Índice

[Índice 1](#_Toc151384713)

[Tablas 1](#_Toc151384714)

[Ilustraciones 1](#_Toc151384715)

[Esquemas 2](#_Toc151384716)

[Glosario de Términos: 2](#_Toc151384717)

[**CAPÍTULO III – “**DISEÑO E IMPLEMENTACIÓN DEL SISTEMA**”** 2](#_Toc151384718)

[3.1 Diseño Sistema General 2](#_Toc151384719)

[3.2 Dispositivo de Laboratorio 3](#_Toc151384720)

[3.2.1 Comunicación con el Sensor 4](#_Toc151384721)

[3.2.2 Arquitectura de la Interfaz Visual 4](#_Toc151384722)

[3.2.3 Arquitectura Base de Datos 7](#_Toc151384723)

[3.2.4 Transferencia de datos 8](#_Toc151384724)

[3.3 Dispositivo de Campo 8](#_Toc151384725)

[3.3.1 Interacción con el Sensor 9](#_Toc151384726)

[3.3.2 Arquitectura de Clases 9](#_Toc151384727)

[3.3.3 Pagina Web 11](#_Toc151384728)

[3.4 Utilización de Redes Neuronales Artificiales 12](#_Toc151384729)

[3.4.1 Arquitectura de Red Neuronal 13](#_Toc151384730)

[3.4.2 Pruebas en Laboratorio 14](#_Toc151384731)

[3.4.3 Pruebas en Campo 14](#_Toc151384732)

[Referencias 14](#_Toc151384733)

[Anexos 14](#_Toc151384734)

# Tablas

[Tabla 1: Formato CSV 7](#_Toc151135761)

# Ilustraciones

[Ilustración 1: Vita de usuarios 5](#_Toc151135762)

[Ilustración 2: Interfaz visual con barral lateral 6](#_Toc151135763)

[Ilustración 3: Página Web Desplegada por ESP32 10](#_Toc151135764)

[Ilustración 4: Interfaces de actualización de modelo en ESP32 10](#_Toc151135765)

# Esquemas

[Esquema 1: Arquitectura General del Sistema. 3](#_Toc151135766)

[Esquema 2: Conexión DL-Sensor 4](#_Toc151135767)

[Esquema 3: Interfaces Web del dispositivo de laboratorio 5](#_Toc151135768)

[Esquema 4: Arquitectura de Base de Datos Relacional 7](#_Toc151135769)

[Esquema 5: Conexión DC-Sensor 8](#_Toc151135770)

[Esquema 6: Flujo de Código del ESP32 9](#_Toc151135771)

Glosario de Términos:

* DL: Dispositivo de Laboratorio
* DC: Dispositivo de Campo
* CSV (Comma Separated Values): Valores Separados por Coma. Es un estándar de formato de documento para representar en forma de tabla la información
* IDE
* VSC
* WSGI
* IP
* Wi-Fi

# **CAPÍTULO III – “**DISEÑO E IMPLEMENTACIÓN DEL SISTEMA**”**

El diseño e implementación de un sistema para detectar adulteraciones en la leche requiere de una correcta implementación tanto del hardware como del software. Ambos componentes son fundamentales para lograr resultados precisos y confiables en la detección de adulterantes en este importante producto.

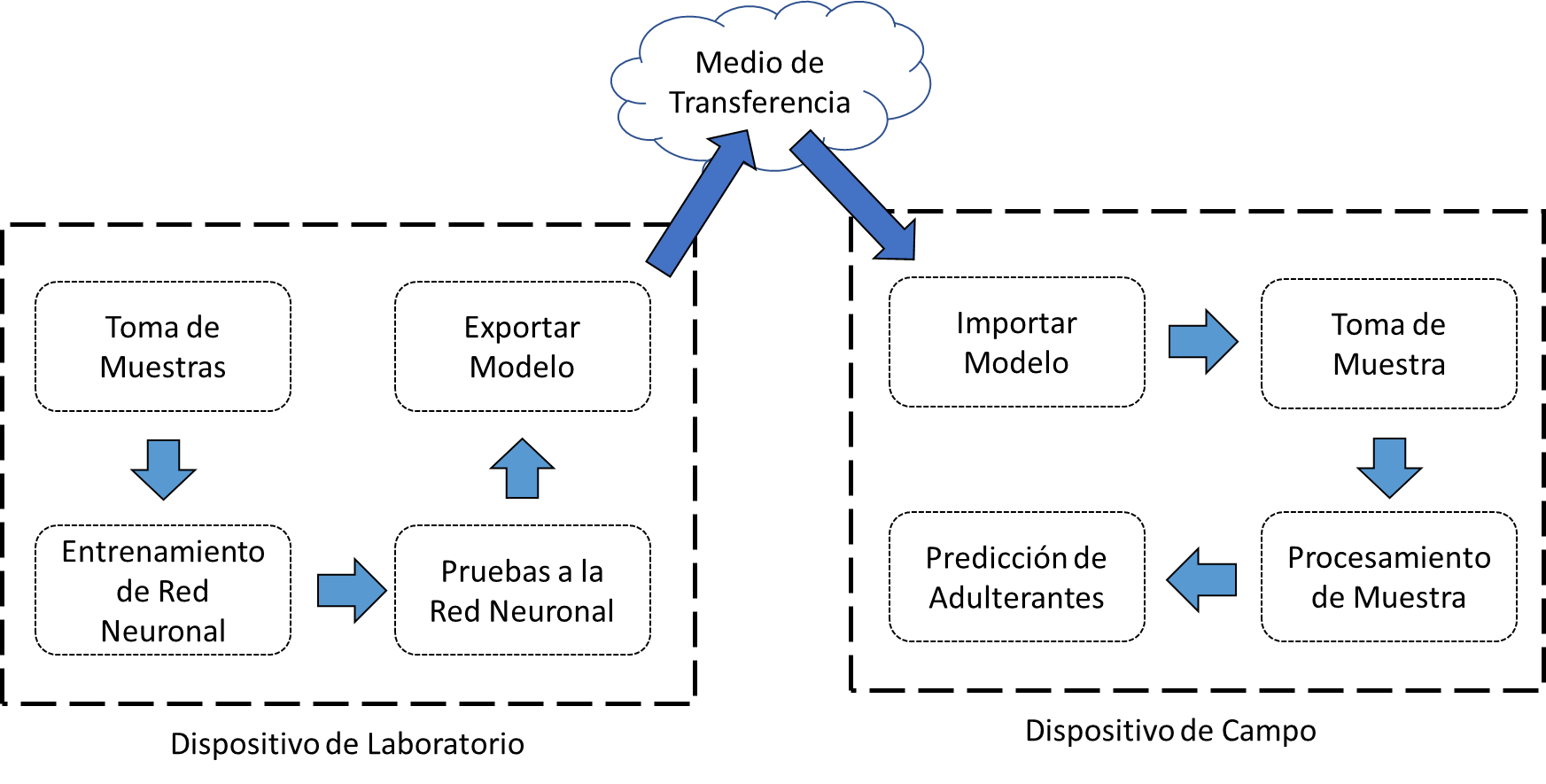
## 3.1 Diseño Sistema General

El sensor AS7265x permite detectar cambios en la composición de la leche. Para ello el sistema a desarrollar tiene que ser capaz de comunicarse con este sensor y tomar las posibles muestras para ser analizadas. Además, el sistema tiene que permitir encontrar patrones específicos para el reconocimiento de anomalías, para ello se utilizaran las grandes capacidades que brindan las RNA para clasificación de variables.

Para utilizar redes neuronales artificiales (RNA) de manera efectiva, es recomendable contar con un dispositivo con un alto nivel computacional. Esto se debe a que el entrenamiento de modelos de RNA puede ser computacionalmente intensivo y requerir recursos significativos, como capacidad de procesamiento y memoria.

Una vez que el modelo ha sido entrenado, es posible transferirlo a otras placas o dispositivos para su uso óptimo. Esto se debe a que la arquitectura y parámetros de la RNA pueden ser exportados y utilizados en otros dispositivos con recursos computacionales más limitados; por lo tanto, el diseño de dividirá en dos etapas o dispositivos:

* Dispositivo de laboratorio
* Dispositivo de campo



Esquema 1: Arquitectura General del Sistema.

## 3.2 Dispositivo de Laboratorio

El DL (dispositivo de laboratorio) en general, tendrá la tarea de tomar las muestras necesarias del sensor AS7265X, y entrenar un modelo óptimo que permita reconocer adulterantes en la leche. Este proceso puede requerir de mucho computo, por lo tanto, se necesita un dispositivo de altas capacidades de trabajo.

Este dispositivo de laboratorio tiene que ser capaz de:

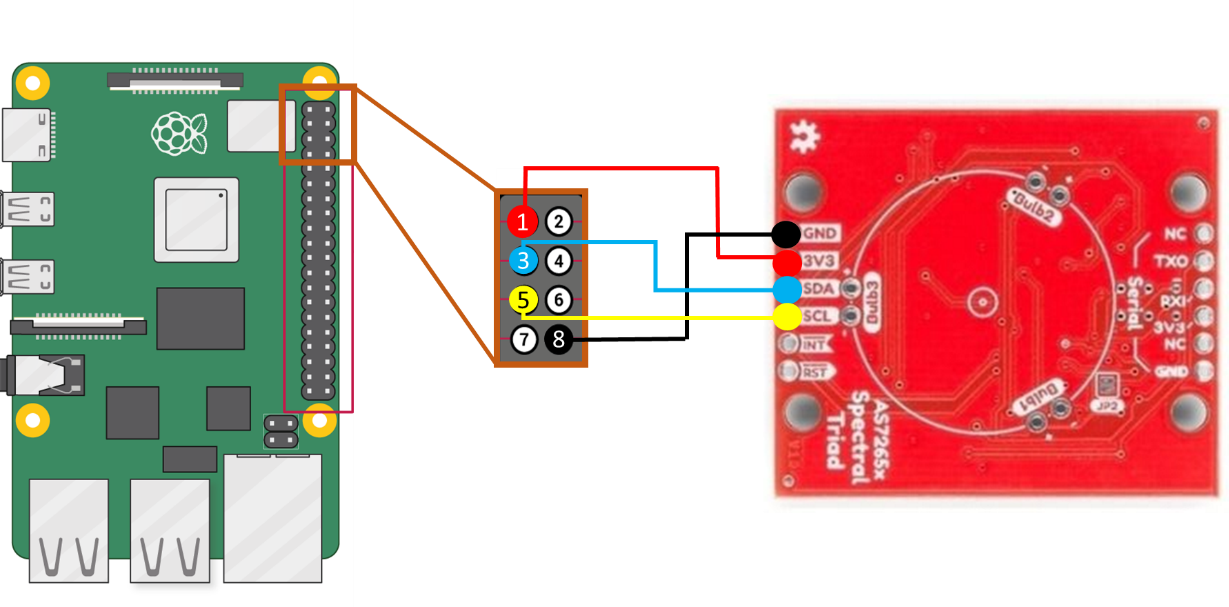
* Comunicación con el sensor y lectura de muestras (I2C o UART)
* Almacenar muestras de forma consistente (Base de datos)
* Entrenamiento del modelo
* Pruebas de predicción con modelo entrenado
* Interfaz visual para la interacción humano-maquina
* Alojar modelo hacia servidores para ser usado por los distintivos de campo

La placa de desarrollo Raspberry tiene el potencial necesario para ser seleccionado como el DL, ya que permite la conectividad de los protocolos establecidos por el sensor, permite lenguajes de bases de datos, además puede implementar Python junto a todas las librerías de redes neuronales (Tensorflow), gestión web (Django) y trabajo con servidores. Se utilizará más específicamente su versión 3 modelo B, debido a que presenta modulo de wifi integrado para una mejor comunicación con el exterior y una mayor capacidad computacional que las versiones anteriores.

### 3.2.1 Comunicación con el Sensor

Para le desarrollo del sistema de hardware, es necesario un bus de comunicación con el sensor, este soporta conexiones I2C, por lo tanto, este protocolo será utilizado en ambos dispositivos (DL y DC) para la recolección de información del espectro.

La Raspberry Pi 3B presenta hasta dos interfaces I2C (Ver Ilustración ???), se utilizarán los pines GPIO02(SDA1) y GPIO03(SCL1) para la conexión del bus. Se utilizara Python junto con la librería “smbus” para la conexión entre los pines físicos y el lenguaje de alto nivel.

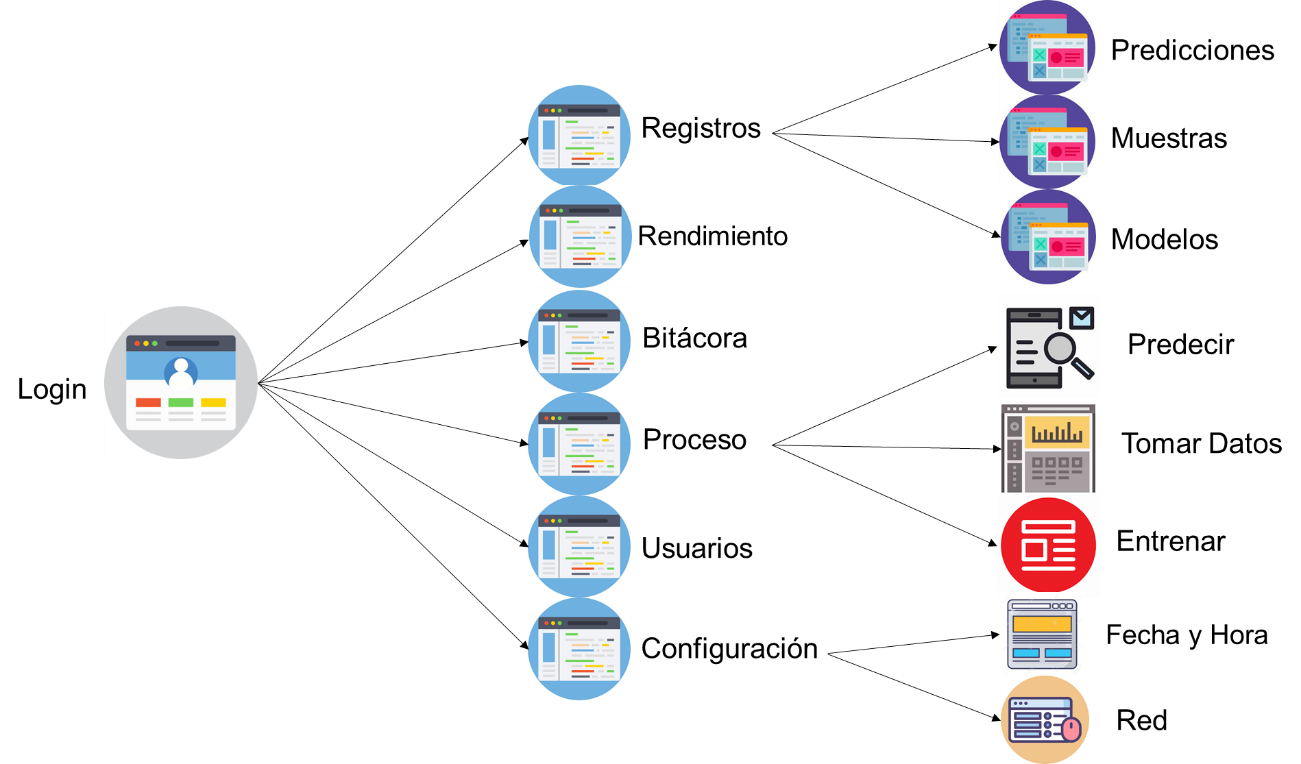


Esquema 2: Conexión DL-Sensor

### 3.2.2 Arquitectura de la Interfaz Visual

Para la interfaz visual se utilizará el marco Django, junto a la plantilla AdminLTE3, las cuales permitirán crear una interfaz simple, agradable y segura para el intercambio de información entre el usuario y el dispositivo.

El servidor de Django fue desplegado con Apache2; servidor que permite alojar páginas web, junto al módulo WSGI para vincularlo con Python. Para acceder a el sitio web, será necesario comunicar la Raspberry mediante Wi-Fi o Ethernet a un dispositivo y acceder a la página mediante el IP y el puerto 88, ejemplo: 172.20.10.4:88.



Esquema 3: Interfaces Web del dispositivo de laboratorio

La programación del marco Django será orientada a “clase basada en vistas” donde se aprovechan las capacidades de este método, para brindar de una amplia seguridad al acceder a las páginas. La autenticación se hará al acceder al sitio siendo esta obligatoria en cualquier URL que se acceda. Para implementarlo se han desarrollado tres niveles de seguridad:

* Desarrollador
* Administrador
* Invitado



Ilustración 1: Vista de usuarios e inicio de sesión

Donde el desarrollador tendrá acceso total a todas las vistas de la página, el administrador no podrá a acceder a vistas como configuraciones internas mientras que el invitado solo podrá visualizar la información de las muestras y modelos, así como los resultados de entrenamiento.

La interfaz contara con una barra lateral de navegación para desplazarse por toda la aplicación, mostrando en todo momento la ubicación del usuario

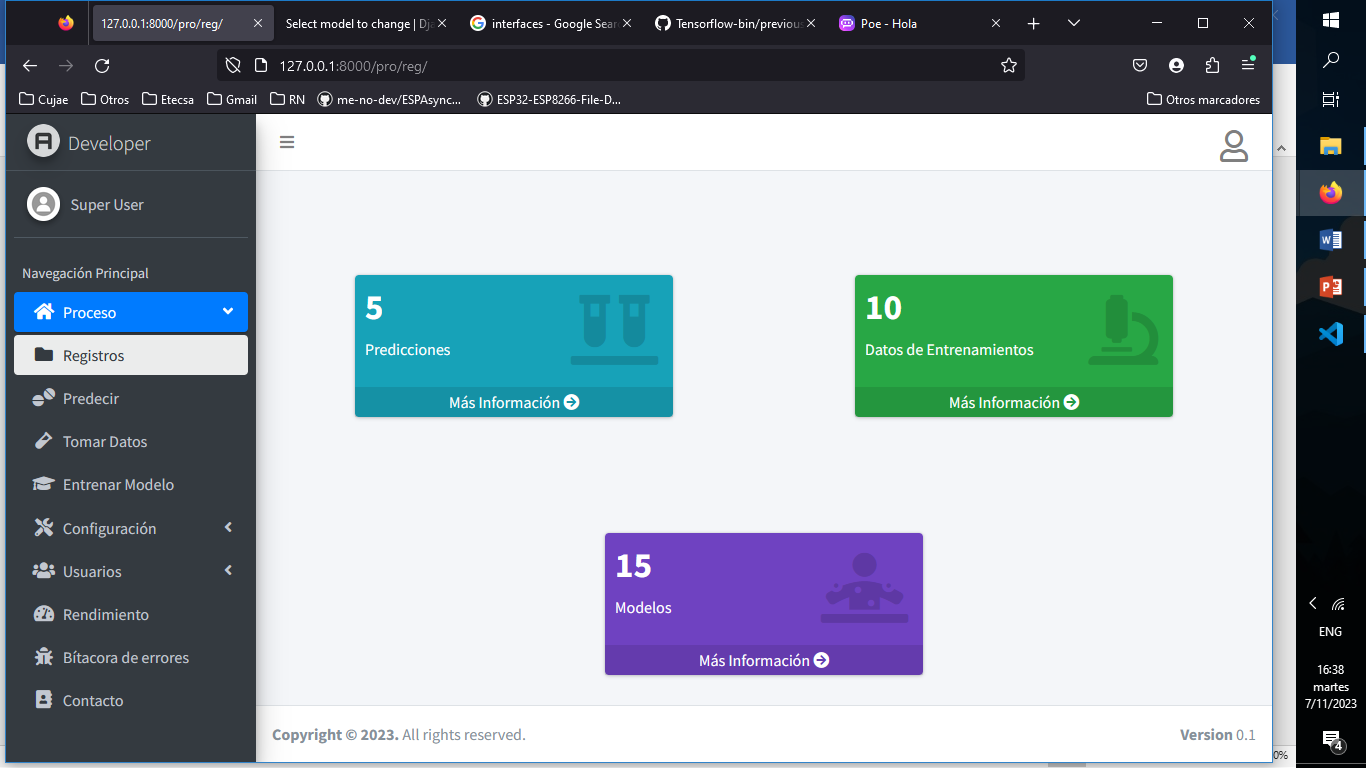


Ilustración 2: Interfaz visual con barral lateral

Para lograr un mayor rendimiento de la aplicación, se utilizará programación multihilo con la librería “Threading” de Python, para así evitar bloqueos en la aplicación. Tanto en la interfaz de predicción, toma de datos, carga de ficheros y entrenamiento se iniciarán en hilos distintos, permitiendo solo uno de cada tipo a la vez, se podrá visualizar el estado de cada proceso en su respectiva interfaz de registro.

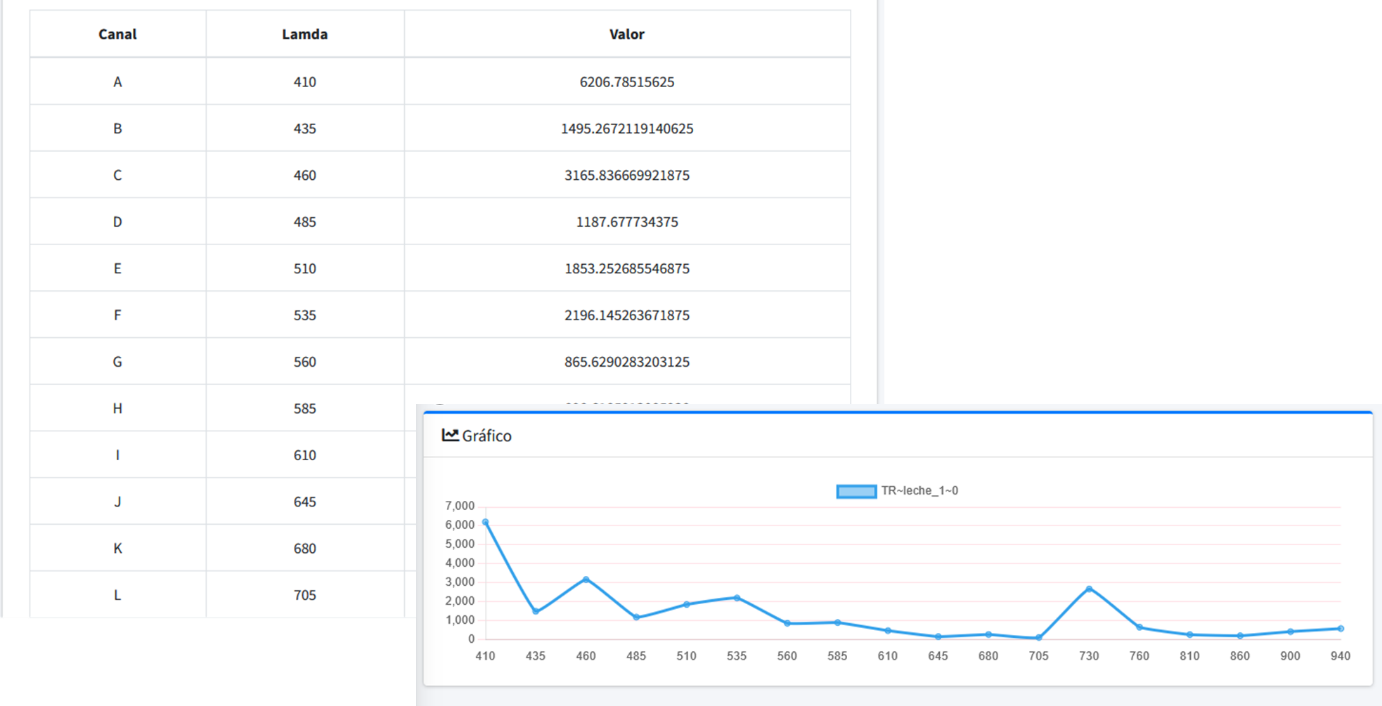
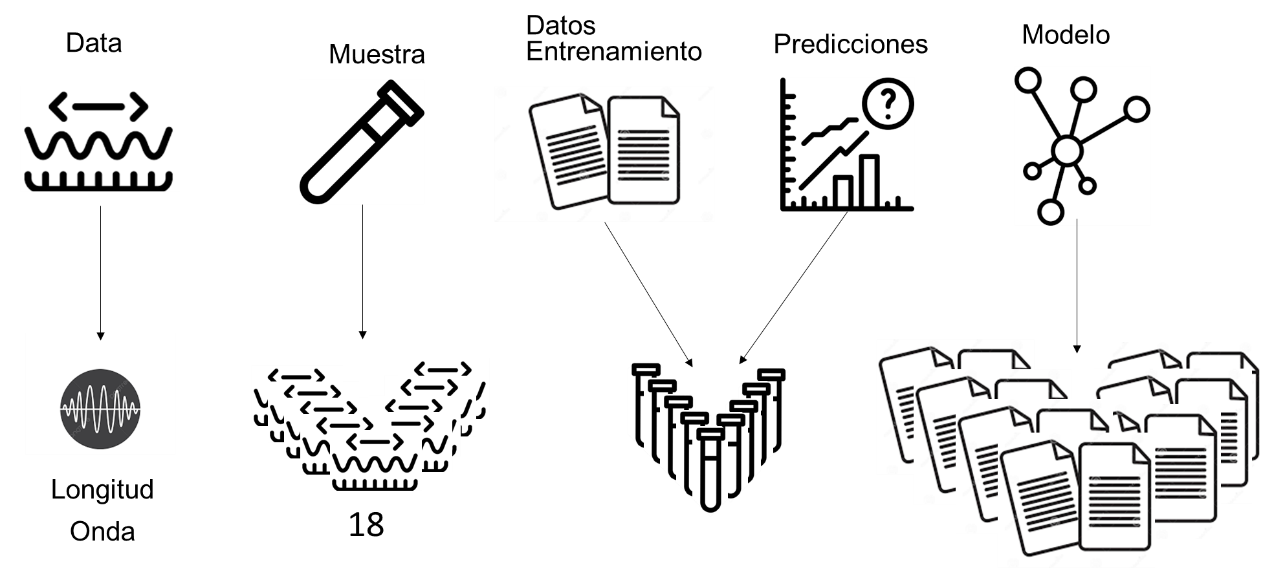


Ilustración 3: Ejemplo Interfaz de una Muestra

### 3.2.3 Arquitectura Base de Datos

Es necesario almacenar de manera consistente y organizada la información proveniente del sensor. Esto permite una búsqueda eficiente de los datos necesarios. Django viene implementado por defecto con la base de datos SQLite3, base de datos que permite alancear de manera organizada las muestras. Para el diseño de la base de datos relacional se establecieron los siguientes modelos:

* **Data**: Modelo base de la base de datos, contiene la información de una longitud de onda. Presenta campo FOREIGN KEY a Muestra
* **Muestra**: Modelo que contiene hasta 18 datas, campo FOREIGN KEY a Datos Entrenamiento.
* **Predicción**: Modelo Puntero a Muestra
* **Datos Entrenamiento:** Campo MANY TO MANY a Modelo
* **Modelo:** Contiene la información del modelo entrena



Esquema 4: Arquitectura de Base de Datos Relacional

### 3.2.4 Transferencia de datos

Para permitir la utilización de diferentes dispositivos para tomar las muestras, es posible almacenar registros con información sobre el estado de la leche en bases de datos separadas. Para facilitar esta funcionalidad, se requiere un prototipo de archivo que permita exportar e importar los datos. Con el fin de lograr este objetivo, se utilizará el formato CSV (Comma Separated Values).

Se han implementado dos arquitecturas para el formato de los archivos CSV (ver tabla ???). Esto asegura la compatibilidad y el intercambio de datos de manera eficiente entre diferentes sistemas.

.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Nombre | Formato | Data | Etiqueta | Adicional |
| Formato 1 | A, B, C, D, E, F, G, H, I, J, K, L, R, S, T, U, V, W, label | A~W | Label |  |
| Formato 2 | date,labels,410,435,460,485,510,535,560,585,610,645,680,705,730,760,810,860,900,940 | 410~940 | Labels | Date (Fecha) |

Tabla 1: Formato CSV

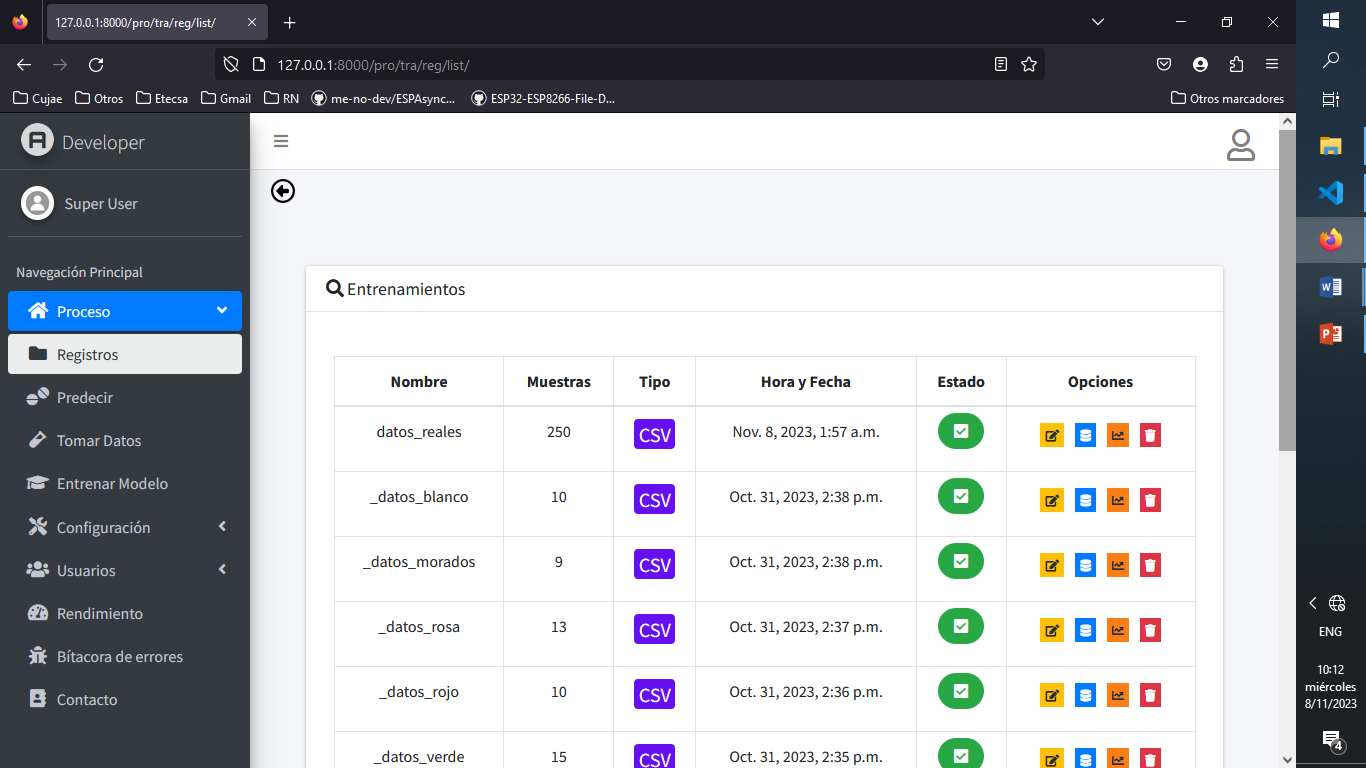


Ilustración 4: Muestras importadas por CSV

### 3.2.5 Diseño Prototipo

Las pruebas de laboratorio desempeñan un papel fundamental en el desarrollo de este proyecto, ya que es necesario contar con un modelo óptimo que permita predecir la presencia de adulterantes. Por lo tanto, la calidad de las pruebas realizadas es de vital importancia para obtener un modelo de red neuronal sólido.

Como se menciona en [TESIS], la influencia de la luz externa en la precisión de la medición del sensor AS7256X es crucial para detectar de manera precisa los adulterantes. En vista de esto, será necesario utilizar un recipiente contenedor de muestras que aísle adecuadamente la luz externa y permita la medición precisa de la luz en el sensor.

Con el fin de satisfacer este requisito, se ha desarrollado un prototipo de un pequeño recipiente de muestras utilizando Autodesk Inventor. Este sistema está diseñado para aislar la luz y se ha fabricado en un color oscuro debido a las propiedades de este color para absorber la luz. Consiste en una caja con un orificio en la parte superior donde se coloca el sensor, apuntando hacia el interior de la caja. Dentro de la caja, se encuentra un recipiente cilíndrico para almacenar las muestras, el cual está equipado con un rodamiento inferior que permite desplazar y cambiar las muestras fácilmente. Finalmente, el sistema puede cerrarse estilo "gaveta", alineando el recipiente con los sensores y asegurándolo para evitar cualquier interferencia causada por la luz externa.

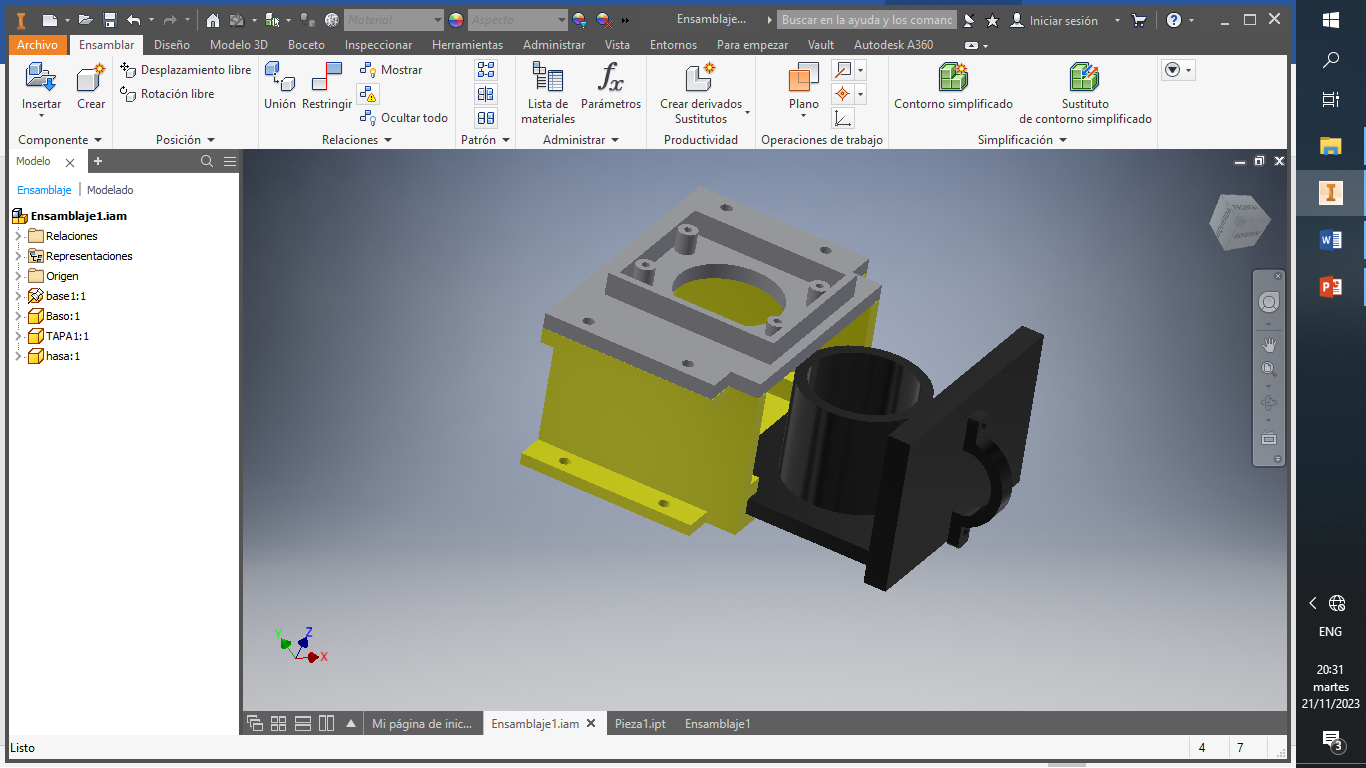


Ilustración 5: Diseño recipiente de muestras

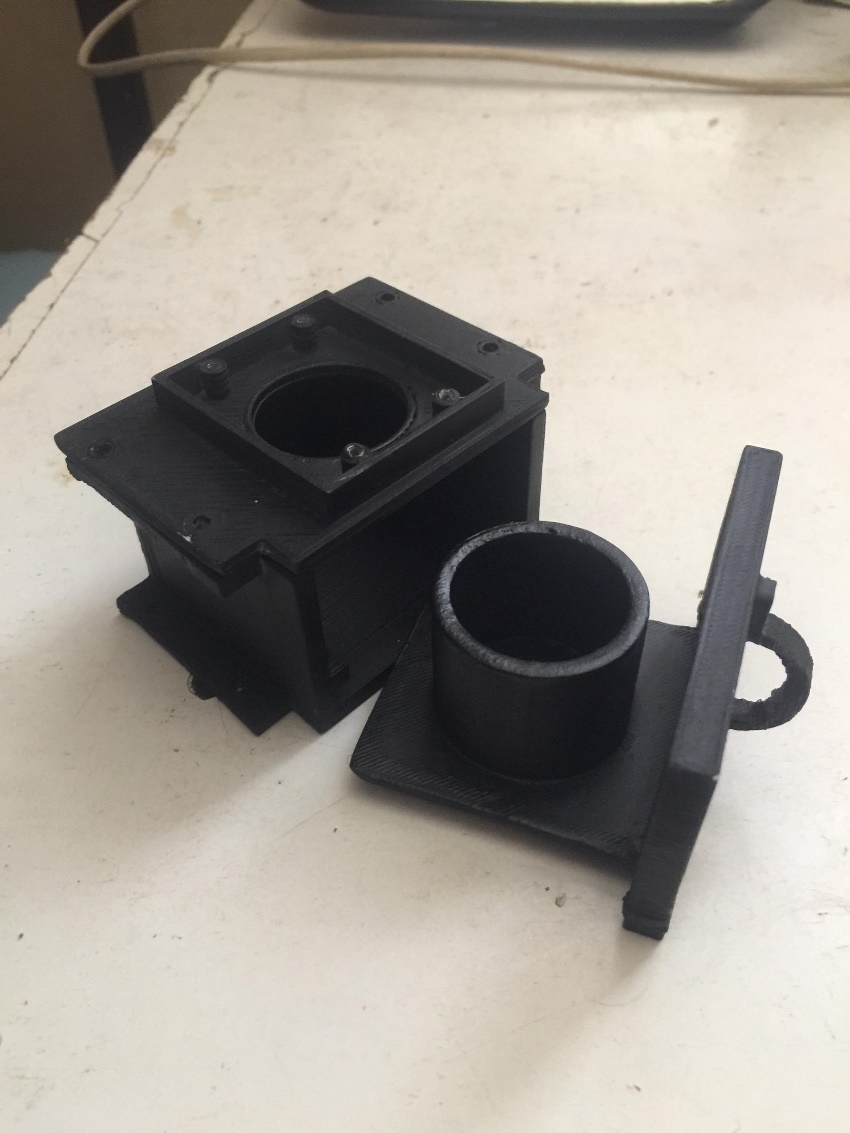


Ilustración 6: Recipiente de muestras

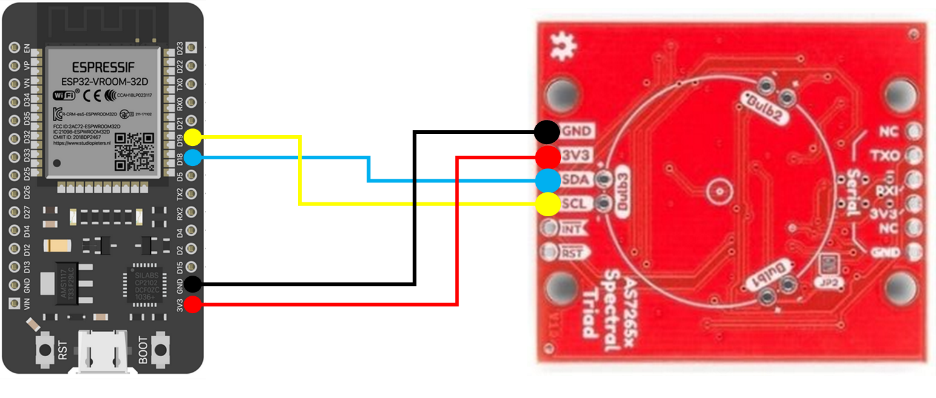
## 3.3 Dispositivo de Campo

El DC (Dispositivo de Campo) tendrá el objetivo que cargar los modelos creados por los de laboratorio, y a partir de muestras nuevas muestras del sensor a través de la red neuronal conocer el estado de adulterantes de la leche.

Este dispositivo solo tendrá que utilizar los modelos ya entrenados, por lo tanto, no requerirá de alto procesamiento como el de laboratorio. Necesitará de una interfaz web para la comunicación con los usuarios, así como la posibilidad de comunicarse con los DL para la actualización de los modelos de redes neuronales; por lo tanto, un ESP32 es adecuado para este dispositivo, además de sus bajos costos, posee los módulos de Wifi los cuales serán necesarios para la comunicación con el dispositivo de laboratorio y el intercambio de modelos e información. Este se puede escribir en lenguaje C, lo que lo hace compatible con la librería de Tensorflow en su versión lite.

### 3.3.1 Interacción con el Sensor

Para la toma de muestras en el ESP32 se necesita de un bus I2C para la comunicación con el AS7265x. El DC no trae por defecto pines asignados para I2C, pero puede asignar hasta 2 pares de pines para la utilización del protocolo. Se utilizaron los pines D18(SDA) y D1(SCL) para la comunicación con el sensor. Para la comunicación entre el lenguaje C con los pines físicos del dispositivo, se utilizará la librería “Wire.h” la cual implementa la comunicación I2C de manera óptima.



Esquema 5: Conexión DC-Sensor

### 3.3.2 Arquitectura de Clases

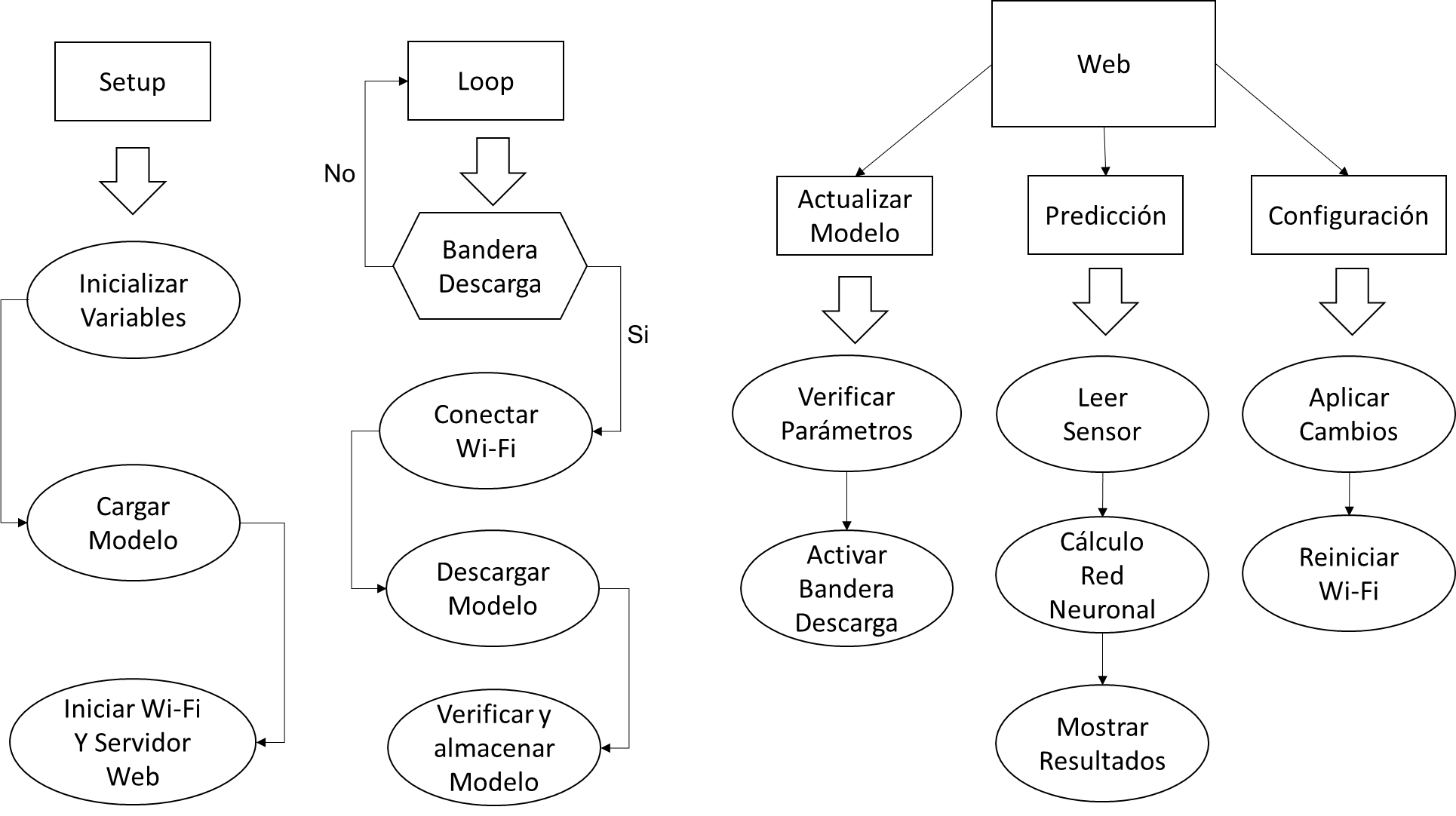
Como ya se conoce, el ESP32 es un microcontrolador programable en lenguaje C el cual será programado utilizando el entorno de desarrollo integrado (IDE) Visual Studio Code (VSC) con la extensión PlatformIO. Se utilizo la arquitectura de la plataforma de Arduino para la programación, por lo tanto este cuenta con las funciones “Setup” y “Loop”. En la función “Setup” se inicializan todas las variables del sistema, donde se carga el modelo de red neuronal, se activa la Wi-Fi y el servidor web.

Diseño de clases

* **As7265x**: Clase encargada de comunicarse con sensor, configurar y leer datos
* **Spiffs**: Clase encargada de manejar la memoria flash del esp32, verifica, carga y guarda información en la memoria
* **Wifi:** Clase encargada de manejar el controlador wifi del esp32, habilitar modos AP, con ssid y password específicos, así como conectarse a wifi para acceder al servidor. Contiene un puntero a la clase Spiffs para acceder a los datos almacenados de ssids.
* **Model**: Maneja el modelo descargado del servidor y almacenado en memoria. Actúa sobre la clase EloquentTinyML, librería que trabaja con elementos estáticos por lo tanto para modificar el modelo será necesario un reinicio de los parámetros de esta librería
* **Download**: Librería encargada de descargar el modelo del servidor. Utiliza HTTPClient para realizar las peticiones. Implementa la clase Spiffs para almacenar modelo provisional
* **Object**: Clase base para casi todas las operaciones que carga información en arreglos dinámicos, con fácil acceso, incremento y borrado.
* **\_18float:** Clase encargada de almacenar y manejar los 18 valores provenientes del sensor
* **Web:** Clase que contiene el mayor peso ya que presenta un puntero a todas las librerías anteriores para poder utilizar sus funcionalidades de una manera ordenada. Contiene toda la implementación del servidor web, montado cobre la librería AsynWebServer.

En la función “Loop” del programa se implantará solamente una bandera de “descarga”, que se activa desde la interfaz visual, debido a que la actualización del modelo a través de la red puede ser de tiempo indefinido, es necesario realizarlo en el programa principal para evitar problemas como la activación del “Watchdog”, el cual reinicia el microcontrolador cuando existe una gran demora no planificada.

Además, es claro mencionar que se utilizara la memoria flash del ESP32 de 4Mb para el almacenamiento de los ficheros estáticos de la página web como los HTML, estilos y JavaScript, así como los modelos de redes neuronales y banderas internas.



Esquema 6: Flujo de Código del ESP32

### 3.3.3 Pagina Web

Para la utilización de un servidor web se utilizado la librería AsynWebServer junto a WIFI del esp32, esta librería permite servir ficheros estáticos de una forma sencilla. Se ha utilizado Bootstrap y JQuery como librerías para la implementación de la página.

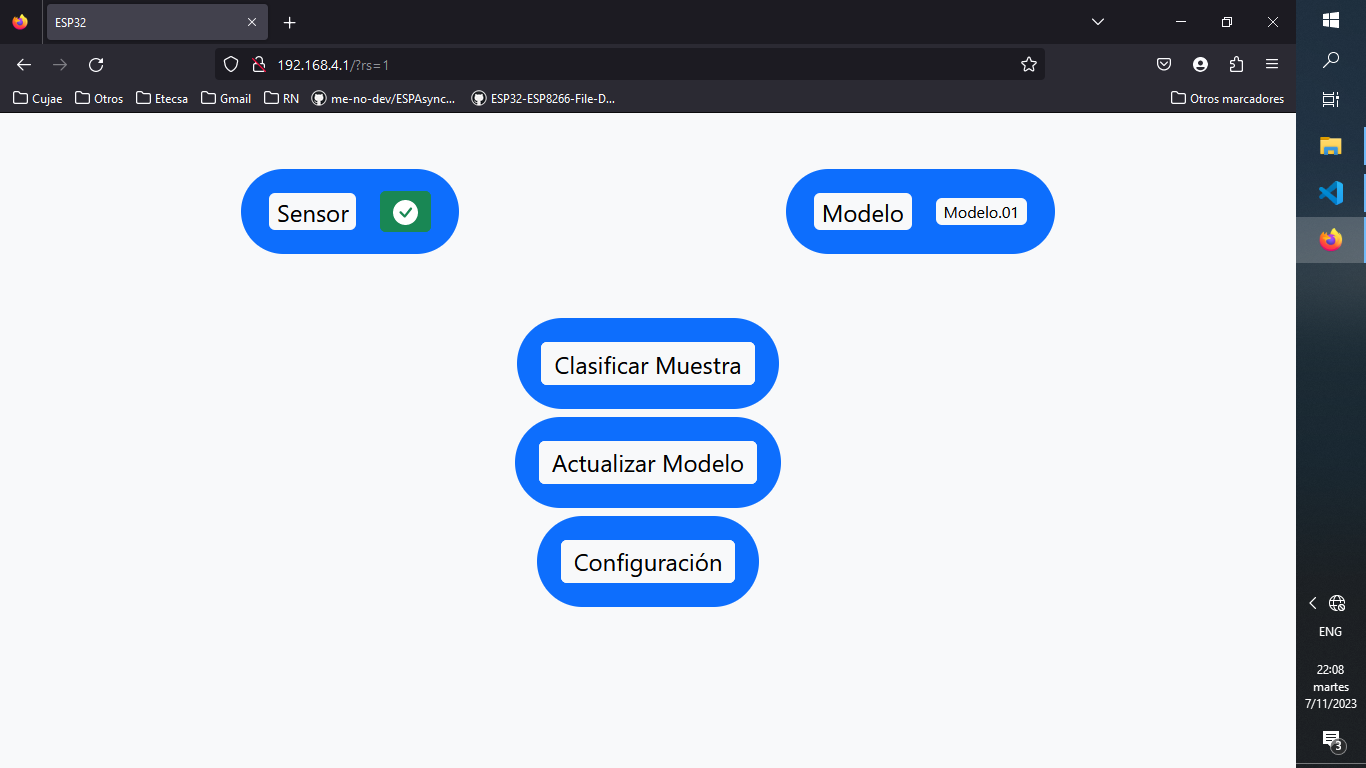


Ilustración 7: Página Web Desplegada por ESP32

Inicialmente el ESP32 arrancara con la Wifi en modo AP, con un usuario y contraseña establecido por defecto o el configurado por el usuario. Al conectarse y acceder a la página se podrá configurar la URL del servidor, el nombre del modelo a actualizar y el SSID y PASSWORD a configurar para la zona Wi-Fi que se utilizar para acceder a la página.

En el proceso de actualización del modelo, se solicitarán los datos del nombre del modelo, y la SSID y Contraseña de una red externa a la cual el ESP32 se conectará, y podrá actualizar el modelo. Para realzar dicha conexión, se detendrá momentáneamente la zona Wi-Fi, por lo tanto, cundo finalice este proceso de actualización, será necesario volver a conectarse y recargar la página, donde se mostrarán los resultados del proceso.

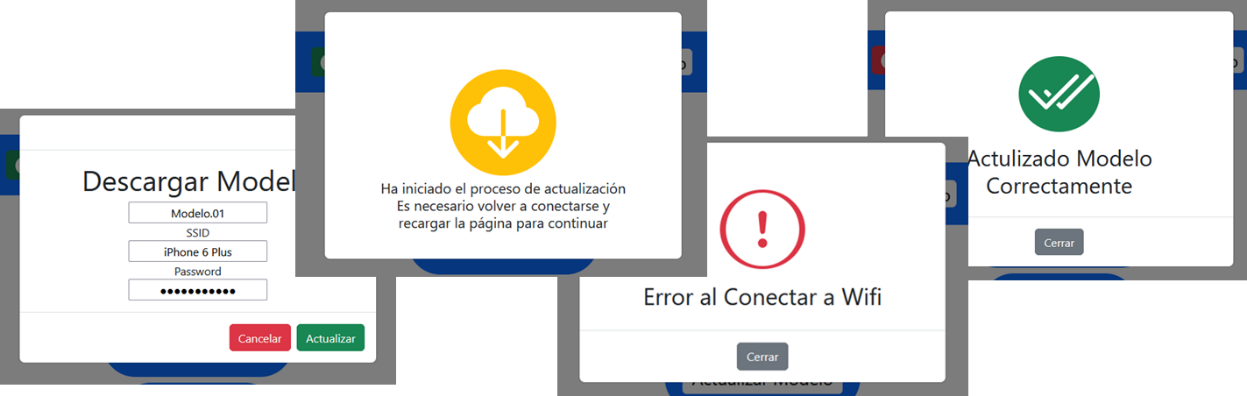


Ilustración 8: Interfaces de actualización de modelo en ESP32

La clasificación de muestras se hace en tiempo real debido a que no leva tanto tiempo de ejecución como la actualización del modelo, por lo tanto, accediendo a” Clasificar Muestra” se podrá iniciar el pocos de predicción, el cual tomará una muestra del sensor la cual será aplicada a la red neuronal para así tomar un valor de salida.



Ilustración 9: Clasificación de una muestra

## 3.4 Arquitectura de Red Neuronal

Como ya se conoce, se utilizará el dispositivo de laboratorio para generar modelos de redes neuronales entrenados para el reconocimiento de adulterantes en la leche. Para su implementación se tendrán 18 neuronas de entrada, referidas a todos los canales obtenidos del sensor, y a la salida se tendrán 5 neuronas con respecto a los siguientes porcentajes de pureza de la leche:

* 100 %
* 75 %
* 50 %
* 25 %
* 0 %

Para obtener modelos con mayor precisión, menor pérdida y tiempo de entrenamiento optimizado, es necesario realizar pruebas con diferentes combinaciones de capas y neuronas en las capas ocultas de una red neuronal. En este proceso de selección, se utilizará la biblioteca Keras-Tuner que permite buscar los hiperparámetros necesarios para la determinación de configuraciones óptimas de funciones de activación, tiempo de aprendizaje y cantidad de neurona en las capas ocultas.

Para realizar la búsqueda, se recolectarán 2500 muestras de leche provenientes de 5 vacas diferentes. Se tomarán 100 muestras de cada etiqueta de salida y se utilizarán estas muestras para buscar los parámetros utilizando el método de Hyperband Search. Se tomarán como constantes los parámetros de optimización, utilizando Adam con entropía cruzada categórica como pérdida. Este método nos permitirá encontrar la configuración óptima de hiperparámetros para nuestro modelo.

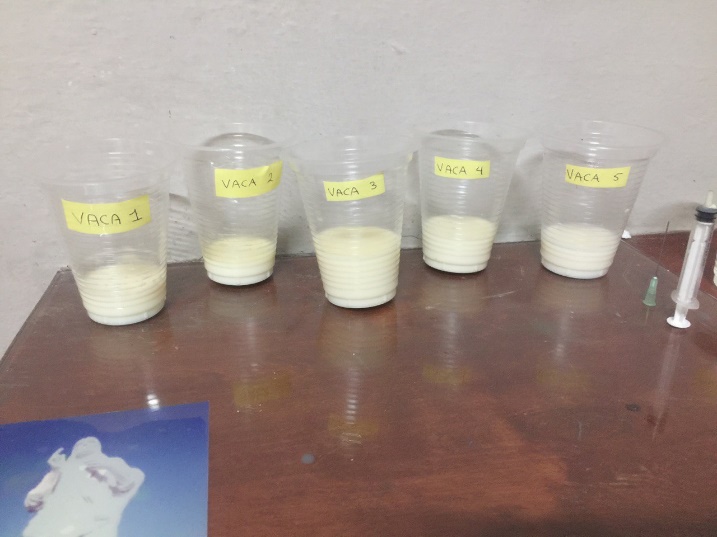


Ilustración 10: Muestras para búsqueda de hiperparametros y entrenamineto



Ilustración 11: Muestras cargadas en base de datos



Ilustración 12: Toma de muestras

Después de 4 minutos y 35 segundos de iniciar el proceso de búsqueda, se obtuvieron los resultados de la búsqueda de hiperparámetros mediante el método de Hyperband Search (ver ilustración ???). Encontramos que la configuración óptima consiste en una capa oculta de 448 neuronas, con una precisión de 0.998 y una pérdida de 0.0199 para los datos de validación tomados, se utilizó la función de activación "sigmoid" y una tasa de aprendizaje de 0.0001. Al analizar las gráficas de precisión/época y pérdida/época (ver ilustración ???), se puede observar que las curvas alcanzan un estado estacionario alrededor de las 500 épocas

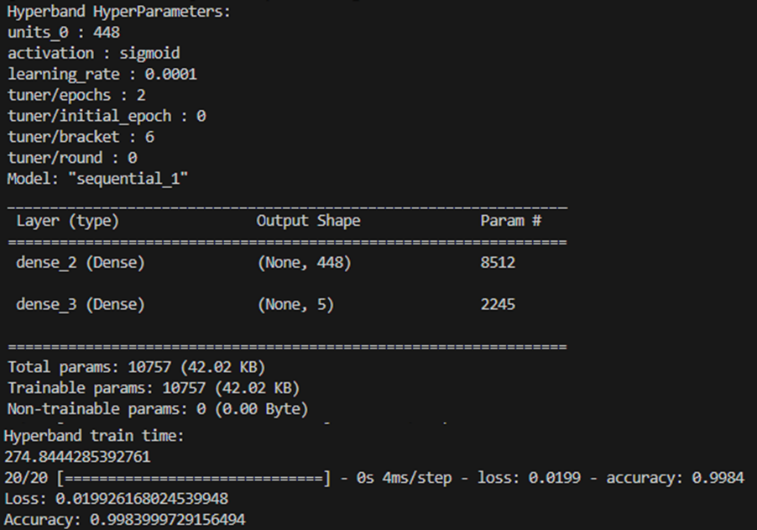


Ilustración 13: Resultados de búsqueda de hiperparámetros

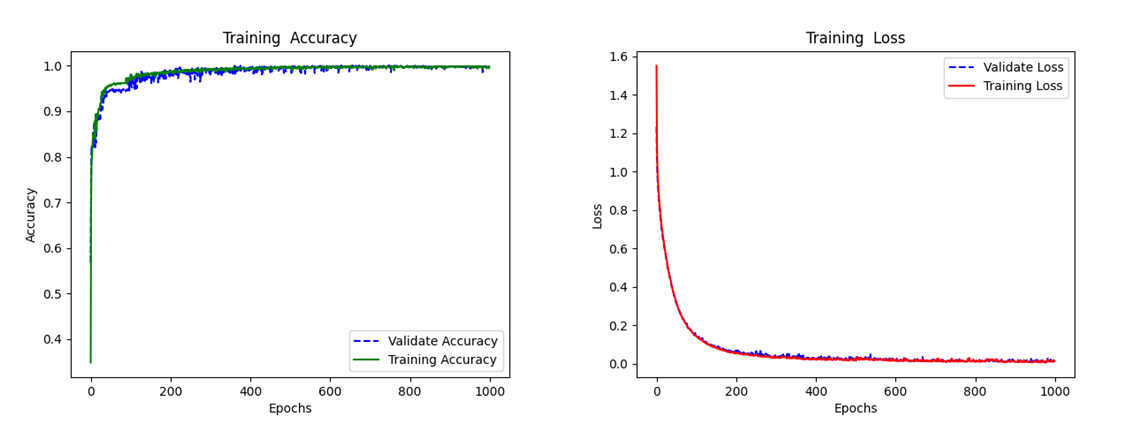


Ilustración 14: Gráficas de Precisión y Pérdida para 1000 épocas

## 3.5 Pruebas y Resultados

Para llevar a cabo pruebas de laboratorio, se procedió a entrenar un modelo de red neuronal utilizando los parámetros específicos obtenidos a través de Keras-Tuner. En esta ocasión, se utilizó un conjunto de 2500 muestras de entrenamiento y se limitó el entrenamiento a solo 500 épocas. Al analizar las gráficas de entrenamiento (ver ilustración ???), se puede observar que la precisión del modelo aumenta y la pérdida disminuye con cada época, lo cual indica que el modelo está aprendiendo correctamente los datos. Para evaluar el desempeño del modelo, se utilizó una matriz de confusión que muestra los resultados de las predicciones realizadas en el conjunto de datos de validación (ver ilustración ???). Este análisis proporciona una visión más detallada sobre cómo el modelo está clasificando las diferentes clases y permite evaluar su rendimiento y capacidad de generalización.

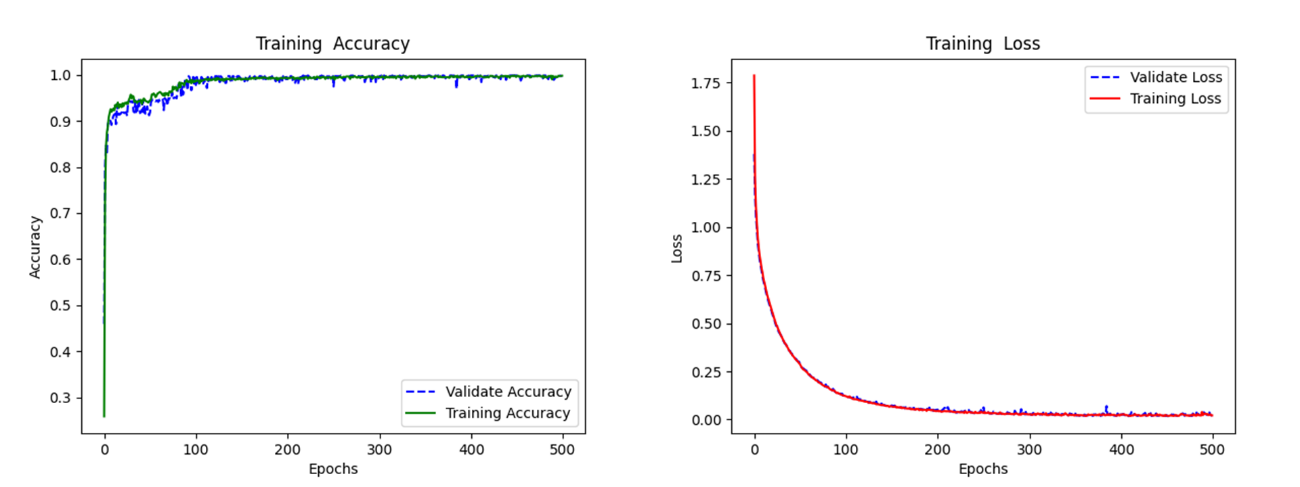


Ilustración 15: Entrenamiento de red neuronal

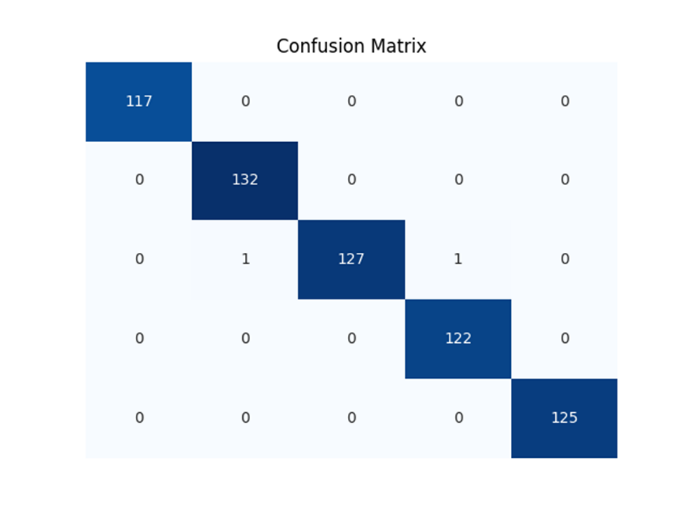


Ilustración 16: Matriz de Confusión de Red Neuronal

Una vez que el modelo ha sido entrenado, se procede a cargar nuevos datos externos al proceso de entrenamiento. Para este propósito, se preparó un fichero CSV con formato 1, que contiene tres muestras de cada etiqueta. Estas muestras se utilizan para realizar predicciones y poner a prueba el modelo. Cada muestra es pasada a través del modelo, el cual emite una predicción y un valor de confiabilidad que indica qué tan acertada fue la respuesta en función de su entrenamiento.

Las pruebas de laboratorio (ver ilustración ???) demuestran que el modelo ha respondido correctamente a todos los datos de muestra, con un alto grado de confiabilidad. Es importante destacar que estas muestras fueron tomadas del mismo conjunto de datos utilizado para el entrenamiento, pero no se incluyeron en dicho proceso.

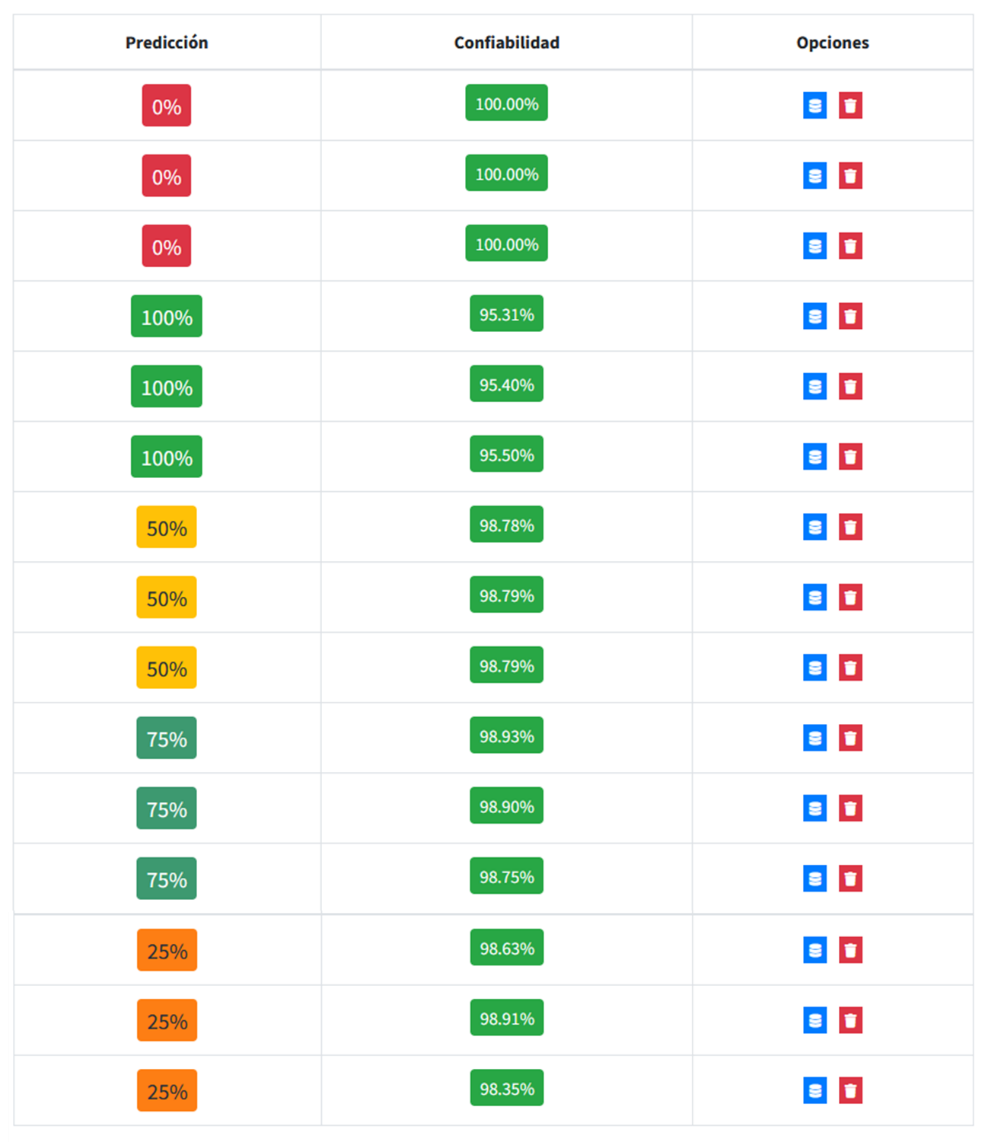


Ilustración 17: Pruebas de Laboratorio

El éxito en la predicción de los nuevos datos externos refuerza la capacidad del modelo para generalizar y realizar predicciones precisas en situaciones no vistas durante el entrenamiento. Esto indica que el modelo ha aprendido patrones relevantes y es capaz de aplicar ese conocimiento a casos similares. Sin embargo, para lograr una predicción aún más precisa y mantener un alto nivel de confiabilidad, es necesario contar con una mayor cantidad de datos de entrenamiento provenientes de diversas fuentes. Esto permitirá abarcar la gran mayoría de casos posibles y capturar una mayor variedad de patrones y características presentes en los datos. Al aumentar la diversidad y cantidad de datos de entrenamiento, el modelo tendrá la oportunidad de aprender con mayor precisión las relaciones y regularidades subyacentes en los datos, lo que resultará en una mejora en su capacidad de predicción y confiabilidad. Por lo tanto, es importante continuar recopilando y agregando datos de entrenamiento de calidad a fin de mejorar aún más el desempeño del modelo.

# Referencias

# Anexos