

Criando Plugins QGIS com pyQGIS

Módulo 5 –Plugin IndexOrama

1 - Introdução

Vamos criar um Plugin que abrirá as bandas 2, 3, 4, 5, 6, 7, 8A, 11 e 12 do Sentinel2 e criar um Stack destas bandas em um único arquivo multibanda (9 bandas). Deste arquivo vamos gerar uma series de índices normalizados, comuns em análises de vegetação, presença de água, umidade e solo exposto.

Os índices gerados serão:

NDVI - Normalized Difference Vegetation Index $(\text{nir-red})/(\text{nir+red})$

NDVIRE1 - red-edge based NDVI $(\text{nir-vnir})/(\text{nir+vnir})$

SAVI - Soil Adjusted Vegetation Index $(\text{nir-red})/(\text{nir+red+0.5})*1.5$

NDWI – Normalized Difference Water Index $(\text{green-nir})/(\text{green +nir})$

MNDI – Modified Normalized Difference Water Index $(\text{green-swir1})/(\text{green +swir1})$

NDMI - Normalized Difference Moisture Index $(\text{nir-swir1})/(\text{nir +swir1})$

NDTI - Normalized Difference Tillage Index $(\text{swir1-swir2})/(\text{swir1 +swir2})$

NDBI - Normalized Difference Built-up Index $(\text{swir1-nir})/(\text{swir1 +nir})$

Onde:

red = banda4 vermelho visível

nir = banda8A infravermelho próximo

vnir – banda5 infravermelho muito próximo

green = banda3 verde visível

swir = banda11 infravermelho de onda curta

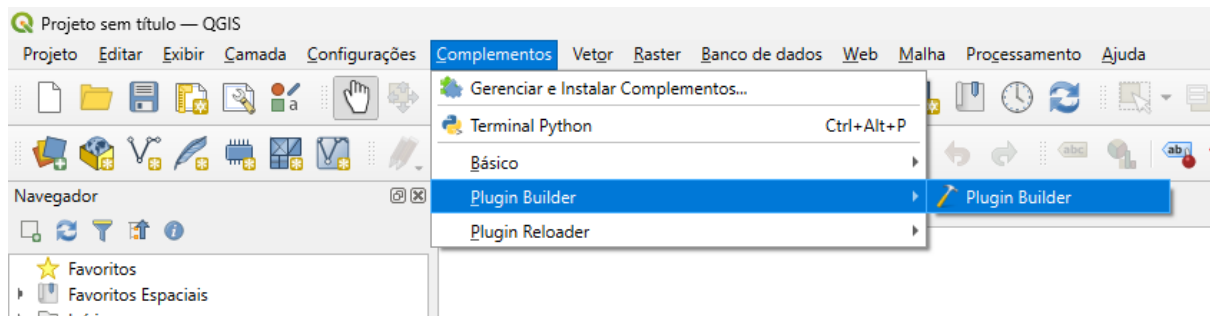
swir2 = banda12 infravermelho de onda curta

As bandas originais foram recortadas e as bandas com 10m de resolução (2,3, 4 e 8A) tiveram a resolução transformada para 20 metros para serem compatíveis com a resolução das bandas 5, 11 e 12.

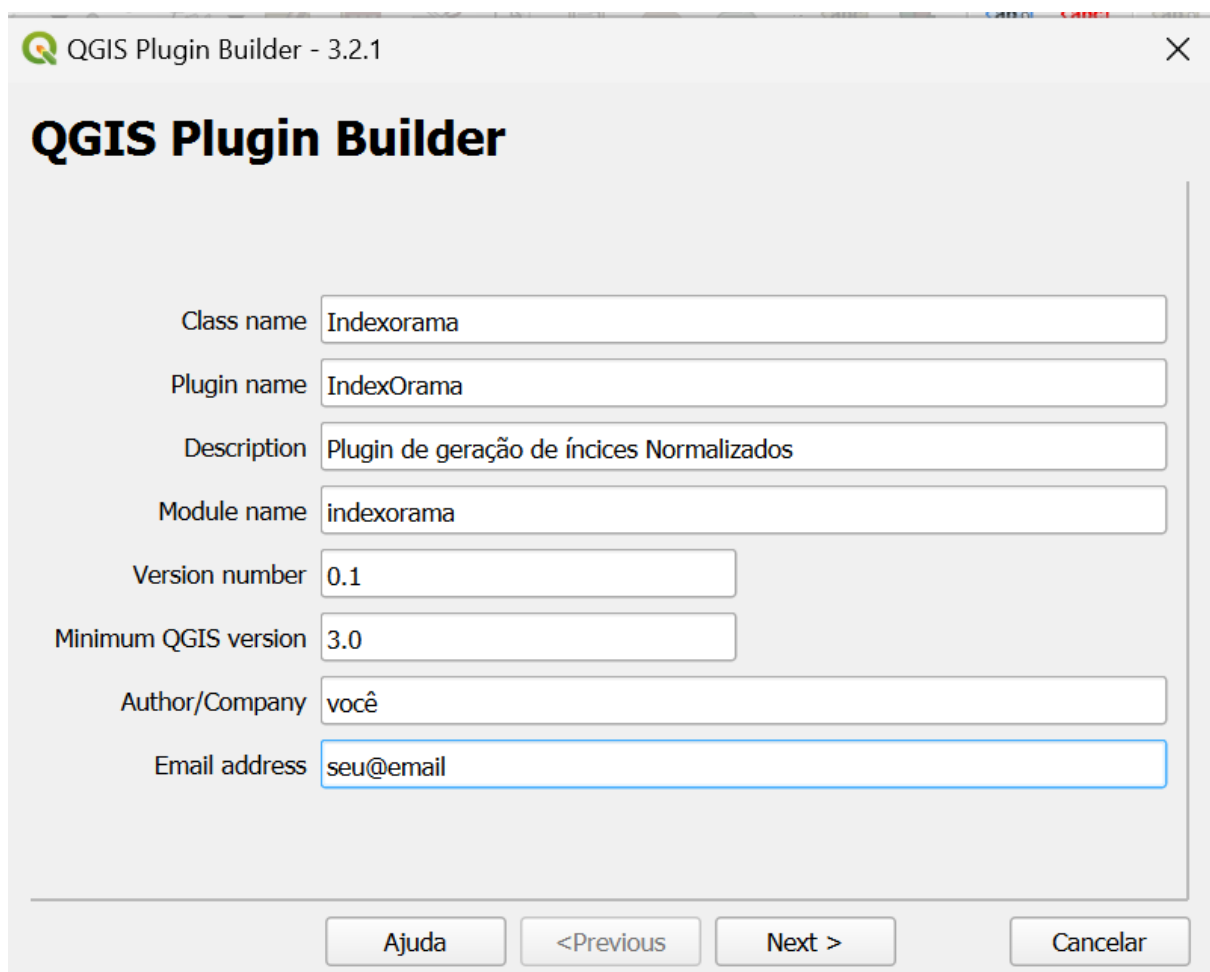
O plugin também gera uma composição RGB com os índices que é boa para a classificação do terreno.

2 - Construindo o esqueleto do IndexOrama no Plugin Builder 3

Inicie o Plugin Builder:



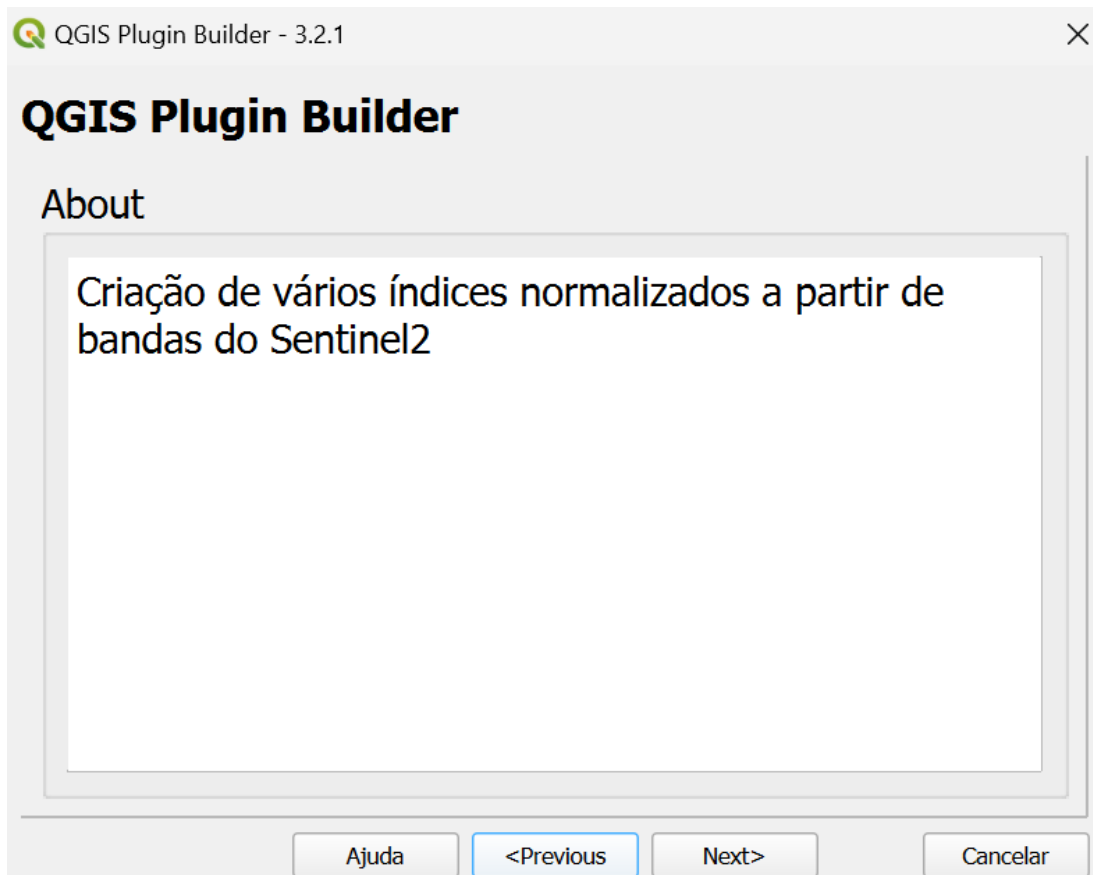
Preencha os campos dos formulários conforme as imagens a seguir:

A screenshot of the 'QGIS Plugin Builder - 3.2.1' dialog box. The title bar shows the QGIS logo and the text 'QGIS Plugin Builder - 3.2.1'. The main heading is 'QGIS Plugin Builder'. Below the heading are several input fields:

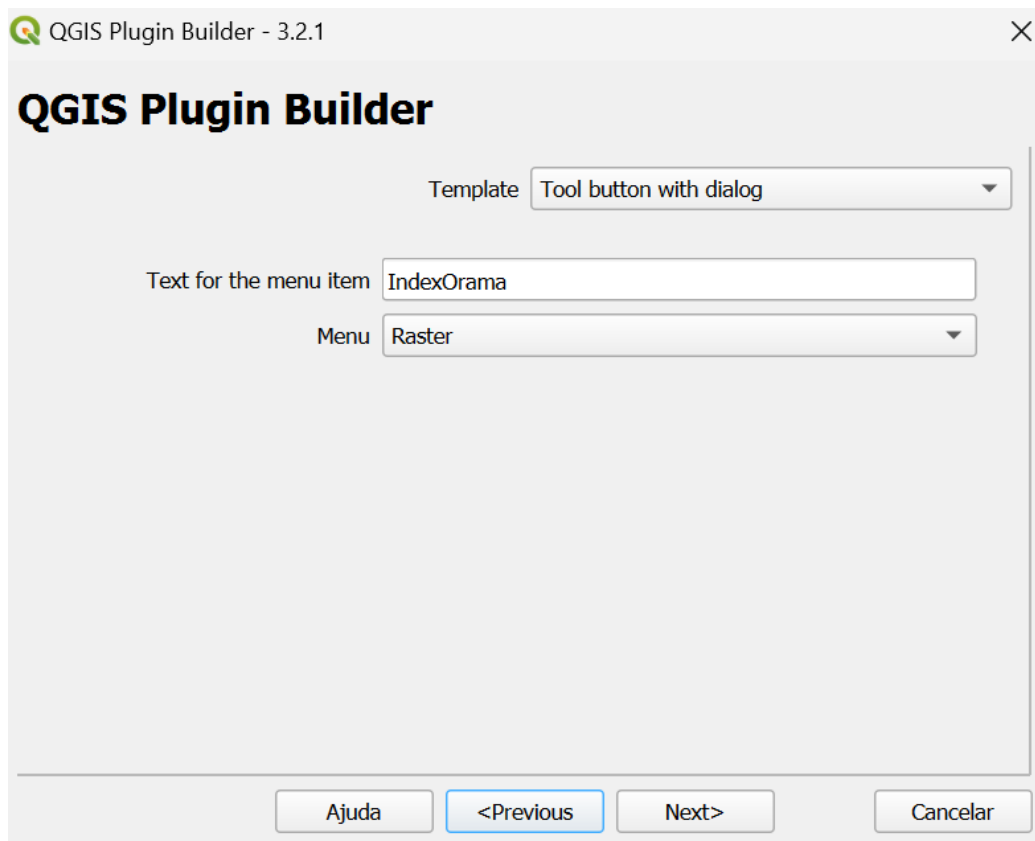
- Class name: Indexorama
- Plugin name: IndexOrama
- Description: Plugin de geração de índices Normalizados
- Module name: indexorama
- Version number: 0.1
- Minimum QGIS version: 3.0
- Author/Company: você
- Email address: seu@email

At the bottom of the dialog, there are four buttons: 'Ajuda', '<Previous', 'Next >', and 'Cancelar'.

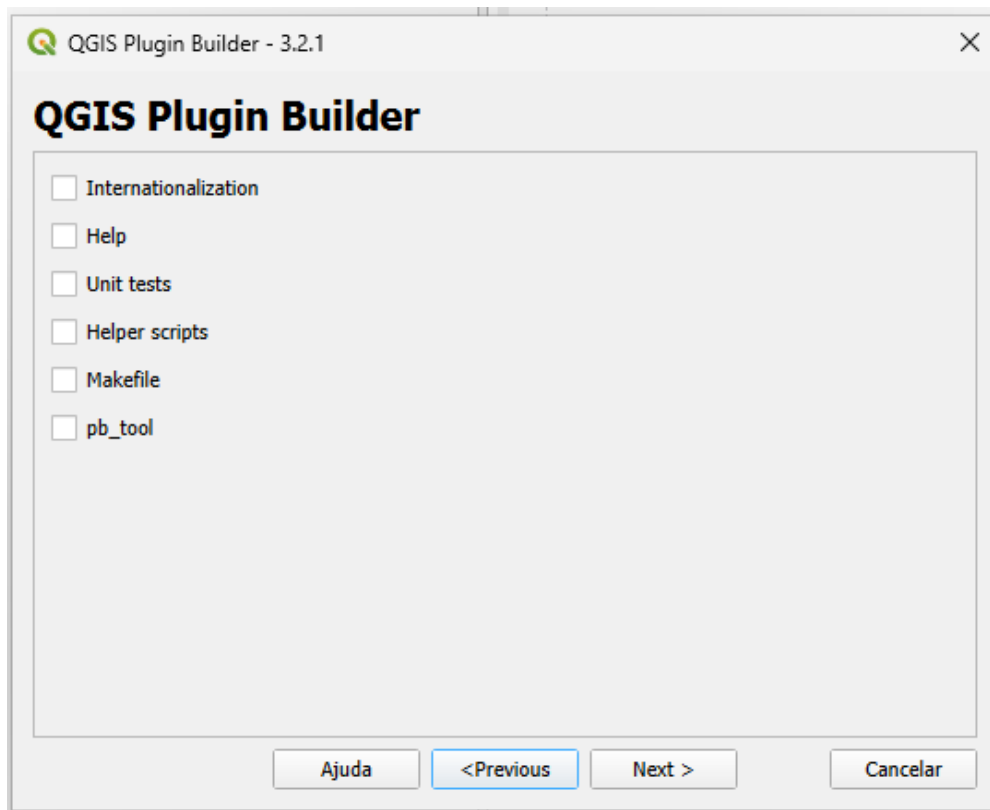
Esse primeiro formulário será usado na criação do arquivo metadata.txt e na definição do nome das classes do plugin.



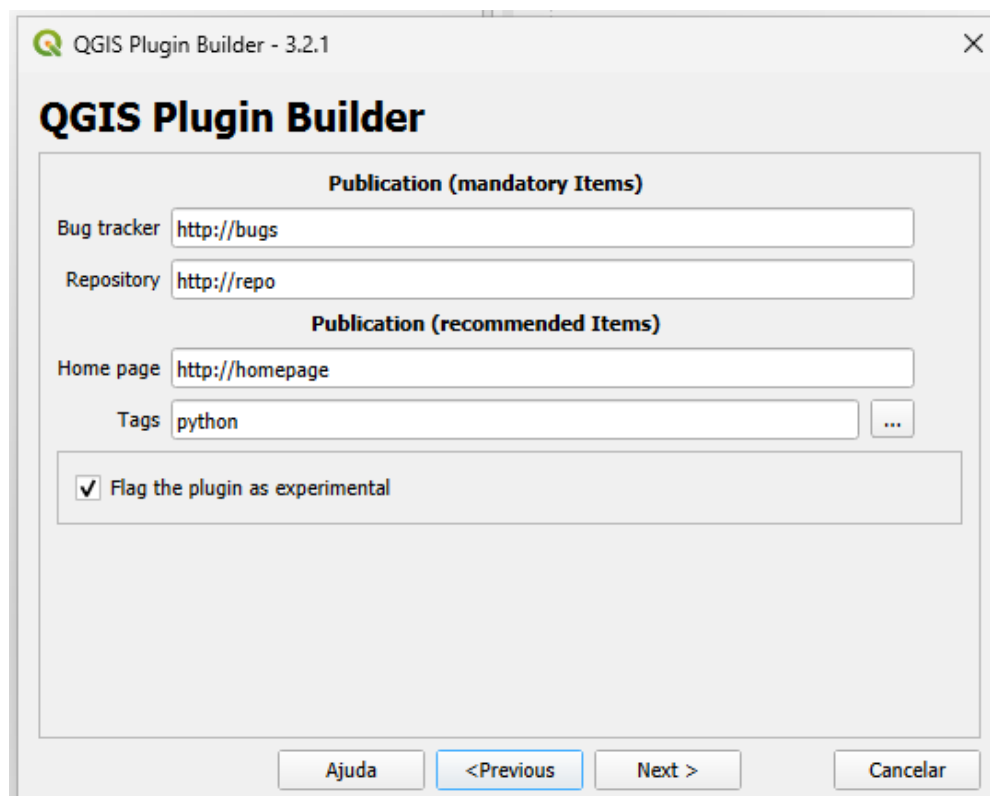
Descrição mais detalhada sobre o plugin que também será colocado no arquivo metadata.txt.



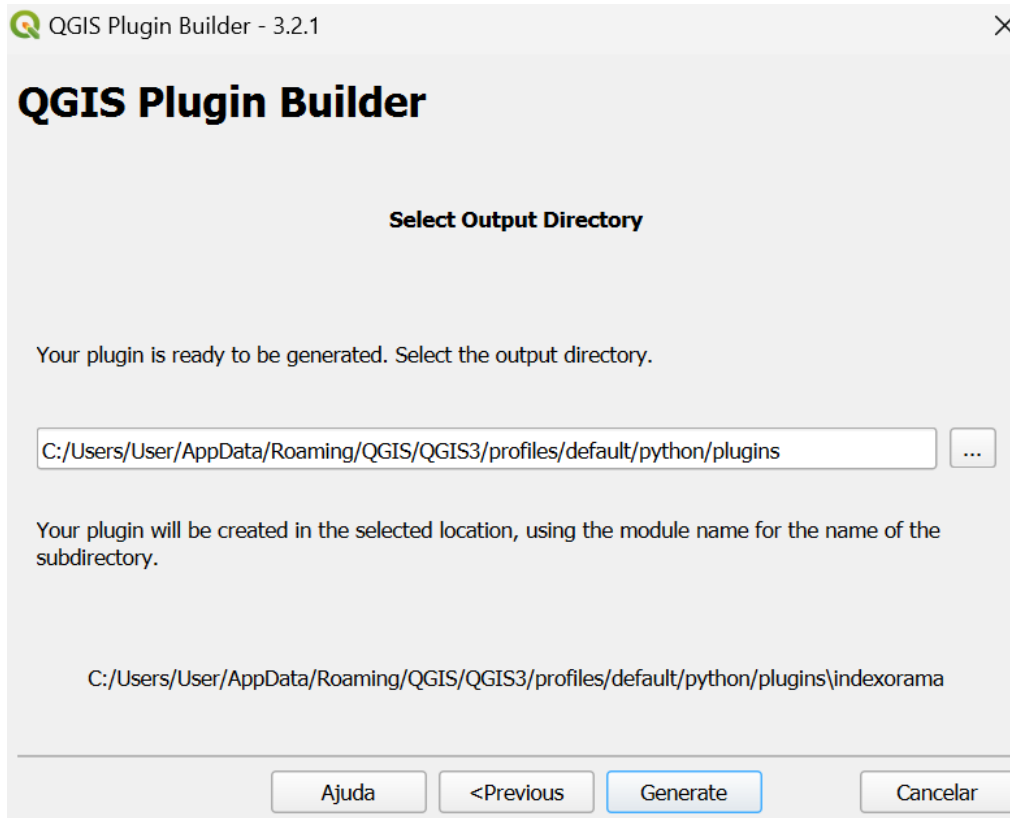
Template (tipo) do plugin, texto que vai aparecer no menu e em qual menu será listado o plugin.



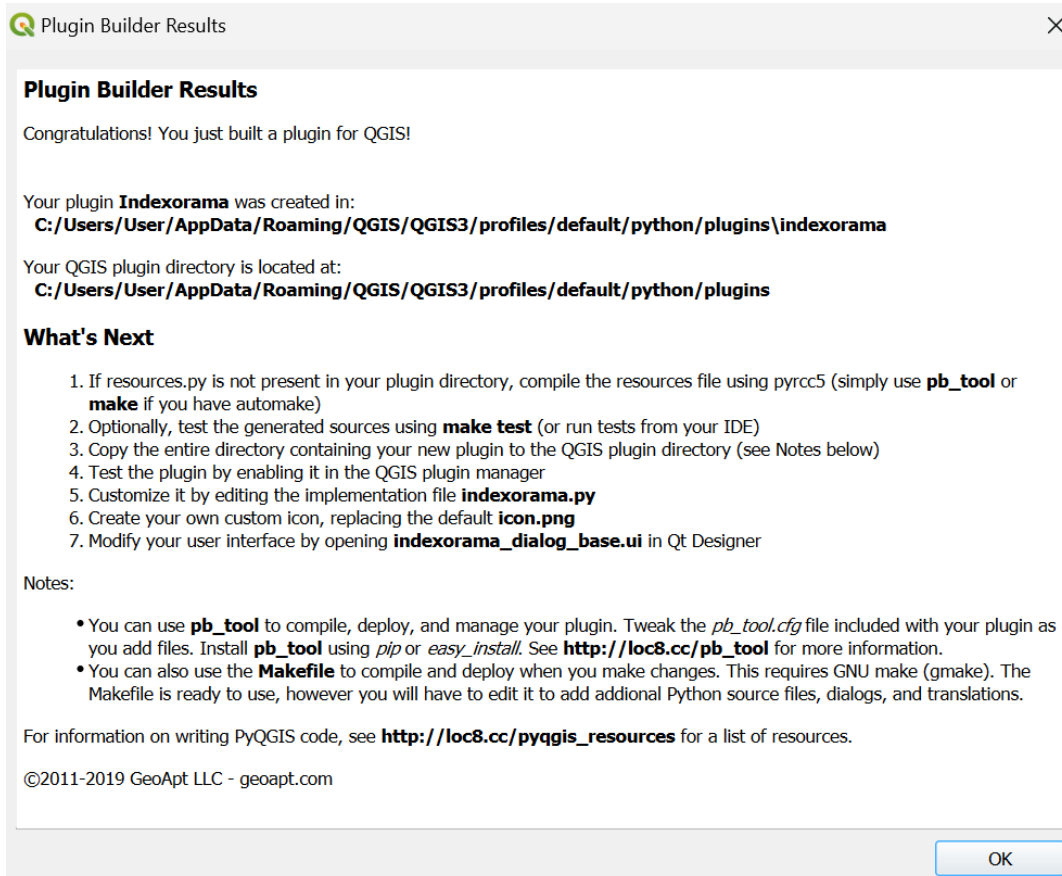
Desmarque todos para esse plugin,



Cheque a caixa de plugin experimental pois não iremos distribuir esse plugin no momento.



A pasta de plugin do sistema (nesse caso em sistema Windows). Clique **Generate** após selecionar o diretório de plugins.

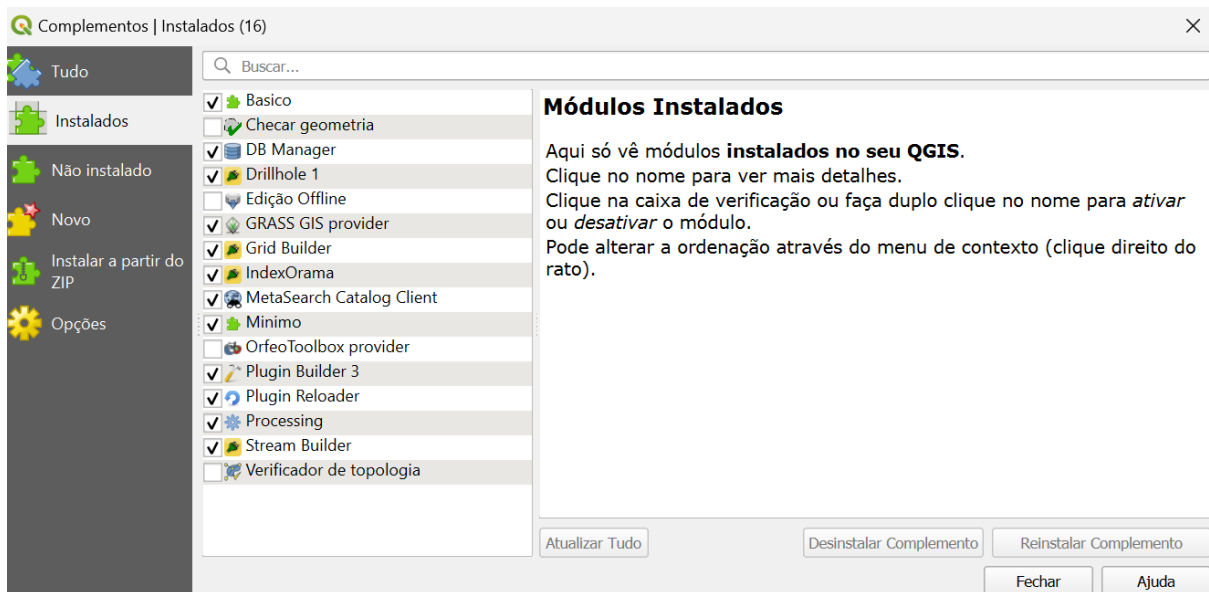


Pronto, os arquivos base de seu plugin foram criados na pasta:

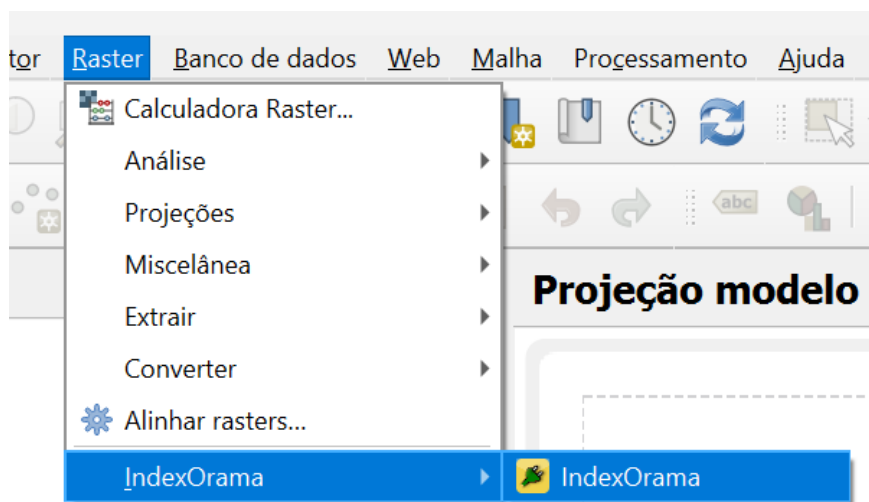
C:/Users/User/AppData/Roaming/QGIS/QGIS3/profiles/default/python/plugins\indexorama

Os oito arquivos necessários mais dois arquivos README com instruções do PluginBuilder foram criados automaticamente.

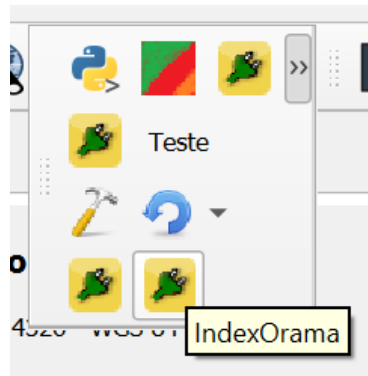
Vamos testar ele iniciando o QGIS e abrindo o **Complementos->Gerenciar e instalar Complementos**. Em Instalados vemos que ele não foi instalado ainda. Marque ele e instale para testarmos.



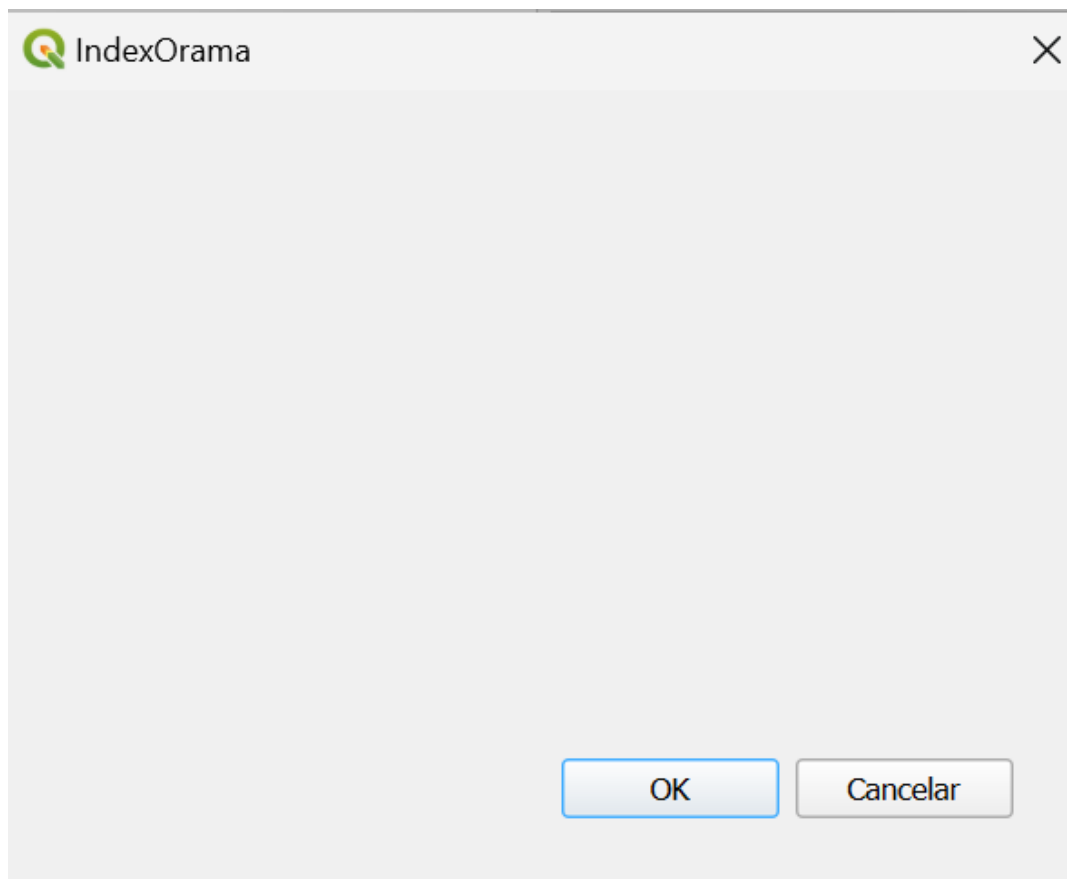
Inicie ele pelo Menu Vetor.



Ou pelo ícone na barra de ferramentas.



Funcionando, mas sem funcionalidade ainda.



Vamos construir a interface gráfica do usuário (GUI) e adicionar a funcionalidade agora na próxima seção.

3 - O plugin IndexOrama

A interface gráfica do plugin IndexOrama que tem como objetivo abrir 9 arquivos correspondentes a cada banda do sentinel2 e um widget para definir a pasta onde os arquivos de índices gerados serão gravados.

NOTA: As bandas usadas têm que estar todas na mesma resolução. Ou seja, todas em 20m resolução ou todas em 10m de resolução.

Os arquivos com as bandas estão disponíveis em <https://gdatasystems.com/pyqgis/index.php>

Agora execute o QtDesigner para criarmos a nossa interface gráfica de usuário (GUI).

Abra o arquivo **drillhole1_dialog_base.ui** localizado em:

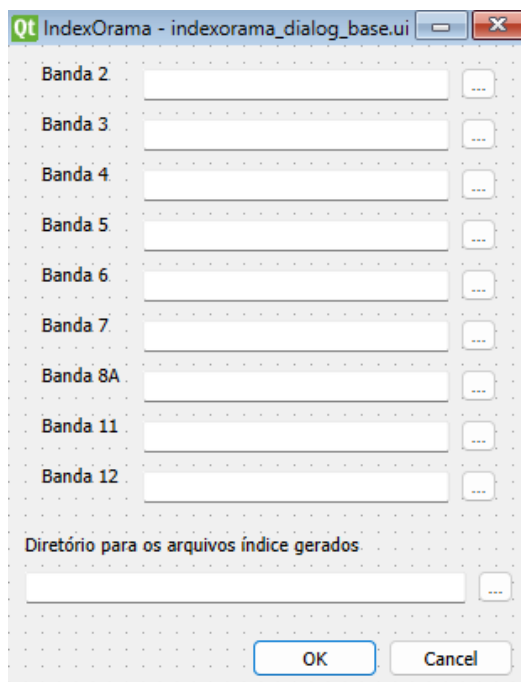
C:\Users\User\AppData\Roaming\QGIS\QGIS3\profiles\default\python\plugins\drillhole1

No primeiro diálogo do programa em **Open**.

Vamos adicionar 5 widgets do tipo Label, 5 widgets do tipo QgsFileWidget e 1 QgsProjectionSelectionWidget. Basta clicar no Widget e arrastar até a janela do diálogo.

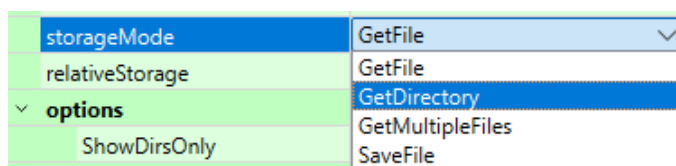


A aparência final da interface deve ser:



Modifique o texto dos campos Label.

No último QgsFileWidget altere o componente **StorageMode** para **GetDirectory**:



Agora altere a propriedade `objectName` dos campos `QgsFileWidget` na seguinte ordem.

mQgsFileWidget_1
mQgsFileWidget_2
mQgsFileWidget_3
mQgsFileWidget_4
mQgsFileWidget_5
mQgsFileWidget_6
mQgsFileWidget_7
mQgsFileWidget_8
mQgsFileWidget_9
mQgsFileWidgetDir

Pronto, salve o diálogo e feche o QtDesigner.

Vamos agora editar o arquivo **indexorama.py** para realizar a tarefa. Vamos ter de adicionar algumas bibliotecas de suporte via **import**.

As bibliotecas no arquivo **indexorama.py** serão (adicionar as faltantes):

```
from qgis.PyQt.QtCore import QSettings, QTranslator, QCoreApplication
from qgis.PyQt.QtGui import QIcon
from qgis.PyQt.QtWidgets import QAction, QMessageBox
from qgis.core import QgsProject, QgsRasterLayer
import processing
# Initialize Qt resources from file resources.py
from .resources import *
# Import the code for the dialog
from .indexorama_dialog import IndexoramaDialog
import os.path
import rasterio
import numpy as np
```

A função **run** ficará assim:

```
def run(self):
    if self.first_start == True:
        self.first_start = False
        self.dlg = IndexoramaDialog()

    # show the dialog
    self.dlg.show()
    # Run the dialog event loop
    result = self.dlg.exec_()
    # See if OK was pressed
    if result:
        b2=self.dlg.mQgsFileWidget_1.filePath()
        b3=self.dlg.mQgsFileWidget_2.filePath()
        b4=self.dlg.mQgsFileWidget_3.filePath()
        b5=self.dlg.mQgsFileWidget_4.filePath()
        b6=self.dlg.mQgsFileWidget_5.filePath()
        b7=self.dlg.mQgsFileWidget_6.filePath()
        b8a=self.dlg.mQgsFileWidget_7.filePath()
```

```

b11=self.dlg.mQgsFileWidget_8.filePath()
b12=self.dlg.mQgsFileWidget_9.filePath()
dire=self.dlg.mQgsFileWidgetDir.filePath()
if b2=="" or b3=="" or b4=="" or b5=="" or b6=="" or b7=="" or b8a=="
or b11=="" or b12=="" or dire=="":
    QMessageBox.warning(self.iface.mainWindow(),
        'Erro',
        "Entre todos os campos por favor\nSaindo...")
    return

band2=rasterio.open(b2)
band3=rasterio.open(b3)
band4=rasterio.open(b4)
band5=rasterio.open(b5)
band6=rasterio.open(b6)
band7=rasterio.open(b7)
band8a=rasterio.open(b8a)
band11=rasterio.open(b11)
band12=rasterio.open(b12)
band2_rgb = band2.profile
band2_rgb.update({"count": 9})
with rasterio.open(dire+"\\fullstack.tiff", 'w', **band2_rgb) as dest:
    dest.write(band2.read(1),1)
    dest.write(band3.read(1),2)
    dest.write(band4.read(1),3)
    dest.write(band5.read(1),4)
    dest.write(band6.read(1),5)
    dest.write(band7.read(1),6)
    dest.write(band8a.read(1),7)
    dest.write(band11.read(1),8)
    dest.write(band12.read(1),9)

stack=QgsRasterLayer(dire+"\\fullstack.tiff","stack")
QgsProject.instance().addMapLayer(stack)
output1 = dire+"\\ndvi.tiff"
processing.run("gdal:rastercalculator",
{'INPUT_A':stack,'BAND_A':7,'INPUT_B':stack,'BAND_B':3,'FORMULA':'((A-
B)/(A+B))','NO_DATA':None,'RTYPE':5,'OPTIONS':'','EXTRA':'','OUTPUT':output1})
output2 = dire+"\\ndvirel.tiff"
processing.run("gdal:rastercalculator",
{'INPUT_A':stack,'BAND_A':7,'INPUT_B':stack,'BAND_B':4,'FORMULA':'((A-
B)/(A+B))','NO_DATA':None,'RTYPE':5,'OPTIONS':'','EXTRA':'','OUTPUT':output2})
output3 = dire+"\\savi.tiff"
processing.run("gdal:rastercalculator",
{'INPUT_A':stack,'BAND_A':7,'INPUT_B':stack,'BAND_B':3,'FORMULA':'((A-
B)/(A+B+0.5)*1.5)','NO_DATA':None,'RTYPE':5,'OPTIONS':'','EXTRA':'','OUTPUT':output
3})
output4 = dire+"\\ndwi.tiff"
processing.run("gdal:rastercalculator",
{'INPUT_A':stack,'BAND_A':7,'INPUT_B':stack,'BAND_B':2,'FORMULA':'((B-
A)/(B+A))','NO_DATA':None,'RTYPE':5,'OPTIONS':'','EXTRA':'','OUTPUT':output4})
output5 = dire+"\\mndwi.tiff"
processing.run("gdal:rastercalculator",
{'INPUT_A':stack,'BAND_A':8,'INPUT_B':stack,'BAND_B':2,'FORMULA':'((B-
A)/(B+A))','NO_DATA':None,'RTYPE':5,'OPTIONS':'','EXTRA':'','OUTPUT':output5})
output6 = dire+"\\ndmi.tiff"
processing.run("gdal:rastercalculator",
{'INPUT_A':stack,'BAND_A':7,'INPUT_B':stack,'BAND_B':8,'FORMULA':'((A-
B)/(A+B))','NO_DATA':None,'RTYPE':5,'OPTIONS':'','EXTRA':'','OUTPUT':output6})
output7 = dire+"\\ndti.tiff"
processing.run("gdal:rastercalculator",
{'INPUT_A':stack,'BAND_A':8,'INPUT_B':stack,'BAND_B':9,'FORMULA':'((A-
B)/(A+B))','NO_DATA':None,'RTYPE':5,'OPTIONS':'','EXTRA':'','OUTPUT':output7})
output8 = dire+"\\ndbi.tiff"
processing.run("gdal:rastercalculator",
{'INPUT_A':stack,'BAND_A':7,'INPUT_B':stack,'BAND_B':8,'FORMULA':'((B-
A)/(B+A))','NO_DATA':None,'RTYPE':5,'OPTIONS':'','EXTRA':'','OUTPUT':output8})
QgsProject.instance().addMapLayer(QgsRasterLayer(output1,"ndvi"))

```

```

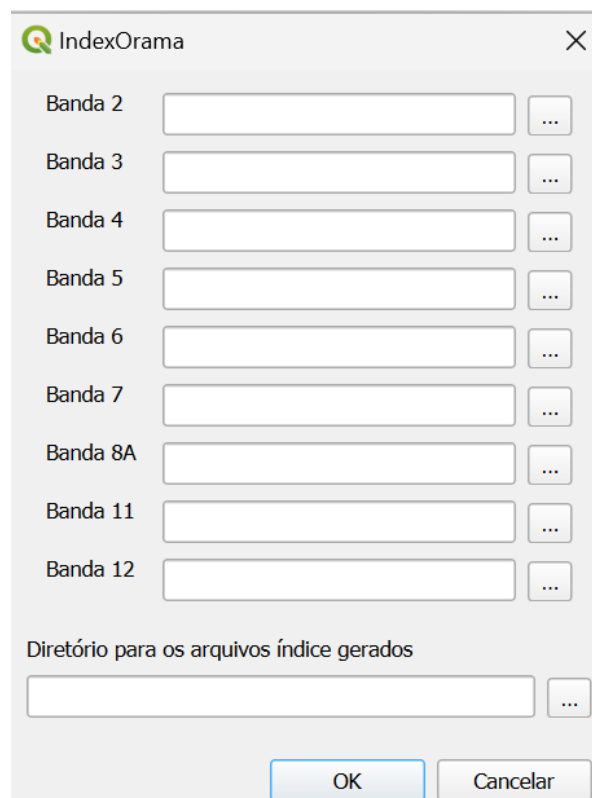
QgsProject.instance().addMapLayer(QgsRasterLayer(output2, "ndvire1"))
QgsProject.instance().addMapLayer(QgsRasterLayer(output3, "savi"))
QgsProject.instance().addMapLayer(QgsRasterLayer(output4, "ndwi"))
QgsProject.instance().addMapLayer(QgsRasterLayer(output5, "mndwi"))
QgsProject.instance().addMapLayer(QgsRasterLayer(output6, "ndmi"))
QgsProject.instance().addMapLayer(QgsRasterLayer(output7, "ndti"))
QgsProject.instance().addMapLayer(QgsRasterLayer(output8, "ndbi"))
NDBI=QgsRasterLayer(output8, "ndbi2")
NDTI=QgsRasterLayer(output7, "ndti2")
output9 = dire+"\\ndbi_ndti.tiff"
processing.run("gdal:rastercalculator",
{'INPUT_A':NDBI,'BAND_A':1,'INPUT_B':NDTI,'BAND_B':1,'FORMULA':'A+B','NO_DATA':None
,'RTYPE':5,'OPTIONS':'','EXTRA':'','OUTPUT':output9})
NDTI_NDBI=QgsRasterLayer(output9, "ndti_ndbi")
bandb=rasterio.open(output5)
bandg=rasterio.open(output3)
bandr=rasterio.open(output9)
band_rgb = bandr.profile
band_rgb.update({"count": 3})
with rasterio.open(dire+'\\classificado.tiff', 'w', **band_rgb) as
dest2:
    dest2.write(bandb.read(1),1)
    dest2.write(bandg.read(1),2)
    dest2.write(bandr.read(1),3)

classi=QgsRasterLayer(dire+"\\classificado.tiff", "classi")
QgsProject.instance().addMapLayer(classi)
QMessageBox.information(self.iface.mainWindow(), 'Pronto', 'IndexOrama
executado!')
return

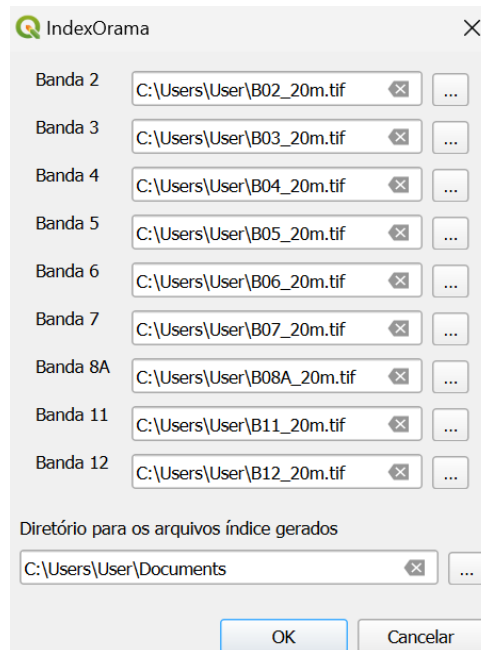
```

Baixe os arquivos com as bandas em <https://gdatasystems.com/pyqgis/index.php>

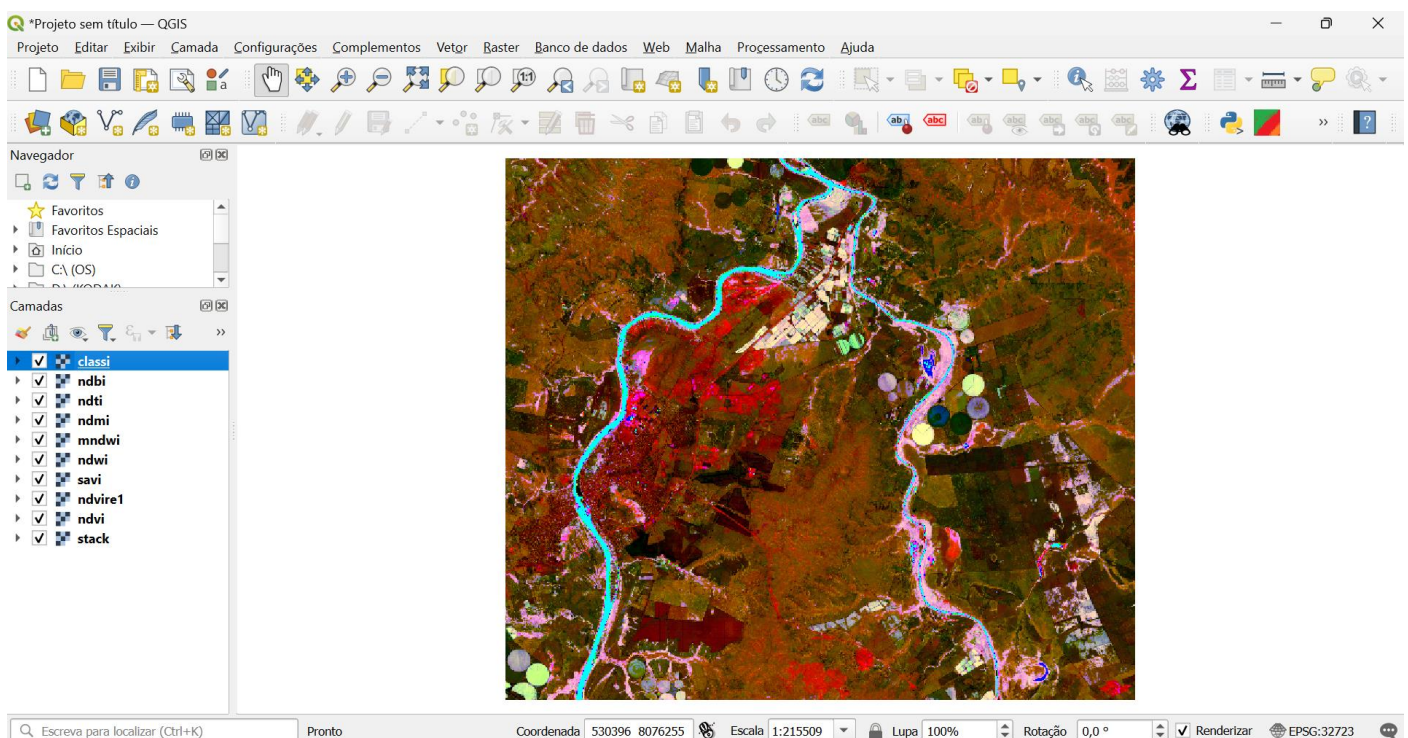
Abra o QGIS e o plugin será carregado já com as alterações feitas. Ao iniciarmos o plugin teremos:



Entre o caminho para os arquivos das bandas de acordo com título. Defina o diretório onde os arquivos gerados serão gravados.



Clique ok e aguarde o processamento dos dados e geração dos arquivos de camadas resultantes. Ao concluir teremos no QGIS 10 camadas raster criadas. Uma composição de 9 bandas chamada stack, 8 camadas de uma banda dos índices e uma composição RGB que usa os índices (MNDWI no azul, SAVI no verde e NDTI+NDBI no vermelho) para uma “Classificação” do terreno.

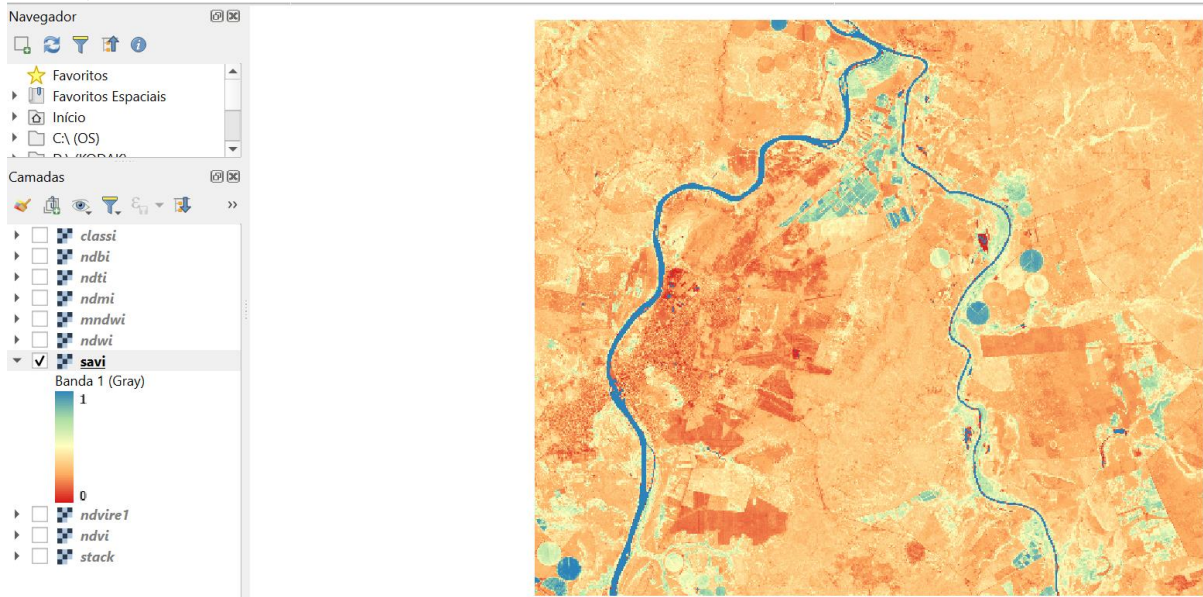


Os índices apresentam-se em escala de cinza e a faixa deve ser ajustada apropriadamente de 0 a 1 para melhor resultado. Podemos também usar falsa cor para melhorar a visualização.

SAVI gerado originalmente:



SAVI ajustado de 0 a 1 e usando falsa cor:



Finalizamos aqui os exemplos de plugins. Fique ligado para novos módulos futuros de plugins avançados.