Diseño de una aplicación móvil que permita la gestión oportuna de dispositivos biomédicos en el laboratorio de simulación clínica

Emmanuel Zambrano Quitián

Trabajo dirigido por:

Tutor

Msc. Jefferson Sarmiento Rojas

UNIVERSIDAD DEL ROSARIO

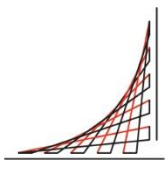
ESCUELA COLOMBIANA DE INGENIERÍA JULIO GARAVITO

PROGRAMA DE INGENIERÍA BIOMÉDICA

BOGOTÁ D.C

2020

Universidad del Rosario



ESCUELA COLOMBIANA DE INGENIERÍA JULIO GARAVITO

Agradecimientos:
Inicialmente vi este apartado como algo puntual, agradecer a aquellas personas que permitieron que el “Diseño de una aplicación móvil que permita la gestión oportuna de dispositivos biomédicos en el laboratorio de simulación clínica” se hiciera posible. Ahora me doy cuenta de lo equivocado que estaba, pese a que sí, este apartado debe cumplir con esa función también pude ser una pequeña despedida de este gran camino que fue estudiar Ingeniería Biomédica, un agradecimiento a todos aquello que hicieron posible el hecho de que llegase al punto de poder entregar este trabajo de grado.

Hay que empezar por el principio, quiero agradecer a Ana María Presiga, por ser, por mucho la mejor jefe que he tenido, por haber confiado en mí y haberme abierto las puertas en esta mi primera experiencia laboral como biomédico, por tenerme la paciencia que aquellos que me conozcan saben que se requiere para poder trabajar junto a mí.

También debo agradecer a Diana Catalina Rodríguez y a los doctores del laboratorio de simulación clínica, por haberme aguantado y explicado cada vez que iba y me ponía como un niño pequeño a preguntar “¿Qué es esto?; ¿Cómo funciona aquello?”.

A mis padres por haber confiado en mí, pese al complicado camino que tuve en mi proceso de educación, por ser los responsables directos de este logro y a mi hermana que siempre le dije que no le iba a agradecer.

Durante la carrera tuve muchos buenos profesores, pero el profesor Luis Alejandro Fonseca y la profesora Nora Yamile, ellos me enseñaron a amar y respetar las matemáticas que sin duda es mi mayor pación, me dieron la habilidad de demostrar la cual es una insignia en mí. Estoy seguro de que mi proceso educativo abría sido muy diferente, mucho más largo y difícil sin ellos, muchos profesores te pueden enseñar, pocos pueden hacerte amar las matemáticas de la manera en que yo lo hago, mi agradecimiento tiende al infinito.

Finalmente, la universidad no habría sido la mejor etapa de mi vida sin las personas que me acompañaron en ella. Con el permiso de los de más primero esta Jess, el amor de mi vida, así como en primer semestre me enamoré de las matemáticas, en tercero descubrí el amor, mi primera y única novia, una persona que me mostró un mundo totalmente diferente.

Además de Jess mi familia se hizo un poco más grande, con increíbles amistades como la de Laura, Mateo, Tenjo, Alejandra, Checho y Carolina. Estas amistades me vieron crecer, reír y llorar, vieron como un niño de 17 años se convirtió en ingeniero de 22, con ellos aprendí el valor de las amistades, pese a que algunas llegaron en primer semestre y otra en los últimos, algunas terminaron y otras serán eternas. Siempre atesoraré las historias, salidas y viajes. Junto a ustedes madure y les debo agradecer la persona que soy hoy.

Finalmente agradezco a Rafael y a Cruz mis dos mejores amigos fuera de este mundo universitario, siempre me recibieron con una buena charla cada vez que necesitaba una escapada.

Contenido

Lista de imágenes

Lista de tablas

Resumen:

Introducción: El propósito de este documento es diseñar una aplicación móvil que permita consultar la base de datos de los equipos biomédicos del laboratorio de simulación clínica de la Escuela de Medicina y Ciencias de la Salud de la Universidad del Rosario. Además, la aplicación se diseñó para permitir la consulta de guías de usuario de simuladores como SimMom 3 [1] personalizados para las necesidades de los docentes, así como una explicación profunda en los modos de uso más comunes y errores frecuentes de este modo.

Objetivo general: Diseñar una aplicación móvil que permita la gestión oportuna de dispositivos biomédicos al personal asistencial del laboratorio de simulación clínica

Metodología: La gestión del proyecto se basó en la metodología Agile, la cual permitía la flexibilidad de trabajar directamente con los profesionales del laboratorio de simulación clínica, haciendo entregas constantes y recibiendo retroalimentación de manera instantánea. Además de esto, como parte de la implementación de la metodología se hizo una extensa investigación de las posibles arquitecturas de software para que la aplicación tuviese el mejor rendimiento posible en cuanto a su funcionamiento y gestión de datos, también se investigó las opciones de almacenamiento y administración de datos.

Resultados: Se desarrollaron dos aplicaciones de consulta, cada una con una opción de almacenamiento y arquitectura diferente. Se comparó el rendimiento de cada una y la percepción que tenían los usuarios (docentes e ingenieros del laboratorio de simulación clínica). Se evaluaron los aspectos más subjetivos como estética, facilidad de uso y relevancia de la información suministrada, tanto de la aplicación como de las guías y aspectos técnicos de la aplicación, como la facilidad de actualizar los datos y rendimiento.

Palabras clave: base de datos, interfaz gráfica, arquitectura de software.

1. INTRODUCCIÓN

La Universidad del Rosario es una universidad privada, oficialmente nombrada desde su fundación, en el año 1653, como “Colegio Mayor de Nuestra Señora del Rosario” por su fundador el arzobispo de Santa Fe de Bogotá, fray Cristóbal de Torres y Motones. Esta institución nació con un carácter laico, cuya ideología y funcionamiento dependía de la Corona española.

A partir de su fundación y hasta el momento, la Universidad del Rosario ha ido evolucionando hasta convertirse en lo que es hoy, una institución sin ánimo de lucro, privada y autónoma, conformada por una estructura organizacional encabezada por la rectoría seguida de diferentes órganos directivos como la Vicerrectoría, la Sindicatura y la Secretaría General.

Está constituida por cinco facultades y tres escuelas, a saber: Economía, Jurisprudencia, Ciencia Política y Gobierno, Relaciones Internacionales, Ciencias Naturales y Matemáticas, Escuela de Medicina y Ciencias de la Salud, Escuela de Ciencias Humanas y Escuela de Administración.[2]

El practicante de ingeniería biomédica en la Universidad del Rosario ingresó al departamento de ambiente. Este departamento desempeña las tareas propias de un departamento de ingeniería y realiza la vigilancia de la central eléctrica y de gases. Respecto a los equipos, realiza la gestión para la adquisición nuevas tecnologías, manejo de inventarios y maneja la tercerización de los mantenimientos de los equipos.

Las labores del practicante se enfocaron en recibir capacitaciones para los nuevos equipos adquiridos durante el periodo 2020-1. Después de recibir las instrucciones de uso y mantenimiento, su labor fue recolectar toda la información de uso, partes, configuraciones e instrucciones de mantenimiento y consignarlos en un manual de usuario, el cual debía ser fácil de entender por parte del personal del laboratorio, tanto doctores como auxiliares e ingenieros.

Durante el periodo de 2020-1 se identificó que el laboratorio de simulación clínica tenía una gran dificultad en el manejo apropiado de los simuladores de paciente. Esta situación se hizo evidente cuando la ingeniera del laboratorio estuvo incapacitada por una semana y el resto del personal tuvo dificultades para manipular los simuladores de paciente.

La dificultad se debió a que estos equipos tienen diferentes modalidades de uso, un gran inventario de piezas intercambiables y diferentes configuraciones, las cuales permiten simular diferentes casos clínicos. Es importante mencionar que una mala manipulación o error a la hora de cambiar la configuración puede generar un cese de las actividades de manera temporal, un daño en el simulador o un desgaste acelerado. Es importante mencionar que el alto costo de los simuladores, cualquier mantenimiento o el cambio total del equipo es un gasto que la institución no puede tomar el riesgo de asumir. Por estos motivos el personal que los opere debe tener un amplio conociendo sobre su desempeño y uso.

Para dar apoyo a esta situación se propuso diseñar una plataforma online con conexión a una app en la cual los usuarios de los simuladores de paciente puedan consultar las guías rápidas y recuerden las instrucciones dadas durante la capacitación, las preguntas frecuentes y las recomendaciones dadas por el fabricante. Además, en la plataforma podrán

encontrar información adicional como la hoja de vida de los equipos y agregar anotaciones como el estado del equipo (si presentó alguna falla), lo cual facilitará la comunicación entre el departamento de ingeniería y el laboratorio. Esta solución presenta una ventaja sobre el método actual de las guías y hojas de vida físicas ya que permite agrupar toda la información relevante del equipo en un solo lugar, además de que pone a la mano y de manera instantánea estos datos. A diferencia del método tradicional las guías tienden a deteriorarse y, en el peor de los casos, a perderse y cuando un fallo o inquietud se presenta es difícil o requiere más tiempo obtener este tipo de información.

Iniciativas como esta son una menester del departamento de ingeniería clínica ya que este busca brindar un servicio a las entidades promotoras de salud (EPS) e Institutos prestadores de salud (IPS) el cual permita gestionar la adquisición de dispositivos y equipos biomédicos[1][cap 12, pp1]. Dentro de lo que se incluye el análisis de necesidades de la institución, estado de la tecnología que se posee, relación costo beneficio y planes de capacitación del personal encargado del uso del dispositivo[3].

Otro enfoque del departamento de ingeniería clínica es brindarle diferentes tipos de mantenimiento a los equipos y simuladores, dentro de los cuales el Invima identifica tres tipos:

-Mantenimiento preventivo: “Mantenimiento que se realiza para prolongar la vida útil del dispositivo y prevenir desperfectos.

-Mantenimiento correctivo: “Proceso para restaurar la integridad, la seguridad o el funcionamiento de un dispositivo después de una avería”, este tipo de mantenimiento se hace a los equipos únicamente cuando presentan una falla o comportamiento anormal.[4]

-Mantenimiento predictivo: “Técnica para prever la frecuencia de avería de determinados tipos de componentes sustituibles” el cual se realiza cuando se espera que un insumo reemplazable esté cerca de fallar. El reemplazo se hace antes de que se dé la falla, de allí previene su nombre.[4]

Según la FDA la mayor causa de fallos en un equipo biomédico se debe a la mala manipulación de este, fallos generados por el desconocimiento de los dispositivos, sus accesorios y formas de operación. Se puede afirmar que una de las principales dificultades del área de mantenimiento son los llamados, lo cual hace referencia a la acción de solicitar soporte al departamento de ingeniería clínica por el mal comportamiento de un equipo, por un presunto daño o “desconfiguración” falsos. Esto quiere decir que se llama al departamento porque se presume que un equipo esta dañado, pero realmente no lo está, el “fallo” se debe a algún tipo de descuido o error como por ejemplo que el equipo no está conectado. Además de esto, otro de los mayores inconvenientes del departamento son los

[1] Dispositivo médico para uso humano: Se entiende por dispositivo médico para uso humano, cualquier instrumento, aparato, máquina, software, equipo biomédico u otro artículo similar o relacionado, utilizado sólo o en combinación, incluyendo sus componentes, partes, accesorios y programas informáticos que intervengan en su correcta aplicación, propuesta por el fabricante para su uso en: Diagnóstico, prevención, supervisión, tratamiento o alivio de una enfermedad, lesión o de una deficiencia. Investigación, sustitución, modificación o soporte de la estructura anatómica o de un proceso fisiológico. [13]

ocasionados por la mala manipulación del equipo, estos fallos se podrían evitar si los usuarios tuviesen un mayor conocimiento de la buena operación del equipo.

Por escenarios como los mencionados previamente (malas manipulaciones) se podía inferir que los fallos generados por el usuario son causados porque no tuvieron una capacitación y no poseen los conocimientos básicos necesarios para operar el equipo. Sin embargo, según el artículo 53 de la Constitución Política de Colombia y el artículo 24 del decreto 2425 de 2005, una empresa debe dar una capacitación pertinente a todo miembro del personal que lo requiera. Por lo tanto, se infiere que los fallos provocados por el mal uso no se deben adjudicar a la falta de una capacitación sino a que esta no fue dada en un momento oportuno. [5]

Por lo general las capacitaciones se brindan a todo el personal de la entidad que posiblemente vaya a manipular el equipo, sin embargo, se debe tener en cuenta que no todos los que reciben la capacitación están trabajando con el equipo en el que fueron capacitados, solo después de un par de rotaciones llegarán a trabajar con él. En el manual de la ingeniería clínica se da una claridad de porqué situaciones como estas dan malos resultados, por medio de la definición y diferencias de educar y entrenar[6].

“Educar genera una competencia general en un campo o tema, adoptando conocimiento o sabiduría”, esta definición se acomoda para la primera parte de la capacitación donde todos los involucrados reciben los conocimientos necesarios para poder trabajar con el dispositivo.[6]

“Entrenar ejercita a alguien en una profesión o lo direcciona a obtener habilidad. Entrenar incluye dar indicaciones y obtener hábitos de pensamiento o acción. El entrenamiento es usado para moldear o desarrollar características a través de la disciplina”. [6]. Comprendiendo las definiciones de educar (entendiendo una capacitación como el proceso por el cual se educa a un usuario en la manipulación de un equipo o simulador) y entrenar se pueden evidenciar las falencias del sistema y provocar que los usuarios de los equipos biomédicos presentan dificultad a la hora de manipular los dispositivos, todo son educados en el equipo y reciben el conocimiento apropiado, pero como no están en constante contacto con el mismo no pueden obtener el entrenamiento apropiado, no desarrollan las habilidades para operarlo y no refuerzan el aprendizaje haciendo que sea fácil de olvidar con el paso del tempo.

2. Objetivos

2.1. Objetivo general

Diseñar una aplicación móvil que permita la gestión oportuna de dispositivos biomédicos al personal asistencial del laboratorio de simulación clínica.

2.2. Objetivos específicos

1. Identificar la mejor arquitectura para que múltiples usuarios puedan consultar una base de datos cambiante.
2. Dar una solución al almacenamiento y manejo de datos de los equipos, haciendo que todos los usuarios tengan acceso a ellos.
3. Establecer cada una de las guías rápidas de los equipos que se encuentren en el laboratorio de simulación clínica de la Universidad del Rosario

3. METODOLOGÍA

En este apartado podrá encontrar el conjunto de métodos usados para el desarrollo del proyecto "Diseño de una aplicación de consulta de la base de datos del laboratorio de simulación clínica la cual está en constante actualización para múltiples usuarios" los planteamientos de la arquitectura del sistema, las opciones de plataformas y los criterios usados para decidir cuáles fueron las mejores herramientas para su creación, las elecciones de diseño y recomendaciones aportadas por el personal profesional (médicos e ingenieros) del laboratorio de simulación clínica.

3.1. Planteamiento del problema

Durante el periodo 2020-1 se pudo identificar la necesidad de generar manuales precisos y claros que puedan ser consultados de manera fácil y rápida por los profesionales del laboratorio de simulación clínica. Esto se debe a que pese a que todo el personal del laboratorio es instruido en el manejo y uso de los simuladores, el personal médico se enfoca en el uso práctico y cómo operarlos durante la clase, en cambio el personal de ingeniería, se centra en aprender cómo configurar, limpiar y dar un mantenimiento básico al equipo o simulador.

Sin embargo en un periodo de ausencia de la ingeniera se evidenció el desconocimiento y dificultad de los profesionales de la salud al hacer configuraciones más avanzadas o la solución de errores frecuentes esto, debido a que el personal de ingeniería es el que tiene un mayor entrenamiento en esta parte de la gestión. Esto generó una preocupación en el personal porque corrían el riesgo de dañar o desconfigurar los simuladores, produciendo un retraso y dificultades a la hora de impartir las clases.

El laboratorio de simulación clínica es un espacio muy importante, el cual recibe alrededor de 1000 estudiantes de la escuela de medicina de la Universidad del Rosario todas las semanas y de todos los semestres imprevistos como la ausencia del miembro de ingeniería no puede permitirse un cese o retraso de sus actividades. Todo simulador tiene un manual de uso el cual es proveído por los fabricantes, sin embargo, estos no siempre tiene toda la información necesaria, ya que las indicaciones y recomendaciones se presentan de manera general. Para poder solucionar alguno de los problemas frecuentes es necesario recurrir al manual de ingeniero o a las recomendaciones del cuerpo técnico que se dieron durante la capacitación. Se trata de información puntual, la cual normalmente es difícil de encontrar en poco tiempo. Además, en los simuladores más antiguos es probable que ya se haya liberado una nueva versión del software y hardware el cual no presentará los mismos errores y las guías pueden diferir con respecto al modelo de la institución, esto hace que el proceso de buscar la información se dificulte e incluso genere más errores al aplicar una recomendación del nuevo modelo.

En primera instancia se planteó hacer manuales "personalizados", los cuales se centren en las recomendaciones dadas durante la capacitación, las necesidades particulares de la institución, como dar prioridad a los modos de uso más frecuentes por los docentes y los errores más comunes percibidos por el personal de ingeniería.

Sin embargo, esta solución no era la más adecuada debido a que solo resuelve una parte del problema, brindar la información de forma precisa teniendo en cuenta las necesidades de la institución, pero no contempla la velocidad y facilidad de obtener dicha

información, ya que con estas guías físicas lo más probable es que sean archivadas por el personal de ingeniería. Esto provocaría que cuando se presente nuevamente la necesidad a solucionar (la ausencia inesperada del personal) no se pueda acceder a ellas ya que solo los ingenieros saben dónde se almacenan, haciendo que los doctores no puedan consultarlas fácilmente.

Debido a esto se plantea una opción digital, una aplicación la cual se puede descargar en el teléfono, basada en un interfaz gráfico de usuario (GUI, por sus siglas en ingles graphical user interface), fácil de usar, con la cual se pueda acceder a la guía personalizada (una guía basada en la primera solución planteada) del equipo e información relevante como la hoja de vida del equipo y un apartado de defectos o daños.

Fases del proyecto

3.2. Metodología Agile:

Para poder llevar el desarrollo este proyecto se implementó la metodología de Agile, esta se desarrolló en el año 2001 por el ingeniero de sistemas Kent Beck el cual la expuso en su libro eXtreme Programming. Agile inicialmente fue creada para poder dirigir proyectos de desarrollo de software, los cuales cambiaban rápidamente según las necesidades del cliente, se buscaba una metodología rápida y ágil, la cual se pudiera adaptar durante el desarrollo del proyecto[7]. En años recientes esta metodología ha sido implementada por grandes empresas fuera de ámbito de las ciencias informáticas como el banco BBVA.[8]

Ventajas de la metodología Agile:

1) Mejora de la calidad: reduce los errores durante los entregables y permite que el proyecto final sea más cercano a la visión del cliente.

2) Mayor compromiso: mejora la satisfacción del cliente.

3) Rapidez: al tener entregables en pequeños periodos de tiempo acorta los ciclos de producción y minimiza los tiempos de toma de decisiones.

4) Aumento de productividad: dado que es un modelo flexible, se pueden redistribuir los recursos según las necesidades del proyecto, esto genera que la productividad aumente. [8]

El método Agile se basa en doce principios de los cuales los que más destacan son los siguientes:

1) La principal prioridad es satisfacer al cliente a través de la entrega temprana y continua de Software con valor.
2) Aceptar que los requisitos cambien, incluso en etapas tardías del desarrollo, los procesos Agile aprovechan el cambio para proporcionar ventajas al cliente.
3) Los clientes y desarrolladores trabajan juntos de forma cotidiana durante el proyecto.

4) El método más eficiente y efectivo para comunicar entre el equipo de desarrollo y el cliente es la comunicación cara a cara.
5) El software funcionando es la medida principal de progreso.

3.3. Pasos para implementar el método Agile**:**

Kent Beck explica en su libro eXtreme Programming, que para poder aplicar la metodología Agile, se debe seguir los siguientes pasos

Diagnóstico: en esta primera fase se hace el análisis de necesidades, el planteamiento de los objetivos y se identifican las principales necesidades a solucionar del cliente.

Diseño: en esta fase se hace la recopilación bibliográfica y documentación, se reúnen las posibles opciones de solución, en el caso de diseño de software se plantean las posibles arquitecturas y se decide cual se va a implementar.

Desarrollo: en esta fase se hace la unión de los grupos, mezclando desarrolladores y el cliente, se generan entregables con cortos periodos, en los cuales se hacen la correcciones y se generan los cambios a necesidad.

Esta etapa se subdivide en cuatro etapas, las cuales van orientadas a la elección de arquitectura, diseño y testeo.

Aseguramiento de calidad: con base en las fases funcionales del software, se identifica si la aplicación sí ha cumplido con las necesidades.

Despliegue: incluye todas las acciones post venta.

Ilustración 1 Metodología Agile

3.4. Justificación de la metodología

La aplicación desarrollada debía ser una herramienta usada principalmente por profesionales de la salud, cuando no tengan apoyo del personal de ingeniería. Generalmente en condiciones poco favorables, como incapacidad o emergencia.

Por lo tanto, la interfaz gráfica debe ser clara, con pocos botones e interacciones, los datos suministrados por el usuario deben ser mínimos para evitar errores y agilizar las consultas, fácil de manejar e intuitiva, las elecciones y apartados deben ser lo suficientemente directos, sin configuraciones o ajustes más allá de suministrar la placa y consultar la información deseada.

Además del diseño de la aplicación, las guías de usuario, la cuales son la consulta principal del personal, deben contener información relevante, relacionada con las necesidades, usos comunes de los simuladores y dispositivos en clase y los errores más frecuentes que se presentan al manipularlos. La única manera de obtener esta información es con la interacción directa con el personal del laboratorio de simulación clínica.

Este es un proyecto que requiere una metodología flexible, la cual permita hacer actualizaciones y cambios sobre la entrega final durante el proceso de desarrollo. La metodología Agile, permite tener esa versatilidad, además de que fue desarrollada principalmente para la creación de software.

3.5. Aplicación de la metodología

Para poder aplicar la metodología Agile se elaboró un cronograma con entregas y revisiones constantes durante el desarrollo. Dado que en el proyecto se desea tener control sobre la creación de dos entregables diferentes, las guías de usuario y la aplicación de consulta, se desarrollaron las siguientes estrategias de la metodología Agile, la cuales permitieran la participación del cuerpo del laboratorio de simulación clínica y la interacción cara a cara como lo indica uno de los principios de la metodología.

Para desarrollar este proyecto se siguió el cronograma mostrado en el anexo, el cual se basa en un diagrama de gantt

3.5.1 Diagnóstico, de las condiciones del laboratorio de simulación clínica

Como se analizó en el planteamiento del problema, el principal usuario va a ser el cuerpo médico cuando no tengan soporte del equipo de ingenieros que los acompañan, por esto la interfaz debe ser clara y fácil de manejar.

Además de esto, debe identificar los principales problemas y dificultades de los profesionales de la salud (docentes) a la hora de manipular los dispositivos y simuladores, para esto es necesario recolectar la mayor cantidad de información proveniente del cuerpo de ingenieros ya que son estos los que tiene la experiencia y saben cuáles son estos puntos claves.

Dado a que en el laboratorio de simulación clínica los equipos están en constante movimiento y actualización y continuamente se traen más equipos ya existentes, como en el caso de los monitores multiparámetros, los cuales se adquieren en función de la necesidad o cuando alguno se daña, esto genera la necesidad de que se actualice la base de datos de los equipos adicionando la placa del nuevo dispositivo al registro.

Adicionalmente, también se adquieren nuevos equipos o simuladores únicos los, cuales exigirán una actualización no solo en el listado de equipos sino también en los manuales.

Es por esto por lo que se observa una problemática en cómo se almacenan los datos ya que estos son muy cambiantes y al tenerlos de manera local generaría la necesidad de tener que actualizar la aplicación con cada compra, aún si se trata de una actualización menor, como la adquisición de un nuevo equipo el cual tenga una guía asignada, como es el ejemplo de los monitores multiparámetro.

3.6. Diseño:

Creación de guías de usuario:

1) El proceso inicia con la primera capacitación, donde un ingeniero especializado describe y muestra las diferentes configuraciones de los simuladores, acá se recopila toda la información relevante a las recomendaciones de fábrica y fallas comunes que ellos hayan percibido.
2) La práctica: junto al personal de ingeniería del laboratorio de simulación clínica, se realizan las configuraciones y prácticas mostradas durante las capacitaciones, todo con el objetivo de identificar dificultades o puntos de atención a la hora de cambiar los modos de operación y hacer configuraciones puntuales.
3) Capacitación médica: por lo general, posteriormente a la capacitación técnica, se da una demostración menos técnica, enfocada en las configuraciones de clase, recolección del performance de los estudiantes y cuáles son las limitaciones desde el punto de vista de la simulación; en este punto se recolecta la información correspondiente a los modos de mayor interés para los docentes y qué atributos del dispositivo o simulador dan más dificultades.
4) Creación de la guía, en un documento se recopila toda la información e imágenes recogidas durante los pasos anteriores y se organiza la información dando jerarquía al interés de los docentes.
5) Retroalimentación: se entregan las guías y se hace una revisión por parte de los docentes e ingenieros del laboratorio de simulación clínica y se realizan las correcciones correspondientes.
6) Reporte de errores: una vez corregida la guía se deja un espacio abierto para los errores comunes reportados por el personal de ingeniería. La guía se va actualizando según la experiencia del personal al usar el dispositivo o simulador.

A continuación encontrara el diagrama de flujo el cual ilustra este procedimiento

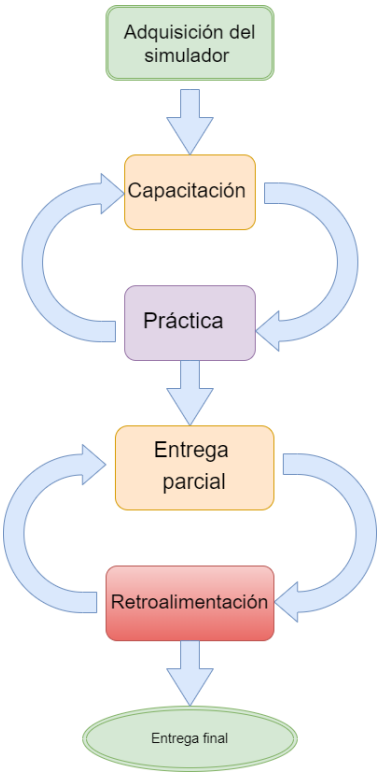


Ilustración 2 Diseño de Guías

Creación de la ampliación:

1) Diseño de la arquitectura: en esta fase, se realiza un análisis de las posibles arquitecturas de la aplicación teniendo en cuenta las necesidades del cliente, estas pueden ser local u online (basado en servicios), distribuida o centralizada entre otras; y las opciones de almacenamiento, usar ficheros o bases de datos.
2) Entorno de desarrollo: con base en los requerimientos encontrados en la primera fase se hace una análisis de los entornos de desarrollo de la aplicación, dependiendo de qué tantas prestaciones y servicios sean necesarios.
3) Desarrollo y correcciones: dado a que se eligió la metodología Agile, estas dos fases se deben realizar en conjunto, durante el desarrollo de la interfaz gráfica (frontend) el usuario debe hacer las correcciones o ajustes con base a su comodidad y facilidad.
4) Testeo: se prueba la aplicación e identifican errores tanto funcionales como conceptuales, es decir que la aplicación no falle, no tenga problemas técnicos y que la información solicitada y las interacciones correspondan con lo esperado.

3.7. Diseño de la arquitectura:

La IEEE define la arquitectura de software como:

"La Arquitectura de Software es la organización fundamental de un sistema encarnada en sus componentes, las relaciones entre ellos y el ambiente y los principios que orientan su diseño y evolución"[9].

Según esta definición una parte fundamental del desarrollo de este proyecto es poder definir de manera apropiada y precisa las partes funcionales del software y como se generan las interacciones de este, esta parte es fundamental ya que se identificó que iba a haber numerosos usuarios, la información alojada en la aplicación va a variar de manera constante y es importante tener claro cómo serán las interacciones para que no se vayan a generar problemáticas en su uso.

Sin embargo, realizar una arquitectura no se debe tomar a la ligera ya que hay diferentes métodos o estilos y subtipos, los cuales pueden brindar un mejor o peor funcionamiento de la aplicación, para poder elegir el tipo de arquitectura Paul Clements [10], define cuatro pasos llevar un buen diseño arquitectura.

1) Selección de la arquitectura
2) Representación de la arquitectura
3) Evaluación de la arquitectura
4) Desarrollo y evolución basados en la arquitectura

3.7.1 Selección de la arquitectura:

El primer paso es el más importante, ya que se según la decisión que se tome respecto a la arquitectura se desarrollaran los otros tres. A continuación, se mostrará una breve descripción de los métodos de arquitectura más populares y se analizará cual es el más conveniente para el desarrollo de la aplicación.

Arquitectura orientada a objetos (OO):

Los autores David Garlan y Mary Shaw definen su arquitectura basada principalmente y, valga la redundancia, en los objetos abstractos que se diseñan en el código. Esta arquitectura está mucho más relacionada con la estructuración directa del código y no con formas más abstractas como se mostrará posteriormente en otros métodos[10].

Esta arquitectura se puede definir usando los mismos principios del estilo de programación de donde hereda su nombre la POO (programación orientada a objetos), la encapsulación, herencia y polimorfismos, la invocación de los objetos y las funciones que lo hacen son el centro de esta arquitectura.

Una de las mayores ventajas de esta arquitectura se obtiene cuando se desee ser recursivo y reutilizar los objetos ya declarados.

Por otra parte, una de sus mayores desventajas se presenta sobre los llamados, estos deben tener un conocimiento total de los objetos y sus estructuras, si se realiza un cambio en algún objeto, este debe ser consecuente con todos sus métodos y llamados.

Arquitectura Basada en Eventos:

Esta arquitectura es usada principalmente en interfaces gráficas y sus acciones se detonan tras una acción tanto del usuario o de un sistema, esta segunda interacción es la que da su principal diferencia con respecto a la arquitectura OO, ya que usa una metodología de publicador-subscritor, donde un evento se suscribe a un tópico el cual

puede o no activar un evento. Un ejemplo de un sistema básico que recurre a esta estructura es una alarma, el evento seria activar el sonido y el publicador es un contador o timer, el cual puede ser un sistema asincrónico.

En resumen, la arquitectura basa en eventos se basa en los llamados a funciones y los sistemas de publicador-subscritor.

En el margen de las ventajas se puede identificar que los llamados son menos puntuales que en OO, lo cual permite una mayor libertad a la hora de modificar los objetos y componentes.

La principal desventaja es que en el momento que se desencadena un evento si no existe una conexión con los demás eventos, un actuador desconoce el estado de sus suscriptores u otros elementos que puedan realizar eventos, esto puede llevar a problemas en el comportamiento del sistema general ya que se podrían activar dos eventos los cuales ocuparán el mismo recurso y si las relaciones de jerarquía no han sido definidas de manera precisa, el código podría fallar.[10]

Arquitectura orientada a servicios SOA:

Este es un tipo de arquitectura que ha tomado relevancia en los últimos años debido a la popularización de las aplicaciones web y la implementación de métodos estandarizados, esta arquitectura nació a raíz de las arquitecturas distribuidas, pero se ha desarrollado al punto que hoy en día es considerada dentro de su propio tipo

Su primera definición fue dada por el grupo de investigación W3C, tras trabajar un año en esta y sus características y esta es:

“Un Web service es un sistema de software diseñado para soportar interacción máquina-a-máquina sobre una red. Posee una interfaz descripta en un formato procesable por máquina (específicamente WSDL). Otros sistemas interactúan con el Web service de una manera prescripta por su descripción utilizando mensajes SOAP, típicamente transportados usando HTTP con una serialización en XML en conjunción con otros estándares relacionados a la Web” [10].

Como se puede evidenciar en la definición dada por W3C, esta es una arquitectura que nace para solucionar los problemas de comunicación entre dispositivos (de los usuarios), dispositivos intermediarios, redes de procesamiento que se encargan de gestionar y “filtrar” la información antes de llegar al usuario final y el servidor, usando protocolos estandarizados en la red, tales como HTTP.

En esta arquitectura usa un método de servicios que es una función y que proporciona un servicio a otras entidades a través de interfaces públicas bien estructuradas.

Como se mencionó, esta arquitectura se basa en los protocolos estándar usados en la web, esto se presenta como la principal desventaja, ya que obliga al programador a ceñirse a las rígidas estructuras de estos protocolos. Sin embargo, esto también es una

ventaja, ya que permite una mayor conexión con herramientas u agentes de terceros ya que personas ajenas el proceso de desarrollo pueden utilizar recursos.

Conociendo las principales arquitecturas, sus ventajas y desventajas se decidió implementar la SOA, esto se debe a la flexibilidad que esta presenta para adaptar la información suministrada por el servidor, ya que como se pretende desarrollar una aplicación móvil esta debe ser pensada para diferentes sistemas operativos tales como Android y IOS, además de tener la opción de mudar el servicio a una versión de escritorio. Un servicio web permite la flexibilidad de mudarse de dispositivo y sistema operativo ya que esta opera bajo el protocolo de HTTP, el cual es propio de la web e indiferente al sistema operativo desde donde sea implementado.

Además de esto se pretende usar una arquitectura web la cual permita más opciones de almacenamiento de datos, con el fin de poder separar los datos del dispositivo, esto da dos ventajas fundamentales; la primera es que hace que la aplicación sea mucho más ligera para el usuario ya que no tiene que descargar todas las guías e información de los numerosos dispositivos del laboratorio de simulación clínica; la segunda y principal ventaja, es que permite usar una base de datos centralizada, la cual todos los usuarios compartan. La información, si se guardara solo de manera local, tendría el gran inconveniente de que cada vez que llegase un nuevo dispositivo los usuarios deberían actualizar su versión, esto podría generar discrepancias en las versiones de las bases de datos y que algunos usuarios (los que no han actualizado) no puedan acceder a toda la información de todos los quipos.

Cosa que no pasaría con el sistema web, cada vez que se realizara un cambio, la información estaría sincronizada online, todos los usuarios accederían a ella sin necesidad de realizar ningún cambio en su dispositivo de manera local.

En resumen en la tabla 1 poder encontrar una lista comparativa de las ventajas y desventajas de cada una de las posibles arquitecturas.

Arquitectura:	**Ventajas**	**Desventajas**
Orientada a objetos	Ser recursivo respecto a los objetos	Se debe tener control sobre todos sus objetos y llamados
Basada en Eventos	No se requiere un control absoluto sobre la declaración de objetos	Se debe establecer una jerarquía muy detallada de los ventos para evitar fallos
Orientada a servicios	Permite un mayor control de contenidos. Permite que todos los usuarios tengan una información sincronizada	Requiere de un servidor externo o plataforma de conexión

Tabla 1 Ventajas y desventajas según la arquitectura

3.7.2 Desarrollo de bases de datos.

Para el desarrollo de la aplicación se plantearon dos opciones de almacenamiento, el local y el online, cada uno posee pros y contras.

El almacenamiento local tiene la ventaja de ser más rápido y que puede funcionar en cualquier condición, esto se debe a que no depende de una conexión a internet, en contra parte se generan problemáticas como la diferencia de versiones, la necesidad de actualizar constantemente y cada vez que se realice un cambio en la base de datos y finalmente el alto peso de la aplicación.

Para almacenar los datos de forma local se podía optar por dos métodos diferentes, el uso de ficheros o una base de datos de tipo SQL. Un fichero es un documento de texto plano, en el cual se puede guardar todo tipo de información en formato String, este se identifica con la dirección de donde fue guardado, cada aplicación tiene un espacio en memoria para estos y únicamente la ampliación que lo creó puede acceder a ese archivo (Android).

La información de un fichero no está estructurada u organizada de ninguna manera y depende del programador asignar una estructura de trabajo, la forma de consulta es línea a línea. Este sistema tiene la ventaja de ser muy ligero y rápido ya que es un documento de texto plano.

Bases de datos SQL: por sus siglas en inglés Structured Query Language, lenguaje de consulta estructurada, es un sistema para almacenar información por medio de tablas interrelacionadas.

Una tabla está conformada por una llave primaria, la cual es el identificador (id) único del dato que se va a guardar y los atributos de este, diferentes tablas se pueden relacionar por medio de llaves secundarias. Para definir una tabla solo es necesario asignarle una llave primaria. Hay muchos métodos para diseñar bases de datos relacionales, sin embargo, una recomendación general es que los datos no sean redundantes.

Para hacer una consulta en una base de datos SQL solo es necesario conocer el id del “objeto” a consultar y el atributo que se desea conocer. Almacenar los datos en una tabla tiene la gran ventaja de que los datos están ordenados y estructurados haciendo que las consultas sean mucho más fáciles y claras.

El segundo conjunto de opciones de memoria se basa en el almacenamiento online en esta opción la información es almacenada en un servidor o gestor de bases de datos, tiene como desventaja que depende de una conexión a internet y es más lenta en comparación a la opción local. La mayor ventaja es la centralización y uniformidad de la información, además esto trae consigo la capacidad de reducir el peso de la aplicación.

Al igual que con el almacenamiento local, en la opción online se puede recurrir a las bases de datos de tipo SQL, sin embargo se debe mencionar que existen diferentes tipos administradores de bases de datos SQL, se deben especificar los tipos, sus ventajas y desventajas. La otra opción de almacenamiento de información online es firebase un sistema de base de datos **no** relacional.

MySQL: Es un administrador de bases de datos tanto de forma local como online, fue inicialmente desarrollada por MySQLAB (empresa fundada por Michael Widenius) y su primera versión fue liberada el 23 de mayo de 1995, hoy en día es propiedad de la organización Oracle Corporation y posee un sistema de licencia mixta, esto quiere decir que tiene una versión de prueba o aprendizaje la cual está limitada y una versión de pago la cual da soporte a servidores online y un almacenamiento mucho mayor.

MariaDB: Este gestor de bases de tipo SQL es un derivado de MySQL, esta fue diseñada con los mismos principios y estructura ya que fue creada por la misma persona que MySQL Michael Widenius, posee la misma API, biblioteca e instrucciones que su predecesora. La principal diferencia entre estas dos es que MrariaDB posee una licencia GPL (General Public License), un tipo de licencia que permite que todo usuario la use libremente sin la necesidad de hacer pagos. Dada la compatibilidad de estas dos, en la mayoría de las aplicaciones una es fácilmente reemplazable por la otra. Cabe mencionar que MariaDB también tiene soporte local como online, una de las desventajas de ser gratis es que posee pocos protocolos de seguridad ante ataque cibernéticos

Dado a que su licencia es abierta MariaDB es una opción viable para solucionar el problema de almacenamiento de datos para este proyecto, ya que permite guardar toda la información de los dispositivos en tablas relacionales, además de tener la posibilidad de ser instalado en un servidor para poder tener la información de forma online.

Es importante mencionar que para poder hacer consultas en bases de datos de tipo SQL online se suele usar una interfaz de programación de aplicaciones o API por sus siglas en ingles, esta es una función la cual se guarda en el servidor y permite conectar a los clientes con la información que este contiene, la ventaja de usar una API es que separa las peticiones del cliente de la estructura interna del servidor. Un cliente solo debe conocer que la variable que va a consultar y como va a recibir la información retornada, no es necesario conocer como ni donde se guarda la información dentro del servidor. La implementación de una API permite una mayor facilidad de uso para el usuario y un mayor nivel de seguridad a nivel interno del servidor.

Firebase: a diferencia de los dos gestores de bases de datos mencionados previamente firebese es un gestor de bases de datos únicamente online, el cual usa bases de datos no relacionales, este es un sistema muy similar al de ficheros, donde se usan documentos de texto plano para trasmitir la información, con la diferencia de que estos son compatibles con el formato JSON (JavaScript Object Notation o notación de objetos de JavaScript). Al uso este formato es mucho más fácil para acceder a las variables independientes o al conjunto de datos que se está usando ya que, como su nombre lo indica, se comporta como un objeto de JavaScript, en el cual el paquete de datos es el objeto y cada una de sus variables es un atributo.

Firebase fue desarrollada por Google en 2011 como una aplicación de sus servicios de google Cloud Platform, la cual reúne todas sus aplicaciones para desarrolladores web. Esta fue desarrollada con el objetivo de trasmitir datos en tiempo real a alta velocidad, es por esto que usa el formato JSON, el cual no tiene el mismo nivel de orden como las tablas relacionales pero es mucho más ligero ya que comparte la mayoría de los atributos del uso de ficheros.

Firebase tiene dos tipos de uso, uno libre pero limitado, el cual permite un máximo de 1GB de datos almacenados, 100 conexiones simultáneas 10GB de datos trasferidos al mes, entre otros valores, al sobrepasar estas limitaciones firebases cobra dependiendo de la cantidad de descargas o datos usados en el mes.

Servicio	Sin costo	Pago por uso
Autenticación	10.000 por mes	0.06 USD por verificación
Documentos almacenados	5 GB	USD 0.026/GB
Lectura de documentos	50.000 por día	USD 0.06 por cada 100,000
Conexiones simultaneas	100	200.000 por base de datos
Almacenamiento base de datos	1 GB	5 USD por GB
GB de descarga	10 GB	1 USD por GB
Descarga de documentos	50,000/día	USD 0.004 por cada 10,000

Tabla 2 precios de Firebase

Pese a las limitaciones que tiene la versión gratuita de firebase, esta también puede ser considerada como una opción de almacenamiento de datos para la aplicación ya que es un sistema de bases de datos mucho más ligero y rápido que las bases de datos SQL, además de también tener soporte online. Además, un documento en pdf de más de 250 páginas con imágenes tiene un peso aproximado de 2.000Kb, esto indica que en un espacio de 1Gb se pueden almacenar alrededor de 524 archivos como este.

Propiedad	MySQL	MariaDB	Firebase
Uso gratuito	no	no	Depende del consumo
Versión de prueba	si	si	Si
Capacidad	Depende del paquete	Depende del servidor	1 GB DB y 5 en documentos
Tipo	SQL	SQL	No relacional
Requiere servidor	si	si	No
Protección	si	no	Si
Poseen una API	no	no	si

Tabla 3 comparación de las bases de datos

3.7.3 Diseño de arquitectura:

EL siguiente apartado de divide en tres partes, la primera representación de la arquitectura, muestra dos grafos para cada una de las arquitecturas, este apartado se limita a dará una representación grafica de las partes de cada una de las arquitecturas implementadas.

Posteriormente encontrará la evaluación de la arquitectura, una análisis de la composición de cada una de las opciones a desarrollar.

Finalmente encontrará el apartado de Desarrollo y evolución basados en la arquitectura, en el cual como su nombre lo indica explicara y mostrara todas las tecnologías y protocolos usados para desarrollar cada una de las arquitecturas

3.7.4 Representación de la arquitectura:

Dado a que firebase y MariaDB son opciones viables, las cuales cumplen con los requisitos de desarrollo, se plantearon dos posibles opciones de arquitectura con base estas dos herramientas.

Bases de datos SQL, basado en una API

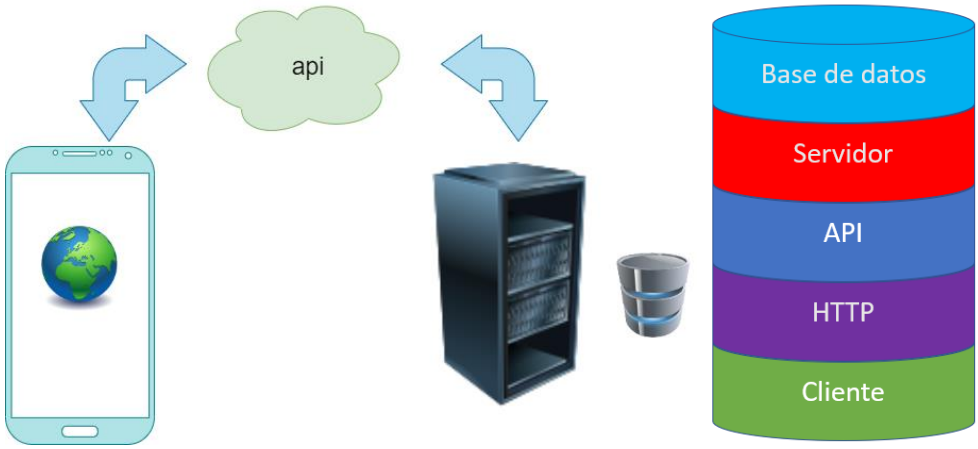


Ilustración 3 Arquitectura basada en api

Firebase

Ilustración 4 Arquitectura basada en firebase

3.7.5 Evaluación de la arquitectura

En el modelo basado en bases de datos SQL, se hace una estructura basada en una interfaz de programación de aplicaciones o API (el servicio), donde se crea un servicio online al cual el cliente hace una petición, la API recibe este llamado y se conecta al servidor, consulta la base de datos y retorna la información solicitada.

La arquitectura basada en la API permite que este servicio no dependa del entorno de desarrollo, tipo de cliente o sistema operativo, por lo tanto, si se desea programar una

aplicación para Android, IOS o una página online, solo es necesario conocer la dirección dende se aloja la API y la información que retorna.

Para que esta la arquitectura basada en API funcione es necesario establecer un protocolo de comunicación entre las partes, como se mencionó anteriormente, la arquitectura con base en los servicios propone usar el protocolo estándar HTTP, el cual será explicado a profundidad en el apartado de **Desarrollo y evolución basados en la arquitectura.** Además de esto se requiere un servidor donde se pueda instalar y almacenar la base de datos SQL, en este caso el gestor seleccionado fue MariaDB.

En este segundo caso la arquitectura basada en bases de datos NO SQL no es necesario configurar un servidor ya que el sistema de Firebase funciona como el gestor entre el cliente y la información de la base de datos.

Para poder conectar y comunicar la aplicación con las bases de datos es necesario usar una serie de métodos y protocolos establecidos en las referencias de la página de firebace. Cabe mencionar que este es un sistema bidireccional, así que la base de datos puede iniciar la comunicación entre el servidor y el cliente, cosa que no pasaba en el modelo anterior (basado en las bases de datos SQL). Esto permite que cuando un cliente haga un cambio en la base de datos esta le comunique a otro cliente sobre este cambio sin necedad de que el segundo solicite esta información.

3.7.6 Desarrollo y evolución basados en la arquitectura

Una vez planteadas las dos opciones de arquitectura, se deben conocer los protocolos de comunicación y diseño, para el caso de la arquitectura basa en API se requiere configurar un servidor y valga la redundancia una api y para el segundo es necesario conocer la documentación y api de firebase.

Protocolo de comunicación HTTP, mecanizo de comunicación entre el cliente, la API y el servidor:

*H*ypertext Transfer Protocol (HTTP) *(o* Protocolo de Transferencia de Hipertexto en español, es un estándar de comunicaciones creado a inicios de la década de los 90s (1991) como una colaboración entre World Wide Web Consortium y la Internet Engineering Task Force. Con el fin de facilitar la comunicación máquina-máquina. En este protocolo el cliente inicia la comunicación entre él y el servidor por medio de un método en el cual realiza una operación (dependiendo del método) y el servidor responde con la información solicitada (si la operación fue exitosa) y un código de notificación[11].

Método	*Función*
Get	*Este se usa para consultar el valor de un recurso especificado*
Post	*Envía datos para que estos sean guardados en el servidor en el recurso especificado*
Put	*Guarda, carga o sube un archivo o fichero en el servidor, a diferencia del método post usa una conexión socket establecida con el servidor para guardar el archivo*
Delete	*Borra el recurso solicitado*
Options	*Retorna los métodos que se pueden usar en un URL especificado*

Tabla 4 métodos HTTP [11]

Como se mencionó anteriormente además, de la información solicitada (si el método retorna consultas) junto a esta va un código de respuesta, esto puede ser usados para identificar si la operación se realizó de manera correcta o porque pudo fallar.

Códigos de respuesta HTTP:

Código del tipo:	Función
1XX	Indica que la solicitud fue recibida y está siendo procesada
2XX	Respuesta correcta, indica que el proceso se realizó de manera exitosa
3XX	Indica que es necesario realizar más operaciones para acceder al recurso o realizar la operación
4XX	Errores generados por el cliente
5XX	Errores generados por el servidor

Tabla 5 códigos de respuesta HTTP [11]

Es importante mencionar algunos códigos de respuesta típicos como el de recurso no encontrado u operación completada exitosamente

Código	Number	Función
100	Continue	Indica que todo se ha realizado de manera exitosa, si desea puede realizar otra petición
102	Procesando	La solicitud se ha recibido y aún se están procesando los datos
202	Ok	La solicitud se ha realizado de manera exitosa
400	Mala solicitud (Bad Request)	El servidor no puede interpretar la solicitud enviada por un error de sintaxis
401	No autorizado	Para obtener los recursos solicitados es necesario hacer una autenticación.
404	No se ha encontrado	El recurso solicitado no se encontró
500	Erro interno del servidor	Ha habido un error en el servidor ajeno al cliente
503	Servidor inhabilitado	El servidor no está listo para recibir peticiones.

Tabla 6 códigos específicos HTTP [12]

Además de usar el formato HTTP como mecanismo de comunicación, este también puede ser usado para diseñar el servidor, por medio de etiquetas HTTP, sin embargo este es un proceso lento y complicado. Para realizar estas tareas es mejor usar PHP, el cual es un lenguaje de programación, destinado al manejo de servidores. PHP viene del acrónimo Hypertext Preprocessor o procesador de hipertextos, es de tipo dinámico, esto quiere decir que las variables pueden cambiar de tipo según la operación a la cual se sometan (Strinn, int, booleana, entre otras), además de esto también es identado, lo cual indica que no es necesario compilarlo.

PHP fue inicialmente desarrollado por Resmus Lerdorf en 1994, con el propósito de monitorizar las visitas en su página web, inicialmente PHP significaba Personal Home Page. Posteriormente, en 1997, Andi Gutmans y Zeev Suraski lo modificaron generando una versión muy parecida a la usada actualmente.

Es ampliamente usado en la industria, se estima que se encuentra implementado en alrededor del 80% de los servidores, algunas de las empresas más reconocidas que lo son Wikipedia, Facebook y WordPress, este último un sistema de gestionar contenidos en páginas web

Una de las funciones más útiles de PHP es manejar las solicitudes y el contenido de respuesta en los servidores, puede recibir como argumento operaciones de HTLM, es el puente entre las acciones del usuario y las respuestas de servidor, mejor conocido como API.

Continuando con los requisitos de la segunda opción de arquitectura. En las referencias de Firebase se encontró que la API usa el formato Json como sistema de transmisión y agrupamiento de datos.

Como se mencionó anteriormente el formato JSON es una estructura de datos usada en internet para que estos sean ligeros y su transmisión sea mucho más rápida.

Por lo general, la trasmisión se hace en formato String (variables de tipo texto) y es necesario usar una función la cual convierte de una variable tipo texto a un objeto. Para poder leer las variables enviadas en un archivo JSON, solo es necesario tratar los datos como el objeto y referirse a las variables como atributos de este.

El formato JSON siempre se abre y cierra con llaves { }, los nombres de las variables van entre comillas y sus valores se asignan con dos puntos, si el valor es de tipo sitring se debe poner entre comillas si es de tipo int, no es necesario, para separar dos variables se usa coma (,) fuera de las comillas.

Ejemplo: {“temperatura”:24, “humedad”:12, “nombre”:”Emmanuel”}

Métodos de firebase: los datos deben ser escritos en una variable asincrónica, esto quiere decir que las variables se pueden modificar de dos maneras diferentes: la primera es de manera síncrona y se hace por medio de una asignación; la segunda hace referencia su modo asíncrono, el cual se da cuando se registra un cambio en la base de datos. Este cambio se hace de manera automática.

Firebase requiere de una variable de tipo referencia, la cual indica dónde esta guardada la información

Instanciar las variables:

```
Private DatabaseReference mDatabase;//
mDatabase = FirebaseDatabase.getInstance().getReference();
```

Para las operaciones básicas de lectura y escritura se pueden usar los métodos de la base de datos mDatabase .setValue() y .getValue(), los cuales son respectivamente, asignar o escribir el valor de la variable y obtener o leer el valor de la variable, la variable para leer debe ser especificada entre los parénesis (argumento de la función).

Para facilitar el estructurado de los datos se recomienda declarar un objeto, el cual indique cuáles serán los atributos de cada valor guardado. Esto se debe a que firebase permite un sistema de herencias para guardar los datos, donde un objeto tiene sus atributos y cada atributo también puede tener sus subatributos, al igual que en la POO y en los objetos de JavaScript.

En firebase se pueden usar diferentes niveles de información, o información anidada, para acceder a ellos se usa el método child, el cual accede a los niveles más bajos de la estructura de información.

Ejemplo:

mDatabase.child("usuarios ").child("nombresDeUsuario").setValue(“Emmauel”);

En el ejemplo se puede ver como se llama al objeto mDatabase, el cual tiene un atributo de usuario, este a su vez tiene atributo de nombres de usuario y se le va a asignar el valor de “Emmanuel”.

3.8. Entorno de desarrollo

Este apartado se debe dividir en dos partes, la primera mostrará las herramientas de desarrollo para la aplicación y en la segunda las usadas para probar, codificar y simular el servidor.

Para poder desarrollar la aplicación es importante definir en qué lenguaje de programación y en cual entorno de desarrollo se programará. Esto es importante ya que, dependiendo de qué entorno se escoja, la ampliación puede tener un acceso más profundo a las funciones del teléfono, como por ejemplo poder funcionar en segundo plano y mostrar notificaciones cuando está cerrado, entre otros.

Las dos plataformas de desarrollo de aplicaciones de Android son App inventor y Android Studio, ambas son de libre uso, la primera es una aplicación web y la segunda es una herramienta que se debe descargar e instalar en el equipo.

App inventor es una herramienta desarrollada por Google y mantenida por los servidores de la escuela de educación superior Massachusetts Institute of Technology (MIT), su primera versión fue liberada el 15 de diciembre del año 2010.

La filosofía de app inventor es enseñar a estudiantes y personas sin altos conocimientos en programación, a poder diseñar sus propias aplicaciones para el sistema operativo Android, el sistema se basa en programar mediante bloques. Las instrucciones son intuitivas y cada componente tiene su propio set de instrucciones y métodos.

App inventor tiene un gran alcance ya que por medio de este se puede acceder al uso de los sensores del dispositivo tales como cámara, giroscopio y parlantes, entre otros. Además de esto, tiene opciones de conectividad como bluetooth y conexión a internet.

Sin embargo, alguna de las limitaciones que tiene este sistema es el acceso a los permisos y usos en segundo plano, esto quiere decir que con app inventor no se pueden obtener permisos para que la aplicación se ejecute en segundo plano, esto genera algunas limitaciones como no poder generar alertas o notificaciones automáticas cuando la aplicación está cerrada. Otra desventaja de esta herramienta es que los elementos gráficos, como botones y notificaciones son los mismos de la versión de Android que se usó para desarrollarles, lo cual hace que la apariencia de la aplicación se vea antigua o desactualizada.

Actualmente APP inventor se encuentra probando una versión para el desarrollo de aplicaciones en el sistema operativo de apple IOS, se debe mencionar que esta versión se encuentra en fase beta.

De manera precisa APP inventor, es una herramienta fácil de usar en la cual no se requiere escribir líneas de código, haciéndola más intuitiva. Tiene algunas desventajas de diseño como la restricción en la creación de interfaces gráficas y acceso a funciones más en segundo plano. Sin embargo, una aplicación desarrollada en este sistema tiene un gran alcance.

Android Studio es la plataforma de desarrollo oficial de Android y fue creada por Google I/O, con el objetivo específico de crear aplicaciones para esta plataforma y reemplazar a Eclipse (entorno de desarrollo creado para programar en el lenguaje de programación java, también usado para crear apps de Android).

Android Studio se vasa en el software IntelliJ IDEA de JetBrain y es de uso libre por medio de la licencia Apache 2.0.

Esta plataforma se basa métodos de programación “tradicionales” en los cuales se requiere escribir extensos bloques de código, declarar funciones y objetos en el lenguaje java. Esto puede ser un inconveniente ya que se requiere de conocimientos previos sobre programación para poder usar esta herramienta. Sin embargo, esto también permite que el programador tenga un mayor control sobre el desarrollo de la aplicación, teniendo acceso a virtualmente todas las configuraciones y capacidades del dispositivo.

Cabe mencionar que como Android Studio está diseñado para programar en (valga la redundancia) Android, no es dependiente del dispositivo y puede ser usado para crear aplicaciones para teléfonos, tabletas, relojes, neveras y otros dispositivos que usen un sistema operativo derivado de Android.

Android Studio es un sistema mucho más complejo, el cual requiere de un conocimiento previo para ser usado, pero en contraparte esta dificultad trae consigo un mayor nivel de control permitiendo al desarrollador llevar una aplicación a sus límites.

Dado que la aplicación se está desarrollando en un entorno educativo, es viable tomar la herramienta de APP inventor para crear la app, ese entorno es ideal para tomarlo como punto de partida e iniciar el desarrollo, inconvenientemente, como no se tiene acceso a todos los permisos del dispositivo se debe buscar otra opción para descargar los documentos por medio del navegador web. Posteriormente, se puede recurrir a la plataforma de Android Studio, la cual permite tener descargas directas y más facilidades de diseño, haciendo la aplicación más atractiva. Además de esto, se debe mencionar que dado a que Android Studio es desarrollado por Google tiene una completa integración con las aplicaciones y servicios de la compañía, tales como firebase. Esto es un punto de decisión fundamental, ya que la opción de firebase sigue en estado de prueba o experimental en app inventor.

Con base en lo anterior, se decidió implementar la arquitectura basada en API en APP inventor y la basada en firebase en Android Studio.

Continuando con las herramientas de desarrollo web para la configuración del servidor y la API (arquitectura basada en API) se encontró Sublime Tex, este es un entorno

de desarrollo multipropósito, esto se debe a que es compatible con más de 40 lenguajes de programación diferentes, entre esto los que más destacan son C, Python y PHP, entre una larga lista.

Sublime Tex es una versión mejorada de Vim, un entorno de desarrollo creado por Bram Moolenaar en 1991, este era un entorno que se ejecutaba sobre una consola y debía ser utilizado únicamente por comandos de teclado, presentaba algunas ventajas como autocompletado de texto, corrección ortográfica y fácil manejo entre ventanas, este diseño minimalista le permitía ser rápido y eficiente ya que una interfaz ocupa memoria y genera dificultades en altos rendimientos.

Para poder hacer pruebas se usó la herramienta XAMPP un simulador de servidores, esta es una herramienta fundamental del desarrollo, que permite crear y simular de manera local un servidor y gestionar bases de datos entre otras herramientas.

XAMPP fue creado por un grupo de desarrolladores conocidos como Apache Friends, su primera versión fue liberada en el 2010. Fue desarrollada para solucionar la dificultad que tenían los desarrolladores para conseguir un espacio en un servidor para probar sus desarrollos, el grupo Apache Friends, creó XAMPP como una herramienta que pudiese usar el almacenamiento interno del dispositivo como un servidor, en el cual cualquier dispositivo conectado a la misma red que el ordenador pudiese conectarse.

XAMPP hace referencia a, **X** (cualquier sistema operativo), **A**pache, como servidor, **M**ariaDB/MySQL, como gestor de bases de datos, **P**HP y **P**erl, lenguajes de programación basado en servidores y C respectivamente.

3.9. Desarrollo y correcciones

Como se mencionó anteriormente una de las fortalezas de la metodología implementada es el trabajo en conjunto del desarrollador y el usuario, en esta fase se trabajó activamente con el personal del laboratorio de simulación clínica y el asesor, presentando entregas bisemanales, del desarrollo de la interfaz gráfica.

Como se mencionó previamente, dos arquitecturas diferentes fueron escogidas para ser implementadas, esto significa que se realizarán dos aplicaciones, las cuales tendrán las mismas funciones y utilidades pero diferente lógica de funcionamiento.

Se decidió iniciar con la arquitectura basada en API, y construirla en el entorno APP inventor, esto debido a que es la primera aproximación a la creación de aplicaciones. Dado que esta aplicación no debe realizar tareas en segundo plano, las limitaciones de este no serán un problema durante el desarrollo, sin embargo, el no tener acceso al administrador de descargas genera la necesidad de encontrar otras opciones para visualizar o descargar las guías.

Para poder crear esta app se requiere construir dos estructuras principales, la primera es la app en sí, el mecanismo por el cual el usuario realzará las búsquedas y visualizar los resultados de sus consultas.

La segunda parte y menos obvia son los servicios del servidor, los cuales a su vez se componen de dos partes, la creación de las tablas y el diseño de la API. Para crear las tablas se usó un administrador de bases de datos de tipo SQL y la API se escribió en PHP.

La principal regla de la creación de tablas es evitar las redundancias, la información de una tabla no debe repetirse en otras. Una estrategia para distribuir las información es generar conexiones entre tablas.

Teniendo en cuenta estas indicaciones se procedió a crear dos tablas, una con la información relevante de los equipos (el criterio de relevancia fue dado por los doctores y principalmente por los ingenieros del laboratorio de simulación clínica), y una "llave" secundaria, la cual permitiría conectar con otra tabla, la de guías.

La segunda tabla únicamente posee las guías de usuario, esta se creó con el propósito de evitar redundancia y aligerar la base de datos. Esto se debe a que cada dispositivo tiene información diferente y única, como el número de placa y las fechas en las que se realizaron los mantenimientos. Sin embargo, una misma línea de dispositivos comparten el mismo manual, entonces en lugar de asignar un manual a cada dispositivo se hizo una tabla con estos. Las tablas se conectan por el tipo de dispositivo, por lo tanto al buscar determinada placa, la primera tabla retornara entre otros valores que tipo de dispositivo y por medio de este valor, se puede consultar en la tabla de manuales.

Una vez la estructura de los datos estuvo definida en sus respectivas tablas, se continuó a diseñar la API, para esto se usó el lenguaje PHP. Para que esta funcionara debía tener acceso a la base de datos, como se van a consultar dos tablas diferentes se decidió crear dos APIs diferentes, las cuales técnicamente tienen el mismo funcionamiento (por medio del método pos publicar un identificador, consultar una bases de datos y retornar la consulta) pero cada una consultará una tabla diferente.

En la estructura de la API se deben dar permisos de consulta al lugar donde se aloja la base de datos, el servidor, especificar en cuales bases de datos (asumiendo que puede existir más de una), especificar la tabla de consulta y la forma en la que se retornaran los datos. Esto es especialmente importante ya que por medio de este retorno es como se conectó la API con la aplicación.

Posteriormente se inició el desarrollo de la aplicación en APP inventor, para esto se deben definir los componentes y servicios que va a usar la aplicación. El diseño se divide en dos partes el frontend y backend, el primero se refiere a todo lo que el usuario va a ver e interactuar y el segundo a qué hacen y cómo se relacionan estos.

El diseño del Frontend es de muy importante ya que es lo que definirá si la aplicación es de fácil uso y amigable con el usuario, en esta parte del desarrollo es donde más se debe comunicar con el cliente, para este caso el personal del laboratorio de simulación clínica. Con ellos se debe definir cuantos botones se deben colocar, cuantas pantallas se deben usar, una para consultar y una para visualizar o todo junto en una sola y si la información está bien distribuida y el uso es claro.

La codificación del Backend estuvo completamente a cargo del desarrollador, ya que acá se coloca el código y la lógica que hace funcionar la aplicación, se definió que la aplicación requería de dos servicios, conexión a web y el uso de la cámara. Dentro de la segunda pantalla se realizan los llamados a la API, en esta parte es cuando se genera la

conexión entre la información y la aplicación, para realizar cada consulta se debe seguir los parámetros estipulados en el API, bajo qué variable se realiza el post y cómo se reciben los datos para finalmente mostrarlos

3.10. Testeo y llenado de base de datos

Una vez completadas las partes del desarrollo, se continua con la parte de probar todas las funciones y la integración de ella en la aplicación, para esto se colocan valores de control en la base de datos, estos valores deben ser de fácil consulta y verificables, esto se hace con el propósito de comparar y contrastar la información consultada con la guardada en el servidor de manera fácil y rápida.

Una vez verificado el comportamiento del sistema, se realiza una encuesta a un usuario, el cual después de usarla dará su retroalimentación, en esta encuesta es fundamental evaluar aspectos como la facilidad y estética de la aplicación.

Una vez terminada la aplicación en APP inventor, se continuó programando la aplicación en Android Studio, esta se dejó de segunda ya que se requiere de un mayor conocimiento técnico para el desarrollo. Una vez se tuvo una familiarización con el desarrollo de aplicaciones se pudo pasar a esta plataforma. Para el diseño de la aplicación se tomó como base el diseño de la primera, debido a que las dos van a tener el mismo uso, su diferencia radica en la arquitectura, cómo funciona la lógica.

Al igual que con la primera aplicación, el desarrollo se debió dividir en dos partes, el diseño en el software de Android, el cual permitirá generar la interfaz gráfica de usuario, realizar las consultas y mostrar la información y la segunda es la configuración en la nube, solo que para este caso, en lugar de tener un servidor, se usó Firebase.

Se inició con la creación de la base de datos, esto se debe hacer en la plataforma online de Firebase, este sistema tiene dos modalidades uno de base de datos en tiempo real, el cual tiene la misma función de SQL, en este espacio se guardarán los datos de los equipos. El segundo es el almacenamiento online, desde este apartado se puede almacenar y guardar documentos de todo tipo.

La principal diferencia entre la configuración del servidor y Firebase, es que no es necesario configurar una API, esto se debe a que la plataforma ya tiene la suya, esto tiene la complejidad de que se debe consultar la documentación sobre cómo hacer los llamados y cómo recibir la información, ya que estos parámetros no están definidos por el programador sino por el servicio.

Una vez configurada la base de datos y el almacenamiento de los documentos, se inició el desarrollo de la aplicación en Android Studio, la ventaja de esta plataforma es que es de Google y, como se mencionó anteriormente, tiene una completa integración con los servicios de esta compañía, esto incluye a Firebase.

Se implementó el mismo diseño que en la primera aplicación, con algunas sutiles diferencias en los puntos donde no era muy claro qué opción de diseño era más eficiente.

Más adelante se compararán los dos diseños de manera directa para elegir el mejor. Finalmente se conecta la base de datos online con la aplicación y se hacen las mismas pruebas que con la primera aplicación.

3.11. Pruebas y encuestas

Como se mencionó se realizaron dos tipos de pruebas, la primera tenía como objetivo evaluar los aspectos subjetivos de las aplicaciones, esta encuesta debía ser respondida por los usuarios y tenía preguntas relativas a la facilidad de uso, si el diseño fue intuitivo, si la información suministrada era relevante para el usuario y la presentación estaba lo suficientemente bien ordenada y clara

El segundo conjunto de pruebas estuvo orientado a comprobar el funcionamiento de la aplicación, usaba una serie de test con los cuales se comprobaba que cada botón y elemento de la interfaz hiciera lo esperado, que la información solicitada coincidiera con la guardada en el servidor y que esta se mostrara en los apartados correspondientes.

Para poder comparar el funcionamiento y calidad de las aplicaciones se aplicaron los mismos test en ambas, haciendo que las dos fueran evaluadas con el mismo parámetro y los resultados pudieran ser fácilmente comparados.

3.12. Despliegue

En el seguimiento posterior a la entrega de las aplicaciones se pretende seguir dando seguimiento a las bases de datos. Dentro de este mantenimiento se contempla, la opción de crear una herramienta para editar y actualizar las bases de datos, esto incluye una opción para subir la nuevas guías, junto a esto se pretende hacer una rúbrica o guía para hacer estos documentos, en los cuales se dejen estipulados algunos lineamentos para que mantengan la estructura de los que se crearon durante el semestre 2020-1

4. RESULTADOS

Se diseñaron dos aplicaciones, la primera se desarrolló en Android Studio siguiendo una arquitectura basada en servicios, usando Firebase. La aplicación cuenta con dos pantallas.

La primera pantalla y principal, se denomina “consulta”, en esta se puede encontrar una barra de texto la cual permite introducir el número de placa que se va a consultar. Además de esto encontrará dos botones, uno para escanear y otro para buscar.

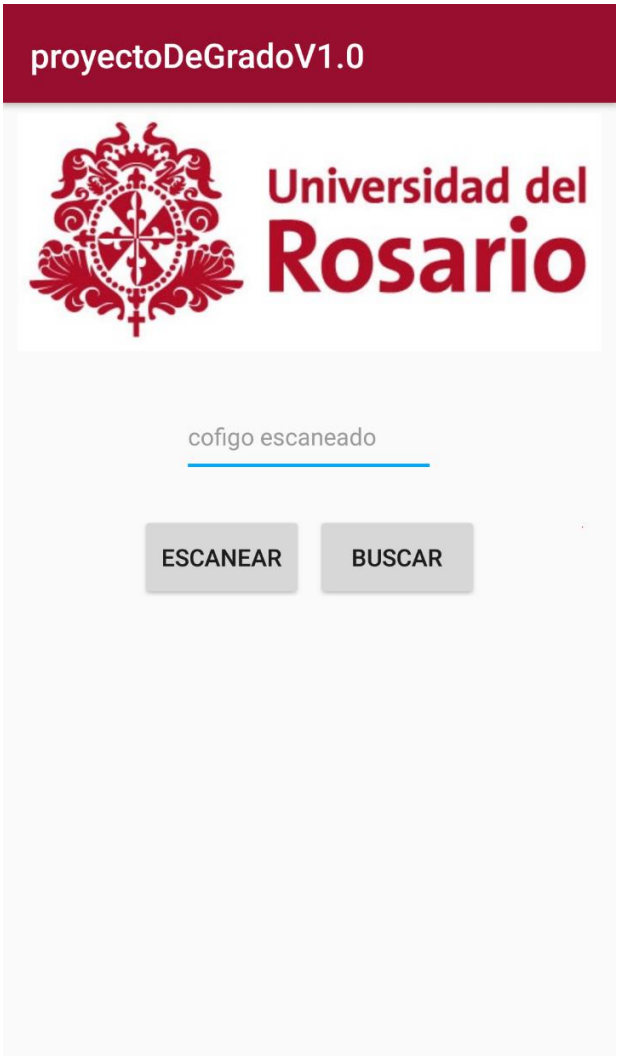


Ilustración 5 Android Studio pantalla principal

El primero, como su nombre lo indica, tiene la función de abrir la cámara y escanear el código de barras, una vez escaneado, el valor se asignará a la barra de texto. El segundo botón funciona para consultar los datos del dispositivo al cual pertenece la placa,

adicionalmente, si no se ha escrito nada en la barra de texto, al presionar el botón buscar este abrirá automáticamente la cámara y escaneará un código de barras.

La primera notificación que puede recibir el usuario es un mensaje emergente, el cuan indica que no se encontró la información en la base de datos, si es que esta no está disponible, de lo contrario se desplegará la segunda pantalla, la de visualización.

La pantalla de visualización posee cuatro etiquetas estáticas en la parte izquierda, las cuales indican que atríbuto están consultando, estos hacen referencia al equipo, el nombre del equipo, marca, modelo y serial.

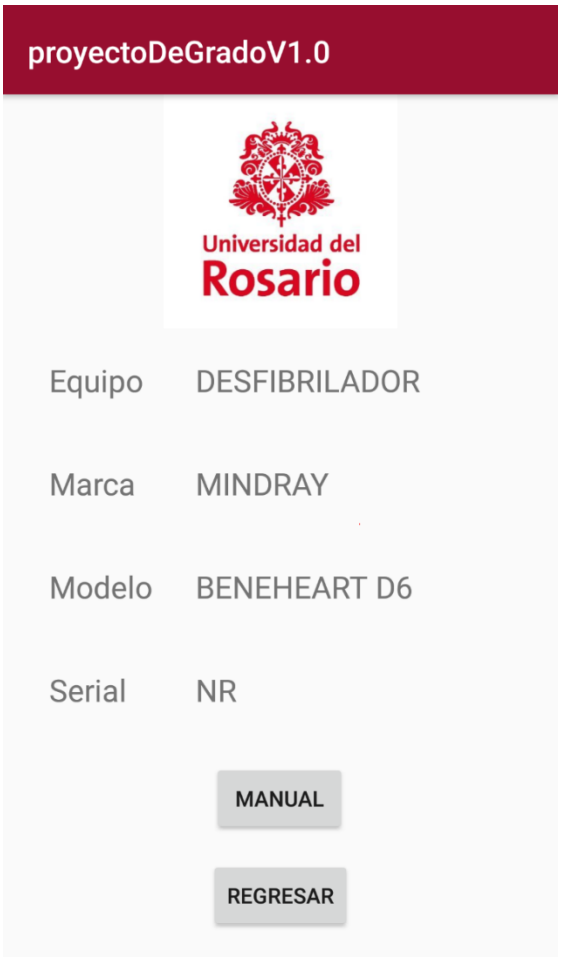


Ilustración 6 Android Studio pantalla de visualización.

En frente a estas etiquetas encontrará otros cuatro valores, estos cambian según la consulta e informan y representan el valor de dicho atributo.

En la parte superior se colocó el logo de la institución, esto no solo tiene un propósito estético, sino que tiene la función de regresar a la pestaña principal. En la parte inferior se encuentran dos botones, uno para volver a la pantalla principal, este está etiquetado con el texto “REGRESAR”.

El otro botón, identificado como “MANUAL”, tiene la función de descargar el manual correspondiente al dispositivo buscado, una vez iniciada la descarga este notificará por medio de un mensaje emergente.

Bases de datos.

Como se mencionó en la metodología, la información de los equipos y sus guías fueron separados en el documento de dispositivos y en el apartado de Storage se guardaron los documentos descargables.

En el documento que contiene la información de dispositivos se creó la rama principal “equipos” en este todos sus “hijos” coinciden con el número de placa de algún dispositivo del laboratorio de simulación clínica.

El mecanismo de consulta de firebase se basa en una estructura de árbol, donde hay un nodo principal, este posee una serie de hijos y un hijo puede tener sus propios hijos o un conjunto de atributos.

Un hijo da acceso a un nivel más profundo de información, datos anidados, en cambio un atributo son los valores asociados a ese nivel de información.

Ilustración 7 Estructura de información firebase.

Para acceder a un atributo se debe hacer por medio de los métodos get del objeto bajo el cual está declarada la estructura de datos, esta consulta se hace internamente en el código, es ajena a firebase.

En cambio, para acceder a los hijos se debe usar el método child, esto se hace dentro de la estructura de consulta de firebase, dos hijos son diferentes y cada uno tiene sus propio conjunto de valores o atributos.

En el diagrama de árbol de la ilustración 7 se puede observar cómo grado-95886 (este es el nombre de la base de datos) tiene el hijo equipos, este a su vez tiene los hijos 2, 91879, 9789588294568 y 9789681640590 la estructura de consulta para el hijo 91879 sería igual a:

grado-95886**.**child(equipo

s)**,**child(91879)

Análogamente esta consulta se hace siguiendo la ruta **grado-95886/equipos/91879,** esto retorna el valor de 91879 junto a todos sus tributos, id, marca, modelo, seria y tipo.

Para facilitar la búsqueda de equipos particulares, la estructura de datos se diseñó de modo que las placas (identificadores únicos) fuesen hijos de equipos, para realizar una consulta solo era necesario establecer la ruta hasta equipos y posteriormente se concatenaba con la placa escaneada, este sistema retornaba el objeto del equipo consultado con los valores de interés, id (tipo de equipo), marca, referencia, serial y tipo

El valor tipo no es una variable la cual se vaya a visualizar en la interfaz gráfica este es un valor que se usa para conectar las guías con la información de los dispositivos.

En firebase las guías de los dispositivos se guardan en la opción de almacenamiento (Storage), en este apartado no se tiene una estructura como la presentada previamente. En cambio, se usó el nombre de los documentos como parámetro de búsqueda. El valor tipo de cada dispositivo coincide con el nombre del archivo de la guía de usuario correspondiente, de esta manera cuando se desea descargar una guía, el sistema busca un documento que tenga como nombre el atributo tipo.

La segunda aplicación se creo en APP inventor, usando la arquitectura basada en servicios API, al igual que la aplicación diseñada en Android Studio, esta posee dos pantallas, una de consulta (principal) y otro de visualización.

En la primera pantalla encontrará una etiqueta, con un mensaje de bienvenido y una breve explicación de qué tareas puede realizar en la aplicación. Seguido a esto se ubica un cuadro de texto donde puede ingresar el valor de la placa, este posee un “Hint”, un texto trasparente que desaparece cuando se empieza a escribir en el cuadro, el cual indica las funciones del botón buscar.

Ilustración 8 App inventor pantalla principal.

En la parte inferior se ubica el botón buscar, a diferencia de la anterior aplicación esta solo posee un botón, este realiza las tareas de escanear y buscar. Si se presiona el botón de buscar y no hay texto en el cuadro de placa, automáticamente se abre la cámara. En cambio, si el usuario sí ingreso algún valor de placa se abrirá la próxima página y mostrará los datos.

La segunda página tiene como encabezado el logo de la Universidad del Rosario, además de tener fines estéticos, el logo también tiene la función de regresar a la pantalla principal.

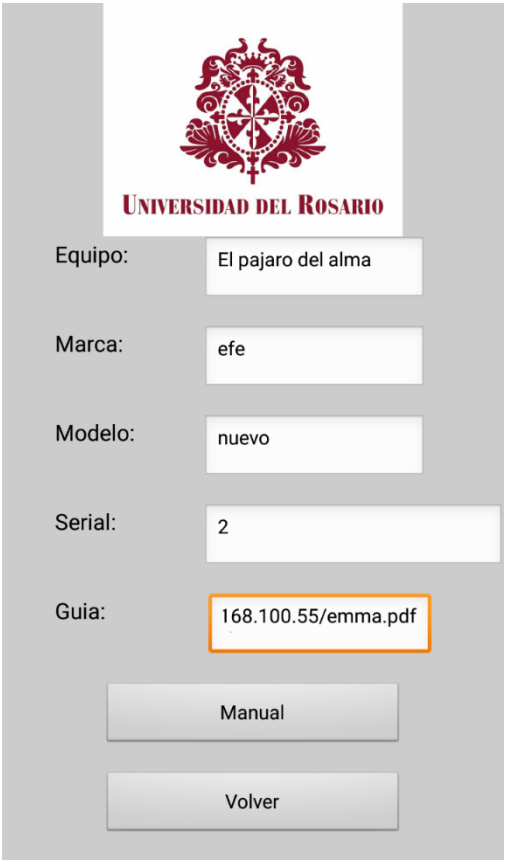


Ilustración 9 App inventor pantalla de visualización.

Inmediatamente debajo del logo se ubican cinco etiquetas, cada una indica un valor de la consulta, Equipo, Marca, Modelo, Serie y guía. Enfrente se encuentran otros 5 cuadros de texto, a estos se les asignará un valor de manera automática, cuando la pantalla carga, esto se asignará en función del resultado de la consulta, con la excepción del de guía.

En la parte inferior se encuentra el botón de Manual, cuando este es presionado tendrá la función de asignar un valor en el apartado de guía. Se mostrará un link el cual si es seleccionado se puede abrir con el lector de pdf de Google drive, o se puede buscar directamente en el navegador para visualizar o descargar.

Administración de datos:

Dado a que en esta opción se usa un tipo de almacenamiento SQL, se crearon dos tablas, una para la información de los dispositivos y la segunda para las guías.

La tabla de equipos posee la misma información que el documento de firebase, esto incluye el atributo tipo, la principal diferencia entre estos dos métodos de almacenamiento radica en cómo se realizan las conexiones entre las sus partes.

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra
1	placa	varchar(255)	utf8mb4_general_ci		No	*Ninguna*		
2	nombre	varchar(255)	utf8mb4_general_ci		No	*Ninguna*		
3	marca	varchar(255)	utf8mb4_general_ci		No	*Ninguna*		
4	modelo	varchar(255)	utf8mb4_general_ci		No	*Ninguna*		
5	serial	varchar(255)	utf8mb4_general_ci		No	*Ninguna*		
6	tipo	varchar(255)	utf8mb4_general_ci		No	*Ninguna*		

Ilustración 10 tabla equipos

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra
1	tipo	varchar(50)	utf8mb4_general_ci		No	*Ninguna*		
2	enlace	varchar(50)	utf8mb4_general_ci		No	*Ninguna*		

Ilustración 11 Tabla guías

Para el caso de esta aplicación la variable tipo, se conecta con tabla de guías, en la segunda tabla, la llave primaria es la variable tipo. El único valor que tiene la tabla además de su llave primaria es el enlace para descargar la guía. Nuevamente todos los equipos que compartan una guía tendrán el mismo valor asignado “tipo”.

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra
1	placa	varchar(255)	utf8mb4_general_ci		No	*Ninguna*		
2	nombre	varchar(255)	utf8mb4_general_ci		No	*Ninguna*		
3	marca	varchar(255)	utf8mb4_general_ci		No	*Ninguna*		
4	modelo	varchar(255)	utf8mb4_general_ci		No	*Ninguna*		
5	serial	varchar(255)	utf8mb4_general_ci		No	*Ninguna*		
6	tipo	varchar(255)	utf8mb4_general_ci		No	*Ninguna*		

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra
1	tipo	varchar(50)	utf8mb4_general_ci		No	*Ninguna*		
2	enlace	varchar(50)	utf8mb4_general_ci		No	*Ninguna*		

Ilustración 12 Estructura de información SQL

Se realizó una encuesta a la ingeniera biomédica de la Universidad del Rosario, quien fue la supervisora de la práctica, sobre las características gráficas, facilidad y calidad de información de las aplicaciones y guías realizadas durante el semestre.

App inventor:

Pregunta	1	2	3	4	5
¿Que tan adecuada es la distribución de los botones?				X	
¿Los botones son de fácil acceso ?				X	
¿La información suministrada es relevante?					X
¿La información suministrada fue suficiente (no hay ausencia de datos) ?				X	
¿La información esta bien distribuida ?				X	
¿Facilidad de uso?				X	
¿Las opciones son claras?				X	
¿Presentación ?				X	
¿Esquema de colores?			X		

Puntaje promedio: 4

Android Studio

Pregunta	1	2	3	4	5
¿Que tan adecuada es la distribución de los botones?				X	
¿Los botones son de fácil acceso ?					x
¿La información suministrada es relevante?				X	
¿La información suministrada fue suficiente (no hay ausencia de datos) ?				X	
¿La información esta bien distribuida ?				X	
¿Facilidad de uso?				X	
¿Las opciones son claras?				X	
¿Presentación ?				X	
¿Esquema de colores?				X	

Puntaje promedio: 4.2

Preguntas respecto a las guías desarrolladas en el semestre

Pregunta	1	2	3	4	5
¿Las imágenes son claras?					**X**
¿Las imágenes son relevantes ?					**X**
¿La imágenes están relacionadas con el contenido?					**X**
¿Las imágenes son oportunas, están cerca al párrafo donde son usadas?					**X**
¿Es clara la redacción ?				**X**	
¿Usa leguaje técnico ?				**X**	
¿Entiende el lenguaje técnico?				**x**	
¿Es fácil la lectura ?					**X**
¿Encontró los modos de uso mas frecuentes ?				**X**	
¿La tabla de partes (inventario) esta completa?				**X**	
¿Es estética la guía ?					**x**

Puntaje promedio: 4.5

En la encuesta realizada se puede evidenciar cómo la información presentada es de interés para el ingeniero, además que la presentación y distribución de esta es adecuada y de su agrado. La aplicación diseñada en Android estudio tuvo una ligera mejor puntuación en los aspectos gráficos, además de la posibilidad de descargar directamente los documentos.

La encuesta evidenció que el desarrollo de la guías también fue exitoso, la información estaba clara y ordenada, las imágenes fueron relevantes y prácticas para su desarrollo, la redacción y calidad de uso también fueron evaluadas con excelencia.

5. DISCUSIÓN

Aspectos gráficos y uso:

Comparando los diseños de la interfaz gráfica de la aplicación, se determinó que la mejor opción fue la diseñada con Android estudio. Esto se debe a la presencia de dos botones, uno para buscar y otro para escanear. Sin embargo, que el botón de buscar también funcione como opción de escáner cuando no se ha escrito ninguna placa no fue del agrado de los usuarios, se recomendó cambiar esto por una aviso emergente, el cual sugiera el uso del escáner.

El usuario encontró de su agrado la posibilidad de descargar los documentos del equipo de manera directa, sin tener la necesidad de abrir otras aplicaciones. Esta opción incluso superó la opción de visualizar directamente el PDF dentro de la aplicación mediante el administrador de PDF online de drive.

Arquitectura:

Ambas arquitecturas tuvieron un buen desempeño técnico, el usuario no pudo percibir una significativa diferencia entre la velocidad de obtener los datos mediante una aplicación y la otra. Por lo tanto, la velocidad extra que proporciona firebase, no se tomó como un criterio a favor como parte de su arquitectura.

En el aspecto de la facilidad del manejo de los datos se encontró una clara diferencia, esto se debe a que para poder importar datos a firebase, estos deben estar en formato Json, teniendo en cuenta que no se pueden exportar directamente de Excel (programa en el que la Universidad del Rosario se guardan los datos), esto genera la necesidad de actualizar los datos de manera manual o usar una herramienta externa la cual permita la convención. Este problema aumenta cuando la base de datos se hace más compleja, entre más ramificaciones tenga, mayor complejidad tendrá convertir el archivo.

En cambio, la base la base de datos SQL requiere de un archivo CSV para poder importar información, esto es una ventaja debido a que Excel permite crear este archivo directamente desde las hojas del inventario de la universidad, el único cuidado que se debe tener para poder hacer esta gestión es ordenar las columnas en el mismo orden que la tabla de SQL.

6. Trabajos futuros

Como se mencionó en las conclusiones, la aplicación en Android Studio tubo un mejor residiendo por su aspecto, facilidad de uso y atributos como la opción de descargar las guías, sin embargo la arquitectura en la cual esta se baso no fue la mejor opción. Por lo tanto un futuro trabajo debería recolectar los mejores aspectos de ambas aplicaciones y crear una tercera versión en Android Studio con una arquitectura basada en API, la cual gestione los datos mediante SQL y permita descargar las guías directamente al almacenamiento del dispositivo.

Además de esto se dejan las guías de los equipos como muestra para poder seguir generando en base a los nuevos equipos que se adquieran en el laboratorio de simulación clínica. Se recomienda el uso abundante de imágenes acompañando cada paso, si es posible que sea de más de un ángulo, además de esto, que el ingeniero que las desarrolle siempre asista a la capacitación para los doctores docentes, esto para poder identificar las modalidades y configuraciones que generan más interés, así mismo con las dificultades de los doctores y recomendaciones de los capacitadores.

7. Conclusiones

- Este proyecto diseño una interfaz gráfica de usuario de fácil manejo en la cual, tras escanear la placa del dispositivo biomédico, se puede consultar la guía de usuario, la hoja de vida y el estado del equipo, permitiendo que los doctores, auxiliares e ingenieros del laboratorio de simulación clínica de la Universidad del Rosario, puedan acceder a la información de manera fácil y oportuna.
- La interfaz gráfica, esquema de colores y facilidad de uso de la aplicación de Android estudio, basada en dos botones para la navegación y posibilidad de descargar los documentos fue la mejor opción, según los usuarios
- La opción elegida para administrar el almacenamiento de los datos es SQL, esto de debe a que las ventajas de velocidad de firebase no son relevantes. Sin embargo, la característica que si hace la diferencia es la facilidad de importar y manejar los datos entre SQL y Excel.
- La mejor arquitectura es la basa en API, ya que esta da mayor flexibilidad y control sobre la gestión de datos, además de la posibilidad de adaptar esta arquitectura a otros servicios.
- Según los datos de la encuesta se pudo evidenciar que el desarrollo de la guías fue exitoso, la información e imágenes contenidas en ellas son prácticas y claras permitiendo un acceso rápido y oportuno.
- La implementación de la metodología Agile permitió que las últimas versiones de las aplicaciones tuvieran muy buenos resultados en su aspecto gráfico y utilidad. Esto se debe a la constante retroalimentación de parte de los usuarios. Este efecto fue más evidente en la creación de las guías, las cuales tuvieron varias revisiones y el usuario tenía más parámetros para comentar y solicitar cambios.

8. Referencias

[1] “NOELLE® Simulador de parto con PEDI® Blue Neonate - 1012732 - W45177 - S550.” .

[2] Pardo Campillo José Alberto, “Editorial: La Universidad del Rosario en la Historia - Universidad del Rosario.” 2016.

[3] A. González Vargas, M. Collazos, L. García, J. Ladino, A. Cano, and S. González, “Análisis del estado actual de la ingeniería clínica en las instituciones hospitalarias de cali,” *Rev. Ing. Biomédica*, vol. 9, no. 18, pp. 73–80, 2015, doi: 10.24050/19099762.n18.2015.770.

[4] invima, “GUÍA PARA MANEJO, MANTENIMIENTO Y LIMPIEZA DE EQUIPOS UTILIZADOS EN BANCOS DE TEJIDOS Y BANCO DE GAMETOS,” 2019.

[5] República de Colombia, “Resolución 4725 de 2005,” *Minist. Protección Soc.*, vol. 2005, no. Diciembre 26, p. 31, 2005.

[6] J. O. Wear, “clinical engineering handbook,” p. 4300083, 2020.

[7] J. Joskowicz, “Reglas y prácticas en eXtreme Programming,” *Univ. Vigo. España*, pp. 1–22, 2008.

[8] María Tena, “¿Qué es la metodología ‘agile’? | BBVA,” *Bbva*. p. 1, 2018.

[9] H. Koning and H. Van Vliet, “Real-life IT architecture design reports and their relation to IEEE Std 1471 stakeholders and concerns,” *Autom. Softw. Eng.*, vol. 13, no. 2, pp. 201–223, 2006, doi: 10.1007/s10515-006-7736-6.

[10] C. B. Reynoso, “Introducción a la Arquitectura de la Información,” *Univ. Buenos Aires*, pp. 1–17, 2016, doi: 10.1016/j.msea.2016.09.043.

[11] Wikipedia, “Protocolo de transferencia de archivos.” .

[12] moz://a, “Códigos de estado de respuesta HTTP - HTTP.” 2019.

[13] MINISTERIO DE LA PROTECCION SOCIAL, “Decreto 4725 De 2005,” vol. 2005, no. Diciembre 26, pp. 1–31, 2005.

9. Anexos

9.1. Anexo 1

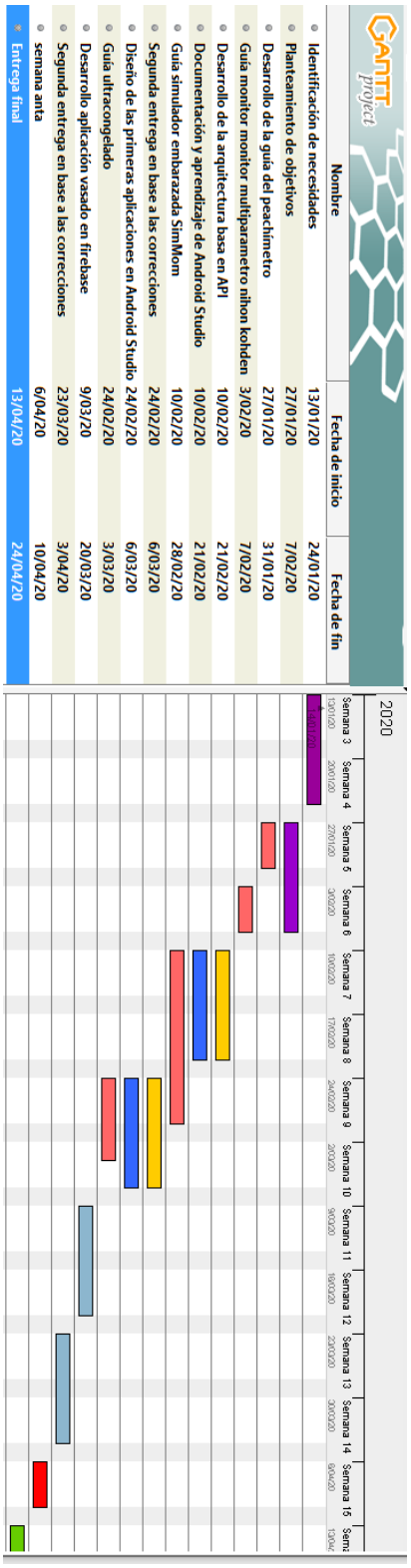


Ilustración 13 Siagrama de Gantt

9.2. Anexo 2

SIMULADOR AVANZADO DE PARTO.
SimMon- Laerdal

EMMANUEL ZAMBRANO.
INGENIERÍA BIOMÉDICA

Contenido

PARTES DEL SIMULADOR AVANZADO SimMon

PARTE DEL SIMULADOR	IMAGEN
Parte superior del simulador, Incluye módulo de ECG en los pines metálicos, se puede entubar y medir pulso yugular.	
Pecho del simulador, posee conexiones para la lectura de ECG y es capaz de recibir descargas de un DEA.	
En el brazo derecho del simulador se puede tomar el pulso radial