Programando las Google Glass

[El Weareable siempre visible]

En Google IO del 2012, cuando aún había señales de *relojes listos* ni nada de eso, el buscador presentó sus *Google Glass*, y desde entonces somos muchos los **frikis** que las hemos querido, sin embargo, es un dispositivo **difícil de conseguir** ya que no está a la venta de forma pública.

Esto para la mayoría de los mortales no suponía un trauma:

La Eurocopa en las redes sociales

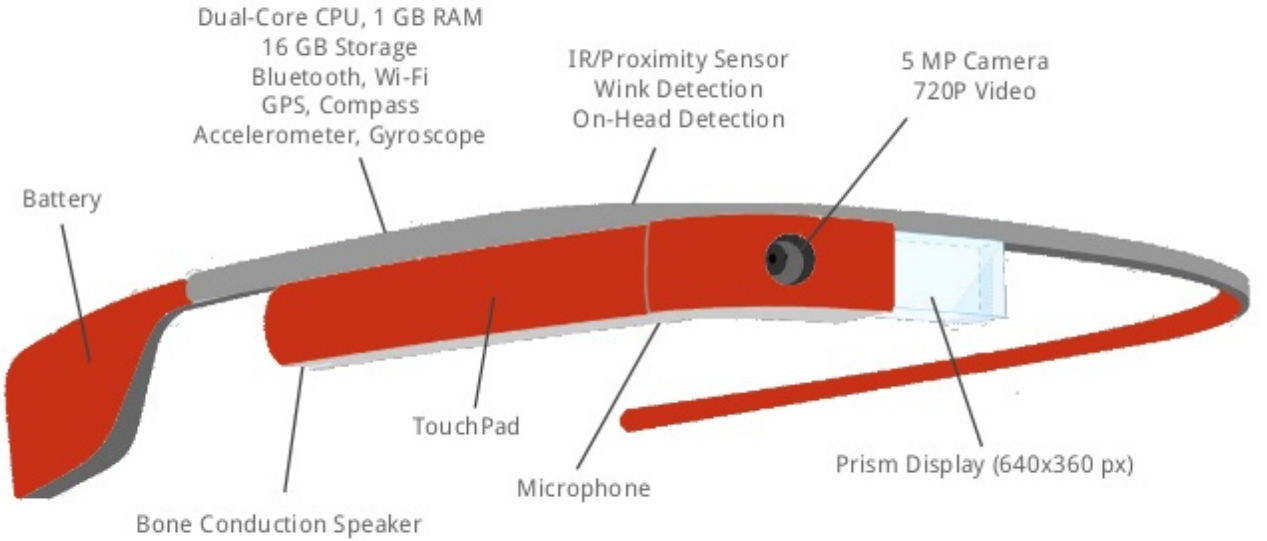
twitter	Estoy viendo la eurocopa
facebook	Me gusta la eurocopa
YouTube	Vídeo-resumen del partido de la eurocopa
foursquare	Aquí es donde estoy viendo la eurocopa
blintic	Os recomiendo ver este gol de la eurocopa
flickr	Foto artística de un gol de la eurocopa
badoo	Quieres ver la eurocopa en mi habitación?
tuenti	Golazoh shulo!!
LinkedIn	Soy experto en la eurocopa
Google+	Cuando salen las gafas de google?

Finalmente, a mediados de Abril 2013 salieron los primeros *explorers* en Estados Unidos y poco a poco más gente ha podido adquirir el dispositivo poder hacer lo que siempre han querido


HIS POWER LEVEL
IT'S OVER NINE THOUSAAAAAAAAAAND!

¿Qué tiene las Glass?

Google Glass Hardware Breakdown



¿Y como se programa el «juguete caro»?

Tarjetas, tarjetas, más tarjetas y algún paginador para las tarjetas...

¿Y esas tarjetas las hago...?

Para programar las *Glass* disponemos de dos APIS **muy diferentes**, por un lado tenemos *Mirror*, que fue la primera en salir, y por otro la *GDK*, la cual salió unos meses después y será en la que nos centraremos.

Mirror

[Insertando notificaciones en nuestro TimeLine]

La API *Mirror* es una **API Web**, la cual nos permite crear *servicios webs* con el cual podremos enviar información a directamente a las Glass. Su funcionamiento es realtivamente sencillo. Nuestro servicio web debía comunicarse con los servidores de Google, pedir persimo para acceder a nuesta cuenta y ya podría enviar «notificaciones» a nuestras gafas.

Esta API está bastante limitada y solo nos ofrece un sistema para mostrar información, pero nos permite crear rápidamente una App sencilla que supla algunas necesidades, como puede ser mostrar la cartelera de un cine.

GDK

[La *API Real* de las Google Glass]

Las *Glass* disponen de un Android 4.4 en su interior, y con él, toda la API de Android que ya conocemos. Tanto a nivel de *core* (servicios, bases de datos, ContentProviders...) como a nivel interfaz gráfico, sin embargo, dado el «diseño» de las *Glass* y la **ausencia de pantalÍa táctil**, estos elementos no suelen acomodarse del todo.

Por ello, surge un *paquete adicional* denominado **Glass Development Kit**, un nombre muy épico para un par de bibliotecas que podrían incluirse en el *Support Library* sin problema.

Principalmente, esta nueva API incorpora una serie de clases para la generación de las *Cards* de forma rápida y un control de gestos.

¿Cómo es una App para Glass?

[Organización y tipo de tarjetas]


JFK
Virgin Airlines
Flight 23
10:31 AM
On Time
80°
60°
30%
New York
72°
tomorrow
now
10:34
ok glass

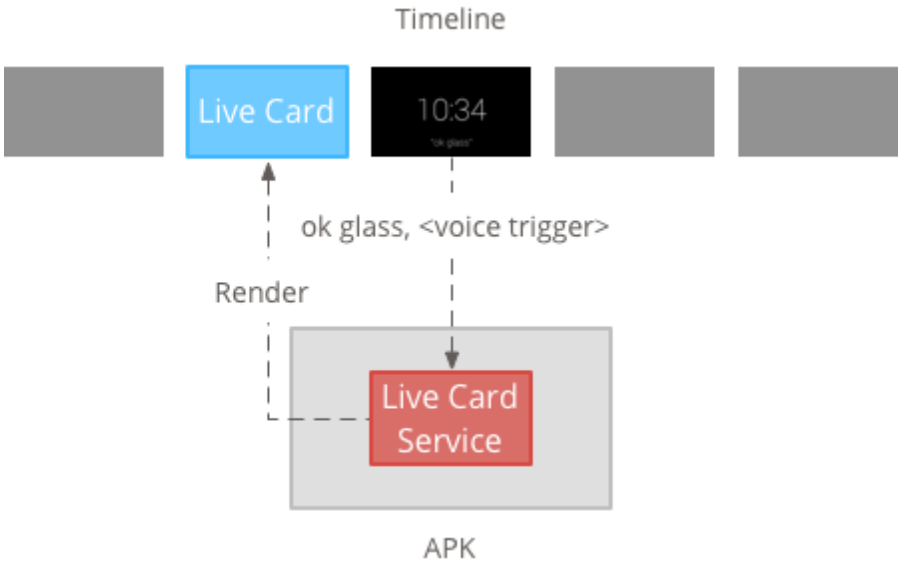
Tipos de Tarjetas

En *Google Glass* podemos distinguir dos tipos de aplicaciones, las **Live Card** y las **Immersion**.

Live Cards

Las Live Cards son aquellas que quedan en nuestro Time Line, siendo una versión vitaminada de las notificaciones de Android.

Un ejemplo claro de son las tarjetas de *Google Now* o las de *GMail*. Elementos que nos aportan una información en un determinado momento a modo de aviso o que podremos consultar rápidamente con unos toques a la patilla.



```
public class LiveCardService extends Service {
    private static final String TAG = "LiveCardDemo";
    private LiveCard mLiveCard;
    private RemoteViews mLiveCardView;

    public int onStartCommand(Intent intent, int flags, int startId) {
        if (mLiveCard != null)
            return START_STICKY;

        mLiveCard = new LiveCard(this, TAG); //Creamos la LC
        //Creamos el View asociado
        mLiveCardView = new RemoteViews(getPackageName(), R.layout.main_layout);
        //Le damos valores a los elementos
        mLiveCardView.setTextViewText(R.id.saludo,  "Hola Peña");
        mLiveCardView.setTextViewText(R.id.texto, "Android Mola"):
        //Insertamos los Views
        mLiveCard.setViews(mLiveCardView);
        //Creamos un Intent para lanzar un menú
        Intent menuIntent = new Intent(this, MenuActivity.class);
        menuIntent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK
                        | Intent.FLAG_ACTIVITY_CLEAR_TASK);

        mLiveCard.setAction(PendingIntent.getActivity(this, 0, menuIntent, 0));
        //Publicamos la tarjeta
        mLiveCard.publish(PublishMode.REVEAL);

        return START_STICKY;
    }
}
```

Immersion Apps

Las *Immersion* son las aplicaciones activas. Básicamente son las aplicaciones tradicionales que estamos acostumbrados a utilizar en Android, con sus Activitys y demás...

Así mismo, una aplicacion *immersiva* también podría tener una LiveCard enlazada.

Principalmente, estas apps se basan en una o varias Activitys, dónde se muestran un paginador de Cards.

Sí, los Fragments también funcionan

Si «jomíos sí», también pueden usarse los Fragments, no es un problema... Salvo los problemas propios de los Fragments.

Aunque funcionan, su uso suele ser poco frecuente dado que bastante complejo que un interfaz gráfica pueda utilizarse tanto en Android convencionales con pantalla táctilc como en Glass.

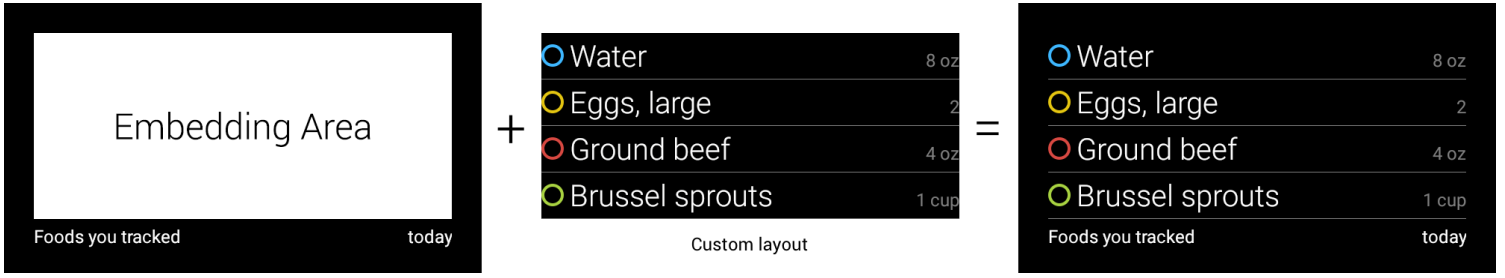
Creando las Cards

Para crear las Cards se suele utilizar el CardBuilder, permitiendo obtener tarjetas *estándar* de forma rápida y sencilla, aunque pueden generarse de forma convencional mediante un XML.


```
View view1 = new CardBuilder(context, CardBuilder.Layout.COLUMNS)
    .setText("This is the COLUMNS layout with dynamic text.")
    .setFootnote("This is the footnote")
    .setTimestamp("just now")
    .addImage(R.drawable.image1)
    .addImage(R.drawable.image2)
    .addImage(R.drawable.image3)
    .addImage(R.drawable.image4)
    .addImage(R.drawable.image5)
    .getView();
```


```
View view2 = new CardBuilder(context, CardBuilder.Layout.CAPTION)
    .setText("The caption layout with an icon.")
    .setFootnote("This is the footnote")
    .setTimestamp("just now")
    .addImage(R.drawable.beach)
    .setIcon(R.drawable.ic_avatar)
    .setAttributionIcon(R.drawable.ic_smile)
    .getView();
```


```
View view = new CardBuilder(context, CardBuilder.Layout.TITLE)
    .setText("TITLE Card")
    .setIcon(R.drawable.ic_phone)
    .addImage(R.drawable.beach)
    .getView();
```




```
View view = new CardBuilder(context, CardBuilder.Layout.EMBED_INSIDE)
    .setEmbeddedLayout(R.layout.food_table)
    .setFootnote("Foods you tracked")
    .setTimestamp("today")
    .getView();
TextView textView1 = (TextView) view.findViewById(R.id.text_view_1);
textView1.setText("Water");
```

CardScrollView

Este elemento nos permite crear *ViewPagers* de *Cards* para las Cards. Utiliza un *Adapters* como el hemos usando durante años en los *ListViews*.

El **CardBuilder** incopora ya un sistema de *ViewHolder* interno lo cual nos permite ahorrar algunas líneas y mantener un código más limpio.

```
public class MyAdapter extends CardScrollAdapter {
    private List<String> textos;
    public MyAdapter(List<String> textos) {
        this.textos = textos;
    }
    @Override
    public int getCount() {
        return textos.size();
    }
    @Override
    public Object getItem(int i) {
        return textos.get(i);
    }
    @Override
    public View getView(int i, View view, ViewGroup viewGroup) {
        CardBuilder builder = new CardBuilder(viewGroup.getContext(), CardBuilder.Layout.TEXT);
        builder.setText(textos.get(i));
        return builder.getView(view, viewGroup);
    }
    @Override
    public int getPosition(Object o) {
        return textos.indexOf(o);
    }
}
```


SanDisk
SDIN5C2-16G
CHINA
3113D040NN
ELPIDA
B8164R3PF-8D-
1306C914300
RNM
87
AV0


">9KI" AW 2013
SanDisk
SDIN5C2-16G