

UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE INFORMÁTICA
CONCEPCIÓN - CHILE



“APLICACIÓN MÓVIL PARA LECTURA DE PROTOCOLOS DE MANTENCIÓN DE EQUIPOS MÉDICOS MEDIANTE TECNOLOGÍAS DE TEXTO A VOZ”

CRISTÓBAL ANDRÉS CARRASCO VERGARA

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO EN INFORMÁTICA

Profesor Guía: Rhoddy Angel Viveros Muñoz

Octubre - 2024

DEDICATORIA

A mis padres, quienes han sido un pilar fundamental en este proceso, entregándome su amor y apoyo constante durante mi vida y durante mi paso por la universidad. Entregándome palabras de aliento y siendo un punto de contención importante para mantener la motivación y las ganas de continuar con todo este proceso para convertirme en un profesional.

Especialmente a mi querida hermana, quien a lo largo de mi vida se ha preocupado y ha cuidado de mí, me ha enseñado y ha sido mi confidente desde que tengo uso de razón, siendo ella una gran influencia en mi vida y en mi desarrollo tanto personal como académico.

A ustedes, mi familia, los amo con todo mi corazón y alma, y esta dedicatoria es tributo al impacto significativo que ustedes han tenido en mi desarrollo personal y a futuro profesional.

AGRADECIMIENTOS

Quiero expresar mi agradecimiento a mi profesor guía, Rhoddy Viveros, cuya orientación y consejos han guiado este proyecto desde el principio. También agradecer a los profesores que nos entregaron conocimientos, herramientas, potenciaron nuestras habilidades y resaltaron nuestras debilidades con el objetivo de mejorarlas y con eso motivarnos a una mejora continua.

Agradezco también a mis amigos y compañeros, los cuales han sido de gran apoyo al momento de realizar este trabajo y en el transcurso de la carrera, dando palabras de aliento, además de entregar su amistad y colaboración durante nuestro paso por la universidad, volviéndola mucho más enriquecedora y totalmente memorable.

Gracias a mis padres, partiendo por mi madre, la cual fue un punto de apoyo emocional gigantesco, además de preocuparse siempre por mi bienestar, si comía, que durmiera bien y por supuesto de que estudiara. También agradecerle a mi padre, el cual me ha entregado las herramientas necesarias para poder desenvolverme en lo académico y en lo personal, ayudándome a potenciar mis habilidades, además de ser el que me ayudó mucho durante el desarrollo de la idea de este proyecto, además de darme consejos y apoyo de sano a sano.

Gracias también a mi querida hermana, la cual también siempre estuvo preocupada por mí, apoyándome en los estudios en la medida que pudiera, siendo un apoyo emocional enorme y entregándome una alegría tan grande como la es mi sobrino, el cual me llena de energía y felicidad.

Y finalmente agradecer a las personas que fueron parte de este viaje, a mis demás familiares, a los que ya no están, a los que llegaron, todos ellos fueron de alguna otra forma contribuyeron para hacer posible este logro.

RESUMEN

Resumen— Este proyecto se enfoca en el desarrollo de una aplicación móvil innovadora utilizando tecnología de texto a voz, destacando la importancia de la usabilidad y la experiencia del usuario. Se conceptualizó y se diseñó la solución, implementándola con una metodología centrada en las mejores prácticas de desarrollo móvil.

Palabras Clave— Síntesis del habla; Desarrollo Móvil; Android; Experiencia del Usuario; Aplicaciones Móviles

ABSTRACT

Abstract— This project focuses on the development of an innovative mobile application using text-to-speech technology, highlighting the importance of usability and user experience. The solution was conceptualized, designed and implemented using a methodology focused on mobile development best practices.

Keywords— Speech Synthesis; Mobile Development; Android; User Experience; Mobile Applications

GLOSARIO

Aquí se deben colocar las siglas mencionadas en el trabajo y su explicación, por orden alfabético. Por ejemplo:

API (Application Programming Interface - Interfaz de Programación de Aplicaciones): Una API es un conjunto de reglas que permiten que las aplicaciones se comuniquen y compartan datos o funciones de manera estandarizada.

Framework (Marco de trabajo): Conjunto de herramientas y pautas que facilita el desarrollo de software al proporcionar una estructura base y soluciones para problemas comunes.

HMM (Hidden Markov Models - Modelos Ocultos de Markov): Técnica estadística utilizada en el procesamiento de lenguaje natural y reconocimiento de patrones para modelar sistemas no directamente observables.

MVC (Model View Controller - Modelo Vista Controlador): Patrón de diseño que divide una aplicación en Modelo (datos y lógica), Vista (presentación) y Controlador (gestión de interacciones).

MVVM (Model View ViewModel - Modelo Vista VistaModelo): Patrón de diseño basado en MVC reemplazando el Controlador por la VistaModelo, mejorando la separación de preocupaciones.

MVP (Model View Presenter - Modelo Vista Presentador): Patrón de diseño que separa la aplicación en Modelo (datos y lógica), Vista (presentación) y Presentador (control de flujo), mejorando la prueba y mantenibilidad del código en comparación con el MVC tradicional.

PWA (Progressive Web App - Aplicación Web Progresiva): Aplicación web que se comporta como una app móvil, con capacidades offline y notificaciones, accesible desde navegadores.

TTS (TextToSpeech - Texto a Voz): Tecnología que convierte texto escrito en voz hablada, utilizada en aplicaciones de asistentes virtuales y accesibilidad.

UI (User Interface - Interfaz de Usuario): Forma en que los usuarios interactúan con sistemas informáticos, incluyendo elementos visuales y navegación.

UX (User Experience - Experiencia del Usuario): La percepción global del usuario al interactuar con un producto o servicio, incluyendo factores como usabilidad y satisfacción.

ÍNDICE DE CONTENIDOS

RESUMEN	IV
ABSTRACT	IV
GLOSARIO	V
ÍNDICE DE FIGURAS	IX
ÍNDICE DE TABLAS	IX
INTRODUCCIÓN	1
CAPÍTULO 1: DEFINICIÓN DEL PROBLEMA	2
1.1 IDENTIFICACIÓN DEL PROBLEMA	2
1.1.1 CONTEXTO DEL PROBLEMA	2
1.2 USUARIOS AFECTADOS POR EL PROBLEMA	3
1.2.1 ¿QUÉ PASARÍA SI NO SE RESOLVIERA EL PROBLEMA?	3
1.3 MOTIVACIÓN DEL PROYECTO	3
1.4 COMPETENCIA	4
CAPÍTULO 2: MARCO CONCEPTUAL	5
2.1 TECNOLOGÍAS DE TEXTO A VOZ	5
2.1.1 DEFINICIÓN DE TEXT TO SPEECH (TTS)	5
2.1.2 MODELOS DE SÍNTESIS DE VOZ	5
2.2 AVANCES RECIENTES EN SÍNTESIS DE VOZ	7
2.3 APLICACIONES DE SÍNTESIS DE VOZ	7
2.3.1 DESAFÍOS Y FUTURAS DIRECCIONES	8
2.4 DESARROLLO MÓVIL	8
2.4.1 DEFINICIÓN DE APLICACIÓN MÓVIL	8
2.4.2 DEFINICIÓN DE DESARROLLO MÓVIL	9
2.4.3 EVOLUCIÓN DEL DESARROLLO MÓVIL	9
2.5 PLATAFORMAS MÓVILES Y SISTEMAS OPERATIVOS	10
2.5.1 ANDROID	10
2.5.2 IOS	12
2.6 HERRAMIENTAS DE DESARROLLO Y LENGUAJES DE PROGRAMACIÓN	13
2.6.1 ANDROID STUDIO	14
2.6.2 XCODE	15
2.7 DISEÑO DE INTERFAZ DE USUARIO Y EXPERIENCIA DEL USUARIO	16
2.7.1 DEFINICIÓN DE UI	16
2.7.2 PRINCIPIOS DE DISEÑO DE UI	16
2.7.3 DEFINICIÓN DE UX	17
2.7.4 CONSIDERACIONES DE UX EN APLICACIONES MÓVILES	18
2.8 ARQUITECTURAS DE SOFTWARE EN DESARROLLO MÓVIL	19

2.8.1	PATRÓN MVC	20
2.8.2	PATRÓN MVVM	21
2.8.3	PATRÓN MVP	23
CAPÍTULO 3: PROPUESTA DE SOLUCIÓN		25
3.1	OBJETIVOS	25
3.1.1	OBJETIVO GENERAL	25
3.1.2	OBJETIVOS ESPECÍFICOS	25
3.2	JUSTIFICACIÓN DEL PROYECTO	25
3.3	PROCESOS	26
3.4	DISEÑO DE LA INTERFAZ DE USUARIO	27
3.5	DISEÑO DE LA APLICACIÓN	28
3.5.1	ELECCIÓN DE HERRAMIENTAS	28
3.5.2	ELECCIÓN DE PATRÓN DE DISEÑO	29
3.5.3	FLUJO DE TRABAJO DE LA APLICACIÓN	29
3.5.4	CASOS DE USO	29
CAPÍTULO 4: VALIDACIÓN DE LA SOLUCIÓN		34
4.1	VALIDACIÓN DE FUNCIONALIDAD	34
4.2	VALIDACIÓN DE LA UX	34
4.2.1	PLAN DE VALIDACIÓN UX	34
4.3	RESULTADO DE VALIDACIÓN DE LA UX	35
4.3.1	RESULTADOS Y ANÁLISIS DE ENCUESTA VALIDACIÓN UX	35
4.3.2	RESULTADOS Y ANÁLISIS DE ENCUESTA VALIDACIÓN UI	40
4.3.3	CONCLUSIÓN DE LOS RESULTADOS GENERALES	43
4.4	VALIDACIÓN DE OPTIMIZACIÓN	43
4.4.1	PLAN DE VALIDACIÓN DE OPTIMIZACIÓN	43
CAPÍTULO 5: CONCLUSIONES		44
5.1	LOGROS	44
5.2	LECCIONES APRENDIDAS	44
5.2.1	EXPERIENCIA EN EL DESARROLLO	44
5.2.2	PLANIFICACIÓN DETALLADA	44
5.3	LIMITACIONES Y ÁREAS DE MEJORA	45
5.3.1	LIMITACIONES	45
5.3.2	FUTURAS MEJORAS	45
5.4	IMPACTO Y RELEVANCIA	45
5.4.1	IMPACTO	45
5.4.2	RELEVANCIA EN LA INDUSTRIA	46
5.5	PERSPECTIVAS FUTURAS	46
5.5.1	INVESTIGACIÓN ADICIONAL	46
5.5.2	EXPANSIÓN DE PLATAFORMAS	46
5.6	REFLEXIÓN FINAL	46
5.6.1	DESAFÍOS SUPERADOS	46

5.6.2 EXPERIENCIA PERSONAL	47
REFERENCIAS BIBLIOGRÁFICAS	48

ÍNDICE DE FIGURAS

1	Diagrama de Arquitectura de plataforma Android. Fuente: https://developer.android.com	11
2	Diagrama de Arquitectura de plataforma iOS. Fuente: https://miro.medium.com	13
3	Patrón de diseño MVC.	20
4	Patrón de diseño MVVM.	22
5	Patrón de diseño MVVM.	23
6	Versión 1.0 de la pantalla de lectura.	27
7	Diagrama de Flujo del uso de la Aplicación.	29
8	Respuestas de la pregunta N°1 Encuesta de validación de UX.	36
9	Respuestas de la pregunta N°2 Encuesta de validación de UX.	36
10	Respuestas de la pregunta N°3 Encuesta de validación de UX.	37
11	Respuestas de la pregunta N°4 Encuesta de validación de UX.	38
12	Respuestas de la pregunta N°4 Encuesta de validación de UX.	38
13	Respuestas de la pregunta N°5 Encuesta de validación de UX.	39
14	Respuestas de la pregunta N°6 Encuesta de validación de UX.	39
15	Respuestas de la pregunta N°1 Encuesta de validación de UI.	40
16	Respuestas de la pregunta N°2 Encuesta de validación de UI.	41
17	Respuestas de la pregunta N°3 Encuesta de validación de UI.	41
18	Respuestas de la pregunta N°4 Encuesta de validación de UI.	42
19	Respuestas de la pregunta N°5 Encuesta de validación de UI.	42

ÍNDICE DE TABLAS

1	Caso de Uso CU-101	30
2	Caso de Uso CU-102	31
3	Caso de Uso CU-103	31
4	Caso de Uso CU-104	32
5	Caso de Uso CU-105	32
6	Caso de Uso CU-106	33

INTRODUCCIÓN

Los avances de la tecnología móvil han transformado radicalmente la forma en la que interactuamos con cosas de la vida cotidiana. En este contexto, el desarrollo de aplicaciones móviles ha emergido como un campo crucial dentro de la ingeniería informática, impulsando de esta manera, la innovación y mejorando la accesibilidad para las personas, a través de dispositivos móviles.

En este documento se abordan específicamente las áreas de la síntesis de voz y del desarrollo móvil, destacando la importancia y la evolución de ambas disciplinas en el marco de la ingeniería informática.

La naturaleza diversa y dinámica dentro del desarrollo móvil plantea desafíos únicos, desde la creación de aplicaciones intuitivas hasta la gestión eficiente de recursos en dispositivos con capacidades variadas. Dentro de este contexto, este trabajo se sumerge en la comprensión profunda de las estrategias, tecnologías y mejores prácticas asociadas con el desarrollo de aplicaciones móviles. Se explorarán tanto plataformas móviles líderes, como Android e iOS, como los enfoques de desarrollo nativos e híbridos, evaluando sus ventajas y desventajas en diferentes contextos de aplicación.

Este proyecto propone una aplicación móvil innovadora que aborda un problema específico dentro del área técnica de la salud, es decir, equipos médicos que requieran de mantenimientos preventivos. La solución propuesta no solo se enfoca en la implementación técnica, sino también en la experiencia del usuario, reconociendo la importancia de la usabilidad y la interactividad en el éxito de las aplicaciones móviles.

A lo largo de este informe, se detallará la metodología aplicada para el desarrollo de la aplicación, incluyendo la selección de plataformas y tecnologías, patrón de diseño utilizado y la elección de los lenguajes de programación. Además, se presentará una revisión exhaustiva de la literatura sobre la síntesis de voz y el desarrollo móvil, contextualizando nuestro enfoque en el marco conceptual existente.

La estructura del documento sigue una secuencia lógica que refleja el proceso de desarrollo, desde la conceptualización y el diseño hasta la implementación y las pruebas. Cada capítulo se adentra en aspectos específicos del proyecto, proporcionando una visión completa y detallada de la propuesta de solución. Con esto se proporciona a las y los lectores una guía clara a medida que se sumergen en los aspectos técnicos y conceptuales de esta memoria.

CAPÍTULO 1

DEFINICIÓN DEL PROBLEMA

1.1. IDENTIFICACIÓN DEL PROBLEMA

En esta memoria se tratará un problema que se encuentra inserto en el área de la salud, más específicamente dentro del área de equipos médicos como un electro bisturí, entre otros equipos.

El principal problema por tratar es la dificultad para seguir un protocolo de mantenimiento preventivo de un equipo médico de manera eficiente y eficaz, esto debido a la extensión de este o la cantidad de pasos que conlleva realizar dicho procedimiento, y también la atención que se requiere al momento de leer el documento, además de la cantidad de recursos que se destinan para el cumplimiento de dichos protocolos. [Malik, 1979]

Actualmente en esta situación específica a la que se quiere apuntar en este proyecto, estas tareas de mantenimiento se realizan siguiendo los instructivos de mantenimiento que se encuentran en archivos PDF, los cuales generalmente necesitan ser leídos por otra persona para la que esté trabajando con el equipo y las herramientas de medición. De igual manera existen diversos modelos de mantenimiento preventivo. [Wu, 2011]

Además cabe destacar que respecto a la destinación de recursos, nos encontramos con una problemática importante, generalmente se destinan dos o más personas para una tarea que podría ser realizable por una persona, lo que significa un malgasto o mal aprovechamiento de recursos, tanto humanos como financieros.

1.1.1. CONTEXTO DEL PROBLEMA

Para contextualizar, un procedimiento de mantención preventiva consiste en realizar labores de mantenimiento sobre el equipo en cuestión, las cuales se encuentran programadas de manera periódica con el fin de evitar futuras anomalías y malfuncionamiento del equipo. Estas se deben llevar a cabo dentro de un periodo de fechas establecidas para lograr un funcionamiento óptimo de los equipos y evitar que se generen fallos o averías graves debido a la falta de estas mantenciones. [Oke y Charles-Owaba, 2006]

Estos procedimientos se llevan a cabo siguiendo un instructivo, el cual se encuentra en formato **PDF** y que siguen una estructura en forma de pasos a seguir. La idea es que la persona encargada de realizar esta labor siga el paso a paso de este instructivo a medida que va llevando a cabo las instrucciones de cada paso.

Adicionalmente, en el proceso de realizar el mantenimiento, se dispone de un grupo de per-

sonas encargadas de esta tarea, asignando roles a cada uno de los integrantes. Generalmente se asigna a un integrante el rol de leer el instructivo para asignarle a otro integrante el rol de llevar a cabo el mantenimiento y utilizar los equipos de medición y prueba sobre el equipo al que se le esté realizando mantenimiento.

1.2. USUARIOS AFECTADOS POR EL PROBLEMA

En cuanto a los usuarios afectados, los más afectados en esta situación son las y los Ingenieros y Técnicos de terreno encargados de realizar estos procesos, e indirectamente el empleador debido que se necesita destinar cierta cantidad de recursos tanto humanos como financieros para realizar estas tareas. Dentro de las dificultades que enfrentan los actores del problema están:

- **Disponibilidad de espacio:** Tanto el equipo bajo prueba como los sistemas de medición utilizan espacio físico, además del documento guía para el protocolo, el cual se puede encontrar de forma física, en una computadora portátil o en otro dispositivo que podría utilizar espacio.
- **Posibilidad de llevar a cabo el protocolo de manera incorrecta:** El personal encargado de realizar la mantención puede realizarlo de manera errónea debido a las distracciones o desviaciones durante la lectura del documento.
- **Falta de tiempo y retrasos:** Debido a la extensión de los protocolos de mantenimiento preventivo, el protocolo se puede prolongar más de lo deseado, generando así retrasos en caso de que la o el Ingeniero/Técnico deba realizar mantenimiento de otro equipo, resultando en una acumulación de tareas.

1.2.1. ¿QUÉ PASARÍA SI NO SE RESOLVIERA EL PROBLEMA?

En el caso de no solucionarse este problema se mantendría una situación en donde no se optimizan recursos y donde se pueden seguir cometiendo errores durante la ejecución de estos procedimientos. Esto implicaría que se seguiría requiriendo de más personal del necesario para realizar estas tareas y además, seguiría siendo un proceso lento y no tan ágil.

1.3. MOTIVACIÓN DEL PROYECTO

La motivación detrás de este proyecto es utilizar y aprovechar las herramientas y avances tecnológicos existentes para optimizar en tiempo y recursos algunos procedimientos y de esta manera, simplificar también el proceso para los actores relacionados con esta idea.

Adicionalmente quiero destacar también, que este proyecto podría aplicarse a distintas áreas industriales que requieran de seguir un protocolo de manera estructurada y donde se requiera atención plena del actor que lleve a cabo esta tarea.

1.4. COMPETENCIA

Es importante mencionar que no existe una competencia directa en cuanto a la lectura de procedimientos o de instructivos, si bien existen diversas herramientas que transforman texto a voz, hasta el momento ninguna ha sido aplicada en este ámbito. Con esto se logró identificar ese espacio dentro de esta área por lo que se ideó este proyecto para presentar una solución a ese espacio.

Adicional a esto, como ya se mencionó anteriormente, este proyecto tiene potencial para ser aplicado en diversas áreas de las industrias. Esto quiere decir que su alcance podría ser amplio en su forma más general. De esta manera se podrían llevar a cabo tareas de mantenimiento de equipos de otras áreas tales como la Construcción, Industrias Químicas, entre otras, debido al uso maquinarias o equipos que requieren de igual manera de mantenimientos preventivos para asegurar un funcionamiento óptimo.

CAPÍTULO 2

MARCO CONCEPTUAL

2.1. TECNOLOGÍAS DE TEXTO A VOZ

Para este proyecto se utilizará tecnología de texto a voz en conjunto con el desarrollo de la aplicación móvil. Para comprender el funcionamiento de estas herramientas debemos hacer un repaso por las distintas opciones existentes junto a sus ventajas y desventajas.

2.1.1. DEFINICIÓN DE TEXT TO SPEECH (TTS)

Las tecnologías de Texto a Voz se refieren a sistemas y métodos que permiten la conversión de texto a voz artificial de manera automatizada. Estas tecnologías desempeñan un papel importante en diversas aplicaciones, tal como la accesibilidad, asistentes virtuales, narración de contenido multimedia y comunicación con dispositivos electrónicos, entre otros. [Allen *et al.*, 1987]

2.1.2. MODELOS DE SÍNTESIS DE VOZ

Existen diversos modelos de síntesis de voz que se utilizan para desarrollar estas tecnologías:

- **Síntesis Concatenativa:** Este enfoque implica la concatenación de fragmentos de voz pregrabados, como fonemas y sílabas para formar palabras y oraciones a través de la combinación de estas. [Khan y Chitode, 2016]
 - Este puede entregar alta calidad y naturalidad en la voz, además de expresividad muy cercana a la de un ser humano real, lo que la hace adecuada para aplicaciones donde se necesita naturalidad y expresividad en la narración, además de ser adaptable a diferentes voces. El problema es que requiere de una gran cantidad de datos para el entrenamiento debido a que se necesitan datos de habla grabada, por lo que se vuelve costoso en términos de almacenamiento y procesamiento. Debido a la forma de síntesis es limitada al momento de personalizar la voz en tiempo real, ya que si se quiere generar otro tipo de voz se necesitan las grabaciones del habla de antemano. Un punto importante para mencionar también es el hecho de que puede tener dificultades al momento de que se requiera de la lectura de palabras que se encuentren fuera del corpus del entrenamiento.

- **Síntesis Basada en Formantes:** En este enfoque, se generan voces sintéticas a partir de modelos matemáticos que simulan la producción de sonido humano. [Chicaiza Ayala, 1996]
 - Esta entrega un control preciso sobre las características de la voz, como la altura, la velocidad y la pronunciación, además no requiere de grandes cantidades de datos como otros enfoques, lo que puede simplificar el proceso de desarrollo y adicionalmente este tipo de síntesis tiende a ser más eficiente en términos de recursos computacionales y almacenamiento. Esta puede tener una calidad limitada debido a la falta de naturalidad, por lo que puede sonar robótica o artificial en comparación a las voces humanas reales, carece también de expresividad, resultando en un habla monótona y falta de matices, además cuenta con dificultades al momento de ajustar la síntesis a otros idiomas y acentos, ya que no logra la misma autenticidad que otros modelos más avanzados.
- **Síntesis basada en modelos HMM:** Este es un método que se basa en modelos ocultos de Márkov, que tiene como objetivo determinar parámetros desconocidos a partir de los parámetros observables. También se le llama síntesis estadística paramétrica. En este caso el HMM modela de manera simultánea el espectro de frecuencias (tracto vocal), a frecuencia fundamental (fuente de la voz) y la duración del habla (prosodia). [Achour *et al.*, 2023]
 - Esta cuenta con una calidad de síntesis de voz aceptable para diversas aplicaciones tales como sistemas de navegación, respuesta automática de voz y lectura de texto en voz alta, además cuenta con un menor uso de recursos a comparación de otros enfoques más complejos; se puede adaptar a diferentes idiomas y voces lo que le da versatilidad. Al contar con una calidad aceptable para aplicaciones simples, se vuelve limitada gracias a que tiende a ser menos natural y expresiva, generando así un discurso monótono que podría no ser el ideal para tareas donde se requiera un discurso de mayor duración. También requiere de datos de entrenamiento extensos para lograr un rendimiento aceptable, lo que resulta en altos costos en término de tiempo y recursos, y definitivamente no es adecuado para aplicaciones donde se requiera una calidad superior por lo antes mencionado.
- **Síntesis Articulatoria:** Se refiere a técnicas computacionales que se basan en el tracto vocal humano y los procesos de articulación que ocurren en él. Debido a la complejidad de este enfoque, es el método con menos estudios de parte de los investigadores hasta ahora. [Lin, 1991] y [2006]
 - Este enfoque logra entregar control y personalización de la voz generada, así como naturalidad y expresividad y adaptabilidad a diversos idiomas y acentos, todo esto gracias a que simula los movimientos de los órganos articulatorios. La problemática es que posee mayor complejidad computacional debido a lo sofisticado de los cálculos y modelos para simular estos movimientos, lo que puede

resultar en altos tiempos de procesamiento, además requiere de datos detallados de entrenamiento para lograr una síntesis de alta calidad, obtener esos datos puede ser un gran desafío, y en resumen puede ser menos eficiente en términos de recursos debido a su complejidad computacional a comparación de otros modelos.

- **Síntesis de ondas sinusoidales:** Es una técnica que genera la voz mediante la combinación de múltiples ondas sinusoidales, cada una con una frecuencia, amplitud y fase específica, para recrear el sonido que se desea. En este enfoque se descompone sonidos en componentes sinusoidales simples, llamados armónicos, lo que permite la generación de una amplia variedad de sonidos mediante la manipulación precisa de dichos componentes. [Macon, 1996] y [McAulay y Quatieri, 1986]
 - Este enfoque entrega un control preciso sobre los detalles del sonido (frecuencia, amplitud y fase) de cada componente sinusoidal, lo que permite generar voces específicas con alta fidelidad, además es eficiente en términos de almacenamiento debido que solo se almacenan parámetros de los componentes sinusoidales y permite una gran flexibilidad para crear sonidos además de las voces, como efectos sonoros y música. Este puede ser complejo de implementar, debido que se necesitan conocimientos de acústica y la matemática involucrada, además que puede contar con un alto tiempo de procesamiento gracias su necesidad intensiva de recursos computacionales y puede tener dificultades para lograr una naturalidad completa en sonidos como la voz, debido a sus complejas características.

2.2. AVANCES RECIENTES EN SÍNTESIS DE VOZ

Cabe destacar la existencia de un gran e importante avance dentro de la síntesis de voz, este es:

- **Personalización de Voces:** Se han desarrollado sistemas con tecnología TTS que permiten a los usuarios la personalización de las voces generadas a través de estos, permitiendo la adaptación a las preferencias del usuario o permitiendo la replicación de voces específicas, lo que tiene aplicaciones en la asistencia médica y la narración de contenido personalizado.

2.3. APLICACIONES DE SÍNTESIS DE VOZ

Las aplicaciones que utilizan esta tecnología son diversas y contemplan desde la accesibilidad para personas con discapacidad visual hasta la interacción con dispositivos mediante el reconocimiento de voz. Algunos ejemplos son:

- **Lectores de Pantalla:** Para personas con discapacidad visual o personas con dislexia y otras dificultades para la lectura, los lectores de pantalla utilizan TTS para convertir texto en voz, permitiendo el acceso a contenido digital. Por ejemplo, dispositivos como teléfonos celulares, computadoras, consolas de videojuegos, entre otros. [Chalamandaris *et al.*, 2010]
- **Asistentes Virtuales:** Asistentes como Siri, Google Assistant, Alexa y Cortana utilizan TTS para proporcionar respuestas en voz a las consultas realizadas por los usuarios. [Stan y Lőrincz, 2021]
- **Usos pedagógicos:** Al momento de enseñar un nuevo idioma o lengua se puede utilizar TTS para practicar la pronunciación. [Liakin *et al.*, 2017]

2.3.1. DESAFÍOS Y FUTURAS DIRECCIONES

A pesar de dichos avances, persisten desafíos en el campo de la tecnología TTS, como la mejora en la naturalidad de las voces sintéticas y la mitigación de la discriminación de género y sesgos en las voces generadas. Las futuras direcciones de investigación podrían centrarse en la optimización de modelos de lenguaje, la personalización de voces más precisa y la exploración de aplicaciones emergentes en campos como la educación y la atención médica.

Otro desafío presente en esta área es respecto a que los sistemas de TTS no comprenden el significado de las palabras, ya que solo se encargan de narrarlas, por ende, esto puede llevar a errores y faltas de comprensión.

La conversión de números y abreviaciones también plantea un desafío importante. Los sistemas TTS pueden inferir la expansión de números basándose en el contexto, pero en el caso de los números romanos y algunas abreviaciones pueden resultar ambiguas. Los sistemas TTS más avanzados pueden predecir la ambigüedad de las abreviaciones, mientras que el común puede generar resultados incoherentes.

2.4. DESARROLLO MÓVIL

En esta sección se explicarán conceptos claves relacionados con las aplicaciones móviles y el desarrollo de estas, adicionalmente se proporcionarán definiciones para comprender el contexto de este proyecto y la planificación del mismo.

2.4.1. DEFINICIÓN DE APLICACIÓN MÓVIL

Una aplicación móvil no es más que un programa informático diseñado para funcionar tal y como su nombre lo indica, en dispositivos móviles, tales como teléfonos, tabletas e inclu-

so relojes inteligentes. Estas pueden abarcar distintas áreas tales como la comunicación, el entretenimiento, negocios, y un extenso etcétera. [Islam *et al.*, 2010]

Dentro de las aplicaciones podemos encontrar distintos tipos de aplicaciones móviles según su construcción y funcionamiento: Nativas, Híbridas y Web. [COLOMBO, 2015]

- **Aplicaciones Nativas:** Estas aplicaciones son desarrolladas específicamente para una plataforma (iOS o Android), lo que resulta en un rendimiento óptimo, un acceso total a las funcionalidades del dispositivo y se integran perfectamente a la plataforma por el hecho de ser nativas. Además estas pueden distribuirse fácilmente a las tiendas respectivas de cada plataforma (Google Play Store o Apple App Store).
- **Aplicaciones Híbridas:** Estas aplicaciones son desarrolladas con enfoque multiplataforma, estas suelen tener un rendimiento menos ligero que las nativas debido que funcionan en un entorno intermedio. En cuanto a la interfaz de usuario, en este caso suele ser más personalizable pero puede ser no tan fluida o nativa en apariencia. Adicionalmente en temas de distribución a tiendas se requiere de un proceso adicional de compilación y empaquetado específico para cada plataforma.
- **Aplicaciones Web:** Estas aplicaciones son desarrolladas para utilizarse a través de un navegador web, en cualquier dispositivo móvil que posea uno. Estas aplicaciones aprovechan las capacidades de los navegadores para ofrecer una experiencia similar a las aplicaciones nativas, pero cuentan con limitaciones al momento de acceder a características del dispositivo en comparación con las aplicaciones nativas.

2.4.2. DEFINICIÓN DE DESARROLLO MÓVIL

El desarrollo de aplicaciones móviles es un área dentro del desarrollo de software enfocada directamente en los dispositivos móviles como los que se ha mencionado anteriormente. Además este contempla el proceso de diseñar, crear y mantener aplicaciones de software específicamente destinadas a estos dispositivos. Este enfoque de desarrollo implica la utilización de lenguajes de programación, herramientas y técnicas adaptadas a las limitaciones de hardware y las características únicas de las plataformas móviles. El desarrollo móvil busca proporcionar a los usuarios experiencias interactivas, funcionales y eficientes a través de aplicaciones instalables o basadas en la web. [Gasimov *et al.*, 2010]

2.4.3. EVOLUCIÓN DEL DESARROLLO MÓVIL

Durante el último tiempo hemos podido observar la importancia y el protagonismo que han adquirido las aplicaciones móviles, su rápido crecimiento y evolución al igual que muchas de las tecnologías hoy presentes, además estas se han ido convirtiendo en parte importante

del día a día del ser humano considerando temas de comunicación y funcionalidades varias tales como recordatorios, alarmas, e incluso entretenimiento.[Phongtraychack *et al.*, 2018]

2.5. PLATAFORMAS MÓVILES Y SISTEMAS OPERATIVOS

Como se ha mencionado anteriormente existen diferentes plataformas en las que se pueden desarrollar y desplegar estas aplicaciones, siendo las principales iOS y Android. A continuación se entregarán definiciones y se explicarán términos para entender el funcionamiento y la importancia de estas plataformas.

2.5.1. ANDROID

Android es un sistema operativo para dispositivos móviles desarrollado por Google, este ha sido importante en el desarrollo de aplicaciones así como en el desarrollo de dispositivos móviles [Singh, 2014]. Para entender un poco el funcionamiento de la plataforma hablaremos de la arquitectura de esta y como se comporta.

Como se muestra en la Figura 1 podemos observar la arquitectura de la plataforma Android, en donde se aprecia un patrón de capas y componentes para otorgar flexibilidad y un rendimiento eficiente en los dispositivos.

Para comprender el funcionamiento de esta arquitectura a continuación se explicará a grandes rasgos la función de cada capa como se explica en [Brahler, 2010] y [Gunasekera, 2012].

- **Linux Kernel:** Esta es la capa mas baja de Android y es la encargada de proporcionar las funcionalidades básicas del sistema operativo, como la gestión de memoria, la administración de procesos, la seguridad y controlar el hardware. Esta funciona como una interfaz entre el hardware y las capas superiores.
- **Capa Abstracta de Hardware:** Esta capa actúa como un puente entre la capa del kernel de Linux y las capas superiores. Esta proporciona una interfaz para los controladores de hardware de hardware específicos del dispositivo, lo que permite que las capas superiores interactúen con el hardware sin preocuparse por detalles de la implementación.
- **Bibliotecas de C/C++ Nativas:** Esta capa incluye librerías de C/C++ que proporcionan funciones esenciales para el sistema. Esto incluye librerías para gráficos 2D y 3D, acceso a base de datos SQLite, conectividad de red, seguridad, entre otras.
- **Entorno de Tiempo de Ejecución de Android:** En esta capa es donde se ejecutan las aplicaciones Android. ART (Android Runtime) es el entorno de tiempo de ejecución

predeterminado para las nuevas versiones de Android. El entorno convierte el código fuente de las aplicaciones (escrito en Java o Kotlin) en código de máquina que el dispositivo puede ejecutar.

- **Java API Framework:** Esta capa es la encargada de proporcionar las bibliotecas y componentes necesarios para el desarrollo de aplicaciones Android. Esta incluye una variedad de funciones como la gestión de notificaciones, el acceso a sensores, entre otras, facilitando el desarrollo de estas aplicaciones.
- **System Apps:** Esta capa incluye las aplicaciones que son parte integral del dispositivo y que son proporcionadas por el fabricante. Algunos ejemplos son el marcador de teléfono, la galería de imágenes, el reproductor de música, entre otras aplicaciones.

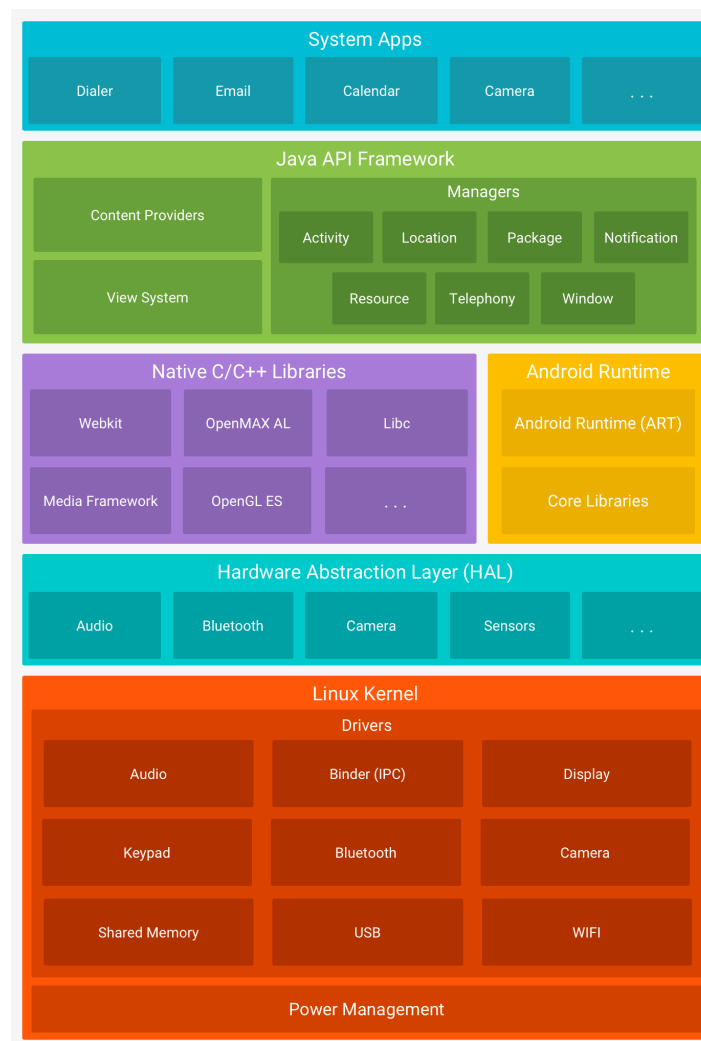


Figura 1: Diagrama de Arquitectura de plataforma Android. Fuente: <https://developer.android.com>

2.5.2. IOS

iOS es el sistema operativo desarrollado por Apple Inc. para dispositivos móviles como iPhone, iPad y iPod Touch. Esta plataforma ha tenido un papel importante dentro de la revolución de dispositivos móviles y ha transformado la forma en la que interactuamos con estas tecnologías. [Novac *et al.*, 2017]

Como podemos observar en la Figura 2 esta es la arquitectura de la plataforma que cuenta con una serie de capas y componentes que trabajan en conjunto para otorgar una experiencia fluida y segura dentro de sus dispositivos.

Para comprender de mejor manera como funciona la arquitectura de iOS se explicarán cada una de sus capas y sus funciones. [Garg y Baliyan, 2021]

- **Cocoa/Cocoa Touch:** Esta capa contiene los frameworks de desarrollo para las aplicaciones siendo Cocoa para macOS y Cocoa Touch la versión para dispositivos móviles, el cual entrega un conjunto de APIs para el desarrollo de interfaces de usuario, gestión de eventos, manipulación de datos y comunicación con el sistema.
- **Media:** Esta capa se centra netamente en el manejo del contenido multimedia, incluyendo gráficos, animación y sonido en las aplicaciones. Además incluye herramientas para la captura y reproducción de medios, lo que permite a las aplicaciones trabajar con imágenes, videos y audio de manera eficiente.
- **Core Services:** Esta capa ofrece una variedad de servicios y funcionalidades clave para las aplicaciones de iOS. Esta incluye herramientas para el acceso a datos, como Core Data (para la administración de bases de datos) y Core Location (para la geolocalización). Además proporciona servicios de res y conectividad, permitiendo a las aplicaciones comunicarse a través de internet y redes locales. Y por último mencionar las herramientas para notificaciones, acceso a contactos y eventos y la gestión de preferencias del usuario.
- **Core OS:** Esta es la capa más baja de la arquitectura de iOS y se centra en la interacción con el hardware, esta proporciona el acceso al kernel del sistema operativo y al hardware específico, lo que permite la administración de recursos. Además es responsable de garantizar la seguridad del sistema, incluyendo la autenticación, el cifrado de datos y la administración de claves. Esta capa desempeña un papel importante en la protección de la plataforma contra amenazas de seguridad.

Todas estas capas trabajan en conjunto para proporcionar una base sólida para el desarrollo de aplicaciones en iOS. Todo esto para ofrecer una experiencia de usuario de alta calidad. Esta arquitectura está diseñada para garantizar coherencia, seguridad y rendimiento en sus distintos dispositivos que ejecutan este sistema operativo.

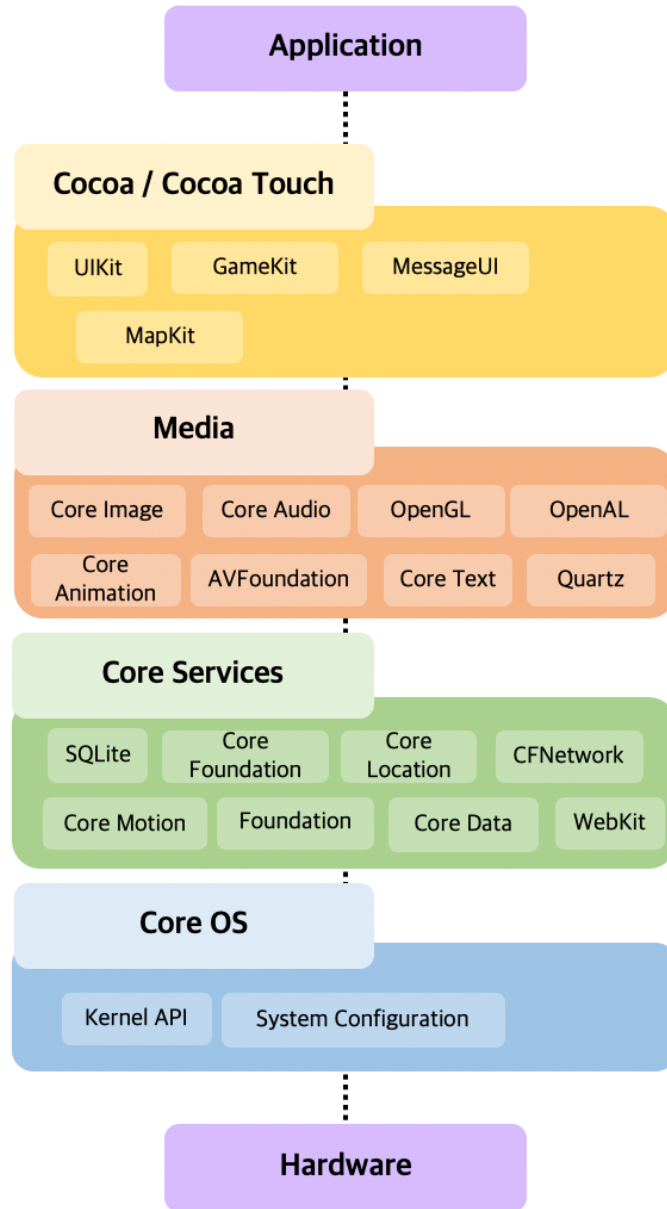


Figura 2: Diagrama de Arquitectura de plataforma iOS. Fuente: <https://miro.medium.com>

2.6. HERRAMIENTAS DE DESARROLLO Y LENGUAJES DE PROGRAMACIÓN

Para ambas plataformas contamos con sus respectivas herramientas para el desarrollo de aplicaciones y lenguajes de programación específicos para cada una.

2.6.1. ANDROID STUDIO

Android Studio es el IDE (Entorno de Desarrollo Integrado) oficial para la plataforma Android el cual cuenta con una importante comunidad de desarrolladores y posee características que lo vuelven una opción viable para el desarrollo de aplicaciones móviles. [Studio, 2017]

- **Características y Componentes Principales:** Este como muchos IDEs cuenta con su propio editor de código, depurador, la administración de proyectos y la emulación de dispositivos. Además Android Studio cuenta con una interfaz intuitiva y herramientas de productividad.
- **Plugins y Extensiones:** Adicional a los componentes principales vistos en el punto anterior, contamos con una gran extensión de plugins y extensiones disponibles para los desarrolladores. Ejemplo de esto tenemos:
 - **Plugin de Kotlin:** Este permite el desarrollo de aplicaciones utilizando el lenguaje de programación Kotlin como alternativa a Java.
 - **Plugin NDK (Kit de Desarrollo Nativo):** Este facilita la integración de código C/C++ en las aplicaciones.
 - **Plugin de Firebase:** Este permite la integración de Firebase dentro de las aplicaciones, entregando herramientas de autenticación, análisis y notificaciones en la nube.
 - **Plugins de Control de Versiones:** Android Studio es compatible con sistemas de control de versiones como Git para poder gestionar las versiones de las aplicaciones dentro del mismo entorno.
- **Lenguajes de programación:** Para el desarrollo de aplicaciones en Android contamos con diferentes opciones al momento de elegir un lenguaje de programación.
 - **Java:** Es uno de los lenguajes de programación mas utilizados al momento de desarrollar aplicaciones gracias a sus ventajas en orientación a objetos, portabilidad robustez. Algunos ejemplos de aplicaciones móviles desarrolladas en Java son Uber, Instagram, Spotify, entre otras conocidas.
 - **Kotlin:** Kotlin es un lenguaje moderno adaptado en el desarrollo de aplicaciones Android. Cuenta con ventajas como la interoperabilidad con Java, además Kotlin simplifica la escritura de código en comparación a Java lo que conlleva a que sea ampliamente adoptado por parte de la comunidad de desarrolladores. Algunos ejemplos de aplicaciones desarrolladas en Kotlin son Pinterest, Trello, Coursera, HBO Max, entre otras.
 - **Lenguajes Nativos (C/C++):** Además de Java y Kotlin podemos utilizar otros lenguajes para el desarrollo de aplicaciones, especialmente en situaciones que requieren un alto rendimiento. Para estos casos se puede utilizar lenguaje nativo

como C/C++ para el desarrollo de componentes específicos. Se puede lograr interoperabilidad entre C/C++ y Java/Kotlin a través del plugin anteriormente mencionado NDK.

- **Otros Lenguajes:** Además los lenguajes antes mencionados podemos utilizar lenguajes como Dart que se utiliza en desarrollo de aplicaciones con Flutter, o JavaScript para desarrollar PWAs (Progressive Web Apps)
- **Selección del Lenguaje:** Es importante elegir un lenguaje apropiado según los requisitos del proyecto y las preferencias del equipo de desarrollo. Es por eso que se deben tener en cuenta ciertos puntos al momento de realizar la elección como el rendimiento, la comunidad de desarrollo y la experiencia previa del equipo, además de la disposición tecnológica que se tenga para desarrollar el proyecto.

2.6.2. XCODE

XCode es el IDE oficial de Apple para la creación de aplicaciones en plataformas como iOS, macOS, watchOS y tvOS. Esta es una herramienta vital para los desarrolladores que quieran crear aplicaciones nativas para los dispositivos de Apple. XCode ofrece una gama de características y herramientas amplia que facilita el desarrollo, la depuración y la distribución de las aplicaciones en el ecosistema de Apple. [Wentk, 2014]

- **Características y Componentes:** XCode dentro de su IDE ofrece un editor de código que admite múltiples lenguajes de programación, incluyendo Swift, Objective-C, herramientas de depuración de código, emulador de iOS para las pruebas de las aplicaciones en distintos dispositivos, gestión de proyectos y acceso a bibliotecas y frameworks nativos de Apple. Presenta además características de resaltado de sintaxis, auto-completado y sugerencias de código para mejorar la productividad del desarrollador. Además los desarrolladores pueden diseñar interfaces de usuario de manera visual utilizando el Interface Builder.
- **Plugin y Extensiones:** XCode permite integrar diferentes plugins y extensiones tales como:
 - **Plugin de Kotlin:** Si bien XCode se enfoca en Swift, los desarrolladores pueden agregar plugins para admitir lenguajes tales como Kotlin.
 - **Plugin de Flutter y Dart:** También se puede instalar un complemento para crear aplicaciones Flutter utilizando el lenguaje Dart.
 - **Plugin de Control de Versiones:** XCode es compatible también con sistemas de control de versiones como Git para facilitar la colaboración en los proyectos y el seguimiento de cambios en el código fuente.
- **Lenguajes de Programación:** XCode es compatible con dos lenguajes de programación.

- **Swift:** Swift el más actual y altamente seguro, tiene una sintaxis clara y concisa.
- **Objective-C:** Anteriormente a Swift existía Objective-C, el cual a pesar de que Swift ha ganado bastante terreno, aún se utiliza en algunas aplicaciones más antiguas.

2.7. DISEÑO DE INTERFAZ DE USUARIO Y EXPERIENCIA DEL USUARIO

Es importante diseñar y mantener una buena UI, es decir, que sea intuitiva y amigable con el usuario en cuestión. Además se debe asegurar una UX satisfactoria, que se logre satisfacer las necesidades y expectativas del usuario mediante la facilidad de uso, la eficiencia, consistencia, mejora continua y por supuesto una estética atractiva. [Thimbleby, 1990]

2.7.1. DEFINICIÓN DE UI

La UI en el contexto del proyecto, es decir, diseño de software y aplicaciones, hace referencia al punto en donde interactúan los usuarios con un sistema informático, en este caso, con la aplicación móvil. Esta es la capa donde se encuentra todo lo visual y táctil que permite al usuario comunicarse con la aplicación, sistema operativo u otro software. Esta contiene todos los elementos para llevar a cabo la comunicación, como menús, iconos, botones, entre otros.

Las UI tienen como objetivo facilitar la interacción del usuario con el software, haciendo que el uso y la navegación de la aplicación sea intuitivo y eficiente. Un buen diseño de UI tiene en cuenta la usabilidad, la accesibilidad y la estética, todo esto para lograr una experiencia de usuario agradable y efectiva.

En el contexto de las aplicaciones móviles, las UI se adaptan a las dimensiones de las pantallas táctiles, es decir, que son responsivas y utilizan elementos específicos para facilitar la interacción con los dedos. El diseño de la interfaz puede incluir también principios de diseño visual, como la elección de colores, la tipografía y la disposición de los demás elementos para mejorar la legibilidad y la estética de la aplicación, por lo que es sumamente importante realizar un buen diseño. [Marcus, 2002]

2.7.2. PRINCIPIOS DE DISEÑO DE UI

El diseño de UI es una etapa crucial al momento de desarrollar un software, debido a esto es que esta sección del desarrollo cuenta con principios importantes para lograr el éxito deseado.

Algunos de estos principios, que además son los más importantes en mi opinión son:

- **Usabilidad:** En donde la interfaz debe ser fácil de aprender, eficiente de usar y ser proporcional al nivel de experiencia del usuario.
 - **Importancia:** Es crucial el cumplimiento de este principio debido que facilita la interacción y reduce la curva de aprendizaje del usuario, lo que mejora la experiencia del mismo.
- **Diseño Responsivo:** La interfaz debe tener la capacidad de adaptarse de manera fluida a diferentes tamaños de pantalla y dispositivos.
 - **Importancia:** Es importante que la aplicación sea responsiva, ya que esto puede garantizar una experiencia de usuario coherente en diversas plataformas en conjunto con la usabilidad.
- **Accesibilidad:** La interfaz debe contar con la accesibilidad para todos los usuarios, independientemente de sus habilidades o discapacidades.
 - **Importancia:** Es vital que las aplicaciones cumplan con los estándares de inclusividad y que garanticen la interacción de todos los usuarios.
- **Estética:** La interfaz debe mantener un diseño visualmente atractivo y armonioso.
 - **Importancia:** Este punto contribuye a una experiencia agradable y positiva para el usuario, además de que puede asegurar una mejor respuesta de parte del usuario de modo que este mantenga su preferencia sobre nuestra aplicación en lugar de utilizar otras opciones.
- **Retroalimentación:** La interfaz debe proporcionar al usuario respuestas claras e inmediatas a sus acciones.
 - **Importancia:** Con esto se mantiene a los usuarios informados sobre el estado de la aplicación, reduciendo así la incertidumbre, y mejorar el entendimiento de la aplicación.

2.7.3. DEFINICIÓN DE UX

La UX se refiere a la percepción general y la respuesta que experimenta un usuario al interactuar con un producto, sistema o servicio, especialmente en términos de facilidad de uso, eficiencia y satisfacción. Esta abarca todo el proceso y los aspectos de la interacción del usuario, desde la primera impresión hasta la navegación continua y la realización de tareas.

La UX se centra principalmente en optimizar todos los puntos de contacto entre el usuario y el producto o servicio, buscando crear una interacción positiva que conduzca a la satisfacción del usuario, de esta manera se fomenta la lealtad y la preferencia por dicho producto o servicio. Todo esto se logra comprendiendo las necesidades y expectativas de los usuarios, diseñando interfaces intuitivas, entregando retroalimentación clara y garantizando que la interacción sea fluida. [Hyun y Marsden, 2023]

2.7.4. CONSIDERACIONES DE UX EN APLICACIONES MÓVILES

Para entregar una UX satisfactoria o exitosa debemos tener en cuenta ciertas consideraciones, las cuales afectan directamente en la interacción del usuario. Algunas de estas son:

- **Diseño Responsivo:** Es importante asegurar que la interfaz se adapte adecuadamente a los diferentes tamaños de pantalla y orientaciones para proporcionar una experiencia consistente en los dispositivos móviles.
- **Interacción Táctil:** Hay que considerar también, gestos comunes, tales como deslizar, pellizcar, tocar, entre otros, para optimizar la navegación y la actividad.
- **Rendimiento y Velocidad:** Es importante optimizar el rendimiento general de la aplicación para garantizar tiempos de carga rápidos y respuesta ágil a los requerimientos del usuario, lo que contribuye a una experiencia fluida y satisfactoria.
- **Priorización de Contenido:** Es crucial organizar el contenido de manera jerárquica, destacando la información relevante para el usuario y permitiendo a estos acceder de manera más sencilla a lo que están buscando.
- **Accesibilidad:** Es importante asegurar que la aplicación sea accesible para usuarios con dificultades o discapacidades, entregando opciones de accesibilidad como el tamaño del texto, contraste adecuado, soporte para lectores de pantalla, entre otras.
- **Contexto de Uso Móvil:** Hay que considerar el contexto específico del uso móvil, es decir, la movilidad del usuario, la conexión de red variable y la posibilidad interrupciones, para de esta manera, diseñar una experiencia de usuario sin mayores inconvenientes.
- **Pruebas en Dispositivos Móviles Reales:** Es importante realizar pruebas en dispositivos móviles reales para identificar posibles problemas de diseño y de funcionalidad específicos de estas plataformas.
- **Diseño en Modo Nocturno:** Es importante incluir opciones de modo nocturno para mejorar la legibilidad en condiciones de poca luz y proporcionar una experiencia visual cómoda para el usuario en el uso nocturno de la aplicación.
- **Actualizaciones y Mejoras Continuas:** Es vital mantener las experiencias de usuario con mejoras continuas, validando la experiencia mediante diferentes técnicas de validación, por ejemplo, encuestas.

Con estas consideraciones podemos asegurar un enfoque integral hacia la creación de una experiencia de usuario satisfactoria y efectiva. [Moser y Moser, 2012]

2.8. ARQUITECTURAS DE SOFTWARE EN DESARROLLO MÓVIL

La arquitectura de software es un pilar importante para el desarrollo de aplicaciones móviles y se refiere a la estructura fundamental y organización de un software, que incluye sus componentes básicos, relaciones, principios y decisiones de diseño. Esto define la forma en que los diferentes módulos o componentes del sistema interactuarán entre sí y con el entorno, proporcionando una visión general del mismo.[Valipour *et al.*, 2009]

En este ámbito contamos con elementos clave para la arquitectura de software, los cuales son:

- **Componentes:** Los componentes son las piezas individuales de software que realizan funciones específicas. Estos pueden incluir módulos, bibliotecas, servicios y otros elementos que componen el sistema. Por ejemplo, en un sistema de comercio electrónico, los componentes pueden incluir módulos como "Gestión de Usuarios", "Procesamiento de pagos", "Gestión de Inventarios", entre otros.
- **Conexión:** Las conexiones son las interacciones y relaciones entre los diferentes componentes. Esto incluye cómo se comunican, cómo comparten datos y cómo dependen entre sí. Por ejemplo, en un sistema de redes sociales, la conexión entre el componente "Publicaciones de Usuario" y "Notificaciones" puede implicar que cuando un usuario publica algo, se notifica a sus seguidores.
- **Estilo Arquitectónico:** Es un conjunto de patrones y principios que guían el diseño de la arquitectura. Ejemplo de algunos estilos incluyen la arquitectura en capas, la orientada a servicios, la basada en micro-servicios, entre otras. Para explicar mejor tomaremos la arquitectura basada en micro-servicios, donde un sistema se compone de servicios independientes y modularizados que se comunican entre sí, permitiendo la escalabilidad y la implementación independiente de cada servicio, por ejemplo, Netflix es una plataforma conocida en el área por su enfoque en micro-servicios, todas sus funcionalidades se implementan como un micro-servicio independiente, tal como la gestión de usuarios, la recomendación de contenido, procesamiento de pagos, entre otras.
- **Decisiones de Diseño:** Estas son elecciones específicas que afectan la forma en que se estructura el sistema. Aquí se pueden incluir decisiones sobre la tecnología, la distribución de componentes, la escalabilidad, la seguridad, entre otras. Por ejemplo, la decisión de utilizar una base de datos relacional en lugar de una no relacional para un sistema de gestión de inventarios.
- **Patrones de Diseño:** En este punto se toman en cuenta soluciones recurrentes para problemas comunes de diseño. Estos patrones encapsulan las mejores prácticas y proporcionan enfoques probados para abordar ciertos desafíos en el diseño de software. Por ejemplo, el patrón de diseño MVP separa en tres componentes la lógica de presentación.

Es importante tomar las decisiones idóneas para levantar una arquitectura adecuada para el contexto del proyecto que se esté trabajando

Retomando el último punto referente a los Patrones de Diseño, explicaremos algunos de los más comunes para el desarrollo de aplicaciones móviles.

2.8.1. PATRÓN MVC

El patrón de diseño MVC divide una aplicación en tres componentes principales: el Modelo, la Vista y el Controlador. Cada uno de estos componentes tiene responsabilidades específicas, lo que facilita la organización y mantenibilidad del código. [Bucanek, 2009] [Sharan y Sharan, 2015]

En la Figura 3 podemos observar dichos componentes y una breve explicación de qué realiza cada una de ellas, lo que se desglosará en el siguiente punto.

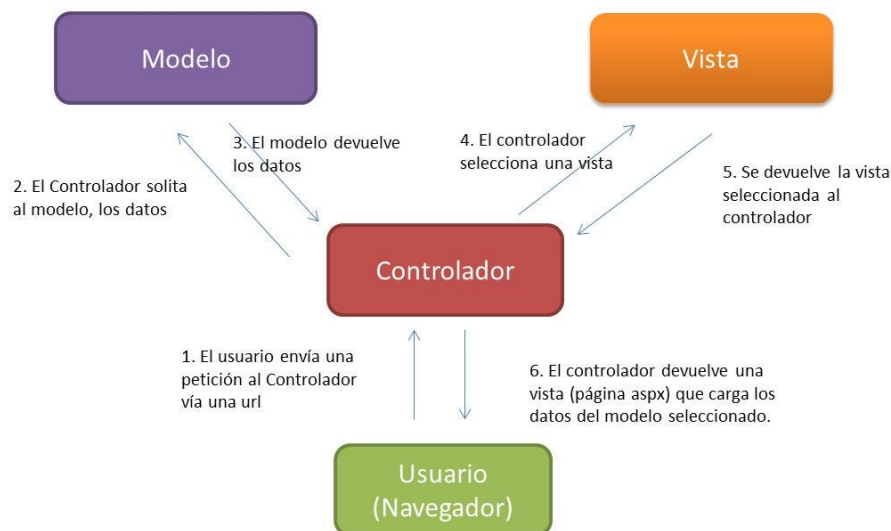


Figura 3: Patrón de diseño MVC.

Fuente: <https://miro.medium.com>.

- **Modelo (Model):** Representa la capa de datos y lógica de la aplicación. Se encarga de almacenar y gestionar datos, así como de realizar operaciones lógicas relacionadas con dichos datos, lo que se conoce como lógica de negocios. El modelo notifica a la vista sobre cambios en los datos.
- **Vista (View):** Representa la interfaz de usuario y se encarga de la presentación. Esta observa al Modelo y actualiza la interfaz gráfica como respuesta. La Vista no realiza la lógica de negocio significativa y se centra en la presentación.

- **Controlador (Controller):** Este actúa como intermediario entre el Modelo y la Vista. Recibe las interacciones del usuario desde la Vista, actualiza el Modelo como respuesta y también actualiza la Vista para reflejar los cambios. El controlador contiene la lógica de manejo de eventos y la lógica de aplicación.

Para entender el funcionamiento a grandes rasgos, podemos describir el flujo de trabajo con el que funciona este patrón:

1. El usuario interactúa con la Vista (por ejemplo, hace clic en un botón).
2. La Vista notifica al Controlador sobre la interacción del usuario.
3. El Controlador actualiza el Modelo según la interacción del usuario.
4. El Modelo notifica a la Vista sobre los cambios en los datos.
5. La Vista actualiza su interfaz gráfica según la información proporcionada por el Modelo.

Algunas de las principales ventajas de este patrón de diseño son:

- **Separación de responsabilidades:** Este patrón permite la separación clara de las responsabilidades de presentación, lógica de la aplicación y de gestión de datos.
- **Mantenibilidad:** Al tener componentes claramente definidos y separados, este patrón facilita el mantenimiento y la modificación de código.
- **Reutilización de componentes:** Nuevamente gracias a la separación de componentes, con este patrón es posible reutilizar Vistas y Modelos en diferentes contextos.

Para ejemplificar podemos pensar en una aplicación de lista de tareas. El Modelo en este caso podría representar las tareas y su estado (Completada/Pendiente), la Vista mostraría la lista de tareas en la interfaz gráfica y el Controlador manejaría las interacciones del usuario, tales como agregar o eliminar tareas, actualizando el Modelo y la Vista.

2.8.2. PATRÓN MVVM

El patrón MVVM es una evolución del patrón MVC anteriormente mencionado y se utiliza generalmente en el desarrollo de aplicaciones de interfaz de usuario.

A diferencia del MVC en vez de un Controlador, en este patrón encontramos otro componente llamado VistaModelo. [Anderson, 2012] [Gaudio, 2010]

En la Figura 4 observamos el componente VistaModelo que reemplaza al Controlador en este patrón.

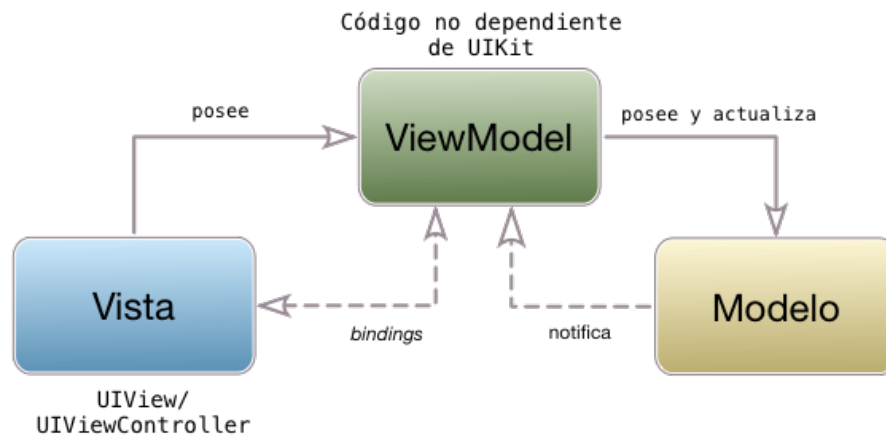


Figura 4: Patrón de diseño MVVM.
Fuente: <https://ottocol.github.io>.

- **Modelo (Model):** Al igual que en el patrón MVC, este componente representa la capa de datos y lógica de la aplicación. Este se encarga de almacenar y gestionar los datos, además de realizar las operaciones relacionadas con esos datos.
- **Vista (View):** Nuevamente al igual que en el patrón MVC, este componente representa la interfaz de usuario y se encarga exclusivamente de la presentación, con la diferencia de que en este caso la Vista observa a la VistaModelo y se actualiza automáticamente cuando se actualiza este cambia.
- **VistaModelo (ViewModel):** Actúa como intermediario entre el Modelo y la Vista. Este contiene la lógica de presentación y expone datos y comandos que la Vista enlaza directamente. La VistaModelo no tiene conocimiento directo de la Vista específica que lo utiliza.

En el caso del flujo de trabajo de este patrón nos encontramos lo siguiente:

1. El Modelo gestiona los datos y realiza operaciones sobre estos.
2. La VistaModelo interactúa con el modelo para obtener y transformar los datos.
3. La Vista observa a la VistaModelo y se actualiza automáticamente cuando este cambia.
4. La Vista puede enviar comandos a la VistaModelo para realizar acciones en el Modelo.

Ejemplificando el mismo caso anterior sobre una aplicación de lista de tareas con el patrón MVVM en este caso, el Modelo gestionaría las tareas y sus estados, la VistaModelo proporcionaría los datos formateados para la interfaz de usuario y mostraría comandos para manipular las tareas, y en el caso de la Vista, esta observaría a la VistaModelo para actualizar la interfaz de usuario según los cambios en esta.

2.8.3. PATRÓN MVP

Al igual que el caso anterior este patrón es una variante del patrón MVC, diseñado para mejorar las pruebas unitarias y al igual que los casos anteriores, la separación de responsabilidades.

En este caso en lugar del Controlador, contamos con el Presentador, el cual está más relacionado con la lógica de presentación y la gestión de eventos. [Zhang y Luo, 2010] [Pau *et al.*, 2010]

Como se observa en la Figura 5, nos encontramos con el Presentador, reemplazando al controlador.

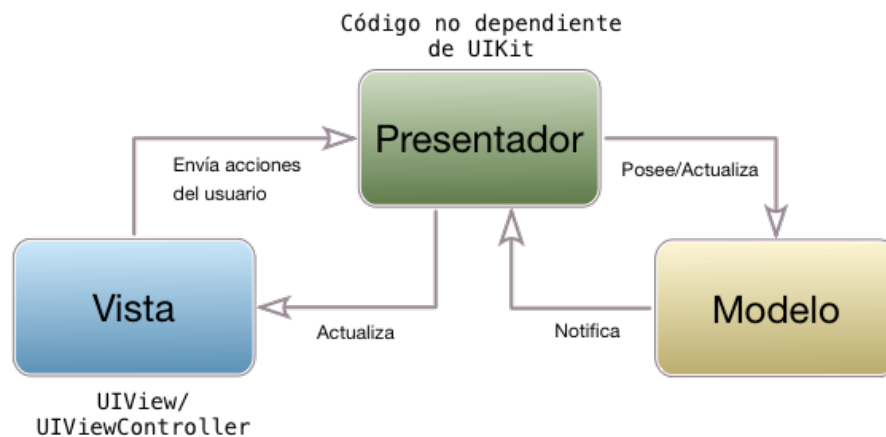


Figura 5: Patrón de diseño MVVM.
Fuente: <https://ottocol.github.io>.

- **Modelo (Model):** Al igual que en el MVC y el MVVM, este se encarga de almacenar y gestionar los datos y realizar operaciones sobre esos datos.
- **Vista (View):** La Vista se encarga de la presentación de los datos. En este caso la Vista observa al Presentador. Esta no contiene la lógica de negocio y se centra únicamente en la presentación.

- **Presentador (Presenter):** Este actúa como intermediario entre el Modelo y la Vista. Este contiene la lógica de presentación y gestiona las interacciones de los usuarios. En este caso el Presentador actualiza la Vista según el estado del Modelo en respuesta a las acciones del usuario.

En cuanto al flujo de trabajo sigue siendo similar debido que es una variante del MVC.

1. El Modelo gestiona los datos y realiza operaciones sobre ellos.
2. La Vista observa al Presentador y responde a las acciones del usuario.
3. El Presentador interactúa con el Modelo para obtener y actualizar los datos.
4. La Vista se actualiza según las indicaciones del Presentador.
5. La Vista puede enviar eventos al Presentador para acciones específicas.

Dentro de las ventajas de este patrón, algunas son adquiridas del patrón MVC:

- **Separación de Responsabilidades:** Al igual que el MVC, este patrón separa las responsabilidades, es decir, que separa la lógica de negocios, la de presentación y la manipulación de datos.
- **Reutilización de Componentes:** Nuevamente, podemos encontrar otra ventaja en común, en este caso se puede reutilizar de Vistas y Presentadores en diferentes contextos.
- **Testabilidad Mejorada:** La lógica de presentación en el Presentador es independiente de la Vista, lo que facilita las pruebas unitarias.

Para ejemplificar, pondremos el caso de una aplicación de reproducción de música utilizando el patrón MVP, el Modelo gestionaría la lista de reproducción y la reproducción de canciones, el Presentador proporcionaría los datos formateados para la interfaz de usuario y manejaría las acciones del usuario, y la Vista observaría al Presentador para actualizar la interfaz de usuario según haya cambios en el Presentador.

CAPÍTULO 3

PROPUESTA DE SOLUCIÓN

3.1. OBJETIVOS

Dentro de esta propuesta se encuentran el Objetivo General y los Objetivos Específicos, los cuales dirigirán el proyecto y se definirán en los siguientes puntos.

3.1.1. OBJETIVO GENERAL

Desarrollar una aplicación móvil con tecnologías de texto a voz, para narrar protocolos de mantenimiento preventivo para equipos médicos optimizando dicho proceso y su ejecución, generando de esta manera una agilización del proceso y optimización de recursos para llevarlo a cabo.

3.1.2. OBJETIVOS ESPECÍFICOS

En cuanto a los Objetivos Específicos están propuestos los siguientes.

- Obtener el documento que contiene el protocolo de mantención del equipo para la extracción de las instrucciones y convertirla en un formato para ingresarla en la aplicación.
- Diseñar el software definiendo el patrón de diseño y diseñando la interfaz de usuario de la aplicación.
- Implementar la aplicación según las especificaciones de diseño.
- Validar parte del objetivo principal de la aplicación, es decir, la optimización del proceso de mantenimiento.

3.2. JUSTIFICACIÓN DEL PROYECTO

Este proyecto se llevará a cabo tomando en cuenta el problema, siendo útil una solución práctica para aquellas y aquellos Ingenieros y Técnicos involucrados en la situación.

De esta manera, con los conocimientos adquiridos durante la carrera, se puede entregar una herramienta que podría optimizar tanto en recursos como en tiempo y facilitar la tarea para dichos usuarios.

3.3. PROCESOS

Para llevar a cabo y poder construir esta solución, es necesario identificar y contextualizar una serie de procesos los cuales nos ayudarán al momento de desarrollar y levantar esta idea para alcanzar el objetivo. Los procesos son los siguientes:

- **Diseñar Interfaz de Usuario:** Se diseñará una interfaz de usuario para las distintas pantallas, las cuales serán simples, usables e intuitivas para el usuario, incluyendo botones y visualizaciones de texto, facilitando el uso de la aplicación, aportando a una experiencia de usuario exitosa.
- **Desarrollar Interfaz de Usuario:** Se desarrollará una interfaz cumpliendo con las especificaciones de diseño.
- **Diseño de la aplicación:** Se diseñará la aplicación eligiendo sistema operativo objetivo, entorno de desarrollo, lenguaje de programación y bibliotecas necesarias.
- **Desarrollo preliminar de la aplicación:** Se desarrollará la base primordial para la aplicación, uniendo la interfaz de usuario junto con la lógica de la aplicación, otorgando un funcionamiento preliminar de esta herramienta, como presentar el texto y navegar en los pasos.
- **Integración de herramientas:** Una vez montada la base de la aplicación se integrarán dentro de ella las bibliotecas de texto a voz para generar la voz narradora, además de la lógica para procesar el texto y narrarlo.
- **Validar funcionamiento preliminar:** Al construir la base de la aplicación se realizarán validaciones intermedias, es decir, se mostrará la aplicación y su funcionamiento al usuario final, siendo la validación su respuesta con respecto a la experiencia de usuario, para analizar que se puede mejorar o cambiar.
- **Procesar el texto:** Se procesará el texto para ser introducido y mostrado en la aplicación mediante la Interfaz de Usuario con la lógica ya creada y para ser emitido por audio mediante la generación de voz antes implementada.
- **Completar el desarrollo:** Una vez de implementen las herramientas y el texto y ya se haya validado la experiencia de usuario, se completará el desarrollo de la aplicación afinando detalles con respecto a la retroalimentación del usuario, completando la aplicación.

- **Validar la aplicación:** Una vez completada e implementada se realizará la validación de la aplicación en distintos ámbitos, tanto en interfaz de usuario, experiencia de usuario y cumplimiento del objetivo mediante el testeo de esta en el área para la que fue ideada.

3.4. DISEÑO DE LA INTERFAZ DE USUARIO

Como primer paso se diseñó la interfaz de usuario para crear una base en la cual montar la aplicación.

Para el diseño de la Interfaz de Usuario se pensó y diseñó una interfaz simple, sin mayores detalles mas que lo necesario, que en este caso sería el texto en cuestión y los botones para navegar en los pasos del instructivo. Todo esto con el objetivo de que sea lo más intuitiva y usable posible.

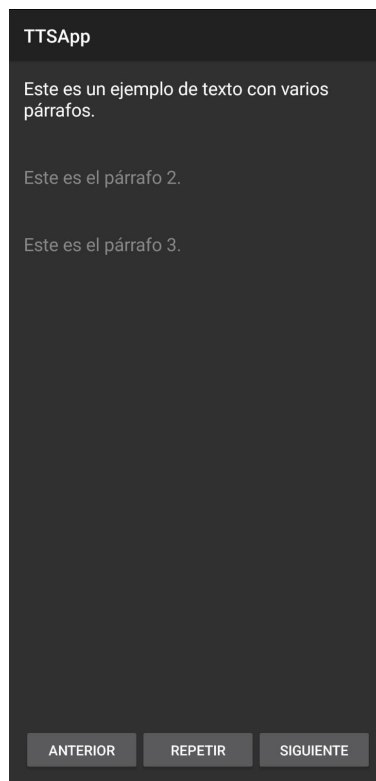


Figura 6: Versión 1.0 de la pantalla de lectura.
Fuente: Elaboración propia.

Como podemos observar en la Figura 6, dentro de la pantalla contamos con 3 textos de ejemplo para mostrar de manera dividida los pasos, además en el primer texto se resalta de color blanco para destacar que es el paso actual, el cual nuestro narrador está leyendo.

En la parte inferior de la pantalla observamos 3 botones los cuales son para retroceder al paso anterior (**ANTERIOR**), para leer o repetir el paso actual (**REPETIR**) y para pasar al paso siguiente (**SIGUIENTE**) respectivamente.

Posteriormente se añadirá un botón para pausar o reanudar la narración, además de un apartado para configurar los parámetros de la voz mencionados en el Caso de Uso 3, de esta manera el usuario podrá personalizar al narrador según su comodidad.

3.5. DISEÑO DE LA APLICACIÓN

Para diseñar la aplicación, hay que tomar en cuenta las herramientas y tecnologías que mencionamos en el capítulo anterior, de esta manera podemos realizar elecciones de cuales utilizaremos para el desarrollo de este proyecto.

3.5.1. ELECCIÓN DE HERRAMIENTAS

Comenzando por el sistema operativo para cual se va a diseñar la aplicación, se decidió que se desarrollará solo para **Android**, mediante **Android Studio**. Todo esto debido que para desarrollar para la plataforma de iOS se requiere de un equipo Mac, como no se dispone de ese equipo ni de los recursos para adquirir uno, la decisión se vió inclinada hacia la plataforma Android.

Continuando por el lenguaje de programación, la aplicación se desarrollará en el lenguaje **Kotlin** debido que existe conocimiento y experiencia previa en este lenguaje como base, además que es una de las opciones de lenguaje que nos ofrece Android Studio, de esta manera se facilita y agiliza el desarrollo de la aplicación al no tener que aprender un lenguaje nuevo y utilizando los conocimientos ya existentes antes mencionados.

En cuanto a la tecnología de Texto a Voz, se utiliza una biblioteca propia de Android, **Android TextToSpeech**, la cual entrega una interfaz para convertir el texto en habla utilizando los motores de síntesis de voz instalados en el dispositivo.

Esta biblioteca utiliza un modelo de síntesis de voz del tipo **concatenativa**, de este modo, la voz generada mantiene una calidad bastante natural lo que la vuelve entendible para el usuario que utilice esta aplicación. Se eligió utilizar esta tecnología debido que de esta manera aprovechamos de mejor manera el entorno que nos ofrece Android para el desarrollo de esta aplicación.

3.5.2. ELECCIÓN DE PATRÓN DE DISEÑO

En el capítulo anterior se mencionaron algunos patrones de diseños utilizados en el desarrollo de aplicaciones móviles, para este proyecto y el desarrollo de la aplicación se utilizará el patrón de diseño **MVP**, de esta forma aprovechamos sus ventajas como la separación de responsabilidades, la facilitación de pruebas unitarias, y la reutilización de componentes. Con esto podemos asegurar una aplicación funcional e integral, realizando las pruebas correspondientes para mejorar de manera continua esta herramienta.

La decisión se vio inclinada hacia el patrón MVP debido que al igual que con el lenguaje de programación anteriormente mencionado, ya existe conocimiento y experiencia previa con este patrón, lo que nos agiliza de igual manera el proceso de desarrollo de la aplicación.

3.5.3. FLUJO DE TRABAJO DE LA APLICACIÓN

En cuanto al flujo de trabajo de la aplicación, se ideó un flujo simple en el cual el usuario solo debe iniciar la aplicación y seleccionar el instructivo que requiera.

Para ejemplificarlo mejor podemos observar el flujo en la Figura 7, donde vemos un flujo totalmente lineal dentro de la aplicación.

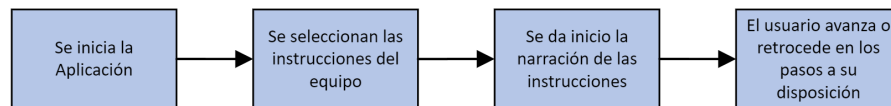


Figura 7: Diagrama de Flujo del uso de la Aplicación.

Fuente: Elaboración propia.

Aquí podemos ver el flujo de trabajo sin contar eventos como la configuración de los parámetros de la voz, debido que ese paso es a preferencia de cada usuario y no comprende el flujo de trabajo habitual, ya que la configuración de los parámetros de la voz no es algo que se vaya a realizar de manera frecuente.

3.5.4. CASOS DE USO

Este es un punto importante dentro del diseño de software. En este punto, se definen los Casos de Uso de la aplicación, identificando y ejemplificando las acciones que el usuario podrá realizar dentro de la misma.

A cada caso de uso se le asignó un **código**, para efectos de identificación de los mismos.

Nombre del caso de uso: Comenzar Narración

Código: CU-101

Actores: Usuario

Descripción: Proceso de comenzar la narración del audio generado a partir del texto procesado. El objetivo principal es entregar al usuario las instrucciones de mantenimiento de manera clara y natural, a través de la voz generada por el sistema.

Resumen: Reproducción de audio con instrucciones mediante voz generada por el sistema.

Tabla 1: Caso de Uso CU-101

Fuente: Elaboración Propia

Acción del Actor	Acción del Sistema
1. El Usuario solicita la narración de la voz generada presionando el botón de reproducir.	
	2. El Sistema reproduce la voz generada comenzando por el primer paso.
3. El Usuario escucha la narración de la voz generada y procede con la tarea.	

Nombre del caso de uso: Pausar/Reanudar Narración

Código: CU-102

Actores: Usuario

Descripción: Proceso de pausar o retomar la narración de la voz generada, brindando al usuario el control sobre la reproducción de la narración, permitiéndole detener esta y volver a reproducir la narración de las instrucciones en cualquier momento.

Resumen: Interacción entre el usuario y el sistema para controlar la reproducción de la voz generada.

Tabla 2: Caso de Uso CU-102

Fuente: Elaboración Propia

Acción del Actor	Acción del Sistema
1. El Usuario solicita pausar la narración de la voz generada presionando el botón de pausa.	
	2. El Sistema detiene la reproducción de la voz hasta que el usuario indique la reanudación.
3. El Usuario retoma la narración presionando el botón para reanudar.	
	4. El sistema reanuda la reproducción de la voz desde el punto en que se pausó.

Nombre del caso de uso: Modificar parámetros de la Voz

Código: CU-103

Actores: Usuario

Descripción: Proceso de modificar los diferentes parámetros de la voz generada por el sistema, como parte de la personalización del asistente en términos de características específicas como la velocidad o el volumen. El objetivo principal es permitir al usuario personalizar la voz de acuerdo con sus preferencias y así generar mayor entendimiento de parte de este.

Resumen: Modificación de parámetros en la generación de voz para la narración.

Tabla 3: Caso de Uso CU-103

Fuente: Elaboración Propia

Acción del Actor	Acción del Sistema
1. El Usuario presiona el botón de configuración para modificar la voz generada.	
	2. El Sistema presenta un menú con opciones para ajustar las características de la voz como el volumen, el tono y la cadencia de las palabras (velocidad).
3. El Usuario realiza los ajustes deseados en la voz.	
	4. El Sistema aplica las modificaciones seleccionadas a la voz generada.

Nombre del caso de uso: Avanzar al siguiente paso

Código: CU-104

Actores: Usuario

Descripción: Proceso en el que el Usuario avanza al siguiente paso dentro del protocolo de mantenimiento para continuar con su labor.

Resumen: Avanzar al siguiente paso para continuar con el proceso de mantenimiento.

Tabla 4: Caso de Uso CU-104

Fuente: Elaboración Propia

Acción del Actor	Acción del Sistema
1. El Usuario avanza al siguiente paso presionando el botón de siguiente paso.	
	2. El Sistema avanza al siguiente paso y lo narra automáticamente.

Nombre del caso de uso: Retroceder al paso anterior.

Código: CU-105

Actores: Usuario

Descripción: Proceso en el que el Usuario retrocede al paso anterior dentro del protocolo de mantenimiento según sus requerimientos.

Resumen: Retroceder al paso anterior según su necesidad.

Tabla 5: Caso de Uso CU-105

Fuente: Elaboración Propia

Acción del Actor	Acción del Sistema
1. El Usuario retrocede al paso anterior presionando el botón de paso anterior.	
	2. El Sistema retrocede al paso anterior y lo narra automáticamente.

Nombre del caso de uso: Repetir el paso actual.

Código: CU-106

Actores: Usuario

Descripción: Proceso en el que el Usuario repite el paso actual dentro del protocolo de mantenimiento según sus requerimientos.

Resumen: Repetir el paso actual según su necesidad.

Tabla 6: Caso de Uso CU-106

Fuente: Elaboración Propia

Acción del Actor	Acción del Sistema
1. El Usuario presiona el botón de repetir paso actual.	
	2. El Sistema comienza a narrar nuevamente el paso actual.

CAPÍTULO 4

VALIDACIÓN DE LA SOLUCIÓN

Llegados a este punto, se debe validar la solución propuesta a lo largo de este documento, por ende se necesita demostrar la validez de la misma.

Para esto se validarán distintos puntos con el objetivo de abarcar cada sector clave de la propuesta.

4.1. VALIDACIÓN DE FUNCIONALIDAD

Se validarán las funcionalidades de la aplicación, incluyendo la síntesis de voz, es decir, que estas funcionen de la manera prevista. Siendo este el punto principal, debido al enfoque de esta memoria y el área en donde se aborda la misma.

Para esto se realizarán pruebas unitarias para las funcionalidades principales en la aplicación.

4.2. VALIDACIÓN DE LA UX

Uno de los puntos importantes a validar es la UX, esta se validará mediante encuestas y entrevistas. Estas se realizarán directamente con los actores que utilizarían la aplicación para el propósito que fue diseñada.

Esto se realiza con el objetivo de obtener retroalimentación de parte de dichos actores, lo que nos permite mantener una mejora continua en el proceso de desarrollo de la aplicación. Todo esto gracias a que la validación de la UX será de forma iterativa, esto quiere decir que a medida que se vayan haciendo cambios, se vayan validando con los usuarios. Con esto nos aseguramos que al momento de entregar la aplicación final, la UX sea la esperada por el usuario y sea satisfactoria.

4.2.1. PLAN DE VALIDACIÓN UX

Para validar la UX, se realizará un plan de validación, es decir, una serie de pasos a seguir. Los pasos propuestos para el plan de validación son:

- **Entregar la aplicación a los usuarios:** En este paso, se hace entrega de la aplicación en su primera fase, con el objetivo de que la exploren, utilicen, analicen tanto su funcionalidad como su UI y se familiaricen con ella.

- **Entregar encuesta sobre la UX:** Luego de probada la aplicación, se le entregará a los usuarios una encuesta. Esta debe ser respondida para obtener la retroalimentación, de esta manera poder mejorar o cambiar los puntos negativos y mejorables destacados por los usuarios.

Para tener una idea, la encuesta se organizó en distintas secciones para su posterior análisis, de esta manera podemos ordenar los puntos que se evalúan, los cuales se dividen en Usabilidad, Navegación, Funcionalidad y Rendimiento.

Esto nos permitirá recopilar información importante para posteriores mejoras o cambios dentro del desarrollo de la aplicación.

- **Entregar encuesta sobre la UI:** Adicionalmente se entregará una encuesta donde se validará la UI, es decir, si el usuario la considera intuitiva, fácil de utilizar, llamativa, entre otras cosas.

Para ello, al igual que la encuesta anterior, se dividió en secciones para su análisis, las cuales son Diseño Visual, Claridad y Comprensión, Organización e Interactividad.

4.3. RESULTADO DE VALIDACIÓN DE LA UX

Una vez realizadas las encuestas correspondientes mostramos y analizamos los resultados de cada una. Con estos resultados podemos determinar los puntos débiles y de mejora dentro del diseño de la aplicación para las siguientes iteraciones.

En este caso, las encuestas se realizaron para la primera iteración para validar preliminarmente los puntos de mejora dentro del prototipo de la aplicación. De esta manera podemos tener una visión hacia donde dirigir la UI para mejorar la UX que nuestra aplicación final brindará.

Cabe destacar que estas encuestas se realizaron a los usuarios principales que harían uso de la aplicación para realizar las tareas para las cuales fue diseñada.

En los siguientes puntos se mostrarán y analizarán los resultados más relevantes de las encuestas, en los cuales se recopilaron las respuestas de 5 usuarios.

4.3.1. RESULTADOS Y ANÁLISIS DE ENCUESTA VALIDACIÓN UX

Dentro de la **Usabilidad**, un punto importante a analizar es la facilidad de uso de nuestra aplicación del lado del usuario, es por eso que se les pidió a dichos usuarios que la calificaran en una escala de 1 a 5.

En una escala de 1 a 5, ¿Cómo calificaría la facilidad de uso de la aplicación?

5 respuestas

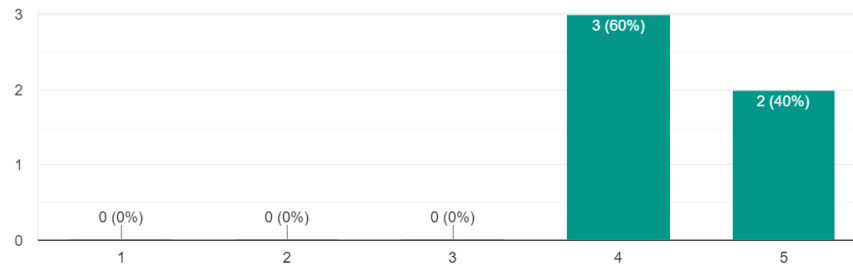


Figura 8: Respuestas de la pregunta N°1 Encuesta de validación de UX.
Fuente: Elaboración propia.

Dentro de estas respuestas presentadas en la Figura 8 podemos ver que los resultados son bastante buenos, considerando que las respuestas se encuentran sobre el punto medio de la calificación. Lo que nos indica que la aplicación tiene un buen recibimiento en cuanto a su facilidad de uso respecto a los usuarios.

Adicionalmente, para tener una idea de lo intuitiva que pueda ser la aplicación, se preguntó a los usuarios cuanto tiempo (aproximadamente) les tomó entender la aplicación.

En minutos, ¿Cuánto tiempo (aproximadamente) le tomó entender el funcionamiento de la aplicación?

5 respuestas

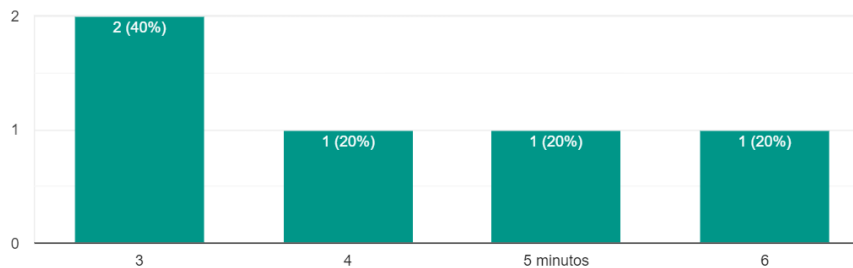


Figura 9: Respuestas de la pregunta N°2 Encuesta de validación de UX.
Fuente: Elaboración propia.

De estas respuestas (Figura 9) podemos rescatar el hecho de que a los usuarios de manera general no les tomó demasiado tiempo para entender el funcionamiento de la aplicación, esto debido a su baja complejidad a nivel de uso y de funcionalidades, gracias a que no es una aplicación especialmente compleja ni con demasiada información que el usuario deba procesar para entender como funciona.

Continuando con la **Navegación**, en esta sección se realizan preguntas enfocadas a lo intuitiva

que pueda ser la aplicación y a la facilidad de navegación dentro de la misma.

Principalmente se preguntó a los usuarios sobre la facilidad para encontrar las funcionalidades de la aplicación.

¿Encontró fácilmente las funcionalidades que estaba buscando?

5 respuestas

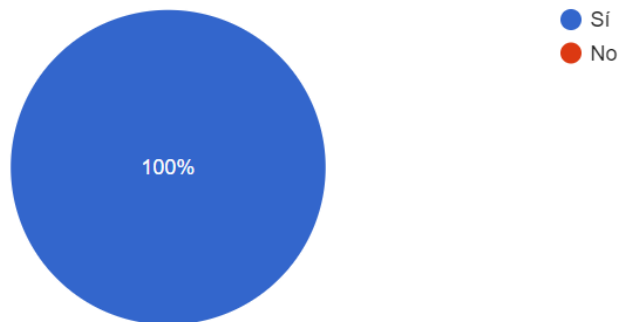


Figura 10: Respuestas de la pregunta N°3 Encuesta de validación de UX.

Fuente: Elaboración propia.

Lo que podemos observar en las respuestas de la Figura 10 es que la totalidad de los usuarios encontró fácilmente las funcionalidades, lo que nos da a entender que la navegación fue planificada y desarrollada de buena manera en este primer prototipo. Cabe destacar que este sería un escenario perfecto pero poco realista, debido a la cantidad baja de usuarios a los cuales se le realizó la encuesta. De todas maneras estos resultados son favorables para el desarrollo futuro de la aplicación.

En relación al punto siguiente, es decir la **Funcionalidad**, se le preguntó a los usuarios sobre las funcionalidades de la aplicación, específicamente si consideran alguna que se podría implementar en versiones futuras de la aplicación.

¿Considera que falte alguna funcionalidad que le gustaría ver en futuras versiones?

5 respuestas

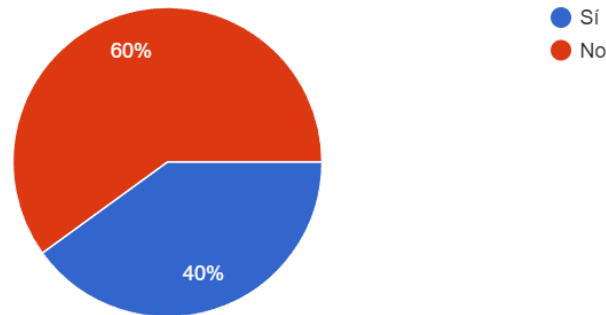


Figura 11: Respuestas de la pregunta N°4 Encuesta de validación de UX.
Fuente: Elaboración propia.

En la figura 11 podemos observar que 2 de los 5 usuarios consideraron que se podían agregar funcionalidades en versiones futuras de la aplicación. Esta pregunta se compone de 2 partes, en la que si la primera respuesta es "Sí", sugerir cuales podrían ser las funciones a implementar.

Si la respuesta anterior fue "Sí", sugiera alguna/s

2 respuestas

ajustar el tamaño de las letras

Poder agrandar el texto

Figura 12: Respuestas de la pregunta N°4 Encuesta de validación de UX.
Fuente: Elaboración propia.

Dentro de estas respuestas (Figura 12) escritas por ambos usuarios, nos encontramos con la misma sugerencia de parte de ambos, la cual sería implementar una función para realizar el ajuste del tamaño del texto dentro de la aplicación. Esto se podría considerar para la siguiente iteración, en la cual se implementaría en la fase de mejoras.

Entrando a la sección de **Rendimiento**, se realizan preguntas clave respecto al rendimiento de la aplicación, las cuales nos entregan información relevante sobre su desempeño en dispositivos.

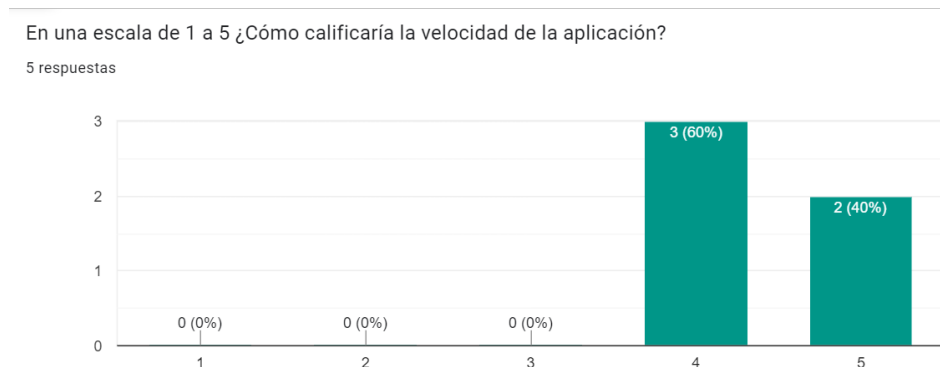


Figura 13: Respuestas de la pregunta N°5 Encuesta de validación de UX.
Fuente: Elaboración propia.

La Figura 13 nos muestra un resultado nuevamente positivo respecto a la velocidad de la aplicación, esto debido al igual que en anteriores casos, a la baja complejidad de la aplicación, por ende la utilización de recursos no es especialmente una carga para el dispositivo, resultando en una aplicación rápida a ojos del usuario, lo cual se debe seguir validando en futuras versiones de la aplicación a medida que se vayan desarrollando e implementando funciones.

¿La aplicación se ha bloqueado o detenido alguna vez mientras la usaba?

5 respuestas

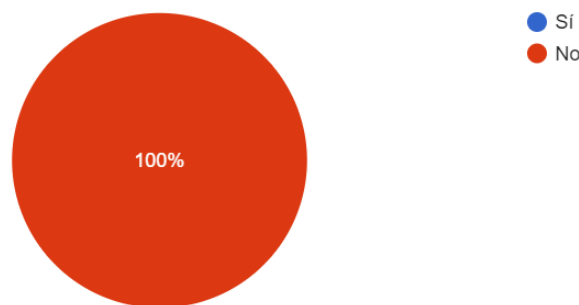


Figura 14: Respuestas de la pregunta N°6 Encuesta de validación de UX.
Fuente: Elaboración propia.

Finalmente, en la Figura 14, podemos observar resultados favorables, nuevamente apelando a la poca cantidad de usuarios de prueba, pero que sin embargo, nos presenta estos resulta-

dos favorables anteriormente mencionados relacionados a los posibles fallos de la aplicación durante su uso. Este punto siendo muy importante para próximas versiones, al igual que el punto anterior, se puede ver afectado por implementaciones nuevas en futuras versiones de la aplicación, por lo que es un foco de atención para trabajos futuros.

4.3.2. RESULTADOS Y ANÁLISIS DE ENCUESTA VALIDACIÓN UI

Dentro del **Diseño Visual**, lo importante a analizar, como su nombre lo indica, el diseño visual de la aplicación, iconos, textos, botones y diversos elementos que pueden mejorar y facilitar la interacción del usuario con la aplicación.

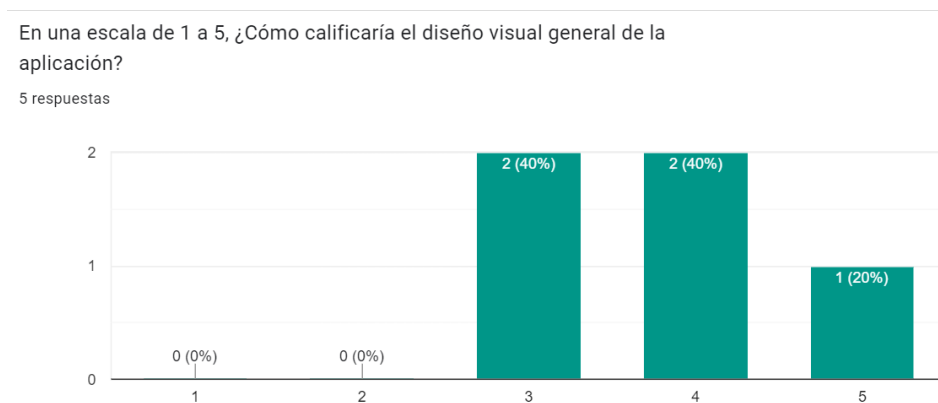


Figura 15: Respuestas de la pregunta N°1 Encuesta de validación de UI.

Fuente: Elaboración propia.

Como podemos observar en la Figura 15, los usuarios tuvieron una respuesta en general positiva sobre el diseño de la aplicación en el primer contacto, la idea sería mejorar el nivel de calificación de parte de los usuarios en futuras versiones de la aplicación, hay que tener en consideración que este fue el primer contacto y puede ir variando en las próximas iteraciones y versiones de la aplicación.

¿El uso de imágenes e íconos le parecería útil y claro?

5 respuestas

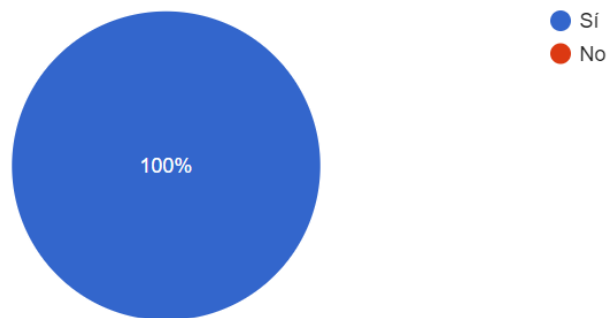


Figura 16: Respuestas de la pregunta N°2 Encuesta de validación de UI.
Fuente: Elaboración propia.

Los resultados presentados en la Figura 16 nos indican un posible punto de mejora para atacar en las siguientes iteraciones, implementando el uso de iconos e imágenes para mejorar el diseño visual de la aplicación.

Entrando en lo que es la **Claridad y Comprensión**, se quiere validar sobre como la información que contiene la aplicación llega al usuario, es decir, que este comprenda e identifique de manera sencilla y rápida los botones, textos y como interactuar con ellos.

¿los textos en la aplicación (botones, etiquetas, instrucciones) son claros y fáciles de entender?

5 respuestas

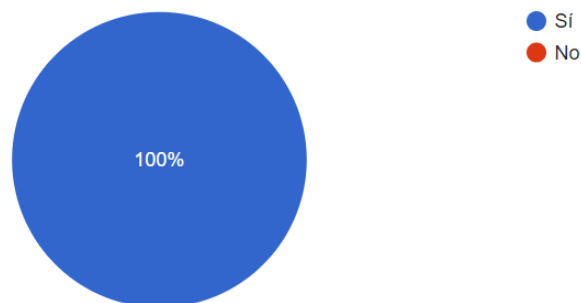


Figura 17: Respuestas de la pregunta N°3 Encuesta de validación de UI.
Fuente: Elaboración propia.

En este punto, en la Figura 17 se presentan los resultados que nos indican que en general los elementos que se muestran en pantalla dentro de la aplicación son claros y comprensibles

por el usuario, lo cual es principalmente importante para el entendimiento y facilitación del uso de la aplicación, siendo el diseño visual un punto a mejorar, pero que la claridad ya se logró.

Finalmente en la sección de **Organización e Interactividad**, evaluamos los puntos en donde el usuario interactúa con la aplicación y su visión respecto a la organización de los distintos elementos presentados en pantalla.

En una escala de 1 a 5, ¿Cómo calificaría la disposición de los elementos en la pantalla?

5 respuestas

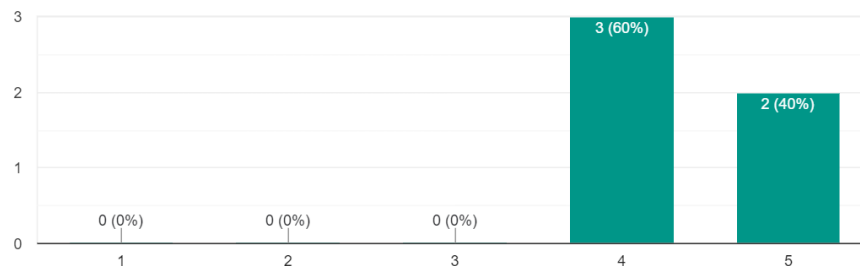


Figura 18: Respuestas de la pregunta N°4 Encuesta de validación de UI.
Fuente: Elaboración propia.

En la Figura 18, los resultados nos indican que se diseñó una buena disposición de los elementos en pantalla, tomando en cuenta que sigue siendo un punto de posibles mejoras, pero que se consiguió un resultado bastante favorable para las próximas iteraciones.

¿los botones y elementos interactivos responden adecuadamente cuando los utilizas?

5 respuestas

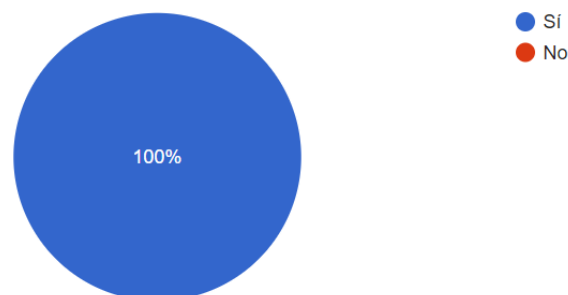


Figura 19: Respuestas de la pregunta N°5 Encuesta de validación de UI.
Fuente: Elaboración propia.

Como podemos observar en la Figura 19, los resultados de esta pregunta, al igual que en las anteriores, obviando el hecho de la baja cantidad de respuestas, nos entrega un resultado nuevamente favorable, debido que el trabajo de desarrollo lógico de los elementos interactivos actúan de manera correcta con las interacciones del usuario, en este caso, los botones responden de manera adecuada al presionarlos y cumplen su función con total éxito.

4.3.3. CONCLUSIÓN DE LOS RESULTADOS GENERALES

Con estos resultados podemos identificar los puntos de mejora para las próximas iteraciones, lo que se incluiría dentro de los trabajos futuros para el desarrollo de la aplicación de manera óptima.

El objetivo es continuar realizando este tipo de encuestas por cada iteración dentro del desarrollo para validar si las mejoras y cambios realizados según la última encuesta realizada nos acercan al objetivo deseado e identificando nuevas mejoras o cambios, generando así un continuo trabajo en el mejoramiento de la UX a medida que se desarrolla la aplicación final.

Con esta primera instancia de validación, podríamos realizar una planificación de manera efectiva para la próxima iteración y su desarrollo, atacando directamente los puntos de mejora y nuevas funciones que se puedan implementar, lo que entraría dentro de los trabajos futuros.

4.4. VALIDACIÓN DE OPTIMIZACIÓN

Otro punto importante a validar dentro de esta memoria es la optimización del proceso de mantenimiento utilizando la solución propuesta.

4.4.1. PLAN DE VALIDACIÓN DE OPTIMIZACIÓN

Para validar este punto se compararán dos versiones del procedimiento en distintos usuarios, una sin la aplicación y una con la aplicación. Para esto se le hará entrega de la aplicación final a los usuarios para que la utilicen en el procedimiento. De esta manera podemos comparar los tiempos que toma realizar esta tarea con la solución propuesta y medir si efectivamente se optimiza el proceso, se mantiene o se obstaculiza en temas de tiempos.

CAPÍTULO 5

CONCLUSIONES

Dentro de este proyecto podemos realizar y tomar conclusiones en sus distintas etapas y en sus diversas áreas de su desarrollo.

5.1. LOGROS

De este proyecto podemos destacar el logro de levantar una aplicación móvil funcional cumpliendo las expectativas desde cero, es decir, comenzando por la identificación de un problema, la elaboración de una idea y finalmente la propuesta de solución. Todo esto aplicando conocimientos y experiencias previas adquiridas durante el paso por la carrera y durante el desarrollo del proyecto, llevando a cabo las buenas practicas y utilizando las diversas herramientas que se nos han otorgado para ser buenos profesionales.

5.2. LECCIONES APRENDIDAS

Durante el desarrollo de este proyecto podemos destacar algunas lecciones aprendidas, las cuales se deben tomar en cuenta para futuros proyectos, de esta manera mejorar continuamente como profesional lo que nos puede asegurar levantar proyectos de manera mas efectiva.

5.2.1. EXPERIENCIA EN EL DESARROLLO

Durante el desarrollo de este proyecto se demostró que una buena base de conocimientos y experiencia nos permite ahorrar tiempo en temas de desarrollo en sí. Pudiendo comenzar con ello desde principios del proyecto.

5.2.2. PLANIFICACIÓN DETALLADA

Una planificación detallada en la fase inicial del proyecto habría mitigado algunos desafíos en el despliegue de la aplicación e incluso poder asegurar una mejor implementación de la misma.

5.3. LIMITACIONES Y ÁREAS DE MEJORA

Cabe destacar además, que en el transcurso del proyecto nos encontramos con algunas limitaciones al momento del desarrollo del mismo, las cuales afectan de manera directa al proyecto y a su implementación.

5.3.1. LIMITACIONES

Principalmente las limitaciones de este proyecto son de carácter técnico, debido a los recursos de hardware para el desarrollo de la aplicación y para sus pruebas en dispositivos móviles.

Otra limitación del proyecto es de carácter académico. Debido a la carga académica que contemplaban ambos semestres, se desviaba en parte la atención y la elaboración de este proyecto, por lo que se dificultaron algunas etapas del desarrollo.

5.3.2. FUTURAS MEJORAS

Desarrollos futuros podrían abordar la optimización del consumo de batería y de otros recursos de hardware, además de la expansión de características y mejora exhaustiva de la aplicación.

5.4. IMPACTO Y RELEVANCIA

Este proyecto cuenta con la generación de un impacto en diversos usuarios, además de tener cierta relevancia en el espacio para el cual fue ideado.

5.4.1. IMPACTO

Este proyecto sugiere un impacto potencial en la mejora de la experiencia de los usuarios, sobre todo con aquellos que cuentan con discapacidades visuales, lo que mejora su experiencia en aplicaciones móviles.

5.4.2. RELEVANCIA EN LA INDUSTRIA

Esta solución destaca la relevancia de incluir e integrar tecnologías y herramientas de accesibilidad en el desarrollo de aplicaciones móviles y software en general.

5.5. PERSPECTIVAS FUTURAS

Tomando en cuenta la relevancia el proyecto se puede optar por continuar en un futuro expandiendo y mejorando la solución.

5.5.1. INVESTIGACIÓN ADICIONAL

Futuras investigaciones podrían explorar la opción de aplicar técnicas de aprendizaje profundo para mejorar la precisión de la voz y adecuarla a los distintos escenarios en los cuales se podría desempeñar.

5.5.2. EXPANSIÓN DE PLATAFORMAS

Se puede también considerar la expansión de esta aplicación a diferentes plataformas y dispositivos, un ejemplo claro sería el hecho de expandirla a la plataforma de iOS. Incluso podría expandirse a los relojes inteligentes en una versión más portátil de la aplicación.

5.6. REFLEXIÓN FINAL

Para finalizar, quiero incluir mi reflexión final y personal sobre el desarrollo de este proyecto, los desafíos y la experiencia ganada gracias a este.

5.6.1. DESAFÍOS SUPERADOS

Reflexionando sobre los desafíos superados durante el proyecto, se destaca la importancia de ser perseverante en el desarrollo de software para poder levantar proyectos y llevar a algo práctico las ideas para solucionar distintas problemáticas que podemos encontrar en el mundo.

5.6.2. EXPERIENCIA PERSONAL

Personalmente considero esta experiencia enriquecedora, proporcionándome aprendizajes significativos y una experiencia mas cercana a lo que sería el mundo laboral real. Entregándome conocimientos y herramientas para utilizar en mi futuro como profesional.

REFERENCIAS BIBLIOGRÁFICAS

- [Achour *et al.*, 2023] Achour, Gabriel and Salunke, Oojas and Payan, Alexia P and Harrison, Evan and Sahbani, Chiheb and Carannante, Giuseppina and Ditzler, Gregory and Bouaynaya, Nidhal and others (2023). Review of automatic speech recognition methodologies. Technical report, United States. Department of Transportation. Federal Aviation Administration
- [Allen *et al.*, 1987] Allen, J., Hunnicutt, M. S., Klatt, D. H., Armstrong, R. C., y Pisoni, D. B. (1987). *From text to speech: The MITalk system*. Cambridge University Press.
- [Anderson, 2012] Anderson, C. (2012). The model-view-viewmodel (mvvm) design pattern. En *Pro Business Applications with Silverlight 5*, pp. 461–499. Springer.
- [Brahler, 2010] Brahler, S. (2010). Analysis of the android architecture. *Karlsruhe institute for technology*, 7(8).
- [Bucanek, 2009] Bucanek, J. (2009). Model-view-controller pattern. *Learn Objective-C for Java Developers*, pp. 353–402.
- [Chalamandaris *et al.*, 2010] Chalamandaris, A., Karabetsos, S., Tsiakoulis, P., y Raptis, S. (2010). A unit selection text-to-speech synthesis system optimized for use with screen readers. *IEEE Transactions on Consumer Electronics*, 56(3):1890–1897.
- [Chicaiza Ayala, 1996] Chicaiza Ayala, J. S. (1996). Síntesis de voz utilizando la reproducción de fonemas y la composición de formantes. B.S. thesis, Quito: EPN, 1996.
- [COLOMBO, 2015] COLOMBO, G. (2015). Experience and comparison between native and hybrid development approaches for mobile devices.
- [Garg y Baliyan, 2021] Garg, S. y Baliyan, N. (2021). Comparative analysis of android and ios from security viewpoint. *Computer Science Review*, 40:100372.
- [Gasimov *et al.*, 2010] Gasimov, A., Tan, C.-H., Phang, C. W., y Sutanto, J. (2010). Visiting mobile application development: What, how and where. En *2010 Ninth International Conference on Mobile Business and 2010 Ninth Global Mobility Roundtable (ICMB-GMR)*, pp. 74–81. IEEE.
- [Gaudio, 2010] Gaudio, V. (2010). Mvvm: Model-view-viewmodel. En *Foundation Expression Blend 4 with Silverlight*, pp. 341–367. Springer.
- [Gunasekera, 2012] Gunasekera, S. (2012). Android architecture. En *Android Apps Security*, pp. 1–12. Springer.
- [Hyun y Marsden, 2023] Hyun, H. y Marsden, J. (2023). The importance of user experience in brand experience.

- [Islam *et al.*, 2010] Islam, R., Islam, R., y Mazumder, T. (2010). Mobile application and its global impact. *International Journal of Engineering & Technology*, 10(6):72–78.
- [Khan y Chitode, 2016] Khan, R. A. y Chitode, J. S. (2016). Concatenative speech synthesis: A review. *International Journal of Computer Applications*, 136(3):1–6.
- [Liakin *et al.*, 2017] Liakin, D., Cardoso, W., y Liakina, N. (2017). The pedagogical use of mobile speech synthesis (tts): Focus on french liaison. *Computer Assisted Language Learning*, 30(3-4):325–342.
- [Lin, 1991] Lin, Q. (1991). Speech production theory and articulatory speech synthesis. *The Journal of the Acoustical Society of America*, 90(4):2203–2203.
- [Macon, 1996] Macon, M. W. (1996). *Speech synthesis based on sinusoidal modeling*. Georgia Institute of Technology.
- [Malik, 1979] Malik, M. A. K. (1979). Reliable preventive maintenance scheduling. *AIIE transactions*, 11(3):221–228.
- [Marcus, 2002] Marcus, A. (2002). Dare we define user-interface design? *interactions*, 9(5):19–24.
- [McAulay y Quatieri, 1986] McAulay, R. y Quatieri, T. (1986). Speech analysis/synthesis based on a sinusoidal representation. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 34(4):744–754.
- [Moser y Moser, 2012] Moser, C. y Moser, C. (2012). *User experience design*. Springer.
- [Novac *et al.*, 2017] Novac, O. C., Novac, M., Gordan, C., Berczes, T., y Bujdosó, G. (2017). Comparative study of google android, apple ios and microsoft windows phone mobile operating systems. En *2017 14th international conference on engineering of modern electric systems (EMES)*, pp. 154–159. IEEE.
- [Oke y Charles-Owaba, 2006] Oke, S. y Charles-Owaba, O. (2006). An approach for evaluating preventive maintenance scheduling cost. *International Journal of Quality & Reliability Management*, 23(7):847–879.
- [2006] Palo, Pertti and others (2006). A review of articulatory speech synthesis. *Master's thesis, Helsinki University of Technology, Department of Electrical and Communications Engineering*.
- [Pau *et al.*, 2010] Pau, V. C., Mihailescu, M. I., y Stanescu, O. (2010). Model view presenter design pattern. *Journal of Computer Science & Control Systems*, 3(1).
- [Phongtraychack *et al.*, 2018] Phongtraychack, Anachack and Dolgaya, Darya and others (2018). Evolution of mobile applications.
- [Sharan y Sharan, 2015] Sharan, K. y Sharan, K. (2015). Model-view-controller pattern. *Learn JavaFX 8: Building User Experience and Interfaces with Java 8*, pp. 419–434.

- [Singh, 2014] Singh, R. (2014). An overview of android operating system and its security. *int. journal of Engineering Research and Applications*, 4(2):519–521.
- [Stan y Lőrincz, 2021] Stan, A. y Lőrincz, B. (2021). Generating the voice of the interactive virtual assistant. En *Virtual Assistant*. IntechOpen.
- [Studio, 2017] Studio, A. (2017). Android studio. *The Official IDE for Android*, 93.
- [Thimbleby, 1990] Thimbleby, H. (1990). *User interface design*. ACM.
- [Valipour et al., 2009] Valipour, M. H., AmirZafari, B., Maleki, K. N., y Daneshpour, N. (2009). A brief survey of software architecture concepts and service oriented architecture. En *2009 2nd IEEE International Conference on Computer Science and Information Technology*, pp. 34–38. IEEE.
- [Wentk, 2014] Wentk, R. (2014). *Xcode 5 developer reference*. John Wiley & Sons.
- [Wu, 2011] Wu, S. (2011). Preventive maintenance models: A review. *Replacement models with minimal repair*, pp. 129–140.
- [Zhang y Luo, 2010] Zhang, Y. y Luo, Y. (2010). An architecture and implement model for model-view-presenter pattern. En *2010 3rd international conference on computer science and information technology*, volumen 8, pp. 532–536. IEEE.