📚 Exemplos de Uso da API

Usando cURL

1. Health Check

```
curl -X GET http://localhost:8000/health
```

2. Classificar Desvio - Apenas Descrição

```
curl -X POST http://localhost:8000/api/classify-deviation \
  -H "Content-Type: application/json" \
  -d '{
    "local": "Setor 3 - Área de Produção",
    "description": "Identificado equipamento com vazamento de óleo, risco de acidente grave"
  }'
```

3. Classificar Desvio - Com Áudio (Base64)

```
# Primeiro, converta o áudio para base64
AUDIO_BASE64=$(base64 -w 0 audio.mp3)

# Envie a requisição
curl -X POST http://localhost:8000/api/classify-deviation \
  -H "Content-Type: application/json" \
  -d "{
    \"local\": \"Setor 2\",
    \"audio\": \"$AUDIO_BASE64\"
  }"
```

4. Classificar Desvio - Upload de Arquivo

```
curl -X POST http://localhost:8000/api/classify-deviation \
  -F "local=Setor 1" \
  -F "description=Problema de segurança" \
  -F "audio=@audio.mp3"
```

Usando Python (requests)

```python
import requests
import base64

# URL da API
BASE_URL = "http://localhost:8000"

# 1. Health Check
def check_health():
    response = requests.get(f"{BASE_URL}/health")
    print(response.json())

# 2. Classificar com descrição
def classify_with_description():
    payload = {
        "local": "Setor 3 - Produção",
        "description": "Equipamento apresentando ruído anormal e superaquecimento"
    }

    response = requests.post(
        f"{BASE_URL}/api/classify-deviation",
        json=payload
    )

    print(f"Status: {response.status_code}")
    print(f"Resultado: {response.json()}")

# 3. Classificar com áudio
def classify_with_audio():
    # Lê o arquivo de áudio
    with open("audio.mp3", "rb") as f:
        audio_bytes = f.read()

    # Codifica em base64
    audio_base64 = base64.b64encode(audio_bytes).decode('utf-8')

    payload = {
        "local": "Área Externa",
        "description": "Condição insegura reportada",
        "audio": audio_base64
    }

    response = requests.post(
        f"{BASE_URL}/api/classify-deviation",
        json=payload
    )

    print(f"Status: {response.status_code}")
    print(f"Resultado: {response.json()}")

# 4. Upload de arquivo multipart
def classify_with_file_upload():
    files = {
        'audio': ('audio.mp3', open('audio.mp3', 'rb'), 'audio/mpeg')
    }

    data = {
        'local': 'Setor 5',
        'description': 'Reportado via áudio'
    }

    response = requests.post(
        f"{BASE_URL}/api/classify-deviation",
```

```
        files=files,
        data=data
    )

    print(f"Status: {response.status_code}")
    print(f"Resultado: {response.json()}")

# Executa exemplos
if __name__ == "__main__":
    check_health()
    classify_with_description()
```

Usando JavaScript (Fetch API)

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

// 1. Health Check
async function checkHealth() {
  const response = await fetch(`${BASE_URL}/health`);
  const data = await response.json();
  console.log('Health:', data);
}

// 2. Classificar com descrição
async function classifyWithDescription() {
  const payload = {
    local: 'Setor 3',
    description: 'Equipamento apresentando problema'
  };

  const response = await fetch(`${BASE_URL}/api/classify-deviation`, {
    method: 'POST',
    headers: {
      'Content-Type': 'application/json'
    },
    body: JSON.stringify(payload)
  });

  const result = await response.json();
  console.log('Classificação:', result);
}

// 3. Classificar com áudio (arquivo)
async function classifyWithAudio(audioFile) {
  const formData = new FormData();
  formData.append('local', 'Setor 1');
  formData.append('description', 'Reportado via áudio');
  formData.append('audio', audioFile);

  const response = await fetch(`${BASE_URL}/api/classify-deviation`, {
    method: 'POST',
    body: formData
  });

  const result = await response.json();
  console.log('Classificação:', result);
}

// Uso com input file
document.getElementById('audioInput').addEventListener('change', (e) => {
  const file = e.target.files[0];
  if (file) {
    classifyWithAudio(file);
  }
});
```

Exemplos de Resposta

Sucesso (200 OK)


```
{
  "gravidade": 0.75,
  "urgencia": 0.85,
  "tendencia": 0.60,
  "tipo": "seguranca",
  "direcionamento": "supervisao_urgente",
  "categoria": "alto"
}
```


Erro de Validação (400 Bad Request)


```
{
  "error": "InvalidInputError",
  "message": "Campo 'local' é obrigatório",
  "details": {}
}
```


Erro de Transcrição (400 Bad Request)


```
{
  "error": "TranscriptionError",
  "message": "Erro ao transcrever áudio",
  "details": {
    "error": "Invalid audio format"
  }
}
```


Erro Interno (500 Internal Server Error)


```
{
  "error": "InternalServerError",
  "message": "Erro interno do servidor",
  "details": {
    "error": "Unexpected error occurred"
  }
}
```


Cenários de Teste

Cenário 1: Emergência Crítica


```
{
  "local": "Área de Produção Principal",
  "description": "URGENTE: Princípio de incêndio detectado próximo ao tanque de com-
bustível. Risco iminente de explosão. Evacuação em andamento."
}
```


Resposta Esperada:

```
{
  "gravidade": 0.95,
  "urgencia": 1.0,
  "tendencia": 0.8,
  "tipo": "seguranca",
  "direcionamento": "emergencia_imediata",
  "categoria": "critico"
}
```


Cenário 2: Problema de Manutenção


```
{
  "local": "Setor 2 - Linha B",
  "description": "Máquina apresentando vibração excessiva e ruído anormal. Necessita inspeção."
}
```


Resposta Esperada:


```
{
  "gravidade": 0.45,
  "urgencia": 0.50,
  "tendencia": 0.55,
  "tipo": "manutencao",
  "direcionamento": "manutencao",
  "categoria": "medio"
}
```


Cenário 3: Observação Ambiental


```
{
  "local": "Área Externa Nordeste",
  "description": "Observado pequeno vazamento de água no sistema de refrigeração. Sem urgência imediata."
}
```


Resposta Esperada:


```
{
  "gravidade": 0.25,
  "urgencia": 0.30,
  "tendencia": 0.40,
  "tipo": "ambiental",
  "direcionamento": "meio_ambiente",
  "categoria": "baixo"
}
```


Dicas de Integração

1. **Retry Logic**: Implemente retry com backoff exponencial
2. **Timeout**: Configure timeout adequado (30-60s para transcrição)
3. **Validação**: Sempre valide o status code antes de processar
4. **Áudio**: Prefira MP3 com taxa de 16kHz para melhor performance

5. **Tamanho**: Limite uploads a 10MB para melhor experiência
6. **Cache**: Considere cachear resultados de classificações idênticas
7. **Monitoramento**: Implemente logging de todas as requisições