Követelményspecifikáció

Fast Fashion Asszisztens

**Feladatkíírás**

Manapság állítólag egy ruha csak pár hétig divatos, utána már nem trendi. A feladat egy olyan divatkövető alkalmazás készítése, mely nyilvántartja a felhasználó különböző kategóriájú ruhadarabjainak stílusát és "életkorát". A gyűjtött adatok alapján pedig értesítéseket, illetve tanácsokat ad (pl.: webshop linkek), hogy miből kell újat venni, illetve mikor lesz esedékes a következő "micro season". Ezen kívül beviteli felülettel is rendelkezik, melyen a frissen vásárol ruhadarab tulajdonságait, illetve képeit is rögzíthetjük.

**A fejlesztői csapat**

A csapat tagjai:

|  |  |  |
| --- | --- | --- |
| Csapattag neve | Neptun-kód | E-mail cím |
| Schneider József | I4ZJUD | [Jozsef.schneider1996@gmail.com](mailto:Jozsef.schneider1996@gmail.com) |
| Imre Gábor | CJXKUS | igaborynwa@gmail.com |

A csapatban a frontenddel Imre Gábor, a backenddel Schneider József foglalkozott.

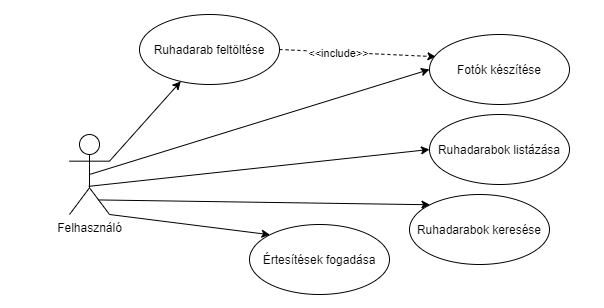
**Részletes feladatleírás**

A felhasználóknak lehetőségük van a frissen vásárolt ruhadarabokat feltölteni a rendszerbe. Ezt különböző kategóriákba sorolhatják, mint pl. nadrág, pulóver stb. A kategória mellett megadhatják a ruhadarab stílusát, életkorát is. A megvásárolt termékekről fotó is mellékelhető a feltöltéshez. Az alkalmazás lehetővé teszi, hogy a ruhadarabokat a felhasználó böngéssze különböző szempontok szerint csoportosítva. Megtekintheti az azonos kategóriába vagy stílusba tartozó ruháit is. A program nyomon követi az egyes felhasználók ruháinak életkorát is. Ezek alapján ha egyes elemek elavultak, akkor értesítést küld, hogy ideje lenne új darabokat beszerezni. Az értesítéseket a felhasználó ízléséhez próbálja igazítani, hogy milyen stílusú ruhát vegyen. Az ajánlott ruhákat webshop linkek formájában juttatja el a felhasználóhoz. Ugyancsak értesítéseket küld a program, ha egy új „micro season” kezdődik. A felhasználó számára elérhető funkciókat az 1.ábrán lévő use-case diagram szemlélteti.

A *Ruhadarab feltöltése* eset jelenti azt, amikor a felhasználó megvesz egy új terméket és ezt feltölti az adatbázisába. Ehhez a funkcióhoz szorosan kapcsolódik a *Fotók készítése*, hiszen a feltöltés során a felhasználó képeket készíthet az új ruhájáról és ezeket is feltöltheti.

A *Ruhadarabok listázása* felel a feltöltött ruhák megjelenítéséért. Többféle szűrő állítható be, így szűrhetünk például stílusra, kategóriára egyaránt. A Ruhadarabok keresése segítségével az adatbázisunkban végezhetünk el kereséseket az ott szereplő ruhákra.

Az *Értesítések fogadása* usecase-ben valósul meg a rendszertől érkező értesítések fogadása. A rendszer értesítéseket küld ha új ruhadarabokat kellene beszerezni, mert a régiek elavultak, valamint ha egy új micro season kezdődött a divat világában.



ábra

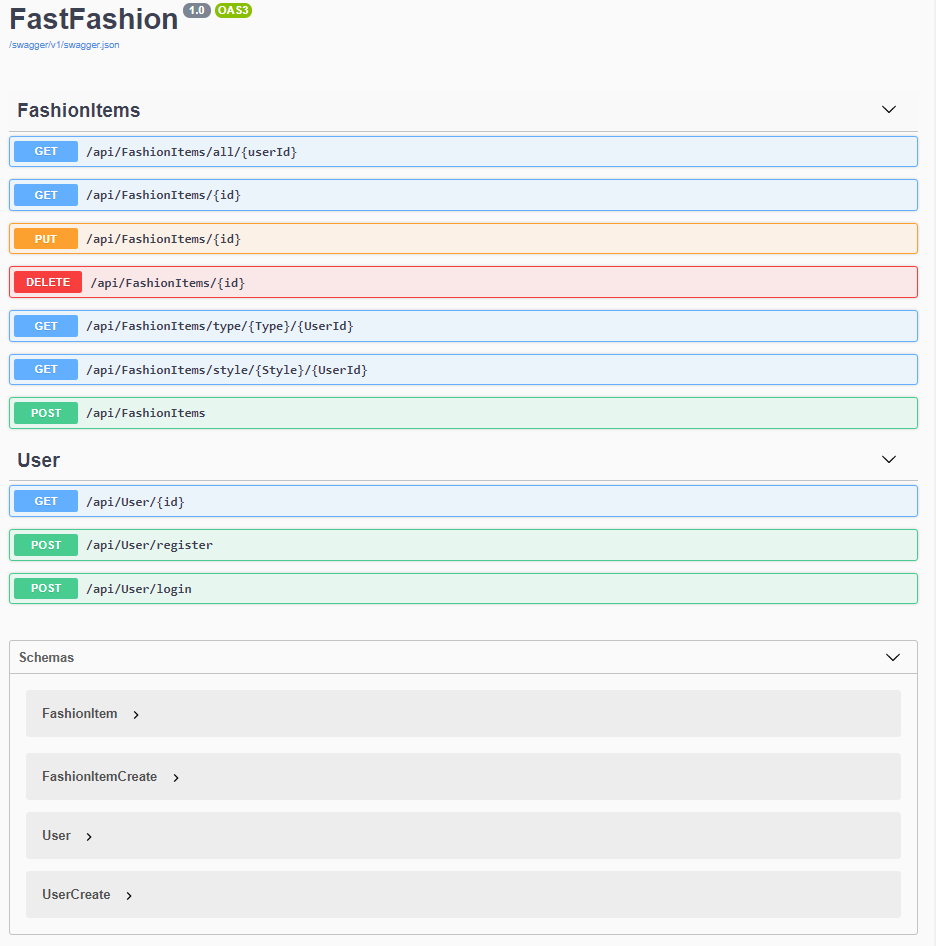
**Megvalósítás**

A rendszer egy kotlin nyelven írt android alapú frontendből, egy c#-ban írt asp.net alapú backendből és hozzá tartozó SQLite adatbázisból épül fel. Az adatbázisnak és a backend alkalmazásnak a Microsoft Azure nyújt platformot a hosztolásra, ezáltal az androidos alkalmazás mindig egyszerűen eltudja azt érni.

Az alkalmazás képes autentikálás után kezelni lokálisan a fényképeket, minden felhasználónak a saját készülékén tárolódnak a képek, ezekről egy URI-t tartunk nyílván az adatbázisban ami alapján tudunk a képekkel dolgozni. A felhasználó tud a ruhadarokbra stílus és típus szerint szűrni, és ha az egyes ruhadarabjai már elavultak, akkor az alkalmazás értesítést küld erről.

**Architektúra**

A FastFashion asszisztens egy háromrétegű alkalmazás, a REST(**RE**presentational **S**tate **T**ransfer) backenddel ami HTTP hívásokkal kommunikál a kliens Android alkalmazással, és ezeket támogatja egy adatbázis még. Az endpointokat a SwaggerUI alkalmazásból kinyert screenshottal mutatom be:



**Adatbázis réteg**

A SwaggerUI az alkalmazásban létező modellekről is készít dokumentációt, ezekből az adatbázisban a FashionItem és az User szerepel, a FashionItemCreate és UserCreate egyfajta DTO(DataTransferObjeckt)-ként működik, ezekkel próbáljuk a szenzitív adatokat rejtve tartani.

Az adatbázis réteg megvalósításához SQLite adatbáziskezelőt választottunk, ez azért volt kényelmes, mert .NET-ben egyszerű kezelni, könnyű migrációt megvalósítani a létező könyvtárakkal és az Azure rendszerben is lehetőségünk volt egy SQLite adatbázis hosztolására.

Az alkalmazás fejlesztése során code-first megvalósítással lettek az adatbázis objektumok létrehozva, ez azt jelenti hogy előszőr c# nyelven lettek definiálva, és abból a kódból egy migráció létrehozása után tudtuk az adatbázist elkészíteni vagy létrehozás után módosítani.

A kódban a FastFashion/models elérési út alatt találhatóak a leírások, az itt található FashionItemContext alapján történik a migrációs eljárás, aminek snapshotjait a migration mappa alatt tudjuk megtalálni.

**Adathozzáférési réteg:**

Célja adatok szolgáltatás az adatbázisból a felhasználói alkalmazásnak.

Feladati:

* + Új entitások létrehozása
  + Adathozzáférés biztosítása
  + Autentikáció kezelése

Ez a kódban a FastFashion/Controllers mappában található.

Itt a függvények felett kommentek és annotációk segítik az eligazodást az egyes metódusok között.

A FashionItem entitáshoz nyújtott lehetőségek:

* Egy felhashnálóhoz az összes ruhadarab lekérdezése.
* Ruhadarab saját id alapján való lekérdezése.
* Ruhadarab stílus/típus szerinti keresése.
* Új ruhadarab hozzáadása.
* Ruhadarab módosítása.

A felhasználók kezelése jelenleg az alábbi funkciókból áll:

* Felhasználó saját id alapján való lekérdezése.
* Felhasználó regisztrálása.
* Felhasználó ellenőrzése bejelentkezéskor.

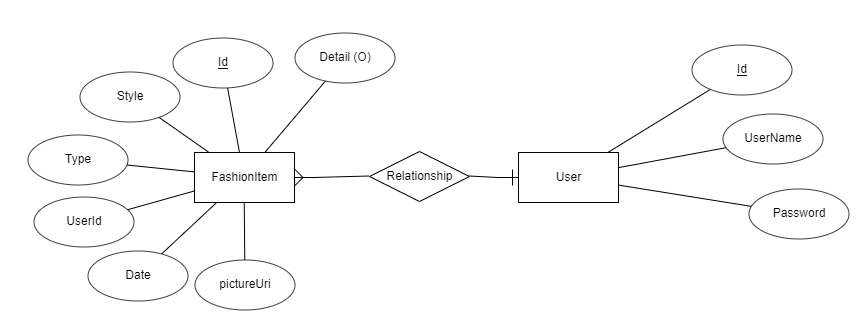
A réteg fő feladata a fenti funkciók megvalósítása a .NET-es DbContext segítségével. A DbContext lehetővé teszi, hogy egyszerűen tudjunk adatbázis műveleteket végrehajtani.

**Adat- és adatbázis terv**

Itt ismertetjük az egyes entitásokat és adatbázisra történő leképezésüket.

Két entitást terveztünk az alkalmazásba, a FashionItemet és a User-t. A user reprezentál egy felhasználót, aki azonosításra kerül belépéskor, a FashionItem pedig maga lesz egy ruhadarab. A két entitás 1:N kapcsolatban áll egymással, egy felhasználónak lehet több ruhadarabjai is.

Az adatbázis entitás-reláció diagrammja:



**C# kód és adatbázisbeli megfeleltetések**

Fashionitem: (Kódban FashionItem osztály, a FashionItemCreate egy DTO hozzá)

|  |  |  |
| --- | --- | --- |
| C# mezőnév | C# adattípus | SQLite adattípus |
| ID | Int | PK INT NOT NULL |
| Type | String | nvarchar not null |
| Detail | String | nvarchar |
| Style | String | nvarchar |
| Date | String | nvarchar |
| pictureUri | String | nvarchar |
| UserId | Int | INT NOT NULL |

* Az id mező tartalmazza az entitás egyedi kulcsát.
* A type mező az entitás típusát, pl. kabát, sál stb.
* A detail mező egy kötetlen leírás az entitáshoz, pl. „Ez a kedvenc kabátom”.
* A style mező a stílusát adja meg a ruhadaraboknak, pl. sálaknál lehet rolytos végű.
* A date mező a vásárlás dátumát tartalmazza.
* A pictureUri a készített képnek az elérési útvonalát tartalmazza.
* A UserId az a felhasználónak az egyedi azonosítója, akihez tartozik az adott FashionItem.

User:

|  |  |  |
| --- | --- | --- |
| C# mezőnév | C# adattípus | SQLite adattípus |
| UserId | Int | PK INT NOT NULL |
| Username | String | nvarchar not null |
| Password | String | nvarchar not null |

A UserId a felhasználók egyedi azonosítója.

A Username a felhasználóknak a szabadon választott egyedi neve.

A Password mező a felhasználónak a jelszava.

**Kliens oldali alkalmazás**

Az alkalmazás kliens oldalát egy android alkalmazásban valósítottuk meg, mely Kotlin nyelven készült el. Ennek a rétegnek a fő feladata, hogy a felhasználók ezen keresztül érhetik el a FastFashion szolgáltatásait. Az alkalmazás biztosítja a felhasználói felületet, mely egyszerűvé teszi a funkciók elérését. A megvalósítás több külső könyvtárat is használ. Az egyik ezek közül a Retrofit, mely a hálózati kommunikációért felelős. Az alkalmazás több olyan engedélyt is használ, melyeket futási időben kell elkérni, mert veszélyes engedélynek minősülnek. Ezek egyszerűbb kezelését a QuickPermissions könyvtár biztosítja.

A kódban található osztályok és ezek szerepe:

* Activity-k: Ezen osztályok valósítják meg és kezelik a felhasználói felületet.
  + DetailsActivity: Megjeleníti az egyes ruhadarabokhoz tartozó adatokat.
  + ListActivity: Egy listában jeleníti meg a felhasználó ruhadarabjait.
  + LoginActivity: A bejelentkezés és regisztrációs felület.
  + MainActivity: A főmenüt megvalósító osztály.
  + UploadActivity: Új ruhadarab feltöltését kezelő felület.
* adapter package:
  + ClothesAdapter: A ListActivity-ben megjelenő RecyclerView-t kezelő adapter. Ez tölti fel adatokkal a listát.
* fragments package:
  + DatePickerDialogFragment: Egy dátum kiválasztó dialógus ablakot valósít meg.
* model package: A hálózati kommunikációhoz szükséges JSON-öket tartalmazza data class osztályokként.
  + FashionItem
  + FashionItemCreate
  + IdResult
  + User
  + UserCreate
* network package: Hálózati kommunikáció megvalósítása. Az api interfészek definiálják az endpointokat, az interactorok pedig elvégzik a hívást egy háttérszálon. A backend mindkét controlleréhez tartozik egy-egy api és interactor.
  + AuthApi
  + AuthInteractor
  + FashionApi
  + FashionInteractor
* notification package: Az értesítések kiküldéséért felelős osztályok.
  + AlarmNotificationReceiver: A BroadcastReceiver, mely azért felelős, hogy minden nap a megadott időpontban kiváltsa a service értesítés küldését.
  + BootReceiver: BroadcastReceiver, mely figyeli a telefon újraindítását és elindítja a service-t ha bekövetkezik az esemény.
  + NewClothesService: Egy service, mely folyamatosan fut a háttérben és megadott időközönként értesítést küld.

**Grafikus felhasználói felület**

**Célja**: A felhasználók számára könnyen elérhetővé teszi az elkészített alkalmazás funkcióit.

Az alkalmazásunk az alábbi ikonra kattintva indítható el.



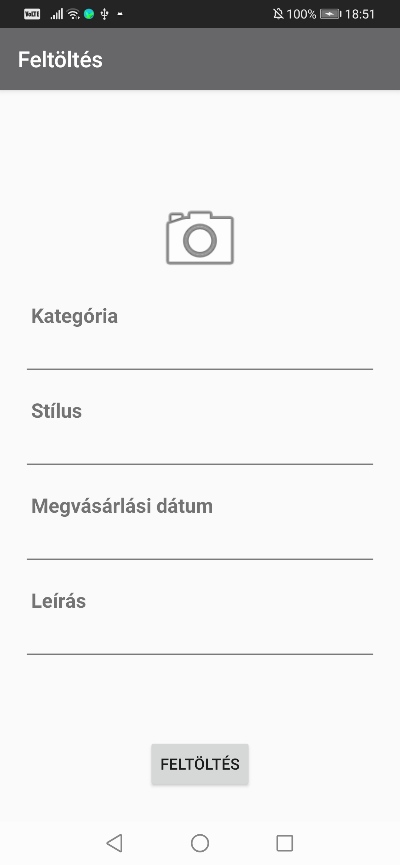
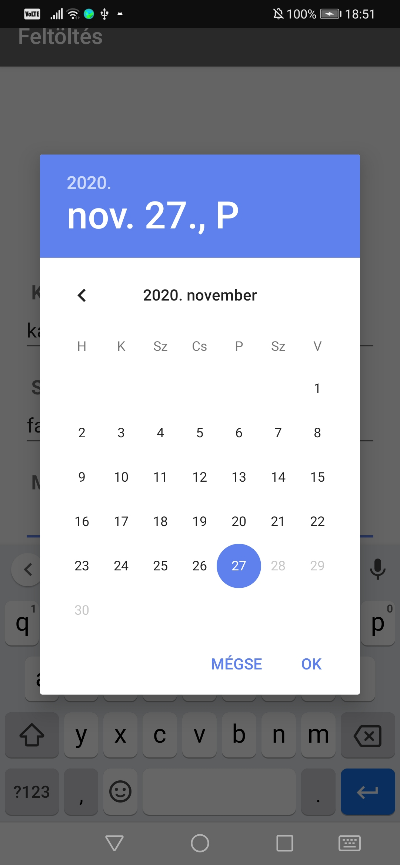
Az alkalmazás indulása után egy bejelentkezést és regisztrációt biztosító felület jelenik meg. Ha a felhasználó helytelen adatot ad meg, akkor az alkalmazás hibát jelez.

Sikeres bejelntkezés után egy főmenü jelenik meg, ahol 3 gomb látható. Lehetőség van kijelentkezésre, Új elem felvételére illetve a ruhadarabok listázására.

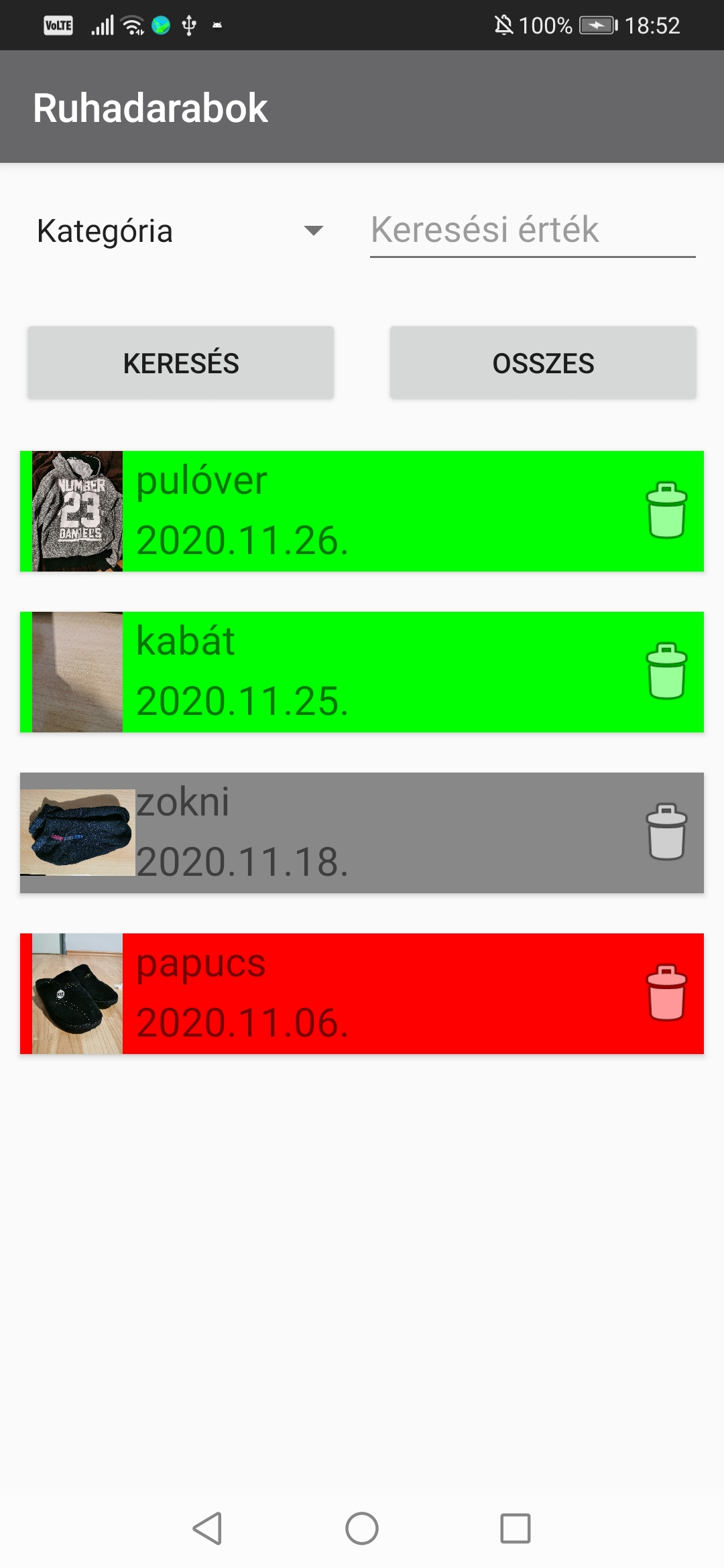


Az új elem gombra kattintva az alábbi felület jelenik meg. A fényképezőgép ikonra kattintva készíthetünk fényképet a ruhadarabról. A megvásárlás dátumát egy datum kiválasztó ablakban adhatjuk meg. Ha nem töltünk ki minden mezőt az alkalmazás hibát jelez. A feltöltés gombra kattintva tölthetjük fel az új ruhánkat az adatbázisba.

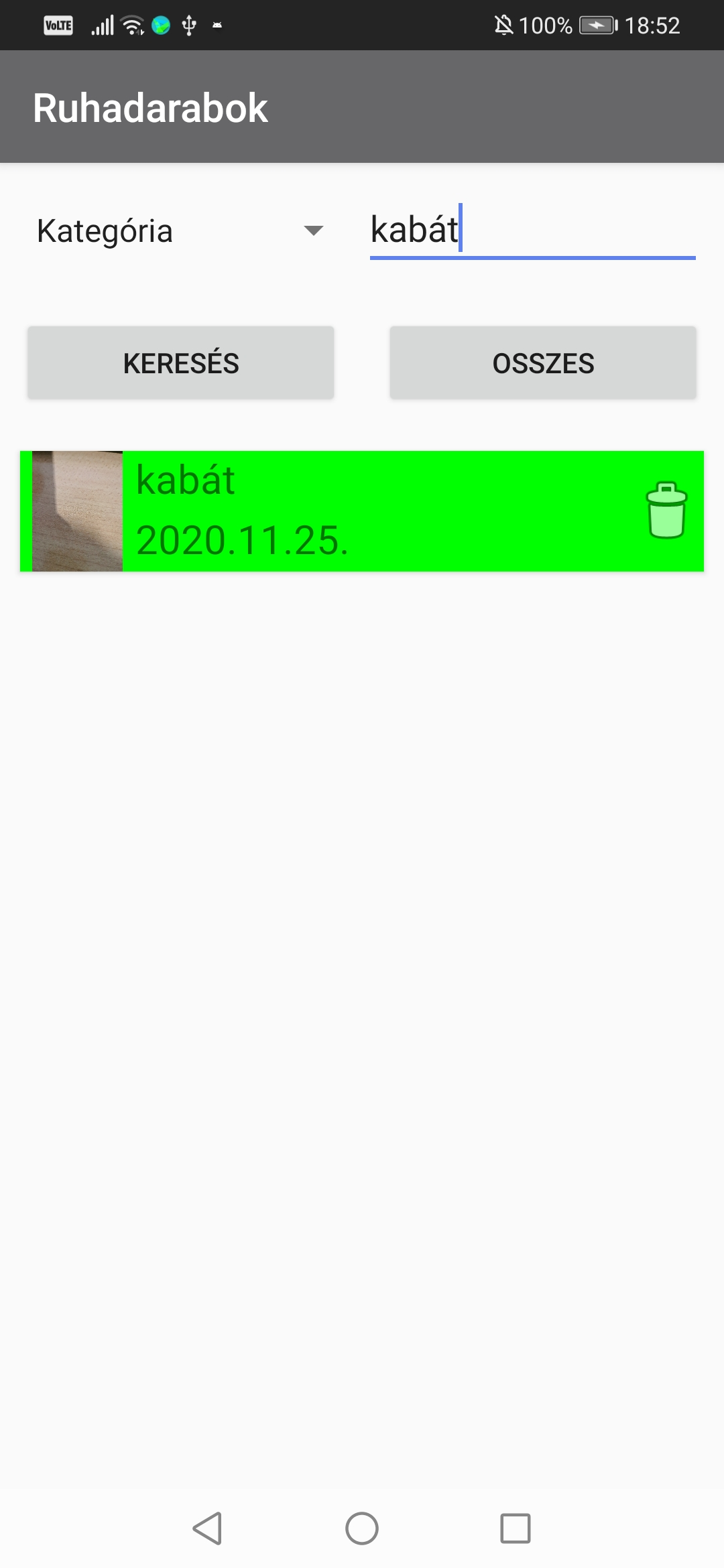
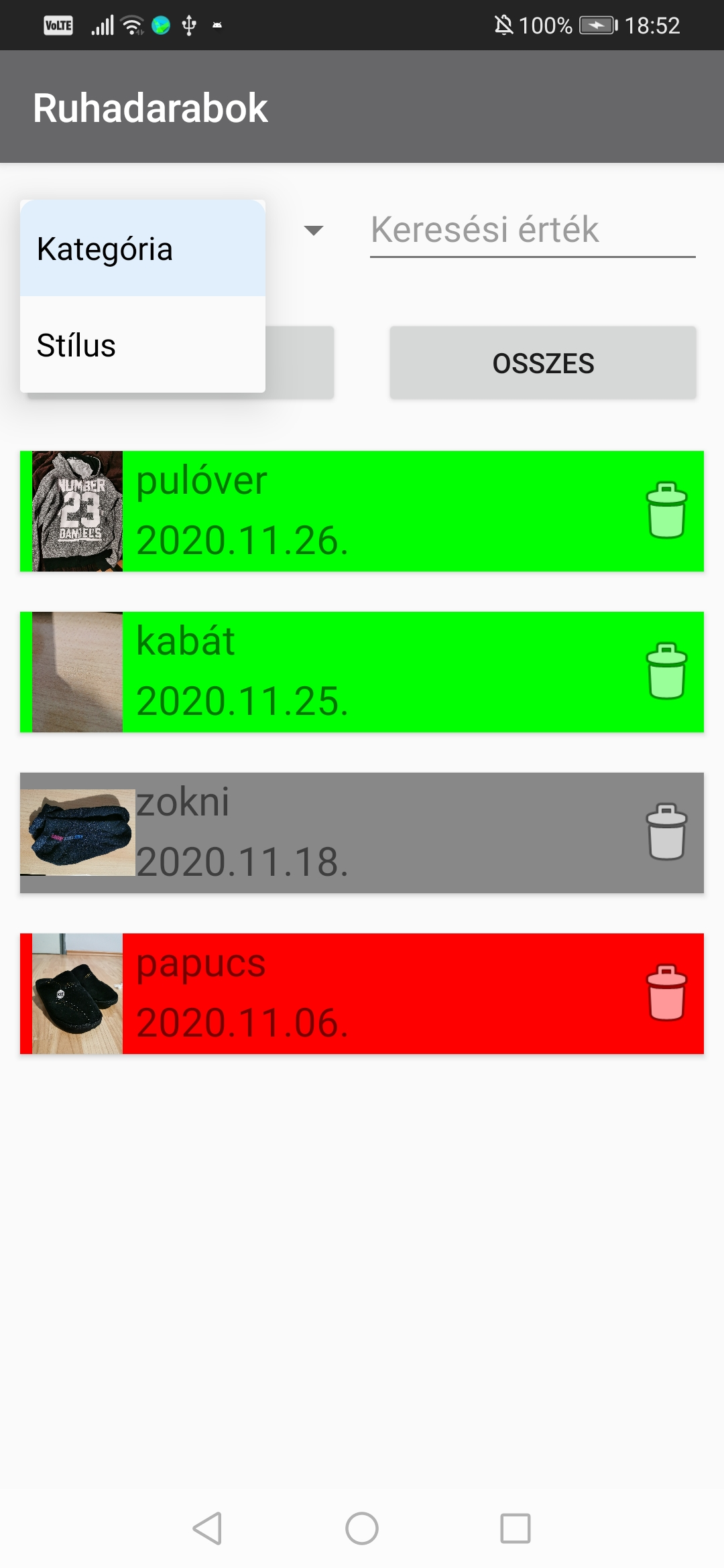
 



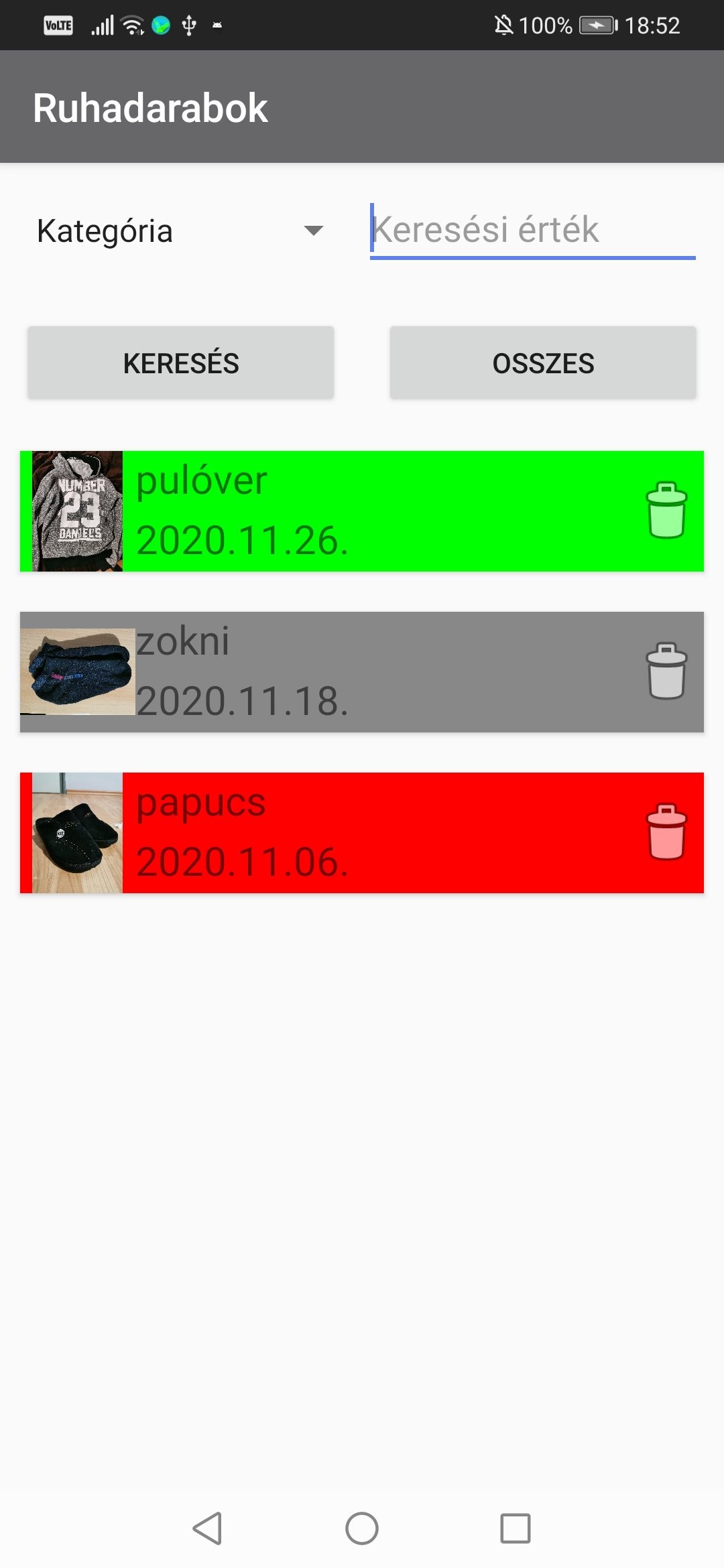
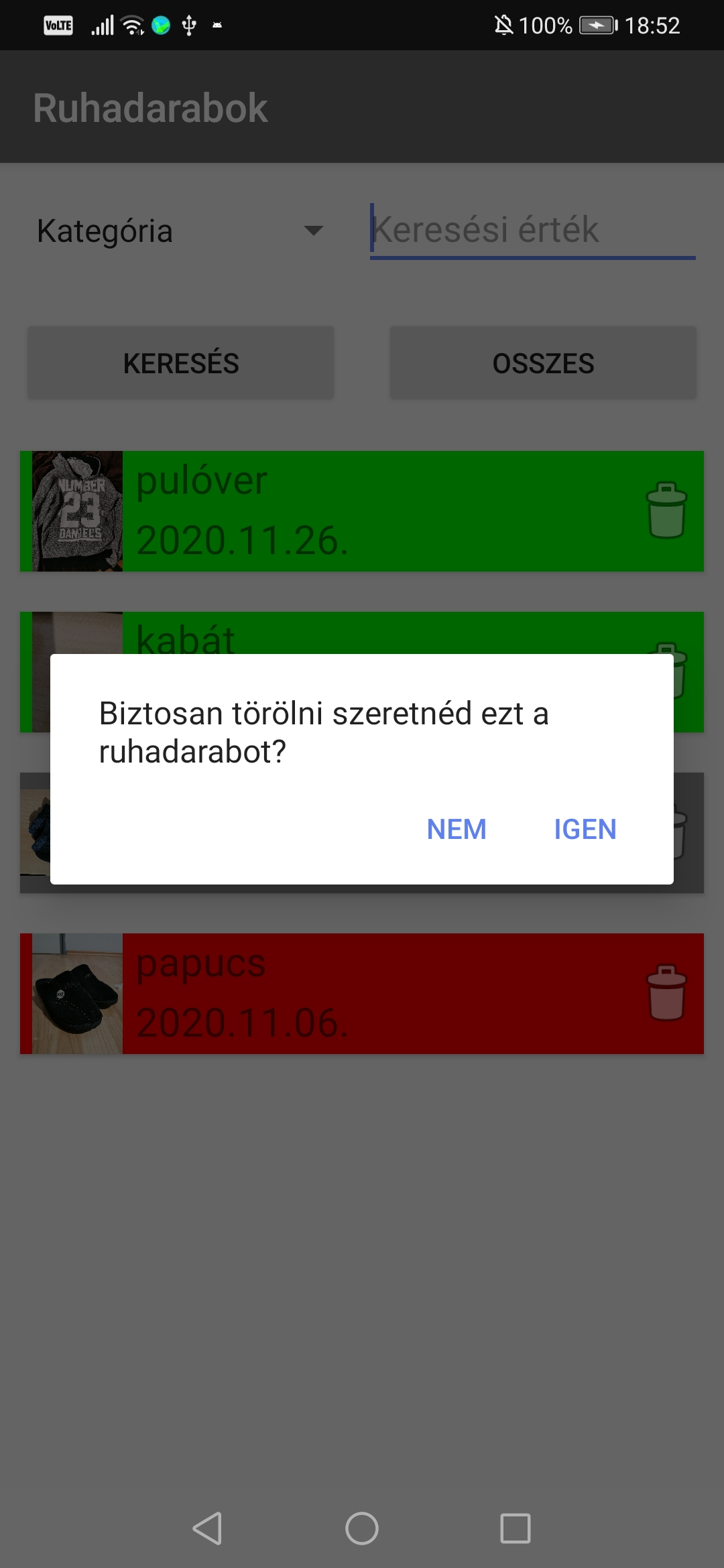
A főmenüből a ruhadarabok gombbal érhetjük el a ruháink listáját. Itt az egye elemek 3 különböző színnel jelennek meg. A 2 hétnél régebbi ruhák pirossal az 1 és 2 hét közöttiek szürkével, az 1 hétnél fiatalabbak zölddel. Ez vizuálisan is jelzi a felhasználó számára mennyire friss a ruhatára.



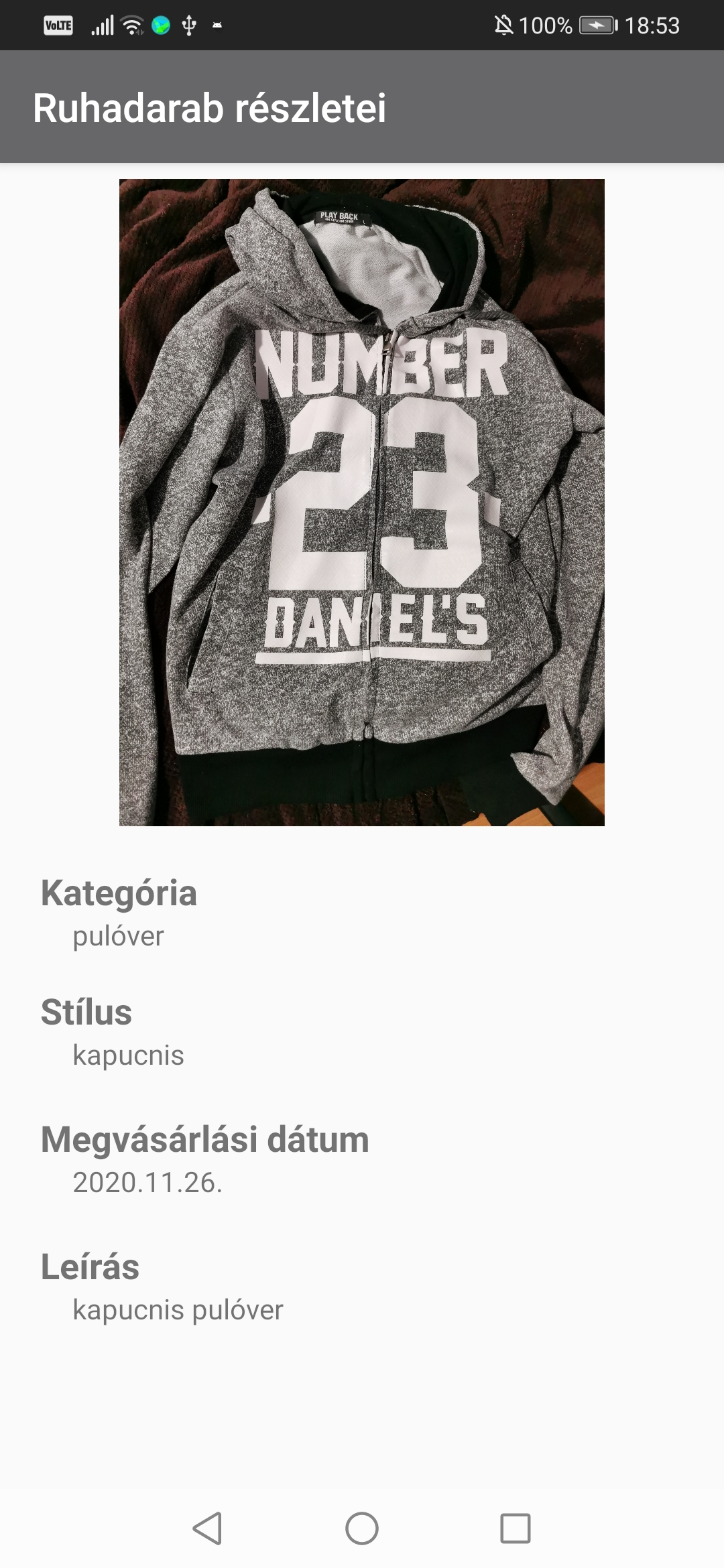
A listában két parameter szerinti keresésre van lehetőség. A ruhadarab kategóriája és stílusa szerint.



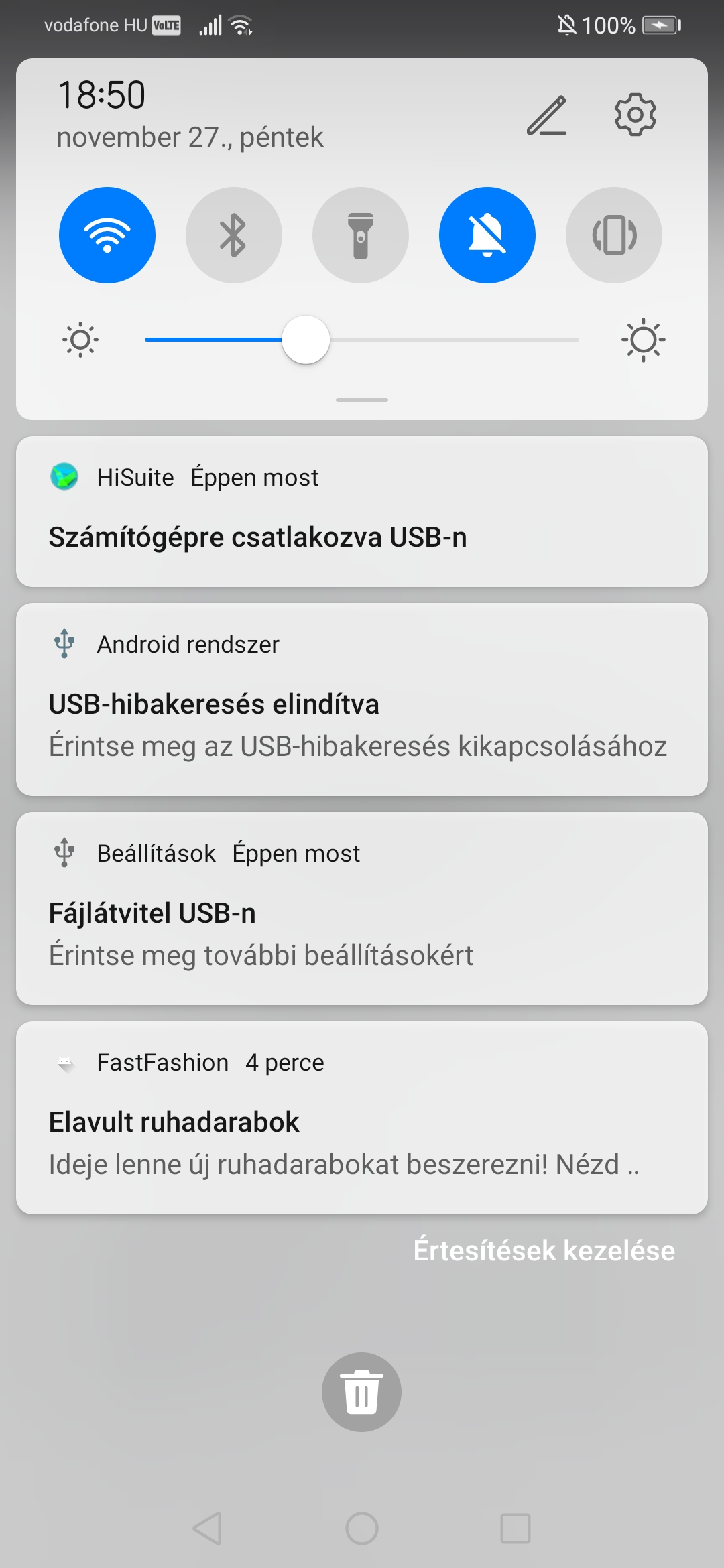
A listánkból lehetőség van törölni is. Ehhez a kuka elemre kell kattintani, majd ezután egy dialógus ablak jelenik meg, mely rákérdez, hogy biztosan szeretnénk-e törölni az elemet.



Ha a listában rákattintunk egy elemre, akkor megjelennek az adott ruhadarabhoz tartozó információk.



Az alkalmazás megadott időközönként értesítést küld, hogy felhívja arra a figyelmet, hogy szükség lenne új ruhadarab beszerzésére.



**Telepítési Útmutatás**

Az alkalmazásnak elég a frontend mobil alkalmazását feltelepíteni, mivel a backendet nyújtó RestApi és hozzátartozó adatbázis a Microsot Azure paltformján van hosztolva. Az android alkalmazáson keresztül minden funkciót el tudunk érni és tudunk kommunikálni az adatbázissal.

A telepítéshez szükségünk van egy android platformú okostelefonra, a használathoz internetelérés is szükséges.

Minimum 4-es verziójú android (15-ös SDK) szükséges az alkalmazás telepítéséhez.

Az alkalmazás egy apk segítségével telepíthető a telefonunkra. Az **APK** (application package file) egy archívum, amit az Android Csomagkezelő segítségével telepíthetünk.

Miután feltelepült az alkalmazás, regisztráció után elérhetjük az összes, alkalmazás által nyújtott funkciót.

**Összefoglalás**

Munkánk során megterveztük, implementáltuk és dokumentáltuk a FastFashion nevű alkalmazást. Az alkalmazás segítségével mindig frissen tarthatjuk a gardróbunkat.

A megvalósított alkalmazás 3 rétegű architektúrát használ: adatbázis réteg, egy adatelérési réteg és egy kliens alkalmazás. Az alkalmazás az adatokat egy felhőszálgatás általá nyújtott adatbázisban tárolja, szintén ugyanazon a platformon érhető el a háttéralkalmazás is. Az alkalmazás képes új ruhadarabokat felvenni az alkalmazásba, ezek között keresni stílus és típus alapján, illetve felhasználói authentikációval is rendelkezik.

Munkánk során a legnagyobb változás a tervezéshez képest az volt, hogy az alkalmazáshoz tartozó RESTapit és adatbázist egy felhőszolgáltatásba hosztoltuk, ez a lépés egyszerűbbé tette a kliens és backend alkalmazás feljesztését és a köztük lévő kommunkáció megoldását.

**Továbbfejlesztési lehetőségek**

Az alkalmazással kapcsolatban számos továbbfejlesztési lehetőség is látható.

Az alkalmazást jelenleg csak saját használatra terveztük, de pl. az Android store-ba történő alkalmazás feltöltés is több változtatást hozna magával.

Ezek közül a lehetőségek közül választunk ki néhányat:

* Több kép készítése az adott ruhadarabhoz.
* Biztonsági szempontból JWT használatának bevezetése, jelszó hashelése.
* Felhasználói adatok módosítása.
* Felhasználóbarátabb UI megvalósítása.
* Kutatás végzése a divatban, ENUM-ok létrehozása ruha típusokra, stílusokra az egységesebb kezelésért.
* Vásárlás helyének megadása egy Google térképen. Kedvenc üzletek kiválasztása.