JEGYZŐKÖNYV

Mobil programozási alapok

Féléves feladat

Puss in Boots - cipő nyilvántartó rendszer

Készítette: **Zagyva Zsombor Gábor**
Neptunkód: **FXVAUU**

Miskolc, 2025

Tartalomjegyzék

A projekt célja és feladatleírás

Android alkalmazás készítése Kotlin nyelven, amely cipők nyilvántartását kezeli **Room ORM** (SQLite) segítségével. A rendszer teljesíti a CRUD-követelményeket:

- **Listázás** (RecyclerView)
- **Felvitel** (Dialog – mennyiséggel, több rekord egyszerre)
- **Módosítás** (Dialog – **brand nem változtatható**)
- **Törlés** (jobbra húzás / swipe)

Kiegészítők:

- **Keresés** (SearchView – címkékben: brand, subBrand, color, size)
- **Téma-váltás** (Light/Dark – AppCompatDelegate)
- **Belépő képernyő** (Login – dummy felhasználó: admin / admin a strings.xml-ben)

Fő funkciók

- **Bejelentkezés**
  - Egyszerű űrlap validációval; csak a megadott admin hiteles.
- **Cipők listázása**
  - Kártyaszerű lista: ikon (brand alapján), subBrand, brand, size, color.
- **Felvitel (CREATE)**
  - **Dialog** több mezővel (brand → subBrand és color dinamikusan töltődik).
- **Mennyiség** mező: egyszerre több azonos cipő is felvihető.
  - Beszúrás Room-on keresztül, visszatérő id-k alapján lista frissítés.
- **Szerkesztés (EDIT)**
  - A sor jobb szélén **ceruza** ikon.
  - **Brand nem módosítható**, a többi mező változtatható.
- **Törlés (DELETE)**
  - Jobbra húzásra törlés, adapter és DB frissítése.
- **Keresés**
  - Címkealapú kliens oldali szűrés (brand / subBrand / color / size).
- **Téma**
  - Kapcsolóval állítható világos/sötét téma.

A projekt struktúrája(részlet)

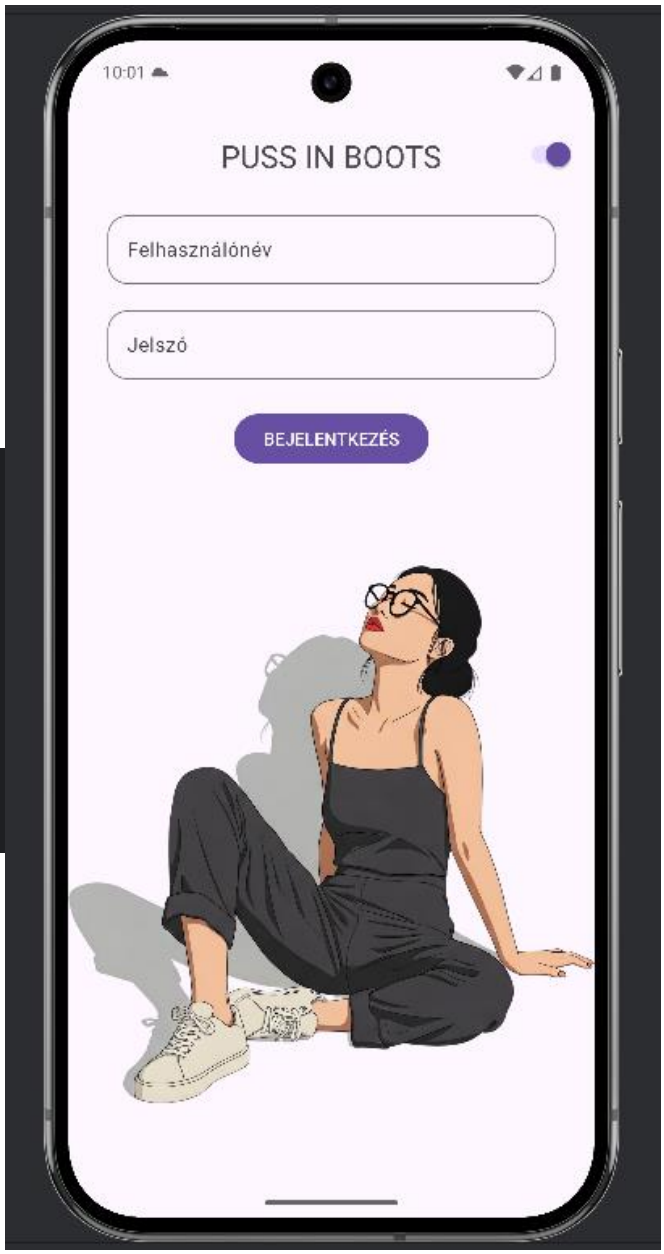
```
app
├─ manifests
│  └─ AndroidManifest.xml
│
├─ kotlin+java
│  └─ com.example.shoeregistry
│     ├─ AppDatabase.kt
│     ├─ DialogShoe.kt
│     ├─ LoginActivity.kt
│     ├─ MainActivity.kt
│     ├─ ShoeAdapter.kt
│     ├─ ShoeDao.kt
│     └─ ShoeEntity.kt
│
├─ res
│  ├─ drawable
│  │  ├─ adidas.png
│  │  ├─ home_screen.png
│  │  ├─ ic_launcher_background.png
│  │  ├─ ic_launcher_foreground.png
│  │  ├─ icon.png
│  │  ├─ nike.png
│  │  ├─ puma.png
│  │  └─ reebok.png
│  │
│  ├─ layout
│  │  ├─ activity_login.xml
│  │  ├─ activity_main.xml
│  │  ├─ dialog_shoe.xml
│  │  └─ item_shoe.xml
│  │
│  └─ values
│     └─ strings.xml
│
└─ …
```

- **manifests/AndroidManifest.xml** – Az app belépési pontjai (LAUNCHER: LoginActivity), engedélyek, alkalmazás téma.
- **kotlin+java/com.example.shoeregistry/**
    - **AppDatabase.kt** – Room adatbázis singleton; shoeDao() elérése.
    - **ShoeEntity.kt** – Room @Entity (mezők: id, subBrand, brand, size, color, imageResId).
    - **ShoeDao.kt** – Room DAO: getAll(), insertAll(), updateShoe(), delete(), deleteAll().
    - **LoginActivity.kt** – Bejelentkezés (dummy admin/admin),téma-váltás
    - **MainActivity.kt** – Lista megjelenítés (RecyclerView), keresés (SearchView), hozzáadás (FAB + dialog), törlés (swipe), téma váltás.
    - **DialogShoe.kt** – Közös CREATE/EDIT dialógus: spinnerek, mennyiség csak CREATE-ben, korutinban DB-műveletek.
- **ShoeAdapter.kt** – RecyclerView adapter: kártya bind, márkaikon, ceruza (EDIT) gomb callback.
- **res/drawable/** – Márkaikonok, alap ikonok
- **res/layout/**
    - **activity_login.xml** – Login képernyő
    - **activity_main.xml** – Főképernyő: cím, theme switch, SearchView, RecyclerView, FAB.
    - **dialog_shoe.xml** – Felvitel/szerkesztés
    - **item_shoe.xml** – Listaelem kártya
- **res/values/strings.xml** – Szöveg erőforrások

Felhasználói felületet bemutatása

Login (activity_login.xml)

ConstraintLayout + Material TextInput-ok + Login gomb. Sikeres belépés → MainActivity.



Téma gomb:

```
themeSwitch.setOnClickListener {
    val mode = if (themeSwitch.isChecked)
        AppCompatDelegate.MODE_NIGHT_NO
    else
        AppCompatDelegate.MODE_NIGHT_YES

    AppCompatDelegate.setDefaultNightMode(mode)
}
```

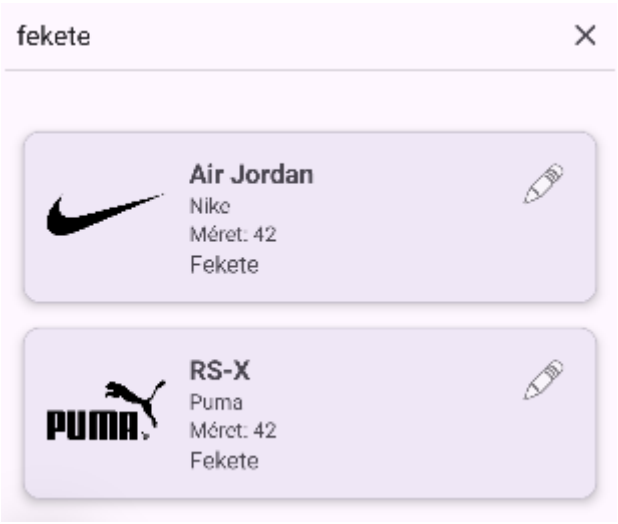
```
    loginButton.setOnClickListener {
        val username = usernameInput.text?.toString()?.trim() ?: " "
        val password = passwordInput.text?.toString()?.trim() ?: " "

        if(username.isEmpty() ||  password.isEmpty() ){
            Toast.makeText(
                context = this,
                "Kérlek add meg a bejelentkezési adataid",
                duration = Toast.LENGTH_SHORT).show()
            return@setOnClickListener
        }
        if(username != "admin" || password != "1234"){
            Toast.makeText(
                context = this,
                "Hibás felhasználónév/jelszó",
                duration = Toast.LENGTH_SHORT).show()
            return@setOnClickListener
        }
        val intent = Intent( packageContext = this,  cls = MainActivity::class.java)
        startActivity(intent)

    }
```

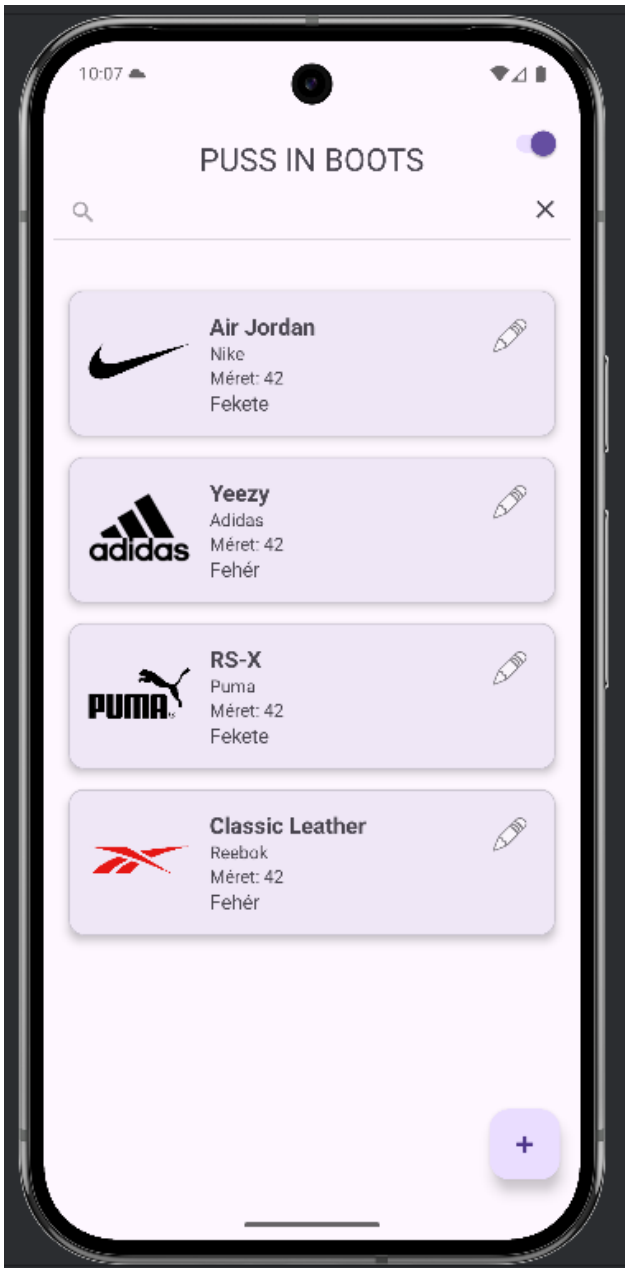
Főképernyő (activity_main.xml)

Fent cím + téma-kapcsoló, alatta **SearchView**, a tartalom **RecyclerView**, jobb alsó sarokban **FAB** („+”) a felvitelhez.

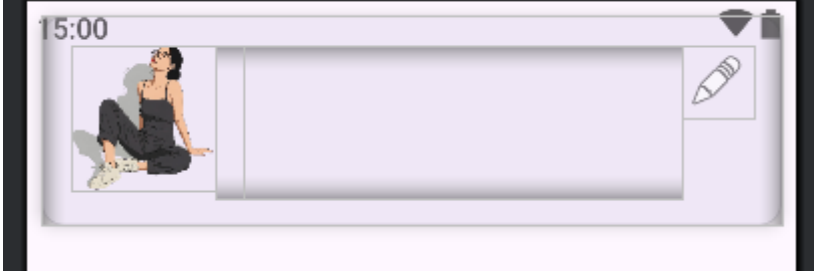
A keresés a kártyák minden mezőjére kiterjednek

lehet márka, almárka, méret, szín szerint.






```
    // Kereső: szűrés a forráslistán, adapternek a szűrt lista megy
    searchView.setOnQueryTextListener(object : SearchView.OnQueryTextListener {
        override fun onQueryTextSubmit(query: String?): Boolean {
            filterShoes( query = query.orEmpty())
            return true
        }

        override fun onQueryTextChange(newText: String?): Boolean {
            filterShoes( query = newText.orEmpty())
            return true
        }
    })
}
```



Listaelem (item_shoe.xml)

MaterialCardView –lineralayout– balra márka ikon, középen adatok, jobbra **ceruza** gomb (edit).

Dialog (dialog_shoe.xml)

Automatikusan feltölti előre a listák első elemével.

Az alamárka és a színek spinner-je mapOf-al van megoldva

```
val brands = listOf("Nike", "Adidas", "Puma", "Reebok")

val subBrands = mapOf(
    "Nike" to listOf("Air Jordan", "Air Max", "Dunk", "Cortez"),
    "Adidas" to listOf("Yeezy", "Originals", "Predator", "Ultraboost"),
    "Puma" to listOf("RS-X", "Suede", "Cali", "Future Rider"),
    "Reebok" to listOf("Classic Leather", "Nano", "Club C", "Zig Kinetica"),
)
```

Ha kiválasztunk egy márkát. a hozzá tartozó szín és almárka jelenik meg a spinnerben

```
fun refreshSubBrandAndColor(selectedBrand: String) {
    val subAdapter = ArrayAdapter(
        context = requireContext(),
        resource = android.R.layout.simple_spinner_dropdown_item,
        objects = subBrands[selectedBrand] ?: emptyList()
    )
    spinnerSubBrand.adapter = subAdapter
```

Ha a mennyiséghez nem írunk semmit. akkor 1db cipőelem készül el.

Room modell és adathozzáférés

Entity

Az id automatikusan generálódik.

```
39 Usages
@Entity
data class ShoeEntity(
    @PrimaryKey(autoGenerate = true) val id: Int = 0,
    val subBrand: String,
    val brand: String,
    val size: Int,
    val imageResId: Int,
    val color: String
    )
```

DAO

```
5 Usages  1 Implementation
@Dao
interface ShoeDao {
    2 Usages  1 Implementation
    @Delete
    suspend fun delete(shoe: ShoeEntity)
    1 Usage  1 Implementation
    @Query( value = "SELECT * FROM ShoeEntity")
    suspend fun getAll(): List<ShoeEntity>

    1 Usage  1 Implementation
    @Insert
    suspend fun insertAll(shoes: List<ShoeEntity>): List<Long>

    1 Usage  1 Implementation
    @Update
    suspend fun updateShoe(shoe: ShoeEntity)

    1 Implementation
    @Query( value = "DELETE FROM ShoeEntity")
    suspend fun deleteAll()

}
```

itt vannak a **tiszta SQL/annotációs** műveletek az entitásokon. A Room a DAO alapján generálja a tényleges kódot.

RoomDatabase

```
8 Usages  1 Implementation
@Database(entities = [ShoeEntity::class], version = 1)
abstract class AppDatabase : RoomDatabase() {
    4 Usages  1 Implementation
    abstract fun shoeDao(): ShoeDao

    8 Usages
    companion object {
        3 Usages
        @Volatile private var instance: AppDatabase? = null

        4 Usages
        fun getDatabase(context: Context): AppDatabase =
            instance ?: synchronized( lock = this) {
                instance ?: Room.databaseBuilder(
                    context.applicationContext,
                    klass = AppDatabase::class.java,
                    name = "shoe_db"
                ).build().also { instance = it }
            }
    }
}
```