

ANDROID ALKALMAZÁSFEJLESZTÉS

Felhasználói felület megismerése
Különböző felbontások támogatása



Sicz-Mesziár János

sicz-mesziar.janos@nik.uni-obuda.hu

Mezei József

mezei.jozsef@nik.uni-obuda.hu

2018. szeptember 24.

Layouts

Ősosztály: [ViewGroup](#)

Olyan [tárolók](#), melynek gyermeke lehet:

- View / Widget
- Tároló / Layout

Deprecated

TableLayout

AbsoluteLayout

TabLayout

Gallery

Speciálisak

ScrollView

HorizontalScrollView

ListView

GridView

ViewPager

ViewFlipper

...



[LinearLayout](#)

- UI elemek egymás után.
- Horizontal vagy Vertical.
- Méretezés arányokkal.



[RelativeLayout](#)

- Egymáshoz képest adhatjuk meg pozíciót.
- Leghatékonyabb layout.



[FrameLayout](#)

- Minden gyermek a bal felső sarokhoz igazodik



[GridLayout](#)

- Mozaik szerű elrendezés
- **Android 4.0-től**
- != GridView!
- [Bővebben itt!](#)

Modern layout: **ConstraintLayout**

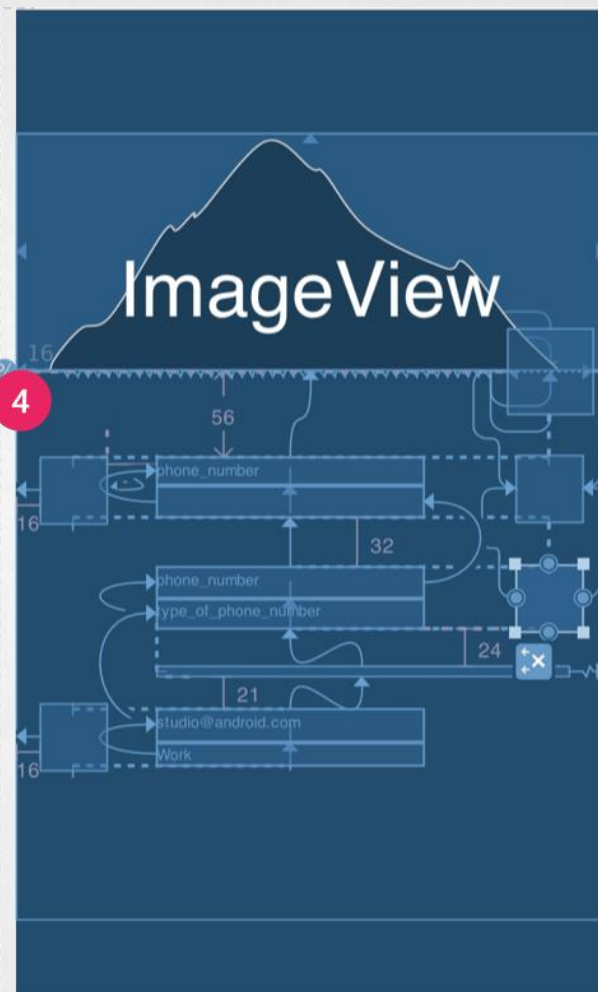
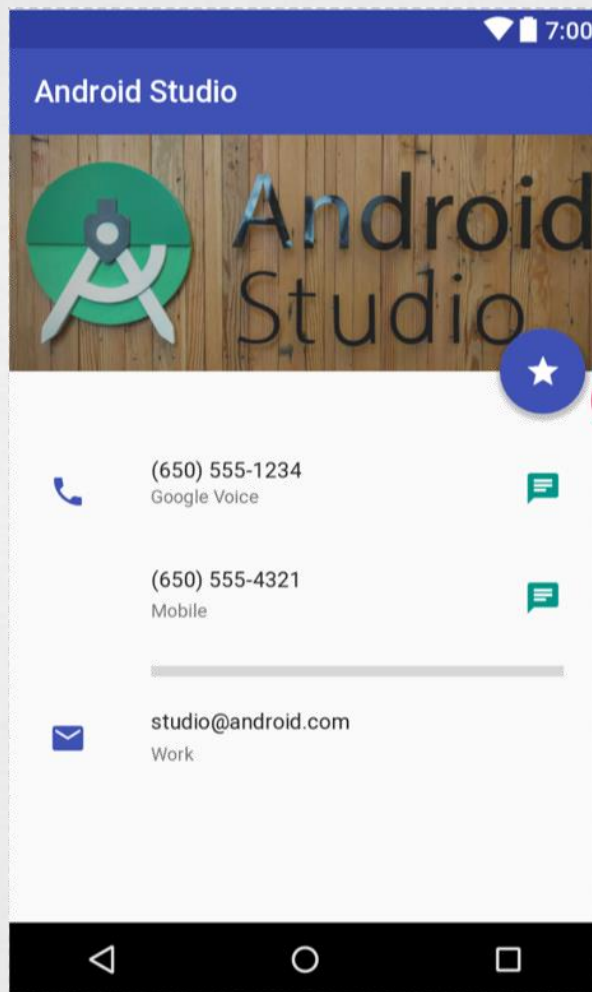
- Hasonló működésű, mint a RelativeLayout
- Gyermekük egymáshoz képest fejezik ki pozíciójukat
- Android 2.3+ támogatott, legalább Android Studio 2.3
- Minden View-nak két constraint kell:
horizontal, vertical
- Multiple constraint támogatott

Bővebben:

<https://developer.android.com/training/constraint-layout/index.html>

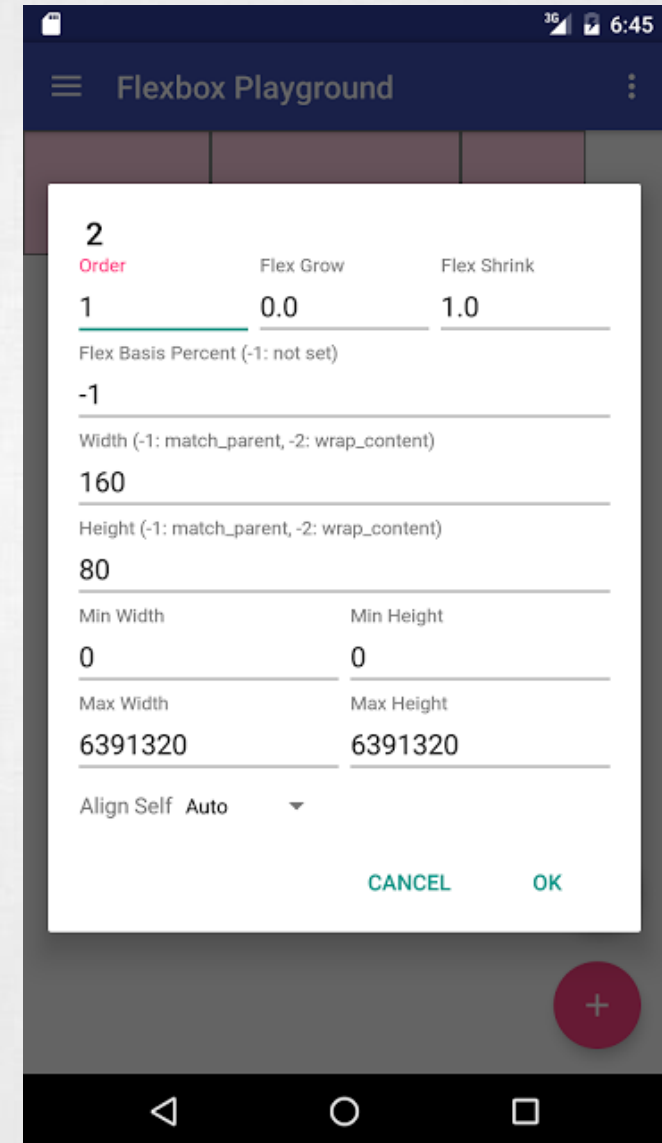
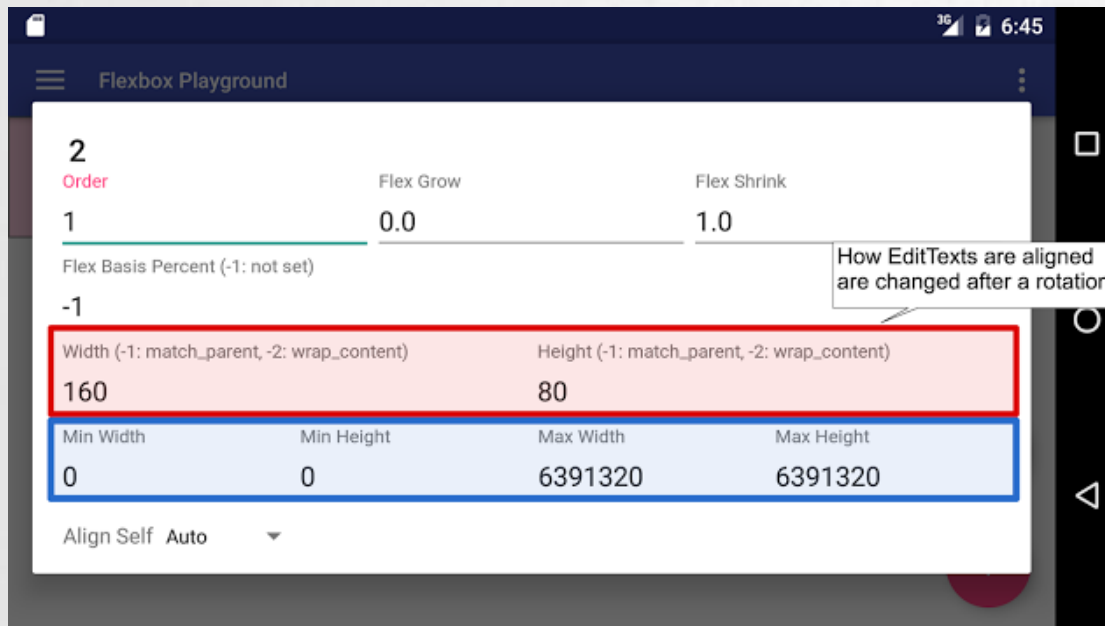
Layout Editor támogatás:

<https://developer.android.com/studio/write/layout-editor.html>

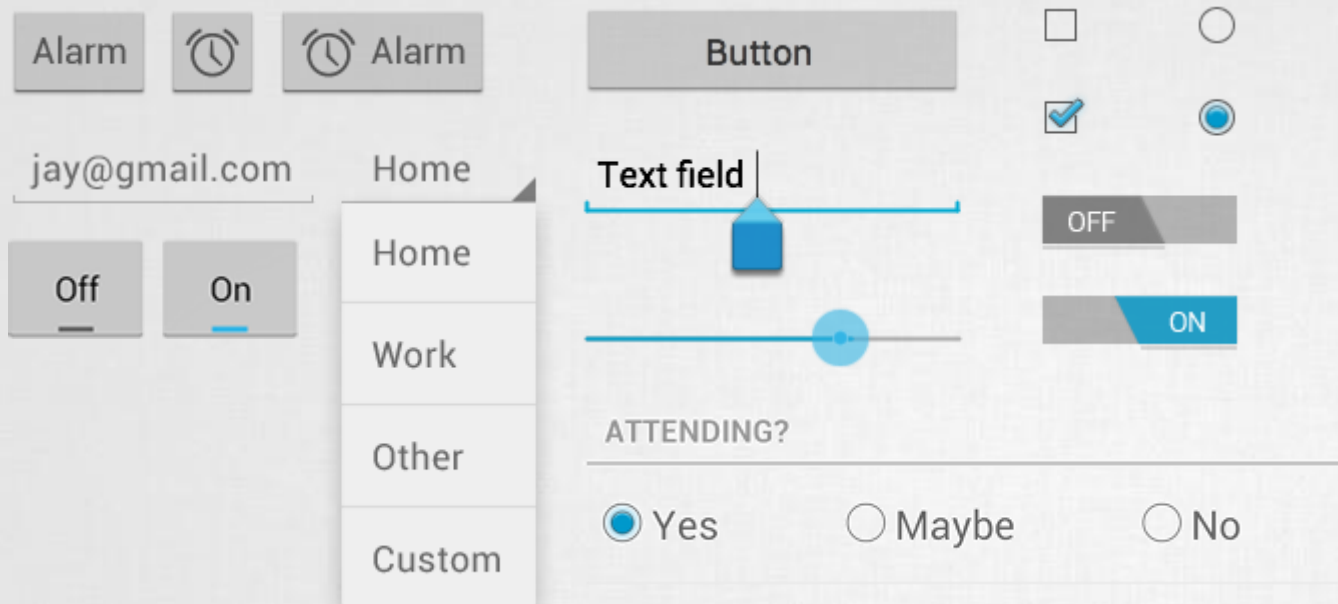


Modern layout: FlexboxLayout

- CSS Flexible Box Layout Module működésének megfelelő
- Jól használható rezponzív felület kialakítására
- Tekinthetjük feltűrbózott LinearLayout-nak is
- Box wrap: nincs elég hely a View-nak új sorba kerül
- Split screen miatt különösen hasznos
- [Android Developers Blog](#) [Open Source](#)



View / Widget



Button, TextView,
EditText, Checkbox,
Radio button, Toogle
button, Spinner,
ImageView,

```
<Button
```

```
    android:id="@+id/button1"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="@string/txt"
```

```
    ...
```

```
>/>
```

Egyedi azonosító, hivatkozás:

- Java: `R.id.button1`
- XML: `@id/button1`

Méret - szélesség, magasság:

- `MATCH_PARENT`
- `WRAP_CONTENT`
- `[SIZE] [dp|px]`

Resource elérés, `res/strings.xml`:

- `<item name="txt">Gomb</item>`

Dialogs

- Ősosztály: Dialog
- Az aktuális Activity / Fragment előtt jelenik meg
- Beépített gombok:
 - PositiveButton
 - NegativeButton
 - NeutralButton

```
AlertDialog.Builder(activity)
```

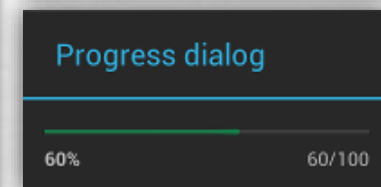
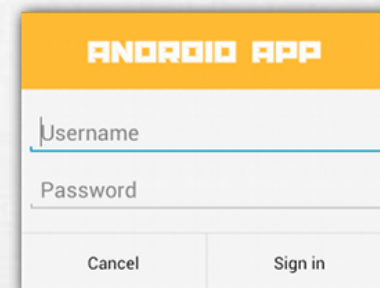
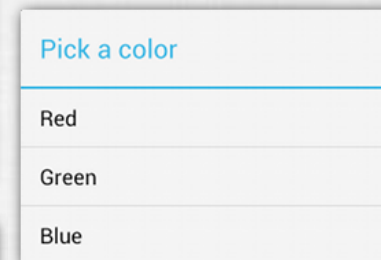
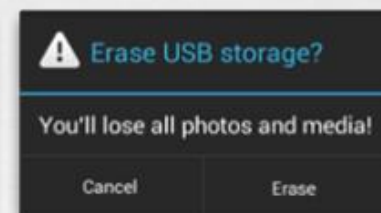
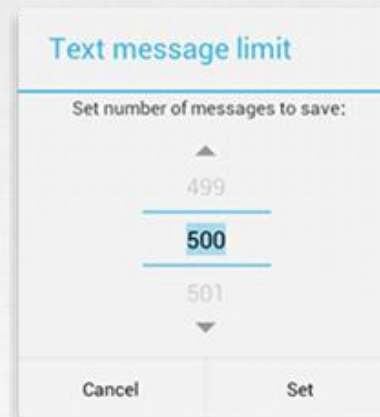
```
    .setMessage(R.string.dialog_message)
```

```
    .setTitle(R.string.dialog_title)
```

```
    .setPositive("OK") {
```

```
        // TODO: click on positive button of dialog
```

```
    }.show()
```

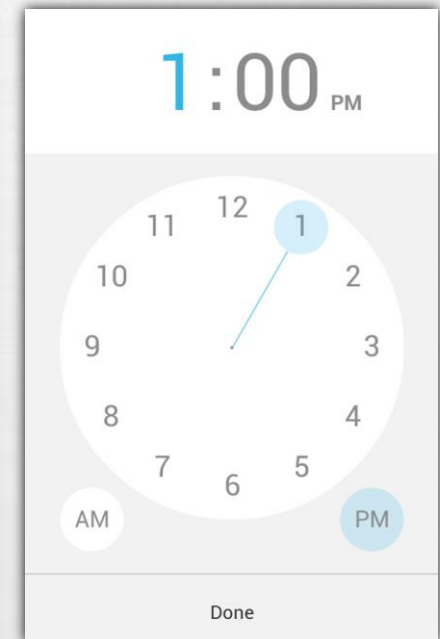
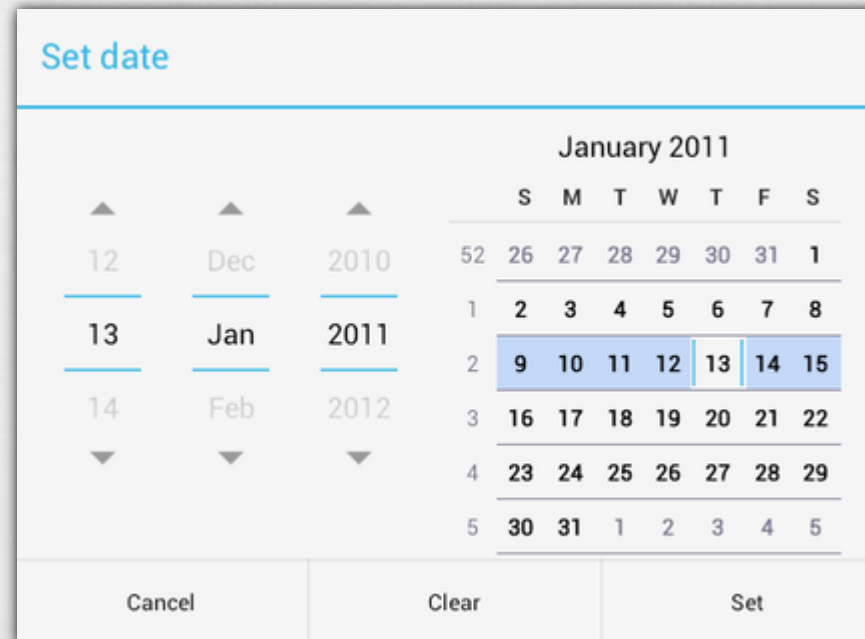
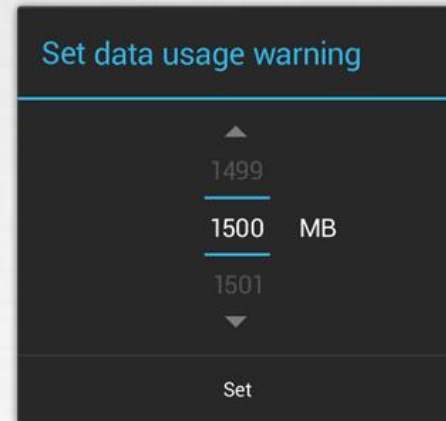


Pickers

Lényegében funkció specifikus Dialog:

- DatePicker
- TimePicker
- Bővebben:

<http://developer.android.com/guide/topics/ui/controls/pickers.html>

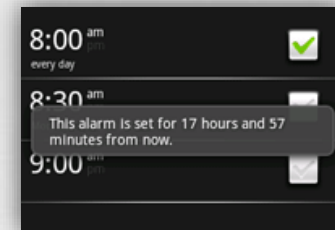


Notifications

Toast notification

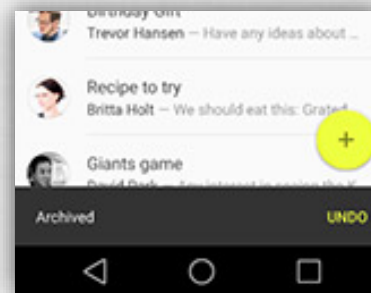
- Szöveges tartalom, felbukkanó buborékban

```
Toast.makeText (
    context,
    "Buborék",
    Toast.LENGTH_SHORT
).show ()
```



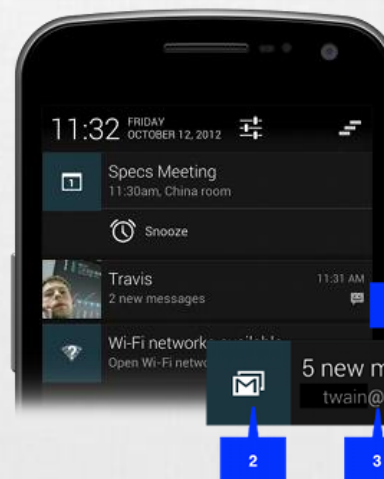
SnackBar

```
Snackbar.makeText (
    context,
    "Buborék",
    Snackbar.LENGTH_SHORT
).show ()
```



Status Bar notification

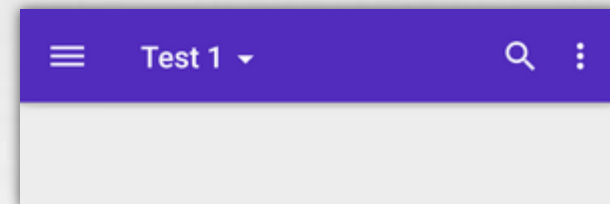
- Egy értesítési terület.
- Jelly Bean óta kibontható.
 - Notifications: Egyszeri értesítés, „Clear” gomb hatására törölhető
 - Ongoing: folyamatban lévőkéről értesítés Pl.: zenelejátszó. Nem törölhető.



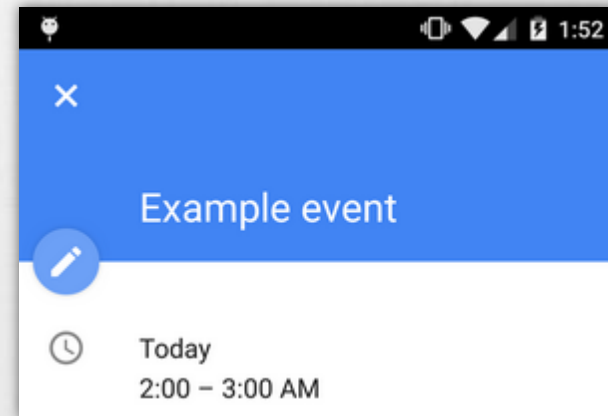
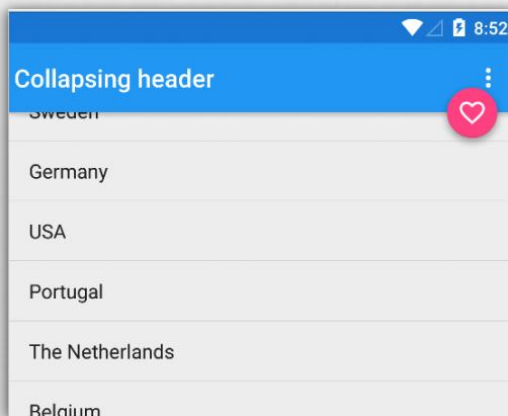
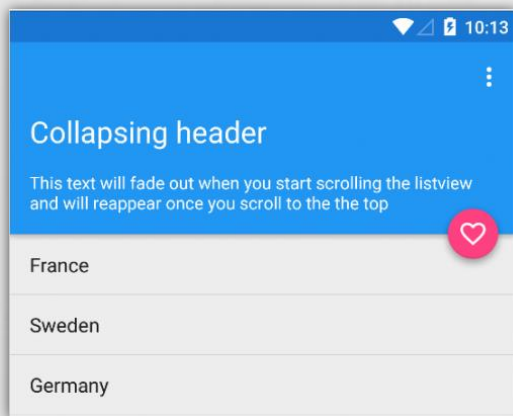
1. Content title
2. Large icon
3. Content text
4. Content info
5. Small icon
6. Time

Toolbar

- <3.0 Window title *Pure design*
- 3.0 – 5.0 ActionBar Holo design
- 5.0+ Toolbar Material design



- Support library visszafele kompatibilis
- Már nem fixen a DecorWindow része, hanem az alkalmazáson belül flexibilisen bárhová elhelyezhető és testreszabható View.
- Tipikusan az alábbi feladatokra jó, mint: navigáció, cím, brand, logó, context menü
- Bővebben: <http://developer.android.com/training/appbar/index.html>



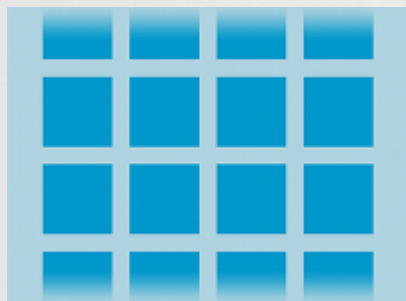
Adapter views



ListView, GridView, Spinner, ViewPager, ~~Gallery~~

Az adatokat egy adapteren keresztül biztosítjuk az UI számára.

- Előre implementált adapter (Pl.: ArrayAdapter)
- Mi implementáljuk (BaseAdapter leszármazott)

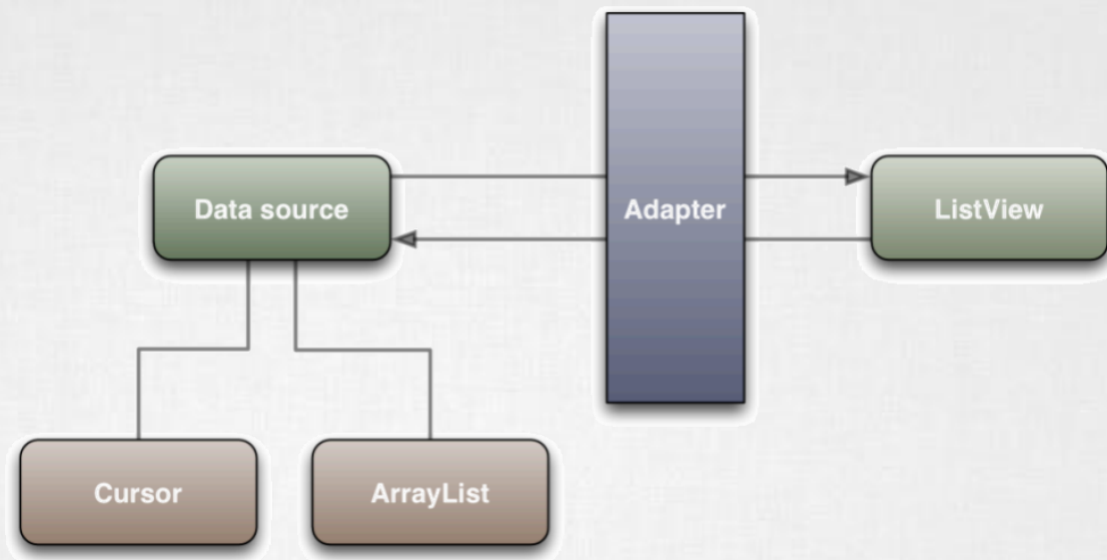


Módszer előnyei

- Az adatok tárolási módja nem meghatározott
Pl.: SD kártya, SQLite adatbázis, internet, ...
Vagy: lista, tömb, hashmap, ...
- UI szétválasztva az adattól
- Optimális erőforrás felhasználás
- Nagy adatmennyiség kezelése
(akár >10E listaelem kezelése)

Adapter views (2) hogyan működik?

- Ősosztály: BaseAdapter
- Implementációk:
 - ArrayAdapter,
 - SpinnerAdapter,
 - CursorAdapter,
 - ...



```
class CustomAdapter : BaseAdapter{  
    fun getCount() : Int { /*...*/ }  
    fun getItem(position: Int) : Object { /*...*/ }  
    fun getItemId(position: Int) : Long { /*...*/ }  
    fun getView(position: Int, convertView: View, parent: ViewGroup) : View {}  
}
```

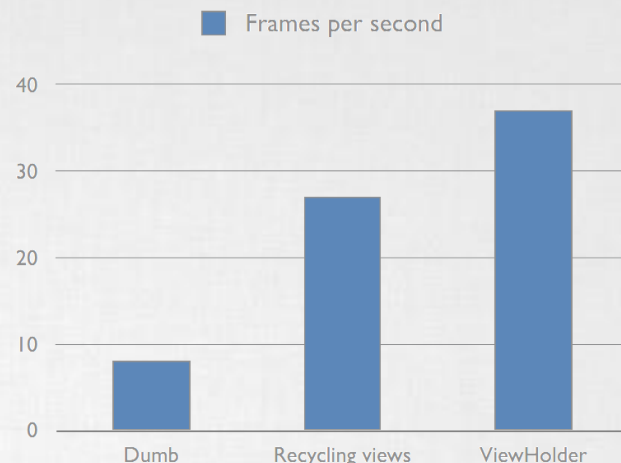
Adapter views (3) hogyan működik jól?

Probléma:

- Minden pozícióban:
`Adapter.getView();`
- Minden esetben új View objektumot létrehozni költséges!
- Több ezer elem esetén?

Nem látható UI elemek újrahasznosítása:

```
fun getView(position: Int, convertView: View, parent: ViewGroup) : View {  
    var holder : ViewHolder? = null  
  
    if (convertView == null) {  
        convertView = inflater.inflate(R.layout.list_item, null)  
        holder = ViewHolder()  
        holder.text = convertView.findViewById(R.id.text)  
        convertView.tag = holder  
    } else  
        holder = convertView.tag as ViewHolder  
  
    holder?.text?.text = DATA[position]  
    return convertView  
}
```

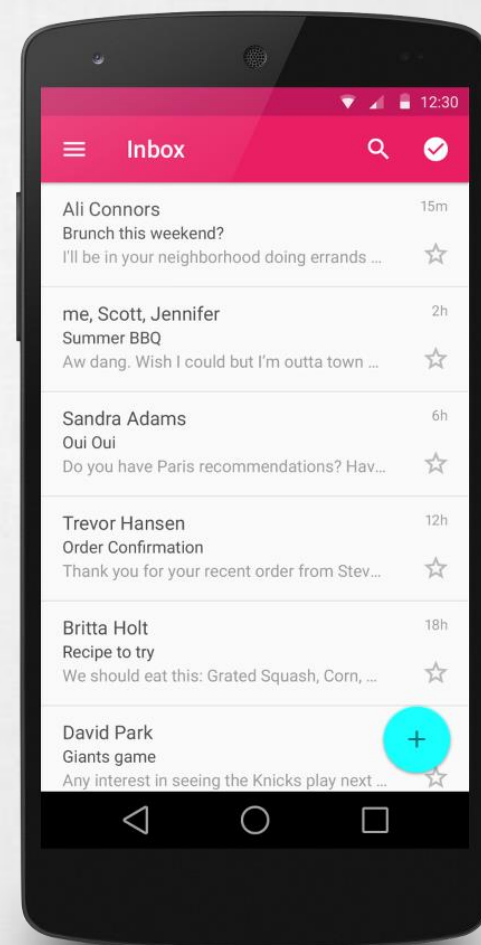


Modern layout: RecyclerView

- Sok AdapterView implementáció helyett egyetlen megoldás
- Adatokat kiszolgáló Adapter mellett leválasztják a Layout, azaz a gyermekek pozícionálásának szabály rendszerét is



- Működéséhez deifniálni kell:
 - RecyclerView.Adapter
 - RecyclerView.LayoutManager
 - ✓ LinearLayoutManager
 - ✓ GridLayoutManager
 - ✓ StaggeredGridLayoutManager
- <https://developer.android.com/guide/topics/ui/layout/recyclerview.html>



XML drawables

- Egyszerűbb alakzatokat, rajzokat, képeket, viselkedéseket leírhatunk XML-ben is
- Példák
 - Layer-list
Több kép együttes kezelése rétegekben.
 - State-list
Különböző állapotokhoz rendelt grafikai elemek.
Például focused, pressed, hover, stb...
 - Level-list
Különböző szintekhez rendelt grafikai elemek.
Például Wi-Fi, fényerő vagy akkumulátor állapotokat ábrázoló képek.
 - Shape
Egy egyszerű alakzat (rectangle, oval, ...) leírása.
Kitöltési szín, vonal szín, sarkok kerekítése, ...



Styles / Themes

- Stílusok alakíthatóak ki, melyeket nagy hatékonysággal lehet újrahasznosítani, és egységesen kezelni.

- res/values/styles.xml

```
<resource>
    <style name="MyStyle" parent="@android:style/Widget.Button">
        <item name="android:background">#556677</item>
        ...
    </style>
</resource>
```

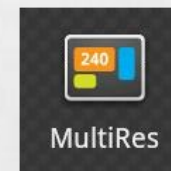
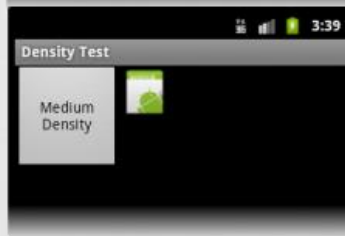
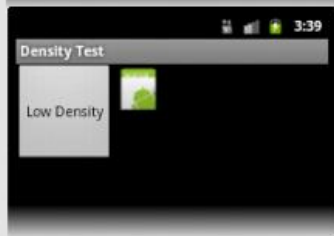
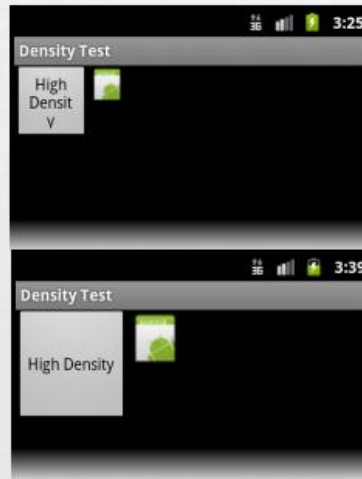
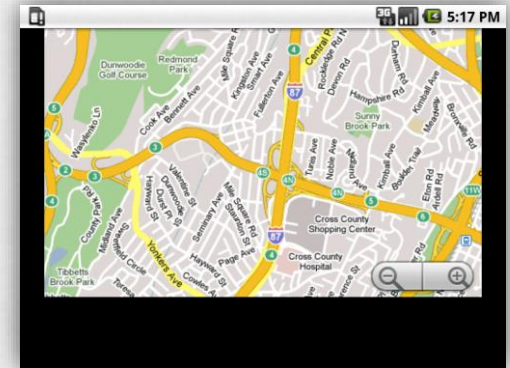
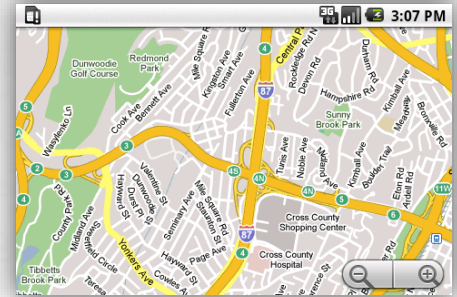
- res/layout/activity_main.xml

```
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Gomb"
    style="@style/MyStyle"
/>
```

Különböző felbontású készülékek támogatása



Problémák



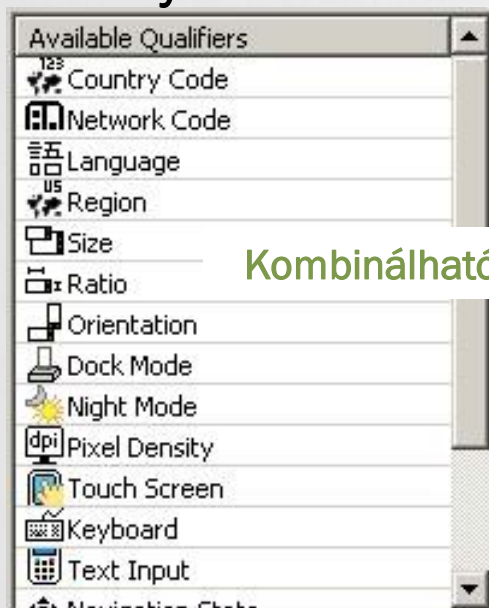
API Level 6+
Launcher Icon

API Level 4+
Launcher Icon

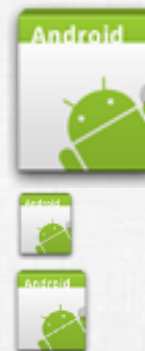
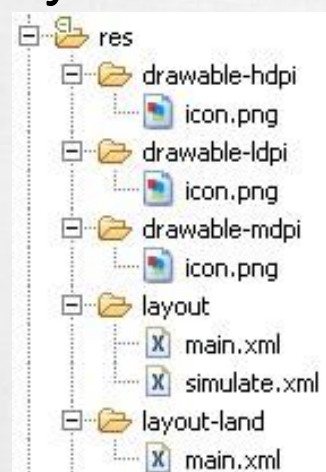
Qualifiers (minősítők)

- Különböző esetekre különböző megközelítés szükséges.
- Sok IF és SWITCH helyett minősítőket definiáltak.
- Automatikus kiértékelés, aktuálisan jellemző mappából dolgozik.

Néhány minősítő:

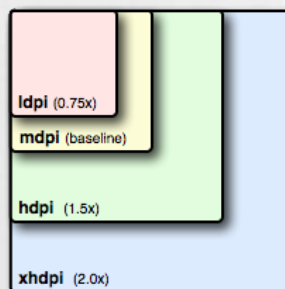


Gyakorlatban:



Ugyanaz a kép 3 különböző méretben

Ugyanazon felület különböző leírása „portrai,” és „landscape”

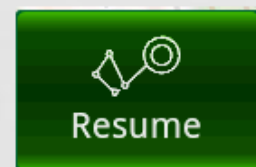


Ikon méretek

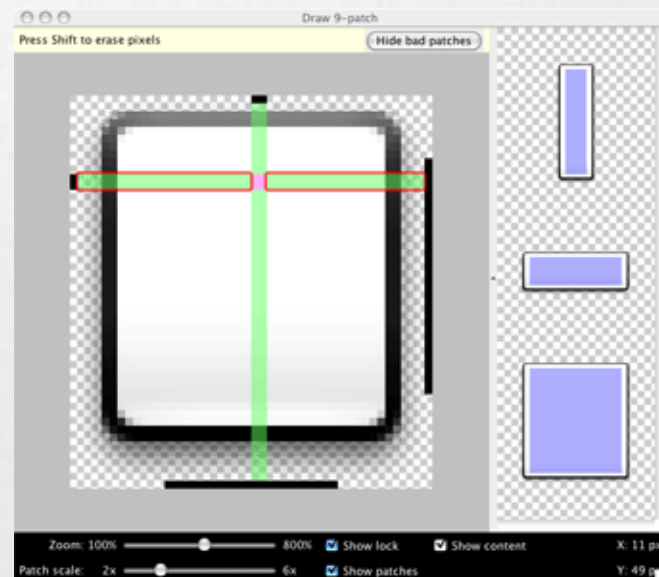
- 36x36 for low-density
- 48x48 for medium-density
- 72x72 for high-density
- 96x96 for extra high-density

9-Patch

- Dinamikus tartalom esetén, ha egyéni hátteret használunk akkor az eltorzulhat. Pl.: egy gombnál
- Speciális PNG fájlal meghatározhatjuk mely részeket nyújthatjuk meg a mintaképen. (bal oldal és fent) Szélén fekete pixelekkkel jelöljük meg ezt a nyújtható területet.
- Jobb oldalt és lent a kitöltési területet jelölhetjük meg.
- Mindig ***.9.png** kiterjesztésű
- Van hozzá eszköz: [SDK path] / tools / draw9patch.bat



Not Scaled	Scaled Horizontally Only	Not Scaled
Scaled Vertically Only	Scaled Horizontally and Vertically	Scaled Vertically Only
Not Scaled	Scaled Horizontally Only	Not Scaled



Vector drawable

- Android 5.0+, support library használatával régebbi is
- Lényegében túrbózott ShapeDrawable: rajzolást írja le, vonalak, görbék, egyenesek, színek
- Minőség romlás nélkül skálázható
- Kisebb APK méret, nincsen több változat a képből
- SVG-hez hasonló, XML leíró használ
- Android Studio support: SVG to Vector drawable converter
- Bővebben:
<https://developer.android.com/guide/topics/graphics/vector-drawable-resources.html>
- AnimatedVectorDrawable is elérhető
<https://developer.android.com/reference/android/graphics/drawable/AnimatedVectorDrawable.html>
- Path morphing:
<https://lewismcgeary.github.io/posts/animated-vector-drawable-pathMorphing/>

Density & Scale Independent Pixel (DP, SP)

DP vagy DIP (**D**ensity-**I**ndependent **P**ixel)

- Egy virtuális pixel-egység, sűrűség-független képpont.
- 160dpi felbontású készüléken 1dp = 1px.
- Eltérő pixelsűrűség esetén automatikusan átváltja az alábbi módon:

$$\text{pixels} = \text{dips} * (\text{density} / 160)$$

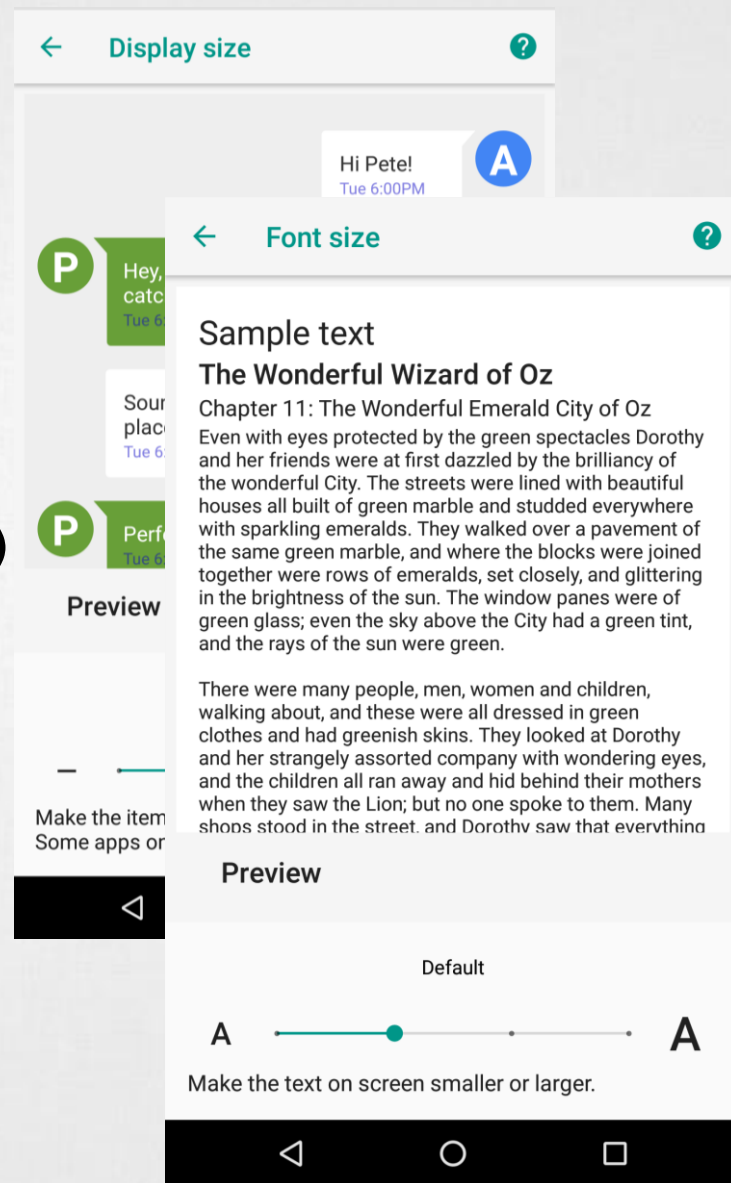
160dpi felbontás esetén, 10dp = 10px

240dpi felbontás esetén, 10dp = 15px

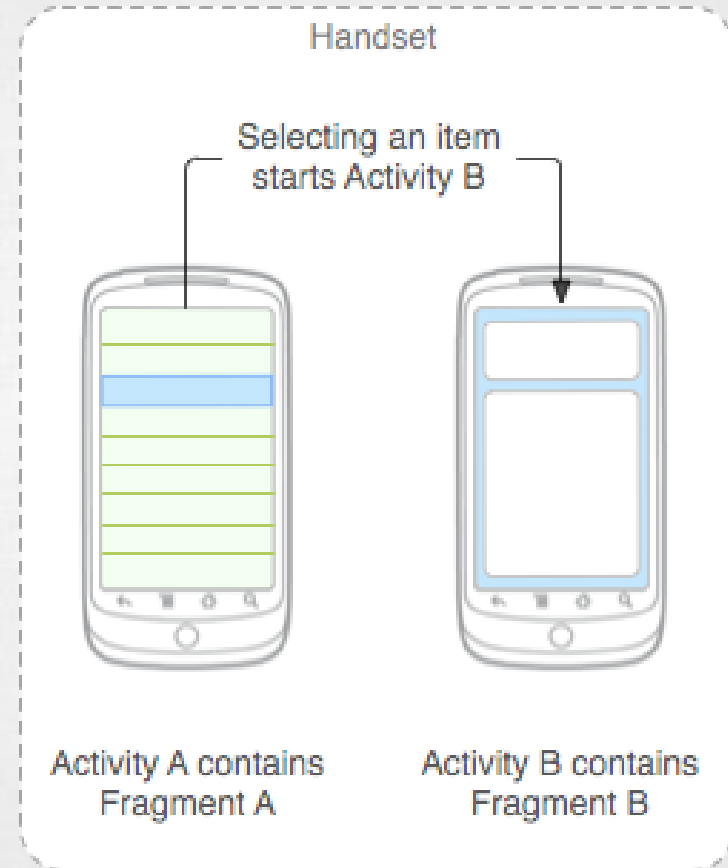
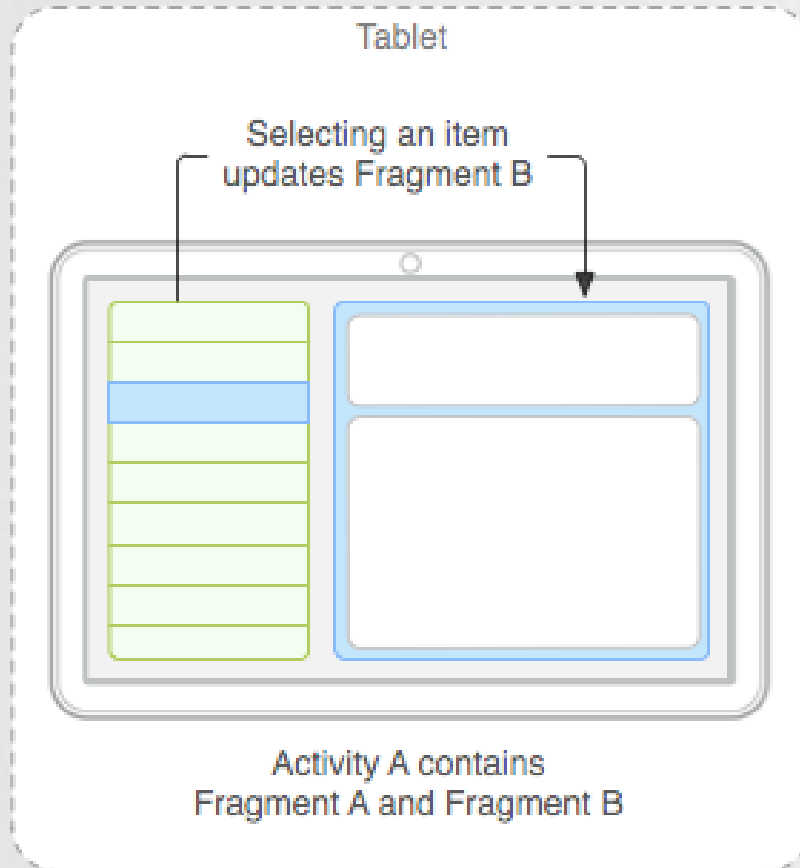
SP vagy SIP (**S**cale-**I**ndependent **P**ixel)

Mint a DP, csak szöveg esetén használatos.

`android:textSize="16sp"`



Adaptív design



Backward compatibility



Probléma: lassú terjedés

Annak ellenére, hogy a rendszer ingyenes az új Android verziók lassan terjednek, melynek potenciális okai:

- Az eszközgyártóknak az eladott készülékek jelentik a bevételt, így maximum 2-3 Android főverzióra való frissítés a jellemző.
- Növekvő hardver igények az újabb Android rendszerek megjelenésével. (Ez részben igaz)
- Egyéni, gyártói megjelenés adaptálása az új rendszerre.

HTC Sense, Samsung TouchWiz, LG UI, ...

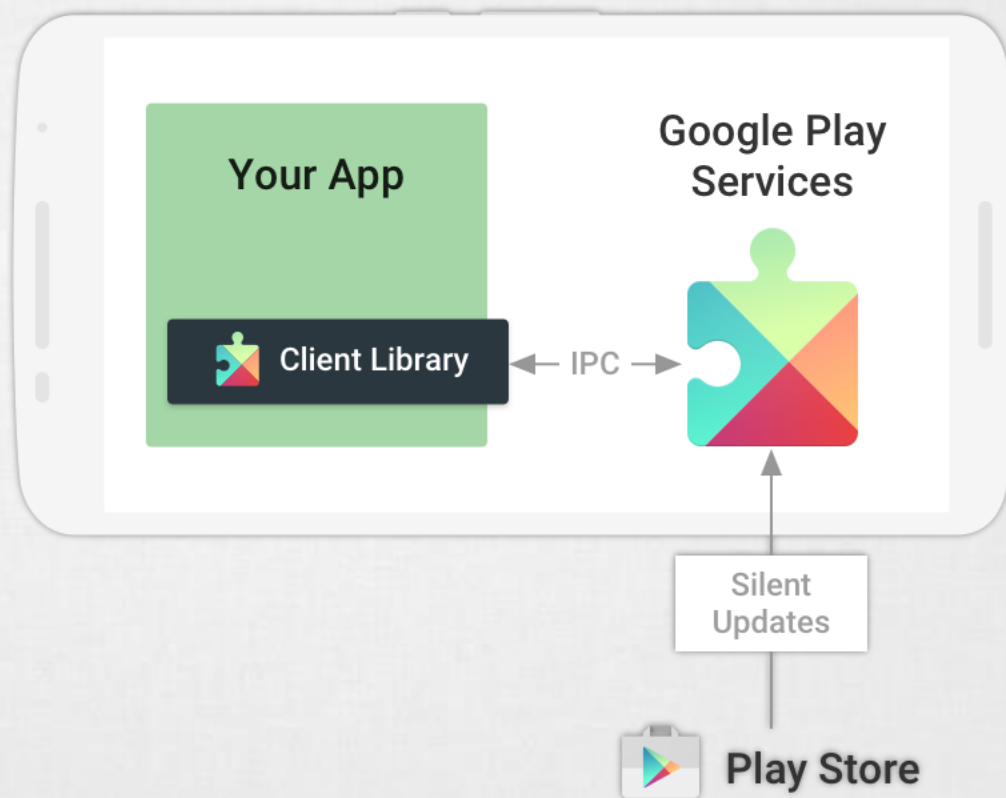
Következmény:

- Nő a széttagoaltság, túl sok különböző Android verzió, amit kezelni kell az alkalmazásokban.
- Új Android arculati elemek adaptálásának problémája.
Pure (<2.3), Holo design(3.0-4.4), Material design (>5.0)
- Új API-k adta lehetőségek nehezen terjednek.

Google válasza: Google Play services

Google egyik válasza a fregmentáció csökkentésére a [Google Play services](#).

- Minden készülékre (>2.2) automatikusan települ, ha a Play Store telepítve van.
- A háttérben automatikusan frissül.
- Tartalmazza az új Google szolgáltatások API-jait, mint például:
 - Google Maps V2,
 - Location API,
 - GCM,
 - In-app purchase,
 - Analytics,
 - Google Drive API,
 - Wearable API,
 - Wallet,
 - Game API,
 - ...



Google válasza: **Support library**

Az egyes Android verziókkal összhangban ki szoktak adni több kompatibilitási csomagot is, ami igény szerint beépíthető az alkalmazásokba. Például a Material Design elemei is ezekkel vihetőek vissza akár Android 2.3 verzióig is.

- Support library (v4, v7, v8, v13)
- V7 Support library: AppCompatActivity, CardView, GridLayout, MediaRouter, Palette, RecyclerView, Preference
- Leanback library
- Design Support library
- Custom Tabs Support library
- Percent Support library
- Multidex Support library
- Annotations Support library

Android Jetpack 2018 őszétől

Lényegében a *support library* világának újragondolása, Android szoftver komponensek gyűjteménye.

Mostanra értelmét veszítette a *support library* API level alapú (~~v4, v7, v8, v13~~) felosztása. Így már gyakoribb frissítések is jöhetnek.

Az *androidx.** csomagnév alá kerül át az összes *support library* funkció. [AndroidX refaktorálási dokumentáció.](#)

4 alapvető kategóriára bomlik:

Foundation

Behavior

Architecture

UI

Forrás: <https://developer.android.com/jetpack/>