CIS2021CP17

Implementation of an algorithm for the diagnostic approach of patients with joint pain

Diego Fernando Arroyo Graciano
Juan Sebastián Castañeda Flórez
Nicolás Daniel Peralta Sopó
Camilo Andrés Ruiz Uribe

PONTIFICIA UNIVERSIDAD JAVERIANA
FACULTAD DE INGENIERIA
SYSTEMS ENGINEERING PROGRAM
BOGOTÁ, D.C.
2021

CIS2021CP17

Implementation of an algorithm for the diagnostic approach of patients with joint pain

Autor(es):

Diego Fernando Arroyo Graciano

Juan Sebastián Castañeda Flórez

Nicolás Daniel Peralta Sopó

Camilo Andrés Ruiz Uribe

UNDERGRADUATE FINAL PROJECT REPORT PERFORMED IN ORDER TO ACCOMPLISH ONE OF THE REQUIREMENTS FOR THE SYSTEMS ENGINEERING DEGREE

Director

Ing. Leonardo Flórez Valencia, Ph.D.

Jurados del proyecto final de pregrado

Dr. Daniel Gerardo Fernández Ávila, M.D MS.c Ph.D.

Ing. Efrain Ortiz Pabón, Ph.D.

PONTIFICIA UNIVERSIDAD JAVERIANA
FACULTAD DE INGENIERÍA
PROGRAMA DE INGENIERÍA DE SISTEMAS
BOGOTÁ, D.C.
Noviembre, 2021

PONTIFICIA UNIVERSIDAD JAVERIANA

FACULTAD DE INGENIERIA

PROGRAMA DE INGENIERÍA DE SISTEMAS

Rector de la Pontificia Universidad Javeriana

Padre Jorge Humberto Peláez Piedrahita, S.J.

Decano de la facultad de ingeniería

Ing. Lope Hugo Barrero Solano Ph.D.

Director(a) del programa de ingeniería de sistemas

Ing. Alexandra Pomares Quimbaya Ph.D.

Director(a) del departamento de ingeniería de sistemas

Ing. César Julio Bustacara Medina Ph.D.

Artículo 23 de la Resolución No. 1 de junio de 1946

"La Universidad no se hace responsable de los conceptos emitidos por sus alumnos en sus proyectos de grado. Sólo velará porque no se publique nada contrario al dogma y la moral católica y porque no contengan ataques o polémicas puramente personales. Antes bien, que se vean en ellos el anhelo de buscar la verdad y la Justicia"

AGRADECIMIENTOS

Nosotros como grupo, queremos expresar nuestro eterno agradecimiento para con todo el departamento de ingeniería de sistemas, en especial, con todos los profesores, que hemos tenido a lo largo de este desafío profesional, ya que cada uno, con su respectiva área del conocimiento, aportó para nuestro crecimiento, tanto personal como profesional y definitivamente, logró crear un impacto suficiente para no desfallecer con los múltiples obstáculos que debimos superar para llegar a este punto, y estar *'ad portas'* de lograr la consecución de nuestro título profesional.

Así mismo, cada uno de los integrantes, agradecemos a nuestras familias, por todo el apoyo, ayuda y esfuerzo invertido en nosotros. Sin estos elementos, nada de lo logrado hubiese sido posible.

CONTENIDO

ABSTRACT

Medicine has been experiencing growth over the last decades. Although, there is a recurring need for specialists. Even though there are countless resources to diagnose nowadays. New tools and techniques are published constantly with the very purpose of helping to lessen this need. Rheumatology is one of the most affected, not only for doctors, but patients too. They are the diseases that deteriorate the most people's lives, above lung and cardiovascular diseases. This solution is aimed to support rheumatology diagnosis by implementing a decision tree based on a study published by Dr. Daniel Fernandez Avila. It will not only help decrease the need for rheumatology specialists, but it will also be a great tool for communities where commodities, such as a stable internet connection, are rare or unusual.

I. INTRODUCCIÓN

El presente documento busca dar a conocer el contexto del problema y el proceso de planeación y desarrollo de la aplicación *RheumatApp*. Esta es una aplicación la cual busca apoyar a los médicos en atención ambulatoria y general a la hora de diagnosticar enfermedades reumáticas. *RheumatApp* busca a través de su funcionalidad principal aumentar el porcentaje de asertividad en el diagnóstico de estas enfermedades usando un algoritmo de decisión. Finalmente, como trabajo futuro, los datos recolectados a través de esta aplicación podrán ser utilizados para *Machine Learning.*

El contenido de este documento pretende dar un contexto general, comunicar los objetivos del trabajo de grado, y la descripción y análisis del problema que se busca solucionar. Además, se abordan temas de diseño, desarrollo y pruebas realizadas a la solución. Finalmente, se reconocen las conclusiones y el trabajo futuro.

II. DESCRIPCIÓN GENERAL

1. Oportunidad, Problema

1.1. Contexto de la problemática

Los diagnósticos médicos han venido cambiado a través de las décadas. Según el informe de O. Rivero: A principios del siglo pasado un médico contaba con un solo elemento de diagnóstico: la entrevista con un paciente. Una confianza en lo que el paciente relataba y una inspección en el área que el paciente considerará era más que suficiente para dar un diagnóstico puesto que no contaban con análisis de laboratorio y estudios radiológicos que les apoyarán a la decisión. [48] Sin embargo, es evidente que estos métodos no daban diagnósticos acertados normalmente.

En la actualidad los recursos de auxilio en diagnostico han aumentado considerablemente, desde estudios de hormonas, células, rayos-x hasta nuevas terapías físicas y terapías medicinales. [48] Sin embargo, en la forma en que ha crecido la industria, la necesidad de especialistas es una necesidad recurrente. Por más recursos de diagnóstico que tenga un médico general los conocimientos necesarios para tener un diagnóstico acertado en una especialidad siguen siendo bajos. Por ello, es importante el desarrollo de herramientas que apoyen la toma de decisiones a médicos generales.

En principio, la implementación de estas herramientas, según un estudio de *DXPlain*, supone una gran ayuda que puede llevar a la mejora de diagnósticos de hasta el 30% y a la reducción de costes en los mismos hospitales por estancias más cortas y eficaces. [46]

Por su lado, la reumatología, según la fundación española de reumatología, es una rama de la medicina que se encarga de prevenir, diagnosticar y tratar las enfermedades musculoesqueléticas y autoinmunes sistemáticas. Existen más de 200 enfermedades reumatológicas y pueden afectar en cualquier rango de edad. Sin olvidar que son las

enfermedades que más deterioran la vida de las personas, por encima de las enfermedades pulmonares y cardiovasculares. [45] Es por ello por lo que una mayor eficiencia en esta especialidad puede significar una mayor calidad de vida en la población.

1.2. Formulación del problema

En el caso de la reumatología la cifra de diagnósticos es preocupante ya que .
Esto se debe a que generalmente las enfermedades reumáticas son principalmente consultadas con médicos generales. [43]

Un estudio llevado a cabo por el asesor Daniel Fernández Ávila, con el apoyo del método *Delphi*, desarrolló un árbol de decisión capaz de diagnosticar pacientes reumatológicos entre 4 enfermedades distintas: artritis reumatoide, espondilo artritis con afectación periférica, lupus eritematoso sistémico con afectación articular y osteoartritis. Con el ejercicio *Delphi* se encontró que el factor común entre estas enfermedades era su síntoma inicial: dolor en las articulaciones. Por consiguiente, este árbol llevo a unos resultados bastante prometedores entre los reumatólogos; con un 37.3% más de acierto en el grupo que implemento esta solución. [44]

1.3. Propuesta de la solución

Por todo lo anteriormente demarcado se propuso una solución enfocada en desarrollo de software y, a futuro, en análisis de datos. La solución consistió en una aplicación multiplataforma que permitió implementar el árbol de decisión propuesto por el Dr. Daniel Fernández Ávila en el documento: "*Effectiveness of the use of an algorithm in the diagnostic approach of joint pain patients by primary care physicians*" [36] que fue publicado en la revista médica "*Clinical Rheumatology*".

A partir de la aplicación se validó la apropiación de los médicos con el sistema propuesto, principalmente del ya mencionado Dr. Fernández, y se recolectaron los distintos diagnósticos realizados por los médicos con los que se busca generar una base de

datos que servirá para hacer un estudio de analítica de datos que permita llevar este sistema inicial a la implementación de un sistema experto más adelante.

1.4. Justificación de la solución

En Colombia actualmente se presenta un grave déficit de médicos reumatólogos, esto ha causado que la gran mayoría de pacientes que ingresan a un centro de atención medica por dolores articulares sean atendidos en primera instancia por médicos generales [5]; lo anterior, debido a que en nuestro sistema de salud el paciente no puede ir directamente con un reumatólogo y debe ser remitido por un médico general. Lo perjudicial de este sistema es que el diagnóstico brindado por el médico general, a veces, es erróneo [2]. Principalmente, cuando la consulta es por dolor articular. Por esto es necesaria el desarrollo de una aplicación que apoye a los médicos generales a la hora de realizar sus diagnósticos y poder brindar una rápida y mejor atención a los pacientes.[44]

Si bien ya hay aplicaciones existentes que ayudan con este problema, ninguna guía al médico a través del proceso, todas se basan en simples preguntas en base a las condiciones del paciente Estas mismas preguntas a veces son muy dirigidas a conocimientos específicos de médicos reumatólogos de tal modo que no ayudan a solucionar el problema real de ayudar a los médicos generales. [19][20][21]. Además, ninguna utiliza un modelo ya aprobado para el diagnóstico de las enfermedades [43] cosa que también puede ser perjudicial a la hora de realizar un proceso. Por último, estas aplicaciones solo están disponibles de manera móvil lo que podría complicar su uso en los casos donde no se cuente con dispositivos móviles.

2. Descripción del proyecto

Mediante las metodologías escogidas (véase VFP y sección III), se propuso crear una aplicación que soporte y valide el diagnóstico médico de condiciones reumatológicas. Para esto, como eje central, se planea implementar el árbol de decisión descrito en la

sección anterior. Dicha aplicación es multiplataforma, de tipo híbrido, centrándose en web y móvil.

Una vez culminado su desarrollo, se hace la recolección y validación de información en forma de múltiples diagnósticos. Esto con el fin de elaborar una base de datos capaz de solventar en un futuro estudios que tengan como propósito la creación de un sistema de mayor envergadura, capaz de emular el comportamiento humano y tomar decisiones relevantes al campo de estudio, tal como lo haría un experto en reumatología.

2.1. Objetivo general

Desarrollar una aplicación multiplataforma con el fin de apoyar y validar el diagnóstico de enfermedades reumáticas basándose en el algoritmo propuesto. [44]

2.2. Objetivos específicos

- Analizar los flujos del árbol de decisión para el diagnóstico de enfermedades reumatológicas.
- Desarrollar el algoritmo para el diagnóstico de pacientes con dolor articular con una interfaz amable con el usuario.
- Desarrollar un aplicativo multiplataforma que implemente el algoritmo de diagnóstico de enfermedades reumáticas.
- Validar el nivel de exactitud de diagnóstico del árbol de decisión y el nivel de satisfacción de una muestra de los usuarios con la aplicación.

2.3. Entregables, estándares y justificación

Tabla 1: *Entregables, estándares y justificación.*

Entregable	**Estándares asociados**	**Justificación**

Propuesta de trabajo de grado (VFP)		Documento en el cual se van a detallar los aspectos más relevantes de la propuesta de trabajo de grado.
SPMP	*ISO/IEC/IEEE 16326:2009* *Systems and software engineering — Life cycle processes — Project management*	En este documento se va a describir las principales actividades del proyecto, los principales hitos, recursos necesarios y presupuesto según el estándar 16326 de la ISO/IEC e IEEE [16].
SRS	*ISO/IEC/IEEE 29148:2011* *Systems and software engineering — Life cycle processes — Requirement's engineering*	Este documento describe los procesos para la captura de requisitos, así mismo como la especificación de los requisitos funcionales y no funcionales, para este se tendrá en cuenta el estándar 29148 de la ISO/IEC/IEEE [15].
SDD	*IEEE 1016-2009 — IEEE Standard for Information Technology—Systems Design—Software Design Descriptions*	Describir el diseño del software a desarrollar, para tener direcciones claras sobre como documentar y cómo diseñar la arquitectura del proyecto. Todo esto según el estándar 1016 de la IEEE. [12]
Código fuente de la aplicación	*Clean Code*	Código fuente de la aplicación ya implementada siguiendo los requisitos definidos en el SRS y las direcciones del SDD, para este utilizaremos las guías del *Clean Code* [12].

Documento de plan de pruebas de software	ISO/IEC/IEEE 29119-2:2013 ISO/IEC/IEEE 29119-3:2013 ISO/IEC 20246:2017	En este se pretende mostrar todas las fases de pruebas realizadas más sus resultados, para asegurar el buen funcionamiento de cada una de las funcionalidades de la aplicación y la calidad del software.
Manual de usuario	*ISO/IEC/IEEE 26512:2011* *Systems and software engineering — Requirements for acquirers and suppliers of user documentation*	Este es un manual para facilitar el uso de la aplicación por parte del usuario y guiarlo a través de esta. Para este se utilizará el estándar 26512 de la ISO, IEC e IEEE.[14]

III. CONTEXTO DEL PROYECTO

1. Background

Las bases del proyecto se sustentan en el *déficit* en materia de diagnósticos acertados como se puede evidenciar en la investigación "*Design of an algorithm for the diagnostic approach of patients with joint pain*" [43]. Partiendo del problema anteriormente citado se plantea una solución que implementa dicho algoritmo, entendiendo que el algoritmo en el contexto de la investigación es un árbol n-ario y tiene como fin dar un diagnóstico relacionado a la enfermedad reumática del paciente, estas enfermedades están relacionadas al desgaste de músculos, huesos, articulaciones, ligamentos y el tejido conectivo.

Las enfermedades reumáticas que el algoritmo está capacitado para diagnosticar son, **Artritis Reumatoide**, esta es una variante de la artritis que no solo afecta a las articulaciones sino distintas partes del cuerpo debido a la inflamación causada por la misma enfermedad. Estas afectadas pueden ser: piel, ojos, pulmones, corazón o vasos sanguíneos. [41], **Espondilo artritis**, esta es otra variante de la artritis que ataca principalmente la columna. Sin embargo, hay personas que sienten afectaciones en sus brazos y piernas o a veces involucran a la piel, los intestinos y los ojos. [40], **Lupus eritematoso sistémico**, es una enfermedad autoinmune donde los anticuerpos atacan diferentes órganos y tejidos provocando inflamaciones. [18], **Osteoartritis**, es la enfermedad articular más común. Se da principalmente debido al envejecimiento, al desgaste o a la ruptura de una articulación. [36]. Una vez el algoritmo identifique alguna de estas enfermedades la aplicación continuará con el proceso de almacenamiento dando la oportunidad de futuros trabajos sobre los datos.

Como se menciona anteriormente, esta solución entre otras cosas pretende proveer las bases de datos para trabajos futuros de ***Machine Learning*** sobre los datos obtenidos de diagnósticos. ***Machine Learning*** se puede definir como un conjunto de métodos matemáticos con los que se pretende que las maquinas aprendan a realizar ciertas acciones sin estar explícitamente programadas para eso, únicamente apoyada por los datos de entrada. Para esto se cuenta con una base de datos no relacional implementada por el servicio de ***Firebase*** de ***Google***. este servicio a su vez es considerado como un *Backend as a Service*. Estos tipos de servicios proveen integraciones como el almacenamiento en nube, gestión de usuarios, entre otras. Finalmente, la aplicación está diseñada como una **aplicación web progresiva (PWA)**. Se conoce como un tipo de software que se consume a través de la web, pero luce como una aplicación nativa y tiene ventajas como el uso *offline* y la versatilidad de ser ejecutada en cualquier plataforma que use un navegador compatible.

2. Análisis del contexto

Actualmente existen otras soluciones que no cumplen a cabalidad el objetivo planteado, entre los desarrollos existentes encontramos aplicaciones únicamente informativas de contenido relacionado a la reumatología [19] y otras que ofrecen un diagnóstico partiendo de los resultados obtenidos en exámenes médicos específicos ordenados por especialistas, además estas aplicaciones solo pueden ser utilizadas en dispositivos móviles [21].

La solución propuesta es una PWA, compatible con cualquier plataforma y su funcionamiento está basado en encaminar al médico a través de un árbol con preguntas que pueden ser resueltas en su mayoría sin necesidad de exámenes especializados. La aplicación también es considerada como punto de partida para el entrenamiento de inteligencia artificial pues ofrece la posibilidad de confirmar y calificar los diagnósticos obtenidos. Finalmente, el árbol implementado en la solución propuesta tiene una efectividad comprobada de más del 90% [44].

2.1. Alternativas

La primera alternativa, y la más importante, una aplicación móvil llamada Rheumahelper para reumatólogos profesionales. Esta es una aplicación gratis, utilizada actualmente por más de 7000 reumatólogos a nivel mundial. [21]

Entre sus funcionalidades principales se encuentra: clasificación de enfermedades y acceso a información relevante [21], esta aplicación no ofrece un diagnóstico, si no una lista de enfermedades y sus posibles síntomas, además solo está disponible en dispositivos móviles.

RheumApp (la solución propuesta en este documento) busca dar una asistencia a médicos generales en el diagnostico a pacientes con síntomas reumatológicos. La principal diferencia vendría siendo la implementación del algoritmo por el cual se guía al médico

a través de una serie de preguntas hasta el diagnostico, además este ya ha probado ser un éxito al momento de diagnosticar.

En segundo lugar, otra aplicación móvil publicada por el American College of Rheumatology con el nombre de Guideline & Criteria App [20]. La aplicación ofrece una gran biblioteca de información sobre tratamientos, síntomas y guías relacionadas a las enfermedades reumáticas. Sin embargo, no presenta asistencia directa al diagnóstico.

Por último, una aplicación móvil dirigida a los pacientes. La aplicación se llama Reumapp lanzada por la coordinación de postgrado de reumatología de la Universidad Nacional de Asunción. En esta se encuentran desde materiales educativos, temas de cuándo debe acudir o hasta los Servicios de Urgencias. [19]

2.2. Comparación

Tabla 2: *Comparación de alternativas*

Alternativa	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Rheumahelper	NO	SI	SI	NO	SI	SI	6
Guideline & Criteria App	NO	SI	SI	NO	SI	SI	1
ReumApp	NO	NO	SI	SI	SI	SI	1
RheumatApp (Solución propia)	*SI*	*NO*	*NO*	*SI*	*SI*	*SI*	*1*

Nota:

1. Diagnóstico de enfermedad

2. Clasificación de enfermedades
3. Tratamiento de enfermedades (Cómo)
4. Aproximación Web
5. Aproximación IOS
6. Aproximación Android
7. Cantidad de lenguajes disponibles

IV. ANALISIS DEL PROBLEMA

1. Requerimientos

Los requerimientos de este proyecto fueron definidos al inicio de este en conjunto por los integrantes del grupo de trabajo, el director y el asesor del trabajo de grado. Estos requerimientos fueron principalmente enfocados a dos aspectos específicos, el primero, una aplicación funcional, práctica y con una interfaz fácil de utilizar para los diferentes usuarios de la aplicación y el segundo, lograr persistir los datos generados por la aplicación en una base de datos para que posteriormente puedan ser utilizados por modelos de inteligencia artificial.

Tabla 3: *Requisitos funcionales principales*

Requerimiento	Descripción
RF_001	El sistema debe solicitar al usuario ingresar las respuestas de un cuestionario médico relevante que deben ser persistidas.
RF_002	El sistema debe validar la información ingresada por el usuario.
RF_003	El sistema debe analizar la información ingresada por el usuario y emitir un diagnóstico.

RF_004	El sistema debe hacer uso del árbol de decisión, para emitir el diagnóstico correspondiente.
RF_005	El sistema debe persistir el diagnóstico en la base de datos.
RF_006	El sistema debe persistir los datos ingresados por el usuario de manera local, en caso de no contar con conexión a internet

2. Restricciones

2.1. Restricciones

Para que la aplicación funcione de manera óptima según lo planeado, se plantearon algunas restricciones que debía cumplir la aplicación. Estas están expuestas en la siguiente tabla cuyas restricciones fueron acordadas desde el principio del proyecto por los integrantes del grupo de trabajo.

Tabla 4: *Restricciones separadas por tipo.*

Tipo	Restricciones
Software	La aplicación debe funcionar, sin necesidad de comunicarse con otros sistemas externos, por lo que no hay dependencias respecto de otros sistemas con excepción a las conexiones con la base de datos. La aplicación debe funcionar como una PWA (*Progressive Web App*), en las distintas plataformas propuestas a través de un solo desarrollo.

Hardware	La aplicación debe funcionar en cualquier dispositivo electrónico (PC, celular, Tablet) con una antigüedad no mayor a 7 años, un sistema operativo no mayor a 5 años de antigüedad y que puedan acceder a un navegador.
Generales	La aplicación debe funcionar normalmente sin necesidad de una conexión activa a internet, de tal modo que pueda funcionar en cualquier situación y lugar.

2.2. Supuestos y dependencias

Para que el usuario tenga una buena experiencia a la hora de usar de la aplicación se plantearon los siguientes supuestos y dependencias. Si alguno de estos no llegara a cumplirse o fuera parcialmente cumplido, la aplicación podría no funcionar de manera óptima. Estos fueron planteados por los integrantes del grupo de trabajo al principio del proyecto.

- El médico debe contar con un dispositivo electrónico con un sistema operativo ya sea Windows MacOS o cualquier distribución de Linux para computadores y Android e IOS en el caso de los celulares y tabletas.
- El dispositivo debe contar con cualquier navegador de los más conocidos como Microsoft Edge, Google Chrome, Mozilla Firefox, etc.
- Se debe tener una conexión a internet estable la primera vez que va a ser usada la aplicación para poder cargar los datos de la aplicación.

3. Especificaciones funcionales

3.1. Diseño

Para el desarrollo del proyecto basados en las restricciones de hardware, como son el desconocimiento de los equipos con los que se cuenta en atención ambulatoria, se propone la implementación de una *Progressive Web Application* (PWA) de esta manera

garantizar el uso en cualquier plataforma, manteniendo los requisitos de hardware mínimos referentes a almacenamiento y procesamiento.

La arquitectura de la aplicación se plantea de 3 capas, donde se identifican capas de presentación, lógica y almacenamiento, además la base de datos debe ser basada en el paradigma relacional debido al trabajo futuro en inteligencia artificial.

3.2. Atributos del sistema de software

Confiabilidad. A continuación, se van a especificar los mecanismos usados para el manejo de la información almacenada y la manera en que se planea mantener operativa la aplicación.

- La aplicación debe tener la capacidad de persistir datos y, en caso de no ser posible, guardar dichos datos.
- La arquitectura de la aplicación debe garantizar la tolerancia a fallos. Es decir, un respaldo para mantener al aire ante un error. Además, debe tener respaldo de la información.

Disponibilidad. La aplicación debe contar con una disponibilidad anual del 98%.

Seguridad. La aplicación debe contar con balanceadores de carga y como único punto de acceso desde internet la capa de presentación. Toda la información personal ha de estar cifrada.

Mantenibilidad. El desarrollo debe ser basado en métodos como Open/Close y *Clean Code* garantizando la auto documentación del código y la mantenibilidad.

Portabilidad. La aplicación debe poder correr en cualquier sistema operativo o cualquier dispositivo con conexión a internet sin importar la calidad de dicha conexión.

3.3. Requerimientos de la base de datos

Para los requisitos de la base esta debe soportar una frecuencia alta de acceso para ejecutar operaciones de escritura y además contar con una capacidad alta de almacenamiento y velocidad de descarga, pues se espera alojar una gran cantidad de datos que serán usados para desarrollo de inteligencia artificial

Los datos almacenados en la base de datos serán en su mayoría de tipo *boolean*, además de algunas cadenas de caracteres, enteros y archivos para guardar los archivos que soporten alguna de las respuestas provistas por el médico.

Finalmente, el diseño de la base de datos esta requerirá de una tabla con información muy básica del paciente que garantice su anonimidad y el flujo que se siguió en el algoritmo de diagnóstico. Además de una tabla adicional donde se almacenará la información referente al médico para llevar un registro de usuarios.

V. DISEÑO DE LA SOLUCIÓN

El diseño de la solución se comenzó a hacer presente desde que se formuló la propuesta de trabajo. Se buscó la forma de emplear las practicas que nos llevaran a tener la mejor aplicación enfocada en la clase de cliente que estábamos ateniendo, es decir, médicos. Cuya necesidad es de una aplicación simple de usar, intuitiva y, sobre todo, que no añada más complejidad a su laburo diario.

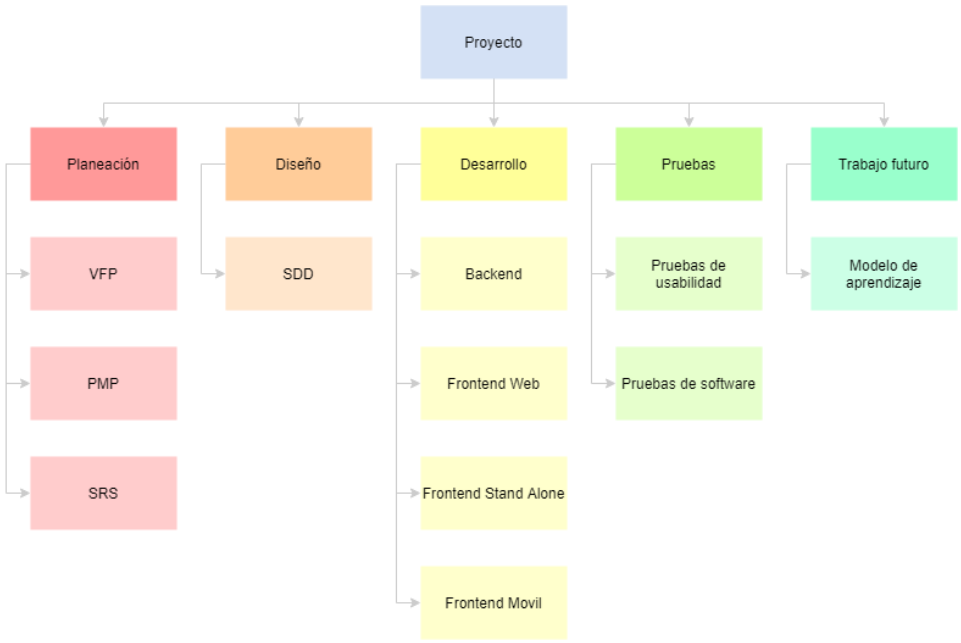
En la presente sección se hablará de los distintos documentos de diseño y sus componentes más relevantes para este apartado de la solución.

1. SPMP (Software Project Management Plan)

Primeramente, se desarrolló el plan de administración del proyecto de software. Allí se aclaró el manejo de cómo se iba a desarrollar el proyecto, como lo ilustra la siguiente figura:

Figura 1

WBS.



Aquí podemos observar las distintas etapas de desarrollo por las que iba pasó el proyecto (detallado en la sección VI). Principalmente, se definieron los distintos flujos de trabajo con los que se iba a operar, teniendo como principal objetivo el uso de metodologías agiles, modelos de cooperación y principalmente, el uso de buenas prácticas durante todo el proyecto.

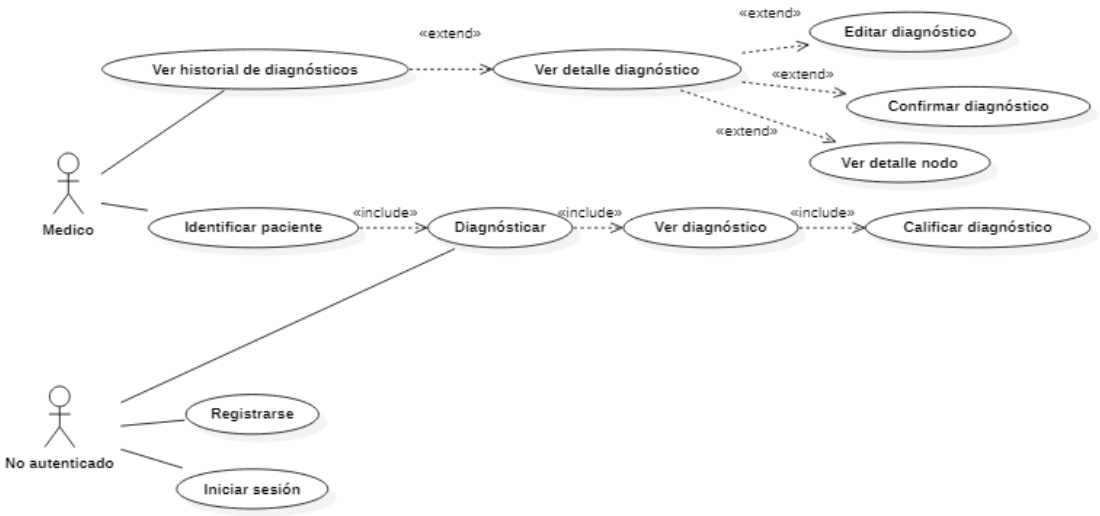
2. SRS (Software Requirement Specification)

Para la especificación de requerimientos, se empezó a considerar las tecnologías que iban a ser utilizadas durante el proyecto. Inicialmente se tuvo un planteamiento separado, donde se iban a hacer 3 desarrollos diferentes. Sin embargo, el descubrimiento de las *PWA* nos facilitó en gran medida el trabajo para agilizarlo y dar un proyecto mucho más completo.

En primer lugar, este documento nos dio la posibilidad de definir las funciones que podría ejecutar cada actor del sistema. Para este, tenemos específicamente dos actores: un usuario autenticado o médico y un usuario no autenticado. La siguiente figura ilustra estas funciones:

Figura 2

Diagrama de casos de uso



Por otro lado, es importante mencionar la tabla de las interfaces de software que cumplieron un importante rol al ser el principal factor para escoger las tecnologías que íbamos a usar. Esto se expone en la siguiente tabla:

Tabla 5. *Interfaces de Software*

Producto de Software	Angular*	Flutter*	MySQL	REST	JDBC	HTTPS
Propósito de Uso	Las últimas versiones de Angular proveen *service workers*	A partir de 2019 se instauró la beta de Flutter	Crear una base de datos segura que soporte	Planea lograr la conexión entre *front.end* y back-*end*. De esta forma	Acceso a la base de datos alojada en *cloud* por me-	Es importante su mención debido a que algunas PWA solo funcionan

	que son componentes que ayudan a la interpretación de la página creada en una PWA. A partir de esto podremos construir nuestra página con un modelo responsive que permite un buen desempeño en móviles sin necesidad de un navegador. Salvo el primer inicio. [2][8]	Web que permite, utilizando el mismo código y componentes de una aplicación móvil, desplegar una versión web. Así se pueden generar sencillamente aplicaciones con un soporte para PWA.	múltiples consultas y análisis con una capacidad de almacenamiento eficiente y una seguridad rígida. [7]	se logrará una conexión estable entre la base de datos y los servicios de la aplicación.	dio de manejadores de conexiones que implementan interfaces en lenguaje Java que se conecten con *Spring Boot* para la adición de filas dentro de la aplicación. [6]	cuando la conexión está cifrada, como por ejemplo las desarrolladas con Angular. Es decir, a la hora de hacer el despliegue es importante que tenga todos sus certificados SSL. [5]

Tomando en cuenta lo que se expone en la tabla, podemos ver que se contaba con múltiples opciones para las tecnologías a utilizar. En el caso del *front-end* se debatía cuál iba a ser el *framework* a utilizar. Entre Angular y Flutter. Sin embargo, se escogió Angular por dos razones principalmente: la familiaridad del equipo de desarrolladores con el *framework* y el poco conocimiento de programación orientada a *widgets* que maneja Flutter. Si bien las dos están ligadas al conocimiento del *framework*, era importante poder contar con factores que permitieran agilizar el proceso de apropiación.

Tabla 6 *Comparación entre Angular y Flutter [4]*.

Angular	**Flutter**
Usa lenguaje *typescript*	Usa lenguaje *dart*
Soporta sistemas operativos móviles y de computadora	Sólo soporta sistemas operativos móviles
Aplicaciones más lentas, comparativamente	Aplicaciones más rápidas

Mayor estabilidad en las apps	Menor estabilidad, comparativamente
Angular funciona como un bloque de la interfaz de usuario	Funciona como SDK
Posee servicios de construcción	Posee widgets específicos de construcción
Componentes: Type Components, Data Bindings e inyección de dependencias	Componentes: Flutter Engine, Librerías fundacionales y Plataforma de Dart

Nota: datos tomados del sitio web GfG

Por otro lado, las tecnologías que se usaron para el *back-end* sí cambiaron radicalmente. La aproximación inicial planteaba el uso del *framework* Spring para el *back-end* y *mySQL* para el almacenamiento. Principalmente por el conocimiento previo que tenía el equipo de este *framework* y la familiaridad que se tenía con Java. No obstante, hubo dos razones que llevaron al intercambio de conceptos:

1. Por recomendación del director, la data era mejor guardarla en una base de datos no relacional, donde pudiéramos tener mayor flexibilidad con los datos que se usaban y de lo que se iba a crear con ello. Por esto se escogió utilizar *Firebase*.

2. El uso de *Firebase* obligaba a tener presente al concepto de *Back-end as a Service* y utilizar Spring con *Firebase* podía ser un gasto de recursos innecesario puesto que Spring terminaría siendo un intermediario que solo estaría consumiendo recursos y ralentizando el sistema en general. Por ello se decidió hacer uso directamente de *Firebase*.

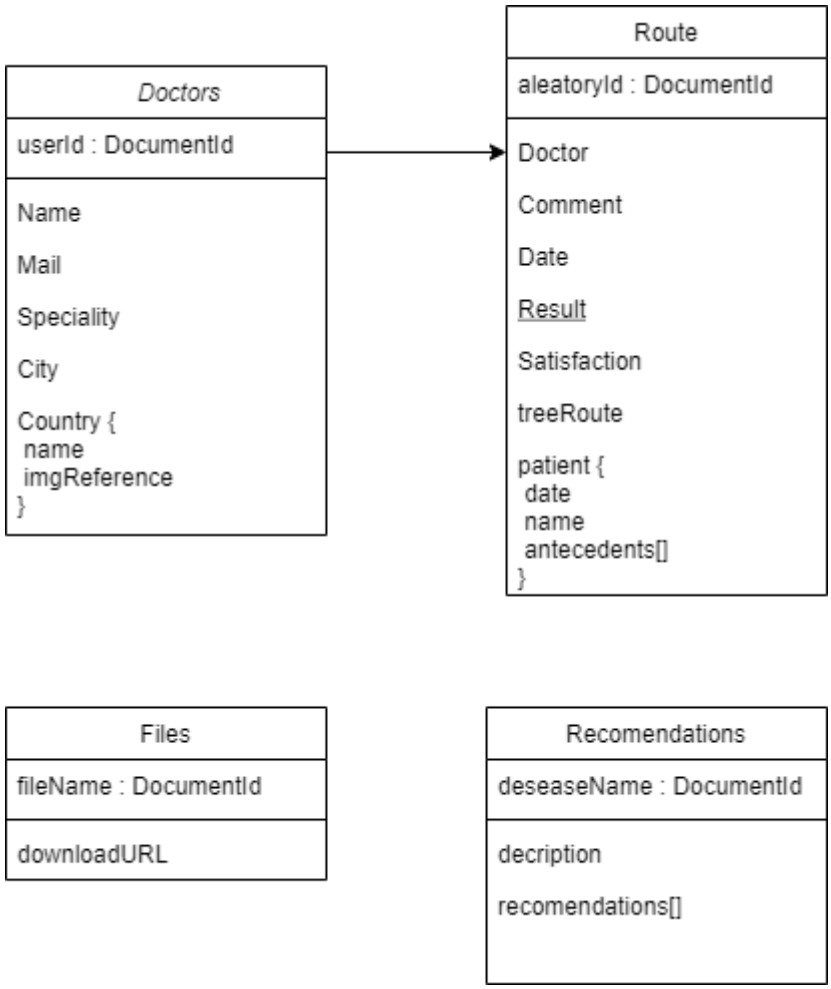
Por lo mencionado anteriormente, las tecnologías finalmente escogidas fueron Angular y Firebase. Conectándose a través de llamados de API REST con *AngularFire*.

Se usaron exactamente 3 servicios de Firebase:

1. **Firebase Authentication:** manejaba todo el sistema de autenticación a través de email y contraseña. Sin embargo, este no guarda los datos del médico, solo el usuario de autenticación.
2. **Firebase Datastore:** donde se guardan todas las colecciones de datos que componen a la aplicación. (se encuentran en detalle en la sección VI, fase 2)

Figura 3

Colecciones de Firebase



3. **Firebase Storage:** donde se almacenan todos los archivos anexos al diagnóstico. Estos se crean con un id único y son referenciados por un documento desde *Firebase Datastore*

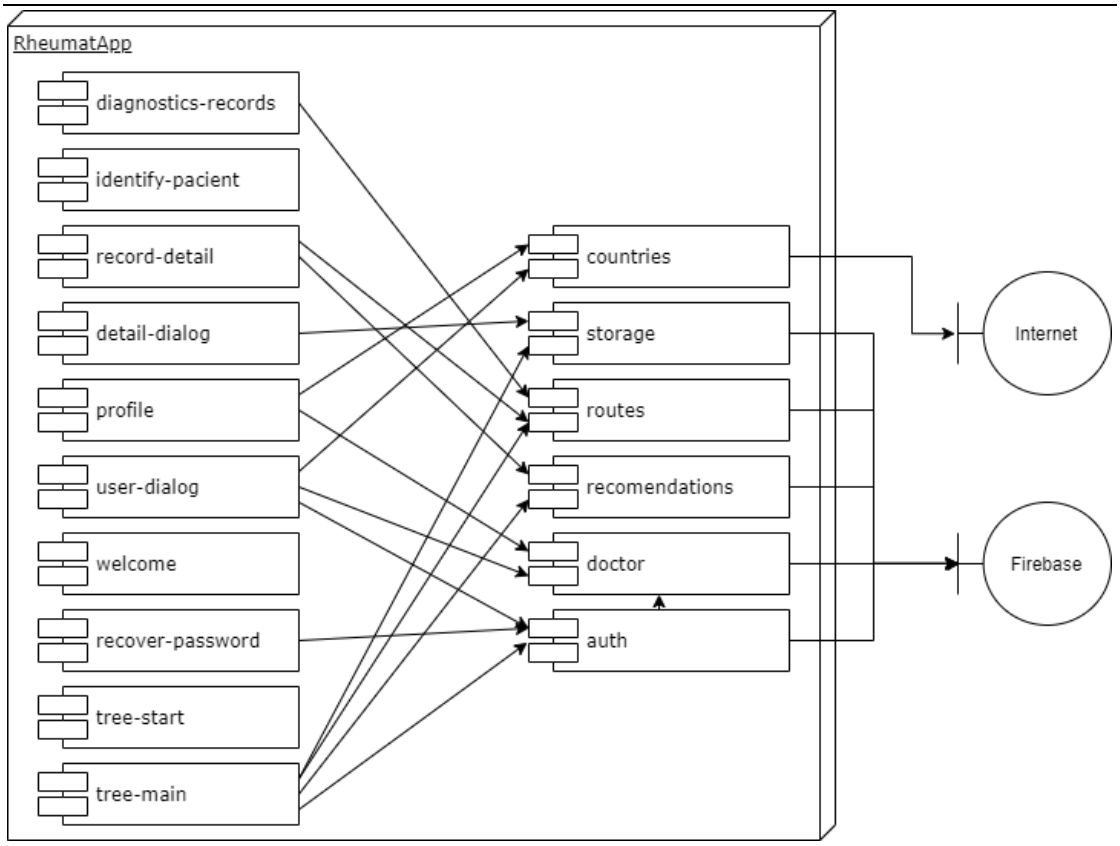
Con esta selección ya teníamos lo que podría ser la arquitectura de la aplicación resultante. No obstante, esta será detallada hasta el tercer documento de diseño, el SDD.

3. SDD (Software Design Description)

Finalmente, como el foco principal de la aplicación, aparte del diagnóstico, era estar disponible para uso con conexiones inestables y/o casi nulas, se diseñó una solución manteniendo una arquitectura de baja complejidad que permitiera acceder a todas las funcionalidades sin importar los problemas de red. Las siguientes figuras ilustran el despliegue y arquitectura del aplicativo.

Figura 4

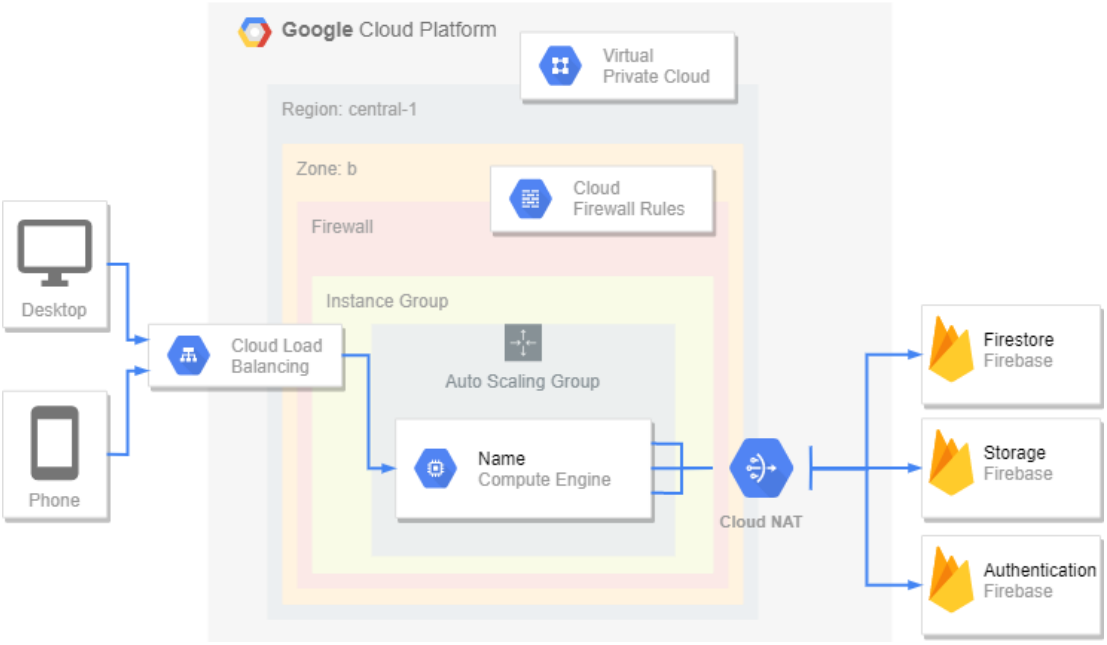
Diagrama de despliegue Rheumatapp.



Como se ve en la ilustración, el aplicativo fue percibido como un monolito con el fin de que este pudiese contenerse en sí misma y así poder proveer las funcionalidades sin conexión. La experiencia completa de la aplicación necesita de internet o al menos de una conexión parcial que permita la autenticación. En caso de tener una conexión inestable se puede autenticar y tener acceso a todas las funcionalidades del aplicativo gracias a la propiedad de *caching* de los *service workers* que permiten el uso de la PWA (véase sección VI, fase 2). Asimismo, la integración de Angular con *Firebase* llamada *AngularFire* permite persistir los nuevos registros con una conexión irregular ya que se crea un hilo independiente de la aplicación principal por cada registro que se intenta guardar. Esto permite que en algún momento estos hilos se puedan enviar; sin embargo, esto puede afectar el rendimiento en caso de no tener éxito con muchos registros.

Figura 5

Arquitectura Rheumatapp



Como lo ilustra la figura anterior, la solución fue probada y desplegada en Google Cloud Computing (GCP), pero la aplicación podría migrar a cualquier otro *Cloud Provider*.

Para los requisitos del sistema, se podría hacer en una maquina Linux o Windows, pero necesita de ciertas instalaciones. En cuanto a las características físicas se recomienda una maquina con 4 CPU y 16GB de memoria RAM. El nodo o contenedor que lo corra debe contar con Angular instalado, preferiblemente la versión 10.0.8, que fue la versión con la cual se creó el proyecto. No obstante, se podría configurar para usar con otra versión, pero tendrían que ajustarse algunas dependencias internas del proyecto. Asimismo, necesita de node.js instalado; en su versión de desarrollo se probó principalmente con las versiones 14.17.6, 16.9.1 y la más reciente, la 16.13.0.

Finalmente, se recomienda configurar un grupo de auto escalamiento que dependa de la carga que este recibiendo la aplicación. Esto, ya que no se sabe exactamente a cuántas personas, regiones u hospitales puede llegar nuestra aplicación por lo que el estrés bajo el que estará es desconocido hasta poder hacer un estudio de alcance formal.

Es importante aclarar que el despliegue NO hace parte de nuestra propuesta de trabajo. Se hace con fines ilustrativos para los jurados, el director y cualquier otro *stakeholder* que le interese.

VI. DESARROLLO DE LA SOLUCIÓN

Para iniciar con el desarrollo de la solución es indispensable retomar cuales fueron las metodologías propuestas para la ejecución del proyecto. El desarrollo del proyecto estuvo dividido en cuatro fases principales.

1. Planeación

En la fase de planeación se buscaba abordar temas de diseño, requisitos funcionales y no funcionales, entendimiento del problema y delegación de tareas. Se le dio mayor enfoque a una correcta disposición y separación de tareas para el desarrollo exitoso del proyecto. Para esta fase solamente se utilizó la metodología Kanban que propone la separación de tareas por niveles de importancia y estados de desarrollo [26]. La disposición del tablero Kanban se preparó de la siguiente manera:

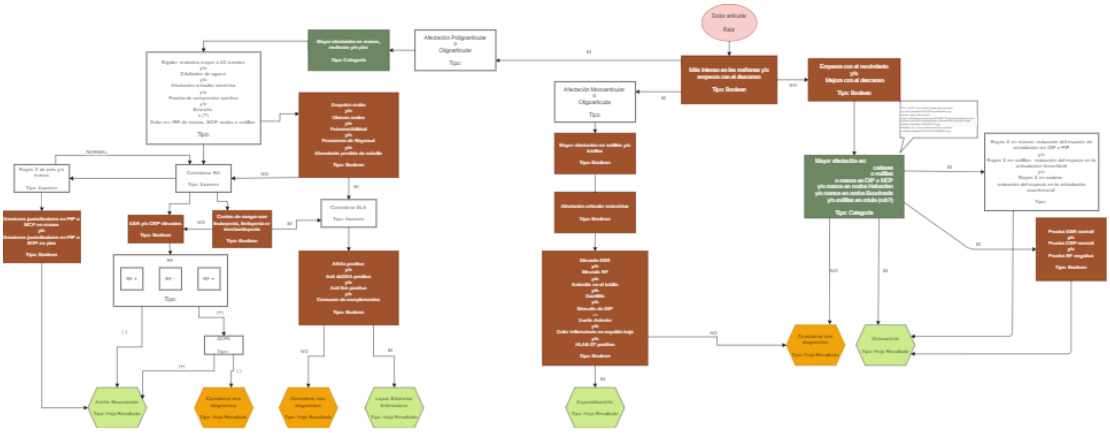
- *Backlog*: sección que almacenó todas las tareas que se pensaban resolver en la fase.
- *Sprint*: sección que almacena todas las tareas a desarrollar en los plazos pactados de sprint que aún no han sido asignadas por integrante.

- *To Do*: sección que almacena las tareas asignadas a cada integrante.
- *Doing*: sección que cuenta con las tareas que ya iniciaron su desarrollo
- *Verify*: sección que almacena las tareas que han sido terminadas y están a la espera de una verificación de calidad por el grupo.

Esta metodología y su disposición buscaban la resolución del primer objetivo específico planteado, ligado a entender el árbol y cómo se iba a implementar. Asimismo, se finalizaron los documentos iniciales de diseño y planeación; es decir, el documento SDD y el árbol de decisión que habíamos planteado inicialmente.

Figura 6

Versión inicial del árbol de decisión



La figura anterior muestra la primera abstracción del árbol propuesto por Fernández [44] que nosotros llevamos al diseño en la etapa de planeación. En esta se pueden observar distintos nodos en el gráfico, mayor definición en los anexos (véase anexo 9):

- El circulo rosado: representa el nodo raíz de donde proviene todo y es el dolor de articulaciones

- Los cuadrados: representa cada uno de los nodos decisión que encontramos del árbol.

- Los hexágonos: son los distintos nodos hoja que se pueden interpretar como diagnósticos. Los verdes representan una enfermedad y los amarillos representan un diagnóstico inconcluso.

- Las burbujas de texto: son indicadores que pusimos como grupo dentro del gráfico que tenían links para el entendimiento de conceptos avanzados de medicina.

Cada uno de los nodos estaba enumerado para poder incluirlo en lo que sería la implementación de la estructura de datos.

El resultado de la etapa, en general, fue exitoso, aunque esta se vio entrelazada con la etapa de validación puesto que para dar cierre a la sección necesitábamos del visto bueno del árbol que habíamos planteado con el Dr. Daniel Fernández. Sin embargo, las reuniones fueron escazas y no pudimos tener esa reunión hasta ya entrados en la etapa de validación.

2. Desarrollo

La fase de desarrollo tuvo como objetivo entregar la aplicación multiplataforma y la entrega de la implementación del árbol de decisión que habíamos abstraído. Para esta fase se utilizó de nuevo el tablero Kanban, XP y *Scrum*. En primer lugar, Kanban, como se puede evidenciar, se volvió una herramienta transversal para todo el desarrollo del proyecto donde utilizamos el mismo tablero expresado en la fase 1. En segundo, de *Xtreme Programming*, solo se hizo uso de su aproximación al concepto de *Pair Programming*. Finalmente, *Scrum* es la metodología principal que se utilizó durante esta fase, esta se implementó a partir de ciclos iterativos que iniciaban en la planeación

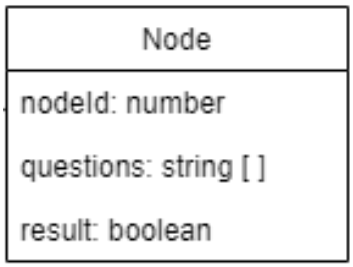
colectiva de las tareas que se iban a desarrollar durante el *sprint*. Estos ciclos se realizaron cada dos semanas con una fase de retroalimentación y revisión de los avances logrados. [24][25][26][27]

El primer punto que se buscó resolver fue la implementación del árbol de decisión que desarrollaríamos, inicialmente basado en el diagrama generado en la fase de planeación. Puesto que uno de los requerimientos funcionales que propusimos fue la persistencia y uso local de este árbol de decisión, la implementación se hizo en una sección aparte del *front-end*. Esta implementación usaba dos estructuras de datos por excelencia.

Primeramente, una estructura *Node* de la siguiente forma:

Figura 7

Estructura Nodo



Donde nodeID es el id del nodo con el que se identificó en el diagrama hecho. El arreglo questions son la o las preguntas que conforman al nodo. Finalmente, result es el veredicto con el que termina el nodo, verdadero o falso.

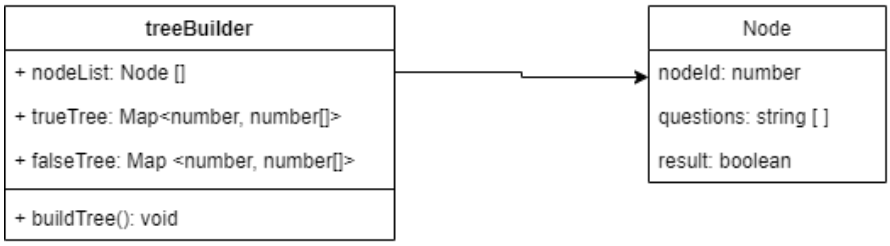
En segundo punto, una estructura TreeBuilder compuesta por 3 atributos principales.

1. **Construir Nodos:** donde se crea el arreglo de nodos, allí se asigna la relación entre los números y su correspondiente pregunta.
2. **Construir Árbol True**: Se asigna el o los valores a los que puede dirigirse un nodo en caso de una respuesta con *result* afirmativo.

3. **Construir Árbol False:** Se asignan el o los valores a los que puede dirigirse un nodo en caso de una respuesta con *result* negativo.

Figura 8

Estructura TreeBuilder



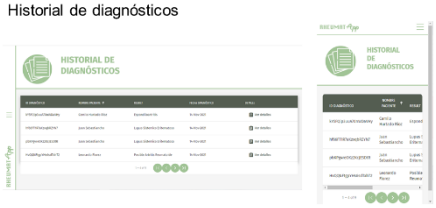
El segundo punto del desarrollo fue la elección de las tecnologías que iban a ser utilizadas en el proyecto (Véase sección V), donde finalmente encontramos que lo más prudente era hacer uso de herramientas con las que ya tuviéramos experiencia. Para el aspecto *front-end* escogimos hacer uso de Angular, un *framework* de programación orientada a componentes basado en *Typescript* y *Node.js.* En Angular se presentó la oportunidad de agilizar el proceso de desarrollo puesto que la mayoría de los integrantes ya tenían experiencia con él y, también, por su facilidad para hacer la conversión de una página web tradicional a una PWA (*Progressive Web App*) que se expone en detalle en la etapa de validación. En las siguientes figuras se encuentran las distintas páginas del aplicativo.

Nota: Puede encontrar las pantallas en el anexo 5 en el mismo orden que están expuestas.

Tabla 7 *Pantallas y funcionalidades del aplicativo*

Pantalla	Descripción
Presentación 	Página principal para aquel que no está autenticado en el aplicativo. Aquí puede decidir si ingresar a su usuario o registrar uno nuevo dentro del aplicativo. Asimismo, puede ingresar directamente al diagnóstico en caso de no poder iniciar sesión.
Inicio de sesión - Médico 	Inicio de sesión donde puede autenticarse con ingresar e-mail y contraseña.
Registro - Médico 	Registro de un nuevo usuario. En él, el médico crea un nuevo usuario donde se recolectarán datos de él como nombre, e-mail, nacionalidad, especialidad, entre otras.
Inicio 	Página principal para aquel que esta autenticado. Inicialmente está planteado para tener un rápido acceso a la sección de diagnóstico.

Barra de navegación 	Barra de navegación que permite acceder a las distintas secciones de la aplicación. Asimismo, se muestran los datos básicos del médico que esta autenticado en la sesión actual.
Identificación de paciente 	Página inicial del diagnóstico. Aquí se piden los datos del paciente importantes para el estudio. Por ello no se piden datos como la cédula. Se agrega una opción para la lectura de un archivo unificado de historial médico. Sin embargo, no tuvimos la información pertinente como para hacer la implementación de esta identificación por archivo.
Confirmación de paciente 	Página de confirmación de los datos ingresados en la pantalla anterior. Si se requiere modificar o eliminar algo de lo ingresado, hay acceso directo a través del botón regresar.
Diagnóstico I - Pregunta Y/N 	Primera página del diagnóstico formal. Aparece en los nodos donde se tiene una pregunta única binaria. Le ofrece la opción de responder sí y no. **Nota:** Esta, como las siguientes páginas de diagnóstico, poseen un botón para agregar un archivo donde el médico actual puede agregar evidencia de la respuesta a la pregunta en un archivo de cualquier tipo.

Diagnóstico II - Pregunta Multiple 	Segunda página del diagnóstico formal. Aparece cuando el nodo tiene múltiples preguntas; por lo cual él usuario debe escoger las que apliquen para el paciente que está tratando.
Diagnóstico III - Multiple opción 	Tercera página del diagnóstico formal. Se obtiene cuando la respuesta de un nodo desemboca en múltiples nodos u opciones. Estos se ven reflejados como pestañas en la pantalla actual.
Diagnóstico IV - Diagnóstico 	Cuarta página del diagnóstico formal. Se obtiene cuando se llega a alguno de los nodos hoja. Aquí podemos ver la información del paciente, información relacionada al diagnóstico como descripción y recomendaciones y una sección de reseña para que el médico califique y comente su experiencia con el aplicativo.
Historial de diagnósticos 	Accedido desde la barra de navegación. El historial de diagnósticos permite observar cada uno de los diagnósticos que se han realizado hasta la fecha. En él se puede ordenar por nombre, fecha o diagnóstico final. Con el paginador se pueden ver 4 resultados por página.

Detallado de diagnóstico - I 	Al escoger un diagnóstico del historial de navegación podemos ver, en un mapa que simboliza el grafo, todos los detalles de los nodos por los que paso y el resultado final que tuvo cada uno de estos nodos junto con la reseña que le había dado el médico en el diagnóstico inicial. Por otro lado, se agrega una nueva pregunta a la sección de reseña donde se busca saber si el diagnóstico resultante fue exitoso, fracaso o si aún no se confirma (selección predeterminada).
Detallado de diagnóstico - II 	Se añade esta exposición para mostrar el resultado responsive donde cambia la disposición para poder ver todo el mapa en la sección superior.
Detallado de diagnóstico - III 	Cuando se escoge un nodo para ver sus detalles obtenemos un modal que nos muestra una pestaña de preguntas donde podemos ver que preguntas se hicieron en dicho nodo y cuáles fueron las respuestas ingresadas.
Detallado de diagnóstico - IV 	Asimismo, el modal muestra otra pestaña en el caso donde en ese nodo se haya ingresado algún archivo. De ser así la nueva pestaña contiene un botón que lo redirige a una opción donde se encuentra el archivo que fue ingresado. Si es observable se verá en una

	pestaña nueva; de no serlo, se le da la opción de descargar el archivo.
Editar perfil - Médico 	Accedida desde la barra de navegación. Es la página donde el usuario puede cambiar la información que allí había consignado. En el caso de la contraseña se le envía un correo para que reinicie su contraseña

A partir del diagrama de navegación que se encuentra en el SDD (véase anexo 8). Para los diseños se hizo un estudio de UI/UX con el fin de sacar la abstracción más amigable con el usuario. Toda la iconografía, tipografía y prototipos de diseño pueden encontrarse en los anexos (véase anexo 7) o en el repositorio del código fuente puede encontrar tipografía e iconografía dentro de la carpeta *assets*. Sin embargo, hubo un grupo de recursos que si fue tomado de internet; la carpeta *countries* que contiene las banderas de todos los países del mundo, utilizado en la parte de identificación de la nacionalidad. [3]

El tercer punto del desarrollo a tener en consideración fue la conexión con el *back-end* de la aplicación (véase sección V). *Firebase* estaba encargado de los temas de autenticación, persistencia de datos de los usuarios, persistencia de las rutas y, finalmente, persistencia de los archivos anexos. Se instauraron 4 colecciones:

- ***Doctors:*** guarda la información de los usuarios.
- ***Files:*** guarda la referencia y la ruta de los archivos que se han introducido en un diagnóstico.
- ***Recommendations:*** guarda la descripción y recomendaciones propuestas por cada uno de los diagnósticos.
- ***Routes:*** guarda la información de la ruta que tomó durante el diagnóstico.

Sin embargo, las estructuras persistidas y archivos serán referenciados en detalle en la etapa de preparación para trabajo futuro.

Finalmente, una vez terminado el desarrollo *front-end* y su integración con el *back-end* se hizo la conversión a PWA. Angular facilita mucho esto a partir del uso de los denominados service workers. Estos, según la página oficial de Angular, son scripts que corren en el navegador web y aplican un sistema de *caching* a la aplicación [2]. Por consiguiente, no sólo habilita la aplicación multiplataforma, sino que facilita el proceso de *caching* para los casos donde la conexión sea baja o incluso nula. Vale la pena resaltar que inicialmente sí es necesaria una conexión estable o mínima para poder descargar e ingresar a la aplicación.

Es importante aclarar que una de las ventajas especiales que nos otorga el uso de *Firebase* es que si la conexión es baja o nula no va a ser posible guardar el diagnóstico directamente en la base de datos. Sin embargo, sus servicios crean hilos independientes al flujo principal de la aplicación. Esto, junto a las propiedades de *caching* de la PWA, permiten un uso parcial de la aplicación sin necesidad de conectividad. Aunque igual si es necesaria la conectividad puesto que estos *features* no estarán disponibles sin antes ser autenticados.

Finalmente, el resultado de la etapa fue exitoso. Para este punto sólo se contaba con un impedimento y era el correcto funcionamiento del árbol puesto que no se había tenido la validación y en algunos nodos no teníamos su camino en respuesta afirmativa o negativa.

3. Validación

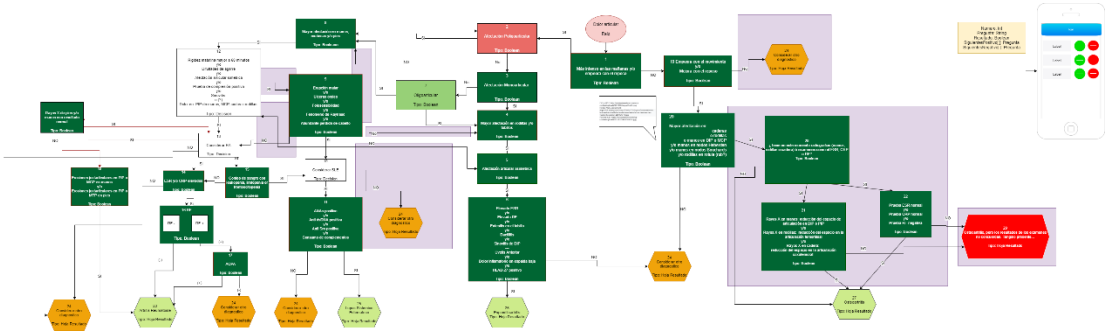
La fase de validación tuvo como objetivo validar la satisfacción de lo implementado y lo que estaba planteado hasta el momento. Principalmente, se buscaba la validación del diseño del árbol de decisión propuesto con el Dr. Daniel Fernández. Para esta etapa se utilizaron dos metodologías *Design Sprint* y *Kanban*. Para el *Design Sprint* se utilizó

la fase cinco que corresponde a la validación donde empleamos distintas técnicas de validación. [23][26]

En esta etapa el mayor cambio lo evidenció el árbol de decisión que habíamos planteado. Luego de la primera reunión con Daniel, se identificaron falencias en las relaciones entre nodos y se completaron las conexiones restantes. Principalmente, la falencia estaba en el reconocimiento de ciertas reglas mutuamente excluyentes que cambiarían el destino del nodo. Por ejemplo, si se pregunta: ¿la afectación articular es asimétrica? Y responde de forma negativa, se debe desplazar directamente al nodo con la respuesta de una afectación articular simétrica.

Figura 9

Diseño final del árbol de decisión



La figura anterior muestra la abstracción final que se obtuvo con la etapa de validación (véase el anexo 10). Las novedades se ven evidenciadas en los cuadros morados que muestran las secciones que sufrieron cambios a causa de esta etapa. Como se puede evidenciar cada uno de los nodos resulto con una respuesta afirmativa y una negativa para que no exista nodo sin cubrir. El nodo rojo es un caso especial opcional si posee ciertos exámenes requeridos. Este caso indica que se diagnostica una osteoartritis. Sin embargo, los resultados de los exámenes que introdujo parecen tener alguna incongruencia con el diagnóstico.

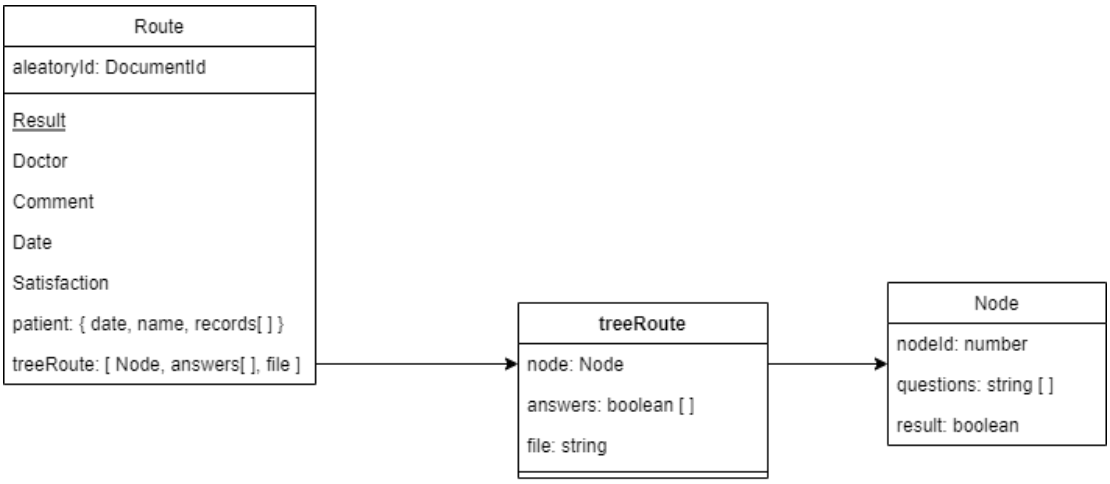
Todo esto llevo culminar la etapa de manera exitosa. El paso a seguir fue un pequeño despliegue de la aplicación para poder mostrar la funcionalidad completa, finalizando la etapa de validación.

4. Preparación para trabajo futuro

Para finalizar el desarrollo, se ejecutó la última fase propuesta donde se buscó dejar una proyección o punto de partida para la futura aplicación de inteligencia artificial con los datos que se almacenan de los diagnósticos. Principalmente tomando en cuenta la colección ***Routes*** que es la que cuenta con las rutas de los diagnósticos. Para ello analizaremos la estructura propuesta de cada ruta:

Figura 10

Estructura documento ruta en Firebase.



La anterior es la estructura que usa un registro de ruta. La mayoría de los campos se deducen fácilmente. Sin embargo, el atributo *treeRoute* posee una estructura particular donde estamos guardando el arreglo de los nodos por los que pasó. Por cada nodo tenemos 3 atributos: nodo que es de objeto *Node* por el que pasó con su *result* actualizado, respuestas que son los resultados de cada una de las preguntas que posee el nodo y

archivo donde encontramos la referencia al archivo que se agregó en ese nodo. Por ello, termina siendo este el núcleo final de la investigación y el detonante para los pasos a seguir.

Por ende, buscando modelar e implementar el sistema de inteligencia artificial inicialmente propuesto se deben abordar los siguientes frentes:

1. **Anonimidad**: la base de datos esta propuesta para ser de fácil abstracción a la hora de revisar el historial de diagnóstico desde la perspectiva de médico. Sin embargo, para poder hacer el estudio de datos se debe de tener una data que no pueda rastrear a ninguno de los sujetos implicados. Esto implicaría sacrificar el nombre del paciente y su fecha de nacimiento. Esta última debería ser utilizada como una variable derivada de edad.

2. **Historias clínicas:** una de las funcionalidades que se querían implementar era la abstracción de los datos del paciente a partir de su historia clínica. Principalmente por dos razones: 1) Facilitar la identificación del paciente al abstraer los datos desde una historia clínica sin tener que duplicarlos manualmente y 2) para poder almacenar estas historias clínicas. Sin embargo, la falta de acceso a esos recursos privados no nos permitió implementarla; por ello, aparece en el aplicativo como una opción aún no implementada.

3. **Aproximaciones con los datos:** la data que se recolectó principalmente tenía el fin de no ser totalmente estructurada para darle flexibilidad al estudio posterior. Por ello podemos encontrar la data estructurada en la forma cómo se está guardando las rutas en la base de datos y data no estructurada como las archivos que se agregan por cada uno de los nodos. Esto permite poder hacer estudios con distintos tipos de técnicas. Para la data estructurada se podría implementar desde *Clustering,* a partir de los diagnósticos, hasta predicciones con

las relaciones entre edad, antecedentes y diagnóstico. Para la data no estructurada se podrían hasta hacer estudios con redes neuronales convolucionales que permitan encontrar patrones en los distintos archivos ingresados; separándolos ya sea por nodo, tipo de archivo o diagnóstico final.

En conclusión, la culminación del proyecto se catalogó como exitosa puesto que se lograron los objetivos propuestos durante la planeación. Es importante recalcar que esto puede crear oportunidades para trabajos muy novedosos con avances increíbles en el campo de la reumatología y del software aplicado a la medicina. Asimismo, el impacto social puede llegar a ser enorme, pero esto será detallado más adelante (véase sección VIII).

VII. RESULTADOS

1. Control y seguimiento del desarrollo

Para el proceso de control de calidad del desarrollo, de todas las actividades, se utilizó un tablero correspondiente a la metodología Kanban, dividido en 4 categorías (*Backlog, To Do, Doing* y *Verify*), tal como se explicó en la sección Planeación, lo cual permitió la creación de tareas o "tickets" como responsabilidades de cada uno de los integrantes del equipo, siendo posible organizarlos en las distintas etapas mencionadas, todo esto, con el fin de acrecentar la trazabilidad de las mismas, y por ende del proyecto. Así, a lo largo de la fase de desarrollo de código y posterior documentación, fueron realizadas 17 tareas principales.

2. Pruebas de desarrollo

Para la ejecución de las pruebas de desarrollo, se tuvieron en cuenta pruebas automatizadas punta a punta o *'end to end'* las cuales, involucran verificar el flujo de ejecución de la aplicación de comiendo a fin. Es decir, los escenarios puestos a prueba combinan

diferentes funcionalidades, con el fin de emular el comportamiento de un usuario final. [49]

En adición a esto, también fueron desarrolladas pruebas automatizadas de interfaz (*UI*), donde el objetivo es la detección de errores propios del *layout* de la aplicación, así como también errores en la lógica de presentación, responsividad o inclusive ortografía [50], centrándose en el *'front-end'* de la aplicación, ejecutando pruebas de integración entre los diferentes componentes visuales. Un desglose completo de lo que se pretende evaluar, puede ser encontrado a detalle en el Plan de pruebas de software de la aplicación (Véase anexo 3).

3. Pruebas de Aceptación

En lo que a las pruebas de aceptación respecta, éstas, fueron llevadas a cabo paulatinamente cada vez que fue puesto en producción un cambio significativo en el desarrollo, siendo ejecutadas a lo largo de dos hitos en la línea de tiempo de desarrollo del proyecto: Las reuniones con el director del trabajo de grado, el profesor Leonardo Flórez Valencia; en donde iterativamente se validaba el diseño y construcción del 'front-end' del sistema y las reuniones que se tuvieron con el asesor, el Dr. Daniel Fernández Ávila, donde fue presentado un prototipo funcional, con el fin de reunir retroalimentación y eventualmente la aprobación. Posterior a esto, el equipo de trabajo realizó pruebas manuales con el fin de alimentar información al sistema, y a su vez validar que se cumplieran los requerimientos de la interfaz, como por ejemplo un aceptable diseño responsive, probando la aplicación desde distintos dispositivos (móvil, tablet, PC, etc.)

4. Análisis de resultados de las pruebas

Para el análisis de los resultados de las pruebas ejecutadas, se generaron reportes de dos tipos:

- Para el equipo de trabajo, se generaron reportes en consola después de la ejecución de las diferentes *'features'* como el presentado a continuación, con el fin de presentar un resumen de la ejecución:

Figura 11

Reporte de ejecución de pruebas hecho para consola

```
    √ Verify login page loads correctly
[13:32:39] W/element - more than one element found for locator By(
[13:32:39] W/element - more than one element found for locator By(
[13:32:39] W/element - more than one element found for locator By(
[16872:15544:1116/133241.418:ERROR:chrome_browser_main_extra_parts
 no report that this ends.
[16872:12316:1116/133241.423:ERROR:device_event_log_impl.cc(214)]
[16872:15544:1116/133241.423:ERROR:chrome_browser_main_extra_parts
[16872:15544:1116/133241.479:ERROR:chrome_browser_main_extra_parts
 no report that this ends.
[16872:15544:1116/133241.514:ERROR:chrome_browser_main_extra_parts
    √ Verify register fields are displayed
[13:32:56] W/element - more than one element found for locator By(
[13:32:56] W/element - more than one element found for locator By(
[13:32:56] W/element - more than one element found for locator By(
    √ Verify login fields are displayed

Executed 3 of 3 specs SUCCESS in 24 secs.
[13:33:00] I/launcher - 0 instance(s) of WebDriver still running
[13:33:00] I/launcher - chrome #01 passed
```

- Para este documento y a quien interese, fue generado un reporte con mayor ayuda visual a través de la herramienta de reporte de pruebas Allure, como, por ejemplo, se evidencian en las siguientes ilustraciones:

Figura 12

Reporte de ejecución de pruebas hecho con Allure

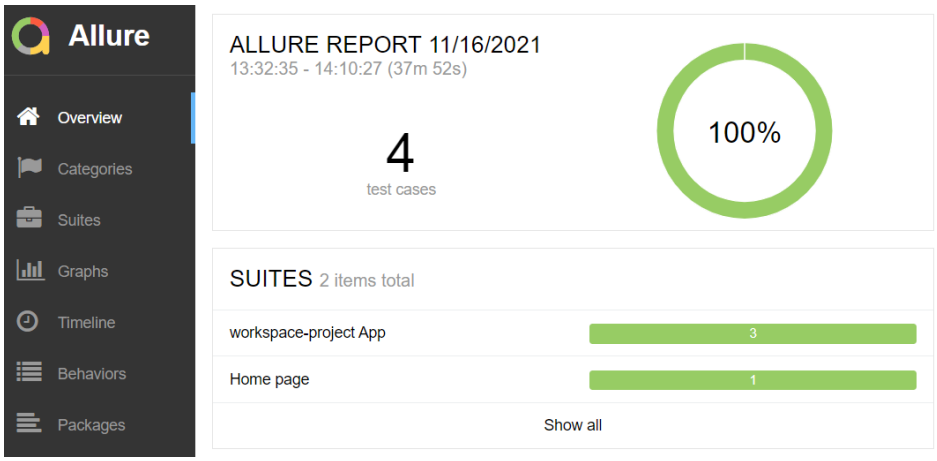
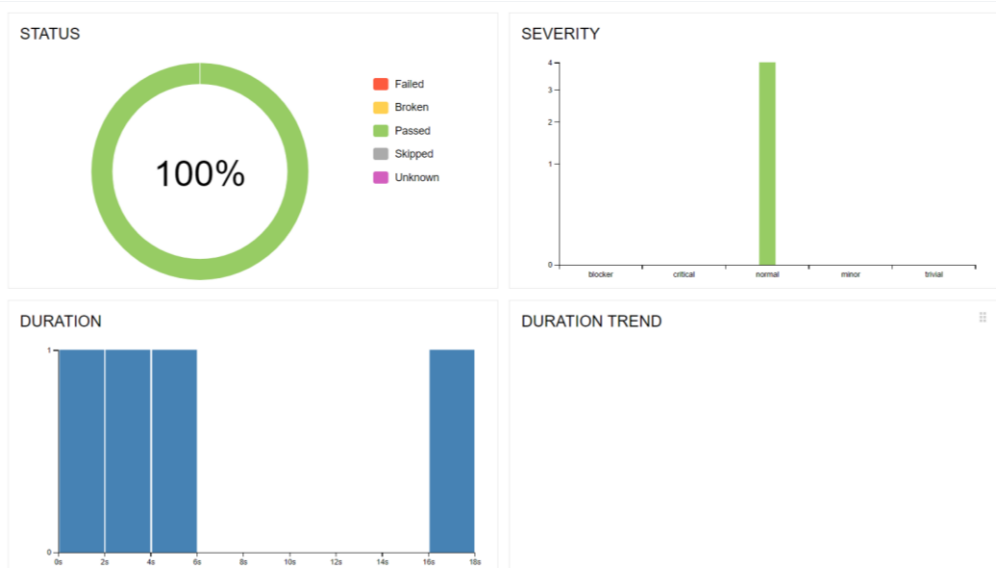


Figura 13

Reporte de ejecución de pruebas hecho con Allure



Una vez realizadas las pruebas descritas anteriormente, una revisión de los resultados obtenidos arroja el cumplimiento de los criterios de aceptación, descritos en el Plan de pruebas de software (Véase anexo 3). De igual manera, dichos reportes, donde se evidencia el resultado de la ejecución de las pruebas automatizadas con mayor detalle, pueden encontrarse en el anexo 4: reportes control de calidad.

VIII. CONCLUSIONES

1. Análisis de impacto del proyecto

Las enfermedades reumáticas han sido un problema grave de salud debido al potencial daño articular y la disminución de la calidad de vida que pueden generar en sus afectados. Sobre todo, en casos donde no son diagnosticadas rápidamente. Es muy importante lograr un buen diagnóstico para empezar un tratamiento oportuno lo antes posible para controlar la enfermedad y evitar daños futuros.

En Colombia, actualmente, se presenta un grave déficit de médicos reumatólogos, esto causa que la gran mayoría de pacientes que ingresan a un centro de atención por dolores articulares sean atendidos en primera instancia por médicos generales [13]. Lo anterior, debido a que en nuestro sistema de salud el paciente no puede ir directamente con un reumatólogo, sino que debe ser remitido por un médico general. Lo perjudicial de este sistema es que el diagnóstico brindado por el médico general, a veces, es erróneo [10]. Principalmente, cuando la consulta es por dolor articular. Por esto es necesario el desarrollo de una aplicación que apoye a los médicos generales a la hora de realizar sus diagnósticos y poder brindar una rápida y mejor atención a los pacientes.[44]

Por eso, el propósito general de esta solución es mejorar los diagnósticos de los médicos generales en reumatología al ofrecerles una aplicación de fácil uso. Con ella, podrán recibir apoyo en el proceso de diagnóstico de las enfermedades y, de tal modo, logren reducir la cantidad de diagnósticos erróneos. Teniendo en cuenta esto, a corto plazo se busca mejorar el porcentaje de diagnósticos correctos de enfermedades reumatológicas a pacientes que ingresen por dolores articulares. Esto permitirá, no sólo mejorar los diagnósticos y la confianza de los pacientes en la consulta general, sino que podremos lograr mejorar su calidad de vida.

Por otro lado, a corto plazo podríamos brindar un apoyo a las comunidades donde una conexión estable a internet es un lujo. Podremos brindarles un servicio, así sea mínimo. Esto principalmente a que los médicos que se encuentran en las zonas más alejadas, normalmente, no tienen una especialidad. Son residentes o médicos generales que carecen de las habilidades que tendría un especialista. Esto les podría dar la oportunidad de priorizar a los pacientes más afectados, de poder enviar a las ciudades a aquellos que realmente lo necesitan y, finalmente, ahorrarles tiempo a los pacientes que terminan yendo con sus propios medios al municipio o metrópoli más cercana con servicios médicos medianamente aceptables.

Complementando lo anterior, como se puede ver en los anexos 11, 12 y 13, la muestra seleccionada de la población médica considera que la solución propuesta podría apoyarlos en el diagnostico de las enfermedades reumatológicas, además la facilidad de uso de la interfaz gráfica presentada es catalogada como media alta, alta y muy alta. Según los resultados obtenidos se espera una buena acogida dado que se ve como una solución a un problema y la complejidad de su uso es baja.

A mediano plazo se espera que la aplicación haya ayudado a mejorar la calidad de vida de una gran cantidad de pacientes que sean diagnosticados rápida y acertadamente gracias a nuestra solución. Además, se buscará que sea utilizada en varios hospitales a lo largo del país y se tenga una base de datos poblada respecto a todos los diagnósticos realizados. Con esta base de datos se pueden iniciar los procesos de modelado y preparación de la data que fueron mencionados en la sección VI.

En cuanto al largo plazo, se espera que el proyecto haya continuado con las bases que fueron dejadas y se hayan implementado mejoras a la aplicación, con modelos de analítica de datos y de aprendizaje de máquina. Y así mejorar la confiabilidad del modelo y tener la posibilidad de ayudar con el diagnostico de aún más enfermedades distintas a las planteadas al principio del proyecto. Todo lo anterior en base a los datos recolectados.

Todo lo mencionado en este documento puede ser un avance de vital importancia para la integración de la medicina con los sistemas inteligentes puesto que, al menos en Colombia, no se había hecho nada con las condiciones y ambición que supone este proyecto. Por ahora, este proyecto está motivado por la Asociación Colombiana de Reumatología. Esto supone que, si el presente obtiene un buen resultado, puede presentar un punto de partida para la realización de otros proyectos de esta índole producidos en masa con otras asociaciones de distintas especialidades. Todo lo anteriormente explorado podría significar un avance increíble para la aplicación de tecnologías en medicina y para la ciencia en general en el país.

2. Conclusiones y trabajo a futuro

A manera de conclusión, se logró el objetivo general que era una aplicación multiplataforma que nos permitiera comunicar y compartir el algoritmo planteado por el Dr. Daniel Fernández. Con las pruebas que hemos visto hasta el momento, es casi un hecho que este aplicativo puede apoyar y validar el diagnóstico de un paciente con dolores articulares. Se lograron analizar los distintos flujos que podría tener el árbol de decisión, se pudo implementar el algoritmo, se creó la aplicación multiplataforma y logramos validar la satisfacción del creador del algoritmo con lo que habíamos implementado. Básicamente, si exploramos los objetivos punto por punto, podemos ver que fue posible cumplirlos a cabalidad. Sin embargo, más allá de un proyecto solamente enfocado en software, se convirtió en un proyecto con una connotación social que realmente se volvió otro de nuestros objetivos a resolver.

También, se dejaron los puntos de partida para un trabajo futuro en detalle (ver sección VI). Más allá de eso, concluimos que el alcance del aplicativo puede ser más grande de lo que se planteó inicialmente. Puede brindarle facilidades, bienestar, confianza y muchos otros dotes a pacientes que se ven tan afectados con estas enfermedades hoy en día.

Por ello, el sentido de todo este plan es poder sentar las bases de un proyecto mucho mayor y, ello, nos hizo poner las cosas en perspectiva. Intentar anticiparnos a los distintos escenarios que podrían ocurrir, tratar de hacer frente a todos los problemas de usuario que podrían aparecer, poder brindarles servicio en cualquier lugar, entre otros factores más, son algunos de los puntos que constantemente cruzaban nuestros cientos de líneas de código o nuestros documentos de diseño con el fin de poder encontrarles una solución y, es ello lo que lleva a concluir que hasta este punto el proyecto ha sido un éxito.

IX. REFERENCIAS

[1]
S. rural, "Ir al médico, toda una odisea en la Colombia rural", *Semana rural*. https://semanarural.com/web/articulo/ir-al-medico-toda-una-odisea-en-la-colombia-rural/527 (consultado nov. 16, 2021).

[2]
"Angular". https://angular.io/guide/service-worker-intro (consultado nov. 16, 2021).

[3]
"Download all country flags of the world for free". https://flagpedia.net/download (consultado nov. 16, 2021).

[4]
"Difference between Flutter and Angular", *GeeksforGeeks*, ago. 09, 2021. https://www.geeksforgeeks.org/difference-between-flutter-and-angular/ (consultado nov. 16, 2021).

[5]
"📄 HTTP Documentation", *IETF HTTP Working Group*. https://httpwg.org/specs/ (consultado nov. 16, 2021).

[6]
"Controladores JDBC | Oracle Colombia". https://www.oracle.com/co/database/technologies/appdev/jdbc.html (consultado nov. 16, 2021).

[7]
"MySQL :: MySQL Documentation". https://dev.mysql.com/doc/ (consultado nov. 16, 2021).

[8]
"Angular". https://angular.io/ (consultado nov. 16, 2021).

[9]
14:00-17:00, "ISO/IEC/IEEE 29119-3:2013", *ISO*. https://www.iso.org/cms/render/live/en/sites/isoorg/contents/data/standard/05/67/56737.html (consultado may 07, 2021).

[10]
D. G. Fernández Ávila, "Simulación Clínica en Reumatología", Universidad Javeriana, 2015.

[11]
C. Mora Karam, A. Beltrán, J. Restrepo, R. Sierra, Y. A. Guerrero, y D. C. Martínez, "Educación y reumatología en el pregrado: ¿enseñamos suficiente?", *Revista Colombiana de Reumatología*, p. S0121812321000190, 2021, doi: 10.1016/j.rcreu.2020.11.006.

[12]
R. C. Martin, Ed., *Clean code: a handbook of agile software craftsmanship*. Upper Saddle River, NJ: Prentice Hall, 2009.

[13]
L. FM, “Colombia afronta un déficit de médicos reumatólogos”, oct. 23, 2018. http://www.lafm.com.co/salud/colombia-afronta-un-deficit-de-medicos-reumatologos (consultado may 07, 2021).

[14]
“IEEE/ISO/IEC Systems and software engineering – Requirements for acquirers and suppliers of user documentation”, *ISO/IEC/IEEE 26512 First Edition, 2011-06-01*, pp. 1–48, jun. 2011, doi: 10.1109/IEEESTD.2011.5871663.

[15]
“ISO/IEC/IEEE International Standard - Systems and software engineering – Life cycle processes –Requirements engineering”, *ISO/IEC/IEEE 29148:2011(E)*, pp. 1–94, dic. 2011, doi: 10.1109/IEEESTD.2011.6146379.

[16]
“ISO/IEC/IEEE International Standard - Systems and Software Engineering–Life Cycle Processes–Project Management”, *ISO/IEC/IEEE 16326 (First edition 2009-12-15)*, pp. 1–32, dic. 2009, doi: 10.1109/IEEESTD.2009.5372630.

[17]
F. Valdivia Navarro y D. Pérez Rosa, “Sistema de soporte a las decisiones clínicas relacionadas con el diagnóstico precoz de enfermedades”, *Revista Cubana de Informática Médica*, vol. 8, pp. 533–544, 2016, Consultado: may 07, 2021. [En línea]. Disponible en: http://scielo.sld.cu/scielo.php?script=sci_abstract&pid=S1684-18592016000300006&lng=es&nrm=iso&tlng=es.

[18]
“Lupus eritematoso sistémico (LES)”. https://www.fesemi.org/informacion-pacientes/conozca-mejor-su-enfermedad/lupus-eritematoso-sistemico-les (consultado may 07, 2021).

[19]
A. Jauregui, “REUMAPP, la aplicación para los pacientes reumatológicos del Hospital de Clínicas”. https://www.med.una.py/index.php/hospital-hc/noticias-del-hospital/2186-reumapp,-la-aplicaci%C3%B3n-para-los-pacientes-reumatol%C3%B3gicos-del-hospital-de-cl%C3%ADnicas (consultado may 07, 2021).

[20]
“ACR Mobile Apps”. https://www.rheumatology.org/Learning-Center/Apps (consultado may 07, 2021).

[21]
“RheumaHelper”. / (consultado may 07, 2021).

[22]
“IBM Docs”, mar. 02, 2021. https://prod.ibmdocs-production-dal-6099123ce774e592a519d7c33db8265e-0000.us-south.containers.appdomain.cloud/docs/es/spss-modeler/SaaS?topic=dm-crisp-help-overview (consultado may 07, 2021).

[23]

trendig, “Design Sprint, Google Ventures’ method of innovation in a team”. https://designsprint.org (consultado may 07, 2021).

[24] “When should I use Extreme Programming”. http://www.extremeprogramming.org/when.html (consultado may 07, 2021).

[25] “What is Extreme Programming (XP)?”, *Agile Alliance* |, jun. 14, 2017. https://www.agilealliance.org/glossary/xp/ (consultado may 07, 2021).

[26] Atlassian, “Kanban - A brief introduction”, *Atlassian*. https://www.atlassian.com/agile/kanban (consultado may 07, 2021).

[27] “2016-Scrum-Guide-Spanish.pdf”. Consultado: may 07, 2021. [En línea]. Disponible en: https://scrumguides.org/docs/scrumguide/v2016/2016-Scrum-Guide-Spanish.pdf#zoom=100.

[28] “IEEE Recommended Practice for Software Requirements Specifications”, *IEEE Std 830-1998*, pp. 1–40, oct. 1998, doi: 10.1109/IEEESTD.1998.88286.

[29] “IEEE Standard for Software Project Management Plans”, *IEEE Std 1058-1998*, pp. 1–28, dic. 1998, doi: 10.1109/IEEESTD.1998.88822.

[30] “ISO/TC 215 - Health informatics”, *iTeh Standards Store*. https://standards.iteh.ai/catalog/tc/iso/4eda7dbc-8b17-4896-843f-4e68c6dc19db/iso-tc-215 (consultado may 07, 2021).

[31] 14:00-17:00, “ISO/IEC 20246:2017”, *ISO*. https://www.iso.org/cms/render/live/en/sites/isoorg/contents/data/standard/06/74/67407.html (consultado may 07, 2021).

[32] 14:00-17:00, “ISO/IEC/IEEE 29119-2:2013”, *ISO*. https://www.iso.org/cms/render/live/en/sites/isoorg/contents/data/standard/05/67/56736.html (consultado may 07, 2021).

[33] “What is UML | Unified Modeling Language”. https://www.uml.org/what-is-uml.htm (consultado may 07, 2021).

[34] “Business Process Model and Notation (BPMN), Version 2.0”, p. 538.

[35] “About the Business Process Model And Notation Specification Version 2.0”. https://www.omg.org/spec/BPMN/2.0/ (consultado may 07, 2021).

[36] “Osteoartritis: MedlinePlus enciclopedia médica”. https://medlineplus.gov/spanish/ency/article/000423.htm (consultado may 07, 2021).

[37]
"Base de datos : ¿qué tipos hay y cómo funciona conectada a un software?", *TIC Portal*, jul. 09, 2019. https://www.ticportal.es/glosario-tic/base-datos-database (consultado may 07, 2021).

[38]
"The GNU General Public License v3.0 - GNU Project - Free Software Foundation". http://www.gnu.org/licenses/gpl-3.0.html (consultado may 06, 2021).

[39]
"Los derechos patrimoniales", *Cerlalc*. https://cerlalc.org/recomendaciones-para-autores/los-derechos-patrimoniales/ (consultado may 06, 2021).

[40]
"Espondiloartritis". https://www.rheumatology.org/I-Am-A/Patient-Caregiver/Enfermedades-y-Condiciones/Espondiloartritis (consultado may 05, 2021).

[41]
"Artritis reumatoide - Síntomas y causas - Mayo Clinic". https://www.mayoclinic.org/es-es/diseases-conditions/rheumatoid-arthritis/symptoms-causes/syc-20353648#:~:text=La%20osteoartritis%2C%20la%20forma%20m%C3%A1s,y%20causa%20dolor%20e%20hinchaz%C3%B3n. (consultado may 05, 2021).

[42]
C. R. Falcon, "Rheumatology", *Magill's Medical Guide (Online Edition)*. Salem Press, 2019, Consultado: may 05, 2021. [En línea]. Disponible en: https://login.ezproxy.javeriana.edu.co/login?url=https://search.ebscohost.com/login.aspx?direct=true&db=ers&AN=89093547&lang=es&site=eds-live.

[43]
D. G. Fernández-Ávila *et al.*, "Design of an algorithm for the diagnostic approach of patients with joint pain", *Clin Rheumatol*, vol. 40, núm. 4, pp. 1581–1591, abr. 2021, doi: 10.1007/s10067-020-05323-w.

[44]
D. G. Fernández-Ávila, M. X. Rojas, C. Ramírez, L. Rodelo, y E. Soriano, "Effectiveness of the use of an algorithm in the diagnostic approach of joint pain patients by primary care physicians", *Rheumatol Int*, vol. 40, núm. 11, pp. 1857–1864, nov. 2020, doi: 10.1007/s00296-020-04552-1.

[45]
"Qué es la Reumatología", *Inforeuma*. https://inforeuma.com/quienes-somos/que-es-la-reumatologia/ (consultado may 03, 2021).

[46]
"Los software para realizar diagnósticos más precisos.", *Blogthinkbig.com*, ago. 24, 2017. https://blogthinkbig.com/software-asistente-medicos-diagnosticos (consultado may 03, 2021).

[47]
"Diagnóstico y tratamiento de las enfermedades Reumáticas | Clínica Reumatológica Dr. Ponce", dic. 21, 2015. https://www.doctorponce.com/diagnostico-y-tratamiento-

de-las-enfermedades-reumaticas/, https://www.doctorponce.com/diagnostico-y-tratamiento-de-las-enfermedades-reumaticas/ (consultado may 03, 2021).

[48] O. Rivero Serrano y L. A. Martínez, "La medicina actual", *Revista de la Facultad de Medicina (México)*, vol. 54, núm. 2, pp. 21–32, 2011, Consultado: may 03, 2021. [En línea]. Disponible en: http://www.scielo.org.mx/scielo.php?script=sci_abstract&pid=S0026-17422011000200004&lng=es&nrm=iso&tlng=es.

Linares-Vásquez, M. and Escobar-Velásquez, C., 2021. E2E Testing. [online] End-2-End testing. Disponible en: <https://miso-4208-labs.gitlab.io/book/chapter2/por-fin-e2e-testing.html> [Consultado: octubre 16, 2021]. [49]

Linares-Vásquez, M. and Escobar-Velásquez, C., 2018. ¿Pruebas de GUI, o pruebas basadas en la GUI? [online] ¿Pruebas de GUI, o pruebas basadas en la GUI? - GitBook. Disponible en: <https://miso-4208-labs.gitlab.io/book/chapter4/pruebas-de-gui-o-pruebas-basadas-en-la-gui.html> [Consultado: octubre 16, 2021]. [50]

X. ANEXOS

Anexo 1: SDD

Anexo 2: Manual de usuario

Anexo 3: Documento de pruebas

Anexo 4: Reporte de pruebas

Anexo 5: Presentación de interfaces

Anexo 6: Prototipos iniciales

Anexo 7: Assets

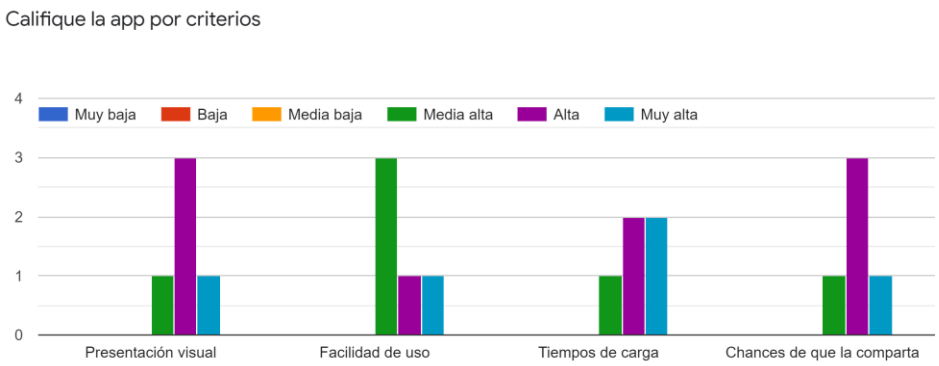
Anexo 8: Diagrama de navegación

Anexo 9: Arbol inicial

Anexo 10: Arbol final

Anexo 11: Encuesta de aceptación

Anexo 12:



Anexo 13: