Roteiro 02 – Arcgis API - Navegação

1. INTRODUÇÃO

Neste roteiro vamos explorar API ArcGis Runtime SDK para Android, para desenvolvermos um App de navegação. Esta api é fornecida pela ESRI, que é a desenvolvedora do software ArcGis, o principal software pago do mercado de geoprocessamento.

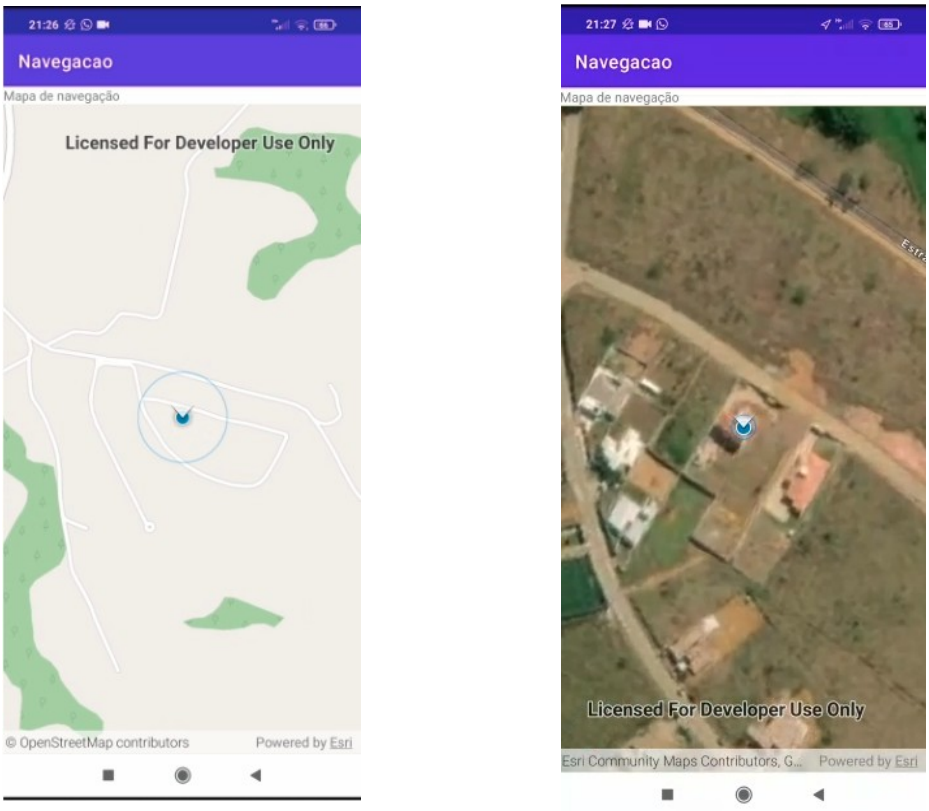

Figura 1 – Aplicativo navegação com visualização de mapas. (a) usando basemap.type.open_street_view. (b) usando basemap.type.imagery.

A vantagem de se utilizar esta api está no fato de ser uma tecnologia de ponta e estar facilmente disponível para testes e desenvolvimento. O ponto negativo é que a tecnologia ArcGis é comercial que geralmente custa muito caro.

Leia também: https://developers.arcgis.com/labs/

2. PRÁTICA

Neste roteiro vamos criar uma aplicação na linguagem Java para exibir um mapa (Street _vector por exemplo) e ir renderizando a posição atual. Ou seja, faremos um App de navegação.

Passo 1. Criar novo projeto

Abra o AndroidStudio e crie um novo projeto configurado para a linguagem Java. Defina o nome **Navegação** para o seu projeto.

Passo 2. Adicionar as dependências

Agora que o seu projeto já está criado vamos adicionar as dependências da api do ArcGis, para que assim seu projeto faça acesso e uso da API. Para isso você fará alterações dentro dos arquivos build.gradle do projeto e do app.
Modifique o arquivo **build.gradle** em nível do projeto da seguinte forma:

```
Buildscript {
 ...
}

allprojects {

    repositories {
        ...
        // adicionar estas linhas dentro de repositorios de allprojects
        maven {
            url 'https://esri.jfrog.io/artifactory/arcgis'
        }
    }
}
```

Agora modifique o seu **build.gradle** no nível do módulo app da seguinte forma:

```
apply plugin: 'com.android.application'
android {
    ...
    // adicionar estas opções no compileOptions
    compileOptions {
        sourceCompatibility JavaVersion.VERSION_1_8
        targetCompatibility JavaVersion.VERSION_1_8
    }
}
dependencies {
    ...
     // adicionar estas linhas dentro das dependencias
    implementation 'com.esri.arcgisruntime:arcgis-android:100.10.0'
}
```

Pronto, agora sincronize o seu projeto no menu *File → Sync Project With Gradle Files*.

Passo 3. Editar o AndroidManifest.xml

Neste momento vamos editar o AndroidManifest.xml para adicionar a permissão para acesso ao GPS, à internet e também a opção para exigir a biblioteca de imagem **opengl** versão 2.

1. Definir permissões	…/app/src/main/AndroidManifest.xml

```
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
```

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-feature android:glEsVersion="0x00020000" android:required="true" />
```

Passo 4. Definir o layout principal

Vamos incluir o mapa dentro do nosso layout. Vamos inserir uma **view** do tipo **MapView**. Esta view é definida na Api do ArcGis. Por isso foi preciso adicionar as dependências no gradle.

2. Definir o Layout — activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity"
    android:orientation="vertical">
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Nosso mapa!" />
    <com.esri.arcgisruntime.mapping.view.MapView
        android:id="@+id/mapView"
        android:layout_width="match_parent"
        android:layout_height="match_parent" >
    </com.esri.arcgisruntime.mapping.view.MapView>
</LinearLayout>
```

Passo 5. Inicializar o Mapa

Inicie o mapa com a camada de vetores das ruas, disponibilizada pela Esri. Além disso, defina as coordenadas geográficas e o nível de zoom do mapa. Esta coordenada você pode setar diretamente na mão, ou criar um botão para pegar automaticamente, como vimos nas aulas anteriores.

3. Inicializar o mapa — MainActivity.java

```
package com.example.mapa01;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import com.esri.arcgisruntime.mapping.ArcGISMap;
import com.esri.arcgisruntime.mapping.Basemap;
import com.esri.arcgisruntime.mapping.view.MapView;
public class MainActivity extends AppCompatActivity {
    private MapView mMapView;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
```

```
        mMapView = findViewById(R.id.mapView);
        setupMap();
    }


 private void setupMap() {
    if (mMapView != null) {
       Basemap.Type basemapType = Basemap.Type.STREETS_VECTOR;
       double latitude = -21.2526;
       double longitude = -43.1511;
       int levelOfDetail = 20;
       ArcGISMap map = new ArcGISMap(basemapType,latitude,longitude,levelOfDetail);
       mMapView.setMap(map);
     }
 }
    @Override
    protected void onPause() {
        if (mMapView != null) {
            mMapView.pause();
        }
        super.onPause();
    }
    @Override
    protected void onResume() {
        super.onResume();
        if (mMapView != null) {
            mMapView.resume();
        }
    }
    @Override
    protected void onDestroy() {
        if (mMapView != null) {
            mMapView.dispose();
        }
        super.onDestroy();
    }
} // fim da classe principal
```

O mapa de base, “Basemap.Type” pode ser de diferentes tipos, faça alguns testes. Na minha região por exemplo `Basemap.Type.STREETS_VECTOR` não está atualizado com a minha rua. Já o `Basemap.Type.OPEN_STREET_MAP` sim, portanto no meu projeto usei o open_street_map.

Também possível usar mapas base de orthophotos (imagens que apresentam ‘fotos’ de nossas casas), de topografia(usaremos quando formos apresentar uma hidrografia por exemplo), e etc. Tem também a opção de mostrar o mapa no formato dia e noite conferindo maior conforto a visão.

Passo 6. Preparar o LocationDisplay

Vamos configurar o atributo **LocationDisplay** do **MapView** para exibir a localização atual. Além disso, aqui se configura o modo automático de movimentação do mapa conforme a localização e a bússola do dispositivo.

Leia também: https://developers.arcgis.com/android/api-reference/reference/com/esri/arcgisruntime/mapping/view/LocationDisplay.html

2. Configurar o LocationDisplay — MainActivity.java

```
// Declarar como atributo da classe principal
private LocationDisplay mLocationDisplay;

// Criar a função; depois incluir sua chamada no oncreate
private void setupLocationDisplay() {
    mLocationDisplay = mMapView.getLocationDisplay();
    mLocationDisplay.setAutoPanMode(LocationDisplay.AutoPanMode.COMPASS_NAVIGATION);
    mLocationDisplay.startAsync();
}
```

Passo 7. Inicializar o GPS

Neste momento vamos criar a função **setuGPS()** que verificará e, se necessário, solicitar as permissões de acesso ao GPS. Além disso, criamos uma função lambda para ficar de ouvinte das alterações do status de localização.

3. Inicializar GPS — MainActivity.java

```
//incluir a chamada desta função no oncreate, logo abaixo da chamada da função setuplocationdisplay().
 private void setupGPS(){
    // Listen to changes in the status of the location data source.
    mLocationDisplay.addDataSourceStatusChangedListener(dataSourceStatusChangedEvent-> {
        if (dataSourceStatusChangedEvent.isStarted() || dataSourceStatusChangedEvent.getError()== null) {
            return;
        }
        int requestPermissionsCode = 2;
        String[] requestPermissions = new String[]{Manifest.permission.ACCESS_FINE_LOCATION,
Manifest.permission.ACCESS_COARSE_LOCATION};
        if (!(ContextCompat.checkSelfPermission(MainActivity.this, requestPermissions[0]) ==
PackageManager.PERMISSION_GRANTED && ContextCompat.checkSelfPermission(MainActivity.this,
requestPermissions[1]) == PackageManager.PERMISSION_GRANTED)) {
            ActivityCompat.requestPermissions(MainActivity.this, requestPermissions,
requestPermissionsCode);
        } else {
            Toast.makeText(MainActivity.this, "Erro.", Toast.LENGTH_LONG).show();
        }
    });
}

@Override
public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions, @NonNull int[]
grantResults) {
    if (grantResults.length > 0 && grantResults[0] == PackageManager.PERMISSION_GRANTED) {
        mLocationDisplay.startAsync();
    } else {
        Toast.makeText(MainActivity.this, "permissão recusada", Toast.LENGTH_SHORT).show();
    }
}
```

Passo 8. Incluir chamadas na OnCreate

Edite a função OnCreate e adicione as chamadas para os métodos criados.

4. Chamar funções	MainActivity.java

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    mMapView = findViewById(R.id.mapView);
    setupMap();
    setupLocationDisplay();
    setupGPS();
}
```

9. Configurar o GPS virtual do emulador

Se você estiver utilizando o emulador para testar a sua aplicação, configure a localização do GPS virtual. É bem simples, execute o emulador. Em seguida, clique no botão de configurações do emulador (Item 1 da Figura 2). Escolha a aba Localização e clique no mapa para inserir um marcador na posição desejada. Por fim, clique em “SET LOCATION” para definir a posição atual.

10. Considerações finais

A adoção da api do ArcGis é uma forma fácil e rápida de incorporar mapas à suas aplicações Android. Para a aplicação de desenvolvimento o recurso está disponível com a marca d’água “Licensed For Developer User Only” e para licenciar existe um custo, o que não é impeditivo para projetos que possuem orçamento. O ideal seria a adoção de ferramentas livres e sem custo operacional, mas em se tratando apis para mapeamento no Android as principais são comerciais. Para a web existem boas alternativas. Nos próximos roteiros vamos incrementar esta solução com mais recursos.

11. Exercícios

Crie o aplicativo para exibir um mapa como apresentado no roteiro.

Envie o seu programa para o professor.