📖 Ejemplos de Uso - API REST + Gemini

Esta guía proporciona ejemplos prácticos de cómo consumir la API en diferentes lenguajes de programación.

📋 Tabla de Contenidos

Python

Configuración Inicial

```
import requests
import json

BASE_URL = "http://localhost:8000"
```

1. Subir una Imagen

```
def upload_image(image_path):
    """Sube y analiza una imagen"""
    url = f"{BASE_URL}/files/upload/image"

    with open(image_path, 'rb') as f:
        files = {'file': f}
        response = requests.post(url, files=files)

    return response.json()

# Uso
result = upload_image('imagen.jpg')
print(json.dumps(result, indent=2))
```

2. Analizar Imagen con IA

```python
def analyze_image_with_ai(image_path, prompt="Describe esta imagen"):
    """Analiza una imagen usando Gemini Vision"""
    url = f"{BASE_URL}/ai/analyze-image"

    with open(image_path, 'rb') as f:
        files = {'file': f}
        data = {'prompt': prompt}
        response = requests.post(url, files=files, data=data)

    return response.json()

# Uso
result = analyze_image_with_ai(
    'endoscopia.jpg',
    'Analiza esta imagen médica e identifica posibles anomalías'
)
print(result['analysis'])
```

3. Chat Conversacional

```python
class GeminiChat:
    """Cliente para chat conversacional con Gemini"""

    def __init__(self, base_url=BASE_URL):
        self.base_url = base_url
        self.chat_url = f"{base_url}/ai/chat"

    def send_message(self, message, model="flash"):
        """Envía un mensaje al chat"""
        data = {
            'message': message,
            'model': model,
            'clear_history': False
        }
        response = requests.post(self.chat_url, data=data)
        return response.json()

    def get_history(self):
        """Obtiene el historial de conversación"""
        url = f"{self.base_url}/ai/chat/history"
        response = requests.get(url)
        return response.json()

    def clear_history(self):
        """Limpia el historial"""
        url = f"{self.base_url}/ai/chat/history"
        response = requests.delete(url)
        return response.json()
```

```
# Uso
chat = GeminiChat()

# Primera pregunta
response1 = chat.send_message("¿Qué es la segmentación de imágenes?")
print(response1['response'])

# Segunda pregunta (mantiene contexto)
response2 = chat.send_message("Dame un ejemplo práctico")
print(response2['response'])

# Ver historial
history = chat.get_history()
print(f"Total de mensajes: {history['total_messages']}")
```

4. Análisis de CSV

```
def analyze_csv(csv_path, question):
    """Analiza un archivo CSV con IA"""
    url = f"{BASE_URL}/ai/analyze-csv"

    with open(csv_path, 'rb') as f:
        files = {'file': f}
        data = {'question': question}
        response = requests.post(url, files=files, data=data)

    return response.json()

# Uso
result = analyze_csv(
    'datos_pacientes.csv',
    '¿Cuáles son las tendencias principales en estos datos médicos?'
)
print(result['analysis'])
```

5. Extraer Información Estructurada

```
def extract_structured_data(document_path, schema):
    """Extrae información estructurada de un documento"""
    url = f"{BASE_URL}/ai/extract-structured-data"

    with open(document_path, 'rb') as f:
        files = {'file': f}
        data = {'schema': json.dumps(schema)}
        response = requests.post(url, files=files, data=data)

    return response.json()
```

```
# Uso
schema = {
    "paciente": "Nombre del paciente",
    "edad": "Edad en años",
    "diagnostico": "Diagnóstico principal",
    "fecha": "Fecha del diagnóstico"
}

result = extract_structured_data('informe_medico.txt', schema)
print(json.dumps(result['extracted_data'], indent=2))
```

6. Traducir Texto

```
def translate_text(text, target_lang, source_lang="auto"):
    """Traduce texto a otro idioma"""
    url = f"{BASE_URL}/ai/translate"

    data = {
        'text': text,
        'target_language': target_lang,
        'source_language': source_lang
    }
    response = requests.post(url, data=data)
    return response.json()

# Uso
result = translate_text(
    "The patient shows signs of improvement",
    "español"
)
print(result['translated_text'])
```

7. Cliente Completo (Clase)

```
class EndoscopeAPI:
    """Cliente completo para la API"""

    def __init__(self, base_url="http://localhost:8000"):
        self.base_url = base_url

    def upload_file(self, file_path, file_type):
        """Sube un archivo según su tipo"""
        url = f"{self.base_url}/files/upload/{file_type}"
        with open(file_path, 'rb') as f:
            files = {'file': f}
            response = requests.post(url, files=files)
        return response.json()
```

```python
    def analyze_text(self, text, model="flash", temperature=0.7):
        """Analiza texto con IA"""
        url = f"{self.base_url}/ai/analyze-text"
        data = {
            'text': text,
            'model': model,
            'temperature': temperature
        }
        response = requests.post(url, data=data)
        return response.json()

    def analyze_sentiment(self, text, detailed=False):
        """Analiza sentimiento de un texto"""
        url = f"{self.base_url}/ai/sentiment"
        data = {
            'text': text,
            'detailed': detailed
        }
        response = requests.post(url, data=data)
        return response.json()

    def summarize(self, text, length="medium", bullet_points=False):
        """Resume un texto"""
        url = f"{self.base_url}/ai/summarize"
        data = {
            'text': text,
            'summary_length': length,
            'bullet_points': bullet_points
        }
        response = requests.post(url, data=data)
        return response.json()

    def health_check(self):
        """Verifica el estado de la API"""
        url = f"{self.base_url}/health"
        response = requests.get(url)
        return response.json()

# Uso
api = EndoscopeAPI()

# Verificar salud
health = api.health_check()
print(f"API Status: {health['status']}")

# Subir imagen
result = api.upload_file('imagen.jpg', 'image')
print(result)

# Analizar texto
analysis = api.analyze_text("¿Qué es una endoscopia?")
print(analysis['response'])
```

JavaScript / Node.js

Configuración Inicial

```
npm install axios form-data
```

```
const axios = require('axios');
const FormData = require('form-data');
const fs = require('fs');

const BASE_URL = 'http://localhost:8000';
```

1. Subir una Imagen

```
async function uploadImage(imagePath) {
    const formData = new FormData();
    formData.append('file', fs.createReadStream(imagePath));

    try {
        const response = await axios.post(
            `${BASE_URL}/files/upload/image`,
            formData,
            {
                headers: formData.getHeaders()
            }
        );
        return response.data;
    } catch (error) {
        console.error('Error:', error.response.data);
        throw error;
    }
}

// Uso
uploadImage('imagen.jpg')
    .then(result => console.log(result))
    .catch(error => console.error(error));
```

2. Analizar Imagen con IA

```
async function analyzeImageWithAI(imagePath, prompt = 'Describe esta imagen') {
    const formData = new FormData();
```

```
    formData.append('file', fs.createReadStream(imagePath));
    formData.append('prompt', prompt);

    try {
        const response = await axios.post(
            `${BASE_URL}/ai/analyze-image`,
            formData,
            {
                headers: formData.getHeaders()
            }
        );
        return response.data;
    } catch (error) {
        console.error('Error:', error.response.data);
        throw error;
    }
}

// Uso
analyzeImageWithAI('endoscopia.jpg', 'Analiza esta imagen médica')
    .then(result => console.log(result.analysis))
    .catch(error => console.error(error));
```

3. Chat Conversacional

```
class GeminiChat {
    constructor(baseUrl = BASE_URL) {
        this.baseUrl = baseUrl;
        this.chatUrl = `${baseUrl}/ai/chat`;
    }

    async sendMessage(message, model = 'flash') {
        const formData = new FormData();
        formData.append('message', message);
        formData.append('model', model);
        formData.append('clear_history', 'false');

        try {
            const response = await axios.post(
                this.chatUrl,
                formData,
                {
                    headers: formData.getHeaders()
                }
            );
            return response.data;
        } catch (error) {
            console.error('Error:', error.response.data);
            throw error;
        }
```

```
    }

    async getHistory() {
        const response = await axios.get(`${this.baseUrl}/ai/chat/history`);
        return response.data;
    }

    async clearHistory() {
        const response = await axios.delete(`${this.baseUrl}/ai/chat/history`);
        return response.data;
    }
}

// Uso
const chat = new GeminiChat();

(async () => {
    // Primera pregunta
    const response1 = await chat.sendMessage('¿Qué es la segmentación?');
    console.log(response1.response);

    // Segunda pregunta (mantiene contexto)
    const response2 = await chat.sendMessage('Dame un ejemplo');
    console.log(response2.response);

    // Ver historial
    const history = await chat.getHistory();
    console.log(`Total mensajes: ${history.total_messages}`);
})();
```

4. Cliente Completo (Clase)

```
class EndoscopeAPI {
    constructor(baseUrl = 'http://localhost:8000') {
        this.baseUrl = baseUrl;
    }

    async uploadFile(filePath, fileType) {
        const formData = new FormData();
        formData.append('file', fs.createReadStream(filePath));

        const response = await axios.post(
            `${this.baseUrl}/files/upload/${fileType}`,
            formData,
            { headers: formData.getHeaders() }
        );
        return response.data;
    }

    async analyzeText(text, model = 'flash', temperature = 0.7) {
```

```
        const formData = new FormData();
        formData.append('text', text);
        formData.append('model', model);
        formData.append('temperature', temperature.toString());

        const response = await axios.post(
            `${this.baseUrl}/ai/analyze-text`,
            formData,
            { headers: formData.getHeaders() }
        );
        return response.data;
    }

    async translateText(text, targetLanguage, sourceLanguage = 'auto') {
        const formData = new FormData();
        formData.append('text', text);
        formData.append('target_language', targetLanguage);
        formData.append('source_language', sourceLanguage);

        const response = await axios.post(
            `${this.baseUrl}/ai/translate`,
            formData,
            { headers: formData.getHeaders() }
        );
        return response.data;
    }

    async healthCheck() {
        const response = await axios.get(`${this.baseUrl}/health`);
        return response.data;
    }
}

// Uso
const api = new EndoscopeAPI();

(async () => {
    const health = await api.healthCheck();
    console.log('API Status:', health.status);

    const result = await api.analyzeText('¿Qué es una endoscopia?');
    console.log(result.response);
})();
```

---

cURL

1. Información de la API

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

2. Health Check

```
curl http://localhost:8000/health
```

3. Subir Imagen

```
curl -X POST "http://localhost:8000/files/upload/image" \
  -F "file=@imagen.jpg"
```

4. Analizar Imagen con IA

```
curl -X POST "http://localhost:8000/ai/analyze-image" \
  -F "file=@endoscopia.jpg" \
  -F "prompt=Describe esta imagen médica en detalle"
```

5. Chat

```
# Primera pregunta
curl -X POST "http://localhost:8000/ai/chat" \
  -F "message=Hola, ¿puedes ayudarme?"

# Ver historial
curl http://localhost:8000/ai/chat/history

# Limpiar historial
curl -X DELETE http://localhost:8000/ai/chat/history
```

6. Analizar Texto

```
curl -X POST "http://localhost:8000/ai/analyze-text" \
  -F "text=¿Qué es la inteligencia artificial?" \
  -F "model=flash" \
  -F "temperature=0.7"
```

7. Traducir

```
curl -X POST "http://localhost:8000/ai/translate" \
  -F "text=Hello world" \
```

```
  -F "target_language=español"
```

8. Resumir Texto

```
curl -X POST "http://localhost:8000/ai/summarize" \
  -F "text=Tu texto largo aquí..." \
  -F "summary_length=short" \
  -F "bullet_points=true"
```

9. Análisis de Sentimiento

```
curl -X POST "http://localhost:8000/ai/sentiment" \
  -F "text=Me encanta este producto, es excelente" \
  -F "detailed=true"
```

10. Subir Múltiples Archivos

```
curl -X POST "http://localhost:8000/files/upload/multiple" \
  -F "files=@imagen1.jpg" \
  -F "files=@imagen2.jpg" \
  -F "files=@documento.pdf"
```

Postman

Configuración

1. **Crear una nueva colección**: "Endoscope API"
2. **Configurar variable de entorno**:
   - Variable: `base_url`
   - Valor: `http://localhost:8000`

Ejemplos de Requests

1. GET - Health Check

```
GET {{base_url}}/health
```

2. POST - Subir Imagen

```
POST {{base_url}}/files/upload/image
Body: form-data
- Key: file (tipo: File)
- Value: Seleccionar archivo
```

3. POST - Analizar Imagen con IA

```
POST {{base_url}}/ai/analyze-image
Body: form-data
- Key: file (tipo: File)
- Key: prompt (tipo: Text) = "Describe esta imagen"
```

4. POST - Chat

```
POST {{base_url}}/ai/chat
Body: form-data
- Key: message (tipo: Text) = "Tu mensaje aquí"
- Key: model (tipo: Text) = "flash"
```

PHP

1. Subir Imagen

```php
<?php
function uploadImage($imagePath) {
    $url = 'http://localhost:8000/files/upload/image';

    $file = new CURLFile($imagePath);
    $data = ['file' => $file];

    $ch = curl_init();
    curl_setopt($ch, CURLOPT_URL, $url);
    curl_setopt($ch, CURLOPT_POST, 1);
    curl_setopt($ch, CURLOPT_POSTFIELDS, $data);
    curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);

    $response = curl_exec($ch);
    curl_close($ch);

    return json_decode($response, true);
}

// Uso
```

```php
$result = uploadImage('imagen.jpg');
print_r($result);
?>
```

2. Analizar Texto con IA

```php
<?php
function analyzeText($text, $model = 'flash') {
    $url = 'http://localhost:8000/ai/analyze-text';

    $data = [
        'text' => $text,
        'model' => $model,
        'temperature' => 0.7
    ];

    $ch = curl_init();
    curl_setopt($ch, CURLOPT_URL, $url);
    curl_setopt($ch, CURLOPT_POST, 1);
    curl_setopt($ch, CURLOPT_POSTFIELDS, http_build_query($data));
    curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);

    $response = curl_exec($ch);
    curl_close($ch);

    return json_decode($response, true);
}

// Uso
$result = analyzeText('¿Qué es una endoscopia?');
echo $result['response'];
?>
```

3. Cliente Completo

```php
<?php
class EndoscopeAPI {
    private $baseUrl;

    public function __construct($baseUrl = 'http://localhost:8000') {
        $this->baseUrl = $baseUrl;
    }

    public function uploadFile($filePath, $fileType) {
        $url = "{$this->baseUrl}/files/upload/{$fileType}";

        $file = new CURLFile($filePath);
        $data = ['file' => $file];
```

```php
        return $this->makeRequest($url, $data);
    }

    public function analyzeImage($imagePath, $prompt = 'Describe esta imagen')
{
        $url = "{$this->baseUrl}/ai/analyze-image";

        $file = new CURLFile($imagePath);
        $data = [
            'file' => $file,
            'prompt' => $prompt
        ];

        return $this->makeRequest($url, $data);
    }

    public function chat($message, $model = 'flash') {
        $url = "{$this->baseUrl}/ai/chat";

        $data = [
            'message' => $message,
            'model' => $model,
            'clear_history' => false
        ];

        return $this->makeRequest($url, $data);
    }

    public function healthCheck() {
        $url = "{$this->baseUrl}/health";

        $ch = curl_init();
        curl_setopt($ch, CURLOPT_URL, $url);
        curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);

        $response = curl_exec($ch);
        curl_close($ch);

        return json_decode($response, true);
    }

    private function makeRequest($url, $data) {
        $ch = curl_init();
        curl_setopt($ch, CURLOPT_URL, $url);
        curl_setopt($ch, CURLOPT_POST, 1);
        curl_setopt($ch, CURLOPT_POSTFIELDS, $data);
        curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);

        $response = curl_exec($ch);
        curl_close($ch);

        return json_decode($response, true);
```

```
    }
}

// Uso
$api = new EndoscopeAPI();

$health = $api->healthCheck();
echo "API Status: " . $health['status'] . "\n";

$result = $api->chat('¿Qué es una endoscopia?');
echo $result['response'] . "\n";
?>
```

Java

1. Dependencias (Maven)

```
<dependency>
    <groupId>com.squareup.okhttp3</groupId>
    <artifactId>okhttp</artifactId>
    <version>4.11.0</version>
</dependency>
<dependency>
    <groupId>com.google.code.gson</groupId>
    <artifactId>gson</artifactId>
    <version>2.10.1</version>
</dependency>
```

2. Cliente Java

```
import okhttp3.*;
import com.google.gson.Gson;
import java.io.File;
import java.io.IOException;
import java.util.Map;

public class EndoscopeAPI {
    private final String baseUrl;
    private final OkHttpClient client;
    private final Gson gson;

    public EndoscopeAPI(String baseUrl) {
        this.baseUrl = baseUrl;
        this.client = new OkHttpClient();
        this.gson = new Gson();
    }
```

```
    public Map<String, Object> uploadImage(String imagePath) throws IOException
{
        File file = new File(imagePath);

        RequestBody requestBody = new MultipartBody.Builder()
                .setType(MultipartBody.FORM)
                .addFormDataPart("file", file.getName(),
                        RequestBody.create(file, MediaType.parse("image/*")))
                .build();

        Request request = new Request.Builder()
                .url(baseUrl + "/files/upload/image")
                .post(requestBody)
                .build();

        try (Response response = client.newCall(request).execute()) {
            String jsonData = response.body().string();
            return gson.fromJson(jsonData, Map.class);
        }
    }

    public Map<String, Object> analyzeText(String text, String model) throws
IOException {
        RequestBody requestBody = new MultipartBody.Builder()
                .setType(MultipartBody.FORM)
                .addFormDataPart("text", text)
                .addFormDataPart("model", model)
                .addFormDataPart("temperature", "0.7")
                .build();

        Request request = new Request.Builder()
                .url(baseUrl + "/ai/analyze-text")
                .post(requestBody)
                .build();

        try (Response response = client.newCall(request).execute()) {
            String jsonData = response.body().string();
            return gson.fromJson(jsonData, Map.class);
        }
    }

    public Map<String, Object> chat(String message) throws IOException {
        RequestBody requestBody = new MultipartBody.Builder()
                .setType(MultipartBody.FORM)
                .addFormDataPart("message", message)
                .addFormDataPart("model", "flash")
                .build();

        Request request = new Request.Builder()
                .url(baseUrl + "/ai/chat")
                .post(requestBody)
                .build();
```

```
        try (Response response = client.newCall(request).execute()) {
            String jsonData = response.body().string();
            return gson.fromJson(jsonData, Map.class);
        }
    }

    public static void main(String[] args) {
        EndoscopeAPI api = new EndoscopeAPI("http://localhost:8000");

        try {
            // Analizar texto
            Map<String, Object> result = api.analyzeText(
                "¿Qué es una endoscopia?",
                "flash"
            );
            System.out.println(result.get("response"));

            // Chat
            Map<String, Object> chatResult = api.chat("Hola, ¿cómo estás?");
            System.out.println(chatResult.get("response"));

        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

C#

1. Cliente C#

```
using System;
using System.IO;
using System.Net.Http;
using System.Threading.Tasks;
using Newtonsoft.Json;

public class EndoscopeAPI
{
    private readonly string baseUrl;
    private readonly HttpClient client;

    public EndoscopeAPI(string baseUrl = "http://localhost:8000")
    {
        this.baseUrl = baseUrl;
        this.client = new HttpClient();
    }

    public async Task<dynamic> UploadImage(string imagePath)
```

```
    {
        using (var content = new MultipartFormDataContent())
        {
            var fileContent = new
ByteArrayContent(File.ReadAllBytes(imagePath));
            fileContent.Headers.ContentType = new
System.Net.Http.Headers.MediaTypeHeaderValue("image/jpeg");
            content.Add(fileContent, "file", Path.GetFileName(imagePath));

            var response = await client.PostAsync($"
{baseUrl}/files/upload/image", content);
            var jsonString = await response.Content.ReadAsStringAsync();
            return JsonConvert.DeserializeObject(jsonString);
        }
    }

    public async Task<dynamic> AnalyzeText(string text, string model = "flash")
    {
        using (var content = new MultipartFormDataContent())
        {
            content.Add(new StringContent(text), "text");
            content.Add(new StringContent(model), "model");
            content.Add(new StringContent("0.7"), "temperature");

            var response = await client.PostAsync($"{baseUrl}/ai/analyze-text",
content);
            var jsonString = await response.Content.ReadAsStringAsync();
            return JsonConvert.DeserializeObject(jsonString);
        }
    }

    public async Task<dynamic> Chat(string message)
    {
        using (var content = new MultipartFormDataContent())
        {
            content.Add(new StringContent(message), "message");
            content.Add(new StringContent("flash"), "model");

            var response = await client.PostAsync($"{baseUrl}/ai/chat",
content);
            var jsonString = await response.Content.ReadAsStringAsync();
            return JsonConvert.DeserializeObject(jsonString);
        }
    }

    public async Task<dynamic> HealthCheck()
    {
        var response = await client.GetAsync($"{baseUrl}/health");
        var jsonString = await response.Content.ReadAsStringAsync();
        return JsonConvert.DeserializeObject(jsonString);
    }
}
```

```
// Uso
class Program
{
    static async Task Main(string[] args)
    {
        var api = new EndoscopeAPI();

        // Health check
        var health = await api.HealthCheck();
        Console.WriteLine($"API Status: {health.status}");

        // Analizar texto
        var result = await api.AnalyzeText("¿Qué es una endoscopia?");
        Console.WriteLine(result.response);

        // Chat
        var chatResult = await api.Chat("Hola, ¿cómo estás?");
        Console.WriteLine(chatResult.response);
    }
}
```

🔗 Enlaces Útiles

- Documentación API (Swagger)
- Documentación API (ReDoc)
- Repositorio GitHub

📝 Notas

- Todos los ejemplos asumen que la API está corriendo en `http://localhost:8000`
- Reemplaza las rutas de archivos con las rutas correctas en tu sistema
- Asegúrate de que el servicio de Gemini esté configurado correctamente

---

¿Tienes más preguntas? Consulta el README principal o la documentación interactiva.