ArcGIS sdk

Mapas base

Repaso

```h
#ifndef MAPA_H
#define MAPA_H

namespace Esri::ArcGISRuntime {
class Map
class MapQuickView;
} // namespace Esri::ArcGISRuntime

#include <QObject>

Q_MOC_INCLUDE("MapQuickView.h")

class Mapa : public QObject
{
  Q_OBJECT

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

public:
  explicit Mapa(QObject *parent = nullptr);
  ~Mapa() 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;
};

#endif // MAPA_H
```

Clase Mapa

Mapa.cpp

Define el Contructor

```cpp
Mapa::Mapa(QObject *parent /* = nullptr */)
    : QObject(parent)
    , m_map(new Map(BasemapStyle::ArcGISStreets, this))
{}
```

Define el Destructor

```cpp
26  Mapa::~Mapa() {}
27
```

```h
#ifndef MAPA_H
#define MAPA_H

namespace Esri::ArcGISRuntime {
class Map
class MapQuickView;
} // namespace Esri::ArcGISRuntime

#include <QObject>

Q_MOC_INCLUDE("MapQuickView.h")

class Mapa : public QObject
{
  Q_OBJECT

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

public:
  explicit Mapa(QObject *parent = nullptr);
  ~Mapa() 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;
};

#endif // MAPA_H
```

Clase Mapa

Mapa.cpp

Define el método privado setMapView y mapView

```cpp
// Set the view (created in QML)
void Mapa::setMapView(MapQuickView *mapView)
{
    if (!mapView || mapView == m_mapView) {
        return;
    }

    m_mapView = mapView;
    m_mapView->setMap(m_map);

    emit mapViewChanged();
}
```

```cpp
MapQuickView *Mapa::mapView() const
{
    return m_mapView;
}
```

```h
#ifndef MAPA_H
#define MAPA_H

namespace Esri::ArcGISRuntime {
class Map
class MapQuickView;
} // namespace Esri::ArcGISRuntime

#include <QObject>

Q_MOC_INCLUDE("MapQuickView.h")

class Mapa : public QObject
{
  Q_OBJECT

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

public:
  explicit Mapa(QObject *parent = nullptr);
  ~Mapa() 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;
};

#endif // MAPA_H
```

Clase Mapa

Mapa.cpp

Se establecen las vistas

```cpp
32
33  // Set the view (created in QML)
34  void Mapa::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  }
```

Clase Mapa

main.cpp

Se establece el mapa basec

```
40
41  const QString apiKey = QString("AAPK1d767ca94cab40c38c6fea6cf92462d0JpfEcsGewRwnT0aXrD1dJrNvO-P0py0KdrB-XAMKHWTtOkxblTgtbDEHlF7JN2n-");
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  }
48
```

Archivo

.qml

```
import QtQuick
import QtQuick.Controls
import Esri.mapa

Item {

    // Create MapQuickView here, and create its Map etc. in C++ code
    MapView {
        id: view
        anchors.fill: parent
        // set focus to enable keyboard navigation
        focus: true
    }

    // Declare the C++ instance which creates the map etc. and supply the view
    Mapa {
        id: model
        mapView: view
    }
}
```

Clase Mapa

main.cpp

```
    // Register the map view for QML
    qmlRegisterType<MapQuickView>("Esri.mapa", 1, 0, "MapView");

    // Register the Mapa (QQuickItem) for QML
    qmlRegisterType<Mapa>("Esri.mapa", 1, 0, "Mapa");

    // Initialize application view
    QQmlApplicationEngine engine;

    // Add the import Path
    engine.addImportPath(QDir(QCoreApplication::applicationDirPath()).filePath("qml"));

    // Set the source
    engine.load(QUrl("qrc:/qml/main.qml"));
```

Clase Mapa

Mapa.h

```h
#ifndef MAPA_H
#define MAPA_H

namespace Esri::ArcGISRuntime {
class Map
class MapQuickView
} // namespace Esri::ArcGISRuntime

#include <QObject>

Q_MOC_INCLUDE("MapQuickView.h")

class Mapa : public QObject
{
   Q_OBJECT

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

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

signals:
   void mapViewChanged();

private:
   Esri::ArcGISRuntime::MapQuickView *mapView() const;
   void setMapView(Esri::ArcGISRuntime::MapQuickView *mapView);
   void setupViewpoint();
   Esri::ArcGISRuntime::Map *m_map = nullptr;
   Esri::ArcGISRuntime::MapQuickView *m_mapView = nullptr;
};

#endif // MAPA_H
```

Mapa.cpp

```cpp
Mapa::Mapa(QObject *parent /* = nullptr */)
    : QObject(parent)
    , m_map(new Map(BasemapStyle::ArcGISStreets, this))
{}

Mapa::~Mapa() {}

MapQuickView *Mapa::mapView() const
{
    return m_mapView;
}

void Mapa::setupViewpoint(){
    const Point centrarPunto(-76.43, 3.21, SpatialReference::wgs84());
    const Viewpoint vistaEscala(centrarPunto,1000);
    m_mapView->setViewpointAsync(vistaEscala);
}

// Set the view (created in QML)
void Mapa::setMapView(MapQuickView *mapView)
{
    if (!mapView || mapView == m_mapView) {
        return;
    }

    m_mapView = mapView;
    m_mapView->setMap(m_map);

    setupViewpoint();

    emit mapViewChanged();
}
```

Se enfoca la vista en un lugar especifico

Se establece la vista

Librerías requeridas

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

```h
#ifndef MAPA_H
#define MAPA_H

namespace Esri::ArcGISRuntime {
class Map
class MapQuickView
class GraphicsOverlay;
}// namespace Esri::ArcGISRuntime

#include <QObject>

Q_MOC_INCLUDE("MapQuickView.h")

class Mapa : public QObject
{
  Q_OBJECT

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

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

signals:
  void mapViewChanged();

private:
  Esri::ArcGISRuntime::MapQuickView *mapView() const;
  void setMapView(Esri::ArcGISRuntime::MapQuickView *mapView);
  void setupViewpoint();
  void crearVectores(Esri::ArcGISRuntime::GraphicsOverlay* capas);
  Esri::ArcGISRuntime::Map * m_map = nullptr;
  Esri::ArcGISRuntime::MapQuickView * m_mapView = nullptr;
};

#endif // MAPA_H
```

Clase Mapa
Mapa.cpp

Define el método privado crearVectores

```
54
55  void Mapa::crearVectores(GraphicsOverlay *capas)
56  {
57
58  }
59
```

Se establecen las vistas

```
71
72      GraphicsOverlay* capas = new GraphicsOverlay(this);
73      crearVectores(capas);
74      m_mapView->graphicsOverlays()->append(capas);
75
```

Librerías requeridas

```
#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"
```

```cpp
#ifndef MAPA_H
#define MAPA_H

namespace Esri::ArcGISRuntime {
class Map
class MapQuickView
class GraphicsOverlay;
}// namespace Esri::ArcGISRuntime

#include <QObject>

Q_MOC_INCLUDE("MapQuickView.h")

class Mapa : public QObject
{
  Q_OBJECT

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

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

signals:
  void mapViewChanged();

private:
  Esri::ArcGISRuntime::MapQuickView *mapView() const;
  void setMapView(Esri::ArcGISRuntime::MapQuickView *mapView);
  void setupViewpoint();
  void crearVectores(Esri::ArcGISRuntime::GraphicsOverlay* capas);
  Esri::ArcGISRuntime::Map * m_map = nullptr;
  Esri::ArcGISRuntime::MapQuickView * m_mapView = nullptr;
};

#endif // MAPA_H
```

```cpp
void Mapa::crearVectores(GraphicsOverlay *capas)
{

    const Point dibujarPunto(-76.43, 3.21, SpatialReference::wgs84());
    // Create symbols for the point
    SimpleLineSymbol* lineaExteriorPunto = new SimpleLineSymbol(SimpleLineSymbolStyle::Solid, QColor("blue"), 3, this);
    SimpleMarkerSymbol* rellenoPunto = new SimpleMarkerSymbol(SimpleMarkerSymbolStyle::Circle, QColor("red"), 10, this);
    rellenoPunto->setOutline(lineaExteriorPunto);

    // Create a graphic to display the point with its symbology
    Graphic* point_graphic = new Graphic(dibujarPunto, rellenoPunto, this);
    // Add point graphic to the graphics overlay
    capas->graphics()->append(point_graphic);


}
```

Define el método privado crearVectores

Se crea un vector punto

Clase Mapa
Mapa.cpp

```cpp
#ifndef MAPA_H
#define MAPA_H

namespace Esri::ArcGISRuntime {
class Map
class MapQuickView
class GraphicsOverlay;
}// namespace Esri::ArcGISRuntime

#include <QObject>

Q_MOC_INCLUDE("MapQuickView.h")

class Mapa : public QObject
{
  Q_OBJECT

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

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

signals:
  void mapViewChanged();

private:
  Esri::ArcGISRuntime::MapQuickView *mapView() const;
  void setMapView(Esri::ArcGISRuntime::MapQuickView *mapView);
  void setupViewpoint();
  void crearVectores(Esri::ArcGISRuntime::GraphicsOverlay* capas);
  Esri::ArcGISRuntime::Map * m_map = nullptr;
  Esri::ArcGISRuntime::MapQuickView * m_mapView = nullptr;
};

#endif // MAPA_H
```

```cpp
void Mapa::crearVectores(GraphicsOverlay *capas)
{

    PolylineBuilder* dibujarLinea = new PolylineBuilder(SpatialReference::wgs84(), this);
    dibujarLinea->addPoint(-76.43, 3.21);
    dibujarLinea->addPoint(-76.42, 3.20);
    // Create a symbol for the line
    SimpleLineSymbol* line_symbol = new SimpleLineSymbol(SimpleLineSymbolStyle::Solid, QColor(Qt::blue), 3, this);

    // Create a graphic to display the line with its symbology
    Graphic* graficarLinea = new Graphic(dibujarLinea->toGeometry(), line_symbol, this);
    // Add line graphic to the graphics overlay
    capas->graphics()->append(graficarLinea);


}
```

Define el método privado crearVectores

Se crea un vector linea

Clase Mapa
Mapa.cpp

```cpp
#ifndef MAPA_H
#define MAPA_H

namespace Esri::ArcGISRuntime {
class Map
class MapQuickView
class GraphicsOverlay;
}// namespace Esri::ArcGISRuntime

#include <QObject>

Q_MOC_INCLUDE("MapQuickView.h")

class Mapa : public QObject
{
  Q_OBJECT

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

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

signals:
  void mapViewChanged();

private:
  Esri::ArcGISRuntime::MapQuickView *mapView() const;
  void setMapView(Esri::ArcGISRuntime::MapQuickView *mapView);
  void setupViewpoint();
  void crearVectores(Esri::ArcGISRuntime::GraphicsOverlay* capas);
  Esri::ArcGISRuntime::Map * m_map = nullptr;
  Esri::ArcGISRuntime::MapQuickView * m_mapView = nullptr;
};

#endif // MAPA_H
```

```cpp
void Mapa::crearVectores(GraphicsOverlay *capas)
{

    const QList<Point> puntosDelPoligono = {
        Point(-76.43, 3.21),
        Point(-76.42, 3.20),
        Point(-76.40, 3.18),
    };
    // Create a polygon using the list of points above
    PolygonBuilder* construirPoligono = new PolygonBuilder(SpatialReference::wgs84(), this);
    construirPoligono->addPoints(puntosDelPoligono);
    // Create symbols for the polygon
    SimpleLineSymbol* pintarBordeDelPoligono = new SimpleLineSymbol(SimpleLineSymbolStyle::Solid, QColor(Qt::blue), 3, this);
    SimpleFillSymbol* RellenarPoligono = new SimpleFillSymbol(SimpleFillSymbolStyle::Solid, QColor(Qt::yellow), pintarBordeDelPoligono, this)
    // Create a graphic to display the polygon with its symbology
    Graphic* polygon_graphic = new Graphic(construirPoligono->toGeometry(), RellenarPoligono, this);
    // Add polygon graphic to the graphics overlay
    capas->graphics()->append(polygon_graphic);

}
```

Define el método privado crearVectores

Se crea un vector poligono

Clase Mapa
Mapa.cpp