Android TV és Wear OS

Ekler Péter

BME VIK AUT, AutSoft

peter.ekler@aut.bme.hu

Tematika

1. Service komponens
2. ContentProvider, Komplex felhasználói felületek
3. Játékfejlesztés
4. Grafikonok, Szenzorok, Külső osztálykönyvtárak, Multimédia alapok
5. Multimédia, további kommunikációs megoldások
6. Biztonságos alkalmazások
7. Android TV és Wear OS
8. Android 9 újdonságok és további helyfüggő szolgáltatások
9. Tesztelési lehetőségek
10. Alkalmazás publikálás, karbantartás (CI/CD)

Tartalom

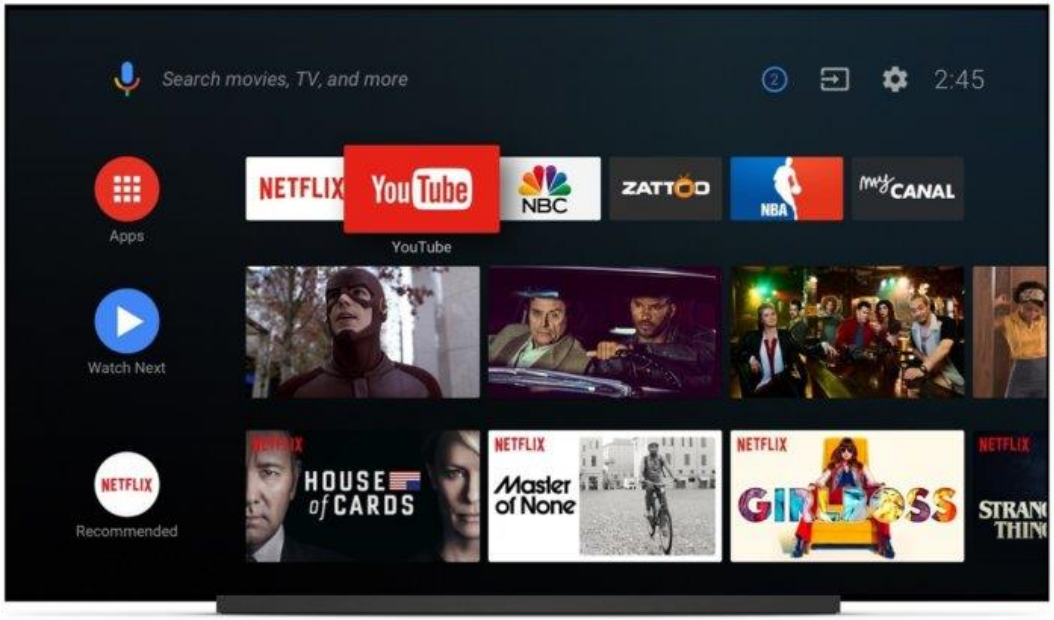
- Android TV
- Wear OS

Az Android TV platform
egy mobilfejlesztő szemével

Balogh Tamás
balogh.tamas@autsoft.hu

Tartalom

- Bevezetés
- Lehetőségek
- Megkötések
- Felépítése
- Fejlesztés



A televízió platformok fejlődése

- Régebben: Az egyes gyártók saját platformjai
  - Samsung, LG…
  - Saját fejlesztésű rendszerek
  - A felhasználóknak minden rendszer meg kell újra ismernie
  - Limitált alkalmazás futtatási lehetőségek
  - Limitált interakció más eszközökkel

A televízió platformok fejlődése

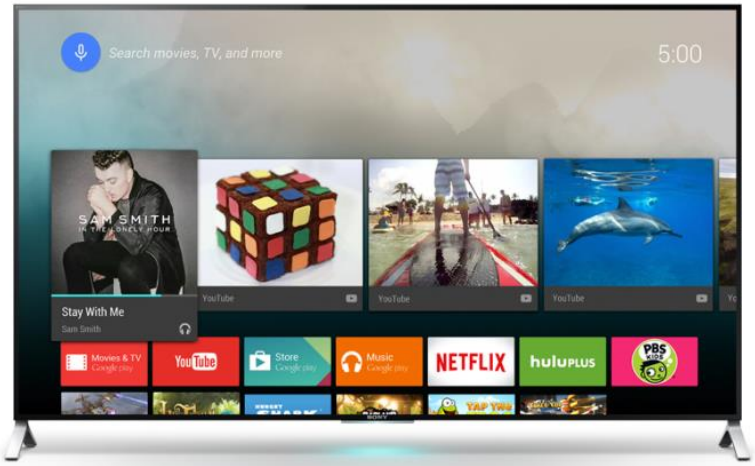
- Ma: A nappali vezető mobiltelefon rendszerek új csatatere
  - Google, Apple …
  - A már ismert platform egy újabb eszközön.
  - Interakció az eszközök között – a tv a telefon/tablet kiterjesztése
  - Akár több TV gyártó (Philips, Sony) nyújtja ugyanazt a platformot.

Előzmények

- Google TV
  - Google első okostévé próbálkozása ~2010
  - Először x86 majd ARM alapú processzor
  - Androidr-a épül: 2.1-től 4.2-ig.
  - Saját távirányító
    - IR
    - QWERTY
  - Fejlesztés:
    - Android SDK Pluging
  - Túl körülményes a használata
  - A TV nem egy nagy telefon/tablet!

Android TV

- 2014 – Google IO bejelentés
- Android 5.0+
  - A 9.0 is elérhető már
- Fejlesztés: Android SDK része
  - Android Library a fejlesztéshez – Leanback support library



Változatok

- TV-be építve: pl. Philips

Változatok

- Külön álló egységként: pl Nexus Player, Nvidia Shield

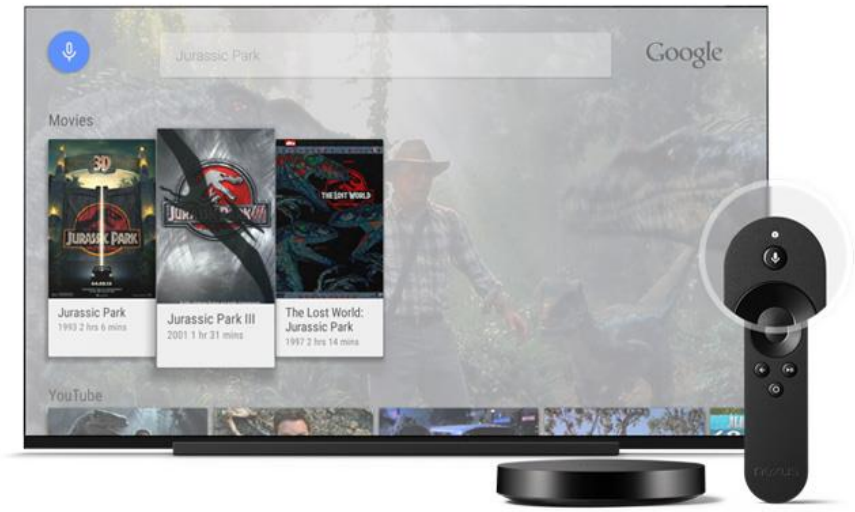
Lehetőségek

Képernyőméret

- Hatalmas kijelző
- Felbontásban lehet ugyan az …
  - 1080p telefon vs. 1080p TV
  - Képpont sűrűség kisebb
- … de méretben nem
  - 5” vs 50”
  - Xhdpi ~ 213ppi
  - Sokkal távolabbról nézi a felhasználó
  - Teljesen más felhasználói élmény

Irányítás

- D-PAD
  - 4 irány, OK, play/pause, back, home
  - Nincs QWERTY
  - Bluetooth
  - Hang alapú keresés
  - Alapfeltétel egy TV alkalmazás számára
  - Kötelező támogatni



Irányítás

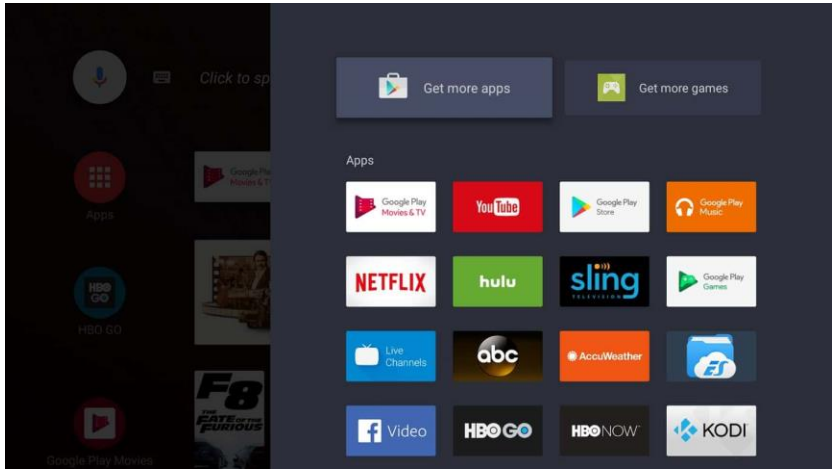
- Gamepad
  - D-PAD funkcionalitás és még sok más
  - Kifejezetten játékokhoz
  - Bluetooth
  - Store-ban külön kategória

Chrome cast

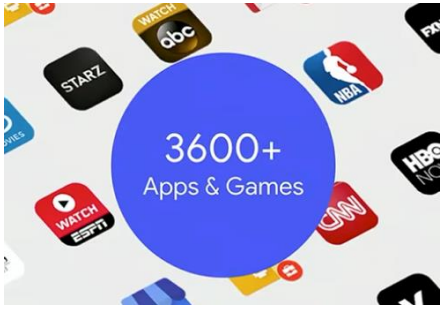
- Beépített Chrome Cast
  - Google Cast kompatibilis eszközről stream-elhetünk tartalmat
  - Nexus eszközről képernyőt is tudunk megosztani

Alkalmazások

- Számos előre telepített app
  - Play Store
  - Youtube
  - Play Games
  - Play Music



- A 3rd party appok száma egyre nő
  - Netflix, Hulu, HBO, Plex ...
  - Tune-In, Spotify...
  - TED, USA Today, Fox News ...
  - Sky Force, Asphalt, Fists of Awesome



Megkötések

Irányítás

- Nincs érintő képernyő
  - > Ez elsőre nyilvánvalónak tűnik, de fejlesztés közben egyáltalán nem az :D
  - > Mindenre a D-PAD-et kell használni
- Nincsenek szenzorok
  - > Nincs gyorsulásmérő > Csak landscape mode
  - > Nincs fény/közelség szenzor

Hardware

- Kevés memória
  - Pl. Nexus Player 1 GB memória
  - A nagy képek és video tartalmak mellet ez néha kevés
- Kevés háttértár
  - Beépítve kevés 4-8 GB tárhely
- X86 alapú processzorok elterjedtek
  - A 64-bites processzorok is elterjedtek
  - Natív kód esetén észben kell tartani

Alkalmazások

- A TV-re nem megfelelő alkalmazások hiányoznak
  - Nincs pl. böngésző
  - Nincs facebook, közösségi oldalak (Google Play van)
  - Nincs azonnali üzenet küldés
  - Nincsenek kamera, dokumentum szerkesztő alkalmazások
  - Nincs naptár, telefónia, contactok …
  - Ezeket főleg Implicit Intenteknél okoznak gondot.
- Nincs Notification Bar

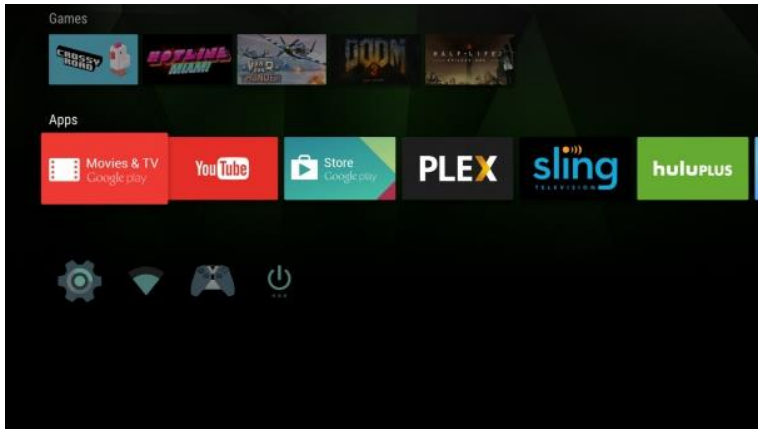
Felépítése

Felépítése

- Nincs Notification bar -> Helyette Recommendations
  - > Nem teljesen arra való mint a notification bar.
  - > Tartalom ajánlásra
  - > Éppen lejátszott média jelzésére (pl. aktuális film, zene), letöltés jelzésére stb…

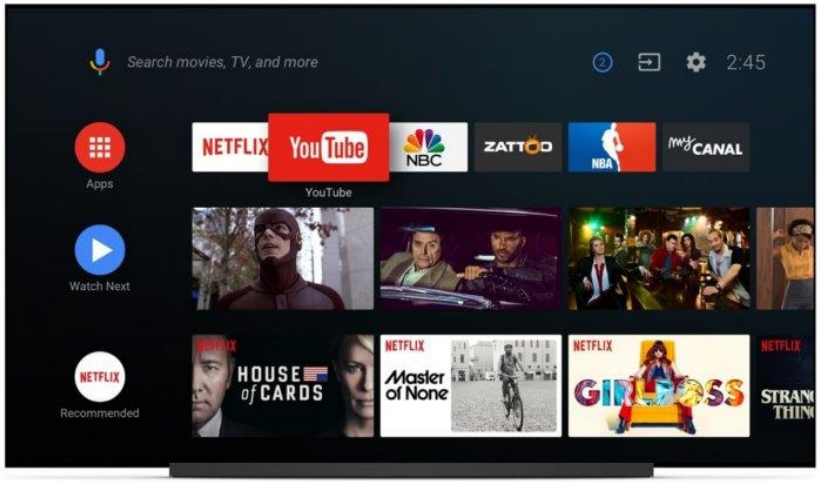
Felépítése

- Alkalmazás ikonok helyett Bannerek
  - > Mindig a leggyakrabban használt appok kerülnek legelőre!
  - > Alkalmazások és játékok külön
  - > Majd legalul a beállítások
  - > Nincs Recent Apps!

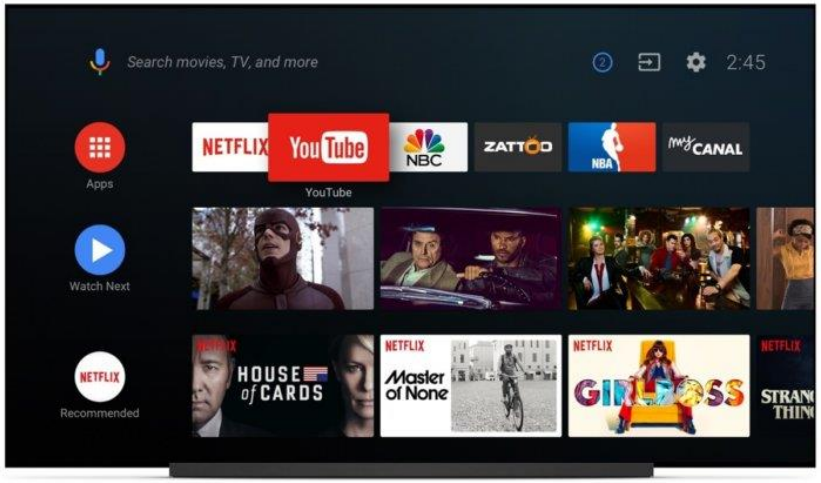


Felépítése

- Android 8.0 – Új Home képernyő
- Channels
  - > Alkalmazás csatornák
  - > Mint a recommendation, de alkalmazásonként
  - > Watch next – közös csatorna



Demo


Search movies, TV, and more
2:45
Apps
NETFLIX
YouTube
NBC
ZATTOO
NBA
myCANAL
YouTube
Watch Next
NETFLIX
Recommended
HOUSE of CARDS
Master of None
GIRLBOSS
STRANGER THINGS

Fejlesztés

Fejlesztés

- Android fejlesztői tapasztalatokra épít
  - > Először mobilra érdemes megtanulni …
  - > … majd azt kamatoztatni TV-s környezetben
  - > De attól még a TV nem egy nagy telefon!

Fejlesztés

- Ugyan az az SDK
  - Ugyanolyan alkalmazás komponensek
    - Activity, Broadcast, Service, Content Provider mind elérhető és használható
    - Fragmentek is használhatóak
  - Ugyan azok a layoutok, view-k elérheetőek
    - LinearLayout, ImageView, RelativeLayout …
    - Van amit nem érdemes használni
      - ViewPager, Pull to refresh, ListView, Google Maps etc…
  - Nyilván a hardware megkötéseket észben kell tartani
    - Nincs kamera, touch, szenzorok stb…

Fejlesztés

- A hangsúly az irányítások van
  - Fontos a Focus, az állapot jelzése
  - Fontosak az irányok
  - A lehető legkevesebb adatbevitel legyen!
    - Képernyő billentyűzet
    - Bluetooth billentyűzet csatolható, de nem feltétlenül elérhető mindig

Hogyan kezdjünk neki

- Project
  - > Android Studio > new Android TV modul
- Függőségek
  - > implementation 'com.android.support:leanback-v17:28.0.0
  - > implementation 'com.android.support:recyclerview-v7:28.0.0'
  - > implementation 'com.android.support:appcompat-v7:28.0.0'

Hogyan kezdjünk neki

- Manifest
  - > Definiálni kell hogy nincs szükség touch-screenre

```
<uses-feature android:name="android.hardware.touchscreen"
android:required="false"/>
```

  - > És hogy a Leanback funkcióra (~TV) szükség van

```
<uses-feature android:name="android.software.leanback"
android:required="true"/>
```

Hogyan kezdjünk neki

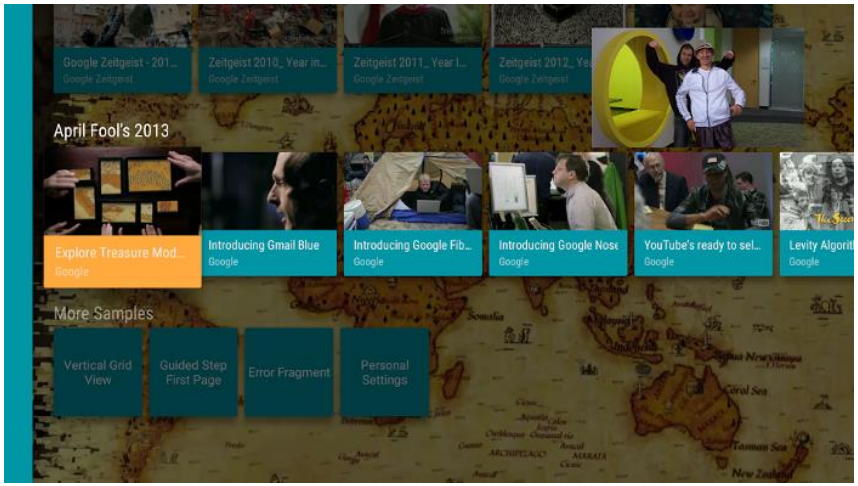
- Fő activitynek kötelező a Banner

```
<activity android:name=".MainActivity"
android:label="@string/app_name"
android:icon="@drawable/app_icon"
android:screenOrientation="landscape"
android:logo="@drawable/app_icon"
android:banner="@drawable/app_banner">
```

Hogyan kezdjünk neki

- TV Launcher Intent Filter

```
<intent-filter>
  <action android:name="android.intent.action.MAIN"/>
   <category android:name=
   "android.intent.category.LEANBACK_LAUNCHER"/>
 </intent-filter>
```



Leanback Support Library

- A beépített alkalmazások fejlesztése során előjött felülteti elemek és funkciók
- Ajánlott minden alkalmazáshoz
- Ne akarjuk újra feltalálni a kereket

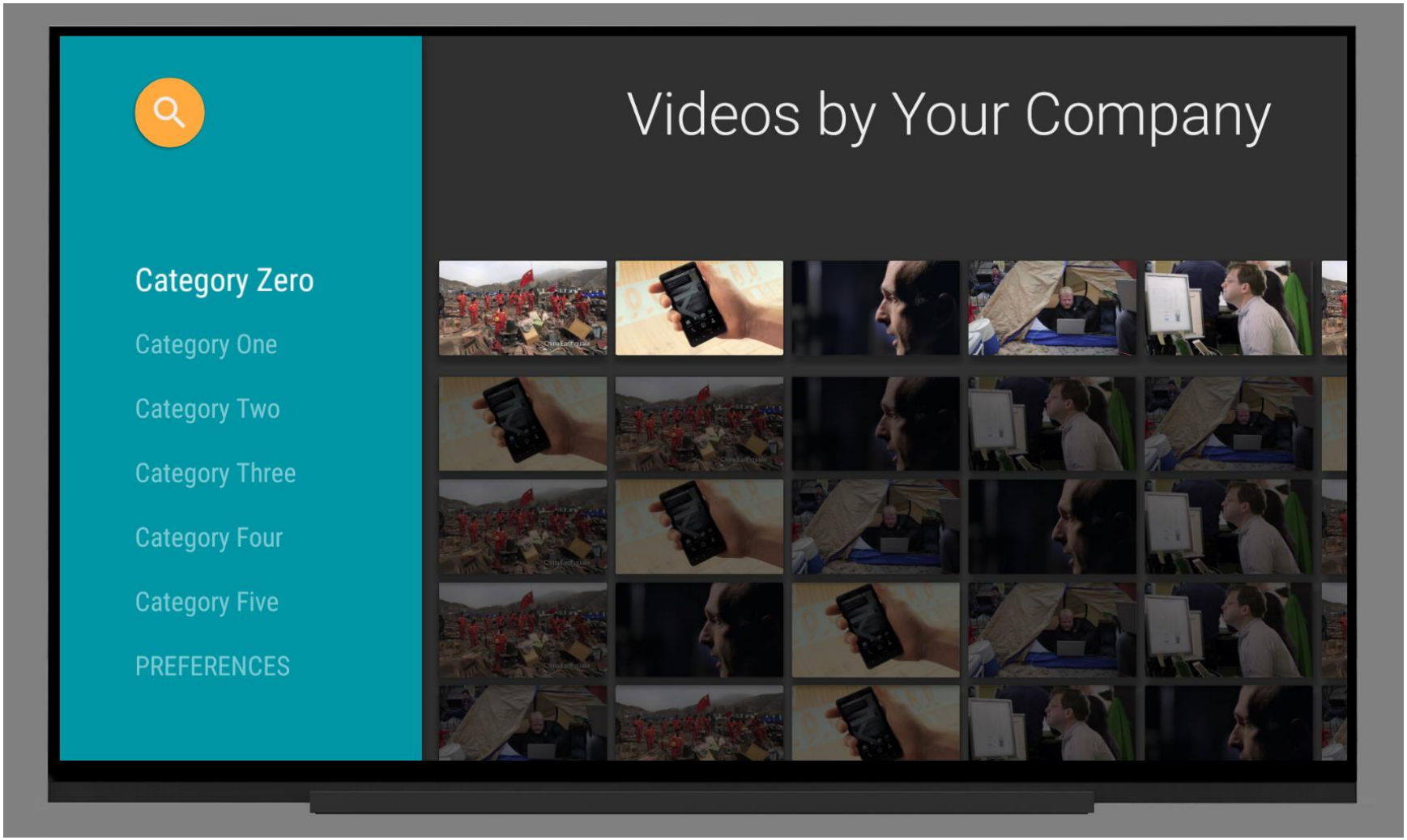
Leanback Support Library

- Beépített D-PAD support
- Látványos és egyértelmű focus kezelés
  - Kiugró kártyák
  - Mozgó listák
  - Színezett menük, header elemek
- Oversampling
  - A TV-s világ öröksége
  - A kép széléről egy kis részt levág a TV
  - Bal Jobb: 48 dp
  - Fent lent: 27 dp
  - Mindig legalább ettől kisebb, de nem pontosan ekkora

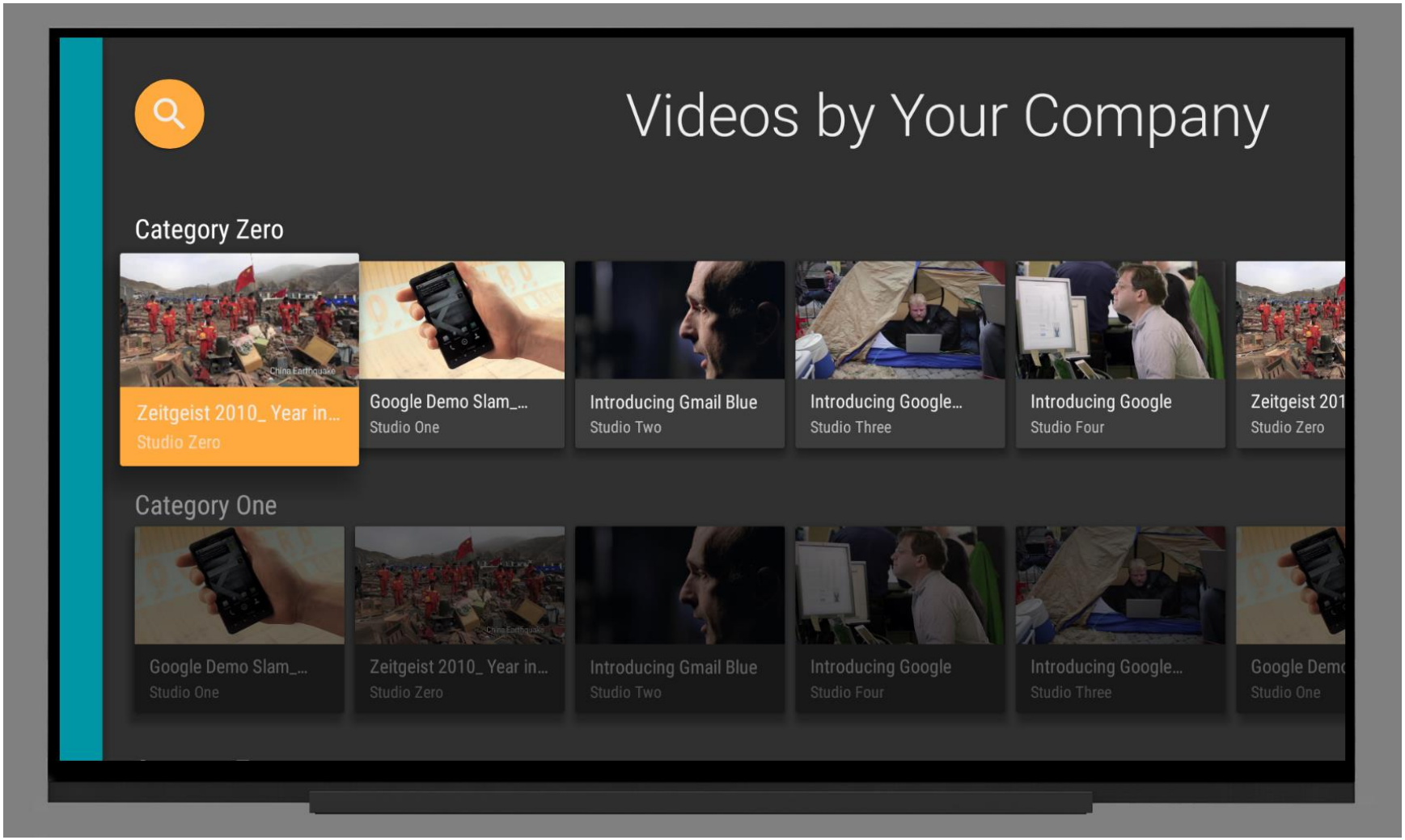
Leanback Support Library

- Beépített Fragmentek
  - Browser Fragment
  - Details Fragment
  - Playback Fragment
- Beépített View-k
  - CardView és Card Presenter
  - Object Adapterek

Browser Fragment



Browser Fragment



Browser Fragment

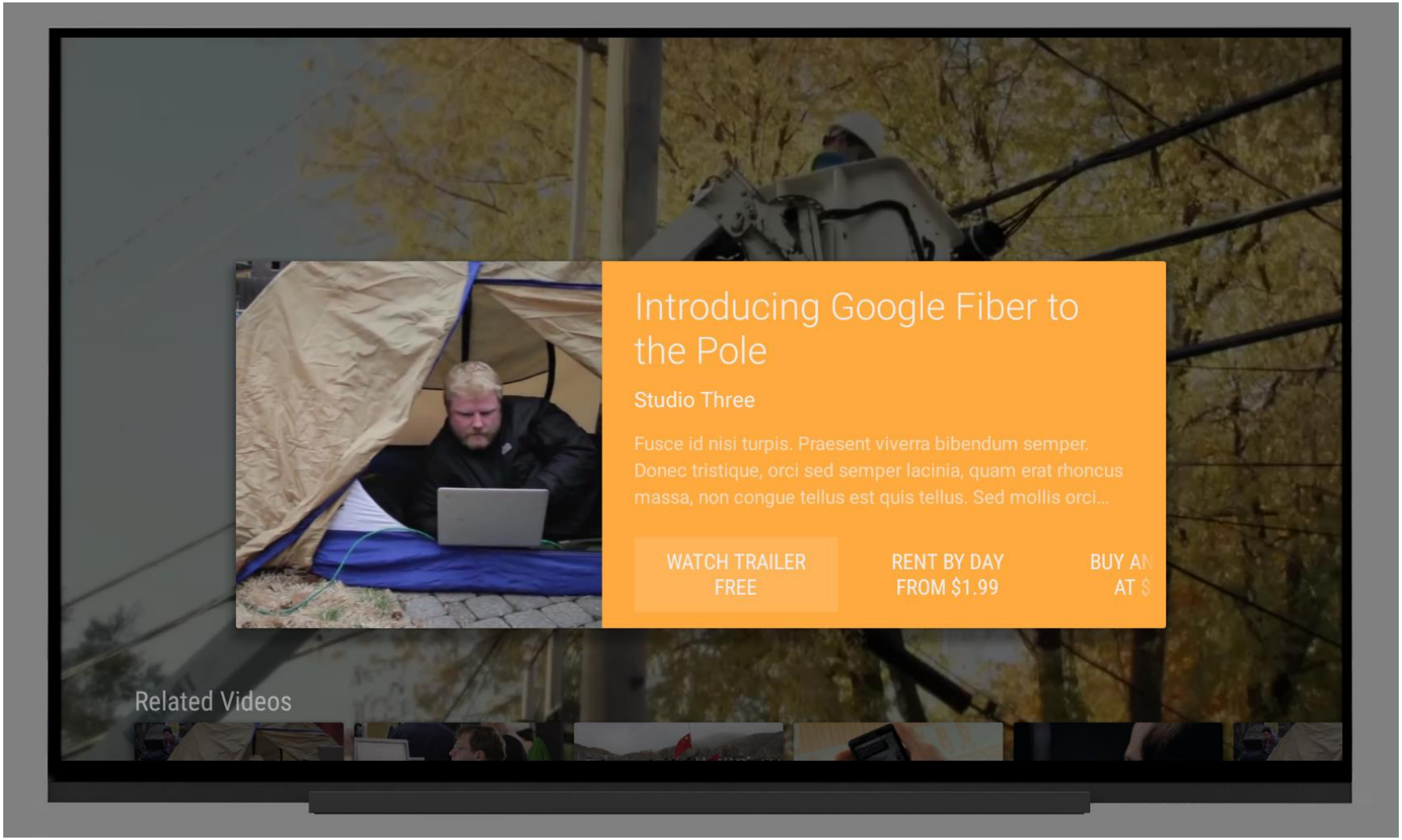
- Branding egyszerűen testre szabható

```
title=getString(R.string.browse_title)
headersState=(HEADERS_ENABLED)
headersTransitionOnBackEnabled= true
brandColor=ContextCompat.getColor(activity,R.color.color)
searchAffordanceColor=...
```

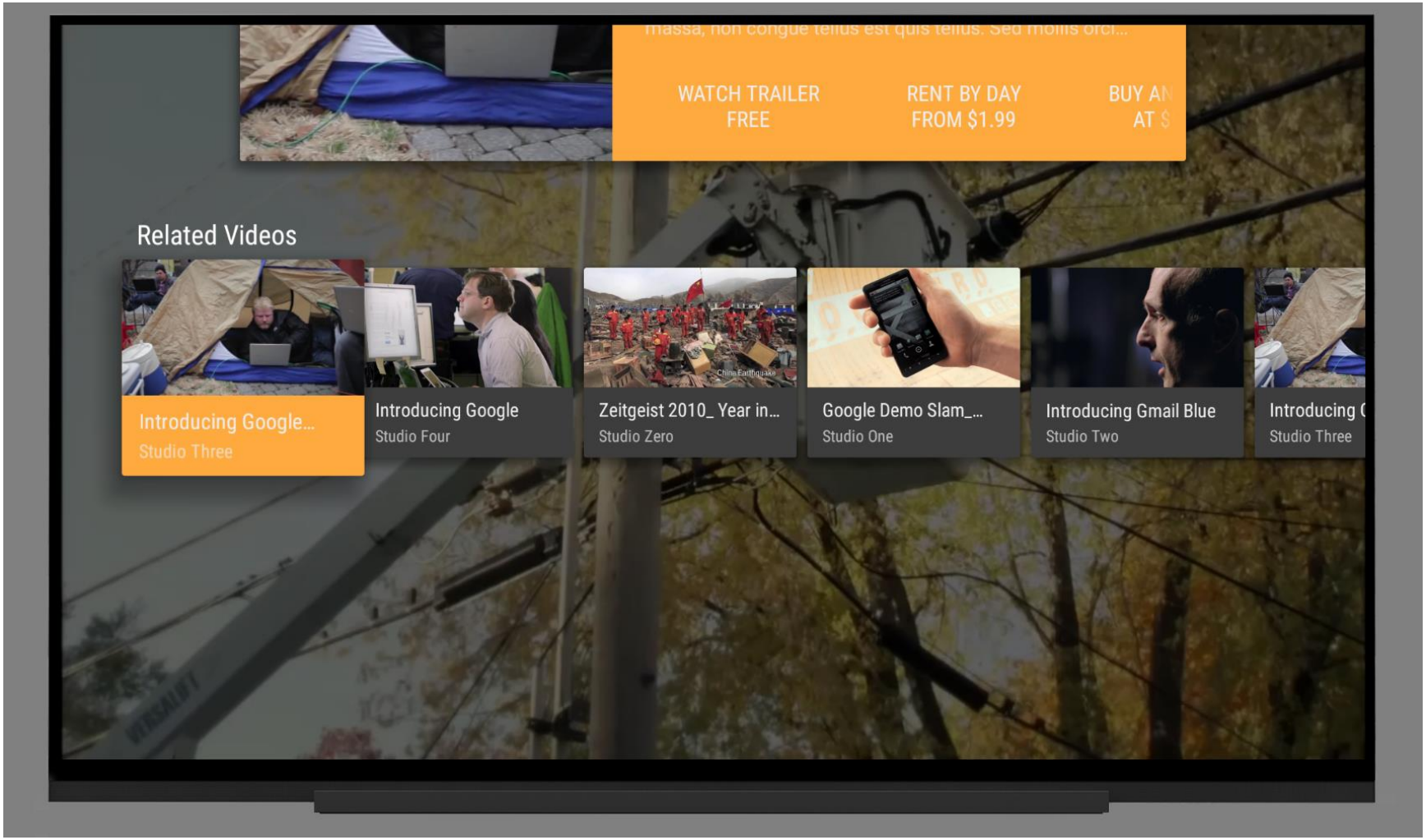
Egyszerű reagálás a beépített eseménekre

> setOnSearchClickedListener{...}
> setOnItemViewSelectedListener{...}
> setOnItemViewClickedListener{...}

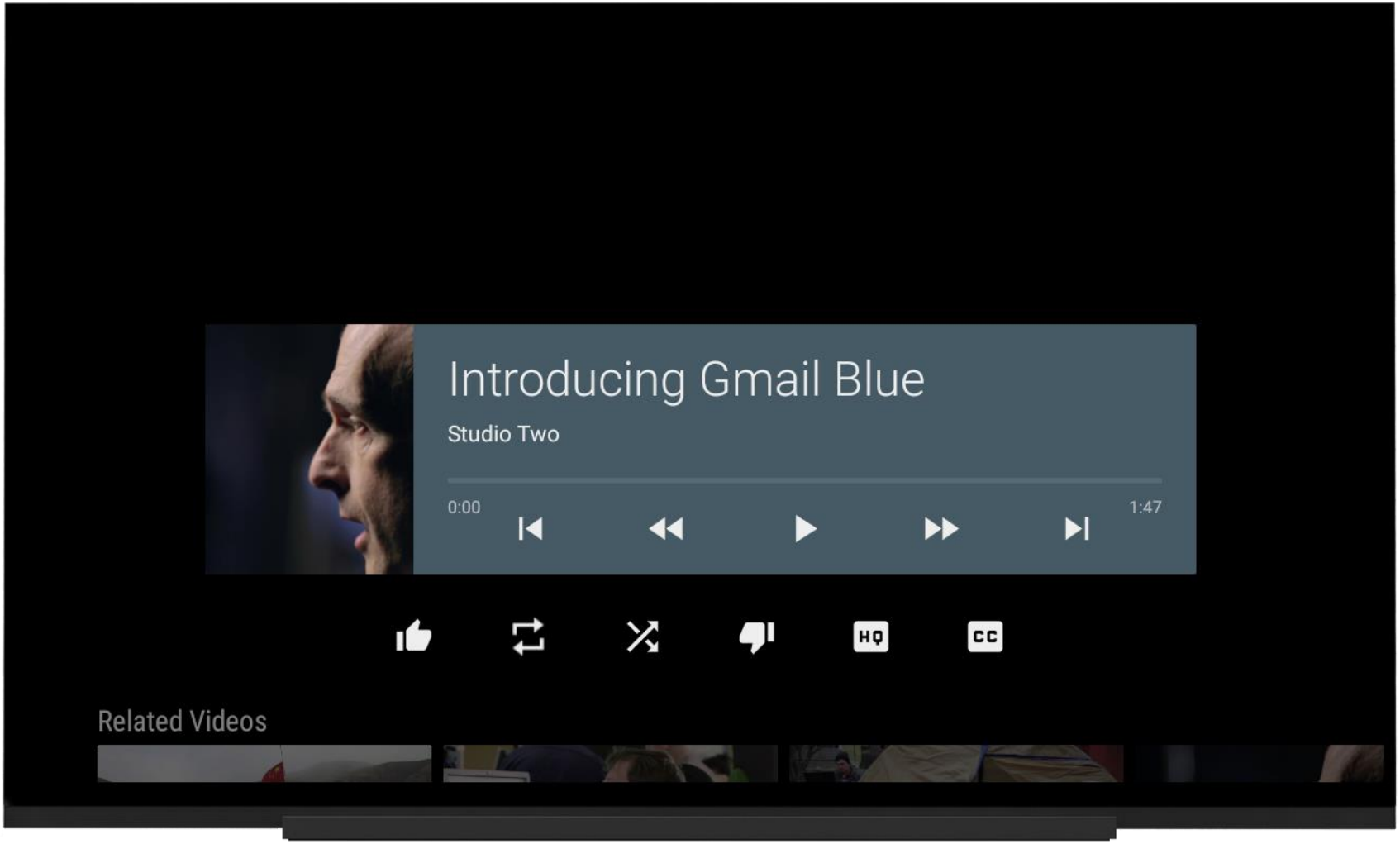
Details Fragment



Details Fragment

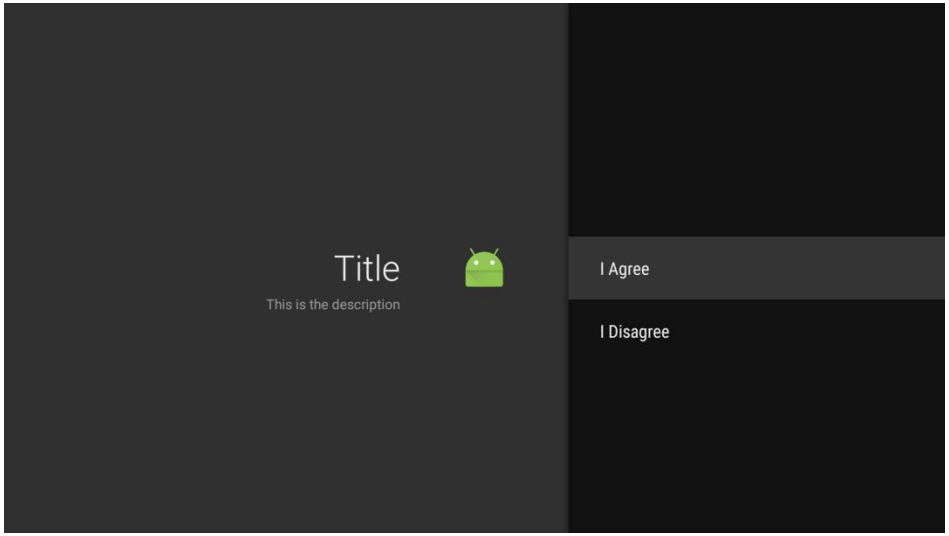


Playback Fragment



Guided Step Fragment

- Dialógusok, beállítások esetén célszerű használni
- Illeszkedik a beépített hibaüzenetek, beállítások stílusába



Search

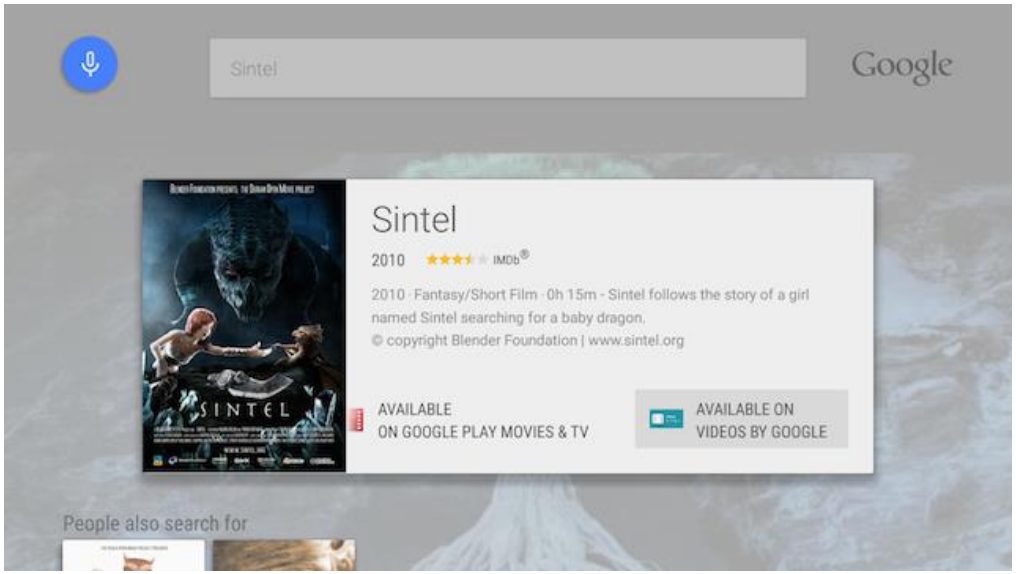
- Keresési lehetőség
  - > Az alkalmazásunkban kereshet a felhasználó
  - > A fő keresőben megjelenhet az alkalmazásunk által ajánlott tartalom
- Definiálni a megfelelő metadata-t
  - > XML, tartalmazza a megfelelő adatforrás leírását

```
<meta-data android:name="android.app.searchable"
        android:resource="@xml/searchable" />
```

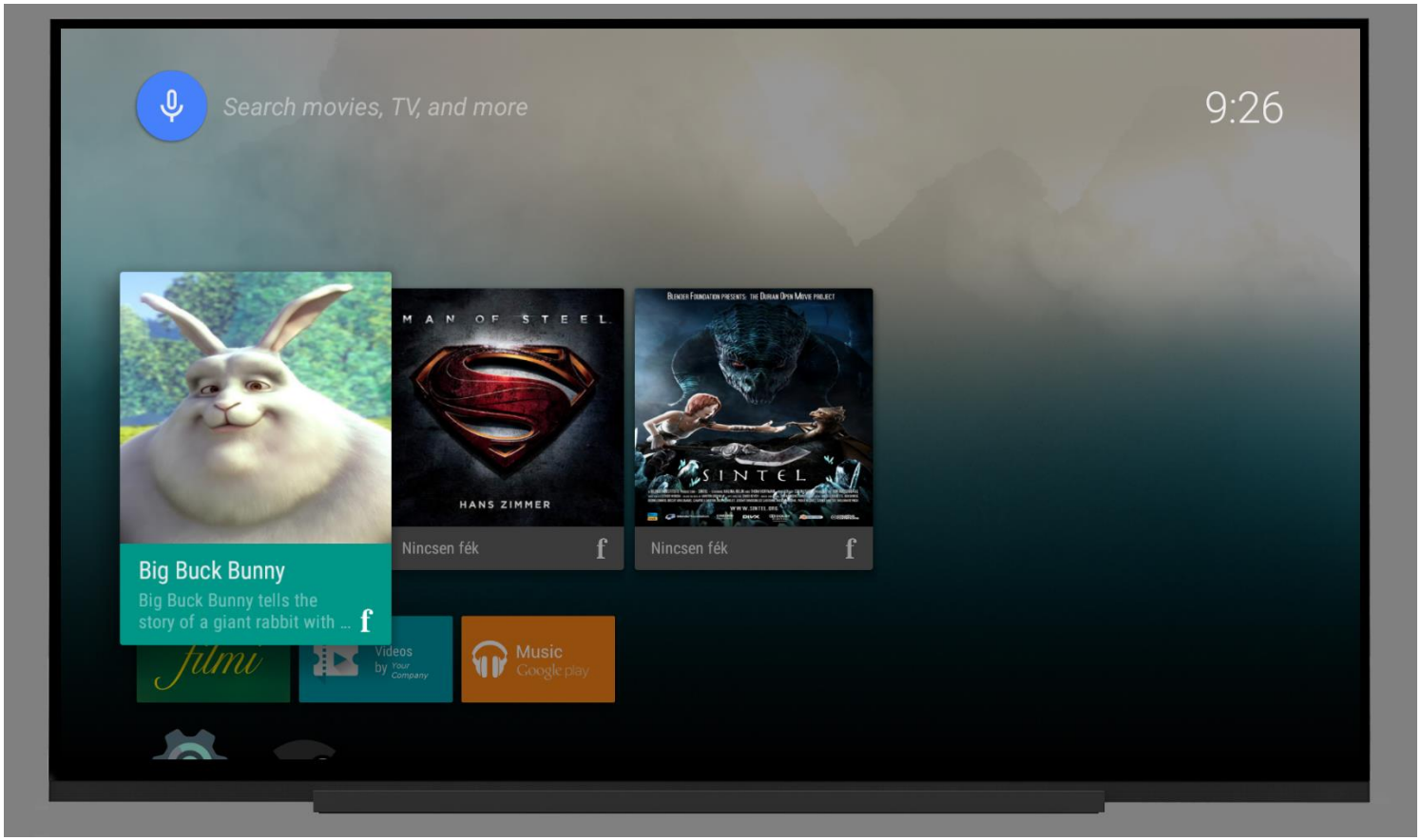
Search

- Beépített támogatás a Content Provider segítségével.
- A keresett elem kiválasztása elkapható intent

```
<action android:name="android.intent.action.SEARCH" />
```



Recommendations



Recommendations

- A főképernyőn tartalom ajánlás
  - > Nem ugyan az mint a notification! Bár hasonlóan érhető el.
- Inicializálás
  - > Indulásra – BOOT_COMPLETED BroadcastReceiver
  - > Adott eseményre – pl. User megvásárol egy tartalmat, elindítja a lejátszást, stb…

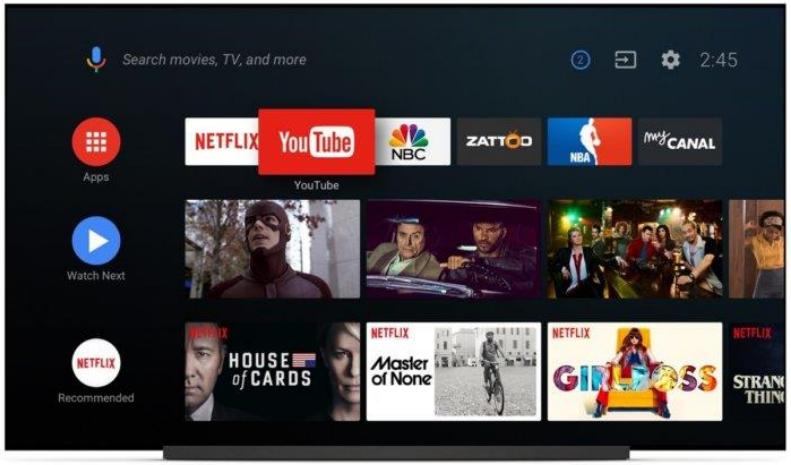
Recommendations

- Célszerű egy IntentService-ben elindítani
- NotificationManager segítségével kezelhető.
- Az egyes elemek: Notification.BigPictureStyle
  - > Csatolható név, leírás, ikon, kép
  - > A kategória: recommendation
  - > PendingIntent csatolható hozzá > pl. Activity indítása

Channels

- Már nem notification
- TV Provider permission szükséges
- Külön könyvtár

```
implementation 'com.android.support:support-tv-provider:28.0.0'
```



Channels

- Channel létrehozása

```
val channel = Channel.Builder().apply {
    // Every channel you create must have the type `TYPE_PREVIEW`
    setType(TvContractCompat.Channels.TYPE_PREVIEW)
    setDisplayName("Channel Name")
    setAppLinkIntentUri(uri)
}.build()

val channelUri = context.contentResolver.insert(TvContractCompat.Channels.CONTENT_URI,
channel.toContentValues())
```

Channels

- Preview készítése

```
val preview = PreviewProgram.Builder().apply {
    setType(TvContractCompat.PreviewPrograms.TYPE_CLIP)
    setTitle("Title")
    setDescription("Program description")
    setPosterArtUri(posterUri)
    setIntentUri(intentUri)
    setInternalProviderId(appProgramId)
    setChannelId(channelId)

}.build()


val programUri = context.contentResolver.insert(
    TvContractCompat.PreviewPrograms.CONTENT_URI, preview.toContentValues()
)
```

WatchNext

- Dedikált channel
- Több alkalmazás is használhatja

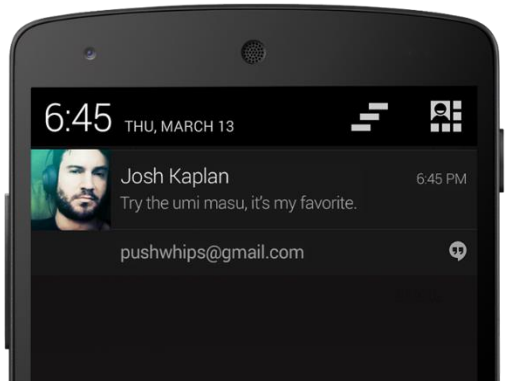
```
val watchNext = WatchNextProgram.Builder().apply {
    setType(TvContractCompat.WatchNextPrograms.TYPE_CLIP)
    setWatchNextType(TvContractCompat.WatchNextPrograms.WATCH_NEXT_TYPE_CONTINUE)
    setLastEngagementTimeUtcMillis(time)
    setTitle("Title")
    setDescription("Program description")
    setPosterArtUri(posterUri)
    setIntentUri(intentUri)
    setInternalProviderId(appProgramId)
}.build()

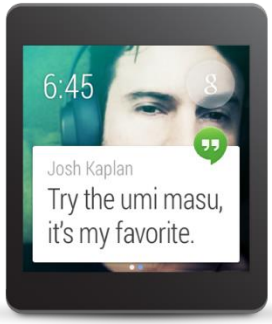

val watchNextProgramUri =
    context.contentResolver.insert(TvContractCompat.WatchNextPrograms.CONTENT_URI,
watchNext.toContentValues())
```

Összefoglalás

- Android, de nem csak telefonra
- A TV nem egy nagy telefon
- Mobilfejlesztői tudás átültethető
- Számos előnyök, de vannak megkötések is
- Használjunk a Leanback Support library-t

Wear OS


6:45
THU, MARCH 13
Josh Kaplan
6:45 PM
Try the umi masu, it's my favorite.
pushwhips@gmail.com

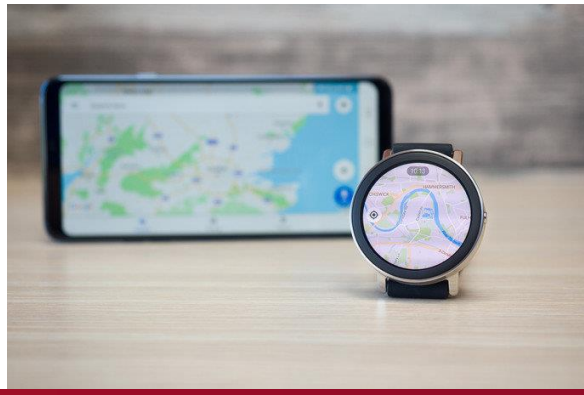

6:45
Josh Kaplan
Try the umi masu, it's my favorite.

Wear OS dióhéjban

- A mobil eszköz kiterjesztése
- Szinkronizált értesítések
- Hangutasítások
- Különálló alkalmazások (Activity, service, listeners)
- Szinkronizált adatok
- Bluetooth LE
- *(Korábban Android Wear)*

Wear OS alkalmzás típusok

- Értesítések, összetett akció lehetőségek
- Wear OS alkalmazások (relatív gazdag API)
- Egyedi felhasználói felület készítési lehetőség
- Adat szinkronizáció és üzenetváltás az óra és a készülék között
- Pozíció meghatározása
- „Watch faces” – óra előlapok



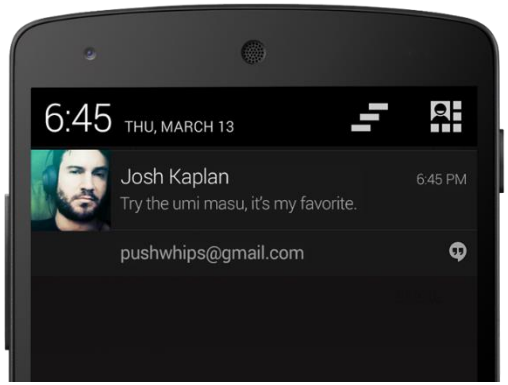
Wear OS emulátor

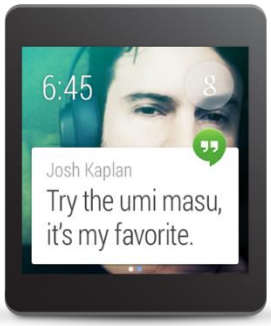
- Wear OS (P)
- Teljes értékű Wear OS emulátor
- Valós eszköz és Wear OS emulátor közti együttműködés
- Szükséges port forward beállítás:

```
adb -d forward tcp:5601 tcp:5601
```

Wear OS értesítések

- Automatikus értesítés megosztás
  - Nincs szükség külön Wear OS alkalmazás fejlesztésére!
- Értesítés -> kártya a „context stream”-en
- Wear OS specifikus akciók
- Hang alapú válaszlehetőség
- Több értesítés kezelése (notification stack)





Egyszerű értesítés

- Automatikus értesítés megosztás a Wear-el:

  `NotificationCompat.Builder`

- PendingIntent megadható:
  - Telefonon/tableten hajtja végre alapértelmezetten
- Notification csatorna létrehozás Android O-tól

```
private val NOTIFICATION_CHANNEL_ID = "my_wear_notifications"
private val NOTIFICATION_CHANNEL_NAME = "My Wear notifications"

@RequiresApi(Build.VERSION_CODES.O)
private fun createNotificationChannel() {
    val channel = NotificationChannel(
        NOTIFICATION_CHANNEL_ID,
        NOTIFICATION_CHANNEL_NAME,
        NotificationManager.IMPORTANCE_DEFAULT)
    val notificationManager =
        getSystemService(Context.NOTIFICATION_SERVICE) as NotificationManager?
    notificationManager?.createNotificationChannel(channel)
}
```

Értesítés megjelenítése

```
private fun showBaiscNotifTime() {
    val notificationId = 1

    val viewPendingIntent = Intent(this, SecondActivity::class.java).let { viewIntent ->
        viewIntent.putExtra("EXTRA_EVENT_ID", 101)
        PendingIntent.getActivity(this, 0, viewIntent, 0)
    }

    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
        createNotificationChannel()
    }

    val notificationBuilder = NotificationCompat.Builder(this, NOTIFICATION_CHANNEL_ID)
        .setSmallIcon(R.drawable.mydroid)
        .setContentTitle("ContentTitle")
        .setContentText(Date(System.currentTimeMillis()).toString())
        .setContentIntent(viewPendingIntent)

    NotificationManagerCompat.from(this).apply {
        notify(notificationId, notificationBuilder.build())
    }
}

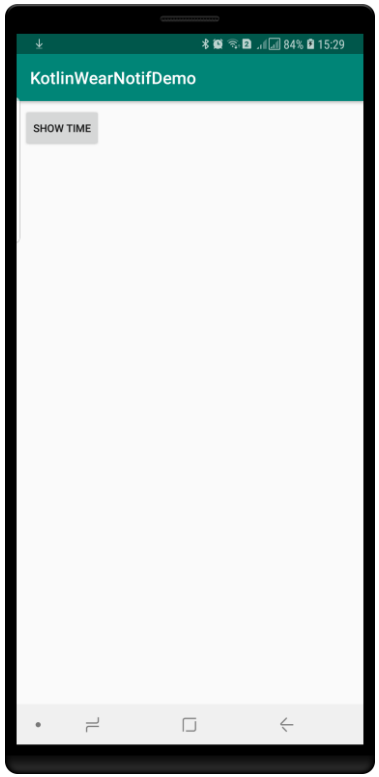
@RequiresApi(Build.VERSION_CODES.O)
private fun createNotificationChannel() {
    val channel = NotificationChannel(
        NOTIFICATION_CHANNEL_ID,
        NOTIFICATION_CHANNEL_NAME,
        NotificationManager.IMPORTANCE_DEFAULT)

    val notificationManager =
        getSystemService(Context.NOTIFICATION_SERVICE) as NotificationManager?

    notificationManager?.createNotificationChannel(channel)
}
```

Gyakoroljunk!

- Készítsünk egy értesítést, mely az aktuális időt megjeleníti az órán ☺

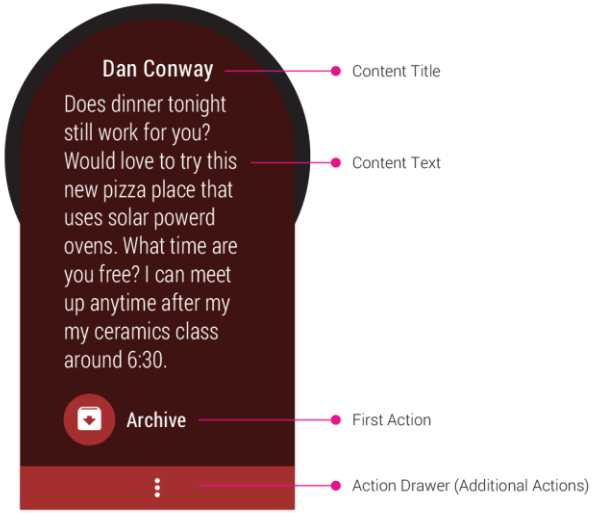


Bővített értesítések (expanded)

- További részletek megjelenítése az értesítésehez
- Akciók/válaszlehetőségek megjelenítése
- Alkalmazás jellegű felhasználói élménye
- Visszajelzés küldése mobil oldalra
  - *RemoteInput*
  - *setChoices()*
- A bővített értesítés kattintásra jelenik meg, ha az alábbiak közöl valamelyik igaz:
  - Az értesítés egy párosított mobil oldalról érkezik
  - Az értesítés nem tartalmaz *ContentIntent*-et
- Bővített értesítés háttér színe a *setColor()*-al állítható

Bővített értesítések - megjelenés

- Extra szöveg megjelenítése:
  - > BigTextStyle
- Extra kép megjelenítése
  - > BigPictureStyle
  - > addPage()-el több kép is adható
- Elsődleges akció:
  - > setContentAction()



Gyakoroljunk!

- Készítsünk egy értesítést, mely egy rendelést jelképez és kattintásra megtekinthető az átvétel helye.
  - Figyeljük meg az értesítés megjelenési módját a telefonon és az órán.

Megoldás

```
private fun showNotifOrderMap() {
    val notificationId = 1

    val mapIntent = Intent(Intent.ACTION_VIEW)
    val mapPendingIntent = Intent(Intent.ACTION_VIEW).let { mapIntent ->
        //mapIntent.data = Uri.parse("geo:0,0?q=Budapest")
        mapIntent.data = Uri.parse("waze://?q=BME&navigate=yes")
        PendingIntent.getActivity(this, 0, mapIntent, 0)
    }

    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
        createNotificationChannel()
    }

    val bigStyle = NotificationCompat.BigTextStyle()
    bigStyle.bigText("You have a new order from our company.
      Please decide how you would like to pick up the item.")

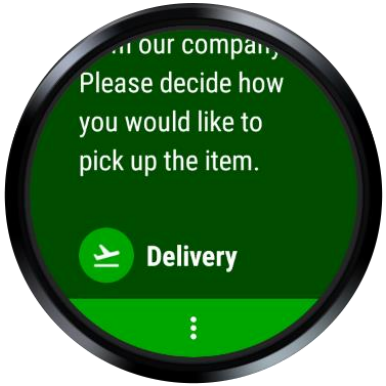
    val notificationBuilder = NotificationCompat.Builder(this, NOTIFICATION_CHANNEL_ID)
        .setSmallIcon(R.drawable.mydroid)
        .setLargeIcon(BitmapFactory.decodeResource(resources, R.drawable.truck))
        .setContentTitle("Order details")
        .setContentText(Date(System.currentTimeMillis()).toString())
        .addAction(R.drawable.ic_flight_takeoff, "Show store", mapPendingIntent)
        .setColor(Color.GREEN)
        .setStyle(bigStyle)

    NotificationManagerCompat.from(this).apply {
        notify(notificationId, notificationBuilder.build())
    }
}
```

További akciógombok

- *Builder addAction(...)* függvényével további akciók definiálhatók
  - A készüléken egy újabb akciógomb az értesítéshez
  - Wear-en egy új akció jobbra lapozáskor
- Lehetőség csak Wear akciók megadására:

```
WearableExtender.addAction()
```



Válaszlehetőségek értesítésre

- Egyszerű válasz lehetőség
- Hang alapú válasz (billentyűzet nincs!)
- Előre definiált válaszok megadhatók
- A készülék oldalon a „válasz” intent-el kiolvasható az eredmény

Gyakoroljunk

- Készítsünk egy rendelés értesítést, melyben megadható, hogy kívánunk-e házhozszállítást. Az órán választott döntést mobil telefonon egy újabb Activity kapja meg.

Válasz kiolvasása

```
class SecondActivity : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_second)

        val remoteResult = getRemoteMessageText(intent)
        if (remoteResult != null) {
            Toast.makeText(applicationContext, remoteResult, Toast.LENGTH_LONG).show()
            tvOrder.text = remoteResult
        }
    }

    private fun getRemoteMessageText(intent: Intent): String {
        val remoteInput = RemoteInput.getResultsFromIntent(intent)
        return if (remoteInput != null) {
            remoteInput.getCharSequence(MainActivity.EXTRA_VOICE_REPLY).toString()
        } else ""
    }
}
```

Több értesítés kezelése

- További információk megjelenítése esetén egy vagy több további oldal adható az értesítéshez
- Gyakorlatilag ez új lapok számára új *Notification* objektumot kell létrehozni
- *WearableExtender addPage(...)* vagy *addPages(...)* függvénye

Házi feladat!

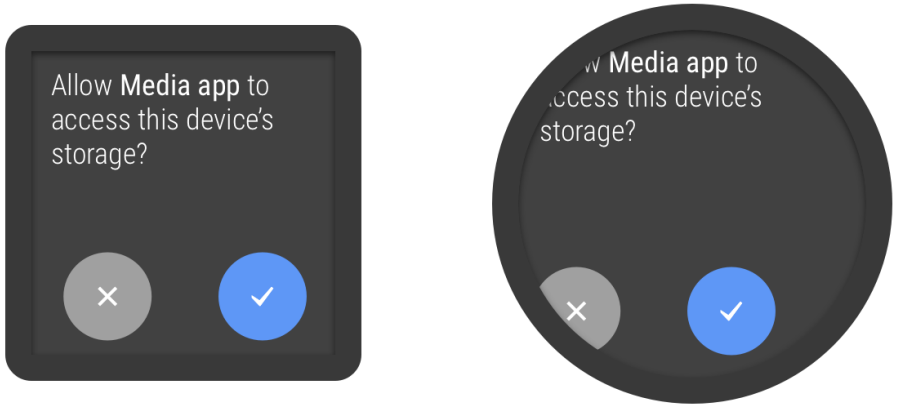
- Készítsünk egy alkalmazást, mely értesíti a Wear-t kimenő hívás esetén!
- Egészítsük ki a megoldást wear specifikus akciókkal!
- Adjunk hozzá hang alapú visszajelzési lehetőséget!

Wear design alapok

- Alkalmazás típusok:
  - Értesítés
  - Watch Face
  - Standalone alkalmazás
- https://developer.android.com/design/wear/index.html
- https://developer.android.com/training/wearables/ui/

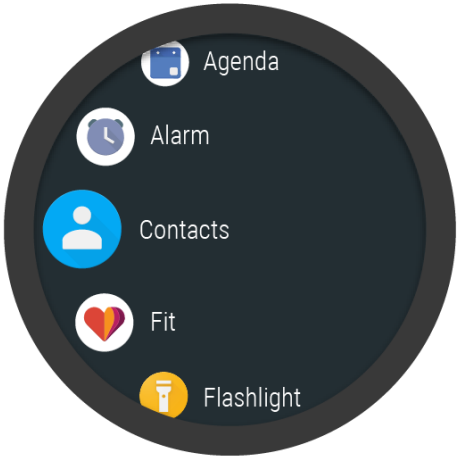
Felületi újdonságok

- WatchViewStub
  - Külön kerek, külön szögletes felület
  - Futásidőben dől el
- BoxInsetLayout
  - Négyzet alakú terület kör alakú órán is



Felületi újdonságok

- Listák kezelése
  - WearableRecyclerView
  - „curved” layout
  - Körkörös scroll lehetőség

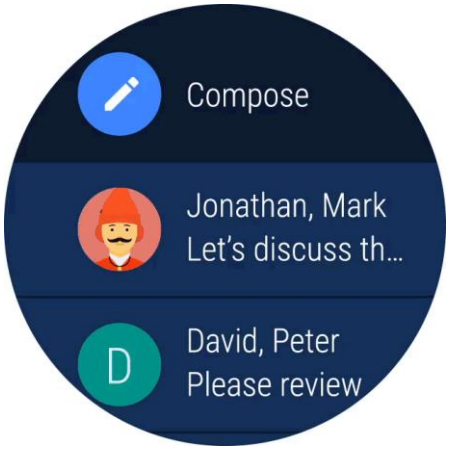



```
<android.support.wear.widget.WearableRecyclerView
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/recycler_launcher_view"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:scrollbars="vertical" />
```

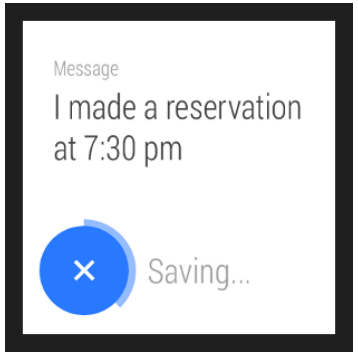
```
mWearableRecyclerView.apply {
    isCircularScrollingGestureEnabled = true
    bezelFraction = 0.5f
    scrollDegreesPerScreen = 90f
}
```

Egyedi óra elemek

- WearableDrawerLayout
- WearableActionDrawerView



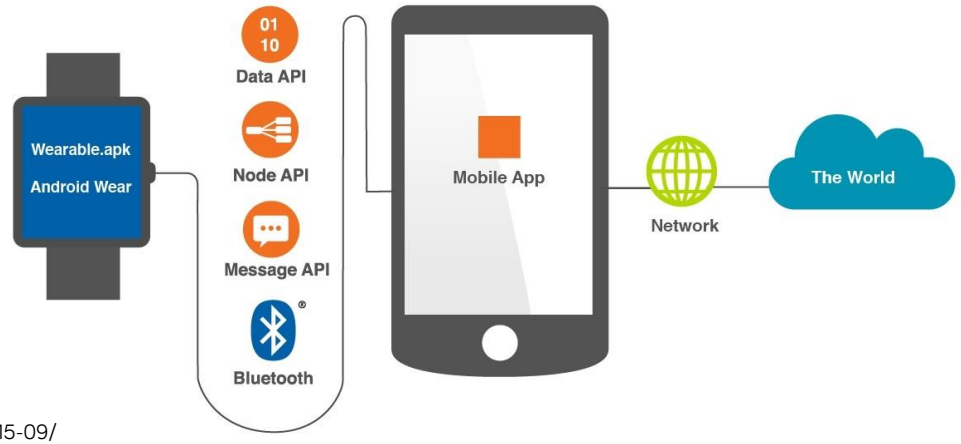
- Confirmations
  - Beépített időzítő
  - API által adott listener
  - ```
    Intent intent = new Intent(this, ConfirmationActivity.class);
    intent.putExtra(ConfirmationActivity.EXTRA_ANIMATION_TYPE,
                    ConfirmationActivity.SUCCESS_ANIMATION);
    intent.putExtra(ConfirmationActivity.EXTRA_MESSAGE,
                    getString(R.string.msg_sent));
    startActivity(intent);
    ```




    ```

Kommunikáció

- Wearable Data Layer API
- Google Play Services része
- Bluetooth kommunikáció
- Google ajánlás szerint ez használandó elsődlegesen
- Többféle adat objektum támogatása különböző célokra
- Wear OS eszközön és Wear AVD-n is működik



Forrás: https://www.electronicsweekly.com/blogs/gadget-master/wireless/build-android-wear-app-2015-09/

Wearable Data Layer API objektum típusok

- DataItem
  - Egyszerű adatok tárolása
  - Automatikus szinkronizálás az eszközök között
  - DataApi-n keresztül használható
- Message
  - Egyszerű utasítások küldése (lejátszás, megállítás)
  - Nincs szinkronizálás, csak csatlakoztatott állapotban küldi el az üzenetet
  - MessageApi-n keresztül használható
- Asset
  - Bináris adatok, pl képek küldése
  - DataItem-hez csatolható
  - Automatikus cache és átvitel
- Channel
  - Nagy tartalmak átvitele (zene, video, stb.)
  - Megbízható csatorna
  - Stream adat (mikrofonról, zene a hálózatról, stb.)
- WearableListenerService
  - Broadcast figyeléséhez
- DataListener
  - Előtérben történő események figyeléséhez

MessageClient - Mobile

```
class MainActivity : AppCompatActivity(), DataClient.OnDataChangedListener,
  MessageClient.OnMessageReceivedListener {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        btnMessage.setOnClickListener { sendMessage() }
    }
    override fun onResume() {
        super.onResume()
        Wearable.getMessageClient(this).addListener(this);
    }
    override fun onPause() {
        super.onPause()
        Wearable.getMessageClient(this).removeListener(this)
    }

    override fun onDataChanged(p0: DataEventBuffer) {
        Toast.makeText(this, "message received", Toast.LENGTH_LONG).show()
    }

    override fun onMessageReceived(p0: MessageEvent) {
        Toast.makeText(this, "message received", Toast.LENGTH_LONG).show()
    }
    private fun sendMessage() {
        Thread {
            val nodeListTask = Wearable.getNodeClient(applicationContext).connectedNodes
            val nodes = Tasks.await(nodeListTask)
            runOnUiThread {
                nodes.forEach {
                    Toast.makeText(this@MainActivity, it.displayName, Toast.LENGTH_LONG).show()
                    Wearable.getMessageClient(this).sendMessage(it.id, "/msg", "Data".toByteArray())
                }
            }
        }.start()
    }
}
```

MessageClient - Wear

```
class MainActivity : WearableActivity(), DataClient.OnDataChangedListener,
  MessageClient.OnMessageReceivedListener  {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        …
    }

    override fun onResume() {
        super.onResume()
        Wearable.getMessageClient(this).addListener(this)
    }

    override fun onPause() {
        super.onPause()
        Wearable.getMessageClient(this).removeListener(this)
    }

    override fun onDataChanged(p0: DataEventBuffer) {
        Toast.makeText(this, "message received", Toast.LENGTH_LONG).show()
    }

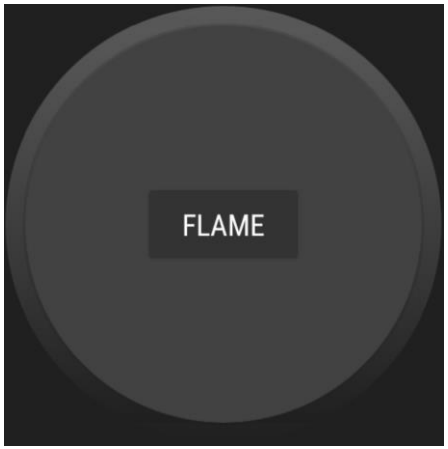
    override fun onMessageReceived(p0: MessageEvent) {
        Toast.makeText(this, "message received "+p0.path, Toast.LENGTH_LONG).show()
    }
}
```

Gyakorljunk!

- Készítsünk standalone wear alklamazást, mely oda-vissza kommunikációt bonyolít le a mobil alkalmazással

Gyakoroljunk

- Készítsünk egy „kandalló” standalone Wear OS alkalmazást



Köszönöm a figyelmet!

peter.ekler@aut.bme.hu

AutSoft