

Felhasználói felület alapok



Ekler Péter

BME VIK AUT, AutSoft

peter.ekler@aut.bme.hu

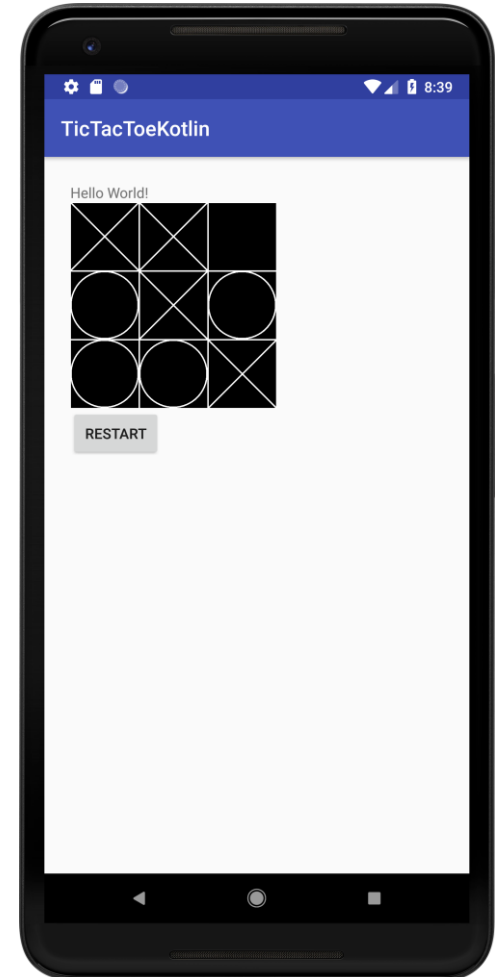


Tematika

1. Android platform bemutatása, Kotlin alapok
2. Alkalmazás komponensek, Kotlin konvenciók
3. Felhasználói felület
4. Fragmentek, haladó UI
5. Listák kezelése hatékonyan
6. Perzisztens adattárolás, adatbázisok, haladó Kotlin
7. Hálózati kommunikáció
8. Felhő szolgáltatások
9. Helymeghatározás, térkép kezelés
10. Architektúra komponensek, JetPack

Gyakorlás

- Készítsünk TicTacToe alkalmazást!
- Érintett témák
 - > Singleton objektum
 - `object TicTacToeModel {...}`
 - > Activity és egyedi nézet közti kapcsolat
 - > Méret felüldefiniálása



Tartalom

- Felhasználói felület fogalmak
- Erőforrás típusok
- Erőforrásminősítők használata, jellemzőik
- Layout erőforrások (LinearLayout, RelativeLayout)
- Nézetek/View-k
- Képek kezelése egyszerűen
- Menükezelés

Felhasználói felület alapfogalmak

Alapfogalmak, erőforrások, erőforrás-minősítők

Különböző képernyők támogatása 1/2

- Az Android futtatható különböző felbontású és sűrűségű képernyőkön
- A rendszer egyfajta mechanizmust biztosít az eltérő képernyők támogatására
- A fejlesztő válláról a legtöbb munkát leveszi
- Csak a megfelelő erőforrásokat kell elkészíteni
- Például egy mobiltelefon és egy tablet képernyője tipikusan eltérő

Különböző képernyők támogatása 2/2

- A rendszer automatikusan is skálázza és átméretezi az alkalmazás felületét, hogy minden készüléket támogasson
- De! mindenképp fontos, hogy a felhasználói felület és az erőforrások (képek) optimalizálva legyenek az egyes felbontásokhoz és sűrűségekhez
- Ezzel nagy mértékben növelhető a felhasználói élmény
- Továbbá valóban az egyes készülékekhez igazítható a megjelenítés, ami növeli a felhasználói elégedettséget
- A módszer követésével minden készüléket támogató alkalmazás készíthető UI szempontjából egyetlen .apk-ba csomagolva

Legfontosabb fogalmak 1/2

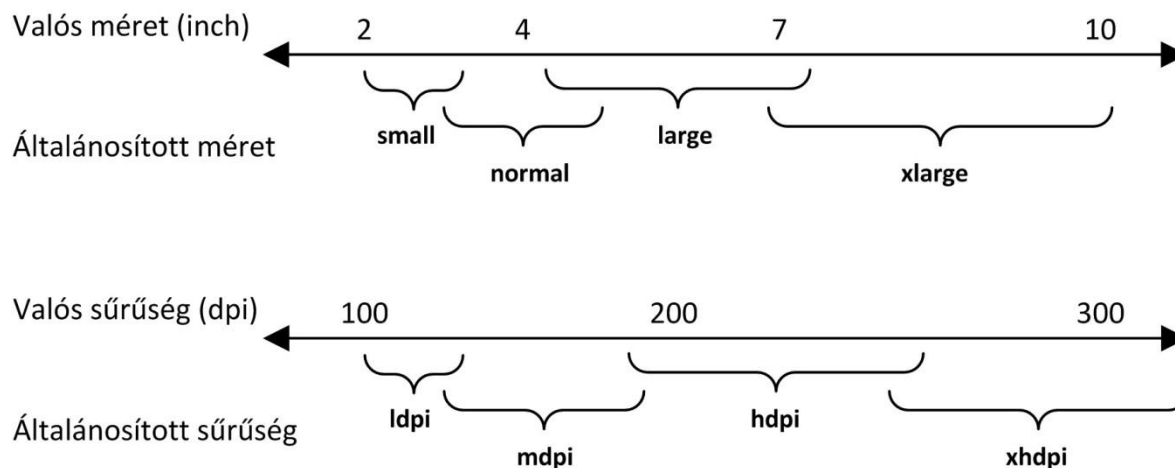
- Képernyő méret (*screen size*):
 - > Fizikai képátló
 - > Az egyszerűség kedvéért az Android 4 kategóriát különböztet meg: small, normal, large, és extra large
- Képernyő sűrűség (*screen density – dpi*): A pixelek száma egy adott fizikai területen belül, tipikusan inchenkénti képpont (dpi – dots per inch)
 - > Az Android 6 kategóriát különböztet meg: low, medium, high és extra high
- Orientáció (*orientation*): A képernyő orientációja a felhasználó nézőpontjából:
 - > Álló (*portrait*)
 - > Fekvő (*landscape*)
 - > Az orientáció futási időben is változhat, például a készülék eldöntésével
 - > Lehetőség van rögzíteni az orientációt

Legfontosabb fogalmak 2/2

- Felbontás (*resolution* – *px*): Képernyő pixelek száma
 - > A UI tervezésekor nem felbontással dolgozunk, hanem mérettel és pixel sűrűséggel
- Sűrűség független pixel (*density-independent pixel* – *dp*)
 - > Virtuális pixel egység, amit UI tervezéskor célszerű használni
 - > Egy dp egy fizikai pixelnek felel meg egy 160 dpi-s képernyőn (160 az egységes középérték)
 - > A rendszer futási időben kezel minden szükséges skálázást a definiált dp-nek megfelelően
 - > $px = dp * (dpi / 160)$
 - > Például egy 240 dpi-s képernyőn, 1 dp 1.5 fizikai pixelnek felel meg

Általánosított képernyő méretek 1/2

- 6 általánosított méret:
 - > small, normal, large és xlarge, stb.
- 6 általánosított sűrűség:
 - > ldpi (low), mdpi (medium), hdpi (high), és xhdpi (extra high), stb.



Általánosított képernyő méretek 2/2

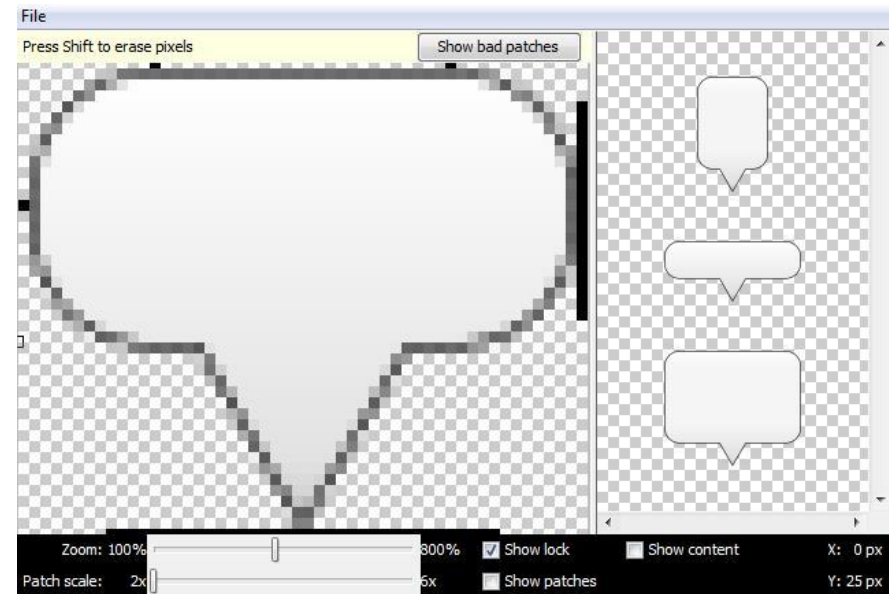
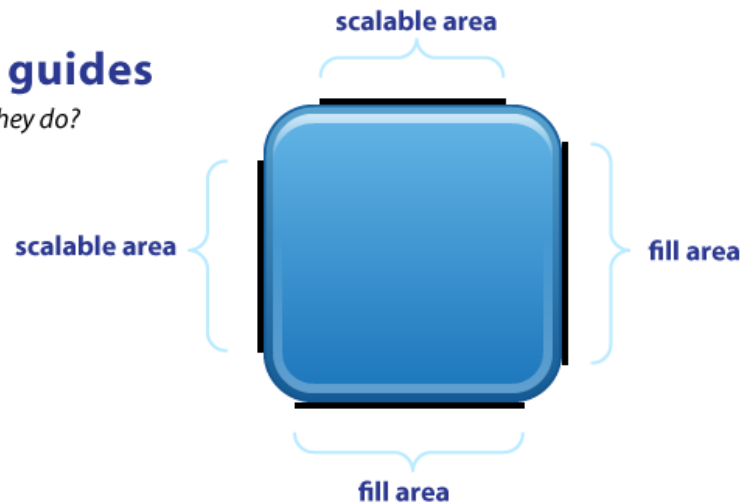
- Definiált minimum küszöbök:
 - > *xlarge*: legalább 960dp x 720dp
 - > *large*: legalább 640dp x 480dp
 - > *normal*: legalább 470dp x 320dp
 - > *small*: legalább 426dp x 320dp
- 3.0-ás verzió alatt lehetnek bugok a normal és large megkülönböztetésében

NinePatch képek

- PNG képek skálázási szabályainak meghatározása
- SDK része: draw9patch.bat

9-patch guides

what do they do?



Mi nem igaz az Android UI támogatására?

- A. Az Android automatikusan átméretezi a képet, ha nincs megfelelően illeszkedő.
- B. Az Android támogatja a sűrűségfüggetlen megjelenítést.
- C. $px = dp * (dpi / 160)$
- D. Közvetlenül pixelben nem adhatók meg a méretek.

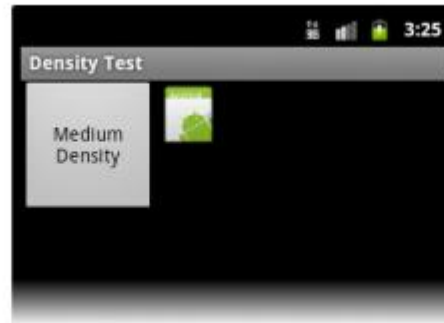
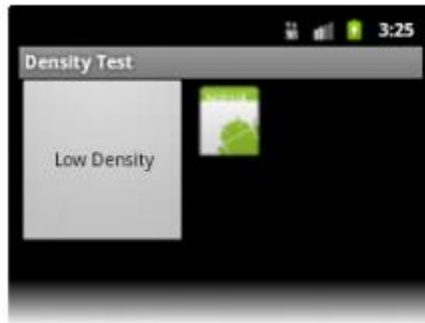
<http://babcomaut.aut.bme.hu/votes/>

Sűrűség függetlenség

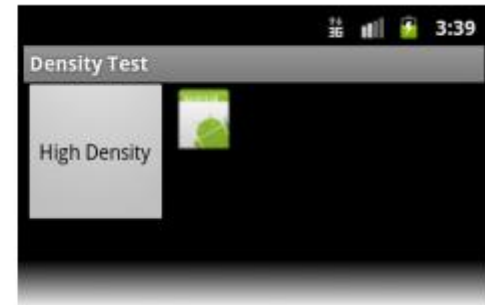
- Az alkalmazás akkor lehet „sűrűség független”, ha a felhasználói felületi elemek a felhasználó szemszögéből megőrzik a fizikai méretüket különböző sűrűségeken
- A „sűrűség függetlenség” fenntartása nagyon fontos, hiszen például egy gomb fizikailag nagyobbnak tűnhet egy alacsonyabb sűrűségű képernyőn
- A képernyő sűrűséghez kapcsolódó problémák jelentősen befolyásolhatják az alkalmazás felhasználhatóságát.
- Az Android kétféle módon is segít elérni a sűrűség függetlenséget:
 - > A rendszer a **dp** kiszámítása alapján skálázza a felhasználói felületet az aktuális képernyő sűrűségnek megfelelően
 - > A rendszer a képernyő sűrűség alapján automatikusan átskálázza a kép erőforrásokat

Példa

- Sűrűség függetlenség támogatás nélkül:



- Sűrűség függetlenség támogatással:



Kép erőforrások átméretezése

- Nem szerencsés, ha a rendszerre bízunk az átméretezést, hiszen így elmosódottak lehetnek a képek nagy felbontáson
- Az Android úgynevezett minősítő „string” (configuration qualifier)-ek segítségével teszi lehetővé, különböző erőforrások használatát
- A minősítő „string”-et az erőforrás könyvtár (res/) neve után kell fűzni (<resources_name>-<qualifier>, pl. *layout-xlarge*):
 - > <resources_name>: standard erőforrás típus, pl. *drawable*, vagy *layout*
 - > <qualifier>: minősítő a képernyőre vonatkozólag, pl. *hdpi*, vagy *large*
 - > Több minősítő is szerepelhet egymás után kötőjellel elválasztva

Legfontosabb minősítő értékek

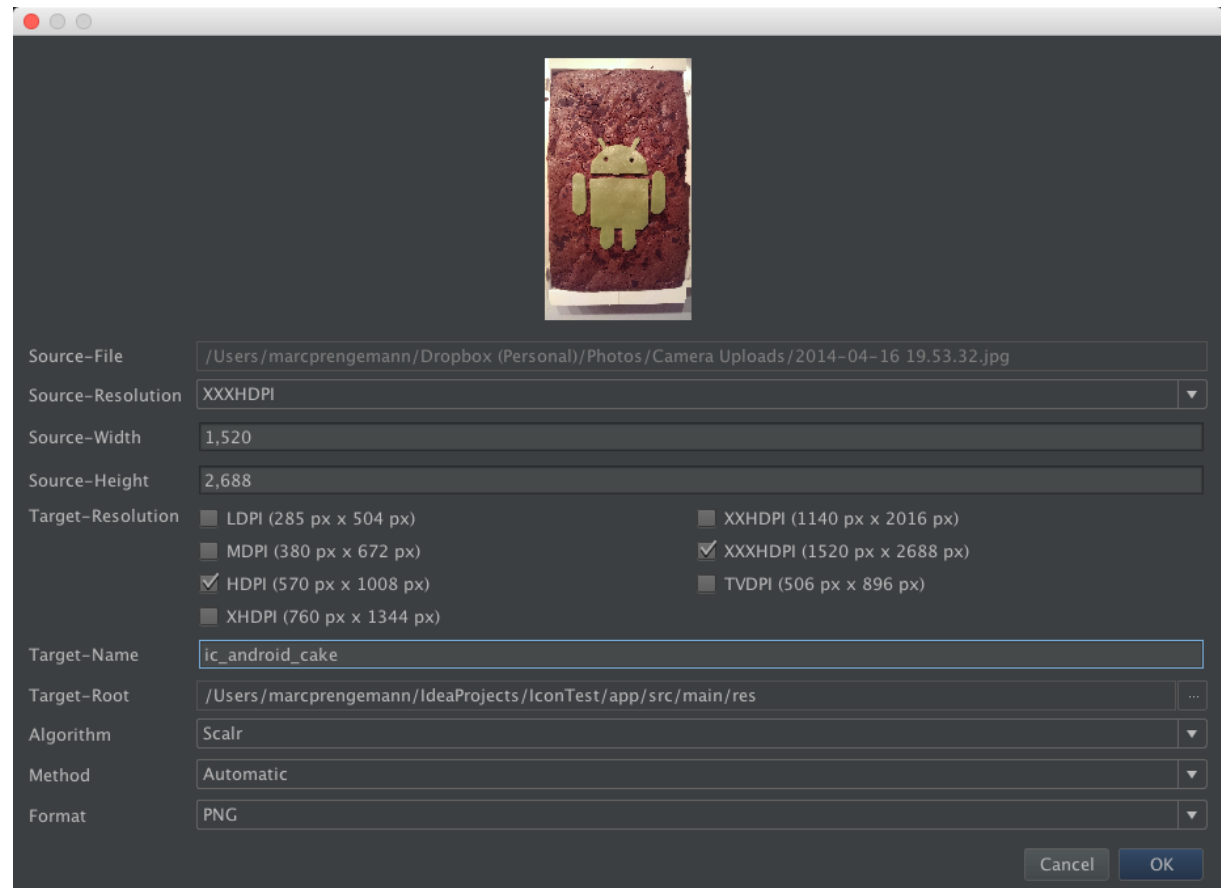
- Méret:
 - > small, normal, large, xlarge
- Sűrűség:
 - > ldpi, mdpi, hdpi, xdpi, nodpi (a rendszer az ebben lévőket nem méretezi át), tvdpi
- Irány:
 - > land, port
- Képarány:
 - > long (a jelentősen szélesebb, vagy magasabb kijelzőkhöz), notlong

Példák

- *res/layout/my_layout.xml*
 - *res/layout-small/my_layout.xml*
 - *res/layout-large/my_layout.xml*
 - *res/layout-xlarge/my_layout.xml*
 - *res/layout-xlarge-land/my_layout.xml*
-
- *res/drawable-mdpi/my_icon.png*
 - *res/drawable-hdpi/my_icon.png*
 - *res/drawable-xhdpi/my_icon.png*

Android Drawable Importer

- <https://plugins.jetbrains.com/plugin/7658?pr=>



Speciális méret-minősítők

- $sw\langle N\rangle dp$ (smallestWidth):
 - > Például $sw600dp$, $sw720dp$
 - > A képernyőn látható legkisebb dimenziót specifikálja
 - > Másképp: a képernyőn elérhető legkisebb magasság és szélesség
 - > A minősítővel biztosítható, hogy a képernyő orientációjától függetlenül biztos, hogy van $\langle N\rangle dp$ szélesség
- $w\langle N\rangle dp$ (available screen width):
 - > A legkisebb szélesség, amivel az erőforrást lehet használni
 - > Orientáció változáskor értesül a rendszer
- $H\langle N\rangle dp$ (available screen height):
 - > A legkisebb magasság, amivel az erőforrást lehet használni
 - > Orientáció változáskor szintén értesül a rendszer

Alkalmazás szintű képernyő követelmények

- *android:requiresSmallestWidthDp*
 - > A legkisebb dimenzió, amivel a képernyőnek rendelkeznie kell
 - > `<manifest ... >`
 `<supports-screens android:requiresSmallestWidthDp="600" />`
 ...
 `</manifest>`
- *android:compatibleWidthLimitDp*
 - > Maximum legkisebb szélesség, amit még az alkalmazás támogat
- *android:largestWidthLimitDp*

Legfontosabb tényezők

- Használjuk a **wrap_content**, **match_parent**, vagy **dp** egységeket, amikor egy felületet készítünk!
- Súlyozás
- Ne használjunk beégetett pixel értékeket!
- Ne használjuk az **AbsoluteLayout**-ot (elavult)!
- Mindenképp készítsünk különböző kép erőforrásokat az eltérő képernyősűrűségekhez
- Szövegek méretezéséhez érdemes használni az **sp** (scale-independent pixel) mértéket, **dp**-hez hasonlóan működik

Mi igaz az Android UI támogatására?

- A. Az Android nem támogatja a sűrűségfüggetleneséget.
- B. Ha nincs a képernyő tulajdonságaihoz illeszkedő erőforrás direkt megadva, akkor kivétel dobódik.
- C. A dp mértékegység helyett a dpi-t javasolt használni.
- D. Az Android futás közben tudja kikeresni a leginkább illeszkedő erőforrást.

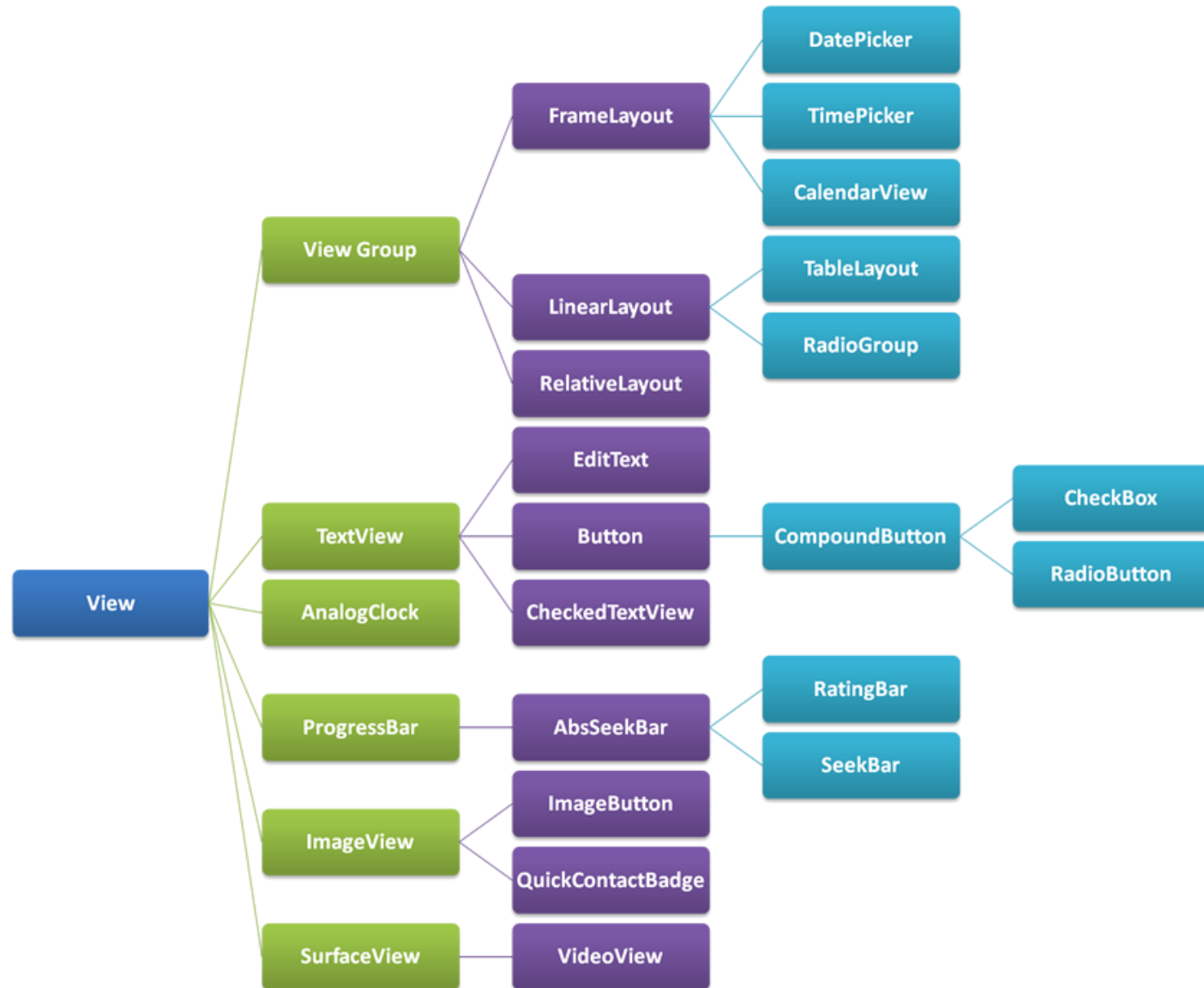
<http://babcomaut.aut.bme.hu/votes/>

Felhasználói felület erőforrások

- Felületek
 - > *res/layout*
- Szöveges erőforrás:
 - > *res/values/strings.xml*
- Kép erőforrások:
 - > *res/drawable-xyz/[kep].[ext]*
- Animáció erőforrások
 - > *res/anim*
- További erőforrások: *animator, szín, menü, nyers (raw), xml fileok*

Layout-ok

Android UI architektúra



Android felhasználói felület felépítése

- Minden elem a View-ból származik le
- Layout-ok (elrendezések):
 - > ViewGroup leszármazottak
 - > ViewGroup is a View-ból származik le!
- ViewGroup-ok egymásba ágyazhatók
- Saját View és ViewGroup is készíthető, illetve a meglevők is kiterjeszthetők

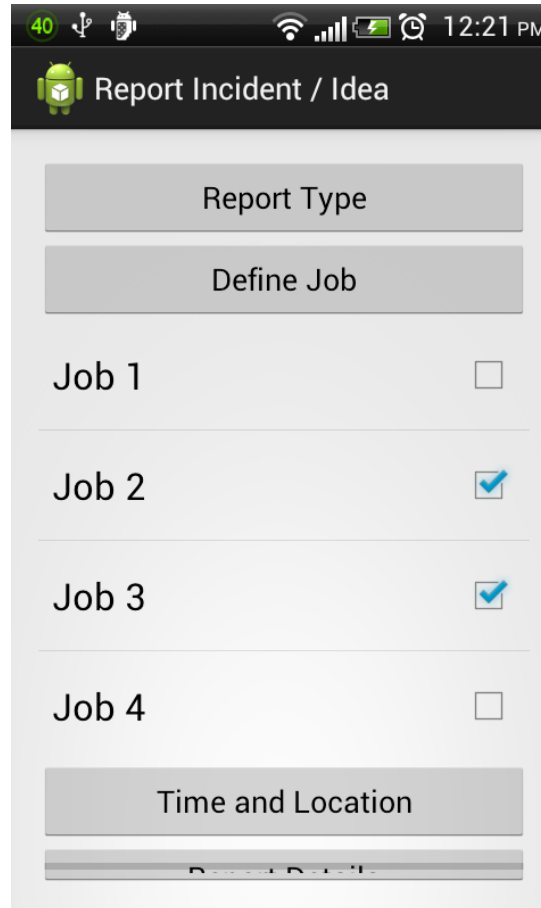
Layout-ok (ViewGroup)

- LinearLayout
- RelativeLayout
- ConstraintLayout
- AbsoluteLayout (NEM használjuk!)
- GridLayout
- RecyclerView
- Teljes lista:
 - > <http://developer.android.com/reference/android/view/ViewGroup.html>

```
public class  
RelativeLayout  
extends ViewGroup  
  
java.lang.Object  
└─ android.view.View  
    └─ android.view.ViewGroup  
        └─ android.widget.RelativeLayout
```

LinearLayout

- LinearLayout != Lista



The screenshot shows an Android application interface with a status bar at the top displaying 40% battery, signal strength, and the time 12:21 PM. The app title bar reads 'Report Incident / Idea' with an Android icon. The main content area is a vertical list of items, each with a text label on the left and a checkbox on the right. The items are 'Job 1', 'Job 2', 'Job 3', and 'Job 4'. 'Job 2' and 'Job 3' have their checkboxes checked with blue checkmarks. Below the list is a button labeled 'Time and Location'. At the very bottom, there is a partially visible button labeled 'Report'.

Job	Selected
Job 1	<input type="checkbox"/>
Job 2	<input checked="" type="checkbox"/>
Job 3	<input checked="" type="checkbox"/>
Job 4	<input type="checkbox"/>

Súlyozás Layout tervezéskor

- Megadható egy layout teljes súly értéke (weightSum)
- Elemek súly értéke megadható és az alapján töltődik ki a layout
 - > layout_weight érték
 - > A megfelelő width/height ilyenkor 0dp legyen!
- Hasonló, mint HTML-ben a %-os méret megadás

Layout súlyozás példa

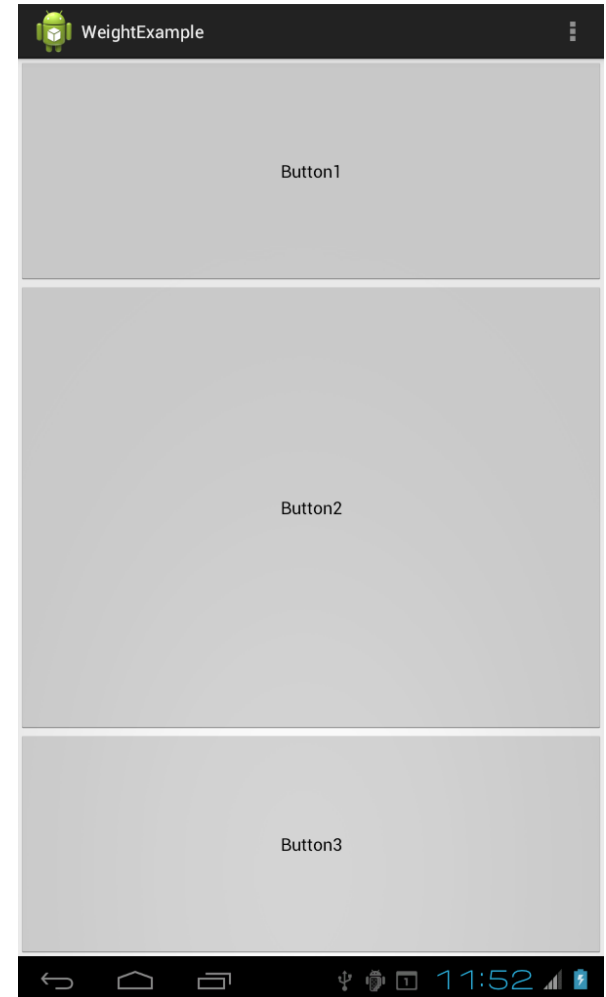
```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:weightSum="4"
    android:orientation="vertical">
```

```
<Button
    android:layout_width="match_parent"
    android:layout_height="0dp"
    android:layout_weight="1"
    android:text="Button1" />
```

```
<Button
    android:layout_width="match_parent"
    android:layout_height="0dp"
    android:layout_weight="2"
    android:text="Button2" />
```

```
<Button
    android:layout_width="match_parent"
    android:layout_height="0dp"
    android:layout_weight="1"
    android:text="Button3" />
```

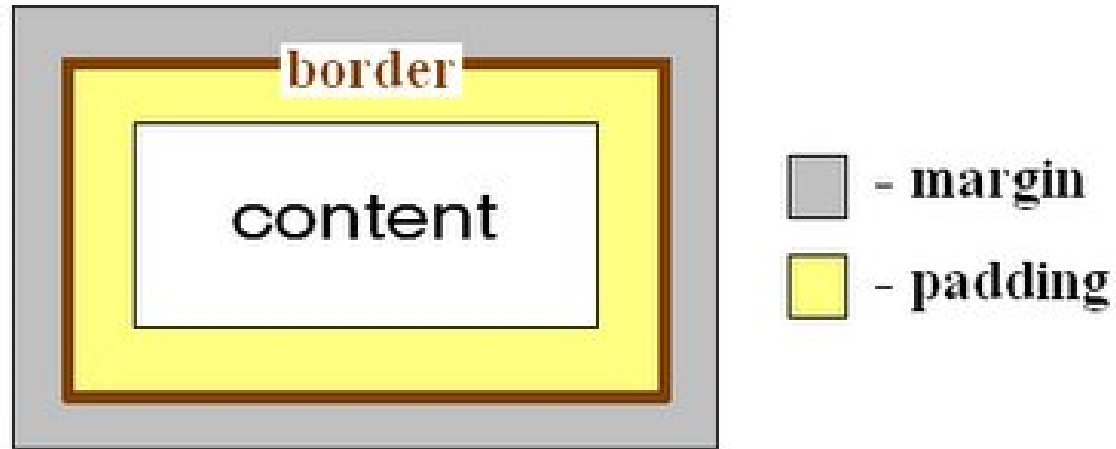
```
</LinearLayout>
```



LinearLayout példák

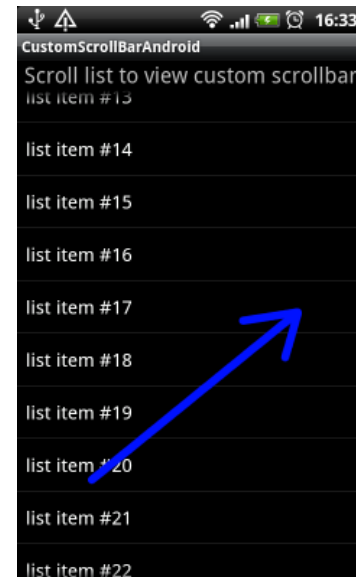
- Jellemző paraméterek:
 - > Margin, padding
 - > Gravity
 - > ScrollView
 - > Weight

Padding és Margin



ScrollView

- ScrollView és HorizontalScrollView
- Layout container, amely scrollozást tesz lehetővé, ha a benne levő tartalom „nagyobb”
- Nem kötelező a teljes képernyőt kitöltenie
- Egy layout/képernyő több ScrollView-t is tartalmazhat



ScrollView példa

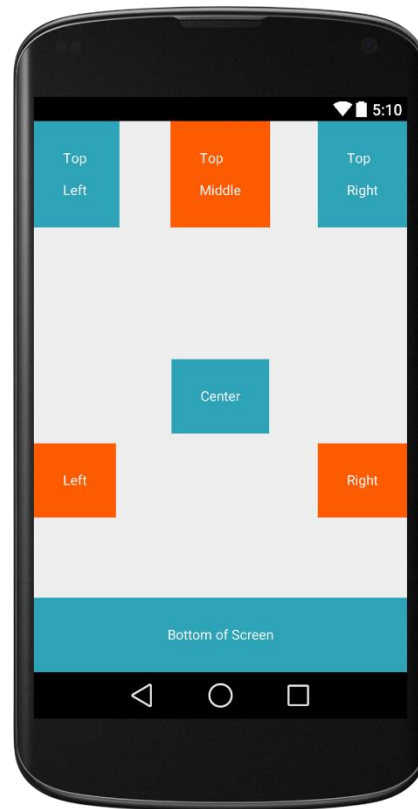
```
<ScrollView xmlns:android=
    "http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="10dp"
    android:fillViewport="false">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical">
        <ImageView
            android:id="@+id/imageView"
            android:layout_width="wrap_content"
            android:layout_height="200dp"
            android:scaleType="centerCrop"
            android:src="@drawable/image" />

            ...
        </LinearLayout>
    </ScrollView>
```

RelativeLayout

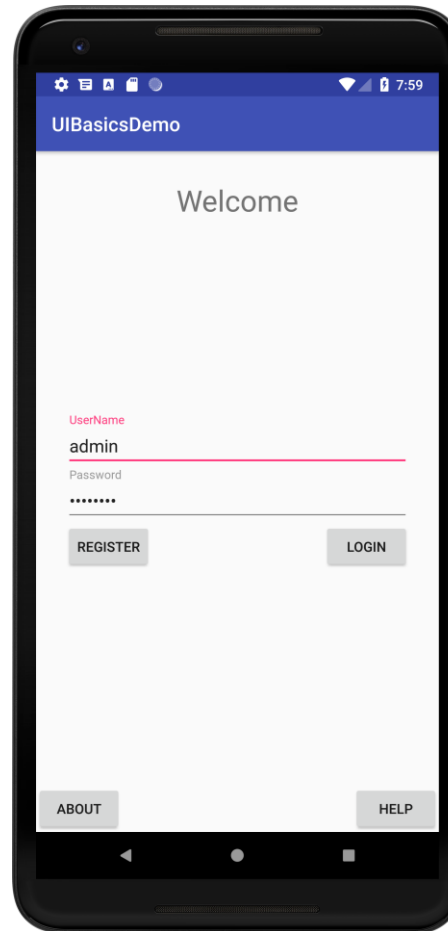
- Elemek egymáshoz való viszonya definiálható
- Demo



Gyakoroljunk

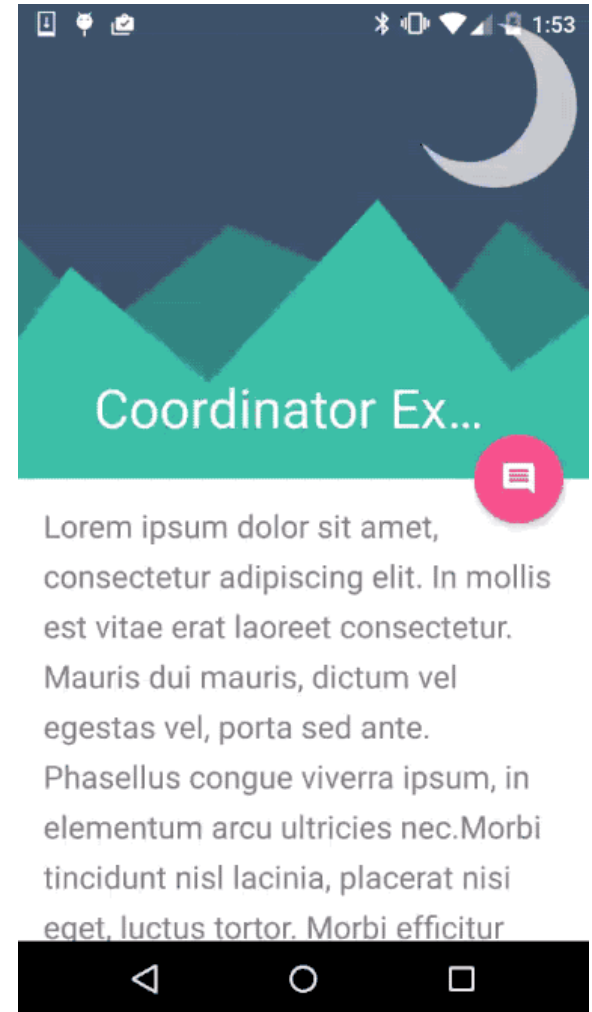


- Készítsünk egy Login képernyőt



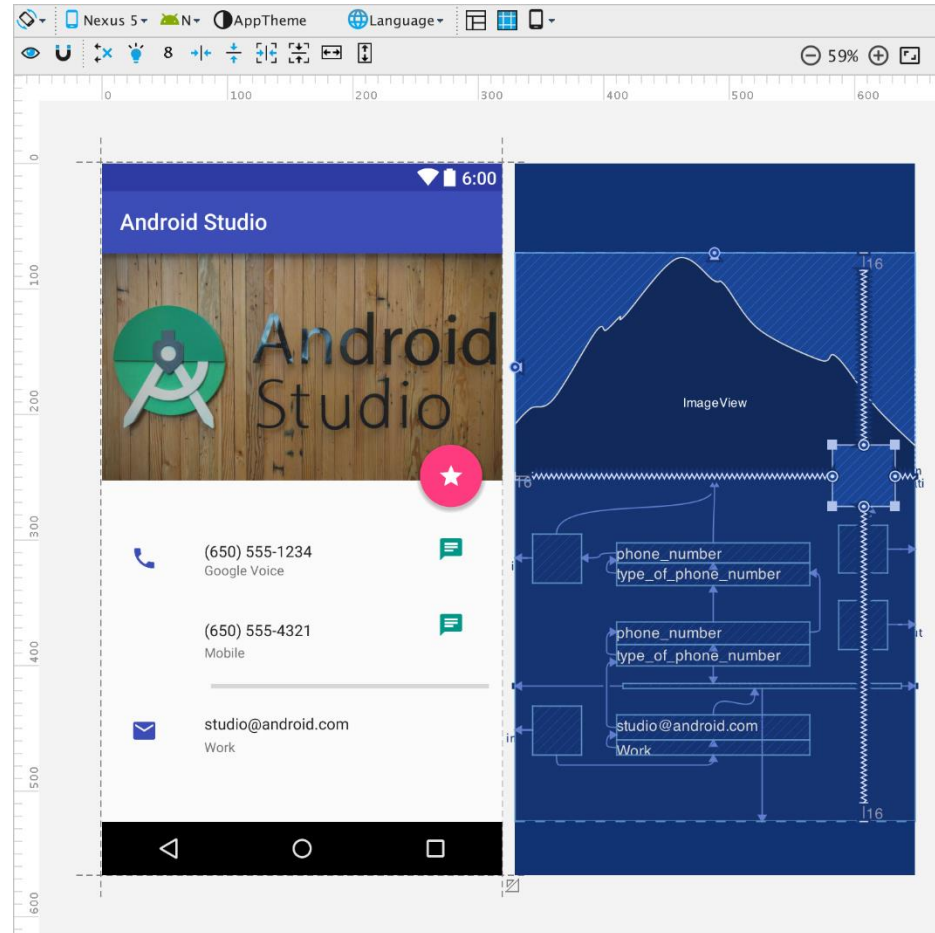
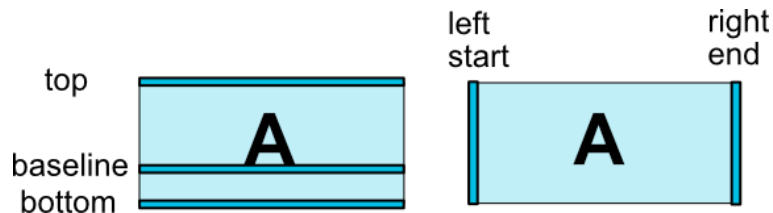
CoordinatorLayout, AppBarLayout

- CoordinatorLayout: továbbfejlesztett FrameLayout
- CoordinatorLayout fő feladatai:
 - > Felső szintű alkalmazás UI irányelv
 - > Konténer, mely támogatja a beépített elemek material stílushoz igazodó elhelyezkedését
- Behavior paraméterekkel meghatározható a kapcsolódó elemek elhelyezése
- AppBarLayout csatolható hozzá, mely a material design-hez illeszkedő scrollozást támogatja



ConstraintLayout

- Továbbgondolt RelativeLayout
- iOS AutoLayout-hoz hasonló
- Fő paraméterek:
 - > Relative positioning
 - > Margins
 - > Centering positioning
 - > Visibility behavior
 - > Dimension constraints
 - > Chains
 - > Virtual Helpers objects



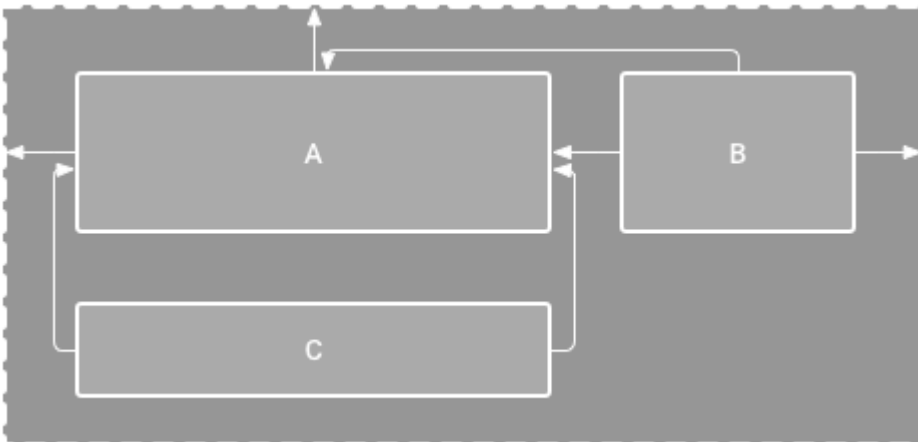
Reszponzív felületek ConstraintLayout-al

- Összetett, komplex layout-ok flat view hierachiával
 - > Nincs szükség egymásba ágyazott layout-okra
- RelativeLayout-hoz hasonló
- Layout Editor támogatás
- Támogatás Android 2.3-tól (API Level 9)
- Komplex példák:
 - > <https://github.com/googlesamples/android-ConstraintLayoutExamples>

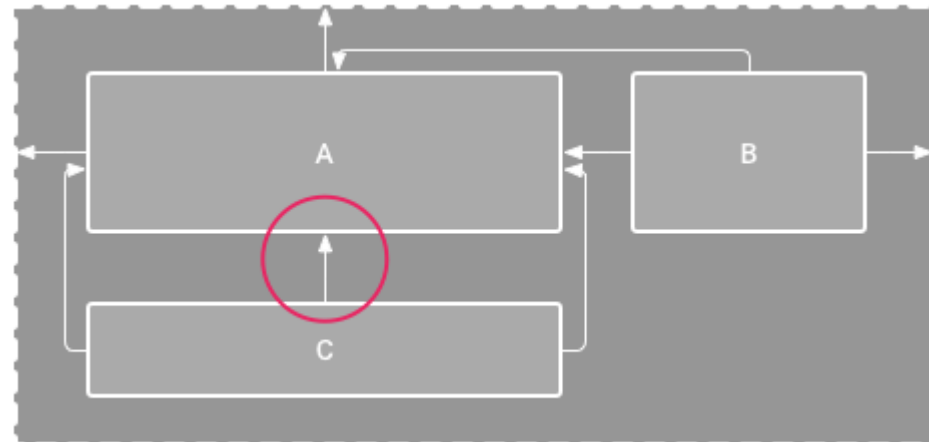
Áttekintés

- Pozíció megadáshoz szükséges:
 - > Horizontális és vertikális „szabály” (constraint)
- Minden szabály egy kapcsolat (connection)/igazítás (alignment):
 - > Egy másik view-hez képest
 - > Szülőhöz képest
 - > Egy láthatatlan sorvezetőhöz (guideline) képest
- Attól még, hogy a *LayoutEditor*-ban jól néz ki, nem biztos, hogy eszközön is jó lesz
- Android Studio jelzi a hiányzó szabályokat

Hibás:

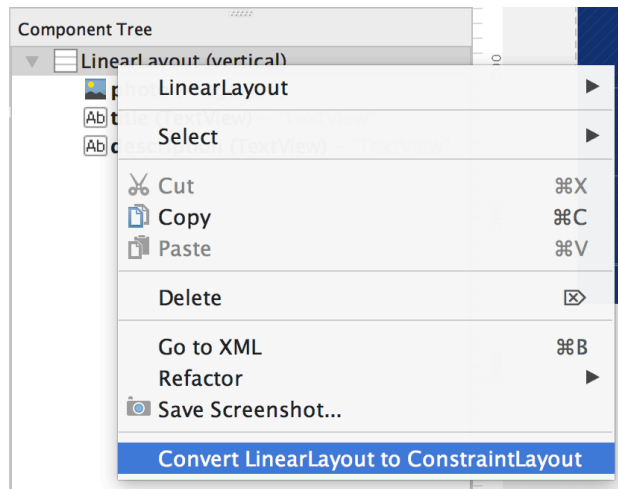


Helyes, mert C tudja, hogy A alatt van:



ConstraintLayout eszközök

- Gradle import:
 - > compile 'com.android.support.constraint:constraint-layout:1.0.2'
- Automatikus átalakítás
 - > Nem tökéletes...

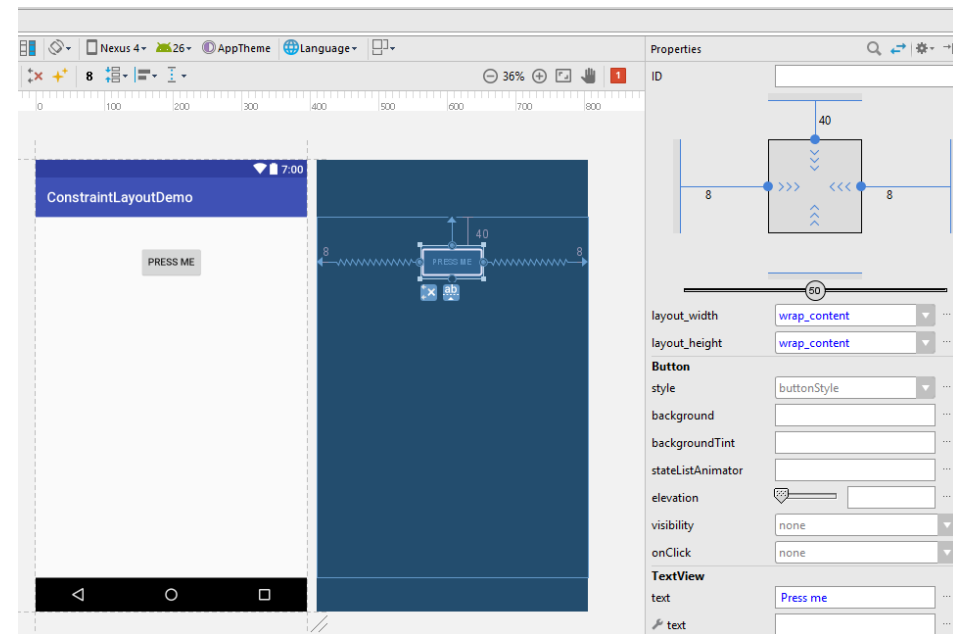


ConstraintLayout használat


- Kötelező legalább egy horizontális és vertikális „szabály”

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Press me"
        app:layout_constraintLeft toLeftOf="parent"
        android:layout_marginLeft="8dp"
        app:layout_constraintRight toRightOf="parent"
        android:layout_marginRight="8dp,,
        app:layout_constraintTop toTopOf="parent"
        android:layout_marginTop="40dp"
    />
</android.support.constraint.ConstraintLayout>
```



Mekkora lesz a gomb mérete?

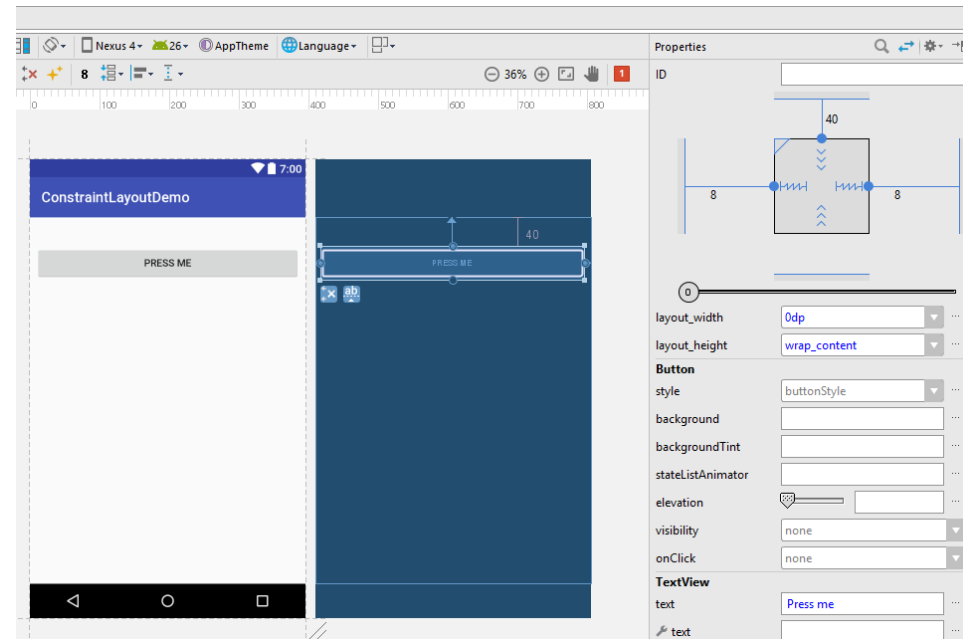
- Nem egyértelmű a szélesség, ellentétes szabályok, jele: 
- > Kettő közé helyezi
- Helyette automata méretezés:
 - > `android:layout_width="0dp"`

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
```

```
    <Button
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:text="Press me"
        app:layout_constraintLeft toLeftOf="parent"
        android:layout_marginLeft="8dp"
        app:layout_constraintRight toRightOf="parent"
        android:layout_marginRight="8dp"
        app:layout_constraintTop toTopOf="parent"
        android:layout_marginTop="40dp"
```

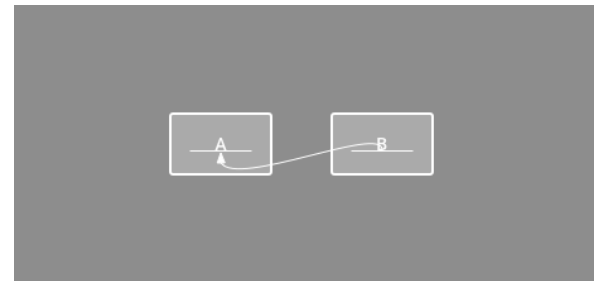
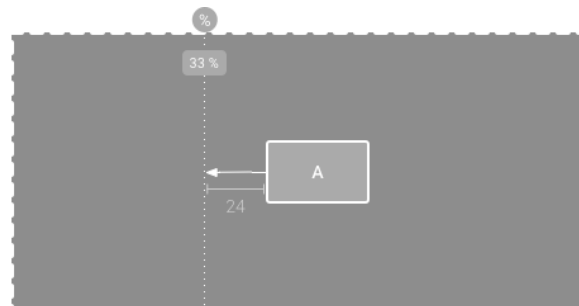
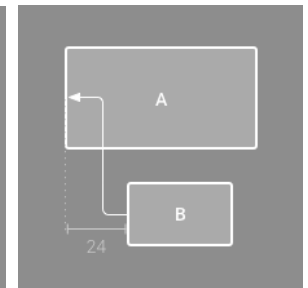
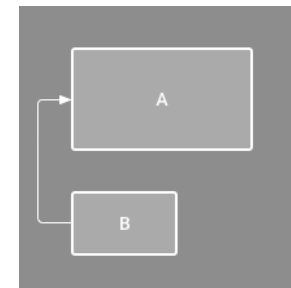
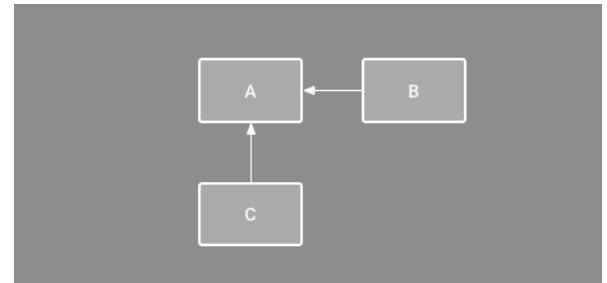
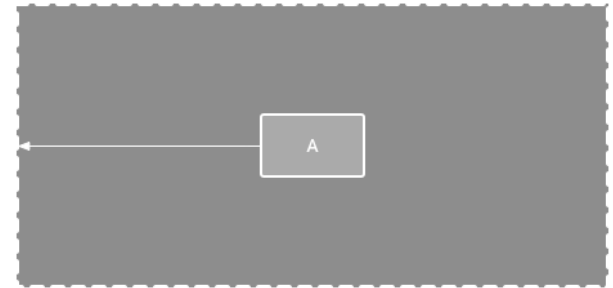
```
    />
```

```
</android.support.constraint.ConstraintLayout>
```



Constraint lehetőségek

- Szülőhöz képest
- Másik View széleihez képest
- Másik View alapvonalához képest
- Guidelinehez (láthatatlan vezetővonalhoz)



ConstraintLayout teljesítmény

- <https://android-developers.googleblog.com/2017/08/understanding-performance-benefits-of.html>

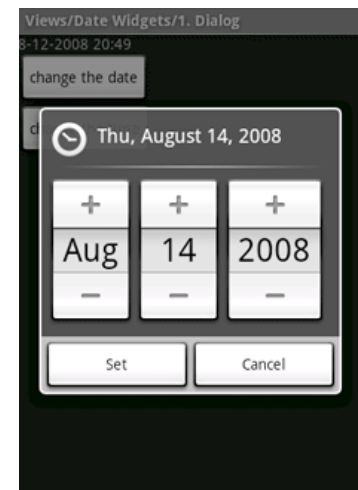
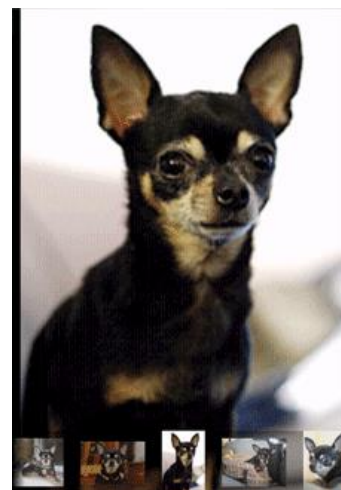
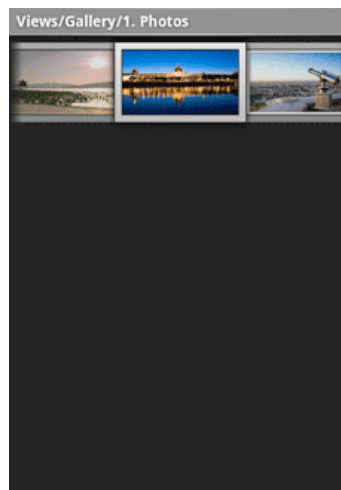
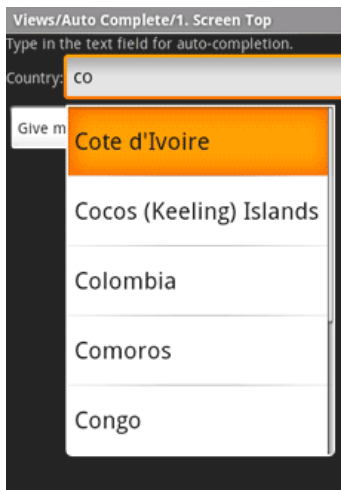
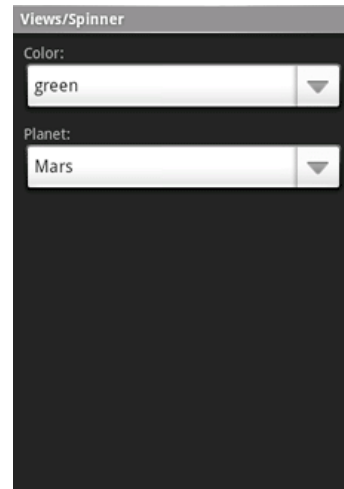
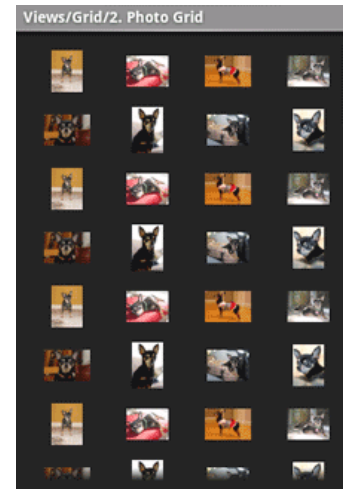
Nézetek (Widgetek/"View"-k)

View-k 1/2



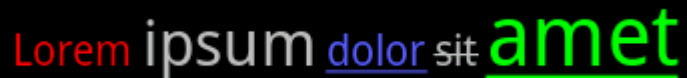
- *Button, EditText, CheckBox, RadioButton, ToggleButton*
- ImageButton*
- ListView*
- GridView*
- Spinner*
- AutoCompleteTextView*
- Gallery*
- ImageSwitcher*
- DatePicker, TimePicker*

View-k 2/2



API gazdagsága (globálisan igaz az Androidra)

- Hogy valósítanak ezt meg? (nem sok *TextView* egymás után😊)



Lorem ipsum dolor sit amet

- Megoldás:
 - > <http://developer.android.com/reference/android/text/SpannableString.html>
 - > <http://androidcocktail.blogspot.hu/2014/03/android-spannablestring-example.html>
- iOS? - NSAttributedString

Egyedi nézetek

- View leszármazott
- Beépített nézetek és *LayoutGroup*-ok is felüldefiniálhatók, pl. saját nézet *RelativeLayout*-ból leszármaztatva
- *<merge>* XML elem
- XML-ek egymásba ágyazhatósága: *<include>*

Egyedi felületi nézet

- Teljesen egyedi felületi elemek definiálása
- Meglévő felületi elemek kiegészítése
- Érintés események kezelése
- Dinamikus rajzolás
 - > Színek, rajzolási stílus
 - > Gyakori alakzatok: vonal, négyzet, kör stb.
 - > Szöveg rajzolása
 - > Képek megjelenítése
- Megjelenítési mérethez való igazodás
- XML-ből is használható!

Dinamikus UI kezelés - LayoutInflater

- LayoutInflater feladata:
 - > XML-ben összeállított felületi elemek példányosítása
- Használati mód:

```
val myView = getLayoutInflater().inflate(  
    R.layout.activity_main, null)
```

Egyedi nézetek – külső könyvtárak

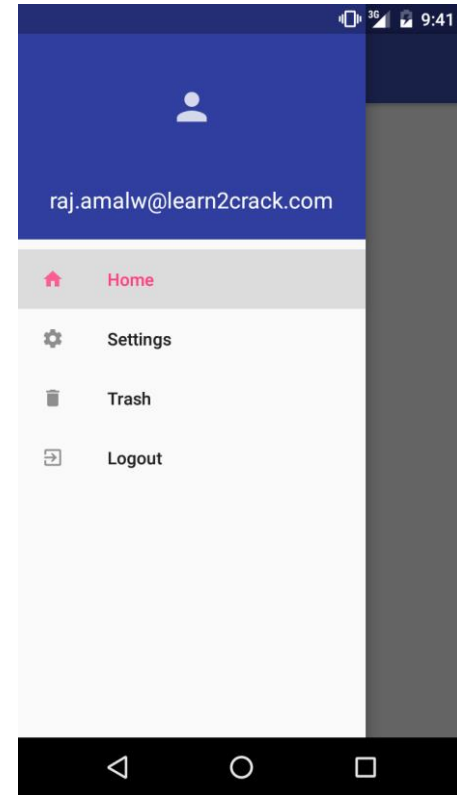
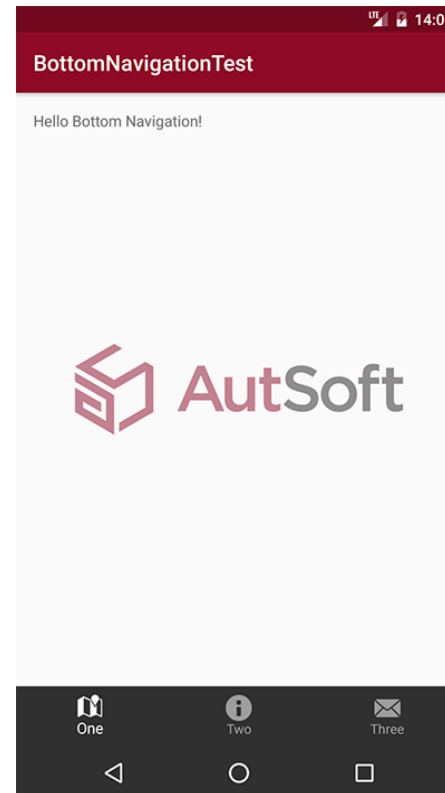
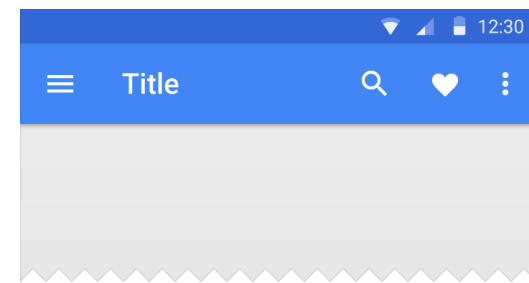
- <https://github.com/wasabeef/awesome-android-ui/>

Menü kezelés

Toolbar, NavigationDrawer, Bottom Navigation Bar

Menü típusok

- „Régi” menü
- „Elavult” ActionBar
- Toolbar
- NavigationDrawer
- Bottom Navigation View



Menük

- Menü típusok:
 - > *ActionBar->Toolbar* része
 - > Navigation Drawer
 - > Bottom Navigation View
- Menü definiálása kódból
- Menü definiálása erőforrásból
- Dinamikus menük
 - > Láthatóság beállítása
 - > Manipuláció Java kódból
- Almenük támogatása

Menü erőforrás

```
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:id="@+id/item1"
        android:title="@string/item1"/>
    <item android:id="@+id/item2"
        android:title="@string/item2"/>
    <item android:id="@+id/submenu"
        android:title="@string/submenu_title">
        <menu>
            <item android:id="@+id/submenu_item1"
                android:title="@string/submenu_item1" />
            <item android:id="@+id/submenu_item2"
                android:title="@string/submenu_item2" />
        </menu>
    </item>
</menu>
```

Menü kezelése

```
override fun onCreateOptionsMenu(menu: Menu): Boolean {  
    menuInflater.inflate(R.menu.menu_main, menu)  
    return super.onCreateOptionsMenu(menu)  
}  
  
override fun onOptionsItemSelected(item: MenuItem): Boolean {  
    when (item.itemId) {  
        R.id.action_start ->  
            Toast.makeText(this, "Start",  
                Toast.LENGTH_SHORT).show()  
        R.id.action_help ->  
            Toast.makeText(this, "Help",  
                Toast.LENGTH_SHORT).show()  
    }  
    return true  
}
```

ActionBar és Menük

- Dedikált alkalmazás menü, logo és cím
- Tipikus felhasználás:
 - > Menü
 - > Branding (logo/background) és alkalmazás ikon
 - > Konzisztens alkalmazás navigáció
 - > Fő funkciók bemutatása



ActionBar specifikus XML menü paraméterek

```
<item android:id="@+id/action_time"  
      android:title="@string/action_show_time"  
      android:orderInCategory="5"  
      android:icon="@drawable/clock_icon"  
      android:showAsAction="always|withText" />
```

ActionBar -> Toolbar

- ActionBar helyett ToolBar
- Sokkal dinamikusabb viselkedés
- Menü erőforrások támogatása
- Custom elemek támogatása
- Manuális pozicionálás
- Toolbar tutorialok:
 - > <http://javatechig.com/android/android-lollipop-toolbar-example>
 - > <http://www.101apps.co.za/index.php/articles/using-toolbars-in-your-apps.html>

Toolbar használat

- ActionBar nélküli téma(styles.xml):

> Theme.AppCompat.NoActionBar

- Layout erőforrás:

```
<android.support.v7.widget.Toolbar
    android:id="@+id/toolbar"
    android:minHeight="?attr/actionBarSize"
    android:background="#2196F3"
    android:layout_width="match_parent"
    android:layout_height="wrap_content">
</android.support.v7.widget.Toolbar>
```

- Activity onCreate(...):

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

    setSupportActionBar(toolbar)
}
```

Összefoglalás

- Felhasználói felület fogalmak
- Erőforrás típusok
- Erőforrásminősítők használata, jellemzőik
- Layout erőforrások (LinearLayout, RelativeLayout)
- Nézetek/View-k
- Képek kezelése egyszerűen
- Menükezelés

A következő alkalommal...

- Telefon és tablet egyidejű támogatása
- Fragment fogalma, Fragment életciklus
- Statikus és dinamikus csatolás
- Dialógus fragmentek
- UI nélküli fragmentek
- Lapozható felületek, tabok
- Komplex menü típusok

Köszönöm a figyelmet!



peter.ekler@aut.bme.hu



AutSoft