PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ FACULTAD DE

CIENCIAS E INGENIERÍA

DISEÑO E IMPLEMENTACIÓN DE UNA SOLUCIÓN DE TI PARA EL MONITOREO DE EQUIPAMIENTO MÉDICO EN UNA INSTITUCIÓN PÚBLICA

ASESOR

César Stuardo Lucho Romero

Lima, julio del 2020

Resumen

La digitalización está contribuyendo grandemente en el desarrollo de la sociedad al reducir la cantidad de trabajo rutinario, mejorar la logística general, mejorar la calidad de vida, aumento de la productividad de las industrias, entre otros muchos beneficios. Cada vez la penetración de las tecnologías va en aumento gracias a las diferentes formas que hay para su implementación y a las nuevas herramientas que permiten una integración cada vez más rápida.

El propósito de la presente tesis es evaluar las herramientas tecnológicas existentes en el mercado, los *frameworks* presentes para el desarrollo de aplicaciones y otros; para finalmente integrarlos y tener una solución específica a un problema presente en los centros de salud. En este caso, se brindará una propuesta para la gestión del equipo médico, digitalizando los datos, documentos, facturas, entre otros relevantes, y permitiendo su gestión a través de un aplicativo intuitivo y de fácil manejo. Para esto se diseñará e implementará una solución TI para la gestión del equipamiento médico en una entidad pública.

Dentro de lo que nos permitirá esta aplicación es generar los códigos para el etiquetado, la lectura de estos, acceso de datos y documentos alojados en bases de datos, automatización de recordatorios para el mantenimiento de los equipos médicos, registro de la ubicación para el seguimiento de los equipos médicos.

Dedicatoria

ÍNDICE

LISTA DE TABLAS

LISTA DE FIGURAS

Introducción

La gestión de equipamiento médico en una institución pública en el sector salud, es una tarea importante para garantizar el correcto funcionamiento de los equipos para cuando se los necesite. Tanto la programación para su mantenimiento como tener la documentación de los equipos. Sin embargo, actualmente los métodos tradicionales como el uso de archivos físicos, hojas de excel, no permiten una correcta gestión de estos.

Ante este problema, se presenta un aplicativo para generar códigos para el etiquetado de los equipos médicos, además, de automatizar los recordatorios para el mantenimiento que corresponde a cada equipo y gestionar el área en la que se ubican estos.

Por tanto, se pretende brindar una solución de bajo costo de implementación y mantenimiento para mejorar la gestión del equipamiento médico a través del uso e integración de diferentes tecnologías. En consecuencia, el personal que lo requiera

tendrá acceso rápido a los datos de los equipos para poder gestionarlos de forma eficiente.

Capítulo 1
Situación actual en la adopción de soluciones TI en los centros de salud

1.1 Problemáticas de la digitalización de los últimos años

La tecnología está avanzando de manera rápida permitiendo a la sociedad un desarrollo más acelerado en distintas áreas. Sin embargo, esto también genera una brecha en diferentes ámbitos (económicamente, calidad de servicio, etc.) entre las empresas que pueden tomar ventaja de las tecnologías de la información con respecto a otras que no. Es por ello que se ve necesario analizar cómo está la situación en el Perú respecto a un marco internacional para poder evaluar el nivel de competitividad y la brecha de desarrollo que puede estar marcando la calidad de vida en la población.

Entre las dificultades que presentó el país durante los últimos años para adoptar las tecnologías, según Ghio informó en el libro "Transformación Digital en el Perú" [1], asegura que la burocracia y la corrupción limitan los avances y pone como ejemplo que la SUNAT ha avanzado grandemente, pero otras instituciones llevan 100 años de atraso, resaltando el hecho de que sus procesos son muy lentos por la falta de automatización . A pesar del avance que se realizó en dichas políticas y la promoción de las tecnologías, hay aún gran trabajo por realizar en muchos de los sectores del país. Entre los problemas actuales está la falta de capacitación a las empresas, desinterés por cambiar sustancialmente su empresa porque, aparentemente, no hay una gran

ventaja en cuanto a la inversión a realizar. Específicamente, en las clínicas un problema que caracteriza a esta institución es la dificultad para modelar los datos necesarios para el diagnóstico de cualquier paciente, es por ello que, en ese sentido, se dificulta la adopción de soluciones relacionadas a una historia clínica digital. Por lo mencionado, que una vía alterna es mantener los documentos en papel ya que han venido funcionando (aunque no tan eficientemente) y agregar soluciones que puedan facilitar los procesos de gestión de las clínicas. En el CADE Digital realizado en Perú el 2019 [1], los emprendedores mencionaron las dificultades que poseían los negocios para empezar una transformación digital. Entre estas, la primera era un problema de creencias, presuponen que sus trabajadores no se podrán adaptar rápidamente al cambio digital y esto perjudicaría su desempeño como empresa. Otra razón importante, es el cambio cultural que tienen que afrontar los líderes de las empresas para promover estas tecnologías y así conocer más acerca de los beneficios que le puede traer a su empresa con diferentes niveles de inversión.

1.2 Uso de las aplicaciones móviles para mejorar procesos en el Perú

Es evidente el uso de las aplicaciones móviles en diferentes rubros de las industrias por sus beneficios, por su practicidad y porque está abocado a un dispositivo muy simple de utilizar, además del gran potencial que presentan para innovaciones, como se concluye en [2]. Estos dispositivos se han difundido ampliamente y son utilizados para diferentes procesos por su gran capacidad que han adquirido en los últimos años para poder procesar cada vez más información y su performance que permite utilizar muchos sensores,

componentes, cámaras, entre otros. Por lo cual los dispositivos móviles son herramientas muy potentes y eficientes. A pesar de las ventajas y la gran cantidad de aplicaciones que existen muchas requieren de una personalización muy particular. Tales el caso de los hospitales que requieren de dispositivos móviles que vayan de acuerdo con los departamentos que poseen las necesidades principales, contemplar las jerarquías que poseen los diferentes usuarios que hay dentro del centro de salud y las funciones que realiza. Es evidente que deben establecerse ciertos roles para el acceso a la información y la manipulación de los equipos, por lo que se hace necesario mostrar diferentes interfaces con diferentes flujos. Es por lo mencionado, principalmente, que el proceso de desarrollo de una aplicación tiene que tomar distintos aspectos y contemplar de manera cercana las necesidades de cada uno de los perfiles de usuarios que hay en el hospital. Dado el nivel de personalización, los centros de salud han preferido mantener métodos tradicionales para gestionar y procesar los requerimientos. Son muy pocos los que han invertido en alguna de estas tecnologías, por lo que se puede apreciar que la realidad de la adopción tecnológica en el país no es mucha. Según la OMS en [4], sobre las aplicaciones móviles para la salud (conocidas como mSalud), se presentan los siguientes problemas:

- La multiplicidad de proyectos experimentales sin ningún plan sólido definido de ampliación.
- Falta de integración entre las diferentes aplicaciones con las estructuras nacionales y con las estrategias nacionales de ciber salud.
- Ausencia de normas y herramientas para la evaluación comparativa de la funcionalidad, la posibilidad de ampliación y el valor

comparativo de las soluciones de mSalud, lo que da lugar a una falta de datos para articular la orientación normativa.

- Falta de enfoque multisectorial dentro del gobierno, especialmente en la colaboración entre los ministerios de salud y los ministerios de tecnologías de la información y la comunicación, y de recomendaciones normativas de colaboración con el sector privado.

Estos puntos nos brindan un buen panorama sobre los problemas que debemos evitar, prevenir y en caso se den tener mecanismos para afrontarlos.

1.3 Problemática a solucionar

Para tener un mejor enfoque sobre la problemática presente en los centros de salud se elaboró el siguiente árbol de problemas.

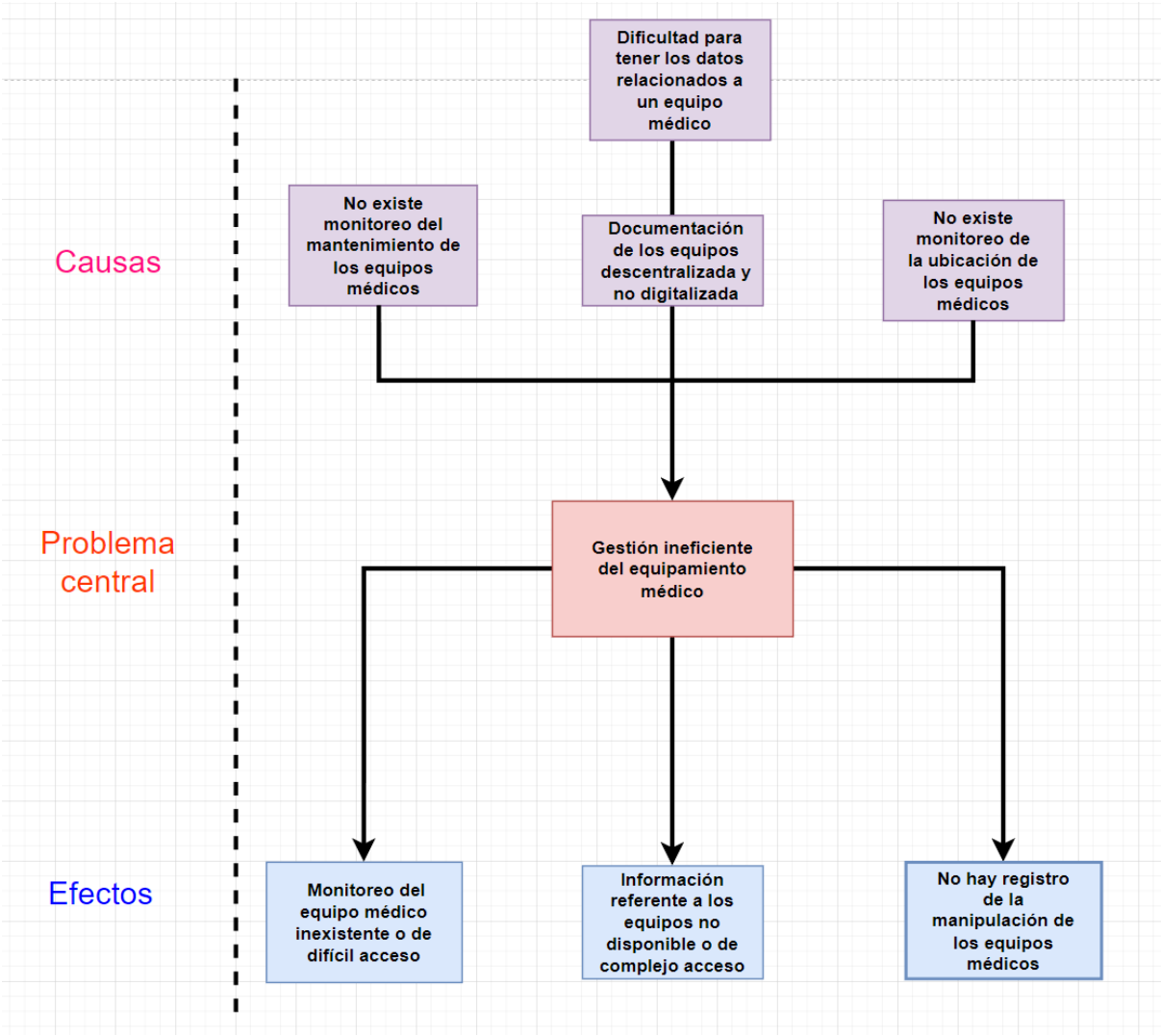

Figura 1 Árbol de problemas

Fuente: "Elaboración propia"

1.4 Objetivos

En esta sección se elaboró un árbol de soluciones también que ilustre los objetivos de nuestra aplicación, tal como se puede observar en la Figura 2**.**

1.4.1 Objetivo principal

El principal objetivo del desarrollo de esta tesis es brindar una aplicación móvil que haga uso de servicios en la nube para almacenar información y automatizar recordatorios para el mantenimiento con una infraestructura que

brinde alta disponibilidad y garantice la seguridad de la información considerando el caso específico de un centro de salud.

1.4.1 Objetivos secundarios

- Centralización de documentos de los equipos médicos.
- Evitar pérdida de ingresos al centro de salud y perjuicio a clientes por daño de equipos debido a la falta de mantenimiento.
- Mantener un correcto monitoreo de los equipamientos médicos, para que se pueda incrementar su ciclo de vida al realizar los mantenimientos rutinarios correspondientes en el tiempo correcto.
- Mantener correctamente ubicados los equipamientos médicos y el(los) responsable(s) de cada uno de ellos.
- Implementar una solución robusta de bajo coste de implementación y mantenimiento que cumpla con estándares de disponibilidad y seguridad.

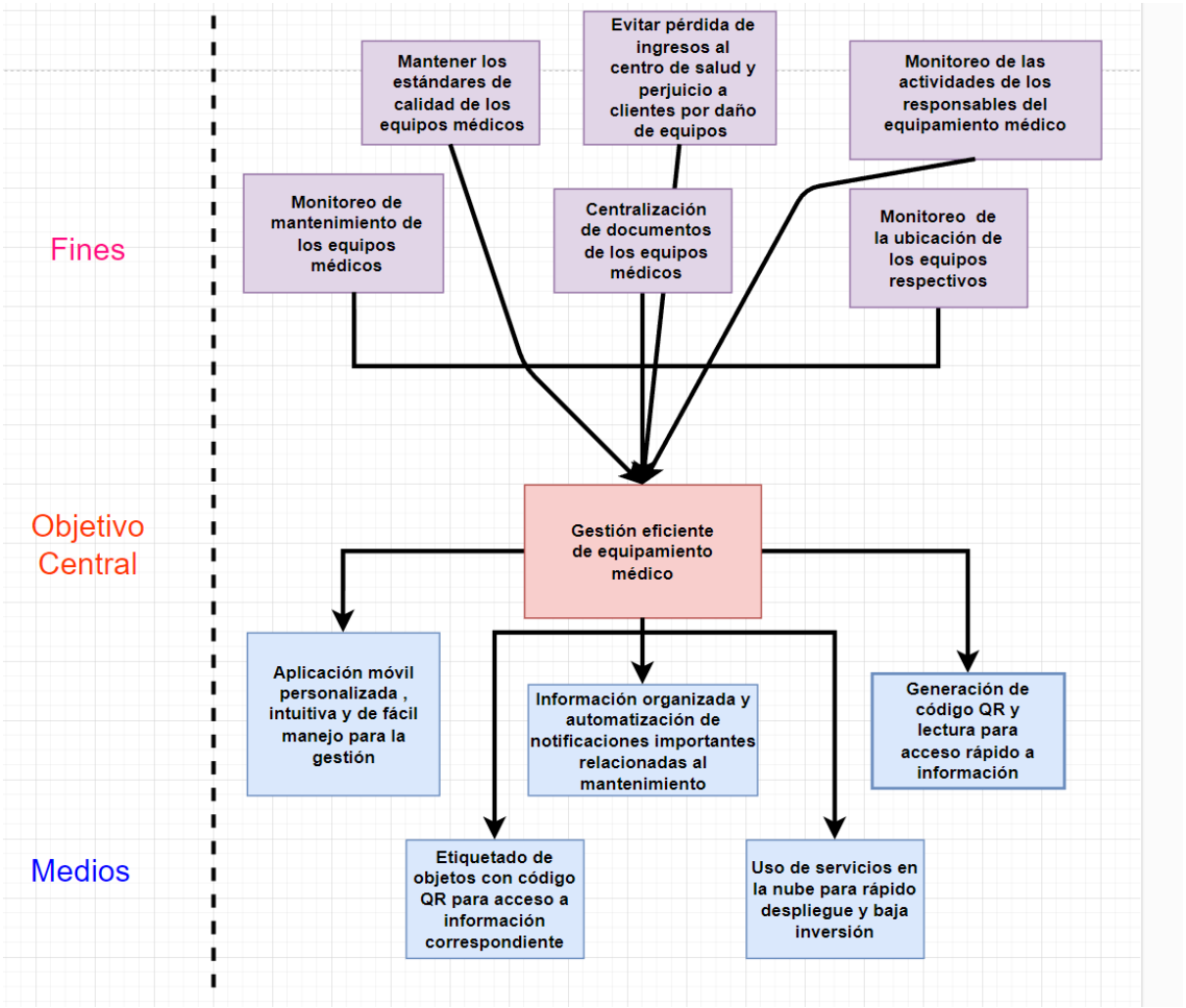

Figura 2:Árbol de objetivos

Fuente: “Elaboración propia”

Capítulo 2
Análisis de requerimientos y tecnologías para el desarrollo de la solución

En esta sección se evaluarán los requerimientos de los centros de salud, presentarán y analizarán las tecnologías disponibles para el desarrollo de la solución. La presente investigación pretende ser aplicada a diferentes centros de salud, pero abordaremos como modelo al Instituto Nacional de Enfermedades Neoplásicas (INEN).

2.1) Necesidades de los centros de salud

Los centros de salud deben desenvolverse de una forma rápida y eficiente por tanto todas las operaciones logísticas deben realizarse de la misma forma. Tales operaciones como el registro del equipamiento médico, inventario de los utensilios en el centro de salud, preparación de ambulancias, registro de citas, entre otras operaciones internas deben tener un grado de automatización que acelere dichas operaciones. Por tanto, una aplicación que

permita una buena gestión del equipamiento médico podrá aportar en esta línea ágil.

2.2) Estructura orgánica de centros de salud

Como se puede apreciar en la Figura 4 los centros de salud poseen diversos departamentos y cada uno de ellos cuenta con equipamiento dedicado para poder atender correctamente a los pacientes, pero, también, necesitan de la compartición de estos entre diversas áreas, por lo que se hace complejo el seguimiento de los equipamientos médicos involucrados en los intercambios.

2.3) Gestión de los equipos médicos en centros de salud

La importancia de los equipos médicos ha ido creciendo desde que se han implementado en la medicina. Por consiguiente, el mantener estos dispositivos óptimos para las diversas operaciones y que se conserven con las mejores características a lo largo del tiempo son puntos cruciales. Su rendimiento debe de mantener la efectividad y confiabilidad para poder realizar las funciones que le competen. En ese sentido, se hace urgente tener un buen control de cada uno de los equipamientos médicos presentes en un centro de salud para garantizar una buena operación y un buen servicio por parte de estos. Además, cabe mencionar que los equipos poseen diferentes tipos de mantenimiento (predictivo, preventivo y correctivo [4]), diferentes intervalos de tiempo, diferentes características, por ello la personalización de cada uno de estos es importante.

2.4) Herramientas potenciales para la implementación

En esta sección se evaluarán algunas de las principales herramientas que se puedan integrar en nuestra solución para posteriormente tomar una decisión

más consensuada. Como base se considerará la arquitectura mostrada en la Figura 2; sin embargo, esto es como etapa inicial ya que después del análisis se realizarán modificaciones para obtener la solución óptima.

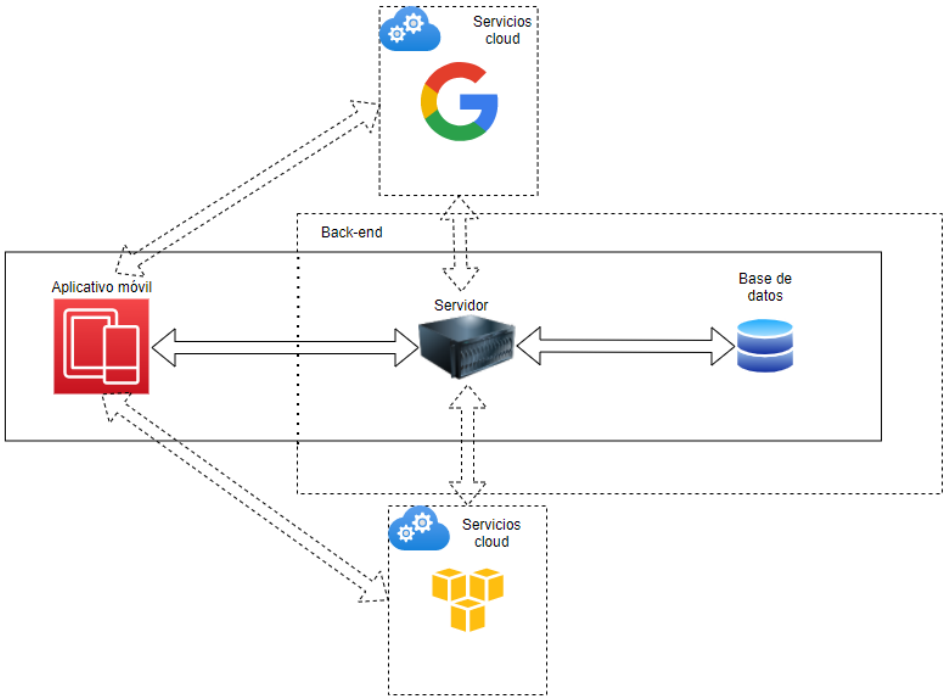

Figura 3 : "Arquitectura base a analizar"

Fuente: "Elaboración propia"

2.4.1) Servidor Backend

Este será el servidor al cual se le realizarán la consulta de los datos (ya que tendrá conexión con la base de datos), procesamiento de estos para enviar al *frontend*, entre otras operaciones similares. Un servidor *Backend* se puede desplegar bajo dos modalidades: *on premise* y en la nube.

2.4.1.1) *Backend on-premise*

Para este fin se requiere de servidores que posean el almacenamiento suficiente y almacén de los datos. Para esto se requiere la compra de los

equipos y abastecer de todos los requerimientos tales como temperatura, fuentes de poder, redundancia, seguridad, entre otros. A su vez debe ser mantenida a lo largo del tiempo y supervisada para que mantenga el rendimiento adecuado y capacidad de respuesta ante fallos. Naturalmente, estos equipos de telecomunicaciones requieren ser renovados a lo largo del tiempo.

2.4.1.2) *Backend* desplegado en la nube

Las soluciones de tecnologías de la información en los últimos años están migrando a la nube, porque se reducen los costos de inversión para la implementación (CAPEX) y se tienen costos de operación (OPEX). Además, se reciben exactamente los mismos beneficios en cuanto a un servidor local incluso con garantías adicionales. Los costos de mantenimiento, la redundancia y otros requerimientos al ser tercerizados pasan a responsabilidad de la compañía en donde está alojado nuestro sistema. Se dará una breve revisión general a los principales proveedores y compararemos los servicios que más se amolden al proyecto. Para ilustrar mejor los tipos de servicios se utilizará la Figura 2.

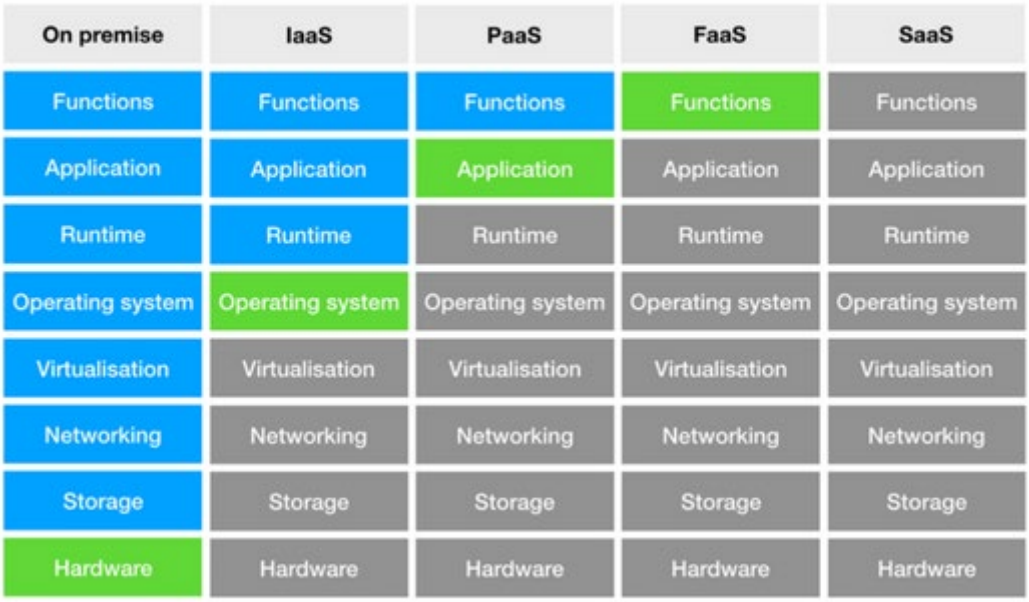

Figura 4: "Tipos de servicios en nube"
Fuente: [5]

En primer lugar, se encuentran los servicios conocidos como IaaS (*Infraestructura as a Service*), y como se ve en la gráfica la abstracción es desde el *hardware* hasta la virtualización. Después se encuentra las plataformas como servicios, se encuentra solamente el manejo de la aplicación y las funciones que se implementarán en ella. Luego, FaaS (*Function as a Service*) solo se insertan porciones de código para que sean ejecutadas funciones específicas. Finalmente, se encuentra SaaS (*Software as a Service o Solution as a Service*) que quiere decir que se entrega una solución completamente integrada para brindar algunos servicios de uso directo.

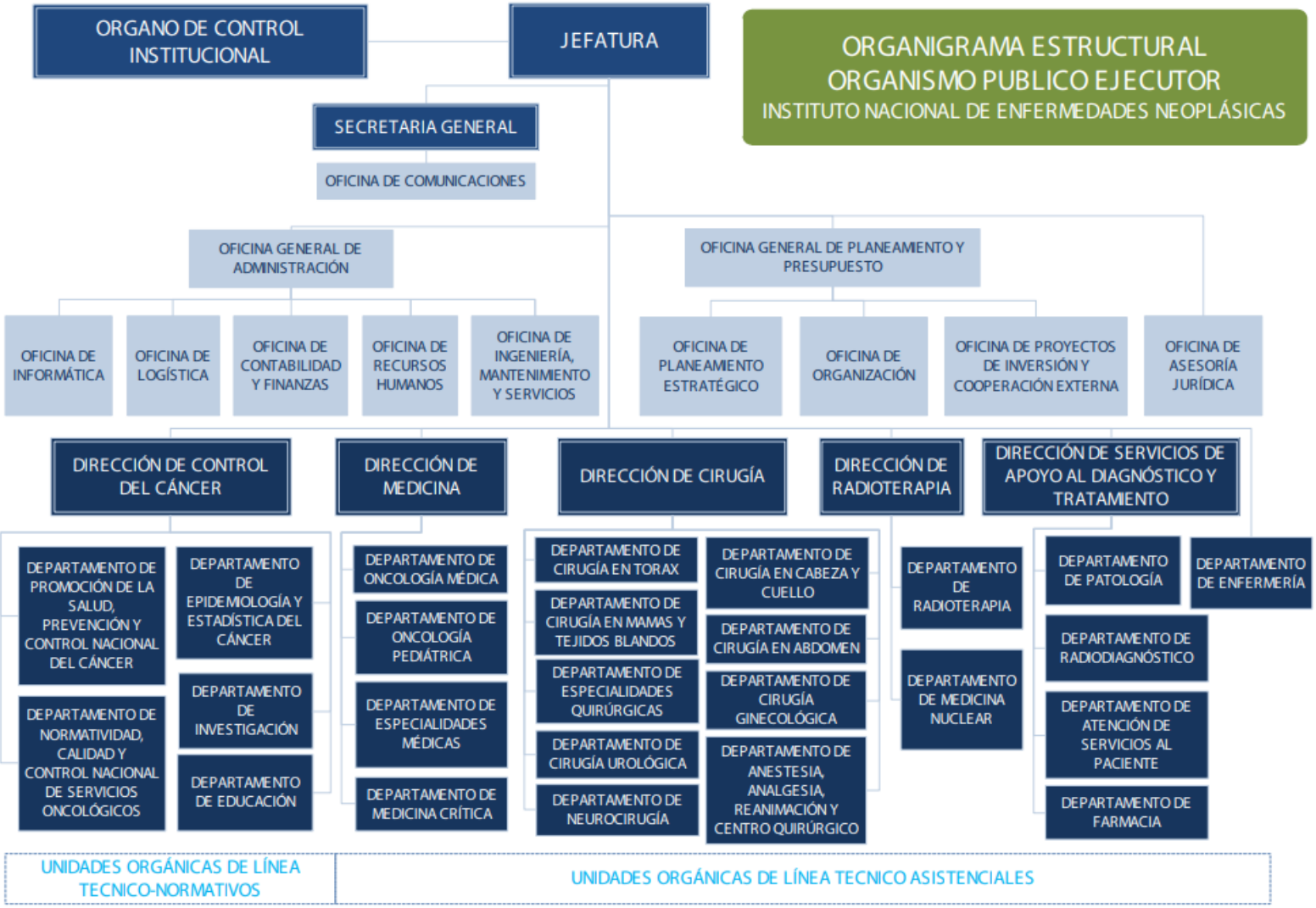

Figura 5: “Organigrama del INEN”

Fuente: [6]

Un punto importante es que la alta disponibilidad de los servicios está garantizada, es decir, el servicio debe estar siempre activo para su uso. Además, se debe considerar que no es un elemento crítico, no hay una fuerte dependencia que pueda afectar a la institución pública en caso se pierda la conexión. También, cabe resaltar que el aplicativo puede estar en teléfonos móviles, además de tabletas, y por tanto pueden acceder a una red de datos.

2.4.2) Sistema operativo de los dispositivos

Los dispositivos requieren ser desplegados en distintas partes de la clínica entre ambientes principales y hasta móviles para que los usuarios puedan desplazarse con estos. En vista de la gran cantidad de equipos presentes en el mercado que cuentan con el sistema Android y IOS según la **Figura 4** se ha optado por escoger estos sistemas ya que agregan un abanico de posibilidades en cuanto a modelos de los dispositivos y otros. Se ven tendencias marcadas que nos definen los sistemas operativos para desarrollar nuestra aplicación móvil. Cabe resaltar el hecho de que Harmony OS, al poseer una gran cantidad de clientes por la marca de celulares Huawei, entrará pronto al mercado, por lo que se debería tener como referencia también.

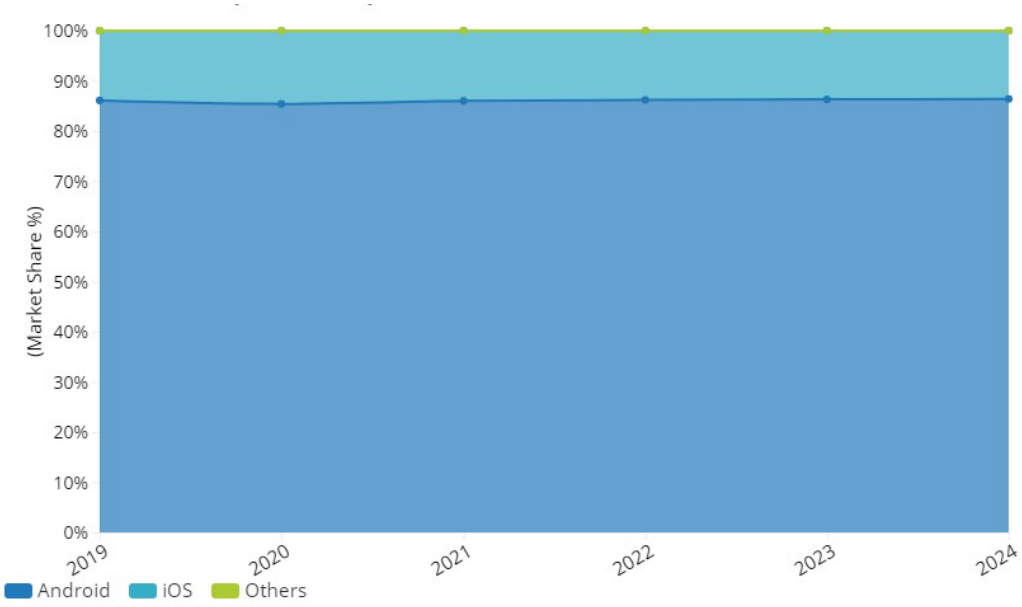

Figura 6 “Compartición del mercado”

Fuente: [3]

Aunque aún no se considera como una tendencia a futuro por algunas fuentes (como en la figura 3), porque aún no habían entrado fuertemente al mercado. Sin embargo, muchas aplicaciones pueden ser desplegadas en el sistema operativo de los móviles Huawei, debido que soporta diversos lenguajes [7]. En consecuencia, si se requiere una aplicación en otro lenguaje como Java, Javascript, etc. (revisar documentación), posteriormente se podría realizar el lanzar el aplicativo en la AppGalllery siempre y cuando cumpla las características mencionadas en [8]. Entonces, teniendo en cuenta estos sistemas operativos y la decisión de las empresas debemos implementar una solución que pueda dar flexibilidad en cuanto a los sistemas operativos que utilice el centro de salud.

2.4.3) Tipos de aplicaciones móviles

Entre los distintos tipos de aplicaciones móviles se encuentran las aplicaciones nativas, las aplicaciones híbridas, las aplicaciones web progresivas y las plataformas de desarrollo cruzado nativas.

2.4.3.1) Aplicaciones nativas

Son aplicaciones desarrolladas con un lenguaje específico del sistema operativo que solo permite que compile en este. Tal es el caso de Java y Kotlin para Android; Swift y Objective-c para el caso de IOS. Estos al ser desarrollados en el lenguaje propio del sistema operativo tienen acceso a todas las funcionalidades tales como los sensores (acelerómetro, cámara, etc.) de manera sencilla y rápida, por lo que ofrece un rendimiento superior que en ocasiones puede ser importante.

2.4.3.2) Aplicaciones híbridas

Las aplicaciones híbridas, en cambio, son desarrolladas en contenedores web View de una aplicación nativa, por lo que usan HTML, CSS y JavaScript. La gran ventaja es que se estas pueden ser usadas en IOS como en Android y las interfaces aparentarán como las nativas. La gran desventaja es que se merma el rendimiento dado que no está compilada en código nativo de la aplicación.

2.4.3.3) Aplicaciones web progresivas (PWA)

Son aplicaciones web que mediante. Son las más limitadas para acceder a todas las funcionalidades del equipo, pero son muy ligeras y, en ocasiones, responden más rápido que las aplicaciones nativas. Permiten una transformación rápida de una página web *responsive* (la interfaz se ajusta a la pantalla del dispositivo) a una aplicación móvil que pueda ser publicada en las tiendas de aplicaciones. Es muy útil si lo que se diseñó en primera instancia es una página web.

2.4.3.4) Plataformas de desarrollo cruzado nativas

Estas aplicaciones tienen la ventaja que pueden compilarse en varias plataformas y tener un buen rendimiento. Naturalmente, no poseen todas las características que puede poseer una aplicación nativa como todos los sensores, pero su desarrollo ha permitido implementarlos fácilmente y en muchos casos satisfacen las características que requieren las soluciones. A diferencia de las PWA estás se ven como aplicaciones nativas y no como una página web, además de otras características como acceso a mayor cantidad de sensores, implementación de interfaces similares a las nativas tanto para IOS y Android de forma rápida.

2.4.3) Modelos de bases de datos:

Entre los modelos de base de datos más comunes están las bases de datos relacionales y las que no son relacionales. Cada una de ellas con sus ventajas las cuales se analizarán en los siguientes incisos.

2.4.3.1) Bases de datos relacionales

Como su nombre lo menciona es una base de datos que se organiza a través de relaciones entre las tablas. En la tabla se muestran las diferencias más importantes entre dos de los principales gestores de bases de datos. Basándonos, en la comparativa hecha en la tabla 1

Mysql Database:

Según [9], este sistema de gestión de base de datos es multihilo y, a su vez, permite la conexión con varios a la vez. Brinda seguridad para el acceso de usuarios restringiendo privilegios y permisos sobre tablas o acciones que

pueden realizar. Es portable ya que usa SQL el cual es un lenguaje estandarizado y se pueden portar a otros sistemas y plataformas las consultas.

Oracle Database:

Este sistema es de paga y posee soporte por parte de la propia empresa. Tiene la gran ventaja de que puede almacenar los datos bajo diferentes modelos lo que se conoce como base de datos multi modelo por lo que soporta modelos de documentos, gráficas, relaciones y llave-valor dentro de las bases de datos.

Feature	Oracle	MySQL
Interfaz	GUI, SQL	SQL
Lenguaje Soportado	C, C#, C++, Java, Ruby y Objective C, entre otros	C, C#, C++, D, Java, Ruby y Objective C, entre otros
Sistema Operativo	Windows, Linux, Solaris, HP-UX, OS X, z/OS, AIX	Windows, Linux, OS X, FreeBSD, Solaris
Licencia	Propietario	Código Libre

Tabla 1:" Comparativa entre Oracle y MySQL"

Fuente: [10]

2.4.3.2) Bases de datos no relacionales:

En la actualidad son ampliamente utilizadas en diferentes aplicaciones, en especial, las que manejan una gran cantidad de datos y requieren tener una baja latencia en la respuesta o consultas posteriores[11] . Por lo que es muy efectiva. Entre los principales sistemas de bases de datos está MongoDB y Apache Cassandra. Analizaremos, el más popular que es MongoDB y por otro lado evaluaremos una solución que se nos brinda, a través, de un servicio.

MongoDB:

Este lenguaje orientado a documentos utiliza el documento tipo JSON como estructura para guardar los datos. Es ampliamente utilizado por diversas compañías por su buen rendimiento y solidez en documentación.

Firebase real time database:

Este servicio que forma parte de los servicios de Firebase, el cual es de Google, posee la gran ventaja de que no se debe de implementar, ni configurar nada en un servidor dado que es una base de datos como servicio. Además, de esto puede ser integrado con otros servicios propios de Firebase como la autenticación con Gmail, Facebook, mensajes automáticos de confirmación de correo, entre otros, de forma rápida y segura, y la información está cifrada automáticamente por el servicio que utiliza el protocolo seguro (HTTPS). Este servicio de Google posee la versatilidad de estar desarrollado para ser integrado en diferentes plataformas y está empaquetado en herramientas de desarrollo, esto da una gran ventaja ya que facilita la integración con aplicaciones móviles y nos permite aprovechar, rápidamente, mensajería instantánea, notificaciones de algún evento, entre otros.

2.4.4) Arquitecturas de software

En esta sección se analizarán algunos patrones. Las arquitecturas

2.4.4.1) Patrón MVC

Este patrón es muy utilizado para diferentes plataformas, por su practicidad y ha respondido correctamente. Sus siglas hacen referencia a las capas modelo vista controlador. Una imagen que ilustra bien este modelo es la figura 3.

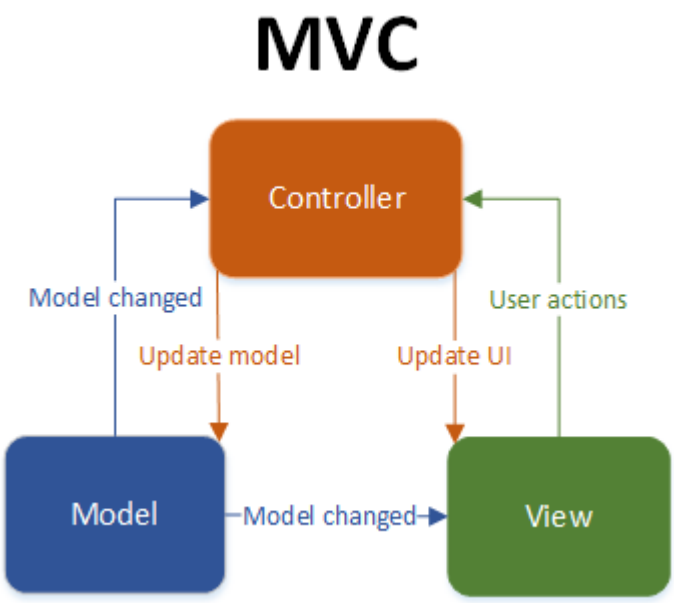

Figura 3 [12]

Modelo: Este componente administra las conexiones con las bases de datos, en general, brinda toda la información que requiera el controlador para realizar alguna tarea. Interactúa, también, con la capa de la vista brindándole información para que muestre al usuario.

Controlador: Se encarga de la parte lógica, toda acción que realiza el usuario pasa por esta capa, es el intermediario entre la vista y el modelo. Hace el procesamiento necesario, gestiona los permisos, ordena una actualización en la vista y pide información al modelo para procesarla.

Vista: En esta capa se muestra al usuario lo recopilado desde el modelo y se da la actualización de la *user interface* (UI), donde se puede ver lo que se ha procesado de forma entendible. En ocasiones se utiliza un objeto Observer para permitir la comunicación directa entre la capa de modelo y vista, para no sobre cargar al controlador.

2.4.4.2) Patrón MVP

Esta arquitectura es derivada del patrón anterior.

La diferencia principal radica en que todo está centralizado en el presentador para facilitar las pruebas unitarias automatizadas.

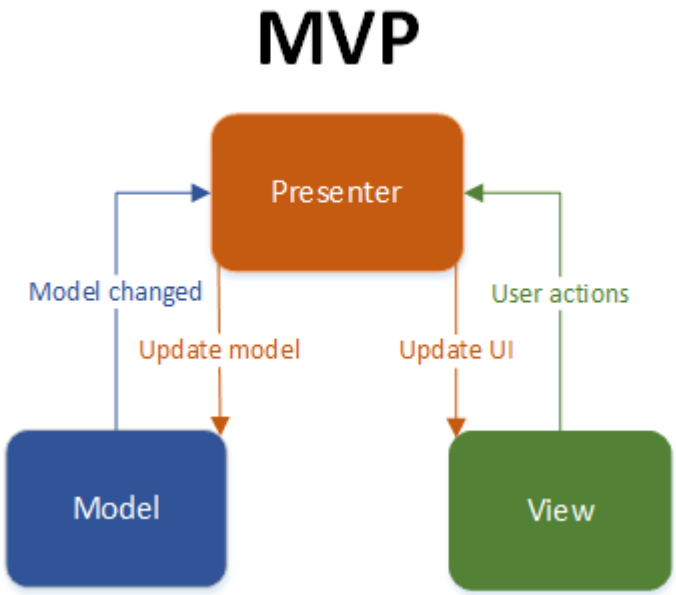

Figura 4 [12]

Modelo: En esta capa está la lógica del negocio o del propósito de la aplicación.

Vista: Presenta los datos enviados por el presentador y el modelo. Envía las solicitudes de los usuarios (eventos) al presentador.

Presentador: Aquí está la lógica de presentación, es decir, lo que se mostrará en la interfaz de usuario. Como intermediario envía y recibe actualizaciones del modelo y notifica (en caso sea necesario) una actualización a la vista con los cambios ocurridos.

2.4.4.2) Patrón MVVM

El patrón

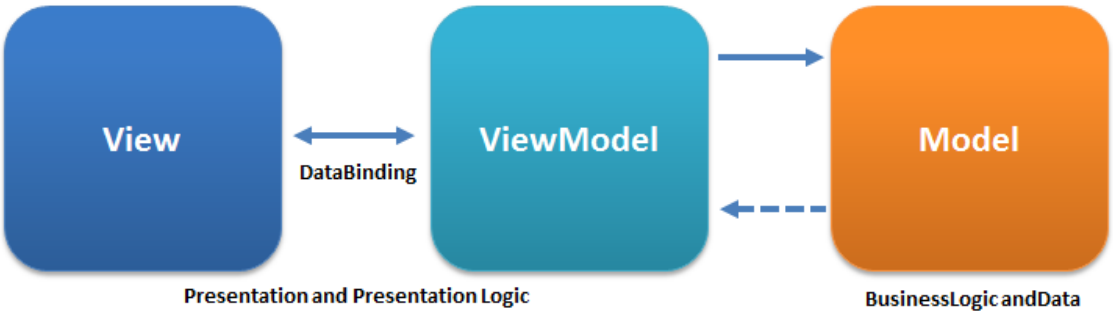

Figura 5 [12]

Modelo: similar a los patrones anteriores, en este componente se define la lógica de negocio y los datos necesarios que se mostrarán en la vista.

ViewModel: En este caso, es el intermediario al igual que en los otros patrones, pero, ahora, está desligado del View, ya que no sabe con qué vista está interactuando.

View: Es la interfaz pasiva que recibe la información que será mostrada para el usuario y los eventos que este haga para trasladarlo al ViewModel.

Binder: Se requiere para la implementación de este patrón para que las capas puedan interactuar, ya que permite el enlace de datos declarativo (*data-binding*), de esta forma se vinculan los datos observables a la interfaz de usuario y se reduce considerablemente la cantidad de código.

2.4.4.4) Patrón BLoC:

Este patrón tiene una diferencia notable respecto a los demás, ya que en este la comunicación entre sus capas no es bidireccional, sino que únicamente en una dirección, lo que permite que las depuraciones sean más rápidas.

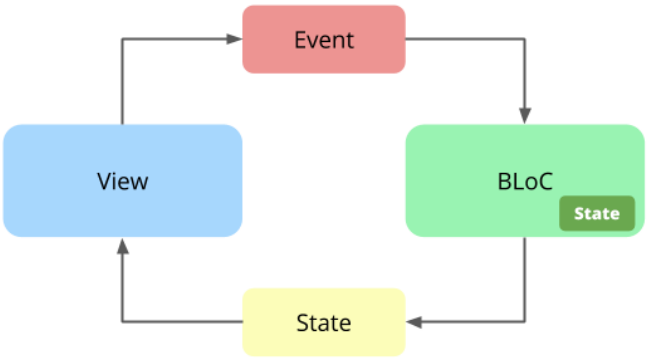

Figura 6 [12]

View: En este segmento es donde se gestiona la interfaz de usuario y los elementos que se visualizarán para cada ventana.

Bloc: Se almacenan los métodos, propiedades de nuestra lógica de negocio o aplicación a la que está orientada nuestro proyecto.

State: Nos indica cómo se encuentra el Bloc en determinado momento.

Event: A través de la gestión de los eventos vamos a modificar los estados del BloC.

Cabe resaltar que este es patrón fue desarrollado por Google en 2018.

2.4.3) Modelos de IDE (Integrated Development Enviroment)

Un entorno integrado provee de herramientas para programadores para el desarrollo de software. Este como mínimo debe brindar editor de código fuente, herramientas para construir la aplicación rápidamente y un depurador.

En esta sección se analizará el entorno de desarrollo más adecuado y óptimo dependiendo de los beneficios que ofrecen dos de los principales entornos de desarrollo para aplicaciones móviles.

2.4.3.1) Visual Studio Code (VSCode):

El entorno de desarrollo Visual Studio desarrollado por Microsoft es de código abierto y posee diversos plugin que permiten agregarle mejoras. Tiene muchas características adicionales que permiten la colaboración entre compañeros o para trabajar en equipo. Además, de ello una sus principales características es que es bastante ligero. [13]

2.4.3.4) Android Studio:

Este IDE desarrollado por IntelliJ IDEA fue planteado principalmente para el desarrollo de aplicaciones en Android y con Java. Sin embargo, actualmente, posee muchas más características, tales como la programación en plataformas cruzadas como con Flutter y la programación con Dart. [14]

El siguiente cuadro comparativo muestra las características más importantes de ambos entornos.

	Visual Studio Code	**Android Studio**
Sistema compilador	Soporta varios (Maven, Gradle, etc.)	Gradle
Emulador de Móvil	No	Sí, y cargado de funciones relacionadas con sensores, entre otros.
Integración con git	Sí, tiene una extensión para GitHub	Sí, con GitHub
Consumo de	Bajo	Alto

recursos		
Cantidad de plugins	Muchos	moderado

Tabla 2 Fuentes [14] y [13]

2.4.5) Código para etiquetado

Para esa sección se evaluará dos formas de etiquetas para el equipamiento médico para que se pueda acceder rápidamente a este para su respectiva gestión.

2.4.5.1) Código QR

Son ampliamente utilizados por diferentes industrias debido a su facilidad y a que puede ser rápidamente leído a través de dispositivos móviles, por lo que puede ser implementado rápidamente como solución para nuestro sistema. Son de dos dimensiones en cuadrado en cual se realizan más cuadrados pequeños y es ahí donde se guarda la información. Además, existen muchas formas para generar estos códigos desde muchas plataformas como NPM, API' s de terceros, etc.

2.4.5.2) Código de Barras

Son utilizados principalmente en las industrias alimentarias, supermercados, en dispositivos, entre otras. Son de una dimensión y el código está en la división de las líneas paralelas negras que varían del mismo ancho. Sin embargo, para la solución que queremos implementar no podrían aportar mucho, dado que los dispositivos móviles no tienen mucha facilidad para leer este tipo de códigos. Se comparará el código de barras más utilizado en

negocios minoristas de E.E.U.U. que es el *Universal Product Code* (UPC).[15]

2.4.5.3) Etiquetas NFC

La tecnología inalámbrica NFC (*Near Field Communication*), que opera en la banda de los 13.56 MHz y alcanza una tasa de transferencia 424 kbit/s,[16] lo que para los fines de nuestra aplicación es suficiente, está siendo una tendencia a nivel global y sus aplicaciones son muy diversas, lo que hace que sea una opción a considerar para muchas soluciones TI. Las etiquetas NFC son un subtipo de las etiquetas RFID, se pueden sobrescribir una gran cantidad de veces dependiendo del vendedor pueden ser hasta 100 000 veces. Es posible bloquear los tags para evitar su sobreescritura, además de poner una contraseña para restringir la escritura de estas. Posee una gran versatilidad dadas las características mencionadas en los puntos anteriores.

2.4.5.4) Etiquetas RFID

Las etiquetas NFC son un subtipo de las etiquetas RFID, en consecuencia, presenta características muy similares. Sin embargo, si se opera a una frecuencia distinta a la de NFC, no se podrá acceder o interactuar desde el dispositivo móvil, ya que los dispositivos móviles no usan otro rango de frecuencias que no sea la banda de 13.56 MHz para este tipo de comunicaciones. Por lo tanto, sería necesario tener un lector RFID para realizar la lectura de nuestras etiquetas.

	Código QR	Código de barras	RFID	NFC
Cantidad de información	4296 caracteres en alfanumérico	Números de 11 dígitos	Hasta 2 KB	96-512 bytes
Corrección y detección de errores	Sí	No, solo detección	Es posible su configuración	Es posible su configuración
Identificador único	Sí	No, si existen productos similares se utilizará el mismo identificador	Sí.	Sí.
Facilidad de escaneo	Fácil, se puede leer de cualquier dirección	No, es más rígido, solo vertical	Fácil, pero se requiere de un lector especial.	Fácil, pero se requiere de cercanía.
Facilidad de mantenimiento	Sencilla, se reimprimen los códigos QR.	Sencilla, se reimprimen los códigos de barra.	Moderada, se requiere de equipo para reconfigurar los RFID's.	Moderada, se requiere de equipo para reconfigurar los tags.
Costo de implementación	Bajo	Bajo	Alto	Moderado

Tabla 3: "Comparación de código QR vs Código de barras"
Fuente: [16][15], [17]

Respecto al costo de implementación se realiza un cálculo aproximado de los montos comparando los precios presentes en el mercado:

En el caso del código QR:

El costo por un pliego de vinil adhesivo (material más adecuado para este propósito) 140x90 cm que posee códigos QR de 5 cm2 con una separación de 5mm, por lo que tendríamos 400 por pliego y se estima unos 2000 equipos médicos a ser etiquetados. El valor de cada pliego es de 85 soles. Por tanto, la suma ascendería a 425 soles por los 4 pliegos requeridos.

En el caso del código NFC:

Cada uno de estos códigos NFC poseen un costo de 2 soles, por lo que para los 2000 que se estiman serán 4000 soles. Además, la configuración es más compleja, que los códigos QR.

Capítulo 3
Diseño e implementación del sistema

En el presente capítulo se justificará las tecnologías escogidas a partir del análisis realizado en el capítulo 2, a fin de evidenciar que son las más eficientes y eficaces para el presente proyecto de tesis.

3.1) Arquitectura a desarrollar

Para el diseño y la implementación a partir del análisis desarrollado en el capítulo anterior, y que se detallará en este se expondrá los cimientos para el desarrollo de la solución. La arquitectura más eficiente que cumple con los requerimientos es la que se visualiza en la Figura 6. En las siguientes secciones se ampliará y explicarán las razones de la elección.

.

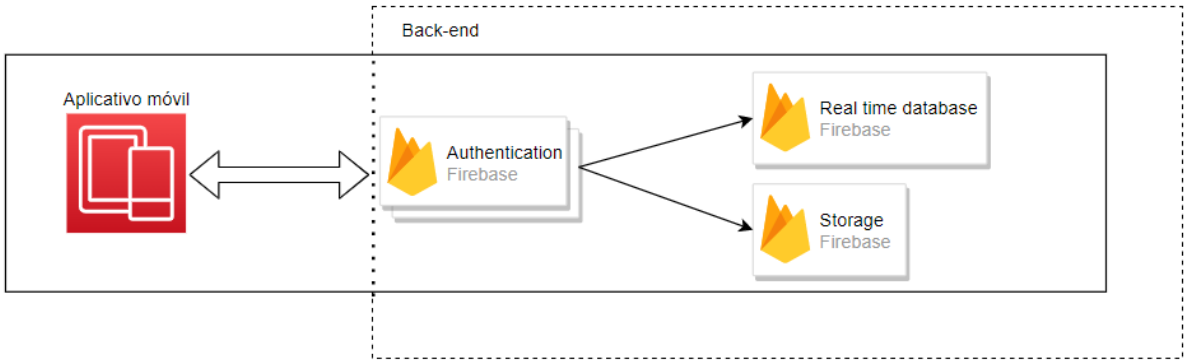

Figura 7 : "Arquitectura de la solución"

Fuente: "Elaboración propia"

3.2) Etapas del proyecto

En la primera etapa, como se observa en la se desarrollará el flujo lógico de nuestra aplicación que se detallará más adelante. En la segunda etapa, se realizará el desarrollo de la solución que incluye la generación y lectura de

los códigos QR para los equipamientos médicos, después se realizará el desarrollo de las interfaces. En la siguiente etapa, se realizará el despliegue de base de datos. Después de ello se compilará y se generarán los archivos para su uso, finalmente en la interacción con los usuarios, ellos podrán realizar todo lo que les corresponda.

3.3) Selección de tecnologías

En esta sección destacaremos las tecnologías que son las más convenientes para la solución a implementar.

3.3.1) Selección de tipo de aplicación a desarrollar

Para resumir las principales características que poseen los diferentes tipos de aplicaciones móviles, se realiza el siguiente resumen en un cuadro comparativo.

Feature	Magento PWA	Responsive Website	Hybrid App	Native App
Multi-platform compatibility	+	+	+	-
Linkable	+	+	-	-
Search engine indexed	+	+	-	-
Offline mode	+	-	- (not in eCommerce)	- (not in eCommerce)
Installable	+	-	+	+
Behind-the-scene updates	+	+	-	-
Push notifications	+	-	+	+
Cost and time efficient development	+	+/- (not efficient if you need to develop an app also)	-	-
Distribution	Web	Web	App store	App store
Mobile-first design	+	-	+	+
Require more space	-	-	+	+
Hardware-heavy	-	-	+	+
App stores' limitations	-	-	+	+
Memory friendly	+	+	-	-
Content shareability	+	+	-	-

Tabla 2: "Cuando comparativo de los tipos de aplicaciones"

Fuente:[2]

Se resalta de esta tabla el hecho de la penetración que pueden tener en los dispositivos, es decir, si es posible su despliegue multiplataforma y el rendimiento que ofrecen en estos. Por tanto, se ha visto como la opción óptima la plataforma de desarrollo cruzado, porque nos permite crear

aplicaciones para los dispositivos existentes en el mercado, la buena interfaz que suelen ofrecer los *frameworks* para el desarrollo y por su buen rendimiento que, en general, es cercano al nativo. Adicionalmente, nos permite el acceso a sensores esenciales de forma rápida, aunque en menor cantidad que las nativas, pero suele ser más soportada que los otros tipos de aplicaciones. Ahora bien, existen dos opciones factibles para el desarrollo de este tipo de aplicaciones

3.3.1.1) React Native:

Este framework hace uso del motor de JavaScript que incluyen integrado y lo posee tanto Android como iOS para ejecutar en sus navegadores web los *scripts*. Por tanto, al correr nuestra aplicación en la máquina virtual de este dispositivo poseeremos acceso a las capacidades nativas del dispositivo.[18]

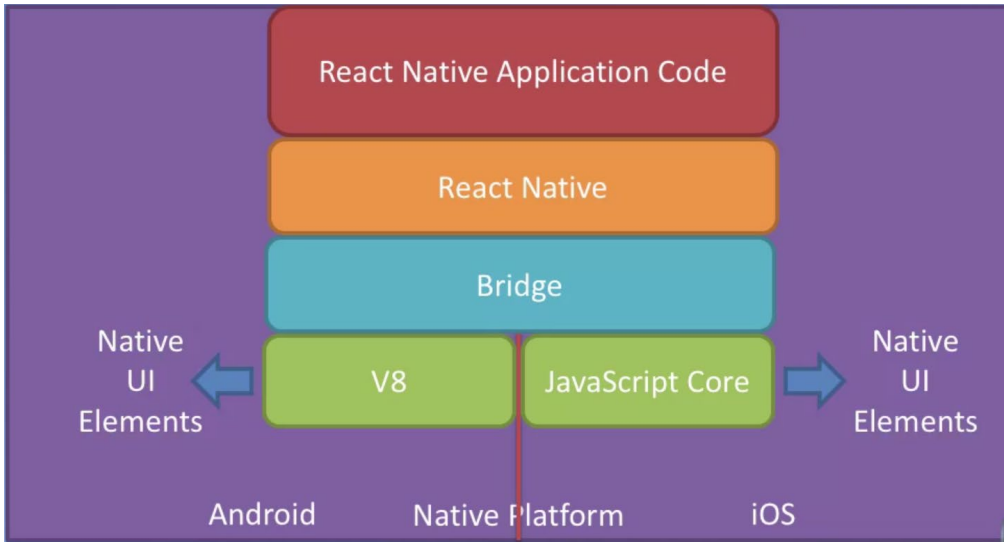

Figura 8: "Uso de JavaScript en Android y IOS"

Fuente:[18]

En esta gráfica se puede apreciar como la aplicación escrita en JavaScript y usando React Native, puede ser utilizado en ambos dispositivos y brindar beneficios con alto rendimiento.

3.3.1.2) Flutter:

Este *software development kit* (SDK) desarrollado por Google, es escrito sobre Dart y posee la virtud que se compila a lenguaje de nativo tanto de Android y iOS por lo que tiene acceso también a sus capacidades nativas. Aunque, en algunos casos, es complejo el acceso a capacidades nativas y poseen algunos errores cuando se quiere acceder a dichas características porque aún se están desarrollando mejoras, ya tiene sólida documentación y una comunidad grande que permite corregir esos errores a su vez que brinda recomendaciones para un mejor rendimiento de la aplicación. Su buen funcionamiento se debe a que minimiza la abstracción (quita algunas capas) que posee un lenguaje de plataformas de desarrollo cruzado nativas. Además, hace que la interacción sea más rápida ya que reduce las partes que se deben actualizar en una interfaz, al localizar únicamente los *Widgets* que requieren de un cambio.[19]

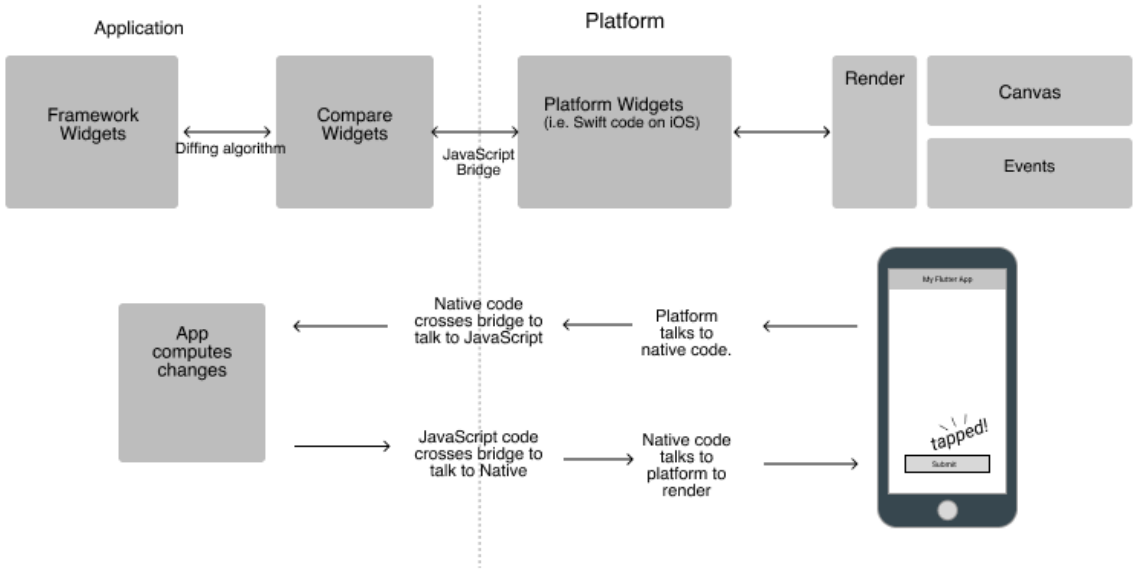

Figura 9: "Flujo de interacción con la UI en React Native"

Fuente: [20]

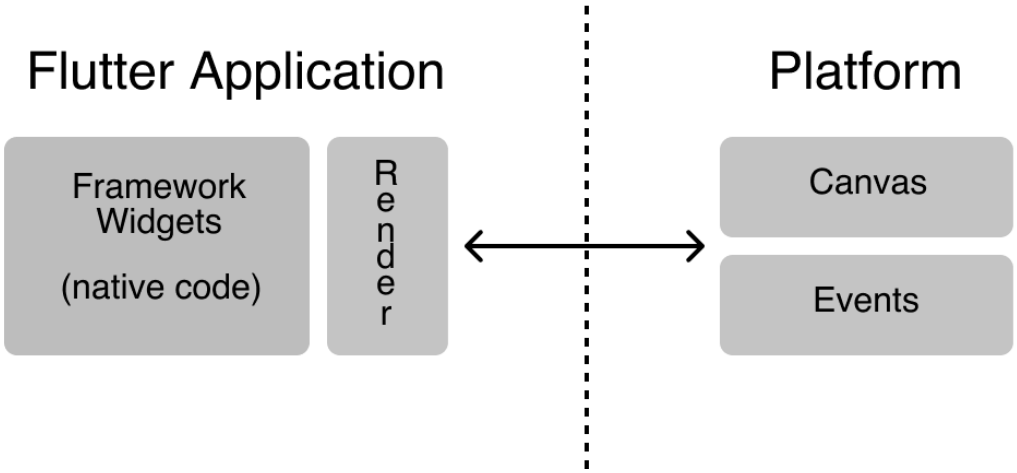

Figura 10: "Flujo de interacción con la UI en Flutter"

Fuente: [20]

En estas imágenes, se aprecian los procesos que siguen tanto React Native como Flutter, lo que demuestra la eficiencia de uno con respecto al otro en cuanto al renderizado.

Cabe resaltar también la interoperabilidad que posee este sistema, ya que si hay aplicaciones desarrolladas en lenguaje nativo y si se requiere agregar funcionalidades con Flutter es posible hacerlo. Sin embargo, no posee la misma efectividad en cuanto a interoperabilidad como lo hace Java y Kotlin, o Swift y Objective-C.

3.3.2) Selección de *framework* para el desarrollo

En cuanto al desarrollo de la aplicación, después de haber seleccionado el tipo, se ve conveniente realizar el desarrollo usando el *framework* Flutter, debido a las características mencionadas en su descripción tales como la

respuesta rápida de las interfaces, la versatilidad para el desarrollo rápido para dos plataformas, el alto rendimiento similar al nativo en IOS y Android, los buenos componentes para desarrollar la UI y su respaldo por Google. Además, cabe mencionar que puede ser rápidamente usado para crear aplicaciones de escritorio y páginas web modificando ligeramente el código fuente.

3.3.3) Selección de Modelo de base de datos

Para decidir sobre la base de datos más adecuada se toma, principalmente, dos pilares para una empresa dedicada al rubro de salud, que su implementación sea de bajo coste y a su vez a lo largo del tiempo se deba realizar un bajo mantenimiento. Para esto se toma como la más conveniente una base de datos NoSQL, dado que suple las necesidades requeridas de forma óptima. Se utilizarán bajo el esquema mostrado en las figuras 10, 11 y 12 .

Figura 11 : "Esquema para los equipos"

Fuente: "Elaboración propia"

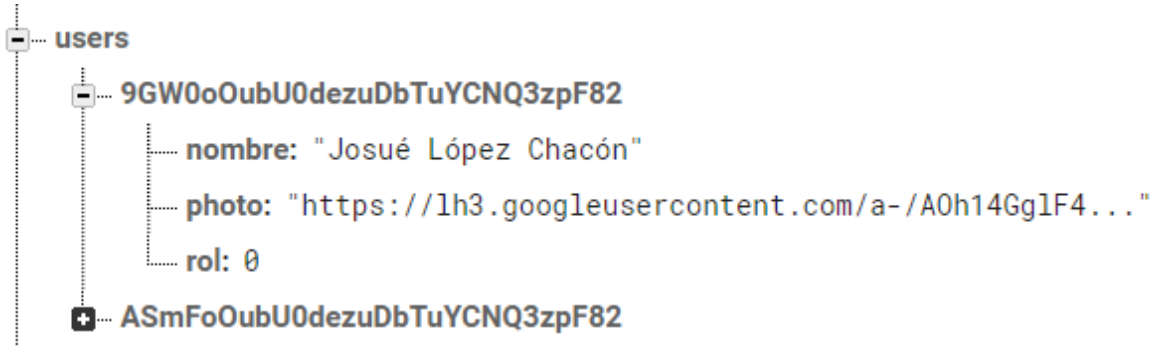

Figura 12: "Esquema para los usuarios"

Fuente: "Elaboración propia"

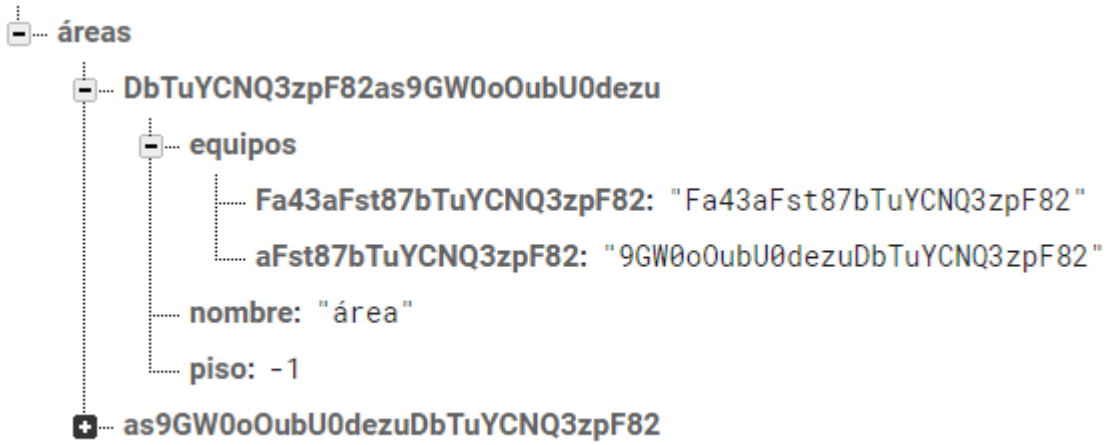

Figura 13: "Esquema para guardar las áreas de la institución

Fuente: "Elaboración propia"

3.3.4) Selección de Arquitectura de software

Dado que se ha escogido Flutter como el *framework* a utilizar para el desarrollo de la aplicación. Sin embargo, aunque se suele recomendar el patrón BLoC, en este caso se utilizará el patrón MVC. Esto es, debido a que el primero agrega una complejidad considerable (aunque para futuras implementaciones es el más apropiado) en tanto el segundo nos permitirá un desarrollo más rápido del aplicativo con un patrón muy reconocido en el mercado al igual que la cantidad de profesionales que dominan este patrón.

3.3.5) Selección de IDE

Aparentemente, la mejor que escoger Android Studio, por su mayor cercanía con el desarrollo de aplicaciones móviles, pero Visual Studio Code, permite la integración de una gran cantidad de *Plugins* que nos brinda diversas características para el desarrollo adecuado en Flutter y realizar un desarrollo más fluido. Además, de ser más ligero por lo que permite un mejor desempeño del computador.

3.3.6) Selección de Servidor *backend*

Como se analizó en la sección anterior, el servidor local posee muchas complicaciones tanto en el mantenimiento como en la renovación de equipos y los costos asociados; a diferencia de un servidor en la nube. Por tanto, se utilizarán diferentes servicios en la nube como se mostró en la arquitectura de la Figura 11. A continuación, se hará una comparativa entre los servicios que ofrecen para las distintas etapas y se escogerán los que se acoplen mejor a la solución. Para sintetizar mejor se realizará un cuadro comparativo entre 3 bases de datos no relacionales que se brindan como servicio por diferentes proveedores.

	Amazon DynamoDB	Firebase Realtime Database	Microsoft Azure Comos DB
Descripción	Servicio de base de datos alojado y escalable de Amazon con los datos almacenados en la nube de Amazon	Almacén de documentos en tiempo real alojado en la nube. Los clientes de iOS, Android y JavaScript comparten una instancia de Realtime Database y reciben automáticamente actualizaciones con los datos más recientes.	Servicio de base de datos multi modelo distribuido globalmente, escalable horizontalmente.
Modelo de base datos primario	Almacenamiento de documentos. Almacenamiento de llave-valor.	Almacenamiento de documentos.	Almacenamiento de documentos. Gráfico DBMS Almacén de llave- valor. Tienda de columna ancha
Puntaje basado en popularidad (sitios web, foros,etc.)	75.20	17.23	36.70
Desarrollador	Amazon	Google	Microsoft
Esquema de datos	Libre de esquema (schema-free)	Libre de esquema (schema-free)	Libre de esquema (schema-free)
API's métodos de acceso.	RESTful HTTP API	Android iOS JavaScript	DocumentDB API Graph API (Gremlin) MongoDB

		API RESTFUL HTTP API	API RESTFUL HTTP API Table API
Durabilidad (soporte para la persistencia de datos)	Sí	Sí	Sí
Soporte de manipulación concurrente de datos	Sí	Sí	Sí
Scripts en el lado del servidor	No	Limitación con el uso de reglas	JavaScript
Triggers	Sí	Se realizan llamadas cuando la data cambia	JavaScript

Figura 14: "Comparativa de Bases de datos como servicio"

Fuente :[21]

A partir de la comparativa en la figura 13, se puede apreciar que presentan varias similitudes en cuanto a sus características. Sin embargo, se destacan dos puntos importantes en realtime database, los cuales son los métodos de acceso que ofrecen y los scripts del lado del servidor. El primero es vital, ya que permite una implementación más rápida en dispositivos móviles, ya que se pueden usar librerías y así mejorar el performance dentro del aplicativo en los dispositivos. El segundo permite, en el caso de realtime database, una rápida implementación de restricciones de acceso en cuanto a lectura y escritura de la base de datos, por tanto nos garantiza mayor seguridad de forma simplificada.

Otro punto importante, para almacenar los documentos (PDF, facturas, imágenes etc.) será necesario usar Firebase Storage, que tiene características similares y nos permite almacenar objetos. Se escoge, respecto a DynamoDB y MicrosoftCosmosDB, por las mismas razones mencionadas en la selección de realtime database.

3.3.7) Selección de código para etiquetado

Evaluando las características y viendo que en dispositivos móviles es mucho más fácil la lectura de los códigos QR que de los de barras, se opta por el primero, además de ofrecer un identificador único para cada equipo, por la mayor capacidad para almacenar información, por la facilidad de lectura y la robustez del código. Este último es importante, dado que estos códigos serán pegados en los equipos médicos y deberán tener una duración considerable, y en caso se dañen debe ser posible aún recuperar los datos con facilidad. Respecto a los tags NFC, son una buena propuesta, pero su inversión inicial cuadruplica a la de los códigos QR, además de que su mantenimiento es más complejo que los QR, ya que habría que escribir nuevamente en el tag y en caso de renovación del tag, habría que reconfigurar cada tag con la información respectiva. Finalmente, los RFID requieren de un lector especial, que le agrega un costo considerable además de la configuración de este dispositivo externo con el aplicativo móvil requeriría otro tiempo importante. Por lo mencionado, la opción más adecuada es el etiquetado mediante códigos QR.

3.4) Desarrollo de la base de datos

Para el desarrollo de la base de datos se consideró que la solución requerirá de tablas que posean la siguiente información:

En esencia la Figura 5 constituye la completa solución para la base de datos para nuestro problema, tal vez, pueda ser que en un futuro se pueda implementar más datos, pero para la solución planteada satisface los requerimientos.

3.5) Flujo Lógico

En esta sección se verá la lógica, a través de diagramas de flujo por los cuales se regirá nuestra aplicación móvil para la implementación de la solución.

3.5.1) Flujo de inicio de sesión

En esta sección se verá el flujo de inicio de sesión, que como se detalló anteriormente será gestionada por un servicio de Firebase, para ello se utiliza como referencia el siguiente gráfico.

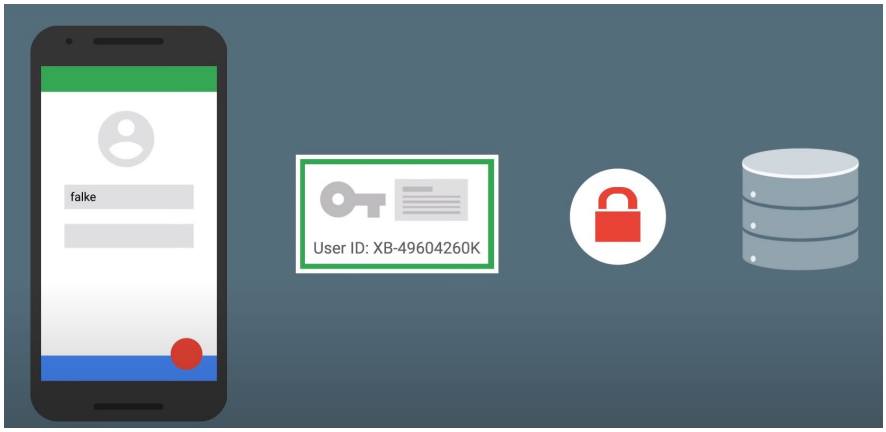

Figura 15: "Ilustración del flujo de inicio de sesión (antes de ingresar credenciales)"
Fuente: [22]

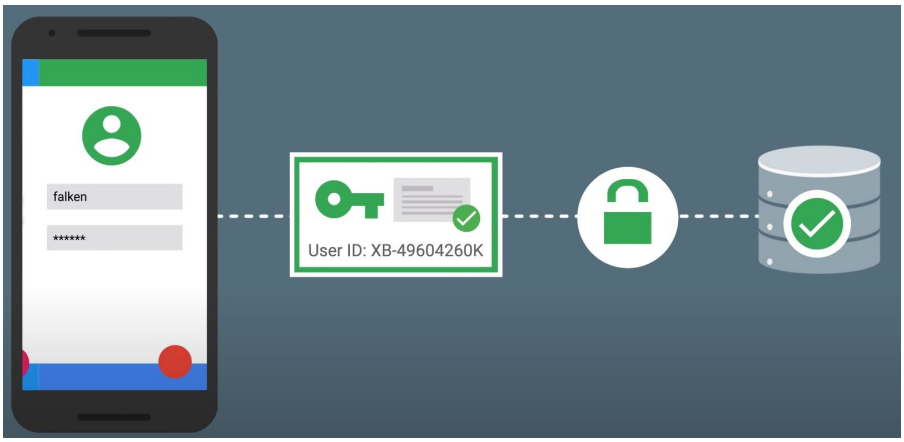

Figura 16: "Ilustración del flujo de inicio de sesión (después de ingresar credenciales)"
Fuente: [22]

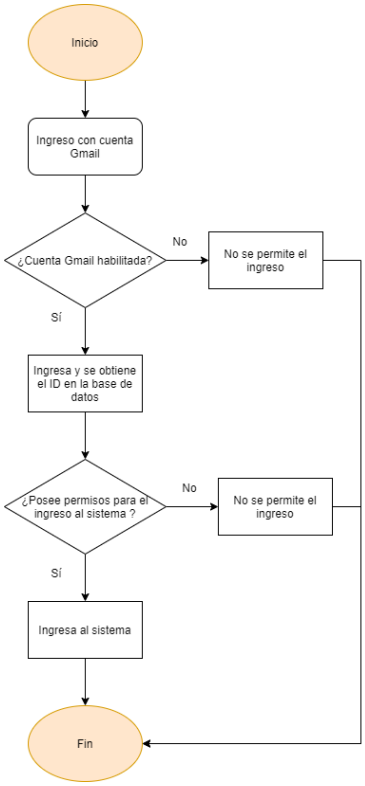

Figura 17: "Flujo de inicio de sesión"
Fuente: "Elaboración propia"

Como se puede apreciar en las figuras 14 , 15 y 16 mostrados anteriormente, la autenticación que ofrece Firebase authentication es de rápida integración y no requiere de muchas configuraciones al poseer librerías fáciles de implementar. Es así que solo se hace una consulta al correo especificado, y se siguen los estándares de la industria como OAuth 2.0 de forma automática como se lee en la documentación oficial en [22]. Bajo este estándar como se puede apreciar en el flujo de la Figura 14, se realiza una solicitud a la entidad federada de la cual queremos requerir los permisos (Google Gmail, Facebook, entre otros), el aplicativo recibe las credenciales y con ellas se hace la solicitud al servidor de autorización que se encargará de verificar que sean válidas. En caso lo sean devuelve el token de acceso que nos permite acceder a los archivos protegidos en el servidor de recursos.

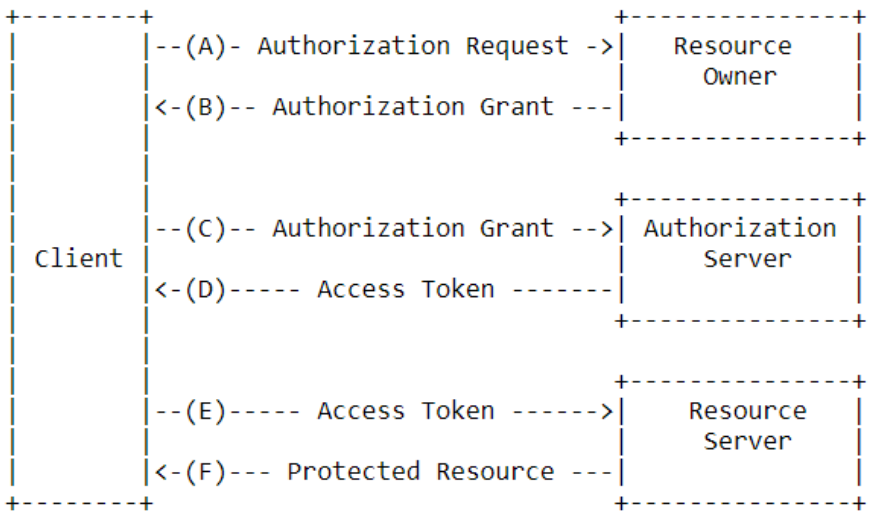

Figura 18: "Flujo del protocolo Oauth2"

Fuente: [23]

Gracias a esto se puede realizar rápidamente la integración con proveedores de identidad federada. En la figura 9, se puede apreciar que, primeramente, la cuenta de Gmail debe estar habilitada., este es un requerimiento propio de Google, por lo que se hará necesario que siempre estén habilitadas las cuentas. Después de este paso, se obtiene el ID que se genera automáticamente, y con este se interactúa con información en la base de datos NoSQL. Posteriormente, se verifica qué usuario está ingresando para brindarle acceso al sistema si es que está autorizado.

3.5.2) Flujo de registro de equipamiento médico

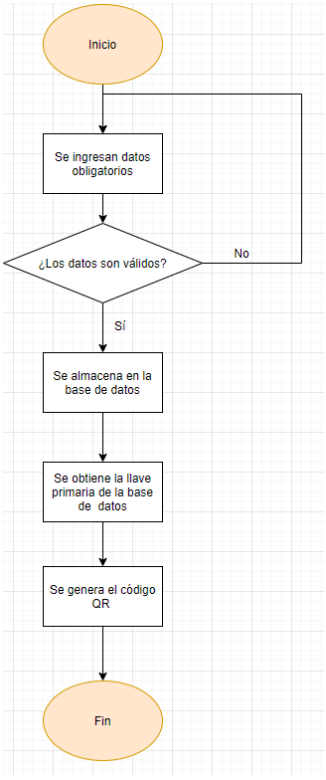

Figura 19: "Flujo de registro de equipamiento"
Fuente: "Elaboración propia"

Como se aprecia en la figura 20, se requerirán los datos obligatorios para el equipo médico, y se validan si estos son correctos, posterior a esto se realiza el almacenamiento en la base de datos. Cuando sucede esto se obtendrá la llave primaria con la cual se generará el código QR para etiquetar el equipo médico. Dicho código será impreso y se adherirá al equipamiento correspondiente.

3.5.3) Flujo de lectura de equipamiento médico

Para la lectura del equipamiento como se puede notar en la, se utilizará el lector de código QR que se implementará en la aplicación. Luego, se realizará la consulta a la base de datos para visualizar los registros almacenados en la aplicación.

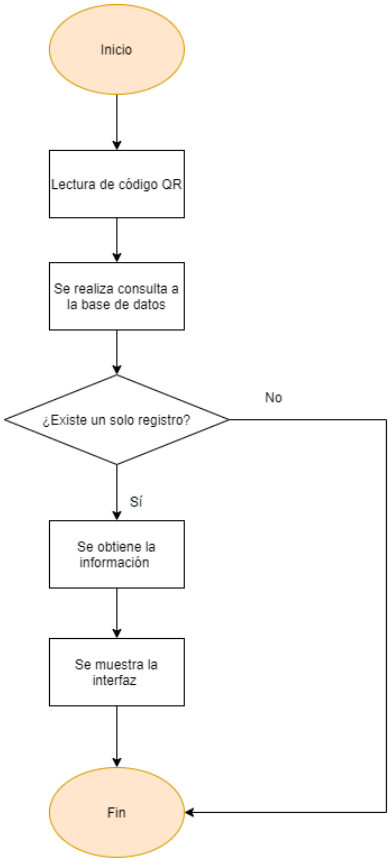

Figura 12: "Lectura de código QR"
Fuente: "Elaboración propia"

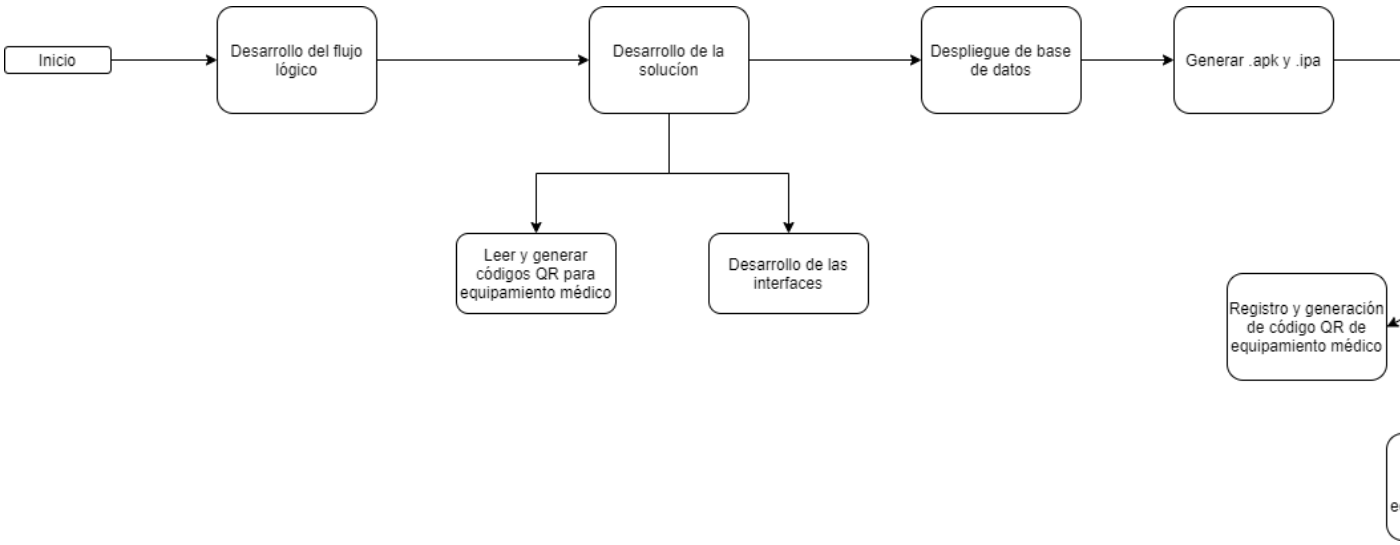

Figura 21: "Diagrama de etapas del proyecto"

Fuente: "Elaboración propia"

3.5.4) Implementación de la base de datos

2.4.3.4) Estructura de la base de datos

Capítulo 4
Pruebas y análisis económico

En el presente capítulo se realizarán las pruebas funcionales y de calidad para garantizar que la solución presentada podrá rendir óptimamente cuando sea puesta en producción.

4.1) Pruebas funcionales

Estás pruebas tienen como propósito verificar que cada una de las etapas contribuye con el flujo de la aplicación y, por tanto, llegará al objetivo esperado.

4.1.1) Pruebas de caja blanca

En esta sección se realizarán las pruebas conociendo las funciones y se podrán los valores más lógicos para corroborar que la aplicación cumple todo el flujo completo.

/* Se realizará las pruebas finales cuando la aplicación esté desplegada*/

4.1.2) Pruebas de caja blanca

En esta sección se pondrán valores extremos con el fin de verificar los límites que permite la aplicación, además, de verificar que exige la modificación de los datos para completar el flujo de los datos.

/* Se realizará las pruebas finales cuando la aplicación esté desplegada*/

4.2) Pruebas de calidad

En esta sección se evaluará si la aplicación cumple con los estándares mínimos para proporcionar disponibilidad del servicio y si bajo pruebas de estrés rinde adecuadamente.

4.2.1) Disponibilidad de servicio

Al ser un proveedor de servicios, se evalúa el SLA (*Service Level Agreement*). En este caso Firebase ofrece que sus servicios al menos tendrán el 99.95% de disponibilidad según [24] . Esto sería suficiente, ya que, al no ser una aplicación crítica, podría realizarse una solicitud en pocos minutos o segundos y el servicio, muy probablemente, responda adecuadamente.

4.2.2) Pruebas de acceso simultáneo

Según la documentación de Firebase en [25], se puede apreciar que soporta 200 000 conexiones simultáneas y que en caso se requiera de más se podrían crear más bases de datos. Las respuestas en simultáneo que se pueden tener desde la base de datos son de 100 000 por segundo. Por lo tanto, considerando que la institución pública, como se puede ver en el documento de personal en su página web [26], posee alrededor de 1800 trabajadores, los cuales son muy inferiores a los límites que nos ofrece FIrebase, por tanto la solución, naturalmente, cumple con nuestras necesidades.

4.3) Evaluación económica

4.3.1) Costo de inversión

Considerando que un tesista en el mercado laboral inicia con un salario de 3200, por lo que nos resulta el pago por hora en 20 soles.

Etapa	Rol	Responsable	Días	Hora/ día	Horas totales	Precio por hora	Subtotal (S/)
Diseño de la solución TI	Consultor	Asesor	5	2	10	S/.50,00	S/.500,00
	Desarrollador	Tesista	7	5	35	S/.20,00	S/.700,00
Desarrollo de aplicativo móvil	Consultor	Asesor	7	3	21	S/.50,00	S/.1.050,00
	Desarrollador	Tesista	30	6	180	S/.20,00	S/.3.600,00
	Diseñador	Tesista	3	6	18	S/.20,00	S/.360,00
	Tester	Tesista	10	6	60	S/.20,00	S/.1.200,00
Despliegue e Integración de aplicativo con Servicios en la nube	Consultor	Asesor	3	5	15	S/.50,00	S/.750,00
	Desarrollador	Tesista	4	6	24	S/.20,00	S/.480,00
	Tester	Tesista	5	5	25	S/.20,00	S/.500,00
Estructuración de base de datos NoSQL	Desarrollador	Tesista	6	4	24	S/.20,00	S/.480,00
						Total	S/.9.620,00

4.3.2) Costos de mantenimiento

Para el uso de los diferentes servicios se evaluará en base a estimaciones de cantidad de usuarios que se esperan tener.

4.3.2.1) Costos de mantenimiento

Conclusiones

1. El aplicativo móvil contribuye con la digitalización en cuanto a la gestión del equipamiento médico de forma efectiva, capturando todas las características del equipo involucrado, generando alertas de mantenimiento, registro de la ubicación y etiquetado respectivo.
2. El costo de implementación del aplicativo y los servicios en nube que utiliza son comparativamente menores respecto a soluciones tradicionales y ofrecen un rendimiento óptimo, y en algunos aspectos es superior.
3. El *backend* posee una seguridad alta al dado que se vincula con el servicio de autenticación del proveedor y, a su vez, puede restringir a los usuarios de la misma institución según el dominio en el correo.
4. El etiquetado utilizado es eficiente ya que ante ciertos daños o deterioro con el tiempo puede aún recuperarse la información, debido a que tiene códigos de corrección de errores.

Referencias

[1] "Testimonios y experiencias de los principales líderes empresariales del Perú."

[2] A. Phongtraychack and D. Dolgaya, "EVOLUTION OF MOBILE APPLICATIONS," doi: 10.1051/matecconf/201815501027.

[3] "EB139/8 2." Accessed: Nov. 05, 2020. [Online]. Available: http://www.who.int/goe/policies/en.

[4] OMS, *Recursos humanos para gestión de dispositivos médicos*. .

[5] S. Lucho, "Clase 13 - AWS - Lambda & API Gateway - Presentaciones de Google." 2020, Accessed: Jan. 03, 2021. [Online]. Available: https://docs.google.com/presentation/d/1bZCwHR8_nT2fV4Y8YwyeFvq8WnIH8NII3-zla67biRk/edit?usp=sharing.

[6] "» Organigrama InstitucionalInstituto Nacional de Enfermedades Neoplásicas." https://portal.inen.sld.pe/organigrama/ (accessed Dec. 17, 2020).

[7] "About HarmonyOS." https://developer.harmonyos.com/en/docs/documentation/doc-guides/harmonyos-overview-0000000000011903 (accessed Jan. 03, 2021).

[8] "HUAWEI AppGallery-AppGallery Review Guidelines." https://developer.huawei.com/consumer/en/doc/30202 (accessed Jan. 03, 2021).

[9] "MySQL :: MySQL 8.0 Reference Manual :: 1.2.2 The Main Features of MySQL." https://dev.mysql.com/doc/refman/8.0/en/features.html (accessed Dec. 17, 2020).

[10] C. Alberto Balcázar Wong, "DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA PARA LA GESTIÓN DE INDICADORES DE CALIDAD EN TELEFONÍA MÓVIL," Pontificia Universidad Católica del Perú, 2015.

[11] "Bases de datos no relacionales | Bases de datos de gráficos | AWS." https://aws.amazon.com/es/nosql/ (accessed Dec. 16, 2020).

[12] "Patrones Arquitectónicos en Android | by Luis Vespa | Medium." https://medium.com/@vespasoft/patrones-arquitectónicos-en-android-ded39f7a2c10 (accessed Dec. 23, 2020).

[13] "Introducción a Android Studio | Desarrolladores de Android."

https://developer.android.com/studio/intro?hl=es-419 (accessed Dec. 26, 2020).
[14] "Get Started with Visual Studio Code." https://code.visualstudio.com/learn (accessed Dec. 26, 2020).
[15] "Codigo de barras – Simbologias – CODIGO DE BARRAS.PE." https://codigodebarras.pe/codigo-de-barras-simbologias/ (accessed Jan. 03, 2021).
[16] "What is the NFC technology | Dipole RFID." https://www.dipolerfid.com/rfid-blog/whats-is-nfc (accessed Jan. 18, 2023).
[17] "QR Codes Vs. Barcodes: Which is best for asset tracking and inventory?" https://itemit.com/qr-codes-vs-barcodes-which-is-best/ (accessed Jan. 03, 2021).
[18] J. Muppala, "Hybrid Mobile App Development | Coursera." https://www.coursera.org/learn/react-native/lecture/WoAoo/hybrid-mobile-app-development (accessed Dec. 15, 2020).
[19] "Introduction to widgets - Flutter." https://flutter.dev/docs/development/ui/widgets-intro (accessed Dec. 26, 2020).
[20] "Flutter: Google's take on cross platform | CSS-Tricks." https://css-tricks.com/flutter-googles-take-on-cross-platform/ (accessed Dec. 26, 2020).
[21] DB-Engines, "Amazon DynamoDB vs. Firebase Realtime Database vs. Microsoft Azure Cosmos DB Comparison." https://db-engines.com/en/system/Amazon+DynamoDB%3BFirebase+Realtime+Database%3BMicrosoft+Azure+Cosmos+DB (accessed Jul. 20, 2021).
[22] "Firebase Authentication." https://firebase.google.com/docs/auth?hl=es-419 (accessed Jan. 03, 2021).
[23] E. Hardt, "The OAuth 2.0 Authorization Framework," *Octubre 2012*. https://datatracker.ietf.org/doc/html/rfc6749 (accessed Jul. 01, 2021).
[24] "Cloud Storage para Firebase." https://firebase.google.com/docs/storage (accessed Jul. 20, 2021).
[25] "Límites de Realtime Database | Firebase Realtime Database." https://firebase.google.com/docs/database/usage/limits?hl=es-419 (accessed Jul. 21, 2021).
[26] "Nómina de personal Activo a Junio 2021." https://portal.inen.sld.pe/wp-

content/uploads/2021/07/NOMBRADO.pdf (accessed Jul. 21, 2021).
[1] “Perú Avances en digitalización”, 2017.
[2] “IDC - Smartphone Market Share - OS”. https://www.idc.com/promo/smartphone-market-share/os (consultado jul. 09, 2020).