API de Detección de Personas

Documentación Técnica

Arquitectura basada en AWS
API Gateway + AWS Lambda + YOLOv8

Versión 1.0

24 de mayo de 2025

Índice

1. Introducción

Esta documentación describe la implementación y uso de una API REST para la detección de personas en imágenes utilizando el modelo YOLOv8. La solución está desplegada en AWS utilizando una arquitectura serverless que combina API Gateway y AWS Lambda.

2. Modelo de Detección

2.1. YOLOv8 (You Only Look Once v8)

El modelo utilizado es YOLOv8, la última versión de la familia YOLO desarrollada por Ultralytics. Este modelo ofrece:

- **Detección en tiempo real**: Optimizado para inferencia rápida
- **Alta precisión**: Mejoras significativas sobre versiones anteriores
- **Flexibilidad**: Soporta múltiples tareas de visión artificial
- **Eficiencia**: Menor uso de recursos computacionales

2.2. Configuración del Modelo

Especificaciones Técnicas

- **Modelo**: YOLOv8n (nano)
- **Clases detectadas**: Persona (class_id = 0)
- **Umbral de confianza**: 0.5 (configurable)
- **Formato de entrada**: Imágenes en base64
- **Resolución**: Adaptable (reescalado automático)

3. Versiones y Dependencias

3.1. Dependencias Principales

```
ultralytics==8.0.196
Pillow==10.0.1
numpy==1.24.3
torch==2.0.1
torchvision==0.15.2
opencv-python-headless==4.8.1.78
```

Listing 1: Dependencias del proyecto

3.2. Entorno de Ejecución

- **Runtime**: Python 3.9
- **Plataforma**: AWS Lambda
- **Memoria**: 2048 MB
- **Timeout**: 30 segundos
- **Arquitectura**: x86_64

4. Arquitectura en AWS

4.1. Diagrama de Arquitectura

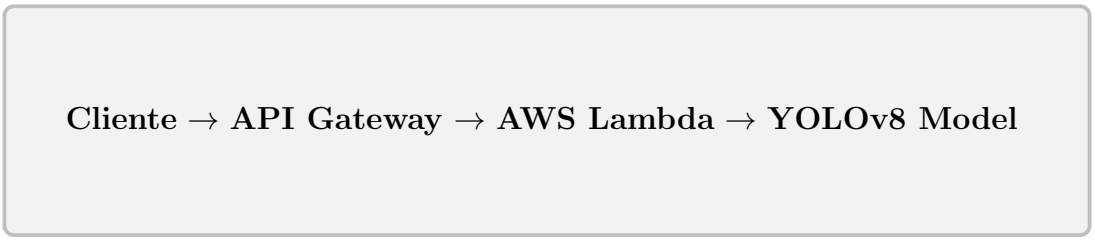

Figura 1: Arquitectura Serverless en AWS

4.2. Componentes

4.2.1. Amazon API Gateway (REST API)

- **Endpoint**: `https://npdvcvx4o8.execute-api.us-east-1.amazonaws.com/prodfaces`
- **Método**: POST
- **Autenticación**: API Key
- **CORS**: Habilitado
- **Usage Plan**: temp-predict-usage-plan (ID: v904mp)

4.2.2. Configuración de Rate Limiting

Usage Plan - temp-predict-usage-plan

- **Request Rate**: 10 requests por segundo
- **Burst Limit**: 10 requests
- **Quota**: 1,000 requests por mes
- **ID del Plan**: v904mp

4.2.3. AWS Lambda

- **Función**: Procesamiento de imágenes con YOLOv8
- **Trigger**: API Gateway
- **Empaquetado**: Container Image o Layer personalizado
- **Escalado**: Automático según demanda

5. Uso de la API

5.1. Endpoint

URL Base

`https://npdvcvx4o8.execute-api.us-east-1.amazonaws.com/prodfaces`

5.2. Autenticación

La API requiere autenticación mediante API Key:

```
Content-Type: application/json
x-api-key: CAx2D1DM4l2lCsaoCVvVjcL88Tm7Tj62LXtjQm39
```

Listing 2: Headers requeridos

5.3. Estructura del Request

5.3.1. Método HTTP

`POST /prodfaces`

5.3.2. Estructura del Body

```
{
  "body": {
    "image": "<IMAGEN_EN_BASE64>"
  }
}
```

Listing 3: Estructura del cuerpo de la petición

Nota Importante

La imagen debe estar codificada en base64. El formato puede ser JPG, PNG, o cualquier formato soportado por Pillow.

5.4. Estructura del Response

5.4.1. Respuesta Exitosa (200)

```
{
  "statusCode": 200,
  "headers": {
    "Content-Type": "application/json",
    "Access-Control-Allow-Origin": "*"
  },
  "body": {
    "detections": [
      {
        "class": "person",
        "confidence": 0.85,
        "bbox": [x1, y1, x2, y2]
      }
    ],
    "total_persons": 1,
    "image_size": [width, height],
    "processing_time": 1.23
  }
}
```

Listing 4: Respuesta exitosa

5.4.2. Respuesta de Error (400/500)

```
{
  "statusCode": 400,
  "headers": {
    "Content-Type": "application/json",
    "Access-Control-Allow-Origin": "*"
  },
  "body": {
    "error": "Descripci n del error"
  }
}
```

Listing 5: Respuesta de error

5.4.3. Errores de Rate Limiting

```
{
  "statusCode": 429,
  "headers": {
    "Content-Type": "application/json",
    "Access-Control-Allow-Origin": "*",
    "X-RateLimit-Limit": "10",
    "X-RateLimit-Remaining": "0",
    "Retry-After": "1"
  },
  "body": {
    "error": "Too Many Requests"
  }
}
```

Listing 6: Error por exceso de requests (429)

```
{
  "statusCode": 403,
  "headers": {
    "Content-Type": "application/json",
    "Access-Control-Allow-Origin": "*"
  },
  "body": {
    "error": "Quota exceeded. Monthly limit of 1000 requests reached."
  }
}
```

Listing 7: Error por cuota agotada (403)

6. Ejemplos de Uso

6.1. Ejemplo con cURL

```
curl -X POST \
  https://npdvcvx4o8.execute-api.us-east-1.amazonaws.com/prodfaces \
  -H 'Content-Type: application/json' \
  -H 'x-api-key: CAx2D1DM4l2lCsaoCVvVjcL88Tm7Tj62LXtjQm39' \
  -d '{
    "body": {
      "image": "/9j/4AAQSkZJRgABAQAAAQABAAD..."
    }
  }'
```

Listing 8: Ejemplo de petición con cURL

6.2. Ejemplo con Python

```
import requests
import base64

# Leer y codificar imagen
with open('imagen.jpg', 'rb') as f:
    image_base64 = base64.b64encode(f.read()).decode('utf-8')

# Configurar petici n
url = 'https://npdvcvx4o8.execute-api.us-east-1.amazonaws.com/prodfaces'
headers = {
    'Content-Type': 'application/json',
    'x-api-key': 'CAx2D1DM4l2lCsaoCVvVjcL88Tm7Tj62LXtjQm39'
}
payload = {
    "body": {
        "image": image_base64
    }
}

# Enviar petici n
response = requests.post(url, json=payload, headers=headers)
result = response.json()

```

```python
print(f"Personas detectadas: {result['body']['total_persons']}")
```

Listing 9: Ejemplo de uso con Python

7. Pruebas con Postman

A continuación se muestran las capturas de pantalla de las pruebas realizadas con Postman:

7.1. Configuración de Headers y Respuesta

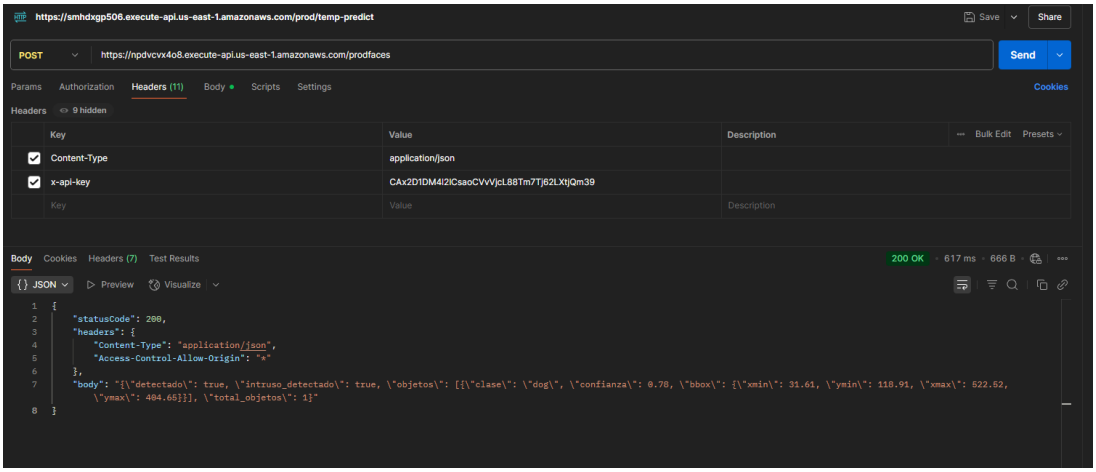

Figura 2: Configuración de headers y respuesta obtenida

7.2. Configuración del Body y Respuesta

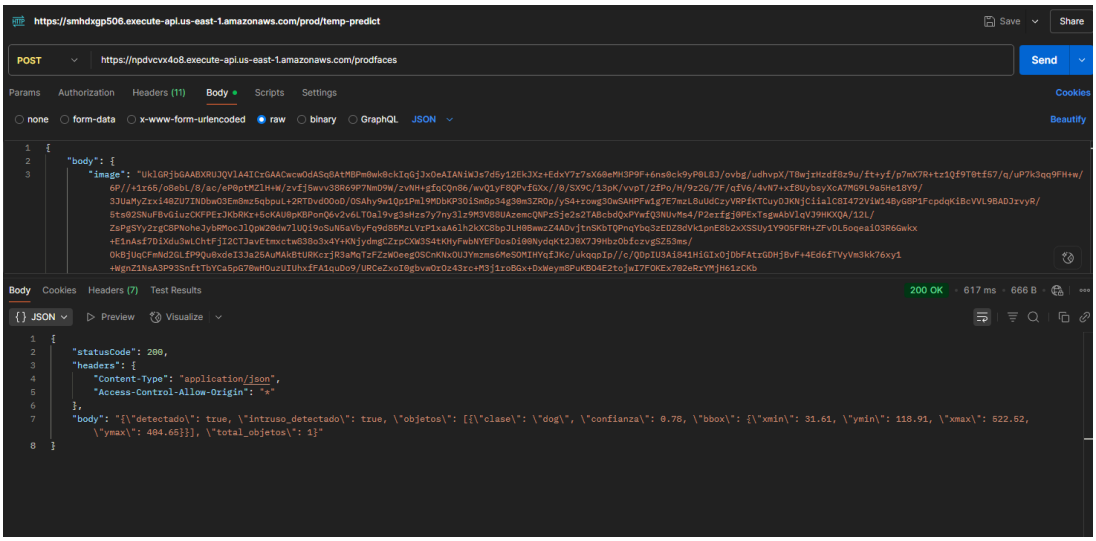

Figura 3: Configuración del body y respuesta detallada

8. Consideraciones de Rendimiento

8.1. Limitaciones

- **Tamaño máximo**: 6 MB (límite de API Gateway)
- **Timeout**: 30 segundos máximo
- **Cold Start**: Primera invocación puede tardar más tiempo
- **Memoria**: 2048 MB asignados
- **Rate Limiting**: 10 requests/segundo (según usage plan)
- **Cuota mensual**: 1,000 requests por API Key
- **Burst limit**: 10 requests simultáneos

8.2. Optimizaciones

- Modelo YOLOv8n para balance precisión/velocidad
- Redimensionado automático de imágenes
- Reutilización de modelo en memoria (warm start)
- Respuestas comprimidas cuando es posible

9. Seguridad

9.1. Medidas Implementadas

- **API Key**: Autenticación requerida
- **HTTPS**: Encriptación en tránsito
- **Rate Limiting**: Prevención de abuso
- **Validación**: Verificación de formato de entrada

9.2. Recomendaciones

- Rotar API Keys periódicamente
- Implementar logging detallado
- Monitorear uso y costos en AWS CloudWatch
- Configurar alertas de CloudWatch para cuotas y rate limits
- Implementar retry logic con exponential backoff en clientes
- Revisar usage plans según patrones de uso reales
- Configurar múltiples API Keys para diferentes servicios/usuarios
- Monitorear métricas de throttling y ajustar límites según necesidad

10. Monitoreo y Logs

10.1. CloudWatch Metrics

- Número de invocaciones
- Duración de ejecución
- Errores y timeouts
- Uso de memoria

10.2. Logs Disponibles

- Logs de API Gateway
- Logs de AWS Lambda
- Métricas de rendimiento
- Errores de procesamiento

11. Conclusiones

La API de detección de personas implementada ofrece una solución escalable y eficiente utilizando tecnologías serverless de AWS. La combinación de YOLOv8 con AWS Lambda proporciona:

- **Escalabilidad automática**
- **Costos optimizados** (pago por uso)
- **Alta disponibilidad**
- **Fácil mantenimiento**