APRENDIZAJE AUTOMÁTICO DE LENGUA DE SEÑAS COLOMBIANA

CIS2210CP03

Cristian Javier Da Cámara Sousa

Kenneth David Leonel Triana

Juan Pablo Ortiz Rubio

Camilo Andrés Sandoval Guayambuco

Noviembre de 2022

CIS2210CP03

Aprendizaje Automático de Lengua de Señas Colombiana

Autores:

Cristian Javier Da Cámara Sousa
Kenneth David Leonel Triana
Juan Pablo Ortiz Rubio
Camilo Andrés Sandoval Guayambuco

MEMORIA DE PROYECTO DE GRADO REALIZADO PARA CUMPLIR UNO DE LOS REQUISITOS DE LA CARRERA DE INGENIERÍA EN SISTEMAS

Director

Ing. Andrea Del Pilar Rueda Olarte

Jurados del Trabajo Final de Grado

Ing. Cécile Gauthier Umaña Ph.D

Ing. Efraín Ortiz Pabón Ph.D

PONTIFICIA UNIVERSIDAD JAVERIANA
FACULTAD DE INGENIERÍA
PROGRAMA DE INGENIERÍA DE SISTEMAS
BOGOTÁ, D.C.
Noviembre,2022

PONTIFICIA UNIVERSIDAD JAVERIANA
FACULTAD DE INGENIERÍA
PROGRAMA DE INGENIERÍA DE SISTEMAS

Rector de la Pontificia Universidad Javeriana

Jorge Humberto Peláez Piedrahita, S.J.

Decano de la Facultad de Ingeniería

Ing. Lope Hugo Barrero Solano

Director del Programa de Ingeniería de Sistemas

Ing. Carlos Andrés Parra Acevedo Ph.D

Director del Departamento de Ingeniería de Sistemas

Ing. Cesar Julio Bustacara Medina Ph.D.

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

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

Agradecimientos

Cristian: Le agradezco en primer lugar a mi familia, a mis papás, y mis hermanos, por todo el apoyo que me han brindado a lo largo de mi vida, y sobre todo en esta época universitaria que ha sido una experiencia muy bonita. Quiero agradecerle a Valentina por siempre, motivarme, soportarme, guiarme y acompañarme a lo largo de todo este proceso y siempre entender y ayudarme en los momentos más difíciles, adicionalmente agradecerle a Magita por siempre estar ahí, y ser la mejor compañía siempre. También le agradezco a mis amigos y compañeros de tesis por siempre dar un buen ambiente de trabajo, es un placer trabajar con ellos, son responsables, trabajadores, comprensivos y motivadores. Por otro lado, también le quería agradecer a nuestra directora Andrea, por toda la orientación, guía, y disposición con nosotros para sacar este proyecto adelante.

Kenneth: Le agradezco primero a Dios por permitirnos terminar nuestro proyecto de grado, a mi familia por darme su inmenso apoyo y comprensión en cada paso, a mis amigos que he cultivado a lo largo de la carrera profesional, ya que de ellos he aprendido un montón, le agradezco a nuestra directora Andrea que nos ha orientado de la mejor manera para solventar nuestras inquietudes, sus recomendaciones hablan de una persona con alta experiencia que sin duda nos enriqueció de muchas formas para la realización del proyecto, y le agradezco inmensamente al profesor Jaime Collazos ya que gracias a su ayuda, nos permitió comprender de mejor manera el contexto al que se dirige nuestro sistema.

Juan: Me gustaría agradecer en primer lugar a Dios, por su guía en todos los procesos que seguimos como grupo y el haber puesto personas que nos ayudaron de manera inmensa en el desarrollo de la tesis, a mis padres, mis compañeros de semestre, y a mis amigos Kenneth, Cristian y Camilo por todo el apoyo, el ánimo, el esfuerzo, la guía y los buenos momentos que pasamos durante todo el proceso de la tesis. Y, por último, quiero agradecer a Jaime Collazos por su ayuda con respecto a los temas del servidor, y a nuestra directora Andrea Rueda por todo el apoyo y la dirección ante nuestras dudas y problemas en el desarrollo de este proyecto de grado.

Camilo: En primer lugar, quisiera agradecerle a Dios por acompañarme en esta etapa de la vida, a mis padres José Eli Sandoval López y Claudia Mónica Guayambuco Pinzón, a mi hermana Alejandra Sandoval Guayambuco por apoyarme y comprenderme durante todo este proceso universitario el cual

fue muy grato y de mucho aprendizaje, a mi perrito Mailo por distraerme cuando la situación estaba estresante, igualmente a mi tío Diego Mauricio Guayambuco Pinzón ya que gracias a él fue que yo inicie en la carrera de ingeniería de sistemas, le agradezco también a mis compañeros de tesis, son el mejor grupo de trabajo que pude haber tenido, cada uno de ellos cuentan con unas aptitudes excepcionales, de ellos aprendí bastante tanto a nivel personal como profesional, a nuestra directora Andrea Del Pilar Rueda Olarte, por confiar en nosotros desde el primer momento en que le comentamos la idea del proyecto y orientarnos de la mejor manera con sus recomendaciones para el desarrollo mismo.

TABLA DE CONTENIDO

LISTA DE ILUSTRACIONES

LISTA DE TABLAS

ABSTRACT

This document shows the creation of a sign language translation system, this system is made with the purpose of achieving integration between people and the deaf community, given that, there is a wide gap in communication and with help of this system, which will use machine learning and neural networks as a basis so that it can detect signs and subsequently translating them. To carry out this system, it was decided to select a work methodology such as *scrumban* so that the development would be more pleasant, so that the use of various tools was established for aspects such as the interface, background of the system and the container where the system is, in turn the operation of the system will be shown with its respective results and the accuracy of the classification of the signs. Finally, the conclusions associated with the project were concluded and how the development of the system could be continued so that it is increasingly robust and that more people and institutions can contribute to it.

I- INTRODUCCIÓN

En el presente documento se muestra la creación de un sistema de traducción de lengua de señas, este se hace con el propósito de lograr la integración entre las personas y la comunidad sorda, dado que, existe una brecha amplia de la comunicación y con ayuda de este sistema, el cual utilizará como base el aprendizaje de máquina y redes neuronales para que este sea capaz de detectar las señas y posteriormente traducirlas. Para realizar este sistema, se optó por seleccionar la metodología de trabajo S*crumban* para que el desarrollo fuera más ameno, de forma que se establecieron el uso de diversas herramientas para aspectos como, la interfaz, trasfondo del sistema y el contenedor donde se alojara el sistema, a su vez se mostrará el funcionamiento del sistema con sus respectivos resultados y la precisión de clasificación de las señas. Finalmente se presentará las conclusiones asociadas al proyecto y cómo se podría continuar con el desarrollo del sistema para que cada vez sea más robusto y que más personas e instituciones puedan contribuir en él.

II- DESCRIPCIÓN GENERAL

1. Oportunidad, Problema

1.1. Contexto del problema

Actualmente, la población sorda del país representa el 1% de la población general de Colombia según el último censo realizado en el año 2019, esto equivale a un total de 550.000 personas con discapacidad auditiva, la cual se ha visto sesgada en las actividades laborales y educativas debido a un bajo acceso a ellos [1]. Las personas sordas que han sido contratadas por empresas tienen en su mayoría un nivel de formación educativa técnica como tendencia máxima, trayendo consigo que no tengan un trabajo estable o, si llegan a obtenerlo, sus ingresos son bajos, procediendo a la falta de garantía de los derechos que poseen como colombianos [2].

En el transcurso de este siglo XXI, la lengua de señas colombiana se ha establecido gracias a la estandarización de su gramática, por medio de entidades estatales como lo es la INSOR, que han sido garantes para poder difundir de manera eficaz, las señas homogenizadas a partir de un diccionario que demuestra cómo realizar la seña para que sea aprendida y empleada por parte de la población sorda. Lamentablemente aún queda un largo recorrido que ayude a las entidades institucionales en instruir de mejor manera a los jóvenes y adultos en diferentes campos como lo es la ciencia, arte, entre otros. Donde estos ayuden a fortalecer las capacidades de la comunidad, mejorándoles su calidad de vida a partir de nuevas oportunidades.

Nuestro país se ha comprometido con ofrecer mejores oportunidades educacionales y laborales a la comunidad sorda, con el fin de que los niños que posean la limitación auditiva opten por obtener sus conocimientos bajo el lenguaje de señas. Tristemente el acceso por parte de las personas sordas ha sido poco, dejando un abismo académico que no permite que se fortalezcan las habilidades adecuadas de acuerdo a las condiciones de los niños y jóvenes. Según las cifras dadas por el DANE, la mayoría de las personas sordas carecen de oportunidades laborales y normativas como consecuencia de los abismos en la educación pública brindada por parte del estado, para los niños y jóvenes con limitaciones de lenguaje.

1.2. Formulación del problema

Para entender el problema nos planteamos la siguiente pregunta: *¿Cómo desarrollar un sistema que traduzca las señas y que pueda ser usado por lo sordos para que sean escuchados socialmente?*, teniendo en cuenta que las herramientas a las que pueden acceder los sordos

para poder comunicarse son pocas, se reduce las oportunidades para ellos; es por ello por lo que como trabajo de grado desarrollaremos un sistema que, con la ayuda de expertos y de la propia comunidad, pueda tener un primer acercamiento y de ahí seguir evolucionando para que este pueda ser usado cotidianamente por la comunidad sorda y que podamos comprender las necesidades de ellos a partir de la traducción de las señas gracias al uso de la Inteligencia Artificial. Nuestro interés común es dar ese impulso que queremos ver en nuestra sociedad a que sea más inclusiva frente a un grupo, que representa aproximadamente el 1 % de la población colombiana y aporta un gran volumen de personas que quieren aprender, mejorar y ser partícipes de mayor manera en nuestra sociedad.

1.3. Propuesta de solución

¿Cómo desarrollar un sistema que sea capaz de traducir la lengua de señas mediante el procesamiento de imágenes? Se propone realizar un sistema que facilite el reconocimiento de la lengua de señas colombiana (LSC) mediante el procesamiento de imágenes y el aprendizaje de máquina. Y a su vez, por medio una interfaz de usuario se pueda acceder a diferentes funcionalidades como lo pueden ser el agregar nuevas señas al sistema, y hacer uso de la cámara para poder representar dicha seña y junto a esto poder recibir la traducción respectiva.

1.4. Justificación de la solución

La solución planteada permite sentar las bases para este problema social a nivel nacional, dado que, mediante el uso de aprendizaje de máquina y el procesamiento de imágenes tomaran un papel importante en el trabajo de grado, debido a que se requiere de una gran cantidad de datos como lo son las palabras de la lengua de señas y la gramática de estas en contextos cotidianos. Adicionalmente, usaremos un modelo automatizado el cual se podrá reentrenar con nuevas palabras (señas), y a su vez, organizaciones y personas interesadas en continuar con la ampliación del proyecto enfocado en otras categorías de la lengua de señas, podrán utilizar este sistema, dado que diseñando esta solución, seríamos uno de los pocos proyectos a nivel nacional que pueda brindar un acercamiento a una solución real a esta problemática, beneficiando así a la población con discapacidad auditiva.

2. Descripción del proyecto

El presente Proyecto tiene como objetivo brindarle a la comunidad con discapacidad auditiva y a las diferentes partes que quieran continuar con el desarrollo del sistema que se pretende realizar, una implementación capaz de dar interpretación de las señas presentes en la lengua de señas Colombiana, sin embargo, es necesario complementar la base de datos del léxico de esta lengua, por lo cual, a los diferentes actores como personas e instituciones interesadas, se les da la posibilidad de ayudar a aumentar la extensibilidad de las señas de forma intuitiva llegando a tener la innovación de nuevos proyectos en los que se desarrolle o se dé lugar a una mayor inclusión social de la población sorda en nuestra sociedad.

El proyecto fundamentalmente necesita de volúmenes de datos, en este caso de imágenes, videos para la previa obtención de información como son las diferentes señas de la LSC que nos permita entrenar los datos no estructurados mediante la utilización de las técnicas de aprendizaje de máquina como lo pueden ser los modelos de entrenamiento, principalmente las redes neuronales. De este modo, nuestro sistema podrá clasificar y mostrar de manera clara el significado de la seña realizada y que con el tiempo se pueda ofrecer un sistema eficiente y aplicable en diversos aspectos sociales y estatales donde se logre reducir esa brecha de la comunicación entre las personas con discapacidad auditiva y las que no.

2.1. Objetivo general

Desarrollar un sistema que sea capaz de identificar y traducir la lengua de señas colombiana mediante el procesamiento de imágenes.

2.2. Objetivos específicos

- Generar como mínimo un conjunto de datos de cinco señas estáticas con un total de 200 imágenes por cada seña.
- Generar como mínimo dos modelos de interpretación de señas, con diferentes técnicas y estructuras para poder ser comparadas.
- Alcanzar una precisión de más del 80% para cada uno de los sets de datos de cada seña en el sistema.
- Construir una interfaz para el usuario en donde se puedan ingresar nuevas señas al sistema.
- Validar mediante una interfaz, las señas ingresadas en el sistema con diferentes entornos y personas.

2.3. Entregables, estándares utilizados, y justificación

Entregable	Estándares asociados	Justificación
Memoria del trabajo de grado	IEEE	Es el documento final de trabajo donde se tiene la información esencial de la tesis desarrollada por el grupo de trabajo
Plan de proyecto	IEEE 16326-2009 [3]	Documento de control para administrar el proyecto, en él se encuentran normas, políticas, tareas, procedimientos, horarios y recursos necesarios para completar el proyecto
Especificación de requisitos	IEEE 830 [4]	Documento el cual contiene los requerimientos y comportamientos asociados al sistema
Especificación del diseño	IEEE 1016-2009 [5]	Documento el cual contendrá todo lo necesario para lograr planear y comprender el desarrollar el sistema
Propuesta del diseño	IEEE 1016-2009 [5]	Documento que contiene la solución propuesta asociada al diseño
Prototipo del sistema	IEEE 730 [6]	Se entrega el código fuente del sistema junto con el documento de control de calidad el cual contiene la descripción y resultados de las pruebas realizadas al prototipo
Plan de pruebas	ISO / IEC / IEEE 29119-3 [7]	Documento del plan de pruebas, donde contendrá el detalle de las pruebas ejercidas del Sistema.
Manual de usuario	ISO/IEC/IEEE - 26512-2011 [8]	Se entregará un manual en donde se le explique a los usuarios las funcionalidades del Sistema.

Tabla 1 Entregables del proyecto

III- CONTEXTO DEL PROYECTO

1. Trasfondo

La lengua de señas colombiana, LSC, es el sistema que es utilizado por la población sorda del país, esta lengua se caracteriza por el uso de gestos visuales y corporales el cual permite la comunicación entre la comunidad. Sin embargo, en Colombia ha habido al menos tres lenguas de señas distintas:

- La primera de ellas es la Lengua de Señas de la Isla Providencia, donde según los referidos autores se trata de una lengua emergente, por lo que un gran porcentaje de las personas sordas que hablan esta lengua están en la isla caribeña, pero los diversos estudios que se han hecho sobre la lengua no indican filiación con otra lengua de señas conocida y tampoco hay evidencias de que esta lengua se use todavía.
- La existencia de un segundo sistema fue uno referido por Fanny Bolívar, una misionera protestante que hacia el año 2006, la cual se encontraba laborando con indígenas de la etnia *Cuiba* en los llanos colombianos y dado que había un número inusualmente alto de nacimientos de niños sordos entre los *Cuibas*, había llevado a que se desarrollará una lengua de señas propia, sin embargo, esta seña no tenía vinculación con la LSC y no se conoce referencias posteriores acerca de esta lengua.
- La tercera lengua de señas es la LSC, esta es ampliamente usada por un número impreciso pero creciente de personas sordas colombianas, en todo el territorio del país.

Adicionalmente, existe una institución pública que traza a nivel nacional las políticas para la educación de las personas sordas, dicha institución es el Instituto Nacional para Sordos (INSOR), el cual lleva desde la década de 1970 coordinando una gran cantidad de programas pedagógicos y de investigación relacionadas con los sordos colombianos [9]. En la lengua de señas tanto la gesticulación de la cara como los gestos corporales juegan un papel importante para que las señas tomen significado, no para todas las señas se usa la gesticulación de la cara, solo si se quiere remarcar la seña mediante una expresión facial. Existen dos tipos de señas:

Las señas estáticas son aquellas que se realizan y no involucran algún movimiento extra, ejemplo de estas son: el abecedario, meses del año, alcalde, balón, etc.

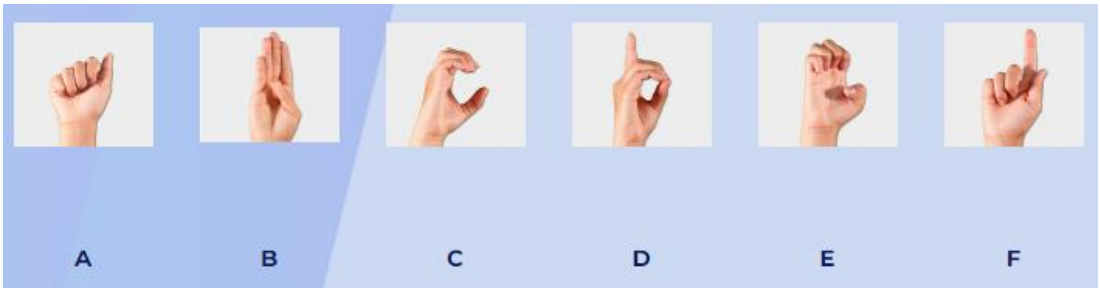

Ilustración 1 Abecedario LSC

Las señas dinámicas son aquellas que involucran movimiento para que tomen significado, este tipo de señas son las que más abundan en la lengua, ejemplo de estas son: invitar, jugar, llevar, etc.

Para efectos del proyecto nos guiaremos del libro "Lenguaje manual colombiano" [10] publicado por la Federación Nacional de Sordos de Colombia, que muestra las señas donde se predominan las manos enfocándonos como primera instancia en las señas manuales estáticas, es decir, que no requieren movimiento para su interpretación, esto es clave para la captura de imágenes y realización de este trabajo.

Ilustración 2 Jugar LSC

Un aspecto fundamental del sistema es que utiliza el aprendizaje de máquina, que permite que los sistemas aprendan sin ser programadas para ello, esto es indispensable para que la maquina sea capaz de identificar patrones entre los datos para hacer predicciones. [11] Dado que el sistema requiere de una gran cantidad de datos, es decir de señas, se utilizan técnicas como el "aumento de datos", que permite aumentar tanto en tamaño como en diversidad el conjunto de datos, es decir, posiciones, formas, colores y tamaños adicionales generados a partir de esta técnica. [12] Por lo que, con ayuda del aprendizaje de máquina se pueden identificar patrones para cada seña y mediante una red neuronal la cual es un método de la inteligencia artificial que enseña a las computadoras a procesar datos de una manera que está inspirada en la forma en que lo hace el cerebro humano por lo que el sistema es capaz de aprender de sus errores y mejorar continuamente la predicción y clasificación de las señas [13], cabe aclarar que en un principio el modelo solo aprenderá algunas señas estáticas, pero en un futuro esta red neuronal puede reentrenarse y aprender tanto las demás señas estáticas como las señas dinámicas, sin embargo para que este aprendizaje se

haga de manera más precisa se hará uso de un algoritmo de ajuste fino, el cual ayuda a mejorar la precisión de una red neuronal haciendo que el proceso de integración y procesamiento de datos sea más eficiente en términos de recursos y de tiempo [14].

2. Análisis de contexto

A nivel mundial y nacional han existido varios proyectos a lo que refiere a los traductores de lengua de señas, dentro de ellos destacan:

- **MIVOS**, es un software desarrollado por un grupo de jóvenes chilenos que traduce en tiempo real el lenguaje de señas generado por una persona sorda a un audio de voz, a través de la cámara de un celular o de un ordenador. Del mismo modo, el sistema escucha la respuesta por voz de una persona y la traduce al lenguaje de señas. El software reconoce los movimientos de las manos y verbaliza las letras del abecedario. Del mismo modo, el sistema escucha la respuesta por voz de una persona que habla y la traduce en lengua de señas [15].

Ilustración 3 Software MIVOS

- **Voz & Señas**, es una aplicación de traducción de lengua de señas mexicana (LSM) favorece la comunicación entre una persona sorda y una persona ordinaria, dentro de sus usos sirve como interprete. Es una herramienta auxiliar para las buenas prácticas en los procesos de alfabetización, redacción de textos y comprensión lectora [16].

Ilustración 4 Interfaz Voz & Señas

- **Showleap**, es un proyecto español que consiste en un traductor de lengua de señas en tiempo real y se trata de una herramienta que convierte la lengua de señas a lenguaje verbal mediante brazaletes que siguen los movimientos del usuario, sin embargo, este funciona solo con pocos símbolos, por lo que ahora usan un sistema de cámaras que detecta el movimiento y un software traduce el texto a voz, cabe aclarar que esta versión está en prueba [17].

Ilustración 5 Brazalete Showleap

- A nivel nacional, la Universidad Católica de Colombia realizó la implementación de un algoritmo, para la clasificación automática de lenguaje de señas colombiano en video usando aprendizaje profundo para la identificación de cinco palabras del lenguaje de señas colombiano, a su vez se realizaron varias muestras para su captura y análisis de ella para obtener una precisión exitosa de la palabra señalada [18].

Ilustración 6 Clasificación número uno - Ucatolica

- Reconocimiento y clasificación del lenguaje de señas usando Kinect e inteligencia artificial, en este proyecto se utiliza un sensor Kinect para la toma de datos del lenguaje de señas, el objetivo de este proyecto consiste en dar la capacidad de entender las señas realizadas por alguien con discapacidad auditiva mediante un programa que utiliza esta tecnología y la implementación de inteligencia artificial [19].

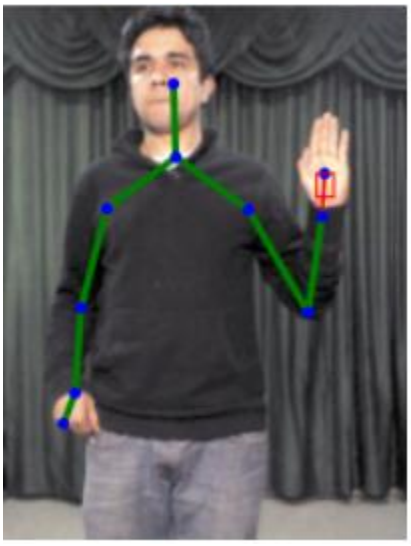

Ilustración 7 Reconocimiento de señas con Kinect

- Guante electrónico para la interpretación y traducción del lenguaje de señas en personas con discapacidad auditiva mediante tecnología como lo es el uso de Arduino e interfaz de visualización por medio de una aplicación en Android. Dicho proyecto consiste en realizar movimientos manuales con el guante electrónico y que el sistema pueda interpretar y traducir el lenguaje de señas donde dicha seña se podía visualizar mediante una interfaz en Android [20].

Con base en los trabajos expuestos previamente, se presenta la siguiente tabla comparativa que resume las alternativas que solucionan el mismo problema propuesto

Comparativa de alternativas				
Alternativa	Tecnología similar	Palabras estáticas (Sin contar el abecedario)	Abierto a futuros colaboradores	Interfaz de usuario
Mivos	Si	No	No	Si
Voz & Señas	No	No	No	Si
Showleap	No	Si	No	No
Aprendizaje profundo U. Católica	Si	Si	No	No
Kinect	No	No	No	No
Guante eléctrico	No	Si	No	Si

Tabla 2 Comparativa de alternativas

De acuerdo con la tabla 1, estos son algunos de los muchos ejemplos que existen en el mundo en cuanto a traductores de lengua de señas, evidentemente existen muchísimos más, sin embargo, consideramos que estos ejemplos fueron los más remarcables en cuanto al reconocimiento y traducción del lenguaje de señas. Teniendo en cuenta estos ejemplos podemos observar que la mayoría de estos proyectos toman como ejemplo el abecedario y a partir de ahí construyen la palabra letra por letra, cosa que no tiene mucho sentido ya que normalmente las personas sordas no hablan deletreando las palabras, adicionalmente, solo se tiene como muestra una cantidad no tan amplia de palabras para su clasificación, otro aspecto es a lo que refiere a la interfaz del usuario, dado que muchos de estos proyectos solo se hacen con fines académicos, es decir, solo reconocimiento y clasificación de las señas y no con el fin de pensar en un beneficio de la comunidad sorda en un futuro y en la manera de cómo lo verían las personas sordas para poder comunicarse. Por eso consideramos que nuestro proyecto tiene una gran ventaja respecto a los demás, en cuanto a la interfaz, se tiene una interfaz amigable al usuario para que este realice la seña y el sistema le diga que seña es, y otro aspecto es que, si bien en un principio tiene un número considerable de señas estáticas registradas en el sistema, este puede quedar abierto para futuros colaboradores y así, seguir ampliando el contenido en lo que a señas se refiere.

IV- ANÁLISIS DEL PROBLEMA

1. Requisitos

Los requisitos funcionales se definen a partir de las características necesarias para ofrecer una experiencia del manejo del sistema apropiada. En línea con la metodología empleada, estos requisitos son priorizados en una escala de 1 a 5, donde 1 representa la máxima prioridad y 5 la mínima.

ID	Requisito	Prioridad
RF-01	El sistema debe mostrar las funcionalidades que puede hacer el usuario.	4
RF-02	El sistema debe capturar una seña a guardar.	1
RF-03	El sistema debe agregar una nueva seña al conjunto de datos.	1
RF-04	El sistema debe permitir el reentrenamiento de datos (señas).	1
RF-05	El sistema debe permitir registrarse.	3
RF-06	El sistema debe permitir hacer inicio de sesión.	3
RF-07	El sistema debe traducir la seña.	1
RF-08	El sistema debe reproducir el sonido de la seña traducida.	4
RF-09	El sistema debe permitir ver las estadísticas de las señas traducidas.	5
RF-10	El sistema debe permitir cambiar los parámetros del entrenamiento.	2
RF-11	El sistema debe permitir visualizar el entrenamiento de las nuevas señas.	4

Tabla 3 Requisitos funcionales

Teniendo en cuenta la tabla anterior, sabiendo que estos requisitos funcionales deben realizarse en un lapso de un semestre, se debe tener en cuenta que para el desarrollo de algunos requisitos depende del cumplimiento de otros, por lo tanto, se presenta el siguiente diagrama donde se muestra el trayecto de la realización de estos.

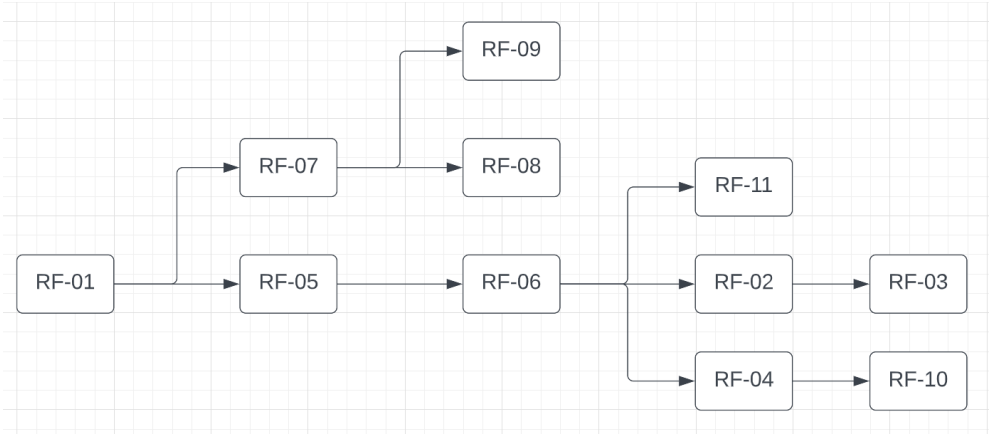

Ilustración 8 Diagrama de precedencias

Adicional a los requisitos funcionales, el proyecto requiere ciertos requisitos no funcionales los cuales se plantean pensando no solo en el desarrollo mismo durante el semestre sino en un proyecto a largo plazo, por lo que aspectos como la mantenibilidad y la disponibilidad son cruciales de cara al crecimiento futuro del sistema. Sin embargo, estos no son los únicos requisitos no funcionales asociados al proyecto, por eso se plantea la siguiente lista de requisitos no funcionales asociados a los atributos de calidad mencionadas.

ID	Requisito	Atributo de calidad
RNF-01	El sistema debe emplear protocolos de comunicación ampliamente utilizados.	Mantenibilidad
RNF-02	El sistema debe desarrollarse empleando tecnologías de amplia aceptación.	Mantenibilidad
RNF-03	El sistema debe estar completamente en español.	Usabilidad
RNF-04	El sistema debe ser sencillo e intuitivo para el usuario.	Usabilidad
RNF-05	El sistema debe mantener conexión con sus componentes el 99% del tiempo.	Disponibilidad

Tabla 4 Requisitos no funcionales

2. Restricciones

Desde un punto de vista técnico no existen restricciones notables a tener en cuenta adicional a los requisitos mencionados para hacer el sistema expansible y a su vez mantenible. No obstante, hay restricciones de otra índole en lo que compete a la construcción del sistema como lo es el alcance y es que si bien, el vocabulario de la LSC es bastante amplio por lo que una restricción de entrada es en cuanto a la cantidad de señas que el sistema es capaz de interpretar, las señas seleccionadas son algunas señas estáticas

3. Especificación funcional

Para una mejor comprensión del sistema, se presentará un diagrama de alto nivel el cual contiene los elementos fundamentales de este, se describirá cada uno de ellos y se irá detallando con mayor precisión.

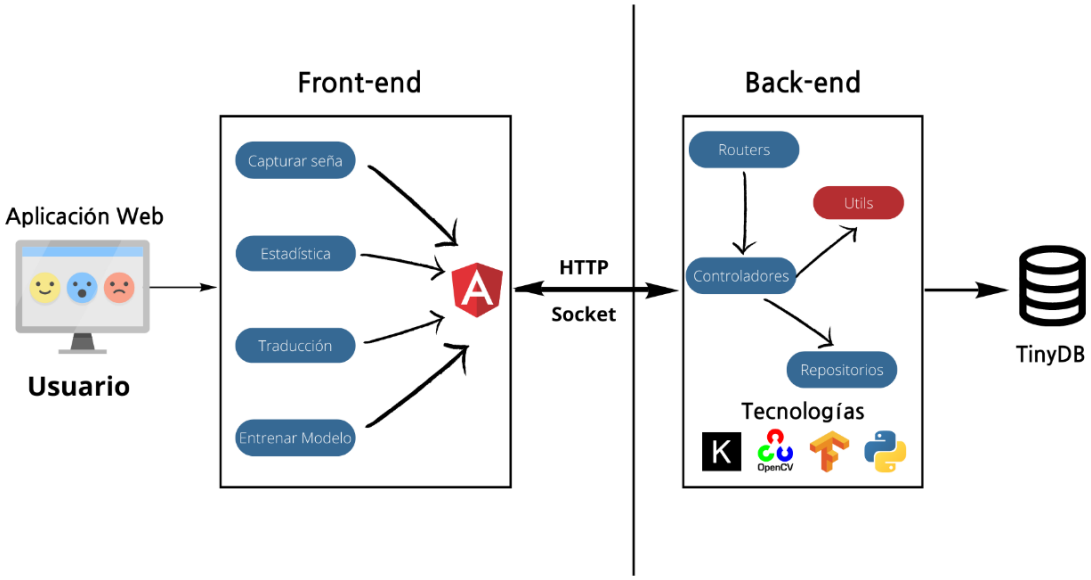

Ilustración 9 Funcionamiento de alto nivel del sistema

Para abarcar de mejor manera el diagrama de la ilustración 9 se hará énfasis como primera instancia, en que el usuario podrá hacer uso del Sistema solo en las instalaciones de la universidad o por medio de la VPN *Arpuj* debido a políticas del servicio y seguridad ofrecida para el despliegue del aplicativo en la maquina dada por el departamento de TI. Una de las principales interacciones que puede desempeñar el usuario es dirigirse desde el menú al componente de Traducción, este nos provee como fin el resultado de la seña producida, este proceso se especificará a lo largo del documento. Cabe recalcar, que el menú cambiará de acuerdo de si el usuario se encuentra autenticado en el Sistema como son los componentes de Captura de señas y entrenamiento del modelo.

Las interacciones que tienen los componentes de Angular con el *back-end* se hacen gracias al uso del *framework FastApi*, dicha comunicación será por medio de peticiones HTTP o por medio de un socket que nos ayuda a establecer una conexión directa para que sea eficiente el envío, procesamiento y recepción de datos (Bidireccional). La razón por la cual se escogió y se usó *FastApi* es debido a que nos ofrece la facilidad de desarrollar y gestionar el progreso del Sistema, gracias a que posee un gran rendimiento, logrando consigo la admisión de la

simultaneidad integrada y solucionando problemas de inyección de dependencias, ayudando en agilizar el progreso del aplicativo y en los posibles cambios en la codificación del proyecto. Siendo vital, ya que detecta inconsistencias en los tipos de datos, dando como ventaja que existan datos integrados que mejoran la modularidad del código y la escalabilidad del sistema [21].

Para finalizar con el diagrama funcional de alto nivel, se debe optar por tener un sistema consistente, es decir, que los datos persistan de acuerdo con lo que el usuario vaya ejerciendo en él, para ello se determinó el uso de la base de datos *TinyDB*, ya que nos ayuda a tener una base de datos orientada a documentos, es decir se va a alojar la información en un archivo JSON donde se realizaran las consultas requeridas por el usuario.

3.1. Diagrama de clases.

En la siguiente ilustración se muestra la estructura de los modelos del sistema en donde tenemos cada modelo representado como clase con diferentes atributos y relaciones entre ellas dado el tipo de algunas de sus variables. Primero que todo tenemos la clase "*Signal*" que representa la entidad de una seña en donde tenemos los atributos *name* que representa el nombre de la se*ña, processed_image* indica si la seña fue procesada o no en el sistema, y por último está la variable *images* representa la lista de las imágenes de tipo "*Image*"; esta clase a su vez está representada de la misma forma que "*Signal*" con la variable *name* e *image*, pero difiere en la variable *image* que es una cadena de caracteres de caracteres que representa el contenido de una imagen en formato de base 64.

Podemos encontrar una clase "*ModelVariables*" que representa todos los datos de un modelo ya sea que esté en proceso de entrenamiento o ya finalizado, este estado es representado por la variable *training_state* que es un enumerador con los estados: CREATED representando la creación de un modelo, STARTED representando que el modelo está listo para iniciar su entrenamiento, PROCESSING representando que el modelo está en proceso de entrenamiento, FINISHED representando que el modelo finalizo su proceso y se guardó como éxitos, y por último está el estado ERROR que representa que el modelo presentó un error en cualquier parte de su proceso. Luego tenemos las variables de identificación como lo son *id* y *name* , variables de datos del modelo como lo son *loss*, *val_loss* que representan el valor de la pérdida del modelo, *acurracy* y *val_acurracy* que indican el valor de la precisión del modelo, *epoch* y *cant_epochs* que muestran la época de ejecución actual y la cantidad total de épocas por entrenar respectivamente, *mean_time_execution* que es el promedio de ejecución de una época en segundos, *begin_time* y *end_time* que representan la fecha de inicio y de fin de la ejecución del modelo y, por último, el *trained*_signals que muestra todas las señas entrenadas por el modelo, estas se representan en un mapa teniendo como llaves el id de la seña y valor el nombre de la seña.

Posteriormente vemos la clase "*CoordSignal*" que representa todas las coordenadas de cada parte del cuerpo generadas por la holística de *MediaPipe*. *Face* representa los puntos del rostro, *pose* los puntos básicos del cuerpo, *handLeft* y *handRight* indican los puntos de las manos y *ea* muestra los puntos extra generados. *Face*, *handLeft* y *handRight* son listas de tipo "*CoordsWithouthVisibility*" en donde se encuentran las variables *x*,*y* y *z* de las coordenadas de cada punto. *Pose*, *ea* son listas de "*Coords*" donde además de tener las variables *x*,*y* y *z* tiene la variable *visibility* que muestra el porcentaje de visibilidad del punto indicado. Esta entidad es la que el sistema utiliza para realizar todos los procesos de entrenamiento y para comprimir los pesos de cada imagen al realizar el aumento de los datos al ser guardados en documentos de tipo "*npy*" como listas planas.

La última entidad es más simple pero necesaria para todo el proceso de autenticación de los usuarios, "*User*" tiene una variable *username* y *name* para representar el usuario de manera única, y *password* es la variable que almacena la contraseña del usuario, pero de manera codificada con *JWT,* y, por último, la variable role es utilizada para que el usuario quede identificado con un rol que muestre los permisos que tiene cada usuario en el sistema.

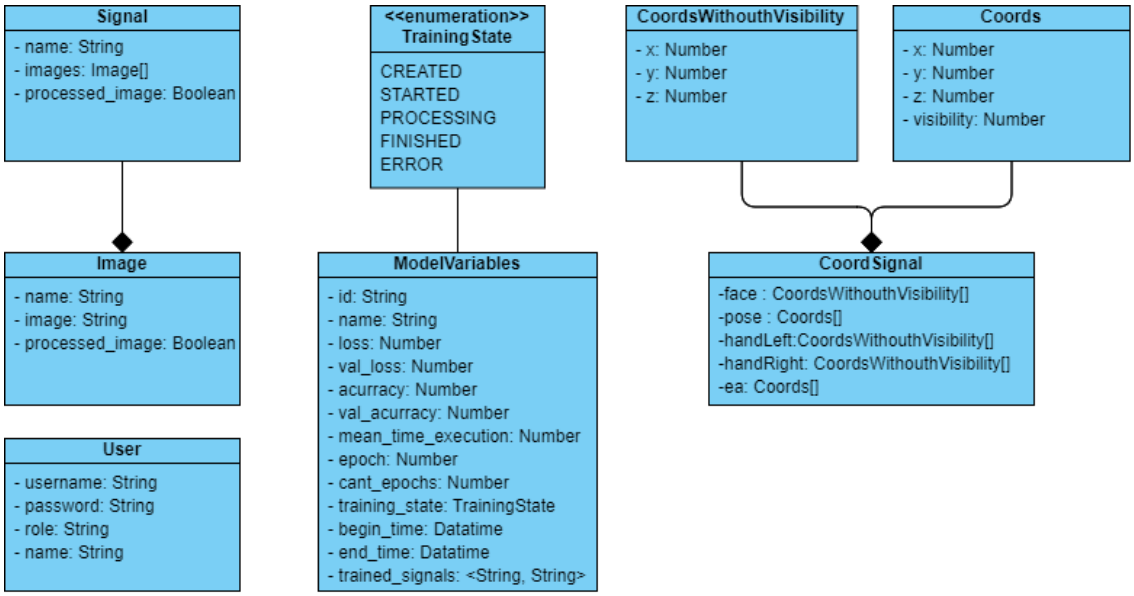

Ilustración 10 Diagrama de clases del sistema

3.2. Procesos BPMN de los componentes relevantes del sistema:

Para comprender de manera sintetizada los procesos donde el usuario interactuará en el sistema, se podrá detallar en los siguientes apartados.

- **Proceso Captura nueva seña:**

Inicia el proceso, se revisa si se tiene permiso para utilizar la cámara, en caso de que no se tenga se deberá permitir y procederá a mostrar los componentes de la interfaz, luego el usuario deberá ingresar el nombre de la seña y tomar mínimo un par de fotos, en caso de que oprima el botón de guardar se procederá a validar los datos de la seña, si es válida la información se enviaran esos datos al *back-end* y terminará el proceso para ser procesados, en caso contrario mostrara un mensaje de error.

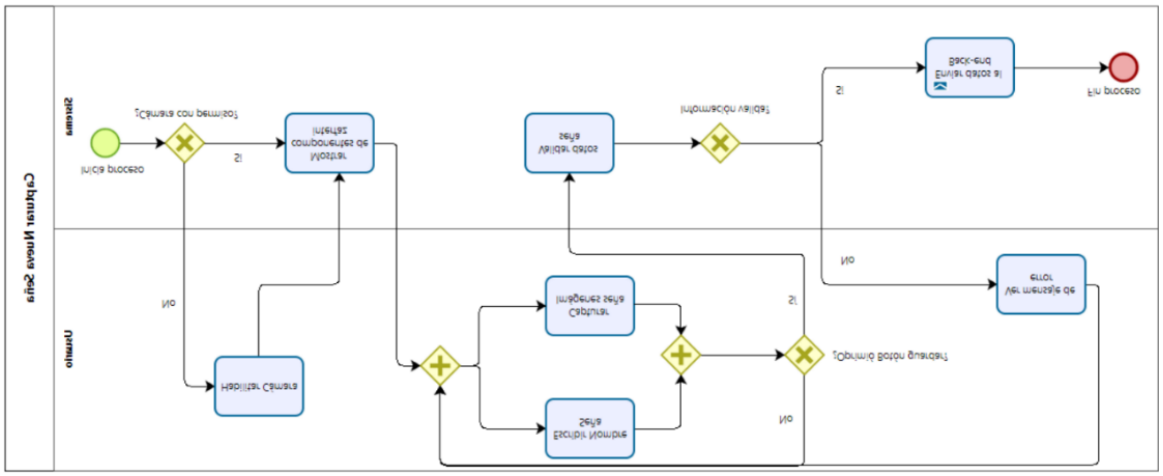

Ilustración 11 Proceso de captura nueva seña

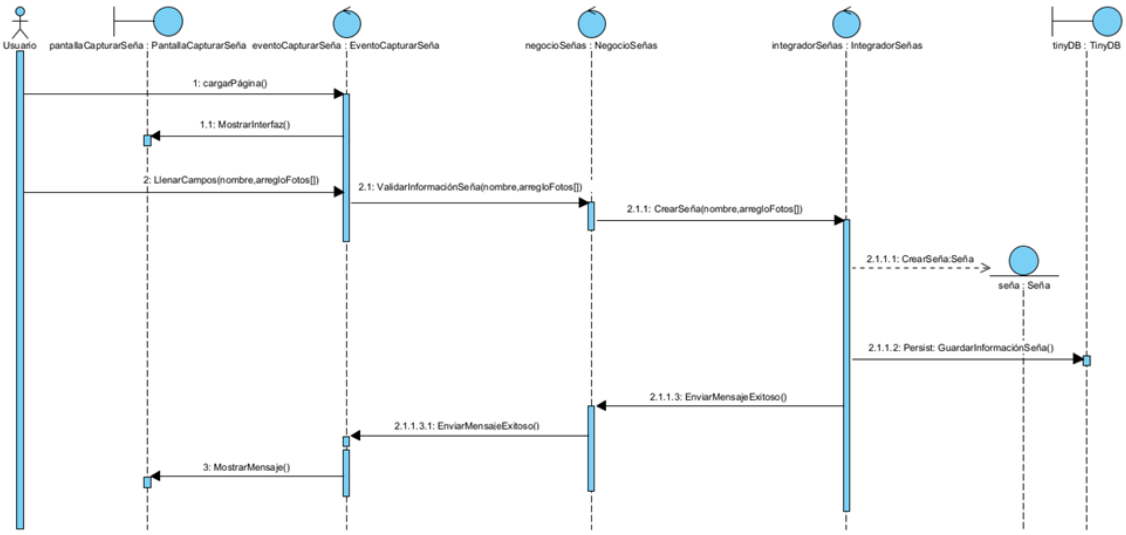

Ilustración 12 Diagrama de secuencia captura nueva seña

- **Proceso Traducción seña:**

El proceso inicia revisando si se tiene permiso para utilizar la cámara, en caso de que no se tenga se debe habilitar, se mostraran los componentes de la interfaz, luego el usuario deberá interactuar con el sistema realizando alguna seña frente a la cámara, se procesaran las coordenadas de las manos, se enviaran estas coordenadas al *back-end*, en donde esta procesara estas coordenadas y clasificara la seña y devolverá el resultado al *front-end*, el *front-end* recibirá la respuesta y mostrará los resultados, si el usuario desea seguir interactuando, podrá realizar una nueva seña frente a la cámara, en caso contrario podría salir del componente y terminará el proceso.

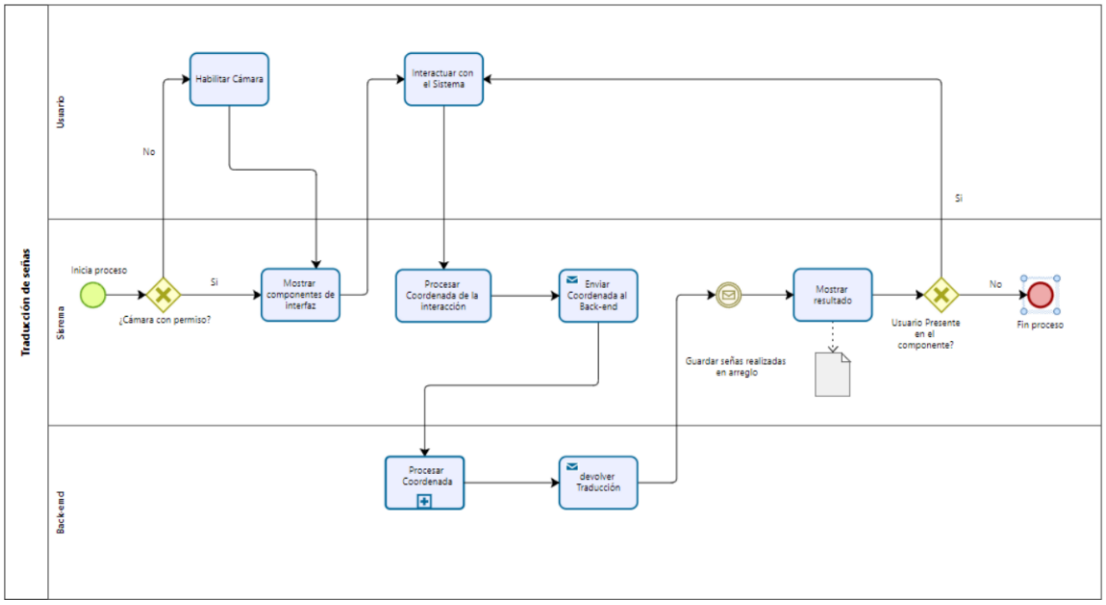

Ilustración 13 Proceso de traducción de seña

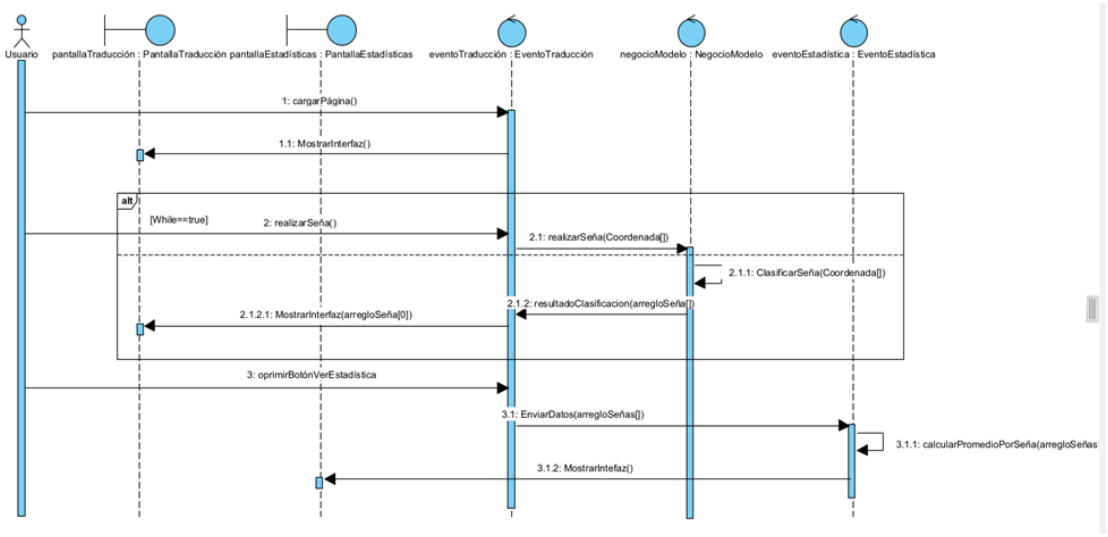

Ilustración 14 Diagrama de secuencia traducción y visualización de seña

- **Proceso ver estadística por seña:**

Inicia el proceso una vez el subproceso de traducción haya sido utilizado, en caso de que se oprima el botón de ver estadísticas, este con la información almacenada calculara el promedio de precisión por seña y mostrará un diagrama de barras verticales con esta información.

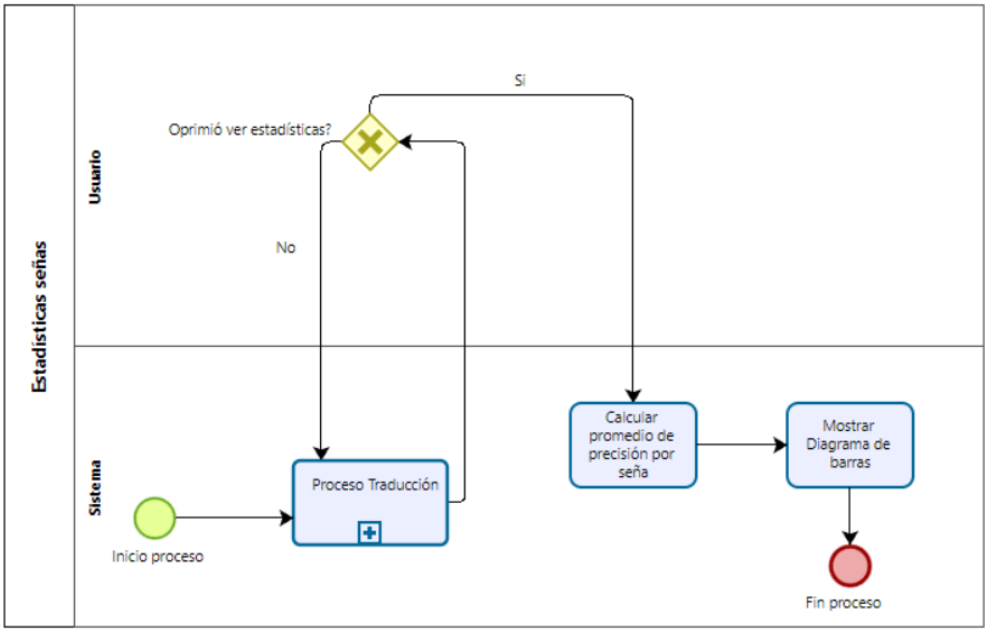

Ilustración 15 Proceso de estadística por señas

- **Proceso entrenamiento modelo**:

Inicia el proceso en el *front-end* donde consulta al *back-end*, para saber si existen señas por ser entrenadas, en caso de que, si existan señas por entrenar, el *back-end* le notifica al *front-end*, en este caso se habilitara el botón de entrenamiento, luego el usuario procederá a oprimir el botón de entrenamiento, por lo que este le notificara al *back-end* debería empezar a entrenar el modelo con las señas nuevas.

Una vez empiece a entrenarse el modelo, se le notificara al *front-end* de esto, y este le solicitara la información necesaria para calcular el tiempo aproximado para que el entrenamiento finalice, esto lo hará cada cinco segundos para asegurarnos de que el tiempo aproximado sea una buena estimación, una vez el *back-end* le entregue esta información al *front-end*, este procederá a calcular este tiempo, y a mostrarlo en pantalla, una vez el entrenamiento termine y el tiempo sea cero, se mostrara un mensaje avisando de que se terminó el proceso.

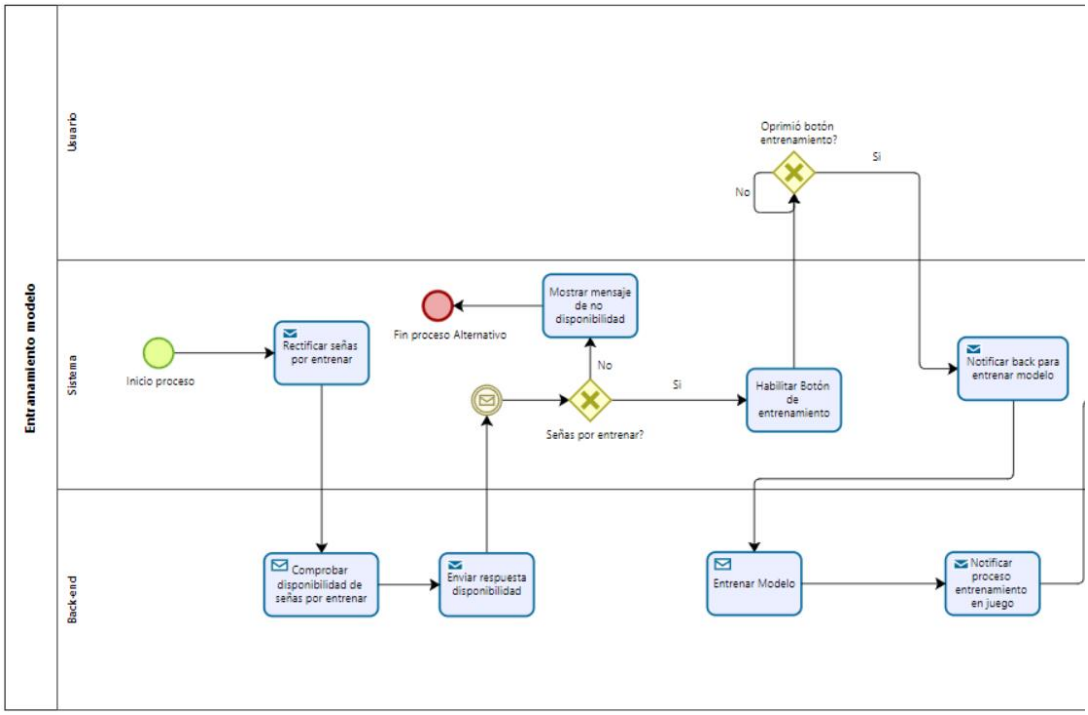

Ilustración 16 Proceso de entrenamiento del modelo parte 1

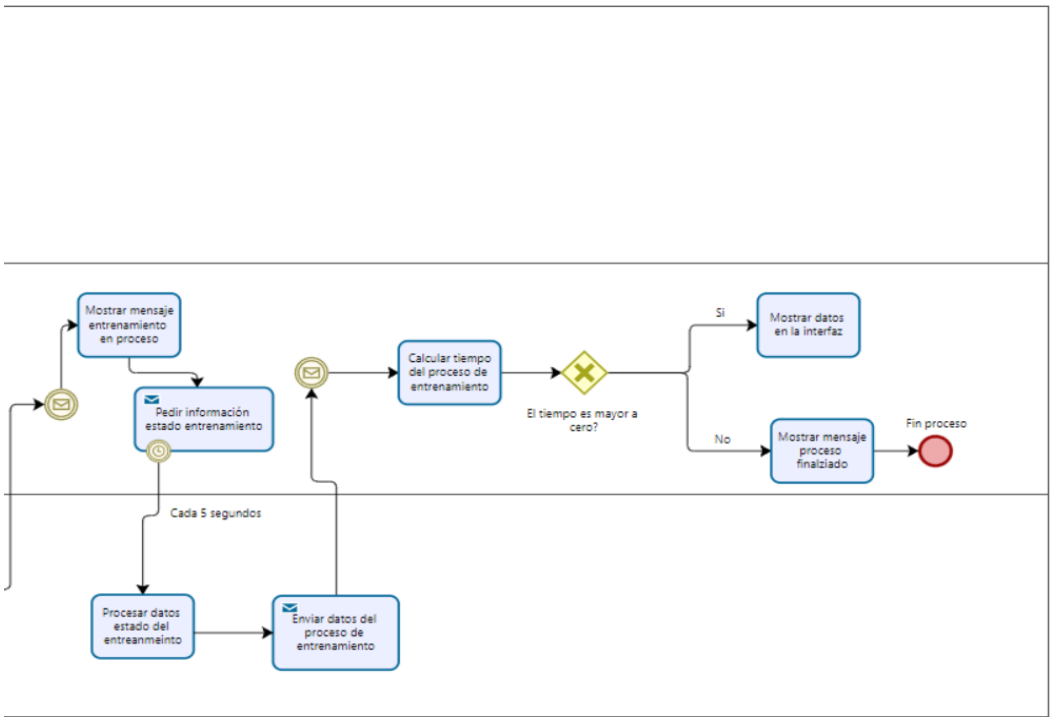

Ilustración 17 Proceso de entrenamiento del modelo parte 2

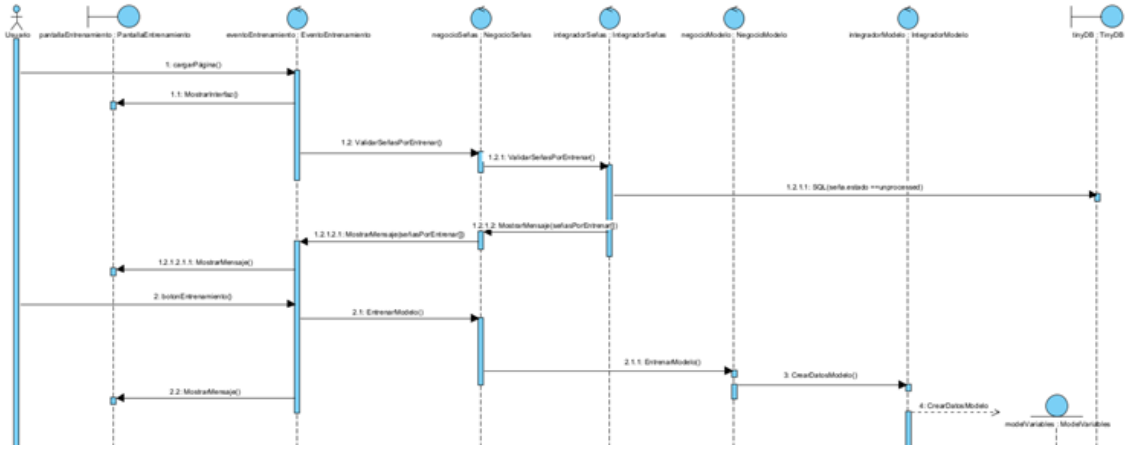

Ilustración 18 Diagrama de secuencia entrenamiento del modelo parte 1

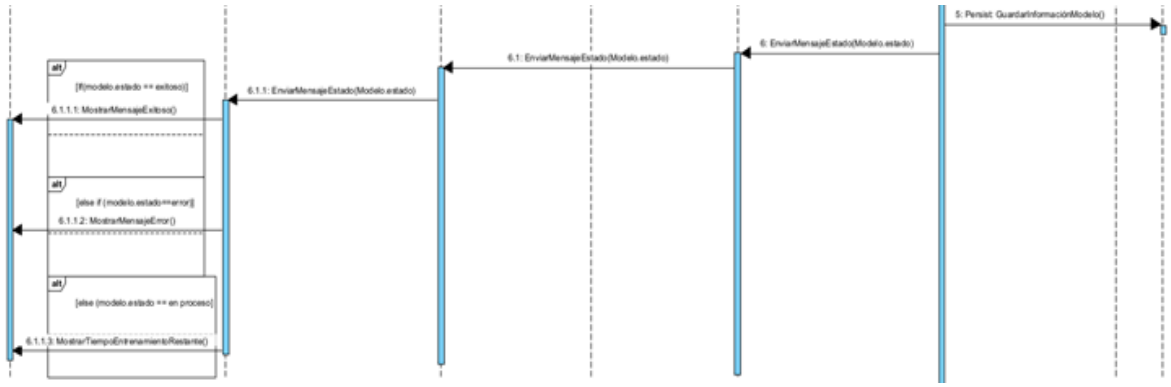

Ilustración 19 Diagrama de secuencia entrenamiento del modelo parte 2

3.3. Historias de usuario

Los requerimientos del Sistema fueron tomadas y sintetizadas en las historias de usuario para poder estructurar y comprender las funcionalidades que se deben regir en el Sistema. Para informase de las diferentes funcionalidades que los usuarios pueden realizar se detallará en la tabla 4 y a su vez, con mayor enfoque en el documento PMP, las cuales son:

Historias de usuario	
Usuario	Desarrollador del sistema
Como usuario, requiero una interfaz para realizar la seña mediante el uso de la cámara para obtener la traducción de esta.	Como desarrollador del sistema, requiero una interfaz para capturar nuevas señas y agregarlas al conjunto de datos
Como usuario, requiero un manual de instrucciones para aprender a utilizar el aplicativo.	Como desarrollador del sistema, requiero una interfaz para añadir nuevas imágenes de la seña a una seña existente en el conjunto de datos
Como usuario, necesito visualizar el comportamiento al reentrenar el modelo.	Como desarrollador del sistema, requiero tener instrucciones adecuadas para poder contribuir al desarrollo del sistema.
Como usuario, requiero poder registrarme para poder colaborar en el desarrollo del sistema	Como usuario, requiero la opción de reentrenar el modelo con las señas que están presentes en el conjunto de datos.

Tabla 5 Historias de usuario

Como se pudo observar en la tabla de historias de usuario, se tienen dos clases de usuario que interactuarán con el sistema, el primero de ellos es el usuario, mejor conocido como usuario final o consumidor del sistema y en este caso serían las personas con discapacidad auditiva para que este a través de la interfaz pueda realizar la seña en la cámara y que el resultado de la traducción se muestre tanto en texto como en audio para que la otra persona con la que este interactuando pueda comprenderlo. El otro usuario es el desarrollador del sistema o también conocido como contribuidor y estos se dividen en dos: aquellos que tienen conocimiento en programación, tendrán acceso al código y pueden bien sea mejorar o agregar funcionalidades al sistema para que este sea cada vez más robusto y los otros son aquellos con conocimientos del lenguaje de señas para que mediante las interfaces, puedan agregar más señas (bien sean estáticas o dinámicas, dependiendo del estado actual del sistema) al conjunto de datos.

Cada historia de usuario tiene asociado diferentes parámetros los cuales están descritos en la siguiente tabla, si desea visualizar todas las especificaciones de las historias de usuario diríjase al Anexo 2.

ID: 1	Usuario: Usuario final - Desarrollador del sistema
Nombre de la historia de usuario: Como usuario, requiero una interfaz para realizar la seña mediante el uso de la cámara para obtener la traducción de esta.	
Prioridad: Alta	Riesgo en desarrollo: Media
Descripción: El usuario podrá utilizar la interfaz del sistema mediante el sitio web y entonces, dirigirse a la pestaña de traducción para realizar la seña que quiere ser traducida y que posteriormente sea mostrado el resultado de la seña traducida.	
Criterios de aceptación: • El usuario debe habilitar permisos de activación de la cámara • La seña que el usuario realiza debe estar previamente almacenada en la base de datos, de lo contrario	

Tabla 6 Especificación de historia de usuario 1

ID: 5	Usuario: Desarrollador del sistema
Nombre de la historia de usuario: Como desarrollador del sistema, requiero una interfaz para capturar nuevas señas y agregarlas al conjunto de datos.	
Prioridad: Alta	Riesgo en desarrollo: Alta
Descripción: El usuario debe ser capaz de utilizar una interfaz en donde pueda ingresar el nombre de la seña que no esté en el sistema y capturar varias fotos para posteriormente agregarlas al conjunto de datos.	
Criterios de aceptación: • Este debe poseer un campo para el nombre de la seña a ingresar. • Este debe poseer un visor de la cámara para la captura de fotos. • Este debe poseer una cuadricula para observar las imágenes capturadas y debe poseer la funcionalidad de eliminar fotos, en caso de que no sea una buena foto para entrenar.	

Tabla 7 Especificación de historia de usuario 5

V- DISEÑO DE LA SOLUCIÓN

1. Arquitectura

Para la arquitectura del sistema, se mostrará la arquitectura tanto para el *front-end* como para el *back-end*, siguiendo el siguiente diagrama de despliegue se ilustra el comportamiento que tiene, con respecto al lugar donde se montó el aplicativo, el nodo de pc cliente, representa como el usuario accede a los servicios mediante el navegador de su preferencia, cabe aclarar que el usuario debe conectarse como primera instancia por la VPN de la Universidad Javeriana *Arpuj*, logrando finalmente comunicarse mediante peticiones HTTP al sitio Web , el usuario podrá utilizar los servicios ofrecidos y estos se conectarán al *back-end* por peticiones HTTP y/o Sockets y para persistir la información se tomó la base de datos *TinyDB*.

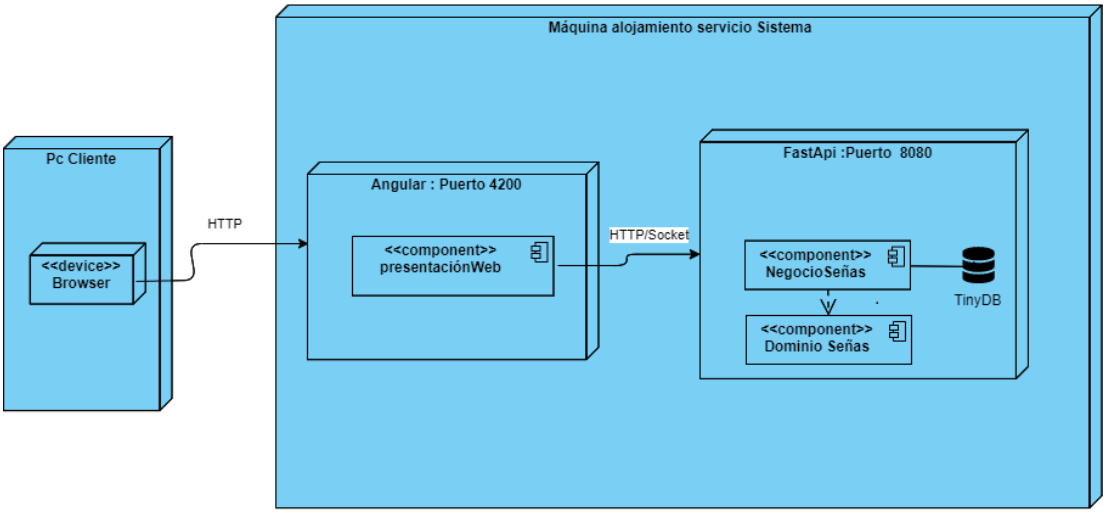

Ilustración 20 Diagrama de despliegue del sistema

1.1. Arquitectura Front-End

Para definir la arquitectura que se ejerció mediante el *framework* de Angular, se estableció el diagrama de estados, en él se resume el flujo de actividades que pueden ocurrir en el sistema y como debe comportarse frente a las acciones que se tomen, para más detalle ver la siguiente ilustración.

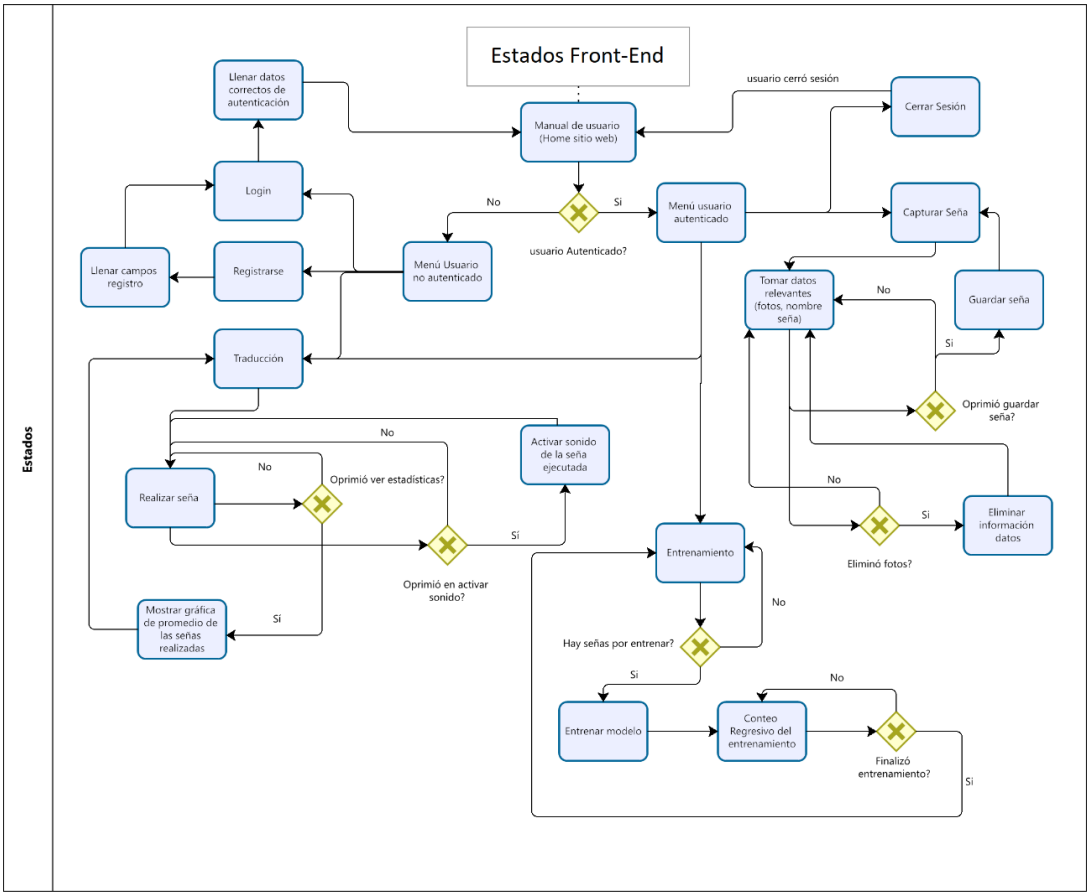

Ilustración 21 Diagrama de estados

Para comprender mejor el componente de presentación Web de la ilustración 21 del documento, trajo consigo, la realización del diagrama de paquetes, en él se puede notar los diferentes componentes del Sistema y como estos interactúan con los servicios realizados, para eventualmente comunicarse por medio de peticiones HTTP y/o Sockets al *back-end*. Haciendo que cumplan con las funcionalidades básicas del aplicativo como lo es recibir, actualizar y crear diferentes datos, que puedan demostrar el correcto funcionamiento del Sistema de interpretación del LSC (Lenguaje de Señas Colombiana).

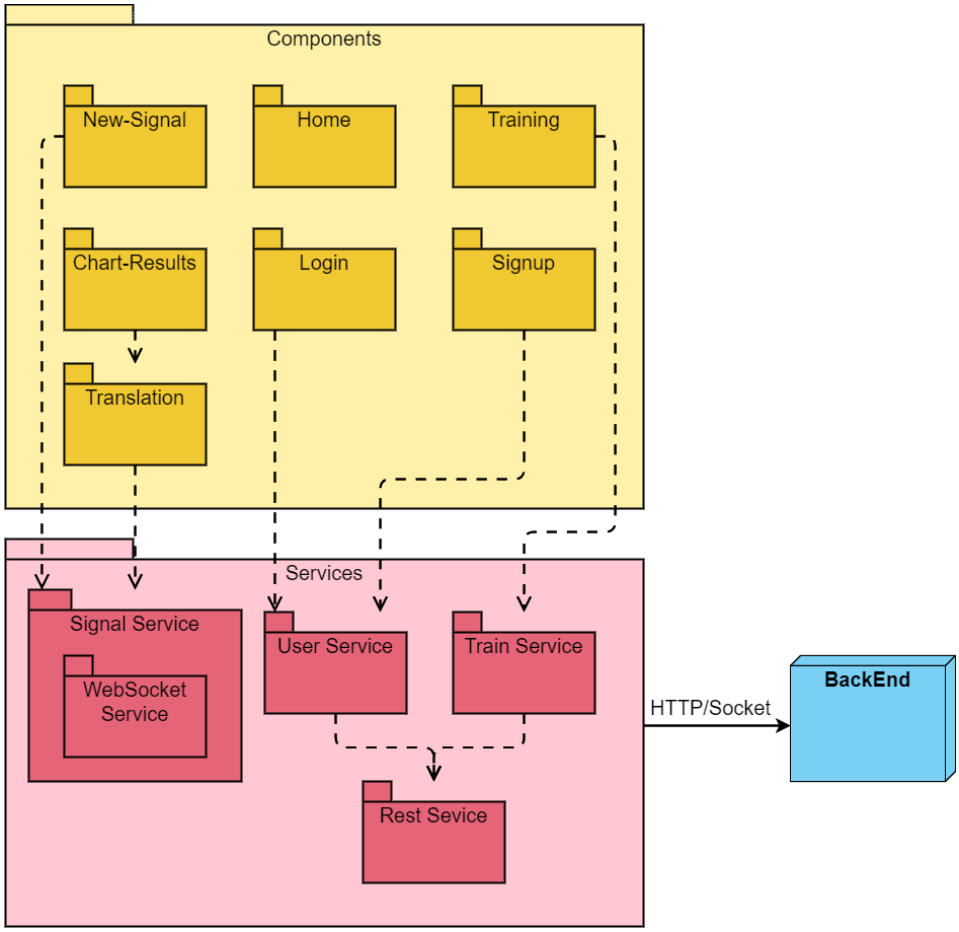

Ilustración 22 Diagrama de paquetes front-end

1.2. Arquitectura Back-end

El diagrama de paquetes back-end será explicado desde el paquete *Routes* (ver ilustración 26) a sus dependencias ya que es ahí donde se reciben los llamados HTTP para ser procesados.

- **Signal Routes**: Este paquete recibirá las peticiones referentes a las señas como es la información de una nueva seña y aceptar la conexión del *socket* para recibir las coordenadas, de los puntos de referencia del cuerpo de la persona, que el usuario envié y de esa forma delegarla al siguiente componente para eventual atender la solicitud respectiva.

- **Train Routes**: Este paquete recibirá las solicitudes que se encargan en realizar el entrenamiento del modelo, retornar los datos demandados y mirar el estado del entrenamiento.
- **User Routes**: Es el encargado de recibir las peticiones de registro y autenticación del usuario, la información obtenida se delega al siguiente componente.

Todas las peticiones recibidas pasan por un proceso de verificación en los *Routes* para posteriormente ser enviadas a los *Controllers*.

Antes de explicar los paquetes *Controllers* se explicará los paquetes *Repository*. Los tres paquetes *Signal, Model y User repository* contienen las funcionalidades *CRUD* para cada entidad mencionada en la sección 3.1; la base de datos utilizada es *TinyDB* la cuál almacena la información en un archivo json, pero junto a esta, para el proceso de las señas, se utilizó funcionalidades de almacenamiento en memoria a través de directorios locales, debido a que todo el contenido de las imágenes y el aumento de los datos maneja archivos de grandes tamaños y con crecimiento de tamaño cada vez mayor dada la cantidad de señas, entonces estas son almacenadas de manera local pero los datos de la información para su procesamiento si se encuentra en *TinyDB.*

Los paquetes Controllers reciben los datos verificados de los *Routes,* y actúan como intermediarios de los paquetes *Repository.* Estos *Controllers* que se encargan de toda la lógica del negocio y la orquestación de los servicios para dar el resultado correspondiente. Cabe aclarar que, aunque los *Controllers* tiene la lógica de negocio del servidor, los paquetes *Utils* y LSC_recognizer*_model* tienen la lógica del proceso de *MediaPipe* y el modelo realizado en *Tensorflow.*

- **Signal Controller**:
    - **Crear seña**: Se encarga de registrar una nueva seña dada la información recibida (nombre, arreglo imágenes), primero mediante una instancia del SignalRepository se rectifica en la base de datos si existe la seña, si la respuesta es que no, se realiza todo el procedimiento pertinente como es tomar el modelo Signal, el cuál es una clase que contiene (nombre, arreglo de imágenes y deja un booleano en falso ya que la seña aún no ha sido procesada), se le asigna los respectivos valores de acuerdo a la solicitud y se persiste la información en la base de datos. En caso contrario si la seña ya existe se actualiza la información de la seña dejando como no procesada las imágenes.
    - **Predicción seña**: recibe por parámetros las coordenadas, estas se componen de atributos como lo son los puntos de pose, *leftHand, rightHand, ea y face* (cada uno de esos puntos contienen un X, un Y, un Z y el de pose contiene

además el de *visibility*). Para realizar la predicción se toma el último modelo entrenado satisfactoriamente, para que se desempeñe bien se utilizan funciones del paquete *Utils* ya que nos provee métodos que se reutilizan constantemente durante el programa. Luego se guardarán las señas entrenadas actualmente y se guardan en las clases, a partir de eso se inicializará el modelo de señas, en donde se le asignará, el resultado de llamar a la red neuronal con el número de clases y el tamaño de la imagen, para luego llamar a la función *get_prediction* del modelo de seña ya creado y luego poder devolver esta predicción.

- **Model Controller**: este controlador este encargado de hacer la configuración, el entrenamiento y además de generar graficas informativas con respecto al entrenamiento del modelo.
    - **Obtener información de entrenamiento**: esta función sirve para obtener de la base de datos el ultimo estado del entrenamiento, no importa el estado de este, así como podría ser creado, comenzado, procesando, terminado o que hubo algún error.
    - **Observador para creación del entrenamiento**: esta función sirve para crear e inicializar un modelo con variables predeterminadas, parámetros del resultado del modelo inicializan en cero, excepto por la cantidad de épocas que serán definidas ya por parámetro de entrada, variables como el tiempo se inicializan en el momento de la creación, y el estado del entrenamiento es colocado como creado inicialmente; por último, este modelo es creado en el repositorio.
    - **Entrenamiento del modelo**: una de las funciones más importantes del proyecto, verifica si existen señas por entrenar, en ese caso coge todas las señas que no están procesadas, delega la responsabilidad para hacer la división del conjunto de datos en tres categorías (entrenamiento, validación y test), y una vez se tengan estas señas no procesadas empieza el proceso de entrenamiento.
    - **Proceso de entrenamiento del modelo**: en primera instancia, carga el conjunto de datos, luego arregla los datos ingresados, que en este caso lo que hace es que los puntos que no son usados se eliminan (por ejemplo, los puntos de la cara), luego se definen los hiperparámetros, y por último empieza el entrenamiento. Además, al finalizar el entrenamiento, este almacenara las gráficas por temas de pruebas e informativas.

- **User Controller**: Para este controlador se toma del paquete *Utils* el sub-paquete de *Auth*, ya que esta nos provee funciones para realizar la autenticación mediante el uso de *JSON Web Tokens (JWT)*.
    - **Registro**: Se destaca esta función ya que dada la información del usuario se toma el modelo de usuario que esta cuenta con lo que sería el nombre, contraseña, correo y el rol del usuario, se rectifica mediante el uso del *user repository* si ya existe un usuario con el correo electrónico significa que no se puede registrar este usuario, pero si es el caso contrario, entonces utilizamos las funciones del paquete *Auth*, como lo es la codificación de la contraseña, ofreciendo seguridad a datos sensibles y guardando el registro en la base de datos exitosamente.
    - **Autenticación**: Se encarga dada la información de ingreso como son correo electrónico y contraseña, consultar mediante el *user repository* si el usuario existe, en caso de que no exista se menciona que las credenciales no son válidas, en caso contrario, se corrobora mediante la funcionalidad del paquete *Auth* si la contraseña si coincide con la del correo escrito, si es así se crea el token de autenticación, este es codificado gracias al uso de *JWT*, siendo de vital importancia ya que de esa forma sabremos si el usuario tiene permiso para ingresar a componentes del Sistema donde es un requisito o restringir el acceso a esta si no existe el token.

Para finalizar no entraremos en detalle con el paquete de *Pixels*, debido a que se implementó para poder realizar el proceso de la creación del modelo mediante el procesamiento de imágenes, se tiene las funcionalidades de *aumento de datos* que hacen que la red neuronal aprenda de manera eficiente gracias a realizar cambios de posición de imagen, contrastes etc, pero en el transcurso de esta se obtuvo una mejor predicción mediante el uso de coordenadas siendo la solución final que ofrecemos en el Sistema.

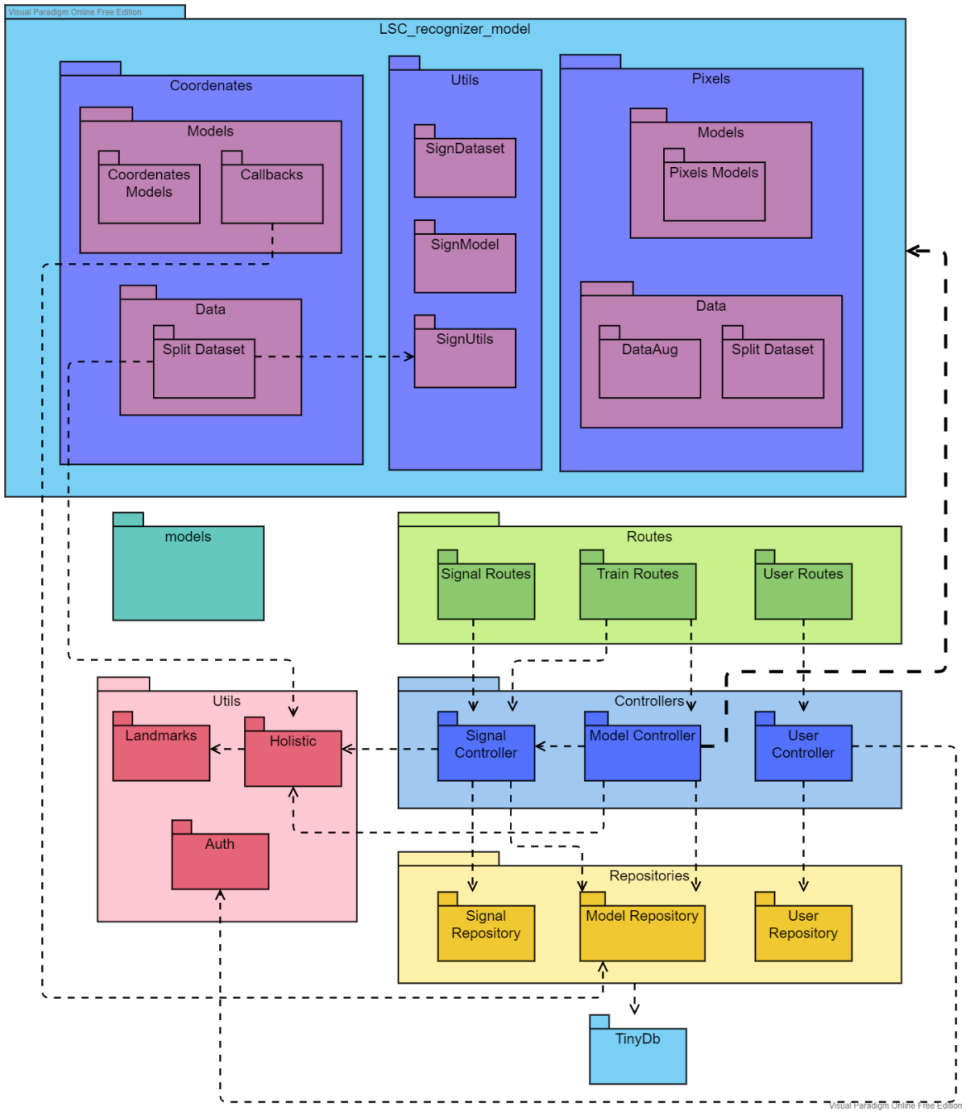

Ilustración 23 Diagrama de paquetes de back-end

1.3. Ilustraciones Resultados

A continuación, se muestra los resultados generados por el último entrenamiento del modelo final a través de una matriz de confusión, en el eje X están las etiquetas de las señas de la predicción y en el eje Y están las etiquetas de las señas recopiladas como datos reales; antes de continuar cabe aclarar que estas etiquetas están representadas del 0 al 18 las cuales se relacionan con los nombres presentes en la tabla 8. Posteriormente podemos ver que en cada

conexión de fila con columna existe un número, este número representa cuántas señas coincidieron entre la etiqueta de predicción con la etiqueta de datos reales, todas las conexiones que tengan la misma etiqueta en X y Y representan la cantidad de señas predichas correctamente, y las conexiones que difieren su etiqueta son predicciones incorrectas. Dada la explicación anterior se puede simplificar el análisis mencionando que entre más concentración de cantidades haya en la diagonal de la matriz representa un mejor resultado en la predicción del modelo.

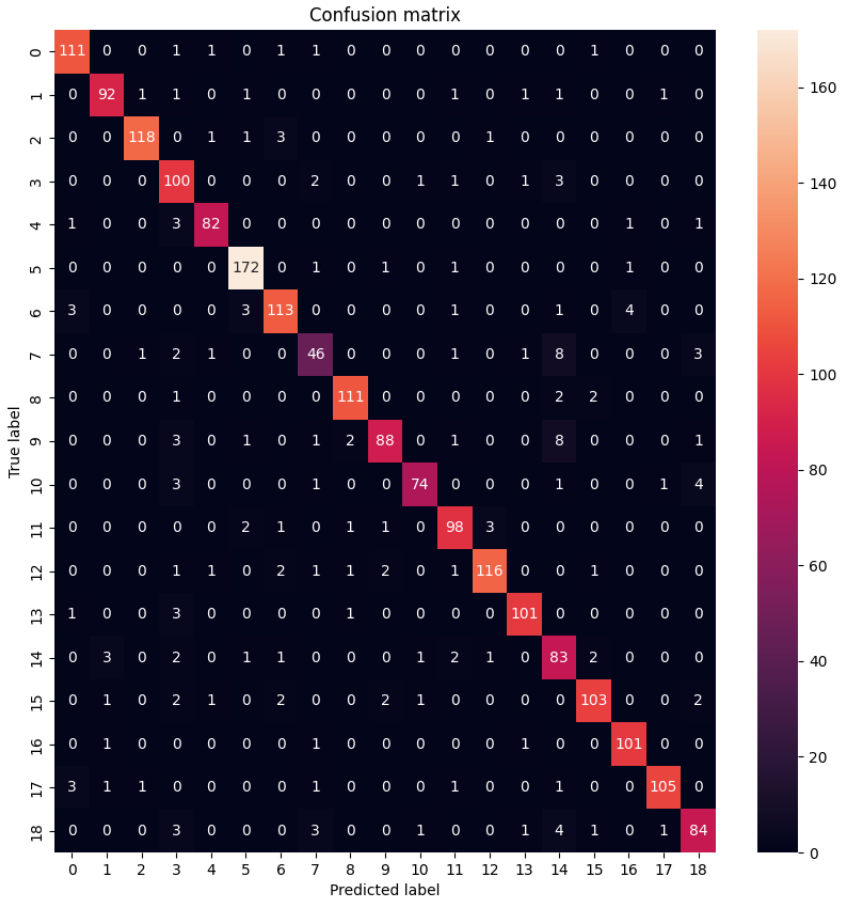

Ilustración 24 Matriz de confusión del modelo

Número de seña	Nombre de seña
0	Alcalde
1	Amor
2	Balón

3	Bien
4	Botella
5	Cama
6	Casa
7	Celular
8	E/Enero
9	F/Febrero
10	Hospital
11	Instituto
12	Lampara
13	Mal
14	Noviembre
15	O/Octubre
16	Rezar
17	Sábado
18	Teléfono

Tabla 8 Señas del sistema

- **Pérdida de valor**

En primer lugar, se presenta la gráfica de perdida, se puede analizar a través de la gráfica, que a medida que se aumentan las épocas de la perdida se disminuye constantemente con el conjunto de datos de entrenamiento, a diferencia de la perdida que ocurre con el conjunto de datos de validación, en donde este disminuye, pero en un punto se queda oscilando en un rango estable indicando que no se presentan más mejorías si se llegara a extender más el proceso de entrenamiento.

Parámetro	Detalle
Training loss	Este parámetro evalúa el error del modelo sobre el conjunto de datos ingresados en el entrenamiento.
Validation loss	Este parámetro evalúa el error del modelo sobre el conjunto de datos utilizado para la validación del modelo.

Tabla 9 Parámetros de pérdida de valor

En este caso notamos que el modelo se está sobre ajustando, es decir que el sistema funciona de manera operativa excelente con los datos de entrenamiento, pero falla un poco con los datos de validación, esto supone que no se generaliza bien el modelo.

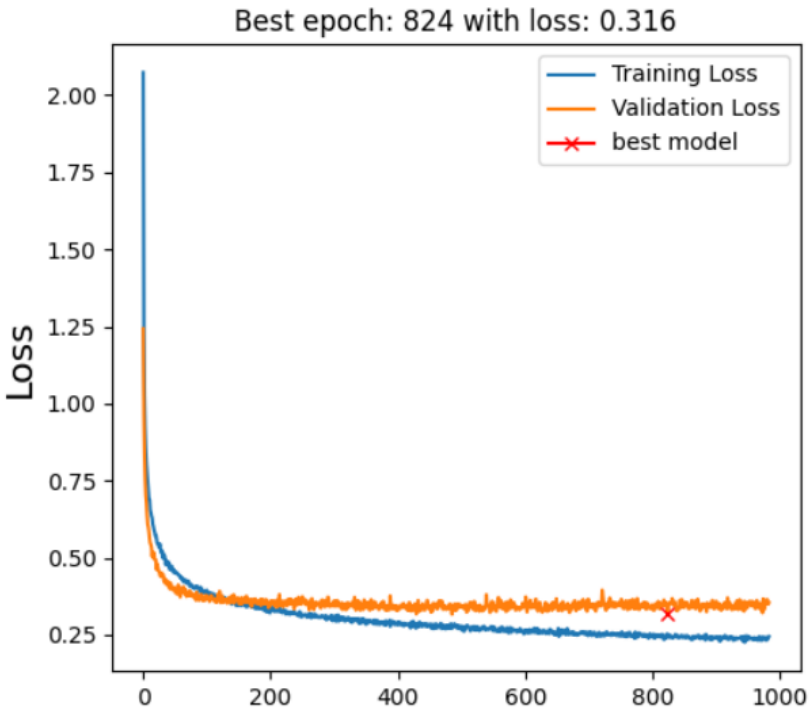

Ilustración 2255 Gráfica de pérdida de valor

- **Precisión del modelo**

También se puede observar la segunda gráfica de precisión, se puede analizar a través de la gráfica, que a medida que se aumentan las épocas, la precisión aumenta constantemente en ambos casos, en donde se utilizan el conjunto de datos de entrenamiento y el de validación, y también se puede notar que la mejor época fue la número 930 con una precisión del 93.2%, que se podría considerar un excelente porcentaje de precisión.

Parámetro	Detalle
Training accuracy	Esta evalúa la precisión del modelo sobre el conjunto de datos ingresado en el entrenamiento.
Validation accuracy	Esta evalúa la precisión del modelo sobre el conjunto de datos utilizado para la validación del modelo.

Tabla 10 Parámetros de precisión del modelo

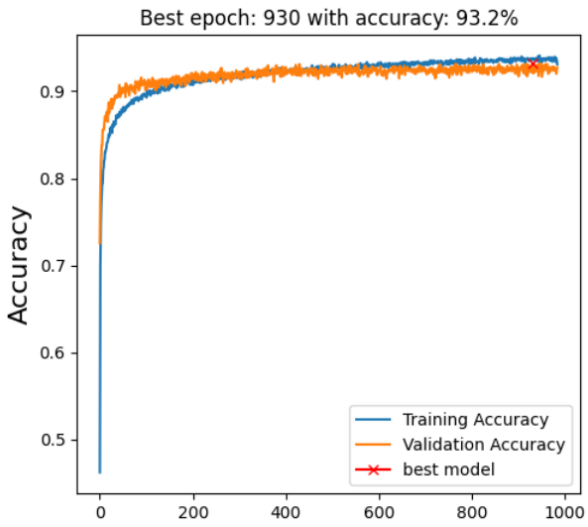

Ilustración 26 Gráfica de precisión del modelo

1.4. Estructura de la red neuronal.

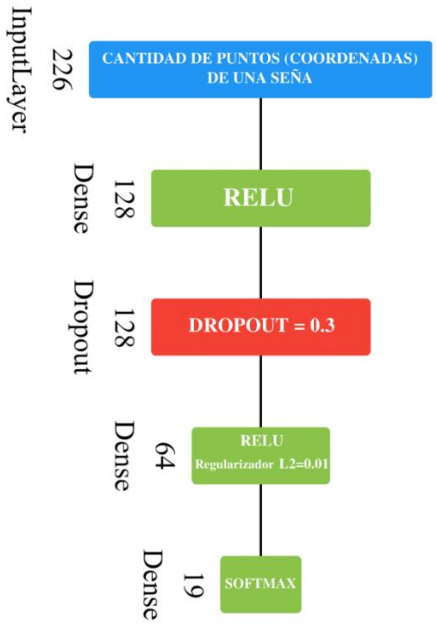

Ilustración 27 Estructura de la red neuronal

La imagen anterior [22], muestra el resultado final de la arquitectura de nuestra red neuronal, habiendo pasado por un largo historial de cambios optimizando hasta este punto. Primeramente, la arquitectura se basaba en una red con una entrada de 300 puntos de coordenada

de una seña, luego una serie de capas densas y por último una capa densa con la cantidad de señas entrenadas siendo las neuronas de salida con una función de activación "*softmax*". Posteriormente procedimos a experimentar con la cantidad de capas densas y su cantidad de neuronas y obtuvimos que los mejores resultados evaluando la precisión y la pérdida del modelo era una capa densa de 128 neuronas y una capa densa de 64 neuronas, ambas con función de activación "*relu*", estando estas en medio de las capas de entrada y salida de la red. Siguiendo en los cambios, se hicieron ajustes externos a la red como el aumento de los datos con diferentes transformaciones a la imagen obteniendo mejores resultados, pero para que la red neuronal evolucionara en la adaptación y generalización de los datos ingresamos una capa de eliminación (*dropout*) que apaga con cierto porcentaje las neuronas de manera estocástica en cada iteración para evitar el sobreajuste de los datos, este porcentaje se empezó probando con un 20%. Se hicieron cambios en las coordenadas quitando varios puntos llegando a bajar el input de la red de 300 a 226, también se normalizaron las coordenadas mejorando la interacción con la función de activación *relu*, y se cambió el porcentaje de la capa de eliminación al 30%. Por último se realizaron dos últimos cambios significativos, uno fue en los parámetros del entrenamiento, usando la función de pérdida de entropía categórica para adaptarse mejor a la naturaleza de los datos entrenados, y se agregó en la capa densa de 64 neuronas una regularización "*ridge*" del kernel de tipo L2 para evitar el sobreajuste del modelo; se utilizó L2 y no L1 o L1_L2 ya que en la naturaleza del problema hay más relevancia en las correlaciones entre los datos (puntos), y no en la eliminación de datos irrelevantes, sabiendo que además se hace limpieza de puntos de la detección de la pose y las manos anterior al entrenamiento en la red neuronal. Por medio de todo este proceso se llegó al resultado final de la arquitectura de la red neuronal, ilustrada en la imagen anterior.

Adicionalmente, el flujo de nuestros datos para la predicción comienza con las capturas de las fotos originales como muestra de la seña a representar, posterior a esto se pasa por un componente encargado de realizar el aumento de los datos con diferentes transformaciones como el giro vertical para simular la seña diestra y zurda, se hace rotación de la imagen de manera aleatoria dentro de ciertos límites, y de igual manera se realiza con un aumento aleatorio. Al finalizar la creación de nuevas imágenes se procesa por el *framework* de *MediaPipe* para obtener los resultados de los puntos de referencia del cuerpo como lo son las manos, el rostro, y el esqueleto del cuerpo en general. Al obtener las coordenadas se procesan eliminando puntos de referencia irrelevantes para la naturaleza de nuestro problema, se normalizan los datos y se llevan a un arreglo plano listo para ser ingresadas en la red neuronal, la cual se encarga de transformar estos puntos a un resultado numérico que se ve reflejado en el *id* de las señas ingresadas dando el resultado de la predicción final, a continuación, se presenta dicho flujo mediante una ilustración para una mejor comprensión.

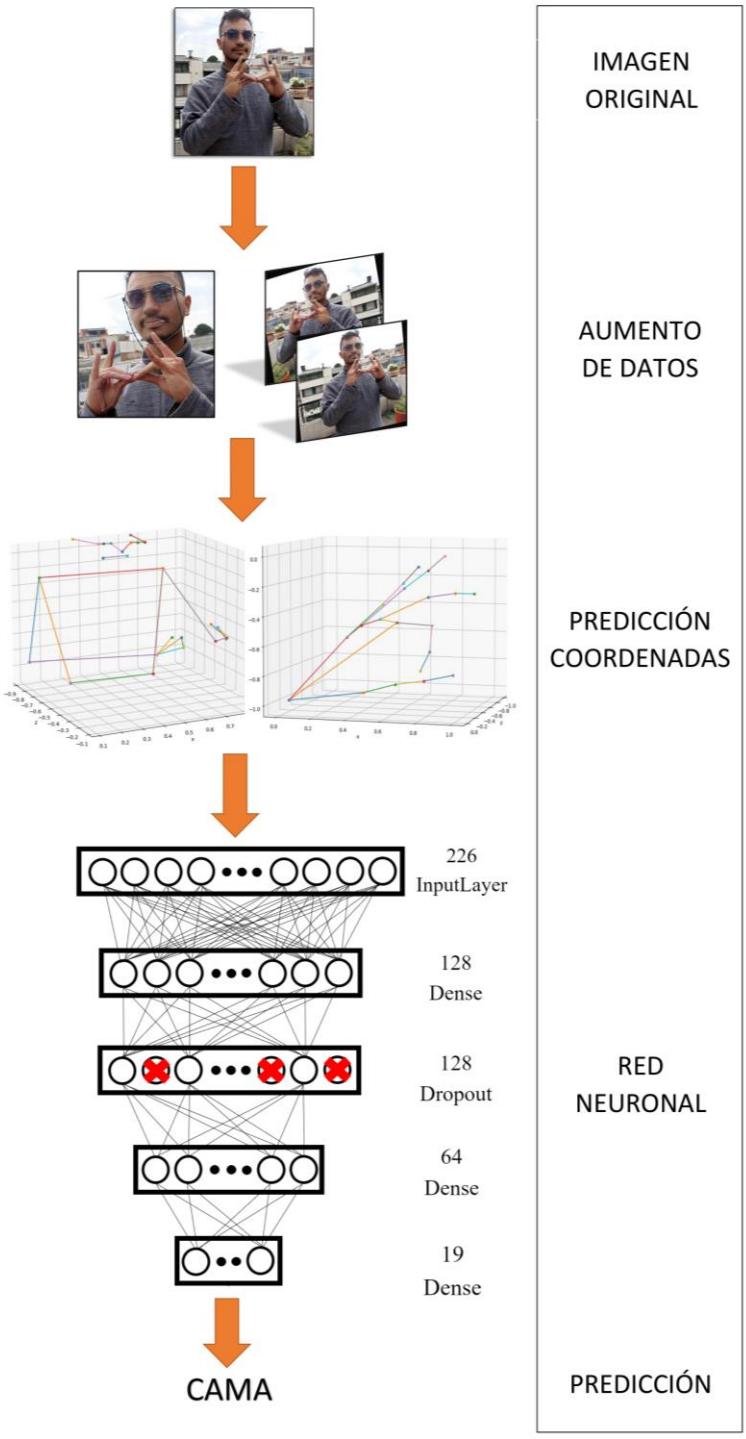

Ilustración 28 Proceso de predicción del sistema

2. Herramientas y tecnologías

Las herramientas y tecnologías utilizadas fueron las siguientes:

Herramienta / Tecnología	Descripción
Tensorflow	Es una librería de código libre para el desarrollo de la inteligencia artificial y el aprendizaje automático, esta permite construir y entrenar redes neuronales para detectar patrones y razonamientos usados por los humanos. [23] Nos pareció una buena opción esta librería porque actualmente es una de las tecnologías más famosas en cuanto al entrenamiento de redes neuronales.
Python	Este lenguaje de programación lidera en lenguajes de desarrollo de aprendizaje de máquina debido a su simplicidad y facilidad de aprendizaje, por tal razón se escogió dicho lenguaje, además de ser utilizado por científicos de datos y desarrolladores para la construcción y análisis de modelos [24].
Angular	Es una plataforma de desarrollo para crear aplicaciones web, este está construido sobre *TypeScript* el cual se compila con JavaScript. Click or tap here to enter text. Dicho *framework* fue escogido debido a que ya se tenían conocimientos de la herramienta y además, este se puede ejecutar en cualquier plataforma.
Mediapipe	Es un *framework* que posee modelos de aprendizaje de máquina para la detección de rostro, seguimiento de manos, detección y seguimiento de objetos, entre otros. [26] En nuestro caso, se utilizó dicha tecnología dado que, al ser un modelo basado en coordenadas, se puede obtener una mejor precisión en cuanto a la detección de señas sin importar lo alterada que se encuentre la imagen, es decir, la cantidad de ruido visual, contraste, brillo, ángulo de la cámara con respecto a las manos, tamaño y color de las manos.
OpenCV	Es una biblioteca de código abierto especializada en el sistema de visión artificial y aprendizaje de máquina.
Docker	Es una herramienta que empaqueta software en contenedores, los cuales tienen todo lo necesario para que este se ejecute, incluye: bibliotecas, herramientas de sistema, código y tiempo de ejecución. [27] Dado que la configuración de nuestro sistema es algo complicada y tediosa de realizar, al tener todos los recursos en un contenedor como lo es Docker, la ejecución del sistema es más sencilla.
GitHub	Es una plataforma de alojamiento, que ofrece a los desarrolladores la posibilidad de crear repositorios de código y guardarlos en la nube de forma segura, usando un sistema de control de versiones [28].
SpeechSynthesis	Es un API que permite seleccionar una pieza de texto y darle salida a través de los altavoces en forma de habla. Esta API se escogió para que la seña traducida pueda ser reproducida mediante la voz de dicha API [29].

Tabla 11 Herramientas y tecnologías

VI- DESARROLLO DE LA SOLUCIÓN

Para el desarrollo de la solución se establecieron tres fases metodológicas: captura de datos, desarrollo y validación del sistema. A continuación, se describen dichas fases junto con sus respectivas actividades.

1. Fase metodológica 1 (Captura de datos)

Para esta primera fase se hizo énfasis en la captura de datos, para ello se deben seleccionar las señas estáticas, señas que serán incorporadas en el sistema.

Actividades

- Realizar entrevistas con expertos en lenguaje de señas colombiano para que nos puedan enseñar a realizar dichas señas.
- Seleccionar las señas estáticas para agregarlas al sistema.
- Fotografiar en diferentes escenarios las señas seleccionadas.
- Agregar las fotos de las señas al repositorio de datos para luego ser utilizado en el sistema.

Resultados obtenidos

- Se realizó la entrevista con el profesor Jaime Collazos Aldana del departamento psicología de la Pontificia Universidad Javeriana quien trabajó con el INSOR hace algunos años, nos dio aspectos clave para el momento de fotografiar las señas, dichos aspectos fueron:
    - La ropa debe ser oscura en el momento de la fotografía para que logre resaltar la seña.
    - Las señas realizadas deben estar dentro de un marco delimitado, esto ya en lo que a programación se refiere.
    - Puede haber cambios en el significado de la seña si lo hace un diestro o un zurdo aun cuando significa lo mismo.
    - Es complejo definir las señas más utilizadas.
    - El INSOR y Cultura Sorda son fuentes de información en cuanto a lo que la lengua de señas compete, por lo que de ahí se buscan las señas a realizar.
- Se seleccionaron diecinueve señas estáticas, las cuales son: alcalde, cama, casa, instituto, sábado, bien, mal, rezar, lámpara, balón, amor, teléfono, celular, hospital, botella, enero, febrero, octubre y noviembre.

- Asimismo, el profesor Jaime Collazos nos recomendó una interprete certificada por el INSOR de nombre Ruth Collazos quien fue la persona que nos ayudó a la captura de las señas seleccionadas y nos enseñó la forma correcta de realizar las señas para nosotros seguir capturando más señas por nuestra cuenta.
- Las fotografías tomadas fueron agregadas al repositorio dependiendo la seña correspondiente, esto con el fin de que las señas estuvieran etiquetadas para una mejor interpretación de las mismas en fases siguientes.

2. Fase metodológica 2 (Desarrollo)

Esta fase consiste en el procesamiento de datos y la construcción del sistema, tanto el *front-end* como el modelo en el *back-end* donde se realizan todos los filtros y ajustes necesarios para obtener el modelo más preciso.

Actividades Back-end.

- Realizar procesos de aumento de datos para generalizar los datos y aumentar la cantidad de estos.
- Construir la estructura de la red neuronal
- Medir el desempeño de cada modelo con el método de ajuste fino para determinar cuál modelo tuvo la mejor precisión y a su vez utilizarlo en el sistema
- Traducir de manera precisa la seña realizada
- Almacenar las nuevas señas introducidas en el sistema
- Configurar el contenedor el cual va a tener todo el sistema
- Conectar el *back-end* con la interfaz de usuario

Resultados obtenidos

- A cada fotografía se le hacen siete transformaciones en cuanto a invertir, rotar y hacer zoom a cada foto almacenada en el repositorio para tener más cantidad de datos en mismos escenarios.
- Se construye la red neuronal para que pueda aprender de las fotos de las señas almacenadas en el repositorio, inicialmente se hace prueba la precisión de la misma con las fotos sin aumento de datos y se obtuvo una precisión del 80% con 300 datos de entrada, en consecuente a eso se hizo varias modificaciones en cuanto a las diferentes transformaciones, épocas utilizadas, normalizaciones, funciones de pérdida, etc. Así hasta obtener una precisión final de 93.2% (Para ver las transformaciones realizadas en cada iteración ver sección de *Resultados*).

Actividades del contenedor.

- Crear docker-compose.yml para tener el listado de instrucciones para empaquetar la aplicación correctamente de los servicios que componen el sistema.

Resultados obtenidos.

- Se construyó el archivo *Dockerfile* de la imagen que contendrá los componentes de *front-end* y *back-end* con sus respectivas librerías y dependencias.
- Se utiliza el comando para ensamblar y compilar la imagen "*docker-compose up*" generando el contenedor.

Actividades Front-End.

- Implementar pantallas de inicio de sesión y registro.
- Implementar pantalla de traducción de señas.
- Implementar pantalla de captura de señas.
- Implementar pantalla de entrenamiento.
- Implementar pantalla de estadísticas.
- Conectar la interfaz de usuario con el modelo del *back-end* mediante sockets y peticiones HTTP.

Resultados obtenidos.

- Se implementan las funcionalidades del usuario contribuidor dado que estos son los únicos que tienen funcionalidades de desarrollador como lo es el inicio de sesión, captura de señas y entrenamiento y esto es, como se ha venido mencionando a lo largo del proyecto, solo aquellas personas que estén interesadas en continuar con el desarrollo del proyecto podrán realizar estas funciones.
- La conexión con el modelo del *back-end* se realizó mediante dos métodos:
    - Sockets, se utiliza para que cada tres segundos el *front-end* envié lo que capta en la pantalla de traducción, es decir, la seña que el usuario está realizando en cámara se le envía al *back-end* y posteriormente el *back-end* recibe dicha seña representada en una coordenada, la cual es procesada para que finalmente el *back-end* retorne al *front-end* el resultado de la traducción de la seña realizada.
    - HTTP, se utiliza para realizar demás funcionalidades de comunicación y envío de datos, en el caso de la pantalla de captura de nueva seña cuando el usuario haya guardado la nueva seña, el *front-end* le enviará la información asociada a esta seña al *back-end* para que este la guarde en el repositorio de señas, en lo que compete a la pantalla de entrenamiento, cuando el usuario digita las épocas con las que quiere que el modelo sea entrenado este campo se le es

enviado al *back-end* para que este entrene el modelo de acuerdo a las épocas recibidas y a su vez establecer el tiempo restante de entrenamiento para ser mostrado en la interfaz.

3. Fase metodológica 3 (Validación)

En esta fase se validará el sistema mediante diferentes formas como lo son las pruebas funcionales al sistema y a su vez, una retroalimentación por parte tanto de expertos en el lenguaje de señas como personas del común.

Actividades

- Realizar pruebas unitarias al sistema tanto en el *front-end* como en el *back-end*.
- Hacer envío del formulario de retroalimentación para obtener sugerencias y/o recomendaciones acerca del sistema.

Resultados obtenidos

- Un documento asociado a las diferentes pruebas que se realizaron al sistema, desde el funcionamiento en la parte de la interfaz donde el usuario interactúa hasta las pruebas en la lógica del sistema junto con sus respectivas características y detalles asociados a cada prueba.
- A partir del formulario se obtuvieron 38 respuestas que fueron tomados en cuenta para rectificar que el proyecto sea lo idóneo según lo propuesto, a partir de esas respuestas el 70% considera que el sistema es visualmente atractivo, además, se constata que el aplicativo es intuitivo, aunque se debe mejorar aspectos de usabilidad. A su vez, el 84,2% considera que el sistema es beneficioso para la comunidad sorda y a su vez el 15,8% considera que tal vez lo puede ser, teniendo en cuenta esto último nos percatamos que el sistema cumple una de las metas principales y es poder beneficiar a la población sorda del país, sin embargo, se preguntó si se percibió algún error en la demostración del sistema, y un 15.8% respondió que sí, y por el otro lado un 84.2% no percibió ningún error, en base a esto, se recibieron comentarios sobre los diferentes errores que se encontraron, estos se toman a consideración para mejorar el aplicativo, como lo puede ser proveer mayor información en cuanto a la descripción de aspectos técnicos, como lo es la definición de una época y porque se utiliza, que significa la precisión de un modelo, entre otros conceptos. También los encuestados se percataron de errores en cuanto a los ciertos aspectos visuales del sistema, esto respecto a lograr una interfaz más llamativa para el usuario ya sea cambiando la paleta de colores utilizada.

4. Producto final

En esta sección se presentan las pantallas de las diferentes funcionalidades del sistema. Las cuales son:

Registrar Usuario: en esta función el objetivo es registrar usuarios para que este pueda utilizar funciones específicas y exclusivas para usuarios registrados. Como se muestra en la ilustración 25, es necesario ingresar el nombre del usuario, el correo escrito correctamente y una contraseña para luego ser utilizada para la autenticación. Aquí utilizamos una base de datos local para el almacenamiento de estos usuarios, y una vez ingresado y validado todos los campos, aparecerá un anuncio de registrado correctamente.

Ilustración 29 Registro usuario

Iniciar Sesión: esta función tiene como objetivo autenticar a los usuarios registrados, con el fin de mostrar opciones exclusivas para estos. Como se muestra en la ilustración 26, se debe ingresar el correo electrónico y la contraseña. Para que esto funcione, se enviará una consulta de usuario con los parámetros ingresados y se validara si existe este, en caso de que este correcto, se mostrara un mensaje de confirmación de inicio de sesión correcto.

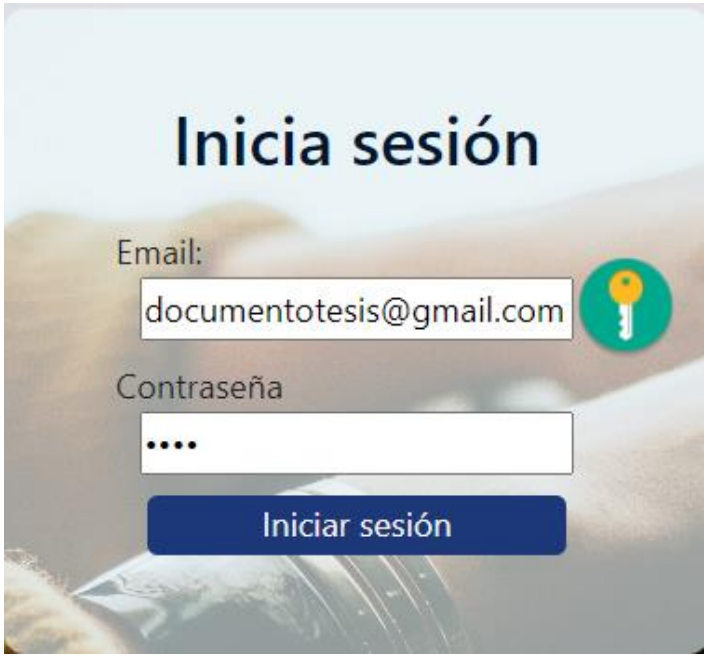

Ilustración 30 Iniciar sesión

Cerrar Sesión: Esta función sirve para cerrar sesión, se muestra un anuncio como la ilustración 27 y el programa deja de mostrar las opciones exclusivas para usuarios autenticados.

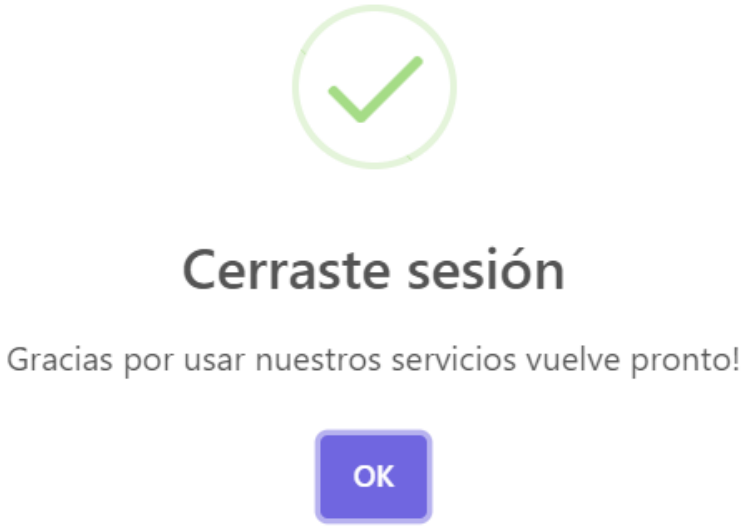

Ilustración 31 Mensaje de cerrar sesión exitosamente

Traducción de Señas: Esta función es una de las soluciones principales del proyecto, la función traducir al ser seleccionada primero pedirá permiso para habilitar la cámara y luego cargará

la librería *mediaPipe*, y a partir de acá, la persona frente a la cámara podrá realizar una seña y debajo se mostrará la traducción a texto, así como se muestra en la ilustración 28, si existe la seña esta imprimirá la seña correspondiente, además si no se detectan las manos, no se imprimirá nada por defecto. Adicionalmente se tienen opciones extras como, por ejemplo, activar audio, que al ser activado este leerá la seña por audio y además leerá la seña cada tres segundos, también tendrá la opción de reproducir audio una vez, que, al ser seleccionado, leerá la última seña realizada. Por último, tendrá un botón que llevará a la opción de ver estadística, que este será documentado más adelante.

Ilustración 32 Traducción de señas

Captura de Señas: esta función sirve para la captura de nuevas señas y posteriormente ser entrenadas, se debe ingresar el nombre de las señas y tomar las fotos, así como se muestra en la ilustración 29. Luego de ser capturadas en el *front-end*, son enviadas como imágenes al *back-end* a través de una petición HTTP. Si se ingresó un nombre y un mínimo de 10 imágenes, esta se guardará correctamente y mostrará un anuncio que se envió correctamente al *back-end*.

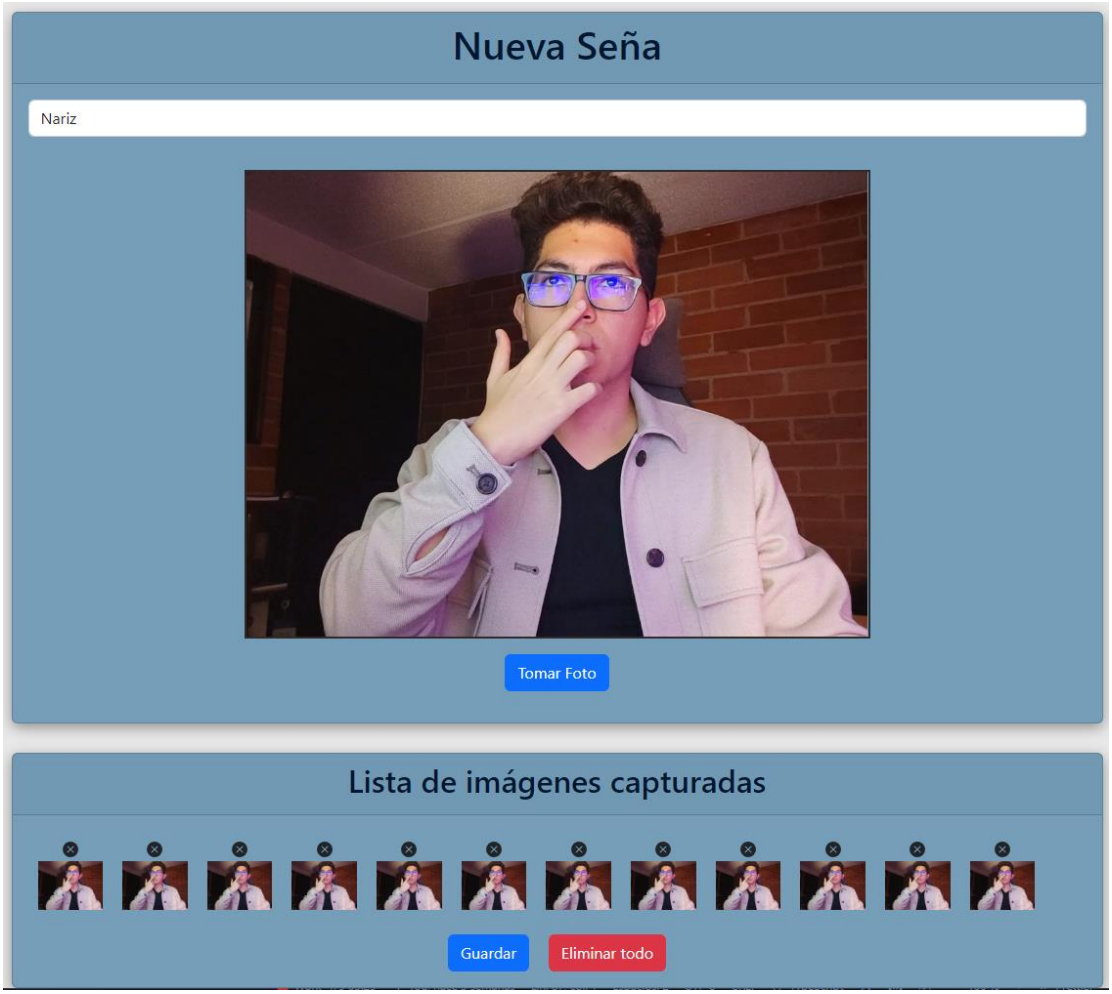

Ilustración 33 Captura de una nueva seña

Entrenamiento: esta función tiene distintas funcionalidades, en primer lugar, se puede comenzar a entrenar el modelo, en caso de que existan señas por entrenar y no esté en proceso de esta y así como se muestra en la ilustración 30, se mostrara el anuncio de que existen señas por entrenar. En caso de que se esté entrenando el modelo, se mostrara un contador como se puede observar en la ilustración 33, de cuanto se demorara aproximadamente el entrenamiento. Por último, en caso de que no haya señas por entrenar, se mostrara un dónde indica que todo está entrenado y no hay nuevas señas.

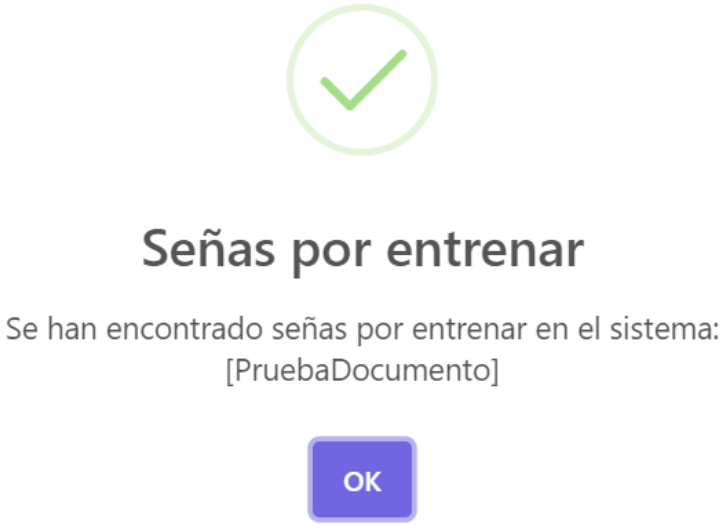

Ilustración 34 Notificación de que existen señas por entrenar

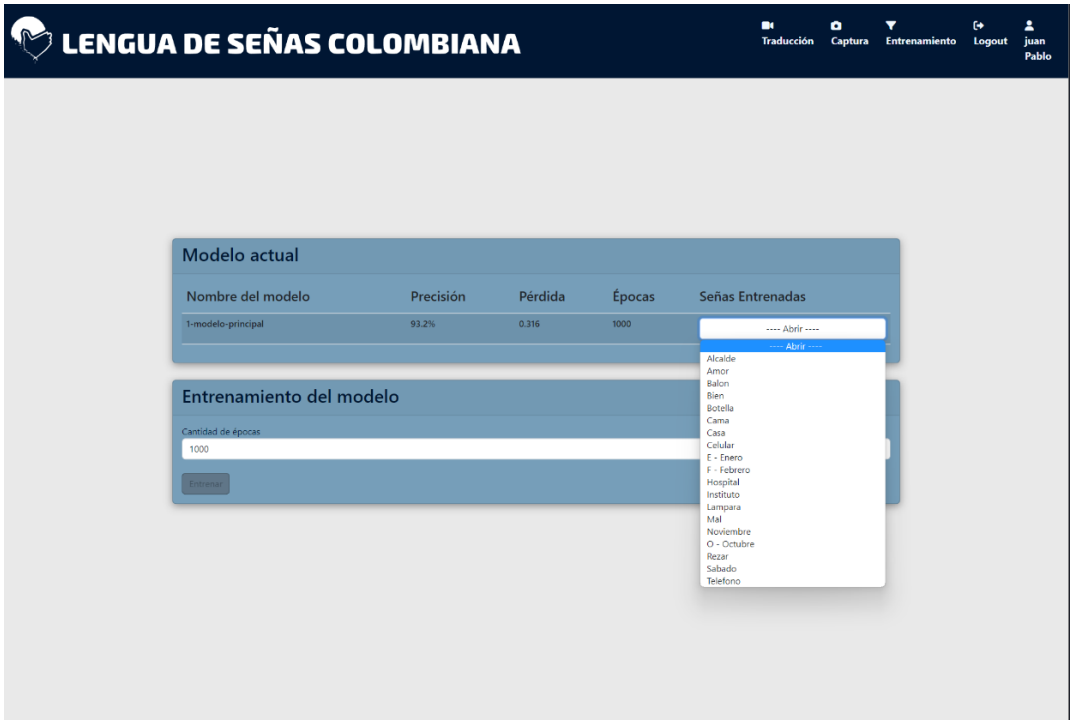

Ilustración 35 Parámetros de entrenamiento

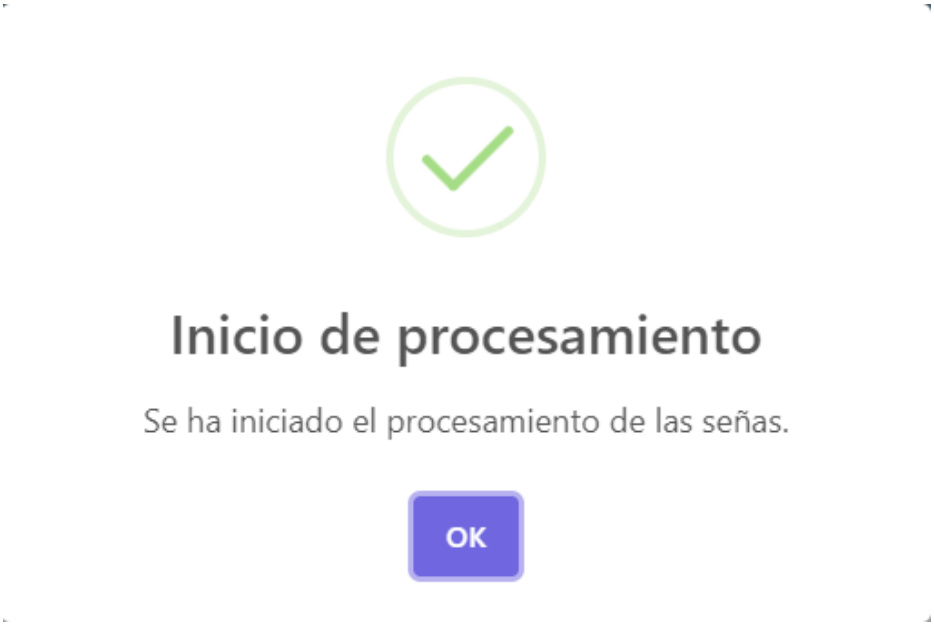

Ilustración 36 Procesamiento iniciado

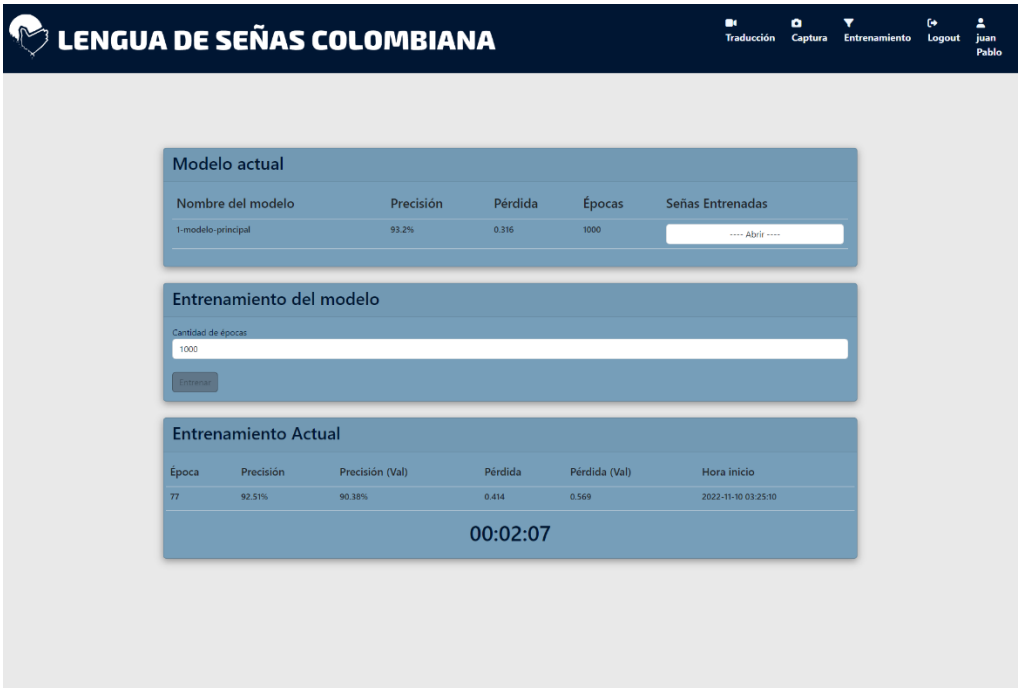

Ilustración 37 Entrenamiento en ejecución

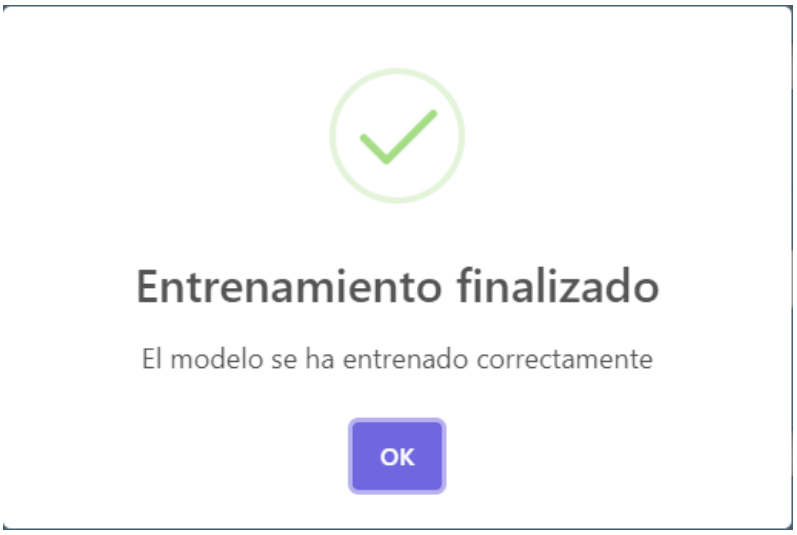

Ilustración 38 Entrenamiento finalizado

Estadísticas: esta función es el producto del llamado de la función a ver estadísticas en el componente de traducción, este funciona si desde que se inició la traducción existen por lo menos dos señas realizadas, una vez realizadas se habilita el botón de ver estadísticas, y se mostrara una gráfica donde compara el promedio del porcentaje de la precisión de cada seña junto al nombre de la seña realizada como se muestra en la ilustración 35.

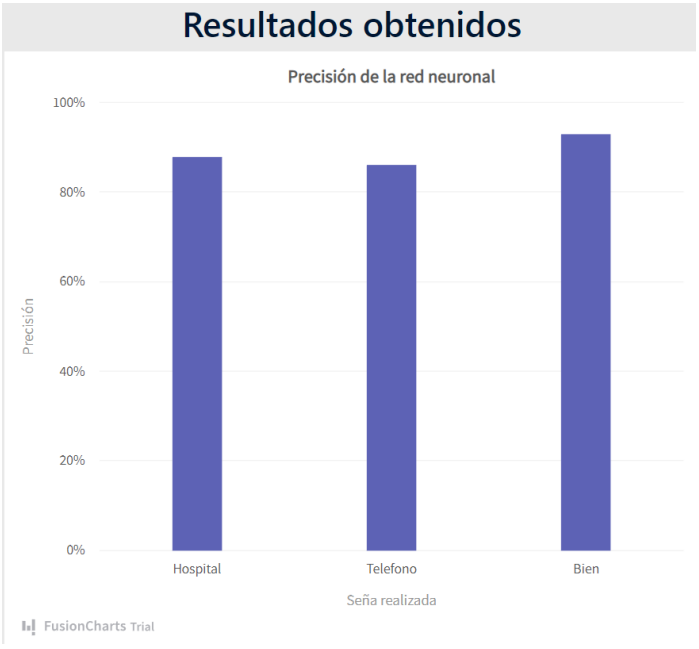

Ilustración 39 Estadística de la precisión de los resultados obtenidos

VII- RESULTADOS

1. Precisión del modelo

Para obtener la mejor precisión la cual el modelo puede llegar, se realizan distintos ajustes, ya sea en cuanto a las diferentes transformaciones, épocas utilizadas, funciones de perdida, funciones de regularización, capas, cantidad de neuronas por capa e hiperparámetros de las capas. A continuación, se presenta más a detalle los cambios realizados en cada iteración al modelo.

Iteración	Cambios realizados	Precisión obtenida
1	- Se hizo un sistema con las coordenadas de la pose y las manos con una red neuronal con 128 neuronas densas, es decir, todas están conectadas entre sí y con una entrada de coordenadas de una seña de 300 neuronas. - Se descartaron las imágenes en las que *mediapipe* no detectaba ninguna mano (si se elegia si detectaba si quiera una mano). - Las fotos utilizadas están sin aumento de datos.	80%
2	Dada la estocasticidad de los modelos, se ejecuta nuevamente el modelo sin realizar cambios, el cual nos arrojó una precisión más alta.	84.9%
3	Se agrego un módulo de aumento de datos en donde se aplicó una rotación aleatoria de 10 grados a las fotos y a todas las fotos se les generó un giro horizontal para simular las señas zurdas.	93.2%
4	- Se agrego un módulo de aumento de datos en donde se aplicó una rotación aleatoria de 10 grados a las fotos. - Se definen 100 épocas, las cuales son iteraciones completas sobre las muestras, es decir el conjunto de datos del entrenamiento.	85.8%
5	- Se agrego un módulo de aumento de datos en donde se aplicó una rotación aleatoria de 10 grados a las fotos y a todas las fotos se les generó un giro horizontal para simular las señas zurdas. - A diferencia del resultado anterior se agregó una capa de eliminación del 0.2 para evitar el sobreajuste.	87%

6	- A diferencia del resultado anterior se subió la eliminación a 0.4 pero no se obtuvo un mejor resultado, por lo menos con 200 épocas.	86%
7	- A diferencia del resultado anterior se bajó la eliminación de 0.4 a 0.3 para probar si se mejoraba el resultado. - Se ejecutó el modelo con 1000 épocas.	90%
8	- Se ejecutó el modelo con 2000 épocas. - Se hizo cambios en las coordenadas quitando algunos parámetros de visibilidad innecesarios (la mano izquierda y derecha). - Se normalizó las coordenadas de 0 a 1 para que tengan mejor rendimiento en la función de activación relu. - Por los cambios de las coordenadas el input cambió de 300 (sin cara) a 226 (sin cara).	92.2%
9	Se cambió la función de pérdida de error cuadrático medio a la entropía categórica.	93.3%
10	- Se cambió la función de pérdida de error cuadrático medio a la entropía categórica. (Ya se probó entropía categórica escasa, pero daba errores).- Se agregó regularización, en la tercera capa (Dense (64)), de tipo [L2 - Regularización Ridge][30] para evitar que el modelo se sobreajuste y se pueda reducir el espacio entre la precisión y pérdida de los datos de entrenamiento y validación. - Se decidió usar L2 y no L1 o L1_L2 porque dada la naturaleza de nuestro problema queremos buscar correlaciones entre los datos, y no eliminar datos irrelevantes, ya que ese proceso lo hacemos en etapas anteriores (los datos son las coordenadas de los puntos de referencia de la pose y las manos). - Cabe aclarar que por este motivo las gráficas van a presentar un fenómeno que se ve que la línea de los datos de validación en las primeras épocas va a estar superando a las de entrenamiento; esto se debe a que se aplica la regularización a los datos de entrenamiento	93.2%

	y no a los de validación, pero no es un error, simplemente es un fenómeno que se da por la regularización [31].	

Ilustración 40 Precisión de los modelos

2. Pruebas del sistema

Todas las pruebas realizadas al sistema se encuentran expuestos en el documento de plan de pruebas en el cual se detallará cada una de ellas, a continuación, se detallarán algunas de ellas.

Pruebas unitarias *back-end*.

Las pruebas unitarias se realizaron mediante la librería *Pytest*, con un total de 26 pruebas enfocadas en gran medida a los *endpoints* del sistema.

Para entender de forma sintetizada en que consiste esta prueba, se empieza con declarar lo que sería un falso token "*fake_secret_token*", el *mock* tiene como intención simular el objeto que recibe la petición donde esta contiene lo que sería el nombre de la seña y el arreglo de imágenes, se realiza toda la configuración de la ruta donde se condiciona que, si no se tiene el token, se devuelve un error que significa que el usuario no tiene autorización, a su vez si la seña no tiene nombre, se devuelve como respuesta que la seña está vacía, por ultimo si el objeto *mock* es igual al de la petición , se muestra con éxito que la seña se creó correctamente.

```python
fake_secret_token = "secret"

mock_db = {"1": RequestSignal(name="name1", images=["images1"])}

router = APIRouter()


@router.post("/new-signal")
async def save_signal(signal: RequestSignal, req: Request):
    if req.headers["Authorization"] != fake_secret_token:
        raise HTTPException(status_code=401, detail="Unauthorized")
    if signal.name == "":
        raise HTTPException(status_code=422, detail="El nombre de la señal está vacío")
    mock_db["2"] = signal
    return {"message": "Señal creada correctamente", "result": signal}
```

Ilustración 41 Prueba endpoint crear una nueva seña

Pruebas Unitarias *Front-End*

Para estas pruebas unitarias de Angular se realizaron mediante la utilización de *Jasmine* y *Karma*, con un total de 44 Pruebas tanto enfocadas a Componentes como a Servicios.

Ilustración 42 Pruebas unitarias Front-End

Prueba de componentes:

Las pruebas unitarias orientadas a componentes tienen como finalidad probar las variables, funciones, corroborar que existan ciertas características visuales del *HTML*, que deban existir al ejecutarse.

En esta prueba se simula que la variable captures que es el arreglo donde se guardan las imágenes de una nueva Seña contiene tres imágenes, con Jasmine esperamos que el arreglo tenga un tamaño de tres, después utilizamos la función del componente llamado *eliminarTodo()*. Para que finalice con éxito esta prueba después de ejecutarse la función se espera que el tamaño del arreglo sea cero.

frontend - new-signal.component.spec.ts

```
it('Comprobar la función de vaciar arreglo de fotos capturadas', () => {
    const fixture = TestBed.createComponent(NewSignalComponent);
    const app = fixture.componentInstance;
    fixture.detectChanges();
    app.captures.push('foto1');
    app.captures.push('foto2');
    app.captures.push('foto3');
    expect(app.captures.length).toBe(3);
    app.deleteAll();
    expect(app.captures.length).toBe(0);
});
```

Ilustración 43 Prueba de componentes

Prueba de Servicios:

Las pruebas enfocadas en servicios son importantes para simular las interacciones que tiene el Front, para ello es necesario utilizar lo que sería un objeto *SPY* que puede simular las peticiones *HTTP (GET, POST)*. Para este caso vamos a Simular la creación de una nueva seña es por ello que se debe crear el objeto que se debe enviar en el servicio y lo que esperamos que nos devuelve el servicio. Se llama el servicio *crearSena()* que necesita como parámetros el arreglo de imágenes junto con el nombre de la seña, para que finalice con éxito la prueba se espera que el resultado que nos devuelve sea el que esperamos.

frontend - signal.service.spec.ts

```
it('Comprobar creación nueva seña', (done: DoneFn) => {
    const signal = {
        name: 'Casa',
        images: ['foto1', 'foto2', 'foto3'],
    };
    const resultado = ['Seña creada correctamente'];
    httpClientSpy.post.and.returnValue(of(resultado));

    const { name, images } = signal;
    service.createSignal(images, name).subscribe((res) => {
        expect(res).toEqual(resultado);
        done();
    });
});
```

Ilustración 44 Prueba de servicios

3. Resultados propuestos

El objetivo del proyecto era poder realizar un sistema capaz de entrenar un modelo de clasificación de lenguaje de señas, y poder traducirlas a través de una cámara en vivo, además de que este pueda ser mejorado y continuado a futuro.

A continuación, se presenta una tabla donde se mostrará cada elemento del alcance del proyecto, si este elemento fue propuesto previamente, la relevancia que tiene este elemento y, por último, el nivel de satisfacción con la que se logró este.

Elementos del alcance de proyecto	Propuestos	Relevancia	Nivel de satisfacción
Captura de seña y guardado	Si	Alta	Deseado
Agregar seña nueva al conjunto de datos	Si	Alta	Deseado
Inicio de sesión	Si	Alta	Deseado
Registro de usuario	Si	Alta	Deseado
Definición de roles de usuario	Si	Baja	Aceptable
Traducir seña	Si	Alta	Deseado
Lectura de resultado de seña por conversión de texto a voz	Si	Media	Deseado
Grafica estadística de la precisión de las señas	No	Media	Deseado
Cambiar parámetros de entrenamiento	No	Media	Deseado
Visualización de entrenamiento	Si	Media	Deseado
Manual de usuario	No	Media	Aceptable

Tabla 12 Resultados propuestos

En conclusión, todos los elementos propuestos fueron cumplidos de manera exitosa, e inclusivamente se agregaron elementos no propuestos previamente.

VIII- CONCLUSIONES

1. Análisis de impacto del proyecto

Los resultados del proyecto afectarían a la comunidad de sordos de manera positiva, ya que esto les dará la posibilidad y la herramienta de poder comunicarse de manera más sencilla con personas que no conocen el lenguaje. Además, si se continua con el proyecto a largo y mediano plazo, se podría llegar a realizar un sistema robusto, en donde sirva para incluir todo el diccionario de señas colombiano, inclusive se podría expandir a otros lenguajes de señas de otros países.

Adicionalmente, se podría motivar a las universidades, a promover a los estudiantes realizar más proyectos relacionados con la comunidad sorda e inclusive llegar más lejos y motivar a estas mismas universidades a crear materias electivas relacionadas con este lenguaje. También se puede llevar a incentivar a empresas, organizaciones y entidades a crear empleos que puedan incluir más a la comunidad sorda ya que esta condición no debería ser un limitante.

Entonces lo que se busca con este proyecto es quitar esa barrera y esa limitación, para que no solo la comunidad sorda pueda comunicarse con mayor facilidad, si no que sean incluidos en más actividades, empleos, entre otros.

2. Conclusiones y trabajo futuro

Los objetivos planteados fueron logrados debido a que el mayor reto fue realizar un sistema capaz de traducir señas estáticas a través del entrenamiento de un modelo de clasificación por medio de coordenadas, fue logrado con éxito, este éxito se debe a la buena planeación que tuvo el equipo durante el desarrollo del proyecto en el que al tener todo planificado con las actividades y tareas a realizar, hizo que la realización de este fuera más ameno, a su vez, al tener opiniones de otras personas tanto expertos en la lengua de señas colombiana como personas del común hizo que expandiéramos nuestros horizontes en cuanto a las sugerencias que estos nos daban y uno de los mayores aspectos que pudimos concluir es que este sistema más allá de ser un programa que traduce señas, es un sistema que establece las bases de algo que pueda beneficiar a la población sorda haciendo que la brecha de exclusión se vea altamente reducida y a su vez, aumentando su integración en la sociedad colombiana, es por eso que en un futuro se recomienda poder realizar diferentes avances y ampliaciones al sistema tales como, ampliar la base de datos ingresando más señas, se podría realizar un ajuste para que se cree un modelo de entrenamiento por país y así expandir este proyecto mundialmente. También hacer modificaciones al producto para que se puedan incluir y agregar señas que presenten movimiento, ya que una gran parte del diccionario de lenguaje de señas incluyen movimientos.

Adicionalmente se podría agregar una nueva funcionalidad, en la cual consistiría en no solo detectar y clasificar las manos en las señas, sino que también se podrían detectar las expresiones faciales, ya que, de igual manera, el sistema almacena internamente las coordenadas de la cara, por lo que a futuro se podría agregar estas al entrenamiento del modelo. Adicionalmente, se podría crear una API publica para el uso del servicio para ser utilizada por cualquier desarrollador o empresa.

Otras de las recomendaciones para trabajo futuro serian poder hacer una buena implementación de roles para los usuarios, con el fin de mejorar la seguridad y la estabilidad del sistema utilizando privilegios. Por último, se recomienda poder hacer la migración del sistema a un servicio de la nube y que este sea asequible 24 horas al día, ya que este no puede ser accedido por todo el mundo, debido a restricciones institucionales donde se alojó el proyecto.

Para que este proyecto pueda ser llevado a un uso cotidiano se debe subir el servidor backend a un servidor público y permitir que el frontend se adapte a cualquier dispositivo, en especial a aplicativo móvil, los cuáles permitirán hacer una traducción en tiempo real en cualquier lugar al tener una comunicación con una persona de la población sorda. A su vez es necesario permitir que la red neuronal desarrollada se mejore para interpretar señas dinámicas, pero principalmente que pueda realizar las relaciones entre las señas en el tiempo para mantener coherencia de una "oración" y así poder realizar traducciones de conversaciones completas de una manera más orgánica.

Como potenciales usos podría llevarse a cabo una implementación del sistema en entidades como lo son bancos, sector de salud, estatales; como lo son notarias, entre otros. Estas entidades pueden ayudar de manera significativa a la comunidad sorda equiparando las brechas sociales como comunicativas que existen actualmente.

Otro potencial uso puede ser en entornos de educación teniendo inclusión de personas de esta población, pero también poder generar aplicativos alternos encargados de enseñar a los demás estudiantes la lengua de señas para que se haga conciencia y se masifique el aprendizaje de esta lengua poco conocida.

Por último, podemos ver que este aplicativo tiene el potencial de expandirse de manera internacional usando los diferentes lenguajes de señas de cada país en todo el mundo haciendo un sistema unificado de traducción para uso público.

IX- REFERENCIAS

[1] Insor, "BOLETÍN OBSERVATORIO SOCIAL POBLACIÓN SORDA COLOMBIANA," 2011, Accessed: Nov. 08, 2022. [Online]. Available: https://www.insor.gov.co/home/wp-content/uploads/filebase/publicaciones/boletin_observatorio02.pdf

[2] W. Leal Pérez, "P.L.303-2021C (POBLACIÓN SORDA)", Accessed: Nov. 08, 2022. [Online]. Available: https://www.camara.gov.co/sites/default/files/2021-09/P.L.303-2021C%20%28POBLACI%C3%93N%20SORDA%29.docx

[3] "ISO/IEC/IEEE 16326:2009 Systems and software engineering — Life cycle processes — Project management," 2009. https://www.iso.org/standard/41977.html (accessed Nov. 08, 2022).

[4] "Especificación de Requisitos según el estándar de IEEE 830," 2008. https://www.fdi.ucm.es/profesor/gmendez/docs/is0809/ieee830.pdf (accessed Nov. 08, 2022).

[5] "IEEE Standard for Information Technology--Systems Design--Software Design Descriptions," 2009, Accessed: Nov. 08, 2022. [Online]. Available: https://standards.ieee.org/ieee/1016/4502/

[6] "IEEE Standard for Software Quality Assurance Processes," 2014. https://standards.ieee.org/ieee/730/5284/ (accessed Nov. 08, 2022).

[7] "ISO/IEC/IEEE 29119-3:2013 Software and systems engineering — Software testing — Part 3: Test documentation," 2013, Accessed: Nov. 08, 2022. [Online]. Available: https://www.iso.org/standard/56737.html

[8] "ISO/IEC/IEEE Systems and software engineering – Requirements for acquirers and suppliers of user documentation," 2011.

[9] A. Oviedo, "Colombia, atlas sordo," *Cultura Sorda*, 2015. https://culturasorda.org/colombia-atlas-sordo/ (accessed Oct. 31, 2022).

[10] H. Mejía, *Lenguaje manual Colombiano. primer nivel*. Bogota, Bogotá D.C: Federación Nacional de Sordos de Colombia, 1995.

[11] T. Alameda, "'Machine learning': ¿qué es y cómo funciona?," *BBVA*, Nov. 08, 2019. https://www.bbva.com/es/machine-learning-que-es-y-como-funciona/ (accessed Oct. 31, 2022).

[12] "Generación de datos artificiales (Data Augmentation)." https://franspg.dev/2020/01/27/generacion-de-datos-artificiales-data-augmentation/ (accessed Oct. 31, 2022).

[13] Amazon, "¿Qué es una red neuronal?," *AWS*. https://aws.amazon.com/es/what-is/neural-network/ (accessed Oct. 31, 2022).

[14] N. Joshi, "How to fine-tune your artificial intelligence algorithms," Jan. 14, 2020. https://www.allerin.com/blog/how-to-fine-tune-your-artificial-intelligence-algorithms (accessed Oct. 31, 2022).

[15] DW, "MIVOS, traductor para sordomudos," *Economia Creativa*, Apr. 05, 2020. https://www.dw.com/es/mivos-traductor-para-sordomudos/av-42523294 (accessed Oct. 31, 2022).

[16] "Voz y Señas, traductor LSM," *Voz y Señas*, 2018. https://www.vozysenas.com/ (accessed Oct. 31, 2022).

[17] F. Mejía, "Conoce Showleap, el traductor de lengua de señas en tiempo real," Apr. 02, 2019. https://www.enter.co/chips-bits/apps-software/conoce-showleap-el-traductor-de-lengua-de-senas-en-tiempo-real/ (accessed Oct. 31, 2022).

[18] L. F. Moreno López and Y. P. Muñoz Reina, "Implementación de un algoritmo para la clasificación automática de lenguaje de señas colombiano en video usando aprendizaje profundo," 2020, [Online]. Available: https://repository.ucatolica.edu.co/handle/10983/24980

[19] G. A. Realpe Fresneda, "Reconocimiento del lenguaje de señas manuales con el Kinect," 2013. Accessed: Oct. 31, 2022. [Online]. Available: https://repositorio.uniandes.edu.co/bitstream/handle/1992/12206/u671097.pdf?sequence=

[20] J. A. MERIÑO GUZMAN and D. GARIZABALO PEDROZO, "Diseño de un guante electrónico para la interpretación y traducción del lenguaje de señas en personas con discapacidad auditiva mediante tecnología arduino e interfaz de visualización por medio de una aplicacion en android." Accessed: Oct. 31, 2022. [Online]. Available: http://repositorio.uts.edu.co:8080/xmlui/handle/123456789/3302

[21] A. Mendes and R. Ferreira, "Flask vs FastAPI: what's better for app development?," Jul. 04, 2022. https://www.imaginarycloud.com/blog/flask-vs-fastapi/#fastapipros (accessed Nov. 08, 2022).

[22] A. Bäuerle, C. van Onzenoodt, and T. Ropinski, "Net2Vis – A Visual Grammar for Automatically Generating Publication-Tailored CNN Architecture Visualizations," *IEEE Trans Vis Comput Graph*, vol. 27, no. 6, pp. 2980–2991, 2021, doi: 10.1109/TVCG.2021.3057483.

[23] J. Larkin Alonso, "¿Qué es TensorFlow y para qué sirve?," *Incentro*, Jun. 15, 2022. https://www.incentro.com/es-ES/blog/que-es-tensorflow (accessed Oct. 31, 2022).

[24] "Lenguajes de programación para machine learning," 2022. https://aprendeia.com/lenguajes-de-programacion-para-machine-learning/#:~:text=Python%20lidera%20en%20lenguajes%20de,son%20nuevos%20en%20Machine%20Learning (accessed Oct. 31, 2022).

[25] M. J. Gonçalves, "¿Qué es Angular y para qué sirve?," *Hiberus blog*, Oct. 13, 2021. https://www.hiberus.com/crecemos-contigo/que-es-angular-y-para-que-sirve/ (accessed Oct. 31, 2022).

[26] G. Solano, "Como instalar MEDIAPIPE | Python," *Omes*, May 13, 2021. https://omes-va.com/como-instalar-mediapipe-python/#:~:text=Este%20es%20un%20framework%20multimodal,escritorio%20o%20en%20la%20web (accessed Oct. 31, 2022).

[27] Amazon, "¿Qué es Docker?," *AWS*, 2022. https://aws.amazon.com/es/docker/ (accessed Oct. 31, 2022).

[28] D. Camacho, "Qué es GitHub y cómo usarlo para aprovechar sus beneficios," *Platzi*, 2021. https://platzi.com/blog/que-es-github-como-funciona/ (accessed Oct. 31, 2022).

[29] S. Memo, "Déjame Escuchar Hablar a Tu Navegador: Usar la API Synthesis Speech," *Tutsplus*, Nov. 12, 2015. https://code.tutsplus.com/es/tutorials/let-me-hear-your-browser-talk-using-the-speech-synthesis-api--cms-24971 (accessed Nov. 08, 2022).

[30] J. Martinez Heras, "Regularización Lasso L1, Ridge L2 y ElasticNet," *IArtificial.net*, Sep. 19, 2020. https://www.iartificial.net/regularizacion-lasso-l1-ridge-l2-y-elasticnet/ (accessed Nov. 09, 2022).

[31] A. Rosebrock, "Why is my validation loss lower than my training loss?," *Pyimagesearch*, Oct. 14, 2019. https://pyimagesearch.com/2019/10/14/why-is-my-validation-loss-lower-than-my-training-loss/ (accessed Nov. 09, 2022).

X- APENDICES

Anexo 1. Project Managing Plan (PMP).

Anexo 2. Software Requirement Specification (SRS).

Anexo 3. Software Design Document (SDD).

Anexo 4. Documento de plan de pruebas.

Anexo 5. Manual de usuario.

Anexo 6. Plan de pruebas.

Anexo 7. Propuesta de trabajo de grado.