CREACIÓN DE UNA HERRAMIENTA DE VOZ A TEXTO UTILIZANDO UN MOTOR DE SOFTWARE LIBRE PARA FACILITAR LA INCLUSIÓN DIGITAL EN EDUCACIÓN TELEPRESENCIAL A LA COMUNIDAD NO OYENTE DE LA UNIVERSIDAD ECCI.

LUIS ALBERTO DUARTE CORTES

UNIVERSIDAD ECCI

Dirección Ingeniería de Sistemas

Proyecto de Grado

Bogotá D.C.

2022.

CREACIÓN DE UNA HERRAMIENTA DE VOZ A TEXTO UTILIZANDO UN MOTOR DE SOFTWARE LIBRE PARA FACILITAR LA INCLUSIÓN DIGITAL EN EDUCACIÓN TELEPRESENCIAL A LA COMUNIDAD NO OYENTE DE LA UNIVERSIDAD ECCI.

Presentado por:

LUIS ALBERTO DUARTE CORTES

Presentado a:

Ing. ALEXANDER SABOGAL RUEDA

Director de Trabajo de Grado

MsC. Ing. MARLA COSTANZA BARRERA BOTERO

Asesor Metodológico

UNIVERSIDAD ECCI

Dirección Ingeniería de Sistemas

Proyecto de Grado

Bogotá D.C.

2022.

Dedicatoria

Este trabajo de grado, lo dedico a la memoria de mi abuelo Juan Francisco Cortés Rigueros (QEPD). Gracias a él, pude entrar en esta Universidad para cumplir mis metas profesionales.

A mi madre, abuela y hermano, por haberme forjado como la persona que soy en la actualidad, estos logros se los debo a ustedes, entre los que se incluye el presente trabajo.

A mi tío Rafael Cortés, por su apoyo incondicional en los buenos y malos momentos.

Finalmente, quiero dedicar este proyecto de grado a toda mi familia, porque con sus recomendaciones, consejos, motivación y aliento, lograron convertirme en una mejor persona, acompañándome siempre en todos mis sueños y metas.

Agradecimientos

Quisiera expresar mi gratitud a Dios, por todas las bendiciones diarias que llegan a mi vida.

Gracias a mi familia, por formarme con reglas, libertades y por motivarme constantemente para alcanzar mis anhelos.

De igual manera, mis agradecimientos a la Universidad ECCI, a toda la facultad de ingeniería de sistemas, a mis profesores, en especial al profesor Luis Efraín Ruiz Suarez y la profesora Ana Rocío León Lugo, quienes con sus valiosos conocimientos y gracias a su enseñanza, apoyaron mi crecimiento profesional, brindándome los espacios y la motivación para desafiar mis propios límites constantemente.

A los grupos de investigación SIGESTECCI y SIRSEG, donde desarrollé una vocación profesional de investigación, gracias por confiar en mí, por asistir a mis charlas, por abrirme las puertas y permitirme realizar todo el proceso investigativo dentro de su semillero.

Finalmente, quiero expresar mi sincero agradecimiento al Ing. Alexander Sabogal Rueda y a la Ing. Marla Constanza Barrera Botero, por su dirección, conocimiento, enseñanza y colaboración, gracias a los cuales fue posible el desarrollo de este trabajo.

.

Abstract

The purpose of this project is to develop a free software prototype tool for assisting hearing-impaired students that struggle with virtual academic environments.

This comes from direct experience of the needs of this community using this platform during the Covid-19 restrictions where captions were unavailable during classes so hearing-impaired students required a full-time sign interpreter during classes to communicate accurately with the teacher.

The main motivation for this development is to achieve easier adoption of new virtuality use cases, and solve the evident absence of this service since it is not provided by the university. This proposal is born from the Free Software Research hotbed from the ECCI University to dig deeply into voice-to-text technologies as a primary source to conduct a state-of-the-art analysis using different voice-to-text engines in search of the one most suitable for developing a functional prototype of a voice-to-text based web application that allows students and the university free access to a subtitle generation service that can be provided during conferences, speeches, and extracurricular activities.

In this document, you will be able to visualize the different proposed phases: analysis, design, implementation, execution, and testing made to the web application in the last part of the document. The conclusions and contributions on further implementation will be reviewed, as well as recommendations for upgrading and scaling up this platform to be widely used by the community.

Resumen

En este documento se muestra el desarrollo de un prototipo de Software Libre para asistir a los estudiantes de la comunidad de discapacidad auditiva en ambientes virtuales, esto a partir de la observación directa de las necesidades especiales de esta población en estas plataformas durante las restricciones de COVID-19 donde los subtítulos de apoyo no estaban disponibles para ellos ya que dependen de un intérprete de lengua de señas para comunicarse adecuadamente con un docente. La motivación principal para desarrollarlo es lograr que los estudiantes sordos se adapten fácilmente a los nuevos usos de la virtualidad, y solucionar la evidente ausencia de este servicio por parte de la Universidad, Para esto se realizó desde el semillero de Software Libre de la Universidad ECCI una investigación a fondo sobre estas tecnologías de reconocimiento de voz como una investigación primaria a partir de la cual se elaboró un estado del arte analizando los diferentes motores de voz en busca del motor más adecuado para desarrollar un prototipo funcional de una herramienta de voz a texto basada en una aplicación web que le permita a los estudiantes y a la Universidad acceder a este servicio como apoyo visual de subtítulos en conferencias charlas y actividades académicas extracurriculares.

En este documento se visualizan las diferentes fases planteadas: análisis, diseño, implementación, ejecución y pruebas realizadas a la aplicación web y en la parte final del documento, se evidencian las conclusiones y aportes sobre la investigación, así como recomendaciones para actualizar y escalar esta plataforma, para que sea ampliamente utilizada por la comunidad.

Tabla de Contenido

Lista de Tablas

Lista de Figuras

1. Título de la Investigación

Diseño y creación de una herramienta de voz a texto, utilizando un motor de Software Libre, para facilitar la inclusión digital en educación telepresencia, a la comunidad de discapacidad auditiva en la Universidad ECCI.

El proyecto investigativo, surgió en la coyuntura de pandemia de COVID-19, donde se identificó la necesidad de una herramienta universal, que permitiera a los estudiantes de la Universidad ECCI de discapacidad auditiva, acceder a subtítulos de apoyo en plataformas donde estos no están disponibles, para poderlos editar, guardar etc. Esta herramienta se basa en un motor de Software Libre, que recibe, analiza e interpreta una fuente de audio y transcribe su contenido a texto en español, esto no busca reemplazar a un traductor de lengua de señas, sino apoyar al estudiante de discapacidad auditiva con un registro escrito de la conversación generada, para aliviar la carga de múltiples tareas y atención que requiere el estudiante, permitiendo que su inclusión digital sea más amena, frente a los escenarios académicos virtuales que puedan no tener interprete. Ya sea por el limitado número de alumnos con discapacidad auditiva o por no tener suficientes intérpretes, para abordar todas las materias.

2. Problema de la Investigación

2.1 Descripción del Problema

La Universidad ECCI, cuenta con programas educativos enfocados hacia la comunidad de discapacidad auditiva, esta enfoca sus esfuerzos para realizar periódicamente, capacitaciones pedagógicas para los docentes y estudiantes, junto con tutorías especializadas, enfocadas a apoyar de manera integral a estos miembros de la comunidad.

A raíz de la pandemia de COVID-19, que cambió nuestra realidad social y académica, la Universidad se ajustó a los lineamientos del estado Colombiano sobre el tema y se implementó un modelo de teleclase en casa, teniendo que migrar de modelos basados en la educación presencial, a modelos apoyado en las "TIC" implementando un modelo virtual y semipresencial en las materias prácticas, cumpliendo así con las exigencias del Ministerio de Educación Nacional (Ministerio de Educación, 2020). En este modelo, las clases son dictadas mediante la adopción de diferentes modelos TIC's, cuyo entorno no estaba preparado, para adaptarse a las necesidades de la comunidad con discapacidad auditiva. En la Tabla N.1, observamos la comparación de los aspectos en soporte auditivo, de subtítulos de asistencia y apoyo a la comunidad de discapacidad auditiva, en diferentes plataformas virtuales, audiovisuales y académicas.

En una investigación de observación preliminar realizada para este proyecto se documentaron las diferentes plataformas virtuales, utilizadas en contextos académicos tele-presenciales, en diferentes clases dictadas en el segundo semestre del 2019, en busca de opciones de accesibilidad, que incluyeran ayudas externas de subtítulos de apoyo sus

características principales como Lenguaje, interacción de usuario y limitaciones, las cuales se pueden observar en la Tabla 1.

Tabla 1 Comparación de subtítulos de apoyo en diferentes plataformas audiovisuales, plataformas académicas y plataformas de contenido audiovisual web

Plataforma audiovisual	***Características de accesibilidad y subtítulos de apoyo***
YouTube	YouTube utiliza el API propietario de Google para traducción de subtítulos simultánea, es el más avanzado y con menor tasa de errores del mercado.
Daily motion	No cuenta con subtítulos de apoyo.
Vimeo	No cuenta con subtítulos de apoyo.
Wikipedia	Los subtítulos de apoyo son archivos adicionados de forma previa al video, no se garantiza la disponibilidad de todos los lenguajes.
Plataforma Academica	***Características de accesibilidad y subtítulos de apoyo***
Google meet	El 17 de marzo 2021 “Google meet”, adicionó el idioma español desde su API a su plataforma, más es imposible copiar, pegar y guardar, una transcripción del mismo.
Zoom	Incluye transcripción de subtítulos en directo, subtítulos de terceros o subtítulos manuales, permite guardar la transcripción.

Microsoft Teams	Incluye “Closed Caption” en múltiples idiomas, utilizando el API. Propietario de Azure, no permite guardar la transcripción generada.

Plataformas de Streaming web	***Características de accesibilidad y subtítulos de apoyo***
Twitch	No cuenta con subtítulos de apoyo.
Vimeo Livestream	No cuenta con subtítulos de apoyo.
YouTube Livestream	Cuenta con disponibilidad de subtítulos limitada según el idioma. Español Latinoamérica no soportado en vivo.

Fuente: Elaboración propia (Feb 2022)

Como se puede observar, solo 2 plataformas audiovisuales incluyen subtítulos de apoyo. Cabe aclarar que, al inicio del proyecto, algunas plataformas académicas aquí descritas, no tenían subtítulos de apoyo en español, este fue agregado posteriormente en actualizaciones y aunque todas las plataformas virtuales académicas mencionadas en el presente documento, tienen la función de subtítulos directos, solo una permite guardar la transcripción generada al autor de la conferencia. El guardar esta transcripción, es fundamental para apoyar el aprendizaje en ambientes virtuales de aprendizaje, enfocados a la comunidad con discapacidad auditiva, ya que el registro escrito les facilita a los miembros de esta, repasar una clase o almacenar este registro en sus apuntes personales, sin tener que transcribir manualmente dicho texto. Por lo tanto, es pertinente realizar una

investigación, sobre formas de implementar herramientas, para facilitar la adaptación de esta comunidad a estos ambientes tele-presenciales.

Como en las conferencias, paneles, encuentros y seminarios académicos hoy en día, se divulgan y se trasmiten en directo mediante plataformas web de streaming, también mencionadas en la Tabla 1, que no cuentan con un apoyo de subtítulos durante trasmisiones en vivo.

Aunque este problema puede dimensionarse más allá de los ambientes virtuales, la demanda de subtítulos de apoyo en ambientes presenciales es creciente y podría adaptarse a actividades académicas extracurriculares, encuentros estudiantiles, Grupos de estudio autónomo y Semilleros de investigación, donde es más difícil garantizar la presencia de un traductor de señas, pues estos normalmente cuentan con un horario curricular fijo de horas semanales. Por lo tanto, es pertinente realizar una investigación, sobre formas de implementar herramientas, para facilitar la adaptación de esta comunidad a estos ambientes virtuales y presenciales.

2.2 Formulación del Problema

A partir de la observación directa de las dificultades presentadas durante el confinamiento por la COVID-19.Nuestros sistemas educativos, se vieron seriamente afectados, por la suspensión de las actividades presenciales en las Universidades colombianas y la imposición de un nuevo modelo de enseñanza, para el cual la Universidad debía asegurar la disponibilidad de un interprete de lengua de señas, además de los recursos tecnológicos para permitir el acceso a la información y formación académica de esta población en igualdad de condiciones. En la Universidad ECCI se evidencia una ausencia del servicio de subtitulados automáticos como asistencia y apoyo visual para estudiantes sordos. Ante la posibilidad de que busque una implementación de una alternativa privada y comercial nace la propuesta del semillero de Software Libre SIGESTECCI, de investigar y desarrollar prototipos de herramientas, para facilitar su adaptación a estos ambientes tele presenciales, por este motivo se realiza la siguiente formulación del problema:

¿Cómo garantizar que la comunidad estudiantil con discapacidad auditiva de la Universidad ECCI tenga acceso a subtítulos de apoyo en español desde cualquier plataforma web?

3. Objetivos de la Investigación

3.1 Objetivo General

Construir un prototipo funcional, de una herramienta, que asista a la comunidad de discapacidad auditiva, con subtítulos de apoyo en idioma español y que sea funcional en cualquier plataforma web.

3.2 Objetivos Específicos

- Identificar diferentes herramientas de reconocimiento de voz de Software Libre, para ser implementadas en el prototipo.
- Diseñar, implementar y validar el prototipo preliminar en un ambiente de pruebas, utilizando un motor de Software Libre, incluyendo el modelo de despliegue con el prototipo final.
- Desarrollar una interfaz gráfica, que enlace y muestre la información procesada en el prototipo preliminar, probando la latencia, velocidad de procesamiento y fiabilidad de traducción, de este prototipo.
- Documentar todo el proceso de instalación, configuración del prototipo y su infraestructura, analizando el resultado obtenido y proponer mejoras, para una eventual implementación en la Universidad.

4. Justificación y Delimitaciones de la Investigación

4.1 Justificación

Se realizará la creación de este prototipo, para que la Universidad ECCI, pueda implementar una herramienta de voz a texto, tomando como referencia este proyecto, para la instalación y la configuración de la infraestructura necesaria, para la generación de subtítulos automáticos. Todos los pasos, quedarán detalladamente documentados y podrán ser utilizados en un futuro proyecto, para desplegar esta plataforma de una forma más rápida y efectiva. Así la Universidad ECCI, podrá proveer el servicio a la comunidad de discapacidad auditiva, para acceder a servicios de subtitulación automática desde cualquier ambiente web, sin necesidad de contratar o depender de herramientas y servicios de software de terceros.

4.2 Delimitaciones

- El proyecto de investigación será conformado por un solo estudiante, con el apoyo de dos asesores: uno para la dirección del trabajo de grado y otro para la metodología, por parte de la Universidad ECCI.
- El proyecto se realizará, iniciando desde el 11 de mayo de 2021 y terminando a finales de octubre de 2022.
- El prototipo, permitirá exportar las transcripciones realizadas por el motor, pero no almacena ningún tipo de información, durante ni después de su funcionamiento.

- El Hosting, los ambientes de prueba y producción, se realizarán en un servidor de propiedad del autor, no institucional y el prototipo funcional, se apagará 30 días después de entregado el trabajo de grado.
- Las instrucciones y documentación de instalación, requieren dependencias definidas en versiones específicas, para garantizar el funcionamiento de esta herramienta.
- Solo se utilizará Software Libre, para el desarrollo y la implementación del prototipo de esta herramienta.

5. Marco de referencia

5.1 Marco Teórico

5.1.1 Reconocimiento de voz a texto.

Es una tecnología multidisciplinaria, que combina ciencias de la computación, ingeniería y lingüísticas computacionales, en un software que permite el reconocimiento y la traducción o transcripción del lenguaje hablado, a cadenas digitales de texto escrito. También es conocido como reconocimiento computacional de voz o por sus Siglas "ASR" (Automatic Speech Recognition) o reconocimiento automático de habla.

El reconocimiento automático de voz a texto, se puede utilizar en una amplia variedad de productos, ya que por sus propias características, es una interface de entrada con lenguaje natural para el ser humano, muy utilizado en asistentes inteligentes, como el Google Home y Alexa, los cuales son parlantes inteligentes, adaptados a asistir a las

personas en sus tareas cotidianas, ya sea brindando información o manipulando dispositivos IoT, Automatización, Aplicaciones de ámbito militar, ámbito judicial Call Centers y sistemas de IVR (Interactive Voice Response) Robótica y Videojuegos. Pues en ambientes digitales esta herramienta es imprescindible para mejorar la comunicación y productividad de las empresas, ofreciendo traducciones en tiempo real sin necesidad de intérpretes presenciales o subtitulación automática en contenido Audiovisual. En resumen el reconocimiento de voz a texto es un esfuerzo tecnológico de conexión para combatir la brecha digital de habla, estas tecnologías le otorgan una dimensión humana a nuestros dispositivos, pero para lograr esto los desarrolladores necesitan muestras enormes de datos de voz y la mayoría de estos modelos son pagos y contenidos en licencias propietarias, por esa razón en este estudio nos enfocamos en las herramientas de código abierto donde estos modelos de lenguaje, bibliotecas y motores de voz están disponibles de forma pública y gratuitas.

En la actualidad existen diferentes formas para trascribir el lenguaje hablado cada uno con infraestructura métodos y características diferentes además de librerías interfaces y modelos de reconocimiento de voz los cuales son necesarios de conocer para diseñar e implementar un prototipo de herramienta con esta funcionalidad. A continuación, se detallarán las diferentes partes y elementos que conforman los componentes esenciales de un software de reconocimiento automático de voz a texto.

5.1.1.1 Partes y elementos del reconocimiento de voz a texto

Para realizar una transcripción de voz a texto se pueden utilizar una gran variedad de métodos, pero todos coinciden con un diseño similar, según lo describen (Sadeen , y otros, 2021) Para entender cómo opera el reconocimiento de voz, hay que describir cómo se originan los sonidos del habla. Todo comienza cuando una persona hace vibrar sus cuerdas vocales estas causan una onda haciendo que el aire vibre y genere ondas de sonido, estas ondas viajan a través del aire donde son recolectadas por el oído y trasmitidas al cerebro para ser interpretadas como sonido. Las palabras consisten de sonidos del habla, también conocidos como fonemas, los fonemas tienen características que le permiten a los humanos identificarlos según el ruido que generan, juntando estos "segmentos sonoros" ya sea con un micrófono o con un archivo de audio en el sistema, se ingresa esta información al "Core" siendo este un "núcleo" o motor, donde esta información es analizada, comparada y clasificada utilizando diferentes métodos, filtros y algoritmos, para limpiar, ordenar e identificar las palabras, aplicando un procesamiento natural de lenguaje sobre esa fuente de audio, para culminar mostrando los resultados de dicha transcripción en una interfaz gráfica como texto digital en cadena, culminando así su función. En la figura 1 se puede observar un diagrama que describe detalladamente este proceso.

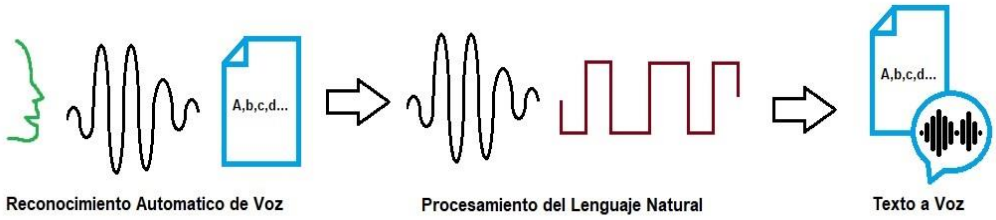


Figura 1 Proceso de Reconocimiento de Habla Mediante ASR

Fuente: elaboración propia utilizando iconos de Software Libre de flaticon.

Cabe aclarar que distintos métodos y formas de lograr este procesamiento, han ido evolucionando vertiginosamente en los últimos años, lo cual ha llevado a los investigadores a separar los términos en los cuales se define esta tecnología, según los actores que intervienen en el proceso y sus resultados finales, acuñando 3 términos para categorías diferentes los cuales son:

ASR (Automatic Speech Recognition) Reconocimiento automático de habla,

MT (Machine Translation) Traducción de máquinas y

SLT (Spoken Language Translation) Traducción de lenguaje hablado. Para entender mejor como estas tecnologías se derivan de la transcripción de voz a texto, se puede observar en la Figura 2, que cada categoría tiene un inicio o un final diferente durante el proceso de transcripción, siendo la finalidad la determinante a la hora de aclarar con que categoría se está trabajando.

Figura 2 ASR MT y SLT

Fuente: elaboración Propia utilizando iconos de Software Libre de flaticon.

Como podemos observar, el reconocimiento automático de voz ASR, involucra desde la generación del sonido en lenguaje natural, hasta la transcripción del mismo. La traducción de máquina MT no involucra la generación del sonido, pues su origen son archivos de audio ya modulados y culmina en la traducción a un lenguaje diferente al hablado y en caso de incluir la generación del mismo, se conocería como traducción del lenguaje Hablado SLT.

En esta investigación se realiza un análisis funcional de los ASR, que partes los componen y que interacciones realizan estas partes entre sí, para lograr el resultado final de la transcripción de texto.

5.1.1.1.1 Limpieza de las muestras de voz

La parte esencial de todos los sistemas de reconocimiento de voz basados en procesamiento de lenguaje natural es la introducción de información proveniente de una fuente de audio, sea un micrófono o un archivo de audio crean un formato donde almacena una onda de sonido está onda de sonido es limpiada de ruidos externos removiendo los sonidos de fondo para brindar una mayor claridad al sonido de la voz, Existen cinco factores que están involucrados en este proceso de controlar y simplificar el reconocimiento de voz, estos son:

- separación de pausas entre palabras
- entonación
- velocidad
- volumen
- ruido del entorno

Estos factores inciden directamente en la correcta identificación de información por parte de los sistemas de reconocimiento de voz, por este motivo toda la información entrante debe ser estar estandarizada, es decir se deben equilibrar todos los niveles de volumen, deben tener un tiempo de duración determinado y compartir un formato de almacenamiento de audio específico.

El formato de almacenamiento determinará la forma de la codificación del audio en formato digital, este puede ser comprimido o descomprimido para cambiar su tamaño en disco y contener metadatos almacenados en una capa extra de almacenamiento.

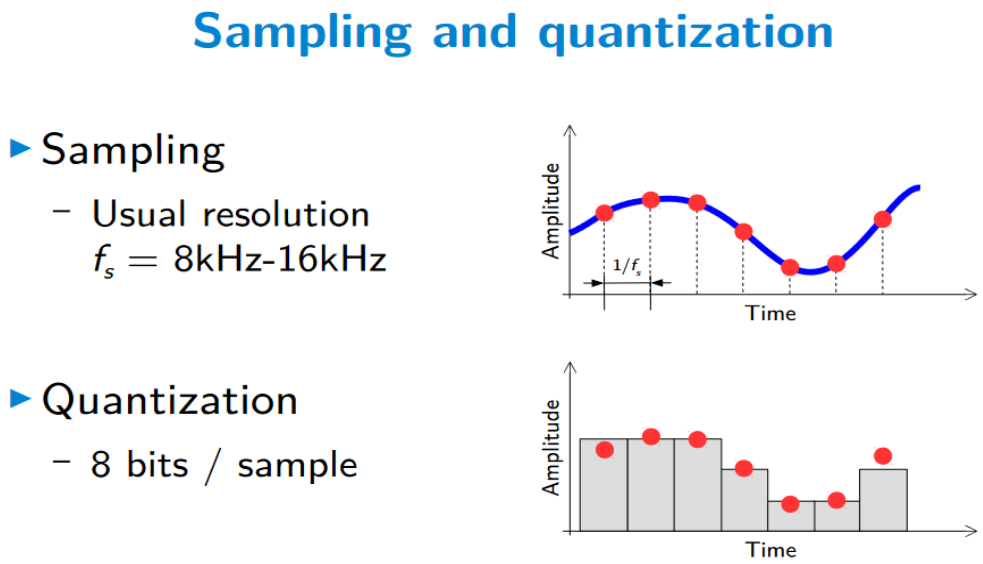


Figura 3 Muestreo y cuantización de una muestra de audio

Fuente: (Lecorvé, 2022)

La limpieza de este archivo de audio hoy en día se realiza de forma automática como se observa en la figura 3. La muestra de audio puede ser parametrizada en función de tiempo y amplitud como una onda, esto permite aplicar filtros de corrección que automatizan el proceso, recortando segmentos de grabaciones que contengan: ruido de estática, música o ruidos de fondo, bajo volumen, mala entonación o interferencias sonoras en el espectrograma de audio. Esto es de suma importancia para lograr la correcta identificación de fonemas pronunciados pues cualquier error en su identificación, posición o reconocimiento, puede terminar en un error de reconocimiento o utilizando una palabra similar que esté fuera de contexto, generando un error en la transcripción resultante.

A continuación, se describen los diferentes métodos de obtención de características, mediante los cuales el motor es capaz de extraer información parametrizada y específica, obtenida de la fuente de voz para posteriormente convertirla en el texto resultante, el cual se acomoda de forma visual en la interfaz gráfica que utiliza el usuario, el texto resultante se puede ir mostrando de forma volátil, ser utilizado para la ejecución de un comando o ser almacenado en un documento de texto para algún posterior procesamiento.

5.1.1.1.2 Extracción de características

Existen diferentes métodos para clasificar, ordenar y realizar el reconocimiento de voz mediante estadística clásica, estos métodos se agrupan en 9 modelos diferentes, cada uno con diferentes estructuras metódicas, que implementan un algoritmo diferente para extraer las características principales de una voz, asociando su entonación, acento y ritmo con características matemáticas que permitan individualizar estas particularidades de la voz en parámetros medibles, estos parámetros son conocidos como características de voz y se obtienen procesando la onda de espectro acústico. Los métodos principales para extraer y comparar estas características de voz se describen en la tabla a continuación.

Tabla 2 Métodos de extracción y comparación de características

Codificación Linear Predictiva (LPC)	La idea primara es que el modelo de habla puede ser descrito como mezcla linear de anteriores muestras de habla, la señal digitalizada es separada en un numero de muestras N, cada muestra individual de audio es enmarcada para minimizar cortes de señal, cada una de estas muestras enmarcadas son auto - asociadas con muestras anteriores.
Coeficiente central de Frecuencia-Mel (MFCC)	Este procedimiento aplica diferentes fases de transformación y modulación a la señal de origen, primero el espectro de onda es cortado para reducir la interferencia, después la señal se separa por pausas de sonidos y a cada una se le aplica una transformada de Fourier discreta, a este resultado se le agrega a un banco de algoritmos donde se filtra su frecuencia Mel y se compara con el anterior para simular la percepción acústica humana.

Deformación de tiempo dinámica	Este proceso se utiliza para medir la probabilidad de entre dos secuencias o más que pueden tener diferente velocidad de lectura, construidas en programación dinámica si el propósito es alinear dos listas con vectores de características de forma, haciendo varias iteraciones hasta que ambas coinciden entre sí.
Clasificación de patrones	Es el método de comparar patrones sin identificar con referencias de patrones de sonido, calculando una similitud entre estos. Después de entrenar el sistema en el periodo de prueba, los patrones son clasificados según su velocidad y las siguientes aproximaciones se basan en estos patrones.
Aproximación estadística.	Este método independientemente de la velocidad se basa en las disimilitudes del lenguaje para realizar una determinación estadística.

Fuente: extraído de: (Naik, y otros, 2021)

Como se puede observar, estos modelos están compuestos en su mayoría por aproximaciones estadísticas y matemáticas, pues su finalidad principal, es extraer las características de tiempo, frecuencia y representación digital, así como la aplicación de transformaciones lineales y no lineales a la frecuencia de voz, creando un espectrograma. Esto con el fin de transformar frecuencias de voz y extra poner sus características únicas, coeficientes de tiempo y frecuencia. Como podemos observar en la Figura 4 Procedimiento general del reconocimiento de voz, a partir de una muestra de audio, se pueden representar en un espectrograma dos tipos de características.

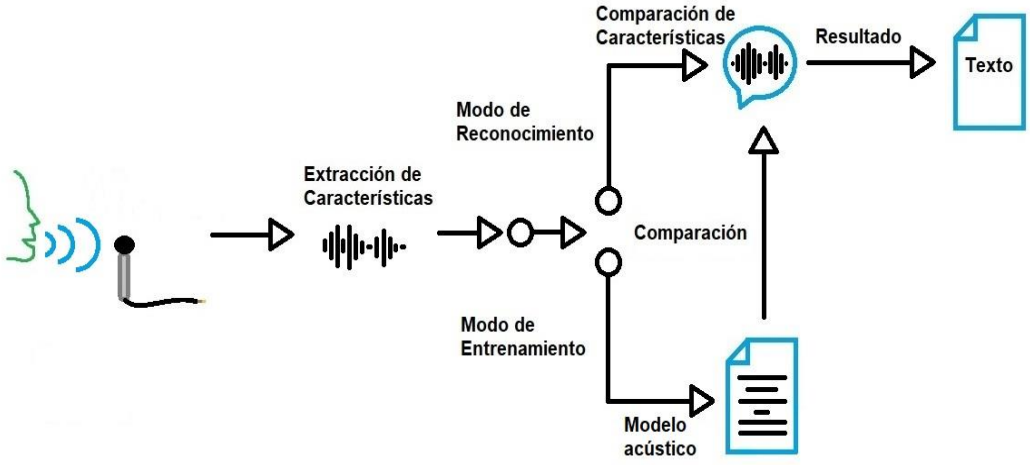


Figura 4 Procedimiento general del reconocimiento de voz

Fuente: elaboración propia.

El reconocimiento de voz se realiza después de extraer las características principales con los métodos mencionados en la Tabla 2 Métodos de extracción y comparación de características este sistema tiene dos modos: uno de entrenamiento, uno de reconocimiento y toda gira en torno a almacenar las características obtenidas, entrenándolas en un modelo acústico para después compararlas con diferentes muestras de audio en el modo de reconocimiento.

5.1.1.1.3 Reconocimiento de voz

Este tipo de reconocimiento extrae los fonemas de la muestra de audio y los formula en palabras, utilizando formulas y modelos matemáticos para identificar y predecir cual es la palabra más similar y probable a la palabra pronunciada, estos modelos crean vínculos entre palabras habladas y modelos de palabras conocidas, para identificar la "mayor probabilidad" de que la palabra sea correctamente identificada. Para esto se necesitan grandes cantidades de datos de información, para crear los modelos acústicos a utilizar.

El proceso de extracción de los fonemas comienza después de que la limpieza del texto ha sido realizada y que sus características han sido extraídas. Lo explican a detalle (Haubold & Kender, 2007) donde en una muestra de audio, filtran el espectro electromagnético y lo analizan comparándolo con muestras de fonemas similares, pues estas son las lecturas dominantes que se obtienen a partir de las lecturas de un espectrograma, haciendo posible identificar un fonema, a partir de frecuencias distintivas que componen señales acústicas producidas por el habla humano, también conocidos

como formante. Como los fonemas son la base fundamental de los lenguajes, el orden de estos determina la organización y el arreglo de los sonidos fonémicos, que son estadísticamente determinantes, para realizar una estimación en la transcripción de estos sonidos fonémicos en cadenas de texto. Esos sonidos fonémicos son estadísticamente determinantes, para realizar una estimación en la transcripción de estos estos sonidos fonémicos en cadenas de texto, como vemos en la figura 5.

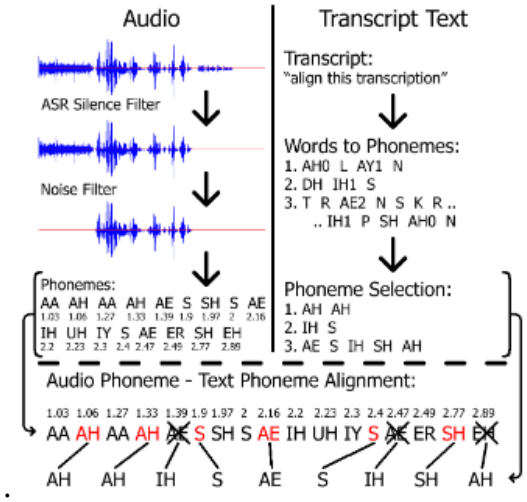


Figura 5 vista de una prueba de alineamiento de una frase muestra sin errores
fuente: (Haubold & Kender, 2007)

Cómo se puede observar el audio es filtrado y seleccionado, se extraen los fonemas utilizando estimaciones de espectro o lecturas de frecuencia dominantes de un espectrograma, el texto temporalmente desalineado es convertido a fonemas y su alineación es realizada utilizando el método de agrupación por distancia estadística.

Existen diferentes métodos para realizar la extracción de los fonemas, cada uno con diferente porcentaje de éxito ante sonidos y fonemas similares, pues las frecuencias

que forman en su definición acústica pueden ser estimadas desde el espectro de frecuencia del sonido, usando un espectrograma o un analizador de espectro acústico. Por lo tanto, siempre es necesario ajustar el método de estimación, orden y transcripción, utilizando un modelo acústico como una referencia estándar aproximada a la entonación, acento y ritmo de la frecuencia ideal, que conforma un formante propiamente característico del lenguaje al cual se desea transcribir. Posterior a esta extracción se realiza un procesamiento matemático que vincula la información pronunciada con cadenas ocultas de Márkov o Redes Neuronales, para predecir tanto el texto actual a transcribir, como la palabra siguiente a ser pronunciada por el interlocutor ordenando así las palabras resultantes en una oración de texto o tomándolas como una ejecución de un comando programado.

5.1.1.1.3.1 HMM Modelos ocultos de Márkov

Los modelos ocultos de Márkov son modelos estadísticos de doble proceso estocástico, con dos variables aleatorias S y O que se transforman de forma secuencial y aleatoria, cuando S representa un estado oculto y O un evento observable, es decir que el proceso estocástico S, no puede ser observado directamente y solo se puede inferir mediante el proceso estocástico O. Como los eventos S no pueden ser observados de forma directa, la finalidad del proceso de Márkov es determinar el proceso oculto S, observando el evento O, esto se realiza observando el cambio de O, influenciado por el cambio de un estado oculto S en un tiempo T, el objetivo final es utilizar la observación O en un tiempo T para adivinar el estado de S. El proceso de Márkov S en sí mismo, no

puede ser observado solo es posible observar la secuencia y el cambio constante en O, partiendo de esa observación, es posible realizar una predicción probabilística para la siguiente observación, teniendo en cuenta que la elección de cada evento S, resultando en una operación O, no depende directamente de los anteriores, pues los parámetros de las variables S son aleatorios y cómo podemos observar el la Figura 5 vista de una prueba de alineamiento de una frase muestra sin errores, cada evento S tiene una vinculación al siguiente, más solo se produce una observación O por cada evento S.

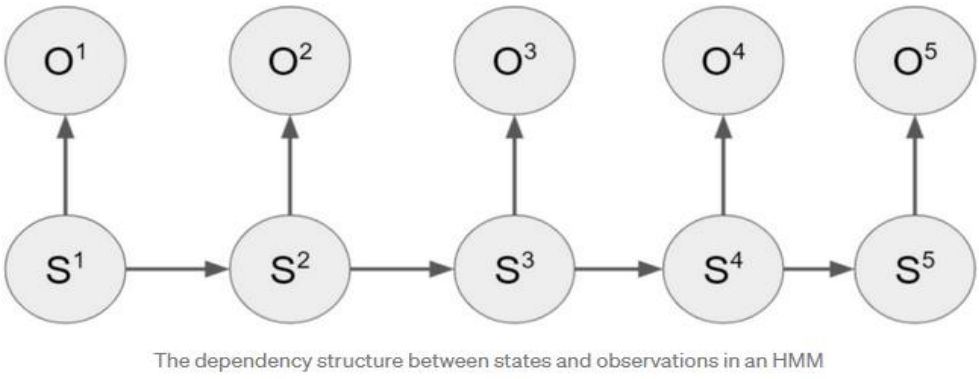


Figura 6 La estructura entre los estados y observaciones en un HMM Fuente: (Field, 2021)

En el modelo oculto de Márkov, se analizan 3 probabilidades. La primera es la probabilidad inicial que se otorga a cada estado S, al comenzar a contar el tiempo T, esta determina su valor inicial en porcentaje único de probabilidad de tener un estado determinado, este comúnmente se representa con el símbolo π. La segunda es la probabilidad de transición, siendo esta la posibilidad de que el estado S cambie o mantenga su estado actual A, de aquí se puede generar una matriz de probabilidades de transición, que representa la combinación de todos los posibles cambios A, que S puede

tener en un determinado T, como observamos en la Figura 6 La estructura entre los estados y observaciones en un HMM Fuente: . Los eventos S generan conexiones A, estas interconexiones representan individualmente la probabilidad de cambio de estados, pues sus conexiones determinan el número máximo de estados a los que pueden cambiar. La tercera es la probabilidad de observación, la cual representa la posibilidad de tener una observación, basada en observaciones anteriores (Vivek C. V., 2020).

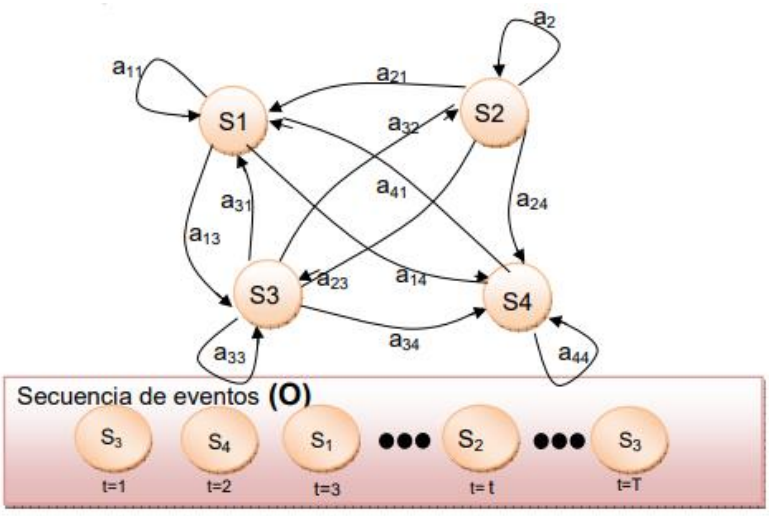


Figura 7 Representación de los Estados S, los enlaces A y los eventos O Fuente: (Mateus, 2008)

Para implementar un modelo HMM en un sistema ASR, se deben relacionar cada uno de los factores anteriormente mencionados, construyendo un modelo acústico donde a cada fonema obtenido del espectro electromagnético, es resultado del proceso representado en la Figura 4. Con estos fonemas se le asigna a esta cadena el número de estados, según el número de fonemas que contiene una palabra pronunciada, según (Mateus, 2008) “por lo regular pueden contener entre 2 y 10 estados” y estos son ordenados en una topología lineal progresiva Figura 8 Topología lineal progresiva

Fuente: , donde los estados se cambian de forma consecuente, según el orden que van entrando, donde a cada estado se le otorga una característica, obtenida en el preprocesamiento de algún método descrito en la Tabla 2.

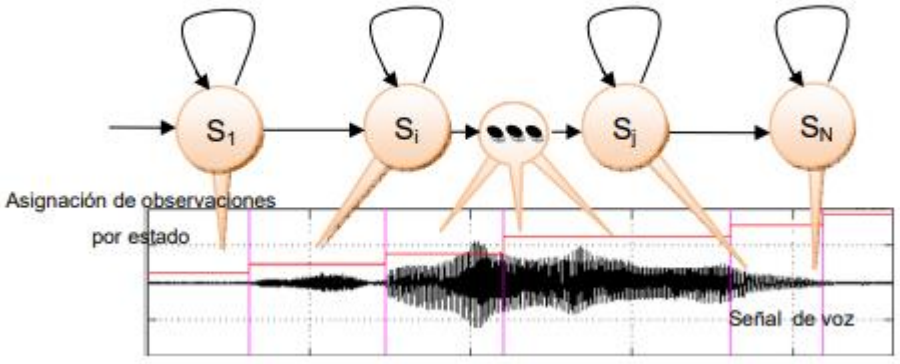


Figura 8 Topología lineal progresiva Fuente: *(Mateus, 2008)*

Para evaluar las observaciones obtenidas de los eventos de cadena, con los sonidos pronunciados por el locutor, es necesario establecer una relación entre cada fonema, silaba y palabra pronunciada, separadas por frases de silencio. Esto se logra relacionando probabilísticamente, la ocurrencia de cada observación con la siguiente, creando así una distribución de probabilidad, basada en las observaciones obtenidas de cada estado, otorgándole un valor probabilístico a esta cadena. Para aumentar ese valor probabilístico, hace falta realizar un ciclo de iteración constante y a esto se le conoce como entrenamiento. Entre más larga es una cadena de observaciones, su distribución de probabilidad es mucho más desarrollada. Por este motivo para obtener una cadena funcional, hace falta entrenar el modelo HMM con bases de datos de diferentes tipos de voz, masculino y femenino, diferentes entonaciones, acentos y edades, para formar

cadenas adaptadas probabilísticamente a diferentes modelos acústicos de voces y locutores.

(Mateus, 2008) Explica que: para clasificar diferentes sistemas de cadenas ocultas de Márkov, se utiliza el algoritmo de Viterbi. Según explican (Churbanov & Winters-Hilt, 2008) “Este algoritmo es un algoritmo de programación dinámico, que encuentra la secuencia más optima, en una sucesión de estados ocultos”. Permitiéndole así al programa de reconocimiento de ASR, identificar la cadena HMM que mejor se ajusta a la distribución de probabilidad, según las características del modelo acústico. Por último, se utiliza el mismo algoritmo de Viterbi dentro de cada cadena, para identificar la palabra desconocida. El algoritmo es capaz de encontrar la palabra, basándose en las observaciones generadas, por la cadena creada de la secuencia de eventos posteriores. Este método se muestra en la Figura 9.

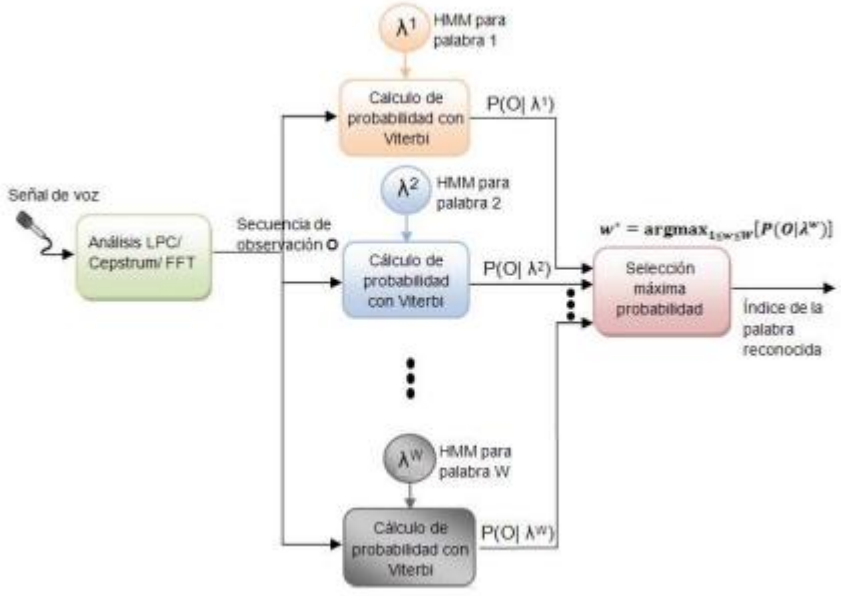


Figura 9 Utilización del algoritmo de Viterbi para identificar palabras Fuente: (Mateus, 2008)

Al basarse en una probabilística, este método requiere un entrenamiento y muestras de tamaño variable, para un estudio comparativo de métodos en ASR en 2014, (Vu, 2014) se utilizaron 100 adultos nativos, para leer 100 frases en 20 idiomas diferentes, para construir diferentes modelos acústicos, complementando la base de datos con la información biométrica de cada participante y comparando la fiabilidad de la traducción, concluyeron que el modelo HMM puede optimizarse junto con una Red Neuronal profunda, para adaptarse a ambientes donde hay muy escasa información de entrenamiento para un nuevo lenguaje, además de cumplir la tarea de transcripción de una forma mucho más eficiente, comparada con otros métodos de reconocimiento de voz dependiente.

5.1.1.1.3.2 Redes Neuronales

Hoy en día las Redes Neuronales, son una de las técnicas más implementadas, especialmente en el reconocimiento de patrones, la Inteligencia Artificial y el reconocimiento automático de subtítulos, estas se componen de neuronas.

El avance más significativo que ocurre en los sistemas ASR, es el cambio de modelo de matemática estadística, para determinar la probabilidad de que varios fonemas compongan una palabra determinada. En los nuevos modelos de Inteligencia Artificial, como las Redes Neuronales artificiales, que incluyen Redes Neuronales de recurrentes y Redes Neuronales Convulsionadas , las cuales se clasifican como Redes Neuronales Profundas y tienen una arquitectura característica, que está inspirada por procesos biológicos, (K. Fukushima 2007) estas son versiones regularizadas de Perceptrones

Multicapa, es decir son redes completamente conectadas, donde cada neurona es una capa, que está interconectada con todas las neuronas de la siguiente capa, como lo podemos observar en la Figura 10.

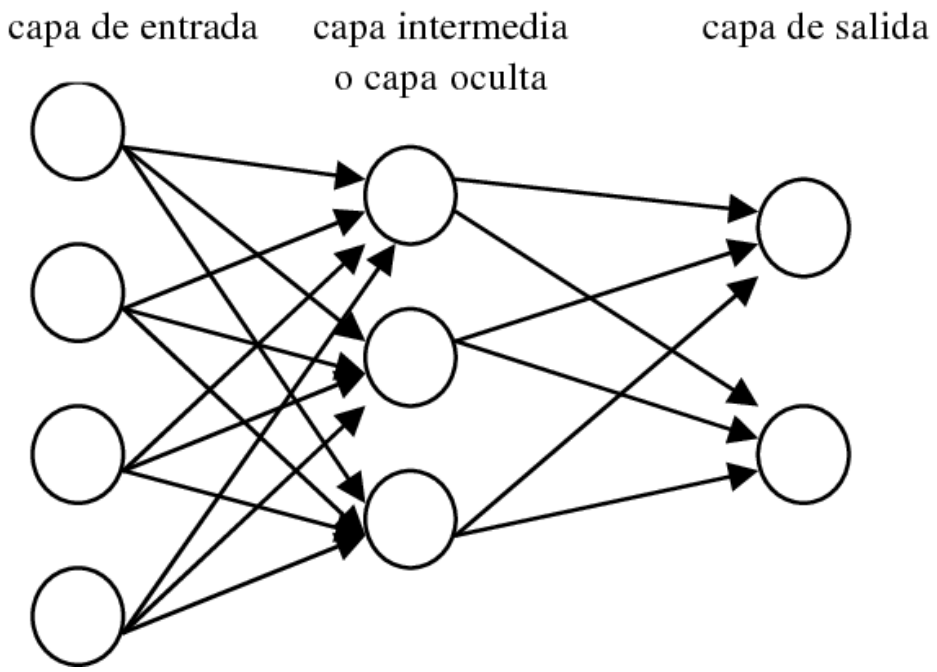


Figura 10 Ejemplo de perceptrón multicapa Fuente: (Botti & Serra, 2001)

Las redes convulsionadas, se diferencian de las HMM, porque las HMM funcionan creando predicciones, estimando los patrones del sistema y asumiendo que sus estructuras son correctas. Pero esta técnica falla si sus predicciones son incorrectas, la implementación de una Red Neuronal no requiere realizar predicciones, este método utiliza una representación de nodos simples, donde sus conexiones son utilizadas para reconocer patrones de habla, a diferencia de las HMM donde el conocimiento o las limitaciones, reglas y procedimientos, no están almacenados en un evento S único, sino que son consecuencia de toda la trayectoria de observación de objetos O, causa que el

conocimiento de esta cadena sea almacenado en una distribución de probabilidad de todos los eventos de información. Mientras que en una Red Neuronal, la identificación de los patrones está condicionada por las conexiones e interacciones de las neuronas de cada capa. Esta arquitectura es más flexible, pues cada neurona ajusta sus conexiones con cada interacción de aprendizaje profundo que realiza.

Las Redes Neuronales también se pueden utilizar en el preprocesamiento del espectro de audio, para obtener características descritas en la Tabla2. Las principales características obtenidas de la señal de audio son: la Frecuencia de Coeficientes MEL (MMFCC) y los coeficientes de codificación linear predictiva (LPCC), ambos resultando en un espectrograma. Según (Trivedi, Pant, Shah, & Sonik, 2018) las Redes Neuronales se pueden utilizar para procesar y clasificar estas características, utilizando 2 capas de entrada, tres capas ocultas y una capa de salida, siendo (LPCC) y (MMFCC) las 2 entradas correspondientes.

Con esta referencia del mapa de espectro visual de las características obtenidas, se procede a segmentarlo, utilizando el algoritmo CTC de clasificación temporal conexionista, algoritmo que permite entrenar la Red Neuronal, para identificar intervalos de tiempo y frecuencia, identificando así pausas y silencios en el espectrograma, para recortar largos segmentos de audio en sonidos simples. Como se puede observar en la Figura 11 el proceso del espectrograma en un algoritmo.

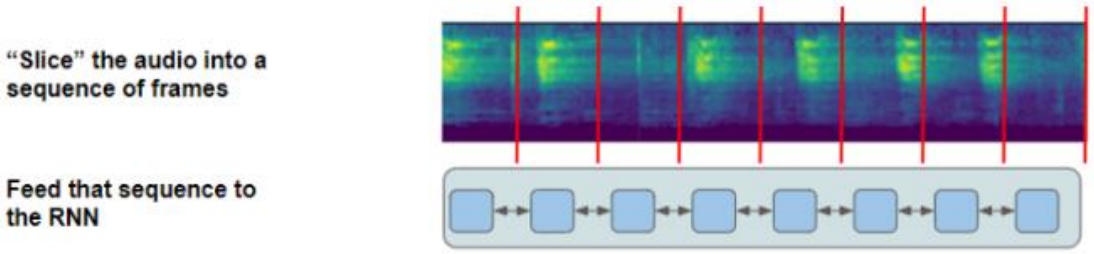


Figura 11 Proceso de recorte de espectrograma según el algoritmo CTC Fuente (Doshi, 2021)

Cada segmento de audio recortado corresponde a un tiempo específico en el espectro del audio, tomando como referencia las características MFCC y LPCC, la Red Neuronal puede determinar las probabilidades de encontrar un fonema en cada segmento, comparando los cambios en su tiempo y frecuencia, con la experiencia previa de fonemas aprendidos en su preentrenamiento. Como lo describe (Mohamed, 2014) "la gran novedad del trabajo presentado en estas secciones, demuestra que se puede lograr un desempeño consistente, realizando un "preentrenamiento" a la Red Neuronal de múltiples capas, una capa a la vez, como un modelo generativo en los segmentos de coeficientes de habla. Este preentrenamiento, hace más simple optimizar Redes Neuronales Profundas, que tienen muchas capas ocultas y también permiten que se puedan utilizar más parámetros, sin sobrepasar su capacidad. Este entrenamiento crea nuevas capas de detección de características, que cada vez se vuelven más complejas, llegando a formar modelos acústicos: "Cuando los fonemas son pronunciados, estos generan una perturbación en forma de frecuencia, como formas de energía temporal en la muestra de audio, pero cada patrón se deforma diferente según su voz". Así que diferentes formas de voz pueden realizar el mismo tipo de sonido, siendo una pequeña diferencia la

deformación en estos patrones que se pueden diferenciar entre sí, abriendo la posibilidad para identificar a un locutor especifico, basándose en el comportamiento de sus deformaciones en su voz, producto de su edad género y su acento.

Como se puede observar en la Figura 12, el fragmento de espectrograma es directamente alimentado a la Redes Neuronales Profundas, que detecta cambios en su frecuencia y tiempo, para identificar patrones similares, a los causados por la deformación de energía en el espectro en busca de fonemas.

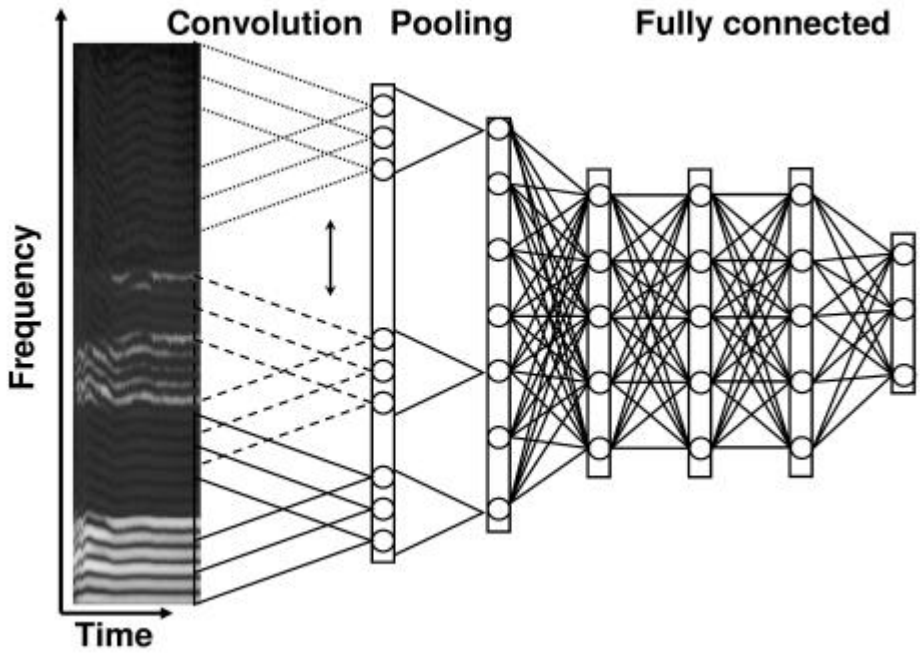


Figura 12 Redes Neuronales Profundas de frecuencia y tiempo. Fuente: (Mohamed, 2014)

Es clave aclarar que estos modelos de Redes Neuronales Convulsionadas pueden ser combinados con otros modelos de reconocimiento de voz, dependiente e independiente de modelos acústicos, para identificar fonemas, transcribirlos y obtener modelos de

lenguaje para reforzar, confirmar o corregir el texto resultante, basándose en la gramática, el léxico y el correcto orden de una oración en un texto generado.

5.1.1.2 Modelo acústico

El modelo acústico es la referencia estadística, mediante la cual operan los modelos HMM y las Redes Neuronales. En él se modela la relación de la señal de audio y las unidades fonéticas de un lenguaje, a diferencia de los modelos de reconocimiento de voz por aproximación estadística, donde solo cuentan con un clasificador de palabras como se observó en la Figura 5 vista de una prueba de alineamiento de una frase muestra sin errores, donde cada palabra única en el modelo acústico tiene como salida una secuencia de fonemas. Esta secuencia de fonemas se puede alimentar directamente a un modelo HMM, donde esta cadena comenzara a modelarse a los patrones repetitivos del lenguaje, utilizando algoritmos de aprendizaje como el algoritmo de Baum-Welsh, las cadenas de HMM aprenden generando un modelo nuevo, a partir del modelo anterior al darle probabilidad a cada fonema en un tiempo determinado. Según (Colompar, 2018) existen diferentes modelos acústicos con diferentes características: mono fonemas, trifonemas, mixturas gaussianas y Redes Neuronales Profundas, cabe aclarar que al utilizar la cadena HMM en el modo de reconocimiento, hay que utilizar el algoritmo de Viterbi.

5.1.1.3 Modelos de lenguaje

Los modelos de lenguaje son una base referencial de datos estadísticos, aplicada a un lenguaje específico. Para que el procesador de lenguaje natural pueda predecir acertadamente la siguiente palabra, se modelan las características del lenguaje natural: el léxico y las reglas del lenguaje (Fonéticas, Gramáticas y Referenciales), siendo los fonemas la estructura más básica que combina estructura de palabras. Según (Reyzábal Manso, 2005), la finalidad del modelo de lenguaje es ayudar al modelo acústico prediciendo mejor la transcripción del audio reconocido, asegurándose que las frases o los conjuntos de palabras, tengan una probabilidad superior ante las frases mal organizadas. Para construir estos modelos, es necesario entrenarlos en un lenguaje específico, para que aprendan a reconocer las características correctas y el orden de las oraciones empleadas en el lenguaje, entre más palabras conoce el sistema, más robusto será el modelo. El entrenamiento se lleva a cabo junto con el modelo acústico, utilizando librerías de voz de forma iterativa y continua, según los requerimientos especificados por el sistema donde se van a utilizar.

5.1.1.4 Librerías

Las librerías voz, son grandes compendios de grabaciones de lectura en diferentes idiomas, estas grabaciones son clasificadas por dialecto, edad, género y acento, por lo general son grabaciones cortas con palabras aleatorias. Como la entonación y el ruido de fondo en el entorno de grabación varía de persona a persona, se deben disponer de criterios de aceptación y validación de las voces almacenadas.

Generar una librería de voz es un proceso arduo y complicado (Vu, 2014), no solo describe el proceso de elaboración del proyecto “Global Phone Corpus”, sino que también hace énfasis en clasificar las librerías de voz en diferentes categorías: primero están los diccionarios de palabras, que contienen la única pronunciación de una sola palabra por archivo de audio, después están las librerías de voz nativas y no nativas, pues los extranjeros a una lengua pronuncian los fonemas de forma diferente. Por último, encontramos los “Dataset” que son los compendios de varias grabaciones, con personas de un mismo punto geográfico, acento, género y edad. Para ordenar todos estos datos e indexarlos a un compendio más grande, Mozilla implementa el sistema de etiquetado IETF BCP-47, el cual es un código estandarizado creado por la “Internet Engineering Task Force” o fuerza de trabajo ingenieril de internet, siendo BCP el acrónimo a “Best Current Practice” o “Mejor práctica actual” en su versión 47, donde cada espacio está separado por varias sub-etiquetas, compuesta enteramente por caracteres en latín. La primera etiqueta denota el lenguaje y las demás las características especiales de ese lenguaje denotando: su alfabeto de escritura, región, variante, extensiones y marcaciones privadas, como se pueden apreciar los ejemplos en la Tabla3.

Tabla 3 Ejemplos de sistema de etiquetado BCP-47

Etiqueta	Forma	Significado
en	Lenguaje	Ingles
de-AT	Lenguaje-Región	Alemán de Austria

es-419	Lenguaje-Región UN M49	Español de Centro y Suramérica, según el código UN M49 de la ONU
de-CH-1901	Lenguaje-Región-Variante	Alemán de Suiza con ortografía de 1901
sr-Cyrl	Lenguaje-Alfabeto	Serbio escrito en alfabeto Cirílico
sr-Cyrl-CS	Lenguaje-Alfabeto-Región	Serbio escrito en alfabeto cirílico de Serbia y Montenegro
sl-Latn-IT-rozaj	Lenguaje-Alfabeto-Región-Variante	Eslovaco escrito en latín utilizado en Italia, con dialecto Resiano.

Fuente: (The World Wide Web Consortium (W3C), 2006)

El BCP-47, permite a los investigadores encontrar el “Dataset” más adecuado para su proyecto con este sistema de etiquetas, pues ellos pueden personalizar su búsqueda por: región, dialecto, alfabetos y variantes, incluso el sistema permite una etiqueta de uso privado, para controlar en las versiones de cada archivo.

Hoy en día, muchas empresas privadas generan sus propias librerías de voz, condicionadas a los estándares adecuados a su localización, generar una librería es un trabajo largo y laborioso, pues se estima que se requieren entre 300 y 500 horas de

grabación, para entrenar un modelo acústico rústico de muy baja calidad y ante mejor infraestructura, además de nuevos modelos de Redes Neuronales, el clamor de los investigadores de Inteligencia Artificial, resulta siendo el mismo. Según (Sean White), en el blog de Mozilla escribió: “una de las razones por las cuales hay muy pocos servicios de voz comercialmente disponibles, es la falta de información, los investigadores o cualquiera que quiera construir tecnologías avanzadas de voz, necesita transcripciones de voz en datos, para entrenar a los algoritmos de aprendizaje, por ahora estos solo pueden acceder a librerías de voz limitadas” (Sean, 2017) y la escasez de estas librerías es evidente, al punto que Amazon y Google empezaron a liberar al público parte de sus librerías de voz, a su vez pequeñas empresas han desarrollado su fórmula de negocio alrededor de esta demanda, produciendo sus propias librerías de datos y mezclándolas con librerías de datos públicas.

5.1.1.5 Interfaces

Existen diferentes interfaces logradas con esta tecnología, mas no existe un estándar general para el formato de salida. Sin embargo, el sistema de reconocimiento de voz, permite interfaces carentes de interacción visual, como consecuencia del reconocimiento de voz. Una de sus aplicaciones principales es en el concepto de computación manos libres, como describe: (Herzog, 2005), donde el resultado del ASR en cadena de texto, es a su vez utilizado como un comando, para ejecutar una acción determinada en un sistema, esto le permite a las personas interactuar de forma natural e intuitiva, la cual juntándose con tecnología TTS o tecnología de texto a voz, elevan la

usabilidad del usuario a una experiencia conversacional. Hoy en día se implementan estas interfaces en diversas áreas de la sociedad, desde asistentes de voz inteligentes en el hogar, hasta ayudantes de navegación en aviones, donde hay sistemas dedicados al reconocimiento automático del control de tráfico, donde la estática y la distorsión causadas por la interferencia atmosférica, confunde a los sistemas ASR, además de ser un sistema embebido y autónomo de conexiones de internet para funcionar. (appareo, 2021)

5.1.2 Ambientes de ejecución

Para que un sistema ASR funcione, este debe estar alojado en un ambiente digital, existen diferentes tipos de ambientes en los cuales se pueden alojar y ejecutar sistemas ASR.

5.1.2.1 Ambientes locales

Estos ambientes, hacen referencia a cualquier sistema de cómputo que cuente con un sistema operativo funcional, en los ambientes locales la principal característica es que el programa de ASR está contenido y ejecutándose de forma local, significa que la ejecución y ubicación del programa, están contenidos en el mismo sistema y por lo tanto, no necesita de una conexión a internet o a otro servicio.

En este ambiente el módulo ASR, se ejecuta como un programa y el sistema anfitrión, le permite utilizar los recursos de hardware conectados al computador, como dispositivos de entrada, le permite asignar alojamiento en directorios de sistema y a su vez controla la utilización de recursos de memoria y procesamiento de su propio sistema.

Cabe aclarar, que en este ambiente es necesaria la instalación del programa principal y sus dependencias de software, para esto es necesario contar con los privilegios de administrador de sistema, el espacio en disco adecuado y la memoria RAM suficiente para ejecutar este aplicativo.

5.1.2.2 Ambientes Embebidos

A diferencia de los ambientes locales, los ambientes embebidos se ejecutan directamente desde un sistema electrónico, el cual ya contiene el programa incrustado en su memoria, siendo este un dispositivo dedicado cuya única función gira entorno al programa y todo el sistema está integrado, según describe (Emilio, 2015) su funcionalidad es operada de forma directa e instantánea. Ejemplo de estos dispositivos: parlantes de asistentes inteligentes, audífonos inteligentes, dispositivos de traducción instantánea, Raspberry-pi y Arduino, en algunos casos los dispositivos necesitarán una conexión a internet o a otro dispositivo auxiliar para su funcionamiento completo, más aún en su lógica primaria más básica, ya contiene el reconocimiento de voz a nivel local.

5.1.2.3 Ambientes en la nube (servidores)

Los ambientes en la nube permiten desplegar el software en un servidor y para acceder al servicio, hay que utilizar una aplicación o un navegador web, el sonido es ingresado por un micrófono, ya sea en conexión directa por la aplicación o a través de un "Web Socket" en el navegador web, el servidor tiene alojado el software de ASR donde recibe, procesa y envía la transcripción, estos ambientes en la nube pueden tener diferentes infraestructuras, sistemas operativos e incluso, pueden funcionar de forma

modular con “kubernetes” de (Google, 2022) o “clústeres” de servidores, para atender las demandas de servicio masivas, el costo aproximado de esta implementación, está relacionado directamente, con la adquisición propia del hardware necesario para alojarlo o el costo de alquiler de un servidor como alojamiento web.

5.1.2.4 Ambientes conexión API

Si no se cuenta con la infraestructura web y se quiere utilizar el ASR como servicio, se puede implementar una conexión API (Aplication Programing Interface), la cual es una conexión, que se incluye desde un programa o un servidor a un servicio alojado en un servidor externo. Las conexiones API permiten conectarse directamente a servicios por demanda, estos también pueden entregar y recibir información de forma directa, Google y Amazon AWS prestan el servicio de traducción ASR mediante este sistema, el costo de la implementación se reduce considerablemente, pues no es necesario implementar infraestructura de hardware para proveer este servicio, más este servicio es pago, en Google su precio es de 0.006 a 0.009 USD por minuto (Google Cloud, 2022) dependiendo de la modalidad, sí almacena o no los datos y la cantidad total de minutos, pues los primeros 60 minutos son gratis.

5.1.3 Infraestructura TI web

La infraestructura TI web, se refiere a todos los componentes necesarios, para que una aplicación web funcione adecuadamente en un ambiente en la nube, esto incluye desde el hardware esencial del servidor, sus componentes: RAM, Procesador, Sistema operativo y ancho de banda con conexión, junto con su infraestructura y topología de red,

conexiones y puertos. Aquí se incluye, una descripción general sobre los servicios de software adicionales, que hacen posible el funcionamiento de la plataforma de forma lógica, sus componentes y su funcionamiento.

La infraestructura general de las aplicaciones en red, se reparte en una topología de red de Front end y Back end, o interface y servidor final, Como podemos observar en la Figura 13 Topología Infraestructura web Elaboración propia, el usuario quien ingresa a la interface a través de una aplicación, soportada por una capa de servicio de conexión, esta petición entra mediante la capa de aplicación directamente al servidor, quien la procesa y responde con información, que a su vez es entregada al usuario por medio de la interface.

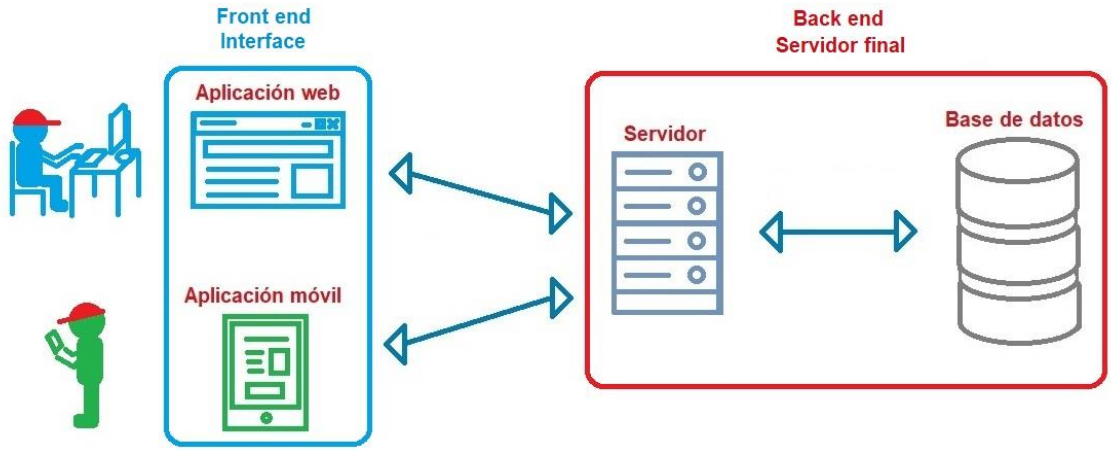


Figura 13 Topología Infraestructura web Elaboración propia

Esta topología de red es modular y escalable, pues permite anexar diferentes capas más complejas de hardware en el servidor final, para atender una demanda mayor de

usuarios, sin realizar cambios en las interfaces o aplicaciones móviles, que afecten la experiencia del usuario.

5.1.3.1 Requisitos del Hardware

Además de los periféricos físicos necesarios, todo programa tiene una descripción de un requisito especifico de hardware Procesador, Memoria RAM, Memoria de video y Disco duro, en donde se describe el mínimo necesario, para ejecutar de forma correcta este programa. Estos requisitos son la sumatoria de todos los requisitos mínimos de cada componente, empezando por el sistema operativo, siguiendo por las dependencias necesarias de cada programa y terminando en la aplicación principal, estos requisitos siempre son agregados en la documentación oficial de cada componente.

5.1.3.2 Diseño del “Front end”

El diseño del Front end encapsula todo lo que corresponde con la interfaz del usuario, todo con lo que el usuario puede ver, leer e interactuar, existen componentes específicos de la interfaz que se interconectan entre sí, para ser la puerta de entrada y salida de la aplicación con la información. En el Front end se aplican técnicas de diseño gráfico, para ubicar y mejorar la usabilidad de la herramienta por parte de los usuarios. Pues una orientación intuitiva y amigable, permite el correcto funcionamiento de la plataforma por parte del usuario, sin que este tenga que enterarse, sobre todos los subprocesos que ocurren en el lado del servidor, pues el usuario no interactúa directamente con él, sino a través de la aplicación.

En el diseño, no solo se ubican los elementos visuales de la aplicación, el diseño incluye detalles como el nombre DNS (servicio de nombres de dominio) de una página web, también incluye elementos modulares que insertan, actualizan e interactúan en tiempo real con el usuario y tienen una programación independiente, además de contenidos multimedia interactivos, en los diseños se puede visualizar como representar toda esta información en un formato simple y adaptado al usuario.

Esta capa interactiva de datos se logra con 2 lenguajes de programación insertados en la interface, quienes en su lógica pueden interactuar directamente con interfaces API, para enviar, obtener datos de terceros y mostrarlos en tiempo real, en navegadores web que soportan esta tecnología, la más utilizada para interfaces actualmente es JavaScript, la cual se explica a continuación.

5.1.3.3 JavaScript

JavaScript, es un lenguaje de programación textual de código abierto y puede ser utilizado tanto en la interface como en el servidor, este lenguaje permite programar contenidos interactivos, a diferencia de HTML y CSS que determinan la ubicación y el estilo de los elementos. JavaScript les permite a las interfaces, crear contenidos de elementos interactivos, que actúen directamente con el usuario y está presente en objetos que utilizamos a diario, como por ejemplo las barras de búsqueda de cualquier sitio de comercio electrónico, cualquier video en formato multimedia, insertado desde una plataforma de videos o el simple hecho de mostrar Tweets o posts de Facebook, siendo JavaScript una herramienta liviana e indispensable para el internet hoy en día. A pesar de

su gran usabilidad, (Cuomo, 2013) describe que JavaScript tiene varias limitaciones, este lenguaje no permite ni la lectura, ni la escritura de documentos por motivos de seguridad, tampoco puede ser utilizado para aplicaciones de conexiones de red, porque no hay soporte para esto y su debilidad más grande, es que no permite el uso de Multi-threading o procesamiento paralelo en los procesadores. Lo que significa que solo un Hilo o "Thread" del procesador, es el encargado de manejar todos los ciclos de eventos de un programa basado en JavaScript, ocasionando un problema de cuello de botella, cuando existen eventos simultáneos en cola de ser procesados de forma asíncrona, para esto en 2009 JavaScript introdujo los "Web Workers", un complemento, que permite correr scripts desde una página HTML, que corre en segundo plano y en un hilo de procesamiento diferente al de la ejecución inicial, a costa de separar e indexar una mayor cantidad de memoria RAM requerida, para escalar el funcionamiento de cada script simultaneo, corriendo en su propio "Hilo". Es por esto que los navegadores modernos, dependen mucho de una RAM con buena capacidad y frecuencia para desempeñarse adecuadamente.

5.1.3.3.1 React

React, es una librería de código abierto construida por Facebook, que opera en JavaScript. Esta librería, permite construir interfaces de usuario, desplegando componentes desde un Documento de modelo de Objetos o DOM, la cual asigna una estructura de datos cache en memoria, procesa los diferentes resultados y actualiza estos mismos en el navegador de forma eficiente. (Reactjs, 2022), React te permite construir

una “SPA” o “Single-Page Aplication”, “Aplicación de una sola página”, que carga en un solo documento de HTML en la primera petición y posteriormente actualiza la porción, o el contenido del cuerpo de la página web, que necesite renovar sus datos. Esto le permite al cliente web, no tener que recargar o actualizar la página, para obtener nueva información y cada vez que un usuario haga una nueva petición, React la intercepta y solo realiza los cambios en las secciones que necesitan cambiarse, sin tener que volver a cargar la página completa, esto permite un mejor desempeño de la página web y una experiencia de usuario más fluida y dinámica.

5.1.3.3.2 Node JS

NodeJs es un entorno de servidor final de código abierto, que funciona en múltiples sistemas operativos y funciona con el motor V8 de JavaScript, lo cual le permite ejecutar código, desde una consola que no está contenida en un navegador, este código funciona de forma asíncrona y sirve para construir soluciones de red escalables. También permite a los programadores escribir códigos de línea de comandos y correr scripts solo en el servidor, para producir contenido dinámico en la página web incluso antes de que esta sea enviada al usuario. Utilizar JavaScript de forma holística, resulta un paradigma de “JavaScript en todas partes” (Cuomo, 2013) , donde JavaScript se transforma en el lenguaje general de implementación, desde la interface hasta el servidor final, su ejecución es lineal y su arquitectura es enfocada en eventos, los cuales pueden ser asíncronos, estos diseños le permiten optimizar la salida de datos y escalar en

aplicaciones web, con muchas operaciones de entrada y salida, así como aplicaciones en tiempo real, comunicación en tiempo real y videojuegos.

5.1.3.4 Diseño del “Back end”

En ambientes TI “Back end” hace referencia al servidor final, donde se ejecutan y se procesan las peticiones provenientes de la interface de usuario, idealmente el usuario no puede interactuar directamente con el servidor final, pues las peticiones siempre serán transmitidas mediante una interfaz de usuario. La infraestructura del servidor, es determinante en el desempeño final de la aplicación, pues determina no solo los tiempos de procesamiento, además del total de conexiones disponibles y el número total de usuarios que puede atender de forma simultánea, cuando se robustece esta infraestructura de servicio esperando una gran afluencia de usuarios, se le conoce como escalamiento y para esto, es necesario separar este servidor en módulos de servicio, utilizando kubernetes y balanceo de cargas, para equilibrar de forma eficiente los recursos, hacia los clientes finales de este servidor.

Es importante recalcar, qué en el proceso de desarrollo, se utilizan diferentes tipos de servidores finales, algunos se utilizan como ambientes de prueba, donde se analiza la estabilidad y los recursos necesarios de una aplicación en pruebas, antes de ejecutarla en un servidor de despliegue, con una escala mayor y topología de red diferente.

5.1.3.5 Python

Python, es un lenguaje de programación interpretado de alto nivel, que permite crear estructuras por niveles y combinarlas con escritura y enlaces, obteniendo

semánticas dinámicas, son muy atractivas para desarrollos de aplicación rápidos y se destaca por su sintaxis, fácil de aprender y de leer, lo cual baja los costos de mantenimiento. En su libro (Beazley, 2009) explica que Python funciona con módulos y paquetes, que permiten mantener la modularidad y reutilizar código, el intérprete de Python, está disponible en código abierto y funciona en todas las plataformas y sistemas operativos.

Cabe aclarar que las diferentes versiones de Python, no son compatibles entre sí y es posible como requisito instalar una versión determinada, para que sea compatible con el programa a utilizar.

5.2 Marco Conceptual

El marco conceptual, es la recopilación, sistematización y exposición de los conceptos principales, para la evolución de una investigación. Por otra parte, permite tanto al investigador, como al lector, compartir los conceptos empleados en el proyecto. (Significados, 2020).

5.2.1 Reconocimiento automático de voz

El reconocimiento automático de voz es una tecnología multidisciplinaria mediante la cual, las maquinas o los sistemas digitales de información, pueden identificar y reconocer el lenguaje natural del ser humano, utilizando un perfilamiento de voz. Esta tecnología tiene múltiples aplicaciones, desarrollos y métodos, en los campos actuales de

investigación y su finalidad principal es la identificación, clasificación y comprensión de nuestro lenguaje, a través de la parametrización de la voz humana.

5.2.2 ASR

ASR, es una sigla en inglés para “Automatic Speech Recognicion” en español, significa reconocimiento automático de habla, a diferencia del reconocimiento automático de voz, este reconocimiento se enfoca en conversaciones con múltiples participantes, por lo cual se enfoca más en identificar oraciones o frases de múltiples palabras, esto se logra por una aproximación estadística y matemática, para predecir en orden secuencial las palabras reconocidas, adaptándolas a un modelo de lenguaje para que tengan sentido gramatical, aunque puede utilizar el reconocimiento automático de voz, para reconocer a los diferentes participantes en una conversación, mediante la parametrización de voz.

5.2.3 Plataforma

En tecnologías de la información y comunicación, los sistemas tienen un Framework o una estructura, la cual se apoya en una plataforma. Las plataformas son los conjuntos de sistemas básicos, para el desarrollo y soporte del software y el hardware. Las plataformas agrupan sistemas similares entre sí, tienen un estándar conjunto y pueden hacer referencia a diferentes conjuntos de sistemas operativos, programas, aplicaciones web o servicios integrados, a nivel de software y hardware.

5.2.4 Web Socket

El Web Socket, es un protocolo de comunicación de sistemas de computación, entre un cliente y un servidor en doble vía, estableciendo una conexión API que permite mediante el navegador, conectarse e interactuar con servidores finales de terceros, para emitir y recibir información en tiempo real. (Internet Engineering Task Force, 2011)

5.2.5 API

API, es el acrónimo en inglés para “Aplication Programming Interface”, que significa interface de aplicación programada, esta interface permite realizar peticiones desde una página web o una aplicación, hacia servidores de terceros con solicitud de información, esta información puede ser entregada o actualizada de forma continua al usuario en la interface. (Bloch, 2018).

5.2.6 Bit-Rate

En sistemas digitales, se utiliza el estándar de “Bit rate” o “tasa de bits”, donde se expresa una tasa de Bits por segundo, para representar un medio continuo de trasmisión de audio o video, después de ser procesado por un codificador y almacenado en un formato de audio. (Smith, 2003) explica que: la tasa de compresión de bits, determina la cantidad de información y datos que se pueden trasmitir en una unidad de tiempo, a mayor tasa de trasmisión, mayor cantidad de datos y mejor calidad de transmisión.

5.2.7 Características de voz

A diferencia de las características de audio, extraídas por la aplicación de funciones matemáticas sobre el audio digitalizado, las características de voz son

singularidades, que ocurren al momento de la pronunciación del lenguaje hablado, estas características están vinculadas con la edad, género y acento de una ubicación geográfica y se dividen en entonación, acento y rimo, estas características se utilizan para construir un modelo de voz e identificar correctamente varias voces diferentes en una

5.2.7.1 Entonación

La entonación, se produce en la vibración de los pliegues vocales, determina el tono con el que se pronuncia un enunciado, la entonación nace a partir de la correcta gesticulación del parlante, generando cambios en las vibraciones y en la frecuencia de las palabras pronunciadas, la relación entre la entonación y las palabras pronunciadas, pueden comunicar emociones e intenciones entre los interlocutores, algo que es muy difícil de imitar por parte de hablantes extranjeros, donde los sistemas entrenados para los hablantes nativos, pueden generar errores de transcripción, pues el modelo de voz está modelado en una entonación diferente a la pronunciada y a pesar de esto, la entonación puede ser un factor parametrizable, para aumentar la eficiencia del reconocimiento automático del habla, según la investigación (Bolaños Araya, Camacho Lozano, & Urrutia, 2017) donde se basan enteramente en el acento, para identificar correctamente el uso de la tilde diacrítica en la transcripción de un texto, basándose enteramente en el tono y no en la intensidad de la pronunciación.

5.2.7.2 Acento

En español, existen diferentes categorizaciones del acento, desde el acento fonético que se divide en:

- Acento prosódico, el cual determina la forma correcta de pronunciar la intensidad en una silaba,
- Acento gramatical, que asigna la tilde (´) para representar la acentuación
- Acento diacrítico, donde la tilde se coloca para diferenciar significados en palabras iguales. (Enciclopedia de Ejemplos, 2022)

En un contexto más elevado, encontramos la definición de acento como una apropiación característica de la apropiación cultural del lenguaje, proveniente de un grupo de personas que se comunican en el mismo idioma, las diferencias son notables entre diferentes regiones geográficas, que cambian radicalmente su pronunciación y sus significados.

5.2.7.3 Ritmo

El ritmo de la voz tipifica el número de palabras que una persona puede pronunciar durante una respiración, incluyendo silencios y pausas entre palabras. La velocidad resultante, puede ser propia de un acento regional o característica de un lenguaje, este parámetro es determinante, no solo para establecer un perfil de voz específico, en especial cuando su velocidad promedio coincide con una muestra de participantes similares, durante el entrenamiento del modelo de lenguaje.

Como el reconocimiento de lenguaje se basa en patrones y amplitudes, un cambio abrupto en la velocidad de pronunciamiento puede resultar en una incomprensión o error, al momento de transcribir los patrones identificados.

5.2.8 Características del lenguaje

La lingüística, es la ciencia que determina la estructura descriptiva de los lenguajes, los cuales son una expresión propia y universal del ser humano. Existen diferentes campos de estudio, que se enfocan en sus particularidades y evoluciones a través del tiempo, gracias a este estudio se pueden identificar, las características esenciales que componen un lenguaje y sus campos de acción, para identificar y segmentar las áreas necesarias, para enfocarse en áreas específicas y de forma multidisciplinar para solucionar los problemas, que interceden en el uso de los lenguajes por parte de las maquinas. Las siguientes ramas de estudio, son las relevantes para esta investigación, ya que gracias a estas se construyen los modelos de lenguaje, donde hoy en día hay investigaciones como (Casado-Mancebo, 2021) , donde se intentan aproximaciones computacionales a esta ciencia, para la correcta interpretación y reconocimiento de estructuras con sus entidades lingüísticas.

5.2.8.1 Fonología

La fonología, es la ciencia que estudia las características particulares de la organización de los sonidos, en los lenguajes y dialectos, además organiza estos sonidos, formando un sistema de fonemas en lenguajes hablados, basado en el alfabeto fonético internacional creado en 1888, donde se utiliza un lenguaje para describir sonidos, utilizando combinaciones de letras para describir “Fonos” que, a diferencia de los fonemas, son sonidos vocales con gesticulación especifica. Estos Fonos, son los que se utilizan para describir los sonidos, que utilizan los fonemas al pronunciarse. (The International Phonetic Association, 2022)

5.2.8.2 Fonemas

Los fonemas, son la unidad más pequeña de un lenguaje, que permite diferenciar palabras entre si basándose en su pronunciación, los fonemas se representan en grafemas. Todos los lenguajes poseen diferentes fonemas en el lenguaje hablado, se clasifican según sus vocales y consonantes, que determinan los rasgos característicos en la pronunciación de un acento regional particular.

5.2.8.3 Morfología

La morfología, estudia la forma en las que se conforman las palabras y sus significados, además de su relación con otras palabras en este mismo lenguaje. Esto se logra, analizando la estructura misma de los fonemas que componen una palabra, para encontrar palabras raíces, prefijos, sufijos y significados en los mismos.

5.2.8.4 Sintaxis

La sintaxis, estudia la combinación de diferentes grupos de palabras y la función, que cada palabra cumple en una oración para expresar significados, además de las reglas y principios gramaticales, que definen la estructura gramatical del lenguaje.

Marco Legal

5.2.9 Leyes

LEY 982 DE 2005 (agosto 2) Por la cual se establecen normas tendientes a la equiparación de oportunidades para las personas sordas y sordociegas y se dictan otras disposiciones. (Anexo 1)

5.2.10 Decretos

Decreto 457 de 2010. Por el cual se el cual se imparten instrucciones para el cumplimiento del Aislamiento Preventivo Obligatorio.

Artículo 1°. Aislamiento. Ordenar el aislamiento preventivo obligatorio de todas las personas habitantes de la República de Colombia, a partir de las cero horas (00:00 a.m.) del día 25 de marzo de 2020, hasta las cero horas (00:00 a.m.) del día 13 de abril de 2020, en el marco de la emergencia sanitaria por causa del Coronavirus COVID-19.

5.2.11 Circulares

Circular 020 del 16 de marzo de 2020, expedida por la Ministra de Educación Nacional, dirigida a gobernadores, alcaldes y secretarios de educación de Entidades Territoriales Certificadas en Educación, en aplicación de lo dispuesto en los numerales 5.1 y 5.2 del artículo 148 de la Ley 115 de 1994, el artículo 5 de la Ley 715 de 2001, y los artículos 2.4.3.4.1. y 2.4.3.4.2 del Decreto 1075 de 2015, Único Reglamentario del Sector Administrativo de Educación Nacional, ordenó a las secretarías de educación en todo el territorio nacional ajustar el calendario académico de Educación, para retomar el trabajo académico a partir del 20 de abril de 2020.

5.2.12 Licencias de Software Libre

En este proyecto, se utilizaron varias licencias de Software Libre, aquí se deja un compendio de cada una con su respectivo Anexo.

5.2.12.1 Mozilla public Licence

Mozilla Public License Version 2.0 (Anexo 2)

Licencias NGIX (Anexo 3)

Licencia Python (Anexo 4)

Licencia NodeJS (Anexo 5)

Licencia React (Anexo 6)

Licencia Tensorflow (Anexo 7)

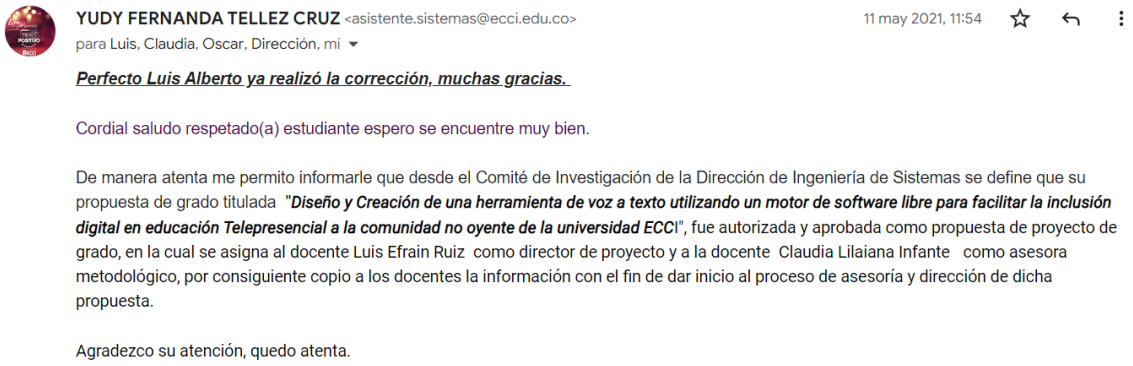
Licencia YAM (Anexo 8)

6. Ingeniería de Requerimientos

6.1 Acta Inicio del Proyecto

El 11 de mayo se dio inicio al proyecto, donde fue aprobado en una reunión tele presencial por el comité de investigación de la Universidad ECCI, debido a la pandemia, el acta de grado quedo consignada como un mensaje de correo electrónico, confirmando

su aprobación desde la coordinación de sistemas de la Universidad ECCI.



YUDY FERNANDA TELLEZ CRUZ <asistente.sistemas@ecci.edu.co> 11 may 2021, 11:54

para Luis, Claudia, Oscar, Dirección, mí

Perfecto Luis Alberto ya realizó la corrección, muchas gracias.

Cordial saludo respetado(a) estudiante espero se encuentre muy bien.

De manera atenta me permito informarle que desde el Comité de Investigación de la Dirección de Ingeniería de Sistemas se define que su propuesta de grado titulada "***Diseño y Creación de una herramienta de voz a texto utilizando un motor de software libre para facilitar la inclusión digital en educación Telepresencial a la comunidad no oyente de la universidad ECCI***", fue autorizada y aprobada como propuesta de proyecto de grado, en la cual se asigna al docente Luis Efrain Ruiz como director de proyecto y a la docente Claudia Lilaiana Infante como asesora metodológico, por consiguiente copio a los docentes la información con el fin de dar inicio al proceso de asesoría y dirección de dicha propuesta.

Agradezco su atención, quedo atenta.

Figura 14 Acta de inicio de proyecto fuente: elaboración propia

6.2 Fases de implementación

6.2.1 Metodología

La metodología aplicada en esta investigación fue la metodología Investigación-Acción, donde primero se realiza una investigación para identificar una problemática, se prueba esta acción sobre un tiempo determinado utilizando observación directa, se evalúa y posteriormente se decide si se aprueba o se cambia por otra acción.

Según (de Luna & Expósito López, 2011) estas son las 4 etapas que construyen un proceso de investigación-acción.

- 1 diagnostico: Identificar el problema.
- 2 planificación: Construir y ejecutar un plan de acción.
- 3 observación: Observar recolectar y analizar la información.
- 4 reflexión: Evaluar e interpretar los cambios y resultados.

Se ha elegido este tipo de investigación ya que la problemática se identificó mediante observación directa y por experiencia propia del investigador, además esta

metodología se alinea directamente con los objetivos planteados por el proyecto al permitir:

- Identificar la problemática.
- Diseñar un modelo de servicio.
- Desarrollar una herramienta.
- Documentar su instalación.

Todo esto en un mismo ciclo investigativo, que le permite al estudiante establecer etapas concisas para cada uno de estos objetivos. A continuación, se explica cómo se aplicaron las etapas de la investigación acción participativa, de acuerdo a la Figura 15 Diagrama flujo Investigación-acción fuente:.

Figura 15 Diagrama flujo Investigación-acción fuente: (de Luna & Expósito López, 2011)

6.2.1.1 Identificación de necesidades

En esta etapa se identificó, como las plataformas académicas utilizadas por los estudiantes en educación tele presencial, no priorizan sus espacios para la inclusión académica de comunidades sordas, se realizó una tabla comparativa de cada plataforma, analizando si cuenta cada una con una funcionalidad de transcripción de subtítulos, para asistir a esta comunidad y se identifica la oportunidad para desarrollar un prototipo de esta herramienta.

Se participó de forma directa en un conversatorio sobre la comunidad sorda realizado por bienestar estudiantil, donde se plantearon preguntas directas para identificar las necesidades de los estudiantes de esta comunidad en ambientes virtuales.

6.2.1.2 Construir y ejecutar un plan de acción

En esta etapa se creó un cronograma de actividades, primero para realizar la documentación del proyecto de investigación y segundo para la creación, diseño y desarrollo del prototipo.

A partir del cronograma de actividades, se ejecutaron cada una de las etapas donde se inició, obteniendo la información necesaria de las actuales plataformas de ASR, a través de esta información se iniciará con la elección de un motor de ASR de Software Libre, para la posterior adaptación, creación y diseño del prototipo, para ser implementado en el entorno de pruebas y posteriormente en el entorno de despliegue.

6.2.1.3 Observar, recolectar y analizar la información

En esta etapa se realiza un estado del arte utilizando el modelo gavilán, para identificar cómo han estado evolucionando las diferentes metodologías y plataformas de ASR hasta el día de hoy, esto para conocer cada una de las características de las mismas y analizar, qué tipo de infraestructura es la que mejor se adapta para la implementación de un prototipo, según las necesidades identificadas en el paso anterior. Basándose en la arquitectura, se identifica cual motor es el más apto para implementar en esta herramienta y posteriormente se documenta toda la información de instalación, configuración e implementación, de las dependencias necesarias para la correcta ejecución del prototipo, además de categorizar y documentar, en que parte de los procesos se presentaban fallas que pudieran afectar la implementación de esta herramienta.

Se realizaron pruebas de estabilidad, conexión y precisión de transcripción para observar la viabilidad de una posible implementación, como servicio funcional de la Universidad ECCI.

Se recolecta la información obtenida de estas pruebas y posteriormente se utiliza, para replanificar su corrección en el desarrollo del prototipo.

6.2.1.4 Reflexionar e interpretar los resultados y replanificar

Según los errores obtenidos se replanificó el desarrollo, corrigiendo y mejorando aspectos técnicos encontrados en la fase anterior.

Posteriormente, se realizó la documentación final de la implementación, la migración del prototipo del entorno de pruebas a el entorno de despliegue, se realiza una implementación final donde se da apertura del servidor al tráfico web y se le asigna un

DNS, para que este prototipo sea verificado en funcionamiento por parte del equipo evaluador.

Se analizaron los resultados de las pruebas realizadas en la fase anterior, con el fin de realizar un informe final de recomendaciones, para una eventual implementación de un servicio de traducción automático brindado por la Universidad, con el objetivo de ser utilizado en un ambiente académico y en base a este, se realizaron las conclusiones finales del proyecto.

6.3 Identificación de necesidades

Este proyecto nació como una iniciativa propia durante el transcurso del segundo semestre académico del año 2020, se identificó la problemática por primera vez mediante observación directa, participando en las mismas clases y grupos de trabajo junto con una compañera de la comunidad sorda, altamente destacada en el ámbito académico por su participación en el proyecto Land-Rover de la NASA, quien reiteradamente expresaba su frustración de no contar con una herramienta para realizar un guardado y edición de los subtítulos automáticos de diferentes plataformas, explicando como la atención de un participante sordo en una clase, se divide entre las explicaciones del profesor, la traducción en señas del traductor y el sistema de subtitulación, por lo cual era indispensable poder interactuar, para realizar la acción constante de "copiar, pegar y exportar” el texto de la transcripción de subtítulos de apoyo en estas plataformas, algo que aún no es posible hoy en día. En el 2020 Google Meet, no contaba con subtítulos de apoyo en idioma español y los docentes utilizaban diferentes plataformas audiovisuales

en sus contenidos, de esta observación se construye la **¡Error! No se encuentra el origen de la referencia.**, donde se evidencia que las múltiples plataformas académicas y audiovisuales, tienen una limitada capacidad en servicio de asistencia de subtítulos, ya que este servicio es prestado mediante plataformas de pago, identificando la necesidad y oportunidad de utilizar Software Libre para disminuir el costo de las licencias implementadas en este servicio.

Posteriormente, se intentó indagar en el área de bienestar con una encuesta realizada al inicio de la pandemia, sobre los navegadores y sistemas operativos más utilizados por los estudiantes, esto para identificar la necesidad de adaptar el desarrollo de este aplicativo hacia la mayor audiencia posible, lo cual fue imposible porque la Universidad se negó a entregar esta información, debido a que la encuesta también incluía información privada de los estudiantes. Ante esta negativa, nació la necesidad de enfocar el desarrollo de esta herramienta hacia un ambiente que integre todos los sistemas operativos y navegadores disponibles, dando como resultado una implementación como aplicación web.

Para finalizar, en junio de 2022 se realizó un encuentro webinar de bienestar universitario, sobre la inclusión digital de la comunidad sorda, titulado “Webinar de inclusión: Conociendo la persona y la cultura sorda” Figura 16

Figura 16 Webinar de inclusión digital Fuente: Bienestar Universidad ECCI

En este espacio se difundieron todos los esfuerzos que realiza la Universidad ECCI y su adaptación académica, para trabajar directamente con esta comunidad. En el espacio de preguntas, la ponente que trabaja de forma muy cercana a estos evidenció la necesidad de contar con todas las tecnologías disponibles de apoyo, adaptadas hacia la colectividad sorda.

De todas estas observaciones, las necesidades se pueden resumir en:

- Funcionamiento en múltiples plataformas y sistemas operativos
- Reducir costos de licencias, utilizando Software Libre
- Subtítulos de apoyo en idioma español.
- Posibilidad de copiar, pegar y exportar el texto traducido.

6.4 Investigación Preliminar estado del arte

Esta investigación, busca determinar cuál es el estado del arte actual en plataformas y motores ASR, esto con el fin de escoger un motor de Software Libre, como núcleo principal de la herramienta de trascripción de voz a texto, en esta investigación se

incluyen motores y plataformas de Software Libre y de uso privado o pago como una alternativa, por si la Universidad decide escalar este servicio sin realizar inversiones en hardware, para esta elección se utiliza el modelo gavilán, el cual es un modelo de recolección y análisis de información, muy utilizado en metodología de la investigación. Una vez sea seleccionado el motor se procede a diseñar y desarrollar la herramienta alrededor de este, para que la interfaz y la infraestructura sean completamente compatibles con el mismo.

6.4.1 Modelo Gavilán

El modelo Gavilán, fue desarrollado por Gabriel Piedrahita y recibe el nombre de gavilán por ser su apodo en la juventud, murió a la edad de 22 años en un accidente aéreo, antes de graduarse como profesor en Harvard. La propuesta del modelo surgió de las dificultades observadas en otros modelos educacionales y en lograr que un estudiante, logre una investigación exhaustiva de alta calidad, de forma que los docentes guías puedan desarrollar actividades, que le permitan al estudiante aprender una metodología específica, para buscar información confiable de forma óptima. Esta metodología, es un modelo simple, fácil de entender y puede ser aplicado a temas y problemas específicos, enfocados a la recolección y búsqueda de información, consiste en cuatro pasos básicos, los cuales se describen a continuación:

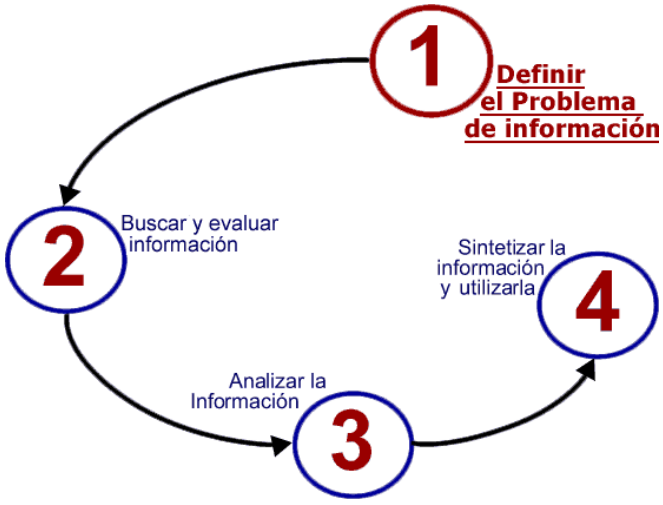


Figura 17 Modelo Gavilan obtenido de http://eduteka.icesi.edu.co/articulos/modelo-gavilan-paso1
Autor: Luisa Fernanda González

Las 4 etapas del modelo gavilán son:

- Definir el problema: Para llevar a cabo una buena investigación, la primera cosa que se necesita realizar es definir y delimitar un problema de investigación a través de una pregunta la cual se va a investigar, esto para enfocar la investigación a identificar en que se está trabajando, cual es el objetivo y que se necesita realizar para lograrlo.
- Encontrar y evaluar información: una vez determinado el tema de la investigación, se planea como se va a investigar y se toma acción, se realiza una búsqueda completa de información en diferentes fuentes, evaluando cual puede ser la más indicada y valida entre todos los tipos de investigación, esta información se debe ordenar y clasificar según su contenido.

- Analizar la información: el paso anterior se enfoca en encontrar, acceder y clasificar fuentes de información, en este se realiza la evaluación critica de esta con el uso de recursos coherentes según la información sobre la cual se trabaja. Con esta se intenta responder la pregunta planteada en el primer paso.
- Sintetizar la información y utilizarla. El último paso se enfoca en tener la información extraída de la investigación, generando contenido o dando respuestas reales a las preguntas y problemas de investigación, dándole un significado y una utilidad al resultado, generando un producto a partir del mismo y a su vez solucionando el problema de investigación.

El modelo gavilán es relativamente reciente, pero su modelo es simple y fácil de aplicarlo en investigaciones académicas universitarias, pues les permite a los estudiantes mejorar sus habilidades de investigación y manejo de la información, los cuales hoy en día son indispensables en una sociedad computarizada. Este modelo, no fue creado tomando en cuenta la alta presencia de información en las tecnologías de información y comunicación actuales, con altos índices de acceso y un gran volumen de fuentes de información no confiable, contradictoria o desactualizada, ya que este modelo se aplica o se adapta a todos los tipos y procesos de investigación.

6.4.2 Problema de investigación

Las aplicaciones de ASR requieren un núcleo o un motor, donde se lleva a cabo el proceso de conversión de voz a texto, no todos los motores son iguales y no todos los motores están adaptados al ambiente donde uno los quiere implementar, por eso se debe evaluar diferentes motores y plataformas que presten este servicio, para basar y orientar el desarrollo de la herramienta alrededor del mismo. Para guiar esta investigación según el modelo gavilán, se debe crear una pregunta problema y preguntas secundarias, con aspectos importantes y relevantes a la investigación. Según las necesidades identificadas previamente, se puede inferir que el motor debe:

- Funcionar en múltiples plataformas y sistemas operativos.
- Reducir costos de licencias utilizando Software Libre.
- Subtítulos de apoyo en idioma español.
- Posibilidad de copiar, pegar y exportar el texto traducido.

Teniendo estas necesidades en cuenta, se procede a plantear la pregunta problema.

¿Qué motor o plataforma ASR, está mejor adaptado para un ambiente web?

Con esta pregunta problema ya planteada, se procede a separar los conceptos asociados a la investigación, de macro a micro para crear un plan de investigación, los conceptos identificados son: Comunicación, Tecnologías de información y comunicación, Reconocimiento de voz a texto, Desarrollo web.

Y se procede a crear las preguntas secundarias:

¿En qué plataforma, ambiente y sistema operativo funcionan?

¿Cuáles de estos motores son de Software Libre?

¿Cuáles incluyen soporte para subtítulos en español?

¿Qué librerías de voz se pueden implementar en el entrenamiento de este motor?

Con estas preguntas planteadas se procede al siguiente paso.

6.4.3 Búsqueda y recolección de información

El plan de investigación determina la posibilidad de elaborar Bitácoras de búsqueda, para llevar una trazabilidad de la investigación, enfocada a resolver las preguntas planteadas. (ANEXO 9: Bitácoras de búsqueda de información)

A partir de estas bitácoras, se recolectó la información necesaria para realizar el estado del arte, sobre las plataformas ASR que se muestra a continuación.

6.4.3.1 Estado del arte Plataformas ASR

6.4.3.1.1 Google API

Google tiene su propio motor de voz integrado en sus aplicaciones, soportan YouTube y Google Clasroom, pero además vende sus servicios como soluciones corporativas en su suite de Google Cloud, donde incluye el servicio de Voz a texto como una solución Cloud, (Google Cloud, 2022) es decir el motor y la aplicación están alojadas directamente en la nube, desde la cual envían y reciben peticiones de traducción de voz a texto a través de un API, esto no solo facilita su implementación sino que posibilita a las empresas a aplicar estas soluciones en diferentes situaciones, desde mejorar el servicio al cliente y transcribir contenido multimedia generando subtítulos automáticos, hasta adaptar dispositivos inteligentes a funcionar con lenguaje hablado.

Google Speech soporta 125 lenguajes y acentos y tiene un costo entre 0.004 y 0.009 centavos de dólar, por cada 15 segundos de traducción y un límite máximo de 1

millón de minutos al mes, por lo cual su licencia es propietaria y comercial, más aún es una alternativa económica, considerando los costos de mantenimiento de una solución propia, más solo en casos donde la conectividad a internet no esté limitada. (Google Cloud, 2022)

6.4.3.1.2 Mozilla Deep Speech

Deep Speech es un motor de voz a texto de código abierto, que utiliza un modelo entrenado en técnicas de Machine Learning basadas en cinco capas de unidades ocultas donde las primeras 3 y la capa 5 utilizan reactivación linear rectificada siendo la cuarta capa, una capa recurrente la última capa implementa una solución Softmax para seleccionar la letra de alfabeto más probable en cada punto de tiempo esto está debidamente documentado y explicado por: (Hashemnia, 2021), implementando herramientas como Tensor Flow que permiten hacer de esta implementación algo más simple y se puede descargar el código fuente con sus librerías ya pre-entrenadas en inglés, para operar este programa de modo “offline” o sin conexión, también permite generar modelos de entrenamiento nuevos, adaptarlos a diferentes lenguajes o acentos lingüísticos y exportar los modelos, para compartirlos de forma libre a la comunidad en su propia página como se puede evidenciar en (Mozilla, 2022), el software de entrenamiento viene integrado junto con el motor de voz, al igual que permite fácilmente importar voces y datos desde la plataforma Mozilla Voice directamente, lo que facilita el proceso de desarrollo para investigaciones “in situ” o implementaciones de software autónomo, que no requieran conexión a internet.

El motor está basado en diferentes Redes Neuronales Convulsionadas (CNN o DCNN), junto con (HMM), estas permiten identificar patrones en espectros de sonido con archivos de audio y lenguaje natural, estas son una evolución de las tradicionales Redes Neuronales de la Inteligencia Artificial. El motor de Mozilla Voice se basó en la investigación (Hannun, y otros, 2014), donde su investigación se enfocó en reconocer voz en dos ambientes con y sin ruido, utilizando grandes volúmenes de librerías de voz, para entrenar el modelo de lenguaje y optimizaron el proceso mediante el uso de tarjetas gráficas (GPU), siendo este un referente en el cambio de paradigma de los sistemas de reconocimiento de voz, pues su mayoría hasta ese momento estaba basado en tecnología HMM.

6.4.3.1.3 IBM

Como parte de sus servicios corporativos, IBM diseño su propia herramienta como una rama de sus servicios y portafolios conocidos por el nombre de Watson, estos servicios provienen de un sistema de cómputo, desarrollado en el proyecto DeepQA por un equipo de investigación liderado por (Ferrucci, Levas, Bagchi, Gondek, & Mueller, 2013), el cual había sido construido posteriormente para participar en el juego televisivo Jeopardy, este servicio ofrece 500 minutos por mes con 38 modelos pre - entrenados gratis y su versión paga a 0.01 y 0.02 dólares el minuto con tiempo ilimitado. (IBM, 2022).

6.4.3.1.4 Amazon Voice

El Gigante de las compras Amazon, compró la empresa Ivona Software en 2013, para competir con el asistente Siri de Apple (Lunden, 2013), con esta tecnología lanzó al mercado a su producto Alexa, como asistente de voz y de hogar, sumando a su portafolio de servicios de Amazon Web Services, la suite de transcripción automática de texto, donde ofrece 60 minutos gratis de traducción al mes, (Amazon, 2022)

6.4.3.1.5 Kaldi

Kaldi, es un conjunto de herramientas para reconocimiento de voz programado en C + +, está licenciado bajo la licencia apache 2.0v, Kaldi fue desarrollado en 2009 en la Universidad Johns Hopkins de Baltimore, Maryland. Su enfoque fue crear un Modelo espaciado mezclador gaussiano (SGMM) (Vivek C. V., 2020) Fue posteriormente mejorado en un encuentro de investigadores, patrocinado por la Universidad tecnológica de Brno en República Checa 2010, su código fue liberado posteriormente en 2012 por Daniel Povey (Povey, 2022), quien trabajaba como investigador para Microsoft, más aún cuenta con la contribución de 70 programadores participantes a lo largo de su trayectoria, incluyendo sus librerías Scripts y actualizaciones. Este conjunto de herramientas fue adaptado a múltiples entornos de sistemas pues Kaldi funciona de manera modular, separando el motor de las librerías permitiendo la flexibilidad a los investigadores de construir o probar sus modelos de voz de forma dinámica, así se han recopilado diferentes adaptaciones e implementaciones de Kaldi en otros lenguajes de software como Python, listos para ser compilados y adaptados a entornos Android, Windows,

MAC o Linux y es activamente actualizado, por una comunidad de desarrolladores libres que adoptó el proyecto.

6.4.3.1.6 Julius

Este motor de reconocimiento de voz utiliza un decodificador de reconocimiento de vocabulario continuo LVCSR y está enfocado para investigadores y desarrolladores, utiliza sistemas HMM con fonemas de trigramas y mantiene estructuras modulares, su plataforma principal es Linux y tiene una licencia de Software Libre BSD y es este, uno de los motores ASR más antiguos, pues su lanzamiento fue en 1991 como un proyecto de la Universidad de Tokio. (Lee, Kawahara, & Shikano, 2001).

6.4.3.1.7 Wav2Letter++

Desarrollado por el equipo de investigación de Inteligencia Artificial de la empresa anteriormente conocida como Facebook hoy en día Meta. Programada en C++ y posteriormente agregado a la suite de Inteligencia Artificial Flash light, Wav2letter++ fue anteriormente liberada como código libre y licencia BSD para los investigadores, su motor se basa en un escáner léxico en el modelo acústico, que busca la estructura directamente en la pronunciación de los fonemas, optimiza su ubicación con Redes Neuronales y cuenta con licencia MIT. (Pratap, y otros, 2019).

6.4.3.1.8 Vosk

Es un set de herramientas basado en Kaldi y pensado para sistemas embebido pues es uno de los motores más livianos incluye soporte para 20 idiomas y cada uno pesa

alrededor de 50 MB, viene con un sistema de conexión API para mejorar la experiencia del usuario, incluye varias librerías de código abierto y permite realizar identificación, reconocimiento de voz entrenamiento de modelos y limpieza de datos. Su licencia es “Apache 2.0”. (Alpha Cephei, 2022).

6.4.3.1.9 Athena

Es una implementación de un motor de voz a texto punto a punto se enfoca en brindarle una plataforma a la industria y al sector académico para experimentar con nuevos modelos de procesamiento de voz, su motor de voz está basado en Redes Neuronales construidas en Tensorflow C++ y trae una licencia “apache 2.0”. (Li, Sun, Lei, Zou, & Zhao, 2022).

6.4.3.1.10 ESPnet

Es un set de herramientas de procesamiento de voz de punto a punto permite hacer identificación reconocimiento y traducción simultánea de texto, utiliza el motor Pytorch de aprendizaje profundo y sigue el procesamiento de estilo de Kaldi, para la extracción y el formato de las características de audio está construido para ser implementado en Python y cuenta con una licencia libre “Apache 2.0”. (Yalta, Hayashi, & Yalta, 2022).

6.4.3.1.11 Coqui

Es un startup de Alemania fundada por desarrolladores de Mozilla , quienes crearon la organización Coqui, dedicada a realizar proyectos de Software Libre, su nombre proviene de una especie de rana nativa de Suramérica, la cual es pequeña, pero tiene una voz muy clara y fuerte, fundada con el apoyo de inversores privados. Coqui ha

tenido una fuerte acogida en su desarrollo como Software Libre. (Casanova, Gölge, Meyer, davis, & Morais, 2022).

6.4.3.2 Librerías de plataformas ASR

Las de voz, son grandes compendios de grabaciones de lectura en diferentes idiomas, estas grabaciones son clasificadas por dialecto, edad, género y acento por lo general son grabaciones cortas con palabras aleatorias, como la entonación y el entorno de grabación varía de persona a persona, se deben disponer de criterios de aceptación y validación de las voces almacenadas.

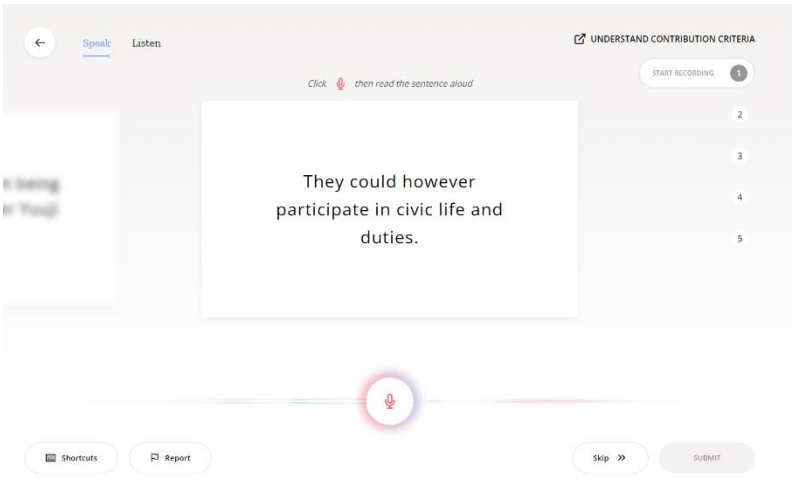


Figura 18 Interfaz de contribución voluntaria Mozilla Voice

Fuente: elaboración propia

Muchas empresas privadas generan sus propias librerías de voz condicionadas a los estándares adecuados a su localización, generar una librería es un trabajo largo y expenderse pues se estima que se requieren entre 300 y 500 horas de grabación, para entrenar un modelo de voz rustico de muy baja calidad y ante mejor infraestructura con

nuevos modelos de Redes Neuronales, el clamor de los investigadores el Inteligencia Artificial resulta siendo el mismo, según Sean White en su blog escribió “ una de las razones por las cuales hay muy pocos servicios de voz comercialmente disponibles, es la falta de información, los investigadores o cualquiera que quiera construir tecnologías avanzadas de voz, necesita transcripciones de voz en datos para entrenar a los algoritmos de aprendizaje, por ahora estos solo pueden acceder a librerías de voz limitadas” y la escasez de estas librerías es evidente, al punto que Amazon y Google, empezaron a liberar al público parte de sus librerías de voz, a su vez pequeñas empresas han desarrollado su fórmula de negocio alrededor de esta demanda, produciendo sus propias librerías de datos y mezclándolas con librerías de datos públicas, a continuación se listarán las librerías de voz más completas del mercado actual.

6.4.3.2.1 Mozilla Voice

La Fundación Mozilla , lanzó en junio de 2017 su proyecto Mozilla Voice llamado Common Voice, el cual es un proyecto de Crowdsourcing para crear una base de datos de software de reconocimiento de voz libre y abierto, el proyecto está apoyado por voluntarios que graban muestras de voz con un micrófono y revisan o aprueban grabaciones de otros usuarios, las frases transcritas por los voluntarios se recolectan en una base de datos de dominio público con licencia CC0 (Creative Commons Zero), que le permite a los desarrolladores el uso de esta base de datos, para aplicaciones de voz a texto sin restricciones legales, costos o limitaciones de licencia.

En su lanzamiento, más de 20000 usuarios en todo el mundo habían grabado 500 horas de lecturas en inglés, en julio de 2020 la biblioteca almacena más de 7226 horas de grabaciones de voz en 54 lenguajes, de las cuales 5591 han sido verificadas por voluntarios.

Mozilla Common Voice incluye una plataforma donde cada usuario es motivado a grabar audio, donando su voz para el proyecto o verificar audios grabados. La grabación se realiza en línea y no es necesario registrarse para participar, más aún si el usuario determina crear una cuenta, Mozilla sugiere incluir información demográfica sobre el mismo, Edad, Género y País, para clasificar la voz del usuario de forma más precisa en sus librerías.

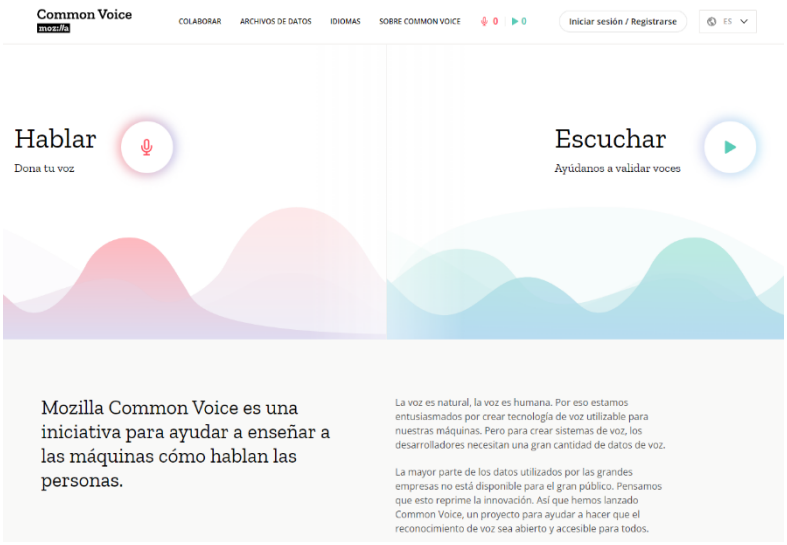


Figura 19 Página principal Mozilla Common Voice. Fuente: Elaboración Propia.

La función escuchar le permite a cualquier persona colaborar, verificando que los audios grabados y previamente donados, correspondan exactamente al texto brindado en el momento de la grabación, según los criterios de contribución una grabación debe ser rechazada, si el participante realiza errores gramaticales o pronunciaciones equivocadas,

las más comunes son omitir o agregar: palabras, tildes o sinónimos similares no descritas en el texto a leer, tener un ruido de fondo excesivo, contaminación de otras voces en la grabación , volumen inadecuado y grabaciones de sintetizadores electrónicos. (Mozilla Corporation, 2022). Este proceso de revisión es esencial para garantizar la calidad de los datos obtenidos ya que si la biblioteca contiene disparidad de información entre los audios y textos de lectura podrían afectar gravemente el funcionamiento de cualquier motor de voz a texto que incluya esta biblioteca, estas grabaciones marcadas como erróneas por dos o más personas no son eliminadas sino reubicadas en librerías de compendio de errores para que los investigadores puedan experimentar con ellas.

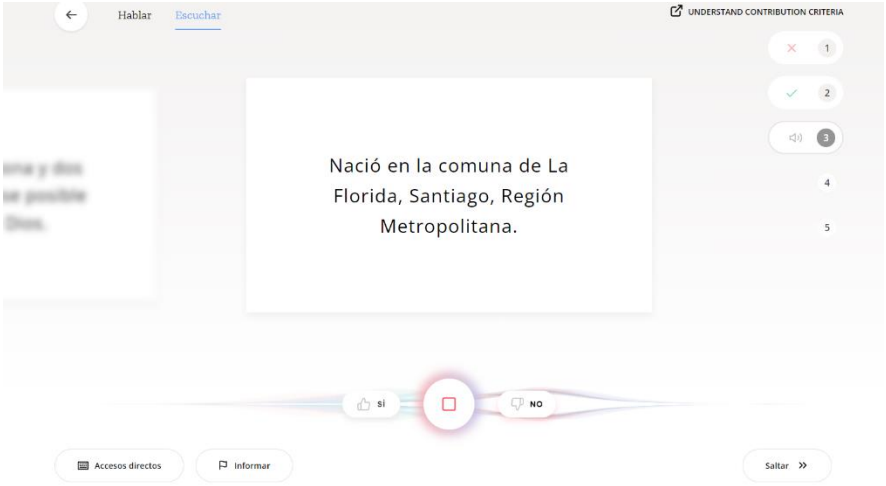


Figura 20 interfaz de verificación Mozilla Voice Commons Elaboración propia

Mozilla es bastante transparente con su sistema de validación, ya que les permite a las comunidades de usuarios registrados, agregar nuevas frases de lectura o personalizarlas a sus temas y contenidos específicos, incluso incluye herramientas para obtener frases cortas de Wikipedia o exportar las frases contenidas ya existentes, al igual que le permite a los usuarios reportar frases inadecuadas, para evitar que sean leídas y

agregadas a la biblioteca. Los usuarios registrados de Mozilla pueden participar en las decisiones de gobernanza del proyecto, según su contribución y experiencia en la plataforma, pueden votar y sugerir propuestas de cambio en los lenguajes ya existentes de la plataforma. Todos los lenguajes compilados incluyen un código BCP-47, el cual es un sistema de ordenamiento flexible que le permite a las comunidades marcar información clave como dialecto, región y ortografía. La idea detrás de este modelo es permitirles la flexibilidad a las comunidades, de crear compendios de datos más pequeños y enfocados en diferenciar vocabulario, gramática, acentos, diferencias de pronunciación y dialectos, para que el data set sea enfocado a una localización demográfica, específica y certera. Hoy en día, Mozilla Voice ya cuenta con más de 80 comunidades establecidas en su sistema abierto de gobernanza. (Mozilla, 2022).

6.4.3.2.2 Google audio set

Este Dataset es una colección a gran escala de clips de sonido de 10 segundos obtenidos de videos de YouTube. Para recolectar estos datos fue necesaria la supervisión de anotadores humanos que verificaron la presencia de voces en cada segmento obtenido. Todo el modelo de obtención, categorización, clasificación y almacenamiento fue documentado en (Gemmeke, y otros, 2017). Estos datos se encuentran separados en 527 categorías de clasificación diferentes, que no solo incluyen conversaciones de voz sino también sonidos: humanos, animales, maquinaria y música. Su descarga es de dinámica libre, pues le permite a los investigadores adaptar el contenido solicitado a sus requerimientos de investigación, sin embargo esta librería de sonidos, no está adaptada

como librería de aprendizaje para entrenar modelos de voz, pues los audios de monólogos y conversaciones, no están separados por: idioma, país, región, ni acento y tampoco cuentan con una transcripción a texto, para comparar resultados y entrenar modelos, esto debido a que el dataset está orientado como prueba a la implementación de motores de Inteligencia Artificial, que buscan identificar idiomas, contenidos o descripciones automáticas de un contenido aleatorio, para modelar "percepción artificial", logrando que las máquinas puedan percibir sonidos, aprender modelos y recolectar grandes cantidades de información de eventos, definiendo relaciones jerárquicas que existen entre diferentes sonidos. Esta librería permite a cualquier investigador solicitar una descarga de sonidos, con los dataset listos para ser utilizados y está licenciada bajo (CC BY-SA 4.0) por lo cual se considera Software Libre.

6.4.3.2.3 OpenSLR

Es una biblioteca compilatoria de recursos recopilatorios para herramientas ASR que incluye formatos de audio, texto y software, ordenados por lenguaje e incluso acento desde charlas TED, hasta compilaciones realizadas por voluntarios; se destaca la compilación SRL72 donde se incluye una librería con voces de Colombia, realizada por la Asociación de recursos de lenguaje europea en 2020. (Trmal, 2022).

6.4.3.2.4 Lingua Libre

Es un proyecto colaborativo desarrollado por Wikipedia, que busca construir una biblioteca colaborativa, multilingüe y audiovisual, bajo una licencia abierta. (CC BY-SA 4.0) Para esto el software permite a los voluntarios, grabar frases y palabras en una lista creada y definida, según las categorías lingüísticas en demanda el participante lee las

palabras que se muestran en la pantalla y el software automáticamente va cambiándolas, cuando detecta un silencio después de una palabra leída, este principio fue tomado de un Software Libre llamado Shtooka, creado por Nicolas Bion y este sistema de grabación, permite grabar cientos de palabras por hora, las grabaciones son subidas automáticamente desde el cliente web a la librería de Wikipedia commons, estas pueden ser consultadas y descargadas desde la misma plataforma o utilizadas, para ilustrar entradas en wikictionarios (Wikipedia, 2022) o pronombres y títulos en artículos de Wikipedia, estas grabaciones son usadas conjuntamente con proyectos de procesamiento de lenguaje natural, entre ellos Mozilla DeepSpeech y Mozilla Voice .

6.4.4 Análisis de la información obtenida

La investigación arrojo un total de 11 Motores y plataformas de Voz y 4 Librerías de plataformas ASR de Software Libre, algunos motores no fueron tenidos en cuenta, ya que utilizan partes componentes y métodos de los motores aquí mencionados, además de enfocarse en otras tecnologías similares como “Text to Speech” o entrenamientos de modelos de lenguaje. En esta investigación se determinó que la compatibilidad de estos motores con las librerías depende de su formato de modelo acústico y de lenguaje, existen diferentes estándares y cada motor es compatible específicamente con algunos formatos, se evidenció que muchos motores de Software Libre llevan bastante tiempo sin ser actualizados.

Se pueden identificar motores con diferentes enfoques donde unos priorizan la estabilidad y servicio mientras que otros mantienen una estructura flexible y abierta, más apta para investigación y experimentación.

Algunos motores tienen una preferencia y especificaciones directas sobre su sistema operativo, mas hoy en día es posible implementarlos en ambientes impropios a los documentados gracias a la virtualización.

Se encontró que todos los motores son entregados como un módulo y su interfaz principal es una consola nativa en el lenguaje que fueron desarrollados.

Aunque los motores utilicen diferentes tecnologías para operar todos requieren dependencias extra para operar.

Se encontraron diferentes comunidades de desarrolladores e investigadores trabajando en conjunto sobre estos motores, para mejorar y optimizar los modelos de lenguaje y la precisión de estos.

6.5 Selección del motor de Software

Antes de realizar la evaluación de los resultados de investigación, se procede a responder las preguntas secundarias en la siguiente tabla, basándose en la información obtenida.

Tabla 4 respuesta a preguntas secundarias fuente: elaboración propia.

Pregunta secundaria	Respuesta
¿En qué plataforma, ambiente y sistema operativo funcionan?	En su mayoría son multiplataforma y funcionan tanto en servidores de Windows como en Linux, los ASR basados en

	API's, no necesitan un sistema operativo para alojarse,
¿Cuáles de estos motores son de Software Libre?	Los motores que funcionan con Software Libre encontrados son: (completar)
¿Cuáles incluyen soporte para subtítulos en español?	Todos los motores incluyen subtítulos en español, si se les instala el modelo de lenguaje con el idioma español. En los API, hay que hacer una modificación en el llamado de la conexión, para establecer el idioma deseado a transcribir.
¿Qué librerías de voz se pueden implementar en el entrenamiento de este motor?	Se realizo una investigación separada, enfocada enteramente en librerías de voz para responder esta pregunta.

Habiendo contestado todas las preguntas secundarias, se procede a realizar una evaluación por criterios percentiles de 10 preguntas con un puntaje total de 100, orientados a evaluar cada motor encontrado y documentado, con fin de responder la pregunta problema principal, para esto se definieron los siguientes criterios de evaluación.

Criterios de evaluación: Para definir si un motor esta está mejor adaptado para un ambiente web, se tomaron los siguientes criterios de evaluación: 1 Sistema operativo

multiplataforma, 2 requisitos de RAM (4GB de RAM o menos), 3 procesador (2.5 GHZ o menos), 4 almacenamiento en disco en megas (menos de 500 MB) , 5 Consumo de ancho de banda estable a máximo 10m/s , 6 número de dependencias requeridas mínimo 2, 7 Soporte de instancias y particionamiento de sesiones , 8 comunidad de desarrollo soporte y trayectoria / más de un año , 9 código libre / si , 10 modelo modular y flexibilidad de plataforma para cambiar o adaptar lenguajes. A cada uno de estos criterios se le asigna un valor entre 1 a 10, al ser 10 criterios en total el máximo puntaje es 100. A continuación, se muestra la tabla de evaluación que compara los criterios con las plataformas investigadas, para encontrar la más ideal a implementar como motor en el proyecto:

Tabla 5 Evaluación de plataformas y criterios

fuente: elaboración propia

Plataforma/criterio	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**Total**
Mozilla Deep Voice	10	10	10	10	10	10	10	10	10	10	100
Kali	5	10	8	10	7	8	5	5	10	5	73
Julius	5	10	7	10	5	10	5	0	10	5	67
WAV2Letter++	10	8	10	10	10	7	10	5	10	10	90
Vosk	10	10	10	10	10	8	10	5	10	10	93
Athena	10	10	10	8	10	10	10	5	10	10	93
ESPnet	10	10	10	9	10	10	10	5	10	10	94
Coqui	10	10	10	10	10	10	10	10	10	10	100

Se puede observar que hay un empate técnico entre Mozilla Deep Voice y coqui, ambos motores son muy similares en sus características principales pero escogió Mozilla Deep Voice, debido a su trayectoria y antigüedad ya que fue lanzada en 2017, a diferencia de coquí que fue fundado en 2016 y su motor de voz a texto fue lanzado en 2021 la trayectoria de una herramienta facilita su implementación, gracias a la experiencia precia que ya han tenido varios usuarios en la comunidad para encontrar y solucionar errores.

6.6 Especificación de requisitos de la infraestructura

6.6.1 Sistema Operativo

Mozilla Deep Voice, funciona tanto en ambientes Linux como Windows, basado en una implementación de Tensor Flow, el cual es un motor de Python y es compatible con todos los sistemas operativos.

6.6.1.1 Windows

Para implementar Python en Windows, el sistema requiere mínimo 2 GB RAM y recomienda 4GB de RAM, cualquier procesador x86 64-bit CPU (Intel / AMD architecture) de 2.00 GHz y Windows 7 a Windows 10 como sistema operativo.

6.6.1.2 Linux

Para implementar Python en Linux, el sistema requiere mínimo 2 GB RAM y recomienda 4GB de RAM, cualquier procesador x86 64-bit CPU (Intel / AMD Architecture) y Linux- Ubuntu 16.04 o 17.10, aunque cualquier versión de Linux que soporte esta configuración de hardware funcionaria sin problema.

6.7 Especificación de requisitos del servicio

En esta sección, se describen los requisitos para prestar el servicio de la herramienta web, basándose en el motor de voz a texto escogido previamente.

6.7.1 Servidores

Es necesario tener un servidor de Hosting para almacenar y proveer el servicio de transcripción de voz mediante la herramienta. Para probar esta herramienta, se utilizó un servidor de pruebas y desarrollo de Microsoft Azure, posteriormente se implementó la herramienta en un servidor de la nube de Google Cloud Platform.

6.7.2 Navegadores WEB

El usuario final, necesita acceder al servicio mediante un navegador web. Este navegador web debe contener soporte para conexión Web Socket y JavaScript, actualmente la mayoría del mercado soporta estas dos tecnologías, siempre hay algunas excepciones.

6.7.2.1 Navegadores soportados

Este modelo de herramienta funciona plenamente en un navegador de última generación, son los más populares y más utilizados actualmente por los usuarios de internet, entre estos navegadores encontramos a Mozilla Firefox v7.1, Google Chrome v106, Opera v 9.1, Safari 14.1.2 y Brave 1.42.97, además de navegadores alternativos basados en el navegador Chromium de Software Libre.

6.7.2.2 Navegadores No soportados

Esta herramienta, no es soportada por navegadores antiguos de generaciones pasadas, como Internet Explorer 3, Netscape 1, Mosaic, Tampoco funciona en navegadores de texto como Lynx. Cabe aclarar que, en navegadores de última generación, existen extensiones anti - publicidad que deshabilitan el uso de JavaScript, lo cual imposibilitaría la ejecución de la herramienta.

6.8 Diseño y elaboración de la propuesta

Se tomaron en cuenta diferentes modelos de conexión con sus respectivos casos de uso, a su vez que la realización de diferentes Mockups, para orientar el desarrollo de la interface en su diseño gráfico.

6.8.1 Diagramas y diseño

En esta sección se exponen los diferentes diagramas modelos y diseños realizados en el desarrollo del proyecto.

6.8.1.1 Modelo de conexión API-P2P

Esta toma muestras de audio constantes del computador anfitrión y las envía al servidor final, donde estas son procesadas y traducidas de voz a texto, quien a su vez entrega el texto vía API a todos los clientes conectados a esa misma instancia o salón de clase, la comunicación inicia en el anfitrión quien se conecta directamente (Web Socket) mediante un cliente al servidor para iniciar la comunicación con varios usuarios.

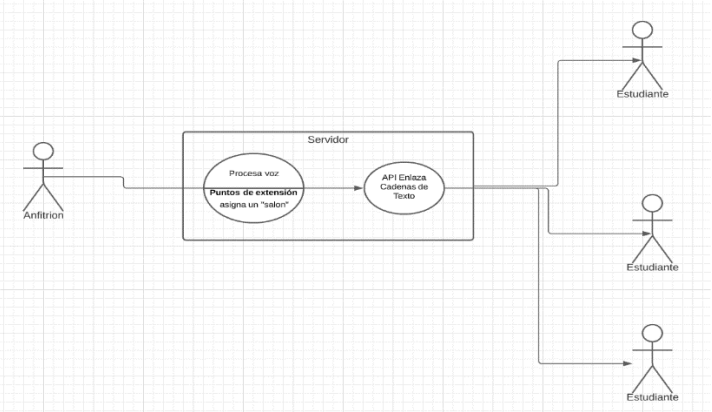


Figura 21 Modelo de caso de uso conexión API-P2P fuente elaboración propia

Este modelo permite una comunicación directa de uno a muchos usuarios este modelo solo es útil para Charlas y no conversatorios donde hay un solo anfitrión, o en situaciones donde no hay más de una persona hablando y participando de la conversación.

6.8.1.2 **Modelo voz en off**

Este utiliza el Micrófono propio del computador del estudiante para procesar el sonido ambiente, siendo el mismo su propio Anfitrión en la instancia del servidor, útil para conversaciones donde participan varios usuarios, contenidos audiovisuales y plataformas donde no hay subtítulos de apoyo.

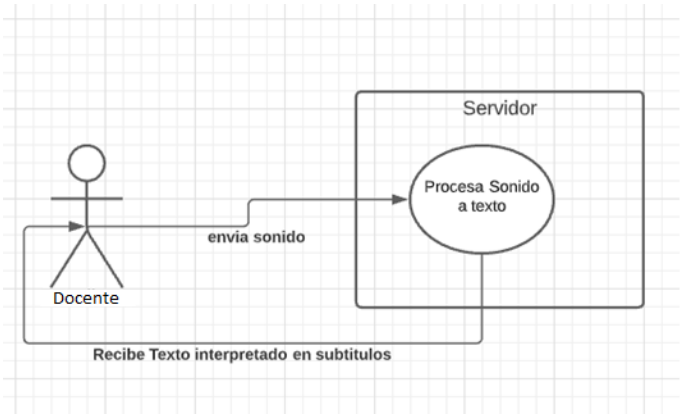


Figura 22 Modelo Voz en off fuente: elaboración propia

En esta infraestructura se destaca la participación individual de un solo usuario, haciendo uso de la herramienta esto supone que cada usuario utiliza un String o una instancia individual en el servidor, en vez de compartir la misma instancia colectiva con diferentes usuarios. Lo cual es mucho más eficiente en términos de instancias pues solo tiene un hilo singular de procesamiento, donde puede integrar distintas voces de profesores y estudiantes de manera simultánea, sin tener que escalar nuevas instancias o abrir nuevos hilos de procesamiento en el servidor final.

6.8.1.3 Comparaciones Ventajas y desventajas

6.8.1.3.1 Modelo Conexión API-P2P

Tabla 6 comparación pros y contras modelo API-P2P fuente: elaboración propia

Pros	• Menos colisión, estática y ruido, ya que solo un participante emite voz. • Fácilmente escalable para varios usuarios, pues una instancia solo tiene que procesar una fuente única de voz. • Los registros de una única fuente de voz, son más livianos que múltiples fuentes de voz, lo cual

	permite guardar estas fuentes en una biblioteca de voz, para entrenar al motor a identificar un tipo de voz específico.
Contras	• Otros participantes quedan fuera de la comunicación. • Difícilmente escalable si se desean agregar más participantes, pues requiere más procesamiento, ancho de banda y conexión.

6.8.1.3.2 Modelo Voz en Off (ejecución del cliente)

Tabla 7 pros y contras modelo voz en off fuente: elaboración propia

Pros	• No requiere la participación de terceros para operar • Permite transcribir conversaciones ASR con múltiples participantes. • Permite transcribir contenidos audiovisuales eventos y conferencias en vivo. • Puede ser ejecutado sin internet, instalando el Motor de transcripción en el mismo dispositivo.
Contras	• Puede sobrecargar al servidor en peticiones si se utilizan muchas estancias al mismo tiempo, pues cada estudiante requiere una instancia propia, incluso si se encuentra en el mismo “salón” o “evento”. • Puede que la calidad del micrófono se vea afectada por

	ruidos de ambiente-estática y esto afectar la calidad de los subtítulos generados. • Requiere que el estudiante tenga en un alto volumen el contenido, para que este sea captado por el micrófono, incomodando a terceros presentes en ese espacio. • Utiliza más recursos de procesamiento y almacenamiento en el servidor.

6.8.1.3.3 Diseño de la interface

El diseño estaba en un principio encaminado hacia el desarrollo de una extensión de navegador ubicada en la parte lateral de la interface como se puede observar en la Figura 23 Mockup extensión navegador fuente: elaboración propia, se planea aplicar primero un piloto de prueba en modo de extensión solo con el fin de comprobar la funcionalidad del servidor final. En el mockup, se observa cómo se pronostica su uso en una plataforma, que no cuenta con subtítulos de apoyo y que actualmente tiene material publicitario de la Universidad ECCI.

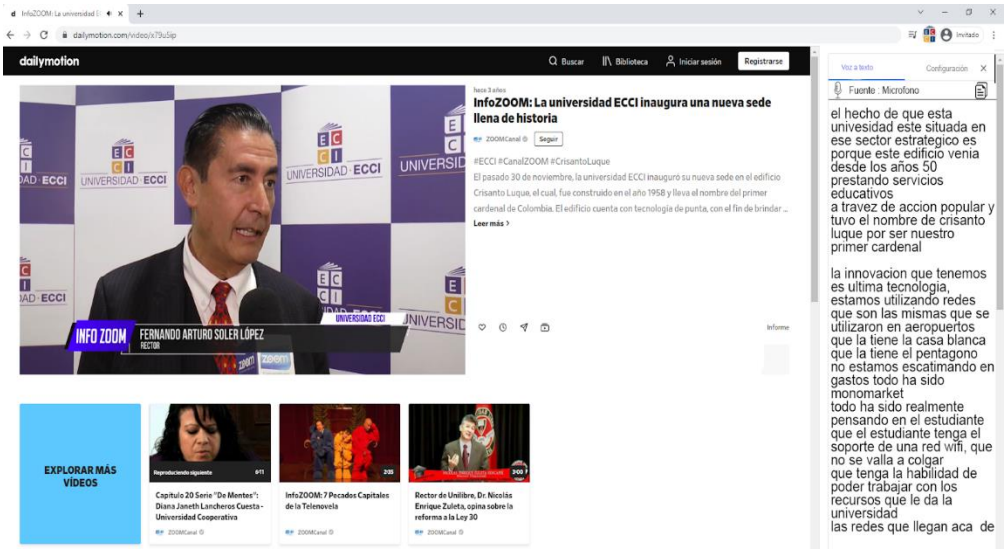


Figura 23 Mockup extensión navegador fuente: elaboración propia

En la **¡Error! No se encuentra el origen de la referencia.**, se observa una descripción de todos los elementos planeados para agregar a la plataforma, con su respectiva descripción de las funcionalidades requeridas, según las necesidades y requerimientos planteados.

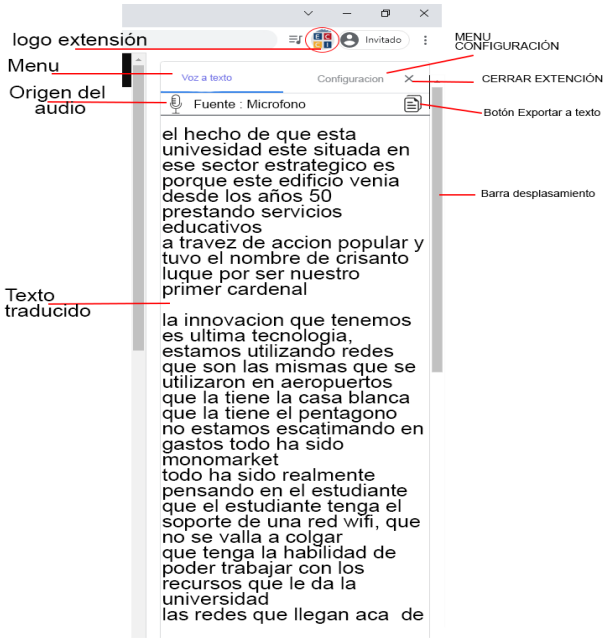


Figura 24 partes y elementos de la herramienta fuente: elaboración propia

Posteriormente después de recopilar y analizar la información, se decidió cambiar la infraestructura de una extensión de navegador hacia una aplicación web, esto con el fin de beneficiar a un grupo mayor de usuarios, pues una extensión de navegador Hubiera limitado la experiencia de usuarios, que utilicen un navegador diferente al cual la extensión estaba planeada, o incluso limitar la usabilidad de teléfonos móviles para esta herramienta.

Tras esta decisión, se decidió reacondicionar los mismos elementos del mockup anterior en una página web HTML con formato horizontal, resultando así en el mockup final basado en una aplicación web como se observa en la Figura 25Mockup de la

herramienta web fuente: elaboración propia.

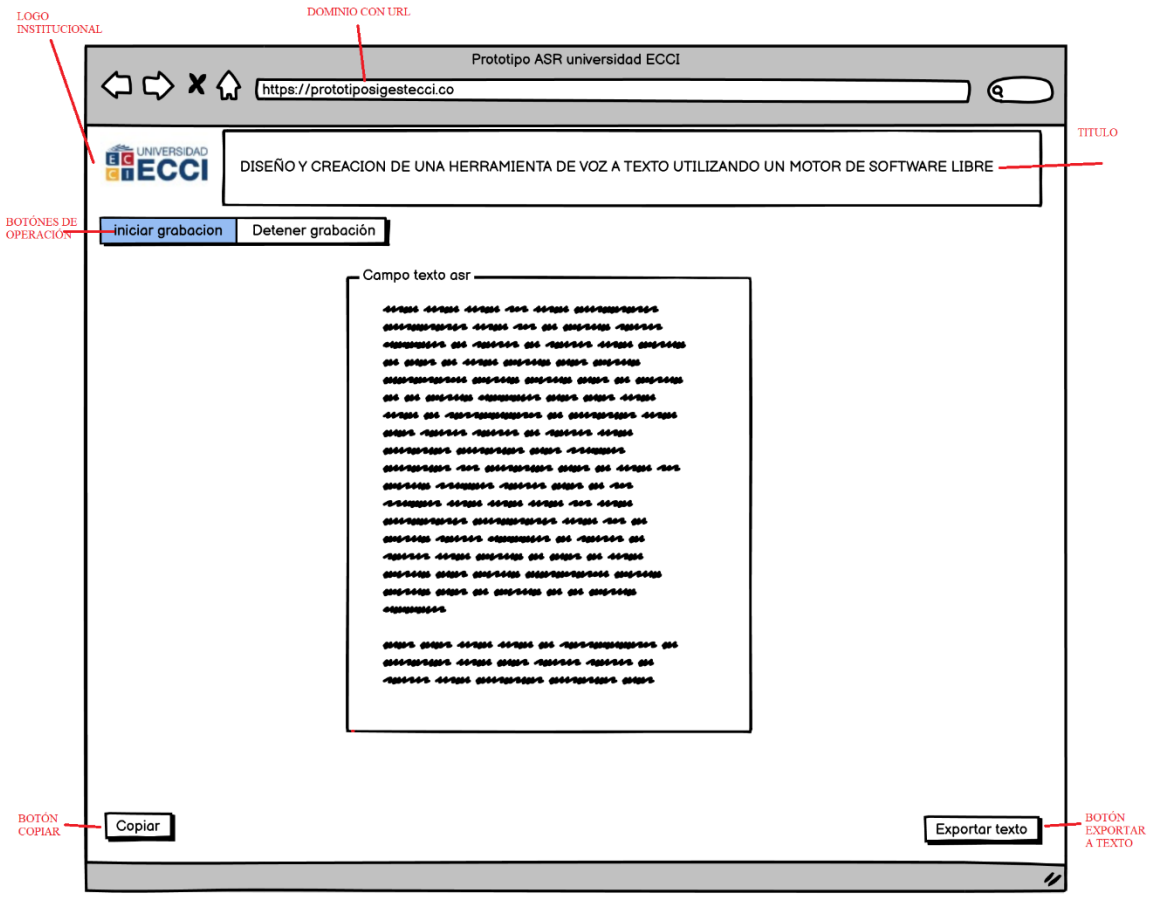


Figura 25Mockup de la herramienta web fuente: elaboración propia

Este es el mockup final que se tomó como diseño objetivo del prototipo. Posteriormente en la implementación se realizaron unos cambios mínimos en la ubicación del campo de texto desplazándolo a la izquierda, para que el texto transcrito pueda ser visto sin problemas desde dispositivos móviles.

6.8.1.4 Escalabilidad

Esta requiere mediciones y estadísticas de pruebas, estimando la utilización de recursos por parte de cada modelo propuesto y cada usuario participante, se puede

estimar que un modelo API-P2P, es mucho más fácilmente escalable que un modelo de ejecución de cliente. La optimización de ambos modelos requerirá de un correcto balance entre la infraestructura de hardware del mismo y la conexión (ancho de banda) dedicada para este fin, de lo contrario las constantes peticiones de transcripción, enviando y grabaciones de audio a una tasa muy elevada, puede afectar el rendimiento y la disponibilidad del mismo.

6.8.2 Infraestructura

Para el prototipo, se implementó un ambiente virtual de pruebas con hosting de Windows Azure. En el primer intento se contaba con limitaciones técnicas, que impedían ejecutar correctamente todas las dependencias del programa, se identificaron limitaciones de memoria, pues el servidor contaba con 1 procesador virtual y 2 GB de memoria RAM, la solución inmediata fue expandir al doble la capacidad, logrando que la herramienta funcionará de forma correcta para llevar a cabo todas las pruebas correspondientes, estos requerimientos de software, se tomarán como los requerimientos mínimos necesarios para instalar esta herramienta y posiblemente aumentarlos para escalarla, buscando una mejor funcionalidad. A continuación, se explicará todo el proceso de instalación y adecuación necesarios para utilizar esta herramienta.

6.8.3 Planimetría de Red

En el servidor de pruebas se realizó la experimentación utilizar NGIX como un “reverse Back proxy” intentando mejorar la conectividad hacia el servidor final ya que

Microsoft Azure lo ubico en ASIA por términos de costos, por medio de un VPN se determinó que el ping de la red incide de forma directamente proporcional a la funcionalidad de la conexión, pues la conexión Web Socket estaba teniendo bastantes, para mantener un flujo de Bit-Rate estable en una conexión desde América Latina. En la Figura 26 información de red servidor de pruebas fuente: elaboración propia, se puede apreciar la configuración de red del servidor de pruebas.

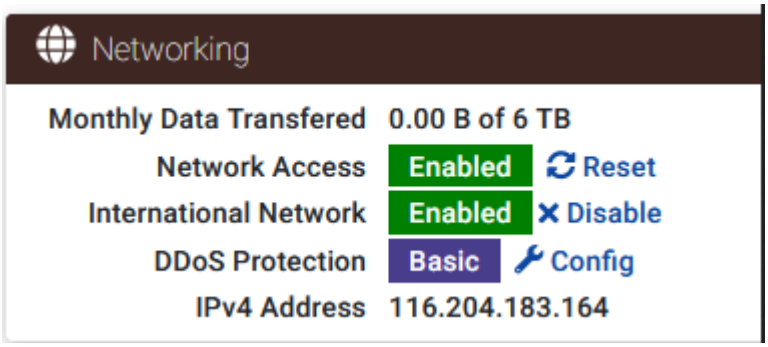


Figura 26 información de red servidor de pruebas fuente: elaboración propia

Posteriormente se migro el servidor a estados unidos para cambiar su locación por cercanía aumentando considerablemente la tasa de conexión y permitiendo el funcionamiento normal y en tiempo real de la plataforma, en la Figura 27 Configuración de firewalls fuente: elaboración propia, se puede observar la configuración de red del servidor de despliegue de Google Cloud.

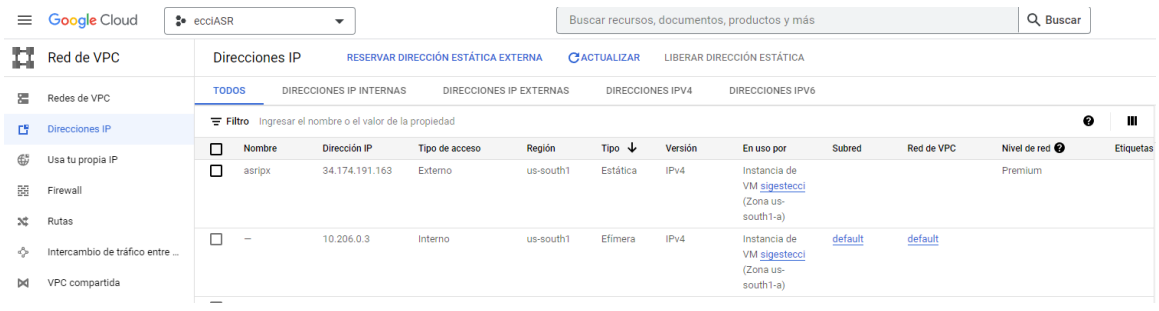


Figura 27 Configuración de firewalls fuente: elaboración propia

Falta agregar a la planimetría de red la configuración correcta del Firewall, para que los servicios de la herramienta se puedan ejecutar de forma correcta, en la Figura 28 conexiones de puertos fuente: elaboración propiase puede apreciar como el firewall permite las conexiones en los puertos 80,443,3389

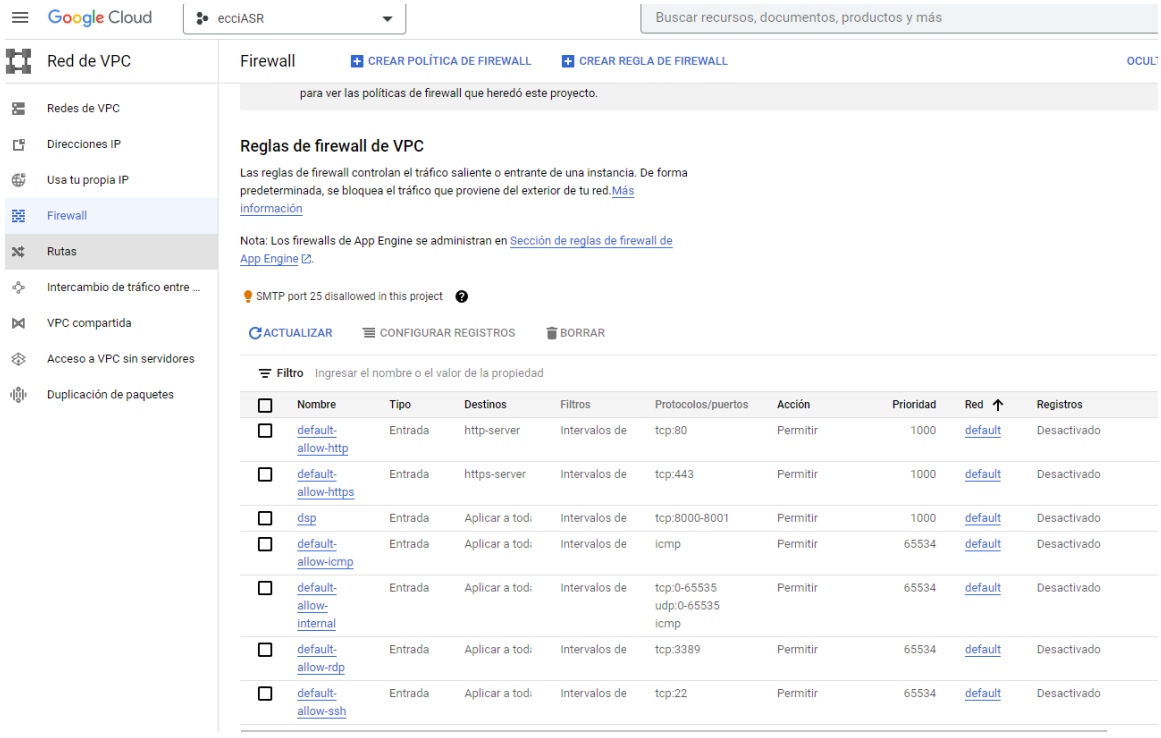


Figura 28 conexiones de puertos fuente: elaboración propia

Gracias a las reglas del firewall se puede utilizar un servidor intermedio entre el aplicativo y la aplicación web para gestionar la conexión web por el puerto 80, la conexión de React Iframes por el puerto 443 y la conexión encriptada de el Web Socket en el puerto 8000 que va cifrado por HTTPS, todo esto llega a un proxy reverso que conecta el puerto 3389 hacia la consola del motor de voz como se puede observar en el siguiente diagrama

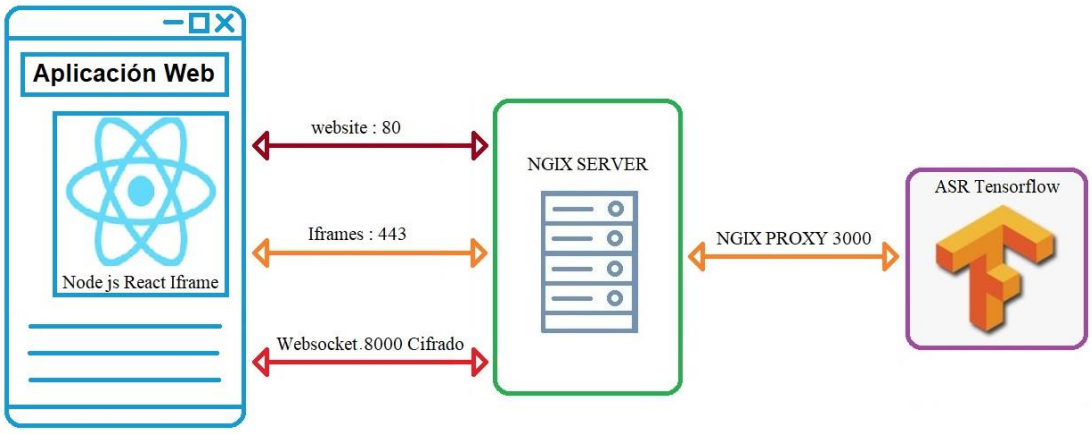


Figura 29 Diagrama de conexiones y puertos de la aplicación: elaboración propia.

Con esto se completa la planimetría de red, pues al contener todo el proyecto alojado en el mismo servidor, no requiere conexiones externas ni a servicios de terceros, aunque para una posible escalabilidad, se identificó la posibilidad de asignar varias instancias a núcleos Kubernets, para cumplir con la demanda de Ambos Modelos (modelo en off y modelo API p2p).

6.8.4 Descripción de servicio

Aquí se realiza una descripción completa y detallada del funcionamiento de la herramienta paso por paso y como cada componente interactúa entre sí para lograr el resultado final.

El servicio de transcripción a texto se presta por demanda, cuando un usuario ingresa la dirección URL del aplicativo web, este recibe la petición en el servidor NGIX y envía la interfaz web como página al usuario, para poder interactuar con el servicio cuando un usuario hace click en el botón Iniciar, se inicia una solicitud de conexión Web Socket con el servidor, el cual envía una solicitud automática al navegador para utilizar el

micrófono, una vez aceptada esta solicitud la conexión Web Socket se establece entre el navegador y el servidor final, donde está contenido el motor de transcripción de voz a texto, que recibe el audio directamente del usuario y envía el resultado transcrito mediante una conexión Bootstrap al contenedor REACT, de la página web con el texto traducido para el usuario.

El botón copiar, es un elemento fluido de REACT y llama la función "clipboard.writeText" asignándole al portapapeles de Windows el contenido completo del contenedor de texto, mientras que el botón exportar texto llama la función "Dump Console" y exporta el texto transcrito a un documento de texto txt. Después de las observaciones realizadas por el director de proyecto, se implementó una función para cambiar el orden en el cual el texto, fuera entregado al usuario para mejorar su usabilidad.

7. Solución propuesta

En esta sección se realiza una descripción general de la solución propuesta como herramienta, en la descripción de sus funcionalidades se explica su desarrollo, la instalación y configuración de la infraestructura a detalle para para su correcto funcionamiento.

7.1 Descripción de la propuesta

La solución propuesta es desarrollar un aplicativo web como prototipo utilizando solo Software Libre tanto en su infraestructura como en su funcionamiento esta herramienta transcribirá la voz convertida a audio en formato digital proveniente de un

micrófono y permitirá que su contenido sea manipulado por el usuario en acciones como copiar y exportar el texto resultante.

Esta solución es posible gracias a un motor de Software Libre: Mozilla Deep Voice que funciona con Redes Neuronales Convulsionadas utilizando la suite de Tensor Flow en la consola Python para identificar correctamente los sonidos pronunciados y transcribirlos a texto en tiempo real. En esta sección se describe el desarrollo realizado para implementar la propuesta utilizando librerías de Software Libre REACT en JavaScript para realizar una conexión Web Socket desde la interfaz del navegador hasta el motor de voz, además de documentar la instalación y la configuración de toda la infraestructura necesaria para su correcto funcionamiento.

7.2 Desarrollo de la propuesta

Lo primero en desarrollarse fue la página web, en esta página se agregó un banner y un título de la Universidad como aspectos gráficos del proyecto, posteriormente se agregan un IFrame de React y los botones de interacción, se realiza la programación del aplicativo en JavaScript, donde se construye la lógica de conexión entre la página web y la aplicación de la consola, por último, se cambia la lógica de el botón exportar texto para presentar el texto de forma acorde a la lectura.

7.3 Instalación de Software principal

En el Anexo 10 se incluye la documentación completa sobre la instalación configuración y ejecución del prototipo (Anexo 12) tanto en la fase de pruebas como en el servidor de despliegue, describe también los errores encontrados y como solucionarlos.

7.4 Pruebas Realizadas a la propuesta

A la propuesta se le realizaron 3 Pruebas, Pruebas de estabilidad de servicio, pruebas de conexión y pruebas de precisión transcripción. La estabilidad de servicio garantiza que la aplicación no se cierre o deje de funcionar durante su operación, mientras que las pruebas de conexión miden la velocidad y estabilidad del servidor en términos de milisegundos con uno o muchos usuarios para determinar su capacidad de servicio, estas pruebas se realizaron tanto en los entornos de pruebas locales como en el entorno de despliegue y sus mediciones se pueden ver a continuación.

7.4.1 Entorno de pruebas locales

El entorno de pruebas locales se desplego en un servidor de Microsoft Azure, en la Figura 30 Configuración Hardware Servidor fuente: elaboración propiase puede apreciar la configuración de hardware en este servidor,

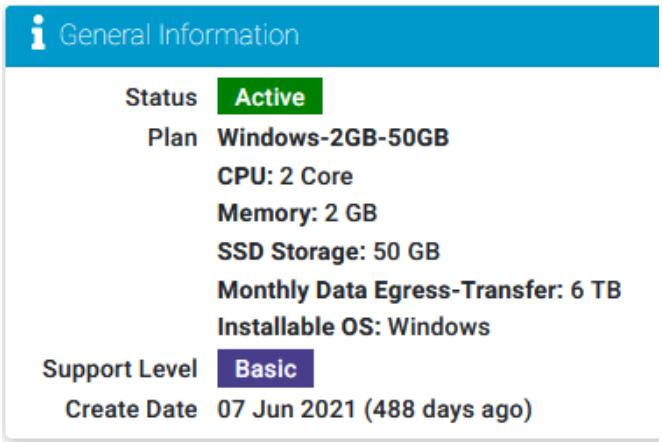


Figura 30 Configuración Hardware Servidor fuente: elaboración propia

El entorno de pruebas está configurado especialmente para medir la ejecución y reportar cualquier error que suceda durante la misma, esto con el fin de identificar problemas de estabilidad configuración o conexión antes de migrar a un servidor estable.

En este servidor de pruebas locales se desplegó la aplicación de forma local, sin conectarla al servidor de NGIX ni conexiones externas o interfaces, esto para verificar que todas las dependencias instaladas en el sistema operativo estuvieran en una versión correcta y operando sin errores. Contar con un servidor de pruebas es indispensable para diagnosticar y predecir el comportamiento de una aplicación durante su fase temprana de configuración y desarrollo, además le permitió al equipo de investigación documentar la instalación de la herramienta en diferentes infraestructuras de la nube y sistemas operativos.

7.4.1.1 Dificultades en el desarrollo

Durante el desarrollo se identificó la primera dificultad relacionada con la conexión Web Socket que conecta el micrófono del usuario con el motor de voz, en el primer intento de implementación de la aplicación en el servidor de prueba, el micrófono no estaba realizando una conexión, el motor de voz tampoco detectaba ninguna entrada, para diagnosticar el error se utilizaron diferentes computadores, navegadores y puntos de conexión de forma infructuosa sin cambios en el error generado, lo cual indicaba que el error estaba contenido en la lógica de la plataforma o en su configuración. Indagando en internet sobre este error, en la documentación de Web Socket en (Internet Engineering Task Force, 2011) se identificó que en las propiedades de la conexión, Web Socket necesita generar una autenticación y una llave privada dando indicios de una conexión

con un cifrado RSA, la necesidad de abrir el puerto SSL y TSL para un correcto empalme con el cliente, por lo cual se procedió a firmar un certificado de seguridad SSL auto firmado y volver a intentar su ejecución resultando en el funcionamiento parcial de la herramienta.

El segundo inconveniente se identificó durante la ejecución de la herramienta en el servidor de pruebas, junto con la presencia y compañía del docente director del proyecto se identificó una respuesta muy tardía e imprecisa por parte de la herramienta, entre un promedio de 20 a 60 segundos de retraso entre palabra pronunciada y transcripción, cabe aclarar que durante las pruebas iniciales en el entorno de desarrollo se utilizó el modelo de lenguaje para Ingles pues es el modelo por defecto que incluye la herramienta, tras probar diferentes configuraciones de red, se identificó que en conexiones cercanas a ASIA había menor demora en la respuesta del servidor, lo cual evidenciaba un problema de conexión tardía por lejanía geográfica ya que la transferencia de datos al servidor interrumpía el flujo de audio desde el micrófono, causando que la herramienta realizara transcripciones de forma errática, este problema fue solucionado posteriormente escogiendo un servidor más cercano geográficamente a nuestro país, solucionando así el problema.

7.4.2 Entorno de despliegue

Para el servidor de despliegue se escogió realizarlo en la plataforma de Google Cloud, la cual cuenta con servidores económicos y cercanos geográficamente, Google

ofrece créditos educativos para estudiantes y desarrolladores para que puedan implementar sus proyectos en su infraestructura a bajo costo, también se cambió su infraestructura a una basada en Software Libre, se escogió “Ubuntu Jammy 2204” con una arquitectura de 4 VCPU y 8GB de RAM para escalar la estabilidad de la plataforma y descartar fallas por hardware.

7.4.2.1 Proceso de migración de pruebas locales a entorno de despliegue

Este proceso comenzó cuando se pudo confirmar que la aplicación cumplía con su funcionalidad básica y en idioma español, ya que la instalación y configuración inicial viene por defecto en inglés, la migración de los archivos del montaje se realizó mediante un archivo comprimido de formato zip, con los archivos principales del motor de voz con un peso total de 1.35GB como se observa en la imagen y se envió al servidor de despliegue utilizando Google Drive por método HTTP.

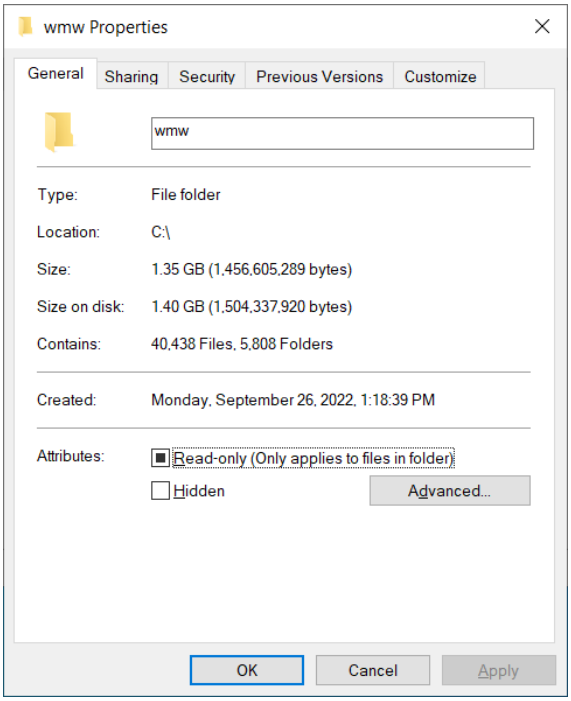


Figura 31 peso total de los archivos principales fuente: elaboración propia

La transferencia tardó 3 horas y 45 minutos, después de realizar la transferencia de los archivos principales, se realizó la configuración e instalación de todas las dependencias necesarias para correr, se reconfiguró el firewall desde el servidor final hacia el servidor NGINX y se establecieron los nuevos API en la página web, junto con los DNS y SSL conectando la interface con el servidor final.

7.4.3 Pruebas

Para verificar la estabilidad, conexión y precisión de la transcripción se le realizaron diferentes pruebas para documentar su comportamiento en el servidor de despliegue, las cuales podemos observar a continuación.

7.4.3.1 Pruebas de estabilidad

El servidor se encuentra encendido desde septiembre 26 del 2022 y hasta la fecha solo ha requerido 2 reinicios de servicio, aproximadamente uno al mes. Como se observa en la FIGURA 32 el último reinicio registrado se realizo el viernes 21 de octubre de 2022, con una estabilidad promedio de 24 días por reinicio.

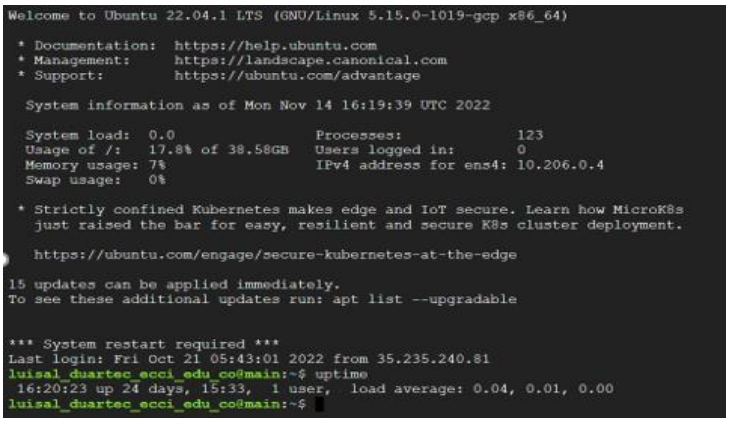


Figura 32 prueba de estabilidad fuente: elaboración propia

Este reinicio ocurre en el servidor de NGINX, pues el funcionamiento continuo hace que la memoria colapse en este plazo y el servidor se caiga, para evitar causar un colapso y un reinicio en el motor. Por este motivo se separó el servidor NGINX en una máquina virtual separada e independiente del motor de voz, para que este mismo cuente con sus propios recursos de hardware como procesador y memoria RAM, sin afectar el funcionamiento del programa de traducción simultánea.

7.4.3.2 Pruebas de conexión

La prueba Tracert nos indica que el cambio a la locación de miami es óptimo en conexión y nuestra señal desde Bogotá realiza 7 saltos para llegar al servidor de destino,

esto es esencial pues una mala conexión puede causar fallas en el Bitrate de audio y afectar la calidad del servicio en la traducción de voz.

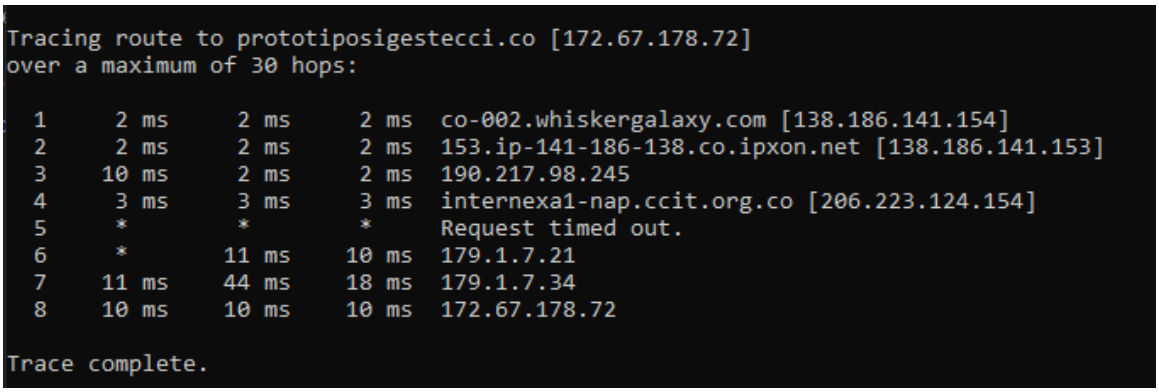


Figura 33 Tracert diagnostico conexión fuente: elaboración propia

Con un ping de 10 milisegundos como se puede apreciar en la imagen, la conexión de nuestro cliente hacia el servidor es muy estable y menor a 10ms, seguramente porque estamos conectados a un proveedor de internet corporativo.

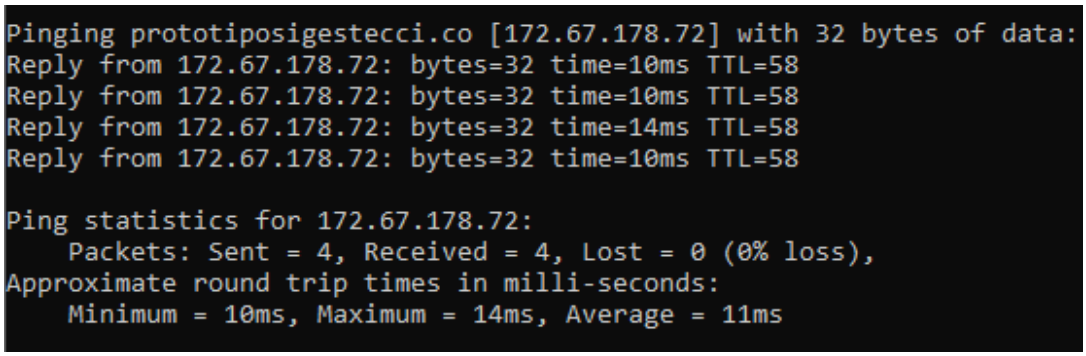


Figura 34 Ping hacia el servidor fuente: elaboración propia

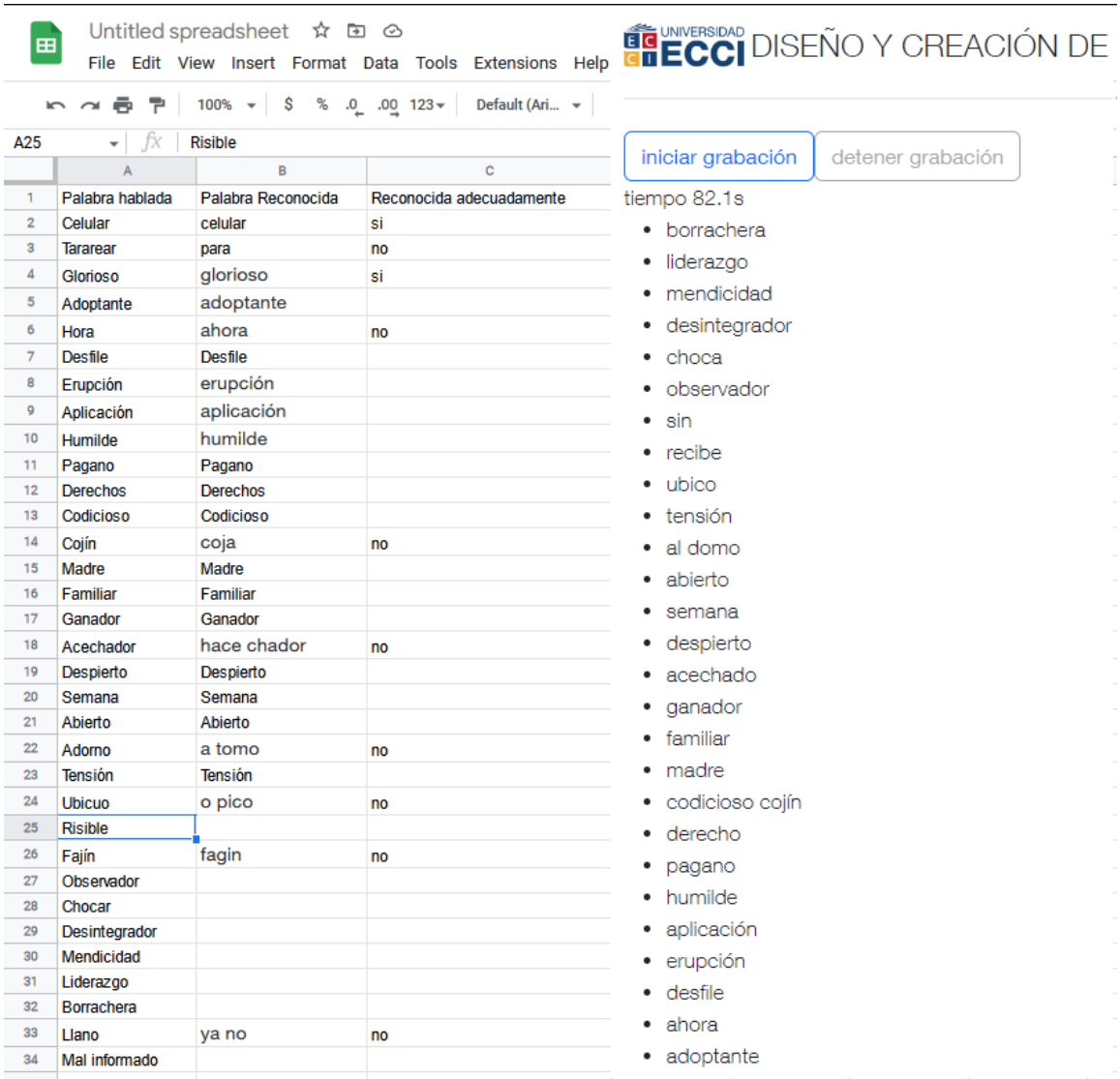
Mientras en pruebas externas a nuestra conexión habitual, tenemos un promedio de 50-60ms lo cual es bastante estable para la trasmisión de audio en vivo, las demoras y los retrasos en que afectan el servicio ocurren sobre los 150-250ms y a partir de los 350-500ms el servidor NGINX cierra la conexión automáticamente buscando una mejor calidad de conectividad. Esta también influye en el tiempo de respuesta del servidor, el

tiempo promedio de respuesta en un ambiente óptimo de conectividad es de 1 a 3 segundos siendo lo que tarda el servidor en mostrar un resultado, mientras que en ambientes de pésima conectividad como el servidor de pruebas ubicado en Asia el tiempo promedio de procesamiento era de 10 a 20 segundos.

7.4.3.3 Pruebas de precisión en la transcripción

El modelo de lenguaje utilizado es el modelo Polyglot CCLMTV de (Bermuth, 2021) tiene un entrenamiento de 660horas de grabación con una calificación de 0.165 WER (Word Error Rate) o Tasa de error de palabras. Calculando esta tasa de error el modelo de voz tiene un 83.5% de probabilidad de reconocimiento positivo en cada palabra pronunciada, esta calificación es congruente con las pruebas realizadas pues de manera manual se realizaron 3 pruebas cada una de 100 palabras aleatorias en un Excel,

como se puede apreciar en la figura 35.



	A	B	C
1	Palabra hablada	Palabra Reconocida	Reconocida adecuadamente
2	Celular	celular	si
3	Tararear	para	no
4	Glorioso	glorioso	si
5	Adoptante	adoptante	
6	Hora	ahora	no
7	Desfile	Desfile	
8	Erupción	erupción	
9	Aplicación	aplicación	
10	Humilde	humilde	
11	Pagano	Pagano	
12	Derechos	Derechos	
13	Codicioso	Codicioso	
14	Cojín	coja	no
15	Madre	Madre	
16	Familiar	Familiar	
17	Ganador	Ganador	
18	Acechador	hace chador	no
19	Despierto	Despierto	
20	Semana	Semana	
21	Abierto	Abierto	
22	Adorno	a tomo	no
23	Tensión	Tensión	
24	Ubicuo	o pico	no
25	Risible		
26	Fajín	fagin	no
27	Observador		
28	Chocar		
29	Desintegrador		
30	Mendicidad		
31	Liderazgo		
32	Borrachera		
33	Llano	ya no	no
34	Mal informado		

Figura 35 Prueba de error en palabras fuente: elaboración propia

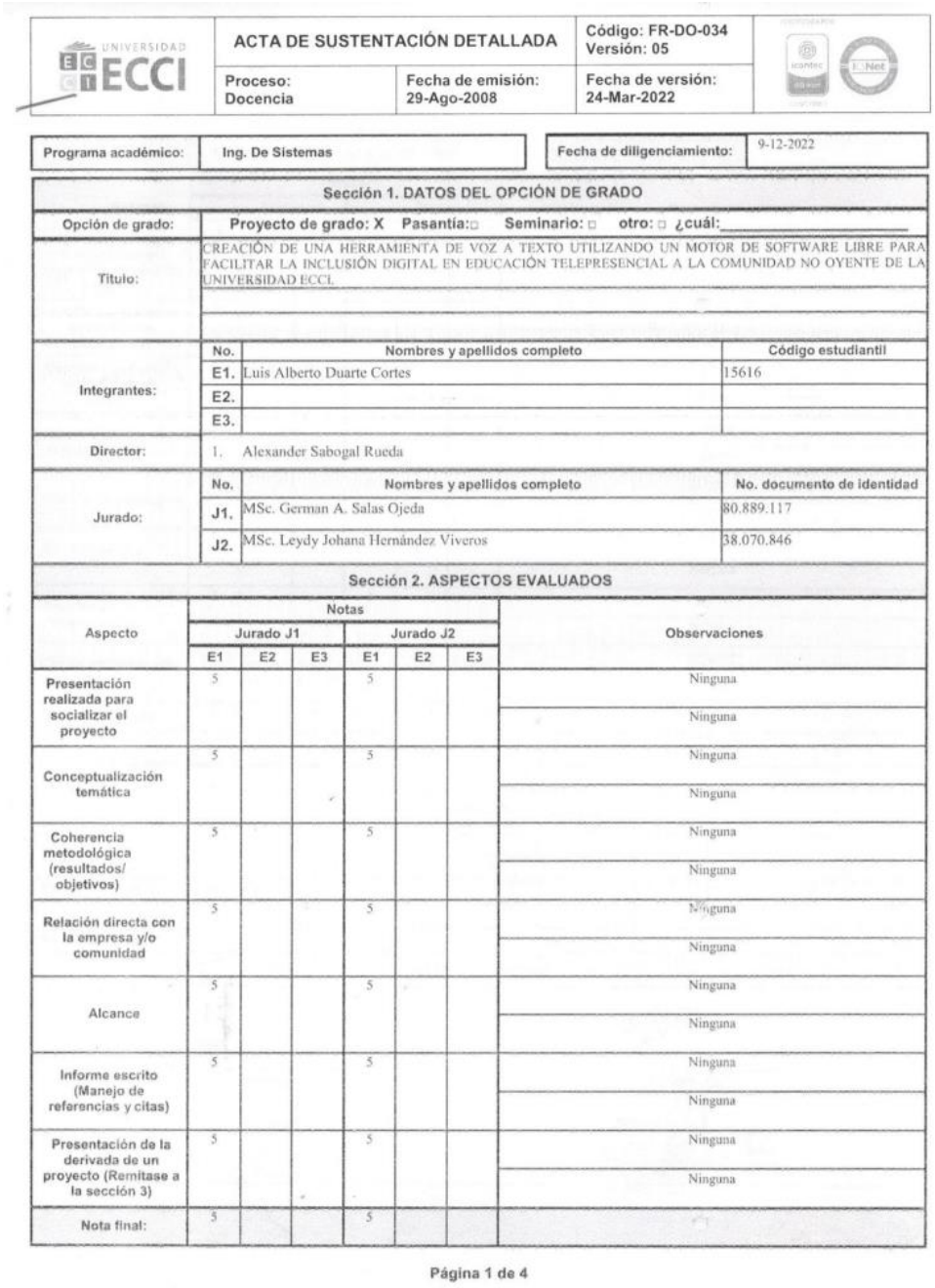
Estas pruebas arrojaron un resultado de acierto del 80% 88% y 90% de aciertos, con un promedio de 86% muy cercano a la tasa de errores descrita por el autor. Cabe aclarar que las pruebas realizadas para alcanzar la tasa de errores oficial, toman alrededor de 25 horas en realizarse. Las pruebas se indexan a este documento como Anexo 11 donde cada libro de Excel es una prueba diferente.

7.5 Análisis de Resultados Obtenidos

Los resultados demuestran que el cambio entre el entorno de pruebas locales y el servidor de despliegue, fue decisivo para mejorar la conexión y funcionamiento, a su vez que separando el servidor NGINX del motor de procesamiento de voz a texto, distribuyendo y escalando los recursos de hardware buscando mayor independencia autonomía y eficiencia, se pudo evidenciar que el modelo de lenguaje utilizado es congruente con su clasificación de tasa de errores en las pruebas publicadas, sin embargo el español utilizado en los data sets está completamente mesclado entre acentos regionales y geográficos, lo cual disminuye la efectividad de reconocimiento para nuestro acento colombiano, lo cual resulta en una propuesta de mejora para adaptar el modelo de lenguaje a nuestro acento o específicamente a un data set creado por las clases grabadas de la misma Universidad.

7.6 Acta Cierre del Proyecto

La siguiente figura muestra el acta de cierre y sustentación detallada del proyecto del Diseño y creación de una herramienta de voz a texto, utilizando un motor de Software Libre para facilitar la inclusión digital en educación tele presencial a la comunidad no oyente de la Universidad ECCI.



UNIVERSIDAD ECCI	ACTA DE SUSTENTACIÓN DETALLADA		Código: FR-DO-034 Versión: 05
	Proceso: Docencia	Fecha de emisión: 29-Ago-2008	Fecha de versión: 24-Mar-2022

Programa académico:	Ing. De Sistemas	Fecha de diligenciamiento:	9-12-2022

Sección 1. DATOS DEL OPCIÓN DE GRADO

Opción de grado:	Proyecto de grado: X Pasantía:□ Seminario: □ otro: □ ¿cuál:_____		
Título:	CREACIÓN DE UNA HERRAMIENTA DE VOZ A TEXTO UTILIZANDO UN MOTOR DE SOFTWARE LIBRE PARA FACILITAR LA INCLUSIÓN DIGITAL EN EDUCACIÓN TELEPRESENCIAL A LA COMUNIDAD NO OYENTE DE LA UNIVERSIDAD ECCI.		
Integrantes:	No.	Nombres y apellidos completo	Código estudiantil
	E1.	Luis Alberto Duarte Cortes	15616
	E2.		
	E3.		
Director:	1. Alexander Sabogal Rueda		
Jurado:	No.	Nombres y apellidos completo	No. documento de identidad
	J1.	MSc. German A. Salas Ojeda	80.889.117
	J2.	MSc. Leydy Johana Hernández Viveros	38.070.846

Sección 2. ASPECTOS EVALUADOS

Aspecto	Notas						Observaciones
	Jurado J1			Jurado J2			
	E1	E2	E3	E1	E2	E3	
Presentación realizada para socializar el proyecto	5			5			Ninguna Ninguna
Conceptualización temática	5			5			Ninguna Ninguna
Coherencia metodológica (resultados/ objetivos)	5			5			Ninguna Ninguna
Relación directa con la empresa y/o comunidad	5			5			Ninguna Ninguna
Alcance	5			5			Ninguna Ninguna
Informe escrito (Manejo de referencias y citas)	5			5			Ninguna Ninguna
Presentación de la derivada de un proyecto (Remítase a la sección 3)	5			5			Ninguna Ninguna
Nota final:	5			5			

Página 1 de 4

Figura 36 acta de cierre 1 Fuente: Coordinación sistemas ECCI

UNIVERSIDAD ECCI	ACTA DE SUSTENTACIÓN DETALLADA		Código: FR-DO-034 Versión: 05
	Proceso: Docencia	Fecha de emisión: 29-Ago-2008	Fecha de versión: 24-Mar-2022

Sección 3. EVALUACIÓN DE PROTOTIPO Y/O SERVICIO (Aplica SI:□ NO:□)

Aspecto	Notas						Observaciones
	Jurado J1			Jurado J2			
	E1	E2	E3	E1	E2	E3	
Formulación del problema y/o idea de negocio	5			5			Ninguna Ninguna
Justificación para el desarrollo del prototipo y/o servicio	5			5			Ninguna Ninguna
Objetivo que cumple el prototipo y/o servicio	5			5			Ninguna Ninguna
Marco teórico y estado del arte	5			5			Ninguna Ninguna
Diseño metodológico	5			5			Ninguna Ninguna
Conclusiones y resultados	5			5			Ninguna Ninguna
Propuestas de mejoramiento	5			5			Ninguna Ninguna
Valor agregado del prototipo y/o servicio	5			5			Ninguna Ninguna
Nota de prototipo:	5			5			

Sección 4. CALIFICACIÓN

Integrantes	Ítem		Notas	
	Desarrollo opción de grado (Asesor)	Sustentación (Jurado)	Número	Letra
	___%	___%		
E1. LUIS ALBERTO DUARTE CORTES		5,0	5,0	Cinco
E2.				
E3.				

Sección 5. FIRMAR

Jurado J 1

Jurado J2

Asesor

Coordinador

Página 2 de 4

Figura 37acta de cierre 2 Fuente: Coordinación sistemas ECCI

UNIVERSIDAD ECCI	FORMATO ACTA DE OPCIÓN DE GRADO		Código: FR-DO-033 Versión: 03
	Proceso: Docencia	Fecha de emisión: 29-Ago-2008	Fecha de versión: 28-Oct-2010

ACTA DE OPCIÓN DE GRADO

INGENIERÍA DE SISTEMAS

Se notifica que el estudiante LUIS ALBERTO DUARTE CORTES, identificado con código estudiantil No. 15616, realizo como opción de grado el *PROYECTO DE GRADO*, **titulado(a):** *"CREACIÓN DE UNA HERRAMIENTA DE VOZ A TEXTO UTILIZANDO UN MOTOR DE SOFTWARE LIBRE PARA FACILITAR LA INCLUSIÓN DIGITAL EN EDUCACIÓN TELEPRESENCIAL A LA COMUNIDAD NO OYENTE DE LA UNIVERSIDAD ECCI"*, obteniendo una calificación de *Cinco Punto Cero* **(5.0).**

Como asesor(es) le hicieron acompañamiento los docentes: Alexander Sabogal Rueda, y como Jurado(s): German Salas O. y Leydy Hernandez V.

Lo anterior se expide en Bogotá D.C., a los *nueve (9)* días del mes de *diciembre* de 2022.

Jurado

Jurado

UNIVERSIDAD ECCI
DIRECCIÓN

Director

Coordinador

Página 1 de 1

Figura 38 Acta de opción de grado Fuente: coordinación sistemas ECCI

8. Recursos

A continuación, se listan los recursos necesarios para llevar a cabo el desarrollo del proyecto.

8.1 Recursos Humanos

8.1.1 Líder de proyecto

Persona encargada de la elaboración del proyecto de investigación con ayuda de los asesores.

8.1.2 Director de proyecto (Universidad ECCI)

Docente de la Universidad ECCI encargado de supervisar el proyecto y su respectiva investigación y aportando ideas de mejoramiento para el mismo.

8.1.3 Asesor de proyecto (Universidad ECCI)

Docente que supervisa el documento del proyecto de grado validando que este cumpla con las normas apa sexta edición.

8.2 Recursos Físicos

- 1 computador portátil
- 1 computador de escritorio
- 1 escritorios para computador
- 1 sillas
- 1 modem

8.3 Recursos Tecnológicos

8.3.1 Servidor de pruebas

Servidor Microsoft Azure Standard_A2_v2 con (3rd Generation Intel® Xeon® Platinum 8370C) 2VCPU 2GB RAM y 50GB SSD

8.3.2 Servidor de despliegue

Custom E2 4 vCPU 8GBRAM Intel Skylake con 8GB SSD y un Custom E2 2 vCPU 4GB Intel Broadwell de 8GB SSD para el servidor NGINX

9. Cronograma de Actividades

En este capítulo se va a representar el cronograma de actividades, donde están plasmadas las actividades que se realizaron para la investigación del proyecto y las fechas en las que se realizaron:

No.	Actividades	Año 2021 / Mes								Año 2022 / Mes											
		5	6	7	8	9	10	11	12	1	2	3	4	5	6	7	8	9	10	11	12
1	Investigación y diseño metodológico																				
2	Solicitud de información estadística preliminar																				
3	Planteamiento Metodológico Desarrollo de problema, objetivos y justificiación																				
4	Desarrollo Marco Teórico																				
5	Estado del Arte, Motores de ASR																				
6	Diseño y desarrollo de infraestructura del prototipo																				
7	Configuración y desarrollo preliminar del prototipo																				
8	Desarrollo de Página Web																				
9	Montaje y prueba preliminar de servicio en ambiente de desarrollo																				
10	Configuración Servidor de despliegue																				
11	Transferencia de ambiente de pruebas a servidor de despliegue																				
12	Montaje y puesta en marcha del prototipo en servidor de despliegue																				
13	Documentación de la plataforma																				
14	Pruebas de precisión y estabilidad de la plataforma																				

Figura 39 Cronograma de actividades fuente: elaboración propia

10. Conclusiones

Se puede concluir que:

Sí es posible construir un prototipo de voz a texto, utilizando Software Libre en toda su infraestructura, que asista a la comunidad sorda utilice subtítulos de apoyo en español y sea funcional en cualquier navegador web manteniendo un modelo modular, que permita adaptar o separar dependencias de esta herramienta facilitando su adaptación a diferentes modelos e infraestructuras.

La Metodología implementada en el proyecto puede ser utilizada como metodología de desarrollo de software, enfocada en investigar e implementar soluciones a problemas que afecten a la comunidad del investigador.

La transcripción de voz a texto, solo es un pequeño espectro de todo el estudio multidisciplinario que involucra la tecnología de reconocimiento de voz.

La identificación de problemas, evidencia que el servicio de reconocimiento de voz a texto tiene una demanda real por parte de la comunidad sorda y una ausencia plena en charlas conversatorios y actividades extracurriculares, donde los traductores de señas no están presentes.

La investigación evidencio que existen diferentes plataformas de Software Libre con diferentes métodos y arquitecturas de transcripción de voz a texto, al igual que multitud de librerías y data sets de código abierto, para generar y entrenar modelos de voz propios.

El estado del arte concluye que, a diferencia de las alternativas privadas, las plataformas de Software Libre requieren una investigación, programación y

configuración necesaria para operar de forma sustentable, ya que estas no se entregan “listas para usar” pero si “listas para implementar”.

Este modelo es escalable y autosustentable, en una implementación propia que la Universidad requiera puede adaptar este modelo a su propia arquitectura de hardware o a su hosting de preferencia.

Se evidenció que es posible adaptar esta aplicación a plataformas fuera de línea y portátiles, que funcionen en sistemas embebidos y no requieran conexión a internet.

Las dificultades presentadas en la instalación y configuración del prototipo, están directamente relacionadas con una mala instalación de las dependencias o una errónea configuración de puertos y conexiones de red.

Se puede mejorar la taza de errores de transcripción creando un dataset con base en las clases tele presenciales dictadas con anterioridad, para entrenar un nuevo modelo de lenguaje adaptado a medida al acento propio de los estudiantes y docentes de la Universidad ECCI.

Al separar el servidor NGINX y el motor de transcripción de voz a texto, ambas dependencias mejoran su desempeño y rendimiento ya que los recursos de hardware pasan de ser exclusivos a dedicados cada uno en diferentes instancias, sin afectar el desempeño o la conexión a la herramienta.

La infraestructura de hardware es directamente proporcional a la calidad de la prestación del servicio.

La Universidad puede implementar este prototipo y adaptarlo a sus servicios de educación virtual y tele presencial, sin afectar ni invertir en herramientas y soluciones de privados o terceros que ofrezcan este servicio.

11. Bibliografía

Alpha Cephei. (15 de Marzo de 2022). *Vosk*. Obtenido de https://alphacephei.com/vosk/

Amazon. (15 de Agosto de 2022). *ASR AWS*. Obtenido de Pricing: https://aws.amazon.com/transcribe/pricing/?nc=sn&loc=3

appareo. (29 de abril de 2021). *AVIATION SPEECH RECOGNITION SYSTEM*. Obtenido de Aviation Speech Recognition System Using Artificial Intelligence: https://appareo.com/aviation/aviation-speech-recognition-system/

Beazley, D. M. (2009). *Python Essential Reference.* Addison-Wesley Professional.

Bermuth, D. a. (2021). Scribosermo: Fast Speech-to-Text models for German and other Languages. *arXiv preprint arXiv:2110.07982*.

Bloch, J. (2018). A Brief, Opinionated History of the API. *QCon* (pág. 1). San Francisco: Enterprise Software Development Community. Obtenido de https://www.infoq.com/presentations/history-api/

Bolaños Araya, C., Camacho Lozano, A., & Urrutia, X. d. (2017). USO DE LA ENTONACIÓN PARA IDENTIFICAR CUÁNDO USAR LA TILDE DIACRÍTICA EN EL RECONOCIMIENTO AUTOMÁTICO DEL HABLA. *Káñina*, 40(4), 13. Obtenido de https://doi.org/10.15517/rk.v40i4.30222

Botti, V., & Serra, J. M. (2001). *Aplicación de una red neuronal para la predicción de la reacción catalítica isomerización del n-Octano.* Valencia: Universitat Politècnica de València.

Casado-Mancebo, M. (2021). Una aproximación a la lingüística computacional. *Revista de Filosofía, Letras y Humanidades,*, 746–761.

Casanova, E., Gölge, E., Meyer, J., davis, k., & Morais, R. (29 de septiembre de 2022). *Coqui*. Obtenido de Make the impossible possible and the painful painless with Coqui: https://coqui.ai/

Celis Nuñez, J. D. (2017). Modelo Acústico y de Lenguaje del Idioma Español para el dialecto Cucuteño, Orientado al Reconocimiento Automático del Habla. *Ingeniería*, 22(3), 362.

Churbanov, A., & Winters-Hilt, S. (2008). Implementing EM and Viterbi algorithms for Hidden Markov Model in linear memory. *BMC Bioinformatics 9*, 224.

Colompar, B. C. (2018). *Desarrollo de un sistema de Reconocimiento Automático del Habla en Rumano para el subtitulado de vídeos educativos.* Valencia: Escola Tècnica Superior d'Enginyeria Informàtica Universitat Politècnica de Valencia.

Cuomo, J. (2013). *Mobile app development, JavaScript.* IBM Software.

de Luna, E. B., & Expósito López, J. (2011). UNIDAD 3. EL PROCESO DE INVESTIGACIÓN EDUCATIVA II: INVESTIGACIÓN-ACCIÓN. *FACULTAD DE CIENCIAS DE LA EDUCACIÓN - UNIVERSIDAD DE GRANADA*, 35-50.

Deng, L., & Yu, D. (2014). Deep Learning: Methods and Applications. En M. d. Jongh, *Foundations and Trends R© in Signal Processing Vol. 7* (págs. 197-387).

Doshi, K. (25 de Mar de 2021). *Audio Deep Learning Made Simple: Automatic Speech Recognition (ASR), How it Works*. Obtenido de Speech-to-Text algorithm and architecture, including Mel Spectrograms, MFCCs, CTC Loss and Decoder, in Plain English: https://towardsdatascience.com/audio-deep-learning-made-simple-automatic-speech-recognition-asr-how-it-works-716cfce4c706

Emilio, M. D. (2015). *Embedded Systems Design for High-Speed Data Acquisition and Contro.* Springer.

Enciclopedia de Ejemplos. (29 de Septiembre de 2022). *Tipos de acentos*. Obtenido de Cuáles son los Tipos de acentos: https://www.ejemplos.co/tipos-de-acentos/

Ferrucci, D., Levas, A., Bagchi, S., Gondek, D., & Mueller, E. T. (2013). Watson: Beyond Jeopardy! *Science Direct*, 93-105.

Field, C. (31 de Agosto de 2021). *Towards Data Science*. Obtenido de Hidden Markov Models: an Overview: https://towardsdatascience.com/hidden-markov-models-an-overview-98926404da0e

Gemmeke, J. F., Ellis, D., Freedman, D., Jansen, A., Lawrence, W., Moore, R. C., . . . Ritter, M. (2017). Audio Set: An ontology and human-labeled dataset for audio events. *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (págs. 776 - 780). New Orleans: IEEE.

Google. (15 de 07 de 2022). *Google Kubernetes Engine*. Obtenido de GKE: https://Cloud.Google.com/kubernetes-engine

Google Cloud. (3 de Mayo de 2022). *Pricing | Cloud speech-to-text | Google Cloud*. Obtenido de https://Cloud.Google.com/speech-to-text/pricing

Google Cloud. (26 de abril de 2022). *Speech-to-Text: Automatic Speech Recognition*. Obtenido de https://Cloud.Google.com/speech-to-text

Hannun, A., Case, C., Caspe, J., Catanzaro, B., Diamos, G., Elsen, E., . . . Ng, A. Y. (2014). Deep Speech: Scaling up end-to-end speech recognition. *Baidu Research Silicon Valley AI Lab*, 12. Obtenido de arXiv:1412.5567.

Hashemnia, S. &. (2021). Human EEG and Recurrent Neural Networks Exhibit Common Temporal Dynamics During Speech Recognition. *Frontiers in Systems Neuroscience*, 617605.

Haubold, A., & Kender, J. (2007). Alignment of Speech to Highly Imperfect Text Transcriptions. *2007 IEEE International Conference on Multimedia and Expo* (págs. 224 - 227). Beijing: IEEE.

Herzog, O. (2005). Applied Wearable Computing, IFAWC. *2nd International Forum on Applied Wearable Computing, IFAWC: Proceedings, March 17-18, 2005 in Zurich, Switzerland* (pág. 188). Zurich, Switzerland: VDE Verlag.

IBM. (26 de Abril de 2022). *Watson Speech to Text*. Obtenido de Convert speech into text using AI-powered speech recognition and transcription: https://www.ibm.com/Cloud/watson-speech-to-text

Internet Engineering Task Force. (9 de 12 de 2011). *The WebSocket Protocol.* Obtenido de Internet Engineering Task Force: https://datatracker.ietf.org/doc/html/rfc6455

Lecorvé, G. (25 de Jul de 2022). *Automatic speech recognition.* Obtenido de Vocal and Acoustic Interactions - Automatic Speech Recognition : http://people.irisa.fr/Gwenole.Lecorve/lectures/ASR.pdf

Lee, A., Kawahara, T., & Shikano, K. (2001). Julius — an Open Source Real-Time Large Vocabulary Recognition Engine. *7th European Conference on Speech Communication and Technology* (págs. 1-4). Scandinavia: INTERSPEECH.

Li, X., Sun, J., Lei, X., Zou, W., & Zhao, S. (22 de septiembre de 2022). *Athena*. Obtenido de What is Athena?: https://athena-team.readthedocs.io/en/latest/introduction/introduction.html

Lunden, I. (24 de Enero de 2013). *techcrunch*. Obtenido de Amazon Gets Into Voice Recognition: https://techcrunch.com/2013/01/24/amazon-gets-into-voice-recognition-buys-ivona-software-to-compete-against-apples-siri/

Mahmood, A., & Köse, U. (15 de Enero de 2021). Speech recognition based on Convolutional neural networks and MFCC algorithm. *Advances in Artificial Intelligence Research (AAIR)*, 6-12.

Mateus, E. O. (2008). *HIDDEN MAKROV MODELS (HMM'S) Y APLICACIONES.* Cartagena de indias D.T y C: Universidad Tecnologica de Bolivar.

Ming, Z., Nan, D., Shujie, L., & Heung-Yeung, S. (Marzo de 2020). Progress in Neural NLP: Modeling, Learning, and Reasoning. (M. R. Asia, Ed.) *Engineering Volume 6, Issue 3*, 275-290.

Ministerio de Educación. (23 de Marzo de 2020). *Decreto 457 mediante el cual se imparten instrucciones para el cumplimiento del Aislamiento Preventivo Obligatorio.* Obtenido de mineducacion.gov.co: https://www.mineducacion.gov.co/1759/w3-printer-394357.html

Mohamed, A. r. (2014). *Deep Neural Network acoustic models for ASR.* Toronto: Department of Computer Science University of Toronto.

Mozilla. (15 de 08 de 2022). *Discourse Mozilla*. Obtenido de Deep Speech forum: https://discourse.mozilla.org/c/deepspeech/247

Mozilla. (10 de febrero de 2022). *Mozilla Voice*. Obtenido de How we're making Common Voice even more linguistically inclusive: https://foundation.mozilla.org/en/blog/how-we-are-making-common-voice-even-more-linguistically-inclusive/

Mozilla Corporation. (21 de Septiembre de 2022). *Mozilla Common Voice*. Obtenido de Common Voice: https://commonvoice.mozilla.org/es/criteria

Naik, S., Naik, N., Prabhu, G., Bhayje, A., Naik, V. P., & Aswale, S. (9 de junio de 2021). A Survey on different approaches for Speech to Text and Text to Speech in Email System for Visually Impaired People. *International Journal of Computer Applications (volume 183 – No. 9)*, 20-23.

Ortega, S. V. (1999). Sobre las relaciones de la morfologia con la sintaxis. *Revista Española de Lingüistica*, 257-2781.

Povey, D. (10 de Mayo de 2022). *Kaldi*. Obtenido de https://kaldi-asr.org/doc/

Pratap, V., Hannun, A., Xu, Q., Cai, J., Kahn, J., Synnaeve, G., . . . Collobert, R. (2019). wav2letter++: The Fastest Open-source Speech Recognition System. *ICASSP 2019* (págs. 6460-6464). Brighton, UK : IEEE.

Reactjs. (15 de Septiembre de 2022). *Refs and the DOM.* Obtenido de Refs provide a way to access DOM nodes or React elements created in the render method.: https://reactjs.org/docs/refs-and-the-dom.html

Reyzábal Manso, M. I. (2005). *Modelos de lenguaje y tecnología del habla.* Recuperado el 15 de Septiembre de 2022, de Educación XX1 2005, 8 ( ): https://www.redalyc.org/articulo.oa?id=70600806

Sadeen , A., Muna, A., Alanoud, A., Turkiayh, A., Raghad, A., Rimah, A., . . . Maha, A. (14 de september de 2021). Automatic Speech Recognition: Systematic Literature Review. *IEEE Access*, 131858 - 131876.

Sean, W. (29 de noviembre de 2017). *Mozilla Press Center*. Obtenido de Announcing the Initial Release of Mozilla's Open Source Speech Recognition Model and Voice Dataset: https://blog.mozilla.org/press/2017/11/announcing-the-initial-release-of-mozillas-open-source-speech-recognition-model-and-voice-dataset/

Shivangi, N., & Ashika, J. (2020). A REVIEW ON METHODS FOR SPEECH-TO-TEXT AND TEXT-TO-SPEECH. *International Research Journal of Engineering and Technology*, 6.

Significados. (16 de Enero de 2020). Obtenido de https://www.significados.com/marco-conceptual/

Smith, D. R. (2003). *Digital Transmission Systems.* Springer.

The International Phonetic Association. (25 de Septiembre de 2022). *Reproduction of The International Phonetic Alphabet*. Obtenido de

https://web.archive.org/web/20121010121927/http://www.langsci.ucl.ac.uk/ipa/ipachart.html

The World Wide Web Consortium (W3C). (15 de 05 de 2006). *Understanding the New Language Tags*. Obtenido de WC3 Internationalization: https://www.w3.org/International/articles/bcp47/

Trivedi, A., Pant, N., Shah, P., & Sonik, S. (2018). Speech to text and text to speech recognition systems-Areview. En N. Pant, *IOSR Journal of Computer Engineering (IOSR-JCE) Volume 20, Issue 2, Ver. I* (págs. 38-39). Mumbai: NMIMS University.

Trmal, J. ". (8 de mayo de 2022). *openslr*. Obtenido de About OpenSLR: https://www.openslr.org/

Vivek, B., Sashi, B., Virender, K., & Vinay, K. (2020). Development of Robust Automatic Speech Recognition System for Children's using Kaldi Toolkit. *2020 Second International Conference on Inventive Research in Computing Applications (ICIRCA)* (págs. 10-13). Coimbatore,India: IEEE.

Vivek, C. V. (18 de Agosto de 2020). *Markov and Hidden Markov Model*. Obtenido de Elaborated with examples: https://towardsdatascience.com/markov-and-hidden-markov-model-3eec42298d75

Vu, T. N. (2014). *Automatic Speech Recognition for Low-resource Languages and Accents Using Multilingual and Crosslingual Information.* Karlsruhe Germany: Karlsruhe Institute of Technology KIT.

Wikipedia. (18 de mayo de 2022). *Wikcionario* . Obtenido de https://es.wikipedia.org/wiki/Wikcionario

Yalta, N., Hayashi, T., & Yalta, N. (10 de septiembre de 2022). *ESPnet:* . Obtenido de end-to-end speech processing toolkit: https://github.com/espnet/espnet