Pontificia Universidad
JAVERIANA
Cali

Facultad de Ingeniería y Ciencias

Ingeniería Biomédica

Monografía de Trabajo de Grado

Desarrollo de un aplicativo móvil para la identificación de arritmias cardíacas mediante procesamiento digital de señales ECG y aprendizaje automático

Karolina María Otero Argel
Jereminth Muñoz de la Torre

Director
Dr. Hernán Darío Vargas Cardona

Codirector
MSc. Valentina Corchuelo Guzmán

26 de enero de 2025

Santiago de Cali, 26 de enero de 2025

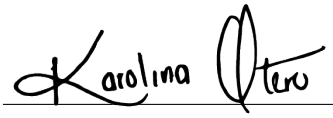
Señores
Pontificia Universidad Javeriana – Cali
Dr. Hernán Camilo Rocha Niño
Decano
Facultad de Ingeniería y Ciencias
Ciudad

Cordial Saludo.

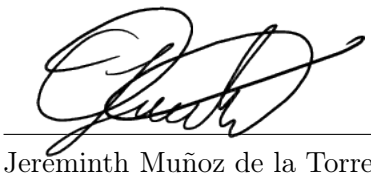
Por medio de la presente nos permitimos presentarle el Trabajo de Grado titulado "Desarrollo de un aplicativo móvil para la identificación de arritmias cardíacas mediante procesamiento digital de señales ECG y aprendizaje automático".

Esperamos que este trabajo reúna todos los requisitos académicos, cumpla el propósito para el cual fue creado y sirva de apoyo para futuros proyectos relacionados con la materia.

Atentamente,



Karolina María Otero Argel



Jeréminth Muñoz de la Torre

Santiago de Cali, 26 de enero de 2025

Señores
Pontificia Universidad Javeriana – Cali
Dr. Hernán Camilo Rocha Niño
Decano
Facultad de Ingeniería y Ciencias
Ciudad

Cordial Saludo.

Certificamos que el presente Trabajo de Grado titulado "Desarrollo de un aplicativo móvil para la identificación de arritmias cardíacas mediante procesamiento digital de señales ECG y aprendizaje automático", realizado por Karolina María Otero Argel y Jereminth Muñoz de la Torre, estudiantes de Ingeniería Biomédica, se encuentra terminado y puede ser presentado para su sustentación.

Atentamente,

Dr. Hernán Darío Vargas Cardona
Director Trabajo de Grado

MSc. Valentina Corchuelo Guzmán
Co-Director Trabajo de Grado

Agradecimientos

En primer lugar, queremos expresar nuestra más profunda gratitud a nuestras familias, en especial a Rosmira Argel, Juan Otero, Lourdes de la Torre y Freddy Muñoz, nuestros padres, por acompañarnos en este camino y brindarnos la oportunidad de realizar nuestros estudios de educación superior en una universidad que nos ofreció tantas posibilidades de crecimiento. Su apoyo incondicional y confianza nos permitieron no solo hacernos profesionales, sino también crecer como seres humanos íntegros y llenos de metas.

Agradecemos especialmente a nuestros profesores, quienes nos guiaron y alentaron en este proceso. En especial, a Hernán Vargas, nuestro director de proyecto, cuyo conocimiento, compromiso y confianza fueron fundamentales en cada etapa del desarrollo de este proyecto. También a Valentina Corchuelo, nuestra profesora y referente en la ingeniería biomédica, por respaldar cada una de las acciones realizadas en este proyecto, motivándonos siempre a dar lo mejor de nosotros.

Queremos también agradecer a nuestro amigo Juan Monroy, por estar presente en cada decisión importante, apoyarnos y llenarnos de energía y confianza bajo todas circunstancias.

Extendemos nuestra gratitud a cada uno de los profesores que compartieron sus conocimientos y herramientas en ingeniería. Gracias a ellos, comprendimos que el verdadero propósito de la ingeniería biomédica es servir y brindar soluciones que mejoren la calidad de vida de las personas, poniendo siempre por encima de cualquier otro interés individual.

No podemos olvidar a nuestros amigos y a todas las personas que, al escuchar nuestra ideas sobre el proyecto, creyeron en nosotros, y todos aquellos que le apuestan a soluciones que tengan gran impacto en la innovación en salud. Su apoyo fue un pilar fundamental para llevar a cabo este proyecto y todo lo que de aquí en adelante evolucioné de esta propuesta.

Finalmente, queremos reconocer nuestro propio esfuerzo y todos esos sacrificios que enfrentamos a lo largo de este camino. Agradecemos la compañía mutua que nos ha fortalecido desde que nos conocimos en el primer semestre de nuestra carrera, cuando nuestros objetivos se alinearon profesionalmente. Esto nos permitió formar un equipo sólido que, a pesar de las dificultades, ha sabido confiar en las capacidades de cada uno y en el poder de nuestro trabajo conjunto. Juntos, hemos demostrado que somos mejores y más fuertes.

Glosario

Símbolos

δ_i Se emplea para denotar el tiempo límite de ejecución para la i-ésima Tarea [ms]
τ_i Se emplea para denotar la i-ésima Tarea

Acrónimos y Abreviaturas

ECV	Enfermedades cardiovasculares.
OMS	Organización Mundial de la Salud.
MSC	Muerte súbita cardíaca.
ECG	Electrocardiograma/ Relacionada con la actividad eléctrica del corazón.
CIE-X	Clasificación Internacional de Enfermedades de la OMS.
CNN	Redes Neuronales Convolucionales.
RNN	Redes Neuronales Recurrentes.
TL	Transferencia de Aprendizaje.
SA	Nodo sinoauricular.
PVC	Contracción Ventricular Prematura.
ML	Machine Learning.
IA	Inteligencia Artificial.
ANN	Redes Neuronales Artificiales.
MLP	Perceptrón multicapa.
DMLP	Redes neuronales multicapa.
LSTM	Red de memoria a corto-largo plazo.
VSCode	Visual Studio Code.
API	Interfaz de programación de aplicaciones.
KNN	K vecinos más próximos.
Acc	Accuracy (métrica de evaluación de modelos).
CWT	Transformada Wavelet Continua.
AWS	Amazon Web Service
VPS	Servidor Privado Virtual (Virtual Private Server).
APK	Paquete de aplicación de Android.
BT	Bluetooth, protocolo inalámbrico para transferencia de datos.
JSON	Formato de intercambio de datos (JavaScript Object Notation).
SSH	Protocolo seguro para acceso remoto.
SSL	Certificado de seguridad que cifra la comunicación.
UI	Interfaz de Usuario (User Interface).
UX	Experiencia del Usuario (User Experience).

Resumen

Las enfermedades cardiovasculares (ECV) representan una preocupación global, siendo una de las principales causas de mortalidad según la Organización Mundial de la Salud (OMS). En Colombia, estas enfermedades ocupan un lugar importante en las estadísticas de mortalidad, destacando las arritmias cardíacas como un factor crítico debido a su potencial para desencadenar muerte súbita cardíaca (MSC). El objetivo de esta propuesta es desarrollar un aplicativo móvil basado en procesamiento digital de señales ECG y aprendizaje automático para identificar arritmias cardíacas y alertar sobre la posibilidad de MSC en pacientes con antecedentes cardiovasculares o enfermedades crónicas. La metodología del estudio se estructura en varias etapas, incluyendo la selección de la base de datos ECG adecuada, el procesamiento de señales, la extracción de características, el entrenamiento de modelos de aprendizaje automático, la validación y evaluación de los modelos, la implementación del aplicativo móvil y la validación del mismo mediante simuladores de ECG. Los resultados obtenidos incluyen un modelo de aprendizaje automático con una precisión del 96 %, lo que demuestra su eficacia en la clasificación de arritmias cardíacas. Además, se logró integrar exitosamente el modelo en el aplicativo móvil denominado KIBO, el cual ofrece tres salidas principales: el ritmo cardíaco, la identificación de arritmias potencialmente peligrosas capaces de desencadenar muerte súbita cardíaca (MSC), y el vector de ECG asociado. Esto permite lograr el fin último de activar de manera oportuna una ruta de atención en emergencias, mejorando la respuesta ante posibles complicaciones relacionadas con MSC y contribuyendo a la prevención de eventos fatales.

Palabras Clave: Arritmias cardíacas, Muerte Súbita Cardíaca, procesamiento de señales, aprendizaje automático, aplicativo móvil.

Abstract

Cardiovascular diseases (CVD) represent a global concern, being one of the main causes of mortality according to the World Health Organization (WHO). In Colombia, these diseases occupy an important place in mortality statistics, highlighting cardiac arrhythmias as a critical factor due to their potential to trigger sudden cardiac death (SCD). The objective of this proposal is to develop a mobile application based on digital processing of ECG signals and machine learning to identify cardiac arrhythmias and alert about the possibility of SCD in patients with cardiovascular history or chronic diseases. The methodology of the study is structured in several stages, including selection of the appropriate ECG database, signal processing, feature extraction, training of machine learning models, validation and evaluation of the models, implementation of the mobile application, and validation of the mobile application using ECG simulators. The results obtained include a machine learning model with an accuracy of 96 %, which demonstrates its effectiveness in the classification of cardiac arrhythmias. In addition, the model was successfully integrated into the mobile application called KIBO, which offers three main outputs: the heart rhythm, the identification of potentially dangerous arrhythmias capable of triggering sudden cardiac death (SCD), and the associated ECG vector. This allows the ultimate goal of timely activation of an emergency care pathway, improving the response to potential SCD-related complications and contributing to the prevention of fatal events.

Keywords: Cardiac arrhythmias, Sudden Cardiac Death, signal processing, machine learning, mobile application.

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Introducción

Las enfermedades cardiovasculares (ECV) representan un desafío creciente en la salud pública global, siendo una de las principales causas de morbilidad y mortalidad en todo el mundo [3]. En particular, las arritmias cardíacas, constituyen una preocupación significativa debido a su potencial para desencadenar complicaciones graves, incluida la muerte súbita cardiaca (MSC). En Colombia, las enfermedades cardiovasculares ocupan una posición destacada entre las causas de mortalidad, con las enfermedades isquémicas del corazón y las cerebrovasculares entre las principales causas de muerte [4].

La muerte súbita cardíaca (MSC) por su parte, conserva una prevalencia considerable, pues se estima que millones de personas en todo el mundo son víctimas de esta condición anualmente, y las arritmias cardíacas son una de las causas subyacentes [5]. Ahora bien, las arritmias cardíacas afectan a miles de personas, lo que representa una amenaza para la salud pública. Además, la falta de una identificación oportuna puede conducir a complicaciones graves y costosas, tanto para el sistema de salud como para los pacientes [6]. La identificación temprana de las arritmias, especialmente aquellas que aumentan el riesgo de MSC, es clave para implementar intervenciones preventivas que reduzcan el impacto negativo para la vida de los pacientes y la carga económica asociada.

Desde la ingeniería biomédica, se busca desarrollar soluciones que aborden necesidades en el campo de la salud cardiovascular, como la detección temprana de arritmias cardíacas y la prevención de la muerte súbita cardíaca. Es así como, la aplicación de modelos de aprendizaje automático emerge como una herramienta prometedora para resolver este problema, ya que, al combinar conocimientos en ingeniería, informática y medicina, se pueden diseñar sistemas inteligentes capaces de analizar grandes volúmenes de datos de señales cardíacas e identificar patrones que podrían indicar la presencia de arritmias.

De acuerdo con lo anterior, este proyecto propone el desarrollo de un aplicativo móvil basado en procesamiento digital de señales ECG y aprendizaje automático como una herramienta para identificar arritmias cardíacas y alertar sobre la posibilidad de muerte súbita cardíaca en pacientes con antecedentes cardiovasculares o enfermedades crónicas. En este trabajo, se presentan los métodos y estrategias empleadas para el desarrollo de un aplicativo móvil con el fin de identificar arritmias cardíacas, mediante el procesamiento de señales ECG y el uso de aprendizaje automático. A lo largo de las fases del proyecto, se han implementado técnicas de procesamiento digital de señales con el objetivo de mejorar la calidad y la precisión de los datos utilizados. Además, se describe el entrenamiento de varios modelos de machine learning, como redes neuronales convolucionales (CNN),

combinadas con redes neuronales recurrentes (RNN), y métodos de Transfer Learning (TL). Estas metodologías permitieron la creación de un sistema robusto capaz de identificar patrones de arritmias cardíacas con alta precisión.

Los resultados obtenidos son presentados en términos de la evaluación de los modelos entrenados, destacando las métricas de rendimiento y la comparación entre los diferentes enfoques aplicados. En la sección de resultados y discusión se exploran los hallazgos más relevantes, destacando el desempeño de los modelos en la identificación de arritmias y la relevancia de estos resultados para la prevención de la muerte súbita cardíaca. Además, se registran las evidencias de las pruebas realizadas con conjuntos de datos no conocidos, así como simulaciones de conexión con simuladores de ECG (ProSim4) que emulan un paciente monitoreado y que hace uso de la aplicación móvil, la cual se nombró KIBO. Esta aplicación integra todo el trabajo realizado en este proyecto, desde la experiencia de usuario hasta la programación de los algoritmos desarrollados y ejecutados en este proyecto. Finalmente, el informe ofrece una visión hacia los trabajos futuros, proponiendo mejoras en el sistema, como la integración de nuevos algoritmos, la ampliación de bases de datos y la optimización del aplicativo móvil para un uso más generalizado. Estas iniciativas buscan contribuir al avance en el campo de la salud cardiovascular mediante el desarrollo e implementación de tecnologías que mejoren la identificación y gestión de las arritmias cardíacas, con el fin último de prevenir complicaciones graves y salvar vidas.

Capítulo 2

Planteamiento del Problema

Según la Organización Mundial de la Salud (OMS), las enfermedades cardiovasculares (ECV) son la principal causa de mortalidad en el mundo. Se señala que, en los últimos 20 años, el número de muertes ha aumentado considerablemente. En el año 2000 se reportaron 2 millones de muertes, cifra que se ha cuadruplicado llegando a cerca de 9 millones de personas en 2019. Para el año 2020, se encontró que las muertes por eventos relacionados con el corazón y los vasos sanguíneos representan el 16 % de todas las ocasionadas en el mundo en este período [3]. Las ECV representan un amplio espectro de condiciones que afectan directamente el sistema cardiovascular, incluyendo enfermedades reumáticas del corazón, enfermedades hipertensivas, cardiomiopatías y otras afecciones del sistema circulatorio. Estas enfermedades, junto con la cardiopatía isquémica y el accidente cerebrovascular, son causas principales de morbilidad y mortalidad en la Región de las Américas [4].

Las enfermedades vinculadas al corazón y al sistema circulatorio revelan una prevalencia, variedad y gravedad a nivel global. De acuerdo con el Ministerio de Salud y Protección Social, en Colombia para los años 2020-2021, las enfermedades isquémicas del corazón, es decir las condiciones producidas por estrechamiento o estenosis de las arterias, se ubicó en el segundo lugar, dentro del listado de las primeras 20 causas de mortalidad, lo que representa el 14 % del total de muertes. Sin embargo, es solo una de las categorías de enfermedades cardiovasculares que se registra como una causa de mortalidad potencial [7]. Las enfermedades cerebrovasculares, hipertensivas, cardiopulmonares y demás eventos cardiovasculares, como las arritmias ocupan porcentajes significativos dentro de las causas de muertes en el país.

Del mismo modo, la muerte súbita cardíaca (MSC) constituye un desafío crítico en la salud pública regional y global, dado su impacto devastador y su continua prevalencia a pesar de los avances médicos. Se estima que anualmente entre 4 y 5 millones de personas en todo el mundo son víctimas de esta condición, siendo las enfermedades cardiovasculares, especialmente las arritmias cardíacas, una de las principales causas subyacentes [8]. Las arritmias se clasifican como trastornos del ritmo cardíaco, es decir, es una condición en la que el corazón experimenta latidos irregulares debido a la disfunción en los impulsos eléctricos que regulan su actividad. Estos latidos irregulares pueden manifestarse como una frecuencia cardíaca rápida (taquicardia) o lenta (bradicardia), así como un patrón de ritmo cardíaco irregular. Si bien algunas arritmias pueden ser inofensivas, otras pueden ser potencialmente mortales y aumentar significativamente el riesgo de eventos cardíacos graves como la muerte súbita cardíaca [9].

Las arritmias cardíacas, ya sean ventriculares o auriculares, representan un espectro diverso

de trastornos que pueden surgir de condiciones preexistentes, como la enfermedad coronaria, las miocardiopatías y los trastornos genéticos. Por ejemplo, el síndrome de QT largo es un trastorno hereditario del ritmo cardíaco que predispone a los individuos a arritmias potencialmente mortales [8]. Además, la isquemia cardíaca aguda y la fibrosis miocárdica pueden crear sustratos favorables para la generación de arritmias malignas. Estos trastornos del ritmo cardíaco pueden interrumpir el ritmo cardíaco normal y conducir rápidamente a la MSC si no se tratan de manera efectiva.

Ahora bien, existe alta prevalencia de arritmias cardíacas en Colombia, con más de 600 mil personas afectadas según datos de la Asociación Colombiana de Cardiología. Esta condición representa una amenaza para la salud pública debido a su potencial mortalidad si no se aborda de manera oportuna. Como ya se indicó, en muchos casos, estas arritmias pueden desencadenar la denominada muerte súbita cardíaca o eventos cardiovasculares graves, incluso en individuos aparentemente sanos.

La gestión de las arritmias cardíacas se enfrenta a un desafío considerable debido a la diversidad de condiciones que pueden desencadenar eventos adversos graves. La fibrilación auricular, una de las arritmias más letales, resalta la complejidad de esta problemática. En Colombia, el tratamiento de esta afección representa una carga económica considerable para el sistema de salud, como se destaca en un estudio reciente de la Revista Colombiana de Cardiología [6]. El principal obstáculo radica en la identificación precisa de las arritmias que conllevan un mayor riesgo de eventos agudos, como los tromboembólicos o los infartos, lo que implica costos asistenciales más elevados. La dificultad diagnóstica adicional se encuentra en la identificación temprana del tipo específico de arritmia que afecta al paciente, lo que permitiría implementar intervenciones preventivas oportunas y, por ende, reducir las complicaciones fatales [10]. La falta de un diagnóstico precoz puede resultar en terapias más intensivas y costosas, además de aumentar el riesgo de complicaciones graves, generando así una carga financiera adicional tanto para el sistema de salud como para los pacientes.

Finalmente, la falta de información y conocimiento detallado sobre las distintas arritmias cardíacas representa una necesidad en la comunidad académica y médica. Aunque se reconoce la existencia de una amplia variedad de arritmias y se comprende su potencial letalidad, la investigación específica sobre sus tipos y consecuencias letales es limitada. Esta brecha de conocimiento dificulta la identificación temprana y precisa de las arritmias que pueden desencadenar eventos cardíacos graves, lo que a su vez afecta la capacidad de intervenir oportunamente y prevenir complicaciones fatales. Dada la notable prevalencia de estas condiciones y su influencia en la mortalidad, se vuelve esencial desarrollar herramientas y estrategias que permitan prevenir y gestionar eficazmente las arritmias cardíacas que comprometen seriamente la vida de las personas afectadas.

Capítulo 3

Justificación

En el panorama actual de la salud cardiovascular, las arritmias cardíacas se convierten en una preocupación crucial. Estas anomalías del ritmo cardíaco, especialmente cuando están asociadas con cardiopatías estructurales, representan una amenaza seria para la vida de los individuos, aumentando el riesgo de muerte súbita cardíaca [9]. Ante esta realidad, la necesidad urgente de intervenir en este problema se hace evidente, ya que la identificación y el manejo de estas condiciones son esenciales para prevenir situaciones evitables y salvar vidas. La muerte súbita cardíaca es un fenómeno caracterizado por un colapso repentino del corazón debido a arritmias, que puede ocurrir tanto en personas con o sin enfermedades cardíacas previas [11]. Cuando esto sucede, la circulación de sangre al cerebro y al resto del cuerpo se interrumpe, lo que puede llevar a la muerte si no se actúa en cuestión de minutos. A nivel mundial, este tipo de muerte representa al menos el 20-30 % de todas las muertes de origen cardíaco. Aunque afecta principalmente a personas de entre 45 y 75 años, también puede presentarse en grupos más jóvenes, especialmente aquellos con condiciones cardíacas preexistentes [5].

En Colombia, la incidencia de arritmias cardíacas es preocupante. Aunque las estadísticas exactas no están disponibles, se estima que aproximadamente 600,000 individuos padecen esta condición, según datos de la Asociación Colombiana de Cardiología, y una parte de estos casos no recibió diagnóstico ni tratamiento adecuado en el momento oportuno, lo que resalta la importancia de abordar este problema de manera integral y efectiva [12]. Por esta razón, la prevención adquiere una gran importancia en este desafío de salud pública, haciendo que los chequeos regulares, el seguimiento médico continuo y la adopción de hábitos de vida saludables aparezcan como estrategias óptimas para afrontarlo. Además, es esencial reforzar los sistemas de atención médica primaria para garantizar el acceso oportuno a servicios de diagnóstico y tratamiento especializado. En este contexto, la implementación de tecnologías como aplicaciones móviles y dispositivos de monitoreo remoto puede desempeñar un papel fundamental en la detección y gestión de las arritmias, facilitando la comunicación entre pacientes, organismos de atención a emergencias e instituciones de salud.

El desarrollo de sistemas de monitorización cardíaca, como el electrocardiograma (ECG), ha sido clave para facilitar la detección y diagnóstico de arritmias. Sin embargo, el análisis manual de los extensos datos generados por estos sistemas representa un desafío para los cardiólogos [13]. Ante esta dificultad, el uso de sistemas automáticos de análisis se ha vuelto una propuesta viable que puede reconocer y clasificar varios tipos de arritmias de manera eficiente. Algunos estudios previos han explorado el potencial de técnicas de aprendizaje automático, como redes neuronales y convolucionales, para automatizar este proceso, mejorando así la atención médica y el tratamiento de

pacientes con problemas cardíacos [13] [14]. Además, se han realizado trabajos que han implementado algoritmos especializados que integran técnicas de análisis de datos de ECG [15] [16]. Estas herramientas ofrecen una solución integral que puede mejorar la precisión y eficiencia en el diagnóstico automatizado de arritmias cardíacas, contribuyendo directamente a la aplicación de sistemas automáticos de análisis en la monitorización cardíaca. Este enfoque permite una identificación más temprana de las arritmias, lo que facilita intervenciones médicas oportunas para los pacientes afectados.

Dentro de la literatura científica, se ha establecido una relación entre ciertos tipos de arritmias y la muerte súbita cardiaca [17]. Por lo tanto, identificar arritmias para alertar sobre la posibilidad de muerte súbita cardíaca se convierte en un objetivo relevante y prometedor en la mejora de la atención médica y la prevención de eventos letales en pacientes con problemas cardíacos. Además, en la actualidad no existen sistemas portátiles de detección de arritmias cardíacas que proporcionen una alerta temprana sobre la posibilidad de muerte súbita cardíaca. Por lo tanto, el desarrollo de una aplicación móvil dedicada a esta tarea se presenta como una solución atractiva en el campo de la salud cardiovascular. Esta aplicación podría llenar el vacío existente en la atención médica al proporcionar a los usuarios una herramienta fácilmente accesible para monitorear su salud cardíaca y recibir alertas sobre posibles riesgos de arritmias y eventos cardíacos adversos, como la muerte súbita cardíaca.

Por otra parte, la viabilidad de este proyecto también se ve respaldada por la disponibilidad de repositorios gratuitos y bases de datos de libre acceso que contienen registros de ECG y datos relacionados con arritmias cardíacas. Esto permite el acceso a una amplia variedad de información que puede ser utilizada para el desarrollo y validación de algoritmos de identificación de arritmias, sin incurrir en costos significativos de adquisición de datos.

Ahora bien, nuestra formación en procesamiento digital de señales, programación y comprensión de parámetros clínicos proporcionada por nuestro programa académico de ingeniería biomédica, nos posiciona de manera favorable para llevar a cabo este proyecto de manera exitosa. Se espera que, el desarrollo de este proyecto aporte materiales y herramientas de identificación de arritmias cardíacas, lo que permitirá a los usuarios tomar medidas preventivas, y tener la facilidad de ser atendidos y buscar atendidos de manera oportuna, evitando complicaciones fatales, como la muerte súbita cardíaca, sentando la base de futuras intervenciones en materia de algoritmos más robustos que pueden identificar patrones para predecir la ocurrencia de estas condiciones antes de que aparezcan. Esto no solo mejorará la calidad de vida de las personas con riesgo de arritmias, sino que también contribuirá a reducir los costos asociados al tratamiento de complicaciones cardíacas graves. En última instancia, este proyecto busca establecer protocolos o rutas más eficientes para responder ante arritmias potencialmente mortales, alertando tanto a los pacientes, como a las líneas de emergencia para garantizar una atención rápida en situaciones críticas.

Finalmente, se espera que la incorporación de la inteligencia artificial como base tecnológica

tenga trascendencia en el ámbito médico, ofreciendo soluciones costo-efectivas en la prevención de las arritmias cardíacas. Desde la ingeniería biomédica, se pueden desarrollar soluciones tecnológicas que impacten positivamente en el día a día de las personas con riesgo de arritmias y posibilidad de muerte súbita cardíaca. Estas soluciones pueden facilitar el monitoreo continuo, la identificación de anomalías y generación de alertas para intervención rápida, contribuyendo así a reducir los riesgos asociados y mejorar el pronóstico de los pacientes afectados.

Capítulo 4

Objetivos

4.1. Objetivo General

Desarrollar un aplicativo móvil basado en procesamiento digital de señales ECG y aprendizaje automatizado que permita identificar arritmias cardíacas y alertar la posibilidad de muerte súbita cardíaca en pacientes con antecedentes cardiovasculares o enfermedades crónicas.

4.2. Objetivos Específicos

- Gestionar una base de datos de señales ECG depurada y estructurada para facilitar el análisis y procesamiento efectivo en la identificación de arritmias cardíacas.
- Extraer características de las señales ECG mediante técnicas de procesamiento digital de señales para el entrenamiento de algoritmos de aprendizaje automático.
- Entrenar algoritmos de aprendizaje automático que permitan la identificación de arritmias cardíacas.
- Evaluar los modelos entrenados usando métricas de clasificación para la selección del mejor método en la identificación de arritmias cardíacas.
- Implementar un aplicativo móvil que recepcione los datos ECG provenientes de un dispositivo Holter y genere alertas de arritmias cardíacas.

Capítulo 5

Marco de Referencia

5.1. Áreas Temáticas

- El sistema cardiovascular y la mecánica de corazón.
- Enfermedades cardiovasculares.
- Arritmias cardíacas y muerte súbita cardiaca.
- Técnicas médicas utilizadas en la detección de arritmias cardíacas.
- Procesamiento digital de señales.
- Machine learning.
- Desarrollo móvil.

5.2. Marco Teórico

5.2.1. El sistema cardiovascular y el funcionamiento del corazón.

Uno de los sistemas vitales del cuerpo humano es el Sistema Cardiovascular. Este sistema comprende una compleja red de componentes que incluyen el corazón, las arterias, las venas y los capilares. Su función principal radica en el transporte de la sangre, que contiene oxígeno y nutrientes esenciales, a todas las células del organismo y, al mismo tiempo, elimina los productos de desecho [18].

- **El corazón:** El corazón es un órgano muscular que desempeña un papel central en este sistema y se considera su núcleo. Actúa como una bomba que impulsa la sangre a través de las arterias, llevando consigo oxígeno y nutrientes desde el corazón hasta los tejidos del cuerpo [18].
- **Funcionamiento del corazón:** El correcto funcionamiento del corazón se basa en la adecuada actividad eléctrica, que se manifiesta a través de la activación secuencial de células especializadas llamadas "marcapasos", en el corazón, así como en la propagación de esta actividad a través de los ventrículos[19]. El impulso cardíaco surge en el nodo sinoauricular (SA), que se encuentra en la pared posterior de la aurícula derecha, este es el principal marcapasos cardíaco, porque tiene una tasa de generación de impulsos muy rápida (60 a 100 lpm, latidos por minuto), con este impulso eléctrico empieza lo que se conoce como ciclo cardíaco, que se define como el período desde que comienza un latido hasta el comienzo del siguiente latido.

- **Señal ECG:** La actividad eléctrica del corazón se puede registrar e interpretar por medio de ondas. Las ondas que registra el ECG son P, Q, R, S, y T, estas ondas representan cada uno de los momentos del latido cardíaco y existen segmentos formados por las ondas que refieren un comportamiento particular [20].

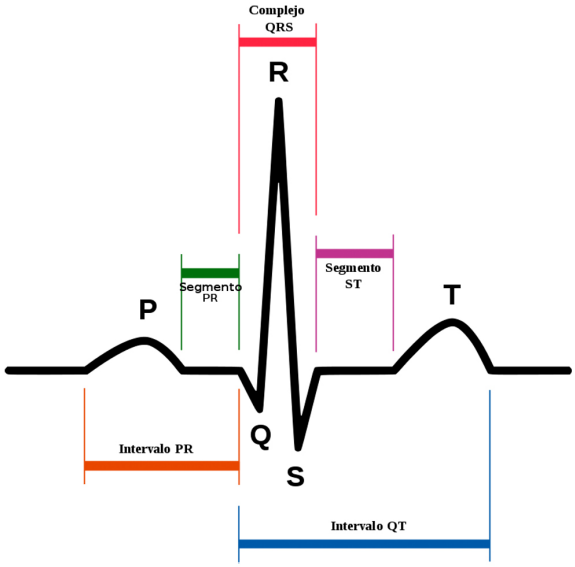


Figura 5.1: Ondas señal ECG. (Fuente: [1])

5.2.2. Enfermedades cardiovasculares

Las enfermedades cardiovasculares (ECV) se refieren a un conjunto muy diverso de trastornos que afectan al corazón y a los vasos sanguíneos. Estas enfermedades pueden variar desde condiciones leves hasta graves, y a menudo implican problemas relacionados con el flujo sanguíneo y el funcionamiento del corazón. Según la décima versión de la Clasificación Internacional de Enfermedades de la OMS (CIE-X) algunos de los grupos de las enfermedades del aparato circulatorio son [21]:

- Cardiopatías reumáticas crónicas - Enfermedades hipertensivas incluyendo la eclampsia (hipertensión durante el embarazo) - Cardiopatía isquémica (infarto de miocardio, angina de pecho) - Enfermedad cardiopulmonar - Muerte súbita cardíaca. - Otras enfermedades del corazón (p.e. arritmias e insuficiencia cardíaca entre otras)

- **Trastornos de ritmo cardíaco:** Los trastornos del ritmo cardíaco, también conocidos como arritmias, son alteraciones en la frecuencia, regularidad o secuencia de los latidos del corazón.

Pueden manifestarse como latidos demasiado rápidos (taquicardia), demasiado lentos (bradicardia), o ritmos irregulares. Estas anomalías pueden ser benignas o potencialmente mortales, y son causadas por diversas condiciones, como enfermedades cardíacas, problemas estructurales, desequilibrios electrolíticos o factores externos como el estrés. A continuación se explorarán los tipos de arritmias más comunes, sus causas y consecuencias [22].

1. **Taquicardia:** La taquicardia se caracteriza por un ritmo cardíaco rápido, con más de 100 latidos por minuto. Puede surgir debido a varios factores, incluyendo la excitación emocional, el estrés, el ejercicio intenso, el consumo de cafeína o la presencia de enfermedades cardíacas subyacentes.
2. **Bradicardia:** La bradicardia implica un ritmo cardíaco lento, con menos de 60 latidos por minuto. Puede ser causada por una disfunción del nodo sinusal, el marcapasos natural del corazón, o por enfermedades como el hipotiroidismo, la enfermedad del nodo sinusal o el uso de ciertos medicamentos.
3. **Arritmias Supraventriculares:** Estas arritmias se originan en las cámaras superiores del corazón, específicamente en las aurículas. Pueden incluir condiciones como la fibrilación auricular, el flutter auricular y el síndrome de Wolff-Parkinson-White. Las causas pueden variar desde factores genéticos hasta enfermedades cardíacas estructurales.
4. **Arritmias Ventriculares:** Las arritmias ventriculares tienen su origen en las cámaras inferiores del corazón, los ventrículos. Ejemplos incluyen la taquicardia ventricular y la fibrilación ventricular. Estas arritmias pueden ser graves y potencialmente mortales, y a menudo están asociadas con enfermedades cardíacas avanzadas, como la cardiopatía isquémica.
5. **Bradiarritmias:** Las bradiarritmias son ritmos cardíacos lentos que generalmente están causados por enfermedades en el sistema de conducción del corazón, como el bloqueo atrioventricular completo o la enfermedad del nodo sinusal. Estas pueden provocar síntomas como fatiga, mareos, desmayos o incluso insuficiencia cardíaca.

Las arritmias cardíacas pueden tener serias consecuencias. Entre ellas se incluyen la reducción de la capacidad del corazón para bombear sangre de manera eficaz, lo que puede llevar a insuficiencia cardíaca. Éstas irregularidades pueden propiciar la formación de coágulos sanguíneos, aumentando así el riesgo de eventos graves como accidentes cerebrovasculares o embolias pulmonares. Los síntomas de las arritmias pueden ser alarmantes, incluyendo palpitaciones, mareos, desmayos, fatiga o dificultad para respirar. En casos extremos, las arritmias pueden desencadenar complicaciones potencialmente mortales, como la fibrilación ventricular, que puede causar muerte súbita cardíaca si no se trata adecuadamente.

- **Muerte súbita cardíaca (MSC):** Se refiere a una muerte inesperada que ocurre generalmente dentro de una hora desde el inicio de los síntomas en una persona aparentemente sana o que no tiene antecedentes conocidos de enfermedad cardíaca. Puede ser causada por una

serie de condiciones subyacentes, como arritmias cardíacas graves, anomalías estructurales del corazón o enfermedades coronarias avanzadas [11].

De acuerdo con el estudio de Rodríguez-Reyes et al., las arritmias cardíacas desempeñan un papel crucial como potencial causa de muerte súbita cardíaca (MSC), una condición que representa un desafío significativo para la salud pública. Según los datos, el 60-70 % de las MSC están asociadas a la cardiopatía isquémica, mientras que las miocardiopatías contribuyen en un 20-30 %. Sin embargo, un 5-10 % de estas muertes se deben a enfermedades arrítmicas primarias, subrayando el impacto directo de las alteraciones eléctricas del corazón. Entre las arritmias más frecuentemente detectadas en estos casos, destacan la fibrilación ventricular y la taquicardia ventricular, responsables del 60-80 % de los eventos [23]. Otras arrtimias que pueden ser desencadenantes de la MSC son:

- Contracción Ventricular Prematura: Las contracciones ventriculares prematuras (PVC) son latidos adicionales que comienzan en los ventrículos del corazón. Si bien las PVCs aisladas no suelen ser peligrosas, en personas con cardiopatía estructural, la presencia de múltiples PVCs puede ser un precursor de arritmias más graves como la taquicardia ventricular o la fibrilación ventricular, que pueden llevar a la muerte súbita.
- Onda de Aleteo Ventricular: El aleteo ventricular es una arritmia grave que se caracteriza por una rápida secuencia de contracciones ventriculares que pueden ser ineficaces para mantener un flujo sanguíneo adecuado. Esto puede progresar a fibrilación ventricular, que es una de las causas más comunes de muerte súbita cardíaca.
- Fusión de Ventricular y Normal: La presencia de latidos de fusión, que son una combinación de una contracción ventricular prematura y un latido normal, puede ser indicativa de una actividad eléctrica ventricular anormal. Aunque no son directamente causantes de muerte súbita, su presencia en un contexto de otras arritmias ventriculares puede aumentar el riesgo.
- Fusión de Rítmica y Normal: Similar a la fusión ventricular, los latidos de fusión rítmica y normal también pueden ser un indicador de una actividad eléctrica anómala que podría predisponer a arritmias más graves si están presentes en un entorno de cardiopatía estructural.

Estas arritmias suelen presentarse de manera súbita y, a menudo, extrahospitalaria (70-89 % de los casos), lo que dificulta la intervención médica oportuna. Con una supervivencia promedio estimada menor al 5 %, enfatiza la necesidad de estrategias preventivas efectivas, como la identificación temprana de factores de riesgo arrítmicos y el uso de tecnologías avanzadas para su monitoreo.

5.2.3. Técnicas médicas utilizadas en la detección de arritmias cardíacas.

El diagnóstico preciso de enfermedades cardiovasculares es esencial para proporcionar el tratamiento adecuado y prevenir complicaciones. Entre las pruebas y procedimientos utilizados

para el diagnóstico de trastornos del ritmo cardíaco se encuentran [24]:

- **Electrocardiograma (ECG):** Esta prueba rápida e indolora registra las señales eléctricas del corazón y es fundamental en el diagnóstico de enfermedades cardiovasculares. El ECG puede revelar si el corazón late demasiado rápido, demasiado lento o presenta ritmos cardíacos irregulares.
- **Monitoreo con Holter:** Para detectar latidos cardíacos irregulares que pueden no ser evidentes en un ECG convencional, se utiliza un monitor Holter, un dispositivo portátil para electrocardiogramas que se lleva durante un día o más mientras se realizan actividades cotidianas

El ECG es la primera línea de evaluación en muchos casos y proporciona información crucial para el diagnóstico temprano y preciso de afecciones cardíacas, lo que a su vez permite iniciar un tratamiento oportuno y mejorar las perspectivas de salud cardiovascular de los pacientes. Por tanto, su uso en la detección y diagnóstico de condiciones cardíacas como las arritmias es indispensable.

5.2.4. Procesamiento digital de señales.

El procesamiento de señales es el campo de la ciencia que implica la manipulación de señales para obtener la forma deseada, transformando una señal del dominio del tiempo al dominio de la frecuencia y viceversa, suavizando la señal, separando el ruido de la señal, es decir, filtrando y extrayendo información de la señal [2].

Conceptos básicos del sistema de procesamiento de señales.

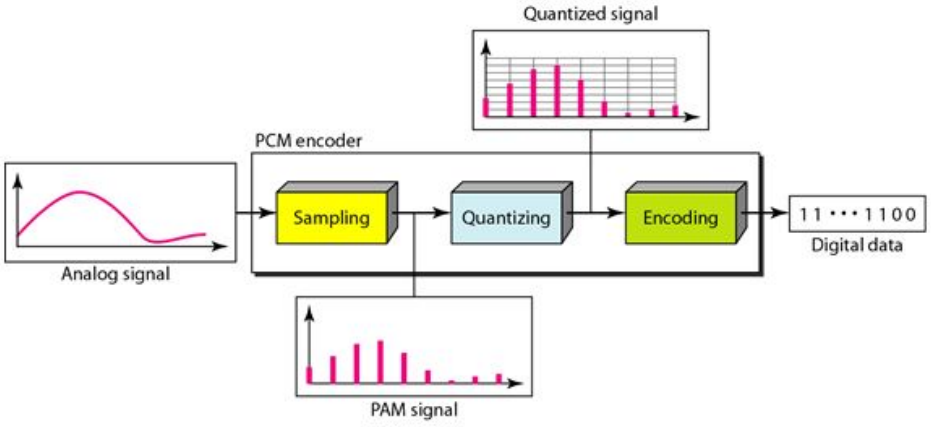


Figura 5.2: Proceso de conversión de la señal. (Fuente: [2])

- **Conversión analógica a digital:** El proceso de conversión analógica a digital consta de varias etapas. Comienza con la señal analógica, que se somete a un muestreo para

obtener una señal en tiempo discreto. Posteriormente, esta señal se cuantifica para obtener una señal cuantizada, que luego se pasa a un codificador para generar la señal digital, representada típicamente por secuencias de números binarios, como por ejemplo, "0101"[2].

- **Convolución:** Es una operación matemática que combina dos funciones en una tercera función, representando cómo una función se modifica al pasar a través de otra función. En el contexto del procesamiento de señales, la convolución se utiliza para caracterizar cómo un sistema transforma una señal de entrada en una señal de salida, mediante la operación matemática de integración del producto de dos funciones a lo largo de un intervalo de tiempo o espacio [2]. Es una herramienta fundamental en el análisis de sistemas lineales y se expresa matemáticamente como:

$$y(t) = \int_{-\infty}^{\infty} x(\tau)h(t-\tau)d\tau$$

donde:

1. $y(t)$ es la señal de salida.
2. $x(t)$ es la señal de entrada.
3. $h(t)$ es la respuesta al impulso del sistema.
4. τ es una variable de integración.

- **La Transformada de Fourier:** Es una herramienta matemática que permite descomponer una señal en una suma de señales sinusoidales de diferentes frecuencias. Esta descomposición se realiza mediante la aplicación de la Transformada de Fourier, que convierte una señal del dominio del tiempo a su representación en el dominio de la frecuencia. Matemáticamente, la Transformada de Fourier se define como:

$$X(f) = \int_{-\infty}^{\infty} x(t)e^{-j2\pi ft}dt$$

donde:

1. $X(f)$ es la representación en el dominio de la frecuencia de la señal $x(t)$.
2. $x(t)$ es la señal en el dominio del tiempo.
3. f es la frecuencia.
4. j es la unidad imaginaria.

- **La Transformada de Gabor:** Es una herramienta en el análisis de señales que permite representar una señal en un dominio tiempo-frecuencia, lo que significa que puede analizar la señal seleccionando intervalos temporales y mostrando las componentes frecuenciales de la señal en cada instante de tiempo [25]. Matemáticamente, la Transformada de Gabor se expresa como:

$$G(t,f) = \int_{-\infty}^{\infty} x(\tau)g^*(\tau-t)e^{-j2\pi f\tau}d\tau$$

donde:

1. $G(t, f)$ es la representación tiempo-frecuencia de la señal $x(t)$.
2. $x(\tau)$ es la señal en el dominio del tiempo.
3. $g(\tau - t)$ es una función de ventana centrada en el tiempo t.
4. f es la frecuencia.
5. j es la unidad imaginaria.
6. $*$ denota el conjugado complejo.

- **La Transformada Wavelet:** Es una técnica de análisis de señales que se emplea para estudiar señales en el dominio tiempo-escala. Para que una onda sea considerada una wavelet, debe cumplir con ciertas condiciones:

  1. Debe ser oscilatoria: La función wavelet debe oscilar entre valores positivos y negativos en el tiempo.
  2. Debe decaer rápidamente a cero: La amplitud de la wavelet debe decrecer rápidamente a medida que se aleja de su punto central, asegurando que tenga una duración finita en el tiempo.
  3. Debe tener un valor medio nulo: El valor medio de la wavelet sobre todo su dominio debe ser cero, indicando que su energía se concentra en fluctuaciones locales en lugar de en una componente de corriente continua.

  Dentro de las diversas aplicaciones de la transformada wavelet, se destacan las familias de wavelets, que se agrupan según sus propiedades específicas para diferentes tareas:

  Wavelets para análisis discreto:

  - Daubechies: Excelente para compresión y análisis multirresolución.
  - Biortogonales: Adecuados para reconstrucción exacta de señales.

  Wavelets para análisis continuo:

  - Morlet: Ideal para análisis tiempo-frecuencia detallado.
  - Meyer: Adecuada para señales suaves con variaciones graduales.

  La elección depende del tipo de señal y objetivo: compresión, eliminación de ruido o detección de características clave. Esto asegura una representación eficiente de datos.

5.2.5. Procesamiento digital de señales de ECG.

Consiste en un plan que dependiendo de qué técnica se usa, será de forma cíclica o lineal, sin embargo, los pasos siempre son los mismos, se basa en los conceptos generales de la adquisición de señales mediante una base de datos o muestras, el procesamiento para el acondicionamiento de las señales y el procesamiento de estas para la adquisición de patrones característicos de la señal ECG.

5.2.6. Machine Learning y la prevención de enfermedades cardiovasculares

El Machine Learning (ML) es una disciplina que se encuentra dentro del campo de la Inteligencia Artificial (IA). El ML se basa en el suministro de una gran cantidad de datos que atraviesan etapas de procesamiento, análisis y aprendizaje de patrones. Uno de los ejemplos más conocidos es el de los clasificadores de imágenes de animales, como la capacidad de aprender a distinguir un gato de un perro mediante la identificación de patrones, como formas, filtros y ejes, lo cual conduce a la toma de decisiones[26].

Los algoritmos de Machine Learning se dividen en tres categorías aprendizaje supervisado, no supervisado y por refuerzo.

- **Aprendizaje Supervisado:** Cuentan con un aprendizaje previo basado en un sistema de etiquetas que está vinculado a datos, lo que les permite tomar decisiones o hacer predicciones. El principal objetivo es construir un modelo que sea capaz de realizar predicciones ya sea con datos nuevos o no conocidos a partir de un conjunto de datos [26]. Las técnicas usadas son clasificación y regresión. Los algoritmos más usados en el aprendizaje supervisado son, Linear regression, logistic regression, random forest, gradient boosted trees, support vector machines, decisión trees, K-nearest neighbor.
- **Aprendizaje no supervisado:** A diferencia de los algoritmos de aprendizaje supervisado, estos no cuentan con un aprendizaje previo, es decir, datos sin etiquetar o datos de estructura desconocida, el objetivo es de examinar la estructura de la información, para extraer información importante, descubrir patrones y categorizarlos [27]. Las técnicas usadas es el clustering y reducción de dimensionalidad. Los algoritmos más usados es el K-means clustering.
- **Aprendizaje por refuerzo:** El objetivo de este algoritmo es de que aprenda partir de su propia experiencia, que sea capaz de tomar las mejores decisiones ante diferentes situaciones con un principio básico de prueba y error donde el peso de las correcta aumenta y penaliza las incorrectas.

Para la detección de trastornos de ritmo cardíaco se debe enfocar en la obtención, análisis y predicción de la señal ECG, siendo esta el dato básico para iniciar un proceso de aprendizaje automatizado. En el mundo de ML se tienen aplicaciones de redes neuronales y Deep Learning para crear clasificadores de detección [28].

5.2.7. Técnicas de Machine Learning en señales ECG.

Las redes neuronales artificiales (ANNs), por sus siglas en inglés, son una herramienta que tiene la capacidad para aprender, generalizar los datos de entrada en diferentes clases, y en la regresión o aproximación de función, predecir un parámetro de salida desconocido. El potencial de las ANNs se encuentra en el reconocimiento de patrones y la predicción de comportamientos [28].

Existen diversos modelos que se pueden emplear para obtener estos resultados, tales como:

- **Multilayer perceptrón (MLP):** es una ANN compuesta por múltiples capas de neuronas ubicadas entre la entrada y la salida, conocidas como capas ocultas. La adición de estas capas permite aumentar el potencial de la red para resolver problemas de clasificación y regresión con conjuntos de datos complejos, que pueden ser no lineales y discontinuos. El algoritmo de entrenamiento más ampliamente utilizado para esta red se conoce como retro propagación (backpropagation o BP).
- **Red neuronal de Hopfield:** la ANN introducida por J. Hopfield funciona como una memoria asociativa en la que cada neurona está conectada a todas las demás neuronas. Durante el proceso de entrenamiento, se proporcionan todos los ejemplos de entrada, y se espera que la red almacene y memorice estos patrones. Una vez que la red ha sido entrenada, su capacidad de memoria asociativa se utiliza para recordar y reconocer los patrones almacenados, incluso en presencia de ruido [29].
- **Las Redes Neuronales Convolucionales:** se encuentran su inspiración en la corteza visual del cerebro humano. Esta corteza se caracteriza por tener mapas de campos receptivos locales que responden a estímulos solo en regiones específicas del campo visual y disminuyen su respuesta a medida que se avanza en la corteza visual. Además, estos campos receptivos se superponen para cubrir todo el campo visual. Estas redes toman como base la arquitectura de las redes neuronales multicapa (DMLP), pero su principal diferencia radica en que cada neurona se conecta únicamente con un parche local de neuronas en la capa siguiente [29].
- **Las Redes Neuronales Recurrentes:** tienen una arquitectura en la que las conexiones entre las neuronas forman un gráfico dirigido a lo largo de una secuencia temporal. Esto les permite ser especialmente útiles para analizar datos en series de tiempo, mostrar un comportamiento dinámico, tener una memoria interna y retener información sobre lo que ocurrió en pasos de tiempo anteriores [30].
- **La red de memoria a corto-largo plazo (LSTM):** Las redes LSTM (Long Short-Term Memory) son un tipo de red neuronal recurrente diseñada para procesar datos en secuencia y recordar información a lo largo del tiempo. Esto las hace ideales para tareas como análisis de texto, predicciones en series temporales y reconocimiento de patrones en datos secuenciales. Utilizan una estructura con compuertas (como la de olvido y actualización) que permiten controlar qué información se mantiene o elimina, imitando cómo los humanos procesamos solo datos relevantes mientras desechamos lo innecesario [31].

Ahora bien, otra técnica de gran utilidad en el área del machine learning es el aprendizaje por transferencia, que es una herramienta donde un modelo entrenado en una tarea se ajusta para realizar una tarea similar, aprovechando el conocimiento adquirido previamente. Esto reduce el tiempo y los recursos necesarios para entrenar nuevos modelos, ya que utiliza datos más pequeños y menos potencia computacional. Además, permite adaptarse a diversos contextos

con mayor eficiencia, mejorando la accesibilidad y el rendimiento en aplicaciones como el reconocimiento de imágenes o el análisis de datos médicos [32].

Dentro de esta técnica existen varias redes que han sido referencia, una de ellas es MobileNet-v2, que es una red neuronal convolucional con 53 capas de profundidad. Puede cargar una versión preentrenada de la red entrenada en más de un millón de imágenes desde la base de datos de ImageNet. La red preentrenada puede clasificar imágenes en 1000 categorías de objetos (por ejemplo, teclado, ratón, lápiz y muchos animales). Como resultado, la red ha aprendido representaciones ricas en características para una amplia gama de imágenes. El tamaño de la entrada de imagen de la red es de 224 por 224 [33].

También, la arquitectura VGG-16 corresponde a un modelo de red neuronal convolucional (CNN) desarrollado por el Visual Geometry Group de la Universidad de Oxford. Esta arquitecura cuenta con 16 capas, divididas entre 13 capas convolucionales y 3 completamente conectadas. Su diseño utiliza capas convolucionales seguidas de agrupación máxima, aumentando progresivamente la profundidad para aprender características visuales jerárquicas. Aunque es más simple que modelos recientes, destaca por su eficacia en tareas como clasificación de imágenes y reconocimiento de objetos, siendo una opción confiable en aplicaciones de aprendizaje profundo por su versatilidad y desempeño [34].

5.2.8. Entorno de desarrollo y herramientas

La descripción detallada de las tecnologías, lenguajes de programación, bibliotecas y herramientas utilizadas en el desarrollo del proyecto es fundamental para comprender la estructura de la arquitectura del código y el manejo de los datos. Se presentan los componentes claves que conforman el entorno de desarrollo, como Python junto con sus bibliotecas especializadas, la implementación de entornos virtuales para una eficiente gestión de dependencias, y el uso del sistema de control de versiones Git.

- **Python:** Es un lenguaje de programación de alto nivel, interpretado y de propósito general. Es conocido por su simplicidad, legibilidad y su amplia comunidad de desarrolladores. Este lenguaje facilita la implementación de soluciones en ciencia de datos, inteligencia artificial, desarrollo web, automatización y más [35].
- **NumPy:** Es la librería fundamental para computación numérica en Python. Permite trabajar con arreglos multidimensionales y realizar operaciones matemáticas de alto rendimiento [36].
- **Pandas:** es una biblioteca del lenguaje de programación Python, dedicada por completo a la Data Science que proporciona estructuras de datos flexibles y herramientas para manipulación y análisis de datos estructurados. Es ideal para la limpieza y transformación de datos [37].
- **Matplotlib:** Es una librería para la creación de gráficos estáticos, interactivos y animados en Python. Es ampliamente utilizada para la visualización de datos [38].

- **TensorFlow:** Es una biblioteca de código abierto para el aprendizaje automático y redes neuronales. Permite entrenar y desplegar modelos en diversas plataformas [39].
- **Flask:** Es un micro-framework de desarrollo web que facilita la creación de aplicaciones ligeras y escalables [40].
- **Virtualenv:** Es una herramienta que permite crear entornos virtuales aislados para proyectos de Python, evitando conflictos de dependencias [41].
- **Git:** Es un sistema de control de versiones distribuido que permite rastrear cambios en el código fuente, facilitando la colaboración y la gestión de proyectos [42].
- **GitHub:** Es una plataforma basada en Git que permite alojar repositorios, colaborar en proyectos y automatizar flujos de trabajo [43].
- **Visual Studio Code (VSCode):** Es un editor de código ligero, potente y extensible que soporta múltiples lenguajes, incluyendo Python, mediante plugins [44].

5.2.9. Desarrollo móvil.

El desarrollo nativo se refiere a crear aplicaciones específicamente diseñadas para una plataforma en particular, como iOS o Android, utilizando los lenguajes de programación y las herramientas de desarrollo propias de esa plataforma. Por ejemplo, para desarrollar una aplicación nativa para iOS, se utilizaría principalmente el lenguaje de programación Swift y las herramientas de desarrollo de Apple, como Xcode.

En contraste, el desarrollo multiplataforma implica crear aplicaciones que puedan funcionar en múltiples plataformas (como iOS, Android, web, escritorio, etc.) utilizando un único código base. Esto puede lograrse mediante el uso de frameworks y tecnologías que permiten escribir el código una vez y luego compilarlo para ejecutarse en diferentes plataformas. Ejemplos de herramientas para el desarrollo multiplataforma incluyen React Native, Flutter, Xamarin y Ionic [45].

- **Desarrollo Nativo:**
  Ventajas.
  1. Rendimiento óptimo: Las aplicaciones nativas están optimizadas para la plataforma específica, lo que proporciona un rendimiento rápido y fluido.
  2. Acceso total a las características de la plataforma: Se puede aprovechar al máximo el hardware y las API específicas de cada plataforma.
  3. Mejor experiencia de usuario: Al estar diseñadas específicamente para cada plataforma, las aplicaciones nativas suelen ofrecer una experiencia de usuario más intuitiva y coherente.

  Desventajas.
  1. Costos y tiempo de desarrollo más altos: Es necesario desarrollar y mantener diferentes versiones de la aplicación para cada plataforma.

2. Mayor curva de aprendizaje: Se requiere conocer los lenguajes y entornos de desarrollo específicos de cada plataforma (por ejemplo, Java para Android, Swift para iOS).

Herramientas comunes.

1. **Android:** Kotlin, Java, Android Studio.
2. **iOS:** Swift, Objective-C, Xcode.

- **Desarrollo Multiplataforma:**
Ventajas.

1. Menor costo y tiempo de desarrollo: Se puede escribir código una vez y ejecutarlo en múltiples plataformas, lo que reduce los costos y el tiempo de desarrollo.
2. Facilidad de mantenimiento: Al tener un único código base, las actualizaciones y correcciones se pueden implementar de manera más eficiente en todas las plataformas.
3. Amplio alcance: Permite llegar a una mayor audiencia al abarcar múltiples plataformas con un solo desarrollo.

Desventajas

1. Rendimiento potencialmente inferior: Las aplicaciones multiplataforma pueden experimentar un rendimiento ligeramente inferior en comparación con las nativas debido a la capa de abstracción adicional.
2. Limitaciones en el acceso a las características específicas de la plataforma: Aunque los frameworks multiplataforma intentan abstraer las diferencias entre las plataformas, es posible que no se pueda acceder a todas las características de manera nativa.

Herramientas comunes.

1. **Flutter:** Framework de Google basado en Dart.
2. **React Native:** Framework de Facebook basado en JavaScript.
3. **Xamarin:** Plataforma de Microsoft basada en C.

5.3. Trabajos Relacionados

La identificación y detección de arritmias cardíacas ha sido objeto de investigación en diversos ámbitos, desde la medicina clínica hasta la ingeniería biomédica. A continuación, se presentan algunos trabajos relevantes que abordan temáticas similares a la propuesta de este proyecto:

Diseño de una Herramienta para la Detección de Arritmias Cardíacas en Electrocardiogramas utilizando Técnicas de Aprendizaje Automático[14].

En este proyecto, realizado por R. Álvarez Patiño y A. Ruiz Luna. (2023), se aborda la problemática de la detección automática de arritmias cardíacas mediante técnicas de aprendizaje

automático aplicadas a los electrocardiogramas (ECG). El desarrollo de este proyecto se justifica en la complejidad y la subjetividad del diagnóstico manual de las arritmias y su objetivo es desarrollar un modelo que pueda identificar automáticamente estos trastornos a partir de los datos extraídos de un ECG. Para su ejecución se utiliza una base de datos de 10,646 pacientes anonimizados, categorizados según once tipos de arritmias previamente etiquetadas por médicos especialistas, se emplearon técnicas de aprendizaje supervisado y no supervisado. Aunque las técnicas no supervisadas no lograron resultados concluyentes, los algoritmos supervisados, como Support Vector Machine, Random Forest y Deep Learning, mostraron prometedores niveles de precisión.

Aprendizaje automático para la clasificación de arritmias cardíacas[13].

En este estudio llevado a cabo por G. Llodrà Bisellachse. (2018), compararon diferentes algoritmos de aprendizaje automático para la identificación de arritmias cardíacas a partir de señales de ECG. Se parte de la premisa de que las enfermedades cardiovasculares son una de las principales causas de muerte en el mundo, siendo las arritmias una de las manifestaciones más comunes y preocupantes dentro de estas enfermedades. El proyecto se centra en evaluar la capacidad de diferentes técnicas de aprendizaje automático, como redes neuronales y redes convolucionales, para clasificar segmentos de ECG como normales o arrítmicos. Se realiza un proceso de preprocesamiento de datos antes de aplicar los modelos de aprendizaje automático, lo que incluye normalización, segmentación y etiquetado de los datos. Los resultados obtenidos muestran que las redes convolucionales, especialmente aquellas con un mayor número de capas, ofrecen resultados prometedores en la detección de arritmias.

Aplicación para la detección de arritmias basado en los algoritmos de Pan Tompkins, Elgendi y Boonperm [15].

En este trabajo realizado por L.M. Casas Quiroz (2021), se diseñó y desarrolló una aplicación para la detección temprana de arritmias basada en los algoritmos Pan & Tompkins, Elgendi y las reglas de decisión de Parayikorn. El proyecto se llevó a cabo con el objetivo de ofrecer una alternativa para la detección precoz de arritmias, dado que los monitores de funciones vitales tradicionales son costosos, no portables y su software no es accesible. Mediante la implementación de estos algoritmos y su evaluación con la base de datos MIT-BIH Arrhythmia Database, se logró una sensibilidad del 37.22 % y una predictividad del 65.09 % en la detección de arritmias. Aunque estos resultados preliminares indican una clasificación aún poco precisa de las arritmias, se destaca una alta sensibilidad del 99.02 % y una predictividad del 99.11 % en la detección de los complejos QRS, lo que sugiere una confiabilidad en esta área específica. Como trabajos futuros, se propone desarrollar una aplicación para visualizar el electrocardiograma en tiempo real y expandir el alcance del algoritmo para detectar otras anomalías cardíacas, como el infarto agudo al miocardio, utilizando bases de datos adicionales del portal Physionet.

Comparación de modelos de inteligencia artificial para el diagnóstico de arritmia

cardiaca mediante información de electrocardiogramas [16].

En este trabajo realizado por C. C. Gelves Higuera. (2012), se desarrolló un estudio de benchmarking de modelos de inteligencia artificial (IA) para determinar la presencia de arritmia cardiaca a partir de datos de electrocardiogramas (ECG). Este proyecto se llevó a cabo con el propósito de identificar el modelo de IA con mayor precisión en el diagnóstico de arritmias, utilizando metodologías de clasificación como reconocedor euclidiano y KNN. Como resultado de este estudio, se observó que los modelos sin reducción de dimensionalidad mostraron una media de clasificación máxima más alta en comparación con aquellos que aplicaron métodos de reducción. Sin embargo, se destacó que los modelos con reducción de dimensionalidad lograron valores máximos superiores en ejecuciones específicas. Además, se encontró que el reconocedor euclidiano superó al método KNN en términos de precisión para la clasificación de ECG.

Arritmias ventriculares y su relación con la muerte súbita cardíaca en deportistas [17].

En este trabajo realizado por los autores A. E Veloz Toaquiza y A.A Tufiño Aguilar (2023) se realizó un estudio sobre la muerte súbita cardíaca y las arritmias ventriculares, con un enfoque particular en su relación con los deportistas. Estos hallazgos representan un avance significativo en nuestra propuesta de identificar las arritmias responsables de la muerte súbita cardíaca. Los resultados preliminares de este estudio muestran la relevancia de un enfoque multidisciplinario en el tratamiento de las arritmias ventriculares en deportistas, así como la eficacia de opciones terapéuticas como la ablación con catéter y el uso de desfibriladores automáticos implantables. Este aporte nos permite avanzar hacia una mejor comprensión y gestión de estas condiciones cardíacas críticas.

Un marco novedoso de aprendizaje automático para la detección automática de arritmias en segmentos de ECG [?].

En este estudio realizado por T.-H. Pham et al. (2021), se presenta un marco de aprendizaje automático diseñado para la detección automática de arritmias en segmentos de electrocardiogramas (ECG). El objetivo principal del trabajo es mejorar la precisión y la eficiencia en la clasificación de arritmias, aprovechando un enfoque que combina preprocesamiento avanzado de datos y modelos robustos de aprendizaje automático. El marco propuesto emplea una arquitectura híbrida que incluye técnicas de extracción de características, como la transformada wavelet y cumulantes de tercer orden, y modelos de clasificación basados en redes neuronales profundas (DNN) y máquinas de soporte vectorial (SVM). Además, se implementó la técnica ADASYN para balancear las clases mediante la generación sintética de muestras adicionales. Entre las características extraídas se incluyeron 18 medidas no lineales relacionadas con entropías y texturas, de las cuales las más significativas se seleccionaron utilizando la prueba t de Student.

El sistema fue diseñado para clasificar ritmos normales y diferenciarlos de arritmias como la Fibrilación Auricular, el Aleteo Auricular y la Fibrilación Ventricular, que son reconocidas como indicadores tempranos de accidentes cerebrovasculares y muerte súbita cardíaca, dos de

las principales causas de mortalidad a nivel global. La evaluación del sistema se llevó a cabo utilizando la base de datos MIT-BIH Arrhythmia Database, con segmentos de ECG de 2 y 5 segundos. Entre los clasificadores probados, el modelo Random Forest demostró el mejor desempeño, alcanzando una precisión global del 98.2 %, sensibilidad del 98.1 % y especificidad del 99.4 % para segmentos de 2 segundos, y mejorando a una precisión del 98.8 %, sensibilidad del 98.8 % y especificidad del 99.6 % para segmentos de 5 segundos. De este estudio, se destaca la importancia dela naturaleza intermitente de las arritmias, y se propone que el análisis de señales ECG de mayor duración mejora la detección de episodios críticos. Este sistema, diseñado para trabajar con segmentos más largos en lugar de latidos individuales, tiene una mejor adaptabilidad clínica y puede integrarse en sistemas de monitoreo en tiempo real. En general, este marco representa una herramienta confiable y eficiente para la detección temprana de arritmias, optimizando así las capacidades de los sistemas de soporte de decisiones clínicas (CDSS).

Estos trabajos proporcionan una base de conocimientos y experiencias previas en el tema de las arritmias cardíacas relacionadas con la MSC y su identificación mediante el uso de tecnologías de inteligencia artificial y procesamiento de señales. Su análisis y comparación ayudarán a contextualizar y fundamentar el presente proyecto de investigación.

Capítulo 6

Materiales y Métodos

El proyecto "Desarrollo de un aplicativo móvil para la identificación de arritmias cardíacas mediante procesamiento digital de señales ECG y aprendizaje automático" se enmarca como un estudio de desarrollo aplicado en el campo de la ingeniería biomédica y la inteligencia artificial. En el desarrollo de este proyecto se emplea un enfoque integral de conocimientos en procesamiento digital de señales biológicas, aprendizaje automático y desarrollo de aplicaciones móviles para el cuidado de la salud. Se propone la implementación de un aplicativo móvil destinado a la identificación de arritmias cardíacas en pacientes con antecedentes cardiovasculares o enfermedades crónicas, que puede alertar sobre la posibilidad de Muerte Súbita Cardíaca. La metodología se desarrolla mediante fases, que van desde la búsqueda de la base de datos de arritmias hasta la validación de funcionamiento de la app con simuladores de ECG. Es importante destacar que, este estudio no involucra experimentación con seres humanos en el sentido tradicional, sino que se enfoca en el desarrollo y la evaluación de algoritmos y aplicaciones basadas en datos previamente recopilados y disponibles. La base de datos utilizada fue depurada y estructurada previamente para garantizar la calidad y la confiabilidad de los resultados obtenidos. Además, se puso especial atención en las implicaciones éticas y legales relacionadas con el manejo de datos de salud sensibles, asegurando el cumplimiento de las regulaciones pertinentes, como la protección de datos personales y la confidencialidad de la información médica que requiera el aplicativo diseñado en trabajos futuros.

Figura 6.1: Metodología de desarrollo de un aplicativo móvil para la identificación de arritmias cardíaca.

En el esquema de la Figura 6.1 se muestra cada una de las fases que se llevan a cabo para el desarrollo de este proyecto. La metodología estuvo soportada en el desarrollo de actividades progresivas que permitieron realizar con éxito cada uno de los objetivos específicos planteados en esta propuesta, teniendo entonces 5 fases: Búsqueda de base de datos; Extracción de características de la señal ECG; Entrenamiento de algoritmos de aprendizaje automático; Evaluación de modelos entrenados e Implementación de un aplicativo móvil.

A continuación, se describen cada una de las fases de este proyecto y los respectivos métodos que se emplearon para ejecutarlas de manera exitosa.

6.1. Fase 1: Búsqueda de base de datos.

La Fase 1, en la que se da respuesta al objetivo específico de gestionar una base de datos de señales ECG depurada y estructurada para facilitar el análisis y procesamiento de la señales en la identificación de arritmias cardíacas, se plantearon 4 actividades.

6.1.1. Consultar y seleccionar diversas bases de datos de señales ECG disponibles en repositorios gratuitos, considerando la calidad, cantidad y diversidad de las señales.

Para llevar a cabo esta actividad, se realizó la búsqueda de artículos y trabajos que han tenido como objetivo clasificar señales de ECG, en especial aquellos, que han identificado distintas arritmias por medio de modelos de inteligencia artificial. Uno de los hallazgos más relevantes en esta etapa fue una revisión de literatura y recopilación de trabajos que registra desde el año 2017 hasta 2023 [46], donde se presentan distintos trabajos que contienen información relacionada con la detección y clasificación de anomalías del ECG, también se proporciona una descripción detallada de las arritmias, sus tipos y clases utilizadas en aplicaciones de detección y clasificación por medio de las técnicas de aprendizaje automático que han dado mejores resultados. Adicionalmente, se presenta una revisión de datos estándar que se utilizan tradicionalmente para entrenar y validar clasificadores Deep Learning.

También el estudio de Xiao et al. (2023) [47] analiza específicamente cómo las técnicas de Deep Learning han avanzado en la clasificación de arritmias utilizando señales de ECG, destacando los métodos más efectivos y los resultados obtenidos en diversas investigaciones. Este estudio hace una revisión sistemática de cómo el aprendizaje profundo (DL) ha sido utilizado para la clasificación de arritmias a partir de señales de electrocardiograma (ECG), analizando diversos enfoques propuestos en la literatura. El artículo resalta el uso de bases de datos estándar como MIT-BIH y Fantasia, que proporcionan datos etiquetados sobre arritmias, lo que facilita la evaluación y el entrenamiento de modelos. Asimismo, se aborda la importancia de las métricas de evaluación, como la precisión, la sensibilidad y la puntuación F1, para medir el rendimiento de los modelos de DL en la clasificación de arritmias.

Para la revisión de literatura y búsqueda de información relacionada con bases de arritmias comúnmente utilizadas en el ámbito de la inteligencia artificial para la tarea de clasificación, se hizo uso de las bases de datos científicas, PubMed, Elsevier, SciELO, Google Scholar, entre otras, donde se realizó la búsqueda prestando especial atención en las bases de datos que estos trabajos emplearon para entrenar sus modelos de machine learning.

Ahora bien, dentro de esta fase se relacionó la información obtenida de la revisión de literatura, específicamente la que se trata de los conjuntos de datos que se han utilizado en los distintos estudios y que resultados positivos o negativos se han obtenido a partir de la utilización de ellos. Estos conjuntos de datos consisten en señales de ECG recopiladas que tienen anotaciones de los eventos y condiciones cardíacos correspondientes. Con la información proporcionada en los trabajos de referencia se consolidó un listado de criterios que debería tener la base de datos para el entrenamiento del modelo de inteligencia artificial de esta propuesta. Dentro de ellos, la disponibilidad de etiquetas de arritmias, el formato de los datos, la cantidad de muestras, la representatividad de la población, la duración de los segmentos de señales, frecuencia de muestreo, número de canales por medio de los cuales se obtuvo la señal ECG. Algunos criterios adicionales, que se lograron definir en esta etapa, están relacionadas con el uso libre de la base de datos y la metodología que utilizó cada uno de los trabajos para obtener los datos y conformar la base de datos que se utilizó en el desarrollo de esta herramienta.

6.1.2. Comparar las bases de datos seleccionadas para determinar cuál es la más adecuada para el desarrollo del trabajo.

De acuerdo con la revisión de literatura en bases de datos científicas, que permitieron extraer información de repositorios con señales ECG de forma continúa para el entrenamiento de modelos de inteligencia artificial, se logra realizar una tabla comparativa, donde se tienen en cuenta la inclusión de etiquetas de arritmias, la duración de los segmentos de señal, el muestreo y adicionalmente, para tomar la decisión, se tiene en cuenta el proceso de adquisición de la señales de ECG que se empleó en el estudio para la formación de la base de datos, al igual que la representatividad de la población monitoreada.

Cuadro 6.1: Resumen de bases de datos utilizadas en estudios de análisis de arrtimias.

Base de Datos	**Etiquetas de Arritmias**	**Duración de los Segmentos**	**Frecuencia de muestreo (Hz)**	**Población representativa**
Base de datos de taquiarritmia ventricular de la Universidad de Creighton (CUDB)	Sí	8 min	250	Pacientes con riesgo de arritmias ventriculares
Base de datos de pruebas de estrés acústico del MIT-BIH (NSTDB)	Sí	30 min	360	Adultos bajo pruebas de estrés
Base de datos de arritmias de 12 derivaciones INCART de San Petersburgo (INCARTDB)	Sí	30 min	257	Pacientes hospitalizados en Rusia
Base de datos de FA a largo plazo (LTAFDB)	Sí	24-25 hrs	128	Adultos mayores con FA recurrente
Base de datos de arritmias del MIT-BIH	Sí	30 min	360	Pacientes hospitalizados (adultos)
Base de datos de fibrilación auricular del MIT-BIH	Sí	10 hrs	250	Pacientes con FA crónica
Base de datos de ritmo sinusal normal del MIT-BIH	No	30 min	360	Individuos sanos

Base de datos de ectopia ventricular maligna del MIT-BIH	Sí	30 min	360	Pacientes con antecedentes de paro cardíaco
Base de datos de arritmias supraventriculares (SVDB) del MIT-BIH	Sí	30 min	250	Adultos con arritmias SV documentadas
Base de datos Holter de Muerte Súbita Cardíaca (SCDDB)	Sí	Variable	250	Pacientes con riesgo de muerte súbita
Base de datos del intervalo RR del ritmo sinusal normal (NSRDB)	No	Variable	128	Individuos sin arritmias detectables
Base de datos de desafío de ECG de 12 derivaciones de Georgia (GA12ECG)	Sí	10 s	500	Adultos con diferentes cardiopatías
Base de datos de ECG de apnea	No	7-10 hrs	250	Pacientes con sospecha de apnea obstructiva
Base de datos de ECG de diagnóstico de PTB	Sí	10 s	1000	Pacientes con diversas cardiopatías
Base de datos europea ST-T	Sí	120 min	250	Pacientes europeos con enfermedad cardíaca coronaria
Computación PhysioNet en el desafío de cardiología 2017 (AFDB)	Sí	30-60 s	300	Pacientes con ritmos cardíacos anómalos

En la Tabla 6.1, se evidencia la caracterización de diversas bases de datos enfocadas en la identificación de arritmias cardíacas para entrenar modelos de inteligencia artificial. Entre las opciones analizadas, se eligió la base de datos MIT-BIH Arrhythmia Database como la fuente principal de datos debido a su reputación como la más completa y fiable dentro de la comunidad científica y médica.

Publicada originalmente en 1980 y posteriormente ampliada y distribuida gratuitamente en PhysioNet en 2005, esta base de datos ha sido utilizada en más de 500 investigaciones globales para la evaluación de detectores de arritmias, consolidándose como un estándar en el análisis de ritmos cardíacos.Esta decisión, estuvo soportada, además de los atributos de la base de datos, en tablas comparativas de una revisión [46], que demuestra el éxito que este conjunto de datos ha tenido en el ámbito del desarrollo de modelos de inteligencia artificial que clasifican arritmias a partir del procesamiento de señales ECG.

Los resultados de este estudio demuestra que aquellos trabajos que utilizaron la base de datos del MIT Arrhythmia Database lograron un mejor desempeño en términos de métricas como el accuaracy, la sensibilidad, la epecificidad y el F1 score. Esto se puede validar en los cuadros 6.3 y 6.2.

Study	**Database**	**# Cl**	**Classifier**	**F1-Score (%)**
Luz et al. (2016)	MIT-BIH	5	GRNN	99.00
Sujadevi et al. (2017)	MIT-BIH	4	GRU	99.99
Faust et al. (2018)	MIT-BIH	5	BiLSTM	98.00
Tan et al. (2018)	Fantasia + INCARTDB	2	CNN-LSTM	99.52
Xiang et al. (2018)	MIT-BIH	8	1D CNN	99.99
Hammad et al. (2020)	MIT-BIH	5	DNN	95.30
Mahmud et al. (2020)	MIT-BIH	5	1D CNN	99.10
Ullah et al. (2020)	MIT-BIH	8	2D CNN	98.00
Peimankar and Puthusserypady (2021)	QT DB	4	CNN-LSTM	99.56
Islam et al. (2022)	MIT-BIH	5	BiGRU + BiLSTM	98.41
Hong et al. (2022)	MIT-BIH	4	ECG Delineator	96.11
Zahid et al. (2022)	MIT-BIH	5	1D Self ONN	99.51
Kim et al. (2022)	MIT-BIH	5	ResNet + BiLSTM	99.20
Sepahvand and Abdali-Mohammadi (2022)	Chapman ECG DB	12	Distilled Models	97.55
Midani et al. (2023)	MIT-BIH	5	CNN + BiLSTM	97.63
Kumar et al. (2023)	MIT-BIH	5	Fuzz-ClustNet	96.34

Cuadro 6.2: F1-scores de modelos de aprendizaje profundo para la detección y clasificación de arritmias ECG.

Study	Database	# Cl	Classifier	Acc (%)	Se (%)	Sp (%)
Luo et al. (2017)	MIT-BIH	4	DNN-SDA	98.80	71.40	99.80
Majumdar and Ward (2017)	MIT-BIH	4	SVM-RBF	97.00	100.00	90.12
Zhang et al. (2017)	MIT-BIH	5	RNN	99.40	97.60	99.70
Xia et al. (2017)	MIT-BIH	3	CNN	98.63	98.79	97.87
Nguyen et al. (2018)	CUDB, MIT-BIH (VFDB)	2	FCN	99.26	97.07	99.44
Jun et al. (2018)	MIT-BIH	4	2D CNN	99.05	99.57	97.85
Yildirim (2018)	MIT-BIH	4	Bi-directional LSTM	99.39	95.66	98.11
Sannino and De Pietro (2018)	MIT-BIH	4	DNN	99.68	99.48	99.83
Faust et al. (2018)	MIT-BIH	5	BiLSTM	98.51	98.32	98.67
Xia and Xie (2019)	MIT-BIH	4	1D CNN + Active Learning	99.20	95.73	98.73
Lui and Chow (2018)	MIT-BIH	4	ML-CNN	96.00	95.40	97.37
Xia et al. (2018)	MIT-BIH Wearable Device	4	DNN	99.80	99.40	99.90
Wang et al. (2019)	MIT-BIH	2	GRNN	97.40	86.70	98.30
Hanbay (2019)	MIT-BIH	4	DNN	96.40	86.41	96.41
Wang and Zhou (2019)	BIDMC-CHF, MIT-BIH NSR, Fantasia	5	LSTM	99.22	99.22	99.72
Chen et al. (2020)	MIT-BIH	4	CNN-LSTM	99.32	97.50	98.70
Fu et al. (2020)	PTB	6	CNN-BiGRUt	99.11	99.02	98.23
Sharma et al. (2021)	MIT-BIH	5	SVM + FFBPNN	98.53	98.24	95.68
Ojha et al. (2022)	MIT-BIH	4	CNN-SVM	99.53	98.24	97.58
Sepahvand and Abdali-Mohammadi (2022)	Chapman ECG DB	12	Distilled Models	98.15	97.11	98.45
Midani et al. (2023)	MIT-BIH	5	CNN + BiLSTM	99.46	97.01	99.57
Kumar et al. (2023)	MIT-BIH	5	Fuzz-ClustNet	98.66	98.92	93.88

Cuadro 6.3: Resumen de estudios con base de datos, número de clases, clasificadores y métricas de desempeño.

La base de datos MIT-BIH Arrhythmia Database se compone de 48 grabaciones de dos canales de ECG, cada una de aproximadamente 30 minutos de duración, obtenidas de una población representativa de pacientes hospitalizados (60 %) y ambulatorios (40 %). Estas grabaciones incluyen más de 110,000 anotaciones validadas por al menos dos cardiólogos inicialmente en su creación, cubriendo una amplia variedad de arritmias cardíacas. Además, aunque las arritmias están etiquetadas según el estándar AAMI EC57, la flexibilidad de esta base permitió adaptar las etiquetas a los objetivos específicos del proyecto, centrándose en ritmos críticos relacionados con la muerte súbita cardíaca, como fibrilación ventricular, fusión de ritmo, contracción ventricular prematura, fusión ventricular y ritmos normales.

El acceso abierto, la calidad de las anotaciones, la diversidad de los datos y su uso extendido en el entrenamiento de algoritmos de aprendizaje automático hacen de la base de datos MIT-BIH Arrhythmia una elección idónea para alcanzar el objetivo general de este proyecto: desarrollar un aplicativo móvil basado en procesamiento digital de señales ECG y aprendizaje automatizado que permita identificar arritmias cardíacas y alertar sobre la posibilidad de muerte súbita cardíaca en pacientes con antecedentes cardiovasculares o enfermedades crónicas.

6.1.3. Implementar un proceso de carga de la base de datos seleccionada y desarrollar un script en Python para visualizar las señales ECG y explorar su distribución y características.

La base de datos MIT-BIH Arrhythmia Database es una colección de grabaciones de señales ECG ampliamente utilizada en investigaciones médicas y desarrollos tecnológicos relacionados con el análisis de ritmos cardíacos. Fue publicada originalmente en 1980 y se ha mantenido como una de las referencias más robustas y confiables en la comunidad científica. La base de datos está disponible en el repositorio PhysioNet, una plataforma de acceso libre que proporciona datos fisiológicos para la investigación [48].

El proceso de carga de la base de datos en un entorno de desarrollo implica utilizar herramientas como Visual Studio Code (VScode). En este caso, se procede a cargar la carpeta que contiene el conjunto de datos, asegurándose de que todos los archivos, incluyendo los archivos de anotación .hea, que contienen información clave para la visualización y el análisis de las señales ECG, se carguen correctamente. Estos archivos .hea son esenciales, ya que contienen las cabeceras de los registros, incluyendo información sobre el formato de los datos, la duración de la grabación, y las etiquetas de los eventos cardíacos anotados.

Para realizar este cargue de bases de datos al editor de código fuente, VScode, fue necesario instalar librerías *wfdb, numpy y matplotlib.*

```python
import wfdb
import numpy as np
import matplotlib.pyplot as plt
# Leer un registro de la base de datos
record = wfdb.rdrecord('mit-bih-arrhythmia-database-1.0.0/101')
annotation = wfdb.rdann('mit-bih-arrhythmia-database-1.0.0/101', 'atr')
# Obtener la se al y las anotaciones
signal = record.p_signal[:, 0]  # Tomamos solo una de las derivaciones
fsOriginal = record.fs  # Frecuencia de muestreo
ann_sample = annotation.sample  # Muestras donde hay anotaciones R
ann_symbol = annotation.symbol  # S mbolos de anotaciones
duration = 10
fsNew = 150
```

Listing 6.1: Cargar primer Record

Para poder visualizar este primer record usamos matplotlib para la gráfica.

```python
segment_duration = 10 * fsOriginal
num_segments = len(filtered_signal) // segment_duration
for i in range(1):
    start = i * segment_duration
    end = start + segment_duration
    segmentFilter = filtered_signal[start:end]
    segmentNoFilter = signal[start:end]
    # Crear el eje de tiempo para el segmento actual
    t = np.arange(start, end) / fsOriginal
    # Graficar el segmento
    plt.figure(figsize=(15, 5))
    plt.plot(t, segmentNoFilter, c='g', label='Filtrada')
    plt.plot(t, segmentFilter, c='b', label='Original', alpha=0.5)
    # A adir anotaciones
    for j in range(len(ann_sample)):
        if start <= ann_sample[j] < end:
            plt.axvline(x=(ann_sample[j]) / fsOriginal, color='r',
                linestyle='--')
            plt.text((ann_sample[j]) / fsOriginal, max(segmentFilter),
                ann_symbol[j], color='r')
    plt.title(f'Se al ECG - Segmento {i+1}')
    plt.xlabel('Tiempo [s]')
    plt.ylabel('Amplitud [mV]')
    plt.legend()
    plt.show()
```

Listing 6.2: visualizar record

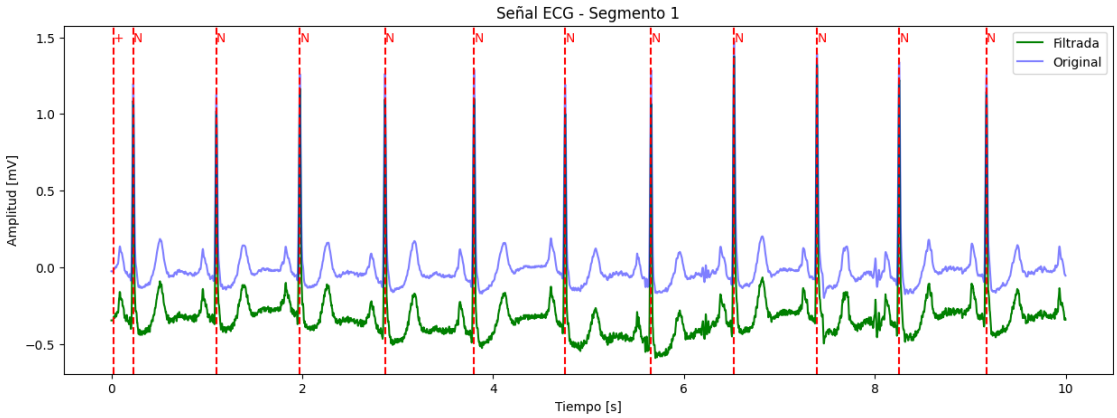


Figura 6.2: Señal ECG de la base de datos MIT-BIH

Al observar el *Listing6.1*, se utilizó la librería `wfdb` para cargar bases de datos. El tipo de archivo cargado es `.dat`, el cual contiene el vector de datos y sus anotaciones. Por otro lado, para visualizar las imágenes se utilizó la librería `matplotlib`, como se muestra en el *Listing 6.2*. Al ejecutar estos códigos, se obtiene la gráfica que se presenta en la Figura 6.2.

6.1.4. Utilizar herramientas de Python iniciar procesos de extracción caracteristicas de interes.

Una vez cargada la base de datos en el editor de código VScode, se tuvo un primer contacto con la información contenida en ella. Dentro de esa primera exploración, se importaron las bibliotecas necesarias para poder visualizar la información. Posterior a esa acción, se realizaron distintas técnicas para conocer el contenido de la base de datos. Empezando por técnicas de extracción de características en el dominio de tiempo, donde se detectaron intervalos de pruebas, como la detección de picos R, que es una técnica fundamental en el análisis de electrocardiogramas (ECG), utilizada para identificar la onda R, el punto más alto del complejo QRS en la señal del ECG. Este pico refleja la actividad eléctrica relacionada con la despolarización de los ventrículos, un momento crucial en el ciclo cardíaco. La identificación precisa de los picos R es clave para extraer características del ECG y clasificarlas, ya que ofrece datos temporales esenciales para calcular la frecuencia cardíaca y detectar posibles arritmias.

También se realizó la identificación de los intervalos R-R, que es el tiempo transcurrido entre dos ondas R consecutivas en el electrocardiograma, de modo que los datos pueden ser vistos como señales muestreadas con periodo de muestreo no constante. Finalmente, se detectan los complejos QRS, y otros segmentos importates dentro de la morfología de la señal ECG. Las líneas de código que ejecutan estas acciones se encuentran en los *Listing 6.3 y 6.4*.

```python
def detect_r_peaks(filtered_signal, fs):
    # Detectar picos en la se al filtrada
    distance = int(0.6 * fs)  # Aproximadamente la distancia m nima entre picos R
    peaks, _ = find_peaks(filtered_signal, distance=distance, height=np.mean(filtered_signal))
    return peaks

def calculate_rr_intervals(r_peaks, fs):
    # Calcular las diferencias entre picos consecutivos
    rr_intervals = np.diff(r_peaks) / fs  # Convertir a segundos
    average_rr_interval = np.mean(rr_intervals)  # Promedio de los intervalos RR
    return rr_intervals, average_rr_interval

def find_qrs_duration(signal, peaks, fs):
    qrs_durations = []
    for peak in peaks:
        start = peak
        end = peak
        while signal[start] >= 0 and start > 0:
            start -= 1
        while signal[end] >= 0 and end < len(signal) - 1:
            end += 1
        qrs_duration = (end - start) / fs
        qrs_durations.append(qrs_duration)
    average_qrs_durations = np.mean(qrs_durations)  # Promedio de los intervalos RR
    return qrs_durations,average_qrs_durations

def detect_qrs_complexes(r_peaks, signal):
    q_peaks = []
    s_peaks = []
    for r in r_peaks:
        # Q es el primer m nimo antes de R
        q_peaks.append(r - (len(signal[r-20:r]) - np.argmin(signal[r-20:r])))  # Aqu  30 es un ejemplo, ajustar seg n datos
        # S es el primer m nimo despu s de R
        s_peaks.append(r + np.argmin(signal[r:r+30]))  # Aqu  30 es un ejemplo, ajustar seg n datos
    return q_peaks, s_peaks
```

Listing 6.3: Definiciones para extracción de caracteristicas en el dominio del tiempo

```python
def detect_t_peaks(s_peaks, signal):
    t_peaks = []
    for r in s_peaks:
        # T es un m ximo despu s del complejo QRS, ajustar seg n datos
        t_peaks.append(r + np.argmax(signal[r:r+100]))  # Aqu  20 y 60 son ejemplos
    return t_peaks

def detect_p_peaks(r_peaks, signal):
    p_peaks = []
    for r in r_peaks:
        if r > 50:  # Aseg rate de que r-50 no sea negativo
            p_segment = signal[r-50:r-30]
            if len(p_segment) > 0:
                p_peak = r - 50 + np.argmax(p_segment)
                p_peaks.append(p_peak)
    return p_peaks
```

Listing 6.4: Definiciones para extracción de caracteristicas en el dominio del tiempo

A partir del análisis aplicado al vector de ECG y su respectiva graficación, se lograron identificar 8 picos R y un período promedio de 0.895 s en los intervalos R-R dentro de un lapso de 8 segundos, tal como se observa en la Figura 6.3.

Posteriormente, se construyó un diagrama de fase, una representación gráfica que permite analizar el comportamiento dinámico de la señal en el espacio de fases. Dicho diagrama facilita la visualización de cómo evoluciona la señal a lo largo del tiempo, revelando patrones y características importantes de su dinámica. Esta representación se puede apreciar en la Figura 6.4.

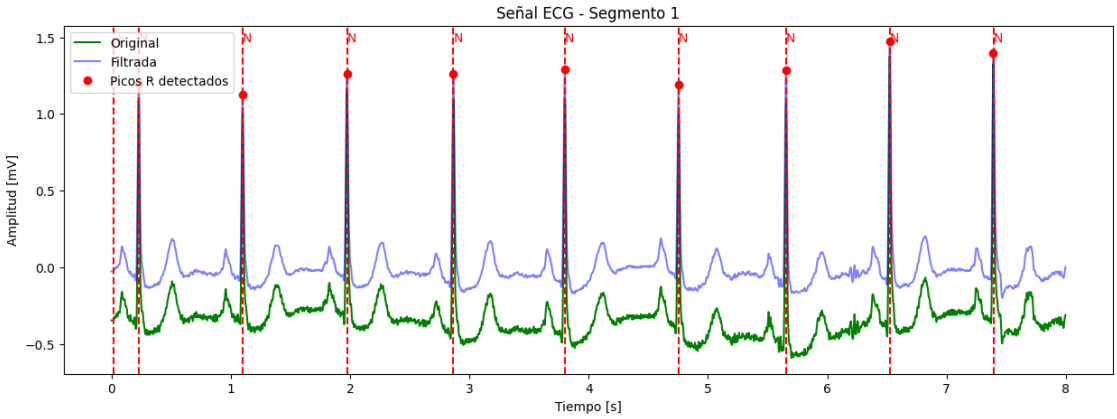


Figura 6.3: Señal ECG con detección picos R

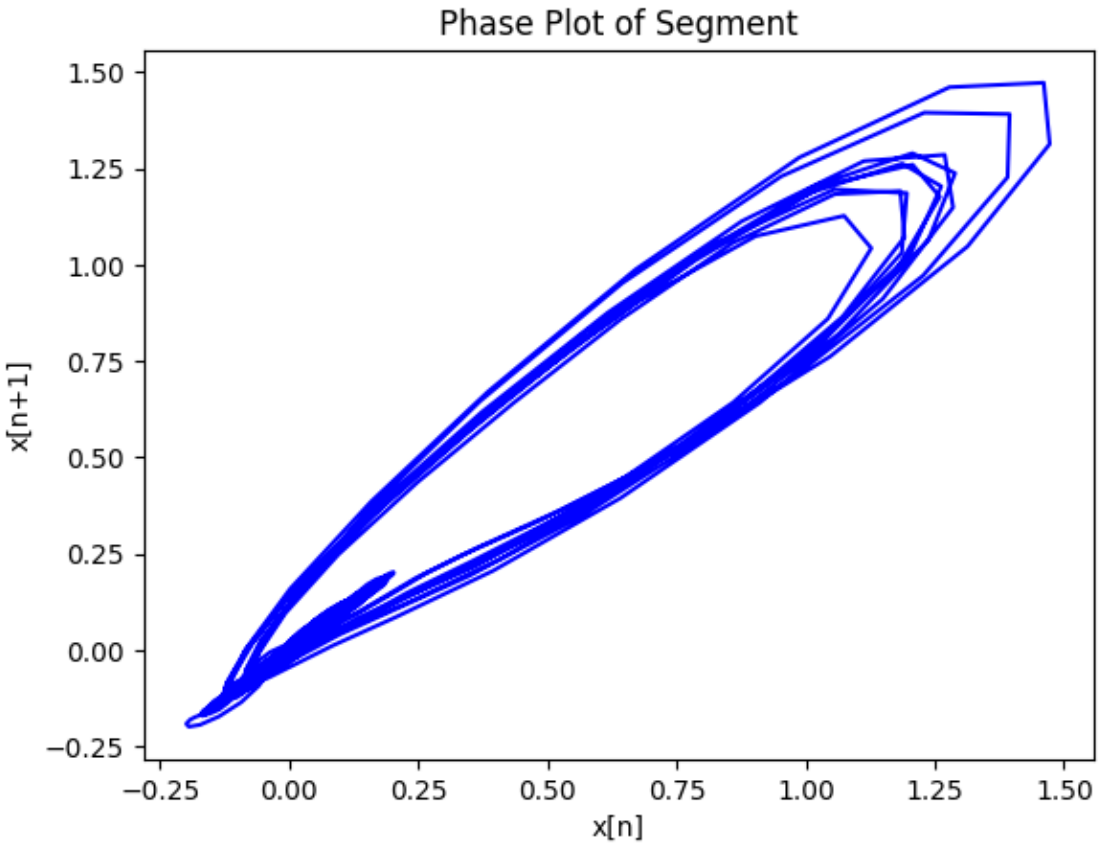


Figura 6.4: Señal ECG en diagrama de PHASE

6.2. Fase 2: Extracción características de las señales ECG.

En la fase 2, que da solución al objetivo de extraer características de las señales ECG mediante técnicas de procesamiento digital de señales para el entrenamiento de algoritmos de aprendizaje automático. Se realizaron las siguientes actividades:

6.2.1. Consultar y seleccionar técnicas de procesamiento de señales ECG, como filtrado, normalización y segmentación, para mejorar la calidad y uniformidad de los datos antes de la extracción de características.

Para esta etapa se realizó una revisión de literatura con el fin de verificar si existe una forma estándar o protocolo que permita realizar un procesamiento de la señal de manera efectiva, en términos de las señales que contiene la base de datos MIT-BIH de ECG. Por medio de esta revisión se encontró que existen estudios y trabajos que coinciden con un paso a paso para realizar la segmentación de las señales, este estudio fue realizado por Mohammad Kachuee

et al [49]. El protocolo fue implementado con el fin de extraer un segmento o método de segmentación de la señal que permita leerla y así identificar las características que esta posee de forma temporal, es decir, en intervalos.

A continuación, se describen los ocho pasos que permiten el procesamiento de los datos de ECG y la segmentación de los latidos.

1. Dividir la señal de ECG continua en ventanas de 10 segundos con una frecuencia de muestreo de 125Hz.
2. Hallar los valores máximos y mínimos de cada ventana de 10 segundos.
3. Escalar los valores de ECG en un rango de cero a uno, tomando como cero el valor mínimo y el valor máximo como uno, ver ecuación 6.1 Xi es el valor perteneciente a cada uno de los datos de la señal ECG, Vmin es el valor mínimo tomado en la ventana de 10 segundos y Vmax es el valor máximo.

$$V_{\text{escal},i} = \frac{X_i - V_{\min}}{V_{\max} - V_{\min}} \tag{6.1}$$

4. Encontrar el conjunto de todos los máximos (ECG R-Peak) de cada ventana de 10 segundos utilizando el cruce por cero de la primera derivada.
5. Validar si cada máximo (Vmax) —ECG R-Peak— es mayor o igual al 80 % del valor máximo.
6. Calcular el tiempo promedio entre picos (R-R) para cada ventana de 10 segundos.
7. Obtener muestras independientes de cada ventana, iniciando de un pico R hasta una longitud de 1.2 del tiempo promedio encontrado.
8. Establecer una longitud predefinida de 187 datos por cada muestra, si es menor a ese valor se rellenan con ceros.

Como paso adicional, para segmentar los datos de forma correcta, es necesario reclasificar las etiquetas correspondientes a las arritmias de interés. Las etiquetas seleccionadas para el desarrollo de esta propuesta son: *'V', '!', 'F', 'f', 'N'* mientras que todas las demás etiquetas que no forman parte del análisis se agrupan bajo la categoría *'O'*, representando otros. La correspondencias de estas etiquetas y tamañao de cada una puede verse a continuación en la tabla 6.4

En la Figura 6.5 se presentan las gráficas de los vectores correspondientes a las arritmias seleccionadas como objeto de estudio en esta propuesta, debido a su relación con la aparición de Muerte Súbita Cardíaca (MSC). Cada etiqueta muestra una forma de onda característica en un segmento específico, con diferencias notables en comparación con el ECG "Normal". Sin embargo, estas diferencias no son fácilmente distinguibles para el modelo de aprendizaje que se pretende realizar, lo que resalta la importancia de esta etapa de caracterización para garantizar el rendimiento del sistema.

Nombre de la arritmia	Etiqueta	Tamaño de muestra
Contracción Ventricular Prematura	V	7855
Onda de Aleteo Ventricular	!	154
Fusión de Ventricular y Normal	F	1135
Fusión de Rítmica y Normal	f	1335
Normal	N	59342
Otras	O	23521

Cuadro 6.4: Clasificación de arritmias (etiquetas y tamaño de muestra).

El proceso de normalización del paso 8 del protocolo asegura que las muestras segmentadas sean homogéneas y comparables entre sí, razón por la cual al final de los segmentos algunas de las señales tienen una cola de ceros que completan una longitud estándar de 187 datos y evita que las diferencias naturales en la duración de los intervalos R-R afecten la calidad del análisis, permitiendo mantener la estructura requerida para el análisis computacional sin alterar los datos originales.

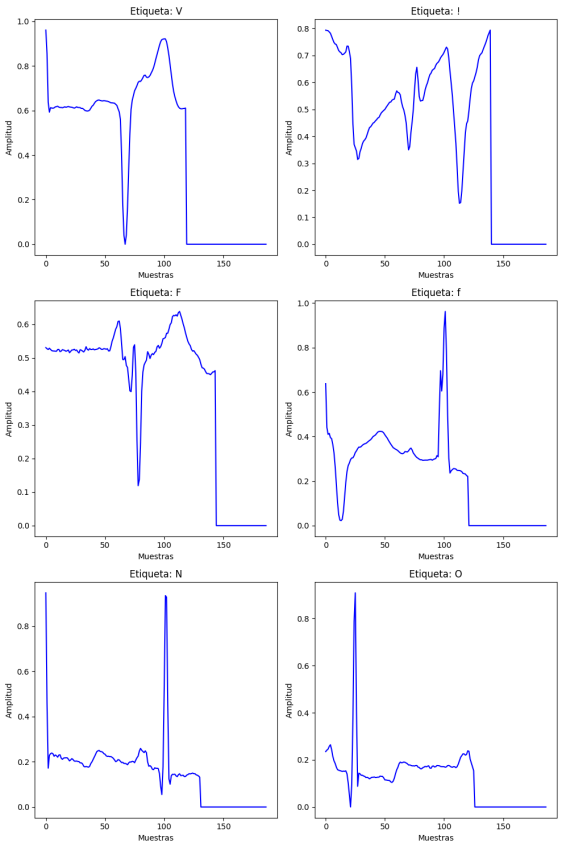


Figura 6.5: forma de la señal de cada clase

Todos los pasos mencionados anteriormente se implementan en el código del *Listing 6.5*.

```
def Segmentar(duration,fs,signal,annSample, annSymbol,prueba=0,hab=0,c
    =1.2,u=0.7): ## Paso numero 1
     segmento = duration * fs
     if prueba == 0:
         numSegments = len(signal) // segmento
     else:
         numSegments = prueba
     n = 0
     matriz = []
     for i in range(numSegments):
         start,end,segmentNew,T,rPeaks = SegmentarPorPartes(i,segmento,
             signal,fs,annSample,annSymbol,hab,u)
         segmentLength = int(c * T * fs)  # Longitud del segmento en
             muestras Revisar
         # Ajustar anotaciones para el segmento actual
         mask = (annSample >= start) & (annSample < end)
         segmentAnnSample = annSample[mask] - start  # Restar 'start'
             para alinear con el segmento
         segmentAnnSymbol = np.array(annSymbol)[mask]  # Filtrar las
             anotaciones correspondientes
         for peak in rPeaks:
             listaSegmento = SegmentarPorRPeacks(peak,segmentAnnSample,
                 segmentAnnSymbol,segmentNew,segmentLength,hab,fs,n,i)
             n += 1
             matriz.append(listaSegmento)
     print(f'Total de filas extraidas: {n}')
     return matriz
```

Listing 6.5: Definición para segmentar y retornar el dataFrame

```
def SegmentarPorPartes(parte,segmento,signal,fs,annSample,annSymbol,
    hab,u):
    start = parte * segmento
    end = start + segmento
    segmentFilter = signal[start:end]
    segmentNew = escalarVoltaje(segmentFilter)
    # Crear el eje de tiempo para el segmento actual
    t = np.arange(start, end) / fs
    rPeaks = encontrarCrucePorCero(segmentNew, fs, u)
    T = promedioRR(rPeaks,fs)
    if hab != 0:
        print(f'Para el segmento {parte} el promedio Intervalo R-R: {T
            } s')
        print(f'tama o del vector de 10 segundo es: {len(segmentNew)}
             muestras')
        printPlot(segmentNew,t,annSample,annSymbol,fs,start,end,parte,
            rPeaks)
    return start,end,segmentNew,T,rPeaks
```

Listing 6.6: Definición para segmentar por intervalos de 10s

```
def SegmentarPorRPeacks(peak,segmentAnnSample,segmentAnnSymbol,
    segmentNew,segmentLength,hab,fs,n,partes):
    listaSegmento = []
    startT = peak
    endT = startT + segmentLength
    rP = 0
    # Asegurarse de que endT no exceda el l mite de la se al
    if endT > len(segmentNew):
        endT = len(segmentNew)
        startT = endT - segmentLength
        rP = peak - startT
    # Ajustar el rango de  ndice  para el segmento
    if fs == 125:
        sampleSegment = 187
    else:
        sampleSegment = 538
    segmentoT = np.pad(segmentNew[startT:endT], (0, max(0,
        sampleSegment   - len(segmentNew[startT:endT]))), mode='
        constant')
    if len(segmentoT) >= sampleSegment:
        segmentoT = segmentoT[:sampleSegment]
    tT = np.arange(startT, startT+sampleSegment) / fs
    for sample in segmentoT:
        listaSegmento.append(sample)
    symbol = etiquetar(segmentAnnSample,startT,endT,segmentAnnSymbol,
```

```
        hab)
23      listaSegmento.append(symbol) #ETIQUETAR
24      if hab != 0:
25          print(f'tama o del vector de 1.2T segundos es: {len(segmentoT
                )} muestras')
26          print(listaSegmento)
27          print(len(listaSegmento))
28          printPlot(segmentoT,tT,segmentAnnSample,segmentAnnSymbol,fs,
                startT,endT,partes,rP,n)
29      return listaSegmento
```

Listing 6.7: Definición para segmentar por cada pico r

```
1       def etiquetar(Sample,start,end,symbol,hab):
2       priority = {'!': 1, 'F': 2, 'f': 3, 'V': 4, 'N': 5, 'O': 6}
3       SymbolList = []
4       selected_char = 'O'
5       selected_priority = priority['O']
6       for j in range(len(Sample)):
7           if start <= Sample[j] < end:
8               SymbolList.append(symbol[j])
9
10      for char in SymbolList:
11          if char in priority and priority[char] < selected_priority:
12              selected_char = char
13              selected_priority = priority[char]
14      if hab != 0:
15          print(selected_char)
16      return selected_char
```

Listing 6.8: Definición para acondicionar las etiquetas

En el *Listing 6.5*, se presenta la definición de la función Segmentar, la cual posee las siguientes entradas: la duración de los segmentos principales (en el caso del protocolo de 10 segundos), la frecuencia de muestreo, el vector de datos, las etiquetas y prueba, que hace referencia a cuántos segmentos de 10 segundos se desean obtener. Si el valor de prueba es 0, la función segmentará todo el largo del vector. Además, hab indica si se desea imprimir los segmentos en gráficas, c corresponde al coeficiente del punto 7 del protocolo (para generar segmentos independientes en función de los picos R) y u define el umbral necesario para detectar los picos R, según lo indicado en el punto 5 del protocolo.

Por otro lado, en la línea 10 del código del *Listing 6.5*, se encuentra la definición de la función *segmentarPorPartes*, cuyo propósito es devolver un segmento de 10 segundos junto con el período de intervalo R-R. El cuerpo de esta función está desarrollado en el *Listing 6.6*.

Al regresar al código del *Listing6.5*, en la línea 16, inicia el proceso correspondiente al punto 7 del protocolo, donde, partiendo de cada pico R, se obtiene un segmento independiente. Para ello, se llama a la función *SegmentarPorRPeaks*, cuya definición también se encuentra en el *Listing6.7*.

Finalmente, en la línea 22 del *Listing 6.7*, se define la función etiquetar, cuyo objetivo es realizar el filtrado adecuado de las etiquetas de interés.

6.2.2. Consultar y seleccionar técnicas de caracterización relevantes de las señales ECG antes del entrenamiento de los modelos.

Para abordar el preprocesamiento de las señales ECG, se implementaron diferentes técnicas de caracterización que permiten transformar los datos en formatos adecuados para alimentar modelos de aprendizaje automático. Estas caracterizaciones se realizan tanto en 1D como en 2D, y cada una tiene sus características y objetivos específicos. A continuación, se detallan las técnicas utilizadas:

- **Caracterización en 1D: Phase 1D**
  Consiste en la representación de las señales ECG en su forma original de unidimensional, pero con un formato ordenado que facilita el análisis. Conserva la estructura temporal de las señales, es ideal para modelos como CNN-1D y LSTM que procesan secuencias temporales directamente.
- **Caracterización en 2D: Transformada de Fase (Phase 2D)**
  La señal se reconfigura para ser representada como una matriz bidimensional, generando imágenes de tamaño (nxn). Este formato se obtiene de una transformación en 2D que captura relaciones espaciales en las características temporales de la señal. Esta caracterización permite usar arquitecturas avanzadas de CNN diseñadas para imágenes (como VGG16 o MobileNetV2)

  Caracterización en 2D: Transformada Wavelet Continua (CWT)
  La CWT convierte la señal en un escalograma, una representación en el dominio tiempo-frecuencia que resalta patrones específicos de la señal ECG. Este tipo de caracterización captura información frecuencial y temporal, lo que mejora la clasificación de arritmias complejas.

En el desarrollo de este proyecto se trabajaron dos tipos de caracterización de la señal. La primera es el diagrama de fase, el cual facilita la visualización de cómo evoluciona la señal a lo largo del tiempo, permitiendo identificar patrones y características importantes de su dinámica. La segunda caracterización corresponde a la Transformada Wavelet Continua, la cual proporciona una representación tiempo-frecuencia de la señal.

Para implementar estas caracterizaciones en Python, se pueden definir funciones que reciban un vector de entrada y devuelvan una salida caracterizada, la cual servirá como entrada para los modelos de entrenamiento. A continuación, se mostrarán las implementaciones correspondientes a cada tipo de caracterización.

Es importante destacar que, en el caso de las Wavelets, existen familias disponibles en la librería de Python llamada PyWavelets [50], entre las cuales se encuentran *cmor*, *cgau* y *gaus*, entre otras. Al consultar la documentación oficial, se pueden observar todas las variantes de estas familias. Al seleccionar una Wavelet madre, es posible visualizar su forma y determinar si esta es similar a la señal de entrada, lo cual resulta fundamental para una caracterización efectiva.

```python
def create_phase_diagram(vector):
    # Crear vectores desplazados
    phase_diagram = np.vstack((vector[:-1], vector[1:])).T
    return phase_diagram
```

Listing 6.9: Definición para caracterizar un vector en fase 1D

```python
def createGaus4(vector):
    wavelet = 'gaus4'
    coefficients, _ = pywt.cwt(vector, np.arange(1, 188), wavelet)
    return coefficients

def createPhase2D(vector):
    length = len(vector)
    phase_diagram_2d = np.zeros((length, length))
    for i in range(length):
        if i < length:
            phase_diagram_2d[i, :length - i] = vector[i:]
    return phase_diagram_2d

def createCmor(vector):
    wavelet = 'cmor0.5-1.0'
    coefficients, _ = pywt.cwt(vector, np.arange(1, 188), wavelet)
    return coefficients
```

Listing 6.10: Definiciones para caracterizar un vector en Wavelet y fase 2D

En los códigos anteriores, se puede observar en el *Listing 6.9* la función que permite devolver un vector en 1D con dimensiones (None, 186, 2). Estas dimensiones representan lo siguiente: el valor None al inicio indica la posición de la fila del vector en una matriz (correspondiendo a la cantidad de muestras), 186 es el tamaño del vector en número de muestras, y 2 representa dos vectores simultáneos. Es importante destacar que, al realizar predicciones individuales, la dimensión debe ajustarse a (1, 186, 2). En la Figura 6.6 se puede apreciar la caracterización de un segmento de la señal ECG.

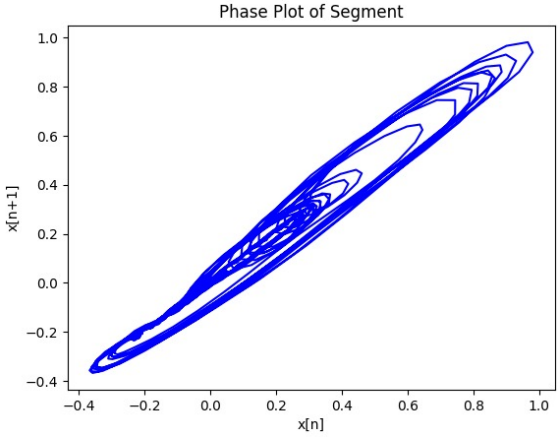


Figura 6.6: Caracterización 1D-Phase

Por otro lado, para el caso 2D, es necesario transformar el vector en una imagen. Las siguientes combinaciones de caracterización se utilizaron:

- Gaus4 con un rango de scales de 1 a 187.
- cmor0.5-1.0 con un rango de scales de 1 a 187.

En las Figuras 6.7 y 6.8 se visualizan las wavelet seleccionadas para las caracterizaciones propuestas, estas fueron seleccionadas por términos de semejanzas con el complejo QRS de la señal ECG, y por cumplir criterios determinantes en caracterización de señales biomédicas con caracteristicas como las del ECG. Y en las Figuras 6.9 y 6.10 se pueden apreciar los escalogramas que corresponden a los segmentos de señal con las funciones respectivas.

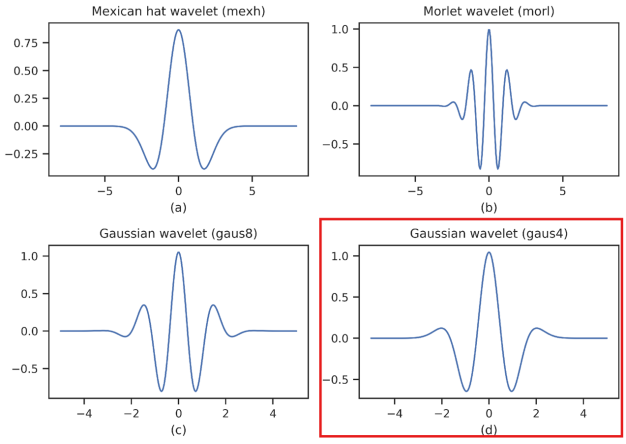


Figura 6.7: Wavelet Gaus4

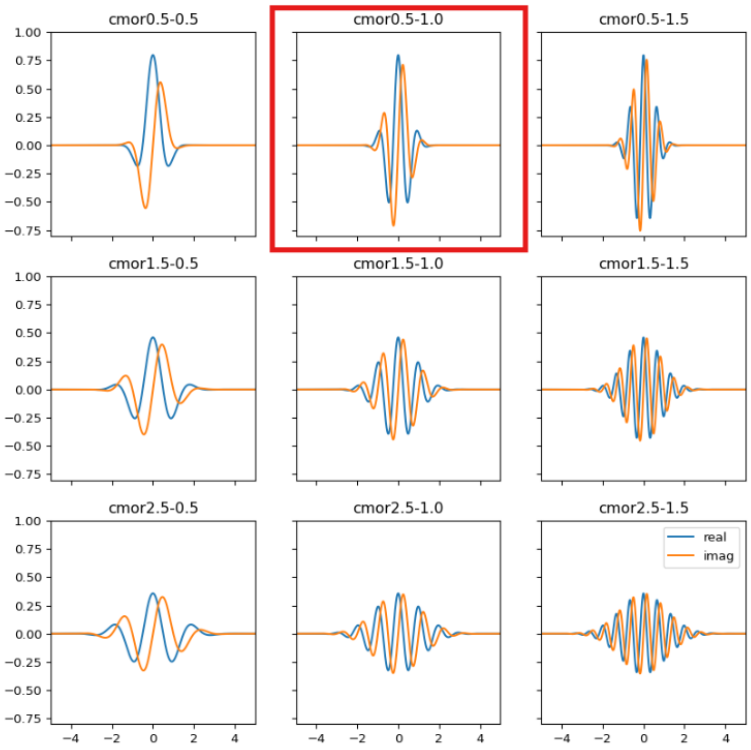


Figura 6.8: Wavelet Cmor0.5-1

Figura 6.9: Caracterización 2D-Gaus4

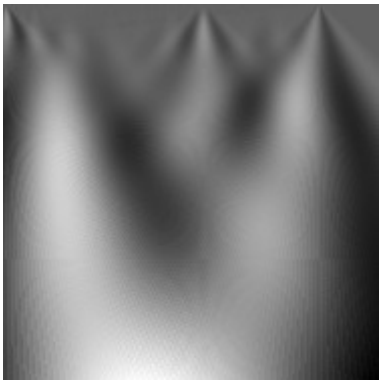


Figura 6.10: Caracterización 2D-Cmor

Estas transformaciones generan una salida ordenada con las dimensiones (None, 187, 187, 1), donde:

- None representa la posición de la fila del vector en la matriz.
- 187x187 corresponde al tamaño de la imagen en escala de grises.
- 1 indica que la imagen tiene un solo canal de color.

Finalmente, la última caracterización corresponde a la imagen 2D del diagrama de fase (PHASE), que produce una salida con las mismas dimensiones (None, 187, 187, 1), manteniendo la estructura mencionada anteriormente para la caracterización 1D. En la Figura 6.11 se puede apreciar la caracterización de un segmento de la señal ECG.

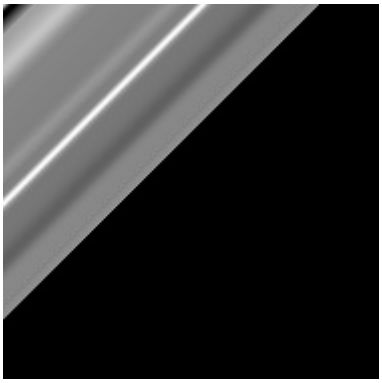


Figura 6.11: Caracterización 2D-PHASE

De esta forma da como resultado las siguientes caracterizaciones:

- Phase 1D (none, 186, 2).

- Phase 2D (none, 187, 187, 1).
- Gaus4 2D (none, 187, 187, 1).
- Cmor0.5-1.0 (none, 187, 187, 1)

6.2.3. Implementar un código en Python que integre el procesamiento de señales, la extracción de características y la preparación de los datos en conjuntos de entrenamiento y prueba, asegurando la reproducibilidad y escalabilidad del proceso.

```
def dataPhase(pathTrain,pathVal):
    train_df = pd.read_csv(pathTrain)
    val_df = pd.read_csv(pathVal)
    # Separar caracter sticas y etiquetas para el conjunto de
       entrenamiento
    X_train = train_df.iloc[:, :-1].values
    y_train = train_df.iloc[:, -1].values
    # Separar caracter sticas y etiquetas para el conjunto de
       validaci n
    X_val = val_df.iloc[:, :-1].values
    y_val = val_df.iloc[:, -1].values
    X_train_phase = np.array([create_phase_diagram(x) for x in X_train
       ], dtype=np.float32)
    X_val_phase = np.array([create_phase_diagram(x) for x in X_val],
       dtype=np.float32)
    y_train = np.array(y_train, dtype=np.int32)
    y_val = np.array(y_val, dtype=np.int32)

    return X_train_phase,y_train,X_val_phase,y_val
```

Listing 6.11: Definicion para procesar la data y separarla en entrenamiento y prueba en 1D

```
def datagaus(X_train, X_val):
    xTrains = np.array([createGaus4(x) for x in X_train], dtype=np.
       float32)
    xVal = np.array([createGaus4(x) for x in X_val], dtype=np.float32)
    xTrains1 = np.expand_dims(xTrains, axis=-1)
    xVal1 = np.expand_dims(xVal, axis=-1)
    return xTrains1,xVal1

def datacmor(X_train, X_val):
    xTrains = np.array([createCmor(x) for x in X_train], dtype=np.
       float32)
    xVal = np.array([createCmor(x) for x in X_val], dtype=np.float32)
    xTrains1 = np.expand_dims(xTrains, axis=-1)
    xVal1 = np.expand_dims(xVal, axis=-1)
```

```
13      return xTrains1,xVal1
14
15  def dataphase2D(X_train, X_val):
16      xTrains = np.array([createPhase2D(x) for x in X_train], dtype=np.
            float32)
17      xVal = np.array([createPhase2D(x) for x in X_val], dtype=np.
            float32)
18      xTrains1 = np.expand_dims(xTrains, axis=-1)
19      xVal1 = np.expand_dims(xVal, axis=-1)
20
21  def data2D(pathTrain, pathVal, tipo):
22      train_df = pd.read_csv(pathTrain)
23      val_df = pd.read_csv(pathVal)
24      X_train = train_df.iloc[:, :-1].values
25      y_train = train_df.iloc[:, -1].values
26      # Separar caracter sticas y etiquetas para el conjunto de
            validaci n
27      X_val = val_df.iloc[:, :-1].values
28      y_val = val_df.iloc[:, -1].values
29      y_train = np.array(y_train, dtype=np.int32)
30      y_val = np.array(y_val, dtype=np.int32)
31
32      if tipo == 'phase':
33          x_train, x_val = dataphase2D(X_train, X_val)
34      elif tipo == 'gaus':
35          x_train, x_val = datagaus(X_train, X_val)
36      elif tipo == 'cmor':
37          x_train, x_val = datacmor(X_train, X_val)
38      else:
39          print('Ese tipo no existe')
40
41      return x_train, y_train, x_val, y_val
```

Listing 6.12: Definiciones para procesar la data y separarla en entrenamiento y prueba en 2D

En los *Listings 6.11 y 6.12*, se presentan las definiciones de las funciones *dataPhase* y *data2D*, respectivamente. El propósito principal de estas funciones es procesar y caracterizar los datos, devolviéndolos organizados y separados en conjuntos de entrenamiento y validación.

Como observación adicional, para implementar modelos de transferencia de aprendizaje como MobileNetV2 y VGG16, cuyas dimensiones de entrada (shape) son (None, 224, 224, 3), es necesario preprocesar y acondicionar las imágenes correctamente. Esto implica ajustar el tamaño de las imágenes a 224x224 píxeles y asegurarse de que tengan 3 canales de color (RGB), cumpliendo así con los requisitos de entrada de estos modelos.

```
1   def preprocessImages(images):
2       resized_images = tf.image.resize(images, (224, 224))
```

```
    rgb_images = tf.image.grayscale_to_rgb(resized_images)
    return rgb_images
```

Listing 6.13: Definición para acondicionar imagenes para modelos de tranferencia de aprendizaje

6.3. Fase 3: Entrenamiento de algoritmos de aprendizaje automático.

Ahora bien, en cuanto al objetivo de entrenar algoritmos de aprendizaje automático que permitan la identificación de arritmias cardíacas, que corresponde a la fase 3 de este proyecto, se implementaron las siguientes actividades.

6.3.1. Consultar y seleccionar diversos algoritmos de aprendizaje automático para la clasificación de señales ECG y la identificación de arritmias cardíacas, tales como Random Forest, Support Vector Machines (SVM), Redes Neuronales Convolucionales (CNN), entre otros.

Para esta etapa se realizó una revisión de literatura de los modelos de aprendizaje profundo para la detección y clasificación de arritmias entre los años 2017 y 2023 [46]. Esto permitió conocer qué algoritmos han logrado mejor desempeño en esta tarea. Se analizaron los pros y contras de los modelos que se plantean en la revisión, teniendo en cuenta algunos criterios, y se compararon los estudios realizados a partir de ellos.

Uno de los criterios de selección para analizar los estudios más destacados en la revisión de literatura fue que los resultados del estudio hayan superado el 98 % para la métrica Accuracy, que hace referencia a la precisión entre las predicciones correctas sobre las predicciones totales. También se tuvo en cuenta aquellos que hayan ejecutado modelos con etiquetas de arritmias relacionadas con la aparición de muerte súbita cardíaca. Finalmente, se consideró el análisis de aquellos modelos que sean producto del uso de técnicas de Transfer Learning y que tengan información respecto al despliegue del modelo, esto con el fin de incursionar con mayor confianza en modelos que hayan tenido éxito en esta tarea de clasificación.

En este contexto, un estudio relevante es el de Pham et al. [51], que presenta un marco para la detección automática de arritmias en segmentos de ECG. Este estudio es importante porque propone un enfoque innovador que integra técnicas avanzadas de procesamiento de señales y aprendizaje automático, permitiendo una identificación más precisa y eficiente de patrones arrítmicos. Utilizando algoritmos de clasificación basados en redes neuronales profundas, el estudio demuestra una mejora significativa en la tasa de detección de arritmias en comparación con métodos tradicionales. Además, la inclusión de un proceso de preprocesamiento de señales, como la eliminación de ruido y la normalización de datos, optimiza la calidad de

las entradas utilizadas en el modelo, lo que contribuye a su capacidad para generalizar en diferentes escenarios clínicos.

Este estudio propone un Sistema de Soporte a la Decisión Asistido por Computadora (CDSS) que utiliza técnicas de aprendizaje automático para discriminar entre estos cuatro tipos de ritmos. Se entrenaron modelos predictivos usando segmentos de ECG de 2 y 5 segundos, con un enfoque en la extracción de características no lineales, como entropías y otras características basadas en texturas. De los métodos utilizados, el Random Forest se destacó como el clasificador con mejor desempeño, alcanzando altas tasas de precisión, sensibilidad y especificidad tanto para los segmentos de 2 segundos (precisión de 98.2 %, sensibilidad de 98.1 %, especificidad de 99.4 %) como para los de 5 segundos (precisión de 98.8 %, sensibilidad de 98.8 %, especificidad de 99.6 %). Estas métricas resaltan la eficacia del modelo, que puede integrarse en sistemas de monitoreo continuo de ECG para proporcionar una detección precisa y confiable de arritmias.

De los modelos analizados en los estudios de las (Tabla 6.5 y 10.1), los Long Short-Term Memory (LSTM) y las Redes Neuronales Convolucionales (CNN) se destacaron como las técnicas más prometedoras tanto por su rendimiento como por su uso frecuente en diferentes estudios.

Las redes LSTM son una variante de las redes neuronales recurrentes (RNN) diseñadas para procesar secuencias temporales de datos. En el contexto de la clasificación de arritmias, las LSTM son especialmente efectivas porque las señales ECG contienen patrones recurrentes y eventos secuenciales que pueden ser críticos para identificar variaciones sutiles asociadas con arritmias específicas [31]. Por ejemplo, el modelo de Zhang et al. (2017) utilizó LSTM en combinación con técnicas de agrupamiento para seleccionar datos representativos, logrando un 99.40 % de Accuracy, mientras que el modelo bidireccional de Yildirim et al. (2018) alcanzó un 99.39 % de Accuracy al incorporar una capa de Wavelet Sequences. Esto muestra que las LSTM no solo son precisas, sino que también se adaptan bien a diferentes enfoques de preprocesamiento y modelado.

Por su parte, las Redes Neuronales Convolucionales (CNN) han demostrado ser altamente efectivas para la clasificación de señales ECG, particularmente cuando se transforman las señales en representaciones bidimensionales, como espectrogramas o imágenes de frecuencia-tiempo. Su capacidad para extraer características jerárquicas de los datos las hace ideales para esta tarea, ya que pueden identificar patrones espaciales y temporales relacionados con las diferentes clases de arritmias. Por ejemplo, Jun et al. (2018) implementaron un modelo basado en imágenes de 128x128 píxeles, logrando un 99.05 % de Accuracy, mientras que Xia y Xie (2019) combinaron CNN unidimensionales con aprendizaje activo, obteniendo un 99.20 % de Accuracy en un sistema portátil. Estas arquitecturas son particularmente útiles para la clasificación de múltiples etiquetas (como 4 o 5 tipos de arritmias), ya que las técnicas de convolución permiten separar las características distintivas de cada clase, incluso en señales complejas y ruidosas.

Ambas técnicas, LSTM y CNN, han sido ampliamente validadas en la literatura y se presentan

como las opciones más confiables para abordar la clasificación de señales ECG en aplicaciones clínicas o de monitoreo en tiempo real.

Estudio	Arquitectura	Resumen	Accuracy (%)
Patient-Specific Deep Architectural Model for ECG Classification (Luo et al., 2017)	4 clases, DNN-SDA	Transformada Wavelet MFSWT. Codificador automático apilado SDA. Clasificación específica del paciente ajustando muestras individuales.	98.80
Patient-specific ECG classification based on recurrent Neural networks and clustering technique (Zhang et al., 2017)	5 clases, RNN	Transformada DTCWT + filtrado de mediana. Uso de LSTM con capas específicas y técnica de agrupamiento basado en densidad para selección de datos representativos.	99.40
Atrial Fibrillation Detection Using Stationary Wavelet Transform and Deep Learning (Xia et al., 2017)	3 clases, CNN	Transformada Wavelet Daubechies 5. Clasificación de fibrilación auricular usando imágenes de frecuencia-tiempo.	98.63
ECG arrhythmia classification using a 2-D convolutional neural network (Jun et al., 2018)	4 clases, 2D CNN	Entrada transformada en imágenes 2D de 128x128 píxeles. Clasificación robusta con técnicas como normalización por lotes y recorte de imágenes.	99.05
A novel wavelet sequences based on deep bidirectional LSTM network model for ECG signal classification (Yildirim et al., 2018)	4 clases, Bi-directional LSTM	Introducción de la capa Wavelet Sequences (WS). Modelo DBLSTM con alta precisión en clasificación sin extracción manual de características.	99.39
A deep learning approach for ECG-based heartbeat classification for arrhythmia detection (Sannino and De Pietro, 2018)	4 clases, DNN	Procesamiento de señales con detección de picos y segmentación de latidos. Red DNN con siete capas ocultas ajustada empíricamente.	99.68

Cuadro 6.5: Estudios de clasificación de arrtimias con modelos de inteligencia artificial Parte I

Estudio	Arquitectura	Resumen	Accuracy
A Novel Wearable Electrocardiogram Classification System Using Convolutional Neural Networks and Active Learning (Xia and Xie, 2019)	4 clases, 1D CNN + Active Learning	Sistema wearable con aprendizaje activo. Clasificador refinado iterativamente mediante etiquetado experto y reentrenamiento.	99.20
An Automatic Cardiac Arrhythmia Classification System With Wearable Electrocardiogram (Xia et al., 2018)	4 clases, DNN	Uso de autoencoder apilado de denoising (SDAE) y aprendizaje activo. Validación en MIT-BIH y dispositivo wearable, mostrando un rendimiento superior a métodos convencionales.	99.80
Automated arrhythmia classification based on a combination network of CNN and LSTM (Chen et al., 2020)	4 clases, CNN + LSTM	Fusión de sub-redes CNN y LSTM. Precisión del 99.32 % en MIT-BIH, con alto rendimiento en identificación de arritmias y segmentación.	99.32
Fuzz-ClustNet: Coupled fuzzy clustering and deep neural networks for Arrhythmia detection from ECG signals (Kumar et al., 2023)	5 clases, Fuzz-ClusNet	Eliminación de ruido, aumento de datos, segmentación y uso de CNN y agrupamiento difuso para la clasificación. Precisión del 98.66 % en MIT-BIH.	98.66
Automated Method for Discrimination of Arrhythmias Using Time, Frequency, and Nonlinear Features of Electrocardiogram Signals (Shirin Hajeb-Mohammadalipour et al., 2018)	No clases, Árbol de decisión y SVM	Segmentación y extracción de características no lineales y en frecuencia. Clasificación de fibrilación ventricular y no ventricular.	99.20

Cuadro 6.6: Estudios de clasificación de arrtimias con modelos de inteligencia artificial Parte II

6.3.2. Entrenar varios modelos de aprendizaje automático utilizando las características extraídas de las señales ECG, utilizando conjuntos de entrenamiento previamente preparados, y ajustar los hiperparámetros de cada modelo para mejorar su desempeño.

Al realizar el entrenamiento de los modelos, se seleccionaron las siguientes arquitecturas para cada dimensión. En el caso de 1D, se propusieron tres modelos en el presente trabajo:

- CNN conv1D + Capa Densa: con una entrada de tipo (None, 186, 2), los hiperparámetros son 50 epocas, batch size de 256, optimizador adam, loss sparse categorical crossentropy.
- CNN conv1D + LSTM + Capa Densa: con una entrada de tipo (None, 186, 2), los hiperparámetros son 50 epocas, batch size de 128, optimizador adam, loss sparse categorical crossentropy.
- CNN conv1D Triple Núcleo + LSTM + Capa Densa: con una entrada del tipo [X, X, X], donde X corresponde a (None, 186, 2), los hiperparámetros son 30 epocas, batch size de 128, optimizador adam, loss categorical crossentropy.

En el caso de los modelos en 2D, se propusieron un modelo propio y dos modelos con transferencia de aprendizaje: MobileNetV2 y VGG16. Para cada modelo, se realizó una repetición con las caracterizaciones propuestas:

- CNN Conv2D + Capa Densa: con una entrada de tipo (None, 187, 187, 1), los hiperparametros son 30 epocas, batch size de 128, optimizador adam, loss sparse categorical crossentropy.
- MobileNetV2 + Capa Densa: con una entrada de tipo (None, 187, 187, 3), los hiperparametros son 30 epocas, batch size de 128, optimizador adam, loss categorical crossentropy.
- VGG16 + Capa Densa: con una entrada de tipo (None, 187, 187, 3), los hiperparametros son 10 epocas, batchsize de 128, optimizador adam, loss categorical crossentropy.

De forma adicional, y siguiendo las recomendaciones de los orientadores del proyecto, se decidió limitar la cantidad de etiquetas utilizadas para entrenar los modelos. Es decir, las etiquetas de interés se redujeron a cuatro: *'V', '!', 'F', 'f'* (ver Tabla 6.4).

En este contexto, es necesario el uso de un modelo específico biclase, que no se desarrolla en el marco de este proyecto, para diferenciar entre ritmo normal y arritmia que se sera un primer filtro, Si la señal es clasificada como arritmia, se procede a utilizar el modelo del proyecto, el cual trabaja exclusivamente con cuatro etiquetas.

Por otro lado, para el entrenamiento de los modelos, se desarrolló un script en Python que posteriormente fue estructurado como una librería llamada *ECGSignal.py*. Además, para facilitar las pruebas paso a paso y la implementación modular, se optó por utilizar Jupyter Notebook. Todos los notebooks estarán disponibles en los anexos del proyecto [ver Anexo 1].

6.4. Fase 4: Evaluación de modelos entrenados.

Para dar solución al objetivo, evaluar los modelos entrenados usando métricas de clasificación para la selección del mejor método en la identificación de arritmias cardíacas. Se realizaron las siguientes actividades:

6.4.1. Realizar estructura de carpetas para guardar los modelos entrenados.

Para el desarrollo de la actividad, se optó por crear una estructura de carpetas organizada que permitió almacenar y categorizar de manera eficiente los resultados de los modelos. La carpeta raíz se denominó *ResultadoModelos* y contiene subcarpetas correspondientes a las categorías de los modelos. A su vez, cada categoría tiene subcarpetas adicionales con la siguiente estructura de nombre: *numeroDelModelo_ caracterización_ dimesión_ numeroDeEpocas_ Epocas_ numeroDeClases*, el contenido de la carpeta es:

- Gráfica de Accuracy y Loss: Representación visual del desempeño del modelo durante el entrenamiento y validación.
- Matriz de confusión: Análisis detallado de las predicciones del modelo.
- Reporte de métricas: Generado con la función *classification_ report* de la librería *sklearn.metrics*, que incluye las métricas de Accuracy, Precisión, Recall y F1-Score.
- Archivos del modelo: Tres archivos en distintos formatos para facilitar la carga del modelo en otros entornos:
  - Formato .h5
  - Formato .keras
  - Formato .pkl

Gracias a esta estructura de carpetas, se facilitó la organización y análisis de los resultados. Los modelos se pueden categorizar y analizar según los siguientes criterios:

- Dimensión (1D o 2D).
- Cantidad de clases (Seis clases, cinco clases, o cuatro clases).
- Nombre del modelo (propio o de transferencia de aprendizaje).

Con esta estructura, se generó una tabla de análisis de resultados que permitió una comparación clara entre los diferentes modelos y sus configuraciones.

6.4.2. Analizar en detalle las métricas de evaluación obtenidas durante el benchmarking para identificar el modelo que mejor desempeña en la tarea de identificación de arritmias cardíacas.

El benchmarking de los resultados se realizó en Excel, utilizando como criterio de selección las siguientes métricas del reporte.

- **Accuracy:** Mide la proporción de todas las predicciones correctas (tanto positivas como negativas) con respecto al total de ejemplos. Correspone a la formula $\frac{TP+TN}{TP+TN+FP+FN}$)
- **Precisión:** Mide la proporción de verdaderos positivos entre todas las predicciones positivas. Correspone a la formula $\frac{TP}{TP+FP}$)
- **Recall:** Mide la proporción de verdaderos positivos con respecto al total de positivos reales (incluso aquellos que fueron clasificados incorrectamente como negativos). Correspone a la formula $\frac{TP}{TP+FN}$)
- **F1-Score:** Es la media armónica entre precisión y recall. Es útil cuando buscas un balance entre ambas métricas, especialmente si las clases están desbalanceadas. Correspone a la formula $2\frac{PrecisionRecall}{Precision+Recall}$)

Para entender estas métricas, es importante conocer que la matriz de confusión compara las etiquetas verdaderas (labels reales) con las predicciones (labels del modelo). Esta comparación genera las siguientes categorías:

- TN (True Negative): Ejemplos correctamente clasificados como negativos.
- FN (False Negative): Ejemplos incorrectamente clasificados como negativos.
- TP (True Positive): Ejemplos correctamente clasificados como positivos.
- FP (False Positive): Ejemplos incorrectamente clasificados como positivos (también llamados "falsas alarmas").

La estructura para la tabla de comparación se organizó de la siguiente manera:

Se separó la tabla en dos categorías principales: 1D y 2D. Dentro de cada categoría, se incluyó un listado de modelos con su respectivo nombre, acompañado de sus métricas correspondientes.

Las columnas de la tabla contienen los resultados de las métricas de cada modelo, como Accuracy, Precisión, Recall Y F1-Score. En primera instancia, se trabajó con los modelos de seis clases, ya que esta fue la configuración inicial. Sin embargo, como se mencionó en la fase 2, también se trabajó con modelos de cinco y cuatro clases, específicamente para modelos como los de 1D y VGG16.

Para el análisis de los resultados del desempeño de los modelos, se consideraron dos factores principales: el comportamiento del modelo en términos de las métricas obtenidas y el costo computacional asociado a cada modelo, tanto para el entrenamiento como para la predicción. Se partió del supuesto de que los modelos en 2D tienen un costo computacional inicial de $O(n^2)$, mientras que los modelos en 1D presentan un costo computacional de $o(n)$.

6.4.3. Seleccionar el mejor modelo basado en el benchmarking realizado, considerando su capacidad para generalizar y su desempeño en validación.

En primera instancia, se utilizó la tabla de comparación con las seis clases iniciales. A partir de ahí, se realizó un primer filtro, seleccionando los mejores candidatos de cada dimensión. Estos candidatos avanzaron a la segunda instancia, en la que cada modelo fue entrenado con cinco y cuatro clases para observar el comportamiento de las métricas. Este proceso permitió evaluar cómo las métricas variaron según la cantidad de clases.

El siguiente criterio de selección fue la complejidad computacional, ya que el sistema está diseñado para monitorear señales ECG de forma continua. El tiempo de procesamiento es crucial, pues debe ser eficiente para realizar predicciones en tiempo real.

Finalmente, se seleccionó el modelo con mejor desempeño, y se procedió a la actividad de repeticiones para validar la estabilidad del modelo. Esto se realizó mediante el cálculo de métricas como el promedio y la desviación estándar.

6.4.4. Realizar repeticiones del modelo seleccionado para comprobar estabilidad.

Para esta actividad, el modelo el mejor desempeño fue entrenado y validado con la base de datos, pero organizada de forma aleatoria en los conjuntos de entrenamiento y validación y eso se logró con la función train_test_split de la libreria sklearn.model_selection, ya que posee algo llamado radom_state el cual si es None será completamente aleatoria la distribución de la data cada vez que se ejecute y con la librería del proyecto *ECGSignal.py* logramos esto de forma automática por medio de la definición *guardarDataSegmentadaInTrainVal* tal como se aprecia en *Listing 6.14*

```
folder_path = "ArchivosCSV/Full-125Hz.csv"
full_df_125 = pd.read_csv(folder_path)
guardarDataSegmentadaInTrainVal(full_df_125,2000,labelToNumber,0.2,"
    2000_80_20_Seis_Clases_Cinco")
# En donde la funcion tiene la siguiente entrada
# df = df completo de vectores segmentados
# maxSignal = el maximos de vectores a trabajar
#labels = la bases de datos vienen con labels en letras, se debe
    tranformar a numeros
#testVal = el porcentaje de validaci n en la separacion 0.2
#path = ruta en donde guardar el df en formato csv
#random = None es el defecto si se agrega un numero funcionara como
    seed
```

Listing 6.14: Definición para procesar la data de forma aleatoria y guardarla.

Se realizaron cinco repeticiones del modelo con mejor desempeño para calcular el promedio y la desviación estándar de los resultados, con el fin de evaluar su estabilidad. En caso de obtener un resultado negativo, se iteraría nuevamente el proceso de selección del modelo y sus repeticiones para asegurar una correcta validación.

Al obtener un modelo estable, este estaba listo para ser validado con un conjunto de datos independientes y no conocidos por el modelo para obtener resultado de comportamiento en un entorno parecido al real.

6.4.5. Validar el modelo seleccionado utilizando un conjunto de datos de prueba independiente, para confirmar su desempeño y garantizar su aplicabilidad en entornos no controlados.

Para esta actividad, se consultó en physionet, bases de datos de ECG relacionadas con las etiquetas de arritmias de interés en este proyecto, para validar el modelo seleccionado con conjunto de datos no conocidos, en donde se puede garantizar una aplicabilidad en entornos no controlados, del cual se seleccionó un conjunto de testeo:

- Brno University of Technology ECG Signal Database with Annotations of P Wave (BUT PDB) [52]: es un recurso creado para abordar la necesidad de un conjunto de datos de ECG con anotaciones precisas y manuales de ondas P, especialmente en registros que presentan patologías cardíacas. A continuación, se ofrece una descripción detallada:

  La base de datos incluye 50 señales ECG de dos minutos y dos derivaciones, con un total de 23 tipos de patologías. Las posiciones de las ondas P fueron anotadas manualmente por dos expertos en ECG con siete años de experiencia en la evaluación de registros Holter en un contexto clínico cardiovascular.

  Los datos de esta base fueron seleccionados por los expertos a partir de tres bases de datos existentes: MIT-BIH Arrhythmia Database (MIT-A), MIT-BIH Supraventricular Arrhythmia Database (MIT-S) y Long Term AF Database (LT-AF). Los expertos eligieron segmentos de dos minutos que presentaran una mayor incidencia de patologías y desafíos en la detección automática de ondas P. Se seleccionaron un total de 38 señales de MIT-A, 5 señales de MIT-S y 7 señales de LT-AF, representando un conjunto diverso de patologías observadas en la práctica médica.

  La base de datos contiene 7,638 complejos QRS, de los cuales 2,120 no tienen ondas P, como en casos de fibrilación auricular, latidos ventriculares o ritmo nodal, y 81 ondas P no están asociadas a complejos QRS, como ocurre en bloqueos auriculoventriculares de segundo grado. En total, la base incluye 5,599 ondas P. Los expertos anotaron las posiciones de las ondas P manualmente y, en casos de dudas, discutieron hasta llegar

a un consenso. Para facilitar el trabajo, se utilizó el software SignalPlant, aunque las anotaciones se realizaron de manera completamente manual.

Cada señal en la base de datos contiene información sobre el diagnóstico dominante y los tipos de complejos QRS presentes, información que fue validada o complementada por los expertos. La base de datos está diseñada para representar un conjunto realista de datos médicos y facilitar el desarrollo de algoritmos más precisos y robustos para el análisis de señales ECG y la detección de ondas P.

Se aplicó el proceso de segmentación y etiquetado en el conjunto de testeo, filtrando únicamente las clases de interés: *['V', '!', 'F', 'f']*. Con el DataFrame organizado, se realizaron las predicciones y se asignó cada resultado a su vector correspondiente. Esto permitió utilizar las funciones de la librería Pandas de Python para analizar los resultados con las mismas métricas empleadas en la *actividad 6.4.2*. Finalmente, se elaboró un cuadro comparativo que muestra el comportamiento de cada base de datos y las métricas obtenidas con dicha data independiente.

6.5. Fase 5: Implementación de un aplicativo móvil.

Para dar solución al último objetivo específico de este proyecto, se implementará un aplicativo móvil que recepcione los datos ECG provenientes de un dispositivo Holter (ECG) y genere alertas de arritmias cardíacas que estén relacionadas con la posibilidad de MSC.

Dentro de los requerimientos del aplicativo móvil se encuentran los siguientes:

6.5.1. Requerimientos del Sistema para el Aplicativo Móvil

El desarrollo del aplicativo móvil abarcará el ciclo completo: diseño, implementación, permisos, base de datos, API en Python con Flask y el desarrollo del frontend con el framework Flutter. A continuación se presentan los requerimientos funcionales y no funcionales:

6.5.1.1. Requerimientos funcionales

- **Diseño UI/UX:** Implementar técnicas de diseño basadas en *UI/UX* para lograr una interfaz intuitiva, cómoda y fácil de usar, asegurando accesibilidad para todo tipo de usuarios.
- **Módulo de autenticación:**
  - **Login:** Validación de credenciales de usuario para garantizar acceso seguro.
  - **Registro:** Funcionalidad de creación de nuevos usuarios mediante un formulario.
  - **Validador de identidad:** Implementar validación de identidad utilizando credenciales legales o integraciones con servicios externos (por ejemplo, verificación mediante email o código OTP).

- **Conectividad:**
  - Conexión con dispositivos externos a través de *Bluetooth* o *WiFi* para la recepción de datos ECG desde el dispositivo Holter.
  - **Sincronización de datos:** Implementar una conexión a un servicio REST API (desarrollado en Flask) para enviar, recibir y almacenar datos en tiempo real.
- **Monitoreo de ubicación:**
  - Registro de la ubicación en tiempo real del usuario.
  - Actualización automática cada 100 metros.
- **Notificaciones de alertas:**
  - Generar notificaciones automáticas en caso de detección de posible cuadro de muerte subita cardíaca.
  - Métodos de notificación: WhatsApp, llamada telefónica, SMS o Telegram.
- **Integración del modelo de IA:**
  - Integrar el modelo de inteligencia artificial entrenado para predecir arritmias cardíacas a partir de las señales ECG.
  - Procesamiento de datos en tiempo real y generación de resultados.
- **Gestión de datos:**
  - Desarrollo de una base de datos para almacenar:
    - Datos del usuario (credenciales y perfil).
    - Historial de señales ECG recibidas.
    - Historial de alertas generadas.
    - Historial de ritmos cardíacos.
  - Integración de la base de datos con la REST API para mantener datos actualizados y centralizados.
- **Permisos del sistema:**
  - Solicitar permisos de ubicación, Bluetooth, WiFi, notificaciones y acceso a llamadas/SMS.
  - Implementar políticas de privacidad y manejo seguro de datos.
- **Perfil del usuario:**
  - Mostrar información del usuario y posibilidad de edición.
  - Opciones de configuración para notificaciones y preferencias.

6.5.1.2. Requerimientos no funcionales

- **Rendimiento:**
  - La aplicación debe procesar señales ECG en tiempo real sin retrasos significativos.

  - Actualización de ubicación en menos de 5 segundos tras recorrer 100 metros.
- **Escalabilidad:** La aplicación debe soportar un aumento en la cantidad de usuarios y datos sin comprometer el rendimiento.
- **Seguridad:**
  - Implementar cifrado de datos (por ejemplo, SSL/TLS) para proteger la comunicación entre el cliente y el servidor.
  - Cifrado de las credenciales de usuario en la base de datos.
- **Usabilidad:** La aplicación debe ser fácil de navegar, con botones y menús intuitivos y accesibles.
- **Compatibilidad:**
  - Desarrollar la aplicación para dispositivos Android e iOS utilizando Flutter.
  - Soporte para versiones recientes de sistemas operativos (Android 8.0+ e iOS 13+).
- **Mantenimiento:** El código debe ser limpio, modular y documentado para facilitar futuras actualizaciones.
- **Pruebas:**
  - Realizar pruebas funcionales, de integración y de rendimiento.
  - Pruebas en un entorno alfa con datos simulados para validación interna.
- **Despliegue:**
  - Desplegar la REST API y base de datos en una plataforma de nube (AWS, Azure o VPS).
  - Habilitar versiones alfa de la aplicación para pruebas controladas.

6.5.2. Actividades planificadas del desarrollo móvil

El desarrollo del proyecto se enfocará en cumplir los requerimientos funcionales identificados. A continuación se presenta un listado de actividades planificadas:

- **Nombrar la aplicación:** Se debe seleccionar un nombre que le de identidad a la aplicación por medio de signos y significados.
- **Realizar mockups a lápiz:** Se crearán bocetos a mano de todas las pantallas de visualización, iteraciones, botones, posibles combinaciones de colores y fuentes de letras. Esta etapa será la más creativa y permitirá diseñar una interfaz de usuario basada en principios de *UI/UX*.
- **Crear mockups digitales:** Se utilizará la herramienta Figma, seleccionada por sus capacidades *no-code* y por ser una plataforma online que facilita el desarrollo de prototipos interactivos. Los diseños serán más fieles a la visualización real de la aplicación.

- **Validar el prototipo con profesionales orientadores:** Los mockups serán revisados y validados por los directores del proyecto. Se confirmará que todas las pantallas necesarias estén incluidas y cumplan con las funcionalidades requeridas.
- **Seleccionar el lenguaje de programación y framework:** Se elegirá la combinación de lenguaje y herramientas que mejor se ajuste al desarrollo móvil, garantizando eficiencia, escalabilidad y compatibilidad con los dispositivos.
- **Desarrollar y validar funcionalidades:** Se procederá a desarrollar cada funcionalidad de la aplicación, validando continuamente su correcto desempeño conforme a los requerimientos definidos.
- **Desarrollar REST API:** Se implementará una API que permita el consumo y envío dinámico de datos generados por la aplicación, facilitando la comunicación entre el servidor y la aplicación móvil.
- **Desarrollar base de datos:** Se diseñará una base de datos para almacenar la información requerida durante las pruebas y validaciones del programa.
- **Desplegar en la nube:** El servicio será desplegado en una plataforma de nube como AWS o un servidor privado virtual (VPS), asegurando disponibilidad y acceso para las pruebas.
- **Realizar pruebas internas:** Se realizarán pruebas en un entorno controlado con datos simulados, validando cada requerimiento funcional de la aplicación.

Todas las pruebas se realizarán de manera controlada con datos simulados. De este modo, el desarrollo estará en la fase de pruebas, sin llegar aún a la etapa de lanzamiento. Esta versión será categorizada como una **versión alfa**, en la cual sólo el equipo del proyecto y los evaluadores seleccionados tendrán acceso a la aplicación. El objetivo principal será verificar la funcionalidad de la aplicación y recopilar retroalimentación para realizar los ajustes necesarios.

6.5.3. Nombrar la aplicación.

Para nombrar el aplicativo se tienen en cuenta tres criterios principales que aseguran la alineación del nombre con la misión y los objetivos del proyecto:

1. **Significado alineado con el propósito:** Se prioriza un nombre que refleje la esencia del proyecto que represente la misión de la aplicación de prevenir riesgos cardiovasculares mediante la detección temprana de arritmias y la activación de rutas de atención oportuna.
2. **Impacto y posicionamiento:** Se busca un nombre breve, memorable y universal, que facilite su reconocimiento y aceptación tanto a nivel local como internacional.
3. **Conexión emocional y valores:** El nombre debe transmitir confianza, innovación y seguridad, conceptos fundamentales en el desarrollo de soluciones tecnológicas para la salud.

Con base en estos criterios, se selecciona el nombre KIBO, que encapsula la misión del proyecto y refuerza su propósito de brindar esperanza y mejorar la calidad de vida de los usuarios.

6.5.4. Realizar mockups a lápiz.

Para esta actividad se utilizaron dos técnicas de diseño que permiten conceptualizar de manera creativa y efectiva la estructura y apariencia de la aplicación:

- **Moodboard UI:** Esta es una técnica visual que se utiliza al inicio del proceso de diseño de interfaces de usuario. Su principal objetivo es exponer los conceptos, emociones y la percepción que se espera que transmita la aplicación. El *moodboard* puede realizarse de forma digital, reuniendo elementos como colores, tipografías, iconos e imágenes inspiracionales que ayudan a definir el estilo visual.
- **UX storyboard:** Una vez definido el concepto visual, se procede a realizar la técnica de *storyboard UX*, enfocada en la experiencia del usuario. Esta técnica permite visualizar de manera secuencial las pantallas, widgets, animaciones y transiciones entre ellas (*go-to screens*). El *storyboard* se realiza a mano para potenciar la creatividad y la exploración de ideas sin restricciones. Se utiliza una plantilla con la forma y escala de un dispositivo móvil para representar con precisión las proporciones de la interfaz.

Estas técnicas permiten establecer una base sólida para el desarrollo de la aplicación, asegurando que tanto el diseño visual (**UI**) como la experiencia del usuario (**UX**) sean intuitivos y atractivos.

6.5.5. Crear mockups digitales

Una vez completados los *mockups* a mano, se procede a digitalizarlos en formato PDF como referencia inicial. A partir de esto, se inicia el proceso de creación del *mockup digital* utilizando la herramienta Figma, la cual permite desarrollar un diseño de alta fidelidad con precisión y funcionalidad.

Para lograr un **UI** responsivo y adaptado a todos los tamaños de pantallas, se aplican buenas prácticas de diseño, tales como:

- Uso de **componentes** reutilizables que permiten mantener la consistencia en la interfaz.
- Implementación de **variantes** para facilitar la personalización y adaptación de los elementos visuales.
- Inclusión de **animaciones** y transiciones suaves entre pantallas.
- Etiquetado correcto y estructurado de todos los elementos, lo que facilita el orden y la colaboración dentro del proyecto.

En cuanto al **UX**, Figma permite integrar funcionalidades como:

- Creación de botones interactivos y enlaces (*go-to screens*) que simulan el flujo de navegación entre las pantallas.
- Definición de condiciones y estados interactivos, que permiten evaluar la experiencia del usuario.

6.5.6. Validar el prototipo con los directores del proyecto

Se realiza la validación de diseño con directores del proyecto para presentar los *mockups* de alta fidelidad desarrollados en Figma. Durante la sesión, se expone cada pantalla del prototipo, destacando las funcionalidades implementadas, la experiencia del usuario (UX) y los componentes visuales del diseño de la interfaz (UI). Se recibe retroalimentación y, con identificación de posibles ajustes o mejoras en el diseño se procede a realizar las modificaciones pertinentes en el prototipo, asegurando que cumpla con los siguientes criterios:

- **Usabilidad e Intuición:** Verificación de que la navegación sea sencilla y las pantallas sean intuitivas para el usuario final.
- **Consistencia Visual:** Garantizar uniformidad en colores, tipografías y componentes visuales.
- **Funcionalidad:** Asegurar que los botones, enlaces y flujos de interacción estén correctamente definidos.

Una vez finalizados los ajustes, se considera completado el diseño del prototipo.

6.5.7. Seleccionar el lenguaje de programación y framework.

Existen diversas alternativas para programar una aplicación móvil. El desarrollo nativo utiliza lenguajes como **Kotlin** para Android o **Swift/Objective-C** para iOS. Por otro lado, los frameworks multiplataforma como **Flutter** (Dart), **React Native** (JavaScript/TypeScript) e **Ionic** (HTML, CSS, JavaScript) permiten desarrollar aplicaciones con una sola base de código para ambos sistemas operativos.

Para esta actividad, se selecciona **Flutter** como el framework de desarrollo principal debido a las siguientes ventajas:

- **Desarrollo multiplataforma:** Flutter permite desarrollar aplicaciones para Android, iOS, Web y Escritorio utilizando un solo código base, lo que reduce significativamente el tiempo y los costos de desarrollo.
- **Rendimiento cercano al nativo:** Flutter compila el código a *nativo* utilizando su motor de renderizado propio llamado *Skia*, lo que garantiza un rendimiento óptimo similar al desarrollo nativo.

- **Fácil integración con Python:** Flutter puede integrarse fácilmente con aplicaciones o APIs desarrolladas en **Python**, permitiendo una comunicación eficiente con el backend mediante servicios REST-API, facilitando la implementación de modelos de inteligencia artificial (IA).
- **Productividad del desarrollador:** La función de *Hot Reload* de Flutter permite visualizar los cambios en tiempo real, acelerando el proceso de desarrollo y depuración.
- **Uso de GetX:** La integración de la librería **GetX** facilita la implementación de una arquitectura ordenada y la navegación entre pantallas. Además, GetX permite la gestión eficiente del estado, rutas y dependencias sin código redundante.
- **Interfaz de usuario (UI) personalizable:** Flutter permite crear interfaces atractivas y personalizadas con un alto grado de control gracias a sus componentes basados en *widgets*, lo que asegura una experiencia de usuario (UX) fluida y adaptable a cualquier tamaño de pantalla ademas en Figma se diseño con base a componentes que son widgets.
- **Soporte para animaciones y gráficos:** Flutter facilita la creación de animaciones complejas y gráficos interactivos, ideales para visualizar datos en tiempo real, como los registros ECG.
- **Reducción de costos y tiempo:** Al ser una solución multiplataforma y de alto rendimiento, Flutter permite reducir costos y tiempos de desarrollo en comparación con el desarrollo nativo.

6.5.8. Desarrollar y validar funcionalidades

Se implementarán todos los requerimientos funcionales, enfocando el objetivo principal de la aplicación: **monitorear, identificar y notificar** posibles episodios de muerte súbita cardiaca. La validación de funcionalidades se realizará mediante el simulador móvil, probando comportamientos como permisos, ubicación, predicciones y gráficas. De manera paralela, se desarrollarán el servicio REST API y la base de datos.

6.5.9. Desarrollar REST API

Se utilizará **Python** con la librería **Flask** para crear un servicio REST API que permita realizar operaciones *GET* y *POST*. Esta API servirá de puente entre el frontend y la base de datos, permitiendo consumir, enviar y organizar la información de manera dinámica. Una vez completado, el servicio se desplegará en la nube para garantizar comunicación constante con la aplicación móvil.

6.5.10. Desarrollar base de datos

Se utilizará **MongoDB Atlas** por su almacenamiento gratuito de hasta 500MB, suficiente para pruebas piloto. La base de datos, llamada *dbkibo*, contendrá las siguientes colecciones:

- **credential:** _id, email, cel_number, pass.
- **users, doctor_user, emergency_user, info_user, medical_info:** Para consultas y actualización de datos.
- **ecg_data, predictions, heart_rate:** Almacenan registros con la estructura: *id_user* y un vector *info* (hora, valor).

La información se recolectará en la aplicación cada 10 minutos y se enviará como instancia a la base de datos para optimizar el tráfico. Aún no se implementa almacenamiento local ni sincronización offline.

6.5.11. Desplegar en la nube

El despliegue de la aplicación se realizará en un VPS proporcionado por **Hostinger**, con las siguientes características:

- 2 núcleos de vCPU
- 8 GB de RAM
- 100 GB de espacio en disco NVMe
- 8 TB de ancho de banda
- Centros de datos distribuidos globalmente
- Sistema operativo Linux

Se utilizará un **cPanel** gratuito para la administración del servicio, y se configurará un dominio con **certificado SSL** también provisto por Hostinger, garantizando la seguridad en las comunicaciones.

6.5.12. Realizar pruebas internas y validar el aplicativo móvil utilizando simuladores de ECG como el PROSIM 4 de Fluke

Las arritmias, especialmente aquellas vinculadas con la ocurrencia de Muerte Súbita Cardíaca (MSC), están asociadas con latidos anormales clasificados según la AAMI, como:

- Contracción Ventricular Prematura
- Onda de Aleteo Ventricular
- Fusión Ventricular
- Fusión Rítmica

Estas etiquetas fueron utilizadas para entrenar el modelo de IA. Para validar el rendimiento del modelo, se utilizarán instrumentos de laboratorio y simulación, específicamente el **ProSim TM 4** de Fluke, que tiene la capacidad de generar arritmias.

Aunque el ProSim no incluye los latidos específicos, su capacidad para generar patrones desencadenantes de arritmias potencialmente mortales como la contracción ventricular prematura (V), **Vtach** y **Vfib**, permitirá asociar estos patrones con la identificación de arritmias en la aplicación móvil.

Para la validación en un entorno controlado, se utilizará un módulo Bluetooth **HC-05** y un Arduino para obtener la señal de ECG del ProSim en tiempo real. La señal será muestreada a 125Hz (frecuencia de interés) y transmitida a la aplicación móvil para verificar su capacidad de monitoreo y detección de las arritmias correctamente.

6.5.13. Entregables de la aplicación móvil.

Para la entrega del proyecto, se desarrollarán los siguientes entregables clave:

- **Manual de uso:** Un documento detallado que servirá como guía para que los usuarios aprendan a utilizar la aplicación correctamente. Este manual incluirá instrucciones claras sobre las funcionalidades principales, cómo navegar en la app, y cómo realizar las principales acciones.
- **Documento de tratamiento de datos personales:** Aunque la versión controlada de la aplicación no será abierta al público, se desarrollará un documento que describe cómo se manejarán los datos personales y sensibles en la app. Este documento será un simulacro de lo que debe cumplir la aplicación para ser validada por entidades externas o para recopilar información real. Cumplirá con las leyes y normativas que rigen el uso de datos personales en Colombia.
- **Código fuente en GitHub:** Todo el código del frontend y REST API será almacenado en un repositorio privado de GitHub, con acceso solo para los colaboradores y evaluadores mediante claves SSH. Esto garantizará el control y seguimiento adecuado del desarrollo.
- **APK para Android:** La versión alfa de la aplicación móvil será empaquetada en un archivo APK para Android. Este APK estará disponible exclusivamente para colaboradores y evaluadores, permitiendo la instalación y prueba de la aplicación en dispositivos Android.

Capítulo 7

Resultados y Discusión

Este proyecto logró resultados consistentes a lo largo de las distintas fases que respaldaron el desarrollo de una aplicación móvil diseñada para identificar arritmias cardíacas mediante procesamiento digital de señales y aprendizaje automático. Como se mencionó en la sección anterior, el desarrollo siguió un enfoque estructurado y lineal.

7.1. Librería para segmentación, extracción de caracteristicas y entrenamiento de modelos

Las tres primeras fases del proyecto concluyeron con la extracción de características y el entrenamiento de modelos de aprendizaje automático para la clasificación de arritmias potencialmente mortales. Todo este proceso puede observarse en los Notebooks de Jupyter incluidos en los anexos. Sin embargo, dicho procedimiento fue manual y extenso, lo que motivó la creación de una librería en Python denominada ECG_Signals. Esta librería centraliza los códigos clave necesarios para realizar estas tareas de manera eficiente, reduciendo significativamente las definiciones de programación requeridas. ECG_Signals abarca desde la segmentación y almacenamiento de la base de datos hasta la automatización del entrenamiento y guardado de los modelos de inteligencia artificial enfocados en la identificación de arritmias.

Con base en lo anterior, se lograron entrenar seis modelos con arquitecturas basadas en redes neuronales convolucionales (CNN). De estos modelos, dos combinaron arquitecturas CNN con redes neuronales recurrentes (RNN), mientras que otros dos implementaron enfoques de aprendizaje por transferencia (transfer learning).

Al realizar de repeticiones de los modelos, se evidenció el uso eficiente de la librería ECG_Signals para automatizar y validar el entrenamiento, permitiendo un análisis detallado y replicable de los resultados obtenidos.

Se puede observar la definición *entrenarModeloGuardarlo1D*, una función del tipo void, lo que indica que no retorna ningún valor. Sin embargo, esta realiza un proceso fundamental: carga los datos de entrenamiento y validación, los envía al modelo que se crea en la misma función e inicia su proceso de entrenamiento. Al finalizar, guarda el modelo en la ruta especificada y almacena los resultados visuales, como la gráfica de Accuracy y Loss, la matriz de confusión y el reporte de métricas. Cabe destacar que esta función se repite tres veces y, con tan solo 9 líneas de código, permite asegurar el entrenamiento y almacenamiento de tres modelos 1D de manera eficiente y automatizada. En las Figuras 7.1 7.2 y 7.3 se observa los resultados arrojados por ejecutar la línea 3 del *Listing 7.1* que corresponden a gráfica de Accuracy y Loss, gráfica de matriz de confusión y reporte de métricas.

```
1 pathTrain = ("Repeticiones-Data/6_Clases/
    train_df_2000_80_20_Seis_Clases_Uno.csv")
2 pathVal = ("Repeticiones-Data/6_Clases/
    val_df_2000_80_20_Seis_Clases_Uno.csv")
3 entrenarModeloGuardarlo1D(pathTrain, pathVal, 64, 30, "Modelos-
    Repeticiones/Prueba_Tres_Nucleos_6_clases_2000_80_20_Uno", ['V', '!
    ', 'F', 'f', 'N', 'O'], 6, [64,128,256], [3,5,10], [1,1,1])
4 pathTrain = ("Repeticiones-Data/6_Clases/
    train_df_2000_80_20_Seis_Clases_Dos.csv")
5 pathVal = ("Repeticiones-Data/6_Clases/
    val_df_2000_80_20_Seis_Clases_Dos.csv")
6 entrenarModeloGuardarlo1D(pathTrain, pathVal, 64, 30, "Modelos-
    Repeticiones/Prueba_Tres_Nucleos_6_clases_2000_80_20_Dos", ['V', '!
    ', 'F', 'f', 'N', 'O'], 6, [64,128,256], [3,5,10], [1,1,1])
7 pathTrain = ("Repeticiones-Data/6_Clases/
    train_df_2000_80_20_Seis_Clases_Tres.csv")
8 pathVal = ("Repeticiones-Data/6_Clases/
    val_df_2000_80_20_Seis_Clases_Tres.csv")
9 entrenarModeloGuardarlo1D(pathTrain, pathVal, 64, 30, "Modelos-
    Repeticiones/Prueba_Tres_Nucleos_6_clases_2000_80_20_Tres", ['V', '
    !', 'F', 'f', 'N', 'O'], 6, [64,128,256], [3,5,10], [1,1,1])
```

Listing 7.1: Definición para hacer repetición con el modelo 1D TRIPLE NUCLEO

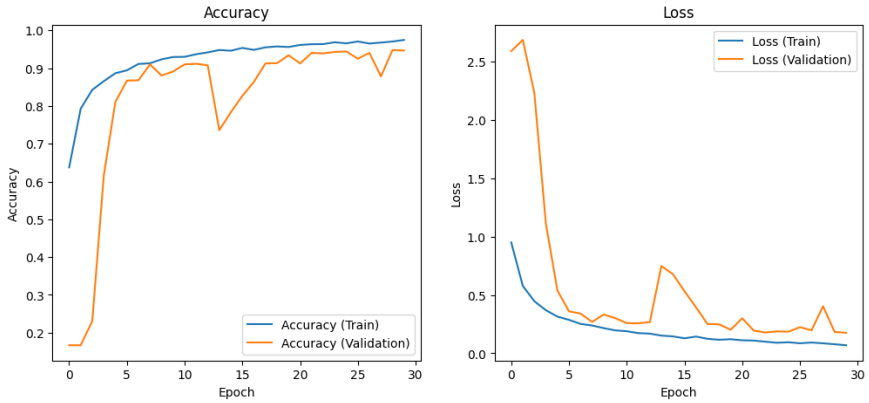


Figura 7.1: Gráfica de Accuracy y Loss

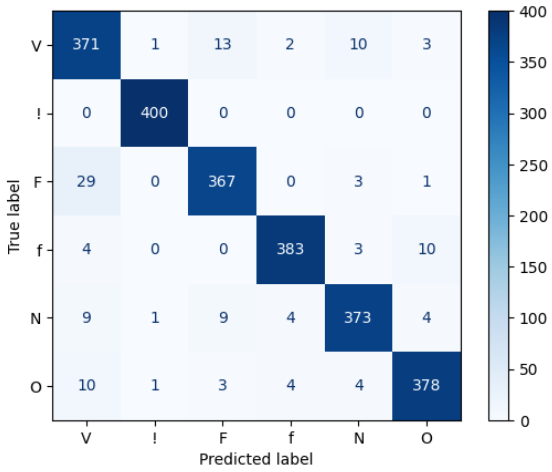


Figura 7.2: Gráfica de matriz de confusión

Reporte de Clasificación:

	precision	recall	f1-score	support
Clase 0	0.88	0.93	0.90	400
Clase 1	0.99	1.00	1.00	400
Clase 2	0.94	0.92	0.93	400
Clase 3	0.97	0.96	0.97	400
Clase 4	0.95	0.93	0.94	400
Clase 5	0.95	0.94	0.95	400
accuracy			0.95	2400
macro avg	0.95	0.95	0.95	2400
weighted avg	0.95	0.95	0.95	2400

Figura 7.3: Reporte de métricas

7.2. Desempeño de los modelos

7.2.1. Benchmarking de los seis modelos en seis clases.

Se evaluaron seis modelos, y en los modelos 2D existen tres caracterizaciones, en el proceso de entrenamiento se utilizaron 2000 muestras por clase en donde 400 muestras fueron utilizadas para validar. Esto dio como resultado 12 matrices de confusión tal como se aprecian en las Figuras 7.4, 7.5, 7.6 y 7.7, los modelos son capaces de hacer predicciones e identificar diferentes arritmias cardíacas.

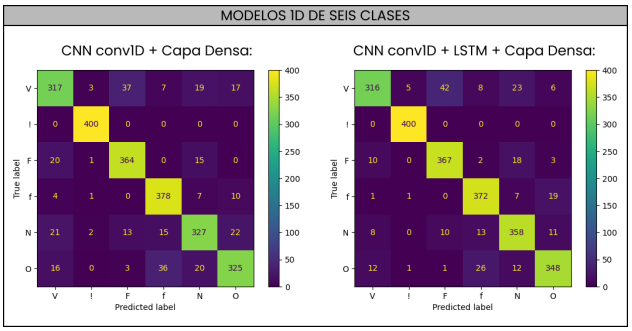


Figura 7.4: Modelos 1D de seis clases

En los modelos 1D presentados en la Figura 7.4, se observa que la clase con el mayor número de verdaderos positivos en ambos modelos fue (!), correspondiente a la Onda de Aleteo Ventricular. Esto se debe a que dicha onda posee una morfología muy particular y fácil de identificar.

Por otro lado, la clase con el menor número de verdaderos positivos fue (V), que representa la Onda de Contracción Ventricular Prematura. Este resultado es preocupante, ya que esta clase registró el mayor número de falsos negativos. En un proyecto enfocado en la predicción de episodios de muerte súbita cardíaca, este comportamiento es crítico, pues implica que numerosos casos reales no fueron detectados correctamente por los modelos.

Finalmente, al analizar en qué clase los modelos presentaron mayores confusiones, se evidencia que ambos modelos confundieron frecuentemente la clase (F), correspondiente a la Fusión de Ventricular y Normal.

En los modelos 2D presentados en la Figura 7.5, se observa que la clase con el mayor número de verdaderos positivos en ambos modelos fue (!), correspondiente a la Onda de Aleteo Ventricular. Esto se debe a que dicha onda posee una morfología muy particular y fácil de identificar.

Por otro lado, la clase con el menor número de verdaderos positivos fue (V) y (O), que representa la Onda de Contracción Ventricular Prematura y Otras clases. Encontramos una

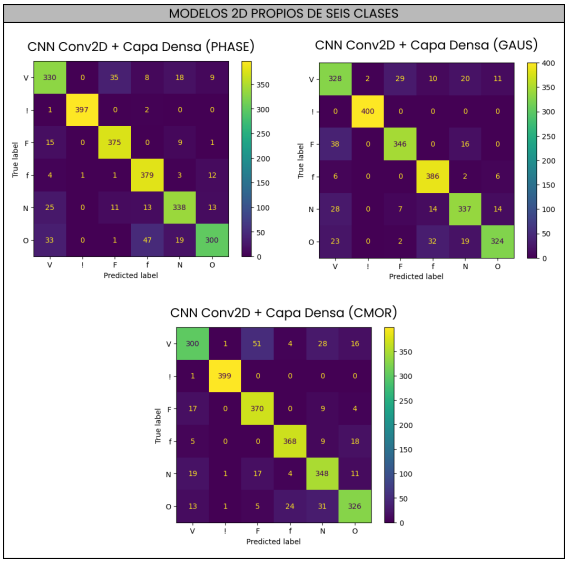


Figura 7.5: Modelos 2D de seis clases

tendencia que los modelos en 2D sin importar la caracterización posee el misma dificultad de los modelos 1D.

Por otro lado, en los resultados obtenidos de los modelos de transfer learning, se observaron similitudes en el comportamiento de las clases. Sin embargo, en el caso presentado en la Figura 7.7, utilizando la caracterización CMOR con la arquitectura VGG16, se evidenció una mejora en la predicción de las clases. La clase (!), correspondiente a la Onda de Aleteo Ventricular, continúa siendo la que registra el mayor número de verdaderos positivos.

Por otro lado, la clase con el menor desempeño fue (N), que representa una señal ECG normal. Aunque este resultado tampoco es ideal, ya que podría generar falsas alarmas, es preferible en comparación con una situación donde una alarma crítica no sea identificada ni notificada. En un contexto donde el objetivo es la detección temprana de arritmias potencialmente mortales, generar una falsa alarma resulta menos riesgoso que omitir una señal importante.

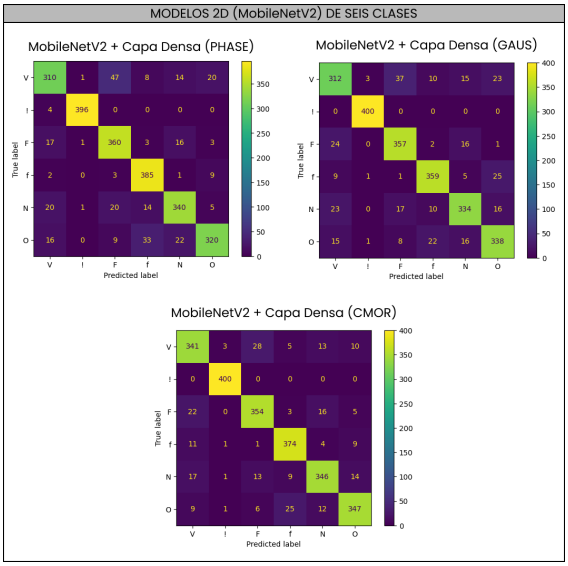


Figura 7.6: Modelo 2D MobileNetV2 de seis clases

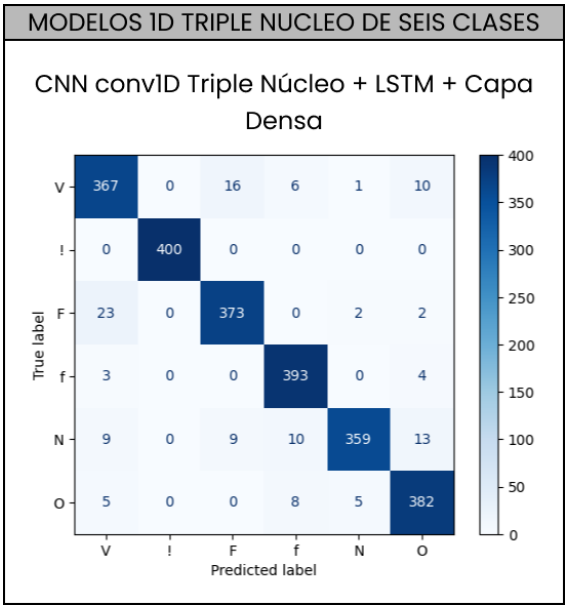


Figura 7.8: MODELO 1D TRIPLE NUCLEO DE SEIS CLASES

Hasta el momento los resultados indican que los mejores modelos son los de transfer learning especificamente el modelo VGG16 sin embargo este modelo es costoso computacionalmente, ya que usa imagenes y de tres canales y procesar un vector a una imagen suele ser costoso y extenso lo que indica que es lento al momento de predecir, por eso se exploro una nueva

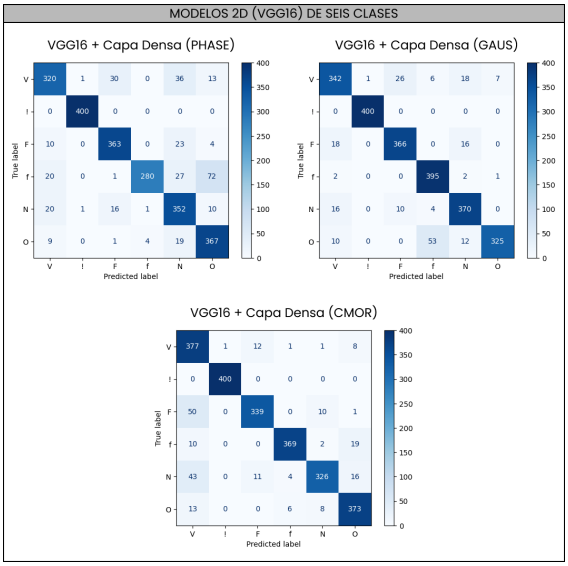


Figura 7.7: Modelo 2D VVG16 de seis clases

alternativa de un modelo 1D que combina el modelo CNN conv1D + LSTM + Capa Densa y multiplica la entrada del vector tres veces a eso se le llama Head.

```
def head(numFilters, numKernelSize,input, mul):
    conv1 = Conv1D(filters=numFilters[0]*mul, kernel_size=
       numKernelSize[0], activation='relu')(input)
    pool1 = MaxPooling1D(pool_size=2)(conv1)
    norm1 = BatchNormalization()(pool1)
    drop1 = Dropout(0.5)(norm1)

    conv1_1 = Conv1D(filters=numFilters[1]*mul, kernel_size=
       numKernelSize[1], activation='relu')(drop1)
    pool1_1 = MaxPooling1D(pool_size=2)(conv1_1)
    norm1_1 = BatchNormalization()(pool1_1)
    drop1_1 = Dropout(0.5)(norm1_1)

    conv1_2 = Conv1D(filters=numFilters[2]*mul, kernel_size=
       numKernelSize[2], activation='relu')(drop1_1)
    pool1_2 = MaxPooling1D(pool_size=2)(conv1_2)
    norm1_2 = BatchNormalization()(pool1_2)
    drop1_2 = Dropout(0.5)(norm1_2)

    # Capa LSTM
    lstm1 = LSTM(100, return_sequences=True)(drop1_2)
    lstm1_1 = LSTM(100)(lstm1)
    flat1 = Flatten()(lstm1_1)
    return flat1
```

Listing 7.2: Definición Head para el modelo 1D triple nucleo

Como se aprecia en el *Listing 7.2*, la definición presentada espera como entrada hiperparámetros, tales como el número de filtros y el tamaño del filtro. Es importante destacar que cada head del modelo posee hiperparámetros diferentes, lo que puede generar variaciones en los pesos del modelo al ser entrenado.

Al analizar la matriz de confusión, se observó que la clase con el menor desempeño fue (N). Sin embargo, de 400 predicciones, el modelo logró acertar 359, lo que representa un 89.7 % de precisión. Al compararlo con el mejor modelo de VGG16, también para la clase (N), encontramos que, de 400 predicciones, acertó 326, lo que indica un 81.5 % de precisión.

A continuación, se presentarán las tablas con las métricas de cada uno de estos modelos, lo que permitirá analizar de manera más clara esta diferencia de rendimiento. No obstante, este análisis abre la posibilidad de optar por la arquitectura 1D, principalmente al considerar el costo computacional.

MODELOS 1D DE SEIS CLASES					
Arquitectura	clases	Métricas de Desempeño			
		Precisión	Recall	F1 Score	Exactitud Clasificación total
CNN conv1D + Capa Densa	V	0.84	0.79	0.81	0.88
	!	0.98	1.00	0.99	
	F	0.87	0.91	0.89	
	f	0.87	0.94	0.90	
	N	0.84	0.82	0.83	
	O	0.87	0.81	0.84	

Arquitectura	clases	Métricas de Desempeño			
		Precisión	Recall	F1 Score	Exactitud Clasificación total
CNN conv1D + LSTM + Capa Densa	V	0.91	0.79	0.85	0.90
	!	0.98	1.00	0.99	
	F	0.87	0.92	0.90	
	f	0.88	0.93	0.91	
	N	0.86	0.90	0.88	
	O	0.90	0.87	0.88	

Figura 7.9: Tablas de métricas - modelos 1D de seis clases

MODELOS 2D PROPIOS DE SEIS CLASES					
Arquitectura	clases	Métricas de Desempeño			
		Precisión	Recall	F1 Score	Exactitud Clasificación total
CNN Conv2D + Capa Densa (PHASE)	V	0.81	0.82	0.82	0.88
	!	1.00	0.99	0.99	
	F	0.89	0.94	0.91	
	f	0.84	0.95	0.89	
	N	0.87	0.84	0.86	
	O	0.90	0.75	0.82	
Arquitectura	clases	Métricas de Desempeño			
		Precisión	Recall	F1 Score	Exactitud Clasificación total
CNN Conv2D + Capa Densa (GAUS)	V	0.78	0.82	0.80	0.88
	!	1.00	1.00	1.00	
	F	0.90	0.86	0.88	
	f	0.87	0.96	0.92	
	N	0.86	0.84	0.85	
	O	0.91	0.81	0.86	
Arquitectura	clases	Métricas de Desempeño			
		Precisión	Recall	F1 Score	Exactitud Clasificación total
MobileNetV2 + Capa Densa (CMOR)	V	0.81	0.82	0.82	0.88
	!	1.00	0.99	0.99	
	F	0.89	0.94	0.91	
	f	0.84	0.95	0.89	
	N	0.87	0.84	0.86	
	O	0.90	0.75	0.82	

Figura 7.10: Tablas de métricas - Modelos 2D de seis clases

MODELOS 2D (MobileNetV2) DE SEIS CLASES					
Arquitectura	clases	Métricas de Desempeño			
		Precisión	Recall	F1 Score	Exactitud Clasificación total
MobileNetV2 + Capa Densa (PHASE)	V	0.84	0.78	0.81	0.88
	!	0.99	0.99	0.99	
	F	0.82	0.90	0.86	
	f	0.87	0.96	0.91	
	N	0.87	0.85	0.86	
	O	0.90	0.80	0.85	

Arquitectura	clases	Métricas de Desempeño			
		Precisión	Recall	F1 Score	Exactitud Clasificación total
MobileNetV2 + Capa Densa (GAUS)	V	0.81	0.78	0.80	0.88
	!	0.99	1.00	0.99	
	F	0.85	0.89	0.87	
	f	0.89	0.90	0.89	
	N	0.87	0.83	0.85	
	O	0.84	0.84	0.84	

Arquitectura	clases	Métricas de Desempeño			
		Precisión	Recall	F1 Score	Exactitud Clasificación total
MobileNetV2 + Capa Densa (CMOR)	V	0.85	0.85	0.85	0.90
	!	0.99	1.00	0.99	
	F	0.88	0.89	0.88	
	f	0.90	0.94	0.92	
	N	0.88	0.86	0.87	
	O	0.90	0.87	0.88	

Figura 7.11: Tablas de métricas - Modelo MobileNetV2 de seis clases

MODELOS 2D (VGG16) DE SEIS CLASES					
Arquitectura	clases	Métricas de Desempeño			
		Precisión	Recall	F1 Score	Exactitud Clasificación total
VGG16 + Capa Densa (PHASE)	V	0.84	0.80	0.82	0.87
	!	1.00	1.00	1.00	
	F	0.88	0.91	0.90	
	f	0.98	0.70	0.82	
	N	0.77	0.88	0.82	
	O	0.79	0.92	0.85	
Arquitectura	clases	Métricas de Desempeño			
		Precisión	Recall	F1 Score	Exactitud Clasificación total
VGG16 + Capa Densa (GAUS)	V	0.88	0.85	0.87	0.92
	!	1.00	1.00	1.00	
	F	0.91	0.92	0.91	
	f	0.86	0.99	0.92	
	N	0.89	0.93	0.90	
	O	0.98	0.81	0.89	
Arquitectura	clases	Métricas de Desempeño			
		Precisión	Recall	F1 Score	Exactitud Clasificación total
VGG16 + Capa Densa (CMOR)	V	0.76	0.94	0.84	0.91
	!	1.00	1.00	1.00	
	F	0.94	0.85	0.89	
	f	0.97	0.92	0.95	
	N	0.94	0.81	0.87	
	O	0.89	0.93	0.91	

Figura 7.12: Tablas de métricas - Modelo VGG16 de seis clases

MODELOS 1D TRIPLE NUCLEO DE SEIS CLASES					
Arquitectura	clases	Métricas de Desempeño			
		Precisión	Recall	F1 Score	Exactitud Clasificación total
CNN conv1D Triple Núcleo + LSTM + Capa Densa	V	0.90	0.92	0.91	0.95
	!	1.00	1.00	1.00	
	F	0.94	0.93	0.93	
	f	0.94	0.98	0.96	
	N	0.98	0.90	0.94	
	O	0.93	0.95	0.94	

Figura 7.13: Tablas de métricas - Modelo tripe núcleo de seis clases

Con respecto a los resultados, estos son muy congruentes con las matrices de confusión y lo mencionado anteriormente. Al observar la exactitud general de cada modelo, se calculó una media que dio como resultado 0.895 para todos los modelos de inteligencia artificial entrenados. Esto indica que los rendimientos generales obtenidos hasta el momento en el proyecto no son satisfactorios.

Ahora bien, como se ha mencionado, en el desarrolló del proyecto se decidió entrenar los modelos con cinco y cuatro clases, descartando las clases que no son de interés. Para evitar repetir todo este proceso con los 12 resultados obtenidos, se seleccionó un candidato por cada dimensión. Para la dimensión 1D, **se eligió la arquitectura de triple núcleo LSTM, y para la dimensión 2D, se seleccionó la arquitectura VGG16 con las tres caracterizaciones**.

7.2.2. Benchmarking de los dos modelos seleccionados en cinco y cuatro clases.

El objetivo fundamental de reducir las clases es evitar sesgar el proyecto con etiquetas de arritmias que no son de interés. El modelo de inteligencia artificial tiene como propósito identificar arritmias potencialmente mortales asociadas a un cuadro de muerte súbita cardíaca. De esta forma, se pretende integrar dicho modelo de monitoreo en una aplicación móvil para el constante seguimiento de señales ECG de un paciente que utiliza un holter.

Todo este vector, junto a las predicciones generadas por el modelo, será almacenado en una base de datos a la cual el personal médico encargado podrá acceder. Si se utiliza un modelo de cinco clases en el que se incluye la clase (N), no será necesario usar un modelo biclase que distinga entre un signo normal o una arritmia. En cambio, si se opta por un modelo de cuatro clases en el que la clase (N) no está presente, sí será necesario el uso de un modelo biclase. Este modelo inicial identificaría si la señal corresponde a una arritmia y, en caso afirmativo, se emplearía el modelo principal del proyecto para clasificar e identificar la arritmia correspondiente, de esta forma pasamos a los resultado de los dos modelos seleccionados para ser entrenados con cinco y cuatro clases.

7.2.2.1. Modelos con cinco clases:

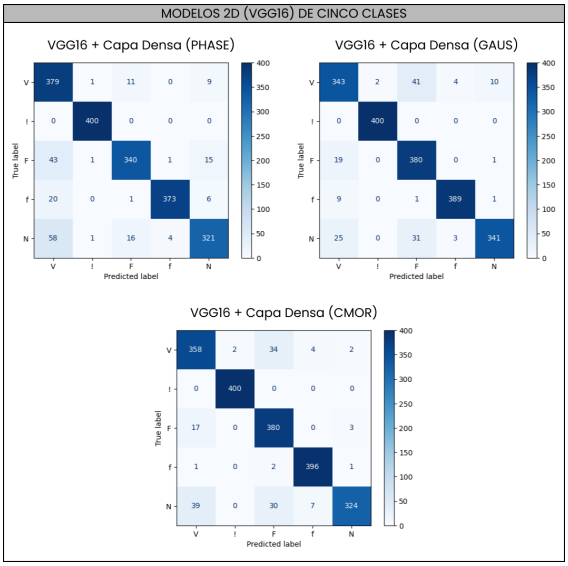


Figura 7.14: Matrices de confusión - Modelo VGG16 de cinco clases

Observando las matrices de confusión, se puede evidenciar una mejora significativa en comparación con los mismos modelos entrenados con seis clases. Sin embargo, se observa que la etiqueta $(!)$ continúa siendo la mejor en predicción, alcanzando un 100 % de precisión en todas las caracterizaciones. Por otro lado, las etiquetas que presentan mayor margen de error son (V) y (N).

De manera general, se podría afirmar que la mejor caracterización corresponde a la caracterización Gaus.

MODELOS 2D (VGG16) DE CINCO CLASES					
Arquitectura	clases	Métricas de Desempeño			
		Precisión	Recall	F1 Score	Exactitud Clasificación total
VGG16 + Capa Densa (PHASE)	V	0.86	0.90	0.88	0.93
	!	1.00	1.00	1.00	
	F	0.85	0.95	0.90	
	f	0.97	0.99	0.98	
	N	0.98	0.81	0.89	
Arquitectura	clases	Métricas de Desempeño			
		Precisión	Recall	F1 Score	Exactitud Clasificación total
VGG16 + Capa Densa (GAUS)	V	0.87	0.85	0.86	0.93
	!	1.00	1.00	1.00	
	F	0.84	0.92	0.89	
	f	0.98	0.97	0.98	
	N	0.97	0.85	0.91	
Arquitectura	clases	Métricas de Desempeño			
		Precisión	Recall	F1 Score	Exactitud Clasificación total
VGG16 + Capa Densa (CMOR)	V	0.86	0.90	0.88	0.93
	!	1.00	1.00	1.00	
	F	0.85	0.95	0.90	
	f	0.97	0.99	0.98	
	N	0.98	0.81	0.89	

Figura 7.15: Tablas de metricas - Modelo VGG16 de cinco clases

Al analizar las métricas en la *figura 7.15* y observar la exactitud total de cada modelo con cada caracterización, se evidencia que todos alcanzan un rendimiento del 93 %. Esto sugiere que la caracterización *Gaus* fue la más estable al trabajar con cinco etiquetas.

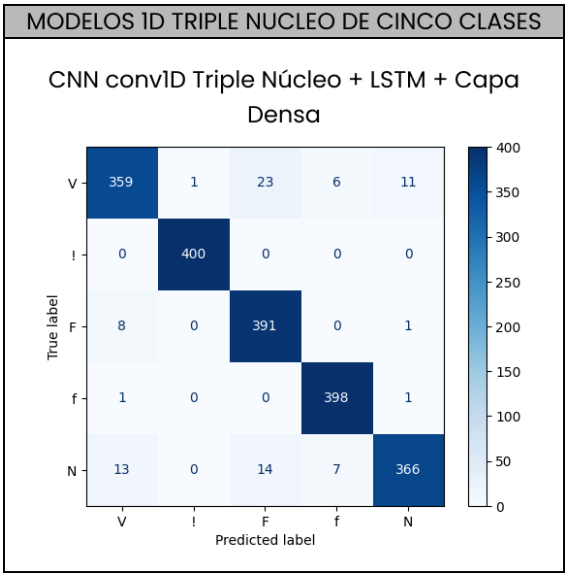


Figura 7.16: Matrices de confusión - Modelo tripe núcleo de cinco clases

Observando la matriz de confusión, se puede evidenciar una mejora significativa en comparación con el modelo de seis clases. Sin embargo, las etiquetas que presentan un mayor margen de error son (V) y (N). No obstante, si se realiza un recuento de todos los casos de emergencia en los que no se notificó, de un total de 1200 casos de emergencia, solo en 13 ocasiones no se generó la alerta correspondiente.

Por otro lado, en cuanto a las métricas, se observa que, de forma general, el rendimiento mejoró significativamente frente a las seis clases, aumentando la exactitud de un 95 % a un 96 %, tal como se muestra en la Figura 7.17.

MODELOS 1D TRIPLE NUCLEO DE CINCO CLASES					
Arquitectura	clases	Métricas de Desempeño			
		Precisión	Recall	F1 Score	Exactitud Clasificación total
CNN conv1D Triple Núcleo + LSTM + Capa Densa	V	0.94	0.90	0.92	0.96
	!	1.00	1.00	1.00	
	F	0.91	0.98	0.94	
	f	0.97	0.99	0.98	
	N	0.97	0.92	0.94	

Figura 7.17: Tabla de métricas - Modelo tripe núcleo de cinco clases

7.2.2.2. Modelos con cuatro clases:

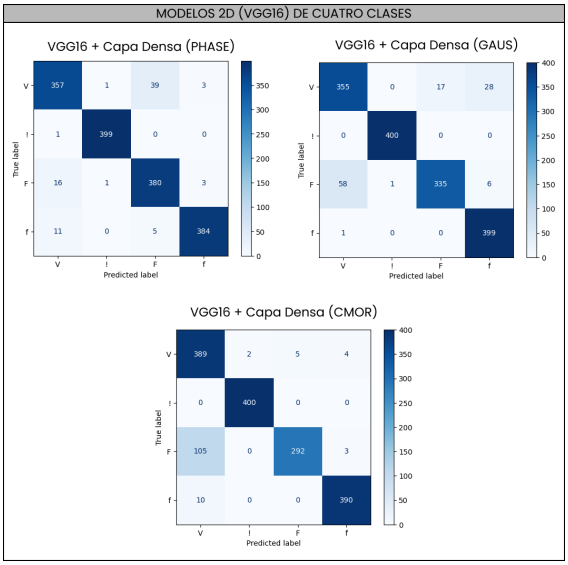


Figura 7.18: Matrices de confusión - Modelo VGG16 de cuatro clases

Observando las matrices de confusión, se puede evidenciar una mejora significativa en comparación con los mismos modelos entrenados con seis clases. Sin embargo, al ser un modelo que no posee la clase (N), se generan confusiones entre las clases al momento de predecir. Se evidencia una cantidad considerable de falsos positivos con la etiqueta (V), la cual se confunde frecuentemente con la etiqueta (F).

Por otro lado, al observar de forma general, la caracterización con mejor rendimiento utilizando cuatro clases fue *Phase*. Esto contrasta con los resultados obtenidos previamente, donde en seis clases la mejor caracterización fue *Cmor*, y en cinco clases fue *Gaus*. Esta variabilidad entre clases sugiere que el desempeño del modelo está influenciado por las características específicas de los datos y la cantidad de etiquetas consideradas.

MODELOS 2D (VGG16) DE CUATRO CLASES

Arquitectura	clases	Métricas de Desempeño			
		Precisión	Recall	F1 Score	Exactitud Clasificación total
VGG16 + Capa Densa (PHASE)	V	0.93	0.89	0.91	0.95
	!	1.00	1.00	1.00	
	F	0.90	0.95	0.92	
	f	0.98	0.96	0.97	

Arquitectura	clases	Métricas de Desempeño			
		Precisión	Recall	F1 Score	Exactitud Clasificación total
VGG16 + Capa Densa (GAUS)	V	0.86	0.89	0.87	0.93
	!	1.00	1.00	1.00	
	F	0.95	0.84	0.89	
	f	0.92	1.00	0.96	

Arquitectura	clases	Métricas de Desempeño			
		Precisión	Recall	F1 Score	Exactitud Clasificación total
VGG16 + Capa Densa (CMOR)	V	0.77	0.97	0.86	0.92
	!	1.00	1.00	1.00	
	F	0.98	0.73	0.84	
	f	0.98	0.97	0.98	

Figura 7.19: Tablas de metricas - Modelo VGG16 de cuatro clases

Al analizar las métricas en la *figura 7.19* y observar la exactitud total de cada modelo con cada caracterización, los resultados fuerón 95 % para Phase, 93 % para Gaus y 92 % para Cmor. Esto sugiere que la caracterización *Phase* fue la más estable al trabajar con cuatro etiquetas.

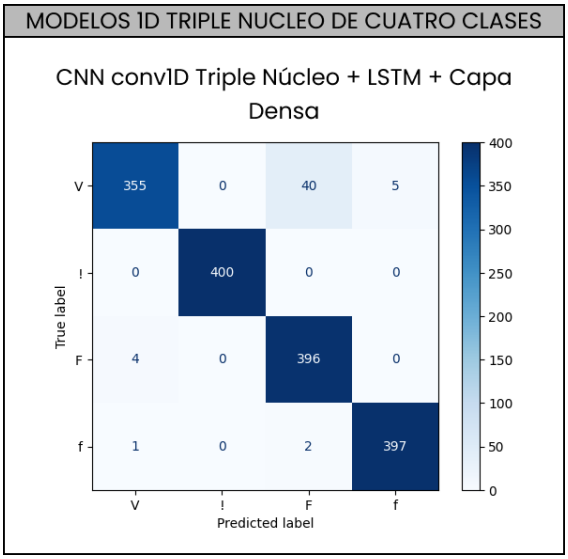


Figura 7.20: Matrices de confusión - Modelo tripe nucleo de cuatro clases

Una vez mas se evidencia que este modelo es estable, en donde ahora que posee cuatro clases hay muy pocos falsos positivos, auqneu por el lado de los falsos negativos se concentran en la etiqueta (V) esto indica que esta etiqueta ha sido con el mayor reto de identificar, pero aunque cuatro etiquetas son de interes igual se genera la alerta ya que no hay etiqueta (N).

Por otro lado, en cuanto a las métricas, se observa que, de forma general, el rendimiento mejoró significativamente frente a las seis clases y cinco clase, aumentando la exactitud de un 95 % a un 97 %, tal como se muestra en la *Figura 7.21*.

MODELOS 1D TRIPLE NUCLEO DE CUATRO CLASES					
Arquitectura	clases	Métricas de Desempeño			
		Precisión	Recall	F1 Score	Exactitud Clasificación total
CNN conv1D Triple Núcleo + LSTM + Capa Densa	V	0.99	0.89	0.93	0.97
	!	1.00	1.00	1.00	
	F	0.90	0.99	0.95	
	f	0.99	0.99	0.99	

Figura 7.21: Tabla de metricas - Modelo tripe nucleo de cuatro clases

De esta forma, al finalizar el análisis de los resultados, se evaluaron los criterios establecidos: rendimiento y costo computacional. Al considerar ambos criterios, se concluyó que el modelo *triple núcleo* en 1D fue el modelo seleccionado para iniciar la actividad de repeticiones para comprobar la estabilidad del modelo.

7.2.3. Repeticiones del modelo seleccionado

Empleando el uso de la librería *ECG_ Signals*, presentada en el primer resultado, se pudo realizar el proceso de repeticiones con pocas líneas de código. Se llevaron a cabo cinco repeticiones con distribuciones diferentes de la base de datos en los conjuntos de entrenamiento y validación para cada tipo de clases: seis, cinco y cuatro. Posteriormente, se generó un cuadro con los promedios generales de cada métrica en cada repetición, calculando una media de las cinco repeticiones por métrica y, adicionalmente, la desviación estándar. Esto permitió analizar si el modelo se comporta de manera estable al trabajar con distintas formas de distribución de los datos de entrenamiento y validación.

MODELO 1D TRIPLE NUCLEO DE SEIS CLASES							
Métricas	Repeticiones para estabilidad						
	Uno	Dos	Tres	Cuatro	Cinco	Promedio	Desviación Estándar
Accuracy	0.95	0.93	0.95	0.95	0.88	0.932	0.030
Precisión	0.95	0.93	0.95	0.95	0.90	0.936	0.022
Recall	0.95	0.93	0.95	0.95	0.88	0.932	0.030
F1 Score	0.95	0.93	0.95	0.95	0.88	0.932	0.030

Figura 7.22: Cuadro de repeticiones del modelo seleccionado seis clases

Se enfoca la atención en la desviación estándar, que indica mayor variación en las métricas en comparación con los modelos de 4 y 5 clases, especialmente en *accuracy*, *recall* y *F1 score* (todas con 0.030). Esto revela cierta inestabilidad al cambiar la distribución.

La repetición 5 muestra un desempeño más bajo en todas las métricas (alrededor de 0.88), lo que influye significativamente en el promedio y sugiere que la división aleatoria de los datos afecta el balance en esta configuración.

MODELO 1D TRIPLE NUCLEO DE CINCO CLASES							
Métricas	Repeticiones para estabilidad						
	Uno	Dos	Tres	Cuatro	Cinco	Promedio	Desviación Estándar
Accuracy	0.94	0.94	0.96	0.92	0.92	0.936	0.017
Precisión	0.95	0.94	0.96	0.93	0.93	0.942	0.013
Recall	0.94	0.94	0.96	0.92	0.92	0.936	0.017
F1 Score	0.94	0.94	0.96	0.92	0.92	0.936	0.017

Figura 7.23: Cuadro de repeticiones del modelo seleccionado cinco clases

Este modelo tiene la menor variabilidad entre las métricas (0.017 para *accuracy*, *recall* y *F1 score*, y 0.013 para *precisión*), lo que indica mayor estabilidad al realizar las repeticiones. Las repeticiones 4 y 5 presentan valores ligeramente inferiores (0.92), pero el impacto en el promedio es menor en comparación con el modelo de 6 clases. Este modelo parece ser más robusto frente a cambios en la división de datos.

MODELO 1D TRIPLE NUCLEO DE CUATRO CLASES							
Métricas	Repeticiones para estabilidad						
	Uno	Dos	Tres	Cuatro	Cinco	Promedio	Desviación Estándar
Accuracy	0.95	0.93	0.96	0.90	0.96	0.940	0.025
Precisión	0.95	0.94	0.96	0.92	0.96	0.946	0.017
Recall	0.95	0.93	0.96	0.90	0.96	0.94	0.025
F1 Score	0.95	0.93	0.96	0.89	0.96	0.938	0.029

Figura 7.24: Cuadro de repeticiones del modelo seleccionado cuatro clases

La desviación estándar es baja, similar al modelo de 5 clases (0.025 para *accuracy*, *recall*, y *F1 score*, y 0.017 para *precisión*), pero presenta ligeras variaciones en comparación con las métricas de precisión. El modelo alcanza su peor rendimiento en la repetición 4 (0.90 en *accuracy* y *recall*, 0.89 en *F1 score*). Este comportamiento puede deberse a desequilibrios específicos en la partición aleatoria. A pesar de ello, el modelo sigue mostrando un rendimiento ligeramente superior en comparación con los de 5 y 6 clases en algunas métricas.

Todos los modelos tuvieron desviaciones estándar muy bajas, por lo que, de forma general, la estabilidad es buena, aunque algunos modelos muestran un mejor rendimiento que otros. Por esta razón, y considerando los resultados de rendimiento, se elige usar el modelo con cuatro clases. Este modelo tiene los mejores valores promedio en *accuracy* (0.940), *recall* (0.940), y *F1 score* (0.938), lo que lo hace ideal si se prioriza un mejor desempeño en métricas absolutas.

7.3. Validar modelo con bases de datos indenpendientes

Como se mencionó en la sección de materiales y métodos, se eligio una bases de datos de interés para evaluar el uso del modelo a forma de testeo.

El proceso de segmentación y caracterización de las señales fue el mismo utilizado para entrenar el modelo. El modelo que se uso fue el de cuatro clases con triple núcleo y se utilizo el modelo guardado de la repetición tres.

La base de datos **Brno University of Technology ECG Signal Database with Annotations of P Wave (BUT PDB)**, la segmentación arrojó 527 muestras para la etiqueta *(V)*, 24 para la etiqueta *(!)*, 40 para la etiqueta *(F)*, y 53 para la etiqueta *(f)*. Se observa la misma limitación respecto a la cantidad reducida de datos en algunas etiquetas. Aun así, se realizó la predicción y se obtuvieron los siguientes resultados:

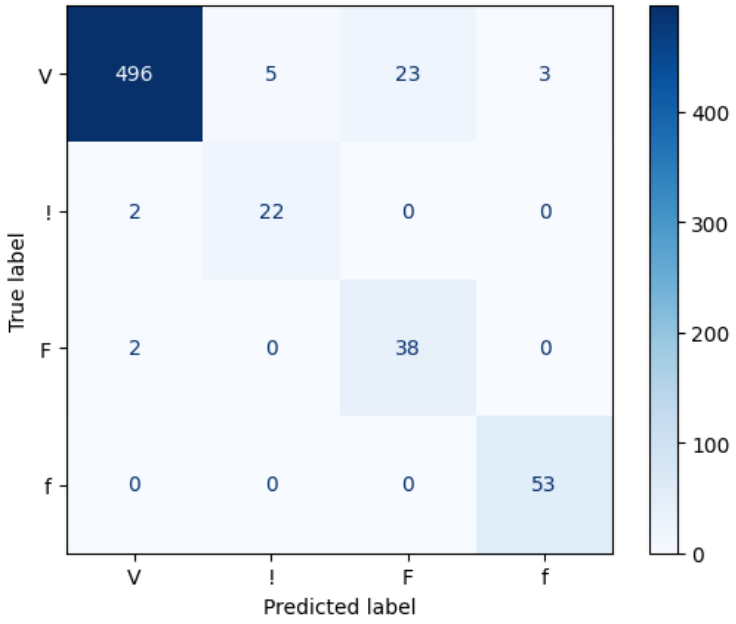


Figura 7.25: Matriz de confusión de la base BUT PDB.

Al analizar el reporte de métricas, se aprecia que el modelo obtuvo un rendimiento de exactitud del **95 %**, lo cual es coherente con los resultados observados durante las validaciones. Por otro lado, la precisión fue del **94.5 %**, con un total de **35 predicciones incorrectas** de 644 muestras.

7.4. Desarrollo de aplicación KIBO

7.4.1. Páginas principales de la aplicación móvil

Después de realizar las primeras actividades del desarrollo, se obtuvieron las páginas principales necesarias para el correcto funcionamiento de la aplicación. Estas se categorizan en **Login**, **Navbar** y **Appbar**, con la siguiente descripción funcional:

- **Login (2 páginas):**
  - **Página de selección:** Permite elegir entre las opciones de "Iniciar sesión" o "Registrarse". En esta versión controlada, solo está habilitada la opción de iniciar sesión.
  - **Página de inicio de sesión:** Contiene los campos para ingresar las credenciales (correo electrónico y contraseña). Posee dos variantes visuales:

- Caso de credenciales correctas: Se valida el ingreso y se redirige al sistema.
- Caso de credenciales incorrectas: Muestra un mensaje de error solicitando la corrección de los datos ingresados.

Navbar (Barra de navegación): La barra de navegación organiza las páginas principales de la aplicación en tres secciones:

- **Home:** Página principal que incluye todas las funcionalidades relacionadas con la **conexión Bluetooth**, permitiendo establecer y monitorear el enlace con el módulo HC-05, que es el que permite la conexión del Holter con el dispositivo móvil.
- **Gráficas:** Muestra las tendencias del ritmo cardíaco en tiempo real. Además, contiene un recuadro en la parte inferior que muestra la última predicción del sistema, la cual puede variar entre:
  - **Sin riesgo:** Valores normales y estabilidad en el ritmo cardiaco.
  - **Con riesgo:** Identificación de anomalías o patrones que representan un riesgo.
- **Ajuste de alarmas:** Permite configurar las siguientes opciones:
  - Habilitar la ubicación.
  - Activar las alarmas, seleccionando todas las etiquetas o alguna en específico.
  - Realizar pruebas locales para validar el funcionamiento con mensajes de texto.

Appbar (Barra superior de navegación): La barra superior contiene dos páginas adicionales enfocadas en la información del usuario:

- **Perfil de usuario:** Muestra la información fundamental del usuario, incluida su imagen de perfil. Esta página permite la edición y actualización de los datos.

- **Información general del usuario:** Contiene detalles personales, información de contacto de emergencia y datos médicos relevantes. Al igual que en el perfil, es posible editar y guardar los cambios realizados.

Las figuras 7.26, 7.27 y 7.28 representan de forma visual todo lo anteriormente mencionado.

Figura 7.26: UI/UX Login

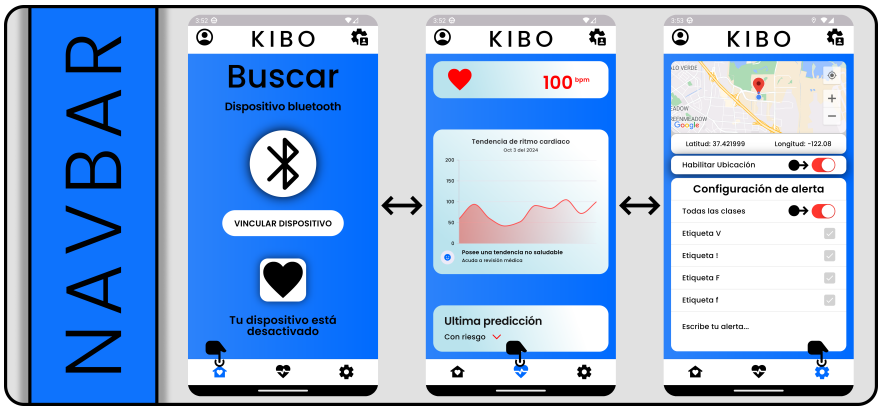


Figura 7.27: UI/UX Navbar

Figura 7.28: UI/UX Appbar

7.4.2. Conexión a dispositivos bluetooth

La aplicación móvil es capaz de conectarse a módulos Bluetooth, específicamente a módulos como el HC-05 o HC-06. A continuación, se presentan los resultados del flujo de conexión Bluetooth en la aplicación tal como se aprecia en la figura 7.29.

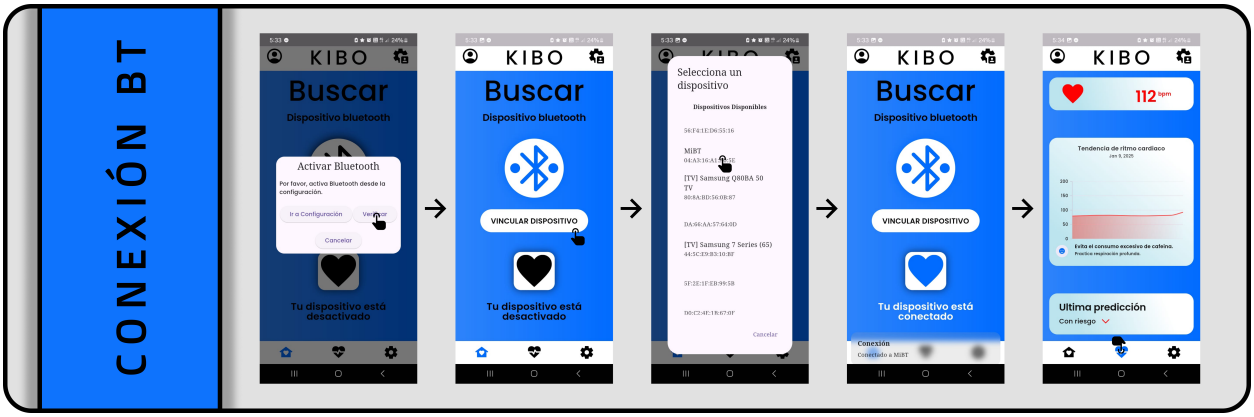


Figura 7.29: flujo BT

7.4.3. Documentación Rest API

En las actividades de las secciones *6.5.8* a *6.5.11*, se desarrolló la *REST API*, la cual es responsable de la comunicación interna entre la base de datos y la aplicación móvil. Esta API gestiona todos los flujos de consulta necesarios para cargar, crear, eliminar y modificar información.

El despliegue se realizó con éxito en un servidor VPS, y se generó una documentación utilizando la herramienta Swagger, la cual puede consultarse en la siguiente ruta: API Documentation.

Por otro lado, la base de datos fue desplegada en *MongoDB Atlas*. En la figura 7.30 se muestra la estructura final de las colecciones resultantes.

dbkibo

LOGICAL DATA SIZE: 5.85MB STORAGE SIZE: 5.98MB INDEX SIZE: 324KB TOTAL COLLECTIONS: 9

CREATE COLLECTION

Collection Name	Documents	Logical Data Size	Avg Document Size	Storage Size	Indexes	Index Size	Avg Index Size
credential	5	509B	102B	36KB	1	36KB	36KB
data_bpm	14	28.36KB	2.03KB	68KB	1	36KB	36KB
data_ecg	14	5.79MB	423.23KB	5.66MB	1	36KB	36KB
data_pred	14	34.51KB	2.47KB	48KB	1	36KB	36KB
doctor_user	6	841B	141B	36KB	1	36KB	36KB
emergency_user	5	1.61KB	331B	36KB	1	36KB	36KB
info_user	5	1.2KB	246B	36KB	1	36KB	36KB
medical_info	4	580B	145B	36KB	1	36KB	36KB
users	5	856B	172B	36KB	1	36KB	36KB

Figura 7.30: Contenido de base de datos

Teniendo en cuenta que las colecciones que almacenan únicamente información de usuarios (sin contar los datos de bpm, ecg y predicciones) ocupan aproximadamente 0.216 MB, y que los registros de las colecciones de data ocupan 5.63 MB, el espacio total utilizado por la base de datos es de aproximadamente 6 MB. Este espacio utilizado representa solo el 1.2 % de la capacidad total de 500 MB disponible.

7.4.4. Mensaje de alerta vía Whatsapp

Parte del cuarto objetivo específico del proyecto es que la aplicación móvil genere alertas de arritmias cardíacas. Para cumplir con este objetivo, se utilizó Twilio, que es una herramienta para generar mensajes de texto. A través de la API de KIBO, se configuró un flujo de envío de mensajes, de manera que, en conjunto con la información proporcionada por la aplicación móvil, se genera un mensaje de alerta que se envía a los números de emergencia registrados por cada usuario. En la figura 7.31, se puede observar el mensaje enviado por la aplicación.

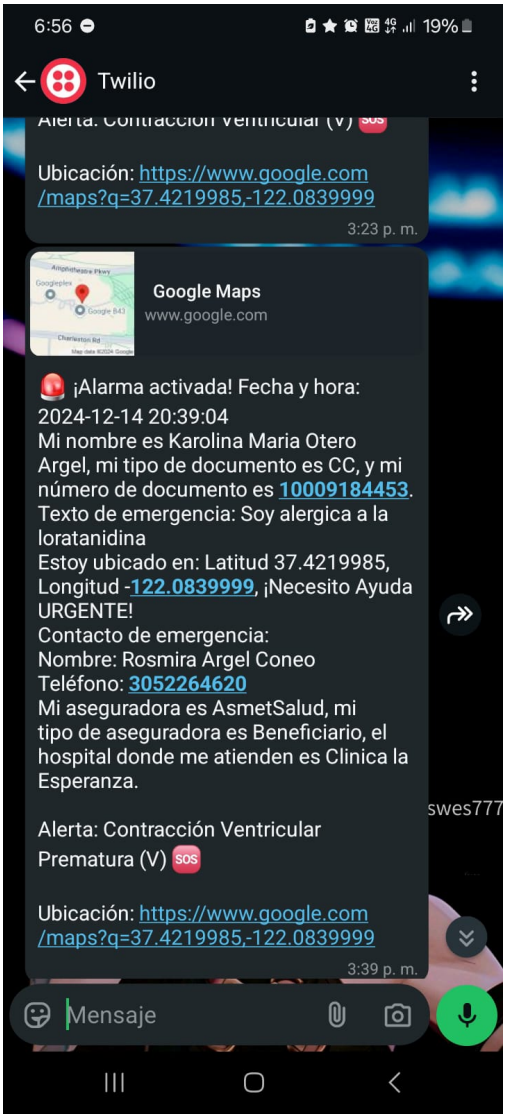


Figura 7.31: Mensaje de alerta

Capítulo 8

Conclusiones

Este proyecto ha abordado una de las problemáticas más comunes en el ámbito de la salud cardiovascular: la detección temprana de arritmias cardíacas asociadas a la muerte súbita cardíaca (MSC). A través de la integración de técnicas avanzadas de procesamiento de señales electrocardiográficas (ECG) y modelos de aprendizaje automático, se logró desarrollar un sistema eficiente y preciso, capaz de identificar patrones arrítmicos con una exactitud del 96 %. Este resultado no solo valida el funcionamiento del modelo propuesto, sino que también resalta su potencial para ser implementado como una herramienta práctica en entornos clínicos y comunitarios.

Asimismo, el desarrollo del aplicativo móvil KIBO representa un avance en la traducción de soluciones tecnológicas al cuidado de la salud. Su capacidad para analizar las señales eléctricas del corazón, detectar arritmias potencialmente peligrosas y activar oportunamente una ruta de atención en emergencias, posiciona esta herramienta como un recurso valioso en la prevención de eventos fatales relacionados con la MSC. Estos logros confirman la viabilidad de utilizar la tecnología para mejorar la calidad de vida de los pacientes con antecedentes cardiovasculares o enfermedades crónicas, destacando el impacto positivo que puede tener este tipo de aplicaciones en la salud pública.

De este trabajo, también se puede destacar, la integración de áreas de la ingeniería que consiguen resultados funcionales, como lo son, la investigación aplicada a la salud, el procesamiento digital de señales, la programación y desarrollo de aplicativos móviles y la experiencia de usuarios, saberes que al combinarse permitieron la creación de una herramienta que proporciona una alternativa para hacerle frente a problemáticas globales como la prevención y/o atención oportuna de muerte súbita cardíaca.

Por otro lado, la metodología implementada en este proyecto permitió simplificar procesos complejos en términos de preprocesamiento de señales ECG. Aunque el diseño original no estaba específicamente orientado a la automatización, el enfoque seguido facilitó la integración de etapas importantes como la segmentación de señales, la extracción de características y el entrenamiento de modelos de aprendizaje automático. Esto dio lugar a una librería propia capaz de ejecutar estas tareas de manera automatizada, generando matrices de confusión, resúmenes de métricas y otros parámetros que optimizan el análisis y la validación de modelos. Además, el uso de conjuntos de datos libres potenció el desarrollo y aseguraron la replicabilidad de los resultados, contribuyendo a que este trabajo no solo sea innovador, sino también accesible para futuras investigaciones.

Capítulo 9

Trabajos futuros

Esta propuesta ha logrado varios avances en la clasificación de señales ECG y la identificación de arritmias por medio de un aplicativo móvil, sin embargo, ciertas limitaciones detectadas en el desarrollo abren la puerta a nuevas líneas de trabajo y mejora. A continuación, se proponen trabajos futuros para abordar estas limitaciones y extender el alcance del proyecto.

9.1. Conjunto de datos y modelo de aprendizaje automático

9.1.1. Ampliar base de datos:

La base de datos actual presenta un desequilibrio significativo en la cantidad de muestras disponibles para ciertas etiquetas, como V=7855, ! = 154, F=1135, y f=1335, en contraste con N=59342, lo que limita la confiabilidad del modelo. Se propone desarrollar un protocolo para recolectar una base de datos más equilibrada y representativa de las arritmias estudiadas, complementando esta estrategia con la generación de datos sintéticos o simulaciones fisiológicas realistas.

9.1.2. Aumento de datos.

Actualmente se emplea una técnica de aumento de datos basada en distribución normal aleatoria ($\mu = 0, \sigma = 0{,}01$). Sin embargo, no está claro si esta es la mejor opción para preservar las características fisiológicas de las señales. Se puede explorar y validar técnicas avanzadas de aumento de datos, como deformaciones no lineales, interpolaciones de señal basadas en aprendizaje automático, y síntesis de características específicas de ECG.

9.1.3. Uso de etiquetas y escalabilidad.

A partir de los resultados se puede considerar ampliar el estudio a etiquetas relacionadas con otras patologías de interés, aprovechando las técnicas de caracterización y modelos ya establecidos. Diseñando un marco de trabajo para escalar este sistema a bases de datos más grandes y heterogéneas, evaluando su rendimiento en escenarios clínicos reales.

9.2. Aplicativo móvil

9.2.1. Diseño UI/UX.

Para este proyecto, se lograron los flujos necesarios para cumplir con los requerimientos funcionales establecidos, asegurando una experiencia de usuario coherente y eficiente. Sin embargo, se identificaron varias interacciones que no están contempladas en la versión actual de la aplicación. En la versión final, se planea incluir dos páginas adicionales: una sección de *Noticias*, que permitirá a los usuarios mantenerse informados sobre actualizaciones relevantes, y una funcionalidad para *guardar reportes clínicos*, brindando la capacidad de almacenar y consultar informes médicos previos.

Estas características pueden ser desarrolladas para mejorar la experiencia del usuario, proporcionando más funcionalidades y un acceso más completo a la información relevante. Además, se explorarán posibles mejoras en la interacción de la interfaz, incluyendo optimizaciones para dispositivos con diferentes tamaños de pantalla y la integración de nuevas herramientas que faciliten el acceso y la navegación dentro de la aplicación.

9.2.2. Compatibilidades.

En la versión actual de este proyecto, la aplicación está diseñada para conectarse exclusivamente a dispositivos que utilicen la tecnología Bluetooth básica. Sin embargo, para futuras versiones se puede ampliar la compatibilidad, permitiendo la conexión con dispositivos que utilicen cualquier versión de Bluetooth, así como con aquellos que se conecten mediante tecnología Wi-Fi, lo que ampliará considerablemente el rango y versatilidad de los dispositivos compatibles.

Por otro lado, la compatibilidad con dispositivos iOS aún no está implementada en esta versión. Para la versión futura, se puede incluir soporte completo para dispositivos iOS, lo que permitirá a un mayor número de usuarios utilizar la aplicación en una amplia variedad de dispositivos Apple, mejorando la accesibilidad y el alcance del proyecto.

9.2.3. Rest API.

La API actual de la aplicación KIBO tiene una arquitectura sólida y cubre todos los requerimientos funcionales. Sin embargo, se pueden incluir funciones adicionales, como la capacidad de borrar y editar información, mejorando la flexibilidad en la gestión de los datos.

9.2.4. Mensajes de alerta.

En este proyecto se utilizó la herramienta Twilio para el envío de alertas. Sin embargo, no se exploraron otras opciones disponibles en el mercado, y la implementación actual solo es adecuada para demostraciones, sin ser escalable para pruebas no controladas. A futuro, se puede explorar el uso de chatbots con inteligencia artificial que gestionen a los usuarios, funcionando como asistentes virtuales para monitorear, alertar y gestionar documentos, medicamentos, signos vitales, entre otros.

9.2.5. Herramienta de gestión de pacientes

Se planea desarrollar una versión de KIBO como herramienta central de información para médicos, centralizando todos los datos de los pacientes en un solo dashboard. Esto permitirá a los médicos acceder a la información en tiempo real. Además, considerando las limitaciones en la expansión de la base de datos, la base de datos almacenaría continuamente las señales ECG de los pacientes esto permite a los especialistas etiquetar correctamente los datos para que puedan ser utilizados en el entrenamiento de modelos futuros.

Capítulo 10

Anexos

1. **Guía de códigos del desarrollo de modelos de Aprendizaje Automático, aplicación móvil, Rest API y diseño de IU/UX en Figma.**
2. **Manual de uso del aplicativo KIBO.**
3. **Consentimiento informado política de tratamiendo de datos de aplicaciones futuras KIBO.**

Anexo 1 – Guía de códigos del desarrollo de modelos de Aprendizaje Automático, aplicación móvil, Rest API y diseño de IU/UX en Figma.

Titulo	**Tipo**	**enlace**
Desarrollo y proceso de segmentación, caracterización y entrenamiento del modelo de IA para la identificación de arritmias cardiacas	Archivo .zip (OneDrive)	ProyectoGrado
Diseño UI/UX en figma	Figma	PrototipoKibo - ver
Desarrollo de la aplicación móvil KIBO	Repositorio GitHub	Kibo
Desarrollo de la Rest API de KIBO	Repositorio GitHub	ApiKibo

Cuadro 10.1: Enlaces a la documentación del desarrollo del proyecto

Anexo 2 – Manual de uso del aplicativo KIBO

KIBO

MANUAL DE USO KIBO APP

Aplicación móvil para la alerta de arritmias potencialmente mortales relacionadas con la posibilidad Muerte Súbita Cardíaca.

Pontificia Universidad
JAVERIANA
Cali

Autores del documento

Karolina M. Otero Argel & Jereminth Muñoz de la Torre.
Estudiantes de Ingeniería Biomédica.
Pontificia Universidad Javeriana Cali, Colombia.

Versión del documento

1.0
Fecha: 04-01-2025

KIBO APP

KIBO APP es una herramienta diseñada, por estudiantes de ingeniería biomédica, que identifica patrones arrítmicos asociados con el riesgo de Muerte Súbita Cardíaca (MSC) y activa rutas de atención inmediata. La aplicación combina conocimientos de ingeniería, informática y medicina para analizar señales cardíacas, emitir alertas oportunas, conectar a los usuarios con servicios de emergencia y reducir complicaciones graves.

Las enfermedades cardiovasculares (ECV) son una de las principales causas de mortalidad global, y las arritmias cardíacas, un factor de riesgo significativo, afectan a miles de personas cada año. La identificación temprana de estas condiciones es esencial para prevenir eventos críticos como la MSC y minimizar el impacto en los pacientes y los sistemas de salud.

Este manual ofrece una guía completa para el uso de KIBO APP, detallando cómo registrarse, ajustar configuraciones, interpretar alertas y acceder a rutas de atención en emergencias cardiovasculares. Su propósito es garantizar un uso eficaz de la aplicación, mejorando la prevención y respuesta ante estas condiciones.

CONTENIDO

1 ¿Cómo acceder a KIBO?

Para acceder a la aplicación, se debe descargar el archivo vector_normal.apk desde un dispositivo móvil con sistema operativo Android. Es importante tener en cuenta que, por el momento, la aplicación no está disponible para iOS ni para otros sistemas operativos. Al abrir el enlace e instalar el archivo, el aplicativo se instalará automáticamente en el dispositivo.

Nota: Esta versión de la aplicación es una demostración diseñada exclusivamente con fines académicos. No se encuentra disponible en tiendas de aplicaciones y no está destinada al uso general. Su implementación está limitada a condiciones controladas para pruebas de funcionamiento relacionadas con el proyecto de grado titulado: *"Desarrollo de un aplicativo móvil para la identificación de arritmias cardíacas mediante procesamiento digital de señales ECG y aprendizaje automático".*

Este proyecto se desarrolla como parte de los requisitos para optar al título de Ingenieros Biomédicos de los autores de este manual.

2 Esquema de navegabilidad

La Fig 1. presenta el esquema de navegación de **KIBO APP**, detallando cada pantalla y el flujo de interacción entre ellas, la instalación de la app, el inicio de sesión y/o registro; la pantalla principal y conexión a BT, el acceso a perfil e información personal y la navegación por las pantallas home, predicción y configuraciones de alarmas y etiquetas. Este esquema guía al usuario a través de las diferentes funciones y ventanas de la aplicación, facilitando una comprensión visual del recorrido que puede realizar para acceder a cada funcionalidad de KIBO APP.

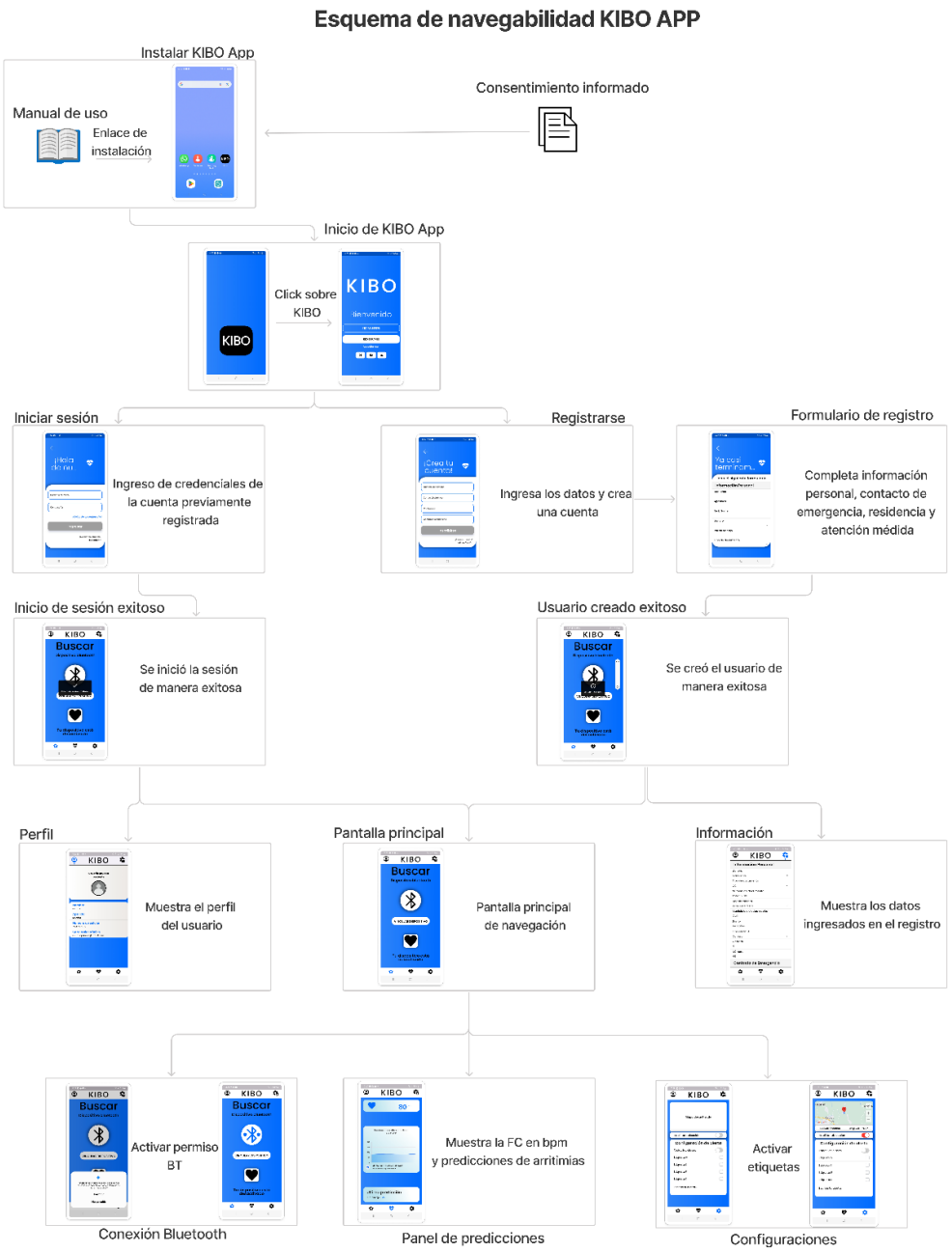


Fig 1. Esquema de Navegabilidad KIBO APP

3 Uso del aplicativo

En esta sección, se detallan los pasos necesarios para instalar, configurar y utilizar **KIBO APP** de manera efectiva. Se abordan las instrucciones de instalación inicial, el proceso de registro e inicio de sesión, así como una guía práctica sobre las principales funciones de la interfaz. Estas funciones incluyen la conexión a dispositivos mediante Bluetooth, la visualización del panel de predicciones, la configuración de permisos y etiquetas, la gestión del perfil de usuario y la edición de los datos registrados. Este apartado está diseñado para garantizar una experiencia óptima y facilitar la interacción con el aplicativo durante las pruebas del proyecto.

3.1. Instalación

Una vez descargado el archivo vector_normal.apk, asegúrate de contar con los permisos necesarios en tu dispositivo Android para instalar aplicaciones desde fuentes externas. Para habilitar esta opción, ve a **Configuración > Seguridad > Fuentes desconocidas** y actívala, si aún no lo has hecho.

Sigue estos pasos para completar la instalación:

1. Localiza el archivo **KIBO.apk** en la carpeta de descargas de tu dispositivo.
2. Toca el archivo para iniciar el proceso de instalación.
3. En la pantalla emergente, selecciona **Instalar** y espera a que el proceso se complete.
4. Una vez instalada, encontrarás el icono de KIBO en el menú principal de tu dispositivo (Fig 2).

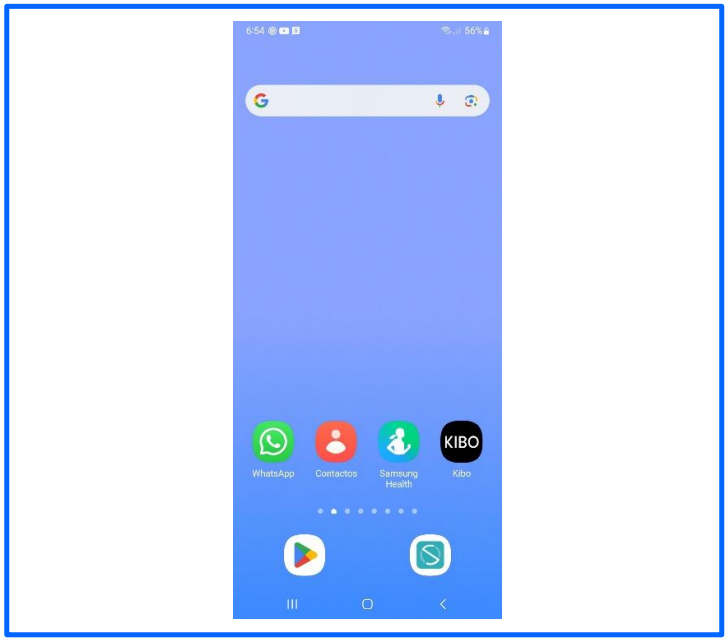


Fig 2. App KIBO instalada en dispositivo Android.

Ahora estás listo para explorar las funciones del aplicativo y realizar las pruebas necesarias dentro del marco del proyecto.

3.1. Registro e inicio de sesión

Al abrir la aplicación por primera vez, se encuentra la pantalla de bienvenida, diseñada para ofrecer acceso rápido a las principales opciones: **Iniciar Sesión**, para usuarios con una cuenta registrada, y **Registrarse**, para quienes necesitan crear una nueva cuenta.

Pantalla de Bienvenida

Fig 3. Pantallas de inicio KIBO APP

En esta pantalla inicial (Fig. 3), el usuario tiene dos alternativas:

- **Iniciar Sesión**: Dirigida a quienes ya cuentan con credenciales previamente creadas.
- **Registrarse**: Diseñada para permitir el registro de nuevos usuarios.

Iniciar Sesión

Los usuarios registrados acceden a su cuenta siguiendo estos pasos (Fig 4):

1. Seleccionan la opción **Iniciar Sesión** en la pantalla de bienvenida.
2. Ingresan su correo electrónico o número de teléfono en el campo correspondiente.
3. Introducen su contraseña en el segundo campo.
4. Presionan el botón **Ingresar** para acceder a la aplicación.
5. Ingresan a la aplicación y reciben un mensaje de aviso de inicio de sesión exitoso.

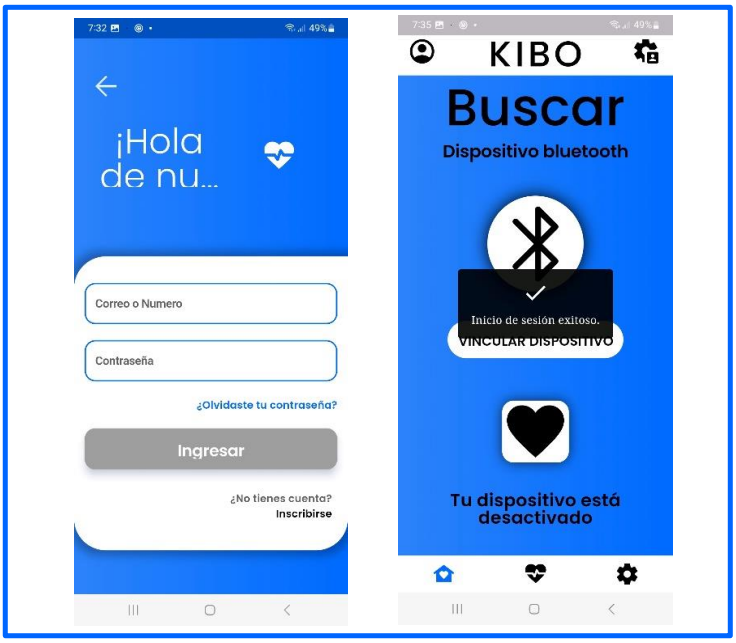


Fig 4. Iniciar sesión KIBO

6. De no existir un registro previo o de ser incorrectos los datos diligenciados aparecerá mensaje de error (Fig 5).

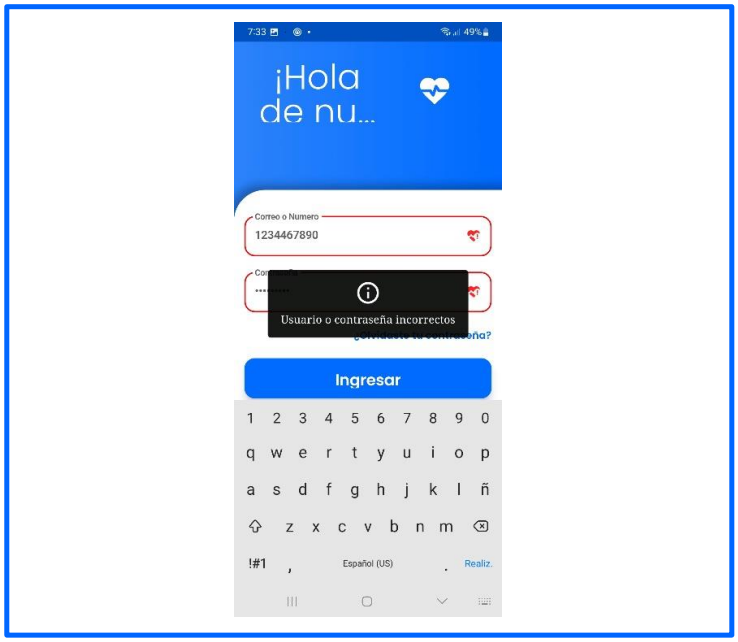


Fig 5. Usuario o contraseña incorrecta al iniciar sesión

En caso de olvidar la contraseña, pueden utilizar el enlace de recuperación disponible en esta misma pantalla para restablecerla de forma segura.

Registrarse

Para los nuevos usuarios, el registro se realiza de la siguiente manera (Fig 6):

1. Seleccionan la opción **Registrarse** en la pantalla de bienvenida.
2. Completan los datos de registro ingresando su número de teléfono, correo electrónico y una contraseña segura.
3. Presionan el botón **Inscribirse** para enviar la información.
4. Se completa el **Formulario** que requiere información personal, de contacto de emergencia e información médica necesaria.
5. Ingresan a la aplicación y reciben un mensaje de aviso de creación de usuario exitoso.

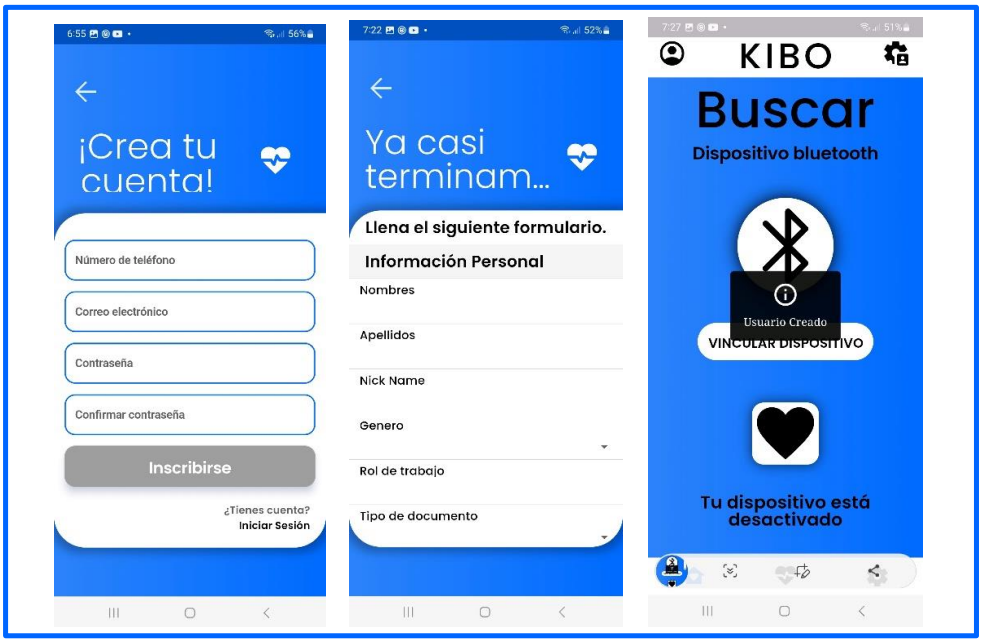


Fig 6. Registrarse como nuevo usuario KIBO

Con estos pasos, los usuarios estarán listos para comenzar a explorar las funcionalidades de **KIBO APP**.

3.3. Principales funciones de la interfaz

KIBO APP está diseñada para proporcionar una experiencia intuitiva y funcional, facilitando a los usuarios la conexión con los dispositivos necesarios, la visualización de predicciones y la gestión de información de sus datos registrados. A continuación, se describen las principales funciones de la interfaz y cómo utilizarlas.

3.3.1 Conexión a Bluetooth

Esta función permite que la aplicación se comunique con los dispositivos necesarios para la adquisición de datos (Fig 7):

1. Acceda al botón de **Dispositivo Bluetooth** desde la interfaz principal.
2. Active el Bluetooth en su dispositivo móvil si no está habilitado.
3. Seleccione el dispositivo compatible en la lista de dispositivos disponibles.
4. Espere la confirmación de conexión exitosa antes de proceder.

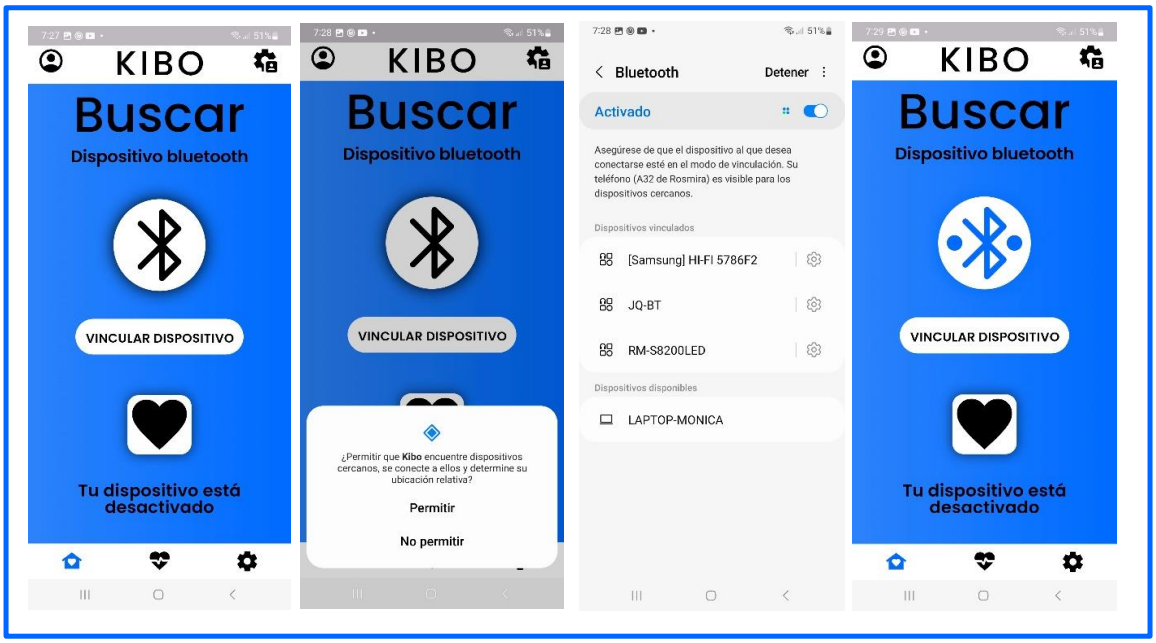


Fig 7. Conectar Bluetooth en pantalla principal de KIBO

Nota: Asegúrese de que el dispositivo externo esté encendido y dentro del rango de conexión.

3.3.2 Panel de predicciones

El panel de predicciones es el corazón de **KIBO APP**, donde se visualizan los resultados obtenidos mediante el análisis de las señales ECG:

1. Al iniciar la actividad, el panel mostrará un gráfico en tiempo real de la frecuencia cardíaca en bpm (latidos por minuto) y los resultados de las predicciones realizadas (Fig 8).

2. Cada predicción estará acompañada de etiquetas y mensajes de alerta internos y al contacto de emergencia para activar la ruta de atención en el menor tiempo posible.

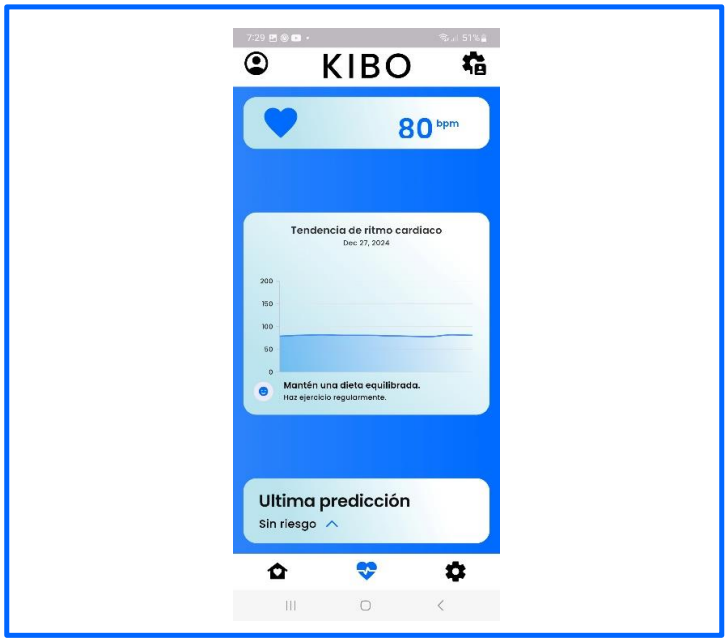


Fig 8. Panel de predicciones y tendencias

3.3.3 Configuración de permisos y etiquetas

Esta sección permite personalizar las opciones de acceso y las etiquetas asociadas a las predicciones:

1. Acceda al menú **Configuración** desde el perfil o el panel principal.
2. Configure los permisos necesarios para garantizar el correcto funcionamiento de la aplicación, como el acceso a la geolocalización en tiempo real (Fig 9).

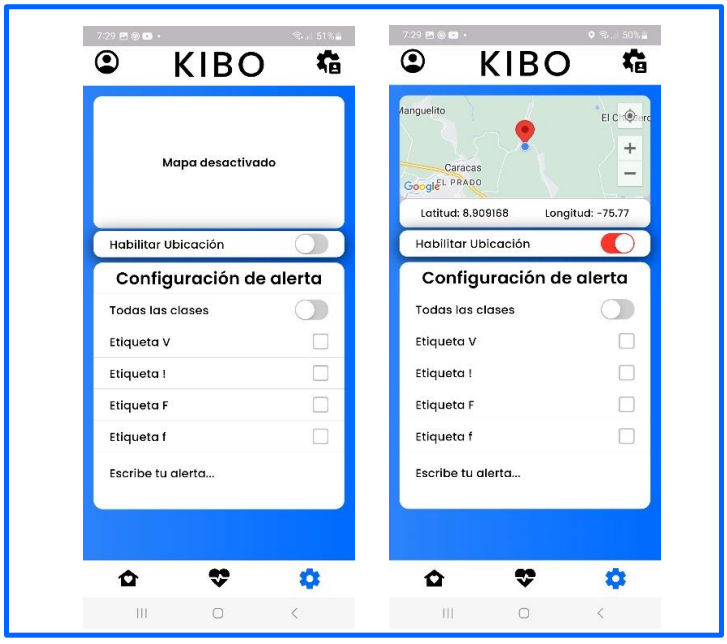


Fig 9. Pantalla de configuración de alertas y geolocalización

3. Personalice las etiquetas de alertas para adaptarlas a las necesidades específicas que se requieran e incluso agregue notas importantes sobre usted en casos de emergencias, como alergias o condiciones específicas de salud (Fig 10).

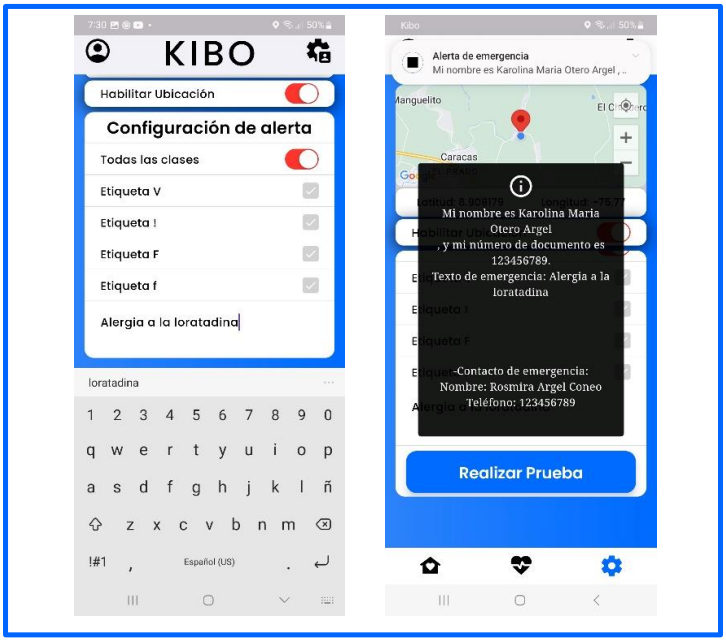


Fig 10. Función para escribir mensajes importantes de seguridad y prueba de mensaje de emergencia

3.3.4 Perfil

La sección de perfil permite al usuario gestionar su información personal:

1. Acceda al menú **Perfil** desde la barra de navegación principal.
2. Revise y actualice información básica, como nombre y dirección de correo electrónico, foto (Fig 11).

Fig 11. Perfil de usuario KIBO

3.3.5 Información

Esta función está diseñada para que el usuario pueda administrar los datos relacionados con su cuenta y las actividades realizadas:

1. Acceda a **Datos de Registro** desde el menú principal.
2. Edite campos específicos, como información personal, de emergencias y médica (Fig 12).

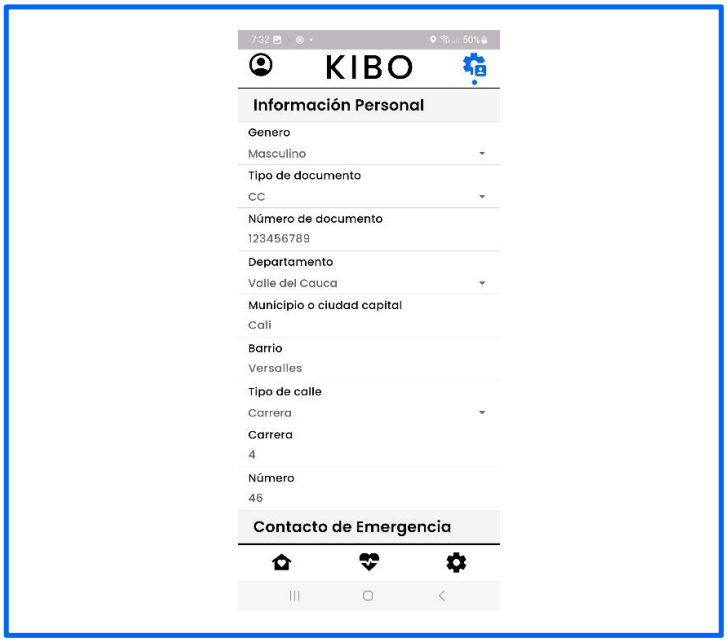


Fig 12. Panel de Información de datos de registro

Anexo 3 – Consentimiento informado política de tratamiento de datos de aplicaciones futuras KIBO

POLÍTICA DE TRATAMIENTO DE DATOS PERSONALES PARA KIBO

Responsable del tratamiento:

KIBO, con domicilio en XXX, Santiago de Cali, Colombia, número de contacto 12345678, y correo electrónico ejemplocorreo@kibo.com.co, es el responsable del tratamiento de los datos personales conforme a la Ley 1581 de 2012 y sus decretos reglamentarios.

Datos recolectados:

Para ofrecer los servicios de la aplicación KIBO de manera óptima, se requerirán los siguientes datos personales:

1. Datos de identificación: nombre, número de identificación, correo electrónico, celular.
2. Datos sensibles: historia clínica, diagnóstico y antecedentes médicos.
3. Ubicación en tiempo real: activación del GPS para geolocalización, exclusivamente en casos de emergencia o asistencia médica.

Finalidad del tratamiento de datos:

El tratamiento de los datos personales tendrá las siguientes finalidades:

1. Prestación de servicios de salud y asistencia médica: acceso a la historia clínica y ubicación para garantizar una atención prioritaria y oportuna.
2. Servicio de ubicación en tiempo real: uso de datos de geolocalización en situaciones de emergencia para facilitar traslados a centros de salud cercanos.
3. Mejoramiento de servicios: análisis de datos para optimizar la plataforma y mejorar la experiencia del usuario.

Consentimiento expreso:

El tratamiento de los datos personales se realiza con el consentimiento expreso, libre e informado del usuario, otorgado al aceptar esta política al momento de registrarse en la aplicación KIBO. Los datos sensibles, como información médica y de ubicación, se tratarán únicamente con la autorización explícita del usuario, obtenida a través de un campo específico en la aplicación.

Los usuarios pueden revocar su consentimiento en cualquier momento enviando una solicitud al correo electrónico ejemplocorreo@kibo.com.co.

Derechos de los titulares:

Conforme a la Ley 1581 de 2012, los usuarios tienen derecho a:

1. Conocer y acceder a sus datos personales.
2. Rectificar la información incompleta, inexacta o desactualizada.
3. Solicitar la eliminación de sus datos personales, salvo que exista una obligación legal para su conservación.
4. Revocar la autorización para el tratamiento de datos personales en cualquier momento.

Seguridad y confidencialidad:

KIBO adopta medidas técnicas, administrativas y físicas para garantizar la protección de los datos personales, incluyendo:

1. Cifrado de datos sensibles y de ubicación para prevenir accesos no autorizados.
2. Acceso restringido y controlado a la información, limitado al personal autorizado.
3. Monitoreo y auditorías periódicas para verificar el cumplimiento de los estándares de seguridad.

Procedimiento para ejercer derechos:

Los usuarios pueden presentar solicitudes, quejas o consultas relacionadas con el tratamiento de sus datos personales a través de:
Correo electrónico: ejemplocorreo@kibo.com.co
Línea telefónica: 12345678
KIBO se compromete a responder estas solicitudes en un plazo máximo de 15 días hábiles, conforme a la normativa aplicable.

Conservación de los datos:

Los datos personales serán conservados únicamente durante el tiempo necesario para cumplir con las finalidades descritas en esta política y de acuerdo con la normativa vigente.
Los datos sensibles relacionados con la historia clínica serán conservados por un período de 10 años desde la última atención, conforme a la Ley 23 de 1981, artículo 34.
Una vez cumplido el propósito para el cual se recolectaron otros datos personales, estos serán tratados de la siguiente manera:

1. Datos anonimizados: algunos datos podrán ser anonimizados para fines estadísticos o de investigación, asegurando que no se pueda identificar al usuario.
2. Eliminación automática: los datos serán eliminados de nuestras bases mediante procedimientos seguros que garanticen su destrucción completa.
3. Procedimiento estándar: se seguirán las mejores prácticas de la industria para la eliminación o anonimización de datos, respetando los estándares legales aplicables.

Geolocalización en tiempo real:

El uso de la ubicación en tiempo real está estrictamente limitado a:

1. Emergencias médicas: cuando sea necesario para localizar al usuario y coordinar su traslado al centro de salud más cercano.
2. Asistencia en salud: para proporcionar información relevante sobre el estado actual del usuario.
3. El sistema de geolocalización se activará únicamente con el consentimiento explícito del usuario y no se empleará para otros fines no autorizados.

Actualización de la política:

KIBO revisará esta política periódicamente para asegurar su cumplimiento con los cambios legales y tecnológicos. En caso de realizarse modificaciones significativas, se notificará a los usuarios con al menos 30 días de antelación, a través de la aplicación y del correo electrónico registrado.

Bibliografía

[1] Laboratorios Clínicos Histocytolab, "Ayudas Cardiodiagnósticas," Disponible en: https://lch.co/ayudas-cardiodiagnosticas/, consultado el: fecha de acceso.

[2] S. W. Smith, *Digital Signal Processing*, 2nd ed. USA: California Technical Publishing, 1999.

[3] S. D. y. B. N. S. Mendis, "Organizational update," vol. 46, no. 5, 2015. [Online]. Available: https://doi.org/10.1161/strokeaha.115.008097

[4] OPS, "La carga de las enfermedades cardiovasculares en la región de las américas, 2000-2019," *Portal de Datos de NMH. Organización Panamericana de la Salud*, 2021.

[5] H. Rodríguez-Reyes, M. M. Gutiérrez, M. F. Márquez, G. P. Garza, E. A. Lafuente, F. O. Galván, S. L. Vaca, and V. A. M. Montero, "Muerte súbita cardiaca. estratificación de riesgo, prevención y tratamiento," *Archivos de Cardiología de México*, vol. 85, no. 4, pp. 329–336, 2015.

[6] F. Mendoza, M. Romero, J. Lancheros, P. A. Alfonso, and L. Huérfano, "Carga económica de la fibrilación auricular en colombia," vol. 27, no. 6, p. 538—544, 2020. [Online]. Available: http://dx.doi.org/10.1016/j.rccar.2019.09.012

[7] DANE, "Principales resultados de estadísticas vitales nacimientos y defunciones para el cuarto trimestre de 2022pr, acumulado del año 2022pr y el año corrido de 2023pr," DANE, Tech. Rep., 2022.

[8] C. de Asmundis and P. Brugada, "Epidemiología de la muerte súbita cardiaca," *Revista Española de Cardiología Suplementos*, vol. 13, pp. 2–6, 2013, nuevos avances en la identificación de pacientes con riesgo de muerte súbita. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1131358713700608

[9] M. Clinic. (2023) Arritmia cardíaca - síntomas y causas - mayo clinic. [Online]. Available: https://www.mayoclinic.org/es/diseases-conditions/heart-arrhythmia/symptoms-causes/syc-20350668

[10] J. Vanegas, "Calidad de vida relacionada con la salud en pacientes con arritmia ventricular una medicion con el asta, en dos momentos," Ph.D. dissertation, Universidad Nacional de Colombia, 2019. [Online]. Available: https://repositorio.unal.edu.co/handle/unal/76220

[11] F. E. del Corazón. Muerte súbita. [Online]. Available: https://fundaciondelcorazon.com/informacion-para-pacientes/enfermedades-cardiovasculares/muerte-subita.html

[12] Ángel A. García-Peña, D. A. Ospina-Buitrago, J. del P. Rico-Mendoza, D. G. Fernández-Ávila, Óscar M. Muñoz-Velandia1, and F. Suárez, "Prevalencia de fibrilación auricular en colombia según información del sistema integral de información de la protección social (sispro)," *Revista Colombiana de Cardiología*, vol. 29, no. 2, 2022.

[13] G. L. Bisellach, "Aprendizaje automático para la clasificación de arritmias cardíacas," Ph.D. dissertation, University of the Balearic Islands, 2018. [Online]. Available: https://dspace.uib.es/xmlui/bitstream/handle/11201/147227/Llodra_Guillem.pdf

[14] R. Álvarez Patiño and A. R. Luna, "Diseño de una herramienta para la detección de arritmias cardíacas en electrocardiogramas utilizando técnicas de aprendizaje automático," Ph.D. dissertation, Universidad de Antioquia, 2023. [Online]. Available: https://bibliotecadigital.udea.edu.co/handle/10495/35511

[15] L. M. C. Quiroz, "Aplicación para la detección de arritmias basado en los algoritmos de pan tompkins, elgendi y boonperm," Ph.D. dissertation, Universidad de Lima, 2021. [Online]. Available: https://hdl.handle.net/20.500.12724/14094

[16] C. C. G. Higuera, "Comparación de modelos de inteligencia artificial para el diagnóstico de arritmia cardiaca mediante información de electrocardiogramas," Ph.D. dissertation, Universidad Autónoma De Bucaramanga, 2021. [Online]. Available: https://repository.unab.edu.co/bitstream/handle/20.500.12749/16075/2021_Tesis_Cristian_Camilo_Gelves.pdf?sequence=1&isAllowed=y

[17] A. E. V. Toaquiza and A. A. T. Aguilar, "Arritmias ventriculares y su relación con la muerte súbita cardiaca en deportistas," Ph.D. dissertation, Universidad Técnica de Ambato, 2021. [Online]. Available: https://doi.org/10.37811/cl_rcm.v6i5.3313

[18] Healthline. Sistema circulatorio: Funciones, órganos y enfermedades. [Online]. Available: https://www.healthline.com/health/es/sistema-circulatorio

[19] J. M. Nerbonne and R. S. Kass, *Molecular Physiology of Cardiac Repolarization.* Physiological Reviews, 2005, vol. 85, no. 4. [Online]. Available: https://doi.org/10.1152/physrev.00002.2005

[20] M. Gavaghan, "Cardiac anatomy and physiology: A review," *AORN Journal*, vol. 67, no. 4, 1998.

[21] O. P. D. L. SALUD, *Clasificación Estadística Internacional de Enfermedades y Problemas Relacionados con la Salud.* Ginebra, 2008, vol. 1, no. 10. [Online]. Available: https://ais.paho.org/classifications/chapters/pdf/volume1.pdf

[22] T. S. of Thoracic Surgeons. Trastornos del ritmo cardíaco - la guía del paciente para corazón, los pulmones y la cirugía esofágica. [Online]. Available: https://ctsurgerypatients.org/es/enfermedades-del-coraz%C3%B3n-en-adultos/trastornos-del-ritmo-card%C3%ADaco

[23] M. M. G. M. F. M. G. P. G. E. A. L. F. O. G. S. L. V. V. A. M. M. Humberto Rodríguez-Reyesa, "Muerte súbita cardiaca. estratificación de riesgo, prevención y tratamiento," *ElSevier*, vol. 85, no. 4, 2015.

[24] M. Clinic. Enfermedad de las arterias coronarias. [Online]. Available: https://www.mayoclinic.org/es/diseases-conditions/coronary-artery-disease/diagnosis-treatment/drc-20350619

[25] L. J. Iglesias, "Diseño de un rele neuronal de protección para lineas aéreas de at con preprocesado de señal mediante la transformada wavelet," *doctoral thesis*, 2004. [Online]. Available: http://hdl.handle.net/10803/6289

[26] S. F. A. P. y C. M. León Mendoza, "Repositorio universidad de guayaquil: Optimización de algoritmos de machine learning aplicados a problemas de arritmias cardíacas." repositorio universidad de guayaquil: Página de inicio." 2023. [Online]. Available: http://repositorio.ug.edu.ec/handle/redug/59960

[27] . M. V. Raschka, S., "Python machine learning: Machine learning and deep learning with python, scikit-learn, and tensorflow," *2nd Edition*, (2017).

[28] E. D. A. J. H. F. R. y. M. v. d. S. A. M. Alaa, T. Bolton, "Cardiovascular disease risk prediction using automated machine learning: A prospective study of 423,604 uk biobank participants," vol. 14, no. 5, 2019. [Online]. Available: https://doi.org/10.1371/journal.pone.0213653

[29] J. L. Sarmiento-Ramos, "Aplicaciones de las redes neuronales y el deep learning a la ingeniería biomédica," vol. 19, no. 4, 2020. [Online]. Available: https://doi.org/10.18273/revuin.v19n4-2020001

[30] D. F. . B. Tuay and G. A. . P. González, "Desarrollo de un sistema para clasificación de patologías cardiovasculares en señales electrocardiográficas (ecg) aplicando inteligencia artificial y cloud computing," vol. 2, no. 1, 2021. [Online]. Available: 10.22490/ECBTI.4812

[31] S. GAMCO. (2021) Lstm: Long short-term memory. [Online]. Available: https://gamco.es/glosario/lstm-long-short-term-memory/

[32] A. W. Service. (2024) ¿qué es el aprendizaje por transferencia? [Online]. Available: https://aws.amazon.com/es/what-is/transfer-learning/#:~:text=El%20aprendizaje%20por%20transferencia%20(TL,para%20una%20nueva%20tarea%20relacionada.

[33] Mathworks. (2024) mobilenetv2. [Online]. Available: https://la.mathworks.com/help/deeplearning/ref/mobilenetv2.html#mw_609c1852-ea25-4857-9b3d-cd7d1916d5ff_sep_mw_6dc28e13-2f10-44a4-9632-9b8d43b376fe

[34] geeks forgeeks. (2024) Vgg-16 | cnn model. [Online]. Available: https://www.geeksforgeeks.org/vgg-16-cnn-model/

[35] Python. (2024) python about. [Online]. Available: https://www.python.org/

[36] Numpy. (2024) numpy about. [Online]. Available: https://numpy.org/

[37] Pandas. (2024) pandas about. [Online]. Available: https://pandas.pydata.org/

[38] Matplotlib. (2024) matplotlib about. [Online]. Available: https://matplotlib.org/

[39] TensorFlow. (2024) tensorflow about. [Online]. Available: https://www.tensorflow.org/

[40] Flask. (2024) flask about. [Online]. Available: https://flask.palletsprojects.com/

[41] Virtualenv. (2024) Virtualenv latest. [Online]. Available: https://virtualenv.pypa.io/en/latest/

[42] Git. (2024) Git about. [Online]. Available: https://git-scm.com/about/branching-and-merging

[43] GitHub. (2024) Github about. [Online]. Available: https://docs.github.com/es/get-started/start-your-journey/about-github-and-git

[44] V. S. Code. (2024) Visual studio code about. [Online]. Available: https://code.visualstudio.com/docs

[45] A. Developers. (2024) Guías para desarrolladores. [Online]. Available: https://developer.android.com/guide?hl=es-419

[46] M. O. Q. K. . S. E. Ansari, Y., "Deep learning for ECG Arrhythmia detection and classification: an overview of progress for period 2017–2023," *Frontiers in Physiology*, vol. 1, no. 1, 2023.

[47] Q. X. K. L. S. A. M. I. A. L. bin Md Pauzi; Qiuxia Zhang; Poh Ying Lim, "Deep learning-based ecg arrhythmia classification: A systematic review," *Applied Sciences*, 2023.

[48] PhysioNet's. (2005) Mit-bih arrhythmia database. [Online]. Available: https://www.physionet.org/content/mitdb/1.0.0/

[49] M. K. S. F. M. Sarrafzadeh, "Ecg heartbeat classification: A deep transferable representation," *IEEE Explore*, 2018.

[50] PyWavelets. (2024) pywavelets about. [Online]. Available: https://pywavelets.readthedocs.io/en/latest/index.html

[51] T. P. V. S. J. M. S. D. O. S. L. J. E. W. K. E. J. C. U. R. Acharya, "A novel machine learning framework for automated detection of arrhythmias in ecg segments," *Journal of Ambient Intelligence and Humanized Computing*, 2021.

[52] L. M. S. N. L. S. M. Vítek, "Brno university of technology ecg signal database with annotations of p wave (but pdb)," *Research Square*, 2021.