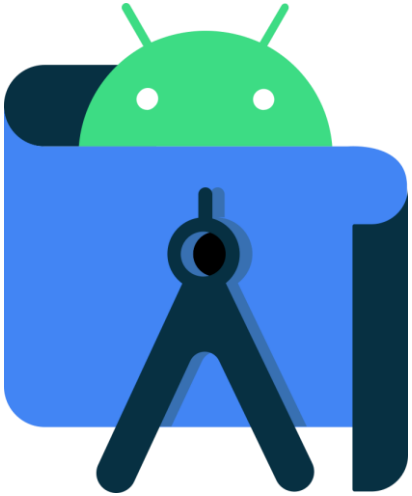
Universidade Federal do Amazonas
Instituto de Computação
IArTES - Especialização em IA para Engenharia de Testes de Software



Mapas no Android

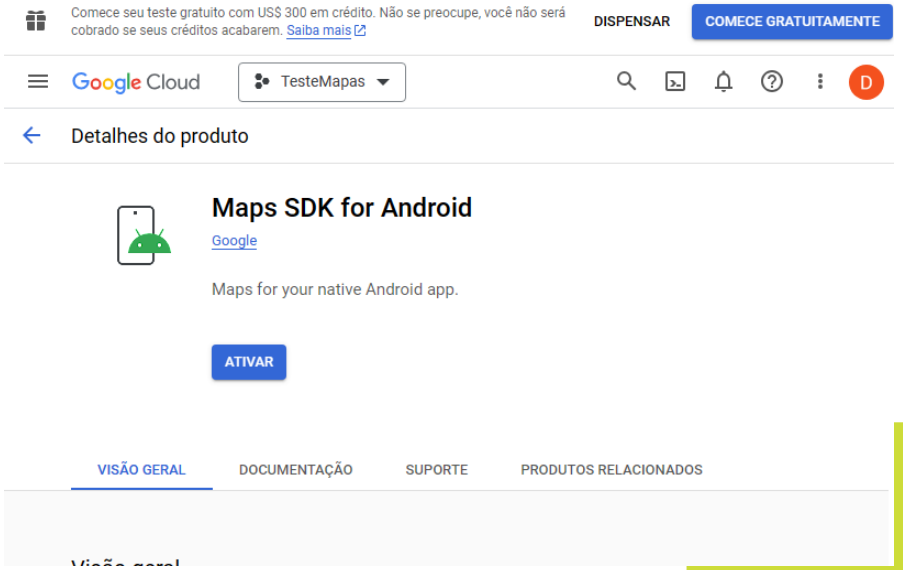
Prof. Diogo Soares

diogosoaresm@ufam.edu.br

Github: github.com/diogosm/android_class_IArTES

Mapas

- Um recurso bastante interessante do Android é o uso do mapa, comum em apps como Uber, Ifood, Whatsapp, etc.
- A biblioteca mais conhecida é a da google Maps via Maps SDK
  - Ótima para produção
  - Ruim para programação inicial
    - *Custos e limites*
    - *Complexo de utilizar*



Mapas

- Por isso iremos utilizar uma biblioteca bastante conhecida da comunidade dos maps
  - A OSM, ou Open Street Map
  - Uma fonte (não apenas código, mas fonte de dados de mapa também!) aberta de mapas que serve tanto para edição (como uma wikipedia de mapas) como para utilizar no lugar da solução da google

- Exemplo (UFAM): OpenStreetMap

Mapas

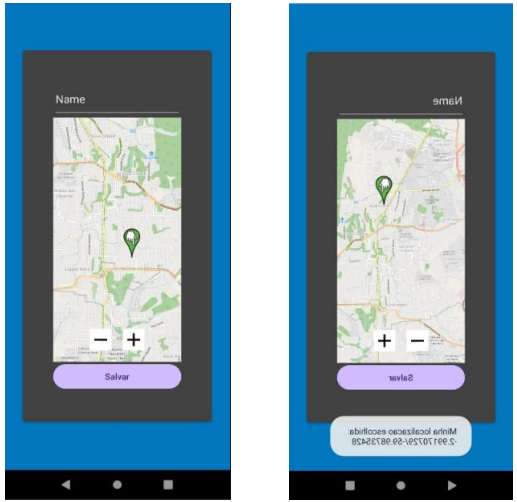
- Sendo assim, usaremos uma lib no android que pega dados do OSM para utilizar em apps android (também aberta!!!)
  - Lib: OSMBonusPack (GitHub - MKergall/osmbonuspack: A third-party library of (very) useful additional objects for osmdroid)
  - Suporte aos mapas, marcadores, rotas, POIs e até desenhos

- Vamos montar um exemplo pegando a localização do gps, com opção de mudar, e imprimindo via Toast

Mapas

- Iremos dividir este slide-tutorial em partes
  - Parte 1: requirements e configuração
  - Parte 2: montando a view
  - Parte 3: Montando a Activity
  - Parte 4: Recebendo os dados após ações de clique em botão

- Resultado esperado:



Mapas – Parte 1

- Primeiramente é preciso adicionar a lib
  - Como ela não está no repositório padrão (da google ou do maven), é preciso adicionar o repositório
  - Há várias formas de fazer isso, mas uma é:
    - *Vá até o arquivo settings.gradle e modifque o bloco abaixo para conter o repo https://jitpack.io*

```
dependencyResolutionManagement {
    repositoriesMode.set(RepositoriesMode.FAIL_ON_PROJECT_REPOS)
    repositories {
        google()
        mavenCentral()
        maven { url "https://jitpack.io" }
    }
}
```

Mapas – Parte 1

- Após isso, adicione a lib no build.gradle

```
implementation 'com.github.MKergall:osmbonuspack:6.9.0'
```

- Sincronize e voilá

Mapas – Parte 1

- Após isso, adicione a lib no build.gradle

```
implementation 'com.github.MKergall:osmbonuspack:6.9.0'
```

- Sincronize e biblioteca instalada, agora vamos configurar o que for pré-requisito

Mapas – Parte 1

- Como estamos lidando com mapas, é natural pensar que vários recursos do device estão sendo utilizados:
  - Internet (para baixar os mapas)
  - GPS (para localização, óbvio)
- Sendo assim, precisamos adicionar algumas permissões no nosso manifesto. As principais:

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

- O código acima deve ser suficiente para pedir a permissão do usuário, mesmo assim também adicione isso (próximo slide):

Mapas – Parte 1

- Logo abaixo do setContentView na função onCreate do activity, verifique se as permissões estão ok para seu app com o código abaixo:

```
if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
    if (ContextCompat.checkSelfPermission(this, Manifest.permission.INTERNET) != PackageManager.PERMISSION_GRANTED ||
            ContextCompat.checkSelfPermission(this, Manifest.permission.WRITE_EXTERNAL_STORAGE) != PackageManager.PERMISSION_GRANTED) {
        String[] permissoes = {Manifest.permission.INTERNET, Manifest.permission.WRITE_EXTERNAL_STORAGE};
        requestPermissions(permissoes, 1);
    }
}
```

- O 1 em request permissions é só um valor para seu app identificar que as permissões estão setadas

Mapas – Parte 2

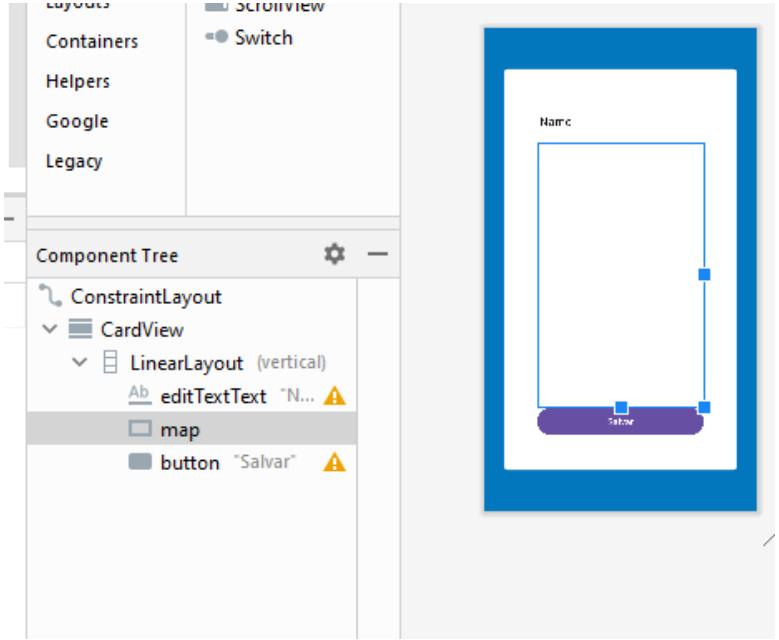
- Agora vamos adicionar no nosso layout.
- Para isso iremos o componente MapView na tela (não o MapView padrão que já existe no editor do android!)
- Mas, o MapView da lib que acabamos de instalar. Inclusive essa é a hora de vê se a instalação da lib foi correta!

```
<org.osmdroid.views.MapView
   android:id="@+id/map"
   android:layout_width="fill_parent"
   android:layout_height="400dp" />
```

- Percebam que o componente basicamente tem um id, largura e altura apenas. É o suficiente para começar!

Mapas – Parte 2

- Adicione componentes para deixar com uma cara mais próxima do que você deseja
- Exemplo:



Mapas – Parte 2

- Neste ponto sua aplicação já pode executar na AVD/Device físico, agora iremos para os códigos dos controllers.

Mapas – Parte 3

- Para adicionar nosso código java, precisamos recuperar o mapa através do findViewById e setar os parâmetros necessários para inicialização do mapa como se permite zoom, multitouch, ponto central, etc. No exemplo, centralizei no x,y da cidade de Manaus

```
//Pega o mapa adicionada no arquivo activity_main.xml
MapView mapa = (MapView) findViewById(R.id.map);
mapa.setTileSource(TileSourceFactory.MAPNIK);
//seta algumas configuracoes ao mapa
mapa.setBuiltInZoomControls(true);
mapa.setMultiTouchControls(true);

//Cria um controller para setar posicoes no mapa
IMapController mapController = mapa.getController();
mapController.setZoom(9.5);
GeoPoint startPoint = new GeoPoint(-3.10719, -60.0261); // ponto inicial
mapController.setCenter(startPoint);
```

Mapas – Parte 3

- Após isso, você pode adicionar algum marcador inicial para sua posição inicial, se quiseres, ou escolher uma posição fixa para ser a inicial (como a UFAM). Para isso, iremos de novo utilizar o gps do device. O GPS_PROVIDER pode ser trocado para WIFI_PROVIDER para quem quiser testar precisão, etc

```
//pega minha pos atual
LocationManager locationManager = (LocationManager) getSystemService(Context.LOCATION_SERVICE);
Location location = locationManager.getLastKnownLocation(LocationManager.GPS_PROVIDER);
GeoPoint minhaLoc = null;
if (location != null) {
   Log.i("debug", "uma localizacao conhecida encontrada!");
   double latitude = location.getLatitude();
   double longitude = location.getLongitude();
   minhaLoc = new GeoPoint(latitude, longitude);
}
```

Mapas – Parte 3

- Caso a localização seja obtida com sucesso, podemos criar um marker (marcador tipo aqueles vermelhos do google maps) no mapa com essa localização

```
if(minhaLoc != null){
    Marker startMarker = new Marker(mapa);
    startMarker.setPosition(minhaLoc);
    startMarker.setTitle("Ponto Inicial");
    startMarker.setAnchor(Marker.ANCHOR_CENTER, Marker.ANCHOR_BOTTOM);
    mapa.getOverlays().add(startMarker);

    //inicia localizacao
    localizacaoEscolhida = minhaLoc;
}
```

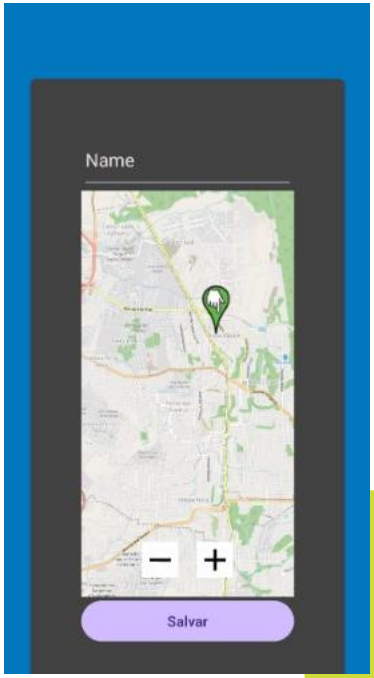
- Percebam que no exemplo aparece uma `localizaçãoEscolhida`, não usada até aqui

Mapas – Parte 3

- `localizaçãoEscolhida` será utilizada mais pra frente, e por isso definimos como um atributor da classe MainActivity.java

```
private GeoPoint localizacaoEscolhida;
```

- Isso é o suficiente para termos nosso mapa funcionando com uma posição inicial já marcada no mapa



Mapas – Parte 4

- Agora podemos adicionar eventos ao nosso mapa, iremos fazer isso através MapEventsReceiver() da OSM.

```
//cria uma variavel pra recebver eventos no mapa
MapEventsReceiver mReceive = new MapEventsReceiver(){
   @Override
   public boolean singleTapConfirmedHelper(GeoPoint p) {
      //troca de localizacao com um toque simples
      localizacaoEscolhida = p;
      //como há apenas um marker sempre, apaga o primeiro
      mapa.getOverlays().remove(
            mapa.getOverlays().size() - 1
      );
      //cria novo marcador
      Marker startMarker = new Marker(mapa);
      startMarker.setPosition(localizacaoEscolhida);
      startMarker.setTitle("Ponto Inicial");
      startMarker.setAnchor(Marker.ANCHOR_CENTER, Marker.ANCHOR_BOTTOM);
      mapa.getOverlays().add(startMarker);
      return false;
   }
   @Override
   public boolean longPressHelper(GeoPoint p) {
      // faz nada
      return false;
   }
};
```

Mapas – Parte 4

- Isso é feito através da implementação das funções já existentes na classe principal, `singleTapConfirmedHelper` e `longPressHelper`.
- Por enquanto só utilizaremos a primeira, que corresponde à quando o usuário dá um pequeno toque na tela do mapa
- Percebam que na implementação feita neste exemplo, apenas deixei um marcador sempre, isto é, se uma nova posição era escolhida, apaga a anterior. Verifique os requisitos do que estiver tentando fazer, pois os marcadores são adicionados nas camadas do mapa como uma lista

Mapas – Parte 4

- Após isso, adicione o callback de receiver para as camadas do mapa

```
//adiciona esse callback de eventos ao mapa
MapEventsOverlay mapEventsOverlay = new MapEventsOverlay(mReceive);
mapa.getOverlays().add(0, mapEventsOverlay);
```

- Por fim, iremos pegar a localização escolhida pelo usuário quando ele clicar no botão
  - Para isso, basta utilizar aquela nossa variável criada anteriormente, a `localizacaoEscolhida`.

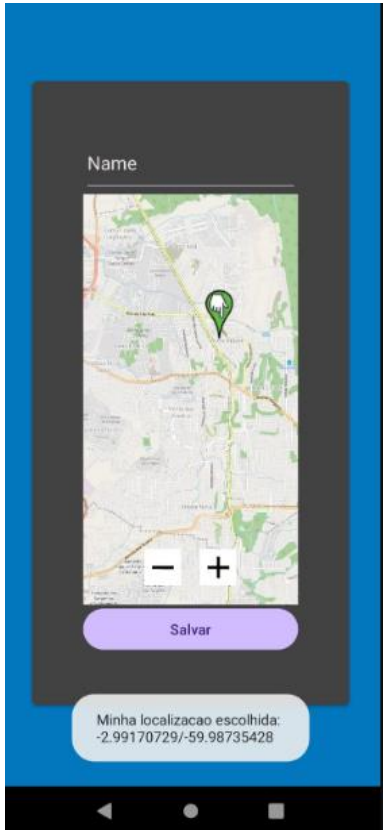
Mapas – Parte 4

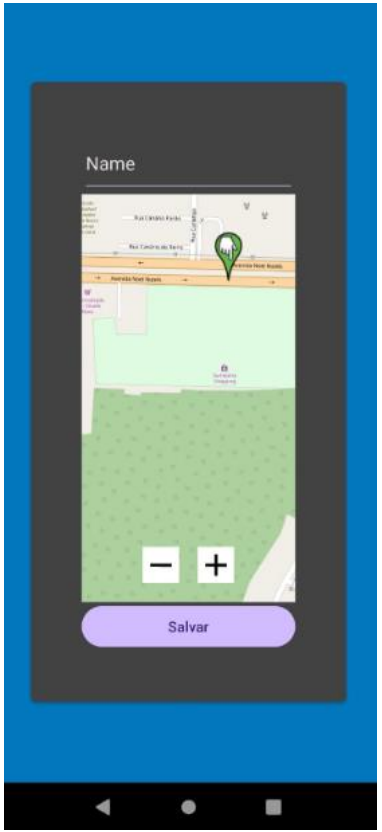
```
//eventos do botao salvar
salvarLocalizacao = (Button) findViewById(R.id.button);
salvarLocalizacao.setOnClickListener(new View.OnClickListener() {
   @Override
   public void onClick(View v) {
      if (localizacaoEscolhida != null) {
         double latitude = localizacaoEscolhida.getLatitude();
         double longitude = localizacaoEscolhida.getLongitude();

         Toast toast = Toast.makeText(
               getApplicationContext(),
               "Minha localizacao escolhida:\n"+
                     latitude + "/" + longitude,
               Toast.LENGTH_LONG
         );
         toast.show();
      }
   }
});
```

Mapas – Parte 4

- Resultado final:





Referências

- Tutorial da própria lib: Tutorial_0 · MKergall/osmbonuspack Wiki · GitHub
- OSMNavigator: OSMNavigator · MKergall/osmbonuspack Wiki · GitHub
- LocationProviders: LocationProvider | Android Developers

- Nosso código está no git: GitHub - diogosm/android_class_IArTES: Código das aulas do módulo de Android no IArTES