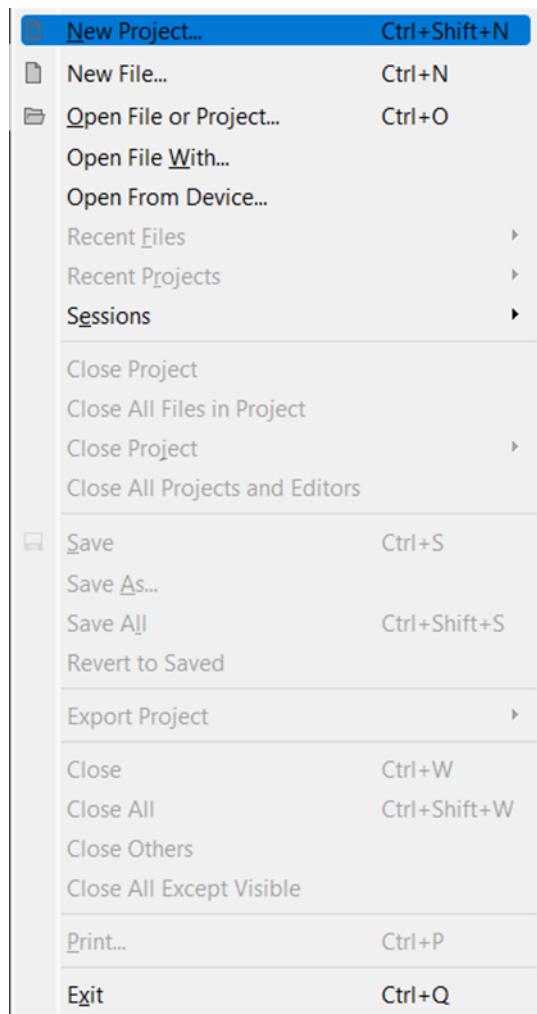


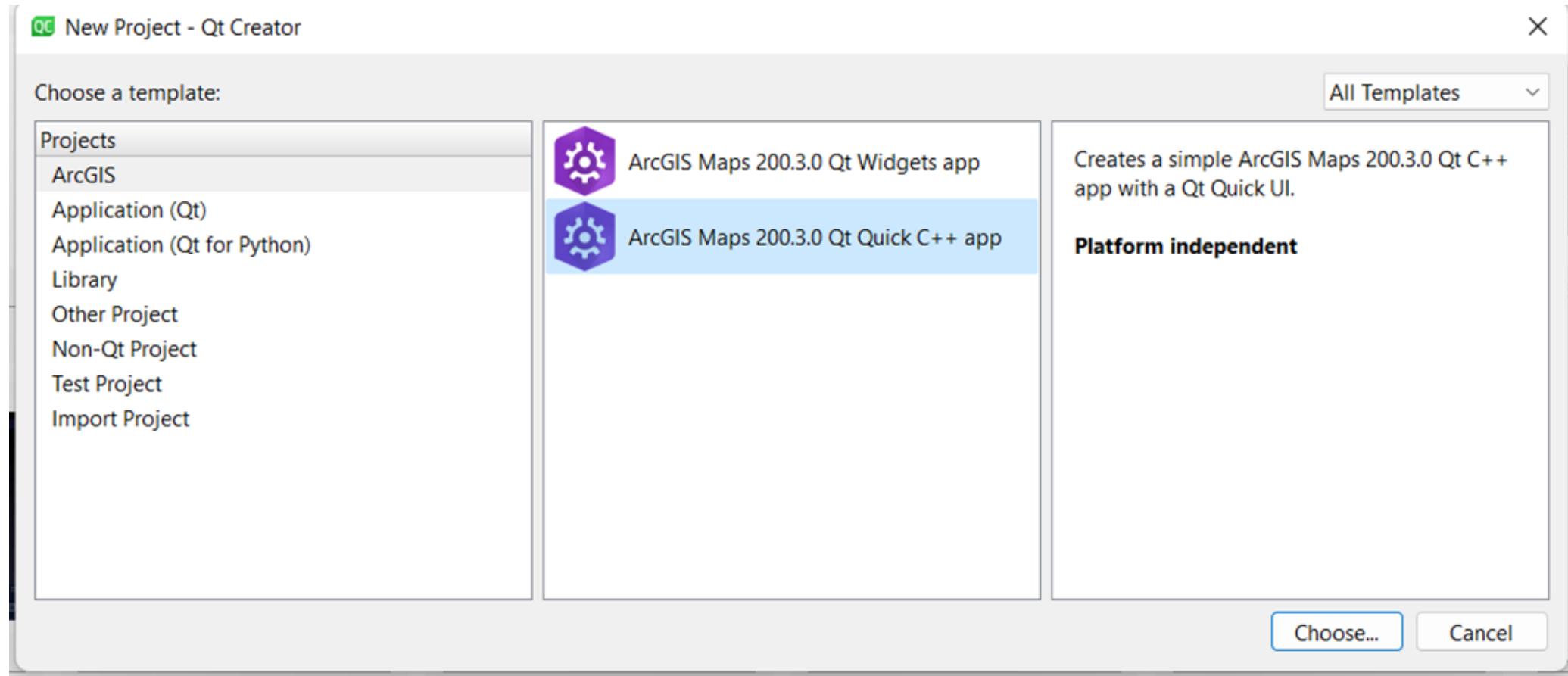
ArcGIS SDK

Visualizar mapa base, proyectar vectores

Proyectar un mapa en ArcGIS SDK



Proyectar un mapa en ArcGIS SDK



Proyectar un mapa en ArcGIS SDK

X

ArcGIS Maps 200.3.0 Qt Quick C++ app

Project Location

Location

Build System Creates a simple ArcGIS Maps 200.3.0 Qt C++ app with a Qt Quick UI.

Details

Kits

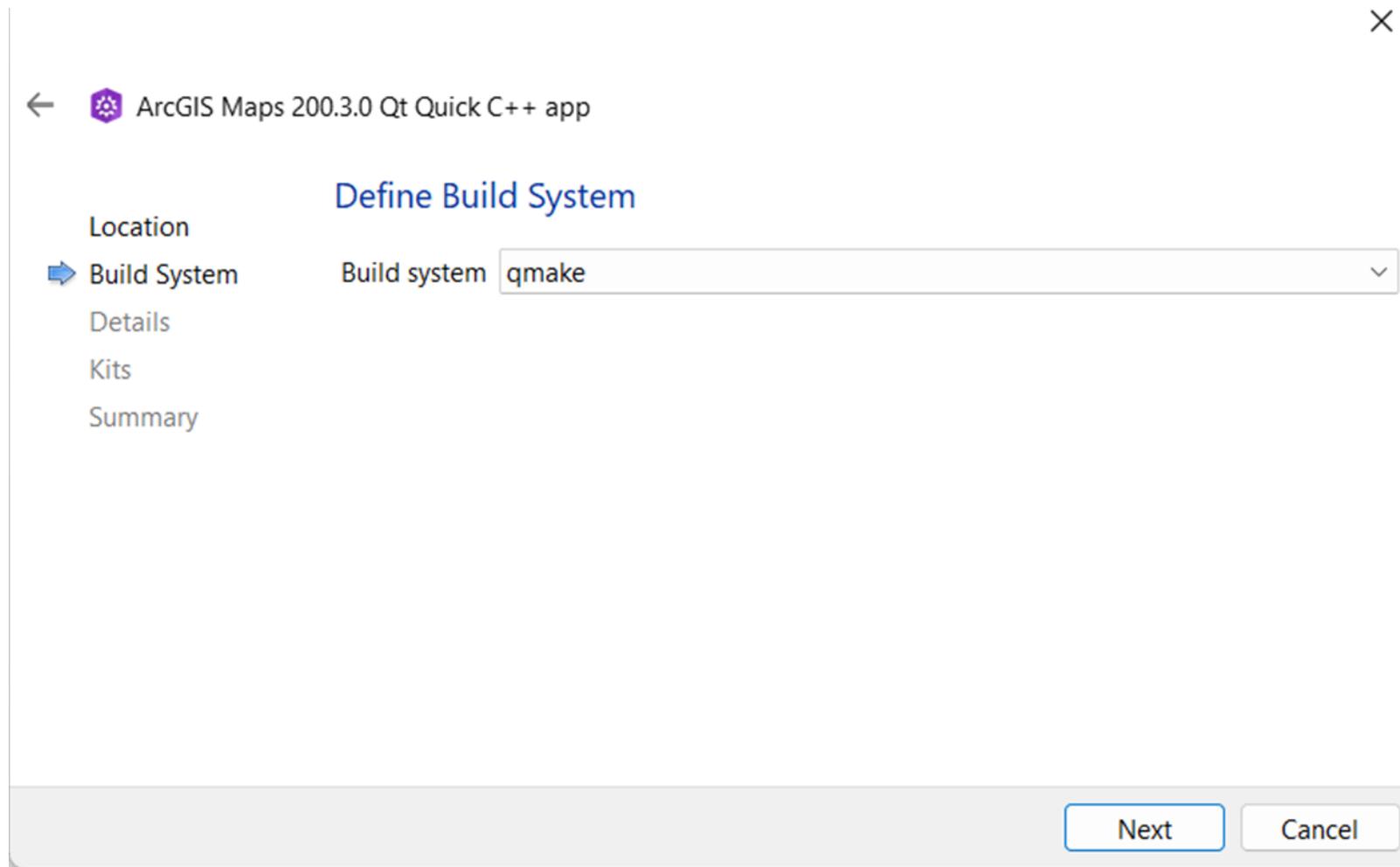
Summary

Name:

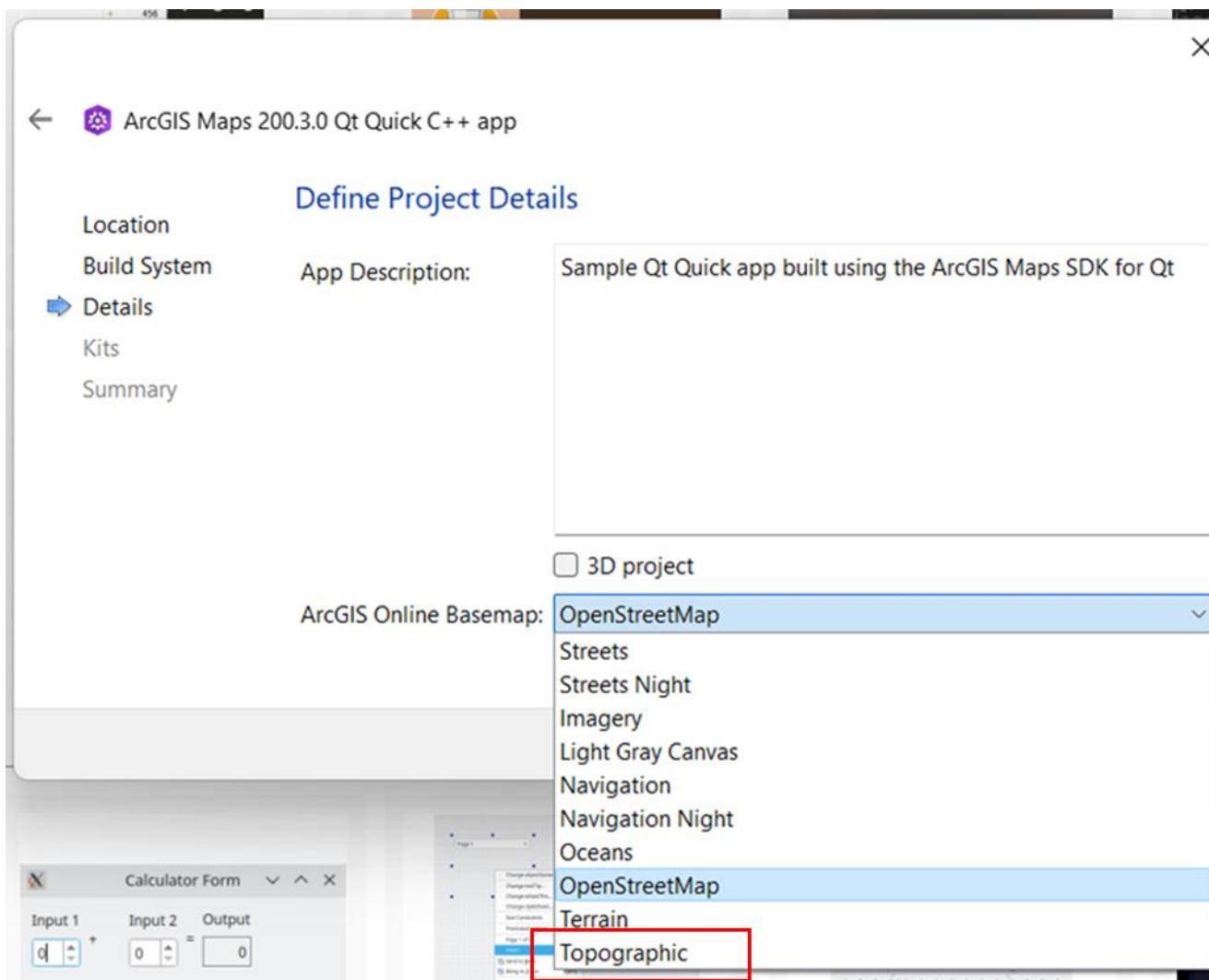
Create in:

Use as default project location

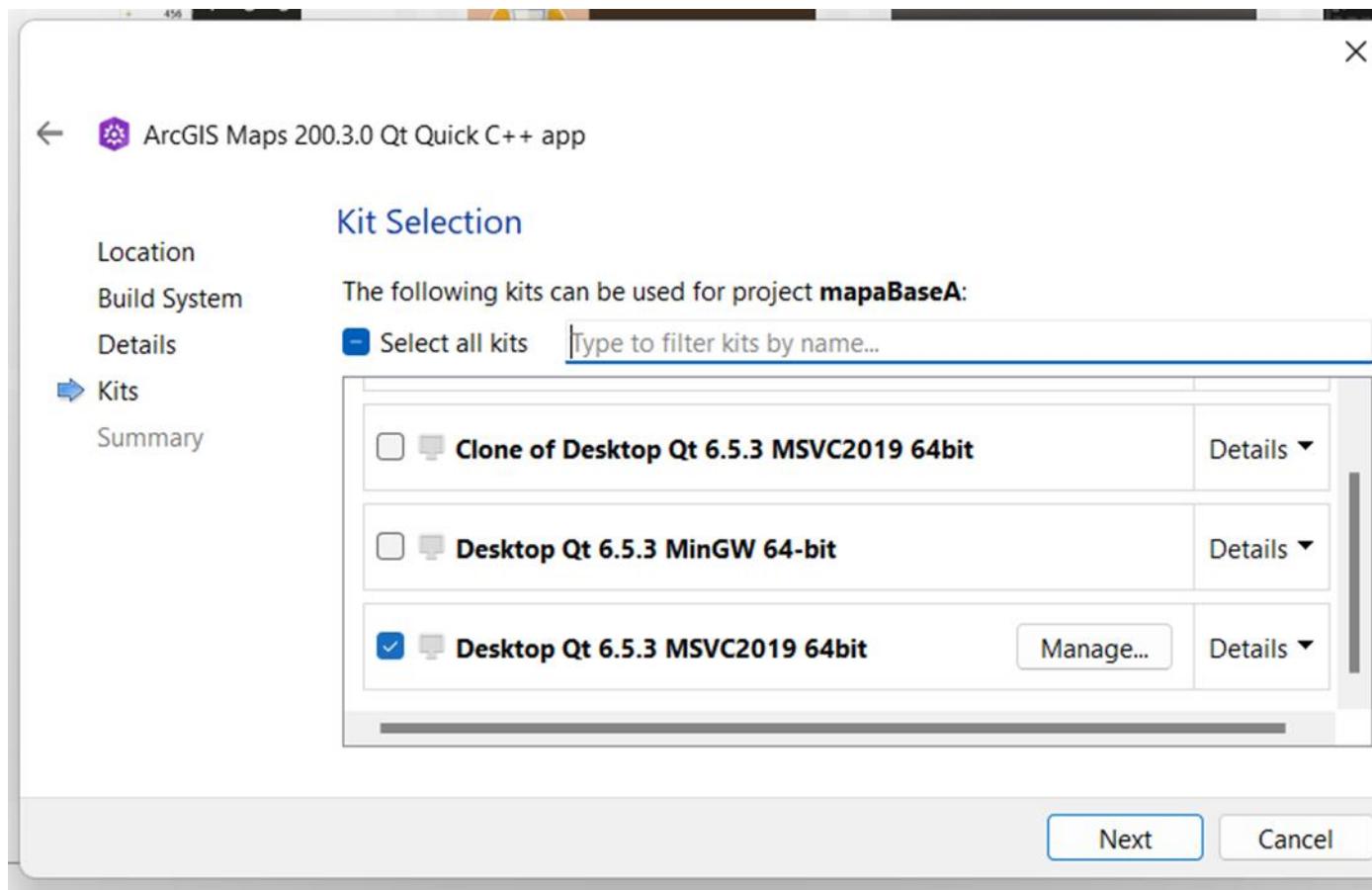
Proyectar un mapa en ArcGIS SDK



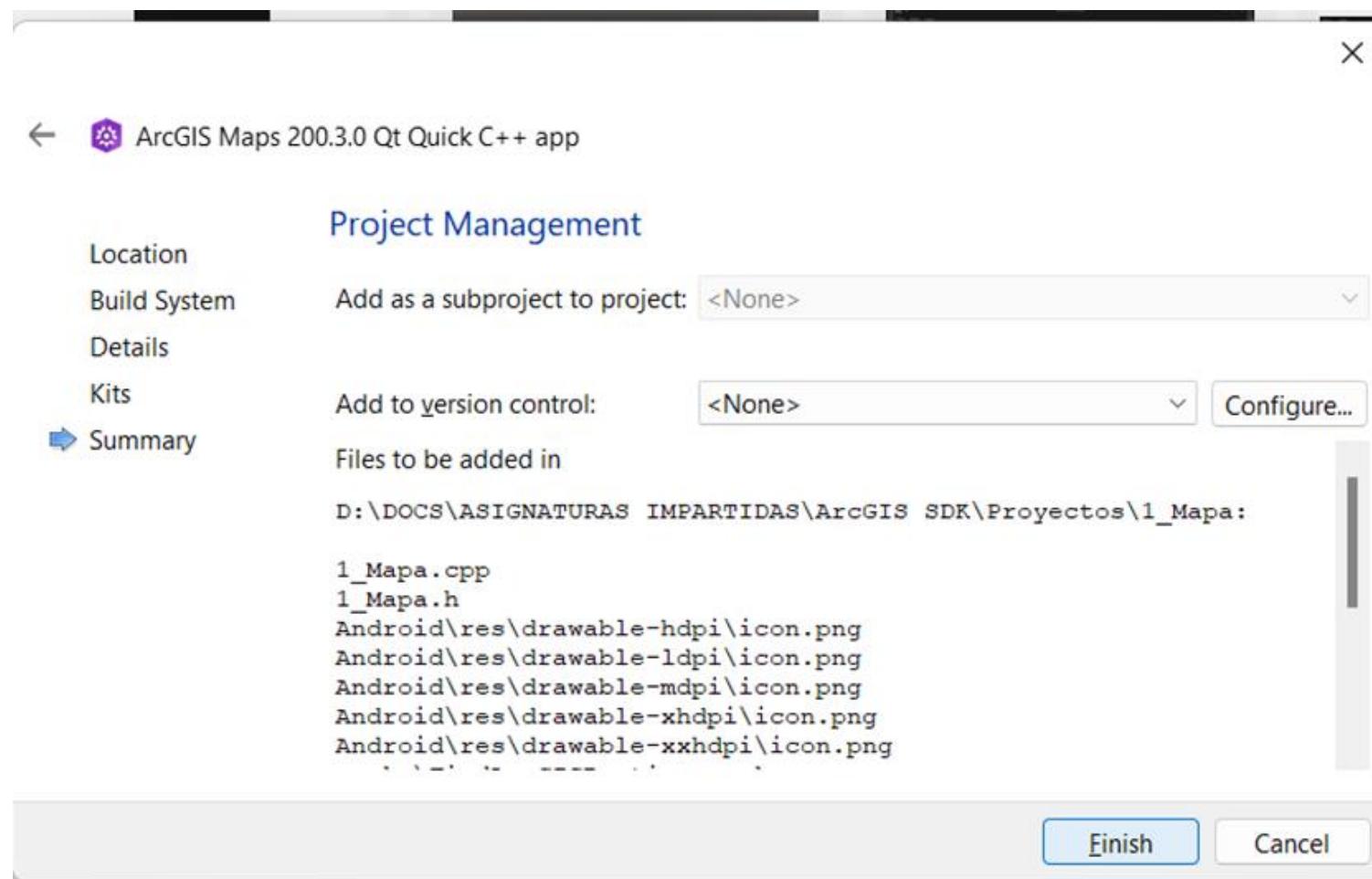
Proyectar un mapa en ArcGIS SDK



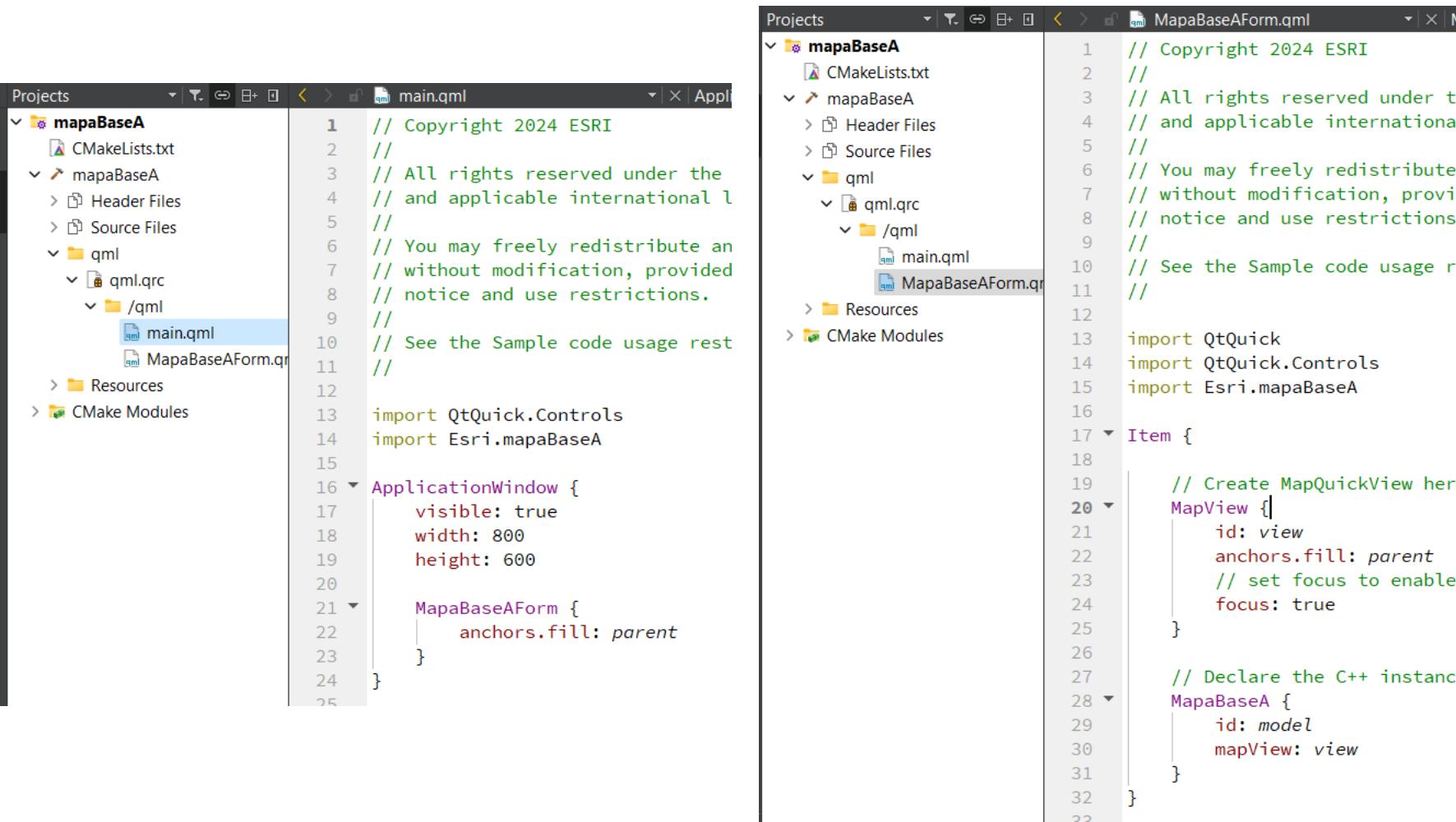
Proyectar un mapa en ArcGIS SDK



Proyectar un mapa en ArcGIS SDK



Proyectar un mapa en ArcGIS SDK – archivos QML



The screenshot shows a Qt-based IDE interface with two main panes. The left pane displays the project structure under 'Projects':

- mapaBaseA (selected)
- CMakeLists.txt
- mapaBaseA (Folder)
 - Header Files
 - Source Files
- qml (Folder)
 - qml.qrc (Selected)
 - /qml (Folder)
 - main.qml
 - MapaBaseAForm.qml
- Resources
- CMake Modules

The right pane shows the content of the selected file, 'main.qml':

```
// Copyright 2024 ESRI
//
// All rights reserved under the
// and applicable international
//
// You may freely redistribute an
// without modification, provided
// notice and use restrictions.
//
// See the Sample code usage rest
//
import QtQuick.Controls
import Esri.mapaBaseA

ApplicationWindow {
    visible: true
    width: 800
    height: 600

    MapaBaseAForm {
        anchors.fill: parent
    }
}
```

Below this, another tab labeled 'MapaBaseAForm.qml' is visible, showing its content:

```
// Copyright 2024 ESRI
//
// All rights reserved under the
// and applicable international
//
// You may freely redistribute an
// without modification, provided
// notice and use restrictions.
//
// See the Sample code usage rest
//
import QtQuick
import QtQuick.Controls
import Esri.mapaBaseA

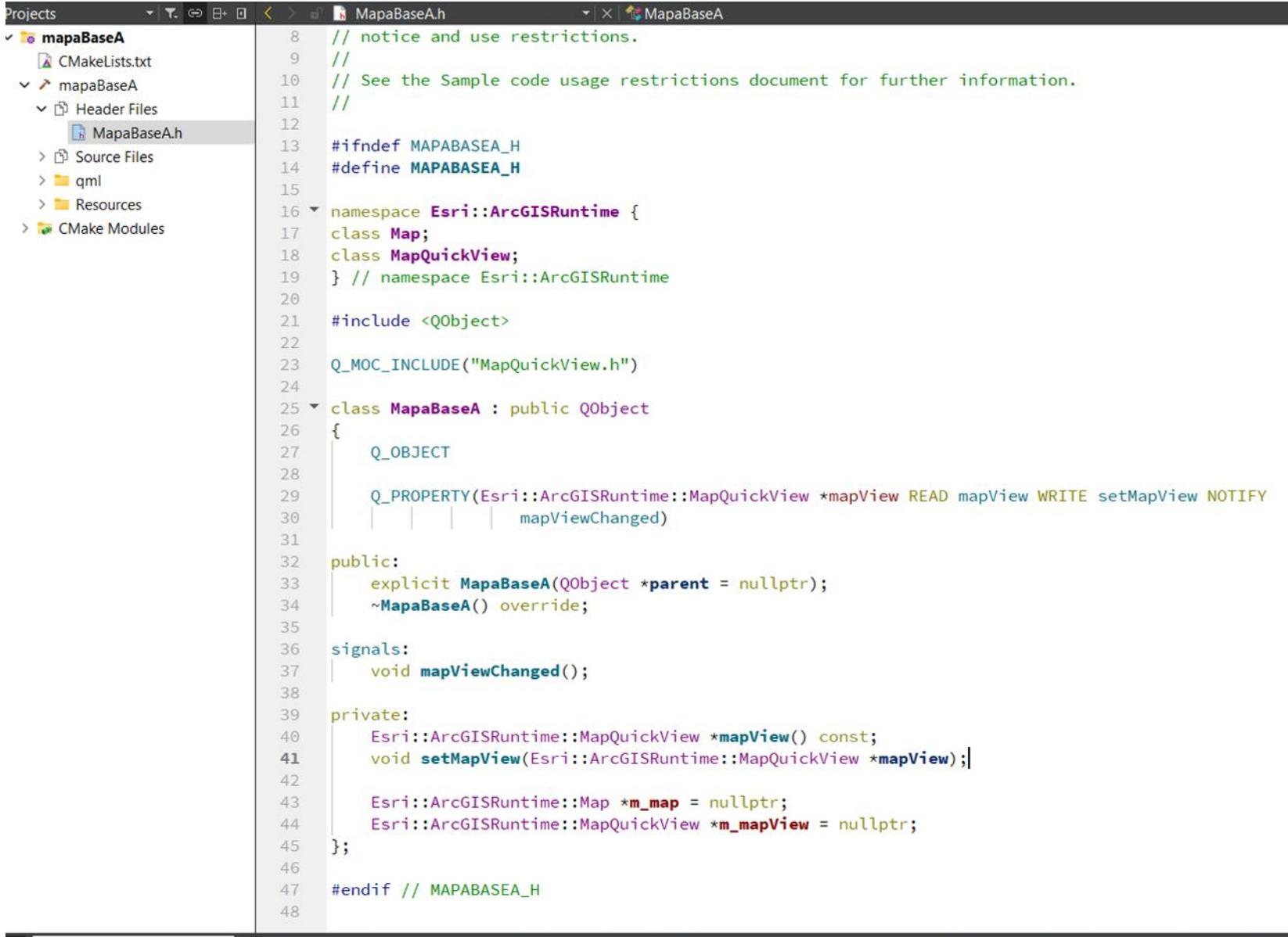
Item {
    // Create MapQuickView here
    MapView {
        id: view
        anchors.fill: parent
        // set focus to enable
        focus: true
    }

    // Declare the C++ instance
    MapaBaseA {
        id: model
        mapView: view
    }
}
```

• ARCHIVOS QML

- los archivos QML (Qt Modelling Language) contiene definiciones de interfaz de usuario, declaraciones de objetos y recursos JavaScript importados.
- Estos archivos se utilizan para crear interfaces gráficas y elementos visuales en aplicaciones desarrolladas con Qt Quick

Proyectar un mapa en ArcGIS SDK – Archivo encabezado .h



The screenshot shows a Qt-based IDE interface with the following details:

- Projects:** A tree view showing a project named "mapaBaseA" with subfolders "CMakeLists.txt", "mapaBaseA", "Header Files" (containing "MapaBaseA.h"), "Source Files", "qml", "Resources", and "CMake Modules".
- Editor:** The main window displays the content of "MapaBaseA.h".
- Code Content:**

```
8 // notice and use restrictions.
9 //
10 // See the Sample code usage restrictions document for further information.
11 //
12 #ifndef MAPABASEA_H
13 #define MAPABASEA_H
14
15 // namespace Esri::ArcGISRuntime {
16 class Map;
17 class MapQuickView;
18 } // namespace Esri::ArcGISRuntime
19
20 #include <QObject>
21
22 Q_MOC_INCLUDE("MapQuickView.h")
23
24 class MapaBaseA : public QObject
25 {
26     Q_OBJECT
27
28     Q_PROPERTY(Esri::ArcGISRuntime::MapQuickView *mapView READ mapView WRITE setMapView NOTIFY
29                 | mapViewChanged)
30
31     public:
32         explicit MapaBaseA(QObject *parent = nullptr);
33         ~MapaBaseA() override;
34
35     signals:
36         void mapViewChanged();
37
38     private:
39         Esri::ArcGISRuntime::MapQuickView *mapView() const;
40         void setMapView(Esri::ArcGISRuntime::MapQuickView *mapView);
41
42         Esri::ArcGISRuntime::Map *m_map = nullptr;
43         Esri::ArcGISRuntime::MapQuickView *m_mapView = nullptr;
44
45 };
46
47 #endif // MAPABASEA_H
```
- Bottom Bar:** Includes tabs for "Type to locate (Ctrl+Shift+F)" and numbered buttons 1 through 9 corresponding to "Issues", "Search Results", "Application Output", "Compile Output", "Terminal", "Version Control", "Test Results", "QML Debugger Console", and "Gene".

- ❖ Los archivos .h permiten dividir el código en módulos más pequeños y manejables.
- ❖ Contienen declaraciones de funciones, clases, variables globales y constantes.
- ❖ Su función principal es exponer la interfaz pública de un módulo o biblioteca.
- ❖ Los archivos .h permiten que otros archivos .cpp accedan a las funciones y clases definidas en ellos.
- ❖ En un archivo .h, se declaran las funciones y clases (es decir, se especifica su firma y parámetros).

Proyectar un mapa en ArcGIS SDK – Archivo encabezado .h

- Las directivas `#ifndef`, `#define`, `#endif` aseguran que los archivos de encabezado (.h) se incluyan solo una vez en un programa, evitando conflictos y redefiniciones.
- `#ifndef` verifica si una macro no está definida y se emplea al comienzo del archivo .h para evitar la inclusión repetida
- `#define` define macros y constantes, en este caso `MAPABASEA_H` es una macro que actúa como guarda inclusión y si no está definida entre `#ifndef` y `#endif` solo se incluirá.
- `#endif` marca el final de la sección protegida, es decir, entre `#ifndef` y `#endif`.

Proyectar un mapa en ArcGIS SDK – Archivo encabezado .h

```
15  
16 namespace Esri::ArcGISRuntime {  
17 class Map;  
18 class MapQuickView;
```

- Esri::ArcGISRuntime es una herramienta para desarrolladores que permite crear aplicaciones de mapeo, ubicación y GIS (Sistemas de Información Geográfica) para escritorio y dispositivos móviles.
- En la sección mostrada en la imagen se declaran dos clases: Map y MapQuickView.
- La clase Map representa un mapa con capas, mapas base y otros contenidos geográficos.
- La clase MapQuickView es una vista para mostrar un mapa.

Proyectar un mapa en ArcGIS SDK – Archivo encabezado .h

- QObject es una clase fundamental en Qt (un marco de trabajo de C++) que proporciona la base para todos los objetos de Qt.
- El moc (Meta-Object compiler) procesa los archivos de encabezado (como MapQuickView.h) para generar código relacionado con el sistema de metaobjetos de Qt. Esto asegura que las clases y objetos definidos en MapQuickView.h estén disponibles para el sistema de metaobjetos de Qt.
- Q_MOC_INCLUDE no es una directiva estándar de C++ y solo es reconocida por el moc de Qt.

Proyectar un mapa en ArcGIS SDK – Archivo encabezado .h

```
class MapaBaseA : public QObject
{
    Q_OBJECT

    Q_PROPERTY(Esri::ArcGISRuntime::MapQuickView *mapView READ mapView WRITE setMapView NOTIFY
               mapViewChanged)

public:
    explicit MapaBaseA(QObject *parent = nullptr);
    ~MapaBaseA() override;

signals:
    void mapViewChanged();

private:
    Esri::ArcGISRuntime::MapQuickView *mapView() const;
    void setMapView(Esri::ArcGISRuntime::MapQuickView *mapView);

    Esri::ArcGISRuntime::Map *m_map = nullptr;
    Esri::ArcGISRuntime::MapQuickView *m_mapView = nullptr;
};
```

- La clase MapaBaseA hereda de QObject, que es una clase fundamental en Qt para crear objetos con características adicionales (como señales y ranuras)
- La macro Q_PROPERTY define una propiedad llamada mapView, la cual representa una instancia de Esri::ArcGISRuntime::MapQuickView. Cada que cambia emite una señal, ya que tiene métodos de lectura (READ) y escritura (WRITE)
- MapaBaseA (en este caso) representa el constructor para inicializar la clase
- ~MapaBaseA (en este caso) representa el destructor cuando se destruye el objeto de esa clase
- mapViewChanged es una señal que se emite cuando cambia la propiedad mapView
- mapView es un método privado que devuelve un puntero a una instancia de Esri::ArcGISRuntime::MapQuickView
- setMapView es un método que establece la instancia de Esri::ArcGISRuntime::MapQuickView para la propiedad mapView
- m_map es un miembro privado que establece un puntero a la instancia de Esri::ArcGISRuntime::Map
- m_mapView es un miembro privado que establece un puntero a la instancia de Esri::ArcGISRuntime::MapQuickView

Proyectar un mapa en ArcGIS SDK – archivo .cpp

The screenshot shows the project structure for 'mapaBaseA' in the left pane, which includes CMakeLists.txt, Header Files, Source Files (containing main.cpp and MapaBaseA.cpp), qml, Resources, and CMake Modules. The right pane displays the code for MapaBaseA.cpp:

```
5 //  
6 // You may freely redistribute and use this sample code, with or  
7 // without modification, provided you include the original copyright  
8 // notice and use restrictions.  
9 //  
10 // See the Sample code usage restrictions document for further information.  
11 //  
12  
13 #include "MapaBaseA.h"  
14  
15 #include "Map.h"  
16 #include "MapQuickView.h"  
17 #include "MapTypes.h"  
18  
19 using namespace Esri::ArcGISRuntime;  
20  
21 MapaBaseA::MapaBaseA(QObject *parent /* = nullptr */)  
22     : QObject(parent)  
23     , m_map(new Map(BasemapStyle::OsmStandard, this))  
24 {}  
25  
26 MapaBaseA::~MapaBaseA() {}  
27  
28 MapQuickView *MapaBaseA::mapView() const  
29 {  
30     return m_mapView;  
31 }  
32  
33 // Set the view (created in QML)  
34 void MapaBaseA::setMapView(MapQuickView *mapView)  
35 {  
36     if (!mapView || mapView == m_mapView) {  
37         return;  
38     }  
39  
40     m_mapView = mapView;  
41     m_mapView->setMap(m_map);  
42  
43     emit mapViewChanged();  
44 }
```

- En el archivo MapaBaseA.cpp implementamos las funciones declaradas en el archivo .h
- Es decir, se implementa el código real de esas funciones y clases mostradas en MapaBaseA.h

Proyectar un mapa en ArcGIS SDK -- main.cpp

- En el archivo main.cpp se emplean las funciones declaradas en el archivo MapaBaseA.h e implementadas en el archivo MapaBaseA.cpp.

```
// without modification, provided you include the original copyright
// notice and use restrictions.
//
// See the Sample code usage restrictions document for further information.
//

#include "MapaBaseA.h"

#include "ArcGISRuntimeEnvironment.h"
#include "MapQuickView.h"

#include <QDir>
#include <QGuiApplication>
#include <QQmlApplicationEngine>

//------------------------------------------------------------------------------

using namespace Esri::ArcGISRuntime;

int main(int argc, char *argv[])
{
    QGuiApplication app(argc, argv);

    // Use of Esri location services, including basemaps and geocoding, requires
    // either an ArcGIS identity or an API key. For more information see
    // https://links.esri.com/arcgis-runtime-security-auth.

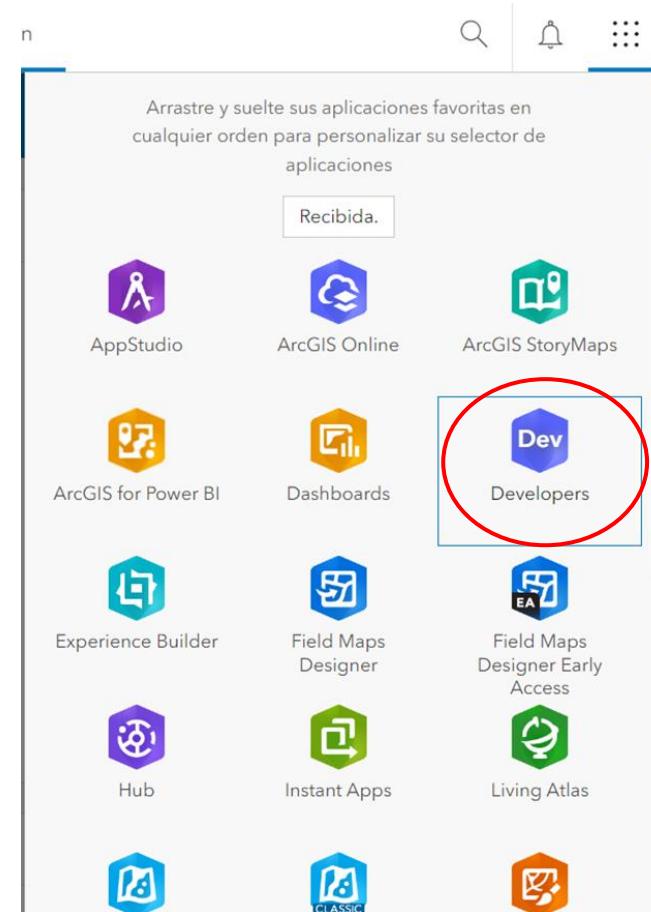
    // 1. ArcGIS identity: An ArcGIS named user account that is a member of an
    // organization in ArcGIS Online or ArcGIS Enterprise.

    // 2. API key: A permanent key that gives your application access to Esri
    // location services. Create a new API key or access existing API keys from
    // your ArcGIS for Developers dashboard (https://links.esri.com/arcgis-api-keys).

    const QString apiKey = QString("");
    if (apiKey.isEmpty()) {
        qWarning() << "Use of Esri location services, including basemaps, requires"
            << "you to authenticate with an ArcGIS identity or set the API Key property.";
    } else {
        ArcGISRuntimeEnvironment::setApiKey(apiKey);
    }
}
```

Proyectar un mapa en ArcGIS SDK – Obtener Api key

- Una vez hemos creado el proyecto en QT creator vamos a iniciar sección en nuestra cuenta de Arcgis online.
- Vamos a seleccionar la opción para desarrolladores



Proyectar un mapa en ArcGIS SDK – Obtener Api key

- Ingresamos a la opción API Keys



Proyectar un mapa en ArcGIS SDK – Obtener Api key

- Para este caso vamos a emplear la API Key que viene por defecto (“Default API Key”) y copiamos el token



Proyectar un mapa en ArcGIS SDK – Obtener Api key

- Copiado el Token vamos al proyecto en QT creator e ingresamos al archivo main.cpp.
- En la opción const QString apikey = QString("PEGAMOS EL TOKEN") pegamos el Token que tomamos de arcgis online.

```
40
41 | const QString apiKey = QString("AAPK1d767ca94cab40c38c6fea6cf92462d0JpfEcsGewRwnT0aXrD1dJrNv0-P0py0KdrB-XAMKHWTt0kxb1TgtbDEHlF7JN2n-");
42 | if (apiKey.isEmpty()) {
43 |     qWarning() << "Use of Esri location services, including basemaps, requires"
44 |             << "you to authenticate with an ArcGIS identity or set the API Key property.";
45 | } else {
46 |     ArcGISRuntimeEnvironment::setApiKey(apiKey);
47 | }
```

Proyectar un mapa en ArcGIS SDK – Configurar librerías y funciones

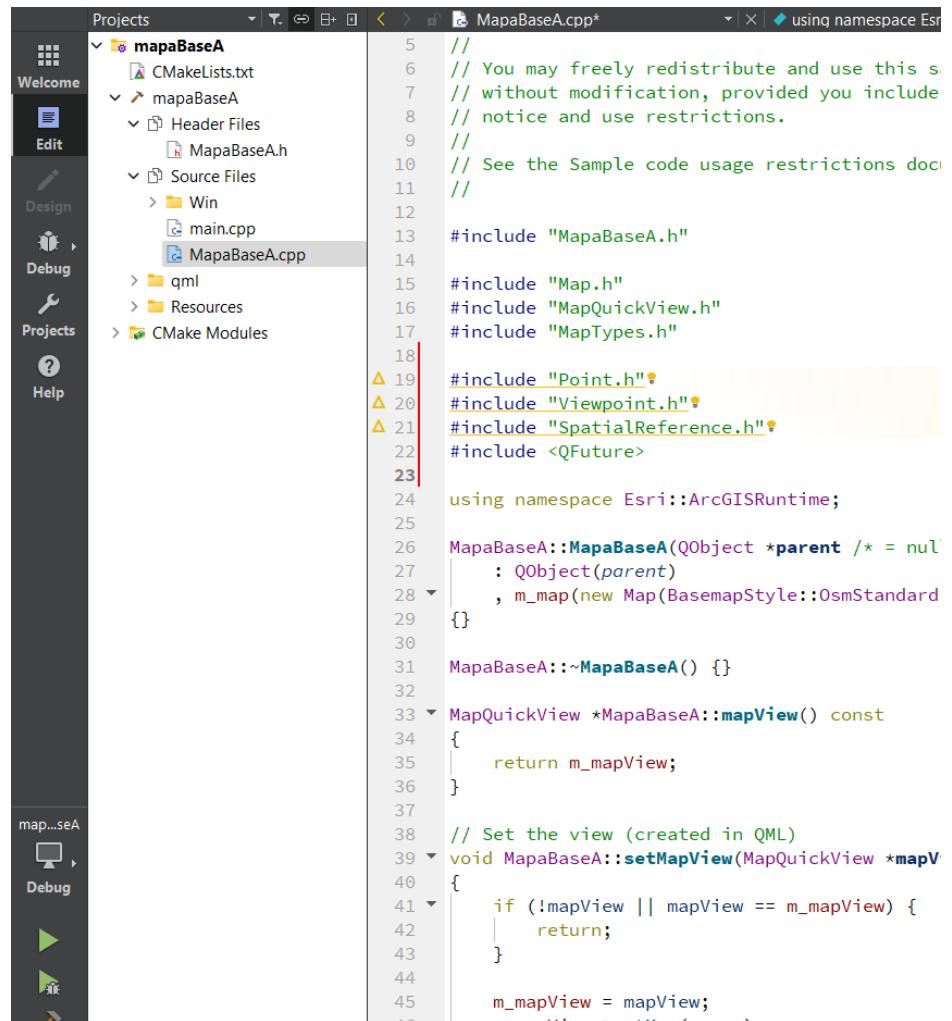
- En el archivo .h vamos a declarar una función privada (**void setupViewpoint();**) para poder visualizar el mapa base.

```
38
39 private:
40     Esri::ArcGISRuntime::MapQuickView *mapView() const;
41     void setMapView(Esri::ArcGISRuntime::MapQuickView *mapView);
42
43     void setupViewpoint();
44
45     Esri::ArcGISRuntime::Map *m_map = nullptr;
46     Esri::ArcGISRuntime::MapQuickView *m_mapView = nullptr;
47 };
48
```

Proyectar un mapa en ArcGIS SDK – Configurar librerías y funciones

- Luego en archivo .cpp (MapaBaseA.cpp) vamos a incluir nuevas librerías para la visualización de nuestro mapa:

- `#include "Point.h"`
- `#include "Viewpoint.h"`
- `#include "SpatialReference.h"`
- `#include <QFuture>`



The screenshot shows a Qt-based IDE interface. On the left, the 'Projects' panel displays a project named 'mapaBaseA' with subfolders 'CMakeLists.txt', 'mapaBaseA' (containing 'Header Files' and 'Source Files'), and 'CMake Modules'. The 'Source Files' folder contains files 'main.cpp' and 'MapaBaseA.cpp', with 'MapaBaseA.cpp' currently selected. The main window shows the code for 'MapaBaseA.cpp'. A red vertical bar highlights the code area from line 19 to line 23, which includes the four header file includes listed in the bullet points above. The code also includes a 'using namespace Esri::ArcGISRuntime;' directive and several member function definitions.

```
5 //  
6 // You may freely redistribute and use this s.  
7 // without modification, provided you include  
8 // notice and use restrictions.  
9 //  
10 // See the Sample code usage restrictions doc  
11 //  
12 #include "MapaBaseA.h"  
13  
14 #include "Map.h"  
15 #include "MapQuickView.h"  
16 #include "MapTypes.h"  
17  
18 #include "Point.h"  
19 #include "Viewpoint.h"  
20 #include "SpatialReference.h"  
21 #include <QFuture>  
22  
23 using namespace Esri::ArcGISRuntime;  
24  
25 MapaBaseA::MapaBaseA(QObject *parent /* = null */  
26 : QObject(parent)  
27 , m_map(new Map(BasemapStyle::OsmStandard  
28 )  
29 )  
30  
31 MapaBaseA::~MapaBaseA() {}  
32  
33 MapQuickView *MapaBaseA::mapView() const  
34 {  
35     return mMapView;  
36 }  
37  
38 // Set the view (created in QML)  
39 void MapaBaseA::setMapView(MapQuickView *mapV  
40 {  
41     if (!mapView || mapView == mMapView) {  
42         return;  
43     }  
44  
45     mMapView = mapView;  
46 }
```

Proyectar un mapa en ArcGIS SDK – Configurar librerías y funciones

- Implementamos la función privada que declaramos en el archivo .h en el archivo MapaBaseA.cpp:

```
void MapaBaseA::setupViewpoint() { const Point  
center(-74.0602115, 4.6607526,  
SpatialReference::wgs84()); const Viewpoint  
viewpoint(center, 100000.0); m_mapView-  
>setViewpointAsync(viewpoint); }
```

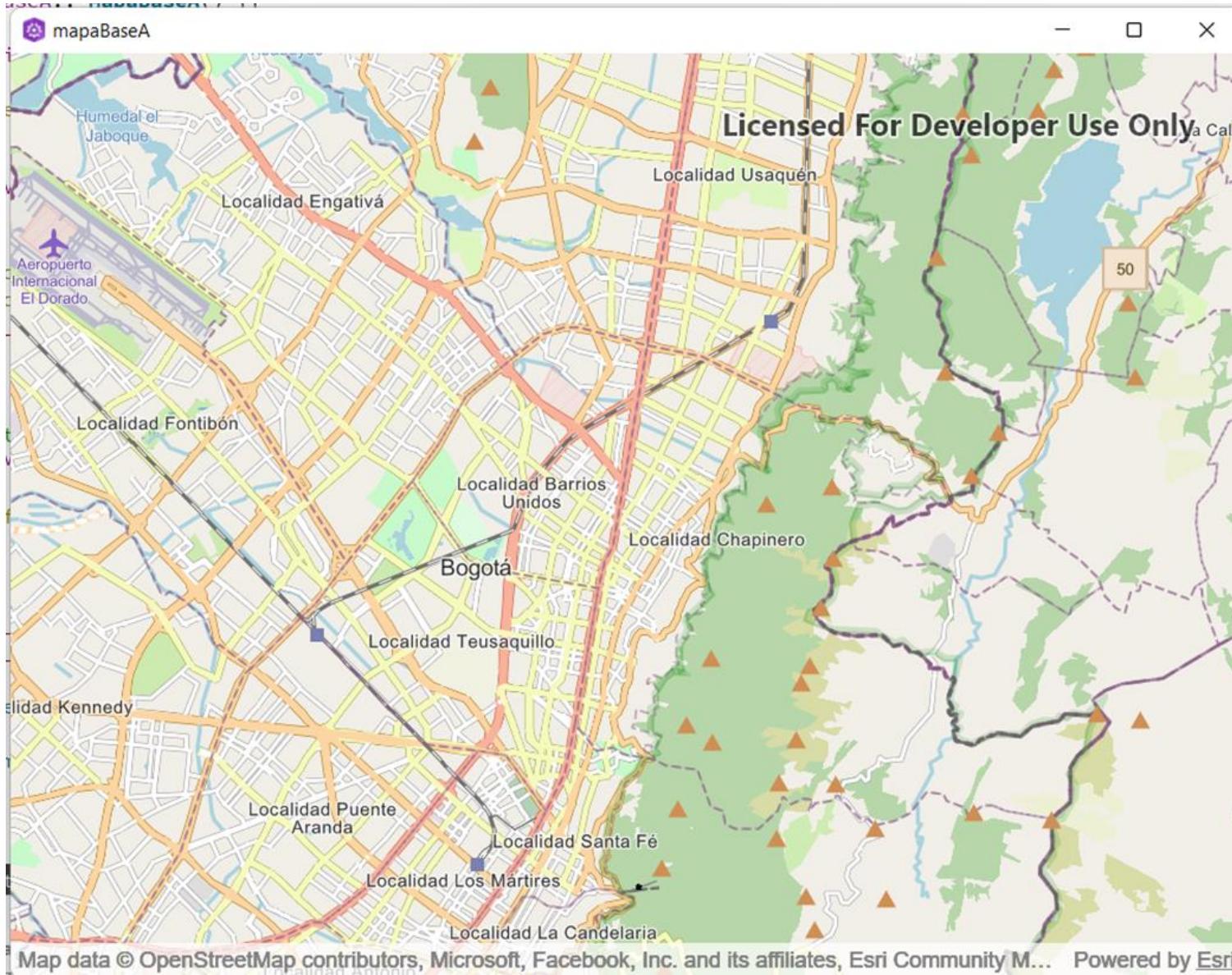
```
33 ▾ MapQuickView *MapaBaseA::mapView() const  
34 {  
35     return m_mapView;  
36 }  
37  
38 ▾ void MapaBaseA::setupViewpoint()  
39 {  
40  
41     const Point center(-118.80543, 34.02700, SpatialReference::wgs84());  
42     const Viewpoint viewpoint(center, 100000.0);  
43     m_mapView->setViewpointAsync(viewpoint);  
44 }  
45  
46 // Set the view (created in QML)  
47  
48 ▾ void MapaBaseA::setMapView(MapQuickView *mapView)  
49 {  
50     if (!mapView || mapView == m_mapView) {  
51         return;  
52     }  
53  
54     m_mapView = mapView;  
55     m_mapView->setMap(m_map);  
56 }
```

Proyectar un mapa en ArcGIS SDK – Configurar librerías y funciones

- Por último, llamamos el archivo setupViewpoint(); declarado en el archivo .h y lo pegamos en el archivo MapaBaseA.cpp. **Corremos.**

```
46
47 // Set the view (created in QML)
48 void MapaBaseA::setMapView(MapQuickView *mapView)
49 {
50     if (!mapView || mapView == m_mapView) {
51         return;
52     }
53
54     m_mapView = mapView;
55     m_mapView->setMap(m_map);
56
57     setupViewpoint();
58
59     emit mapViewChanged();
60 }
61
```

Proyectar un mapa en ArcGIS SDK



Proyectar un mapa en ArcGIS SDK – agregar vectores

- Para crear y visualizar vectores de tipo punto, línea y polígono debemos emplear la clase **class GraphicsOverlay**; de ArcGISRuntime localizada en el archivo .h

```
12  
13 #ifndef MAPABASEA_H  
14 #define MAPABASEA_H  
15  
16 namespace Esri::ArcGISRuntime {  
17     class Map;  
18     class MapQuickView;  
19     class GraphicsOverlay;  
20 } // namespace Esri::ArcGISRuntime  
21
```

Proyectar un mapa en ArcGIS SDK – agregar vectores

- Luego declaramos la función `void createGraphics(Esri::ArcGISRuntime::GraphicsOverlay* overlay);` en la parte privada de nuestra clase localizada en el archivo .h

```
39
40 private:
41     Esri::ArcGISRuntime::MapQuickView *mapView() const;
42     void setMapView(Esri::ArcGISRuntime::MapQuickView *mapView);
43
44     void setupViewpoint();
45
46     void createGraphics(Esri::ArcGISRuntime::GraphicsOverlay* overlay);|
47
48     Esri::ArcGISRuntime::Map *m_map = nullptr;
49     Esri::ArcGISRuntime::MapQuickView *m_mapView = nullptr;
50 };
51
52 #endif // MAPABASEA_H
53
```

Proyectar un mapa en ArcGIS SDK – agregar vectores

- En nuestro archivo MapaBaseA.cpp agregamos las siguientes librerías:

```
#include "Graphic.h"
#include "GraphicListModel.h"
#include "GraphicsOverlay.h"
#include "GraphicsOverlayListModel.h"
#include "PolylineBuilder.h"
#include "PolygonBuilder.h"
#include "SimpleFillSymbol.h"
#include "SimpleLineSymbol.h"
#include "SimpleMarkerSymbol.h"
#include "SymbolTypes.h"
```

Proyectar un mapa en ArcGIS SDK – agregar vectores

The screenshot shows a Qt-based IDE interface with the following details:

- Left Sidebar:** Projects, Welcome, Edit, Design, Debug, Projects, Help.
- Central Area:** A code editor window titled "MapaBaseA.cpp*" showing C++ code for a class "MapaBaseA". The code includes numerous header file includes for ArcGIS Runtime components like "Graphic.h", "GraphicListModel.h", etc. Lines 24 through 33 are highlighted in yellow.
- Bottom Area:** Application Output window showing logs of QML debugging and starting the application.

```
//  
//  
#include "MapaBaseA.h"  
  
#include "Map.h"  
#include "MapQuickView.h"  
#include "MapTypes.h"  
  
#include "Point.h"  
#include "Viewpoint.h"  
#include "SpatialReference.h"  
#include <QFuture>  
  
#include "Graphic.h"  
#include "GraphicListModel.h"  
#include "GraphicsOverlay.h"  
#include "GraphicsOverlayListModel.h"  
#include "PolylineBuilder.h"  
#include "PolygonBuilder.h"  
#include "SimpleFillSymbol.h"  
#include "SimpleLineSymbol.h"  
#include "SimpleMarkerSymbol.h"  
#include "SymbolTypes.h"  
  
using namespace Esri::ArcGISRuntime;  
  
MapaBaseA::MapaBaseA(QObject *parent /* = nullptr */)  
    : QObject(parent)  
    , m_map(new Map(BasemapStyle::OsmStandard, this))  
{}  
  
MapaBaseA::~MapaBaseA() {}
```

Application Output:
mapaBaseA
QML debugging is enabled. Only use this in a safe environment.
10:54:54: D:\DOCS\ASIGNATURAS IMPARTIDAS\ArcGIS SDK\Proyectos\b

10:55:09: Starting D:\DOCS\ASIGNATURAS IMPARTIDAS\ArcGIS SDK\Pr
QML debugging is enabled. Only use this in a safe environment.
10:56:04: D:\DOCS\ASIGNATURAS IMPARTIDAS\ArcGIS SDK\Proyectos\b

Proyectar un mapa en ArcGIS SDK – agregar vectores

- Como miembro de la función Display_a_map::setMapView definida en el archivo MapaBaseA.cpp vamos a incluir el método:

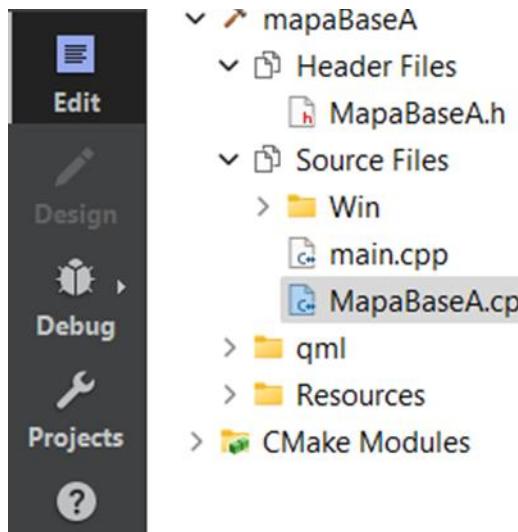
```
GraphicsOverlay* overlay = new  
GraphicsOverlay(this); createGraphics(overlay);  
m_mapView->graphicsOverlays()->append(overlay);
```

```
51  
52 // Set the view (created in QML)  
53 void MapaBaseA::setMapView(MapQuickView *mapView)  
54 {  
55     if (!mapView || mapView == m_mapView) {  
56         return;  
57     }  
58  
59     m_mapView = mapView;  
60     m_mapView->setMap(m_map);  
61  
62     setupViewpoint();  
63  
64     GraphicsOverlay* overlay = new GraphicsOverlay(this);  
65     createGraphics(overlay);  
66     m_mapView->graphicsOverlays()->append(overlay);  
67  
68     emit mapViewChanged();  
69  
70 }
```

Proyectar un mapa en ArcGIS SDK – agregar vectores

- Vamos a crear un nuevo método llamado **Display_a_map::createGraphics** justo después del método **Display_a_map::setupViewpoint** en el archivo **MapaBaseA.cpp**

```
void MapaBaseA::createGraphics(GraphicsOverlay *overlay) { }
```



The code editor displays the **MapaBaseA.cpp** file. It contains two methods: **setupViewpoint()** and the newly added **createGraphics()**.

```
void MapaBaseA::setupViewpoint()
{
    const Point center(-74.0602115, 4.6607526, SpatialReference::wgs84());
    const Viewpoint viewpoint(center, 10000.0);
    m_mapView->setViewpointAsync(viewpoint);
}

void MapaBaseA::createGraphics(GraphicsOverlay *overlay) { }
```

Proyectar un mapa en ArcGIS SDK – agregar vectores

- **Crear Punto:**
- Dentro del método creado **Display_a_map::createGraphics** vamos a definir nuestro punto en el archivo MapaBaseA.cpp :

```
const Point Universidad(-74.0602115, 4.6607526, SpatialReference::wgs84());
// símbolos del punto
SimpleLineSymbol* point_outline = new SimpleLineSymbol(SimpleLineSymbolStyle::Solid, QColor("blue"),
3, this);
SimpleMarkerSymbol* point_symbol = new SimpleMarkerSymbol(SimpleMarkerSymbolStyle::Circle,
QColor("red"), 10, this);
point_symbol->setOutline(point_outline);

// grafica para proyectar el punto y su simbología
Graphic* point_graphic = new Graphic(Universidad, point_symbol, this);
// Agregar punto
overlay->graphics()->append(point_graphic);
```

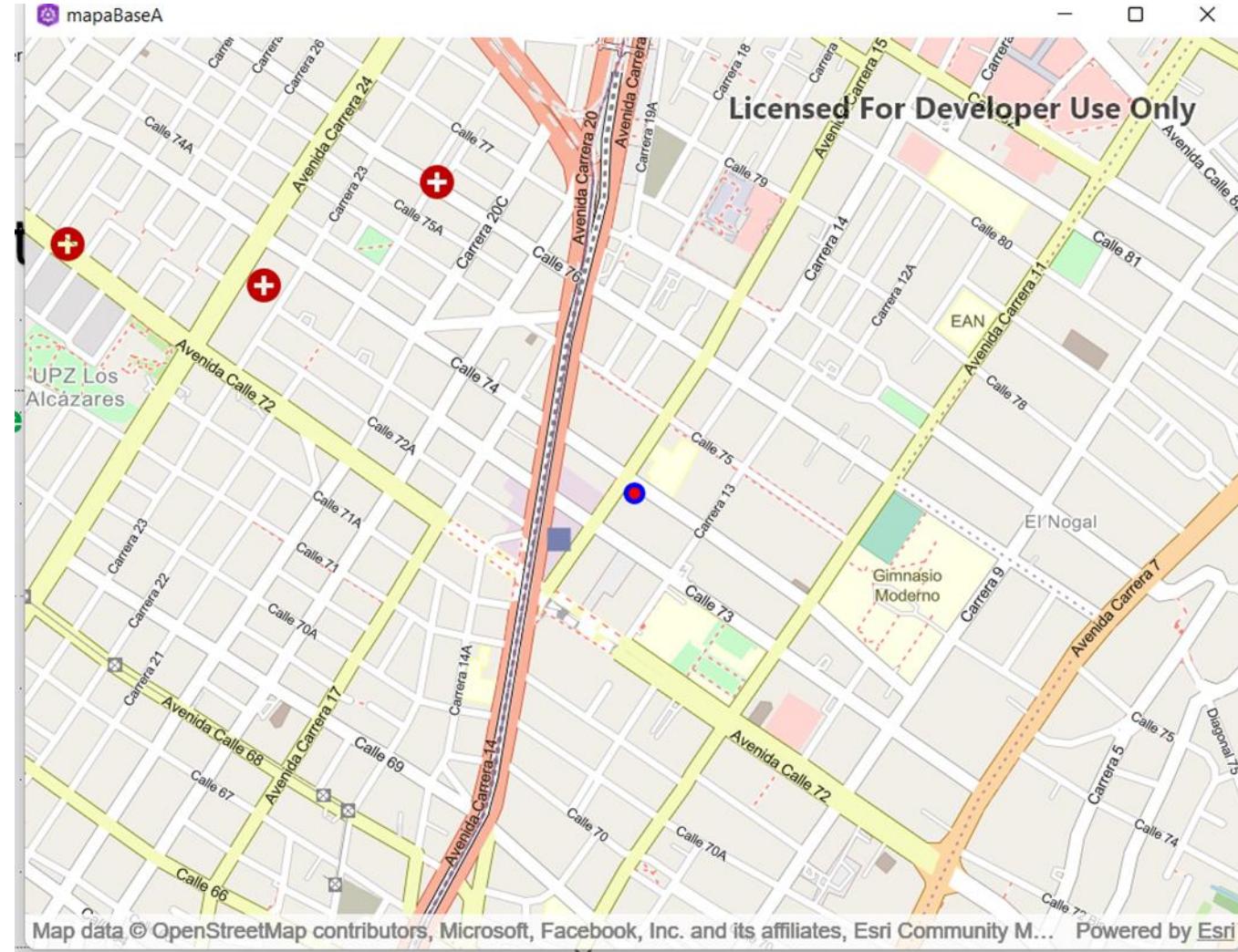
Proyectar un mapa en ArcGIS SDK – agregar vectores

- Crear Punto:

```
57
58 void MapaBaseA::createGraphics(GraphicsOverlay *overlay)
59 {
60     const Point Universidad(-74.0602115, 4.6607526, SpatialReference::wgs84());
61     // símbolos del punto
62     SimpleLineSymbol* point_outline = new SimpleLineSymbol(SimpleLineSymbolStyle::Solid, QColor("blue"), 3, this);
63     SimpleMarkerSymbol* point_symbol = new SimpleMarkerSymbol(SimpleMarkerSymbolStyle::Circle, QColor("red"), 10, this);
64     point_symbol->setOutline(point_outline);
65
66     // grafica para proyectar el punto y su simbología
67     Graphic* point_graphic = new Graphic(Universidad, point_symbol, this);
68     // Agregar punto
69     overlay->graphics()->append(point_graphic);
70 }
71
72
73 // Set the view (created in QML)
74 void MapaBaseA::setMapView(MapQuickView *mapView)
75 {
```

Proyectar un mapa en ArcGIS SDK – agregar vectores

- **Crear Punto:**



Proyectar un mapa en ArcGIS SDK – agregar vectores

- **Crear Línea:**

- Dentro del método creado **Display_a_map::createGraphics** vamos a definir nuestra línea en el archivo MapaBaseA.cpp :

```
// símbolos de línea
PolylineBuilder* PolylineBuilder = new class PolylineBuilder(SpatialReference::wgs84(), this);
PolylineBuilder ->addPoint(-74.0602115, 4.6607526);
PolylineBuilder ->addPoint(-74.0502115, 4.6507526);
PolylineBuilder ->addPoint(-74.0402115, 4.6407526);
// Crear símbolo de la línea
SimpleLineSymbol* line_symbol = new SimpleLineSymbol(SimpleLineSymbolStyle::Solid, QColor(Qt::blue), 3, this);
// grafica para crear la línea
Graphic* PoliLinea = new Graphic(PolylineBuilder->toGeometry(), line_symbol, this);
// Agregar líneas
overlay->graphics()->append(PoliLinea);
```

Proyectar un mapa en ArcGIS SDK – agregar vectores

- Crear Línea:

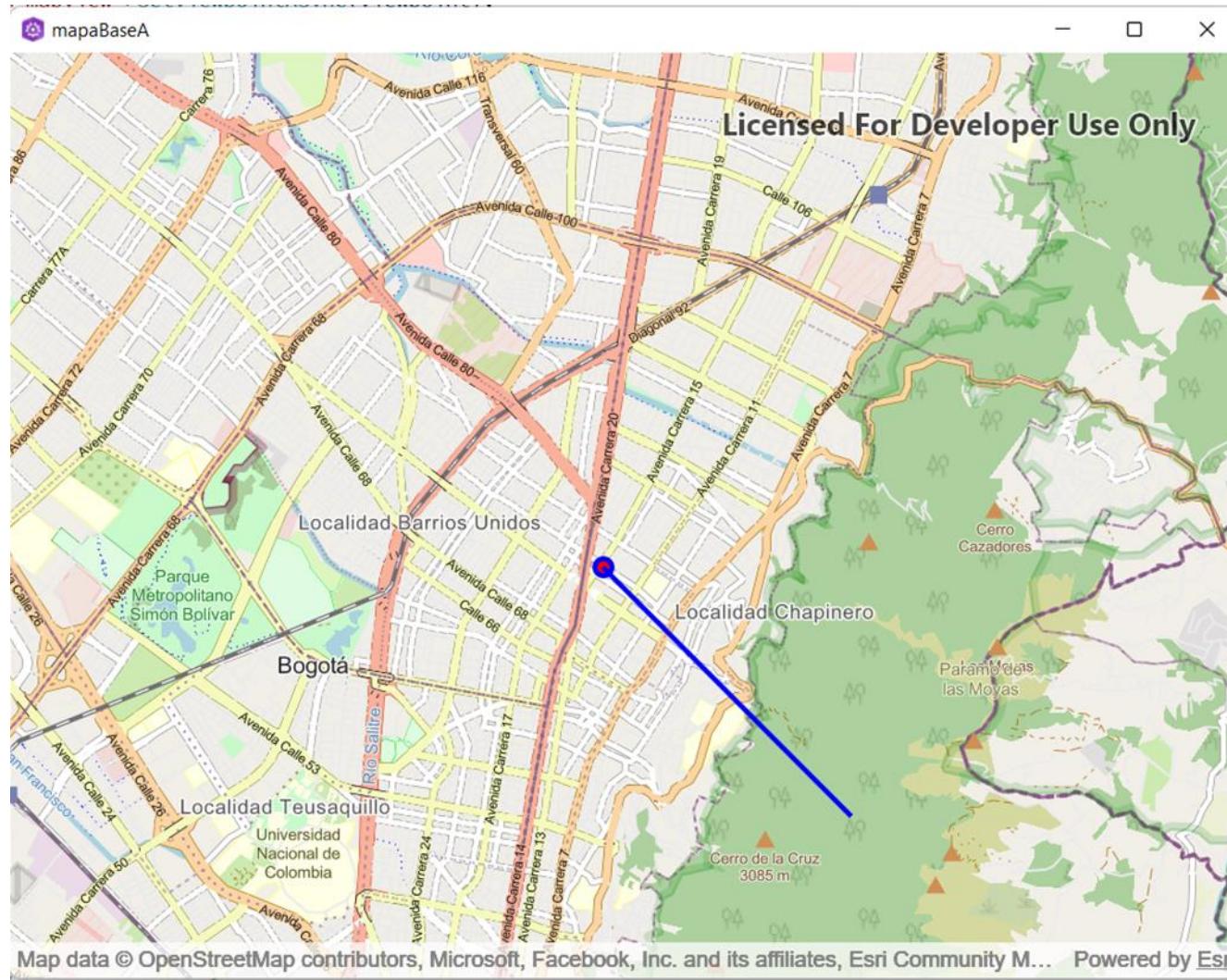
The screenshot shows a Qt-based IDE interface. On the left, the project structure is displayed under 'Source Files': Header Files (MapaBaseA.h), Source Files (Win, main.cpp, MapaBaseA.cpp), qml, Resources, and CMake Modules. The 'MapaBaseA.cpp' file is currently selected. The code editor on the right contains the following C++ code:

```
58 void MapaBaseA::createGraphics(GraphicsOverlay *overlay) {
59
60     const Point Universidad(-74.0602115, 4.6607526, SpatialReference::wgs84());
61     // símbolos del punto
62     SimpleLineSymbol* point_outline = new SimpleLineSymbol(SimpleLineStyle::Solid, QColor("blue"), 3, this);
63     SimpleMarkerSymbol* point_symbol = new SimpleMarkerSymbol(SimpleMarkerSymbolStyle::Circle, QColor("red"), 10, this);
64     point_symbol->setOutline(point_outline);
65     // grafica para proyectar el punto y su simbología
66     Graphic* point_graphic = new Graphic(Universidad, point_symbol, this);
67     // Agregar punto
68     overlay->graphics()->append(point_graphic);
69
70     // símbolos del línea
71     PolylineBuilder* PolylíneBuilder = new class PolylineBuilder(SpatialReference::wgs84(), this);
72     PolylíneBuilder ->addPoint(-74.0602115, 4.6607526);
73     PolylíneBuilder ->addPoint(-74.0502115, 4.6507526);
74     PolylíneBuilder ->addPoint(-74.0402115, 4.6407526);
75     // Crear símbolo de la línea
76     SimpleLineSymbol* line_symbol = new SimpleLineSymbol(SimpleLineStyle::Solid, QColor(Qt::blue), 3, this);
77     // grafica para crear la línea
78     Graphic* Polilínea = new Graphic(PolylineBuilder->toGeometry(), line_symbol, this);
79     // Agregar líneas
80     overlay->graphics()->append(Polilínea);
81
82
83
84 }
```

The code implements a `createGraphics` method that adds a point symbol and a line symbol to a `GraphicsOverlay`. The point symbol is a red circle with a blue outline at coordinates (-74.0602115, 4.6607526). The line symbol is a solid blue line connecting three points at approximately (-74.0602115, 4.6607526), (-74.0502115, 4.6507526), and (-74.0402115, 4.6407526).

Proyectar un mapa en ArcGIS SDK – agregar vectores

- Crear Línea:



Proyectar un mapa en ArcGIS SDK – agregar vectores

- **Crear Polígono:**
- Dentro del método creado `Display_a_map::createGraphics` vamos a definir nuestro polígono en el archivo MapaBaseA.cpp :

Proyectar un mapa en ArcGIS SDK – agregar vectores

- **Crear Polígono:**

```
// Simbolo de polígono
const QList<Point> definirPuntos = {
    Point(-74.0602115, 4.6607526),
    Point(-74.0602115, 4.6407526),
    Point(-74.0402115, 4.6407526),
    Point(-74.0402115, 4.6607526),
};

// Crear poligono empleando los puntos de arriba
PolygonBuilder* constructorDePolig = new PolygonBuilder(SpatialReference::wgs84(), this);
constructorDePolig->addPoints(definirPuntos);

// Crear simbologia del poligono
SimpleLineSymbol* simbologiaLineasPolig = new SimpleLineSymbol(SimpleLineSymbolStyle::Solid, QColor(Qt::blue), 3,
this);

SimpleFillSymbol* llenarSimbolo = new SimpleFillSymbol(SimpleFillSymbolStyle::Solid, QColor(Qt::yellow),
simbologiaLineasPolig, this);

// Visualizar poligono
Graphic* graficarPoligono = new Graphic(constructorDePolig->toGeometry(), llenarSimbolo, this);

// Agregar a la capa
overlay->graphics()->append(graficarPoligono);
```

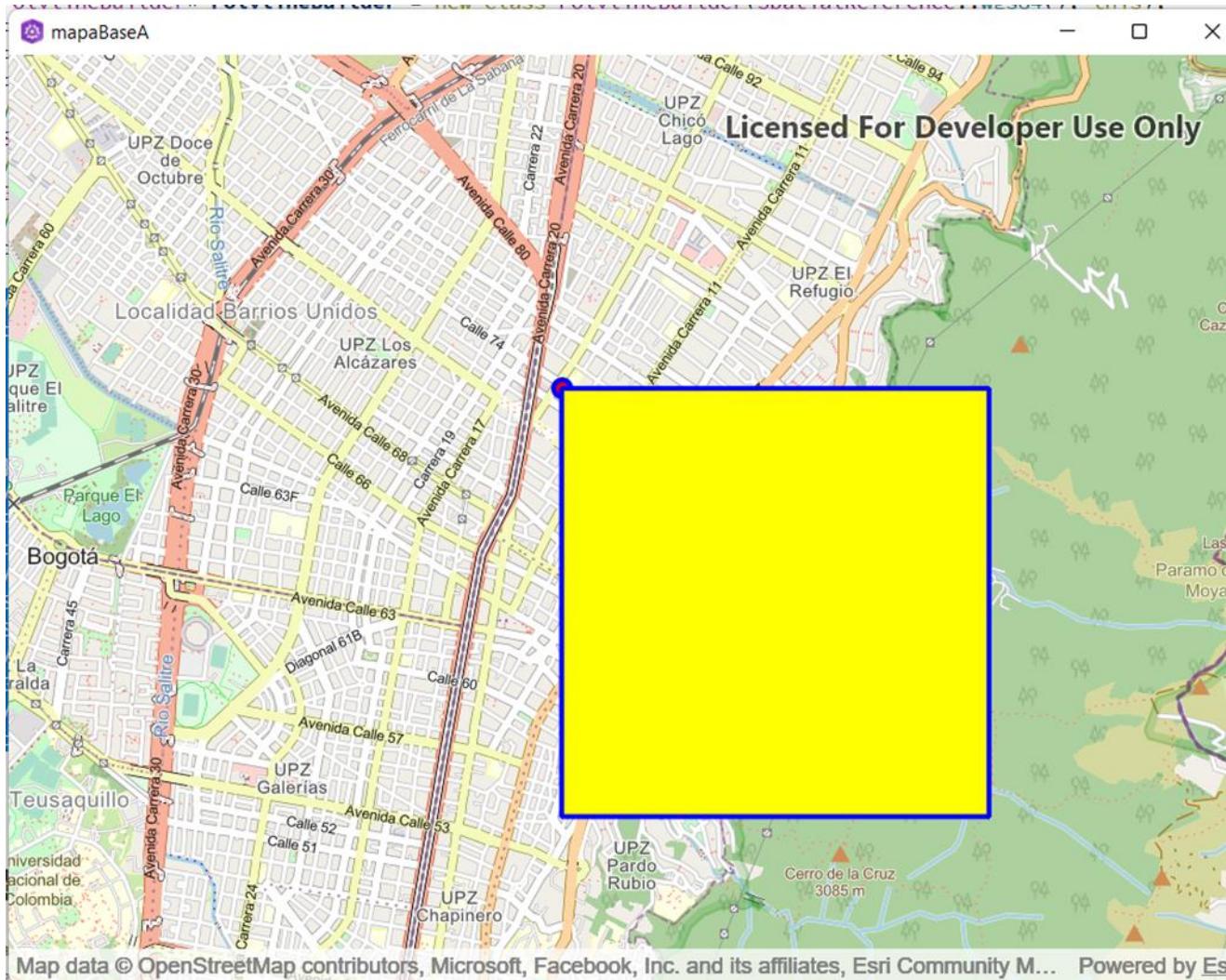
Proyectar un mapa en ArcGIS SDK – agregar vectores

- Crear Polígono:

```
> MapaBaseA.cpp      ▾ X MapaBaseA::createGraphics(GraphicsOverlay *) -> void
68     overlay->graphics()->append(point_graphic);
69
70     // símbolos de línea
71     PolylineBuilder* PolylineBuilder = new class PolylineBuilder(SpatialReference::wgs84(), this);
72     PolylineBuilder ->addPoint(-74.0602115, 4.6607526);
73     PolylineBuilder ->addPoint(-74.0502115, 4.6507526);
74     PolylineBuilder ->addPoint(-74.0402115, 4.6407526);
75     // Crear símbolo de la línea
76     SimpleLineSymbol* line_symbol = new SimpleLineSymbol(SimpleLineSymbolStyle::Solid, QColor(Qt::blue), 3, this);
77     // grafica para crear la línea
78     Graphic* PoliLinea = new Graphic(PolylineBuilder->toGeometry(), line_symbol, this);
79     // Agregar líneas
80     overlay->graphics()->append(PoliLinea);
81
82     // Simbolo de polígono
83     const QList<Point> definirPuntos = {
84         Point(-74.0602115, 4.6607526),
85         Point(-74.0602115, 4.6407526),
86         Point(-74.0402115, 4.6407526),
87         Point(-74.0402115, 4.6607526),
88     };
89     // Crear polígono empleando los puntos de arriba
90     PolygonBuilder* constructorDePolig = new PolygonBuilder(SpatialReference::wgs84(), this);
91     constructorDePolig->addPoints(definirPuntos);
92     // Crear simbología del polígono
93     SimpleLineSymbol* simbologiaLineasPolig = new SimpleLineSymbol(SimpleLineSymbolStyle::Solid, QColor(Qt::blue), 3, this);
94     SimpleFillSymbol* llenarSimbolo = new SimpleFillSymbol(SimpleFillSymbolStyle::Solid, QColor(Qt::yellow), simbologiaLineasPolig, this);
95     // Visualizar polígono
96     Graphic* graficarPoligono = new Graphic(constructorDePolig->toGeometry(), llenarSimbolo, this);
97     // Agregar a la capa
98     overlay->graphics()->append(graficarPoligono);
99
```

Proyectar un mapa en ArcGIS SDK – agregar vectores

- **Crear Polígono:**



Proyectar un mapa en ArcGIS SDK – Inicializar el mapa usando un mapa web

- Vamos a remover las librerías

```
#include "Point.h"
```

```
#include "Viewpoint.h"
```

```
#include "SpatialReference.h"
```

```
#include <QFuture>
```

de nuestro archivo MapaBaseA.cpp

Proyectar un mapa en ArcGIS SDK – Inicializar el mapa usando un mapa web

The screenshot shows a code editor window with the following details:

- Project Explorer:** Shows a project named "mapaBaseA" with the following structure:
 - CMakeLists.txt
 - mapaBaseA
 - Header Files: MapaBaseA.h
 - Source Files: main.cpp, MapaBaseA.cpp
 - qml
 - Resources
 - CMake Modules
- Code Editor:** The file "MapaBaseA.cpp" is open, displaying C++ code. The code includes various header files from the ArcGIS Runtime library, such as Point.h, Viewpoint.h, SpatialReference.h, QFuture, Graphic.h, GraphicListModel.h, GraphicsOverlay.h, GraphicsOverlayListModel.h, PolylineBuilder.h, PolygonBuilder.h, SimpleFillSymbol.h, SimpleLineSymbol.h, SimpleMarkerSymbol.h, and SymbolTypes.h. It also includes the Esri::ArcGISRuntime namespace and defines a constructor for the MapaBaseA class that initializes a map with an OsmStandard basemap style.

```
//  
#include "MapaBaseA.h"  
#include "Map.h"  
#include "MapQuickView.h"  
#include "MapTypes.h"  
#include "Point.h"  
#include "Viewpoint.h"  
#include "SpatialReference.h"  
#include <QFuture>  
  
#include "Graphic.h"  
#include "GraphicListModel.h"  
#include "GraphicsOverlay.h"  
#include "GraphicsOverlayListModel.h"  
#include "PolylineBuilder.h"  
#include "PolygonBuilder.h"  
#include "SimpleFillSymbol.h"  
#include "SimpleLineSymbol.h"  
#include "SimpleMarkerSymbol.h"  
#include "SymbolTypes.h"  
  
using namespace Esri::ArcGISRuntime;  
  
MapaBaseA::MapaBaseA(QObject *parent /* = nullptr */)  
    : QObject(parent)  
    , m_map(new Map(BasemapStyle::OsmStandard, this))  
{}  
  
MapaBaseA::~MapaBaseA() {}
```

Proyectar un mapa en ArcGIS SDK – Inicializar el mapa usando un mapa web

- Vamos a remover la línea del constructor , `m_map(new Map(BasemapStyle::OsmStandard, this))` de nuestro archivo `MapaBaseA.cpp`

```
29 |
30     using namespace Esri::ArcGISRuntime;
31
32     MapaBaseA::MapaBaseA(QObject *parent /* = nullptr */)
33         : QObject(parent)
34         , m_map(new Map(BasemapStyle::OsmStandard, this))
35     {}
36
37     MapaBaseA::~MapaBaseA() {}
38
```

Proyectar un mapa en ArcGIS SDK – Inicializar el mapa usando un mapa web

- Vamos a nuestro usuario de ArcGIS online y a copiar el ID de algún mapa base que hallamos empleado anteriormente

Proyectar un mapa en ArcGIS SDK – Inicializar el mapa usando un mapa web

Inicio Galería Mapa Escena Grupos Contenido Organización Victor Augusto Lizcan... salizavi

Alturas y precipitaciones de California

Editar vista en miniatura Este mapa muestra la altimetría y las precipitaciones acumuladas promedio del estado de California Editar Web Map de salizavi Elemento creado: 2 abr 2024 Elemento actualizado: 2 abr 2024 Ver recuento: 10 Agregar a favoritos

Abrir en Map Viewer Abrir en ArcGIS Desktop Abrir en Field Maps Designer Crear aplicación web Compartir

Descripción Agregue una descripción detallada del elemento.

Capas Precipitaciones promedio mensuales acumuladas del estado de California Feature layer Curvas de nivel de California Feature layer Altimetría de California

Información del elemento Más información Bajo Alto Mejora principal: Agregar una descripción

Detalles Tamaño: 43,68 KB Id.: e5a269c05f0f41e4b80ca2fedfd6dd72

Proyectar un mapa en ArcGIS SDK – Inicializar el mapa usando un mapa web

- Vamos a remover el contenido del método setupViewPoint()

```
43
44  void MapaBaseA::setupViewpoint()
45  {
46
47      const Point center(-74.0602115, 4.6607526, SpatialReference::wgs84());    ○ Incomplete type 'Esri::ArcGISRuntime::SpatialRef
48      const Viewpoint viewpoint(center, 10000.0);    ○ Variable has incomplete type 'const Viewpoint'C:/Program Files/ArcGIS SDKs/
49      m_mapView->setViewpointAsync(viewpoint);
50
51 }
52
53 void MapaBaseA::createGraphics(GraphicsOverlay *overlay) {
54
55     const Point Universidad(-74.0602115, 4.6607526, SpatialReference::wgs84());    ○ Incomplete type 'Esri::ArcGISRuntime::Spati
56     // símbolos del punto
57     SimpleLineSymbol* point_outline = new SimpleLineSymbol(SimpleLineStyle::Solid, QColor("blue"), 3, this);
```

Proyectar un mapa en ArcGIS SDK – Inicializar el mapa usando un mapa web

- Lo vamos a reemplazar con:

```
void MapaBaseA::setupViewpoint()
{
    const QString item_id("e5a269c05f0f41e4b80ca2fedfd6dd72");
    const QUrl portal_url(QString("https://arcgis.com/sharing/rest/content/items/" + item_id));
    m_map = new Map(portal_url, this);
    m_mapView->setMap(m_map);
}
```

Proyectar un mapa en ArcGIS SDK – Inicializar el mapa usando un mapa web

```
41     |     return m_mapView;
42     |
43     | }
44     | void MapaBaseA::setupViewpoint()
45     | {
46     |
47     |     const QString item_id("e5a269c05f0f41e4b80ca2fedfd6dd72");
48     |     const QUrl portal_url(QString("https://arcgis.com/sharing/rest/content/items/" + item_id));
49     |     m_map = new Map(portal_url, this);
50     |     m_mapView->setMap(m_map);
51     |
52     | }
53     |
54     | void MapaBaseA::createGraphics(GraphicsOverlay *overlay) {
55     |
56     | }
```

Proyectar un mapa en ArcGIS SDK – Inicializar el mapa usando un mapa web

- Vamos a remover los puntos líneas y polígonos que habíamos creado previamente



```
main.cpp
MapaBaseA.cpp
> qml
> Resources
> CMake Modules

43 void MapaBaseA::setupViewpoint()
44 {
45
46     const QString item_id("e5a269c05f0f41e4b80ca2fedfd6dd72");
47     const QUrl portal_url(QString("https://arcgis.com/sharing/rest/content/items/" + item_id));
48     m_map = new Map(portal_url, this);
49     m_mapView->setMap(m_map);
50
51 }
52
53
54 void MapaBaseA::createGraphics(GraphicsOverlay *overlay) {
55
56 }
57
58
59 // Set the view (created in QML)
60 void MapaBaseA::setMapView(MapQuickView *mapView)
61 {
```

Proyectar un mapa en ArcGIS SDK – Inicializar el mapa usando un mapa web

