JEGYZŐKÖNYV

Mobil Programozás alapjai

Féléves feladat/beadandó

Android applikáció

Készítette: **Simkó Levente**

Neptunkód: **YG5YAO**

Dátum: **2025. 01. 26.**

**Feladat:**

A feladat az, hogy egy meglévő android applikáción kell módosítást végezni, az adatbázis kezelőt kell alatta kicserélni, plusz adattagokat kell hozzáadni és módosítani kell a stílusát. A feladat elvégzéséhez úgy döntöttem, hogy 2 új adat kerül bevezetésre és minimális módosítást végzek azokon a dolgokon, ami stílusában nem tetszik a programnak, illetve megkülönböztetem a megvásárolt terméknek, így vizuálisan is visszacsatolást is növelem a programban.

**Megoldás:**

Mivel az adatbázis kezelő rendszer cseréje először bonyolultabbnak tűnt mint, gondoltam így kezdjük a stílus változtatásokkal. Az első észrevehető különbség, hogy a új termék felvitelét biztosító gomb kinézete megváltozott, rózsaszín háttérről és egy üzenet ikonról egy zöld háttérre és egy toll ikonra, ugyanis szerintem jobban kapcsolható így a hozzáadáshoz a zöld szín miatt és a toll pedig a tényleges papírra írt bevásárló lista írást szimbolizálja.

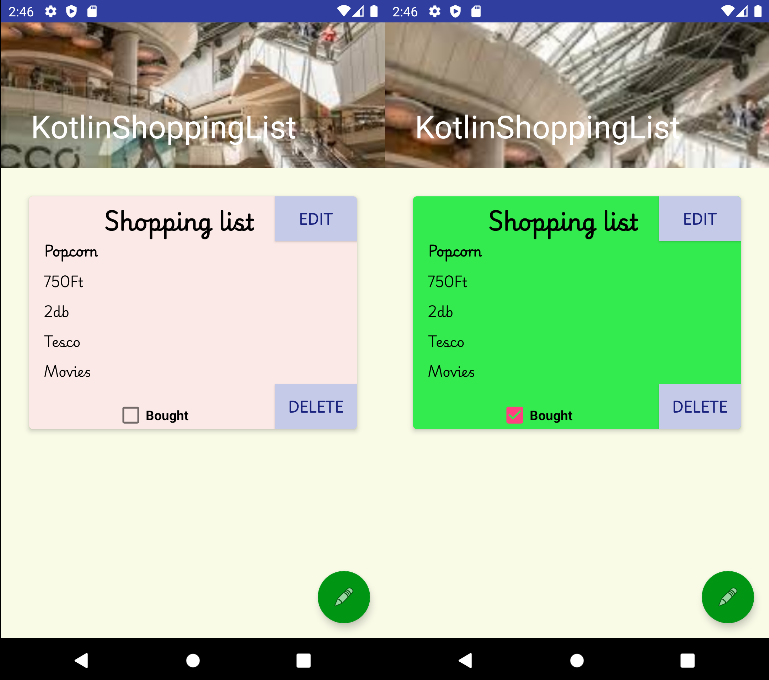
A képen szöveg, képernyőkép, szoftver, multimédia látható

Automatikusan generált leírás

Mára a fenti képen is látszik, hogy a „jegyzet” betűtípusa nem a normális betűtípus. Igyekeztem olyan betűtípust keresni, ami hasonlít a kézíráshoz, viszont ugye az változó kultúrák és emberek között is, így a tanított írott betűkhöz hasonlító betűtípust kerestem és így esett a választás a Playwrite England Joined betűtípusra esett a választás (viszont így a magyar betűkészlet ékezetes betűi nem támogatottak).

Ezért a betűtípus módosításért a row\_item.xml-ben a különböző komponensekben elhelyezett app:fontFamily="@font/playwritegbj\_variablefont\_wght" felel.

Másik dolog, ami észrevehető, hogy a termék megvásárolt állapotban van és ez azért fontos, mert attól függően a háttere megváltozik.



A fenti képen látszik a különbség a két mód között, így az is megfigyelhető, hogy el lehet különíteni a két terméket egymástól, meg akkor is, ha nem látjuk a megvásárlást jelző checkboxot mert több termék van egymás alatt így nem is kell mindet külön végi nézni elég csak háttér színt figyelni.

Annak érdekében, hogy ez megoldható legyen programkódi változtatással a háttérnek (esetünkben ez egy ConstraintLayout adunk egy Id-t majd a Shopping adapterben létrehozunk neki egy változót a ViewHolder osztályban constraintCL néven, majd a checkbox állapotát figyelő részen a változót felhasználva egy feltétel esetén beállítjuk a háttért megfelelő színre.

A képen szöveg, képernyőkép, Betűtípus látható

Automatikusan generált leírás

**Adatbázis módosítás:**

Annak érdekében, hogy 2 új adattagot hozzá tudjak adni, új változót kell hozzáadni és ShoppingItem konstruktorát is módosítani kell (ami annak érdekében lett megtartva, hogy minél kevesebb dolgot kelljen átírni a kódban) és bármelyik felületen, ahol ezek a új adattagok megjelennek be kell hozzájuk vezetni változót mint például megjelenítés, adatbetöltés, vagy beviteli mező.

A képen szöveg, képernyőkép, Betűtípus látható

Automatikusan generált leírás

Ha ez megvan és az összes megjelenítő és adat bekérő felületet ennek megfelelően módosítjuk majd tesztelésre felhasználjuk az eredeti adatbázis kezelőt utána akkor jöhet az adatbázis tényleges cseréje.

Mivel eredetileg kicsit félreértettem a feladatot (úgy kezdtem el, hogy mást tárol a program nem bevásárló listát így bizonyos funkciók és változók neve eltérhet) és egyszer sikerült annyira elrontani a kódot, hogy nem tudtam visszaállítani így 2 projektből csináltam 1-et. Az eredeti kódsorhoz írtam hozzá a másikból sikeresen kimentett és tesztelt kódot.

Az eredetileg Room-ot használó adatbázist átírtam SQLite-ra. Ezzel a DAO és az Adatbázis kezelő része kódnak feleslegessé vált, hiszen teljesen másik driver és módszer szükséges a csatlakozáshoz.

**DataBaseHandler:**

Ez a osztály fele az adatbázison keresztüli kommunikációért és az adatokon végrehajtott módosítások elvégzésért azaz a CRUD műveletekért. Ez leszármaztatja SQLite csatlakozásra használandó osztályt

A képen szöveg, képernyőkép, szoftver, Betűtípus látható

Automatikusan generált leírás

A onCreate() fele az adatbázis létrehozásáért ha még nem létezik, ha pedig „migrálni” szeretnénk az onUpgrade() függvény fut le ami a meglévő táblát eldobja (ergo adatvesztéssel jár) majd létrehozza a új táblát (fontos ilyenkor az adatbázis verziószámát növelni kell, ellenkező esetben hibát dob).

Ha ez megvan jöhet a tényleges crud műveletek, de ezekből csak pár kerül bemutatóra.

A képen szöveg, képernyőkép, Betűtípus látható

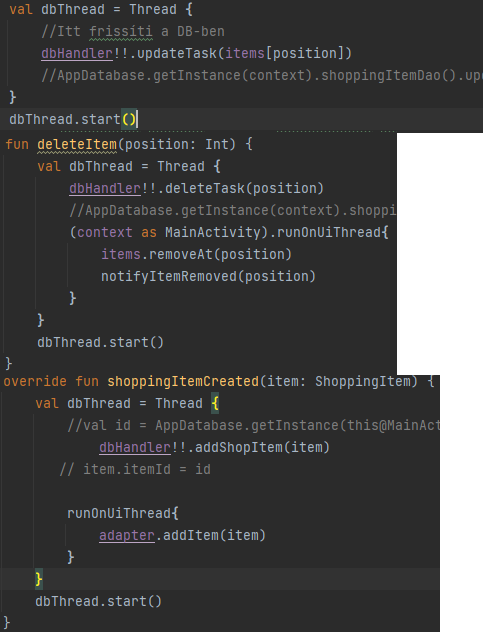
Automatikusan generált leírás

A képen szöveg, képernyőkép, szoftver látható

Automatikusan generált leírás

Látható a lekérdező függvény esetén, hogy rawQuery-t használ, azaz kb. tiszta SQL -ként írt parancsokat futtatnánk az adatbázison (a legtöbb esetben valami hasonlóra fordul a ténylegesen a háttérben a megírt kód, a Room esetén is megfigyelhető ez, csak az magának elvégzi ezek generálását), valamint, hogy szükséges egy kurzor, amivel nyomon követi az adattáblát és kimenti a számára fontos adatokat (amire a lekérdezés irányult).

Ezt követően az adatbázis többi függvénye hasonló mintát követ, majd áttérünk a tényleges részekre, ahol ezek a függvények meghívásra kerülnek. Mind a MainActivity-ben mind ShoppingAdapter-ben vannak erre példák, a legtöbb esetben külön thread részeként.



A dbHandler egy DataBaseHandler példánya, ami rendelkezik egy kontextussal, ami szükséges a kapcsolat felvételhez. Megfigyelhető, hogy „!!” -vel van ellátva, ami arra tesz biztosítást, hogy a megkapott adat nem null, szóval valami vissza fog térni az adatbázisból.

Egyes függvények a adatbázis kezelőben el vannak látva visszatérési típussal ami boolean annak érdekében, hogy azt fel lehessen használni, ha visszajelzést szeretnénk adni a felhasználónak viszont ennek használata nem kötelező.

A két projekt egybe mosása után a dokumentáció készítése közben tűnt fel, hogy a githubon fentlévő kódrészletben az egyik újonnan hozzáadott adattag egy az egyben megfelel az enyémnek a (quantity) és nem volt kedvem módosítani rajta akkor már így az ugyan úgy megmaradt, de bevezettem a Why azaz Miért-et hogy nyomon követhető legyen, miért akarjuk megvenni azt a terméket.