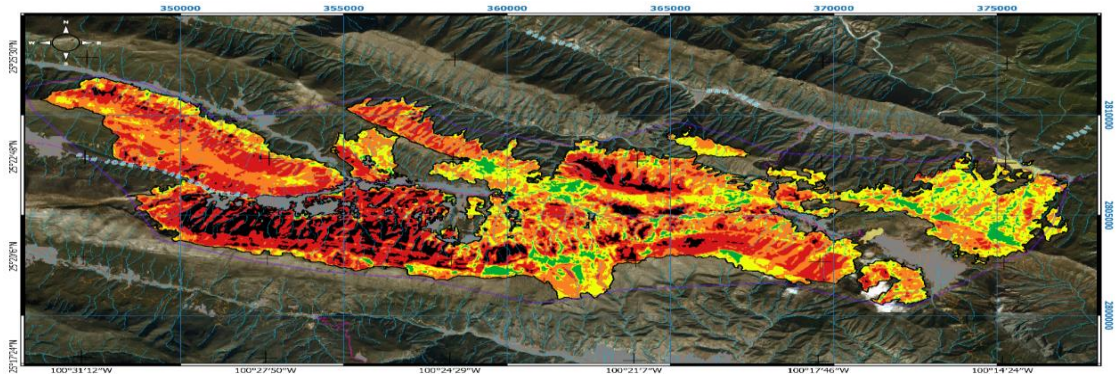
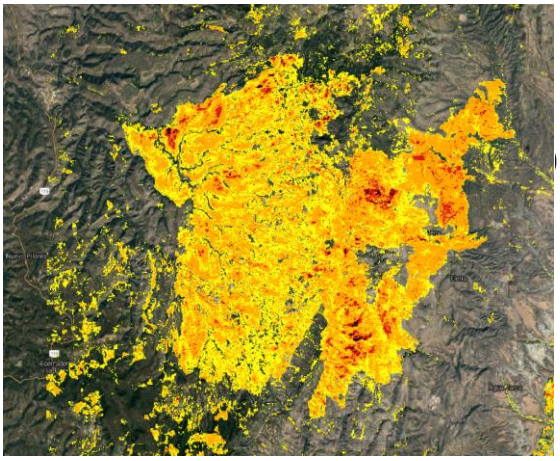
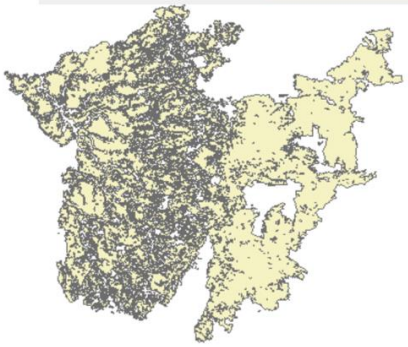
Anexo 2.1.2. Documentación y código del Plug in de Qgis 3.16 para el mapeo de área quemada a partir de imágenes Sentinel.





Shape *	Id	AQ_has
Polygon	0	25476



Informe del Proyecto "Reforzamiento al Sistema Nacional de Predicción de Peligro de Incendios Forestales de México para el pronóstico de conglomerados y área quemada", con clave B-S-131553, convocatoria CONAFOR-2018-C02

Fondo Sectorial para la Investigación, el Desarrollo y la Innovación Tecnológica Forestal CONAFOR-CONACYT

Autores: Dr. Daniel José Vega Nieva, Dr. Jaime Briseño Reyes, Mc. Carlos Briones Herrera, Ing. Adrián I. Silva-Cardoza, (FCF-UJED).

1. **<u>Documentación del proceso de cálculo del área quemada a partir de imágenes Sentinel.</u>**

El diagrama a continuación resume los procesos para el cálculo del polígono de área quemada a partir de una imagen raster del índice RBRc de Sentinel y un conglomerado de puntos de calor MODIS y VIIRS del incendio correspondiente, descargados siguiendo los procesos descritos en el Manual de usuario de las herramientas de mapeo de área quemada a partir de imágenes Sentinel (Briones-Herrera et al., 2021 (**Anexo 2.1.3).**

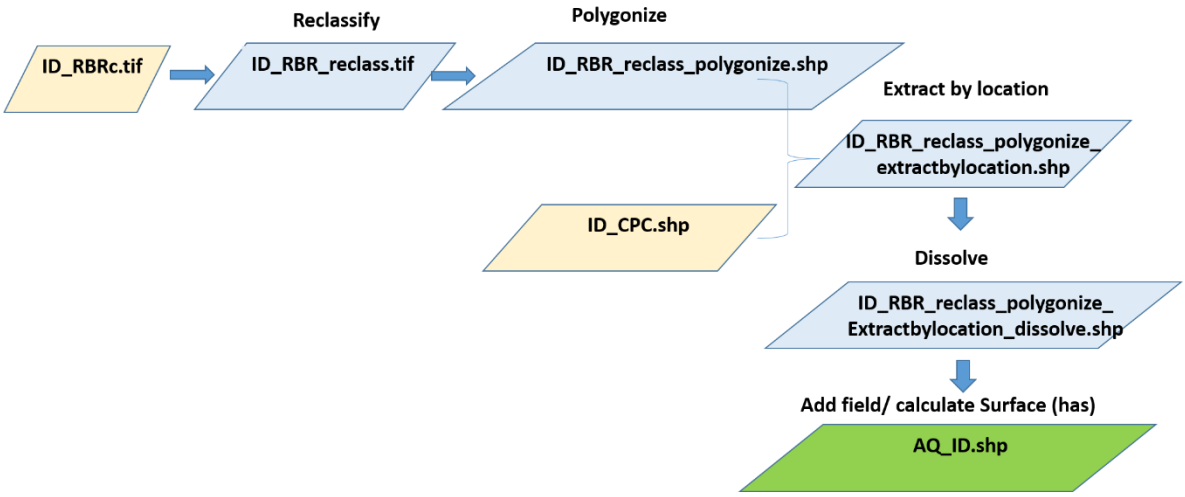


Diagrama general del Plug-in de Qgis3.16 para el cálculo de área quemada a partir de imágenes Sentinel 2. Se muestran en amarillo las entradas, en verde la salida. Los archivos mostrados en azul corresponden con procesos intermedios, que se guardan en la carpeta tmp.

Entradas:

1. **ID_RBRc.tif:** Archivo en formato .tif del índice RBRc para Sentinel 2, usando el código de descarga de Google Earth Engine (**Anexo 2.1.1 y 2.1.3**). Donde: ID es el identificador del conglomerado usado para la descarga (campo CPC_ID en los conglomerados del sistema de peligro de incendios de México).
2. **ID_CPC.shp:** Archivo en formato .shp del conglomerado de puntos de calor (CPC) de MODIS y VIIRS, descargado del sistema de peligro, para el incendio a analizar. Donde: ID es el identificador del conglomerado (campo CPC_ID).
   Nota: En caso de no existir un conglomerado de puntos de calor (para incendios de pequeño tamaño, generalmente de < 200 has), se puede emplear en su lugar un polígono medido en campo, o un buffer a una coordenada de registro del incendio, tal y como se describe en la sección 7 del Manual de usuario de las herramientas de mapeo de área quemada a partir de imágenes Sentinel (**Anexo 2.1.3**, Briones-Herrera et al., 2021).

Salidas:

1. **AQ_ID.shp:** Archivo en formato .shp del área quemada del incendio obtenida a partir de imágenes Sentinel. Donde: ID es el identificador del conglomerado usado para la descarga.

2. **Tmp.** Contiene los archivos temporales de los procesos intermedios del cálculo del área quemada (ID_RBR_reclass.shp, ID_RBR_reclass_polygonize.shp, ID_RBR_reclass_polygonize_extractbylocation.shp, ID_RBR_reclass_polygonize_extractbylocation_dissolve.shp)

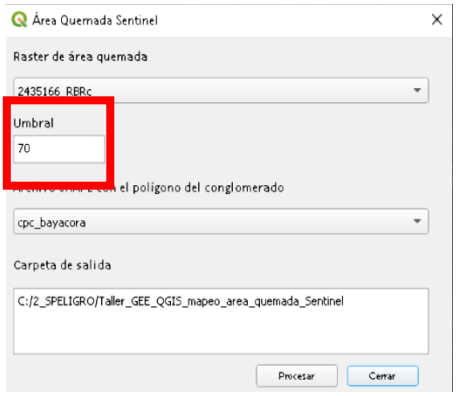
Subprocesos de cálculo de área quemada.

El cálculo del polígono de área quemada consta de los siguientes subprocesos, realizados de manera automatizada por el Plugin de Área Quemada para QGIS 3.16:

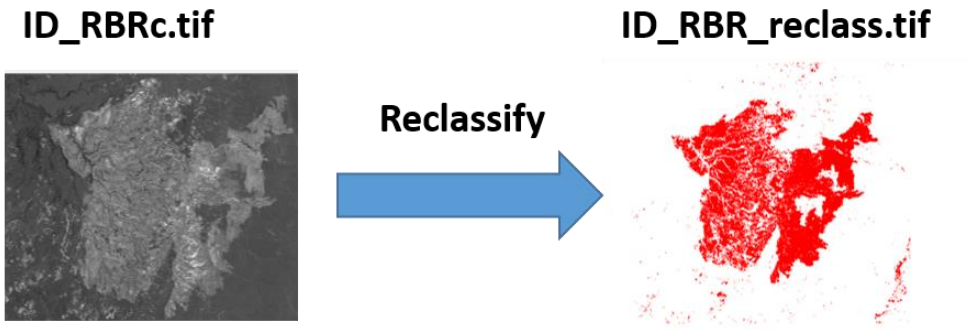
1. **Reclasificado** del ráster ID_RBRc.tif considerando un umbral de área quemada.
2. **Poligonización** (vectorización) del archivo raster reclasificado a formato shapefile.
3. **Intersección espacial** del shapefile del RBR reclasificado con el conglomerado de puntos de calor.
4. **Disolver** (dissolve) el polígono resultante.
5. **Cálculo de superficie** del polígono de área quemada.

Se describen a continuación estos procesos:

1. **Reclasificado del raster ID_RBRc.tif.**
   A partir de la imagen **ID_RBRc.tif,** el proceso de reclasificado devuelve un archivo temporal, **ID_RBR_reclass.tif,** con valor de 1 para los valores del raster iguales o superiores al **umbral** de área quemada, reclasificando como "no data" los valores inferiores al umbral de área quemada. El usuario especifica el umbral de área quemada en la ventana de la interfaz del plug-in:

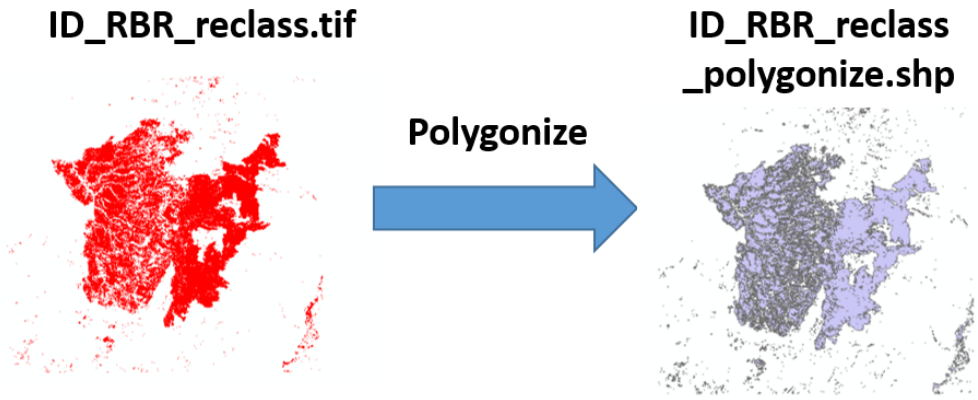


En base a las investigaciones de área quemada en México (Silva-Cardoza, 2021), se recomienda un umbral de área quemada de 50 a 100. Este valor óptimo de umbral de área quemada puede variar entre incendios, por lo que se sugiere una inspección visual del perímetro resultante, teniendo en cuenta la presencia de puntos de calor y posible información de campo.



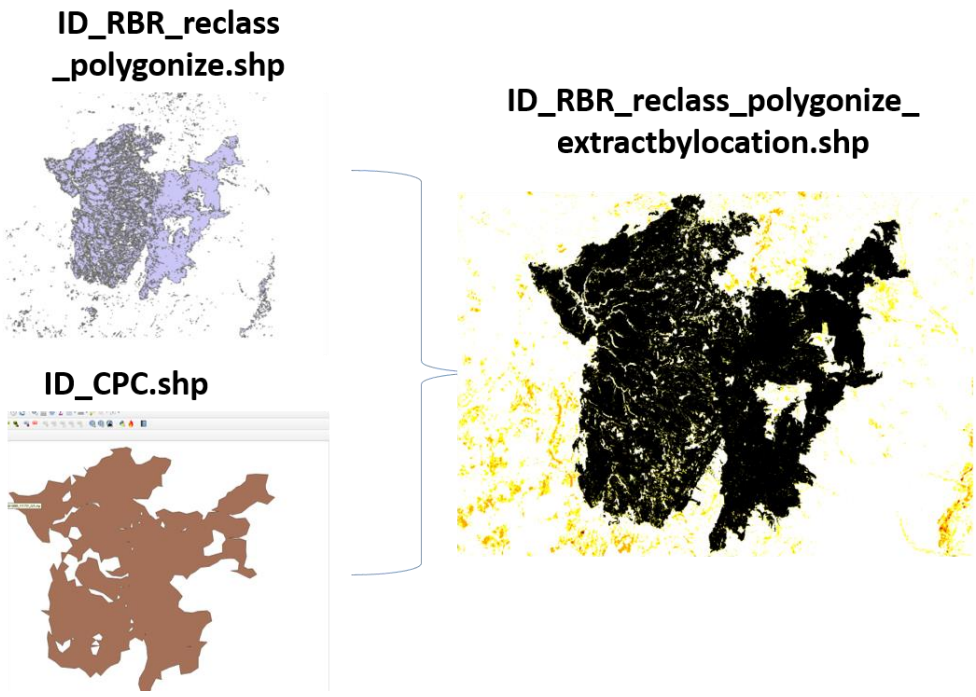
Proceso de reclasificado de la imagen raster ID_RBRc.tif.

2. **Poligonización (vectorización) del archivo raster reclasificado a formato shapefile.** Crea, a partir del archivo temporal , **ID_RBR_reclass.tif** del paso anterior, un nuevo archivo temporal, **ID_RBR_reclass_polygonize.shp,** en formato shapefile, del tipo polígono vectorial.



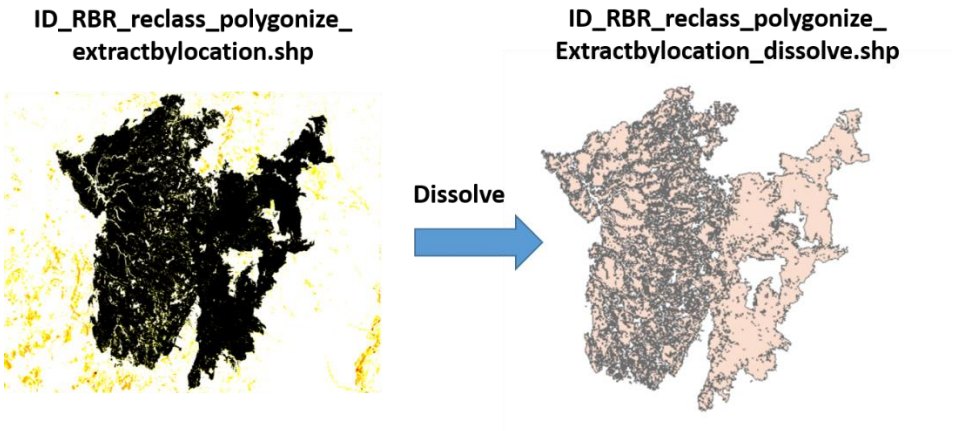
Proceso de poligonización del raster reclasificado ID_RBR_reclassify.tif.

3. **Intersección espacial del shapefile del RBR reclasificado con el conglomerado de puntos de calor.**
Selecciona espacialmente los polígonos del archivo **ID_RBR_reclass_polygonize.shp** que intersecan con el conglomerado de puntos de calor ID_CPC.shp y lo guarda como un nuevo archivo temporal, **ID_RBR_reclass_polygonize_extractbylocation.shp** en formato shapefile, del tipo polígono vectorial.



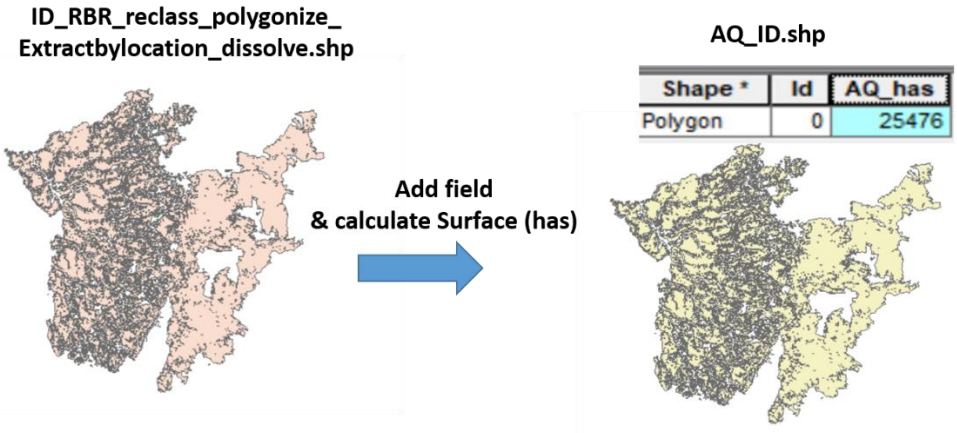
Proceso de intersección espacial del shapefile del RBR reclasificado con el conglomerado de puntos de calor.

4. **Disolver el polígono resultante.**
   En este proceso se disuelve el polígono del paso anterior, **ID_RBR_reclass_polygonize_extractbylocation.shp,** potencialmente integrado por varios polígonos adjuntos seleccionados, generando un nuevo archivo temporal que contiene un único polígono disuelto, **ID_RBR_reclass_polygonize_Extractbylocation_dissolve.shp,** en formato shapefile, del tipo polígono vectorial.



Disolución del polígono ID_RBR_reclass_polygonize_extractbylocation.shp

5. **Cálculo de superficie del polígono de área quemada.**
   Genera el archivo final de salida, con nombre **AQ_ID.shp,** (donde ID es el nombre del archivo de entrada) que se guarda en la carpeta de salida especificada. El archivo final AQ_ID contiene el **campo AQ_has,** que muestra la superficie del área quemada del incendio en hectáreas.



Cálculo de superficie del polígono de área quemada AQ_ID.shp.

Se puede consultar un ejemplo en el Manual de usuario de las herramientas de mapeo de área quemada a partir de imágenes Sentinel (Briones-Herrera et al., 2021) (**Anexo 2.1.3**).

Referencias.

Briones-Herrera, C.I.; Silva-Cardoza, A. Vega-Nieva, D.J.; Briseño-Reyes J. 2021. Manual de usuario de las herramientas de mapeo de área quemada y severidad de incendios forestales a partir de imágenes Sentinel. 45 pg. Anexo 2.1.3 al informe de la segunda etapa del Proyecto CONAFOR-CONACYT-2018-C02-B-S-131553 "Reforzamiento al sistema nacional de predicción de peligro de incendios forestales de México para el pronóstico de conglomerados y área quemada Disponible en: http://forestales.ujed.mx/incendios2/descargas/Manual%20AQ_Sentinel_v1_1_071021.pdf

Silva-Cardoza, A. 2021. Evaluación y mapeo de la severidad en la sierra madre occidental de Durango. Tesis de Maestría. 55 pg. Maestría en Geomática Aplicada a Recursos Ambientales y Forestales. Universidad Juárez del Estado de Durango.

2-ANEXO. Código de Plugin de Área Quemada Sentinel para QGis

Versión de Qgis requerida: 3.16 LTS

El Plugin está conformado por los siguientes carpetas y archivos.

```
25/05/2021  11:12 a. m.    <DIR>          help
24/05/2021  08:24 p. m.    <DIR>          i18n
25/05/2021  12:09 p. m.             3,849 icon.png
25/05/2021  11:12 a. m.             8,163 Makefile
16/07/2021  04:35 p. m.            24,154 mapeado_3.py
25/05/2021  11:12 a. m.             2,042 mapeado_3_dialog.py
06/07/2021  11:08 a. m.             6,046 mapeado_3_dialog_base.ui
25/05/2021  11:12 a. m.             1,064 metadata.txt
25/05/2021  11:42 a. m.             2,950 pb_tool.cfg
24/05/2021  08:24 p. m.             3,447 plugin_upload.py
24/05/2021  08:24 p. m.             8,798 pylintrc
25/05/2021  11:12 a. m.             1,920 README.html
25/05/2021  11:12 a. m.               966 README.txt
25/05/2021  12:17 p. m.            17,897 resources.py
25/05/2021  11:12 a. m.               105 resources.qrc
24/05/2021  08:24 p. m.    <DIR>          scripts
25/05/2021  11:12 a. m.    <DIR>          test
25/05/2021  11:12 a. m.             1,574 __init__.py
              15 archivos         82,975 bytes
```

Módulo *mapeado_3.py*:

```
# -*- coding: utf-8 -*-
"""
/***************************************************************************
 Mapeado3
                                 A QGIS plugin

 Sistema de Predicción de Peligro de Incendios Forestales de México (SPPIF)

   Mapeado de Area Quemada Sentinel

   Autor: Universidad Juárez del Estado de Durango
          Facultad de Ciencias Forestales
          Jaime Briseño Reyes <jaime.briseno@gmail.com>

   Fecha: 25/mayo/2021

  Generated by Plugin Builder: http://g-sherman.github.io/Qgis-Plugin-Builder/
                              -------------------
        begin                : 2021-05-25
        git sha              : $Format:%H$
        copyright            : (C) 2021 by Jaime Briseño R.
        email                : jaime.briseno@gmail.com
 ***************************************************************************/

/***************************************************************************
 *                                                                         *
 *   This program is free software; you can redistribute it and/or modify  *
 *   it under the terms of the GNU General Public License as published by  *
 *   the Free Software Foundation; either version 2 of the License, or     *
 *   (at your option) any later version.                                   *
 *                                                                         *
 ***************************************************************************/
"""
```

```
from qgis.PyQt.QtCore import QSettings, QTranslator, QCoreApplication
from qgis.PyQt.QtGui import QIcon
from qgis.PyQt.QtWidgets import QAction
from qgis.core import *

from qgis.PyQt.QtWidgets import QProgressBar
from qgis.PyQt.QtCore import *

# Initialize Qt resources from file resources.py
from .resources import *
# Import the code for the dialog
from .mapeado_3_dialog import Mapeado3Dialog
import os.path
import subprocess
import processing
import glob
import time
import shutil

from processing.algs.grass7.Grass7Utils import Grass7Utils
Grass7Utils.checkGrassIsInstalled()

class Mapeado3:
    """QGIS Plugin Implementation."""

    def __init__(self, iface):
        """Constructor.

        :param iface: An interface instance that will be passed to this class
            which provides the hook by which you can manipulate the QGIS
            application at run time.
        :type iface: QgsInterface
        """
        # Save reference to the QGIS interface
        self.iface = iface
        # initialize plugin directory
        self.plugin_dir = os.path.dirname(__file__)
        # initialize locale
        locale = QSettings().value('locale/userLocale')[0:2]
        locale_path = os.path.join(
            self.plugin_dir,
            'i18n',
            'Mapeado3_{}.qm'.format(locale))

        if os.path.exists(locale_path):
            self.translator = QTranslator()
            self.translator.load(locale_path)
            QCoreApplication.installTranslator(self.translator)

        # Declare instance attributes
        self.actions = []
        self.menu = self.tr(u'&Mapeado 3')

        # Check if plugin was started the first time in current QGIS session
        # Must be set in initGui() to survive plugin reloads
        self.first_start = None

    # noinspection PyMethodMayBeStatic
    def tr(self, message):
        """Get the translation for a string using Qt translation API.

        We implement this ourselves since we do not inherit QObject.

        :param message: String for translation.
```

```
        :type message: str, QString

        :returns: Translated version of message.
        :rtype: QString
        """
        # noinspection PyTypeChecker,PyArgumentList,PyCallByClass
        return QCoreApplication.translate('Mapeado3', message)


    def add_action(
        self,
        icon_path,
        text,
        callback,
        enabled_flag=True,
        add_to_menu=True,
        add_to_toolbar=True,
        status_tip=None,
        whats_this=None,
        parent=None):
        """Add a toolbar icon to the toolbar.

        :param icon_path: Path to the icon for this action. Can be a resource
            path (e.g. ':/plugins/foo/bar.png') or a normal file system path.
        :type icon_path: str

        :param text: Text that should be shown in menu items for this action.
        :type text: str

        :param callback: Function to be called when the action is triggered.
        :type callback: function

        :param enabled_flag: A flag indicating if the action should be enabled
            by default. Defaults to True.
        :type enabled_flag: bool

        :param add_to_menu: Flag indicating whether the action should also
            be added to the menu. Defaults to True.
        :type add_to_menu: bool

        :param add_to_toolbar: Flag indicating whether the action should also
            be added to the toolbar. Defaults to True.
        :type add_to_toolbar: bool

        :param status_tip: Optional text to show in a popup when mouse pointer
            hovers over the action.
        :type status_tip: str

        :param parent: Parent widget for the new action. Defaults None.
        :type parent: QWidget

        :param whats_this: Optional text to show in the status bar when the
            mouse pointer hovers over the action.

        :returns: The action that was created. Note that the action is also
            added to self.actions list.
        :rtype: QAction
        """
        self.dlg = Mapeado3Dialog() # Se instancia la clase del cuadro de diálogo

        icon = QIcon(icon_path)
        action = QAction(icon, text, parent)
        action.triggered.connect(callback)
        action.setEnabled(enabled_flag)
```

```
        if status_tip is not None:
            action.setStatusTip(status_tip)

        if whats_this is not None:
            action.setWhatsThis(whats_this)

        if add_to_toolbar:
            # Adds plugin icon to Plugins toolbar
            self.iface.addToolBarIcon(action)

        if add_to_menu:
            self.iface.addPluginToMenu(
                self.menu,
                action)

        self.actions.append(action)

        return action

    def initGui(self):
        """Create the menu entries and toolbar icons inside the QGIS GUI."""

        icon_path = ':/plugins/mapeado_3/icon.png'
        self.add_action(
            icon_path,
            text=self.tr(u'Área Quemada Sentinel'),
            callback=self.run,
            parent=self.iface.mainWindow())

        # will be set False in run()
        #self.first_start = True


        # Establece valor inicial del umbral
        self.dlg.textUmbral.setPlainText("100")
        # Inicializa acciones de botones
        self.dlg.cmbLayersRaster.currentIndexChanged.connect(self.cmbLayersRaster_onChange)
        self.dlg.cmbLayersShape.currentIndexChanged.connect(self.cmbLayersShape_onChange)
        self.dlg.btnProcess.clicked.connect(self.btnProcess_onClick)


    def unload(self):
        """Removes the plugin menu item and icon from QGIS GUI."""
        for action in self.actions:
            self.iface.removePluginMenu(
                self.tr(u'&Mapeado 3'),
                action)
            self.iface.removeToolBarIcon(action)


    def run(self):
        """Run method that performs all the real work"""

        # Valida que haya al menos dos capas cargadas en el área de trabajo
        if len(QgsProject.instance().layerTreeRoot().findLayers()) < 2:
            self.iface.messageBar().pushMessage("Error", "No se han cargado las capas !", level=Qgis.Critical)
            return 0

        # Muestra fecha/hora de inicio
        fecha = time.strftime("%d/%m/%Y")
        hora = time.strftime("%H:%M:%S")
        print("v 0.1 20210715.2006")
        print("Inicio: {} {}".format(fecha, hora))
```

```
        # Llena contenido de combo box de layers raster
        self.dlg.cmbLayersRaster.clear()
        layers = []
        for tree_layer in QgsProject.instance().layerTreeRoot().findLayers():
            layers.append(tree_layer.layer())
        layer_list = []
        for layer in layers:
            layer_list.append(layer.name())
        layer_list.insert(0,'Seleccione una capa raster')
        self.dlg.cmbLayersRaster.addItems(layer_list)

        # Llena contenido de combo box de layers shape
        self.dlg.cmbLayersShape.clear()
        layers = []
        for tree_layer in QgsProject.instance().layerTreeRoot().findLayers():
            layers.append(tree_layer.layer())
        layer_list = []
        for layer in layers:
            layer_list.append(layer.name())
        layer_list.insert(0,'Seleccione una capa vectorial')
        self.dlg.cmbLayersShape.addItems(layer_list)

        self.progressMessageBar = self.iface.messageBar().createMessage("Procesando...")
        self.progress = QProgressBar()
        self.progress.setMaximum(100)
        self.progress.setAlignment(Qt.AlignLeft|Qt.AlignVCenter)
        self.progressMessageBar.layout().addWidget(self.progress)
        self.iface.messageBar().pushMessage("Ok", "Se ha iniciado correctamente ...", level=Qgis.Info)

        # Create the dialog with elements (after translation) and keep reference
        # Only create GUI ONCE in callback, so that it will only load when the plugin is started
        """if self.first_start == True:
            self.first_start = False
            self.dlg = Mapeado3Dialog()"""

        # show the dialog
        self.dlg.show()
        # Run the dialog event loop
        result = self.dlg.exec_()
        # See if OK was pressed
        if result:
            # Do something useful here - delete the line containing pass and
            # substitute with your code.
            pass

    def cmbLayersShape_onChange(self):
        # Obtiene ruta/nombre de salida
        if (self.dlg.cmbLayersShape.currentText() != 'Seleccione una capa vectorial'):
            selectLayerShapeIndex = self.dlg.cmbLayersShape.currentIndex() - 1
            layers = [tree_layer.layer() for tree_layer in QgsProject.instance().layerTreeRoot().findLayers()]
            selectedLayer = layers[selectLayerShapeIndex]
            shp_in = selectedLayer.dataProvider().dataSourceUri()

            # Llena el campo de "Carpeta de salida"
            # Le pone la misma ruta donde se ubica el archivo vecrtorial de entrada
            (directoryName, rasterFileName) = os.path.split(shp_in)
            self.dlg.textCarpetaSalida.setPlainText(directoryName)

    def btnProcess_onClick(self):
        print("Iniciando...")

        #
        # Validaciones iniciales
```

```
        #
        #
        # Valida que haya al menos dos capas cargadas en el mapa
        if len(QgsProject.instance().layerTreeRoot().findLayers()) < 2:
            self.iface.messageBar().pushMessage("Error", "No se han cargado las capas !", level=Qgis.Critical)
            return 1

        # Valida que el combo de la capa raster tenga una capa seleccionada
        if self.dlg.cmbLayersRaster.currentIndex() == 0:
            self.iface.messageBar().pushMessage("Error", "No ha seleccionado la capa raster !", level=Qgis.Critical)
            return 1

        # Valida el rango del umbral
        umbral = self.dlg.textUmbral.toPlainText()
        if int(umbral) < 0 or int(umbral) > 300:
            self.iface.messageBar().pushMessage("Error", "El valor del umbral debe ser de 0 a 300 !", level=Qgis.Critical)
            return 1

        # Valida que el combo de la capa vectorial tenga una capa seleccionada
        if self.dlg.cmbLayersShape.currentIndex() == 0:
            self.iface.messageBar().pushMessage("Error", "No ha seleccionado la capa vectorial !", level=Qgis.Critical)
            return 1

        # Valida que haya especificado la ruta de salida
        ruta_out = self.dlg.textCarpetaSalida.toPlainText()
        if not ruta_out:
            self.iface.messageBar().pushMessage("Error", "No ha indicado la carpeta de salida !", level=Qgis.Critical)
            return 1

        # Si no existe la carpeta de salida, Valida que se pueda crear
        if not os.path.isdir(ruta_out):
            try:
                os.mkdir(ruta_out)
            except:
                self.iface.messageBar().pushMessage("Error", "No fue posible crear la carpeta de salida !", level=Qgis.Critical)
                return 1

        # Valida que se pueda crear la carpeta temporal
        ruta_tmp = "{}/tmp".format(ruta_out)
        if not os.path.isdir(ruta_tmp):
            try:
                os.mkdir(ruta_tmp)
            except:
                self.iface.messageBar().pushMessage("Error", "No fue posible crear la carpeta temporal !", level=Qgis.Critical)
                return 1

        #
        # Funciones
        #
        def elimina_archivos_temporales(directoryName):
            tmp_files = glob.glob(directoryName + "/tmp__*.*")
            for tmp_file in tmp_files:
                if os.path.isfile(tmp_file):
                    os.remove(tmp_file)

        def run_cmd(cmd):
            osgeo_cmd='cd "C:\\Program Files\\QGIS*\\bin" &&'
            process=subprocess.call(osgeo_cmd+' '+cmd,shell=True)

        def run_py(cmd):
            process=subprocess.call(cmd, shell=True)

        # Elimina archivos temporales pre-existentes en la ruta del raster de entrada
        # En caso de que haya un proceso anterior fallido
```

```
        elimina_archivos_temporales(ruta_tmp)

        # Obtiene lista de layers del mapa
        layers = [tree_layer.layer() for tree_layer in QgsProject.instance().layerTreeRoot().findLayers()]

        # Obtiene ruta/nombre del archivo raster de entrada
        selectLayerRasterIndex = self.dlg.cmbLayersRaster.currentIndex() - 1
        selectedLayer = layers[selectLayerRasterIndex]
        raster_in = selectedLayer.dataProvider().dataSourceUri()

        # Obtiene ruta/nombre del archivo vectorial de entrada
        selectLayerShapeIndex = self.dlg.cmbLayersShape.currentIndex() - 1
        selectedLayer = layers[selectLayerShapeIndex]
        shp_in = selectedLayer.dataProvider().dataSourceUri()

        # Si hay capa raster cargada, continua el proceso
        selectLayerRasterIndex = self.dlg.cmbLayersRaster.currentIndex() - 1
        if selectLayerRasterIndex >= 0:

            self.iface.messageBar().pushMessage("Ok", "Procesando ...", level=Qgis.Info)
            self.iface.messageBar().pushWidget(self.progressMessageBar, Qgis.Info)

            # Obtiene nombre de la carpeta y archivo de entrada de las capas
            (rasterDirectoryName, rasterFileName) = os.path.split(raster_in)
            (shapeDirectoryName, shapeFileName) = os.path.split(shp_in)

            # revisa que la capa raster sea un archivo .tif
            if not ".TIF" in rasterFileName.upper():
                self.iface.messageBar().pushMessage("Error", "El archivo raster debe tener formato TIF !", level=Qgis.Critical)
                return 0

            #
            # 1. Reclasificar
            #
            print('Reclasificando...')
            raster_reclass = "{}/tmp__{}".format(ruta_tmp, rasterFileName.replace(".tif", "_reclass.tif"))
            txtrules = """
                -5000 thru {}  = NULL
                {} thru 5000 = 1 area quemada
                        * = NULL
                """.format(umbral, umbral)
            parameters = {  'input' : raster_in,
                        'output' : raster_reclass,
                        'txtrules' : txtrules,
                        'GRASS_REGION_PARAMETER' : 'None',
                        'GRASS_REGION_CELLSIZE_PARAMETER' : 0
                    }
            processing.run('grass7:r.reclass', parameters)
            self.progress.setValue(10)

            #
            # 2. Convertir raster a shapefile de polígono.
            #
            print('Raster a polígonos...')
            shp_polygonize_tmp = raster_reclass.replace("_reclass.tif", "_reclass_polygonize_tmp.shp")
            cmd = 'gdal_polygonize.bat "{}" "{}" -b 1 -f "ESRI Shapefile" polygons DN'.format(raster_reclass,
shp_polygonize_tmp)
            run_py(cmd)
            self.progress.setValue(20)

            ### fix geometries
            # Se corrigen las geometrias porque el proceso "polygonize" genera archivos con geometrías inválidas
            print('Fixing geometries (polygonized)...')
            shp_polygonize = shp_polygonize_tmp.replace("_polygonize_tmp.shp", "_polygonize.shp")
```

```
        processing.run("native:fixgeometries", { 'INPUT': shp_polygonize_tmp, 'OUTPUT': shp_polygonize })
        self.progress.setValue(30)

        # Proyecta poligonized a "México ITRF2008 / LCC" (EPSG:6372)
        shp_polygonize_itrf2008_lcc = shp_polygonize.replace("_polygonize.shp", "_polygonize_itrf2008_lcc.shp")
        print('Proyecta poligonized a "Mexico ITRF2008 / LCC"...')
        parameters = {  'INPUT' : shp_polygonize,
                        'TARGET_CRS' : QgsCoordinateReferenceSystem('EPSG:6372') ,
                        'OUTPUT' : shp_polygonize_itrf2008_lcc
                    }
        processing.run('qgis:reprojectlayer', parameters)
        self.progress.setValue(40)

        #
        # Se proyecta archivo poligonizado y shape de entrada a "México ITRF2008 / LCC" para hacer
        # la intersección espacial con archivos vectoriales con la misma proyección
        #
        # Se corrigen las geometrias del archivo shape de entrada por precaución, en caso de que tenga geometrías inválidas
        print('Fixing geometries (CPC)...')
        shp_in_tmp = "{}/tmp__{}".format(ruta_tmp, shapeFileName.replace(".shp", "_tmp.shp"))
        processing.run("native:fixgeometries", { 'INPUT': shp_in, 'OUTPUT': shp_in_tmp })
        self.progress.setValue(50)

        # Proyecta vectorial de entrada a "México ITRF2008 / LCC" (EPSG:6372)
        shp_in_itrf2008_lcc = shp_in_tmp.replace("_tmp.shp", "_itrf2008_lcc.shp")
        print('Proyecta poligonized a "Mexico ITRF2008 / LCC" para cálculo de área...')
        parameters = {  'INPUT' : shp_in_tmp,
                        'TARGET_CRS' : QgsCoordinateReferenceSystem('EPSG:6372') ,
                        'OUTPUT' : shp_in_itrf2008_lcc
                    }
        processing.run('qgis:reprojectlayer', parameters)
        self.progress.setValue(60)

        #
        # 3. Seleccionar espacialmente los polígonos que intersectan al conglomerado
        #
        print('Spatial intersection...')
        shp_extractbylocation = shp_polygonize_itrf2008_lcc.replace("_polygonize_itrf2008_lcc.shp",
"_polygonize_itrf2008_lcc_extractbylocation.shp")
        parameters = {  'INPUT' : shp_polygonize_itrf2008_lcc,
                        'INTERSECT' : shp_in_itrf2008_lcc,
                        'OUTPUT' : shp_extractbylocation,
                        'PREDICATE' : [0]
                    }
        processing.run('qgis:extractbylocation', parameters)
        self.progress.setValue(70)

        #
        # 4. Dissolve
        #
        print('Dissolve...')
        shp_dissolve = shp_extractbylocation.replace("_extractbylocation.shp", "_extractbylocation_dissolve.shp")
        processing.run("native:dissolve", {'FIELD' : [], 'INPUT':shp_extractbylocation, 'OUTPUT':shp_dissolve})
        self.progress.setValue(75)

        #
        # 5. Añadir campo AQ_ha y calcular superficie en has
        #
        print('Adding ha field...')
        shp_ha = shp_dissolve.replace("_dissolve.shp", "_dissolve_ha.shp")
        parameters = {
                        'INPUT' : shp_dissolve,
                        'FIELD_NAME' : 'AQ_ha',
                        'FIELD_TYPE' : 1,
```

```
                'FIELD_LENGTH' : 10,
                'FIELD_PRECISION' : 3,
                'FORMULA' : 'value = $geom.area()/10000.0',
                'OUTPUT' : shp_ha
            }
        processing.run('qgis:advancedpythonfieldcalculator', parameters)
        self.progress.setValue(80)

        # Proyecta vectorial de salida a la misma proyección que el archivo de entrada
        print('Proyecta archivo de salida...')
        layer_shp_in = QgsVectorLayer(shp_in, "shape_in")
        epsg = layer_shp_in.crs().authid()
        shp_ha_projected = shp_ha.replace("_ha.shp", "_ha_project.shp")
        parameters = {  'INPUT' : shp_ha,
                'TARGET_CRS' : QgsCoordinateReferenceSystem("{}".format(epsg)) ,
                'OUTPUT' : shp_ha_projected
            }
        processing.run('qgis:reprojectlayer', parameters)
        self.progress.setValue(90)

        #
        # Repara geometrias de archivo de salida
        #
        print('Fixing geometries (Out)...')
        nombre_shp_out = "AQ_{}".format(rasterFileName[0:-4])
        shp_out = "{}/{}.shp".format(ruta_out, nombre_shp_out)
        processing.run("native:fixgeometries", { 'INPUT': shp_ha_projected, 'OUTPUT': shp_out })
        self.progress.setValue(100)

        # Muestra archivo de salida en Mapa
        print('Cargando capa de área quemada al mapa actual...')
        layer = QgsVectorLayer(shp_out, nombre_shp_out)
        QgsProject.instance().addMapLayer(layer)

        self.iface.messageBar().clearWidgets()

        # Muestra fecha/hora de término de la ejecución
        fecha = time.strftime("%d/%m/%Y")
        hora = time.strftime("%H:%M:%S")
        print("\nTerminado: {} {}".format(fecha, hora))
```

Código XML de *mapeado_3_dialog_base.ui* (Interface gráfica):

```
<?xml version="1.0" encoding="UTF-8"?>
<ui version="4.0">
 <author>Jaime Briseño Reyes &lt;jaime.briseno@gmail.com&gt;</author>
 <class>Mapeado3DialogBase</class>
 <widget class="QDialog" name="Mapeado3DialogBase">
  <property name="geometry">
   <rect>
    <x>0</x>
    <y>0</y>
    <width>444</width>
    <height>331</height>
   </rect>
  </property>
  <property name="windowTitle">
   <string>Área Quemada Sentinel</string>
  </property>
```

```
<widget class="QDialogButtonBox" name="button_box">
 <property name="geometry">
  <rect>
   <x>340</x>
   <y>300</y>
   <width>71</width>
   <height>21</height>
  </rect>
 </property>
 <property name="sizePolicy">
  <sizepolicy hsizetype="Minimum" vsizetype="Fixed">
   <horstretch>0</horstretch>
   <verstretch>0</verstretch>
  </sizepolicy>
 </property>
 <property name="font">
  <font>
   <pointsize>7</pointsize>
  </font>
 </property>
 <property name="orientation">
  <enum>Qt::Horizontal</enum>
 </property>
 <property name="standardButtons">
  <set>QDialogButtonBox::Close</set>
 </property>
</widget>
<widget class="QLabel" name="label_raster_entrada">
 <property name="geometry">
  <rect>
   <x>10</x>
   <y>9</y>
   <width>411</width>
   <height>16</height>
  </rect>
 </property>
 <property name="font">
  <font>
   <family>Calibri</family>
   <pointsize>10</pointsize>
   <weight>50</weight>
   <bold>false</bold>
  </font>
 </property>
 <property name="text">
  <string>Raster de área quemada</string>
 </property>
</widget>
<widget class="QComboBox" name="cmbLayersRaster">
 <property name="geometry">
  <rect>
   <x>10</x>
   <y>37</y>
   <width>411</width>
   <height>24</height>
  </rect>
 </property>
 <property name="font">
  <font>
   <pointsize>8</pointsize>
  </font>
 </property>
</widget>
<widget class="QPushButton" name="btnProcess">
```

```
  <property name="geometry">
   <rect>
    <x>250</x>
    <y>300</y>
    <width>81</width>
    <height>21</height>
   </rect>
  </property>
  <property name="font">
   <font>
    <pointsize>7</pointsize>
   </font>
  </property>
  <property name="text">
   <string>Procesar</string>
  </property>
 </widget>
 <widget class="QLabel" name="label_umbral">
  <property name="geometry">
   <rect>
    <x>10</x>
    <y>70</y>
    <width>101</width>
    <height>16</height>
   </rect>
  </property>
  <property name="font">
   <font>
    <family>Calibri</family>
    <pointsize>10</pointsize>
    <weight>50</weight>
    <bold>false</bold>
   </font>
  </property>
  <property name="text">
   <string>Umbral</string>
  </property>
 </widget>
 <widget class="QTextEdit" name="textUmbral">
  <property name="geometry">
   <rect>
    <x>10</x>
    <y>90</y>
    <width>91</width>
    <height>26</height>
   </rect>
  </property>
  <property name="font">
   <font>
    <pointsize>8</pointsize>
   </font>
  </property>
  <property name="horizontalScrollBarPolicy">
   <enum>Qt::ScrollBarAlwaysOff</enum>
  </property>
  <property name="placeholderText">
   <string>0</string>
  </property>
 </widget>
 <widget class="QComboBox" name="cmbLayersShape">
  <property name="geometry">
   <rect>
    <x>10</x>
    <y>157</y>
```

```
    <width>411</width>
    <height>24</height>
   </rect>
  </property>
  <property name="font">
   <font>
    <pointsize>8</pointsize>
   </font>
  </property>
 </widget>
 <widget class="QLabel" name="label_shape_entrada">
  <property name="geometry">
   <rect>
    <x>10</x>
    <y>129</y>
    <width>411</width>
    <height>16</height>
   </rect>
  </property>
  <property name="font">
   <font>
    <family>Calibri</family>
    <pointsize>10</pointsize>
    <weight>50</weight>
    <bold>false</bold>
   </font>
  </property>
  <property name="text">
   <string>Archivo SHAPE con el polígono del conglomerado</string>
  </property>
 </widget>
 <widget class="QTextEdit" name="textCarpetaSalida">
  <property name="geometry">
   <rect>
    <x>10</x>
    <y>230</y>
    <width>411</width>
    <height>61</height>
   </rect>
  </property>
  <property name="sizePolicy">
   <sizepolicy hsizetype="Expanding" vsizetype="Fixed">
    <horstretch>0</horstretch>
    <verstretch>0</verstretch>
   </sizepolicy>
  </property>
  <property name="font">
   <font>
    <pointsize>8</pointsize>
   </font>
  </property>
  <property name="verticalScrollBarPolicy">
   <enum>Qt::ScrollBarAlwaysOff</enum>
  </property>
  <property name="horizontalScrollBarPolicy">
   <enum>Qt::ScrollBarAsNeeded</enum>
  </property>
  <property name="sizeAdjustPolicy">
   <enum>QAbstractScrollArea::AdjustIgnored</enum>
  </property>
  <property name="placeholderText">
   <string/>
  </property>
 </widget>
```

```
  <widget class="QLabel" name="label_shape_entrada_2">
   <property name="geometry">
    <rect>
     <x>10</x>
     <y>200</y>
     <width>411</width>
     <height>16</height>
    </rect>
   </property>
   <property name="font">
    <font>
     <family>Calibri</family>
     <pointsize>10</pointsize>
     <weight>50</weight>
     <bold>false</bold>
    </font>
   </property>
   <property name="text">
    <string>Carpeta de salida</string>
   </property>
  </widget>
 </widget>
 <resources/>
 <connections>
  <connection>
   <sender>button_box</sender>
   <signal>accepted()</signal>
   <receiver>Mapeado3DialogBase</receiver>
   <slot>accept()</slot>
   <hints>
    <hint type="sourcelabel">
     <x>20</x>
     <y>20</y>
    </hint>
    <hint type="destinationlabel">
     <x>20</x>
     <y>20</y>
    </hint>
   </hints>
  </connection>
  <connection>
   <sender>button_box</sender>
   <signal>rejected()</signal>
   <receiver>Mapeado3DialogBase</receiver>
   <slot>reject()</slot>
   <hints>
    <hint type="sourcelabel">
     <x>20</x>
     <y>20</y>
    </hint>
    <hint type="destinationlabel">
     <x>20</x>
     <y>20</y>
    </hint>
   </hints>
  </connection>
 </connections>
</ui>
```