UNIVERSITAS SCIENTIARUM SZEGEDIENSIS
UNIVERSITY OF SZEGED
Department of Software Engineering

Mobil alkalmazásfejlesztés - Architektúra

Dr. Bilicki Vilmos

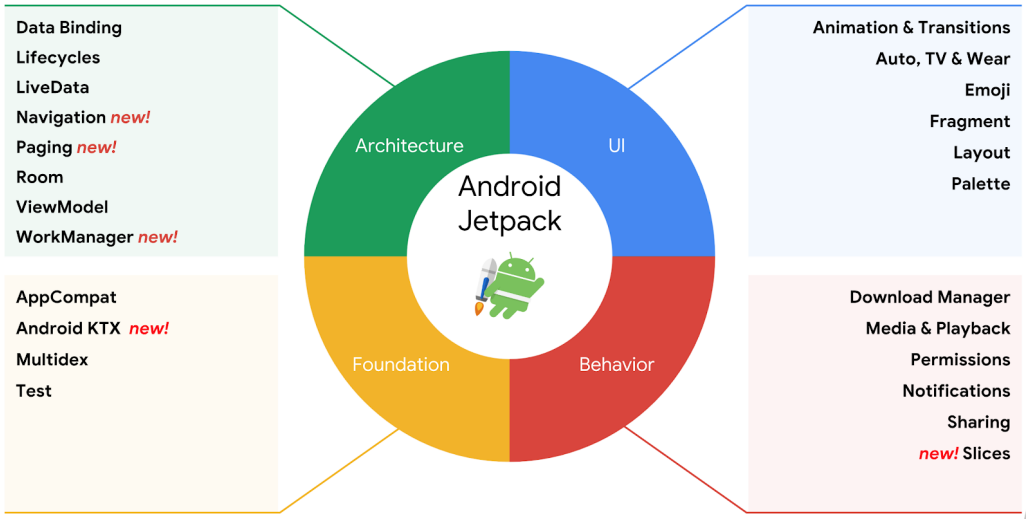
Szoftverfejlesztés Tanszék

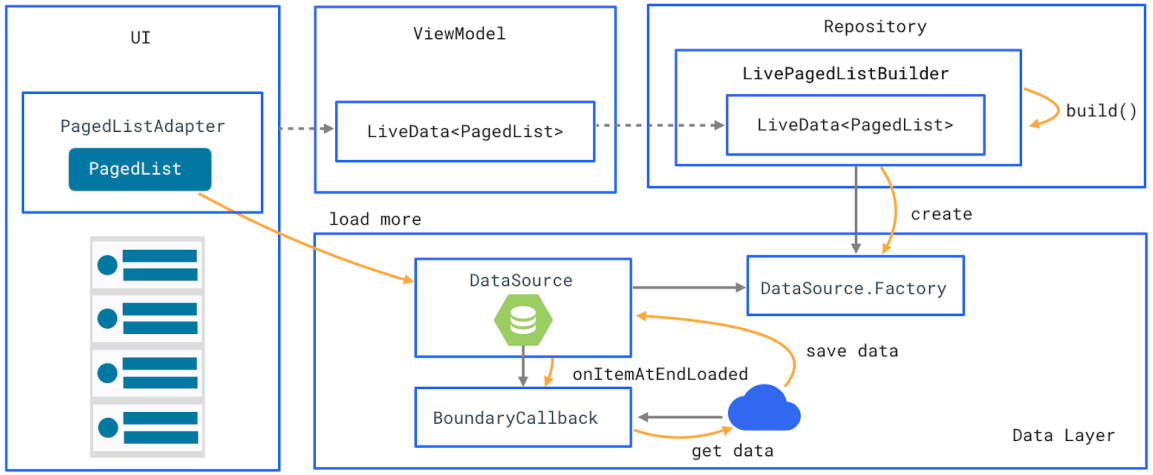
A fóliákhoz felhasznált anyagok:
Google: Android Developer Fundamentals (Version 2), https://developer.android.com/courses/fundamentals-training/overview-v2

Összefoglaló

- Alkalmazás architektúra
  - ViewModel
  - Repository
  - LiveData
- Android összefoglaló

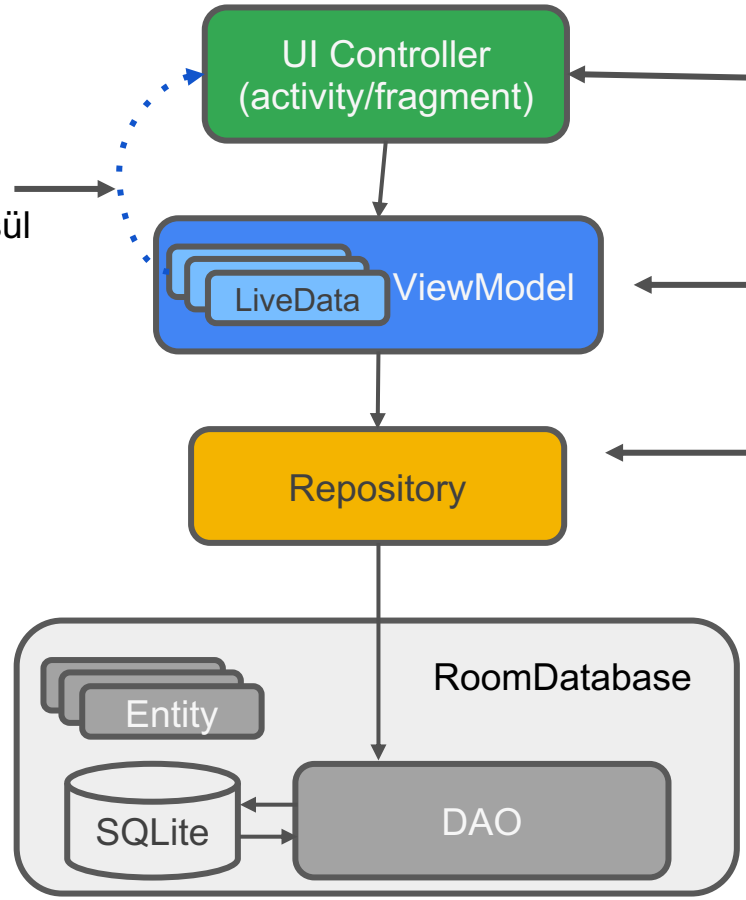
Android components (2017)





Áttekintés

UNIVERSITAS SCIENTIARUM SZEGEDIENSIS
UNIVERSITY OF SZEGED
Department of Software Engineering



Az adatot árbázolja és továbbítja az eseményeket

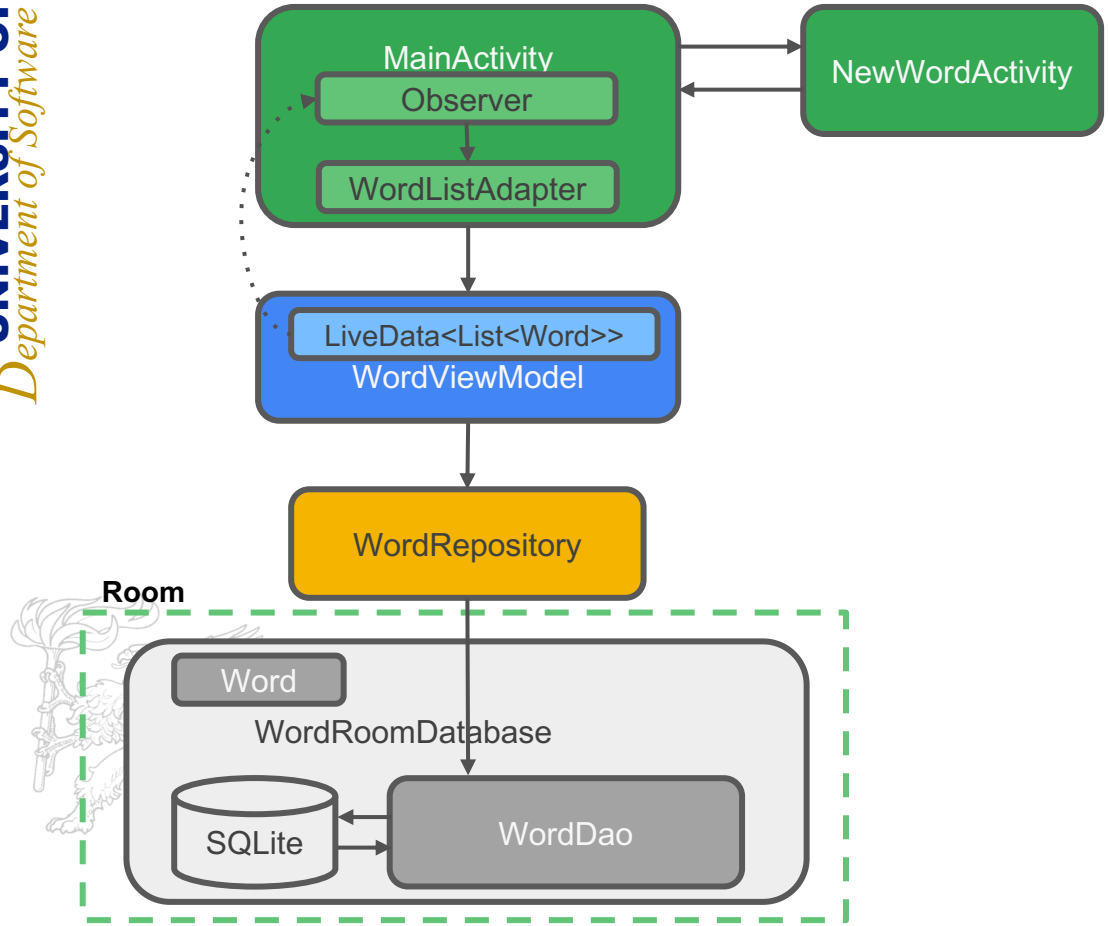
Az observation segítségével értesül a változásokról

Az UI-hoz szükéshes adtot tartalmazza

Az alkamlazáshoz szükséges adot egyetlen forrása “Single source of truth”; Letisztult API az UI-hoz

A helyi adatkezlés valósítja meg SQLite adatforrással

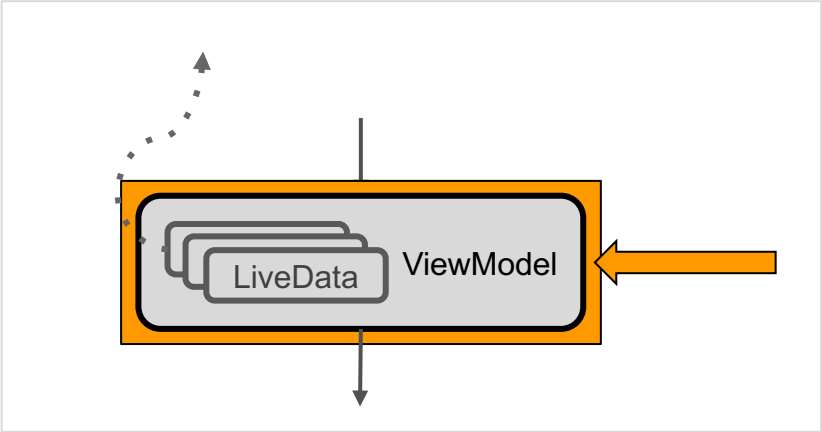
Megvalósítása



UNIVERSITAS SCIENTIARUM SZEGEDIENSIS
UNIVERSITY OF SZEGED
Department of Software Engineering

ViewModel

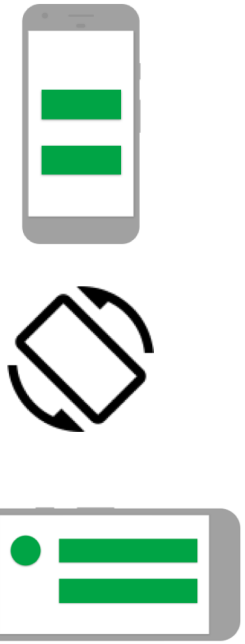
- A View model-ek
  - Adatot biztosítnak az UI-nak
  - Túlélik a konfiguráció változásokat

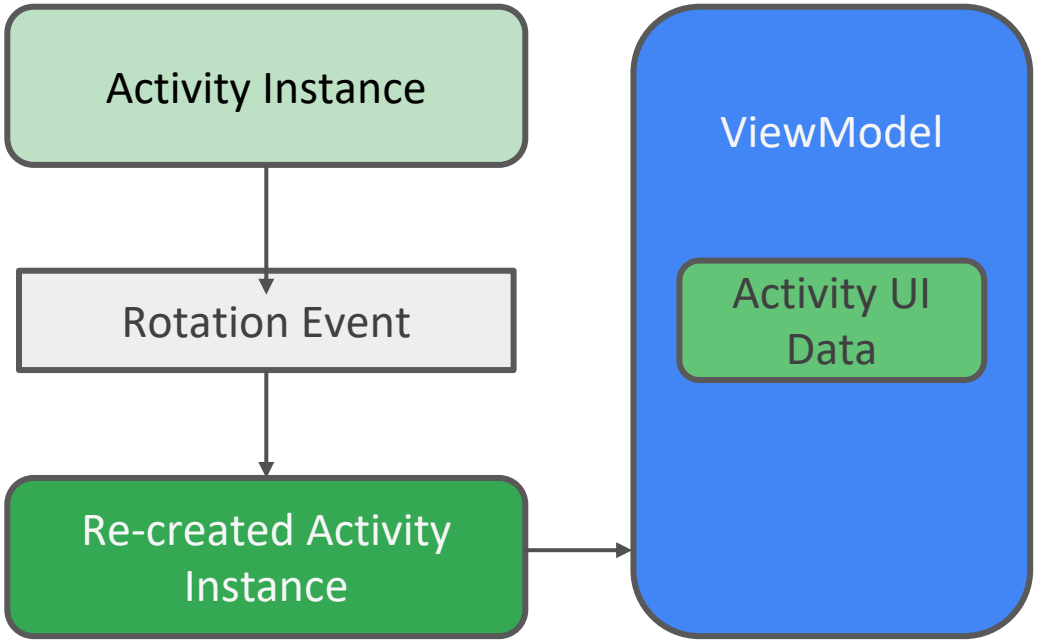


ViewModel

- Testreszabott adat az UI-nak
- Tartósabb mint az Activity
- ViewModel segítségével UI alatti elemek között is megosztahtuk adatot (Fragment)
- A lifecycle library eleme

Túléli a konfiguráció változásokat





UNIVERSITAS SCIENTIARUM SZEGEDIENSIS
UNIVERSITY OF SZEGED
Department of Software Engineering

A ViewModel adatokkal szolgál ki

- Az UI-t adatokkal szolgálja ki
- Az adat Room adatbáziból, vagy más forrából is jöhet
- A ViewModel szerepe az adatok visszaadása (segít mekeresni vagy generálni)

UNIVERSITAS SCIENTIARUM SZEGEDIENSIS
UNIVERSITY OF SZEGED
Department of Software Engineering

A legjobb gyakoralt: repository alkalmazása

- Javasolt megoldás:
  - A repository-t használjuk az adatok beszerzésére
  - Tartsuk a ViewModel meg egy letisztult interfészként

UNIVERSITAS SCIENTIARUM SZEGEDIENSIS
UNIVERSITY OF SZEGED
Department of Software Engineering

Étterem analógia

UNIVERSITAS SCIENTIARUM SZEGEDIENSIS
UNIVERSITY OF SZEGED
Department of Software Engineering

- A vásároló egy ételt rendel a pincértől
- A pincér elviszi ezt a szakácsnak
- A szakács elkészítit az ételt
- A pincér kiszolgálja az ételt

- Az UI adatot kér a ViewModel-től
- A ViewModel adatot kér a Repository-tól
- A Repository megkapja az adatot
- A ViewModel átadja az adatot az UI-nak

ViewModel és repository minta

```
public class WordViewModel extends AndroidViewModel {

    private WordRepository mRepository;
    private LiveData<List<Word>> mAllWords;

    // Initialize the repository and the list of words
    public WordViewModel (Application application) {
        super(application);
        mRepository = new WordRepository(application);
        mAllWords = mRepository.getAllWords();
    }
```

UNIVERSITAS SCIENTIARUM SZEGEDIENSIS
UNIVERSITY OF SZEGED
Department of Software Engineering

ViewModel folytatása

```
LiveData<List<Word>> getAllWords() {
    return mAllWords;
}
public void insert(Word word) {
    mRepository.insert(word);
}
public void deleteWord(Word word) {
    mRepository.deleteWord(word);
}
```

Ne adjukn át kontextust a ViewModel-nek

- A ViewModel-nek ne adjunk át kontextus változókat (memória szivárgás)
  - Ne tároljunk Activity, Fragment, vagy View példányokat vagy a kontextusukat ViewModel-ben
- Egy Activity a ViewModel élettartama alatt többször is megsemmisülhet, keletkezhet
- Amennyiben szükségünk van alkalmazás kontextusra akkor származtassunk a AndroidViewModel-ből

UNIVERSITAS SCIENTIARUM SZEGEDIENSIS
UNIVERSITY OF SZEGED
Department of Software Engineering

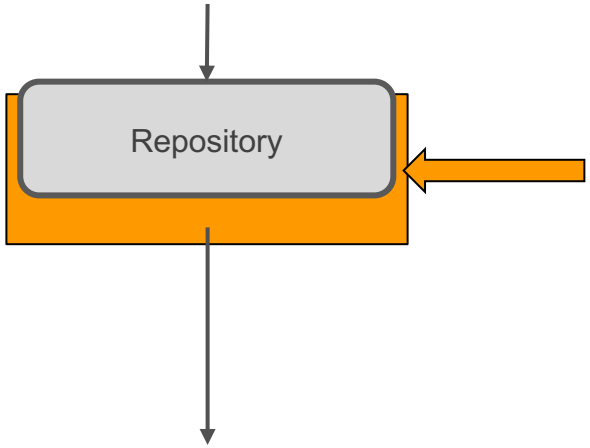
A ViewModel Nem léi túl az alklamazás bezárását

- A ViewModel túléli a konfiguráció változásokat de az alkalmazás leállását nem
- A ViewModel nem helyettesíti a onSaveInstanceState() (amennyiben nem Room segítségével mentjük el)

UNIVERSITAS SCIENTIARUM SZEGEDIENSIS
UNIVERSITY OF SZEGED
Department of Software Engineering

Repository

- Legjobb gyakorlat, nem része az Architecture Components könyvtárnak
- Célszerű megvalósítani, hogy egy letisztult API legyen az alkalmazás adatoknak



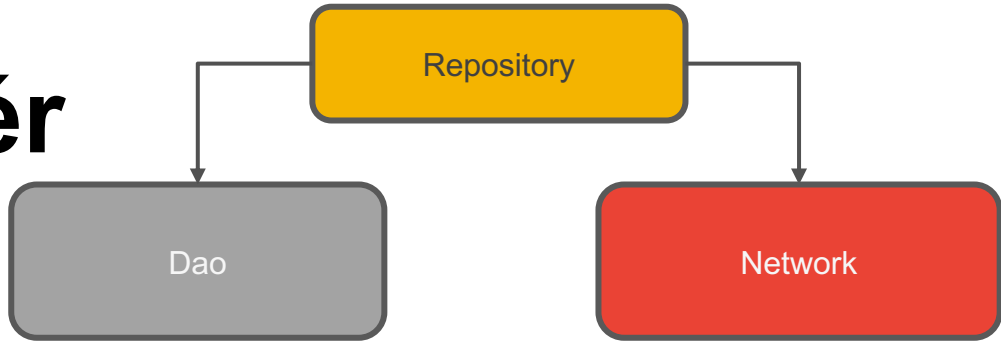
UNIVERSITAS SCIENTIARUM SZEGEDIENSIS
UNIVERSITY OF SZEGED
Department of Software Engineering

A Repository adatotkat tölt le vagy hoz létre

- Az adtok háttérben történő letöltését itt célszerű megvalósítani
- Analógia: a szakács a konyhában készíti az ételt

UNIVERSITAS SCIENTIARUM SZEGEDIENSIS
UNIVERSITY OF SZEGED
Department of Software Engineering

Többfajta háttér



- A repository lekérdezés szálakat tud kezelni és több háttérendszerhez is csatlakozhat
- Példa: itt valósíthatjuk meg azt a logikát amely dönt arról, hogy töltsünk le új adatokat a hálózatról, vagy használjuk a meglévőket

UNIVERSITAS SCIENTIARUM SZEGEDIENSIS
UNIVERSITY OF SZEGED
Department of Software Engineering

Repository példa

```
public class WordRepository {
  private WordDao mWordDao;
  private LiveData<List<Word>> mAllWords;
  WordRepository(Application application) {
    WordRoomDatabase db = WordRoomDatabase.getDatabase(application);
    mWordDao = db.wordDao();
    mAllWords = mWordDao.getAllWords();
  }
  [... more code…]
}
```

Get és insert data

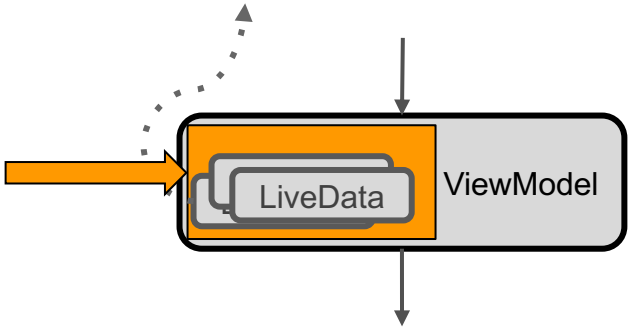
```
    LiveData<List<Word>> getAllWords() {
        return mAllWords;
    }

    // Must insert data off the main thread
    public void insert (Word word) {
        new insertAsyncTask(mWordDao).execute(word);
    }

   [... more code…]
}
```

Insert a fő szál kíméléssével

```
private static class insertAsyncTask extends AsyncTask
   <Word, Void, Void> {

        private WordDao mAsyncTaskDao;

        insertAsyncTask(WordDao dao) {
            mAsyncTaskDao = dao;
        }

        @Override
        protected Void doInBackground(final Word... params) {
            mAsyncTaskDao.insert(params[0]);
            return null;
        }
   }
```

LiveData

- A LiveData egy életciklust figyelmebe vévő adattároló osztály.
- Megfigyelhető értékekekt tárol.
- Segítségével az UI naprakészen tartaható.



UNIVERSITAS SCIENTIARUM SZEGEDIENSIS
UNIVERSITY OF SZEGED
Department of Software Engineering

LiveData

- A LiveData megfigyelhető adat (observable)
- Változás esetén értesíti a megfigyelőt
- Figyelmebe veszi az életciklust:
  - tudja, hogy ez eszköz mikor forog, vagy az alkalmazás mikor áll le

Az UI naprakészen tartásához használjuk a LiveData-t

- Hozzunk létre egy a LiveData-t megfigyelő megfigyelőt
- A LiveData értesíti a megfigyelő objektumokat változás esetén
- Ennek hatására frissíthetjük az UI-t

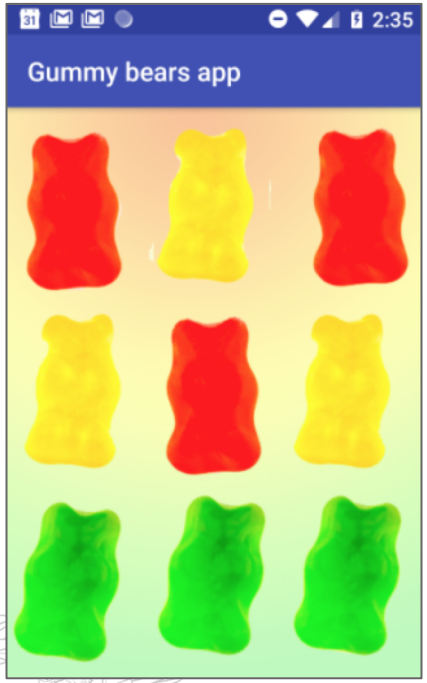
UNIVERSITAS SCIENTIARUM SZEGEDIENSIS
UNIVERSITY OF SZEGED
Department of Software Engineering

LiveData létrehozása

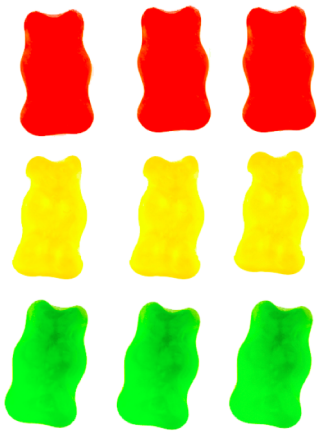
LiveData-ként adjuk vissza, hogy megfigyelhető legyen

```
@Query("SELECT * from word_table)
LiveData<List<Word>> getAllWords();
```

LiveData és Room használata



A Room a `LiveData` frissítéséhez szükséges minden kódot legenrál ez fut le a mikor az adatbázis frissül



LiveData továbbadása rétegeken át

- A különböző rétegeben minden esetben LiveData típust kell alkalmaznunk:
  - DAO
  - ViewModel
  - Repository

UNIVERSITAS SCIENTIARUM SZEGEDIENSIS
UNIVERSITY OF SZEGED
Department of Software Engineering

LiveData rétegeken át

- DAO:

```
@Query("SELECT * from word_table")
LiveData<List<Word>> getAllWords();
```

- Repository:

```
LiveData<List<Word>> mAllWords =
                    mWordDao.getAllWords();
```

- ViewModel:

```
LiveData<List<Word>> mAllWords =
                    mRepository.getAllWords();
```

LiveData megfigyelése

- Hozzunk létre a megfigyelőt az Acitvity onCreate() metódusában
- Írjuk felül a megfigyelő onChanged() metódusát

Amikor a LiveData változik akkor, a megfigyle értesül és az onChanged()-ben megadott kód lefut

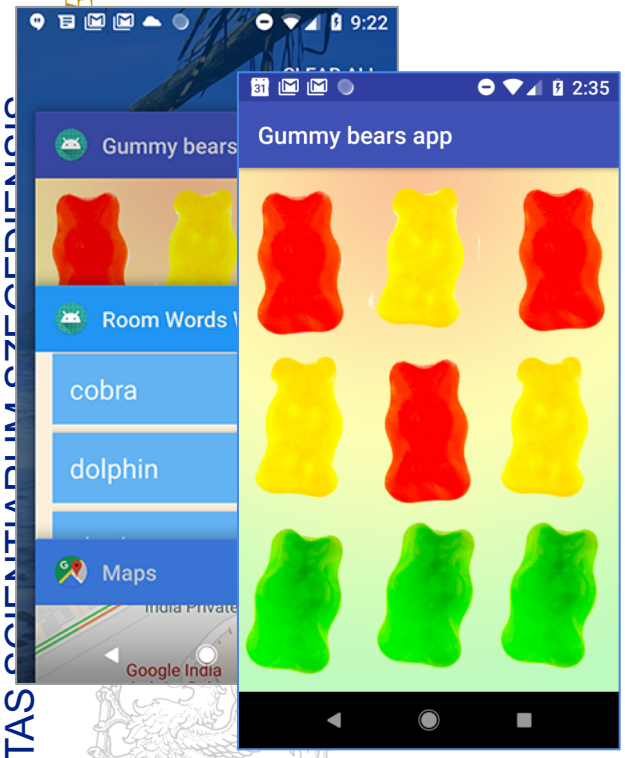
LiveData megfigyelése: példa

```
final Observer<String> nameObserver =
  new Observer<String>() {
    @Override
    public void onChanged(@Nullable final String newName) {
        // Update the UI, in this case, a TextView.
        mNameTextView.setText(newName);
    }
};

mModel.getCurrentName().observe(this, nameObserver);
```

Nincs memória szivárgás

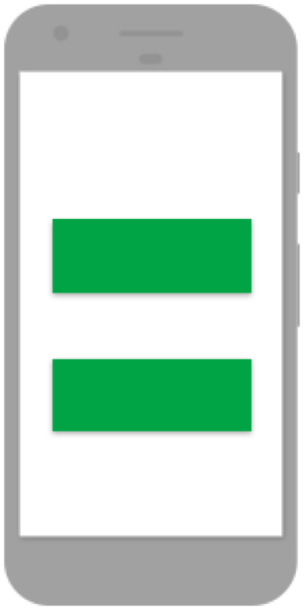
- A megfigyelők az életciklus objektumokhoz kötöttek, ezek pedig olyan objektumok amelyeknek van Android életciklusuk
- A megfigyelők feltakarítanak maguk után miután az életciklusuk megsemmisül

A LiveData naprakész



- Amennyiben egy életciklussal bíró objketum inaktív állapotból aktívvá válik megkapja a legújabb adatot
- Példa: egy activity-t előtérbe hozunk a háttérből, ez esetben megkapja a legújabb adatot

A LiveData kezeli a konfiguráció változásokat



Amennyiben egy activity-t fragmentet újra létrehoznak a konfiguráció változás miatt akkor megkapja a legújabb adatokat

UNIVERSITAS SCIENTIARUM SZEGEDIENSIS
UNIVERSITY OF SZEGED
Department of Software Engineering

Erőforrások megosztása

- Kiegészíthetjük a LiveData objektumot a singleton mintával szolgáltatás vagy adatbázis megosztására
- A LiveData objektum kapcsolódik adott erőforráshoz és bárkinek szüksége van erre az erőforrásra annak csak figyelnie kell a LiveData objektumomt

Életciklus ismerő komponensek

- Ahelyett, hogy az életciklushoz kötött alklamazásokak a megfelelő életciklus onStart(), onStop(), metódusban kezelnénk
  - eseményekre is felíratkozhatunk és ezekre reagálhatunk

UNIVERSITAS SCIENTIARUM SZEGEDIENSIS
UNIVERSITY OF SZEGED
Department of Software Engineering

Életciklus ismerő komponensek

- Az ilyen objektumok más komponens életciklus váltására tudnak reagálni
- Például: az Activity elindulása vagy léállása hátására egy listener is ezt az életciklust követheti

UNIVERSITAS SCIENTIARUM SZEGEDIENSIS
UNIVERSITY OF SZEGED
Department of Software Engineering

@OnLifecycleEvent

`@OnLifecycleEvent` indicates life cycle methods

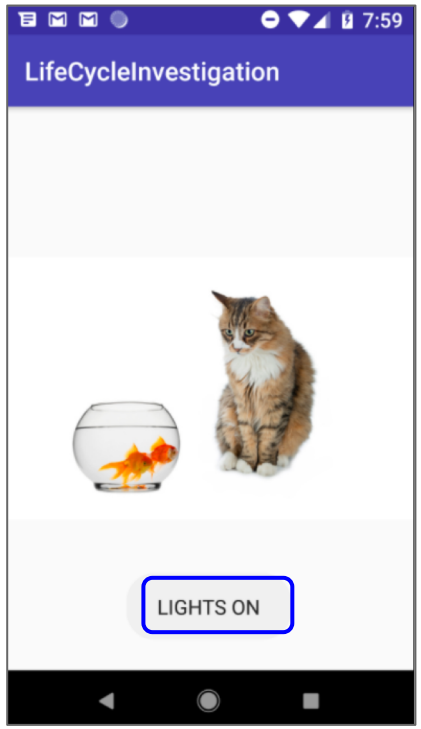
```
@OnLifecycleEvent(Lifecycle.Event.ON_START)
public void start() {...}

@OnLifecycleEvent(Lifecycle.Event.ON_STOP)
public void start() {...}
```

See Lifecycle.event reference for more lifecycle events

A POJO-k is lehetenek életciklust ismerők

Bármely osztály reagálhat életciklus eseményre

Az alkalmazásban, a `Toast`-ot egy egyszerű POJO hozza létre



UNIVERSITAS SCIENTIARUM SZEGEDIENSIS
UNIVERSITY OF SZEGED
Department of Software Engineering

POJO kiegésíztése életciklus követéssel

```
public class Aquarium {

  // Constructor takes Application and lifecycle
  public Aquarium(final Application app,
                  Lifecycle lifecycle) {
    ...
  }
```

UNIVERSITAS SCIENTIARUM SZEGEDIENSIS
UNIVERSITY OF SZEGED
Department of Software Engineering

A konstruktor

```
public Aquarium(final Application app, Lifecycle lifecycle) {
  // Add a new observer to the lifecycle.
  lifecycle.addObserver(new LifecycleObserver() {
      @OnLifecycleEvent(Lifecycle.Event.ON_START)
      public void start() {
        Toast.makeText(app, "LIGHTS ON", Toast.LENGTH_SHORT).show();
      }
  });
}
```

Példányosítás

```
public class MainActivity extends AppCompatActivity {
   private Aquarium myAquarium;
   protected void onCreate(...) {
       ...
       // Create aquarium.
       // Pass context and this activity's lifecycle
       myAquarium = new Aquarium(this.getApplication(),
                                 getLifecycle());
   }
}
```

Android eszencia

- UI elrendezések és erőforrások
  - Nézet, Nézet csoport, nézet hierarchia
  - Elrendezés szerkesztő, ConstrainLayout
  - Eseménykezelés
  - Erőforrások és mérésük
- Szöveg és gördülő nézetek
- Activity-k és intent-ek

Amit látunk az nézet (view)

- Minden interfész elem egy-egy nézet

ViewGroup gyermek nézeteket tartalmaz

- ConstraintLayout: Az UI elemeket adott egymáshoz és az elrendezés széleihez viszonyított kényszerek mentén helyezi el
- ScrollView: egy elemet tartalmaz és megengedi a görgetést
- RecyclerView: egy elem listát tartalmaz és a listát tudja bővíteni csökkenteni és görgetni

UNIVERSITAS SCIENTIARUM SZEGEDIENSIS
UNIVERSITY OF SZEGED
Department of Software Engineering

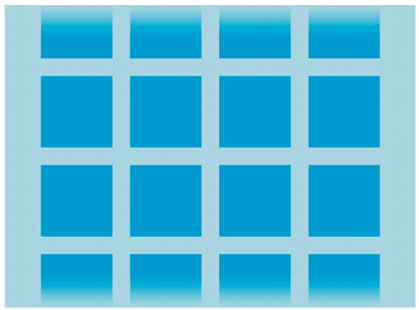
ViewGroup az elrendezésekhez

- Layot – elrendezés
  - A ViewGroup alosztályai
  - Gyermek nézeteket tartalmaznak
  - Lehetnek: sor, oszlop, rács, tábla és abszolút elrendezésűek

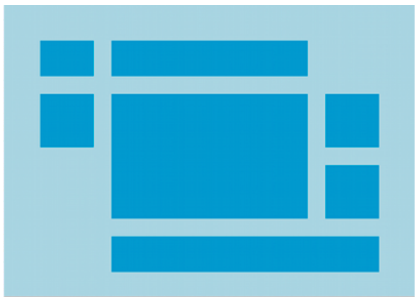
Gyakori elrendezés osztályok

LinearLayout

ConstraintLayout



GridLayout

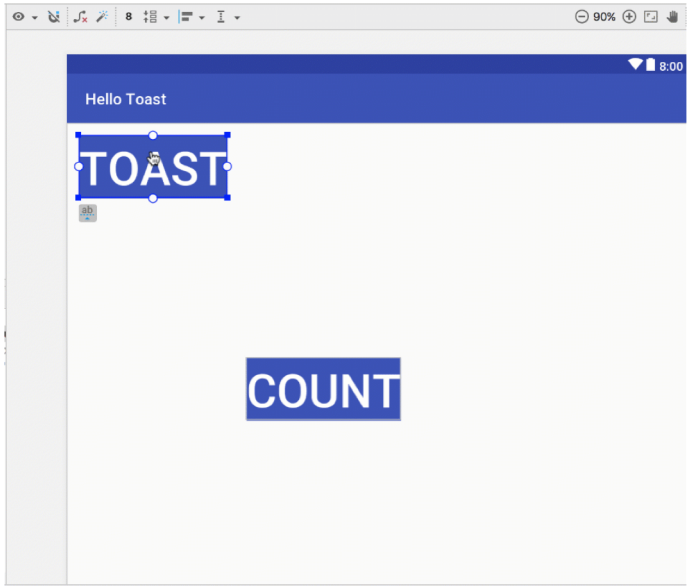


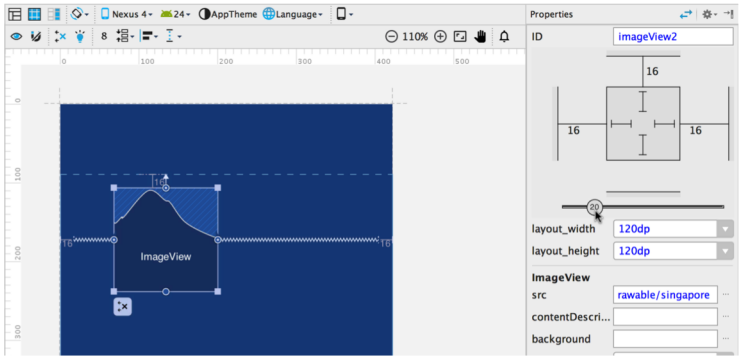
TableLayout

UNIVERSITAS SCIENTIARUM SZEGEDIENSIS
UNIVERSITY OF SZEGED
Department of Software Engineering

Elrendezés szerkesztő ConstraintLayout

- Az UI elemek szülőhöz kötése
- Elemek pozícionálása és átméretezése
- Elemek egymáshoz képesti elrendezése
- Dimenziók és szélek beállítása
- Tulajdonságok beállítása





UNIVERSITAS SCIENTIARUM SZEGEDIENSIS
UNIVERSITY OF SZEGED
Department of Software Engineering

ConstraintLayout

- az Android Studio alapértelmezett elrendezése
- egy kellő flexibilitás nyújtó ViewGroup
- az elemek helyét és méretét kényszerek segítségével határozza meg
- a kényszer egy kapcsolat egy másik nézet, a szülő elrendezés vagy vezető elemhez

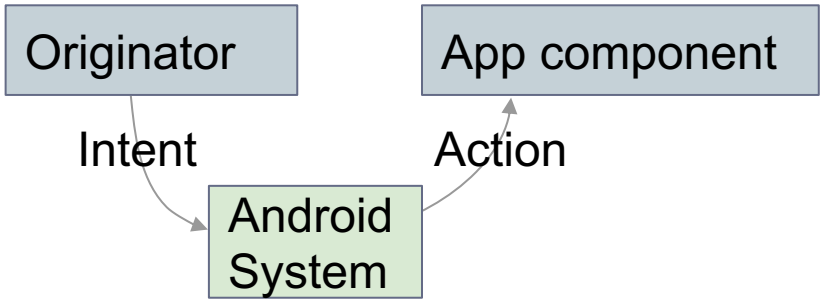
Android	iOS
top	top
bottom	bottom
left	left
right	right
start	leading
end	trailing
	centerX
	centerY
baseline	baseline

Aktivitások

- Az Activity egy alkalmazás komponens
  - Eseményeket kezel
  - Egy ablakot, egy nézet hierarchiát ábrázol
  - Indíthat más aktivitásokat
  - Életciklusa van
  - Általában van elrendezése (Layout)
- Általában kitölti a képernyőt, de lehet beágyazva, vagy lebeghet is
- Java osztály – tipikusan egy Activity egy fájl
- Alkalmazás vs. Activity
  - Lazán csatolva alkotnak egy alkalmazást
  - „main activity”
  - Hierarchiába szervezhetőek (navigációt segítendő)

Intent

- Az Intent a végrehajtandó művelet leírása
- Az Intent segítségével kér az egyik alkalmazás komponens a másik alkalmazás komponenstől műveletet az Android futtatási környezet segítségével



Explicit és implicit intent

- Explicit intent
  - Egy meghatározott Activity-t indít el
    - Kávét kérünk Zsuzsától
    - A fő Activity a ViewShoppingCart Activity-t indítja el
- Implicit intent
  - Megkérjük a rendszert, hogy keressen egy megfelelő Activity-t amely a kését teljesítheti
    - Keressünk egy boltot ahol zöld teát vehetünk
    - A megosztás gombra kattintva a megosztást kezelő alkalmazások listáját látjuk

Activity indítása intent-tel

Explicit:

1. Intent-t létrehozása
   - `Intent intent = new Intent(this, ActivityName.class);`
2. Indítsuk el vele az Activity-t
   - `startActivity(intent);`

Implicit:

1. Intent-t létrehozása
   - `Intent intent = new Intent(action, uri);`
2. Indítsuk el vele az Activity-t
   - `startActivity(intent);`

UNIVERSITAS SCIENTIARUM SZEGEDIENSIS
UNIVERSITY OF SZEGED
Department of Software Engineering

Implicit Intent példák

Weboldal megnyitása

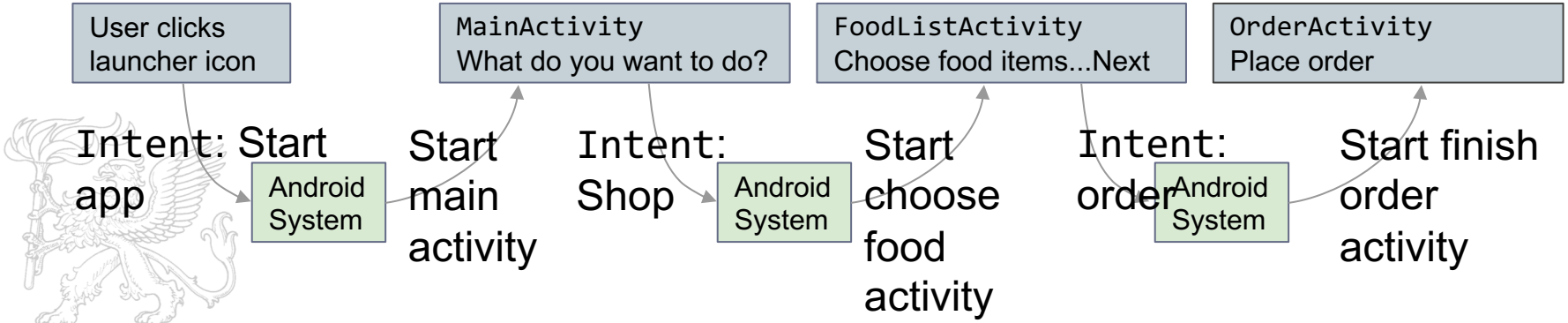
```
Uri uri = Uri.parse("http://www.google.com");
Intent it = new Intent(Intent.ACTION_VIEW,uri);
startActivity(it);
```

Telefonszám felhívása

```
Uri uri = Uri.parse("tel:8005551234");
Intent it = new Intent(Intent.ACTION_DIAL, uri);
startActivity(it);
```

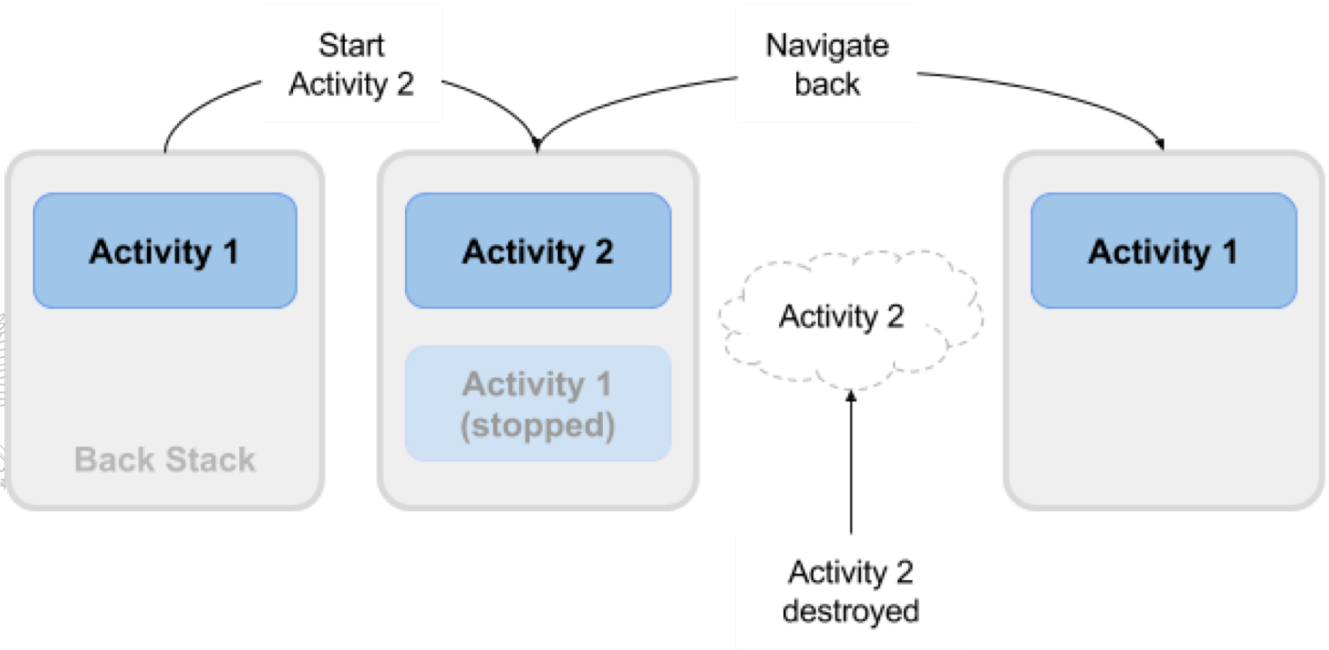
Hogyan futnak az Activity-k

- Az Android futási környezeti kezeli az életciklusukat
- Intent-tel indulnak (egy üzenettel)



Acitivity életciklusa

- Egy irányított gráf az Aktivitás minden állapotával és a megfelelő callback-ekkel

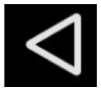


Láthatóság és Activity állapotok

- Created (nem látható még)
- Started (látható)
- Resume (látható)
- Paused(részben látható)
- Stopped (rejtett)
- Destroyed (el lett távolítva a memóriából)

- Állapot változás lehet a felhasználói események vagy konfiguráció változás hatására (meg lett fordítva a a telefon)

Navigáció



- Vissza (ideiglenes) navigáció
  - Az eszköz vissza gombja segítségével
  - Az Android vissza verem kezeli

- Ős (fel) navigáció
  - Az alkalmazás sáv fel gombja segítségével
  - Az AndroidManifest.xml-ben megadott gyermek szülő viszony alapján

RecyclerView

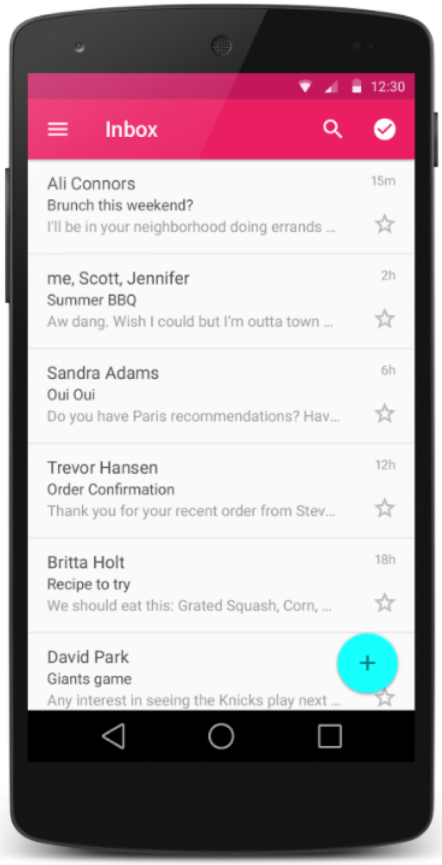


Figure 1. A list using `RecyclerView`

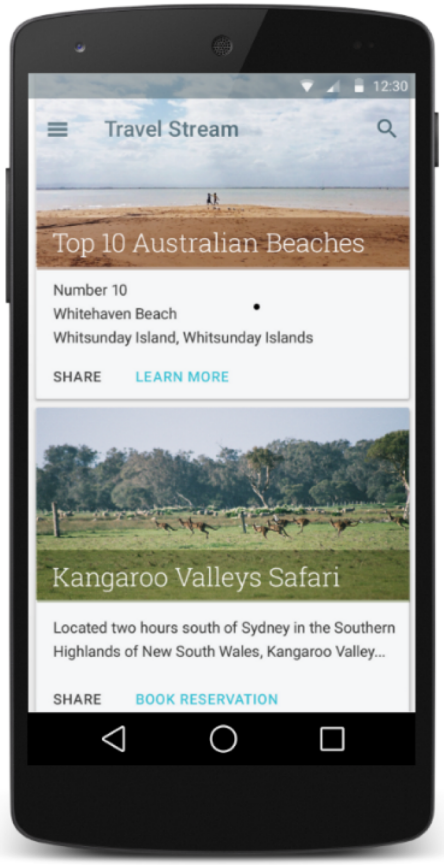
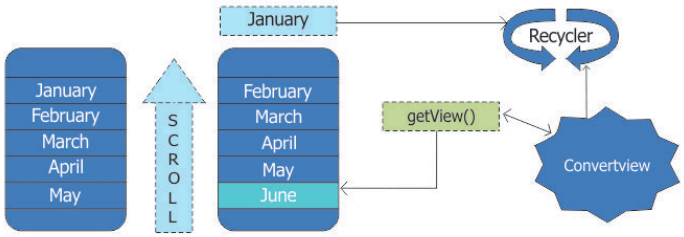
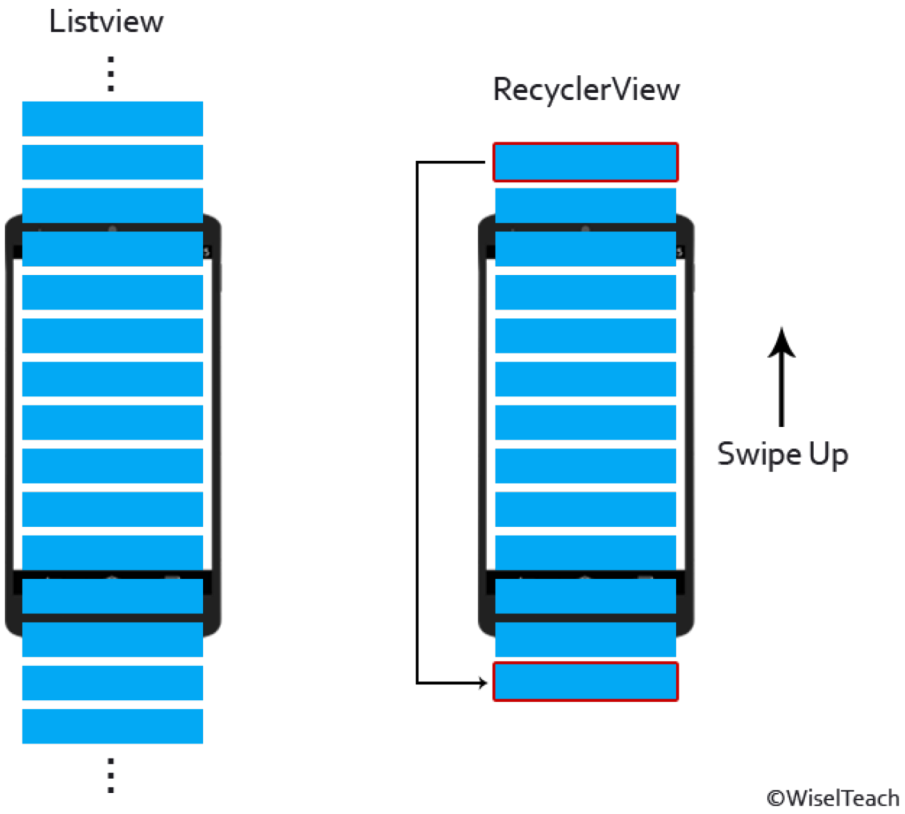


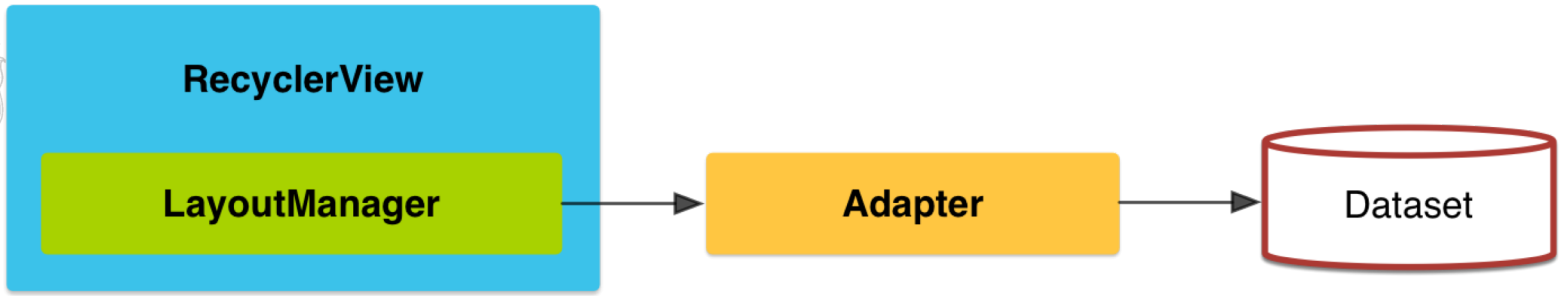
Figure 2. A list also using `CardView`

RecylerView

- Komponensek
- Megvalósítása

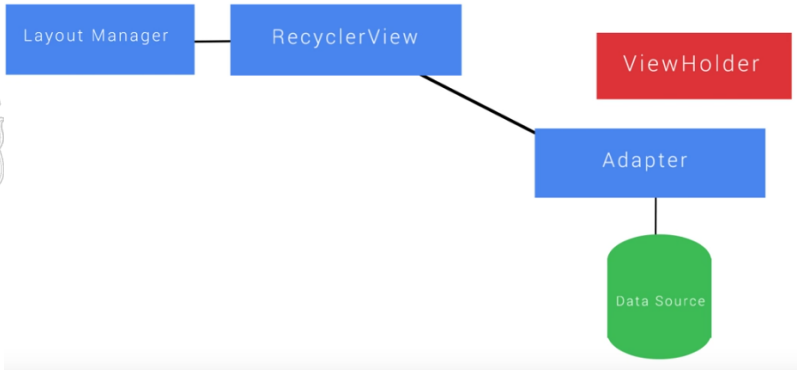




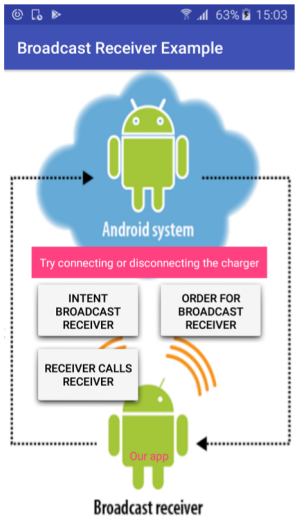


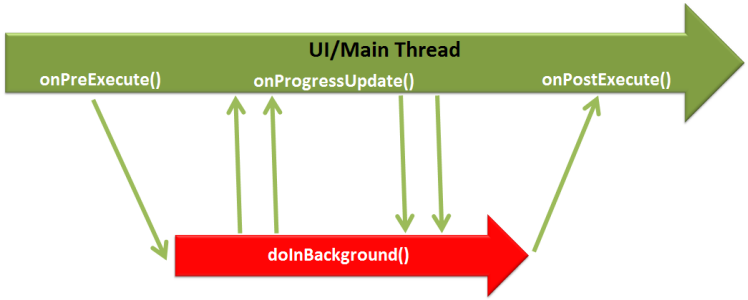
RecyclerView

- A `RecyclerView` egy nagy adathalmaz kezelésére is alkalmas görgethető tároló
- Hatékony mert
  - Újrahasznosít adott számú nézet elemet hasznosít
  - Az adat válotásit gyorsan követi

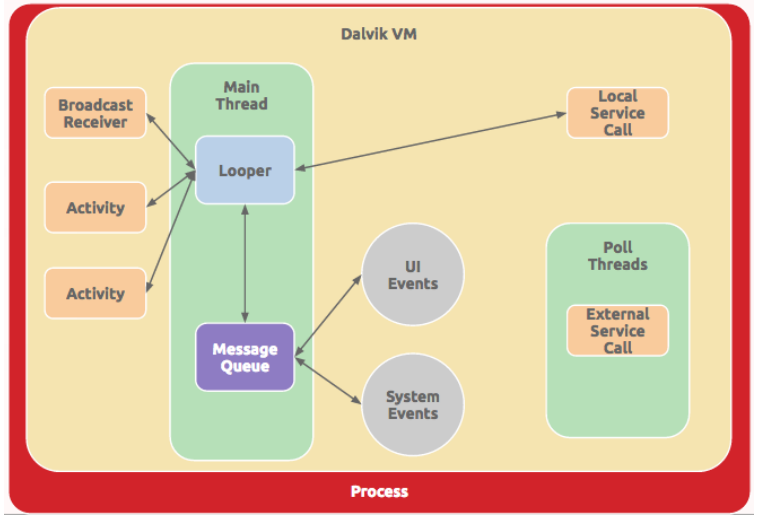


- Háttér folyamatok
  - AsyncTask és AsyncTaskLoader
  - Internet kapcsolat





AsyncTask in Android



Intent
Activity
Service
Broadcast Receiver
Content Provider

Have a condition?
Local work or sync?
JobDispatcher
Content provider?
SyncAdapter
Exact time?
AlarmManager (sparingly)
Online trigger?
Real-time messaging?
Google Cloud Messaging + JobDispatcher
Low Priority
High Priority
Something the user will understand?
Use a foreground service
You got to the bottom?
Don't do it in the background

Összefoglaló

- Alkalmazás architektúra
  - ViewModel
  - Repsitory
  - LiveData
- Android áttekintés