

APÊNDICE B - *WORKFLOW* DE TRANSPORTES

Sumário

1. IDENTIFICAR GEOMETRIAS INVÁLIDAS.....	2
2. IDENTIFICAR GEOMETRIAS COM MAIS DE UMA PARTE.....	5
3. IDENTIFICAR FEIÇÕES DUPLICADAS.....	9
4. IDENTIFICAR VÉRTICE NÃO COMPARTILHADO NAS INTERSECÇÕES.....	13
5. IDENTIFICAR VÉRTICE NÃO COMPARTILHADO NOS SEGMENTOS COMPARTILHADOS.....	16
6. IDENTIFICAR VÉRTICE PRÓXIMO DE ARESTA.....	19
7. IDENTIFICAR GEOMETRIAS COM DENSIDADE INCORRETA DE VÉRTICES.....	22
8. IDENTIFICAR ÂNGULOS PEQUENOS.....	26
9. IDENTIFICAR ÂNGULOS PEQUENOS ENTRE CAMADAS.....	30
10. IDENTIFICAR Z.....	34
11. IDENTIFICAR OVERLAPS DENTRO DA MESMA CAMADA.....	37
12. IDENTIFICAR <i>UNDERSHOOT</i> COM MOLDURA E CONEXÃO DE LINHAS.....	40
13. IDENTIFICAR LINHAS SEGMENTADAS COM MESMO CONJUNTO DE ATRIBUTOS.....	44
14. IDENTIFICAR LINHAS NÃO SEGMENTADAS NAS INTERSECÇÕES.....	48
15. IDENTIFICAR ELEMENTOS PEQUENOS NA REDE.....	52
16. IDENTIFICAR ERROS NA CONSTRUÇÃO DAS REDES DE RODOVIÁRIAS E FERROVIÁRIAS.....	56

1. IDENTIFICAR GEOMETRIAS INVÁLIDAS

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

b) **Arquivo:** identifica_geometrias_invalidas_transportes_carta_orto.model3.

c) **Algoritmos:**

dsgtools: batchrunalgorithm;

dsgtools: identifyandfixinvalidgeometries.

d) **Composição do Modelo:**

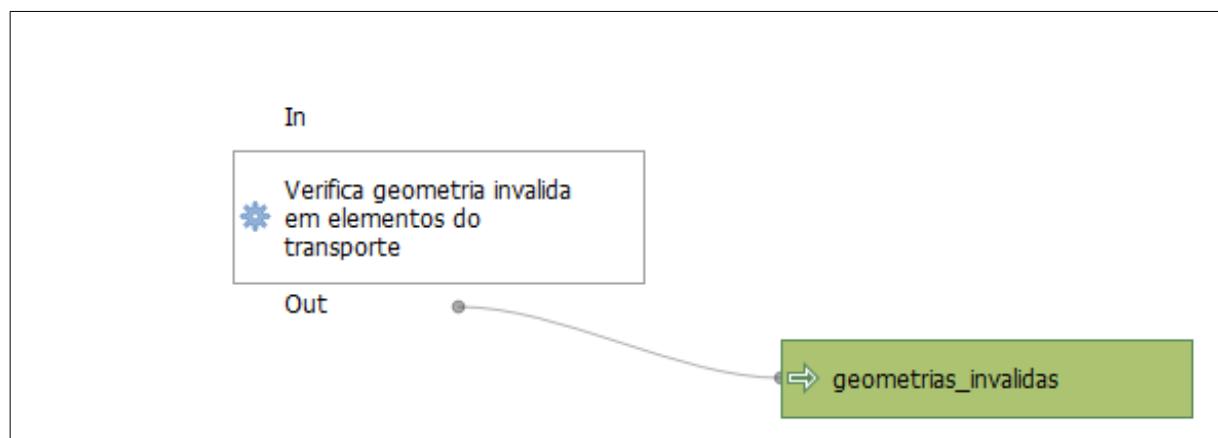


Figura 1: Modelo de identificação de geometria inválida.

O modelo é composto por uma etapa:

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

e) Parâmetros:

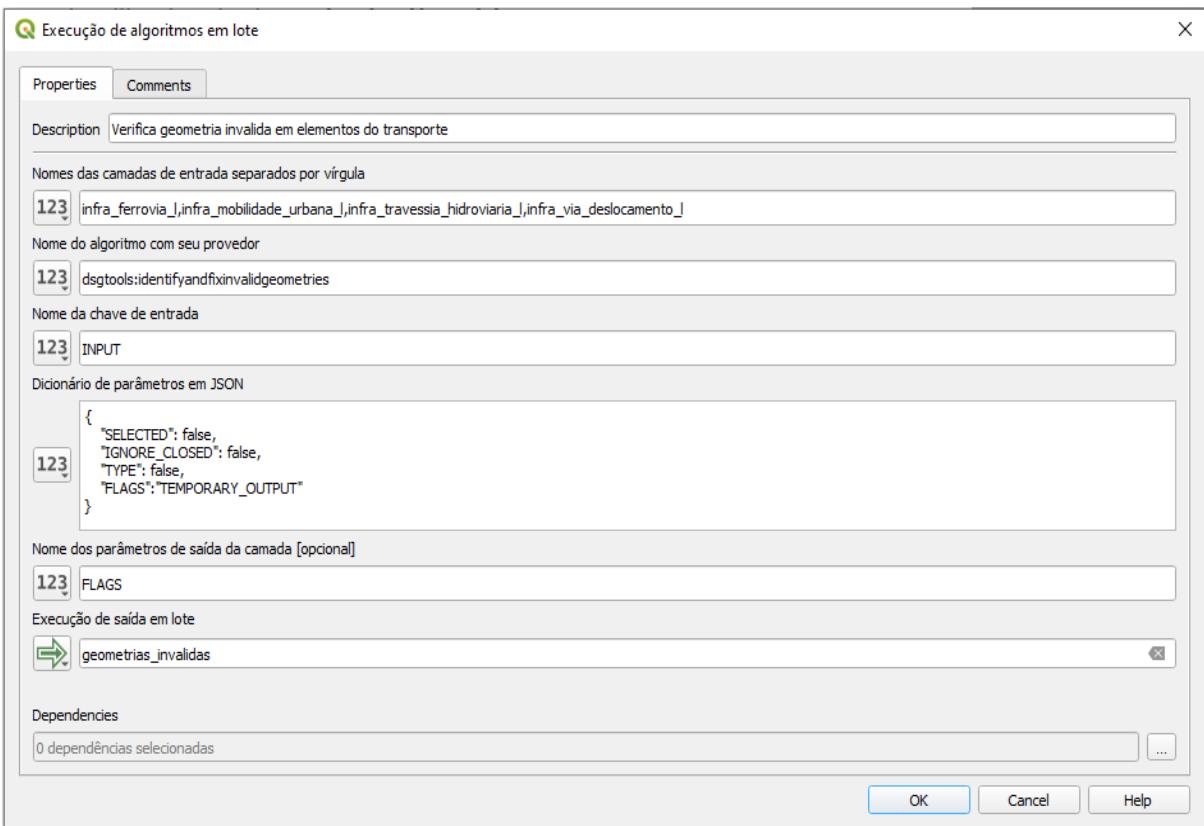


Figura 2: Extrato dos Parâmetros.

Camadas de entrada:

Tipo de Geometria	Camadas de Entrada
Linha	infra_ferrovia_1, infra_mobilidade_urbana_1, infra_travessia_hidroviaria_1, infra_via_deslocamento_1.

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

Verifica geometria invalida em elementos do transporte:

```
{  
    "SELECTED": false,  
    "IGNORE_CLOSED": false,  
    "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.

f) Nome da camada de *flags* gerada: geometrias_invalidas.

g) Resultado do processo e exemplo de erro:

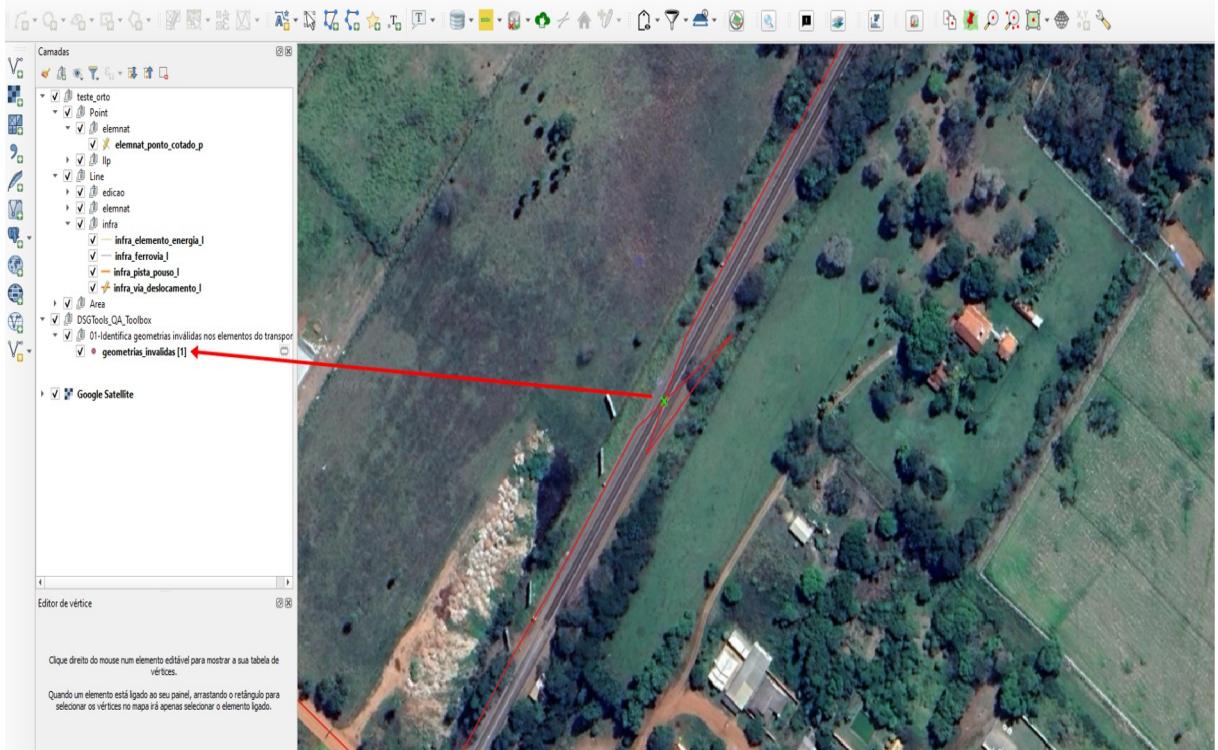


Figura 3: Exemplo de erro de geometria inválida.

2. IDENTIFICAR GEOMETRIAS COM MAIS DE UMA PARTE

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

b) Arquivo: identifica_multipart_transportes_carta_orto.model3.

c) Algoritmos:

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

d) Composição do Modelo:

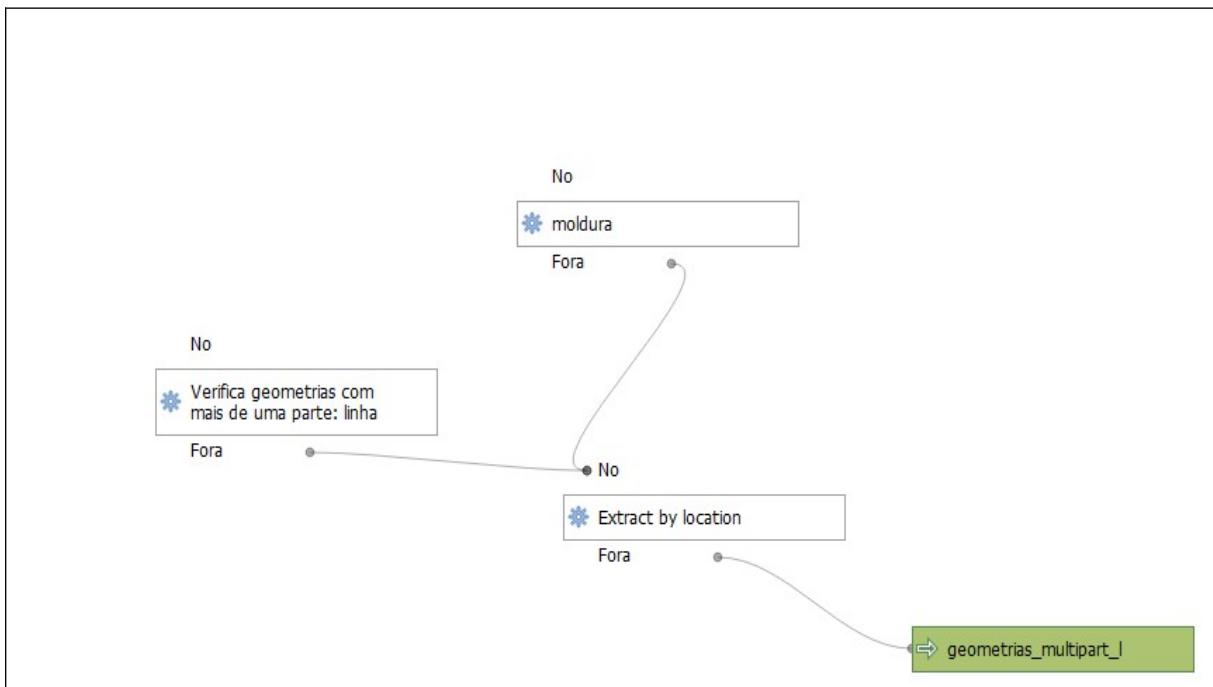


Figura 4: Modelo de identificação de geometria multiparte.

O modelo é composto por três etapas:

Verifica geometrias com mais de uma parte na camada de linha: Identifica geometrias com mais de uma parte na camada de linha usando o algoritmo *identifymultigeometries*.

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

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

e) Parâmetros:

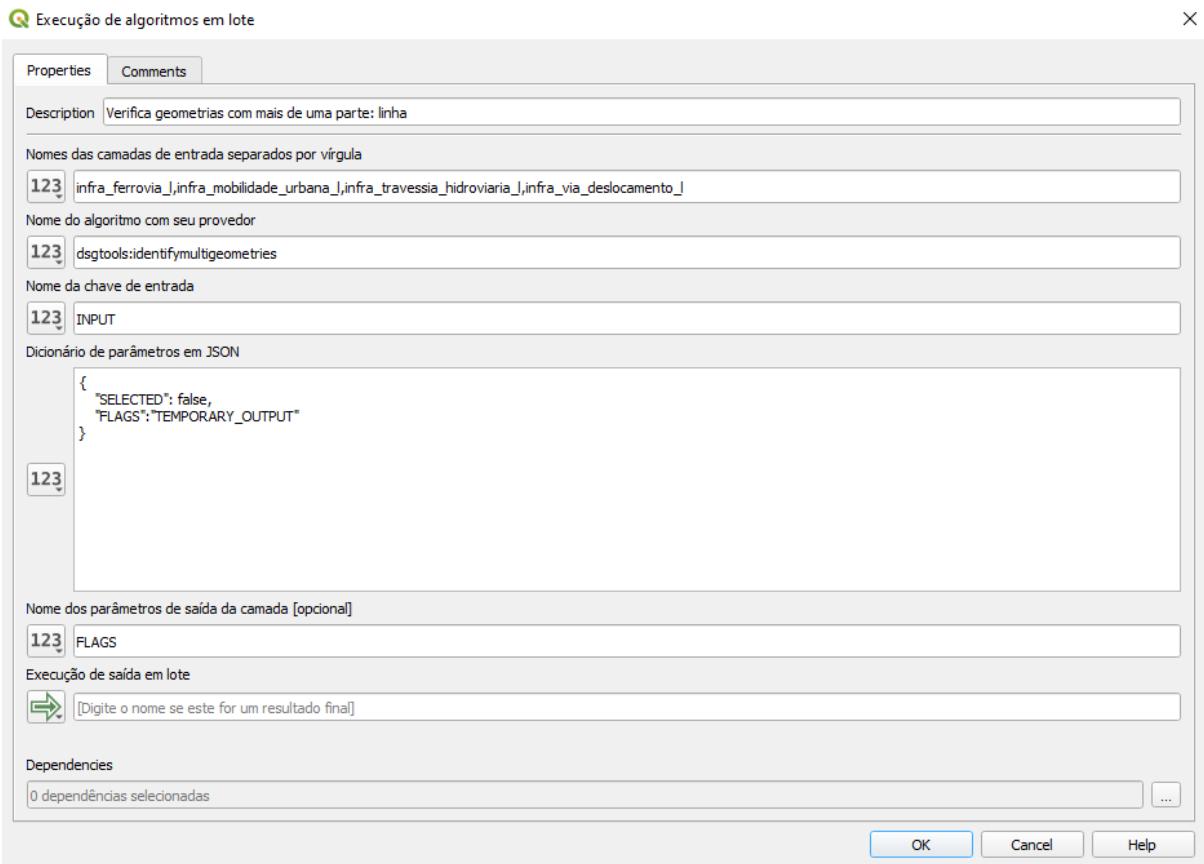


Figura 5: Extrato dos parâmetros.

Camadas de entrada:

Tipo de geometria	Camadas de entrada
Linha	infra_ferrovia_1, infra_mobilidade_urbana_1, infra_travessia_hidroviaria_1, infra_via_deslocamento_1.

Parâmetros em JSON:

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

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

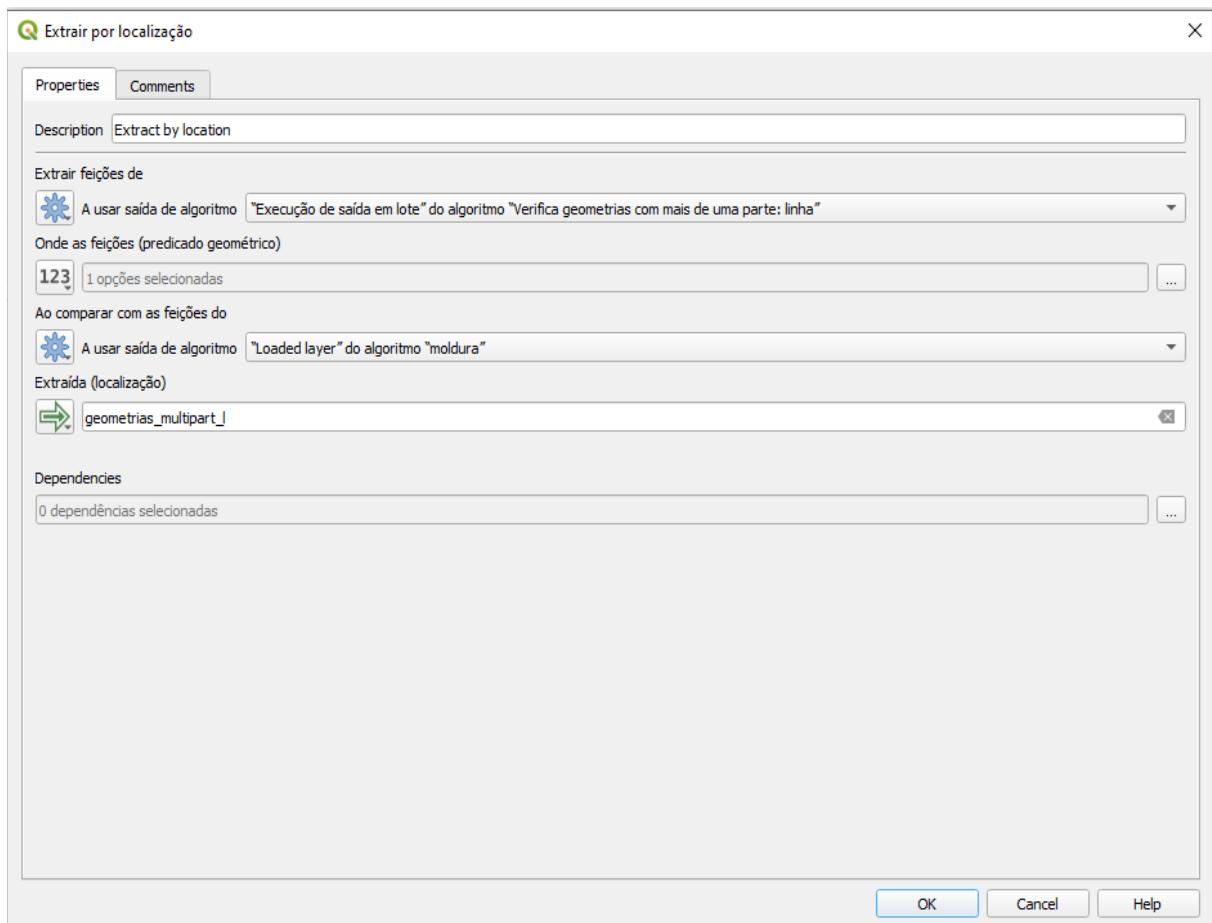


Figura 6: Extrair por localização.

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_1.

g) Resultado do processo e exemplo de erro:

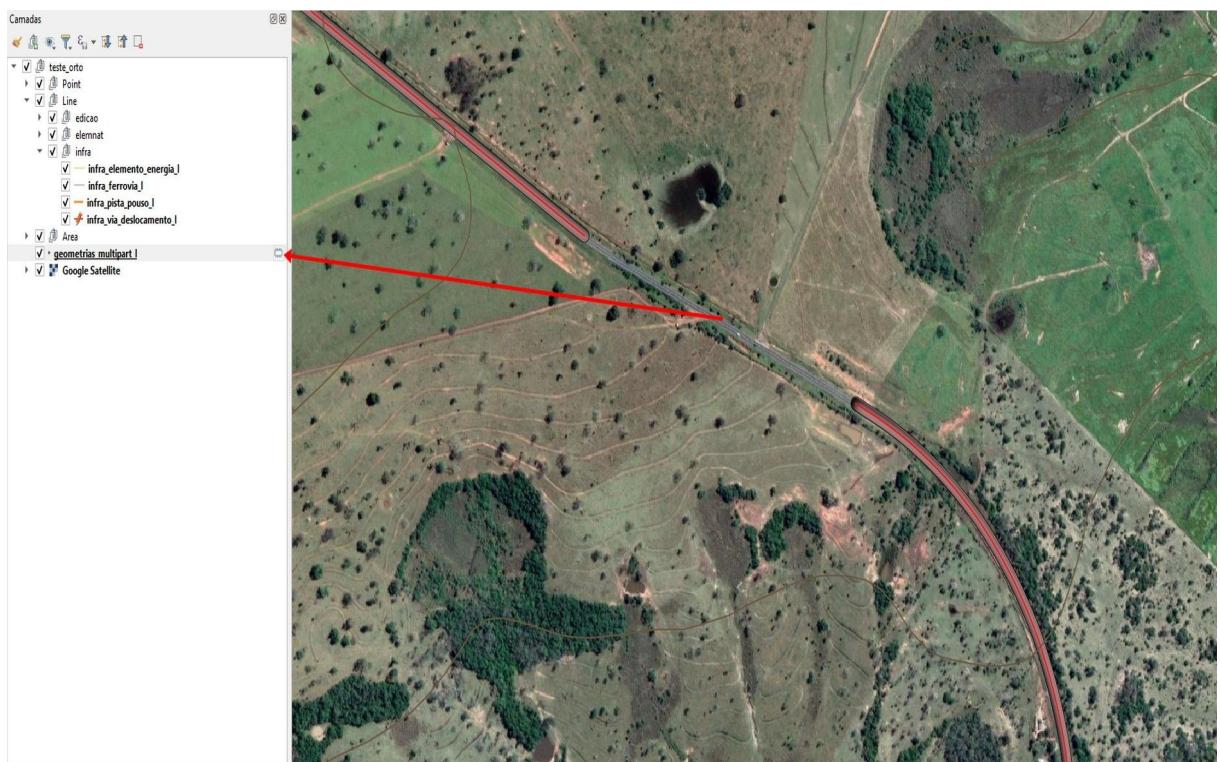


Figura 7: Exemplo de geometrias multiparte.

3. IDENTIFICAR FEIÇÕES DUPLICADAS

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

b) Arquivo: identifica_feicoes_duplicadas_transportes_carta_orto.model3.

c) Algoritmos:

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

d) Composição do Modelo:

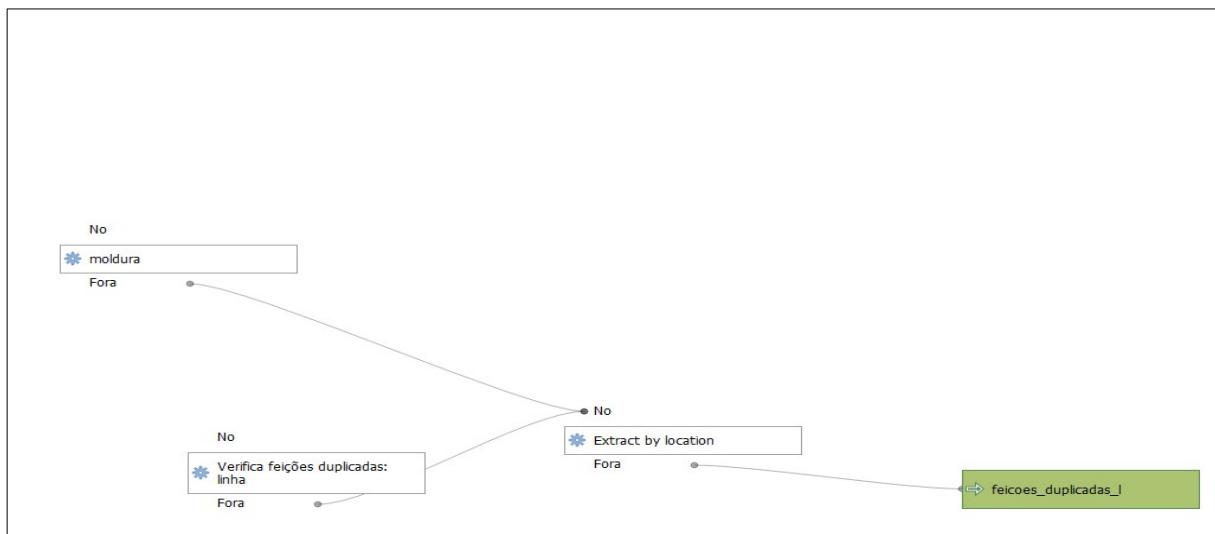


Figura 8: Modelo de identificação de geometrias duplicadas.

O modelo é composto por três etapas:

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

Verifica feições duplicadas: Identifica feições duplicadas nas camadas de linha usando o algoritmo *identifyduplicatedfeatures*.

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

e) Parâmetros:

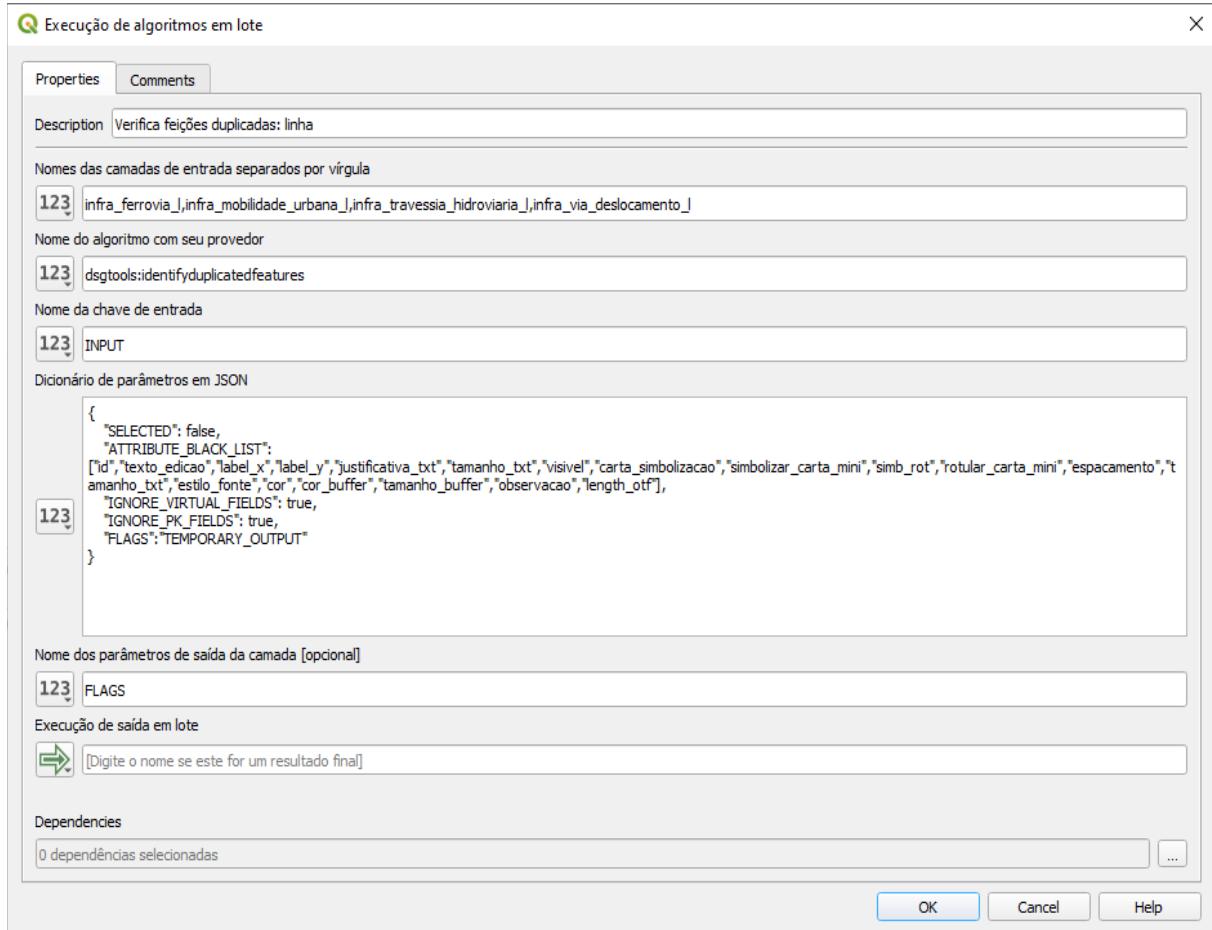


Figura 9: Extrato dos parâmetros.

Camadas de entrada:

Tipo de geometria	Camadas de entrada
Linha	infra_ferrovia_1, infra_mobilidade_urbana_1, infra_travessia_hidroviaria_1, infra_via_deslocamento_1.

Parâmetros em JSON:

```
{  
    "SELECTED": false,  
    "ATTRIBUTE_BLACK_LIST":  
        ["id","texto_edicao","label_x","label_y","justificativa_txt","tamanho_txt","visivel","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_1.

g) Resultado do processo e exemplo de erro:

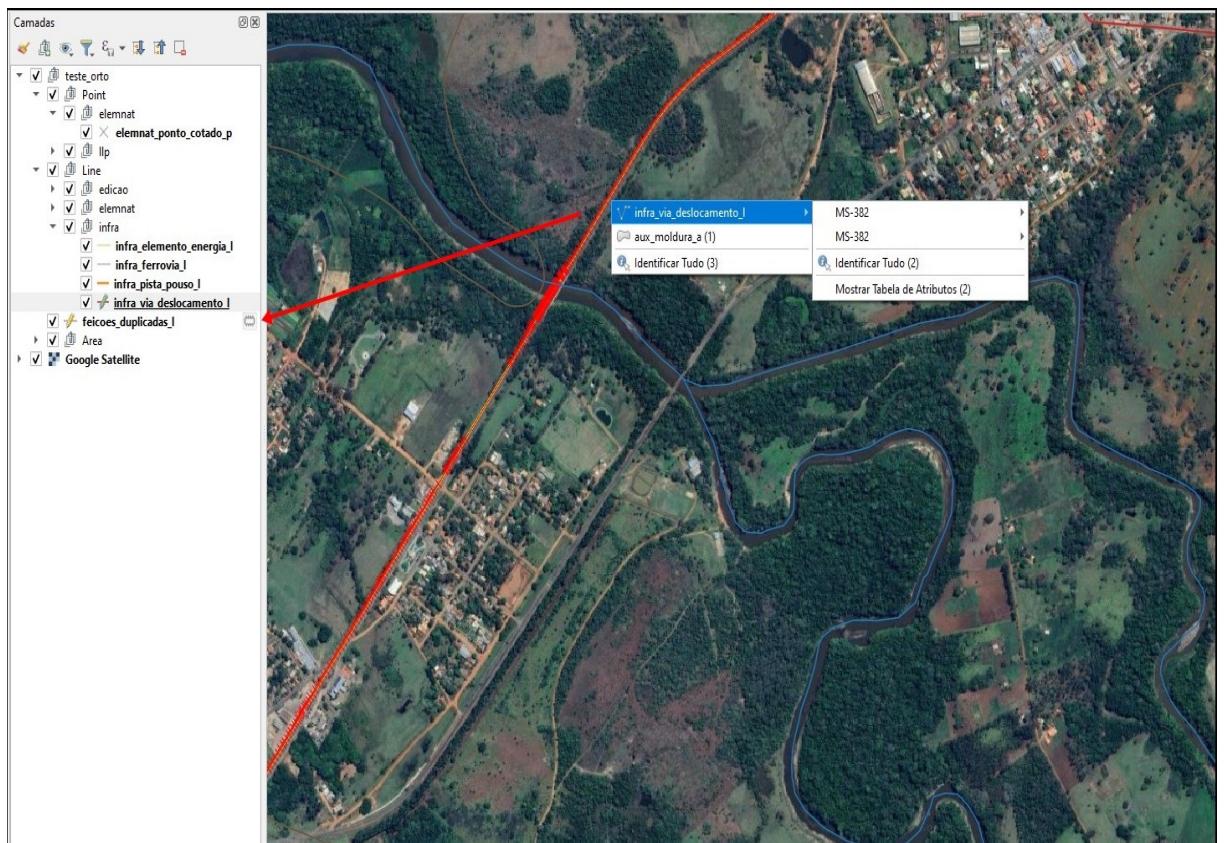


Figura 10: Exemplo de feições duplicadas.

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_transportes_carta_orto.model3.

c) Algoritmos:

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

d) Composição do Modelo:

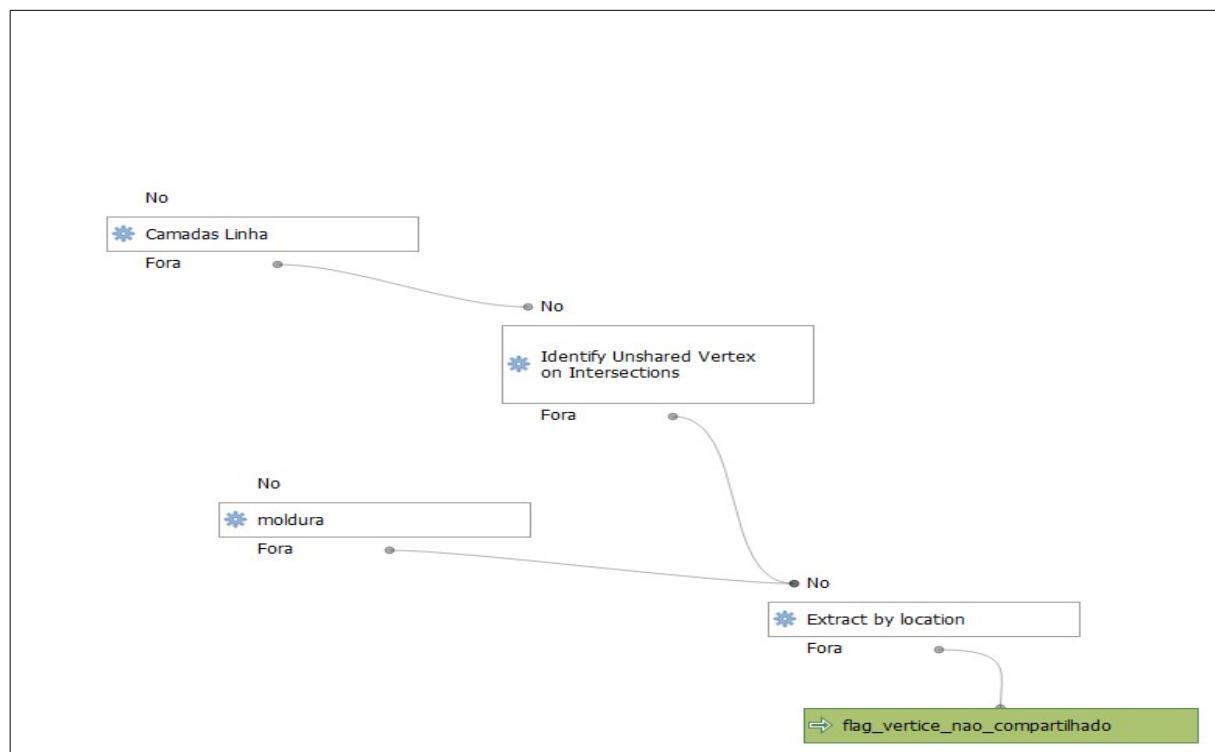


Figura 11: Modelo de identificação de vértices não compartilhados nas intersecções.

O modelo é composto por quatro etapas:

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 de entidades usando o algoritmo *stringcsvtofirstlayerwithelementsalgorithm*.

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

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

e) Parâmetros:

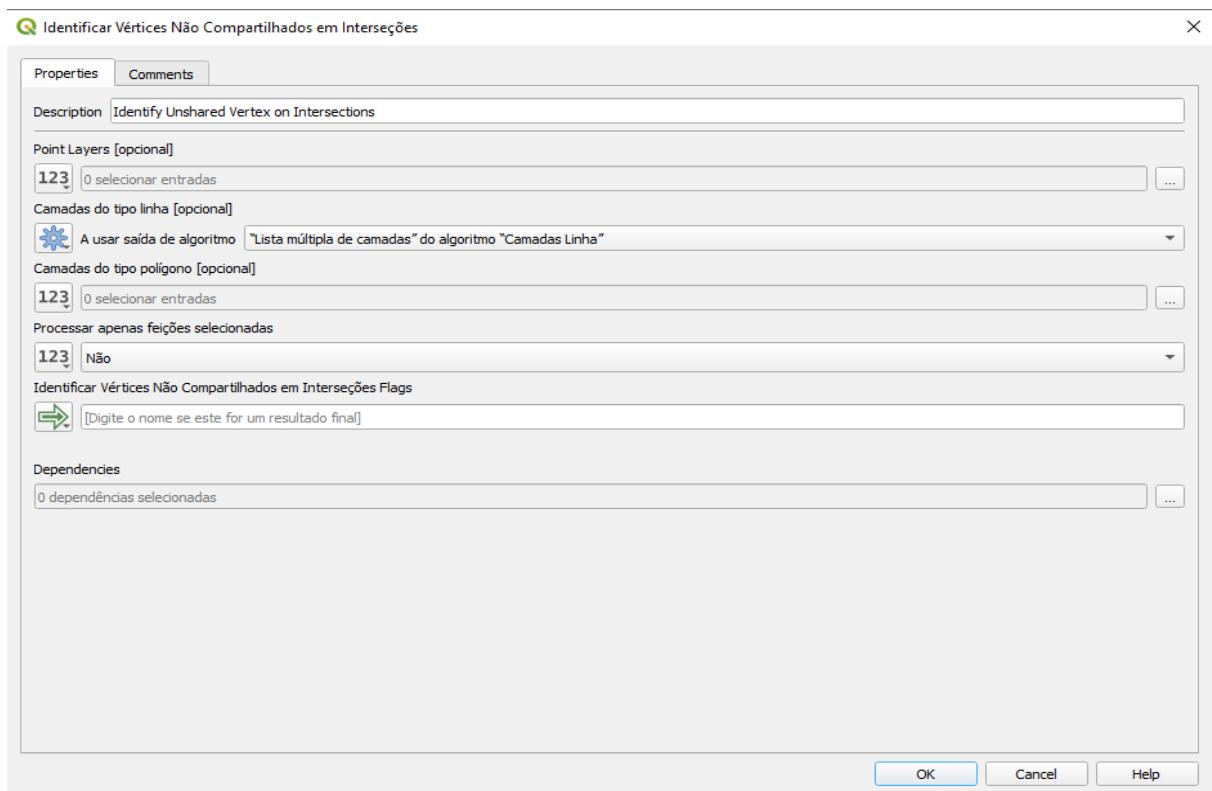


Figura 12: Extrato dos parâmetros.

Camadas de entrada:

Tipo de geometria	Camadas de entrada
Linha	infra_ferrovia_1, infra_mobilidade_urbana_1, infra_travessia_hidroviaria_1, infra_via_deslocamento_1.

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

g) Resultado do processo e exemplos de erros:

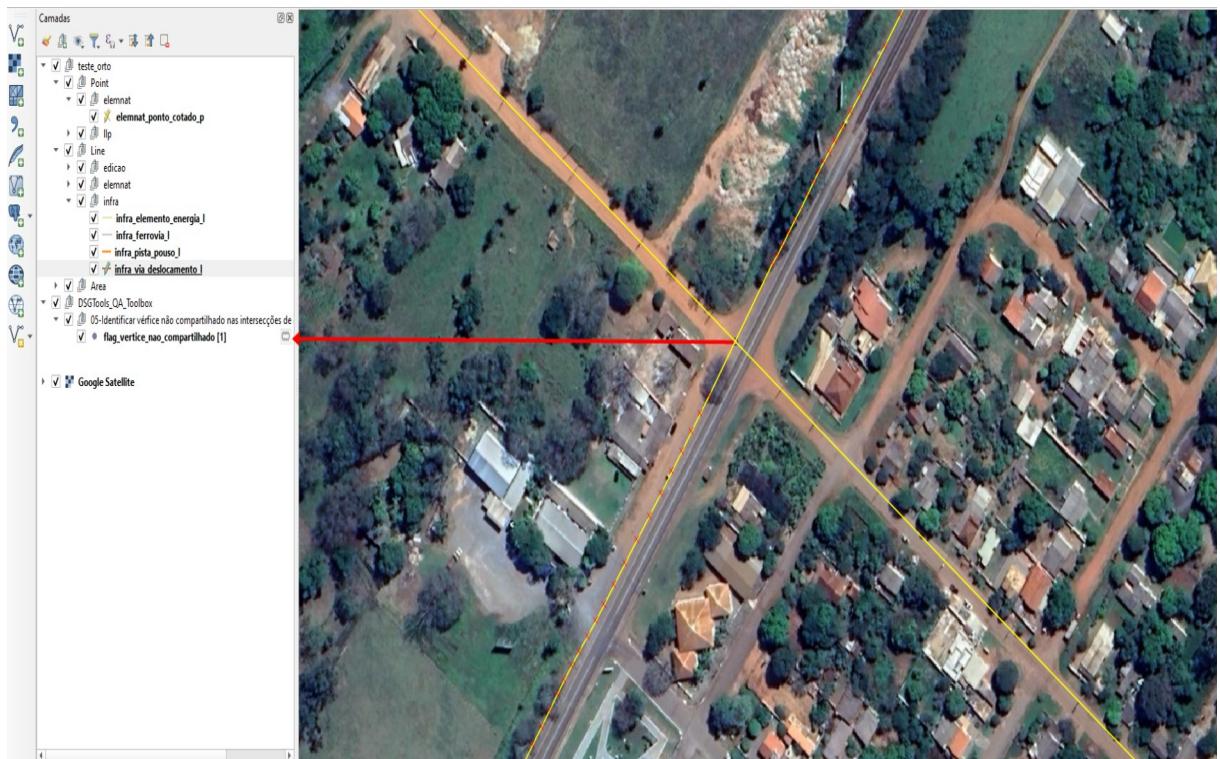


Figura 13: Exemplo de vértice não compartilhado.

5. IDENTIFICAR VÉRTICE NÃO COMPARTILHADO NOS SEGMENTOS COMPARTILHADOS

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

b) Arquivo:

identifica_vertice_nao_compartilhado_nos_segmentos_compartilhados_transportes_carta_ort.o.model3.

c) Algoritmos:

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

d) Composição do Modelo:

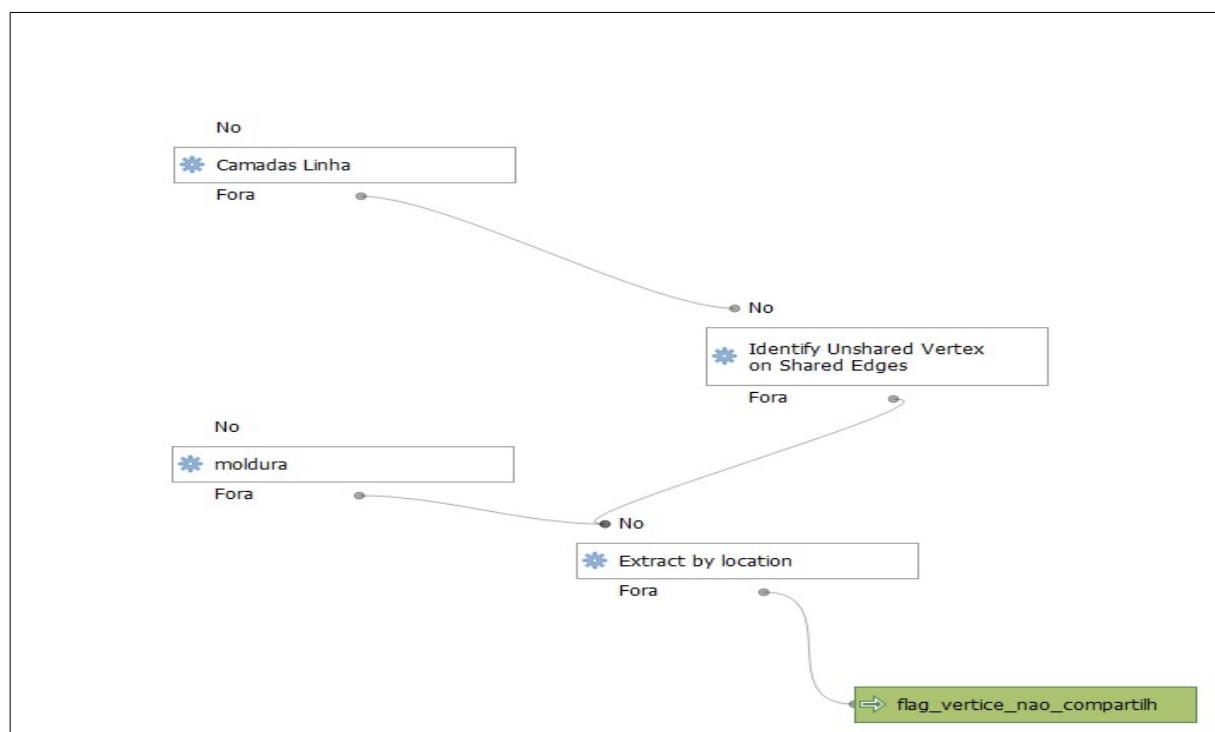


Figura 14: Modelo de identificação de vértices não compartilhados nos segmentos compartilhados.

O modelo é composto por quatro etapas:

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 de entidades 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*.

e) Parâmetros:

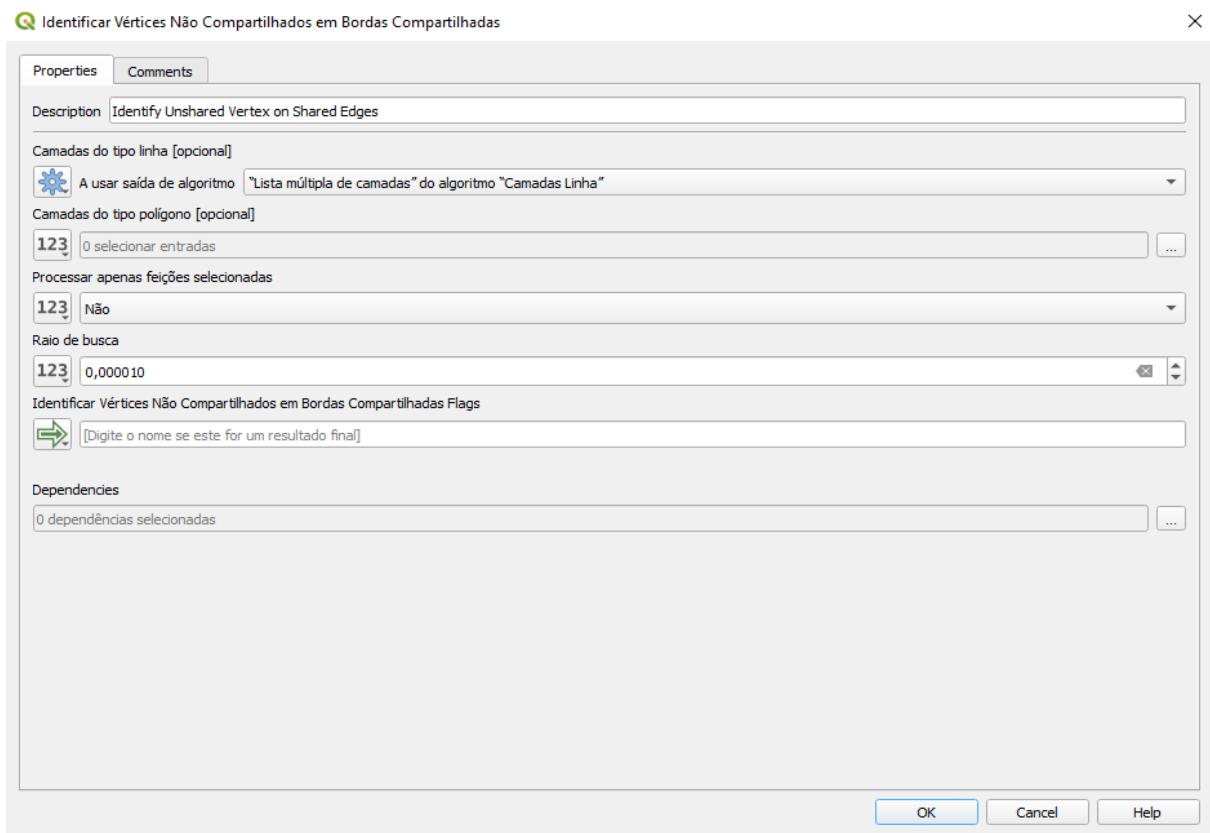


Figura 15: Extrato dos parâmetros.

Camadas de entrada:

Tipo de geometria	Camadas de entrada
Linha	infra_ferrovia_l, infra_mobilidade_urbana_l, infra_travessia_hidroviaria_l, infra_via_deslocamento_l.

f) Nome da camada de *flags*:

Flag_vertice_nao_compartilhado_em_seg_compartilhado.

g) Resultado do Processo e exemplos de erros:

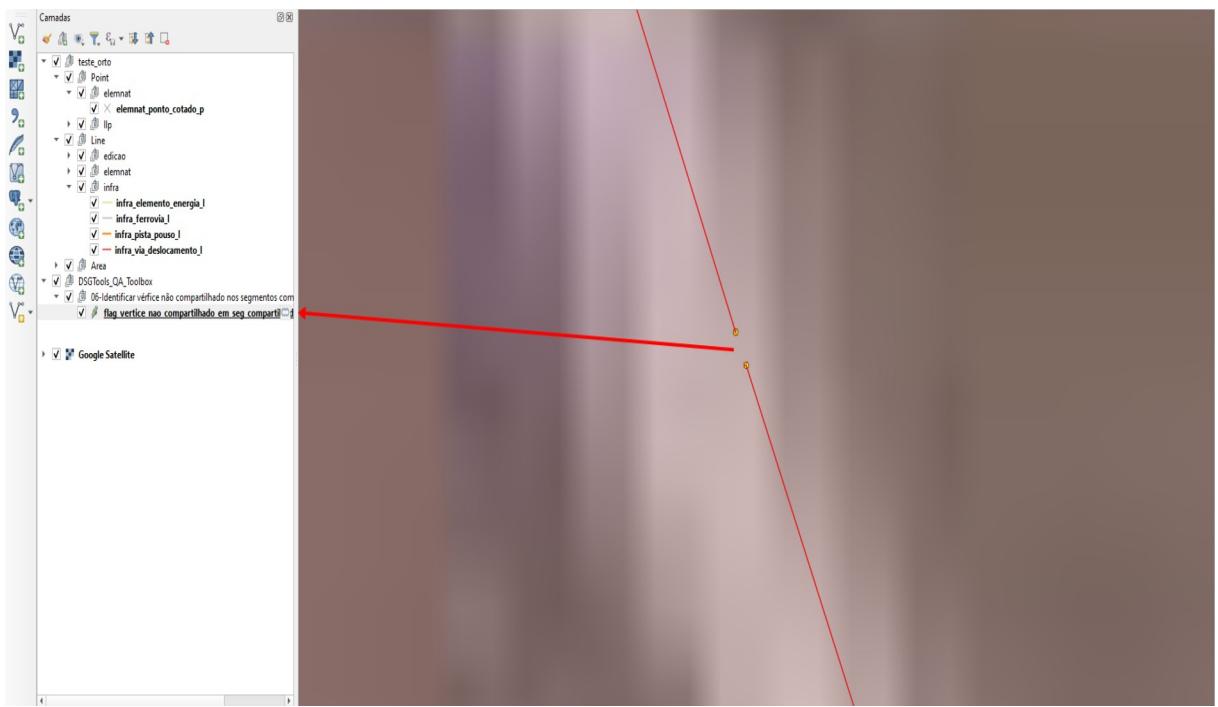


Figura 16: Exemplo de vértice não compartilhado nos segmentos compartilhados.

6. IDENTIFICAR VÉRTICE PRÓXIMO DE ARESTA

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

b) Arquivo: identifica_vertice_proximo_de_aresta_transportes_carta_orto.model3.

c) Algoritmo:

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

d) Composição do Modelo:

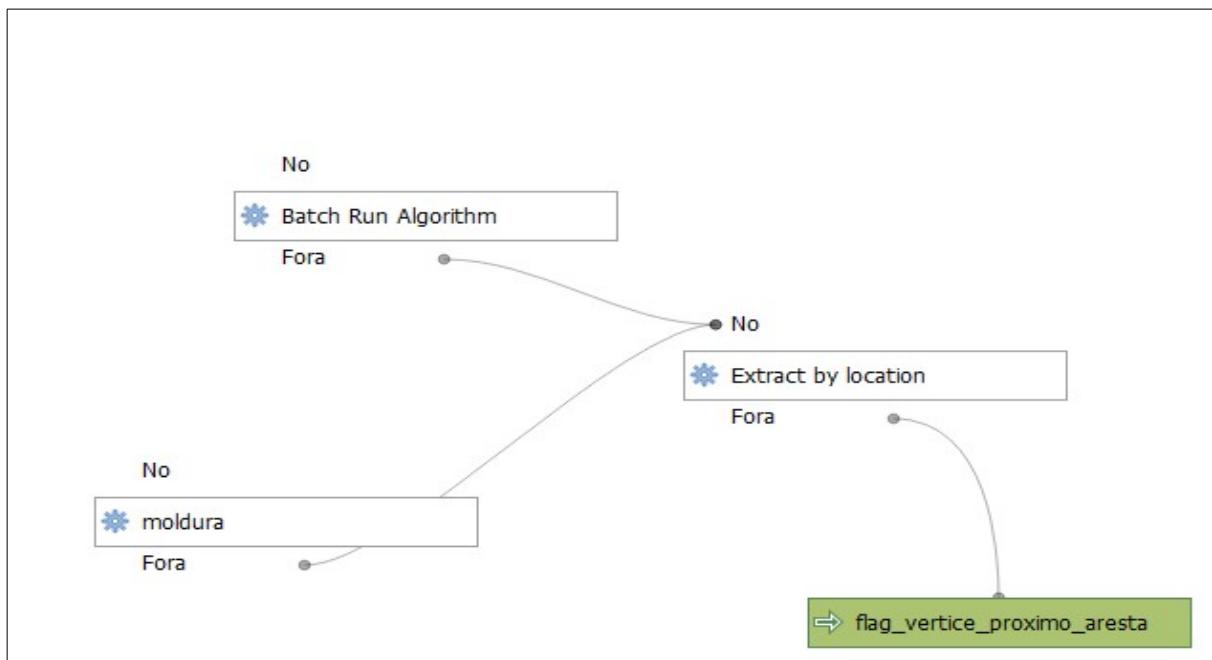


Figura 17: Modelo de identificação de vértices próximo a aresta.

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 de entidades usando o algoritmo *stringcsvtofirstlayerwithelementsalgorithm*.

ExtractByLocation: Extrai os vértices próximos de arestas que estão dentro da moldura usando o algoritmo *extractbylocation*.

e) Parâmetros:

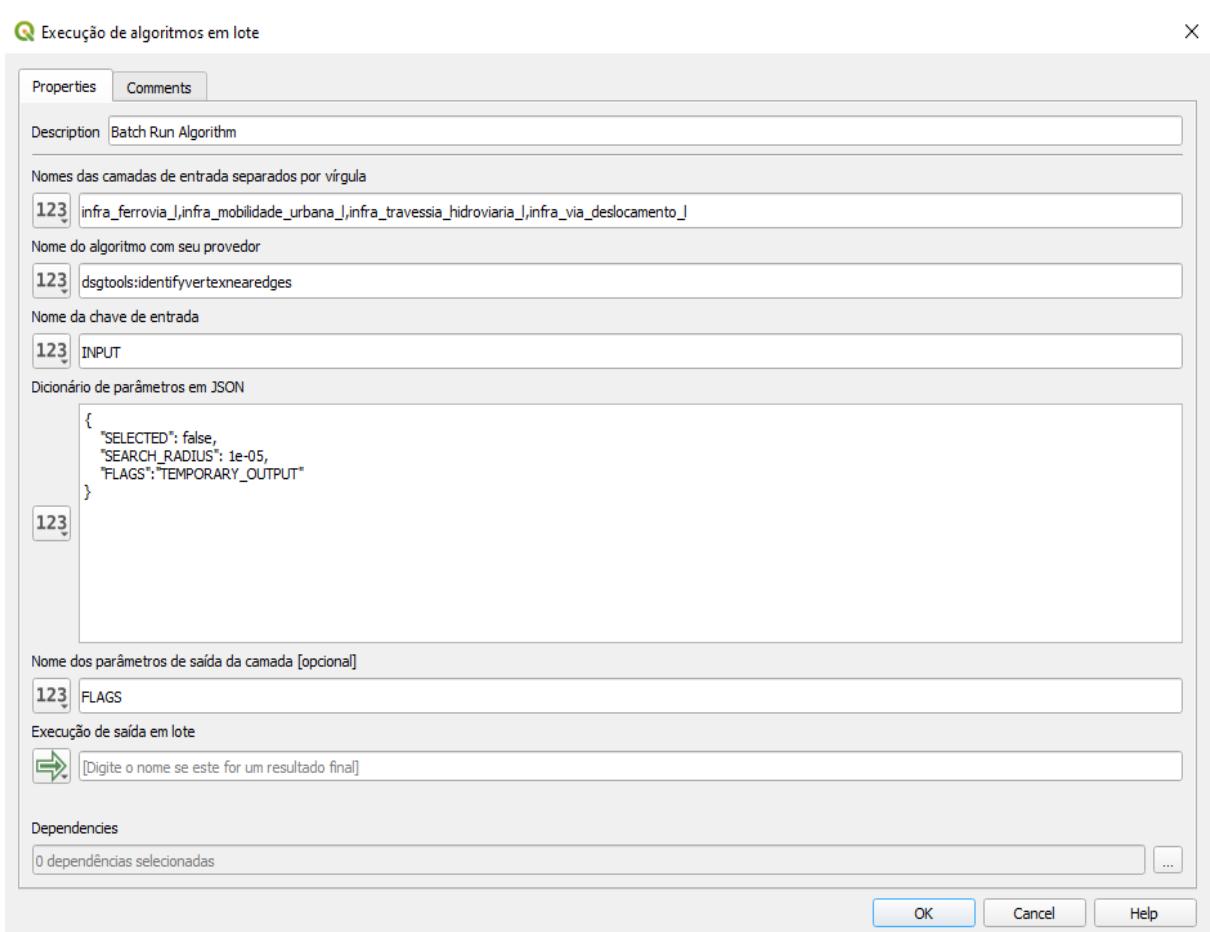


Figura 18: Extrato dos parâmetros.

Camadas de entrada:

Tipo de geometria	Camadas de entrada
Linha	infra_ferrovia_l, infra_mobilidade_urbana_l, infra_travessia_hidroviaria_l, infra_via_deslocamento_l.

Parâmetros em JSON:

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

"SELECTED": false - Esta propriedade é uma chave que tem um valor booleano associado a ela. O valor "false" indica que o item não está selecionado ou marcado.

"SEARCH_RADIUS": 1e-05 - Esta propriedade é uma chave que tem um valor numérico associado a ela. O valor "1e-05" é uma notação científica que representa o número decimal 0.00001. Esse valor pode ser usado como um raio de busca para encontrar objetos dentro de uma determinada distância.

"FLAGS": "TEMPORARY_OUTPUT" - Esta propriedade é uma chave que tem uma *string* associada a ela. A *string* "TEMPORARY_OUTPUT" pode ser um sinalizador ou uma opção que é usada para indicar que um determinado resultado ou saída é temporário e não deve ser armazenado permanentemente.

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

g) Resultado do processo e exemplo de erros:



Figura 19: 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_transportes_carta_orto.model3.

c) Algoritmo:

dsgtools:identifygeometrieswithlargevertexdensityalgorithm;
dsgtools:batchrunalgorithm;

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

d) Composição do Modelo:

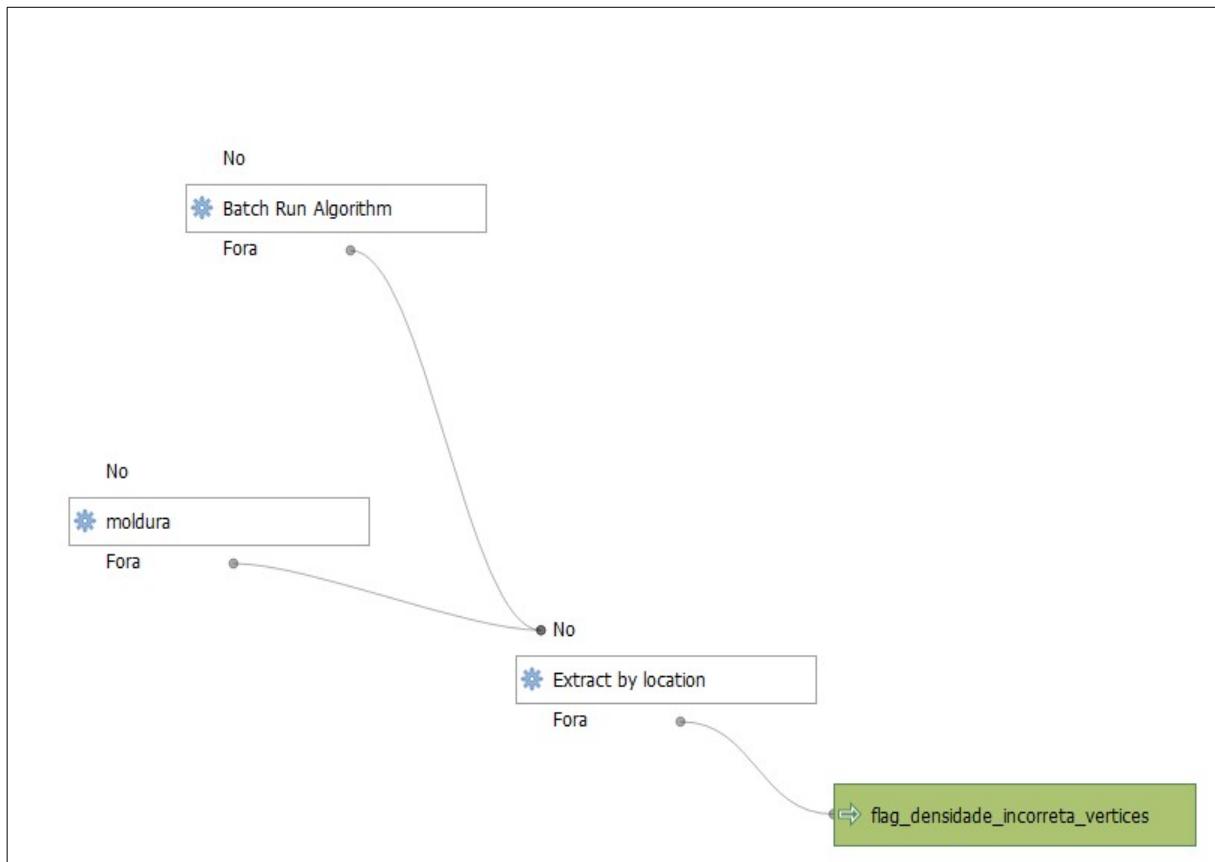


Figura 20: Modelo de identificação de geometrias com densidade incorreta de vértices.

O modelo é composto por três etapas:

BatchRunAlgorithm: Executa o algoritmo *identifygeometrieswithlargevertexdensityalgorithm* com os parâmetros descritos conforme na 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:

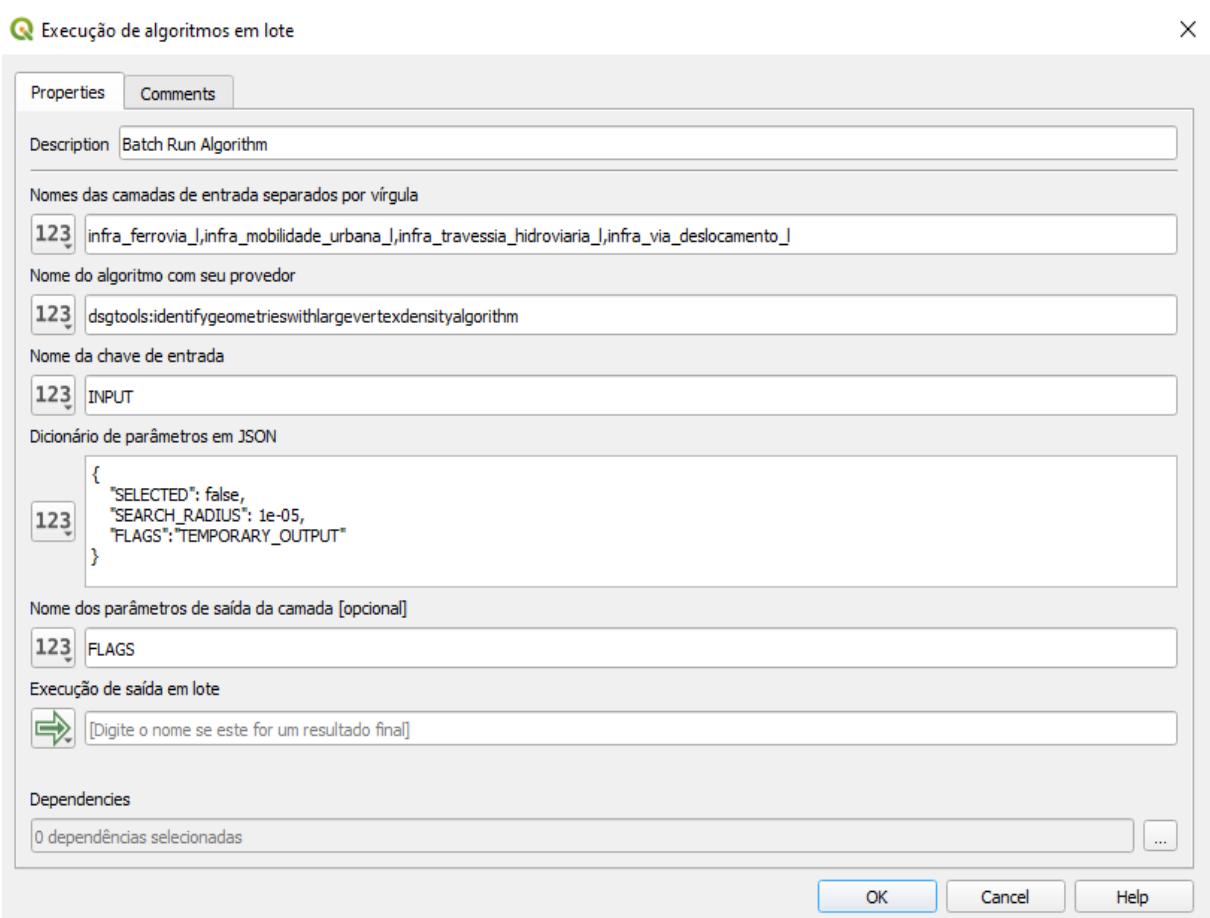


Figura 21: Extrato dos parâmetros.

Camadas de entrada:

Tipo de geometria	Camadas de entrada
Linha	infra_ferrovia_1, infra_mobilidade_urbana_1, infra_travessia_hidroviaria_1, infra_via_deslocamento_1.

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:

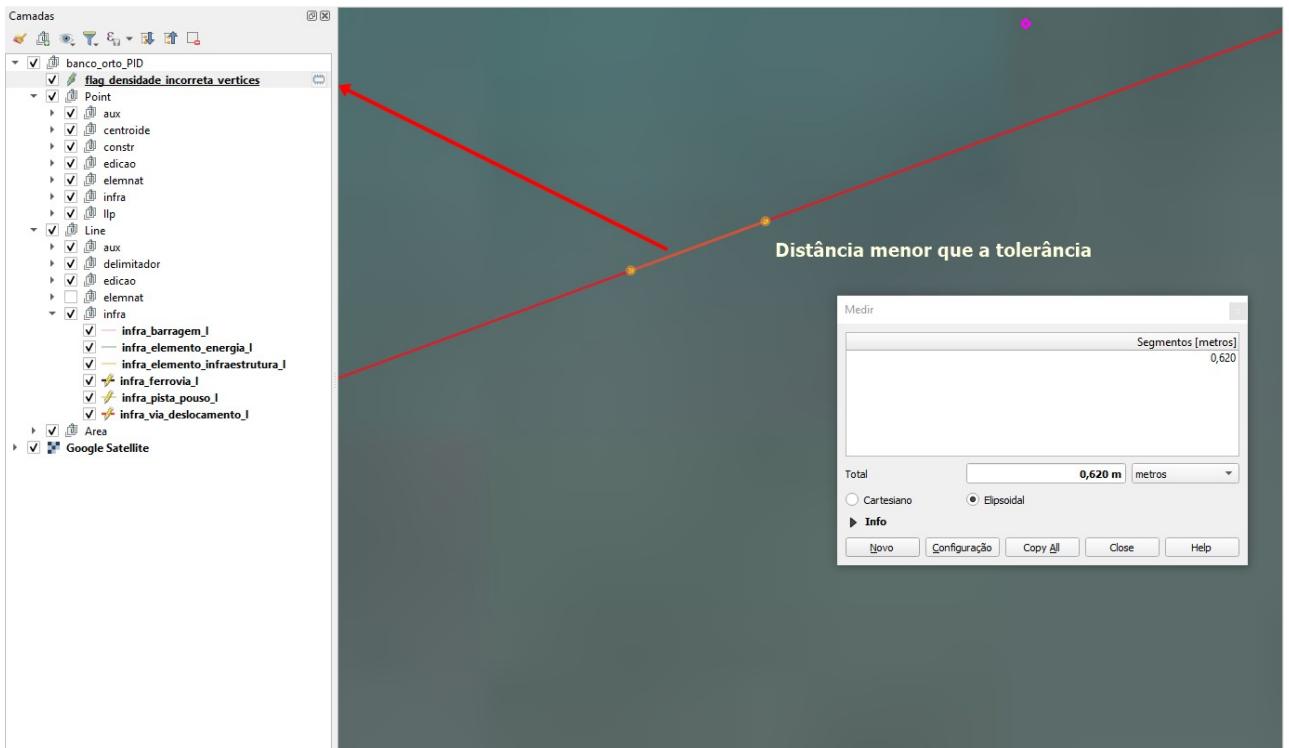


Figura 22: Exemplo de geometria com densidade incorreta de vértices.

8. IDENTIFICAR ÂNGULOS PEQUENOS

a) Descrição: Este algoritmo identifica ângulos pequenos em uma lista de camadas de linha.

b) Arquivo: identifica_angulos_pequenos_transportes_carta_ortho.model3.

c) Algoritmo:

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

d) Composição do Modelo:

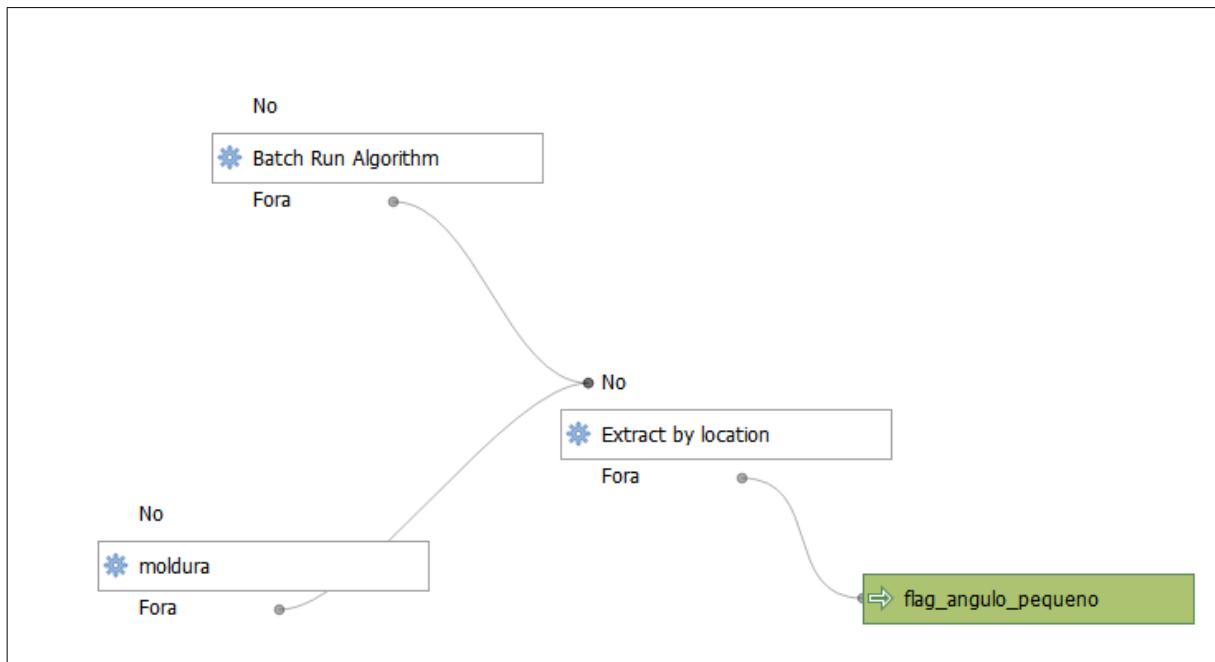


Figura 23: Modelo de identificação de ângulos pequenos.

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:

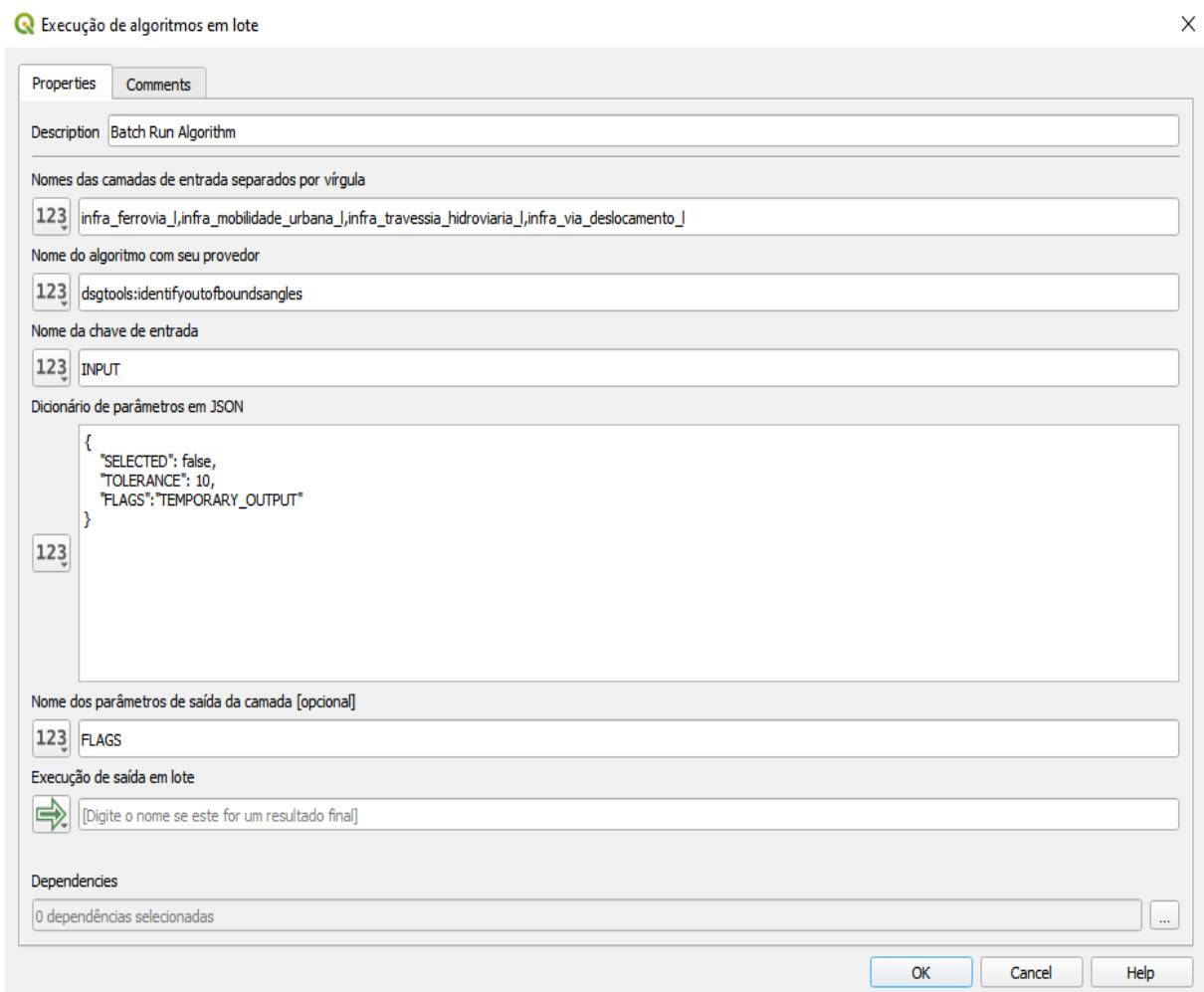


Figura 24: Extrato dos parâmetros.

Camadas de entrada:

Tipo de geometria	Camadas de entrada
Linha	infra_ferrovia_1, infra_mobilidade_urbana_1, infra_travessia_hidroviaria_1, infra_via_deslocamento_1.

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, por exemplo, `infra_via_deslocamento_1` muito acentuadas. 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 erro:

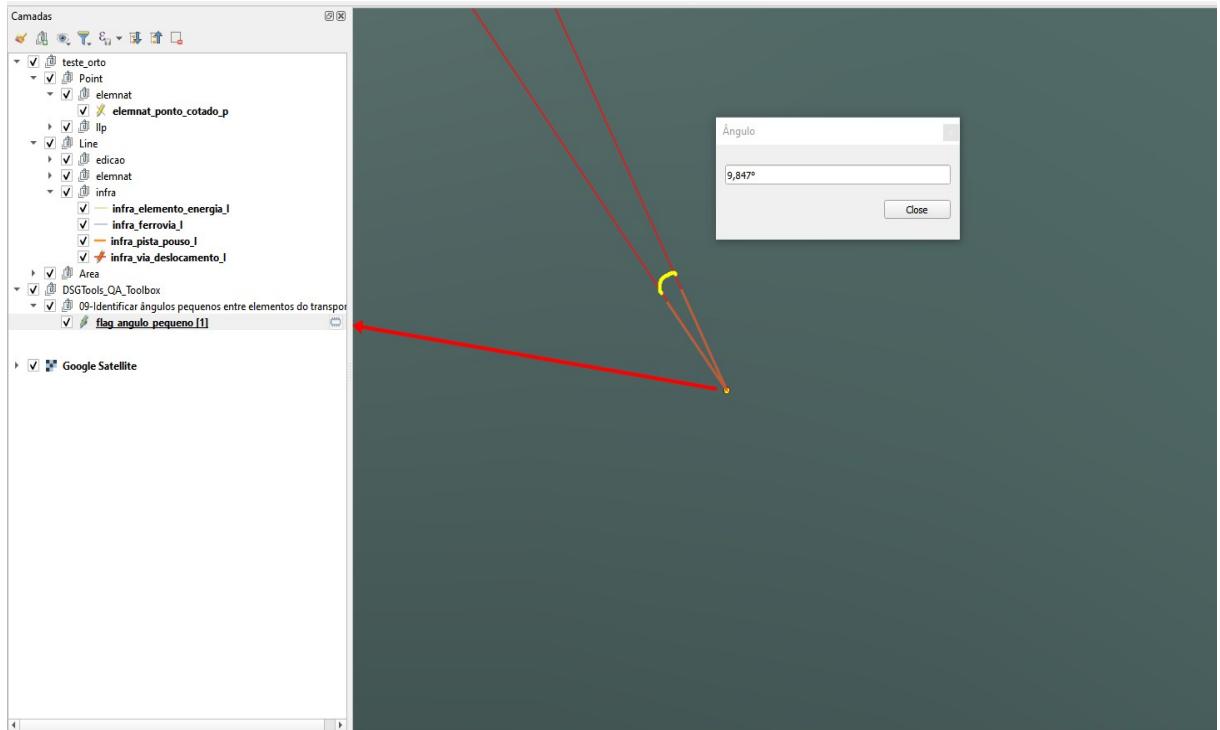


Figura 25: Exemplo de ângulo pequeno.

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:

identifica_angulos_pequenos_entre_camadas_transportes_carta_ortho.model3.

c) Algoritmo:

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

d) Composição do Modelo:

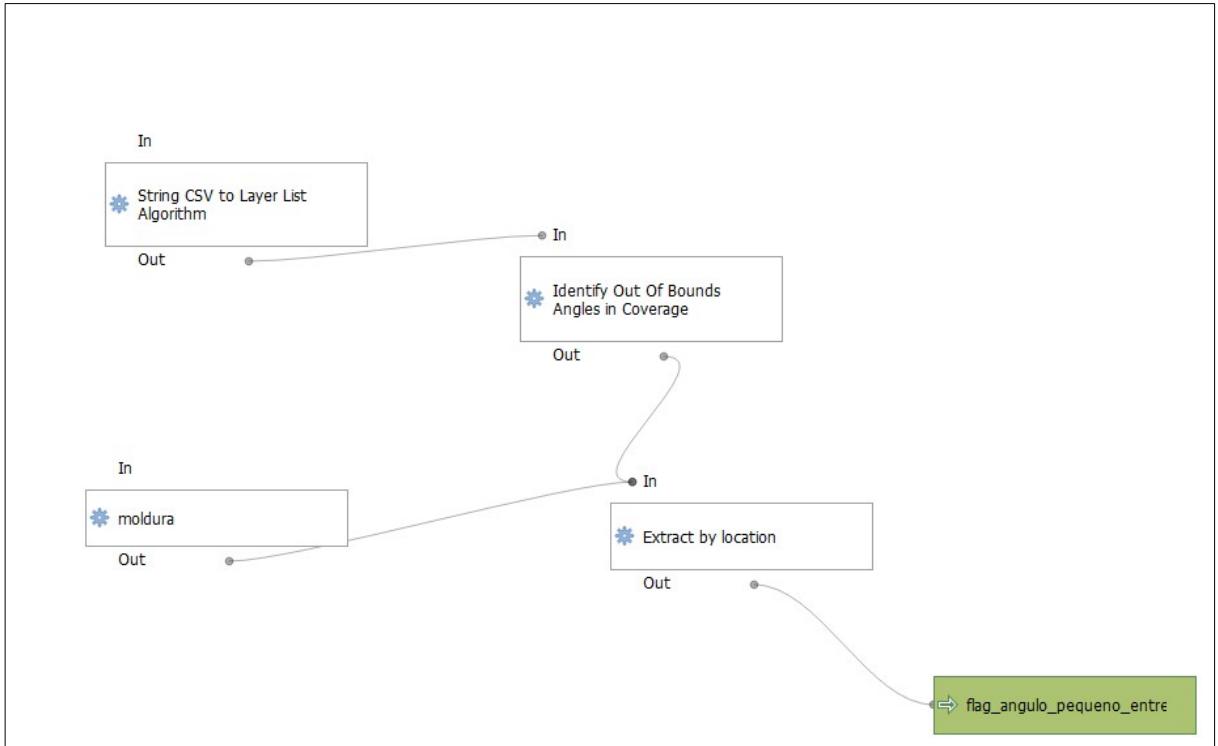


Figura 26: Modelo de identificação de ângulos pequenos entre camadas

O modelo é composto por quatro etapas:

StringCsvToLayerListAlgorithm: Converte uma *string* CSV em uma lista de camadas, usando o algoritmo *stringcsvtolayerlistalgorithm*.

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:

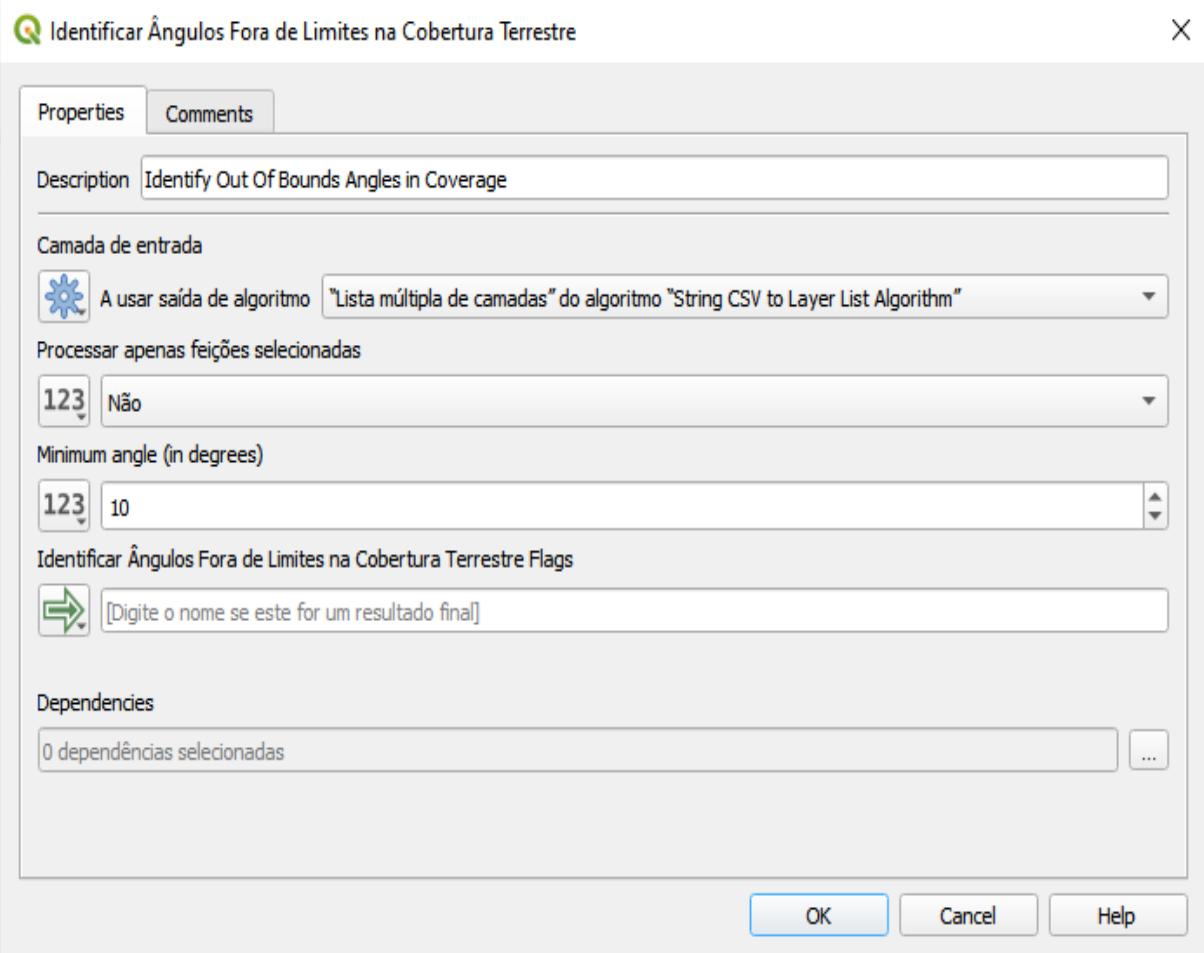


Figura 27: Extrato dos parâmetros.

Camadas de entrada:

Tipo de geometria	Camadas de entrada
Linha	infra_ferrovia_1, infra_mobilidade_urbana_1, infra_travessia_hidroviaria_1, infra_via_deslocamento_1.

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 infra_via_deslocamento_1 muito acentuadas. 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 erro:

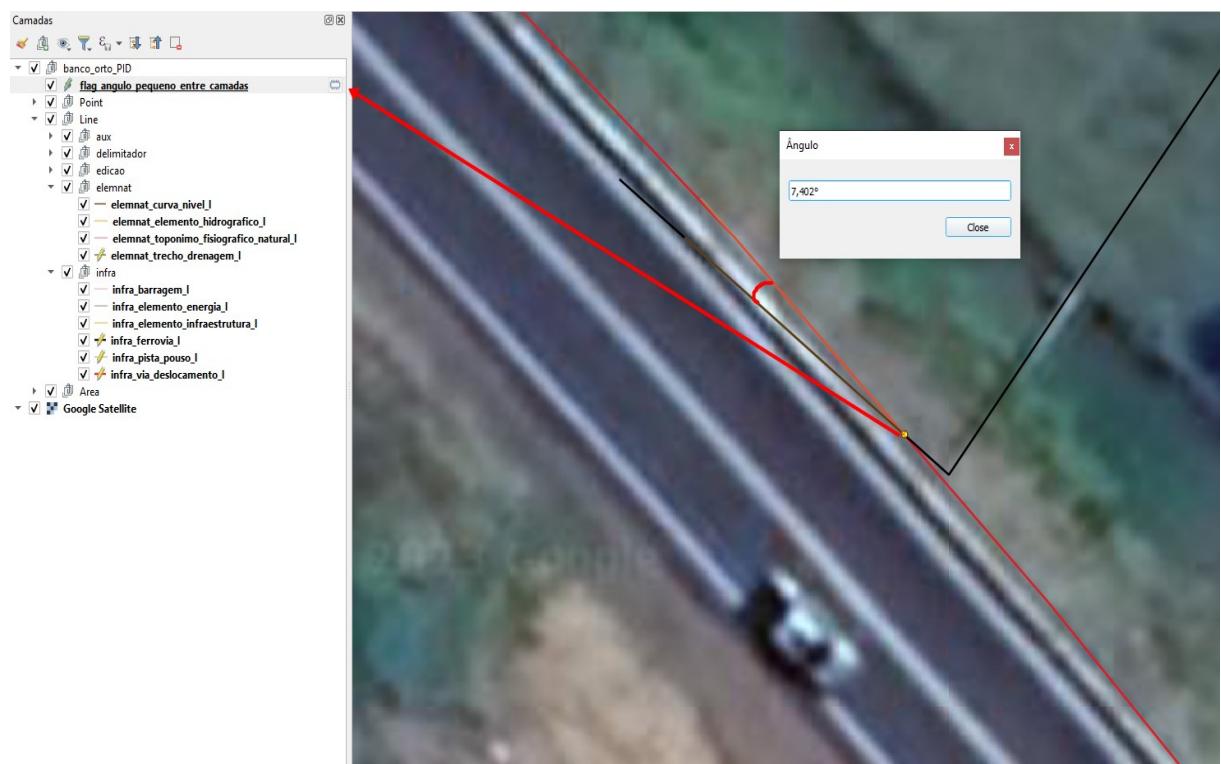


Figura 28: Exemplo de geometria com ângulo menor que a tolerância entre camadas distintas.

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_transportes_carta_ortho.model3.

c) Algoritmo:

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

d) Composição do Modelo:

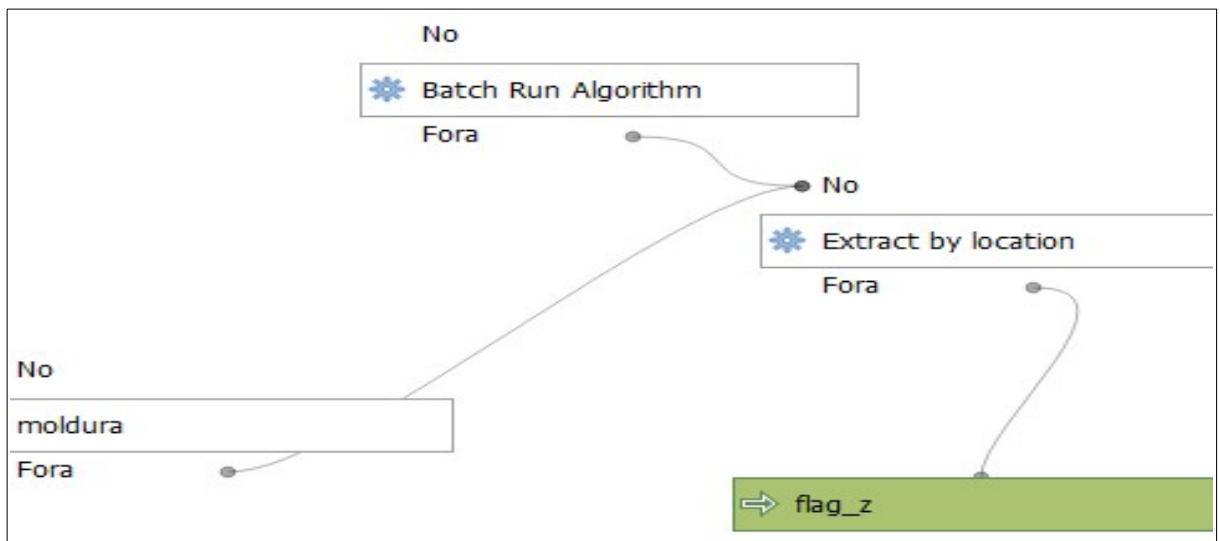


Figura 29: Modelo de identificação de ângulos em Z.

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:

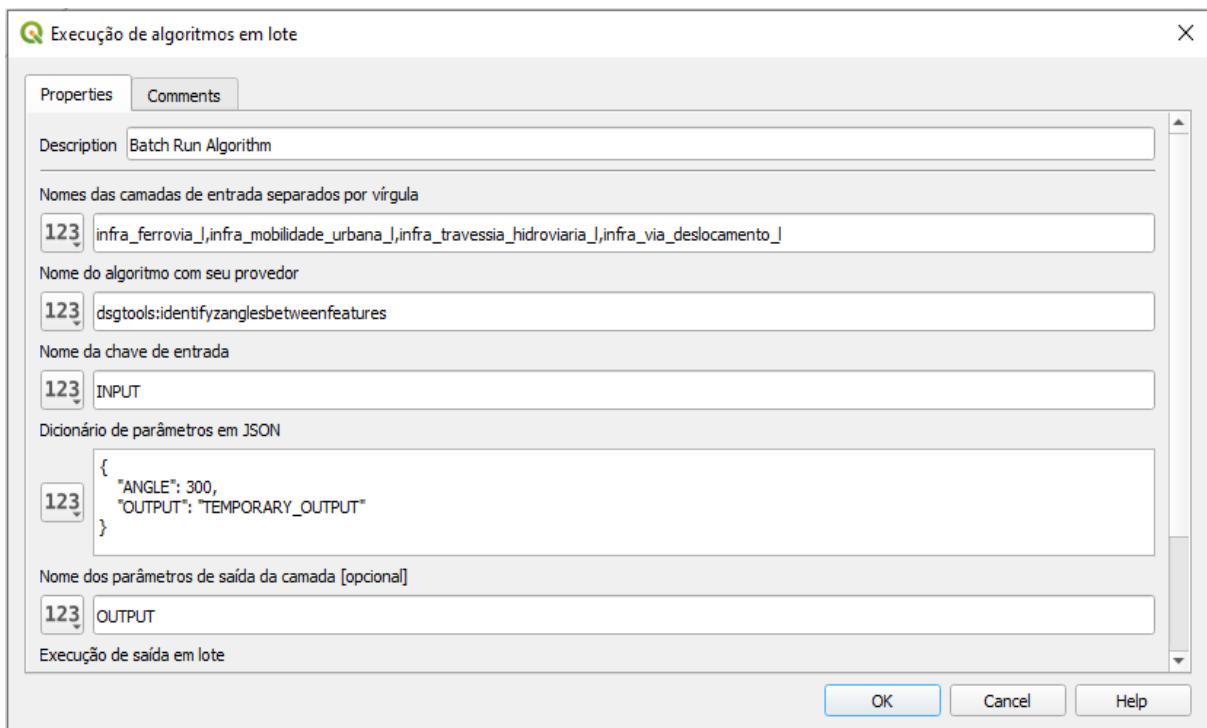


Figura 30: Extrato dos parâmetros.

Camadas de entrada:

Tipo de geometria	Camadas de entrada
Linha	infra_ferrovia_1, infra_mobilidade_urbana_1, infra_travessia_hidroviaria_1, infra_via_deslocamento_1.

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:

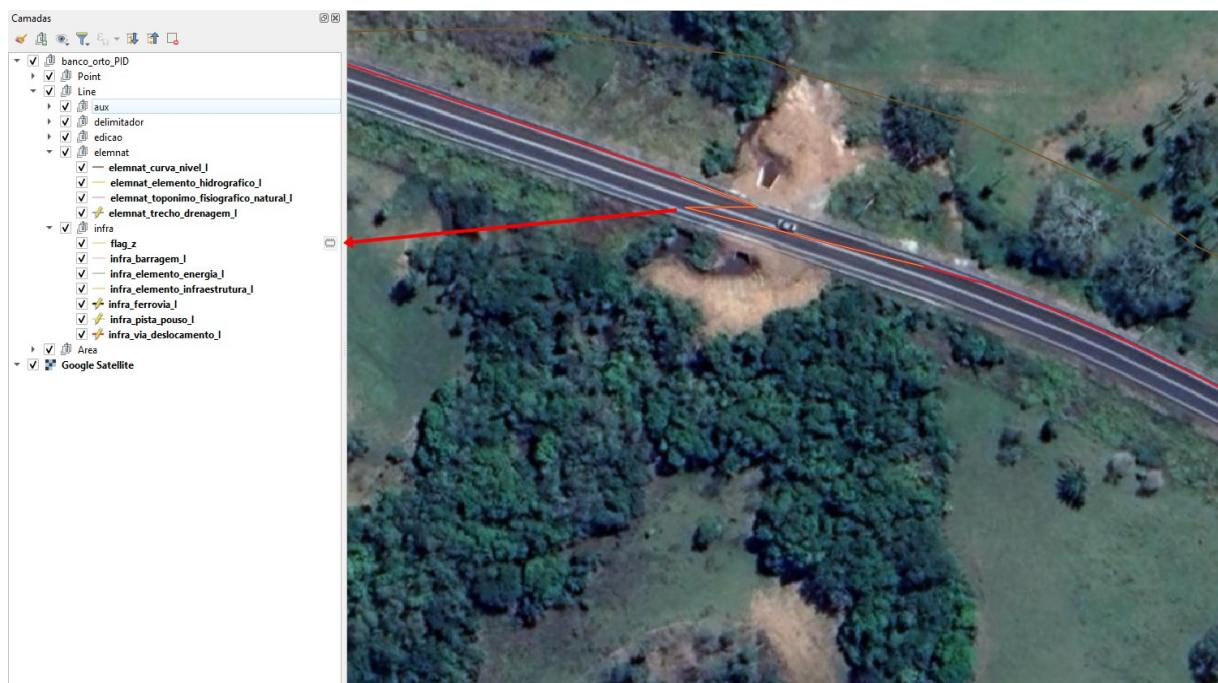


Figura 31: Exemplo de geometria com ângulos em formato Z.

11. IDENTIFICAR *OVERTLAPS* DENTRO DA MESMA CAMADA

a) Descrição: Este algoritmo identifica sobreposições dentro da mesma camada em uma lista de camadas de linha.

b) Arquivo: identifica_overlaps_linhas_transportes_carta_orto.model3.

c) Algoritmo:

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

d) Composição do Modelo:

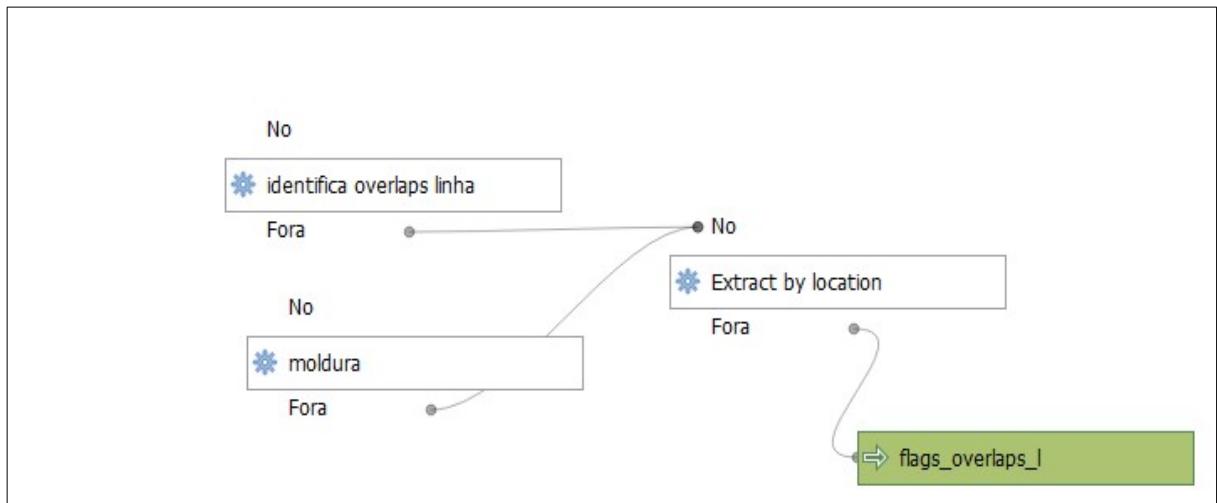


Figura 32: Modelo de identificação de sobreposições dentro da mesma camada.

O modelo é composto por três etapas:

Identifica *overlaps* linha: Executa o algoritmo *identifyoverlaps* com os parâmetros descritos na “seção e” para as camadas de linha.

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:

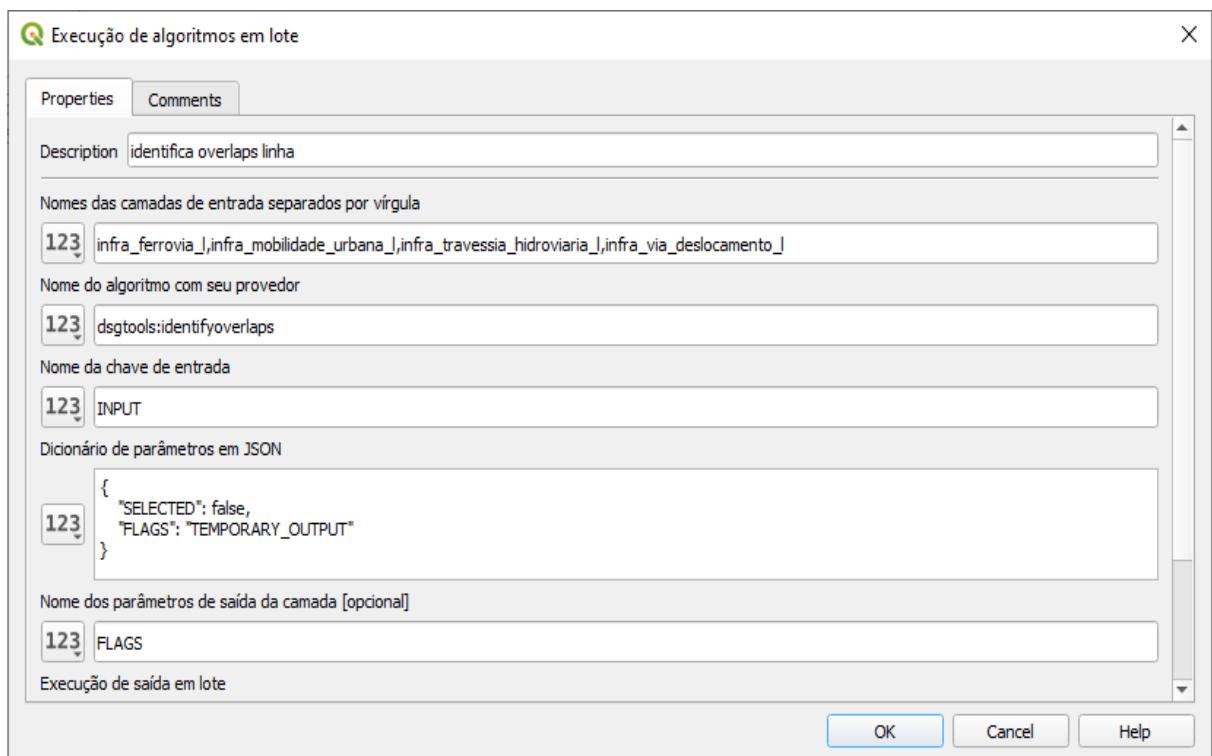


Figura 33: Extrato dos parâmetros.

Camadas de entrada:

Tipo de geometria	Camadas de entrada
Linha	infra_ferrovia_1, infra_mobilidade_urbana_1, infra_travessia_hidroviaria_1, infra_via_deslocamento_1.

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* encontrados durante a validação.

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

g) Resultado do processo e exemplo de erro:

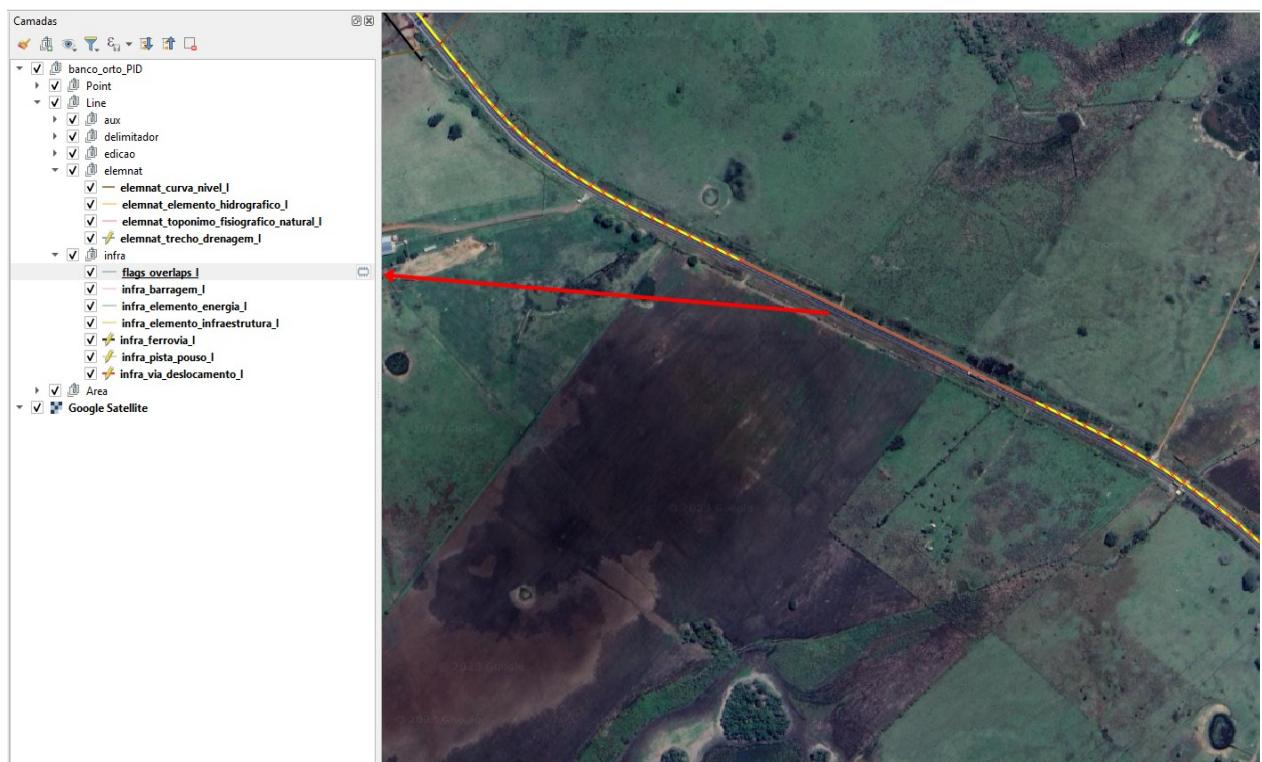


Figura 34: Exemplo de sobreposição.

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.

b) Arquivo: identifica_undershoot_moldura_conexao_linhas_transportes.model3.

c) Algoritmo:

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

d) Composição do Modelo:

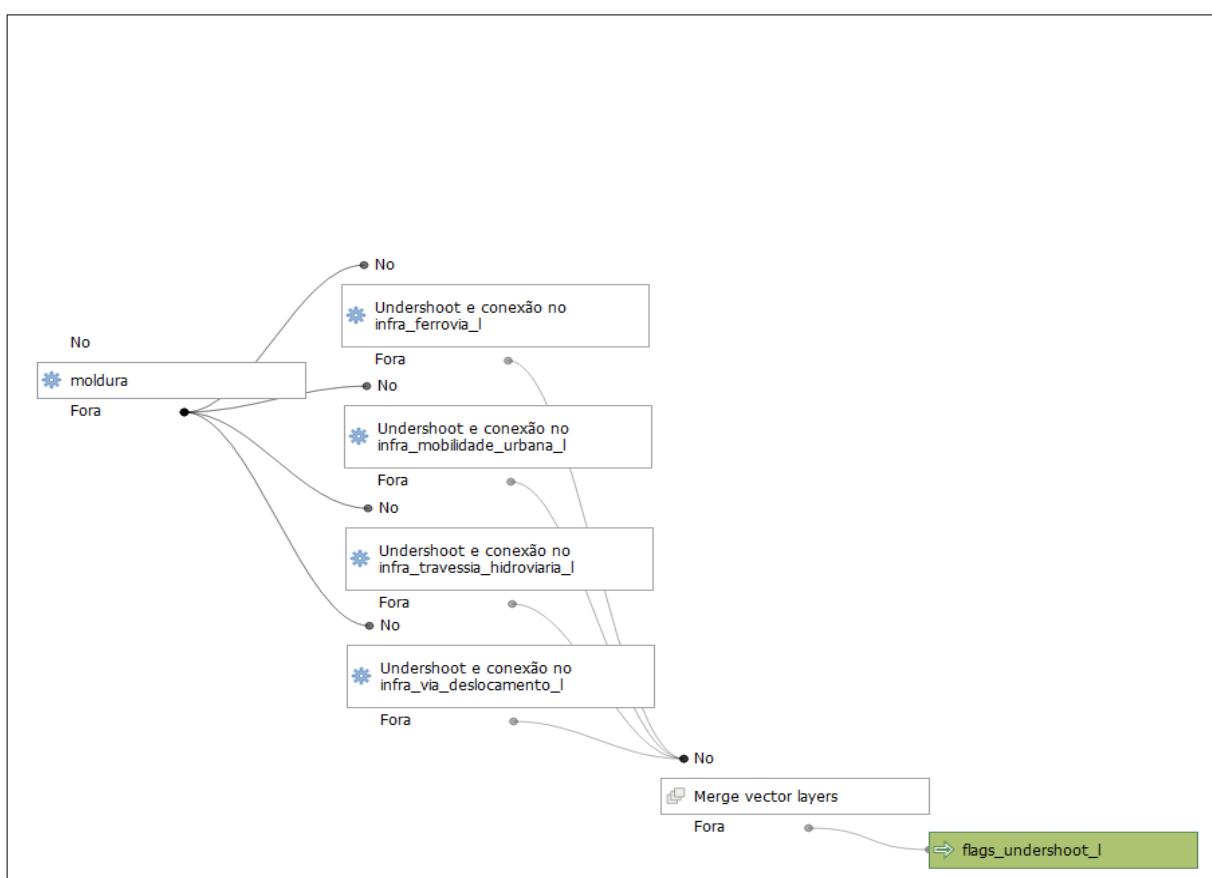


Figura 35: Modelo de identificação de pontas soltas.

O modelo é composto por três etapas:

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

Undershoot e conexão: Executa os algoritmos *identifydangles* para as camadas de linha 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:

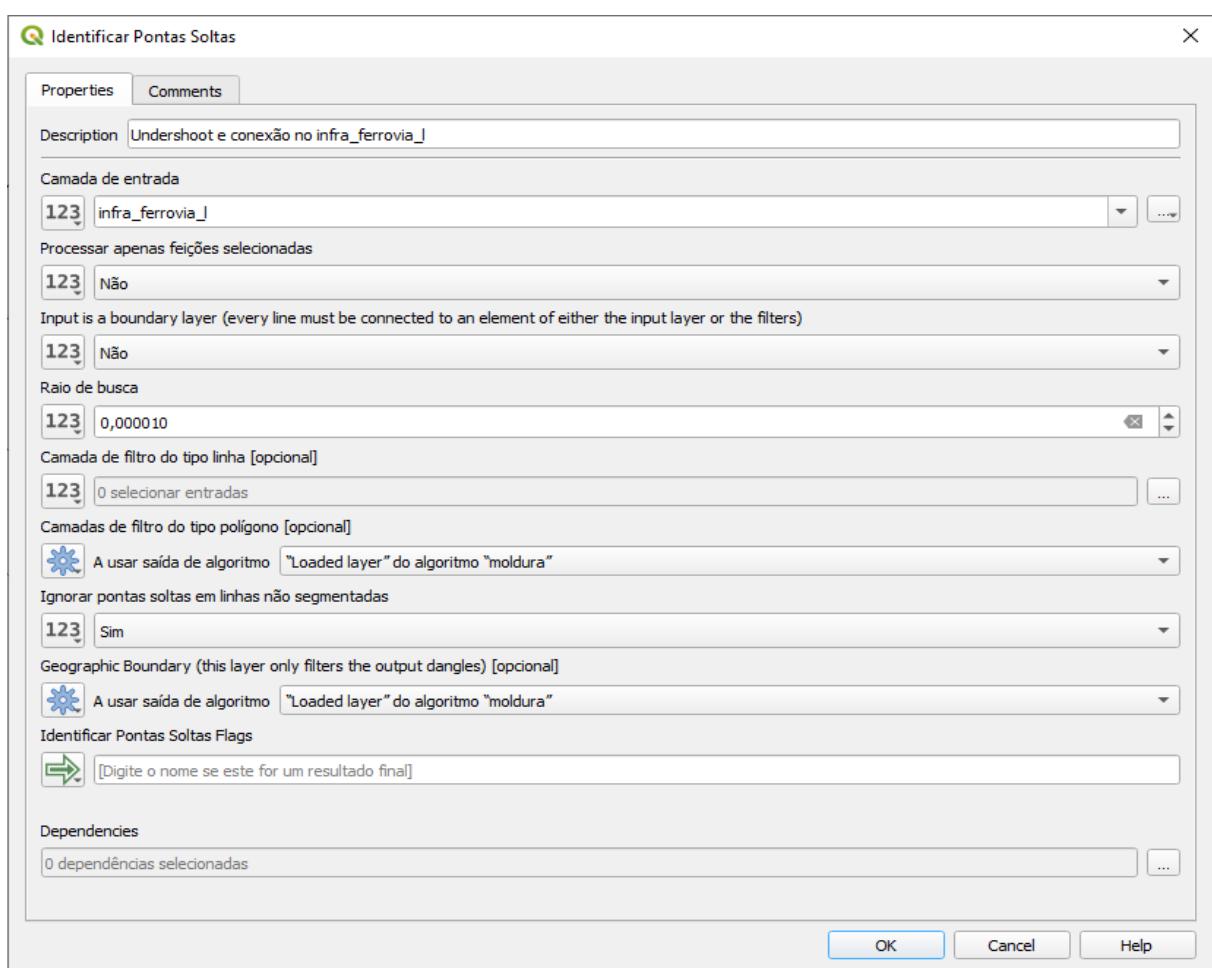


Figura 36: Extrato dos parâmetros.

Camadas de entrada:

Tipo de geometria	Camadas de entrada
Linha	infra_ferrovia_1, infra_mobilidade_urbana_1, infra_travessia_hidroviaria_1, infra_via_deslocamento_1.

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 e polígonos que se cruzam nas intersecções. No modelo, o valor padrão do raio de busca é definido como 0,00001 unidades do sistema de coordenadas utilizado no processamento. 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*: *flags_undershoot_l*.

g) Resultado do processo e exemplo de erros:

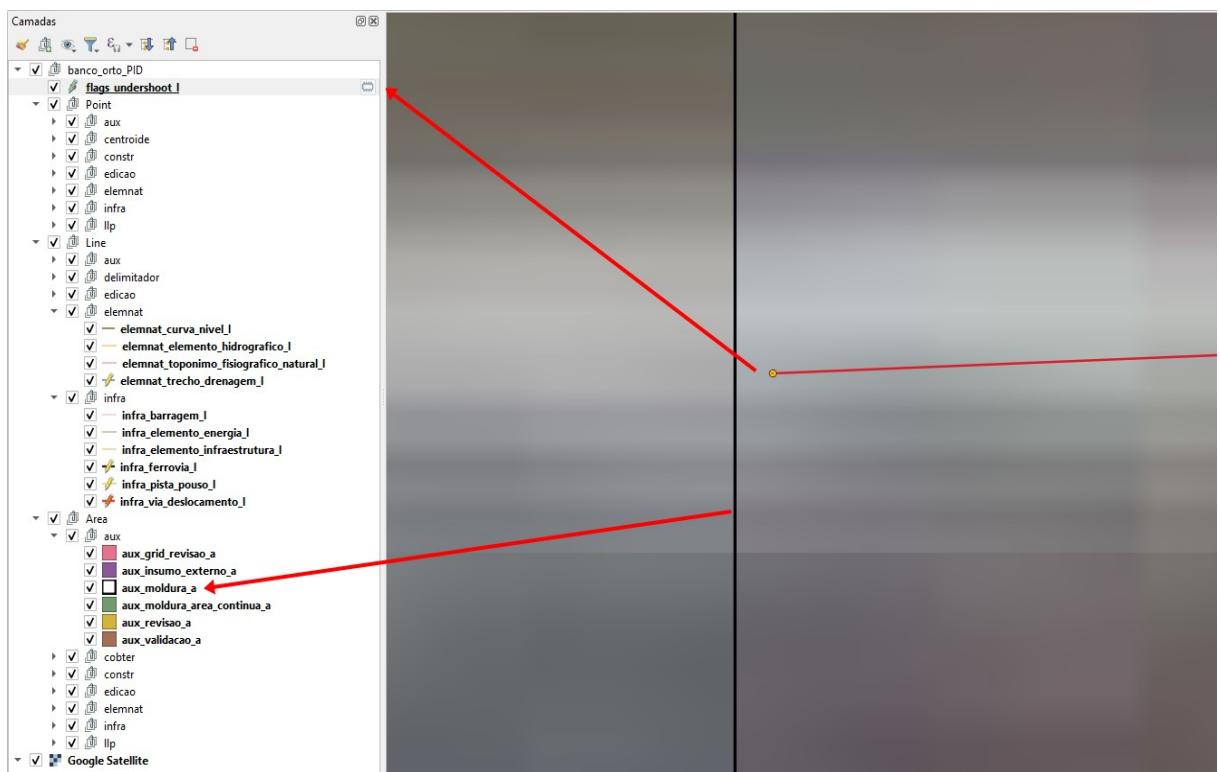


Figura 37: Exemplo de ponta solta.

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_transportes_carta_orto.
model3.`

c) Algoritmo:

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

d) Composição do Modelo:

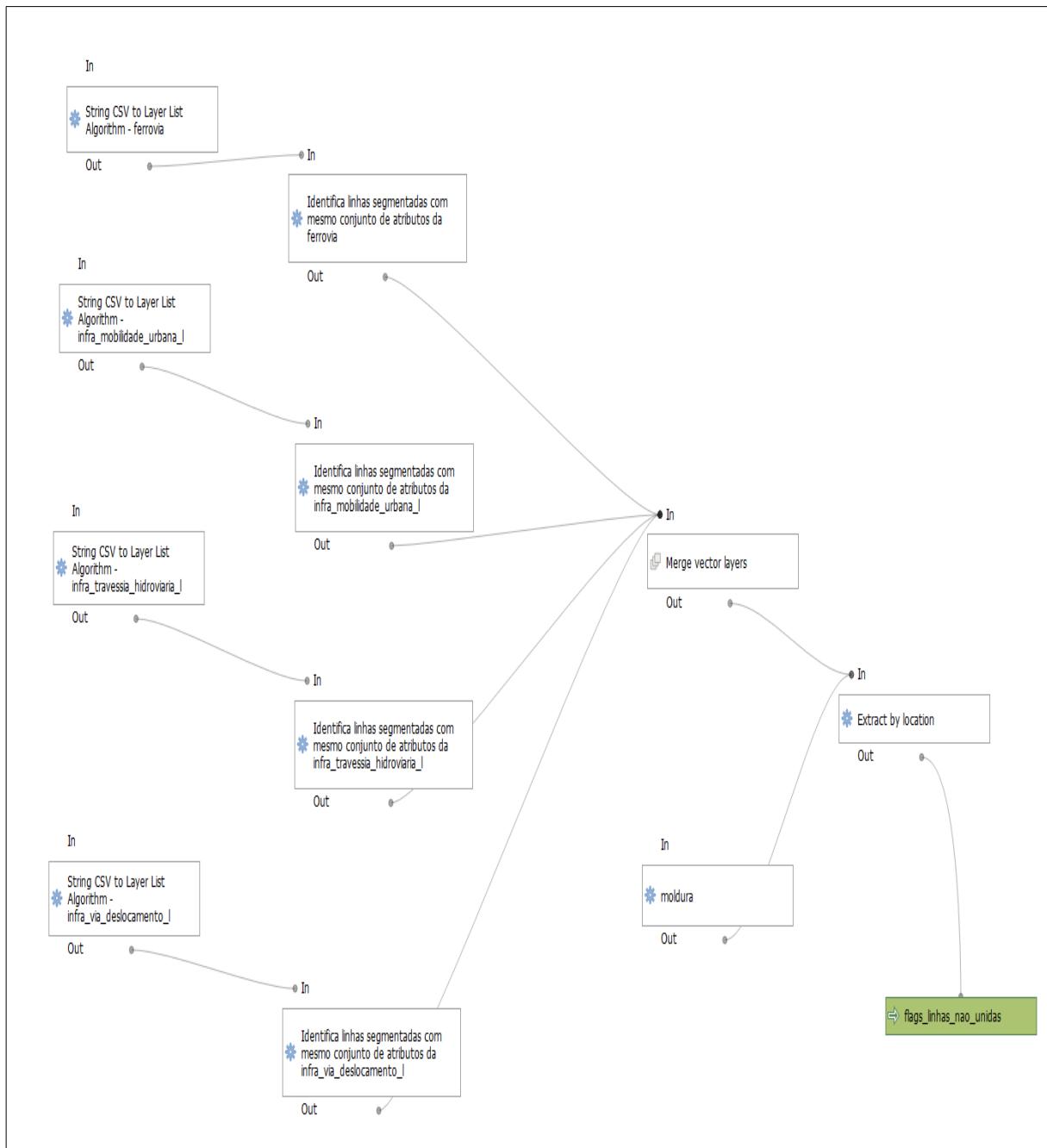


Figura 38: Modelo de identificação de linhas segmentadas com o mesmo conjunto de atributos.

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 na “seção e” utilizando o algoritmo *identifyunmergedlineswithsameattributeset*.

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:

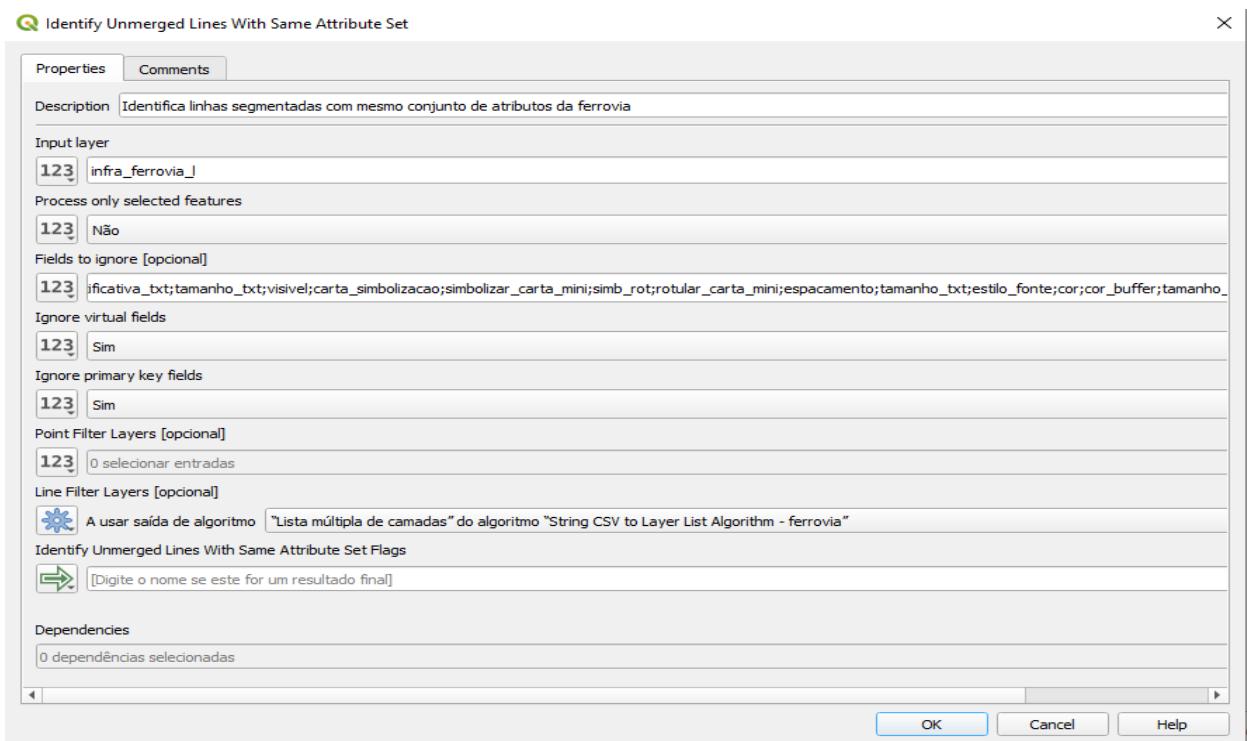


Figura 39: Extrato dos parâmetros.

Camadas de entrada:

Tipo de geometria	Camadas de entrada
Linha	infra_ferrovia_l, infra_mobilidade_urbana_l, infra_travessia_hidroviaria_l, infra_via_deslocamento_l.

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 negra de atributos (*ATTRIBUTE_BLACKLIST*) que não serão considerados na comparação das feições, a fim de evitar falsos positivos.

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

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 "**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:

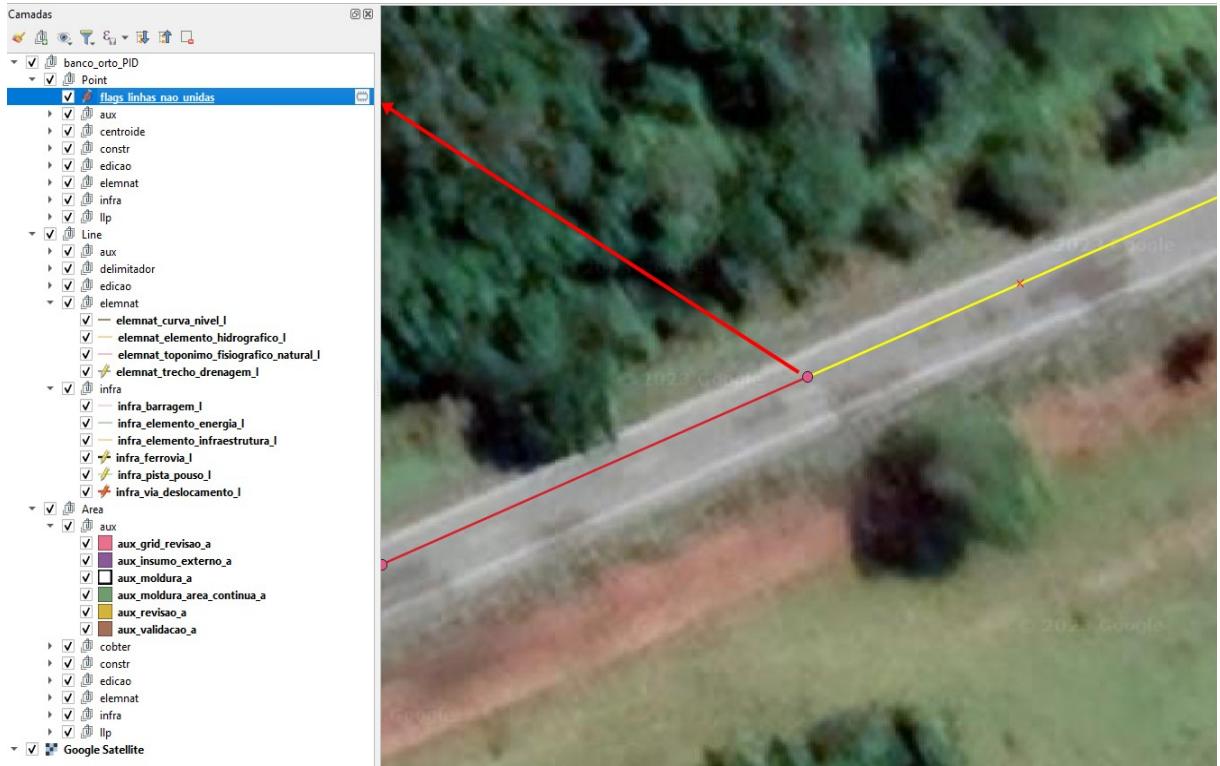


Figura 40: Exemplo de linhas segmentadas sem motivo.

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_transportes.model3`.

c) Algoritmo:

```
dsgtools:stringcsvtofirstlayerwithelementsalgorithm;
dsgtools:identifydangles;
```

native:mergevectorlayers.

d) Composição do Modelo:

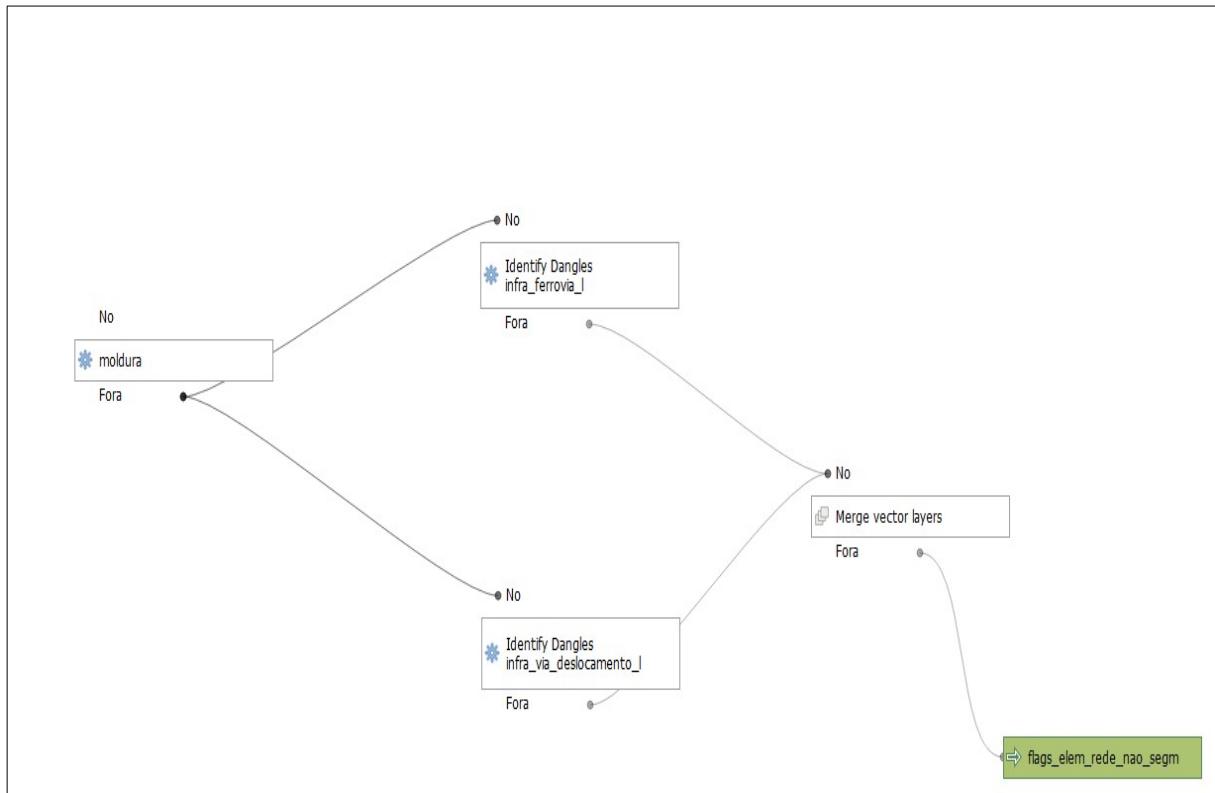


Figura 41: Modelo de identificação de linhas não segmentadas nas intersecções.

O modelo é composto por três etapas:

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

IdentifyDangles: Identifica pontas soltas na camada de *infra_ferrovia_1* e *infra_via_deslocamento* com o algoritmo *identifydangles*.

MergeVectorLayers: Realiza uma operação de unir as camadas de saída utilizando o algoritmo *mergevectorlayers*.

e) Parâmetros:

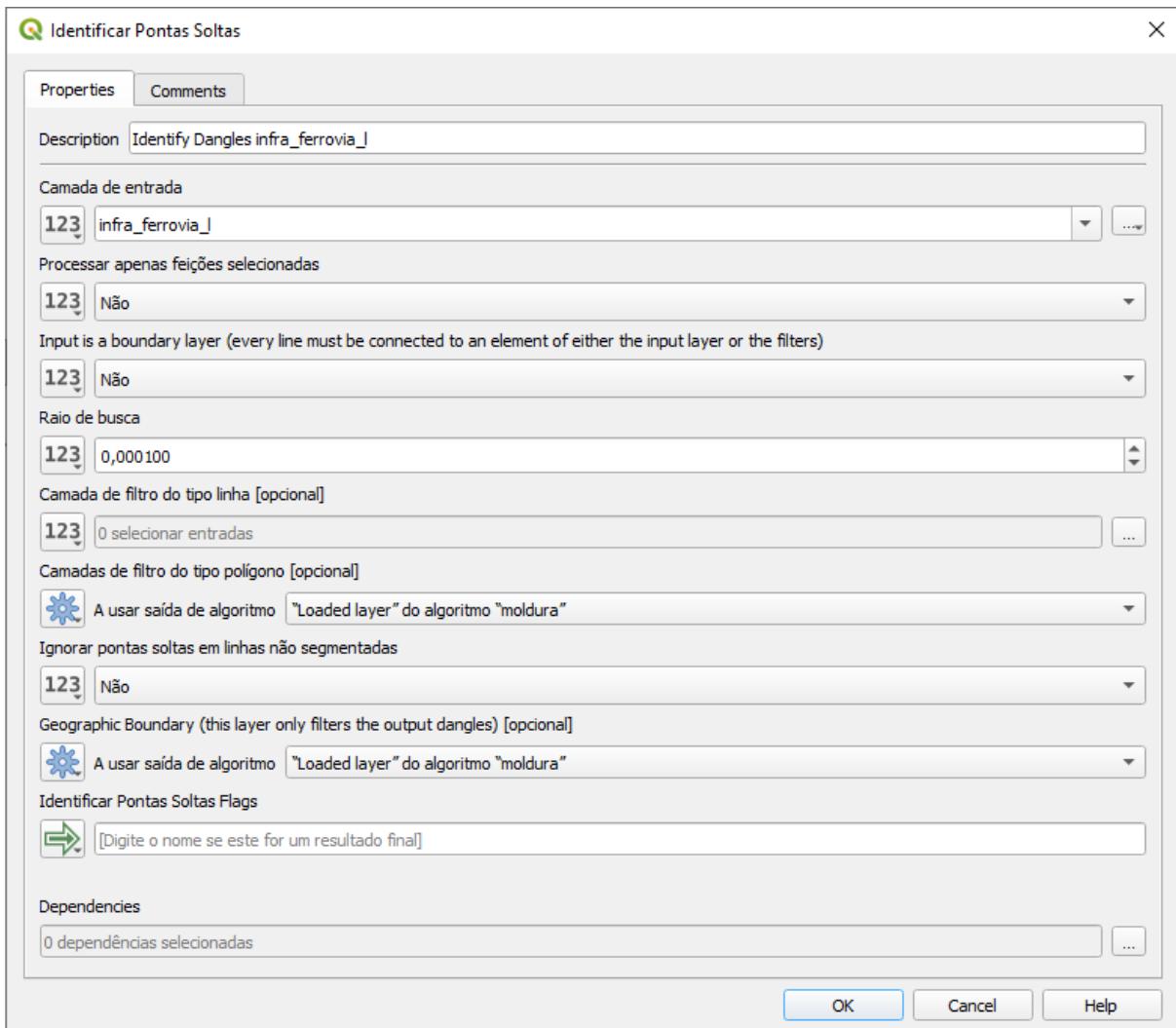


Figura 42: Extrato dos parâmetros.

Camadas de entrada:

Tipo de geometria	Camadas de entrada
Linha	infra_ferrovia_1, infra_via_deslocamento_1.

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 0,0001 unidades do sistema de coordenadas utilizado no processamento. 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 utilizado 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_elem_rede_nao_segmentados*.

g) Resultado do processo e exemplo de erros:



Figura 43: Exemplo de linha não segmentada nas intersecções.

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 transporte que se encontram desconectados da rede.

b) Arquivo: `identificar_elementos_pequenos_na_rede_transportes.model3`.

c) Algoritmo:

```
dsgtools:stringcsvtofirstlayerwithelementsalgorithm;  
dsgtools:identifysmallfirstorderdangles;  
native:mergevectorlayers;  
native:extractbylocation.
```

d) Composição do Modelo:

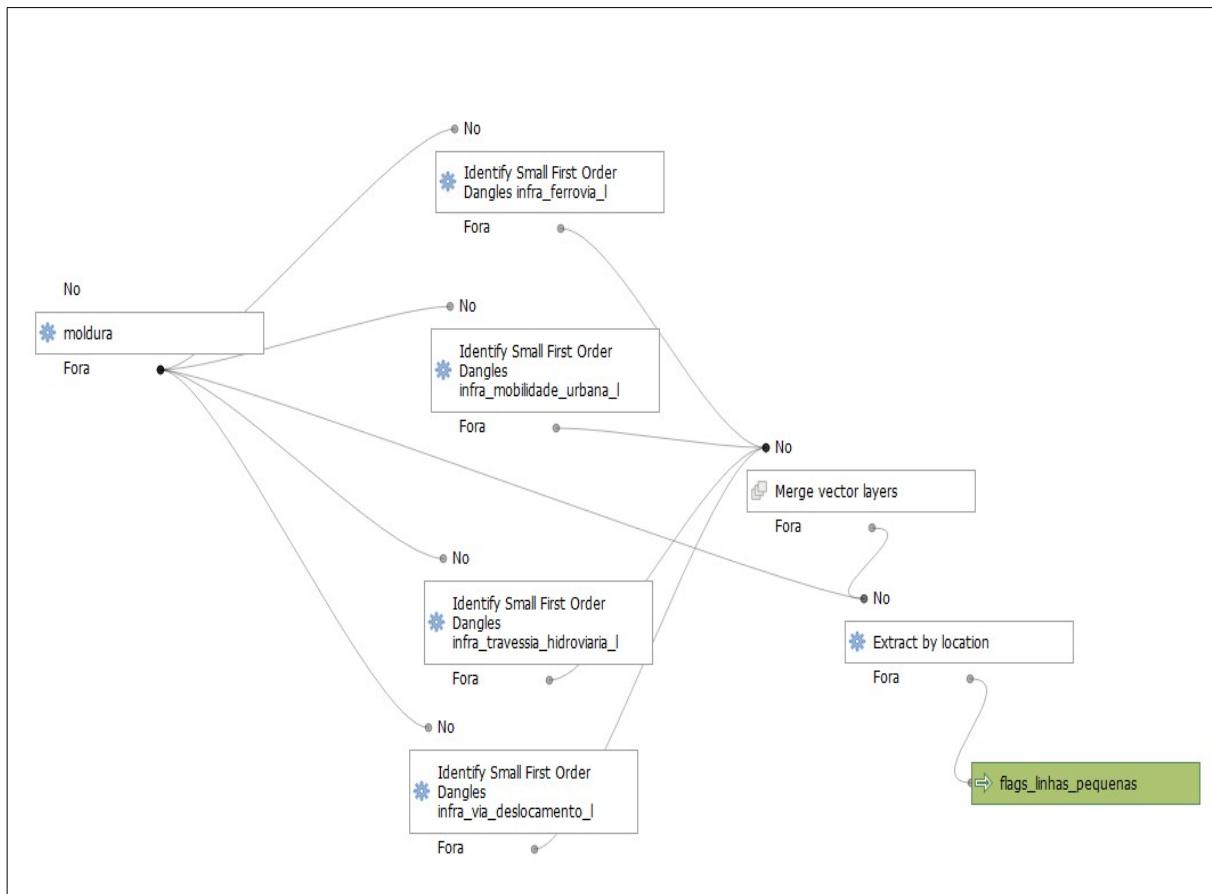


Figura 44: Modelo de identificação de elementos pequenos.

O modelo é composto por quatro etapas:

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

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

MergeVectorLayers: 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:

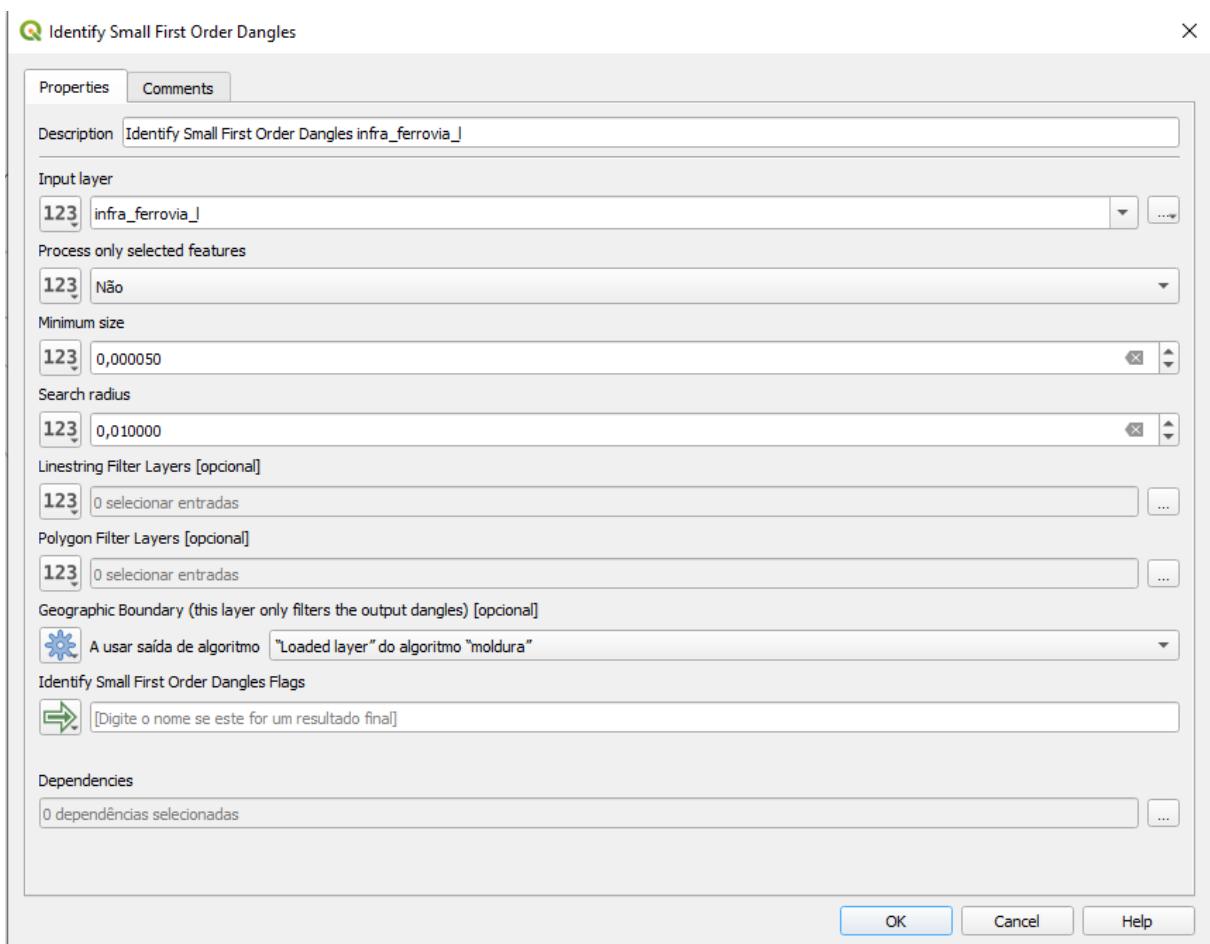


Figura 45: Extrato dos parâmetros.

O algoritmo "*Identify Small First Order Dangles*" é uma ferramenta utilizada para identificar feições que não estão conectadas na 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.

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 apenas *dangles* pequenos, ou seja, aqueles que têm um comprimento mínimo de 0,00005 e estão localizados dentro de um raio de pesquisa definido pelo parâmetro "**SEARCH_RADIUS**" de 0,01 permitindo identificar possíveis problemas na rede.

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.

Camadas de entrada:

Tipo de geometria	Camadas de entrada
Linha	infra_ferrovia_1, infra_mobilidade_urbana_1, infra_travessia_hidroviaria_1, infra_via_deslocamento_1.

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

g) Resultado do processo e exemplo de erros:

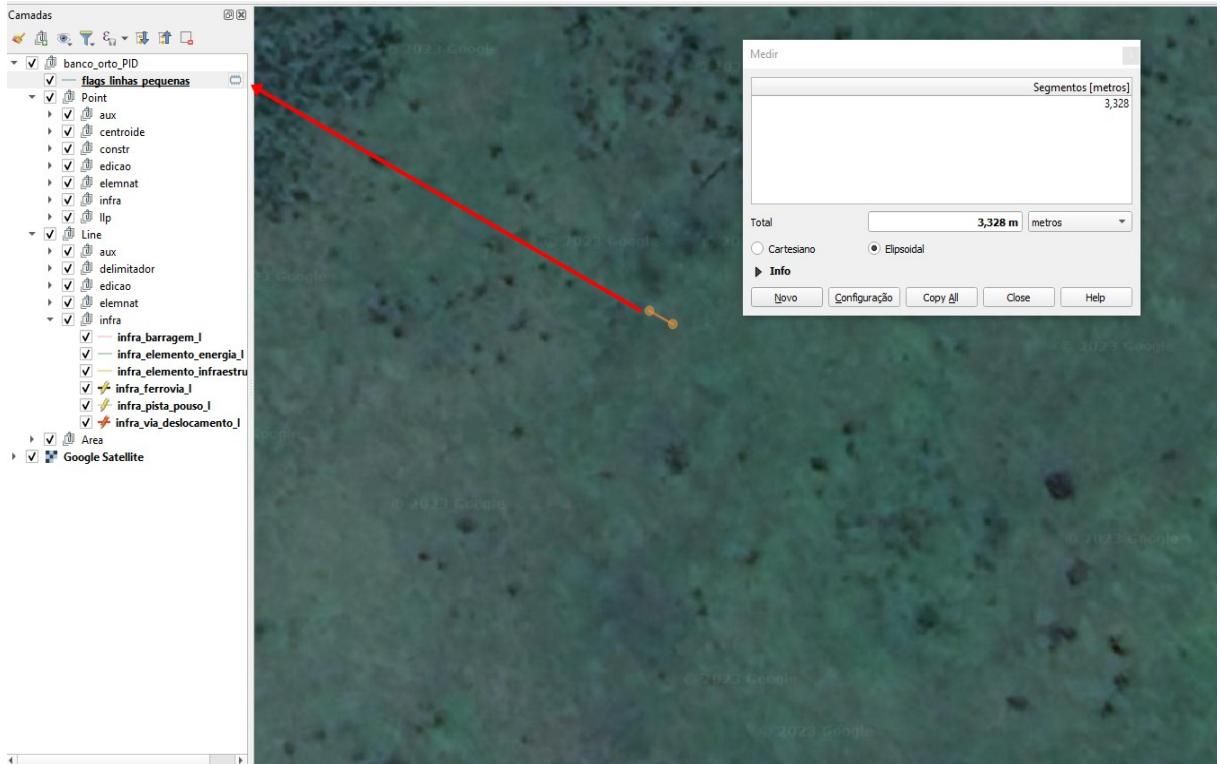


Figura 46: Exemplo de linha pequena.

16. IDENTIFICAR ERROS NA CONSTRUÇÃO DAS REDES DE RODOVIÁRIAS E FERROVIÁRIAS

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

b) Arquivo: `identificar_erros_rede_transporte.model3`.

c) Algoritmos:

```
dsgtools:stringcsvtolayerlistalgorithm;
dsgtools:identifyunmergedlineswithsameattributeset;
dsgtools:stringcsvtofirstlayerwithelementsalgorithm;
dsgtools:identifynetworkconstructionissues;
```

native:mergevectorlayers;
native:extractbylocation.

d) Composição do Modelo:

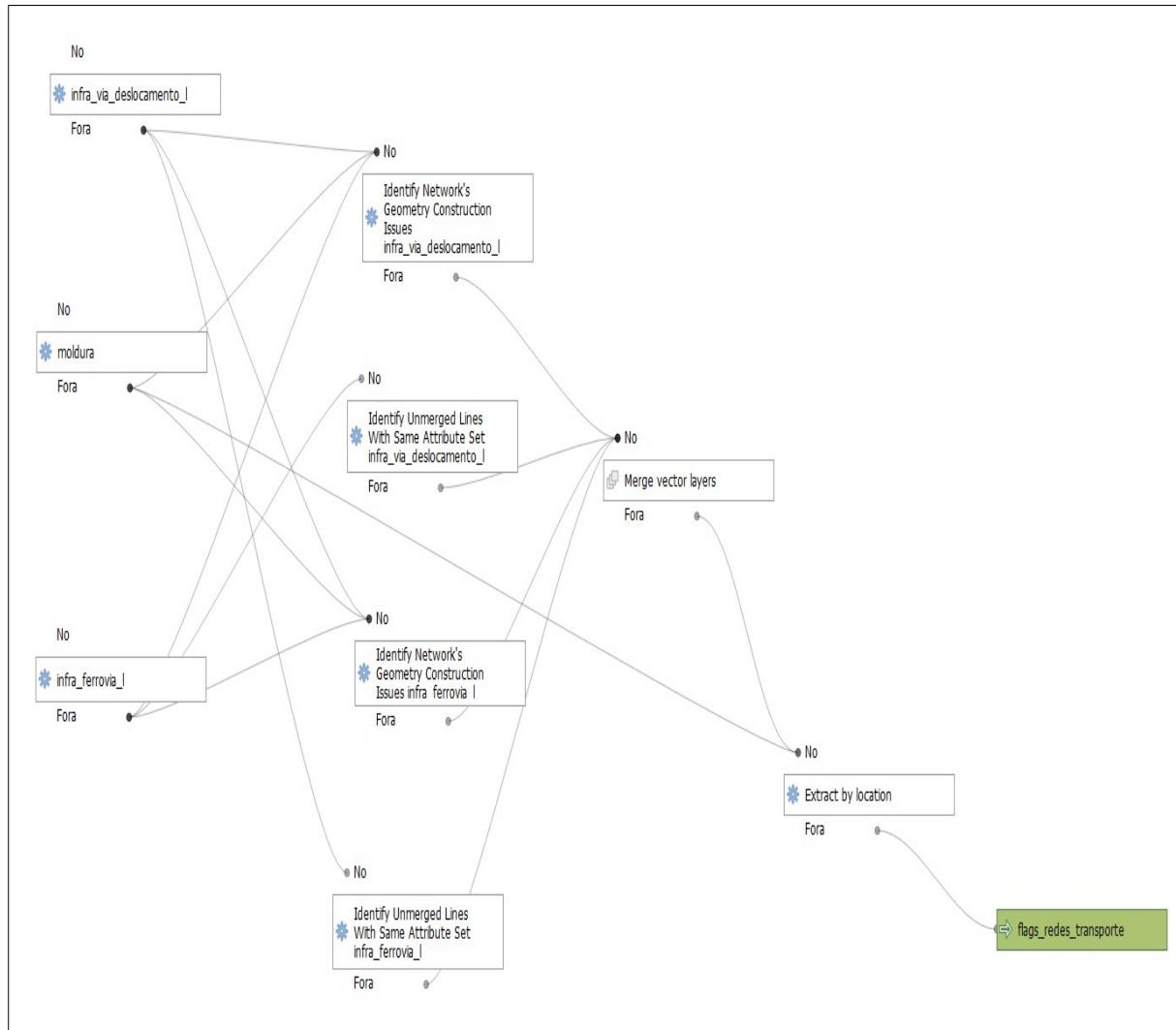


Figura 47: Modelo de identificação de erros na construção das redes de rodoviárias e ferroviárias.

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*.

String CSV to First Layer With Elements: Converte uma string csv em uma camada especificada usando o algoritmo *stringcsvtofirstlayerwithelementsalgorithm*.

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

IdentifyUnmergedLinesWithSameAttributeSet: 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*.

MergeVectorLayers: 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:

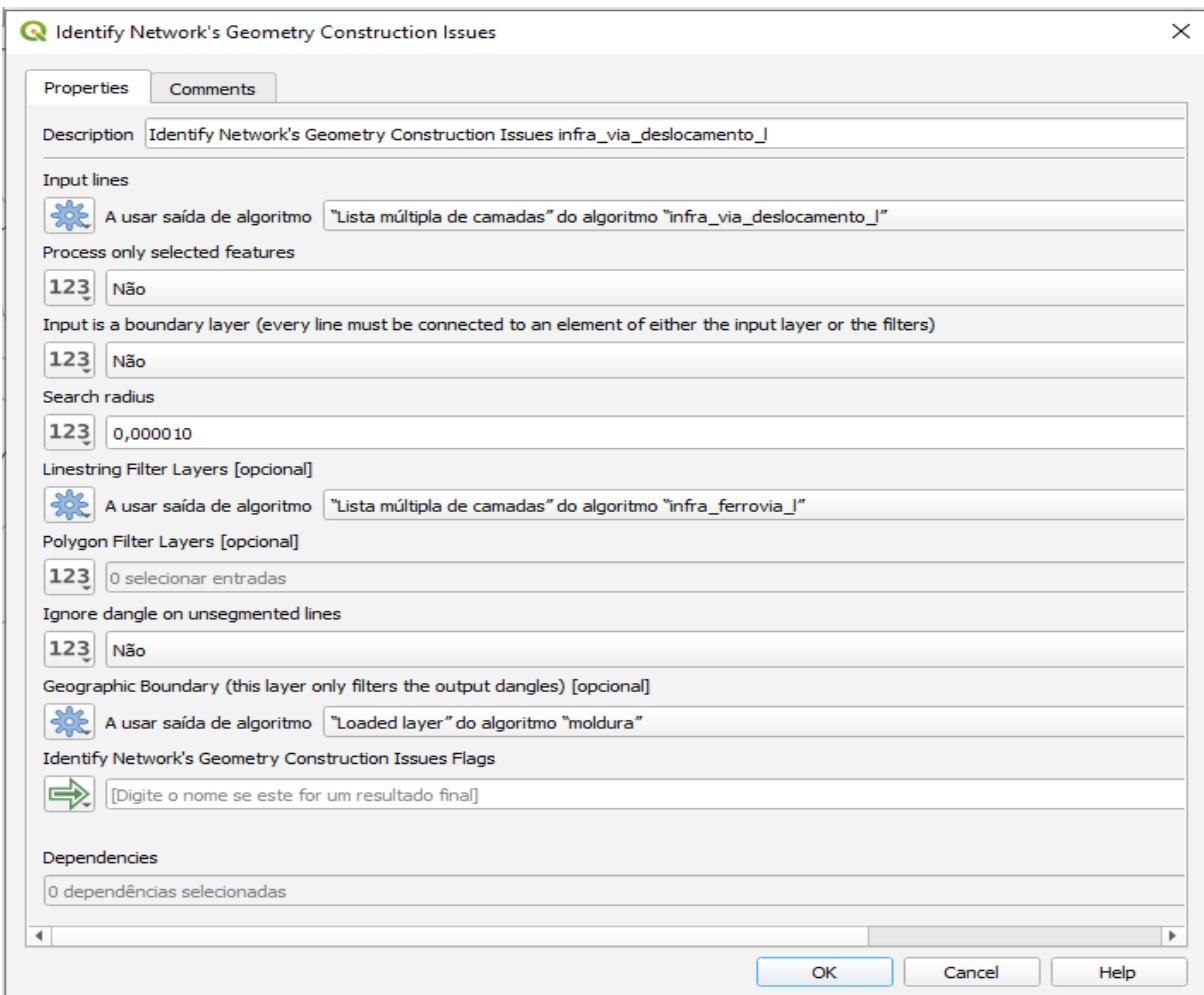


Figura 48: Extrato dos parâmetros

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 unidades do sistema de coordenadas utilizado no processamento. 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 transportes. 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 transportes.

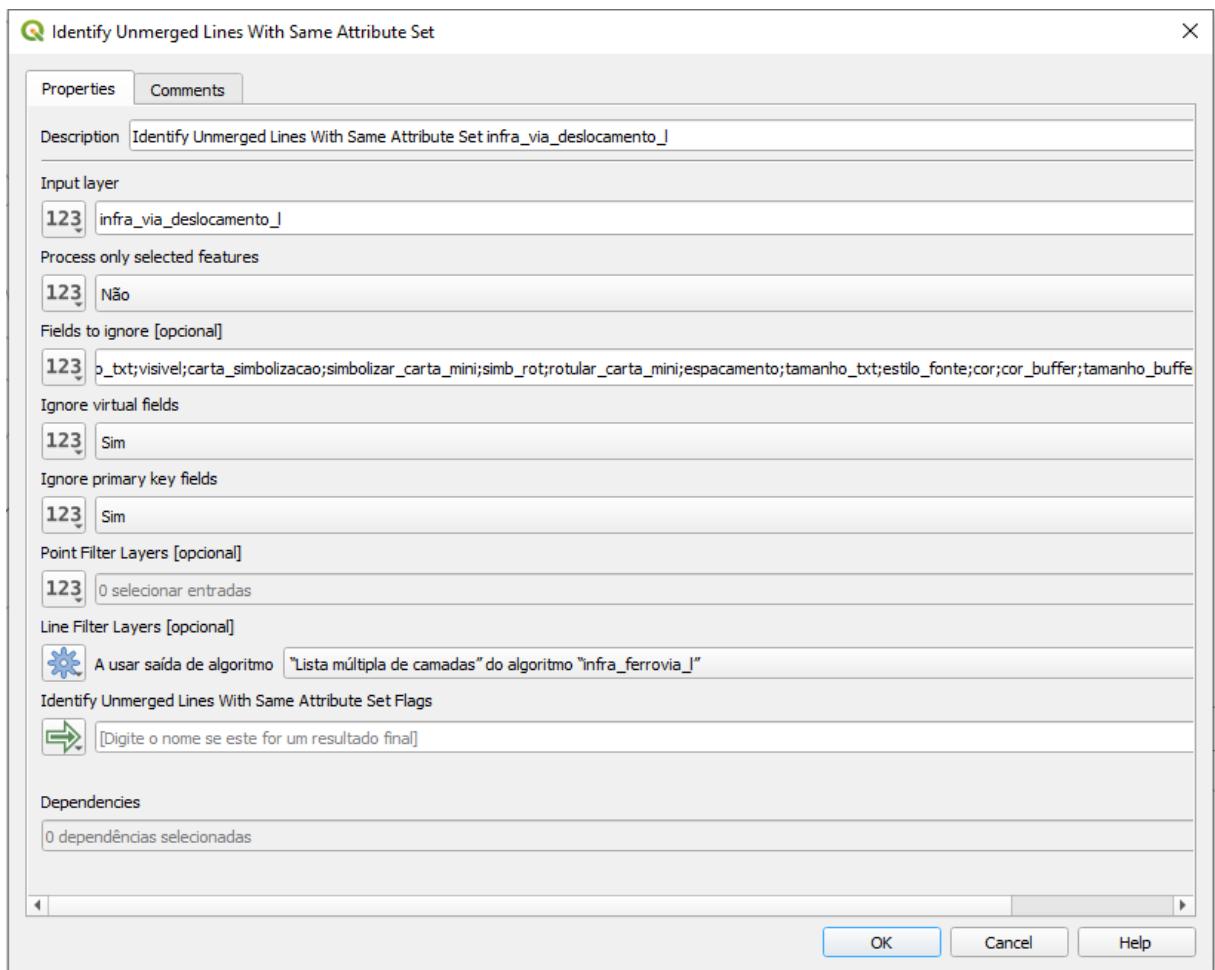


Figura 49: Extrato dos parâmetros.

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 negra de atributos (*ATTRIBUTE_BLACKLIST*) que não serão considerados na comparação das feições, a fim de evitar falsos positivos.

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

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.

No parâmetro "**FILTER LAYER**" as camadas de filtro presentes são utilizadas 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.

Camadas de entrada:

Tipo de geometria	Camadas de entrada
Linha	infra_via_deslocamento_1, infra_ferrovia_1.

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

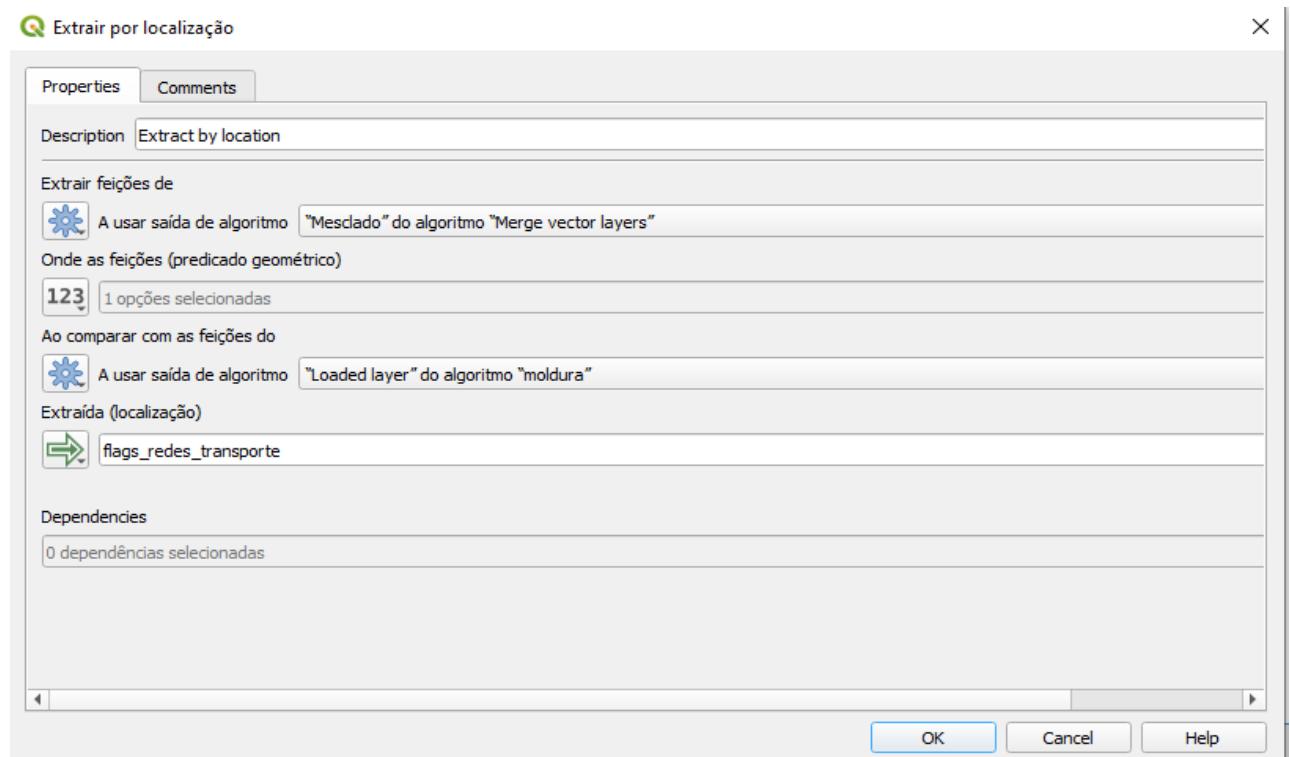


Figura 50: Extrair por localização.

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_redes_transporte*.

g) Resultado do processo e exemplo de erros:

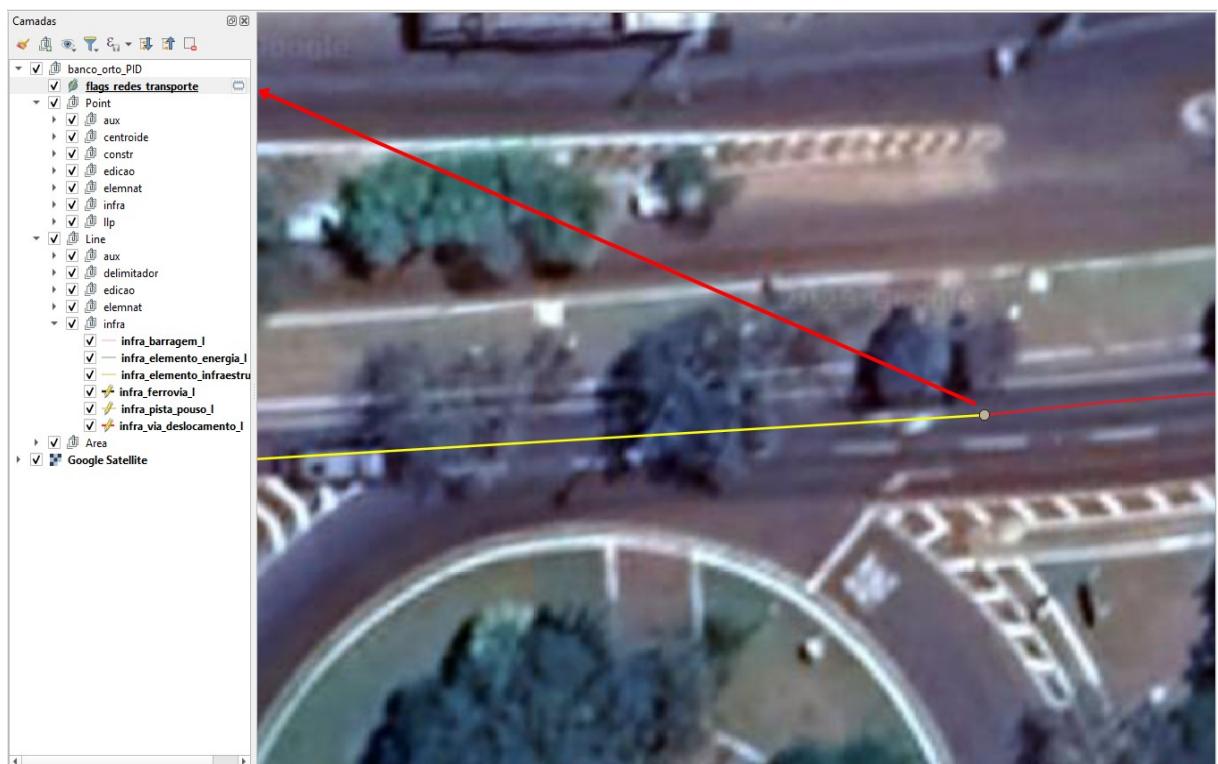


Figura 51: Exemplo de linhas não unidas.