**1.dia**

Schütz Máté vagyok, elsőéves mérnökinformatikus MSc-s hallgató, szoftverfejlesztés specializáción és ez az Önlab1-es projektem bemutatója. Projektem témája egy Online Bevásárló Lista app elkészítése Android platformon. Azért ezt a témát választottam, mert egyrészt korábban már foglalkoztam Android appok készítésével, és ebben a félévben mélyebben is bele akartam magam vetni a témába. Illetve a háztartásban, ahol élek, folyamatos probléma volt, hogy a bevásárlások során mindig elfelejtődött néhány termék, vagy akár dupla annyi lett vásárolva, mint amennyire szükség volt. Ennek megoldására jött az ötlet az app elkészítéséhez. Az alkalmazás lényege, hogy a bevásárló listák, illetve a rajtuk lévő termékek online vannak eltárolva a Firebase Firestore funkciója segítségével, és akár több felhasználó is feliratkozhat egy adott listára, így közösen szerkeszthetik azt.

**2.dia**

Néhány szó a felhasznált technológiákról: az app Android platformra készült és Kotlinban írtam, mivel ma már ez a Google által is preferált androidos fejlesztési nyelv. Az alkalmazásom központi eleme a Google által fejlesztett Firebase platform. Az Authentication modult a felhasználó kezeléshez, a Firestore-t a listák és termékek online tárolásához, a Storage modult pedig a felhasználók profilképeinek tárolására használtam. Mivel a profilképek online vannak tárolva, ezért a megjelenítésükhöz a Glide open-source library-t használtam, hisz ez nagyban megkönnyíti az implementációt. A dialógusablakokat a Material Dialogs ingyenes, open-source könyvtárral készítettem el, erről még később részletesebben is lesz szó.

**3.dia**

Most röviden bemutatnám az alkalmazás használatának menetét. Ahogy a felhasználó elindítja az appot, az üdvözlő képernyő fogadja, ahonnan csak úgy tud tovább lépni, ha bejelentkezik, vagy regisztrál egy új fiókkal. Az app a sima e-mail címes és Google fiókos regisztrációt is támogatja. A bejelentkezési és regisztrációs képernyőket a FirebaseUI Authentication moduljával készítettem el. Ha a felhasználó egyszer bejelentkezett, onnantól kezdve az alkalmazás minden indításánál automatikusan a főképernyőre lesz navigálva. A főképernyőn az alsó navigációs sávval választhatunk a saját listák megtekintése, új lista hozzáadása és a profil megjelenítése funkciók között.

**4.dia**

Ha a listák közül rányomunk az egyikre, megjelennek az adott listán lévő termékek. Egy adott termékre nyomva a neve melletti checkbox kipipálódik, jelezve, hogy vásárlás közben a terméket már a kosarunkba tettük. A vásárlás gombra nyomva az összes kipipált termék eltűnik a listánkról. A jobb oldalt láthatjuk a profil képernyőt, ahol a felhasználó email címe, neve és profilképe jelenik meg. A profilképre nyomva lecserélhetjük azt egy tetszőleges képre az Android készülékünkről.

**5.dia**

Itt az alkalmazásom osztálydiagramja látható, amelyen csak az architektúra szempontjából fontos osztályokat ábrázoltam. Az appom a Model-View-ViewModel architekturális mintát követi. Azért döntöttem emellett, mivel a Google is ezt ajánlja az egyszerűbb Android alkalmazásokhoz. A fenti User, ShoppingList és ShoppingItem osztályok képezik a Model réteget, míg a MainViewModel és a ShoppingListViewModel a ViewModel réteget, valamint alul láthatóak a View réteg osztályai. Az alkalmazásom egy single-activity app, amely azt jelenti, hogy egyetlen Activity-vel rendelkezik és erre rajzolódnak ki a különböző képernyőknek megfelelő fragmensek.

**6.dia**

Most bemutatnék néhány érdekesebb részt, illetve funkciót az alkalmazásomból. Az első, a termékek valósidejű frissítése egy adott listában. A Firestore API-ja lehetővé teszi, hogy egy adott kollekcióra beregisztáljunk egy listener-t, így minden alkalommal amikor a kollekció változik, az appunk kap róla egy snapshot-ot a szervertől. Itt a kódomban **itemsCollectionReference** egy referencia a megfelelő Firestore kollekcióra, és a listenToShoppingItems() függvényemben az addSnapshotListener() függvénnyel megadhatom, hogy mi történjen, amikor az alkalmazásom kap egy új snapshotot a Firebase-től. Ebből látszik, hogy a Firebase mennyire leegyszerűsíti a backenddel való kommunikációt.

**7.dia**

A Material Dialogs egy githubon elérhető library, aminek segítségével néhány sornyi kóddal szép Material stílusú dialógusablakokat tudunk létrehozni az appunkban. Itt látszik a kódomban, hogy mennyire egyszerű létrehozni például egy úgynevezett bottomsheet. Ez az az alulról beúszó dialógusablak, amely egy termékre hosszan nyomva jelenik meg.

**8.dia**

Az utolsó érdekesség, amit megmutatnék, az az alkalmazás témájának beállítása. Manapság elég fontos, hogy a felhasználó választhasson világos és sötét kinézetek között. Ezt az AndroidX Preference Library-vel viszonylag egyszerűen meg lehet oldani. Nekünk csak az egyes beállításokat kell megadnunk egy xml resource fájlban, és a library létrehozza ehhez a User Interface-t és a SharedPreferences API-val elintézi nekünk a beállítások perzisztálását is. A kódban látszik, hogy egy sima listenert állítunk a themePref preference-re és annak az új értékétől függően egyszerűen átállíthatjuk az app témáját. Ezt az értéket az app minden indulásakor kiolvasom a SharedPreferences tárból, és a felület kirajzolása előtt beállítom a megfelelő témát.

**9.dia**

Továbbfejlesztési lehetőségek például a Push Notificationök megvalósítása a Firebase Cloud Messaging moduljával, így a felhasználót értesíthetjük akkor is, amikor nem fut az app a készülékén. Emellett további fejlesztés lenne, hogy különböző szerepeket és szabályokat definiálhassunk a listákhoz, így pl egy listát csak a készítője szerkeszthet, mindenki más csak megtekintheti. Emellett egy további lehetőség a Firestore lecserélése, és egy saját backend használata a listák tárolásához.

**10.dia**

Köszönöm a figyelmet, és ha vannak kérdések, akkor szívesen válaszolok rájuk.