Android TV és Wear OS

Ekler Péter

BME VIK AUT, AutSoft

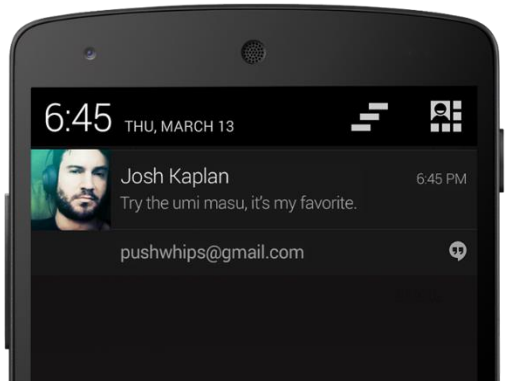
peter.ekler@aut.bme.hu

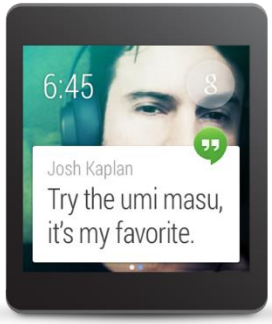
Tematika

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)

Android TV – *külön diasor*

Wear OS


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

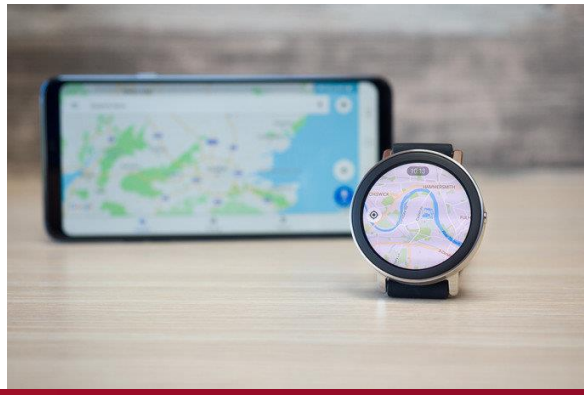

6:45
Josh Kaplan
Try the umi masu, it's my favorite.

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



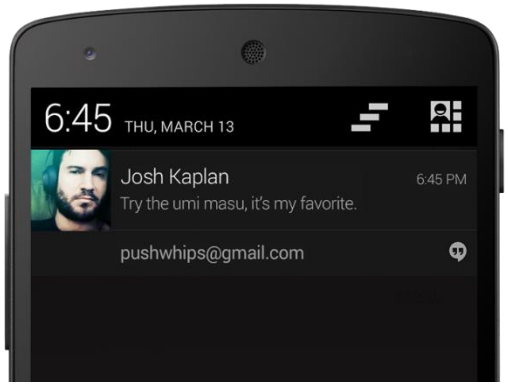
Wear OS emulátor

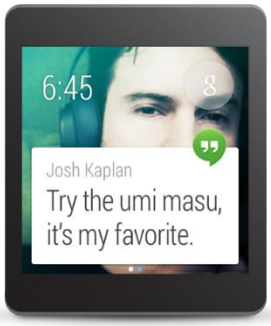
- Wear OS (P)
- Teljes értékű Wear OS emulátor
- Valós eszköz és Wear OS emulátor közti együttműködés
- Szükséges port forward beállítás:

```
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())
    }
}

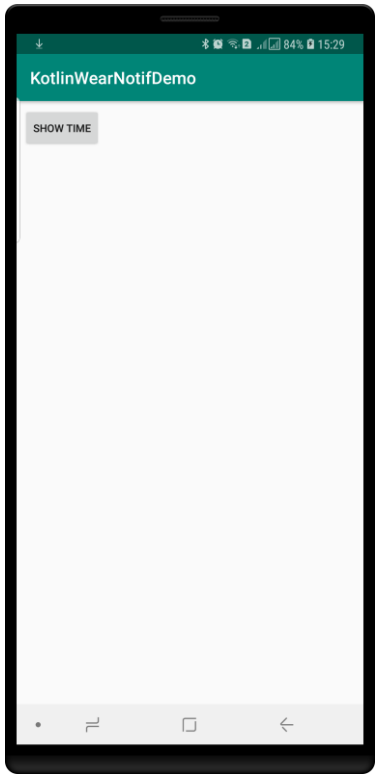
@RequiresApi(Build.VERSION_CODES.O)
private fun createNotificationChannel() {
    val channel = NotificationChannel(
        NOTIFICATION_CHANNEL_ID,
        NOTIFICATION_CHANNEL_NAME,
        NotificationManager.IMPORTANCE_DEFAULT)

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

    notificationManager?.createNotificationChannel(channel)
}
```

Gyakoroljunk!

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

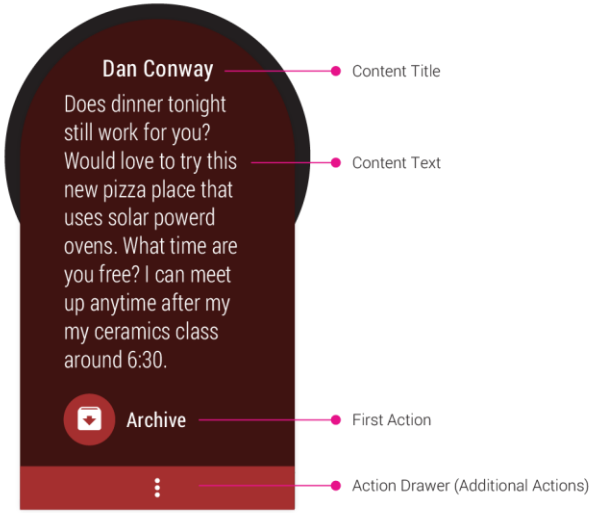


Bővített értesítések (expanded)

- 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.")

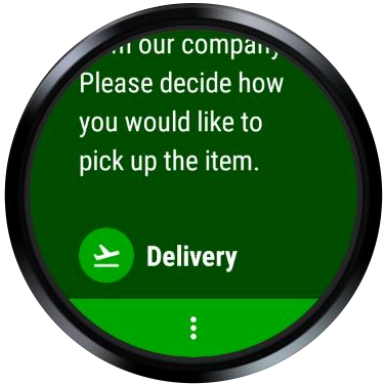
    val notificationBuilder = NotificationCompat.Builder(this, NOTIFICATION_CHANNEL_ID)
        .setSmallIcon(R.drawable.mydroid)
        .setLargeIcon(BitmapFactory.decodeResource(resources, R.drawable.truck))
        .setContentTitle("Order details")
        .setContentText(Date(System.currentTimeMillis()).toString())
        .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!

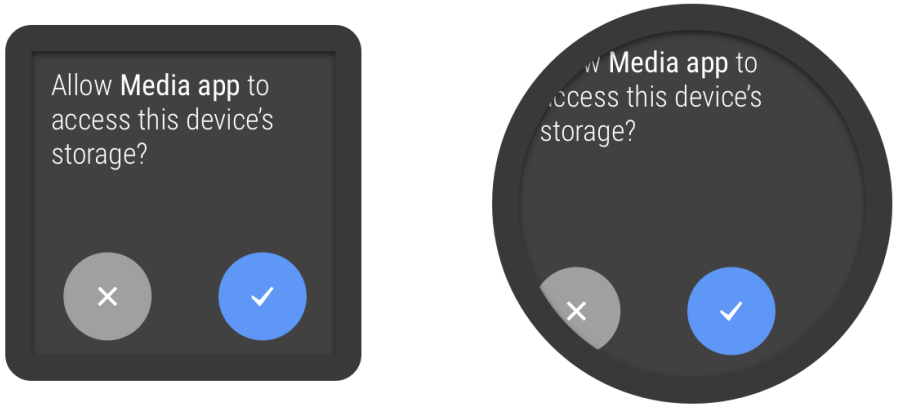
- Készítsünk egy alkalmazást, mely értesíti a Wear-t kimenő hívás esetén!
- Egészítsük ki a megoldást wear specifikus akciókkal!
- Adjunk hozzá hang alapú visszajelzési lehetőséget!

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/

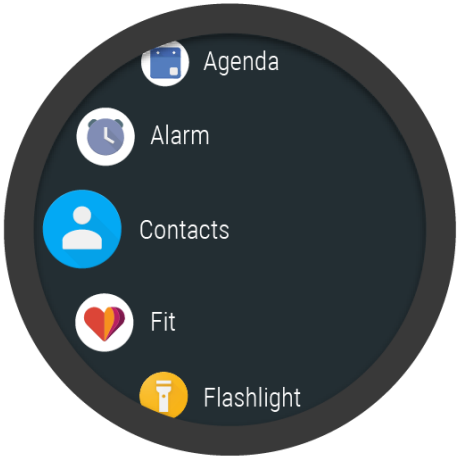
Felületi újdonságok

- 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

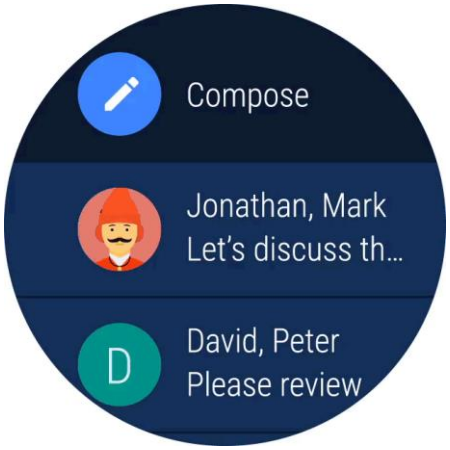



```
<android.support.wear.widget.WearableRecyclerView
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/recycler_launcher_view"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:scrollbars="vertical"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

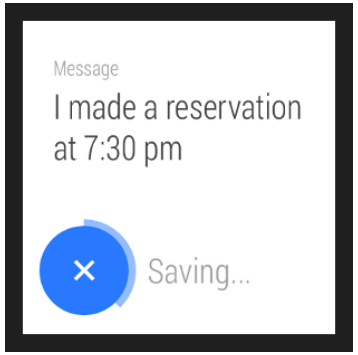
```
mWearableRecyclerView.apply {
    isCircularScrollingGestureEnabled = true
    bezelFraction = 0.5f
    scrollDegreesPerScreen = 90f
}
```

Egyedi óra elemek

- WearableDrawerLayout
- WearableActionDrawerView



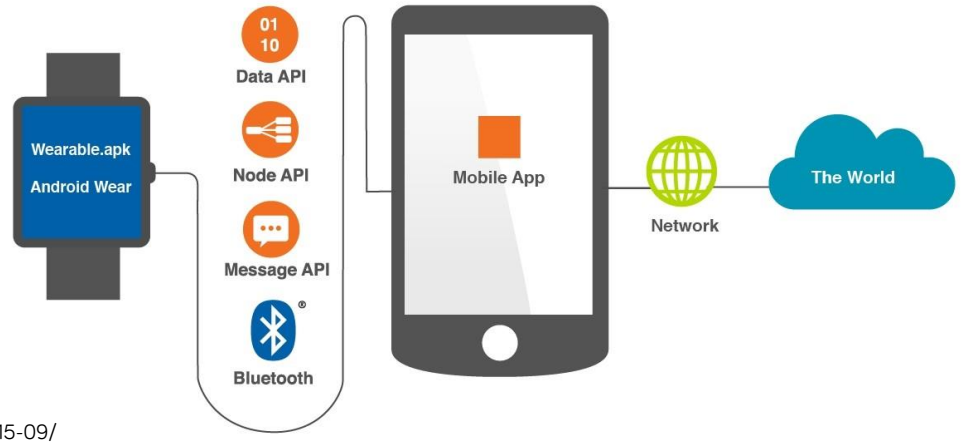
- Confirmations
  - Beépített időzítő
  - API által adott listener
  - ```
    Intent intent = new Intent(this, ConfirmationActivity.class);
    intent.putExtra(ConfirmationActivity.EXTRA_ANIMATION_TYPE,
                    ConfirmationActivity.SUCCESS_ANIMATION);
    intent.putExtra(ConfirmationActivity.EXTRA_MESSAGE,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

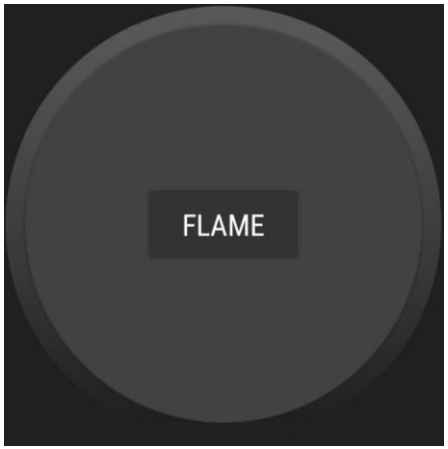
    override fun onMessageReceived(p0: MessageEvent) {
        Toast.makeText(this, "message received "+p0.path, Toast.LENGTH_LONG).show()
    }
}
```

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