UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

Desarrollo de una aplicación móvil para la educación digital "Educatronicapp"

TESIS

Que para obtener el título de

Ingeniero en Computación

P R E S E N T A

Edgar Hernández Hernández

DIRECTOR(A) DE TESIS

Dra. Josefina Bárcenas López

Ciudad Universitaria, Cd. Mx., 2024

"La educación es el arma más poderosa que puedes usar para cambiar el mundo."
Nelson Mandela

AGRADECIMIENTOS

A mi directora Dra. Josefina Bárcenas López por ser una guía de enseñanzas, oportunidades y consejos mostrados a lo largo de este proyecto, su paciencia, confianza, experiencia y dedicación han sido fundamentales para cumplir mi objetivo, sus consejos, comentarios y sugerencias fueron invaluables para superar desafíos que encontré en el camino.

Al M. en C Víctor Hugo García Ortega por acompañarme y orientarme durante el desarrollo de este proyecto, gracias por las horas de trabajo y por compartir su conocimiento, su experiencia fue fundamental para llevar mi proyecto a un nivel superior.

Al Dr. Enrique Ruiz Velasco Sánchez, por permitirme participar en el proyecto "Educatrónica", ha sido una gran experiencia que me ha permitido aprender y crecer personalmente, su compromiso con la mejora de la educación y su visión inspiradora ha sido una fuente constante de inspiración para mí.

A la Universidad Nacional Autónoma de México, por darme la formación académica, sus principios y valores estarán presentes a lo largo de mi vida, es un orgullo para mi ser azul y oro.

Agradezco a la Dirección General de Asunto del Personal Académico (DGAPA) de la Universidad Nacional Autónoma de México (UNAM) por apoyar el proyecto PAPIITIT400222, quien me proporcionó el apoyo para el desarrollo de este trabajo.

Agradezco al Instituto de Ciencias Aplicadas y Tecnología (ICAT) por permitirme conocer a grandes personas que me impulsaron a mi desarrollo.

DEDICATORIA

A mis padres Jovita Hernández Collado y Orlando Hernández Moran, por permitirme llegar a esta etapa de mi vida, por todo su apoyo incondicional, por sus valores, por el amor, por la confianza que me han brindado durante mi educación. Les agradezco cada uno de los esfuerzos que han realizado, siempre serán un ejemplo por seguir y siempre estaré orgulloso de ustedes.

A mi hermana Jaqueline Hernández Hernández por ser parte de mi vida, porque gracias a su amor, comprensión y respeto me ha impulsado a ser una mejor persona, siempre contaras con mi apoyo incondicional y estoy orgulloso de ti.

A mi familia, a ustedes tíos, tías, abuelos, sobrinos y sobrinas por ser un motor de vida, les dedico este logro con todo mi amor, cada lección aprendida son un reflejo de su influencia en mi vida.

A mis primos, por su apoyo incondicional, son un ejemplo por seguir, gracias por la hermandad que tenemos.

Resumen

El desarrollo de este documento se centra en la creación de una aplicación móvil multiplataforma (Android e IOS) desarrollada en Expo React Native, esta aplicación lleva por nombre “Educatrónicapp”, su objetivo es hacer que sus usuarios se introduzcan en los conceptos básicos de la programación haciendo uso de un lenguaje natural definido. Además, esta aplicación móvil forma parte del proyecto “Laboratorio de Educatrónica” a cargo del Doctor Enrique Ruiz Velasco Sánchez, la Doctora Josefina Bárcenas López y el Maestro en Ciencias Víctor Hugo García Ortega; proyecto en el que se mezclan distintos aspectos educativos como la programación, la electrónica y la robótica pedagógica, por lo que la aplicación “Educatrónicapp” cuenta con una comunicación mediante tonos DTMF para interactuar con una interfaz electrónica y un robot pedagógico (elevador).

“Educatrónicapp” tiene la premisa de estar orientada a la inclusividad para ser usada por usuarios con algún tipo de discapacidad motriz, por lo que se incluyeron tecnologías de inteligencia artificial como el reconocimiento de voz usando un API REST de nombre Speech to Text perteneciente a Open AI.

Abstract

The development of this document focuses on the creation of a multi-platform mobile application (Android and IOS) developed in Expo React Native, this application is called “Educatrónicapp”, its objective is to introduce its users to the basic concepts of programming using a defined natural language. In addition, this mobile application is part of the “Educatrónica Laboratory” project led by Doctor Enrique Ruiz Velasco Sánchez, Doctor Josefina Bárcenas López and Master of Science Víctor Hugo García Ortega; project in which different educational aspects are mixed such as programming, electronics and pedagogical robotics, so the “Educatrónicapp” application has communication using DTMF tones to interact with an electronic interface and a pedagogical robot (elevator).

“Educatrónicapp” has the premise of being oriented towards inclusivity to be used by users with some type of motor disability, which is why artificial intelligence technologies such as voice recognition are included using a REST API called Speech to Text belonging to Open AI.

ÍNDICE

Contenido

ÍNDICE DE FIGURAS

INTRODUCCIÓN

El presente documento describe el desarrollo de una aplicación móvil para la educación digital a la que se le ha denominado “Educatrónicapp”. Esta propuesta surge dentro del proyecto PAPIIT “Laboratorio de Educatrónica” que se desarrolla en el grupo académico de “Telemática para la Educación” perteneciente al Instituto de Ciencias Aplicadas y Tecnología en la Universidad Nacional Autónoma de México (ICAT-UNAM), busca crear nuevas tecnologías que se adapten a las necesidades educativas para mejorar la educación. Esta aplicación móvil está orientada a introducir a sus usuarios a obtener conocimientos básicos de programación, con la premisa de estar orientada a usuarios con algún tipo de discapacidad motriz por lo que se implementó el diseño incorporando tecnologías adecuadas para la usabilidad de esta población.

Situación Actual:

“Laboratorio de Educatrónica” es un proyecto a cargo del Doctor Enrique Ruiz Velasco Sánchez (Investigador de la UNAM, IISUE), su sistema está basado en la mezcla de la robótica pedagógica, la electrónica, la programación y las aplicaciones móviles. Básicamente se tiene un robot pedagógico conectado a una interfaz que detecta distintos tonos y hace reaccionar al robot, dichos sonidos son emitidos por una aplicación móvil (Educatrónica) donde se programa el comportamiento del robot.

La usabilidad de la aplicación móvil que se requiere en “Educatrónica” presenta desafíos significativos para personas con discapacidad motriz, lo que genera dificultad para el uso de la aplicación hacia este grupo de usuarios, como consecuencia genera una exclusión, perdida de oportunidades y falta de recursos para mejorar la educación.

Se presentan algunos fallos de secuencia dentro de “Educatrónica” que deben ser corregidos. Para algunos usuarios la interfaz que se utiliza en el App no es lo suficientemente dinámica.

Educatrónica: Es una aplicación móvil gratuita la cual usa un lenguaje natural para la programación y manipulación de un robot pedagógico (elevador), sus comandos están definidos para orientar a los usuarios a construir la estructura de un lenguaje de programación a partir de su propio lenguaje natural.

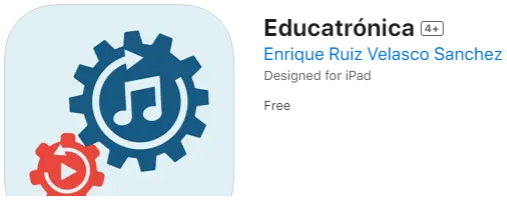


Figura 1. Logotipo Educatrónica. Obtenida de AppStore.

Figura 2. Encuentro Educatrónica. Obtenida de Francisco Cabiedes Contreras.

Algunas características de la aplicación son:

- Permite la elaboración y ejecución de un programa para el control de un elevador.
- Contiene un lenguaje natural definido.
- Permite ver la simulación de la programación.
- Versión nativa para Android e IOS.
- Contiene algunos errores relevantes relacionados con la ejecución de un programa o comportamiento del elevador.

Rompí: Es una aplicación móvil gratuita el cual usa un lenguaje natural para la programación y manipulación de un robot pedagógico (carrusel), sus comandos están definidos para orientar a los usuarios a construir la estructura de un lenguaje de programación apartir de su propio lenguaje natural.

Figura 3. Logotipo Rompí. Obtenida de AppStore.

Algunas características de la aplicación son:

- Permite la elaboración y ejecución de un programa para el control de un carrusel.
- Contiene un lenguaje natural definido.
- Versión nativa para Android y IOS.

CAPITULO 1
LA IMPORTANCIA DE LAS TECNOLOGÍAS DE LA INFORMACIÓN EN LA EDUCACIÓN INCLUSIVA

1.1 ANTECEDENTES

La educación digital según el Doctor en Educación y Máster en Educación Abierta, Andrés Núñez Álvarez de Florida State University, atribuye el concepto:

> *"Se entiende por Educación Digital al uso de tecnologías digitales que tienen como objetivo la adquisición de competencias y habilidades para aprender a aprender, estas tecnologías se enfocan a una educación presencial o a distancia durante un proceso de formación permanente."* (Riande Juárez, N.A & Derechos Humanos, s. f)

Algunas características que menciona dentro de su definición son:

- La educación digital no contiene restricciones de tiempo ni de espacio, es permanente, se encuentra disponible a toda hora, lugar y en todo momento.
- Cuando se hace uso de educación digital desaparecen las diferencias entre la educación presencial y a distancia.
- La educación digital nos proporciona un cambio de paradigmas.

El concepto de educación digital ha tenido un proceso evolutivo conforme al avance a nuevas tecnologías digitales, para algunos la educación digital no es un concepto como tal sino más bien un conjunto de nuevas habilidades para impartir conocimientos con el uso de herramientas digitales con las que interactúan los seres humanos y a su vez usarlas como un instrumento tecnológico para aprender. Es decir, no solo es la integración de dispositivos y herramientas digitales si no que supone una transformación educativa que mejore el aprendizaje.

Para la Organización de las Naciones Unidas para la Educación, la Ciencia y la Cultura (UNESCO), uno de los principales objetivos de la educación digital es proporcionar por igual las oportunidades para acceder a la educación y cubrir las deficiencias de la educación presencial debido a la falta de recursos económicos y académicos, este acceso debe ser a todas y todos por igual incluyendo a las personas con dificultades geográficas, económicas o físicas que permita a estas personas enfrentar los retos de su sociedad, respetando sus intereses personales, necesidades y condiciones.

En la sociedad actual el desarrollo científico la ciencia y la innovación han venido evolucionando de forma progresiva y de forma dinámica, con ello la sociedad ha tenido que adaptar los cambios tecnológicos hacia el futuro. La UNESCO piensa que estas tecnologías pueden mejorar el impacto en las oportunidades de enseñanza/aprendizaje facilitando una formación para desarrollar habilidades para la educación.

Las TIC (Tecnologías de la Información y Comunicación) son soluciones con recursos y herramientas innovadoras para ordenar y eficientar la información y la comunicación, través de elementos tecnológicos como: teléfonos, computadoras, software e internet. Al hablar de Tecnologías de la Información la educación, la comunicación, la política, la vivienda, o la salud son algunos escenarios que integran Tecnologías de la información y Comunicación aplicando recursos tecnológicos.

Las Tecnologías de la Información y Comunicación (TIC) se han incorporado a la educación con herramientas digitales, didácticas y tecnológicas, con el fin de mejorar los procesos de enseñanza y aprendizaje, muchas de estas tecnologías están a nuestro alcance y forman parte de nuestro día a día, por ejemplo: celulares, computadoras, tabletas e internet. Con esto se ofrecen nuevas alternativas para la accesibilidad.

Desafíos que intenta resolver la educación digital en la educación:

1- La era digital no se creó de la noche a la mañana: Con todo lo sucedido con el COVID-19 se debe saber que las conferencias, seminarios, exámenes y clases en línea no necesariamente constituyen una educación digital. Es un cambio generacional que lleva su tiempo.
2- No siempre se puede implementar de forma independiente: Cabe destacar que no todos los usuarios pueden poner en práctica educación digital ya que algunos consideran la necesidad de expertos en tecnología que contribuyan una solución de herramientas y soluciones tecnológicas.
3- La transformación digital tiene riesgos: Una capacitación deficiente puede dejar a los docentes sin una orientación de cómo crear contenido didáctico apropiado.

La educación digital es ya una alternativa al proceso educativo tradicional, con el surgimiento de internet se ha logrado hacer a la educación más interactiva y práctica, al no limitarse solo a libros y aulas físicas si no que ha creado una fusión con la tecnología, el aprendizaje y el contenido digital.

LAS APLICACIONES MÓVILES COMO ALIADAS DE LA EDUCACIÓN

Las nuevas tecnologías, han sido un factor importante en el aspecto laboral para reducir las situaciones de dependencia e incrementar la autonomía de las personas, son aliadas en la mejora de la calidad de vida de las personas con discapacidad.

El uso pedagógico de dispositivos móviles se denomina aprendizaje móvil (*m-learning*) y consiste en la adquisición de conocimiento por medio de alguna tecnología de cómputo móvil, con esto se permite acceder a los contenidos digitales de forma más natural, brindando la oportunidad de mejorar las condiciones de los servicios de educación que se ofrecen, agilizar los procesos, generar comunidad y sobre todo innovar. (Richard E. Mayer, 2020)

Lo que hoy en día más se hace en el entorno de aprendizaje móvil son el uso de aplicaciones móviles, utilizadas en dispositivos móviles y diseñadas para ofrecer una solución en las áreas de conocimiento como el sector educativo incluyendo tecnologías para la educación en línea, comunicación, y control. Estás aplicaciones no solo agregan un valor más a la educación, también brindan la oportunidad de mejorar la interacción con una comunidad.

Con la situación del COVID-19, las aplicaciones educativas fueron la llave para la educación a distancia, esto no represento problema alguno para los jóvenes por la naturaleza con la que conviven con dispositivos móviles. Esto ofrece nuevas alternativas para la accesibilidad.

Lo característico en la educación digital es que el proceso formativo es a través de un entorno virtual donde se produce una interacción.

Actualmente contamos con aplicaciones móviles para el sector educativo que incluyen tecnologías para la educación en línea, comunicación, y control. Estás aplicaciones no solo agregan un valor más a la educación, también brindan la oportunidad de mejorar la interacción con una comunidad. En 2020, el COVID-19 puso en apuros a la educación digital, tuvo que acelerarse para adaptarse a los nuevos sistemas o planes educativos que se estaban implementando, las aplicaciones educativas fueron la llave para la educación a distancia, esto no presento problema alguno para los jóvenes por la naturaleza con la que conviven con dispositivos móviles, haciendo uso de aplicaciones como Zoom, Meet o Teems. (NACIONES UNIDAS, 2020)

No obstante, la educación digital no depende de un escenario ni de la mejor tecnología, depende de aprovechar y adaptar las tecnologías que faciliten el camino ante el aprendizaje. Cuando se logra juntar la educación digital con el uso de aplicaciones móviles, se generan soluciones para mejorar y proyectar la educación de forma tecnológica, eficiente y accesible. Algunas de las ventajas de la educación digital son:

Aprendizaje individualizado	Actualización digital
Información ilimitada	Acceso remoto
Aulas inteligentes	Facilidad para compartir contenido
Constante innovación	Inclusión

LA EDUCACIÓN CON LA INCLUSIÓN EN APLICACIONES MÓVILES

El derecho a la Educación es mundialmente reconocido desde 1948 con la Declaración Universal de Derechos Humanos, estableciendo que todos tengan acceso a la educación sin importar su raza, edad, posición económica o discapacidad.

En México la Educación es un derecho humano reconocido en el artículo 3° de la Constitución Política de los Estados Unidos Mexicanos que señala "todo individuo tiene derecho a recibir educación". (Schmelkes del Valle, 2014)

Por otra parte, según la UNICEF, la "Educación Inclusiva es un plan que tiene como objetivo que todos ya sea niños o adolescentes tengan acceso igualitario al aprendizaje en todos los contextos." (UNICEF, s. f)

Finalmente, con estas definiciones es como se da pie a la integración de aplicaciones móviles inclusivas permitiendo adaptar y simplificar funcionalidades con ayuda de un dispositivo móvil conforme a sus características, necesidades o discapacidades.

Con el sin fin de aplicaciones móviles que existen en los mercados digitales y ahora se puede ayudar a mejorar el estilo de vida de personas con alguna discapacidad o deficiencia. Ahora es más fácil resolver una dificultad con un dispositivo móvil, por ejemplo, guiar a un invidente o ayudar a una persona con problemas de movilidad o comunicación.

La Educación inclusiva facilita el desarrollo humano, permitiendo el acceso a todas las personas a la educación y crear valores a una sociedad más justa y tolerante.

LA DISCAPACIDAD MOTRIZ

La discapacidad se define como un fenómeno que abarca las deficiencias, limitaciones y restricciones que refleja una interacción entre las características del organismo humano y de su sociedad, en la que se presenta una limitación o impedimento de realizar una actividad dentro del margen que se considera normal para el ser humano. (Infobae, 2021)

Existen distintos tipos de discapacidad tales como discapacidad motriz, sensorial e intelectual. La discapacidad física motora (movimiento) surge cuando una persona presenta un estado físico que le impide de forma parcial o permanente el moverse a plenitud y desenvolverse de manera convencional, esto deriva como consecuencia una dificultad específica para manipular objetos o acceder a diferentes espacios. (Observatorio de la Discapacidad Física, 2016)

Algunos tipos de discapacidad motriz que se presentan en personas son:

- Lesión medular: Causan una parálisis parcial o total, afectando la movilidad.
- Amputación: La pérdida de alguna extremidad como un brazo, pierna, dedos.
- Ataxia: Afecta la coordinación y el equilibrio para la elaboración de actividades cotidianas.
- Parálisis cerebral: Crea un trastornó del movimiento y la postura afectando la movilidad, fuerza muscular y coordinación.
- Distrofia muscular: Debilitan los músculos con el tiempo lo que va provocando la perdida de movilidad total.

CAPITULO 2
EL SOFTWARE Y LOS MÓVILES

2.1 INGENIERIA DE SOFTWARE

La ingeniería de software es una disciplina o área de la Informática o Ciencias de la Computación que integra y ofrece procesos, herramientas o métodos para el desarrollo y mantenimiento de software.

"Es la disciplina tecnológica y de gestión que concierne a la producción y el mantenimiento sistemático de productos de software desarrollados y modificados dentro de plazos estipulados y costes." (R. Fairley, 1985)

"Ingeniería de Software es el estudio de los principios y metodologías para desarrollo y mantenimiento de sistemas de software". (Zelkovitz, 1978)

La ingeniería de software trata con áreas muy diversas de las ciencias de la computación, informática o la creación de sistemas operativos, su campo de aplicación puede ser áreas como la investigación científica, negocios o desarrollo inteligente. Realiza un análisis conforme a la creación de software y pruebas necesarias para garantizar un funcionamiento correcto.

Objetivos de la ingeniería de software:

1. Crear soluciones con programas informáticos que satisfagan las necesidades de la sociedad.
2. Solucionar problemas de programación.
3. Estructurar la elaboración de evidencias que comprueben el correcto funcionamiento respecto al análisis y diseño.

El software es un producto elaborado por la Ingeniería de Software, definido por un conjunto de programas, documentación y datos para construir un sistema.

Características del software (Aparicio. A, 2012):

- El software se desarrolla, no se fabrica: es decir se utiliza un modelo de desarrollo que comprende a un análisis, diseño, desarrollo, implementación y evaluación.
- No se estropea, pero si sufre desgaste: El software durante su periodo de vida sufre cambios por lo que es común encontrar fallos que si no se corrigen el software comienza a deteriorarse.
- Se construye a medida: Conforme el software evoluciona se crean estándares para ser reutilizable.

Aplicaciones del software:

Software de sistemas	Programas creados para usar otros programas como compiladores o sistemas operativos.
Software de ingeniería y científico	Uso de algoritmos para un sistema.
Software de inteligencia artificial	Uso de algoritmos no numéricos para dar solución a problemas complejos.

Etapas del proceso de elaboración de software:

Análisis de requerimientos -> Diseño-> Desarrollo -> Pruebas-> Liberación

Con su origen en 1990 los diagramas UML forman parte de la ingeniería de software como una herramienta estandarizada de representar visualmente diferentes aspectos del software y procesos relacionados a este, de esta forma se puede comunicar de forma más clara y efectiva el desarrollo de software.

2.2 LENGUAJE DE MODELO UNIFICADO (UML)

UML (Lenguaje de Modelado Unificado) es una herramienta para especificar o describir procesos qué permiten la creación de un sistema en el desarrollo de software. Su objetivo es mostrar de forma visual y sencilla los conceptos, procesos y funciones del sistema.

Algunos modelos de diagramas UML son:

Diagramas UML Estructurales:

- Diagrama de clases.
- Diagrama de objetos.
- Diagrama de componentes.
- Diagrama de estructura compuesta.
- Diagrama de despliegue.
- Diagrama de paquetes.
- Diagrama de perfiles.

Diagramas UML de Comportamiento:

- Diagrama de casos de uso.
- Diagrama de descripción general de interacción.
- Diagrama de secuencia.
- Diagrama de actividades.
- Diagrama de interacción.
- Diagrama de máquina de estados.
- Diagrama de comunicación.
- Diagrama de tiempos.

Los diagramas UML ayudan a forman un conjunto de herramientas para desarrollar software, nacen hace varias décadas y se han ido transformando conforme avanza la tecnología. Desde las primeras computadoras hasta los móviles actuales.

Historia del software y los móviles:

1962 IBM implementa el reconocimiento de voz en la IBM Shoebox.

1968 Surge la Ingeniería de Software.

1970 Surgen las metodologías de desarrollo de software.

1973 Se crea el primer teléfono celular.

1980 Xerox da inicio al Mobile Learning.

1990 Se crean las primeras aplicaciones móviles.

1993 En México comienza la integración de la educación inclusiva.

1996 PalmOS primer sistema operativo móvil.

1990-2000 Primeros smartphones.

1999 Se crea el termino IoT por Kevin Ashton.

2007 Nace IOS 1.0 y Android 1.0.

2011 Nace Siri, un asistente de voz parte de un dispositivo móvil.

2014 Surgen asistentes personales como Alexa o Google Assistant.

2015-Actualidad Surgen las nuevas generaciones de dispositivos móviles.

Figura 4. Historia del software y los móviles.
Elaboración propia.

2.3 DISPOSITIVOS MÓVILES

Un dispositivo móvil es comúnmente definido como un aparato de pequeñas dimensiones o de tamaño pequeño, cuenta con una capacidad de procesamiento, con memoria regularmente limitada y está diseñado para cumplir con una función específica.

Ejemplos de dispositivos móviles:

- Teléfonos Inteligentes o Smartphone
- Tabletas
- E-Reader (Lector de libros electrónicos).
- Videoconsolas portátiles
- Reproductores digitales

2.3.1 TELEFONOS INTELIGENTES O SMARTPHONES

Los teléfonos portátiles surgen cerca de los años 80 del siglo XX, es un dispositivo electrónico que tiene la funcionalidad y las características similares a las de un ordenador personal. Su principal función es la comunicación, pero con el pasar de los años se incorporan nuevas tecnologías agregando funciones como mensajería de texto, juegos, agendas, cámara, acceso a internet, reproductor de audio e incluso GPS, dando origen a los teléfonos inteligentes o smartphones. Una característica de los teléfonos inteligentes es que desaparece el teclado numérico habitual y el espacio permite colocar una pantalla más grande vinculado a pantallas táctiles, contienen bocina, micrófono, batería y antena de mejor calidad a las convencionales. Regularmente los usuarios pueden añadir distintas funcionalidades en forma de aplicaciones móviles o programas qué se deben instalar. (Gobierno de Navarra, s. f)

Estas aplicaciones llevan por nombre “Apps” que pueden encontrarse en un mercado de aplicaciones en la red de internet y su compatibilidad depende del sistema operativo con el que cuenta el móvil.

Cada *smartphone* contiene una parte esencial para su funcionamiento, el cual permite instalar estas "Apps" fungiendo como intermediario entre el hardware del dispositivo y las aplicaciones móviles. En los más recientes años han surgido varios sistemas operativos para móviles, cada uno con características, novedades, actualizaciones y funcionalidades distintas buscando mejorar la interactividad entre el usuario y el móvil, todos compitiendo en un mercado que cada vez es más demandado.

2.4 SISTEMAS OPERATIVOS MÓVILES

Un sistema operativo es el software principal de un dispositivo, es el encargado de gestionar recursos del sistema, siendo fundamental para el uso correcto de su funcionamiento, consiste en el uso de interfaces gráficas, entornos de escritorio y gestores de ventanas de lo que se realiza en ese momento con un dispositivo para coordinar una comunicación entre el usuario y ordenador, haciendo posible un uso cómodo y sencillo.

Sistemas operativos más populares actualmente:

ANDROID
Es un sistema operativo móvil de código abierto, adquirido por Google en 2005, usado en Smartphones, Pantallas inteligentes, Tabletas y Relojes.

Su mercado de aplicaciones lleva por nombre "Play Store" una tienda y plataforma en línea que permite a los usuarios descargar aplicaciones móviles.
Versión actual 2023: Android 13 lanzado el 15 de agosto de 2022

IOS

Es un sistema operativo móvil de código cerrado, posicionado como el segundo más utilizado perteneciente a la empresa Apple Inc. Usado en *smartphones* conocidos como iPhone, tabletas y relojes.

Su mercado de aplicaciones lleva por nombre "App Store" tienda en línea para descargar aplicaciones para dispositivos móviles.

Versión actual 2023: iOS 17 lanzado el 18 de septiembre de 2023

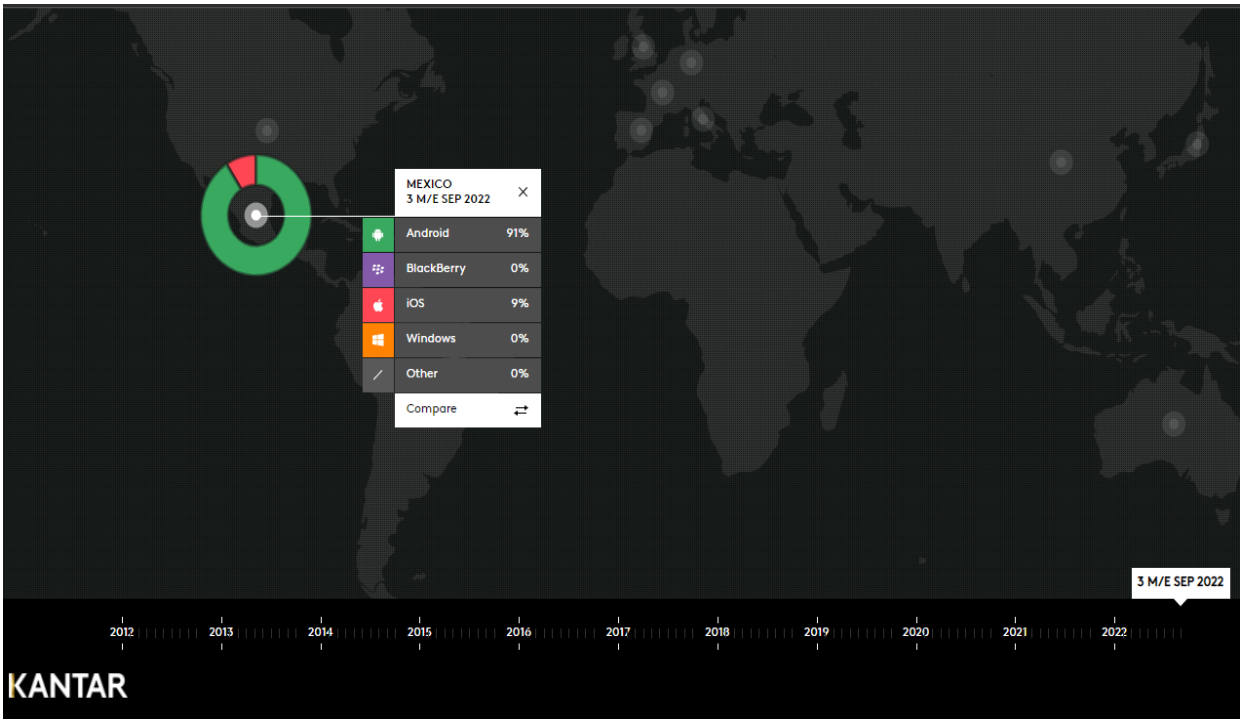


Figura 5. Sistemas Operativos Móviles en México. Obtenida de KANTAR.com.

En (Figura 5), se muestra un estudio realizado en el año 2022 sobre los sistemas operativos de *smartphones* más utilizados en México, realizado por KANTAR, empresa inglesa dedicada al análisis de datos. Datos observados:

Android: 91% de la población.

IOS: 9% de la población.

Otros: Menor al 1% de la población.

La usabilidad de los sistemas operativos en México desempeña un factor importante para la creación de aplicaciones móviles, impulsando los avances tecnológicos, siendo Android e IOS como los dominantes en el mercado. Asi las empresas crean cada día más aplicaciones móviles para que sus usuarios puedan acceder a distintas experiencias con el uso del móvil. Cada “App” se comporta y está orientada de manera diferente, algunos casos de uso se relacionan con la educación, el ocio, la productividad, la comunicación, el entrenamiento o la música.

2.5 APLICACIONES MÓVILES

Una aplicación móvil comúnmente conocida como “App” es un tipo de programa diseñado para ejecutarse en un dispositivo móvil y desarrollar una función en específico. Actualmente existen tres tipos de aplicaciones móviles:

Apps Nativas: Una aplicación nativa esta desarrollada en un solo sistema operativo determinado Android o iOS, es conocida como SDK (Software Development Kit). La ventaja que presentan estas aplicaciones es que permite acceder al hardware móvil de una forma más sencilla como a: la cámara, GPS o dispositivo de almacenamiento.

Apps Web: Se le conoce como aplicación web o web App a las aplicaciones que se pueden ejecutar dentro de un navegador web a través de una URL, se utilizan lenguajes comunes, entre el ámbito de desarrollo como es HTML, CCS o JavaScript. La ventaja que se presenta en este tipo de aplicaciones es la posibilidad de programar independientemente del sistema operativo a donde se va a ocupar la aplicación.

Apps Hibridas: Las aplicaciones hibridas es una combinación entre una aplicación nativa y una aplicación web, se puede usar lenguajes como HTML o CSS, pero de igual forma es posible adaptar códigos de otros lenguajes de programación como Java.

La programación de las aplicaciones móviles se genera a través de un *framework* que es un entorno de trabajo que cuenta con herramientas para el desarrollo de una aplicación móvil. Las aplicaciones móviles tienen diferentes enfoques y están orientadas a distintas poblaciones, se categorizan de forma específica, algunas categorías son:

-Aplicaciones educativas.

-Aplicaciones sociales.

-Aplicaciones de redes sociales u ocio.

-Aplicaciones musicales.

2.5.1 APLICACIONES EDUCATIVAS

Las aplicaciones educativas son una herramienta conocida como software educativo, diseñado para fusionar el conocimiento de niños y niñas de manera digital en diferentes áreas de aprendizaje.

Comúnmente, las aplicaciones cuentan con un modelo didáctico para ayudar a los usuarios a procesar mejor la información haciendo uso de sonidos, animaciones e imágenes. Disponibles en mercados de aplicaciones como PlayStore o AppStore.

Ejemplos:

Kahoot (Figura 6): Aplicación móvil gratuita que permite la creación de cuestionarios de evaluación, sirviendo para reforzar el aprendizaje.

Figura 6. Logotipo Kahoot. Obtenida de AppStore.

Khan Academy: Su objetivo es crear un conjunto de herramientas en línea que ayuden a la educación de los estudiantes con el uso de videos y ejercicios.

Duolingo (Figura 7): Funciona como una plataforma web adaptada para dispositivos móviles destinada para el aprendizaje de idiomas, donde los usuarios predestinan su propio ritmo.

Figura 7. Logotipo Duolingo. Obtenida de AppStore.

2.5.2 APLICACIONES INCLUSIVAS

Las aplicaciones inclusivas son un tipo de aplicación móvil desarrolladas para personas con algún tipo de discapacidad, adaptando tecnologías con el fin de ayudar a sus usuarios.

Ejemplos:

Figura 8. Logotipo Accessibility Plus. Obtenida de AppStore.

Accessibility Plus (Figura 8): Funciona como un sistema de geolocalización para que los usuarios con movilidad reducida puedan localizar puntos de interés.

Disable Park (Figura 9): Aplicación disponible que permite localizar aparcamientos para discapacitados.

Figura 9. Logotipo Disable Park. Obtenida de AppStore.

Sordo ayuda: Aplicación que integra reconocimiento de voz y traduce a texto o texto a voz, facilitando la comunicación.

2.6 FRAMEWORKS PARA DESARROLLO DE APLICACIONES MÓVILES

Un *framework* es un marco de trabajo, con una estructura para elaborar un proyecto en específico, caracterizado por contener programas, bibliotecas y herramientas que ayudan a unir diferentes elementos de un proyecto.

En el desarrollo de aplicaciones móviles, los *framework* y las tecnologías se han adaptado con el fin de crear nuevas aplicaciones más funcionales.

Dentro de las ventajas al usar un *framework* como *React, Angular* u otro destacan: acelerar el proceso de trabajo, permite el desarrollo más colaborativo, reduce errores que puedan presentarse en el proyecto.

IONIC

Surgido en el año 2013, es un SDK (Kit de Desarrollo de Software) para desarrolladores *Frontend*, de código abierto orientado a desarrolladores de aplicaciones hibridas (IOS, ANDROID, WEB), usando tecnologías web como HTML, CSS y JAVASCRIPT. Ofrece una biblioteca de componentes, gestos y herramientas para interfaces de usuario, además permite integrar *frameworks Frontend* populares como Angular, React y Vue.

FLUTTER

Surgido en el año 2017, es un *framework* de código abierto para desarrollo de aplicaciones hibridas y orientado a desarrolladores *Frontend*, se basa en tecnologías de lenguaje Dart perteneciente a Google sucesor de JavaScript.

REACTNATIVE

React Native surgido en el año 2015, es un *framework* JavaScript similar a React, de código abierto para crear aplicaciones nativas iOS o Android, orientado a desarrolladores *Frontend*, se basa en la construcción de bloques visuales empleados por JavaScript. Una de las ventajas es que permite crear aplicaciones que sean ejecutadas tanto en IOS como en Android con el mismo código base, sin preocuparse tanto por un lenguaje nativo de Android o IOS.

EXPO REACT NATIVE

Expo es un sistema de herramientas y bibliotecas que facilitan el uso para crear aplicaciones móviles en React Native. Básicamente para su funcionamiento se debe contar con 2 elementos fundamentales: Expo CLI (línea de comandos) y Expo Client (cliente), la comunicación entre estas dos herramientas nos permite programar una aplicación móvil en un ambiente de desarrollo Expo CLI y probar las aplicaciones dentro de Expo Client con un dispositivo móvil o el emulador dentro de Android Studio.

VENTAJAS/DESVENTADAS EXPO REACT NATIVE – CODIGO NATIVO:

EXPO REACT NATIVE:

Ventajas	Desventajas
Programación base en JavaScript.	Algunas bibliotecas no funcionales.
Fácil de entender.	Dificultad de acceder a funcionalidades 100% nativas.
Multiplataforma.	Errores poco descriptivos.
Reutilización de código.	Aplicaciones de mayor peso a las de código nativo.
Menos costo de desarrollo.	Curva de aprendizaje mejor a codigo nativo.
Desarrollo más rápido.	Menos control sobre el hardware.

CÓDIGO NATIVO:

Ventajas	Desventajas
Mejor rendimiento.	Inconsistencias entre sistemas operativos.
Actualización de nuevas funcionalidades.	Desarrollo para equipos más grandes.
Soporte oficial.	Incompatibilidad entre versiones.

EXPO REACT NATIVE CLIENT

Cliente para usuarios Android:

Para los usuarios con sistema operativo Android se requiere de instalar la aplicación Expo GO localizada en el mercado de aplicaciones PlayStore. Al ingresar al App de Expo GO se necesita escanear el código QR que surge al levantar nuestro servidor de Expo CLI.

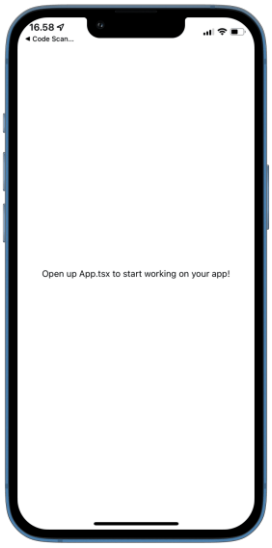
Cliente para usuarios IOS:

Para los usuarios con sistema operativo IOS se requiere de instalar la aplicación Expo GO localizada en el mercado de aplicaciones AppStore. Para observar nuestra App es necesario escanear el QR que surge al levantar nuestro servidor de Expo CLI desde nuestra cámara del dispositivo.

COMUNICACIÓN EXPO CLI y EXPO CLIENT

Visualización del App en un dispositivo móvil emulado localmente.

QR Generado al iniciar el servidor local de desarrollo Expo React Native



Visualización del App en un dispositivo móvil físico.

Figura 10. Comunicación EXPO CLI y EXPO CLIENT. Elaboración propia.

A continuación [Tabla 1], se muestra un cuadro comparativo entre distintos *frameworks* de desarrollo móvil multiplataforma:

FRAMEWORKS PARA DESARROLLO DE APLICACIONES MÓVILES			
Framework	IONIC	REACTNATIVE	FLUTTER
Tecnologías de lenguaje	HTML, CSS, JAVASCRIP	JAVASCRIPT, TYPESCRIPT	DART
Comunidad	Muy Popular	Activa y popular	Popular
Reusabilidad	90 % de código	80 – 90 % de código	50 – 80 % de código
Interfaz	HTML, CSS	Componentes nativos	Componentes propios
Experiencia necesaria	Básica	Básica - Intermedia	Intermedia
Ejecución	Navegador web – Entorno grafico	En un entorno grafico	Navegador web - Entorno grafico
Rendimiento móvil	Bajo	Excelente	Muy bueno
Despliegue	Escritorio, Web, Móvil	Móvil	Escritorio, Web, Móvil
Costo de uso	Gratuito	Gratuito	Gratuito
Aplicaciones creadas	Diesel, JustWatch	Instagram, Facebook	Google

Tabla 1. Comparación de frameworks de desarrollo multiplataforma.

2.7 PROCESAMIENTO DE VOZ

El procesamiento de voz es el estudio de señales de voz que son procesadas a partir de una representación digital, cuenta con distintas categorías, entre ellas destacan: el reconocimiento de voz, análisis de voz y síntesis de voz.

Reconocimiento de voz:

El reconocimiento de voz es una rama que forma parte de la inteligencia artificial, con el objetivo de facilitar la comunicación de los humanos con los sistemas informáticos. Otorga la capacidad a una maquina o programa pueda identificar palabras o frases.

Un SARH (Sistema Automático de Reconocimiento del Habla) es un sistema computacional capaz de procesar la señal de voz emitida por un individuo, mediante un proceso que inicia con la digitalización para obtener parámetros (muestras), estas son interpretadas por el sistema para denotar su comportamiento e implementar procesos enfocados al reconocimiento.

Esto ha generado una ventaja a los sistemas de comunicación, dado que con el surgir de los medios de comunicación iniciando en los teléfonos (año 1854) donde la comunicación es mediante voz.

¿Cómo funciona el reconocimiento de voz?

En general el procedimiento para el funcionamiento del reconocimiento de voz se realiza el siguiente proceso:

El sistema detecta palabras emitidas por la voz de un individuo -> Se detectan y genera un lenguaje máquina -> El sistema de Software ofrece una respuesta ejecutando una orden.

La señal de voz se conjunta por medio de sonidos emitidos por el aparato fonador y es representada por medio de la superposición de diferentes señales elementales a base de senos y cosenos, regularmente la señal de voz está definida por un rango de 20Hz a 20kHz.

Aplicaciones del reconocimiento de voz:

- Dispositivos móviles: Aplicado en relojes o teléfonos móviles permitiendo introducir datos de maneta natural con el uso del habla.
- Telefonía: Se pueden ejecutar ordenes mediante el habla, sustituyendo el uso de botones físicos.
- Dictado de voz: Es un sistema que escribe automáticamente lo que se dicta.
- Sistemas de control: Permite realizar algunas tareas mediante el control de voz, dentro los que destacan los asistentes personales.
- Sistemas para personas con discapacidad: Dependiendo de la discapacidad con la que cuenta el usuario, se puede utilizar un sistema de reconocimiento que ayude a solventar su problema para realizar ciertas tareas como el teclear con fluidez un dispositivo.

Las empresas saben que el futuro está en la inteligencia artificial, por ello han traído tecnologías más notables a la industria apostando por los asistentes personales inteligentes, mediante el uso del reconocimiento de voz para interactuar con dispositivos generando demasiada popularidad en los últimos años, el claro ejemplo es:

Apple (Siri): Año de lanzamiento 2011.

Microsoft (Cortana): Año de lanzamiento 2014.

Google (Google Assistant): Año de lanzamiento 2016.

Amazon (Alexa): Año de lanzamiento 2014.

2.8 MEDIOS DE COMUNICACIÓN IOT

El internet de las cosas (IoT) es la red de objetos físicos que tienen incorporados software, sensores y tecnologías, con el fin de conectar datos con otros dispositivos y sistemas por medio de internet o la nube. (SAS, s. f)

El internet de las cosas funciona a través de un software o aplicación encargada de gestionar y controlar en tiempo real y de forma remota los objetos conectados a internet. Un sistema IoT que está conformado por 3 componentes esenciales.

Dispositivos Inteligentes: Abarcan dispositivos cotidianos como una cámara de seguridad, un asistente personal o un televisor.

Interfaz de usuario gráfica: El conjunto de dispositivos de IoT pueden administrarse a través de una interfaz gráfica de usuario, es decir se puede controlar desde un sitio web o una aplicación móvil.

Aplicación IoT: Una aplicación IoT es un conjunto de software o servicios que integra datos de los dispositivos IoT, utilizando tecnologías como inteligencia artificial o *Machine Learning* para analizar datos y tomar decisiones que son recibidas por los dispositivos IoT y este responda de una manera inteligente.

2.8.1 EL INTERNET DE LAS COSAS (IoT) COMO MEDIO DE COMUNICACIÓN

El uso de esta tecnología ha tenido beneficios que ayudan a la sociedad a integrarse a las nuevas formas de interactuar con el mundo. Algunos ejemplos donde los sistemas IoT actúan:

Hogares inteligentes y conectados: Los dispositivos inteligentes en el hogar se centran en mejorar la eficiencias y seguridad de una casa, además sirve como un apoyo remoto cuando no se encuentra uno en el hogar. Pudiendo controlar cerraduras, cámaras de seguridad o amenazas con los dispositivos adecuados.

Ciudades inteligentes: Las aplicaciones IoT han podido hacer más eficientes la urbanización y el mantenimiento en la infraestructura ayudando a medir calidades de aire o niveles de humedad.

Asistentes personales inteligentes: Los asistentes personales capaces de interpretar el reconocimiento de voz y ejecutar un conjunto de acciones, este modelo se basa en un dispositivo de audio (bocina), una interfaz de usuario (App) y dispositivos inteligentes (luces, focos, audio, televisores) que se conectan entre sí por medio de la red.

2.8.2 ASISTENTES PERSONALES INTELIGENTES

Un asistente personal inteligente es un agente de software que puede realizar tareas y ofrecer servicios a un individuo con la mínima interacción hombre-maquina.

Apple (Siri): Siri es una inteligencia artificial con funciones de asistente personal, perteneciente a Apple Inc. Fue lanzado en 2007 por Adam Cheyer.

Microsoft (Cortana): Cortana es un asistente virtual creado por Microsoft. Fue lanzado el 2 de abril de 2014.

Google (Google Assistant): El Asistente de Google es un asistente virtual desarrollado por Google. Fue lanzado el 18 de mayo de 2016.

Amazon (Alexa): Alexa es el nombre del asistente virtual creado por Amazon. Fue lanzado en noviembre de 2014.

2.8.3 COMANDOS DE VOZ

Los comandos de voz sirven para controlar aplicaciones o sistemas sin la necesidad de utilizar las manos o un periférico como *mouse* o teclas, estos sistemas de comando de voz se llevan a cabo por medio de una interacción y una comunicación natural y directa con los sistemas informáticos por medio de frases predefinidas.

Su uso más actual es en los ya mencionados asistentes personales inteligentes o en inteligencia artificial, donde su principal funcionamiento es mediante comandos de voz predefinidos por su software y respondiendo con una acción o una frase.

Ejemplos de uso de comandos de voz:

- "*Siri… márcale a mi papá*". Siri procederá a marcar a un contacto de nombre "papá".

- "*Alexa… pon canciones para dormir*". Alexa procederá a reproducir música para dormir en un servicio de música.

- "*Siri… abre Google Maps y pon la ruta a mi casa*". Siri procederá abrir la aplicación de Google Maps y direccionará una ruta.

- "*Hey Google… dime como está el clima*". Google Assistant procederá a indicar el estado del clima.

- Televisores inteligentes: "*Pon Netflix y reproduce una película*". El televisor abrirá la aplicación de Netflix y pondrá una película.

Se puede observar que la mayoría de la estructura de un comando de voz se compone de una palabra de lanzamiento o activación, seguido de una orden.

Cabe mencionar que los comandos de voz muestran ciertos problemas a la hora del reconocimiento, es común que nuestro dispositivo no nos escuche bien e interprete otra cosa a la que le indicamos.

Curiosidad: Un grupo de investigación de reconocimiento del habla en Apple tuvo un caso ambigüedad donde indicaban *"I helped Apple wreck a nice beach" – "Yo ayude a Apple a estropear una buena playa"* cuya pronunciación era similar a *"I helped Apple recognize speech" - "Yo ayude a Apple a reconocer el habla"*.

2.8.4 DTMF (Frecuencia Múltiple de Doble Tono)

DTMF es un sistema que emite tonos por medio de la combinación de 2 frecuencias simultaneas y son convertidas en señales digitales. Específicamente una señal DTMF es la composición de dos tonos específicos, y las diferentes combinaciones genera que sus armónicos no se encuentren relacionados. Al pulsar una tecla distinta en el teléfono, una central decodificada recibe la combinación de frecuencias que emiten dos ondas sinusoidales y a través de filtros hace la detección de los impulsos para identificar que digito se marcó.

Tabla de Frecuencias DTMF

	1209 Hz	1336 Hz	1477 Hz	1633 Hz
697 Hz	1	2	3	A
770 Hz	4	5	6	B
852 Hz	7	8	9	C
941 Hz	*	0	#	D

Figura 11. Tabla tonos DTMF. Obtenida de https://es.wikipedia.org/wiki/Marcación _por_tonos

Característica técnica:

- Cada frecuencia generada y transmitida pude variar entre el 1,5% de la frecuencia nominal.
- Una señal DTMF valida no debe ser menor a una duración de 23ms.

Señal DTMF

Señal emitida de la suma de 2 señales sinusoidales correspondientes a:

-697 Hz
-1209 Hz
Dando como resultado al presionar la tecla 1.

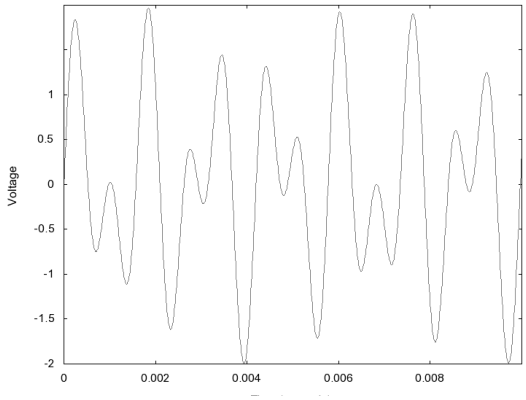


Figura 12. Señal DTMF emitida por 697Hz y 1209Hz. Obtenida de https://es.wikipedia.org/wiki/Marcación_por_tonos

Generador DTFM Online:

DTMF Tone Generator es un servidor *online* para la creación y descarga de tonos DTMF, útil para crear señales multifrecuencia de tono dual, su particularidad es que nos permite crear un tono DMTF con duración y muestreo especifico.

File Generator

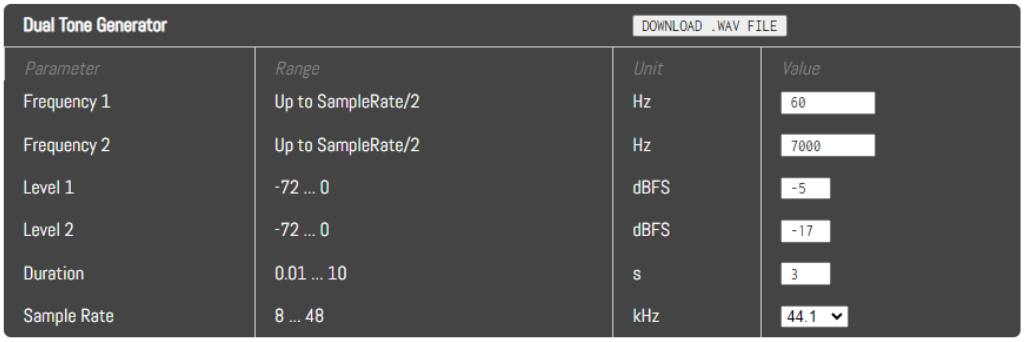


Figura 13. DTMF Tone Generator Online.
Obtenida de https://www.audiocheck.net/index.php

2.9 MOBILE LEARNING

El M-Learning o Mobile Learning es conocido como el aprendizaje electrónico, es una estrategia educativa que usa los contenidos que se encuentran en Internet haciendo uso de dispositivos móviles como celulares o tabletas. Su interacción se da a través de aplicaciones móviles, interacciones sociales, juegos o programas educativos que permiten a los estudiantes tener acceso a materiales educativos desde cualquier lugar y a cualquier hora. M-Learning no busca reemplazar la enseñanza tradicional: profesor – alumno, si no que busca apoyar el proceso de enseñanza y aprendizaje mediante el uso de herramientas tecnológicas. (García-Bullé. S, 2022)

Este modelo educativo tiene como objetivo principal el autoaprendizaje y mejorar la construcción de conocimiento para resolver problemas.

Existen diferentes características que lo identifican:

- Portable: El tamaño del sistema permite la movilidad para usuario.

- Personal: La experiencia es propia de cada usuario.
- Flexible: Se adapta ante las necesidades de cada usuario.
- Ubicuo: Cuenta con la posibilidad de acceder en cualquier lugar.
- Inmediato: Da la oportunidad de acceder desde cualquier momento.
- Accesible: El costo de uso es más bajo que el uso de otras herramientas.
- Conexión a internet: Si se requiere, permite el acceso a la red de internet.
- Motivante: Genera un potencial de motivación al usuario.

Es común encontrar esta técnica de aprendizaje en lugares donde se generan grupos o comunidades que comparten un fin en común, resultando muy efectiva en situaciones que involucra: resolver problemas, aprender un idioma, adquirir una habilidad, generar un intercambio de información entre el alumno y profesor o facilitar el trabajo en equipo.

La UNESCO estableció una serie de directrices para implementar el M-Learning. (UNESCO, 2013)

- Crear contenidos pedagógicos para utilizarse en dispositivos móviles y optimiza los ya existentes.
- Ampliar la conectividad garantizando la equidad.
- Capacitar a los docentes para que impulsen el aprendizaje haciendo uso de tecnologías móviles.
- Crear políticas relacionadas con el aprendizaje móvil.
- Promover un uso seguro, responsable y saludable de las tecnologías móviles.
- Elaborar estrategias para otorgar acceso como igualdad a todos los usuarios.
- Aumentar la conciencia sobre el uso de aplicaciones móviles para la educación.
- Utilizar la tecnología móvil para bien, mejorando la comunicación y la educación.
- Generar igualdad de género.
- Favorecer el aprendizaje continuo.

2.10 LAS APIS (INTERFAZ DE PROGRAMACIÓN DE APLICACIONES)

API es una interfaz de programación compuesta de aplicaciones que permite una comunicación entre dos componentes mediante el uso de protocolos. El uso de un API implica un conjunto de funcionalidades sobre un "servidor" para ser utilizado por aplicaciones llamadas "cliente" con procedimientos conocidos, la cual realiza una interacción software-software. (AWS, s. f)

Ejemplos de uso API:

- Automatización: Elaborar un script que lleva el control de labores cotidianas.
- Funcionalidad: Integrar una funcionalidad de consumir los datos de una aplicación dentro de otra aplicación.

Tipos de API:

- API Privada: Son internas, privadas y usadas para sistemas dentro de una empresa.
- API de Socios: Solo pueden acceder desarrolladores autorizados y asociados a empresas.
- API Publica: Esta abierta al público y al uso libre sin necesidad de autorización o costo.
- API Compuesta: Suelen ser compuestas de 2 o más APIS distintas para resolver problemas complejos.

Clasificación de APIS:

- APIS REST: Son las más comunes y se encuentran en la red donde el cliente envía una solicitud por medio de HTTP al servidor y le devuelve datos al cliente.
- API de WebSocket: Utiliza objetos JSON para proporcionar datos, admitiendo una comunicación bidimensional entra las aplicaciones de cliente-servidor.

- API de RPC: El API realiza una comunicación con procedimientos remotos donde el cliente realiza un procedimiento en un servidor y el servidor proporciona el resultado al cliente.

Ejemplos de API usadas:

- Google Maps: Servicio ofrecido por Google Company que permite el uso de mapas y el uso de geolocalización para dispositivos.

- API de Mensajería: Permite otorgar un servicio de chat similar a WhatsApp, contiene servicios de inicio de sesión, notificaciones, login o chatbots.

- API de Método de Pago: Son APIS para realizar pagos o transacciones similares a PayPal y usados en páginas o aplicaciones de *e-commerce*.

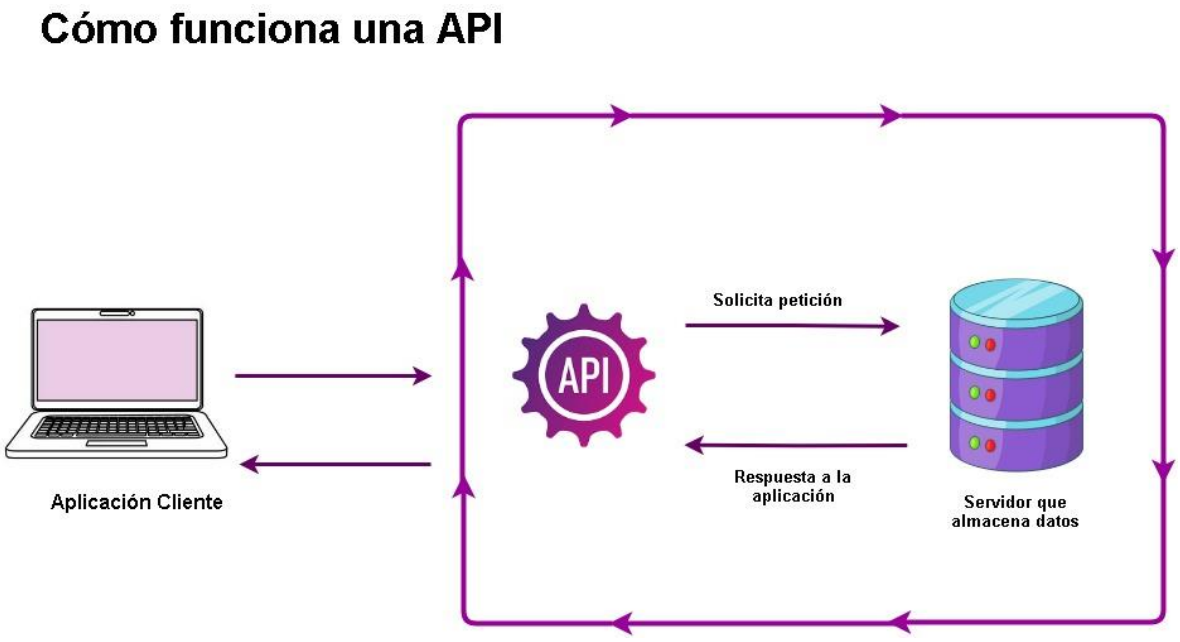


Figura 14. Descripción del funcionamiento de un API. Obtenida de Fynanty.com

2.10.1 API SPEECH TO TEXT (OPEN AI)

Speech to Text es un servicio de conversión de voz a texto perteneciente a Open AI basado en inteligencia artificial, está disponible atreves de un API REST permitiendo a los desarrolladores integrarla a sus proyectos.

Contiene distintos idiomas para 2 tipos de *endpoints*: "*transcriptions*", "*translations*" permitiendo hacer transcripción y traducción de voz a texto del audio recibido, incluyendo español, inglés, croata, alemán, griego, sueco, árabe, ente otros.

Su modelo "*whisper*" es capaz de recibir archivos de audio en formato .mp3, .wav, .mp4, .mpeg, .m4a, .webm, estos audios están limitados a máximo 25MB por archivo y no dependen de una duración mínima.

Su modelo de aprendizaje o algoritmo se define "GTP-3", es un modelo de datos masivos de texto y audio capaz de aprender las relaciones de los sonidos del habla y las palabras de texto, para el proceso de conversión de voz a texto. Se divide de la siguiente forma:

- Segmentación del audio: El audio se divide en pequeños segmentos, cada uno es distinto dependiendo del sonido del habla.
- Representación del audio: Cada segmento representa una secuencia.
- Predicción de texto: El modelo GTP-3 predice las palabras que corresponden a la secuencia generada en la representación del audio.

Algunos detalles adicionales sobre el algoritmo:

- El modelo fue entrenado con un conjunto de datos de 500 mil millones de palabras, y más de 156 trillones de tokens de audio.
- El modelo GTP-3 tiene alrededor de 175 mil millones de parámetros.

El API es de tipo REST lo que significa que utiliza un protocolo HTTPS para comunicarse con los clientes, además requiere de un *endpoint* para realizar peticiones por métodos *get, post, put, delete*. Para realizar una transcripción de voz a texto se requiere el método post, utilizado para subir un archivo a un servidor.

Consumo de API Speech to Text:

Para poder usar el API Speech to Text como un desarrollador, primero debemos crear una cuenta en la plataforma Open AI, después habilitaremos un APIKEY disponible en el menú “API keys”.

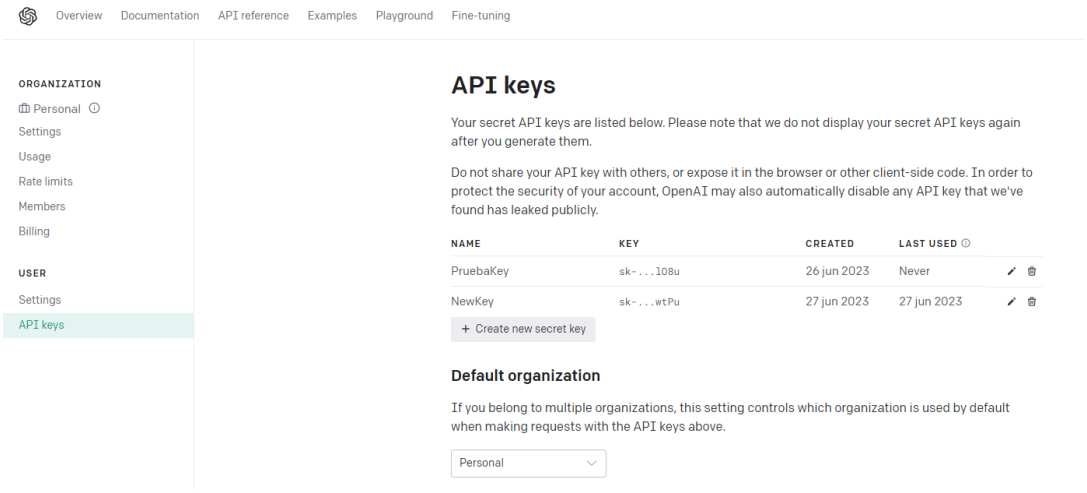


Figura 15. Generación de API Key
Elaboración propia.

Costo de uso:

Speech to Text (Open AI)

Modelo-Uso

Whisper: $0.006 dólares / minute (0.11 pesos mexicanos a precio actual del dólar en México).

Actualmente se dispone de un plan gratuito por 3 meses.

Speech-to-Text (Google Cloud)

Google Cloud ofrece un plan de $300 en créditos gratuitos que te permite transcribir hasta 60 minutos de audio al mes.

Costo después del plan gratuito: $0,006 dólares.

Una alternativa a Open AI que convierta voz a texto pudo haber sido Speech to Text de Google Cloud, el consumo es prácticamente de la misma forma, por cuestiones de costo y efectividad en la transcripción empleada para el desarrollo de la aplicación presentada en este trabajo se eligió Open AI.

2.11 HARDWARE INDEPENDIENTE

Educatrónicapp tiene la capacidad de establecer comunicación con una interfaz electrónica compuesta por distintos componentes electrónicos, la finalidad es transmitir tonos DMTF generados por Educatrónicapp, las cuales son detectados y decodificados por la interfaz electrónica para el control de un robot pedagógico.

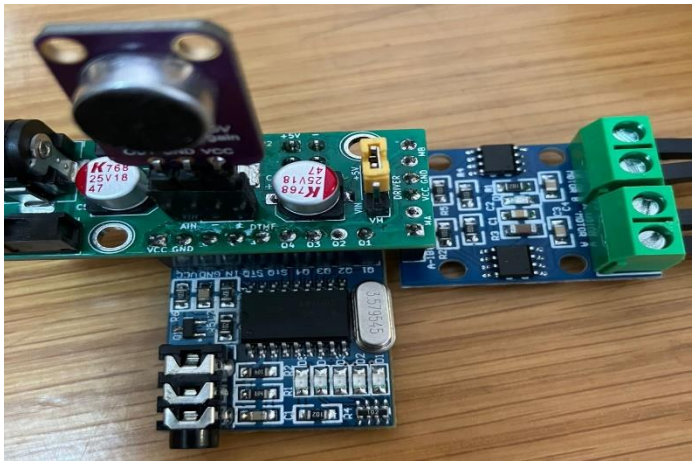


Figura 16. Interfaz electrónica.
Elaboración propia.

2.11.1 Decodificador DTMF modelo mt8870

Es un decodificador DTMF que se utiliza para convertir señales de tonos DTMF. Este tipo de dispositivos es comúnmente usado en aplicaciones de automatización, control y telecomunicaciones.

Salida digital: 4 bits de datos binarios que representan el dígito DTMF detectado.

Rango de frecuencia de entrada: 67 Hz a 3.2 kHz.

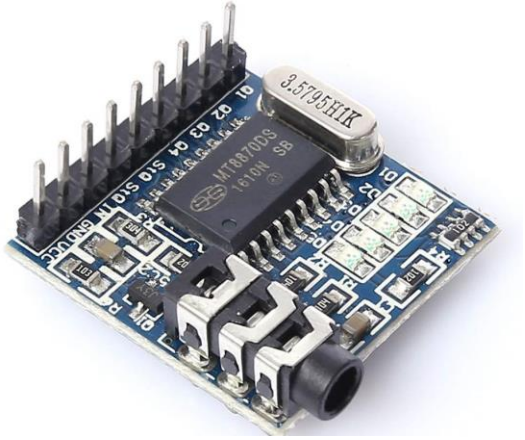
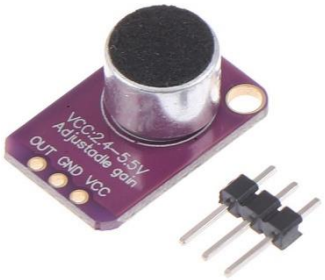


Figura 17. Decodificado DTMF mt8870.
Elaboración propia.

2.11.2 Micrófono Electret Max4466

Es un micrófono comúnmente utilizado en dispositivos electrónicos, como teléfonos móviles, cámaras de video u otros dispositivos portátiles debido a su tamaño compacto y su alta eficiencia.



Suele estar diseñado para capturar una amplia gama de frecuencias, aproximadamente 20 Hz hasta 20 kHz.

Figura 18. Micrófono electret.
Elaboración propia.

2.11.3 Amplificador de Audio

Es un amplificador de bajo ruido y alta ganancia, diseñado específicamente para aplicaciones de micrófono electret. Se utiliza para amplificar la señal de audio generada por el micrófono antes de ser procesada o grabada.

Figura 19. Amplificado de audio.
Elaboración propia

2.11.4 Drivers I9119S

Es un controlador de doble motor diseñado para controlar la dirección y velocidad de motores de corriente continua (DC).

Recibe señales digitales en código binario.

Figura 20. Drivers I9119S.
Elaboración propia.

Figura 21. Prototipo base para pruebas de robot elevador. Elaboración propia.

CAPITULO 3
METODOLOGÍAS DE DESARROLLO DE EDUCATRÓNICAPP

3.1 METODOLOGÍAS PARA EL DESARROLLO DE APLICACIONES MÓVILES

El avance en la tecnología y la creación de las aplicaciones móviles dieron un cambio al desarrollo de software, las distintas tareas ahora pueden ser ejecutadas desde la palma de la mano en pequeños dispositivos móviles. Esto abrió el camino a varias interrogantes acerca de la calidad que ofrecen estas aplicaciones y generó un conflicto para la ingeniería de software. Debido a las variantes características que presentan los nuevos dispositivos móviles, actualmente el 80% de la población mundial hacen uso de un dispositivo móvil, generando nuevas experiencias de usuario y nuevas utilidades por cubrir.

Es aquí donde entran las metodologías de desarrollo de software móvil, estas no son más que un conjunto de procedimientos e instrucciones que están conformadas por distintas fases para elaborar un software de calidad, fungiendo como un manual de técnicas u organigrama. Los recursos con los que dispone el desarrollo y necesidades a cubrir forman parte del análisis para elegir la metodología necesaria que reduzca la complejidad, gasto innecesario de recursos, se eviten errores y que facilite el mantenimiento.

Todo proyecto tiene un escenario diferente, lo que implica metodologías diferentes, con el pasar del tiempo y desde la aparición de las primeras metodologías han sufrido cierta evolución, comenzando con metodologías tradicionales, después fueron remplazadas por metodologías agiles y estas a su vez por metodologías hibridas.

A continuación, se mencionarán algunas metodologías posibles para usar en el desarrollo de este proyecto:

3.1.1 METODODOLOGÍA ÁGIL

La metodología ágil o *agile* del desarrollo de software adopta un concepto de la gestión de proyectos en función de necesidades de un cliente y expectativas del usuario, es decir se le da mayor valor al individuo, a la colaboración del cliente y al desarrollo incremental del software con iteraciones muy cortas enfocándose en el desarrollo y lo demás no es importante. *Ágile* surge en el año de 1995 por Jeff Sutherland y Ken Schwaber con bases previas de la investigación de Hirotaka Takeuchi e Ikujiro Nonaka (1986), como propósito de un planteamiento de trabajo para el desarrollo y mantenimiento de proyectos complejos.

La palabra *Ágile* hace enfoque a un cambio rápido y conciso, no solo a la adopción de procesos, los principios que dan origen al manifiesto implican la satisfacción del cliente mediante entregas tempranas y continuas de software que funcionen, con cambios en cualquier etapa del proyecto.

Ágile se basa en 4 conceptos esenciales:

- Individuos e interacciones que van por encima de los procesos y herramientas: Los procesos deben contener ayuda y un soporte como guía de trabajo.
- Primero es el software funcional antes que una documentación: Los documentos son eficientes para soporte del software, registran información, en muchas cuestiones legales son obligatorios, pero menos importantes que los productos funcionales.
- Colaboración con el cliente por encima de cualquier negociación: En el desarrollo ágil el cliente es un miembro más del equipo, que se integra y colabora en el grupo de trabajo.
- Respuestas ante el cambio por encima de algún plan: El desarrollo que surge de entornos inestables, tienen como factor inherente el cambio y la evolución rápida y continua, es por ello, por lo que no conlleva un plan de trabajo, si no todo surge al momento, siendo más útil para desarrolladores experimentados.

Además de los conceptos básicos, la metodología ágil contiene principios con los que se definen como diferente ante otras metodologías de desarrollo de software.

- Hacer entregas rápidas y continuas.
- El ritmo de trabajo es constante.
- Los responsables y desarrolladores trabajan juntos.
- Aceptar cambios incluso al final del proyecto.
- El equipo debe ser efectivo y preciso.
- Generar la menor documentación posible.

Al final podemos decir que la metodología ágil permite mejorar la satisfacción del cliente cuando tiene un proyecto para un corto tiempo, además se le permite ahorrar costes en tareas que pueden ser frustrantes, potenciar los equipos de desarrollo en eficiencia y corrección de errores. El usuario también es parte de cierta forma del proyecto puesto que es puesto a prueba en las entregas rápidas para saber si el trabajo es eficiente y con una experiencia positiva. Aunque algunas veces las metodologías ágiles no siempre son las recomendadas en los proyectos ya que todo se desarrolla en un tiempo muy corto y requiere de desarrolladores muy experimentados. Actualmente se han creado metodologías que usan los principios básicos de las metodologías ágiles, por ejemplo:

KANBAN: Desarrollado por Taiichi Ohno en 1940.

SCRUM: Desarrollado por Hirotaka Takeuchi Japón, año 1986.

EXTREME PROGRAMMING: Originario por Kent Beck, años 90.

Ventajas	Desventajas
Enfoque único en cada una de las etapas.	Falta de documentación.
Trabajo más rápido.	Requiere una interacción entre cliente y desarrolladores.
Corrección de errores de forma más específica.	Desarrolladores con bases sólidas y habilidades multidisciplinarias.

3.1.2 MOBILE-D

Mobile-D es una metodología de desarrollo creada en los años 2004-2005 de origen finlandés, específicamente creado para el desarrollo de aplicaciones móviles, forma parte del grupo de metodologías agiles, consta de cinco fases: exploración, iniciación, producción, estabilización y pruebas de sistema. Su principal objetivo es entregas rápidas en grupos de desarrollo pequeños.

Como las demás metodologías agiles, es utilizada por equipos pequeños de desarrollo de no más de 10 desarrolladores, Mobile-D se enfoca en entregar documentación acerca del proyecto que orienta a los desarrolladores en el desarrollo de la aplicación móvil, la calidad del producto al final del proyecto suele ser buena, debido al poco tiempo que conlleva el desarrollo, este minimiza los costos y es eficiente cuando se tienen pocos recursos.

Las fases de Mobile-D se mencionan a continuación:

- Exploración: Se genera un plan de trabajo con conceptos básicos y objetivos generales sobre el proyecto, además se elige el rol que tomara cada desarrollador. El propósito es identificar y establecer los grupos necesarios para las fases siguientes de forma organizada y controlada.

- Iniciación: Los desarrolladores identifican los recursos necesarios físicos y técnicos, se capacita al equipo de desarrollo, se integran los casos de uso, las funcionalidades y se crea un diseño de interfaz de su usuario UI.

- Producción: En la fase de producción, interviene la programación, repitiéndose hasta implementar todas las funcionalidades que se diseñaron. Se preparan las pruebas que se realizaran una vez que se termine la codificación.
  En esta fase suele definirse el día de lanzamiento, en base a una versión funcional, al finalizar esta fase se recolecta la siguiente documentación: estado de funcionalidades puestas aprueba, anotaciones sobre el desarrollo, experiencias de usuario y modificaciones.

- Estabilización: Se llevan a cabo las ultimas integraciones del sistema para que este completo y se verifica que esté preparado para funcionar correctamente, sus tareas son muy similares a las de la etapa de producción y es la etapa más importante porque se integran todas las funcionalidades de la aplicación y, sobre todo, se genera la mayor parte de la documentación terminada del proyecto.

- Pruebas: Las pruebas se realiza de forma general según lo establecido y definido en la etapa de exploración, pueden realizarse las pruebas que sean necesarios hasta tener una versión estable y funcional del sistema. Se le hace una demostración al cliente y se corrigen o eliminan si es el caso los últimos detalles, una vez finalizado el proyecto y la documentación, se genera la producción de la aplicación para ser publicada y entregada al usuario final.

Ventajas	Desventajas
Bajo costo.	Desarrolladores con experiencia.
Entregas rápidas.	No hay mucha comunicación en el equipo.
Proyecto de alta calidad.	Corrección de errores al final del proyecto.

3.1.3 WATERFALL O CASCADA

Creada en el año de 1970 por el computólogo Winston W. Royce. Waterfall será la metodología utilizada para el desarrollo de este proyecto.

El modelo Waterfall está definido como un proceso de desarrollo en forma secuencial y suele utilizarse mucho en desarrollo de software, por su comodidad también es usado muy recientemente en el desarrollo móvil. (Digital Talent Agency, s. f). Esta metodología conlleva un conjunto de etapas que se ejecutan de forma secuencial una tras otra.

El objetivo de esta metodología es llevar un orden justificado por sus diferentes fases, ya que el avance depende una de otra, impidiendo avanzar si no se ha terminado la fase previa. Esta metodología es aplicable en proyectos donde los requisitos están bien fijados y no cambiaran durante el desarrollo del proyecto, registrando el progreso a través de la documentación de forma exhaustiva.

El primer modelo Waterfall de Winston W. Royce posee las siguientes características de cada fase:

- Análisis de requisitos del sistema y del software: En base a una consulta con usuarios se analizan las restricciones, metas y servicios con los que contara el sistema.
- Diseño: Se define una arquitectura a grandes rasgos, describiendo los elementos que formaran el producto final.
- Implementación y testing de unidades: Se ejecutan unidades desarrolladas de forma general o individual para verificar que cumplen sus funciones.
- Integración y testing de sistema: Cada parte del desarrollo se conjuntan en un producto piloto donde se realizan pruebas en conjunto y de integración.
- Mantenimiento: Se instala el sistema y se pone en marcha el producto y en base a las experiencias con los usuarios se pone en marcha una subetapa de mejora y corrección para la entrega final.

De forma general se mencionan algunas características fundamentales con las que trata de cumplir la metodología:

- Es una metodología muy organizada, evitando que se mezclen sus fases.
- Es la metodología menos compleja que existe.
- Al final de cada fase se realiza una documentación exhaustiva.
- Se basa en metodologías agiles.
- El equipo de trabajo se centra en una única fase y no se permite avanzar hasta que se finalice.

A continuación, Figura [22], se mostrará las fases del modelo Waterfall con algunas adaptaciones de mejora que han surgido en el avance de los años y como se estiman ser aplicadas para el desarrollo de este proyecto:

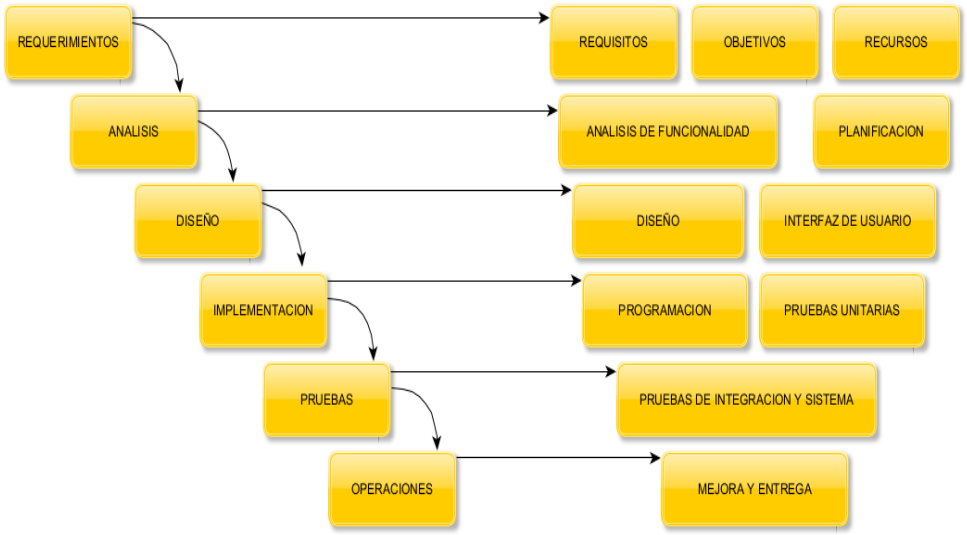


Figura 22. Descripción de metodología Waterfall.
Elaboración propia.

3.1.3.1 REQUERIMIENTOS

El análisis de requerimientos comienza teniendo un problema y conociéndolo por la cual se planea un proyecto que de una posible solución. Los requerimientos son una tarea que permite a los lideres de proyecto e ingenieros especificar las características operacionales de software, los objetivos y recursos, sirviendo como una guía o ruta para los desarrolladores y entender cómo se enlazan las diferentes piezas del software, ayudan a definir qué características contendrá el producto final. Si los objetivos no están bien definidos puede conducir a un deslizamiento del proyecto es decir el proyecto comienza a expandirse más allá de lo previsto.

Uno de los objetivos más importantes que proporcionan los requerimientos es a identificar los riesgos al principio del desarrollo y orienta la posible solución o uso de las tecnologías, dependiendo de la metodología los errores pueden encontrarse hasta la etapa final, pero con Waterfall es un poco más factible detectarlos a una etapa más temprana.

La fase de planeación o análisis de requerimientos es la fase inicial de la metodología Waterfall por lo que en base a la problemática se definieron todos los requerimientos, objetivos y recursos necesarios para el desarrollo de este proyecto.

- Requerimientos Generales:
  - Población/Usuarios de interés por hacer uso de la aplicación.
  - Contar con una problemática.
  - Disponer de un equipo de cómputo para realizar desarrollo móvil.
  - Conocimientos básicos de desarrollo móvil o web.
  - Contar con una cuenta de desarrollador para publicar un proyecto en el mercado de aplicaciones de los dispositivos móviles.
  - Buscar un *framework* gratuito para desarrollo móvil.
  - Simplificar el desarrollo de una forma eficiente, teniendo un diseño para la experiencia de los usuarios, cuidando colores, iconos y tamaños de letra.
  - Buscar tecnologías que ayuden a cumplir los objetivos y solucionar la problemática.

- Requerimientos funcionales:
  - Contar con una interfaz responsiva, dinámica y de fácil accesibilidad.
  - Integrar un editor de texto para capturar los comandos.
  - Contar con una simulación del robot.
  - Integrar un medio de comunicación aplicación-robot.
  - Definir acciones del robot como subir, bajar, detenerse, abrir, cerrar puertas.
  - Definir un lenguaje natural.
  - Permitir guardar, cargar y borrar programas.
  - Permitir reconocimiento de voz para interactuar con la aplicación.
- Requerimientos no funcionales:
  - Rendimiento de funciones en distintos sistemas operativos móviles.

- Garantizar seguridad del usuario respecto al reconocimiento de voz.
- La compatibilidad y disponibilidad en distintos sistemas operativos móviles.
- Centralizar la escalabilidad de la aplicación con nuevas funciones.
- Contar con una documentación correcta para corrección de errores o de mejoras.

➢ Objetivos:

- Crear una aplicación para la enseñanza digital introduciendo a sus usuarios en los conceptos básicos de la programación.
- Que la aplicación sea de uso gratuito y libre acceso.
- Realizar una aplicación móvil con una experiencia de usuario de fácil acceso.
- Utilizar tecnologías de inteligencia artificial para gestionar la usabilidad de usuarios con discapacidad motriz, es decir una interfaz para reconocimiento de voz.
- Facilitar el uso de "Educatrónicapp" para usuarios sin y con discapacidad motriz.
- Crear un lenguaje natural para la creación de un programa.
- Crear la codificación para la interpretación del lenguaje natural definido.
- Crear un simulador para manipulación del elevador.

➢ Recursos:

- Problemática.
- Conocimientos básicos de programación web/móvil.
- Conocimiento de *frameworks* de código abierto para desarrollar aplicaciones móviles de sistema operativo hibrido.
- Dispositivos móviles físicos para realizar pruebas físicas.
- Tecnologías de código abierto y uso gratuito para hacer uso del reconocimiento de voz por ejemplo APIS.

3.1.3.2 ANÁLISIS

Con el pasar de los años Waterfall ha sufrido algunas modificaciones para su mejora y adaptación con nuevos proyectos, la etapa de análisis suele combinarse con la etapa de requerimientos para determinar la usabilidad de las distintas secciones de la aplicación, en este proyecto el análisis se divide para definir la funcionalidad del proyecto y la planificación de este:

Análisis de funcionalidad:

TABS o PESTAÑAS

La aplicación móvil dispondrá de 3 secciones principales para interactuar de forma diferente y contendrá 2 secciones secundarias para información y ayuda. El usuario podrá moverse por las distintas secciones de forma ágil apoyado de iconos y texto que serán de ayuda para identificar en que sección se encuentra el usuario.

INTERACCIÓN CON LA PROGRAMACIÓN EDUCATIVA

Se definirá un lenguaje natural para que los usuarios que están comenzando en el ámbito de la programación no tengan que pensar en algún lenguaje de programación especifico como lenguaje C, Java o Python, si no que el lenguaje natural este definido por sentencias de uso común, enfocándose más en la habilidad de crear lógica en sus programas.

SIMULACIÓN

Educatrónicapp contará con una sección de simulación que permitirá a los usuarios interactuar de forma gráfica, haciendo más atractiva la parte visual.

INTERFAZ DE VOZ

Se incluirá una interfaz de voz para ayudar al comportamiento y control de la aplicación por medio del reconocimiento de comandos de voz.

PROGRAMACIÓN

Se dispondrá de una sección para ingresar los comandos para el controlar el robot, esta sección será la encargada de capturar los comandos y emitir un sonido DTFM correspondiente a su lenguaje natural definido.

3.1.3.3 DISEÑO

El diseño implica al equipo de trabajo de una forma general la estructuración a nivel visual del proyecto o algún otro detalle como el lenguaje de programación, interfaz de usuario a partir de las funcionalidades. Se mapea de forma general el concepto al que se quiere llegar con el producto final. De esta forma es más sencillo para el desarrollador crear el modelo a la hora de programar y no perder tiempo con diseñar y programar al mismo tiempo.

Es de suma importancia analizar si la aplicación se ejecuta en un *smartphone* o una tableta, sistema operativo y tipo de aplicación, se organiza conforme a lo que queremos que haga el usuario para que toda la información este organizada, se obtenga una buena navegación y la funcionalidad sea correcta, permitiendo una experiencia al usuario agradable, a veces el tener demasiadas secciones y demasiados elementos en un entorno puede causar estrés para los usuarios, o caso contrario un vacío puede dar pie al aburrimiento.

Algunos de los pasos para crear el diseño de una aplicación son considerar:

- Estudiar al usuario modelo que hará uso de la aplicación móvil.
- Definir la navegación con un numero de pantallas y acciones asociadas a ellas.
- Se prosigue elaborando un mapeo para pasar a la parte de UI o diseño de una interfaz de usuario.
- Cuando se valida el concepto se diseñan el resto de los elementos UI, es decir elementos que contendrá cada pantalla, colores, tamaños y animaciones.

EL UX - UI DESIGN:

La Interfaz de Usuario (UI) es el esquema donde se realizan las interacciones entre el humado y una máquina.

La Interfaz de la experiencia de Usuario (UX) consta del diseño visual como apariencia de un producto y del diseño de interacción donde se organizan los elementos funcionales y lógica.

El objetivo del diseño de una interfaz de usuario es crear una interfaz que sea fácil, eficiente y agradable, haciendo que el usuario deba minimizar el esfuerzo. Los diseñadores hacen uso de diferentes métodos para el diseño de una aplicación o un proyecto, realizan en ocasiones: entrevistas con usuarios, encuestas de que les gustaría ver, observaciones directas, para poder entender al público y asegurarse que el lenguaje visual que se transmite se adapte bien a ellos.

Es de suma importancia hacer que la interfaz de usuario sea del gusto de usuario ya que suelen ser más utilizables de acuerdo con el efecto de usabilidad estética.

Jakob Nielsen define el termino *usability* como una característica del software reuniendo múltiples componentes para que el sistema sea intuitivo, creando una interfaz usable en la que los usuarios pueden interactuar de forma fácil, cómoda, segura e inteligente posible. (Nielsen Jacob, 1993).

Un buen diseño de interfaz de usuario es fundamental para que un producto sea exitoso, si no se cuenta con él, se crea un producto inutilizable, no es solo ser agradable visualmente, por ello los diseñadores siguen un conjunto de principios de diseño en los cuales se debe tomar en cuenta distintos parámetros y elementos visuales que presenten la información o interactividad con los usuarios haciendo uso de: tipos de botones, menús, itinerarios, bloques de elementos.

Funciones de un UI Design:

- Integrar elementos de diseño, incluyendo fuentes de letra, colores, botones, menús e imágenes.
- Optimizar el estilo visual de sitios web o aplicaciones, centrándose en atraer al usuario.
- Crear una interacción optima con los elementos que contiene el producto.

Existen varios tipos de diseño de interfaz de usuario, dentro de los que destacan o que suelen ser más utilizados son:

- Interfaz Gráfica (GUI): La interacción de usuarios respecto a la información es mediante manipulación de objetos visuales con un dispositivo digital mediante el tacto.
  Ejemplo: Un sistema operativo (Windows) o el menú de una aplicación.

- Interfaz de Voz (VUI): Suelen ser interfaces con un asistente de voz, creando una interfaz de usuario conversacional en el que la interfaz reacciona conforme a comandos de voz definidos por palabras o frases.
  Ejemplo: Dispositivo Alexa (Amazon) o Siri (Apple).

TÉCNICAS Y HERRAMIENTAS

Existen distintas herramientas que facilitan el diseño de interfaces, *Figma, Adobe XD, Sketch, Invision* son herramientas más comunes para los UI Designers, con elementos básicos que son fáciles y sencillos de utilizar y con un sinfín de posibilidades ilimitadas para desarrollar un prototipo.

Figma: Es un editor de gráficos vectoriales utilizada como una herramienta generadora de prototipos, es de uso libre muy popular para diseño de interfaces web o móviles.

3.1.3.4 DESARROLLO

Como se menciona en el capítulo 2, ahora con los *frameworks* para el desarrollo de aplicaciones móviles es más factible desarrollar aplicaciones sin tanta experiencia. Para el desarrollo de Educatrónicapp se eligió Expo React Native, con un proceso descrito a continuación:

1- Desarrollar la plantilla donde tendremos las diferentes secciones de la aplicación, es decir el proyecto en general.
2- Desarrollar una interfaz que permita crear, cargar o eliminar programas, se ubicará en la sección codificación.
3- Desarrollar la simulación para que los usuarios puedan interactuar de forma gráfica.
4- Desarrollar ventanas modales o sub-ventanas que servirán de ayuda a la hora de programar o de información de Educatrónicapp.
5- Corregir errores de programación de las distintas fases.
6- Implementar colores, tamaños de texto, y botones en el proyecto general.
7- Desarrollar la interfaz de voz para el uso de Educatrónicapp, orientado hacia una población con dificultad motriz.
8- Mezclar las distintas fases del desarrollo en el proyecto en general.
9- Generar pruebas.
10- Obtener una versión beta de la aplicación Educatrónica para evaluar el desempeño a baja escala.
11- Corregir los últimos detalles y errores para la versión final.
12- Preparar la última versión para entrega y publicación en el mercado de aplicaciones.

3.1.3.5 PRUEBAS

En la metodología Waterfall, los desarrolladores entregan el proyecto en una versión beta para las pruebas pertinentes, se buscan errores para repararse antes de la entrega final, además de medir rendimiento de las funcionalidades aplicadas en la aplicación, de esta forma es posible saber si el software cumple con las exigencias definidas con anterioridad.

Las pruebas de aplicaciones móviles son un tipo de método en cual consiste en probar aplicaciones de software, se suele evaluar la usabilidad, rendimiento, funcionalidades y estabilidad, se basan en la verificación y validación para hacer efectiva la experiencia del usuario, saber si la población a la que va dirigida es la correcta, su comportamiento en los dispositivos eléctricos y si es una aplicación efectiva.

Normalmente la forma de realizar pruebas es manual o mediante una automatización. Si se lograr encontrar un error se podrá consultar la documentación para saber cómo resolverlo.

Regularmente la calidad de productos de software cuenta con varias normas de calidad, que realizan una recopilación de criterios de calidad orientadas con la funcionalidad y experiencia del usuario, algunas otras relacionadas con los tamaños de letra y colores que cumplan un diseño dinámico. La usabilidad es considerada como unos de los puntos más importantes, dicho requerimiento demuestra la comodidad con a que el usuario final puede hacer uso de la aplicación móvil.

Una investigación realizada por Rachel Harrison demuestra que la mayoría de los modelos de usabilidad usados en las aplicaciones móviles son incompletos porque solo se enfocan efectividad, eficiencia y satisfacción, dejando de lado la operatividad, estética y aprendizaje. (Harrison & Flood, 2013)

Agrega que lo más adecuado en un proyecto de desarrollo de software es conveniente tener un grupo especializado para llevar a cabo las pruebas y permita hacer una evaluación más crítica y no caer en el efecto de que el desarrollador apruebe algo que para el ya funciona.

En la Tabla [2] muestra una descripción de las distintas pruebas que se pueden tener en el desarrollo de software:

Prueba	Descripción
Prueba Unitaria	Se enfoca en analizar cada módulo de la aplicación para agilizar el proceso una vez que se integren en el proyecto final. Permite corrección de errores de forma específica.
Prueba de Sistema	Se asegura del funcionamiento general con todos los módulos integrados.
Prueba de Integración	Identifica los errores por la combinación de integrar un módulo con otro.
Prueba de Desempeño	Mide el tiempo de respuesta conforme al proceso creado.
Prueba de Regresión	Detecta si los cambios o correcciones que se realizan no tienen un afecto adverso en otras partes del proyecto.
Prueba de Estrés	Prueba el sistema funcional con memoria baja o si conlleva muchos procesos se lleva al límite.
Pruebas de Compatibilidad	Busca problemas de compatibilidad en distintos sistemas, por ejemplo, tabletas o dispositivos con sistema operativo Android o IOS.
Prueba de Usabilidad	Se concentra en la experiencia UX, como si es fácil de usar y el diseño.

Tabla 2. Tipos de pruebas para el desarrollo de un software.

Las pruebas piloto o de versión beta de Educatrónicapp constaran de reunir una población específica para realizar las pruebas del funcionamiento correcto de la aplicación, además servirá para saber que errores hay que corregir para la entrega final.

3.1.3.6 MANTENIMIENTO Y ENTREGA

Una vez realizadas las pruebas y correcciones correspondientes, se llega a la etapa de entrega donde el proyecto es ofrecido hacia los usuarios. Las ventajas y desventajas de esta etapa se describen a continuación:

Ventajas	Desventajas
Fácil de administrar.	Proyecto lineal.
Proyectos de menor tamaño donde los requisitos no son tan complicados.	Posibilidad de encontrar tarde errores a la hora de mezclar los procesos de la aplicación.
Procesos y resultados bien documentados.	Si los objetivos no son claros puede afectar los resultados hasta la etapa final.
Progresión ordenada y controlada.	Excluye al usuario final durante el desarrollo.
Fácil para desarrollares sin mucha experiencia.	Difícil de adaptar si se presentan cambios de última hora.

METODOLOGIA WATERWALL Y EDUCATRÓNICAPP

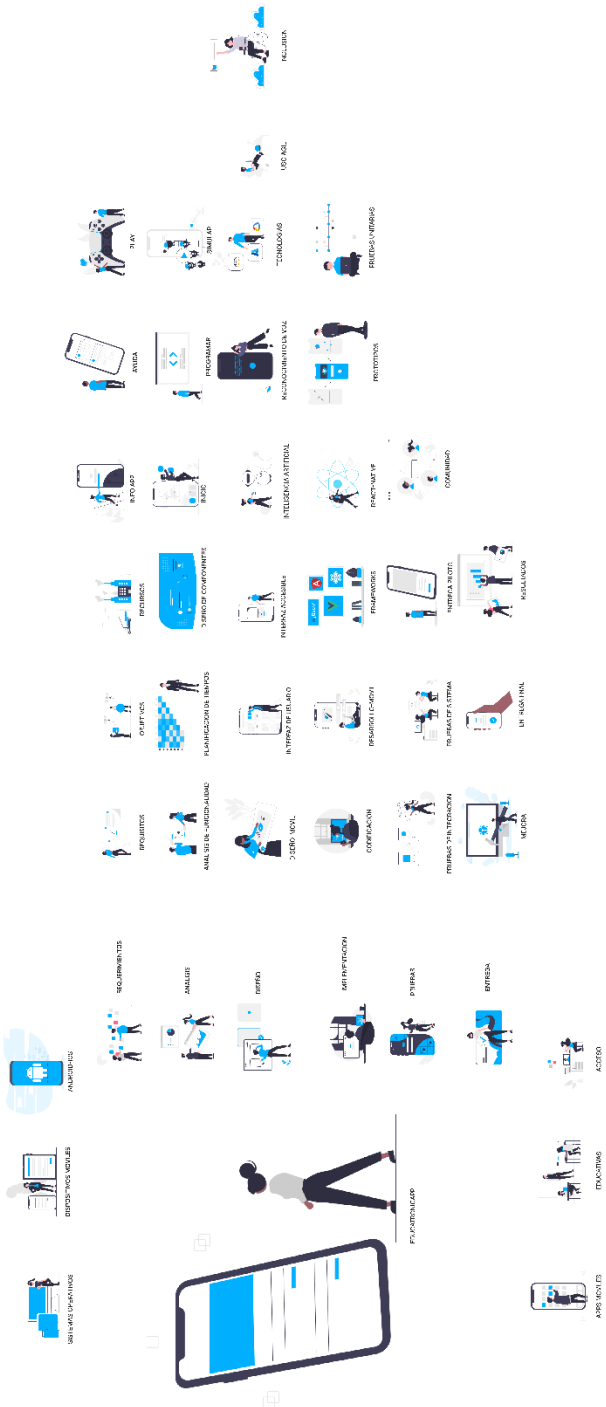


Figura 23. Descripción de metodología Waterfall adaptada a Educatrónicapp.
Elaboración propia.

CAPITULO 4
DESARROLLO DEL PROYECTO EDUCATRÓNICAPP

En la búsqueda continua de crear soluciones innovadoras y eficientes, los investigadores y estudiantes buscan soluciones que se adapten hacia los problemas educativos, dependiendo de su enfoque hacen uso de herramientas tecnológicas ayudando a potenciar su investigación de forma efectiva.

Mi objetivo principal es desarrollar una aplicación móvil que apoye la formación y educación digital donde los usuarios puedan introducirse a los conceptos básicos de la programación, asi mismo fortalecer sus competencias y habilidades , además se adaptaran herramientas de usabilidad para usuarios que cuentan con algún tipo de discapacidad motriz (movimiento), creando una interfaz en la que los usuarios pueden interactuar de forma fácil, cómoda, segura e inteligente posible.(Nielsen Jacob, 1993).

Educatrónicapp es parte de un conjunto de herramientas que conforman un grupo de trabajo donde todos sus elementos interactúan de forma complementaria. En el caso de Educatrónicapp los usuarios podrán interactuar de forma física con un robot, para poder hacer dicho control se adaptará una comunicación aplicación-robot, impulsando el aprendizaje e interactuando de forma más dinámica y divertida.

Para fundamentar mi propuesta de solución he realizado una investigación científica sobre algunos productos que abordan problemáticas similares o que mencionan alguna relación, fungiendo como una referencia ante mi solución.

A continuación, se mencionan algunos productos comerciales:

- Lego Mindstorms *ver anexo pag.121*
- BB8-Robot *ver anexo pag.122*
- Sara-Stimey *ver anexo pag.123*
- Root *ver anexo pag.124*

La arquitectura general-lógica del proyecto Fig. [24] y Fig. [25] es la siguiente:

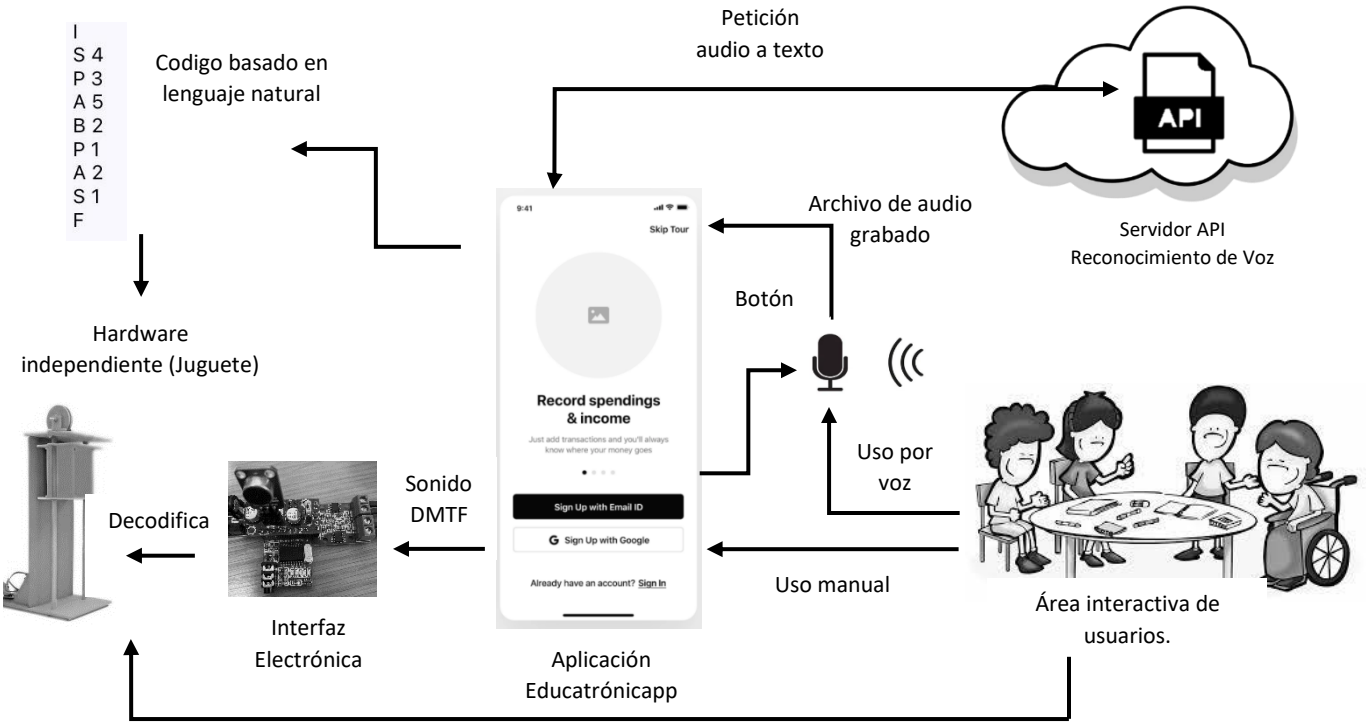


Figura 24. Descripción de arquitectura general de Educatrónicapp. Elaboración propia.

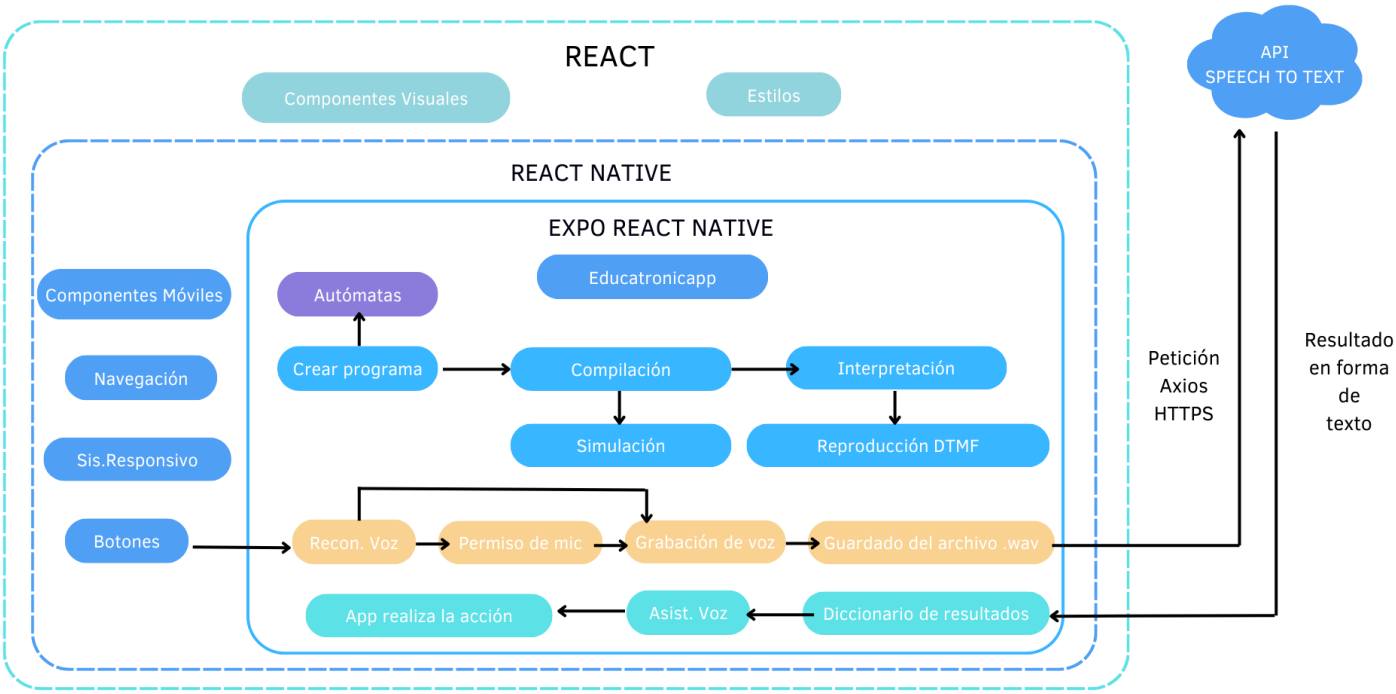


Figura 25. Descripción de arquitectura lógica de Educatrónicapp. Elaboración propia.

Descripción de componentes:

Aplicación Educatrónicapp: Sera el centro de control principal que comunica todos los demás componentes.

Código basado en lenguaje natural: Sintaxis de uso común o uso diario para la programación, definida dentro del App fungiendo como un control del juguete.

Hardware Independiente: Es un robot pedagógico que tiene la funcionalidad de actuar como un elevador.

Sonido DTMF: Sonido emitido por el App, será el medio de comunicación App-Robot.

Voz: Sera el medio de comunicación para la usabilidad del App por usuarios con problemas de movilidad.

Área interactiva de usuarios: Son grupos de usuarios que interactúan con el App y el robot.

Servidor API de reconocimiento de voz: Sera la encargada de recibir un archivo de audio por voz y mandar la respuesta en forma de texto a Educatronicapp.

Botón: Es un botón fundamental para activar el reconocimiento de voz.

Interfaz Electrónica: Sera el encargado de recibir los sonidos DTMF emitidos por el APP para hacer reaccionar el juguete.

Archivo de audio grabado: Es un archivo en formato *.WAV*, se guarda al finalizar la grabación de audio, contiene ciertas configuraciones específicas de audio.

Juguete pedagógico: Consta de un robot pedagógico que actúa como un elevador, constara de 7 niveles y permite abrir-cerrar puertas.

DISEÑO DE EDUCATRÓNICAPP

La siguiente Figura [26], indica el diseño general de Educatrónicapp, en la Tabla [3] se explica la descripción de cada una de las pantallas:

Figura 26. Diseño general de elementos de pantallas pertenecientes a Educatrónicapp.
Elaboración propia.

Sección	Descripción
Pantallas Principales	
Home	Será la pantalla principal al abrir Educatrónicapp, tendrá un diseño llamativo y contendrá las distintas secciones en la parte inferior para poder desplazarse. Dispondrá de una pantalla secundaria Help App.
Coding	En la sección Coding, dispondremos de un bloque para colocar texto y asi crear un programa nuevo, también dispondrá de botones que ayudaran a la interacción para su ejecución, contendrá una subsección para ver la ejecución de forma visual.
Simulations	Sera una sección en la cual los usuarios puedan interactuar de forma visual, teniendo elementos que fungen como control de su elevador de forma manual.
Pantallas Secundarias	
Help Coding	Sera una subsección que contendrá información de ayuda para poder crear un programa y un ejemplo básico.
Help App	Contendrá información acerca de la usabilidad de la aplicación, como un tutorial.
Pantallas Modal	
Details App	Contendrá información acerca de la aplicación, como el nombre del desarrollador, tecnologías usadas y versión.

Tabla 3. Descripción de pantallas de Educatrónicapp.

PREDISEÑO DE PANTALLAS PRINCIPALES EDUCATRÓNICAPP

Figura 27. Diseño general de pantallas de Educatrónicapp.
Elaboración propia. Realizadas en Figma.

DIAGRAMA DE CASOS DE USO.

- Iniciar App

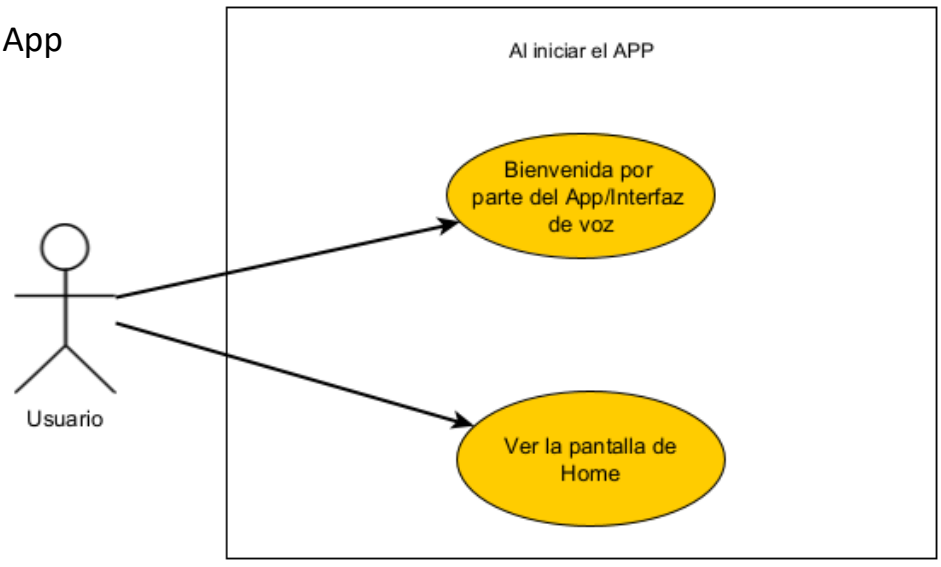


Figura 28. Diagrama de caso de uso al Iniciar el App.
Elaboración propia.

- Al estar dentro del App

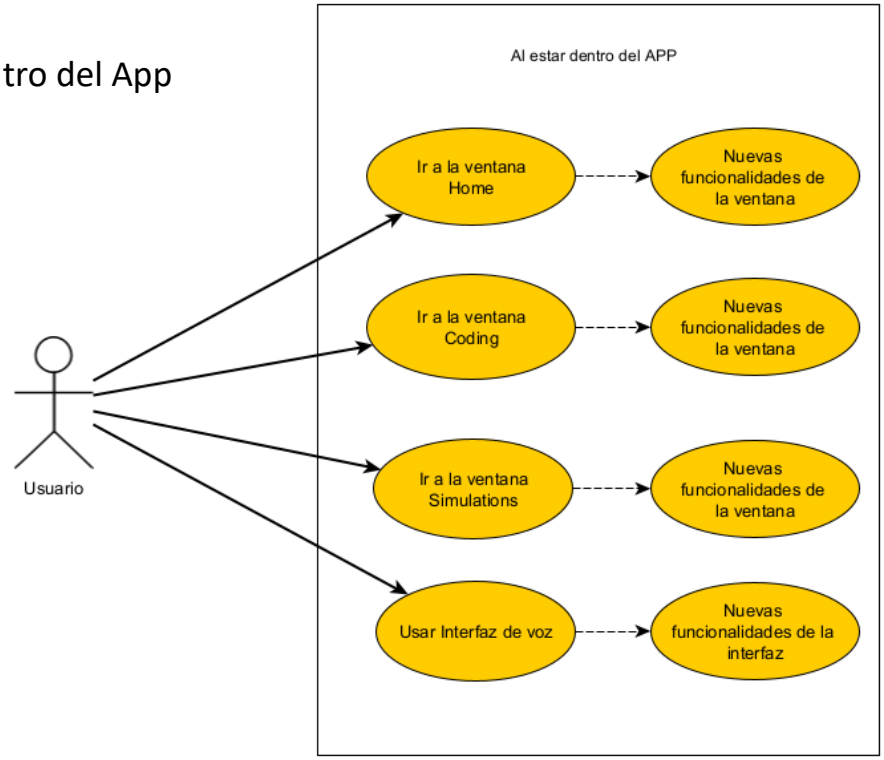


Figura 29. Diagrama de caso de uso al estar dentro del App.
Elaboración propia.

- Al estar en la pantalla Home

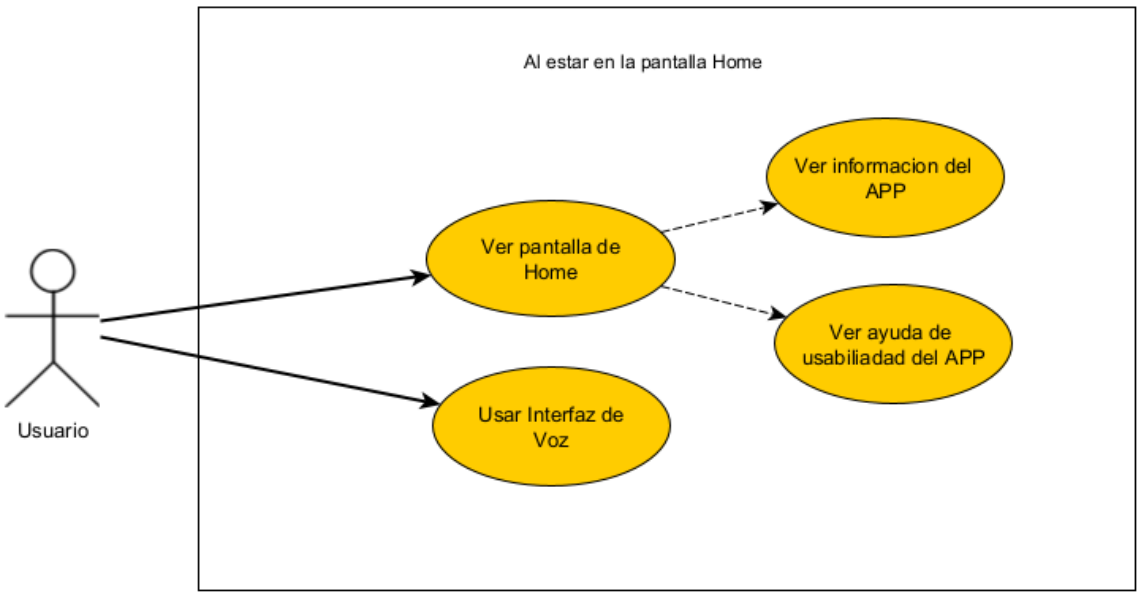


Figura 30. Diagrama de caso de uso pantalla Home.
Elaboración propia.

- Al estar en la pantalla Simulations

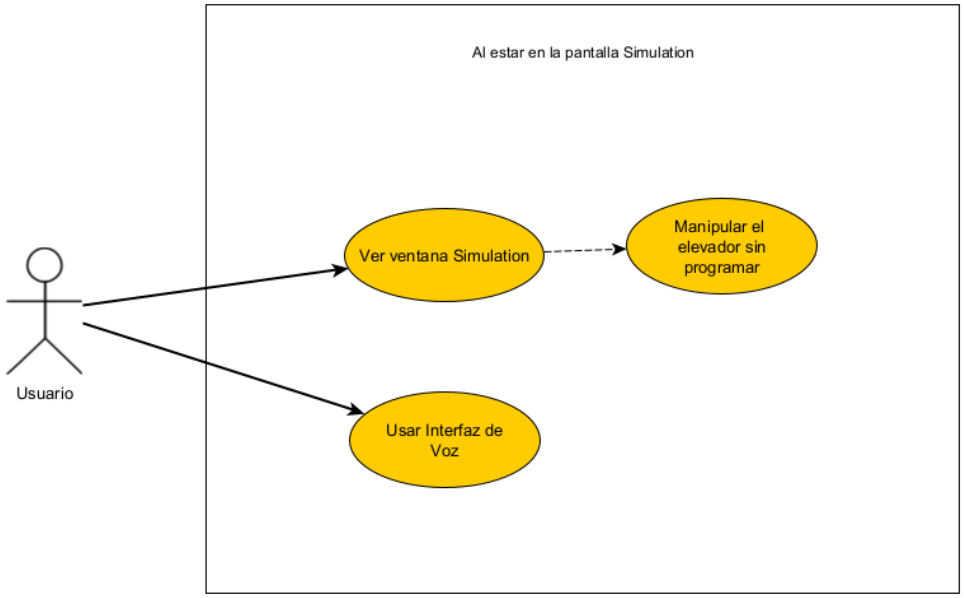


Figura 31. Diagrama de caso de uso pantalla Simulation.
Elaboración propia.

- Al estar en la pantalla Coding

Figura 32. Diagrama de caso de uso pantalla Coding.
Elaboración propia.

- Al usar la interfaz de voz

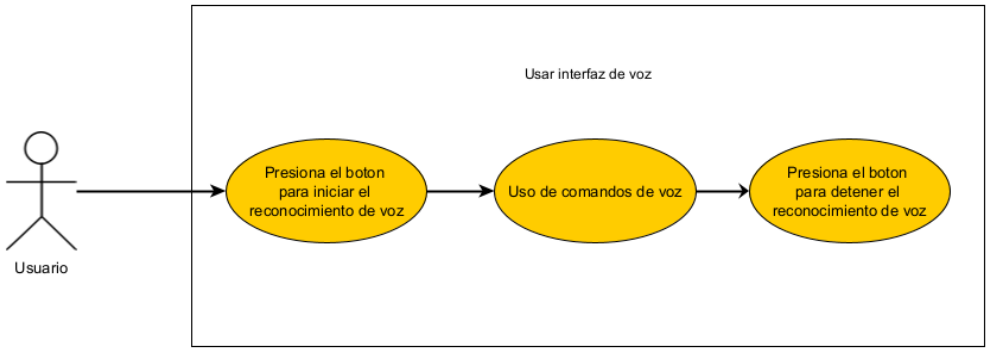


Figura 33. Diagrama de caso de uso interfaz de voz.
Elaboración propia.

DIAGRAMAS DE DESCRIPCIÓN GENERAL DE INTERACCIÓN.

- Uso de APP-Navegación por usuario con discapacidad motriz.

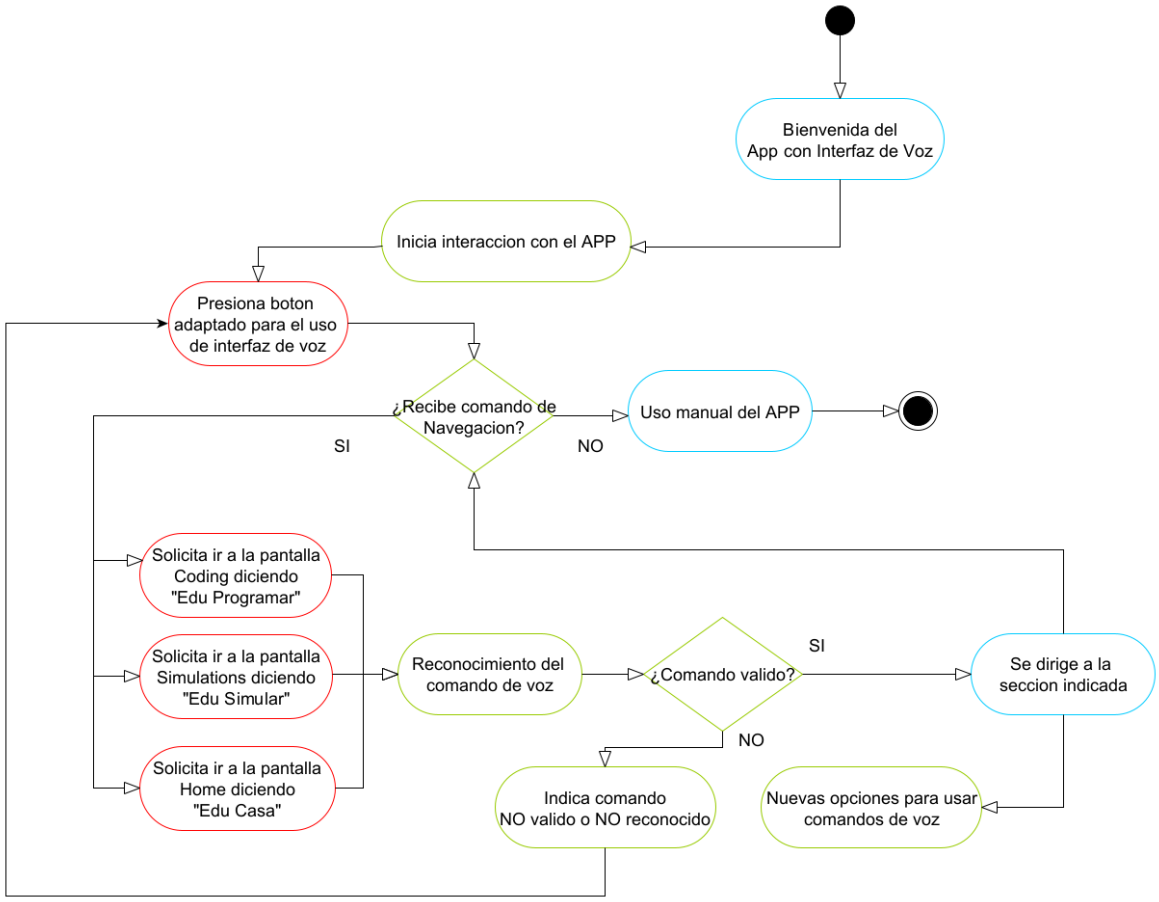


Figura 34. Diagrama de interacción para navegación por usuario con discapacidad motriz.
Elaboración propia.

- Uso de APP-Navegación por usuario común.

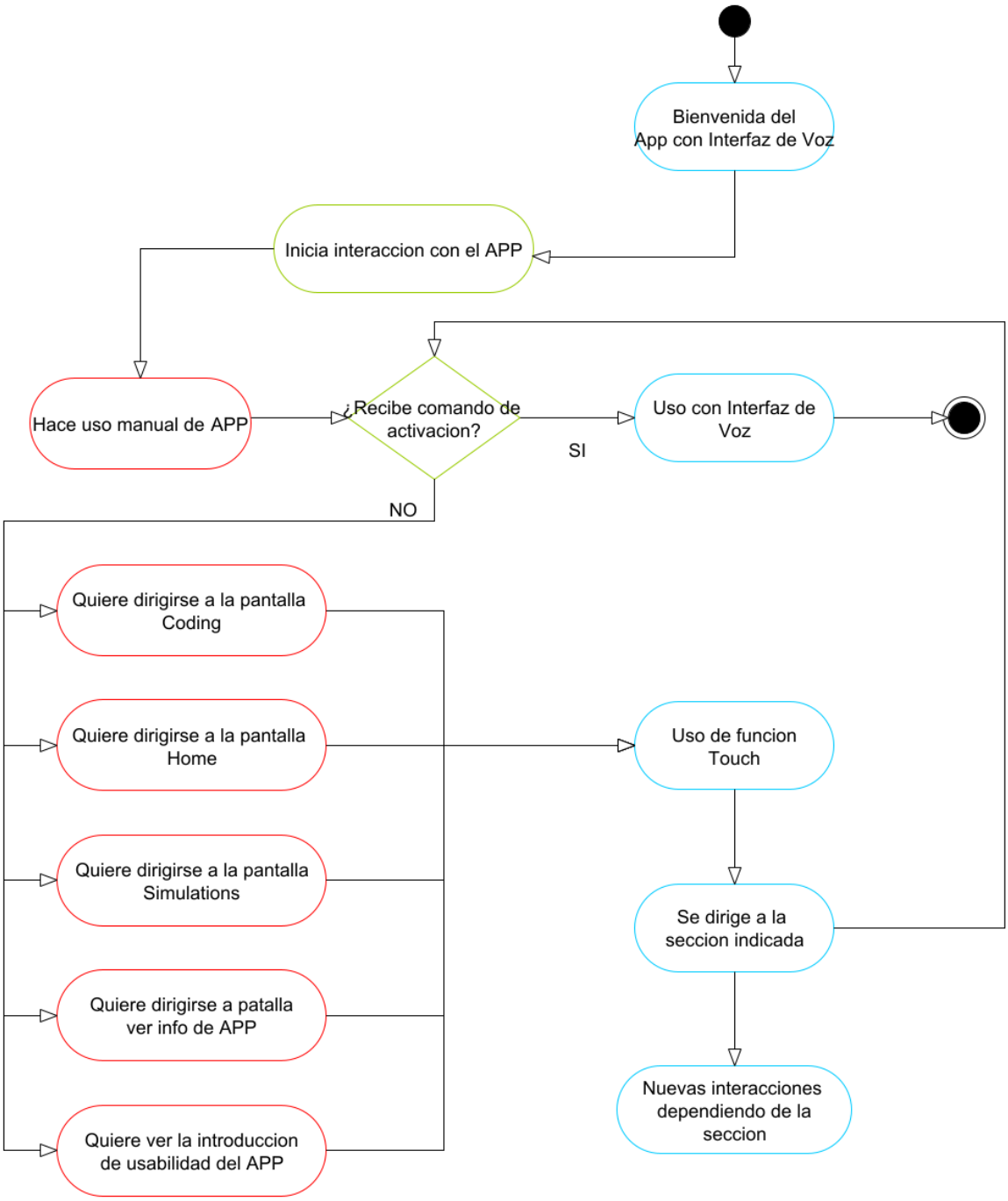


Figura 35. Diagrama de interacción para navegación por usuario sin discapacidad motriz.
Elaboración propia.

- Uso de APP dentro de la sección Coding-Usuario Discapacitado

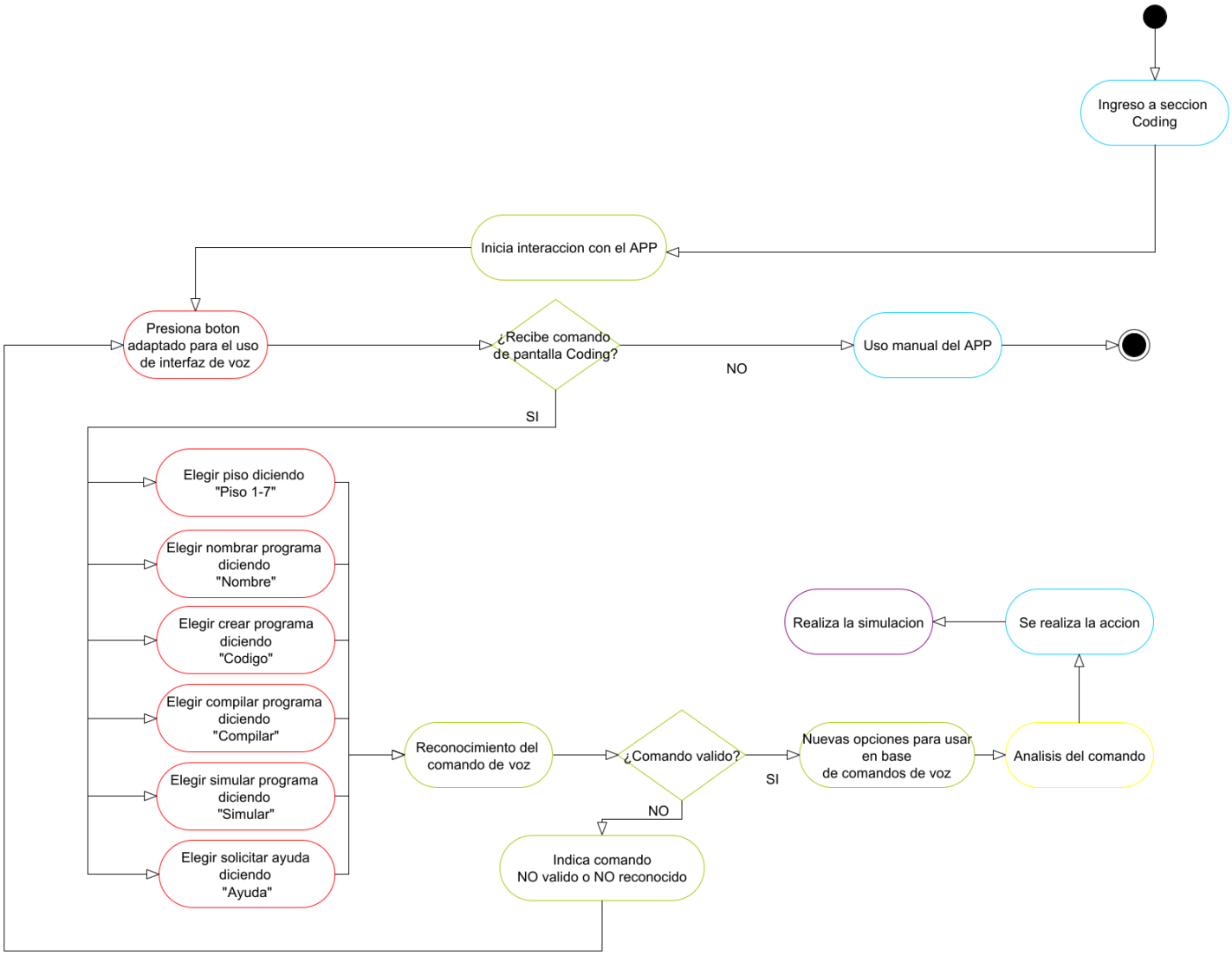


Figura 36. Diagrama de interacción en sección Coding por usuario con discapacidad motriz.
Elaboración propia.

- Uso de APP dentro de la sección Coding-Usuario Común

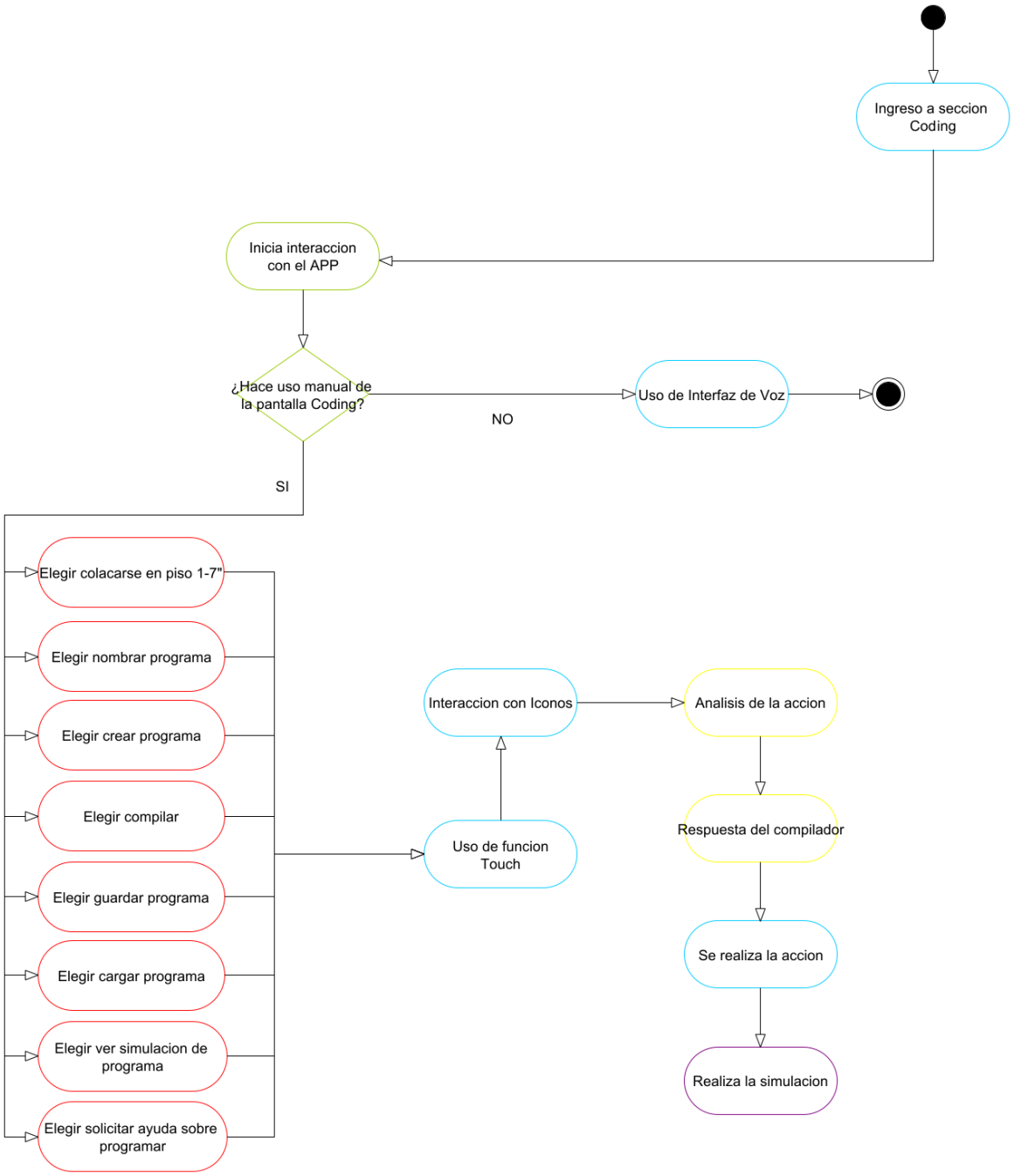


Figura 37. Diagrama de interacción en pantalla Coding por usuario sin discapacidad motriz.
Elaboración propia.

- Uso de APP dentro de la sección Simulations-Usuario Discapacitado

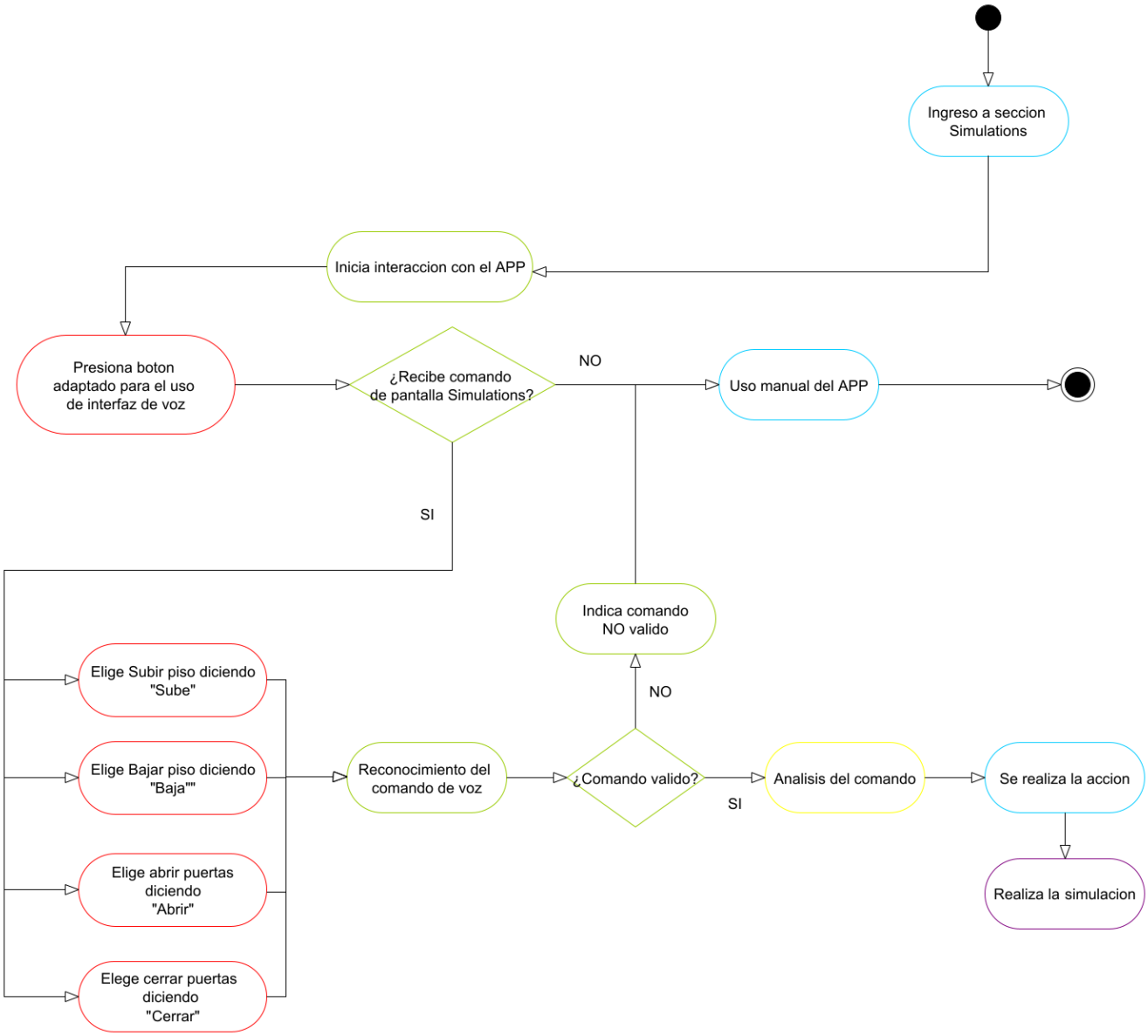


Figura 38. Diagrama de interacción en pantalla Simulations por usuario sin discapacidad motriz.
Elaboración propia.

- Uso de APP dentro de la sección Simulations-Usuario común

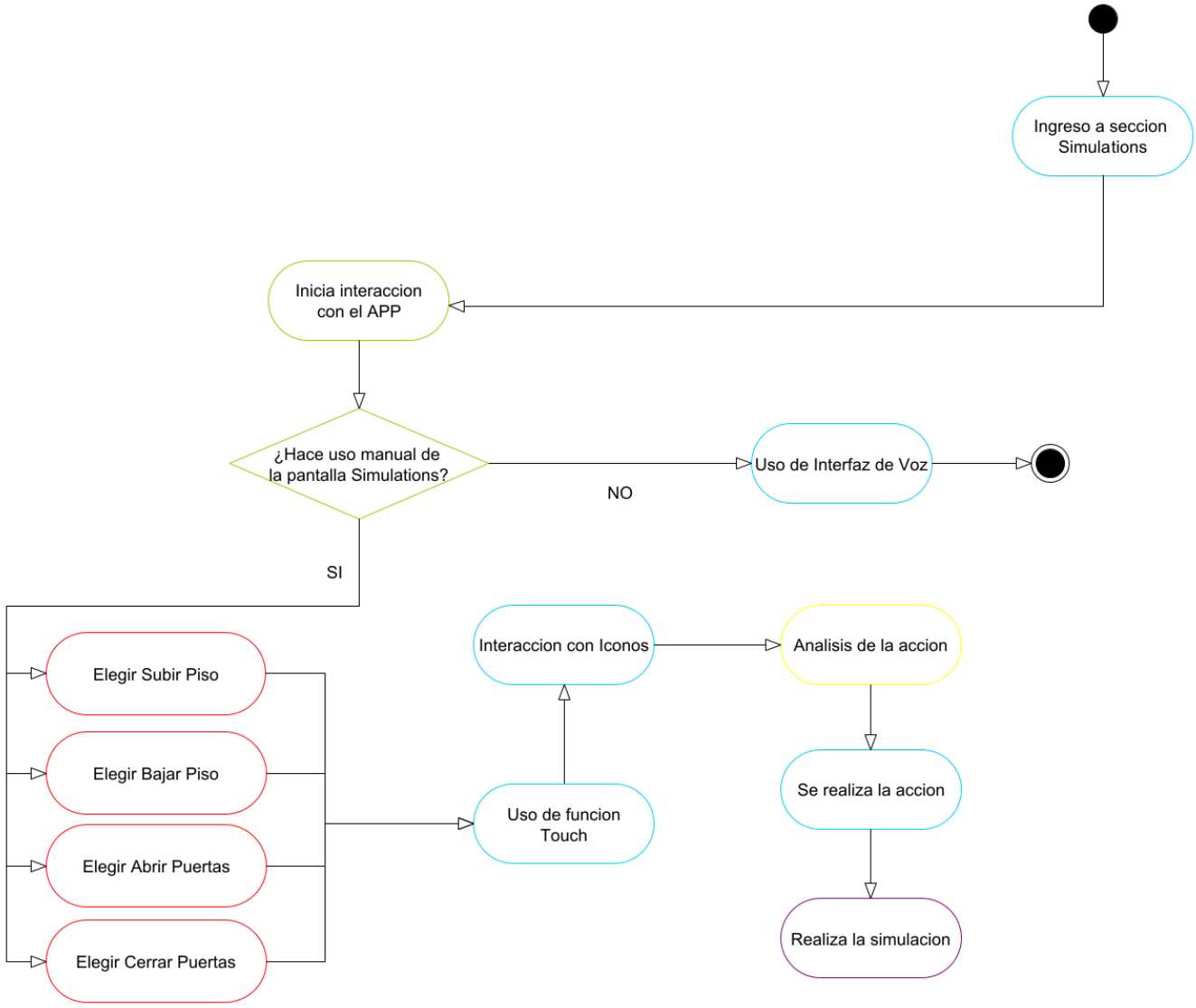


Figura 39. Diagrama de interacción en pantalla Simulations por usuario sin discapacidad motriz.
Elaboración propia.

Tabla [4] descripción de comandos para comunicación con hardware independiente:

Leyenda: "\s" = Espacio en blanco. "\n" = Enter.
"+" = Al menos 1 o más. "*" = Cero o muchos.

Acción	Comando	Expresión Regular	Identificador DTMF
Inicio	I, i	"\s*" (I, i) "\n+"	dtmf_12
Fin	F, f	"\s*" (F, f) "\n*"	dtmf_d
Subir	S, s	"\s*" (S, s) "\s+" (1-6) "\n+"	dtmf_2
Bajar	B, b	"\s*" (B, b) "\s+" (1-6) "\n+"	dtmf_1
Parar	P, p	"\s*" (P, p) "\s+" (1-9) "\n+"	dtmf_3
Abrir-Cerrar	A, a	"\s*" (A, a) "\s+" (1-9) "\n+"-	dtmf_8 – dtmf_4

Tabla 4. Descripción de comandos de Educatrónicapp para comunicarse con hardware independiente.

Autómatas basados en las expresiones regulares de los comandos:

Comando Inicio

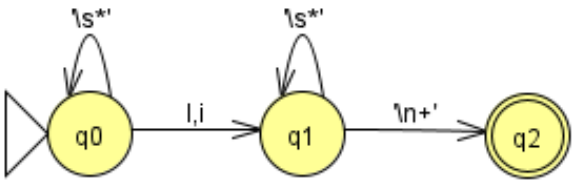


Figura 40. Expresión regular de comando Inicio.
Elaboración propia.

Comando Fin

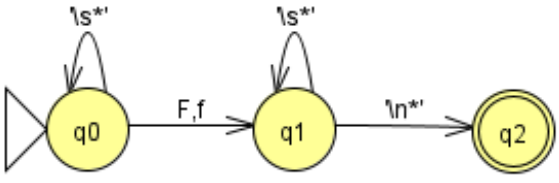


Figura 41. Expresión regular de comando Fin.
Elaboración propia.

Comando Parar

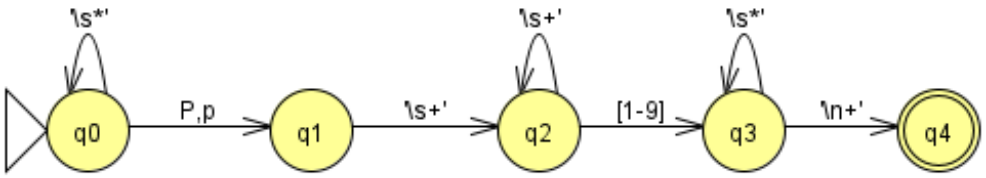


Figura 42. Expresión regular de comando Parar.
Elaboración propia.

Comando Abrir-Cerrar

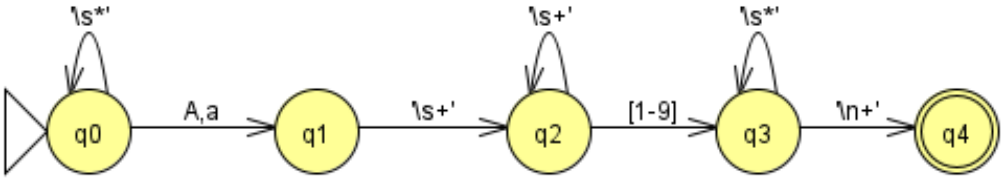


Figura 43. Expresión regular de comando Abrir-Cerrar.
Elaboración propia.

Comando Subir

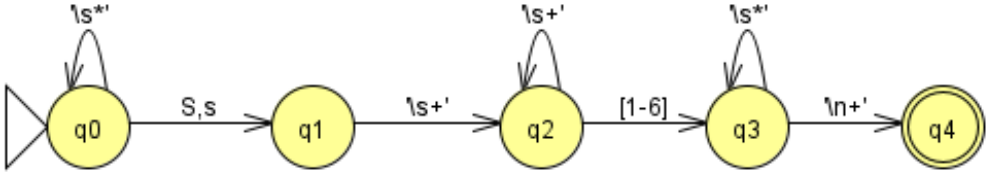


Figura 44. Expresión regular de comando Subir.
Elaboración propia.

Comando Bajar

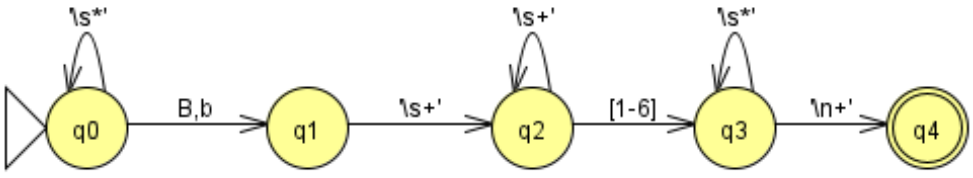


Figura 45. Expresión regular de comando Bajar.
Elaboración propia.

COMPONENTES DE TRABAJO

Ordenador	
Elemento	Descripción
Sistema Operativo	Windows 10
Procesador	Intel Core i5
Memoria RAM	16 GB
Arquitectura	x64

Dispositivo Móvil 1	
Elemento	Descripción
Sistema Operativo	IOS 17.2
Procesador	A15 Bionic
Memoria RAM	4 GB
Arquitectura	x64

Dispositivo Móvil 2	
Elemento	Descripción
Sistema Operativo	Android 16.6
Procesador	Octa Core
Memoria RAM	6 GB
Arquitectura	x64

Dispositivo Móvil 3	
Elemento	Descripción
Sistema Operativo	Emulador Android 11
Procesador	Exynos 1380
Memoria RAM	1536 MB
Arquitectura	x86

1. Para comenzar nuestro proyecto se requiere instalar la versión más reciente de Node.JS y NPM.

2. Para instalar la línea de comandos de Expo CLI se debe ingresar el siguiente comando en la terminal, Figura [46].

```
1  npm install --global expo-cl
```

Figura 46. Comando para instalar ExpoCLI.
Elaboración propia.

3. Para crear nuestro proyecto se utilizan los siguientes comandos, Figura [47].

```
1  npx create-expo-app Educatronicapp // Creacion del proyecto "Educatronicapp"
2  cd Educatronicapp // Ingresamos a la carpeta del proyecto
3  npx expo start // Inicializamos el proyecto
```

Figura 47. Comandos para crear proyecto Educatrónicapp.
Elaboración propia.

4. Estructura del Proyecto, Figura [48]

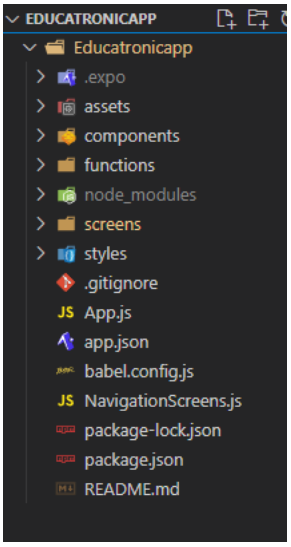


Figura 48. Directorio del proyecto.
Elaboración propia.

El directorio del proyecto Educatrónicapp contiene diferentes archivos y carpetas, a continuación, se explicará de forma breve el funcionamiento de cada una de ellas:

- ***Assets:*** Contiene recursos de la aplicación como imágenes, audios, o archivos que necesitemos utilizar en nuestra aplicación.
- ***Node_modules:*** Contiene los distintos paquetes NPM instalados en el proyecto.
- ***.gitignore:*** Archivo para un control de versiones del proyecto específico de Git, permite omitir ciertos ficheros del proyecto.
- ***App.js:*** Archivo principal de nuestro proyecto, este archivo será el primero en mostrarse al iniciar nuestra aplicación móvil en un dispositivo.
- ***App.json:*** Archivo que contiene información sobre el proyecto como el nombre, versión de Expo, sus plataformas soportadas y algunas configuraciones básicas.
- ***Components:*** Contiene componentes principales para reutilizar.
- ***Functions:*** Contiene funciones principales de sonido y de renderización.
- ***NavigatioScreens.js:*** Este archivo se encarga del enrutamiento de las diferentes pantallas de la aplicación.
- ***Screens:*** Contiene los archivos principales de pantallas de Educatrónicapp.
- ***Styles:*** Estilos de color, tamaño, posicionamiento de los elementos de las pantallas principales.
- ***Babel.config.js:*** Es un archivo de configuración Babel nos permite transformar código JS de última generación a un código de JavaScript que Node.js pueda entender.
- ***Package-lock.json:*** Fichero que muestra los paquetes, dependencias y versiones instaladas en el proyecto de forma más específica.
- ***Package.json:*** Fichero que muestra los paquetes, dependencias y versiones instaladas en el proyecto.

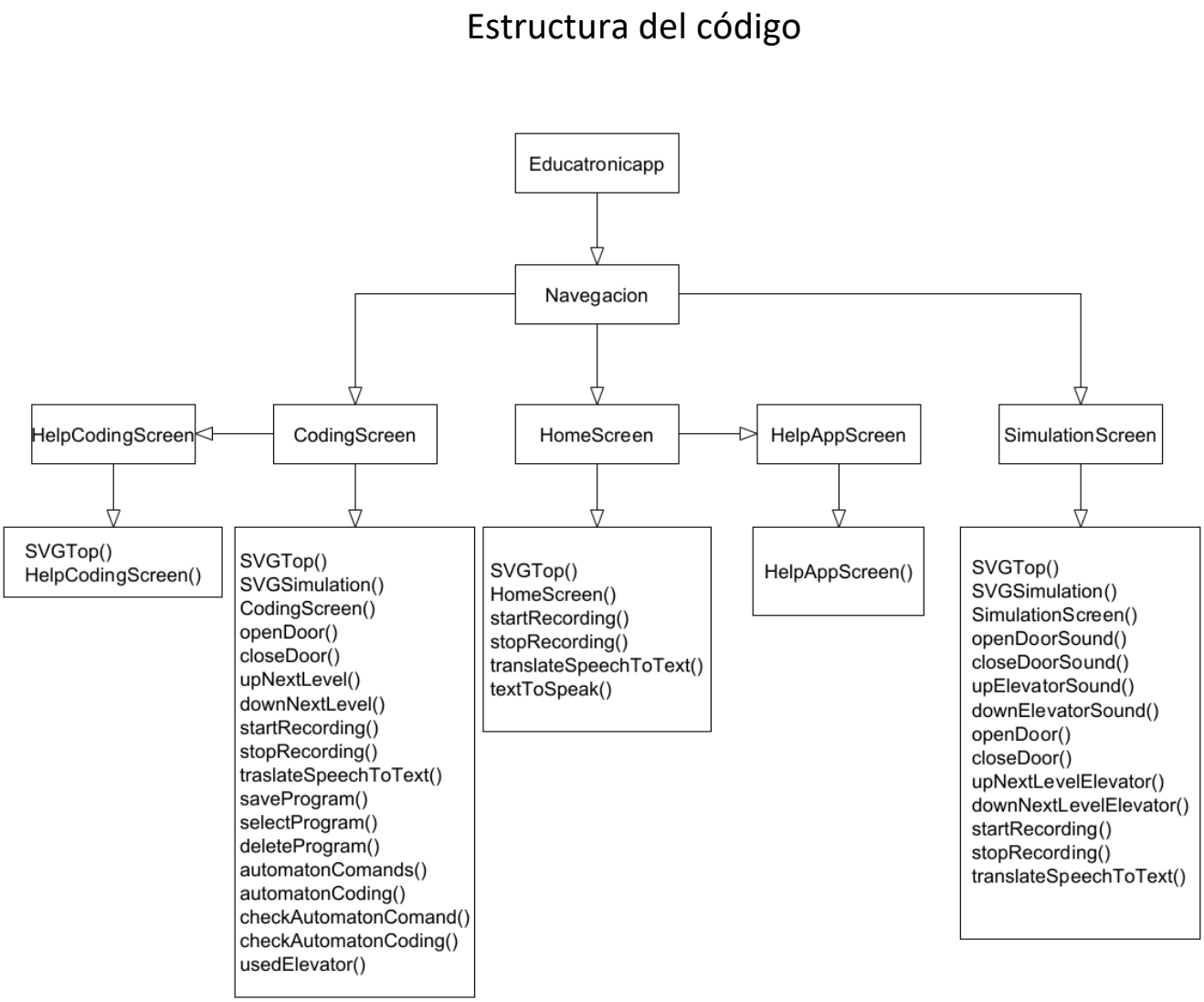


Figura 49. Estructura de código proyecto Educatrónicapp.
Elaboración propia.

Código

Si se desea ver a fondo el código de “Educatrónicapp” se puede visitar el enlace del repositorio en GitHub.

Enlace de GitHub para ver el codigo:
https://github.com/EdgarHdzHdz17/Educatronicapp

Figura 50. Logo Educatrónicapp.
Elaborada por: Karla Daniela Cruz Núñez

Funciones principales renderizados elementos SVG

SVGTop()

```
//Componente SVG Top
function SVGTop() {
return (
      <Svg
        xmlns="http://www.w3.org/2000/svg"
        width={321}
        height={70}
        fill="none"
      >
        <Path
          fill="#56D0F7"
          d="M0 25C0 11.193 11.193 0 25 0h271c13.807 0 25 11.193 25 25v53H0V25Z"
        />
        <Path
          fill="#56D0F7"
          d="M0 25C0 11.193 11.193 0 25 0h271c13.807 0 25 11.193 25 25v53H0V25Z"
        />
        ……
    </Svg>
   );
  }
```

1. *SVGTop()* es la función que se encarga de renderizar gráficos vectoriales para darle el estilo decorativo del prediseño de Educatrónicapp.
2. Se basa proporcionarle en codigo JavaScript el tamaño del elemento y posición vectorial, gracias a esto podemos modelar cualquier tipo de figura con un color, tamaño, posición personalizada dentro de un componente de Expo React Native.

SVGSimulation()

```
function SVGSimulation({ pathWidth, pathHeight,levelXElevator,statusDoor}) {

  //Componentes del edificio
  const widthBuild = pathWidth * 0.5; // Ancho del edificio
  const heightBuild = pathHeight * 1 ; // Alto del edificio

  const widthWindow = widthBuild * 0.3; // Ancho de la ventana
  const heightWindow = heightBuild * 0.09;// Alto de la ventana

  const widthWindowLevel1 = widthBuild * 0.15; // Ancho de la ventana del nivel 1
  const heightWindowLevel1 = heightBuild * 0.15;// Alto de la ventana del nivel 1

  const heightDoorLevel1= heightBuild * 0.25;// Alto de la puerta del nivel 1
  const widthDoorLevel1 = widthBuild * 0.15; // Ancho de la puerta del nivel 1

  const window1PositionX = ((widthBuild - widthWindow) / 2) - (widthBuild * 0.20);//Posicion de las
ventanas izquierdas en X
  const window2PositionX = ((widthBuild + widthWindow) / 2) - (widthBuild * 0.10);//Posicion de las
ventanas derechas en X

  const door1PositionX = ((widthBuild - widthWindow) / 2) - (widthBuild * 0.01);//Posicion de la puerta
izquierda en X
  const door2PositionX = ((widthBuild + widthWindow) / 2) - (widthBuild * 0.15);//Posicion de la puerta
derecha en X

  const window1Level1PositionX = ((widthBuild - widthWindow) / 2) - (widthBuild * 0.25);//Posicion de la
ventana izquierda del nivel 1 en X
  const window2Level1PositionX = ((widthBuild + widthWindow) / 2) - (widthBuild * -0.10);//Posicion de la
ventana derecha del nivel 1 en X

  const windowsLevel1PositionY =  heightBuild * 0.80; // 65%

  //Posicion de las ventanas dentro del edificio
  const windowLevel7 = heightBuild * 0.10; // 10%
  const windowLevel6 = heightBuild * 0.20; // 20%
  const windowLevel5 = heightBuild * 0.30; // 30%
  const windowLevel4 = heightBuild * 0.40; // 40%
  const windowLevel3 = heightBuild * 0.50; // 50%
  const windowLevel2 = heightBuild * 0.60; // 60%
  const windowLevel1 = heightBuild * 0.74; // 74%

  //Componentes del elevador
  const widthElevator = pathWidth * 0.15; // Ancho del elevador
  const heightElevator = pathHeight * 0.2 ; // Alto del elevador

  const PositionXElevator = (pathWidth-widthBuild*0.6)//Posicion del elevador en el eje X

  const heightDoorElevator= heightElevator * 0.85;// Alto de la puerta del elevador
  const widthDoorElevator = widthElevator * 0.60; // Ancho de la puerta del elevador

  const halfElevatorHeight = heightElevator * 0.5;//Calcula la mitad de la altura del elevador

  const doorPositionYElevator = levelXElevator - (halfElevatorHeight * -0.30);//La posicion en Y de la
puerta del elevador se desplaza cada que cambia la posicion el elevador
//Componentes del marco de la puerta
  const heightFrameDoorElevator= heightElevator * 0.88;// Alto del contorno de la puerta
  const widthFrameDoorElevator = widthElevator * 0.70; // Ancho del contorno de la puerta
  const positionFrameDoor = (pathWidth-widthBuild*0.555)//Posicion del marco de la puerta en X
  const doorFramePositionYElevator = levelXElevator - (halfElevatorHeight * -0.23);////Posicion del marco
en Y
```

```
//Componentes del indicador del elevador
  const heightIndicatorElevator= heightElevator * 0.08;// Alto del indicador de la puerta
  const widthIndicatorElevator = widthElevator * 0.50; // Ancho del indicador de la puerta
  const positionIndicatorElevator = (pathWidth-widthBuild*0.52)//Posicion del indicador de la puerta en X
  const indicatorPositionYElevator = levelXElevator - (halfElevatorHeight * -0.03);//Posicion del
indicador en Y

  //Componentes del control del elevador
  const heightControlElevator= heightElevator * 0.08;// Alto del control de la puerta
  const widthControlElevator = widthElevator * 0.08; // Ancho del control de la puerta
  const positionControlElevator = (pathWidth-widthBuild*0.335)//Posicion del control de la puerta en X
  const controlPositionYElevator = levelXElevator - (halfElevatorHeight * -0.99);////Posicion del control
en Y

  return (
    <Svg xmlns="http://www.w3.org/2000/svg" width={pathWidth} height={pathHeight} fill="none">
      <Path fill="#F2994A" d={`M0 0h${widthBuild}v${heightBuild}H0z`}/*Build*/ />
      <Path fill="#cd853f" d={`M${PositionXElevator}
${levelXElevator}h${widthElevator}v${heightElevator}H${PositionXElevator}z`}/*Elevator*/ />
      <Path fill="#a9a9a9" d={`M${positionFrameDoor}
${doorFramePositionYElevator}h${widthFrameDoorElevator}v${heightFrameDoorElevator}H${positionFrameDoor}z`}
/*Mark*/ />
      <Path fill="#a9a9a9" d={`M${positionIndicatorElevator}
${indicatorPositionYElevator}h${widthIndicatorElevator}v${heightIndicatorElevator}H${positionIndicatorElev
ator}z`}/*Indicator*/ />
      <Path fill="#a9a9a9" d={`M${positionControlElevator}
${controlPositionYElevator}h${widthControlElevator}v${heightControlElevator}H${positionControlElevator}z`}
/*Control*/ />
      <Path fill="#8b4513" d={`M${statusDoor}
${doorPositionYElevator}h${widthDoorElevator}v${heightDoorElevator}H${statusDoor}z`}/*Door*/ />
      <Path fill="#C5E4F9" d={`M${window1PositionX}
${windowLevel7}h${widthWindow}v${heightWindow}H${window1PositionX}z`}/*Level 7*/ />
      <Path fill="#C5E4F9" d={`M${window2PositionX}
${windowLevel7}h${widthWindow}v${heightWindow}H${window2PositionX}z`}/*Level 7*/ />
      <Path fill="#C5E4F9" d={`M${window1PositionX}
${windowLevel6}h${widthWindow}v${heightWindow}H${window1PositionX}z`}/*Level 6*/ />
      <Path fill="#C5E4F9" d={`M${window2PositionX}
${windowLevel6}h${widthWindow}v${heightWindow}H${window2PositionX}z`}/*Level 6*/ />
      <Path fill="#C5E4F9" d={`M${window1PositionX}
${windowLevel5}h${widthWindow}v${heightWindow}H${window1PositionX}z`}/*Level 5*/ />
      <Path fill="#C5E4F9" d={`M${window2PositionX}
${windowLevel5}h${widthWindow}v${heightWindow}H${window2PositionX}z`}/*Level 5*/ />
      <Path fill="#C5E4F9" d={`M${window1PositionX}
${windowLevel4}h${widthWindow}v${heightWindow}H${window1PositionX}z`}/*Level 4*/ />
      <Path fill="#C5E4F9" d={`M${window2PositionX}
${windowLevel4}h${widthWindow}v${heightWindow}H${window2PositionX}z`}/*Level 4*/ />
      <Path fill="#C5E4F9" d={`M${window1PositionX}
${windowLevel3}h${widthWindow}v${heightWindow}H${window1PositionX}z`}/*Level 3*/ />
      <Path fill="#C5E4F9" d={`M${window2PositionX}
${windowLevel3}h${widthWindow}v${heightWindow}H${window2PositionX}z`}/*Level 3*/ />
      <Path fill="#C5E4F9" d={`M${window1PositionX}
${windowLevel2}h${widthWindow}v${heightWindow}H${window1PositionX}z`}/*Level 2*/ />
      <Path fill="#C5E4F9" d={`M${window2PositionX}
${windowLevel2}h${widthWindow}v${heightWindow}H${window2PositionX}z`}/*Level 2*/ />
      <Path fill="#C5E4F9" d={`M${door1PositionX}
${windowLevel1}h${widthDoorLevel1}v${heightDoorLevel1}H${door1PositionX}z`}/*Level 1*//>
      <Path fill="#C5E4F9" d={`M${door2PositionX}
${windowLevel1}h${widthDoorLevel1}v${heightDoorLevel1}H${door2PositionX}z`}/*Level 1*/ />
      <Path fill="#C5E4F9" d={`M${window1Level1PositionX}
${windowsLevel1PositionY}h${widthWindowLevel1}v${heightWindowLevel1}H${window1Level1PositionX}z`}/*Level
1*//>
      <Path fill="#C5E4F9" d={`M${window2Level1PositionX}
${windowsLevel1PositionY}h${widthWindowLevel1}v${heightWindowLevel1}H${window2Level1PositionX}z`}/*Level
1*//>
    </Svg>
  );
}
```

1. *SVGSimulation()* Es la función encargada de renderizar gráficos vectoriales para representar los elementos visuales de un elevador, un edificio, ventanas y una puerta, estos serán los encargados para fungir como una guía de simulación a un programa creado.
2. Se define tamaños, posiciones iniciales de cada elemento, en función a parámetros calculados como el ancho y alto de la pantalla, esto es para hacer las dimensiones de cada elemento de forma responsiva.
3. Se define colores y coordenadas a colocarse, algunas coordenadas cambian en varianza al comportamiento de cada programa creado.

Funciones principales de reproducción de sonidos DTMF

upElevatorSound

```
//Reproducir sonido para subir
  const upElevatorSound = async () => {
    try {
      const soundObject1 = new Audio.Sound();
      await soundObject1.loadAsync(require('../audio/dtmf_2.wav'));
      await soundObject1.playAsync();

      // Esperamos 300 ms
      await new Promise(resolve => setTimeout(resolve, 300));

      const soundObject2 = new Audio.Sound();
      await soundObject2.loadAsync(require('../audio/dtmf_3.wav'));
      await soundObject2.playAsync();
    } catch (error) {
      console.error('Error reproduciendo el sonido:', error);
    }
  };
```

downElevatorSound

```
//Reproducir sonido para bajar
  const downElevatorSound = async () => {
    try {
      const soundObject1 = new Audio.Sound();
      await soundObject1.loadAsync(require('../assets/audio/dtmf_1.wav'));
      await soundObject1.playAsync();

      // Esperamos 300 ms
      await new Promise(resolve => setTimeout(resolve, 300));

      const soundObject2 = new Audio.Sound();
      await soundObject2.loadAsync(require('../assets/audio/dtmf_3.wav'));
      await soundObject2.playAsync();
    } catch (error) {
      console.error('Error reproduciendo el sonido:', error);
    }
  };
```

1. Se crea una función de reproducción asíncrona, en la cual se reproducen 2 sonidos en secuencia, entre cada sonido hay un tiempo de retraso para emitir el siguiente sonido.
2. Los sonidos son extraídos del directorio de archivos dentro del proyecto, además corresponden a la tabla definida de nuestros comandos para subir y bajar el elevador.

openDoorSound

```
//Reproducir sonido para abrir
  const openDoorSound = async () => {
    try {
      const soundObject1 = new Audio.Sound();
      await soundObject1.loadAsync(require('../assets/audio/dtmf_8.wav'));
      await soundObject1.playAsync();

      // Esperamos 300 ms
      await new Promise(resolve => setTimeout(resolve, 300));

      const soundObject2 = new Audio.Sound();
      await soundObject2.loadAsync(require('../assets/audio/dtmf_3.wav'));
      await soundObject2.playAsync();
    } catch (error) {
      console.error('Error reproduciendo el sonido:', error);
    }
  };
```

closeDoorSound

```
//Reproducir sonido para cerrar
  const closeDoorSound = async () => {
    try {
      const soundObject1 = new Audio.Sound();
      await soundObject1.loadAsync(require('../assets/audio/dtmf_4.wav'));
      await soundObject1.playAsync();

      // Esperamos 300 ms
      await new Promise(resolve => setTimeout(resolve, 300));

      const soundObject2 = new Audio.Sound();
      await soundObject2.loadAsync(require('../assets/audio/dtmf_3.wav'));
      await soundObject2.playAsync();
    } catch (error) {
      console.error('Error reproduciendo el sonido:', error);
    }
  };
```

1. Se crea una función de reproducción asíncrona, en la cual se reproducen 2 sonidos en secuencia, entre cada sonido hay un tiempo de retraso para emitir el siguiente sonido.
2. Los sonidos son extraídos del directorio de archivos dentro del proyecto, además corresponden a la tabla definida de nuestros comandos para abrir y cerrar la puerta del elevador.

Funciones principales de reconocimiento de voz

startRecording()

```
//Funcion para iniciar grabacion
async function startRecording() {
    try {//Permisos de microfono
      const permission = await Audio.requestPermissionsAsync();

      if (permission.status === "granted") {
        await Audio.setAudioModeAsync({
          allowsRecordingIOS: true,
          playsInSilentModeIOS: true,
          interruptionModeIOS: 0,
        });
        //Configuracion del audio
        const recordingOptions = {
          android: {
            extension: '.wav',
            outputFormat: Audio.RECORDING_OPTION_ANDROID_OUTPUT_FORMAT_WAV,
            audioEncoder: Audio.RECORDING_OPTION_ANDROID_AUDIO_ENCODER_ACC,
            sampleRate: 44100,
            numberOfChannels: 2,
            bitRate: 128000,
          },
          ios: {
            extension: '.wav',
            audioQuality: Audio.RECORDING_OPTION_IOS_AUDIO_QUALITY_HIGH,
            sampleRate: 44100,
            numberOfChannels: 2,
            bitRate: 128000,
            linearPCMBitDepth: 16,
            linearPCMIsBigEndian: false,
            linearPCMIsFloat: false,
          },
        };
        const { recording } = await Audio.Recording.createAsync(recordingOptions);
        setRecording(recording);
      } else {
        console.error('No se aceptó los permisos');
      }
    } catch (err) {
      console.error('Fallo en accesibilidad', err);
    }
  }
```

1. *startRecording()* es la función encargada de iniciar la grabación de un audio, lo que primero genera es la solicitud de permitir realizar una grabación de audio en la aplicación.
2. Se verifica si los permisos fueron aceptados y continua con la grabación de audio.
3. Se establece la configuración del audio dentro de cada sistema operativo, aquí se especifica el formato de salida, calidad del audio, frecuencia de muestreo, tasa de bits, toda esta configuración es proporcionada por la documentación de Expo Speech.
4. Se crea la grabación y se almacena estableciéndola como la grabación actual, donde setRecording (recording) es un estado que varía para realizar la acción *stopRecording(),* esta manipulada por un botón dentro de Educatrónica.

stopRecording()

```
//Detencion de grabacion
  async function stopRecording() {
    setRecording(undefined);
    await recording.stopAndUnloadAsync();
    await Audio.setAudioModeAsync(
      {
        allowsRecordingIOS: false,//Evitar que IOS mande al cualquier reproduccion al auricular
      }
    );
    let updatedRecordings = [...recordings];
    //Nueva localizacion del archivo de audio
    const fileUri = `${FileSystem.documentDirectory}recording${Date.now()}.wav`;

    await FileSystem.copyAsync({
      from: recording.getURI(),
      to: fileUri,
    });

    const { sound} = await recording.createNewLoadedSoundAsync();
    updatedRecordings.push({
      sound: sound,
      file: fileUri,
    });

    setRecordings(updatedRecordings);//Actualizamos la lista de grabacione
    translateSpeechToText(fileUri);//Una vez que se termina de grabar se manda a llamar el API
    setRecordings([]);//Limpiamos la lista de grabaciones

  }
```

1. *stopRecording()* es una función para detener una grabación en curso, para que esta función sea ejecutada es necesaria una acción preliminar llamada *startRecording()*.
2. Se detiene la grabación actual y se configura el modo de salida en el sistema operativo IOS para evitar una reproducción de audio por medio del auricular, acción emitida por la biblioteca Expo Speech.
3. Se crea una copia del registro de grabaciones existentes.
4. Se genera una ubicación del archivo grabado.
5. Actualizamos la lista de grabaciones.
6. Se llama a una función *translateSpeechToText()* automaticamente para realizar una llamada al API SpeechToText de OpenAI una vez detenida la grabación.
7. Se limpia la lista de grabaciones.

translateSpeechToText()

```
async function translateSpeechToText(fileUri) {

    try {
      await Audio.setAudioModeAsync({
        allowsRecordingIOS: false,
      });
      const formData = new FormData();
      formData.append('file', {
        uri: fileUri,
        type: 'audio/x-wav',
        name: 'audio.wav',
      });
      formData.append('model', 'whisper-1');
      formData.append('response_format', 'text');
      formData.append('language', 'es');
      const response = await axios.post('https://api.openai.com/v1/audio/transcriptions', formData, {
        headers: {
          'Content-Type': 'multipart/form-data',
          'Authorization': 'Bearer APIKEY',
        },
      });

      await Audio.setAudioModeAsync(
        {
          allowsRecordingIOS: false,
        }
      );

      const resultSpeech = response.data;//Resultado del reconocimiento del API

      console.log(resultSpeech);//Resultado del reconocimiento del API en concola
      //Comandos solo para navegacion

      //Comando para activar ir Ver simulacion por voz
      if
(resultSpeech.includes("Simular")||resultSpeech.includes("simular")||resultSpeech.includes("SIMULAR")) {
        navigation.navigate("Simulations");
      }
      //Comando para activar ir Programar por voz
      if
(resultSpeech.includes("Programar")||resultSpeech.includes("programar")||resultSpeech.includes("PROGRAMAR"
)) {
        navigation.navigate("Coding");
        Speech.speak("Comando Programar detectado");
      }
      //Comando para activar ir Inicio por voz
      if
(resultSpeech.includes("Inicio")||resultSpeech.includes("inicio"||resultSpeech.includes("INICIO"))) {
        navigation.navigate("Home");
        Speech.speak("Comando Inicio detectado");
      }

      //Eliminamos el archivo del App
      try {
        await FileSystem.deleteAsync(fileUri);
        console.log(`El archivo ${fileUri} se ha eliminado correctamente.`);
      } catch (error) {
        console.error(`Error al eliminar el archivo ${fileUri}: ${error.message}`);
      }

    } catch (error) {
      console.error('Fallo de transcripcion', {...error});
      Alert.alert("El reconocimiento de voz aun no es posible en Android o hubo un error externo.");
    }
```

```
      if (
        !resultSpeech.includes("simular") &&
        !resultSpeech.includes("Simular") &&
        !resultSpeech.includes("SIMULAR") &&
        !resultSpeech.includes("programar") &&
        !resultSpeech.includes("Programar") &&
        !resultSpeech.includes("PROGRAMAR") &&
        !resultSpeech.includes("inicio") &&
        !resultSpeech.includes("Inicio") &&
        !resultSpeech.includes("INICIO") &&
        !resultSpeech.includes("tutorial") &&
        !resultSpeech.includes("Tutorial") &&
        !resultSpeech.includes("TUTORIAL")
      ) {
        Speech.speak("Comando de voz no válido o posiblemente no lo detecte bien");
      }

      //Eliminamos el archivo del App
      try {
        await FileSystem.deleteAsync(fileUri);
        console.log(`El archivo ${fileUri} se ha eliminado correctamente.`);
      } catch (error) {
        console.error(`Error al eliminar el archivo ${fileUri}: ${error.message}`);
      }

    } catch (error) {
      console.error('Fallo de transcripcion', {...error});
    }
  }
```

1. *translateSpeechToText()* es la función encargada de hacer la petición de una transcripción de audio a texto al API Speech to Text de Open AI.
2. Se le pasa como argumento la ruta del archivo(fileUri) recién grabado.
3. Se verifica pasar el audio en un formato .wav, este formato es compatible con el API.
4. Se utiliza un protocolo HTTPS para realizar una petición POST por medio de AXIOS al API con la configuración proporcionada por la documentación de Speech to Text, incluidos parámetros de configuración.
5. Se muestra en consola el resultado de la transcripción del API, para almacenarla como una variable resultSpeech.
6. Se hace un filtro para que si la variable resultSpeech contiene cierta cadena de texto se realice una acción diferente en Educatrónicapp, cada filtro puede variar dependiendo de los comandos definidos dentro de cada pantalla.
7. Para controlar el almacenamiento de Educatrónicapp se elimina el audio dentro de la aplicación después de haber recibido respuesta del API.

recordButtonContainer

```
<TouchableOpacity style={styles.recordButtonContainer }>
        <FontAwesome5 name={recording ?"microphone-slash":"microphone"} size={35} color="#f0ffff"
onPress={recording ? stopRecording : startRecording}/>
      </TouchableOpacity>
```

1. recordButtonContainer será el elemento principal para realizar las acciones *startRecording()* y *stopRecording(),* este botón esta renderizado por 2 iconos que varían dependiendo de la función que se está realizando.

Función principal de comandos en lenguaje natural

```
function automatonComands(input) {
    let currentState = 0; // El estado inicial es 0
    setResult("Ingresa tus comandos"); //El valor de result es NULL
    for (let i = 0; i < input.length; i++) {
      const char = input.charAt(i);
      switch (currentState) {
        case 0: // Estado 0
          if (char === "S" || char === "s") {
            currentState = 1; // Si char es S o s pasa al estado 1
            setResult("Comando Subir detectado");
          } else if (char === "B" || char === "b") {
            currentState = 1; //Si char es B o b pasa al estado 1
            setResult("Comando Bajar detectado");
          } else if (char === "P" || char === "p") {
            currentState = 3; //Si char es P o p pasa al estado 3
            setResult("Comando Parar detectado");
          } else if (char === "A" || char === "a") {
            currentState = 3; //Si char es A o a pasa al estado 3
            setResult("Comando Abrir detectado");
          } else if (char === "I" || char === "i") {
            currentState = 5; //Si char es I o i pasa al estado 5
            setResult("Comando Inicio detectado");
          } else if (char === "F" || char === "f") {
            currentState = 6; //Si char es F o f pasa al estado 6
            setResult("Comando Fin detectado");
          } else if (char === "\n") {
            setResult("Comando Enter detectado");
            continue; //Continua en el estado si hay espacios antes de cualquier caracter
          } else if (char === " ") {
            continue; //Continua en el estado si hay espacios antes de cualquier caracter
          } else {
            setResult(`Invalido: El símbolo '${char}' no pertenece al lenguaje de comandos`);
            return false; // Si no es invalido
          }
          break;
        case 1: // Estado 1
          if (char === " ") {
            currentState = 2; // Si hay un espacio o mas, pasa al estado 2
            setResult("Sintaxis valida");
          } else {
            setResult("Invalido: Debe haber un espacio despues del comando");
            return false; // Si no hay un espacio, el input es inválido
          }
          break;
```

```
      case 2: // Estado 2
        if (char > 0 && char < 7) {
          currentState = 5; // Si char es mayor a 0 y menor 7, pasar al estado 5
          setResult(`Piso '${char}' detectado`);
        } else if (char === " ") {
          continue; //Continua en el estado si hay espacios antes de elegir piso
        } else {
          setResult(`Invalido: El numero '${char}' no corresponde a un piso`);
          return false; // Si no hay un numero mayor y menor  de 0 a 8 regresa un invalido
        }
        break;
      case 3: // Estado 3
        if (char === " ") {
          currentState = 4; // Si hay un espacio o mas, pasa al estado 4
          setResult("Sintaxis valida");
        } else {
          setResult("Invalido: Debe haber un espacio despues del comando");
          return false; // Si no hay un espacio, el input es inválido
        }
        break;
      case 4: // Estado 4
        if (char > 0 && char < 10) {
          currentState = 5; // Si char es mayor a 0 y menor que 10, pasa al estado 5
          setResult(`Tiempo '${char}' detectado`);
        } else if (char === " ") {
          continue; //Continua en el estado si hay espacios antesde elegir el tiempo
        } else {
          setResult(`Invalido tiempo '${char}' NO valido`);
          return false; // Si no hay un numero mayor de 0 y menor que 10 regresa un invalido
        }
        break;
      case 5: // Estado 5
        if (char === "\n") {
          setResult("Sintaxis valida");
          currentState = 0; // Si char es un enter pasa al estado 0
        } else if (char === " ") {
          setResult("Sintaxis valida");
          continue; //Continua en el estado si hay espacios antes de dar enter
        } else {
          setResult("Invalido: Debe haber un enter");
          return false; // Si no hay salto de linea el input es invalido
        }
        break;
      case 6: // Estado 6
        if (char === "\n") {
          setResult("Sintaxis valida");
        } else if (char === " ") {
          setResult("Sintaxis valida");
          continue; //Continua en el estado si hay espacios antes de dar enter
        } else {
          setResult("Sintaxis invalida");
          return false; // Si no hay salto de linea el input es invalido
        }
        break;
    }
  }
}
```

La función se encarga de recibir una cadena de texto “input” como entrada y realiza un análisis sintáctico de los comandos(I,S,B,P,A,F), espacios en blanco, saltos de línea y números enteros definidos en los autómatas de la Tabla 4. El objetivo es validar la cadena de caracteres ingresados en el input, su uso se basa en un automata finito para procesar la entrada y verificar si es correcto o incorrecto, además proporciona un mensaje donde encontró un error en la sintaxis.

Función principal del elevador

```
//Funcion para usar elevador
  async function usedElevator(selectedFloor) {
    if (nameProgram === "") {
      Alert.alert("Falta nombre","Por favor, ingresa un nombre para el programa.");
    } else if (inputTextCoding === "") {
      Alert.alert("Falta Codigo", "Por favor, ingresa el codigo del programa.");
    } else {
      if (!isValidCoding) {
        Alert.alert("Error", "La compilacion NO es posible, tienes errores.");
        return;
      } else {
        console.log("----------Compilacion en proceso----------")
        setButtonDisabled(true);//Desabilitamos el boton compilar
        setCompilationInProgress(true);//Actulizamos el estado de que se esta compilando
        setIconCompile('clock-o');//Cambiamos el icono de compilar mientras se compila

        // Se reproduce el sonido según el valor de selectedFloor para subir al elevador(Robot) en el piso
indicado
        if (selectedFloor !== 1) {
          console.log('Subiendo el elevador al piso...', selectedFloor)
          await playSound(require("../assets/audio/dtmf_2.wav"), selectedFloor);
        }

        let currentFloor = selectedFloor;
        console.log("Piso elegido:", selectedFloor);
        console.log("Piso actual:", currentFloor);
        console.log("Texto ingresado:\n", inputTextCoding);

        const floorMax = 7;
        const floorMin = 1;
        let numSeg;
        let i = 0;
        let numFloors;

        //Funcion que recorre todo el inputTextCoding para representar el sonido que corresponde a cada
caracter
        while (i < inputTextCoding.length) {
          //Comando I
          if (inputTextCoding[i] === "I" || inputTextCoding[i] === "i") {
            await new Promise((resolve) => setTimeout(resolve, 1000));
            console.log("Comando Inicio detectado");
            playSound(require("../assets/audio/dtmf_12.wav"), 1);
            i++;
            //Comando S
          } else if (inputTextCoding[i] === "S" || inputTextCoding[i] === "s") {
            await new Promise((resolve) => setTimeout(resolve, 1000));
            i++; // Avanzar al siguiente caracter después de la "S" o "s"
            while (inputTextCoding[i] === " ") {
              i++; // Saltar los espacios en blanco
            }
            // Verificar si el siguiente caracter es un número
            if (!isNaN(inputTextCoding[i])) {
              numFloors = parseInt(inputTextCoding[i]);
              console.log("Comando Subir detectado, subiendo:",numFloors,"pisos");
              for (let j = 0; j < numFloors; j++) {
                await new Promise((resolve) => setTimeout(resolve, 1000)); // Esperamos 1 segundo
                if (currentFloor < floorMax) {
                  currentFloor++;
                  console.log("Subiendo a piso:", currentFloor);
                  setSelectedFloor(currentFloor);
                  upNextLevelElevator();
                  playSound(require("../assets/audio/dtmf_2.wav"), 1);
```

```
            } else {
              console.log("El elevador no puede subir más. Límite de piso alcanzado:",floorMax);
              Alert.alert("Piso maximo alcanzado ","Pasaremos al siguiente comando");
              break; // Detenemos el bucle si el elevador alcanza el piso máximo
            }
          }
        } else {
          console.log("Comando Subir detectado, pero el siguiente caracter no es un número.");
        }
        //Comando B
      } else if (inputTextCoding[i] === "B" || inputTextCoding[i] === "b") {
        await new Promise((resolve) => setTimeout(resolve, 1000));
        i++; // Avanzar al siguiente caracter después de la "B" o "b"
        while (inputTextCoding[i] === " ") {
          i++; // Saltar los espacios en blanco
        }
        // Verificar si el siguiente caracter es un número
        if (!isNaN(inputTextCoding[i])) {
          numFloors = parseInt(inputTextCoding[i]);
          console.log("Comando Bajar detectado, bajando:",numFloors,"pisos");
          for (let j = 0; j < numFloors; j++) {
            await new Promise((resolve) => setTimeout(resolve, 1000)); // Esperamos 1 segundo
            if (currentFloor > floorMin) {
              currentFloor--;
              console.log("Bajando a piso:", currentFloor);
              setSelectedFloor(currentFloor);
              downNextLevelElevator();
              playSound(require("../assets/audio/dtmf_1.wav"), 1);
            } else {
              console.log("El elevador no puede bajar más. Límite de piso alcanzado:",floorMin);
              Alert.alert("Piso minimo alcanzado ","Pasaremos al siguiente comando");
              break; // Detenemos el bucle si el elevador alcanza el piso minimo
            }
          }
        } else {
          console.log("Comando Bajar detectado, pero el siguiente caracter no es un número.");
        }
        //Comando F
      } else if (inputTextCoding[i] === "F" || inputTextCoding[i] === "f") {
        await new Promise((resolve) => setTimeout(resolve, 1000));
        console.log("Comando Fin detectado");
        playSound(require("../assets/audio/dtmf_d.wav"), 1);
        i++; // Avanzar al siguiente caracter después de la "F" o "f"
        //Comando P
      } else if (inputTextCoding[i] === "P" || inputTextCoding[i] === "p") {
        await new Promise((resolve) => setTimeout(resolve, 1000));
        i++; // Avanzar al siguiente caracter después de la "P" o "p"
        while (inputTextCoding[i] === " ") {
          i++; // Saltar los espacios en blanco
        }
        // Verificar si el siguiente caracter es un número
        if (!isNaN(inputTextCoding[i])) {
          let segInicial = 0;
          numSeg = parseInt(inputTextCoding[i]);
          console.log("Comando Parar detectado, parando:",numSeg,"segundos");
          for (let j = 0; j < numSeg; j++) {
            segInicial++;
            await new Promise((resolve) => setTimeout(resolve, 1000)); // Esperamos 1 segundo
            console.log("Pausa del:", segInicial, "segundo");
            playSound(require("../assets/audio/dtmf_3.wav"), 1);
          }
        } else {
          console.log("Comando Parar detectado, pero el siguiente caracter no es un número.");
        }
```

```
          //Comando A
        } else if (inputTextCoding[i] === "A" || inputTextCoding[i] === "a") {
          await new Promise((resolve) => setTimeout(resolve, 1000));
          console.log("Comando Abrir detectado, abriendo puertas");
          playSound(require("../assets/audio/dtmf_8.wav"), 1);
          await new Promise((resolve) => setTimeout(resolve, 1000));
          console.log("Deteniendo puertas");
          openDoor();
          playSound(require("../assets/audio/dtmf_3.wav"), 1);
          i++; // Avanzar al siguiente caracter después de la "A" o "a"
          while (inputTextCoding[i] === " ") {
            i++; // Saltar los espacios en blanco
          }
          // Verificar si el siguiente caracter es un número
          if (!isNaN(inputTextCoding[i])) {
            await new Promise((resolve) => setTimeout(resolve, 1000));
            let segInicial = 0;
            numSeg = parseInt(inputTextCoding[i]);
            console.log("Puerta abierta por:",numSeg,"segundos");
            for (let j = 0; j < numSeg; j++) {
              segInicial++;
              await new Promise((resolve) => setTimeout(resolve, 1000)); // Esperamos 1 segundo
              console.log("Apertura del:", segInicial, "segundo");
              playSound(require("../assets/audio/dtmf_3.wav"), 1);
            }
          } else {
            console.log("Comando Abrir detectado, pero el siguiente caracter no es un número.");
          }
          await new Promise((resolve) => setTimeout(resolve, 1000));
          console.log("Cerrando puertas");
          playSound(require("../assets/audio/dtmf_4.wav"), 1);
          await new Promise((resolve) => setTimeout(resolve, 1000));
          console.log("Deteniendo puertas");
          closeDoor();
          playSound(require("../assets/audio/dtmf_3.wav"), 1);
        }
        i++;
      }
      await new Promise((resolve) => setTimeout(resolve, 1000));
      console.log("Piso final:", currentFloor); // Imprimimos el piso final
      await new Promise((resolve) => setTimeout(resolve, 1000));
      console.log("Valores restablecidos"); //Reiniciamos todo
      Alert.alert("Programa terminado","Reiniciando el elevador")
      console.log("Bajando el elevador al piso... 1");
      for (let i = currentFloor; i > 1; i--) {
        currentFloor--;
        await playSound(require("../assets/audio/dtmf_2.wav"), 1);
      }
      setSelectedFloor(1);
      setCurrentLevelXElevator(levelsElevator[0])
      setIconCompile('play-circle');
      setButtonDisabled(false);
      setCompilationInProgress(false);
      console.log("----------Fin de la Compilacion----------");
    }
  }
}
```

PRUEBA DE SISTEMA

Sistema		
Descripción:	La aplicacion móvil puede desplegarse en ambos sistemas operativos.	
Sistema Operativo:	Dispositivos IOS	Dispositivos Android
Resultado:	Exitoso	Exitoso

PRUEBAS UNITARIAS

NavigationScreens		
Descripción:	Creación de diferentes pantallas.	
Sistema Operativo:	Dispositivos IOS	Dispositivos Android
Resultado:	Exitoso	Exitoso

Componentes visuales		
Descripción:	Se crean elementos visuales como colores, botones, iconos, texto, componentes SVG.	
Sistema Operativo:	Dispositivos IOS	Dispositivos Android
Resultado:	Exitoso	Exitoso

LottieView		
Descripción:	Se renderiza un archivo tipo JSON para representar una imagen animada, llamado Lottie.	
Sistema Operativo:	Dispositivos IOS	Dispositivos Android
Resultado:	Exitoso	Exitoso

Iconos		
Descripción:	Se crea 1 icono qué reconoce cuando lo presionamos.	
Sistema Operativo:	Dispositivos IOS	Dispositivos Android
Resultado:	Exitoso	Exitoso

textToSpeak		
Descripción:	Una frase en texto es reproducida por la interfaz de voz.	
Sistema Operativo:	Dispositivos IOS	Dispositivos Android
Resultado:	Exitoso	Exitoso

Picker		
Descripción:	Componente que despliega una lista para elegir una opción.	
Sistema Operativo:	Dispositivos IOS	Dispositivos Android
Resultado:	Exitoso	Exitoso

TextInput		
Descripción:	Permite ingresar texto en un campo.	
Sistema Operativo:	Dispositivos IOS	Dispositivos Android
Resultado:	Exitoso	Exitoso

SVG		
Descripción:	Crea componentes SVG(Gráficos Vectoriales Escalables).	
Sistema Operativo:	Dispositivos IOS	Dispositivos Android
Resultado:	Exitoso	Exitoso

saveProgram()		
Descripción:	Guarda el texto ingresado en un TextInput.	
Sistema Operativo:	Dispositivos IOS	Dispositivos Android
Resultado:	Exitoso	Exitoso

selectProgram()		
Descripción:	Selecciona un elemento de una lista.	
Sistema Operativo:	Dispositivos IOS	Dispositivos Android
Resultado:	Exitoso	Exitoso

deletedProgram ()		
Descripción:	Borra un elemento de una lista y el texto ingresado en InputText.	
Sistema Operativo:	Dispositivos IOS	Dispositivos Android
Resultado:	Exitoso	Exitoso

automatonComands()		
Descripción:	Crea 1 autómata sobre un TextInput, se encarga de las reglas gramaticales definidas en el lenguaje natural.	
Sistema Operativo:	Dispositivos IOS	Dispositivos Android
Resultado:	Exitoso	Exitoso

automatonCoding()		
Descripción:	Crea 1 autómata sobre un TextInput, se encarga de mantener una estructura a seguir.	
Sistema Operativo:	Dispositivos IOS	Dispositivos Android
Resultado:	Exitoso	Exitoso

checkAutomatonComand()		
Descripción:	Es un componente que verifica lo que se escribe en un TextInput relacionado a automatonComands().	
Sistema Operativo:	Dispositivos IOS	Dispositivos Android
Resultado:	Exitoso	Exitoso

checkAutomatonCoding()		
Descripción:	Es un componente que verifica la estructura en un TextInput relacionado a automatonCoding().	
Sistema Operativo:	Dispositivos IOS	Dispositivos Android
Resultado:	Exitoso	Exitoso
playSound		
Descripción:	Emite un sonido de audio.	
Sistema Operativo:	Dispositivos IOS	Dispositivos Android
Resultado:	Exitoso	Exitoso

Alert		
Descripción:	Se crea un cuadro de texto, funcional para mostrar un mensaje.	
Sistema Operativo:	Dispositivos IOS	Dispositivos Android
Resultado:	Exitoso	Exitoso

usedElevator()		
Descripción:	Se encargará de recorrer el TextInput para realizar cierta acción correspondiente al texto ingresado.	
Sistema Operativo:	Dispositivos IOS	Dispositivos Android
Resultado:	Exitoso	Exitoso

closeDoor()		
Descripción:	Cambia la posición del elemento doorSVG en la simulación, es decir cierra la puerta.	
Sistema Operativo:	Dispositivos IOS	Dispositivos Android
Resultado:	Exitoso	Exitoso

openDoor()		
Descripción:	Cambia la posición del elemento doorSVG en la simulación, es decir abre la puerta.	
Sistema Operativo:	Dispositivos IOS	Dispositivos Android
Resultado:	Exitoso	Exitoso

upNextLevelElevator()		
Descripción:	Los elementos elevatorSVG suben al siguiente piso.	
Sistema Operativo:	Dispositivos IOS	Dispositivos Android
Resultado:	Exitoso	Exitoso

downNextLevelElevator ()		
Descripción:	Los elementos elevatorSVG bajan al siguiente piso.	
Sistema Operativo:	Dispositivos IOS	Dispositivos Android
Resultado:	Exitoso	Exitoso

startRecording()		
Descripción:	Comienza una grabación de audio.	
Sistema Operativo:	Dispositivos IOS	Dispositivos Android
Resultado:	Exitoso	Exitoso

stopRecording()		
Descripción:	Detiene la grabación de audio y guarda la localización del audio grabado.	
Sistema Operativo:	Dispositivos IOS	Dispositivos Android
Resultado:	Exitoso	Exitoso

translateSpeechToText()		
Descripción:	Manda llamar el API de OpenAI y envía el audio grabado, el API recibe el audio y envía una respuesta en forma de texto.	
Sistema Operativo:	Dispositivos IOS	Dispositivos Android
Resultado:	Exitoso	Exitoso

PRUEBAS DE COMANDOS DE VOZ

La siguiente tabla describe los comandos de voz definidos y si el comando de voz fue detectado en acierto/fallo.

Comandos	Persona 1 "Acierto /Fallo"	Persona 2 "Acierto /Fallo"	Persona 3 "Acierto /Fallo"	N. de pruebas por persona /totales	N. de comandos acertados	N. de comandos fallidos	Porcentaje de acierto promedio	Porcentaje de fallo promedio
"Programar"	18/2	19/1	18/2	20/60	55	5	91.66%	8.33%
"Casa"	20/0	19/1	18/2	20/60	57	3	95%	5%
"Simular"	20/0	20/0	19/1	20/60	59	1	98.33%	1.77%
"Nombre"	19/1	20/0	20/0	20/60	59	1	98.33%	1.77%
"Código"	19/1	20/0	20/0	20/60	59	1	98.33%	1.77%
"Piso Uno"	20/0	20/0	20/0	20/60	60	0	100%	0%
"Piso Dos"	20/0	20/0	20/0	20/60	60	0	100%	0%
"Piso Tres"	20/0	19/1	20/0	20/60	59	1	98.33%	1.77%
"Piso Cuatro"	20/0	20/0	20/0	20/60	60	0	100%	0%
"Piso Cinco"	20/0	20/0	20/0	20/60	60	0	100%	0%
"Piso Seis"	19/1	20/0	19/1	20/60	58	2	96.66%	3.33%
"Piso Siete"	19/1	20/0	20/0	20/60	59	1	98.33%	1.77%
"Simulación"	20/0	20/0	18/2	20/60	58	2	96.66%	0%
"Compilar"	20/0	20/0	19/1	20/60	59	1	98.33%	1.77%
"Guardar"	19/1	20/0	20/0	20/60	59	1	98.33%	1.77%
"Borrar"	19/1	20/0	18/2	20/60	57	3	95%	5%
"Inicio"	20/0	20/0	20/0	20/60	60	0	100%	0%
"Sube"	19/1	20/0	18/2	20/60	57	3	95%	5%
"Baja"	20/0	20/0	20/0	20/60	60	0	100%	0%
"Para"	20/0	18/2	19/1	20/60	57	3	95%	5%

“Abrir”	19/1	17/3	19/1	20/60	55	5	91.66%	8.33%
“Cerrar”	18/2	19/1	19/1	20/60	56	4	93.33	6.66%
“Fin”	20/20	20/0	20/0	20/60	60	0	100%	0%
“Enter”	18/2	19/1	19/1	20/60	56	4	93.33	6.66%
“Uno”	20/0	20/0	20/0	20/60	60	0	100%	0%
“Dos”	20/0	20/0	20/0	20/60	60	0	100%	0%
“Tres”	19/1	20/0	20/0	20/60	59	1	98.33%	1.77%
“Cuatro”	20/0	20/0	20/0	20/60	60	0	100%	0%
“Cinco”	20/0	20/0	20/0	20/60	60	0	100%	0%
“Seis”	20/0	20/0	20/0	20/60	60	0	100%	0%
“Siete”	19/1	20/0	20/0	20/60	59	1	98.33%	1.77%
“Ayuda”	20/0	20/0	20/0	20/60	60	0	100%	0%
“Tutorial”	20/0	19/1	20/0	20/60	59	1	98.33%	1.77%

¿Por qué existen errores? R: Los errores pueden surgir por distintas causas:

- El resultado que regresa el API puede estar fuera del diccionario definido dentro de Educatrónicapp, debido a las variaciones de la palabra, por lo que se define un diccionario de palabras posibles al comando en el APP.
- La palabra puede incluir sílabas que no sean de una pronunciación sencilla para algunos usuarios.
  Ejemplo la palabra “Programar” emitida por un usuario de edad mayor a 8 años puede variar si es emitida por un usuario de edad menor.
  El usuario puede padecer “Rotacismo”, es el trastorno de la dislalia una incapacidad de pronunciar ciertos sonidos o grupo de sonidos por ejemplo la letra “r”.
- Puede interferir algún ruido del ambiente que afecte el audio.
- La respuesta del API no es en el idioma español.

PRUEBAS DE INTEGRACIÓN

Interfaz de voz bienvenida		
Descripción:	La interfaz de voz con el mensaje de bienvenida se logra escuchar en ambos sistemas operativos al iniciar el App.	
Prueba relacionada:	textToSpeak.	
Sistema Operativo:	Dispositivos IOS	Dispositivos Android
Resultado:	Exitoso	Exitoso

Navegación		
Descripción:	La aplicación móvil permite la navegación por las diferentes pantallas del App sin ningún problema.	
Prueba relacionada:	NavigationScreens, Iconos,.	
Sistema Operativo:	Dispositivos IOS	Dispositivos Android
Resultado:	Exitoso	Exitoso

Renderizado de App		
Descripción:	Todos los componentes correspondientes a cada pantalla se renderizan correctamente es decir botones, iconos, títulos, texto, componentes SVG.	
Prueba relacionada:	Renderizado de componente visuales y SVG	
Sistema Operativo:	Dispositivos IOS	Dispositivos Android
Resultado:	Exitoso	Exitoso

Respuesta de iconos		
Descripción:	Todos los iconos responden adecuadamente a sus funciones.	
Prueba relacionada:	Iconos.	
Sistema Operativo:	Dispositivos IOS	Dispositivos Android
Resultado:	Exitoso	Exitoso

Elementos Lottie para tutorial		
Descripción:	Todos los elementos Lottie qué conforman el tutorial de usabilidad se renderizan correctamente.	
Prueba relacionada:	LottieView.	
Sistema Operativo:	Dispositivos IOS	Dispositivos Android
Resultado:	Exitoso	Exitoso

Elegir piso		
Descripción:	Los usuarios pueden elegir posicionarse en un piso distinto.	
Prueba relacionada:	Picker.	
Sistema Operativo:	Dispositivos IOS	Dispositivos Android
Resultado:	Exitoso	Exitoso

Ingresar nombre del programa		
Descripción:	Los usuarios pueden colocar nombre al programa.	
Prueba relacionada:	TextInput.	
Sistema Operativo:	Dispositivos IOS	Dispositivos Android
Resultado:	Exitoso	Exitoso

Ingresar codigo de programa		
Descripción:	Los usuarios pueden colocar codigo al programa.	
Prueba relacionada:	TextInput.	
Sistema Operativo:	Dispositivos IOS	Dispositivos Android
Resultado:	Exitoso	Exitoso

Crear programa		
Descripción:	El usuario puede crear un programa correspondiente a los a los autómatas definidos.	
Prueba relacionada:	automatonComands(), automatonCoding().	
Sistema Operativo:	Dispositivos IOS	Dispositivos Android
Resultado:	Exitoso	Exitoso

Guardar programa		
Descripción:	El usuario puede guardar su programa.	
Prueba relacionada:	saveProgram().	
Sistema Operativo:	Dispositivos IOS	Dispositivos Android
Resultado:	Exitoso	Exitoso

Cargar programa		
Descripción:	El usuario selecciona un programa guardado.	
Prueba relacionada:	selectProgram().	
Sistema Operativo:	Dispositivos IOS	Dispositivos Android
Resultado:	Exitoso	Exitoso

Borrar programa		
Descripción:	El usuario puede borrar un programa seleccionado y limpia los TextInput, además reestablece el Picker en 1.	
Prueba relacionada:	selectProgram(), deletedProgram(), Picker.	
Sistema Operativo:	Dispositivos IOS	Dispositivos Android
Resultado:	Exitoso	Exitoso

Compilación de un programa		
Descripción:	La ejecución del programa funciona adecuadamente en cuestión de posición del elevador, emisión de sonido correspondiente, verificación de autómatas, reconocimiento de texto ingresado.	
Prueba relacionada:	automatonComands(), automatonCoding(), Picker, TextInput, Iconos, usedElevator().	
Sistema Operativo:	Dispositivos IOS	Dispositivos Android
Resultado:	Exitoso	Exitoso

Simulación de un programa		
Descripción:	La simulación del programa funciona adecuadamente en cuestión de posición del elevador, posición de la puerta, subir o bajar pisos.	
Prueba relacionada:	upNextLevelElevator(),downNextLevelElevator(), openDoor(), closeDoor(), usedElevator(), Picker, SVG.	
Sistema Operativo:	Dispositivos IOS	Dispositivos Android
Resultado:	Exitoso	Exitoso

Subir piso de forma manual		
Descripción:	Los usuarios pueden subir de piso en piso su elevador, de forma manual.	
Prueba relacionada:	SVG, upNextLevelElevator().	
Sistema Operativo:	Dispositivos IOS	Dispositivos Android
Resultado:	Exitoso	Exitoso

Bajar piso de forma manual		
Descripción:	Los usuarios pueden bajar de piso en piso su elevador de forma manual.	
Prueba relacionada:	SVG, downNextLevelElevator(),.	
Sistema Operativo:	Dispositivos IOS	Dispositivos Android
Resultado:	Exitoso	Exitoso

Abrir puertas de forma manual		
Descripción:	Los usuarios pueden abrir la puerta de su elevador, de forma manual.	
Prueba relacionada:	SVG, openDoor().	
Sistema Operativo:	Dispositivos IOS	Dispositivos Android
Resultado:	Exitoso	Exitoso

Cerrar puertas de forma manual		
Descripción:	Los usuarios pueden cerrar la puerta de su elevador, de forma manual.	
Prueba relacionada:	SVG, closeDoor().	
Sistema Operativo:	Dispositivos IOS	Dispositivos Android
Resultado:	Exitoso	Exitoso

Reconocimiento de voz		
Descripción:	La aplicación permite la activación del reconocimiento de voz.	
Prueba relacionada:	startRecording(),stopRecording(),Iconos, translateSpeechToText().	
Sistema Operativo:	Dispositivos IOS	Dispositivos Android
Resultado:	Exitoso	Exitoso

Resultado del API		
Descripción:	El App recibe el resultado del API Speech to Text	
Prueba relacionada:	translateSpeechToText()	
Sistema Operativo:	Dispositivos IOS	Dispositivos Android
Resultado:	Exitoso	No exitoso

El reconocimiento de voz NO es posible en Android porque el API Speech To Text recibe en un formato incorrecto el audio grabado y enviado por Educatrónicapp.
La programación realizada en la función startRecording() existe una configuración de grabación para dispositivos Android(ver pag.81), al verificar el archivo grabado por Educatrónicapp este archivo no contiene la configuración definida en la función, probablemente por cuestiones de la biblioteca usada en Expo React o con configuraciones nativas de Android.

Configuración obtenida de un archivo grabado en Android:
[Media Foundation: Unknown, ASCII 8000Hz, Channels: 1, Samples: 31200, Flags: 0]

Configuración obtenida de un archivo grabado en IOS:
[Media Foundation: Wave, ASCII 44100kHz, Channels: 1, Samples: 31200, Flags: 0]

PRUEBAS USABILIDAD

Reconocimiento de voz		
Descripción:	Los usuarios con/sin discapacidad motriz miden la usabilidad del reconocimiento de voz en el App, este proceso contempla la acción de presionar el botón para iniciar el reconocimiento de voz y detenerlo.	
Tipo de usuario:	Sin ninguna discapacidad motriz	Con discapacidad motriz
Resultado:	Efectiva	Semi efectiva Puede darse el caso que algunos usuarios dispongan una inmovilidad completa. Para usuarios que aun disponen de movilidad es factible.

Navegación		
Descripción:	Los usuarios con/sin discapacidad motriz miden la usabilidad de la navegación empleada en el App.	
Tipo de usuario:	Sin ninguna discapacidad motriz	Con discapacidad motriz
Resultado:	Efectiva	Efectiva
Descripción del resultado:	Para los usuarios es más fácil acceder a las distintas secciones del App mediante iconos y de forma rápida.	

Crear un programa		
Descripción:	Los usuarios con/sin discapacidad motriz miden la usabilidad de crear un programa empleada en el App.	
Tipo de usuario:	Sin ninguna discapacidad motriz	Con discapacidad motriz
Resultado:	Efectiva	Efectiva

Crear un programa usando lenguaje natural		
Descripción:	Los usuarios con/sin discapacidad motriz miden la forma de crear un programa empleada en el App.	
Tipo de usuario:	Sin ninguna discapacidad motriz	Con discapacidad motriz
Resultado:	Efectiva	Efectiva
Descripción del resultado:	El uso del lenguaje natural definido es de mayor facilidad para los usuarios, al estar definido por palabras de uso común.	

Tutorial de usabilidad y ayuda		
Descripción:	¿Para los usuarios es factible disponer de una sección que funda como un tutorial de usabilidad del APP y una sección de ayuda para crear su programa?	
Tipo de usuario:	Sin ninguna discapacidad motriz	Con discapacidad motriz
Resultado:	Efectiva	Efectiva
Descripción del resultado:	Para los usuarios es efectivo disponer de esas secciones por si no recuerdan como hacer un programa o como usar el reconocimiento de voz.	

Disponer de una simulación de la compilación de un programa		
Descripción:	¿Para los usuarios es factible disponer de una sección que funda como una simulación a su programa creado?	
Tipo de usuario:	Sin ninguna discapacidad motriz	Con discapacidad motriz
Resultado:	Efectiva	Efectiva
Descripción del resultado:	Educatrónica es un conjunto de herramientas con las que se interactúa, se dispone de un robot pedagógico, si no se cuenta con dicho robot la simulación ayuda a que los usuarios vean de forma interactiva el comportamiento de su programa.	

Uso de componentes SVG, Lottie, Iconos		
Descripción:	¿Los usuarios encuentran atractiva la aplicación con los elementos visuales usados?	
Tipo de usuario:	Sin ninguna discapacidad motriz	Con discapacidad motriz
Resultado:	Efectiva	Efectiva
Descripción del resultado:	Los usuarios se sienten atraídos por los elementos visuales que integran el App, les gusta la decoración, colores y es más fácil identifican acciones con los iconos.	

Disponer de una sección para la manipulación del robot		
Descripción:	¿Para los usuarios es factible disponer de una sección para manipular su robot sin necesidad de programar?	
Tipo de usuario:	Sin ninguna discapacidad motriz	Con discapacidad motriz
Resultado:	Efectiva	Efectiva
Descripción del resultado:	Si ya que el robot pedagógico recibe sonidos DTMF para comunicarse, las acciones del robot son posibles manipularse sin necesidad de programar.	

Tiempos de uso de usuarios con discapacidad motriz		
Descripción:	Se hace una medición de cuánto tiempo tardan los usuarios con discapacidad motriz en crear 1 programa. La prueba es crear 1 programa que contenga la siguiente estructura: I S 3 F	
Usabilidad de:	Con reconocimiento de voz	Sin reconocimiento de voz
Tiempo del usuario:	5.33 Segundos	10.81 Segundos

Tiempos de uso de usuarios con discapacidad motriz		
Descripción:	Se hace una medición de cuánto tiempo tardan los usuarios con discapacidad motriz en crear 1 programa. La prueba es crear 1 programa que contenga la siguiente estructura: I S 3 P 2 B 2 F	
Usabilidad de:	Con reconocimiento de voz	Sin reconocimiento de voz
Tiempo del usuario:	10.05 Segundos	40.63 Segundos

Tiempos de uso de usuarios con discapacidad motriz		
Descripción:	Se hace una medición de cuánto tiempo tardan los usuarios con discapacidad motriz en crear 1 programa. La prueba es crear 1 programa que contenga la siguiente estructura: I P 2 A 2 B 2 F	
Usabilidad de:	Con reconocimiento de voz	Sin reconocimiento de voz
Tiempo del usuario:	10.23 Segundos	45.26 Segundos

PRUEBAS DE EJECUCIÓN

Crear 1 programa que suba 2 pisos	
Descripción:	Se crea 1 programa que sube 2 pisos partiendo del piso 1
Resultado esperado	Elevador en el piso 3
Resultado obtenido	Piso 3


```
----------Compilacion en proceso----------
 LOG  Piso elegido: 1
 LOG  Piso actual: 1
 LOG  Texto ingresado:
 I
S 2
F
 LOG  Comando Inicio detectado
 LOG  Comando Subir detectado, subiendo: 2 pisos
 LOG  Subiendo a piso: 2
 LOG  Subiendo a piso: 3
 LOG  Comando Fin detectado
 LOG  Piso final: 3
 LOG  Valores restablecidos
 LOG  Bajando el elevador al piso... 1
 LOG  ----------Fin de la Compilacion----------
```

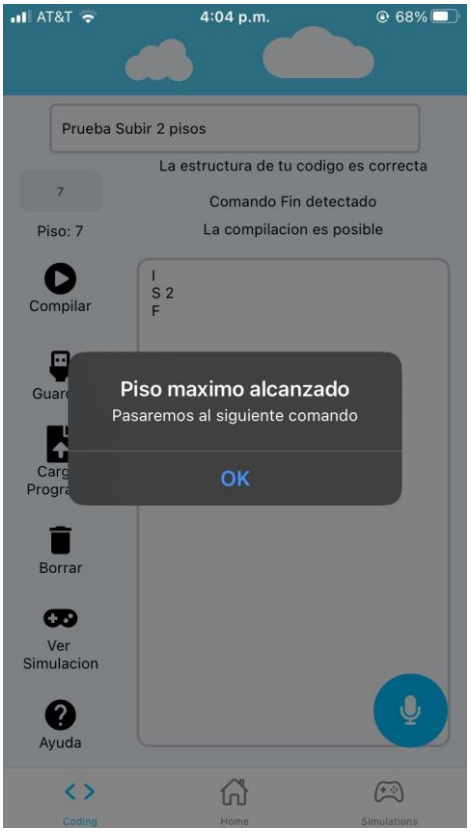
Figura 51. Captura y resultado de la prueba Subir.
Elaboración propia.

Crear 1 programa que suba 3 pisos	
Descripción:	Se crea 1 programa que sube 3 pisos partiendo del piso 3
Resultado esperado	Elevador en el piso 6
Resultado obtenido	Piso 6


```
LOG  ----------Compilacion en proceso-------
---

 LOG  Subiendo el elevador al piso... 3

 LOG  Piso elegido: 3

 LOG  Piso actual: 3

 LOG  Texto ingresado:

 I

S 3

F

 LOG  Comando Inicio detectado

 LOG  Comando Subir detectado, subiendo: 3
pisos

 LOG  Subiendo a piso: 4

 LOG  Subiendo a piso: 5

 LOG  Subiendo a piso: 6

 LOG  Comando Fin detectado

 LOG  Piso final: 6

 LOG  Valores restablecidos

 LOG  Bajando el elevador al piso... 1

 LOG  ----------Fin de la Compilacion-------
---
```

Figura 52. Captura de prueba y resultado de la prueba Subir.
Elaboración propia.

Crear 1 programa que suba 2 pisos	
Descripción:	Se crea 1 programa que sube 2 pisos partiendo del piso 6
Resultado esperado	Elevador en el piso 7
Resultado obtenido	Piso 7,ya que solo se cuenta con 7 pisos y no puede subir más.




```
LOG  ----------Compilacion en proceso---------
-

 LOG  Subiendo el elevador al piso... 6

 LOG  Piso elegido: 6

 LOG  Piso actual: 6

 LOG  Texto ingresado:

 I

S 2

F

 LOG  Comando Inicio detectado

 LOG  Comando Subir detectado, subiendo: 2
pisos

 LOG  Subiendo a piso: 7

 LOG  El elevador no puede subir más. Límite
de piso alcanzado: 7

 LOG  Comando Fin detectado

 LOG  Piso final: 7

 LOG  Valores restablecidos

 LOG  Bajando el elevador al piso... 1

 LOG  ----------Fin de la Compilacion---------
-
```

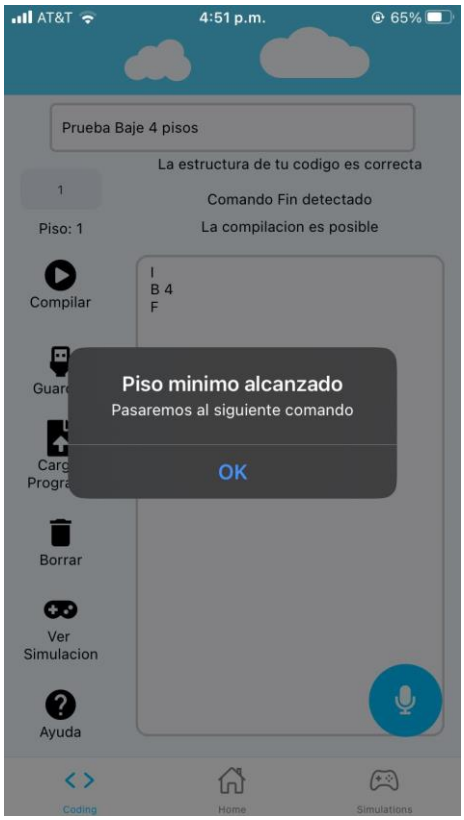
Figura 53. Captura de prueba y resultado de la prueba Subir.
Elaboración propia.

Crear 1 programa que baje 2 pisos	
Descripción:	Se crea 1 programa que baje 2 pisos partiendo del piso 5
Resultado esperado	Elevador en el piso 3
Resultado obtenido	Piso 3


```
LOG  ----------Compilacion en proceso----------

 LOG  Subiendo el elevador al piso... 5

 LOG  Piso elegido: 5

 LOG  Piso actual: 5

 LOG  Texto ingresado:

 I

B 2

F

 LOG  Comando Inicio detectado

 LOG  Comando Bajar detectado, bajando: 2 pisos

 LOG  Bajando a piso: 4

 LOG  Bajando a piso: 3

 LOG  Comando Fin detectado

 LOG  Piso final: 3

 LOG  Valores restablecidos

 LOG  Bajando el elevador al piso... 1

 LOG  ----------Fin de la Compilacion----------
```

Figura 54. Captura de prueba y resultado de la prueba Bajar.
Elaboración propia.

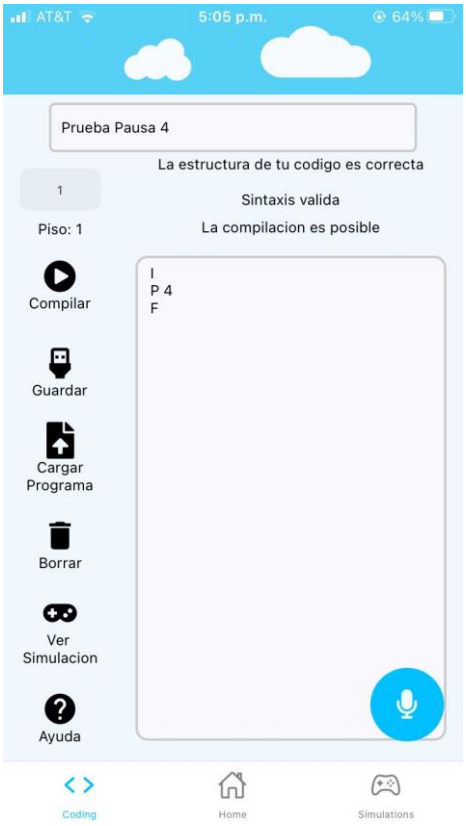
Crear 1 programa que baje 4 pisos	
Descripción:	Se crea 1 programa que baje 4 pisos partiendo del piso 3
Resultado esperado	Elevador en el piso 3
Resultado obtenido	Piso 1, ya que solo se cuenta con 7 pisos y no puede bajar más.




```
LOG  ----------Compilacion en proceso-------
---

 LOG  Subiendo el elevador al piso... 3

 LOG Piso elegido: 3

 LOG Piso actual: 3

 LOG Texto ingresado:

I

B 4

F

 LOG Comando Inicio detectado

 LOG Comando Bajar detectado, bajando: 4
pisos

 LOG Bajando a piso: 2

 LOG Bajando a piso: 1

 LOG El elevador no puede bajar más. Límite
de piso alcanzado: 1

 LOG Comando Fin detectado

 LOG Piso final: 1

LOG  Valores restablecidos

 LOG  ----------Fin de la Compilacion-------
---
```

Figura 55. Captura de prueba y resultado de la prueba Bajar.
Elaboración propia.

Crear 1 programa que pause 4 segundos	
Descripción:	Se crea 1 programa que pause 4 segundos
Resultado esperado	Pausa de 4 segundos
Resultado obtenido	Pausa de 4 segundos




```
LOG  ----------Compilación en proceso----
------

 LOG Piso elegido: 1

 LOG Piso actual: 1

 LOG Texto ingresado:

I

P 4

F

 LOG Comando Inicio detectado

 LOG Comando Parar detectado, parando: 4
segundos

 LOG Pausa del: 1 segundo

 LOG Pausa del: 2 segundo

 LOG Pausa del: 3 segundo

 LOG Pausa del: 4 segundo

 LOG Comando Fin detectado

 LOG Piso final: 1

 LOG  Valores restablecidos

 LOG  ----------Fin de la Compilacion----
------
```

Figura 56. Captura de prueba y resultado de la prueba Pausa.
Elaboración propia.

Crear 1 programa que abra puertas	
Descripción:	Se crea 1 programa que abra puertas 5 segundos
Resultado esperado	Abrir, mantener 5 segundo la puerta abierta y cerrar puertas
Resultado obtenido	Abrir, mantener 5 segundo la puerta abierta y cerrar puertas


```
LOG  ----------Compilación en proceso--------
--

 LOG Piso elegido: 1

 LOG Piso actual: 1

 LOG Texto ingresado:

I

A 5

F

 LOG Comando Inicio detectado

 LOG Comando Abrir detectado, abriendo
puertas

 LOG Deteniendo puertas

 LOG Puerta abierta por: 5 segundos

 LOG Apertura del: 1 segundo

 LOG Apertura del: 2 segundo

 LOG Apertura del: 3 segundo

 LOG Apertura del: 4 segundo

 LOG Apertura del: 5 segundo

 LOG Cerrando puertas

 LOG Deteniendo puertas

 LOG Comando Fin detectado

 LOG Piso final: 1

 LOG  Bajando el elevador al piso... 1

 LOG  Valores restablecidos

 LOG  ----------Fin de la Compilacion--------
```

Figura 57. Captura de prueba y resultado de la prueba Abrir.
Elaboración propia.

Crear 1 programa que no sea correcto	
Descripción:	Se crea 1 programa de codigo incorrecto
Resultado esperado	No permite compilar
Resultado obtenido	Mensaje de error

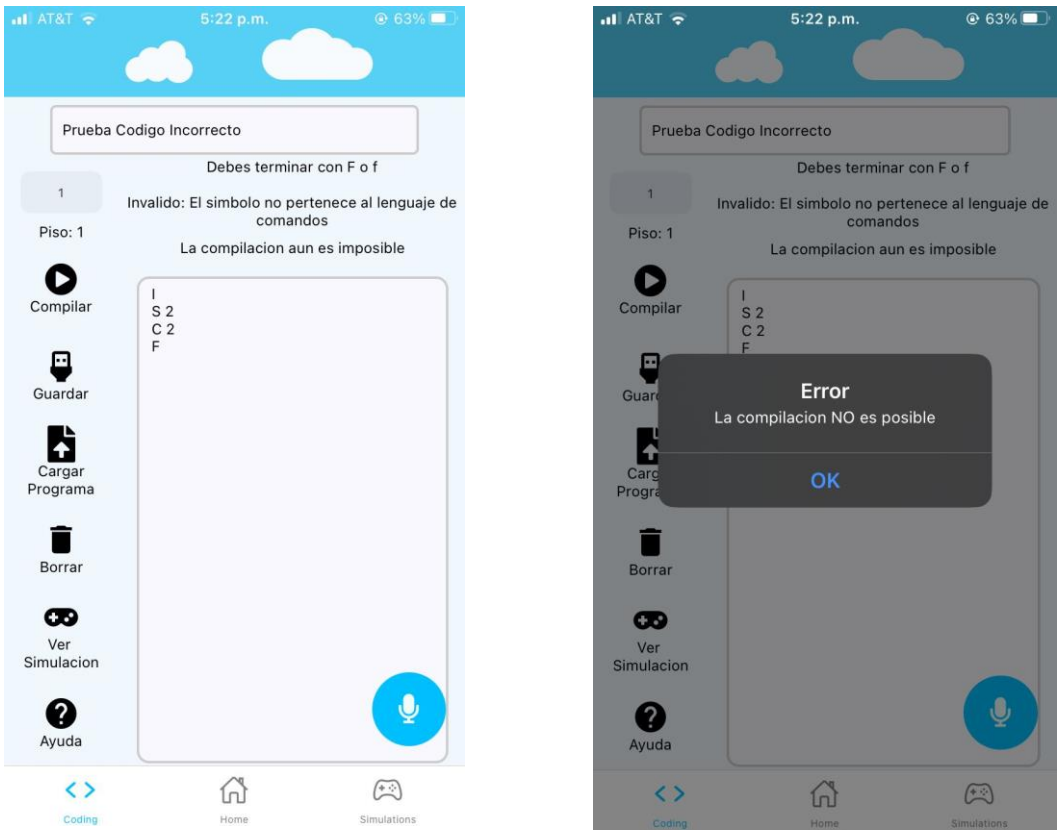


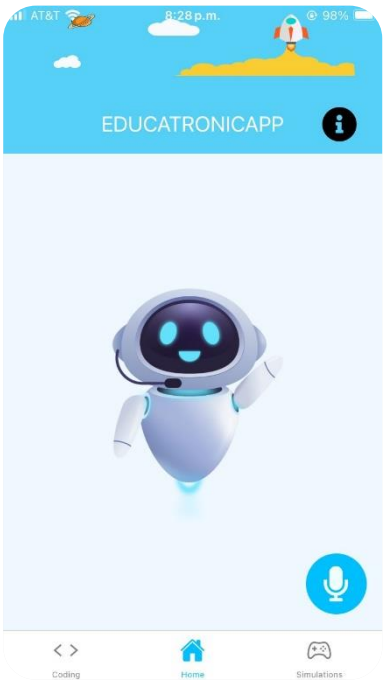
Figura 58. Captura de prueba y resultado de la prueba codigo incorrecto.
Elaboración propia.

Crear 1 programa con varias instrucciones	
Descripción:	Se crea 1 programa que suba 4 pisos, pause 3, abra 5, baje 2, pause 1, abra 2, suba 1, partiendo del piso 2
Resultado esperado	Piso final 5
Resultado obtenido	Piso 5

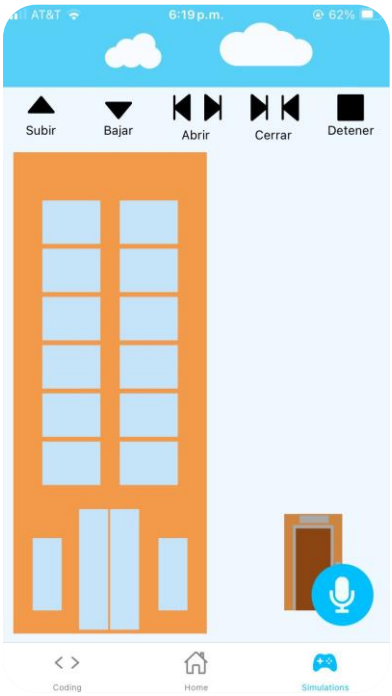

```
LOG  ----------Compilación en proceso----------
LOG  Subiendo el elevador al piso... 3
LOG Piso elegido: 2
LOG Piso actual: 2
LOG Texto ingresado:
I
S 4
P 3
A 5
B 2
P 1
A 2
S 1
F
LOG Comando Inicio detectado
LOG Comando Subir detectado, subiendo: 4 pisos
LOG Subiendo a piso: 3
LOG Subiendo a piso: 4
LOG Subiendo a piso: 5
LOG Subiendo a piso: 6
LOG Comando Parar detectado, parando: 3 segundos
LOG Pausa del: 1 segundo
LOG Pausa del: 2 segundo
LOG Pausa del: 3 segundo
LOG Comando Abrir detectado, abriendo puertas
LOG Deteniendo puertas
LOG Puerta abierta por: 5 segundos
LOG Apertura del: 1 segundo
LOG Apertura del: 2 segundo
LOG Apertura del: 3 segundo
LOG Apertura del: 4 segundo
LOG Apertura del: 5 segundo
LOG Cerrando puertas
LOG Deteniendo puertas
LOG Comando Bajar detectado, bajando: 2 pisos
LOG Bajando a piso: 5
LOG Bajando a piso: 4
LOG Comando Parar detectado, parando: 1 segundos
LOG Pausa del: 1 segundo
LOG Comando Abrir detectado, abriendo puertas
LOG Deteniendo puertas
LOG Puerta abierta por: 2 segundos
LOG Apertura del: 1 segundo
LOG Apertura del: 2 segundo
LOG Cerrando puertas
LOG Deteniendo puertas
LOG Comando Subir detectado, subiendo: 1 pisos
LOG Subiendo a piso: 5
LOG Comando Fin detectado
LOG Piso final: 5
LOG  Valores restablecidos
LOG  ----------Fin de la Compilacion----------
```

Figura 59. Captura de prueba y resultado de la prueba de varias instrucciones. Elaboración propia.

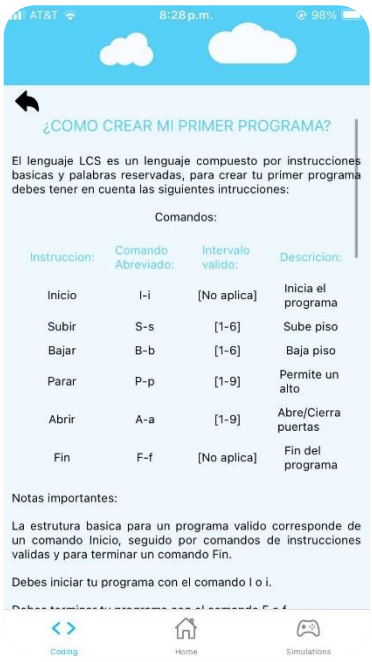
CodingScreen



HomeScreen

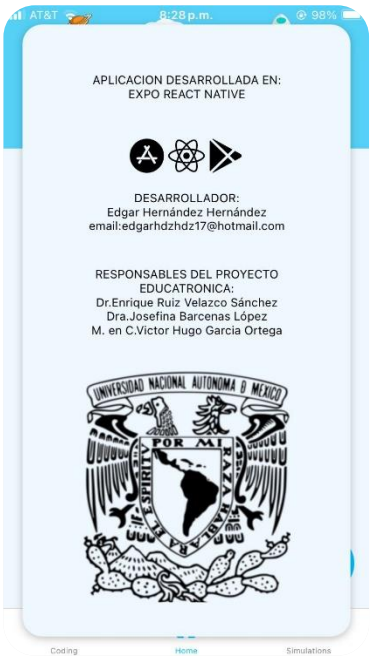


SimulationsScreen



HelpCodingScreen

HelpAppScreen



DetailsApp

Figura 60. Capturas de Educatrónicapp.
Elaboración propia.

CAPITULO 5
RESULTADOS Y CONCLUSIONES

Resultados:

A partir del desarrollo de “Educatrónicapp” se pudo observar que los usuarios comprenden mejor los conceptos básicos de la programación utilizando el lenguaje natural, debido a que les resulta más fácil utilizar palabras de uso común. Durante la fase final del proyecto “Educatrónicapp” se presentó en distintos encuentros organizados por el equipo del proyecto “Laboratorio de Educatrónica”:

En el encuentro “Mini taller de robótica pedagógica móvil” realizando en septiembre 2023.

El segundo encuentro “Robótica Pedagógica” realizado en octubre 2023.

Ambos encuentros tuvieron lugar en el Museo de las Ciencias (Universum) ubicado en la Universidad Nacional Autónoma de México (UNAM).

Además, el proyecto participo en la presentación de ponencia modalidad poster en el primer congreso interdisciplinario ICAT diciembre 2023, los resultados encontrados mostraron que los usuarios se mostraban entusiasmados debido a que encontraban una forma más dinámica de interactuar con la aplicación y su robot.

Integrar reconocimiento de voz resultó atractivo, ya que permite que usuarios con discapacidades motoras participen en este tipo de actividades donde a menudo suelen estar excluidos y que desarrollen habilidades que mejoren su educación. Ahora, los usuarios pueden acceder a herramientas como “Educatrónicapp” de forma gratuita para desempeñarse mejor en un área como la programación que cada día va creciendo.

El reconocimiento de voz obtuvo un porcentaje de efectividad de alrededor de 90%-98% de efectividad ante el proceso de convertir audio a texto, prácticamente los errores solían ser externos ya sea que el usuario podría tener problemas relacionados con el habla, por palabras no emitidas

correctamente o ruido ambiente que afectara el audio, mejorando asi la usabilidad de la aplicación por esta población. Actualmente no es posible el reconocimiento de voz en Android por cuestiones relacionadas a la compatibilidad del archivo de audio.

Respecto a la emisión de tonos DTMF emitidos por “Educatrónicapp” no se produjo error alguno debido a que la generación del tono era correcta respecto a su comando definido o la secuencia de un programa realizado.

Figura 61. Encuentro Feria de la Ciencias y Humanidades-UNIVERSUM. Elaboración propia.

Figura 62. Encuentro Feria de la Ciencias y Humanidades-UNIVERSUM. Elaboración propia.

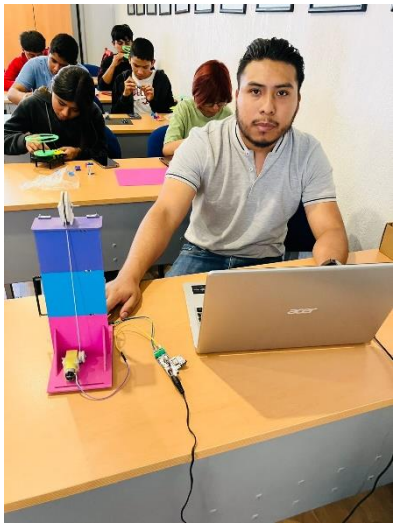


Figura 63. Presentación de Educatrónicapp Encuentro “Mini taller de robótica pedagógica móvil-UNIVERSUM. Elaboración propia.

Figura 64. Participación en el primer congreso estudiantil ICAT diciembre 2023 modalidad cartel. Elaboración propia.

Conclusiones:

Desarrollar “Educatrónicapp” fue una experiencia profundamente gratificante para mí a nivel personal. Este proyecto me brindo la oportunidad de poner en práctica los conocimientos adquiridos durante mi estancia en la universidad, al mismo tiempo que me permitió adentrarme en el mundo del desarrollo móvil.

Lo que más valore de esta experiencia fue poder crear una herramienta destinada a mejorar la educación digital con áreas como la programación, la electrónica y la robótica de una manera más dinámica. En cada fase del desarrollo comprendí la importancia de realizar un análisis detallado y me pude dar cuenta que las aplicaciones móviles educativas también incorporan aspectos pedagógicos esenciales para garantizar una interacción efectiva con los usuarios.

Un aspecto que me inspiro esencialmente fue la inclusión de funcionalidades destinadas a usuarios con discapacidad motora y que mi proyecto puede funcionar como una alternativa para que esta población pueda desempeñarse mejor en muchas áreas y tener acceso a la educación de forma gratuita, que este puede ser el camino base para mejorar la calidad de la educación, que podemos aprovechar el uso de dispositivos móviles como los teléfonos, tabletas u otro dispositivo similar para desempeñarnos en cualquier área.

En resumen “Educatrónicapp” no solo me brindo una satisfacción personal profunda, también me enseño valiosas lecciones sobre el desarrollo de aplicaciones móviles, la pedagogía, la inclusión y que sin duda este es el camino correcto para mejorar la educación eliminando las barreras físicas y sociales.

ANEXO

LEGO MINDSTORMS

Interaction of visual interface and academic levels with young students' anxiety, playfulness, and enjoyment in programming for robot control

Lego Mindstorms es un proyecto del Instituto de Tecnología de Massachussets y Lego Company que permite a los estudiantes controlar de una forma sencilla robots, haciendo uso de un lenguaje de programación basado en bloques.

La programación es un curso con alta tasa de abandono para estudiantes jóvenes sobre su introducción a la programación computacional, para estos estudiantes es más complejo y genera una baja motivación de aprendizaje comenzar con algún tipo de lenguaje como C, C# o C++ que incluyen una cantidad enorme de sintaxis compleja para principiantes, aun haciendo uso de un IDE compatible que brinde una interfaz más amigable que el uso de línea de comandos en una terminal.

Para los investigadores y educadores la programación en bloques es una alternativa para enseñar a programar a jóvenes principiantes. La programación en bloques es una programación visual que utiliza un tipo de marcador o una gramática básica para representar un significado, de forma convencional usa flechas, líneas, cubos y bloques que se comportan como un mapa mental secuencial, brindando mayor confianza y motivación a los estudiantes.

Para *Lego Mindstorms* esta estrategia fue eficiente donde los estudiantes pueden conectarse algún tipo de software de programación en bloques, por ejemplo: *Alice, Ardublock, mBlock o Scratch* para la parte de programación visual y amigable, por otro lado, podrían complementar la estrategia con el uso de juguetes Lego y realizar la conectividad entre la programación y el robot con hardware Arduino.

BB8-ROBOT

Design of an Interactive BB8-Like Robot

BB8-Robot un robot inspirado en la película de Star Wars, es un proyecto de bajo costo y robot móvil interactivo controlado de forma inalámbrica por medio de una aplicación móvil.

Basándose en el juguete BB8 decidieron integrar componentes para mejorar la interacción robot-humano, adaptando un sistema de comunicación, control inalámbrico, motores tipo CC, controlador de puente H, placa Arduino y leds.

La aplicacion móvil que complementa al robot se llama *Elegoo BLE*, esta aplicación se conecta con un módulo Bluetooth dentro de una placa Arduino que se encuentra dentro del robot, una vez conectada permite realizar movimientos para ordenarle que gire, retroceda o avance.

Otra interacción es por medio de un módulo de sonido en formato .mp3 que emite un sonido a través de un altavoz, permitiendo al robot encender una luz emitida por un led, para la detección de tal sonido es necesario un micrófono ubicado en la parte superior del robot.

SARA-STIMEY

Social Educational Robotics Application: Architecture and Interconnectivity

SARA un asistente social robótico que tiene el objetivo de convertirse en un asistente y compañero educativo para aumentar la motivación de los estudiantes y recompensar el proceso de aprendizaje hacia estudios STEM.

La principal arquitectura se compone de una base de conocimientos, un sistema de programación de robots, una interfaz de usuario y una plataforma de comunicación. La base de conocimientos se utiliza para almacenar la información necesaria para la programación del robot, como su comportamiento, habilidades y conocimientos. El sistema de programación de robots proporciona una interfaz gráfica para programar el comportamiento del robot, y la interfaz de usuario proporciona una forma fácil de interactuar con el robot. La plataforma de comunicación se utiliza para conectar el robot con otros dispositivos y sistemas, como aplicaciones móviles y sistemas de gestión de aprendizaje.

El robot cuenta con un esqueleto humanoide con brazos, piernas y cabeza además de integrar hardware para integrar el uso de comandos de voz incorporando un API de Reconocimiento de Voz y conexión serie TCP asíncrona bidireccional para conectarse con el microcontrolador del robot, haciendo que los usuarios puedan tener una interacción con un lenguaje casi natural y el uso de la voz. La plataforma *STYMEY* implementan un servicio API REST que permite la comunicación cliente-servidor para almacenar información de autenticación, datos de usuario, configuraciones e interacciones pasadas.

El software permite la comunicación por medio de comandos como “Levanta tu mano izquierda” o “Dime información sobre tal tema”, si el robot recibe un comando valido se emplea un procesamiento para el análisis y generar una respuesta. También cuenta con un campo para entrada de texto donde pueden indicar comando por medio de un tipo de programación de forma estructurada tipo lenguaje HTML, ejemplo *[RobotFace type=BlinkLeft] [/RobotFace*], el robot procederá a realizar una acción facial parpadeando el ojo izquierdo.

ROOT

iRobot: iRobot's New Education Robot Makes Learning to Code a Little More Affordable

Root es un robot educativo perteneciente a la compañía iRobot, diseñado en el año 2016 por el Instituto Wyss de Harvard, hecho como una herramienta practica para que los niños de todas las edades desarrollen habilidades de lógica y codificación en áreas STEM (*Science, Technology, Engineering and Mathematics*) proporcionando una alternativa para la introducción a la programación y a la robótica.

Root es un robot magnético compuesto por hardware, puede usarse en superficies lisas como el piso o una mesa, contiene componentes eléctricos que permiten interpretar cualquier movimiento incluyendo su velocidad y rotación. Sus elementos interactivos son diseñados para ser usado por niños de hasta 4 años, haciendo que puedan programar Root con el uso de una interfaz en su aplicación móvil, su entorno de programación Square proporciona distintos niveles de programación, desde una programación grafica basada en bloques para niños más pequeños hasta una programación basada en texto para niños mayores. Con esta herramienta los niños pueden sumergirse directamente en la programación y la resolución de problemas computacionales sin tener que lidiar con la complejidad de un lenguaje de programación, además permite una codificación más compleja para quienes quieren aprender de una forma avanzada permitiendo el uso de lenguaje JavaScript, Python o Swift, dependiendo del nivel de programación que sus usuarios elijan.

Su software llamado *Root Coding* compatible con plataformas IOS, Android, Chrome OS, Windows proporciona la opción de simular cualquier programa que se realice de forma accesible para quienes no puedan comprar el robot con un costo de 200 dólares. La conectividad y control es realizada mediante Bluetooth, Conector USB-C o con placas como Raspberry Pi, Arduino o BBC Micro: Bit. Actualmente *Root* y *Root Coding* se usa en aulas escolares e intervenciones en el mercado comercial.

DIAGRAMA DE GANTT

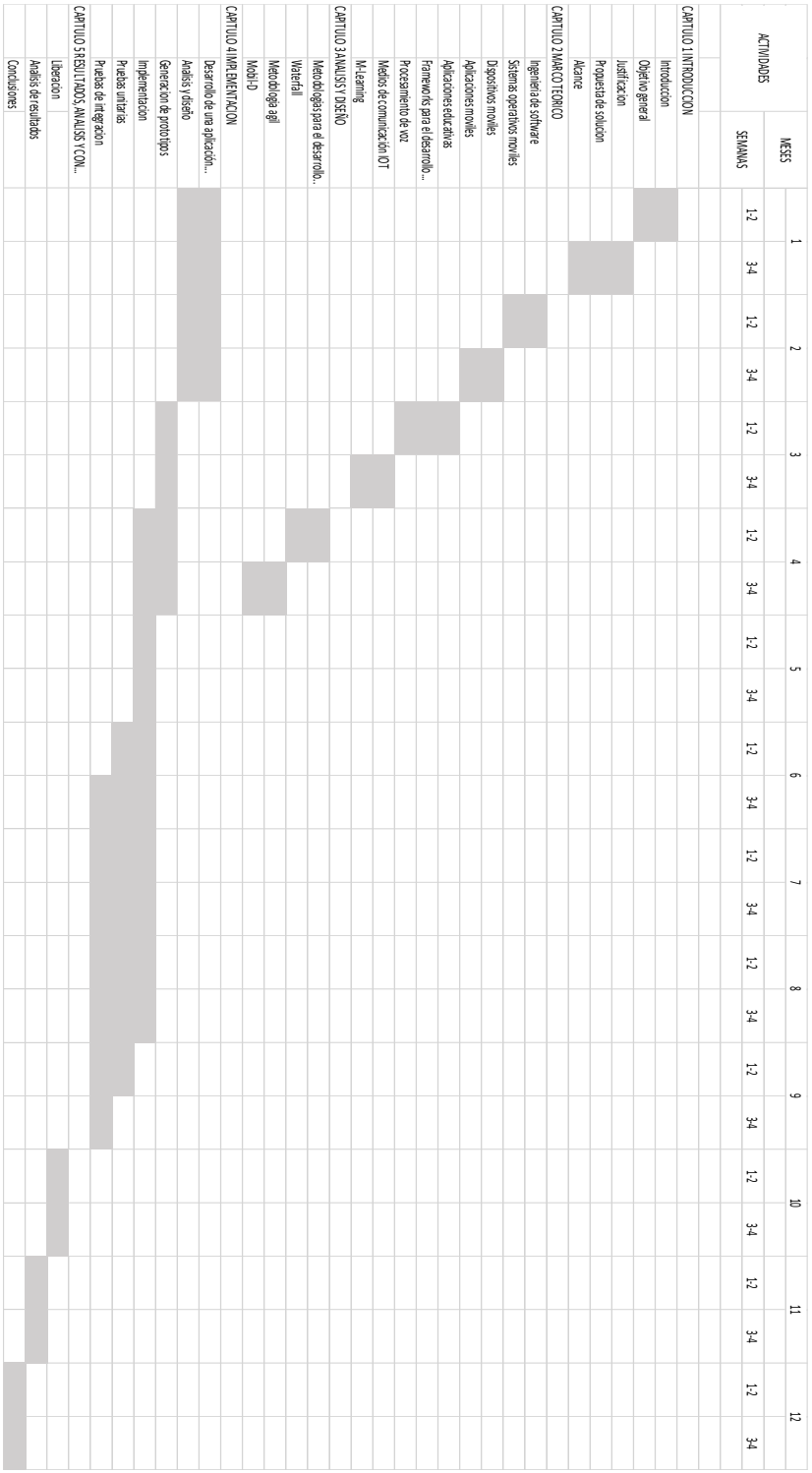


Figura 65. Diagrama de Gantt.
Elaboración propia.

GLOSARIO

API: Interfaz de Programación de Aplicaciones.

CSS: Hoja de Estilos en Cascada.

DTMF: Frecuencia Múltiple de Doble Tono.

Framework: Conjunto de herramientas que proporciona una estructura para el desarrollo de software.

HTML: Lenguaje de Marcado de Hipertexto.

IOT: Internet de las cosas.

UML: Lenguaje de Modelo Unificado.

REST: Transferencia de Estado Representacional.

SARH: Sistema Automático de Reconocimiento del Habla.

SDK: Kit de Desarrollo de Software.

Smartphone: Teléfono móvil inteligente que ofrece funcionalidades avanzadas.

UI: Interfaz de Usuario.

UX: Experiencia de Usuario.

REFERENCIAS

Aparicio, A. (2012). Ingeniería de Software.

Aprendizaje digital y transformación de la educación. (s. f.). UNESCO. https://www.unesco.org/es/digital-education

Camargo Serrano, J. (2010). Sistema de reconocimiento de voz humana por hardware. Universidad Pontificia Bolivariana, Bucaramanga.

Cortez Vásquez, A., Vega Huerta, H., & Pariona Quispe, J. (2009). Procesamiento de lenguaje natural. Revista de ingeniería de sistemas e informática, 6(2).

Da Silva Gillig, J., Pokress, S., & Cherney, R. (2023, 23 marzo). How Root wants to bring coding to every classroom. IEEE Spectrum. https://spectrum.ieee.org/how-root-wants-to-bring-coding-to-every-classroom

Digital Talent Agency. (s. f.). Metodologías de gestión de proyectos: Tema 1 - Metodología Waterfall o en Cascada.

Educación inclusiva. (s. f.). UNICEF. https://www.unicef.org/lac/educación -inclusiva

Expo documentation. (s. f.). https://docs.expo.dev/

García-Bullé, S. (2022). ¿Qué es el m-learning? ¿Es una opción viable para la educación del siglo XXI? Observatorio / Instituto para el Futuro de la Educación. https://observatorio.tec.mx/edu-news/que-es-mobile-learning/

Gobierno de Navarra. (s. f.). Uso de dispositivos móviles (teléfonos móviles, «smartphones», «e-books», GPS y «tablets»). acércate a las TIC.

IoT applications retailers are using today. (s. f.). SAS. https://www.sas.com/es_mx/insights/articles/big-data/five-iot-applications-retailers-are-using-today.html

INNE-Informe, 2014: EL DERECHO A LA EDUCACIÓN. Instituto Nacional Para Evaluacion de La Educacion.

NACIONES UNIDAS. (2020). La educación durante la COVID-19 y después de ella.

Nielsen Jacob, Usability engineering. AP Professional, Boston, MA, 1993.

OpenAI Platform. (s. f.). https://platform.openai.com/docs/guides/speech-to-text

Qué es una API - Explicación de interfaz de programación de aplicaciones - AWS. (s. f.). Amazon Web Services, Inc. https://aws.amazon.com/es/what-is/api/

Riande Juárez, N. A. & Derechos Humanos. Revista Praxis N.27: LA EDUCACIÓN DIGITAL EN MÉXICO Y EN EL MUNDO. Tribunal Federeal de Justicia Administrativa, pag.3

Richard E. Mayer, Multimedia Learning. Third Edition, Cambridge University Press. 2020

Sánchez, E. R. (2007). Educatrónica: Innovación en el aprendizaje de las ciencias y la tecnología. Educatrónica. https://dialnet.unirioja.es/servlet/libro?codigo=299746

UNESCO. (2013). Directrices de la UNESCO para las políticas de aprendizaje móvil. Directrices para las políticas de aprendizaje móvil, pág. 30-39.

UNESCO. (2021). La UNESCO hace un llamado para atender las necesidades educativas de las personas con discapacidad. UNESCO. https://www.unesco.org/es/articles/la-unesco-hace-un-llamado-para-atender-las-necesidades-educativas-de-las-personas-con-discapacidad

Software Engineering Concepts. New York: McGraw-Hill, 1985.

React native · Learn once, write anywhere. (s. f.). https://reactnative.dev/