

Plugin para determinación de riesgo de inundación en Colegios

PROGRAMACIÓN OPEN SOURCE

CESAR BERNARDO CORTES

MASTER EN GEOTECNOLOGÍAS CARTOGRÁFICAS – 2019-2021

TABLA DE CONTENIDO

1	OBJETIVO DEL PROYECTO	3
2	PLANTEAMIENTO DEL ALCANCE DEL PROYECTO	3
3	DESARROLLO DEL PLUGIN	4
3.1	Entrada de la Información geográfica y alfanumérica	4
3.2	Ejecución de Geoprocesos	7
3.2.1	Generación de Buffer	7
3.2.2	Generación de Intersección	9
4	PUBLICACIÓN DE LA APLICACIÓN EN GITHUB	12
5	CONCLUSIONES	13
6	ARCHIVOS ADJUNTOS CON LA ENTREGA	15

INDICE DE ILUSTRACIONES

ILUSTRACIÓN 1 - VUSUALIZACIÓN DE FORMULARIO PARA ENTRADA DE DATOS	6
ILUSTRACIÓN 2 - VISUALIZACIÓN DE VENTANA DE EJECUCIÓN CARGA DE DATOS	7
ILUSTRACIÓN 3 - ESTRUCTURA DE TABLA DE DATOS FINAL - COLEGIOS EN RIESGO	11
ILUSTRACIÓN 4 - SALIDA GRAFICA DEL PROCESO EJECUTADO DE CLASIFICACIÓN DEL RIESGO	12
ILUSTRACIÓN 5 - PUBLICACIÓN DE DOCUMENTOS EN GITHUB	13

1 Objetivo del proyecto

Definir una serie de procesos que permita con una información de entrada básica de entrada se definirá a través de la aplicación un proceso que permita establecer los colegios que están localizados dentro de un rango de riesgo según su cercanía a los ríos, al final del proceso, se deberá entregar un archivo geográfico de puntos con el cruce correspondientes de los procesos realizados con la información de entrada.

Es importante mencionar que el proceso se realizará empleando tres herramientas que se han definido como esenciales dentro del curso, QGis (En mi caso 3.4) que es el manejador de datos geográficos, QT que permite interrelacionar las librerías y definir la salida gráfica de la herramienta que se construya y el PyCharm para la definición del código de programación en Python

2 Planteamiento del Alcance del proyecto

Con el conocimiento básico en la programación, se planteó la definición de un plugin que contemplara las siguientes funcionalidades:

Información de entrada

- Archivo con la información general de colegios en formato texto (Archivo Plano)
- Nivel geográfico de ríos de Colombia (Formato Shape)
- Nivel geográfico de departamentos de Colombia (Formato Shape)

Carga de archivo plano

- Definir una ventana principal con un menú que permita entre otras leer un fichero TXT, que cargue en una tabla de datos una nube de puntos con coordenadas y atributos.
- Con los datos leídos crear una capa punto (Colegios) y desplegarlos con algunos atributos predefinidos.

Procesos Geoespaciales

- Cargar unos layer tipo línea (Rios) con datos predefinidos
- Generar un buffer con valores definidos por el usuario, para establecer niveles de riesgo.
- Hacer el cruce de los puntos del paso 1 para determinar cuántos fueron afectados por el buffer.

Salida de información

- Crear una capa con los afectados
- Exportar txt con los colegios por tipo de afectación

- Por el desconocimiento que tengo, puede ser que me esté excediendo en el alcance por lo que les agradezco sus comentarios y sugerencias.

Archivo de datos de partida

- Archivo TXT con atributos de coordenadas X,Y y algunos atributos de identificación de cada punto (Colegios).
- Archivo SHP con la capa de ríos (Formato polígono), podría servir en formato línea también pero la que tengo es tipo polígono.
- Archivo SHP con la capa de departamentos de Colombia

Archivo de datos de salida

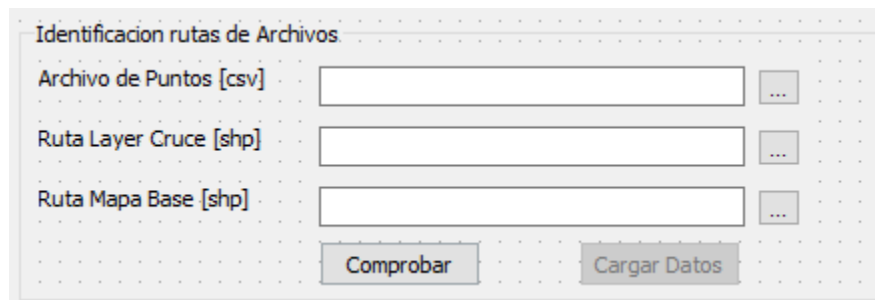
- Archivo plano txt
- Archivo SHP con cruces

3 Desarrollo del plugin

Inicialmente con el apoyo de dos herramientas que vienen incluidas en la interfase del QGis se definió la librería inicial con la que se realizará todo el desarrollo, éstas fueron “Plugin builder 3” y “Plugin Reloader”, para el concepto de nuestra aplicación se estableció también que la realizaríamos como una ventana emergente y no una embebida en el lateral de la pantalla.

3.1 Entrada de la Información geográfica y alfanumérica

Para la entrada de la información, se estableció dentro del formulario la capturar de la información de las rutas de los archivos empleando QgsFileWidget



Como se puede observar el texto indicativo define el tipo de archivo y características de ingreso, posteriormente al ingreso de la información se hace una comprobación en cuanto a que se hallan ingresado las rutas.

El código para la ejecución de este componente está definido en el procedimiento que denominamos **recogerRuta** y contiene el siguiente código:

```
def recogerRuta(self):
    #if self.txt_ruta.setText(str(self.selector.filePath())) True
    csvPath= str(self.selector.filePath())
    #self.lineEdit.setText(csvPath)
    shp1Path= str(self.sel_cruce.filePath())
    shp2Path= str(self.sel_base.filePath())

    datos = "Ok"
    if csvPath != "": # Verifico si se cargó el archivo CSV

        if shp1Path != "": # Verifico si se cargó el archivo CSV

            if shp2Path != "": # Verifico si se cargó el archivo CSV
                QMessageBox.information(self, "Mensaje de Comprobacion",
                    "Comprobación Exitosa")

            else:
                QMessageBox.warning(self, "Mensaje de Error",
                    "No lee la ruta del archivo base")
                datos = "Fallo"

        else:
            QMessageBox.critical(self, "Mensaje de Error", "No lee la ruta
del archivo de cruce")
            datos = "Fallo"

    else:
        QMessageBox.information(self, "Mensaje de Error", "No lee la ruta se
lee")
        datos = "Fallo"

    if datos == "Fallo":
        self.btn_carga.setEnabled(False)
    else:
        # Si las condiciones de carga se cumplieron activo el siguiente
        grupo de procesos
        self.btn_carga.setEnabled(True)
        self.groupBox_2.setEnabled(True)
        self.btn_geoproc.setEnabled(True)
        #self.error_carga()

    # Coloco la ruta seleccionada en el cuadro de texto, convertida en
    string
```

La ejecución de este componente arrojará un mensaje de éxito o fracaso según sea el caso

Master en Geotecnologías Cartográficas – Universidad de Salamanca

Programación Open Source

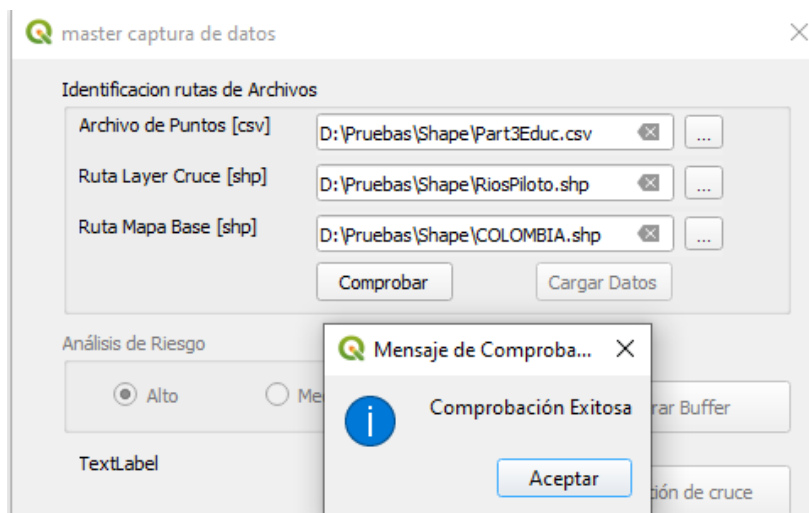


Ilustración 1 - Visualización de Formulario para entrada de datos

Luego de la realización de la comprobación se habilitará el botón **“Carga Datos”** que permitirá hacer la carga de los datos que adicionalmente realizará la conversión del archivo texto en un archivo de puntos con las coordenadas que vienen incluidas en éste código:

```
def cargadatos (self):
    rutaDP = str(self.sel_base.filePath())
    lydepto = QgsVectorLayer(rutaDP, "Departamentos", "ogr")
    QgsProject.instance().addMapLayer(lydepto)

    csvPath = str(self.selector.filePath())
    print(csvPath)
    uri =
    "file:///"+csvPath+"?delimiter=;&crs=EPSG:4686&xField=LONGITUD&yField=LATITUD&decimalPoint=,"

    vlayer = QgsVectorLayer(uri, 'Ptos Colegios', "delimitedtext")
    print(len(list(vlayer.getFeatures())))
    QgsProject.instance().addMapLayer(vlayer)

    #rutaRL = "D:\Pruebas\Shape\River_lines.shp"
    rutaRL = str(self.sel_cruce.filePath()) # asigno la ruta del layer de
    ríos según eleccion del usuario
    lyrios = QgsVectorLayer(rutaRL, "Rios Evaluación", "ogr")
    QgsProject.instance().addMapLayer(lyrios)
```

La visualización después de la ejecución de este proceso sería algo como esto:

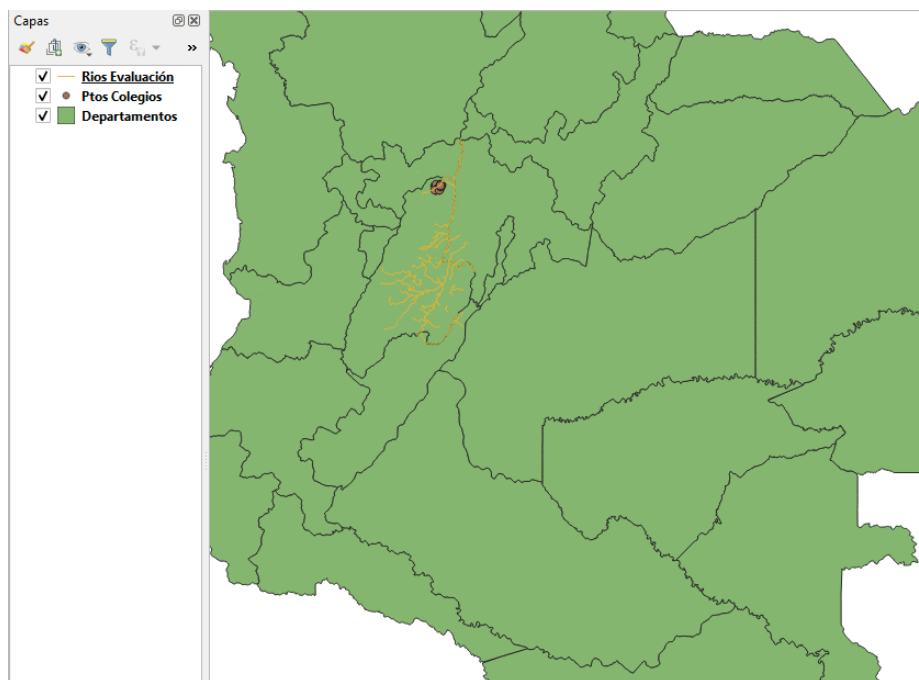


Ilustración 2 - Visualización de ventana de ejecución carga de datos

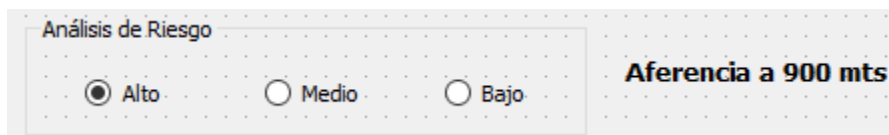
Hasta aquí se ha cumplido el primer proceso definido dentro del alcance propuesto de la aplicación.

3.2 Ejecución de Geoprocesos

Los geoprocesos a ejecutar se dividieron en dos, el primero de ellos es la generación de un buffer después de seleccionar un nivel de clasificación de riesgo, y el segundo será el cruce de la nueva capa generada a partir del buffer con la capa de puntos generada a partir del archivo texto.

3.2.1 Generación de Buffer

Para el primer geoproceso luego de la ejecución de la carga de los datos, se activará el botón “**Generar Buffer**”, que dependerá de la selección del nivel de riesgo, es decir a menor nivel de riesgo mas grande el buffer será. En la pantalla veremos algo así:



Cada vez que el usuario pique será ajustado el texto indicativo de la distancia aproximada a la que se ejecutará el buffer.

Y el código definido para su ejecución esta establecido en el procedimiento denominado “geoproceso1” y se escribió de la siguiente forma:

```
def geoproceso1(self):
    global riesgo
    file_output = str(self.FileOutput.filePath())
    if file_output=="":
        QMessageBox.information(self, "Aviso de error","Falta definir la
ruta del archivo buffer de salida")
    else:

        buff = 0.1
        if self.btns_alto.isChecked():
            buff = 0.008
            self.label_4.setText("Aferencia a 0.9 Km")
            riesgo = "Alto"
        elif self.btns_medio.isChecked():
            buff = 0.010
            self.label_4.setText("Aferencia a 1.1 Km")
            riesgo = "Medio"
        elif self.btns_bajo.isChecked():
            buff = 0.014
            self.label_4.setText("Aferencia a 1.5 Km")
            riesgo = "Bajo"

        # Acondicionado de ejercicios propuestos para la materia del
Programación Open Source - Maestría USAL

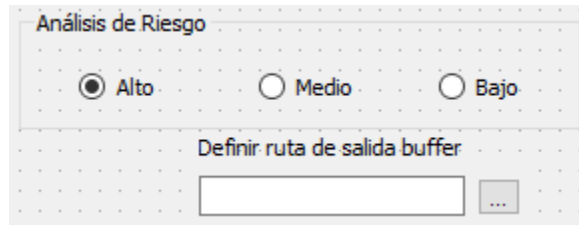
        layer = iface.activeLayer()
        #file_output = 'D:/Pruebas/Shape/lin_buffer1.shp'
        # Comprobación de lectura de ruta
        # QMessageBox.information(self, " Mensaje ruta", file_output)
        title = "Geoprocesamiento:Buffer"
        processing.run('native:buffer', {'INPUT': layer,
                                         'DISTANCE': buff,
                                         'SEGMENTS': 10,
                                         'DISSOLVE': True,
                                         'END_CAP_STYLE': 0,
                                         'JOIN_STYLE': 0,
                                         'MITER_LIMIT': 10,
                                         'OUTPUT': file_output})

        (ruta, shpname)= os.path.split(file_output)
        shpname = shpname.split('.')[0]
        # Divido el nombre del archivo y tomo solo el primer componente, nombre
        lyrBuffer = QgsVectorLayer(file_output, shpname, "ogr")
        # Guardo el archivo creado en la ruta seleccionado
        if lyrBuffer.isValid():
            QgsProject.instance().addMapLayers([lyrBuffer])
            # Agrega la capa generada al proyecto actual
            lyrBuffer.setOpacity(50)
            iface.mapCanvas().setExtent(lyrBuffer.extent())
            msg = "Buffer generado correctamente para riesgo "+ riesgo
            self.btn_geoproc_2.setEnabled(True)
            QMessageBox.information(iface.mainWindow(), title, msg)
        else:
            msg = "El Buffer no se ha podido generar"
```

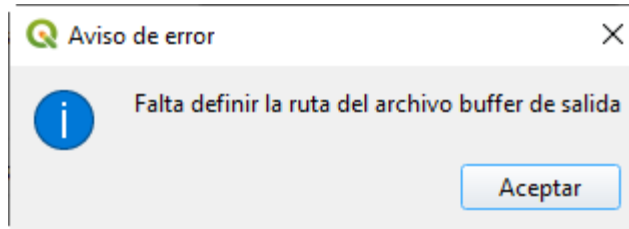


```
QMessageBox.information(iface.mainWindow(), title, msg)
# layer = iface.activeLayer("Rios Evaluación")
```

El usuario podrá elegir el sitio (Ruta) donde desea guardar el archivo generado con el buffer con un proceso de captura de esta información que se implementó con rutas relativas, para ello se dispuso la siguiente opción de captura de la información:



Adicionalmente en la misma opción se incorporó un proceso de validación que comprueba si el usuario incorporó o no la ruta y en caso que no sea así emitirá mensaje de error para corregir el proceso.



3.2.2 Generación de Intersección

Las capas generadas con el insumo para la utilización de esta ultima parte de la herramienta, hasta aquí tendremos una capa de puntos denominada “Ptos Colegios” y una capa tipo polígono con el buffer derivado de la elección del usuario “lin_Buffer1”, ahora lo que requeriremos será generar la tercera y ultima capa que será el producto de la intersección de estas dos, para este proceso que en el código se denominó **“geoproceso2”** tenemos escritas las siguientes líneas:

```
def geoproceso2 (self):

    # -*- coding: utf-8 -*-

    '''*****
    *****
    CBC INCORPORACION
    Adaptación de script:
    *****
    *****'''

    #pydevd.settrace('localhost', port=54100, stdoutToServer=True,
    stderrToServer=True)

    '''Primero creo una capa en la memoria para asignar los objetos que
    sean seleccionados desde la base
    Esta capa no contendrá los mismo campos solo:
```

```
IdCole, COD_SEDE, NombreSEDE, Coord_X y Coord_Y
'''

# Referencia de código en:
https://gis.stackexchange.com/questions/168266/pyqgis-a-geometry-
intersects-b-geometry-wouldnt-find-any-intersections
#def crealayer():
    qg=qgis.core
# Creación de la capa en memoria# Hago la lectura de las capas activas
mapcanvas1 = iface.mapCanvas()
# Asigno las capas activas a una lista denominada "layers"
layers = mapcanvas1.layers()
ly_intersect = qg.QgsVectorLayer("Point?crs=EPSG:4686", "Colegios en
Riesgo", "memory")
# Definición de los campos que llevará el layer "ly_intersect"
IdColeg = qg.QgsField("IdColeg", QVariant.Int)
CodSede = qg.QgsField("Cod_Sede", QVariant.String)
NomCole = qg.QgsField("NombreColegio", QVariant.String)
Coord_X = qg.QgsField("Coord_X", QVariant.Double)
Coord_Y = qg.QgsField("Coord_Y", QVariant.Double)
# Ahora asigno campos que llevará el layer "ly_intersect"
ly_intersect.dataProvider().addAttributes([IdColeg, CodSede, NomCole,
Coord_X, Coord_Y])
ly_intersect.updateFields()
# Verificación de la generación de los campos
#ly_intersect.fields().toList()
#ly_intersect.fields().toList()[1].name()

# Ahora genero una variable par apoder hacer la edición directa del layer
provider = ly_intersect.dataProvider
#campos = provider.fields()
seleccion = []
# Declaro los layer en una lista
cont=1
for f in layers[2].getFeatures(): # Layer 2 es Puntos Colegio
    for a in layers[0].getFeatures():
        if a.geometry().intersects(f.geometry()):
            # Valida si los elementos de las dos capas se intersectan,
            de ser así lo cargará en una nueva capa
            interseccion = a.geometry().intersection(f.geometry())
            #Creo un elemento de tipo QgsFeature para alimentarlo con
            los valores del elemento
            elemento = qg.QgsFeature()
            elemento.setFields(ly_intersect.fields())
            elemento.setGeometry(interseccion)
            # Busco el cada atributo del elemento "f" de la
            intersección y los asigno a la variable correspondiente
            elemento.setAttribute("IdColeg", cont) #
            elemento.setAttribute("Cod_Sede", f.attributes()[6])
            elemento.setAttribute("NombreColegio", f.attributes()[10])
            elemento.setAttribute("Coord_X", f.attributes()[12]) #
            Longitud
            elemento.setAttribute("Coord_Y", f.attributes()[11]) #
            Latitud

            print(f.attributes()[10])
            # Ahora los agrego a la capa de memoria
            cont =cont+1
```

```

        seleccion.append(elemento)
    ly_intersect.dataProvider().addFeatures(seleccion)
    #msg = "Se ha generado la capa - Colegios en Riesgo - correctamente"
    #QMessageBox.information( iface.mainWindow(), title, msg)
    ly_intersect.updateExtents()
    QgsProject.instance().addMapLayer(ly_intersect) # Agrego la capa
    ColRiesgo al proyecto
    print(len(seleccion))
    print(layers[2].selectedFeatureCount())
    # Establece la cuenta de elementos que cruzan
    print(layers[0].featureCount())

```

El resultado de este proceso es una capa nueva de tipo punto denominada “Colegios en Riesgo”, que contiene los “puntos” con la identificación de los colegios que se intersectaron con el buffer generado a la capa de rios, esta capa adicionalmente se ha configurado en el proceso con unos datos particulares de la capa origen (El archivo de texto inicialmente cargado), los atributos definidos para el ejercicio son:

- IdColeg : Identificador consecutivo de la capa, es un dato de tipo entero.
- Cod_Sede: Identifica la sede según los datos de entrada definidos para los colegios.
- Nombre Colegio: Almacena el nombre del colegio, dato de tipo String
- Coord_X: Entrega la longitud para el punto identificado, dato de tipo Double
- Coord_Y: Entrega la longitud para el punto identificado, dato de tipo Double

La visualización del producto de salida como se dijo anteriormente es una capa tipo punto que cuya tabla de datos presenta la siguiente estructura:

 Colegios en Riesgo :: Objetos totales: 19, Filtrados: 19, Seleccionados: 0

	IdColeg	Cod_Sede	NombreColegio	Coord_X	Coord_Y
1	1	17.344.300.027....	I.E.T. MORENO ...	-74,89	5,205
2	2	17.344.300.027....	ELIAS CAJELI B...	-74,89383	5,19968
3	3	17.344.300.027....	SAN DIEGO	-74,93304	5,18279
4	4	17.344.300.027....	SAN JUAN	-74,88806	5,21114
5	5	17.344.300.031....	I.E. GONZALO J...	-74,89853	5,19998
6	6	17.344.300.003....	I.E.T. FRANCISC...	-74,89853	5,19998
7	7	17.344.300.003....	POLICARPA SA...	-74,89111	5,21639
8	8	17.344.300.003....	RESU POLICARPA SALAVARRIETA	63	5,19616
9	9	17.344.300.002....	I.E. SANTA ANA...	-74,89833	5,19889
10	10	17.344.300.002....	CARLOTA ARM...	-74,89708	5,20163

Ilustración 3 - Estructura de tabla de datos final - Colegios en Riesgo

La anterior tabla es exportable por el usuario en el formato que desee con las funciones básicas del QGIS, este resultado fue el trazado originalmente con la propuesta del proyecto.

Adicionalmente el resultado del proceso completo de la ejecución efectuada se puede observar en la siguiente imagen:

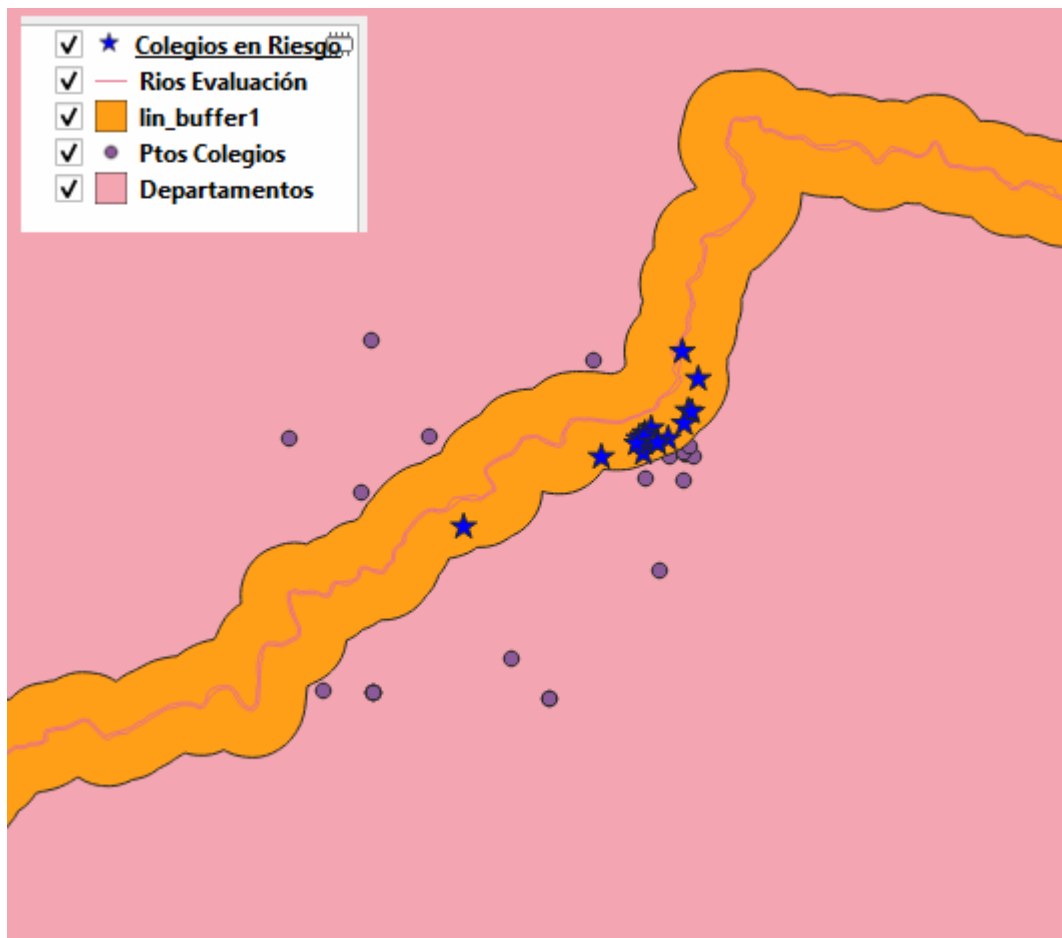


Ilustración 4 - Salida grafica del proceso ejecutado de clasificación del riesgo

Como se puede apreciar, la información del proceso es bastante comprensible de manera visual, permite establecer claramente el flujo y la comprobación de los resultados, por lo que se considera que su aplicación es especialmente sencilla de comprender para cualquier usuario.

4 Publicación de la aplicación en GitHub

La publicación de la aplicación en GitHub, es una de las mejores maneras de compartir el conocimiento y las lecciones aprendidas, por ello y dado uno de los requerimientos para la ejecución de este curso, se hizo la publicación del resultado final de este proyecto en la misma, el acceso a esta publicación se puede lograr a través del link

<https://github.com/cbcortesr/MaestriaUSAL> y pantallazo es como se muestra a continuación:

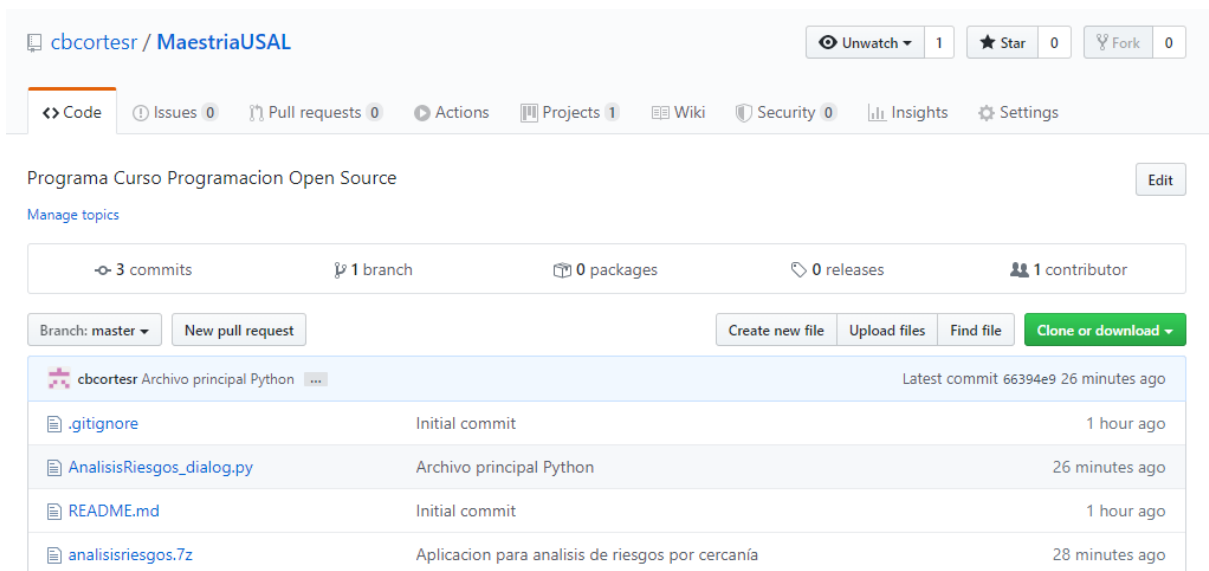


Ilustración 5 - Publicación de documentos en GitHub

5 Conclusiones

La programación en Python para principiantes me ha parecido un poco más compleja que la misma programación en C++, aunque no es tan rigurosa en la escritura de algunos caracteres, si que lo es en otros y esto puede hacer que se dificulten algunos procesos en la ejecución práctica de la programación.

La integración de las 3 herramientas (QGIS, Pycharm y QT Designer) facilita muchísimo el proceso de programación, sobre todo en lo que se refiere a la articulación de los componentes y la posibilidad de validar las líneas de comando incluidas de manera simultánea, esto hace que el ahorro en tiempos de procesamiento en pruebas sea sustancial.

El desarrollo de una aplicación es una de las maneras prácticas de poder lograr un aprendizaje en la programación, aunque la fundamentación en la estructura lógica de la construcción del lenguaje y de cómo funciona la API en el QGIS podría ser más fuerte al inicio del curso, esto facilitaría mucho más lograr resultados en el proceso de construcción del plugin.

6 Archivos Adjuntos con la entrega

Archivo	Contenido
Analisisriesgos.7z	Carpeta comprimida con el plugin de la aplicación.
Informe de ejecución Plugin.pdf	Informe de ejecución con el proceso descriptivo de la construcción de la aplicación.
Uso e instalación de la aplicación.pdf	Manual de Uso e instalación de la aplicación