Estudo e experimentação do Android

Sistemas de localização

Mapas

Sistemas de localização

- Na grande maioria dos dispositivos móveis atuais são disponibilizados diversos meios que auxiliam a localização dos equipamentos
  - GPS
  - Triangulação GSM
  - Wi-Fi
  - …
- Estes sistemas permitem o acesso a informação sobre localização
  - Latitude
  - Longitude
  - Altitude
  - Precisão da informação
  - …

Location Manager

- A informação sobre localização pode ser obtida recorrendo ao serviço *Location Manager* do *Android*

```
val lm = getSystemService(LOCATION_SERVICE) as LocationManager
```

- Com o auxílio do *Location Manager* podemos manifestar o interesse em receber informação sobre diversas fontes de localização

```
lm.requestLocationUpdates( provider : String,
                           minTime : Long, minDistance : Float,
                           listener : LocationListener)
```

Location Manager

- Parâmetros da função `requestLocationUpdates`
  - *Provider* – fonte da informação
    - `GPS_PROVIDER`
    - `NETWORK_PROVIDER`
    - `PASSIVE_PROVIDER`
    - `FUSED_PROVIDER` (API 31)
  - *minTime* – intervalo mínimo (*ms*) entre atualizações de localização
  - *minDistance* – distância mínima (*m*) entre atualizações de localização
  - *Listener* – *listener* que será chamado sempre que existir uma atualização de informação
    - Implementado por uma classe existente
    - Definir uma nova classe para o efeito
    - Definir uma classe anónima
    - Necessário definir o método abstrato do interface `LocationListener`
      - onLocationChange
    - Podemos executar o método `requestLocationUpdates` para manifestar o interesse em diversos tipos de localização e, eventualmente, fazer o seu processamento usando o mesmo *listener*

Location Manager

- O projecto tem que pedir permissões para aceder aos métodos de localização
  - Incluir no manifesto estruturas `<uses-permission …>` adequadas
    - GSM

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

    - GPS

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

- Quando a aplicação já não necessitar de *updates* deve usar a função `removeUpdates` para o *listener* registado

Location Services

- Existe um conjunto de objetos que permitem gerir de forma mais eficaz a obtenção da localização
  - `LocationServices`, `LocationRequest`, ...
  - Disponível na biblioteca `play-services-location`

```
implementation 'com.google.android.gms:play-services-location:18.0.0'
```

Autonomia energética vs localização

- Uma vantagem que se pode obter dos novos objetos é, por exemplo, uma melhor gestão da bateria
- A utilização de sistemas de localização podem ter um impacto significativo na autonomia energética de um dispositivo
  - Escolher os métodos mais adequados para os objetivos da aplicação
    - *Será necessário obter uma localização com precisão elevada para obter a previsão do tempo?*

Criar uma configuração de localização

- Parâmetros a ter em consideração no `LocationRequest`
  - Precisão da informação
    - `priority : Int`
      - `PRIORITY_BALANCED_POWER_ACCURACY (100m)`
      - `PRIORITY_HIGH_ACCURACY`
      - `PRIORITY_LOW_POWER (10km)`
      - `PRIORITY_NO_POWER`
  - Frequência com que se recebe
    - `interval : Long`
    - `fastestInterval : Long`
  - Latência
    - `maxWaitTime : Long`

Exemplo LocationRequest

```
...
fLoc = FusedLocationProviderClient(this)
fLoc.requestLocationUpdates(locReq, locationCallback, null)
...

val locReq = LocationRequest.create().apply {
    interval = 4000
    fastestInterval = 2000
    priority = LocationRequest.PRIORITY_HIGH_ACCURACY
    maxWaitTime = 10000
}

var locationCallback = object : LocationCallback() {
    override fun onLocationResult(p0: LocationResult?) {
        Log.i(TAG, "onLocationResult: ")
        p0?.locations?.forEach {
            Log.i(TAG, "locationCallback: ${it.latitude}  ${it.longitude}")
        }
    }
}
```

Outras capacidades de localização

- *GeoFencing*
  - https://developer.android.com/training/location/geofencing
- Deteção do tipo de atividade do utilizador
  - Andar a pé, de bicicleta, de carro, ...
  - https://developer.android.com/guide/topics/location/transitions

Google Maps

- Adicionar *Google Play Services* ao projecto
  - Adicionar às dependências ("*dependencies*") do ficheiro `build.gradle` do módulo a seguinte linha, onde deverá ser alterada a versão para a que se tem instalada no ambiente

```
implementation 'com.google.android.gms:play-services-maps:18.0.0'
```

  - Com esta linha apenas está a ser incluída a biblioteca referente ao serviço *Maps*. Pode-se incluir toda a biblioteca `play-services`

Google Maps

- Incluir no elemento `<application … />` no ficheiro de manifesto

```
<meta-data
      android:name="com.google.android.gms.version"
      android:value="@integer/google_play_services_version" />
```

- *Nota: esta configuração é realizada por omissão quando são incluídas as bibliotecas referidas anteriormente*

Google Maps

- Obter a chave de assinatura da aplicação usada na criação do *apk*
  - Em modo *debug*
    - Abrir uma consola no diretório "`.android`" existente no diretório do perfil do utilizador
    - Executar (`keytool` é uma ferramenta do Java SDK):

```
keytool -list -v -keystore debug.keystore
        -alias androiddebugkey -storepass android
        -keypass android
```

Copy&paste: `keytool -list -v -keystore debug.keystore -alias androiddebugkey -storepass android -keypass android`

    - Copiar a chave SHA1
  - Em modo *release*
    - Semelhante mas usando o certificado específico criado para a assinatura da aplicação a distribuir através da *PlayStore*

Google Maps

- Obter um certificado e uma chave
  - Aceder ao site *Google APIs Console*

    `https://console.cloud.google.com/`
    `https://console.developers.google.com/`
  - Criar um novo projeto
  - Ativar a API *Google Maps SDK for Android*
  - Adicionar "credenciais" do tipo *API key*
    - *Restrict Key* + Associar uma *API Key Android*
    - Preencher o campo *Package Name*
    - Preencher o campo *SHA-1*
    - Colocar a chave no formato: `<SHA1>;<package app>`
      - Na nova versão da consola existem campos separados
  - Copiar a *API key* criada e adicionar ao elemento `<application … />` o seguinte elemento:

```
<meta-data
    android:name="com.google.android.geo.API_KEY"
    android:value="<colocar a API_KEY>"/>
```

Google Maps

- Incluir no ficheiro de manifesto o pedido de permissão para acesso à Internet

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

  - As versões atuais das bibliotecas já incluem estas permissões

  - Opcionalmente incluir

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

- Exigir o suporte para *OpenGL ES v2*

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

  - As versões atuais das bibliotecas já incluem este pedido

Google Maps

- Adicionar o elemento visual (*fragment*) do *Google Maps*

```
<fragment
          android:id="@+id/map"
          android:layout_width="match_parent"
          android:layout_height="match_parent"
          android:name="com.google.android.gms.maps.SupportMapFragment"/>
```

- Testar a aplicação

MapFragment e GoogleMap

- No *slide* anterior exemplificou-se a inclusão do fragmento através do ficheiro *xml* de *layout*. Caso se queira adicionar em *runtime*

```
var mMapFragment = SupportMapFragment.newInstance()
val fragmentTransaction: FragmentTransaction =
supportFragmentManager.beginTransaction()
fragmentTransaction.add(R.id.my_container, mMapFragment)
fragmentTransaction.commit()
```

- Para obter referência para o mapa

```
val map: GoogleMap =
   (supportFragmentManager.findFragmentById(R.id.map) as SupportMapFragment)
                                                        .getMap() // deprecated
```

Exemplo de configurações

```
class MainActivity : AppCompatActivity(),OnMapReadyCallback {
  val ISEC = LatLng(40.1925, -8.4115)
  val DEIS = LatLng(40.1925, -8.4128)
...
    (supportFragmentManager.findFragmentById(R.id.map) as? SupportMapFragment)?.getMapAsync(this)
...
  @SuppressLint("MissingPermission")
  override fun onMapReady(map: GoogleMap) {
    map.isMyLocationEnabled = true
    map.mapType = GoogleMap.MAP_TYPE_HYBRID
    map.uiSettings.isCompassEnabled = true
    map.uiSettings.isZoomControlsEnabled = true
    map.uiSettings.isZoomGesturesEnabled = true
    val cp = CameraPosition.Builder().target(ISEC).zoom(17f)
            .bearing(0f).tilt(0f).build()
    map.animateCamera(CameraUpdateFactory.newCameraPosition(cp))
    map.addCircle(CircleOptions().center(ISEC).radius(150.0)
            .fillColor(Color.argb(128, 128, 128, 128))
            .strokeColor(Color.rgb(128, 0, 0)).strokeWidth(4f))
    val mo = MarkerOptions().position(ISEC).title("ISEC-IPC")
            .snippet("Instituto Superior de Engenharia de Coimbra")
    val isec = map.addMarker(mo)
    isec?.showInfoWindow()
    map.addMarker(MarkerOptions().position(DEIS).title("DEIS-ISEC"))
  }
}
```