🔍

Filtro Diário de Editais

Documentação do Fluxo de Licitações Dautin

Controle de Versão: Responsável - Anderson Corrêa de Liz

Última Atualização: 07/08/2025

Status: Em produção

---

1. Visão Geral

1.1 Objetivo

O **Filtro de Licitações Dautin** automatiza a identificação de oportunidades em licitações públicas brasileiras, entregando diariamente às empresas:

- **Oportunidades relevantes** filtradas por perfil de negócio
- **Redução de 90% no tempo** de busca manual
- **Cobertura nacional ou regional** conforme plano contratado

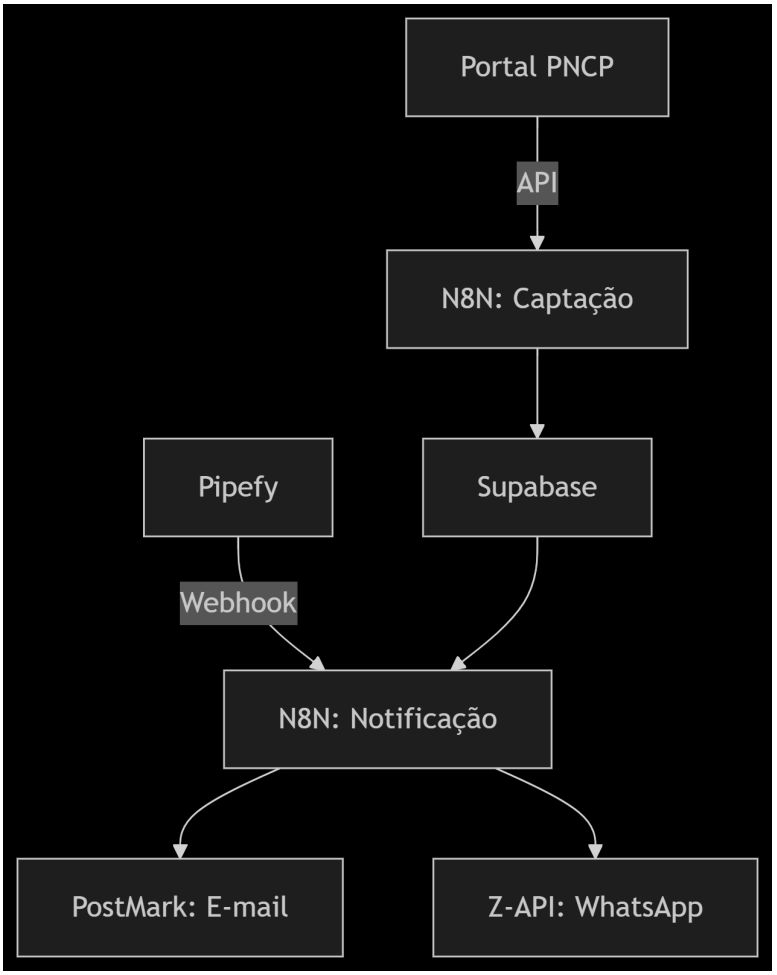
1.2 Benefícios Chave

Benefício	Impacto
Eliminação de busca manual	Economia de 4-6h/dia
Filtros personalizados	85%+ de relevância nas oportunidades
Notificação multicanal	Agilidade na tomada de decisão

---

2. Arquitetura da Solução

2.1 Diagrama de Fluxo



2.2 Stack Tecnológica

Camada	Tecnologia	Função
Orquestração	n8n	Automação de workflows
Armazenamento	Supabase	Banco de dados relacional
Comunicação	PostMark + Z-API	Notificações por e-mail e WhatsApp
Gerenciamento	Pipefy	Cadastro de clientes

3. Fluxo Detalhado

3.1 Captação de Licitações (PNCP → Supabase)

O fluxo de processamento para obtenção das licitações do PNCP (Portal Nacional de Contratações Públicas) segue uma série de etapas automatizadas

usando o N8N. O processo inicia com um gatilho agendado que executa diariamente, consulta a API do PNCP para obter as licitações mais recentes, processa os dados e os armazena no banco de dados Supabase para consulta posterior.

Este processo é executado para várias modalidades de licitação, cada uma com seu próprio fluxo

Frequência: Diária (01:00h)

Modalidades Suportadas:

1. Concorrência Eletrônica (ID 4)
2. Concorrência Presencial (ID 5)
3. Pregão Eletrônico (ID 6)
4. Pregão Presencial (ID 7)
5. Dispensa de Licitação (ID 8)

3.1.1 Nós do N8N (Captação de Licitações)

Descrição	Tipo	Função	Anexos
Concorrência - Eletrônica (4)	Schedule Trigger	Todos os dias à 1hr executar o workflow	
Criação Variáveis3	Code	Criação de 3 variáveis (ontem, numeroPagina, modalidade)	1. Criação Variáveis3
HTTP Request3	HTTP Request	Fazer requisição de URL no PNCP	2. Ex. requisição webhook
Edit Fields4	Edit Fields	Extrair dados desta primeira requisição como o (totalPaginas)	3. Resultado do nó
If4	If	Condição que se o número de páginas for zero (vazio) encerra o processo, senão ele envia	

		para outro nó (continuidade)	
Code5	Code	Cria várias requisições, conforme o número de páginas verificado anteriormente. Para cada item há um URL único.	4. Criação de chamadas webhook
HTTP Request4	HTTP Request	Faz a requisição de todos os URLs gerados no fluxo anterior, diretamente no site PNCP	
Tratamento de dados3	Code	Captura informações relevantes sobre as licitações que foram solicitadas	5. Código do tratamento
Salvar no Banco de Dados3	Supabase	Envio das informações para o banco de dados Supabase para registro e para futuras consultas	6. Ex. resultado JSON de registro

3.2 Processamento - Notificação de Licitações

Este fluxo automatizado recebe informações de clientes cadastrados no **Pipefy** via **Webhook**, processa os dados, consulta licitações no **Supabase** com base no plano do cliente (Avançado ou Intermediário), filtra oportunidades por palavras-chave e envia notificações por **e-mail (PostMark)** e **WhatsApp (Z-API)**, conforme as preferências do cliente.

Frequência: Diária (07:30h)

3.2.1 Nós do N8N (Filtro de Licitações)

Descrição	Tipo	Função	Anexos

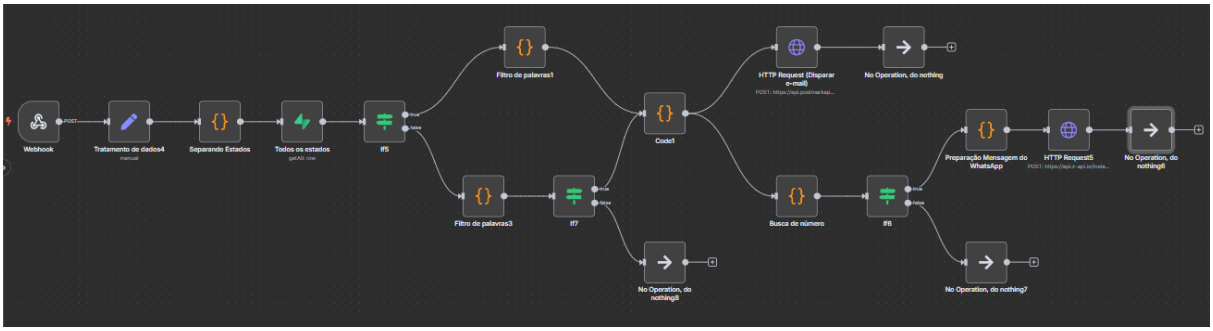
Webhook	Webhook	Recebe através de Webhook do Pipefy em modo POST informações do cliente cadastrado que receberá a notificação.	7. Ex. JSON recebido do Pipefy
Tratamento de dados4	Edit Fields	Extraí dados pertinentes para o fluxo	8. Ex. JSON com informações tratadas
Separando Estados	Code	Separa estados para variáveis únicas e cria data em formato padrão para consulta no banco de dados	9. Criando novas variáveis e tratando
Todos os estados	Supabase	Consultando o banco de dados todas as licitações com base na data atual, variáveis que foram criadas no fluxo anterior	
If5	If	Nesta parte envia o cliente dependendo do plano que ele está, se for avançado receber notificação de todos os estados, se for intermediário somente de 5 estados escolhidos.	
Filtro de palavras1	Code	Para clientes **avançados** é feito uma busca com as palavras chaves cadastradas em todos os estados.	10. Código de filtro de palavras (Avançado)
Filtro de palavras3	Code	Para clientes **intermediários** é feito uma busca com	11. Código Filtro de palavras (Intermediário)

		as palavras chaves cadastradas nos 5 estados definidos.	
If7	If	Caso não tenha oportunidades de licitações nos 5 estados, é finalizado o workflow. Se houver licitações é passado para a próxima fase	
Code1	Code	Com os resultados do nó anterior é criado váriaves (from, To, Subject, HtmlBody) que são componentes do e-mail que será disparado	12. Criando e-mail
HTTP Request (Disparar e-mail)	HTTP Request	Envio de informações para o PostMark para envio de e-mail para o cliente	
Busca de número	Code	Buscar informações nos nós iniciais como (telefone, mensagem do e-mail e aceite de notificação pelo celular/whatsApp)	13. Obtendo variáveis para envio no WhatsApp
If6	If	Verifica a condição se o cliente quer ou não quer receber a notificação pelo whatsApp. Podendo parar o fluxo ou continuar	
Preparação Mensagem do WhatsApp	Code	Preparar mensagem para o envio de notificação no WhatsApp.	14. Preparação de Mensagem WhatsApp

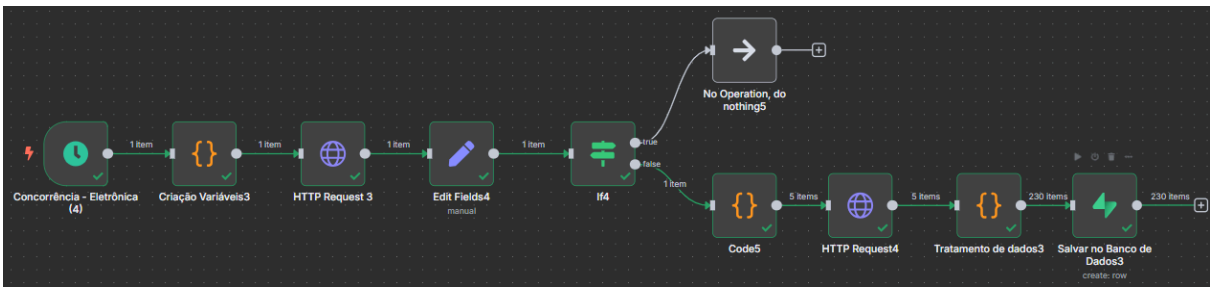
		Elencando as mais “caras”. E sendo enviadas apenas 5 oportunidades	
HTTP Request5	HTTP Request	Envio de informações para o Z-API para envio de mensagem no WhatsApp do cliente	

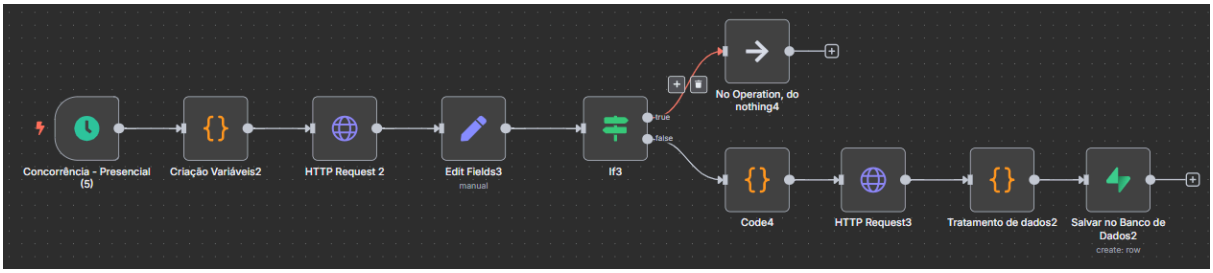
4. Fluxos n8n

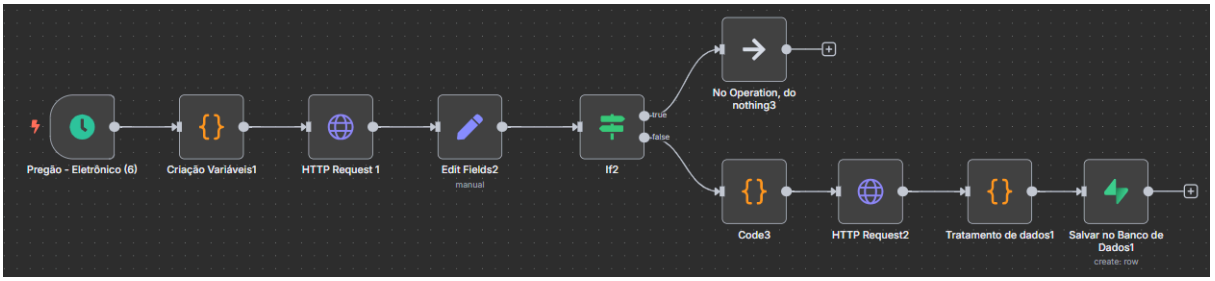
4.1. Filtro de licitações

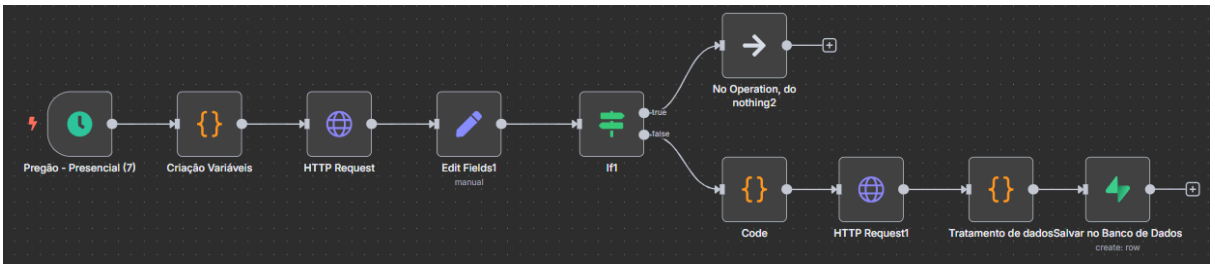


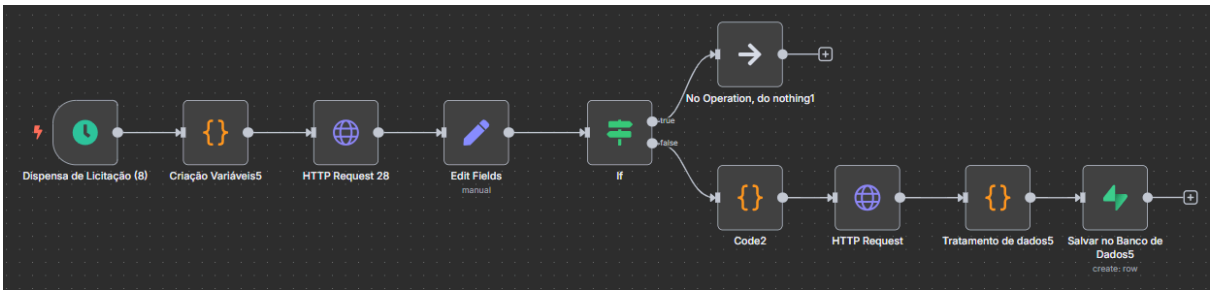
4.2. Captura de licitações











5. Tecnologias Utilizadas

Tecnologia	Função no Fluxo	Site Oficial
Pipefy	Recebe dados do cliente via Webhook (JSON).	https://www.pipefy.com
PostMark	Disparo de e-mails transacionais.	https://postmarkapp.com
VPS da Hostinger	Hospedagem do servidor onde o N8N está rodando.	https://www.hostinger.com.br
N8N	Automação do workflow (orquestração das etapas).	https://n8n.io
Z-API	Envio de mensagens automatizadas no WhatsApp.	https://www.z-api.io

Tecnologia	Função no Fluxo	Site Oficial
Supabase	Banco de dados para armazenar e consultar licitações.	https://supabase.com

6. Anexos

6.1. Criação Variáveis3

```
// Data de hoje
const hoje = new Date();

// 1. Criando a variável 'ontemFormatado' no formato aaaammdd
const ontem = new Date(hoje.getTime() - 24 * 60 * 60 * 1000);
const anoOntem = ontem.getFullYear();
const mesOntem = String(ontem.getMonth() + 1).padStart(2, '0');
const diaOntem = String(ontem.getDate()).padStart(2, '0');
const ontemFormatado = `${anoOntem}${mesOntem}${diaOntem}`;

// 2. Variável 'pagina' com valor 1
const numeroPagina = 1;

// 3. Criação da modalidade
const modalidade = 4;
// Saída no n8n
return [
  {
    json: {
      ontem: ontemFormatado,
      numeroPagina: numeroPagina,
      modalidade: modalidade
    }
  }
];
```

6.2. Ex. requisição webhook

```
https://pncp.gov.br/api/consulta/v1/contratacoes/publicacao?dataInicial={{ $json.ontem }}&dataFinal={{$json.ontem}}&codigoModalidadeContratacao={{ $json.modalidade }}&pagina=1&tamanhoPagina=50
```

6.3. Resultado do nó

```
[
  {
    "paginas": 5,
    "ontem": 20250805,
    "modalidade": 4
  }
]
```

6.4. **Criação de chamadas webhook**

```
// 1. Obter todos os dados do node anterior
const inputData = $input.all()[0].json;

// 2. Verificar se os parâmetros necessários existem
if (!inputData.paginas || !inputData.ontem || !inputData.modalidade) {
  throw new Error(`Parâmetros faltando:
    - paginas: ${inputData.paginas || 'não encontrado'}
    - ontem: ${inputData.ontem || 'não encontrado'}
    - modalidade: ${inputData.modalidade || 'não encontrado'}

    Verifique se esses parâmetros estão sendo enviados pelo node anterior.
`);
}

// 3. Array para armazenar as requisições
const requests = [];

// 4. Gerar requisições para cada página
for (let pagina = 1; pagina <= inputData.paginas; pagina++) {
  // Construir a URL com os parâmetros na ordem correta
```

```
  const url = `https://pncp.gov.br/api/consulta/v1/contratacoes/publicacao?`
+
          `dataInicial=${inputData.ontem}&` +
          `dataFinal=${inputData.ontem}&` +
          `codigoModalidadeContratacao=${inputData.modalidade}&` +
          `pagina=${pagina}&` +
          `tamanhoPagina=50`;

  requests.push({
    json: {
      url: url,
      method: 'GET',
      headers: {
        'Accept': 'application/json'
      }
    }
  });
}

// 5. Retornar as requisições
return requests
```

6.5. Código do tratamento

```
// No n8n, os dados de entrada estão disponíveis no objeto $input
const dadosOriginais = $input.all();

function formatarData(dataString) {
  if (!dataString) return null;

  try {
    const data = new Date(dataString);
    if (isNaN(data.getTime())) return null; // Verifica se é uma data válida

    const dia = String(data.getDate()).padStart(2, '0');
    const mes = String(data.getMonth() + 1).padStart(2, '0');
    const ano = data.getFullYear();
    const horas = String(data.getHours()).padStart(2, '0');
```

```
    const minutos = String(data.getMinutes()).padStart(2, '0');
    const segundos = String(data.getSeconds()).padStart(2, '0');

    return `${dia}-${mes}-${ano} ${horas}:${minutos}:${segundos}`;
  } catch (e) {
    return null;
  }
}

function transformarDados(dados) {
  if (!dados || !Array.isArray(dados)) {
    return [];
  }

  const dadosTransformados = [];

  // Processa todos os itens recebidos, não apenas o primeiro
  for (const entrada of dados) {
    if (!entrada || !entrada.json || !entrada.json.data) continue;

    const itens = entrada.json.data || [];

    for (const item of itens) {
      const novoItem = {
        orgao: item.unidadeOrgao?.nomeUnidade || null,
        portal: item.linkSistemaOrigem || null,
        estado: item.unidadeOrgao?.ufNome || null,
        municipio: item.unidadeOrgao?.municipioNome || null,
        pregao: item.processo || null,
        numeroPNCP: item.numeroControlePNCP || null,
        dataInicial: formatarData(item.dataInclusao),
        dataFinal: formatarData(item.dataAberturaProposta),
        dataPublicacao: formatarData(item.dataPublicacaoPncp),
        modalidade: item.modalidadeNome || null,
        IdModalidade: item.modalidadeId || null,
        esfera: item.orgaoEntidade?.esferaId || null,
        objeto: item.objetoCompra || null,
        valorTotal: item.valorTotalEstimado || null,
```

```
        cnpj: item.orgaoEntidade?.cnpj || null,
        razaoSocial: item.orgaoEntidade?.razaoSocial || null,
        valorHomologado: item.valorTotalHomologado || null,
        situacao: item.situacaoCompraNome || null
      };

      dadosTransformados.push(novoItem);
    }
  }

  return dadosTransformados;
}

// Processa todos os dados recebidos
const resultado = transformarDados(dadosOriginais);

// Retorna os dados transformados para o n8n
return resultado.map(item => ({ json: item }));
```

6.6. Ex. resultado JSON de registro


```
[
  {
    "id": 88045,
    "orgao": "Prefeitura Municipal de Palmeira",
    "portal": null,
    "estado": "Santa Catarina",
    "municipio": "Palmeira",
    "pregao": "CC_07/2025",
    "numeroPncp": "01610566000106-1-000052/2025",
    "dtInicial": "05-08-2025 00:00:08",
    "dtFinal": "04-08-2025 19:00:01",
    "dtPublicacao": "05-08-2025 00:00:08",
    "modalidade": "Concorrência - Eletrônica",
    "esfera": "M",
    "objeto": "CONTRATAÇÃO DE EMPRESA ESPECIALIZADA EM CONSTRU
ÇÃO DE QUIOSQUE DA PRAÇA FEIRA DE AGRICULTURA FAMILIAR CONFO
RME PROJETOS, MEMORIAL DESCRITIVO, PLANILHA ORÇAMENTÁRIA, CR
```

ONOGRAMA FÍSICO FINANCEIRO, ART E DEMAIS DOCUMENTOS COMPLE
MENTARES, PARTE INTEGRANTE DO EDITAL.",
"valorTotal": "133774.26",
"cnpj": 1610566000106,
"razaoSocial": "MUNICIPIO DE PALMEIRA",
"valorHomologado": null,
"situacao": null,
"created_at": "2025-08-06T19:52:59.080671+00:00",
"modalidadeId": 4
},
{
"id": 88046,
"orgao": "ESCOLA DE ESPECIALISTAS DE AERONAUTICA",
"portal": "https://cnetmobile.estaleiro.serpro.gov.br/comprasnet-web/pu
blic/landing?destino=acompanhamento-compra&compra=1200640390003
2025",
"estado": "São Paulo",
"municipio": "Guaratinguetá",
"pregao": "67540004723202427",
"numeroPncp": "00394429000100-1-001834/2025",
"dtInicial": "05-08-2025 07:01:29",
"dtFinal": "05-08-2025 08:00:00",
"dtPublicacao": "05-08-2025 07:01:29",
"modalidade": "Concorrência - Eletrônica",
"esfera": "F",
"objeto": "Contratação de Obra de Engenharia, para REFORMA DO ESTA
NDE DE TIRO DA GUARNAE-GW.",
"valorTotal": "994850.73",
"cnpj": 394429000100,
"razaoSocial": "COMANDO DA AERONAUTICA",
"valorHomologado": null,
"situacao": null,
"created_at": "2025-08-06T19:52:59.080671+00:00",
"modalidadeId": 4
},
{
"id": 88047,
"orgao": "DEFENSORIA PÚBLICA GERAL DO ESTADO DO ACRE",

```
    "portal": "https://cnetmobile.estaleiro.serpro.gov.br/comprasnet-web/public/landing?destino=acompanhamento-compra&compra=459931039000120 25",
    "estado": "Acre",
    "municipio": "Rio Branco",
    "pregao": "03021/2025-74",
    "numeroPncp": "04581375000143-1-000010/2025",
    "dtInicial": "05-08-2025 07:01:37",
    "dtFinal": "05-08-2025 08:00:00",
    "dtPublicacao": "05-08-2025 07:01:37",
    "modalidade": "Concorrência - Eletrônica",
    "esfera": "E",
    "objeto": "Contratação de empresa de engenharia e para a obra de construção da unidade da Defensoria Pública-Geral do Estado do Acre, no município de Capixaba, subsidiada através da emenda Parlamentar nº. 73000 e contrapartida com recursos próprios nos termos da tabela abaixo, conforme condições e exigências estabelecidas neste instrumento e seus anexos.",
    "valorTotal": "620079.34",
    "cnpj": 4581375000143,
    "razaoSocial": "DEFENSORIA PUBLICA -GERAL DO ESTADO DO ACRE",
    "valorHomologado": null,
    "situacao": null,
    "created_at": "2025-08-06T19:52:59.080671+00:00",
    "modalidadeId": 4
  }
]
```


6.7. Ex. JSON recebido do Pipefy


```
[
  {
    "headers": {
      "host": "n8n-n8n-start.13cqhc.easypanel.host",
      "user-agent": "Pipefy-UserAgent",
      "content-length": "356",
      "accept": "application/json",
      "content-type": "application/json",
      "elastic-apm-traceparent": "00-e15629c2772b5c900f1ccebe56c38b37
```

```
-25e0b282223b921a-00",
     "traceparent": "00-e15629c2772b5c900f1ccebe56c38b37-25e0b2822
23b921a-00",
     "tracestate": "es=s:0.0",
     "x-forwarded-for": "129.153.135.244",
     "x-forwarded-host": "n8n-n8n-start.13cqhc.easypanel.host",
     "x-forwarded-port": "443",
     "x-forwarded-proto": "https",
     "x-forwarded-server": "b0a51531dac1",
     "x-real-ip": "129.153.135.244",
     "accept-encoding": "gzip"
   },
   "params": {},
   "query": {},
   "body": {
     "Empresa": "Patrícia Mannes de Liz",
     "CNPJ": "84.018.375/0001-79",
     "Email": "patymannes89@gmail.com",
     "Plano": "Avançado (Todos os estados)",
     "NotificaWhats": "Sim",
     "NumeroWhats": "+55 47 99654-6459",
     "Estados": "",
     "Palavrachave1": "Papelaria",
     "Palavrachave2": "",
     "Palavrachave3": "",
     "Palavrachave4": "",
     "Palavrachave5": ""
   },
   "webhookUrl": "https://n8n-n8n-start.13cqhc.easypanel.host/webhook/c
d80f09b-b614-4453-9b17-8fee176de1df",
   "executionMode": "production"
 }
]
```


6.8. Ex. JSON com informações tratadas


```
[
 {
```

```
  "body": {
    "Empresa": "Patrícia Mannes de Liz",
    "CNPJ": "84.018.375/0001-79",
    "Email": "patymannes89@gmail.com",
    "Plano": "Avançado (Todos os estados)",
    "Estados": "",
    "Palavrachave1": "Papelaria",
    "Palavrachave2": "",
    "Palavrachave3": "",
    "Palavrachave4": "",
    "Palavrachave5": ""
  },
  "NotificaçãoWhats": "Sim",
  "NumeroWhats": "+55 47 99654-6459"
 }
]
```

6.9. Criando novas variáveis e tratando

```
const dados = items[0].json.body;

const estadosArray = dados.Estados.split(',').map(e => e.trim());

const resultado = {};

// Cria dinamicamente as variáveis estado1, estado2, etc.
estadosArray.forEach((estado, index) => {
 resultado[`estado${index + 1}`] = estado;
});

// --- Cálculo da data de ontem ---
const hoje = new Date(); // Data atual
const ontem = new Date(hoje);

ontem.setDate(hoje.getDate() - 1); // Subtrai 1 dia

// Formatação manual para garantir "DD-MM-YYYY 00:00:00"
const dia = String(ontem.getDate()).padStart(2, '0');
```

```
const mes = String(ontem.getMonth() + 1).padStart(2, '0');
const ano = ontem.getFullYear();

// Início do dia (00:00:00)
resultado.dataOntemInicio = `${dia}-${mes}-${ano} 00:00:00`;

// Fim do dia (23:59:59)
resultado.dataOntemFim = `${dia}-${mes}-${ano} 23:59:59`;

// Adiciona as novas variáveis
resultado.NotificacaoWhats = items[0].json["NotificaçãoWhats"]; // Manté
m o valor original
resultado.NomeroWhats = items[0].json["NumeroWhats"].replace(/\D/g, '');
// Remove tudo que não for dígito

// Retorna os dados originais + estados + datas + novas variáveis
return [
  {
    json: {
      ...dados,
      ...resultado,
    }
  }
];
```

6.10. Código de filtro de palavras (Avançado)

```
// Configuração das palavras-chave (modifique conforme necessidade)
const rawKeywords = [
  $('Tratamento de dados4').first().json.body.Palavrachave1,
  $('Tratamento de dados4').first().json.body.Palavrachave2,
  $('Tratamento de dados4').first().json.body.Palavrachave3,
  $('Tratamento de dados4').first().json.body.Palavrachave4,
  $('Tratamento de dados4').first().json.body.Palavrachave5,
];

// Filtra apenas as palavras-chave que não são vazias ou nulas
const keywords = rawKeywords.filter(keyword =>
```

```
  keyword !== null && keyword !== undefined && keyword.toString().trim() !
== ''
);

// Função para normalizar texto (remove acentos e padroniza)
function normalizeText(text) {
  if (!text) return '';
  return text.toString()
    .normalize("NFD").replace(/[\u0300-\u036f]/g, "")
    .toLowerCase()
    .replace(/\s+/g, ' ');
}

// Processamento principal
try {
  let filteredItems = [];

  // Se não houver palavras-chave válidas, retorna todos os itens
  if (keywords.length === 0) {
    return items.map(item => ({
      json: item.json,
      pairedItem: item.pairedItem
    }));
  }

  // Verifica se existem items para processar
  if (items && items.length > 0) {
    filteredItems = items.filter(item => {
      if (!item.json || !item.json.objeto) return false;

      const normalizedSearch = normalizeText(item.json.objeto);

      // Verifica correspondência com pelo menos uma palavra-chave válida
      return keywords.some(keyword => {
        const normalizedKeyword = normalizeText(keyword);
        return normalizedKeyword && normalizedSearch.includes(normalizedK
eyword);
      });
```

```
    });
  }

  // Retorna no formato esperado pelo n8n
  return filteredItems.map(item => {
    return {
      json: item.json,
      pairedItem: item.pairedItem
    };
  });

} catch (error) {
  // Log de erro para debug
  console.error('Erro no processamento:', error);
  return [];
}
```

6.11. Código Filtro de palavras (Intermediário)

```
// Configuração das palavras-chave (modifique conforme necessidade)
const rawKeywords = [
  $('Tratamento de dados4').first().json.body.Palavrachave1,
  $('Tratamento de dados4').first().json.body.Palavrachave2,
  $('Tratamento de dados4').first().json.body.Palavrachave3,
  $('Tratamento de dados4').first().json.body.Palavrachave4,
  $('Tratamento de dados4').first().json.body.Palavrachave5,
];

// Filtra apenas as palavras-chave que não são vazias ou nulas
const keywords = rawKeywords.filter(keyword =>
  keyword !== null && keyword !== undefined && keyword.toString().trim() !
== ''
);

// Função para normalizar texto (remove acentos e padroniza)
function normalizeText(text) {
  if (!text) return '';
  return text.toString()
```

```
    .normalize("NFD").replace(/[\u0300-\u036f]/g, "")
    .toLowerCase()
    .replace(/\s+/g, ' ');
}

// Processamento principal
try {
  let filteredItems = [];

  // Se não houver palavras-chave válidas, retorna todos os itens
  if (keywords.length === 0) {
    return items.map(item => ({
      json: item.json,
      pairedItem: item.pairedItem
    }));
  }

  // Verifica se existem items para processar
  if (items && items.length > 0) {
    filteredItems = items.filter(item => {
      if (!item.json || !item.json.objeto) return false;

      const normalizedSearch = normalizeText(item.json.objeto);

      // Verifica correspondência com pelo menos uma palavra-chave válida
      return keywords.some(keyword => {
        const normalizedKeyword = normalizeText(keyword);
        return normalizedKeyword && normalizedSearch.includes(normalizedK
eyword);
      });
    });
  }

  // Retorna no formato esperado pelo n8n
  return filteredItems.map(item => {
    return {
      json: item.json,
      pairedItem: item.pairedItem
```

```
    };
  });

} catch (error) {
  // Log de erro para debug
  console.error('Erro no processamento:', error);
  return [];
}
```

6.12. Criando e-mail

```
// Código para o nó de JavaScript no N8N
// Certifique-se de que os dados estão sendo passados corretamente
// Se você está usando o modo "Run Once" para testes, mude para "Run fo
r Each Item"

// Obter todos os itens
const items = $input.all();

// Verifique se os itens estão sendo capturados corretamente
if (!items || items.length === 0) {
  throw new Error('Nenhum dado de licitação recebido');
}

// Obter palavras-chave
const palavrasChave = [
  $('Tratamento de dados4').first().json.body.Palavrachave1,
  $('Tratamento de dados4').first().json.body.Palavrachave2,
  $('Tratamento de dados4').first().json.body.Palavrachave3,
  $('Tratamento de dados4').first().json.body.Palavrachave4,
  $('Tratamento de dados4').first().json.body.Palavrachave5
].filter(palavra => palavra && palavra.trim() !== '');

// Função para destacar palavras-chave no texto
function destacarPalavrasChave(texto) {
  if (!texto || typeof texto !== 'string') return texto;

  let resultado = texto;
```

```
  palavrasChave.forEach(palavra => {
    const regex = new RegExp(`(${palavra.replace(/[-\/\\^$*+?.()|[\]{}]/g,
'\\$&')})`, 'gi');
    resultado = resultado.replace(regex, '<span style="background-color: #
CEF09D; padding: 2px 4px; border-radius: 3px; font-weight: bold;">$1</sp
an>');
  });

  return resultado;
}

// Função para converter string monetária para número
function parseCurrency(value) {
  if (!value || value === 'null' || value === 'undefined') return 0;

  try {
    // Remove todos os caracteres não numéricos exceto vírgula e ponto
    const cleaned = String(value).replace(/[^\d,.-]/g, '');

    // Verifica se tem vírgula como separador decimal (formato brasileiro)
    if (/,/.test(cleaned)) {
      // Remove pontos de milhar e converte vírgula decimal para ponto
      return parseFloat(cleaned.replace(/\./g, '').replace(',', '.'));
    }
    // Se tem ponto como separador decimal (formato internacional)
    return parseFloat(cleaned);
  } catch (e) {
    console.error('Erro ao converter valor:', value, e);
    return 0;
  }
}

// Função para formatar valores monetários
function formatCurrency(value) {
  if (value === null || value === undefined || value === 'null' || value === 'u
ndefined') return 'Não informado';

  try {
```

```
    const num = typeof value === 'number' ? value : parseCurrency(value);
    return num.toLocaleString('pt-BR', {
      style: 'currency',
      currency: 'BRL',
      minimumFractionDigits: 2,
      maximumFractionDigits: 2
    });
  } catch (e) {
    console.error('Erro ao formatar valor:', value, e);
    return 'Não informado';
  }
}

// Função para formatar datas
function formatDate(dateString) {
  if (!dateString || dateString === 'null' || dateString === 'undefined') return
'Não informada';

  // Verifica se já está no formato DD-MM-YYYY
  if (/^\d{2}-\d{2}-\d{4}/.test(dateString)) {
    return dateString;
  }

  try {
    const date = new Date(dateString);
    return date.toLocaleDateString('pt-BR');
  } catch (e) {
    return dateString;
  }
}

// Calcular valor total estimado CORRIGIDO
function calcularValorTotal() {
  let total = 0;
  items.forEach(item => {
    const valor = parseCurrency(item.json.valorTotal);
    if (!isNaN(valor)) {
      total += valor;
```

```
    }
  });
  return total;
}

const valorTotalEstimado = calcularValorTotal();

// Construir o HTML do e-mail
let html = `
<!DOCTYPE html>
<html lang="pt-BR">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.
0">
    <title>Licitações Diárias</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            line-height: 1.6;
            color: #002939;
            max-width: 1000px;
            margin: 0 auto;
            padding: 20px;
            background-color: #f9f9f9;
        }
        h1 {
            color: #002939;
            text-align: center;
            border-bottom: 2px solid #b0d9e6;
            padding-bottom: 12px;
            margin-bottom: 25px;
            background-color: white;
            padding: 20px;
            border-radius: 8px;
            box-shadow: 0 1px 3px rgba(0,41,57,0.05);
        }
        h2 {
```

```
        color: #226677;
        margin-top: 30px;
        border-left: 2px solid #b0d9e6;
        padding-left: 12px;
        background-color: rgba(176,217,230,0.2);
        padding: 8px 12px;
        border-radius: 0 8px 8px 0;
    }
    .licitacao {
        background-color: white;
        border: 1px solid #e2e2e2;
        border-radius: 8px;
        padding: 20px;
        margin-bottom: 20px;
        box-shadow: 0 1px 4px rgba(0,41,57,0.03);
        transition: transform 0.2s, box-shadow 0.2s;
    }
    .licitacao:hover {
        transform: translateY(-2px);
        box-shadow: 0 2px 8px rgba(0,41,57,0.08);
    }
    .licitacao h3 {
        color: #002939;
        margin-top: 0;
        border-bottom: 1px solid #e2e2e2;
        padding-bottom: 8px;
    }
    .detalhes {
        display: grid;
        grid-template-columns: repeat(auto-fill, minmax(300px, 1fr));
        gap: 12px;
        margin-top: 15px;
    }
    .detalhe-item {
        margin-bottom: 10px;
    }
    .detalhe-item strong {
        display: inline-block;
```

```
        width: 150px;
        color: #226677;
    }
    .portal-link {
        display: inline-block;
        background-color: #5db7de;
        color: white;
        padding: 10px 20px;
        text-decoration: none;
        border-radius: 6px;
        margin-top: 15px;
        font-weight: bold;
        transition: background-color 0.3s;
    }
    .portal-link:hover {
        background-color: #226677;
    }
    .resumo {
        background-color: white;
        padding: 20px;
        border-radius: 8px;
        margin-bottom: 30px;
        border-left: 2px solid #b0d9e6;
        box-shadow: 0 1px 4px rgba(0,41,57,0.03);
    }
    .resumo p {
        font-size: 1.1em;
        margin: 12px 0;
    }
    .resumo strong {
        color: #002939;
    }
    .footer {
        text-align: center;
        margin-top: 40px;
        padding-top: 20px;
        border-top: 1px solid #e2e2e2;
        color: #226677;
```

```
            font-size: 0.9em;
        }
        .estado-titulo {
            font-size: 1.4em;
            font-weight: 600;
            color: white;
            background-color: #226677;
            padding: 10px 15px;
            border-radius: 6px;
            margin: 25px 0 15px 0;
            display: inline-block;
        }
        @media (max-width: 600px) {
            .detalhes {
                grid-template-columns: 1fr;
            }
            .detalhe-item strong {
                width: 120px;
            }
            .licitacao {
                padding: 15px;
            }
        }
    </style>
</head>
<body>
    <h1>Filtro de Licitações - Dautin</h1>

    <div class="resumo">
        <h2>Resumo do Dia</h2>
        <p>Total de licitações encontradas: <strong>${items.length}</strong></p>
        <p>Valor total estimado: <strong>${formatCurrency(valorTotalEstimado)}</strong></p>
        ${palavrasChave.length > 0 ? `<p>Palavras-chave destacadas: <strong>${palavrasChave.join(', ')}</strong></p>` : ''}
    </div>
```

```
    <h2>Detalhes das Licitações</h2>
`;

// Agrupar por estado
const licitacoesPorEstado = {};
items.forEach(item => {
    const estado = item.json.estado || 'Não informado';
    if (!licitacoesPorEstado[estado]) {
        licitacoesPorEstado[estado] = [];
    }
    licitacoesPorEstado[estado].push(item.json);
});

// Adicionar licitações agrupadas por estado ao HTML
for (const estado in licitacoesPorEstado) {
    html += `<h3 class="estado-titulo">Estado: ${estado}</h3>`;

    licitacoesPorEstado[estado].forEach(licitacao => {
        // Gerar link PNCP automaticamente
        let portalLink = '';
        const numeroPncp = licitacao.numeroPncp;

        if (numeroPncp) {
            try {
                // Formatar o número PNCP (ex: 63761985000198-1-000052/202
5 → 63761985000198/2025/000052)
                const partes = numeroPncp.split('-');
                const cnpj = partes[0];
                const sequencial = partes[2].split('/')[0];
                const ano = partes[2].split('/')[1];

                const linkPncp = `https://pncp.gov.br/app/editais/${cnpj}/${ano}/
${sequencial}`;
                portalLink = `<a href="${linkPncp}" class="portal-link" target="_b
lank">Acessar Portal PNCP</a>`;
            } catch (e) {
                if (licitacao.portal) {
                    portalLink = `<a href="${licitacao.portal}" class="portal-link" ta
```

```
rget="_blank">Acessar Portal</a>`;
            }
          }
      } else if (licitacao.portal) {
          portalLink = `<a href="${licitacao.portal}" class="portal-link" target
="_blank">Acessar Portal</a>`;
      }

      // Aplicar destaque às palavras-chave no objeto da licitação
      const objetoDestacado = destacarPalavrasChave(licitacao.objeto || 'N
ão informado');

      html += `
      <div class="licitacao">
          <h3>${licitacao.orgao || 'Não informado'}</h3>
          <p><strong>Objeto:</strong> ${objetoDestacado}</p>

          <div class="detalhes">
              <div class="detalhe-item"><strong>Município:</strong> ${licitac
ao.municipio || 'Não informado'}</div>
              <div class="detalhe-item"><strong>Modalidade:</strong> ${licit
acao.modalidade || 'Não informado'}</div>
              <div class="detalhe-item"><strong>Número PNCP:</strong> ${n
umeroPncp || 'Não informado'}</div>
              <div class="detalhe-item"><strong>Pregão/Processo:</strong>
${licitacao.pregao || 'Não informado'}</div>
              <div class="detalhe-item"><strong>Data Publicação:</strong>
${formatDate(licitacao.dtPublicacao)}</div>
              <div class="detalhe-item"><strong>Valor Total:</strong> ${form
atCurrency(licitacao.valorTotal)}</div>
              <div class="detalhe-item"><strong>CNPJ:</strong> ${licitacao.c
npj || 'Não informado'}</div>
              <div class="detalhe-item"><strong>Razão Social:</strong> ${lici
tacao.razaoSocial || 'Não informado'}</div>
              <div class="detalhe-item"><strong>Portal:</strong> ${licitacao.
portal ? `<a href="${licitacao.portal}" target="_blank">${licitacao.portal}</
a>` : 'Não informado'}</div>
          </div>
```

```
        ${portalLink}
      </div>
      `;
   });
}

html += `
   <div class="footer">
      <div style="background-color: #e8f4fc; padding: 20px; border-radius: 8px; margin-bottom: 20px; border-left: 4px solid #5db7de;">
         <h3 style="color: #002939; margin-top: 0;">Garanta a Integridade dos Seus Documentos</h3>
         <p style="margin-bottom: 15px;">A <strong>Dautin Blockchain</strong> oferece registro imutável de documentos para processos licitatórios:</p>

         <div style="margin-bottom: 15px;">
            <p style="margin: 8px 0;"><strong>✓ Proteção contra alterações indevidas</strong></p>
            <p style="margin: 8px 0;"><strong>✓ Certificação de autenticidade</strong></p>
            <p style="margin: 8px 0;"><strong>✓ Validade jurídica</strong></p>
            <p style="margin: 8px 0;"><strong>✓ Registro instantâneo e seguro</strong></p>
         </div>

         <a href="https://www.dautin.com"
            style="display: inline-block; background-color: #002939; color: white; padding: 10px 20px; text-decoration: none; border-radius: 6px; margin-top: 5px; font-weight: bold;"
            target="_blank">
             Conheça nossas soluções
         </a>
      </div>

      <p style="margin-top: 20px; font-size: 0.9em; color: #226677;">
```

```
        Este e-mail foi gerado automaticamente com base nas licitações pu
blicadas.<br>
        © ${new Date().getFullYear()} - Todos os direitos reservados
      </p>
    </div>
</body>
</html>
`;

// Construir o payload completo para a API do Postmark
const postmarkPayload = {
  "From": "dautin@dautin.com",
  "To": $('Separando Estados').first().json.Email,
  "Subject": "🔎 Filtro Diário - Editais de Licitações",
  "HtmlBody": html,
  "MessageStream": "licitacoes-notifier"
};

// Retornar o payload JSON completo para ser usado no nó de e-mail
return [{ json: postmarkPayload }];
```

6.13. Obtendo variáveis para envio no WhatsApp

```
// 1. Obter todos os inputs de forma segura
const rawInput = $input.all();

// 2. Debug - ver a estrutura completa que está chegando
console.log('Input completo recebido:', JSON.stringify(rawInput, null, 2));

// 3. Verificar se temos dados válidos
if (!rawInput || rawInput.length === 0 || !rawInput[0].json) {
  throw new Error('Nenhum dado válido recebido');
}

// 4. Pegar o primeiro item (ajuste se precisar processar outro índice)
const dados = rawInput[0].json;

// 5. Versão segura que não quebra se os campos não existirem
```

```
const resultado = {
  NotificacaoWhats: dados.NotificacaoWhats || 'Campo não encontrado',
  NomeroWhats: dados.NomeroWhats || 'Campo não encontrado',
  HtmlBody: dados.HtmlBody || 'Campo não encontrato',

  // Debug adicional - mostra todas as chaves disponíveis
  camposDisponiveis: Object.keys(dados)
};

// 6. Mostrar o resultado no log
console.log('Resultado processado:', JSON.stringify(resultado, null, 2));

// 7. Retornar apenas os campos desejados (remova esta linha se quiser o debug completo)
return {
  NotificacaoWhats: $('Separando Estados').first().json.NotificacaoWhats || null,
  NomeroWhats: $('Separando Estados').first().json.NomeroWhats || null,
  HtmlBody: $input.first().json.HtmlBody || null
};
```

6.14. Preparação de Mensagem WhatsApp

```
/**
 * Extrai informações de licitações de uma string HTML, associando cada licitação ao seu estado correto.
 * @param {string} htmlString A string HTML contendo os dados das licitações.
 * @returns {Array<Object>} Um array de objetos, onde cada objeto representa uma licitação com suas propriedades.
 */
function extractLicitacoesFromHtml(htmlString) {
    const licitacoes = [];
    // Mapeamento de nome de estado por extenso para sigla
    const estadoSiglas = {
        'minas gerais': 'MG',
        'são paulo': 'SP',
        'rio de janeiro': 'RJ',
```

```
        'bahia': 'BA',
        'rio grande do sul': 'RS',
        'santa catarina': 'SC',
        'paraná': 'PR',
        'goiás': 'GO',
        'distrito federal': 'DF',
        'ceará': 'CE',
        'pernambuco': 'PE',
        'amazonas': 'AM',
        'pará': 'PA',
        'espírito santo': 'ES',
        'mato grosso': 'MT',
        'mato grosso do sul': 'MS',
        'rondônia': 'RO',
        'tocantins': 'TO',
        'acre': 'AC',
        'roraima': 'RR',
        'amapá': 'AP',
        'piauí': 'PI',
        'maranhão': 'MA',
        'rio grande do norte': 'RN',
        'paraíba': 'PB',
        'alagoas': 'AL',
        'sergipe': 'SE',
    };

    // Expressão regular para encontrar tanto os títulos de estado quanto os blocos de licitação.
    const regex = /(<h3 class="estado-titulo">Estado:\s*.*?<\/h3>|<div class="licitacao">.*?<\/div>\s*(?=<h3 class="estado-titulo">|<div class="licitacao">|$))/gs;
    let match;
    let currentEstadoSigla = '';

    // Itera sobre todos os matches encontrados na string HTML
    while ((match = regex.exec(htmlString)) !== null) {
        const block = match[0];
```

```
        if (block.startsWith('<h3 class="estado-titulo">')) {
            // Se o bloco é um título de estado, extrai e atualiza a sigla do estad
o atual
            const estadoTituloMatch = block.match(/<h3 class="estado-titulo">
Estado:\s*(.*?)<\/h3>/);
            if (estadoTituloMatch && estadoTituloMatch[1]) {
                const estadoNomeCompleto = estadoTituloMatch[1].trim().toLowe
rCase();
                currentEstadoSigla = estadoSiglas[estadoNomeCompleto] || '';
            }
        } else if (block.startsWith('<div class="licitacao">')) {
            // Se o bloco é uma licitação, extrai suas informações
            let objeto = '';
            let valorText = '';
            let link = '';
            let municipio = 'Não informado';

            const objetoMatch = block.match(/<p><strong>Objeto:<\/strong>(.
*?)<\/p>/s);
            if (objetoMatch && objetoMatch[1]) {
                objeto = objetoMatch[1].trim();
            }

            const valorMatch = block.match(/<strong>Valor Total:<\/strong>([^
<]+?)<\/div>/);
            if (valorMatch && valorMatch[1]) {
                valorText = valorMatch[1].trim();
            }

            const municipioMatch = block.match(/<strong>Município:<\/strong
>(.*?)<\/div>/s);
            if (municipioMatch && municipioMatch[1]) {
                municipio = municipioMatch[1].trim();
            }

            const linkMatch = block.match(/<a href="(.*?)" class="portal-lin
k"/);
            if (linkMatch && linkMatch[1]) {
```

```
            link = linkMatch[1].trim();
        }

        if (objeto && link) {
            let valor = null;
            if (valorText) {
                const parsedValue = parseFloat(valorText.replace('R$', '').repla
ce(/\./g, '').replace(',', '.').trim());
                if (!isNaN(parsedValue)) {
                    valor = parsedValue;
                }
            }

            // Adiciona a licitação com a sigla do estado atual
            licitacoes.push({
                objeto: objeto,
                valor: valor,
                link: link,
                municipio: municipio,
                estadoSigla: currentEstadoSigla
            });
        }
    }
  }
  return licitacoes;
}

// Obtém as palavras-chave das variáveis especificadas
const palavrasChave = [
  $('Tratamento de dados4').first().json.body.Palavrachave1,
  $('Tratamento de dados4').first().json.body.Palavrachave2,
  $('Tratamento de dados4').first().json.body.Palavrachave3,
  $('Tratamento de dados4').first().json.body.Palavrachave4,
  $('Tratamento de dados4').first().json.body.Palavrachave5
].filter(palavra => palavra && palavra.trim() !== '');

// Assume que $input.item.json.HtmlBody contém a string HTML fornecida
const htmlContent = $input.item.json.HtmlBody;
```

```
// Extrai as licitações do conteúdo HTML
const licitacoesExtraidas = extractLicitacoesFromHtml(htmlContent);

// Ordena as licitações por valor total em ordem decrescente
licitacoesExtraidas.sort((a, b) => {
    const valorA = a.valor !== null ? a.valor : -Infinity;
    const valorB = b.valor !== null ? b.valor : -Infinity;
    return valorB - valorA;
});

// Pega as top 5 licitações
const top5Licitacoes = licitacoesExtraidas.slice(0, 5);

// Calcula o total de licitações para a mensagem, adicionando 1 como solici
tado
const totalLicitacoesParaMensagem = licitacoesExtraidas.length + 1;

let whatsappMessage = "*🔎 Filtro Diário - Editais de Licitações - Dautin*
\n\n";

if (top5Licitacoes.length === 0) {
    whatsappMessage = "Olá! Não encontramos novas licitações de alto val
or hoje. Fique atento ao seu e-mail para mais atualizações!";
} else {
    // Regex para encontrar as palavras-chave (case insensitive)
    const regexPalavrasChave = palavrasChave.length > 0
        ? new RegExp(`(${palavrasChave.join('|')})`, 'gi')
        : null;

    top5Licitacoes.forEach((licitacao, index) => {
        // Remove tags HTML
        let objetoTexto = licitacao.objeto.replace(/<[^>]*>/g, '');

        // Destaca palavras-chave em negrito se existirem
        if (regexPalavrasChave) {
            objetoTexto = objetoTexto.replace(regexPalavrasChave, '*$1*');
        }
```

```
        const local = licitacao.estadoSigla ? `${licitacao.municipio} - ${licitacao.estadoSigla}` : licitacao.municipio;

        whatsappMessage += `${index + 1}. *Local:* ${local}\n`;
        const valorFormatado = licitacao.valor !== null
          ? `R$ ${licitacao.valor.toLocaleString('pt-BR', { minimumFractionDigits: 2, maximumFractionDigits: 2 })}`
          : 'Não informado';
        whatsappMessage += `    *Valor Total:* ${valorFormatado}\n`;
        whatsappMessage += `    *Link:* ${licitacao.link}\n`;
        whatsappMessage += `    *Objeto:* ${objetoTexto}\n`;

        if (index < top5Licitacoes.length - 1) {
          whatsappMessage += "\n";
        }
    });

    if (licitacoesExtraidas.length > 5) {
        whatsappMessage += `\n*📧 Para ter acesso a todas as ${totalLicitacoesParaMensagem} licitações, acesse seu e-mail.*`;
    } else if (licitacoesExtraidas.length > 0) {
        whatsappMessage += `\n*📧 Para ver todas as ${totalLicitacoesParaMensagem} licitações disponíveis, confira seu e-mail.*`;
    }
}

// Retorna o resultado para o próximo nó
return [{ json: { whatsappMessage: whatsappMessage } }];
```

1. Criação Variáveis3

2. Ex. requisição webhook

3. Resultado do nó

4. Criação de chamadas webhook

5. Código do tratamento