ANDROID VIII.

KIV/MKZ 2016

L. Pešička

OBSAH

- SMS
- Android wearable
- Android TV

BROADCAST RECEIVER

- přijímá záměry poslané sendBroadcast()
- přijímač záměru definovaný jako třída, která implementuje rozhraní **BroadcastReceiver**
- metoda onReceive()
- deklarace BR v manifestu:

<receiver android:name=“.MujBRClass”/>

je pouze po dobu provádění onReceive()

SMS zprávy

- třída android.telephony.SmsManager
- pro získání instance – getDefault()
- **sendTextMessage()**
  - cílová adresa – telefonní číslo
  - adresa SMSC centra, null – ze SIM karty
  - text zprávy ☺
  - info o odeslání zprávy
    - instance třídy PendingIntent, lze i null
  - info o doručení zprávy
    - instance třídy PendingIntent, lze i null

SMS PŘÍKLAD

```
void sendMessage(String prijemce,String zprava) {
  SmsManager sm = SmsManager.getDefault();
  sm.sendTextMessage(prijemce,null,zprava,null,null);
 }
```

zavoláme: sendMessage(”5556”, ”pokusna zprava”)

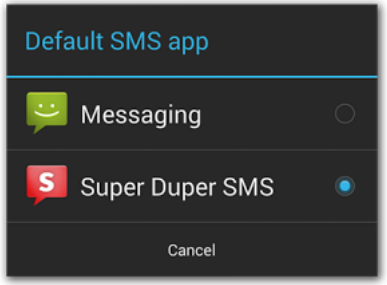
můžeme spustit dva emulátory a posílat sms zprávu z jednoho na druhý

komplexní příklad – doporučuji projít:
http://mobiforge.com/developing/story/sms-messaging-android

SMS - PRÁVA V MANIFESTU

```
<uses-permission
android:name="android.permission.SEND_SMS"/>

<uses-permission
android:name="android.permission.RECEIVE_SMS"/>
```

DEFAULTNÍ SMS APLIKACE (OD ANDROID 4.4)



- Vybraná uživatelem v nastavení systému
- Jen ona může zapisovat do SMS Provideru
  - Tabulky v Telephony Class
- Jen ona přijme SMS_DELIVER_ACTION

- Musí splňovat určitá pravidla (obsluha intentů, viz SDK)

NON DEFAULTNÍ SMS APLIKACE

- Může posílat SMS zprávy přes SmsManagera
- Aplikace co nejsou defaultní mohou jen číst ze SMS providera, můžou být notifikovány, když přijde SMS
- Systém automaticky zapíše odeslané SMS do SMS providera (defaultní je za své zápisy sama odpovědná)

NASTAVENÍ DEFAULTNÍ APLIKACE

1. Query the current default SMS app's package name and save it.

```
String defaultSmsApp = Telephony.Sms.getDefaultSmsPackage(context);
```

2. Request the user change the default SMS app to your app in order to restore SMS messages (you must be the default SMS app in order to write to the SMS Provider).

```
Intent intent = new Intent(context, Sms.Intents.ACTION_CHANGE_DEFAULT);
intent.putExtra(Sms.Intents.EXTRA_PACKAGE_NAME, context.getPackageName());
startActivity(intent);
```

3. When you finish restoring all SMS messages, request the user to change the default SMS app back to the previously selected app (saved during step 1).

```
Intent intent = new Intent(context, Sms.Intents.ACTION_CHANGE_DEFAULT);
intent.putExtra(Sms.Intents.EXTRA_PACKAGE_NAME, defaultSmsApp);
startActivity(intent);
```

http://android-developers.blogspot.cz/2013/10/getting-your-sms-apps-ready-for-kitkat.html

SMS - VYUŽITÍ VESTAVĚNÉ APLIKACE

```
public void Posli_SMS_Intentem (View v) {
    Uri smsUri = Uri.parse("tel:123456");
    Intent intent = new Intent(Intent.ACTION_VIEW, smsUri);
    intent.putExtra("sms_body", "sms text");
    intent.setType("vnd.android-dir/mms-sms");
    startActivity(intent);


}
```

TELEFONNÍ SLUŽBY

- Třída TelephonyManager
- Context.getSystemService (Context.TELEPHONY_SERVICE)
- Ochrana přístupu pomocí práv
- <uses-permission android:name="android.permission.READ_PHONE_STATE"/>

TELEFONNÍ SLUŽBY

- třída TelephonyManager
- getCallState()
  - CALL_STATE_IDLE
  - CALL_STATE_RINGING
  - CALL_STATE_OFFHOOK (probíhá hovor)
- getSubscriberId()
  - zjistí číslo SIM karty (IMSI)
- getPhoneType()
  - zjistí typ telefonu (GSM)
- getNetworkType()
  - typ připojení (GPRS, EDGE)

TELEFONNÍ SLUŽBY

```
TelephonyManager tm = (TelephonyManager)getSystemService(TELEPHONY_SERVICE);
tm.listen(mPhoneListener, PhoneStateListener.LISTEN_CALL_STATE);

// somewhere else
private PhoneStateListener mPhoneListener = new PhoneStateListener() {
    public void onCallStateChanged(int state, String incomingNumber) {
        try {
            switch (state) {
            case TelephonyManager.CALL_STATE_RINGING:
                // do something...
                break;

            case TelephonyManager.CALL_STATE_OFFHOOK:
                // do something...
                break;

            case TelephonyManager.CALL_STATE_IDLE:
                // do something...
                break;
            default:
                Log.d(TAG, "Unknown phone state=" + state);
            }
        } catch (RemoteException e) {}
    }
};
```

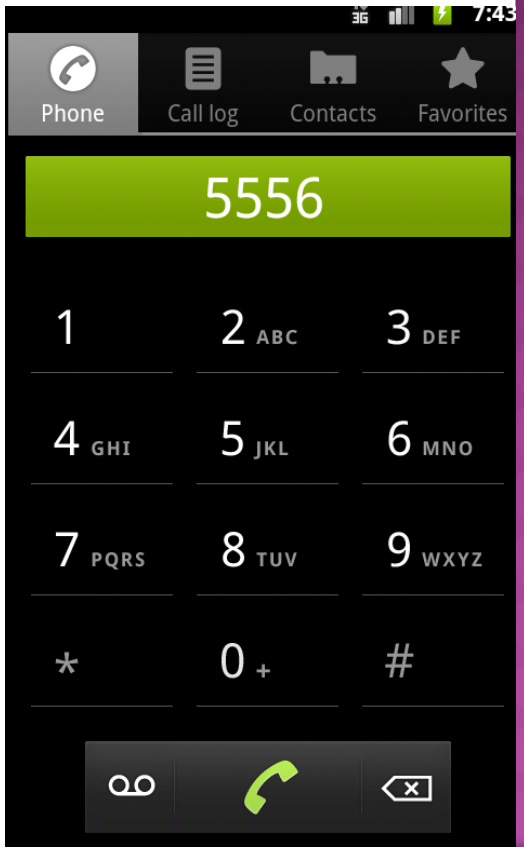
```
<uses-permission android:name="android.permission.READ_PHONE_STATE"/>
```

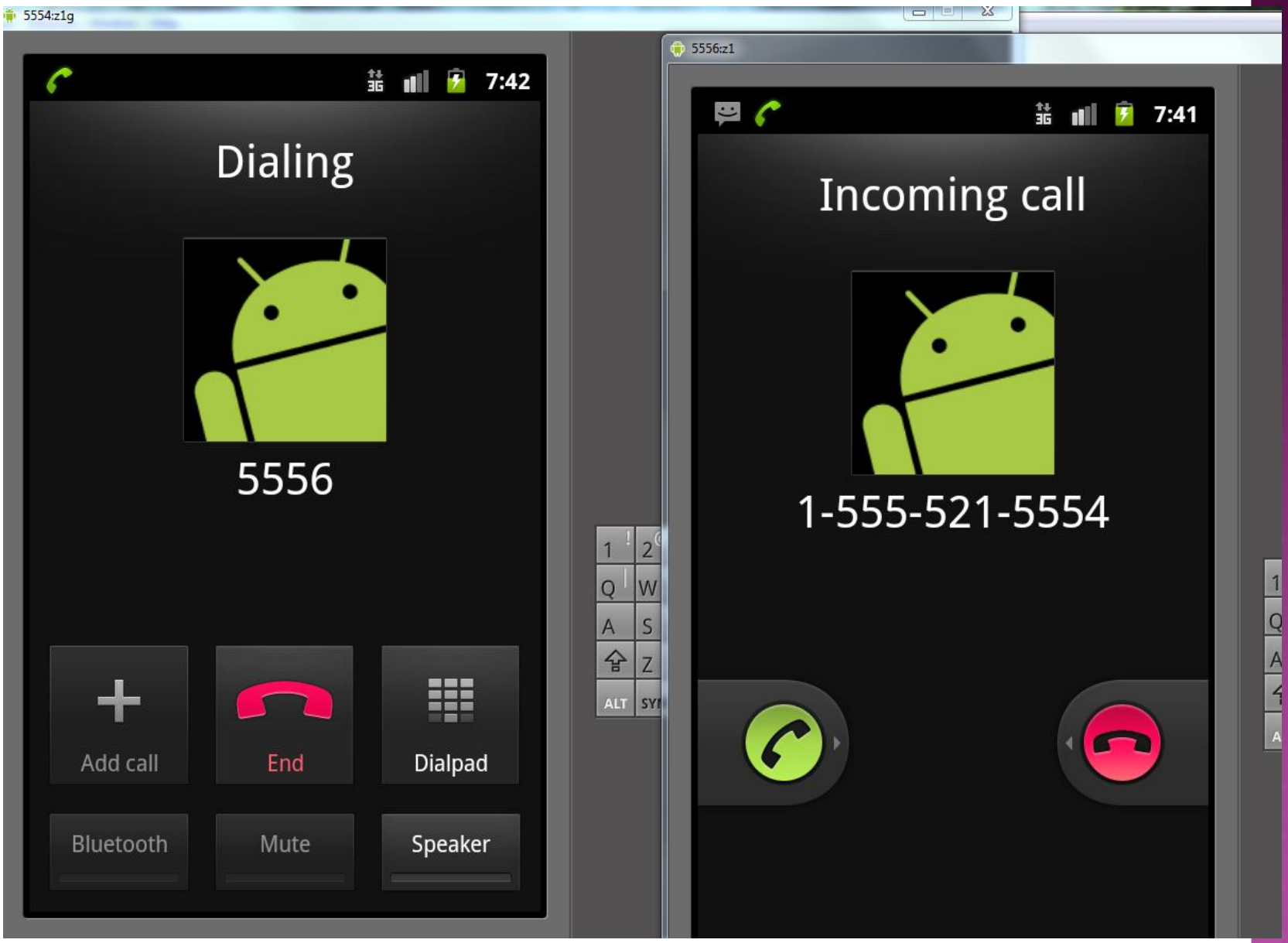
zdroj: http://stackoverflow.com/questions/4324659/android-handle-phone-call

VOLÁNÍ

- vytvořit záměr typu **ACTION_DIAL**
- Uri obsahuje tel:12345
- použít záměr – startActivity()

spustí aktivitu určenou k volání telefonních čísel




5554:z1g
7:42
Dialing
5556
Add call
End
Dialpad
Bluetooth
Mute
Speaker
5556:z1
7:41
Incoming call
1-555-521-5554

SÍŤ - HTTP KLIENT

Postup:

- mít objekt URL
- URLconnection – zavolat openConnection
- nad spojením stream
- číst

- http, https, file, jar

SÍŤ - HTTP KLIENT

```
URL url = new URL(eText.getText().toString());
URLConnection conn = url.openConnection();

// získání odpovědi
BufferedReader rd = new BufferedReader(new
InputStreamReader(conn.getInputStream()));
    String line = "";
    while ((line = rd.readLine()) != null) {
        …
    }
```

FRAGMENT

- Nezávislá komponenta zapouzdřená do aktivity
- Ovlivněn životním cyklem aktivity

- Běžící aktivita může manipulovat s každým fragmentem nezávisle (přidat, odstranit)

Fragment transaction

- Lze přidat do back stage spravovaného aktivitou
- Na tlačítko back – vrátit předchozí fragment

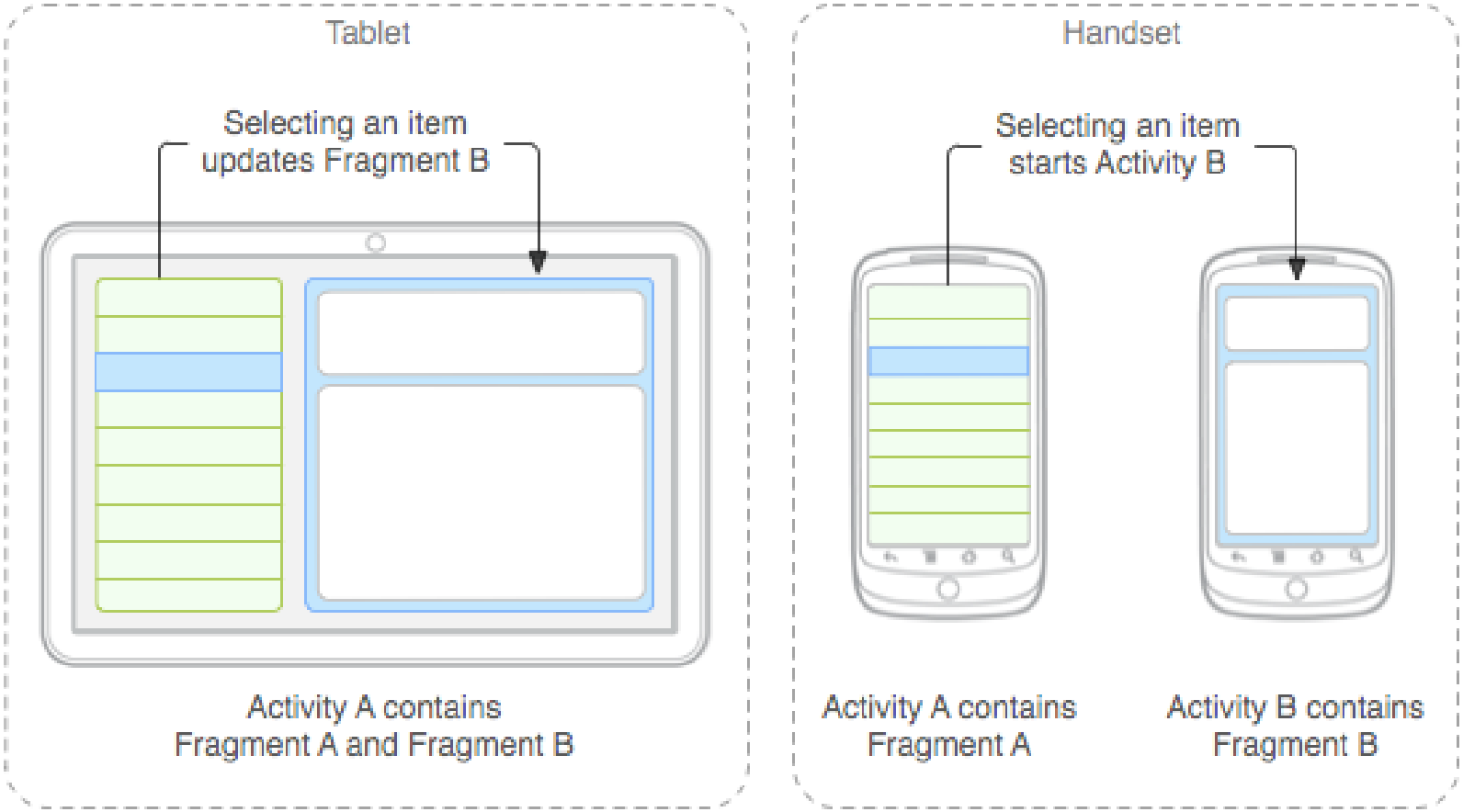
FRAGMENTY

- Fragment uvnitř ViewGroup
  <fragment>
- Lze také použít fragment bez UI (worker pro aktivitu)
- Od Android 3.0 (API 11)
  podpora větších obrazovek – tabletů

klasický příklad:

- vlevo - fragment - seznam článků
- vpravo - fragment - detail vybraného článku
- vše uvnitř jedné aktivity

FRAGMENTY: 1 OR 2 AKTIVITY



Zdroj: Android SDK dokumentace

FRAGMENT

- vlastní layout
- vlastní chování, callbacky
- jeden fragment lze do více aktivit
- vyhnout se manipulaci fragmentu z jiného fragmentu

- Vytvořit podtřídu fragmentu
  onCreate
  onCreateView
  onStart
  onResume
  onPause
  onStop

FRAGMENT

další podtřídy:

- DialogFragment
- ListFragment
- PreferenceFragment

FragmentManager:
FragmentTransaction
(begin, add, commit)

fragment může přidávat položky
do menu i ActionBaru

FRAGMENTY

tutorial:
http://www.vogella.com/articles/AndroidFragments/article.html

PŘÍKLAD - DETAIL FRAGMENT

```
package com.example.android.rssfeed;

import android.app.Fragment;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;

public class DetailFragment extends Fragment {

  @Override
  public View onCreateView(LayoutInflater inflater, ViewGroup container,
      Bundle savedInstanceState) {
    View view = inflater.inflate(R.layout.fragment_rssitem_detail,
        container, false);
    return view;
  }

  public void setText(String item) {
    TextView view = (TextView) getView().findViewById(R.id.detailsText);
    view.setText(item);
  }
}
```

PŘÍKLAD - ODPOVÍDAJÍCÍ LAYOUT

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >

    <TextView
        android:id="@+id/detailsText"
        android:layout_width="wrap_content"
        android:layout_height="match_parent"
        android:layout_gravity="center_horizontal|center_vertical"
        android:layout_marginTop="20dip"
        android:text="Default Text"
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:textSize="30dip" />

</LinearLayout>
```

fragment_rssitem_detail.xml

PŘÍKLAD - LAYOUT AKTIVITY SE DVĚMA FRAGMENTY

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="horizontal" >

    <fragment
        android:id="@+id/listFragment"
        android:layout_width="0dp"
        android:layout_weight="1"
        android:layout_height="match_parent"
        android:layout_marginTop="?android:attr/actionBarSize"
        class="com.example.android.rssfeed.MyListFragment" ></fragment>

    <fragment
        android:id="@+id/detailFragment"
        android:layout_width="0dp"
        android:layout_weight="2"
        android:layout_height="match_parent"
        class="com.example.android.rssfeed.DetailFragment" >
        <!-- Preview: layout=@layout/details -->
    </fragment>

</LinearLayout>
```

KOMUNIKACE APLIKACE S FRAGMENTY

- fragmenty by neměli mezi sebou navzájem přímo komunikovat -> přes aktivitu
- fragment definuje rozhraní, které aktivita musí implementovat
- v onAttach lze zkontrolovat, zda aktivita rozhraní implementuje
- příklad
  fragment by měl předat hodnotu Aktivitě
  http://www.vogella.com/articles/AndroidFragments/article.html

PŘÍKLAD - AKTIVITA

```
package com.example.android.rssfeed;

import android.os.Bundle;
import android.app.Activity;
import android.view.Menu;

public class RssfeedActivity extends Activity implements MyListFragment.OnItemSelectedListener{

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_rssfeed);
    }

    // if the wizard generated an onCreateOptionsMenu you can delete
    // it, not needed for this tutorial

  @Override
  public void onRssItemSelected(String link) {
    DetailFragment fragment = (DetailFragment) getFragmentManager()
            .findFragmentById(R.id.detailFragment);
        if (fragment != null && fragment.isInLayout()) {
          fragment.setText(link);
        }
  }

}
```

ZMĚNA FRAGMENTU ZA BĚHU

- prostřednictvím transakcí
- nový fragment nahradí předchozí
- lze doplnit animací
- lze přidat do back stacku – reaguje na tl. zpět

```
FragmentTransaction
  ft =  getFragmentManager().beginTransaction();

ft.replace(R.id.your_placehodler, new YourFragment());

ft.commit();
```

DEVICE AWAKE STATE - SCREEN

- idle -> dim -> turn off screen -> turn off CPU
- někdy potřebujeme ponechat vzhůru
  - obrazovka nebo jen CPU

- programově v aktivitě obrazovka vzhůru:

```
getWindow().addFlags(
  WindowManager.LayoutParams.FLAG_KEEP_SCREEN_ON);

getWindow().clearFlags(
  WindowManager.LayoutParams.FLAG_KEEP_SCREEN_ON);
```

- nebo v XML popisu Layoutu (viz další slide):

```
android:keepScreenOn="true"
```

UKÁZKA LAYOUTU

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:keepScreenOn="true">
    ...
</RelativeLayout>
```

AWAKE - CPU

- wake lock (PowerManager systémová služba)
- vytvořit a držet jen po nejkratší dobu
- vliv na baterii

- právo v manifestu
  <uses-permission
  android:name="android.permission.WAKE_LOCK" />

```
PowerManager powerManager = (PowerManager)
getSystemService(POWER_SERVICE);
Wakelock wakeLock =
powerManager.newWakeLock(PowerManager.PARTIAL_WAKE_LOCK,
        "MyWakelockTag");
wakeLock.acquire();
  ...
wakeLock.release();
```

AWAKE - CPU - WAKEFULBROADCASTRECEIVER

- Často se wakelock nevyužívá takto přímo, ale přes službu
- WakefulBroadcastReceiver

```
Intent service = new Intent(context, MyIntentService.class);
startWakefulService(context, service);
```

Zavolá se completeWakefulIntent
pro uvolnění wake_locku

PŘÍKLAD

```
<receiver android:name=".MyWakefulReceiver"></receiver>
```

```
public class MyWakefulReceiver extends WakefulBroadcastReceiver {

    @Override
    public void onReceive(Context context, Intent intent) {

        // Start the service, keeping the device awake while the service is
        // launching. This is the Intent to deliver to the service.
        Intent service = new Intent(context, MyIntentService.class);
        startWakefulService(context, service);
    }
}
```

```
public class MyIntentService extends IntentService {
    public static final int NOTIFICATION_ID = 1;
    private NotificationManager mNotificationManager;
    NotificationCompat.Builder builder;
    public MyIntentService() {
        super("MyIntentService");
    }
    @Override
    protected void onHandleIntent(Intent intent) {
        Bundle extras = intent.getExtras();
        // Do the work that requires your app to keep the CPU running.
        // ...
        // Release the wake lock provided by the WakefulBroadcastReceiver.
        MyWakefulReceiver.completeWakefulIntent(intent);
    }
}
```

ALARMY

- třída AlarmManager
- např. pustit službu jednou za den

co umí:

- fire Intent („vypustit Intent“)
- nastavit čas nebo interval
- spolu s broadcast receivery pro spuštění služby
- mimo lifetime aplikace, ta nemusí běžet, zařízení může spát

- elapsed real time - čas od bootu zařízení (např. každých 30 sekund)
- real time clock – „wall clock“ time
- wake up verze – vzbudit CPU ze spánku zařízení

TYPY ALARMŮ

ELAPSED_REALTIME	Odvozen od času bootu zařízení, nebudí zařízení (např. každých 30 sekund)
ELAPSED_REALTIME_WAKEUP	Vzbudí zařízení a vyvolá pending intent po uplynutí doby od bootu zařízení
RTC	V specifický čas, nevzbudí zařízení když spí
RTC_WAKEUP	Vzbudí zařízení v daný čas

PŘÍKLADY

Vzbudí za 30 min a potom každých 30 minut

```
// Hopefully your alarm will have a lower frequency than this!
alarmMgr.setInexactRepeating(AlarmManager.ELAPSED_REALTIME_WAKEUP,
        AlarmManager.INTERVAL_HALF_HOUR,
        AlarmManager.INTERVAL_HALF_HOUR, alarmIntent);
```

Vzbudí za minutu, neopakující se

```
private AlarmManager alarmMgr;
private PendingIntent alarmIntent;
...
alarmMgr = (AlarmManager)context.getSystemService(Context.ALARM_SERVICE);
Intent intent = new Intent(context, AlarmReceiver.class);
alarmIntent = PendingIntent.getBroadcast(context, 0, intent, 0);

alarmMgr.set(AlarmManager.ELAPSED_REALTIME_WAKEUP,
        SystemClock.elapsedRealtime() +
        60 * 1000, alarmIntent);
```

PŘÍKLADY (RTC WAKEUP)

Vzbudí v 14:00 a opakuj každý den

```
// Set the alarm to start at approximately 2:00 p.m.
Calendar calendar = Calendar.getInstance();
calendar.setTimeInMillis(System.currentTimeMillis());
calendar.set(Calendar.HOUR_OF_DAY, 14);

// With setInexactRepeating(), you have to use one of the AlarmManager interval
// constants--in this case, AlarmManager.INTERVAL_DAY.
alarmMgr.setInexactRepeating(AlarmManager.RTC_WAKEUP, calendar.getTimeInMillis(),
        AlarmManager.INTERVAL_DAY, alarmIntent);
```

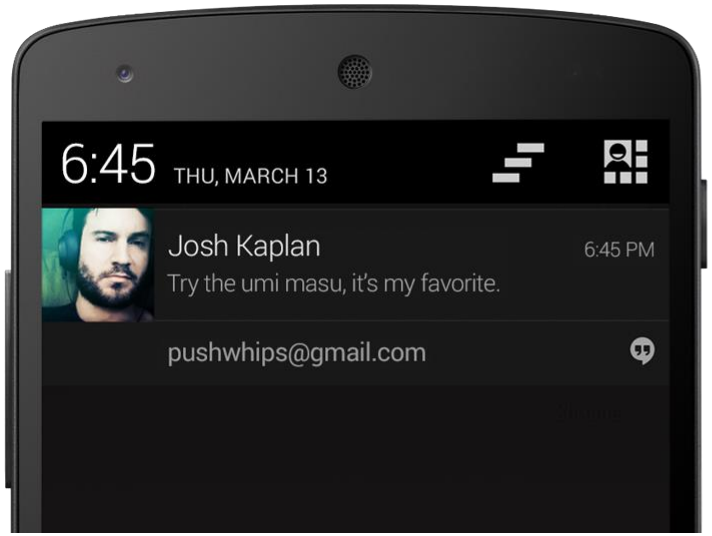
DALŠÍ

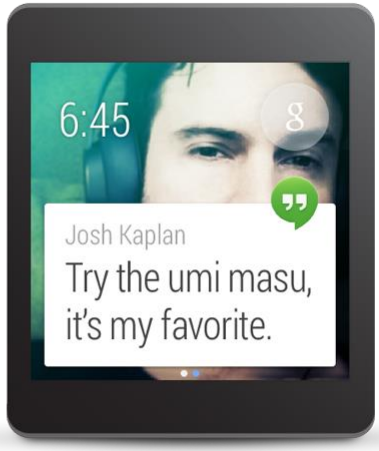
- gesta
- OpenGL ES, herní frameworky
- testování aplikace
- publikování aplikace
- reklamy
- nakupování v aplikaci
- …

ANDROID WEAR

TERMINOLOGIE

- Android handheld (telefon, table)
- Android wereable (hodinky)





Zdroj:
https://developer.android.com/training/wearables/notifications/index.html

ANDROID WEARABLE

Pro spolupráci s wearable:

- Android 4.3
- Android Support Library
- Google Play Services

Synced Notifications

Notifications on handhelds can automatically sync to wearables, so design them with both devices in mind.

Build notifications

Voice Actions

Register your app to handle voice actions, like "Ok Google, take a note," for a hands-free experience.

Integrate voice actions

Build Wearable Apps

Create custom experiences with activities, services, sensors, and much more with the Android SDK.

Create wearable apps

Send Data

Send data and actions between handhelds and wearables with data replication APIs and RPCs.

Work with the Data Layer

MOŽNOSTI VÝVOJE

- Synchronizované notifikace
  - Notifikace mobil -> hodinky
  - Karta ve streamu
- Hlasové pokyny (voice actions)
  - Registrace aplikace pro obsluhu hlasových akcí
  - “OK Google, umyj nádobí”
- Wearable aplikace
  - Full screen UI aplikace
- Výměna dat
  - Data replication API, RPC

NOTIFIKACE

- Telefon **sdílí notifikace** s hodinkami
- Každá notifikace se zobrazuje jako nová karta v context streamu
- K notifikacím lze přidat wereable-specific funkcionalitu

CONTEXT STREAM

- Vertikální seznam karet
- Každá zobrazuje nějakou informaci
- V danou chvíli pouze 1 karta zobrazena
- Obrázek (fotka) v pozadí karty
- Uživatel přepíná (swipe) mezi kartami
- Aplikace může vytvořit kartu a vložit ji do streamu
- Karta je víc než notifikace
  - Horizontální posun – další informace
  - Posun vlevo – odstranění ze streamu
  - Tlačítka – akce nad danou notifikací

UŽIVATEL

- Návrhy (suggest)
  - Proaktivní informace uživatele context streamem
  - Uživatel nepouští aplikace, ale čte si karty

- The Cue Card
  - Otevře se proslovem „OK Google“
  - Otevře se také dotykem pozadí home screenu
  - Seznam navrhovaných hlasových pokynů
  - Voice command aktivuje určitý intent
  - Více aplikací se může registrovat pro určitý hlasový pokyn => uživatel si vybere

Reakce aplikace na pokyn

- Přidání / update stream card
- Spuštění fullscreen aplikace

- Jednoduchá potvrzovací animace (přidání připomínky)

UI

- Home screen
  - Pozadí – dle první karty nebo watch face
  - Dotyk na pozadí nebo OK, Google -> voice query
  - Stavové indikátory
    - Konektivita, stav napájení, airplane mode
- Watch faces
  - Čas a nejlépe hodnocená karta navrch
  - Dlouhý stisk – volba vzhledu
- Ambient mode
  - Šetří napájení při nepoužívání
  - Ztmavení obrazovky

UI

- Pohyb dolů z home screen
  - Datum a stav baterie
  - Mute mode
- Setting screen
  - Vypnutí a restart zařízení
  - Nastavení jasu, airplane mode atd.
- Full screen aplikace

FILOZOFIE WEARABLE - ROZDÍL

- Telefon
  - Klikneme na ikonu a spustí se aplikace
- Wearable
  - Správná informace v pravý čas
  - Aplikace zná kontext uživatele – čas, polohu, fyzickou aktivitu
  - Vloží kartu do streamu ve vhodný čas
  - Uživatel hodinkám věnuje jen „mrknutí“
  - Aplikace by neměla uživatele zdržovat, odvádět jeho pozornost
  - Jednoduché minimální interakce
  - Osobní asistent

PRAVIDLO 5 SEKUND

- Aplikace by neměla vyžadovat pozornost uživatele pro každou akci na více než 5 sekund
- Gesta – nevyžadovaná přesnost
  - Nedávat např. 5 titěrných voleb
  - Fotka předchozí / následující -> OK
- Představit si modelové situace, kdy uživatel vaší aplikaci využije

ODKAZ

- https://developer.android.com/design/wear/structure.html

NOTIFIKACE

- NotificationCompat.Builder

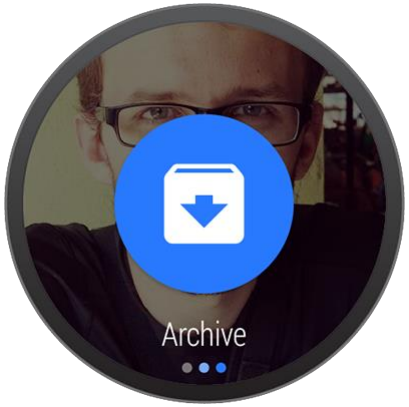
- Na mobilu
  - Klasické zobrazení
  - Poklepáním na notifikaci se vyvolá Intent

- Wearable
  - Notifikaci posuneme vlevo
  - Objeví se Open akce, která vyvolá Intent na mobilu

NOTIFIKACE VČETNĚ INTENTU

```
int notificationId = 001;
// Build intent for notification content
Intent viewIntent = new Intent(this, ViewEventActivity.class);
viewIntent.putExtra(EXTRA_EVENT_ID, eventId);
PendingIntent viewPendingIntent =
        PendingIntent.getActivity(this, 0, viewIntent, 0);

NotificationCompat.Builder notificationBuilder =
        new NotificationCompat.Builder(this)
        .setSmallIcon(R.drawable.ic_event)
        .setContentTitle(eventTitle)
        .setContentText(eventLocation)
        .setContentIntent(viewPendingIntent);

// Get an instance of the NotificationManager service
NotificationManagerCompat notificationManager =
        NotificationManagerCompat.from(this);

// Build the notification and issues it with notification manager.
notificationManager.notify(notificationId, notificationBuilder.build());
```

NOTIFIKACE + AKČNÍ TLAČÍTKO




```
NotificationCompat.Builder notificationBuilder =
        new NotificationCompat.Builder(this)
        .setSmallIcon(R.drawable.ic_event)
        .setContentTitle(eventTitle)
        .setContentText(eventLocation)
        .setContentIntent(viewPendingIntent)
        .addAction(R.drawable.ic_map,
                getString(R.string.map), mapPendingIntent);
```

ZOBRAZENÍ

- Na mobilu
  - Tlačítko v notifikaci
- Na wearable
  - Velké tlačítko, když uživatel posune notifikaci vlevo

NOTIFIKACE - SPECIFICKÉ PRO WERABLE

- Chceme zobrazit jiné akce na mobilu a jiné na hodinkách
- WearableExtender.addAction()
- Hodinky zobrazí jen tyto akce
- Naopak mobil je nezobrazí

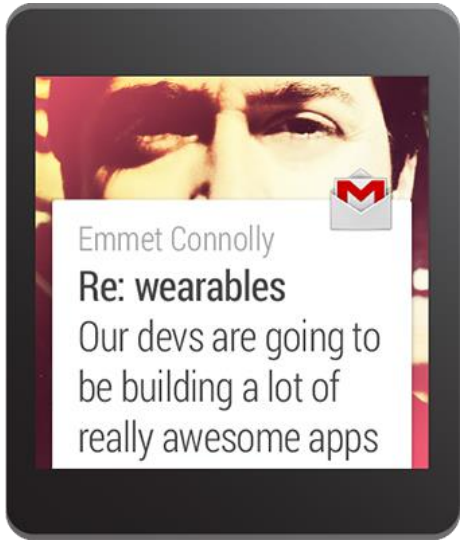
```
// Create an intent for the reply action
Intent actionIntent = new Intent(this, ActionActivity.class);
PendingIntent actionPendingIntent =
        PendingIntent.getActivity(this, 0, actionIntent,
                PendingIntent.FLAG_UPDATE_CURRENT);

// Create the action
NotificationCompat.Action action =
        new NotificationCompat.Action.Builder(R.drawable.ic_action,
                getString(R.string.label), actionPendingIntent)
                .build();

// Build the notification and add the action via WearableExtender
Notification notification =
        new NotificationCompat.Builder(mContext)
                .setSmallIcon(R.drawable.ic_message)
                .setContentTitle(getString(R.string.title))
                .setContentText(getString(R.string.content))
                .extend(new WearableExtender().addAction(action))
                .build();
```

NOTIFIKACE - BIG VIEW

- Rozšířený obsah notifikace (více textu)
- Mobil – při expanzi notifikace
- Wearable – defaultně

- Na objekt NotificationCompat.Builder zavoláme metodu setStyle()

```
NotificationCompat.Builder notificationBuilder =
        new NotificationCompat.Builder(this)
        .setSmallIcon(R.drawable.ic_event)
        .setLargeIcon(BitmapFactory.decodeResource(
                getResources(), R.drawable.notif_background))
        .setContentTitle(eventTitle)
        .setContentText(eventLocation)
        .setContentIntent(viewPendingIntent)
        .addAction(R.drawable.ic_map,
                getString(R.string.map), mapPendingIntent)
        .setStyle(bigStyle);
```



VOICE INPUT V NOTIFIKACI

- Problém zadání textu (odpověď na e-mail)

- Mobil – pustí aktivitu, zadání textu
- Wearable – nemá klávesnici
  - Nadiktovat
  - Předdefinované zprávy (RemoteInput)

- Odpověď se připojí k Intentu a ten se spustí na mobilu
- Emulátor – nemá hlasový vstup, lze zvolit hardware keyboard present a zadat ručně

AKCE PODPORUJÍCÍ VOICE INPUT

- Instance RemoteInput.Builder
  - Lze přidat k notifikační akci
  - Řetězec, který systém použije jako klíč pro hlasový vstup

```
// Key for the string that's delivered in the action's intent
private static final String EXTRA_VOICE_REPLY = "extra_voice_reply";

String replyLabel = getResources().getString(R.string.reply_label);

RemoteInput remoteInput = new RemoteInput.Builder(EXTRA_VOICE_REPLY)
        .setLabel(replyLabel)
        .build();
```

PŘEDDEFINOVANÉ ODPOVĚDI

- Lze předdefinovat až 5 textových odpovědí, které může uživatel vybrat
- Volání setChoices()

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string-array name="reply_choices">
        <item>Yes</item>
        <item>No</item>
        <item>Maybe</item>
    </string-array>
</resources>
```

PŘEDDEFINOVANÉ ODPOVĚDI

```
public static final String EXTRA_VOICE_REPLY = "extra_voice_reply";
...
String replyLabel = getResources().getString(R.string.reply_label);
String[] replyChoices = getResources().getStringArray(R.array.reply_choices);

RemoteInput remoteInput = new RemoteInput.Builder(EXTRA_VOICE_REPLY)
        .setLabel(replyLabel)
        .setChoices(replyChoices)
        .build();
```

Přidání hlasového vstupu jako notifikační akce

Zde je celkový kód:

https://developer.android.com/training/wearables/notifications/voice-input.html#AddAction

JAK DOSTAT HLASOVÝ VSTUP JAKO STRING?

- Zavoláme getResultsFromIntent()
  - Vrací Bundle obsahující odpověď

```
private CharSequence getMessageText(Intent intent) {
    Bundle remoteInput = RemoteInput.getResultsFromIntent(intent);
    if (remoteInput != null) {
        return remoteInput.getCharSequence(EXTRA_VOICE_REPLY);
    }
    return null;
}
```

Přidání stránek do notifikace

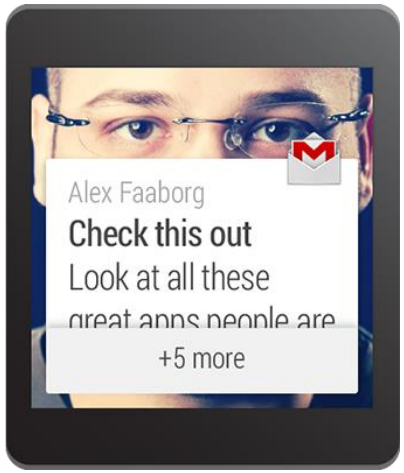
- Lze přidat stránky k notifikaci
- Zobrazí se vpravo od notifikační karty

1. Hlavní notifikace, jak se zobrazí na mobilu
2. Přídavné stránky pro wearable
3. Přidat stránky do hlavní notifikace přes addPages()

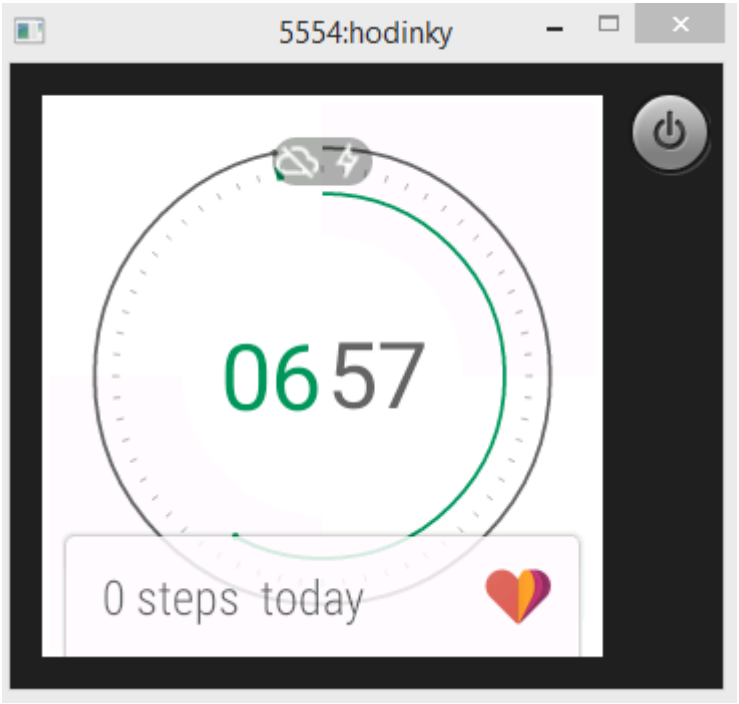
STOHOVÁNÍ NOTIFIKACÍ

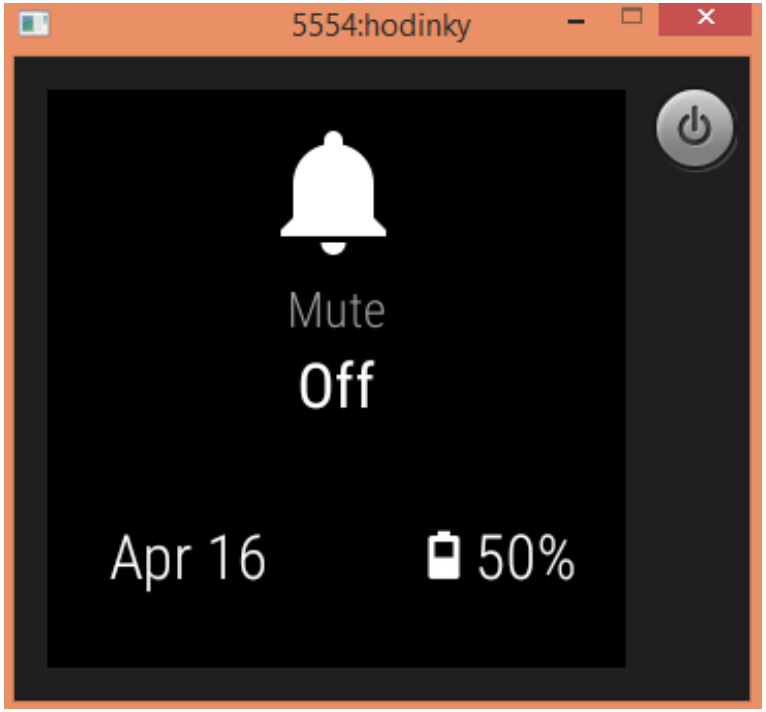
- Agregace podobných do sumární notifikace
- Např. 4 nové zprávy

- Stack notifikací
  - Zobrazí se jako jedna karta
  - Uživatel může expandovat a zobrazit detaily každé notifikace
  - Metoda setGroup()
  - Metoda setGroupSummary()

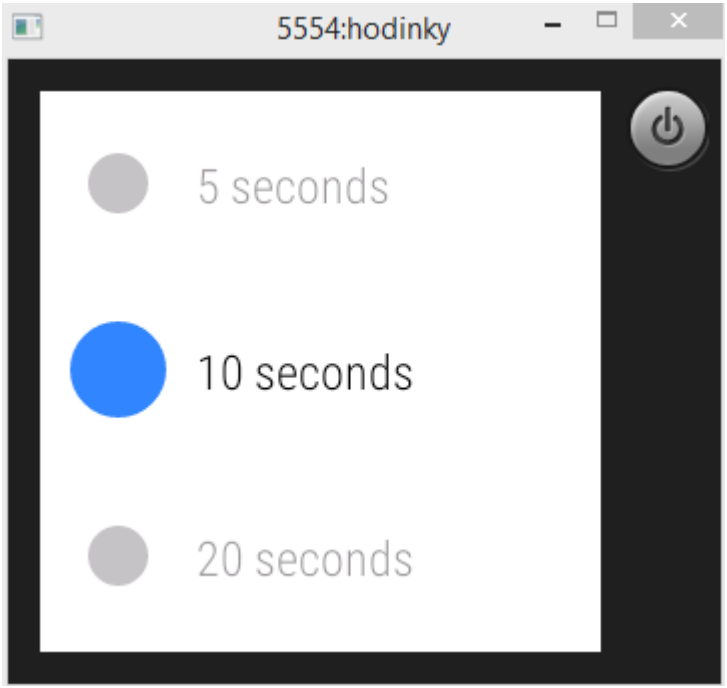


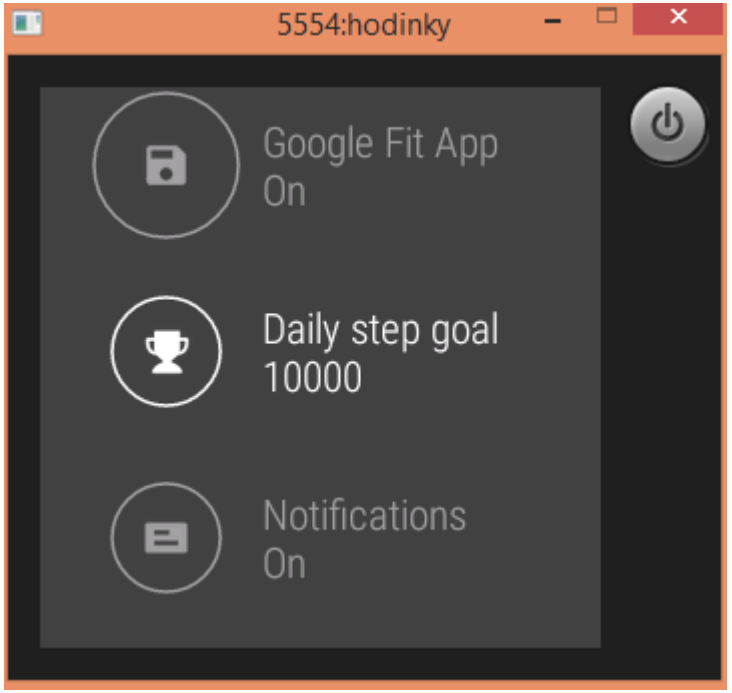
WEARABLE V EMULÁTORU ☺





UKÁZKY


5554:hodinky
5 seconds
10 seconds
20 seconds


5554:hodinky
Google Fit App
On
Daily step goal
10000
Notifications
On

WEARABLE APLIKACE

- Běží přímo na zařízení
- Přístup k HW – senzory, GPU

- Timeout
  - Uživatel neinteraguje, zařízení se uspí
  - Po probuzení – home obrazovka, ne naše aktivita
  - Perzistentní zobrazení -> využít notifikace

- Rozdělení funkcionality
  - Malé aplikace
  - Práci dělá mobil, výsledky zobrazuje werable

WEARABLE APLIKACE

- Instalace aplikací
  - Nedistribuují se samostatně
  - Uvnitř aplikace pro mobil
  - Uživatel nainstaluje pro mobil, automaticky na hodinky
- Podmnožina Android API
  - Nejsou android.webkit, appwidget a další
  - Lze volat hasSystemFeature()

HLASOVÉ POKYNY

- Systémové
- Aplikační

Aktivita bude reagovat na “Take a note”

```
<activity android:name="MyNoteActivity">
    <intent-filter>
        <action android:name="android.intent.action.SEND" />
        <category android:name="com.google.android.voicesearch.SELF_NOTE" />
    </intent-filter>
</activity>
```

HLASOVÉ POKYNY

Zavolej taxi	OK Google, get me a taxi	Akce com.google.android.gms.actions.RESERVE_TAXI_RESERVATION
Vytvoř poznámku	OK Google, take a note OK Google, note to self	
Nastavení budíku	OK Google, set an alarm for 8 AM	
Počet kroků	OK Google, how many steps have I taken?	

https://developer.android.com/training/wearables/apps/voice.html

APLIKAČNÍ HLASOVÉ POKYNY

- Spuštění naší aplikace pokynem: „Start MyActivityName“

- Registrace pro akci Start -> stejné jako na mobilu, že bude aktivita na launcheru
- Text který říci – definovat pomocí label

```
<application>
  <activity android:name="StartRunActivity" android:label="MyRunningApp">
      <intent-filter>
          <action android:name="android.intent.action.MAIN" />
          <category android:name="android.intent.category.LAUNCHER" />
      </intent-filter>
  </activity>
</application>
```

ROZPOZNÁVÁNÍ ŘEČI

- Vestavěný speech recognizer
- Vyhledávání, text zprávy, …

- Zavolat startActivityForResult() s využitím ACTION_RECOGNIZE_SPEECH
- Zpracujeme výsledek onActivityResult()

Celý kód:
https://developer.android.com/training/wearables/apps/voice.html

PUBLIKOVÁNÍ WERABLE APLIKACE

- Wearable aplikace uvnitř mobilní aplikace
- Při stažení mobilní automatická instalace

- Při vývoji – debug klíč
  - adb install ručně, nebo Android Studio

- Práva v manifestu werable aplikace do manifestu mobilní aplikace
- Stejné package name a čísla verzí mobilní i wearable aplikace

DEBUG PŘES BLUETOOTH

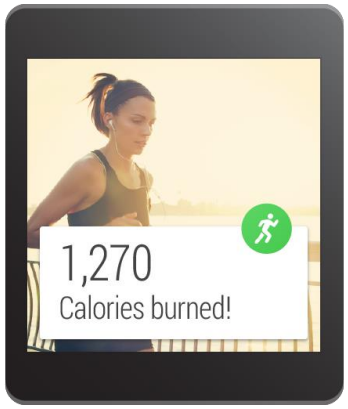
- Debug output na mobil -> připojený k PC vývojáře
- Nemá-li mobil debug menu, poklepat na About phone, scrollovat dolů a poklepat 7x na build number…
- Hodinky: Setting – About, scroll dolů, 7x build number, Developer Options, Debug over Bluetooth

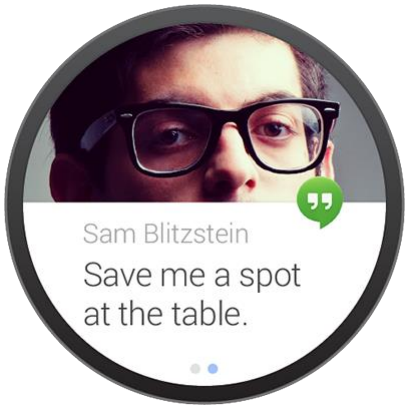
KONTEXT AWARENESS

- Co uživatele zajímá – na letišti, doma, na sjezdovce, v restauraci

PRO PŘÍZNIVCE OPERAČNÍCH SYSTÉMŮ…

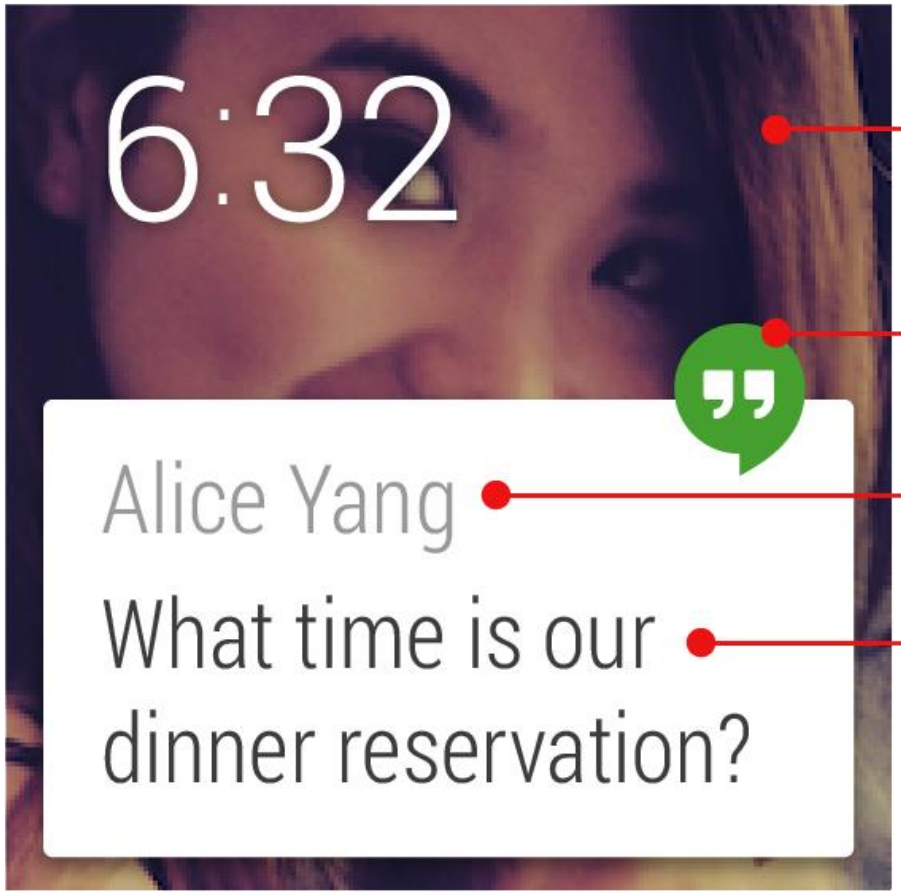
RŮZNÁ VELIKOST A TVAR HODINEK



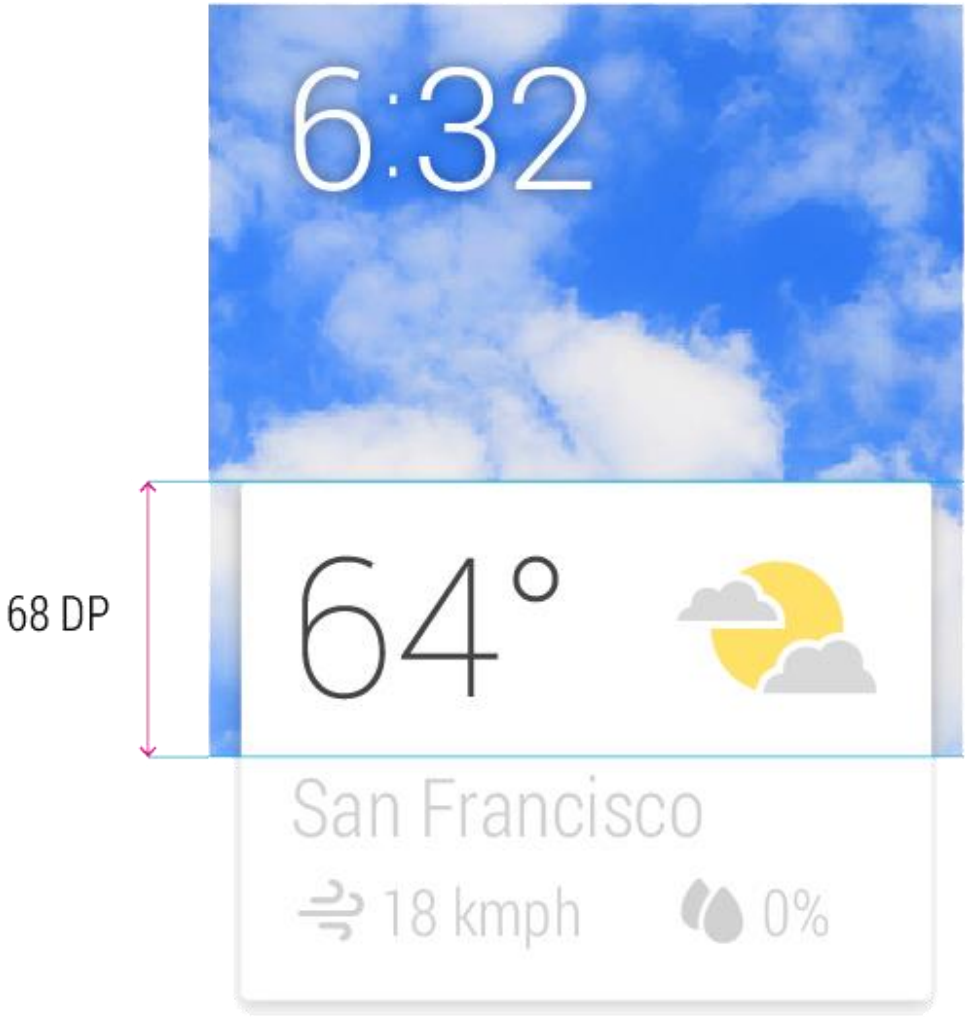


Vyzkoušet na emulátoru oba tvary
WatchViewStub – můžeme detekovat tvar zařízení

PRVKY OBRAZOVKY

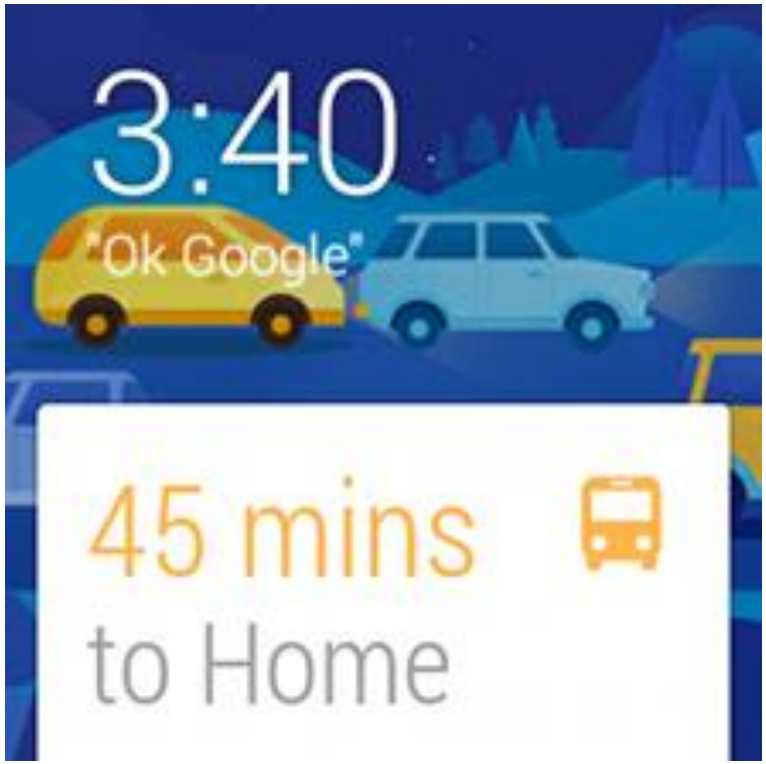


VELIKOST PÍSMEN - SNADNÉ PŘEČTENÍ



Swipe karty
Pro další informace

JEN NEJPODSTATNĚJŠÍ INFORMACE

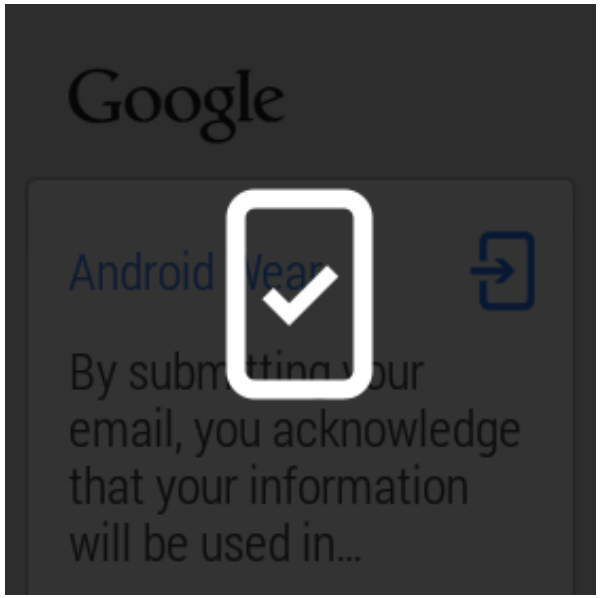


I fotografie v pozadí může dát významnou informaci

INFORMACE NA ČÁSTI

Přidání stránky addPage()

POKRAČOVÁNÍ AKTIVITY NA TELEFONU



Nejlépe co nejvíce na wearable
Když není možné a je třeba přejít na telefon, informovat uživatele

PRIVÁTNOST INFORMACÍ

- Personální zařízení, ne úplně privátní
- Senzitivní informace na další stránce – vyžaduje akci uživatele

- Notifikace o schůzce
- Zdravotní stav

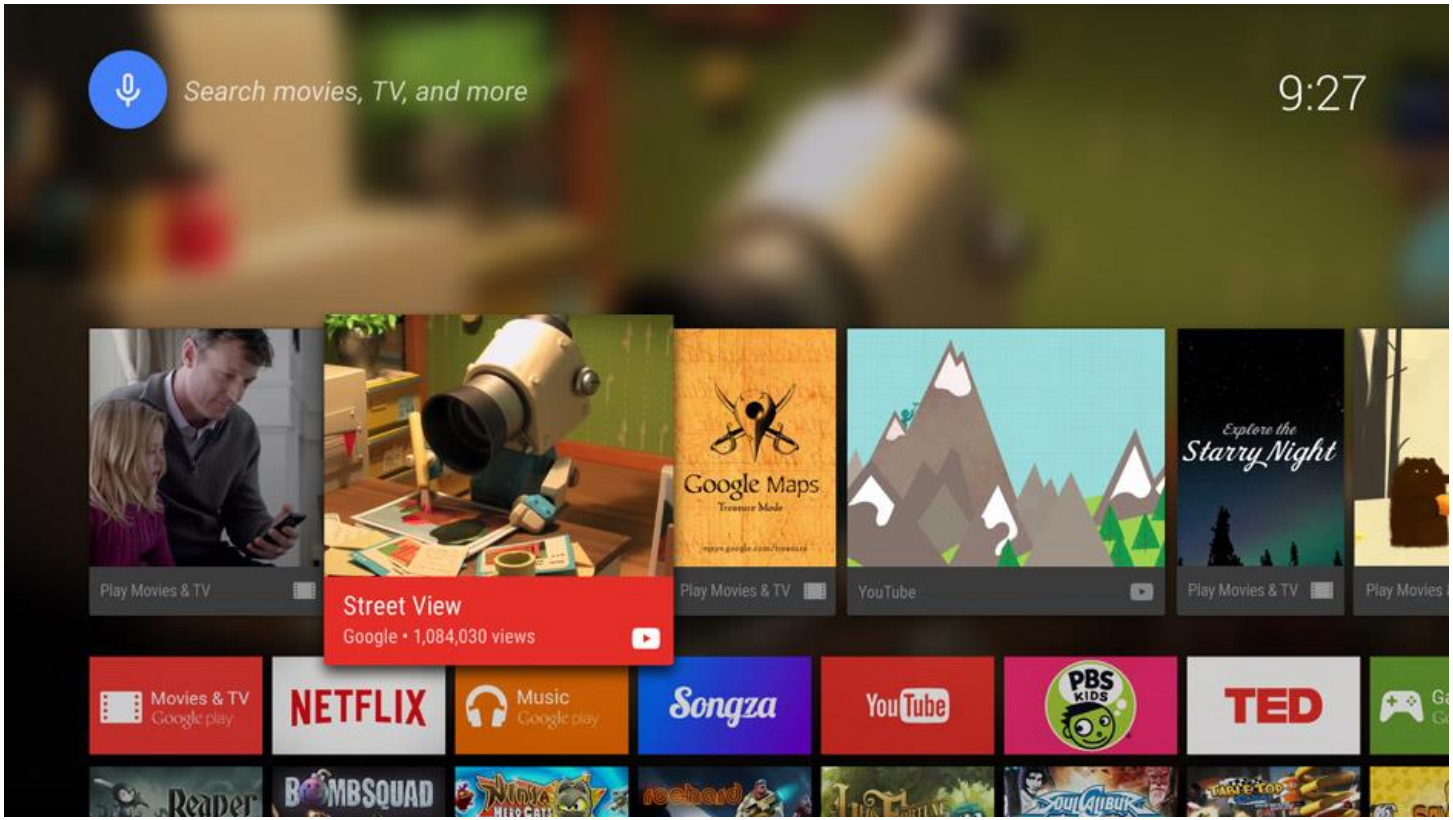
POTVRZOVACÍ ANIMACE

JAK NAVRHNOUT UI?

- Pročíst průvodce
- https://developer.android.com/design/wear/index.html

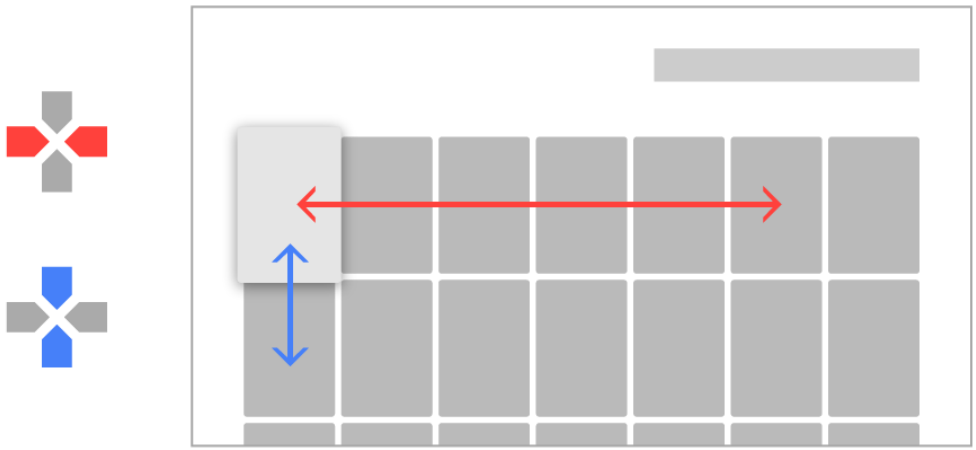
ANDROID TV

- Nejsou jen wearable (hodinky)

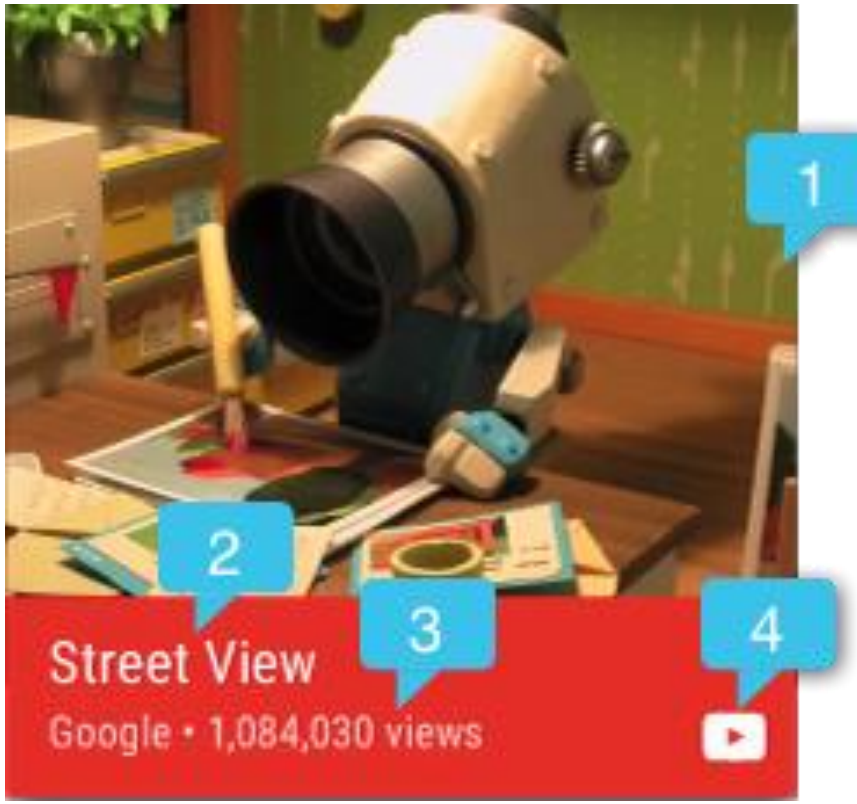


OVLÁDÁNÍ V ANDROID TV

- Directional pad (D-PAD)
- Up, down, left, right
- Seznamy, mřížky, focusovaný objekt



VZHLED DLAŽDICE

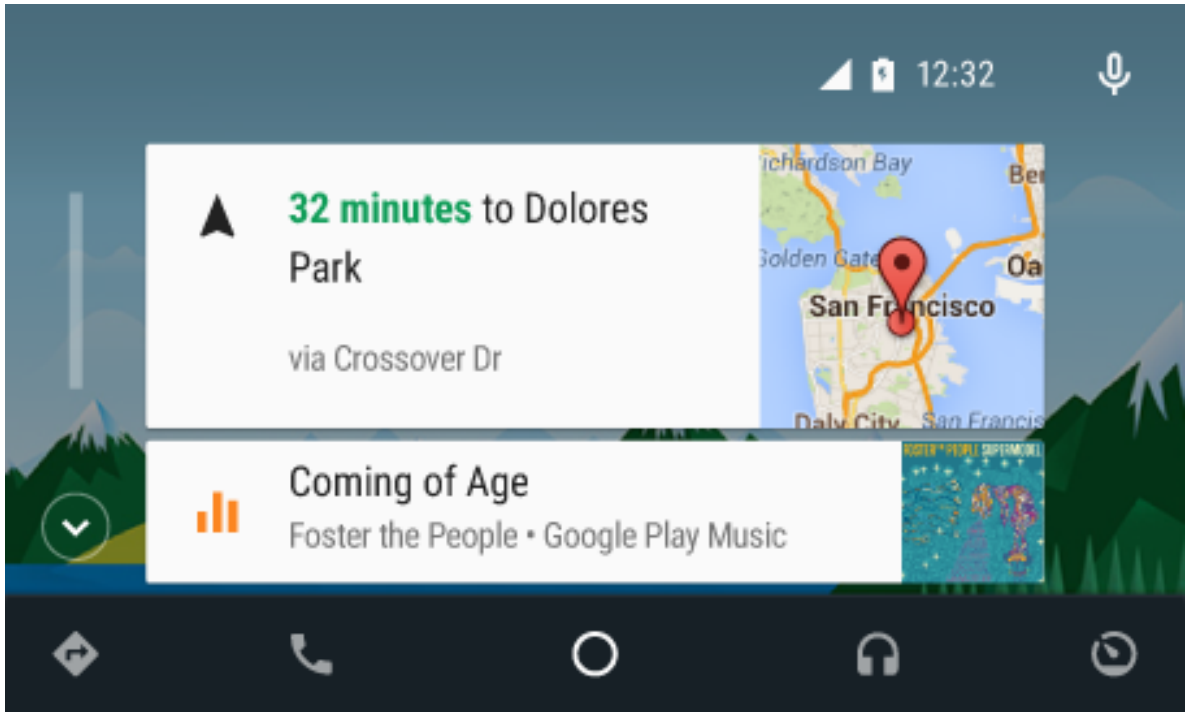


APLIKACE PRO TELEVIZI

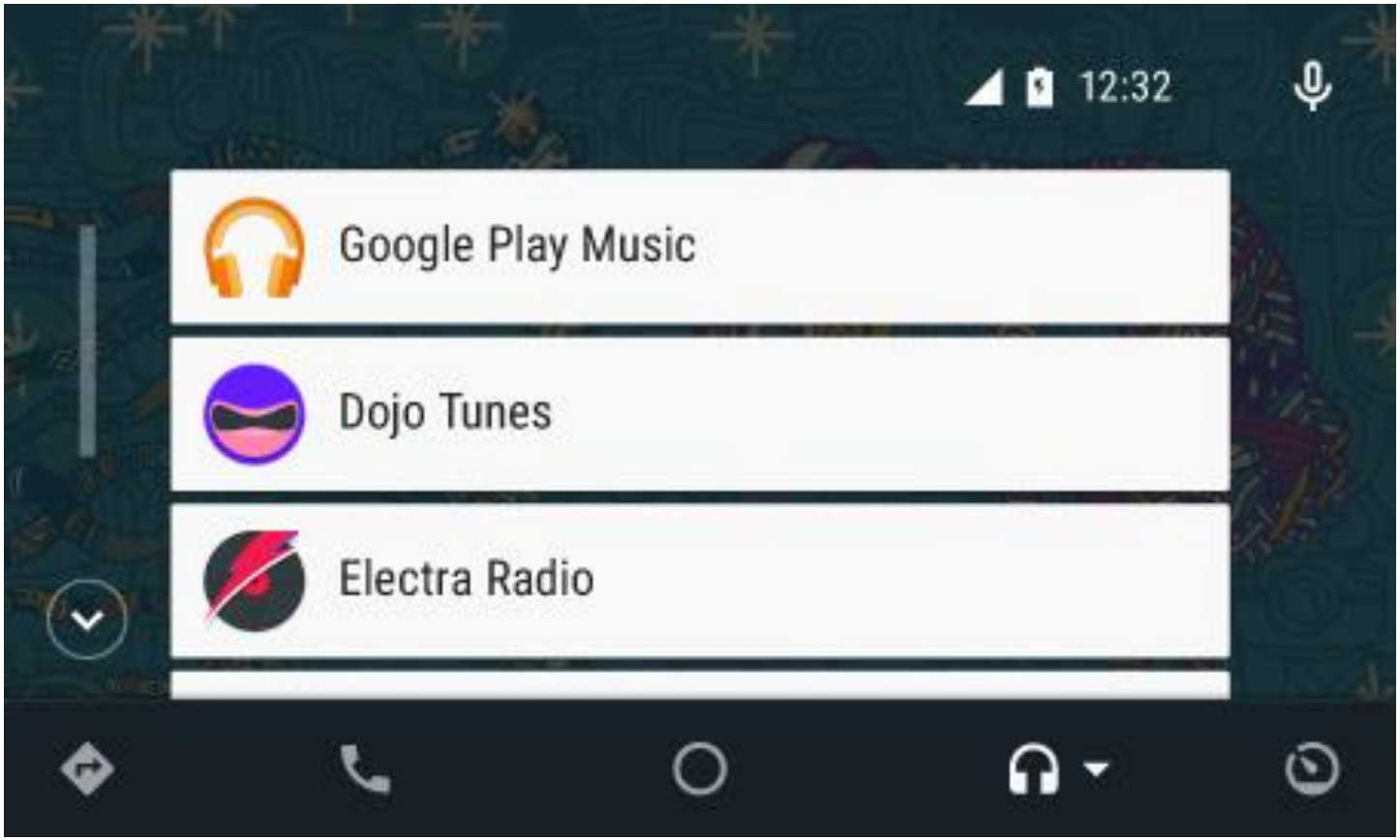
- Lze modifikovat stávající mobilní aplikace, aby běželi na TV
- Activity for TV (vyžadovaná)
- TV Support Libraries (optional)
  - Widgety pro UI
- V manifestu touchscreen not required
- V emulátoru lze vytvořit virtuální TV ☺

AUTO

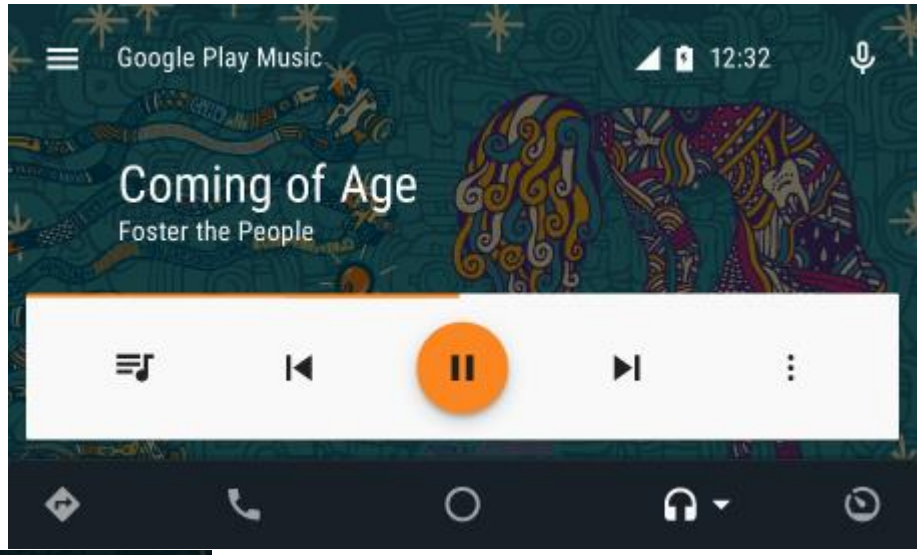
- Standardizované UI a user interaction model pro auta
- Overview Screen
  - Kontextové karty (dle lokace, čas..)

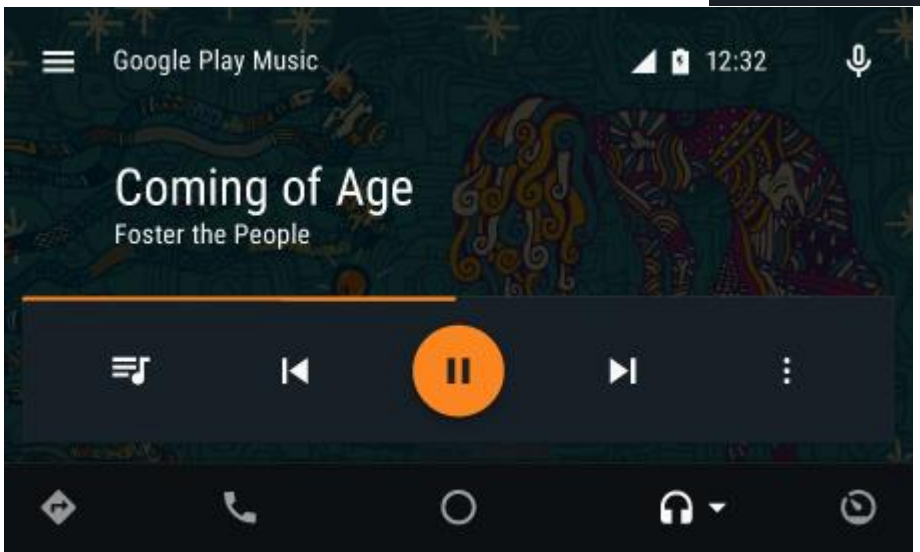


AUTO - AUDIO APP LAUNCHER



AUTO - DEN A NOC





POUŽITÍ

- Uživatel připojí svůj mobil s Android 5 do kompatibilního auta ☺
- Auto user interface poskytne car-optimized na obrazovce auta
- Hlasové akce
- Dotyková obrazovka, dashboard buttons

SOUČASNÝ STAV

Podpora dvou typů aplikací:

- Audio aplikace
  - hudba
- Messaging aplikace
  - Notifikace, text-to-speech, odpovědi přes hlasové rozhraní