



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

---

## Segundo Reporte Parcial

---

### Lista de actividades

- Maquetación web
- Investigación de la documentación del módulo 3G/4G LTE-Base Hat
- Enlace de Amazon S3 con el sistema backend
- Creación de los servicios backend
- Familiarizarse con el entorno de desarrollo de la NVIDIA Jetson Nano
- Implementar algoritmo para la detección del rostro y ojos
- Implementar puntos faciales en el rostro y métrica MOR

#### *Autores:*

Alan Eduardo Gamboa Del  
Ángel  
Maite Paulette Díaz Martínez

#### *Asesores:*

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

21 de Abril 2023

# Índice

<b>1. Maquetación web</b>	<b>4</b>
1.1. Objetivo . . . . .	4
1.2. Descripción . . . . .	4
1.3. Resultados . . . . .	4
<b>2. Investigación de la documentación del módulo 3G/4G LTE-Base Hat</b>	<b>7</b>
2.1. Objetivo . . . . .	7
2.2. Descripción . . . . .	7
2.3. Resultados . . . . .	8
<b>3. Enlace de Amazon S3 con el sistema backend</b>	<b>10</b>
3.1. Objetivo . . . . .	10
3.2. Descripción . . . . .	10
3.3. Resultados . . . . .	12
<b>4. Creación de los servicios backend</b>	<b>13</b>
4.1. Objetivo . . . . .	13
4.2. Descripción . . . . .	13
4.3. Resultados . . . . .	19
<b>5. Familiarizarse con el entorno de desarrollo de la NVIDIA Jetson Nano</b>	<b>20</b>
5.1. Objetivo . . . . .	20
5.2. Descripción . . . . .	20
5.3. Resultados . . . . .	21
<b>6. Implementar algoritmo para la detección del rostro y ojos</b>	<b>22</b>
6.1. Objetivo . . . . .	22
6.2. Descripción . . . . .	22
6.3. Resultados . . . . .	22
<b>7. Implementar puntos faciales en el rostro y la metrica MOR</b>	<b>23</b>
7.1. Objetivo . . . . .	23
7.2. Descripción . . . . .	23
7.3. Resultados . . . . .	23
<b>8. Conclusiones</b>	<b>24</b>
<b>9. Bibliografía</b>	<b>25</b>

## Índice de figuras

1.	Página Principal - Layout.jsx . . . . .	5
2.	Vista Reporte Incidencia Incidencia - Incidencia.jsx . . . . .	5
3.	Vista Ubicacion . . . . .	6
4.	Vista Conductores - Conductores.jsx . . . . .	6
5.	Módulo Base-Hat SIM7600G-H . . . . .	7
6.	Instalación de librerías . . . . .	7
7.	Instalación de librerías . . . . .	8
8.	Minicom . . . . .	8
9.	Generación de Endpoint de GraphQL . . . . .	9
10.	Configuración de servicio de almacenamiento S3 . . . . .	10
11.	Generación de Endpoint de GraphQL . . . . .	10
12.	Generación de Endpoint de GraphQL . . . . .	10
13.	Generación de Endpoint de GraphQL . . . . .	11
14.	Generación de Endpoint de GraphQL . . . . .	11
15.	Generación de Endpoint de GraphQL . . . . .	11
16.	Generación de Endpoint de GraphQL . . . . .	12
17.	Tablas generadas mediante los schemas definidos . . . . .	12
18.	Función getConductor . . . . .	13
19.	Función listConductors . . . . .	14
20.	Función getIncidencia . . . . .	14
21.	Función listIncidencias . . . . .	15
22.	Función createConductor . . . . .	15
23.	Función updateConductor . . . . .	16
24.	Función deleteConductor . . . . .	16
25.	Función createIncidencia . . . . .	17
26.	Función oncreateIncidencia . . . . .	17
27.	Consola de AWS . . . . .	18
28.	Funcionamiento de Appsync . . . . .	18
29.	Crear Conductor . . . . .	19
30.	Resultado . . . . .	19
31.	Funcionamiento de Amazon Cognito . . . . .	21

## Índice de tablas

# 1. Maquetación web

## 1.1. Objetivo

Crear la aplicación web e implementar el sistema de diseño de manera local.

## 1.2. Descripción

### ReactJs

Para el desarrollo del front-end del presente proyecto, se hará uso de la librería de diseño de ReactJs. ReactJs facilita la creación de componentes reutilizables e interactivos para las interfaces de usuario.

Los componentes que darán lugar a las vistas del presente proyecto son los siguientes:

- `Layout.jsx`: Este componente será la vista principal de la Aplicación Web. Se trata de un diseño tipo *dashboard* que contendrá una sección principal que contendrá etiquetas para poder ingresar a las diferentes vistas de la aplicación. Además estará compuesta también de una sección secundaria que mostrará el contenido de dichas vistas.
- `Incidencias.jsx`: Este componente se encargará de mostrar todas las incidencias registradas en la base de datos. Las incidencias serán desplegadas en forma de lista.
- `Incidencias.jsx`: Este componente de mostrar un reporte de incidencia a detalle. Contendrá una ventana que permitirá ver el video del momento de la incidencia registrada. Así como los datos de la fecha y hora. Además de botones para poder confirmar o rechazar la incidencia. Finalmente contendrá el nombre del conductor además de una opción para poder consultar la ubicación en tiempo real del conductor.
- `Ubicación.jsx`: Este componente mostrará la ubicación en tiempo real del conductor con ayuda del servicio de diseño de mapas Leaflet.
- `Conductores.jsx`: Este componente mostrará todos los conductores registrados en la base de datos en forma de lista.

## 1.3. Resultados

De acuerdo con los componentes explicados anteriormente, las vistas que contendrá la aplicación web son las siguientes:

- Página Principal

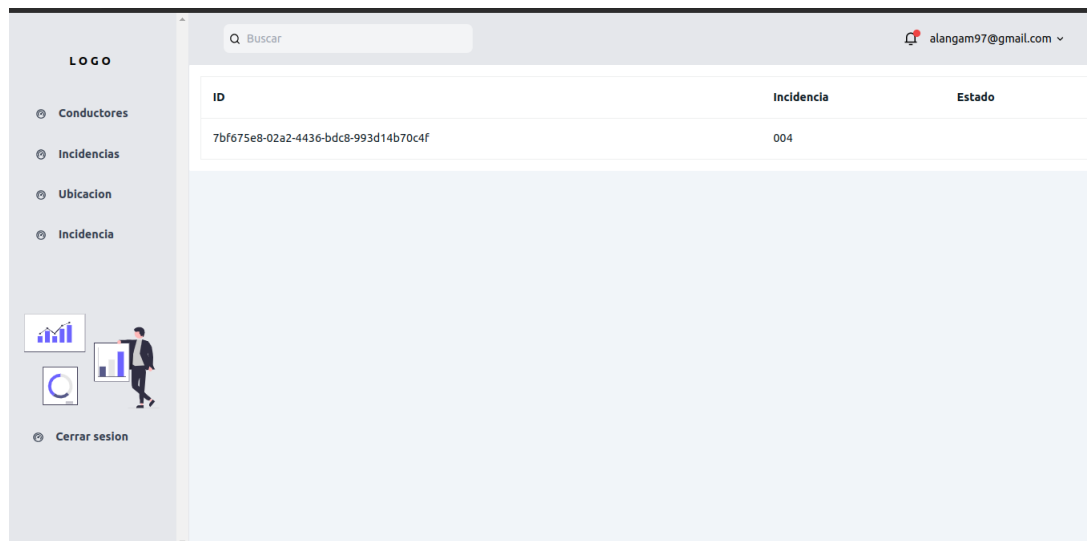


Figura 1: Página Principal - Layout.jsx

- Reporte de Incidencia

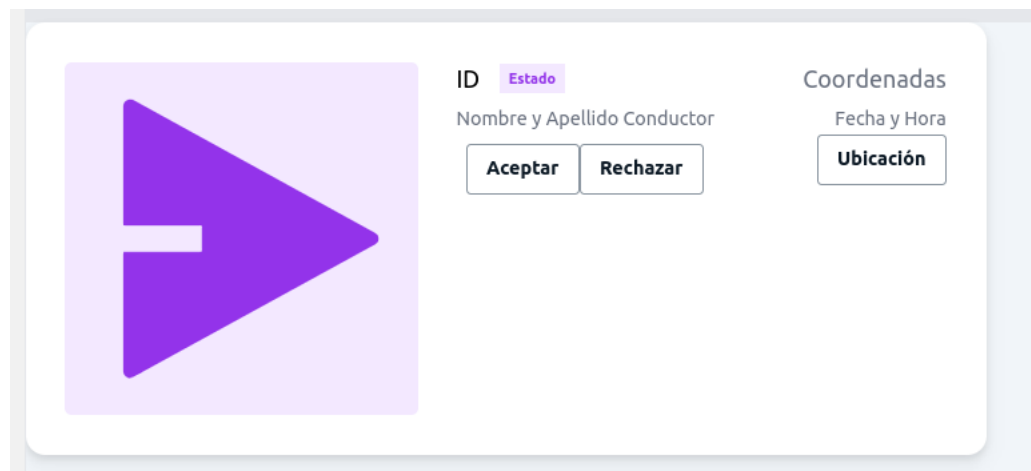


Figura 2: Vista Reporte Incidencia Incidencia - Incidencia.jsx

- Ubicación

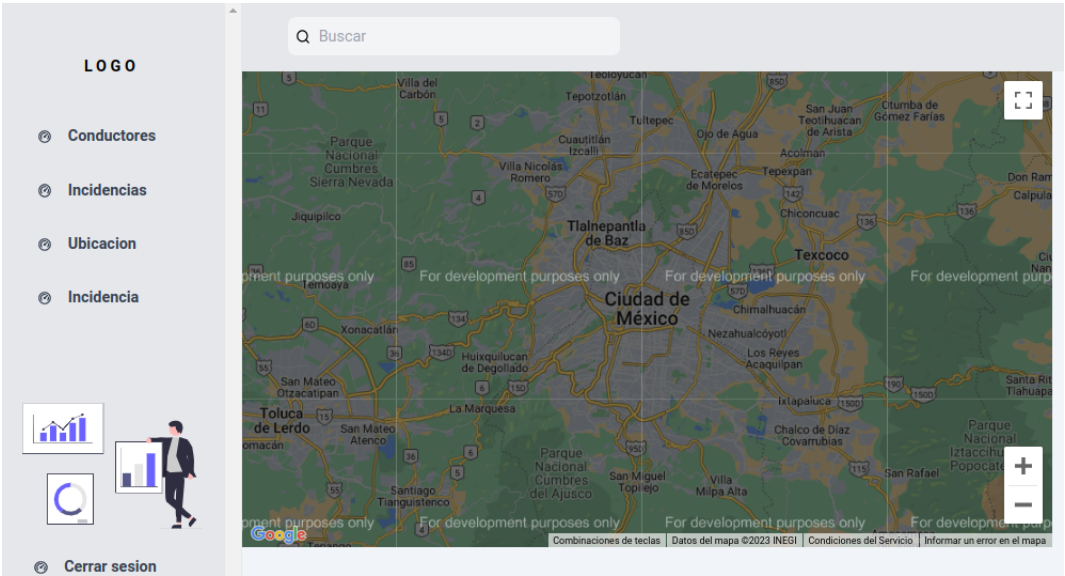


Figura 3: Vista Ubicacion

■ Conductores

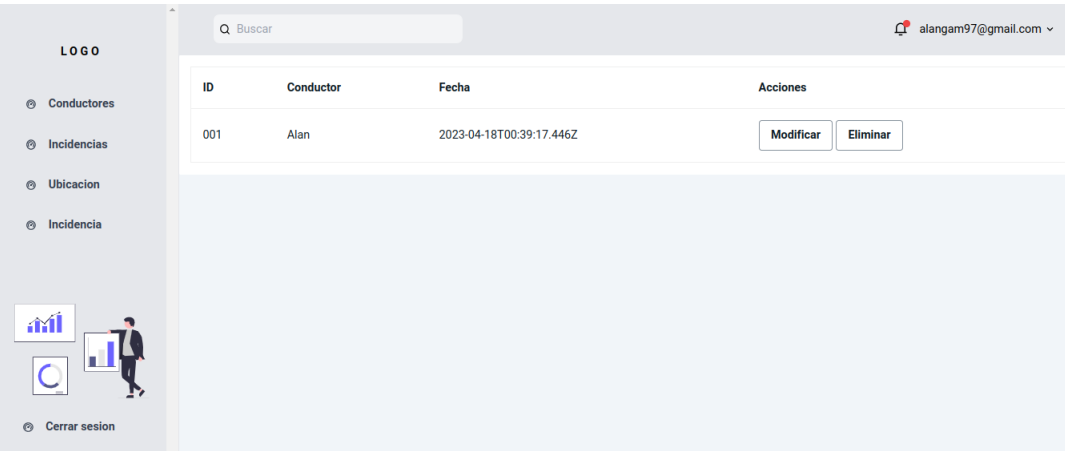


Figura 4: Vista Conductores - Conductores.jsx

## 2. Investigación de la documentación del módulo 3G/4G LTE-Base Hat

### 2.1. Objetivo

Familiarizarse con las distintas funciones y comandos, así como el entorno de desarrollo que ofrece el dispositivo Base-Hat

### 2.2. Descripción

Para el presente proyecto, se hará uso del Base-Hat SIM7600G-H 4G para Jetson Nano.



Figura 5: Módulo Base-Hat SIM7600G-H

En primer lugar, utilizando la terminal del sistema operativo Ubuntu, se ingresan los siguientes comandos:

```
sudo apt-get update
sudo apt-get install python3-pip
sudo pip3 install pyserial
mkdir -p ~/Documents/SIM7600X_4G_for_JETSON_NANO
wget -P ~/Documents/SIM7600X_4G_for_JETSON_NANO/ https://www.waveshare.com/w/upload/6/64/SIM7600X_4G_for_JETSON_NANO.tar.gz
cd ~/Documents/SIM7600X_4G_for_JETSON_NANO/
tar -xvf SIM7600X_4G_for_JETSON_NANO.tar.gz
sudo pip3 install Jetson.GPIO
sudo groupadd -f -r gpio
sudo usermod -a -G gpio your_user_name
sudo udevadm control --reload-rules && sudo udevadm trigger
sudo apt-get install minicom
```



Figura 6: Instalación de librerías

Los comandos ingresados en la figura 6 se encargan de instalar todas las librerías y el software necesario para poder comenzar a utilizar la red LTE mediante el módulo SIM7600G-H. Además también se crea un directorio que contendrá la configuración de usuario así como un cuenta enlazada al módulo.

Posteriormente, probamos que el puerto GPIO de nuestra Jetson Nano esté funcionando con los siguientes comandos:

```
echo 200 > /sys/class/gpio/export
echo out > /sys/class/gpio200/direction
echo 1 > /sys/class/gpio200/value
echo 0 > /sys/class/gpio200/value
```

Figura 7: Instalación de librerías

Después de haber realizado los pasos anteriores, el pin con el nombre NET deberá parpadear constantemente, lo que significa que el módulo está listo para ser utilizado.

Para realizar la comunicación LTE, primero se ingresa a la librería minicom utilizando los siguientes comandos:

```
sudo su
killall ModemManager
minicom -D /dev/ttyUSB2
```

Figura 8: Minicom

## 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 9: Generación de Endpoint de GraphQL

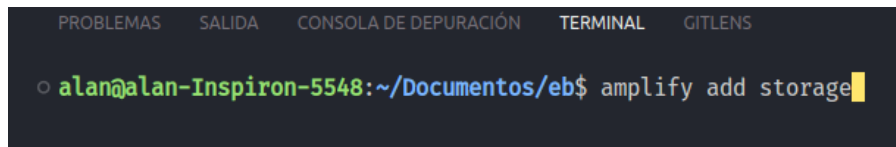
### 3. Enlace de Amazon S3 con el sistema backend

#### 3.1. Objetivo

Configurar e implementar la comunicación entre el sistema de alojamiento Amazon S3 y el sistema backend

#### 3.2. Descripción

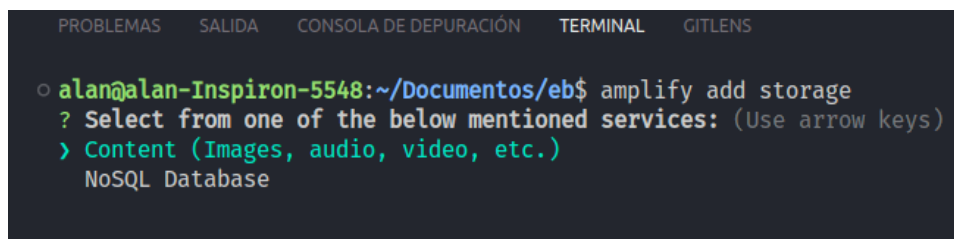
Utilizando un editor de código, y desde el directorio raíz de la aplicación, se deberá introducir el siguiente comando:



```
PROBLEMAS  SALIDA  CONSOLA DE DEPURACIÓN  TERMINAL  GITLENS
alan@alan-Inspiron-5548:~/Documentos/eb$ amplify add storage
```

Figura 10: Configuración de servicio de almacenamiento S3

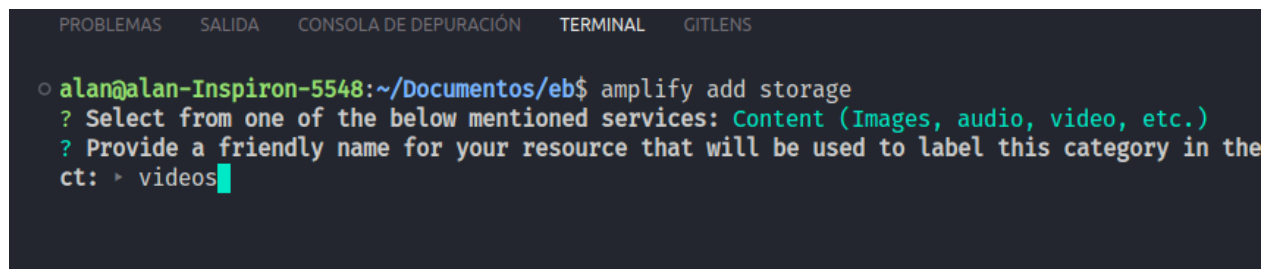
Posteriormente, se requiere especificar que tipo de servicio de almacenamiento se integrará a la aplicación (multimedia o base de datos NoSQL). Para el presente proyecto, se utilizará el almacenamiento de contenido multimedia, por lo tanto, se seleccionará dicha opción.



```
PROBLEMAS  SALIDA  CONSOLA DE DEPURACIÓN  TERMINAL  GITLENS
alan@alan-Inspiron-5548:~/Documentos/eb$ amplify add storage
? Select from one of the below mentioned services: (Use arrow keys)
> Content (Images, audio, video, etc.)
  NoSQL Database
```

Figura 11: Generación de Endpoint de GraphQL

Ingresamos el nombre de nuestro espacio de almacenamiento:



```
PROBLEMAS  SALIDA  CONSOLA DE DEPURACIÓN  TERMINAL  GITLENS
alan@alan-Inspiron-5548:~/Documentos/eb$ amplify add storage
? Select from one of the below mentioned services: Content (Images, audio, video, etc.)
? Provide a friendly name for your resource that will be used to label this category in the ct: > videos
```

Figura 12: Generación de Endpoint de GraphQL

Después, se necesita establecer cuantos usuarios, así como cuales podrán acceder a dicho servicio:

```
o alan@alan-Inspiron-5548:~/Documentos/eb$ amplify add storage
? Select from one of the below mentioned services: Content (Images, audio, video, etc.)
✓ Provide a friendly name for your resource that will be used to label this category in
  ct: · videos

✓ Provide bucket name: · videos
? Who should have access: ... (Use arrow keys or type to filter)
> Auth users only
  Auth and guest users
```

Figura 13: Generación de Endpoint de GraphQL

```
o alan@alan-Inspiron-5548:~/Documentos/eb$ amplify add storage
? Select from one of the below mentioned services: Content (Images, audio, video, etc.)
✓ Provide a friendly name for your resource that will be used to label this category in
  ct: · videos

✓ Provide bucket name: · videos
✓ Who should have access: · Auth and guest users
? What kind of access do you want for Authenticated users? ... (Use arrow keys or type to
  ● create/update
  ● read
  >● delete
  (Use <space> to select, <ctrl + a> to toggle all)
```

Figura 14: Generación de Endpoint de GraphQL

```
? Select from one of the below mentioned services: Content (Images, audio, video, etc.)
✓ Who should have access: · Auth and guest users
✓ What kind of access do you want for Authenticated users? · create/update, read, delete
? What kind of access do you want for Guest users? ... (Use arrow keys or type to filter)
>● create/update
  ● read
  o delete
  (Use <space> to select, <ctrl + a> to toggle all)
```

Figura 15: Generación de Endpoint de GraphQL

```

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
✓ Successfully pulled backend environment dev from the cloud.

Current Environment: dev

```

Category	Resource name	Operation	Provider plugin
Storage	videos	Create	awscloudformation
Auth	ebase35f92a6b	Update	awscloudformation
Function	S3Trigger6956e03e	No Change	awscloudformation
Api	ebase	No Change	awscloudformation

```

? Are you sure you want to continue? (Y/n) >

```

Figura 16: Generación de Endpoint de GraphQL

### 3.3. Resultados

Al ingresar a la consola de servicios de AWS, en la sección de buckets de S3, se puede observar que se encuentra el bucket recién creado llamado *videos175126-dev*.

Los buckets son contenedores de datos almacenados en S3. <a href="#">Más información</a>				
	Copiar ARN	Vaciar	Eliminar	Crear bucket
<input type="text" value="Buscar buckets por nombre"/>				
	Nombre	Región de AWS	Acceso	Fecha de creación
<input type="radio"/>	<a href="#">amplify-ebase-dev-175126-deployment</a>	EE. UU. Este (Norte de Virginia) us-east-1	Los objetos pueden ser públicos	28 Feb 2023 5
<input type="radio"/>	<a href="#">ebase6c6dcb3b214e4c3fa82df5d47dce7c22175126-dev</a>	EE. UU. Este (Norte de Virginia) us-east-1	Los objetos pueden ser públicos	25 Mar 2023 1
<input type="radio"/>	<a href="#">videos175126-dev</a>	EE. UU. Este (Norte de Virginia) us-east-1	Los objetos pueden ser públicos	17 Apr 2023 1

Figura 17: Tablas generadas mediante los schemas definidos

## 4. Creación de los servicios backend

### 4.1. Objetivo

Desarrollar los servicios que constituyen las funciones de la API, tales como obtención, creación y eliminación. Además, crear de los servicios que llevarán a cabo las funciones internas en la plataforma.

### 4.2. Descripción

GraphQL trabaja con 3 tipos de archivos:

- *Queries*: Este archivo contiene las funciones que permitirán acceder a los datos.
- *Mutations*: En este archivo se encuentran todas las funciones que permitirán realizar el manejo de datos (actualizar, eliminar, agregar)
- *Subscriptions*: Las *subscriptions* en GraphQL son funciones de consulta especiales, que se envían a través de un punto de conexión websocket. Permiten realizar cierta operación cada vez que se ejecuta una acción en el backend.

Para comenzar con el archivo de *Queries* se tienen las siguientes funciones:

```
export const getConductor = /* GraphQL */ `
  query GetConductor($id: ID!) {
    getConductor(id: $id) {
      id
      nombre
      apellido
      incidencias {
        items {
          id
          estado
          url_video
          ubicacion
          fecha_hora
          createdAt
          updatedAt
          conductorIncidenciasId
        }
      }
      nextToken
    }
    num_incidencias
    createdAt
    updatedAt
  }
`;
```

Figura 18: Función getConductor

La función de la figura ??, obtiene los datos de un solo Conductor.

```
export const listConductors = /* GraphQL */ `
  query ListConductors(
    $filter: ModelConductorFilterInput
    $limit: Int
    $nextToken: String
  ) {
    listConductors(filter: $filter, limit: $limit, nextToken: $nextToken) {
      items {
        id
        nombre
        apellido
        incidencias {
          nextToken
        }
        num_incidencias
        createdAt
        updatedAt
      }
      nextToken
    }
  }
`;
```

Figura 19: Función listConductors

La función de la figura 19 obtiene todos los datos de todos los conductores almacenados en la base de datos.

```
export const getIncidencia = /* GraphQL */ `
  query GetIncidencia($id: ID!) {
    getIncidencia(id: $id) {
      id
      conductor {
        id
        nombre
        apellido
        incidencias {
          nextToken
        }
        num_incidencias
        createdAt
        updatedAt
      }
      estado
      url_video
      ubicacion
      fecha_hora
      createdAt
      updatedAt
      conductorIncidenciasId
    }
  }
`;
```

Figura 20: Función getIncidencia

La función de la figura 20 obtiene los datos de una sola Incidencia.

```
port const listIncidencias = /* GraphQL */ `
query ListIncidencias(
  $filter: ModelIncidenciaFilterInput
  $limit: Int
  $nextToken: String
) {
  listIncidencias(filter: $filter, limit: $limit, nextToken: $nextToken) {
    items {
      id
      conductor {
        id
        nombre
        apellido
        num_incidencias
        createdAt
        updatedAt
      }
      estado
      url_video
      ubicacion
      fecha_hora
      createdAt
      updatedAt
      conductorIncidenciasId
    }
  }
  nextToken
}
```

Figura 21: Función listIncidencias

La función de la figura 21 obtiene los datos de todas las incidencias de almacenadas en la base de datos.

En cuanto al archivo de *Mutations* se tienen las siguientes funciones:

```
export const createConductor = /* GraphQL */ `
mutation CreateConductor(
  $input: CreateConductorInput!
  $condition: ModelConductorConditionInput
) {
  createConductor(input: $input, condition: $condition) {
    id
    nombre
    apellido
    incidencias {
      items {
        id
        estado
        url_video
        ubicacion
        fecha_hora
        createdAt
        updatedAt
        conductorIncidenciasId
      }
    }
    nextToken
  }
  num_incidencias
  createdAt
  updatedAt
}
```

Figura 22: Función createConductor



La función de la figura 22 se encarga de crear el registro de un conductor en la base de datos.

```
export const updateConductor = /* GraphQL */ `
mutation UpdateConductor(
  $input: UpdateConductorInput!
  $condition: ModelConductorConditionInput
) {
  updateConductor(input: $input, condition: $condition) {
    id
    nombre
    apellido
    incidencias {
      items {
        id
        estado
        url_video
        ubicacion
        fecha_hora
        createdAt
        updatedAt
        conductorIncidenciasId
      }
    }
    nextToken
  }
  num_incidencias
  createdAt
  updatedAt
}
```

Figura 23: Función updateConductor

La función de la figura 23 se encarga de modificar datos del registro de un conductor.

```
export const deleteConductor = /* GraphQL */ `
mutation DeleteConductor(
  $input: DeleteConductorInput!
  $condition: ModelConductorConditionInput
) {
  deleteConductor(input: $input, condition: $condition) {
    id
    nombre
    apellido
    incidencias {
      items {
        id
        estado
        url_video
        ubicacion
        fecha_hora
        createdAt
        updatedAt
        conductorIncidenciasId
      }
    }
    nextToken
  }
  num_incidencias
  createdAt
  updatedAt
}
```

Figura 24: Función deleteConductor

La función de la figura 24 se encarga de eliminar un conductor de la base de datos.

```
;
export const createIncidencia = /* GraphQL */ `
mutation CreateIncidencia(
  $input: CreateIncidenciaInput!
  $condition: ModelIncidenciaConditionInput
) {
  createIncidencia(input: $input, condition: $condition) {
    id
    conductor {
      id
      nombre
      apellido
      incidencias {
        nextToken
      }
      num_incidencias
      createdAt
      updatedAt
    }
    estado
    url_video
    ubicacion
    fecha_hora
    createdAt
    updatedAt
    conductorIncidenciasId
  }
}
`;
```

Figura 25: Función createIncidencia

La función de la figura 25 se encarga de crear una Incidencia en la base de datos. Para el archivo de *subscriptions* se tienen la siguiente función:

```
export const onCreateIncidencia = /* GraphQL */ `
subscription OnCreateIncidencia(
  $filter: ModelSubscriptionIncidenciaFilterInput
) {
  onCreateIncidencia(filter: $filter) {
    id
    conductor {
      id
      nombre
      apellido
      incidencias {
        nextToken
      }
      num_incidencias
      createdAt
      updatedAt
    }
    estado
    url_video
    ubicacion
    fecha_hora
    createdAt
    updatedAt
    conductorIncidenciasId
  }
}
`;
```

Figura 26: Función onCreateIncidencia

La función de la figura 26 se encarga de realizar una consulta cada vez que una Incidencia nueva es dada de alta en la base de datos.

Al estar trabajando con GraphQL dentro del proyecto, la manera en que se podrán realizar las operaciones *CRUD* - (*Create, Read, Update, Delete*), será mediante funciones JSON.

Para comprobar que la API permite dichas operaciones, se debe ingresar a la consola de AWS y dirigirse a la sección de AWS AppSync.



Figura 27: Consola de AWS

Posteriormente se necesita ingresar a la sección de consultas.

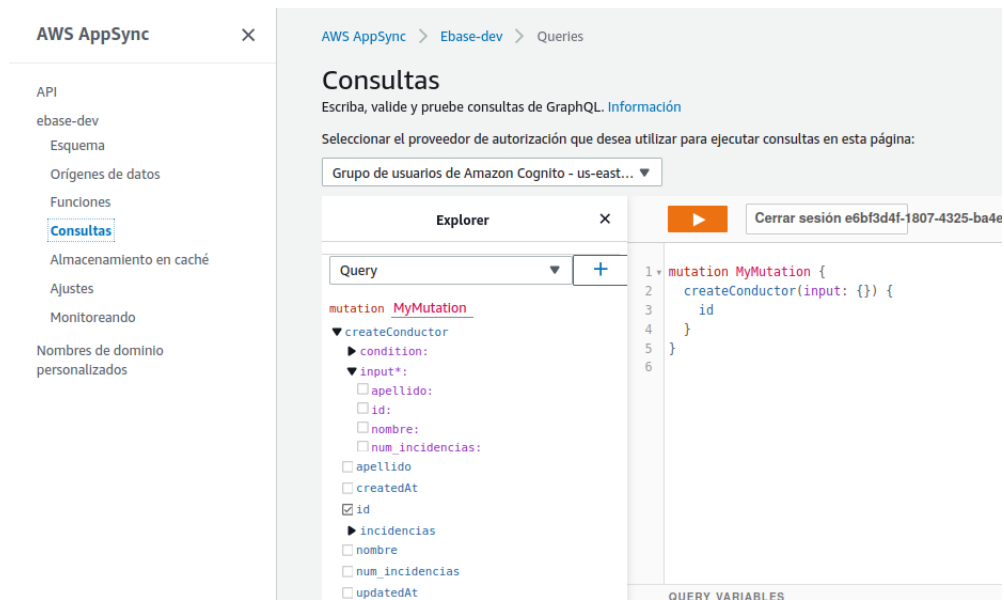


Figura 28: Funcionamiento de Appsync

AWS permite elegir si realizar un *query*, *mutation*, o *subscription* mediante código JSON

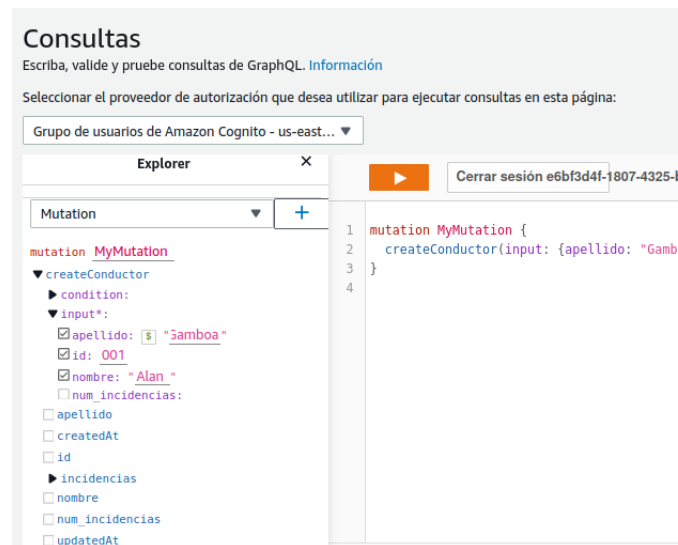


Figura 29: Crear Conductor

En la figura 30 se puede apreciar una sentencia JSON que permite utilizar las funciones previamente creadas para dar de alta un conductor.

### 4.3. Resultados

Como resultado tenemos un log que nos muestra la entrada creada



Figura 30: Resultado

## 5. Familiarizarse con el entorno de desarrollo de la NVIDIA Jetson Nano

### 5.1. Objetivo

Investigar la documentación ofrecida por NVIDIA sobre el uso y entorno de desarrollo de la jetson nano.

### 5.2. Descripción

#### Instalación en la tarjeta microSD

El Jetson Nano Developer Kit utiliza una tarjeta microSD como dispositivo de arranque y almacenamiento principal. Por tanto fue necesario instalar un entorno de desarrollo en la propia placa, para lo cual se requirió de una tarjeta microSD de un mínimo recomendado de 32 GB de acuerdo a la documentación Nvidia. [5].

Como primer paso se descargo el *Jetson Nano Developer Kit SD Card Image* [6] y posteriormente se instalo en la tarjeta microSD desde el sistema operativo Linux, utilizando los siguientes pasos:

1. Se abrio una terminal y se inserto la tarjeta microSD.
2. Se uso el siguiente comando para mostrar que dispositivo de disco se le asignó:

```
dmesg | tail | awk '\$3 == "sd" {print}'
```

3. Se escribio la imagen de la tarjeta SD comprimida (previamente descargada) en la tarjeta microSD con el comando:

```
/usr/bin/unzip -p ~/Downloads/jetson_nano_devkit_sd_card.zip |  
sudo /bin/dd of=/dev/sda bs=1M status=progress
```

4. Finalmente se expulso del dispositivo de disco desde la linea de comando utilizando:

```
sudo eject /dev/sda
```

#### Configuración y primer arranque

Jetson Nano Developer Kit permite dos formas de interactuar, la primera es por medio de otra computadora y la segunda haciendo uso de una pantalla, teclado y mouse conectados. Adicionalmente el kit de desarrollo no cuenta con una fuente de alimentación incluida por lo que se utilizó una fuente de alimentación Micro-USB(5V-2A).

Para iniciar el kit de desarrollo se conectó el mouse, la pantalla, el teclado y la funete de alimentación, posteriormente se realizó la configuración inicial del sistema operativo el cual incluyó lo siguiente:

- Revisar y aceptar el EULA del software NVIDIA Jetson.
- Seleccionar el idioma del sistema, la distribución del teclado y la zona horaria
- Crear nombre de usuario, contraseña y nombre de la computadora.
- Seleccione el tamaño de partición de la aplicación: se recomienda utilizar el tamaño máximo sugerido.

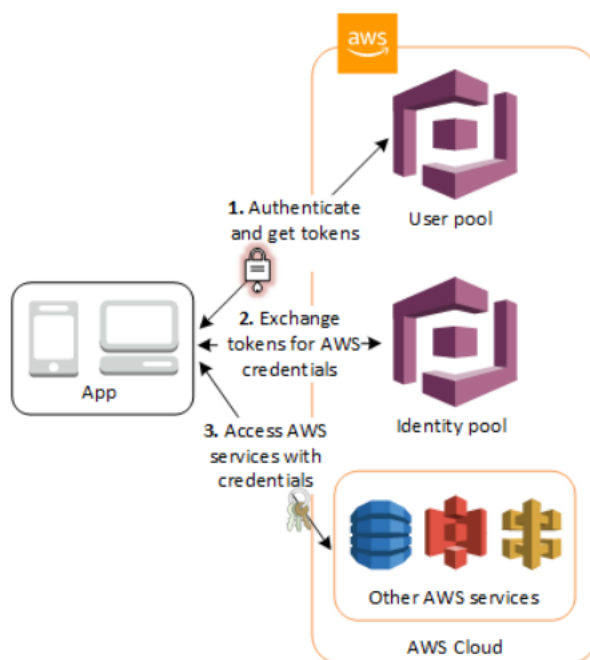


Figura 31: Funcionamiento de Amazon Cognito

### 5.3. Resultados

## **6. Implementar algoritmo para la detección del rostro y ojos**

### **6.1. Objetivo**

realizar la detección de rostro y ojos en la Jetson Nano .

### **6.2. Descripción**

### **6.3. Resultados**

## **7. Implementar puntos faciales en el rostro y la metrica MOR**

### **7.1. Objetivo**

Implementar la detección con puntos faciales en el rostro y la metrica mor para detectar si la boca se encuentra abierta o cerrada.

### **7.2. Descripción**

### **7.3. Resultados**



## 8. Conclusiones

## 9. Bibliografía

### Referencias

- [1] F. Martinez. *Protección de rutas con React Router Dom*, DEV Community. <https://dev.to/franklin030601/proteccion-de-rutas-con-react-router-dom-144j> (accedido el 7 de marzo de 2023).
- [2] *¿Qué es Amazon DynamoDB?*, Amazon Docs. <https://docs.aws.amazon.com/es-es/amazondynamodb/latest/developerguide/Introduction.html> (accedido el 2 de marzo de 2023).
- [3] *API sin servidor de GraphQL y de publicación o suscripción – AWS AppSync – Amazon Web Services*. Amazon Web Services, Inc. <https://aws.amazon.com/es/appsync/> (accedido el 12 de marzo de 2023).
- [4] *AWS — Gestión de identidades y autenticación de usuario en la nube*, Amazon Web Services, Inc. <https://aws.amazon.com/es/cognito/> (accedido el 8 de marzo de 2023).
- [5] 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].
- [6] 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].
- [7] 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].