Universitat Politècnica de València

[School of Engineering in Geodesy, Cartography and Surveying](http://www.upv.es/entidades/ETSIGCT/indexi.html)

**FINAL PROJECT**

Desarrollo de aplicaciones SIG

**Vicente Ortí Llull**

Eliška Vlčková

C:\Users\Efox\Downloads\logo_upv2_va.gif

València 2017

INDEX

[1 GOAL 3](#_Toc486193046)

[2 DESCRIPTION 4](#_Toc486193047)

[2.1 Interface 4](#_Toc486193048)

[2.2 Map template 5](#_Toc486193049)

[2.3 Functionality 6](#_Toc486193050)

[2.3.1 Interface functionality 6](#_Toc486193051)

[2.3.2 Enabling and disabling widgets 9](#_Toc486193052)

[2.3.3 Automatic loading data 10](#_Toc486193053)

[2.4 Creating thematic maps 11](#_Toc486193054)

[2.4.1 Access to the list of layers 11](#_Toc486193055)

[2.4.2 Apply interpolation 11](#_Toc486193056)

[2.4.3 Enable wather layer 12](#_Toc486193057)

[2.4.4 Draw graphic elements 12](#_Toc486193058)

[2.4.5 Zooming to selected territory 13](#_Toc486193059)

[2.4.6 Exporting maps 14](#_Toc486193060)

[2.4.7 Template restart 14](#_Toc486193061)

[3 CODE 15](#_Toc486193062)

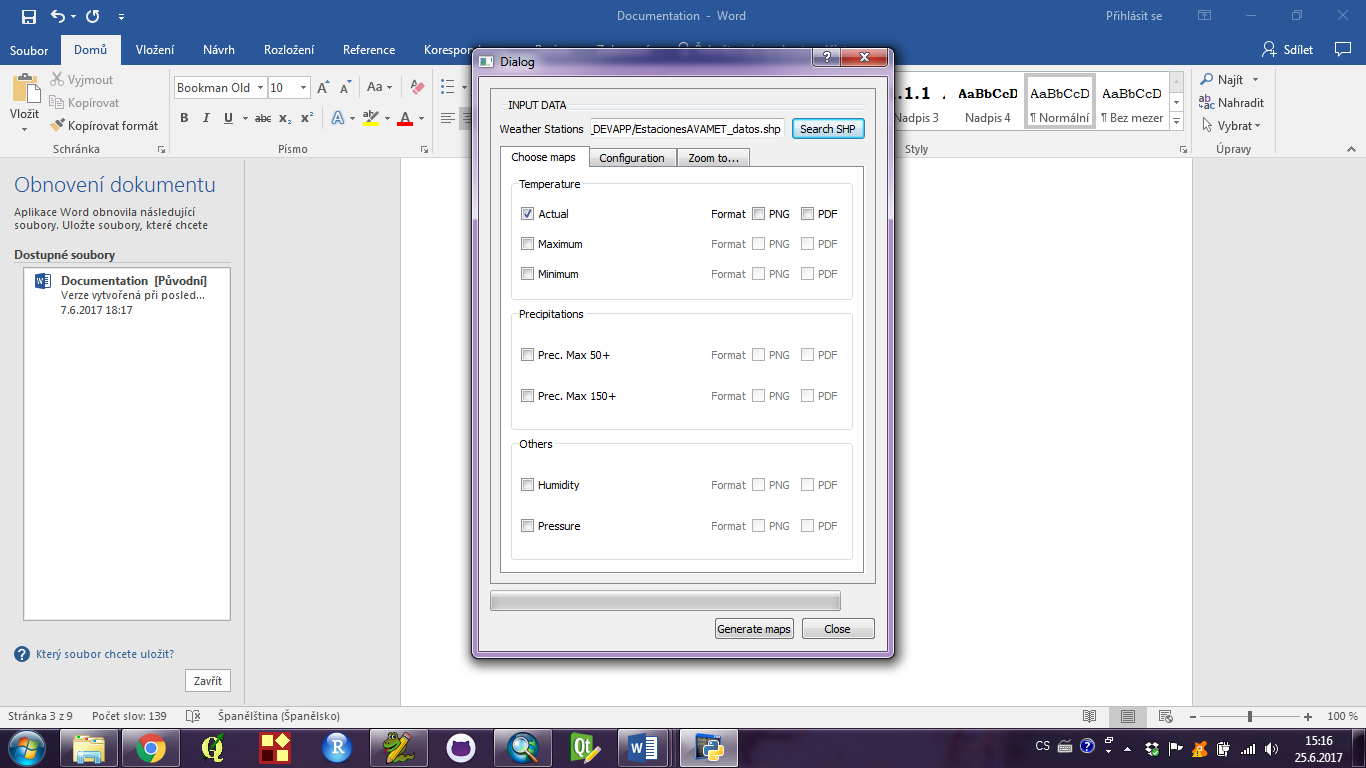
[4 RESULTS 22](#_Toc486193063)

[5 CONCLUSION 24](#_Toc486193064)

[6 BIBLIOGRAPHY 25](#_Toc486193065)

# GOAL

The main goal was to create stand-alone tool, what performs creating maps using interpolation algorithm (*Figure 1*). Based on input data, there is several types of maps, what can be generated: actual temperature, maximum temperature, minimum temperature, precipitations, humidity and presure – both have output option for png format and pdf format. Configuration options let user to display legend, scale, time and logo. Additionally, the user is allowed to define supplementary elements of the topographic background. For example, the administrative boundaries (region, province, etc.) and other backgrounds (hillshade, rivers, cities, or meteorological data input data) can be selected. The last option is the „zoom option“ on the given territory, which is mentioned in the last tab of the interface.



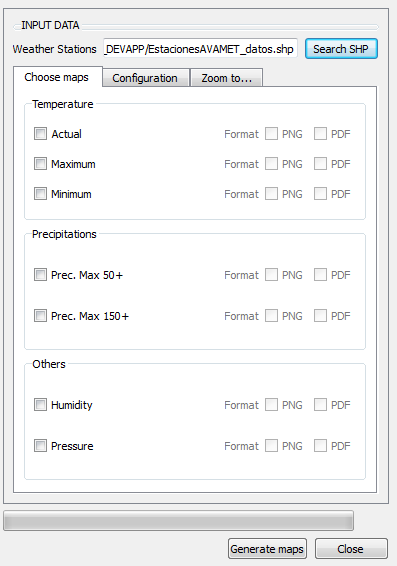
*Figure 1: interface of application*

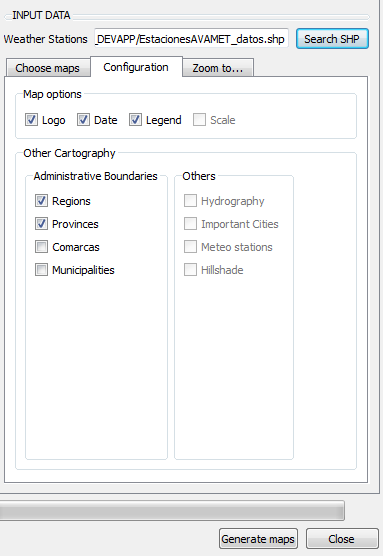
# DESCRIPTION

A graphical interface with PyQt4 had been created to build this tool. Once the interface had been built, an .ui file was generated and edited with Python consoles in order to assign functions to the buttons.

## Interface

Since the objects and their functions and classes have been sufficiently described in the previous personal assignment, space in this project is devoted to another things. The interface itself was divided into three parts: choosing propriet data for interpolation (*Figure 2*), configuration parameters for map creating (*Figure* 3) and possibility to zoom to given territory (*Figure* 4).

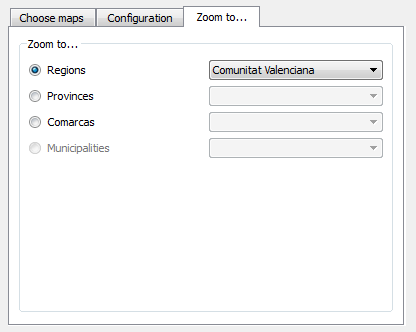


**

*Figure 2: choose maps interface Figure 3: configuration map interface*

The application requires as input shapefile with weather data. Those data must unfortunatelly contain already joined weather conditions. And as was said in previous chapture, the application allows to choose different weather maps and their formats (Figure 2). In configuration tab can user define, whether want to draw into the map administrative boundaries, another topographic background, or to enable or disable basic map components, such as logo, date and legend.

It’s possible to zoom the map to a determinated zone. As default, map will be generated in regional scale. However, it’s possible to zoom the map to a specific province or region (*Figure 4*)



*Figure 4: zooming options in interface*

Once are all options defined, the programme will generate the weather maps according to user’s specifications (Figure 5).

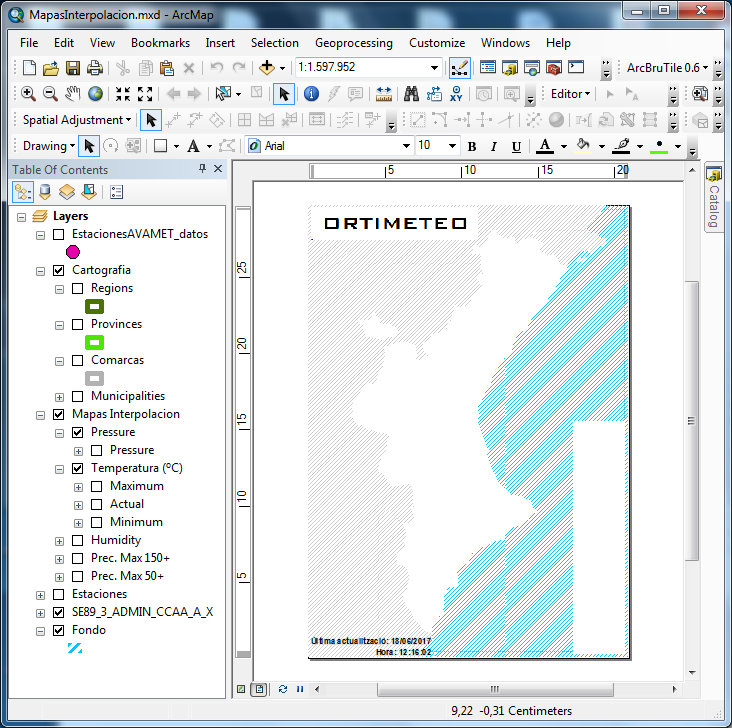


*Figure 5*: maps generating button

## Map template

After interface was created, a print template map has been built, attending to the user specifications defined on the interface (*Figure 6*).

An easy-identification nomenclature of the map objects (logo, legend, date, layers) has been used, as in the software interface, to facilitate the implementation of functions with python.



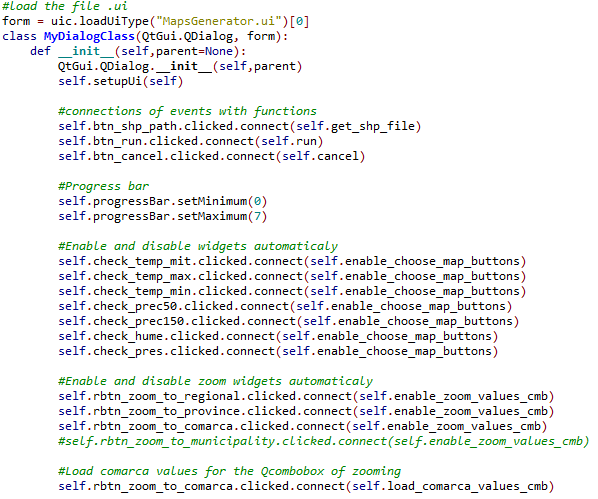
*Figure 6: map template shown in ArcMap*

## Functionality

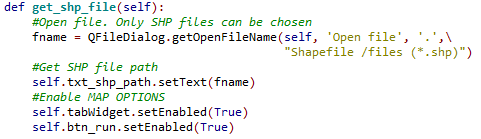
This chapter contains a light introduction to the functions and buttons what were used and their connection to the user environment.

### Interface functionality

Firstly, a connection between buttons in interface and functions written in Python had to be introduced (see the following code).



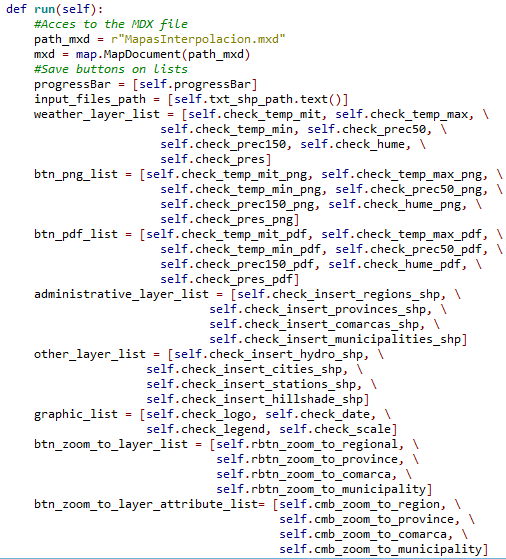
As in the interface are 3 buttons, the functions for them had to be introoduced as well. First one contains of function for loading shp data, second one for running the program itself and maps generating and the last one for closing the window. On the code below, is firstly shown the function for getting shapefile from user. This function uses the QFileDialog class, what provides a dialog that allow users to select files or directories. As can be seen, after putting the shapefile inside the programme, another functions in the interface are getting enabled.



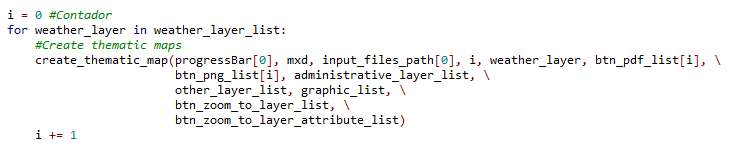
After was created the function for closing the window.

Výřez obrazovky

The last function, „generate maps“ is connected to run function written in Python. After clicking on this button, the function will generate the map according to the data specifications. For understanding the code bellow, here is a little inctroduction. Firstly, it was neccessary to access a MXD file (this was created in pararel with map template what is descibed in previous chapture). And after, for better assebility and simplification, the widgets of the interface had been saved in different lists (see the code bellow).

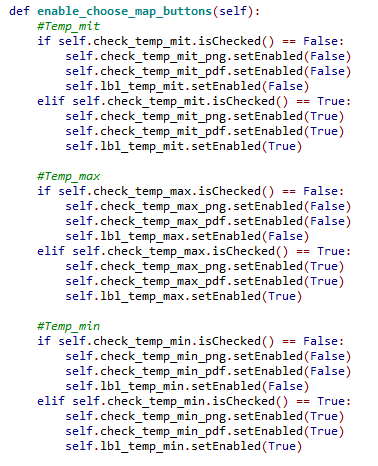


The last section in running function contained the creating thematic map function (described later). This function will be applied to all possible maps and saved in list („weather layer list“).



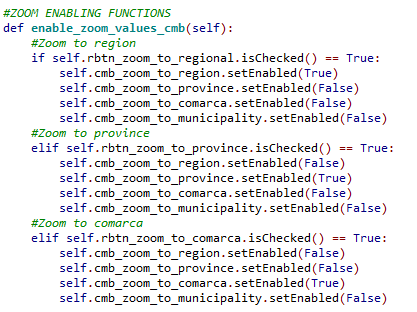
### Enabling and disabling widgets

In order to facilitate the order of the parameters for checking the program, two enabling and disabling widgets functions were created. First one for the format output map widgets, and other for the zooming ones.

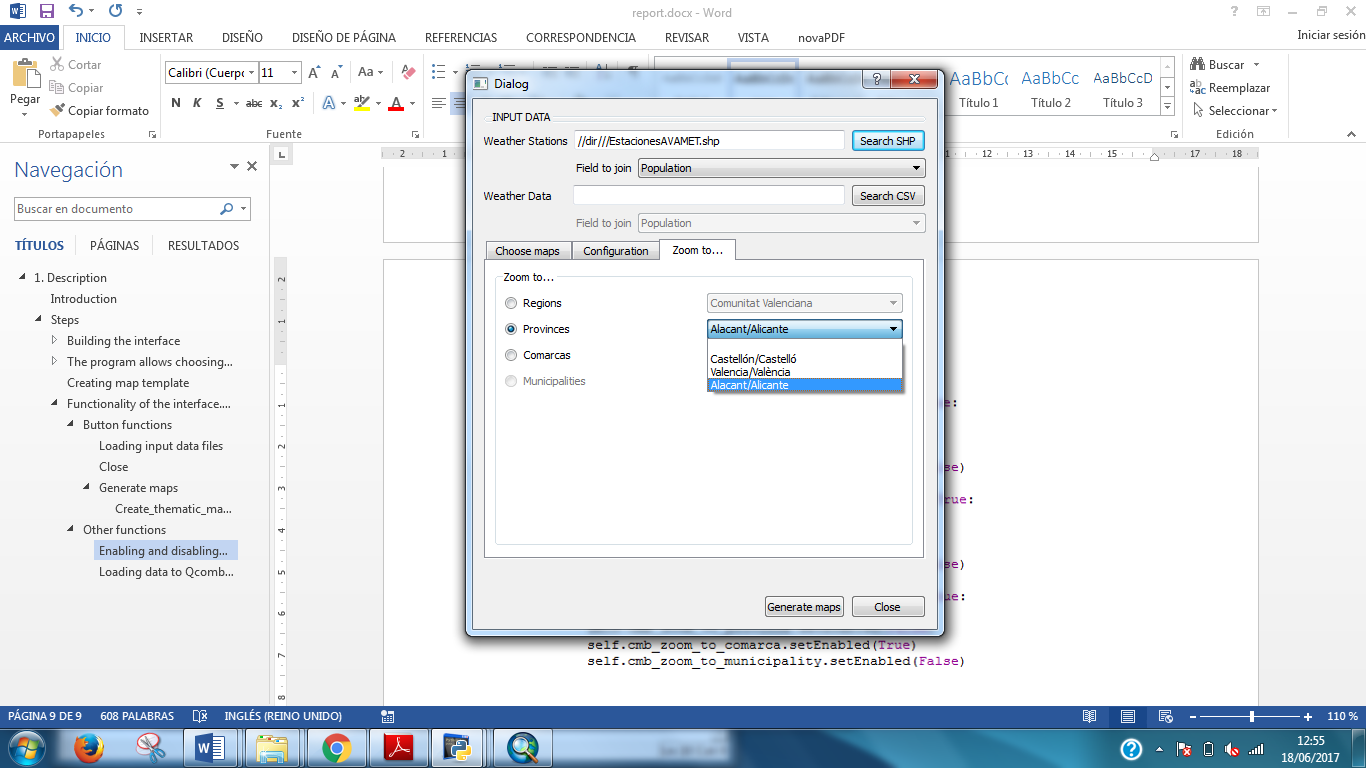


The code above shows only an example of enabling and disabling buttons, accordingly, the rest of meteorological variables was created in the same way.

For zooming enable options were used similar process (the code below). As in the code above, the whole code for all options is not shown.



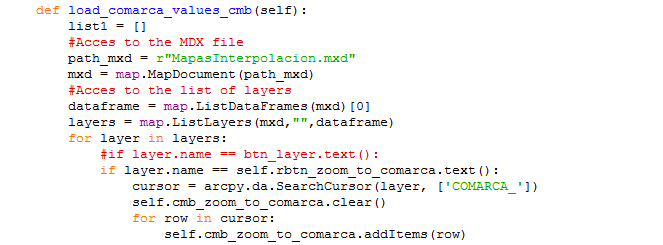
An example of disabling zooming options is shown on the following picture *(Figure 6*)

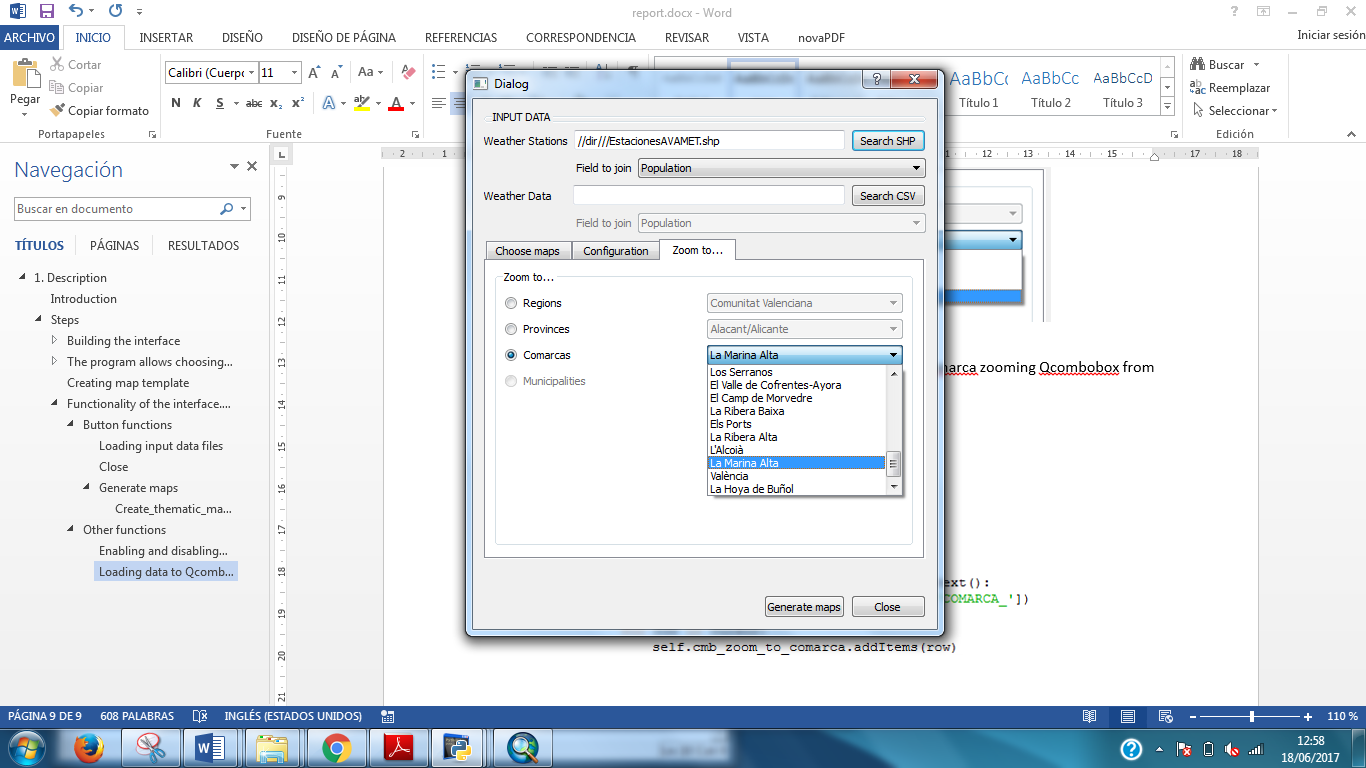


*Figure 6: zooming options (enable for provinces)*

### Automatic loading data

Writing about zooming options, there were one thing what had to be done more – automatic loading data into QComboBox (*Figure 7*). It consists of automatically loaded field names from the layer what had been created before.

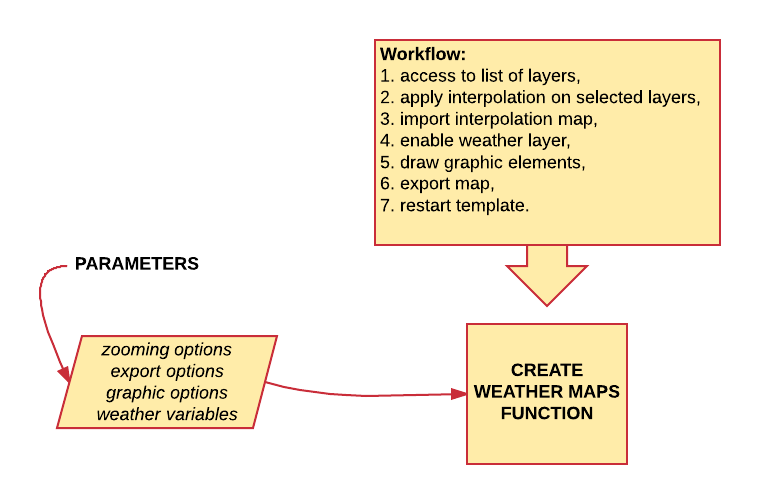




*Figure 7: automatically loading data*

## Creating thematic maps

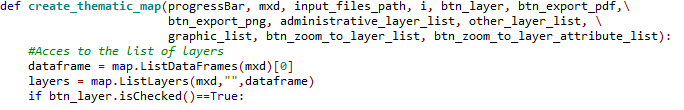
In this section are described several functions for generating weather maps. The workflow for generating was following (*Figure* 8):



*Figure 8: workflow for creating thematic map*

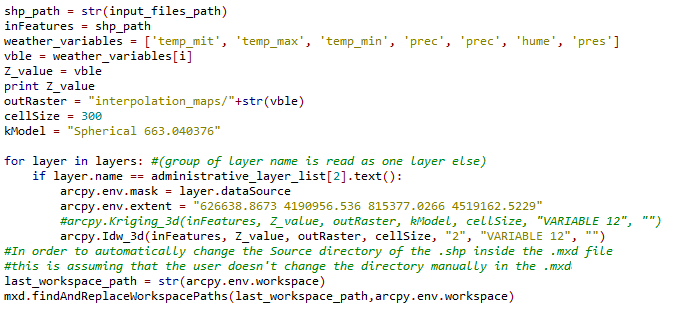
### Access to the list of layers

The following code shows definition of function with its parameters and accessing to the list of layers.



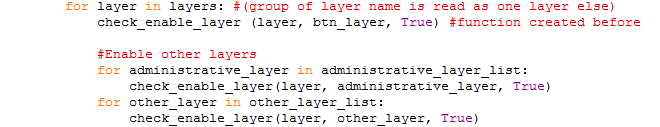
### Apply interpolation

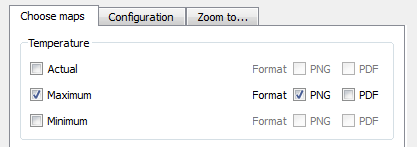
Even though in the code (see below) are implemented two interpolation algorithms (IDW and Kriging), it was decided to use IDW, because in Kriging is appropriate manual data exploration analysis. After checking extension, the code was following:



### Enable wather layer

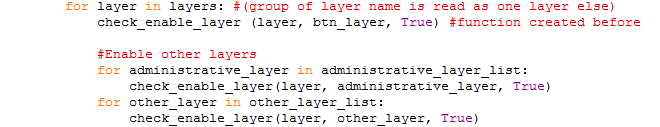
The programme enables the weather layer chosen by user (*Figure 9*).



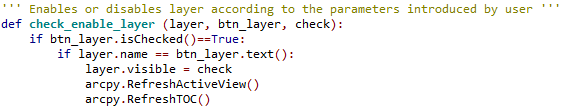


*Figure 9: enabling layers*

Moreover, other layers for users are enabled in configuration window.

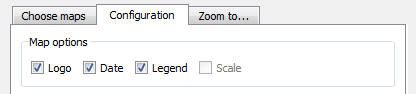


In order to do this, programme uses *check\_enable\_layer* function. By default, no layer is enabled. This function enables the layer introduced by user on the interface.



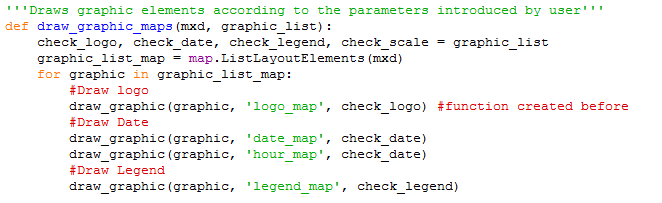
### Draw graphic elements

Next step was to draw graphic elements (logo, date, legend) chosen by user in Configuration layer (*Figure 10*).

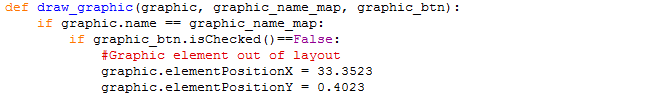


*Figure 10: draw graphic elemens*

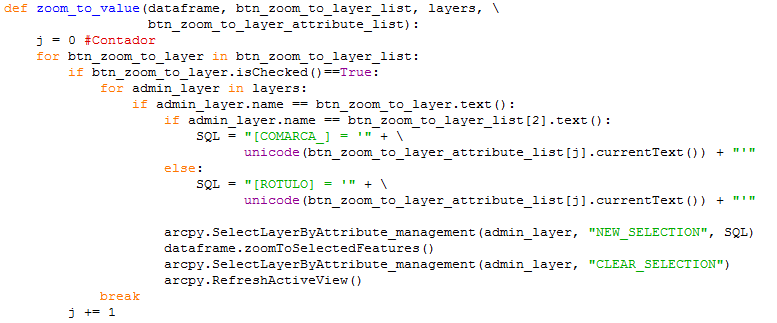
The code for drawing graphics element is following:



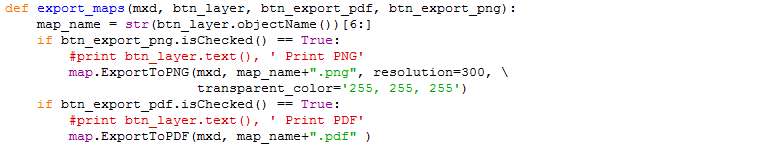
This function uses another function called *draw\_graphic* that draws the graphic element out of layout if user haven’t chosen it.



### Zooming to selected territory

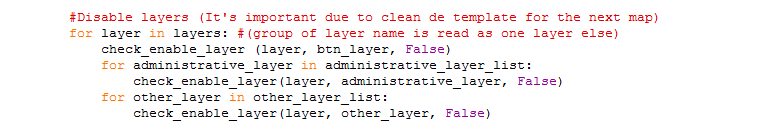
When layer maps are generated and added to the template and graphic objects are enabled or disabled, program zooms the layout to a specific area, according to the user’s specifications.

### Exporting maps

Once are all output maps configured according to the parameters introduced in the interface, maps are exported in the formats what were selected.

### Template restart

Because programs allows generating more than one map, it’s important to restart the layer’s template configuration. So after exporting map all layer will began disable again.



# CODE

1. # -\*- coding: utf-8 -\*-

4. #################
5. ### LIBRARIES ###
6. #################
7. **import** sys
8. **from** PyQt4 **import** QtCore, QtGui, uic
9. **from** PyQt4.QtCore **import** \*
10. **from** PyQt4.QtGui **import** \*
11. **import** arcpy
12. **import** arcpy.mapping as map
13. **import** sys, csv, os
14. **from** arcpy **import** env
16. #################
17. ### FUNCTIONS ###
18. #################
19. ''''' Enables or disables layer according to the parameters introduced by user '''
20. **def** check\_enable\_layer (layer, btn\_layer, check):
21. **if** btn\_layer.isChecked()==True:
22. **if** layer.name == btn\_layer.text():
23. layer.visible = check
24. arcpy.RefreshActiveView()
25. arcpy.RefreshTOC()
27. '''''Draws graphic element'''
28. #Without position
29. **def** draw\_graphic(graphic, graphic\_name\_map, graphic\_btn):
30. **if** graphic.name == graphic\_name\_map:
31. **if** graphic\_btn.isChecked()==False:
32. #Graphic element out of layout
33. graphic.elementPositionX = 33.3523
34. graphic.elementPositionY = 0.4023
35. #print graphic\_name\_map, ": Disabled"

38. '''''Draws graphic elements according to the parameters introduced by user'''
39. **def** draw\_graphic\_maps(mxd, graphic\_list):
40. check\_logo, check\_date, check\_legend, check\_scale = graphic\_list
41. graphic\_list\_map = map.ListLayoutElements(mxd)
42. **for** graphic **in** graphic\_list\_map:
43. #Draw logo
44. draw\_graphic(graphic, 'logo\_map', check\_logo) #function created before
45. #Draw Date
46. draw\_graphic(graphic, 'date\_map', check\_date)
47. draw\_graphic(graphic, 'hour\_map', check\_date)
48. #Draw Legend
49. draw\_graphic(graphic, 'legend\_map', check\_legend)

52. ##            #Draw Scale
53. ##            draw\_graphic(graphic, 'scale\_map', check\_scale)
55. ''''' Zooms the map according to the parameters introduced by user'''
56. **def** zoom\_to\_value(dataframe, btn\_zoom\_to\_layer\_list, layers, \
57. btn\_zoom\_to\_layer\_attribute\_list):
58. j = 0 #Contador
59. **for** btn\_zoom\_to\_layer **in** btn\_zoom\_to\_layer\_list:
60. **if** btn\_zoom\_to\_layer.isChecked()==True:
61. **for** admin\_layer **in** layers:
62. **if** admin\_layer.name == btn\_zoom\_to\_layer.text():
63. **if** admin\_layer.name == btn\_zoom\_to\_layer\_list[2].text():
64. SQL = "[COMARCA\_] = '" + \
65. unicode(btn\_zoom\_to\_layer\_attribute\_list[j].currentText()) + "'""'"
66. else:
67. SQL = "[ROTULO] = '" + \
68. unicode(btn\_zoom\_to\_layer\_attribute\_list[j].currentText()) + "'"
70. arcpy.SelectLayerByAttribute\_management(admin\_layer, "NEW\_SELECTION", SQL)
71. dataframe.zoomToSelectedFeatures()
72. arcpy.SelectLayerByAttribute\_management(admin\_layer, "CLEAR\_SELECTION")
73. arcpy.RefreshActiveView()
74. **break**
75. j += 1
77. '''''Export maps according to the parameters introduced by user '''
78. **def** export\_maps(mxd, btn\_layer, btn\_export\_pdf, btn\_export\_png):
79. map\_name = str(btn\_layer.objectName())[6:]
80. **if** btn\_export\_png.isChecked() == True:
81. #print btn\_layer.text(), ' Print PNG'
82. map.ExportToPNG(mxd, map\_name+".png", resolution=300, \
83. transparent\_color='255, 255, 255')
84. **if** btn\_export\_pdf.isChecked() == True:
85. #print btn\_layer.text(), ' Print PDF'
86. map.ExportToPDF(mxd, map\_name+".pdf" )
88. '''''Creates thematic maps according to the parameters introduced by user '''
89. '''''This function uses other functions: draw\_graphic(),
90. check\_enable\_layer(), export\_maps()'''
91. **def** create\_thematic\_map(progressBar, mxd, input\_files\_path, i, btn\_layer, btn\_export\_pdf,\
92. btn\_export\_png, administrative\_layer\_list, other\_layer\_list, \
93. graphic\_list, btn\_zoom\_to\_layer\_list, btn\_zoom\_to\_layer\_attribute\_list):
94. #Acces to the list of layers
95. dataframe = map.ListDataFrames(mxd)[0]
96. layers = map.ListLayers(mxd,"",dataframe)
97. **if** btn\_layer.isChecked()==True:
99. # Check out any necessary licenses
100. arcpy.CheckOutExtension("3D")
101. arcpy.CheckOutExtension('Spatial')
102. arcpy.env.overwriteOutput=True
104. #Join Input data
105. progressBar.setValue(1)
107. shp\_path = str(input\_files\_path)
109. inFeatures = shp\_path
110. weather\_variables = ['temp\_mit', 'temp\_max', 'temp\_min', 'prec', 'prec', 'hume', 'pres']
111. vble = weather\_variables[i]
112. Z\_value = vble
113. **print** Z\_value
114. outRaster = "interpolation\_maps/"+str(vble)
115. cellSize = 300
116. kModel = "Spherical 663.040376"
118. **for** layer **in** layers: #(group of layer name is read as one layer else)
119. **if** layer.name == administrative\_layer\_list[2].text():
120. arcpy.env.mask = layer.dataSource
121. arcpy.env.extent = "626638.8673 4190956.536 815377.0266 4519162.5229"
122. #arcpy.Kriging\_3d(inFeatures, Z\_value, outRaster, kModel, cellSize, "VARIABLE 12", "")
123. arcpy.Idw\_3d(inFeatures, Z\_value, outRaster, cellSize, "2", "VARIABLE 12", "")
124. #In order to automatically change the Source directory of the .shp inside the .mxd file
125. last\_workspace\_path = str(arcpy.env.workspace)
126. mxd.findAndReplaceWorkspacePaths(last\_workspace\_path,arcpy.env.workspace)

129. #Enable weather layer
130. progressBar.setValue(3)
131. **for** layer **in** layers: #(group of layer name is read as one layer else)
132. check\_enable\_layer (layer, btn\_layer, True) #function created before
134. #Enable other layers
135. progressBar.setValue(4)
136. **for** administrative\_layer **in** administrative\_layer\_list:
137. check\_enable\_layer(layer, administrative\_layer, True)
138. **for** other\_layer **in** other\_layer\_list:
139. check\_enable\_layer(layer, other\_layer, True)

142. #Draws graphic elements
143. progressBar.setValue(5)
144. draw\_graphic\_maps(mxd, graphic\_list) #function created before
146. #Zoom to
147. progressBar.setValue(6)
148. zoom\_to\_value(dataframe, btn\_zoom\_to\_layer\_list, layers, \
149. btn\_zoom\_to\_layer\_attribute\_list) #function created before
151. #Export map
152. progressBar.setValue(7)
153. export\_maps(mxd, btn\_layer, btn\_export\_pdf, \
154. btn\_export\_png) #function created before
156. #Disable layers
157. **for** layer **in** layers: #(group of layer name is read as one layer else)
158. check\_enable\_layer (layer, btn\_layer, False)
159. **for** administrative\_layer **in** administrative\_layer\_list:
160. check\_enable\_layer(layer, administrative\_layer, False)
161. **for** other\_layer **in** other\_layer\_list:
162. check\_enable\_layer(layer, other\_layer, False)
164. #################
165. ### INTERFACE ###
166. #################
168. #load the file .ui
169. form = uic.loadUiType("MapsGenerator.ui")[0]
171. **class** MyDialogClass(QtGui.QDialog, form):
172. **def** \_\_init\_\_(self,parent=None):
173. QtGui.QDialog.\_\_init\_\_(self,parent)
174. self.setupUi(self)
176. #connections of events with functions
177. self.btn\_shp\_path.clicked.connect(self.get\_shp\_file)
178. self.btn\_run.clicked.connect(self.run)
179. self.btn\_cancel.clicked.connect(self.cancel)
181. #Progress bar
182. self.progressBar.setMinimum(0)
183. self.progressBar.setMaximum(7)
185. #Enable and disable widgets automaticaly
186. self.check\_temp\_mit.clicked.connect(self.enable\_choose\_map\_buttons)
187. self.check\_temp\_max.clicked.connect(self.enable\_choose\_map\_buttons)
188. self.check\_temp\_min.clicked.connect(self.enable\_choose\_map\_buttons)
189. self.check\_prec50.clicked.connect(self.enable\_choose\_map\_buttons)
190. self.check\_prec150.clicked.connect(self.enable\_choose\_map\_buttons)
191. self.check\_hume.clicked.connect(self.enable\_choose\_map\_buttons)
192. self.check\_pres.clicked.connect(self.enable\_choose\_map\_buttons)
194. #Enable and disable zoom widgets automaticaly
195. self.rbtn\_zoom\_to\_regional.clicked.connect(self.enable\_zoom\_values\_cmb)
196. self.rbtn\_zoom\_to\_province.clicked.connect(self.enable\_zoom\_values\_cmb)
197. self.rbtn\_zoom\_to\_comarca.clicked.connect(self.enable\_zoom\_values\_cmb)
198. #self.rbtn\_zoom\_to\_municipality.clicked.connect(self.enable\_zoom\_values\_cmb)
200. #Load comarca values for the Qcombobox of zooming
201. self.rbtn\_zoom\_to\_comarca.clicked.connect(self.load\_comarca\_values\_cmb)
202. #self.rbtn\_zoom\_to\_municipality.clicked.connect(self.load\_municipality\_values\_cmb)

205. ####################################
206. ### ENABLE AND DISABLE FUNCTIONS ###
207. ####################################
209. **def** enable\_choose\_map\_buttons(self):
210. #Temp\_mit
211. **if** self.check\_temp\_mit.isChecked() == False:
212. self.check\_temp\_mit\_png.setEnabled(False)
213. self.check\_temp\_mit\_pdf.setEnabled(False)
214. self.lbl\_temp\_mit.setEnabled(False)
215. **elif** self.check\_temp\_mit.isChecked() == True:
216. self.check\_temp\_mit\_png.setEnabled(True)
217. self.check\_temp\_mit\_pdf.setEnabled(True)
218. self.lbl\_temp\_mit.setEnabled(True)

221. #Temp\_max
222. **if** self.check\_temp\_max.isChecked() == False:
223. self.check\_temp\_max\_png.setEnabled(False)
224. self.check\_temp\_max\_pdf.setEnabled(False)
225. self.lbl\_temp\_max.setEnabled(False)
226. **elif** self.check\_temp\_max.isChecked() == True:
227. self.check\_temp\_max\_png.setEnabled(True)
228. self.check\_temp\_max\_pdf.setEnabled(True)
229. self.lbl\_temp\_max.setEnabled(True)
231. #Temp\_min
232. **if** self.check\_temp\_min.isChecked() == False:
233. self.check\_temp\_min\_png.setEnabled(False)
234. self.check\_temp\_min\_pdf.setEnabled(False)
235. self.lbl\_temp\_min.setEnabled(False)
236. **elif** self.check\_temp\_min.isChecked() == True:
237. self.check\_temp\_min\_png.setEnabled(True)
238. self.check\_temp\_min\_pdf.setEnabled(True)
239. self.lbl\_temp\_min.setEnabled(True)
241. #Prec\_50
242. **if** self.check\_prec50.isChecked() == False:
243. self.check\_prec50\_png.setEnabled(False)
244. self.check\_prec50\_pdf.setEnabled(False)
245. self.lbl\_prec50.setEnabled(False)
246. **elif** self.check\_prec50.isChecked() == True:
247. self.check\_prec50\_png.setEnabled(True)
248. self.check\_prec50\_pdf.setEnabled(True)
249. self.lbl\_prec50.setEnabled(True)
251. #Prec\_150
252. **if** self.check\_prec150.isChecked() == False:
253. self.check\_prec150\_png.setEnabled(False)
254. self.check\_prec150\_pdf.setEnabled(False)
255. self.lbl\_prec150.setEnabled(False)
256. **elif** self.check\_prec150.isChecked() == True:
257. self.check\_prec150\_png.setEnabled(True)
258. self.check\_prec150\_pdf.setEnabled(True)
259. self.lbl\_prec150.setEnabled(True)
260. #Hume
261. **if** self.check\_hume.isChecked() == False:
262. self.check\_hume\_png.setEnabled(False)
263. self.check\_hume\_pdf.setEnabled(False)
264. self.lbl\_hume.setEnabled(False)
265. **elif** self.check\_hume.isChecked() == True:
266. self.check\_hume\_png.setEnabled(True)
267. self.check\_hume\_pdf.setEnabled(True)
268. self.lbl\_hume.setEnabled(True)
270. #Pres
271. **if** self.check\_pres.isChecked() == False:
272. self.check\_pres\_png.setEnabled(False)
273. self.check\_pres\_pdf.setEnabled(False)
274. self.lbl\_pres.setEnabled(False)
275. **elif** self.check\_pres.isChecked() == True:
276. self.check\_pres\_png.setEnabled(True)
277. self.check\_pres\_pdf.setEnabled(True)
278. self.lbl\_pres.setEnabled(True)
280. #ZOOM ENABLING FUNCTIONS
282. **def** enable\_zoom\_values\_cmb(self):
283. #Zoom to region
284. **if** self.rbtn\_zoom\_to\_regional.isChecked() == True:
285. self.cmb\_zoom\_to\_region.setEnabled(True)
286. self.cmb\_zoom\_to\_province.setEnabled(False)
287. self.cmb\_zoom\_to\_comarca.setEnabled(False)
288. self.cmb\_zoom\_to\_municipality.setEnabled(False)
289. #Zoom to province
290. **elif** self.rbtn\_zoom\_to\_province.isChecked() == True:
291. self.cmb\_zoom\_to\_region.setEnabled(False)
292. self.cmb\_zoom\_to\_province.setEnabled(True)
293. self.cmb\_zoom\_to\_comarca.setEnabled(False)
294. self.cmb\_zoom\_to\_municipality.setEnabled(False)
295. #Zoom to comarca
296. **elif** self.rbtn\_zoom\_to\_comarca.isChecked() == True:
297. self.cmb\_zoom\_to\_region.setEnabled(False)
298. self.cmb\_zoom\_to\_province.setEnabled(False)
299. self.cmb\_zoom\_to\_comarca.setEnabled(True)
300. self.cmb\_zoom\_to\_municipality.setEnabled(False)
302. ##        #Zoom to municipality
303. ##        elif self.rbtn\_zoom\_to\_municipality.isChecked() == True:
304. ##            self.cmb\_zoom\_to\_region.setEnabled(False)
305. ##            self.cmb\_zoom\_to\_province.setEnabled(False)
306. ##            self.cmb\_zoom\_to\_comarca.setEnabled(False)
307. ##            self.cmb\_zoom\_to\_municipality.setEnabled(True)

310. **def** load\_comarca\_values\_cmb(self):
311. list1 = []
312. #Acces to the MDX file
313. path\_mxd = r"MapasInterpolacion.mxd"
314. mxd = map.MapDocument(path\_mxd)
315. #Acces to the list of layers
316. dataframe = map.ListDataFrames(mxd)[0]
317. layers = map.ListLayers(mxd,"",dataframe)
318. **for** layer **in** layers:
319. #if layer.name == btn\_layer.text():
320. **if** layer.name == self.rbtn\_zoom\_to\_comarca.text():
321. cursor = arcpy.da.SearchCursor(layer, ['COMARCA\_'])
322. self.cmb\_zoom\_to\_comarca.clear()
323. **for** row **in** cursor:
324. self.cmb\_zoom\_to\_comarca.addItems(row)

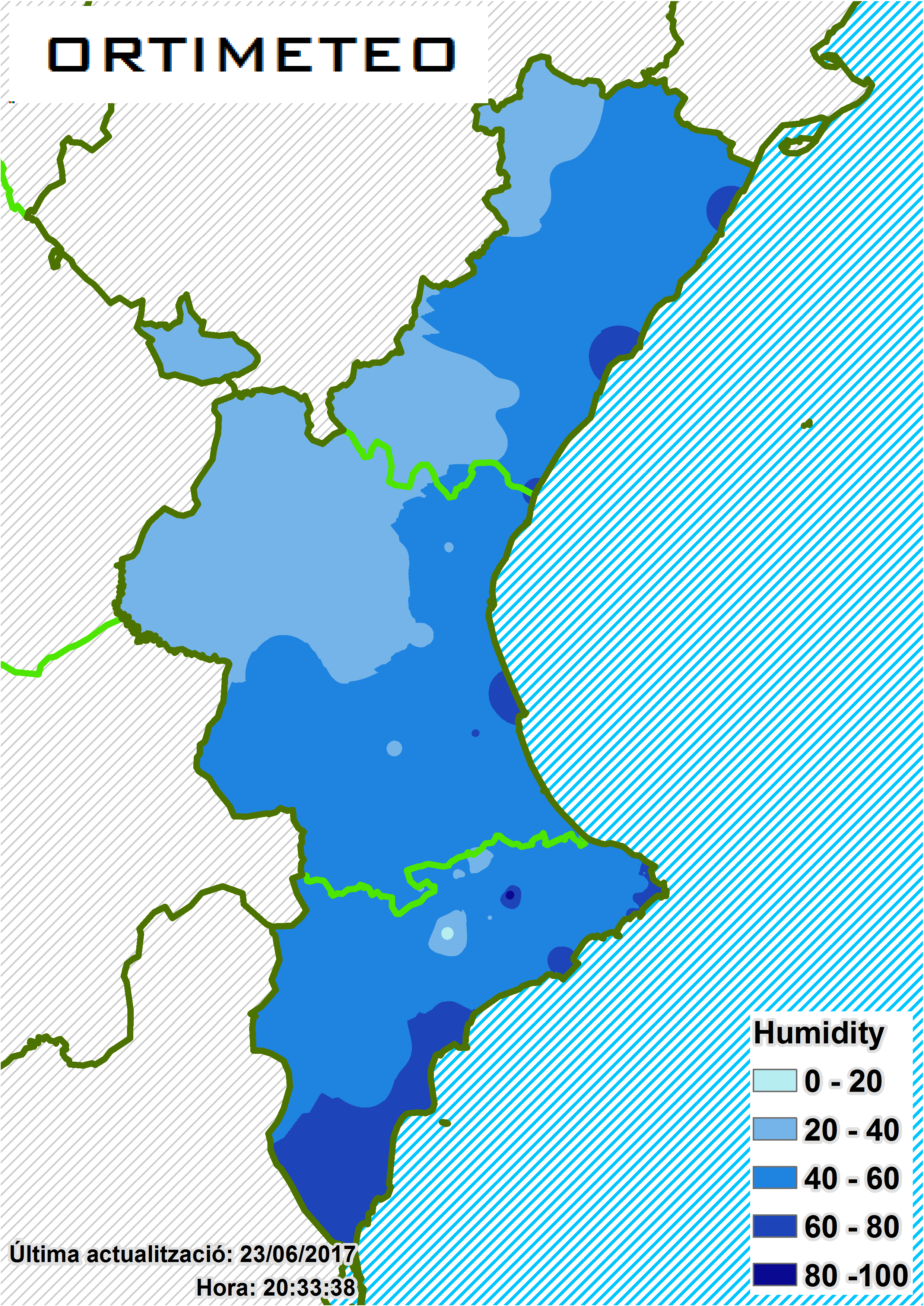
327. ########################
328. ### BUTTON FUNCTIONS ###
329. ########################
330. **def** get\_shp\_file(self):
331. #Open file. Only SHP files can be chosen
332. fname = QFileDialog.getOpenFileName(self, 'Open file', '.',\
333. "Shapefile /files (\*.shp)")
334. #Get SHP file path
335. self.txt\_shp\_path.setText(fname)
336. #Enable MAP OPTIONS
337. self.tabWidget.setEnabled(True)
338. self.btn\_run.setEnabled(True)
340. **def** run(self):
341. #Acces to the MDX file
342. path\_mxd = r"MapasInterpolacion.mxd"
343. mxd = map.MapDocument(path\_mxd)
345. #Save buttons on lists
346. progressBar = [self.progressBar]
348. input\_files\_path = [self.txt\_shp\_path.text()]
350. weather\_layer\_list = [self.check\_temp\_mit, self.check\_temp\_max, \
351. self.check\_temp\_min, self.check\_prec50, \
352. self.check\_prec150, self.check\_hume, \
353. self.check\_pres]
355. btn\_png\_list = [self.check\_temp\_mit\_png, self.check\_temp\_max\_png, \
356. self.check\_temp\_min\_png, self.check\_prec50\_png, \
357. self.check\_prec150\_png, self.check\_hume\_png, \
358. self.check\_pres\_png]
359. btn\_pdf\_list = [self.check\_temp\_mit\_pdf, self.check\_temp\_max\_pdf, \
360. self.check\_temp\_min\_pdf, self.check\_prec50\_pdf, \
361. self.check\_prec150\_pdf, self.check\_hume\_pdf, \
362. self.check\_pres\_pdf]
363. administrative\_layer\_list = [self.check\_insert\_regions\_shp, \
364. self.check\_insert\_provinces\_shp, \
365. self.check\_insert\_comarcas\_shp, \
366. self.check\_insert\_municipalities\_shp]
367. other\_layer\_list = [self.check\_insert\_hydro\_shp, \
368. self.check\_insert\_cities\_shp, \
369. self.check\_insert\_stations\_shp, \
370. self.check\_insert\_hillshade\_shp]
371. graphic\_list = [self.check\_logo, self.check\_date, \
372. self.check\_legend, self.check\_scale]
374. btn\_zoom\_to\_layer\_list = [self.rbtn\_zoom\_to\_regional, \
375. self.rbtn\_zoom\_to\_province, \
376. self.rbtn\_zoom\_to\_comarca, \
377. self.rbtn\_zoom\_to\_municipality]
379. btn\_zoom\_to\_layer\_attribute\_list= [self.cmb\_zoom\_to\_region, \
380. self.cmb\_zoom\_to\_province, \
381. self.cmb\_zoom\_to\_comarca, \
382. self.cmb\_zoom\_to\_municipality]
384. i = 0 #Contador
385. **for** weather\_layer **in** weather\_layer\_list:
386. #Create thematic maps
387. create\_thematic\_map(progressBar[0], mxd, input\_files\_path[0], i, weather\_layer, btn\_pdf\_list[i], \
388. btn\_png\_list[i], administrative\_layer\_list, \
389. other\_layer\_list, graphic\_list, \
390. btn\_zoom\_to\_layer\_list, \
391. btn\_zoom\_to\_layer\_attribute\_list)
392. i += 1

395. **def** cancel(self):
396. self.close()

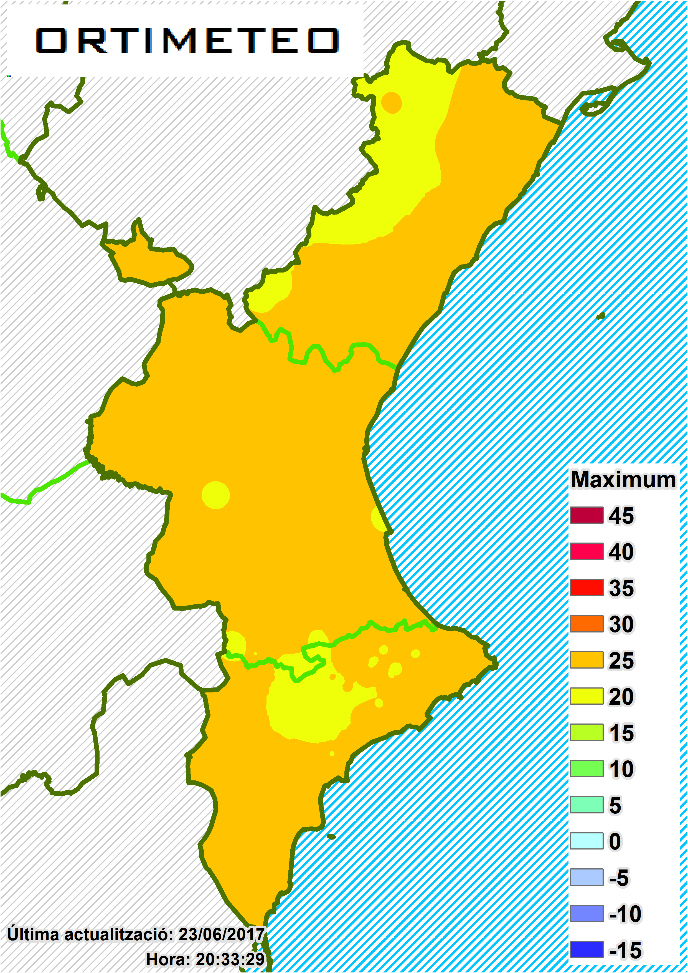
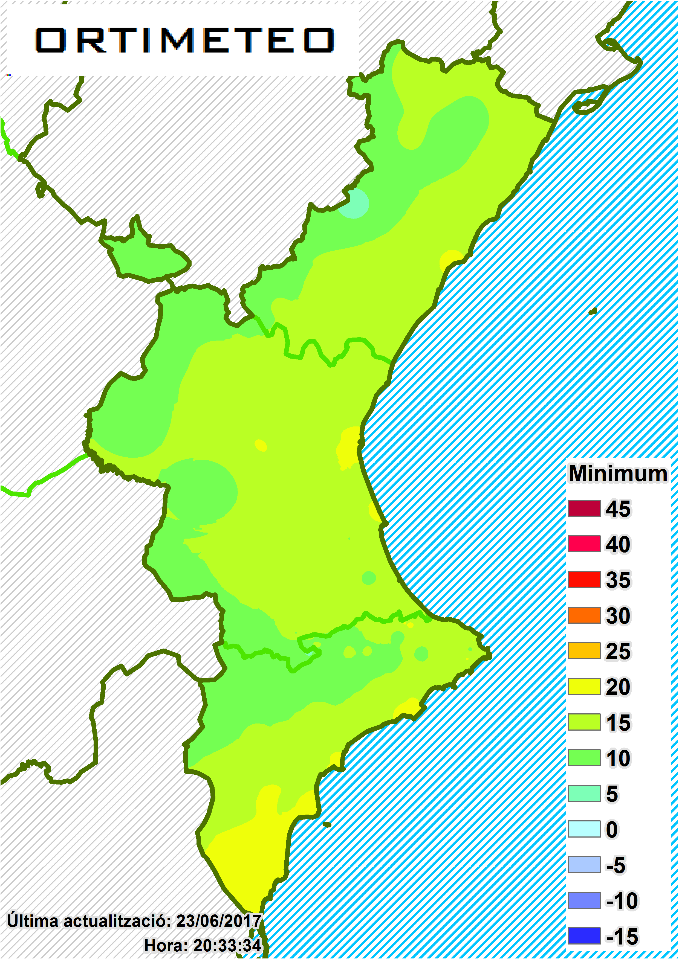
399. app = QtGui.QApplication(sys.argv)
400. myDialog = MyDialogClass(None)
401. myDialog.show()
402. app.exec\_()

# RESULTS

The tool was tested several times based on shapefile “EstacionesAVAMET\_datos”, what contais 144 records from weather stations (after join). On the picture below (Figure 11) can be seen the first result – interpolation of humidity.



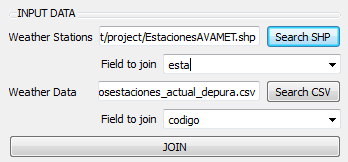
*Figure 11: humidity weather map*

Next were tested temperature in its maximum and minimum (*Figure 12*)

*Figure 12: minimum and maximum temperature maps*

# CONCLUSION

The application, what was created, can find a wide use in different fields of research. Firstly, it is a pity, that the join function wasn’t performed. This would simplify the proces on one level higher (*Figure 12*). Unfortunatelly, even it was managed to load both shp and csv shapefile and get their field names, the join function itself didn’t work. And after browsing discussing forums, it was found out, that there is huge amount of people whom both function what can perform join in arcmap don’t work.



*Figure 12: permorming join*

But even without join function (it can be considered, that the user is able to perform join function in GIS software very easily) can bet his application very helpful. But still, there are possibilities, how to improve this tool, for example creating map with different interpolation algorithms, in case of kriging compute the uncertaitity of interpolation and so on.

In conclusion, an application was created. As input is neccessary to use already joined data of meteorological stations and after that, can be chosen what variable user want to interpolate (can be interpolated all in the same time).

# BIBLIOGRAPHY

* LLOYD, C.D a P.M ATKINSON. Assessing uncertainty in estimates with ordinary and indicator kriging. *Computers & Geosciences* [online]. 2001, **27**(8), 929-937 [cit. 2017-06-07]. DOI: 10.1016/S0098-3004(00)00132-1. ISSN 00983004.
* Turning off layers in ArcMap via ArcPy? *GIS Stack Exchange* [online]. [cit. 2017-06-21]. Available at: <https://gis.stackexchange.com/questions/73433/turning-off-layers-in-arcmap-via-arcpy>
* Using Arcpy to zoom to selected feature? *GIS Stack Exchange* [online]. [cit. 2017-06-21]. Available at: <https://gis.stackexchange.com/questions/1711/using-arcpy-to-zoom-to-selected-feature>
* How to get content to ComboBox. *Stack Overflow* [online]. [cit. 2017-06-21]. Available at: <https://stackoverflow.com/questions/6061893/how-do-you-get-the-current-text-contents-of-a-qcombobox>
* Improve you python classes. *Jeff Knupp: python programmer* [online]. [cit. 2017-06-21]. Available at: <https://jeffknupp.com/blog/2014/06/18/improve-your-python-python-classes-and-object-oriented-programming/>
* Technical Vignette 3:: Kriging, interpolation, and uncertainty. *Berkley university* [online]. [cit. 2017-06-21]. Available at: <http://www.stat.berkeley.edu/~paciorek/research/techVignettes/techVignette3.pdf>
* Geostatistical Analyst Tutorial. *ArcGIS Help* [online]. [cit. 2017-06-21]. Available at: http://help.arcgis.com/en/arcgisdesktop/10.0/pdf/geostatistical-analyst-tutorial.pdf
* Exercise 40: Modules, Classes, and Objects. *Learn Python* [online]. [cit. 2017-06-21]. Available at: <https://learnpythonthehardway.org/book/ex40.html>
* Writing a QTableWidget to a .csv or .xls. *Stack Overflow* [online]. [cit. 2017-06-21]. Available at: <https://stackoverflow.com/questions/12608835/writing-a-qtablewidget-to-a-csv-or-xls>
* Change QLabel text dynamically in PyQt4. *Stack Overflow* [online]. [cit. 2017-06-21]. Available at: <https://stackoverflow.com/questions/12608835/writing-a-qtablewidget-to-a-csv-or-xls>
* Problem adding a join to a table via Python using addJoin\_management. *GIS Stack Exchange* [online]. [cit. 2017-06-21]. Available at: https://gis.stackexchange.com/questions/71116/problem-adding-a-join-to-a-table-via-python-using-addjoin-management
* How do you get the current text contents of a QComboBox? *Stack Overflow* [online]. [cit. 2017-06-21]. Available at: <https://stackoverflow.com/questions/6061893/how-do-you-get-the-current-text-contents-of-a-qcombobox>
* Select Layer By Attribute. *ArcGIS for Desktop* [online]. [cit. 2017-06-21]. Available at: <http://desktop.arcgis.com/en/arcmap/10.3/tools/data-management-toolbox/select-layer-by-attribute.htm>
* Export to PNG. *ArcGIS for Desktop* [online]. [cit. 2017-06-21]. Available at: <http://desktop.arcgis.com/es/arcmap/10.3/analyze/arcpy-mapping/exporttopng.htm>
* QProgressBar. *PyQt4* [online]. [cit. 2017-06-21]. Available at: <http://pyqt.sourceforge.net/Docs/PyQt4/qprogressbar.html#setFormat>
* Syntax Highlight Code In Word Documents. *Planet B* [online]. [cit. 2017-06-21]. Available at: <http://www.planetb.ca/syntax-highlight-word>