



Começando a Programar com o Python no QGIS

André Silva

GEOeduc



CAPÍTULO 6:

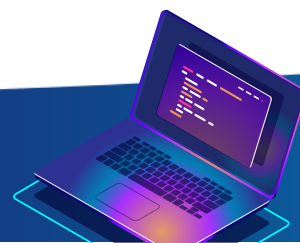
Criação de Plugins

CAPÍTULO 6: Criação de Plugins

Neste capítulo iremos abordar a criação de plugins, para uso interno na interface do QGIS Desktop. Estes plugins são ferramentas, similares a toolboxes do ArcGIS, aonde é possível criar novas capacidades de processamento e operações em geoprocessamento, porém fazendo uso de uma interface gráfica, em outras palavras, plugins (ou complementos) fazem a leitura de um script python e permite que o usuário tenha interação com o plugin não mais através de uma sintaxe ou código, o processo de interação com o script, é feito através de uma interface gráfica. Iremos verificar também que é possível usar plugins dentro da nossa instalação do QGIS, mesmo que este plugin não tenha sido feito por nós mesmo, ou seja o plugin pode ser adquirido através da internet ou de uma comunidade.

Objetivos do capítulo:

- Desenvolvimento de plugins
- Modelagem de ferramentas
- Divulgação para comunidade open source





EXERCÍCIO 1A: FUNCIONALIDADES BÁSICAS DO QGIS

Os plugins são uma excelente maneira de aumentar a funcionalidade do QGIS. Você pode desenvolver plugins usando Python, que podem variar de adicionar apenas um botão ou conjuntos de ferramentas complexas. Abaixo iremos desenvolver um plugin de média complexidade, para que possamos entender o processo de criação de plugins. Nos próximos exercícios abaixo, iremos criar um plugin que exporta os dados tabulares de uma camada para um arquivo .csv.



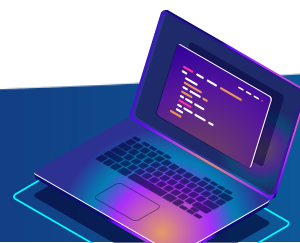
Passo 1 – Pré configuração

Iremos criar um plugin que será chamado de *exporta atributos seu nome*. Para este desenvolvimento iremos usar ferramentas adicionais como por exemplo o *Notepad++*, para que possamos fazer edições nos scripts criados pelo próprio plugin e também usaremos o *QT Designer*, que é um componente que já foi instalado junto com seu QGIS.



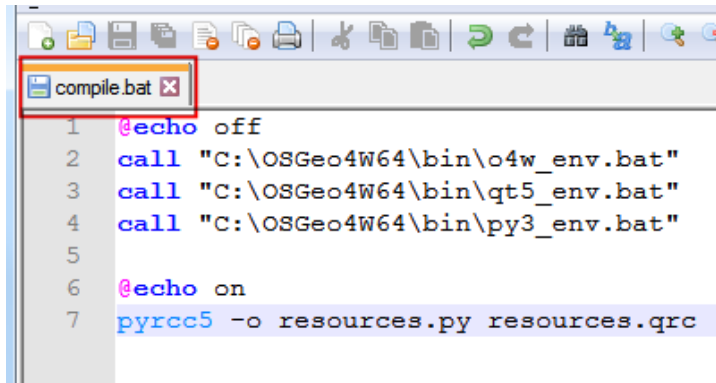
Dica: É possível que em outros tutoriais ou apostilas, você encontre como software necessário o *QT Creator*, porém o *QT Designer*, é uma versão mais leve que o *QT Creator*, e atende perfeitamente a criação de plugins. Caso deseje usar futuramente o *QT Creator*, poderá usar o seguinte link: <https://www.qt.io/offline-installers>

- 1) As conexões python necessárias estão incluídas na instalação do QGIS no Windows. Mas, para usá-los na pasta de plugins, precisamos indicar o caminho para a instalação do QGIS. Crie um arquivo de lote no Windows (extensão .bat) com o conteúdo abaixo e salve-o no seu computador com o nome de *compile.bat*. Posteriormente, copiaremos esse arquivo para a pasta do plugin. Comumente este é o caminho de instalação: C: \ OSGeo4W64\bin\ , porém é necessário encontrar o caminho correto em seu computador.



```
@echo off
call "C:\OSGeo4W64\bin\o4w_env.bat"
call "C:\OSGeo4W64\bin\qt5_env.bat"
call "C:\OSGeo4W64\bin\py3_env.bat"

@echo on
pyrcc5 -o resources.py resources.qrc
```



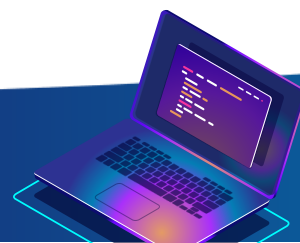
Agora iremos usar um plugin que já existe no QGIS, chamada *Plugin Builder*, necessário para criar todos os arquivos necessários e o código padrão para o nosso plugin a ser criado.

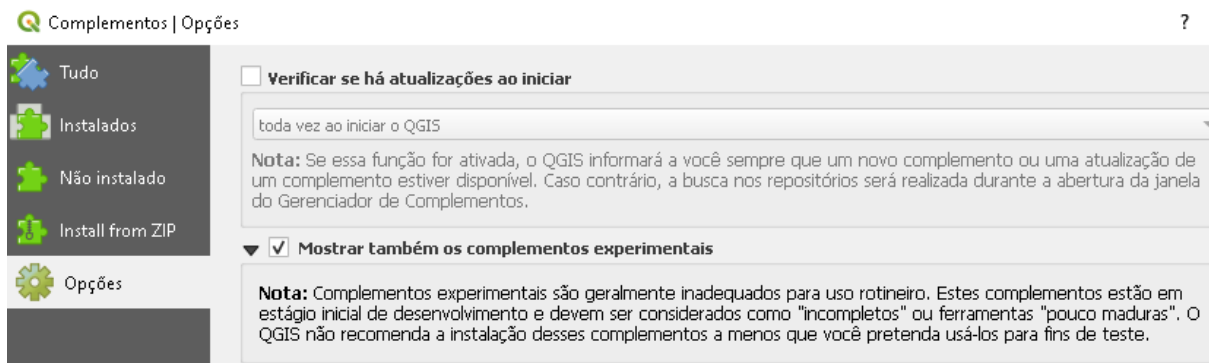
Outro plugin necessário é o *Plugins Reloader*, ele será responsável por atualizar nosso plugin, todas as vezes que realizarmos uma alteração em nosso protótipo.

Para instalar os dois vá até **Complementos > Gerenciar e instalar complementos > Tudo**, depois faça a pesquisa dos plugins necessários, escolhendo a opção **instalar complemento**.



Dica: Plugin reloader é um plugin experimental, para habilitar ele é necessário habilitar a opção *experimental plugins*, que pode ser encontrado em *Plugin Manager*. **Complementos > Gerenciar e instalar complementos > Opções**.





- 2) Vamos agora construir nosso plugin de modo propriamente dito. Abra o QGIS, depois vá até a guia **Complementos > Plugin Builder > Plugin Builder**. Uma caixa será aberta, siga até o próximo passo.
- 3) Um formulário aparecerá, então você deverá preencher com as seguintes informações abaixo, e após isso clique em Next:

Class name: exporta atributos <seu nome>

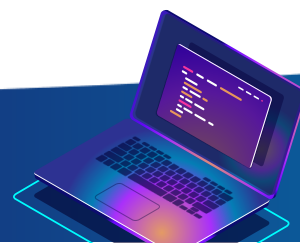
Plugin name: exporta atributos <seu nome>

Description: Este plugin salva os atributos de um vetor com um arquivo CSV.

Module name: exporta atributos <seu nome>

Author/Company: <seu nome>

Email address:<seu e-mail>



QGIS Plugin Builder - 3.1

QGIS Plugin Builder

Class name

Plugin name

Description

Module name

Version number

Minimum QGIS version

Author/Company

Email address

Help <Previous **Next >** Cancel

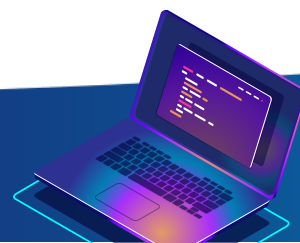
4) Na próxima tela insira uma breve descrição do plugin e então clique em Next.

QGIS Plugin Builder - 3.1

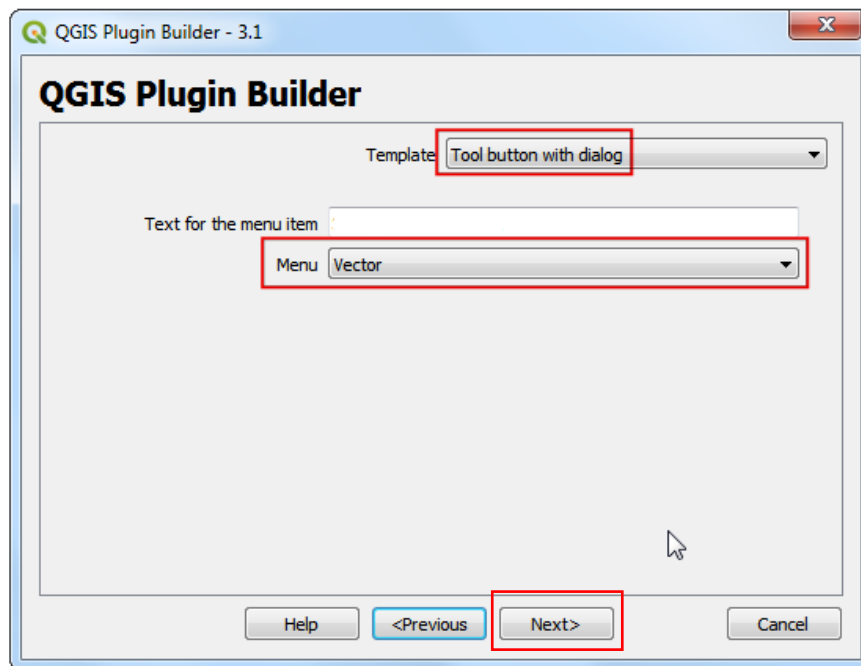
QGIS Plugin Builder

About

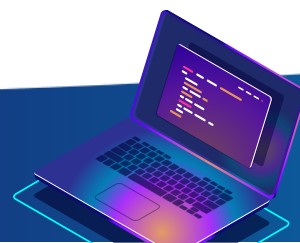
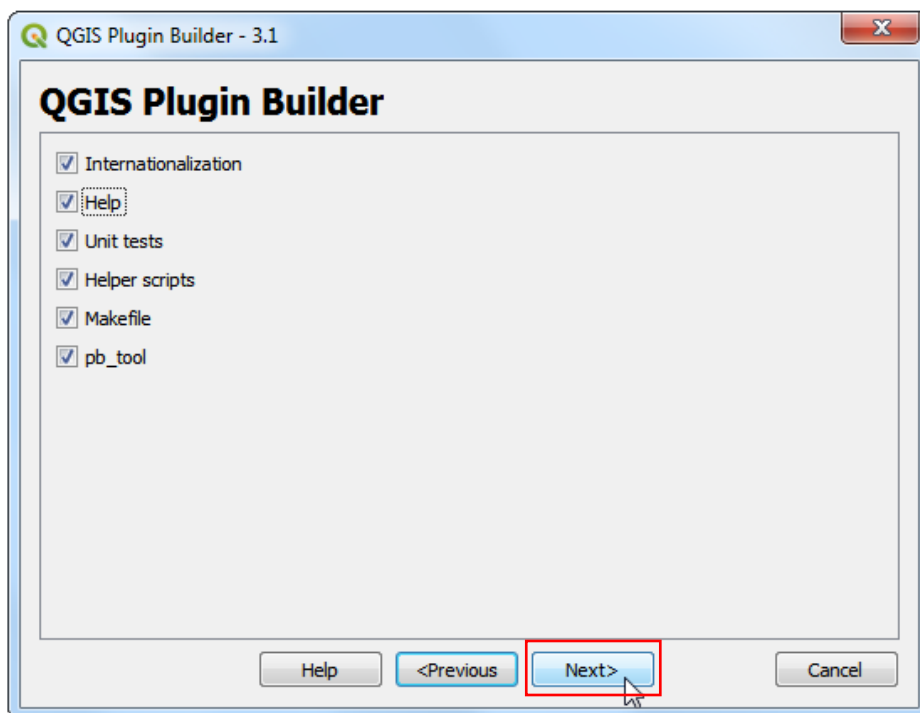
Help <Previous **Next >** Cancel



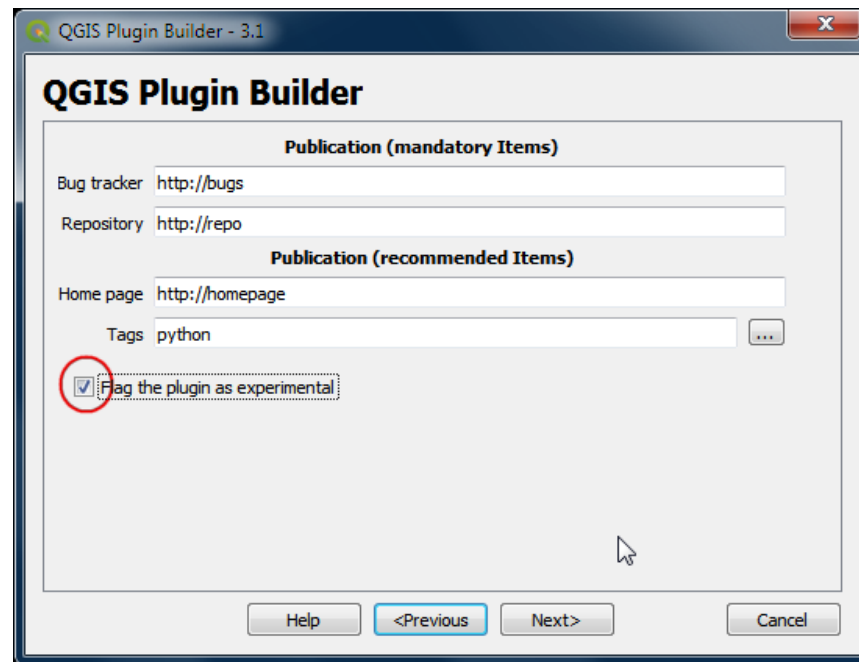
- 5) Em seguida selecione para o campo *Template*: *Tool button with dialog*, para *Text for the menu item*: <uma rápida descrição do que o plugin faz>.



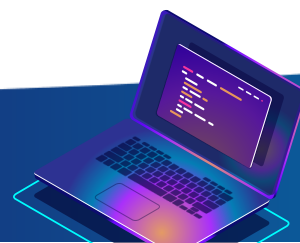
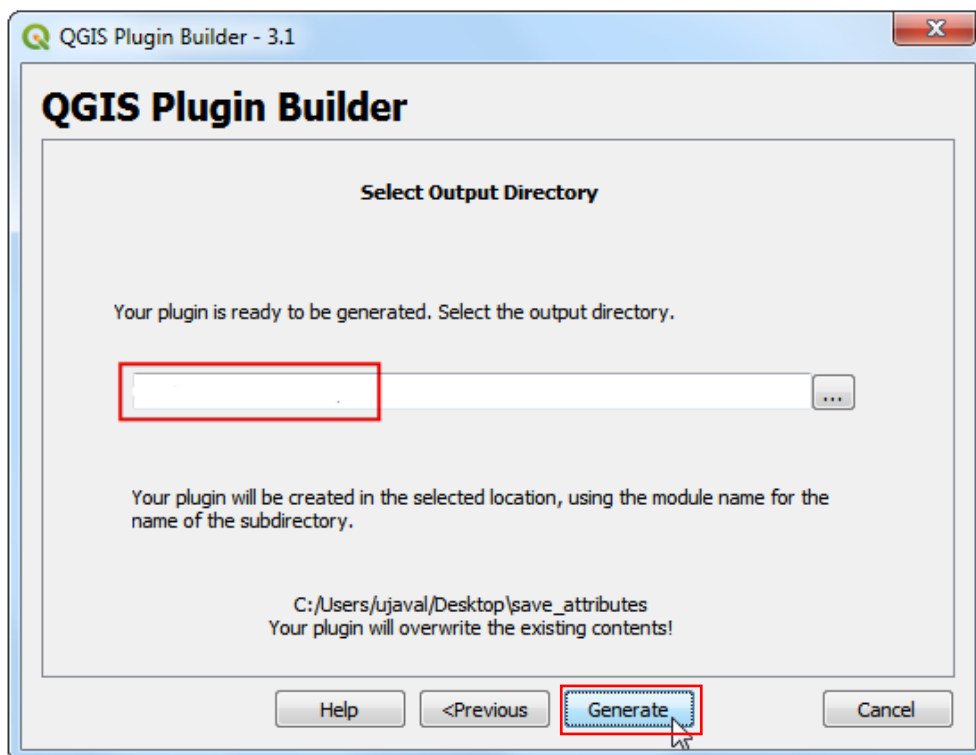
- 6) O plugin builder irá perguntar na nova tela, quais os tipos de arquivos deverão ser gerados. Mantenha a seleção padrão e clique em Next.



- 7) Como não será obrigatório a publicação do plugin, você pode deixar os valores de Bug tracker e os outros valores como estão, você apenas precisa deixar selecionado o *Flag the plugin as experimental*.



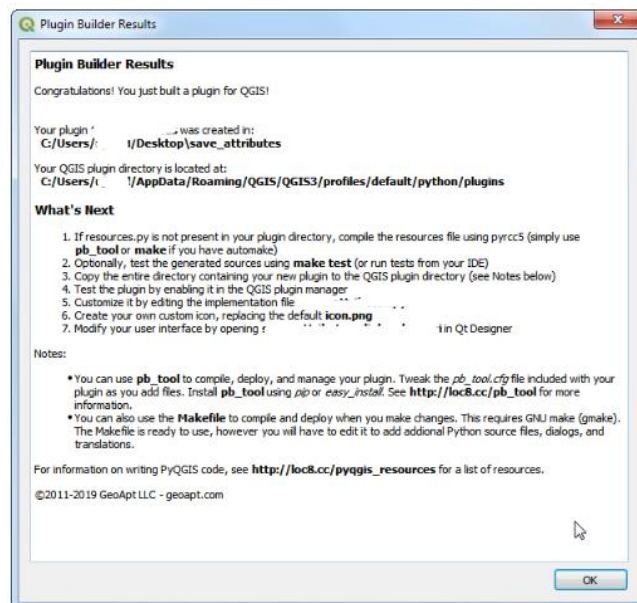
- 8) Haverá uma pergunta sobre aonde o plugin será gerado, por este momento escolha um diretório fácil de ser localizado e logo em seguida clique em Generate.





Dica: Ignore a mensagem que o pyrc5 não foi encontrado.

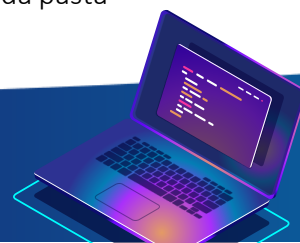
9) A tela de resultados aparecerá, apenas clique em OK:



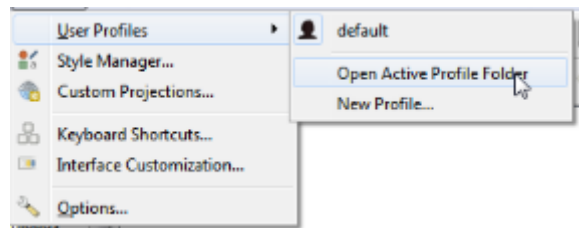
Passo 2 – Customização da interface e alteração de código

Para podermos usar o plugin recém-criado, precisamos compilar o arquivo `resources.qrc` que foi criado pelo Plugin Builder. Este arquivo faz parte do Qt Resource System, que faz referência a todos os arquivos binários usados pelo plugin. Para este plugin, ele terá apenas o ícone do plugin. A compilação desse arquivo gera o código do aplicativo que pode ser usado no plugin independentemente da plataforma em que o plug-in está sendo executado. Siga as instruções específicas da plataforma para esta etapa.

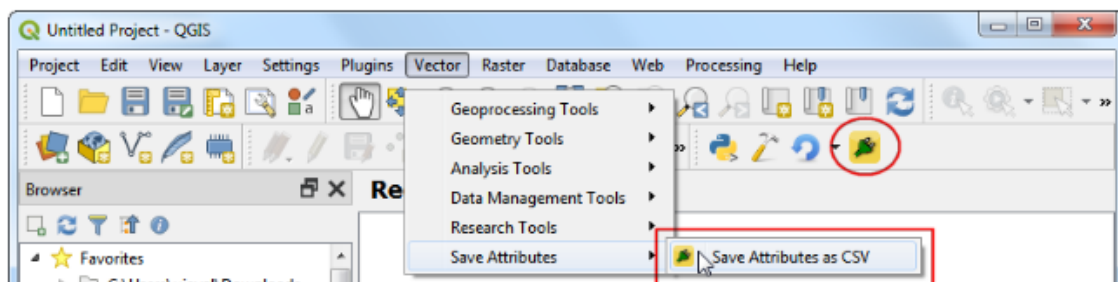
- 1) Agora você pode copiar o arquivo `compile.bat`, criado no início, na pasta aonde foi criado o plugin. Depois de copiado, clique duas vezes no arquivo para executá-lo. Se tudo correr bem, você verá um novo arquivo, chamado `resources.py`.
- 2) Os plugins dentro do QGIS, ficam armazenados em uma pasta diferenciada. Antes de usar o nosso plugin em criação, devemos copiar a pasta do plugin, para dentro da pasta



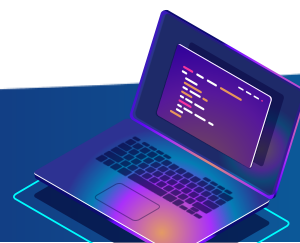
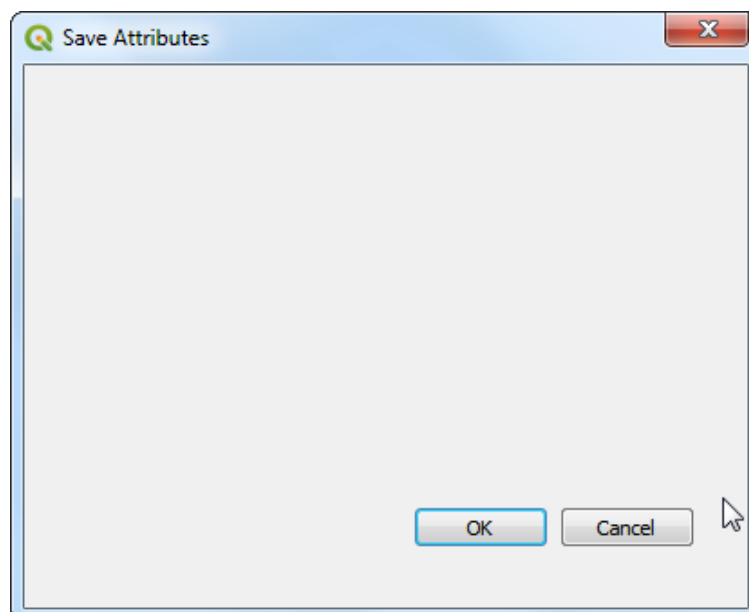
aonde o QGIS armazena seus plugins instalados. Para verificar esta pasta, vá até **Configurações > User Profiles > Open Active Profile Folder**.



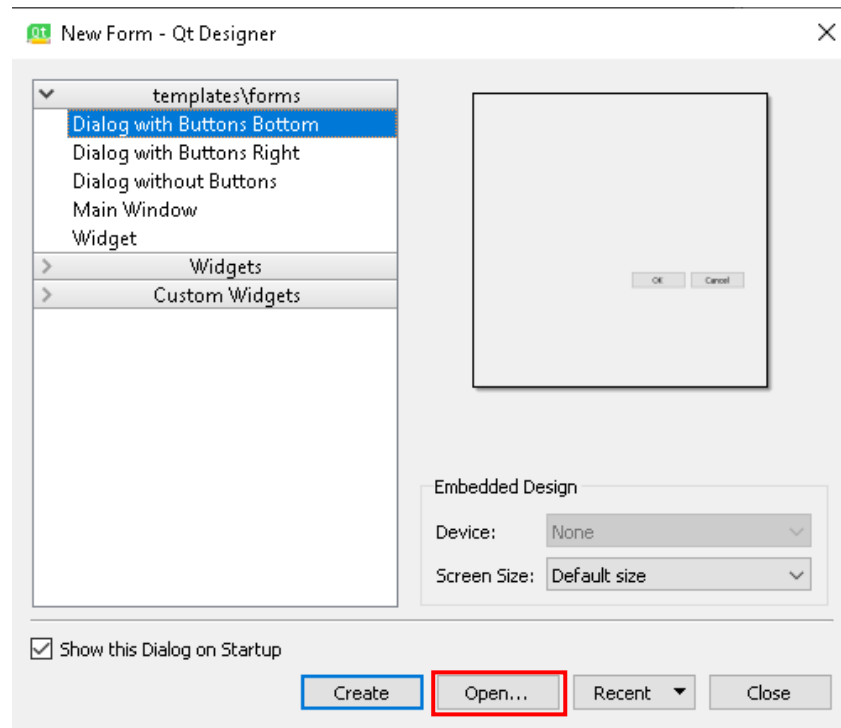
- 3) Ao abrir a pasta Profile, copie o plugin para **python > plugins**.
- 4) Feche o QGIS e abra-o novamente, vá até **Complementos > Gerenciar e instalar complementos > Instalados**, você verá o plugin criado por você, ative-o.
- 5) Você notará que um novo plugin irá aparecer em sua barra de ferramentas e também dentro da guia Vetor, ao localizá-lo, abra seu plugin.



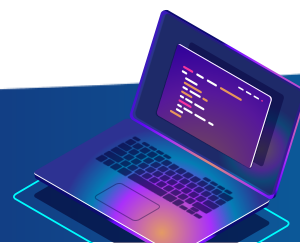
- 6) Verifique que seu plugin, está praticamente em branco. Feche a caixa de diálogo.

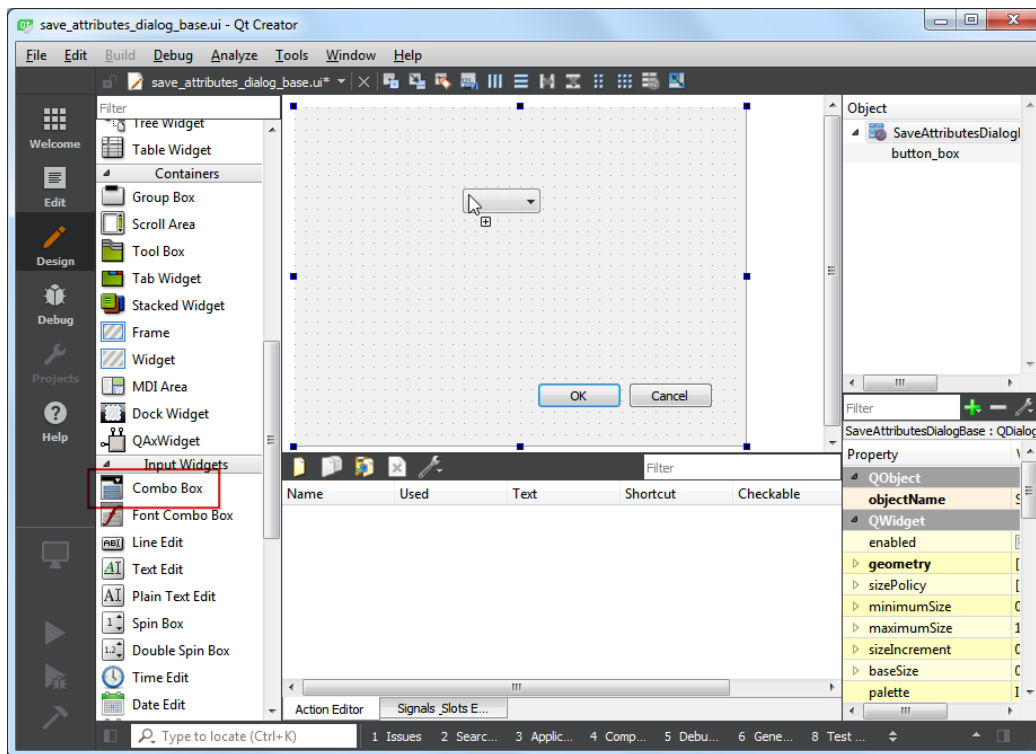


- 7) Agora será necessário inserir alguns elementos, para interação com o usuário. Abra o QT Designer e clique em Open.

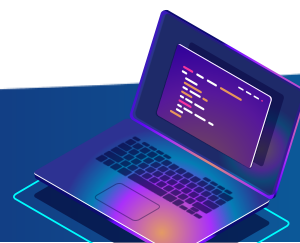
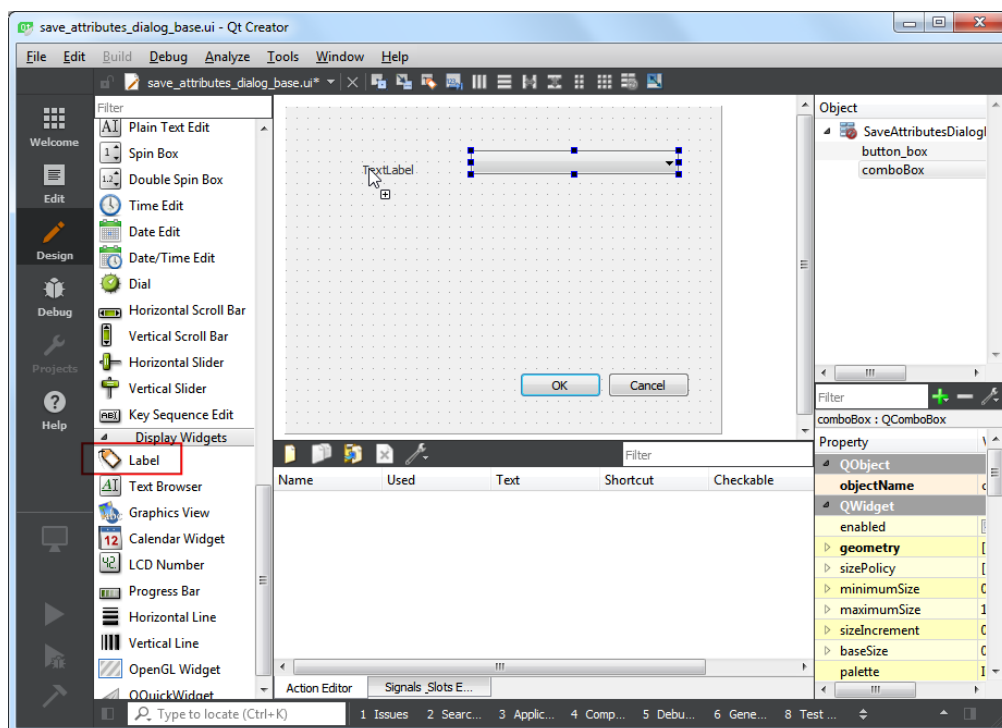


- 8) Navegue para o diretório do plugin e selecione o arquivo *exportar atributos<seu nome>.ui*. E depois clique em Abrir.
- 9) Você verá a caixa de diálogo em branco do plugin. Você pode arrastar e soltar elementos no painel esquerdo da caixa de diálogo. Adicionaremos um tipo de caixa de combinação de widgets de entrada. Arraste-o para a caixa de diálogo do plugin.

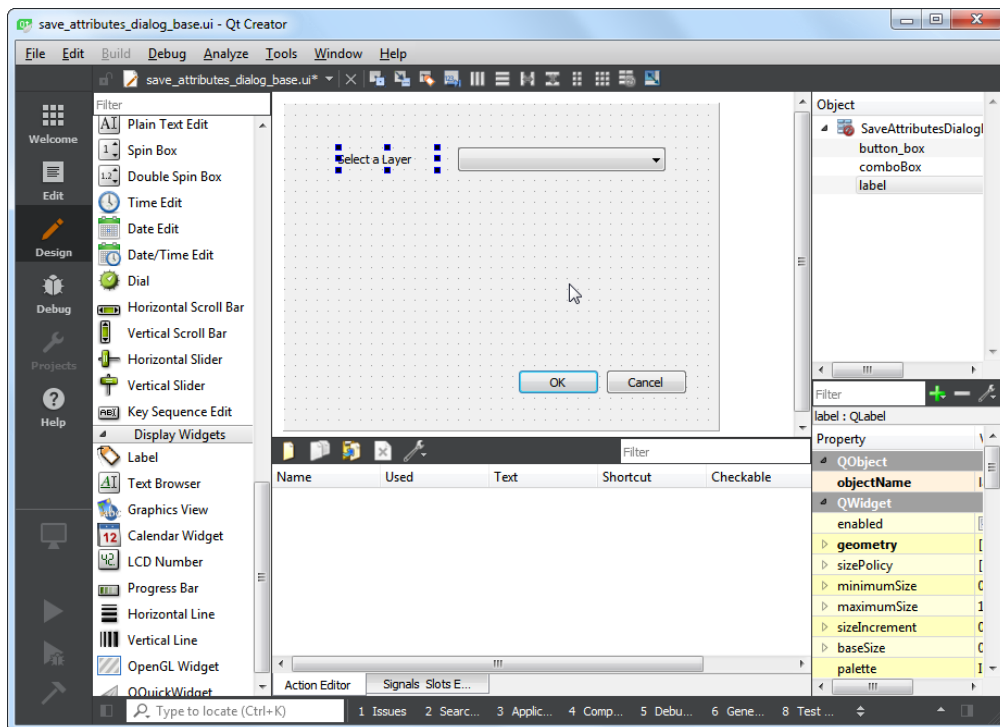




- 10) Redimensione o combo box e ajuste seu tamanho. Agora arraste um widget de exibição do tipo Label na caixa de diálogo.



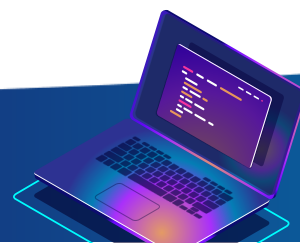
11) Clique no texto do Label e escreva Select a layer.

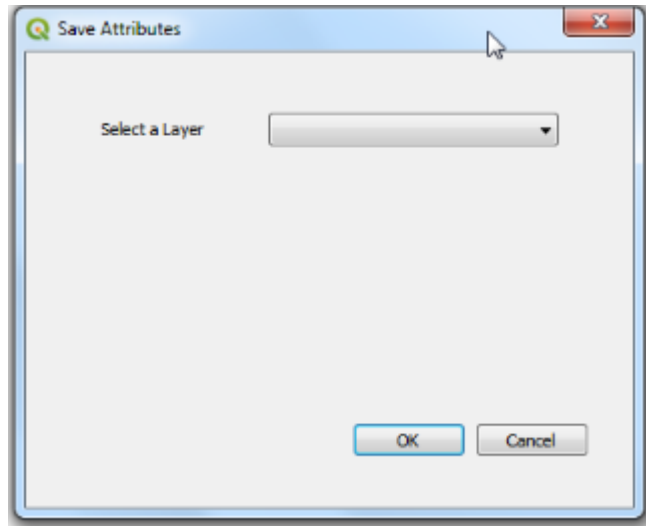


12) Salve o projeto, clicando em **File > Save**. Observe que o nome do objeto da caixa de combinação é comboBox. Para interagir com esse objeto usando o código python, teremos que nos referir a ele com esse nome.

13) Vá para o QGIS, acione o complemento *Plugin Reloader*, e depois escolha o plugin que você acabou de criar. Lembre-se *Plugin Reloader*, atualiza as alterações no plugin criado.

14) Clique no botão *Plugin Reloader* para carregar a versão mais recente do plugin. Clique no ícone referente ao seu plugin, para abrir a caixa de diálogo.





15) Vamos adicionar alguma lógica ao plugin que preencherá a caixa de combinação com as camadas carregadas no QGIS. Vá para o diretório do plugin e carregue o arquivo *exportar atributos <seu nome>.py* em um editor de texto. Primeiro, insira na parte superior do arquivo com as outras importações:

```
from qgis.core import QgsProject
```

16) Em seguida, desça até o final e encontre o método *run (self)*. Este método será chamado quando você clicar no botão da barra de ferramentas ou selecionar o item de menu do plugin. Adicione o seguinte código no início do método. Esse código obtém as camadas carregadas no QGIS e o adiciona ao objeto comboBox na caixa de diálogo do plugin.

```
# Obtém as camadas carregadas no QGIS
```

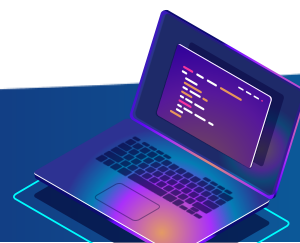
```
layers = QgsProject.instance().layerTreeRoot().children()
```

```
# Limpa o conteúdo do combo box das execuções anteriores
```

```
self.dlg.comboBox.clear()
```

```
# Preenche o combo box com os layers carregados.
```

```
self.dlg.comboBox.addItem([layer.name() for layer in layers])
```



```

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 PyQt5.QtCore import QSettings, QTranslator, qVersion, QCoreApplication
25  from PyQt5.QtGui import QIcon
26  from PyQt5.QtWidgets import QAction
27  from qgis.core import QgsProject
28

```

```

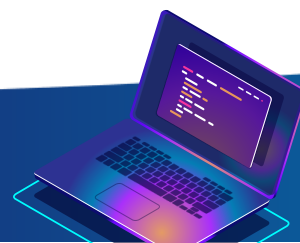
185
186  def run(self):
187      """Run method that performs all the real work"""
188
189      # Create the dialog with elements (after translation) and keep
190      # reference
191      # Only create GUI ONCE in callback, so that it will only load when
192      # the plugin is started
193      if self.first_start == True:
194          self.first_start = False
195          self.dlg = SaveAttributesDialog()
196
197      # Fetch the currently loaded layers
198      layers = QgsProject.instance().layerTreeRoot().children()
199      # Clear the contents of the comboBox from previous runs
200      self.dlg.comboBox.clear()
201      # Populate the comboBox with names of all the loaded layers
202      self.dlg.comboBox.addItem([layer.name() for layer in layers])
203
204      # show the dialog

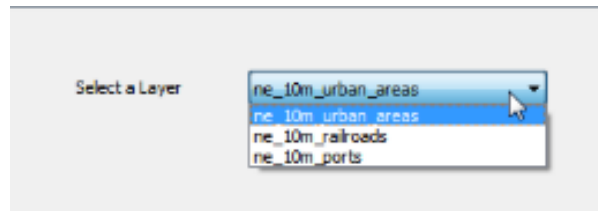
```



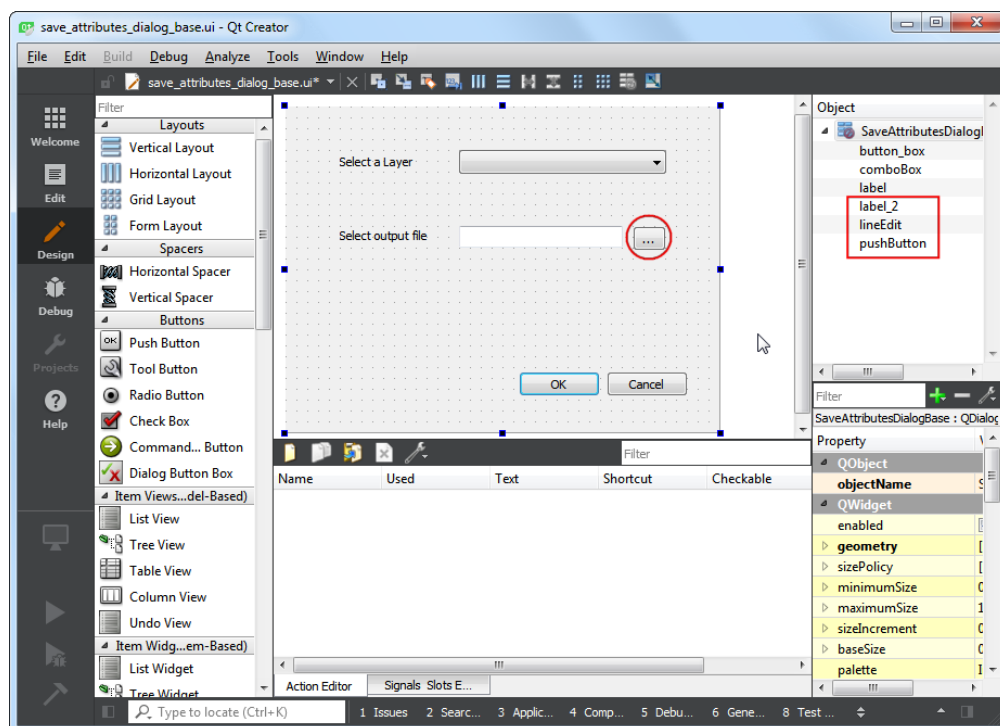
Dica: O script acima, possivelmente foi indentado usando espaços e não a tecla TAB, ao inserir novos trechos de código faça o ajuste da indentação usando espaços também.

17) De volta à janela principal do QGIS, recarregue o plugin clicando no botão *Plugin Reloader*. Para testar a nova funcionalidade, precisamos carregar algumas camadas no QGIS. Depois de carregar algumas camadas, inicie o plugin clicando no ícone do seu plugin. Você verá que nossa caixa de combinação agora está preenchida com os nomes das camadas carregadas no QGIS.

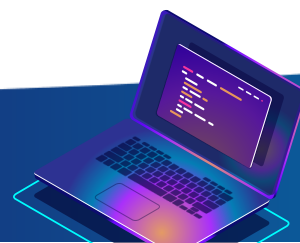




- 18) Necessário adicionar os elementos restantes para interface com o usuário. Volte para o *Qt Designer* e carregue o arquivo *exportar atributos <seu nome>.ui*. Adicione um widget de exibição *Label* e altere o texto para *Select output file*. Adicione um widget de entrada do tipo *LineEdit* que mostrará o caminho do arquivo de saída que o usuário escolheu. Em seguida, adicione um *Push Button* e altere o rótulo do botão para Observe os nomes dos objetos dos widgets que precisaremos usar para interagir com eles. Ao final salve o arquivo.



- 19) Agora, adicionaremos código python para abrir um navegador de arquivos quando o usuário clicar no botão ... e mostrar o caminho de seleção no widget de edição de linha. Abra o arquivo *exportar atributos <seu nome>.py* em um editor de texto. Adicione *QFileDialog* à lista de importações *QtWidgets* na parte superior do arquivo.



```

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 PyQt5.QtCore import QSettings, QTranslator, qVersion, QCoreApplication
25  from PyQt5.QtGui import QIcon
26  from PyQt5.QtWidgets import QAction, QFileDialog
27  from qgis.core import QgsProject

```

20) Adicione um novo método chamado *select_output_file* com o seguinte código. Esse código abrirá um navegador de arquivos e preencherá o widget de edição de linha com o caminho do arquivo que o usuário escolheu. Observe como *getSaveFileName* retorna uma tupla com o nome do arquivo e o filtro usado.

```

def select_output_file(self):
    filename, _filter = QFileDialog.getSaveFileName(
        self.dlg, "Select output file","", '*.csv')
    self.dlg.lineEdit.setText(filename)

```

```

177  def unload(self):
178      """Removes the plugin menu item and icon from QGIS GUI."""
179      for action in self.actions:
180          self.iface.removePluginVectorMenu(
181              self.tr(u'Save Attributes'),
182              action)
183          self.iface.removeToolBarIcon(action)
184
185  def select_output_file(self):
186      filename, _filter = QFileDialog.getSaveFileName(
187          self.dlg, "Select output file","", '*.csv')
188      self.dlg.lineEdit.setText(filename)
189
190  def run(self):
191      """Run method that performs all the real work"""
192

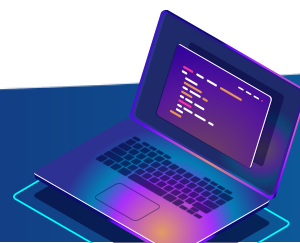
```

21) Agora precisamos adicionar código para que, quando o botão ... for clicado, o método *select_output_file* seja chamado. Role para baixo até o método *run* e adicione a seguinte linha no bloco em que a caixa de diálogo é inicializada. Este código conectará o método *select_output_file* para o *clicked* do widget do Push Button.

```

self.dlg.pushButton.clicked.connect(self.select_output_file)

```



```

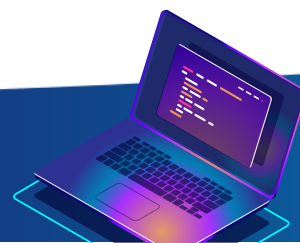
185 def select_output_file(self):
186     filename, _filter = QFileDialog.getSaveFileName(
187         self.dlg, "Select output file ", "", '*.csv')
188     self.dlg.lineEdit.setText(filename)
189
190 def run(self):
191     """Run method that performs all the real work"""
192
193     # Create the dialog with elements (after translation) and keep
194     # reference
195     # Only create GUI ONCE in callback, so that it will only load when
196     # the plugin is started
197     if self.first_start == True:
198         self.first_start = False
199         self.dlg = SaveAttributesDialog()
200         self.dlg.pushButton.clicked.connect(self.select_output_file)

```

22) Volte ao QGIS, recarregue(Plugin Reloader) o plugin e execute-o. Se tudo der certo, você poderá clicar no botão ... e selecionar o local e o nome do arquivo .csv a ser gravado.



23) Quando você clica em OK na caixa de diálogo do plugin, nada acontecerá. Isso ocorre porque não adicionamos a lógica para extrair informações de atributos da camada e gravá-las no arquivo de texto. Agora temos todas as peças para fazer exatamente isso. Encontre o local no método *run* onde diz *pass*. Substitua-o pelo código abaixo.



```

filename = self.dlg.lineEdit.text()
with open(filename, 'w') as output_file:
    selectedLayerIndex = self.dlg.comboBox.currentIndex()
    selectedLayer = layers[selectedLayerIndex].layer()
    fieldnames = [field.name() for field in selectedLayer.fields()]
    # Escreve cabeçalho
    line = ','.join(name for name in fieldnames) + '\n'
    output_file.write(line)
    # Escreve o atributo das features de saída.
    for f in selectedLayer.getFeatures():
        line = ','.join(str(f[name]) for name in fieldnames) + '\n'
        output_file.write(line)

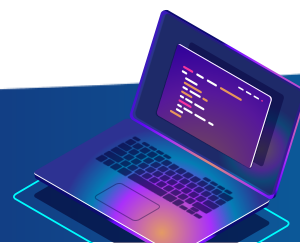
```

```

198 self.dlg.pushButton.clicked.connect(self.select_output_file)
199
200
201 # Fetch the currently loaded layers
202 layers = QgsProject.instance().layerTreeRoot().children()
203 # Clear the contents of the comboBox from previous runs
204 self.dlg.comboBox.clear()
205 # Populate the comboBox with names of all the loaded layers
206 self.dlg.comboBox.addItems([layer.name() for layer in layers])
207
208 # show the dialog
209 self.dlg.show()
210 # Run the dialog event loop
211 result = self.dlg.exec_()
212 # See if OK was pressed
213 if result:
214     filename = self.dlg.lineEdit.text()
215     with open(filename, 'w') as output_file:
216         selectedLayerIndex = self.dlg.comboBox.currentIndex()
217         selectedLayer = layers[selectedLayerIndex].layer()
218         fieldnames = [field.name() for field in selectedLayer.fields()]
219         # write header
220         line = ','.join(name for name in fieldnames) + '\n'
221         output_file.write(line)
222         # write feature attributes
223         for f in selectedLayer.getFeatures():
224             line = ','.join(str(f[name]) for name in fieldnames) + '\n'
225             output_file.write(line)
226

```

24) Por fim temos que notificar o usuário que o arquivo foi salvo com sucesso. A maneira de enviar notificações ao usuário no QGIS é através do método `self.iface.messageBar()`. `PushMessage()`. Adicione `Qgis` à lista de importações `qgis.core` na parte superior do arquivo e adicione o código abaixo no final do método `run`.



```
self.iface.messageBar().pushMessage(
    "Sucesso", "O arquivo de saída foi gravado " + filename,
    level=Qgis.Success, duration=3)
```

```

19 * the Free Software Foundation; either version 2 of the License, or
20 * (at your option) any later version.
21 *
22 *****
23 """
24 from PyQt5.QtCore import QSettings, QTranslator, qVersion, QCoreApplication
25 from PyQt5.QtGui import QIcon
26 from PyQt5.QtWidgets import QAction, QFileDialog
27 from qgis.core import QgsProject, Qgsis
28

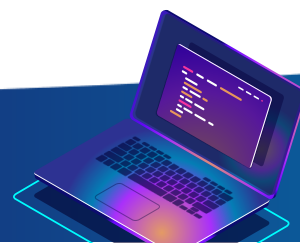
```

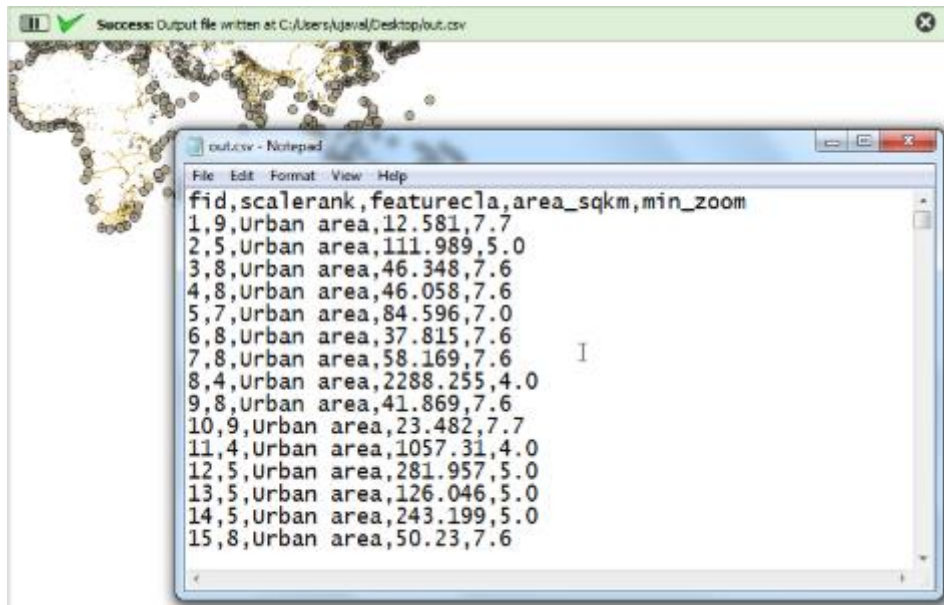
```

213 if result:
214     filename = self.dlg.lineEdit.text()
215     with open(filename, 'w') as output_file:
216         selectedLayerIndex = self.dlg.comboBox.currentIndex()
217         selectedLayer = layers[selectedLayerIndex].layer()
218         fieldnames = [field.name() for field in selectedLayer.fields()]
219         # write header
220         line = ','.join(name for name in fieldnames) + '\n'
221         output_file.write(line)
222         # write feature attributes
223         for f in selectedLayer.getFeatures():
224             line = ','.join(str(f[name]) for name in fieldnames) + '\n'
225             output_file.write(line)
226         self iface.messageBar().pushMessage(
227             "Success", "Output file written at " + filename,
228             level=Qgis.Success, duration=3)
229

```

25) Recarregue o plugin novamente e execute o seu plugin. Você verá que seu arquivo de saída será gerado com as informações tabulares.





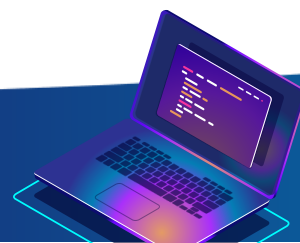
26) Você pode compactar a pasta do plugin e compartilhá-lo com outras pessoas. Eles podem descompactar o conteúdo na pasta de plugins do QGIS e experimentar o plugin criado por você.



Passo 3 – Publicação do plugin para comunidade open source (Opcional)

O plugin construído na plataforma pode ser divulgado com toda a comunidade open source. Em outras palavras você pode fazer downloads de novos plugins, feitos por outros usuários, como também publicar o plugin que você acabou de fazer, através do site: <https://plugins.qgis.org/plugins/>. Permitindo assim que outros façam a mesma operação que você acabou de desempenhar.

- 1) Entre no site <https://plugins.qgis.org/plugins/>, será necessário a realização de um cadastro, mas para realização deste cadastro, por questões de segurança, você deve solicitar um mantra para mantra-request@osgeo.org.
- 2) Após receber a resposta com o mantra, você deve entrar em <https://id.osgeo.org/ldap/create>.



Create OSGeo Userid

OSGeo user IDs provide access to many OSGeo services, in

Due to a recent increase in registration of fake users as spam

In order to verify your request, we will need to know a couple

- Why do you need to create a new OSGeo User?
- If it is to report a bug, for example, mention which project (OSGeo User)
- Or alternatively, please display one public affiliation that you have (Twitter, GitHub, BitBucket, or your posts archived in a public place)
- And BTW you don't need an account to download qgis

Email mantra-request@osgeo.org

Should you need any further assistance, you can find OSGeo

Join [#osgeo-sac](#) IRC chat

Enter mantra



3) Deve-se criar abaixo um cadastro, similar ao da imagem:



Create OSGeo Userid

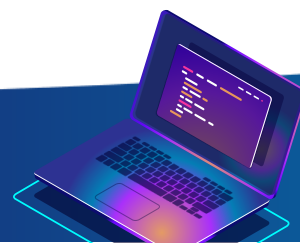
Submit this form to create an [OSGeo userid](#).


If you have any problem with the process contact sysadmin@osgeo.org or ask for help in the [#osgeo](#) IRC channel of the freenode network.


IMPORTANT: do not enter any special characters (such as "&") anywhere on the form.

Userid:	<input type="text"/>
Full name:	<input type="text"/> (eg. "Sol Katz")
Surname:	<input type="text"/> (eg. "Katz")
Email:	<input type="text"/>
Password:	<input type="password"/>
Password(confirm):	<input type="password"/>
<input type="submit" value="Submit"/>	

4) Após completar o cadastro e entrar na plataforma você será redirecionado para a tela de plugins.




[QGIS HOME](#)
[ABOUT PLUGINS](#)
[PLUGINS](#)
[PLANET](#)
[USER MAP](#)
[LOGOUT](#)

 Upload a plugin





Plugins

- [My plugins](#)
- [Featured](#)
- [All](#)
- [Stable](#)
- [Fresh](#)
- [Experimental](#)
- [Popular](#)
- [Most voted](#)
- [Top downloads](#)
- [Most rated](#)
- [QGIS Server plugins](#)


QGIS Python Plugins Repository


All plugins

1210 records found — [Click to toggle descriptions.](#)

Name	Stars	Author	Created on	Stars (votes)	Stable	Exp.
 A-Maps	— 5034	Riccardo Klinger	May 25, 2019	★★★★★ (12)	—	0.2
 AGIS	— 429	Matjaž Mori, ZVKDS CPA	March 19, 2020	★★★★★ (1)	1.0.0	—
 AGRC Geocoding Toolbox	— 10098	Scott Davis	Oct. 3, 2017	★★★★★ (11)	1.0.0	0.1
 AGT - Archaeological Geophysics Toolbox	— 15839	INRAP - Guillaume Hulin, Francois-Xavier Simon	Aug. 29, 2017	★★★★★ (28)	2.1	—

5) Por fim, uma página para upload de plugin abrirá, e você poderá informar que seu plugin é experimental, e deixar explícito que podem haver erros de operação. Para fazer o upload do seu plugin, é necessário compactar a pasta referente a ele.


[QGIS HOME](#)
[ABOUT PLUGINS](#)
[PLUGINS](#)
[PLANET](#)
[USER MAP](#)
[LOGOUT](#)

 Upload a plugin

Plugins

- [My plugins](#)
- [Featured](#)
- [All](#)
- [Stable](#)
- [Fresh](#)
- [Experimental](#)
- [Popular](#)
- [Most voted](#)
- [Top downloads](#)
- [Most rated](#)

QGIS Python Plugins Repository

Upload a plugin

To upload a new plugin or update an existing one, you can specify the zipped file in this form.

Alternatively, to update an existing plugin, you can also open the plugin's details view and add a new version from there.

Experimental ☐

Please check this box if the plugin is experimental. Please note that this field might be overridden by metadata (if present).

Plugin package:

Nenhum arquivo selecionado

Please select the zipped file of the plugin.



Parabéns! Chegamos ao final de mais capítulo, no qual você aprendeu sobre Criação de plugins usando scripts no PYQGIS.

