

# Criando Plugins QGIS com pyQGIS

Módulo 2 – QtDesigner, PluginBuilder e um plugin de verdade

# 1 - O QtDesiner e o PluginBuilder

Para facilitar a nossa vida existem duas ferramentas que auxiliam, e bastante, na criação de plugins. A primeira é o QtDesigner para construirmos as interfaces gráficas do usuário (GUI) e o PluginBuilder que cria praticamente todos os arquivos do plugin para nós.



O QtDesigner já vem com o QGIS e a tela inicial dele é:

Usaremos ele para criar o nosso plugin de verdade deste módulo.

O PluginBuilder precisa ser instalado e fazemos isso abrindo o QGIS e indo menu **Complementos->Gerenciar e instalar Complementos**. Selecione o botão Tudo e digite builder na caixa de busca. O Plugin Builder 3 deve aparecer. Instale o complemento.



Outra ferramenta interessante de se ter é o Plugin Reloader. Instale ele também.

🔇 Complementos   Tude	o (1287)	×
tudo 🏠	Q Reloa	≪
Instalados	<ul> <li>EnviroCat</li> <li>Plugin Reloader</li> <li>QRestart</li> </ul>	Plugin Reloader 🥠 🕯
<ul> <li>Não instalado</li> <li>Instalar a partir do ZIP</li> <li>Opções</li> </ul>	<ul> <li>Reloader</li> <li>Selection Sets Reloaded</li> </ul>	Reloads a chosen plugin in one click         This tool is only useful for Python Plugin Developers!
		Marcadores python, development, reload, reloader, developer Mais informações página inicial rastreador
		Atualizar Tudo     Instalar Complemento       Fechar     Ajuda

Vamos agora usar essas ferramentas na construção do nosso primeiro plugin funcional.

# 2 - Construindo o esqueleto do Grid\_Builder no Plugin Builder 3

Vamos iniciar pelo Plugin Builder:

🔇 Projeto sem título — QGIS		
Projeto <u>E</u> ditar <u>E</u> xibir <u>C</u> amada <u>C</u> onfigurações	<u>Complementos</u> Vet <u>or</u> <u>R</u> aster <u>B</u> anco de dados <u>W</u> eb <u>M</u> a	lha Pro <u>c</u> essamento <u>A</u> juda
🗅 📂 🗐 🔝 💸 🚺 🐡	🌲 Gerenciar e Instalar Complementos	III (I) 🔁   IRI - 🖻
	network Ctrl+Alt+P	
🕊 📽 Vi 🖍 端 🔛 🕖 🦷	<u>B</u> ásico ▶	
Navegador 🖉 🕅	<u>P</u> lugin Builder ▶	者 Plugin Builder
🗔 🔁 🍸 🗊 🕖	<u>P</u> lugin Reloader ▶	
☆ Favoritos ▶ III Favoritos Espaciais ▶ □ + C		

Inicie o plugin e preencha os campos dos formulários conforme as imagens a seguir:

GIS Plugin	Builder
Class name	GridBuilder
Plugin name	Grid Builder
Description	Plugin para criação de malha de pontos regulares
Module name	grid_builder
Version number	0.1
Minimum QGIS version	3.0
Author/Company	Você
Email address	seu@email

Esse primeiro formulário será usado na criação do arquivio 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.

🔇 QGIS Plugin Builder - 3.2.1				>
<b>QGIS Plugin Build</b>	ler			
	Template	Tool button with dialog	9	•
Text for the menu item	Grid Builder			
Menu	Vector		Ŧ	]
Ajuda	a <prev< td=""><td>vious Next &gt;</td><td>Cance</td><td>lar</td></prev<>	vious Next >	Cance	lar

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

	Dunue	1	 	 
Internationalization				
Help				
Unit tests				
Helper scripts				
Makefile				
pb_tool				

Desmarque todos para esse plugin,

Bug tracker	http://bugs
Repository	http://repo
	Publication (recommended Items)
Home page	http://homepage
Tags	python
✓ Flag th	ne plugin as experimental

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

QGIS Plugin Builder - 3.2.1	
QGIS Plugin Builder	
Select Output Directory	
Your plugin is ready to be generated. Select the output directory.	
C:/Users/User/AppData/Roaming/QGIS/QGIS3/profiles/default/python/plugins	
Your plugin will be created in the selected location, using the module name for the name of subdirectory.	fthe
C:/Users/User/AppData/Roaming/QGIS/QGIS3/profiles/default/python/plugins\grid_bu	uilder

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

Q Plugin Builder Results	×
Plugin Builder Results	
Congratulations! You just built a plugin for QGIS!	
Your plugin GridBuilder was created in: C:/Users/User/AppData/Roaming/QGIS/QGIS3/profiles/default/python/plugins\grid_builder	
Your QGIS plugin directory is located at: C:/Users/User/AppData/Roaming/QGI5/QGI53/profiles/default/python/plugins	
What's Next	
<ol> <li>If resources.py is not present in your plugin directory, compile the resources file using pyrcc5 (simply use pb_tool or make if you have automake)</li> <li>Optionally, test the generated sources using make test (or run tests from your IDE)</li> <li>Copy the entire directory containing your new plugin to the QGIS plugin directory (see Notes below)</li> <li>Test the plugin by enabling it in the QGIS plugin manager</li> <li>Customize it by editing the implementation file grid_builder.py</li> <li>Create your own custom icon, replacing the default icon.png</li> <li>Modify your user interface by opening grid_builder_dialog_base.ui in Qt Designer</li> </ol>	
Notes:	
<ul> <li>You can use pb_tool to compile, deploy, and manage your plugin. Tweak the pb_tool.cfg file included with your plugin as you add files. Install pb_tool using pip or easy_install. See http://loc8.cc/pb_tool for more information.</li> <li>You can also use the Makefile to compile and deploy when you make changes. This requires GNU make (gmake). The Makefile is ready to use, however you will have to edit it to add addional Python source files, dialogs, and translations.</li> </ul>	
For information on writing PyQGIS code, see http://loc8.cc/pyqgis_resources for a list of resources.	
©2011-2019 GeoApt LLC - geoapt.com	
ОК	5

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

#### C:/Users/User/AppData/Roaming/QGIS/QGIS3/profiles/default/python/plugins\grid\_builder

Users\User\AppData\Roamin	q\QGIS\QGIS3\profiles\default\pytho	n\plugins				
	, ,				📑 Deta	alh
lome	Data de modificação	Тіро	Tamanho			
basico	14/07/2024 09:08	Pasta de arquivos				
grid_builder	14/07/2024 11:59	Pasta de arquivos				
minimo	13/07/2024 15:49	Pasta de arquivos				
plugin_reloader	14/07/2024 11:35					
pluginbuilder3	14/07/2024 11:31	Pasta de arquivos				

## Os arquivos gerados:

📒 grid	l_builder ×	+				- 0 ×
$\leftarrow$	→ ↑ C 🗖	> ··· QGIS > QGIS3 > pi	rofiles > default > python	> plugins > grid_builder	Pesquisar em grid_builder	٩
🕀 No	vo ~ 🔏 🗘 🛅	▲ Classif	iicar 🗸 🗮 Visualizar 🗸 🚥			Detalhes
	Nome	^	Data de modificação	Тіро	Tamanho	
> (	🔟init		14/07/2024 11:59	Arquivo PY	2 KB	
~	🧧 grid_builder		14/07/2024 11:59	Arquivo PY	7 KB	
	🥁 grid_builder_dialog		14/07/2024 11:59	Arquivo PY	2 KB	
	📄 grid_builder_dialog_base.ui		14/07/2024 11:59	Arquivo UI	2 KB	
>	icon		14/07/2024 11:31	Arquivo PNG	2 KB	
~ <mark> </mark>	📄 metadata		14/07/2024 11:59	Documento de Texto	2 KB	
>	README		14/07/2024 11:59	Chrome HTML Document	2 KB	
>	README		14/07/2024 11:59	Documento de Texto	1 KB	
>	resources		14/07/2024 11:59	Arquivo PY	6 KB	
	resources.qrc		14/07/2024 11:59	Arquivo QRC	1 KB	
>						
>						
>						
>						
>						
S 10 itens						

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

#### Com base no README gerado abaixo vamos aos próximos passos.

#### Plugin Builder Results

Congratulations! You just built a plugin for QGIS!

Your plugin GridBuilder was created in: C:/Users/User/AppData/Roaming/QGIS/QGIS3/profiles/default/python/plugins\grid\_builder

Your QGIS plugin directory is located at: C:/Users/User/AppData/Roaming/QGIS/QGIS3/profiles/default/python/plugins

#### What's Next

- 1. If resources.py is not present in your plugin directory, compile the resources file using pyrcc5 (simply use **pb\_tool** or **make** if you have automake)
- 2. Optionally, test the generated sources using make test (or run tests from your IDE)
- 3. Copy the entire directory containing your new plugin to the QGIS plugin directory (see Notes below)
- 4. Test the plugin by enabling it in the QGIS plugin manager
- 5. Customize it by editing the implementation file grid\_builder.py
- 6. Create your own custom icon, replacing the default icon.png
- 7. Modify your user interface by opening grid\_builder\_dialog\_base.ui in Qt Designer

Notes:

- You can use pb\_tool to compile, deploy, and manage your plugin. Tweak the *pb\_tool.cfg* file included with your plugin as you add files. Install pb\_tool using *pip* or *easy\_install*. See http://loc8.cc/pb\_tool for more information.
- You can also use the Makefile to compile and deploy when you make changes. This requires GNU make (gmake). The Makefile
  is ready to use, however you will have to edit it to add addional Python source files, dialogs, and translations.

For information on writing PyQGIS code, see http://loc8.cc/pyqgis\_resources for a list of resources.

©2011-2019 GeoApt LLC - geoapt.com

O arquivo resouces.py foi gerado automaticamente, desta forma não precisaremos de executar o pyrcc5, somente se quisermos mudar o icone padrão do PluginBuilder.

Os arquivos já estão no diretório correto de plugins.

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.

Q Complementos   Inst	alados (13)	×
🆄 Tudo	Q Buscar	
	🗸 🏩 Basico	
lnstalados	🥪 Checar geometria	Plugin para criação de malha de pontos regulares
🏇 Não instalado 👞 Instalar a partir do	<ul> <li>✓ ■ DB Manager</li> <li>✓ Edição Offline</li> <li>✓ GRASS GIS provider</li> <li>✓ Grid Builder</li> </ul>	Plugin para criação de malha de pontos regulares usando uma camada p delimitador
ZIP	MetaSearch Catalog Client	Catagoria Master
🔆 Opções	✓ Minimo     ✓ Minimo     ✓ Minimo     ✓ OrfeoToolbox provider     ✓ Plugin Builder 3	Marcadores python Mais informações página inicial rastreador de problemas repositóri
	V Plugin Beloader	Autor Você
	V  Processing Verificador de topologia	Versão instalada 0.1
		Atualizar Tudo Desinstalar Complemento Reinstalar Complemento
		Fechar Ajuda

Inicie ele pelo Menu Vetor.

s	Vet <u>o</u> r <u>R</u> as	ter	<u>B</u> anco	de da	dos	; <u>\</u>	<u>V</u> eb	<u>M</u> alh	a	Pro	cessa	me	ento
3	Geopro	ocess	amento		۲				Ĵ	Ū	1	)	3
_	G <u>e</u> ome	etrias			×						9		-
	<u>A</u> nalisa	ir			×	B	Ê		•		¢.		ab
s	<u>G</u> rid Bu	uilde	r		Þ	Þ	Grid	Build	er			I	
	<u>I</u> nvesti	gar			Þ	-							
	Gerend	iar d	ados		Þ								

Ou pelo ícone na barra de ferramentas.

		_
	-	
_		
	1.00	
	-	

Funcionando, mas sem funcionalidade ainda.



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

# 3 - O plugin Grid\_Builder

O plugin Grid Builder, como o próprio nome já fala constrói malha de pontos regulares usando uma camada espacial do tipo polígono para delimitar onde a malha será criada. Essa ferramenta, embora simples, é muito útil para criar malhas de amostragem.

Vamos iniciar pelo QtDesigner para criarmos a nossa interface gráfica de usuário (GUI). Abra o arquivo **grid\_builder\_dialog\_base.ui** localizado em C:\Users\User\AppData\Roaming\QGIS\QGIS3\profiles\default\python\plugins\grid\_builder no primeiro diálogo do programa em **Open.** 

New Form - Qt Designer		$\times$
<ul> <li>✓ templates\forms</li> <li>Dialog with Buttons Bottom</li> <li>Dialog with Buttons Right</li> <li>Dialog without Buttons</li> <li>Main Window</li> <li>Widget</li> <li>&gt; Widgets</li> <li>&gt; Custom Widgets</li> </ul>	5. God	
Show this Dialog on Startup	Embedded Design Device: None Screen Size: Default size	
Create	Open Recent 🔻	Close
Open Form		×
$\leftarrow$ $\rightarrow$ $\checkmark$ $\uparrow$ $\checkmark$ QGIS $\rightarrow$ QGIS $\rightarrow$ profiles $\rightarrow$ default $\rightarrow$ pytho	on > plugins > grid_builder $\lor$ $\bigcirc$	Pesquisar em grid_builder 🛛 🔎
Organizar ▼ Nova pasta		≣ ▾ 🖬 🚷
MODELO Nome	Data de modificação	Тіро
ashpycache	14/07/2024 12:22	Pasta de arquivos
check_list	14/07/2024 11:59	Arquivo UI
> OneDrive		
✓ 📮 Este Computad		
> 💾 OS (C:)		
> 🥳 KODAK (D:)		
✓ KODAK (D:)		
> ANDRF		
Nome: grid_builder_dialog_base.ui		Designer UI files (*.ui) 🗸 🗸
		Abrir Cancelar

## A seguinte tela aparecerá:

Qt Designer		-	• ×
File Edit Form View Settings Wi	ndow Help		
D 🥔 🔡 🗖 🕒 📑 🎙	L R R R III = H II III III III III III III III		
Widget Box 🗗 🗙	01 Grid Builder - arid builder dialog baseui Object Inspector		đΧ
Filter	Filter		
<ul> <li>Layouts</li> </ul>	Object	Class	
Vertical Layout	🗸 🖼 GridBuilderDialog	Base ODialog	
III Horizontal Layout	button_box	QDialogButtonBox	
Grid Lavout			
88 Form Lavout			Ψ
× Spacers	Property Editor		đ×
2221 Horizontal Spacer	Filter	4	- /
Vertical Spacer	GridBuilderDialonBase : ODi	alon.	- 8-
Buttons	Property /	Value	
Push Button	v Oblivet	Value	
N Tool Button	ov Carel	GridBuilderDialogBase	
Radio Button	Y QWidget	, , , , , , , , , , , , , , , , , , ,	
Charle Barr	enabled		
	→ geometry	[(0, 0), 400 x 300]	
Command Link Button	→ sizePolicy	[Preferred, Preferred, 0, 0	1]
Dialog Button Box	> minimumSize	0 x 0	
<ul> <li>Item Views (Model-Based)</li> </ul>	> maximumSize	16777215 x 16777215	
List View	≥ sizeIncrement	0 x 0	
Tree View	Resource Browser		e ×
Table View		Filter	
Column View	<resource root=""></resource>		
Undo View			
<ul> <li>Item Widgets (Item-Based)</li> </ul>			
List Widget			
STree Widget			
Table Widget	Signal/Slot Editor Activ	on Editor Resource Brows	er

Vamos adicionar 5 widgets do tipo Label, 4 widgets do tipo Line Edit e um widget do tipo MapLayerComboBox. Basta clicar no Widget e arrastar até a janela do diálogo.

	St. 1. 147.1	<ul> <li>Input Widgets</li> </ul>	
× L ♥ Label	Display Widgets	Combo Box	QgsMapLayerComboBox
		🗾 Font Combo Box	
		E Line Edit	
		AI Text Edit	

## Teremos o diálogo mais ou menos com o seguinte formato:

Qt Grid Build	er - grid_builder_dialog_base.ui* 📃 💌
TextLabel	
Texti abel	
Tayti shal	· · · · · · · · · · · · · · · · · · ·
Tautiabal	· · · · · · · · · · · · · · · · · · ·
TextLabel	
I extLabel .	· · · · · · · · · · · · · · · · · · ·
	OK Cancel

Clique no Primeiro TextLabel e no property Editor altere o campo text de TextLabel para **Layer:** conforme abaixo:

🔟 Qt Designer			— C	x c
File Edit Form View Settings Wir	ndow Help			
Widget Box 🗗 🗙	🔃 Grid Builder - grid_builder_dialog_base.ui* 🗖 💌	Object Inspector		ē ×
Filter		Filter		
List Widget	V dyer.	Object Cla	55	
Tree Widget	Tevti abel	Single Control State Contro	ialog	
Table Widget	· · · · · · · · · · · · · · · · · · ·	button_box QD	ialogButtonBox	
<ul> <li>Containers</li> </ul>	TextLabel	label QLa	abel	
Group Box	. TextLabel	Property Editor		Ξ×
Scroll Area		Elitar		-
Tool Box	lextLadel	Filter		_ / ·
Tab Widget		label : QLabel		
Charlosd Wide at	OK Cancel	Property Value		_
Stacked Widget				
Frame		> text Layer:		_ <u>•</u>
Widget		textFormat AutoTe	ext Layer:	
MDI Area		Resource Browser		đΧ
Dock Widget		10	Filter	
QAxWidget		· · · · · · · · · · · · · · · · · · ·		
<ul> <li>Input Widgets</li> </ul>		<resource root=""></resource>		
Combo Box				
🖌 Font Combo Box				
nex) Line Edit		Signal/Slot Editor Action Editor	Resource Browser	
			Resource browser	

Repita o procedimento para os demais TextLabel renomeando eles para

#### Espaçamento de linha:

## Espaçamento de Amostra:

#### Rotação:

## Tampão:

Qt Designer			- 0	×
File Edit Form View Settings Window Help				
	- R - H I II - H I			
Widget Box 🗗 🗙	Ot Grid Builder - grid builder dialog base.ui* 📃 💌	Object Inspector		8×
Filter		Filter		
List Widget	· · · · · · · · · · · · · · · · · · ·	Object	Class	
Tree Widget	Espacamento de linha:	<ul> <li>GridBuilderDialogE</li> </ul>	Base QDialog	1
Table Widget		button_box	QDialogButtonBox	
<ul> <li>Containers</li> </ul>	Espaçamento de Amostra:	label	QLabel	
Group Box	Rotação:	Property Editor		8×
Scroll Area	Tampão:	Filter	·	-
Tool Box	· · · · · · · · · · · · · · · · · · ·	GridBuilderDialogBase : ODia		8 -
Tab Widget		Broperty	Value	
Stacked Widget		lavoutDirection	LeftToRight	
Frame		autoFillBackground		
Widget		styleSheet		'
MDI Area	I I I I I I I I I I I I I I I I I I I			-
Dock Widget		Resource Browser		e, x
		/ C	Filter	
QAXWidget		<resource root=""></resource>		
Combo Box				
🖌 Font Combo Box				
ABX Line Edit		Signal/Slot Editor Actio	n Editor Resource Browser	

O widget mMapLayerComboBox ficará inalterado, faremos a configuração pelo código python.

Agora altere a propriedade objectName dos campos LineEdit para:

lineEditSpY

lineEditSpX

lineEditRota

lineEditBuf

Qt Designer				-		×
File Edit Form View Settings Wi	indow Help					
	L 🔁 🖼 💷 🚍 HH 🏗 III 🗄 🗳					
Widget Box 🗗 🗙	Qt Grid Builder - grid_builder_dialog_base.ui* 📃 💌	Object Inspector				δ×
Filter	laver	Filter				
List Widget		Object	Class			
Tree Widget	Espacamento de linha:	label_4	QLabel			
Table Widget	· · · · · · · · · · · · · · · · · · ·	label_5	QLabel			
Containers	Espaçamento de Amostra:	lineEditSpY	QLineEdit			
Group Box	Rotação:	Property Editor				đΧ
Scroll Area	Tampão	Filter			L _	é
Tool Box		line Editory + Otion Edit			•	8 -
Tab Widget		Inecatopy : QLinecat	Malua			
Stacked Widget	OK Cancel	Property	value			- U
Frame		objectName	lineEditSpV			
Widget		<ul> <li>OWidget</li> </ul>	intecortopi			
MDI Area	· · · · · · · · · · · · · · · · · · ·		-			
MDI Area		Resource Browser				Β×
Dock Widget		/ C	Filt	er		
QAxWidget		<resource root=""></resource>				
<ul> <li>Input Widgets</li> </ul>						
Combo Box						
🗾 Font Combo Box						
REX Line Edit		Signal/Slot Editor Ac	rtion Editor Deer	urca Brow	Car	
		Signay orot Editor Ac	Resu	urce brow	301	

Pronto, salve o diálogo e feche o QtDesigner.

Vamos agora editar o arquivo **grid\_builder.py** para realizar a tarefa. Vamos ter de adicionar algumas bibliotecas de suporte via **import** e adicionar o código na função **run** que vai fazer a validação inicial dos campos e a criação da camada de pontos da malha.

#### As bibliotecas serão (adicionar as faltantes):

```
from qgis.PyQt.QtCore import QSettings, QTranslator, QCoreApplication
from qgis.PyQt.QtGui import QIcon, QIntValidator
from qgis.PyQt.QtWidgets import QAction,QMessageBox
# Initialize Qt resources from file resources.py
from .resources import *
# Import the code for the dialog
from .grid_builder_dialog import GridBuilderDialog
import os.path
from qgis.core import QgsProject, QgsPointXY, QgsGeometry, QgsVectorLayer,QgsFeature
from qgis.core import QgsPoint, QgsField, QgsMapLayerType, QgsWkbTypes
```

#### A função run ficará assim:

```
def run(self):
    if self.first_start == True:
        self.first_start = False
        self.dlg = GridBuilderDialog()
        self.dlg.mMapLayerComboBox.setShowCrs(True)
```

```
self.map layers = QgsProject.instance().mapLayers().values()
            self.allow list = [
                                 lyr.id() for lyr in self.map layers if lyr.type() ==
QgsMapLayerType.VectorLayer
                                 and lyr.geometryType() == QgsWkbTypes.PolygonGeometry
                             1
            self.except list = [l for l in self.map layers if l.id() not in self.allow list]
            self.dlg.mMapLayerComboBox.setExceptedLayerList(self.except list)
            onlyInt = QIntValidator() # mudar para QDoubleValidator se for usar lat long também
            self.dlg.lineEditSpY.setText('400')
            self.dlg.lineEditSpY.setValidator(onlyInt)
            self.dlg.lineEditSpX.setText('200')
            self.dlg.lineEditSpX.setValidator(onlyInt)
            self.dlg.lineEditRota.setText('0')
            self.dlg.lineEditRota.setValidator(onlyInt)
            self.dlg.lineEditBuf.setText('1000')
            self.dlg.lineEditBuf.setValidator(onlyInt)
        self.dlg.show()
        result = self.dlg.exec ()
        if result:
            if self.dlg.lineEditSpY.text() == '' or self.dlg.lineEditSpX.text() == '' or
self.dlg.lineEditRota.text() == '' or self.dlg.lineEditBuf.text() == '':
                 QMessageBox.warning(self.iface.mainWindow(),
                          'Erro',
                          "Entre todos os campos por favor\nSaindo...")
                return
            layer = self.dlg.mMapLayerComboBox.currentLayer()
            feats = [ feat for feat in layer.getFeatures()]
            points = []
            spacing y = int(self.dlg.lineEditSpY.text())
            spacing x = int(self.dlg.lineEditSpX.text())
            rotacao= int(self.dlg.lineEditRota.text())
            extensao= int(self.dlg.lineEditBuf.text())
            #executar o código
            # - -
            for feat in feats:
                 centroid = feat.geometry().centroid().asPoint()
                 extent = feat.geometry().boundingBox()
                xmin=int(round(extent.xMinimum()-extensao, -2))
ymin=int(round(extent.yMinimum()-extensao, -2))
                 xmax=int(round(extent.xMaximum()+extensao, -2))
                ymax=int(round(extent.yMaximum()+extensao, -2))
                rows = int(((ymax) - (ymin))/spacing y)
                cols = int(((xmax) - (xmin))/spacing_x)
                x = xmin
                y = ymax
                geom_feat = feat.geometry()
                 for i in range(rows+1):
                     for j in range(cols+1):
                         pt = QgsPointXY(x,y)
                         tmp pt = QgsGeometry.fromPointXY(pt)
                         tmp_pt.rotate(rotacao, centroid)
if tmp_pt.within(geom_feat):
                            points.append(tmp pt.asPoint())
                         x += spacing x
                    x = xmin
                     y -= spacing_y
            epsg = layer.crs().postgisSrid()
            #gerando pontos
            uri = "PointZ?crs=epsg:" + str(epsg) + "&field=id:integer""&index=yes"
            mem_layer = QgsVectorLayer(uri, 'gridpoints', 'memory')
            prov = mem layer.dataProvider()
            feats = [ QgsFeature() for i in range(len(points)) ]
            for i, feat in enumerate(feats):
                 feat.setAttributes([i])
                 feat.setGeometry(QgsPoint(points[i].x(),points[i].y(),0.0))
            prov.addFeatures(feats)
            QgsProject.instance().addMapLayer(mem_layer)
            QMessageBox.information(self.iface.mainWindow(),
                          'Pronto',
                          "Pontos de amostragem criados!")
            Return
```

Antes de comentarmos o código adicionado vamos testar o plugin. Abra o QGIS e o plugin será carregado já com as alterações feitas. Ao iniciarmos o plugin teremos:

🔇 Grid Builder	×
Layer:	•
Espaçamento de linha:	400
Espaçamento de Amostra:	200
Rotação:	0
Tampão:	1000
	OK Cancelar

Primeiro crie ou abra uma camada do tipo polígono para executar o plugin Grid Builder.

Clique em cancelar e crie um polígono (em coordenadas UTM, lat long funciona, mas os espaçamentos e o tampão devem ser entrados como decimal de grau e os campos de entrada validam somente inteiros, para aceitar decimais é preciso mudar o "validator" para QDoubleValidator no código).

Caso prefira carregue a camada polígono do site em : https://gdatasystems.com/pyqgis/index.php

Q *Projeto sem título — QGIS	- 0 ×
Projeto <u>E</u> ditar <u>Exibir</u> <u>C</u> amada <u>C</u> onfigurações (	Çomplementas Vetgr Baster Basto de dados Web Malha Progessamento Ajuda
🗅 🚞 🗒 🌄 😪 🕄 🖤	μ 戸 辺 (1)
🕷 🎕 Vi 🔏 🖏 🎛 🖉 🛛	/ 長/-名友-該自べの回らけ! 🔍 🧠 🔍 忠忠忠 (像) 🐟 🧮 🖉 🎟 🖉 🤉 👘 🥂 🤈 🕫 🔲
Navegador 🖉 🛞	
🖫 😋 🝸 🖆 O	
🚖 Favoritos	
Image: Imag	
CA (OS)	
Di (KODAK)     Geoßadases	
SpatiaLite	
PostgreSQL	
MS SQL Server	
Oracle	
WMS/WMIS Scenes	
Wector Tiles	
<ul> <li>XVZ Tiles</li> <li>FSRI</li> </ul>	
Mapzen Global Terrain	
OpenStreetMap	
WFS / OGC API - Features	
ArcGIS REST Servers	
Camadas 🔞 🕅	
🗸 🕼 🔍 🍸 🏭 🛪 🗊 🖬	
V poligono	
D Economica pages Installings (Ctd.)	Caerdenada 612192 8235316 🐝 Escala 1:124296 🛪 🖨 Luna 100% 👶 Retarão 0.0 ° 🚖 🗸 Renderizar 🖤 EPSC-31982 🚥

Camada polígono carregada.

Execute o plugin com os seguintes parâmetros:

Q Grid Builder	×
Layer:	🗭 poligono [EPSG:31982] 💌
Espaçamento de linha:	1000
Espaçamento de Amostra:	500
Rotação:	-15
Tampão:	1000
	OK Cancelar

A seguinte mensagem aparece e a camada temporária gridpoint é criada.

	Q Pronto	×	
	i	Pontos de amostragem criados! OK	
Q *Projeto sem titulo — QGIS Besiato Editar Eviliar Camada Configu	rendar Complementor Vietor Parter Banco de dador Web Malha Deverramento	Alicela	- 0 ×
0 🖿 🗟 🖸 📽 🛍 🔘		🔣 - 📴 - 🖏 - 🔍 - 🔍 🛗 🌞 Σ 🛅 - 🚍 - 🤛 🔍 -	
🧟 🎕 Vi 🔏 🖷 🔛		भू 👒 🕮 भा भा भा भा भा भा 🛞 🏩 😹 📷 🎢 🗩 🔝	
Navegador	(1) (B)		
Anounce Expension     Anounce Expension     Anounce Expension     Anounce Expension     Constance     Constance     Constance     Westy Wats     Senses     Westy Wats     Anounce     Westy Wats     Constance     Westy Wats     Westy Take     House Close Anounce     Westy Occ. AP - Features     Westy			
Q. Escreva para localizar (Ctrl+K)	1 entrada da legenda removida.	Coordenada 625235 6216311 🛞 Escala 1:124796 💌 🔒 Lapa 100% 🌣	Rotação 0,0 * 🗘 🗸 Renderizar 💮 EPSG:31982 📿

Agora é só salvar a camada criada.

Este bloco do código inicializa os widgets do plugin carregando os polígonos abertos no combo box e adicionando os valores iniciais nas linhas editáveis.

```
def run(self):
        if self.first_start == True:
            self.first start = False
            self.dlg = GridBuilderDialog()
            self.dlg.mMapLayerComboBox.setShowCrs(True)
            self.map layers = QgsProject.instance().mapLayers().values()
            self.allow_list = [
                                lyr.id() for lyr in self.map layers if lyr.type() ==
QgsMapLayerType.VectorLayer
                                and lyr.geometryType() == QgsWkbTypes.PolygonGeometry
                            1
            self.except list = [l for l in self.map layers if l.id() not in self.allow list]
            self.dlg.mMapLayerComboBox.setExceptedLayerList(self.except list)
            onlyInt = QIntValidator() mudar para QDoubleValidator se for usar lat long também
            self.dlg.lineEditSpY.setText('400')
            self.dlg.lineEditSpY.setValidator(onlyInt)
            self.dlg.lineEditSpX.setText('200')
            self.dlg.lineEditSpX.setValidator(onlyInt)
            self.dlg.lineEditRota.setText('0')
            self.dlg.lineEditRota.setValidator(onlyInt)
            self.dlg.lineEditBuf.setText('1000')
            self.dlg.lineEditBuf.setValidator(onlyInt)
```

O bloco seguinte checa se os campos estão todos preenchidos e assinala eles às variáveis do programa.

Aqui lemos o polígono e assinalamos sua extensão, calculamos o buffer e checamos qual sistema de coordenada devemos usar para gerar os pontos.

```
#executar o código
#------
#
for feat in feats:
    centroid = feat.geometry().centroid().asPoint()
    extent = feat.geometry().boundingBox()
    xmin=int(round(extent.xMinimum()-extensao, -2))
    ymin=int(round(extent.yMinimum()-extensao, -2))
    xmax=int(round(extent.yMaximum()+extensao, -2))
    ymas=int(round(extent.yMaximum()+extensao, -2))
    rows = int(((ymax) - (ymin))/spacing_y)
    cols = int(((xmax) - (xmin))/spacing_x)
    x = xmin
    y = ymax
```

Finalmente criamos os pontos e carregamos eles em um arquivo na memória.

```
geom feat = feat.geometry()
    for i in range (rows+1):
        for j in range(cols+1):
            pt = QgsPointXY(x, y)
            tmp pt = QgsGeometry.fromPointXY(pt)
            tmp_pt.rotate(rotacao, centroid)
if tmp_pt.within(geom_feat):
                points.append(tmp_pt.asPoint())
            x += spacing x
        x = xmin
        y -= spacing_y
epsg = layer.crs().postgisSrid()
uri = "PointZ?crs=epsg:" + str(epsg) + "&field=id:integer""&index=yes"
mem layer = QgsVectorLayer(uri, 'gridpoints', 'memory')
prov = mem layer.dataProvider()
feats = [QgsFeature() \text{ for i in range(len(points))}]
for i, feat in enumerate(feats):
    feat.setAttributes([i])
    feat.setGeometry(QgsPoint(points[i].x(),points[i].y(),0.0))
prov.addFeatures(feats)
QgsProject.instance().addMapLayer(mem layer)
QMessageBox.information(self.iface.mainWindow(),
             'Pronto',
             "Pontos de amostragem criados!")
Return
```

No próximo módulo avançaremos um pouco mais com um novo exemplo de plugin que usa uma biblioteca externa, não padrão do QGIS. Até lá!