

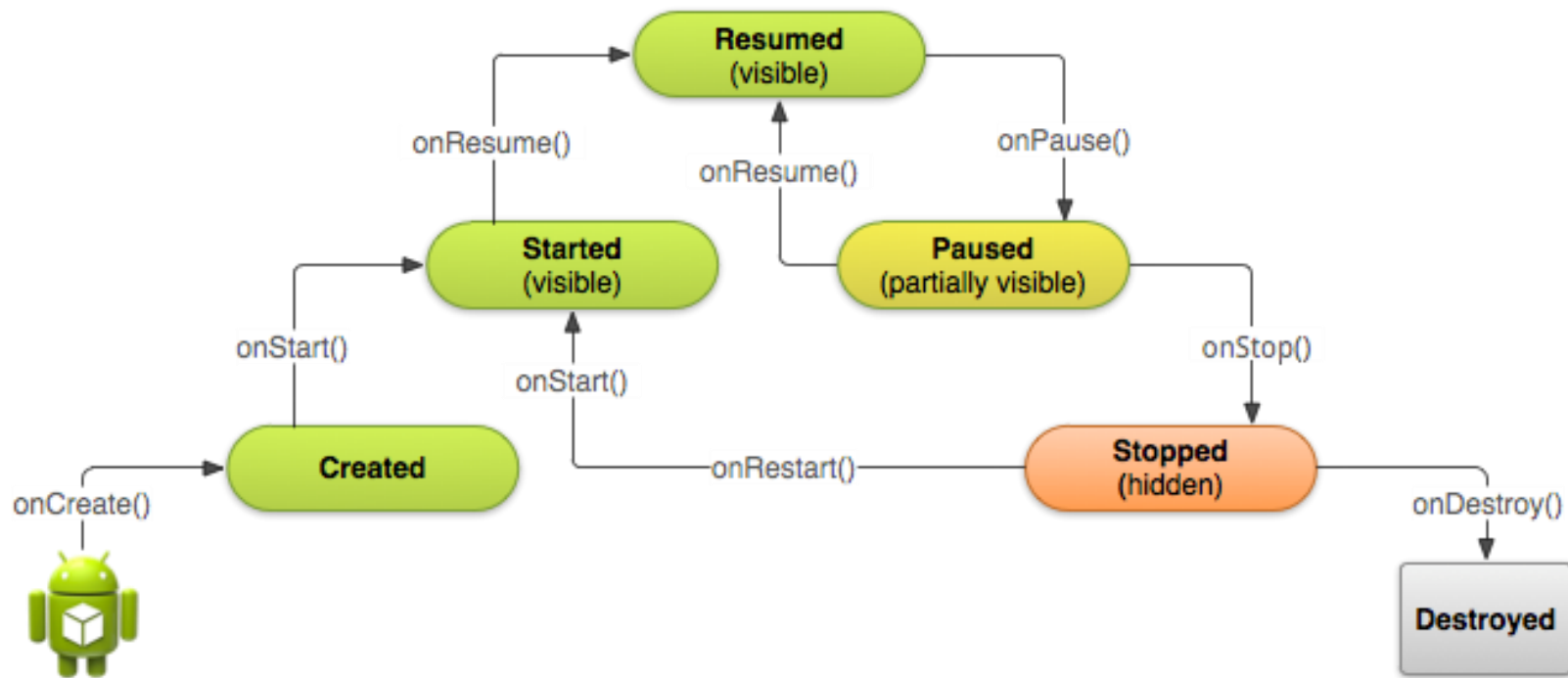
Android

Felhasználói felület, listák

Dr. Ekler Péter

peter.ekler@aut.bme.hu

Activity életciklus



Hogy is volt?

- Egy Android alkalmazás milyen komponensekből épülhet fel?
- Mi a Service komponens?
- Miket kell tartalmaznia a manifest állománynak?
- Az Activity callback élelciklus-függvények felüldefiniálásakor meg kell-e hívni kötelezően az ősz osztály implementációját?
- Ha A Activity-ből átváltunk B Activity-re, milyen sorrendben hívódnak meg az élelciklus függvények?
- Magyarázza el az Activity Back Stack működési elvét!

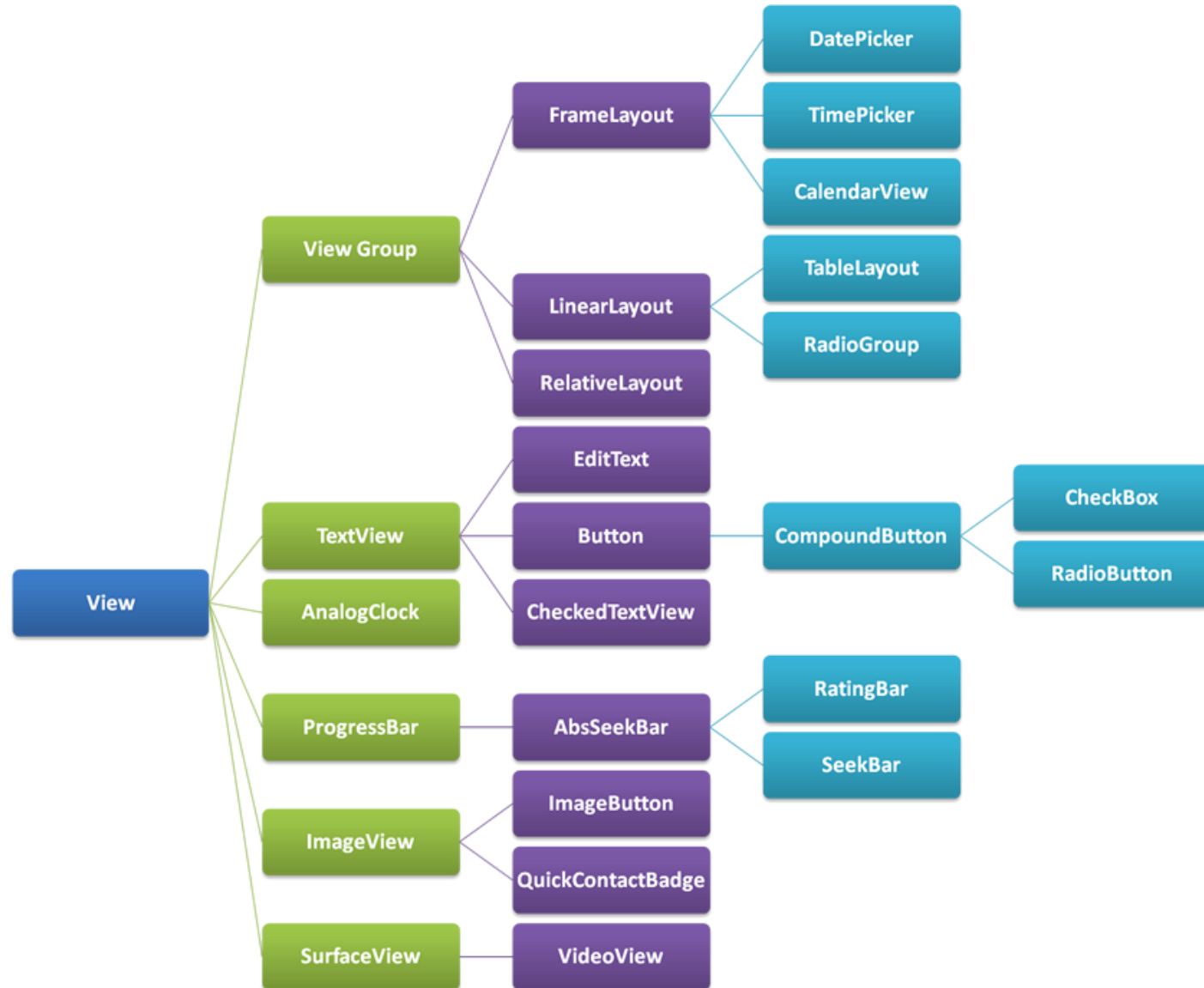
Hogy is volt?

- Mit értünk a sűrűségfüggetlen pixel fogalom alatt?
- Egy 320 dpi-s képernyőn, 1 dp mennyi fizikai pixelnek felel meg ?
- Vázolja fel egy Android alkalmazás kódját, mely egy gombot jelenít meg és a gombot lenyomva a „Clicked!” szöveg jelenik meg egy Toast-ban!
- Hogy biztosítja az Android a lokalizáció támogatását?

Tartalom

- UI építő elemek
 - > Layout (ViewGroup)
 - > View-k
- Android Fragment framework
 - > Mik azok a Fragment-ek, hogyan használjuk?
- Support library
 - > Fragment backport, ViewPager, TabStrip, egyebek
- RecyclerView és egyedi felület elem tervezés
- Felugró ablakok, animációk, rajz erőforrások
- Android felület tervezési javaslatok
- Összefoglalás

Android UI architektúra



Layout-ok

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

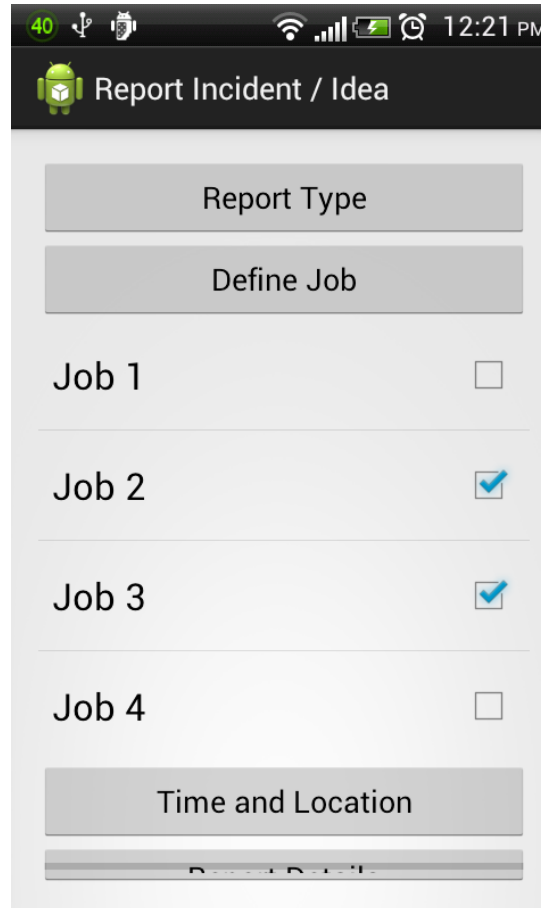
Layout-ok (ViewGroup)

- 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: 'Report Type' (disabled), 'Define Job' (disabled), 'Job 1' (unchecked), 'Job 2' (checked), 'Job 3' (checked), 'Job 4' (unchecked), and 'Time and Location' (disabled). The list is separated by thin horizontal lines.

Item	Checkbox
Report Type	<input type="checkbox"/>
Define Job	<input type="checkbox"/>
Job 1	<input type="checkbox"/>
Job 2	<input checked="" type="checkbox"/>
Job 3	<input checked="" type="checkbox"/>
Job 4	<input type="checkbox"/>
Time and Location	<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

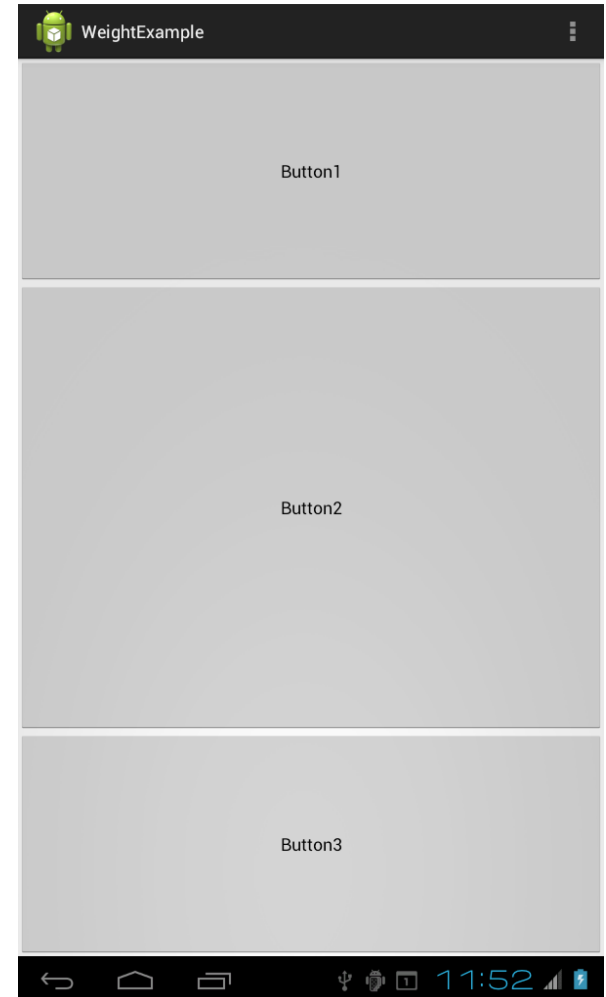
```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:weightSum="4"
    android:orientation="vertical">

    <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" />

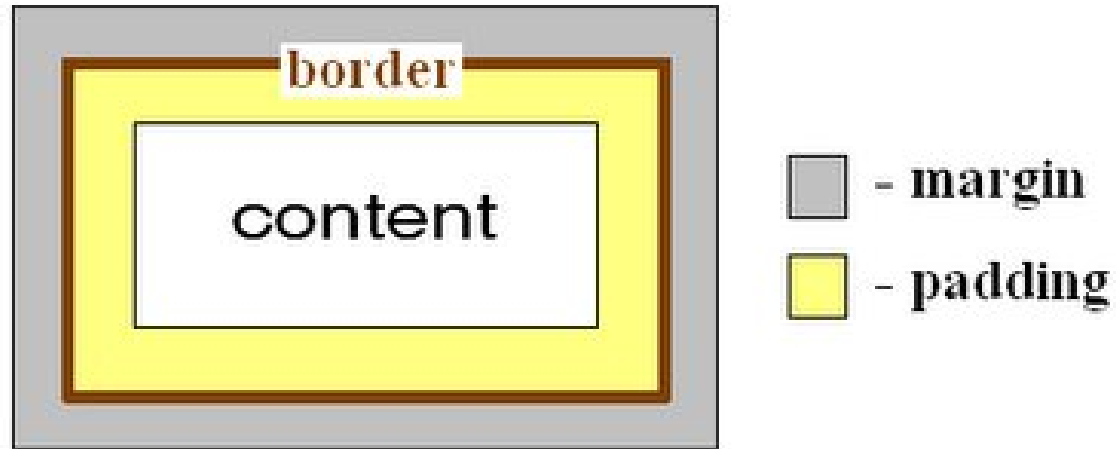
</LinearLayout>
```



LinearLayout példák

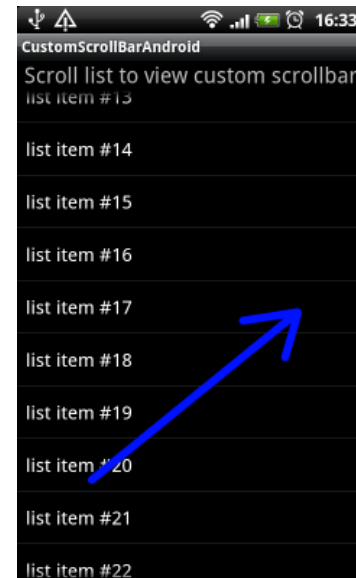
- Jellemző paraméterek:
 - > Margin, padding
 - > Gravity
 - > ScrollView
 - > Weight

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

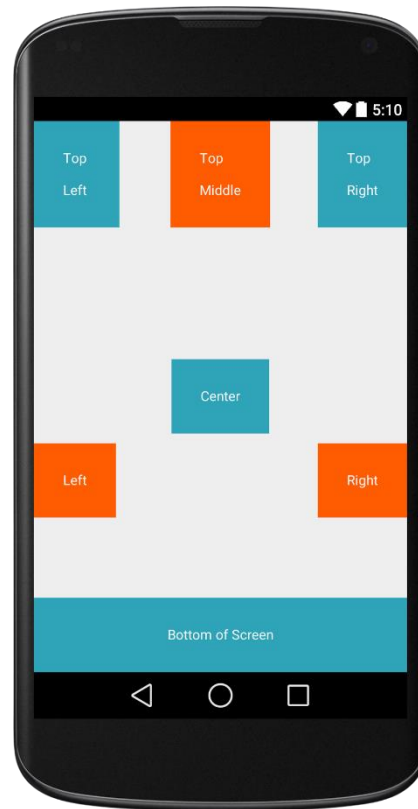
```
<ScrollView xmlns:android=
    "http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="10dp"
    android:fillViewport="false">

    <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



CONSTRAINTLAYOUT

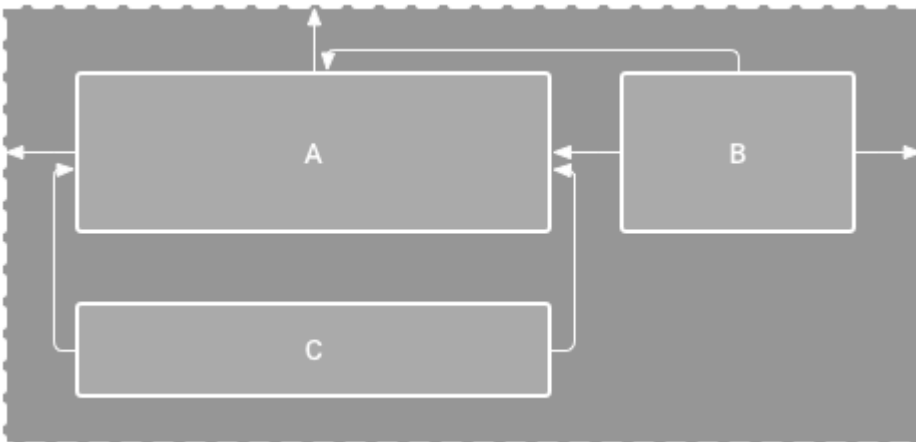
Reszponzív felületek ConstraintLayout-al

- 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>

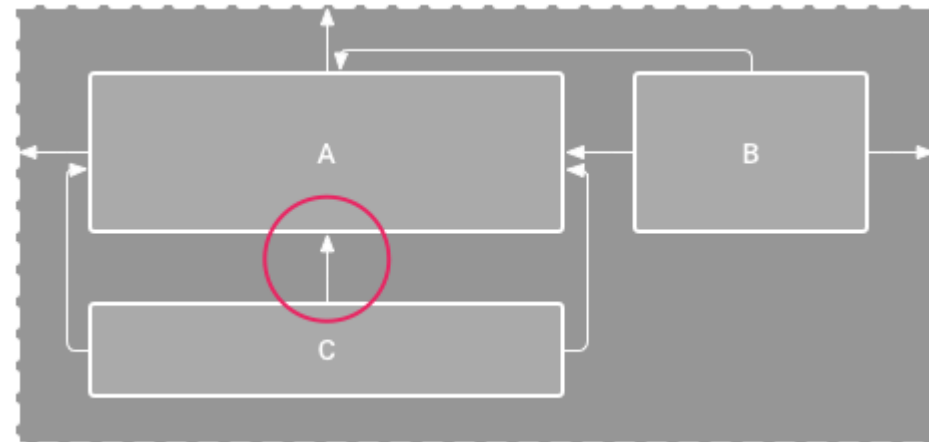
Áttekintés

- 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:

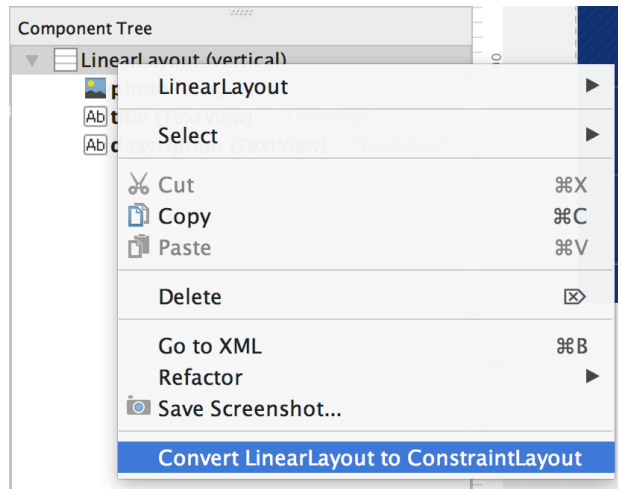


Helyes, mert C tudja, hogy A alatt van:



ConstraintLayout eszközök

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

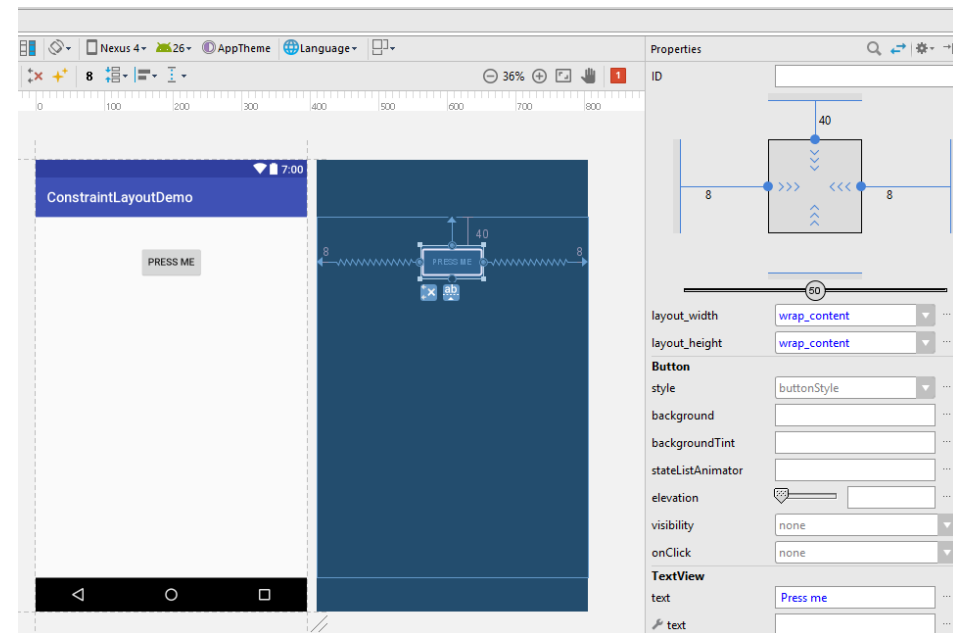


ConstraintLayout használat


- 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?

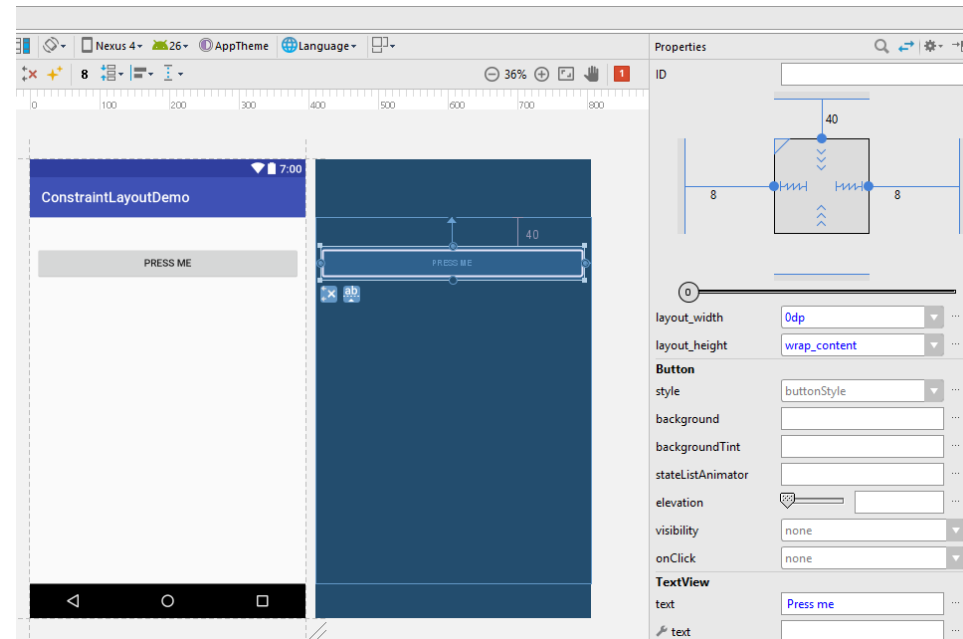
- Nem egyértelmű a szélesség, ellentétes szabályok, jele: 
- > Kettő közé helyezi
- Helyette automata méretezés:
 - > `android:layout_width="0dp"`

```
<?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"
```

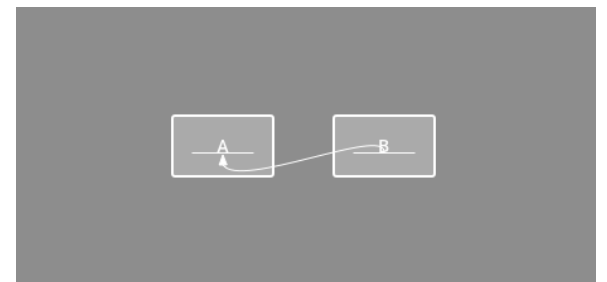
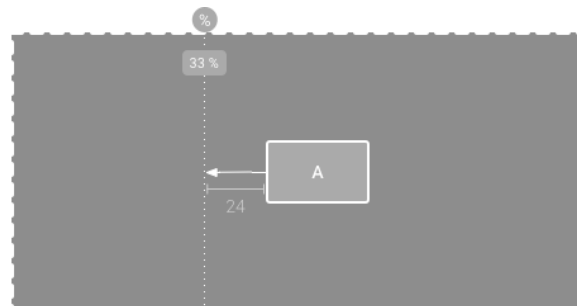
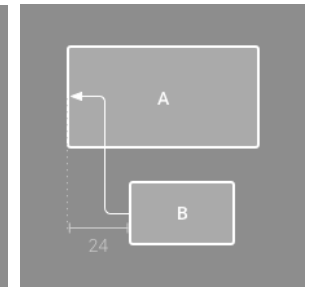
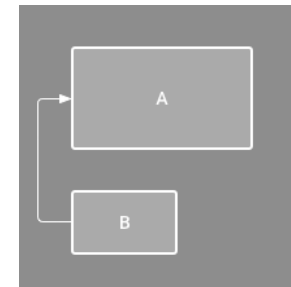
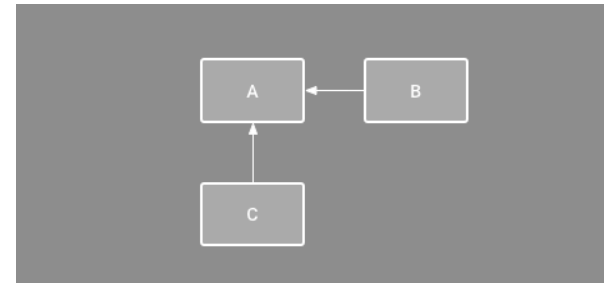
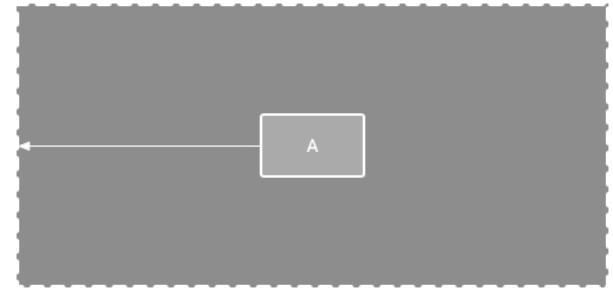
```
    />
```

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



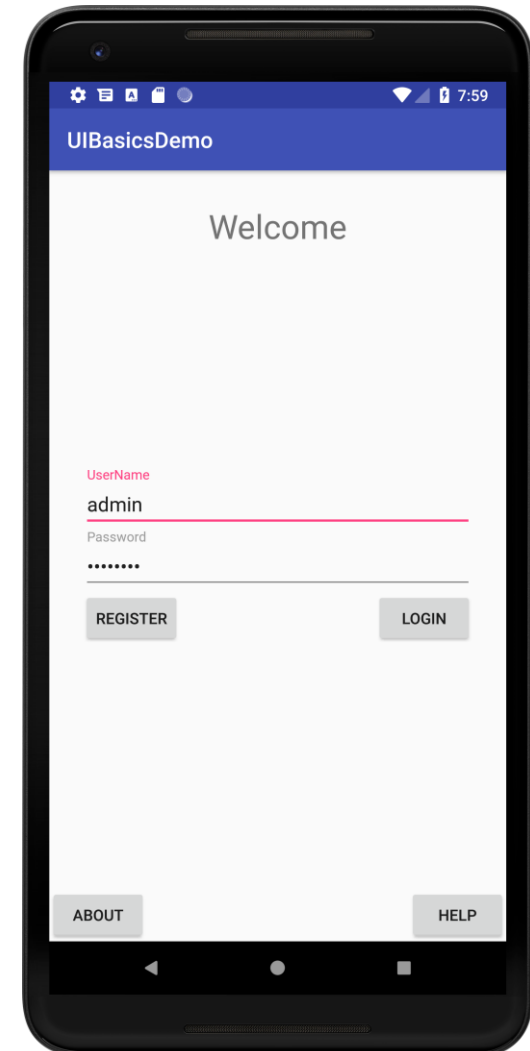
Constraint lehetőségek

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



Gyakoroljunk!

- Készítsük el az alábbi *Login* képernyőt!

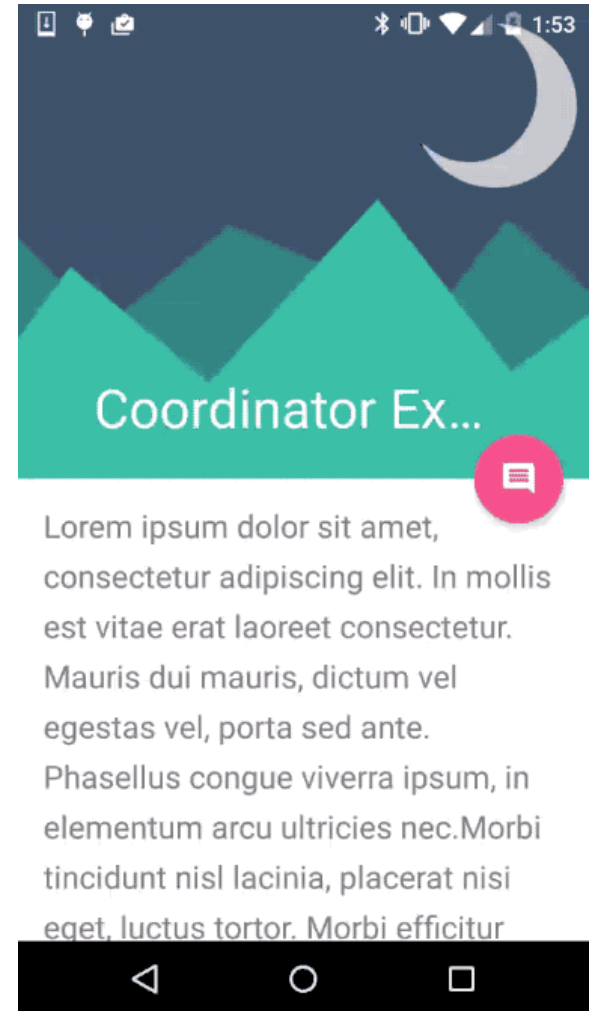


ConstraintLayout részletek

- Dokumentáció:
 - > <https://developer.android.com/training/constraint-layout/>
- Teljesítmény:
 - > <https://android-developers.googleblog.com/2017/08/understanding-performance-benefits-of.html>

CoordinatorLayout, AppBarLayout

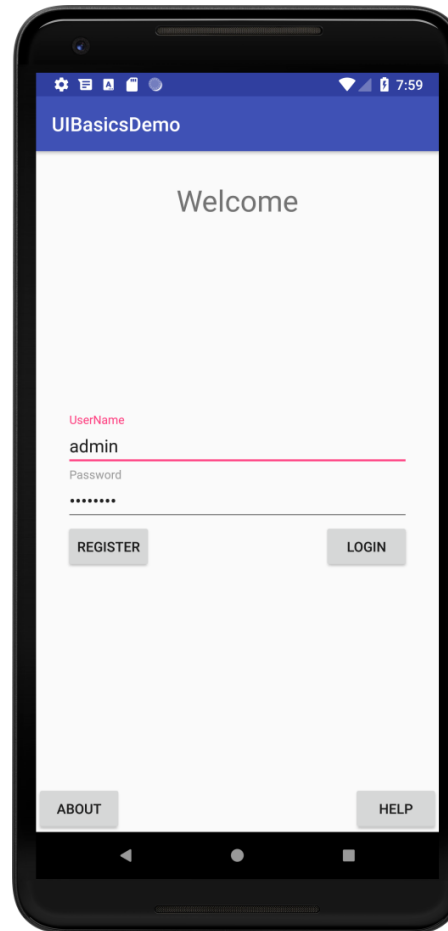
- CoordinatorLayout: továbbfejlesztett FrameLayout
- CoordinatorLayout fő feladatai:
 - > Felső szintű alkalmazás UI irányelv
 - > Konténer, mely támogatja a beépített elemek material stílushoz igazodó elhelyezkedését
- Behavior paraméterekkel meghatározható a kapcsolódó elemek elhelyezése
- AppBarLayout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



Gyakoroljunk



- Készítsünk egy Login képernyőt



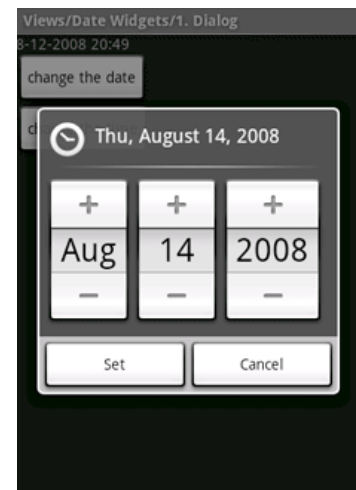
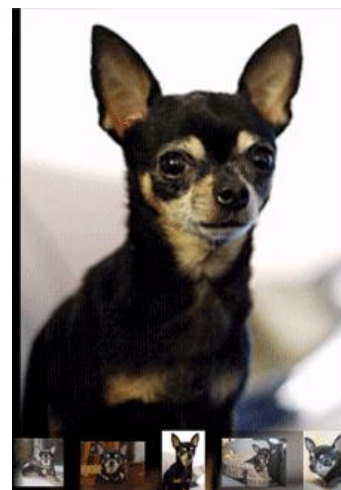
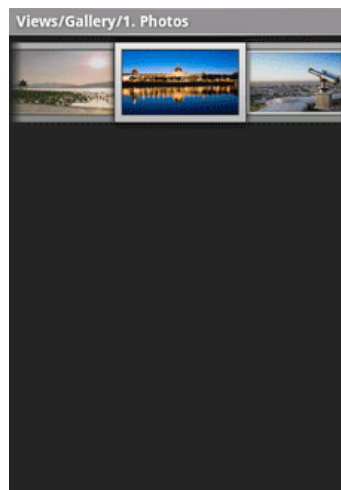
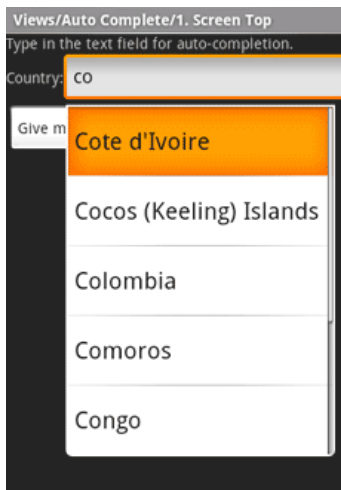
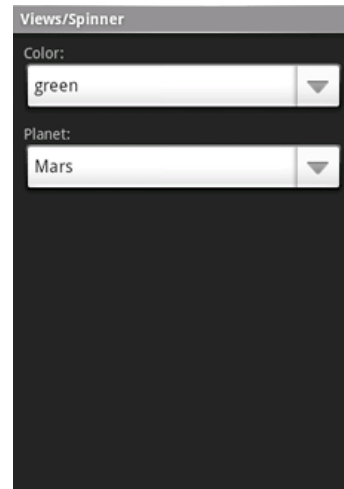
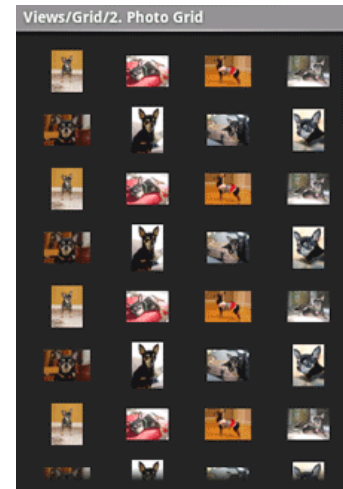
Nézetek (Widgetek/"View"-k)

View-k 1/2



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

View-k 2/2



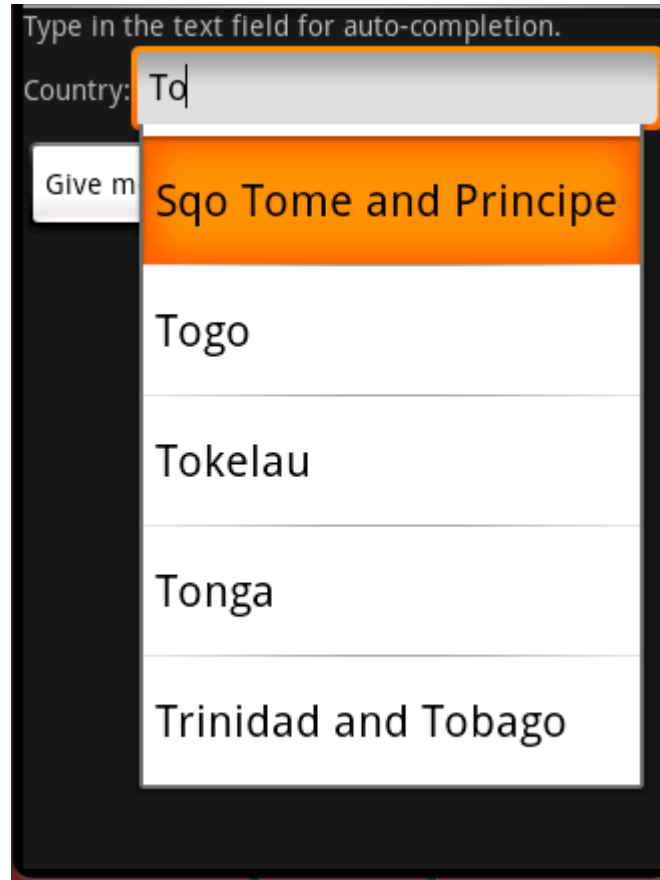
API gazdagsága (globálisan igaz az Androidra)

- 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>

AutoCompleteTextView 1/3



AutoCompleteTextView 2/3

```
<?xml version="1.0" encoding="utf-8"?>

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >

    <AutoCompleteTextView
        android:id="@+id/autoCompleteTextViewCities"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="" >

        <requestFocus />

    </AutoCompleteTextView>

</LinearLayout>
```

AutoCompleteTextView 3/3

```
class MainActivity : AppCompatActivity() {  
  
    private val cityNames = arrayOf("Budapest", "Bukarest", "New York",  
    "New Delhi")  
  
    override fun onCreate(savedInstanceState: Bundle?) {  
        super.onCreate(savedInstanceState)  
        setContentView(R.layout.activity_main)  
  
        val cityAdapter = ArrayAdapter(  
            this,  
            android.R.layout.simple_dropdown_item_1line, cityNames  
        )  
        autoCompleteTextViewCities.setAdapter(cityAdapter)  
  
    }  
  
}
```

RadioButton - demo

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:layout_margin="20dp"
    tools:context=".MainActivity">

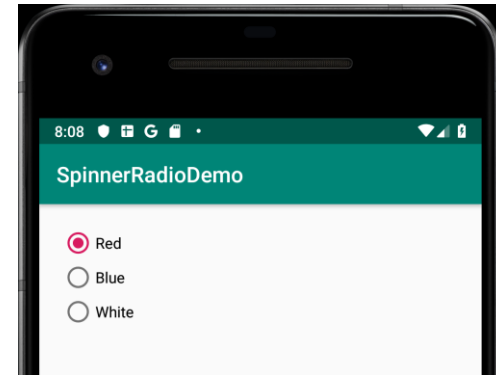
    <RadioGroup
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="vertical">

        <RadioButton
            android:id="@+id/btnRed"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:checked="true"
            android:text="Red" />

        <RadioButton
            android:id="@+id/btnBlue"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Blue" />

        <RadioButton
            android:id="@+id/btnWhite"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="White" />

    </RadioGroup>
</LinearLayout>
```



```
class MainActivity : AppCompatActivity() {

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

        btnRed.setOnClickListener{
            Toast.makeText(this@MainActivity,
                "Red", Toast.LENGTH_LONG).show()
        }
        btnBlue.setOnClickListener{
            Toast.makeText(this@MainActivity,
                "Blue", Toast.LENGTH_LONG).show()
        }
        btnWhite.setOnClickListener{
            Toast.makeText(this@MainActivity,
                "White", Toast.LENGTH_LONG).show()
        }
    }
}
```

Spinner - demo

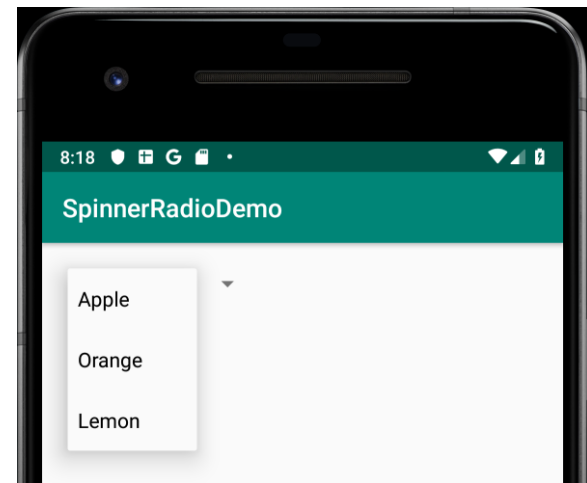
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:layout_margin="20dp"
    tools:context=".MainActivity">

    <Spinner
        android:id="@+id/spinnerFruits"
        android:layout_width="150dp"
        android:layout_height=
            "wrap_content" />

</LinearLayout>
```

res/values/fruits_array.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string-array name="fruits_array">
        <item>Apple</item>
        <item>Orange</item>
        <item>Lemon</item>
    </string-array>
</resources>
```



```
class MainActivity : AppCompatActivity(),
    AdapterView.OnItemClickListener {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        val fruitsAdapter = ArrayAdapter.createFromResource(
            this,
            R.array.fruits_array, android.R.layout.
                simple_spinner_item
        )
        fruitsAdapter.setDropDownViewResource(
            android.R.layout.simple_spinner_dropdown_item)
        spinnerFruits.adapter = fruitsAdapter
        spinnerFruits.onItemSelectedListener = this
    }
    override fun onNothingSelected(parent: AdapterView<*>?) {
    }
    override fun onItemSelected(parent: AdapterView<*>?,
        view: View?, position: Int, id: Long) {
        Toast.makeText(this, parent.getItemAtPosition(
            position).toString(), Toast.LENGTH_LONG).show()
    }
}
```

Egyedi nézetek – külső könyvtárak

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

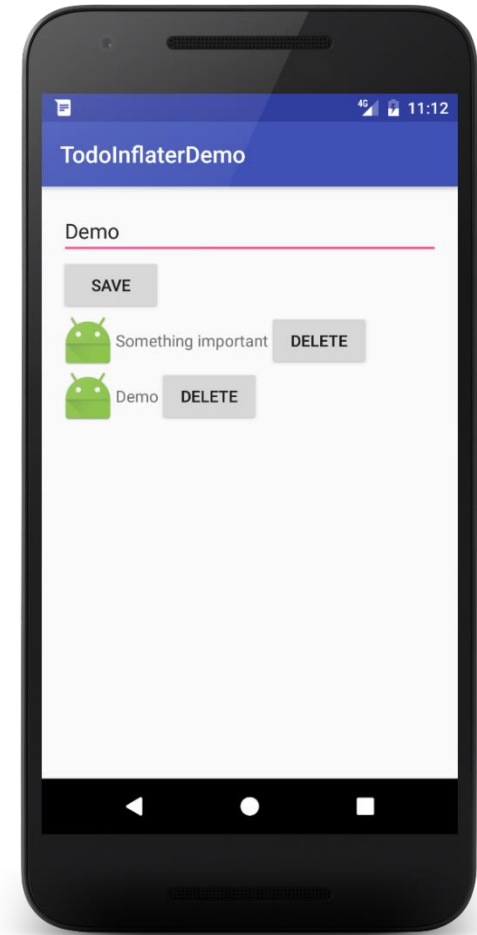
Dinamikus UI kezelés - LayoutInflater

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

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

Dinamikus UI példa

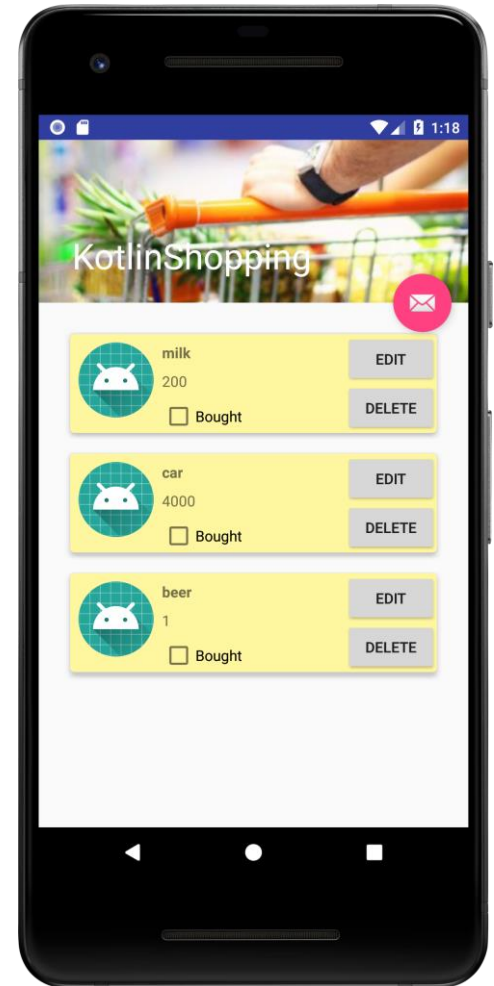
- Készítsünk egy egyszerű TODO alkalmazást



LISTÁK KEZELÉSE

RecyclerView

- Listák hatékony kezelése
- Gyors scrollozás
- Általános érintés gesztusok támogatása (swipe, move, stb.)
- *ViewHolder* minta a gyors működés érdekében
- Hatékony elem újrafelhasználás
- *Flexibilis*



RecyclerView.Adapter<ViewHolder> 1/3

- Inicializálás, konstruktor

```
private val context: Context
private val items: MutableList<ShoppingItem> = mutableListOf<ShoppingItem> (
    ShoppingItem("milk", 200, false),
    ShoppingItem("car", 4000, false),
    ShoppingItem("beer", 1, false)
)

constructor(context: Context) : super() {
    this.context = context
}
```

- Egy sor nézetének beállítása: onCreateViewHolder

```
override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): ViewHolder {
    val view = LayoutInflater.from(parent.context).inflate(
        R.layout.row_item, parent, false
    )
    return ViewHolder(view)
}
```

ViewHolder implementáció

```
class ViewHolder(itemView: View?) : RecyclerView.ViewHolder(itemView) {  
    val tvName = itemView.tvName  
    val tvPrice = itemView.tvPrice  
    val cbBought = itemView.cbBought  
    val btnEdit = itemView.btnEdit  
}
```

RecyclerView.Adapter<ViewHolder> 2/3

- Sorban levő elemek értékeinek beállítása
- Eseménykezelők beállítása
- ViewHolder binding

```
override fun onBindViewHolder(holder: ViewHolder, position: Int) {  
    val (name, price, bought) = items[holder.adapterPosition]  
    holder.tvName.text = name  
    holder.tvPrice.text = price.toString()  
    holder.cbBought.isChecked = bought  
  
    holder.btnEdit.setOnClickListener{  
        (context as MainActivity).showEditTodoDialog(items[holder.adapterPosition])  
    }  
}
```

RecyclerView.Adapter<ViewHolder> 3/3

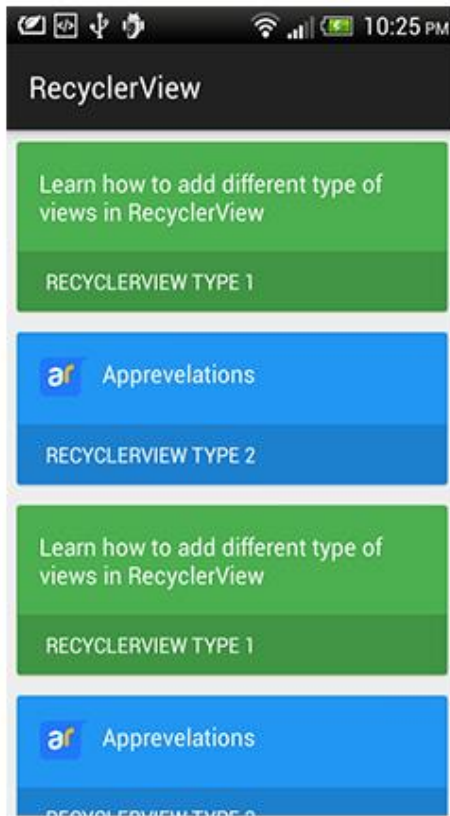
- Elemek száma, hozzáadás, törlés

```
override fun getItemCount() = items.size
```

```
fun addItem(item: ShoppingItem) {  
    items += item  
    notifyItemInserted(items.lastIndex)  
}
```

```
private fun deleteItemBasedOnPosition(position: Int) {  
    items.removeAt(position)  
    notifyItemRemoved(position)  
}
```

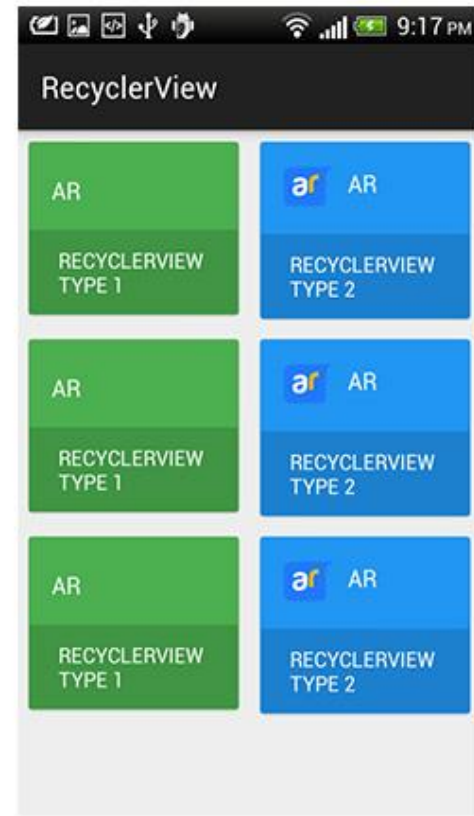
RecyclerView LayoutManager-ek



Linear Layout View



Staggered Grid View



Grid View

Különböző elem megjelenítés a RecyclerView-ban

- Elemek/sorok típusa megadható pozíció alapján a `getItemViewType(...)` felüldefiniálásával
- `viewType` paraméter jelzi a megfelelő függvényekben a sor/elem típusát, amely alapján a megjelenítés szabályozható
- `ViewHolder` ismeri a `viewType`-jét

```
// determine which layout to use for the row
@Override
public int getItemViewType(int position) {
    Item item = itemList.get(position);
    if (item.getType() == Item.ItemType.ONE_ITEM) {
        return TYPE_ONE;
    } else if (item.getType() == Item.ItemType.TWO_ITEM) {
        return TYPE_TWO;
    } else {
        return -1;
    }
}

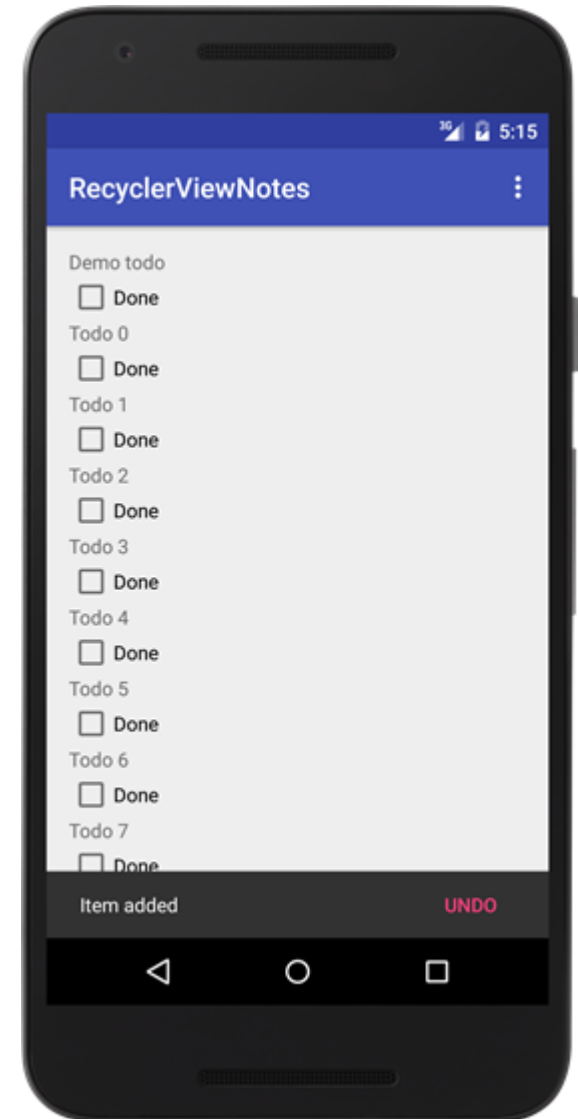
// specify the row layout file and click for each row
@Override
public RecyclerView.ViewHolder onCreateViewHolder(
    ViewGroup parent, int viewType) {
    if (viewType == TYPE_ONE) {
        View view =
            LayoutInflater.from(parent.getContext()).inflate(
                R.layout.list_item_type1, parent, false);
        return new ViewHolderOne(view);
    } else if (viewType == TYPE_TWO) {
        View view =
            LayoutInflater.from(parent.getContext()).inflate(
                R.layout.list_item_type2, parent, false);
        return new ViewHolderTwo(view);
    } else {
        throw new RuntimeException(
            "The type has to be ONE or TWO");
    }
}
```


Lista készítés – fő lépések

1. Data class
2. Egy sor layout-ja
3. RecyclerView – lista hol legyen
4. Adapter – megmondja hogy mi legyen a RecyclerView-ba

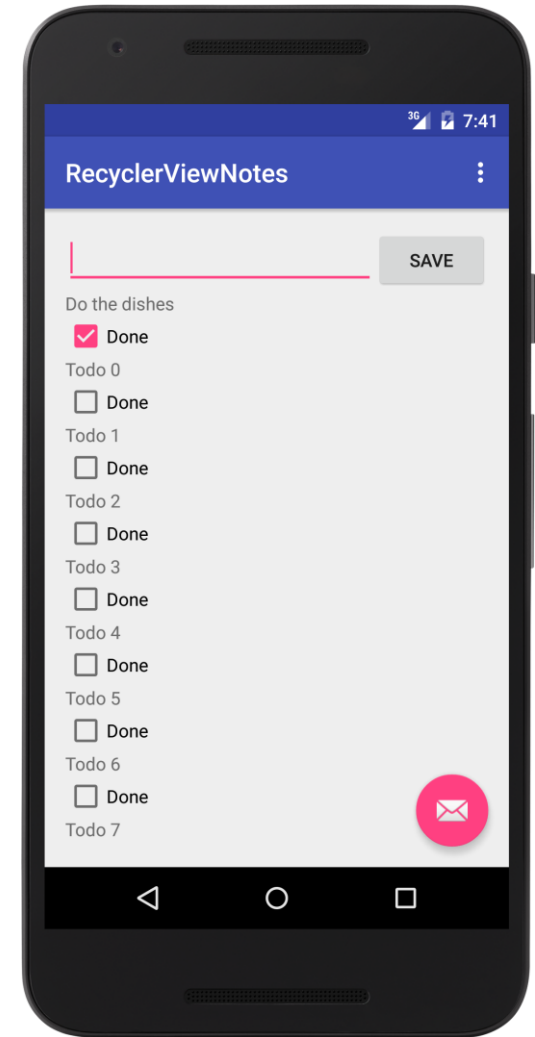
Gyakoroljunk!

- Készítsünk egy Todo alkalmazást RecyclerView-val
- Jelenítsünk meg egy checkbox-ot minden elem előtt
- Valósítsunk meg hozzáadás után egy undo lehetőséget



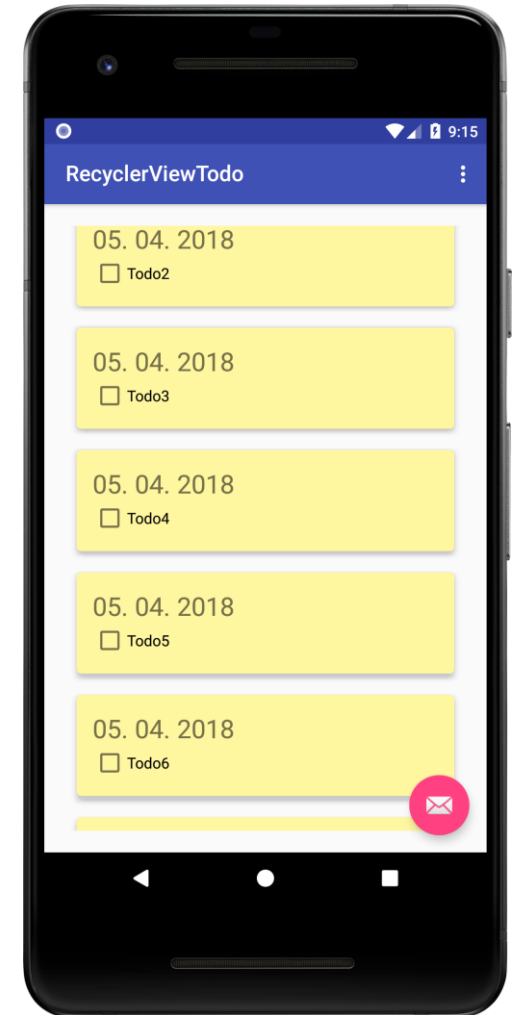
Gyakoroljunk!

- Egészítsük ki az alkalmazást Todo hozzáadása funkcióval
- Alternatívaként a *DialogFragment* is kipróbálható



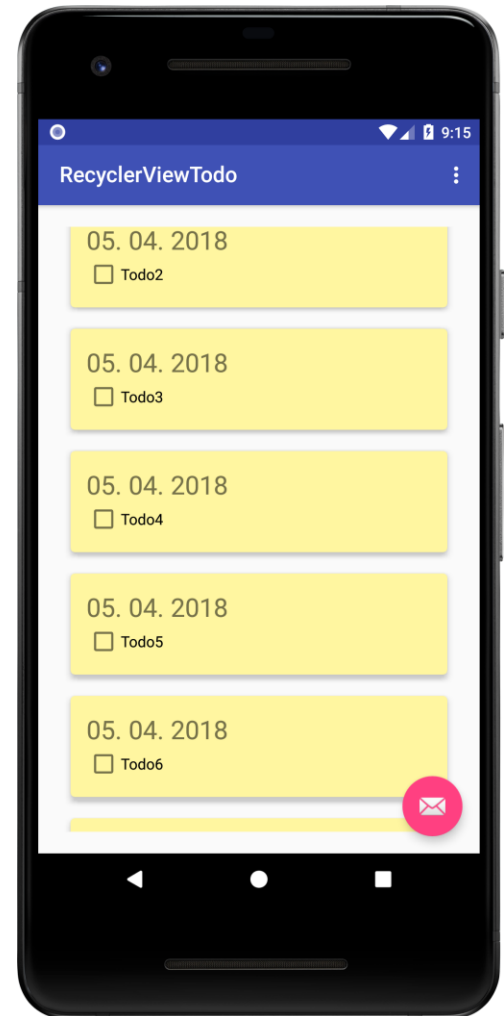
Gyakoroljunk

- Készítsünk egy Todo alkalmazást!



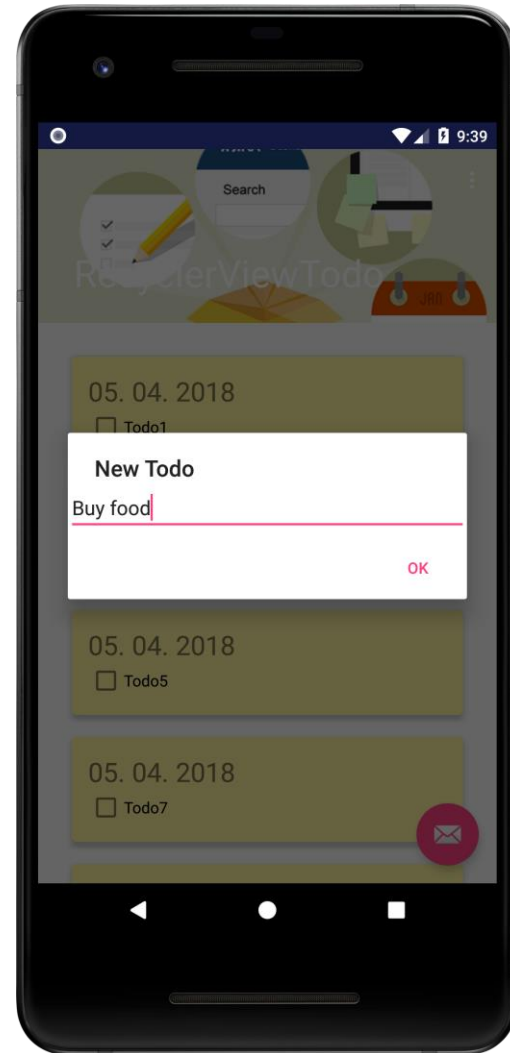
CardView

- Adat megjelenítő „konténer”



Új Todo - dialógussal

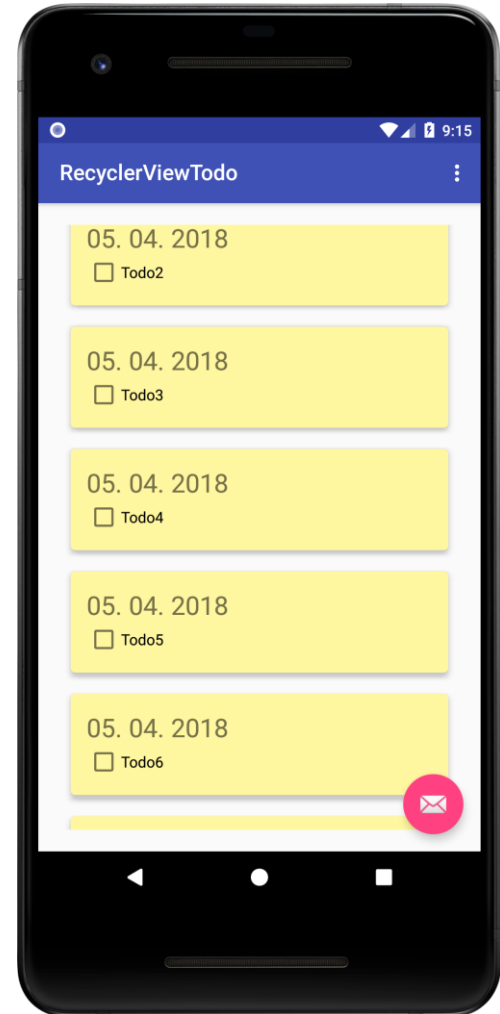
```
val builder = AlertDialog.Builder(this@MainActivity)
builder.setTitle("Enter Todo")
val input = EditText(this@MainActivity)
builder.setView(input)
builder.setPositiveButton("OK", { dialog, which ->
    todoRecyclerAdapter.addToDo(
        Todo(input.text.toString(), false))
    recyclerTodo.scrollToPosition(0)
})
builder.setNegativeButton("Cancel") {
    dialog, which -> dialog.cancel()
}
builder.show()
```



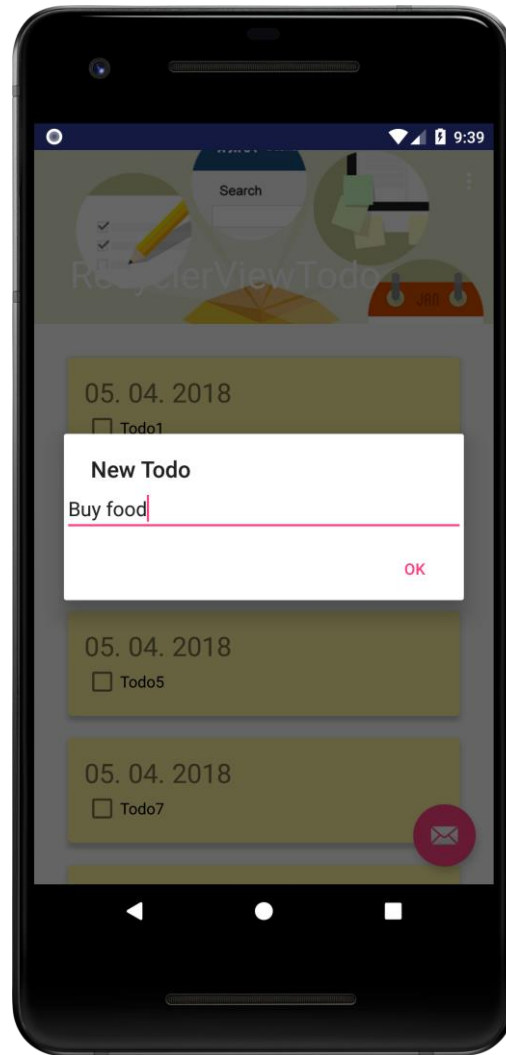
CardView demo

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.v7.widget.CardView
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    xmlns:card_view="http://schemas.android.com/apk/res-auto"
    android:id="@+id/card_view"
    android:layout_margin="10dp"
    card_view:cardBackgroundColor="#fff6a0"
    card_view:cardCornerRadius="4dp"
    card_view:cardElevation="4dp">
    <LinearLayout
        android:orientation="vertical"
        android:layout_width="match_parent"
        android:layout_height="wrap_content">
        <TextView
            android:id="@+id/tvDate"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="date"
            android:textSize="24sp"/>

        ...
    </LinearLayout>
</android.support.v7.widget.CardView>
```



Új Todo - DialogFragment



Elválasztó vonalak

```
val itemDecoration = DividerItemDecoration(this,  
    DividerItemDecoration.VERTICAL)  
recyclerTodo.addItemDecoration(itemDecoration)
```



Dany Targaryen Valyria



Rob Stark Winterfell



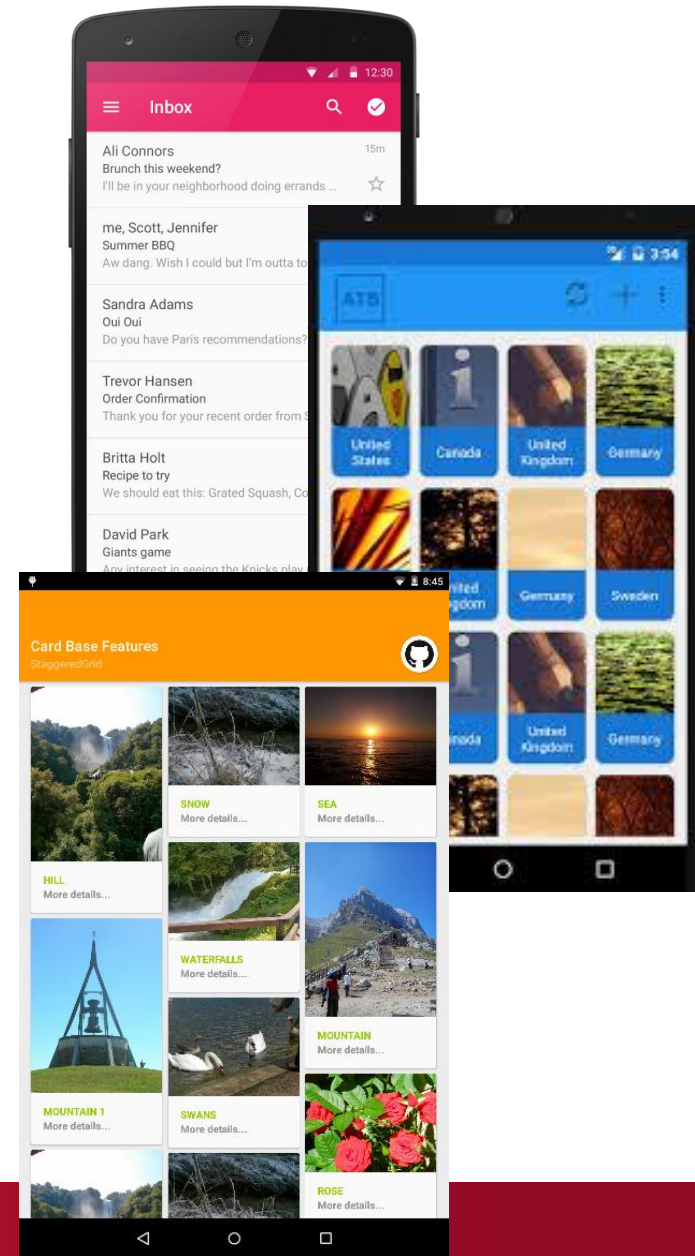
Jon Snow Castle Black



Tyrion Lanister King's Landing

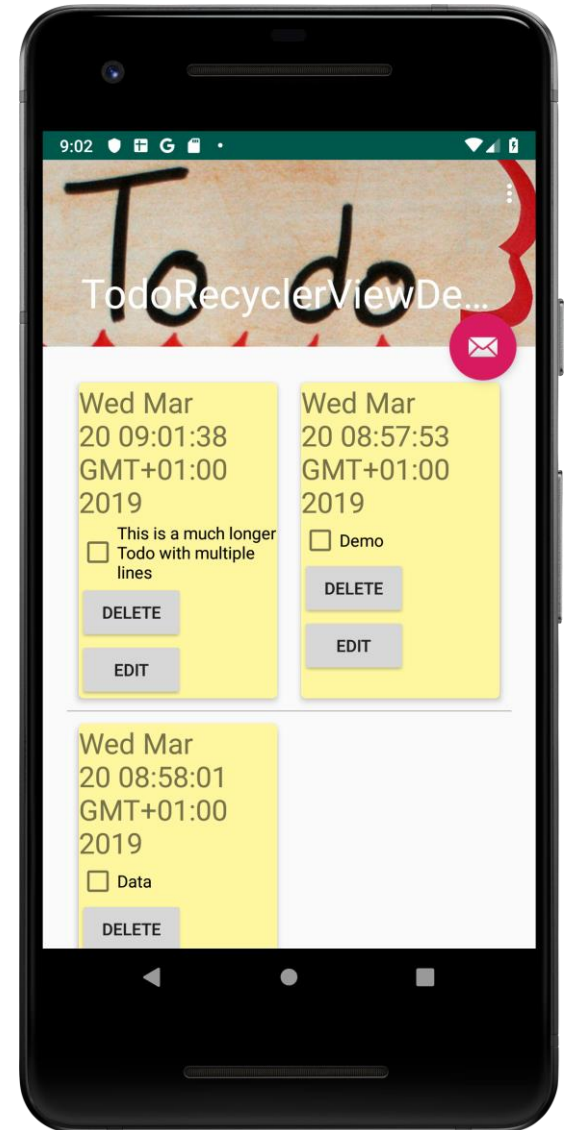
RecyclerView – LayoutManager típusok

- LinearLayoutManager
- GridLayoutManager
- StaggeredGridLayoutManager



GridLayoutManager demo

```
recyclerTodo.layoutManager =  
    GridLayoutManager(this, 2)
```



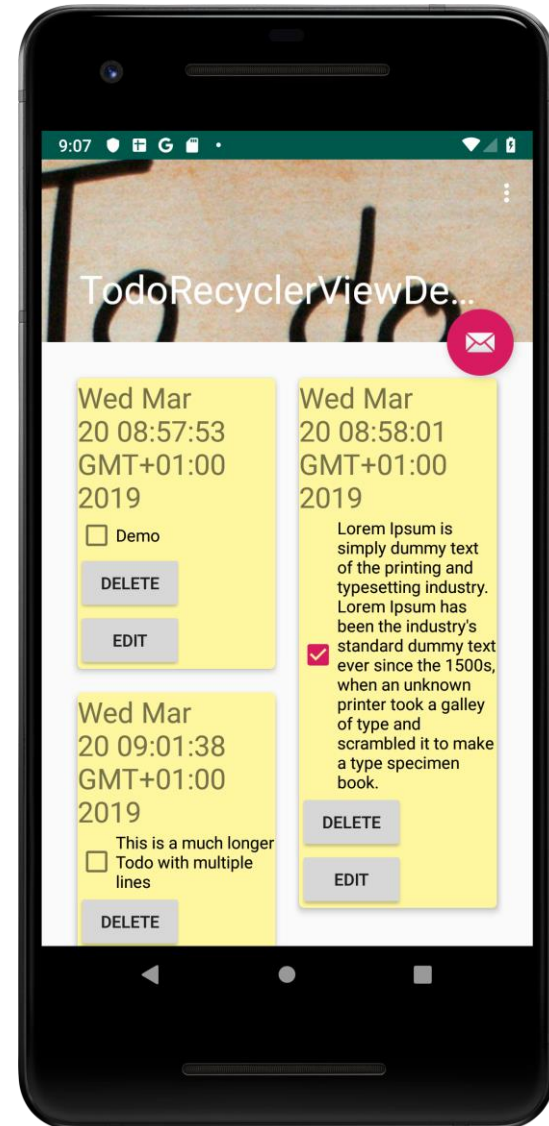
StaggeredGridLayoutManager demo

```
/*  
public StaggeredGridLayoutManager (int  
spanCount, int orientation)  
Creates a StaggeredGridLayoutManager with given  
parameters.
```

Parameters

spanCount : If orientation is vertical,
spanCount is number of columns. If orientation
is horizontal, spanCount is number of rows.
orientation : VERTICAL or HORIZONTAL
*/

```
// Define a layout for RecyclerView  
recyclerTodo.layoutManager =  
StaggeredGridLayoutManager (  
    2, StaggeredGridLayoutManager.VERTICAL)
```



KenburnsView

- Animált *ImageView*



Swipe és drag&drop gesztusok

- Távolítsuk el az elemeket swipe hatására
- Tegyük lehetővé az elemek átrendezését drag&drop-pal
- *RecyclerView* támogatás:
 - > `ItemTouchHelper.Callback`

ItemTouchHelper.Callback 1/2

- *isLongPressDragEnabled()*:
 - > True visszatérés ha a drag&drop támogatott
- *isItemViewSwipeEnabled()*:
 - > True visszatérésé ha a swipe támogatott
- *onMove(...)*:
 - > Elem mozgatás esetén hívódik meg
- *onSwipe(...)*:
 - > Swipe esetén hívódik meg

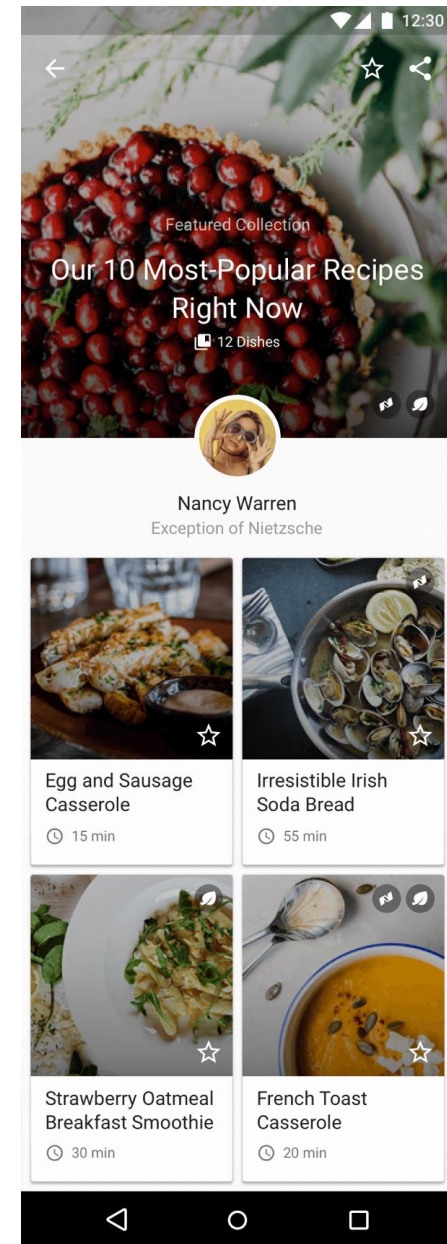
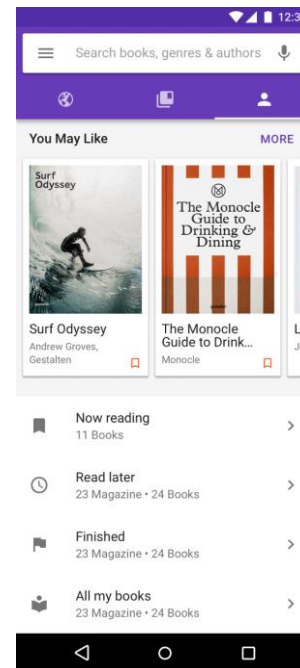
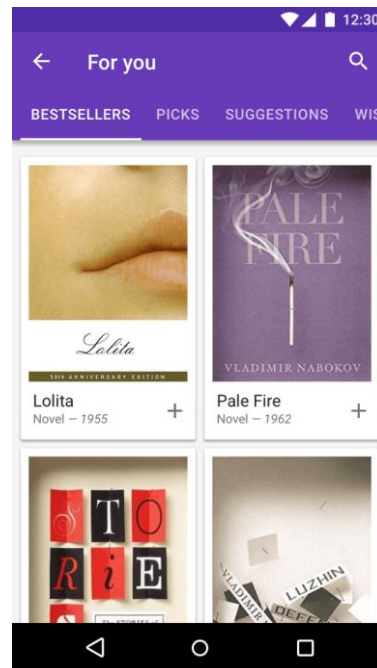
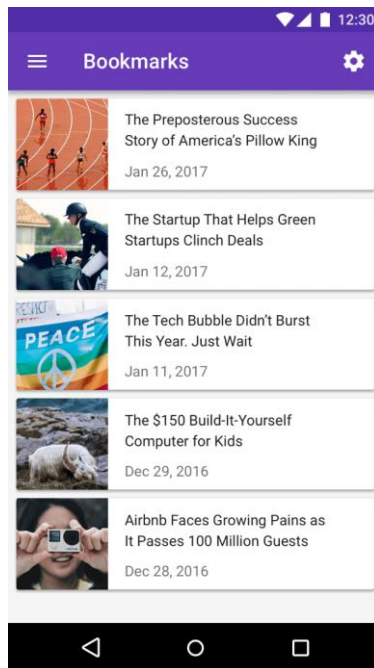
ItemTouchHelper.Callback 2/2

- Drag és swipe irányok beállítása:

```
override fun getMovementFlags(recyclerView: RecyclerView,  
    viewHolder: RecyclerView.ViewHolder): Int {  
    val dragFlags = ItemTouchHelper.UP or ItemTouchHelper.DOWN  
    val swipeFlags = ItemTouchHelper.START or ItemTouchHelper.END  
    return ItemTouchHelper.Callback.makeMovementFlags(dragFlags, swipeFlags)  
}
```


Design ötletek

- <https://materialdesignkit.com/templates/>
- <https://www.materialpalette.com/colors>
- <https://material.io/resources/color/#!/?view.left=0&view.right=0>
- <https://material.io/design/color/#color-usage-palettes>



Összefoglalás

- UI építő elemek
 - > Layout (ViewGroup)
 - > View-k
- RecyclerView
- Összefoglalás

Kérdések

