🔗 Portal NewLoc - Integração com n8n

📋 Visão Geral

Este guia mostra como integrar o Portal NewLoc com o n8n para automações e integrações externas.

🔐 1. Fluxo de Autenticação

1.1. Workflow de Login

```
{
  "name": "NewLoc - Login",
  "nodes": [
    {
      "parameters": {
        "url": "https://app.newloc.com/api/auth/login",
        "authentication": "none",
        "requestMethod": "POST",
        "jsonParameters": true,
        "options": {
          "allowUnauthorizedCerts": false,
          "followRedirect": true
        },
        "bodyParametersJson": "={\"email\": \"{{ $json.email }}\", \"password\":
\"{{ $json.password }}\"}"
      },
      "name": "Login Request",
      "type": "n8n-nodes-base.httpRequest",
      "typeVersion": 1,
      "position": [250, 300]
    },
    {
      "parameters": {
        "keepOnlySet": true,
        "values": {
          "string": [
            {
              "name": "token",
              "value": "={{ $json.token }}"
            },
            {
              "name": "userId",
              "value": "={{ $json.user.id }}"
            },
            {
              "name": "userType",
              "value": "={{ $json.user.tipo }}"
            },
            {
              "name": "userEmail",
              "value": "={{ $json.user.email }}"
            }
          ]
        }
      },
      "name": "Extrair Token",
      "type": "n8n-nodes-base.set",
      "typeVersion": 1,
      "position": [450, 300]
    }
  ],
  "connections": {
    "Login Request": {
      "main": [
        [
          {
            "node": "Extrair Token",
            "type": "main",
            "index": 0
          }
        ]
      ]
```

```
    }
  }
}
```

Como usar:
1. Crie um novo workflow no n8n
2. Adicione um node “Webhook” ou “Schedule” como trigger
3. Configure o HTTP Request com a URL da API
4. Use o node Set para extrair e armazenar o token

📄 2. Operações com Documentos

2.1. Listar Todos os Documentos

Node: HTTP Request

```
// Configuração do Node
{
  "method": "GET",
  "url": "https://app.newloc.com/api/documentos",
  "authentication": "none",
  "headerParameters": {
    "parameters": [
      {
        "name": "Cookie",
        "value": "session_token={{ $json.token }}"
      }
    ]
  },
  "options": {
    "response": {
      "response": {
        "fullResponse": false,
        "responseFormat": "json"
      }
    }
  }
}
```

2.2. Obter Documento Específico

Node: HTTP Request

```
{
  "method": "GET",
  "url": "https://app.newloc.com/api/documento/{{ $json.documentoId }}",
  "authentication": "none",
  "headerParameters": {
    "parameters": [
      {
        "name": "Cookie",
        "value": "session_token={{ $json.token }}"
      }
    ]
  }
}
```

2.3. Processar Imagem do Documento

```
// Node Function para extrair e processar imagem
const documento = $input.all()[0].json;

// Extrair base64 da imagem
const imagemBase64 = documento.documentacaoImagem;

// Remover prefixo data:image se existir
const base64Clean = imagemBase64.replace(/^data:image\/\w+;base64,/, '');

// Converter para buffer (se necessário salvar)
const imageBuffer = Buffer.from(base64Clean, 'base64');

return {
  json: {
    documentoId: documento.id,
    cliente: documento.cliente,
    remessa: documento.remessa,
    imagemBase64: base64Clean,
    imagemTamanho: imageBuffer.length
  }
};
```

👥 3. Gerenciamento de Usuários (Admin)

3.1. Criar Novo Cliente

Node: HTTP Request

```
{
  "method": "POST",
  "url": "https://app.newloc.com/api/usuarios",
  "authentication": "none",
  "headerParameters": {
    "parameters": [
      {
        "name": "Cookie",
        "value": "session_token={{ $json.token }}"
      },
      {
        "name": "Content-Type",
        "value": "application/json"
      }
    ]
  },
  "bodyParametersJson": "={\"email\": \"{{ $json.email }}\", \"password\":
\"{{ $json.password }}\", \"cliente\": \"{{ $json.cliente }}\"}"
}
```

3.2. Listar Todos os Usuários

```
{
  "method": "GET",
  "url": "https://app.newloc.com/api/usuarios",
  "authentication": "none",
  "headerParameters": {
    "parameters": [
      {
        "name": "Cookie",
        "value": "session_token={{ $json.adminToken }}"
      }
    ]
  }
}
```

🔄 4. Workflow Completo: Sincronização de Documentos

Este workflow autentica, busca documentos e processa as informações.

```
{
  "name": "NewLoc - Sincronização Completa",
  "nodes": [
    {
      "parameters": {
        "rule": {
          "interval": [
            {
              "field": "hours",
              "hoursInterval": 1
            }
          ]
        }
      },
      "name": "Schedule Trigger",
      "type": "n8n-nodes-base.scheduleTrigger",
      "typeVersion": 1,
      "position": [250, 300]
    },
    {
      "parameters": {
        "values": {
          "string": [
            {
              "name": "email",
              "value": "admin@newloc.com"
            },
            {
              "name": "password",
              "value": "Admin@123"
            }
          ]
        }
      },
      "name": "Credenciais",
      "type": "n8n-nodes-base.set",
      "typeVersion": 1,
      "position": [450, 300]
    },
    {
      "parameters": {
        "url": "https://app.newloc.com/api/auth/login",
        "requestMethod": "POST",
        "jsonParameters": true,
        "bodyParametersJson": "={\"email\": \"{{ $json.email }}\", \"password\":
\"{{ $json.password }}\"}"
      },
      "name": "Login",
      "type": "n8n-nodes-base.httpRequest",
      "typeVersion": 1,
      "position": [650, 300]
    },
    {
      "parameters": {
        "url": "https://app.newloc.com/api/documentos",
        "requestMethod": "GET",
        "headerParameters": {
          "parameters": [
            {
              "name": "Cookie",
              "value": "session_token={{ $json.token }}"
            }
```

```
          ]
        }
      },
      "name": "Buscar Documentos",
      "type": "n8n-nodes-base.httpRequest",
      "typeVersion": 1,
      "position": [850, 300]
    },
    {
      "parameters": {
        "functionCode": "// Processar cada documento\nconst documentos = $input.all()
[0].json;\n\nreturn documentos.map(doc => ({\n  json: {\n    id: doc.id,\n    cliente:
doc.cliente,\n    remessa: doc.remessa,\n    contrato: doc.contrato,\n    operacao:
doc.operacao,\n    status: doc.status,\n    qtdPatrimonios: doc.patrimonios.length,
\n    dataDocumento: doc.dataDocumento,\n    criadoEm: doc.createdAt\n  }\n}));"
      },
      "name": "Processar Dados",
      "type": "n8n-nodes-base.function",
      "typeVersion": 1,
      "position": [1050, 300]
    }
  ],
  "connections": {
    "Schedule Trigger": {
      "main": [
        [
          {
            "node": "Credenciais",
            "type": "main",
            "index": 0
          }
        ]
      ]
    },
    "Credenciais": {
      "main": [
        [
          {
            "node": "Login",
            "type": "main",
            "index": 0
          }
        ]
      ]
    },
    "Login": {
      "main": [
        [
          {
            "node": "Buscar Documentos",
            "type": "main",
            "index": 0
          }
        ]
      ]
    },
    "Buscar Documentos": {
      "main": [
        [
          {
            "node": "Processar Dados",
            "type": "main",
            "index": 0
```

```
            }
          ]
        ]
      }
    }
  }
```

📊 5. Workflow: Receber Documento do WhatsApp (Evolution API)

Este workflow recebe imagens do WhatsApp, converte para base64 e cria documentos.

```
{
  "name": "NewLoc - WhatsApp para Portal",
  "nodes": [
    {
      "parameters": {
        "path": "whatsapp-newloc",
        "responseMode": "responseNode",
        "options": {}
      },
      "name": "Webhook WhatsApp",
      "type": "n8n-nodes-base.webhook",
      "typeVersion": 1,
      "position": [250, 300]
    },
    {
      "parameters": {
        "functionCode": "// Extrair dados do WhatsApp\nconst msg = $in-
put.first().json;\n\nreturn {\n  json: {\n    from: msg.key.remoteJid,\n    message-
Type: msg.message?.imageMessage ? 'image' : 'text',\n    caption:
msg.message?.imageMessage?.caption || '',\n    imageUrl:
msg.message?.imageMessage?.url || null\n  }\n};"
      },
      "name": "Processar Mensagem",
      "type": "n8n-nodes-base.function",
      "typeVersion": 1,
      "position": [450, 300]
    },
    {
      "parameters": {
        "url": "={{ $json.imageUrl }}",
        "requestMethod": "GET",
        "options": {
          "response": {
            "response": {
              "responseFormat": "file"
            }
          }
        }
      },
      "name": "Baixar Imagem",
      "type": "n8n-nodes-base.httpRequest",
      "typeVersion": 1,
      "position": [650, 300]
    },
    {
      "parameters": {
        "functionCode": "// Converter imagem para base64\nconst binaryData = $in-
put.first().binary.data;\nconst base64 = binaryData.toString('base64');\nconst dataUri
= `${base64}`;\n\nreturn {\n  json: {\n    imagemBase64:
dataUri\n  }\n};"
      },
      "name": "Converter Base64",
      "type": "n8n-nodes-base.function",
      "typeVersion": 1,
      "position": [850, 300]
    },
    {
      "parameters": {
        "values": {
          "string": [
            {
              "name": "email",
```

```
              "value": "admin@newloc.com"
            },
            {
              "name": "password",
              "value": "Admin@123"
            }
          ]
        }
      },
      "name": "Credenciais Admin",
      "type": "n8n-nodes-base.set",
      "typeVersion": 1,
      "position": [1050, 300]
    },
    {
      "parameters": {
        "url": "https://app.newloc.com/api/auth/login",
        "requestMethod": "POST",
        "jsonParameters": true,
        "bodyParametersJson": "={\"email\": \"{{ $json.email }}\", \"password\":
\"{{ $json.password }}\"}"
      },
      "name": "Login Portal",
      "type": "n8n-nodes-base.httpRequest",
      "typeVersion": 1,
      "position": [1250, 300]
    },
    {
      "parameters": {
        "functionCode": "// Preparar dados do documento\nconst items = $input.all();
\nconst imagemData = items.find(i => i.json.imagemBase64);\nconst loginData =
items.find(i => i.json.token);\n\n// Extrair informações da caption (formato: CLIENTE|
REMESSA|CONTRATO)\nconst caption = imagemData.json.caption || '';\nconst [cliente,
remessa, contrato] = caption.split('|').map(s => s.trim());\n\nreturn {\n  json:
{\n    token: loginData.json.token,\n    documento: {\n      date: new
Date().toISOString(),\n      cliente: cliente || 'Cliente WhatsApp',\n      dataDocu-
mento: new Date().toISOString(),\n      remessa: remessa || 'REM-WPP-' + Date.now(),
\n      contrato: contrato || 'CTR-WPP-' + Date.now(),\n      operacao: 'recebi-
mento_whatsapp',\n      patrimonios: [],\n      documentacaoImagem: imagem-
Data.json.imagemBase64,\n      status: 'pendente'\n    }\n  }\n};"
      },
      "name": "Preparar Documento",
      "type": "n8n-nodes-base.function",
      "typeVersion": 1,
      "position": [1450, 300]
    }
  ]
}
```


Formato da Caption no WhatsApp:

```
Construtora Silva|REM-2024-001|CTR-2024-001
```

🔧 6. Tratamento de Erros

6.1. Node de Tratamento de Erro

```
// Function Node para tratar erros
try {
  const response = $input.first().json;

  if (response.error) {
    return {
      json: {
        sucesso: false,
        erro: response.error,
        timestamp: new Date().toISOString()
      }
    };
  }

  return {
    json: {
      sucesso: true,
      dados: response,
      timestamp: new Date().toISOString()
    }
  };
} catch (error) {
  return {
    json: {
      sucesso: false,
      erro: error.message,
      timestamp: new Date().toISOString()
    }
  };
}
```

6.2. Retry Logic

Configurar no HTTP Request Node:
- **Retry on Fail:** Ativado
- **Max Tries:** 3
- **Wait Between Tries:** 1000ms

📨 7. Notificações

7.1. Enviar Email ao Criar Documento

```
// Após criar documento, enviar notificação
{
  "name": "Enviar Email",
  "type": "n8n-nodes-base.emailSend",
  "parameters": {
    "fromEmail": "noreply@newloc.com",
    "toEmail": "={{ $json.emailDestino }}",
    "subject": "Novo Documento - {{ $json.remessa }}",
    "text": "Um novo documento foi criado:\n\nCliente: {{ $json.cliente }}\nRemessa:
{{ $json.remessa }}\nContrato: {{ $json.contrato }}\n\nAcesse: https://app.newloc.com/
documento/{{ $json.id }}"
  }
}
```

🔐 8. Boas Práticas de Segurança

8.1. Armazenar Credenciais

Nunca hardcode credenciais! Use n8n Credentials:

1. No n8n, vá em **Settings → Credentials**
2. Crie um novo credential tipo “Header Auth”
3. Configure:
   - **Name:** `NewLoc Session`
   - **Header Name:** `Cookie`
   - **Value:** `session_token=YOUR_TOKEN`
4. Nos HTTP Request nodes, selecione este credential

8.2. Renovar Sessão

As sessões expiram em 8 horas. Implemente lógica de renovação:

```
// Function Node - Verificar Expiração
const loginTime = $json.loginTimestamp;
const now = Date.now();
const hours = (now - loginTime) / (1000 * 60 * 60);

if (hours >= 7.5) {
  // Renovar sessão
  return {
    json: {
      needsRenewal: true
    }
  };
}

return {
  json: {
    needsRenewal: false,
    token: $json.token
  }
};
```

📋 9. Checklist de Integração

- [ ] Configurar URL base da API
- [ ] Testar endpoint de health check
- [ ] Criar credenciais no n8n
- [ ] Testar fluxo de login
- [ ] Configurar tratamento de erros
- [ ] Implementar retry logic
- [ ] Testar operações com documentos
- [ ] Configurar webhooks (se necessário)
- [ ] Implementar renovação de sessão
- [ ] Configurar logs e monitoramento
- [ ] Testar em ambiente de produção
- [ ] Documentar workflows criados

🆘 10. Troubleshooting

Erro 401 - Não Autenticado

- Verificar se o token está sendo enviado corretamente
- Verificar se a sessão não expirou
- Tentar fazer login novamente

Erro 403 - Acesso Negado

- Verificar permissões do usuário
- Cliente tentando acessar dados de outro cliente
- Operação requer privilégios de admin

Erro 500 - Erro Interno

- Verificar logs do servidor
- Verificar conexão com banco de dados
- Verificar formato dos dados enviados

Timeout

- Aumentar timeout no HTTP Request node
- Verificar conexão de rede
- Imagens muito grandes podem causar timeout

📞 Suporte

Para mais informações, consulte:
- Documentação da API (./API_DOCUMENTATION.md)
- Guia de Deploy (./DEPLOY_GUIDE.md)
- Logs do servidor: `docker compose logs -f web`