MAPFRE - Procesador de Volantes Médicos

API para extraer información de volantes médicos utilizando Google Gemini Vision.

🎯 Descripción

Esta API recibe una **imagen** o **PDF** de un volante médico en formato Base64 y extrae automáticamente todos los campos del documento utilizando el modelo Gemini 2.5-pro de Google, devolviendo un JSON estructurado con la información.

✨ Características

- ☑ Procesamiento de **imágenes** (JPG, PNG) y **PDFs** ✨ NUEVO
- ☑ Extracción automática de 15 campos del volante médico MAPFRE Salud
- ☑ Gemini Vision API (gemini-2.5-pro) con Vertex AI
- ☑ Respuestas en formato JSON estructurado
- ☑ Interfaz web para subir y procesar archivos
- ☑ Logging detallado de todas las operaciones
- ☑ API REST con FastAPI
- ☑ Documentación interactiva (Swagger/ReDoc)
- ☑ Listo para Docker

📋 Requisitos

- Python 3.12+
- Poetry (para gestión de dependencias)
- Credenciales de Google Cloud Platform con Vertex AI habilitado
- Docker (opcional, para deployment)

🚀 Instalación

1. Clonar el repositorio

```
git clone <url-del-repositorio>
cd lmops_project
```

2. Instalar Poetry

Poetry es un gestor de dependencias moderno para Python. Si no lo tienes instalado:

Windows (PowerShell)

```
(Invoke-WebRequest -Uri https://install.python-poetry.org -
UseBasicParsing).Content | py -
```

Después de la instalación, añade Poetry al PATH (reinicia la terminal después):

```
$env:Path += ";$env:APPDATA\Python\Scripts"
```

Linux / macOS / WSL

```
curl -sSL https://install.python-poetry.org | python3 -
```

Añade Poetry al PATH (añade esto a tu `.bashrc` o `.zshrc`):

```
export PATH="$HOME/.local/bin:$PATH"
```

Verificar instalación

```
poetry --version
```

3. Crear entorno virtual e instalar dependencias

Poetry creará automáticamente un entorno virtual aislado para el proyecto:

```
# Instalar todas las dependencias del proyecto
poetry install
```

Esto creará un entorno virtual en `.venv` (o en la ubicación configurada por Poetry) e instalará todas las dependencias especificadas en `pyproject.toml`.

4. Activar el entorno virtual

Opción A: Usar Poetry shell (Recomendado)

```
poetry shell
```

Esto activa el entorno virtual en un subshell. Para salir, escribe `exit`.

Opción B: Ejecutar comandos con Poetry

Sin activar el entorno, puedes ejecutar comandos con `poetry run`:

```
poetry run python ejemplo_uso.py
poetry run uvicorn app.app:app --reload
```

5. Configurar variables de entorno

Crea un archivo `.env` en la raíz del proyecto:

```
# Copiar plantilla de ejemplo (si existe)
cp .env.example .env
```

Edita el archivo `.env` y configura las variables necesarias:

```
# API Key de Gemini (REQUERIDA)
GEMINI_API_KEY=tu_api_key_aqui

# Modelo de Gemini (opcional)
GEMINI_MODEL=gemini-2.0-flash-exp

# Nivel de logging (opcional)
LOG_LEVEL=INFO

# Entorno (opcional)
ENVIRONMENT=dev
```

Obtén tu API key de Gemini en: https://aistudio.google.com/api-keys

```
## 🎮 Uso

### Iniciar el servidor

```bash
poetry run uvicorn app.app:app --host 0.0.0.0 --port 8000 --reload
```

Acceso a la aplicación

- 🌐 **Frontend**: http://localhost:8000/
- 📡 **API**: http://localhost:8000/v1/image/process-image
- 📖 **Docs (Swagger)**: http://localhost:8000/docs
- 📓 **ReDoc**: http://localhost:8000/redoc
- 💚 **Health**: http://localhost:8000/healthcheck

📡 API Endpoint
```

POST `/v1/image/process-image`

Procesa una imagen o PDF del volante médico y extrae la información.

Request Body

```
{
  "file_base64": "base64_encoded_content",
  "mime_type": "application/pdf",
  "prompt": "opcional - usa el prompt por defecto si se omite"
}
```

Parámetros

Campo	Tipo	Requerido	Descripción
`file_base64`	string	☑ Sí	Archivo codificado en Base64 (imagen o PDF)
`mime_type`	string	✗ No	MIME type: `image/jpeg`, `image/png`, `application/pdf`. Default: `image/jpeg`
`prompt`	string	✗ No	Prompt personalizado. Si se omite, usa el prompt optimizado para volante MAPFRE

Nota: También puedes usar `image_base64` en lugar de `file_base64` para retrocompatibilidad.

Response (200 OK)

```
{
  "extracted_data": {
    "filiacion_asegurado": "Juan García López",
    "codigo_servicio_concertado": "12345",
    "numero_documento": "12345678A",
    "Profesional_prescriptor": "Dr. María Martínez",
    "Numero_de_colegiado": "280012345",
    "Especialidad": "Medicina General",
    "prescripcion": "Radiografía de tórax",
    "fecha_primeros_sintomas": "15/10/2024",
    "motivos_sintomas": "Dolor torácico",
    "prestacion_sanitaria": "Diagnóstico por imagen",
    "numero_autorizacion": "AUTH123456",
    "codigo_servicio_realizador": "67890",
    "firma_profesional_realizador": true,
    "firma_asegurado": true,
    "firma_sello_prescriptor": true,
    "fecha_realizacion": "20/10/2024",
    "origen_patologia": "Accidente"
  }
}
```

Ejemplos de Uso

Con Imagen (PowerShell)

```powershell
$imageBase64 = [Convert]::ToBase64String([IO.File]::ReadAllBytes("volante.jpg"))

$body = @{
    file_base64 = $imageBase64
    mime_type = "image/jpeg"
} | ConvertTo-Json

Invoke-RestMethod -Uri "http://localhost:8000/v1/image/process-image" `
    -Method Post -Body $body -ContentType "application/json"
```

Con PDF ✨ (PowerShell)

```powershell
$pdfBase64 = [Convert]::ToBase64String([IO.File]::ReadAllBytes("volante.pdf"))

$body = @{
    file_base64 = $pdfBase64
    mime_type = "application/pdf"
} | ConvertTo-Json

Invoke-RestMethod -Uri "http://localhost:8000/v1/image/process-image" `
    -Method Post -Body $body -ContentType "application/json"
```

Con Python

```python
import base64
import requests

# Procesar PDF
with open("volante.pdf", "rb") as file:
    file_base64 = base64.b64encode(file.read()).decode('utf-8')

response = requests.post(
    "http://localhost:8000/v1/image/process-image",
    json={
        "file_base64": file_base64,
        "mime_type": "application/pdf"
    }
)

print(response.json())
```

Desde el Frontend

1. Abre http://localhost:8000/
2. Arrastra un archivo (imagen o PDF) o haz clic para seleccionar
3. Click en "🚀 Procesar Volante"
4. ¡Listo! Visualiza los datos extraídos

🐳 Docker

Construcción

```
docker build -t autorizaciones-salud-icai:latest .
```

Ejecución

⚠️ **IMPORTANTE**: Debes pasar la API key de Gemini como variable de entorno en el momento de ejecutar el contenedor:

```
docker run -d \
  --name autorizaciones-salud-icai \
  -p 8000:8000 \
  -e GEMINI_API_KEY=tu_api_key_aqui \
  autorizaciones-salud-icai:latest
```

PowerShell (Windows)

```
docker run -d `
  --name autorizaciones-salud `
  -p 8000:8000 `
  -e GEMINI_API_KEY=tu_api_key_aqui `
  autorizaciones-salud:latest
```

Nota: Reemplaza `tu_api_key_aqui` con tu API key real de Gemini obtenida desde https://makersuite.google.com/app/apikey

Verificar que está funcionando

```
# Ver logs del contenedor
docker logs autorizaciones-salud

# Verificar el healthcheck
curl http://localhost:8000/healthcheck
```

Ver guía completa en: DOCKER_DEPLOYMENT.md

📊 Campos Extraídos

La API extrae automáticamente los siguientes 15 campos del volante médico:

1. Datos del Asegurado

- `filiacion_asegurado` - Nombre completo del asegurado
- `numero_documento` - DNI/NIE

2. Datos del Servicio

- `codigo_servicio_concertado` - Código del centro médico que deriva
- `codigo_servicio_realizador` - Código del centro que realiza el servicio

3. Datos del Profesional Prescriptor ✨

- `Profesional_prescriptor` - Nombre del médico que prescribe
- `Numero_de_colegiado` - Número de colegiado del médico
- `Especialidad` - Especialidad médica del prescriptor

4. Información Médica

- `prescripcion` - Descripción de la prueba/tratamiento prescrito
- `fecha_primeros_sintomas` - Fecha de inicio de síntomas
- `motivos_sintomas` - Descripción de síntomas/motivo consulta
- `prestacion_sanitaria` - Tipo de prestación solicitada
- `origen_patologia` - Origen de la patología (accidente, enfermedad común, etc.)

5. Autorizaciones y Firmas

- `numero_autorizacion` - Número de autorización (si existe)
- `firma_profesional_realizador` - ¿Tiene firma del profesional? (boolean)
- `firma_asegurado` - ¿Tiene firma del asegurado? (boolean)
- `firma_sello_prescriptor` - ¿Tiene firma y sello del prescriptor? (boolean)
- `fecha_realizacion` - Fecha de realización del servicio

🏗️ Estructura del Proyecto

```
autorizaciones-salud/
├── app/
│   ├── app.py                    # Aplicación principal FastAPI
│   ├── constants.py              # Prompts y configuración
│   ├── auth/
│   │   └── credentials_tts.json  # Credenciales GCP (no en repo)
│   ├── routers/
│   │   └── agent.py              # Endpoint /process-image
│   └── services/
│       ├── ai_service.py         # Servicio Gemini Vision
```

```
│       └── logging_service.py     # Logging
├── frontend/
│   ├── index.html                 # Interfaz web principal
│   ├── simple.html                # Interfaz simplificada
│   └── serve.py                   # Servidor local de desarrollo
├── Dockerfile                     # Configuración Docker
├── pyproject.toml                 # Dependencias Poetry
├── README.md                      # Este archivo
├── DOCKER_DEPLOYMENT.md           # Guía de despliegue Docker
├── CHANGELOG_PDF_SUPPORT.md       # Cambios del soporte PDF
└── test_pdf_support.py            # Script de testing
```

Testing

Script de prueba incluido

```
# Procesar un PDF
python test_pdf_support.py volante.pdf

# Procesar una imagen
python test_pdf_support.py volante.jpg

# Comparar imagen vs PDF
python test_pdf_support.py volante.jpg volante.pdf
```

Healthcheck

```
curl http://localhost:8000/healthcheck
```

Configuración

Configuración de Gemini

Definida en `app/services/ai_service.py`:

```
self.gemini_client = genai.Client(
    vertexai=True,
    project="pre-ami-mespana-c8aa",
    location="europe-west1",
)
self.model_name = "gemini-2.5-pro"
```

Prompt del Volante MAPFRE

El prompt optimizado está en `app/constants.py` como `ImagePrompts.VOLANTE_MAPFRE_PROMPT`.

Incluye:

- Instrucciones detalladas para OCR
- Definición de los 15 campos a extraer
- Manejo de campos vacíos/ilegibles
- Validación de firmas y sellos
- Formato JSON de salida

Changelog

v2.0 - Soporte para PDFs ✨

- ☑ Soporte nativo para archivos PDF
- ☑ Campo `mime_type` en request
- ☑ Preview adaptativo en frontend (imagen vs PDF)
- ☑ Retrocompatibilidad con `image_base64`
- ☑ Logging mejorado

Ver detalles completos en: CHANGELOG_PDF_SUPPORT.md

v1.0 - Release Inicial

- ☑ Procesamiento de imágenes JPG/PNG
- ☑ 15 campos extraídos del volante
- ☑ Frontend web
- ☑ API REST con FastAPI
- ☑ Docker support

Seguridad

⚠ **IMPORTANTE**: Las credenciales de GCP se copian en la imagen Docker durante el build.

Para desarrollo: Está bien **Para producción**: Usar Azure Managed Identity, Key Vault o Kubernetes Secrets

Ver mejores prácticas en: DOCKER_DEPLOYMENT.md

Desarrollo

Instalar dependencias de desarrollo

```
poetry install --with dev
```

Ejecutar tests

```
poetry run pytest
```

Formatear código

```
poetry run black app/
poetry run isort app/
```

Licencia

MAPFRE - Uso interno

Contacto

Para preguntas o soporte: jaandr7@mapfre.com

Última actualización: 3 de noviembre de 2025
Versión: 2.0 (Soporte PDF)