

# Jegyzőkönyv

Mobil Programozási Alapok

Féléves feladat

Bevásárlólista

Készítette: **Takács Bálint Zétény**

Neptunkód: **GJWXEU**

Dátum: **2026.01.12.**

**Miskolc, 2026**

## **Tartalomjegyzék:**

1. [Tartalomjegyzék](#)
2. [Bevezetés](#)
3. [Az alkalmazás kinézete](#)
4. [Az alkalmazás működése](#)

## **Bevezetés**

Ebben a projektben egy egyszerű, de teljes értékű bevásárlólistás alkalmazást készítettem Androidra, amit Kotlin nyelven írtam. Az alkalmazás képes elmenteni a termékeket és mennyiségeket, listázni őket, új elemeket hozzáadni, meglevőket szerkeszteni és törölni is.

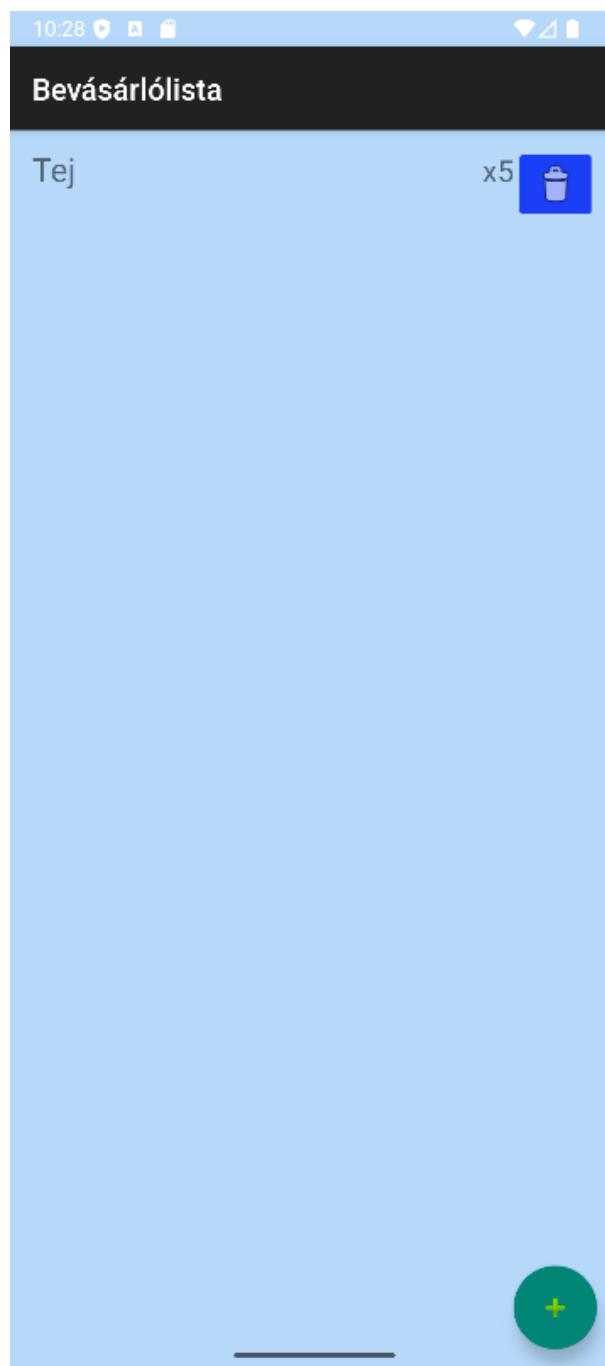
Room adatbázist használtam az adatok tárolására, ami SQLite alapú, de sokkal egyszerűbben kezelhető. RecyclerView-val jelenik meg a lista, a színeket és gombokat pedig Material Design szerint alakítottam ki.

Az alkalmazásban látszik a teljes bevásárlólista, lent egyplusz gombbal lehet új terméket felvenni, minden elemnél van törlési és szerkesztési lehetőség is. A szerkesztésnél az űrlap automatikusan kitöltődik a meglévő adatokkal.

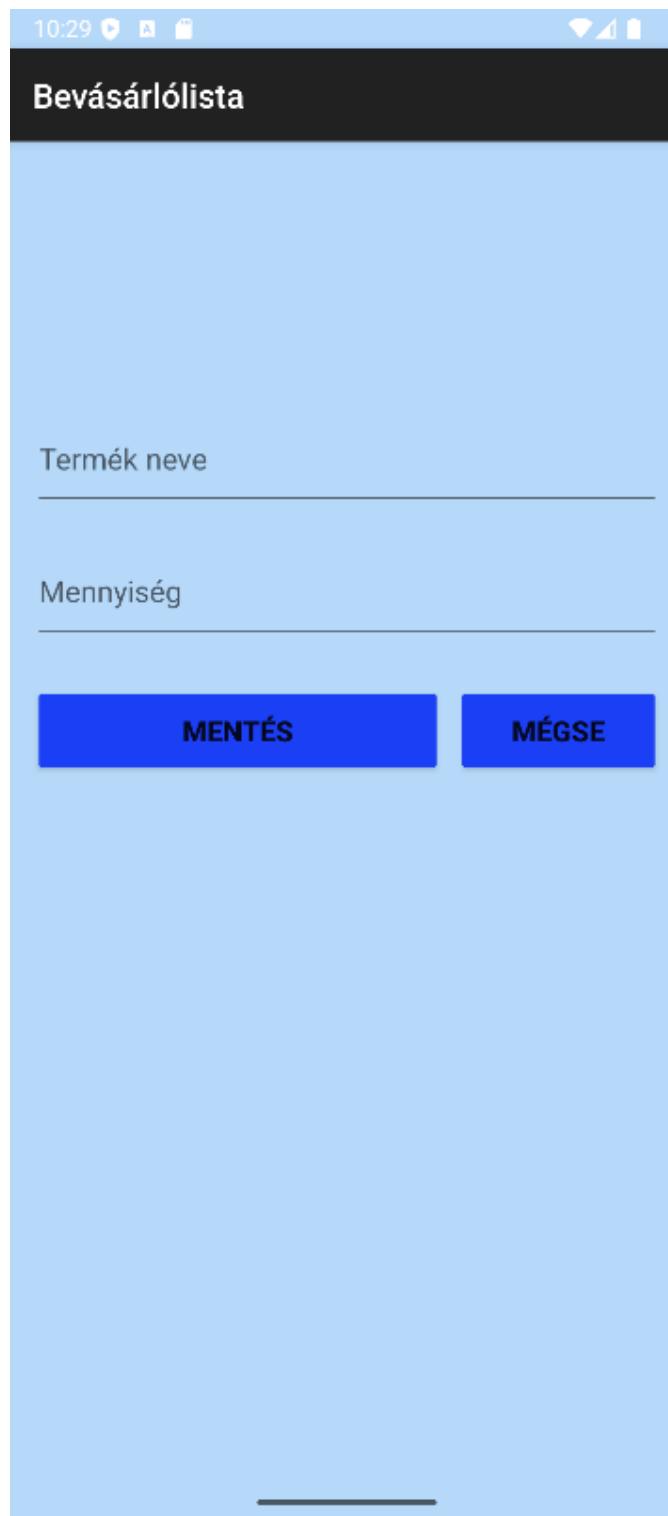
Meg kellett oldanom néhány layout problémát, például a gombok méretét és elrendezését, a Toolbar alá rejtett mezőket, valamint a szerkesztés logikáját.

## Az alkalmazás kinéze

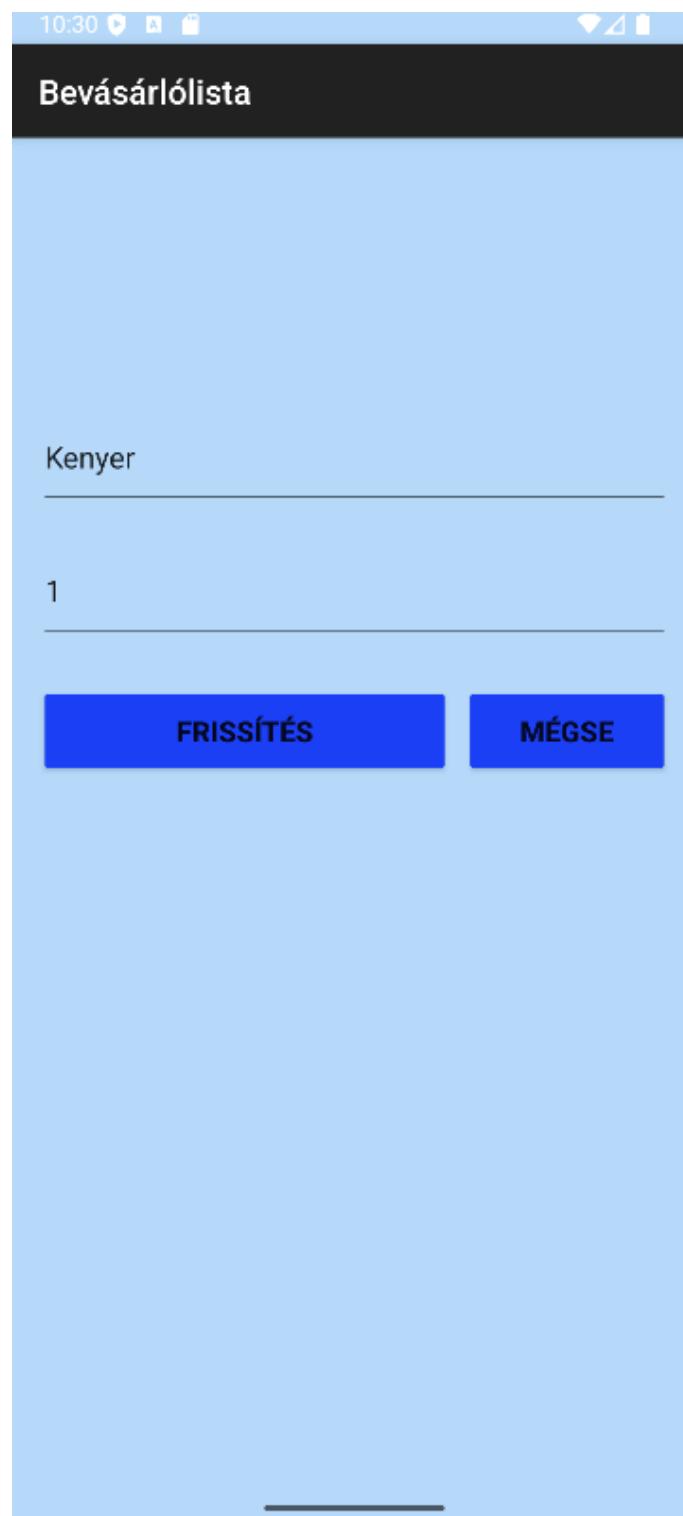
**Főoldal(lista):** Egy listában tartalmazza az elemeket, nevüket és mennyiségüket. A lista minden eleménél van egy törlés gomb, rákattintva lehet kitörölni az adott elemet. Az elemre kattintva pedig lehet szerkeszteni, a nevét és a mennyiségét egyaránt. A jobb alsó sarokban lévő + jellel lehet hozzáadni új elemeket a listához.



**Hozzáadás:** Elem nevét és mennyiségét megadva lehet hozzáadni a listához, de ha a felhasználó meggondolja magát, akkor visszaléphet hozzáadás nélkül.



**Szerkesztés:** Az elem nevére kattintva lehet szerkeszteni a tulajdonságait, majd a frissítés gombra nyomva lehet véglegesíteni, de ha a felhasználó itt is meggondolná magát, akkor vissza lépet szerkesztés nélkül.



## Az alkalmazás működése

- **MainActivity.kt:** Ez az alkalmazás fő képernyője, ahol a bevásárlólista látható. Itt hoztam létre a RecyclerView-t, ami a lista elemeit jeleníti meg. A ViewModel-en keresztül lekértem az összes terméket az adatbázisból, amit LiveData figyel, így ha valamit módosítok, a lista azonnal frissül. A képernyő alján lévő FAB gombbal lehet az új elem felvenni.

```
itemViewModel.allItems.observe(this, Observer { items ->
    adapter.submitList(items)
})
```

1. ábra: LiveData megfigyelésével a lista azonnal frissül, ha változás történik az adatbázisban.

- **NewItemActivity.kt:** Ez az űrlap képernyő, ahol új terméket lehet felvenni vagy meglévőt szerkeszteni. Két szöveges mező van benne (név és mennyiség), valamint Mentés és Mégse gombok. Az intent-ből megnéztem, hogy "add" vagy "edit" módban van-e, és ha szerkesztésről van szó, akkor automatikusan kitölți a mezőket a meglévő adatokkal. A Mentés gombra létrehozok egy Item objektumot, amit RESULT\_OK kóddal visszaadok a MainActivity-nek.

```
val mode = intent.getStringExtra("mode") ?: "add"
if (mode == "edit") {
    etName.setText(intent.getStringExtra("ITEM_NAME")
}
```

2. ábra: Intent EXTRA-ból megnézi, hogy szerkesztés vagy új elem, és kitölți a mezőket.

- **ItemDao.kt:** Ez a Room adatbázis interfész, ami az összes adatbázis műveletet kezeli. A @Query lekéri az összes elemet, @Insert új terméket ad hozzá, @Update meglévőt módosít, @Delete pedig töröl.

```
@Update
suspend fun update(item: Item)
```

3. ábra: Room annotációval SQL UPDATE parancs automatikusan generálódik.

- **Item.kt:** Ez a Room Entity osztály, ami megmondja, hogy az items táblában milyen oszlopok legyenek. id az elsődleges kulcs (autoGenerate), name String és quantity Int típusú. A @Entity annotációval Room felismeri, hogy ez egy tábla.

```
@Entity(tableName = "items")
data class Item(
    @PrimaryKey(autoGenerate = true)
    val id: Long = 0,
    val name: String,
    val quantity: Int = 1
) : Serializable
```

4. ábra: Room Entity - megmondja a tábla szerkezetét és az oszlopokat.

- **AppDatabase.kt:** A Room Database absztrakt osztály, ami összeköti az Entity-t a DAO-val. Singleton mintával készült, hogy csak egyszer jöjjön létre az adatbázis.

```
@Database(entities = [Item::class], version = 1, exportSchema =
false)
abstract class AppDatabase : RoomDatabase() {
    abstract fun itemDao(): ItemDao
```

5. ábra: Room Database összeköti az Entity-t a DAO-val.

- **item list item.xml**: Ez minden lista sor layout-ja. Egy LinearLayout-ban van a termék neve (TextView), mellette a Törlés gomb.

```
<TextView  
    android:id="@+id/tv_name"  
    android:layout_width="0dp"  
    android:layout_height="wrap_content"  
    android:layout_weight="1"  
    android:textSize="22sp" />
```

6. ábra: Listában szereplő nevek szövegdobozza.

- **activity new item.xml**: Az új elem űrlap layout-ja. Függőleges LinearLayout-ban vannak két EditText (név és mennyiség) és egy vízszintes LinearLayout a két gombbal.

```
<Button  
    android:id="@+id/btn_save"  
    android:text="@string/save"  
    android:layout_width="0dp"  
    android:layout_height="56dp"  
    android:layout_weight="2"  
    android:minHeight="56dp"  
    android:layout_marginEnd="8dp"  
    android:backgroundTint="@color/blue3"  
    android:textStyle="bold"  
    android:textSize="17sp"/>
```

7. ábra: Hozzáadásnál lévő Mentés gomb.