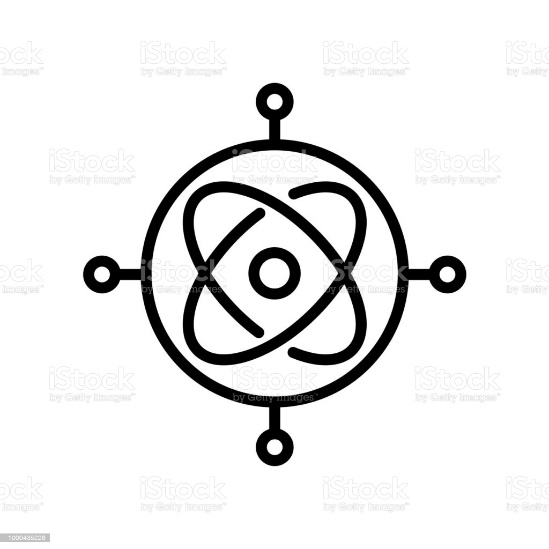
Mariana Santos Fernandes  *N 28*

**Recursos dos Dispositivos**



*Giroscópio*

*Giroscópio precisa primeiramente instanciando o objeto após a importação.*

//Instanciando o objetos SensorManager

private SensorManager sensorManager;

private Sensor sensor;

//Colocando dentro destes objetos os atributos de funcionalidade

sensorManager = (SensorManager) getSystemService(Context.SENSOR\_SERVICE);

//instalando a função do sensor físico dentro do objeto

sensor = sensorManager.getDefaultSensor(Sensor.TYPE\_GAME\_ROTATION\_VECTOR);

*Depois disso calculamos o ângulo na qual esta disposto o sensor giroscópio para interpreta-lo em uma determinada direção.*

// Rotação do dispositivo depende do acelerômetro e do magnetômetro

final float[] rotationMatrix = new float[9];

SensorManager.getRotationMatrix(rotationMatrix, null,

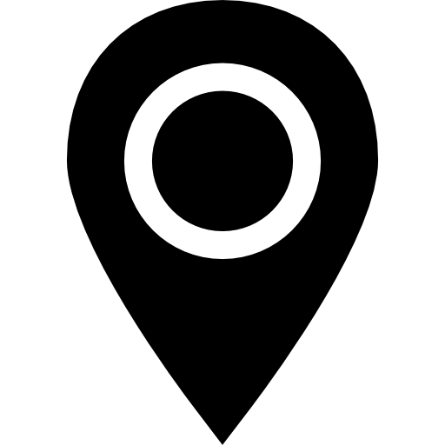
accelerometerReading, magnetometerReading);

// Apos a coleta dos dados podemos retornara dados para 3 angulos diferente que podemos colocar dentro do objeto

final float[] orientationAngles = new float[3];

SensorManager.getOrientation(rotationMatrix, orientationAngles);

*E com esses dados podemos tratá-los e utilizarmos a nosso favor em algumas de nossas aplicações.*



*Geolocalização*

*Geolocalização podemos utilizar para pegarmos a localização do aparelho, e para conseguirmos esse tipo de informação, antes pedimos a permissão do usuário para assim executarmos, pois se não fosse feito tal, quebraríamos normas restritivas, de segurança e de privacidade*

//Gradle//

//adicionando dependencia

dependencies {

compile 'com.google.android.gms:play-services:8.1.0'

}

//AndroidManifest.xml//

//fazendo o pedido de permissao

<uses-permission android:name="android.permission.ACCESS\_FINE\_LOCATION" />

*Primeiro Instanciando classes GoogleApiClient, porque será através dela que iremos acessar os serviços que existem no Google Services, como o Location Service.*

googleApiClient = new GoogleApiClient.Builder(this)

.addConnectionCallbacks(this) //Be aware of state of the connection

.addOnConnectionFailedListener(this) //Be aware of failures

.build();

//Tentando conexão com o Google API. Se a tentativa for bem sucessidade, o método onConnected() será chamado, senão, o método onConnectionFailed() será chamado.

googleApiClient.connect();

@Override

protected void onStop() {

super.onStop();

pararConexaoComGoogleApi();

}

public void pararConexaoComGoogleApi() {

//Verificando se está conectado para então cancelar a conexão!

if (googleApiClient.isConnected()) {

googleApiClient.disconnect();

}

}

/\*\*

\* Depois que o método connect() for chamado, esse método será chamado de forma assíncrona caso a conexão seja bem sucedida.

@Override

public void onConnected(Bundle bundle) {

//Conexão com o serviços do Google Service API foi estabelecida!

}

/\*\*

\* Esse método é chamado quando o client está temporariamente desconectado. Isso pode acontecer quando houve uma falha ou problema com o serviço que faça com que o client seja desligado.

\* Nesse estado, todas as requisições e listeners são cancelados.

\* Não se preocupe em tentar reestabelecer a conexão, pois isso acontecerá automaticamente.

\* As aplicações devem desabilitar recursos visuais que estejam relacionados com o uso dos serviços e habilitá-los novamente quando o método onConnected() for chamado, indicando reestabelecimento da conexão.

\*/

@Override

public void onConnectionSuspended(int i) {

// Aguardando o GoogleApiClient reestabelecer a conexão.

}

/\*\*

\* Método chamado quando um erro de conexão acontece e não é possível acessar os serviços da Google Service.

\*

\* @param connectionResult

\*/

@Override

public void onConnectionFailed(ConnectionResult connectionResult) {

//A conexão com o Google API falhou!

pararConexaoComGoogleApi();

}

}

*Vamos Adicionar o Location Service, usaremos uma referencia para a chamada e instanciaremos para após pegarmos a ultima localização registrada.*

//Instanciando

googleApiClient = new GoogleApiClient.Builder(this)

.addConnectionCallbacks(this)

.addOnConnectionFailedListener(this)

.addApi(LocationServices.API)

.build();

//jogamos dentro de uma variável “lastLocation” e usando o método getLastLocation temos acesso a essa localização registrada

@Override

public void onConnected(Bundle bundle) {

//Google API connection has been done successfully

Location lastLocation =

LocationServices.FusedLocationApi.getLastLocation(googleApiClient);

}

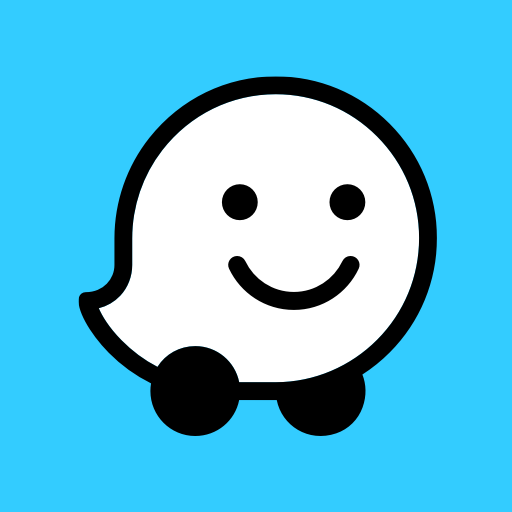
//e se caso seja mais apropriado posso pegar longitude e latitude da localização atual com estes métodos

Location.getLatitude()

Location.getLongitude()

***Apps que utilizam recursos***

***Relacionados a Componente Curricular***



**Waze**

Um aplicativo de trânsito e navegação que se baseia na comunidade, o Waze foi criado como uma ferramenta de navegação social para carros particulares.

Ele utiliza os 2 recursos explicados acima, GeoLocalização que faz o rastreamento de um dispositivo por meio de uma conexão remota. Essa conectividade varia entre três métodos: GPS (sistema de posicionamento geográfico), GSM (sistema global para comunicações móveis) e wireless (via Wi-Fi, por exemplo). E o Giroscopio no waze é utilizado para selecionar a direção na qual o celular esta virado para guiar no trajeto. Waze se baseia na Geografia.



**Duolingo**

É o maior site para aprendizagem gratuita de idiomas do mundo. Pela plataforma, os falantes de português brasileiro podem aprender três idiomas: espanhol, francês e inglês. Este Aplicativo se relaciona com a matéria de Língua Estrangeira Inglesa auxiliando ao aprendizado.