



“SISTEMA PARA EL MONITOREO, DETECCIÓN Y ALERTA DE  
SOMNOLENCIA DEL CONDUCTOR MEDIANTE VISIÓN ARTIFICIAL,  
COMUNICACIÓN INALÁMBRICA Y GEOLOCALIZACIÓN”

---

## Primer Reporte Parcial

---

### Lista de actividades

- Definir rutas del frontend
- Diseño de rutas del backend
- Conexión Backend con Mongo DB
- Sistema de acceso con credenciales
- Creación de la base de datos no relacional
- Diseño de una red neuronal convolucional capaz de detectar ojos cerrados y abiertos

#### *Autores:*

Alan Eduardo Gamboa Del  
Ángel  
Maite Paulette Díaz Martínez

#### *Asesores:*

M.en C. Niels Henrik Navarrete  
Manzanilla  
Dr. Rodolfo Vera Amaro

05 de Octubre 2023

# Índice

<b>1. Definir rutas del frontend</b>	<b>2</b>
1.1. Objetivo . . . . .	2
1.2. Descripción . . . . .	2
1.3. Resultados . . . . .	5
<b>2. Definir rutas del backend</b>	<b>6</b>
2.1. Objetivo . . . . .	6
2.2. Descripción . . . . .	6
2.3. Resultados . . . . .	8
<b>3. Creación de la base de datos No Relacional</b>	<b>10</b>
3.1. Objetivo . . . . .	11
3.2. Descripción . . . . .	11
3.3. Resultados . . . . .	11
<b>4. Conexión Backend con la base de datos</b>	<b>12</b>
4.1. Objetivo . . . . .	12
4.2. Descripción . . . . .	12
4.3. Resultados . . . . .	12
<b>5. Sistema de acceso con credenciales</b>	<b>14</b>
5.1. Objetivo . . . . .	14
5.2. Descripción . . . . .	14
5.3. Resultados . . . . .	16
<b>6. Diseño de una red neuronal convolucional capaz de detectar ojos cerrados y abiertos</b>	<b>18</b>
6.1. Objetivo . . . . .	18
6.2. Descripción . . . . .	18
6.3. Resultados . . . . .	22
<b>7. Bibliografía</b>	<b>24</b>

# 1. Definir rutas del frontend

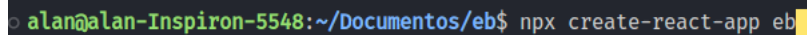
## 1.1. Objetivo

Definir e implementar las rutas que tendrá la aplicación, así como si serán públicas o privadas y la información que se desplegará en cada una de las mismas.

## 1.2. Descripción

### Rutas públicas vs rutas protegidas

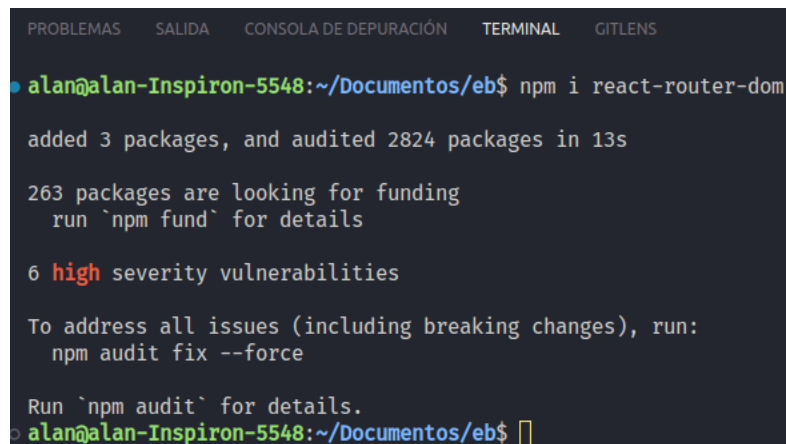
Cuando se habla de una ruta protegida en React, se refiere a programar un bloqueo en ciertas rutas a la cual se le restringe el acceso al usuario. Esto comunmente se realiza para la validación de inicio de sesión de usuarios. Sí el usuario no tiene una sesión iniciada, no podrá acceder a las rutas protegidas de la aplicación. Por otro lado, las rutas públicas son todas aquellas las cuales no requieren contar con una sesión iniciada, y pueden ser accesadas por cualquier tipo de usuario[?]. Como primer paso, se necesita crear un proyecto de React utilizando el siguiente comando:



```
alan@alan-Inspiron-5548:~/Documentos/eb$ npx create-react-app eb
```

Figura 1: Creación proyecto de react

Posteriormente, se realiza la instalación del modelo *React Router Dom* utilizando el siguiente comando:



```
PROBLEMAS  SALIDA  CONSOLA DE DEPURACIÓN  TERMINAL  GITLENS

alan@alan-Inspiron-5548:~/Documentos/eb$ npm i react-router-dom

added 3 packages, and audited 2824 packages in 13s

263 packages are looking for funding
  run `npm fund` for details

6 high severity vulnerabilities

To address all issues (including breaking changes), run:
  npm audit fix --force

Run `npm audit` for details.
alan@alan-Inspiron-5548:~/Documentos/eb$
```

Figura 2: Instalación React Router DOM

Utilizando la librería *useAuthenticator* ofrecida por el paquete de React Dom, crearemos un archivo de nombre *RequireAuth.js* el cuál contendrá una función la cual se encargará de validar si existe una sesión iniciada previamente.

```
src > RequireAuth.js > RequireAuth
1  import { useLocation, Navigate } from 'react-router-dom';
2  import { useAuthenticator } from '@aws-amplify/ui-react';
3
4  export function RequireAuth({ children }) {
5    const location = useLocation();
6    const { route } = useAuthenticator((context) => [context.route]);
7    if (route !== 'authenticated') {
8      return <Navigate to="/login" state={{ from: location }} replace />;
9    }
10   return children;
11 }
```

Figura 3: Función RequireAuth

Posteriormente, se necesita contar con una página de Login, la cuál permitirá validar que se encuentre una sesión iniciada por parte del usuario, para así poder acceder a las rutas protegidas.

```
src > Login.js > ...
1  import { useEffect } from "react";
2  import { Authenticator, useAuthenticator, View } from '@aws-amplify/ui-react';
3  import '@aws-amplify/ui-react/styles.css';
4  import { useNavigate, useLocation } from 'react-router';
5
6  export function Login() {
7    const { route } = useAuthenticator((context) => [context.route]);
8    const location = useLocation();
9    const navigate = useNavigate();
10   let from = location.state?.from?.pathname // '/';
11   useEffect(() => {
12     if (route === 'authenticated') {
13       navigate(from, { replace: true });
14     }
15   }, [route, navigate, from]);
16   return (
17     <View className="auth-wrapper">
18       <Authenticator></Authenticator>
19     </View>
20   );
21 }
```

Figura 4: Componente Login

Para indicar a React, que se desea implementar una ruta protegida, se necesita ir al componente de dicha ruta e ingresar el siguiente código:

```

src > Home.js > Home
1  import { useAuthenticator, Authenticator } from "@aws-amplify/ui-react";
2
3  export default function Home() {
4    const { route } = useAuthenticator((context) => [context.route]);
5    const message =
6      route === 'authenticated' ? 'FIRST PROTECTED ROUTE!' : 'Loading ...';
7    return (
8      <Authenticator>
9        ({ { signOut, user } }) => (
10          <main>
11            <h1>Bienvenido a Home</h1>
12
13            <button onClick={signOut}>Sign out</button>
14          </main>
15        )
16      </Authenticator>
17    );
18
19  }
20

```

Figura 5: Ruta protegida del componente Home

En dicho componente, se hace uso de las librerías *useAuthenticator* y *Authenticator* las cuales son ofrecidas por los servicios de Amazon Amplify. Se tendrá que hacer esto para todos los componentes que deseemos mantener como rutas protegidas.

Finalmente, dentro de nuestro componente **App.js**, crearemos una función que contendrá el directorio de rutas tanto públicas como protegidas:

```

src > App.js > MyRoutes
37
38  export function MyRoutes(){
39    return (
40      <BrowserRouter>
41        <Routes>
42          <Route path="/" element={<Layout />} />
43          <Route index element={<Home />} />
44          <Route
45            path="/conductor"
46            element={
47              <RequireAuth>
48                <Conductor />
49              </RequireAuth>
50            }
51          />
52          <Route
53            path="/incidencia"
54            element={
55              <RequireAuth>
56                <Incidencia />
57              </RequireAuth>
58            }
59          />
60
61          <Route
62            path="/ubicacion"
63            element={

```

Figura 6: Directorio de rutas

Las rutas protegidas, estarán dentro de las etiquetas `<RequireAuth>`/`</RequireAuth>`, mientras que las públicas, irán dentro de las etiquetas `<Route>`/`</Route>`.

### 1.3. Resultados

Como resultado de todo lo anterior, se obtuvo una página de Login que utilizando los servicios de AWS Amplify y Cognito, permitirá iniciar sesión así como registrar a nuevos usuarios.

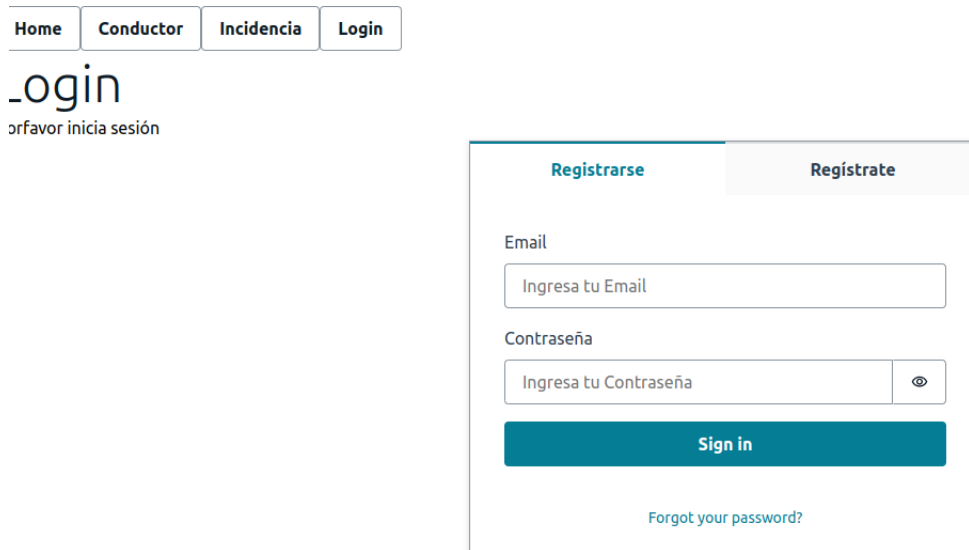


Figura 7: Página de Login

Si intentamos ingresar a las paginas de Conductores, Incidencias o Ubicacion, nos redigirá a la página de Login, debido a que estas páginas fueron definidas como rutas protegidas. Por lo tanto el usuario debe haber iniciado sesión para poder acceder a las mismas.

- **Path:** /  
**Descripción:** En esta dirección se encontrará la el formulario para poder iniciar sesión o registrarse
- **Path:** /home  
**Descripción:** Esta dirección será la página principal de la aplicación dónde se mostrarán las incidencias más recientes así como una lista de todos los conductores
- **Path:** /conductor/  
**Descripción:** Esta dirección mostrará el perfil del conductor de id correspondiente
- **Path:** /detalle\_incidencia  
**Descripción:** Esta dirección mostrará cada incidencia mostrando detalles como hora, fecha, coordenadas
- **Path:** /conductor/id/ubicacion  
**Descripción:** En esta vista se mostrará la ubicación en tiempo real de cada conductor
- **Path:** /conductor/id/incidencias  
**Descripción:** En esta vista se mostrará todas las incidencias registradas por cada conductor


## 2. Definir rutas del backend

### 2.1. Objetivo

Crear las rutas mediante las que el cliente realizará las peticiones y tendrá acceso a las operaciones, así como su funcionamiento en cuanto a obtención de datos y comunicación con el resto de la aplicación.

### 2.2. Descripción

Para poder utilizar los servicios de Amazon Amplify, necesitamos dirigirnos al directorio root de nuestro proyecto y ejecutar el siguiente comando:

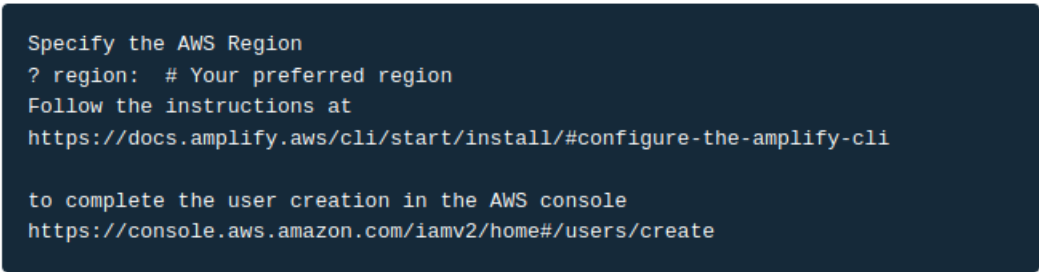


```
npm install -g @aws-amplify/cli
```

[copy](#)

Figura 8: Instalación Amplify CLI

Posteriormente, se necesita especificar la región en la cual queremos alojar nuestra aplicación web:




```
Specify the AWS Region
? region: # Your preferred region
Follow the instructions at
https://docs.amplify.aws/cli/start/install/#configure-the-amplify-cli

to complete the user creation in the AWS console
https://console.aws.amazon.com/iamv2/home#/users/create
```

Figura 9: Configuración del proyecto

Utilizando un editor de código, se necesita especificar que estará utilizando el *SDK* de Amazon Amplify:



```
const AWS = require('aws-sdk')
const awsServerlessExpressMiddleware = require('aws-serverless-express/middleware')
const bodyParser = require('body-parser')
const express = require('express')

AWS.config.update({ region: process.env.TABLE_REGION });
```

Figura 10: Implementación del SDK de Amplify

Posteriormente, declaramos una aplicación de ExpressJs la cuál nos permitirá ejecutar entre otras cosas, peticiones HTTP para la comunicación con la base de datos.

```
// declaracion de una app de express
const app = express()
app.use(bodyParser.json())
app.use(awsServerlessExpressMiddleware.eventContext())
```

Figura 11: Creación de aplicación de Express

Finalmente, se necesitan definir las rutas de las APIs que se estarán utilizando en el proyecto, las cuales serán dos, la primera se encargará de la aplicación web, y la segunda de realizar la comunicación con el Módulo Central de Procesamiento.

Para agregar una api, se necesita ejecutar el siguiente comando desde el directorio raíz del proyecto.

```
alan@alan-Inspiron-5548:~/Documentos/eb$ amplify add api
```

Figura 12: Amplify add api

La consola de AWS Amplify requerirá introducir parámetros con los cuales serán construida nuestra API, para el presente proyecto se estará utilizando una arquitectura mediante GraphQL, por lo cual seleccionaremos dicha opción.

```
alan@alan-Inspiron-5548:~/Documentos/eb$ amplify add api
? Select from one of the below mentioned services: GraphQL
? Here is the GraphQL API that we will create. Select a setting to edit or continue Continue
? Choose a schema template: One-to-many relationship (e.g., "Blogs" with "Posts" and "Comments")

△ WARNING: your GraphQL API currently allows public create, read, update, and delete access to all models via a
n API Key. To configure PRODUCTION-READY authorization rules, review: https://docs.amplify.aws/cli/graphql/autho
rization-rules

✓ GraphQL schema compiled successfully.

Edit your schema at /home/alan/Documentos/eb/amplify/backend/api/ebase/schema.graphql or place .graphql files in
a directory at /home/alan/Documentos/eb/amplify/backend/api/ebase/schema
✓ Do you want to edit the schema now? (Y/n) - yes
```

Figura 13: Configuración de parámetros de GraphQL

Antes de poder publicar nuestra API en AWS Amplify, se necesita definir el Schema con el cuál se hará el manejo de datos. A contiunación se muestran dichos schemas:



```
amplify > backend > api > ebase > ✨ schema.graphql
1  # This "input" configures a global authorization rule to enable
2  # all models in this schema. Learn more about authorization rule
3  input AMPLIFY { globalAuthRule: AuthRule = { allow: public } } #
4
5  type Conductor @model {
6    id: ID
7    nombre: String
8    apellido: String
9    incidencias: [Incidencia] @hasMany
10   num_incidencias: Int
11 }
12
13
14 type Incidencia @model {
15   id: ID
16   title: String
17   estado: Boolean
18   conductor: Conductor @belongsTo
19   detalles: [Detalles] @hasOne
20   fecha_hora: Date
21 }
22
23 type Detalles @model {
24   id: ID
25   incidencia: Incidencia @belongsTo
26   ubicacion: String
27   url_video: String
28 }
29
```

Figura 14: Definición de Schemas para el manejo de datos

Finalmente, ejecutaremos el comando *amplify push*, el cuál publicará la API hacáa AWS amplify, generando el endpoint correspondiente a dicha API

## 2.3. Resultados

Como resultado, tenemos el endpoint de nuestro modelo de GraphQL y nuestra API Key generada por Amplify

```
PROBLEMAS  SALIDA  CONSOLA DE DEPURACIÓN  TERMINAL  GITLENS

Deployment completed.
Deploying root stack ebase [ ===== ] 1/3
  amplify-ebase-dev-175126    AWS::CloudFormation::Stack  UPDATE_IN_PROGRESS
  apiebase                   AWS::CloudFormation::Stack  CREATE_IN_PROGRESS
  authebase35f92a6b          AWS::CloudFormation::Stack  UPDATE_COMPLETE
Deployed api ebase [ ===== ] 9/9
  GraphQLAPI                 AWS::AppSync::GraphQLApi    CREATE_COMPLETE
  GraphQLAPINONEDS95A13CF0    AWS::AppSync::DataSource    CREATE_COMPLETE
  GraphQLAPIDefaultApiKey215A6D... AWS::AppSync::ApiKey        CREATE_COMPLETE
  GraphQLAPITransformerSchema3C... AWS::AppSync::GraphQLSchema  CREATE_COMPLETE
  Blog                       AWS::CloudFormation::Stack  CREATE_COMPLETE
  Comment                    AWS::CloudFormation::Stack  CREATE_COMPLETE
  Post                       AWS::CloudFormation::Stack  CREATE_COMPLETE
  ConnectionStack            AWS::CloudFormation::Stack  CREATE_COMPLETE
  CustomResourcesjson         AWS::CloudFormation::Stack  CREATE_COMPLETE

✓ Generated GraphQL operations successfully and saved at src/graphql
Deployment state saved successfully.

GraphQL endpoint: [REDACTED].appsync-api.us-east-1.amazonaws.com/graphql
GraphQL API KEY: ([REDACTED])jm

GraphQL transformer version: 2

alan@alan-Inspiron-5548:~/Documentos/eb$
```

Figura 15: Generación de Endpoint de GraphQL

### 3. Creación de la base de datos No Relacional

Debido a problemas de integración junto con los servicios de autenticación y de despliegue de Amplify, se decidió utilizar el sistema de gestión de bases de datos DynamoDB.

DynamoDB es un servicio de base de datos NoSQL Ofrecido por Amazon Web Services. DynamoDB trabaja con tablas. Estas a su vez, contienen parámetros importantes que se mencionarán a continuación.

- **Primary Key:** Se trata de una clave primaria simple, compuesta por un solo atributo denominado clave de partición. Una clave primaria puede ser una clave de partición o una combinación de clave de partición y clave de ordenación. La clave primaria debe ser única en toda la tabla.
- **Partition Key:** Es la llave principal por la cual se agruparán los datos, y determina cómo se particiona la información.
- **Sort Key:** Es llave de ordenamiento de los datos.

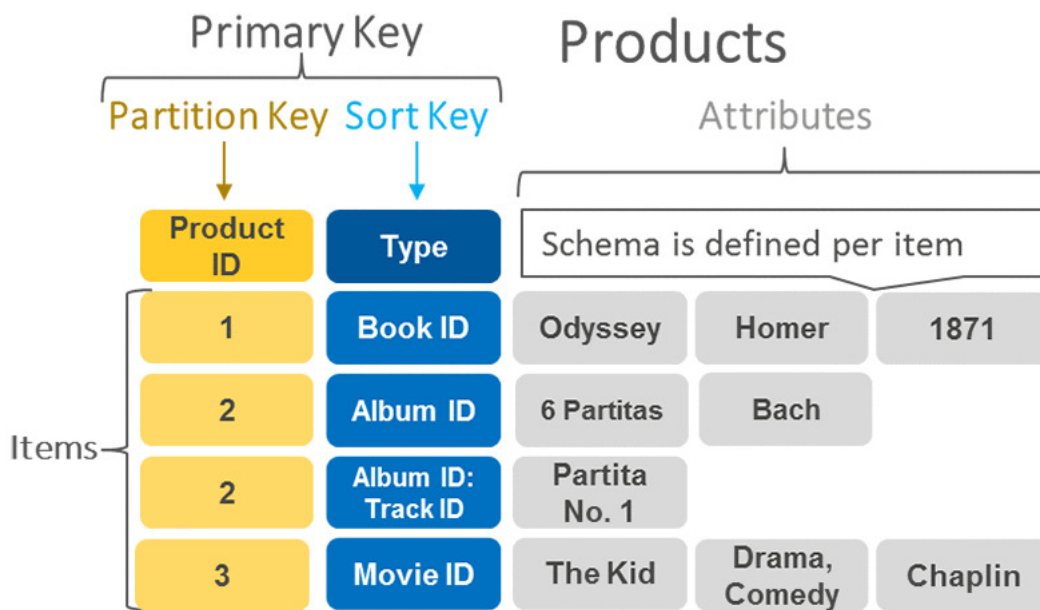


Figura 16: Tablas en DynamoDB

DynamoDB almacena los datos como grupos de atributos, conocidos como elementos. Los elementos son similares a las filas o registros de otros sistemas de bases de datos. DynamoDB almacena y recupera cada elemento en función del valor de la clave principal, que debe ser único.

DynamoDB utiliza el valor de la clave de partición como parámetro de entrada para una función hash interna. El resultado de la función hash determina la partición en la que se almacena el elemento. La ubicación de cada elemento viene determinada por el valor hash de su clave de partición.

Todos los elementos con la misma clave de partición se almacenan juntos y, para las claves de partición compuestas, se ordenan por el valor de la clave de ordenación. DynamoDB divide las particiones por clave de ordenación si el tamaño de la colección crece más de 10 GB[?].

### 3.1. Objetivo

Crear la base de datos en MongoDB.

### 3.2. Descripción

Para crear nuestras tablas de DynamoDB, se debe ingresar a la consola de AWS.



Figura 17: Consola de AWS

### 3.3. Resultados

Una vez dentro, ingresamos a la sección del servicio de DynamoDB

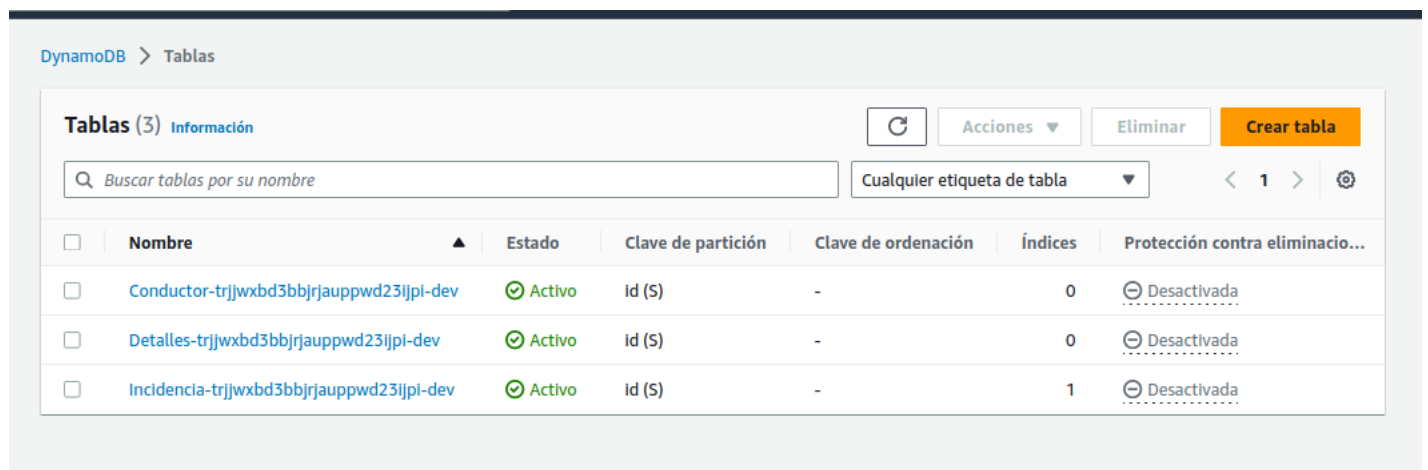


Figura 18: Tablas generadas mediante los schemas definidos

Como se puede observar, gracias a los pasos realizados en la sección 2, DynamoDB crea automáticamente las tablas creadas en base a los schemas definidos previamente.

## 4. Conexión Backend con la base de datos

### 4.1. Objetivo

Realizar la conexión de NodeJs con la base de datos MongoDB.

### 4.2. Descripción

Para realizar la conexión y la integración de los servicios de DynamoDB hacia nuestra API, se hará uso de AppSync. Las API de GraphQL creadas con AWS AppSync brindan a los desarrolladores frontend la capacidad de consultar varias bases de datos, microservicios y API desde un único punto de conexión de GraphQL.

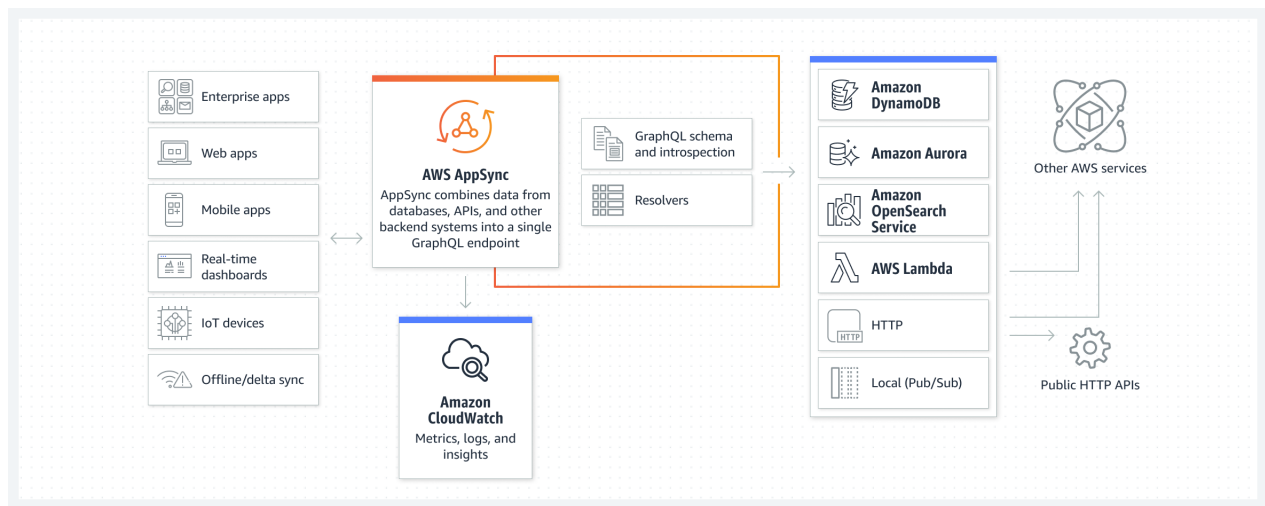


Figura 19: Funcionamiento de Appsync

AWS AppSync crea las API sin servidor de GraphQL y de publicación o suscripción que simplifican el desarrollo de aplicaciones a través de un único punto de conexión para consultar, actualizar o publicar datos.

### 4.3. Resultados

Después de haber realizado los pasos de la sección 2, si ejecutamos el comando `amplify status`, la consola de Amplify nos retornará el endpoint de GraphQL junto con AppSync correspondiente, el cual se utilizará para realizar todas las operaciones con respecto al almacenamiento de datos.

```
• alan@alan-Inspiron-5548:~/Documentos/eb$ amplify status

Current Environment: dev



| Category | Resource name | Operation | Provider plugin   |
|----------|---------------|-----------|-------------------|
| Auth     | ebase35f92a6b | No Change | awscloudformation |
| Api      | ebase         | No Change | awscloudformation |



GraphQL endpoint: https://ckqxuivcobc4rgh72skzudaixy.appsync-a
GraphQL transformer version: 2

○ alan@alan-Inspiron-5548:~/Documentos/eb$ █
```

Figura 20: Generación de endpoint de GraphQL

## 5. Sistema de acceso con credenciales

### 5.1. Objetivo

Establecer los roles de cada tipo de usuario con sus respectivos permisos de acceso a la aplicación web utilizando Amazon Cognito

### 5.2. Descripción

Amazon Cognito funciona utilizando *pools* de usuarios. Un pool de usuarios es un directorio almacenado en los servicios de Amazon[?]. Los beneficios que ofrece estar registrado en un pool de usuarios de Amazon Cognito son los siguientes:

- Servicio de registro e inicio de sesión
- Gestión del directorio de usuarios
- Servicios de seguridad tales como verificación de dos pasos
- Acceso a servicios de la suite de AWS tales como S3 o Dynamodb

#### Funcionamiento

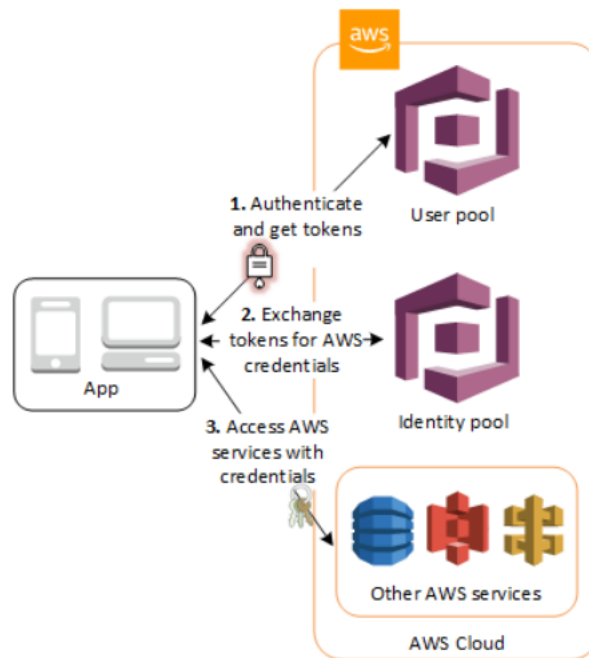


Figura 21: Funcionamiento de Amazon Cognito

- Como primer paso el usuario inicia sesión a través de un grupo de usuarios y recibe tokens del grupo de usuarios después de una autenticación exitosa.
- Posteriormente la aplicación intercambia los tokens del grupo de usuarios por las credenciales de AWS a través de un grupo de identidades.

- Finalmente, el usuario puede usar esas credenciales de AWS para acceder a otros servicios de AWS, como Amazon S3 o DynamoDB.

## Implementación

Para poder hacer uso de Amazon Cognito en nuestra aplicación debemos de introducir el siguiente comando:

```
amplify add auth
```

[copy](#)

Figura 22: Implementación de Amazon Cognito en el proyecto

Obteniendo el siguiente menú:

```
alan@alan-Inspiron-5548:~/Documentos/eb$ amplify update auth
Please note that certain attributes may not be overwritten if you choose to use defaults settings.

You have configured resources that might depend on this Cognito resource. Updating this Cognito resource could
have unintended side effects.

Using service: Cognito, provided by: awscloudformation
What do you want to do? Walkthrough all the auth configurations
Select the authentication/authorization services that you want to use: (Use arrow keys)
> User Sign-Up, Sign-In, connected with AWS IAM controls (Enables per-user Storage features for images or other
content, Analytics, and more)
  User Sign-Up & Sign-In only (Best used with a cloud API only)
  I want to learn more.
```

Figura 23: Menu de configuración de Amazon Cognito

Dejamos seleccionado la primera opción, la cual nos permitirá utilizar los servicios de autenticación ofrecidos por Amazon Cognito, además de otros servicios de la suite de AWS.

Finalmente utilizamos el comando *amplify push* para desplegar los cambios a nuestra aplicación.



### 5.3. Resultados



Figura 24: Implementación completada

Entrando a nuestra consola de AWS, en la sección de Amazon Cognito, se puede observar dos grupos de usuarios, uno de tipo Administrador, el cual puede realizar cambios a la configuración de la aplicación, y otro de tipo de Usuario, el cual sólo puede hacer uso del sistema de inicio de sesión y registro ofrecido por Cognito.

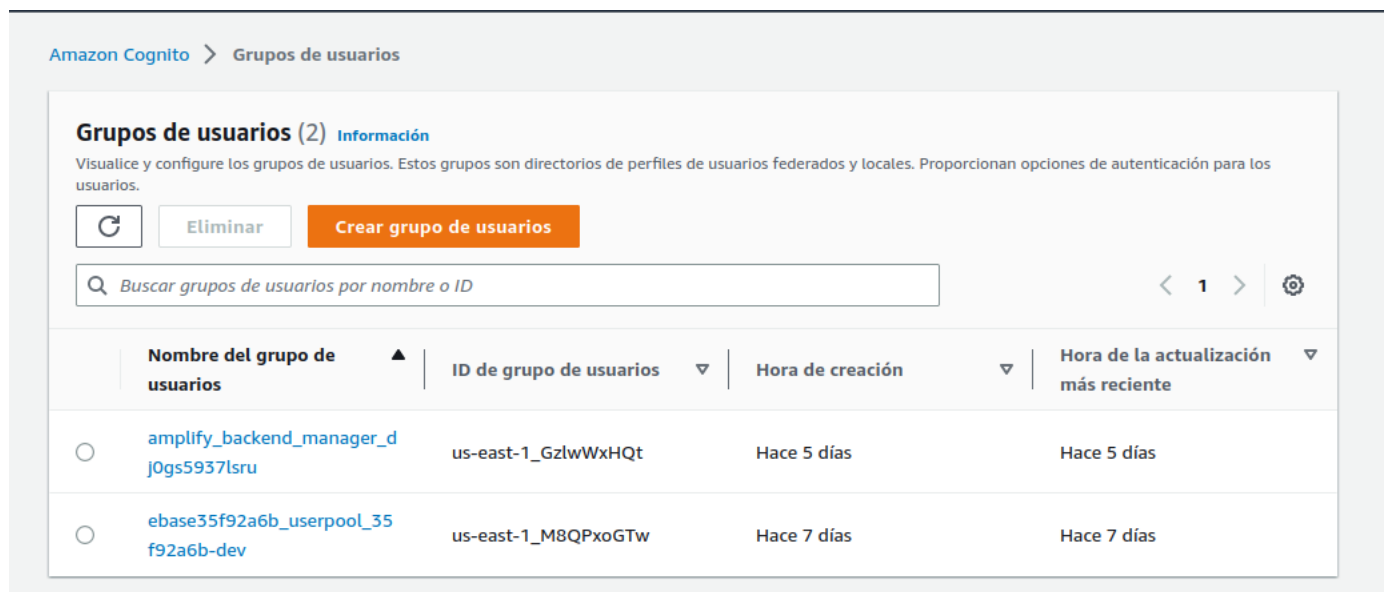


Figura 25: Grupos de usuario

Información general sobre el grupo de usuarios

Nombre del grupo de usuarios  
amplify\_backend\_manager\_dj0gs5937lsru

ID de grupo de usuarios  
us-east-1\_GzlwWxHQ

ARN  
arn:aws:cognito-idp:us-east-1:387678068467:userpool/us-east-1\_GzlwWxHQ

Número estimado de usuarios  
1

Hora de creación  
2 de marzo de 2023, 13:46 GMT-6

Hora de la actualización más reciente  
2 de marzo de 2023, 13:46 GMT-6

Introducción

Usuarios

Grupos

Experiencia de inicio de sesión

Experiencia de inscripción

Mensajería

Integración de aplicaciones

Propiedades del grupo de usuarios

Usuarios (1) Información

Vea, edite y cree usuarios en el grupo de usuarios. Los usuarios habilitados y confirmados podrán iniciar sesión en el grupo de usuarios.

Nombre de usuario

Buscar usuarios por atributo

< 1 >

Nombre de usuario

Dirección de correo...

Correo electrónico ...

Estado de la confirmación

Estado

☐

aws-amplify-admin

-

No

Confirmado

Habilitado

Figura 26: Grupos de usuario

<

Usuarios

Grupos

Experiencia de inicio de sesión

Experiencia de inscripción

Mensajería

Integración de aplicaciones

Propied...

>

Usuarios (5) Información

Vea, edite y cree usuarios en el grupo de usuarios. Los usuarios habilitados y confirmados podrán iniciar sesión en el grupo de usuarios.

Eliminar usuario

Crear usuario

Nombre de usuario

Buscar usuarios por atributo

< 1 >

Nombre de usuario

Dirección de correo elec...

Correo electrónico verifi...

Estado de la confirmación

Estado

☐

19bd77dc-441b-4d3e-8...

maite.diazm98@gmail.com

Sí

Confirmado

Habilitado

☐

473bad4c-b959-45e4-8...

alangam97@gmail.com

No

No confirmado

Habilitado

☐

849ca9f3-6fd3-4750-a8...

mayte\_dm@hotmail.com

No

No confirmado

Habilitado

☐

d1df2a48-8374-4fd2-be...

mayte\_dm@hotmail.es

No

No confirmado

Habilitado

☐

e6bf3d4f-1807-4325-ba...

alangam97@gmail.com

Sí

Confirmado

Habilitado

Figura 27: Grupos de usuario

Proyecto Terminal 2

17

## 6. Diseño de una red neuronal convolucional capaz de detectar ojos cerrados y abiertos

### 6.1. Objetivo

Diseñar y realizar pruebas de los modelos de redes neuronales convolucionales previamente investigados para determinar el rendimiento y la precisión de cada uno.

### 6.2. Descripción

Para realizar el entrenamiento de los modelos EfficientNet y MobileNet se hizo uso de google colab el cual nos permite ejecutar código de Python en el navegador.

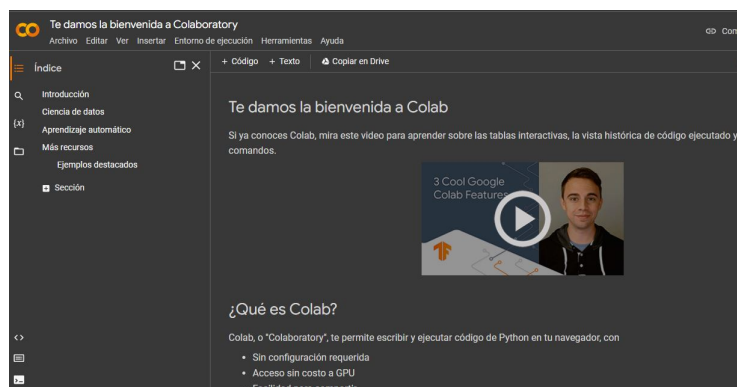


Figura 28: Entorno de trabajo Google Colab.

Se requerirá de un dataset con imágenes de ojos abiertos y cerrados para entrenar los modelos, el dataset que se utilizó es el siguiente: MRL Eye Dataset[?].

Parte de los datos del data set original se separaron en dos carpetas, en la primera carpeta se colocaron las imágenes con ojos cerrados y en la segunda carpeta las imágenes de ojos abiertos. Posteriormente se subieron las carpetas a un repositorio en drive para trabajar desde Google Colab. El dataset con el cual se trabajó contiene 3242 imágenes de las cuales 1840 son imágenes de ojos cerrados y 1402 son de con ojos abiertos.

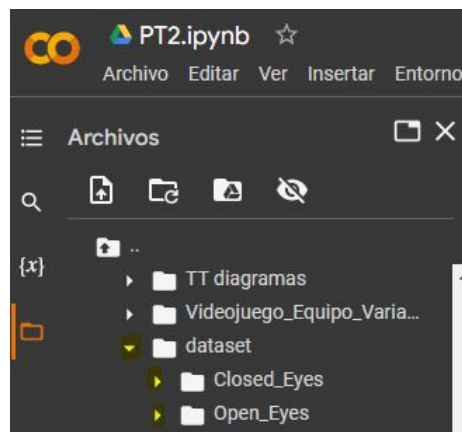
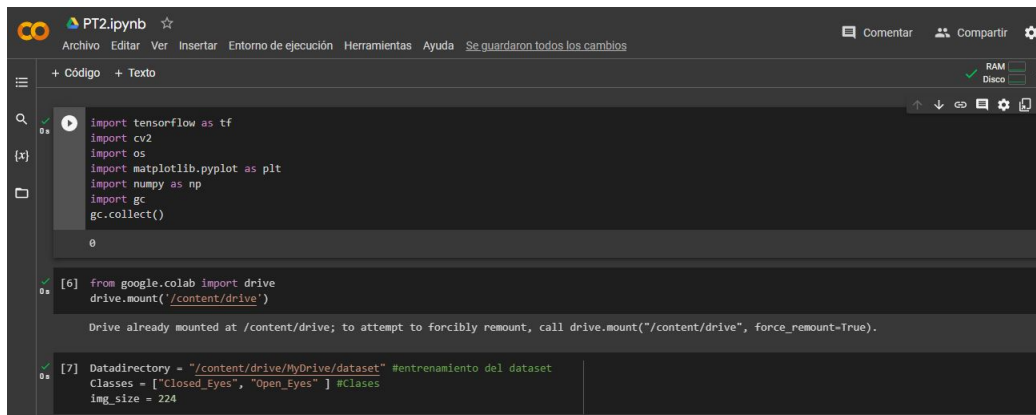


Figura 29: Dataset a trabajar desde Google Colab.

Se importaron las librerías a utilizar y el repositorio de drive para hacer uso del dataset, así mismo se definieron 2 clases, la primera para ojos abiertos y la segunda para ojos cerrados. Ya que los modelos pre entrenados con los que se trabajó son en base a imágenes con dimensión de 244 x 244, se definió el tamaño de la imagen en 224.



```

import tensorflow as tf
import cv2
import os
import matplotlib.pyplot as plt
import numpy as np
import gc
gc.collect()

0

[6] from google.colab import drive
drive.mount('/content/drive')

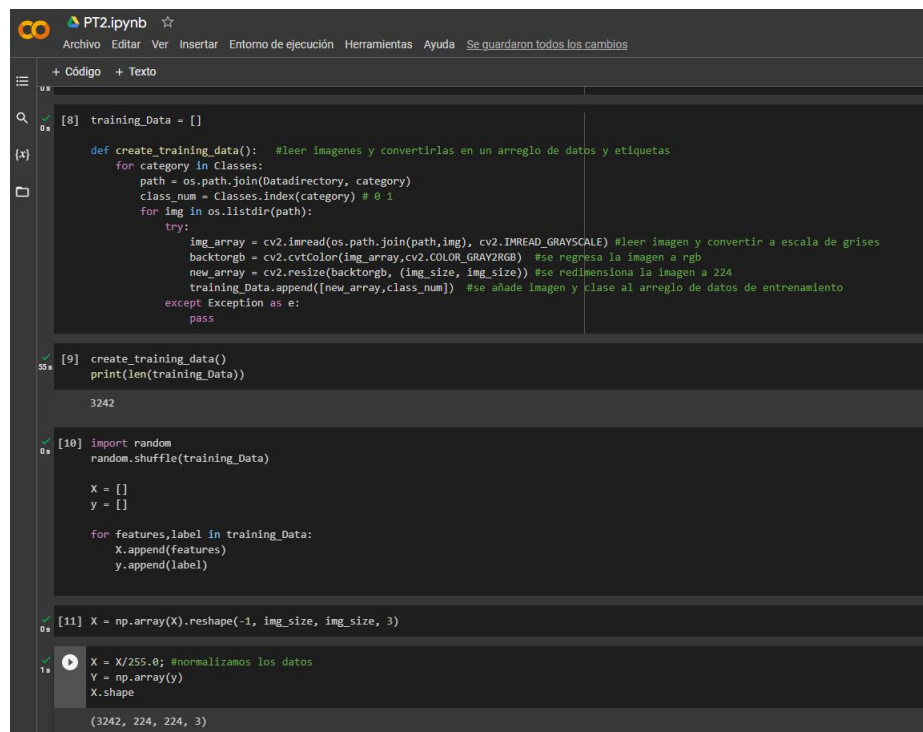
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).

[7] Datadirectory = "/content/drive/MyDrive/dataset" #entrenamiento del dataset
Classes = ["Closed_Eyes", "Open_Eyes"] #Clases
img_size = 224

```

Figura 30: Librerías utilizadas y repositorio en drive del dataset.

Es necesario convertir las imágenes en un arreglo de datos con su respectiva clase por lo que se creó una función con dicho propósito, posteriormente se etiquetaron los datos y se normalizaron para obtener las entradas de datos, donde X contiene los píxeles de las imágenes en tamaño 224x224 en 3 planos (rgb) y Y contiene el arreglo de etiquetas 0 y 1, que corresponden a la clase a la que pertenece cada imagen.



```

[8] training_Data = []

def create_training_data(): #leer imagenes y convertirlas en un arreglo de datos y etiquetas
    for category in Classes:
        path = os.path.join(Datadirectory, category)
        class_num = Classes.index(category) # 0 1
        for img in os.listdir(path):
            try:
                img_array = cv2.imread(os.path.join(path,img), cv2.IMREAD_GRAYSCALE) #leer imagen y convertir a escala de grises
                backtorgb = cv2.cvtColor(img_array,cv2.COLOR_GRAY2RGB) #se regresa la imagen a rgb
                new_array = cv2.resize(backtorgb, (img_size, img_size)) #se redimensiona la imagen a 224
                training_Data.append([new_array,class_num]) #se añade imagen y clase al arreglo de datos de entrenamiento
            except Exception as e:
                pass

[9] create_training_data()
print(len(training_Data))

3242

[10] import random
random.shuffle(training_Data)

X = []
y = []

for features,label in training_Data:
    X.append(features)
    y.append(label)

[11] X = np.array(X).reshape(-1, img_size, img_size, 3)

X = X/255.0; #normalizamos los datos
Y = np.array(y)
X.shape

(3242, 224, 224, 3)

```

Figura 31: Preparación de los datos de entrada

Posteriormente se exportó el modelo y se visualizó la arquitectura de la red.

```

- Entrenamiento utilizando EfficientNetB0

import tensorflow as tf
from tensorflow import keras

from tensorflow.keras import layers
from tensorflow.keras.applications import EfficientNetB0

model = tf.keras.applications.efficientnet.EfficientNetB0()
model.summary()

```

block6d_drop (Dropout)	(None, 7, 7, 192)	0	['block6d_project_bn[0][0]']
block6d_add (Add)	(None, 7, 7, 192)	0	['block6d_drop[0][0]', 'block6c_add[0][0]']
block7a_expand_conv (Conv2D)	(None, 7, 7, 1152)	221184	['block6d_add[0][0]']
block7a_expand_bn (BatchNormalization)	(None, 7, 7, 1152)	4608	['block7a_expand_conv[0][0]']
block7a_expand_activation (Activation)	(None, 7, 7, 1152)	0	['block7a_expand_bn[0][0]']
block7a_dwconv (DepthwiseConv2D)	(None, 7, 7, 1152)	10368	['block7a_expand_activation[0][0]']
block7a_bn (BatchNormalization)	(None, 7, 7, 1152)	4608	['block7a_dwconv[0][0]']
block7a_activation (Activation)	(None, 7, 7, 1152)	0	['block7a_bn[0][0]']
block7a_se_squeeze (GlobalAveragePooling2D)	(None, 1152)	0	['block7a_activation[0][0]']
block7a_se_reshape (Reshape)	(None, 1, 1, 1152)	0	['block7a_se_squeeze[0][0]']
block7a_se_reduce (Conv2D)	(None, 1, 1, 48)	55344	['block7a_se_reshape[0][0]']
block7a_se_expand (Conv2D)	(None, 1, 1, 1152)	56448	['block7a_se_reduce[0][0]']
block7a_se_excite (Multiply)	(None, 7, 7, 1152)	0	['block7a_activation[0][0]', 'block7a_se_expand[0][0]']
block7a_project_conv (Conv2D)	(None, 7, 7, 320)	368640	['block7a_se_excite[0][0]']
block7a_project_bn (BatchNormalization)	(None, 7, 7, 320)	1280	['block7a_project_conv[0][0]']

Figura 32: Exportaci3n del modelo MobileNet

Se aplicaron modificaciones a partir de la cuarta 3ltima capa, se agregaron nuevas capas y la funci3n de activaci3n sigmoid.

```

base_input = model.layers[0].input #Capa de entrada
base_output = model.layers[-2].output

Flat_layer = layers.Flatten()(base_output)
final_output = layers.Dense(1)(Flat_layer)
final_output = layers.Activation('sigmoid')(final_output)

new_model = keras.Model(inputs = base_input, outputs= final_output)
new_model.summary()

```

block7a_expand_conv (Conv2D)	(None, 7, 7, 1152)	221184	['block6c_add[0][0]']
block7a_expand_bn (BatchNormalization)	(None, 7, 7, 1152)	4608	['block6d_add[0][0]']
block7a_expand_activation (Activation)	(None, 7, 7, 1152)	0	['block7a_expand_conv[0][0]']
block7a_dwconv (DepthwiseConv2D)	(None, 7, 7, 1152)	10368	['block7a_expand_activation[0][0]']
block7a_bn (BatchNormalization)	(None, 7, 7, 1152)	4608	['block7a_dwconv[0][0]']
block7a_activation (Activation)	(None, 7, 7, 1152)	0	['block7a_bn[0][0]']
block7a_se_squeeze (GlobalAveragePooling2D)	(None, 1152)	0	['block7a_activation[0][0]']
block7a_se_reshape (Reshape)	(None, 1, 1, 1152)	0	['block7a_se_squeeze[0][0]']
block7a_se_reduce (Conv2D)	(None, 1, 1, 48)	55344	['block7a_se_reshape[0][0]']
block7a_se_expand (Conv2D)	(None, 1, 1, 1152)	56448	['block7a_se_reduce[0][0]']
block7a_se_excite (Multiply)	(None, 7, 7, 1152)	0	['block7a_activation[0][0]', 'block7a_se_expand[0][0]']
block7a_project_conv (Conv2D)	(None, 7, 7, 320)	368640	['block7a_se_excite[0][0]']
block7a_project_bn (BatchNormalization)	(None, 7, 7, 320)	1280	['block7a_project_conv[0][0]']
top_conv (Conv2D)	(None, 7, 7, 1280)	409600	['block7a_project_bn[0][0]']
top_bn (BatchNormalization)	(None, 7, 7, 1280)	5120	['top_conv[0][0]']
top_activation (Activation)	(None, 7, 7, 1280)	0	['top_bn[0][0]']

Figura 33: Modificaci3n del modelo Mobilnet

Finalmente se entrena el modelo MobileNet

```
[14] #ajustes para la clasificación binaria (abierto cerrado)
#model.compile(loss = "binary_crossentropy", optimizer="adam", metrics=["accuracy"])
#hist = model.fit(X,Y, epochs = 1 ,validation_split=0.2) #entrenamiento

new_model.compile(loss = "binary_crossentropy", optimizer="adam", metrics=["accuracy"])
hist = new_model.fit(X,Y, epochs = 5 ,validation_split=0.2) #entrenamiento transfer learning

Epoch 1/5
82/82 [=====] - 682s 8s/step - loss: 0.1598 - accuracy: 0.9487 - val_loss: 2.7340 - val_accuracy: 0.5932
Epoch 2/5
82/82 [=====] - 661s 8s/step - loss: 0.0598 - accuracy: 0.9753 - val_loss: 1.2245 - val_accuracy: 0.7596
Epoch 3/5
82/82 [=====] - 662s 8s/step - loss: 0.0592 - accuracy: 0.9734 - val_loss: 0.0797 - val_accuracy: 0.9661
Epoch 4/5
82/82 [=====] - 659s 8s/step - loss: 0.0220 - accuracy: 0.9907 - val_loss: 0.0564 - val_accuracy: 0.9753
Epoch 5/5
82/82 [=====] - 658s 8s/step - loss: 0.0376 - accuracy: 0.9838 - val_loss: 0.2774 - val_accuracy: 0.9091
```

Figura 34: Entrenamiento del modelo MobileNet

Para el modelo EfficientNet se realiz? el mismo procedimiento que en el modelo EfficientNet.

```
[18] #entrenamiento
new_model.compile(loss = "binary_crossentropy", optimizer="adam", metrics=["accuracy"])
hist = new_model.fit(X,Y, epochs = 5 ,validation_split=0.2) #entrenamiento transfer learning

Epoch 1/5
82/82 [=====] - 76s 320ms/step - loss: 0.1273 - accuracy: 0.9476 - val_loss: 3.7610 - val_accuracy: 0.5609
Epoch 2/5
82/82 [=====] - 22s 265ms/step - loss: 0.0741 - accuracy: 0.9707 - val_loss: 1.4369 - val_accuracy: 0.5609
Epoch 3/5
82/82 [=====] - 22s 266ms/step - loss: 0.0668 - accuracy: 0.9784 - val_loss: 2.2189 - val_accuracy: 0.5609
Epoch 4/5
82/82 [=====] - 22s 268ms/step - loss: 0.0519 - accuracy: 0.9769 - val_loss: 1.3085 - val_accuracy: 0.5609
Epoch 5/5
82/82 [=====] - 22s 270ms/step - loss: 0.0462 - accuracy: 0.9853 - val_loss: 2.4268 - val_accuracy: 0.5609
```

Figura 35: Entrenamiento del modelo EfficientNet

La gr?fica muestra el las epocas y la precisi?n obtenida en cada una de ellas, as? como la precisi?n de los datos ocupados para la validaci?n dentro del entrenamiento.

MobileNet

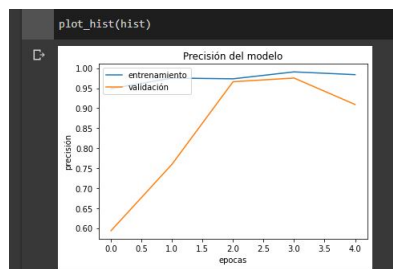


Figura 36: Gr?fica de precisi?n en cada epoca del modelo MobileNet.

EfficientNet

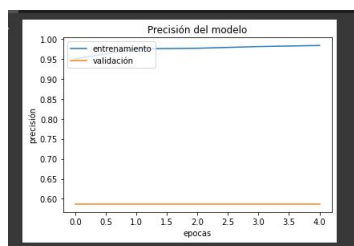


Figura 37: Gr?fica de precisi?n en cada epoca del modelo EfficientNet.

### 6.3. Resultados

Una vez entrenados los modelos se realizaron las pruebas de los modelos EfficientNet y MobileNet para ojos cerrado y abiertos. Se obtuvieron los siguientes resultados:

MobileNet

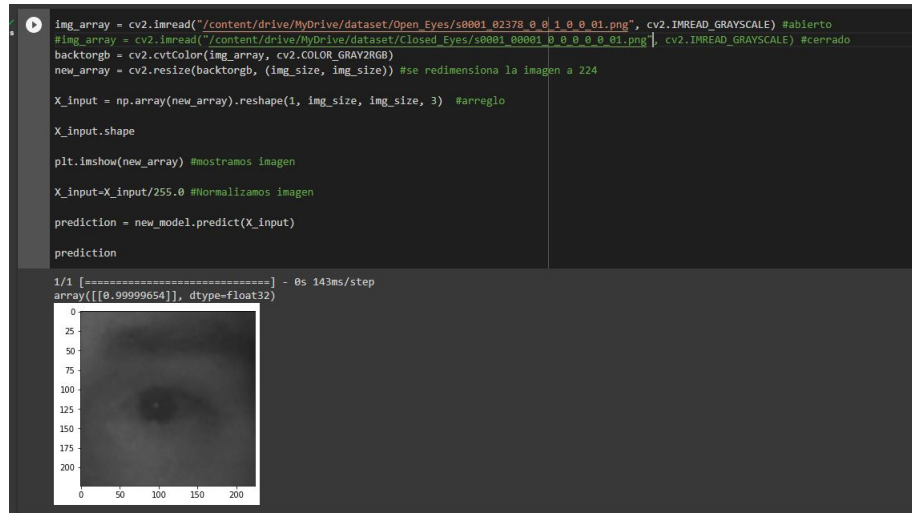


Figura 38: Predicción 1 para del modelo MobileNet.

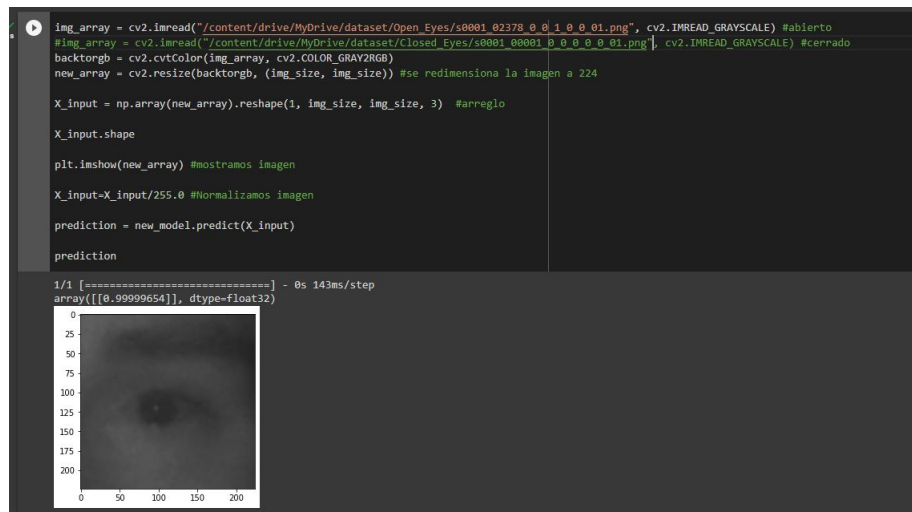


Figura 39: Predicción 2 para del modelo MobileNet.

EfficientNet

```
#img_array = cv2.imread("/content/drive/MyDrive/dataset/Open_Eyes/s0001_02378_0_0_1_0_0_01.png", cv2.IMREAD_GRAYSCALE) #abierto
img_array = cv2.imread("/content/drive/MyDrive/dataset/Closed_Eyes/s0001_00001_0_0_0_0_0_01.png", cv2.IMREAD_GRAYSCALE) #cerrado
backtorgb = cv2.cvtColor(img_array, cv2.COLOR_GRAY2RGB)
new_array = cv2.resize(backtorgb, (img_size, img_size)) #se redimensiona la imagen a 224

X_input = np.array(new_array).reshape(1, img_size, img_size, 3) #arreglo
X_input.shape

plt.imshow(new_array) #mostramos imagen

X_input=X_input/255.0 #Normalizamos imagen

prediction = new_model.predict(X_input)

prediction

1/1 [=====] - 0s 105ms/step
array([[0.25493553]], dtype=float32)
```

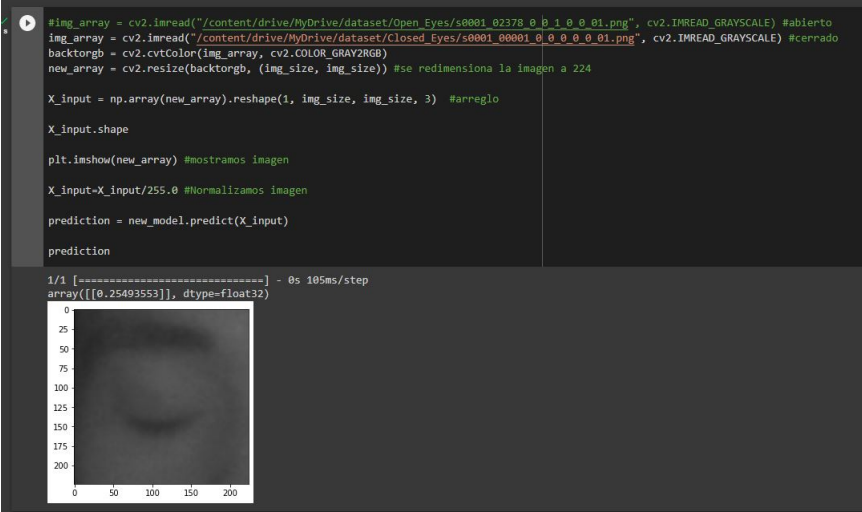


Figura 40: Predicci?n 1 para del modelo EfficientNet.

```
img_array = cv2.imread("/content/drive/MyDrive/dataset/Open_Eyes/s0001_02378_0_0_1_0_0_01.png", cv2.IMREAD_GRAYSCALE) #abierto
#img_array = cv2.imread("/content/drive/MyDrive/dataset/Closed_Eyes/s0001_00001_0_0_0_0_0_01.png", cv2.IMREAD_GRAYSCALE) #cerrado
backtorgb = cv2.cvtColor(img_array, cv2.COLOR_GRAY2RGB)
new_array = cv2.resize(backtorgb, (img_size, img_size)) #se redimensiona la imagen a 224

X_input = np.array(new_array).reshape(1, img_size, img_size, 3) #arreglo
X_input.shape

plt.imshow(new_array) #mostramos imagen

X_input=X_input/255.0 #Normalizamos imagen

prediction = new_model.predict(X_input)

prediction

1/1 [=====] - 2s 25s/step
array([[0.25561202]], dtype=float32)
```

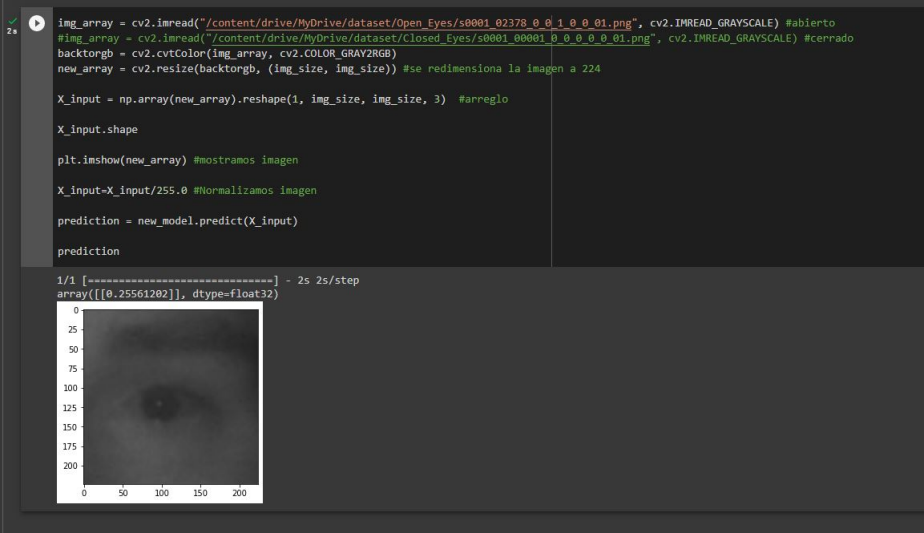


Figura 41: Predicci?n 2 para del modelo EfficientNet.



## 7. Bibliografía

### Referencias

- [1] React Dev Team, *React*, React. <https://react.dev/> (accedido el 1 de abril de 2023).
- [2] Waveshare Electronics, *SIM7600G-H 4G for Jetson Nano - Waveshare Wiki*, Waveshare Electronics [https://www.waveshare.com/wiki/SIM7600G-H\\_4G\\_for\\_Jetson\\_Nano\\_4G\\_connecting](https://www.waveshare.com/wiki/SIM7600G-H_4G_for_Jetson_Nano_4G_connecting) (accedido el 16 de abril de 2023).
- [3] Facebook Dev Team, *Introduction to GraphQL — GraphQL — A query language for your API* <https://graphql.org/learn/> (accedido el 4 de abril de 2023).
- [4] NVIDIA, "Get Started with the Jetson Nano Developer Kit", NVIDIA Developer, 2019. [Online]. Disponible: <https://developer.nvidia.com/embedded/learn/get-started-jetson-nano-devkit#intro>. [Accedido: Abril 02 2023].
- [5] Nvidia Developer, "Get Started with Jetson Nano Devkit," Nvidia Developer. [En línea]. Disponible: <https://developer.nvidia.com/embedded/learn/get-started-jetson-nano-devkit#write>. [Accedido: 2 de abril de 2023].
- [6] Dusty, N. "Building the Repo - NVIDIA Jetson Inference," GitHub. [Online]. Disponible en: <https://github.com/dusty-nv/jetson-inference/blob/master/docs/building-repo-2.md>. [Accedido en: 02-abr-2023].