Jegyzőkönyv

Tantárgy nyilvántartó rendszer

Készítette:

Név: Kola Sándor

Neptun-kód: B8GU52

Tartalomjegyzék

1. A feladat leírása:

Mobil programozás alapjai – Android alkalmazás készítése Kotlin nyelven, Room ORM adatbázis használatával.

Az elkészített alkalmazás egy tantárgylista kezelő rendszer, amely támogatja az adatok:

- listázását,
- felvitelét,
- módosítását,
- törlését, valamint tartalmaz egy demo bejelentkezési felületet is.

2. Fejlesztési környezet:

Operációs rendszer: Windows 11

Fejlesztőkörnyezet: Android Studio

Programozási nyelv: Kotlin

UI technológia: Jetpack Compose

Adatbázis: Room ORM (SQLite)

Tesztelés: Fizikai Android telefon USB kapcsolattal

3. Az alkalmazás funkciói:

3.1 Bejelentkezés

Az alkalmazás indításakor egy bejelentkezési képernyő jelenik meg. A belépés egy demo felhasználóval történik.

- Felhasználónév: admin
- Jelszó: 1234

Sikeres bejelentkezés után a felhasználó átkerül a tantárgylista képernyőre. Hibás adatok esetén hibaüzenet jelenik meg.

Kódrészlet (bejelentkezés vezérlése a MainActivity-ben):

```
if (isLoggedIn) {
    SubjectListScreen(onLogout = { isLoggedIn = false })
} else {
    ModernLoginScreen { isLoggedIn = true }
}
```

3.2 Tantárgy hozzáadása:

A felhasználó megadhatja:

- a tantárgy nevét,
- a kredit értékét.

A „Hozzáadás” gomb megnyomásával új tantárgy kerül az adatbázisba, és azonnal megjelenik a listában.

Kódrészlet (tantárgy felvétele ViewModel-ből):

```
fun addSubject(name: String, credit: Int) {
    viewModelScope.launch {
        subjectDao.insertSubject(
            Subject(name = name, credit = credit)
        )
    }
}
```

3.3 Tantárgy módosítása:

A listában található minden tantárgy mellett megjelenik a „Szerkesztés” gomb ikonja.

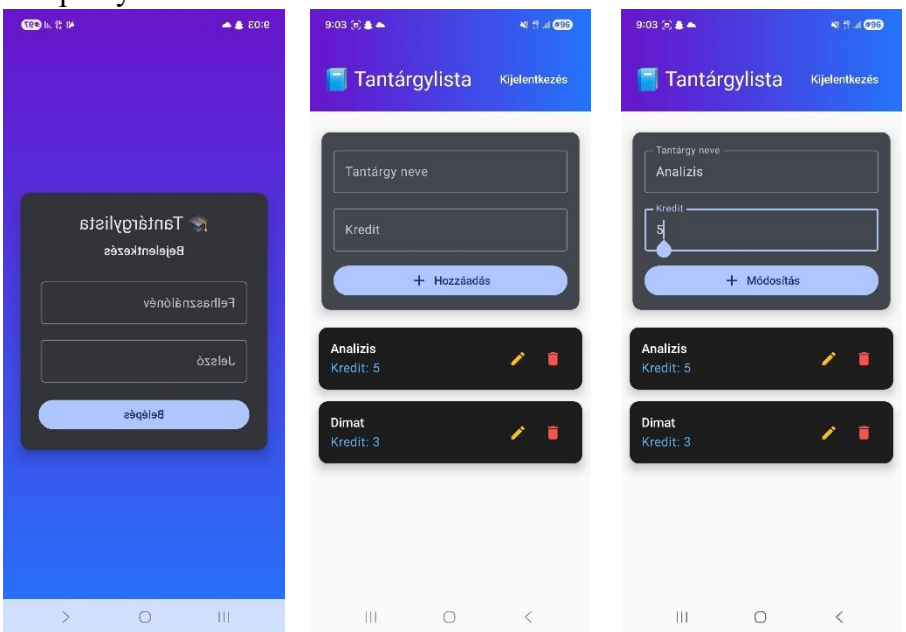
A gomb megnyomásakor a kiválasztott tantárgy adatai bekerülnek a beviteli mezőkbe, majd a módosítás menthető.

3.4 Tantárgy törlése:

Minden tantárgy mellett elérhető a „Törlés” gomb ikonja. A gomb megnyomásával a kiválasztott tantárgy véglegesen törlődik az adatbázisból.

3.5 Kijelentkezés:

A lista felső részén található „Kijelentkezés” gomb segítségével a felhasználó visszatérhet a bejelentkezési képernyőre.



4. Adatbázis felépítése:

4.1 Subject entitás:

A Subject osztály írja le egy tantárgy adatainak szerkezetét:

- id (elsődleges kulcs)
- name (tantárgy neve)
- credit (kredit érték)

4.2 SubjectDao:

A SubjectDao interfész tartalmazza az adatbázis műveletekhez szükséges függvényeket:

- lekérdezés (getAllSubjects)
- beszúrás (insertSubject)
- módosítás (updateSubject)
- törlés (deleteSubject)

```
@Dao
interface SubjectDao {

    @Query("SELECT * FROM subjects")
    fun getAllSubjects(): Flow<List<Subject>>

    @Insert
    suspend fun insertSubject(subject: Subject)

    @Update
    suspend fun updateSubject(subject: Subject)

    @Delete
    suspend fun deleteSubject(subject: Subject)
}
```

4.3 AppDatabase:

Az AppDatabase osztály kezeli a Room adatbázis példányosítását és biztosítja a Dao elérését az alkalmazás számára.

```
@Database(entities = [Subject::class], version = 1)
abstract class AppDatabase : RoomDatabase() {

    abstract fun subjectDao(): SubjectDao

    companion object {
        @Volatile
        private var INSTANCE: AppDatabase? = null

        fun getDatabase(context: Context): AppDatabase {
            return INSTANCE ?: synchronized(this) {
                val instance = Room.databaseBuilder(
                    context.applicationContext,
                    AppDatabase::class.java,
                    "subject_database"
                ).build()

                INSTANCE = instance
                instance
            }
        }
    }
}
```

5. ViewModel működése:

A SubjectViewModel felelős az alkalmazás üzleti logikájáért:

- kapcsolatot tart az adatbázissal,
- figyeli az adatváltozásokat Flow segítségével,
- biztosítja a hozzáadás, módosítás és törlés műveleteit a felhasználói felület számára.

```
class SubjectViewModel(application: Application) :
AndroidViewModel(application) {

    private val subjectDao =
        AppDatabase.getDatabase(application).subjectDao()

    private val _subjects = MutableStateFlow<List<Subject>>(emptyList())
    val subjects: StateFlow<List<Subject>> = _subjects

    init {
        viewModelScope.launch {
```

```
            subjectDao.getAllSubjects().collectLatest {
                _subjects.value = it
            }
        }
    }

    fun addSubject(name: String, credit: Int) {
        viewModelScope.launch {
            subjectDao.insertSubject(
                Subject(name = name, credit = credit)
            )
        }
    }

    fun updateSubject(subject: Subject) {
        viewModelScope.launch {
            subjectDao.updateSubject(subject)
        }
    }

    fun deleteSubject(subject: Subject) {
        viewModelScope.launch {
            subjectDao.deleteSubject(subject)
        }
    }
}
```

6. Felhasználói felület:

Az alkalmazás felhasználói felülete Jetpack Compose segítségével készült. A dizájn:

- modern,
- sötét tónusú,
- színes fejlécet tartalmaz gradient háttérrel,
- kártyás elrendezést használ a tantárgyak megjelenítésére.

7. Összegzés:

Az alkalmazás egy stabilan működő, modern megjelenésű tantárgykezelő rendszer, amely lehetővé teszi a tantárgyak egyszerű felvételét, módosítását és törlését. A Room adatbázisnak köszönhetően az adatok tartósan, biztonságosan tárolódnak, és azonnal frissülnek a felhasználói felületen.

A Jetpack Compose alapú felület letisztult, reszponzív és jól használható mobil eszközökön. A bejelentkezési és kijelentkezési funkció révén a program kezelése felhasználóbarát és

áttekinthető. A tesztelés során az alkalmazás megbízhatóan működött fizikai Android készüléken.

Összességében az elkészült program egy teljes funkcionalitású, esztétikus és stabil Android alkalmazás, amely megfelel a gyakorlati használat alapvető elvárásainak.