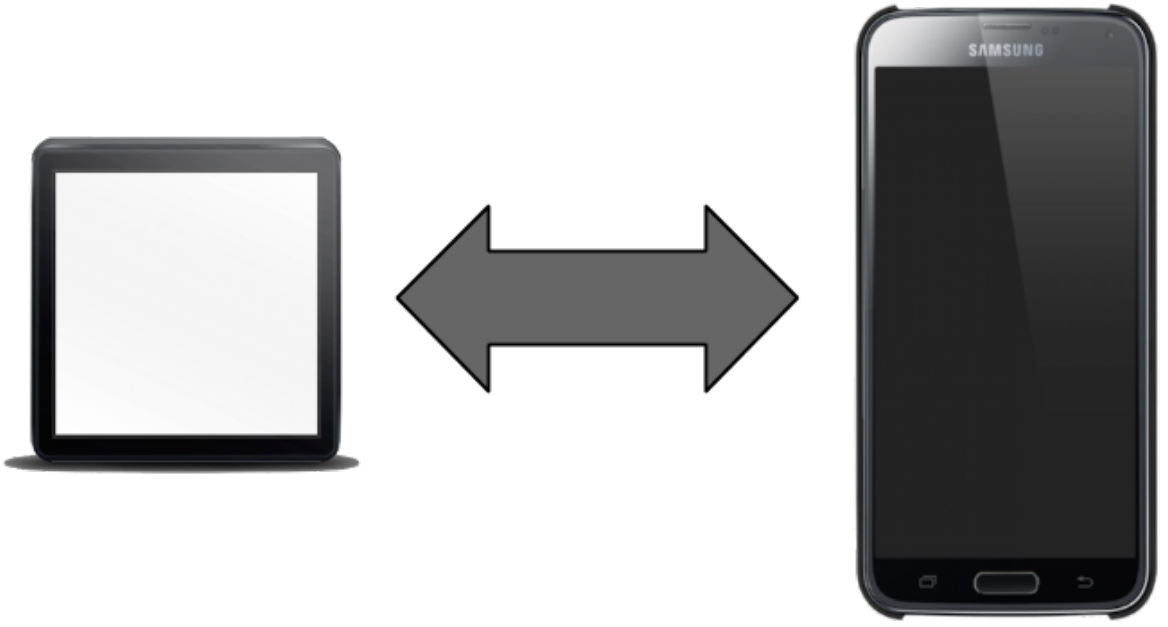
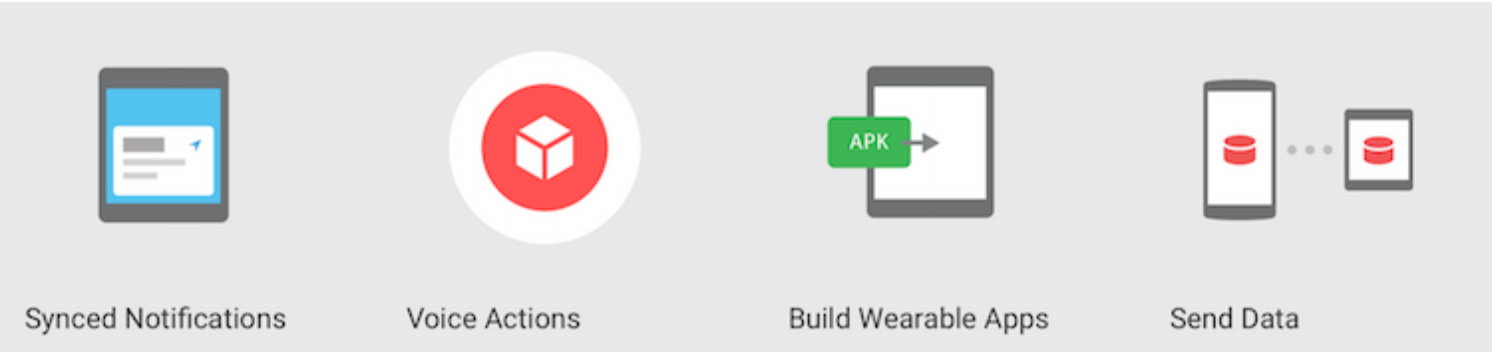
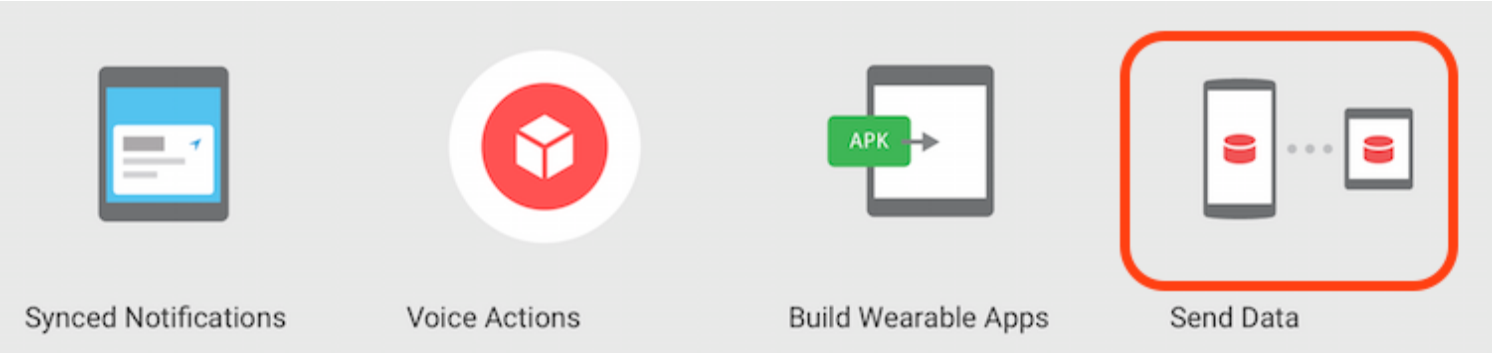
Comunicación Wear - Phone


SAMSUNG

Android Wear nos permite comunicarnos con nuestro dispositivo host(en este caso se trataría de nuestro smartphone); por medio de la API que nos provee Google.


APK
Synced Notifications
Voice Actions
Build Wearable Apps
Send Data


APK
Synced Notifications
Voice Actions
Build Wearable Apps
Send Data

Data API

Data API es una de las 4 API que nos provee Google para interactuar con nuestro SmartWatch. En este caso se trata de un canal de comunicación con nuestro SmartPhone(o cualquier otro dispositivo conectado disponible).

Este API nos permite comunicarnos con los siguientes Objetos:

- **Data Item**: Permite guardar datos genéricos para ser almacenados el nuestro SmartPhone.
- **Asset**: Se trata de un objeto que podemos guardar; como por ejemplo una imagen.
- **Message**: Se trata de un mensaje que podemos utilizar para hacer Llamadas a procedimientos remotos.

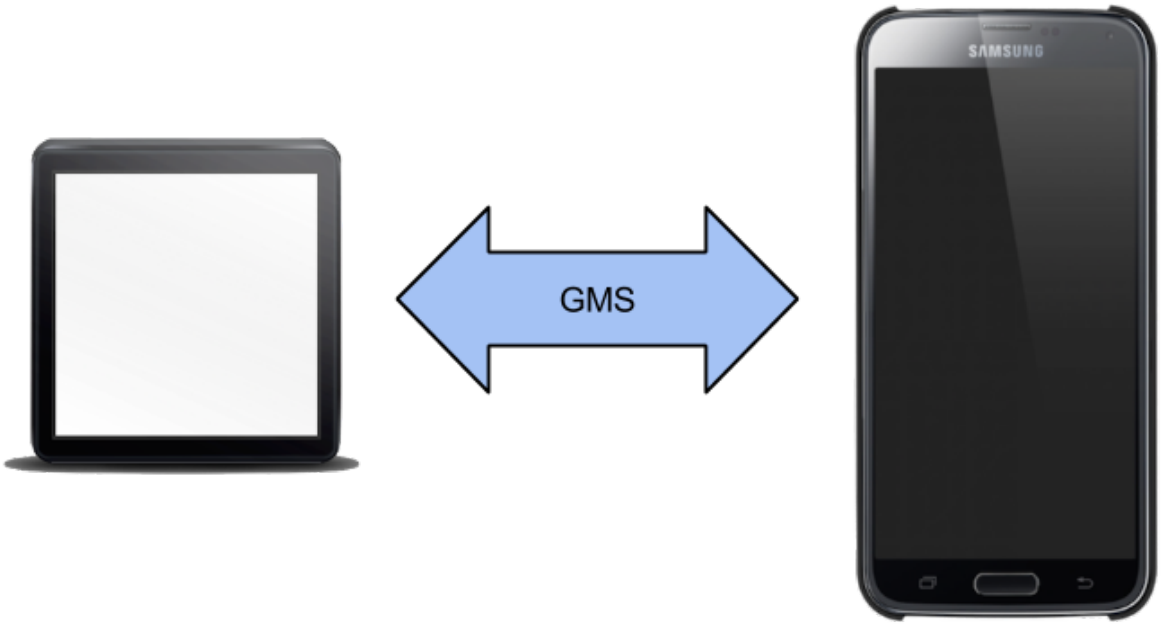
Vale, pero ¿Cómo me comunico con mi dispositivo?

GCMS

[Google Cloud Messages Services]

Google Clod Messages Services es una API que nos permite comunicarnos tanto con nuestros dispositivos, como con los Servicios de Google(Youtube, Maps, Calendar,etc...).

Gracias a esta API podemos comunicarnos con nuestro Dispositivo por medio de la Api especifica para ello la `wearable.API`



Implementación de una comunicación con nuestro dispositivo

[knigth Rider]

Proyecto

Crearemos un proyecto con 2 módulos; uno para móvil y otro para nuestro dispositivo Wear:

- Módulo *mobile*: versión mínima 2.3.3
- Módulo *wear*: versión mínima 4.4.4.

Dependencias

Necesitaremos las siguientes librerías que añadiremos a nuestro archivo gradle en nuestros proyectos:

```
compile 'com.google.android.support:wearable:1.0.0'
compile 'com.google.android.gms:play-services-wearable:6.5.87'
```

Wear

Nos centraremos en el dispositivo en primer lugar. Para ello, crearemos una actividad que nos permitirá comunicarnos con el dispositivo.

También crearemos un cliente para ponder conectarnos a la API. por lo que nos crearemos un objeto `GoogleApiClient`.

```
this.apiclient = new GoogleApiClient.Builder(this).addConnectionCallbacks(this)
                .addApi(Wearable.API)
                .build();
this.apiclient.connect();
```

En este caso hemos añadido la funcionalidad para los Wearable con el método `addApi()`.

Wear

Una vez conectados, podemos mandar un mensaje al nodo conectado(en este caso a nuestro SmartPhone).

```
NodeApi.GetConnectedNodesResult rawNodes =
            Wearable.NodeApi.getConnectedNodes(apiClient).await();

Wearable.MessageApi.sendMessage(apiClient,rawNodes.getNodes().get(0).getId(),path,null);
```

Nota: En este caso solo hay un nodo conectado pero puede darse el caso de que haya más de uno.

Mobile

Una vez que tenemos lista la comunicación de nuestro SmartWatch, tenemos que pasar a nuestro SmartPhone. En primer lugar, necesitaremos poder recibir los mensajes y conexiones desde nuestro SmartWatch. Para ello, usamos el servicio `WearableListenerService`.

Este servicio, será el encargado de recibir las conexiones desde nuestro SmartpHone y de recibir los mensajes que enviemos.

```
public class KnigthRiderService extends WearableListenerService {

    public APIController controller;
    public GoogleApiClient gapiClient;

    public KnigthRiderService() {
        this.controller= new APIController(this);


    }

    @Override
    public void onMessageReceived(MessageEvent messageEvent) {
       String path=messageEvent.getPath();
        Toast.makeText(this,"Mensaje Recibido",Toast.LENGTH_SHORT).show();
        this.controller.ControlKIT(path);

    }
```

Con todo esto ya tenemos lista la comunicación de nuestro Reloj con Nuestro Smartphone y podemos mandar distintos datos a este. Seguidamente vamos a pasar a mostrar el código de la aplicación.

Referencias:

- Android Wear: https://developer.android.com/training/wearables/data-layer/index.html
- Repositorio con ejemplos: https://github.com/zerasul/Hellowear