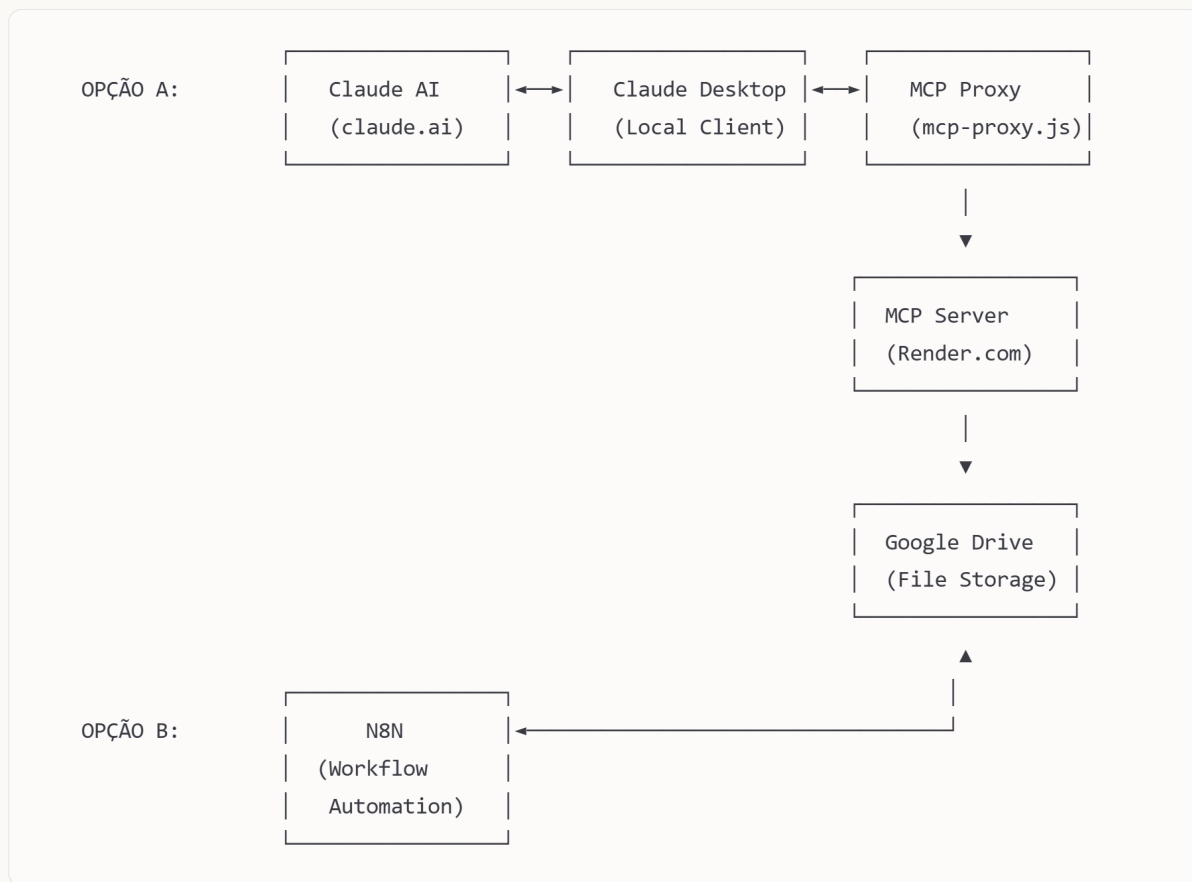


Documentação Completa do Sistema CSV Query MCP Server

Visão Geral da Solução

O CSV Query MCP Server é um sistema distribuído que permite análise de arquivos CSV através de diferentes interfaces cliente. O sistema é composto por múltiplos componentes interconectados que trabalham em conjunto para fornecer uma solução completa de análise fiscal brasileira.

Arquitetura do Sistema



Componentes Principais

1. **MCP Server** - Servidor principal rodando no Render.com
2. **Google Drive** - Armazenamento de arquivos ZIP contendo CSVs
3. **Clientes disponíveis:**
 - **Claude Desktop** (Opção A) - Interface AI completa
 - **N8N** (Opção B) - Automação e workflows

Parte 1: Servidor MCP (Render.com)

Pré-requisitos para Desenvolvimento

- Conta no GitHub
- Conta no Render.com
- Git instalado no Windows
- Node.js instalado
- PowerShell 5.1 ou superior

Passo 1: Instalar Git no Windows

1. Baixe o Git de git-scm.com
2. Execute o instalador com configurações padrão
3. Verifique a instalação no PowerShell:
4. `git --version`

Passo 2: Configurar Repositório GitHub

O código fonte está em: <https://github.com/alcosta35/csv-query-mcp-server>

Para contribuir com o projeto:

1. Fork o repositório no GitHub
2. Clone seu fork localmente usando PowerShell:
3. `git clone https://github.com/[SEU_USUARIO]/csv-query-mcp-server.git`
4. `Set-Location csv-query-mcp-server`
5. Configure o upstream:
6. `git remote add upstream https://github.com/alcosta35/csv-query-mcp-server.git`

Passo 3: Comandos Git Essenciais (PowerShell)

Atualizar código local:

`git fetch upstream`

`git checkout main`

`git merge upstream/main`

Fazer mudanças:

`git add .`

`git commit -m "Descrição das mudanças"`

`git push origin main`

Verificar status:

git status

git log --oneline -10 # Últimos 10 commits

Deployment automático:

O Render.com está configurado para deployment automático. Qualquer push para a branch main irá:

1. Trigger um novo build no Render
2. Executar npm install && npm run build
3. Reiniciar o serviço automaticamente

Passo 4: Configuração no Render.com

Variáveis de Ambiente Necessárias:

1. Acesse o Dashboard do Render.com
2. Vá para o serviço csv-query-mcp
3. Acesse **Environment**
4. Configure as seguintes variáveis:

NODE_ENV=production

MCP_AUTH_TOKEN=mcp24d91738b41c1-92e8498715c086acd16c7e9d84124590b1d5562881d5d4f7

GOOGLE_DRIVE_FOLDER_ID=11KiQQCN1yR0jV9DDgZ1BLZ5qhIL_BE38

GOOGLE_CLIENT_ID=994566912326-trn9aap99rnglrrtgrp1754mterd8i60t.apps.googleusercontent.com

GOOGLE_CLIENT_SECRET=GOCS PX-CHsy59aluu4QH4rYIsVY5pkEDbeH

GOOGLE_REDIRECT_URI=https://csv-query-mcp.onrender.com/auth/callback

GOOGLE_REFRESH_TOKEN=[TOKEN_OBTIDO_VIA_OAUTH]

PORT=3000

Configuração de Build:

- **Build Command:** npm install && npm run build
- **Start Command:** npm run start:prod
- **Environment:** Node
- **Auto-Deploy:** Enabled

Passo 5: API do Render

Use PowerShell para interagir com a API do Render:

Listar serviços

```
$headers = @{  
    "Authorization" = "Bearer [SEU_TOKEN]"  
    "Content-Type" = "application/json"  
}
```

Invoke-RestMethod -Uri "https://api.render.com/v1/services" -Method GET -Headers \$headers

Trigger deploy manual

Invoke-RestMethod -Uri "https://api.render.com/v1/services/[SERVICE_ID]/deploys" -Method POST -
Headers \$headers

Passo 6: Endpoints Disponíveis

O servidor oferece os seguintes endpoints HTTP:

Status e Saúde:

Invoke-RestMethod -Uri "https://csv-query-mcp.onrender.com/health" -Method GET

Autenticação Google Drive:

Start-Process "https://csv-query-mcp.onrender.com/auth"

Listar arquivos do Drive:

```
$headers = @{  
    "Authorization" = "Bearer mcp24d91738b41c1-  
92e8498715c086acd16c7e9d84124590b1d5562881d5d4f7"  
}
```

Invoke-RestMethod -Uri "https://csv-query-mcp.onrender.com/drive/files" -Method GET -Headers
\$headers

Endpoint MCP principal:

```
$body = @{  
    "jsonrpc" = "2.0"  
    "method" = "tools/call"  
    "params" = @{
```

```
        "name" = "list_drive_files"
        "arguments" = @{}
    }
    "id" = 1
} | ConvertTo-Json -Depth 3
```

```
$headers = @{
    "Authorization" = "Bearer mcp24d91738b41c1-
92e8498715c086acd16c7e9d84124590b1d5562881d5d4f7"
    "Content-Type" = "application/json"
}
```

```
Invoke-RestMethod -Uri "https://csv-query-mcp.onrender.com/mcp" -Method POST -Body $body -
Headers $headers
```

Passo 7: Scripts de Teste PowerShell

Crie um script de teste (test-mcp.ps1):

```
# Test MCP Server Health
```

```
Write-Host "Testing MCP Server Health..." -ForegroundColor Green
```

```
$health = Invoke-RestMethod -Uri "https://csv-query-mcp.onrender.com/health" -Method GET
```

```
$health | ConvertTo-Json -Depth 3
```

```
# Test Drive Files List
```

```
Write-Host "`nTesting Drive Files List..." -ForegroundColor Green
```

```
$headers = @{
```

```
    "Authorization" = "Bearer mcp24d91738b41c1-
92e8498715c086acd16c7e9d84124590b1d5562881d5d4f7"
}
```

```
try {
```

```
$files = Invoke-RestMethod -Uri "https://csv-query-mcp.onrender.com/drive/files" -Method GET -
Headers $headers
```

```
$files | ConvertTo-Json -Depth 3
```

```
Write-Host "✓ Drive integration working" -ForegroundColor Green
```

```
} catch {
```

```
Write-Host "✗ Drive integration failed: $($_.Exception.Message)" -ForegroundColor Red
```

```
}
```

```
# Test MCP Tools List
```

```
Write-Host "`nTesting MCP Tools..." -ForegroundColor Green
```

```
$body = @{
```

```
    "jsonrpc" = "2.0"
```

```
    "method" = "tools/list"
```

```
    "id" = 1
```

```
} | ConvertTo-Json
```

```
$headers["Content-Type"] = "application/json"
```

```
try {
```

```
    $tools = Invoke-RestMethod -Uri "https://csv-query-mcp.onrender.com/mcp" -Method POST -Body
    $body -Headers $headers
```

```
    Write-Host "Available tools: $($tools.result.tools.Count)" -ForegroundColor Green
```

```
    $tools.result.tools | ForEach-Object { Write-Host " - $($_.name)" }
```

```
} catch {
```

```
    Write-Host "✗ MCP tools failed: $($_.Exception.Message)" -ForegroundColor Red
```

```
}
```

```
Execute o script:
```

```
.\test-mcp.ps1
```

Componente Google Drive

O Google Drive serve como repositório centralizado para os arquivos ZIP contendo CSVs de notas fiscais brasileiras.

Estrutura dos Dados

- **Arquivo principal:** 202401_NFs.zip
- **Conteúdo:**
 - 202401_NFs_Cabecalho.csv - Dados do cabeçalho das notas fiscais
 - 202401_NFs_Itens.csv - Dados dos itens das notas fiscais

Autenticação OAuth 2.0

O sistema usa OAuth 2.0 para acessar o Google Drive de forma segura:

1. **Client ID:** Identifica a aplicação
2. **Client Secret:** Chave secreta da aplicação
3. **Redirect URI:** URL de callback após autenticação
4. **Refresh Token:** Token para renovação automática de acesso

Teste de Conectividade Google Drive

Use PowerShell para testar a conectividade:

Verificar autenticação Google Drive

```
$response = Invoke-RestMethod -Uri "https://csv-query-mcp.onrender.com/health" -Method GET
```

```
if ($response.authenticated) {
```

```
    Write-Host "✓ Google Drive authenticated" -ForegroundColor Green
```

```
} else {
```

```
    Write-Host "✗ Google Drive not authenticated" -ForegroundColor Red
```

```
    Write-Host "Visit: https://csv-query-mcp.onrender.com/auth" -ForegroundColor Yellow
```

```
}
```

Parte 2: Opções de Cliente

Opção A: Claude Desktop (Recomendado)

O Claude Desktop oferece a experiência mais integrada, permitindo conversar com o Claude AI enquanto usa as ferramentas de análise CSV.

Pré-requisitos

- Windows 10/11
- PowerShell 5.1 ou superior
- Acesso à internet
- Conta no GitHub (opcional, para download direto)

Passo 1: Instalar o Claude Desktop

1. Acesse claude.ai e baixe o Claude Desktop
2. Execute o instalador e siga as instruções padrão
3. Faça login com sua conta Claude AI

Passo 2: Baixar Arquivos de Configuração

Os arquivos necessários estão disponíveis no GitHub: <https://github.com/alcosta35/i2a2>

Opção A: Download via GitHub Web

1. Acesse <https://github.com/alcosta35/i2a2>
2. Clique em "Code" → "Download ZIP"
3. Extraia os arquivos em um diretório temporário

Opção B: PowerShell (se tiver Git instalado)

git clone <https://github.com/alcosta35/i2a2.git>

Set-Location i2a2

Passo 3: Configurar o MCP Proxy

1. Crie o diretório para o proxy usando PowerShell:
2. `$ProxyPath = "$env:USERPROFILE\Documents\i2a2-mcp-proxy"`
3. `New-Item -ItemType Directory -Path $ProxyPath -Force`
4. Copie o arquivo mcp-proxy.js para este diretório:
5. `Copy-Item -Path ".\mcp-proxy.js" -Destination $ProxyPath`
6. Verifique se o Node.js está instalado:
7. `node --version`

Se não estiver instalado, baixe de nodejs.org

Passo 4: Configurar o Claude Desktop

1. Navegue até o diretório de configuração do Claude usando PowerShell:
2. `$ConfigPath = "$env:APPDATA\Claude"`

3. Set-Location \$ConfigPath
4. Se o arquivo claude_desktop_config.json não existir, crie-o:
5. if (!(Test-Path "claude_desktop_config.json")) {
6. New-Item -ItemType File -Name "claude_desktop_config.json"
7. }
8. Edite o arquivo com o conteúdo apropriado:
9. @"

```
{ "mcpServers": { "csv-query-server": { "command": "node", "args":  
["$env:USERPROFILE\Documents\i2a2-mcp-proxy\mcp-proxy.js"], "env": {} } } } "@ | Out-File -FilePath  
"claude_desktop_config.json" -Encoding UTF8
```

Passo 5: Reiniciar e Verificar

1. Feche completamente o Claude Desktop:

```
```powershell
```

```
Get-Process | Where-Object {$_.Name -like "*Claude*"} | Stop-Process -Force
```

2. Reinicie o Claude Desktop
3. Verifique se as ferramentas estão disponíveis:
  - Abra o Claude Desktop
  - Vá em **Configurações** → **Integrações**
  - Procure por **CSV-Query-Server**

### Passo 6: Configurar Permissões

1. Em **Configurações** → **Integrações** → **CSV-Query-Server** → **Ferramentas**
2. Configure TODAS as 4 ferramentas como **"Allow Unsupervised"**:
  - load\_csv\_from\_drive
  - list\_drive\_files
  - query\_loaded\_csv
  - analyze\_nf\_data

**CRÍTICO:** As ferramentas NÃO devem estar como "Always ask permission"






## Verificação Final - Claude Desktop

Digite no Claude Desktop:




Por favor, liste os arquivos disponíveis no Google Drive

Se tudo estiver configurado corretamente, você verá uma lista de arquivos do Google Drive.

## Vantagens do Claude Desktop

-  Interface AI completa e conversacional
-  Integração nativa com MCP
-  Ferramentas aparecem automaticamente no chat
-  Contexto mantido durante análises
-  Não requer conhecimento técnico para usar

## Limitações do Claude Desktop

-  Requer instalação local
-  Menos flexível para automação
-  Interface limitada ao chat

## Opção B: N8N (Para Automação)

O N8N oferece uma interface de workflow que permite criar automações complexas e personalizadas usando o MCP server.

## Pré-requisitos N8N

- N8N instalado (local ou cloud)
- Conhecimento básico de workflows
- Conta OpenAI (para análise AI)

## Passo 1: Instalar N8N

### Instalação local via PowerShell:

# Instalar N8N globalmente

```
npm install n8n -g
```

# Ou usar via npx (sem instalação global)

```
npx n8n
```

### Ou usar N8N Cloud:

Acesse [n8n.cloud](https://n8n.cloud) e crie uma conta.

## Passo 2: Configurar Credenciais N8N

1. **HTTP Header Auth** para MCP Server:
  - **Header Name:** Authorization
  - **Header Value:** Bearer mcp24d91738b41c1-92e8498715c086acd16c7e9d84124590b1d5562881d5d4f7
2. **OpenAI API** (para análise AI):
  - Configure sua chave da API OpenAI

## Passo 3: Importar Workflow

1. Baixe o arquivo `Google_Drive_CSV_Analysis.json` do GitHub
2. No N8N, vá em **Workflows** → **Import from File**
3. Selecione o arquivo JSON baixado

## Passo 4: Configurar o Workflow

O workflow inclui os seguintes nós:

1. **Chat Trigger** - Recebe perguntas do usuário
2. **List Google Drive Files** - Lista arquivos disponíveis
3. **Extract ZIP File Info** - Extrai informações do arquivo ZIP
4. **Load CSV from ZIP** - Carrega dados dos CSVs
5. **Format Data for AI** - Formata dados para análise
6. **AI Analysis** - Análise com OpenAI GPT-4
7. **Memory** - Mantém contexto da conversa

## Passo 5: Teste do Workflow N8N

Execute cada nó individualmente para verificar:

# Teste via PowerShell (exemplo de chamada que o N8N faria)

```
$body = @{
 "jsonrpc" = "2.0"
 "method" = "tools/call"
 "params" = @{
 "name" = "analyze_nf_data"
```







```
"arguments" = @{
 "analysis_type" = "total_nfs"
}
}

"id" = 1
} | ConvertTo-Json -Depth 3
```





```
$headers = @{
 "Authorization" = "Bearer mcp24d91738b41c1-
92e8498715c086acd16c7e9d84124590b1d5562881d5d4f7"
 "Content-Type" = "application/json"
}
}
```

Invoke-RestMethod -Uri "https://csv-query-mcp.onrender.com/mcp" -Method POST -Body \$body -  
Headers \$headers

### Vantagens do N8N

-  Automação completa de workflows
-  Interface visual para criação de fluxos
-  Integração com múltiplos serviços
-  Agendamento e triggers diversos
-  Customização avançada de lógica
-  Pode ser usado por equipes técnicas

### Limitações do N8N

-  Curva de aprendizado mais alta
-  Requer configuração técnica
-  Interface menos intuitiva para usuários finais
-  Não é conversacional como Claude

### Casos de Uso N8N

- **Relatórios automatizados:** Gerar análises fiscal em horários específicos

- **Monitoramento:** Verificar novos arquivos no Drive periodicamente
- **Integração:** Conectar com outros sistemas empresariais
- **Alertas:** Enviar notificações baseadas em análises
- **Dashboards:** Alimentar sistemas de BI com dados processados

### Comparação das Opções

Aspecto	Claude Desktop	N8N
Facilidade de Uso	★★★★★	★★★★★
Automação	★★	★★★★★
Interface Conversacional	★★★★★	★
Flexibilidade	★★★	★★★★★
Configuração Inicial	★★★★	★★
Integração com Sistemas	★★	★★★★★
Análise Ad-hoc	★★★★★	★★
Escalabilidade	★★★	★★★★★

### Recomendações de Uso

#### Use Claude Desktop quando:

- Precisar fazer análises pontuais e conversacionais
- Usuários finais sem conhecimento técnico
- Análise exploratória de dados
- Perguntas específicas sobre notas fiscais
- Interface amigável e intuitiva

#### Use N8N quando:

- Precisar de automação e workflows
- Integração com outros sistemas
- Relatórios periódicos automatizados
- Equipe técnica disponível
- Processamento em lote de dados

## Troubleshooting

### Problemas Comuns

1. **Erro de autenticação:** Verifique o MCP\_AUTH\_TOKEN
2. **Google Drive erro 401:** Renovar GOOGLE\_REFRESH\_TOKEN
3. **Servidor não responde:** Verificar logs no Render Dashboard

### Scripts de Diagnóstico PowerShell

#### Verificar servidor MCP:

# Verificar saúde do servidor

```
$health = Invoke-RestMethod -Uri "https://csv-query-mcp.onrender.com/health" -Method GET
```

```
Write-Host "=== MCP Server Status ===" -ForegroundColor Blue
```

```
Write-Host "Status: $($health.status)" -ForegroundColor $(if($health.status -eq 'healthy') {'Green'} else {'Red'})
```

```
Write-Host "Authenticated: $($health.authenticated)" -ForegroundColor $(if($health.authenticated) {'Green'} else {'Red'})
```

```
Write-Host "Loaded Tables: $($health.loadedTables -join ', ')" -ForegroundColor Cyan
```

```
Write-Host "Timestamp: $($health.timestamp)" -ForegroundColor Gray
```

#### Verificar configuração Claude Desktop:

# Verificar arquivos de configuração

```
$ConfigPath = "$env:APPDATA\Claude\claude_desktop_config.json"
```

```
$ProxyPath = "$env:USERPROFILE\Documents\i2a2-mcp-proxy\mcp-proxy.js"
```

```
Write-Host "=== Claude Desktop Configuration ===" -ForegroundColor Blue
```

```
if (Test-Path $ConfigPath) {
```

```
 Write-Host "✓ Claude config found" -ForegroundColor Green
```

```
 $config = Get-Content $ConfigPath | ConvertFrom-Json
```

```
 Write-Host "MCP Servers: $($config.mcpServers.PSObject.Properties.Name -join ', ')"
```

```
} else {
```

```
Write-Host "X Claude config missing: $ConfigPath" -ForegroundColor Red
}
```

```
if (Test-Path $ProxyPath) {
 Write-Host "✓ MCP Proxy found" -ForegroundColor Green
} else {
 Write-Host "X MCP Proxy missing: $ProxyPath" -ForegroundColor Red
}
```

# Verificar Node.js

```
try {
 $nodeVersion = node --version
 Write-Host "✓ Node.js version: $nodeVersion" -ForegroundColor Green
} catch {
 Write-Host "X Node.js not installed or not in PATH" -ForegroundColor Red
}
```

### Logs

- **Cliente Claude Desktop:** \$env:USERPROFILE\Documents\i2a2-mcp-proxy\mcp-proxy.log
- **Servidor:** Render Dashboard → Logs
- **Claude Desktop:** Configurações → Logs
- **N8N:** Interface N8N → Executions

### Monitorar logs em tempo real:

# Monitorar log do proxy (Claude Desktop)

```
Get-Content "$env:USERPROFILE\Documents\i2a2-mcp-proxy\mcp-proxy.log" -Wait -Tail 10
```

### Comandos de Manutenção PowerShell

# Limpar logs antigos

```
Remove-Item "$env:USERPROFILE\Documents\i2a2-mcp-proxy\mcp-proxy.log" -ErrorAction SilentlyContinue
```

# Reiniciar Claude Desktop

```
Get-Process | Where-Object {$_.Name -like "*Claude*"} | Stop-Process -Force
```

Start-Sleep 2

# Claude Desktop deve ser iniciado manualmente

# Atualizar repositório local

```
Set-Location "$env:USERPROFILE\Documents\csv-query-mcp-server"
```

```
git fetch upstream
```

```
git checkout main
```

```
git merge upstream/main
```

# Reiniciar N8N (se instalado localmente)

```
Get-Process | Where-Object {$_.Name -like "*n8n*"} | Stop-Process -Force
```

Start-Sleep 2

```
npx n8n
```

## Suporte

Para suporte técnico, consulte:

- **Issues GitHub:** <https://github.com/alcosta35/csv-query-mcp-server/issues>
- **Documentação Render:** <https://render.com/docs>
- **Claude MCP Docs:** <https://modelcontextprotocol.io>
- **N8N Documentation:** <https://docs.n8n.io>

---

**Versão:** 1.0.0

**Última atualização:** Dezembro 2024

**Autor:** Sistema CSV Query MCP Server