
Aula 6

2024 - 11 - 04

ANA LUISA MAFFINI
2024

Estruturas Básicas

Estruturas Básicas

Algumas das estruturas mais importantes para python:

- 1. Condicionais (if, elif, else):** Permitem executar diferentes blocos de código com base em condições booleanas.
- 2. Laços de repetição (for, while):** Permitem executar blocos de código várias vezes.
- 3. Listas:** São coleções ordenadas de itens que podem ser modificados.
- 4. Tuplas:** São coleções ordenadas imutáveis de itens.
- 5. Dicionários:** São estruturas de dados que armazenam pares chave-valor.
- 6. Funções:** São blocos de código reutilizáveis que executam uma tarefa específica.
- 7. Classes e objetos:** Permitem a criação de tipos de dados personalizados.

Estruturas Básicas

Algumas das estruturas mais importantes para python:

1. Condicionais (if, elif, else): Permitem executar diferentes blocos de código com base em condições booleanas.

```
if idade >= 18:  
    print("Você é maior de idade.")  
else:  
    print("Você é menor de idade.")
```

Estruturas Básicas

Algumas das estruturas mais importantes para python:

2. Laços de repetição (for, while): Permitem executar blocos de código várias vezes.

```
for i in range(5):  
    print(i)
```

```
contador = 0  
while contador < 5:  
    print(contador)  
    contador += 1
```

Estruturas Básicas

Algumas das estruturas mais importantes para python:

3. Listas: São coleções ordenadas de itens que podem ser modificados.

```
minha_lista = [1, 2, 3, 4, 5]
```

Estruturas Básicas

Algumas das estruturas mais importantes para python:

4. **Tuplas:** São coleções ordenadas imutáveis de itens.

```
minha_tupla = (1, 2, 3, 4, 5)
```

Estruturas Básicas

Algumas das estruturas mais importantes para python:

5. Dicionários: São estruturas de dados que armazenam pares chave-valor.

```
meu_dicionario = {'nome': 'João', 'idade': 25, 'cidade': 'São Paulo'}
```


Estruturas Básicas

Algumas das estruturas mais importantes para python:

6. Funções: São blocos de código reutilizáveis que executam uma tarefa específica.

```
def soma(a, b):  
    return a + b
```

Estruturas Básicas

Algumas das estruturas mais importantes para python:

7. Classes e objetos: Permitem a criação de tipos de dados personalizados.

```
class Pessoa:
    def __init__(self, nome, idade):
        self.nome = nome
        self.idade = idade

pessoa1 = Pessoa("Maria", 30)
```

Estruturas Básicas

Classes:

As **classes** são estruturas fundamentais para **criar objetos e definir seu comportamento**.

Uma classe é um **modelo** que define propriedades e comportamentos de objetos que serão criados com base nesse modelo. Funciona como um plano ou um esboço para a criação de objetos.

As classes são compostas por **atributos** (variáveis) e **métodos** (funções) que descrevem características e comportamentos dos objetos que serão instanciados a partir dessa classe.

Por exemplo, uma classe "Carro" pode ter atributos como cor, modelo e ano, e métodos como ligar, desligar, acelerar, entre outros.

Estruturas Básicas

```
class Carro:
    def __init__(self, cor, modelo, ano):
        self.cor = cor
        self.modelo = modelo
        self.ano = ano

    def ligar(self):
        print("O carro está ligado.")

    def desligar(self):
        print("O carro está desligado.")
```

Estruturas Básicas

Neste exemplo:

- `__init__` é um método especial (conhecido como construtor) que é chamado automaticamente quando um novo objeto é criado a partir da classe. Ele inicializa os atributos do objeto.
- **`self`** é uma referência ao próprio objeto que está sendo criado.
- `cor`, `modelo` e `ano` são atributos da classe `Carro`.
- **`ligar()`** e **`desligar()`** são métodos da classe `Carro` que definem o comportamento do objeto.

Estruturas Básicas

Neste exemplo, para criar um **objeto** (também chamado de instância) a partir desta classe, você faria algo assim:

```
meu_carro = Carro("vermelho", "XYZ", 2023)
```

Isso cria um objeto **meu_carro** com base na classe Carro, com os atributos específicos fornecidos.

Github

Github

O **Git**Hub é uma plataforma de **hospedagem** de **código-fonte** e **colaboração** para desenvolvimento de software.

Ele é usado principalmente para **controle de versão** usando o sistema Git, permitindo que indivíduos e equipes trabalhem juntos em projetos, acompanhem as mudanças feitas no código, colaborem em diferentes partes de um projeto e gerenciem o desenvolvimento de software de maneira eficiente.

O GitHub oferece recursos como rastreamento de problemas (issue tracking), ferramentas de revisão de código, integração contínua, hospedagem de sites estáticos e muito mais.

É amplamente utilizado pela comunidade de desenvolvedores para **compartilhar**, **contribuir** e **colaborar** em uma ampla variedade de projetos, desde pequenos projetos individuais até grandes projetos de código aberto mantidos por comunidades inteiras.

Github

No GitHub, um "**fork**" é uma **cópia de um repositório** de código de outra pessoa para o seu próprio perfil no GitHub.

Quando você faz um fork de um repositório, você está criando uma **cópia separada** desse repositório na sua conta, permitindo que você trabalhe nele de forma independente.

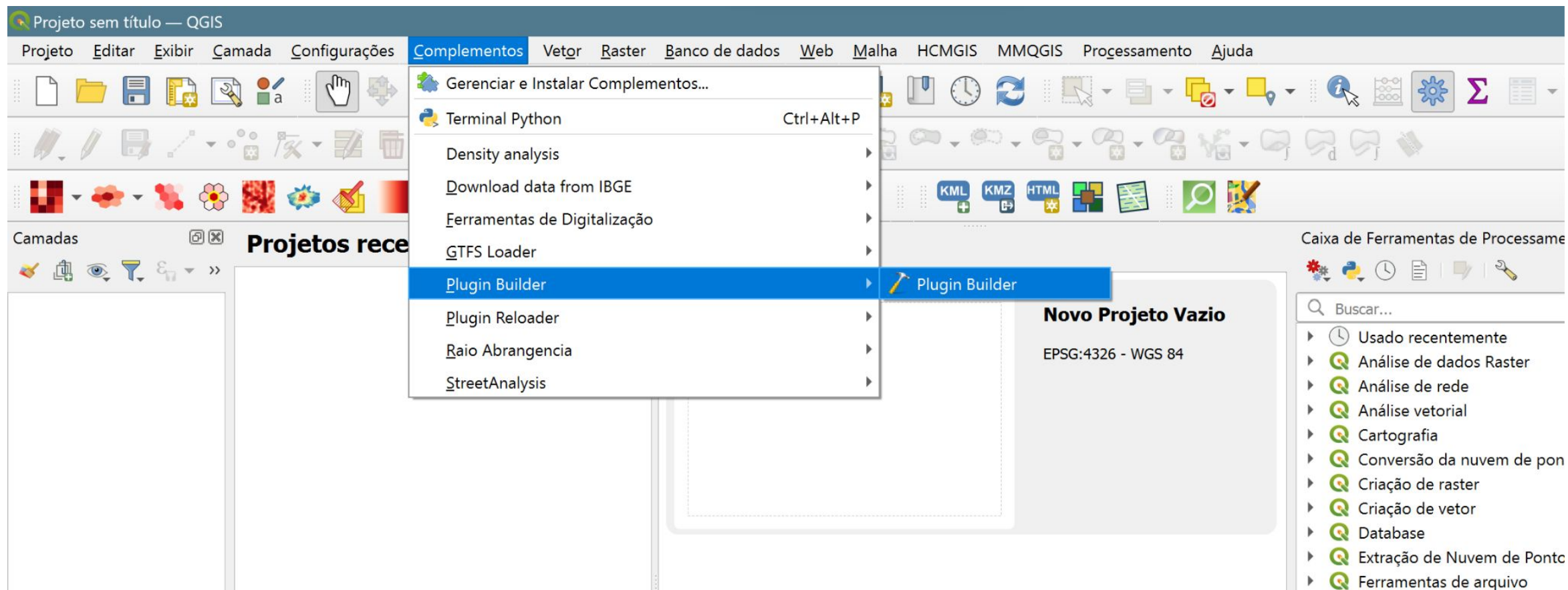
O fork é usado em projetos de código aberto, onde os desenvolvedores desejam contribuir com alterações ou melhorias a um projeto existente mantido por outra pessoa ou organização.

Ao fazer um fork, você pode modificar o código, corrigir bugs, adicionar recursos ou fazer outras alterações **sem afetar diretamente o repositório original**.

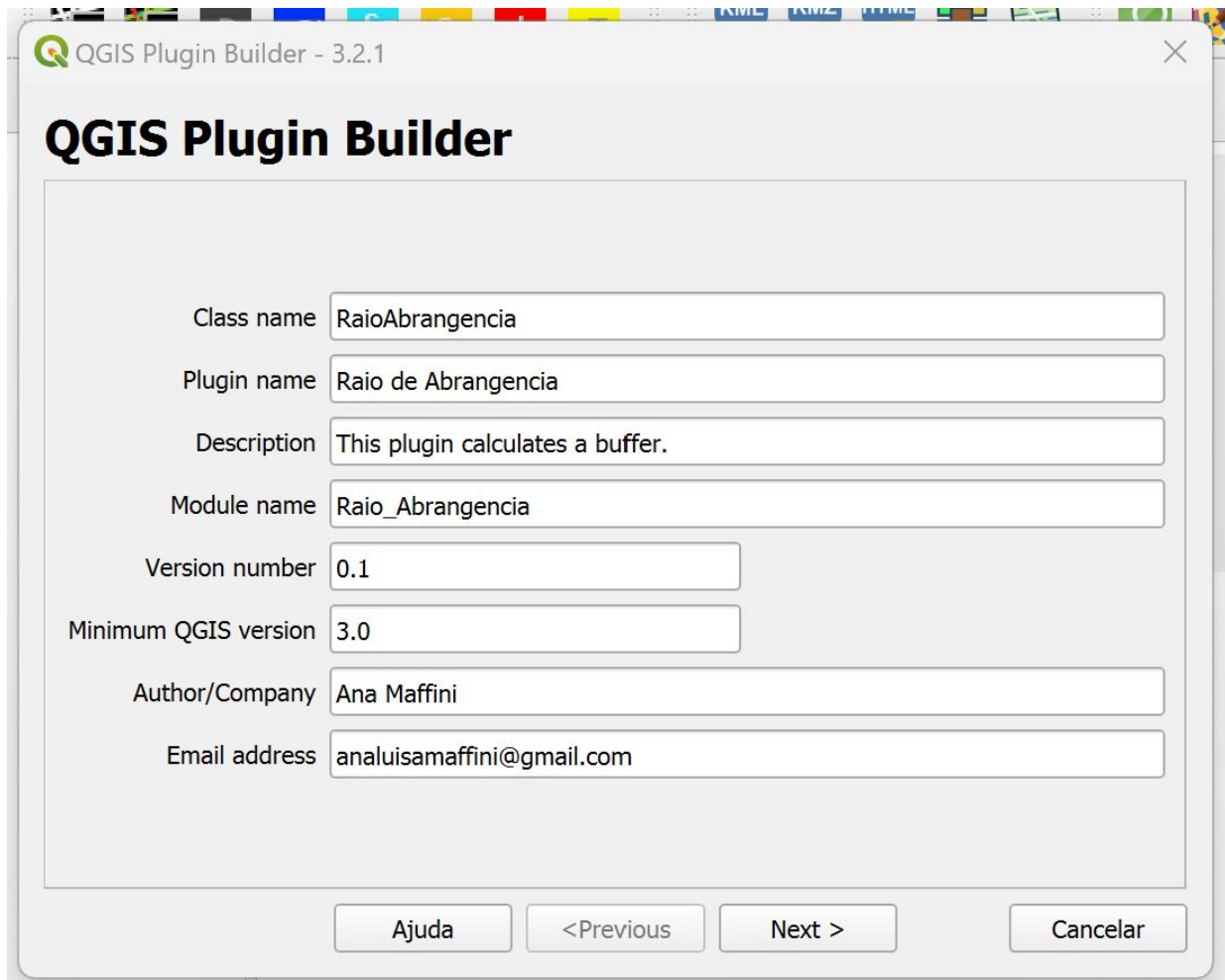
Depois de fazer suas modificações no repositório forkado, você pode sugerir essas alterações de volta para o repositório original por meio de um "pull request" (solicitação de pull), permitindo que o proprietário do repositório original revise suas mudanças e decida se as incorpora ao projeto.

Criação de Plugin - Parte 2

Criação de Plugin



Criação de Plugin



The image shows a screenshot of the 'QGIS Plugin Builder' dialog box, version 3.2.1. The dialog has a title bar with the QGIS logo and the text 'QGIS Plugin Builder - 3.2.1'. The main area is titled 'QGIS Plugin Builder' and contains several input fields for plugin metadata. The fields are: 'Class name' with the value 'RaioAbrangencia', 'Plugin name' with 'Raio de Abrangencia', 'Description' with 'This plugin calculates a buffer.', 'Module name' with 'Raio_Abrangencia', 'Version number' with '0.1', 'Minimum QGIS version' with '3.0', 'Author/Company' with 'Ana Maffini', and 'Email address' with 'analuisamaffini@gmail.com'. At the bottom of the dialog, there are four buttons: 'Ajuda', '<Previous', 'Next >', and 'Cancelar'.

QGIS Plugin Builder - 3.2.1

QGIS Plugin Builder

Class name: RaioAbrangencia

Plugin name: Raio de Abrangencia

Description: This plugin calculates a buffer.

Module name: Raio_Abrangencia

Version number: 0.1

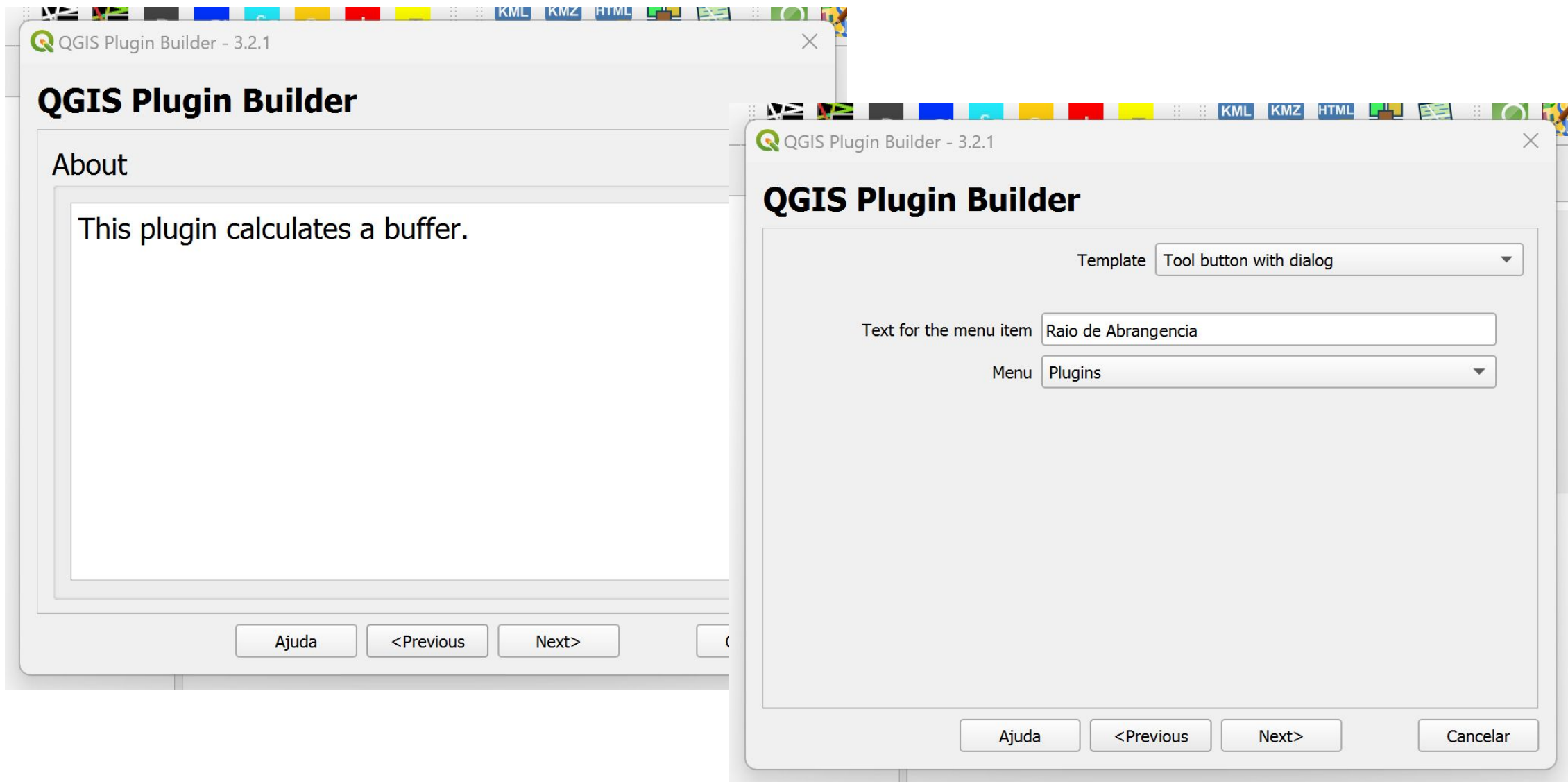
Minimum QGIS version: 3.0

Author/Company: Ana Maffini

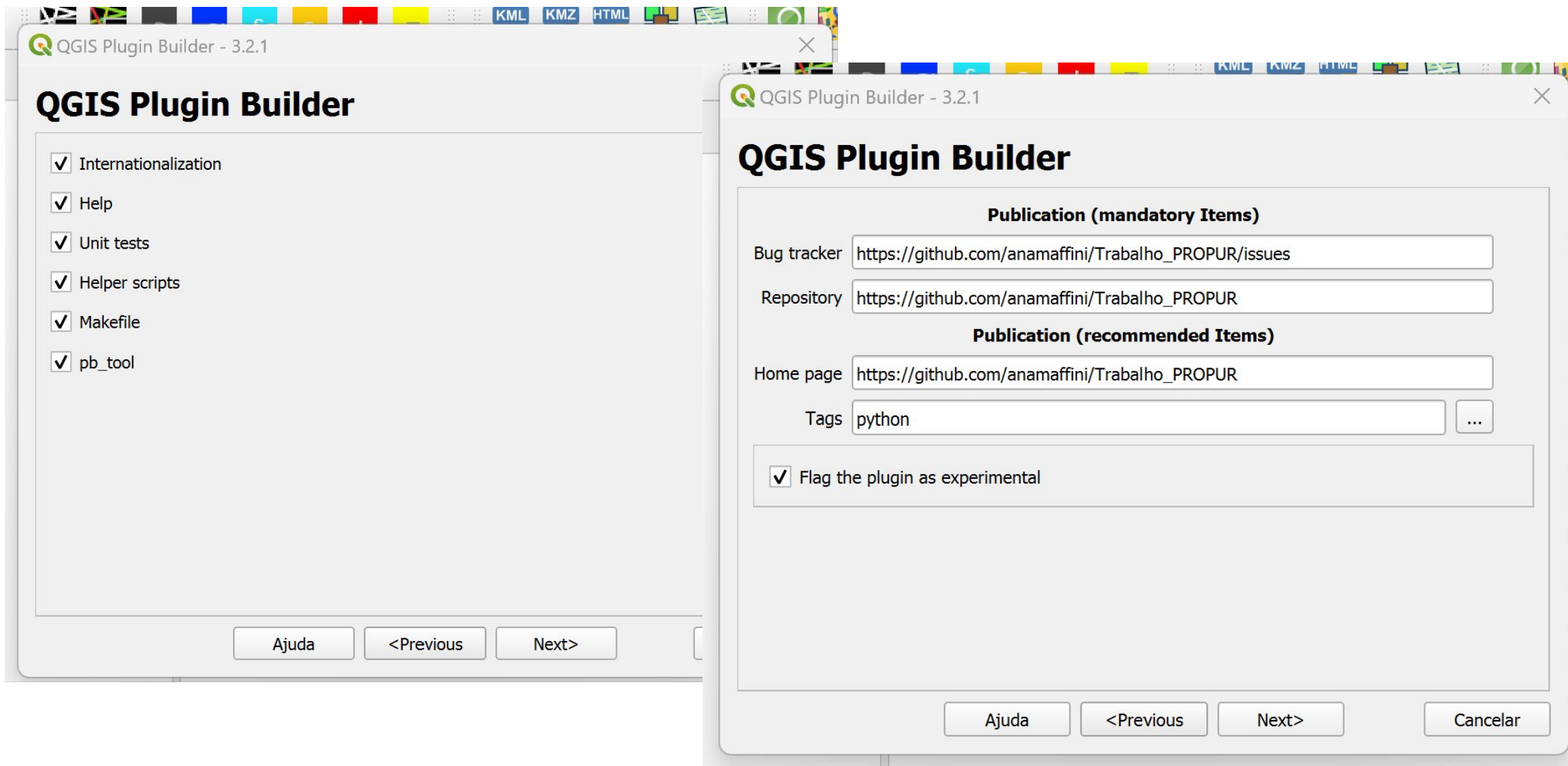
Email address: analuisamaffini@gmail.com

Ajuda <Previous Next > Cancelar

Criação de Plugin



Criação de Plugin



QGIS Plugin Builder

☒ Internationalization

☒ Help

☒ Unit tests

☒ Helper scripts

☒ Makefile

☒ pb_tool

Ajuda <Previous Next>

QGIS Plugin Builder

Publication (mandatory Items)

Bug tracker

Repository

Publication (recommended Items)

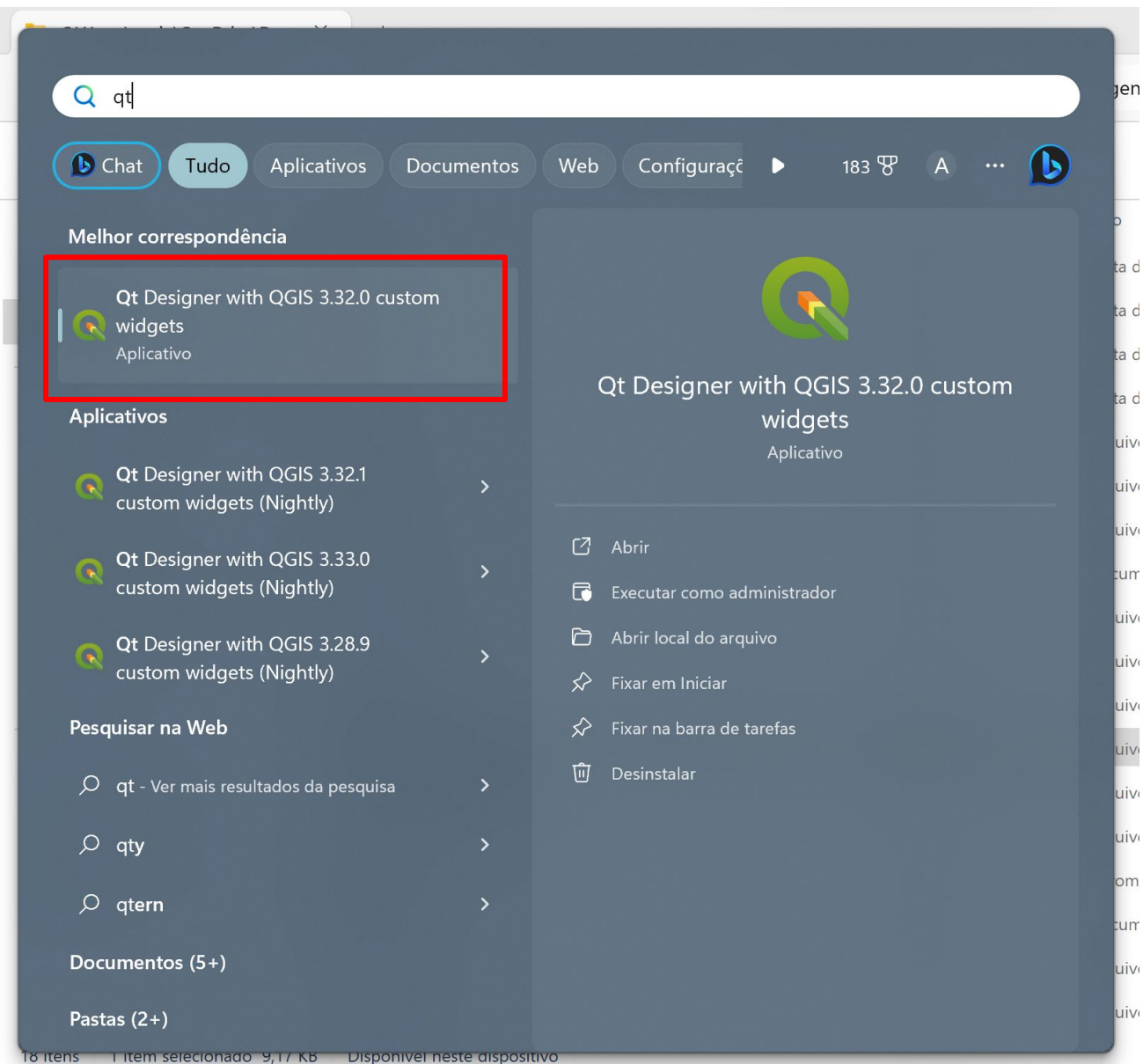
Home page

Tags ...

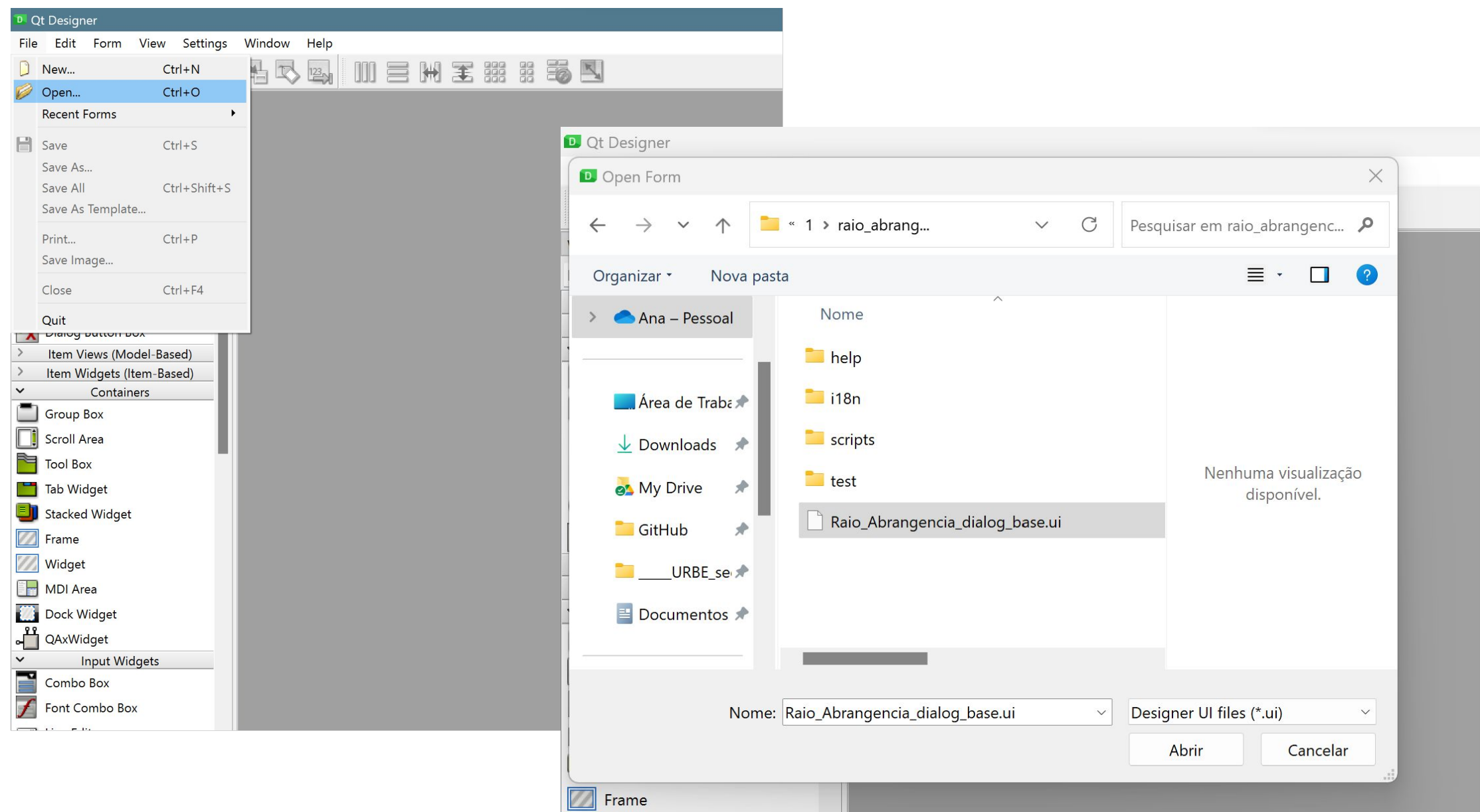
☒ Flag the plugin as experimental

Ajuda <Previous Next> Cancelar

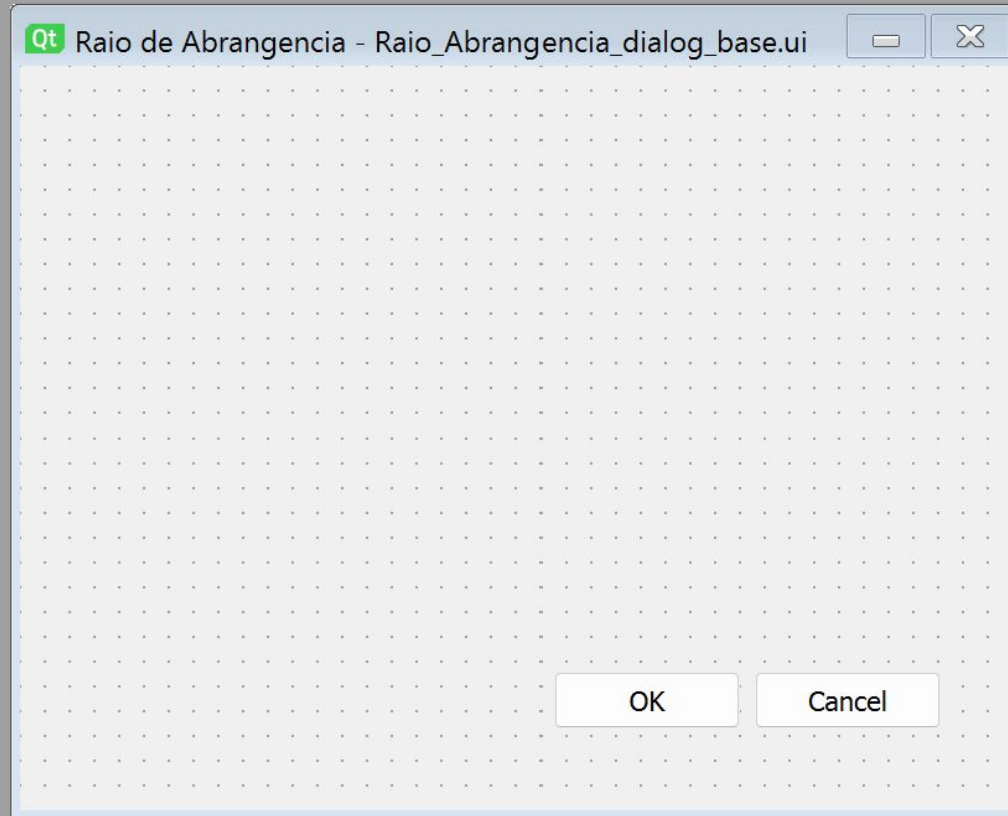
Criação de Plugin



Criação de Plugin



Criação de Plugin



Criação de Plugin

Qt Designer

File Edit Form View Settings Window H

Widget Box

Filter

Buttons

OK

Push Button

Tool

Tool Button

Radio

Radio Button

Check

Check Box

Command

Command Link Button

Dialog

Dialog Button Box

Item Views (Model-Based)

List

List View

Tree

Tree View

Table

Table View

Column

Column View

Undo

Undo View

Item Widgets (Item-Based)

Containers

Input Widgets

Combo

Combo Box

Font

Font Combo Box

Line

Line Edit

Text

Text Edit

Plain

Plain Text Edit

Spin

Spin Box

Double

Double Spin Box

Time

Time Edit

Date

Date Edit

Date/Time

Date/Time Edit

Dial

Dial

Qt

Raio de Abrangencia - Raio_Abrangencia_dialog_...

Input Point Vector:

Radius:

0

Output Polygon Vector: ...

OK Cancel

Object Inspector

Filter

| Object | Class |
|-----------------------------|------------------|
| ▼ RaioAbrangenciaDialogBase | QDialog |
| InputPoint | QComboBox |
| Output | QLineEdit |
| Radius | QSpinBox |
| button_box | QDialogButtonBox |
| label | QLabel |
| label_2 | QLabel |
| label_3 | QLabel |
| toolButton | QToolButton |

Property Editor

Filter

RaioAbrangenciaDialogBase : QDialog

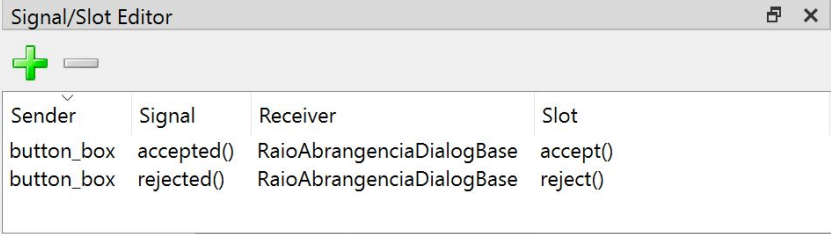
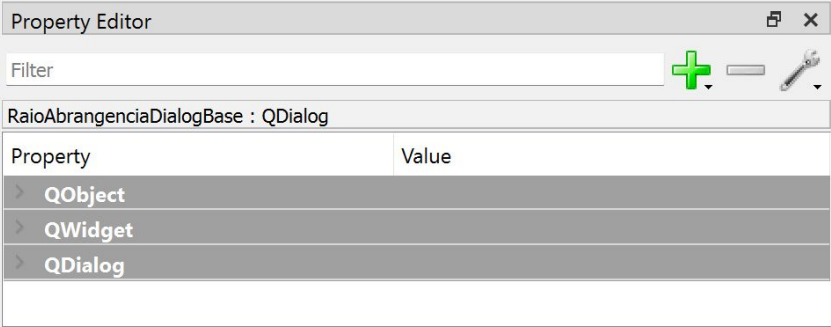
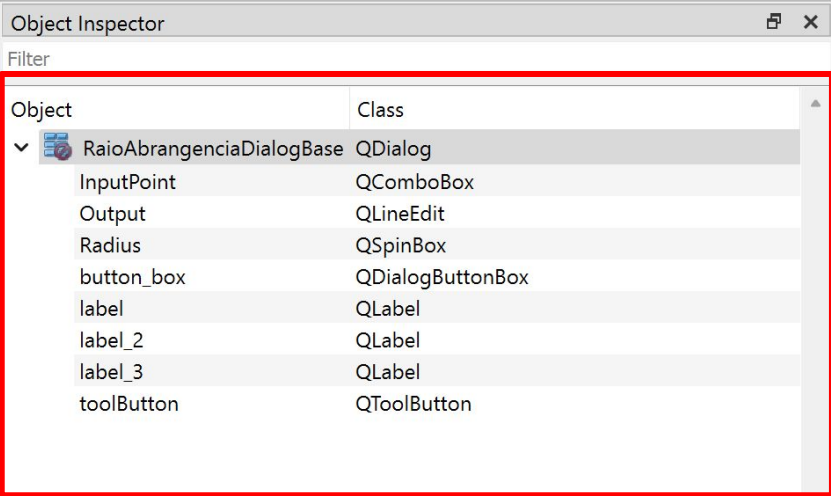
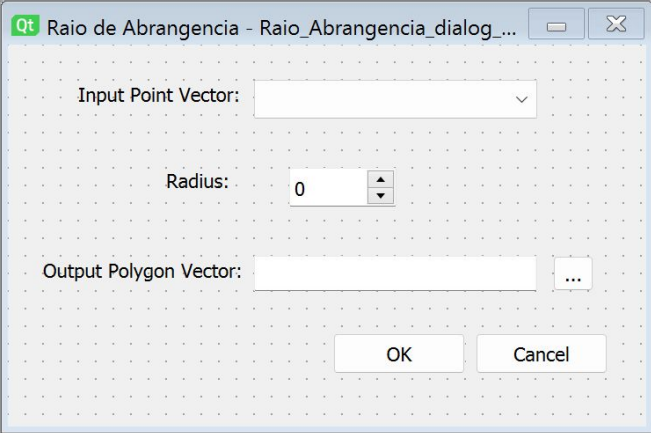
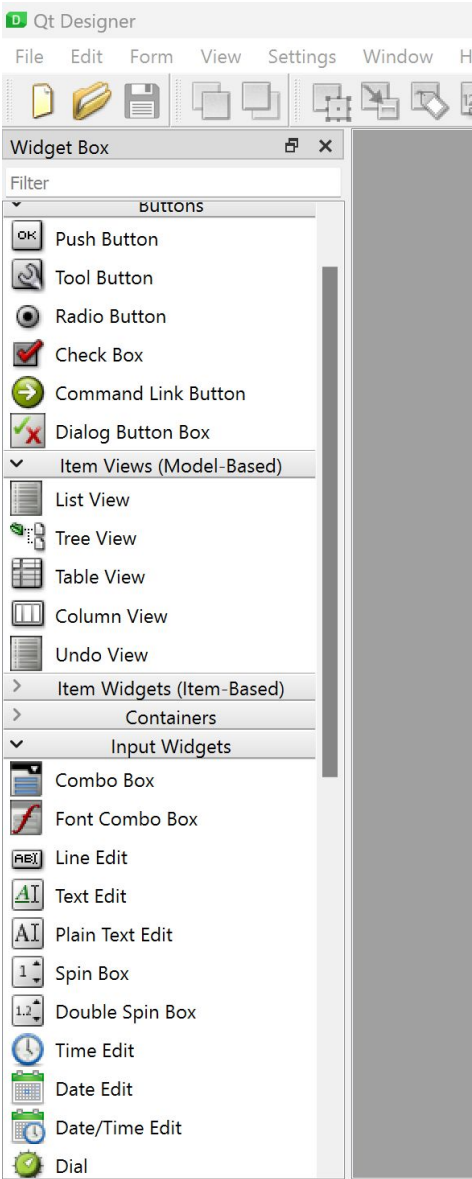
| Property | Value |
|----------|-------|
| QObject | |
| QWidget | |
| QDialog | |

Signal/Slot Editor

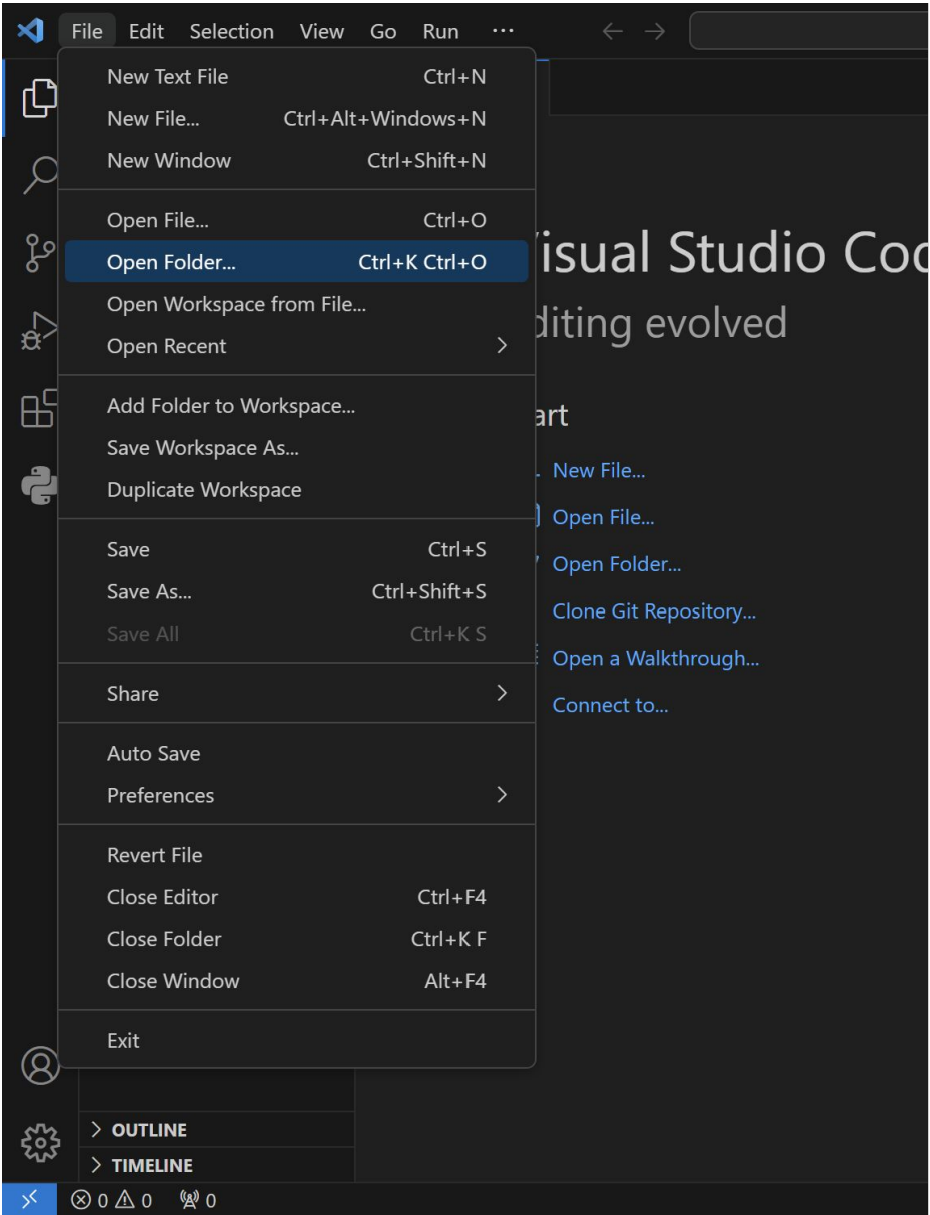
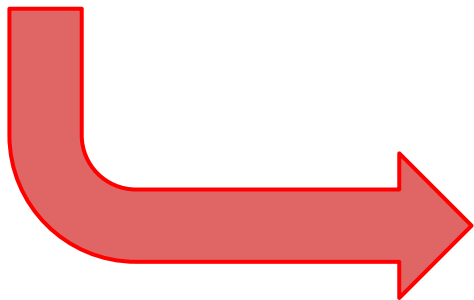
| Sender | Signal | Receiver | Slot |
|------------|------------|---------------------------|----------|
| button_box | accepted() | RaioAbrangenciaDialogBase | accept() |
| button_box | rejected() | RaioAbrangenciaDialogBase | reject() |

Signal/Slot Editor Action Editor Resource Browser

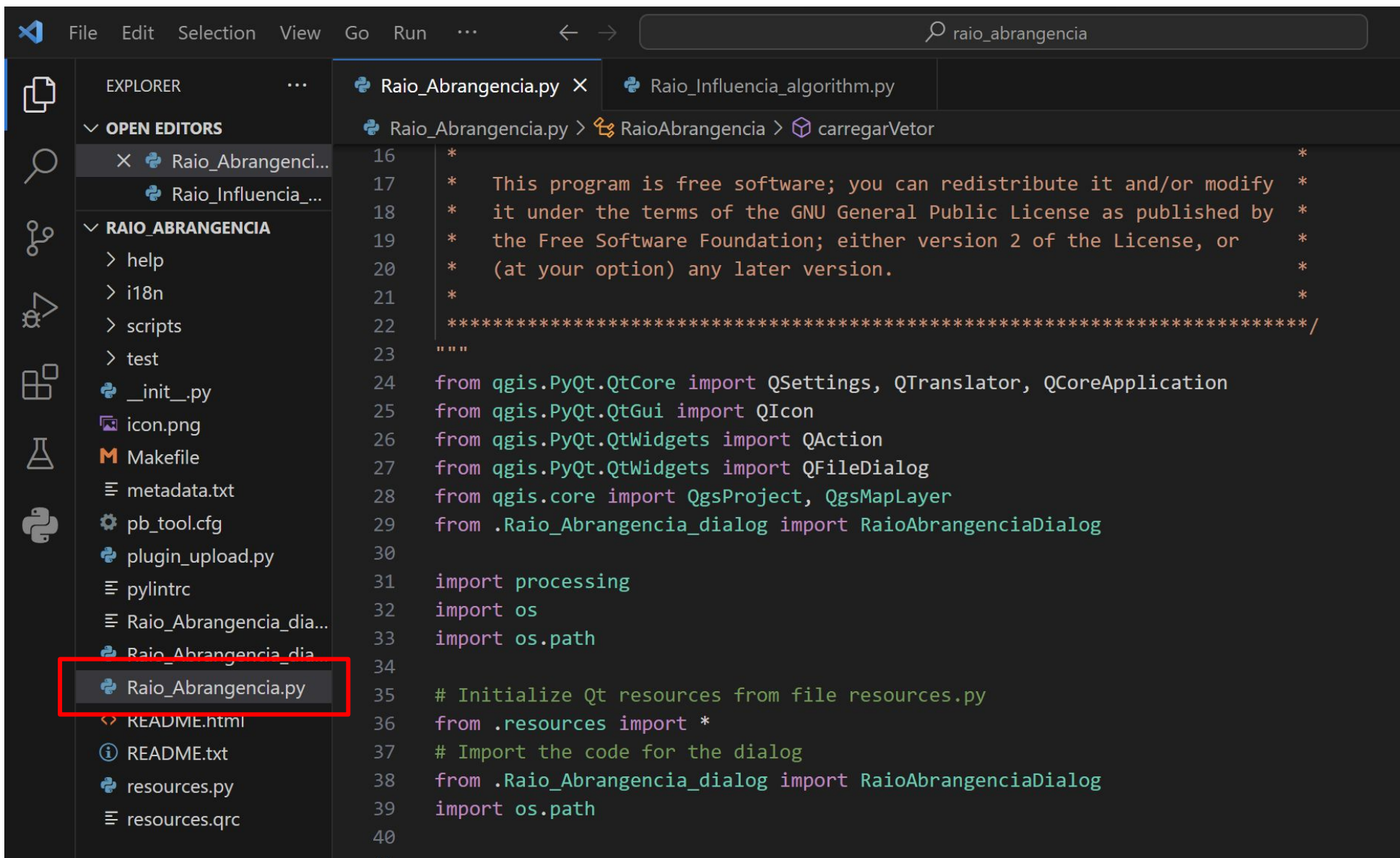
Criação de Plugin



Criação de Plugin



Criação de Plugin



```
File Edit Selection View Go Run ... raio_abrangencia
```

EXPLORER

OPEN EDITORS

- Raio_Abrangenci...
- Raio_Influencia_...

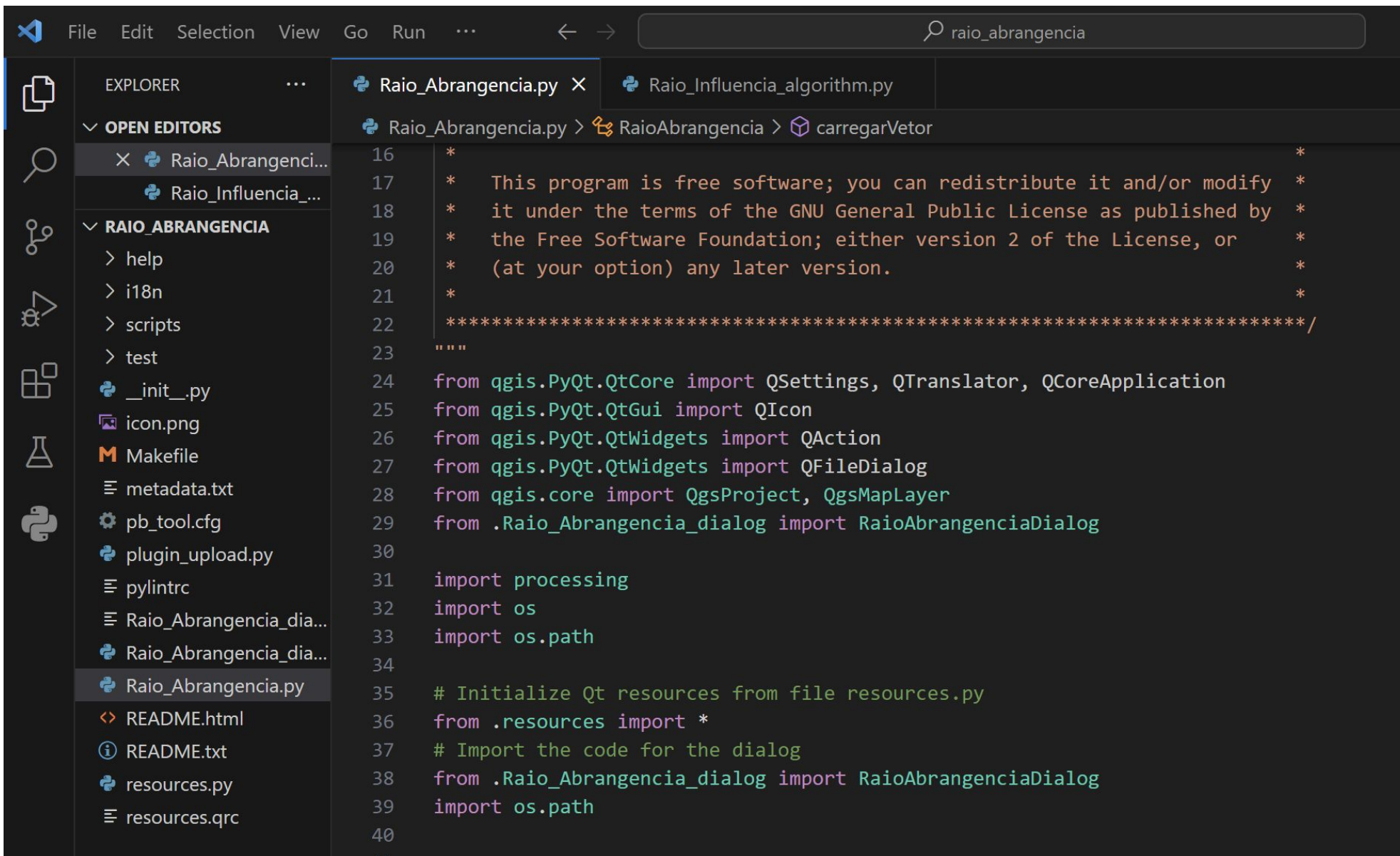
RAIO_ABRANGENCIA

- help
- i18n
- scripts
- test
- __init__.py
- icon.png
- Makefile
- metadata.txt
- pb_tool.cfg
- plugin_upload.py
- pylintrc
- Raio_Abrangencia_dia...
- Raio_Abrangencia_dia...
- Raio_Abrangencia.py**
- README.html
- README.txt
- resources.py
- resources.qrc

Raio_Abrangencia.py

```
16 *
17 * This program is free software; you can redistribute it and/or modify *
18 * it under the terms of the GNU General Public License as published by *
19 * the Free Software Foundation; either version 2 of the License, or *
20 * (at your option) any later version. *
21 *
22 *****/
23 """
24 from qgis.PyQt.QtCore import QSettings, QTranslator, QCoreApplication
25 from qgis.PyQt.QtGui import QIcon
26 from qgis.PyQt.QtWidgets import QAction
27 from qgis.PyQt.QtWidgets import QFileDialog
28 from qgis.core import QgsProject, QgsMapLayer
29 from .Raio_Abrangencia_dialog import RaioAbrangenciaDialog
30
31 import processing
32 import os
33 import os.path
34
35 # Initialize Qt resources from file resources.py
36 from .resources import *
37 # Import the code for the dialog
38 from .Raio_Abrangencia_dialog import RaioAbrangenciaDialog
39 import os.path
40
```


Criação de Plugin



```
File Edit Selection View Go Run ... raio_abrangencia

EXPLORER
OPEN EDITORS
  X Raio_Abrangenci...
  Raio_Influencia_...
  RAIO_ABRANGENCIA
    > help
    > i18n
    > scripts
    > test
    _init_.py
    icon.png
    Makefile
    metadata.txt
    pb_tool.cfg
    plugin_upload.py
    pylintrc
    Raio_Abrangencia_dia...
    Raio_Abrangencia_dia...
    Raio_Abrangencia.py
    README.html
    README.txt
    resources.py
    resources.qrc

Raio_Abrangencia.py X Raio_Influencia_algorithm.py
Raio_Abrangencia.py > RaioAbrangencia > carregarVetor

16 *
17 * This program is free software; you can redistribute it and/or modify *
18 * it under the terms of the GNU General Public License as published by *
19 * the Free Software Foundation; either version 2 of the License, or *
20 * (at your option) any later version. *
21 *
22 *****/
23 """
24 from qgis.PyQt.QtCore import QSettings, QTranslator, QCoreApplication
25 from qgis.PyQt.QtGui import QIcon
26 from qgis.PyQt.QtWidgets import QAction
27 from qgis.PyQt.QtWidgets import QFileDialog
28 from qgis.core import QgsProject, QgsMapLayer
29 from .Raio_Abrangencia_dialog import RaioAbrangenciaDialog
30
31 import processing
32 import os
33 import os.path
34
35 # Initialize Qt resources from file resources.py
36 from .resources import *
37 # Import the code for the dialog
38 from .Raio_Abrangencia_dialog import RaioAbrangenciaDialog
39 import os.path
40
```

Criação de Plugin

```
180
181 def unload(self):
182     """Removes the plugin menu item and icon from QGIS GUI."""
183     for action in self.actions:
184         self.iface.removePluginMenu(
185             self.tr(u'&Raio de Abrangencia'),
186             action)
187         self.iface.removeToolBarIcon(action)
188
189 def carregarVetor(self):
190     self.dlg.InputPoint.clear()
191     lista_layers = [layer for layer in QgsProject.instance().mapLayers().values()]
192     lista_layer_vetor = []
193     for layer in lista_layers:
194         if layer.type() == QgsMapLayer.VectorLayer: lista_layer_vetor.append(layer.name())
195     self.dlg.InputPoint.addItems(lista_layer_vetor)
196
197 def saidaVetor(self):
198     camada_salvar = str(QFileDialog.getSaveFileName(caption="Defina a layer de saída...", filter="Shapefiles (*.shp)")[0])
199     self.dlg.Output.setText(camada_salvar)
200
201 def variaveis(self):
202     # Get the name of the selected layer from the combo box
203     layer_name = self.dlg.InputPoint.currentText()
204     # Retrieve the layer object using the layer name
205     self.camada = QgsProject.instance().mapLayersByName(layer_name)[0] if layer_name else None
206     self.saida = self.dlg.Output.text()
207     self.radius = self.dlg.Radius.value()
208
209
210 def run(self):
211     """Run method that performs all the real work"""
212
```


Criação de Plugin

```
210 def run(self):
211     """Run method that performs all the real work"""
212
213     # Create the dialog with elements (after translation) and keep reference
214     # Only create GUI ONCE in callback, so that it will only load when the plugin is started
215     if self.first_start == True:
216         self.first_start = False
217         self.dlg = RaioAbrangenciaDialog()
218
219     # show the dialog
220     self.dlg.show()
221
222     #adicionando as funções criadas
223     self.carregarVetor()
224     self.dlg.toolButton.clicked.connect(self.saidaVetor)
225
226     # Run the dialog event loop
227     result = self.dlg.exec_()
228
229     # See if OK was pressed
230     if result:
231         #chamando as variaveis
232         self.variaveis()
233
234         #aplicando o buffer - usando a ferramenta do processamento do qgis
235         buffer = processing.run(
236             "native:buffer",
237             {'INPUT': self.camada,
238             'DISTANCE': self.radius,
239             'DISSOLVE': False,
240             'END_CAP_STYLE': 0,
241             'JOIN_STYLE': 0,
242             'OUTPUT': self.saida})
243
244         #carregando o resultado em tela
245         self.iface.addVectorLayer(self.saida, str.split(os.path.basename(self.saida),".") [0], "ogr")
246         pass
```

Criação de Plugin

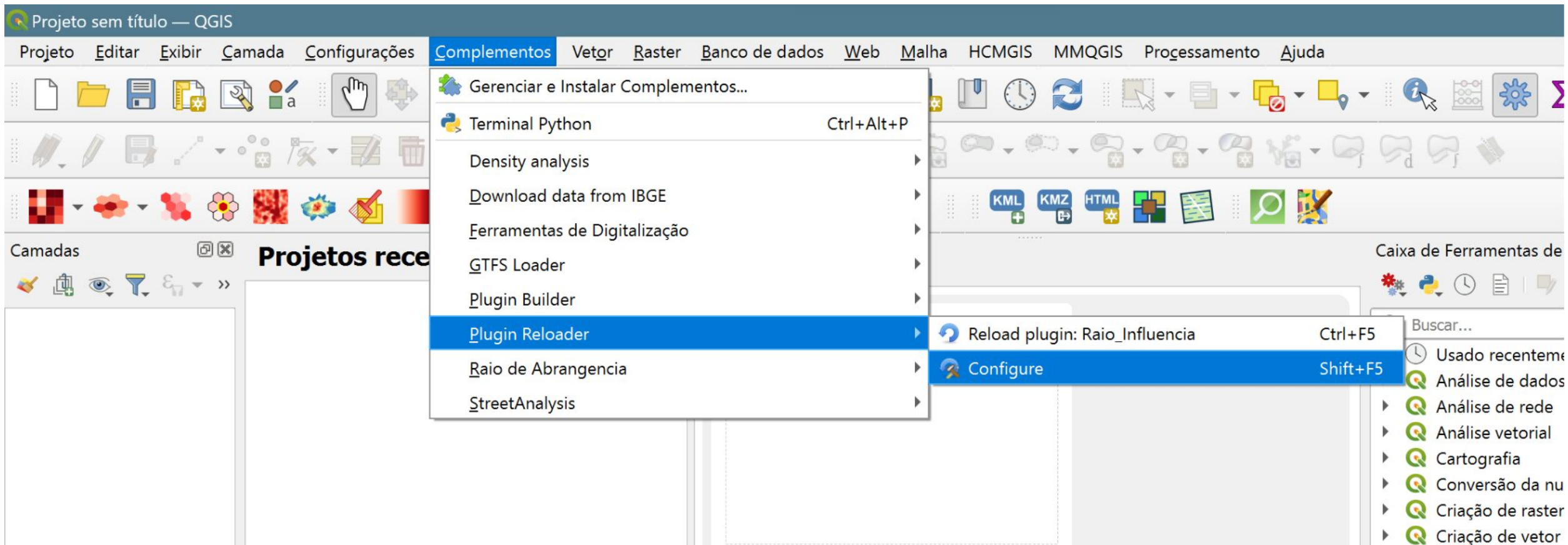
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\analú\OneDrive\Documents\GitHub\Trabalho_PROPUR\1\raio_abrangencia> pb_tool deploy
```

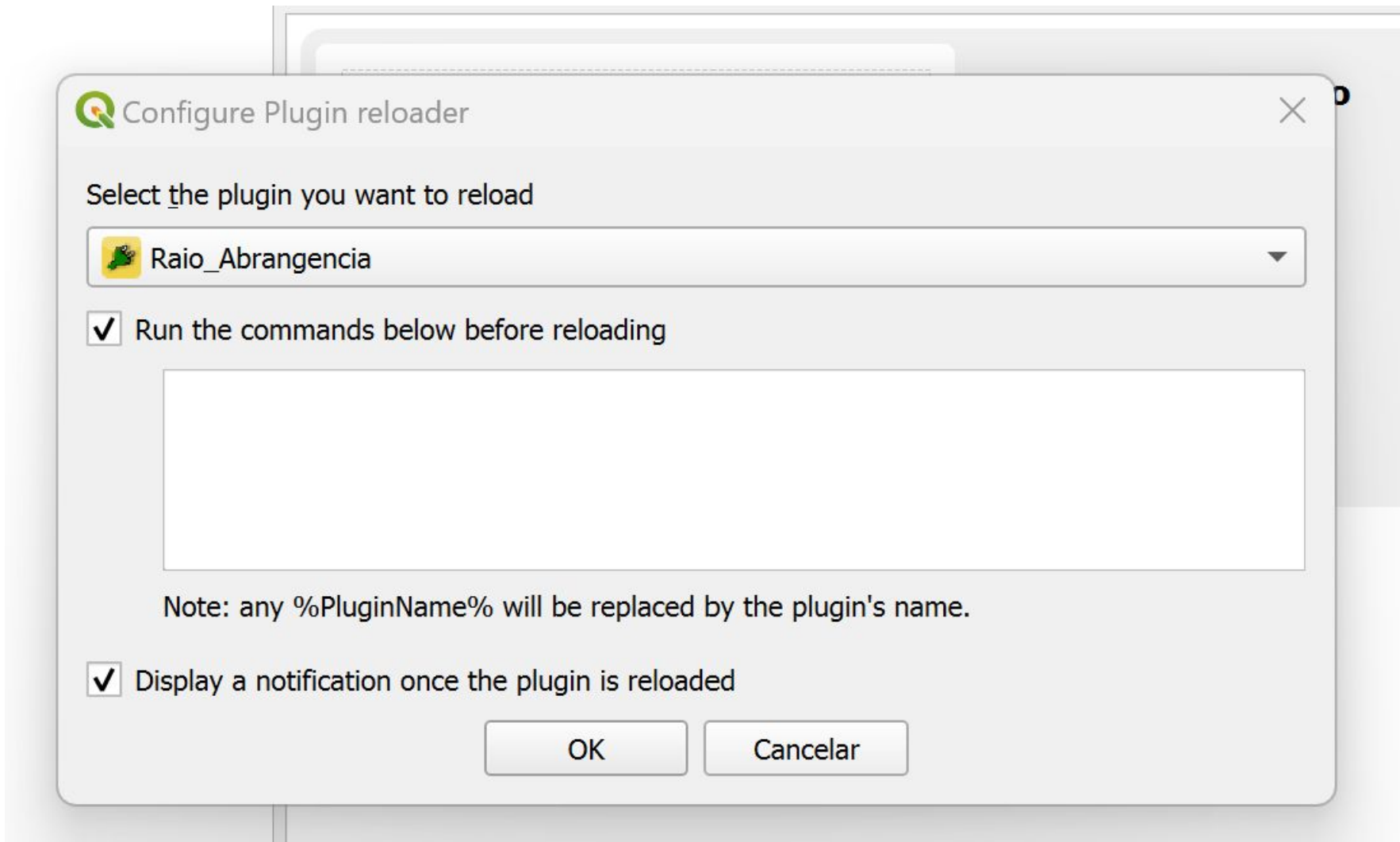
```
PS C:\Users\analú\OneDrive\Documents\GitHub\Trabalho_PROPUR\1\raio_abrangencia> pb_tool deploy
Deploying to C:/Users/analú/AppData/Roaming\QGIS/QGIS3/profiles/default/python/plugins\Raio_Abrangencia
Deploying will:
    * Remove your currently deployed version
    * Compile the ui and resource files
    * Build the help docs
    * Copy everything to your C:/Users/analú/AppData/Roaming\QGIS/QGIS3/profiles/default/python/plugins\Raio_Abrangencia directory

Proceed? [y/N]: y
```

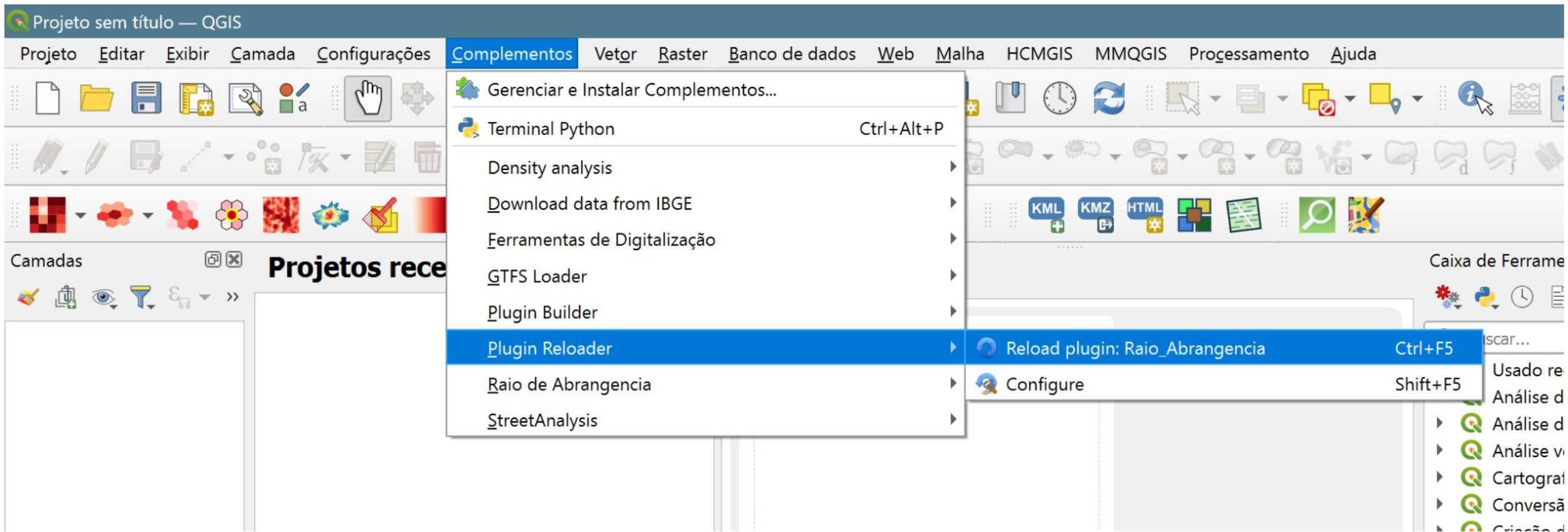
Criação de Plugin



Criação de Plugin



Criação de Plugin



Criação de Plugin

