DOCUMENTO DEL DISEÑO DEL SOFTWARE

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

Tabla de contenido

Lista de ilustraciones

1. INTRODUCCIÓN

En el presente documento se describe el diseño del software desarrollado como parte del trabajo de grado *Aprendizaje automático de lengua de señas colombiana.* La documentación proporcionada tiene como objetivo explicar y justificar las decisiones de diseño tomadas por el equipo durante el desarrollo del sistema. En este sentido se presentarán distintos diagramas que explican a detalle el funcionamiento del módulo en diferentes niveles de abstracción y tiempo tales como el diagrama de despliegue, componentes, paquetes, clases, estados, BPMN que representan los principales procesos del sistema, asimismo, la forma de persistir los datos del sistema.

2. ARQUITECTURA DEL SISTEMA

2.1 Vista lógica del sistema

En este apartado se abarcará el diagrama de componentes que representa el funcionamiento del sistema, activos arquitecturales como es la en este caso la base de datos TinyDB, se denotan los protocolos requeridos para la correcta comunicación que son HTTP y por medio de sockets. Se detalla el dominio del Sistema donde nos permite tener una coherencia en la creación de datos y que esta sea persistente en la base de datos.

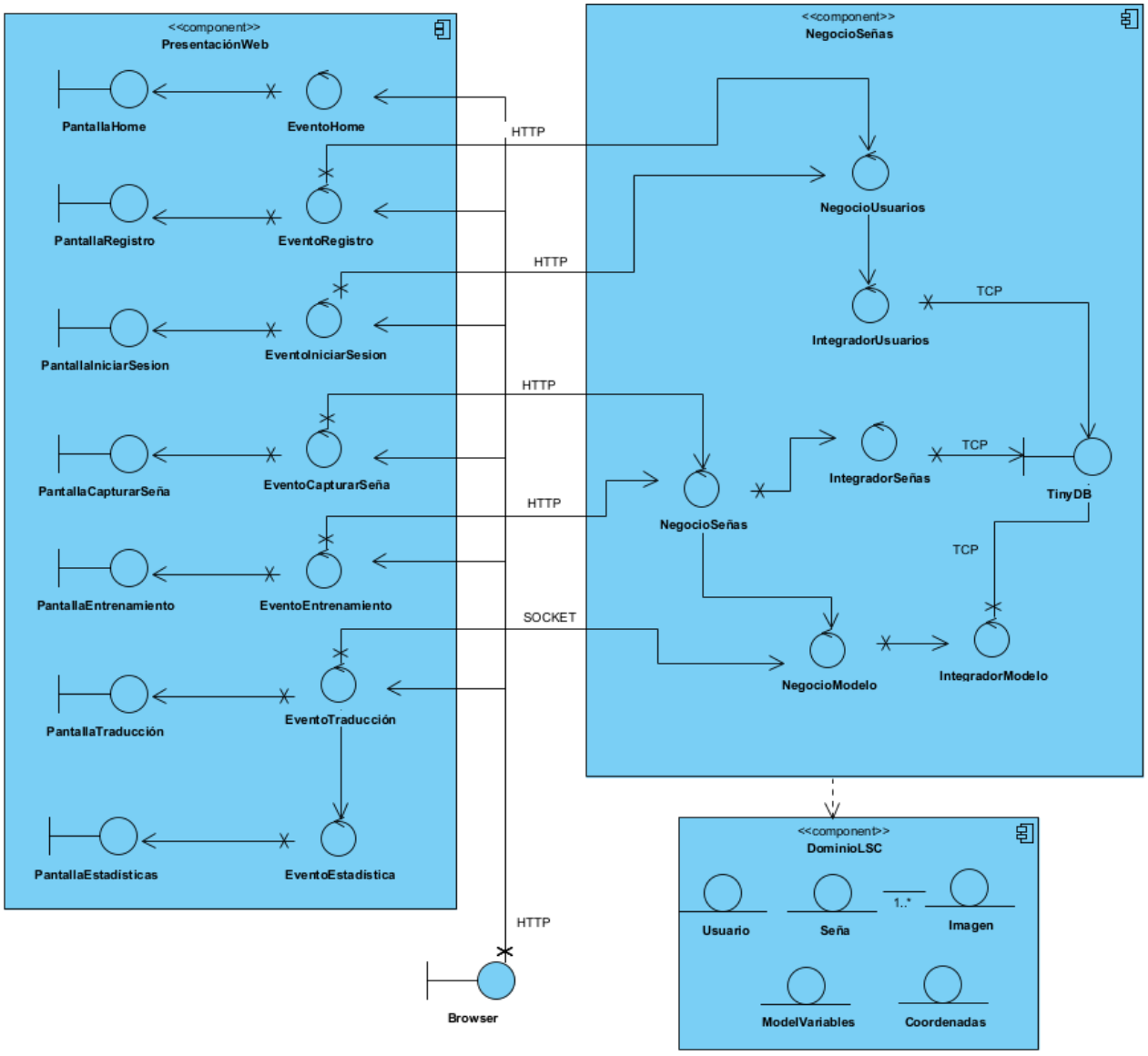

Ilustración 1 Diagrama de componentes del sistema

- Front-end

Para definir la arquitectura se estableció un diagrama de estados el cual resume el flujo de actividades que pueden ocurrir en el sistema y como debe comportarse frente a las acciones que se tomen, así como se muestra en la siguiente ilustración

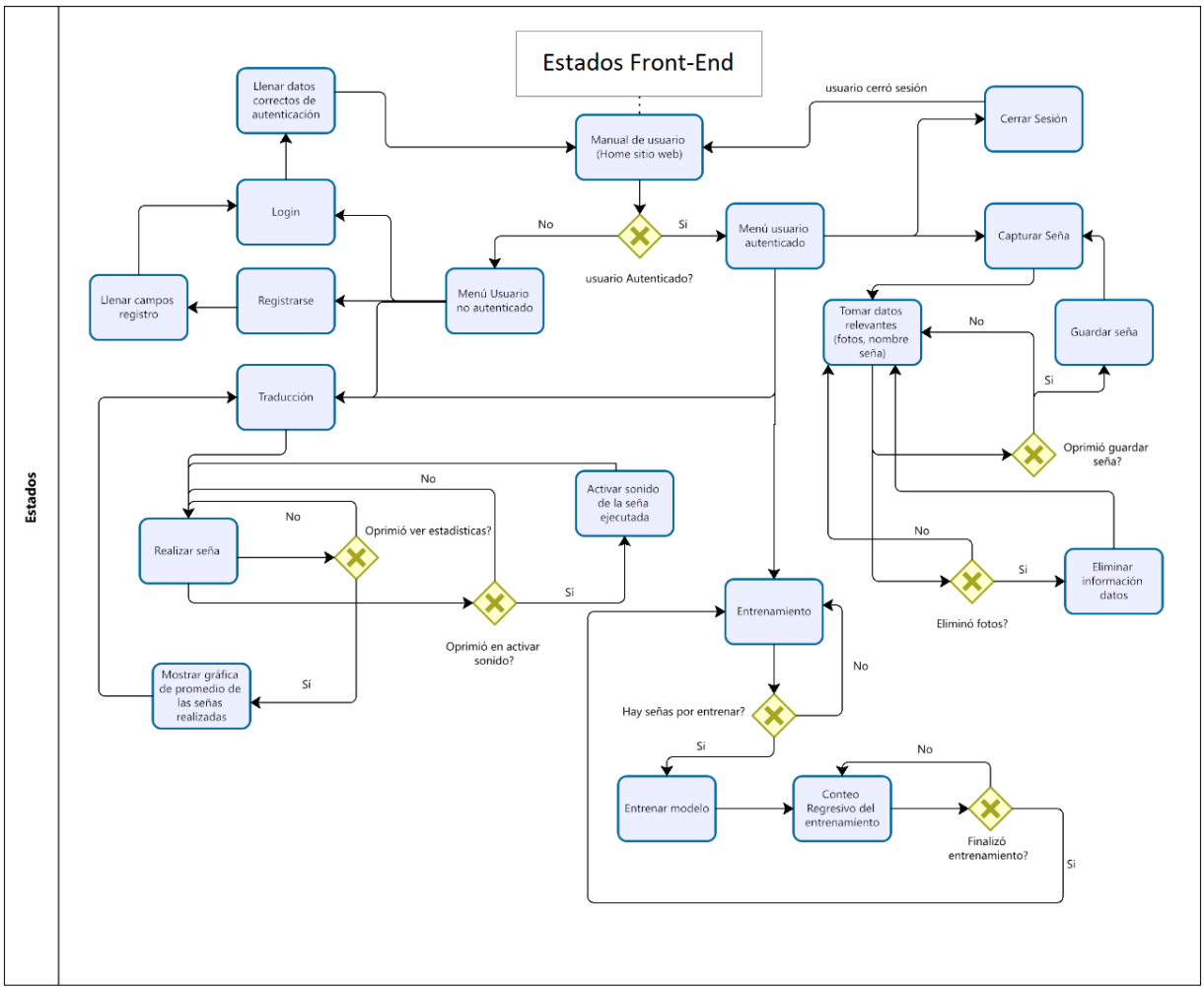

Ilustración 2 Diagrama de estados front-end del sistema

Para comprender mejor el componente de presentación web, 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).

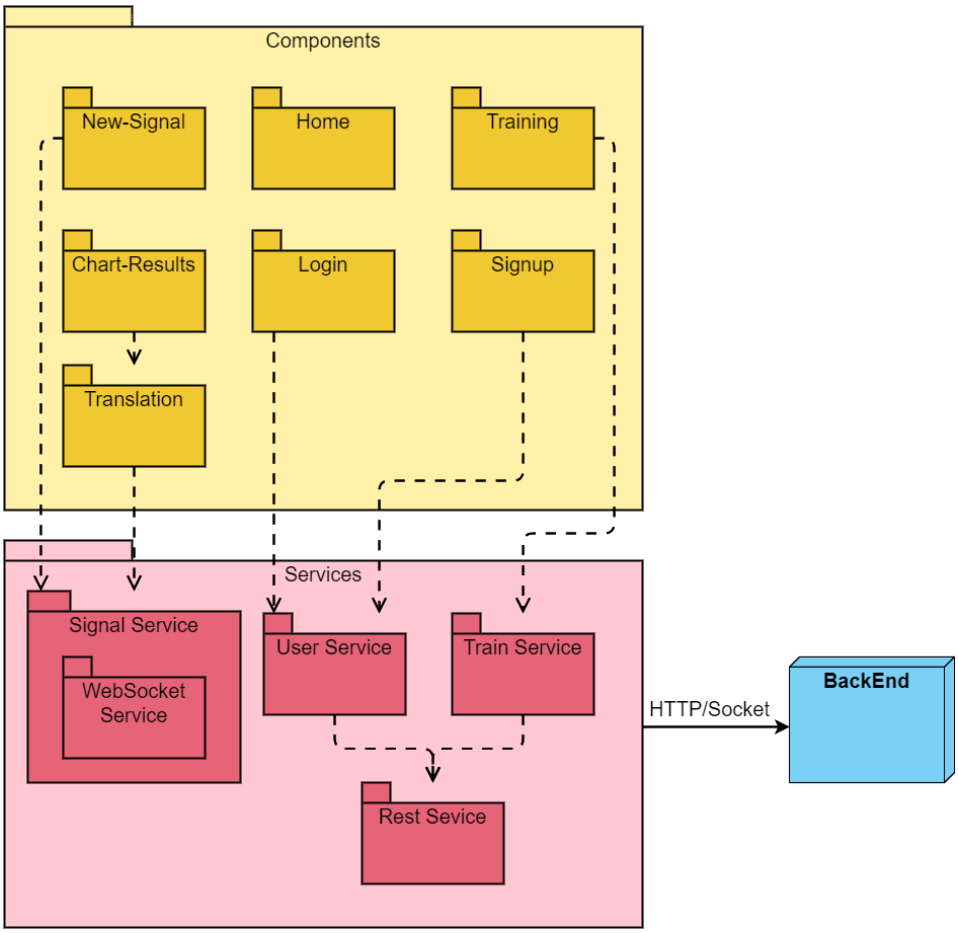

Ilustración 3 Diagrama de paquetes front-end

- Back-end

El diagrama de paquetes del *back-end* se explicará primero por medio del paquete *Routes* (paquete con el colore verde), este contiene las rutas donde el *back-end* recibirá las peticiones del usuario y se las delegará al paquete *Controllers*.

- **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.

Para entender de mejor manera el paquete *Controllers* se explicará el paquete *Repositories* (paquete de color amarillo), este contiene tres paquetes que son *Signal*, *Model* y *User repository*, donde estos consultan a la base de datos *TinyDB* la información que se desea almacenar o leer.

Siguiendo lo anterior, el paquete de *Controllers* (paquete con el color azul oscuro), tendrá la mayor parte de la lógica de negocio, ya que de esta forma podrá hacer uso de funcionalidades que posee el paquete *Utils* que más adelante se describirá el rol que ejerce. Para ello se realizará una breve descripción de la operatividad de cada uno de los *controllers*.

- **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.

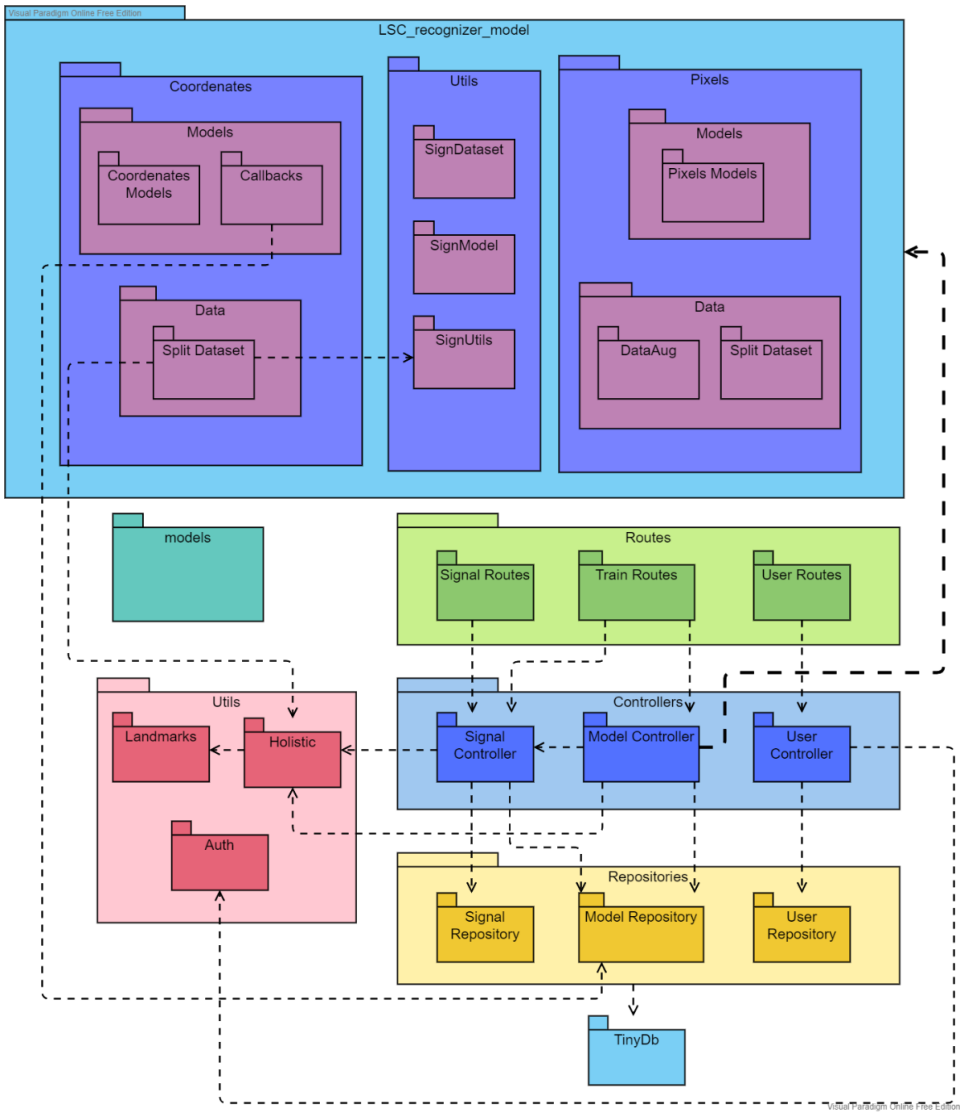

Ilustración 4 Diagrama de paquetes de back-end

2.2 Vista física del sistema

Para la arquitectura del sistema, se mostrará 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,que 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*.

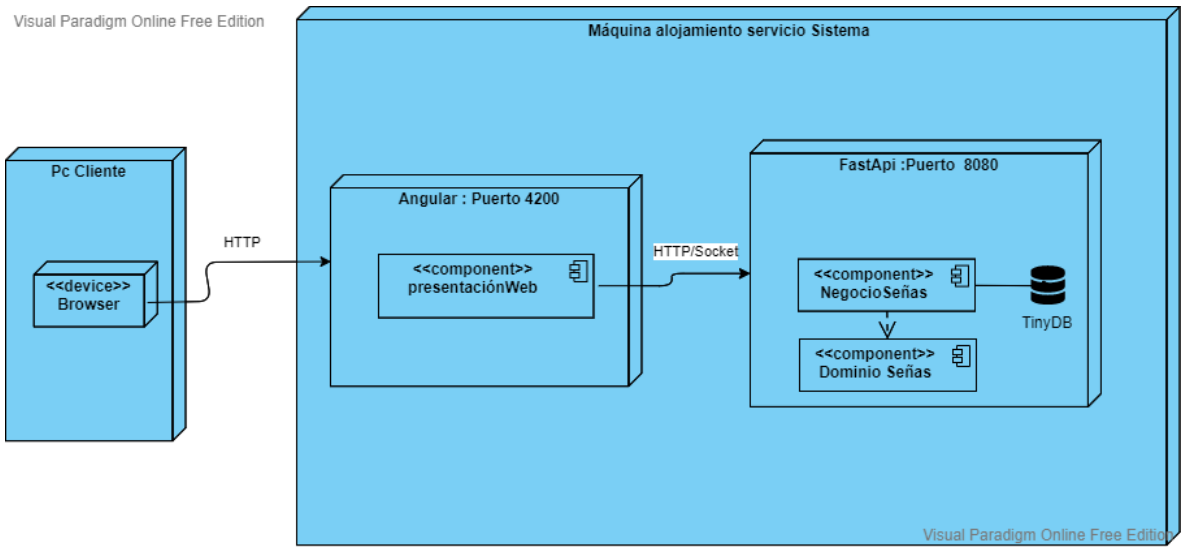

Ilustración 5 Diagrama de despliegue del sistema

2.3 Vista de procesos 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.

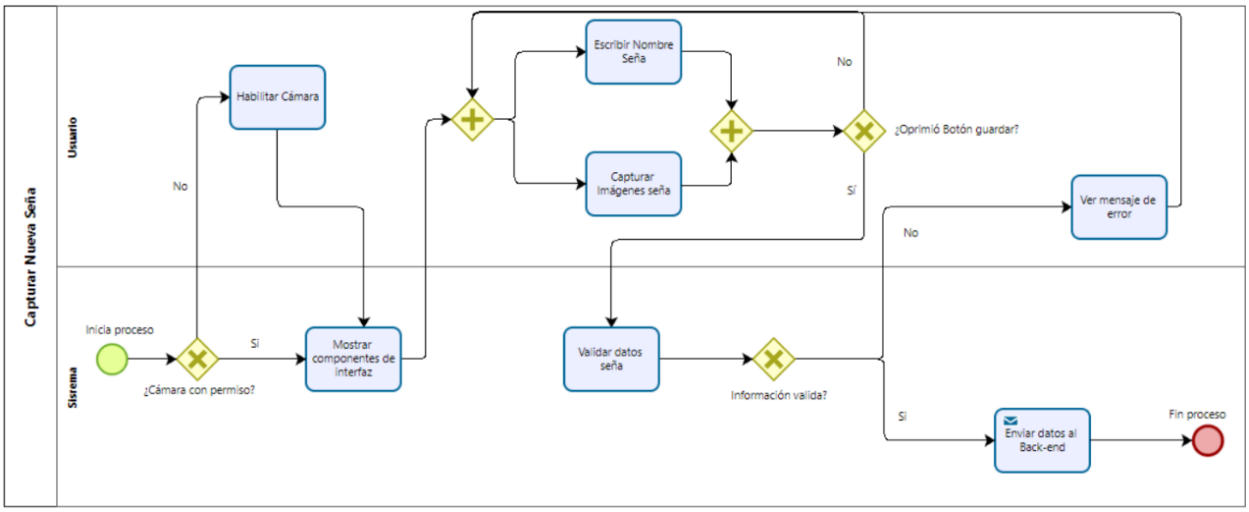

Ilustración 6 Proceso de captura de nueva seña

- **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 mostrara 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 terminara el proceso.

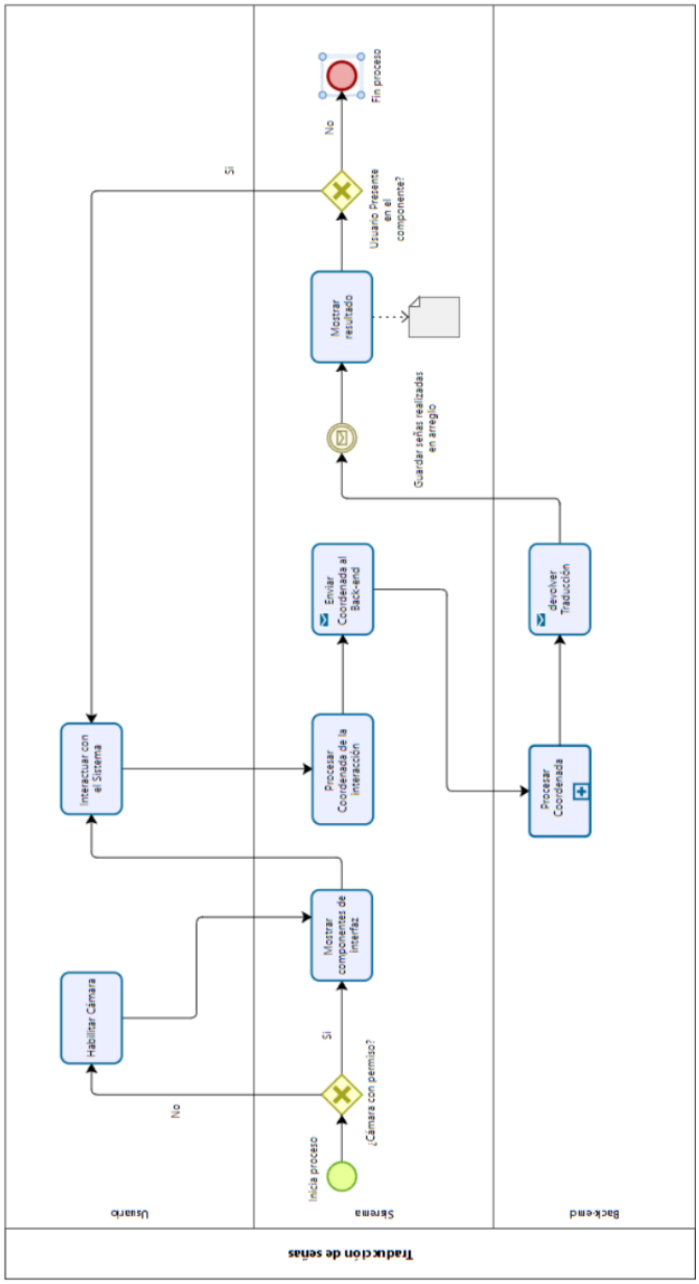

Ilustración 7 Proceso de traducción de seña

- **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 mostrara un diagrama de barras verticales con esta información.

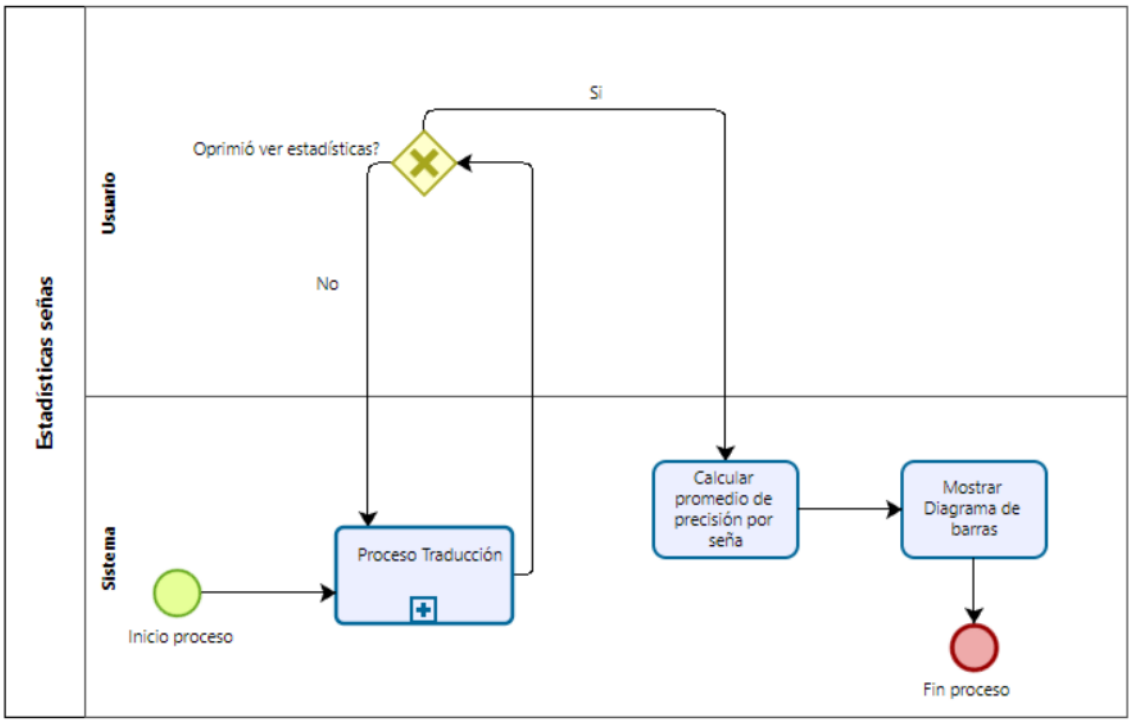

Ilustración 8 Proceso de visualización de estadística por seña

- **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.

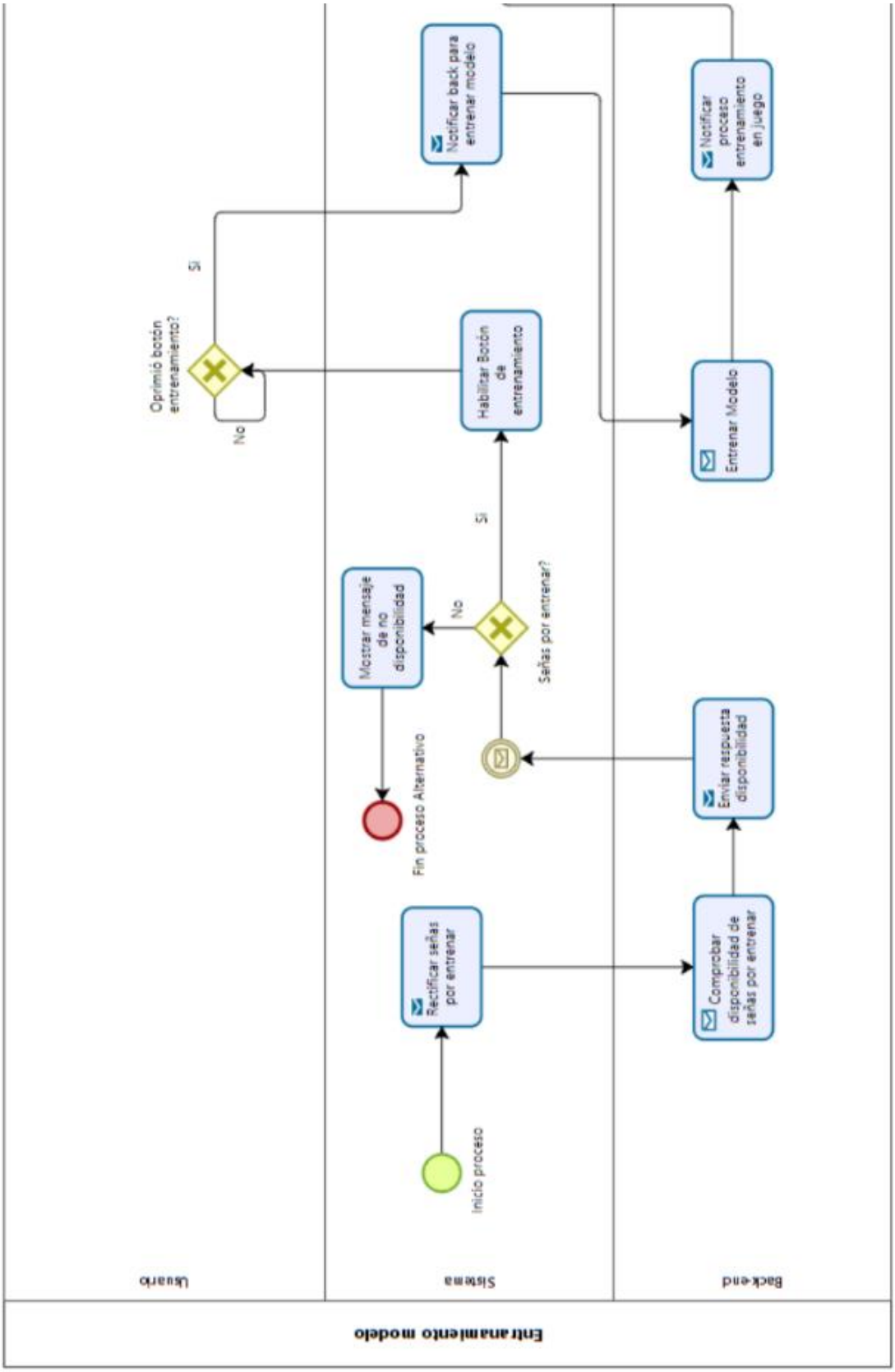

Ilustración 9 Proceso entrenamiento del modelo parte 1

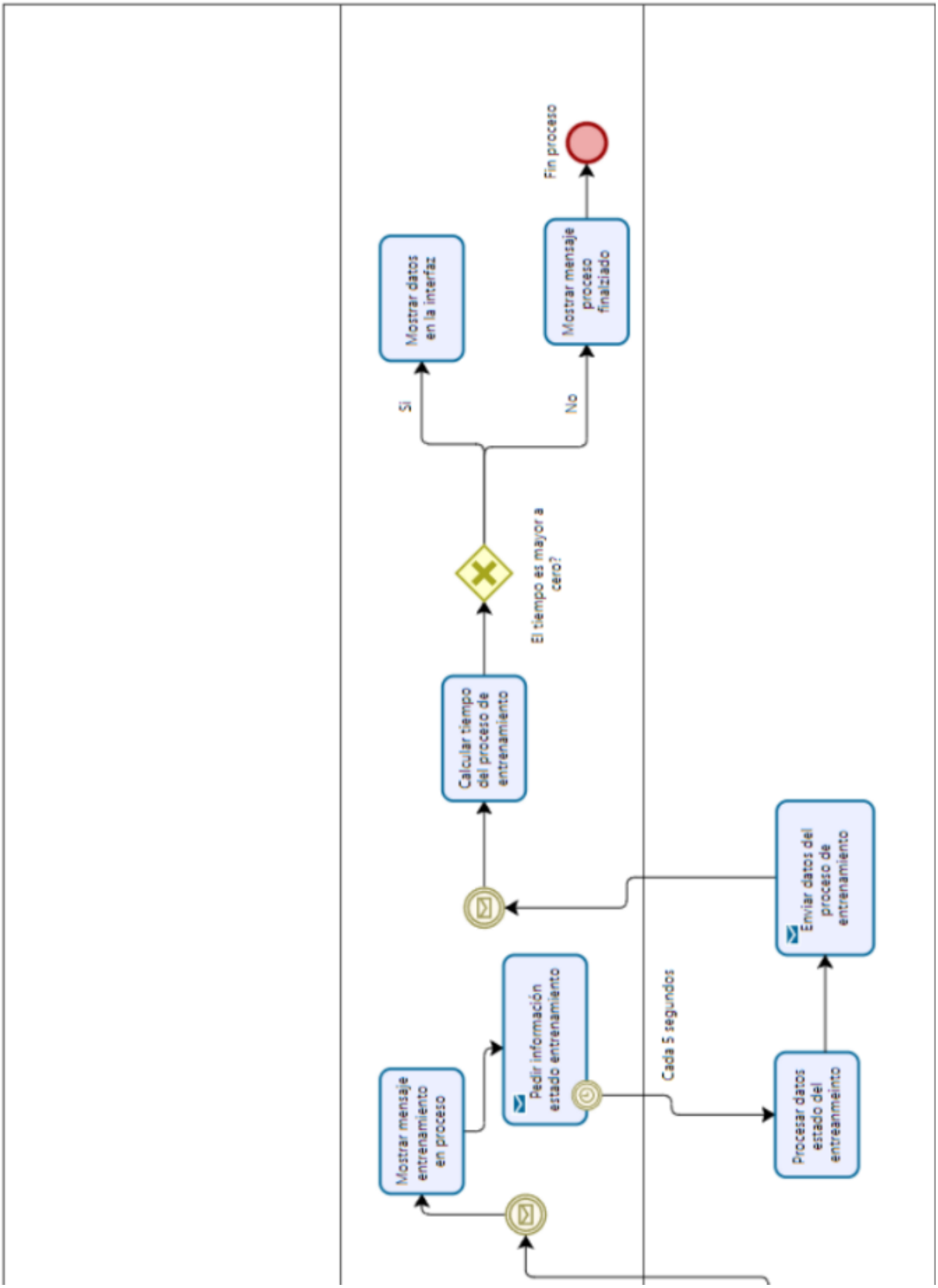

Ilustración 10 Proceso entrenamiento del modelo parte 2

- **Proceso de registro**

Para registrarse en el sistema, es necesario que el usuario diligencie el formulario suminstrado en pantalla, el cual requiere campos de nombre, correo electrónico, contraseña y rol asociado (Administrador y contribuidor), cuando el usuario haya finalizado de llenar el formulario y haya presionado el botón de *registrarse* el *front-end* del sistema validará que los datos llenados estén en el formato correcto, de lo contrario el sistema arrojará un mensaje de error, posteriormente el *front-end* le enviará la información del formulario diligenciado al *back-end* en el que este revisará dicha información, principalmente el correo electrónico y en caso de que el correo ya se encuentre registrado en el sistema, este les mostrará un mensaje de error diciendo que este correo ya se encuentra vinculado a otra cuenta registrada, de lo contrario, el *back-end* guardará esa información vinculada a una cuenta y a su vez, se muestra un mensaje de confirmación de registro.

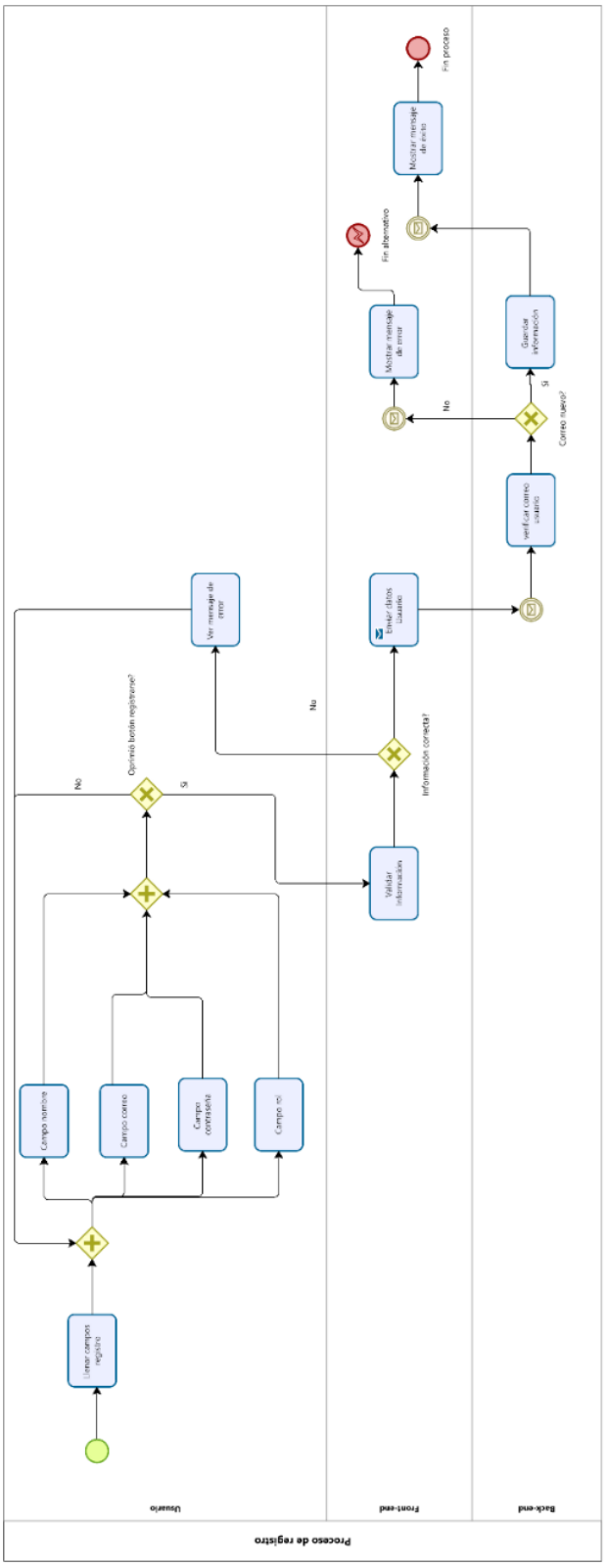

Ilustración 11 Proceso de registro al sistema

- **Proceso de iniciar sesión**

Para iniciar sesión, el usuario debe llenar el formulario con los campos de correo electrónico, contraseña y luego presionar el botón de iniciar sesión, de este modo el *front-end* validará que los datos llenados estén en el formato correcto, de lo contrario el sistema arrojará un mensaje de error, en caso contrario, el *front-end* le enviará los datos diligenciados al *back-end* en donde verificará si el correo electrónico está registrado en la aplicación se le muestra un mensaje de ingreso correctamente, de lo contrario, se mostrará un mensaje de que las credenciales son incorrectas.

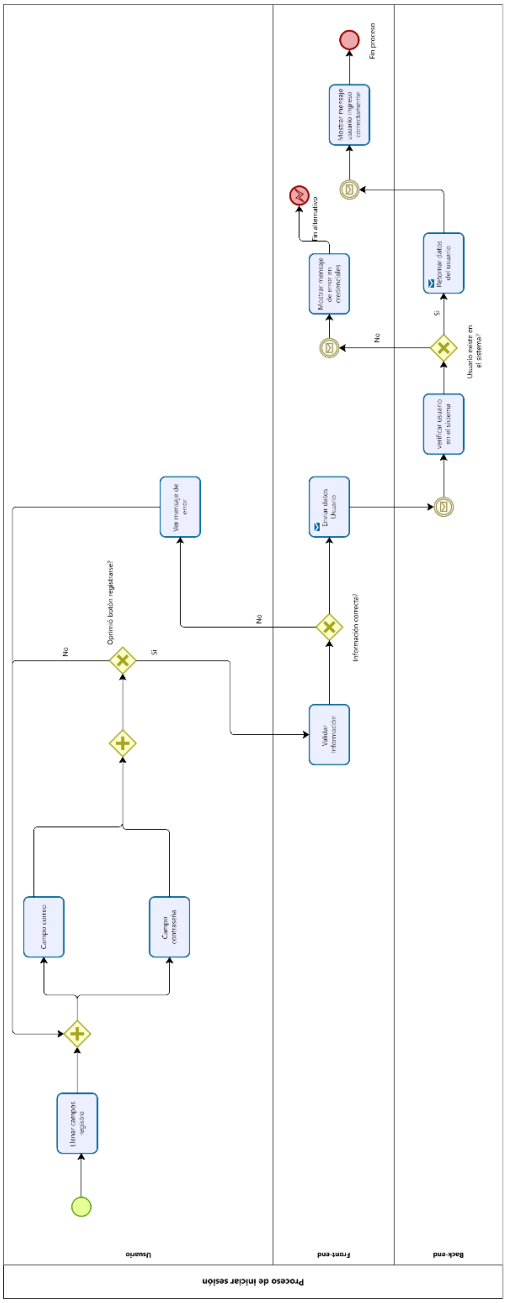

Ilustración 12 Proceso de inicio de sesión al sistema

3. DISEÑO DETALLADO

En este apartado se presenta en como las clases que componen la estructura del sistema se comunican para realizar la extracción y carga de la información.

3.1 Estructura del sistema

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 seña*, 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.

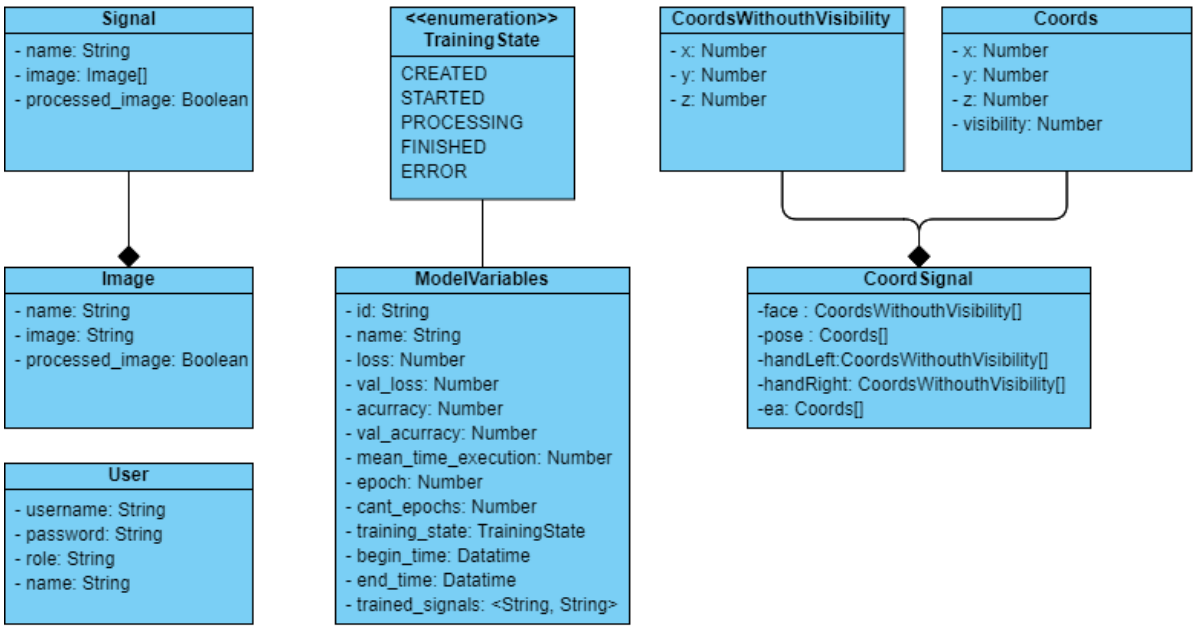

Ilustración 13 Diagrama de clases

3.2 Comportamiento del sistema

Si bien el sistema al tener diferentes funcionalidades como se explicó previamente en los procesos BPMN, el comportamiento del sistema se va a representar por cada funcionalidad que se puede realizar en el sistema como se muestra a continuación.

- Registrar usuario

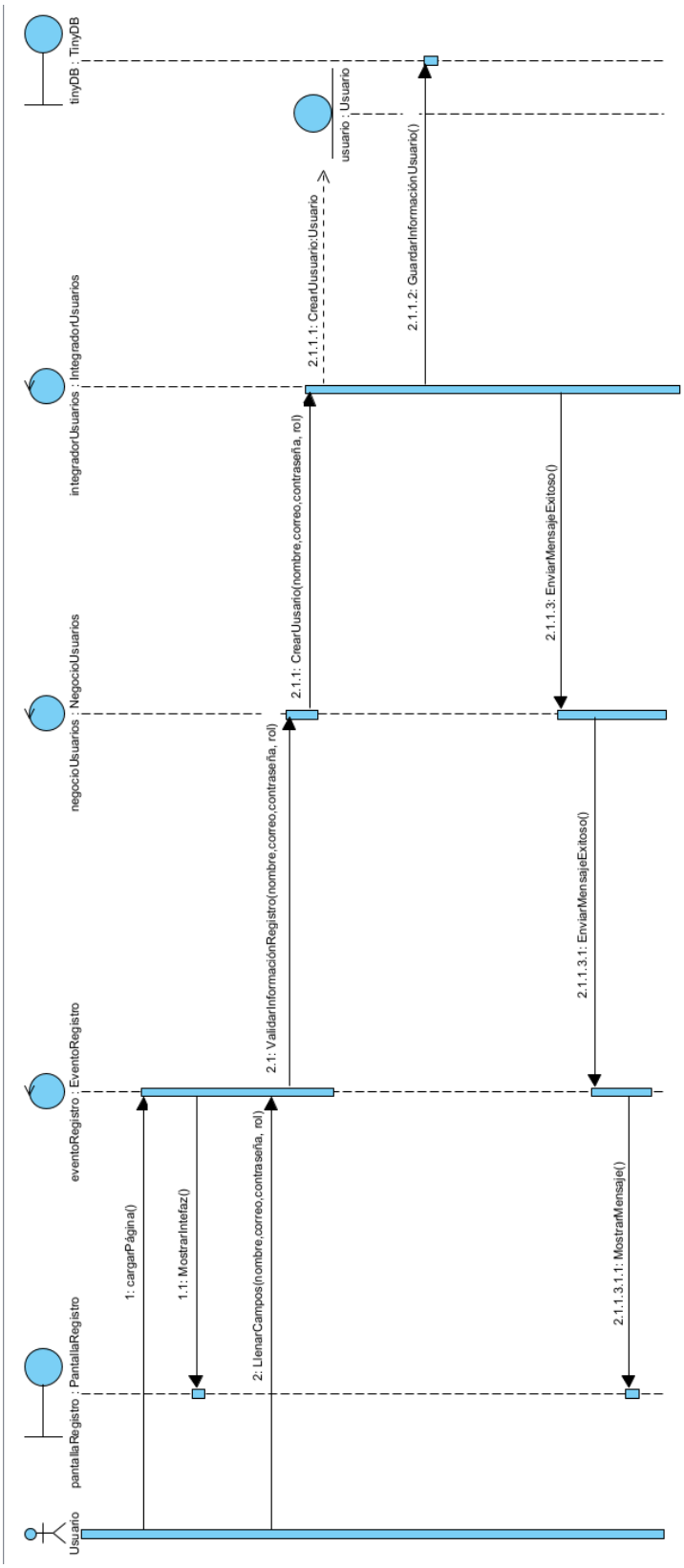

Ilustración 14 Diagrama de secuencia Registrar Usuario

El proceso de registrar usuario inicia cuando el usuario abre la pestaña de *Registrarse*, luego el controlador de eventos de registros le muestra la pantalla al usuario con el formulario a llenar, después el usuario llena los campos nombre, correo, contraseña, rol. Posteriormente, el controlador de eventos valida la información del registro y en caso de que la información este correcta, el controlador de negocio de usuarios creará el usuario con la información diligenciada por el usuario y a su vez, el integrador de usuario creará el modelo usuario y almacenará la información en la base de datos, a su vez, se le retornará un mensaje de confirmación al usuario, el cual se mostrará en pantalla.

- Autenticar usuario

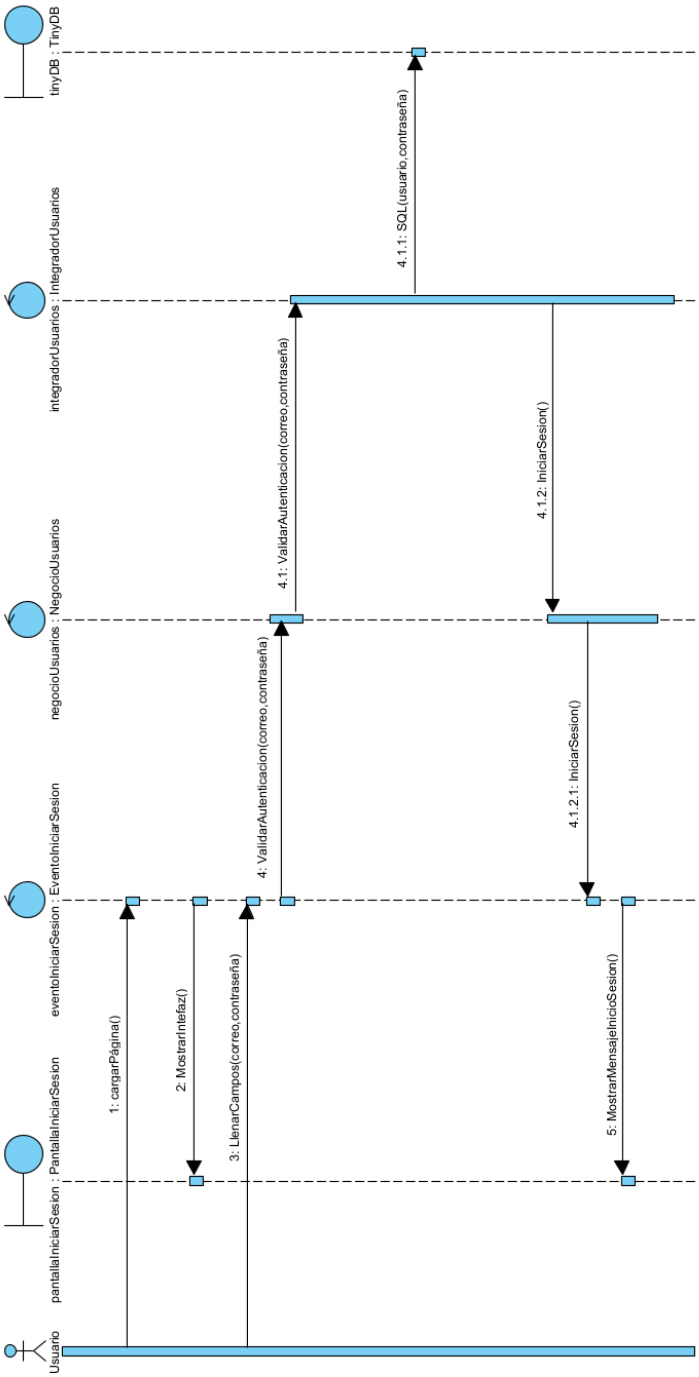

Ilustración 15 Diagrama de secuencia Autenticar Usuario

El proceso de autenticación de un usuario inicia cuando el usuario abre la pestaña *Iniciar Sesión,* luego el controlador de eventos mostrará en pantalla el formulario a llenar para que posteriormente el usuario se autentique llenando los campos de correo electrónico y contraseña, luego el controlador de eventos validará que los datos diligenciados estén correctos consultando en la base de datos si este usuario existe, en caso de que exista se iniciará sesión mostrando en pantalla un mensaje de sesión correctamente.

- Capturar seña

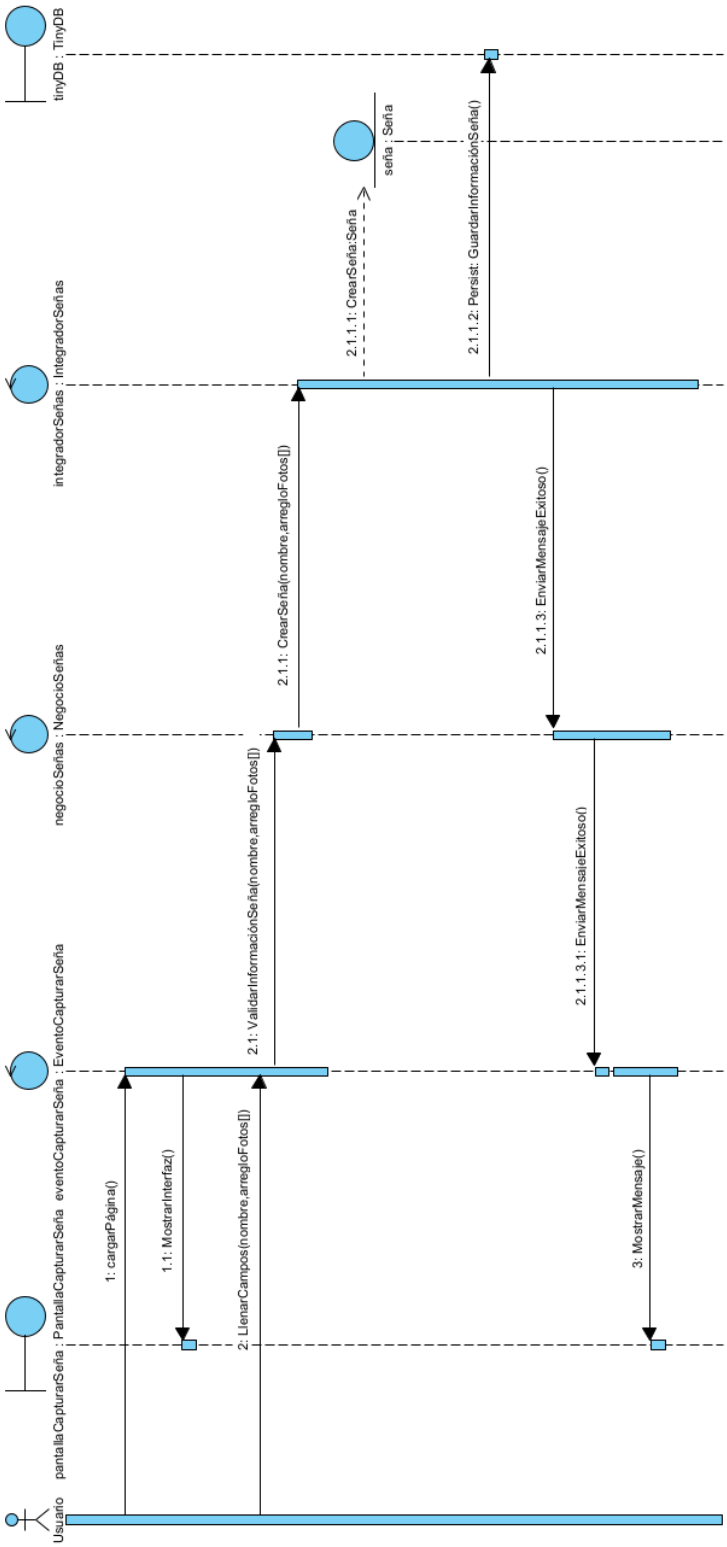

Ilustración 16 Diagrama de secuencia Capturar Seña

El proceso de capturar una nueva seña, inicia primeramente cuando el usuario inicia sesión y selecciona la pestaña *Capturar,* luego el controlador de eventos le envía el contenido de la página a la pantalla para que el usuario llene los campos requeridos para capturar una seña como lo es el nombre de la nueva seña y las fotos asociadas a la nueva seña, es decir un arreglo de fotos, por consiguiente el controlador de eventos validará la información diligenciada y en caso de que este correcta, se le enviará dicha información al integrador para que cree el modelo de la seña y guarde la información en la base de datos, en consecuencia, se le mostrará en pantalla al usuario un mensaje de seña agregada correctamente.

- Entrenamiento del modelo

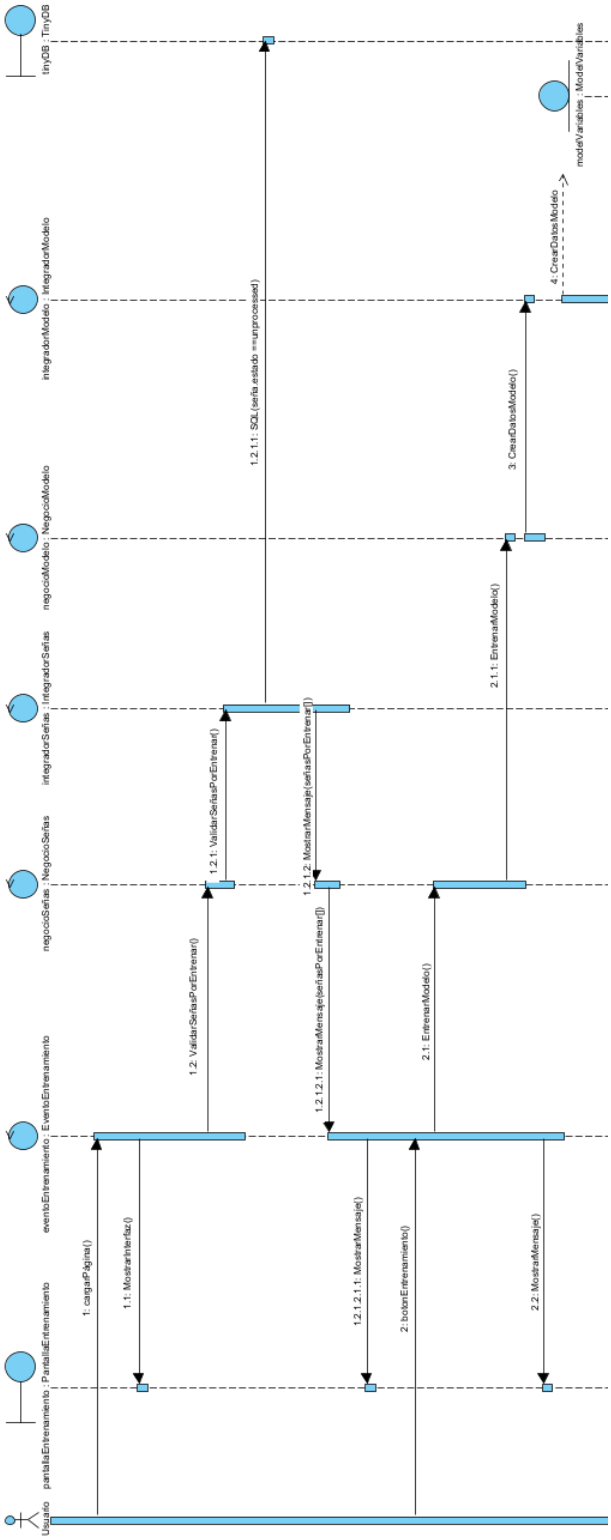

Ilustración 17 Diagrama de secuencia entrenamiento del modelo parte 1

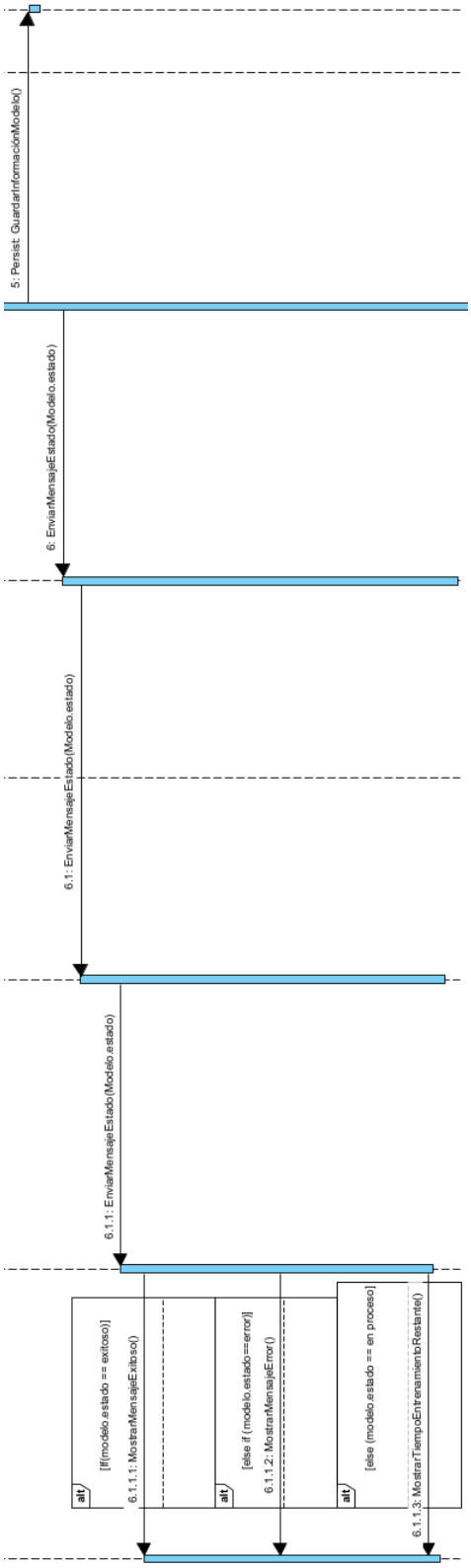

Ilustración 18 Diagrama de secuencia entrenamiento del modelo parte 2

El proceso de entrenamiento inicia primeramente cuando el usuario inicia sesión y selecciona la pestaña de *Entrenar,* después, el controlador de eventos validará que haya señas por entrenar haciendo consulta a la base de datos, en caso de que si haya alguna(s) nueva(s) seña(s) se muestra en pantalla un mensaje de que hay nuevas señas disponibles para entrenar, seguidamente, el usuario selecciona la seña a entrenar y la cantidad de épocas, presiona el botón de entrenamiento y el controlador de eventos le envía los datos de entrenamiento al modelo de negocio para que este cree los datos del modelo *modelvariables* al entrenar con la información suministrada por el usuario, a su vez, esta información se guarda en la base de datos y finalmente, se muestra en pantalla el tiempo restante de entrenamiento junto con la información asociada al entrenamiento del modelo.

- Traducción de señas y visualización de estadísticas

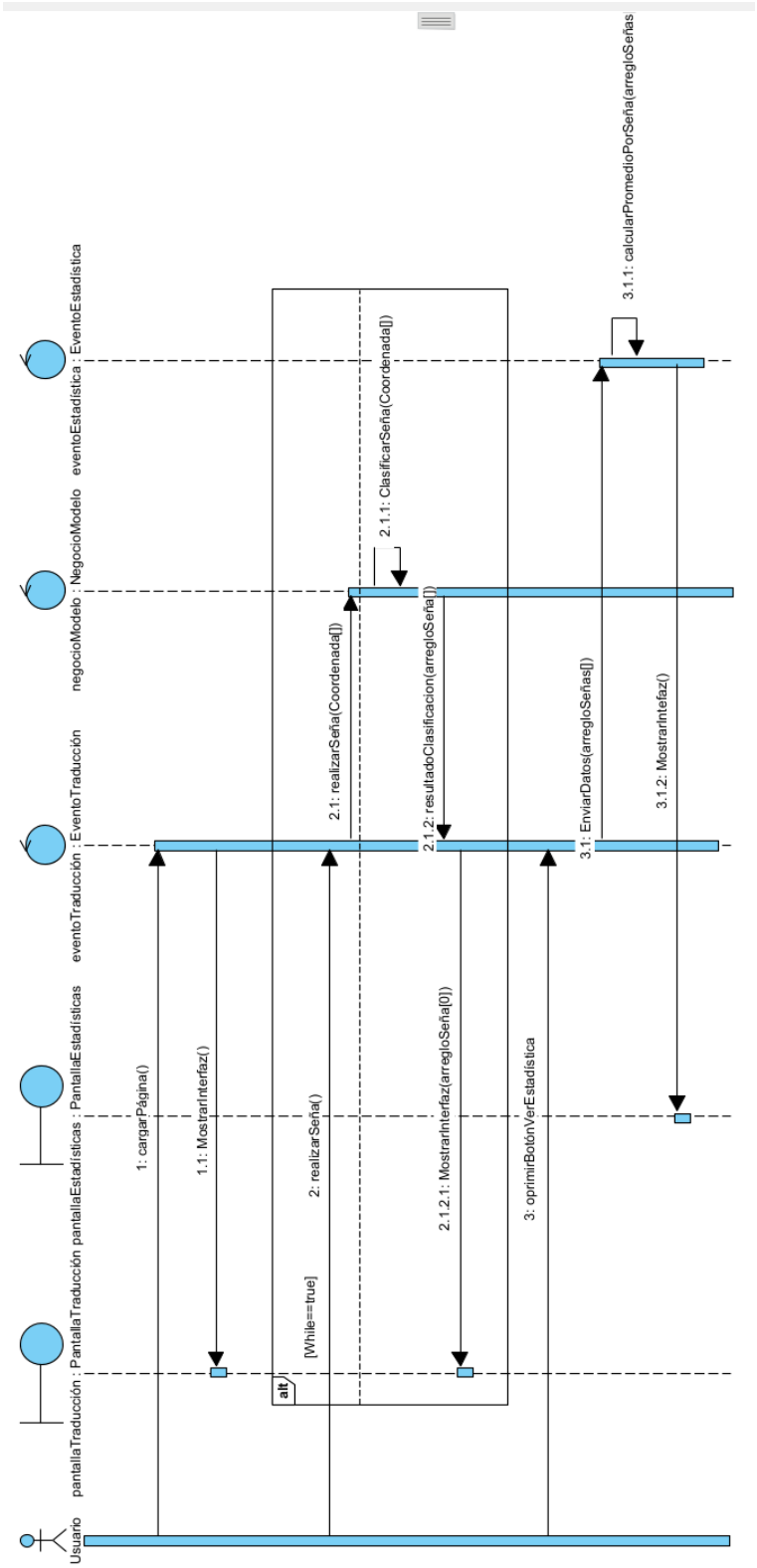

Ilustración 19 Diagrama de secuencia Traducción y visualización de estadísticas

Se representan los diferentes componentes que realiza el Sistema, cuando el usuario se dirige al componente de traducción el controlador de *eventoTraduccion* se encarga de mostrar la interfaz, este proceso se hará infinitamente en el cual consta en que el usuario realiza la seña, el controlador de *eventoTraduccion* le manda los datos que es la coordenada donde esta contiene la información de todos los puntos (Face,Hand,Pose), donde el controlador de *negocioModelo* utiliza el modelo entrenada para clasificar la coordenada enviada por el usuario, este resultado se retorna hasta mostrarle el resultado en la interfaz. Si el usuario oprime el botón de ver estadística el *eventoTraduccion* le manda el arreglo de señas que se han realizado y se comunica con el *eventoEstadistica* y calcula el promedio por seña, de esta forma se muestra loa interfaz mediante un diagrama de barras.

3.3 Persistencia.

Con el objetivo de persistir los datos obtenidos a lo largo del proyecto, se realizó mediante la base de datos *TinyDB* donde este es un componente no visible que permite almacenar datos, estos son guardados por medio de archivos *JSON*, gracias a esto se guardara la información de los usuarios y lo relacionado con las señas como es el caso de contener en una estructura todo lo concurrente con el proceso de entrenamiento.