Android fejlesztés

RecyclerView lehetőségek, perzisztens adattárolás

peter.ekler@aut.bme.hu

BME AUT
Department of Automation and Applied Informatics

Miről volt szó az előző alkalommal? ☺

- Dinamikus felhasználói felület tervezés
- Stílusok és témák használata
- Animációk
- Listák kezelése – RecyclerView
- CardView – kompakt tartalom megjelenítés
- Dialógusok használata

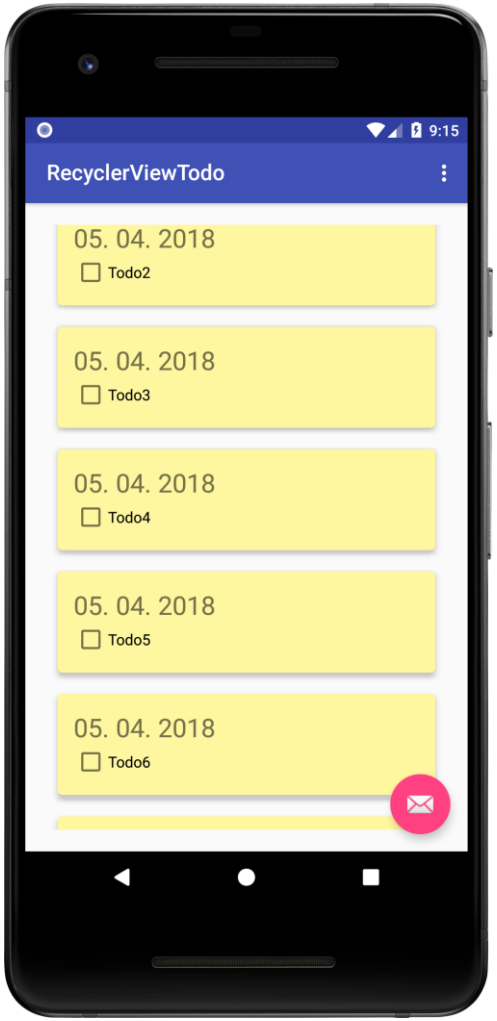
Tartalom

- RecyclerView további képességek
- Perzisztens adattárolás
  - >SQLite

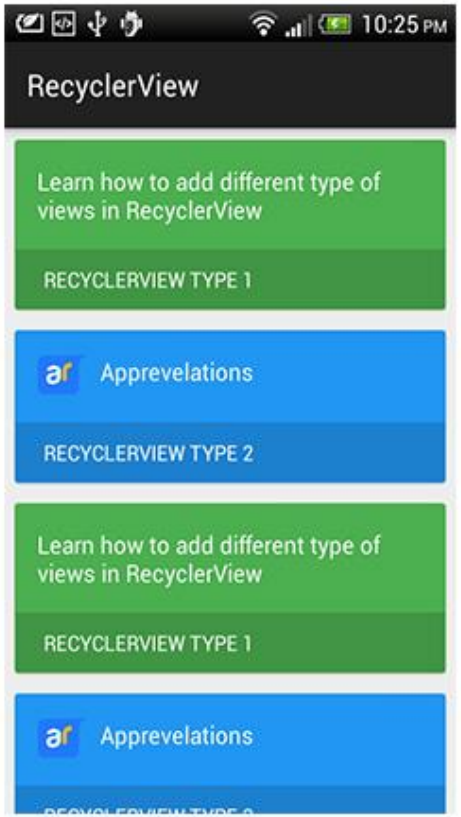
RECYCLERVIEW TOVÁBBI LEHETŐSÉGEK

Gyakoroljunk

- Készítsünk egy Todo alkalmazást RecyclerVeiw-val
- Használjunk CheckBox-ot
- Készítsünk „Undo” funkciót, mely elérhet elem létrehozása után néhány másodpercig

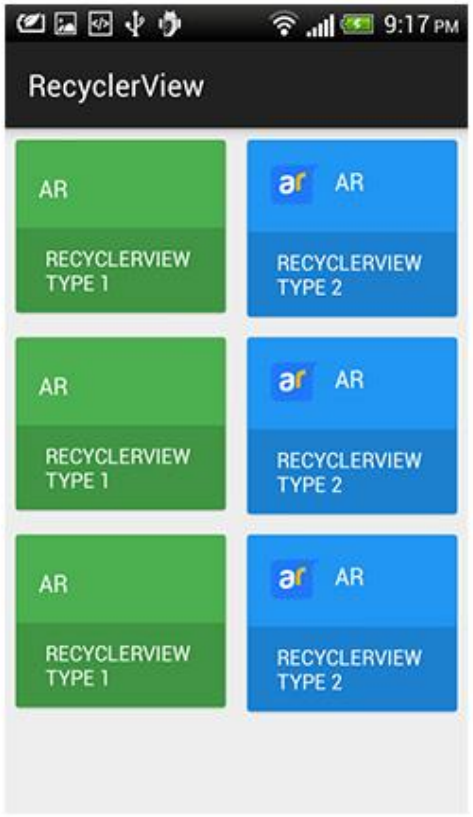


RecyclerView LayoutManager-ek



Linear Layout View

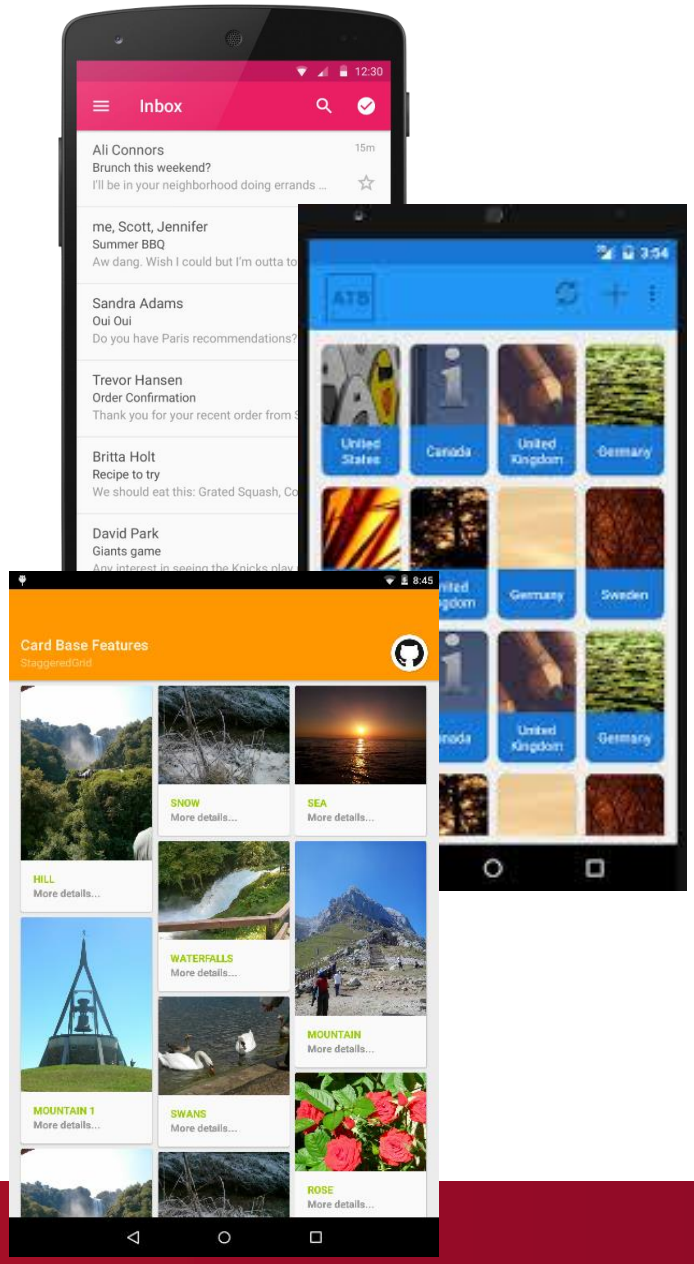
Staggered Grid View



Grid View

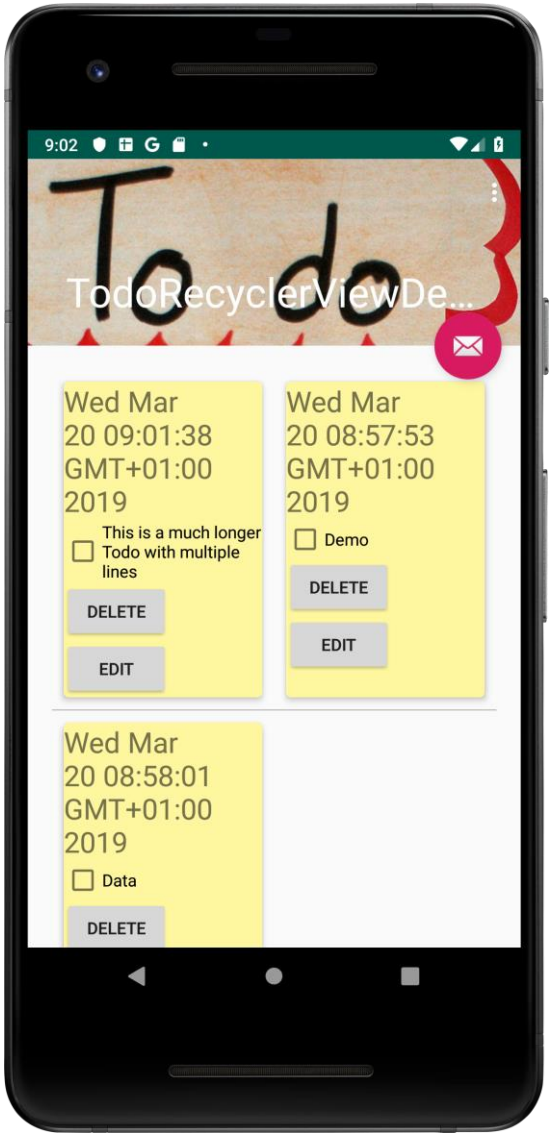
RecyclerView – LayoutManager típusok

- **LinearLayoutManager**
- **GridLayoutManager**
- **StaggeredGridLayoutManager**



GridLayoutManager demo

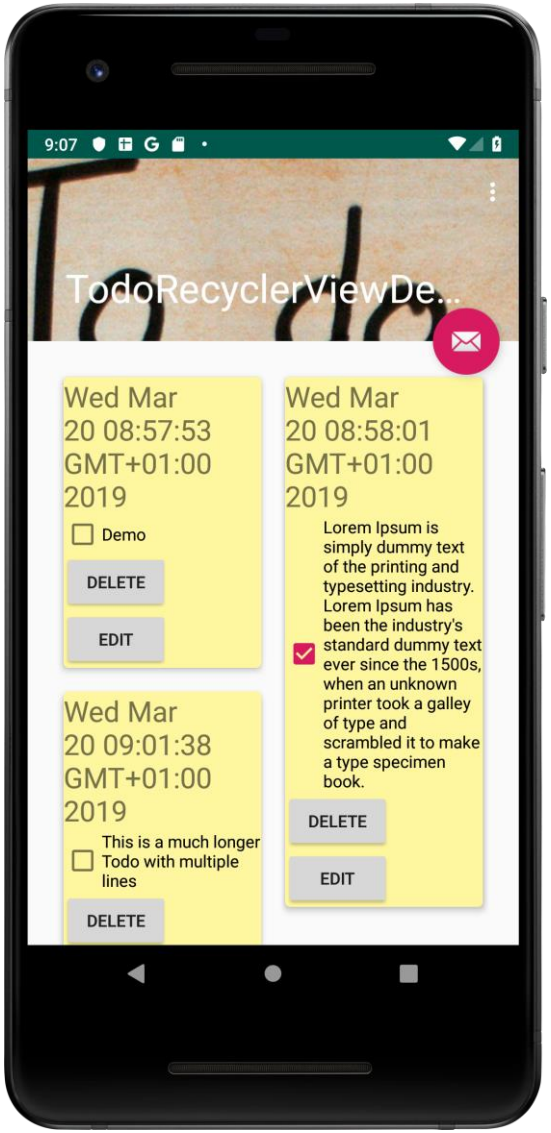
```
recyclerTodo.layoutManager =
  GridLayoutManager(this, 2)
```



StaggeredGridLayoutManager demo

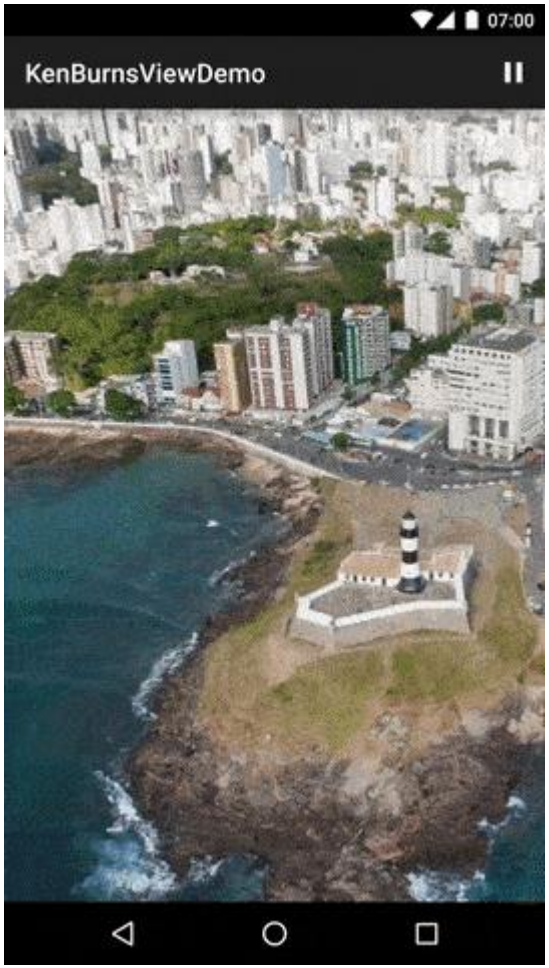
```
/*
public StaggeredGridLayoutManager (int
spanCount, int orientation)
Creates a StaggeredGridLayoutManager with given
parameters.

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

- Az *ImageView* kiterjesztése
- Képek automatikus animálása (pan&zoom)



Elválasztó vonalak használata

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

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*-ját

```
// 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");
    }
}
```

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

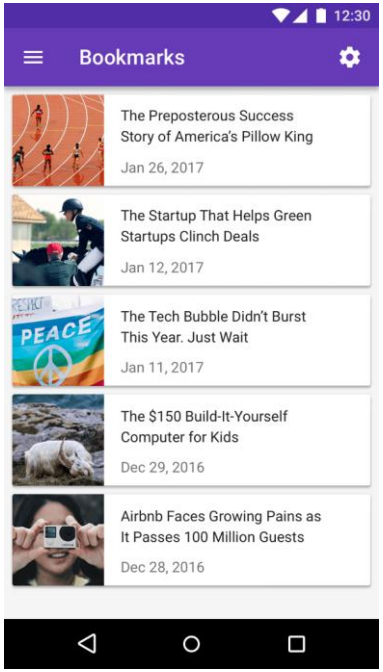
ItemTouchHelper.Callback 2/2

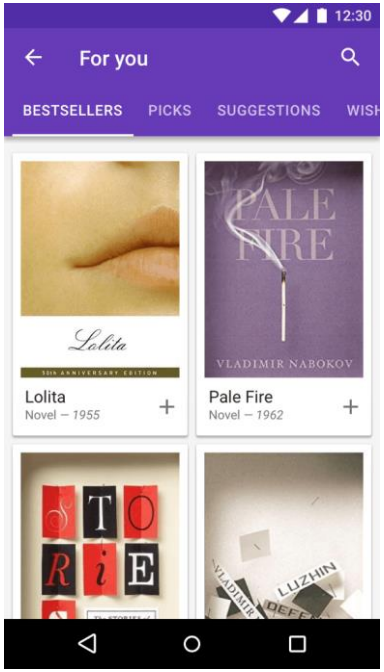
- Drag és swipe irányok beállítása:

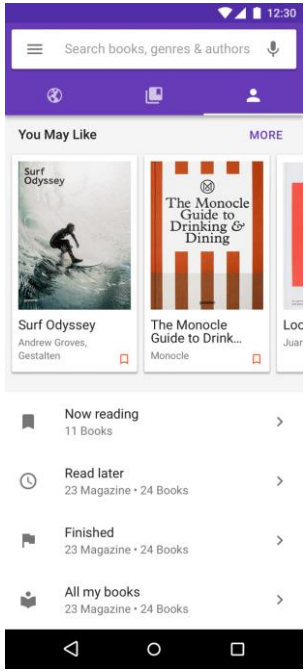
```
override fun getMovementFlags(recyclerView: RecyclerView,
  viewHolder: RecyclerView.ViewHolder): Int {
    val dragFlags = ItemTouchHelper.UP or ItemTouchHelper.DOWN
    val swipeFlags = ItemTouchHelper.START or ItemTouchHelper.END
    return ItemTouchHelper.Callback.makeMovementFlags(dragFlags,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

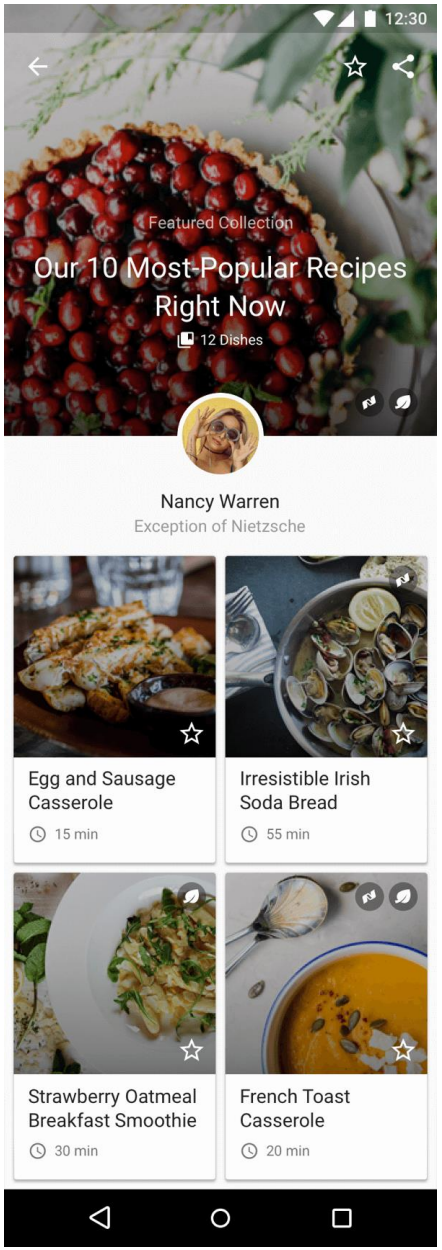
Design ötletek

- 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









Perzisztens adattárolás

Perzisztens adattárolás

- Gyakorlatilag minden Android alkalmazásnak kell perzisztensen tárolnia bizonyos adatokat
  - > Beállítások szinte mindig vannak
  - > Kamera alkalmazások: új fénykép fájl mentése
  - > Online erőforrásokat használó appok: lokális cache
  - > Email alkalmazások: levelek indexelt adatbázisa
  - > Bejelentkezést tartalmazó appok: be van-e jelentkezve a felhasználó
  - > Első indításkor tutorial megjelenítése: első vagy későbbi indítás?
  - > Picasa, Dropbox: elsődleges tárhely a felhőben

Adattárolási megoldások

- Androidon minden igényre van beépített megoldás:
  - **SQLite adatbázis**: strukturált adatok tárolására
  - ***SharedPreferences***: alaptípusok tárolása kulcs-érték párokban
  - **Privát lemezterület**: nem publikus adatok tárolása a fájlrendszerben
  - **SD kártya**: nagy méretű adatok tárolása, nyilvánosan hozzáférhető
  - **Hálózat**: saját webszerveren vagy felhőben tárolt adatok

SQLite

SQLite

- Az Android alapból tartalmaz egy teljes értékű relációs adatbáziskezelőt
  - SQLite – majdnem MySQL
- Strukturált adatok tárolására ez a legjobb választás
- Alapból nincs objektum-relációs réteg (ORM) fölötte, nekünk kell a sémát meghatározni és megírni a query-ket
- Külső ORM osztálykönyvtár:
  - http://ormlite.com/sqlite_java_android_orm.shtml
- Mivel SQL, érdemes minden táblában elsődleges kulcsot definiálni
  - autoincrement támogatás
  - Ahhoz, hogy *ContentProvider*-rel ki tudjuk ajánlani (később), illetve UI elemeket Adapterrel feltölteni (pl. list, grid), **kötelező egy ilyen oszlop**, melynek neve: „_id”

```
Class Person {
  String name;
  String address;
  Int age;

}
```

Android SQLite jellemzői 1/2

- Standard relációs adatbázis szolgáltatások:
  - SQL szintaxis
  - Tranzakciók
  - Prepared statement
- Támogatott oszlop típusok (a többit ilyenekre kell konvertálni):
  - TEXT (Java String)
  - INTEGER (Java long)
  - REAL (Java double)
- Az SQLite nem ellenőrzi a típust adatbeíráskor, tehát pl Integer érték automatikusan bekerül Text oszlopba szövegként

Android SQLite jellemzői 2/2

- Az SQLite adatbázis elérés file rendszer elérést jelent, ami miatt lassú lehet!
- Adatbázis műveleteket érdemes aszinkron módon végrehajtani (pl *AsyncTask* használata v. *Loader*)

OBJECT RELATION MAPPING (ORM)

Mi az ORM?

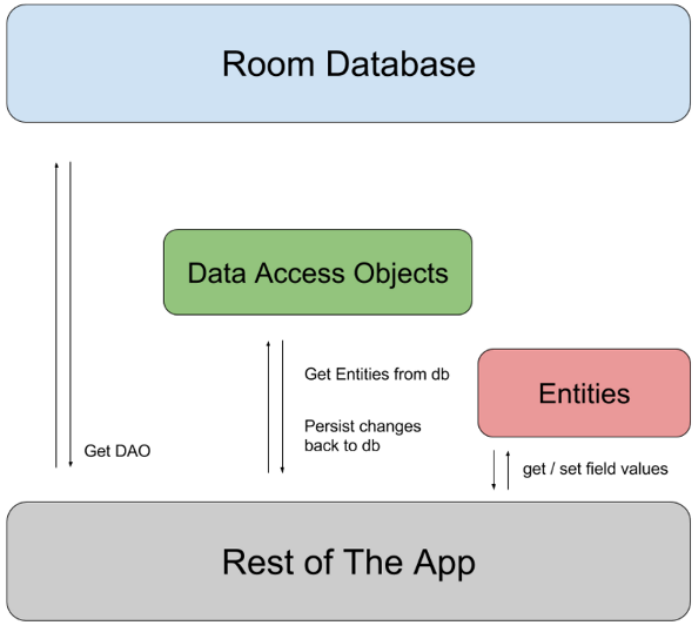
- Java/Kotlin objektumok tárolása relációs adatbázisban
- Alapelvek:
  - Osztálynév -> Tábla név
  - Objektum -> Tábla egy sora
  - Mező -> Tábla oszlopa
  - Stb.

ORM könyvtárak Androidon

- Sugar-ORM
  - > http://satyan.github.io/sugar/index.html
- Realm.io (NoSQL), nem SQLite-ot használ
  - > http://realm.io
- Objectbox
  - > https://objectbox.io/
- ORMLite
  - > http://ormlite.com/
- GreenDAO
  - > http://greendao-orm.com/

Room Persistence Library

- Absztrakciós réteg az SQLite felett
- SQLite teljes képességeinek használata
- Room architektúra:



Szálkezelés

- *Thread*
  - https://developer.android.com/guide/components/processes-and-threads.html
- A felhasználói felület csak a fő szálról módosítható:
  - *runOnUiThread(runnable: Runnable)*
- Szálakat le kell állítani
  - Biztosítani kell, hogy a run() függvény befejeződjön, ne maradjon végtelen ciklusban

Szál példa

```
private inner class MyThread : Thread() {
    override fun run() {
        while (threadEnabled) {
            runOnUiThread {
                Toast.makeText(this@MainActivity,
                        "Message", Toast.LENGTH_LONG).show()

            }
            Thread.sleep(6000)
        }
    }
}



MyThread().start()
```

Room példa - Entity

```
@Entity(tableName = "grade")
data class Grade(
        @PrimaryKey(autoGenerate = true) var gradeId: Long?,
        @ColumnInfo(name = "studentid") var studentId: String,
        @ColumnInfo(name = "grade") var grade: String
)
```

Room példa - DAO

```
@Dao
interface GradeDAO {
    @Query("""SELECT * FROM grade WHERE grade="B"""")
    fun getBGrades(): List<Grade>

    @Query("SELECT * FROM grade")
    fun getAllGrades(): List<Grade>

    @Query("SELECT * FROM grade WHERE grade = :grade")
    fun getSpecificGrades(grade: String): List<Grade>

    @Insert
    fun insertGrades(vararg grades: Grade)

    @Delete
    fun deleteGrade(grade: Grade)
}
```

RoomDatabase

```
@Database(entities = arrayOf(Grade::class), version = 1)
abstract class AppDatabase : RoomDatabase() {

    abstract fun gradeDao(): GradeDAO

    companion object {
        private var INSTANCE: AppDatabase? = null

        fun getInstance(context: Context): AppDatabase {
            if (INSTANCE == null) {
                INSTANCE = Room.databaseBuilder(context.applicationContext,
                        AppDatabase::class.java, "grade.db").build()
            }
            return INSTANCE!!
        }

        fun destroyInstance() {
            INSTANCE = null
        }
    }
}
```

Room használat

- Insert

```
val grade = Grade(null, etStudentId.text.toString(),
          etGrade.text.toString())

val dbThread = Thread {
    AppDatabase.getInstance(this@MainActivity).gradeDao().insertGrades(grade)
}
dbThread.start()
```

- Query

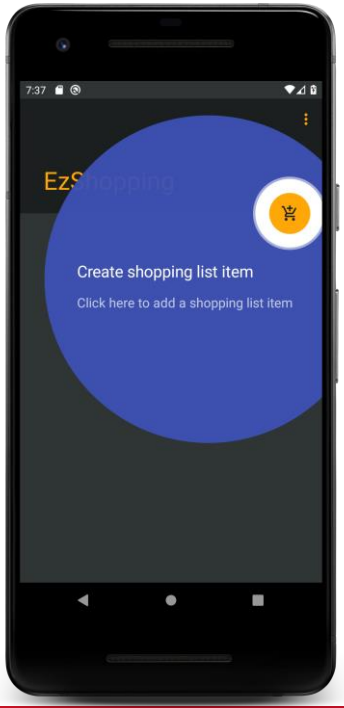
```
val dbThread = Thread {
        val grades = AppDatabase.getInstance(this@MainActivity).gradeDao()
                .getSpecificGrades("A+")
        runOnUiThread {
            tvResult.text = ""
            grades.forEach {
                tvResult.append("${it.studentId} ${it.grade}\n")
            }
        }
}
dbThread.start()
```

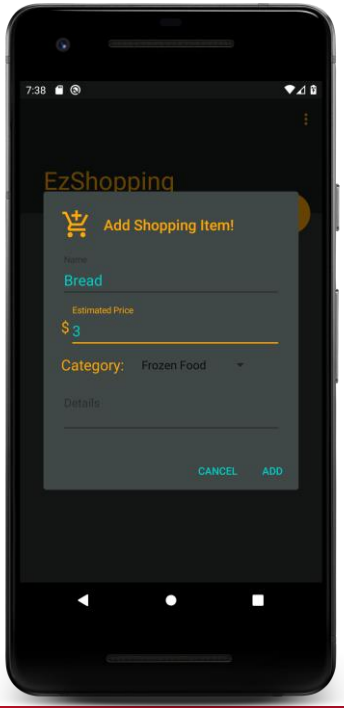
Adatbázis verzió növelés – migration policy

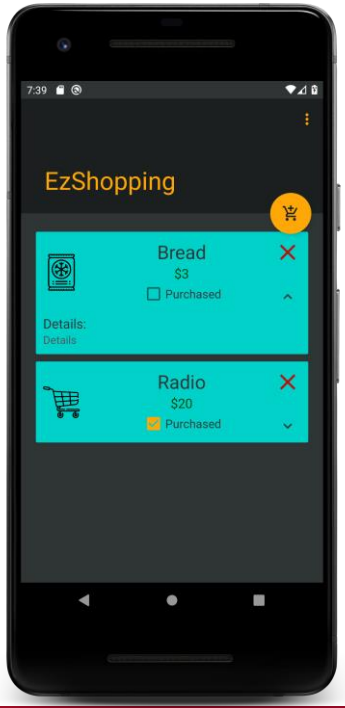
- https://stackoverflow.com/questions/44273272/android-room-persistent-library-how-to-change-database-version

Házi feladat

- Todo alkalmazásban a checkbox-ot összekötni adatbázissal
- Bevásárló lista alkalmazás kiegészítése adatbázis kezeléssel







Összefoglalás

- RecyclerView további képességek
- Perzisztens adattárolás
  - >SQLite
  - >SharedPreferences
- Fragmentek

Köszönöm a figyelmet!

peter.ekler@aut.bme.hu