

## **APÊNDICE A - WORKFLOW DE HIDROGRAFIA E ALTIMETRIA**

O objetivo do processo de correção de altimetria e hidrografia é identificar e corrigir erros comuns em dados geoespaciais por meio de 18 rotinas, sendo 17 de identificação e uma de manipulação para gerar as áreas com delimitadores e centroide após as correções topológicas, garantindo a execução correta.

As etapas iniciais do fluxo visam identificar geometrias inválidas, como vértices não compartilhados em interseções, sobreposições de feições e geometrias com topologia incorreta, entre outros erros que podem surgir devido a problemas na coleta, digitalização ou processamento de dados geoespaciais. A correção desses erros é essencial para evitar falhas na execução das etapas seguintes do processo de identificação de erros, tornando a etapa de identificação de geometrias inválidas uma prioridade na correção.

Também é importante ressaltar que existem modelos que demandam a execução paralela de etapas no fluxo de trabalho. Essa necessidade de execução paralela levou à decisão de separar essas rotinas, devido à sua frequente execução. A compilação de todas essas rotinas em um único fluxo de trabalho acarretaria a execução de todas as rotinas presentes, o que resultaria em atrasos no processo de validação.

Por esse motivo que neste apêndice também estão incluídas rotinas que não estão presentes no *workflow* de validação, mas que também possuem sua importância.

# Sumário

1. IDENTIFICAR GEOMETRIAS INVÁLIDAS.....	3
2. IDENTIFICAR GEOMETRIAS COM MAIS DE UMA PARTE.....	7
3. IDENTIFICAR FEIÇÕES DUPLICADAS.....	12
4. IDENTIFICAR VÉRTICE NÃO COMPARTILHADO NAS INTERSECÇÕES.....	16
6. IDENTIFICAR VÉRTICE PRÓXIMO DE ARESTA.....	25
8. IDENTIFICAR ÂNGULOS PEQUENOS.....	33
9. IDENTIFICAR ÂNGULOS PEQUENOS ENTRE CAMADAS.....	36
10. IDENTIFICAR ÂNGULOS EM Z.....	40
11. IDENTIFICAR OVERLAPS DENTRO DA MESMA CAMADA.....	43
12. IDENTIFICAR <i>UNDERSHOOT</i> COM MOLDURA E CONEXÃO DE LINHAS.....	47
13. IDENTIFICAR LINHAS SEGMENTADAS COM MESMO CONJUNTO DE ATRIBUTOS.....	53
14. IDENTIFICAR LINHAS NÃO SEGMENTADAS NAS INTERSECÇÕES.....	57
15. IDENTIFICAR ELEMENTOS PEQUENOS NA REDE.....	60
16. IDENTIFICAR ERROS NA CONSTRUÇÃO DA REDE DE DRENAGEM.....	67
17. IDENTIFICAR ERROS NA CONSTRUÇÃO DAS CURVAS DE NÍVEL.....	74
18. FECHAR POLÍGONOS DE MASSA D'ÁGUA.....	80
19. MODELOS EXECUTADOS FORA DO <i>WORKFLOW</i> .....	84
19.1. ROTINAS DE VALIDADE GEOMÉTRICA.....	85
19.2. ROTINAS DE VALIDADE DE VÉRTICES.....	87
19.3. IDENTIFICA PONTAS SOLTAS EM DELIMITADORES DE CORPOS DE ÁGUA.....	91
19.4. IDENTIFICA PONTAS SOLTAS EM DELIMITADORES DE ELEMENTOS HIDROGRÁFICOS.....	95
19.5. FECHAR ELEMENTOS HIDROGRÁFICOS E ILHAS.....	97

## 1. IDENTIFICAR GEOMETRIAS INVÁLIDAS

**a) Descrição:** Este processo identifica geometrias inválidas nas camadas especificadas.

**b) Arquivo:** identifica\_geometrias\_invalidas\_alt\_hid\_carta\_orto.model3.

**c) Algoritmos:**

```
dsgtools:batchrunalgorithm;  
dsgtools:identifyandfixinvalidgeometries;  
native:mergevectorlayers.
```

**d) Composição do Modelo:**

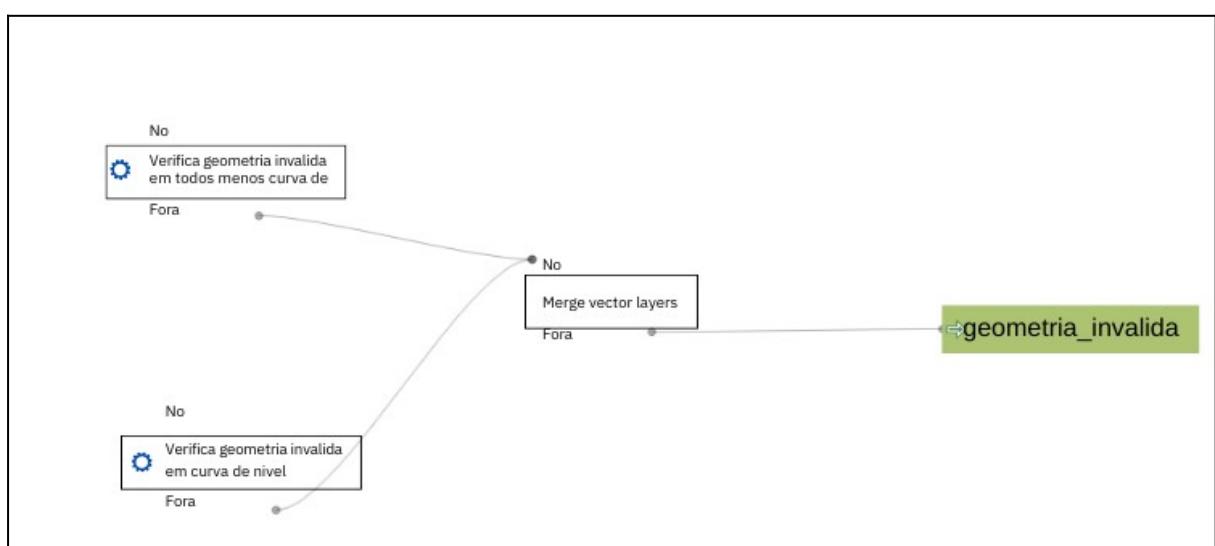


Figura 1: Modelo de identificação de geometrias inválidas.

**O modelo é composto por três etapas:**

Verifica geometria inválida: Identifica geometrias inválidas em todas as camadas de entrada, exceto a camada de curva de nível, utilizando o algoritmo "*identifyandfixinvalidgeometries*".

Verifica geometria inválida: Identifica geometrias inválidas somente na camada de curva de nível utilizando o algoritmo "*identifyandfixinvalidgeometries*".

*Merge Vector Layers:* Realiza uma operação de unir as camadas de saída utilizando o algoritmo *mergevectorlayers*.

### e) Parâmetros:

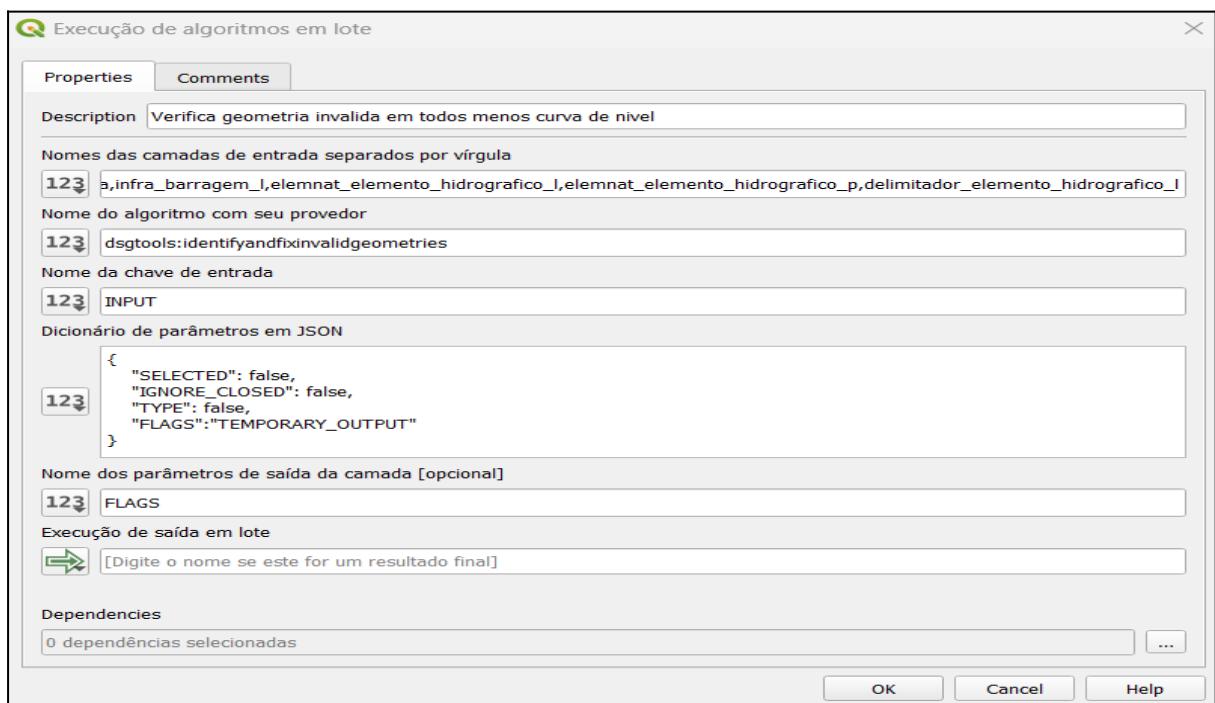


Figura 2: Extrato dos Parâmetros.

### Camadas de entrada:

Tipo de geometria	Camadas de entrada
Ponto	elemnat_ponto_cotado_p, centroide_massa_dagua_p, centroide_elemento_hidrografico_p, elemnat_elemento_fisiografico_p, elemnat_elemento_hidrografico_p, centroide_ilha_p.
Linha	delimitador_massa_dagua_l, elemnat_trecho_drenagem_l,

Linha	elemnat_elemento_fisiografico_1, infra_barragem_1, elemnat_elemento_hidrografico_1, elemnat_curva_nivel_1.
Polígono	cobter_massa_dagua_a, infra_barragem_a aux_moldura_a, moldura, aux_moldura_area_continua_a.

Para lidar com a peculiaridade da curva de nível, em que o início e o fim se encontram, foi criada uma verificação separada para o elemento elemnat\_curva\_nivel\_1 com o parâmetro "*IGNORE\_CLOSED*": *true*. Com isso, o algoritmo assegura que as curvas de nível fechadas não sejam erroneamente identificadas como falhas, evitando resultados falsos positivos no algoritmo.

**Parâmetros em JSON:** definem o comportamento do algoritmo provedor durante sua execução.

#### Verifica geometria inválida em todos, exceto curva de nível:

```
{
  "SELECTED": false,
  "IGNORE_CLOSED": false,
  "TYPE": false,
  "FLAGS": "TEMPORARY_OUTPUT"
}
```

### Verifica geometria inválida na curva de nível:

```
{  
    "SELECTED": false,  
    "IGNORE_CLOSED": true,  
    "TYPE": false,  
    "FLAGS": "TEMPORARY_OUTPUT"  
}
```

O parâmetro "**SELECTED**" determina se a validação deve ser realizada apenas nas geometrias selecionadas pelo usuário. Quando marcado como "*true*", a validação é realizada apenas nas geometrias selecionadas. Caso contrário, todas as geometrias são validadas.

O parâmetro "**IGNORE\_CLOSED**" determina se a rotina deve ignorar as geometrias fechadas, como os polígonos. Quando marcado como "*true*", as geometrias fechadas não são consideradas na validação. Caso contrário, todas as geometrias são avaliadas.

O parâmetro "**TYPE**" é usado para definir o tipo de geometria que será validada. Quando marcado como "*true*", a validação é realizada apenas nas geometrias do tipo especificado (linha, ponto ou polígono). Caso contrário, todas as geometrias são avaliadas.

O parâmetro "**FLAGS**" é utilizado para definir o nome da camada que será criada pela rotina para armazenar as informações de validação. Neste caso, o valor "**TEMPORARY\_OUTPUT**" especifica que uma camada temporária será gerada para armazenar informações sobre as geometrias inválidas encontradas durante a validação.

Para finalizar o processo, é realizada uma operação de "**MERGE VECTOR LAYERS**" para combinar todas as camadas temporárias de *FLAGS* em uma única camada.

f) **Nome da camada de *flags* gerada:** geometrias\_invalidas.

### g) Resultado do processo e exemplo de erros:

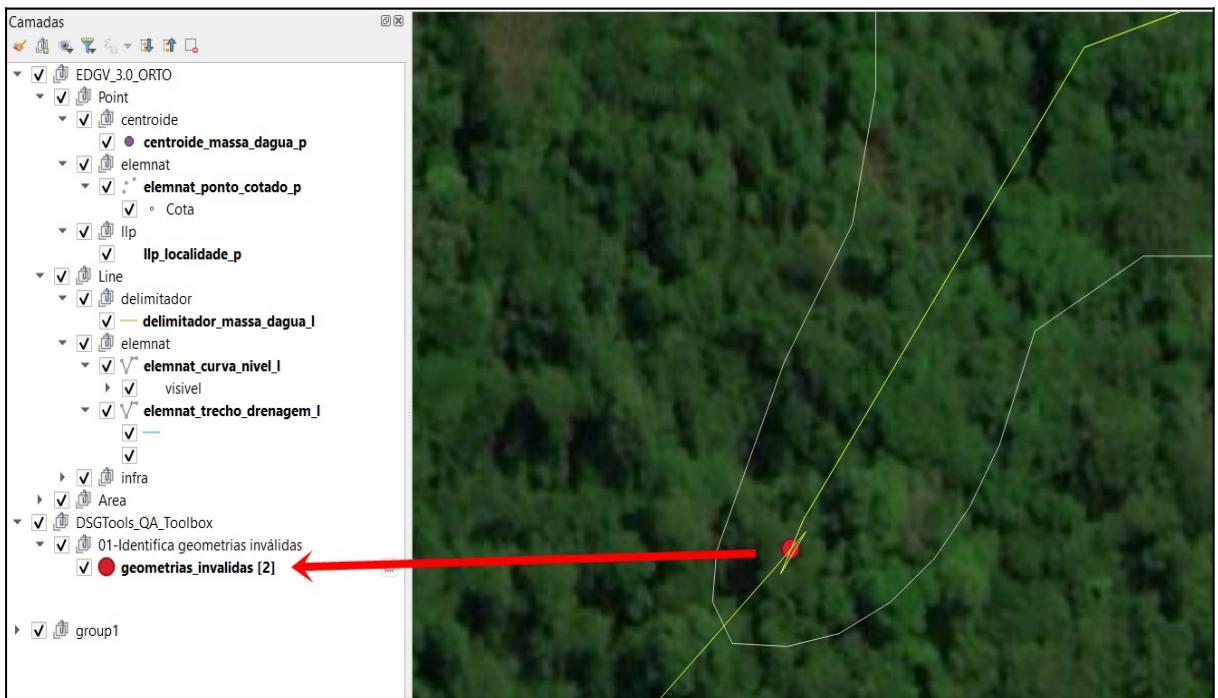


Figura 3: Exemplo de erro de geometria inválida.

## 2. IDENTIFICAR GEOMETRIAS COM MAIS DE UMA PARTE

**a) Descrição:** O algoritmo em questão identifica geometrias multipartidas (geométricas composta por múltiplas partes) em diferentes tipos de feições (linha, ponto e polígono) dentro de camadas específicas.

**b) Arquivo:** identifica.multipart.alt\_hid\_carta\_orto.model3.

**c) Algoritmos:**

```
dsgtools:batchrunalgorithm;  
dsgtools:identifymultigeometries;  
native:extractbylocation;  
dsgtools:stringcsvtofirstlayerwithelementsalgorithm.
```

#### d) Composição do Modelo:

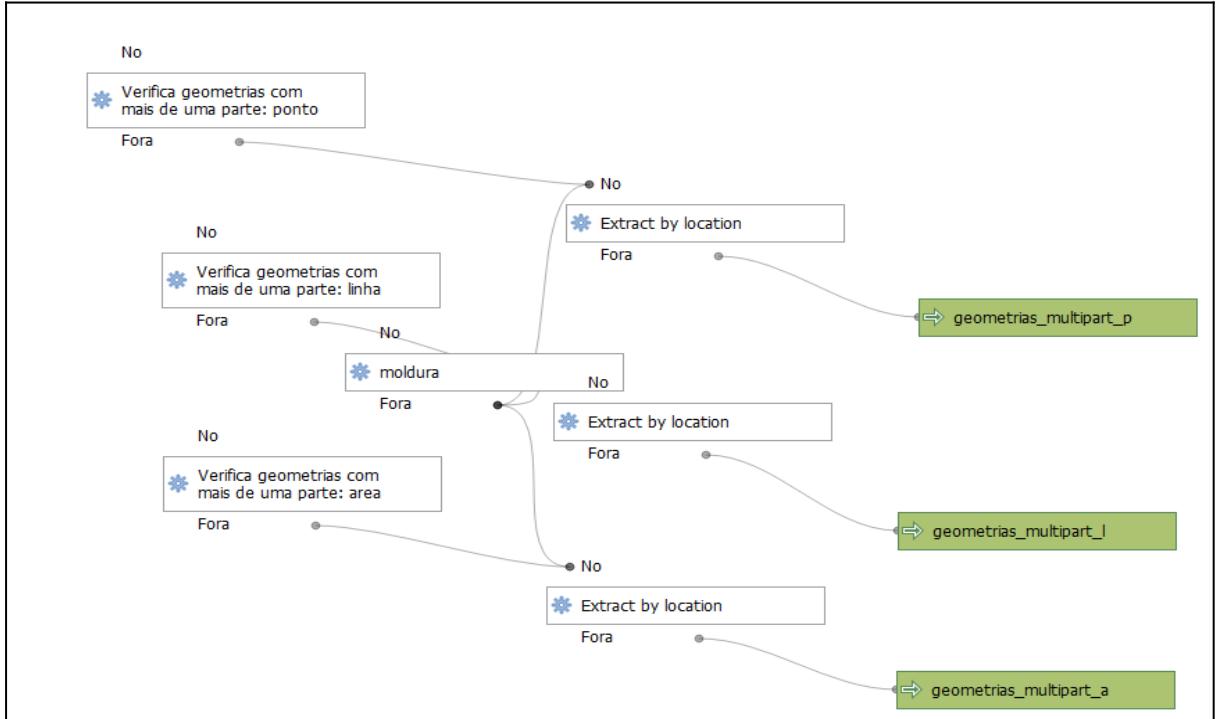


Figura 4: Modelo de identificação de geometria multiparte.

#### O modelo é composto por três etapas:

Verifica geometrias com mais de uma parte nas camadas de ponto, linha e polígono: Identifica geometrias com mais de uma parte nas camadas usando o algoritmo *identifymultigeometries*.

Moldura: Converte uma *string csv* em uma camada especificada usando o algoritmo *stringcsvtofirstlayerwithelementsalgorithm*.

*ExtractByLocation*: Extrai as geometrias com mais de uma parte que estão dentro da moldura usando o algoritmo *extractbylocation*.

### e) Parâmetros:

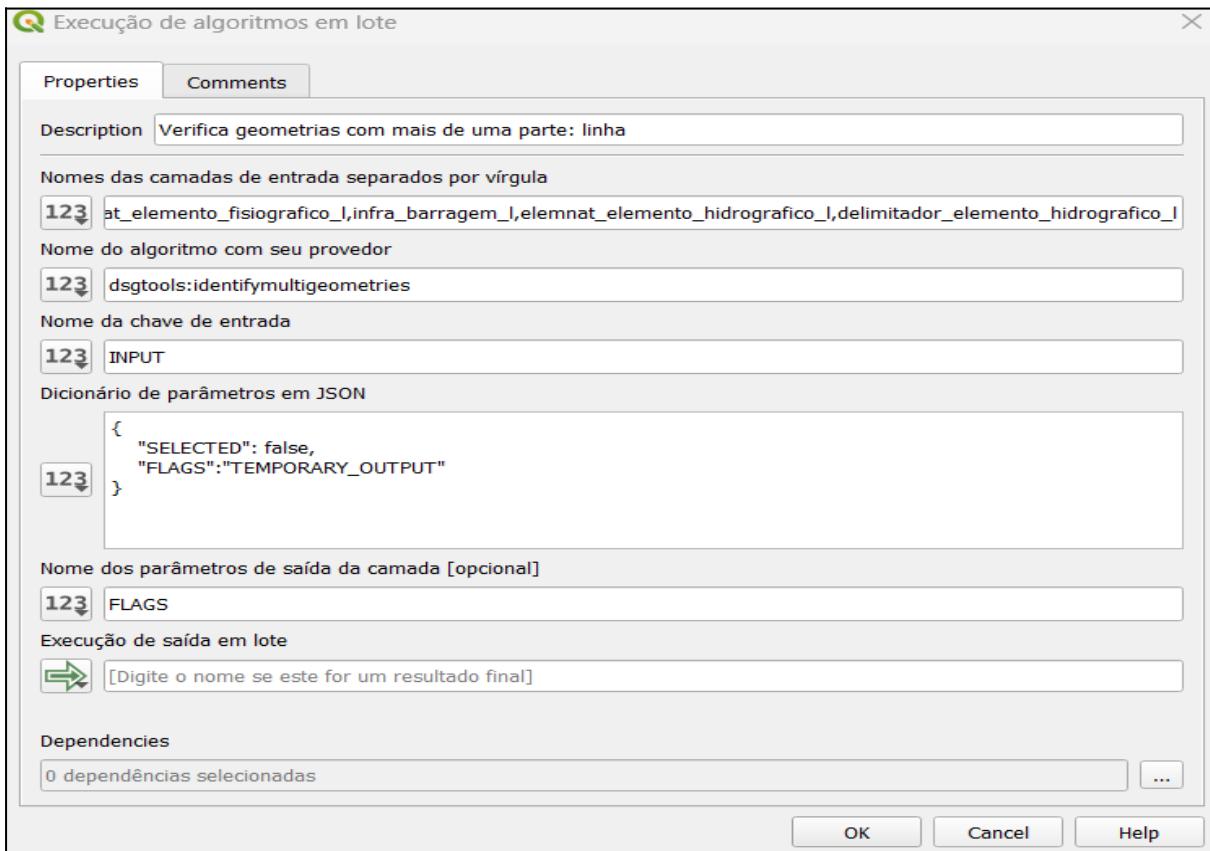


Figura 5: Extrato dos parâmetros.

### Camadas de entrada:

Tipo de geometria	Camadas de entrada
Ponto	elemnat_ponto_cotado_p, centroide_massa_dagua_p, centroide_elemento_hidrografico_p, elemnat_elemento_fisiografico_p, elemnat_elemento_hidrografico_p.
Linha	delimitador_massa_dagua_l, elemnat_trecho_drenagem_l, elemnat_curva_nivel_l,

Linha	elemnat_elemento_fisiografico_1, infra_barragem_1, elemnat_elemento_hidrografico_1, delimitador_elemento_hidrografico_1.
Polígono	cobter_massa_dagua_a, infra_barragem_a, aux_moldura_a, moldura, aux_moldura_area_continua_a.

Para cada tipo de geometria, o modelo utiliza o algoritmo "*dsgtools:identifymultigeometries*" para identificar as geometrias multipartidas presentes na camada.

#### Parâmetros em JSON:

```
{"SELECTED": false,  
"FLAGS": "TEMPORARY_OUTPUT"}
```

Na sequência, o modelo usa o algoritmo do QGIS "*native:extractbylocation*" para extrair apenas as geometrias que intersectam com a camada "moldura" no modo interseção (parâmetro "*PREDICATE*" = [0]). Esse parâmetro significa que somente as geometrias que têm intersecção com a moldura serão extraídas.

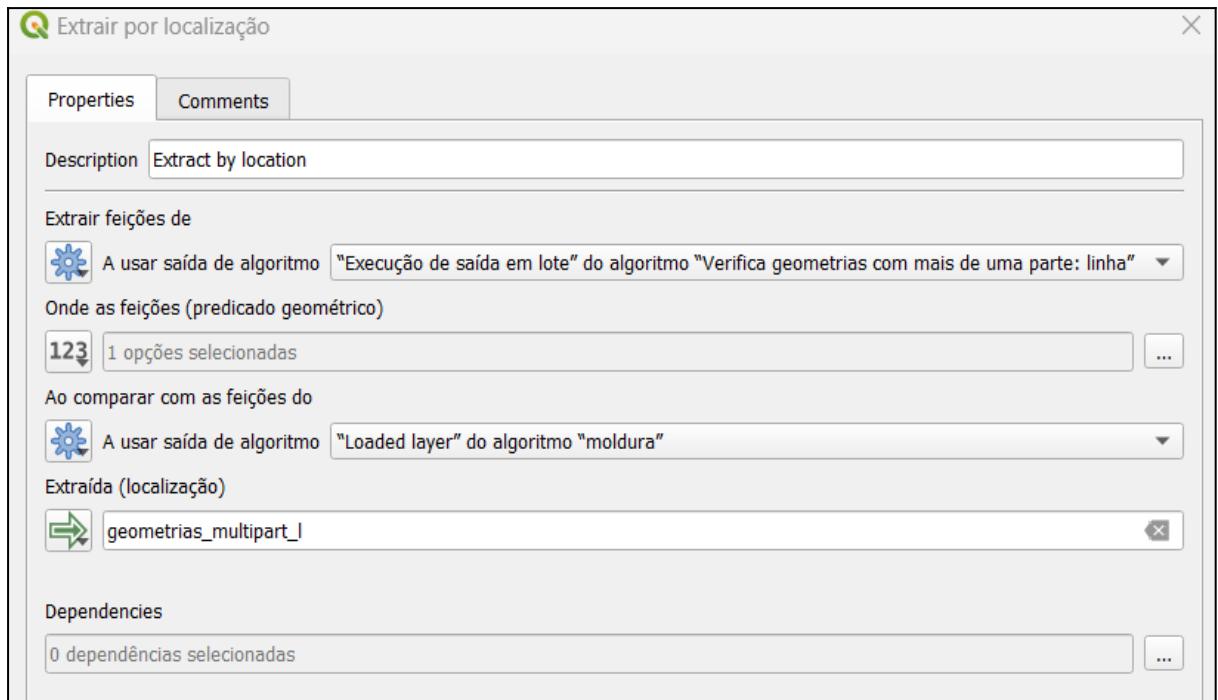


Figura 6: Extrair por localização.

O objetivo é restringir as análises apenas às geometrias que estão dentro de uma área de interesse, no caso a moldura.

**f) Nome da camada de flags:**

geometrias\_multipart\_p;  
geometrias\_multipart\_l;  
geometrias\_multipart\_a.

### g) Resultado do processo e exemplo de erros:

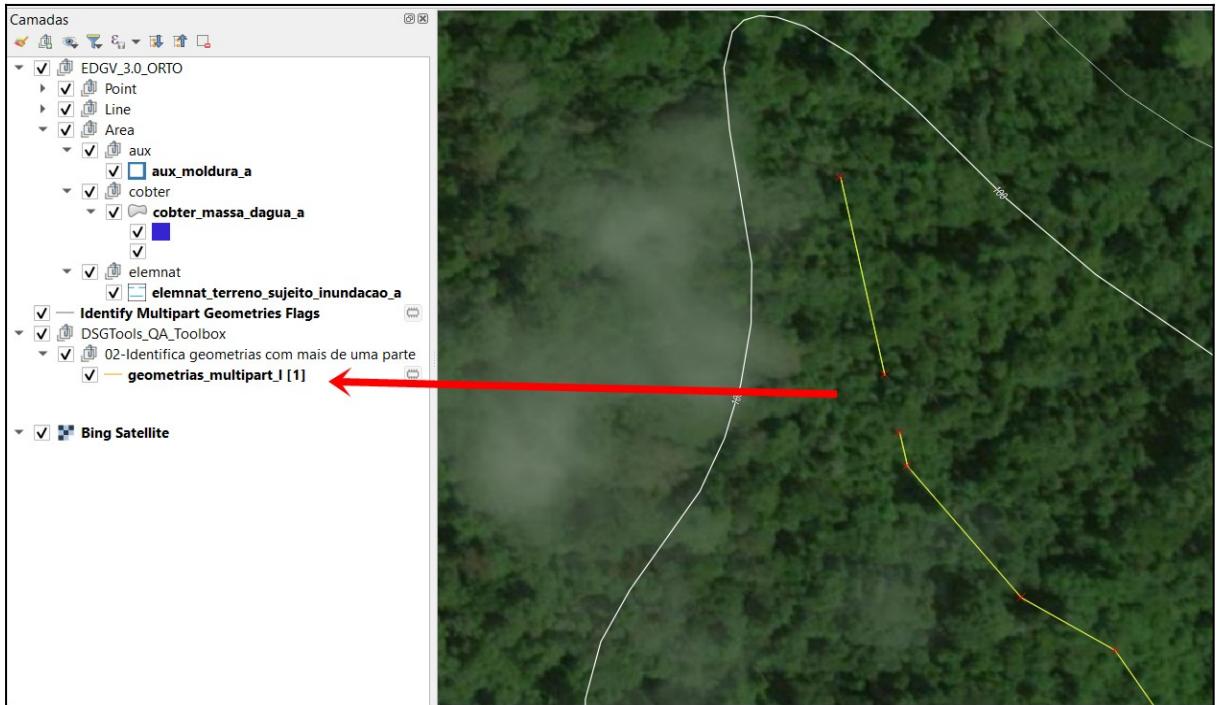


Figura 7: Exemplo de geometrias multiparte.

## 3. IDENTIFICAR FEIÇÕES DUPLICADAS

**a) Descrição:** Essa etapa tem como objetivo identificar feições duplicadas em diferentes camadas, que podem ter sido criadas acidentalmente ou durante o processo de validação. Para isso, o modelo utiliza uma lista de campos a ignorar de atributos (*ATTRIBUTE\_BLACKLIST*) que não serão considerados na comparação das feições, a fim de evitar falsos positivos. As camadas de ponto, linha e polígono são verificadas separadamente e as feições duplicadas encontradas são sinalizadas com flags nas camadas correspondentes.

**b) Arquivo:** `identifica_feicoes_duplicadas_alt_hid_carta_orto.model3`.

**c) Algoritmos:**

```
dsgtools:identifyduplicatedfeatures;  
dsgtools:batchrunalgorithm;  
dsgtools:identifyunsharedvertexonintersectionsalgorithm;
```

*dsgtools:stringcsvtofirstlayerwithelementsalgorithm.*

#### d) Composição do Modelo:

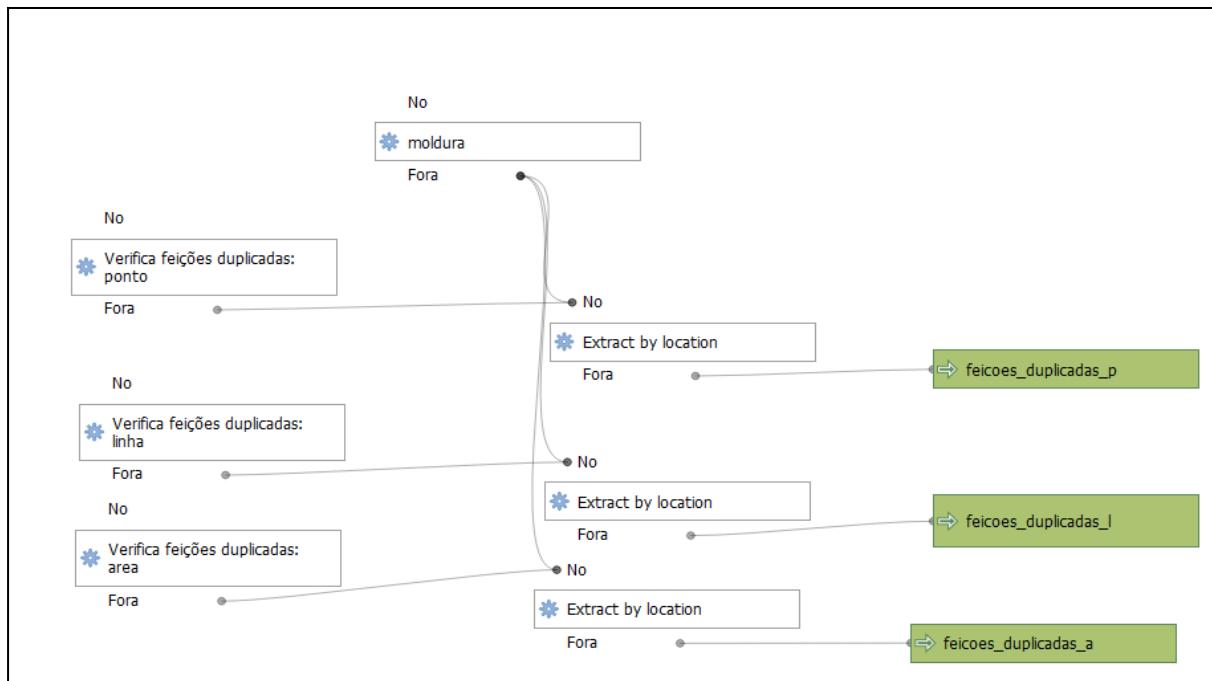


Figura 8: Modelo de identificação de geometrias duplicadas.

#### O modelo é composto por três etapas:

Moldura: Converte uma *string* csv em uma camada especificada usando o algoritmo *stringcsvtofirstlayerwithelementsalgorithm*.

Verifica feições duplicadas: Identifica feições duplicadas nas camadas de ponto, linha e polígonos usando o algoritmo *identifyduplicatedfeatures*.

ExtractByLocation: Extrai as feições duplicadas que estão dentro da moldura usando o algoritmo *extractbylocation*.

### e) Parâmetros:

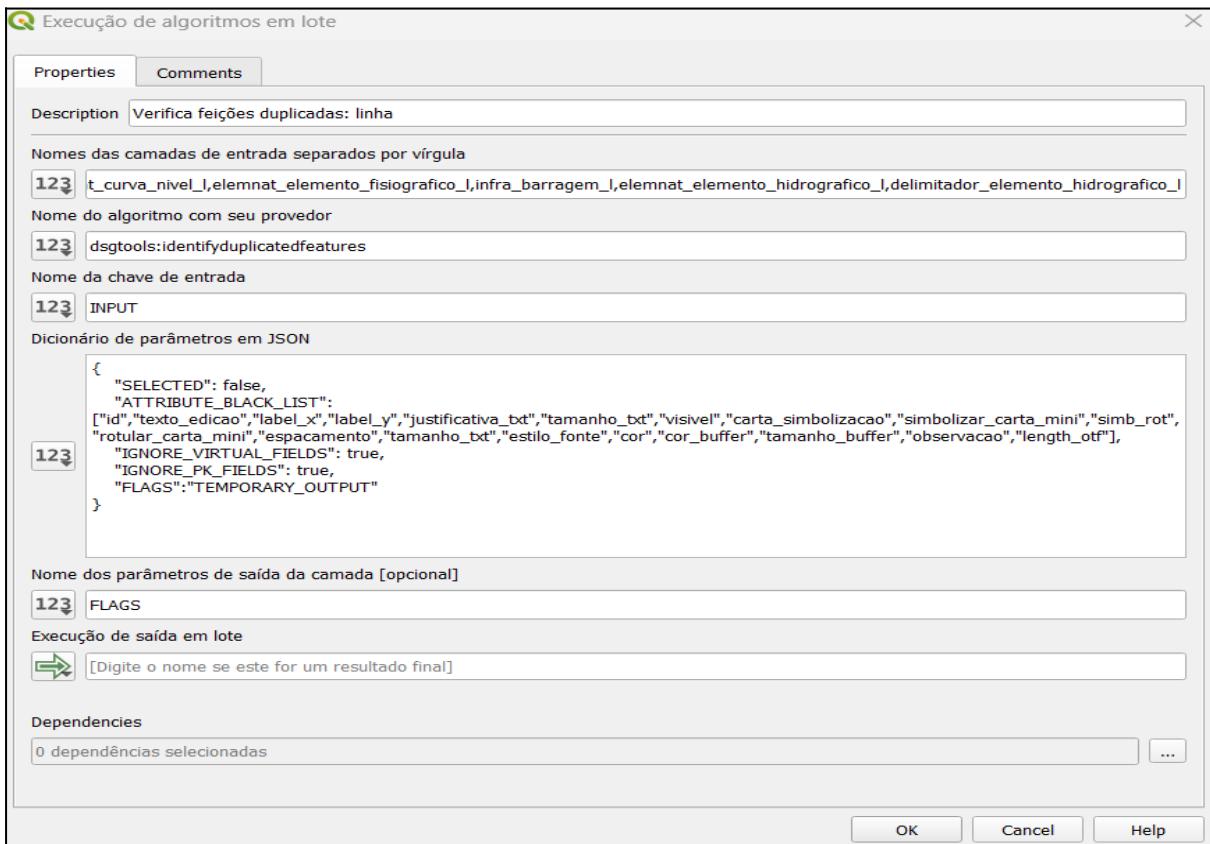


Figura 9: Extrato dos parâmetros.

### Camadas de entrada:

Tipo de geometria	Camadas de entrada
Ponto	elemnat_ponto_cotado_p, centroide_massa_dagua_p, centroide_elemento_hidrografico_p, elemnat_elemento_fisiografico_p, elemnat_elemento_hidrografico_p.
Linha	delimitador_massa_dagua_l, elemnat_trecho_drenagem_l, elemnat_curva_nivel_l,

Linha	elemnat_elemento_fisiografico_1, infra_barragem_l, elemnat_elemento_hidrografico_1, delimitador_elemento_hidrografico_1.
Polígono	cobter_massa_dagua_a, infra_barragem_a, aux_moldura_a, moldura, aux_moldura_area_continua_a.

**Parâmetros em JSON:**

```
{"SELECTED": false,
"ATTRIBUTE_BLACK_LIST":
["id", "texto_edicao", "label_x", "label_y", "justificativa_txt", "tamanho_txt", "visivel", "operador_criacao", "data_criacao", "operador_atualizacao", "data_atualizacao", "carta_simbolizacao", "simbolizar_carta_mini", "simb_rot", "rotular_carta_mini", "espacamento", "tamanho_txt", "estilo_fonte", "cor", "cor_buffer", "tamanho_buffer", "observacao", "length_otf"],
"IGNORE_VIRTUAL_FIELDS": true,
"IGNORE_PK_FIELDS": true,
"FLAGS": "TEMPORARY_OUTPUT"}
```

**f) Nome da camada de *flags*:**

feicoes\_duplicadas\_p;  
feicoes\_duplicadas\_l;  
feicoes\_duplicadas\_a.

### g) Resultado do processo e exemplo de erros:

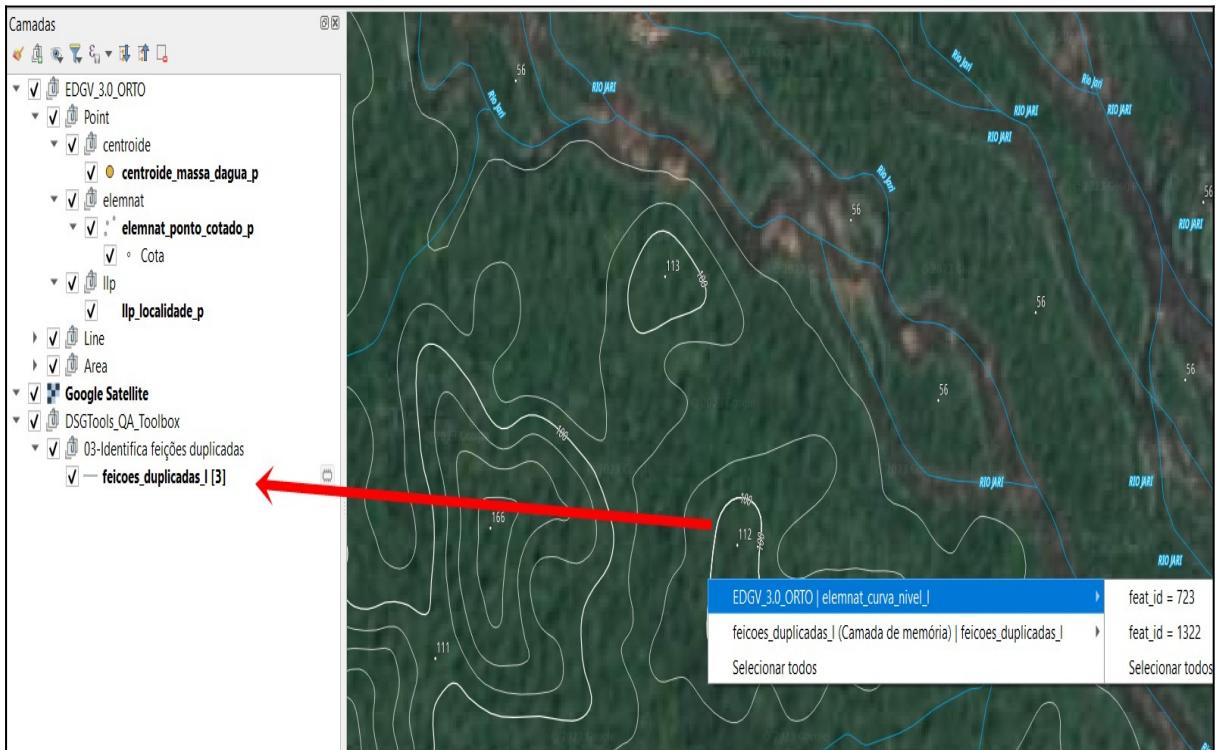


Figura 10: Exemplo de feições duplicadas.

## 4. IDENTIFICAR VÉRTICE NÃO COMPARTILHADO NAS INTERSECÇÕES

**a) Descrição:** Este algoritmo identifica vértices não compartilhados nas intersecções entre camadas de linha e polígono.

### b) Arquivo:

identifica\_vertice\_nao\_compartilhado\_nas\_interseccoes\_alt\_hid\_carta\_orto.model3.

### c) Algoritmos:

```
dsgtools:stringcsvtofirstlayerwithelementsalgorithm;
dsgtools:stringcsvtolayerlistalgorithm;
dsgtools:identifyunsharedvertexonintersectionsalgorithm;
native:extractbylocation.
```

#### d) Composição do Modelo:

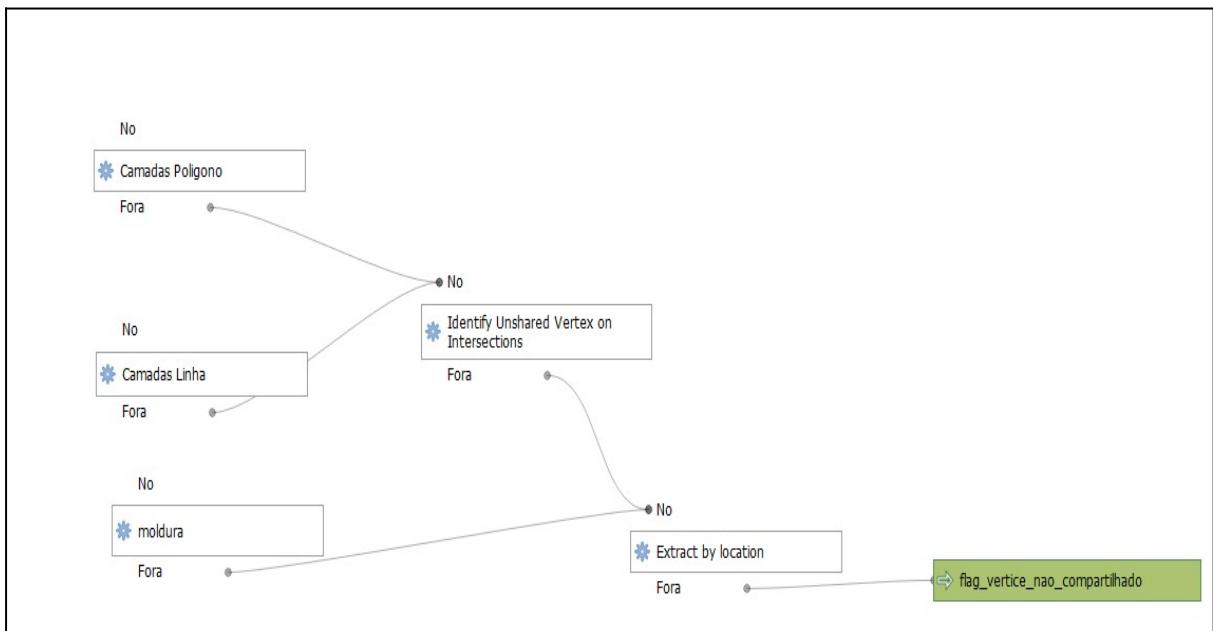


Figura 11: Modelo de identificação de vértices não compartilhados nas intersecções.

#### O modelo é composto por cinco etapas:

Camadas Polígono: Converte uma string csv em uma lista de camadas de polígono usando o algoritmo *stringcsvtolayerlistalgorithm*.

Camadas Linha: Converte uma string csv em uma lista de camadas de linha usando o algoritmo *stringcsvtolayerlistalgorithm*.

Moldura: Converte uma string csv em uma camada especificada usando o algoritmo *stringcsvtofirstlayerwithelementsalgorithm*.

*IdentifyUnsharedVertexOnIntersections*: Identifica vértices não compartilhados nas intersecções entre as camadas de linha e polígono usando o algoritmo *identifyunsharedvertexonintersectionsalgorithm*.

*ExtractByLocation*: Extrai os vértices não compartilhados que estão dentro da moldura usando o algoritmo *extractbylocation*.

### e) Parâmetros:



Figura 12: Extrato dos parâmetros.

**Camadas de entrada:** Uma lista de camadas de linha e polígono.

**Processar apenas feições selecionadas:** Um booleano que indica se apenas as entidades selecionadas na camada de entrada serão processadas.

**Flags:** Uma camada de saída temporária que armazena as intersecções identificadas com vértices não compartilhados.

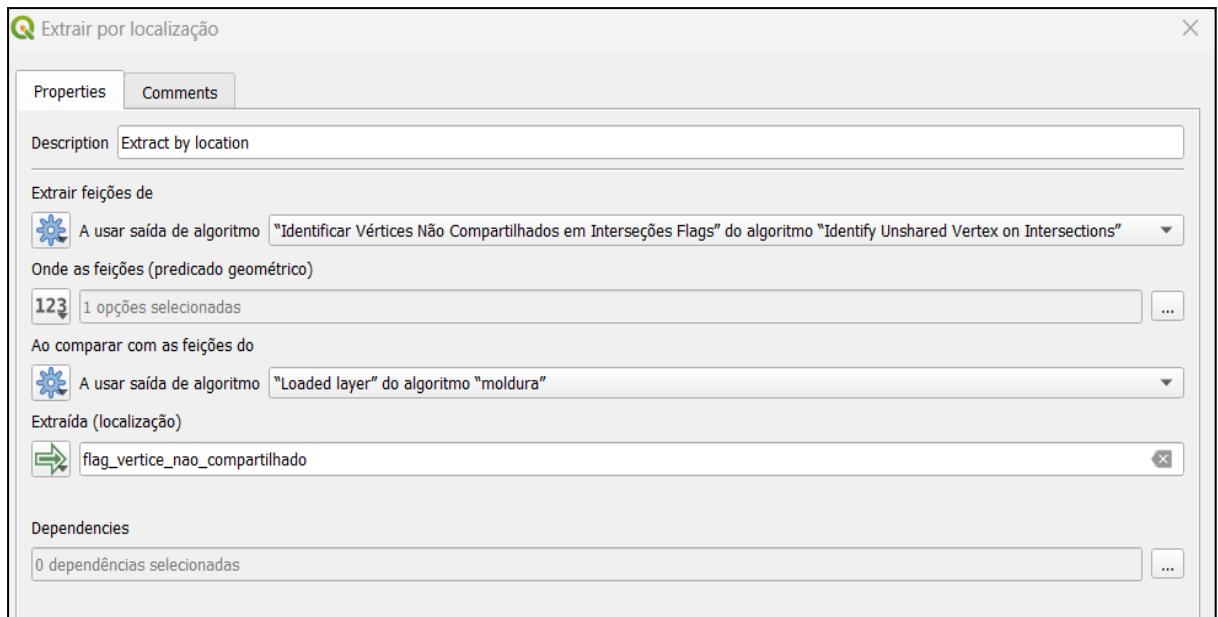


Figura 13: Extrato dos parâmetros.

**Extrair feições de:** Utiliza os dados de saída do procedimento anterior.

**Predicado geométrico:** Uma lista que especifica o tipo de relação espacial entre as geometrias que devem ser consideradas na extração. Neste caso, o valor [0] significa que apenas as geometrias que se intersectam serão extraídas.

**Ao comparar com as feições do:** Camada de polígono que será usada para extrair as intersecções.

**Flags:** Camada de saída final que armazenará os vértices não compartilhados identificados.

### Camadas de entrada:

Tipo de geometria	Camadas de entrada
Linha	delimitador_massa_dagua_1, elemnat_trecho_drenagem_1, infra_barragem_1, elemnat_elemento_hidrografico_1.
Polígono	cobter_massa_dagua_a, infra_barragem_a aux_moldura_a, moldura, aux_moldura_area_continua_a.

f) Nome da camada de *flags*: *Flag\_vertice\_nao\_compartilhado*.

g) Resultado do processo e exemplo de erros:

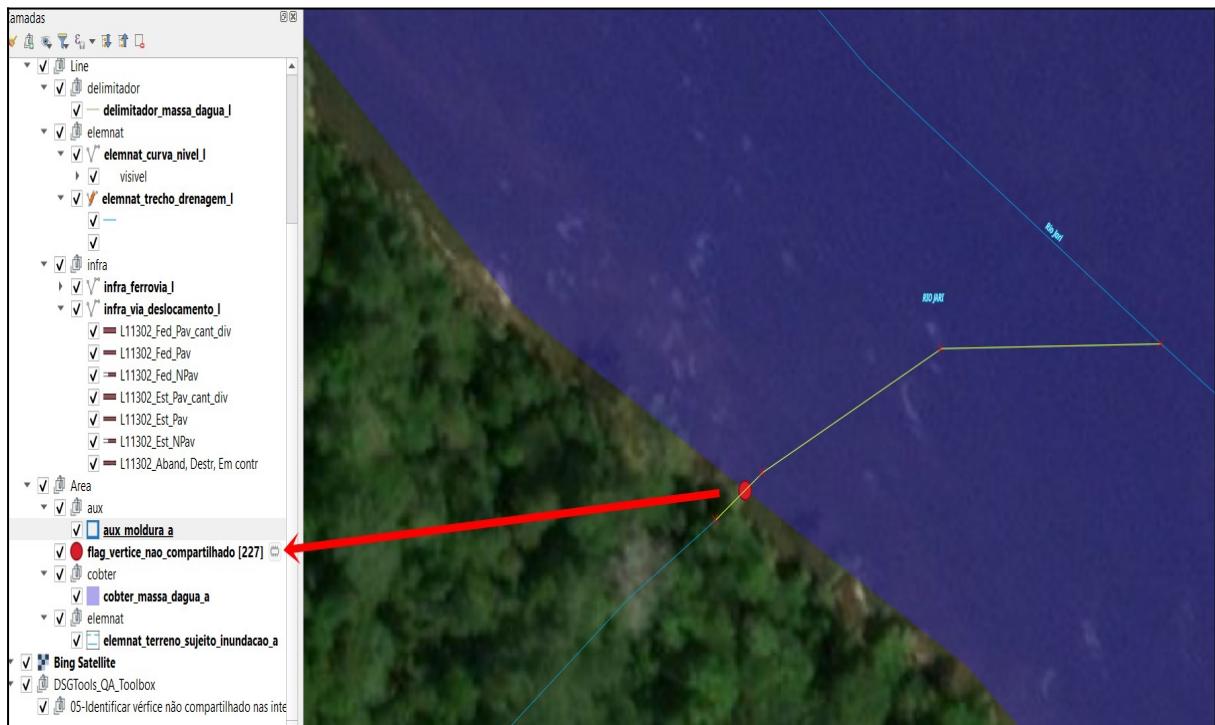


Figura 14: Exemplo de vértice não compartilhado.

## 5. IDENTIFICAR VÉRTICE NÃO COMPARTILHADO NOS SEGMENTOS COMPARTILHADOS

**a) Descrição:** Este algoritmo identifica vértices não compartilhados em segmentos compartilhados entre camadas de linha e polígono.

### b) Arquivo:

identificar\_vertice\_nao\_compartilhado\_nos\_segmentos\_compartilhados.model3.

### c) Algoritmos:

```
dsgtools:stringcsvtolayerlistalgorithm;  
dsgtools:stringcsvtofirstlayerwithelementsalgorithm;  
dsgtools:identifyunsharedvertexonsharededgesalgorithm;  
native:extractbylocation.
```

### Composição do Modelo:

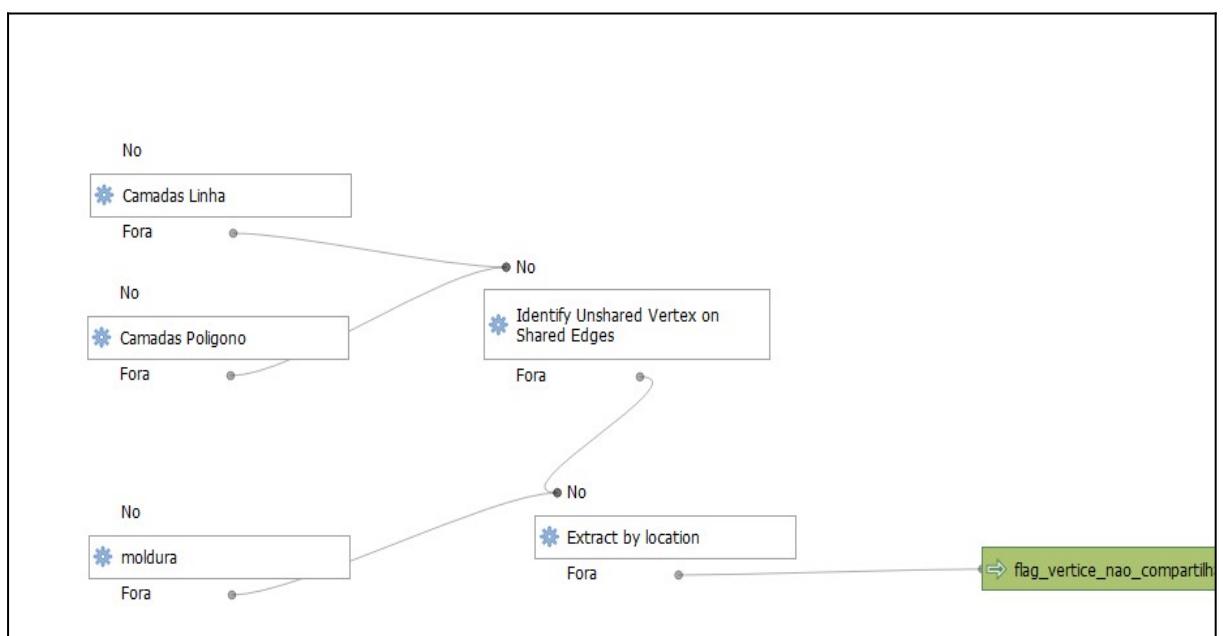


Figura 15: Modelo de identificação de vértices não compartilhados nos segmentos compartilhados.

## O modelo é composto por cinco etapas:

Camadas Linha: Converte uma *string* csv em uma lista de camadas de linha usando o algoritmo *stringcsvtolayerlistalgorithm*.

Camadas Polígono: Converte uma *string* csv em uma lista de camadas de polígono usando o algoritmo *stringcsvtolayerlistalgorithm*.

Moldura: Converte uma *string* csv em uma camada especificada usando o algoritmo *stringcsvtofirstlayerwithelementsalgorithm*.

*IdentifyUnsharedVertexOnSharedEdges*: Identifica vértices não compartilhados nos segmentos compartilhados entre as camadas de linha e polígono usando o algoritmo *identifyunsharedvertexonsharededgesalgorithm*.

*ExtractByLocation*: Extrai os vértices não compartilhados que estão dentro da moldura usando o algoritmo *extractbylocation*.

### d) Parâmetros:

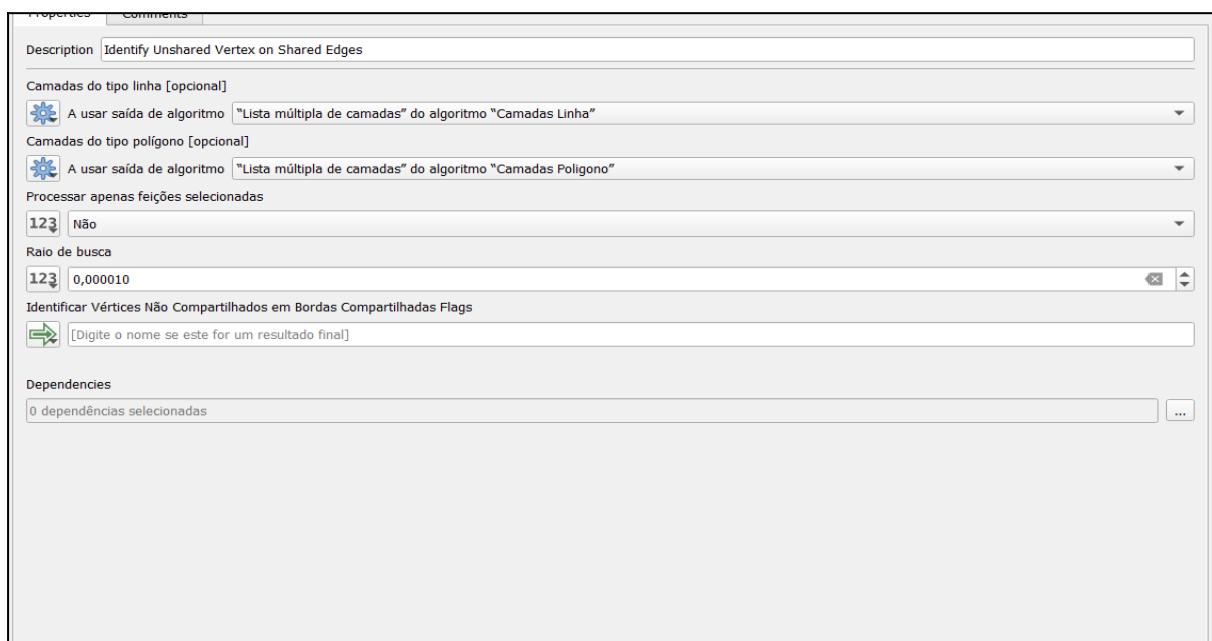


Figura 16: Extrato dos parâmetros.

**Processar apenas feições selecionadas:** Um booleano que indica se apenas as entidades selecionadas na camada de entrada serão processadas.

**O Parâmetro Raio de busca:** O raio de busca do algoritmo determina a distância máxima que será percorrida para encontrar vértices não compartilhados em relação aos segmentos de linha e polígonos que se cruzam nas intersecções. No modelo, o valor padrão do raio de busca é definido como 0,00001 graus no sistema de coordenadas geográfico que corresponde a um valor aproximado de 1 metro. No entanto, é importante lembrar que esse valor pode ser ajustado de acordo com as necessidades específicas, considerando a escala e precisão dos dados utilizados no processamento. Outro ponto importante que precisa ser ressaltado é que deve ser levado em consideração que se houver vértices muito próximos em uma linha, dentro do raio de busca, será gerado uma *flag* apontando o erro como vértice não compartilhado, nesse caso a geometria precisa ser simplificada, reduzindo os vértices muito próximos.

**Flags:** Uma camada de saída temporária que armazena as intersecções identificadas com vértices não compartilhados.

#### **Camadas de entrada:**

Tipo de geometria	Camadas de entrada
Linha	delimitador_massa_dagua_1, elemnat_trecho_drenagem_1, infra_barragem_1, elemnat_elemento_hidrografico_1.
Polígono	cobter_massa_dagua_a, infra_barragem_a, aux_moldura_a, moldura, aux_moldura_area_continua_a.

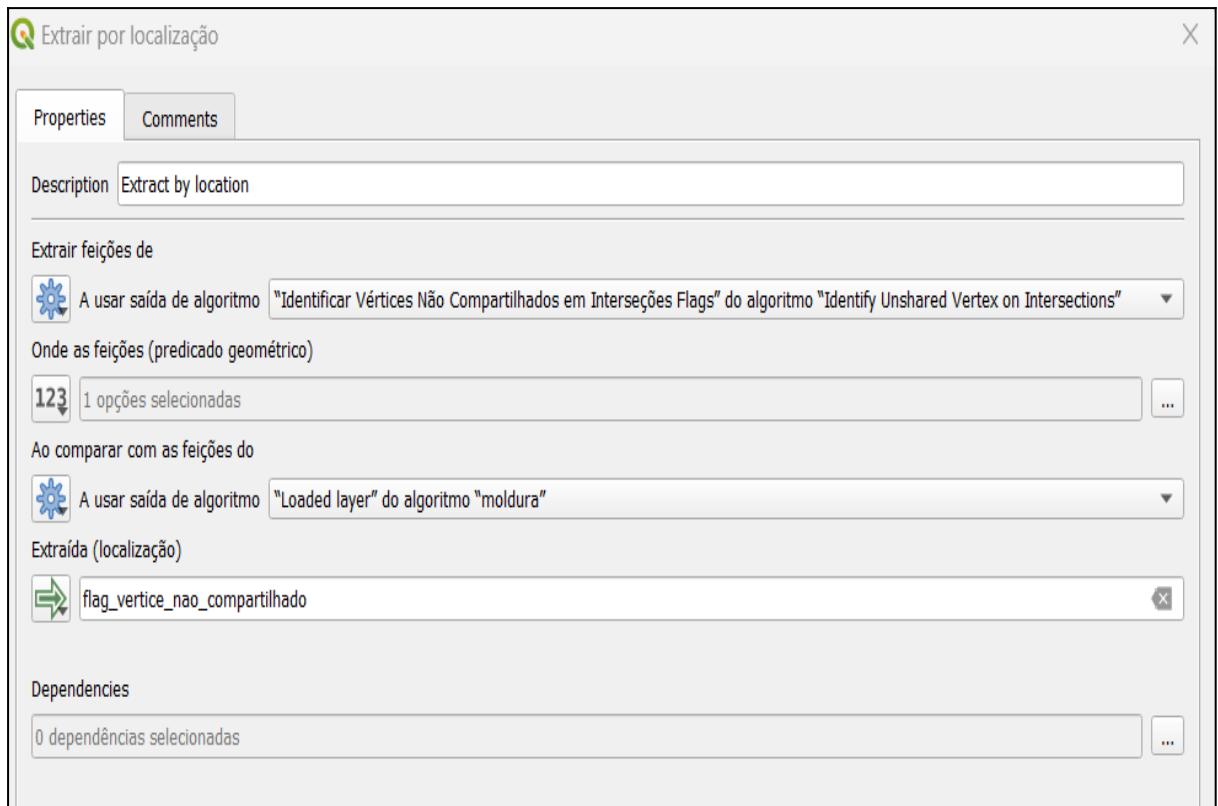


Figura 17: Extrair por localização.

**Extrair feições de:** Utiliza os dados de saída do procedimento anterior.

**Predicado geométrico:** Uma lista que especifica o tipo de relação espacial entre as geometrias que devem ser consideradas na extração. Neste caso, o valor [0] significa que apenas as geometrias que se intersectam serão extraídas.

**Ao comparar com as feições do:** Camada de polígono que será usada para extrair as intersecções.

**Flags:** Camada de saída final que armazenará os vértices não compartilhados identificados.

#### f) Nome da camada de flags:

*Flag\_vertice\_nao\_compartilhado\_em\_seg\_compartilhado.*

### g) Resultado do processo e exemplo de erros:

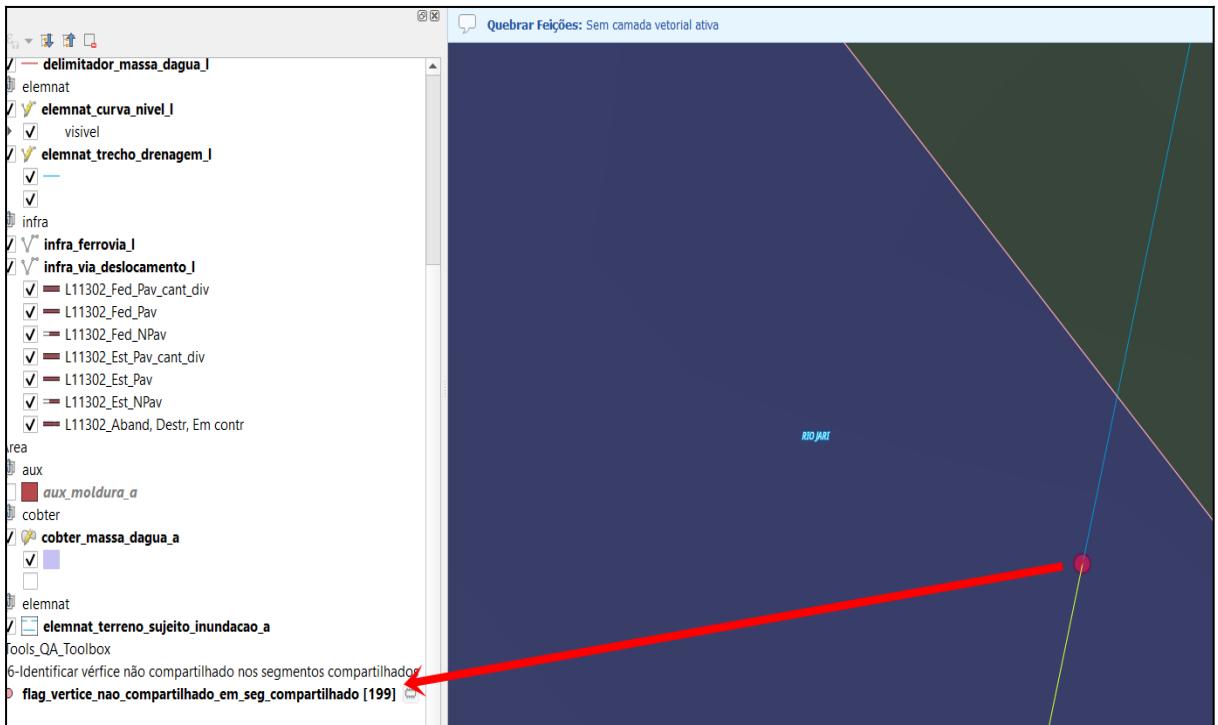


Figura 18: Exemplo de vértice não compartilhado nos segmentos compartilhados.

## 6. IDENTIFICAR VÉRTICE PRÓXIMO DE ARESTA

**a) Descrição:** Este algoritmo identifica vértices muito próximos de arestas em uma lista de camadas.

**b) Arquivo:** `identificar_vertice_proximo_de_aresta.model3`.

**c) Algoritmo:**

```
dsgtools:batchrunalgorithm;  
dsgtools:stringcsvtofirstlayerwithelementsalgorithm;  
native:extractbylocation;  
dsgtools:identifyvertexnearedges.
```

#### d) Composição do Modelo:

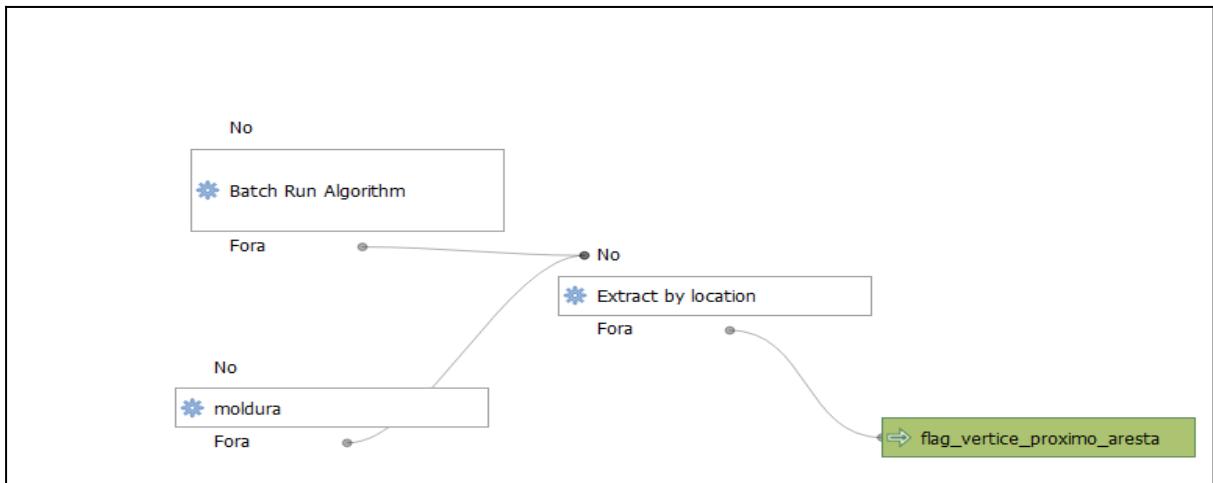


Figura 19: Modelo de identificação de vértices próximo a aresta.

#### O modelo é composto por três etapas:

*BatchRunAlgorithm*: Executa o algoritmo *identifyvertexneareddges* em uma lista de camadas de linha.

Moldura: Converte uma *string csv* em uma camada especificada usando o algoritmo *stringcsvtofirstlayerwithelementsalgorithm*.

*ExtractByLocation*: Extrai os vértices que estão muito próximos de arestas, utilizando um raio pequeno de busca, e que estão dentro da moldura usando o algoritmo *extractbylocation*.

### e) Parâmetros:

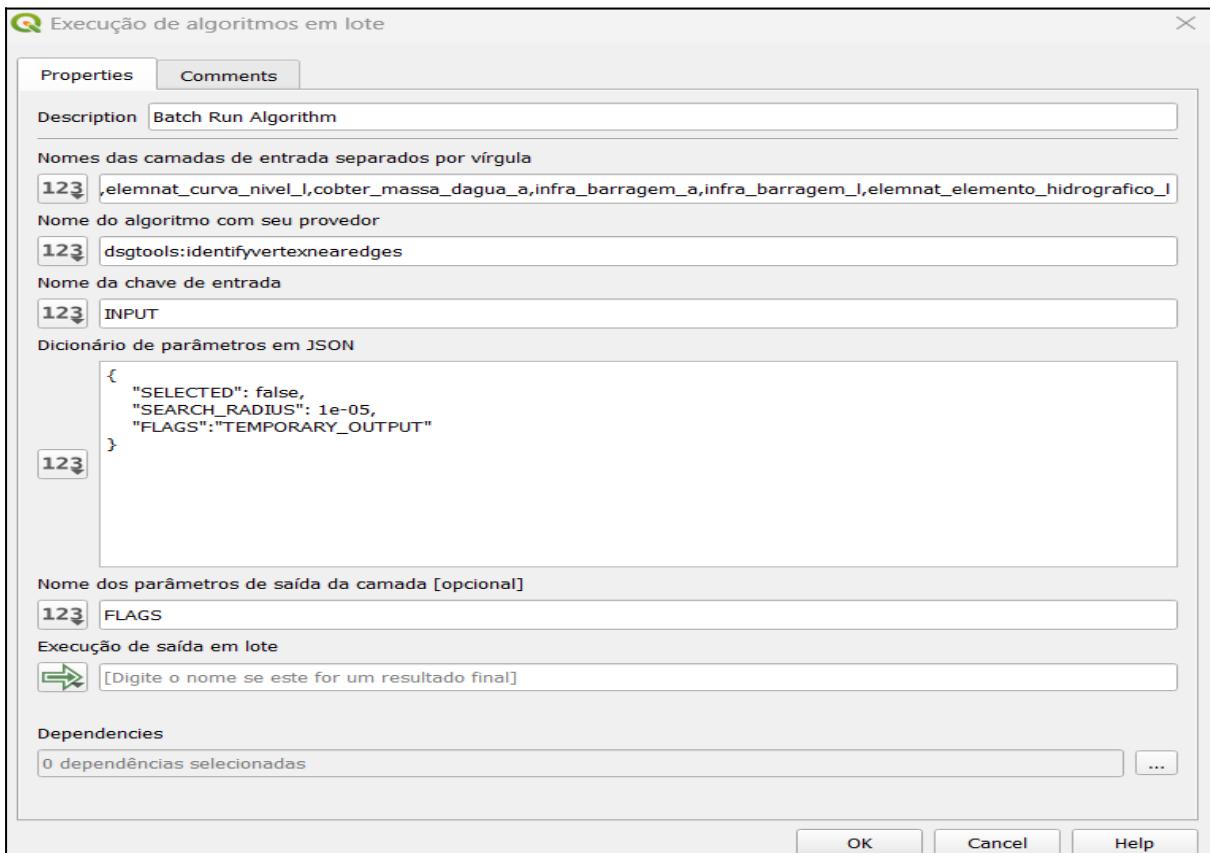


Figura 20: Extrato dos parâmetros.

### Camadas de entrada:

Tipo de geometria	Camadas de entrada
Linha	delimitador_massa_dagua_l, elemnat_trecho_drenagem_l, elemnat_curva_nivel_l, infra_barragem_l, elemnat_elemento_hidrografico_l.
Polígono	cobter_massa_dagua_a, infra_barragem_a, aux_moldura_a, moldura, aux_moldura_area_continua_a.

### Parâmetros em JSON:

```
{ "SELECTED": false,  
  "SEARCH_RADIUS": 1e-05,  
  "FLAGS": "TEMPORARY_OUTPUT" }
```

O parâmetro "**SELECTED**" determina se a validação deve ser realizada apenas nas geometrias selecionadas pelo usuário. Quando marcado como "true", a validação é realizada apenas nas geometrias selecionadas. Caso contrário, todas as geometrias são validadas.

No parâmetro "**SEARCH\_RADIUS**" o raio de busca de 0,00001 graus no sistema de coordenadas geográfico, que corresponde a aproximadamente 1 metro, determina a distância máxima que será percorrida, nesse caso com o objetivo de buscar distâncias muito pequenas entre os vértices que poderiam vir a colapsar a geometria.

O parâmetro "**FLAGS**" é utilizado para definir o nome da camada que será criada pela rotina para armazenar as informações de validação. Neste caso, o valor "**TEMPORARY\_OUTPUT**" especifica que uma camada temporária será gerada para armazenar informações sobre os vértices próximo a aresta encontradas durante a validação.

f) Nome da camada de *flags*: *Flag\_vertice\_proximo\_aresta*.

g) Resultado do processo e exemplo de erros:

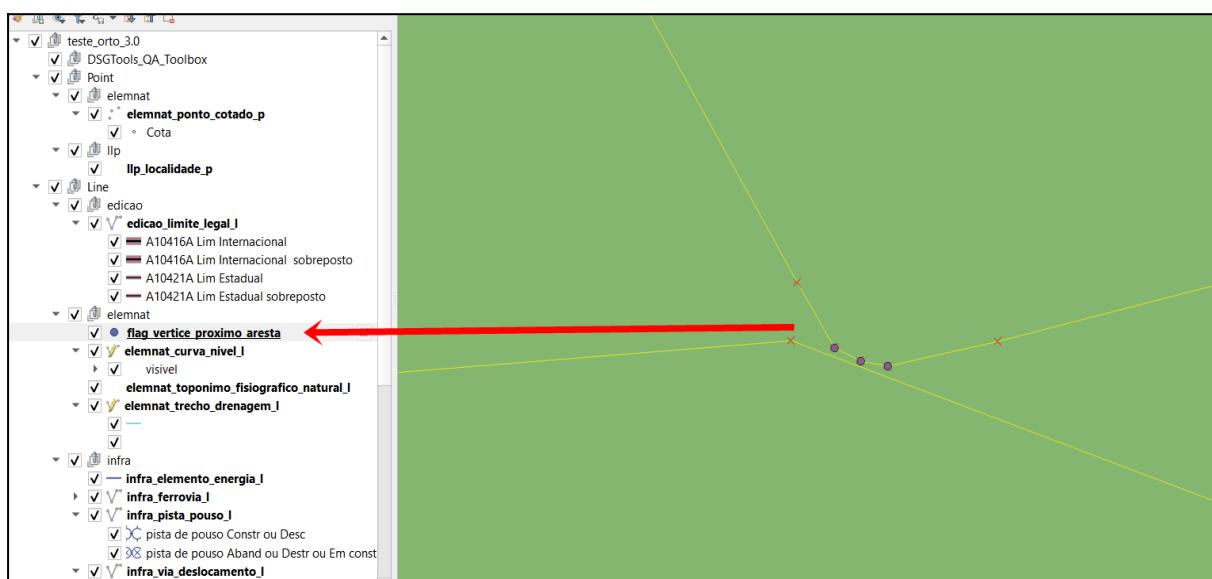


Figura 21: Exemplo de vértice próximo a aresta.

## 7. IDENTIFICAR GEOMETRIAS COM DENSIDADE INCORRETA DE VÉRTICES

**a) Descrição:** O objetivo deste algoritmo é realizar a identificação de geometrias que apresentam uma densidade de vértices considerada inadequada. Essa identificação é baseada em uma distância específica dentro de uma tolerância preestabelecida. O propósito principal é reduzir a quantidade de vértices presentes nas geometrias, buscando simplificar e otimizar sua representação.

**b) Arquivo:**

identifica\_geometrias\_com\_densidade\_incorreta\_de\_vertices\_alt\_hid\_carta\_orto.model3.

**c) Algoritmo:**

```
dsgtools:batchrunalgorithm;  
dsgtools:stringcsvtofirstlayerwithelementsalgorithm;  
native:extractbylocation;  
dsgtools:identifygeometrieswithlargevertexdensityalgorithm.
```

**d) Composição do Modelo:**

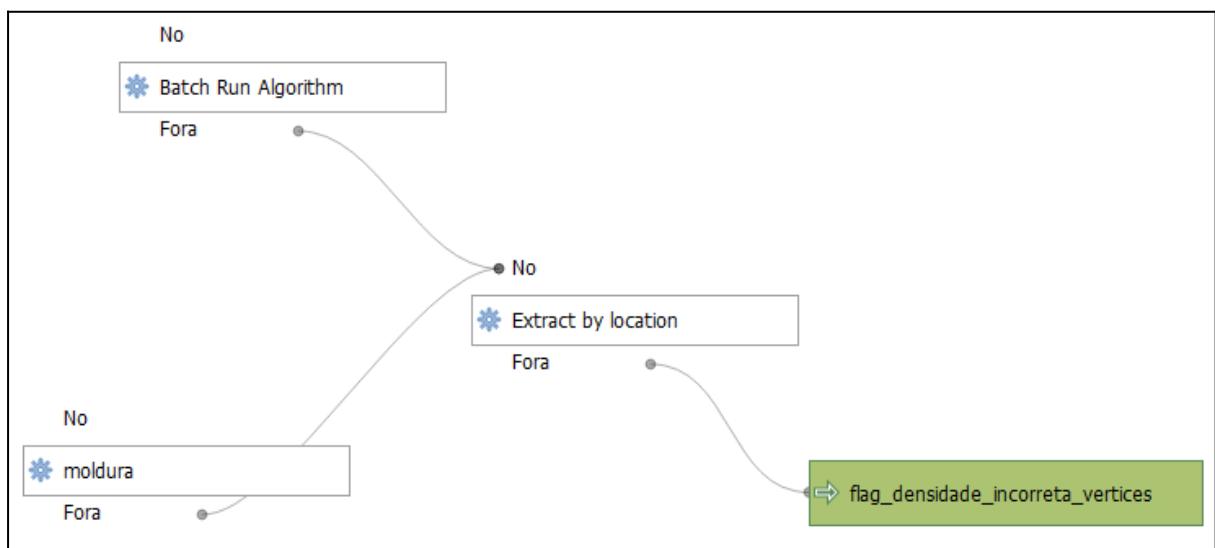


Figura 22: Modelo de identificação de geometrias com densidade incorreta de vértices.

## O modelo é composto por três etapas:

*BatchRunAlgorithm*: Executa o algoritmo *identifygeometrieswithlargevertexdensityalgorithm* com os parâmetros descritos conforme a seção e.

Moldura: Converte uma *string csv* em uma camada especificada usando o algoritmo *stringcsvtofirstlayerwithelementsalgorithm*.

*ExtractByLocation*: Extrai os pontos com densidade incorreta que estão dentro da moldura usando o algoritmo *extractbylocation*.

### e) Parâmetros:

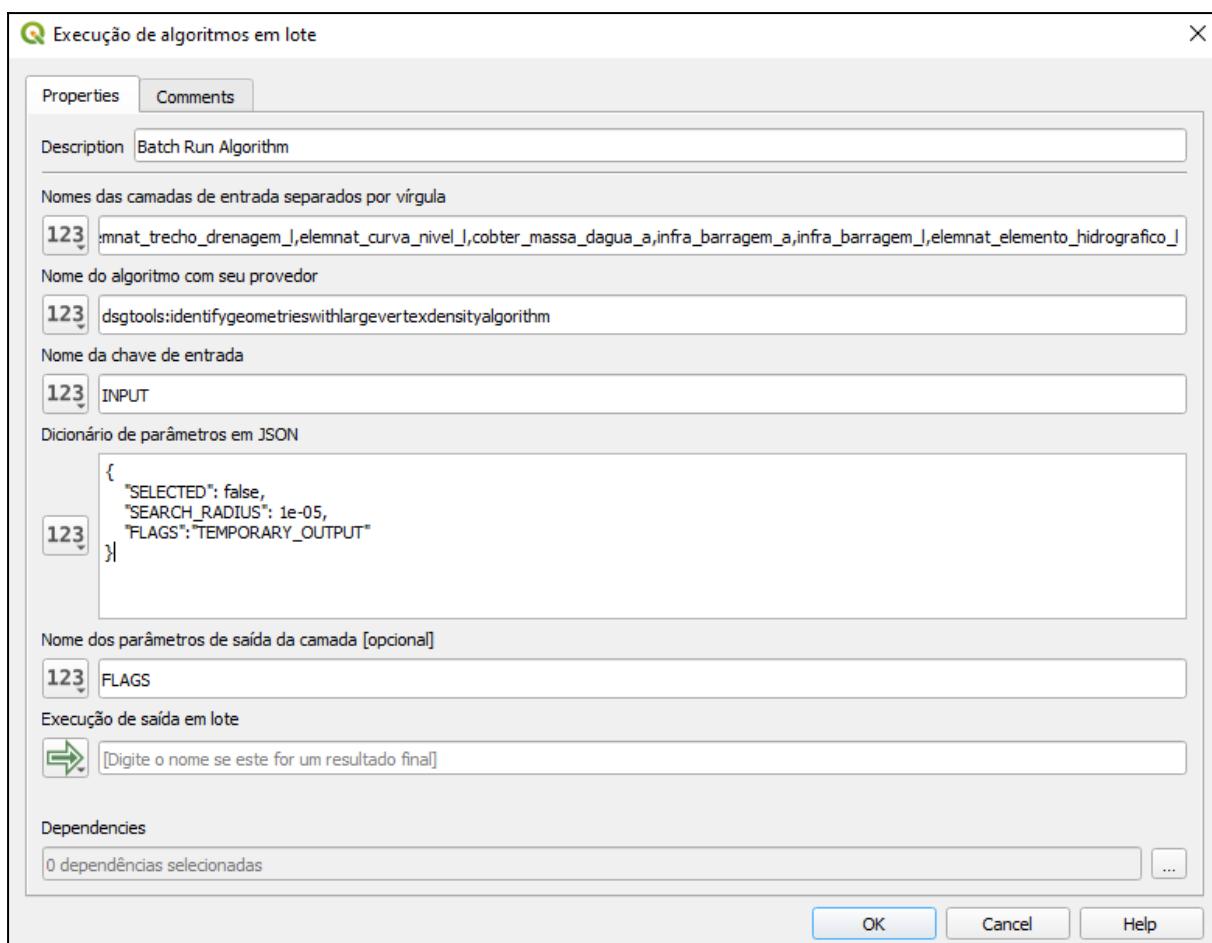


Figura 23: Extrato dos parâmetros.

### Camadas de entrada:

Tipo de geometria	Camadas de entrada
Linha	delimitador_massa_dagua_l, elemnat_trecho_drenagem_l, elemnat_curva_nivel_l, infra_barragem_l, elemnat_elemento_hidrografico_l.
Polígono	cobter_massa_dagua_a, infra_barragem_a, aux_moldura_a, moldura, aux_moldura_area_continua_a.

### Parâmetros em JSON:

```
{
    "SELECTED": false,
    "SEARCH_RADIUS": 1e-05,
    "FLAGS": "TEMPORARY_OUTPUT"
}
```

O parâmetro "**SELECTED**" determina se a identificação de erros deve ser realizada apenas nas geometrias selecionadas pelo usuário. Quando marcado como "*true*", a validação é realizada apenas nas geometrias selecionadas. Caso contrário, todas as geometrias são validadas.

O parâmetro "**SEARCH\_RADIUS**" do algoritmo determina a distância máxima que será percorrida para encontrar vértices próximos um dos outros. No modelo, o valor padrão do

raio de busca é definido como 0,00001 graus no sistema de coordenadas geográfico que corresponde a um valor aproximado de 1 metro. No entanto, é importante lembrar que esse valor pode ser ajustado de acordo com as necessidades específicas, considerando a escala e precisão dos dados utilizados no processamento.

O parâmetro "**FLAGS**" é utilizado para definir o nome da camada que será criada pela rotina para armazenar as informações de onde estão os erros. Neste caso, o valor "**TEMPORARY\_OUTPUT**" especifica que uma camada temporária será gerada para armazenar informações sobre a densidade incorreta de vértice.

**f) Nome da camada de *flags*: *flag\_densidade\_incorreta\_vertices*.**

**g) Resultado do processo e exemplo de erros:**

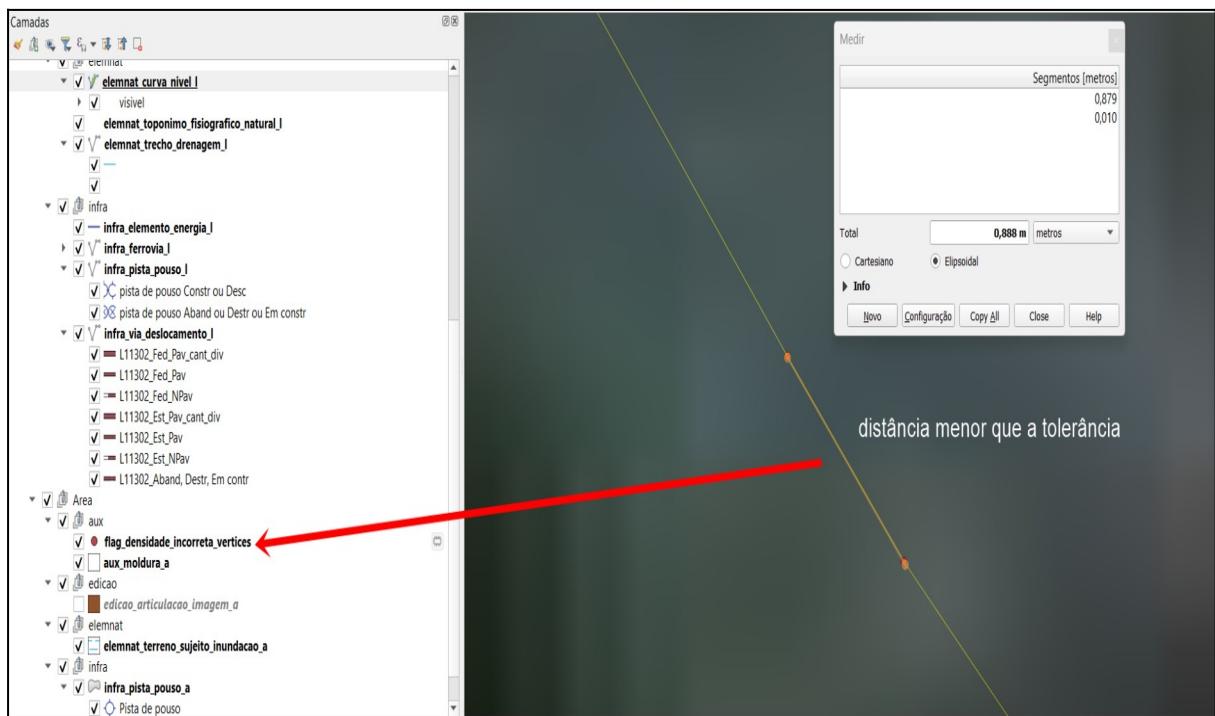


Figura 22: Exemplo de geometria com densidade incorreta de vértices.

## 8. IDENTIFICAR ÂNGULOS PEQUENOS

**a) Descrição:** Este algoritmo tem como propósito identificar ângulos menores do que o limite estabelecido na tolerância. Essa identificação é realizada nos vértices de feições, sejam elas polígonos ou linhas, que possuam ângulos entre vértices consecutivos inferiores ao limite predefinido.

**b) Arquivo:** identifica\_angulos\_pequenos\_alt\_hid\_carta\_orto.model3.

**c) Algoritmo:**

```
dsgtools:batchrunalgorithm;  
dsgtools:stringcsvtofirstlayerwithelementsalgorithm;  
native:extractbylocation;  
dsgtools:identifyoutofboundsangles.
```

**d) Composição do Modelo:**

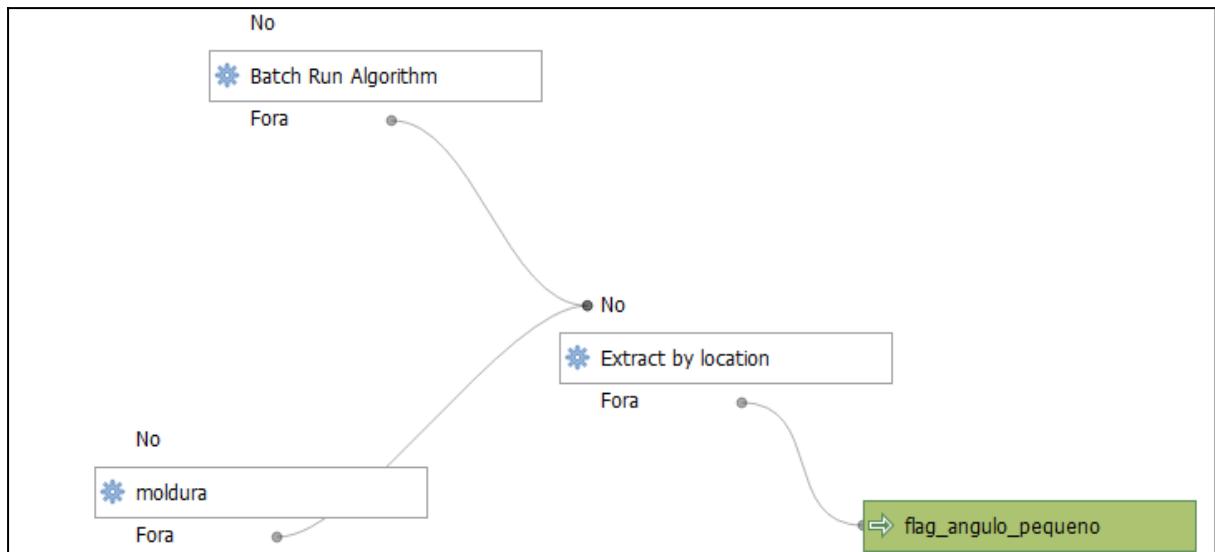


Figura 25: Modelo de identificação de ângulos pequenos.

## O modelo é composto por três etapas:

*BatchRunAlgorithm*: Executa o algoritmo *identifyoutofboundsangles* com os parâmetros selecionados na seção e.

Moldura: Converte uma *string csv* em uma camada especificada usando o algoritmo *stringcsvtofirstlayerwithelementsalgorithm*.

*ExtractByLocation*: Extrai os ângulos pequenos que estão dentro da moldura usando o algoritmo *extractbylocation*.

### e) Parâmetros:

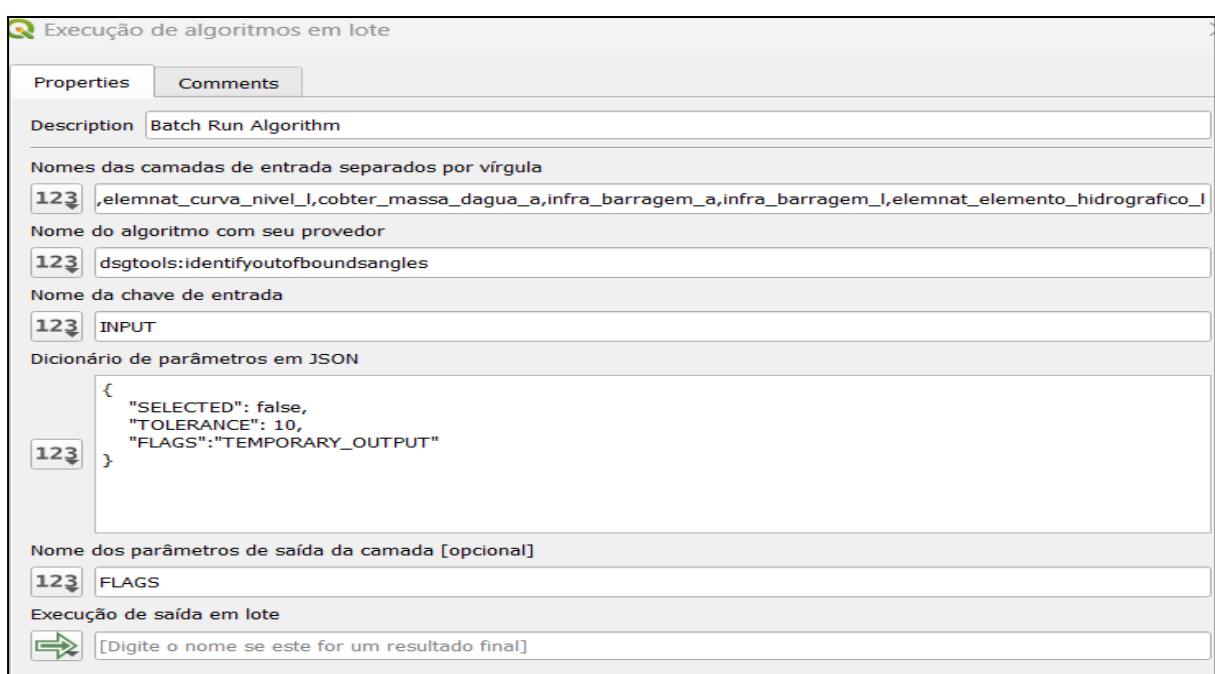


Figura 26: Extrato dos parâmetros.

### Camadas de entrada:

Tipo de geometria	Camadas de entrada
Linha	delimitador_massa_dagua_l, elemnat_trecho_drenagem_l, elemnat_curva_nivel_l, infra_barragem_l, elemnat_elemento_hidrografico_l.
Polígono	cobter_massa_dagua_a, infra_barragem_a, aux_moldura_a, moldura, aux_moldura_area_continua_a.

### Parâmetros em JSON:

```
{"SELECTED": false,
" TOLERANCE": 10,
"FLAGS": "TEMPORARY_OUTPUT"}
```

O parâmetro "**SELECTED**" determina se a validação deve ser realizada apenas nas geometrias selecionadas pelo usuário. Quando marcado como "true", a validação é realizada apenas nas geometrias selecionadas. Caso contrário, todas as geometrias são validadas.

No parâmetro "**TOLERANCE**" um ângulo de tolerância de 10 graus indica que qualquer ângulo abaixo desse valor será considerado pequeno. Essa medida é usada para identificar irregularidades na geometria, tais como curvas de nível muito acentuadas ou dobras em trechos de drenagem. Entretanto, é importante mencionar que para outras aplicações, uma tolerância maior ou menor pode ser mais apropriada.

O parâmetro "**FLAGS**" é utilizado para definir o nome da camada que será criada pela rotina para armazenar as informações de validação. Neste caso, o valor "**TEMPORARY\_OUTPUT**" especifica que uma camada temporária será gerada para armazenar informações sobre os ângulos pequenos encontrados durante a validação.

f) Nome da camada de *flags*: *Flag\_angulo\_pequeno*.

g) Resultado do processo e exemplo de erros:

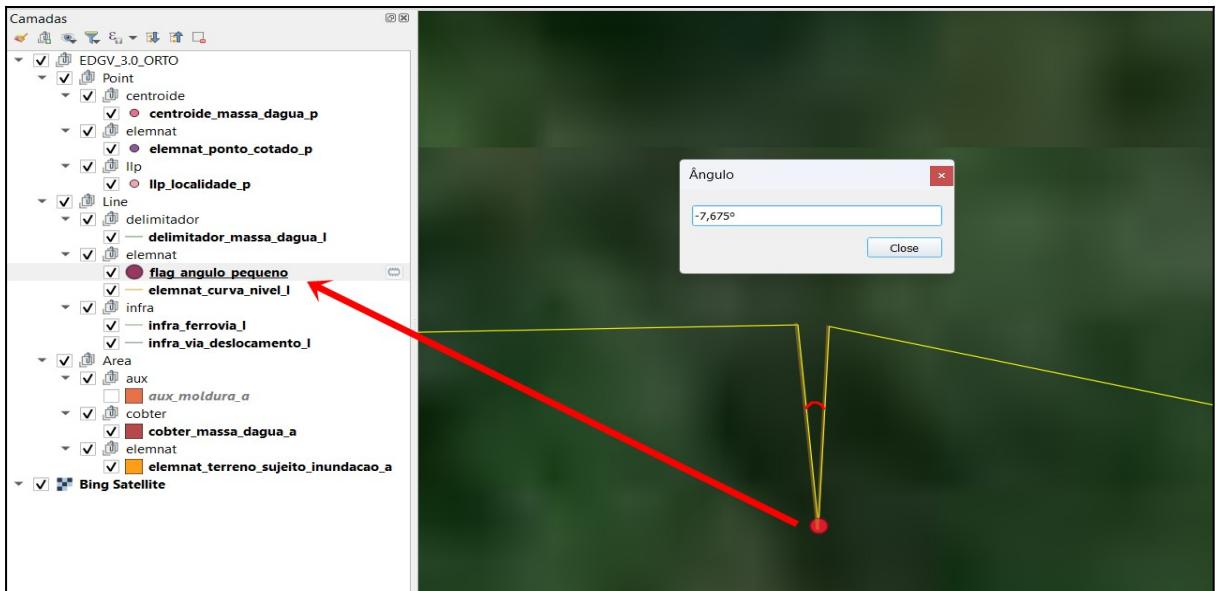


Figura 25: Exemplo de ângulo pequeno.

## 9. IDENTIFICAR ÂNGULOS PEQUENOS ENTRE CAMADAS

a) **Descrição:** Este algoritmo funciona de maneira similar à rotina anterior, identificando os ângulos pequenos em uma lista de camadas de linha. A diferença é que as camadas são unificadas, a fim de se comportarem como uma única camada.

b) **Arquivo:** *identificar\_angulos\_pequenos\_entre\_camadas.model3*.

c) **Algoritmo:**

```
dsgtools:stringcsvtolayerlistalgorithm;
native:extractbylocation;
dsgtools:identifyoutofboundsanglesincoverage;
dsgtools:stringcsvtofirstlayerwithelementsalgorithm.
```

#### d) Composição do Modelo:

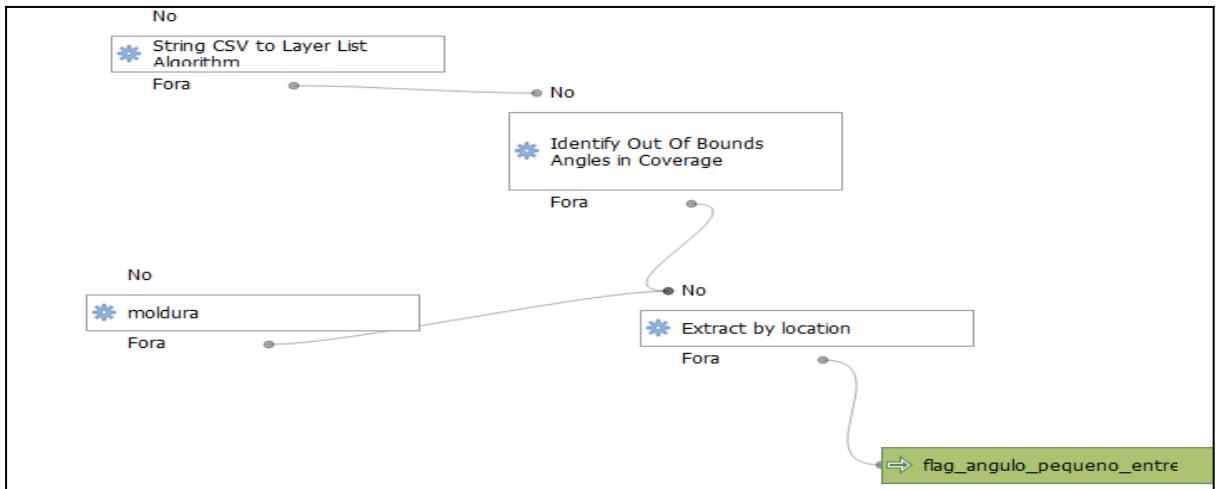


Figura 28: Modelo de identificação de ângulos pequenos entre camadas.

**O modelo é composto por quatro etapas:**

*StringCsvToLayerListAlgorithm*: Converte uma string CSV em uma lista de camadas.

*IdentifyOutOfBoundsAnglesInCoverage*: Identifica ângulos pequenos em cada camada de entrada, com base no parâmetro *TOLERANCE*.

Moldura: Converte uma string csv em uma camada especificada usando o algoritmo *stringcsvtofirstlayerwithelementsalgorithm*.

*ExtractByLocation*: Extrai os ângulos pequenos que estão dentro da área delimitada pela moldura usando o algoritmo *extractbylocation*.

### e) Parâmetros:

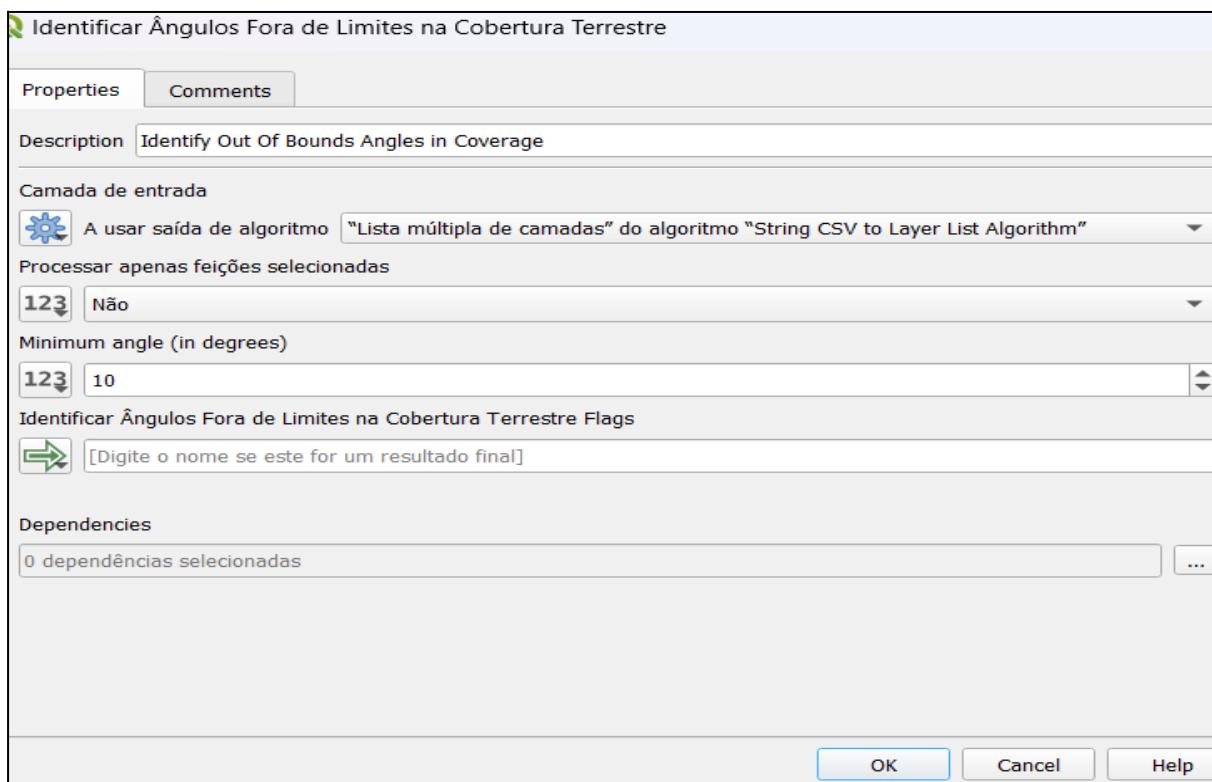


Figura 29: Extrato dos parâmetros.

### Camadas de entrada:

Tipo de geometria	Camadas de entrada
Linha	delimitador_massa_dagua_l, elemnat_trecho_drenagem_l, elemnat_curva_nivel_l, infra_barragem_l, elemnat_elemento_hidrografico_l.
Polígono	aux_moldura_a, moldura, aux_moldura_area_continua_a.

O parâmetro "**SELECTED**" determina se a validação deve ser realizada apenas nas geometrias selecionadas pelo usuário. Quando marcado como "*true*", a validação é realizada apenas nas geometrias selecionadas. Caso contrário, todas as geometrias são validadas.

No parâmetro "**TOLERANCE**" um ângulo de tolerância de 10 graus indica que qualquer ângulo abaixo desse valor será considerado pequeno. Essa medida é usada para identificar irregularidades na geometria, tais como curvas de nível muito acentuadas ou dobras em trechos de drenagem. Entretanto, é importante mencionar que para outras aplicações, uma tolerância maior ou menor pode ser mais apropriada.

O parâmetro "**FLAGS**" é utilizado para definir o nome da camada que será criada pela rotina para armazenar as informações de validação. Neste caso, o valor "**TEMPORARY\_OUTPUT**" especifica que uma camada temporária será gerada para armazenar informações sobre os ângulos pequenos encontrados durante a validação.

**f) Nome da camada de *flags*: *Flag\_angulo\_pequeno\_entre\_camadas*.**

**g) Resultado do processo e exemplo de erros:**

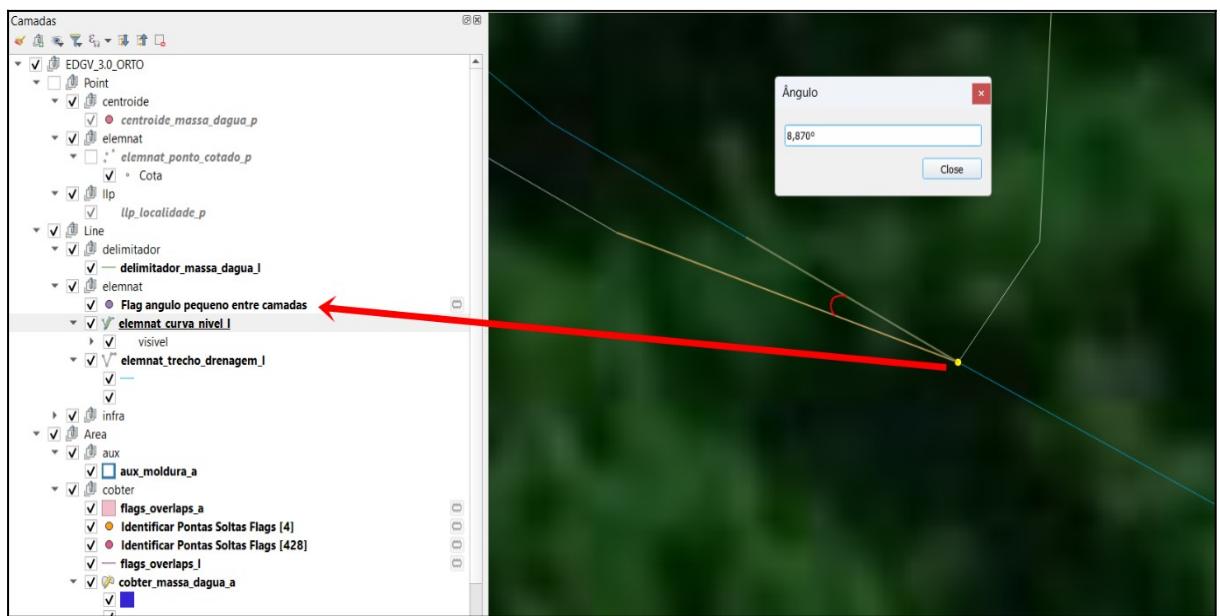


Figura 30: Exemplo de geometria com ângulo menor que a tolerância entre camadas distintas.

## 10. IDENTIFICAR Z

**a) Descrição:** O ângulo Z é a diferença entre ângulos formados por três pontos consecutivos. O modelo tem por objetivo encontrar esses ângulos em linhas e polígonos.

**b) Arquivo:** identifica\_z\_alt\_hid\_carta\_orto.model3.

**c) Algoritmo:**

```
dsgtools:stringcsvtolayerlistalgorithm;  
dsgtools:batchrunalgorithm;  
native:extractbylocation;  
dsgtools:identifyzanglesbetweenfeatures.
```

**d) Composição do Modelo:**

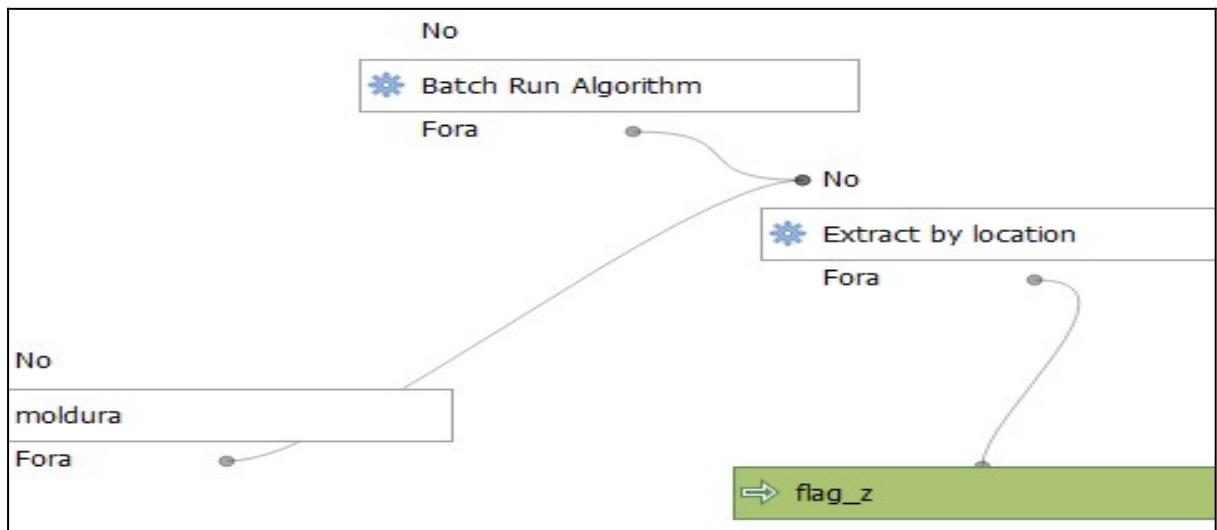


Figura 31: Modelo de identificação de ângulo em Z.

**O modelo é composto por três etapas:**

*Batchrunalgorithm:* executa o algoritmo *identifyzanglesbetweenfeatures* com os parâmetros selecionados na seção e.

Moldura: Converte uma *string csv* em uma camada especificada usando o algoritmo *stringcsvtofirstlayerwithelementsalgorithm*.

*ExtractByLocation*: Extrai os ângulos pequenos que estão dentro da área delimitada pela moldura usando o algoritmo *extractbylocation*.

#### e) Parâmetros:

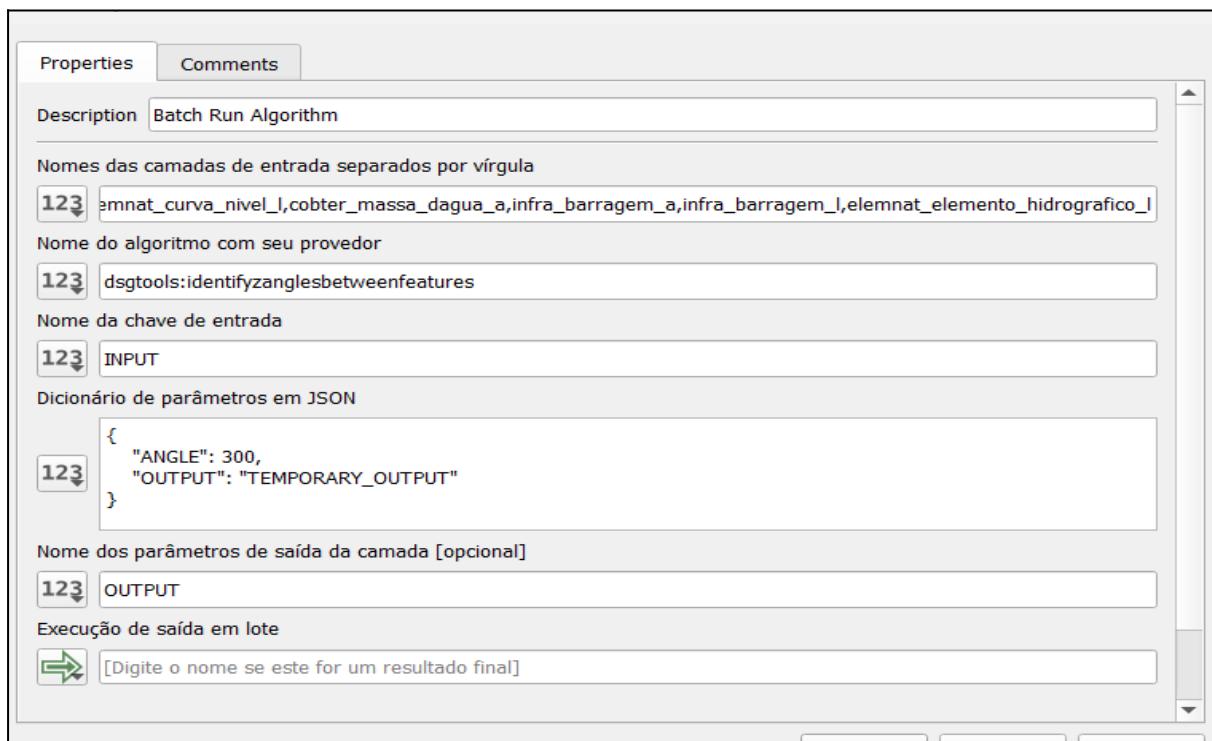


Figura 32: Extrato dos parâmetros.

#### Camadas de entrada:

Tipo de geometria	Camadas de entrada
Linha	delimitador_massa_dagua_l, elemnat_trecho_drenagem_l, elemnat_curva_nivel_l, infra_barragem_l, elemnat_elemento_hidrografico_l.

Polígono	cobter_massa_dagua_a, infra_barragem_a, aux_moldura_a, moldura, aux_moldura_area_continua_a.
----------	--

**Parâmetros em JSON:**

```
{"ANGLE": 300,
"OUTPUT":"TEMPORARY_OUTPUT"}
```

O parâmetro "**ANGLE**" indica a medida de um ângulo em graus. Quando definido como 300 graus, o algoritmo calculará a diferença necessária para completar 360 graus, ou seja, 60 graus. Assim, se as duas partes do ângulo em Z forem inferiores a 60 graus, elas serão consideradas como um ângulo em Z. Essa medida é usada para identificar irregularidades na geometria, entretanto, é importante mencionar que para outras aplicações, uma tolerância maior ou menor pode ser mais apropriada.

O Parâmetro "**OUTPUT**" é utilizado para definir o nome da camada que será criada pela rotina para armazenar as informações de validação. Neste caso, o valor "**TEMPORARY\_OUTPUT**" especifica que uma camada temporária será gerada para armazenar informações sobre os ângulos inválidos encontrados durante a validação.

f) nome da camada de *flags*: *Flag\_z*.

### g) Resultado do processo e exemplo de erros:

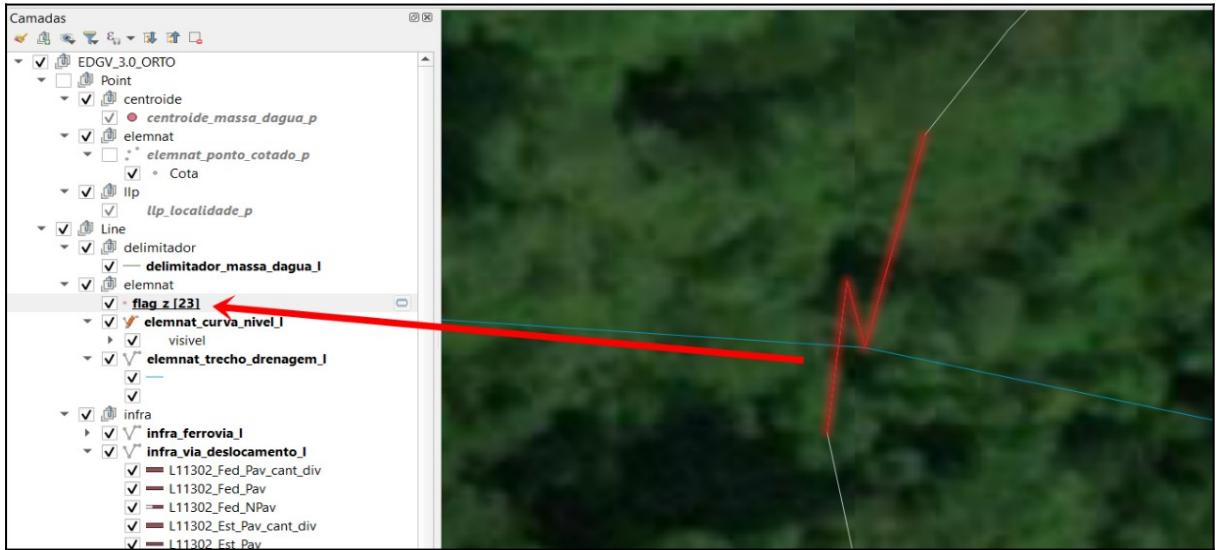


Figura 33: Exemplo de geometria com ângulo em formato Z.

## 11. IDENTIFICAR *OVERLAPS* DENTRO DA MESMA CAMADA

**a) Descrição:** Este algoritmo identifica sobreposições dentro da mesma camada em uma lista de camadas de linha e polígono. Para as camadas de linha, considera-se sobreposição quando há dois vértices compartilhados na mesma camada.

**b) Arquivo:** identifica\_overlaps\_dentro\_da\_mesma\_camada.model3.

**c) Algoritmo:**

```
dsgtools:stringcsvtofirstlayerwithelementsalgorithm;  
dsgtools:batchrunalgorithm;  
native:extractbylocation;  
dsgtools:identifyoverlaps.
```

#### d) Composição do Modelo:

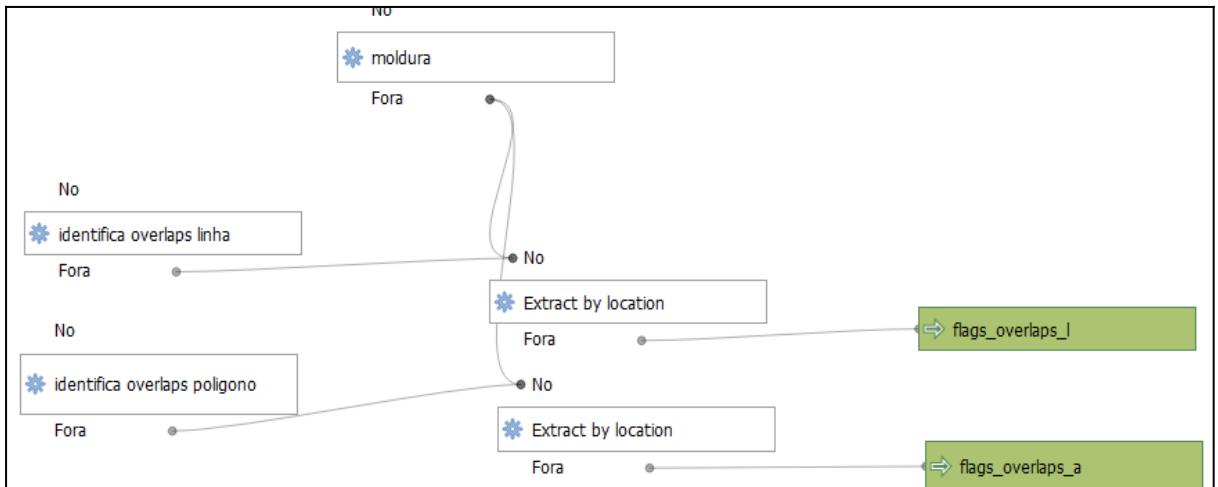


Figura 34: Modelo de identificação de sobreposições dentro da mesma camada.

#### O modelo é composto por três etapas:

*Batchrunalgorithm:* Executa o algoritmo `identifyoverlaps` com os parâmetros selecionados abaixo para as camadas de linha e separadamente executa para polígonos.

*Moldura:* Converte uma `string csv` em uma camada especificada usando o algoritmo `stringcsvtofirstlayerwithelementsalgorithm`.

*ExtractByLocation:* Extrai os ângulos pequenos que estão dentro da área delimitada pela moldura usando o algoritmo `extractbylocation`.

### e) Parâmetros:

**Properties** **Comments**

Description `identifica overlaps linha`

Nomes das camadas de entrada separados por vírgula  
`sa_dagua_l,elemnat_trecho_drenagem_l,elemnat_curva_nivel_l,infra_barragem_l,elemnat_elemento_hidrografico_l`

Nome do algoritmo com seu provedor  
`dsgtools:identifyoverlaps`

Nome da chave de entrada  
`INPUT`

Dicionário de parâmetros em JSON  
`{
 "SELECTED": false,
 "FLAGS": "TEMPORARY_OUTPUT"
}`

Nome dos parâmetros de saída da camada [opcional]  
`FLAGS`

Execução de saída em lote  
 [Digite o nome se este for um resultado final]

Figura 35: Extrato dos parâmetros.

### Camadas de entrada:

Tipo de geometria	Camadas de entrada
Linha	<code>delimitador_massa_dagua_l,</code> <code>elemnat_trecho_drenagem_l,</code> <code>elemnat_curva_nivel_l,</code> <code>infra_barragem_l,</code> <code>elemnat_elemento_hidrografico_l.</code>
Polígono	<code>cobter_massa_dagua_a,</code> <code>infra_barragem_a,</code> <code>aux_moldura_a,</code> <code>moldura,</code> <code>aux_moldura_area_continua_a.</code>

### Parâmetros em JSON:

```
{"SELECTED": false,  
"FLAGS": "TEMPORARY_OUTPUT"}
```

O parâmetro "**SELECTED**" determina se a validação deve ser realizada apenas nas geometrias selecionadas pelo usuário. Quando marcado como "*true*", a validação é realizada apenas nas geometrias selecionadas. Caso contrário, todas as geometrias são validadas.

"**FLAGS**" é utilizado para definir o nome da camada que será criada pela rotina para armazenar as informações de validação. Neste caso, o valor "**TEMPORARY\_OUTPUT**" especifica que uma camada temporária será gerada para armazenar informações sobre os *overlaps* encontradas durante a validação.

### f) nome da camada de *flags*:

*flags\_overlaps\_l*;

*flags\_overlaps\_a*.

### g) Resultado do processo e exemplo de erros:

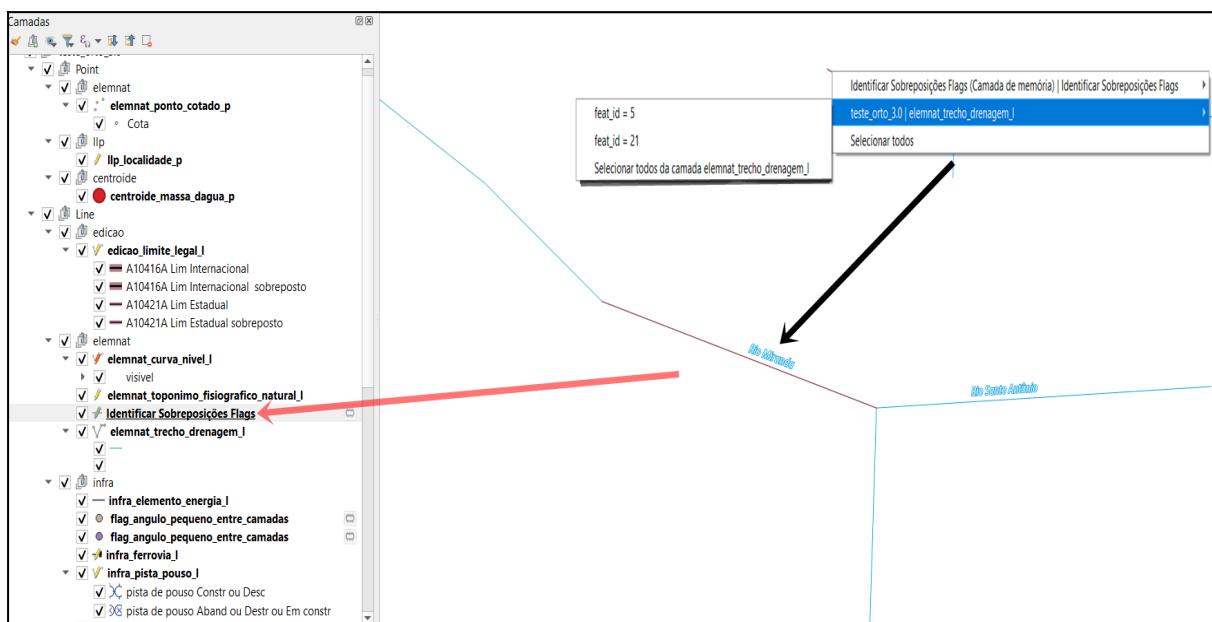


Figura 36: Exemplo de sobreposição.

## 12. IDENTIFICAR *UNDERSHOOT* COM MOLDURA E CONEXÃO DE LINHAS

**a) Descrição:** Este algoritmo identifica pontas soltas dentro da mesma camada em uma lista de camadas de linha e polígono.

**b) Arquivo:** identifica\_undershoot\_alt\_hid\_carta\_orto.model3.

**c) Algoritmo:**

```
dsgtools:stringcsvtofirstlayerwithelementsalgorithm;  
dsgtools:identifypolygonundershoots;  
native:mergevectorlayers;  
dsgtools:identifydangles.
```

**d) Composição do Modelo:**

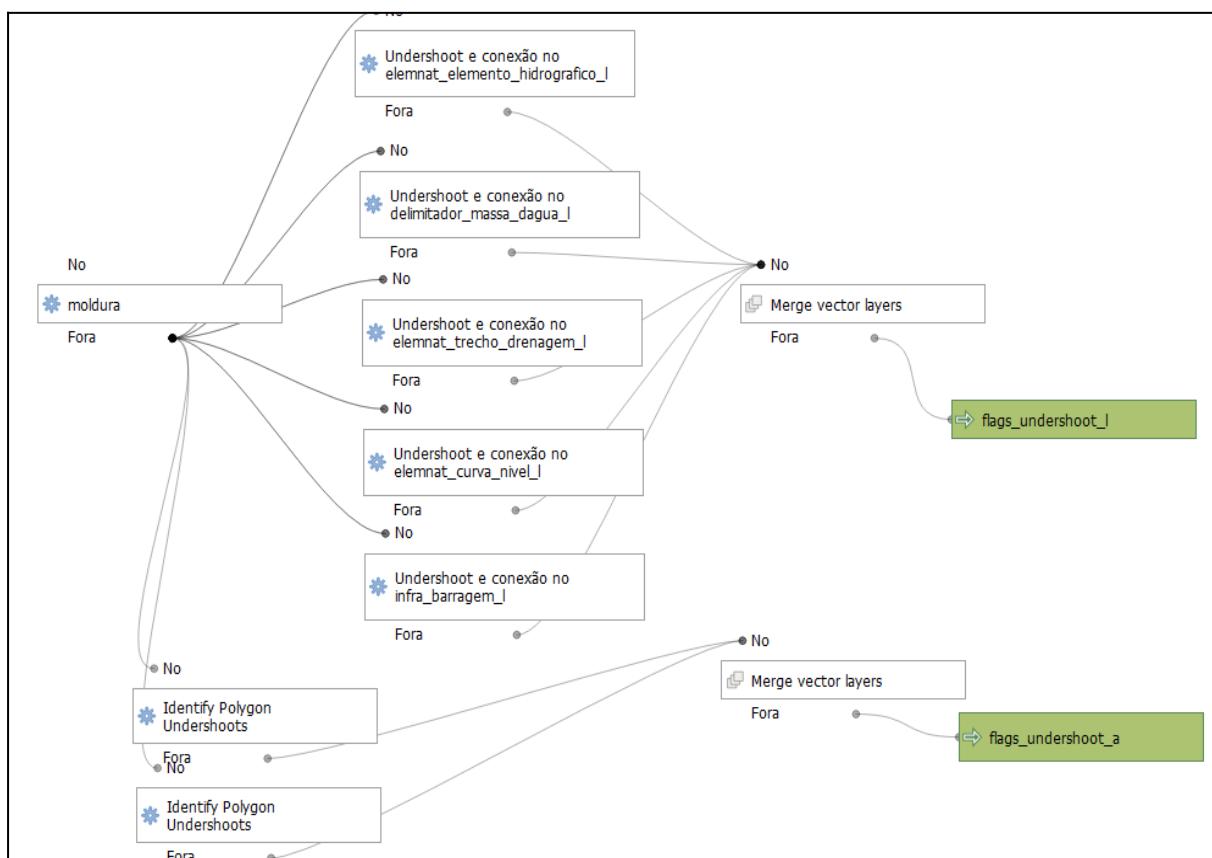


Figura 37: Modelo de identificação de pontas soltas.

## O modelo é composto por quatro etapas:

Moldura: Converte uma *string* csv em uma camada especificada usando o algoritmo *stringcsvtofirstlayerwithelementsalgorithm*.

*Undershoot e conexão*: Executa o algoritmo *identifydangles* para as camadas de linha com os parâmetros descritos na seção e.

*Identify Polygon Undershoots*: Executa o algoritmo *identifypolygonundershoots* para as camadas de polígono com os parâmetros descritos na seção e

*Merge Vector Layers*: Realiza uma operação de unir as camadas de saída utilizando o algoritmo *mergevectorlayers*.

### e) Parâmetros:

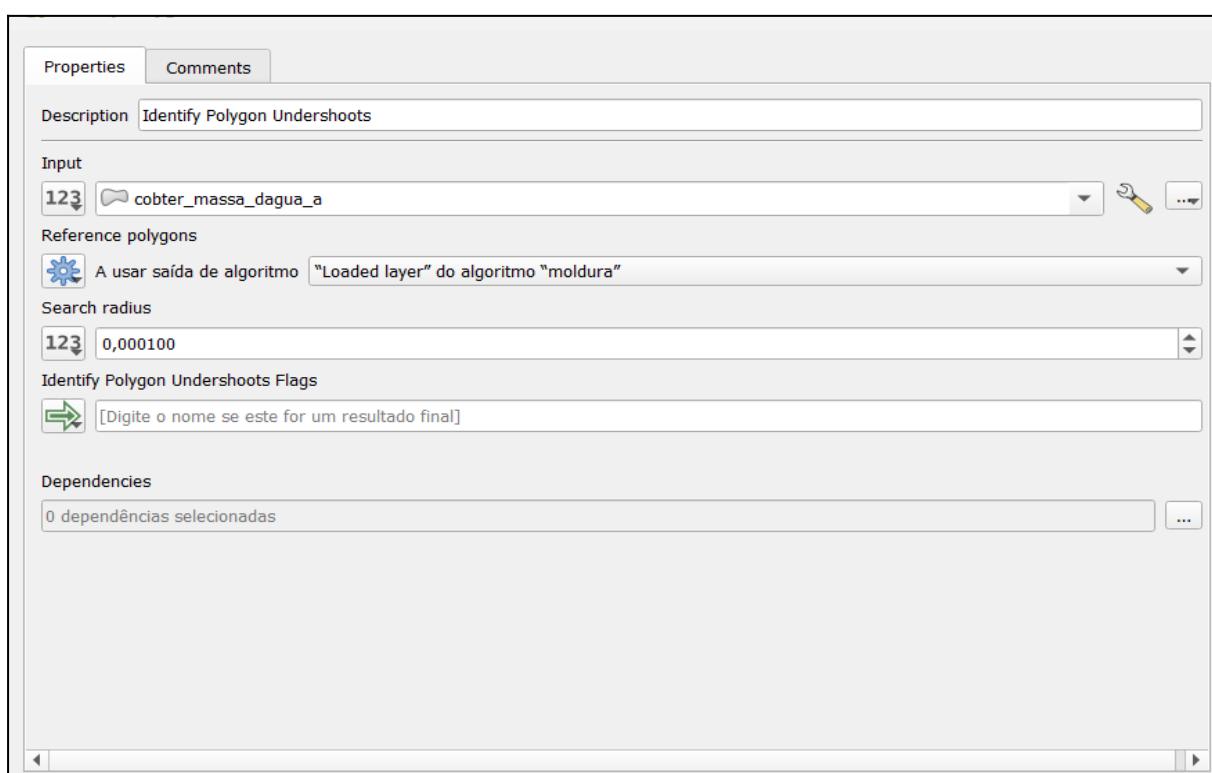


Figura 38: Extrato dos parâmetros.

### Camadas de entrada:

Tipo de geometria	Camadas de entrada
Polígono	cobter_massa_dagua_a, infra_barragem_a, aux_moldura_a, moldura, aux_moldura_area_continua_a.

O parâmetro "**INPUT**" determina se a validação deve ser realizada apenas nas geometrias selecionadas pelo usuário.

O parâmetro "**REFERENCE POLYGONS**" utiliza a saída do algoritmo anterior que basicamente converte uma lista csv em camadas, no caso em específico utiliza a moldura para delimitar a área de interesse.

O Parâmetro "**SEARCH RADIUS**" do algoritmo determina a distância máxima que será percorrida para encontrar pontas soltas em relação aos segmentos de linha e polígonos que se cruzam nas intersecções. No modelo, o valor padrão do raio de busca é definido como 0,00001 graus no sistema de coordenadas geográfico que corresponde a um valor aproximado de 1 metro. No entanto, é importante lembrar que esse valor pode ser ajustado de acordo com as necessidades específicas, considerando a escala e precisão dos dados utilizados no processamento.

O parâmetro "**FLAGS**" é utilizado para definir o nome da camada que será criada pela rotina para armazenar as informações de validação.

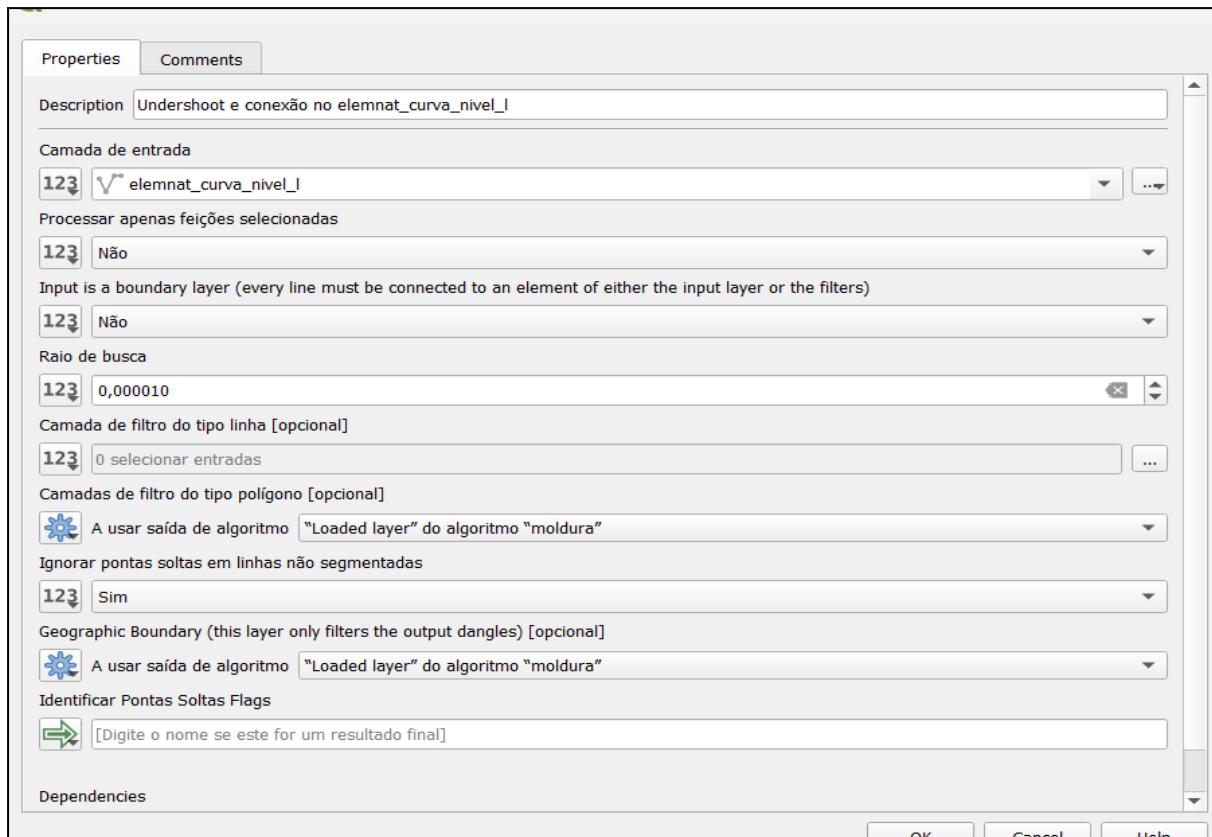


Figura 39: Extrato dos parâmetros.

### Camadas de entrada:

Tipo de geometria	Camadas de entrada
Linha	delimitador_massa_dagua_l, elemnat_trecho_drenagem_l, elemnat_curva_nivel_l, infra_barragem_l, elemnat_elemento_hidrografico_l.
Polígono	aux_moldura_a, moldura, aux_moldura_area_continua_a.

O parâmetro "**INPUT**" determina se a validação deve ser realizada apenas nas geometrias selecionadas pelo usuário.

O parâmetro "**SELECTED**" determina se a validação deve ser realizada apenas nas geometrias selecionadas pelo usuário. Quando marcado como "*true*", a validação é realizada apenas nas geometrias selecionadas. Caso contrário, todas as geometrias são validadas.

No parâmetro "**INPUT IS A BOUNDARY LAYER**" se a opção "*false*" estiver definida no algoritmo, isso significa que a moldura não precisa estar conectada a elementos da camada de entrada ou dos filtros. É importante ressaltar que o campo elemnat\_curva\_nivel a opção precisa ser "*true*", por se tratar de um contorno e ter um comportamento particular.

O Parâmetro "**SEARCH RADIUS**" do algoritmo determina a distância máxima que será percorrida para encontrar pontas soltas em relação aos segmentos de linha e polígonos que se cruzam nas intersecções. No modelo, o valor padrão do raio de busca é definido como 0,00001 graus no sistema de coordenadas geográfico que corresponde a um valor aproximado de 1 metro. No entanto, é importante lembrar que esse valor pode ser ajustado de acordo com as necessidades específicas, considerando a escala e precisão dos dados utilizados no processamento

O parâmetro "**CAMADA DE FILTRO**" tem como proposta identificar pontas soltas próximas às camadas especificadas. Essas pontas soltas serão adicionadas à regra de verificação de proximidade durante a busca por feições próximas às analisadas, a fim de minimizar a ocorrência de falsos positivos.

O parâmetro "**IGNORAR PONTAS SOLTAS EM LINHAS NÃO SEGMENTADAS**" quando marcado como "*true*" permite que o algoritmo ignore pontas soltas em linhas não segmentadas. Ou seja, ele não considera como *undershoots* aquelas pontas soltas que não estão conectadas a outros segmentos de linha. Ao ignorar essas pontas, o algoritmo pode concentrar-se apenas nas linhas que realmente importam.

O parâmetro "**GEOGRAPHIC BOUNDARY**" se refere à delimitação geográfica para a execução do processamento em uma região determinada. A "moldura" representa a área delimitada em questão, isso evita que sejam incluídas informações irrelevantes ou imprecisas de outras áreas que estão fora da delimitação.

O parâmetro "**FLAGS**" é utilizado para definir o nome da camada que será criada pela rotina para armazenar as informações de validação.

**f) nome da camada de *flags*:**

*flags\_undershoot\_a;*

*flags\_undershoot\_l.*

**g) Resultado do processo e exemplo de erros:**

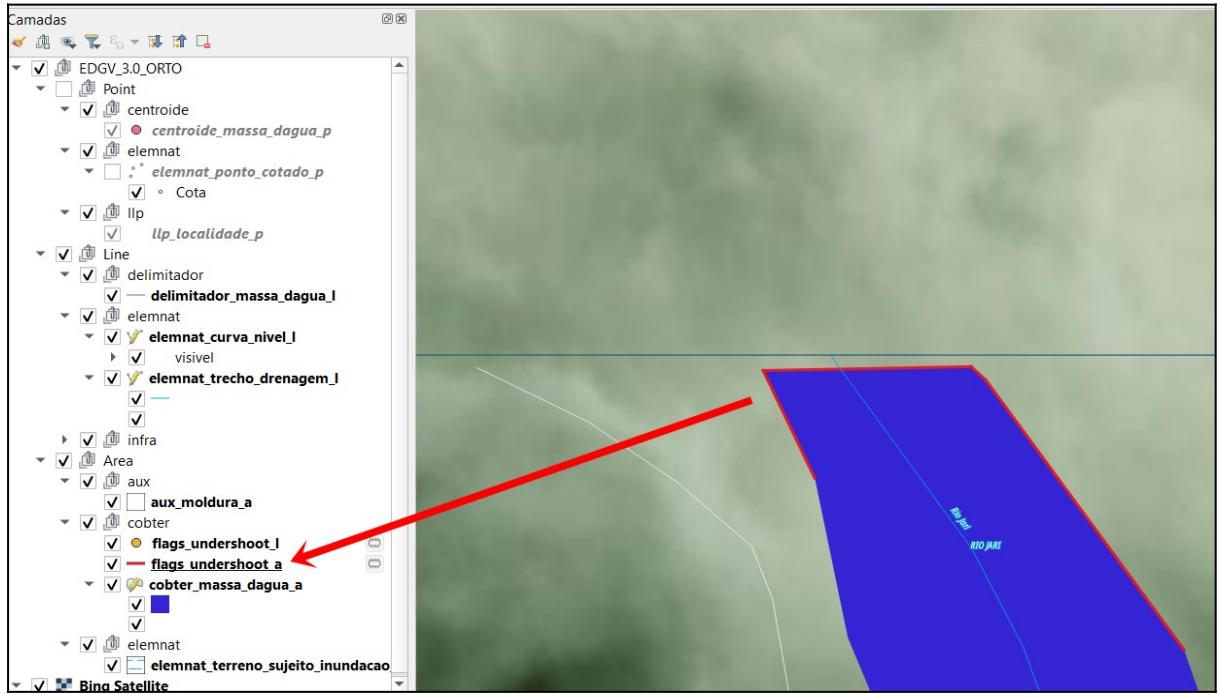


Figura 40: Exemplo de área não conectada a moldura.



Figura 41: Exemplo de pontas soltas.

### 13. IDENTIFICAR LINHAS SEGMENTADAS COM MESMO CONJUNTO DE ATRIBUTOS

**a) Descrição:** Este algoritmo identifica linhas segmentadas com mesmo conjunto de atributos em uma lista de camadas de linha.

**b) Arquivo:**

identifica\_linhas\_segmentadas\_com\_mesmo\_conjunto\_de\_atributos.model3.

**c) Algoritmo:**

```
dsgtools:stringcsvtolayerlistalgorithm;
dsgtools:stringcsvtofirstlayerwithelementsalgorithm;
native:mergevectorlayers;
native:extractbylocation.
dsgtools:identifyunmergedlineswithsameattributeset.
```

#### d) Composição do Modelo:

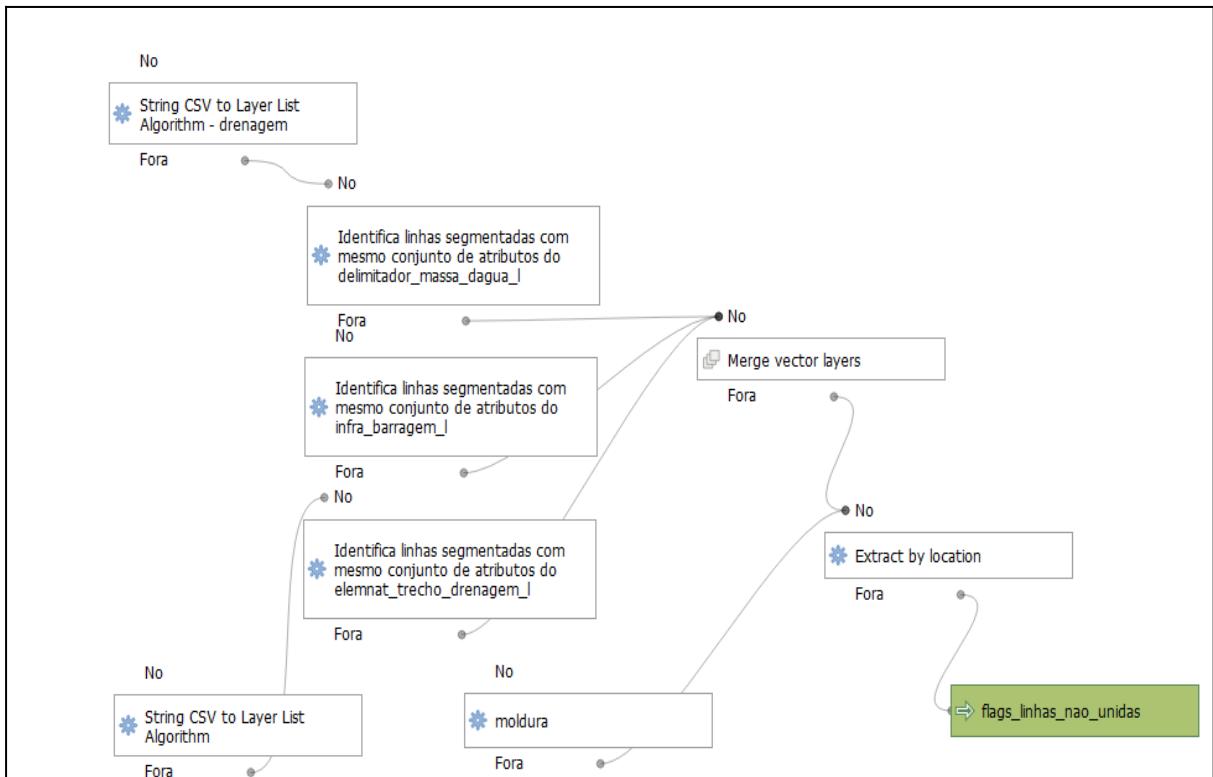


Figura 42: Modelo de identificação de linhas segmentadas com o mesmo conjunto de atributos.

**O modelo é composto por cinco etapas:**

*String CSV to Layer List Algorithm:* Converte uma string csv em uma camada especificada usando o algoritmo *stringcsvtolayerlistalgorithm*.

**Identificar linhas Segmentadas:** Identifica as linhas segmentadas com os parâmetros definidos abaixo utilizando o algoritmo *identifyunmergedlineswithsameatributeset*.

*Merge Vector Layers:* Realiza uma operação de unir as camadas de saída utilizando o algoritmo *mergevectorlayers*.

**Moldura:** Converte uma string csv em uma camada especificada usando o algoritmo *stringcsvtofirstlayerwithelementsalgorithm*.

*ExtractByLocation*: Extrai os ângulos pequenos que estão dentro da área delimitada pela moldura usando o algoritmo *extractbylocation*.

### e) Parâmetros:

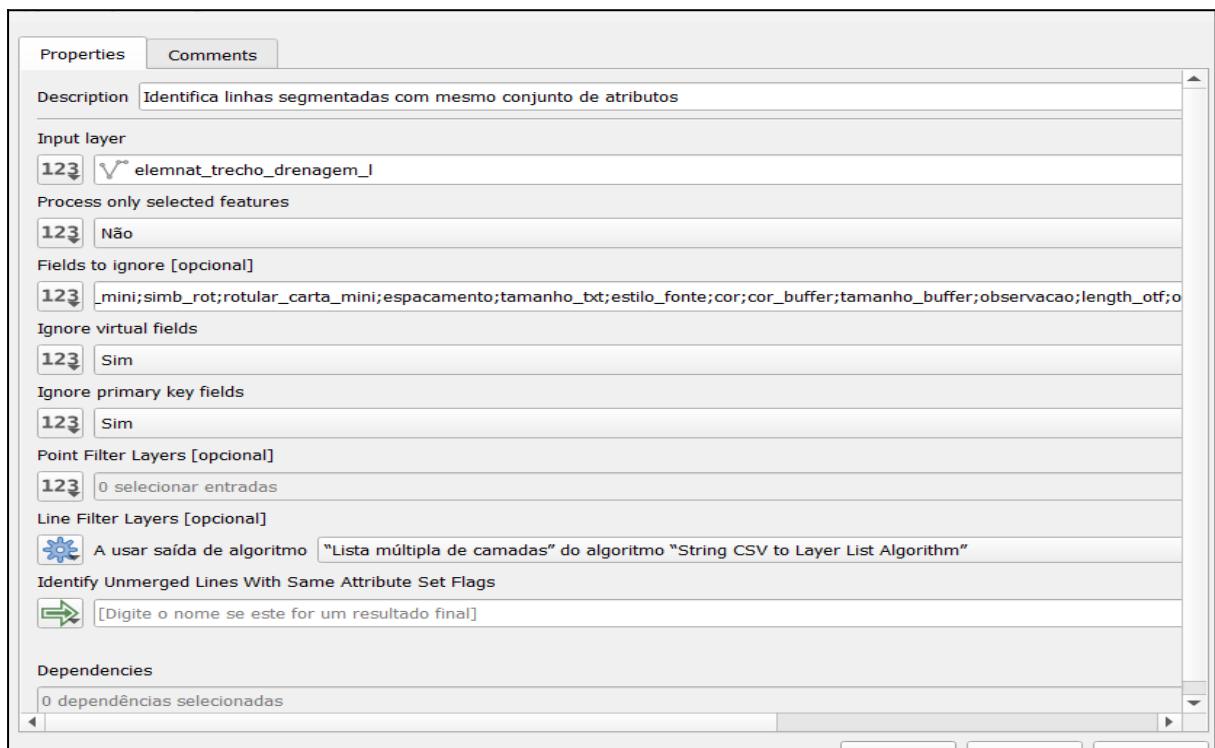


Figura 43: Extrato dos parâmetros.

### Camadas de entrada:

Tipo de geometria	Camadas de entrada
Linha	delimitador_massa_dagua_l, elemnat_trecho_drenagem_l, infra_barragem_l.
Polígono	aux_moldura_a, moldura, aux_moldura_area_continua_a.

O parâmetro "**SELECTED**" determina se a validação deve ser realizada apenas nas geometrias selecionadas pelo usuário. Quando marcado como "true", a validação é realizada apenas nas geometrias selecionadas. Caso contrário, todas as geometrias são validadas.

No parâmetro "**FIELDS TO IGNORE**" é definido uma lista de campos a ignorar de atributos (*ATTRIBUTE\_BLACKLIST*) que não serão considerados na comparação das feições, a fim de evitar falsos positivos.

O parâmetro "**IGNORE\_PK\_FIELDS**" é um booleano que indica se os campos da chave primária das camadas de entrada devem ser ignorados na comparação. (apesar de estar marcado como "true", o campo de chave primária "id" já está sendo ignorado na lista negra de atributos).

O parâmetro "**IGNORE\_VIRTUAL\_FIELDS**" é um booleano que indica se os campos virtuais (calculados dinamicamente) devem ser ignorados na comparação.

O parâmetro "**FLAGS**" é utilizado para definir o nome da camada que será criada pela rotina para armazenar as informações de validação.

**f) nome da camada de flags: *flags\_linhas\_nao\_unidas*.**

**g) Resultado do processo e exemplo de erros:**

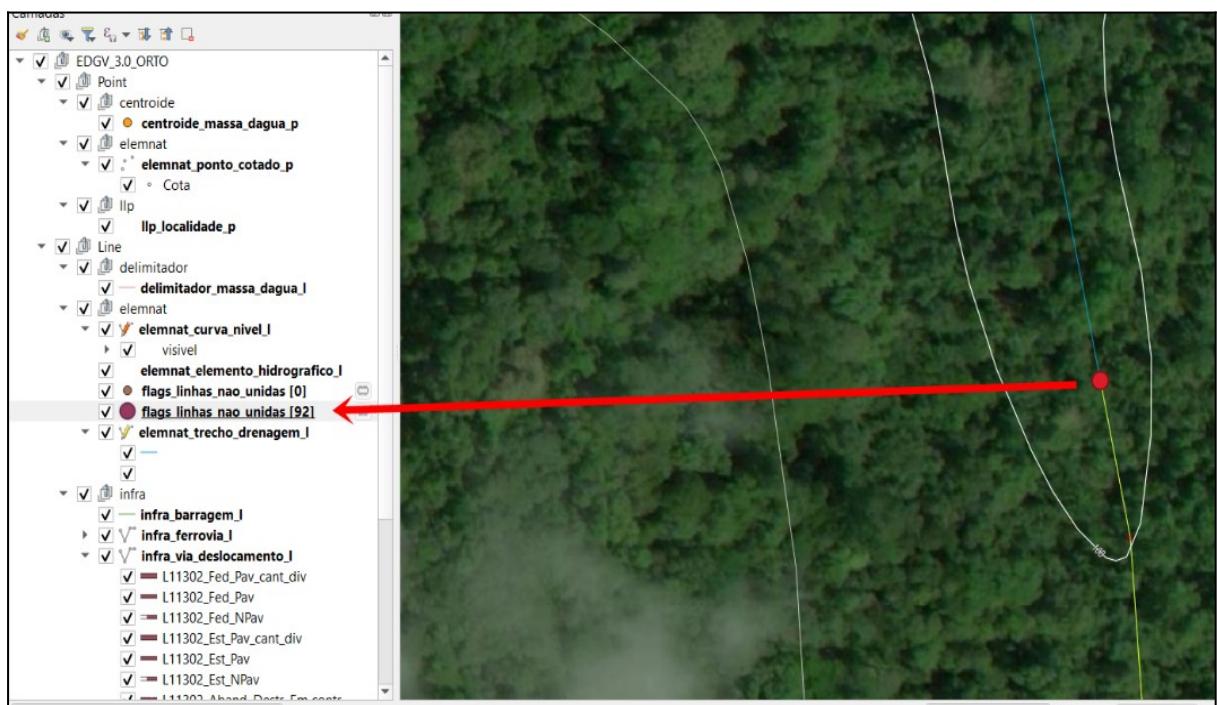


Figura 44: Exemplo de linhas segmentadas sem motivo.

## 14. IDENTIFICAR LINHAS NÃO SEGMENTADAS NAS INTERSECÇÕES

**a) Descrição:** O algoritmo em questão tem como finalidade detectar linhas não segmentadas nas intersecções de camadas. Essa tarefa é realizada através da identificação de geometrias não conectadas que não compartilham vértices. A partir dessa análise, é possível determinar se as linhas com vértices não conectados em sua extensão devem ser classificadas como "*dangles*" (pontas soltas) e, portanto, descartadas, ou se devem ser consideradas como não segmentadas dentro das camadas especificadas em uma lista de camadas de linha.

**b) Arquivo:** `identificar_linhas_nao_segmentadas_nas_intersecoes_alt_hid.model3`.

**c) Algoritmo:**

```
dsgtools:stringcsvtolayerlistalgorithm;  
dsgtools:stringcsvtofirstlayerwithelementsalgorithm;  
dsgtools:identifydanglesalgorithm.
```

**d) Composição do Modelo:**

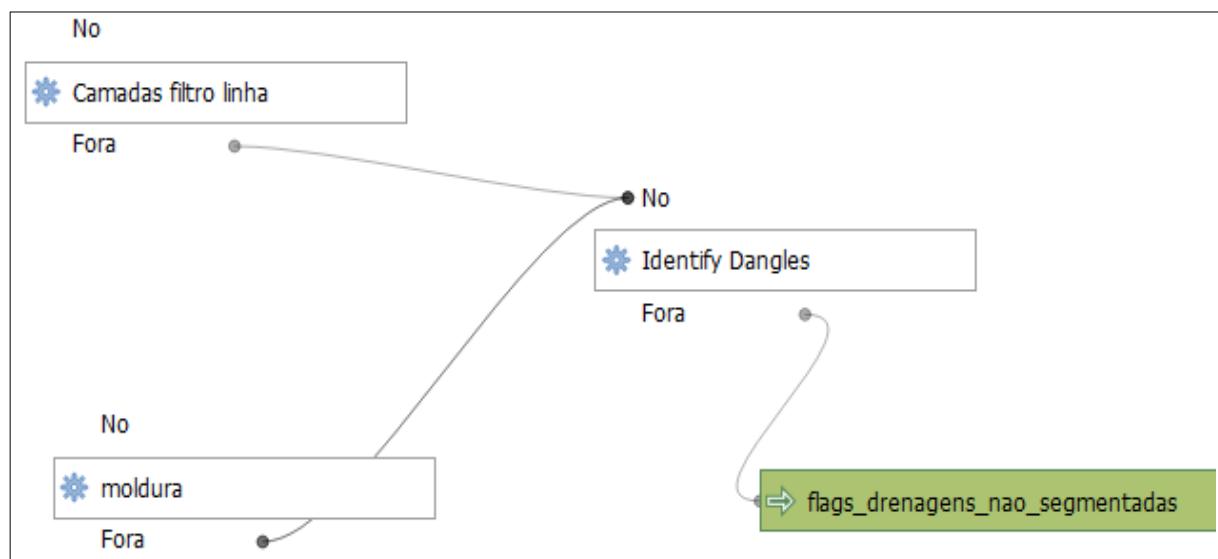


Figura 45: Modelo de identificação de linhas não segmentadas nas intersecções.

## O modelo é composto por três etapas:

Camadas filtro linha: Converte uma *string* csv em uma lista de camadas usando o algoritmo *stringcsvtolayerlistalgorithm*.

Moldura: Converte uma *string* csv em uma camada especificada usando o algoritmo *stringcsvtofirstlayerwithelementsalgorithm*.

*Identify Dangles*: Identifica pontas soltas na camada de *elemnat\_trecho\_drenagem\_1* usando o algoritmo *identifydanglesalgorithm*.

### e) Parâmetros:



Figura 46: Extrato dos parâmetros.

## Camadas de entrada:

Tipo de geometria	Camadas de entrada
Linha	elemnat_trecho_drenagem_1 delimitador_massa_dagua_1, infra_barragem_1.
Polígono	aux_moldura_a, moldura, aux_moldura_area_continua_a.

O parâmetro "**INPUT**" determina se a validação deve ser realizada apenas nas geometrias selecionadas pelo usuário.

O parâmetro "**SELECTED**" determina se a validação deve ser realizada apenas nas geometrias selecionadas pelo usuário. Quando marcado como "*true*", a validação é realizada apenas nas geometrias selecionadas. Caso contrário, todas as geometrias são validadas.

No parâmetro "**INPUT IS A BOUNDARY LAYER**" se a opção "*false*" estiver definida no algoritmo, isso significa que a moldura não precisa estar conectada a elementos da camada de entrada ou dos filtros.

O Parâmetro "**SEARCH RADIUS**" do algoritmo determina a distância máxima que será percorrida para encontrar pontas soltas em relação aos segmentos de linha que se cruzam nas intersecções. No modelo, o valor padrão do raio de busca é definido como 0,00001 graus no sistema de coordenadas geográfico que corresponde a um valor aproximado de 1 metro. No entanto, é importante lembrar que esse valor pode ser ajustado de acordo com as necessidades específicas, considerando a escala e precisão dos dados utilizados no processamento.

O parâmetro "**CAMADA DE FILTRO**" é usado para limitar as camadas que serão analisadas no algoritmo.

O parâmetro "**IGNORAR PONTAS SOLTAS EM LINHAS NÃO SEGMENTADAS**" quando marcado como "*false*" permite que o algoritmo encontre pontas soltas em linhas não segmentadas, ou seja, considera como *undershoots* aquelas pontas soltas que não estão conectadas a outros segmentos de linha.

O parâmetro "**GEOGRAPHIC BOUNDARY**" se refere à delimitação geográfica para a execução do processamento em uma região determinada. A "moldura" representa a área delimitada em questão, isso evita que sejam incluídas informações irrelevantes ou imprecisas de outras áreas que estão fora da delimitação.

O parâmetro "**FLAGS**" é utilizado para definir o nome da camada que será criada pela rotina para armazenar as informações de validação.

**f) nome da camada de flags:** *flags\_drenagens\_nao\_segmentadas*.

**g) Resultado do processo e exemplo de erros:**

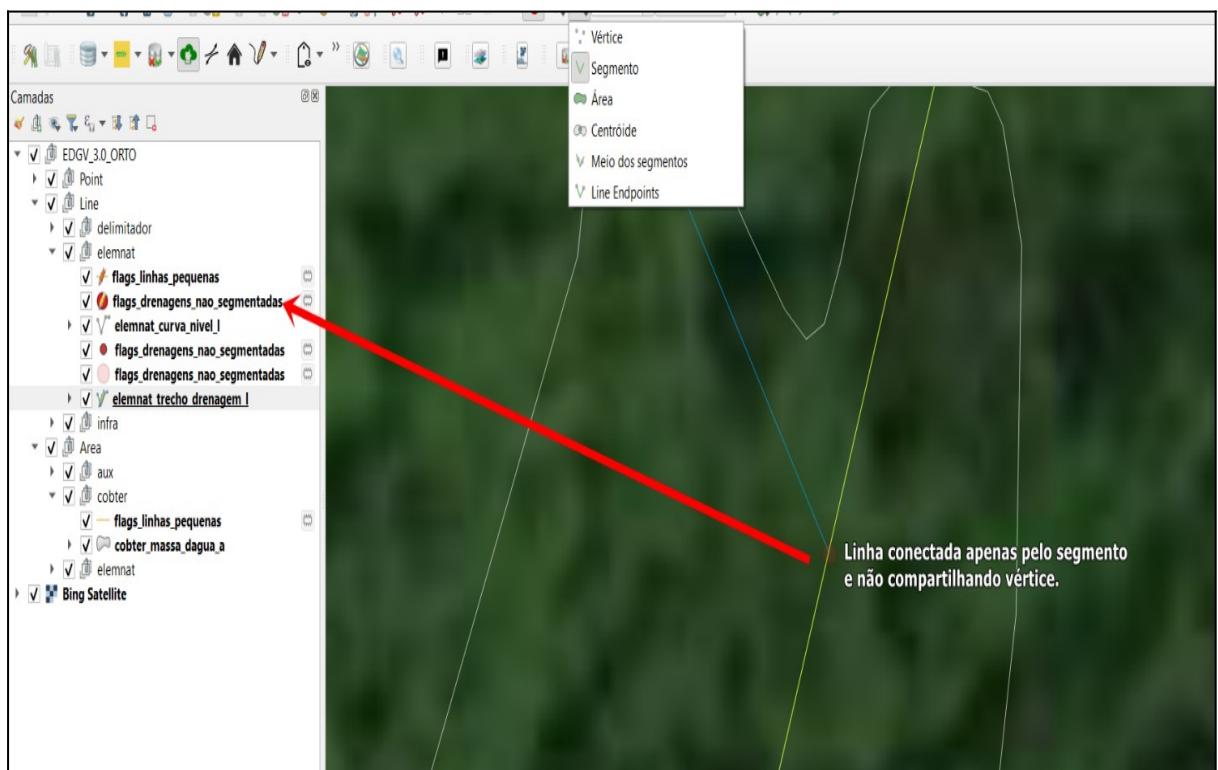


Figura 47: Exemplo de linha não segmentada na intersecção.

## 15. IDENTIFICAR ELEMENTOS PEQUENOS NA REDE

**a) Descrição:** Este algoritmo realiza a identificação de elementos com tamanhos menores que o estabelecido na tolerância em uma camada predeterminada, ao mesmo tempo em que detecta segmentos de drenagem que se encontram desconectados da rede.

**b) Arquivo:** identificar\_elementos\_pequenos\_na\_rede.model3.

**c) Algoritmo:**

```
dsgtools:stringcsvtolayerlistalgorithm;  
dsgtools:batchrunalgorithm;  
dsgtools:identifysmalllines;  
dsgtools:stringcsvtofirstlayerwithelementsalgorithm;  
dsgtools:identifysmallfirstorderdangles;  
native:mergevectorlayers;  
native:extractbylocation.
```

**d) Composição do Modelo:**

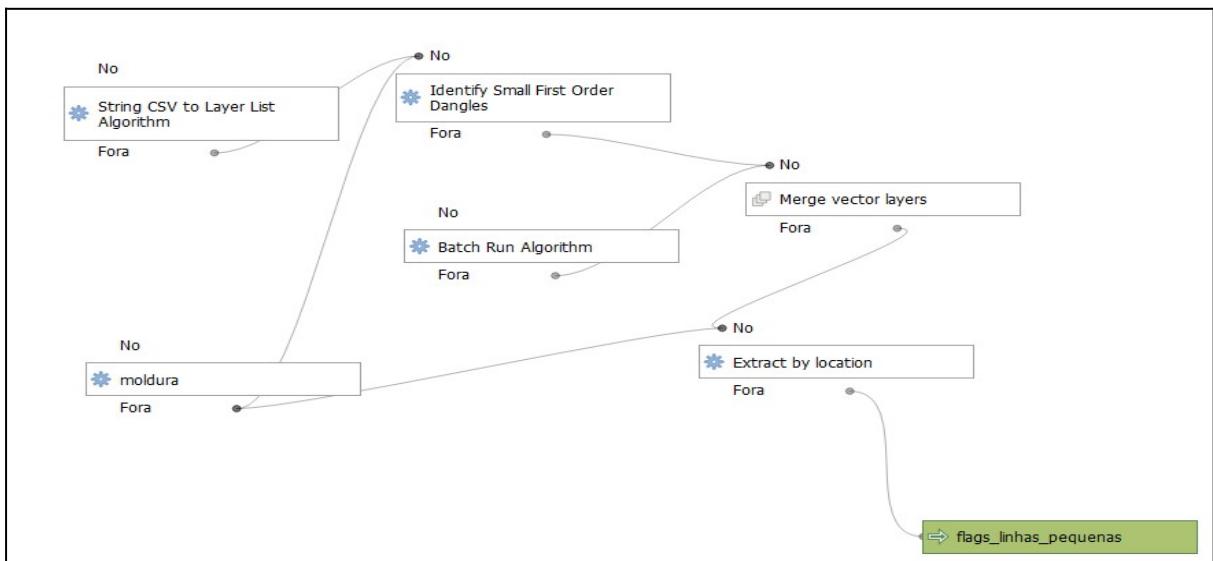


Figura 48: Modelo de identificação de elementos pequenos.

**O modelo é composto por seis etapas:**

*String CSV to Layer List Algorithm:* Converte uma string csv em uma camada especificada usando o algoritmo *stringcsvtolayerlistalgorithm*.

Moldura: Converte uma string csv em uma camada especificada usando o algoritmo *stringcsvtofirstlayerwithelementsalgorithm*.

*Batchrunalgorithm*: Executa o algoritmo *identifysmalllines* com os parâmetros selecionados na seção e para as camadas de linha e separadamente executa para polígonos.

*Identify Small First Order Dangles*: Procura por pequenos segmentos de linha que não se conectam a outros segmentos ou estão muito próximos de uma interseção, executando o algoritmo *identifysmallfirstorderdangles*.

*Merge Vector Layers*: Realiza uma operação de unir as camadas de saída utilizando o algoritmo *mergevectorlayers*.

*ExtractByLocation*: Extrai os elementos pequenos que estão dentro da área delimitada pela moldura usando o algoritmo *extractbylocation*.

#### e) Parâmetros:

Properties	Comments
Description <input type="text" value="Batch Run Algorithm"/>	
Nomes das camadas de entrada separados por vírgula <input type="text" value="123 delimitador_massa_dagua_l,infra_barragem_l,elemnat_elemento_hidrografico_l,elemnat_trecho_drenagem_l"/>	
Nome do algoritmo com seu provedor <input type="text" value="123 dsgtools:identifysmalllines"/>	
Nome da chave de entrada <input type="text" value="123 INPUT"/>	
Dicionário de parâmetros em JSON <pre>{     "SELECTED": false,     "TOLERANCE": 0.00001,     "FLAGS": "TEMPORARY_OUTPUT" }</pre>	
Nome dos parâmetros de saída da camada [opcional] <input type="text" value="123 FLAGS"/>	
Execução de saída em lote <input type="text" value="➡ [Digite o nome se este for um resultado final]"/>	
Dependencies <input type="text" value="0 dependências selecionadas"/> ...	

Figura 49: Extrato dos parâmetros.

## Camadas de entrada:

Tipo de geometria	Camadas de entrada
Linha	delimitador_massa_dagua_1, infra_barragem_1, elemnat_elemento_hidrografico_1 elemnat_trecho_drenagem_1.

## Parâmetros em JSON:

```
{"SELECTED": false,  
"TOLERANCE": 0.00001,  
"FLAGS": "TEMPORARY_OUTPUT"}
```

O parâmetro "**SELECTED**" determina se a validação deve ser realizada apenas nas geometrias selecionadas pelo usuário. Quando marcado como "*true*", a validação é realizada apenas nas geometrias selecionadas. Caso contrário, todas as geometrias são validadas.

O parâmetro "**TOLERANCE**" determina a tolerância em relação à distância e ao tamanho das linhas detectadas pelo algoritmo. Quanto menor o valor, mais sensível o algoritmo será na detecção de linhas pequenas. O valor padrão é 0,00001 graus no sistema geográfico que corresponde a aproximadamente 1 metro.

O parâmetro "**FLAGS**" é utilizado para definir o nome da camada que será criada pela rotina para armazenar as informações de validação. Neste caso, o valor "**TEMPORARY\_OUTPUT**" especifica que uma camada temporária será gerada para armazenar informações sobre as linhas pequenas encontradas durante a validação.

O algoritmo "**Identify Small First Order Dangles**" é uma ferramenta utilizada para identificar trechos de drenagem que não estão conectados ao restante da rede, também conhecidos como "*dangles*" ou pontas soltas. Ele busca por trechos de linha que possuem um único vértice que não é conectado a nenhum outro trecho. Para evitar a identificação de *dangles* não relevantes, o algoritmo se concentra em identificar apenas aqueles que têm um comprimento mínimo especificado.

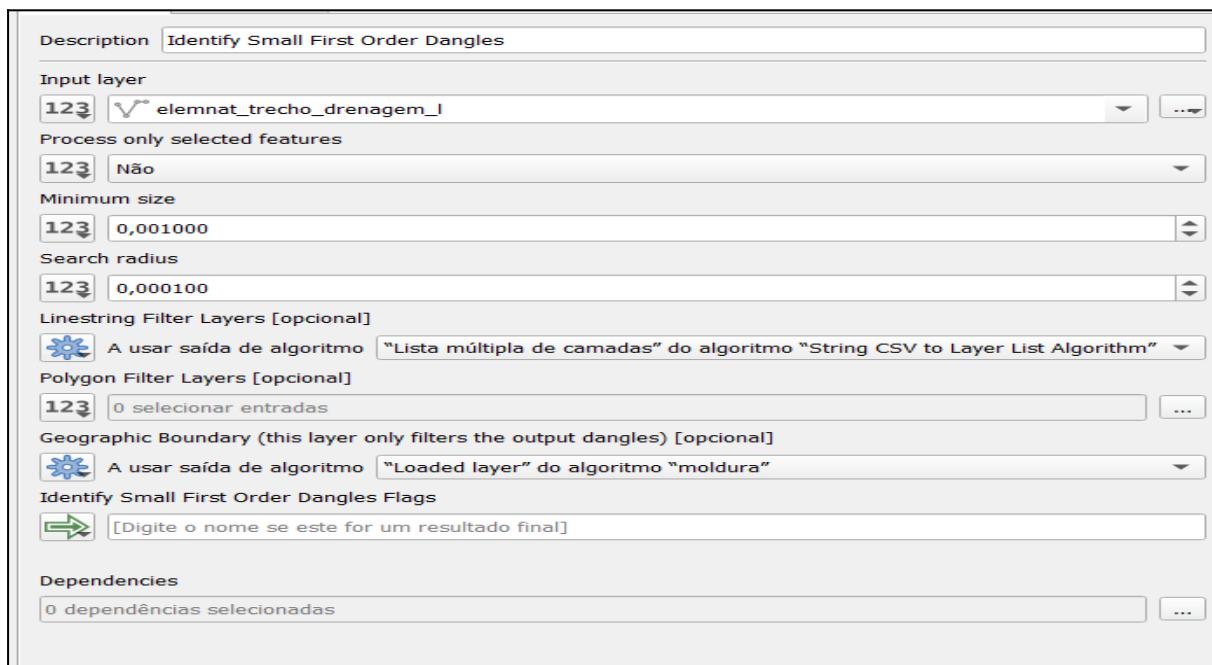


Figura 50: Extrato dos parâmetros.

### Camadas de entrada:

Tipo de geometria	Camadas de entrada
Linha	elemnat_trecho_drenagem_1.
Polígono	aux_moldura_a, moldura, aux_moldura_area_continua_a.

O parâmetro "**SELECTED**" determina se a validação deve ser realizada apenas nas geometrias selecionadas pelo usuário. Quando marcado como "*true*", a validação é realizada apenas nas geometrias selecionadas. Caso contrário, todas as geometrias são validadas.

O parâmetro "**MINIMUM\_SIZE**" se concentra em identificar pontas soltas que têm um comprimento mínimo de 0,001 e estão localizados dentro de um raio de busca definido pelo parâmetro "**SEARCH\_RADIUS**" de 0,0001. Permitindo identificar possíveis problemas na rede, como drenagens muito próximas da confluência, porém sem conexão.

O parâmetro "**FLAGS**" é utilizado para definir o nome da camada que será criada pela rotina para armazenar as informações de validação. Neste caso, o valor

**"TEMPORARY\_OUTPUT"** especifica que uma camada temporária será gerada para armazenar informações sobre os elementos pequenos e possíveis problemas encontrados na rede.

Por fim é utilizado o algoritmo de extrair pela localização para filtrar os elementos que estão dentro da moldura.

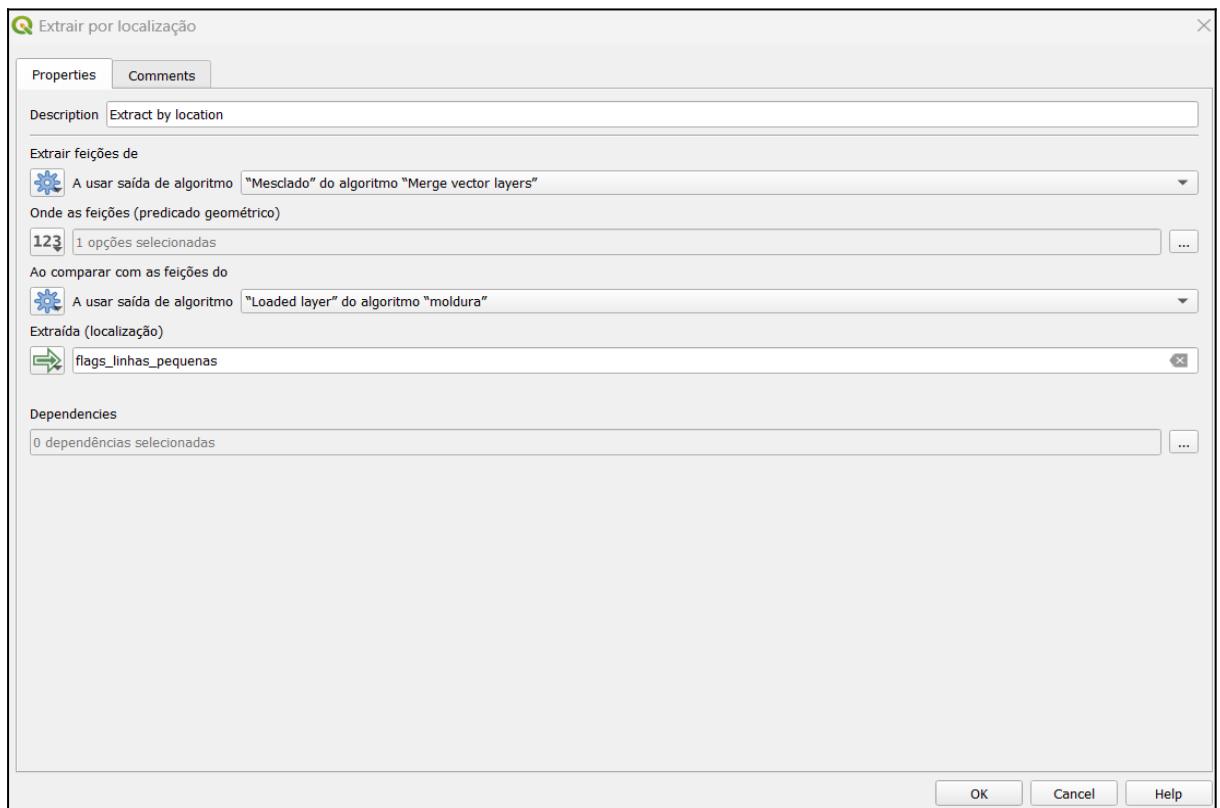


Figura 51: Extrair por localização.

**Extrair feições de:** Utiliza os dados de saída do procedimento anterior.

**Predicado geométrico:** Uma lista que especifica o tipo de relação espacial entre as geometrias que devem ser consideradas na extração. Neste caso, o valor [0] significa que apenas as geometrias que se intersectam serão extraídas.

**f) nome da camada de *flags*:** flags\_linhas\_pequenas.

### g) Resultado do processo e exemplo de erros:

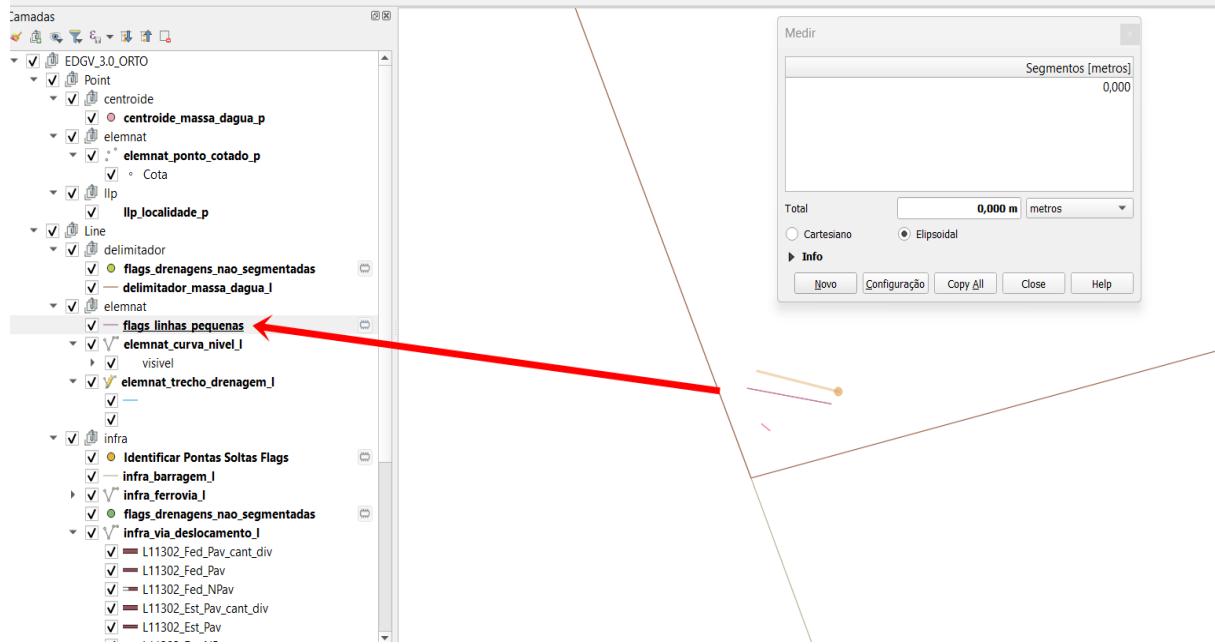


Figura 52: Exemplo de linha pequena.

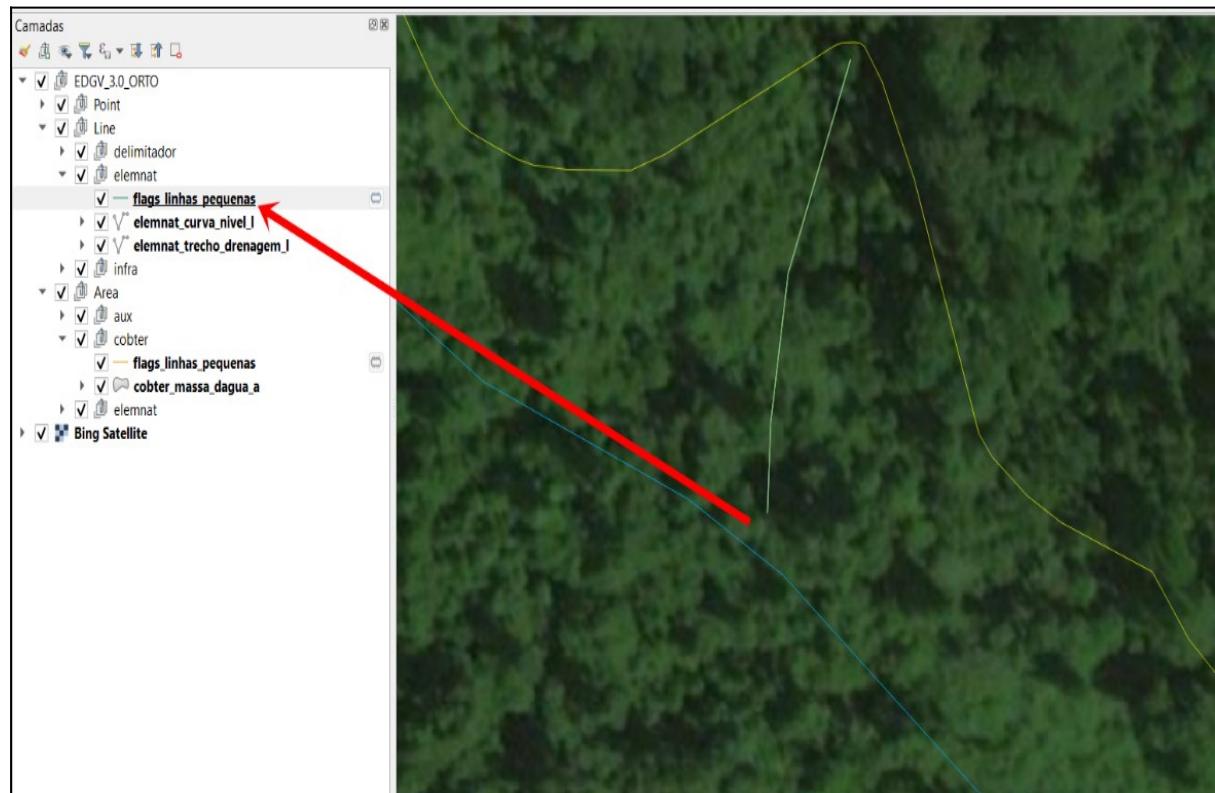


Figura 53: Exemplo de segmento desconectado da rede.

## 16. IDENTIFICAR ERROS NA CONSTRUÇÃO DA REDE DE DRENAGEM

**a) Descrição:** Tem como objetivo detectar e identificar erros relacionados à construção da rede de drenagem através da detecção de ângulos agudos, pontas soltas e linhas não segmentadas desconectadas da rede.

**b) Arquivo:** identificar\_erros\_rede\_drenagem.model3.

**c) Algoritmos:**

```
dsgtools:stringcsvtolayerlistalgorithm;  
dsgtools:stringcsvtofirstlayerwithelementsalgorithm;  
dsgtools:identifyunmergedlineswithsameatributeset;  
dsgtools:identifynetworkconstructionissues;  
native:mergevectorlayers;  
native:extractbylocation.
```

**d) Composição do Modelo:**

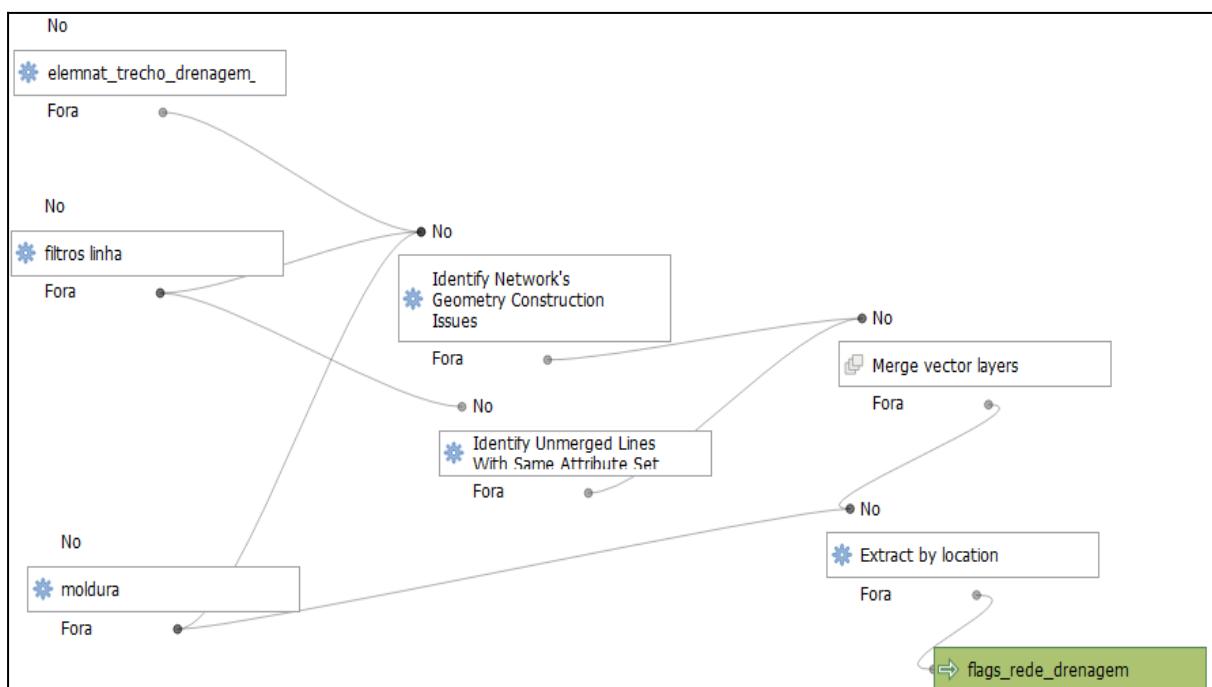


Figura 54: Modelo de identificação de erros na construção das redes de drenagem.

## **O modelo é composto por cinco etapas:**

*String CSV to Layer List Algorithm:* Converte uma *string csv* em uma camada especificada usando o algoritmo *stringcsvtolayerlistalgorithm*.

*Identify Network's Geometry Construction Issues:* Tem por finalidade a identificação de problemas na construção da geometria da rede, em especial na detecção de potenciais erros na rede de drenagem. Seu objetivo consiste em encontrar, de maneira automatizada, ângulos agudos, pontas soltas, linhas não segmentadas desconectadas da rede utilizando o algoritmo *identifynetworkconstructionissues*.

*Identify Unmerged Lines With Same Attribute Set:* Responsável por identificar linhas não mescladas (*unmerged*) com o mesmo conjunto de atributos. Ele compara os atributos de todas as linhas dentro de uma camada especificada e verifica se duas ou mais linhas estão com o mesmo conjunto de atributos utilizando o algoritmo *identifyunmergedlineswithsameattributeset*.

*Merge Vector Layers:* Realiza uma operação de unir as camadas de saída utilizando o algoritmo *mergevectorlayers*.

*ExtractByLocation:* Extrai os erros que estão dentro da área delimitada pela moldura usando o algoritmo *extractbylocation*.

### e) Parâmetros:

Properties		Comments
Description	Identify Network's Geometry Construction Issues	
Input lines	<input checked="" type="checkbox"/> A usar saída de algoritmo "Lista múltipla de camadas" do algoritmo "elemnat_trecho_drenagem_l"	
Process only selected features	<input checked="" type="checkbox"/> Não	
Input is a boundary layer (every line must be connected to an element of either the input layer or the filters)	<input checked="" type="checkbox"/> Não	
Search radius	<input checked="" type="checkbox"/> 0,000010	
Linestring Filter Layers [opcional]	<input checked="" type="checkbox"/> A usar saída de algoritmo "Lista múltipla de camadas" do algoritmo "filtros linha"	
Polygon Filter Layers [opcional]	<input checked="" type="checkbox"/> 0 selecionar entradas	
Ignore dangle on unsegmented lines	<input checked="" type="checkbox"/> Não	
Geographic Boundary (this layer only filters the output dangles) [opcional]	<input checked="" type="checkbox"/> A usar saída de algoritmo "Loaded layer" do algoritmo "moldura"	
Identify Network's Geometry Construction Issues Flags	<input checked="" type="checkbox"/> [Digite o nome se este for um resultado final]	
Dependencies	<input checked="" type="checkbox"/> 0 dependências selecionadas	

Figura 55: Extrato dos parâmetros.

### Camadas de entrada:

Tipo de geometria	Camadas de entrada
Linha	elemnat_trecho_drenagem_l, delimitador_massa_dagua_l, infra_barragem_l.
Polígono	aux_moldura_a, moldura, aux_moldura_area_continua_a.

O parâmetro "**SELECTED**" determina se a validação deve ser realizada apenas nas geometrias selecionadas pelo usuário. Quando marcado como "*true*", a verificação é realizada apenas nas geometrias selecionadas. Caso contrário, todas as geometrias são verificadas.

No parâmetro "**INPUT IS A BOUNDARY LAYER**" se a opção "*false*" estiver definida no algoritmo, isso significa que a moldura não precisa estar conectada a elementos da camada de entrada ou dos filtros.

O Parâmetro "**SEARCH RADIUS**" do algoritmo determina a distância máxima que será percorrida para encontrar pontas soltas em relação aos segmentos de linha e polígonos que se cruzam nas intersecções. No modelo, o valor padrão do raio de busca é definido como 0,00001 graus no sistema de coordenadas geográfico que corresponde a um valor aproximado de 1 metro. No entanto, é importante lembrar que esse valor pode ser ajustado de acordo com as necessidades específicas, considerando a escala e precisão dos dados utilizados no processamento.

No parâmetro "**FILTER LAYER**" as camadas de filtro presentes são utilizadas com o propósito de especificar as camadas que serão empregadas para filtrar a identificação de linhas que apresentem problemas de construção na rede de drenagem. As camadas filtradas são, então, empregadas para a identificação de problemas geométricos na construção da rede.

O parâmetro "**GEOGRAPHIC BOUNDARY**" se refere à delimitação geográfica para a execução do processamento em uma região determinada. A "moldura" representa a área delimitada em questão, isso evita que sejam incluídas informações irrelevantes ou imprecisas de outras áreas que estão fora da delimitação.

O parâmetro "**FLAGS**" é utilizado para definir o nome da camada que será criada pela rotina para armazenar as informações de validação. Neste caso, o valor "**TEMPORARY\_OUTPUT**" especifica que uma camada temporária será gerada para armazenar informações sobre os problemas de construção na rede de drenagem.

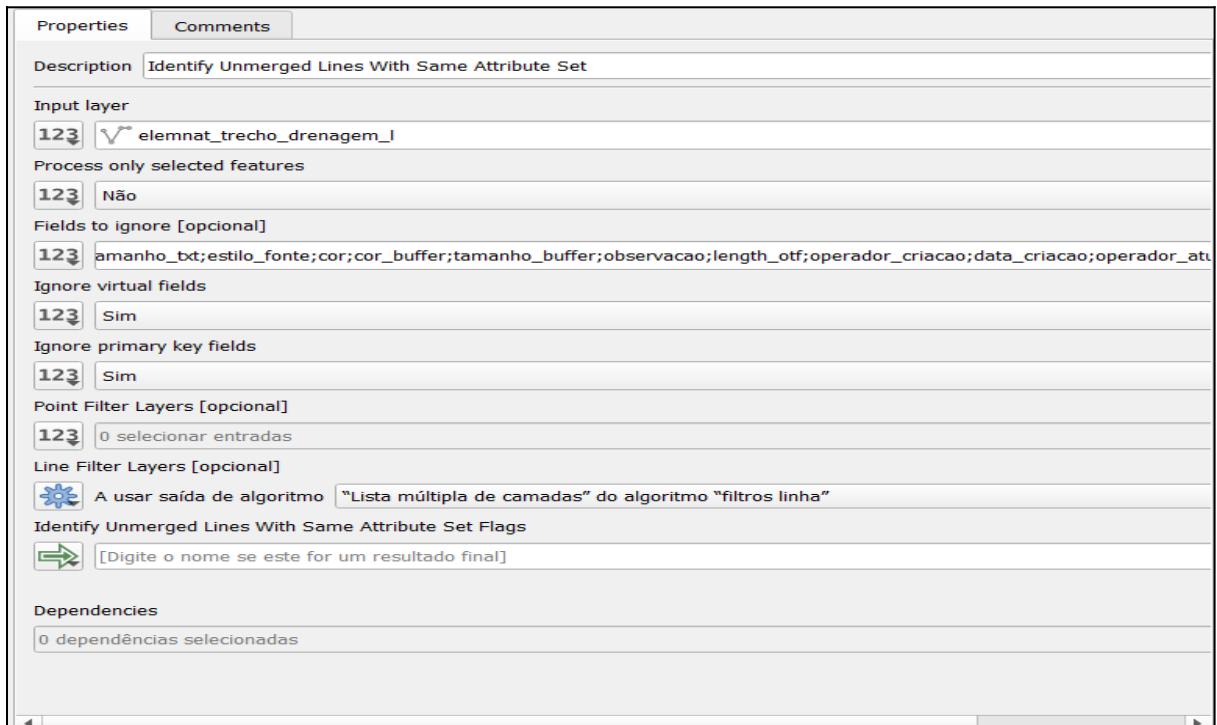


Figura 56: Extrato dos parâmetros.

### Camadas de entrada:

Tipo de geometria	Camadas de entrada
Linha	elemnat_trecho_drenagem_l.
Polígono	aux_moldura_a, moldura, aux_moldura_area_continua_a.

O parâmetro "**SELECTED**" determina se a validação deve ser realizada apenas nas geometrias selecionadas pelo usuário. Quando marcado como "*true*", a validação é realizada apenas nas geometrias selecionadas. Caso contrário, todas as geometrias são validadas.

No parâmetro "**FIELDS TO IGNORE**" é definido uma lista de campos a ignorar de atributos (*ATTRIBUTE\_BLACKLIST*) que não serão considerados na comparação das feições, a fim de evitar falsos positivos.

O parâmetro "**IGNORE\_PK\_FIELDS**" é um booleano que indica se os campos da chave primária das camadas de entrada devem ser ignorados na comparação. (apesar de estar

marcado como "true", o campo de chave primária "id" já está sendo ignorado na lista negra de atributos).

O parâmetro "**IGNORE\_VIRTUAL\_FIELDS**" é um booleano que indica se os campos virtuais (calculados dinamicamente) devem ser ignorados na comparação.

No parâmetro "**FILTER LAYER**" as camadas de filtro presentes são usadas para especificar camadas que serão usadas para identificar linhas que não foram unidas corretamente.

O parâmetro "**FLAGS**" é utilizado para definir o nome da camada que será criada pela rotina para armazenar as informações que necessitam de correção. Neste caso, o valor "**TEMPORARY\_OUTPUT**" especifica que uma camada temporária será gerada para armazenar informações sobre as linhas que não foram unidas corretamente.

Por fim é utilizado o algoritmo de extrair pela localização para filtrar os elementos que estão dentro da moldura.

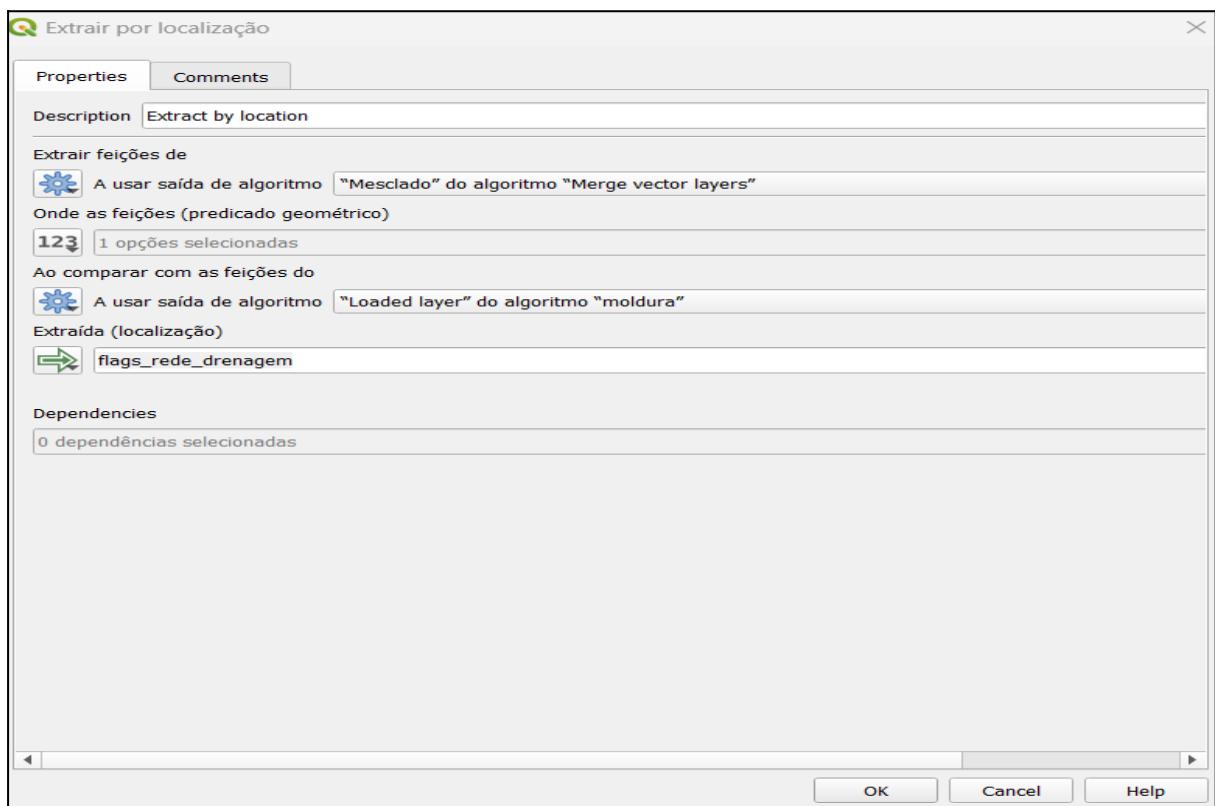


Figura 57: Extrair por localização.

**Extrair feições de:** Utiliza os dados de saída do procedimento anterior.

**Predicado geométrico:** Uma lista que especifica o tipo de relação espacial entre as geometrias que devem ser consideradas na extração. Neste caso, o valor [0] significa que apenas as geometrias que se intersectam serão extraídas.

**f) nome da camada de *flags*: *flags\_rede\_drenagem*.**

**g) Resultado do processo e exemplo de erros:**

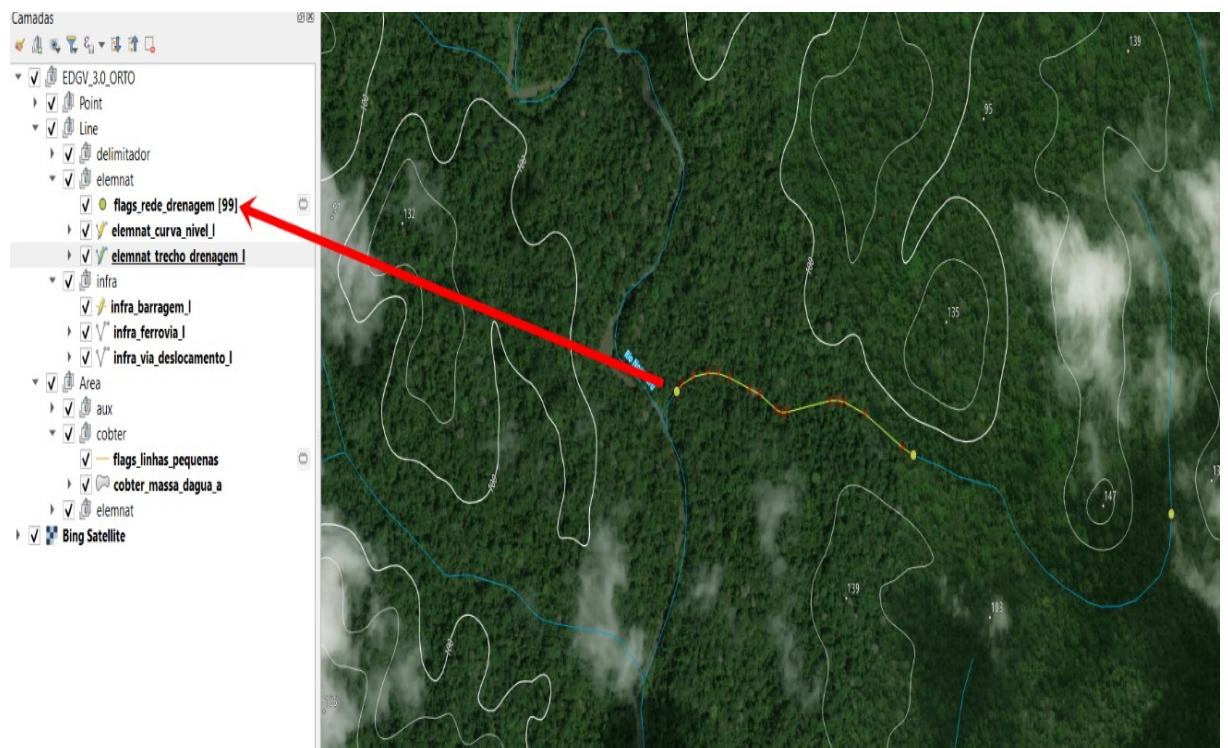


Figura 58: Exemplo de linhas não unidas.



Figura 59: Exemplo de erros na construção das redes de drenagem.

## 17. IDENTIFICAR ERROS NA CONSTRUÇÃO DAS CURVAS DE NÍVEL

**a) Descrição:** Este modelo tem por objetivo identificar falhas na construção de curvas de nível, as quais são caracterizadas por apresentarem equidistâncias específicas entre seus pontos. Para tanto, torna-se necessária a adaptação do modelo a fim de que seja inserido o arquivo de acordo com a escala utilizada pelo usuário. Tendo em vista que somente a equidistância varia entre diferentes escalas, então para fins didáticos será apresentado um exemplo de modelo para a escala de 1:50.000.

### b) Arquivos:

identificar\_erros\_na\_construcao\_das\_curvas\_de\_nivel\_25k.model3;  
 identificar\_erros\_na\_construcao\_das\_curvas\_de\_nivel\_50k.model3;  
 identificar\_erros\_na\_construcao\_das\_curvas\_de\_nivel\_100k.model3;  
 identificar\_erros\_na\_construcao\_das\_curvas\_de\_nivel\_250k.model3.

### c) Algoritmos:

*dsgtools:stringcsvtofirstlayerwithelementsalgorithm;*  
*dsgtools:identifyterrainmodelerrorsalgorithm;*

`dsgtools:verifycountourstacking.`

#### d) Composição do Modelo:

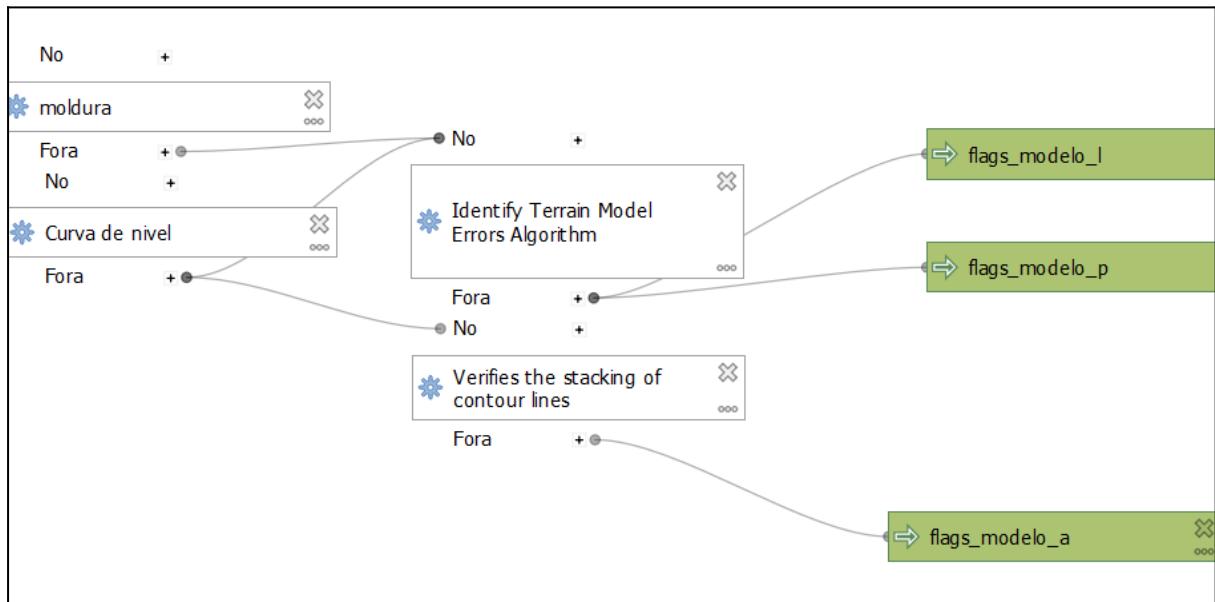


Figura 60: Modelo de identificação de erros na construção das curvas de nível.

#### O modelo é composto por três etapas:

*String CSV to Layer List Algorithm*: Converte uma string csv em uma camada especificada usando o algoritmo `stringcsvtofirstlayerwithelementsalgorithm`.

*Identify Terrain Model Errors*: Tem como função identificar erros na modelagem do terreno, como, por exemplo, erros na construção de curvas de nível. Ele utiliza a camada específica de curvas de nível como entrada e verifica se as equidistâncias entre as curvas estão corretas de acordo com os valores definidos para cada escala utilizando o algoritmo `identifyterrainmodelerrorsalgorithm`.

*Verifies the stacking of contour lines*: tem como objetivo verificar se as curvas de nível estão empilhadas corretamente, ou seja, se não há sobreposição utilizando o algoritmo `verifycountourstacking`.

### e) Parâmetros:

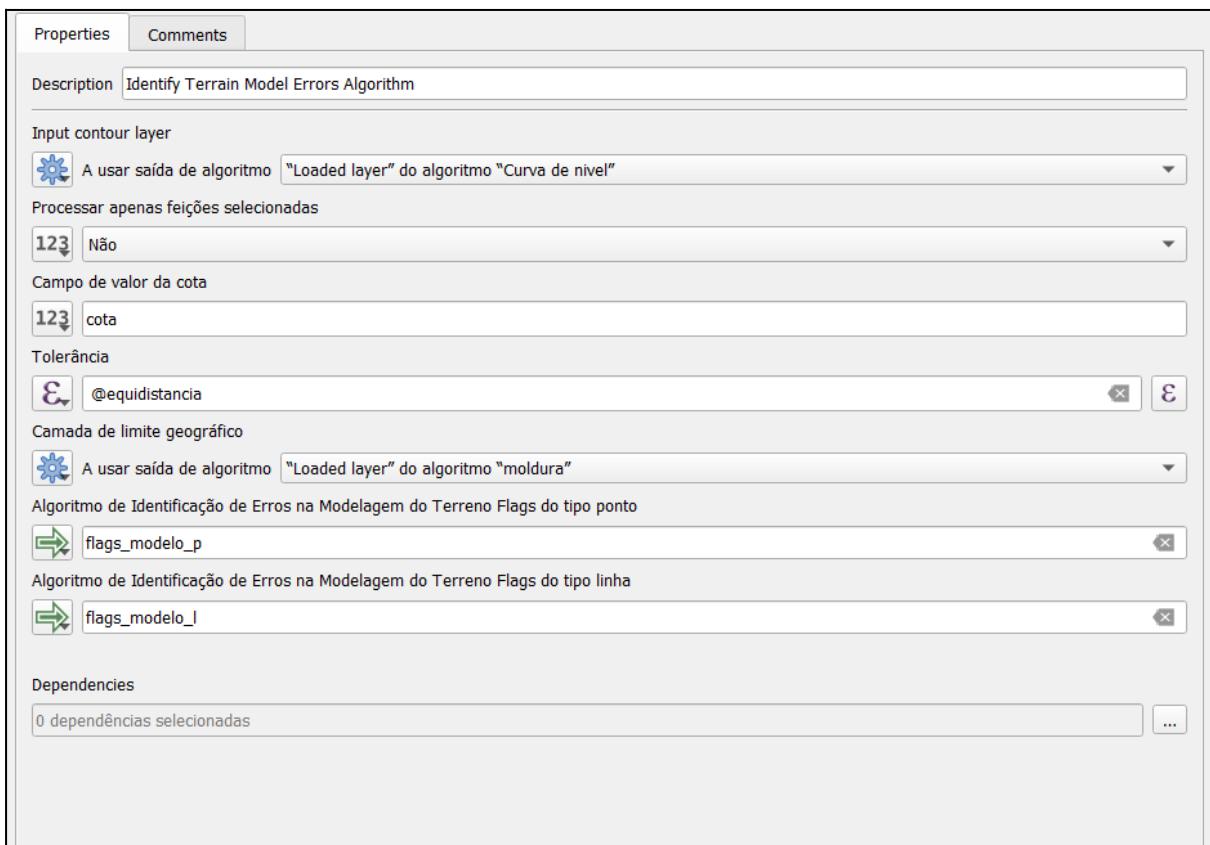


Figura 61: Extrato dos parâmetros.

### Camadas de entrada:

Tipo de geometria	Camadas de entrada
Linha	elemnat_curva_nivel_1.
Polígono	aux_moldura_a, moldura, aux_moldura_area_continua_a.

O parâmetro "**SELECTED**" determina se a validação deve ser realizada apenas nas geometrias selecionadas pelo usuário. Quando marcado como "*true*", a verificação é realizada apenas nas geometrias selecionadas. Caso contrário, todas as geometrias são verificadas.

O parâmetro "**CAMPO DE VALOR DA COTA**" requer que seja fornecida a coluna contendo os valores de cota. Tal entrada é utilizada para detecção de erros na construção das curvas de nível, por meio da comparação dos valores de cota.

No Parâmetro "**TOLERÂNCIA**" o algoritmo utiliza a variável `@equidistancia`, que representa a distância entre as curvas de nível no modelo e tem um valor predefinido de 20 metros para a escala 1:50000. Esse valor pode variar de acordo com o arquivo do modelo utilizado. Se a diferença entre as curvas de nível adjacentes não estiver de acordo com a tolerância, determinada por `@equidistancia`, o algoritmo identifica esses pontos como erros e gera "*flags*" para indicá-los.

O parâmetro "**CAMADA DE LIMITE GEOGRÁFICO**" se refere à delimitação geográfica para a execução do processamento em uma região determinada. A "moldura" representa a área delimitada em questão, isso evita que sejam incluídas informações irrelevantes ou imprecisas de outras áreas que estão fora da delimitação.

O parâmetro "**FLAGS**" é utilizado para definir o nome da camada que será criada pela rotina para armazenar as informações dos lugares onde necessitam de correção.

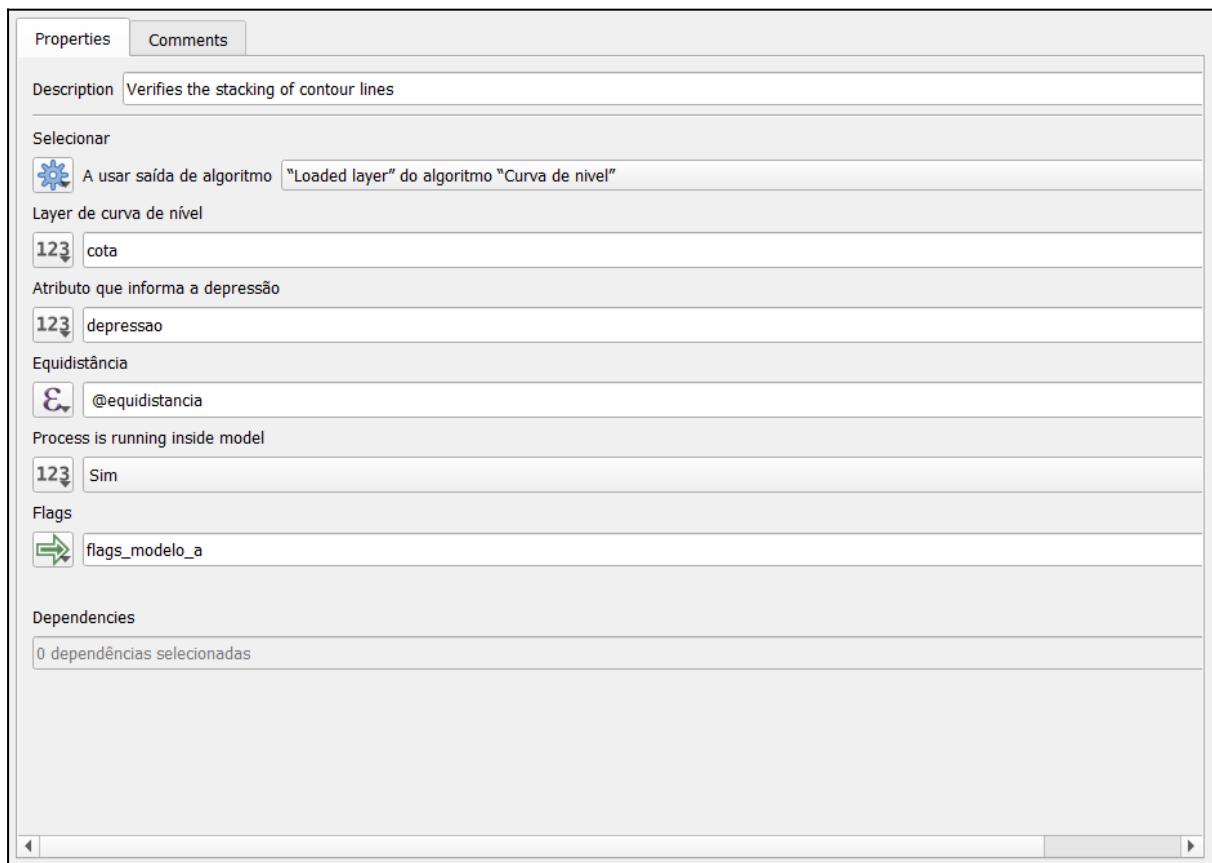


Figura 62: Extrato dos parâmetros.

### **Camadas de entrada:**

Tipo de geometria	Camadas de entrada
Linha	elemnat_curva_nivel_l.

O parâmetro "**LAYER DE CURVA DE NIVEL**" requer que seja fornecida a coluna contendo os valores de cota. Tal entrada é utilizada para detecção de erros na construção das curvas de nível, por meio da comparação dos valores de cota.

O Parâmetro "**ATRIBUTO QUE INFORMA A DEPRESSÃO**" requer que seja fornecida a coluna contendo os valores de depressão.

No Parâmetro "**EQUIDISTÂNCIA**" o algoritmo utiliza a variável @equidistancia, que representa a distância entre as curvas de nível no modelo e tem um valor predefinido de 20 metros para a escala 1:50000. Esse valor pode variar de acordo com o arquivo do modelo utilizado.

O parâmetro "**CAMADA DE LIMITE GEOGRÁFICO**" se refere à delimitação geográfica para a execução do processamento em uma região determinada. A "moldura" representa a área delimitada em questão, isso evita que sejam incluídas informações irrelevantes ou imprecisas de outras áreas que estão fora da delimitação.

O parâmetro "**RUNNING INSIDE MODEL**" quando definido como "true", o algoritmo ajusta seu comportamento para lidar com os parâmetros de entrada que foram fornecidos por outros algoritmos dentro do mesmo modelo de processamento.

O parâmetro "**FLAGS**" é utilizado para definir o nome da camada que será criada pela rotina para armazenar as informações dos lugares onde necessitam de correção.

#### **f) nome da camada de flags:**

*flag\_modelo\_p;*  
*flag\_modelo\_l;*  
*flag\_modelo\_a.*

### g) Resultado do processo e exemplo de erros:

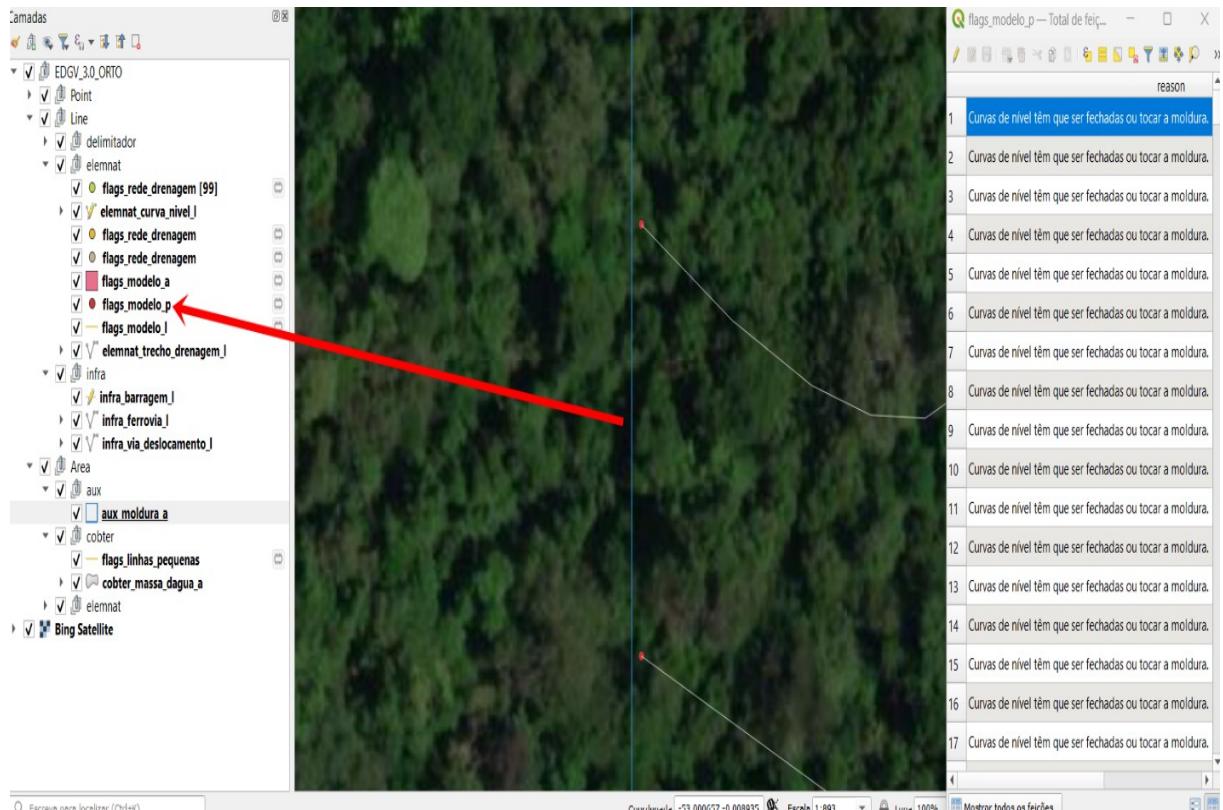


Figura 63: Exemplo de curvas de nível não conectadas a moldura.

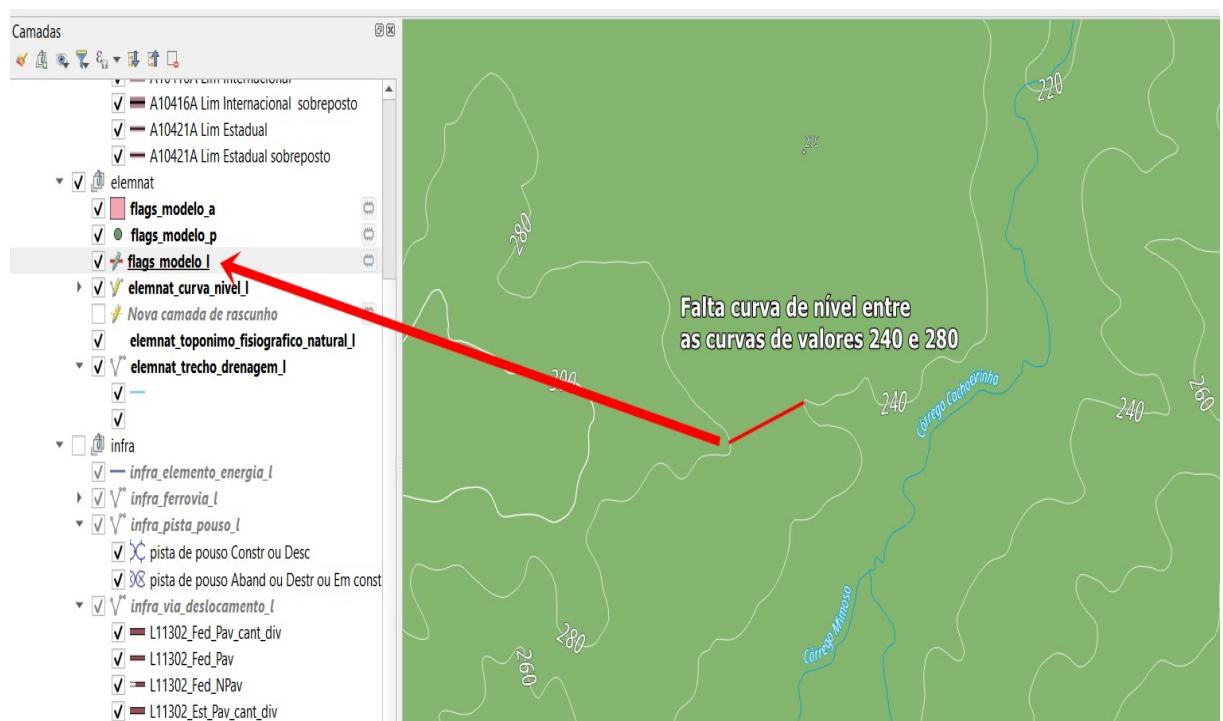


Figura 64: Exemplo de curva de nível faltante.

## 18. FECHAR POLÍGONOS DE MASSA D'ÁGUA

**a) Descrição:** Este modelo tem como objetivo a geração de polígonos de massa d'água por meio da manipulação vetorial, utilizando a camada delimitadora de linha como entrada. Destaca-se que esta rotina é a única responsável pela manipulação dos vetores presentes no fluxo.

**b) Arquivos:** fechar\_poligonos\_massa\_dagua.model3.

**c) Algoritmos:**

```
dsgtools:stringcsvtofirstlayerwithelementsalgorithm;  
dsgtools:stringcsvtolayerlistalgorithm;  
dsgtools:buildpolygonsfromcenterpointsandboundariesalgorithm.
```

**d) Composição do Modelo:**

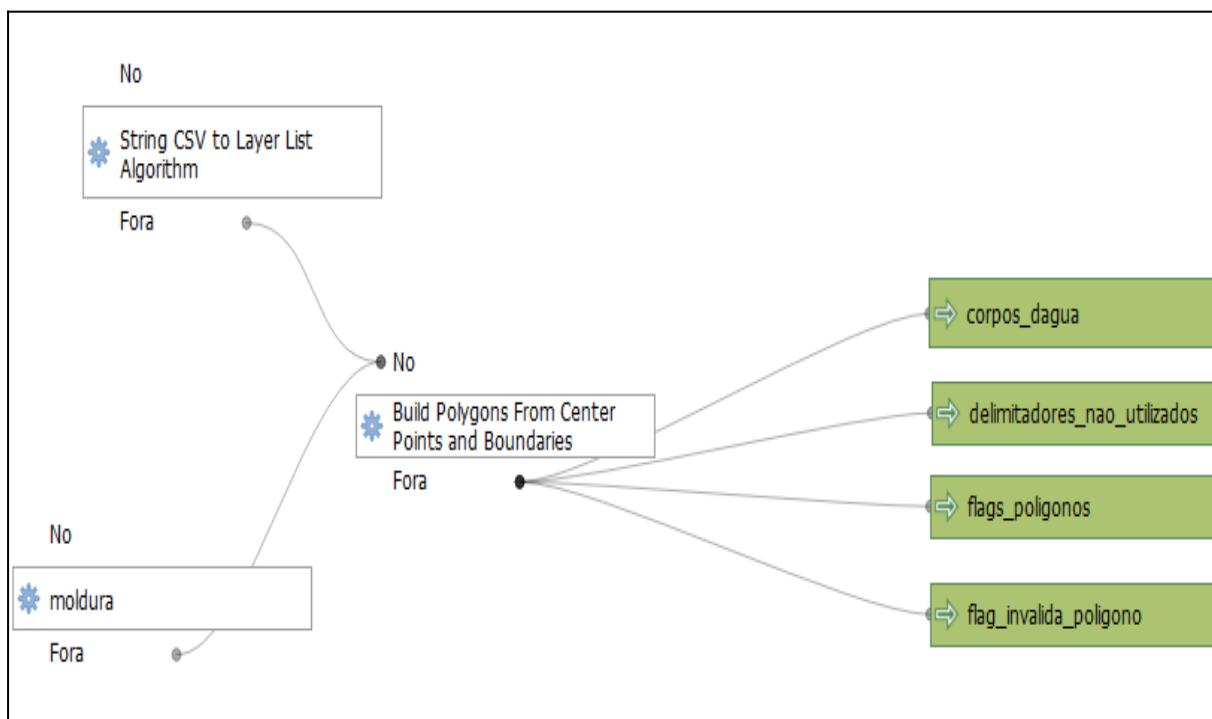


Figura 65: Modelo de geração de áreas de corpo de água.

**O modelo é composto por três etapas:**

*String CSV to Layer List Algorithm:* Converte uma *string* csv em uma camada especificada usando o algoritmo *stringcsvtolayerlistalgorithm*.

Moldura: Converte uma *string* csv em uma camada especificada usando o algoritmo *stringcsvtofirstlayerwithelementsalgorithm*.

Construir polígonos a partir dos centroides: O algoritmo em questão tem como objetivo criar polígonos que representem as massas d'água a partir de linhas de restrição (delimitadores), as quais delimitam os limites do polígono a ser construído. Para tanto, utiliza-se um ponto central que contém as informações principais acerca da geometria a ser gerada. Ademais, o polígono recém-construído herda os atributos do ponto de centroide, este procedimento é realizado utilizando o algoritmo *buildpolygonsfromcenterpointsandboundariesalgorithm*.

### e) Parâmetros:

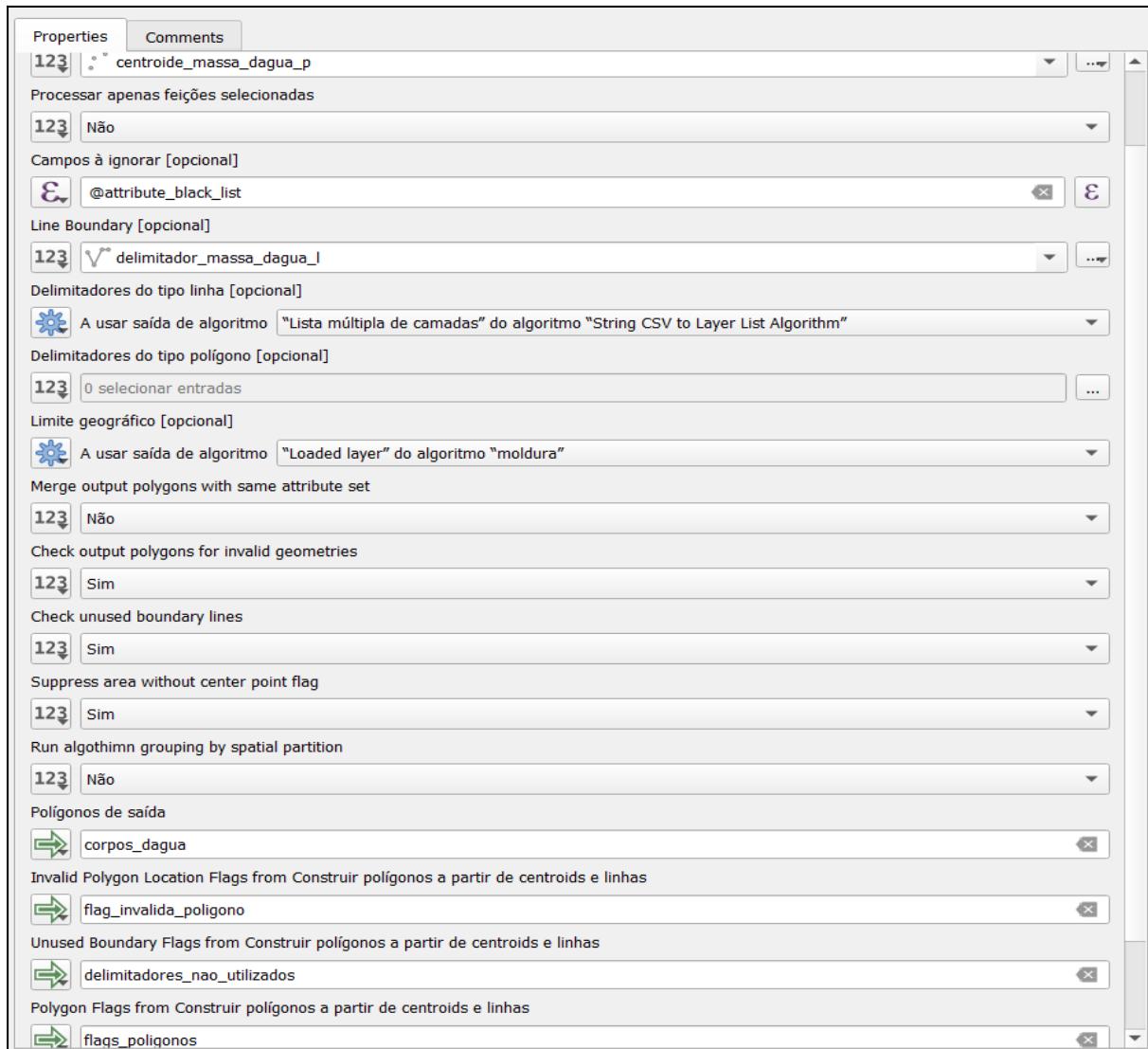


Figura 66: Extrato dos parâmetros.

### Camadas de entrada:

Tipo de geometria	Camadas de entrada
Ponto	centroide_massa_dagua_p.
Linha	delimitador_massa_dagua_l.

Polígono	aux_moldura_a, moldura, aux_moldura_area_continua_a.
----------	--

O parâmetro "**SELECTED**" determina se a geração dos polígonos deve ser realizada apenas nas geometrias selecionadas pelo usuário. Quando marcado como "*true*", a execução é realizada apenas nas geometrias selecionadas. Caso contrário, todas as geometrias são processadas.

No parâmetro "**CAMPOS A IGNORAR**" é definido uma lista negra de atributos (*attribute blacklist*) que não serão considerados na hora de herdar os atributos dos pontos de centroide. O valor do parâmetro *@attribute\_black\_list* refere-se a uma variável presente nas variáveis do modelo que contém uma lista dos campos a serem ignorados, separados por vírgula.

O Parâmetro "**LINE BOUNDARY**" é usado para definir a camada de linha que será usada como limite (delimitadores) para criar polígonos a partir dos centroides.

Nos parâmetros "**DELIMITADORES DO TIPO LINHA E POLIGONO**" é usado para definir as demais camadas de linhas ou de polígonos que será usada como limite (delimitadores) para criar os polígonos a partir dos centroides.

O parâmetro "**CAMADA DE LIMITE GEOGRÁFICO**" se refere à delimitação geográfica para a execução do processamento em uma região determinada. A "moldura" representa a área delimitada em questão, isso evita que sejam incluídas informações irrelevantes ou imprecisas de outras áreas que estão fora da delimitação.

O parâmetro "**MERGE OUTPUT POLYGONS**" com o valor "*false*" no algoritmo indica que os polígonos resultantes não serão mesclados em um único polígono, mas sim mantidos separados.

O parâmetro "**CHECK OUTPUT POLYGONS**" é usado para verificar se os polígonos produzidos são válidos ou não. Quando definido "*true*", o algoritmo verifica se há polígonos inválidos na saída e se houver retorna um conjunto de geometrias inválidas na "*flag*" de parâmetro: "*INVALID\_Polygon\_LOCATION*".

O parâmetro "**SUPPRESS AREA WITHOUT CENTER POINT FLAG**" com o valor "*true*", indica que áreas sem centroide devem ser suprimidas na saída, ou seja, somente as

áreas que possuem um ponto central definido serão consideradas na construção dos polígonos. Esse parâmetro é utilizado para evitar a criação de polígonos em áreas sem informação.

O parâmetro "**FLAGS**" é composto por dois tipos distintos de informações: o primeiro refere-se às localizações que necessitam de correção, enquanto o segundo se relaciona às camadas de massa d'água recém-criadas que serão posteriormente copiadas.

#### f) nome da camada de *flags*:

*flags* de correção: *flag\_poligonos*, *flag\_invalida\_poligono*, *delimitadores\_nao\_utilizados*.

*flags* das camadas criadas: *corpos\_dagua*.

#### g) Resultado do processo e exemplo de erros:



Figura 67: Exemplo de área gerada.

## 19. MODELOS EXECUTADOS FORA DO WORKFLOW

Existem modelos que requerem execução paralela às etapas presentes no fluxo de trabalho. As etapas iniciais têm como objetivo compilar as rotinas anteriores executadas, sendo elas as rotinas de validade geométrica e rotinas de validade de vértice, podendo servir como uma forma de revisão da validação ou somente para garantir a ausência de erros nas próximas fases sem a necessidade de executar todo o fluxo novamente. Além disso, é

importante considerar que, após a execução do fluxo de trabalho, a rotina de geração de áreas utilizando delimitadores e centroides requer correções constantes. Portanto, é necessário executar continuamente as rotinas de pontas soltas nos delimitadores e, posteriormente, o processo de manipulação para gerar as áreas. Por essa razão, a separação dessas rotinas tornou-se necessária devido à sua frequente execução, e compilá-las em um único fluxo de trabalho exigiria a execução de todas as rotinas presentes, atrasando o processo de validação. Nesse sentido, as rotinas de geração de massa d'água, ilhas e elementos hidrográficos, assim como suas respectivas rotinas de identificação de pontas soltas, estão separadas do fluxo de trabalho.

## 19.1. ROTINAS DE VALIDADE GEOMÉTRICA

**a) Descrição:** Tem como objetivo realizar a validação geométrica de feições. Este algoritmo consiste em uma composição de rotinas: identificação de geometrias inválidas, identificação de geometrias com mais de uma parte e identificação de feições duplicadas.

**b) Arquivos:** rotinas\_validade\_geometrica\_alt\_hid.model3.

**c) Algoritmos:**

*model:01*-Identifica geometrias inválidas;

*model:02*-Identifica geometrias com mais de uma parte;

*model:03*-Identifica feições duplicadas;

*native:mergevectorlayers*.

#### d) Composição do Modelo:

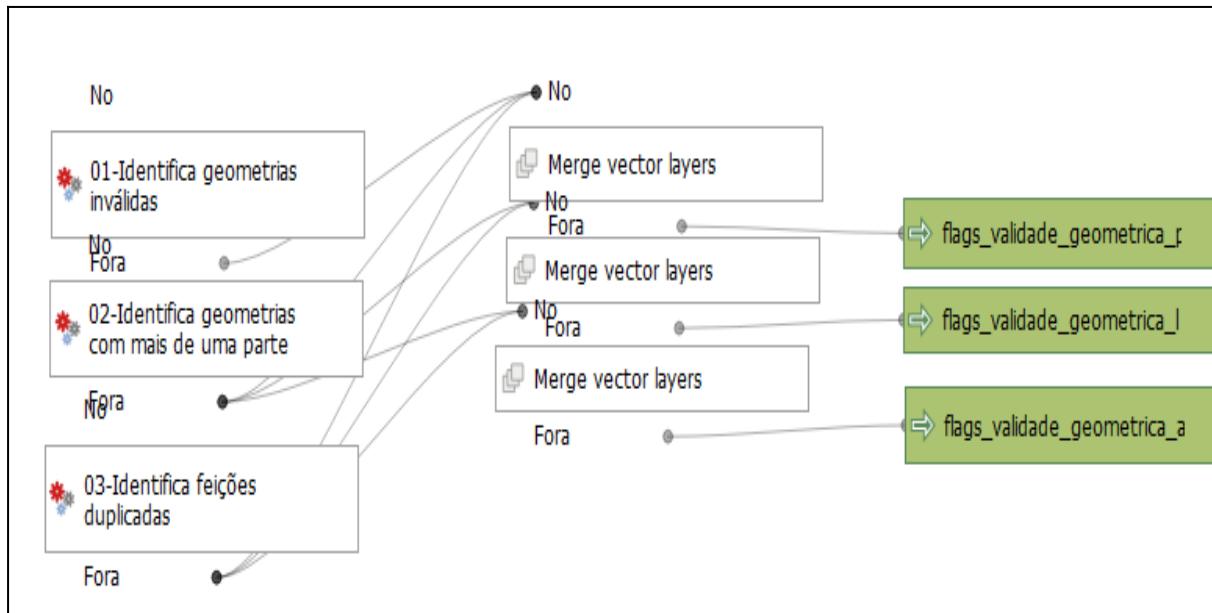


Figura 68: Modelo composto por modelos de identificação da validade geométrica.

#### O modelo é composto por quatro etapas:

A primeira etapa do algoritmo é a "01-Identifica geometrias inválidas", que visa identificar geometrias que não estão de acordo com as regras geométricas estabelecidas.

A segunda etapa do algoritmo é a "02-Identifica geometrias com mais de uma parte", que tem como objetivo identificar geometrias compostas por múltiplas partes, o que pode indicar erros na estruturação das feições.

A terceira etapa é a "03-Identifica feições duplicadas", que tem a finalidade de identificar feições que apresentam duplicidade, ou seja, feições que possuem a mesma representação espacial.

A quarta etapa o modelo utiliza o algoritmo nativo para mesclar as camadas de flags de saídas de forma que gere somente uma para cada tipo de geometria.

### e) Parâmetros:

Os parâmetros utilizados nas rotinas estão inseridos dentro de cada modelo, uma vez que este conjunto é composto por múltiplos modelos. As informações referentes a esses parâmetros são detalhadas nos itens 1, 2 e 3 deste apêndice.

### f) nome da camada de *flags*:

flags\_validade\_geometrica\_a,  
flags\_validade\_geometrica\_l,  
flags\_validade\_geometrica\_p.

### g) Resultado do processo e exemplo de erros:

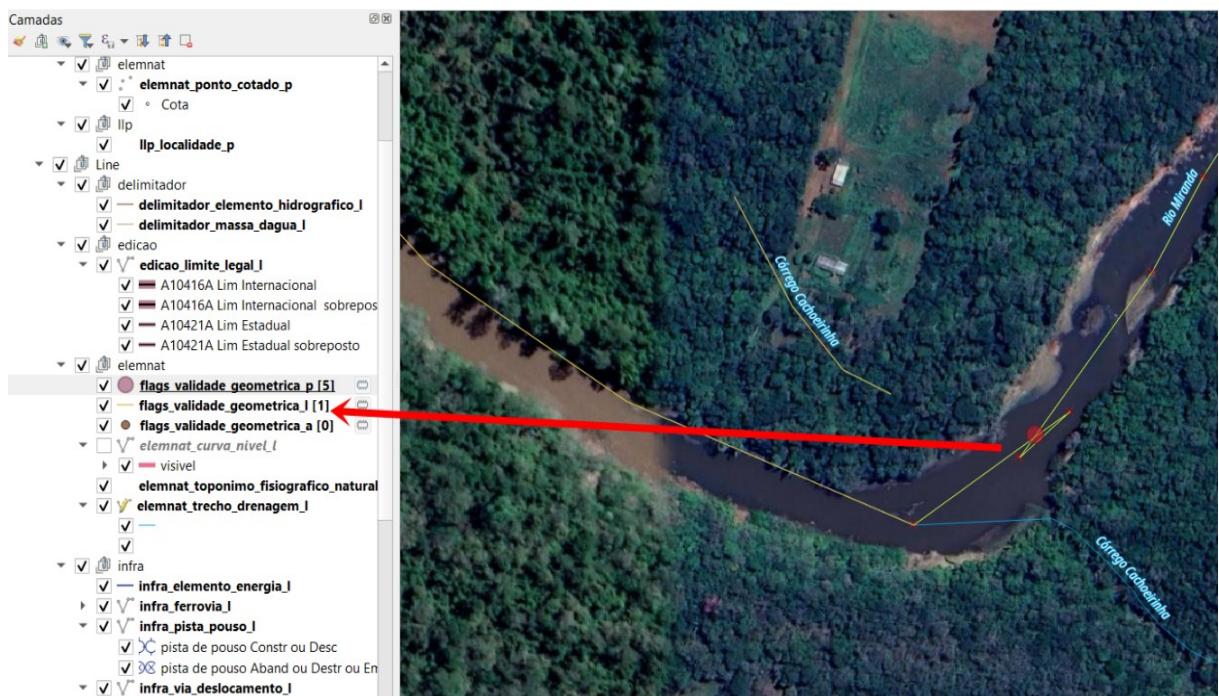


Figura 69: Exemplo de erros na geometria.

## 19.2. ROTINAS DE VALIDADE DE VÉRTICES

**a) Descrição:** Consiste em uma sequência de etapas que realizam a validação geométrica de feições.

**b) Arquivos:** rotinas\_validade\_de\_vertices\_alt\_hid.model3.

**c) Algoritmos:**

model:05-Identificar vértice não compartilhado nas intersecções;  
model:06-Identificar vértice não compartilhado nos segmentos compartilhados;  
model:07-Identificar vértice próximo de aresta;  
model:08-Identificar geometrias com densidade incorreta de vértices;  
model:09-Identificar ângulos pequenos;  
model:10-Identificar ângulos pequenos entre camadas;  
model:11-Identificar Z;  
native:mergevectorlayers.

**d) Composição do Modelo:**

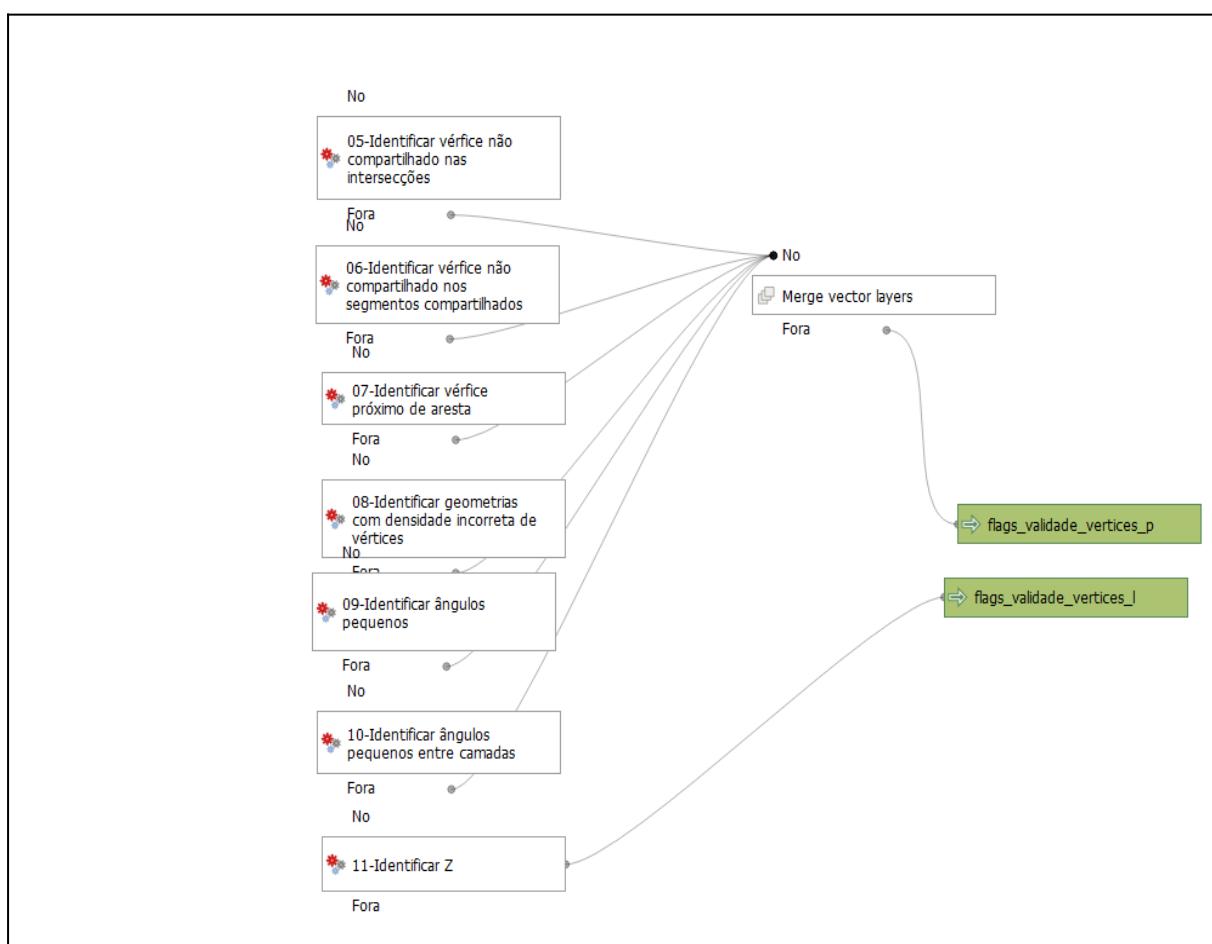


Figura 70: Modelo composto por modelos de identificação da validade de vértice.

**O modelo é composto por oito etapas:**

- 05-Identificar vértice não compartilhado nas intersecções;
- 06-Identificar vértice não compartilhado nos segmentos compartilhados;
- 07-Identificar vértice próximo de aresta;
- 08-Identificar geometrias com densidade incorreta de vértices;
- 09-Identificar ângulos pequenos;
- 10-Identificar ângulos pequenos entre camadas;
- 11-Identificar Z;

*Merge vector layers:* utiliza o algoritmo nativo *native:mergevectorlayers* para mesclar as camadas de *flags* de saída em uma única camada.

**e) Parâmetros:**

Os parâmetros utilizados nas rotinas estão inseridos dentro de cada modelo, uma vez que este conjunto é composto por múltiplos modelos. As informações referentes a esses parâmetros são detalhadas nos itens 5, 6,7,8,9,10 e 11 deste apêndice.

**f) nome da camada de *flags*:**

*flags\_validade\_vertices\_p;*  
*flags\_validade\_vertices\_l.*

### g) Resultado do processo e exemplo de erros:

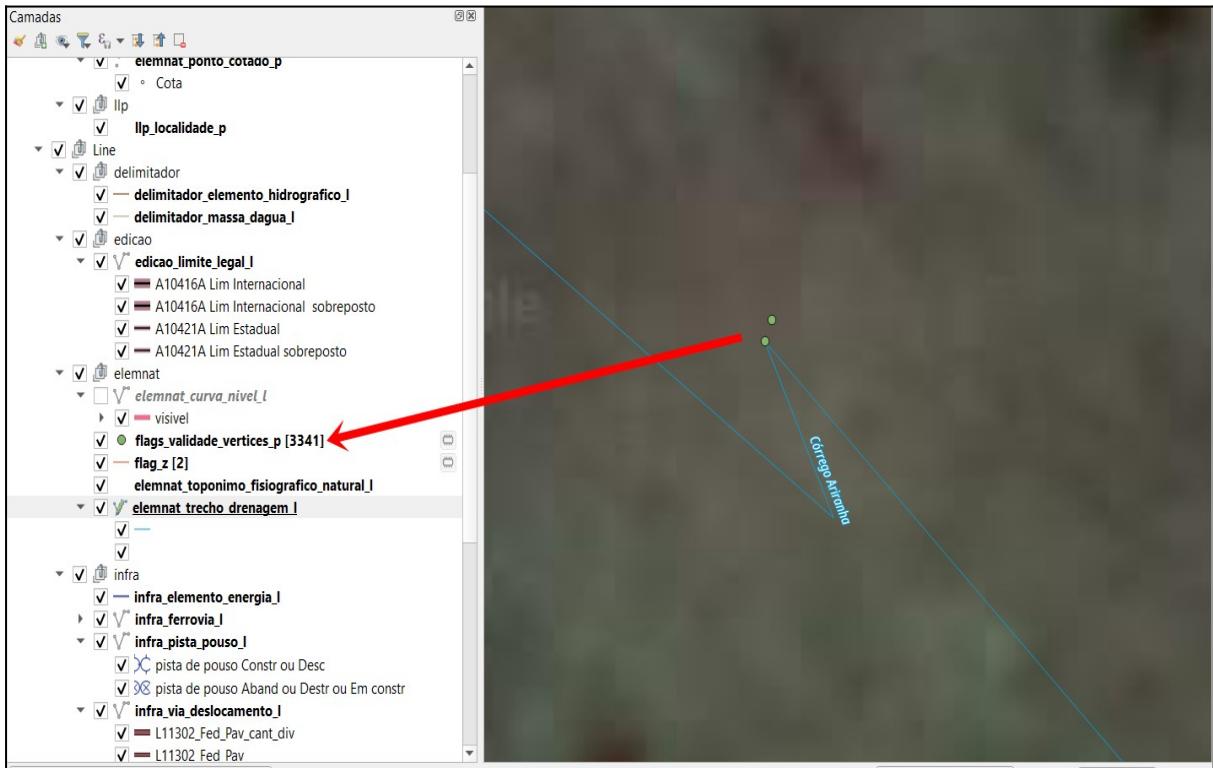


Figura 71: Exemplo de ângulos com construção incorreta.

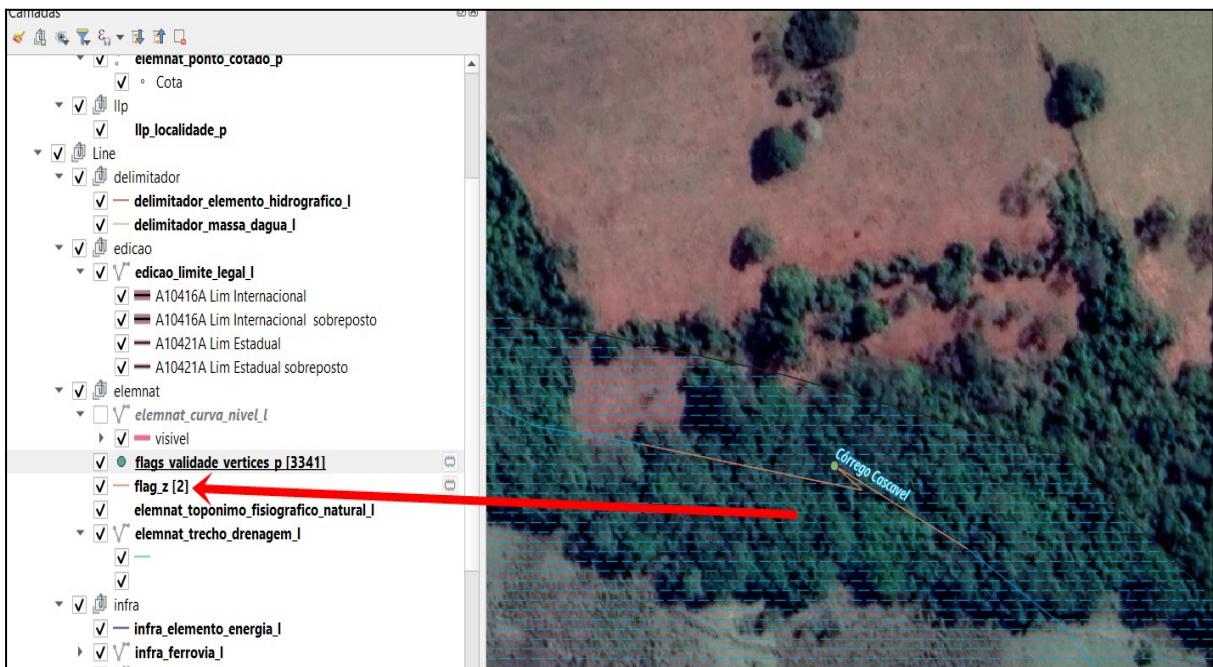


Figura 72: Exemplo de ângulos com construção incorreta.

### 19.3. IDENTIFICA PONTAS SOLTAS EM DELIMITADORES DE CORPOS DE ÁGUA

**a) Descrição:** Este algoritmo tem a função de identificar pontas soltas na camada de delimitadores, a fim de buscar linhas que não estão fechadas corretamente, resultando em erros ao gerar áreas.

**b) Arquivo:** identifica\_pontas\_livres\_limite\_massa\_dagua\_alt\_hid.model3.

**c) Algoritmo:**

```
dsgtools:stringcsvtofirstlayerwithelementsalgorithm;  
dsgtools:stringcsvtolayerlistalgorithm;  
dsgtools:identifydangles.
```

**d) Composição do Modelo:**

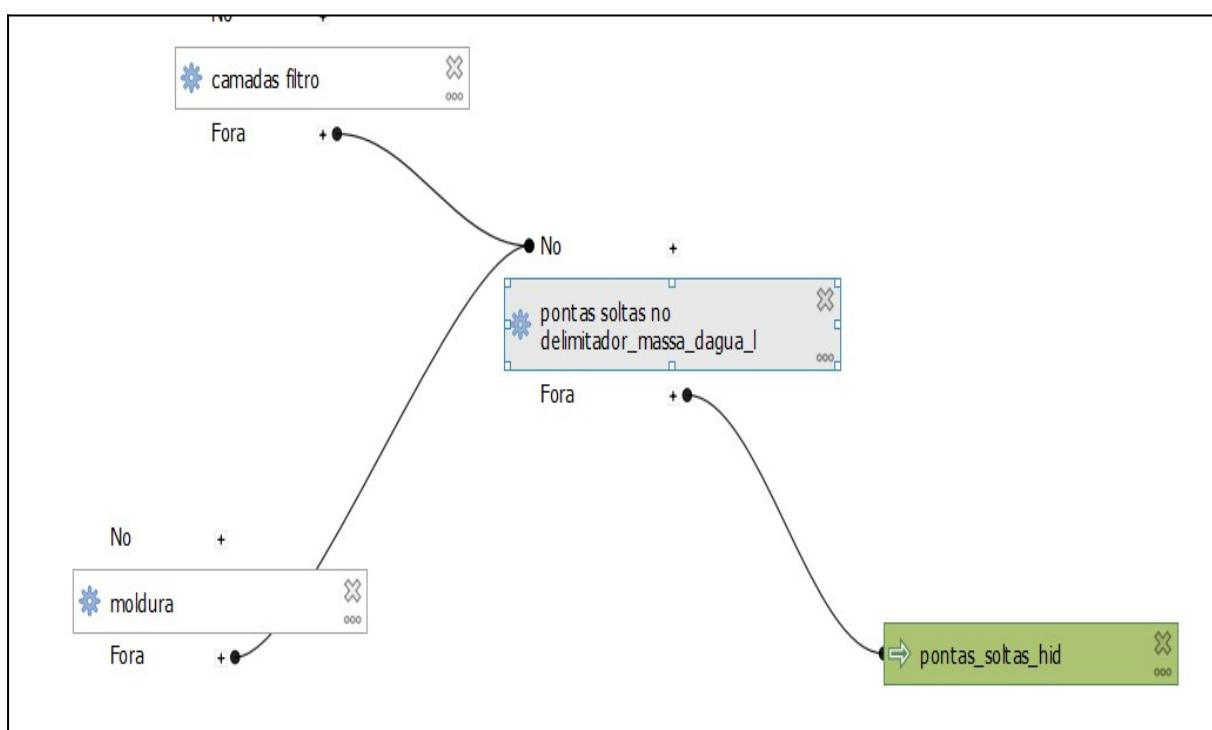


Figura 73: Modelo de identificação de pontas soltas nos delimitadores de massa d'água.

## O modelo é composto por três etapas:

Moldura: Converte uma *string* csv em uma camada especificada usando o algoritmo *stringcsvtofirstlayerwithelementsalgorithm*.

Camadas de filtro: converte uma *string* csv em camadas especificadas sendo posteriormente utilizadas como filtro no algoritmo de identificação de pontas soltas utilizando o algoritmo *stringcsvtolayerlistalgorithm*.

Pontas soltas no delimitador: Executa os algoritmos *identifydangles* para as camadas de linha.

### e) Parâmetros:

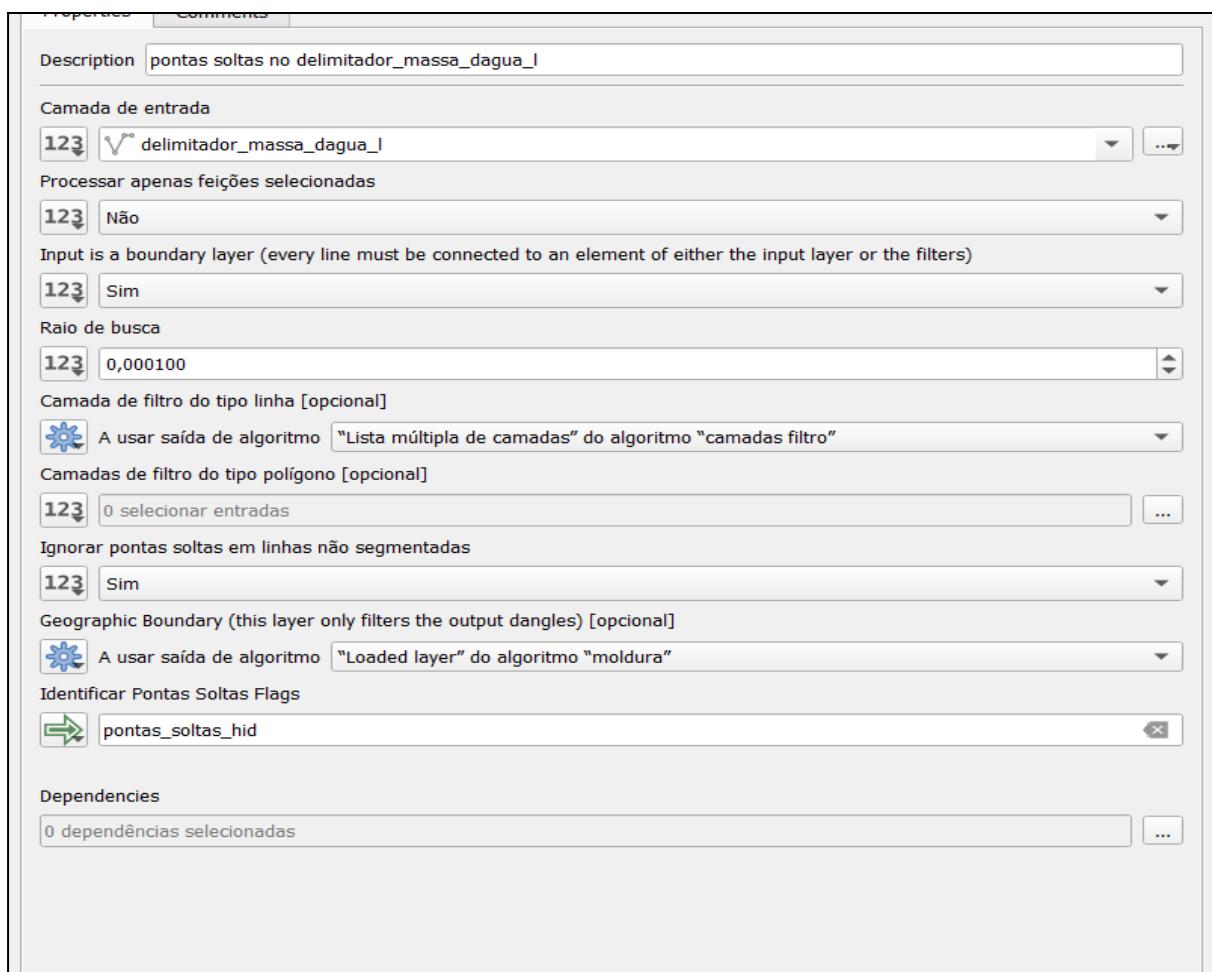


Figura 74: Extrato dos parâmetros.

### Camadas de entrada:

Tipo de geometria	Camadas de entrada
Linha	infra_barragem_1, elemnat_elemento_hidrografico_1, delimitador_massa_dagua_1.
Polígono	aux_moldura_a, moldura, aux_moldura_area_continua_a.

O parâmetro "**INPUT**" determina se a validação deve ser realizada apenas nas geometrias selecionadas pelo usuário.

O parâmetro "**SELECTED**" determina se a validação deve ser realizada apenas nas geometrias selecionadas pelo usuário. Quando marcado como "*true*", a validação é realizada apenas nas geometrias selecionadas. Caso contrário, todas as geometrias são validadas.

No parâmetro "**INPUT IS A BOUNDARY LAYER**" se a opção "*true*" estiver definida no campo “delimitador\_massa\_dagua\_1”, isso significa que a linha precisa ser fechada corretamente, ou seja, estar conectada a elementos da camada de entrada ou dos filtros.

O Parâmetro "**SEARCH RADIUS**" do algoritmo determina a distância máxima que será percorrida para encontrar pontas soltas em relação aos segmentos de linha e polígonos que se cruzam nas intersecções. No modelo, o valor padrão do raio de busca é definido como 0,00001 graus no sistema de coordenadas geográfico que corresponde a um valor aproximado de 1 metro. No entanto, é importante lembrar que esse valor pode ser ajustado de acordo com as necessidades específicas, considerando a escala e precisão dos dados utilizados no processamento

O parâmetro "**CAMADA DE FILTRO**" tem como proposta identificar pontas soltas próximas às camadas especificadas. Essas pontas soltas serão adicionadas à regra de verificação de proximidade durante a busca por feições próximas às analisadas, a fim de minimizar a ocorrência de falsos positivos.

O parâmetro "**IGNORAR PONTAS SOLTAS EM LINHAS NÃO SEGMENTADAS**" quando marcado como "*true*" permite que o algoritmo ignore pontas

soltas em linhas não segmentadas, ou seja, ele não considera como *undershoots* aquelas pontas soltas que não estão conectadas a outros segmentos de linha. Ao ignorar essas pontas, o algoritmo pode concentrar-se apenas nas linhas que realmente importam e fornecer resultados mais precisos.

O parâmetro "**GEOGRAPHIC BOUNDARY**" se refere à delimitação geográfica para a execução do processamento em uma região determinada. A "moldura" representa a área delimitada em questão, isso evita que sejam incluídas informações irrelevantes ou imprecisas de outras áreas que estão fora da delimitação.

O parâmetro "**FLAGS**" é utilizado para definir o nome da camada que será criada pela rotina para armazenar as informações de validação.

**f) nome da camada de flags:** pontas\_soltas\_hid.

**g) Resultado do processo e exemplo de erros:**

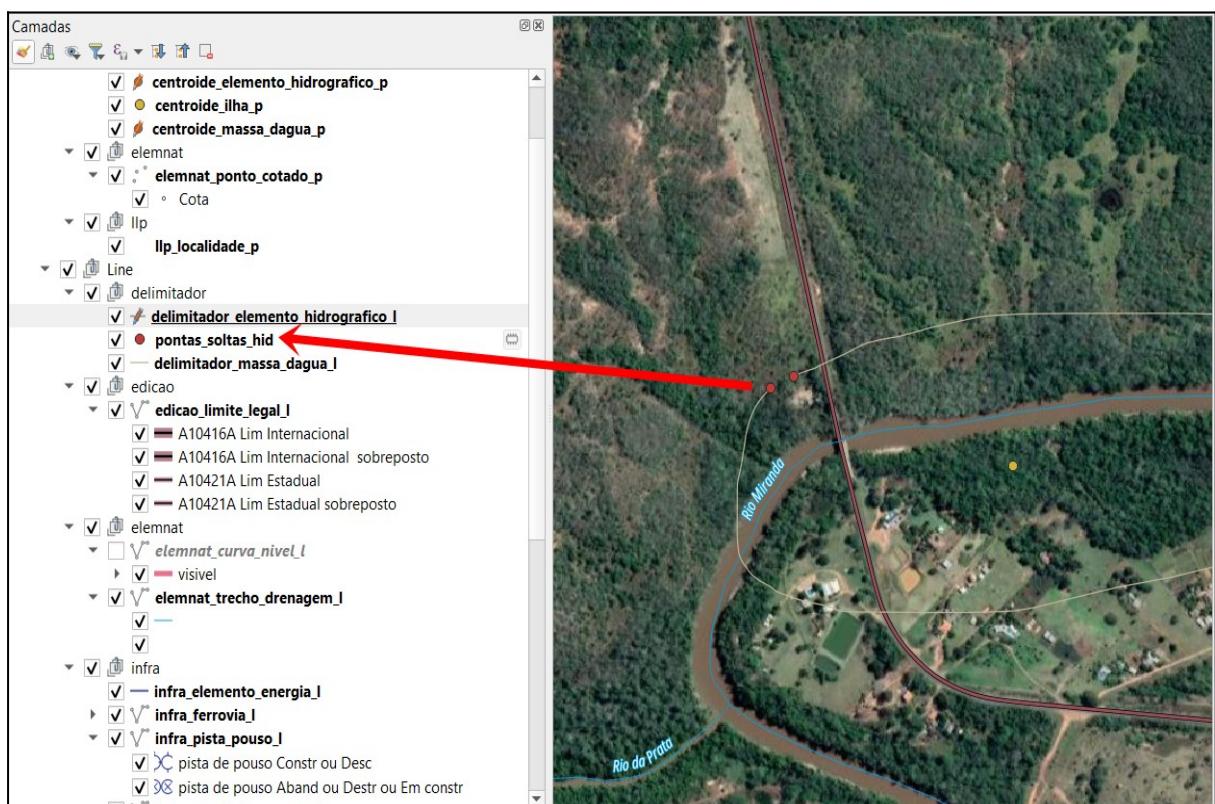


Figura 75: Exemplo de ponta solta no delimitador.

## 19.4. IDENTIFICA PONTAS SOLTAS EM DELIMITADORES DE ELEMENTOS HIDROGRÁFICOS

**a) Descrição:** Este algoritmo tem a função de identificar pontas soltas na camada de delimitadores de elementos hidrográficos, a fim de buscar linhas que não estão fechadas corretamente, resultando em erros ao gerar áreas.

**b) Arquivo:** identifica\_pontas\_livres\_elem\_hidrografico\_alt\_hid.model3.

**c) Algoritmo:**

```
dsgtools:stringcsvtofirstlayerwithelementsalgorithm;  
dsgtools:stringcsvtolayerlistalgorithm;  
dsgtools:identifydangles.
```

**d) Composição do Modelo:**

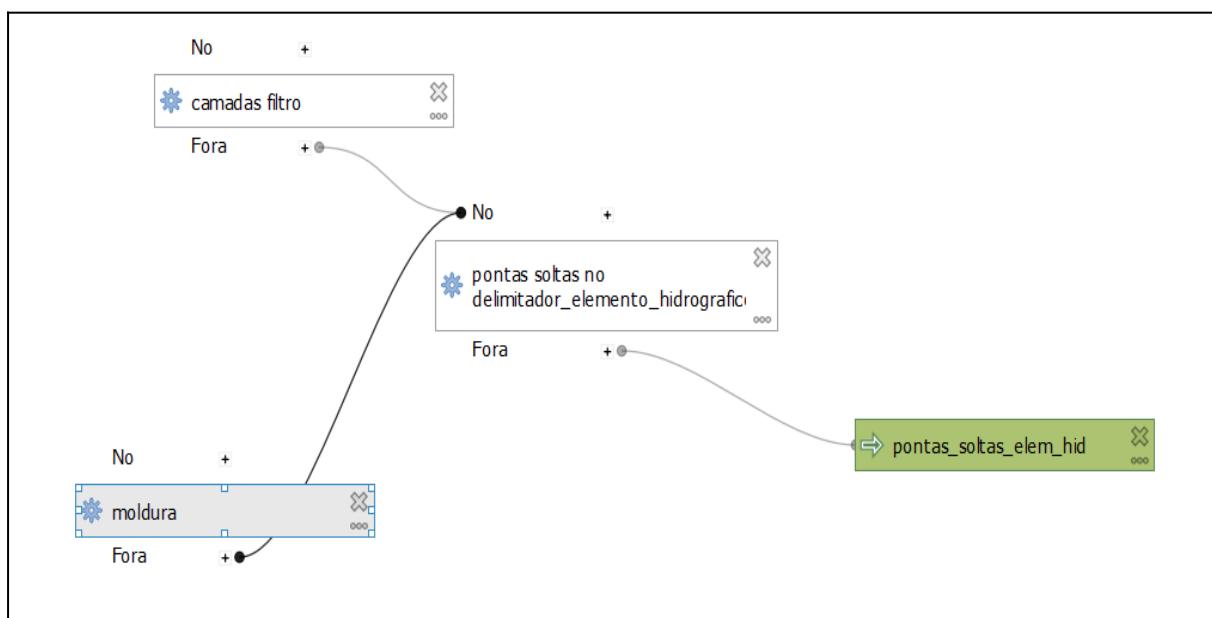


Figura 76: Modelo de identificação de pontas soltas no delimitador de elemento hidrográfico.

### **O modelo é composto por três etapas:**

Moldura: Converte uma *string* csv em uma camada especificada usando o algoritmo *stringcsvtofirstlayerwithelementsalgorithm*.

Camadas de filtro: converte uma *string* csv em camadas especificadas sendo posteriormente utilizadas como filtro no algoritmo de identificação de pontas soltas utilizando o algoritmo *stringcsvtolayerlistalgorithm*.

Pontas soltas no delimitador: Executa os algoritmos *identifydangles* para as camadas de linha com os parâmetros descritos na seção e.

#### **e) Parâmetros:**

Os parâmetros presentes neste modelo correspondem ao da rotina: “Fechar delimitadores de corpos de água”, com a única modificação relacionada à camada de entrada.

#### **f) nome da camada de *flags*:** *pontas\_soltas\_elem\_hid*.

### g) Resultado do processo e exemplo de erros:

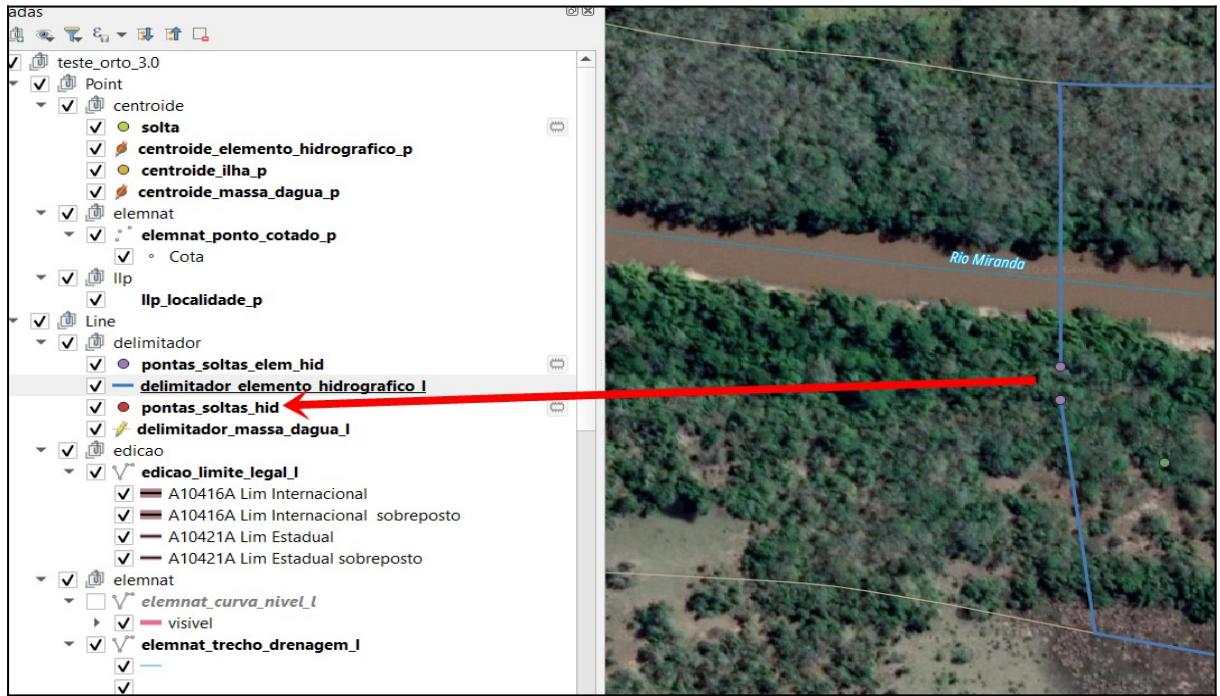


Figura 77: Exemplo de ponta solta no delimitador.

## 19.5. FECHAR ELEMENTOS HIDROGRÁFICOS E ILHAS

**a) Descrição:** Este modelo tem como objetivo a geração de polígonos de ilha e elementos hidrográficos por meio da manipulação vetorial, utilizando a camada delimitadora de linha como entrada.

**b) Arquivos:** fechar\_poligonos\_elem\_hidrograficos\_e\_ilhas.model3.

**c) Algoritmos:**

```
dsgtools:stringcsvtofirstlayerwithelementsalgorithm;
dsgtools:stringcsvtolayerlistalgorithm;
dsgtools:buildpolygonsfromcenterpointsandboundariesalgorithm;
native:mergevectorlayers.
```

#### d) Composição do Modelo:

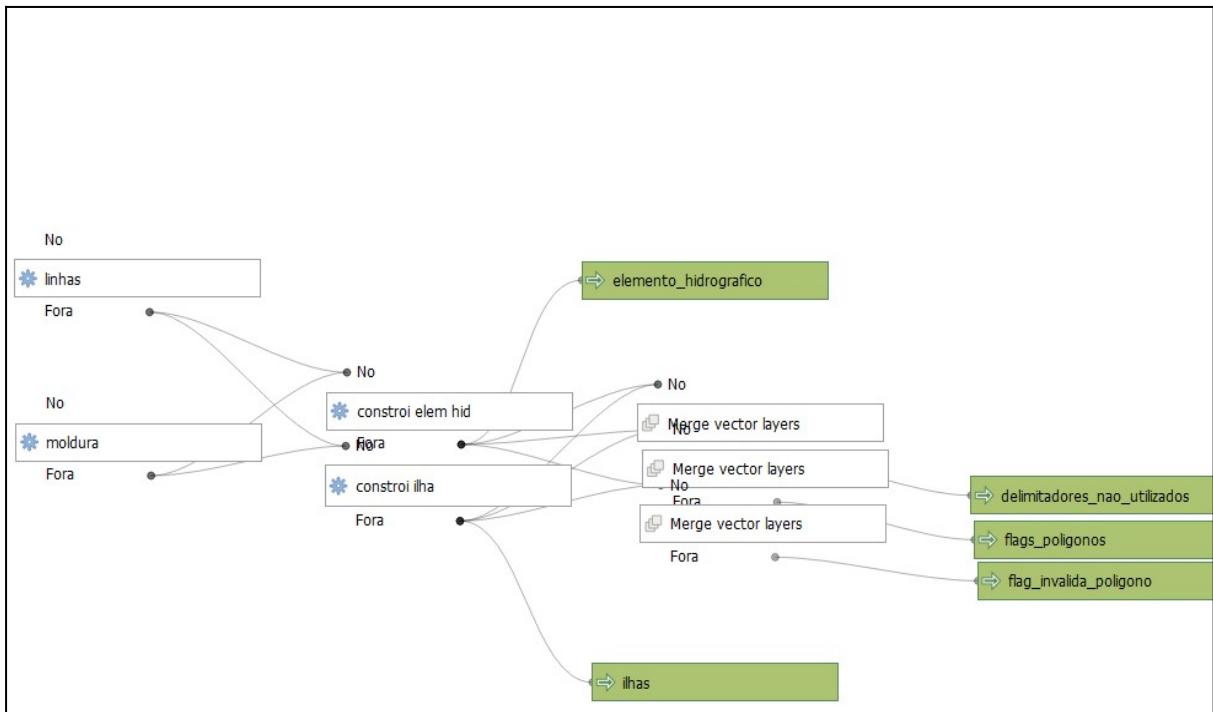


Figura 78: Modelo de construção de áreas de elemento hidrográfico e ilhas.

#### O modelo é composto por quatro etapas:

*String CSV to Layer List Algorithm:* Converte uma string csv em uma camada especificada usando o algoritmo *stringcsvtolayerlistalgorithm*.

Moldura: Converte uma string csv em uma camada especificada usando o algoritmo *stringcsvtofirstlayerwithelementsalgorithm*.

Construir polígonos a partir dos centroides: O algoritmo em questão tem como objetivo criar polígonos que representem as ilhas e os elementos hidrográficos a partir de linhas de restrição (delimitadores), as quais delimitam os limites do polígono a ser construído. Para tanto, utiliza-se um ponto central que contém as informações principais acerca da geometria a ser gerada. Ademais, o polígono recém-construído herda os atributos do ponto de centroide. É relevante destacar que o algoritmo emprega o parâmetro *delimitador\_massa\_dagua\_1* para gerar as

ilhas, enquanto utiliza o parâmetro `delimitador_elemento_hidrografico_l` para gerar os elementos hidrográficos. Este procedimento é realizado utilizando o algoritmo:

*Buildpolygonsfromcenterpointsandboundariesalgorithm.*

*Merge Vector Layers:* Realiza uma operação de unir as camadas de saída utilizando o algoritmo *mergevectorlayers*.

#### e) Parâmetros:

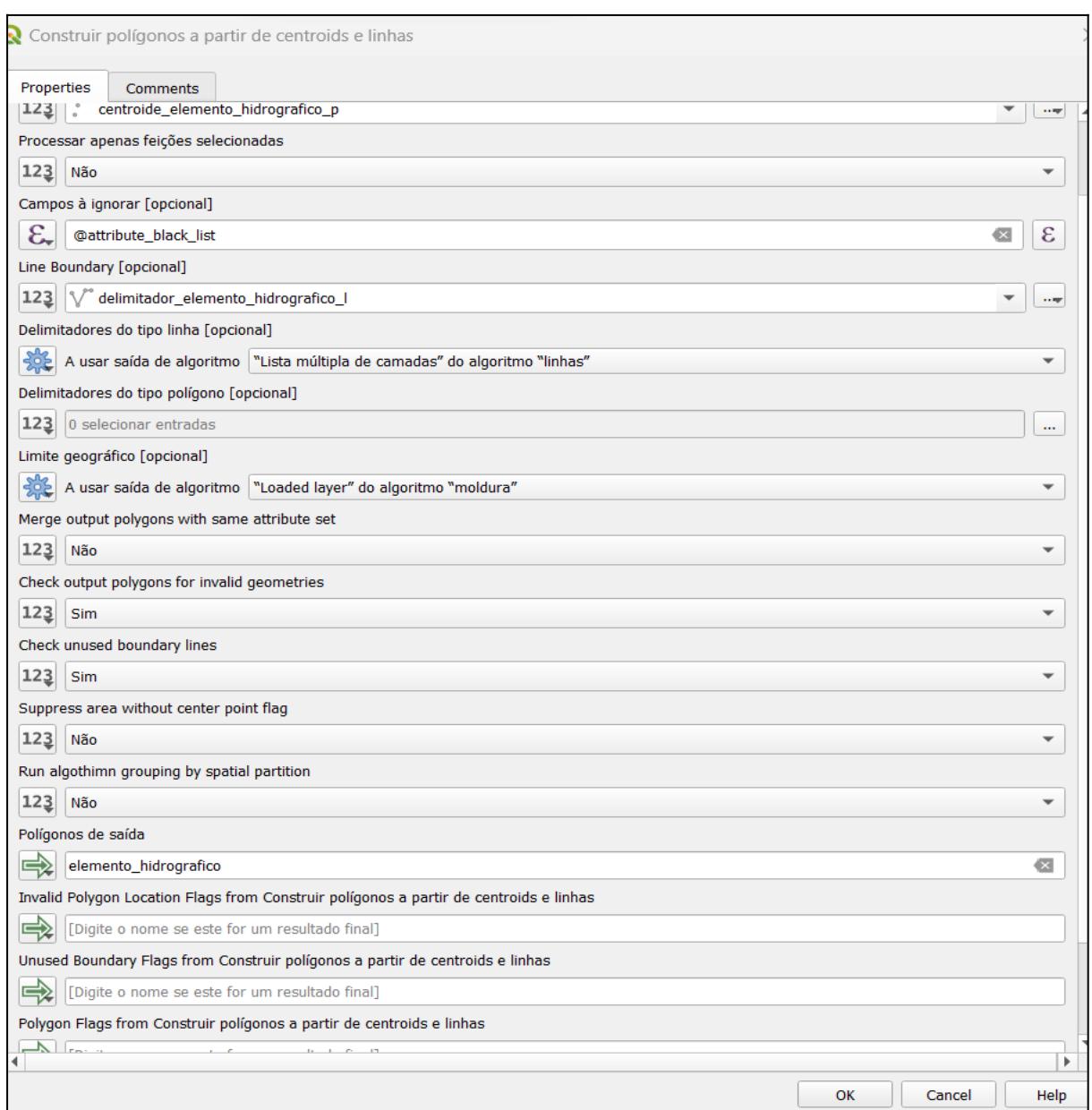


Figura 79: Extrato dos parâmetros.

## Camadas de entrada:

Tipo de geometria	Camadas de entrada
Ponto	centroide_ilha_p, centroide_elemento_hidrografico_p.
Linha	delimitador_massa_dagua_l, delimitador_elemento_hidrografico_l.
Polígono	aux_moldura_a, moldura, aux_moldura_area_continua_a.

O parâmetro "**SELECTED**" determina se a geração dos polígonos deve ser realizada apenas nas geometrias selecionadas pelo usuário. Quando marcado como "true", a execução é realizada apenas nas geometrias selecionadas. Caso contrário, todas as geometrias são processadas.

No parâmetro "**CAMPOS A IGNORAR**" é definido uma lista negra de atributos (*attribute blacklist*) que não serão considerados na hora de herdar os atributos dos pontos de centroide. O valor do parâmetro `@attribute_black_list` refere-se a uma variável presente nas variáveis do modelo que contém uma lista dos campos a serem ignorados, separados por vírgula.

O Parâmetro "**LINE BOUNDARY**" é usado para definir a camada de linha que será usada como limite (delimitadores) para criar polígonos a partir dos centroides.

Nos parâmetros "**DELIMITADORES DO TIPO LINHA E POLIGONO**" é usado para definir as demais camadas de linhas ou de polígonos que será usada como limite (delimitadores) para criar os polígonos a partir dos centroides.

O parâmetro "**CAMADA DE LIMITE GEOGRÁFICO**" se refere à delimitação geográfica para a execução do processamento em uma região determinada. A “moldura” representa a área delimitada em questão, isso evita que sejam incluídas informações irrelevantes ou imprecisas de outras áreas que estão fora da delimitação.

O parâmetro "**MERGE OUTPUT POLYGONS**" com o valor "false" no algoritmo indica que os polígonos resultantes não serão mesclados em um único polígono, mas sim mantidos separados.

O parâmetro "**CHECK OUTPUT POLYGONS**" é usado para verificar se os polígonos produzidos são válidos ou não. Quando definido "true", o algoritmo verifica se há polígonos inválidos na saída e se houver retorna um conjunto de geometrias inválidas na "flag" de parâmetro: "**INVALID\_Polygon\_LOCATION**".

O parâmetro "**SUPPRESS AREA WITHOUT CENTER POINT FLAG**" com o valor "true", indica que áreas sem centroide devem ser suprimidas na saída, ou seja, somente as áreas que possuem um ponto central definido serão consideradas na construção dos polígonos. Esse parâmetro é utilizado para evitar a criação de polígonos em áreas sem informação.

O parâmetro "**FLAGS**" é composto por dois tipos distintos de informações: o primeiro refere-se às localizações que necessitam de correção, enquanto o segundo se relaciona às camadas de massa de ilhas e elementos hidrográficos recém-criados que serão posteriormente copiados.

#### f) nome da camada de flags:

*flags* de correção: *flag\_poligonos*, *flag\_invalida\_poligono*, *delimitadores\_nao\_utilizados*;

*flags* das camadas criadas: *elemento\_hidrografico*, Ilhas.

#### g) Resultado do processo e exemplo de erros:



Figura 80: Exemplo de construção de Elementos hidrográficos e ilha.