ANDROID PROGRAMIRANJE

Vežba 1: Kreiranje jednostavne *GroceryList* aplikacije

Ova vežba je inspirisana *IJ Apps* tutorijalom koji je dostupan na sledećem YouTube linku: https://www.youtube.com/watch?v=9nFGR8dIu_w

Takođe, ko želi da pogleda znatno ozbiljniji projekat na ovu temu, može da vidi i sledeću plejlistu: https://www.youtube.com/playlist?list=PLesEwfOYAU4YcaYveepDhq3wQI4iqLlPe
Ona sadrži niz tutorijala gde je opisano dosta stvari kojima ćemo se baviti u sklopu ovog kursa, tako da je definitivno preporučujem.

U nastavku slede objašnjenja ključnih komponenti koje ova aplikacija treba da ima, kao i osnovni vodič kroz Android Studio razvojno okruženje. Možete koristiti i Javu i Kotlin, nema ograničenja. Što se dizajna aplikacije tiče, imate slobodu u smislu boja, fontova, slika, buttona, polja...Bitno je samo da bude uredno i pregledno, dakle *user-friendly,* i da zadovoljava traženu funkcionalnost.

Inicijalizacija

- Otvorite Android Studio i izaberite "New Project".
- Odaberite "Empty Activity" kao tip projekta.
- Dodelite ime projekta **GroceryList**.
- Postavite minimalnu verziju SDK-a (npr. API 21).

Grafičke komponente

- Pri pokretanju aplikacije, ideja je da se na ekranu pojave polja za dodavanje nove stavke, zajedno sa “Add” i “Reset” buttonima.
- Svaka stavka ima svoj naziv, željenu količinu i cenu po komadu. Ovo možete realizovati sa tri tekstualna polja koja su praćena odgovarajućim labelama.
- Na samom dnu ekrana bi trebalo da se nađu opcije za *Kraj dodavanja* i *Novo dodavanje*. Kraj dodavanja bi inicirao prikupljanje svih unetih elemenata i izračunavanje ukupne cene željenih proizvoda. Ovaj iznos može biti posebno naznačen i ispisan bilo gde na ekranu. Novo dodavanje bi oznavačavalo poništenje svih unetih elemenata i početak nove liste.
- Takođe, klikom na svaki element liste treba omogućiti prikaz ikonice za njegovo brisanje. Više o ovome možete pročitati u odeljku za funkcionalnost.
- Ukoliko još uvek nema elemenata, treba onemogućiti kraj dodavanja ili novo dodavanje i ispisati adekvatne poruke (npr “Molimo dodajte bar jedan proizvod u listu”). Takođe, onemogućiti dodavanje nove stavke u listu ukoliko nisu sva tri polja pravilno popunjena.

Funkcionalnost

- Preporuka je da se kreira klasa Proizvod, koja će imati polja za naziv, količinu i cenu po komadu. Na taj način ćemo lakše kreirati listu elemenata, imaćemo ArrayList<Proizvod> u samoj aplikaciji. Naravno, ko želi može da koristi drugačiji pristup i podeli svoje ideje.
- Add će dodavati novu stavku u listu i prikazivati je na ekranu. Može prikazati samo njen naziv, a može biti i neki String format, kao recimo: *Sladoled Kapri, 3 kom, 150 rsd/kom*.
- Reset će brisati sve do tada unete vrednosti iz polja a lista već unetih podataka treba da ostane nepromenjena. Reset će se odnositi samo na trenutnu stavku.
- Kao što je već napomenuto, klikom na svaki element treba omogućiti njegovo brisanje. Jedna ideja je ovo realizovati preko ikonica (kantica za otpatke) koja bi se pojavile sa desne strane kada kliknemo na element iz liste. Besplatne ikonice možete pronaći na sajtu flaticon.com a u projektu ih treba smestiti u okviru **res/drawable** foldera. Druga ideja se može realizovati kada držimo duži klik na element liste, pa onda dobijemo pop-up poruku (**Alert dialog*)** da li zaista želimo da obrišemo element i shodno tome nastavimo dalje. Izuzetno je bitno da se lista osveži nakon brisanja, kako se uklonjeni elementi ne bi videli.
- Kada se odabere opcija za kraj unosa, treba onemogućiti dodavanje novih elemenata ili brisanje trenutnih (možemo da sakrijemo ili uradimo disable za ta polja), na ekranu treba biti prikazana konačna cena u dinarima a korisniku treba biti dozvoljeno samo da pokrene novu shopping listu, bez mogućnosti da bilo šta dodatno radi sa postojećom.

Testiranje

- Proverite da li su svi zavisnosti pravilno instalirane i da nema grešaka u kodu koristeći Code Analysis (Code > Inspect Code).
- Pokrenite aplikaciju u emulatoru koristeći dugme Run (izgleda kao play dugme).
- Testirajte sve funkcionalnosti aplikacije, uključujući dodavanje proizvoda, resetovanje unosa, brisanje proizvoda, izračunavanje ukupnih troškova i pokretanje nove liste.
- U slučaju grešaka, koristite **Logcat*** za pregled logova i poruka greške.
- Proverite aplikaciju na različitim emulatorima ili fizičkim uređajima sa raznim verzijama Androida kako biste osigurali široku kompatibilnost.

***Logcat** u Android Studiju je alat za debagovanje koji prikazuje log poruke s vašeg uređaja ili emulatora. Otvorite ga putem View > Tool Windows > Logcat. Možete filtrirati poruke prema tipu (Debug, Info, Warning, Error) i TAG-u koji ste definisali. Log poruke se generišu pomoću metoda:

```
Log.d(TAG, "Aplikacija je pokrenuta."); // Debug poruka
Log.i(TAG, "Informacije o inicijalizaciji."); // Info poruka

try {
    int result = divideNumbers(10, 0); // Deljenje sa nulom
} catch (ArithmeticException e) {
    Log.e(TAG, "Greška pri deljenju: " + e.getMessage()); // Error poruka
}
```

Klikom na određene log poruke možete dobiti više informacija o grešci ili obaveštenju. Možete i obrisati stare logove koristeći Clear Logcat dugme, snimiti log poruke u fajl, ili koristiti pretraživač unutar Logcat-a za brzo pronalaženje specifičnih informacija.

***Alert dialog** je pop-up prozor koji pruža korisnicima važne informacije ili traži da donesu neku odluku. Obično se koristi za prikazivanje upozorenja, potvrda, ili poruke o greškama. Može imati više dugmadi, kao što su "Da", "Ne", "OK", ili "Cancel", i omogućava korisnicima da reaguju na prikazanu situaciju. Što se osnovnih elemenata dijaloga tiče, možemo izdvojiti sledeće:

- AlertDialog.Builder se koristi za kreiranje dijaloga
- setTitle() postavlja naslov dijaloga
- setMessage() postavlja poruku dijaloga
- setPositiveButton() i setNegativeButton() postavljaju dugmad sa određenim radnjama
- show() prikazuje dijalog korisniku

Inače, u Androidu, pored Alert Dialoga, postoje tipovi i kao što su DatePicker dijalog, TimePicker dijalog, Progress dijalog, ali i Custom dijalog, koji služi za kreiranje personalizovanih dijaloga.

Evo jednog primera dijaloga za brisanje elementa (sa dugmadima Da/Ne):

```
Button deleteButton = findViewById(R.id.deleteButton);

deleteButton.setOnClickListener(new View.OnClickListener() {

    @Override

    public void onClick(View v) {

        new AlertDialog.Builder(MainActivity.this)

            .setTitle("Brisanje")

            .setMessage("Da li ste sigurni da želite da obrišete ovaj element?")

            .setPositiveButton("Da", new DialogInterface.OnClickListener() {

                @Override

                public void onClick(DialogInterface dialog, int which) {

                    // … Logika za brisanje elementa …

                }})

            .setNegativeButton("Ne", null)

            .show();

    }

});
```

Sledi primer kako bi izgledalo da rezultat (ukupnu vrednost svih proizvoda sa šoping liste) želimo da prikažemo preko dijaloga koji ima samo dugme OK.

```
Button totalButton = findViewById(R.id.totalButton);

totalButton.setOnClickListener(new View.OnClickListener() {

    @Override

    public void onClick(View v) {

        new AlertDialog.Builder(MainActivity.this)

            .setTitle("Ukupna vrednost")

            .setMessage("Ukupna vrednost proizvoda na listi je: " + ukupno + " dinara.")

            .setPositiveButton("OK", null)

            .show();

    }});
```