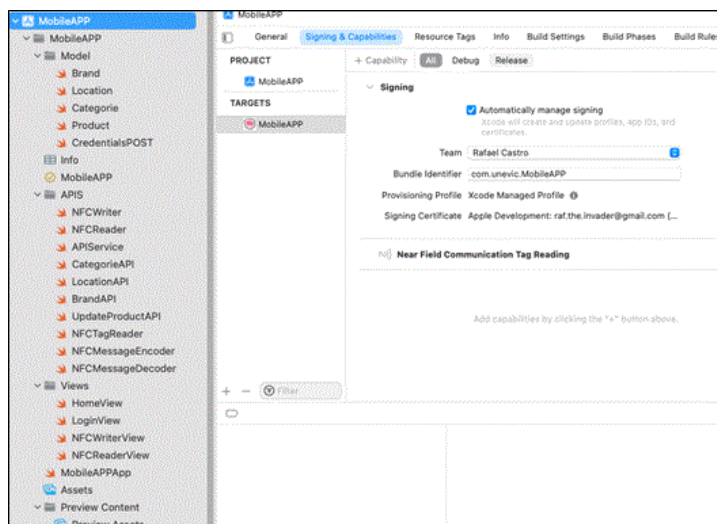


Figura 3. 35 Diagrama de casos de uso de la aplicación móvil.
Fuente: Autor.

Views es donde se almacena las vistas o pantallas que están al alcance del usuario, que las utilizará para manipular los datos.

3.6.1. Login de usuario.

En esta pantalla, el usuario debe ingresar las mismas credenciales que se encuentran en el sistema web. Al estar el usuario en la misma base de datos solo se necesita consumir un servicio web desarrollado en PHP que devuelve un objeto JSON el cual se utiliza para verificar la existencia del usuario registrado.

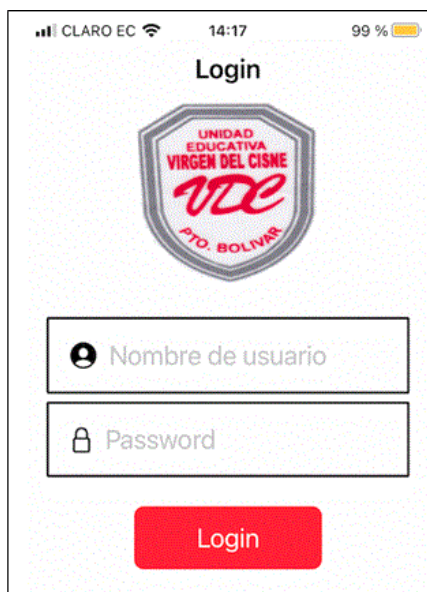


Figura 3. 36 Pantalla de login de la aplicación móvil.
Fuente: Autor.

Para poderse conectar interactuar con la base de datos la aplicación hace uso de un servicio web creado en PHP. El servicio recibe una petición desde la aplicación por medio de POST y devolverá un objeto JSON como respuesta conteniendo los datos necesarios para la autenticación.

```
<?php
$requestMethod = $_SERVER["REQUEST_METHOD"];
include('includes/Rest.php');
$api = new Rest();
switch ($requestMethod) {
    case 'POST':
        if($_POST['code']== '1'){
            $api->getUser($_POST);
        }
}
```

Figura 3. 37 Servicio Web de gestión.

Fuente: Autor.

El servicio web principal distingue el tipo de petición y a su vez un dato adicional como código de acción el cual utiliza la función específica de la clase Rest.php donde está implementada la consulta.

```
public function getUser($userData)
{
    $usernameP = $userData["username"];
    $passwordP = $userData["password"];
    $userQuery = $this->dbConnect->prepare("SELECT username, password, name, user_level FROM u
    $userQuery->bind_param('ss', $usernameP, $passwordP);
    $userQuery->execute();
    $result = $userQuery->get_result();
    if ($result->num_rows > 0) {
        $row = $result->fetch_array(MYSQLI_ASSOC);
        $userLog = $row['username'];
        $nameUser = $row['name'];
    } else {
        $userLog = "none";
        $nameUser = "none";
    }
    $userResponse = array(
        'username' => $userLog,
        'name' => $nameUser
    );
    header('Content-Type: application/json');
    echo json_encode($userResponse);
}
```

Figura 3. 38 Función para obtener usuario.

Fuente: Autor.

Para poder enviar la información a la base de datos por medio del método login() el cual es creado en una clase que tiene la propiedad de ser Observable para que de esta manera pueda ser visualizado por la vista

```
func login(){
    guard let requestUrl = URL( string: "http://192.168.1.4:8080/inventario/webservice.php") else{
        return
    }
    let parameters : [String:Any] = ["username":self.username,"password":self.password, "code": "1"]
    var request = URLRequest(url: requestUrl)

    var components = URLComponents()
    var queryItems = [URLQueryItem]()

    for (key, value) in parameters {
        let queryItem = URLQueryItem(name: key, value: String(describing: value))
        queryItems.append(queryItem)
    }
    components.queryItems = queryItems
    let queryItemData = components.query?.data(using: .utf8)
    request.httpBody = queryItemData
    request.httpMethod = "POST"

    request.setValue("application/x-www-form-urlencoded", forHTTPHeaderField: "Content-Type")
}
```

Figura 3. 39 Función login.

Fuente: Autor.

Después de haber obtenido los parámetros se debe agregarlos a la petición POST además de las cabeceras del mensaje que serán enviados hacia el servicio web. Se crea una sesión de tipo URLSession, que será la encargada de generar la petición (REQUEST)

```
request.httpBody = queryItemData
request.httpMethod = "POST"

request.setValue("application/x-www-form-urlencoded", forHTTPHeaderField: "Content-Type")

let session = URLSession.shared
let task = session.dataTask(with: request){ (data, response, error) in

    guard let data = data, error == nil else{

```

Figura 3. 40 Función login- creación de petición.

Fuente: Autor.

Al momento de obtener la respuesta se debe utilizar DispatchQueue.main.async para poder hacer cambios en la vista de manera asíncrona y así evitar retrasos por caso donde no se pueda obtener respuesta inmediatamente. La respuesta viene dada en un objeto JSON el cual se debe decodificar utilizando una estructura que será acorde a la respuesta esperada.

```
import Foundation

struct CredentialsPOST: Codable {

    var username : String = ""
    var name : String = ""
}
```

Figura 3. 41 Estructura de credencial.

Fuente: Autor.

Los datos están encapsulados por lo que es necesario realizar la decodificación por medio de JSONDecoder. La estructura implementa la API Codable la cual permite mucha de la interacción con el decodificador.

```
do{
let unwrappedData = try JSONDecoder().decode(CredentialsPOST.self, from: data)

DispatchQueue.main.async {

    if(unwrappedData.username != "none"){
        self.showProgressView = true
        self.name = unwrappedData.name
        self.getUserName = unwrappedData.username
        self.showAlert = false
    }
    else{
        self.showProgressView = false
        self.messageAlert = self.messageOption(option: "2")
        self.showAlert = true
    }
}
}
catch{
print(String(describing: error))
DispatchQueue.main.async {
    self.messageAlert = self.messageOption(option: "3")
    self.showAlert = true
}
}
```

Figura 3. 42 Función login-Decodificando data.

Fuente: Autor.

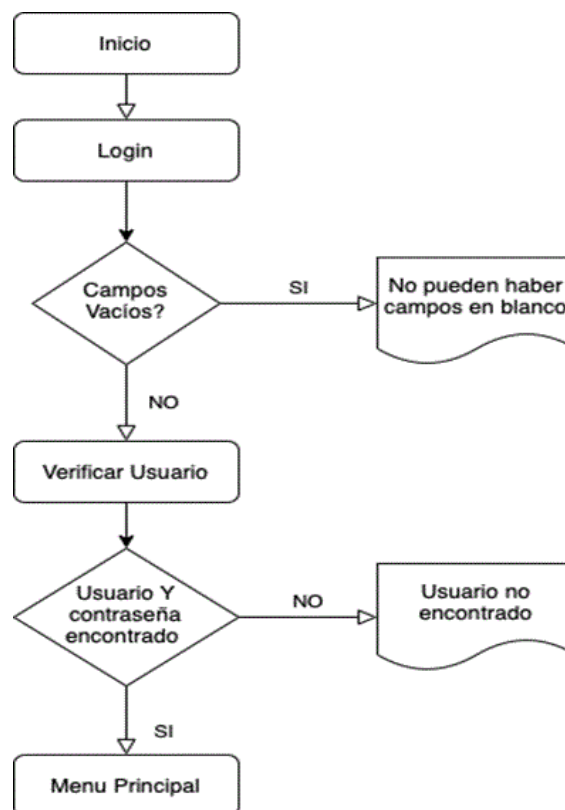


Figura 3. 43 Diagrama de flujo Login.

Fuente: Autor.

3.6.2. Menú Principal

Una vez que se ha accedido con las respectivas credenciales, entran al menú principal. Dentro del Menú se cuenta con las opciones de Lectura y Escritura de las etiquetas NFC.



Figura 3. 44 Menú principal aplicación web.

Fuente: Autor.

La interfaz presentada al ser solo un contenedor de las vistas relacionadas a Escritura y Lectura no cuenta como tal con un servicio web.

```
var body: some View {
    VStack{
        Picker("Menu Principal",selection: $selectOptions){
            ForEach(ChooseAction.allCases, id: \.self){
                Text($0.rawValue)
            }
        }
        .pickerStyle(SegmentedPickerStyle())
        .padding()
        Spacer()
        ChosenView(selectAction: selectOptions)
        Spacer()
    }
    .navigationBarTitle("Menu Principal")
    .navigationBarTitleDisplayMode(.inline)
    .navigationBarBackButtonHidden(true)

    .navigationBarItems(leading: Button(action: {
        self.presentation.wrappedValue.dismiss()
    }) {
        HStack {
            Image(systemName: "person.fill.badge.minus")
                .background(Color.red)
                .foregroundColor(Color.white)
            Text("Logout")
                .foregroundColor(Color.red)
        }
    })
}
```

Figura 3. 45 Código fuente vista de interfaz HomeView.

Fuente: Autor.

Las interfaces de escritura y lectura están contenidas dentro de la interfaz home por medio de un picker creado a partir de vistas o views.


```

enum ChooseAction : String, CaseIterable{
    case writer = "Escritor NFC"
    case reader = "Lector NFC"
}

struct ChosenView : View{
    var selectAction : ChooseAction
    var body: some View{

        switch selectAction {
        case .writer:
            //cambio de vistas

            NFCWriterView()
        case .reader:

            NFCReaderView()
        }
    }
}

```

Figura 3. 46 Código fuente selector de vistas dentro de Home.
Fuente: Autor.

3.6.3. Escritor NFC

Para realizar la escritura se deben llenar los campos que se encuentran en la vista y presionar el botón escribir.

Figura 3. 47 Formulario Escritura NFC.
Fuente: Autor.

Al cargar el formulario, se llama a 3 objetos JSON que vienen desde la base de datos por medio del servicio web y tienen la finalidad de llenar los selectores o pickers de categoría, marca y locación.

```

public function getLocation(){
    $locationQuery = "SELECT * FROM locations";
    $result = $this->dbConnect->query($locationQuery);
    $rows = array();

    while ($r = mysqli_fetch_assoc($result)) {
        $rows['data'][] = $r;
    }
    header('Content-Type: application/json');
    echo json_encode($rows);
}

public function getBrand(){
    $brandQuery = "SELECT * FROM brand";
    $result = $this->dbConnect->query($brandQuery);
    $rows = array();

    while ($r = mysqli_fetch_assoc($result)) {
        $rows['data'][] = $r;
    }
    header('Content-Type: application/json');
    echo json_encode($rows);
}

public function getCategorie(){
    $categorieQuery = "SELECT * FROM categories";
    $result = $this->dbConnect->query($categorieQuery);
    $rows = array();

    while ($r = mysqli_fetch_assoc($result)) {

```

Figura 3. 48 Código fuente servicio web pickers.

Fuente: Autor.

De lado de la aplicación móvil se implementó 3 clases asociadas con el acceso a la base de datos por el lado de los selectores y cada clase tiene asociada sus diferentes estructuras que implementa la API Codable.

```

import Foundation

struct Categorie : Codable{
    let data : [CategorieItems]
}

struct CategorieItems : Codable{
    let id : String
    let name : String
}

```

Figura 3. 49 Estructura de categorías.

Fuente: Autor.

```

import Foundation

struct Brand : Codable{
    let data : [BrandItems]
}

struct BrandItems : Codable{
    let id : String
    let name : String
}

```

Figura 3. 50 Estructura de marcas.

Fuente: Autor.

```

import Foundation

struct Location : Codable{

    let data : [LocationItems]
}

struct LocationItems : Codable{
    let id : String
    let name : String
}

```

Figura 3. 51 Estructura de locaciones.

Fuente: Autor.

Los datos requeridos al ser similares poseen una estructura similar para enviar las peticiones al servidor. A continuación, la API de categoría.

```

class CategorieAPI : ObservableObject{

    @Published var categorias = [String]()

    func loadCategoriePicker(){

        guard let requestUrl = URL(string: "http://192.168.1.15:8080/inventario/webservice.php") else{
            print("URL no encontrada")
            return
        }
        let parameters : [String: Any] = ["code": "2"]

        var request = URLRequest(url: requestUrl)

        var components = URLComponents()
        var queryItems = [URLQueryItem]()

        for (key, value) in parameters {
            let queryItem = URLQueryItem(name: key, value: String(describing: value))
            queryItems.append(queryItem)
        }
        components.queryItems = queryItems
        let queryItemData = components.query?.data(using: .utf8)
        request.httpBody = queryItemData
        request.httpMethod = "POST"
        request.setValue("application/x-www-form-urlencoded", forHTTPHeaderField: "Content-Type")
    }
}

```

Figura 3. 52 Función cargar categoría.

Fuente: Autor.

Se decodifican los datos usando el decodificador de JSON.

```

do{
    let json = try JSONSerialization.jsonObject(with: unwrappedData, options: [])
    print(json)

    if let result = try? JSONDecoder().decode(Categorie.self, from: unwrappedData){
        DispatchQueue.main.async {

            for(_, item) in result.data.enumerated(){
                self.categorias.append(item.name)
            }
            self.categorias = self.categorias.uniqued()
            print(self.categorias)
        }
    }

}
catch(let errorJson){
    print("error", errorJson.localizedDescription)
}
else{
    print("No hay datos")
}
}

```

Figura 3. 53 Decodificando JSON de categoría.

Fuente: Autor.

Para poder realizar la escritura de la etiqueta se necesita implementar el protocolo NFCNDEFSessionReaderDelegate en la clase NFCWriter, el cual por defecto provee de los métodos didInvalidateWithError para control de errores, didDetectNDEFS para leer mensajes y didDetect que está escuchando para poder encontrar una etiqueta.

```
func readerSession(_ session: NFCNDEFReaderSession, didInvalidateWithError error: Error) {
}

func readerSession(_ session: NFCNDEFReaderSession, didDetectNDEFS messages: [NFCNDEFMessage]) {
}

@available(iOS 13.0, *)
private func write(_ message: NFCNDEFMessage, to tag: NFCNDEFTag) {
    tag.queryNDEFStatus() { (status: NFCNDEFStatus, _, error: Error?) in
        guard status == .readWrite else {
            self.nfcSession?.invalidate(errorMessage: "No se puede escribir la etiqueta")
            return
        }

        tag.writeNDEF(message) { (error: Error?) in
            if (error != nil) {
                self.nfcSession?.invalidate(errorMessage: error?.localizedDescription ?? "Se produjo un error")
            }
            self.nfcSession?.alertMessage = "Mensaje guardado exitosamente"
            self.nfcSession?.invalidate()
        }
    }
}

@available(iOS 13.0, *)
func readerSession(_ session: NFCNDEFReaderSession, didDetect tags: [NFCNDEFTag]) {
}
```

Figura 3. 54 Métodos del protocolo NFCNDEFSessionReaderDelegate.

Fuente: Autor.

Dentro del método de didDetect se deben colocar los casos para los cuales se desarrolla la sesión y se debe escribir el payload o los datos a guardar en la etiqueta según los formatos soportados. En este caso se usa wellKnowTypeTextPayload.

```
@available(iOS 13.0, *)
func readerSession(_ session: NFCNDEFReaderSession, didDetect tags: [NFCNDEFTag]){

    guard let tag = tags.first else {
        session.invalidate(errorMessage: "No se puede leer la etiqueta")
        return
    }
    guard let message = self.nfcMessage else {
        session.invalidate(errorMessage: "Mensaje Invalido")
        return
    }

    session.connect(to: tag) { (error: Error?) in
        if error != nil {
            session.invalidate(errorMessage: "Error de Conexion.")
            return
        }
        self.write(message, to: tag)
    }
}

}
```

Figura 3. 55 Agregando payload en didDetect.

Fuente: Autor.

Los datos son encapsulados usando la clase NFCMessageEncoder donde se crea el payload que para este proyecto es tomado de datos tipo String y este se debe volver un mensaje NDEF que son los que recibe el método de escritura. Cada campo que se ingresa se volverá un mensaje individual.

```
import CoreNFC

class NFCMessageEncoder {

    func message(with urls: [URL], and texts: [String]) -> NFCNDEFMessage? {
        let payloads = self.payloads(from: texts) + self.payloads(from: urls)
        return (payloads.count > 0) ? NFCNDEFMessage(records: payloads) : nil
    }

    func message(with keys: [String], and texts: [String]) -> NFCNDEFMessage? {
        let payloads = self.payloads(from: keys) + self.payloads(from: texts)
        return (payloads.count > 0) ? NFCNDEFMessage(records: payloads) : nil
    }

    func message(texts: [String]) -> NFCNDEFMessage? {
        let payloads = self.payloads(from: texts)
        return (payloads.count > 0) ? NFCNDEFMessage(records: payloads) : nil
    }

    private func payloads(from texts: [String]) -> [NFCNDEFPayload] {
        return texts.compactMap { text in
            return NFCNDEFPayload.wellKnownTypeTextPayload(
                string: text,
                locale: Locale(identifier: "En")
            )
        }
    }
}
```

Figura 3. 56 Encapsulado del mensaje en un payload.

Fuente: Autor.

3.6.4. Lector NFC

Dentro del lector escribir se cuenta con una caja de texto y dos botones uno para leer la etiqueta y otro para escribir los datos obtenidos en la base de datos

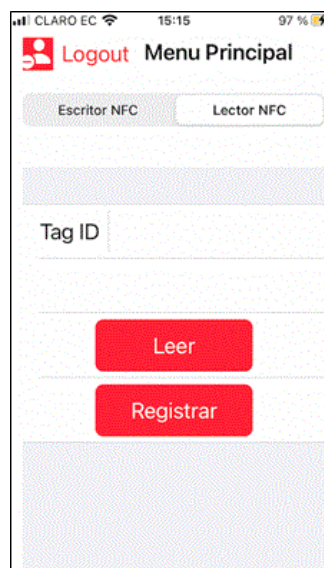


Figura 3. 57 Leer etiqueta.

Fuente: Autor.

Para efectuar la lectura de la etiqueta se recurre a la clase NFCWriter que implementa el protocolo NFCNDEFSessionReaderDelegate, al igual que la anterior clase que se usa para escritura, en esta se implementa los mismos métodos que vienen por defecto de manera obligatoria con el protocolo. El método en el cual se crea la lógica de lectura es didDetectNDEFS. Se debe decodificar el dato que viene de la etiqueta y obtener su respectivo payload.

```
private func read(_ tag: NFCNDEFTag) {
    tag.readNDEF { (message: NFCNDEFMessage?, error: Error?) in
        guard error == nil else {
            self.nfcSession?.invalidate(errorMessage: error!.localizedDescription)
            return
        }
        guard let message = message else {
            self.nfcSession?.invalidate(errorMessage: "No hay ningún mensaje")
            return
        }
        self.successReading?(message)
        self.nfcSession?.alertMessage = "Mensaje leído exitosamente"
        self.nfcSession?.invalidate()
    }
}

@available(iOS 13.0, *)
func readerSession(_ session: NFCNDEFReaderSession, didDetect tags: [NFCNDEFTag]) {
    guard let tag = tags.first else {
        session.invalidate(errorMessage: "No se pudo leer la etiqueta")
        return
    }
    session.connect(to: tag) { (error: Error?) in
        if error != nil {
            session.invalidate(errorMessage: "Error de conexión, trate de nuevo.")
            return
        }
    }
}
```

Figura 3. 58 Código fuente Leer etiqueta.

Fuente: Autor.

La decodificación se realiza por medio de la clase NFCMessageDecoder, la cual se encarga de devolver el tipo exacto de payload según los datos que encuentre almacenado en la etiqueta.

```
class NFCMessageDecoder {
    func decode(_ message: NFCNDEFMessage) -> [String] {
        return message.records.compactMap { payload in
            return self.decode(payload)
        }
    }

    private func decode(_ payload: NFCNDEFPayload) -> String? {
        let type = PayloadType(rawValue: payload.type.decode())
        switch type {
        case .uri:
            return payload.wellKnownTypeURIPayload()?.absoluteString
        case .text:
            return payload.wellKnownTypeTextPayload()?.0
        default:
            return nil
        }
    }
}

extension Data {
    func decode() -> String {
        guard let value = String(data: self, encoding: .utf8) else {
            return ""
        }
        return value
    }
}
```

Figura 3. 59 Clase decodificadora de payload.

Fuente: Autor.

3.6.4.1. Método de registro

En el método de registro se obtiene la información del último registro almacenado en la etiqueta y se escribe en la base de datos, pero primero se obtiene el id único de la etiqueta NFC para poder realizar actualizaciones en la base de datos en caso de existir. Para eso se utiliza el protocolo NFCTagReaderSessionDelegate.

```
func tagReaderSessionDidBecomeActive(_ session: NFCTagReaderSession) {
    print("Session empezó")
}

func tagReaderSession(_ session: NFCTagReaderSession, didInvalidateWithError error: Error) {
    print(error.localizedDescription)
}

func tagReaderSession(_ session: NFCTagReaderSession, didDetect tags: [NFCTag]) {
```

Figura 3. 60 Protocolo NFCTagReaderSessionDelegate.

Fuente: Autor.

El método que se va a modificar es el de didDetect para el cual se debe identificar el tipo de etiqueta que se va a leer que esté dentro de los estándares ISO respectivos.

```
func scan(){
    // ISO 14443 and ISO 15693 tags
    self.session = NFCTagReaderSession(pollingOption: [.iso14443, .iso15693], delegate: self)
    self.session?.begin()
}
```

Figura 3. 61 Opción de tipo de etiqueta ISO 14443.

Fuente: Autor.

```
func tagReaderSession(_ session: NFCTagReaderSession, didDetect tags: [NFCTag]) {
    if tags.count > 1 {
        let retryInterval = DispatchTimeInterval.milliseconds(500)
        session.alertMessage = "Mas de una etiqueta conectada"
        DispatchQueue.global().asyncAfter(deadline: .now() + retryInterval, execute: {
            session.restartPolling()
        })
        return
    }

    let tag = tags.first!
    session.connect(to: tag){(error) in
        if nil != error{
            session.invalidate(errorMessage: "Fallo de conexion")
        }

        if case let .mifare(sTag) = tag{
            let UID = sTag.identifier.map{String(format: "%.2hhx", $0)}.joined()
            print("UID:", UID)
            session.invalidate()
            DispatchQueue.main.async {
                self.tagUID = UID
            }
        }
    }
}
```

Figura 3. 62 Método para capturar UID con formato.

Fuente: Autor.

Una vez obtenido el UID se procede a realizar la escritura en la base de datos y para ello se recurre nuevamente al servicio web en donde se utiliza el método de inserción por medio de la siguiente función en el servicio web. Se debe verificar que no exista ya registrado una id de una misma etiqueta con la finalidad de guardar la última actualización del producto. Para eso se recurre a una función para verificar la existencia del id y en caso de existir se actualiza, caso contrario se inserta.

```
public function getTagID($productData)
{
    $tagID = $productData["tag_id"];
    $tagQuery = $this->dbConnect->prepare("SELECT tag_id FROM products WHERE tag_id = ? ");
    $tagQuery->bind_param('s', $tagID);
    $tagQuery->execute();
    $result = $tagQuery->get_result();
    if ($result->num_rows > 0) {
        $found = true;
    } else {
        $found = false;
    }
    return $found;
}
```

Figura 3. 63 Verificar Existencia de ID de tarjeta NFC en el servicio web.

Fuente: Autor.

La función de la Figura 3.54 ayuda a identificar si la etiqueta existe y devolver un valor de verdadero o falso. Si fue verdadero se actualiza de la siguiente manera:

```
public function getTagID($productData)
{
    $tagID = $productData["tag_id"];
    $tagQuery = $this->dbConnect->prepare("SELECT tag_id FROM products WHERE tag_id = ? ");
    $tagQuery->bind_param('s', $tagID);
    $tagQuery->execute();
    $result = $tagQuery->get_result();
    if ($result->num_rows > 0) {
        $found = true;
    } else {
        $found = false;
    }
    return $found;
}
```

Figura 3. 64 Ingreso del registro en servicio web.

Fuente: Autor.

Se puede observar que en caso de que la etiqueta si existiera en la base de datos se debe actualizar en vez de insertar por lo que se ejecuta el siguiente método:


```

public function updateProduct($productDataParamU)
{
    $product_name = $productDataParamU['Nombre'];
    $administrador = $productDataParamU['Administrador'];
    $categorie_id = intval($this->getCategorieID($productDataParamU));
    $location_id = intval($this->getLocationID($productDataParamU));
    $brand_id = intval($this->getBrandID($productDataParamU));
    $serial_code = $productDataParamU['Serial'];
    $product_status = $productDataParamU['Estado'];
    $features = $productDataParamU['Caracteristicas'];
    $productDate = $productDataParamU['Fecha'];
    $tag = $productDataParamU['tag_id'];
    $queryUpdate = "UPDATE products SET name=?,administrador=?, categorie_id=?, location_id=?,
    $queryProcess = $this->dbConnect->prepare($queryUpdate);
    $queryProcess->bind_param('ssiiissss', $product_name, $administrador, $categorie_id, $location_id, $brand_id, $serial_code, $product_status, $features, $productDate, $tag);
    if ($queryProcess->execute()) {
        $status = "2";
        $messgae = "Registro Actualizado exitoso";
    } else {
        $status = "0";
        $messgae = "Fallo al Actualizar el registro";
    }
    $updateResponse = array(
        'status' => $status,
        'status_message' => $messgae
    );
    header('Content-Type: application/json');
    echo json_encode($updateResponse);
}

```

Figura 3. 65 Actualización del registro en servicio web.

Fuente: Autor.

En la aplicación móvil se debe escribir la estructura encargada de recibir la respuesta y proceso que enviará los datos.

```

import Foundation

struct Product: Codable {

    var status : String = ""
    var status_message : String = ""
}

```

Figura 3. 66 Estructura de producto.

Fuente: Autor.

Se debe enviar los datos obtenidos del TextEditor a la API de actualización que simplemente mandará todos los campos encontrados en el último registro de la etiqueta NFC.

```

func editDictionary(value: [String]){
    var dictionary : [String : String] = [:]
    var data = [String]()
    var Key : String = ""
    var Value : String = ""

    for i in 0..

```

Figura 3. 67 Estructurando los parámetros del TextEditor.

Fuente: Autor.

```

func updateProductDB(dicData: [String : String]){

    guard let requestUrl = URL( string: "http://192.168.1.4:8080/inventario/webservice.php") else{
        return
    }

    var parameters : [String:Any] = dicData
    parameters["code"] = "5"

    var request = URLRequest(url: requestUrl)

    var components = URLComponents()
    var queryItems = [URLQueryItem]()

```

Figura 3. 68 Estructurando los parámetros del TextEditor

Fuente: Autor.

Por último, se manda todos los datos por medio de la URL utilizando POST.

```

for (key, value) in parameters {
    let queryItem = URLQueryItem(name: key, value: String(describing: value))
    queryItems.append(queryItem)
}
components.queryItems = queryItems
let queryItemData = components.query?.data(using: .utf8)
request.httpBody = queryItemData
request.httpMethod = "POST"

request.setValue("application/x-www-form-urlencoded", forHTTPHeaderField: "Content-Type")

let session = URLSession.shared
let task = session.dataTask(with: request){ (data, response, error) in

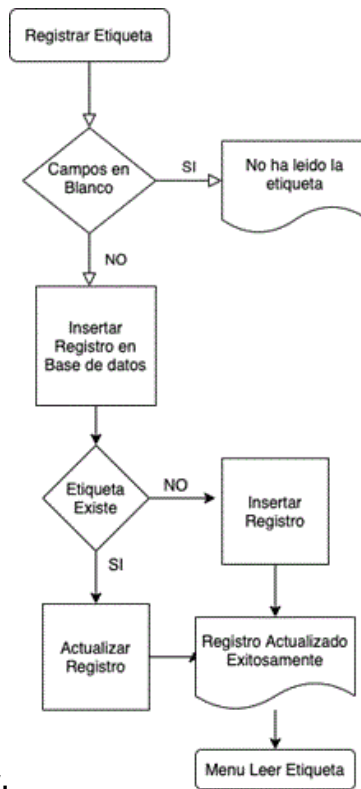
    guard let data = data, error == nil else{

        return
    }
    do{
        let unwrappedData = try JSONDecoder().decode(Product.self, from: data)

        DispatchQueue.main.async {
            print(unwrappedData.status_message)
            if(unwrappedData.status != "0"){
                self.showProgressView = true
            }
        }
    }
}

```

Figura 3. 69 Envío de Petición de registro a la base de datos.



Fuente: Autor.

Figura 3. 70 Diagrama de flujo de registro.

Fuente: Autor

Capítulo 4. Pruebas y Resultados

4.1. componentes de hardware y software implementados

Los componentes utilizados para el desarrollo de las aplicaciones se detallan en la siguiente tabla:

Tabla 4. 1 Característica de las etiquetas NTAG25

Hardware/ Software	Características	
Servidor Linux XAMPP	Sistema Operativo	Linux debían 4.19.0-17
	Servidor	Apache/2.2.48 (Unix)
	Version de PHP	8.0.10
	Base de datos	10.4.21- MariaDB
Entorno de desarrollo Integrado	XCode Version	13
Editor de Texto	Microsoft Visual Studio Code	Version: 1.61
Etiqueta NFC	Marca	THONSEN
	Modelo	NTAG25
Terminal Celular	Marca	Apple
	Iphone	8
	IOS	15.0.2

Fuente: Autor.

Las pruebas fueron realizadas en la terminal descrita en la tabla 4.1 y las etiquetas utilizadas son del tipo NTAG25 y pueden ser visualizadas en la figura 4.1 que se encuentra a continuación.



Figura 4. 1 Etiquetas NFC.

Fuente: Autor

4.2. Escenario de pruebas

Las pruebas realizadas cuentan con las siguientes características:

- La aplicación móvil debe estar instalada en el equipo celular.
- La aplicación móvil y la aplicación web deben contar con conectividad a la red.
- Deben colocarse las etiquetas NTAG25.
- Las etiquetas utilizadas deben ser compatibles con IOS.

- El equipo celular utilizado debe soportar tecnología NFC .

4.3. Pruebas de funcionalidad

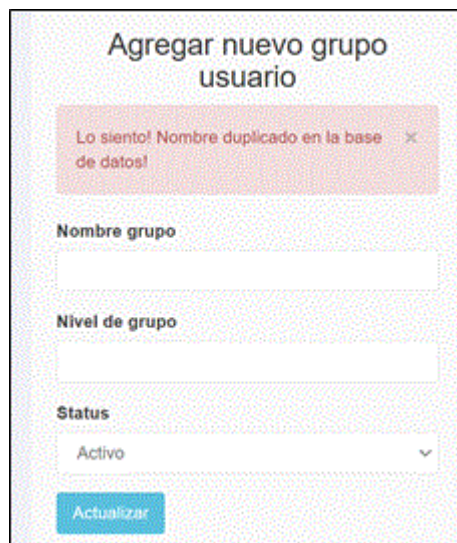
A continuación, se presentan las pruebas hechas tanto a la aplicación web como móvil.

4.3.1. Pruebas de aplicación web

4.3.1.1. Añadir un nuevo grupo

Una de las acciones a realizar es añadir un nuevo grupo como se detalla en el capítulo 3. Los escenarios que se puedan presentar se detallan a continuación:

Si el grupo que se quiere añadir ya existe, se devuelve un mensaje indicando su duplicidad por lo que no se registra. Los grupos son asociados a los usuarios y manejan los permisos con los que cuentan.



La imagen muestra una interfaz de usuario para agregar un nuevo grupo de usuario. El título es "Agregar nuevo grupo usuario". Hay un mensaje de error rojo que dice "Lo siento! Nombre duplicado en la base de datos!". El formulario tiene tres campos: "Nombre grupo", "Nivel de grupo" y "Status" (con "Activo" seleccionado). Un botón "Actualizar" está al final.

Figura 4. 2 Nombre de grupo Duplicado.

Fuente: Autor

Si los campos están vacíos se devuelve un mensaje indicando el error y no se completa la acción.

Figura 4. 3 Nombre de grupo en blanco.
Fuente: Autor

Por último, cuando los campos son ingresados correctamente se tiene la siguiente pantalla:

Figura 4. 4 Grupo Ingresado Correctamente.
Fuente: Autor

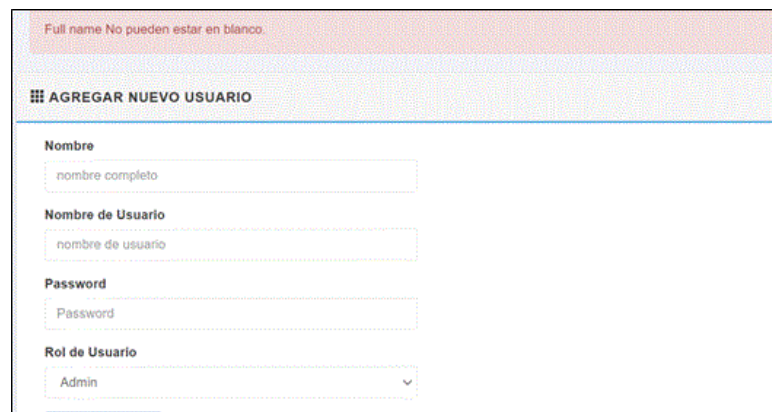
La lista de los grupos creados se actualizará después de que agrega un nuevo registro:

GRUPOS		AGREGAR NUEVO GRUPO		
#	Nombre de grupo	Nivel de grupo	Estado	Acciones
1	Admin	1	Active	
2	Special	2	Active	
3	User	3	Active	
4	Pruebas	4	Active	

Figura 4. 5 Listado de Grupos.
Fuente: Autor

4.3.1.2. Añadir un nuevo usuario

En la aplicación se pueden agregar usuarios con su respectivo nombre de usuario y contraseña. A su vez estos usuarios son agregados a un rol respectivo asignado a los grupos antes creados. Pueden presentarse dos casos el primero en el caso de que los campos estén incompletos la aplicación web devuelve un mensaje de error.



Full name No pueden estar en blanco.

AGREGAR NUEVO USUARIO

Nombre
nombre completo

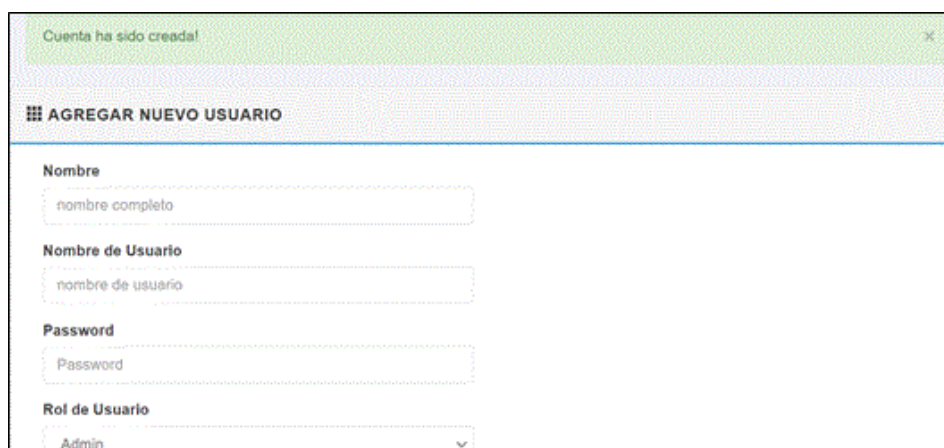
Nombre de Usuario
nombre de usuario

Password
Password

Rol de Usuario
Admin

Figura 4. 6 Datos de usuario en blanco.
Fuente: Autor

En el siguiente caso los datos están completos y almacena exitosamente el usuario por lo que es agregado a la lista de usuarios activos.



Cuenta ha sido creada!

AGREGAR NUEVO USUARIO

Nombre
nombre completo

Nombre de Usuario
nombre de usuario

Password
Password

Rol de Usuario
Admin

Figura 4. 7 Datos de usuario creados.
Fuente: Autor



#	Nombre	Nombre de usuario	Rol	Estado	Ultima conexion	Acciones
1	Pruebas De Tesis	PruebasT	Admin	Active		
2	Rafael	Raf	Admin	Active	October 16, 2021, 5:26:07 am	
3	Usuario Administrador	Admin	Admin	Active	October 31, 2021, 8:43:53 am	

Figura 4. 8 Lista de usuario creados.
Fuente: Autor

4.3.1.3. Añadir una nueva categoría

Para poder agregar un nuevo bien se debe asociar una categoría, marca y locación donde el bien será almacenado. Así se podrá tener una constancia física de donde es guardado el recurso de manera más detallada



Figura 4. 9 Lista de Categorías.
Fuente: Autor



Figura 4. 10 Categoría agregada.
Fuente: Autor

4.3.1.4. Añadir una nueva marca

Al igual que el caso de la categoría, se debe agregar una marca que será asociada al bien.



Figura 4. 11 Listado de Marcas.
Fuente: Autor



Figura 4. 12 Marca agregada.
Fuente: Autor

4.3.1.5. Añadir una nueva locación

Por último, se puede agregar locaciones donde serán almacenados físicamente los bienes muebles.

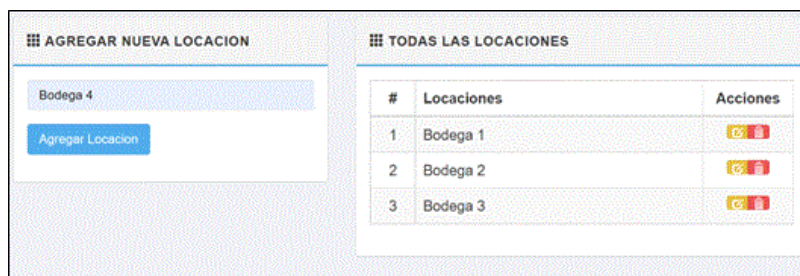


Figura 4. 13 Listado de locaciones.
Fuente: Autor

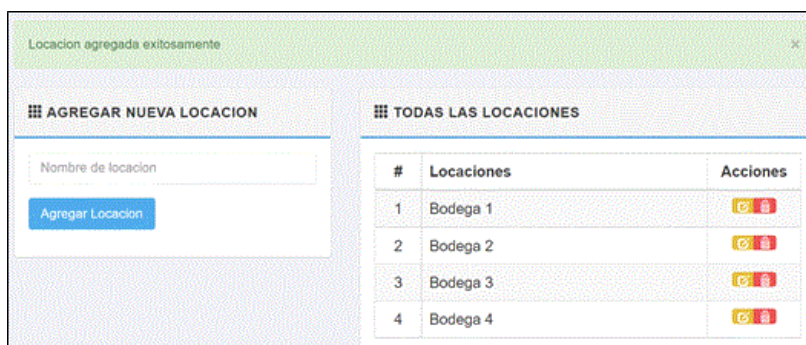


Figura 4. 14 Locaciones agregadas.
Fuente: Autor

4.3.1.6. Añadir un nuevo bien

Como se indicó con anterioridad, la aplicación web también permite añadir bienes que no necesiten o no se les pueda agregar una etiqueta NFC. Pueden dejarse ciertos datos en blanco excepto los más importantes como nombre, categoría, marca, etc. A continuación, un escenario donde se deja un campo importante incompleto.

Product administrator No pueden estar en blanco.

AGREGAR NUEVO BIEN

Nombre del bien

Nombre del administrador

Codigo serial

estado del bien

caracteristicas

Seleccione categoria

Seleccione la locacion

Figura 4. 15 Agregar bien con datos incompletos.
Fuente: Autor

Cuando los campos están completos, se emite un mensaje de guardado exitoso y se procede a ingresar el bien al listado de inventario registrado.

AGREGAR NUEVO BIEN

Computador

Rafael

LK56743HGF

Nuevo

computador portatil

Computador Personal

Bodega 2

Apple

Seleccione imagen del producto

1

\$ 0.00

Fecha Mantenimiento: 28/10/2021

Fecha de Compra: 28/10/2021

Figura 4. 16 Proceso de nuevo registro.
Fuente: Autor

Bien agregado

AGREGAR NUEVO BIEN

Nombre del bien

Nombre del administrador

Codigo serial

estado del bien

caracteristicas

Seleccione categoria

Seleccione la locacion

Figura 4. 17 Mensaje de registro de bien exitoso.
Fuente: Autor

Después de haber guardado los datos, los mismos pueden ser verificados por medio del listado general de inventarios registrados.

AGREGAR NUEVO											
Título de la	Categoría	Cantidad	Precio	Precio de venta	Bien agregado	Fecha Compra	Fecha Mantenimiento	Encargado	Serial	Locacion	M
Computador	Computador Personal	1	0.00	0.00	November 2, 2021, 7:56:43 am	2021-10-28 00:00:00	2021-10-28 00:00:00	Rafael	LK56743HGF	Bodega 2	A

Figura 4. 18 Listado actualizado con un nuevo bien.
Fuente: Autor

4.3.1.7. Editar cuenta de usuario

La aplicación web también permite que se pueda editar los datos de los usuarios ya creados.

The screenshot shows a user profile page with two main sections. On the left, under 'CAMBIAR MI FOTO', there is a profile picture placeholder and a 'Cambiar' button. On the right, under 'EDITAR MI CUENTA', there are three input fields: 'Nombre' (containing 'Usuario Administrador'), 'Nombre Usuario' (containing 'Admin'), and a 'Cambiar contraseña' button. An 'Actualizar' button is also present.

Figura 4. 19 Edición de cuenta.
Fuente: Autor

This screenshot shows the same 'EDITAR MI CUENTA' form after a successful update. A green notification banner at the top reads 'Account updated'. The 'Nombre' field now contains 'Rafael Edicion', while the 'Nombre Usuario' field remains 'Admin'. The 'Actualizar' and 'Cambiar contraseña' buttons are still visible.

Figura 4. 20 Cuenta editada.
Fuente: Autor

4.3.2. Pruebas de aplicación móvil

La aplicación móvil fue probada en un teléfono iPhone 8 con sistema operativo IOS.

4.3.2.1. Login

Una de las acciones a realizar es añadir un nuevo grupo como se detalla en el capítulo 3. Los escenarios que se puedan presentar se detallan a continuación:

El primer caso en que no se coloque ni el usuario ni la contraseña y se trata de ingresar a la aplicación, saltará un mensaje de alerta que indica que debe completar los campos.

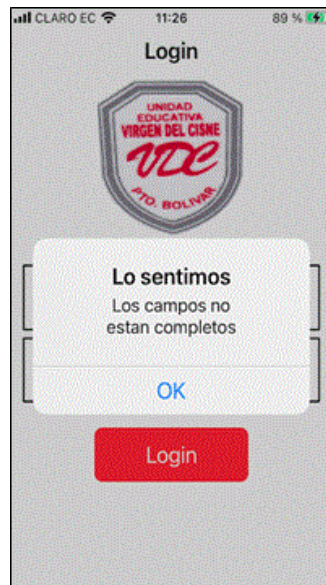


Figura 4. 21 Validación de campos en blanco en pantalla de Login.

Fuente: Autor

En el caso de que tanto el usuario y la contraseña proporcionado sean incorrectos, se muestra un mensaje de alerta que indica lo sucedido.

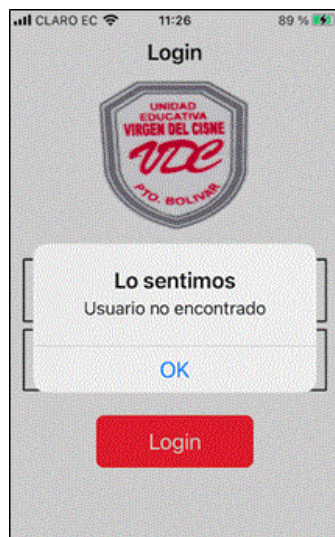


Figura 4. 22 Validación de usuario no encontrado en pantalla de Login.

Fuente: Autor

Si el usuario ha sido encontrado, entra directamente al menú principal.



Figura 4. 23 Menú principal con el nombre del usuario.
Fuente: Autor

4.3.2.2. Escribir etiqueta

En la vista de escribir etiquetas, se procede a llenar los datos donde solo el nombre de usuario no es necesario, ya que este se trae desde la interfaz de Login para identificar quien es la persona que está modificando la etiqueta. Se procede a ingresar los campos.

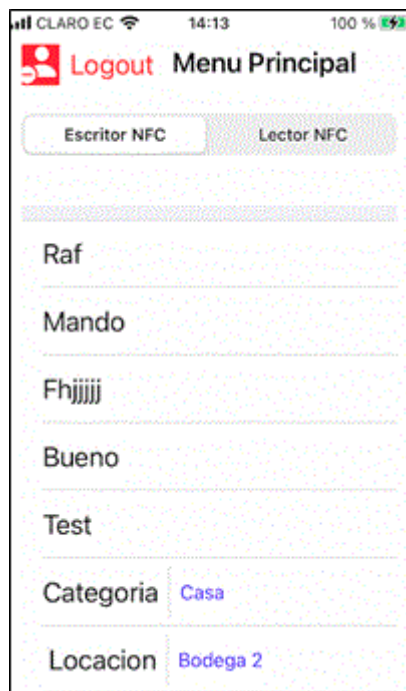


Figura 4. 24 Ingreso de campos en escritor NFC.
Fuente: Autor

Después de haber ingresado los campos, se procede a presionar el botón escribir, lo que mandará un mensaje para que se acerque el dispositivo a la etiqueta.

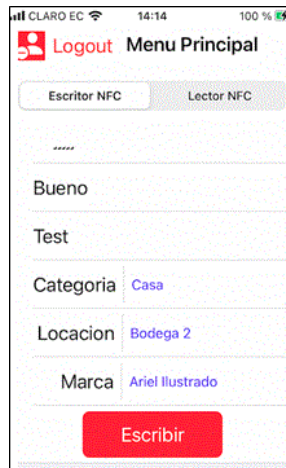


Figura 4. 25 Botón escribir escritor NFC.
Fuente: Autor

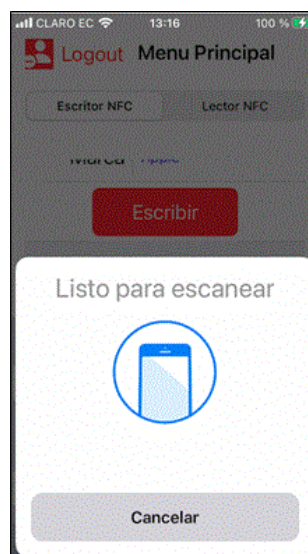


Figura 4. 26 Mensaje de escaneo de escritor NFC.
Fuente: Autor

Al terminar el mensaje de escaneo de la Figura 4.25, la etiqueta estará escrita con los datos ingresados. Cada valor ingresado constituirá un mensaje de tipo NDEF, por lo que por ejemplo el campo administrador con su respectivo nombre será un solo mensaje. Varios campos, significan varios mensajes.

4.3.2.3. Leer etiqueta

La opción de leer etiqueta está enlazada con la opción de guardar el registro en la base de datos. No se puede realizar la transacción hacia la base si primero no se obtenido el UID (Unique ID) de la etiqueta NFC, ya

que con este código servirá para darle un identificador único irrepitable en la base de datos.

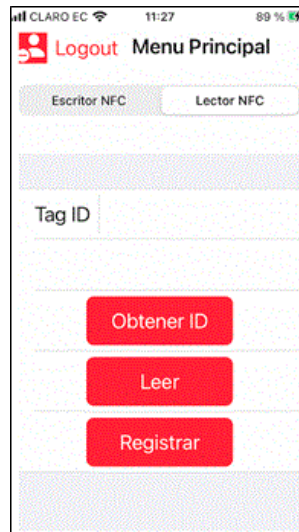


Figura 4. 27 Lector NFC.

Fuente: Autor

Primero se escanea el ID de la etiqueta, luego se escanea la etiqueta en si para obtener los mensajes NDEF que tiene almacenados y se carga el texto en la pantalla.

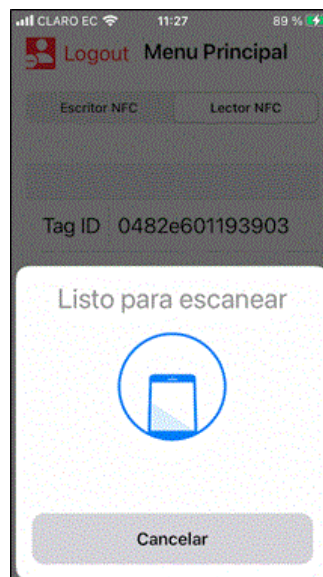


Figura 4. 28 Escaneo etiqueta NFC.

Fuente: Autor

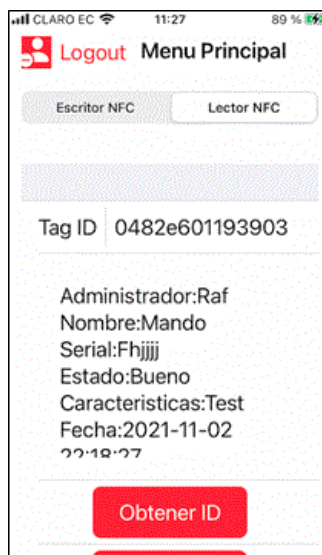


Figura 4. 29 Etiqueta NFC escaneada.
Fuente: Autor

4.3.2.3. Registro de etiqueta

Después de haber escaneado los datos necesarios de la etiqueta como sus mensajes NDEF y su identificador UID, se procede a registrar la información de la etiqueta en la base de datos.

Titulo de la foto	Categoria	Cantidad	Precio de venta	Bien agregado	Fecha Compra	Fecha Mantenimiento	Encargado	Serial	Locacion	Marca
Computador	Computador Personal	1	0.00	0.00	November 2, 2021, 7:56:43 am	2021-10-28 00:00:00	Rafael	LK56743HGF	Bodega 2	Apple
proeba	Casa	1	0.00	0.00	November 3, 2021, 12:00:00 am	2021-11-03 00:00:00	Raf	ond ncd	Bodega 1	Apple
Mando	Casa	1	0.00	0.00	November 2, 2021, 10:00:00 pm		Raf	Fhjjj	Bodega 2	Ariel Ilustrad

Figura 4. 30 Nuevo registro agregado.
Fuente: Autor

4.4. Análisis de los resultados

4.4.1. Login

La aplicación cuenta con dos roles de usuarios los que son usuario administrador y especial. Solo los usuarios que son administradores pueden agregar registros a la base de datos. Los administradores son los únicos que pueden crear nuevos usuarios y otorgar las credenciales necesarias para el ingreso. Este usuario también es el único que puede editar la contraseña en caso de pérdida.

4.4.2. Registro en la base de datos

Tanto la aplicación móvil como la web solo pueden ingresar los registros en la base de datos sin cumplir con los campos obligatorios, sin embargo, se pueden dejar ciertos registros en blanco según el tipo de aplicación.

4.4.3. Visualización de la información procesada

Se comprueba que cada registro ha sido ingreso en la página de Administrar Bienes de la aplicación web. El respaldo también queda almacenado en la base de datos. Se cuenta con dos registros que no son visibles para el usuario que son la fecha actual y la de mantenimiento. La fecha actual se graba cuando se registra la primera vez y de ahí en adelante se graba la de mantenimiento.

4.4.4. Lectura y escritura

Al realizar las operaciones de lectura y escritura, se puede comprobar que se presentan errores cuando al tratar de leer una etiqueta el dispositivo móvil no se encuentran a la distancia mínima requerida para la conexión, por lo que es recomendable que se encuentre a una distancia aproximada de 1 centímetro. No afecta en la conexión que el dispositivo móvil tenga algún protector o carcasa. Los valores que se almacenan en la etiqueta NFC y los que se escriben por parte del usuario son cadenas de caracteres. Cualquier valor numérico es convertido a cadena.

4.4.5. Errores de conexión

Tanto la aplicación móvil como la web dependen de la conectividad de Internet. En el caso de la aplicación móvil, el servicio web implementado es su principal medio de comunicación con la base de datos. En caso de que se susciten fallas en la red, no se podría acceder a ninguno de los dos servicios.

Conclusiones

De acuerdo a los resultados, el prototipo implementado cumple con la finalidad de brindar a la Unidad Educativa Particular Mixta “Virgen del Cisne” un sistema de administración de inventarios con las opciones de llevar registros mas precisos gracias a la tecnología NFC.

NFC permite una eficiencia optimizada del sistema por la facilidad de configuración de las etiquetas y su disponibilidad en sistemas operativos IOS. El campo de acción de esta es limitada a unos cuantos centímetros y no es necesario implementar un complejo protocolo de seguridad.

Se implementó la base de datos MariaDB, la cual es una versión Open Source de Mysql disponible para el público en general. La base de datos es el componente fundamental de las aplicaciones ya que en la misma se almacenan todos los registros de los bienes ingresados. Al utilizar este software, no se limita el número de nuevas filas y columnas que pueden ser ingresados a la base, por lo que, si el usuario lo requiere, pueden ingresarse mas campos.

La aplicación web cumple con las funciones deseadas para la administración del sistema como el control de bienes muebles y usuarios designados. La interacción del lado del usuario es amigable gracias a su estructura desarrollada en HTML y CSS, lo que permite fáciles cambios en caso de ser necesarios y su portabilidad a cualquier SO, debido a que funciona con un explorador. A su vez la comunicación con el servidor se da mediante el lenguaje PHP, ampliamente utilizado en la actualidad.

La aplicación móvil desarrollada con XCode, cumple con los objetivos planteados; escribe la etiqueta, la lee y permite visualizar los datos almacenados del bien mueble. Permite ingresar un nuevo inventario ingresando los campos requeridos como: nombre del bien, administrador, código serial del bien, fecha de registro, categoría, marca y locación. Para poder visualizar el registro del bien, se puede verificar fácilmente con la opción de lectura.

La comunicación de la aplicación móvil se produce por medio del servicio web desarrollado en PHP, lo que permite que el usuario final pueda acceder a la base de datos interactuando con la aplicación.

Recomendaciones

- Para un correcto funcionamiento el dispositivo móvil deberá acercarse lo suficiente a la etiqueta NFC hasta que la aplicación muestre el mensaje de éxito en el proceso de escritura o lectura.
- No se debe utilizar el mismo identificador para varios bienes ya que esto puede desencadenar inconsistencias con el sistema de la base de datos, por eso solo se deben escribir los registros que tienen asociado un mismo ID solo en casos de actualización.
- Se recomienda el uso de etiquetas tipo NTAG12x que son compatibles con la aplicación e IOS ya que son por lo general del tipo Mifare.
- Se podría mejorar el sistema agregando una opción para visualizar el inventario desde la aplicación móvil para que el usuario pueda verificar que se ha ingresado en la base de datos el nuevo registro.
- Se puede utilizar el sistema NFC para almacenar un registro de los bienes que sean prestados a los estudiantes de la institución para que sean devueltos en los tiempos establecidos.
- Se podrían utilizar otro tipo de etiquetas de la serie NTAG12x que sean para diferentes superficies.

Bibliografía

- Ahson, S., & Ilyas, M. (2008). *RFID handbook : applications, technology, security, and privacy*. CRC Press.
- Apple. (2011). *wayback machine*. Obtenido de Carbon: <https://web.archive.org/web/20121012135554/https://developer.apple.com/carbon/>
- Apple. (16 de Septiembre de 2015). *Documentation Archive*. Obtenido de Kernel and Device Drivers Layer: https://developer.apple.com/library/archive/documentation/MacOSX/Conceptual/OSX_Technology_Overview/SystemTechnology/SystemTechnology.html#//apple_ref/doc/uid/TP40001067-CH207-BCICAIFJ
- Apple. (16 de Septiembre de 2015). *Documentation Archive*. Obtenido de Kernel and Device Drivers Layer: https://developer.apple.com/library/archive/documentation/MacOSX/Conceptual/OSX_Technology_Overview/SystemTechnology/SystemTechnology.html#//apple_ref/doc/uid/TP40001067-CH207-BCICAIFJ
- Apple. (4 de Junio de 2018). *Documentation Archive*. Obtenido de Cocoa Core Competencies: <https://developer.apple.com/library/archive/documentation/General/Conceptual/DevPedia-CocoaCore/Cocoa.html>
- Bhattacharyya, M. (7 de Mayo de 2018). *ResearchGate*. Recuperado el Noviembre de 2020, de An ultra-low-power RFIDNFC frontend chip using 018 um CMOS technology for passive tag applications: https://www.researchgate.net/figure/A-brief-overview-of-the-available-NFC-standards_tbl1_325016662
- Bulavin, V. (7 de Octubre de 2019). *Yet Another Swift Blog*. Obtenido de Advanced iOS Memory Management with Swift: ARC, Strong, Weak and Unowned Explained: <https://www.vadimbulavin.com/swift-memory-management-arc-strong-weak-and-unowned/>
- Castillo, E. (2018). *Diseño de una etiqueta pasiva sin chip para aplicaciones RFID en UWB*. Guayaquil: UCSG.
- Cedeño, V. A. (Agosto de 2016). *Repositorio Universidad de Guayaquil*. Obtenido de DISEÑO E IMPLEMENTACIÓN DE UN PORTAL WEB PARA EL CONSEJO NACIONAL ELECTORAL (CNE) A FIN DE AYUDAR EN LA CAPACITACIÓN A LOS CIUDADANOS DEL ECUADOR ACERCA DEL CÓDIGO DE LA DEMOCRACIA, APLICANDO METODOLOGÍA SCRUM PARA LA GESTIÓN DE PROYECTOS ÁGILES EN: <http://repositorio.ug.edu.ec/bitstream/redug/17058/1/UG-FCMF-B-CINT-PTG-N.112.pdf>
- Chandra Karmakar, N. (2010). *Handbook of smart antennas for RFID systems*. John Wiley & Sons.
- Costa, F., Genovesi, S., & Monorchio, A. (2013). A Chipless RFID Based on Multiresonant High-Impedance Surfaces. *IEEE TRANSACTIONS ON MICROWAVE THEORY AND TECHNIQUES*, 61(1).

- Gangula, M. R. (Marzo de 2019). *St. Cloud State University theRepository at St. Cloud State*. Obtenido de Overcoming Forensic Implications with Enhancing Security in iOS: https://repository.stcloudstate.edu/cgi/viewcontent.cgi?article=1108&=&context=msia_etds&=&sei-redir=1&referer=https%253A%252F%252Fscholar.google.es%252Fscholar%253Fq%253DDarwin%252BiOS%252Bkernel%2526hl%253Des%2526as_sdt%253D0%25252C5%2526as_ylo%253D2018
- Gomez, J. L. (2016). *Desarrollo de aplicaciones web en el entorno servidor UF1844*. Madrid: Paraninfo.
- Heras, A., las, F., Gómez, C., Franco, M. E., & Marzábal, M. (2009).
- Honan, M. (8 de Enero de 2007). *Macworld*. Obtenido de Apple unveils iPhone: <https://www.macworld.com/article/183056/iphone-5.html>
- IBM. (2013). *IBM Documentation*. Obtenido de API concepts: <https://www.ibm.com/docs/en/i/7.2?topic=interfaces-api-concepts>
- Jirasereeamornkol, J. W. (2005). Power Harvest Design for Semi-Passive UHF RFID Tag Using a Tunable Impedance Transformation. *9th Internatinal Symposium on Communications and Information Tech*, 1441-1445.
- Matillion. (18 de Noviembre de 2020). *Matillion*. Obtenido de The Types of Databases (with Examples): <https://www.matillion.com/resources/blog/the-types-of-databases-with-examples>
- Mitchell, L. J. (2016). *PHP Web Services 2nd Edition*. O'Reilly Media, Inc.
- Munk, B. A. (2000). *Frequency Selective Surfaces. Theory and Design*. John Wiley & Sons.
- Neuburg, M. (2020). *iOS 14 Programming Fundamentals with Swift, Seventh Edition*. (R. Roumeliotis, Ed.) California, United States: O'Reilly Media, Inc.
- O'Reilly. (2021). *O'Reilly*. Obtenido de Chapter 4. Introducing NDEF: <https://www.oreilly.com/library/view/beginning-nfc/9781449324094/ch04.html>
- Oracle. (2021). *Oracle*. Obtenido de What Is a Database?: <https://www.oracle.com/database/what-is-database/>
- Pereira de Siqueira Campos, A. (2008). *Superfícies Seletivas em Frequência: análise e projeto*. IFRN.
- PHP. (2021). *PHP*. Obtenido de What is PHP?: <https://www.php.net/manual/en/intro-what-is.php>
- Preradovic, S., & Karmakar, N. C. (2012). *Multiresonator-Based Chipless RFID. Barcode of the future*. Springer.
- Robert P. Sabella. (2016). *NFC for dummies* (Vol. 1). Hoboken, New Jersey, United States: John Wiley & Sons, Inc.
- Sabella, R. P. (2016). Hoboken, New Jersey, United States: John Wiley & Sons, Inc.
- Sabella, R. P. (2016). *Dummies*. Obtenido de The NFC Tag Initiation Secuence: <https://www.dummies.com/consumer-electronics/nfc-tag-initiation-sequence/>
- Sabella, R. P. (2016). *Dummies* . Obtenido de The 5 NFC Tag Types: <https://www.dummies.com/consumer-electronics/5-nfc-tag-types/>

- San José, J., Pastor, J., & García, A. (2012). RFID: La Identificación por Radiofrecuencia como futuro de la identificación de objetos. Obtenido de <https://www.researchgate.net/publication/275020704>
- Sánchez, R. (2014). *ETIQUETA PASIVA DE RFID SIN CHIP PARA SENSADO DE MATERIALES*. Tesis en opción al grado de Maestro en Ciencias en la especialidad de Electrónica., Instituto Nacional de Astrofísica, Óptica y Electrónica., Puebla. México.
- Santos, P. (2016). *Diseño de una antena tag RFID pasiva de orden cero (ZOR) en UHF con metamateriales*. Guayaquil: UCSG.
- Sharma, P. (27 de Septiembre de 2019). *DZone*. Obtenido de Automatic Reference Counting (ARC) and Memory Management in Swift: <https://dzone.com/articles/automatic-reference-counting-arc-and-memory-manage>
- Singh, A. (2006). *Mac OS X Internals: A Systems Approach*. Westford, Massachusetts, United States: Addison-Wesley Professional.
- Suero, L. E. (2017). *Universidad Señor de Sipan*. Obtenido de Análisis y evaluación de las arquitecturas REST y SOAP para el desarrollo de servicios web aplicados al ERP AdrisERP y su versión móvil en Android: <https://repositorio.uss.edu.pe/handle/20.500.12802/4067>
- Swift. (2021). *Swift*. Obtenido de Automatic Reference Counting: <https://docs.swift.org/swift-book/LanguageGuide/AutomaticReferenceCounting.html>
- Telectrónica. (2006). *Introducción a la identificación por Radio Frecuencia-RFID*.
- Vanderkay, J. (18 de Marzo de 2004). *NFC Forum*. Recuperado el Noviembre de 2020, de Nokia, Philips And Sony Establish The Near Field Communication (NFC) Forum: <https://nfc-forum.org/nokia-philips-and-sony-establish-the-near-field-communication-nfc-forum/>
- Wells, D. M. (1994). *wayback machine*. Obtenido de A Trusted, Scalable, Real-Time Operating System Environment: <https://web.archive.org/web/20170822053715/https://pdfs.semantic-scholar.org/03ac/1296f530719497b49d7580b55a2d9b8353ab.pdf>

ANEXOS

ANEXO A

Código de la aplicación móvil

Modelos:

Brand.swift

```
import Foundation
struct Brand : Codable{
    let data : [BrandItems]
}
struct BrandItems : Codable{
    let id : String
    let name : String
}
```

Location.swift

```
import Foundation
struct Location : Codable{
    let data : [LocationItems]
}
struct LocationItems : Codable{
    let id : String
    let name : String
}
```

Categorie.swift

```
import Foundation
struct Categorie : Codable{
    let data : [CategorieItems]
}
struct CategorieItems : Codable{
    let id : String
    let name : String
}
```

Product.swift

```
import Foundation
struct Product: Codable {
    var status : String = ""
    var status_message : String = ""
}
```

CredentialPOST.swift

```
import Foundation
struct CredentialsPOST: Codable {
    var username : String = ""
    var name : String = ""
}
```

APIS:

NFCWriter.swift

```
import Foundation
import CoreNFC
import SwiftUI
class NFCWriter : NSObject, ObservableObject, NFCNDEFReaderSessionDelegate{
    var tag_id : String = ""
    var completeMessage : String = ""
    var dataProduct : Product?
    let now = Date()
    let formatter = DateFormatter()
    var last_maintenance : Date = Date()
    @Published var name: String = ""
    @Published var blankAlert : Bool = false
    @Published var serial_Code : String = ""
    @Published var administrador : String = ""
    @Published var product_status : String = ""
    @Published var features : String = ""
    @Published var categorieSelect : String = ""
    @Published var locationSelect : String = ""
    @Published var brandSelect : String = ""
    private var nfcSession : NFCNDEFReaderSession?
    private var nfcMessage: NFCNDEFMessage? = nil
    private var encoder : NFCMessageEncoder = NFCMessageEncoder()
    func cleanForm (){
        self.name = ""
        self.serial_Code = ""
        self.product_status = ""
        self.features = ""
        self.categorieSelect = ""
        self.locationSelect = ""
        self.brandSelect = ""
    }
    func formatTag(){
        self.formatter.dateFormat = "yyyy-MM-dd HH:mm:ss"
        let dateWithFormat = formatter.string(from: self.now)
        if(self.name != "" && self.administrador != "" && self.product_status != "" &&
self.features != "" && self.serial_Code != ""){
            var dataArray = [String]()
            dataArray.append("Administrador:" + "\(self.administrador)")
            dataArray.append("Nombre:" + "\(self.name)")
            dataArray.append("Serial:" + "\(self.serial_Code)")
        }
    }
}
```



```

class LocationAPI : ObservableObject{
    @Published var locaciones = [String]()
    func loadLocationPicker(){
        guard let requestUrl = URL( string:
"http://192.168.1.4:8080/inventario/webservice.php") else{
            print("URL no encontrada")
            return
        }
        let parameters : [String: Any] = ["code": "3"]
        var request = URLRequest(url: requestUrl)
        var components = URLComponents()
        var queryItems = [URLQueryItem]()
        for (key, value) in parameters {
            let queryItem = URLQueryItem(name: key, value: String(describing: value))
            queryItems.append(queryItem)
        }
        components.queryItems = queryItems
        let queryItemData = components.query?.data(using: .utf8)
        request.httpBody = queryItemData
        request.httpMethod = "POST"
        request.setValue("application/x-www-form-urlencoded", forHTTPHeaderField: "Content-Type")
        let session = URLSession.shared
        let task = session.dataTask(with: request){ (data,res,error) in
            if let unwrappedError = error {
                print("error", unwrappedError.localizedDescription)
                return
            }
            if let unwrappedData = data{
                do{
                    let json = try JSONSerialization.jsonObject(with: unwrappedData, options: []) print(json)
                    if let result = try? JSONDecoder().decode(Location.self, from: unwrappedData){
                        DispatchQueue.main.async {
                            for(_, item) in result.data.enumerated(){
                                self.locaciones.append(item.name)
                            }
                            self.locaciones = self.locaciones.uniqued()
                            print(self.locaciones)
                        }
                    }
                } catch(let errorJson){
                    print("error", errorJson.localizedDescription)
                }
            } else{
                print("No hay datos")
            }
        }
        task.resume()
    }
}

BrandAPI.swift
import Foundation
class BrandAPI : ObservableObject{
    @Published var marcas = [String]()
    func loadBrandPicker(){
        guard let requestUrl = URL( string:
"http://192.168.1.4:8080/inventario/webservice.php") else{
            print("URL no encontrada")
            return
        }
        let parameters : [String: Any] = ["code": "4"]
        var request = URLRequest(url: requestUrl)
        var components = URLComponents()
        var queryItems = [URLQueryItem]()
        for (key, value) in parameters {
            let queryItem = URLQueryItem(name: key, value: String(describing: value))
            queryItems.append(queryItem)
        }
        components.queryItems = queryItems
        let queryItemData = components.query?.data(using: .utf8)
        request.httpBody = queryItemData
        request.httpMethod = "POST"
        request.setValue("application/x-www-form-urlencoded", forHTTPHeaderField: "Content-Type")
        let session = URLSession.shared
        let task = session.dataTask(with: request){ (data,res,error) in
            if let unwrappedError = error {
                print("error", unwrappedError.localizedDescription)
                return
            }
            if let unwrappedData = data{
                do{
                    let json = try JSONSerialization.jsonObject(with: unwrappedData, options: []) print(json)
                    if let result = try? JSONDecoder().decode(Brand.self, from: unwrappedData){
                        DispatchQueue.main.async {
                            for(_, item) in result.data.enumerated(){
                                self.marcas.append(item.name)
                            }
                        }
                    }
                }
            }
        }
        task.resume()
    }
}

```

```

        self.marcas = self.marcas.uniqued()      print(self.marcas)
        }
    }
    }
    catch(let errorJson){
        print("error", errorJson.localizedDescription)
    }
    }
    else{
        print("No hay datos")
    }
}
task.resume()
}
}

UpdateProductAPI.swift
import Foundation
class UpdateProductAPI : ObservableObject{
    @Published var showProgressView : Bool = false
    @Published var blankData : Bool = false
    @Published var message = ""
    @Published var tag = ""
    func editDictionary(value: [String]){
        var dictionary : [String : String] = [:]
        var data = [String]()
        var Key : String = ""
        var Value : String = ""
        for i in 0..

```



```

NFCMessageDecoder.swift
import Foundation
import CoreNFC
class NFCMessageDecoder {
    func decode(_ message: NFCNDEFMessage) -> [String] {
        return message.records.compactMap { payload in
            return self.decode(payload)
        }
    }
    private func decode(_ payload: NFCNDEFPayload) -> String? {
        let type = PayloadType(rawValue: payload.type.decode())
        switch type {
        case .uri:
            return payload.wellKnownTypeURIPayload()?.absoluteString
        case .text:
            return payload.wellKnownTypeTextPayload().0
        default:
            return nil
        }
    }
}
extension Data {
    func decode() -> String {
        guard let value = String(data: self, encoding: .utf8) else {
            return ""
        }
        return value
    }
}
enum PayloadType: String {
    case text = "T"
    case uri = "U"
    case unknown
    init(rawValue: String) {
        switch rawValue {
        case "U": self = .uri
        case "T": self = .text
        default: self = .unknown
        }
    }
}
}
Vistas:
HomeView.swift
import SwiftUI
struct HomeView: View {
    @Environment(\.presentationMode) var presentation
    @State private var selectOptions: ChooseAction = .writer
    @State var nameFromUser : String = ""
    var body: some View {
        VStack{
            Picker("Menu Principal",selection: $selectOptions){
                ForEach(ChooseAction.allCases, id: \.self){
                    Text($0.rawValue)
                }
            }.pickerStyle(SegmentedPickerStyle())
                .padding()
            Spacer()
            ChosenView(selectAction: selectOptions)
            Spacer()
        }
        .navigationBarTitle("Menu Principal")
        .navigationBarTitleDisplayMode(.inline)
        .navigationBarBackButtonHidden(true)
        .navigationBarItems(leading: Button(action: {
            self.presentation.wrappedValue.dismiss()
        }) {
            HStack {
                Image(systemName: "person.fill.badge.minus")
                    .background(Color.red)
                    .foregroundColor(Color.white)
                Text("Logout")
                    .foregroundColor(Color.red)
            }
        })
    }
}
struct HomeView_Previews: PreviewProvider {
    static var previews: some View {
        HomeView()
    }
}
enum ChooseAction : String,CaseIterable{
    case writer = "Escriitor NFC"
    case reader = "Lector NFC"
}

```

```

}
struct ChosenView : View{
    var selectAction : ChooseAction
    var body: some View{
        switch selectAction {
            case .writer:
                //cambio de vistas
                NFCWriterView()
            case .reader:
                NFCReaderView()
        }
    }
}
struct OptionView : View{
    var option : String
    var body: some View{
        Text(option)
        .padding()
    }
}
struct NavigationConfigurator: UIViewControllerRepresentable {
    var configure: (UINavigationController) -> Void = { _ in }
    func makeUIViewController(context:
UIViewControllerRepresentableContext<NavigationConfigurator>) -> UIViewController {
        UIViewController()
    }
    func updateUIViewController(_ viewController: UIViewController, context:
UIViewControllerRepresentableContext<NavigationConfigurator>) {
        if let nc = viewController.navigationController {
            self.configure(nc)
        }
    }
}
LoginView.swift
import SwiftUI
struct LoginView: View {
    @StateObject var apiLogin: APIService = APIService()
    let lighGreyColor = Color(red: 239.0/255.0, green: 243.0/255.0, blue: 244.0/255.0)
    // inicio body
    var body: some View {
        // Inicio navigationView
        NavigationView{
            ZStack{
                //Color(red: 239.0, green:170.0 , blue: 170.0).edgesIgnoringSafeArea(.all)
                //Inicio VStack
                VStack{
                    Image("vdc")
                    .resizable()
                    .scaledToFit()
                    .frame(width: 150, height: 150)
                    .clipped()
                    //inicio segundo Vstack
                    VStack{
                        HStack{
                            Image(systemName: "person.crop.circle.fill")
                            TextField("Nombre de usuario", text: $apiLogin.username)
                            //.background(Color.white)
                        }
                        .padding()
                        .overlay(RoundedRectangle(cornerRadius: 1).stroke(lineWidth:
2).foregroundColor(Color.black))
                        .background(Color.white)
                        HStack{
                            Image(systemName: "lock")
                            .fixedSize()
                            SecureField("Password", text: $apiLogin.password)
                            //.background(Color.white)
                        }
                        .padding()
                        .overlay(RoundedRectangle(cornerRadius: 1).stroke(lineWidth:
2).foregroundColor(Color.black))
                        .background(Color.white)
                        Text("Login")
                        .frame(width: 150, height: 50)
                        .background(Color.red)
                        .foregroundColor(Color.white)
                        .cornerRadius(8)
                        .padding()
                    }
                    .onTapGesture {
                        if(apiLogin.username == "" || apiLogin.password == "")
                        {
                            apiLogin.showAlert = true
                            apiLogin.messageAlert = apiLogin.messageOption(option: "1")
                        }
                        else{
                            apiLogin.login()
                        }
                    }
                }
            }
        }
    }
}

```

```

        }
        if(apiLogin.showProgressView){
            NavigationLink(destination: HomeView(), isActive:
$apiLogin.showProgressView ){}
        }
    }
    //fin segundo vstack
    .padding()
    Spacer()
}
//fin VStack
}
.navigationBarTitle("Login", displayMode: .inline)
.alert(isPresented: $apiLogin.showAlert) Alert(title: Text("Lo sentimos"),
message: Text(apiLogin.messageAlert), dismissButton: .default(Text("OK")))
}
}
.navigationViewStyle(StackNavigationViewStyle())
.environmentObject(apiLogin)
//fin navigationView
}
//fin body
}
}
struct LoginView_Previews: PreviewProvider {
    static var previews: some View {
        LoginView()
    }
}
NFCWriterView.swift
import SwiftUI
import CoreNFC
struct NFCWriterView: View {
    let lighGreyColor = Color(red: 239.0/255.0, green: 243.0/255.0, blue: 244.0/255.0)
    @EnvironmentObject var api : APIService
    @ObservedObject var nfcWriter : NFCWriter = NFCWriter()
    @ObservedObject var categorieData : CategorieAPI = CategorieAPI()
    @ObservedObject var locationData : LocationAPI = LocationAPI()
    @ObservedObject var brandData : BrandAPI = BrandAPI()
    @State var selectionCategorie = 0
    @State var selectionLocation = 0
    @State var selectionBrand = 0
    var body: some View {
        //Vista de Lector de Tarjeta
        // inicio vstack
        VStack{
            //inicio form
            Form{
                Group{
                    TextField("Nombre de usuario", text: $nfcWriter.administrador) .disabled(true)
                    TextField("Nombre de Producto", text: $nfcWriter.name)
                    TextField("Serial", text: $nfcWriter.serial_Code)
                    TextField("Estado", text: $nfcWriter.product_status)
                    TextField("Caracteristicas", text: $nfcWriter.features)
                }
            }
            HStack{
                //inicio picker
                Text("Categoria")
                // .foregroundColor(Color.gray)
                Divider()
                Picker("Categoria",selection:$selectionCategorie){
                    ForEach(0 ..< categorieData.categorias.count , id: \.self){ index in
                        Text(self.categorieData.categorias[index]).tag(index)
                        // .foregroundColor(self.categorieData.obje)
                    }
                }
                //
                .pickerStyle(MenuPickerStyle())
            }
            //fin picker
            HStack{
                //inicio picker Locacion
                Text(" Locacion")
                Divider()
                // .foregroundColor(Color.gray)
                Picker("Locacion",selection:$selectionLocation){
                    ForEach(0 ..< locationData.locaciones.count , id: \.self){ index in
                        Text(self.locationData.locaciones[index]).tag(index)
                        // .foregroundColor(self.categorieData.obje)
                    }
                }
                //
                // .labelsHidden()
                .pickerStyle(MenuPickerStyle())
            }
            // fin picker locacion
        }
        HStack{
            //inicio picker marca

```



```

        Text("      Marca")
        Divider()
        //      .foregroundColor(Color.gray)
        Picker("Marca", selection:$selectionBrand ){
            ForEach(0 ..< brandData.marcas.count , id: \.self){ index in
                Text(self.brandData.marcas[index]).tag(index)
                //      .foregroundColor(self.categorieData.objje)
            }
        }
        //      .labelsHidden()
        .pickerStyle(MenuPickerStyle())
        //fin picker marca
    }
    HStack{
        Spacer()
        Button(action: {
            nfcWriter.categorieSelect =
categoriedata.categorias[selectionCategorie]
            nfcWriter.locationSelect =
locationData.locaciones[selectionLocation]
            nfcWriter.brandSelect = brandData.marcas[selectionBrand]
            nfcWriter.formatTag()
        }) {
            Text("Escribir")
        }
        .frame(width: 150, height: 50)
        //      .ali
        .background(Color.red)
        .foregroundColor(Color.white)
        .cornerRadius(8)
        Spacer()
    }
}
// fin de form

// }//fin scroll
Spacer()
}
.onAppear{
    categoriedata.loadCategoriePicker()
    locationData.loadLocationPicker()
    brandData.loadBrandPicker()
    nfcWriter.administrador = api.username
}
.alert(isPresented: $nfcWriter.blankAlert) {
    Alert(title: Text("Lo sentimos"), message: Text("Debe colocar los campos
completos"), dismissButton: .default(Text("OK")))
}
.padding()
Spacer()
// fin vstack
}
}
}
struct NFCWriterView_Previews: PreviewProvider {
    static var previews: some View {
        NFCWriterView()
    }
}
}
NFCReaderView.swift
import SwiftUI
struct NFCReaderView: View {
    let lighGreyColor = Color(red: 239.0/255.0, green: 243.0/255.0, blue: 244.0/255.0)
    @ObservedObject var nfcReader : NFCReader = NFCReader()
    @ObservedObject var nfcTag : NFCTagReader = NFCTagReader()
    @ObservedObject var product : UpdateProductAPI = UpdateProductAPI()
    var body: some View {
        /// Vista de Escritor de tarjeta
        VStack{
            Form{
                HStack{
                    Text("Tag ID")
                    Divider()
                    Text(nfcTag.tagUID)
                    .disabled(true)
                }
                TextEditor(text:$nfcReader.stringForText)
                .padding()
                HStack{
                    Spacer()
                    Button(action: {
                        nfcTag.scan()
                    }) {
                        Text("Obtener ID")
                    }
                }
                .frame(width: 150, height: 50)
                //      .ali

```



```

        break;
    default:
        header("HTTP/1.0 405 Method Not Allowed");
        break;
}
?>
Rest.php
<?php
ini_set('display_errors',1);
ini_set('display_startup_errors',1);
error_reporting(E_ALL);
class Rest
{
    private $host = 'localhost';
    private $user = 'root';
    private $password = "";
    private $database = 'oswa_inv';
    private $dbConnect = false;
    public function __construct()
    {
        if (!$this->dbConnect) {
            $conn = new mysqli($this->host, $this->user, $this->password, $this->database);
            if ($conn->connect_error) {
                die("Error failed to connect to MySQL: " . $conn->connect_error);
            } else {
                $this->dbConnect = $conn;
            }
        }
    }
    public function getUser($userData)
    {
        $usernameP = $userData["username"];
        $passwordP = $userData["password"];
        $userQuery = $this->dbConnect->prepare("SELECT username, password, name, user_level FROM users
WHERE username = ? AND password = ?");
        $userQuery->bind_param('ss',$usernameP,$passwordP);
        $userQuery->execute();
        $result = $userQuery->get_result();
        if($result->num_rows > 0){
            $row = $result->fetch_array(MYSQLI_ASSOC);
            $userLog = $row['username'];
            $nameUser = $row['name'];
        }
        else{
            $userLog = "none";
            $nameUser = "none";
        }
        $userResponse = array(
            'username' => $userLog,
            'name' => $nameUser
        );
        header('Content-Type: application/json');
        echo json_encode($userResponse);
    }
    public function getLocation(){
        $locationQuery = "SELECT * FROM locations";
        $result = $this->dbConnect->query($locationQuery);
        $rows = array();
        while ($r = mysqli_fetch_assoc($result)) {
            $rows['data'][] = $r;
        }
        header('Content-Type: application/json');
        echo json_encode($rows);
    }
    public function getBrand(){
        $brandQuery = "SELECT * FROM brand";
        $result = $this->dbConnect->query($brandQuery);
        $rows = array();
        while ($r = mysqli_fetch_assoc($result)) {
            $rows['data'][] = $r;
        }
        header('Content-Type: application/json');
        echo json_encode($rows);
    }
    public function getCategorie(){
        $categorieQuery = "SELECT * FROM categories";
        $result = $this->dbConnect->query($categorieQuery);
        $rows = array();
        while ($r = mysqli_fetch_assoc($result)) {
            $rows['data'][] = $r;
        }
        header('Content-Type: application/json');
        echo json_encode($rows);
    }
    public function getTagID($productData){

```

```

$tagID = $productData["tag_id"];
$tagQuery = $this->dbConnect->prepare("SELECT tag_id FROM products WHERE tag_id = ?");
$tagQuery->bind_param('s', $tagID);
$tagQuery->execute();
$result = $tagQuery->get_result();
if($result->num_rows >0)
{
    $found = true;
}
else {
    $found = false;
}
return $found;
}
public function getCategorieID($categorieData){

    $categorieIDQuery = "SELECT * FROM categories WHERE name = ?";
    $stmt = $this->dbConnect->prepare($categorieIDQuery);
    $stmt->bind_param('s', $categorieData['Categoria']);
    $stmt->execute();
    $result = $stmt->get_result();
    $row = $result->fetch_array(MYSQLI_ASSOC);
    return $row['id'];
}
public function getBrandID($brandData)
{
    $brandIDQuery = "SELECT * FROM brand WHERE name = ?";
    $stmt = $this->dbConnect->prepare($brandIDQuery);
    $stmt->bind_param('s', $brandData['Marca']);
    $stmt->execute();
    $result = $stmt->get_result();
    $row = $result->fetch_array(MYSQLI_ASSOC);
    return $row['id'];
}
public function getLocationID($locationData)
{
    $locationIDQuery = "SELECT * FROM locations WHERE name = ?";
    $stmt = $this->dbConnect->prepare($locationIDQuery);
    $stmt->bind_param('s', $locationData['Locacion']);
    $stmt->execute();
    $result = $stmt->get_result();
    $row = $result->fetch_array(MYSQLI_ASSOC);
    return $row['id'];
}
public function updateProduct($productDataParamU){
    $product_name = $productDataParamU['Nombre'];
    $administrador = $productDataParamU['Administrador'];
    $categorie_id = intval($this->getCategorieID($productDataParamU));
    $location_id = intval($this->getLocationID($productDataParamU));
    $brand_id = intval($this->getBrandID($productDataParamU));
    $serial_code = $productDataParamU['Serial'];
    $product_status = $productDataParamU['Estado'];
    $features = $productDataParamU['Caracteristicas'];
    $productDate = $productDataParamU['Fecha'];
    $tag = $productDataParamU['tag_id'];
    $queryUpdate = "UPDATE products SET name=?, administrador=?, categorie_id=?, location_id=?,
brand_id=?, serial_code=?, product_status= ?, features= ?, last_maintenance = ? WHERE tag_id=?";
    $queryProcess = $this->dbConnect->prepare($queryUpdate);
    $queryProcess->bind_param('ssiiii>ssss',
$product_name,$administrador,$categorie_id,$location_id,$brand_id, $serial_code,$product_status,
$features,$productDate,$tag);
    if ($queryProcess->execute()) {
        $status = "2";
        $messgae = "Registro Actualizado exitoso";
    } else {
        $status = "0";
        $messgae = "Fallo al Actualizar el registro";
    }
    $updateResponse = array(
        'status' => $status,
        'status_message' => $messgae
    );
    header('Content-Type: application/json');
    echo json_encode($updateResponse);
}
public function insertProduct($productDataParamI){
    $checkAvalaible = $this->getTagID($productDataParamI);
    switch ($checkAvalaible) {
        case true:
            $this->updateProduct($productDataParamI);
            break;
        case false:
            $product_name = $productDataParamI['Nombre'];
            $administrador = $productDataParamI['Administrador'];

```

```

        $quantity = floatval($productDataParamI['Cantidad']);
        $categoríe_id = intval($this->getCategoríeID( $productDataParamI));
        $location_id = intval($this->getLocationID( $productDataParamI));
        $brand_id = intval($this->getBrandID($productDataParamI));
        $tag_id = $productDataParamI['tag_id'];
        $serial_code = $productDataParamI['Serial'];
        $product_status = $productDataParamI['Estado'];
        $features = $productDataParamI['Características'];
        $productDate = $productDataParamI['Fecha'];
        $query = "INSERT INTO products SET
name=?,administrador=?,quantity=?,categoríe_id=?,location_id=?,brand_id=?,tag_id=?,serial_code=?,
product_status=?, features=?, date=?" ;
        $insertQuery = $this->dbConnect->prepare($query);
        $insertQuery->bind_param('ssdiissss',
$product_name,$administrador,$quantity,$categoríe_id,$location_id,$brand_id,$tag_id,$serial_code,$produ
ct_status,$features,$productDate);
        if($insertQuery->execute()){
            $status = "1";
            $messgae = "Registro Insertado exitoso";
        }
        else{
            $status = "0";
            $messgae = "Fallo al insertar registro";
        }
        $insertResponse = array(
            'status' => $status,
            'status_message' => $messgae
        );
        header('Content-Type: application/json');
        echo json_encode($insertResponse);
        break;
    default:
        break;
    }
}
}
}
?>

```

ANEXO C

Código de la aplicación Web

Inventario/Includes:

config.php

```

<?php
define('DB_HOST', 'localhost' ); // Host de la base
define('DB_USER', 'root' ); // Usuario de la base
define('DB_PASS', '' ); // Password de la base
define('DB_NAME', 'oswa_inv' ); // Nombre de la base
?>

```

database.php

```

<?php
require_once(LIB_PATH_INC.DS."config.php");
class MySqlI_DB {
    private $con;
    public $query_id;
    function __construct() {
        $this->db_connect();
    }
}
/*-----*/
/* Función para abrir la conexión de la base de datos
/*-----*/
public function db_connect()
{
    $this->con = mysqli_connect(DB_HOST,DB_USER,DB_PASS);
    if(!$this->con)
    {
        die(" Conexion a la base de datos fallo:". mysqli_connect_error());
    } else {
        $select_db = $this->con->select_db(DB_NAME);
        if(!$select_db)
        {
            die("Fallo al seleccionar base de datos". mysqli_connect_error());
        }
    }
}
/*-----*/
/* Función para cerrar la conexión a la base de datos
/*-----*/
public function db_disconnect()
{
    if(isset($this->con))
    {
        mysqli_close($this->con);
        unset($this->con);
    }
}
}

```

```

/*-----*/
/* Funcion para consulta mysqli
/*-----*/
public function query($sql)
{
    if (trim($sql != "")) {
        $this->query_id = $this->con->query($sql);
    }
    if (!$this->query_id)
        die("Error on this Query :<pre> " . $sql . "</pre>");
    return $this->query_id;
}
/*-----*/
/* Función para ayuda de consulta
/*-----*/
public function fetch_array($statement)
{
    return mysqli_fetch_array($statement);
}
public function fetch_object($statement)
{
    return mysqli_fetch_object($statement);
}
public function fetch_assoc($statement)
{
    return mysqli_fetch_assoc($statement);
}
public function num_rows($statement)
{
    return mysqli_num_rows($statement);
}
public function insert_id()
{
    return mysqli_insert_id($this->con);
}
public function affected_rows()
{
    return mysqli_affected_rows($this->con);
}
/*-----*/
/* Función para remover los caracteres especiales de escape en las
/* declaraciones de SQL
/*-----*/
public function escape($str){
    return $this->con->real_escape_string($str);
}
/*-----*/
/* Función bucle while
/*-----*/
public function while_loop($loop){
    global $db;
    $results = array();
    while ($result = $this->fetch_array($loop)) {
        $results[] = $result;
    }
    return $results;
}
}
$db = new MySQLi_DB();
?>
functions.php
<?php
$errors = array();
/*-----*/
/* Función para remover caracteres especiales de una cadena de
/* texto
/*-----*/
function real_escape($str){
    global $con;
    $escape = mysqli_real_escape_string($con,$str);
    return $escape;
}
/*-----*/
/* Función para remover caracteres HTML
/*-----*/
function remove_junk($str){
    $str = nl2br($str);
    $str = htmlspecialchars(strip_tags($str, ENT_QUOTES));
    return $str;
}
/*-----*/
/* Función para Volver mayúscula el primer carácter
/*-----*/
function first_character($str){

```

```

    $val = str_replace('-', " ", $str);
    $val = ucfirst($val);
    return $val;
}
/*-----*/
/* Función para chequear que no haya campos vacíos
/*-----*/
function validate_fields($var){
    global $errors;
    echo $var;
    foreach ($var as $field) {
        $val = remove_junk($_POST[$field]);
        if(isset($val) && $val==''){
            $errors = $field . " No pueden estar en blanco.";
            return $errors;
        }
    }
}
/*-----*/
/* Función para mostrar mensajes de sesión
/*-----*/
function display_msg($msg='') {
    $output = array();
    if(!empty($msg)) {
        foreach ($msg as $key => $value) {
            $output = "<div class=\"alert alert-{$key}\">>";
            $output .= "<a href=\"#\" class=\"close\" data-dismiss=\"alert\">&times;</a>";
            $output .= remove_junk(first_character($value));
            $output .= "</div>";
        }
        return $output;
    } else {
        return "" ;
    }
}
/*-----*/
/* Función para redireccionar
/*-----*/
function redirect($url, $permanent = false)
{
    if (headers_sent() === false)
    {
        header('Location: ' . $url, true, ($permanent === true) ? 301 : 302);
    }
    exit();
}
/*-----*/
/* Función para convertir el formato de la fecha
/*-----*/
function read_date($str){
    if($str)
        return date('F j, Y, g:i:s a', strtotime($str));
    else
        return null;
}
/*-----*/
/* Función para formato de fecha
/*-----*/
function make_date(){
    return strftime("%Y-%m-%d %H:%M:%S", time());
}
/*-----*/
/* Función para formato de fecha
/*-----*/
function count_id(){
    static $count = 1;
    return $count++;
}
/*-----*/
/* Función para crear cadena aleatoria
/*-----*/
function randString($length = 5)
{
    $str='';
    $cha = "0123456789abcdefghijklmnopqrstuvwxyz";
    for($x=0; $x<$length; $x++)
        $str .= $cha[mt_rand(0,strlen($cha))];
    return $str;
}
/*-----*/
/* Función para modificar fecha de bootstrap
/*-----*/
function read_date_parse($str){
    if($str)

```

```

        {$date= str_replace('/', '-', $str);
        return date('Y-m-d', strtotime($date));
    }
    else
        return null;
}
?>
load.php
<?php
// -----
// Definición de alias de los separadores
// -----
define("URL_SEPARATOR", '/');
define("DS", DIRECTORY_SEPARATOR);
// -----
// Definición de direcciones raiz
// -----
defined('SITE_ROOT')? null: define('SITE_ROOT', realpath(dirname(__FILE__)));
define("LIB_PATH_INC", SITE_ROOT.DS);
require_once(LIB_PATH_INC.'config.php');
require_once(LIB_PATH_INC.'functions.php');
require_once(LIB_PATH_INC.'session.php');
require_once(LIB_PATH_INC.'upload.php');
require_once(LIB_PATH_INC.'database.php');
require_once(LIB_PATH_INC.'sql.php');
?>
sesión.php
<?php
session_start();
class Session {
public $msg;
private $user_is_logged_in = false;
function __construct(){
    $this->flash_msg();
    $this->userLoginSetup();
}
public function isUserLoggedIn(){
    return $this->user_is_logged_in;
}
public function login($user_id){
    $_SESSION['user_id'] = $user_id;
}
private function userLoginSetup()
{
    if(isset($_SESSION['user_id']))
    {
        $this->user_is_logged_in = true;
    } else {
        $this->user_is_logged_in = false;
    }
}
public function logout(){
    unset($_SESSION['user_id']);
}
public function msg($type = '', $msg = ''){
    if(!empty($msg)){
        if(strlen(trim($type)) == 1){
            $type = str_replace( array('d', 'i', 'w', 's'), array('peligro', 'info',
'advertencia', 'exito'), $type );
        }
        $_SESSION['msg'][$type] = $msg;
    } else {
        return $this->msg;
    }
}
private function flash_msg(){
    if(isset($_SESSION['msg'])) {
        $this->msg = $_SESSION['msg'];
        unset($_SESSION['msg']);
    } else {
        $this->msg;
    }
}
}
}
$session = new Session();
$msg = $session->msg();
?>
sql.php
<?php
require_once('includes/load.php');
/*-----*/
/* Función para traer todos los datos de una tabla
/*-----*/
function find_all($table) {

```



```

global $db;
if(tableExists($table))
{
    return find_by_sql("SELECT * FROM ".$db->escape($table));
}
}
/*-----*/
/* Función para realizar consultas
/*-----*/
function find_by_sql($sql)
{
    global $db;
    $result = $db->query($sql);
    $result_set = $db->while_loop($result);
    return $result_set;
}
/*-----*/
/* Función para encontrar los datos de la tabla por Id
/*-----*/
function find_by_id($table,$id)
{
    global $db;
    $id = (int)$id;
    if(tableExists($table)){
        $sql = $db->query("SELECT * FROM {$db->escape($table)} WHERE id='{$db->escape($id)}' LIMIT
1");
        if($result = $db->fetch_assoc($sql))
            return $result;
        else
            return null;
    }
}
/*-----*/
/* Función para borrar los datos de una tabla por ID
/*-----*/
function delete_by_id($table,$id)
{
    global $db;
    if(tableExists($table))
    {
        $sql = "DELETE FROM ".$db->escape($table);
        $sql .= " WHERE id=".$db->escape($id);
        $sql .= " LIMIT 1";
        $db->query($sql);
        return ($db->affected_rows() === 1) ? true : false;
    }
}
/*-----*/
/* Función para contar el Id por nombre de tabla
/*-----*/
function count_by_id($table){
    global $db;
    if(tableExists($table))
    {
        $sql = "SELECT COUNT(id) AS total FROM ".$db->escape($table);
        $result = $db->query($sql);
        return($db->fetch_assoc($result));
    }
}
/*-----*/
/* Verificar si la base de datos existe
/*-----*/
function tableExists($table){
    global $db;
    $table_exit = $db->query('SHOW TABLES FROM '.DB_NAME.' LIKE "'.$db->escape($table)."'');
    if($table_exit) {
        if($db->num_rows($table_exit) > 0)
            return true;
        else
            return false;
    }
}
/*-----*/
/* Ingresar con los datos que vienen desde POST
/* por medio del formulario de ingreso
/*-----*/
function authenticate($username='', $password='') {
    global $db;
    $username = $db->escape($username);
    $password = $db->escape($password);
    $sql = sprintf("SELECT id,username,password,user_level FROM users WHERE username = '%s' LIMIT 1",
$username);
    $result = $db->query($sql);
    if($db->num_rows($result)){

```

```

        $user = $db->fetch_assoc($result);
        $password_request = sha1($password);
        if($password_request === $user['password'] ){
            return $user['id'];
        }
    }
    return false;
}
/*-----*/
/* Buscar el usuario que está conectado
/*-----*/
function current_user(){
    static $current_user;
    global $db;
    if(!$current_user){
        if(isset($_SESSION['user_id'])){
            $user_id = intval($_SESSION['user_id']);
            $current_user = find_by_id('users',$user_id);
        }
        endif;
    }
    return $current_user;
}
/*-----*/
/* Encontrar todos los usuarios
/*-----*/
function find_all_user(){
    global $db;
    $results = array();
    $sql = "SELECT u.id,u.name,u.username,u.user_level,u.status,u.last_login,";
    $sql .= "g.group_name ";
    $sql .= "FROM users u ";
    $sql .= "LEFT JOIN user_groups g ";
    $sql .= "ON g.group_level=u.user_level ORDER BY u.name ASC";
    $result = find_by_sql($sql);
    return $result;
}
/*-----*/
/* Función para actualizar la última conexión del usuario
/*-----*/

function updateLastLogIn($user_id)
{
    global $db;
    $date = make_date();
    $sql = "UPDATE users SET last_login='{$date}' WHERE id='{$user_id}' LIMIT 1";
    $result = $db->query($sql);
    return ($result && $db->affected_rows() === 1 ? true : false);
}
/*-----*/
/* Encontrar todos los nombres de los grupos
/*-----*/
function find_by_groupName($val)
{
    global $db;
    $sql = "SELECT group_name FROM user_groups WHERE group_name = '{$db->escape($val)}' LIMIT 1 ";
    $result = $db->query($sql);
    return($db->num_rows($result) === 0 ? true : false);
}
/*-----*/
/* Encontrar nivel de grupo
/*-----*/
function find_by_groupLevel($level)
{
    global $db;
    $sql = "SELECT group_level FROM user_groups WHERE group_level = '{$db->escape($level)}' LIMIT 1 ";
    $result = $db->query($sql);
    return($db->num_rows($result) === 0 ? true : false);
}
/*-----*/
/* Permisos de usuario por página
/*-----*/
function page_require_level($require_level){
    global $session;
    $current_user = current_user();
    $login_level = find_by_groupLevel($current_user['user_level']);
    if (!$session->isUserLoggedIn(true)):
        $session->msg('d','Please login...');
        redirect('index.php', false);
    elseif($login_level['group_status'] === '0'):
        $session->msg('d','This level user has been band!');
        redirect('home.php', false);
    elseif($current_user['user_level'] <= (int)$require_level):
        return true;
    else: $session->msg("d", "Sorry! you dont have permission to view the page.");
}

```

```

        redirect('home.php', false);
    endif;
}
/*-----*/
/* Encontrar todos los nombres de productos
/*-----*/
function join_product_table(){
    global $db;
    $sql = " SELECT p.id,p.name,p.quantity,p.buy_price,p.sale_price,p.date,p.media_id,p.purchase_date
AS fechacompra,p.last_maintenance AS fechamantenimiento,p.administrador AS creador,p.serial_code AS
serialcode,";
    $sql .= " c.name AS categoria,m.file_name AS imagen, l.name AS locacion, b.name AS marca";
    $sql .= " FROM products p";
    $sql .= " LEFT JOIN categories c ON c.id = p.categorie_id";
    $sql .= " LEFT JOIN media m ON m.id = p.media_id";
    $sql .= " LEFT JOIN brand b ON b.id = p.brand_id";
    $sql .= " LEFT JOIN locations l ON l.id = p.location_id";
    $sql .= " ORDER BY p.id ASC";
    return find_by_sql($sql);
}
/*-----*/
/* Encontrar todos los nombres de productos sugeridos
/*-----*/
function find_product_by_title($product_name){
    global $db;
    $p_name = remove_junk($db->escape($product_name));
    $sql = "SELECT name FROM products WHERE name like '%$p_name%' LIMIT 5";
    $result = find_by_sql($sql);
    return $result;
}
/*-----*/
/* Encontrar toda la información del producto por nombre del mismo
/*-----*/
function find_all_product_info_by_title($title){
    global $db;
    $sql = "SELECT * FROM products ";
    $sql .= " WHERE name ='{$title}'";
    $sql .= " LIMIT 1";
    return find_by_sql($sql);
}
/*-----*/
/* Actualizar cantidad de producto
/*-----*/
function update_product_qty($qty,$p_id){
    global $db;
    $qty = (int) $qty;
    $id = (int)$p_id;
    $sql = "UPDATE products SET quantity=quantity -'{$qty}' WHERE id = '{$id}'";
    $result = $db->query($sql);
    return($db->affected_rows() === 1 ? true : false);
}
/*-----*/
/* Mostrar productos recientemente agregados
/*-----*/
function find_recent_product_added($limit){
    global $db;
    $sql = " SELECT p.id,p.name,p.sale_price,p.media_id,c.name AS categorie,";
    $sql .= " m.file_name AS image FROM products p";
    $sql .= " LEFT JOIN categories c ON c.id = p.categorie_id";
    $sql .= " LEFT JOIN media m ON m.id = p.media_id";
    $sql .= " ORDER BY p.id DESC LIMIT ".$db->escape((int)$limit);
    return find_by_sql($sql);
}
}
?>

```

Inventario/layouts:

admin_menu.php

```

<ul>
<li>
<a href="admin.php">
<i class="glyphicon glyphicon-home"></i>
<span>Inicio</span>
</a>
</li>
<li>
<a href="#" class="submenu-toggle">
<i class="glyphicon glyphicon-user"></i>
<span>Administracion</span>
</a>
<ul class="nav submenu">
<li><a href="group.php">Administrar grupos</a> </li>
<li><a href="users.php">Administrar Usuarios</a> </li>
</ul>
</li>
<li>

```

```

        <a href="categor\u00ede.php">
            <i class="glyphicon glyphicon-indent-left"></i>
            <span>Categor\u00edas</span>
        </a>
    </li>
    <li>
    <li>
        <a href="brand.php">
            <i class="glyphicon glyphicon-indent-left"></i>
            <span>Marcas</span>
        </a>
    </li>
    <li>
    <li>
        <a href="location.php">
            <i class="glyphicon glyphicon-indent-left"></i>
            <span>Locaciones</span>
        </a>
    </li>
    <li>
        <a href="media.php">
            <i class="glyphicon glyphicon-picture"></i>
            <span>Media</span>
        </a>
    </li>
    <li>
        <a href="#" class="submenu-toggle">
            <i class="glyphicon glyphicon-th-large"></i>
            <span>Bienes</span>
        </a>
        <ul class="nav submenu">
            <li><a href="product.php">Administrar bienes</a> </li>
            <li><a href="add_product.php">Agregar Bienes</a> </li>
        </ul>
    </li>
</ul>
</div>
</div>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.2/jquery.min.js"></script>
<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.4/js/bootstrap.min.js"></script>
<script src="//cdnjs.cloudflare.com/ajax/libs/bootstrap-datepicker/1.3.0/js/bootstrap-
datepicker.min.js"></script>
<script type="text/javascript" src="libs/js/functions.js"></script>
</body>
</html>
<?php if(isset($db)) { $db->db_disconnect(); } ?>
header.php
<?php $user = current_user(); ?>
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="UTF-8">
        <title><?php if (!empty($page_title))
            echo remove_junk($page_title);
            elseif(!empty($user))
            echo ucfirst($user['name']);
            else echo "Sistema simple de inventario";?>
        </title>
        <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.4/css/bootstrap.min.css"/>
        <link rel="stylesheet" href="//cdnjs.cloudflare.com/ajax/libs/bootstrap-
datepicker/1.3.0/css/datepicker3.min.css" />
        <link rel="stylesheet" href="libs/css/main.css" />
    </head>
    <body>
        <?php if ($session->isUserLoggedIn(true)): ?>
            <header id="header">
                <div class="logo pull-left"> UNEVIC INVENTARIO </div>
                <div class="header-content">
                <div class="header-date pull-left">
                    <strong><?php echo date("F j, Y, g:i a");?></strong>
                </div>
                <div class="pull-right clearfix">
                    <ul class="info-menu list-inline list-unstyled">
                        <li class="profile">
                            <a href="#" data-toggle="dropdown" class="toggle" aria-expanded="false">
                                
                                    <span><?php echo remove_junk(ucfirst($user['name'])); ?> <i class="caret"></i></span>
                                </a>
                                <ul class="dropdown-menu">
                                    <li>
                                        <a href="profile.php?id=<?php echo (int)$user['id'];?>">
                                            <i class="glyphicon glyphicon-user"></i>

```

```

        Perfil
    </a>
</li>
<li>
    <a href="edit_account.php" title="edit account">
        <i class="glyphicon glyphicon-cog"></i>
        Configuracion
    </a>
</li>
<li class="last">
    <a href="logout.php">
        <i class="glyphicon glyphicon-off"></i>
        Salir
    </a>
</li>
</ul>
</li>
</ul>
</div>
</div>
</header>
<div class="sidebar">
    <?php if($user['user_level'] === '1'): ?>
    <?php include_once('admin_menu.php');?>
    <?php elseif($user['user_level'] === '2'): ?>
    <?php include_once('special_menu.php');?>
    <?php elseif($user['user_level'] === '3'): ?>
    <?php include_once('user_menu.php');?>
    <?php endif;?>
</div>
<?php endif;?>
<div class="page">
    <div class="container-fluid">
special_menu.php
<ul>
    <li>
        <a href="home.php">
            <i class="glyphicon glyphicon-home"></i>
            <span>Inicio</span>
        </a>
    </li>
    <li>
        <a href="categorie.php" >
            <i class="glyphicon glyphicon-indent-left"></i>
            <span>Categoria</span>
        </a>
    </li>
    <li>
        <a href="#" class="submenu-toggle">
            <i class="glyphicon glyphicon-th-large"></i>
            <span>Bienes</span>
        </a>
        <ul class="nav submenu">
            <li><a href="product.php">Administrar bienes</a> </li>
            <li><a href="add_product.php">Agregar Bienes</a> </li>
        </ul>
    </li>
    <li>
        <a href="media.php" >
            <i class="glyphicon glyphicon-picture"></i>
            <span>Media</span>
        </a>
    </li>
</ul>
Inventario:
auth.php
<?php include_once('includes/load.php'); ?>
<?php
$req_fields = array('username', 'password' );
validate_fields($req_fields);
$username = remove_junk($_POST['username']);
$password = remove_junk($_POST['password']);
if(empty($errors)){
    $user_id = authenticate($username, $password);
    if($user_id){
        $session->login($user_id);
        updateLastLogIn($user_id);
        $session->msg("s", "Bienvenido al sistema de inventarios.");
        redirect('home.php', false);
    } else {
        $session->msg("d", "Password o contrasena incorrectos.");
        redirect('index.php', false);
    }
}

```

```

} else {
    $session->msg("d", $errors);
    redirect('index.php',false);
}
?>
add_group.php
<?php
    $page_title = 'Agregar grupo';
    require_once('includes/load.php');
    page_require_level(1);
?>
<?php
if(isset($_POST['add'])){
    $req_fields = array('group-name','group-level');
    validate_fields($req_fields);
    if(find_by_groupName($_POST['group-name']) === false ){
        $session->msg('d','<b>Lo siento!</b> Nombre duplicado en la base de datos!');
        redirect('add_group.php', false);
    }elseif(find_by_groupLevel($_POST['group-level']) === false) {
        $session->msg('d','<b>Lo siento!</b> Nivel de grupo duplicado en la base de datos!');
        redirect('add_group.php', false);
    }
    if(empty($errors)){
        $name = remove_junk($db->escape($_POST['group-name']));
        $level = remove_junk($db->escape($_POST['group-level']));
        $status = remove_junk($db->escape($_POST['status']));
        $query = "INSERT INTO user_groups (";
        $query .= "group_name,group_level,group_status";
        $query .= ") VALUES (";
        $query .= "'{$name}', '{$level}', '{$status}'";
        $query .= ")";
        if($db->query($query)){
            $session->msg('s',"Grupo ha sido creado! ");
            redirect('add_group.php', false);
        } else {
            $session->msg('d',' fallo al crear el grupo!');
            redirect('add_group.php', false);
        }
    }
} else {
    $session->msg("d", $errors);
    redirect('add_group.php',false);
}
?>
<?php include_once('layouts/header.php'); ?>
<div class="login-page">
    <div class="text-center">
        <h3>Agregar nuevo grupo usuario</h3>
    </div>
    <?php echo display_msg($msg); ?>
    <form method="post" action="add_group.php" class="clearfix">
        <div class="form-group">
            <label for="name" class="control-label">Nombre grupo</label>
            <input type="name" class="form-control" name="group-name">
        </div>
        <div class="form-group">
            <label for="level" class="control-label">Nivel de grupo</label>
            <input type="number" class="form-control" name="group-level">
        </div>
        <div class="form-group">
            <label for="status">Status</label>
            <select class="form-control" name="status">
                <option value="1">Activo</option>
                <option value="0">Inactivo</option>
            </select>
        </div>
        <div class="form-group clearfix">
            <button type="submit" name="add" class="btn btn-info">Actualizar</button>
        </div>
    </form>
</div>
<?php include_once('layouts/footer.php'); ?>
add_user.php
<?php
    $page_title = 'Agregar Usuario';
    require_once('includes/load.php');
    // Checkin What level user has permission to view this page
    page_require_level(1);
    $groups = find_all('user_groups');
?>
<?php
if(isset($_POST['add_user'])){
    $req_fields = array('full-name','username','password','level' );
    validate_fields($req_fields);

```

```

if(empty($errors)){
    $name = remove_junk($db->escape($_POST['full-name']));
    $username = remove_junk($db->escape($_POST['username']));
    $password = remove_junk($db->escape($_POST['password']));
    $user_level = (int)$db->escape($_POST['level']);
    $password = sha1($password);
    $query = "INSERT INTO users (";
    $query .= "name,username,password,user_level,status";
    $query .= ") VALUES (";
    $query .= "'{$name}', '{$username}', '{$password}', '{$user_level}','1'";
    $query .= ")";
    if($db->query($query)){
        //success
        $session->msg('s',"Cuenta ha sido creada! ");
        redirect('add_user.php', false);
    } else {
        //failed
        $session->msg('d',' No se pudo crear la cuenta!');
        redirect('add_user.php', false);
    }
} else {
    $session->msg("d", $errors);
    redirect('add_user.php',false);
}
}
?>
<?php include_once('layouts/header.php'); ?>
<?php echo display_msg($msg); ?>
<div class="row">
    <div class="panel panel-default">
        <div class="panel-heading">
            <strong>
                <span class="glyphicon glyphicon-th"></span>
                <span>Agregar nuevo usuario</span>
            </strong>
        </div>
        <div class="panel-body">
            <div class="col-md-6">
                <form method="post" action="add_user.php">
                    <div class="form-group">
                        <label for="name">Nombre</label>
                        <input type="text" class="form-control" name="full-name" placeholder="nombre completo">
                    </div>
                    <div class="form-group">
                        <label for="username">Nombre de Usuario</label>
                        <input type="text" class="form-control" name="username" placeholder="nombre de usuario">
                    </div>
                    <div class="form-group">
                        <label for="password">Password</label>
                        <input type="password" class="form-control" name="password" placeholder="Password">
                    </div>
                    <div class="form-group">
                        <label for="level">Rol de Usuario</label>
                        <select class="form-control" name="level">
                            <?php foreach ($groups as $group ):?>
                                <option value="<?php echo $group['group_level'];?>"><?php echo
ucwords($group['group_name']);?></option>
                            <?php endforeach;?>
                        </select>
                    </div>
                    <div class="form-group clearfix">
                        <button type="submit" name="add_user" class="btn btn-primary">Agregar usuario</button>
                    </div>
                </form>
            </div>
        </div>
    </div>
</div>
<?php include_once('layouts/footer.php'); ?>
admin.php
<?php
    $page_title = 'Admin pagina';
    require_once('includes/load.php');
    page_require_level(1);
?>
<?php
    $c_categorie = count_by_id('categories');
    $c_product = count_by_id('products');
    $c_sale = count_by_id('sales');
    $c_user = count_by_id('users');
    $products_sold = find_higest_saleing_product('10');
    $recent_products = find_recent_product_added('5');
    $recent_sales = find_recent_sale_added('5')

```

```

?>
<?php include_once('layouts/header.php'); ?>

<div class="row">
  <div class="col-md-6">
    <?php echo display_msg($msg); ?>
  </div>
</div>
<div class="row">
  <div class="col-md-3">
    <div class="panel panel-box clearfix">
      <div class="panel-icon pull-left bg-green">
        <i class="glyphicon glyphicon-user"></i>
      </div>
      <div class="panel-value pull-right">
        <h2 class="margin-top"> <?php echo $c_user['total']; ?> </h2>
        <p class="text-muted">Usuarios</p>
      </div>
    </div>
  </div>
  <div class="col-md-3">
    <div class="panel panel-box clearfix">
      <div class="panel-icon pull-left bg-blue">
        <i class="glyphicon glyphicon-shopping-cart"></i>
      </div>
      <div class="panel-value pull-right">
        <h2 class="margin-top"> <?php echo $c_product['total']; ?> </h2>
        <p class="text-muted">Bienes</p>
      </div>
    </div>
  </div>
</div>
<div class="row">
  <div class="col-md-12">
    <div class="panel">
      <div class="jumbotron text-center">
        <h1>Resumen del Sistema de Inventarios</h1>
      </div>
    </div>
  </div>
</div>
<div class="row">
  <div class="col-md-4">
    <div class="panel panel-default">
      <div class="panel-heading">
        <strong>
          <span class="glyphicon glyphicon-th"></span>
          <span>Ultimos bienes agregados</span>
        </strong>
      </div>
      <div class="panel-body">
        <div class="list-group">
          <?php foreach ($recent_products as $recent_product): ?>
            <a class="list-group-item clearfix" href="edit_product.php?id=<?php
echo (int)$recent_product['id'];?>">
              <h4 class="list-group-item-heading">
                <?php if($recent_product['media_id'] === '0'): ?>
                  
                <?php else: ?>
                  
                <?php endif;?>
                <?php echo remove_junk(first_character($recent_product['name']));?>
                <span class="label label-warning pull-right">
                  $<?php echo (int)$recent_product['sale_price']; ?>
                </span>
              </h4>
              <span class="list-group-item-text pull-right">
                <?php echo remove_junk(first_character($recent_product['categorie'])); ?>
              </span>
            </a>
          <?php endforeach; ?>
        </div>
      </div>
    </div>
  </div>
</div>
<div class="row">
</div>
<?php include_once('layouts/footer.php'); ?>
ajax.php
<?php
require_once('includes/load.php');
if (!$session->isUserLoggedIn(true)) { redirect('index.php', false);}

```



```

?>
<?php
    $html = '';
    if(isset($_POST['product_name']) && strlen($_POST['product_name']))
    {
        $products = find_product_by_title($_POST['product_name']);
        if($products){
            foreach ($products as $product):
                $html .= "<li class='list-group-item'>";
                $html .= $product['name'];
                $html .= "</li>";
            endforeach;
        } else {
            $html .= "<li onClick='fill(\"'.addslashes().'\')' class='list-group-item'>";
            $html .= 'Not found';
            $html .= "</li>";
        }
        echo json_encode($html);
    }
?>
<?php
    if(isset($_POST['p_name']) && strlen($_POST['p_name']))
    {
        $product_title = remove_junk($db->escape($_POST['p_name']));
        if($results = find_all_product_info_by_title($product_title)){
            foreach ($results as $result) {
                $html .= "<tr>";
                $html .= "<td id='s_name'>".$result['name'].</td>";
                $html .= "<input type='hidden' name='s_id' value='{ $result['id'] }'>";
                $html .= "<td>";
                $html .= "<input type='text' class='form-control' name='price'
value='{ $result['sale_price'] }'>";
                $html .= "</td>";
                $html .= "<td id='s_qty'>";
                $html .= "<input type='text' class='form-control' name='quantity' value='1'>";
                $html .= "</td>";
                $html .= "<td>";
                $html .= "<input type='text' class='form-control' name='total'
value='{ $result['sale_price'] }'>";
                $html .= "</td>";
                $html .= "<td>";
                $html .= "<input type='date' class='form-control datePicker' name='date' data-date
data-date-format='yyyy-mm-dd'>";
                $html .= "</td>";
                $html .= "<td>";
                $html .= "<button type='submit' name='add_sale' class='btn btn-primary'>Add
sale</button>";
                $html .= "</td>";
                $html .= "</tr>";
            }
        } else {
            $html = "<tr><td>product name not resgister in database</td></tr>";
        }
        echo json_encode($html);
    }
?>
brand.php
<?php
$page_title = 'Todas las marcas';
require_once('includes/load.php');
page_require_level(1);
    $all_brands = find_all('brand')
?>
<?php
    if (isset($_POST['add_brand'])) {
        $req_field = array('brand-name');
        validate_fields($req_field);
        $brand_name = remove_junk($db->escape($_POST['brand-name']));
        if (empty($errors)) {
            $sql = "INSERT INTO brand (name)";
            $sql .= " VALUES ('{$brand_name}')";
            if ($db->query($sql) {
                $session->msg("s", "Marca agregada exitosamente");
                redirect('brand.php', false);
            } else {
                $session->msg("d", "Fallo al crear marca.");
                redirect('brand.php', false);
            }
        } else {
            $session->msg("d", $errors);
            redirect('brand.php', false);
        }
    }
}
?>

```

```

<?php include_once('layouts/header.php'); ?>
<div class="row">
  <div class="col-md-12">
    <?php echo display_msg($msg); ?>
  </div>
</div>
<div class="row">
  <div class="col-md-5">
    <div class="panel panel-default">
      <div class="panel-heading">
        <strong>
          <span class="glyphicon glyphicon-th"></span>
          <span>Agregar nueva Marca</span>
        </strong>
      </div>
      <div class="panel-body">
        <form method="post" action="brand.php">
          <div class="form-group">
            <input type="text" class="form-control" name="brand-name" placeholder="Nombre de marca">
          </div>
          <button type="submit" name="add_brand" class="btn btn-primary">Agregar marca</button>
        </form>
      </div>
    </div>
  </div>
  <div class="col-md-7">
    <div class="panel panel-default">
      <div class="panel-heading">
        <strong>
          <span class="glyphicon glyphicon-th"></span>
          <span>Todas las marcas</span>
        </strong>
      </div>
      <div class="panel-body">
        <table class="table table-bordered table-striped table-hover">
          <thead>
            <tr>
              <th class="text-center" style="width: 50px;">#</th>
              <th>Marcas</th>
              <th class="text-center" style="width: 100px;">Acciones</th>
            </tr>
          </thead>
          <tbody>
            <?php foreach ($all_brands as $brand) : ?>
              <tr>
                <td class="text-center"><?php echo count_id(); ?></td>
                <td><?php echo remove_junk(ucfirst($brand['name'])); ?></td>
                <td class="text-center">
                  <div class="btn-group">
                    <a href="edit_brand.php?id=<?php echo (int)$brand['id']; ?>" class="btn btn-xs btn-warning" data-toggle="tooltip" title="Edit">
                      <span class="glyphicon glyphicon-edit"></span>
                    </a>
                    <a href="delete_brand.php?id=<?php echo (int)$brand['id']; ?>"
                      class="btn btn-xs btn-danger" data-toggle="tooltip" title="Remove">
                      <span class="glyphicon glyphicon-trash"></span>
                    </a>
                  </div>
                </td>
              </tr>
            <?php endforeach; ?>
          </tbody>
        </table>
      </div>
    </div>
  </div>
</div>
</div>
<?php include_once('layouts/footer.php'); ?>
categorie.php
<?php
$page_title = 'Todas las categorias';
require_once('includes/load.php');
page_require_level(1);
$all_categories = find_all('categories')
?>
<?php
if(isset($_POST['add_cat'])){
  $req_field = array('categorie-name');
  validate_fields($req_field);
  $cat_name = remove_junk($db->escape($_POST['categorie-name']));
  if(empty($errors)){
    $sql = "INSERT INTO categorias (name)";
    $sql .= " VALUES ('{$cat_name}')";
  }
}

```

```

        if($db->query($sql){
            $session->msg("s", "Categoria agregada exitosamente");
            redirect('categoriae.php',false);
        } else {
            $session->msg("d", "Fallo al crear categoria.");
            redirect('categoriae.php',false);
        }
    } else {
        $session->msg("d", $errors);
        redirect('categoriae.php',false);
    }
}
?>
<?php include_once('layouts/header.php'); ?>
<div class="row">
    <div class="col-md-12">
        <?php echo display_msg($msg); ?>
    </div>
</div>
<div class="row">
    <div class="col-md-5">
        <div class="panel panel-default">
            <div class="panel-heading">
                <strong>
                    <span class="glyphicon glyphicon-th"></span>
                    <span>Agregar nueva categoria</span>
                </strong>
            </div>
            <div class="panel-body">
                <form method="post" action="categoriae.php">
                    <div class="form-group">
                        <input type="text" class="form-control" name="categoriae-name" placeholder="Nombre de categoria">
                    </div>
                    <button type="submit" name="add_cat" class="btn btn-primary">Agregar categoria</button>
                </form>
            </div>
        </div>
    </div>
    <div class="col-md-7">
        <div class="panel panel-default">
            <div class="panel-heading">
                <strong>
                    <span class="glyphicon glyphicon-th"></span>
                    <span>Todas las categorias</span>
                </strong>
            </div>
            <div class="panel-body">
                <table class="table table-bordered table-striped table-hover">
                    <thead>
                        <tr>
                            <th class="text-center" style="width: 50px;">#</th>
                            <th>Categorias</th>
                            <th class="text-center" style="width: 100px;">Acciones</th>
                        </tr>
                    </thead>
                    <tbody>
                        <?php foreach ($all_categories as $cat):?>
                            <tr>
                                <td class="text-center"><?php echo count_id();?></td>
                                <td><?php echo remove_junk(ucfirst($cat['name'])); ?></td>
                                <td class="text-center">
                                    <div class="btn-group">
                                        <a href="edit_categoriae.php?id=<?php echo (int)$cat['id'];?>" class="btn btn-
xs btn-warning" data-toggle="tooltip" title="Edit">
                                            <span class="glyphicon glyphicon-edit"></span>
                                        </a>
                                        <a href="delete_categoriae.php?id=<?php echo (int)$cat['id'];?>" class="btn
btn-xs btn-danger" data-toggle="tooltip" title="Remove">
                                            <span class="glyphicon glyphicon-trash"></span>
                                        </a>
                                    </div>
                                </td>
                            </tr>
                        <?php endforeach; ?>
                    </tbody>
                </table>
            </div>
        </div>
    </div>
</div>
<?php include_once('layouts/footer.php'); ?>
location.php
<?php

```

```

$page_title = 'Todas las locaciones';
require_once('includes/load.php');
page_require_level(1);
'all_locations = find_all('locations')
?>
<?php
if (isset($_POST['add_loc'])) {
    $req_field = array('location-name');
    validate_fields($req_field);
    $loc_name = remove_junk($db->escape($_POST['location-name']));
    if (empty($errors)) {
        $sql = "INSERT INTO locations (name)";
        $sql .= " VALUES ('{$loc_name}')";
        if ($db->query($sql)) {
            $session->msg("s", "Locacion agregada exitosamente");
            redirect('location.php', false);
        } else {
            $session->msg("d", "Fallo al crear locacion.");
            redirect('location.php', false);
        }
    } else {
        $session->msg("d", $errors);
        redirect('location.php', false);
    }
}
?>
<?php include_once('layouts/header.php'); ?>
<div class="row">
    <div class="col-md-12">
        <?php echo display_msg($msg); ?>
    </div>
</div>
<div class="row">
    <div class="col-md-5">
        <div class="panel panel-default">
            <div class="panel-heading">
                <strong>
                    <span class="glyphicon glyphicon-th"></span>
                    <span>Agregar nueva locacion</span>
                </strong>
            </div>
            <div class="panel-body">
                <form method="post" action="location.php">
                    <div class="form-group">
                        <input type="text" class="form-control" name="location-name" placeholder="Nombre de locacion">
                    </div>
                    <button type="submit" name="add_loc" class="btn btn-primary">Agregar Locacion</button>
                </form>
            </div>
        </div>
    </div>
    <div class="col-md-7">
        <div class="panel panel-default">
            <div class="panel-heading">
                <strong>
                    <span class="glyphicon glyphicon-th"></span>
                    <span>Todas las locaciones</span>
                </strong>
            </div>
            <div class="panel-body">
                <table class="table table-bordered table-striped table-hover">
                    <thead>
                        <tr>
                            <th class="text-center" style="width: 50px;">#</th>
                            <th>Locaciones</th>
                            <th class="text-center" style="width: 100px;">Acciones</th>
                        </tr>
                    </thead>
                    <tbody>
                        <?php foreach ($all_locations as $loc) : ?>
                            <tr>
                                <td class="text-center"><?php echo count_id(); ?></td>
                                <td><?php echo remove_junk(ucfirst($loc['name'])); ?></td>
                                <td class="text-center">
                                    <div class="btn-group">
                                        <a href="edit_location.php?id=<?php echo (int)$loc['id']; ?>"
                                        class="btn btn-xs btn-warning" data-toggle="tooltip" title="Edit">
                                            <span class="glyphicon glyphicon-edit"></span>
                                        </a>
                                        <a href="delete_location.php?id=<?php echo (int)$loc['id']; ?>"
                                        class="btn btn-xs btn-danger" data-toggle="tooltip" title="Remove">
                                            <span class="glyphicon glyphicon-trash"></span>
                                        </a>
                                    </div>
                                </td>
                            </tr>
                        </tbody>
                    </table>
            </div>
        </div>
    </div>
</div>

```

```

        </td>
      </tr>
    <?php endforeach; ?>
  </tbody>
</table>
</div>
</div>
</div>
</div>
</div>
</div>
<?php include_once('layouts/footer.php'); ?>
users.php
<?php
    $page_title = 'All User';
    require_once('includes/load.php');
?>
<?php
    page_require_level(1);
    $all_users = find_all_user();
?>
<?php include_once('layouts/header.php'); ?>
<div class="row">
    <div class="col-md-12">
        <?php echo display_msg($msg); ?>
    </div>
</div>
<div class="row">
    <div class="col-md-12">
        <div class="panel panel-default">
            <div class="panel-heading clearfix">
                <strong>
                    <span class="glyphicon glyphicon-th"></span>
                    <span>Usuarios</span>
                </strong>
                <a href="add_user.php" class="btn btn-info pull-right">Agregar nuevo usuario</a>
            </div>
            <div class="panel-body">
                <table class="table table-bordered table-striped">
                    <thead>
                        <tr>
                            <th class="text-center" style="width: 50px;">#</th>
                            <th>Nombre </th>
                            <th>Nombre de usuario</th>
                            <th class="text-center" style="width: 15%;">Rol</th>
                            <th class="text-center" style="width: 10%;">Estado</th>
                            <th style="width: 20%;">Ultima conexion</th>
                            <th class="text-center" style="width: 100px;">Acciones</th>
                        </tr>
                    </thead>
                    <tbody>
                        <?php foreach($all_users as $a_user): ?>
                            <tr>
                                <td class="text-center"><?php echo count_id();?></td>
                                <td><?php echo remove_junk(ucwords($a_user['name']));?></td>
                                <td><?php echo remove_junk(ucwords($a_user['username']));?></td>
                                <td class="text-center"><?php echo remove_junk(ucwords($a_user['group_name']));?></td>
                                <td class="text-center">
                                    <?php if($a_user['status'] === '1'): ?>
                                        <span class="label label-success"><?php echo "Active"; ?></span>
                                    <?php else: ?>
                                        <span class="label label-danger"><?php echo "Deactive"; ?></span>
                                    <?php endif;?>
                                </td>
                                <td><?php echo read_date($a_user['last_login']);?></td>
                                <td class="text-center">
                                    <div class="btn-group">
                                        <a href="edit_user.php?id=<?php echo (int)$a_user['id'];?>" class="btn btn-xs btn-warning" data-toggle="tooltip" title="Edit">
                                            <i class="glyphicon glyphicon-pencil"></i>
                                        </a>
                                        <a href="delete_user.php?id=<?php echo (int)$a_user['id'];?>" class="btn btn-xs btn-danger" data-toggle="tooltip" title="Remove">
                                            <i class="glyphicon glyphicon-remove"></i>
                                        </a>
                                    </div>
                                </td>
                            </tr>
                        </tbody>
                    </table>
                </div>
            </div>
        </div>
    </div>
</div>
</div>
</div>
</div>

```

```

    <?php include_once('layouts/footer.php'); ?>
product.php
<?php
$page_title = 'All Product';
require_once('includes/load.php');
page_require_level(2);
$products = join_product_table();
?>
<?php include_once('layouts/header.php'); ?>
<div class="row">
    <div class="col-md-12">
        <?php echo display_msg($msg); ?>
    </div>
    <div class="col-md-12">
        <div class="panel panel-default">
            <div class="panel-heading clearfix">
                <div class="pull-right">
                    <a href="add_product.php" class="btn btn-primary">Agregar nuevo</a>
                </div>
            </div>
            <div class="table-responsive">
                <table class="table table-bordered">
                    <thead>
                        <tr>
                            <th class="text-center" style="width: 50px;">#</th>
                            <th> Foto</th>
                            <th> Titulo de la foto </th>
                            <th class="text-center" style="width: 10%;"> Categoria </th>
                            <th class="text-center" style="width: 10%;"> Cantidad </th>
                            <th class="text-center" style="width: 10%;"> Precio </th>
                            <th class="text-center" style="width: 10%;"> Precio de venta </th>
                            <th class="text-center" style="width: 10%;"> Bien agregado </th>
                            <th class="text-center" style="width: 10%;"> Fecha Compra</th>
                            <th class="text-center" style="width: 10%;"> Fecha Mantenimiento </th>
                            <th class="text-center" style="width: 10%;"> Encargado </th>
                            <th class="text-center" style="width: 10%;"> Serial </th>
                            <th class="text-center" style="width: 10%;"> Locacion </th>
                            <th class="text-center" style="width: 10%;"> Marca </th>
                            <th class="text-center" style="width: 100px;"> Acciones </th>
                        </tr>
                    </thead>
                    <tbody>
                        <?php foreach ($products as $product) : ?>
                            <tr style="">
                                <td class="text-center"><?php echo count_id(); ?></td>
                                <td>
                                    <?php if ($product['media_id'] === '0') : ?>
                                        
                                    <?php else : ?>
                                        
                                    <?php endif; ?>
                                </td>
                                <td> <?php echo remove_junk($product['name']); ?></td>
                                <td class="text-center"> <?php echo remove_junk($product['categoria']); ?></td>
                                <td class="text-center"> <?php echo remove_junk($product['quantity']); ?></td>
                                <td class="text-center"> <?php echo remove_junk($product['buy_price']); ?></td>
                                <td class="text-center"> <?php echo remove_junk($product['sale_price']); ?></td>
                                <td class="text-center"> <?php echo read_date($product['date']); ?></td>
                                <td class="text-center"> <?php echo remove_junk($product['fechacompra']); ?></td>
                                <td class="text-center"> <?php echo remove_junk($product['fechamantenimiento']); ?></td>
                                <td class="text-center"> <?php echo remove_junk($product['creador']); ?></td>
                                <td class="text-center"> <?php echo remove_junk($product['serialcode']); ?></td>
                                <td class="text-center"> <?php echo remove_junk($product['locacion']); ?></td>
                                <td class="text-center"> <?php echo remove_junk($product['marca']); ?></td>
                                <td class="text-center">
                                    <div class="btn-group">
                                        <a href="edit_product.php?id=<?php echo (int)$product['id']; ?>" class="btn btn-info btn-xs" title="Edit" data-toggle="tooltip">
                                            <span class="glyphicon glyphicon-edit"></span>
                                        </a>
                                        <a href="delete_product.php?id=<?php echo (int)$product['id']; ?>" class="btn btn-danger btn-xs" title="Delete" data-toggle="tooltip">
                                            <span class="glyphicon glyphicon-trash"></span>
                                        </a>
                                    </div>
                                </td>
                            </tr>
                        <?php endforeach; ?>
                    </tbody>
                </table>
            </div>
        </div>
    </div>
</div>

```

```

<?php include_once('layouts/footer.php'); ?>
profile.php
<?php
    $page_title = 'My profile';
    require_once('includes/load.php');
    page_require_level(3);
?>
<?php
    $user_id = (int)$_GET['id'];
    if(empty($user_id)):
        redirect('home.php', false);
    else:
        $user_p = find_by_id('users', $user_id);
    endif;
?>
<?php include_once('layouts/header.php'); ?>
<div class="row">
    <div class="col-md-4">
        <div class="panel profile">
            <div class="jumbotron text-center bg-red">
                
                <h3><?php echo first_character($user_p['name']); ?></h3>
            </div>
            <?php if( $user_p['id'] === $user['id']):?>
            <ul class="nav nav-pills nav-stacked">
                <li><a href="edit_account.php"> <i class="glyphicon glyphicon-edit"></i> Editar perfil</a></li>
            </ul>
            <?php endif;?>
        </div>
    </div>
</div>
<?php include_once('layouts/footer.php'); ?>
index.php
<?php
    ob_start();
    require_once('includes/load.php');
    if($session->isUserLoggedIn(true)) { redirect('home.php', false);}
?>
<?php include_once('layouts/header.php'); ?>
<div class="login-page">
    <div class="text-center">
        <h1>Bienvenido</h1>
        <p>Ingresa para iniciar sesión</p>
    </div>
    <?php echo display_msg($msg); ?>
    <form method="post" action="auth.php" class="clearfix">
        <div class="form-group">
            <label for="username" class="control-label">Nombre de usuario</label>
            <input type="text" class="form-control" name="username" placeholder="Nombre de Usuario">
        </div>
        <div class="form-group">
            <label for="Password" class="control-label">Password</label>
            <input type="password" name="password" class="form-control" placeholder="password">
        </div>
        <div class="form-group">
            <button type="submit" class="btn btn-info pull-right">Login</button>
        </div>
    </form>
</div>
<?php include_once('layouts/footer.php'); ?>

```

DECLARACIÓN Y AUTORIZACIÓN

Yo, Castro Merino Rafael Guillermo, con C.C: # 0703819110 autor del trabajo de titulación: Diseño e implementación de un sistema de inventario usando la tecnología NFC para la Unidad Educativa Particular Virgen del Cisne en la ciudad de Machala mediante una aplicación con sistema operativo IOS., previo a la obtención del título de Magíster en Telecomunicaciones en la Universidad Católica de Santiago de Guayaquil.

1.- Declaro tener pleno conocimiento de la obligación que tienen las instituciones de educación superior, de conformidad con el Artículo 144 de la Ley Orgánica de Educación Superior, de entregar a la SENESCYT en formato digital una copia del referido trabajo de titulación para que sea integrado al Sistema Nacional de Información de la Educación Superior del Ecuador para su difusión pública respetando los derechos de autor.

2.- Autorizo a la SENESCYT a tener una copia del referido trabajo de titulación, con el propósito de generar un repositorio que democratice la información, respetando las políticas de propiedad intelectual vigentes.

Guayaquil, 1 de diciembre de 2021

f.  _____

Nombre: Castro Merino Rafael Guillermo

C.C: 0703819110



REPOSITORIO NACIONAL EN CIENCIA Y TECNOLOGÍA		
FICHA DE REGISTRO DE TESIS/TRABAJO DE TITULACIÓN		
TÍTULO Y SUBTÍTULO:	Diseño e implementación de un sistema de inventario usando la tecnología NFC para la Unidad Educativa Particular Virgen del Cisne en la ciudad de Machala mediante una aplicación con sistema operativo IOS	
AUTOR(ES)	Castro Merino Rafael Guillermo	
REVISOR(ES)/TUTOR	MSc. Luis Córdova Rivadeneira; MSc. Edgar Quezada Calle / MSc. Manuel Romero Paz	
INSTITUCIÓN:	Universidad Católica Santiago de Guayaquil	
FACULTAD:	Sistema de Posgrado	
PROGRAMA:	Maestría en Telecomunicaciones	
TITULO OBTENIDO:	Magister en Telecomunicaciones	
FECHA DE PUBLICACIÓN:	Guayaquil, 1 de diciembre de 2021	No. DE PÁGINAS: 151
ÁREAS TEMÁTICAS:	Tecnología NFC, Aplicación móvil, aplicación web, Arquitectura IOS, Modos de operación, Etiquetas	
PALABRAS CLAVES/KEYWORDS:	NFC, NDEF, IOS, Apple, web, inventario.	
RESUMEN: El proyecto presentado en este documento está relacionado con el uso e implementación de la tecnología Near Field Communication, abreviada NFC, mediante una aplicación móvil, y un servidor web para almacenar y procesar la información obtenida con el fin de mantener el control sobre los ítems que componen el Inventario de la empresa. El prototipo desarrollado en este proyecto utiliza un servidor web, una base de datos, etiquetas NFC y la aplicación hecha para trabajar con el sistema operativo IOS de la empresa Apple que es la segunda marca móvil más utilizada del mercado. El capítulo uno se enfoca en el tipo de proyecto a realizar. El capítulo dos presenta los conceptos sobre tecnología NFC, bases de datos, sistema operativo IOS, lenguajes de programación y todo el software incluido. Además, los objetivos y alcance del proyecto. El capítulo tres describe la arquitectura y el desarrollo general de las aplicaciones y cómo estas interactúan entre ellas. El capítulo cuatro involucra los datos de prueba y los parámetros para el uso correcto del software. Finalmente describe las conclusiones, recomendaciones y prácticas de seguridad que se obtuvieron a través de este proyecto		
ADJUNTO PDF:	<input checked="" type="checkbox"/> SI	<input type="checkbox"/> NO
CONTACTO CON AUTOR/ES:	Teléfono: +593-992808581	E-mail: raf.castro@hotmail.com
CONTACTO CON LA INSTITUCIÓN (COORDINADOR DEL PROCESO UTE):	Nombre: Romero Paz Manuel de Jesús	
	Teléfono: +593-994606932	
	E-mail: manuel.romero@cu.ucsg.edu.ec	
SECCIÓN PARA USO DE BIBLIOTECA		
No. DE REGISTRO (en base a datos):		
No. DE CLASIFICACIÓN:		
DIRECCIÓN URL (tesis en la web):		