P279	**FICHA TÉCNICA**	
	PoC y MVP de reconocimiento de imágenes	Fecha Act.: 05/02/2024

ANÁLISIS TÉCNICO

1. **Información General**

Título	Reconocimiento de imágenes	**Código del Proyecto**	P279
Aplicación	reconomiento-imagenes	**Responsable del servicio**	Hassan Taleb
Elaborado por	Claudio Andreé Ampuero Ramos		
Fecha de Versión	05/02/2024	**Versión**	1.0.2

2. **Control de versiones**

Versión	Fecha de la versión	Descripción del cambio
1.0.1	22/11/2024	Especificación del servicio API Rest
1.0.2	23/12/2024	Especificación de arquitectura, detalles del proyecto, componentes y clases, resultados de pruebas
1.0.3	05/02/2024	Modificaciones para MVP: • Inclusión de Api Gateway y Cognito • Eliminacion de DynamoDB • Eliminacion de caso ID_IN_FACE • Finalización de especificación de API Rest
1.0.4	06/02/2024	Correcciones

P279	**FICHA TÉCNICA**	
	PoC y MVP de reconocimiento de imágenes	Fecha Act.: 05/02/2024

3. Detalle de la solución técnica

El sistema se ha desarrollado con las siguientes especificaciones:

- El sistema realiza el match entre un rostro a validar (target) y un rostro tomado como fuente de confianza(source).
- El rosotro fuente de confianza puede venir en 2 formas:
  - Rostro de la fotografía principal de un documento de identidad oficial (DNI o Pasaporte)
  - Rostro de una fotografía tomada a la persona y de la que se tenga confianza de ser real.
- Se plantean 2 casos de uso para el procedimiento de validación de rostro:
  - FACE_TO_FACE: Se realiza una comparación entre una imagen con un rostro que se considera valido y la imagen con el rostro de la persona a verificar.
  - ID_TO_FACE: Se realiza la comparación entre la fotografía de un documento de identidad oficial (DNI o pasaporte) tomado como fuente de confianza, y, la fotografía de la persona a validar.
- Se hace uso de una función lambda como orquestador de la lógica de validación y uso de los servicios.
- Se hace uso de 2 buckets para el almacenamiento de las imágenes source y target.
- Se hace uso de Rekognition para el proceso de reconocimiento de rostros, reconocimiento de documentos y comparación de rostros.
- La función lambda realiza procesamientos de corte sobre las imágenes para reducir la zona de acción sobre las que se realiza la comparación de rostros en las imágenes enviadas.
- La función lambda se expone a través de un API Gateway para ser consumida como API Rest
- Se hace uso de OAuth 2.0 client credentials por medio del servicio AWS Cognito para la autorización del uso del servicio.
- La lambda puede recibir los siguientes campos en el body request:
  - source_path: Ruta donde se encuentra la imagen source para la imagen target subida. La ruta tiene la siguiente forma: ruta/a/archivo.jpg.
  - type: Indica el tipo de caso de uso a invocar en la lambda (FACE_TO_FACE y ID_TO_FACE)
  - threshold: Valor umbral mínimo de similitud para considerar que 2 rostros hacen match. Si no se especifica, se usa el threshold por defecto del servicio (80).
  - threshold_label: Valor umbral mínimo de confianza para el reconocimiento de labels en una imagen. Esta funcionalidad se usa para el caso ID_TO_FACE para reconocer la existencia de documentos de identidad de la imagen fuente. Si no se especifica, se usa el threshold por defecto del servicio (55).

- id_request: Código uuid para hacer trazabilidad del request.
- Se hace uso de una lifecycle rule en el bucket target para trasladar los objetos de más de 30 días de antigüedad a la capa Infrequent Access de S3.

4. Arquitectura AWS

En el siguiente diagrama se muestra la infraestructura de la solución propuesta la cual consiste en los siguientes componentes:

- *Bucket*s S3: Se hace uso de este servicio para el almacenamiento de las imágenes a usar para el proceso de validación de rostros. Se hace uso de hasta 2 buckets s3:
  - Source bucket: Para almacenar las imágenes usadas como bases de verdad (ground truth) para el proceso de validación (imagen de persona o documento de identidad).
  - Target bucket: Para almacenar las imágenes que serán validadas contra las bases de verdad. Este bucket tiene un lifecycle rule para trasladar los objetos a una capa Infrequent Access luego de 30 días de haber sido creados.
- *Lambda:* Contiene el código con la lógica necesaria para orquestar los servicios de S3 y Rekognition para realizar el proceso de validación. La lambda incluye la siguiente lógica
  - Validación del request de entrada.
  - Procesamiento de la validación según caso (FACE_TO_FACE y ID_TO_FACE).
- *Api Gateway:* Se encarga de exponer la lambda en forma de API Rest.
- *Cognito*: Se encarga de la autorización del uso del API con OAuth2.0 client credentials. El usuario final consume un endpoint de autorización de cognito para obtener el token. El token es enviado al Api Gateway el cual la valida con Cognito.
- *Rekognition:* Usado para procesar las imágenes. Se hace uso de 3 funciones del servicio.
  - detect_labels:
    - Función para detectar las etiquetas presentes en una imagen.
    - Si el grado de confianza es suficiente, también se detalla los bounding boxes de cada instancia de la etiqueta en la imagen.
    - Se hace uso para verificar la existencia de documentos de identidad para el caso ID_TO_FACE.
    - También, el bounding box es usado para demarcar la porción de imagen que coincide con un documento de identidad.
    - Los labels usados para reconocer un documento de identidad validos son: Driving License y Passport.
    - El umbral por defecto de la función es 55%. Sin embargo, es posible especificarlo desde el body request que recibe el api.
  - detect_faces:
    - Función para detectar los rostros presentes en una imagen por medio de bounding boxes.
    - Se hace uso para verificar la existencia de personas en la imagen.

- Se hace uso para demarcar la porción de imagen con el rostro de la persona, tanto para los documentos de identidad como para los rostros de las personas a validar.
- El umbral por defecto de la función es 80%.

- compare_faces:
  - Función para comparar la similitud entre 2 rostros.
  - Usado en todos los casos de uso para comparar la similitud entre rostros.
  - El umbral por defecto de la función es 80%.
  - Para reducir el ruido de la comparación se hace uso de las 2 funciones anteriores para reducir el espacio de acción de las imágenes a comparar.

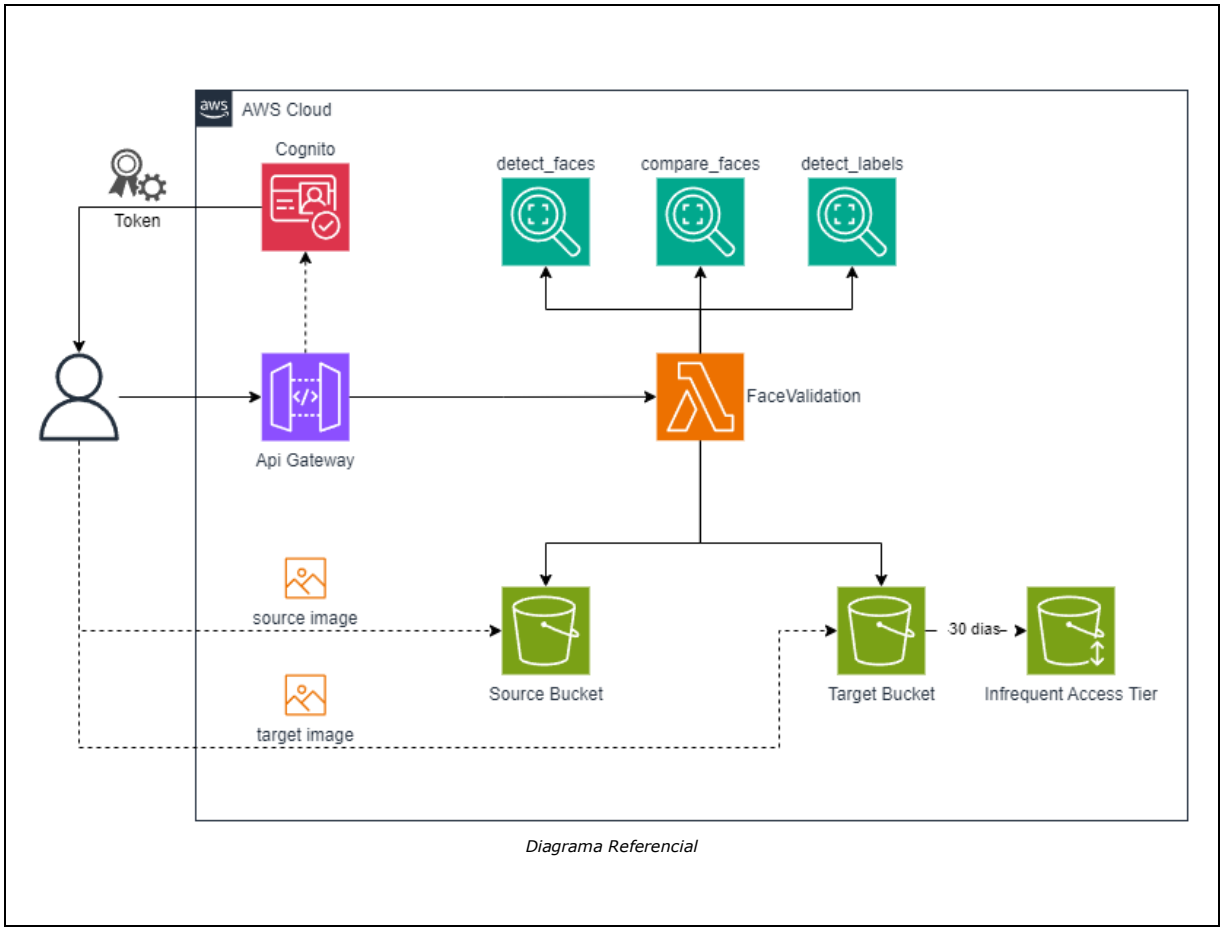


Diagrama Referencial

5. Objetos de la aplicación

a. Lista de componentes externos

Ítem	Paquete o componente	Versión	Plataforma	Descripción	Servidor

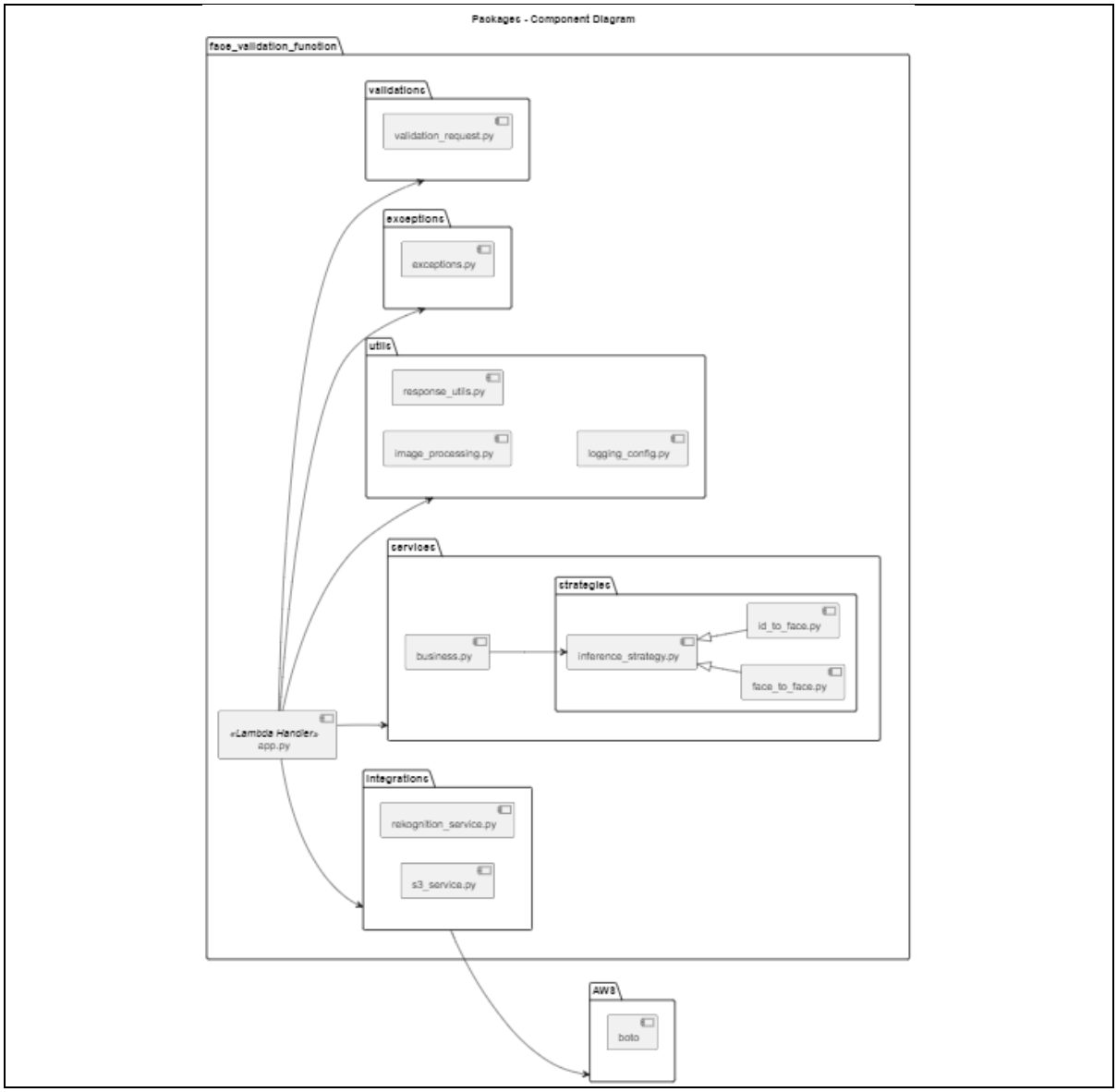
1	requests	-	Python	Servidor web	AWS Lambda
2	six	-	Python	Capa de compatibilidad de versiones de python 2 y python 3.	AWS Lambda
4	regex	-	Python	Dependencia para usar patrones regex en python.	AWS Lambda
5	boto3	-	Python	Dependencia para consumir servicios de AWS.	AWS Lambda
6	Pillow	-	Python	Dependencia para manipular imágenes	AWS Lambda
7	pydantic	-	Python	Dependencia para validar, analizar y convertir datos en python.	AWS Lambda

b. Diagrama de componentes de la aplicación

En la siguiente imagen se muestra el diagrama de componentes del código de la lambda:

- face_validation_function: Paquete con los subpaquetes y componentes necesarios para la ejecución de la lambda de validación.
- services: Paquete de la capa de servicios (lógica de negocio). Este paquete incluye:
  - business.py
  - strategies: Paquete con la interfaz y estrategias aplicables para la validacion de rostros.
    - inference_strategy.py: Contiene la interfaz de estrategias
    - face_to_face.py: Contiene la definición de la estrategia face to face.
    - id_to_face.py: Contiene la definición de la estrategia id to face.
- integrations: Paquete de la capa de integración a los servicios externos. En este caso a servicios de AWS.
  - rekognition_service.py
  - s3_service.py
- Validations: Paquete con las utilidades necesarias para realizar validaciones de objetos
  - validation_request.py
- exceptions: Paquete con las excepciones propias del sistema
  - Exceptions.py
- utils: Paquete con las utilidades usadas a lo largo del sistema.
  - image_processing.py
  - response_utils.py
  - logging_config.py

P279	**FICHA TÉCNICA**	
	PoC y MVP de reconocimiento de imágenes	Fecha Act.: 05/02/2024



c. Diagrama de clases

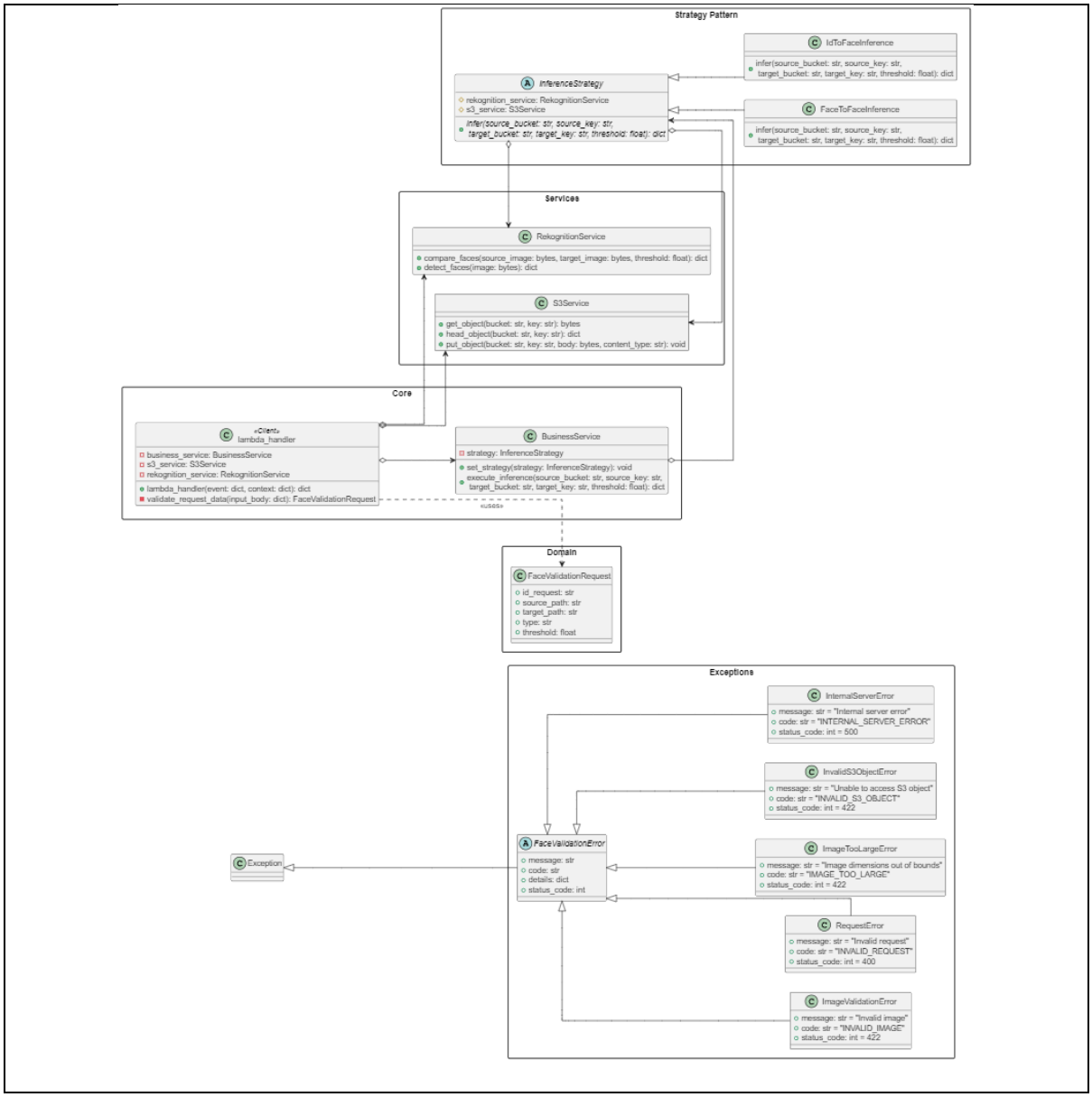
En la siguiente imagen se muestra el diagrama de clases, de la cual se puede resaltar lo siguiente:

- lambda_handler: Esta no es una clase como tal, sino es la función handler de la lambda. En el ámbito de esta función se definen el servicio de negocio (business_service) y los servicios externos (s3_service, rekognition_service). Adicionalmente, se tienen 3 funciones las cuales son:

    - validate_request_data: Contiene la lógica de validación y manejo de errores de validación del request de entrada.
- BusinessService: Esta clase funge de clase "context" del patrón strategy. Esta clase mantiene una referencia a una clase que implemente la interfaz InferenceStrategy. Así, el cliente que haga uso de la clase (lambda_handler) es el encargado de definir el tipo de strategia a usar (FACE_TO_FACE o ID_TO_FACE). Esta clase tiene las siguientes funciones:
    - set_strategy: Usado para establecer la estrategia que debe usar la clase BusinessService.
    - execute_inference: usado para llamar a la función *execute* que implementan todas las clases estrategia.
- InferenceStrategy: Clase abstracta que funge de interfaz a ser implementada por toda clase estrategia.
- FaceToFaceInference: Clase estrategia para implementar el caso de uso FACE_TO_FACE. Incluye la siguiente lógica en su función *execute:*
    - Descarga la imagen del bucket source
    - Corrige la orientación de la imagen source con los metadatos EXIF.
    - Valida que en la imagen aparezca un rostro humano usando rekognition (detect_faces).
    - Recorta la zona de la imagen source que es del rostro humano.
    - Descarga la imagen target del bucket target
    - Corrige la orientación de la imagen target con los metadatos EXIF.
    - Valida que la imagen aparezca un rostro humano usando rekognition (detect_faces).
    - Recorta la zona de la imagen source que es del rostro humano.
    - Compara ambos rostros usando rekognition (compare_faces).
    - Retorna el resultado de la comparación.
- IDToFaceInference: Clase estrategia para implementar el caso de uso ID_TO_FACE. Incluye la siguiente lógica en su función *execute:*
    - Descarga la imagen del documento del bucket source.
    - Corrige la orientación de la imagen source con los metadatos EXIF.
    - Detecta los labels presentes en la imagen usando rekognition (detect_labels)
    - Con los labels se valida que en la imagen existan instancias de documentos de identidad oficiales. Se hace uso de los labels 'Driving License' y 'Passport'.
    - Se recorta la zona de la imagen que es reconocida como documento de identidad.
    - Obtiene el rostro más prominente en la zona del documento (detect_faces)
    - Recorta la imagen del documento con la zona del rostro reconocido.
    - Descarga la imagen de la persona del bucket target.
    - Corrige la orientación de la imagen target con los metadatos EXIF.
    - Valida que exista un rostro humano en la imagen target (detect_faces).
    - Recorta la zona de la imagen con el rostro humano identificado.
    - Compara ambos rostros usando rekognition (compare_faces)
    - Retorna el resultado de la comparación
- RekognitionService: Servicio para abstraer las acciones sobre el servicio rekognition de aws.

- S3Service: Servicio para abstraer las acciones sobre el servicio S3 de aws.
- FaceValidationRequest: Clase para validar el input request de entrada de la lambda.
- FaceValidationError: Clase que hereda de la clase Exception. Sirve para definir las excepciones personalizadas del sistema. De esta heredan las siguientes excepciones:
    - RequestError: Clase excepción para los errores de validación del input request de la lambda.
    - ImageValidationError: Clase excepción para cuando alguna de las imágenes no supera las validaciones.
    - ImageTooLargeError: Clase excepción para cuando la imagen referenciada tiene dimensiones mayores a las aceptadas por rekognition.
    - InvalidS3ObjectError: Clase excepción para cuando la referencia a la imagen en el bucket S3 no refiere a una imagen existente.
    - InternalServerError: Clase excepción para cualquier caso que este fuera de los casos anteriormente descritos.

P279	**FICHA TÉCNICA**	
	PoC y MVP de reconocimiento de imágenes	Fecha Act.: 05/02/2024



d. Lista de variables de entorno

Ítem	Nombre	Descripción	Valor por defecto
1	SOURCE_BUCKET	Nombre del bucket donde se guardarán las imágenes usadas como fuente de confianza.	-

2	TARGET_BUCKET	Nombre del bucket donde se guardarán las imágenes target a validar.	-
3	LOG_LEVEL	Nivel de logging que implementará el lambda.	DEBUG

6. Pruebas sobre la aplicación

a. Preparación de data de prueba

La prueba fue realizada con las imágenes de prueba enviadas por el cliente. Las pruebas se estructuraron en 3 carpetas por cada caso de uso a probar. En cada carpeta se crearon carpetas numeradas donde dentro de cada uno se encontraban las imágenes usadas para la prueba. La estructura de carpetas fue la siguiente.

```
+---face_to_face
|   +---001
|   |       target_face.jpeg
|   |       true_face.jpeg
|   |
|   +---002
|   |       target_face.jpg
|   |       true_face.png
|   |
|   +---003
|   |       target_face.jpeg
|   |       true_face.jpeg
|   ...
+---id_to_face
|   +---001
|   |       dni.jpeg
|   |       target_face.jpeg
|   |
|   +---002
|   |       dni.png
|   |       target_face.jpg
|   |
|   +---003
|   |       dni.jpeg
|   |       target_face.jpeg
|   |
|   ...
\---id_in_face
    +---001
    |       id_in_face.jpeg
    |
    +---002
    |       id_in_face.jpg
    |
```

```
+---003
|       id_in_face.jpeg
|
...
```

Adicionalmente, algunas imágenes fueron recortadas para crear casos de face_to_face.

b. Planteamiento de la prueba

Se definió un archivo csv con la ubicación de cada uno de los casos de prueba y se siguió la siguiente lógica:

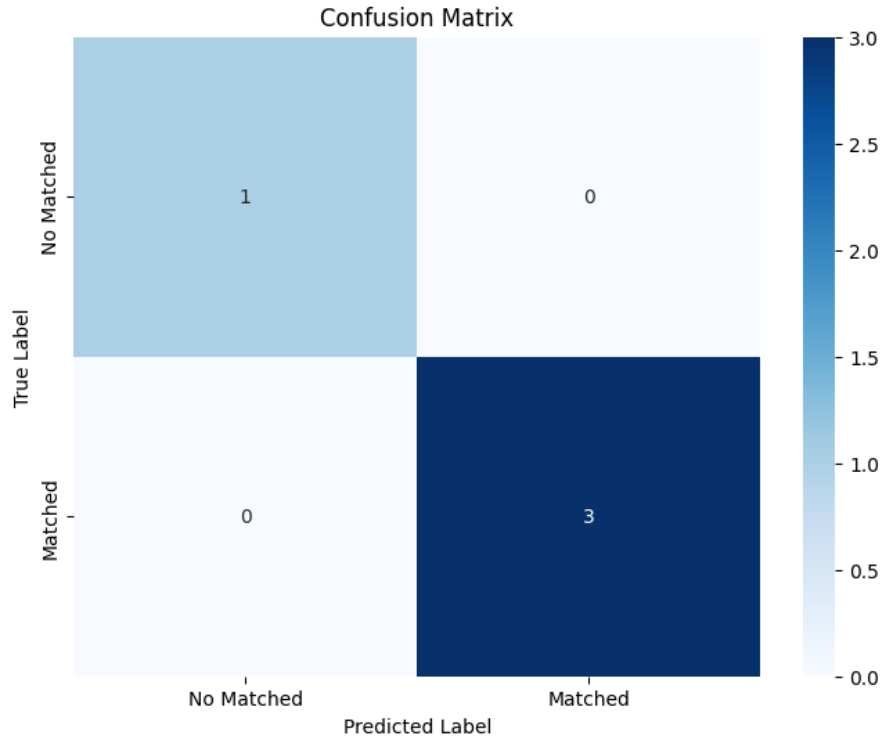
- Carga del csv en un dataframe
- Recorrido por cada fila del dataframe
- Carga de la imagen source al correspondiente bucket
- Carga de la imagen target al correspondiente bucket (se añade metadatos para indicar el tipo de caso de uso a ejecutar)
- Espera y recuperación del resultado en la tabla dynamo db
- Escritura de los resultados en la fila del dataframe

c. Resultados de la prueba

Una vez ejecutado la prueba, se obtuvo los siguientes resultados para cada caso de uso.

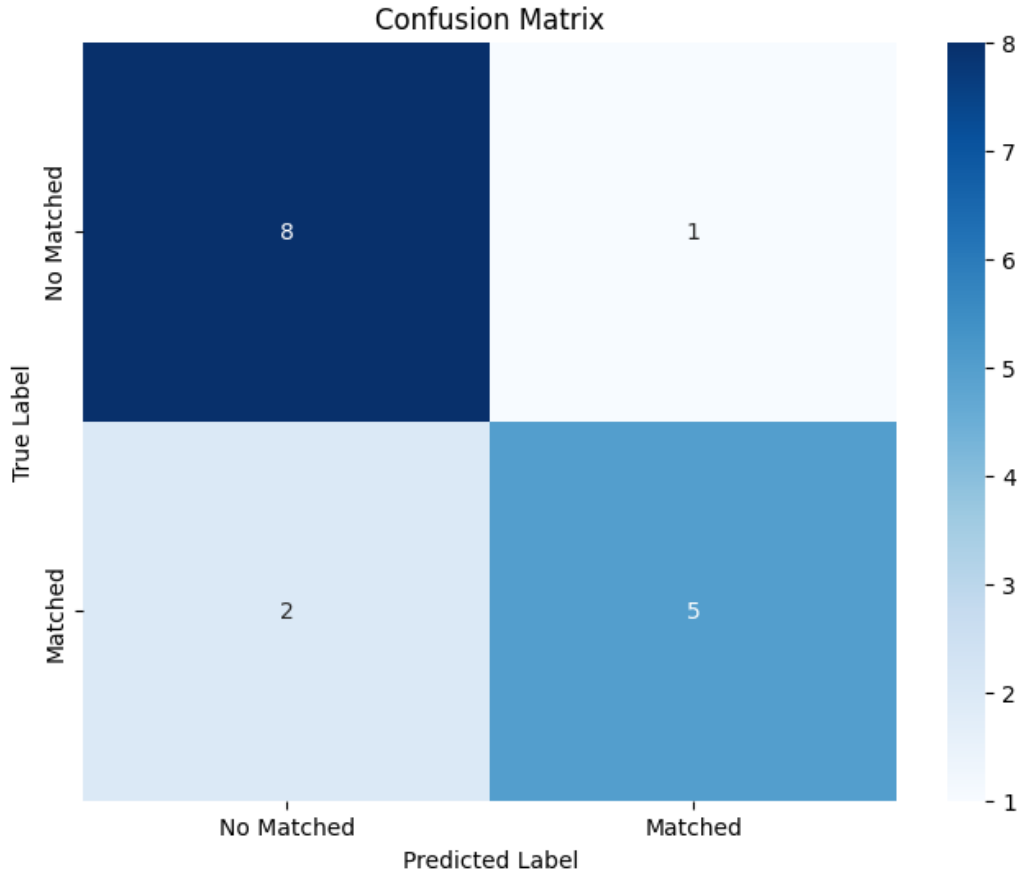
	FACE TO FACE	ID TO FACE	ID IN FACE
True Positives	3	5	2
True Negatives	1	8	9
False Positives	0	1	2
False Negatives	0	2	2
Accuracy	1	0.812	0.733
Precision	1	0.833	0.500
Recall	1	0.714	0.500
F1 Score	1	0.769	0.500

P279	**FICHA TÉCNICA**	
	PoC y MVP de reconocimiento de imágenes	Fecha Act.: 05/02/2024

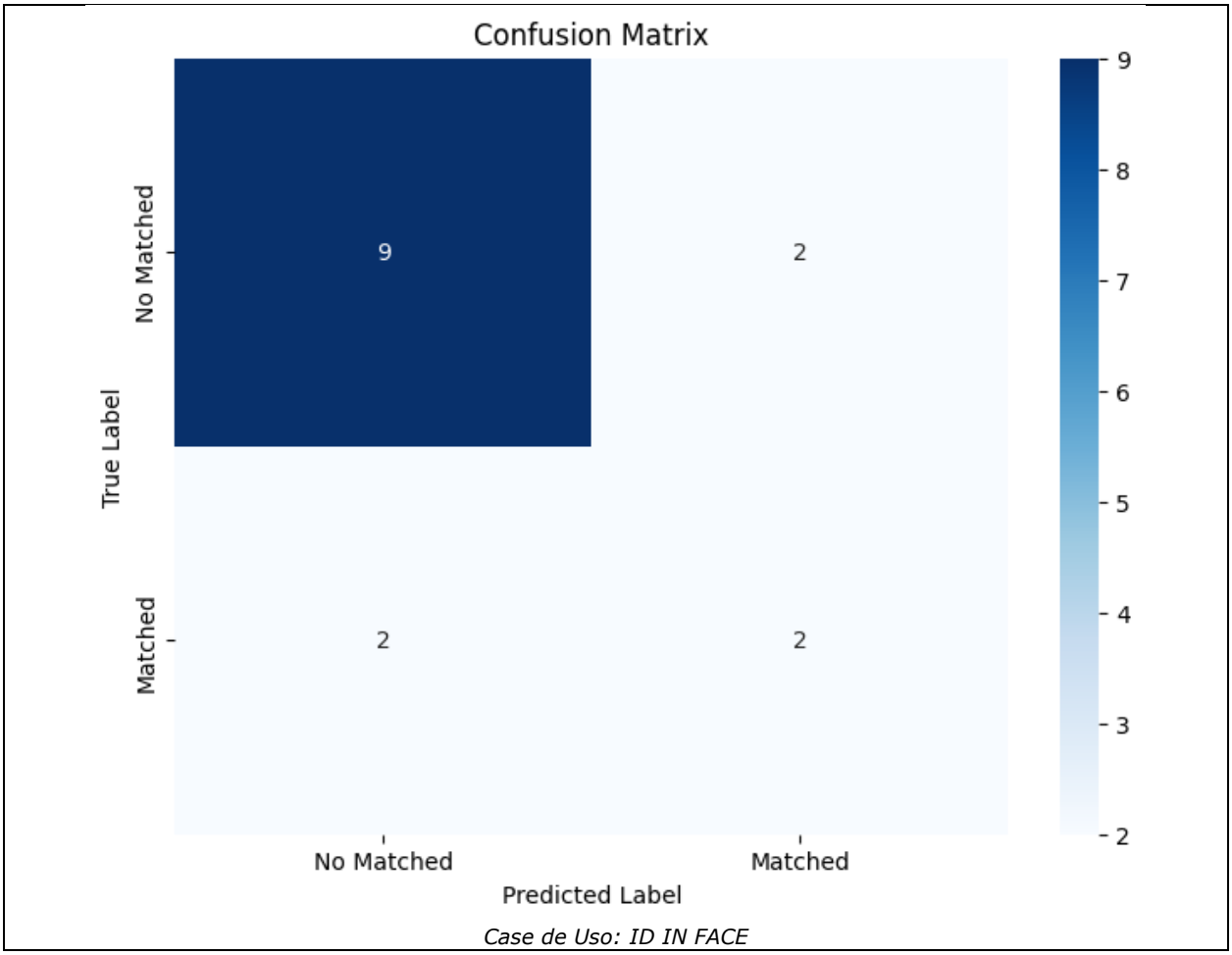


Case de Uso: FACE TO FACE

P279	**FICHA TÉCNICA**	
	PoC y MVP de reconocimiento de imágenes	Fecha Act.: 05/02/2024



Case de Uso: ID TO FACE



Case de Uso: ID IN FACE

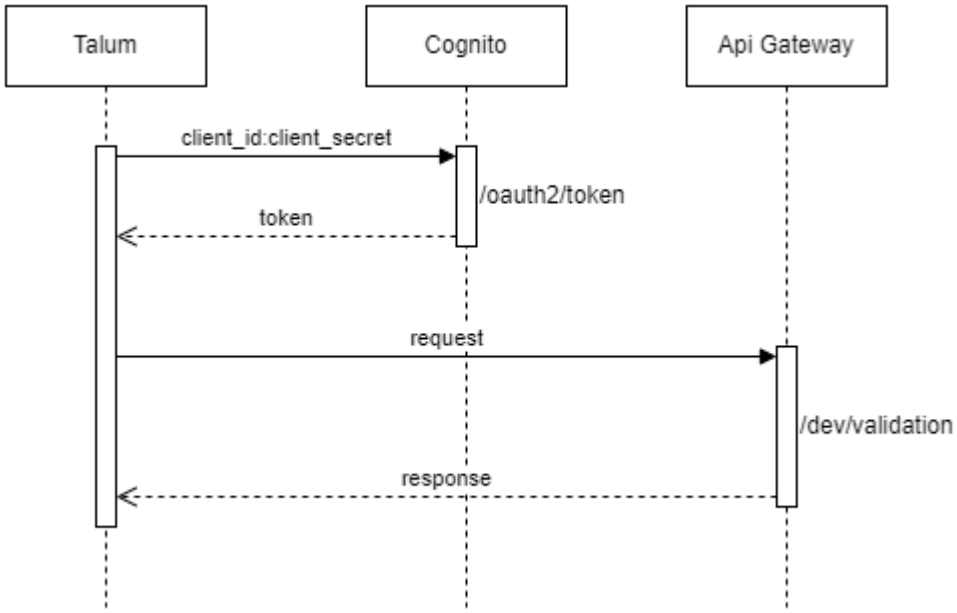
7. API

a. Seguridad

El api se expone mediante el servicio de Api Gateway de AWS. El api es público y protegido con AWS Cognito.

La autorización se implementó haciendo uso de OAuth2.0 y con el grant de client credentials. Para que las aplicaciones clientes accedan al api, estas deben de obtener un access token a partir del servicio de autenticación. El access token solo será válido por un tiempo limitado antes de expirar. Depende de la aplicación cliente realizar la lógica para obtener y gestionar el token.

P279	**FICHA TÉCNICA**	
	PoC y MVP de reconocimiento de imágenes	Fecha Act.: 05/02/2024



b. **Tabla resumen de API's**

ID	PATH	MÉTODO	DESCRIPCIÓN CORTA
1	https://talum-face-validation.auth.us-east-1.amazoncognito.com/oauth2/token	POST	Servicio de autorización para obtener token de acceso y hacer uso del api rest del servicio de validación
2	https://cumj0yz8zc.execute-api.us-east-1.amazonaws.com/dev/validation	POST	Servicio de validación de rostros

c. Credenciales del API

ID	CAMPO	VALOR
1	CLIENT ID	4uunhvnq6opq4vl8ttfr96usa9
2	CLIENT SECRET	1a02h3mos866c3sd05fnvan30e5rnnseqsoks47j2c6bcv5cit32

3	SCOPE	face-detection/write

d. **Detalles de API's**

i. **POST - /oauth2/token**

Servicio de autorización para obtener token de acceso y hacer uso del api rest de validación de rostros.

Request

Headers

- "Content-Type": "application/x-www-form-urlencoded"

Alternativamente, también es posible enviar el client_id y client_secret como header en lugar del body.

- "Authorization": "Basic <Base64Encode(client_id:client_secret)>"

Body

El servicio recibe un conjunto de llaves-valores de tipo x-www-form-urlencoded, donde cada campo a enviar es:

Campo	Valor	Descripción
grant_type	client_credentials	Tipo de grant del flujo oauth2.0
scope	face-detection/write	Scope que determina el ambito de la autorización
client_id	Especificado en la tabla superior	Client Id de la app cliente. Es un valor otorgado. Debe de almacenarse en lugar seguro por la app cliente. No enviar en body en caso de ya enviarse en el header
client_secret	Especificado en la tabla superior	No enviar en body en caso de ya enviarse en el header

Responses

Código	Descripción	Body response

200	Se retorna un objeto json con el access token, tiempo de expiración y tipo de token.	{
		"access_token": "xxxxxxxx", (string)
		"expires_in": 3600, (integer)
		"token_type": "Bearer" (string)
		}

ii. POST - /validation

Servicio de validación de rostros usando comparación de rostros con AWS Rekognition.

Request

Headers

- "Content-Type": "application/json"

Body

El servicio recibe un array de objetos JSON, donde cada objeto JSON tiene la siguiente estructura

Campo	Tipo	Notas	Requerido
id_request	string	Identificador del request. Sirve para la trazabilidad del request. El id debe ser generado con uuid v4. Si no se envía, la lambda generará uno automáticamente.	NO
threshold	double	Valor del umbral de similitud entre rostros (0 - 100). Si la similitud entre rostros es mayor o igual a este umbral, entonces se considera que ambos rostros hacen match. En caso de no enviarse este umbral, el sistema usará el umbral por defecto de rekognition de 80.	NO
threshold_label	double	Valor del umbral de confianza para la detección de labels (0 - 100). Si la confianza de la presencia del label (Driving License o Passport) es mayor a este umbral, entonces se considera que sí aparece el label.	NO

		En caso de no enviarse este umbral, el sistema usará el umbral por defecto de rekognition de 55.	
source_path	string	Ruta en el bucket s3 de la imagen de referencia. Dependiendo del caso, la imagen puede ser: • Foto de rostro • Foto de documento de identidad La imagen debe estar en formato JPG o PNG.	SI
target_path	string	Ruta del bucket s3 de la imagen a validar. Dependiendo del caso, la imagen puede ser: • Foto de rostro La imagen debe estar en formato JPG o PNG.	SI
type	string	Tipo de reconocimiento. Se consideran 2 casos: • FACE_TO_FACE: Se envían 2 direcciones de imágenes (source_path y target_path). El sistema presupone que ambas imágenes son de rostros y realiza la comparación respectiva. • ID_TO_FACE: Se envían 2 direcciones de imágenes (source_path y target_path). El sistema presupone que la imagen del source_path es un documento de identidad en buena calidad y en la orientación correcta. El sistema también presupone que la imágen del target_path es la imagen de un rostro humano en buena calidad y en la orientación correcta. El sistema de encarga de extraer la fotografía del rostro del documento de identidad y compararla con la fotografía del target_path.	SI

Ejemplo del body:

```
{
  "id_request": "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx",
  "threshold": 95.5,
  "threshold_label": 85,
  "source_path": "path/to/source.jpg",
  "tareget_path": "path/to/target.jpg",
  "type": "ID_TO_FACE"
}
```

Response

Código	Descripción	Body response

200	Se retorna un objeto json indicando si la validación ha sido exitosa o no.	{
		"id_request": "...",
		"matched": true,
		"similarity": 99.99
		}

Más detalle de los campos alerta devueltos.

Campo	Tipo	Notas
id_request	string	Identificador del request enviado en el request. Sirve para hacer trazabilidad del request.
matched	boolean	Campo que indica si los rostros en las imágenes enviadas hacen match o no
similarity	double	Campo que indica el grado de similitud en las imágenes enviadas

Errors

Todos los errores tienen la siguiente estructura

Campo	Tipo	Notas
id_request	string	Identificador del request enviado en el request. Sirve para hacer trazabilidad del request.
error	object	Objeto json con los detalles del error
error.code	string	Código del error
error.message	string	Mensaje descriptivo del error
error.details	string	Detalles adicionales del mensaje en caso sea necesario.

```
{
  "id_request": "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx",
  "error": {
    "code": "<error_code>",
    "message": "<error_message>",
```

```
    "details": "<error_details>",
  }
}
```

Los casos a tener en cuenta son los siguientes:

Nombre	**Descripción**	**HTTP**
RequestError	Error de validación de los datos de entrada del request	400
ImageTooLargeError	La imagen es demasiado grande para procesar. Las dimensiones de la imagen deben ser: • Mínimo: 80 x 80 píxeles • Máximo: 4096 x 4096 píxeles	422
InvalidS3ObjectError	No es posible acceder a las imágenes con los path proporcionados	422
InternalServerError	Error inesperado o interno de servidor	500

2. **Aprobado por:**

Nombres y Apellidos	***Cargo / Función***	***Firma***
Claudio Andreé Ampuero Ramos	Machine Learning Engineer	