

# Guia de Implementação: Integração Perplexity e Remoção do Sistema de Scraping

## Contexto Importante

**Problema:** O sistema atual usa web scraping para coletar conteúdo, mas a visão do produto é usar Perplexity API para buscar informações recentes (últimos 7 dias). Precisamos substituir completamente o sistema de scraping pelo Perplexity.

**Objetivo:** Implementar busca em tempo real com Perplexity e remover com segurança todo código de scraping.

## Pré-requisitos

Antes de começar, você precisa:


1. Acesso ao repositório
2. Chave da API Perplexity (solicitar ao time)
3. Node.js 18+ instalado
4. Entender o fluxo básico: Usuário → Gera Ideias → Busca Info → Cria Conteúdo

## Parte 1: Entendendo o Código Atual (O que vamos remover)

### Arquivos do Sistema de Scraping (SERÃO REMOVIDOS)

```
server/  
├─ scraping.ts           # 500+ linhas – REMOVE  
├─ content-pipeline.ts  # Usa scraping – REFATORAR  
├─ profile-scraping-sync.ts # REMOVE  
└─ routes.ts           # Tem rotas de scraping – LIMPAR
```

### Código Problemático Exemplo:

```
typescript  
  
//  ATUAL (content-pipeline.ts) – Isso será removido  
async generateContentWithScraping() {  
  const scrapedData = await this.scrapeFreshContent(params.expertId);  
  // Scraping demorado e desnecessário  
}
```

## Parte 2: Implementação Passo a Passo

### Passo 1: Configurar Ambiente (.env)

bash

*# Adicione estas variáveis*

PERPLEXITY\_API\_KEY=plx-xxxxx *# Solicitar chave ao time*

PERPLEXITY\_ENABLED=true

USE\_LEGACY\_SCRAPING=true *# Começamos com true, depois false*

## Passo 2: Criar Novo Pipeline (content-pipeline-v2.ts)

Crie o arquivo `server/content-pipeline-v2.ts`:



```

import { perplexityService } from './services/perplexity';
import { storage } from './storage';
import { generateContentIdeas } from './anthropic';

export class ContentPipelineV2 {
  /**
   * Gera conteúdo usando Perplexity (busca em tempo real)
   * ao invés de scraping
   */
  async generateContent(params: {
    topicId: number;
    platform: string;
    expertId: number;
  }) {
    console.log('[NEW-PIPELINE] Iniciando geração com Perplexity');

    try {
      // 1. Buscar dados do banco
      const topic = await storage.getTopic(params.topicId);
      const profile = await storage.getExpertProfile(params.expertId);

      if (!topic || !profile) {
        throw new Error('Tópico ou perfil não encontrado');
      }

      // 2. Montar busca para Perplexity
      const searchQuery = `
        Encontre conteúdo recente (últimos 7 dias) sobre "${topic.title}"
        para publicar no ${params.platform}.
        Foco em: ${topic.description}
        Área: ${profile.primaryExpertise}
      `;

      console.log('[NEW-PIPELINE] Buscando com Perplexity...');

      // 3. Buscar informações recentes
      const searchResult = await perplexityService.search(searchQuery, {
        recency: 'week', // Últimos 7 dias
        maxResults: 5
      });

      // 4. Gerar ideias com Claude usando o contexto
      const ideas = await generateContentIdeas({
        topic: topic.title,
        description: topic.description || '',
        platform: params.platform,
      });
    } catch (error) {
      console.error('Erro ao gerar conteúdo:', error);
    }
  }
}

```

```

    viewpoints: [], // Simplificado por agora
    expertiseKeywords: profile.expertiseKeywords || [],
    voiceTone: profile.voiceTone || [],
    // Passar o contexto do Perplexity
    additionalContext: searchResult.content,
    realSources: searchResult.sources
  });

  // 5. Salvar e retornar
  const savedIdeas = [];
  for (const idea of ideas) {
    const saved = await storage.createContentIdea({
      topicId: params.topicId,
      platform: params.platform,
      title: idea.title,
      description: idea.description,
      format: idea.format,
      keyPoints: idea.keyPoints,
      sources: searchResult.sources.slice(0, 3) // Máximo 3 fontes
    });
    savedIdeas.push(saved);
  }

  console.log('[NEW-PIPELINE] Sucesso! Ideias geradas:', savedIdeas.length);

  return {
    ideas: savedIdeas,
    sourcesUsed: searchResult.sources,
    timestamp: new Date()
  };
} catch (error) {
  console.error('[NEW-PIPELINE] Erro:', error);
  throw error;
}
}

export const contentPipelineV2 = new ContentPipelineV2();

```

### Passo 3: Modificar Rotas com Switch Seguro

Edite `server/routes.ts`:



```

// Importe o novo pipeline
import { contentPipelineV2 } from './content-pipeline-v2';
import { contentPipeline } from './content-pipeline'; // Antigo

// Modifique a rota de geração de conteúdo
app.post('/api/generate-content-ideas', async (req, res) => {
  try {
    const { topicId, platform, expertId } = req.body;

    // SWITCH SEGURO: Usa variável de ambiente
    const useLegacy = process.env.USE_LEGACY_SCRAPING === 'true';

    if (useLegacy) {
      console.log('[ROUTES] Usando pipeline ANTIGO (scraping)');
      // Código antigo continua funcionando
      const result = await contentPipeline.generateContentWithScraping({
        topicId,
        platform,
        expertId
      });

      return res.json({
        ...result,
        metadata: { engine: 'legacy-scraping' }
      });
    } else {
      console.log('[ROUTES] Usando pipeline NOVO (Perplexity)');
      // Código novo
      const result = await contentPipelineV2.generateContent({
        topicId,
        platform,
        expertId
      });

      return res.json({
        ...result,
        metadata: { engine: 'perplexity-realtime' }
      });
    }
  } catch (error) {
    console.error('[ROUTES] Erro:', error);
    res.status(500).json({
      error: error.message,
      // Informação útil para debug

```

```

        pipeline: process.env.USE_LEGACY_SCRAPING === 'true' ? 'legacy' : 'new'
    });
}
});

```

## Passo 4: Atualizar Anthropic para Aceitar Contexto

Edite `server/anthropic.ts`:

typescript

```

// Adicione estes parâmetros à interface
interface ContentIdeaGenerationParams {
    // ... parâmetros existentes ...
    additionalContext?: string; // NOVO
    realSources?: string[];      // NOVO
}

export async function generateContentIdeas(params: ContentIdeaGenerationParams) {
    // Modifique o prompt para incluir contexto se disponível
    const systemPrompt = `You are a content strategist.
    ${params.additionalContext ? `
    RECENT INFORMATION CONTEXT:
    ${params.additionalContext}

    Use this recent information to generate relevant content ideas.` : ''}

    Generate content ideas for ${params.platform}.`;

    // No prompt do usuário, mencione as fontes reais
    const userPrompt = `
    Topic: ${params.topic}
    Platform: ${params.platform}

    ${params.realSources ? `Available sources:
    ${params.realSources.map(s => `- ${s}`).join('\n')}` : ''}

    IMPORTANT: Use ONLY these sources in your response.` : ''}

    Generate 2 content ideas as JSON.`;

    // Resto do código continua igual...
}

```

## Parte 3: Removendo o Sistema de Scraping (Com Segurança)



## Fase 1: Teste o Novo Sistema (1 semana)

```
bash

# Configure para usar o novo pipeline
USE_LEGACY_SCRAPING=false

# Teste manualmente:
# 1. Crie um tópico
# 2. Gere ideias de conteúdo
# 3. Verifique se as fontes são dos últimos 7 dias
# 4. Se funcionar, continue. Se não, reverta:
USE_LEGACY_SCRAPING=true
```

## Fase 2: Desabilitar Rotas de Scraping

```
typescript

// Em server/routes.ts, comente estas rotas:

// ❌ COMENTAR (não deletar ainda)
// app.post('/api/scrape-url', ...)
// app.post('/api/bulk-scrape', ...)
// app.post('/api/sync-scraping-targets', ...)
// app.get('/api/scraped-content', ...)
```

## Fase 3: Remover Imports Não Usados

```
typescript

// Em server/routes.ts, remova:
// ❌ import { WebScraper } from './scraping';
// ❌ import { profileScrapingSync } from './profile-scraping-sync';
```

## Fase 4: Deletar Arquivos de Scraping (após 2 semanas)

```
bash

# APENAS após confirmar que tudo funciona:
git rm server/scraping.ts
git rm server/profile-scraping-sync.ts
git rm tests/scraping.test.ts # Se existir

# Commit com mensagem clara
git commit -m "refactor: remove legacy scraping system – replaced with Perplexity API"
```

## Fase 5: Limpar Banco de Dados (opcional)

```
sql

-- Após 30 dias, você pode limpar as tabelas:
-- CUIDADO: Fazer backup primeiro!

-- Verificar o que será deletado
SELECT COUNT(*) FROM scraped_content;
SELECT COUNT(*) FROM scraping_targets;
SELECT COUNT(*) FROM expert_content_relevance;

-- Se decidir deletar (IRREVERSÍVEL):
-- DROP TABLE scraped_content;
-- DROP TABLE scraping_targets;
-- DROP TABLE expert_content_relevance;
```

## Parte 4: Testes Essenciais

### Teste Manual Básico

1. **Antes de começar:** Anote o tempo que leva para gerar conteúdo
2. **Configure:** `USE_LEGACY_SCRAPING=false`
3. **Teste:**

```
bash

# Via curl ou Postman
POST /api/generate-content-ideas
{
  "topicId": 1,
  "platform": "linkedin",
  "expertId": 1
}
```

#### 4. Verifique:

- Resposta tem `metadata.engine: "perplexity-realtime"`
- Sources são URLs válidas
- Tempo < 5 segundos (deve ser mais rápido)

### Teste Automatizado Simples

typescript

```
// tests/content-pipeline-v2.test.ts
describe('ContentPipelineV2', () => {
  it('deve gerar conteúdo com Perplexity', async () => {
    const result = await contentPipelineV2.generateContent({
      topicId: 1,
      platform: 'linkedin',
      expertId: 1
    });

    expect(result.ideas).toHaveLength(2);
    expect(result.sourcesUsed.length).toBeGreaterThan(0);
    expect(result.sourcesUsed[0]).toContain('http');
  });
});
```

## Parte 5: Monitoramento

### Logs para Acompanhar

typescript

```
// Adicione estes logs para debug:
console.log('[PIPELINE] Tipo:', process.env.USE_LEGACY_SCRAPING ? 'LEGACY' : 'NEW');
console.log('[PIPELINE] Tempo:', Date.now() - startTime, 'ms');
console.log('[PIPELINE] Fontes encontradas:', sources.length);
```

### Métricas Importantes

- **Tempo de resposta:** Deve cair de 10-30s para 2-5s
- **Taxa de erro:** Deve se manter < 5%
- **Qualidade das fontes:** Todas devem ser dos últimos 7 dias

### Troubleshooting

#### Erro: "Perplexity API key invalid"

bash

```
# Verifique a chave
echo $PERPLEXITY_API_KEY
# Deve começar com "pplx-"
```

#### Erro: "Rate limit exceeded"

typescript

```
// Aumente o delay em rate-limiter.ts
```

```
private readonly MIN_INTERVAL_MS = 5000; // De 3s para 5s
```

## Erro: "No sources found"

- Verifique se a query está bem formulada
- Tente uma busca mais genérica
- Verifique os logs do Perplexity

## ✅ Checklist Final

### Semana 1:

- ☐ Configurar .env com chaves
- ☐ Criar content-pipeline-v2.ts
- ☐ Modificar routes.ts com switch
- ☐ Testar com USE\_LEGACY\_SCRAPING=false
- ☐ Verificar que fontes são recentes

### Semana 2:

- ☐ Desabilitar rotas de scraping
- ☐ Remover imports não usados
- ☐ Monitorar erros e performance
- ☐ Documentar problemas encontrados

### Semana 3:

- ☐ Deletar arquivos de scraping
- ☐ Atualizar documentação
- ☐ Treinar equipe
- ☐ Remover USE\_LEGACY\_SCRAPING do código

### Mês 2:

- ☐ Considerar limpeza do banco
- ☐ Remover código legado restante
- ☐ Celebrar! 🎉



## Notas Importantes

1. **SEMPRE teste com flag antes de deletar código**
2. **Faça backup do banco antes de qualquer limpeza**

3. **Monitore logs por pelo menos 1 semana após mudanças**
4. **Tenha um plano de rollback (USE\_LEGACY\_SCRAPING=true)**

### **Dica Final**

Se algo der errado, você sempre pode voltar ao sistema antigo mudando uma variável:

```
bash
```

```
USE_LEGACY_SCRAPING=true
```

Isso garante que o sistema continue funcionando enquanto você corrige problemas.