

JEGYZŐKÖNYV

Mobil Programozás

Féléves feladat

Mobil webshop létrehozása

Készítette: **Bartók-Balog Péter**
Neptunkód: **ASNQPH**

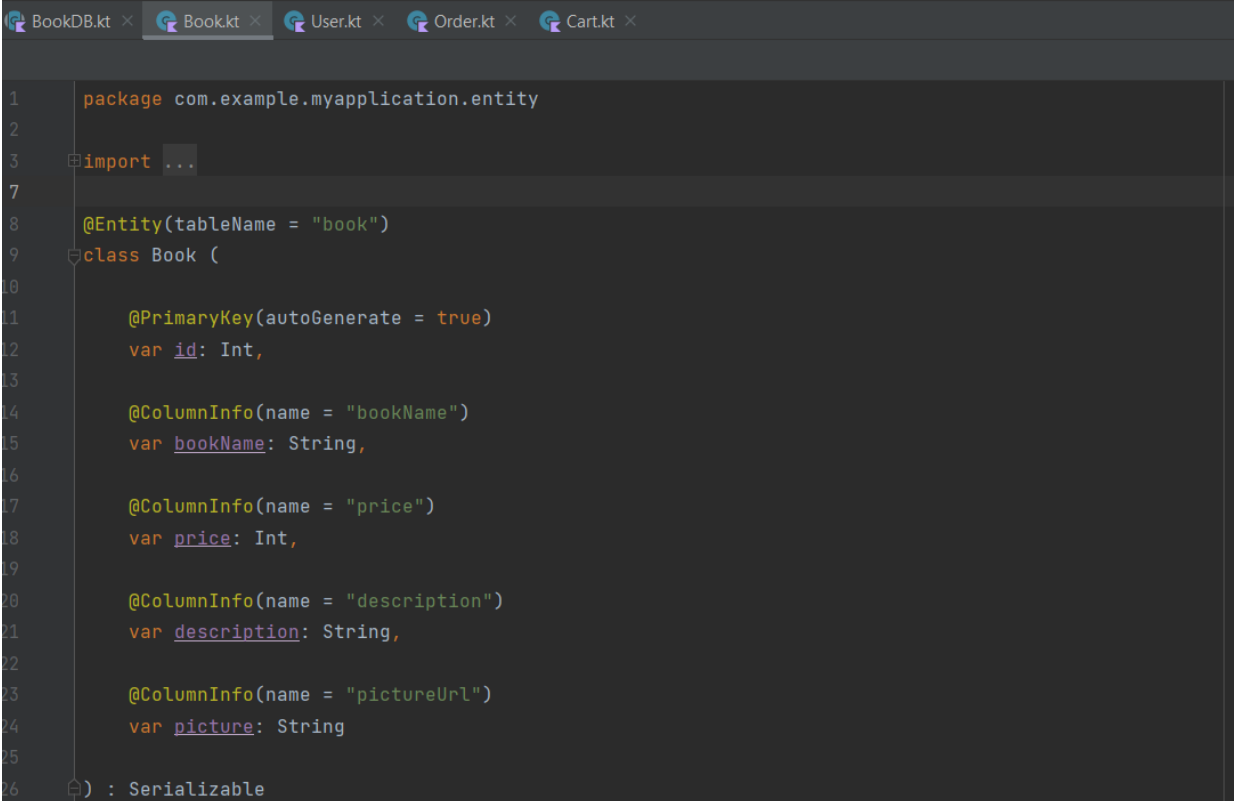
A feladat leírása:

A feladat egy mobil alkalmazás létrehozása Kotlin technológiát alkalmazva, a megadott feltételek alapján. A feladat megoldásához létre kell hozni egy bejelentkező felületet, illetve a regisztráció, és a bejelentkezés után, létre kell hozni egy valamilyen termék nyilvántartására alkalmas kezelőfelületet. A feladat megoldása kezdődik a backend létrehozásával, majd a frontend megalkotásával. A feladat teljesítéséhez szükséges egy dokumentáció, amiben bemutatásra kerül a felhasználói felület és a megírt kód.

Kód, és a felhasználói felület bemutatása:

Backend:

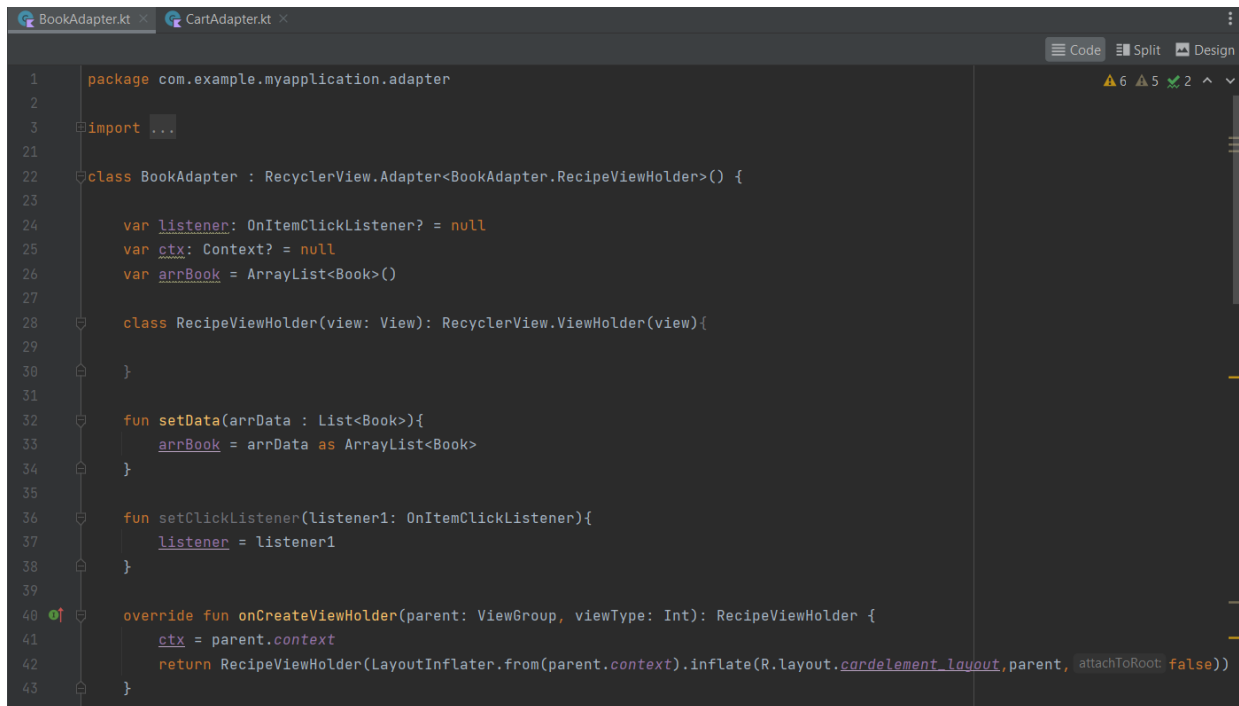
Kezdetben létrehozzuk a megfelelő fájlszerkezetet és létrehozzuk a kezdeti identitásainkat (táblákat) melyek adatait majd később fel akarjuk használni. Ahhoz, hogy ezekkel a táblákkal biztonságban tudjunk dolgozni, létrehozunk egy velük megegyező szerkező DAO gyűjteményt is, mellyel majd fogadunk, és továbbítunk adatokat a frontend és backend között.

A screenshot of an IDE window showing Kotlin code for a Book entity. The window has five tabs: BookDB.kt, Book.kt, User.kt, Order.kt, and Cart.kt. The Book.kt tab is active, displaying the following code:

```
1 package com.example.myapplication.entity
2
3 import ...
4
5
6
7
8 @Entity(tableName = "book")
9 class Book (
10
11     @PrimaryKey(autoGenerate = true)
12     var id: Int,
13
14     @ColumnInfo(name = "bookName")
15     var bookName: String,
16
17     @ColumnInfo(name = "price")
18     var price: Int,
19
20     @ColumnInfo(name = "description")
21     var description: String,
22
23     @ColumnInfo(name = "pictureUrl")
24     var picture: String
25
26 ) : Serializable
```

Ezt követően létrehozzuk az adaptereket, azaz az entitásokhoz kapcsolódó kezelőmodulokat, melyek majd a táblákkal történő alapvető, és táblaspecifikus

műveleteket fogják kezelni. Ilyen művelet az id alapján lekérdezés, a listázás, törlés, módosítás, létrehozás, számolás, vásárlás stb.



```
1 package com.example.myapplication.adapter
2
3 import ..
4
21
22 class BookAdapter : RecyclerView.Adapter<BookAdapter.RecipeViewHolder>() {
23
24     var listener: OnItemClickListener? = null
25     var ctx: Context? = null
26     var arrBook = ArrayList<Book>()
27
28     class RecipeViewHolder(view: View): RecyclerView.ViewHolder(view){
29
30     }
31
32     fun setData(arrData : List<Book>){
33         arrBook = arrData as ArrayList<Book>
34     }
35
36     fun setClickListener(listener1: OnItemClickListener){
37         listener = listener1
38     }
39
40     override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): RecipeViewHolder {
41         ctx = parent.context
42         return RecipeViewHolder(LayoutInflater.from(parent.context).inflate(R.layout.recipeelement_layout,parent, attachToRoot: false))
43     }
44 }
```

Frontend:

Amint a backend készen áll, folytatjuk a munkát a kliens oldallal, tehát azzal, amit a felhasználó fog fizikailag kezelni, és használni.

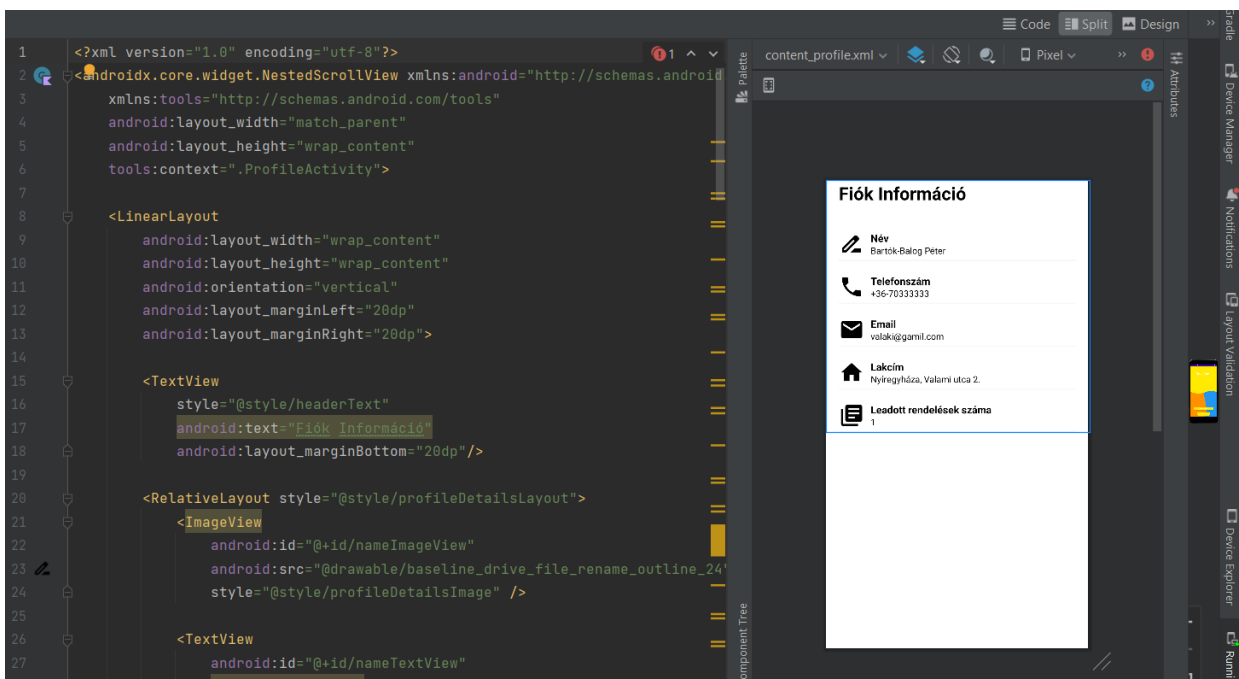
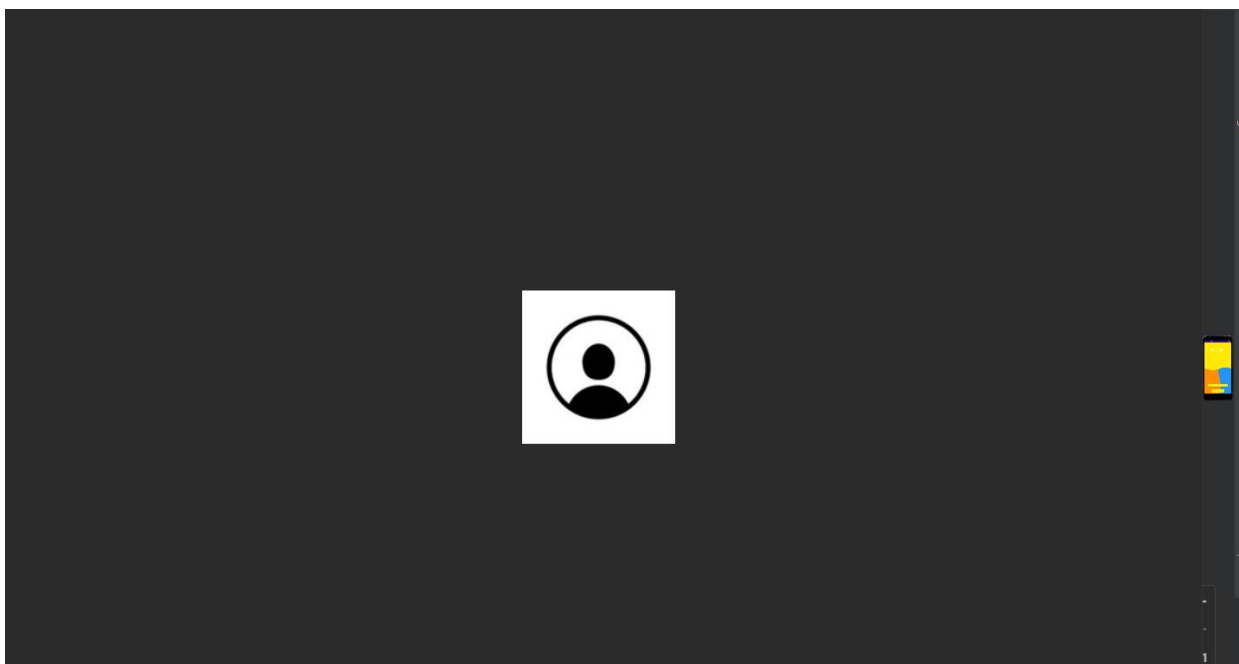
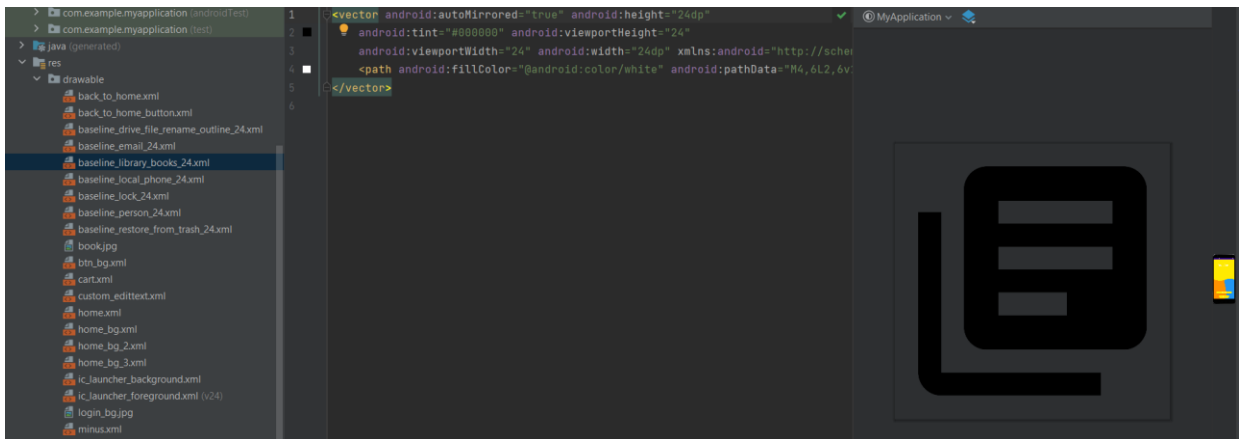
Minden fontosabb oldalhoz, kezelőhöz létrehozunk egy-egy aktivitást.

Elsősorban megírjuk a fájlban a komponens műveleteit, melyet majd a felhasználó az alkalmazás kezelésére fog használni, Behúzzuk a megfelelő importokat, példányosítunk, illetve inicializáljuk a változókat.

```
HomeActivity.kt>LoginActivity.ktProfileActivity.ktRegisterActivity.ktFirstScreenActivity.kt
1package com.example.myapplication
2
3import ...
4
5
6
7
8
9
10class HomeActivity : ComponentActivity() , View.OnClickListener {
11
12    private lateinit var btnStart: Button
13    private lateinit var btnLogin: Button
14
15    override fun onCreate(savedInstanceState: Bundle?) {
16        super.onCreate(savedInstanceState)
17        setContentView(R.layout.homescreen_layout)
18        val sharedPreferences = getSharedPreferences( name: "MyAppPrefs", Context.MODE_PRIVATE)
19        val editor = sharedPreferences.edit()
20
21        editor.remove("loggedInUser")
22        editor.apply()
23
24        btnStart = findViewById(R.id.btnStart)
25        btnStart.setOnClickListener(this)
26
27        btnLogin = findViewById(R.id.btnLogin)
28        btnLogin.setOnClickListener { it: View?
29            var intent = Intent( packageContext: this@HomeActivity, LoginActivity::class.java)
30            startActivity(intent)
```

```
HomeActivity.kt>LoginActivity.ktProfileActivity.ktRegisterActivity.ktFirstScreenActivity.kt
1package com.example.myapplication
2
3import ...
4
5
6
7
8
9
10
11
12
13
14
15
16class LoginActivity : AppCompatActivity() {
17
18    private lateinit var btnLogin: Button
19    lateinit var username: EditText
20    lateinit var password: EditText
21    lateinit var goToRegister: LinearLayout
22
23    lateinit var myDb : BookDB
24    lateinit var userDao: UserDao
25
26    override fun onCreate(savedInstanceState: Bundle?) {
27        super.onCreate(savedInstanceState)
28        setContentView(R.layout.login_layout)
29
30        btnLogin = findViewById(R.id.loginButton)
31        username = findViewById(R.id.username)
32        password = findViewById(R.id.password)
33        goToRegister = findViewById(R.id.goToRegister)
34
35        myDb = Room.databaseBuilder( context: this, BookDB::class.java, name: "myApp2")
36            .allowMainThreadQueries().fallbackToDestructiveMigration().build()
37        userDao = myDb.getUserDao()
38
39        btnLogin.setOnClickListener(View.OnClickListener { it: View?
40
```

Amint ezzel megvagyunk behúzzuk a fizikális kezelőfelülethez szükséges resources fájlokat, azaz ikonokat, formákat, elrendezéseket, színeket, stb. Az android studio számos default vektort, layout-ot tartalmaz, amik közül tudunk válogatni, illeszkedve az alkalmazás témájához.



Az alkalmazás képes a következőkre:

- Regisztráció
- Bejelentkezés
- Kijelentkezés
- Könyvek listázása
- Könyvek módosítása
- Könyvek, címek törlése
- Új felhasználó létrehozása
- Könyvek vásárlása

