UNIVERSIDAD DE LAS FUERZAS ARMADAS - ESPE

DEPARTAMENTO DE ELÉCTRICA, ELECTRÓNICA Y TELECOMUNICACIONES

CARRERA DE INGENIERÍA EN ELECTRÓNICA, AUTOMATIZACIÓN Y CONTROL

"IMPLEMENTACIÓN DE UN SISTEMA DE CLASIFICACIÓN BINARIA BASADO EN EL USO DE SERVICIOS WEB COGNITIVOS Y REDES NEURONALES PROFUNDAS"

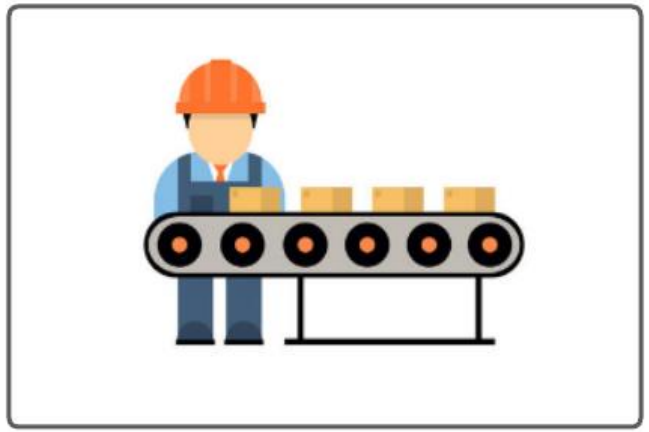
AUTORES: CHANCUSIG CASA, CRISTIAN ALEXANDER
TUMBACO CASA, SERGIO DAVID

DIRECTOR: ING. ALULEMA FLORES, DARWIN OMAR, PhD.

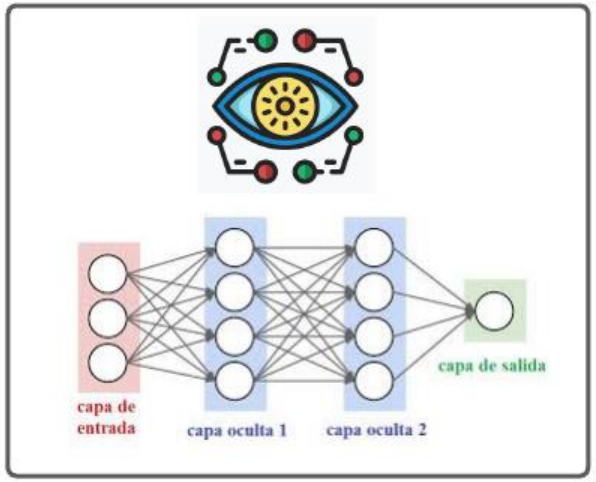
Contenido

Introducción

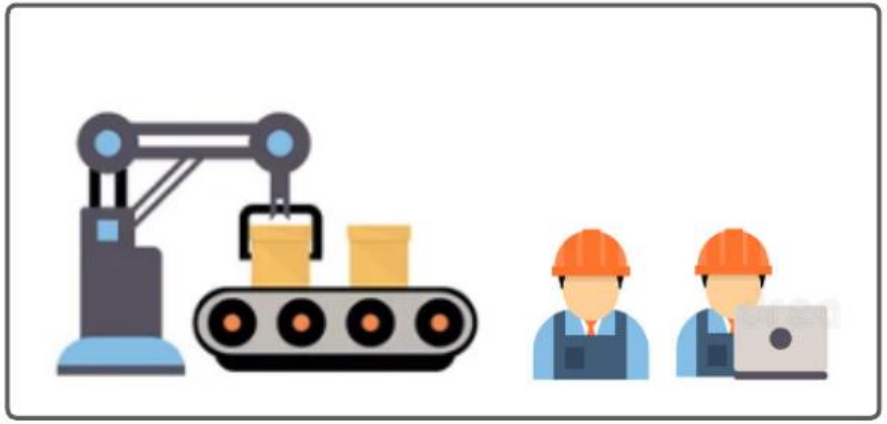
Clasificación manual



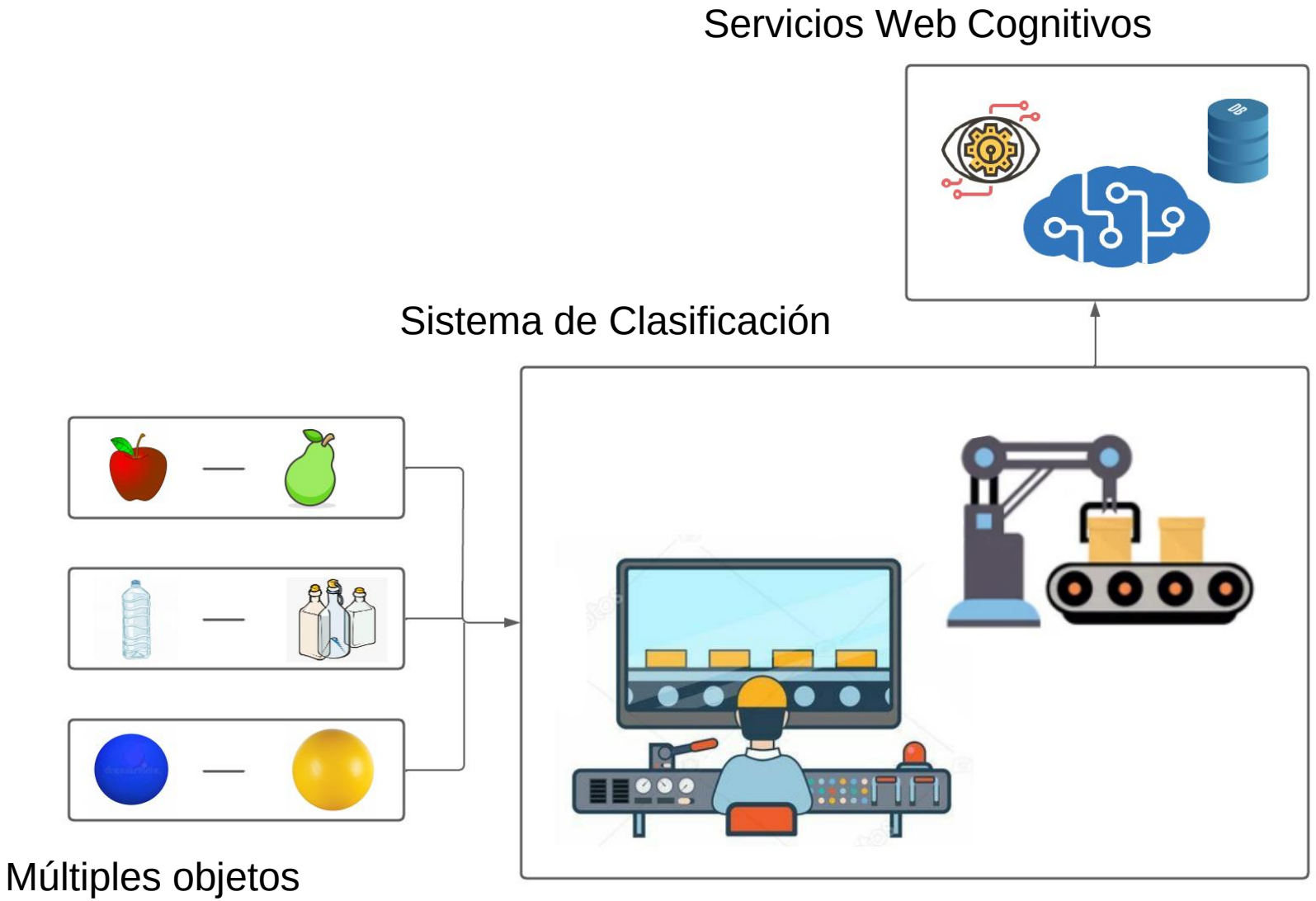
Visión por computadora



Clasificación automática



Introducción



Objetivos

General

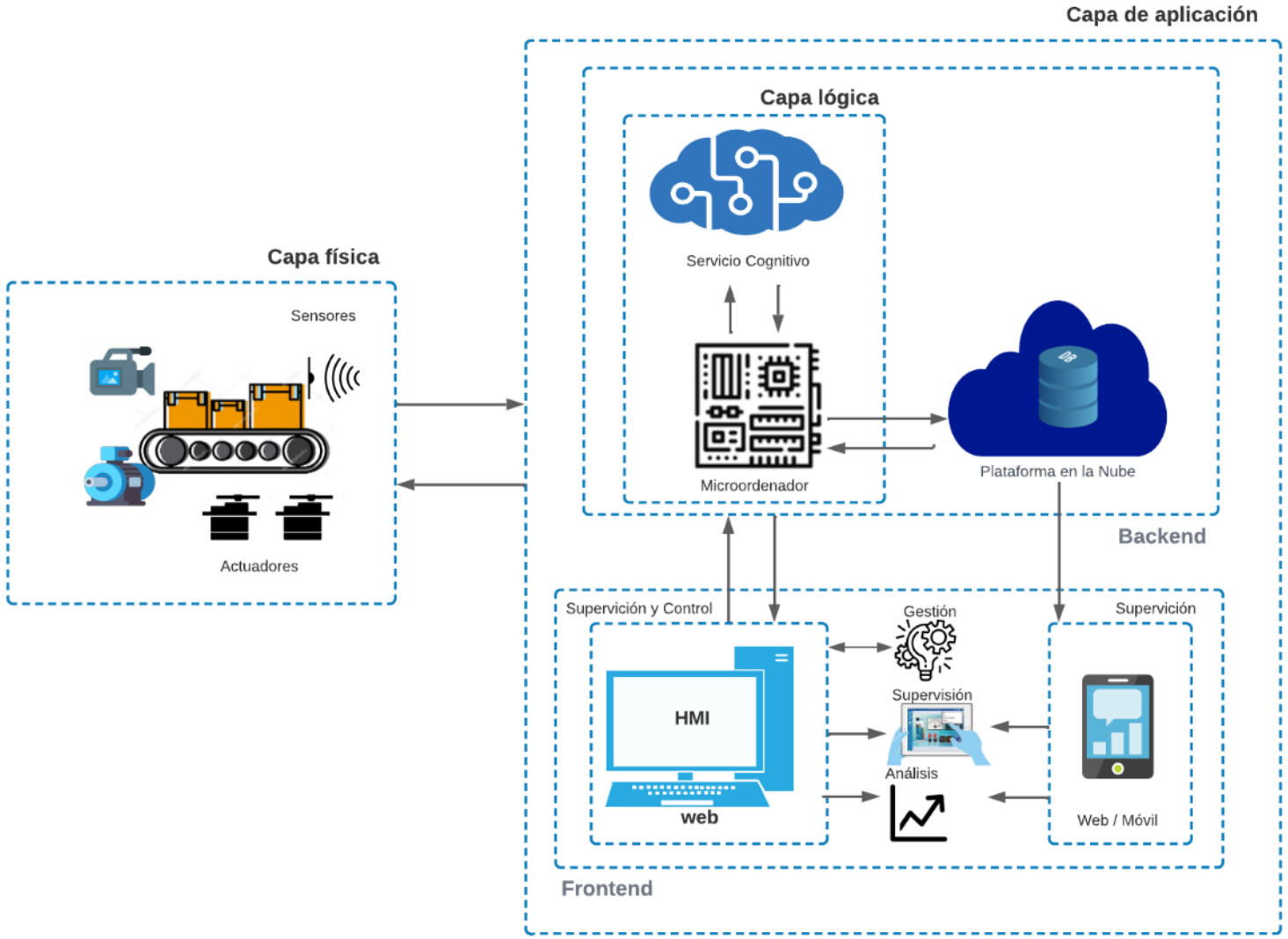
Implementar un prototipo de un sistema de clasificación binaria basado en el uso de servicios web cognitivos y redes neuronales profundas.

Específicos

- Identificar fuentes de información relevantes para el desarrollo del estado del arte para sistemas de clasificación con imágenes.
- Determinar que metodologías se usarán para la aplicación de los algoritmos de identificación y clasificación basados en el consumo de servicios web cognitivos.
- Ajustar un modelo para su uso en el sistema de clasificación de múltiples objetos.
- Diseñar una interfaz que permita el uso del modelo de clasificación.
- Diseñar un prototipo mecánico y electrónico, tal que, su estructura permita la clasificación de objetos en dos grupos determinados.
- Generar una arquitectura de integración de los componentes mecánicos y electrónicos con los servicios en la nube para su gestión mediante una interfaz web/móvil.
- Evaluar el sistema mediante pruebas de carga, usabilidad.

Diseño

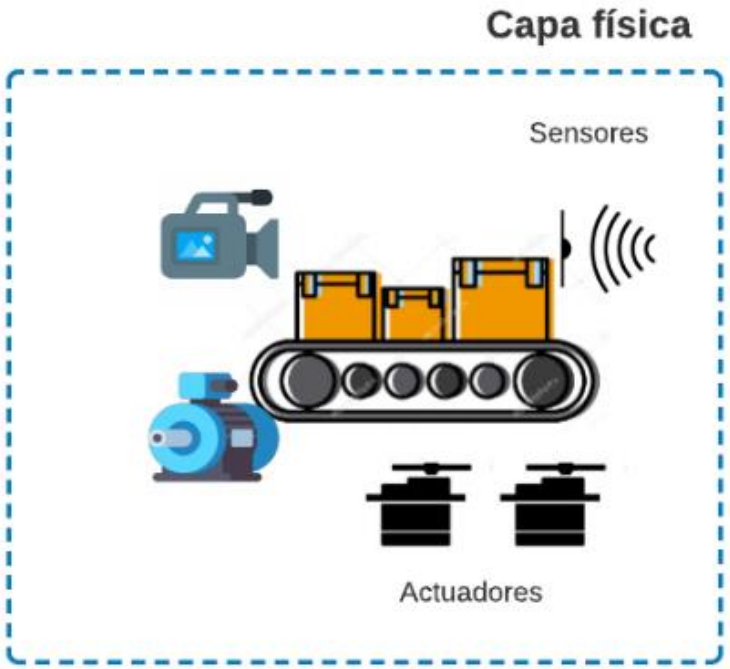
Arquitectura del sistema



Diseño

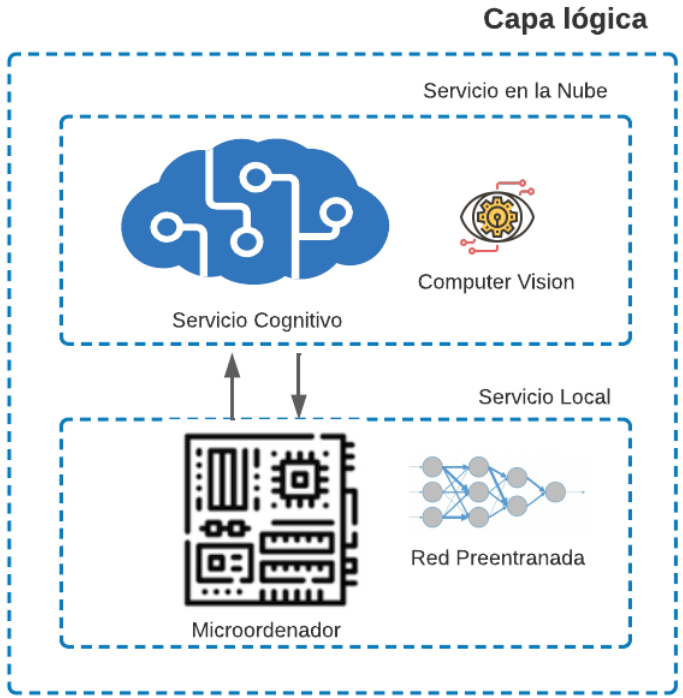
Capa Física

- Se implementa la estructura con los actuadores y sensores para la toma de datos.
- La configuración de los puertos de entrada y salida del microordenador se lo realiza desde el Backend.



Capa Lógica

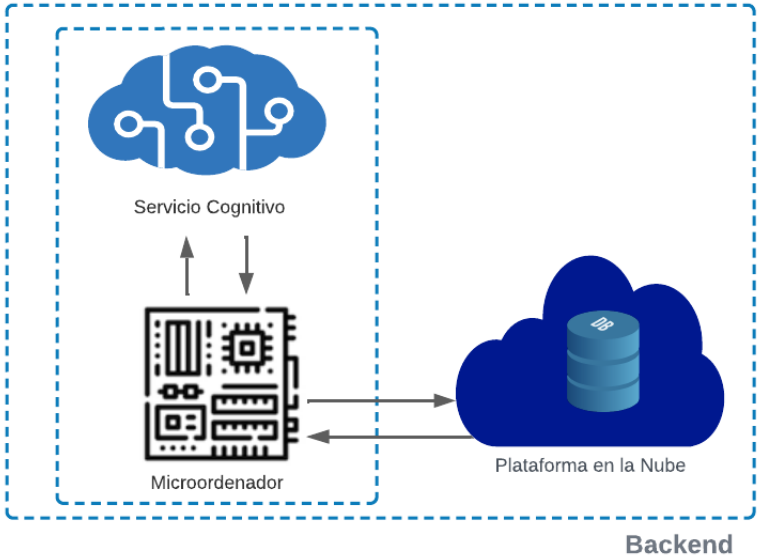
- Se establece la conexión con el servicio web cognitivo.
- Se desarrolla aprendizajes para nuevos modelos con objetos que no se pudieron identificar.

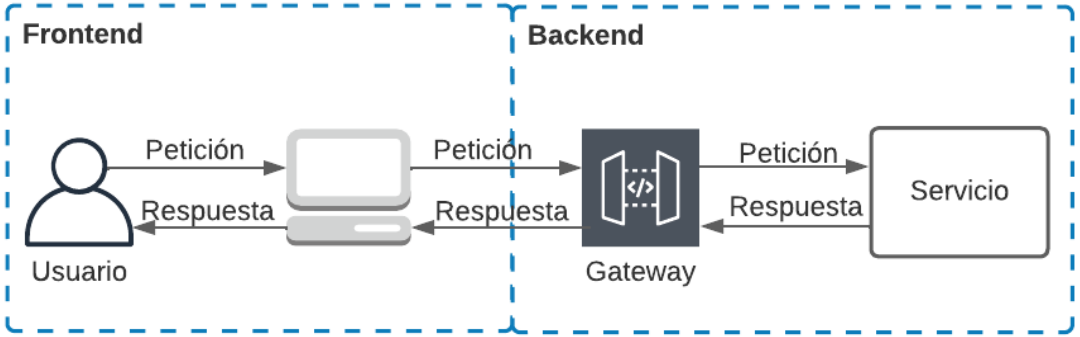


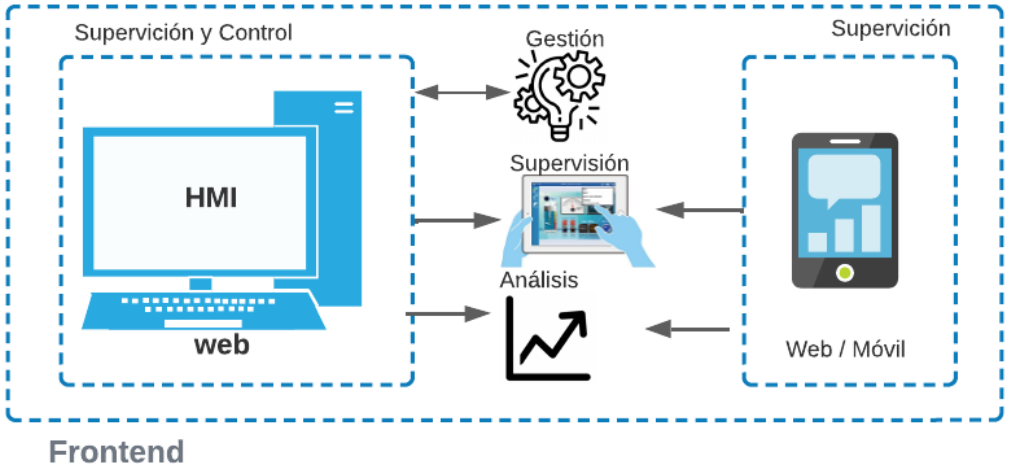
Diseño

Capa de Aplicación

- Permite la comunicación de los datos con la capa física.
- El desarrollo de las interfaces se dividen en dos partes principales, la interfaz local para el control y supervisión del sistema, y la interfaz remota para la supervisión.
- La estructura del software se dividen en Backend y Frontend.



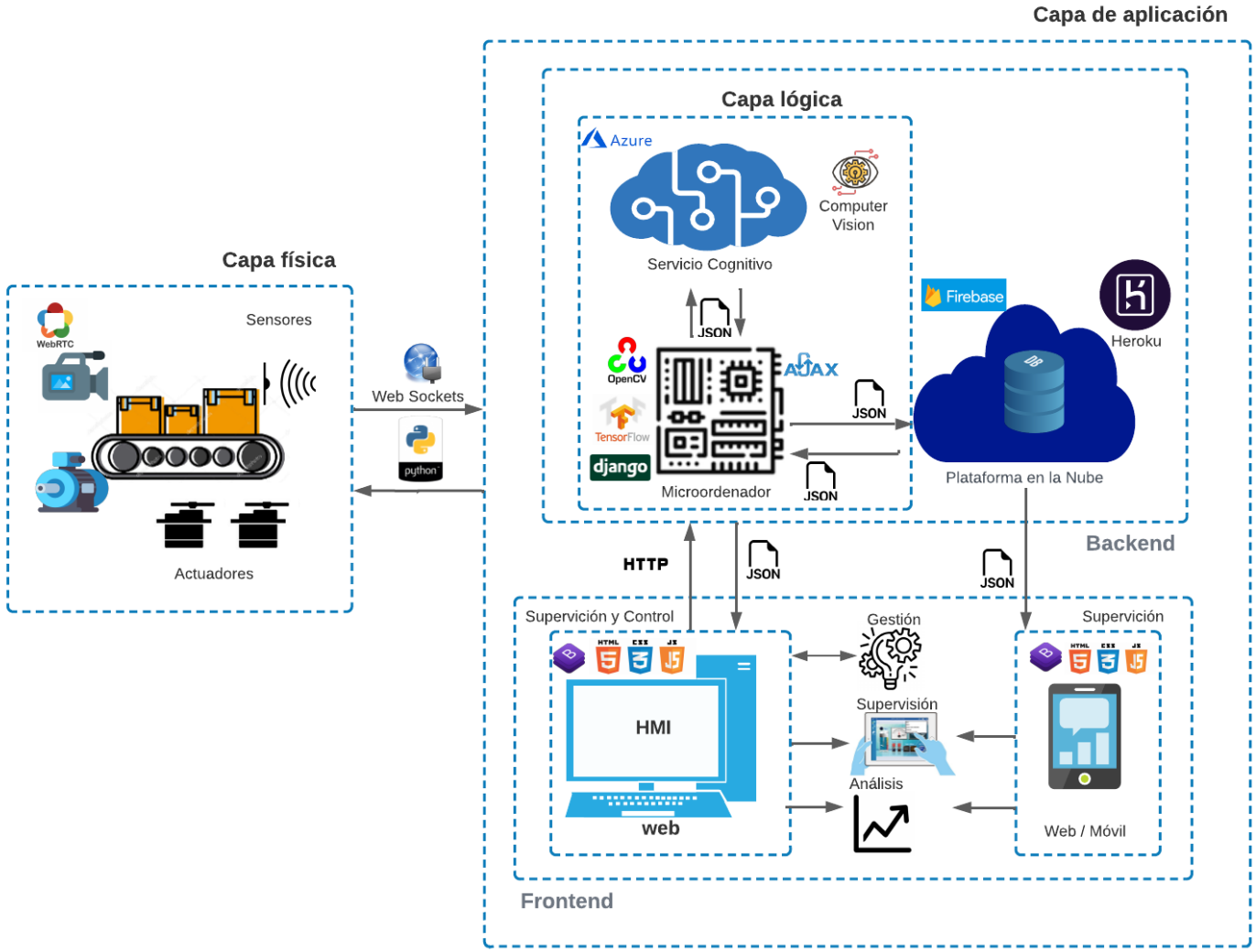




Implementación

Arquitectura del sistema detallada

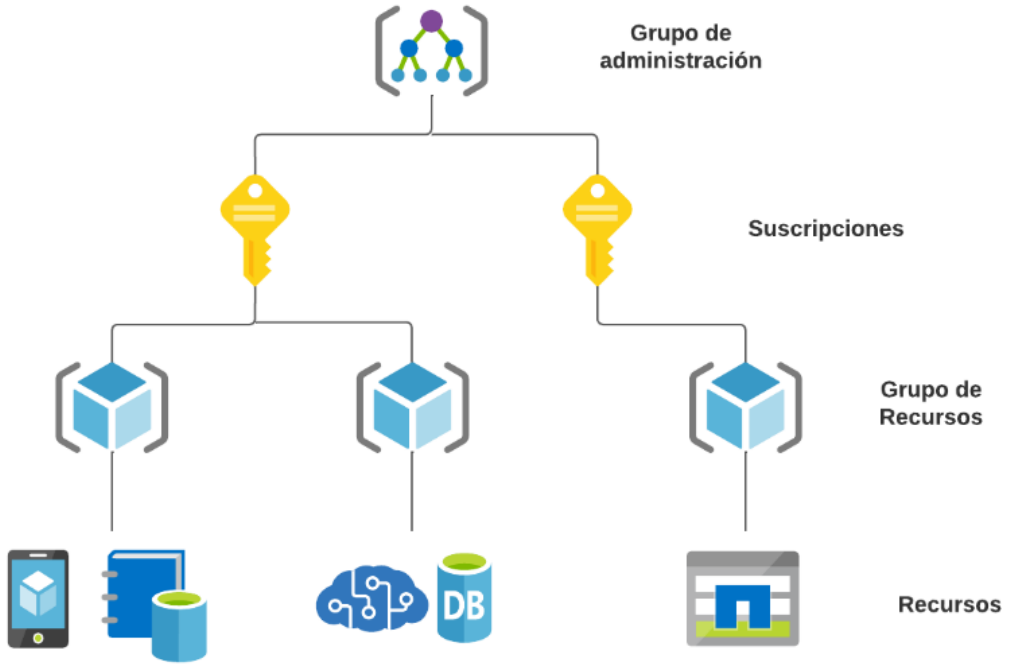
- Se establecen protocolos de comunicación para el envío y recepción de datos entre capas.
- Selección de herramientas para la estructura de programación.
- Selección de PaaS para funcionalidades como base de datos y de servidores en la nube.
- Selección de herramientas de diseño para la generación de interfaces.



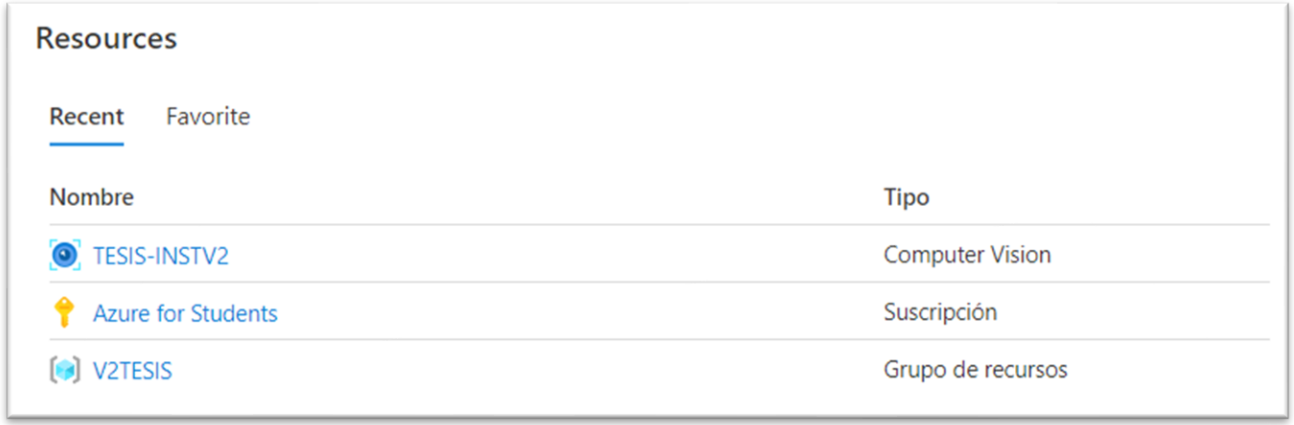
Implementación

Capa Lógica

Servicio en la nube – Microsoft Azure



- Se crea una suscripción para el acceso a un grupo de recursos, y en este administrar cada recurso de manera individual. Para este proyecto, se accede al servicio cognitivo Computer Vision, y a la FaaS de etiquetas de imágenes, para acceder a las funcionalidades de la API



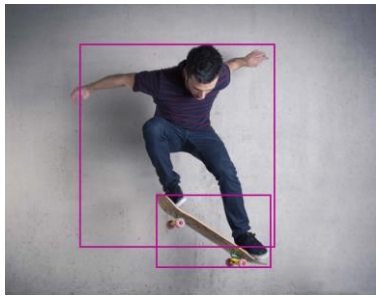
Implementación

Capa Lógica

Servicio en la nube – FaaS tag image

- Genera la lista de etiquetas relevantes del contenido de la imagen que fue ingresada. Se admiten dos métodos de entrada: el primero mediante la carga de una imagen o el segundo al especificar una URL de imagen. Se devolverá una respuesta exitosa en JSON.

```
tag_image_in_stream(image, language='en’,
      model_version='latest', custom_headers=None, raw=False,
      callback=None, **operation_config)
```



Función	Valor
Objetos	[{ "rectangle": { "x": 238, "y": 299, "w": 177, "h": 117 }, "object": "Skateboard", "confidence": 0.903 }, { "rectangle": { "x": 118, "y": 63, "w": 305, "h": 321 }, "object": "person", "confidence": 0.955 }]
Etiquetas	[{ "name": "skating", "confidence": 0.999951363 }, { "name": "snowboarding", "confidence": 0.9893889 }, { "name": "sports equipment", "confidence": 0.9722208 }, { "name": "person", "confidence": 0.959769964 }, { "name": "roller skating", "confidence": 0.946092963 }, { "name": "skiing", "confidence": 0.92313683 }, { "name": "man", "confidence": 0.9193816 }, { "name": "outdoor", "confidence": 0.9109124 }, { "name": "boardsport", "confidence": 0.907244742 }, { "name": "riding", "confidence": 0.8984571 }, { "name": "sport", "confidence": 0.871290743 }]
Formato	“Jpeg”

Implementación

Capa Lógica

Servicio Local – Transfer Learning

Crear una instancia de un modelo base, y cargar los pesos previamente entrenados.

Congelar todas las capas en el modelo base configurado en el paso anterior

Crear un nuevo modelo de una o varias capas sobre el modelo base.

Entrenar el modelo creado con el conjunto de datos nuevo

```
resnet_model = Sequential()
pretrained_model =
      tf.keras.applications.ResNet50(include_top=False,
input_shape=(img_height, img_width, 3), pooling='avg',
      weights='imagenet')
```

```
for layer in pretrained_model.layers:
        layer.trainable = False
```

```
resnet_model.add(pretrained_model)
    resnet_model.add(Flatten())
    resnet_model.add(Dense(512, activation='relu'))
    resnet_model.add(Dense(2, activation='softmax'))
```

```
resnet_model.compile(optimizer='adam',loss='categorical_
      crossentropy', metrics=['accuracy'])
resnet_model.fit(train_ds,validation_data=val_ds,
      epochs=7)
```

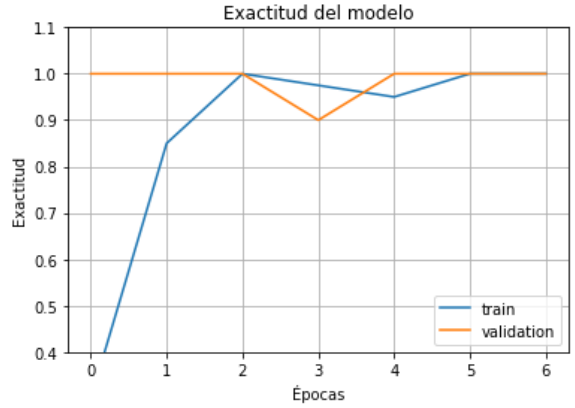
Implementación

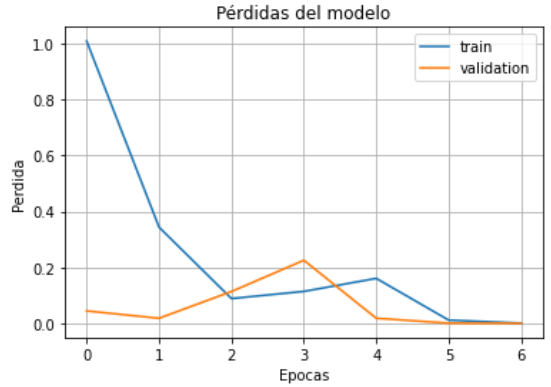
Capa Lógica

Servicio Local – Desempeño del modelo

- El *accuracy* o exactitud del modelo tiende a 1 a partir de la época 5 a 6 para cada entrenamiento
- La función de perdida tiende a 0 a partir de la época 6 de cada entrenamiento.
- Se busca la reducción de procesamiento local.

	Num. de imágenes	Épocas
Entrenamiento 1	60	10
Entrenamiento 2	60	7
Entrenamiento 3	50	10
Entrenamiento 4	50	7

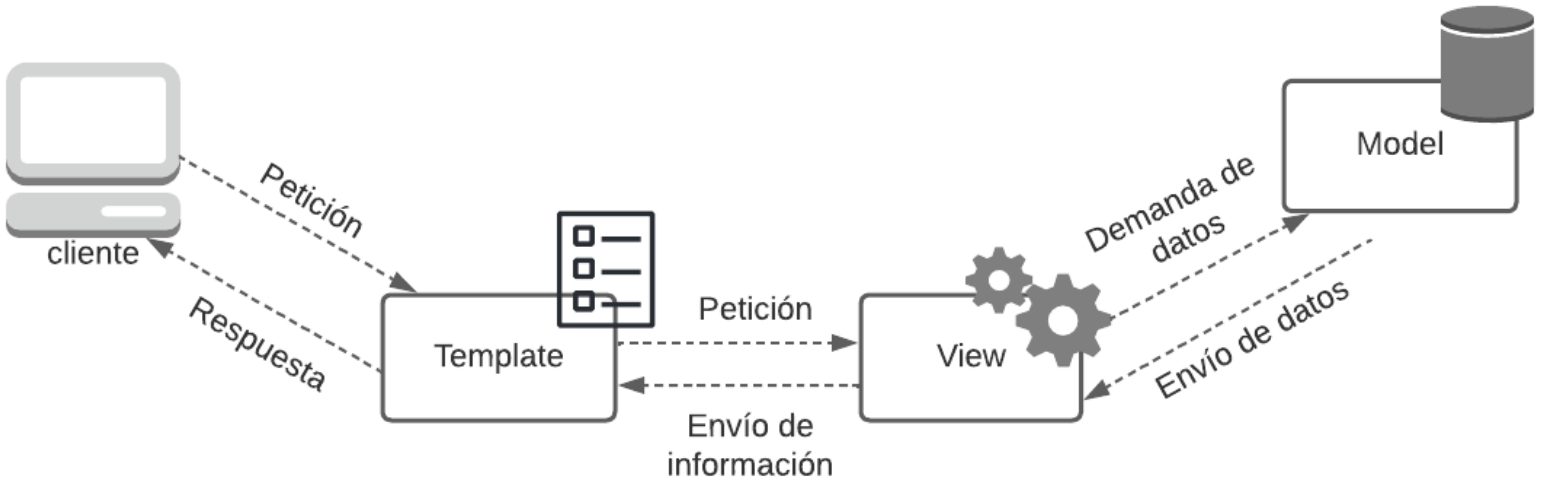




Implementación

Capa de Aplicación - Backend

Estructura del programa



Implementación

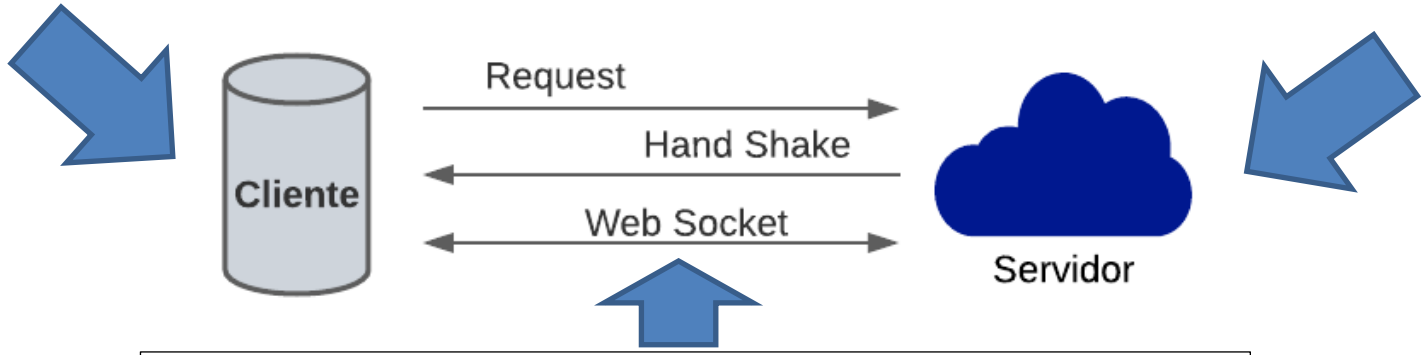
Capa de Aplicación - Backend

Recepción de datos

```
      var socket = new WebSocket('ws://localhost:8000/ws/redAzure/')
      socket.onmessage = function (event) {
        var data = JSON.parse(event.data);
        if (data.banF) {
          document.querySelector('#predicAzure').innerText =
data.message;
          document.getElementById("imgAzure").src = data.fotoA;
          document.querySelector('#contAzObj1').innerText =
data.contAzObj1;
        actuadores(data.banda,'idEstadoBanda',"ENCENDIDA","APAGADA");
       }
```

Envío de datos al Frontend

```
      class WSConsumerAzure(AsyncWebsocketConsumer):
          async def connect(self):
              await self.accept()
              while True:
                  if (GPIO.input(23) == 0):
                      apagarBanda()
      datosEnv={'message':prediccionRed1,'fotoA':dirFoto,'contAzObj1':co
ntObj1,'contAzObj2'……}
                  await self.send(json.dumps(datosEnv))
                  await sleep(0.1)
```




```
      ws_urlpatterns = [
          path('ws/redAzure/', WSConsumerAzure.as_asgi()),
          path('ws/redResnet/', WSConsumerResnet.as_asgi()),
      ]
```

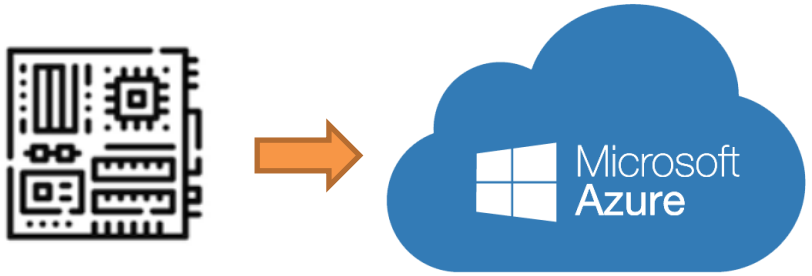
Enlaces de conexión entre capas

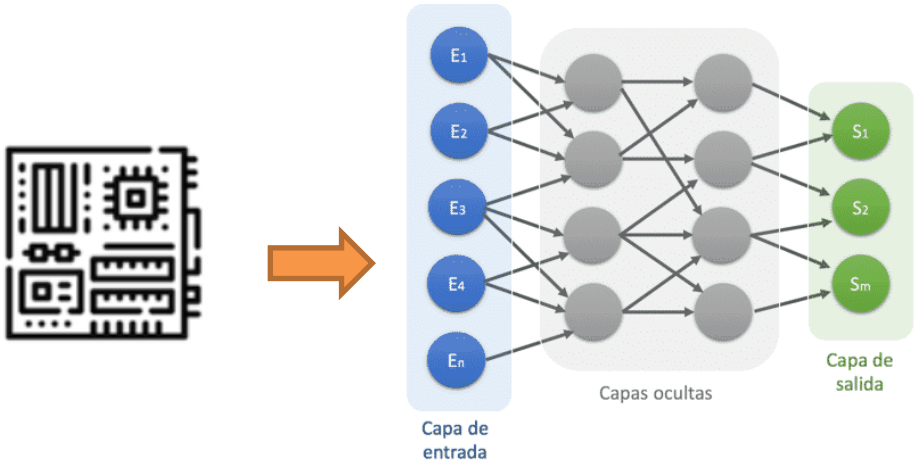
Implementación

Capa de Aplicación - Backend

Uso del servicio web

```
        # Llamar a la API Azure
        tags_result_remote = cv_client.tag_image(image_stream, language="es")
        # Obtención de predicción de Azure y confianza
        if (len(tags_result_remote.tags) == 0):
            aviso = "No hay objetos detectado."
        else:
            for tag in tags_result_remote.tags:
                tagsOb.append(tag.name.capitalize())
    return tagsOb
```





Uso del servicio local

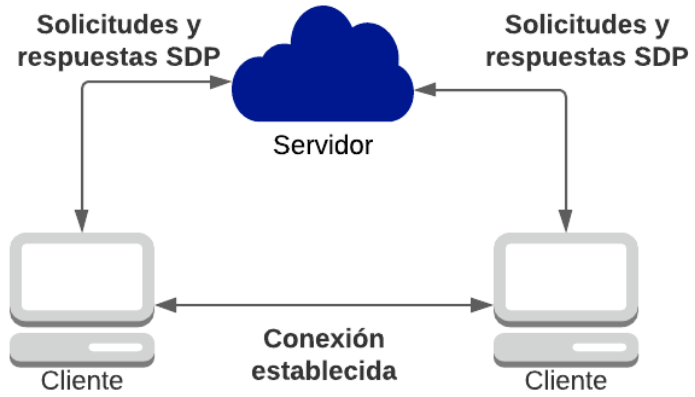
```
    image = cv2.imread(direccionF)
    image_resized = cv2.resize(image, (img_height, img_width))
    image = np.expand_dims(image_resized, axis=0)
    pred = modelo_final.predict(image)
    prediccionR = class_name[np.argmax(pred)]
```

Implementación

Capa de Aplicación - Backend

WebRTC

```
function startStreaming() {
  idcamera = 0
  var mediaSupport = 'mediaDevices' in navigator;
  if (mediaSupport && null == cameraStream) {
    navigator.mediaDevices.getUserMedia({
        video: {
          deviceId: myVideoInputs[idcamera].deviceId
        }
      })
      .then(function (mediaStream) {
        cameraStream = mediaStream;
        stream.srcObject = mediaStream;
        stream.play();
      })
      .catch(function (err) {
        console.log("Acceso denegado a la cámara: " + err);
      });
  } else {
    alert('Está en uso la cámara.');
    return;
  }
}
```



Implementación

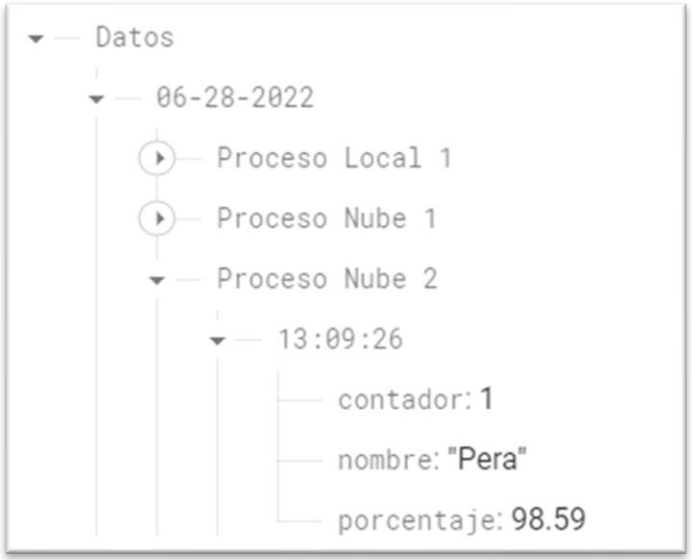
Capa de Aplicación - Backend

Firebase

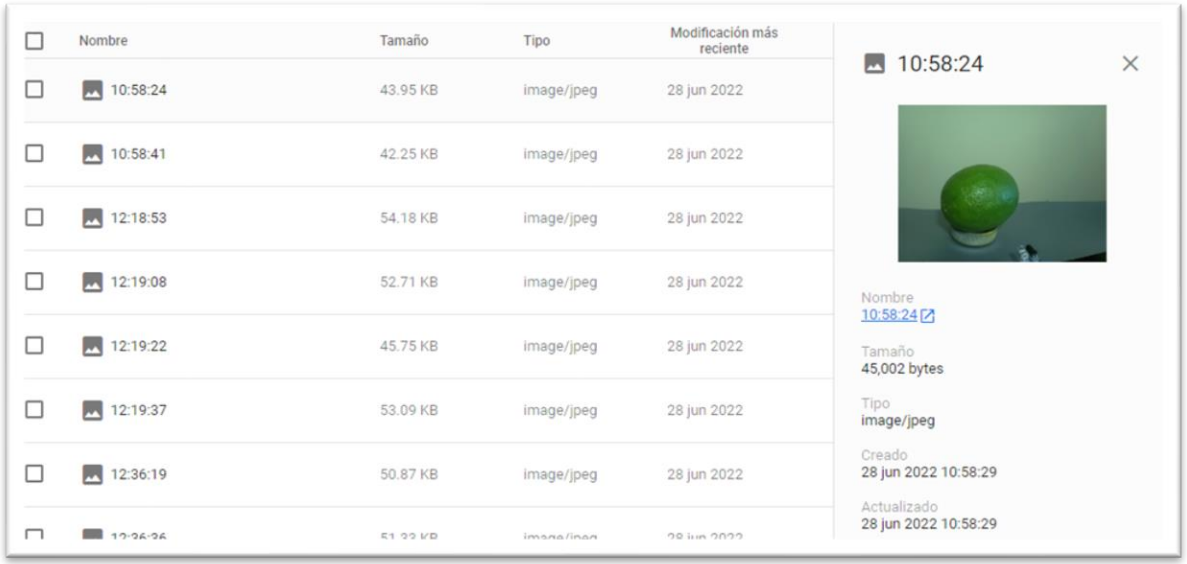
```
storage.child(imgFbase).put(direccionF)
urlImg = storage.child(imgFbase).get_url(None)
data={"nombre":nombre,'contador':contador,'porcentaje':porcentaje}
    database.child('Datos').child(fecha).child(procesoA).child(hora).set(data)
```

Firebase

Realtime



Storage



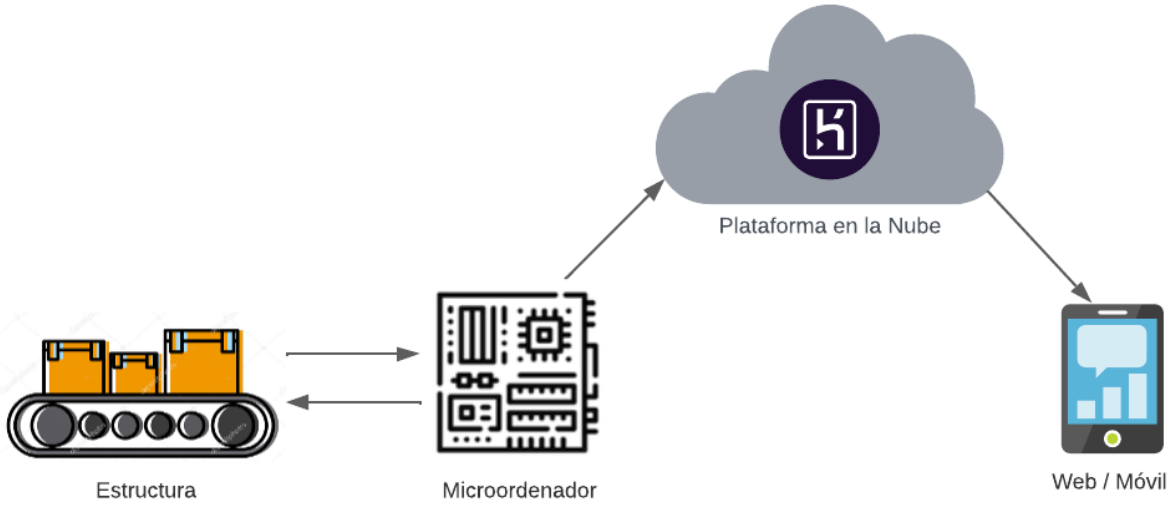
ESPE
UNIVERSIDAD DE LAS FUERZAS ARMADAS
INNOVACIÓN PARA LA EXCELENCIA

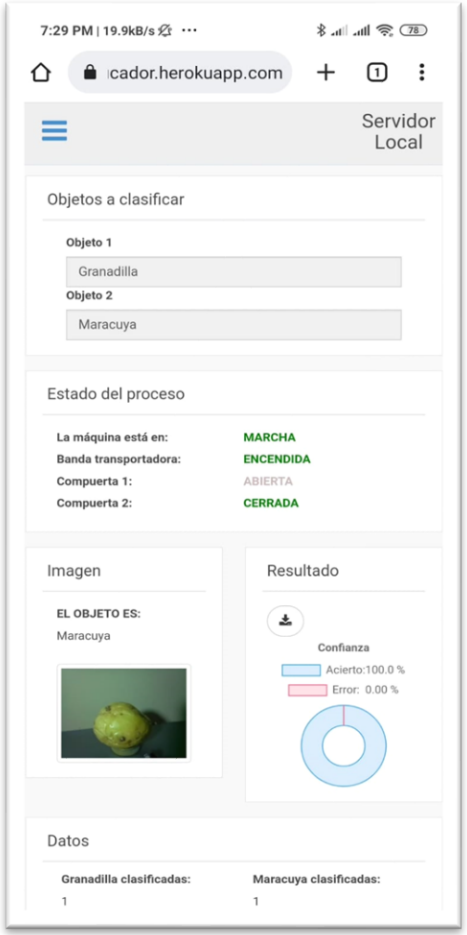
Implementación

Capa de Aplicación - Backend

Heroku

https://sistemaclasificador.herokuapp.com/

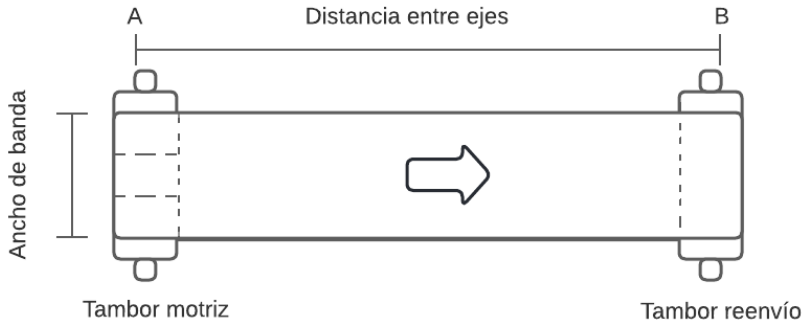


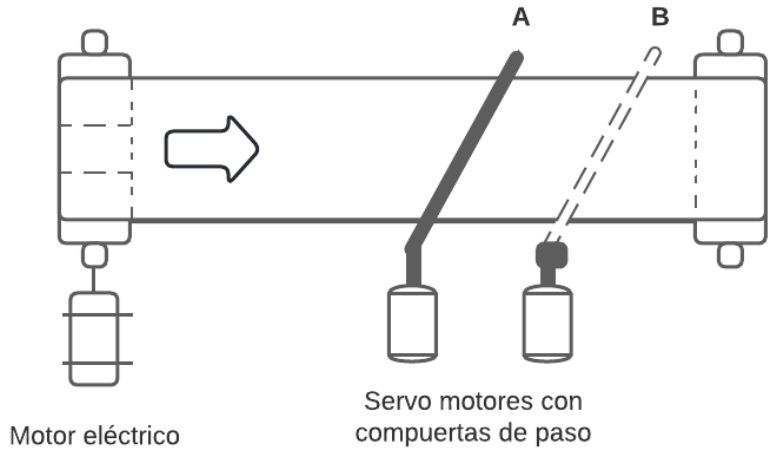


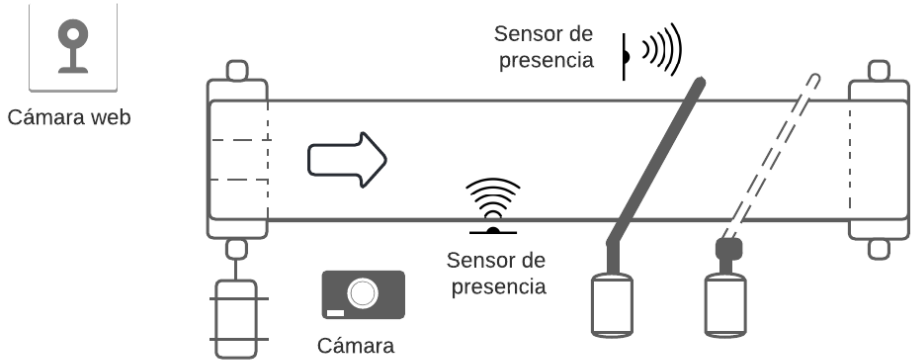
Implementación

Capa Física

Estructura mecánica



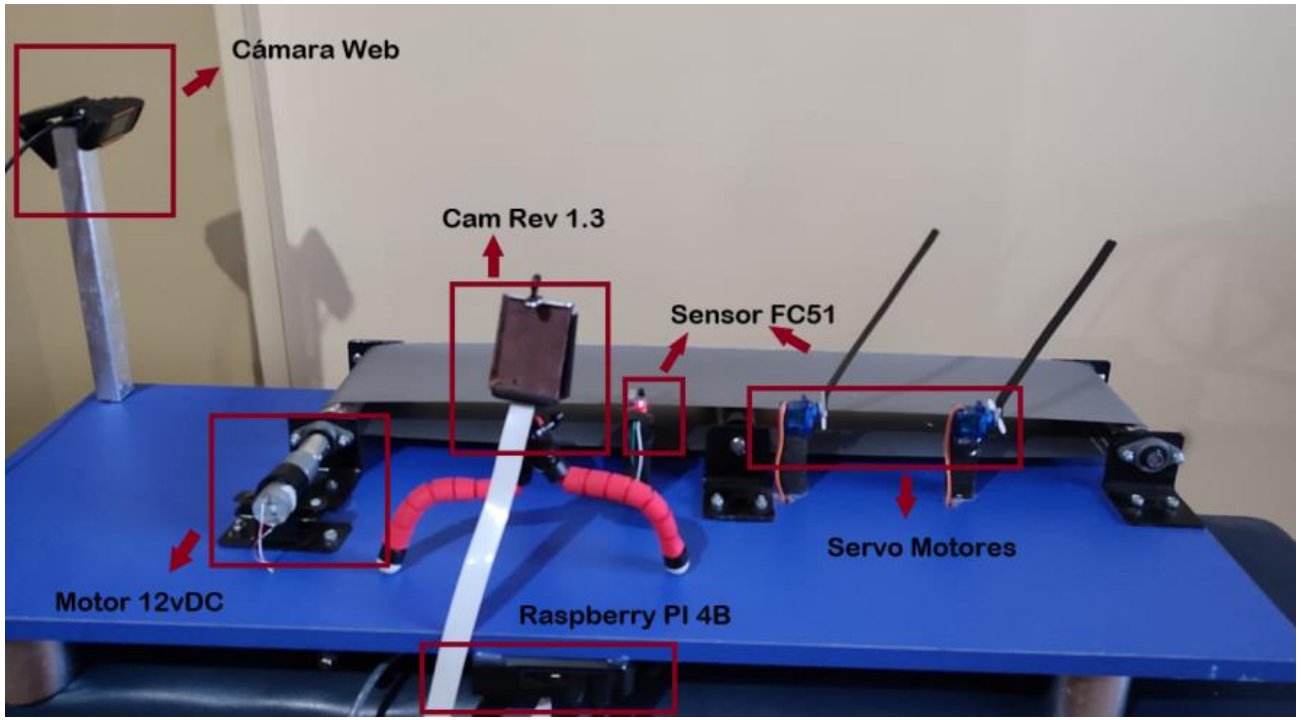




Implementación

Capa Física

Estructura mecánica – electrónica



Elemento	Función
Motor 12V DC 60 Rpm	Movimiento tambor motriz
Sensor FC 51	Detector de presencia
Cámara Rev 1.3	Captura de imagen para predicción
Servo SG90	Compuertas
Cámara Web 720p	Video Stream
Raspberry PI 4B	Microrordenador

Pruebas y Resultados

Pruebas de funcionamiento

Vídeo

Pruebas y Resultados

Pruebas de funcionamiento

Parámetros de desempeño

Actual	**Positivo**	Verdaderos Positivos (VP)	Falsos Negativos (FN)
	Negativo	Falsos Positivos (FP)	Verdaderos Negativos (VN)
		Positivo	**Negativo**
		Predicción	

$$Exactitud = \frac{VP + VN}{Total}$$

$$Precisión = \frac{VP}{VP + FP}$$

$$Tasa\ de\ error = \frac{FP + FN}{Total}$$

$$VPN = \frac{VN}{VN + FN}$$

$$Sensibilidad = \frac{VP}{VP + FN}$$

$$Especificidad = \frac{VN}{VN + FP}$$

$$VF = 2 * \frac{precisión * exhaustividad}{precisión + exhaustividad}$$

Pruebas y Resultados

Pruebas de funcionamiento

Parámetros de desempeño

Servicio en la Nube

		Exactitud [%]	Tasa de error [%]	Sensibilidad [%]	Especificidad [%]	Precisión [%]	VPN [%]	Valor F [%]
Forma	***Prueba 1***	95	5	100	90	91	100	95
	Prueba 2	95	5	100	90	91	100	95
Textura	***Prueba 3***	90	10	90	90	90	90	90
	Prueba 4	100	0	100	100	100	100	100
Color	***Prueba 5***	100	0	100	100	100	100	100

Servicio Local

	Exactitud [%]	Tasa de error [%]	Sensibilidad [%]	Especificidad [%]	Precisión [%]	VPN [%]	Valor F [%]
Prueba 1	95	5	100	90	91	100	95
Prueba 2	95	5	90	100	100	91	95
Prueba 3	95	5	100	90	91	100	95

Pruebas y Resultados

Pruebas de usabilidad

Sistema de Escalas de Usabilidad (SUS)

Preguntas	
Pregunta 1	¿Creo que me gustaría usar este sistema con frecuencia?
Pregunta 2	¿Encontré el sistema innecesariamente complejo?
Pregunta 3	¿Pensé que el sistema era fácil de usar?
Pregunta 4	¿Creo que necesitaría el apoyo de un técnico para poder utilizar este sistema?
Pregunta 5	¿Descubrí que las diversas funcionalidades de este sistema estaban bien integradas?
Pregunta 6	¿Pensé que había demasiadas inconsistencias en este sistema?
Pregunta 7	¿Me imagino que la mayoría de la gente aprendería a usar este sistema muy rápidamente?
Pregunta 8	¿Encontré el sistema muy engorroso de usar?
Pregunta 9	¿Me sentí muy confiado usando el sistema?
Pregunta 10	¿Necesitaba aprender muchas cosas antes de poder ponerme en marcha con este sistema?

Preguntas / Usuarios	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	Puntaje SUS
U1	4	1	5	1	4	1	5	1	5	1	95,0
U2	3	3	3	2	3	3	5	3	5	2	65,0
U3	4	1	5	1	5	1	5	1	5	1	97,5
U4	5	1	5	1	5	1	5	1	5	1	100,0
U5	4	1	4	1	5	1	4	2	5	1	90,0
U6	5	1	5	1	4	1	4	2	5	2	90,0
U7	4	1	5	1	4	1	5	2	5	1	92,5
U8	4	1	5	1	4	1	5	1	4	1	92,5
U9	5	1	4	1	5	1	4	2	5	2	90,0
U10	5	1	4	1	4	1	5	2	5	1	92,5
							Promedio:				90,5

Conclusiones y Trabajos Futuros

Conclusiones

- La investigación realizada en este proyecto permitió determinar una metodología válida para la implementación de un sistema de clasificación binario. Esta consume el servicio web cognitivo ofrecido por Microsoft Azure denominado Computer Vision. A través de la FaaS de etiquetas de imágenes accede a las funcionalidades de la API para obtener etiquetas en formato JSON que son identificadas como relevantes dentro del contenido de la imagen que es capturada y suministrada al sistema, y poder así ser usadas para el proceso de identificación y clasificación.
- Para cubrir la necesidad de clasificación de objetos que no son identificados o son identificados de manera incorrecta por el servicio en la nube, se implementa una red neuronal convolucional, la cual generará un modelo de aprendizaje para cada nuevo par de objetos que se desea clasificar. Para ello es necesario el uso de la técnica de transferencia de aprendizaje, reduciendo así los requerimientos de procesamiento del microordenador. La red neuronal preentrenada ResNet 50 logró cubrir esta necesidad.
- El modelo de clasificación empleado por el servicio en la nube presenta valores de exactitud y de valor F de entre el 90 % al 100 %, determinando que el desempeño en su proceso de clasificación dentro del sistema es correcto para grupos en los que los objetos a clasificar pertenecen a una categoría más general. Así también, los modelos de clasificación generados por parte del servicio local presentan valores de exactitud y valor F del 95 % concluyendo que el desempeño es adecuado para cuando los objetos a clasificar pertenezcan a categorías más específicas o con características más distintivas, como los presentados en las pruebas de funcionalidad.

Conclusiones y Trabajos Futuros

Conclusiones

- La estructura de Django junto con el lenguaje de programación Python permitieron la integración de las herramientas y servicios del sistema, el Backend como administrador de funciones, y el Frontend para la generación de interfaces de supervisión y control .
- El uso de la plataforma como servicio PaaS Heroku permitió el despliegue de la aplicación Django en un servidor en la nube, generando un enlace público para el acceso a la interfaz web/ móvil remota para supervisión del sistema. Por su parte, el uso de la plataforma de Google FireBase y sus servicios Cloud Storage para almacenamiento de imágenes y RealTime como base de datos, dieron al sistema la capacidad de almacenamiento de información y sincronización de los datos en las interfaces de supervisión y control.
- El prototipo mecánico y electrónico diseñado e implementado simula el proceso de una banda transportadora con dos actuadores como compuertas, empleadas para la clasificación de los objetos en dos grupos previamente determinados. Permitió validar de manera correcta el funcionamiento del servicio en la nube y del servicio local, ambos implementados en un microordenador Raspberry Pi 4B.
- Las pruebas de carga determinaron que para la interfaz remota, cuando el número de usuarios es mayor a 1000 las peticiones realizadas comienzan a fallar con un 1%, para 4000 el porcentaje de fallo se eleva a 6%, sin embargo cumple, el servidor con las necesidades del proyecto.
- Las pruebas de usabilidad, culminaron con un puntaje de 90.5/100, por lo que, se puede concluir que la interfaz de la página web local y remota son amigables para el usuario, el texto informativo en la página de ayuda de la interfaz local permitió también la facilidad del uso del sistema, además cabe recalcar que se pudo manejar el sistema independientemente de los conocimientos técnicos de cada uno.

Conclusiones y Trabajos Futuros

Trabajos Futuros

- Llevar a un procesador en la nube la tarea de la generación de modelos de clasificación que en este proyecto se lo realiza de manera local, pues existe límites en el data set que puede ser ingresado así como las épocas empleadas para su entrenamiento, pues esto limita al sistema al reconocimiento de objetos que poseen muchas características similares pero son de diferentes clases, se plantea el uso de técnicas como Web Scraping para acceder a la plataforma de Google Colab y el uso de máquinas virtuales para la generación del modelo y su descarga en el microordenador, consiguiendo que sea solo necesario cargarlo y usarlo en el sistema.
- La interfaz local de supervisión y control es ejecutada dentro de un servidor local generado por Django, se plantea poder llevarlo a un servidor en la nube y tener acceso al control de la máquina desde un lugar remoto, además de la generación de niveles de accesibilidad de usuarios, controlando las actividades permitidas dentro del sistema.
- Finalmente se plantea poder ampliar el sistema incrementando el número de clases para cada modelo de clasificación, para lo cual será necesario la modificación de ambos servicios y de la estructura mecánica.

GRACIAS POR SU ATENCIÓN.