Configuração de APIs - DMCA Guard Platform

Este guia detalha como configurar e integrar as APIs externas necessárias para o funcionamento completo da plataforma DMCA Guard.

OpenAI API

A OpenAI API é utilizada para processamento inteligente de conteúdo, análise de similaridade e geração de notificações DMCA personalizadas.

1. Criação da Conta e API Key

Passo a Passo

1. **Acesse**: https://platform.openai.com (https://platform.openai.com)
2. **Registre-se** ou faça login com sua conta
3. **Navegue** para "API Keys" no menu lateral
4. **Clique** em "Create new secret key"
5. **Nomeie** a chave (ex: "DMCA Guard Production")
6. **Copie** a chave gerada (começa com `sk-`)

Configuração de Billing

```
# Acesse: https://platform.openai.com/account/billing
# Configure um método de pagamento
# Defina limites de uso para controlar custos
```

2. Configuração na Aplicação

Variáveis de Ambiente

```
# .env.local
OPENAI_API_KEY="sk-your-openai-api-key-here"
OPENAI_ORG_ID="org-your-organization-id" # Opcional
OPENAI_MODEL="gpt-4" # ou gpt-3.5-turbo para economia
OPENAI_MAX_TOKENS="2000"
OPENAI_TEMPERATURE="0.3"
```

Configuração Avançada

```
// lib/openai.js
import OpenAI from 'openai';

const openai = new OpenAI({
  apiKey: process.env.OPENAI_API_KEY,
  organization: process.env.OPENAI_ORG_ID,
});

export const openaiConfig = {
  model: process.env.OPENAI_MODEL || 'gpt-3.5-turbo',
  maxTokens: parseInt(process.env.OPENAI_MAX_TOKENS) || 1500,
  temperature: parseFloat(process.env.OPENAI_TEMPERATURE) || 0.3,
  timeout: 30000, // 30 segundos
};

export default openai;
```

3. Funcionalidades Implementadas

Análise de Conteúdo

```
// lib/ai/content-analysis.js
export async function analyzeContent(content) {
  const prompt = `
    Analise o seguinte conteúdo e determine:
    1. Se é conteúdo adulto
    2. Nível de similaridade com conteúdo original
    3. Possíveis violações de direitos autorais

    Conteúdo: ${content}

    Responda em formato JSON.
  `;

  const response = await openai.chat.completions.create({
    model: openaiConfig.model,
    messages: [{ role: 'user', content: prompt }],
    max_tokens: openaiConfig.maxTokens,
    temperature: openaiConfig.temperature,
  });

  return JSON.parse(response.choices[0].message.content);
}
```

Geração de Notificações DMCA

```
// lib/ai/dmca-generator.js
export async function generateDMCANotice(contentInfo) {
  const prompt = `
    Gere uma notificação DMCA profissional em português para:

    Plataforma: ${contentInfo.platform}
    URL do conteúdo infrator: ${contentInfo.infringingUrl}
    Descrição do conteúdo original: ${contentInfo.originalDescription}
    Nome da criadora: ${contentInfo.creatorName}

    Use o template legal apropriado e linguagem formal.
  `;

  const response = await openai.chat.completions.create({
    model: openaiConfig.model,
    messages: [
      {
        role: 'system',
        content: 'Você é um assistente jurídico especializado em direitos autorais e
DMCA.'
      },
      { role: 'user', content: prompt }
    ],
    max_tokens: 2000,
    temperature: 0.2,
  });

  return response.choices[0].message.content;
}
```

4. Monitoramento de Uso

Tracking de Tokens

```
// lib/ai/usage-tracker.js
export class OpenAIUsageTracker {
  static async trackUsage(response, operation) {
    const usage = {
      operation,
      promptTokens: response.usage.prompt_tokens,
      completionTokens: response.usage.completion_tokens,
      totalTokens: response.usage.total_tokens,
      model: response.model,
      timestamp: new Date(),
    };

    // Salvar no banco de dados
    await prisma.aiUsage.create({ data: usage });

    return usage;
  }

  static async getMonthlyUsage() {
    const startOfMonth = new Date();
    startOfMonth.setDate(1);
    startOfMonth.setHours(0, 0, 0, 0);

    return await prisma.aiUsage.aggregate({
      where: {
        timestamp: { gte: startOfMonth }
      },
      _sum: {
        totalTokens: true,
        promptTokens: true,
        completionTokens: true,
      }
    });
  }
}
```

5. Otimização de Custos

Estratégias de Economia

```
// lib/ai/cost-optimization.js
export const costOptimization = {
  // Usar modelo mais barato para tarefas simples
  getModelForTask: (taskComplexity) => {
    switch (taskComplexity) {
      case 'simple':
        return 'gpt-3.5-turbo';
      case 'medium':
        return 'gpt-3.5-turbo-16k';
      case 'complex':
        return 'gpt-4';
      default:
        return 'gpt-3.5-turbo';
    }
  },

  // Cache de respostas para evitar chamadas repetidas
  cacheKey: (prompt) => {
    return crypto.createHash('md5').update(prompt).digest('hex');
  },

  // Rate limiting para controlar uso
  rateLimiter: new Map(),
};
```

SendGrid API

SendGrid é usado para envio de emails transacionais, notificações DMCA e comunicação com usuários.

1. Configuração da Conta

Criação da Conta

1. **Acesse**: https://sendgrid.com (https://sendgrid.com)
2. **Registre-se** para conta gratuita (100 emails/dia)
3. **Verifique** seu email
4. **Complete** o processo de verificação

Verificação de Domínio

```
# 1. Acesse: Settings > Sender Authentication
# 2. Clique em "Authenticate Your Domain"
# 3. Adicione os registros DNS fornecidos:

# Exemplo de registros DNS
CNAME s1._domainkey.yourdomain.com -> s1.domainkey.u123456.wl.sendgrid.net
CNAME s2._domainkey.yourdomain.com -> s2.domainkey.u123456.wl.sendgrid.net
CNAME em123.yourdomain.com -> u123456.wl.sendgrid.net
```

2. Criação da API Key

Passo a Passo

1. **Navegue** para “Settings > API Keys”
2. **Clique** em “Create API Key”
3. **Escolha** “Restricted Access”
4. **Configure** permissões:
   - Mail Send: Full Access
   - Template Engine: Read Access
   - Suppressions: Read Access

```
{
  "name": "DMCA Guard Production",
  "scopes": [
    "mail.send",
    "templates.read",
    "suppressions.read"
  ]
}
```

3. Configuração na Aplicação

Variáveis de Ambiente

```
# .env.local
SENDGRID_API_KEY="SG.your-sendgrid-api-key-here"
SENDGRID_FROM_EMAIL="noreply@yourdomain.com"
SENDGRID_FROM_NAME="DMCA Guard"
SENDGRID_REPLY_TO="suporte@yourdomain.com"

# Templates (opcional)
SENDGRID_TEMPLATE_WELCOME="d-welcome-template-id"
SENDGRID_TEMPLATE_DMCA="d-dmca-template-id"
SENDGRID_TEMPLATE_NOTIFICATION="d-notification-template-id"
```

Configuração do Cliente

```
// lib/sendgrid.js
import sgMail from '@sendgrid/mail';

sgMail.setApiKey(process.env.SENDGRID_API_KEY);

export const sendgridConfig = {
  from: {
    email: process.env.SENDGRID_FROM_EMAIL,
    name: process.env.SENDGRID_FROM_NAME,
  },
  replyTo: process.env.SENDGRID_REPLY_TO,
  templates: {
    welcome: process.env.SENDGRID_TEMPLATE_WELCOME,
    dmca: process.env.SENDGRID_TEMPLATE_DMCA,
    notification: process.env.SENDGRID_TEMPLATE_NOTIFICATION,
  },
};

export default sgMail;
```

4. Templates de Email

Template de Boas-vindas

```
<!-- SendGrid Template: Welcome -->
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>Bem-vinda à DMCA Guard</title>
</head>
<body>
    <div style="max-width: 600px; margin: 0 auto; font-family: Arial, sans-serif;">
        <h1>Bem-vinda à DMCA Guard!  </h1>

        <p>Olá {{firstName}},</p>

        <p>Sua conta foi criada com sucesso! Agora você pode proteger seu conteúdo com
nossa plataforma.</p>

        <div style="background: #f8f9fa; padding: 20px; border-radius: 8px; margin:
20px 0;">
            <h3>Próximos passos:</h3>
            <ol>
                <li>Configure seu primeiro perfil de marca</li>
                <li>Inicie uma sessão de monitoramento</li>
                <li>Explore nossos recursos de proteção</li>
            </ol>
        </div>

        <a href="{{dashboardUrl}}" style="background: #007bff; color: white; padding:
12px 24px; text-decoration: none; border-radius: 4px; display: inline-block;">
            Acessar Dashboard
        </a>

        <p>Se precisar de ajuda, nossa equipe está sempre disponível!</p>

        <p>Atenciosamente,<br>Equipe DMCA Guard</p>
    </div>
</body>
</html>
```

Template de Notificação DMCA

```
<!-- SendGrid Template: DMCA Notice -->
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>Notificação DMCA - {{platformName}}</title>
</head>
<body>
    <div style="max-width: 600px; margin: 0 auto; font-family: Arial, sans-serif;">
        <h1>Notificação DMCA</h1>

        <p><strong>Para:</strong> {{platformEmail}}</p>
        <p><strong>Assunto:</strong> Solicitação de Remoção de Conteúdo - DMCA</p>

        <div style="background: #fff3cd; border: 1px solid #ffeaa7; padding: 15px;
border-radius: 4px; margin: 20px 0;">
            <p><strong>⚠ Esta é uma notificação oficial DMCA</strong></p>
        </div>

        <div style="background: #f8f9fa; padding: 20px; border-radius: 8px;">
            {{dmcaContent}}
        </div>

        <p><strong>URLs do conteúdo infrator:</strong></p>
        <ul>
            {{#each infringingUrls}}
            <li><a href="{{this}}">{{this}}</a></li>
            {{/each}}
        </ul>

        <p>Aguardamos a remoção do conteúdo em até 72 horas.</p>

        <p>Atenciosamente,<br>{{creatorName}}<br>{{creatorEmail}}</p>
    </div>
</body>
</html>
```

5. Implementação de Envio

Serviço de Email

```
// lib/email/email-service.js
import sgMail, { sendgridConfig } from '../sendgrid.js';

export class EmailService {
  static async sendWelcomeEmail(user) {
    const msg = {
      to: user.email,
      from: sendgridConfig.from,
      templateId: sendgridConfig.templates.welcome,
      dynamicTemplateData: {
        firstName: user.firstName,
        dashboardUrl: `${process.env.APP_URL}/dashboard`,
      },
    };

    try {
      await sgMail.send(msg);
      console.log(`Welcome email sent to ${user.email}`);
    } catch (error) {
      console.error('Error sending welcome email:', error);
      throw error;
    }
  }

  static async sendDMCANotice(dmcaData) {
    const msg = {
      to: dmcaData.platformEmail,
      from: sendgridConfig.from,
      replyTo: dmcaData.creatorEmail,
      templateId: sendgridConfig.templates.dmca,
      dynamicTemplateData: {
        platformName: dmcaData.platformName,
        platformEmail: dmcaData.platformEmail,
        dmcaContent: dmcaData.content,
        infringingUrls: dmcaData.urls,
        creatorName: dmcaData.creatorName,
        creatorEmail: dmcaData.creatorEmail,
      },
    };

    try {
      const response = await sgMail.send(msg);

      // Log para auditoria
      await this.logEmailSent({
        type: 'dmca_notice',
        to: dmcaData.platformEmail,
        messageId: response[0].headers['x-message-id'],
        status: 'sent',
      });

      return response;
    } catch (error) {
      console.error('Error sending DMCA notice:', error);

      await this.logEmailSent({
        type: 'dmca_notice',
        to: dmcaData.platformEmail,
```

```
        status: 'failed',
        error: error.message,
      });

      throw error;
    }
  }

  static async logEmailSent(logData) {
    await prisma.emailLog.create({
      data: {
        ...logData,
        timestamp: new Date(),
      },
    });
  }
}
```

6. Webhooks do SendGrid

Configuração de Webhooks

```
// app/api/webhooks/sendgrid/route.js
import { NextResponse } from 'next/server';
import crypto from 'crypto';

export async function POST(request) {
  const body = await request.text();
  const signature = request.headers.get('x-twilio-email-event-webhook-signature');

  // Verificar assinatura
  const expectedSignature = crypto
    .createHmac('sha256', process.env.SENDGRID_WEBHOOK_SECRET)
    .update(body)
    .digest('base64');

  if (signature !== expectedSignature) {
    return NextResponse.json({ error: 'Invalid signature' }, { status: 401 });
  }

  const events = JSON.parse(body);

  for (const event of events) {
    await processEmailEvent(event);
  }

  return NextResponse.json({ success: true });
}

async function processEmailEvent(event) {
  const eventData = {
    messageId: event.sg_message_id,
    event: event.event,
    email: event.email,
    timestamp: new Date(event.timestamp * 1000),
    reason: event.reason,
    url: event.url,
  };

  await prisma.emailEvent.create({ data: eventData });

  // Processar eventos específicos
  switch (event.event) {
    case 'delivered':
      await handleEmailDelivered(event);
      break;
    case 'bounce':
      await handleEmailBounce(event);
      break;
    case 'click':
      await handleEmailClick(event);
      break;
  }
}
```

7. Monitoramento e Analytics

Dashboard de Email

```
// lib/email/analytics.js
export class EmailAnalytics {
  static async getEmailStats(dateRange) {
    const stats = await prisma.emailEvent.groupBy({
      by: ['event'],
      where: {
        timestamp: {
          gte: dateRange.start,
          lte: dateRange.end,
        },
      },
      _count: {
        event: true,
      },
    });

    return {
      sent: stats.find(s => s.event === 'delivered')?._count.event || 0,
      opened: stats.find(s => s.event === 'open')?._count.event || 0,
      clicked: stats.find(s => s.event === 'click')?._count.event || 0,
      bounced: stats.find(s => s.event === 'bounce')?._count.event || 0,
    };
  }

  static async getDMCAResponseRate() {
    const dmcaEmails = await prisma.emailLog.findMany({
      where: { type: 'dmca_notice' },
      include: {
        events: true,
      },
    });

    const totalSent = dmcaEmails.length;
    const opened = dmcaEmails.filter(email =>
      email.events.some(event => event.event === 'open')
    ).length;

    return {
      totalSent,
      opened,
      openRate: totalSent > 0 ? (opened / totalSent) * 100 : 0,
    };
  }
}
```

Configurações Adicionais

Rate Limiting

```
// lib/api/rate-limiter.js
export const apiRateLimiter = {
  openai: {
    requests: 60, // por minuto
    tokens: 150000, // por minuto
  },
  sendgrid: {
    requests: 600, // por minuto
    emails: 100, // por dia (plano gratuito)
  },
};
```

Fallbacks e Redundância

```
// lib/api/fallbacks.js
export class APIFallbacks {
  static async sendEmailWithFallback(emailData) {
    try {
      return await EmailService.sendEmail(emailData);
    } catch (error) {
      console.error('SendGrid failed, trying fallback:', error);

      // Fallback para SMTP direto
      return await SMTPService.sendEmail(emailData);
    }
  }

  static async generateContentWithFallback(prompt) {
    try {
      return await OpenAIService.generate(prompt);
    } catch (error) {
      console.error('OpenAI failed, using template:', error);

      // Fallback para template estático
      return TemplateService.generateFromTemplate(prompt);
    }
  }
}
```

Testes das APIs

Testes de Integração

```
// tests/api/openai.test.js
describe('OpenAI Integration', () => {
  test('should generate DMCA notice', async () => {
    const contentInfo = {
      platform: 'Test Platform',
      infringingUrl: 'https://example.com/content',
      originalDescription: 'Test content',
      creatorName: 'Test Creator',
    };

    const notice = await generateDMCANotice(contentInfo);

    expect(notice).toContain('DMCA');
    expect(notice).toContain(contentInfo.platform);
    expect(notice).toContain(contentInfo.creatorName);
  });
});

// tests/api/sendgrid.test.js
describe('SendGrid Integration', () => {
  test('should send welcome email', async () => {
    const user = {
      email: 'test@example.com',
      firstName: 'Test',
    };

    await expect(EmailService.sendWelcomeEmail(user))
      .resolves.not.toThrow();
  });
});
```

Scripts de Teste

```
#!/bin/bash
# scripts/test_apis.sh

echo "  Testando APIs..."

# Testar OpenAI
echo "Testing OpenAI API..."
curl -H "Authorization: Bearer $OPENAI_API_KEY" \
     -H "Content-Type: application/json" \
     -d '{"model":"gpt-3.5-turbo","messages":
[{"role":"user","content":"Test"}],"max_tokens":5}' \
     https://api.openai.com/v1/chat/completions

# Testar SendGrid
echo "Testing SendGrid API..."
curl -X POST https://api.sendgrid.com/v3/mail/send \
     -H "Authorization: Bearer $SENDGRID_API_KEY" \
     -H "Content-Type: application/json" \
     -d '{
       "personalizations": [{"to": [{"email": "test@example.com"}]}],
       "from": {"email": "test@yourdomain.com"},
       "subject": "Test",
       "content": [{"type": "text/plain", "value": "Test"}]
     }'
```

Monitoramento de Custos

Dashboard de Custos

```jsx
// components/admin/CostDashboard.jsx
export function CostDashboard() {
  const [costs, setCosts] = useState({
    openai: { current: 0, limit: 100 },
    sendgrid: { current: 0, limit: 50 },
  });

  useEffect(() => {
    fetchCostData();
  }, []);

  return (
    <div className="grid grid-cols-2 gap-4">
      <CostCard
        title="OpenAI"
        current={costs.openai.current}
        limit={costs.openai.limit}
        unit="USD"
      />
      <CostCard
        title="SendGrid"
        current={costs.sendgrid.current}
        limit={costs.sendgrid.limit}
        unit="emails"
      />
    </div>
  );
}
```

Próximos Passos: Após configurar as APIs, consulte o Guia de Deploy (deploy.md) para colocar sua aplicação em produção.