|  |  |
| --- | --- |
| Brasão da UFC |  |

Disciplina de Computação Móvel e Ubíqua – Aula Prática da Awareness API

Professor Dr Windson Viana de Carvalho

Parte 1 – Preparação

Como o Awareness faz parte do Google Play Services, ele exige algumas preparações iniciais que devem ser realizadas no ambiente de desenvolvimento relacionados à geração de chaves (API Keys) e também o download dos módulos adicionais do Google Play Services

- Cheque no link a seguir se a sua máquina já possui o SDK configurado e crie um API Key para seu projeto

<https://developers.google.com/awareness/overview>

Parte 2 – Aprendendo sobre a API

1) Sobre a API Awareness responda:

a) Qual é a diferença de um Fence para um Snapshot?

b) Cite quatro Snapshots disponíveis na API

c) Explique os passos para criação de um Fence com atributo único e um Fence com múltiplos atributos

d) Liste 4 permissões necessários para o uso da API

Parte 3 - Implementação

O primeiro passo do Awareness consiste na conexão com API ainda dentro do onCreate() da Activity com o código semelhante ao abaixo

**client** = **new** GoogleApiClient.Builder(MainActivity.**this**)  
 .addApi(Awareness.***API***)  
 .build();  
**client**.connect();

Em seguida, é possível registrar Fences e Snapshot, vale lembrar que mesmo o Snapshot exige um mecanismo de ResultCallback/Listener para receber os resultados dos métodos. Por exemplo, o método a seguir exibe um snapshot do status do headphone

**public void** getHeadphoneState(**final** HeadphoneListener listener) {  
 Awareness.***SnapshotApi***.getHeadphoneState(**client**)  
 .setResultCallback(**new** ResultCallback<HeadphoneStateResult>() {  
 @Override  
 **public void** onResult(@NonNull HeadphoneStateResult headphoneStateResult) {  
 **if** (!headphoneStateResult.getStatus().isSuccess()) {  
 Log.*e*(***TAG***, **"Erro ao identificar conexão com headphone"**);  
 **return**;  
 }  
 HeadphoneState state = headphoneStateResult.getHeadphoneState();  
 **if** (state.getState() == HeadphoneState.***PLUGGED\_IN***) {  
 Log.*i*(***TAG***, **"Headphones plugados"**);  
 listener.completion(**true**);  
 } **else** {  
 Log.*i*(***TAG***, **"Headphones desplugados"**);  
 listener.completion(**false**);  
 }  
 }  
 });  
}

Crie um listener HeadPhoneListener

**public interface** HeadphoneListener {  
 **void** completion(Boolean state);  
}

Para utilizar o método você deve passar um HeadPhoneListener como parâmetro. Veja a seguir, um exemplo usando *um inner class*

getHeadphoneState(**new** HeadphoneListener() {  
 @Override  
 **public void** completion(Boolean state) {  
 **if** (state){  
 Log.*i*(***TAG***, **"Headphones plugados"**);  
 **headphoneText**.setText(**"Plugados"**);  
 } **else** {  
 Log.*i*(***TAG***, **"Headphones desplugados"**);  
 **headphoneText**.setText(**"Desplugados"**);  
 }  
 }  
});

1. Crie uma aplicação que exibe após um clique em um botão se o headphone está ou não plugado no aparelho
2. Crie um método semelhante para checar a localização do usuário por meio Places usando a API do Awareness

Dicas aqui: <https://developers.google.com/awareness/android-api/snapshot-get-data#get_nearby_places>

FENCES

Para usar FENCES três passos são necessários: criar uma FENCE, registrá-la e definir um BroadcastReceiver para receber o resultado como o trigger do FENCE obtiver sucesso.

A seguir um exemplo para a detecção de mudança de status do HeadPhone (https://developers.google.com/awareness/android-api/fence-api-overview)

// Declare variables for pending intent and fence receiver.  
private PendingIntent myPendingIntent;  
private MyFenceReceiver myFenceReceiver;  
  
// Initialize myPendingIntent and fence receiver in onCreate().  
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    ...  
    Intent intent = new Intent(FENCE\_RECEIVER\_ACTION);  
    myPendingIntent = PendingIntent.getBroadcast(this, 0, intent, 0);  
    MyFenceReceiver = new myFenceReceiver();  
    registerReceiver(myFenceReceiver, new IntentFilter(FENCE\_RECEIVER\_ACTION));  
    ...  
}  
  
// Create a fence.  
AwarenessFence headphoneFence = HeadphoneFence.during(HeadphoneState.PLUGGED\_IN);  
  
// Register the fence to receive callbacks.  
// The fence key uniquely identifies the fence.  
Awareness.FenceApi.updateFences(  
        mGoogleApiClient,  
        new FenceUpdateRequest.Builder()  
            .addFence("headphoneFenceKey", headphoneFence, myPendingIntent)  
            .build())  
        .setResultCallback(new ResultCallback<Status>() {  
            @Override  
            public void onResult(@NonNull Status status) {  
                if (status.isSuccess()) {  
                    Log.i(TAG, "Fence was successfully registered.");  
                } else {  
                    Log.e(TAG, "Fence could not be registered: " + status);  
                }  
            }  
        });  
  
// Handle the callback on the Intent.  
public class MyFenceReceiver extends BroadcastReceiver {  
    @Override  
    public void onReceive(Context context, Intent intent) {  
        FenceState fenceState = FenceState.extract(intent);  
  
        if (TextUtils.equals(fenceState.getFenceKey(), "headphoneFenceKey")) {  
            switch(fenceState.getCurrentState()) {  
                case FenceState.TRUE:  
                    Log.i(TAG, "Headphones are plugged in.");  
                    break;  
                case FenceState.FALSE:  
                    Log.i(TAG, "Headphones are NOT plugged in.");  
                    break;  
                case FenceState.UNKNOWN:  
                    Log.i(TAG, "The headphone fence is in an unknown state.");  
                    break;  
            }  
        }  
    }  
}

1. Modifique o código anterior para que a se plugar o HeadPhone o celular do usuário vibre
2. Crie um Fence para vibrar o celular quando o usuário estiver andando
3. Crie uma agregação de Fence para caso o usuário esteja andando e o headphone ser plugado a “lanterna” do celular é ativada