UFC

DEPARTAMENTO DE COMPUTAÇÃO

MDCC

GREat

Awareness API

Professores Windson Viana e Fernando Trinta

Disciplina de Computação Móvel e Ubíqua

Curso de Sistemas e Mídias Digitais

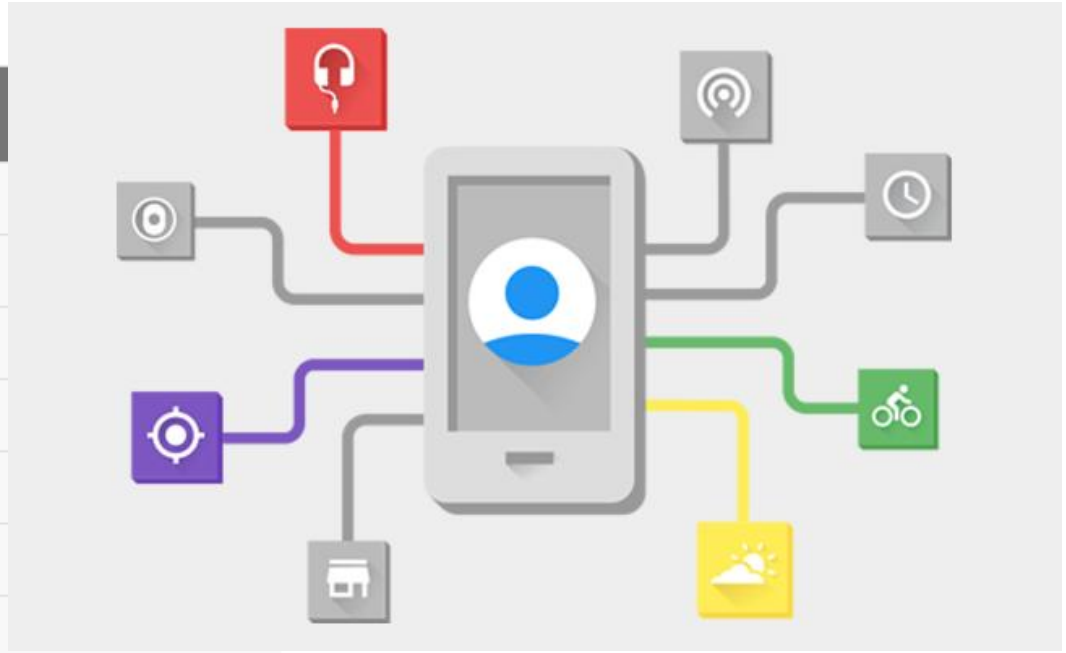
Awareness – Google API

Context types

Context is at the heart of the Awareness API. Contextual data includes sensor-derived data such as location (lat/lng), place (home, work, coffee shop), and activity (walking, driving). These basic signals can be combined to extrapolate the user's situation in more specific detail.

The following table describes the basic context types currently offered by the Awareness API:

Context type	Example
Time	Current local time
Location	Latitude and longitude
Place	Place, including place type
Activity	Detected user activity (walking, running, biking)
Beacons	Nearby beacons (including namespace, type, and content)
Headphones	Are headphones plugged in?
Weather	Current weather conditions



Google Awareness

Facilidade de implementação

Você só precisa adicionar uma única API ao seu aplicativo, o que simplifica bastante a integração e melhora sua produtividade

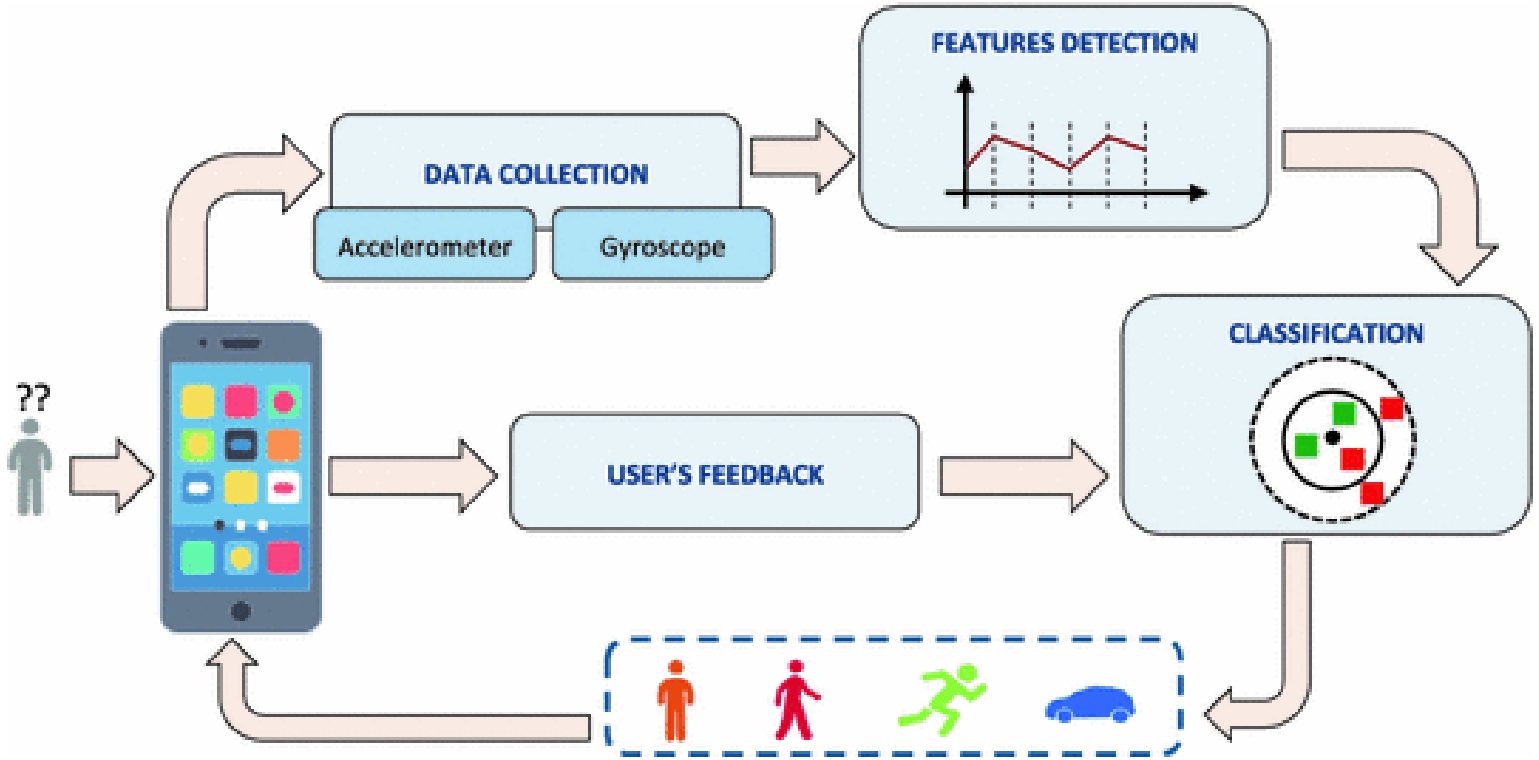
Integridade e saúde do sistema

A API do Awareness gerencia automaticamente seu impacto na vida útil da bateria e no uso de dados

Google Awareness

Dados de contexto de alto nível:

Os sinais brutos são processados para melhorar a qualidade da informação. Ex: Atividade do usuário com um alto nível de precisão



Google Awareness

Usa o conceito de Fences e Snapshots para definição de regras sobre as informações contextuais

- Métodos de Callback
- Registro para notificação

Possibilidade de agregar e combinar mais de um ContextType

Iniciando a conexão

Uso do Google Play Services - necessidade de uma API Key

```
dependencies {
    implementation 'com.google.gms:play-services-awareness:16.0.0'
    implementation 'com.google.gms:play-services-places:16.0.0'
}
```

```
android.content.Context context;
GoogleApiClient client = new GoogleApiClient.Builder(context)
        .addApi(Awareness.API)
        .build();
client.connect();
```

Snapshot API – "Get the Current Context"

Ver a última atividade detectada

```
Awareness.SnapshotApi.getDetectedActivity(mGoogleApiClient)
        .setResultCallback(new ResultCallback<DetectedActivityResult>() {
            @Override
            public void onResult(@NonNull DetectedActivityResult detectedActivityResult) {
                if (!detectedActivityResult.getStatus().isSuccess()) {
                    Log.e(TAG, "Could not get the current activity.");
                    return;
                }
                ActivityRecognitionResult ar = detectedActivityResult.getActivityRecognitionResult();
                DetectedActivity probableActivity = ar.getMostProbableActivity();
                Log.i(TAG, probableActivity.toString());
            }
        });
```

Snapshot API – "Get the Current Context"

Ver a última atividade detectada

`ActivityRecognitionApi` for details on how to obtain a `DetectedActivity`.

Constant Summary

int	IN_VEHICLE	The device is in a vehicle, such as a car.
int	ON_BICYCLE	The device is on a bicycle.
int	ON_FOOT	The device is on a user who is walking or running.
int	RUNNING	The device is on a user who is running.
int	STILL	The device is still (not moving).
int	TILTING	The device angle relative to gravity changed significantly.
int	UNKNOWN	Unable to detect the current activity.
int	WALKING	The device is on a user who is walking.

Permissões

Snapshot API

Method	Required Android Permission
getDetectedActivity()	com.google.android.gms.permission.ACTIVITY_RECOGNITION
getBeaconState()	android.permission.ACCESS_FINE_LOCATION
getHeadphoneState()	none
getLocation()	android.permission.ACCESS_FINE_LOCATION
getPlaces()	android.permission.ACCESS_FINE_LOCATION
getWeather()	android.permission.ACCESS_FINE_LOCATION

Fence Example

```
// 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);
                }
            }
        });
```

Fence Example

```
// 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(), "headphoneFence")) {
            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;
            }
        }
    }
}
```

Permissões

Fence API

Fence API fence type	Required Android permission
DetectedActivityFence	com.google.android.gms.permission.ACTIVITY_RECOGNITION
BeaconFence	android.permission.ACCESS_FINE_LOCATION
HeadphoneFence	none
LocationFence	android.permission.ACCESS_FINE_LOCATION
TimeFence	none

Agregação

Create a combination fence

Combination fences combine multiple primitive fence types by using boolean operators. The following example shows creating a combination fence that activates when the user is walking AND the headphones are plugged in:

```
// Create the primitive fences.
AwarenessFence walkingFence = DetectedActivityFence.during(DetectedActivityFence.WALKING);
AwarenessFence headphoneFence = HeadphoneFence.during(HeadphoneState.PLUGGED_IN);

// Create a combination fence to AND primitive fences.
AwarenessFence walkingWithHeadphones = AwarenessFence.and(
      walkingFence, headphoneFence
);
```

Nested trees of `AND`, `OR` and `NOT` are valid, so that any boolean combination of fences is possible. The following example shows a fence that is triggered when a user moves more than 100 meters from the current location, OR over an hour has elapsed since the current time.

API Android 10

Notice: The Places and Weather contextual signals (exposed via the getPlaces(), and getWeather() methods), are deprecated as of August 7, 2019. The Places contextual signal will be turned off by October 30, 2019. New implementations must use **the Places SDK** for Android instead of the Places contextual signal.

The Weather contextual signal will be turned off by January 31, 2020. Continued use of this signal is restricted to customers with existing implementations, through January 31, 2020. **Google does not offer** alternative functionality for the Weather contextual signal.

Hands on na próxima aula