Extract face from Digital ID

Account credentials

In [ ]:
```
#Face API
FACE_API_KEY = "key"
FACE_ENDPOINT = "https://face-recognition-udacity.cognitiveservices.azure.com"
blob_url_id = "https://001finalproject.blob.core.windows.net/step3/digital_id_Al
blob_url_vid="https://001finalproject.blob.core.windows.net/step3/KeyFrameThumbn
url_id_encoded = quote(blob_url_id, safe=":/?&=%")
url_vid_encoded = quote(blob_url_vid, safe=":/?&=%")
face_client = FaceClient(FACE_ENDPOINT, CognitiveServicesCredentials(FACE_API_KE
```

Extract Face

In [73]:
```
from io import BytesIO
from PIL import Image
import requests
from azure.cognitiveservices.vision.face import FaceClient
from azure.cognitiveservices.vision.face.models import RecognitionModel
from msrest.authentication import CognitiveServicesCredentials

# --- Configuración Face ---
face_client = FaceClient(FACE_ENDPOINT,
                         CognitiveServicesCredentials(FACE_API_KEY))

# --- Preprocesado: PNG → JPEG, ≤1024 px, ≤6 MB ---
def preprocess_image(image_bytes: bytes) -> BytesIO:
    img = Image.open(BytesIO(image_bytes)).convert("RGB")  # quita canal alfa
    img.thumbnail((1024, 1024))                            # reduce si es grande
    buf = BytesIO()
    img.save(buf, format="JPEG", quality=90)
    buf.seek(0)
    return buf                                            # stream listo p/Face

# --- Detección y obtención de faceId ---
def get_face_id_from_bytes(image_bytes: bytes) -> str:
    stream = preprocess_image(image_bytes)
    print(f"🧾 Size after preprocess: {stream.getbuffer().nbytes/1024:.1f} KB")
    faces = face_client.face.detect_with_stream(
        image=stream,
        detection_model="detection_01",
        recognition_model=RecognitionModel.recognition_03,
        return_face_id=True
    )
    if not faces:
        raise RuntimeError("No face detected in image")
    return faces[0].face_id

# --- Descarga de las imágenes (aquí defines img_id / img_vid) ---
img_id  = requests.get(blob_url_id).content
img_vid = requests.get(blob_url_vid).content


try:
    face_id_1 = get_face_id_from_bytes(img_id)
```

```
except APIErrorException as e:
    print("----- FACE API ERROR -----")
    print(e.response.text)          # 🔍 Muestra el JSON con la causa exacta
    raise
# --- Obtención de faceId y comparación ---
face_id_1 = get_face_id_from_bytes(img_id)
face_id_2 = get_face_id_from_bytes(img_vid)

verify = face_client.face.verify_face_to_face(face_id_1, face_id_2)
print(f"Match confidence: {verify.confidence*100:.2f}%")
print(f"Same identity?  : {'Yes' if verify.is_identical else 'No'}")
```

```
📄 Size after preprocess: 79.0 KB
----- FACE API ERROR -----
{
  "error": {
    "code": "InvalidRequest",
    "message": "Invalid request has been sent.",
    "innererror": {
      "code": "UnsupportedFeature",
      "message": "Feature is not supported, missing approval for one or more of t
he following features: Identification,Verification. Please apply for access at ht
tps://aka.ms/facerecognition"
    }
  }
}
```

```
---------------------------------------------------------------------------
APIErrorException                         Traceback (most recent call last)
Cell In[73], line 41
     37 img_vid = requests.get(blob_url_vid).content
     40 try:
---> 41     face_id_1 = get_face_id_from_bytes(img_id)
     42 except APIErrorException as e:
     43     print("----- FACE API ERROR -----")

Cell In[73], line 25, in get_face_id_from_bytes(image_bytes)
     23 stream = preprocess_image(image_bytes)
     24 print(f"📄 Size after preprocess: {stream.getbuffer().nbytes/1024:.1f} K
B")
---> 25 faces = face_client.face.detect_with_stream(
     26     image=stream,
     27     detection_model=               ,
     28     recognition_model=RecognitionModel.recognition_03,
     29     return_face_id=True
     30 )
     31 if not faces:
     32     raise RuntimeError("No face detected in image")

File ~\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.11_qbz5n2kfra8p0
\LocalCache\local-packages\Python311\site-packages\azure\cognitiveservices\vision
\face\operations\_face_operations.py:782, in FaceOperations.detect_with_stream(se
lf, image, return_face_id, return_face_landmarks, return_face_attributes, recogni
tion_model, return_recognition_model, detection_model, face_id_time_to_live, cust
om_headers, raw, callback, **operation_config)
    779 response = self._client.send(request, stream=False, **operation_config)
    781 if response.status_code not in [200]:
--> 782     raise models.APIErrorException(self._deserialize, response)
    784 deserialized = None
    785 if response.status_code == 200:

APIErrorException: (InvalidRequest) Invalid request has been sent.
```