UNIVERSIDAD POLITÉCNICA SALESIANA
SEDE QUITO

CARRERA:
INGENIERÍA DE SISTEMAS

Trabajo de titulación previo a la obtención del título de:
Ingeniera e Ingeniero de Sistemas

TEMA:
DESARROLLO DE UNA APLICACIÓN MÓVIL, PARA S.O ANDROID, QUE AUTOMATICE EL LEVANTAMIENTO DE LA INFORMACIÓN DEL INVENTARIO PARA LA ZONA 9 (PICHINCHA) DEL MSP

AUTORES:
NATALIA PAMELA CUEVA CORONADO
HENRY RAFAEL AVILA JUCA

TUTOR:
ALONSO RENÉ ARÉVALO CAMPOS

Quito, septiembre de 2017

CESIÓN DE DERECHOS DE AUTOR

Nosotros Natalia Pamela Cueva Coronado, con documento de identificación N° 1722735568 y Henry Rafael Avila Juca, con documento de identificación N° 0604231381, manifestamos nuestra voluntad y cedemos a la Universidad Politécnica Salesiana la titularidad sobre los derechos patrimoniales en virtud de que somos autores del trabajo de titulación con tema: DESARROLLO DE UNA APLICACIÓN MÓVIL, PARA S.O ANDROID, QUE AUTOMATICE EL LEVANTAMIENTO DE LA INFORMACIÓN DEL INVENTRARIO PARA LA ZONA 9 (PICHINCHA) DEL MSP, mismo que ha sido desarrollado para optar por el título de: INGENIEROS DE SISTEMAS, en la Universidad Politécnica Salesiana, quedando la universidad facultada para ejercer los derechos cedidos anteriormente.

En la aplicación a lo determinado por la Ley de Propiedad Intelectual, en nuestra condición de autores nos reservamos los derechos morales de la obra antes citada.

En concordancia, suscribo este documento en el momento que hago entrega del trabajo final en formato impreso y digital a la Biblioteca de la Universidad Politécnica Salesiana.

NATALIA PAMELA
CUEVA CORONADO
CI: 1722735568

HENRY RAFAEL
AVILA JUCA
CI: 0604231381

Quito, septiembre de 2017

DECLARATORIA DE COAUTORÍA DEL DOCENTE TUTOR

Yo declaro que bajo mi dirección y asesoría fue desarrollado el proyecto técnico, DESARROLLO DE UNA APLICACIÓN MÓVIL, PARA S.O ANDROID, QUE AUTOMATICE EL LEVANTAMIENTO DE LA INFORMACIÓN DEL INVENTRARIO PARA LA ZONA 9 (PICHINCHA), realizado por Natalia Pamela Cueva Coronado y Henry Rafael Avila Juca, obteniendo un producto que cumple con todos los requisitos estipulados por la Universidad Politécnica Salesiana, para ser considerados como trabajo final de titulación.

Quito, septiembre de 2017

ALONSO RENÉ ARÉVALO CAMPOS

C.I.: 1400164891

ÍNDICE

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

Resumen

Los dispositivos móviles inteligentes conocidos como dispositivos smarts se han convertido en una herramienta de trabajo indispensable, debido que permiten realizar y recibir llamadas, tener conexión continua a internet para permanecer comunicados, instalar aplicaciones útiles para llevar a cabo las tareas diarias entre otras funcionalidades. En la actualidad la mayoría de los dispositivos disponibles en el mercado cuentan con el Sistema Operativo Android debido que es una plataforma de acceso libre.

Después de realizar una pequeña introducción de las características y herramientas de esta plataforma, se analizará, diseñará y desarrollará una aplicación para el levantamiento de la información del inventario de la ficha familiar del Ministerio de Salud Pública.

La función principal de esta aplicación es el ingreso de las fichas familiares, además de conectar con un servidor para almacenar la información y permitir la descarga de las fichas para el seguimiento anual que realiza la institución.

Esta aplicación contendrá en su mayoría listas desplegables para el registro de los datos y el mínimo ingreso de estos campos de manera manual permitirá evitar retrasos, incompatibilidades, perdidas y poca legibilidad en la entrega de información convirtiéndose así en un instrumento fundamental para establecer las condiciones de salud de la población de la zona, sus necesidades, el entorno económico y social, problemas de salud del grupo familiar para la toma de decisiones en pro de la comunidad.

Abstract

Smart mobile devices known as smart devices have become an indispensable work tool, so you can make and receive calls, have a connection for permanent communication, and install applications useful to carry out daily tasks among other functionalities. At present, most of the devices available in the market have the Android operating system because it is a free access platform.

After a brief introduction of the features and tools of this platform, an application will be analyzed, designed and developed for the collection of information from the inventory of the family file of the Ministry of Public Health.

The main function of this application is the entry of family files, in addition to connecting with a server to store the information and allow the download of the tabs for the annual monitoring carried out by the institution.

This application will contain mostly drop-down lists for data recording and the minimum entry of these fields manually will allow to avoid delays, incompatibilities, losses and little readability in the delivery of information becoming a fundamental instrument to establish the conditions of Health of the population of the area, their needs, the economic and social environment, health problems of the family group to make decisions for the community.

INTRODUCCIÓN

Antecedentes

El Ministerio de Salud Pública del Ecuador Zona 9 (Pichincha) crea un convenio con la Universidad Politécnica Salesiana y propone el desarrollo de una aplicación para dispositivos móviles que permita el ingreso de información, consulta y actualización de las fichas familiares en sincronización con el servidor de la institución, con el fin de realizar el llenado de los formularios de manera ágil y obtener información de calidad para la toma de decisiones.

“ La Zona 9 del Ministerio de Salud cuentan con un formulario para el levantamiento de la información de inventario de la ficha familiar de los hogares de la zona que tiene por objeto contribuir con información sobre las diversas características de la vivienda tales como: tenencia, infraestructura, servicios básicos, sistema de alumbrado, combustible de cocina, presencia de animales y otros vectores, así como establecer los problemas de salud de los miembros del hogar, causas de fallecimiento de los integrantes para desarrollar actividades inherentes a la salud por parte de la unidad de atención” (Ministerio de Salud Pública del Ecuador, 2016).

La información de cada formulario de inventario o también llamado ficha familiar se llena en hojas impresas de manera manual dando paso a información incompleta, incompatible, inentendible y pérdida de la misma; perdiendo veracidad para el análisis por parte de los encargados y conduciéndolos a tomar malas decisiones.

Problema

La Zona 9 del Ministerio de Salud Pública cuenta con un formulario impreso para el levantamiento de información de inventario de la ficha familiar convirtiéndose en una actividad tediosa y repetitiva debido que toda la información se la llena de

manera manual, lo que ocasiona retrasos, incompatibilidad, inentendible y pérdida de la información. Evitando así tener información actualizada para la toma de decisiones acertadas para el desarrollo de actividades en bien de la comunidad.

Justificación

En la Zona 9 del MSP, se implementará una aplicación para dispositivos móviles, la cual permitirá el levantamiento del inventario de la ficha familiar del Ministerio de Salud con información de las afecciones a la salud, clasificación CIE10, más parámetros relacionados con la dotación de servicios básicos, datos de salud del grupo familiar, datos socioeconómicos de los hogares en la provincia de Pichincha.

Al desarrollar esta aplicación se permitirá optimizar los recursos necesarios para mejorar la productividad, debido que los recursos bien empleados evitaran retrasos, incompatibilidades, perdidas, y poca legibilidad en la entrega de información, además de reducir el trabajo de los encargados de esta actividad.

Objetivos

Objetivo General

Desarrollar una aplicación móvil en S.O Android, para levantar el inventario con información de las afecciones a la salud, clasificación CIE10 para la Zona 9 del MSP.

Objetivos Específicos

Diseñar una aplicación móvil que permita automatizar el levantamiento de la información de los formularios relacionados con afecciones a la salud, utilizando la clasificación CIE10, para la zona 9 del MSP.

Implementar un servicio web que posibilite transmitir la información de los formularios levantados a una base de datos PostgreSQL en un servidor previsto para almacenar la información para su posterior análisis.

Asegurar la consistencia y valides de información obtenida en el levantamiento del inventario en relación con el contenido del formulario.

Alcance

Automatizar el formulario de inventario de la ficha familiar para el ingreso, consulta y actualización de la información en el dispositivo móvil.

Sincronizar el dispositivo móvil con el servidor de la Zona 9 del Ministerio de Salud Pública a través de un servicio web para la obtención de catálogos del formulario y manejo de la información.

Exportar la información obtenida en el dispositivo móvil al servidor de la institución para su análisis y toma de decisiones posteriores.

Importar información desde el servidor al dispositivo móvil para actualizar los catálogos y formularios para dar seguimiento a los mismos.

Marco Metodológico

Para la automatización de los formularios se utilizará una metodología ágil y flexible llamada SCRUM y una segunda metodología utilizada en el desarrollo de aplicaciones Web llamada UWE.

Metodología SCRUM

Es una metodología de desarrollo de software ágil y flexible, una de las más utilizadas en la actualidad que tiene como directrices un conjunto de buenas prácticas, al implementar esta metodología se puede llevar a cabo cambios

funcionales necesarios o de prioridad solicitados por la parte interesada o stakeholders. Además, que se puede realizar entregables funcionales durante el desarrollo de la misma debido que esta metodología trabaja con iteraciones (ciclo repetitivo para el desarrollo de los proyectos) que duran de 1 a 4 semanas.

Algunos de los beneficios que presenta esta metodología son: disminución de riesgos debido que cuando se conoce las prioridades en cuanto a las funcionalidades y como se las está llevando a cabo, se consigue prevenir algunos riesgos que pueden afectar al proyecto.

La tolerancia a cambios es otro beneficio debido que la capacidad de adaptación y reacción es alta ante los requerimientos cambiantes que se presenten en el proyecto por parte de los interesados y el establecimiento de tiempos ya que se puede estimar de una manera fácil el tiempo de cada iteración, es decir, el posible momento de entrega de una versión funcional del proyecto.

El equipo que conforma el desarrollo de un proyecto SCRUM son:

Stakeholders: también llamados parte interesada son aquellos que si no son gestionados de una manera adecuada afectan el correcto desarrollo y cumplimiento de los objetivos del proyecto.

Maestro Scrum: “Encargado de guiar y orientar al equipo de desarrollo del proyecto para alcanzar los objetivos fijados para la finalización del proyecto”, se encarga de la motivación del equipo y verifica la participación de los mismos. (Rodriguez del Valle, 2009)

Equipo de trabajo: encargados de llevar a cabo las actividades para cumplir con los objetivos del proyecto.

Dueño del producto: también llamado Product Owner tiene conocimiento de cómo funciona el negocio y es su prioridad es que se trabaje de la manera más adecuada para obtener en lo posible los resultados en el tiempo indicado.

Metodología UWE

"UWE es una metodología para el desarrollo de aplicaciones Web que se basa en diagramas UML para sus modelos y métodos. La metodología define claramente la construcción de cada uno de los elementos del modelo, al ser un proceso iterativo, presenta la posibilidad de realizar algún cambio de prioridad del proyecto en cualquier instante del mismo sin presentar inconveniente alguno" (Universidad Carlos III de Madrid).

La tolerancia a cambios que presenta esta metodología "al contar con los diagramas de las distintas etapas del proceso de desarrollo se tiene la capacidad de reacción y adaptación inmediata ante los cambios generados por los involucrados" (Universidad Carlos III de Madrid) y el **t**rabajo en equipo "permite la participación de todos los miembros del equipo de una manera activa en todo el desarrollo" (Universidad Carlos III de Madrid). Además de la optimización "en costos de diseño, rediseño y mantenimiento aumentando la satisfacción y comodidad del usuario" (Universidad Carlos III de Madrid) son algunos de los beneficios que se obtiene en el uso de esta metodología para el desarrollo de proyectos de software.

Las etapas para el desarrollo de esta metodología son:

Modelo de casos de uso: "Modelo que permite capturar los requisitos funcionales del sistema a través de un diagrama de casos de uso conformado por los elementos actor y caso de uso. Los actores son los usuarios de la aplicación". (Uicab Aldana, Ucán Pech, & Aguilar Vera, 2016)

Modelo de contenido: “Modelo conceptual para el desarrollo del contenido mediante un diagrama de clases, detalla los conceptos implicados en la aplicación”. (Uicab Aldana, Ucán Pech, & Aguilar Vera, 2016)

Modelo de usuario: “Modelo de navegación, es decir, representa cómo interactúan los objetos dentro de la aplicación y un conjunto de estructuras como índices, menús, consultas, etc.” (Uicab Aldana, Ucán Pech, & Aguilar Vera, 2016)

Modelo de presentación: “Modelo que representa la interfaz de usuario y ciclo de vida del objeto”. (Uicab Aldana, Ucán Pech, & Aguilar Vera, 2016)

Modelo de proceso: “Representa el aspecto que tiene las actividades que se conectan en cada clase de procesos”. (Uicab Aldana, Ucán Pech, & Aguilar Vera, 2016)

Capítulo 1

Estado del Arte

1.1 Dispositivos móviles

Un dispositivo móvil se lo define como un aparato electrónico cuyas características principales se asemejan a un computador de escritorio o personal, pero sus prestaciones y funcionalidades son un poco más limitadas en cuanto a la capacidad de procesamiento y almacenamiento, otra de las características que lo diferencia de un computador es su tamaño y batería como lo menciona (Enriquez & Casas, 2013) "Los dispositivos móviles son suficientemente livianos como para ser transportados por personas y disponen de la capacidad de batería adecuada para funcionar de forma autónoma".

Dispositivos Móviles

Figura 1: Dispositivos móviles
Fuente: (Enriquez & Casas, 2013)

La evolución de los dispositivos móviles es significativo día a día debido reúnen elementos de software y hardware que aumentan sus funcionalidades y prestaciones, algunos de estos elementos son: cámara de video y fotografías, conexión a internet permanente o interrumpida a una red, localizador GPS, memoria externa, pantalla táctil, programas para visualización de documentos, navegadores, etc. Debido a esta

evolución los dispositivos móviles son utilizados como herramientas de trabajo para obtener mejoras en las actividades diarias de los usuarios.

En la actualidad los dispositivos más utilizados por los usuarios son los dispositivos SMART, ya que permiten la instalación de diversas aplicaciones que les permiten mantenerse comunicados, cumplir con actividades de trabajo, llevar a cabo actividades de entretenimiento o trabajo, cuidar su salud, etc.

1.1.1 Dispositivos Smart

Dispositivo que contiene características similares a una computadora personal que además trabaja como un teléfono móvil, pero con capacidad y tamaño reducido. Las características principales de estos dispositivos es la instalación de programas o aplicaciones móviles, pantallas táctiles, sistema operativo, conectividad a internet a través de WIFI, permite recibir o realizar llamadas, lectura de documentos en diversos formatos, entre otras funcionalidades.

A diferencia de las computadoras personales la capacidad de ciertos componentes es limitada en memoria RAM, memoria de almacenamiento interno, menor capacidad de procesamiento, etc. En este grupo se encuentran las tablets, smartphones, smartwatch, etc.

Smartphones o teléfono inteligente

Figura 2: Smartphone
Fuente: (Eficazz, s.f.)

Una de las actividades que es de nuestro interés el uso de estos dispositivos es para levantamiento de la información.

En este campo estos dispositivos se han convertido en elementos de utilidad debido que permiten que las personas estén en contacto independientemente del tiempo o lugar en el que se hallen, permitiendo así recolectar información real y necesaria para la toma de decisiones en el campo de investigación que lo está solicitando.

1.2 Sistemas Operativos para dispositivos móviles

Los dispositivos móviles de manera similar que una computadora personal necesita de un sistema operativo (SO) para que funcione de una manera adecuada, estos sistemas son simples y de tamaño reducido debido a la capacidad de almacenamiento de dichos dispositivos.

Los sistemas operativos para dispositivos móviles se encuentran dirigidos a la conectividad inalámbrica, formatos multimedia y ejecución de aplicaciones para llevar a cabo funciones de uso diario del usuario.

En la actualidad los SO de mayor uso son:

1.2.1 Android

“Basado en Linux, desarrollado por Android Inc. y comprado por Google en 2005. El éxito de este OS se debe en gran parte a que su plataforma es abierta y puede ser adoptada por más de un fabricante, además fomenta a desarrolladores de la industria a seguir generando aplicaciones con el código abierto soportando tecnologías de Java y Flash” (Molina Martínez, 2012) .

1.2.2 Symbian

“Liderado por Nokia (Symbain.org 2008) en colaboración con Sony Ericsson, Samsung, LG, etc. También es una plataforma abierta, pero actualmente se encuentra en obsolescencia en comparación con las tendencias actuales” (Molina Martínez, 2012).

1.2.3 iOS

“Anteriormente llamado iPhone OS es un sistema operativo móvil de Apple desarrollado originalmente para el iPhone desde enero de 2007. Es un derivado del sistema operativo de escritorio Mac OS X. No soporta Adobe Flash, ni Java su arquitectura es cerrada” (Molina Martínez, 2012).

1.2.4 Windows Phone

“Desarrollado por Microsoft desde septiembre de 2010, como sucesor de la plataforma Windows Mobile. Está pensado para el mercado de consumo generalista en lugar del mercado empresarial” (Molina Martínez, 2012).

Para llevar a cabo el desarrollo de la aplicación se ha seleccionado la plataforma Android debido que no es u software propietario y al crecimiento en su uso en los últimos años.

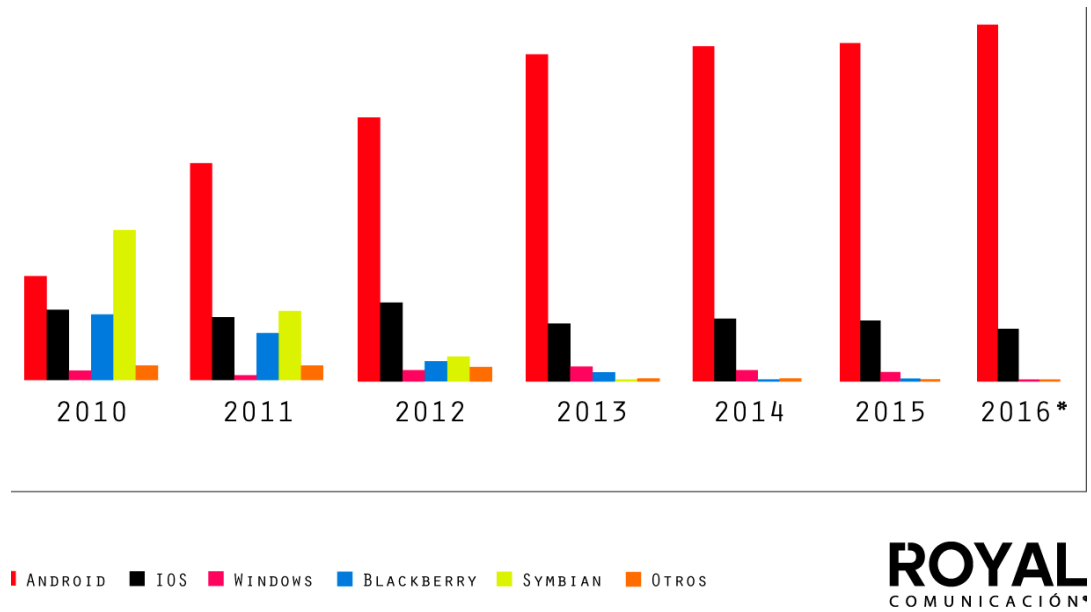


Figura 3: Venta de Sistemas Operativos de Smartphones
Fuente: Royal Comunicación

1.3 Aplicaciones móviles

Una aplicación móvil o APP, es un programa cuyo objetivo es ayudar en las actividades de las personas para mejorar su rendimiento, están creadas para ser instaladas y utilizadas en los smartphones o dispositivos inteligentes.

1.3.1 Aplicaciones Nativas

Desarrolladas para un tipo de SO y dispositivo particular, son código ejecutable en el dispositivo que pueden acceder a las funciones del mismo. Existen sitios en donde se encuentran alojadas estas aplicaciones para ser descargadas, estas son creadas para plataformas específicas.

1.3.2 Aplicaciones Web móviles

"Las aplicaciones móviles de este tipo se encuentran ejecutando en servidores, estas incluyen página web optimizadas para ser visualizadas por dispositivos móviles y se pueden desarrollar en HTML, Java Script, CSS, etc. Por definición serán accedidas utilizando algún navegador web. El problema que presentan es que no pueden

acceder a las funcionalidades propias del dispositivo. Por ejemplo, una aplicación web no puede emplear la cámara de un smartphone" (Enriquez & Casas, 2013).

Como se indica en la Figura 3 el Sistema Operativo Android es el de mayor uso en la actualidad, por tal motivo seleccionamos las aplicaciones más utilizadas en dispositivos móviles que cuentan con este sistema operativo.

Según el informe emitido por (AVG Latinoamérica, 2016) las aplicaciones más utilizadas son Play Store, Google Plus, Youtube, Google Chrome, Google Mapas, Facebook, etc.g

Figura 4: 50 Apps más utilizadas en Android
Fuente: (AVG Latinoamérica, 2016)

1.4 Java 2 Micro Edition

La revolución de la tecnología presentada en estos últimos tiempos llevó a los expertos a buscar soluciones personalizadas para cubrir las necesidades del medio y la tecnología utilizada, es así que Sun crea Java con sus distintas ediciones para dar soluciones a los problemas.

Java Enterprise Edition (Java EE), dirigida al ambiente corporativo o empresarial.

Java Standard Edition (Java SE), dirigida a desarrollar aplicaciones independientes de la plataforma.

Java Micro Edition (Java ME), dirigida a dispositivos con capacidades mínimas.

En el siguiente grafico se muestra la evolución de cada una de las ediciones de Java, cabe recalcar, que Java Enterprise Edition es la contenedora de la mayoría por no decir todos los paquetes que contienen las versiones mínimas a estas.

Cada edición es usada en el ambiente que el desarrollador lo requiere, pero se debe tomar en cuenta las consideraciones anteriormente mencionadas.

Ediciones de Java

Figura 5: Arquitectura de Java 2
Fuente: (Gálvez Rojas & Ortega Díaz, 2014)

Cada una de estas ediciones permite desarrollara aplicaciones dirigidas a cada uno de los ambientes antes mencionados, pero comparten un conjunto de API básicas de JAVA que se encuentran agrupadas en paquetes como: java. lang y java.io.

Para nuestro estudio es de interés Java Micro Edition ya que es una edición creada para desarrollar, instalar y ejecutar aplicaciones de escritorio en dispositivos electrónicos móviles que cada vez aumentan su capacidad de almacenamiento, procesamiento y despliegue de gráficos como lo son los dispositivos móviles.

1.4.1 Java 2 Micro Edition (J2ME)

J2ME no define un nuevo lenguaje, sino que mantiene la compatibilidad, hacia arriba, con la edición J2SE, adaptando la tecnología Java a su uso en dispositivos móviles, lo que proporciona un entorno de desarrollo para crear aplicaciones destinadas a familias de dispositivos móviles, contando con las librerías de clases necesarias para construir interfaces gráficas, almacenamiento persistente, manejo de eventos, conectividad de red y juegos (Jiménez & Castro , 2014).

1.4.2 Componentes de J2ME

Para desarrollar aplicaciones en J2ME se debe tomar en cuenta ciertos componentes, la combinación de los mismos se ayuda a cumplir con las expectativas del cliente; algunos de ellos son:

Máquina virtual interpreta el código intermedio de los programas escritos en leguaje Java, es decir, ejecuta el código obtenido de la compilación del código fuente al de la plataforma a utilizar. Existen dos máquinas virtuales dependiendo las características del dispositivo.

KVM “Su nombre proviene de kilobyte (haciendo referencia a la baja ocupación de memoria, entre 40 kb y 80 kb). Se trata de una implementación de máquina virtual reducida y especialmente orientada a dispositivos con bajas capacidades computacionales y de memoria, alta portabilidad, modulable” (Gálvez Rojas & Ortega Díaz, 2014).

CVM “Compact Virtual Machine ha sido tomada como máquina virtual de Java soporta las mismas características que la máquina virtual de J2SE. Está orientada a dispositivos electrónicos con procesadores de 32 bits de alta gama y entorno de 2Mb

o más de memoria, portabilidad, rápida sincronización" (Gálvez Rojas & Ortega Díaz, 2014).

Configuraciones "conjunto mínimo de APIs Java que permiten desarrollar aplicaciones para un grupo de dispositivos. Estas APIs describen las características básicas, comunes a todos los dispositivos como: características soportadas del lenguaje de programación Java, características soportadas por la máquina virtual Java, bibliotecas básicas de Java y APIs soportadas" (Gálvez Rojas & Ortega Díaz, 2014).

Las configuraciones se clasifican en dos:

Connected limited Device configuración (CLDC) orientada a dispositivos con restricciones de procesamiento y memoria.

Connected Device Configuration CDC orientada a dispositivos sin limitación de recursos.

Perfiles conjunto de APIs (bibliotecas) orientado a un ámbito de aplicación determinado. "Los perfiles identifican un grupo de dispositivos por la funcionalidad que proporcionan (electrodomésticos, teléfonos móviles, etc.) y el tipo de aplicaciones que se ejecutaran en ellos" (Gálvez Rojas & Ortega Díaz, 2014).

Estos perfiles se utilizan sobre las configuraciones descritas anteriormente es así que si se utiliza un CDC se debe usar la máquina virtual CVM, al contrario de usar la configuración CLDC se usa la máquina virtual KVM.

1.5 Trabajos relacionados

1.5.1 Prototipo de una aplicación móvil para el manejo del historial clínico y control médico del paciente.

El problema se encuentra cuando un médico debe realizar la visita médica a los pacientes a sus domicilios, debe registrar la historia clínica de manera manual lo que con lleva a errores en el registro de la información, además que si ya se encuentra registrada la historia clínica el médico tiene la tarea de buscar los documentos de dicha historia y trasladarlo consigo corriendo el riesgo de la perdida de estos debido que tratar a los pacientes sin la información necesaria puede presentar un servicio deficiente. Por este motivo se desarrolló "un prototipo de software de historias clínicas y control médico a través de dispositivos móviles y apoyados en tecnología GPS garantizando la disponibilidad de la información clínica" (Babatina Goyeneche, Briceño Novoa, & Cardoso Gómez, 2016).

El desarrollo de esta aplicación móvil permitió registrar la información de la historia clínica reduciendo los tiempos de esta actividad, además de descargar la información de historia clínicas previamente ingresadas del servidor al dispositivo móvil para poder ofrecer un servicio de mejor calidad al paciente.

Esta aplicación fue desarrollada como proyecto de titulación de la Universidad Distrital "Francisco José de Caldas", en la ciudad de Bogotá- Colombia en el año 2016.

APP para el manejo del historial clínico y control médico del paciente

Figura 6: app para el manejo del historial clínico y control médico del paciente
Referencia: (Babatina Goyeneche, Briceño Novoa, & Cardoso Gómez, 2016)

1.5.2 Desarrollo de un Sistema de Información Móvil, utilizando RUP, para el registro de pedidos en la empresa PALFARMA E.I.R.L.

Esta aplicación se desarrolló debido que para registrar un nuevo pedido de productos médicos se lo realiza de manera manual a través de hojas impresas lo que genera problemas tales como: errores en el registro de la información, pérdida de pedidos, mala distribución de los productos e insatisfacción de los clientes, "los pedidos se realiza de manera rudimentaria, manualmente con hojas y lapiceros conllevando algunas veces errores en el registro y por consiguiente generando la insatisfacción de los clientes" (Guerra Salazar & Lucas Granda, 2013).

La app permitirá agilizar y optimizar el proceso de registro de pedidos sin importar el lugar en donde se encuentre el operario de la aplicación, además del registro de nuevos clientes y la consulta sobre productos de medicina que se encuentran disponibles en percha.

La aplicación móvil fue desarrollada como un proyecto de titulación de la Universidad Privada Tacna- Facultad de Ciencias Empresariales, de la ciudad de Tacna- Perú en el año 2013.

1.5.3 GeoDynamic

“Es una aplicación desarrollada para teléfonos celulares y tablets con sistema operativo Android, que permiten al encuestador levantar cualquier tipo de información de acuerdo con una campaña previamente creada en el componente web de acuerdo con los requerimientos y objetivos del cliente final” (Location World). Esta aplicación se la utiliza después de ingresar los campos de consulta en un servidor web y sincronización posterior con el dispositivo móvil para el levantamiento de la información.

Esta aplicación tiene un costo anual para el uso de reportes y obtención de la información en la aplicación web, como se indica está compuesta por dos partes en la cual es usuario interactúa con la aplicación móvil la cual se descarga desde las tiendas disponibles para su comercialización sin costo alguno.

GEODYNAMIC

Figura 7: app GEODYNAMIC
Fuente: (Location World)

Capítulo 2

Marco Teórico

2.1 Marco Referencial

2.1.1 Formulario de inventario de la ficha familiar del MSP

"La ficha familiar, es instrumento de archivo de la unidad, para lo cual debe destinarse el espacio respectivo dentro del área de estadística. Este instrumento no reemplaza la historia clínica individual, es un complemento. La apertura de la ficha familiar y la primera consulta, permiten conocer la realidad social de la comunidad, clasificar a la población en grupos de riesgos, realizar un diagnóstico de salud a nivel local y establecer prioridades en la atención" (Ministerio de Salud Pública del Ecuador, 2012, págs. 77-78).

Este formulario se convierte en un instrumento fundamental para establecer las condiciones de salud de la población de la zona, sus necesidades, el entorno económico y social, problemas de salud del grupo familiar que lo rodea para establecer las mejores políticas de salud para garantizar su derecho a la salud.

2.2 Marco Teórico

2.2.1 Plataforma Android

"Android es una pila de software de código abierto basado en Linux creada para una variedad amplia de dispositivos" (Android Developers, 2016), el objetivo primordial de esta plataforma es juntar todos los elementos necesarios tales como: APIs y herramientas de desarrollo, compilación y evolución que permitan al desarrollador acceder y aprovechar las funcionalidades dadas por los dispositivos móviles para crear aplicaciones estandarizadas a ser ejecutadas en cualquiera de ellos, claro está que estos dispositivos deben contener este sistema operativo.

Algunas de las características de Android son:

Desarrollar aplicaciones reutilizables, portables y gratuitas entre los distintos dispositivos móviles.

Para ejecutar las aplicaciones cuenta con la máquina virtual Dalvik.

Maneja a través de una librería una base de datos llamada SQLite.

Maneja diversos formatos multimedia.

Mantiene el control sobre diversos elementos de hardware

Al ser una plataforma Android está constituida por una arquitectura de 4 capas, cada una de ellas "utiliza servicios ofrecidos por las anteriores y ofrece los suyos a las capas superiores" (Almenárez Mendoza, 2011)

Arquitectura de Android

Figura 8: Arquitectura de Android
Fuente: (Almenárez Mendoza, 2011)

Capa de aplicaciones *c*ontiene las aplicaciones propias del dispositivo o aquellas instaladas por el usuario.

Capa Framework de aplicaciones “conjunto de herramientas (APIs) para desarrollar cualquier aplicación, en donde se gestiona el ciclo de vida de las aplicaciones, permite compartir información entre ellas y nos permite construir y controlar la interfaz de usuario”. (Almenárez Mendoza, 2011)

Capa de librerías “se encuentran escritas en C++ y proporcionan a Android sus características, los usuarios utilizan estas de acuerdo con sus necesidades de desarrollo” (Almenárez Mendoza, 2011).

Capa Runtime en esta capa se encuentra la máquina virtual Dalvik que responde a las limitaciones de los dispositivos de tal forma que maximiza la memoria, también cuenta con las librerías para el lenguaje Java.

Capa Kernel de Linux “Android está construido sobre el núcleo de Linux, pero se ha modificado dramáticamente para adaptarse a dispositivos móviles. Esta elección está basada en la excelente potabilidad, flexibilidad y seguridad que Linux presenta”. (Revelo, 2014)

En esta capa se presta servicios tales como: seguridad, manejo de memoria, protocolos, soporte a drivers de los dispositivos, entre otros, cabe aclarar que al proporcionar servicios de drivers para el dispositivo es la única que tiene dependencia con el hardware.

Aplicaciones**,** son programas desarrollados para cumplir una función en el dispositivo o para ayudar en alguna actividad al usuario.

Emulador Android es un dispositivo virtual que se ejecuta en la computadora personal ayuda en el desarrollo de aplicaciones debido que simula un dispositivo real. Máquina Virtual Dalvik: “La máquina virtual de Dalvik (DVM) es un intérprete de códigos que han sido transformados de bytecode de Java a bytecode de Dalvik.

Propiamente Dalvik esta compilado a código nativo, mientras que las librerías que interpreta están escritas en Java" (Obando Núñez & Orrala Paredes, 2013).

Dalvik no cambia nada en el proceso de compilación, sencillamente interviene al final como receptor de un archivo ejecutable producto de una recopilación de los archivos .class de java. Las fases de la construcción de una aplicación Java son: El primer paso es generar el código fuente (archivos .java), luego este es traducido por el Java Compiler (javac) y obtenemos un fichero tipo bytecode (archivos .class). Finalmente, la máquina virtual de Java (JVM) interpreta en tiempo real este archivo y la aplicación es ejecutada. (Revelo, 2014)

2.2.2 SQLite

"SQLite es una base de datos Open Source, es muy popular en muchos dispositivos pequeños, como Android. Las ventajas que presenta utilizar SQLite es que no requiere configuración, no tiene un servidor de base de datos ejecutándose en un proceso separado y es relativamente simple para su empleo. SQLite a diferencia de otras bases de datos SQL, no tiene un proceso servidor independiente. SQLite lee y escribe directamente a archivos de disco ordinarios, es una base de datos completa de SQL con tablas, índices, triggers y vistas, está contenida en un archivo de disco único" (SQLite).

SQLite es un sistema gestor de base de datos relacional que se encuentra en una librería de Android escrita en C, considerado una solución embebida, es decir, es una librería que esta enlazada a las aplicaciones, aparece como la solución a la necesidad de manipular la información en un dispositivo móvil.

"La base de datos se almacena en un único fichero a diferencia de otros DBMS que hacen uso de varios archivos. SQLite emplea registros de tamaño variable de forma

tal que se utiliza el espacio en disco que es realmente necesario en cada momento" (EcuRed, 2017).

Las operaciones que se realiza para el manejo de la información de la base de datos se las lleva a cabo dentro de la aplicación mediante llamadas a funciones contenidas en esta librería, utiliza las sentencias de SQL al igual que los otros sistemas gestores de base de datos.

Para utilizar la base de datos SQLite en Android se debe crear una clase que sea un extends de la librería SQLiteOpenHelper en la cual escribiremos los métodos necesarios para obtener la funcionalidad con la base de datos deseada. Sus métodos principales son onCreate en el cual se creará la base de datos en el caso de no existir y onUpgrade nos servirá para eliminar la base datos en el caso de existir y crearla nuevamente.

2.2.3 PostgreSQL

"Es un sistema de gestión de bases de datos objeto-relacional, tiene licencia BSD, su código fuente es libre, además utiliza un modelo cliente/servidor y usa multiprocesos en vez de multi-hilos para asegurar la estabilidad del sistema". (Postgresql Org)

Componentes de un sistema PostgreSQL

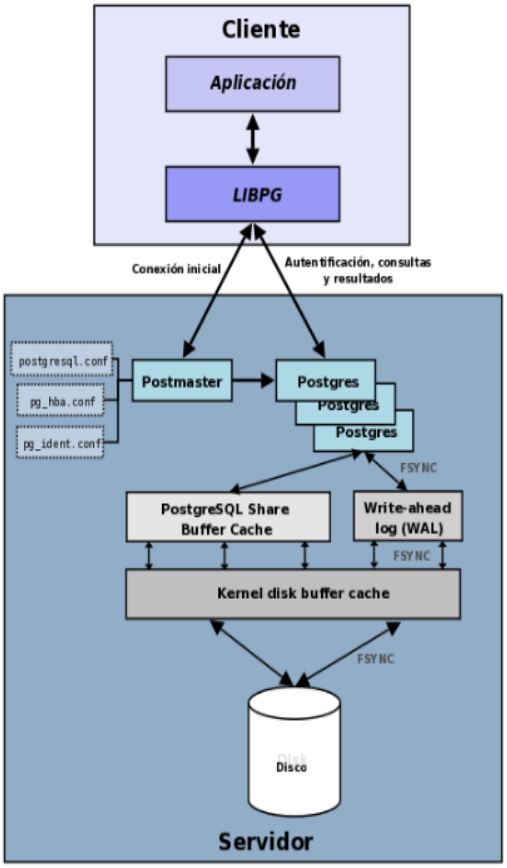


Figura 9: Componentes de un sistema PostgreSQL
Fuente: (Postgresql Org)

2.2.4 Servicio Web

"Un servicio web (WS de sus siglas en inglés) es un servicio ofrecido por una aplicación que expone su lógica a clientes de cualquier plataforma mediante una interfaz accesible a través de la red utilizando protocolos de internet. Un servicio web específica una interfaz de software que describe un conjunto de operaciones a las cuales se puede acceder por la red a través de mensajería XML, usa protocolos basados en XML para describir una operación a ejecutar o intercambiar datos con otro servicio web. Un grupo de servicios web que interactúa de esta forma define la aplicación de un servicio web específico en una arquitectura orientada a servicios (SOA)" (Quishpe, Rivero, & Rivas, 2016).

Servicio Web

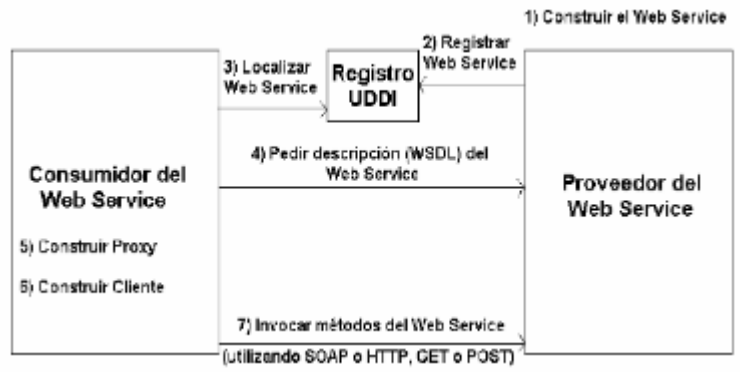


Figura 10: Funcionamiento de un Servicio Web
Fuente: (Chanchí Golondrino, Arciniegas Herrera, & Campo Muñoz, 2015)

Servicio web basados en Rest "Representational State Transfer (REST, por sus siglas en inglés), plantea una arquitectura cliente-servidor, en la cual un servicio es visto como un recurso y es identificado a través de una dirección Uniform Resource Locator (URL, por sus siglas en inglés), mediante la cual puede ser consumido. REST define a partir de HTTP, cuatro métodos: GET, PUT, DELETE y POST, de los cuales los más utilizados son: GET y PUT. El primero de los métodos es usado para enviar la representación de un recurso o servicio al cliente, mientras que el otro es usado para transferir el estado de un cliente al recurso. Para el intercambio de información entre cliente y servidor a través de REST, se puede hacer uso de diversos formatos y lenguajes: XML, HTML, JSON" (Chanchí Golondrino, Arciniegas Herrera, & Campo Muñoz, 2015).

JSON "Es un formato ligero basado en de texto, cuya sintaxis es tomada de JavaScript. Debido a su sencillez es fácil generar y procesar un documento con este formato. JSON usa convenciones para el manejo de datos, que son comunes a la familia de lenguajes: C, C++, Java, Perl, Python, etc. Lo anterior hace que JSON sea ideal para el intercambio de datos entre aplicaciones cliente servidor" (Chanchí Golondrino, Arciniegas Herrera, & Campo Muñoz, 2015).

Archivo .war o Web Archive, es un paquete de empaquetamiento y despliegue de los componentes de una aplicación web. “Se trata de un archivo de empaquetamiento destinado al despliegue de la aplicación Web en el servidor de aplicación. Sera el contenedor Web del servidor de aplicación quien se encargue de la puesta en marcha de la aplicación Web” (Aumaille, 2002, pág. 29).

2.2.5 Maven

“Maven es una herramienta principalmente utilizada en el desarrollo de software Java. Aparece ante la necesidad de modelar el concepto de "proyecto" y artefacto en forma estándar independiente del IDE de desarrollo” (Programación de Interfaces de Usuario, s.f.), además se basa en un tipo de objeto de proyecto (POM).

Archivo POM sus siglas en ingles significan Project Object Model o Proyecto de modelo de objetos, se convierte en el archivo base para el trabajo en Maven, siendo un archivo XML nombrado como pom.xml dentro del proyecto, que contiene las dependencias y plugins para llevar acabo el trabajo.

Las declaraciones principales que se realizan en este archivo son:

groupId: representa la organización autora/dueña del artefacto. Por ejemplo: com.ibm, org.apache, org.jboss, etc.

Artefacto: nombre del proyecto/artefacto actual. Por ejemplo: http-client, hibernate, tomcat, commons-collections, etc.

gVersión: como su nombre dice, identifica al número de versión del artefacto. (Programación de Interfaces de Usuario, s.f.)

Maven utiliza dependencias que no son más que librerías, plugins o artefactos que se utilizaran dentro del proyecto buscando dentro de todos los repositorios remotos o locales a los que tiene acceso.

2.2.6 Apache Tomcat 7

Tomcat es un contenedor de servlets, de Java que se usa como un servidor de aplicaciones web, su código es abierto.

Arquitectura de Tomcat

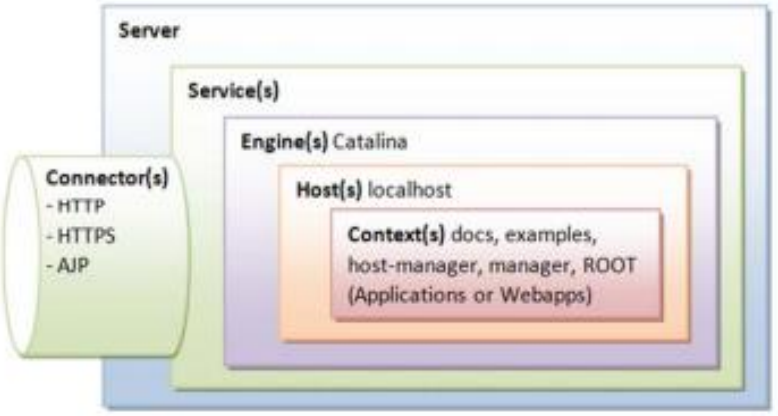


Figura 11: Arquitectura y organización de Apache Tomcat
Fuente: (Pavón Maestras, 2012)

Organización de directorios: Para un mejor manejo y ejecución de Tomcat este se organiza en directorios, siendo algunos de ellos bin directorio en el cual se encuentran los archivos ejecutables y scripts de Tomcat, conf contiene la configuración global para las aplicaciones aquí a desarrollar, web apps contienen los archivos para las aplicaciones web, etc.

Aplicaciones Web, dentro de Apache Tomcat las aplicaciones de este tipo se organizan en diversos directorios como:

Directorio raíz de la aplicación Web: Ficheros HTML, JSP, CSS, JS, imágenes, etc. que son visibles a los clientes de la aplicación

/WEB-INF/web.xml: Fichero XML que describe los servlets y otros componentes de la aplicación, además de parámetros de inicialización y restricciones de seguridad.

/WEB-INF/class/: Clases Java y recursos asociados: Servlets y no servlets que no estén contenidos en ficheros JAR.

/WEB-INF/lib/: "Ficheros JAR: Librerías de clases, drivers JDBC, etc." (Pavón Maestras, 2012).

2.2.7 Java Persistence API

"JPA o Java Persistence API es el estándar de Java encargado de automatizar dentro de lo posible la persistencia de nuestros objetos en base de datos" (Álvarez Caules , 2013)

Persistence.xml, este fichero se encuentra alojado en el directorio META-INF, es el encargado de llevar a cabo la conexión con la base de datos, además de definir las entidades que se van a gestionar.

Este archivo es de importancia dentro de Java Persistence API debido que permite llevar a cabo la persistencia de cada uno de los objetos de la base de datos y las clases contenedoras de la información de cada una de las tablas, cabe recalcar que dentro de este archivo se utiliza las librerías previamente exportadas para su uso dependiendo de las necesidades del desarrollador.

Al ser un archivo .xml se utiliza para llevar a cabo la persistencia sin importar el lenguaje en el que se esté trabajando.

Persistence.xml

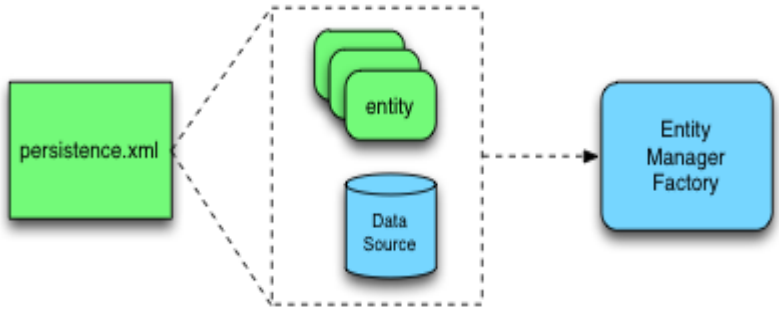


Figura 12: Diagrama de manejo de entidades, Persistence.xml
Fuente: (Álvarez Caules , 2013)

EntityManagerFactory, permite gestionar las entidades de nuestra base de datos, a través del objeto tipo EntityManager creado.

EntityManager

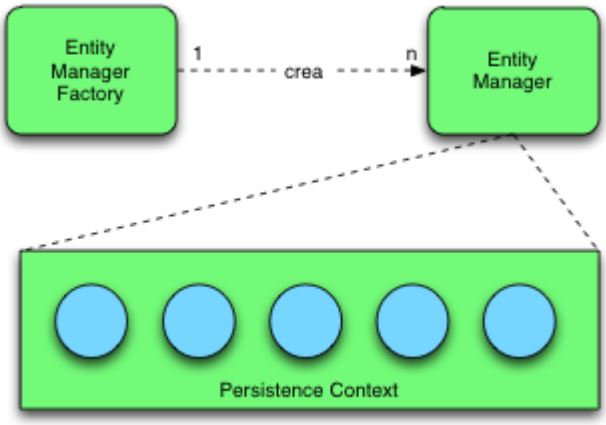


Figura 13: Manejo de entidades, EntityManager
Fuente: (Álvarez Caules , 2013)

PersistenceContext, objetos creados bajo el dominio de EntityManeger y que son manipulados y controlados a través del mismo.

Capítulo 3

Análisis, Diseño y Desarrollo

3.1 Análisis

3.1.1 Análisis del problema

La Zona 9 del Ministerio de Salud Publica en la actualidad cuenta con un formulario impreso para el levantamiento de información de inventario de la ficha familiar convirtiéndose en una actividad tediosa y repetitiva debido que toda la información se la llena de manera manual, lo que ocasiona retrasos, incompatibilidad, inentendible y pérdida de la información, lo que ocasiona tener información actualizada para la toma de decisiones acertadas para el desarrollo de actividades en bien de la comunidad.

A partir de este formulario se estableció tiempos de desarrollo, prioridades, requerimientos funcionales y no funcionales definiendo así las tareas y actividades de desarrollo de la aplicación móvil aplicando las metodologías Scrum y UWE.

3.1.2 Roles

Tabla 1: Definición de Roles

Rol	Responsable
Administrador del proyecto	Ing. René Arévalo
Dueño del producto	Econ. Nancy Gallegos
Maestro Scrum	Ing. Gabriela Dávila
Equipo de trabajo	Henry Avila Natalia Cueva
Equipo de pruebas	Responsable área de TI MSP zona 9

Nota: Descripción de las personas asignadas para cumplir cada rol dentro de la metodología SCRUM

En la siguiente tabla se define cada uno de los interesados en el desarrollo de la aplicación móvil para el levantamiento de la información de la ficha familiar del MSP, además de identificar sus intereses y la influencia de tal manera que se los gestione de la manera adecuada para concluir con éxito el desarrollo de la app.

Tabla 2: Interesados *(*Stakeholders*)*

#	Nombre	Cargo	Contacto	Intereses	Influencia
1	Econ. Nancy Gallegos	Directora del Departamento de Planificación Zona 9 Ministerio de Salud Pública	nancy.gallegos@mspz9.gob.ec	Automatizar el ingreso, actualización y consulta la ficha familiar del Ministerio de Salud Pública en un dispositivo móvil.	10
2	Ing. Gabriela Dávila	Analista del Departamento de Planificación Zona 9 Ministerio de Salud Pública	0987606344 gabriela.davila@mspz9.gob.ec	Conectar la aplicación móvil a un servidor dentro de la institución. Descargar formularios del servidor al dispositivo móvil. Exportar la información recolectada de las fichas familiares del dispositivo móvil al servidor.	10
3	Henry Avila Natalia Cueva	Miembros del proyecto de titulación Estudiantes de la Universidad Politécnica Salesiana	09940299314 havilaj@est.ups.edu.ec 09995396969 ncuevac@est.ups.edu.ec	Crear una aplicación móvil sobre el Sistema Operativo Android que permita la automatización de la ficha familiar del ministerio de Salud Pública Implementar proyecto de titulación para la obtención del título de ingeniero de sistemas.	10
4	René Arévalo	Tutor / Docente	0995459598 *aarevalo@ups.edu.ec*	Evaluar el desarrollo y calidad del proyecto presentado	10
5	Médico	Unidades Operativas	023931020 *www.salud.gob.ec*	Uso de la aplicación móvil	9

3.1.3 Pila del producto

Tabla 3: Pila del producto

Elemento	Descripción
Identificador del requerimiento	1
Descripción del requerimiento	Especificación de requerimientos con el cliente
Priorización	20
Estimación del esfuerzo	15 días
Observaciones	Establecer los requerimientos del sistema en reuniones conjuntas con el cliente y los desarrolladores
Persona asignada	Henry Avila/ Natalia Cueva
# de Sprint ejecuta	1
Módulo del sistema al que pertenece	Requerimientos
Identificador del requerimiento	2
Descripción del requerimiento	Diseño de la aplicación móvil
Priorización	5
Estimación del esfuerzo	6 días
Observaciones	Diseñar los diagramas de casos de uso, base de datos, diagrama de navegación, diccionario de datos, prototipo de interfaz de usuario para la creación de la aplicación móvil.
Persona asignada	Henry Avila/ Natalia Cueva
# de Sprint ejecuta	2
Módulo del sistema al que pertenece	Aplicación móvil
Identificador del requerimiento	3
Descripción del requerimiento	Insertar la información de una nueva ficha familiar, los usuarios deberán registrar la información requerida en la ficha familiar del MSP.
Priorización	5
Estimación del esfuerzo	3 días
Observaciones	La app permitirá al usuario registrar la información de la ficha familiar. El usuario debe ingresar los datos tales como: unidad operativa, coordenadas de ubicación de la vivienda, manzana, edificio, datos de la familia, datos de personas fallecidas, etc.
Persona asignada	Henry Avila/ Natalia Cueva
# de Sprint ejecuta	2
Módulo del sistema al que pertenece	Aplicación móvil
Identificador del requerimiento	4
Descripción del requerimiento	Descargar (Importar) información de la ficha

	familiar, el sistema ofrecerá al usuario la posibilidad de consulta de las fichas ya registradas en el servidor.
Priorización	5
Estimación del esfuerzo	5 días
Observaciones	La app permitirá al usuario consultar información de las fichas familiares ya ingresadas en el servidor. El usuario debe ingresar los siguientes datos tales como: número de formulario o un rango de números.
Persona asignada	Henry Avila/ Natalia Cueva
# de Sprint ejecuta	3
Módulo del sistema al que pertenece	Integración
Identificador del requerimiento	5
Descripción del requerimiento	Exportar información de la ficha familiar, el sistema ofrecerá al usuario la posibilidad de almacenar la información de las fichas familiares levantadas en el servidor de la institución.
Priorización	5
Estimación del esfuerzo	5 días
Observaciones	La app permitirá al usuario insertar información de las fichas familiares ya ingresadas en el dispositivo móvil al servidor de la institución.
Persona asignada	Henry Avila/ Natalia Cueva
# de Sprint ejecuta	3
Módulo del sistema al que pertenece	Integración
Identificador del requerimiento	6
Descripción del requerimiento	Actualizar catálogos, al sincronizar con el servidor los usuarios podrán actualizar información de los catálogos.
Priorización	5
Estimación del esfuerzo	7 días
Observaciones	Permite al usuario actualizar los catálogos del formulario de la ficha familiar.
Persona asignada	Henry Avila/ Natalia Cueva
# de Sprint ejecuta	3
Módulo del sistema al que pertenece	Aplicación móvil
Identificador del requerimiento	7
Descripción del requerimiento	Actualizar la información de ficha familiar nueva, los usuarios podrán actualizar la información requerida en la ficha familiar del

	MSP.
Priorización	5
Estimación del esfuerzo	5 días
Observaciones	Permite al usuario modificar los datos de las fichas familiares almacenadas en el dispositivo, para dar el seguimiento a la misma.
Persona asignada	Henry Avila/ Natalia Cueva
# de Sprint ejecuta	2
Módulo del sistema al que pertenece	Aplicación móvil
Identificador del requerimiento	8
Descripción del requerimiento	Interfaz de la app, el sistema presentara una interfaz de usuario sencilla para que sea de fácil manejo a los usuarios del sistema. El sistema deberá de tener una interfaz de usuario, teniendo en cuenta las características de la ficha familiar física del MSP.
Priorización	5
Estimación del esfuerzo	3 días
Observaciones	El sistema debe tener una interfaz intuitiva y sencilla y ajustarse a las características de la ficha familiar del MSP. Véase anexo 1 ficha familiar física
Persona asignada	Henry Avila/ Natalia Cueva
# de Sprint ejecuta	2
Módulo del sistema al que pertenece	Aplicación móvil
Identificador del requerimiento	9
Descripción del requerimiento	Creación de la documentación
Priorización	5
Estimación del esfuerzo	20 días
Observaciones	Creación de los manuales de usuario, instalación y técnico. Documentación para la presentación en la unidad de titulación
Persona asignada	Henry Avila/ Natalia Cueva
# de Sprint ejecuta	4
Identificador del requerimiento	10
Descripción del requerimiento	Investigación de información complementaria requerida en la aplicación
Priorización	5
Estimación del esfuerzo	5 días

Observaciones	Investigar información de provincias, cantos, parroquias del Ecuador en la página del Instituto Ecuatoriano de Estadísticas y Censos (INEC).
Persona asignada	Henry Avila/ Natalia Cueva
# de Sprint ejecuta	1
Módulo del sistema al que pertenece	Requerimientos
Identificador del requerimiento	11
Descripción del requerimiento	Llenado de listas desplegables de la ficha familiar
Priorización	5
Estimación del esfuerzo	4 días
Observaciones	Conexión con la base de datos SQLite y llenado de listas desplegables del formulario
Persona asignada	Henry Avila/ Natalia Cueva
# de Sprint ejecuta	2
Módulo del sistema al que pertenece	Requerimientos

Elaborado por: Henry Avila, Natalia Cueva

3.1.4 Plan de trabajo/Sprints

Para el desarrollo de la aplicación móvil se definieron 4 sprints en los cuales se llevará a cabo distintas tareas para el cumplimiento del objetivo de nuestro trabajo.

Tabla 4: Detalle de actividades Sprint I

Sprint I: Levantamiento de Requerimientos	
Duración: 4 semanas	
Actividad	**Tiempo**
Definición de requerimientos por parte de los stakeholders	3 días
Reunión con el Dueño del producto	2 días
Establecer información a almacenar	3 días
Definición de la información a utilizar de la ficha familiar física	2 días
Investigar información de provincias, cantos, parroquias del Ecuador en la página del Instituto Ecuatoriano de Estadísticas y Censos (INEC)	2 días
Definición de requisitos para el uso de la aplicación	1 día
Determinación de las herramientas para el desarrollo	1 día
Preparación del documento para la especificación de requerimientos	3 días

Correcciones del documento de especificación de requerimientos	2 días

Elaborado por: Henry Avila, Natalia Cueva

Tabla 5: Detalle de actividades Sprint II

Sprint II: Creación de la aplicación móvil	
Duración: 4 semanas	
Actividad	**Tiempo**
Definición de los bloques del formulario, prototipo	1 día
Creación del modelo conceptual y físico de la base de datos	2 días
Creación de los casos de uso y diccionario de datos	2 días
Definición de listas desplegables (catálogos), campos de ingreso de texto, botones, etc.	1 día
Creación de la base de datos en SQLite	1 día
Creación de las pantallas del formulario en Android Studio	3 días
Conexión y carga de datos en las listas desplegables utilizando BDD SQLite	4 día
Almacenamiento de la información en la BDD SQLite	3 días
Consulta y modificación de la información	3 días
Pruebas y correcciones	2 días

Elaborado por: Henry Avila, Natalia Cueva

Tabla 6: Detalle de actividades Sprint III

Sprint III: Integración de la aplicación con el servidor	
Duración: 4 semanas	
Actividad	**Tiempo**
Creación de la pantalla para la recolección de información para la conexión con el servidor	1 día
Creación de la base de datos en PostgreSQL	1 día
Creación Web Service para la conexión con el servidor y actualización de catálogos	4 días
Creación Web Service para la exportación de los formularios al servidor	4 días
Creación Web Service para la importación de los formularios al dispositivo móvil	4 días

Pruebas y correcciones	3 días

Elaborado por: Henry Avila, Natalia Cueva

Tabla 7: Detalle de actividades del Sprint IV

Sprint IV: Generación de la documentación/ Pruebas de ambiente y producción	
Duración: 4 semanas	
Actividad	**Tiempo**
Creación del archivo. Apk	1 día
Creación del manual de usuario	1 día
Creación del manual de instalación	1 día
Creación del manual técnico y documentación de titulación	12 días
Pruebas de ambiente	3 días
Pruebas de producción	2 días

Elaborado por: Henry Avila, Natalia Cueva

3.1.5 Análisis de la base de Datos

Las tablas principales de la base de datos son: formulario en el cual se almacena la información referente a la vivienda (bloque 1 al 5 de la ficha familiar), mortalidad almacena información referente a las personas fallecidas del hogar (bloque 6 ficha familiar) y personas la información almacenada refiere a las personas que habitan la vivienda (bloque 6 y 7 de la ficha familiar), estas dos últimas tablas se encuentran ligadas a la tabla formulario.

Cada una de estas almacena la clave principal de cada una de las tablas catálogos las cuales se explicará posteriormente estas claves son utilizadas como claves foráneas, para llevar un control adecuado en el almacenamiento de la información y que de esta manera no exista inconsistencia en la misma para que las decisiones se tomen en base a información adecuada.

La información aquí almacenada es la registrada en cada una de las salidas al campo por parte del personal encargado.

Tablas principales de la Base de Datos

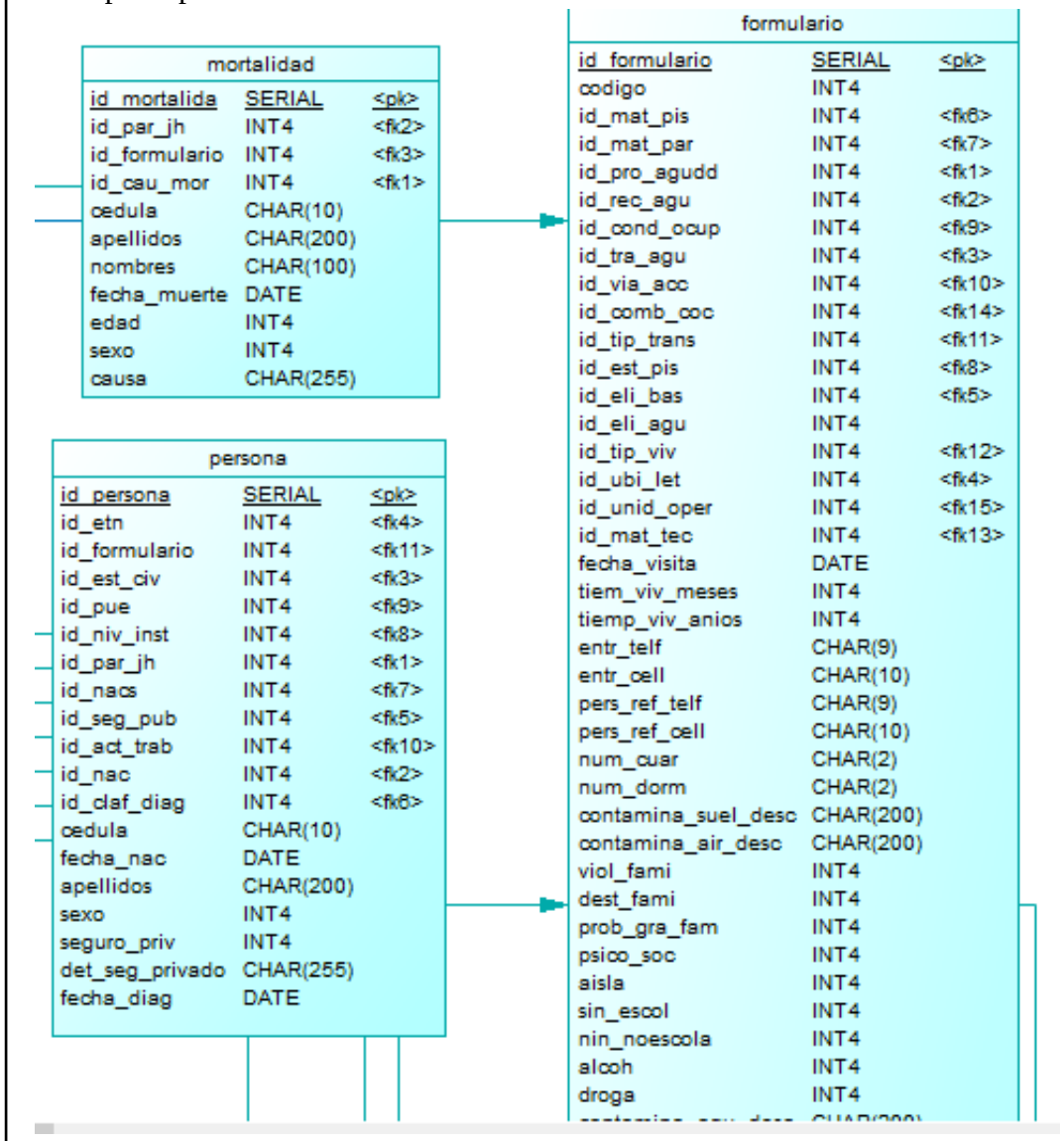


Figura 14: Tablas Formulario, Mortalidad y Persona
Elaborado por: Henry Avila, Natalia Cueva

Las listas desplegables de la ficha familiar o para nuestro desarrollo llamada inventario se denominarán catálogos y son todas aquellas tablas que contengan información de selección por parte del usuario, ejemplo: cond_ocup de la vivienda, tip_mat_techo, etc.

La información almacenada en estas tablas consta de una clave principal la cual es asignada con fines de desarrollo, además del código usado por parte del MSP,

descripción del campo y un campo que determina si este registro se encuentra en uso.

Ejemplo tablas catálogos

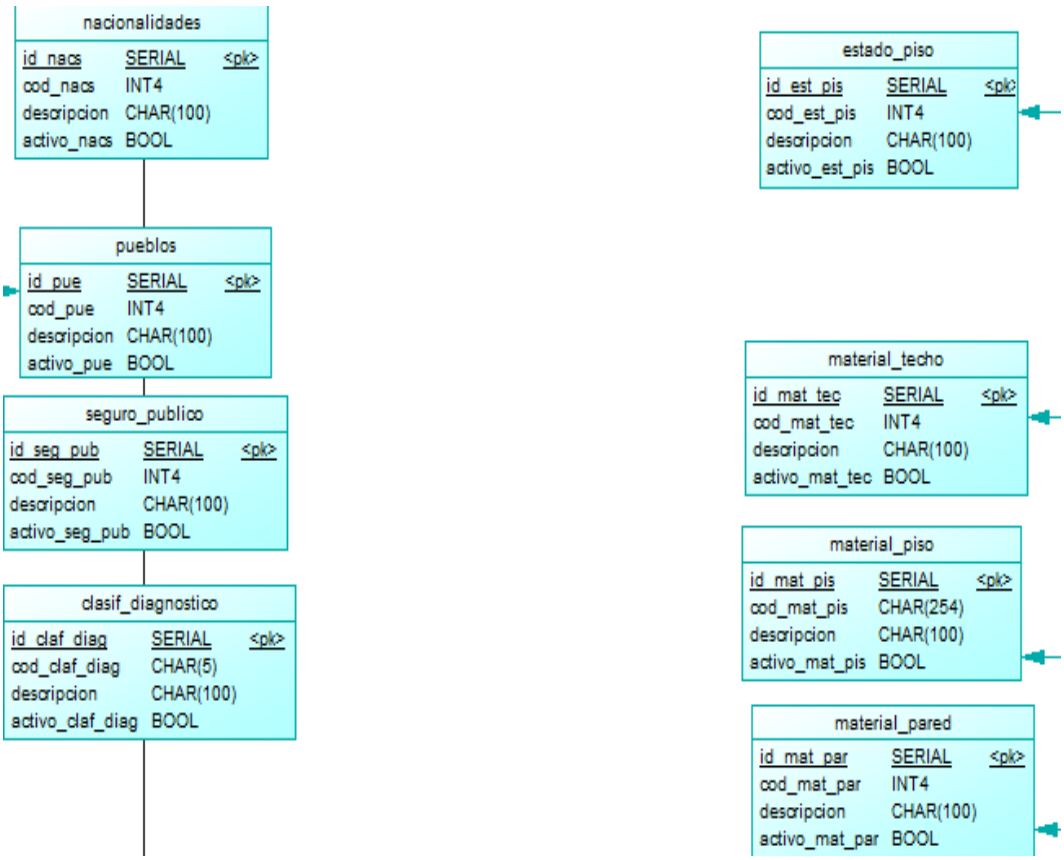


Figura 15: Ejemplo de catálogos
Elaborado por: Henry Avila, Natalia Cueva

Para determinar la ubicación de la unidad operativa es necesario establecer la provincia, cantón, parroquia, administración zonal y distrito por lo cual se crean las tablas con nombres similares relacionándose entre ellas y siendo su cabeza la tabla unidad_operativa que se relaciona con la tabla formulario.

Diagrama de la relación para la ubicación de la Unidad Operativa

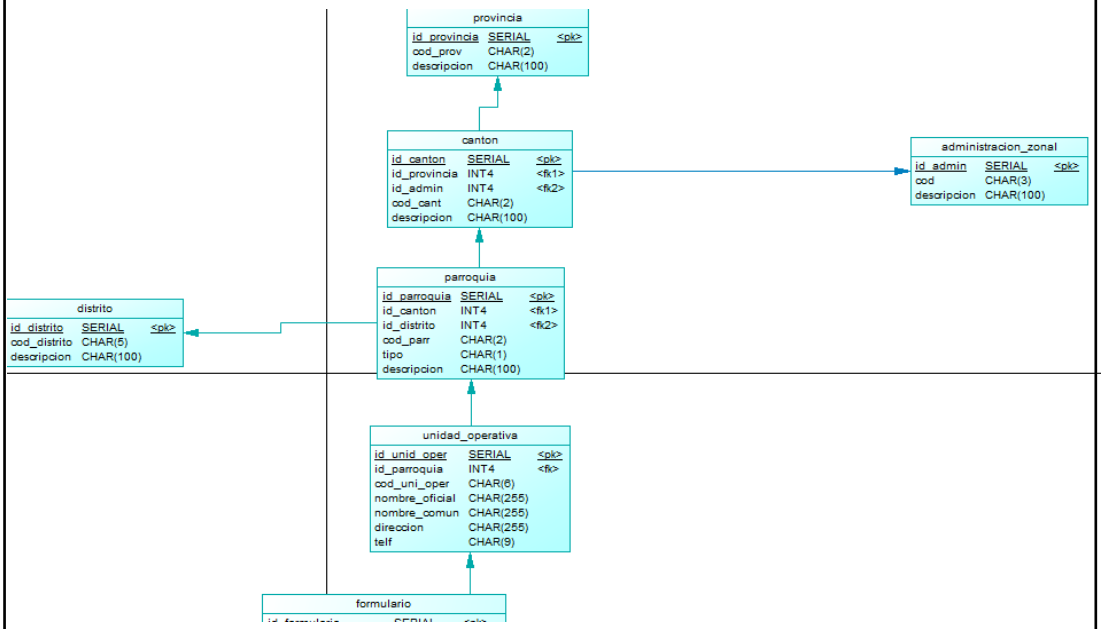


Figura 16: Diagrama Físico de la relación de la ubicación de la Unidad Operativa
Elaborado por: Henry Avila, Natalia Cueva

Esta estructura de base de datos se utilizará para la creación en PostgreSQL y SQLite para tener una estructura similar de las bases de datos y facilitar la integración de la aplicación con el servidor.

3.1.6 Análisis de la interfaz

Por petición expresada por el cliente en el documento de especificación de requerimientos, en la interfaz de ingreso de la ficha familiar se mantiene similitud en el formato de la ficha familiar física que maneja el Ministerio de Salud Pública.

A continuación se describe la interfaz de ingreso de la ficha familiar debido que el objetivo principal es automatizar el levantamiento de la información de la ficha familiar, la cual está compuesta de tres pantallas siendo la primera pantalla contenedora de la información de ingreso de la ficha familiar bloque 1 al 5 (Véase Anexo1) ubicados en la parte inferior tres botones para el almacenamiento de este bloque y los otros dos botones para el llamado a las otras pantallas , la segunda

pantalla miembros del hogar refiere al bloque 6 y 7 (Véase Anexo1) y la última pantalla bloque 6 de personas fallecidas (Véase Anexo1), además de un menú principal para llevar contenedor de las opciones de la aplicación.

Pantalla Menú de la aplicación

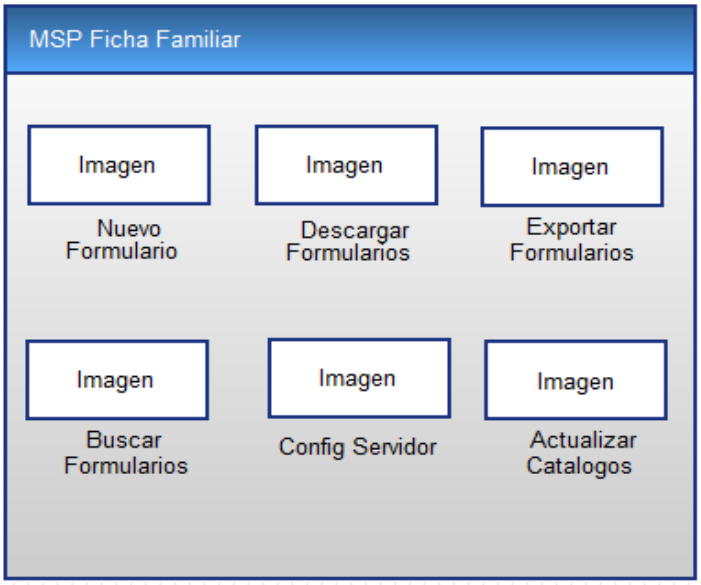


Figura 17: Prototipo interfaz pantalla menú de la aplicación
Elaborado por: Henry Avila, Natalia Cueva

Pantalla formulario de la aplicación

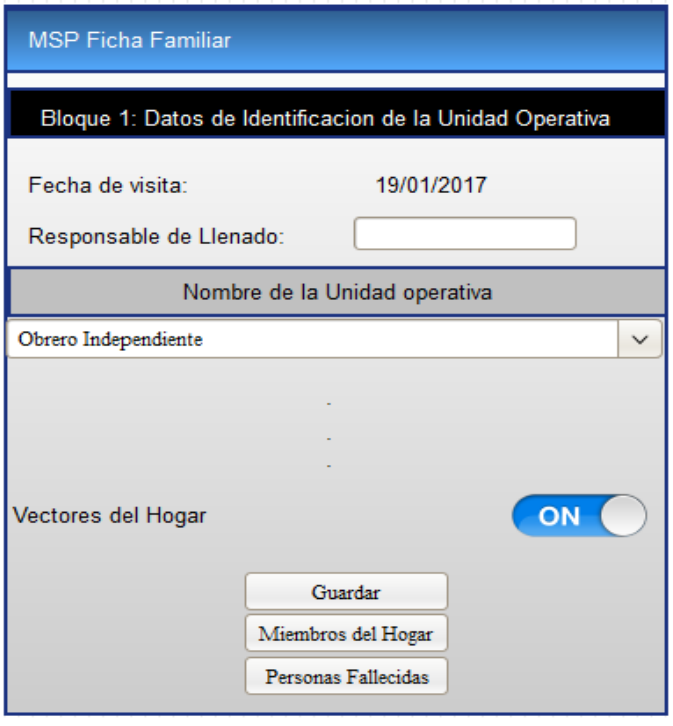


Figura 18: Pantalla bloque 1 al 5 ficha familiar
Elaborado por: Henry Avila, Natalia Cueva

Pantalla miembros del hogar de la aplicación

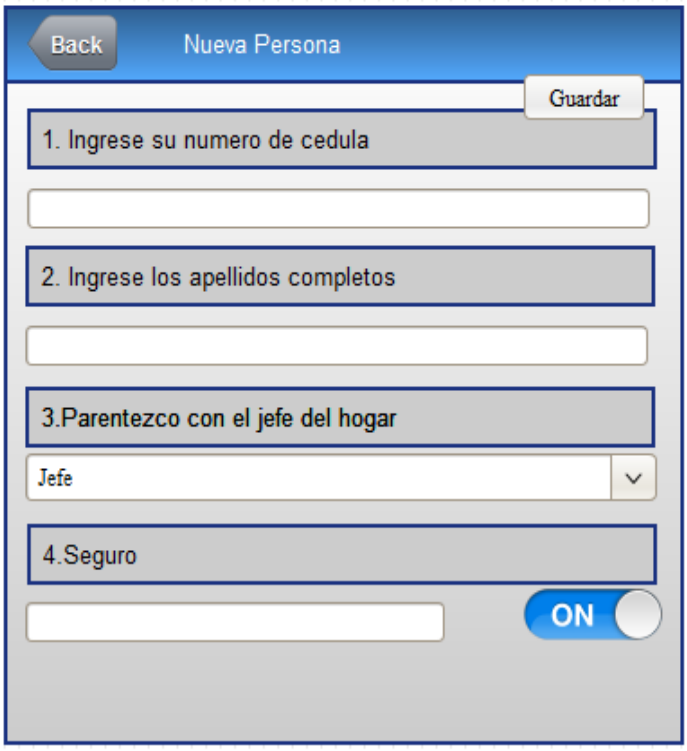


Figura 19: Pantalla Bloque 6 y 7 de la ficha familiar
Elaborado por: Henry Avila, Natalia Cueva

Pantalla personas fallecidas de la aplicación

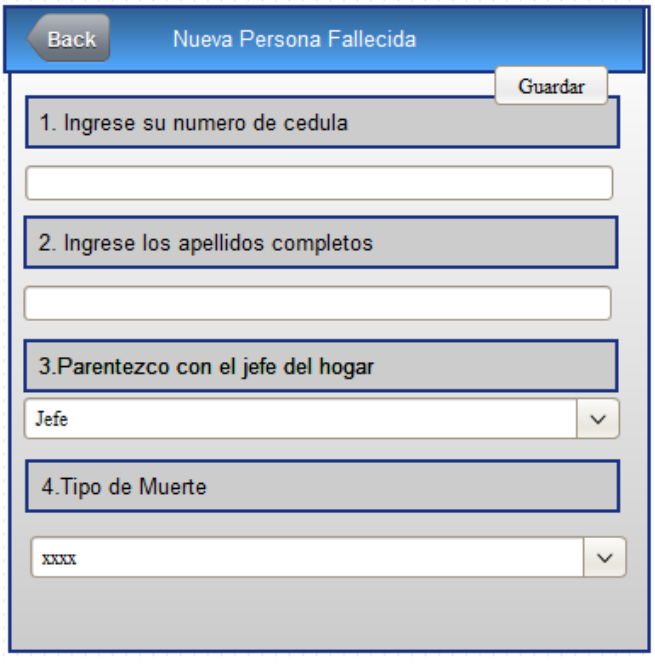


Figura 20: Pantalla Bloque 6 de la ficha familiar

Elaborado por: Henry Avila, Natalia Cueva

3.2 Diseño

Para el desarrollo del diseño de la aplicación se utiliza diagramas UML para la elaboración de los diagramas de casos de uso, diagramas de secuencia y diagramas de la base de datos.

3.2.1 Diagramas de casos de uso

El diagrama de caso de uso muestra los pasos a seguir para la actualización de catálogos de la aplicación móvil.

Diagrama de caso de uso actualizar catálogos

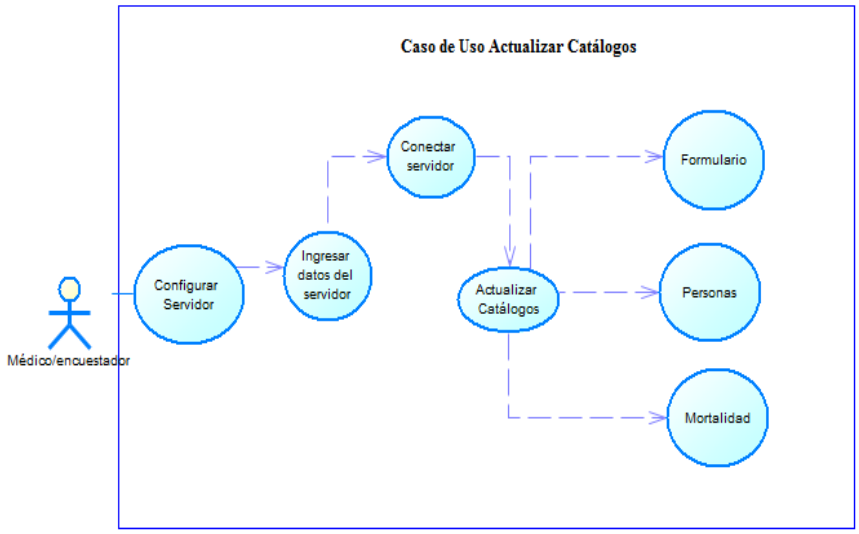


Figura 21: Diagrama de caso de uso actualizar catálogos
Elaborado por: Henry Avila, Natalia Cueva

Tabla 8: Descripción caso de uso actualizar catálogos

Descripción caso de uso: actualizar catálogos	
Actores:	Médico/ encuestador
Objetivo:	Configurar la conexión con el servidor y permitir editar la información del mismo. Actualizar los catálogos de la ficha familiar
Descripción:	• Ingresar datos de configuración del servidor • Almacenar la información • Seleccionar Actualizar Catálogos ➢ Formulario ➢ Persona ➢ Mortalidad

Diagrama de caso de uso formulario

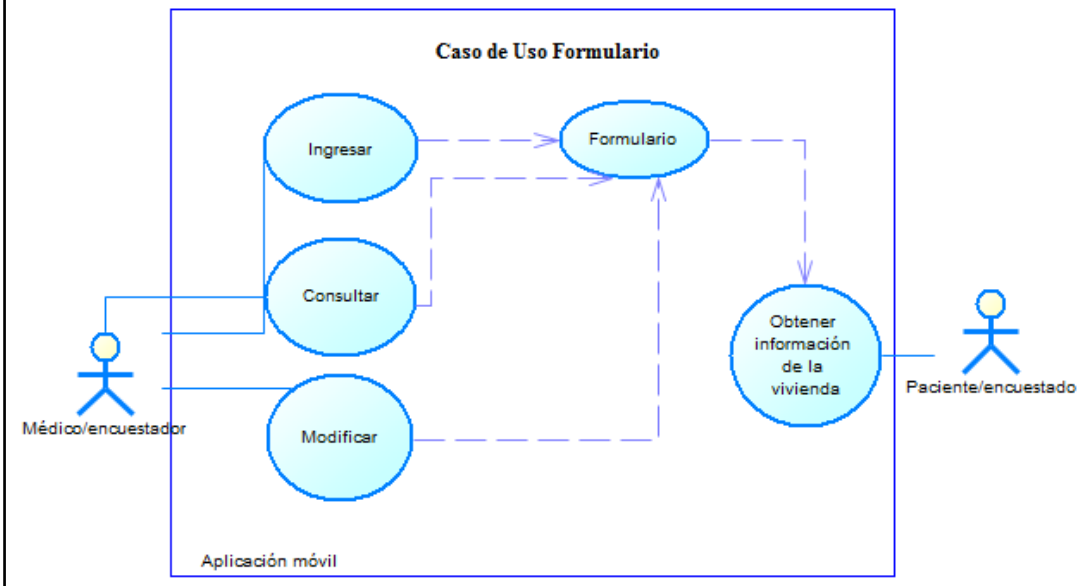


Figura 22: Diagrama de caso de uso formulario
Elaborado por: Henry Avila, Natalia Cueva

Tabla 9: Descripción caso de uso formulario

Descripción caso de uso: formulario	
Actores:	Medico/ encuestador, paciente
Objetivo:	Ingresar la información de la ficha familiar bloques 1 al 5, llamado formulario y permitir modificar o consultar la información. La información aquí ingresada se almacena en el dispositivo móvil para su posterior almacenamiento en el servidor previsto por el MSP.
Descripción:	Ingresar información de formulario. Consultar la información de formulario Modificar la información de formulario Obtener la información del paciente

Diagrama caso de uso persona

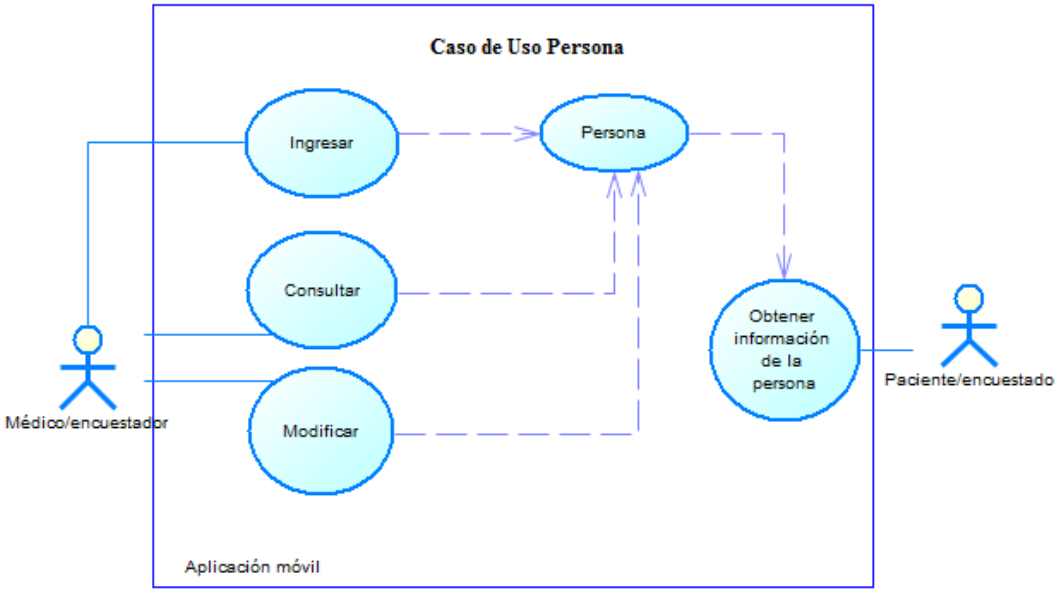


Figura 23: Diagrama caso de uso persona
Elaborado por: Henry Avila, Natalia Cueva

Tabla 10: Descripción caso de uso persona

Descripción caso de uso: persona	
Actores:	Medico/ encuestador, paciente
Objetivo:	Ingresar la información de la ficha familiar bloques 6, llamado persona y permitir modificar o consultar la información. La información aquí ingresada se almacena en el dispositivo móvil para su posterior almacenamiento en el servidor previsto por el MSP.
Descripción:	Ingresar información de persona. Consultar la información de persona Modificar la información de persona Obtener la información del paciente

Diagrama caso de uso mortalidad

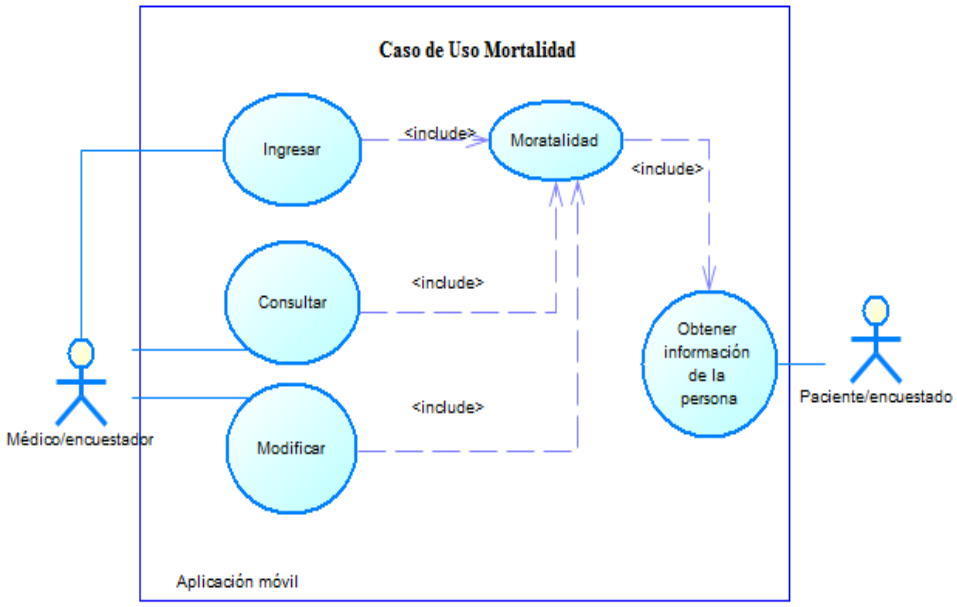


Figura 24: Diagrama caso de uso mortalidad
Elaborado por: Henry Avila, Natalia Cueva

Tabla 11: Descripción caso de uso mortalidad

Descripción caso de uso: mortalidad	
Actores:	Medico/ encuestador, paciente
Objetivo:	Ingresar la información de la ficha familiar bloques 6, llamado mortalidad y permitir modificar o consultar la información.
Descripción:	Ingresar información de mortalidad. Consultar la información de mortalidad Modificar la información de mortalidad Obtener la información del paciente

Los casos de uso que se describen a continuación permiten llevar a cabo la importación de los formularios al servidor que almacenara los datos, verificando la consistencia de la información aquí ingresada.

Diagrama caso de uso importar formulario

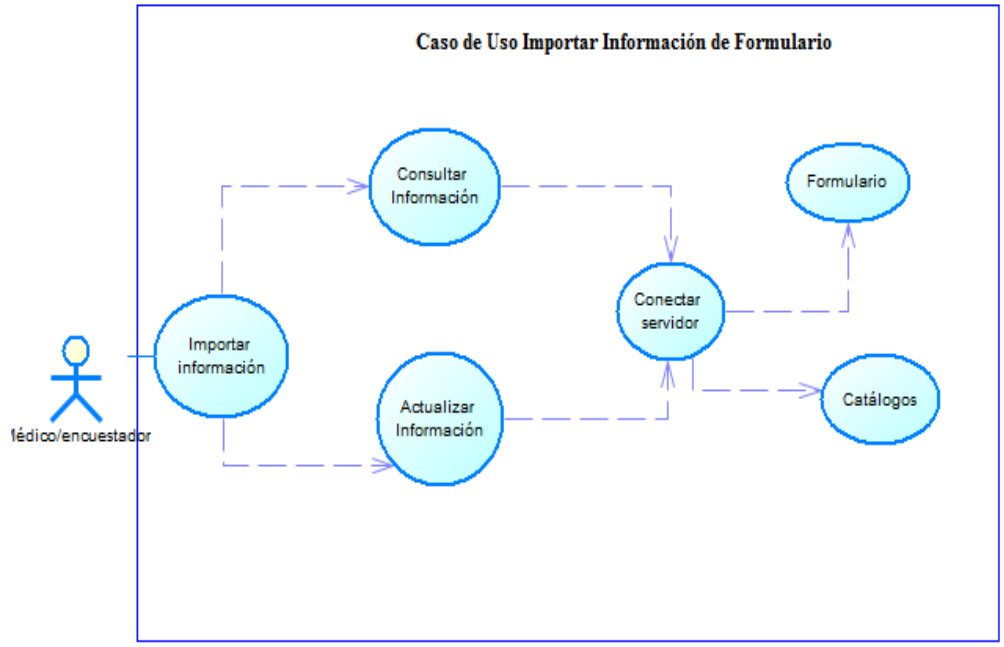


Figura 25: Diagrama caso de uso importar formulario
Elaborado por: Henry Avila, Natalia Cueva

Tabla 12: Descripción caso de uso importar formulario

Descripción caso de uso: importar información de formulario	
Actores:	Medico/ encuestador
Objetivo:	Importar la información almacenada en el servidor al dispositivo móvil
Descripción:	Importar información Consultar información Conectar Servidor Actualizar información Conectar servidor Formulario Catálogos

Diagrama caso de uso importar persona

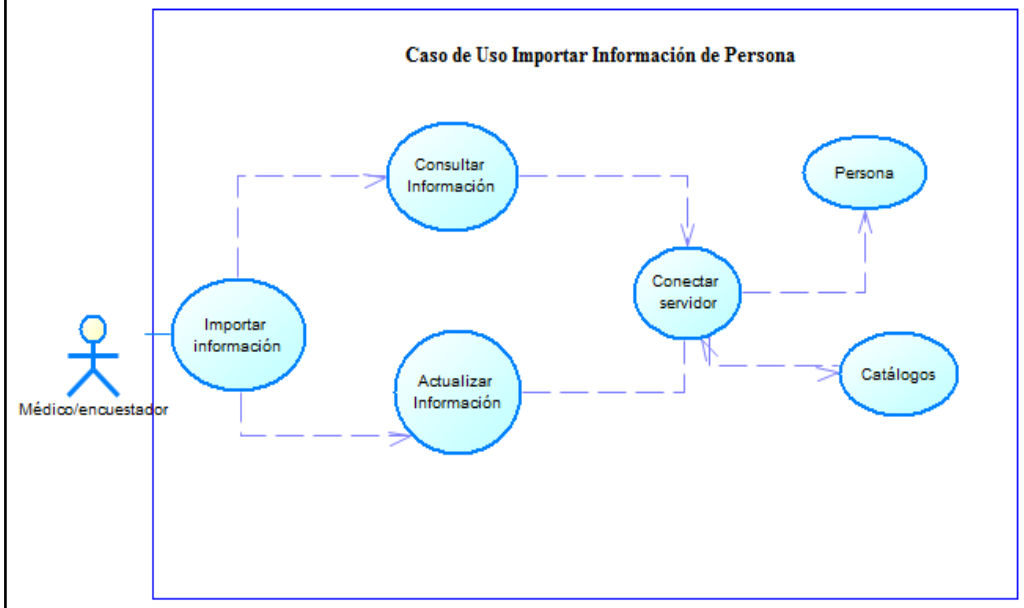


Figura 26: Diagrama caso de uso importar persona
Elaborado por: Henry Avila, Natalia Cueva

Tabla 13: Descripción caso de uso importar persona

Descripción caso de uso: importar información de persona	
Actores:	Medico/ encuestador
Objetivo:	Importar la información almacenada en el servidor al dispositivo móvil
Descripción:	Importar información Consultar información Conectar Servidor Actualizar información Conectar servidor Persona Catálogos

Diagrama caso de uso importar persona

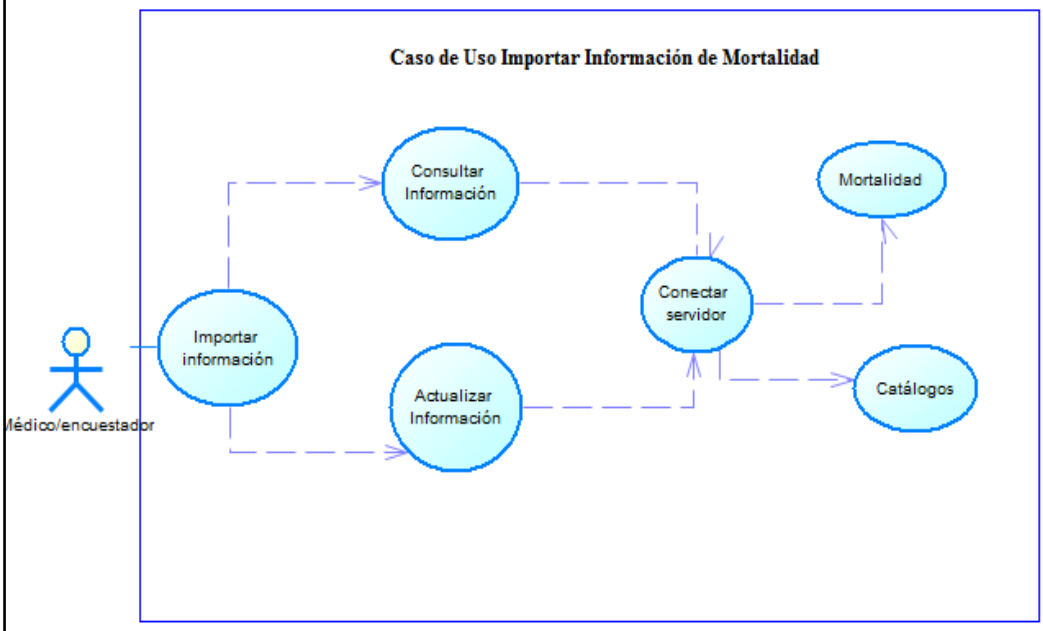


Figura 27: Diagrama caso de uso importar mortalidad
Elaborado por: Henry Avila, Natalia Cueva

Tabla 14: Descripción caso de uso importar mortalidad

Descripción caso de uso: importar información de mortalidad	
Actores:	Medico/ encuestador
Objetivo:	Importar la información almacenada en el servidor al dispositivo móvil
Descripción:	Importar información Consultar información Conectar Servidor Actualizar información Conectar servidor Mortalidad Catálogos

Diagrama caso de uso exportar formulario

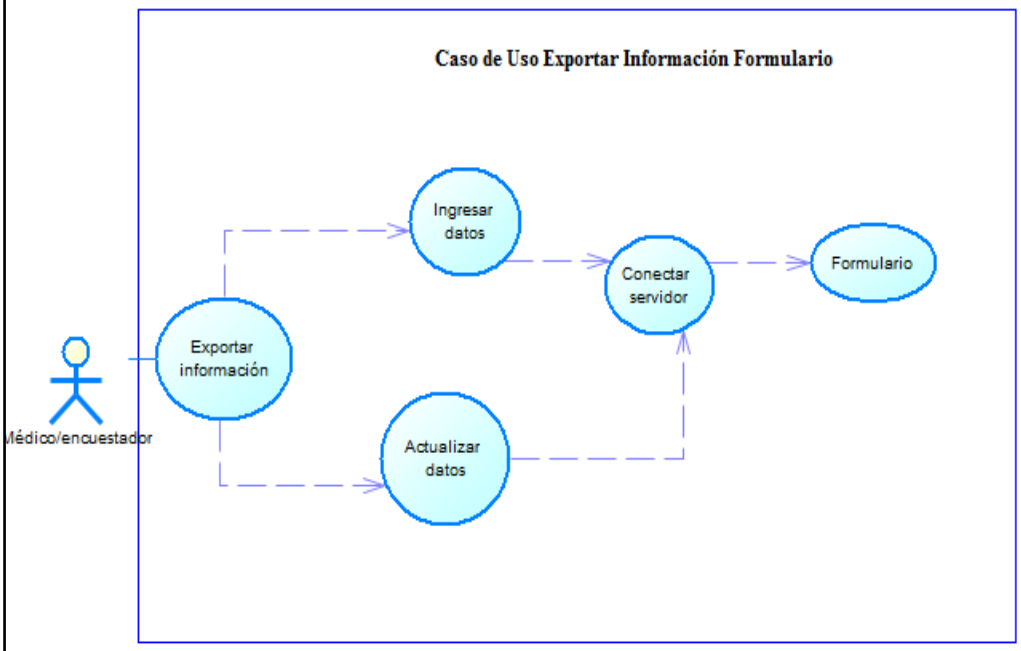


Figura 28: Diagrama caso de uso exportar formulario
Elaborado por: Henry Avila, Natalia Cueva

Tabla 15: Descripción caso de uso exportar formulario

Descripción caso de uso: exportar información de formulario	
Actores:	Medico/ encuestador
Objetivo:	Descargar la información almacenada en el dispositivo móvil al servidor
Descripción:	Exportar información Ingresar datos Conectar Servidor Actualizar datos Conectar servidor Formulario

Diagrama caso de uso exportar persona

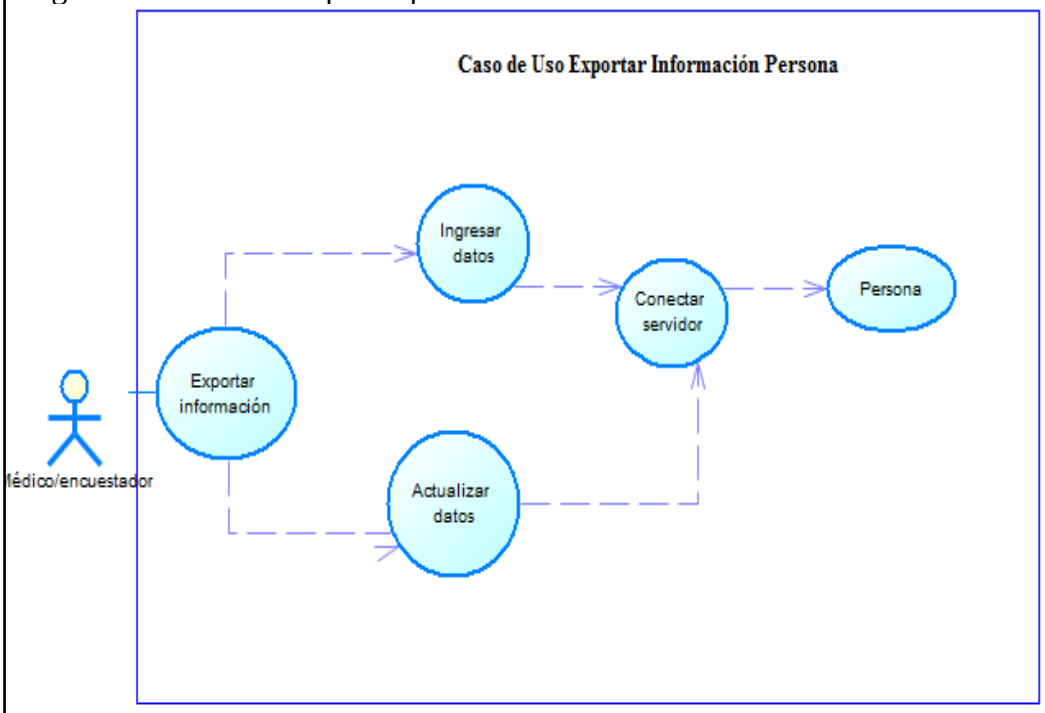


Figura 29: Diagrama caso de uso exportar persona
Elaborado por: Henry Avila, Natalia Cueva

Tabla 16: Descripción caso de uso exportar persona

Descripción caso de uso: exportar información de persona	
Actores:	Medico/ encuestador
Objetivo:	Descargar la información almacenada en el dispositivo móvil al servidor
Descripción:	Exportar información Ingresar datos Conectar Servidor Actualizar datos Conectar servidor Persona

Diagrama caso de uso exportar mortalidad

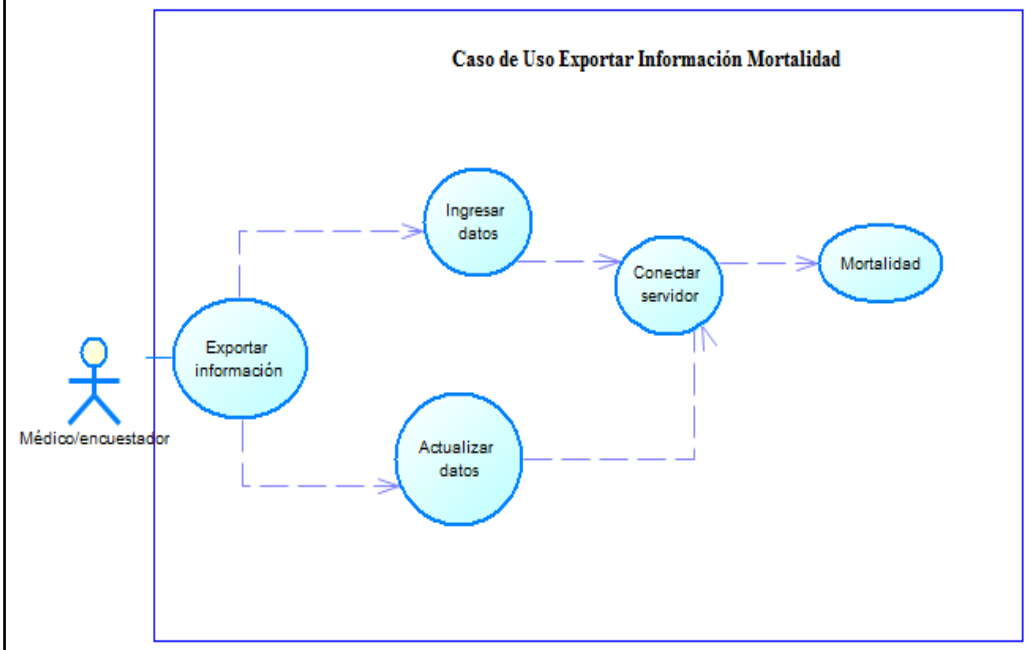


Figura 30: Diagrama caso de uso exportar mortalidad
Elaborado por: Henry Avila, Natalia Cueva

Tabla 17: Descripción caso de uso exportar mortalidad

Descripción caso de uso: exportar información de mortalidad	
Actores:	Medico/ encuestador
Objetivo:	Descargar la información almacenada en el dispositivo móvil al servidor
Descripción:	Exportar información Ingresar datos Conectar Servidor Actualizar datos Conectar servidor Mortalidad

3.2.2 Diagrama de secuencia

El diagrama de secuencia de la siguiente imagen indica el registro o ingreso de una nueva ficha familiar en la aplicación móvil, para comprender como esta se lleva a cabo en la aplicación móvil para su posterior almacenamiento en el servidor dispuesto por Ministerio de Salud Pública Zona 9, para verificar todos los diagramas de secuencia del sistema verificar el Anexo 4.

Diagrama de secuencia ingreso ficha familiar nueva

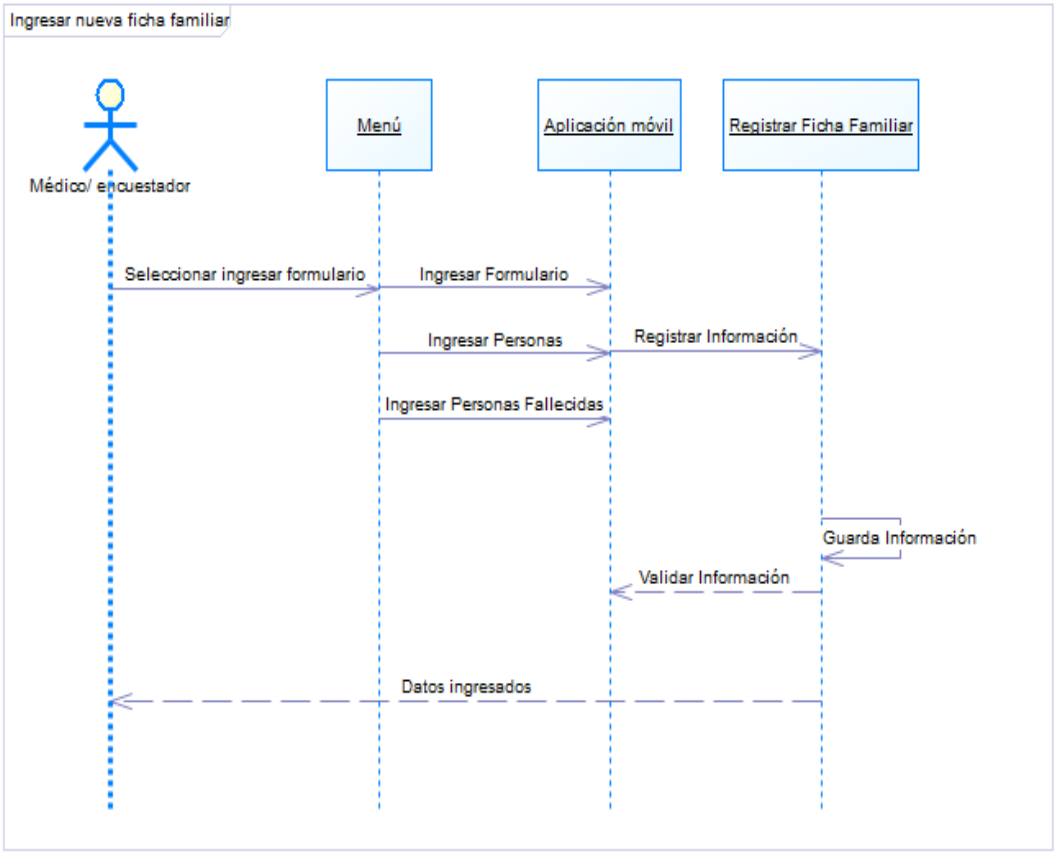


Figura 31: Diagrama de secuencia ingresar nueva ficha familiar
Elaborado por: Henry Avila, Natalia Cueva

3.2.3 Diagrama de navegación

El diagrama de navegación muestra la relación y el orden de las pantallas con el contenido de la aplicación móvil, se describe esta relación en la siguiente imagen, en donde el enlace directo está representado por una línea continua color azul e indica que primero se debe ingresar a las pantallas anteriores para el registro de la información.

El hipervínculo se encuentra definido dentro del diagrama por una línea entrecorda de color azul y define que si no se ingresó información a través de las pantallas anterior a esta no se puede llevar a cabo la operación descrita por esta opción dentro de la aplicación móvil, además que cada una de las opciones de la aplicación se encuentran enmarcada dentro de un rectángulo color azul.

Diagrama de navegación

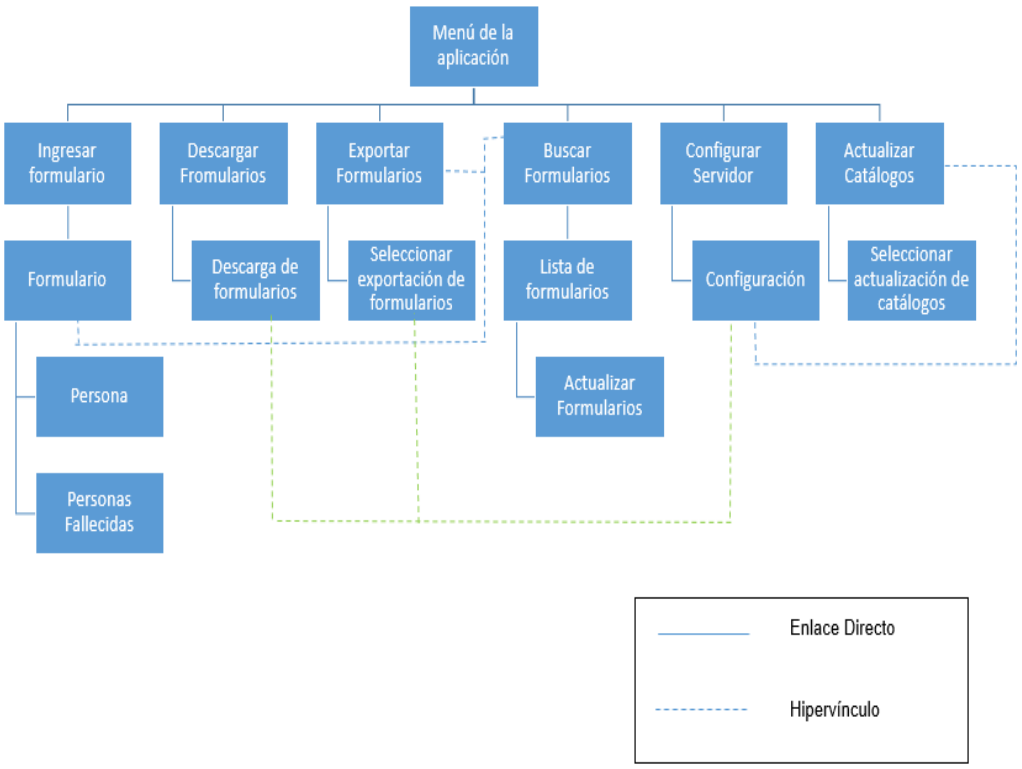


Figura 32: Diagrama de navegación
Elaborado por: Henry Avila, Natalia Cueva

3.2.4 Diagrama base de datos

Este diagrama es la referencia para la creación de la base de datos PostgreSQL y SQLite. Se Debe considerar que en la base de datos SQLite sirve de referencia en la creación de la tablas y campos, pero no en las relaciones entre estos debido que al ser una base de datos contenida en una librería y encontrarse en un dispositivo móvil su capacidad es reducida y la prioridad es el almacenamiento de datos para la posterior exportación a la base de datos creada en PostgreSQL, las claves primarias se definieron en cada una de las tablas para evitar la duplicidad de las mismas.

Al igual que la base de datos PostgreSQL se establece que las tablas principales son formulario, personas y mortalidad.

Diagrama Físico de la Base de Datos

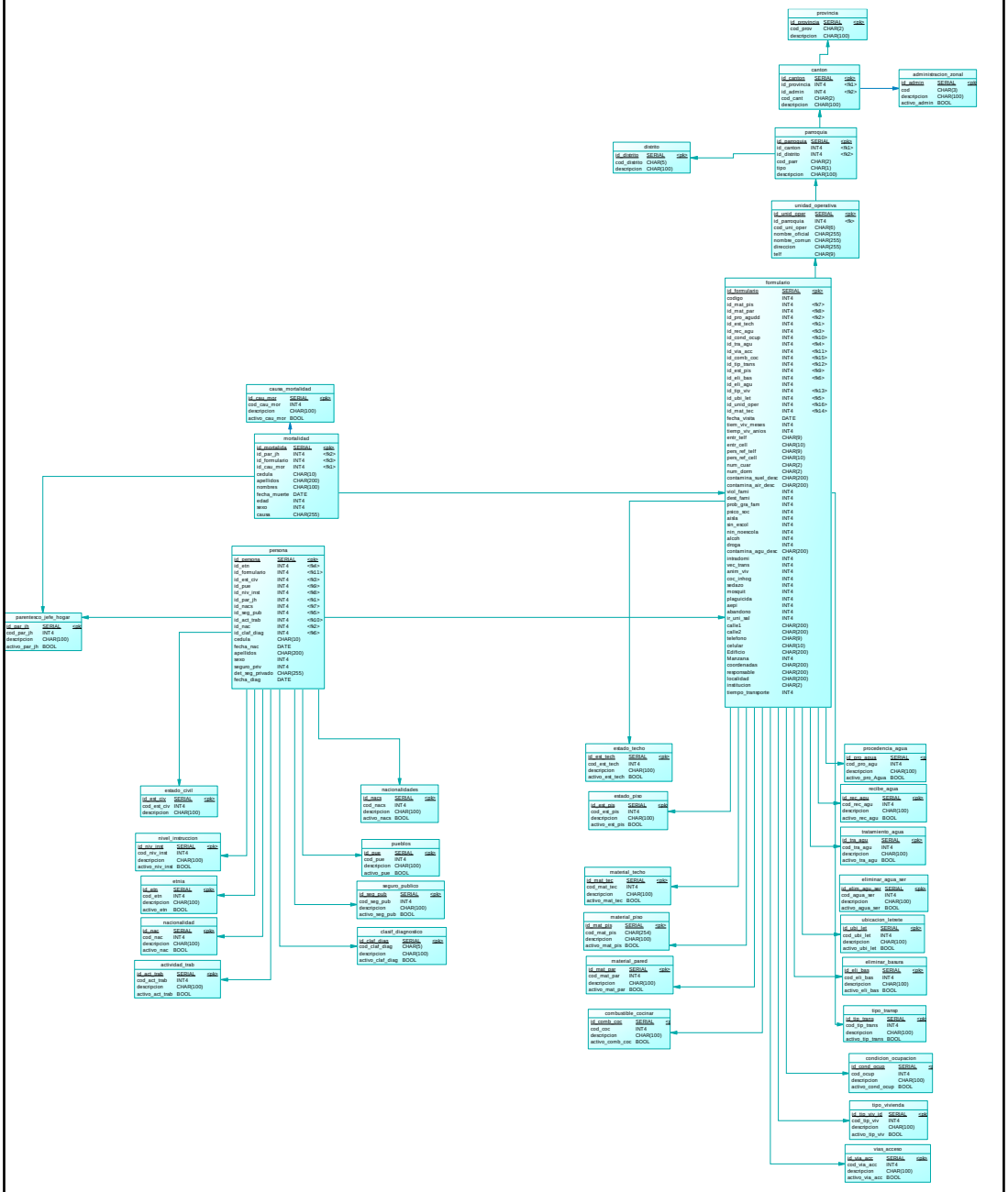


Figura 33: Diagrama Físico de la base de datos

Elaborado por: Henry Avila, Natalia Cueva

3.2.5 Diagrama de funcionamiento de la aplicación

Diagrama de funcionamiento de la app

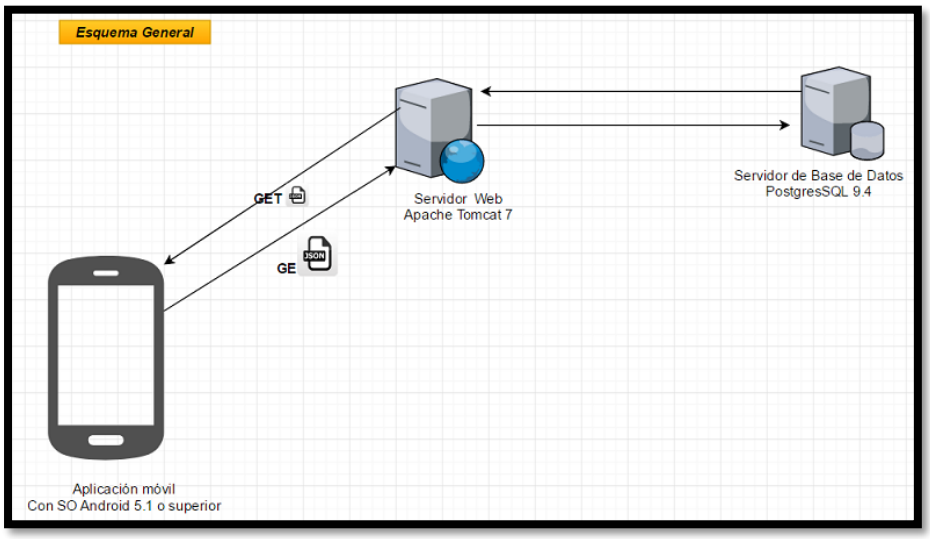


Figura 34: Diagrama interacción de componentes de la app
Elaborado por: Henry Avila, Natalia Cueva

3.3 Desarrollo

3.3.1 Interfaz de usuario de la aplicación móvil

La interfaz de usuario tiene una construcción según lo mencionado en el análisis de interfaz anteriormente, se mantiene el formato de la ficha familiar física del Ministerio de Salud.

El cual cuenta con un menú principal para llevar a cabo las distintas opciones tales como: ingresar nuevo formulario, descargar formularios, exportar formularios, buscar formularios, configurar servidor y actualizar los catálogos.

Las pantallas aquí mostradas son del ingreso de una nueva ficha familiar debido que la app fue concebida con el fin de registrar el levantamiento de la información de la ficha familiar del MSP, para conocer el u el uso y manejo de la aplicación se recomienda revisar el manual de usuario (véase Anexo 3).

Pantalla menú de la aplicación móvil

Figura 35: Pantalla menú de la aplicación móvil
Elaborado por: Henry Avila, Natalia Cueva

Las pantallas de la Figura 36, permiten que el usuario registre la información de los bloques 1 al 5 de la ficha familiar, además de contener los botones para el registro de la información de un nuevo miembro del hogar y personas fallecidas.

Como se observa en las imágenes el usuario realiza el ingreso de la información a través de listas desplegables evitando en lo posible el registro de la misma de manera manual, evitando así inconsistencia, inentendibilidad de la información.

El formato que se mantiene es similar a la ficha familiar física proporcionada por el Ministerio de Salud Pública del Ecuador.

Pantalla ingresar nueva ficha familiar de la aplicación móvil

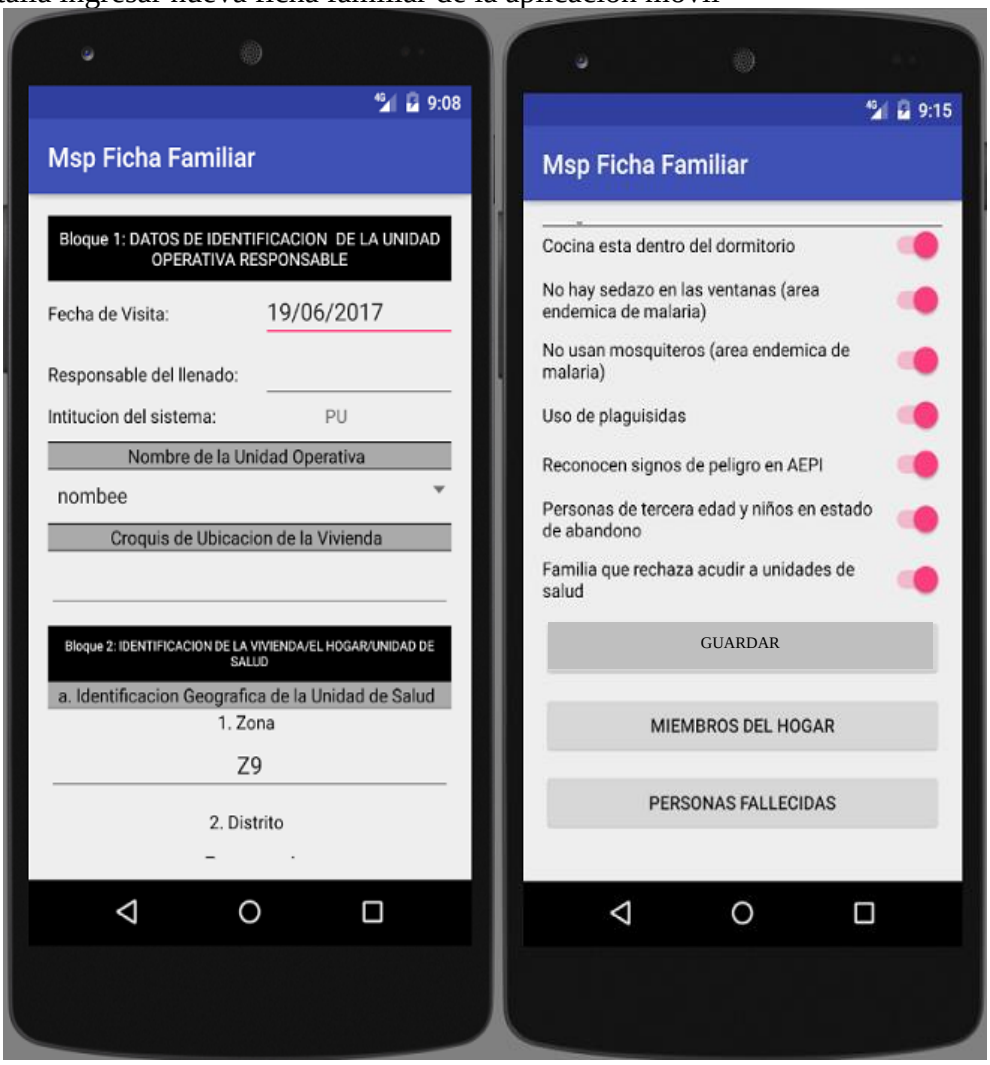


Figura 36: Pantalla ingresar nueva ficha familiar de la aplicación móvil
Elaborado por: Henry Avila, Natalia Cueva

En la Figura 37 se observa las pantallas para el levantamiento de la información de un nuevo miembro del hogar y personas fallecidas dentro del núcleo familiar.

De manera similar a las pantallas anteriores se mantiene listas desplegables para el registro de la información evitando en lo posible el ingreso manual.

Se mantiene el formato del bloque 6-7 de la ficha familiar del Ministerio de Salud Pública del Ecuador, para mantener la familiaridad con los encargados del levantamiento de la información.

Pantallas ingresar nuevo miembro del hogar y personas fallecidas de la app

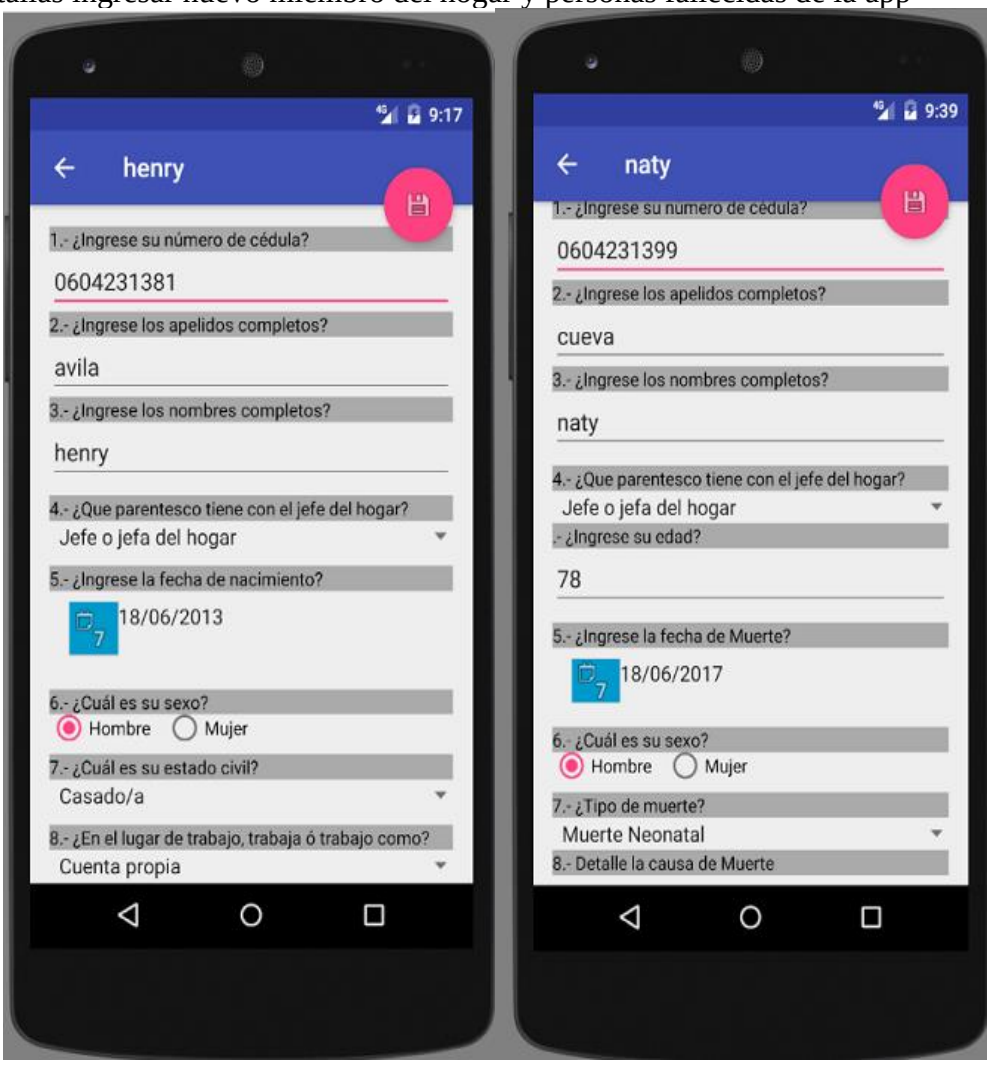


Figura 37: Pantalla ingresar nuevo miembro de hogar y persona fallecida de la ficha familiar de la aplicación móvil
Elaborado por: Henry Avila, Natalia Cueva

3.3.2 Creación de la base de datos SQLite

La creación dela base de datos del dispositivo móvil se realiza implementando la librería SQLiteOpenHelper en el desarrollo de la aplicación móvil**,** debido que permite la manipulación los datos de la base de datos esta implementación se realiza en la aplicación en la clase DatabaseHandler**.**

En esta parte del código se indica el nombre de la base de datos a usar y su versión, así como el constructor de la clase que contiene el contexto a usar, nombre de la base y la versión.

Creación Base de datos SQLite

```
public class DatabaseHandler extends SQLiteOpenHelper {
    // All Static variables
    // Database Version
    private static final int DATABASE_VERSION = 14;
    private static final String DATABASE_NAME = "ficha_familiar_msp";
    private Context context;

    public DatabaseHandler(Context context) {
        super(context, DATABASE_NAME, null, DATABASE_VERSION);
    }
```

Figura 38: Creación Base de datos SQLite
Elaborado por: Henry Avila, Natalia Cueva

Otro de los métodos para la creación es OnCreate que permite la creación de la base en el caso de no existir. Esta realiza un llamado a la clase para ejecutar la creación de las tablas dentro de la base de datos, extraer el código SQL para la creación de las mismas.

La clase Tables se explicará más adelante contiene os atributos mapeados de cada una de las tablas de la base de datos, esto se explicará más adelante.

Creación Base de datos SQLite

```
@Override
public void onCreate(SQLiteDatabase db) {

    System.out.println("********CREATE TABLES DATABASE*************");
    System.out.println("****************************************");
    /*ENABLE FOREGIN KEY*/
    db.execSQL("PRAGMA foreign_keys = ON");//ON - OFF
    for (TABLES tables : TABLES.values()) {
        System.out.println(tables.getQuery());
        db.execSQL(tables.getQuery());

    }
}
```

Figura 39: Método onCreate
Elaborado por: Henry Avila, Natalia Cueva

Otro método por defecto es onUpgrade que lleva acabo la actualización de la base de datos en caso de existir, eliminando las tablas y llamando al método onCreate para la actualización de la base de datos.

Creación Base de datos SQLite

```
@Override
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {

    System.out.println("********UPDATE DATA BASE*************");
    System.out.println("*********************************************");
    for (TABLES tables : TABLES.values()) {
        System.out.println("DROP TABLE IF EXISTS " + tables.getTablaName());
        db.execSQL("DROP TABLE IF EXISTS " + tables.getTablaName());
    }
    onCreate(db);
}
```

Figura 40: Método onUpgrade
Elaborado por: Henry Avila, Natalia Cueva

La base de datos se encuentra en la ruta /data/data/database/z9.msp.gob/ficha_familiar_msp. Esta ruta no es accesible por el usuario del dispositivo móvil debido a la seguridad que proporciona el Sistema Operativo Android sobre una carpeta del sistema.

La clase TABLES contiene la creación de las tablas de la base de datos en el dispositivo móvil como se indica en el recuadro señalado con el número 1 se tiene el nombre de la tabla separado (2) por una coma (,) realiza la referencia a la clase de la tabla en el cual se encuentra el mapeo de la misma y por último (3) el código SQL para la creación de esta.

Código de creación de las tablas base SQLite

```
public enum TABLES {
    CONFIG_SERVER("config_server",null,"CREATE TABLE config_server ( id INTEGER PRIMARY KEY  NOT NULL , ip VARCHAR NOT NULL , puerto INTEGER NOT NULL,servicio_cat VARCHAR,serv
    PROVINCIA("provincia",Provincia.class,"CREATE TABLE provincia ( id INTEGER PRIMARY KEY  NOT NULL , cod_prov VARCHAR NOT NULL , descripcion VARCHAR NOT NULL )"),
    CANTON("canton",Canton.class,"CREATE TABLE canton ( id INTEGER PRIMARY KEY  NOT NULL ,id_provincia Integer NOT NULL  DEFAULT (null) ,cod_cant VARCHAR NOT NULL ,descripcion
    PARROQUIA("parroquia",Parroquia.class,"CREATE TABLE parroquia ( id INTEGER PRIMARY KEY  NOT NULL , id_canton INTEGER NOT NULL , id_distrito INTEGER NOT NULL , cod_parr VAR
    DISTRITO("distrito",Distrito.class,"CREATE TABLE distrito ( id INTEGER PRIMARY KEY  NOT NULL , cod_distrito VARCHAR NOT NULL , descripcion VARCHAR NOT NULL )"),
    ADMISNISTRACION_ZONAL("administracion_zonal",AdministracionZonal.class,"CREATE TABLE administracion_zonal( id INTEGER PRIMARY KEY  NOT NULL ,cod character VARCHAR NOT NULL
    UNIDAD_OPERATIVA("unidad_operativa",UnidadOperativa.class,"CREATE TABLE unidad_operativa ( id INTEGER PRIMARY KEY  AUTOINCREMENT  NOT NULL , id_parroquia INTEGER NOT NULL
    ELIMINAR_BASURA("eliminar_basura",EliminarBasura.class,"CREATE TABLE eliminar_basura ( id INTEGER PRIMARY KEY  NOT NULL  DEFAULT (null) ,cod_eli_bas INTEGER NOT NULL ,desc
    UBICACION_LETRETE("ubicacion_letrete",UbicacionLetrete.class,"CREATE TABLE ubicacion_letrete ( id INTEGER PRIMARY KEY  NOT NULL  DEFAULT (null) ,cod_ubi_let INTEGER NOT NU
    ELIMINAR_AGUA_SERVIDAS("eliminar_agua_ser",EliminarAguaSer.class,"CREATE TABLE eliminar_agua_ser ( id INTEGER PRIMARY KEY  NOT NULL , cod_agua_ser INTEGER NOT NULL , descr
    RECIBE_AGUA("recibe_agua",RecibeAgua.class,"CREATE TABLE recibe_agua ( id INTEGER PRIMARY KEY  NOT NULL  DEFAULT (null) ,cod_rec_agu INTEGER NOT NULL ,descripcion VARCHAR
    TRATAMIENTO_AGUA("tratamiento_agua",TratamientoAgua.class,"CREATE TABLE tratamiento_agua ( id INTEGER PRIMARY KEY  NOT NULL  DEFAULT (null) ,cod_tra_agu INTEGER NOT NULL
    PROCEDENCIA_AGUA("procedencia_agua",ProcedenciaAgua.class,"CREATE TABLE procedencia_agua ( id INTEGER PRIMARY KEY  NOT NULL  DEFAULT (null) ,cod_pro_agu INTEGER NOT NULL
```

Figura 41: Creación de tablas SQLite
Elaborado por: Henry Avila, Natalia Cueva

3.3.3 Creación de la base de datos PostgreSQL

La creación de la base de datos se realizó con el código ya conocido SQL y se realizó en el gestor de base de datos pgAdmin. A continuación, se indica como referencia la creación de las tablas principales formulario, personas y mortalidad, así también del catálogo pueblo.

Código SQL creación tabla formulario

```
CREATE TABLE formulario
(
  id_formulario serial NOT NULL,
  codigo integer,
  id_mat_pis integer,
  id_mat_par integer,
  id_pro_agudd integer,
  id_est_tech integer,
  id_rec_agu integer, -- AGUA QUE RECIBE EL HO
  id_cond_ocup integer,
  id_tra_agu integer,
  id_via_acc integer,
  id_comb_coc integer,
  id_tip_trans integer,
  id_est_pis integer,
  id_eli_bas integer,
  id_eli_agu integer,
  id_tip_viv integer,
  id_ubi_let integer,
  id_unid_oper integer,
  id_mat_tec integer,
  fecha_visita date NOT NULL,
  tiem_viv_meses integer,
  tiemp_viv_anios integer,
  entr_telf character(9),
  entr_cell character(10),
  pers_ref_telf character(9),
  pers_ref_cell character(10),
  num_cuar character(2),
  num_dorm character(2),
  contamina_suel_desc character varying(200),
  contamina_air_desc character varying(200),
  viol_fami integer,
  dest_fami integer,
```

Figura 42: Código SQL formulario
Elaborado por: Henry Avila, Natalia Cueva

Código SQL creación tabla formulario

```
CREATE TABLE mortalidad
(
  id_mortalida serial NOT NULL,
  id_par_jh integer,
  id_formulario integer,
  id_cau_mor integer,
  cedula character(10),
  apellidos character(200),
  nombres character(100),
  fecha_muerte date,
  edad integer,
  sexo integer,
  causa character(255),
  CONSTRAINT pk_mortalidad PRIMARY KEY (id_mortalida),
  CONSTRAINT fk_mortalid_reference_causa_mo FOREIGN KEY (id_cau_mor)
      REFERENCES causa_mortalidad (id_cau_mor) MATCH SIMPLE
      ON UPDATE RESTRICT ON DELETE RESTRICT,
  CONSTRAINT fk_mortalid_relations_formular FOREIGN KEY (id_formulario)
      REFERENCES formulario (id_formulario) MATCH SIMPLE
      ON UPDATE RESTRICT ON DELETE RESTRICT,
  CONSTRAINT fk_mortalid_relations_parentes FOREIGN KEY (id_par_jh)
      REFERENCES parentesco_jefe_hogar (id_par_jh) MATCH SIMPLE
      ON UPDATE RESTRICT ON DELETE RESTRICT,
  CONSTRAINT ckc_sexo_mortalid CHECK (sexo IS NULL OR sexo >= 1 AND sexo <= 2)
)
```

Figura 43: Código SQL mortalidad
Elaborado por: Henry Avila, Natalia Cueva

Código SQL creación tabla formulario

```
CREATE TABLE persona
(
  id_persona serial NOT NULL,
  id_etn integer,
  id_formulario integer,
  id_est_civ integer,
  id_pue integer,
  id_niv_inst integer,
  id_par_jh integer,
  id_nacs integer,
  id_seg_pub integer,
  id_act_trab integer,
  id_nac integer,
  id_claf_diag integer,
  cedula character(10),
  fecha_nac date,
  apellidos character varying(200),
  sexo integer,
  seguro_priv integer,
  det_seg_privado character varying(255),
  fecha_diag date,
  nombres character varying(200),
  CONSTRAINT pk_persona PRIMARY KEY (id_persona),
  CONSTRAINT fk_persona_nac FOREIGN KEY (id_nac)
      REFERENCES nacionalidad (id_nac) MATCH SIMPLE
      ON UPDATE RESTRICT ON DELETE RESTRICT,
  CONSTRAINT fk_persona_nacs FOREIGN KEY (id_nacs)
      REFERENCES nacionalidades (id_nacs) MATCH SIMPLE
      ON UPDATE RESTRICT ON DELETE RESTRICT,
  CONSTRAINT fk_persona_relations_activida FOREIGN KEY (id_act_trab)
      REFERENCES actividad_trab (id_act_trab) MATCH SIMPLE
      ON UPDATE RESTRICT ON DELETE RESTRICT,
  CONSTRAINT fk_persona_relations_clasif_d FOREIGN KEY (id_claf_diag)
      REFERENCES clasif_diagnostico (id_claf_diag) MATCH SIMPLE
      ON UPDATE RESTRICT ON DELETE RESTRICT,
```

Figura 44: Código SQL persona
Elaborado por: Henry Avila, Natalia Cueva

Código SQL creación tabla formulario

```
CREATE TABLE pueblos
(
  id_pue serial NOT NULL,
  cod_pue integer NOT NULL,
  descripcion character(100) NOT NULL,
  activo_pue boolean,
  CONSTRAINT pk_pueblos PRIMARY KEY (id_pue)
)
```

Figura 45: Código SQL pueblos
Elaborado por: Henry Avila, Natalia Cueva

3.3.4 Creación Core de integración

Se toma como referencia el objeto "Pueblos", para explicar cómo está desarrollado el Core de integración. El objeto pueblos, con sus atributos (tabla base de datos) debe ser mapeado a su correspondiente clase.

Tabla pueblos de la base de datos

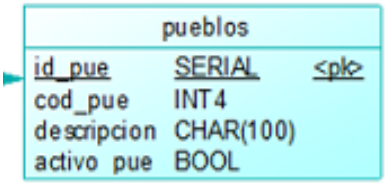


Figura 46: Tabla pueblos
Elaborado por: Henry Avila, Natalia Cueva

El termino mapeo hace referencia a la descripción de los campos de cada una de las tablas de la base de datos, conteniendo el nombre del campo, el tipo de dato, estableciendo que es un campo tipo columna y si es o no una clave primaria.

Se lleva a cabo este proceso para mantener la similitud en los campos de cada una de las tablas las bases de datos PostgreSQL y SQLite y permitir la transferencia de los datos sea del dispositivo móvil al servidor contendor de la base de datos PostgreSQL o del servidor a la base de datos SQLite del dispositivo móvil.

Mapeo de la Tabla pueblos

Pueblo.java

```
1  package entity;
2
3  import java.io.Serializable;
5
6
7  /**
8   * The persistent class for the pueblos database table.
9   *
10  */
11 @Entity
12 @Table(name="pueblos")
13 @NamedQuery(name="Pueblo.findAll", query="SELECT p FROM Pueblo p WHERE p.activoPue='TRUE'")
14 public class Pueblo implements Serializable {
15     private static final long serialVersionUID = 1L;
16
17     @Id
18     @Column(name="id_pue")
19     private Integer id_pue;
20
21     @Column(name="cod_pue")
22     private Integer cod_pue;
23
24     private String descripcion;
25
26     @Column(name="activo_pue")
27     private Boolean activoPue;
28
29     public Pueblo() {
30     }
31
```

Figura 47: Mapeo tabla pueblos
Elaborado por: Henry Avila, Natalia Cueva

En la clase Pueblos tenemos atributos como @Entity (identifica la clase que va a utilizar la persistencia), @Table (indica a JPA contra que tabla de la base debe mapear la entidad), @Id (define la clave primary), @Column (nombra una columna), todos los atributos de la clase deben tener sus correspondientes Get y set. Este procedimiento se realiza con las tablas restantes de la base de datos.

Dentro del fichero Persitence.xml se define la conexión con la base de datos del servidor, los drivers, usuario y contraseña a utilizar, además de ubicar la ruta de las entidades previamente mapeadas como ejemplo continuamos con la ruta de la clase pueblo.

Código de conexión a la base de datos PostgreSQL

```
<property name="javax.persistence.jdbc.url" value="jdbc:postgresql://localhost:5432/msp"/>
<property name="javax.persistence.jdbc.user" value="postgres"/>
<property name="javax.persistence.jdbc.password" value="1234"/>
<property name="javax.persistence.jdbc.driver" value="org.postgresql.Driver"/>
roperties>
```

Figura 48: Código de conexión a la base de datos con JPA
Elaborado por: Henry Avila, Natalia Cueva

Ruta de acceso clase pueblo

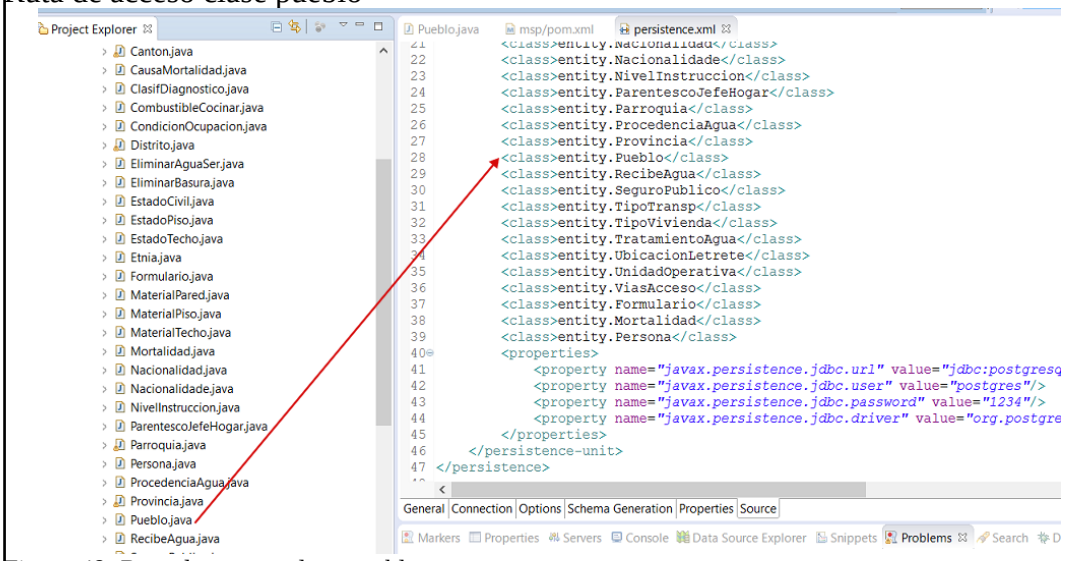


Figura 49: Ruta de acceso clase pueblo
Elaborado por: Henry Avila, Natalia Cueva

Para la creación del servicio web de catálogos se utilizó una enumeración, para discriminar los objetos de negocio que serán expuestos por el servicio web.

Ruta de acceso clase pueblo

```
public enum TABLES {
        ELIMINAR_BASURA(EliminarBasura.class),
        UBICACION_LETRETE(UbicacionLetrete.class),
        ELIMINAR_AGUA_SERVIDAS(EliminarAguaSer.class),
        RECIBE_AGUA(RecibeAgua.class),
        TRATAMIENTO_AGUA(TratamientoAgua.class),
        PROCEDENCIA_AGUA(ProcedenciaAgua.class),
        COMBUSTIBLE_COCINA(CombustibleCocinar.class),
        ESTADO_PISO(EstadoPiso.class),
        ESTADO_TECHO(EstadoTecho.class),
        MATERIAL_PARED(MaterialPared.class),
        MATERIAL_PISO(MaterialPiso.class),
        MATERIAL_TECHO(MaterialTecho.class),
        TIPO_TRANSPORTE(TipoTransp.class),
        VIAS_ACCESO(ViasAcceso.class),
        TIPO_VIVIENDA(TipoVivienda.class),
        CONDICION_OCUPACION(CondicionOcupacion.class),
        NACIONALIDAD(Nacionalidad.class),
        NACIONALIDADES(Nacionalidade.class),
        ETNIA(Etnia.class),
        SEGPUBLICO(SeguroPublico.class),
        PARENT_JE_HO(ParentescoJefeHogar.class),
        ESTADO_CIVIL(EstadoCivil.class),
        NIVEL_INSTRS(NivelInstruccion.class),
        ACTIVIADAD_TRABAJO(ActividadTrab.class),
        PUEBLOS(Pueblo.class),
        CAUSA_MORTALIDAD(CausaMortalidad.class),
        PROVINCIA(Provincia.class),
        CANTON(Canton.class),
        ADMIN_ZONAL(AdministracionZonal.class),
        DISTRITO(Distrito.class),
        PARROQUIA(Parroquia.class),
        UNIDAD_OPERATIVA(UnidadOperativa.class),
```

Figura 50: Código creación enum
Elaborado por: Henry Avila, Natalia Cueva

En la clase catalog existe el método que itera la enumeración TABLES, que contiene todos los objetos Java que corresponde a catálogos a ser expuestos por nuestro servicio web.

Al revisar la clase Catalog, nos encontramos con la etiqueta @NamedQuery, la cual busca todos los elementos de la tabla pueblos, donde la columna activoPue se encuentra activo (True).

Código clase Catalog

```
@Entity
@Table(name="pueblos")
@NamedQuery(name="Pueblo.findAll", query="SELECT p FROM Pueblo p WHERE p.activoPue='TRUE'")
public class Pueblo implements Serializable {
    private static final long serialVersionUID = 1L;

    @Id
    @Column(name="id_pue")
    private Integer id_pue;

    @Column(name="cod_pue")
    private Integer cod_pue;

    private String descripcion;

    @Column(name="activo_pue")
    private Boolean activoPue;

    public Pueblo() {
    }
```

Figura 51: Código clase Catalog
Elaborado por: Henry Avila, Natalia Cueva

Para retornar todos los elementos en un solo objeto usamos JsonObject donde por cada lista de resultado se agregará a este objeto, y el resultante se convierte a String y será el objeto resultado Json.

Creación de objeto JSon

```
43
44 @Path("/integracion")
45 public class Catalog {
46 private static final String UNIT_PERSISTENCE="msp";
47@    @GET
48    @Path("/catalogos")
49    @Produces(MediaType.APPLICATION_JSON + "; charset=UTF-8")
50    public Response catalog() {
51        EntityManagerFactory emf = Persistence.createEntityManagerFactory(UNIT_PERSISTENCE);
52        EntityManager em = emf.createEntityManager();
53        final Gson gson =new Gson();
54        JsonObject jo = new JsonObject();
55        for (TABLES mTable : TABLES.values()) {
56            System.out.println(mTable);
57            String entity = mTable.getJPQLClass().getName().replace(".", "-");
58            String className[] = entity.split("-");
59            List<?> lista = em.createNamedQuery(className[1] + ".findAll", mTable.getJPQLClass()).getResultList();
60            if (lista != null && lista.size() > 0) {
61                JsonElement je = gson.toJsonTree(lista);
62                Table annotations = (Table) mTable.getJPQLClass().getAnnotation(Table.class);
63                if(annotations!=null){
64                jo.add(annotations.name(), je);
65                }else{
66                    jo.add(className[1].toLowerCase(), je);
67                }
68            }
69        }
70
71        em.close();
72        String representacionJSON = jo.toString();
73        return Response.status(200).entity(representacionJSON).build();
74    }
```

Figura 52: Creación objeto JSon
Elaborado por: Henry Avila, Natalia Cueva

Ahora probaremos el servicio web de catálogos, para lo cual ejecutamos el proyecto, y abrimos el navegador. Como observamos en la imagen inferior el resultado es un formato Json.

Resultado JSon Catálogos

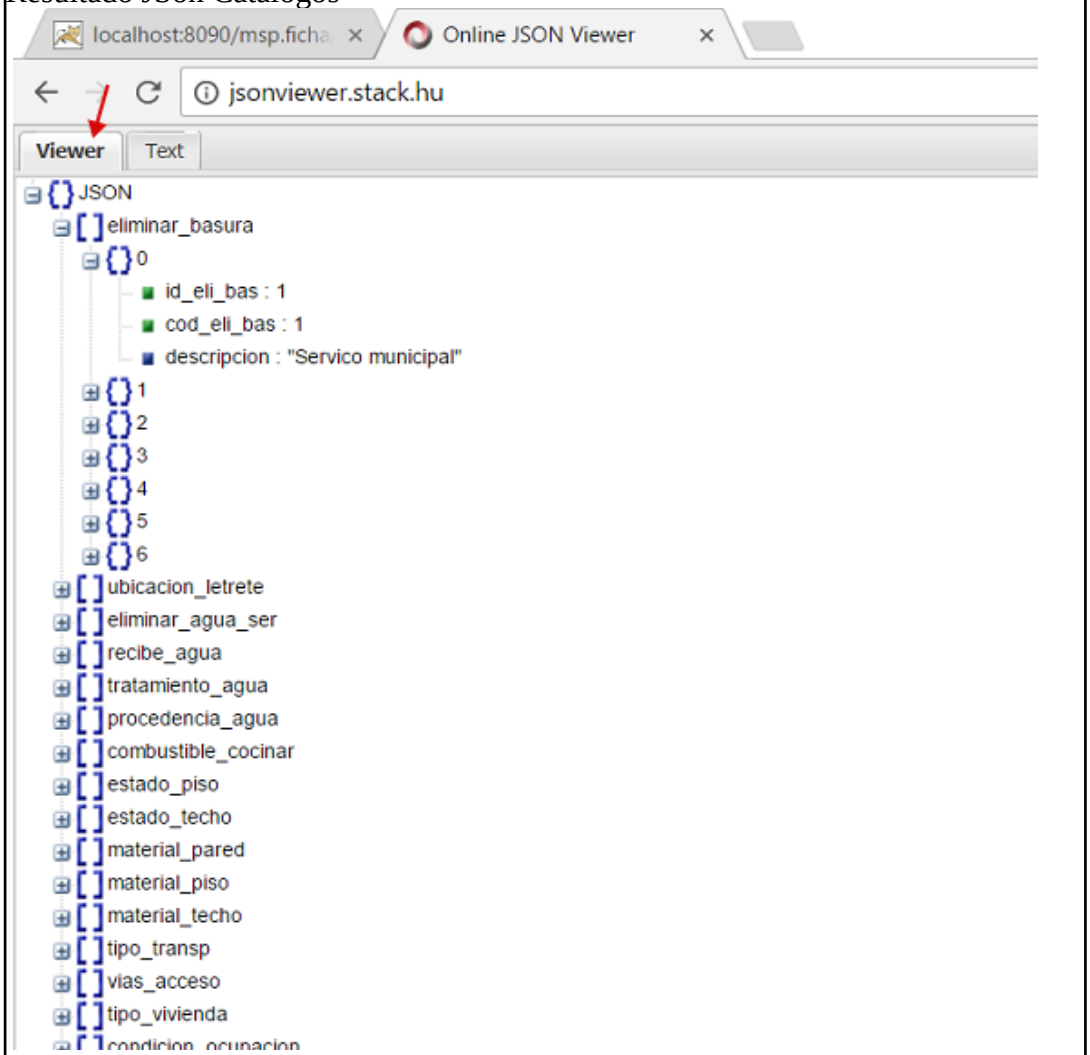


Figura 53: JSon catálogos
Elaborado por: Henry Avila, Natalia Cueva

3.3.5 Conexión de la aplicación con el servidor

En el proyecto de la aplicación móvil se maneja la clase MasterPageActivity que internamente tiene una clase llamada GetCommentTask la cual hereda de AsyncTask que nos permite ejecutar tareas en background. Cuando una clase hereda de AsyncTask deberá sobrescribir los métodos doInBrackground y onPostExecute. En el método doInBackGround, usamos la clase HttpURLConnection, perteneciente al API JAVA NET que posee clases para implementaciones de aplicaciones en red.

Código de conexión servidor

```
try {

    // Establecer la conexión
    con = (HttpURLConnection)url.openConnection();
    con.setConnectTimeout(5000);
    // Obtener el estado del recursocon.responseCode
    int statusCode = con.getResponseCode();

    if(statusCode!=200) {
        comments.add(S.responseServer+statusCode);

    }
```

Figura 54: Código de conexión con el servidor
Elaborado por: Henry Avila, Natalia Cueva

Si la respuesta del servidor es exitosa, construimos un StringBuilder con el resultado del servidor, el cual es un objeto Json. Después de terminar la lectura en del servidor en el objeto Out, vaciamos los datos de la base de datos SQLite, y el objeto Out lo convertimos a un objeto Json, con la ayuda de la librería Gson realizamos la conversión de un String a un objeto java, y lo procedemos a insertar en la base de datos del dispositivo.

La implementación de los servicios Web exportar formularios y descargar formularios es de manera similar a la implementación del servicio web de catálogos. La diferencia radica que para llevar a cabo la descarga de formularios se requiere el id de los mismos.

JSon descarga de formularios

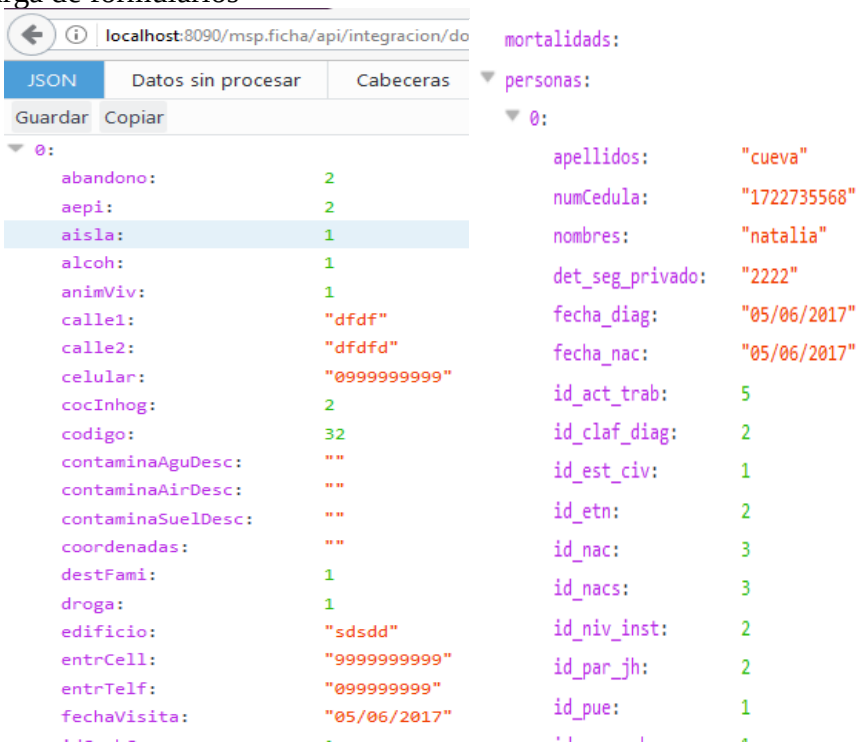


Figura 55: Código de inserción base de datos SQLite
Elaborado por: Henry Avila, Natalia Cueva

Código de inserción en la base de datos del dispositivo

```
JSONObject response = new JSONObject(out.toString());
Iterator<?> keys = response.keys();
db.deleteData();
while( keys.hasNext() ) {
    String tableName = (String)keys.next();
    try {
        JSONArray listObject=response.getJSONArray(tableName);
        for(int i=0;i<listObject.length();i++){
            TABLES table=TABLES.findByTableName(tableName);
            if(table!=null) {
                Gson gson = new Gson();
                Object entity = gson.fromJson(listObject.getString(i), table.getaClass());
                db.insert(table, entity);
            }

        }
    } catch (JSONException e) {
        e.printStackTrace();
    }
```

Figura 56: Código de inserción base de datos SQLite
Elaborado por: Henry Avila, Natalia Cueva

3.3.6 Codificación para la manipulación de componentes en la aplicación móvil

Para la manipulación de los componentes se utiliza findViewById que encuentra el objeto identificado por el nombre R.id.nombrecomponente para posteriormente obtener o agregar la información en él.

En esta selección de código se busca el componente etxtfecha tipo textview, se asigna un formato a la fecha del sistema y se agrega esta información al componente.

Código para manipulación de componentes

```
fecha=new Date();
fecha_completa= (TextView) findViewById(R.id.etxtfecha);

SimpleDateFormat fecc=new SimpleDateFormat(Utilitarios.formateDate_ddMMyyyy);
fechacComplString = fecc.format(fecha);
fecha_completa.setText(fechacComplString);
```

Figura 57: Código manipulación de componentes en Android Studio
Elaborado por: Henry Avila, Natalia Cueva

Para la modificación de las listas desplegables se utiliza el siguiente código

Código para manipulación de listas desplegables

```
spinnerTipoVivienda = (Spinner)findViewById(R.id.spinnerTipoVivienda);
adapterTipoVivienda= new SimpleCursorAdapter(this,
        android.R.layout.simple_spinner_item,
        db.getAllGeneric("tipo_vivienda"),
        new String[]{"descripcion"},
        new int[]{android.R.id.text1},
        SimpleCursorAdapter.FLAG_REGISTER_CONTENT_OBSERVER);
spinnerTipoVivienda.setAdapter(adapterTipoVivienda);
```

```
spinnerTipoVivienda = (Spinner)findViewById(R.id.spinnerTipoVivienda);
adapterTipoVivienda= new SimpleCursorAdapter(this,
        android.R.layout.simple_spinner_item,
        db.getAllGeneric("tipo_vivienda"),
        new String[]{"descripcion"},
        new int[]{android.R.id.text1},
        SimpleCursorAdapter.FLAG_REGISTER_CONTENT_OBSERVER);
spinnerTipoVivienda.setAdapter(adapterTipoVivienda);
spinnerTipoVivienda.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener(){
    @Override
    public void onItemSelected(AdapterView<?> parent, View view, int position, long id) {
        int idSpinner = parent.getId();
        switch(idSpinner) {
            case R.id.spinnerTipoVivienda:
                Cursor c = (Cursor) parent.getItemAtPosition(position);
                tipo_vivienda= c.getString(
                        c.getColumnIndex(db.getAllGeneric("tipo_vivienda").getColumnName(0)));
                break;
        }
    }
```

Figura 58: Código manipulación de listas desplegables en Android Studio
Elaborado por: Henry Avila, Natalia Cueva

En la imagen marcado con el número 1 se define el tipo de objeto Spinner y se lo ubica por el id asignado en el activity contenedor del componente, posterior a esto marcado con el número 2 se agrega un layout simple que contendrá la información de la lista añadiendo todos los registros de la tabla indicada de la base de datos señalado, en el nuero 3 de la imagen se indica el campo a visualizar en la lista desplegable. Y con el número 4 se añade esta información al componente. Posterior

a esto para obtener la información del Spinner se utiliza el oyente setOnItemSelectedListener para obtener el identificador asociado al elemento seleccionado en la tabla de la base de datos para su almacenamiento posterior.

Capítulo 4

Pruebas y resultado

4.1 Resultados

En las siguientes imágenes se observa el resultado final de la aplicación móvil para el levantamiento de información del inventario de la ficha familiar del MSP.

Pantalla menú principal App

Figura 59: Resultado pantalla principal app
Elaborado por: Henry Avila, Natalia Cueva

Pantalla ingresar nueva ficha familiar de la app

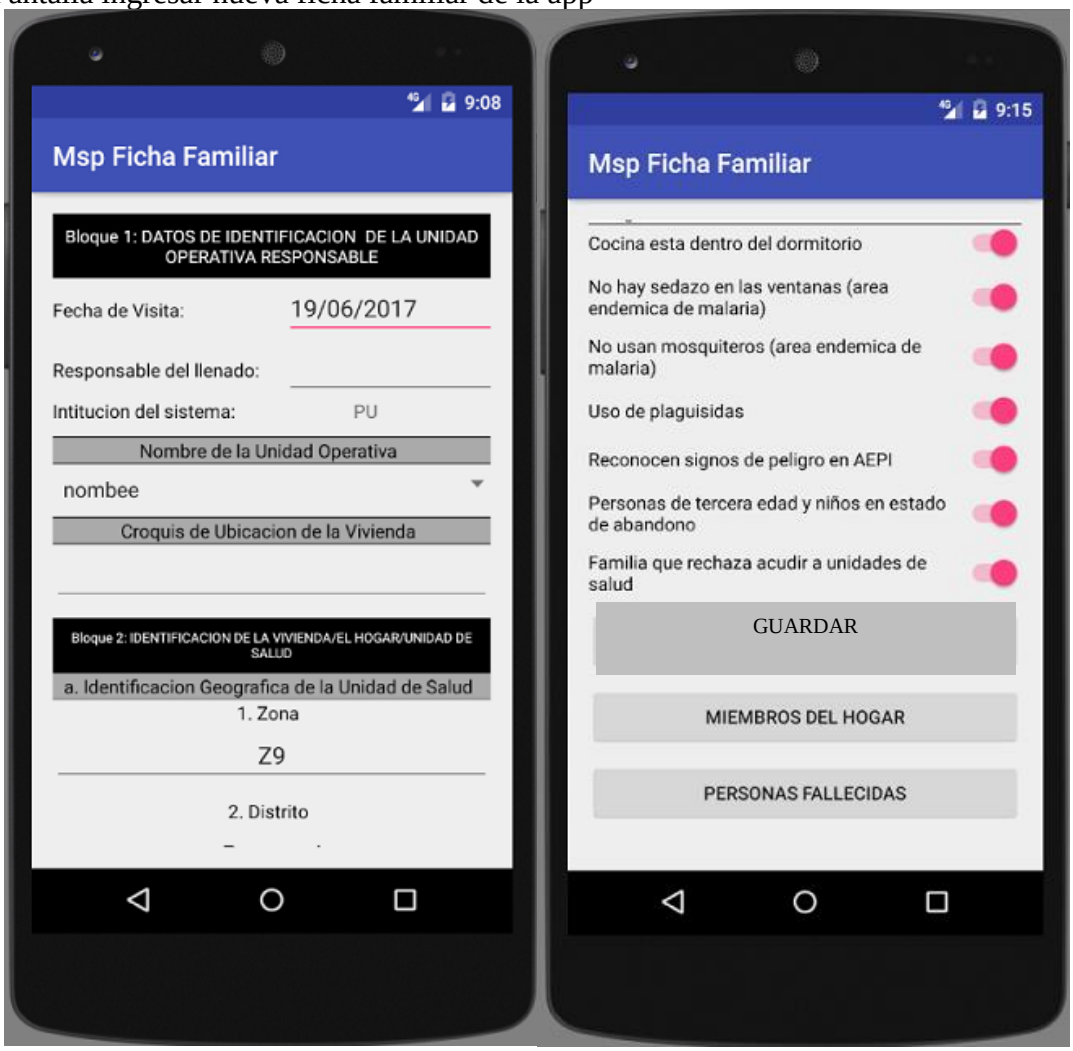


Figura 60: Pantalla ingresar nueva ficha familiar de la aplicación móvil
Elaborado por: Henry Avila, Natalia Cueva

Pantallas ingresar nuevo miembro del hogar y personas fallecidas app

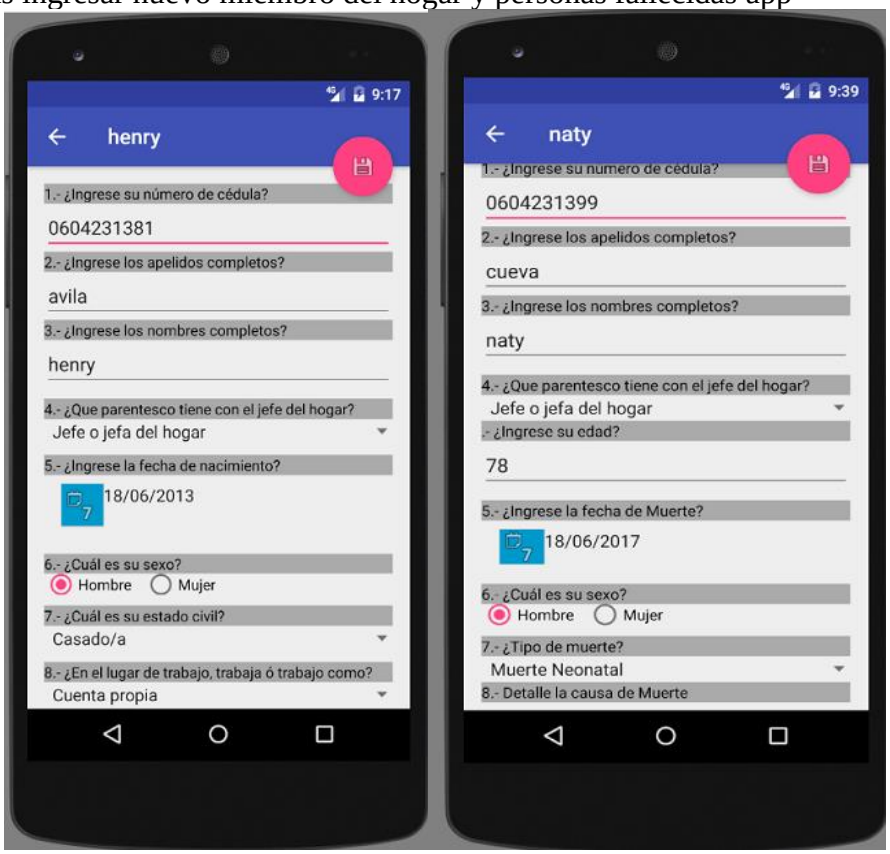


Figura 61: Pantalla ingresar nuevo miembro de hogar y persona fallecida de la ficha familiar de la aplicación móvil
Elaborado por: Henry Avila, Natalia Cueva

Pantalla configuración del servidor app

Figura 62: Pantalla ingreso de información configuración del servidor
Elaborado por: Henry Avila, Natalia Cueva

Pantalla descargar formulario app

Figura 63: Pantalla descarga de formularios
Elaborado por: Henry Avila, Natalia Cueva

Pantalla exportar formularios app

Figura 64: Pantalla exportar formularios
Elaborado por: Henry Avila, Natalia Cueva

Pantalla actualizar catálogos app

Figura 65: Pantalla actualizar catálogos
Elaborado por: Henry Avila, Natalia Cueva

4.2 Pruebas

Para probar el funcionamiento adecuado de la aplicación móvil se aplicó pruebas de caja blanca, caja negra y pruebas de rendimiento obteniendo los resultados descritos a continuación.

Las pruebas de caja blanca o también denominadas pruebas estructurales ya que permiten llevar a cabo pruebas sobre el código de la aplicación, es decir, probar las instrucciones, bloques, etc. que se ejecutan en el código fuente. Se utiliza la herramienta JUnit para la prueba de cada uno de los métodos de la aplicación móvil.

Configuración para el desarrollo de la prueba JUnit

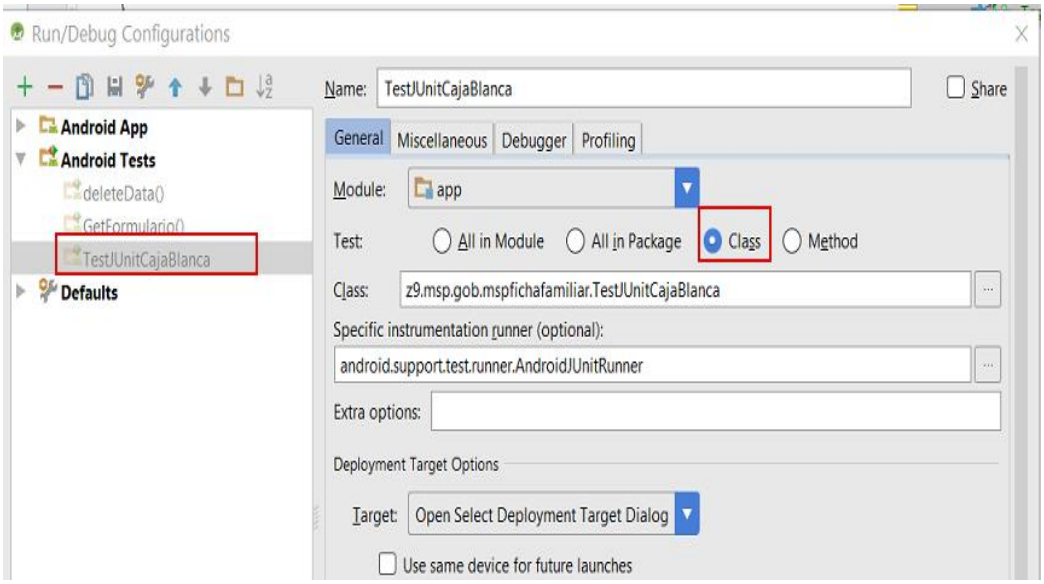


Figura 66: Pruebas de caja blanca con JUnit
Elaborado por: Henry Avila, Natalia Cueva

Pruebas de caja blanca con JUnit

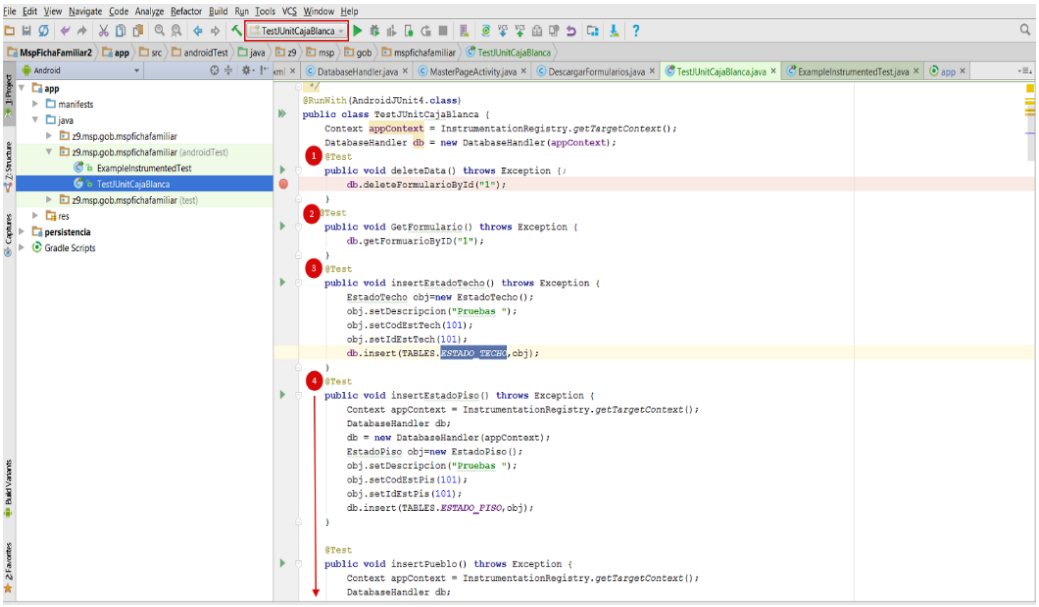


Figura 67: Pruebas de caja blanca con JUnit
Elaborado por: Henry Avila, Natalia Cueva

Resultado de pruebas de Caja Blanca

Figura 68: Resultado de pruebas de caja blanca con JUnit
Elaborado por: Henry Avila, Natalia Cueva

Resultado de pruebas de Caja Blanca

Figura 69: Resultado de pruebas de caja blanca con JUnit para cada método de la app
Elaborado por: Henry Avila, Natalia Cueva

Las pruebas a continuación realizadas son las llamadas de caja negra también conocidas como pruebas de funcionalidad ya que permiten llevar a cabo las pruebas de interacción con el sistema por parte del usuario para cumplir con los requerimientos sin importar la programación que se encuentra por detrás. Los resultados se reflejan en la siguiente tabla, estas pruebas fueron realizadas por personal de la administración zonal que serán los encargados del levantamiento de la información del formulario de inventario de la ficha familiar.

Tabla 18: Prueba de caja negra de la aplicación móvil

Requerimiento	Resultado	Observación
Ingreso de datos de acuerdo con la lógica de la ficha familiar física	OK	El ingreso de una nueva ficha conserva la misma lógica que la ficha familiar física.
El registro de la fecha de la ficha familiar es automático al seleccionar el ingreso de un nuevo formulario	OK	El registro de la fecha es automático y el que registra el dispositivo
Se selecciona la lista desplegable de unidad operativa y los datos de zona, distrito, provincia cantón, parroquia se cargan automáticamente en cada uno de los campos.	OK	Carga automáticamente al seleccionar una unidad operativa.
Los campos de ingreso obligatorio se encuentran debidamente identificados	OK	Los campos de ingreso obligatorio al no ser registrados se pintan de rojo o emiten una alerta para su ingreso.
Al seleccionar en la lista desplegable de condición de ocupación de la vivienda una opción distinta a "Ocupada" se bloquea el ingreso de la información a continuación.	OK	Al seleccionar ocupada permite el ingreso de la información a continuación requerida, caso contrario se bloquea y no permite ingresar nueva información.
Los campos de verificación si/no permiten únicamente la selección de estas opciones	OK	La selección se realiza en un switch, con la opción de si cuando este se encuentra prendido y no cuando se encuentra apagado.
Validación de cedula en miembros del hogar y personas fallecidas	OK	Se realiza a través del método de validación por digito de la cedula
Actualizar información de las fichas familiares	OK	Se realiza a través de la opción buscar formulario, en donde se despliega una lista al seleccionar uno de estos en las pantallas se carga la información contenida en esta ficha.

Agregar varios miembros del hogar y personas fallecidas	OK	Se añadió la funcionalidad de agregar desde 0 persona a varios
Descargar formularios para el seguimiento	OK	Para la descarga de formularios se lo realiza con el id asignado en el servidor, y se lo puede realizar uno a uno o en un rango de formularios. Se verifica la conexión con el servidor y de no existir no se lleva a cabo la acción.
Exportar formularios	OK	Los formularios se descargan al seleccionar la opción exportar formulario y confirmar en el cuadro de diálogo. Se verifica la conexión con el servidor y de no existir no se lleva a cabo la acción. Al no existir consistencia en la información del dispositivo y el servidor no se realiza la exportación.
Actualizar información de las listas desplegables	OK	En el menú se selecciona la opción de actualizar catálogos y la información se carga de manera automática en cada una de las listas. Se verifica la conexión con el servidor y de no existir no se lleva a cabo la acción.
Conexión con el servidor	OK	Permite almacenar la configuración del servidor para la conexión posterior con el mismo. Si el dispositivo no se encuentra en la misma red del servidor no se lleva a cabo la conexión.

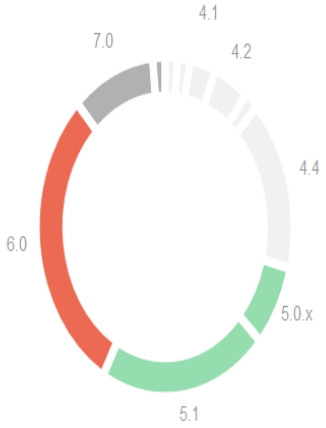
Elaborado por: Henry Avila, Natalia Cueva

Las pruebas de rendimiento permiten conocer cuál es el funcionamiento de la aplicación, como esta utiliza los recursos del dispositivo y otros. Se utiliza la herramienta Monkop para el monitoreo del apk, la aplicación se probó en 7 tipos de dispositivos con diferentes características para obtener los resultados más próximos.

Test de rendimiento aplicación Ficha Familiar MSP

Monkop Robo-Test

Android Market (% API level)



Executions

Ride	Device	OS	Size/Density	Execution	Startup
	Google Nexus 5	6.0 (API:23)	normal/xxhdpi	10m09s.655	.323
	Samsung Galaxy S 4 (SCH-R 970)	5.0.x (API:21)	normal/xxhdpi	10m04s.685	.961
	Google Nexus 9	6.0 (API:23)	xlarge/xhdpi	10m08s.996	01s.681
	Google Nexus 10	5.1 (API:22)	xlarge/xhdpi	10m06s.825	.592
	Moto G (XT1031)	5.1 (API:22)	normal/xhdpi	10m02s.555	.789
	Moto X 2014 (XT1096)	5.1 (API:22)	normal/xxhdpi	10m04s.801	.513
	Droid Turbo 2 (XT1585)	6.0 (API:23)	normal/xxxhdpi	10m04s.679	.877

Figura 70: Prueba en distintos dispositivos de la aplicación móvil Ficha Familiar MSP
Elaborado por: Henry Avila, Natalia Cueva

Para verificar su funcionamiento se obtiene el informe del dispositivo Samsung Galaxi S4, se detalla en la siguiente taba los resultados obtenidos.

Tabla 19: Prueba de rendimiento de la aplicación móvil

Requerimiento	**Resultado**	**Observación**
Nivel mínimo de API compatible	21	Uso a partir de sistemas operativos Android 5.0.x
Nivel máximo de API compatible	indefinido	Se puede utilizar en niveles superiores de sistema operativo Android 5.0.x
Tamaño de pantalla del dispositivo	Pequeña, mediana, grande, extra grande	Se visualiza correctamente en cualquier tamaño de pantalla
Uso de la CPU	12% - 33%	Se encuentra en porcentajes que no afectan el correcto funcionamiento del dispositivo móvil
Uso de la memoria RAM	2.5 MB	Uso de la memoria en las últimas 3 horas
Tiempo de inicio de la aplicación	2 segundos	Tiempo de inicio de la aplicación para cada vez que se requiera su uso
Velocidad para subir datos al servidor	613 KB	
Velocidad de descarga de	808KB	

datos del servidor		
Uso de la batería	Nivel bajo	
Uso de la memoria de almacenamiento interno	24 MB	
Actualización de catálogos	Tiempo 7ms	El tiempo de actualización de catálogos dura
Almacenamiento de la información del formulario en SQLite	Tiempo 7ms	Este tiempo es un estimado por cada formulario almacenado en la base de datos SQLite
Conexión WIFI en la red de trabajo	Si	Necesaria conexión del dispositivo en la red de trabajo para carga y descarga de datos.

Elaborado por: Henry Avila, Natalia Cueva

CONCLUSIONES

- El desarrollo de la aplicación móvil tuvo resultados favorables para el levantamiento de la información de la ficha familiar del Ministerio de Salud, debido que permitió pasar del proceso de ingreso de la información manual a automático reduciendo así tiempo de registro de la misma y optimizando los recursos disponibles.
- El uso del Web Service facilita la integración debido que su función es compartir datos, lógica de negocio y procesos, de esta forma la información ubicada en el servidor central puede ser transferida al dispositivo móvil y viceversa, además, el uso de Java Persistence API aumenta la velocidad de respuesta de la base de datos PostgreSQL ubicada en el servidor, lo que facilita la disponibilidad de la información para su posterior uso.
- La automatización del inventario de la ficha familiar del Ministerio de Salud Pública evita retrasos, incompatibilidad y pérdida de la información, facilitando su disponibilidad para su uso posterior para las diferentes áreas involucradas en el proceso de la toma de decisiones en beneficio de la comunidad.
- Se concluyó con éxito el desarrollo de la aplicación debido que se cumplió lo planificado con la institución en base a los requerimientos establecidos.

RECOMENDACIONES

- Usar el API de Google Maps debido que esta se puede ubicar dentro de la aplicación móvil llenando el campo croquis de forma automática, sin necesidad de recurrir a la aplicación externa Maps ME y llenar este campo de manera manual.
- Los dispositivos móviles deben contar con conexión a internet para sincronizar automáticamente la información requerida en los catálogos de la ficha familiar.
- Se sugiere agregar nuevas funcionalidades a la aplicación móvil así, por ejemplo: mensajes de alerta de las fichas familiares que se encuentran ingresadas un año atrás para el seguimiento anual, lista automática de los responsables del levantamiento de la información, etc.
- Como recomendación para la institución, se debe tener claro que la modificación de archivos de configuración se lo realizará únicamente con la autorización del administrador, un cambio inadecuado puede implicar la no disponibilidad del servicio.

TRABAJOS FUTUROS

Desarrollo de la aplicación web para generación de reportes a través del análisis de la información ingresada en la aplicación.

Desarrollo de una aplicación móvil para seguimiento de actividades asignadas a cada familia.

Desarrollo de una aplicación móvil para el levantamiento de la información de la ficha familiar del MSP para cualquier dispositivo móvil, es decir, independiente del sistema operativo.

GLOSARIO DE TÉRMINOS

Zona 9 Administración Zonal zona 9 Pichincha

MSP Ministerio de Salud Pública del Ecuador

S.O Android Sistema operativo para dispositivos móviles

App aplicación móvil

LISTA DE REFERENCIAS

Almenárez Mendoza, F. (14 de octubre de 2011). Universidad Carlos III de Madrid Departamento de Ingeniería Telemática. Recuperado el 30 de mayo de 2017, de Programación en otras plataformas móviles: http://ocw.uc3m.es/ingenieria-telematica/software-de-comunicaciones-1/UDs_JME.

Álvarez Caules , C. (17 de septiembre de 2013). Arquitectura Java. Recuperado el 05 de junio de 2017, de Ejemplo deJPA, Introducción (I): http://www.arquitecturajava.com/ejemplo-de-jpa/

Android Developers. (2016). Developers. Obtenido de Arquitectura de Android: https://developer.android.com

Aumaille, B. (2002). J2EE Desarrollo de Aplicaciones Web. Barcelona: Francesc Garcia.

AVG Latinoamérica. (16 de agosto de 2016). AVG-LA. Recuperado el 08 de junio de 2017, de 5 Tendencias sobre Aplicaciones y Smartphones que te sorprenderán: https://www.avg-la.com/2016/08/16/5-tendencias-sobre-aplicaciones-y-smartphones-que-te-sorprenderan/

Babatina Goyeneche, A. M., Briceño Novoa, P. D., & Cardoso Gómez, O. J. (2016). Universidad Distrital José de Caldas. Recuperado el 29 de julio de 2017, de Prototipo de una aplicación móvil para el manejo del historial clínico y control médico del paciente: http://repository.udistrital.edu.co/bitstream/11349/5406/1/BabativaGoyenecheAng%c3%a9licaMaria2017.pdf

Chanchí Golondrino, G. E., Arciniegas Herrera, J. L., & Campo Muñoz, W. Y. (2015). Construcción y evaluación de servicios interactivos en entornos de TVDi. Ingeniería, 21(1), 67. doi:23448693

EcuRed. (junio de 2017). EcuRed Conocimiento con todos y para todos. Recuperado el 08 de junio de 2017, de SQLite: https://www.ecured.cu/SQLite

Eficazz. (s.f.). Eficazz. Obtenido de Dispositvos móviles: http://soyeficazz.com/st2/

Enriquez, J. G., & Casas, S. I. (2013). Usabilidad en Aplicaciones Móviles. Revista de Informe Científicos y Técnicos de la Universidad Nacional de la Patagonia Austral, 5(2), 25-47. doi:1852-4516

Gálvez Rojas, S., & Ortega Díaz, L. (2014). Java a Tope: J2ME (Java 2 Micro Edition). Málaga: Universidad de Málaga. doi:8468847046

Guerra Salazar, J., & Lucas Granda, H. A. (25 de julio de 2013). Universidad privada de Tacna. Obtenido de Desarrollo de un sistema de información móvil, utilizando RUP, para el registro de pedidos en la empresa PALFARMA E.I.R.L: https://www.academia.edu/16774680/Plantesis-Metodolog%C3%ADa_RUP

Jiménez , C., & Castro , H. (2014). Solución móvil con J2ME para monitorear y supervisar variables del proceso de la planta industrial del Centro de Electricidad y Automatización Industrial del SENA Regional del Valle. Ingenium, VIII(19), 27-35. Obtenido de http://revistas.usc.edu.co/index.php/

Location World. (s.f.). GeoDynamic. Obtenido de LocationWorld: http://location-world.com/gbs/images/productos/geocoding/FichaGeoDynamic_2.pdf

Ministerio de Salud Pública del Ecuador. (2012). Ministerio de Salud Pública del Ecuador. Obtenido de Manual del Modelo de Atención Integral de Salud-MAIS: http://instituciones.msp.gob.ec

Ministerio de Salud Pública del Ecuador. (1 de marzo de 2016). Ministerio de Salud Pública del Ecuador. Obtenido de Coordinación Zonal 9 - Salud: http://instituciones.msp.gob.ec/cz9

Molina Martínez, R. (enero - junio de 2012). Las tecnologías de información y comunicación: su evolución y aplicación en los negocios internacionales. INCEPTUM, 8(12), 439-466. Recuperado el 29 de mayo de 2017, de Las Tecnologías de Información y Comunicación: Su evolución y aplicación en los negocios internacionales: http://inceptum.umich.mx

Obando Núñez, A. D., & Orrala Paredes, F. (mayo de 2013). Escuela Politécnica del Litoral. Obtenido de Evaluación de posibilidades de procesamiento de imágens den Real-Time para sistemas móviles: http://www.dspace.espol.edu.ec

Pavón Maestras, J. (2012). Universidad Complutense de Madrid. Obtenido de Java EE- Apache Tomcat: https://www.fdi.ucm.es/profesor/jpavon/web/42-tomcat.pdf

Postgresql Org. (s.f.). Postgresql. Recuperado el 30 de mayo de 2017, de Sobre PostgreSQL: http://www.postgresql.ors.es

Programación de Interfaces de Usuario. (s.f.). Recuperado el 05 de junio de 2017, de Maven: http://algo3.uqbar-project.org/temario/01-intro-UI/maven

Quishpe, S., Rivero, D., & Rivas, F. (2016). Diseño de un Sistema Web para Asignación de Becas de Integración e Interoperabilidad en Base a un Bus de Servicio. Politécnica, 37(2), 3. Recuperado el 01 de junio de 2017, de http://www.revistapolitecnica.epn.edu.ec/images/revista/

Revelo, J. (7 de agosto de 2014). Hermosa Programación. Recuperado el 01 de junio de 2017, de Aprendiendo sobre la Arquitectura de Android: http://www.hermosaprogramacion.com

Rodriguez del Valle, A. N. (2009). Universidad Nacional de La Plata. Obtenido de Metodologías de diseño usadas en ingeniería: http://postgrado.info.unlp.edu.ar/Carreras/Especializaciones/Tecnologia_Informatica_Aplicada_en_Educacion/Trabajos_Finales/Rodriguez_Ana.pdf

SQLite . (s.f.). SQLite. Obtenido de SQLite: https://www.sqlite.org/about.html

Tardáguila Moro, C. (enero de 2009). Universidad Oberta de Catalunya. Obtenido de Dispositivos móviles y Multimedia: http://s3.amazonaws.com/academia.edu.documents/36519489/dispositivos_moviles_y_multimedia.pdf?AWSAccessKeyId=AKIAIWOWYYGZ2Y53UL3A&Expires=1496365023&Signature=yozaAJqaL%2FbGOzF2zpt0I1Q8KoQ%3D&response-content-disposition=inline%3B%20filename%3DDispositivo

Uicab Aldana, O., Ucán Pech, J. P., & Aguilar Vera, R. A. (2016). Una Herramienta para el Análisis de la Colaboración Diseñada con UWE- A Collaboration Analysis Tool Designed with UWE. Revista Latinoamericana de Ingeniería de Software, 4(6), 235-242.

Universidad Carlos III de Madrid. (s.f.). Estudio de UWE (UML-based Web Engineering). Madrid, España. Obtenido de Madrid.