Primjena Material Design standarda u izradi korisničkog sučelja

Lenart, Petar

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Sveučilišni studij

PRIMJENA MATERIAL DESIGN STANDARDA U IZRADI KORISNIČKOG SUČELJA

Završni rad

Petar Lenart

Osijek, 2021.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

Obrazac Z1P - Obrazac za ocjenu završnog rada na preddiplomskom sveučilišnom studiju

Osijek, 07.07.2021.

Odboru za završne i diplomske ispite

Prijedlog ocjene završnog rada na preddiplomskom sveučilišnom studiju

Ime i prezime studenta:	Petar Lenart
Studij, smjer:	Preddiplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	R4233, 26.07.2018.
OIB studenta:	05703747607
Mentor:	Izv. prof. dr. sc. Časlav Livada
Sumentor:	
Sumentor iz tvrtke:	
Naslov završnog rada:	Primjena Material Design standarda u izradi korisničkog sučelja
Znanstvena grana rada:	**Obradba informacija (zn. polje računarstvo)**
Predložena ocjena završnog rada:	Izvrstan (5)
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: 3 bod/boda Postignuti rezultati u odnosu na složenost zadatka: 2 bod/boda Jasnoća pismenog izražavanja: 3 bod/boda Razina samostalnosti: 3 razina
Datum prijedloga ocjene mentora:	07.07.2021.
Datum potvrde ocjene Odbora:	

Potpis mentora za predaju konačne verzije rada u Studentsku službu pri završetku studija:	Potpis:
	Datum:

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA
I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

IZJAVA O ORIGINALNOSTI RADA

Osijek, 23.08.2021.

Ime i prezime studenta:	Petar Lenart
Studij:	Preddiplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	R4233, 26.07.2018.
Turnitin podudaranje [%]:	3

Ovom izjavom izjavljujem da je rad pod nazivom**: Primjena Material Design standarda u izradi korisničkog sučelja**

izrađen pod vodstvom mentora Izv. prof. dr. sc. Časlav Livada

i sumentora

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis studenta:

SADRŽAJ

1. UVOD 1

1.1. Zadatak završnog rada 1

2. MATERIAL DESIGN 2

2.1. Okruženje 2

2.2. Raspored 9

2.3. Navigacija 10

2.3.1. Bočna navigacija 10

2.3.2. Navigacija prema naprijed 11

2.3.3. Navigacija prema natrag 12

2.4. Boje 13

2.5. Tipografija 15

2.6. Ikonografija 17

3. KOMPONENTE 19

3.1. Gornja aplikacijska traka 19

3.2. Gumbovi 22

3.3. Plutajući akcijski gumb 27

3.4. Kartica 30

3.5. Birač datuma 33

3.6. Dijaloški okvir 35

3.7. Izbornik 38

3.8. Navigacijska ladica 40

3.9. Pokazatelj napretka 44

3.10. Snackbar 47

3.11. Tekstualno polje 49

4. ZAKLJUČAK 51

LITERATURA 52

SAŽETAK 54

ABSTRACT 55

ŽIVOTOPIS 56

PRILOZI 57

1. UVOD

Brzim razvojem mobilne tehnologije u posljednjem desetljeću pojavio se problem nedosljednosti dizajna korisničkog sučelja. Različiti operacijski sustavi koristili su različite koncepte. Neki od njih koristili su već postojeće elemente iz prijašnjih sustava, dok su drugi pokušavali osmisliti elemente jedinstvene za njihovu platformu. Sve to je dovodilo do izrazite razlike, kako u načinu, tako i iskustvu korištenja uređaja. Korisnik se nije mogao jednostavno prebaciti s korištenja jedne platforme na drugu, pri tome znajući način njenog funkcioniranja, nego je morao poznavati više sustava. Taj je problem pokušao riješiti Google predstavljanjem i integracijom Material Design sustava u vlastiti Android operacijski sustav. Material Design sastoji se od smjernica i komponenti pomoću kojih se mogu izraditi konzistentne, responzivne aplikacije neovisne o platformi.

Tema ovog završnog rada izrada je Android aplikacije slijedeći smjernice navedenog dizajn sustava. Aplikacija se zove OrganizeMe i njome se korisnik može poslužiti za organizaciju slobodnih ili poslovnih aktivnosti te na njima surađivati s drugim korisnicima. Aplikacija ima mogućnost registriranja i prijave te uređivanja svog profila. Korisnik može kreirati ploču (engl. *Board*) u kojoj se nalazi popis zadataka (radnji). Unutar njih postoje kartice s pojedinim zadatcima kojima se može mijenjati redoslijed te na koje korisnik može kliknuti i postaviti detalje poput boje kartice, osoba koje sudjeluju, datuma i slično. Aplikacija je napisana u programskom jeziku Kotlin, u razvojnom okruženju Android studio. Za bazu podataka korištena je Google-ova Firebase platforma.

1.1. Zadatak završnog rada

Zadatak završnog rada je opis smjernica Material Design-a te izrada funkcionalne Android aplikacije prema navedenim smjernicama. Primjer i primjena odabranih dostupnih komponenti navedenog dizajn sustava te opis njihove implementacije.

2. MATERIAL DESIGN

Material Design je sustav kojeg je osmislio Google s ciljem stvaranja dosljednog dizajn jezika s responzivnim elementima koji se s lakoćom može prilagoditi za različite platforme na uređajima različitih zaslona. Predstavljan je na godišnjoj Google I/O developerskoj konferenciji 2014. godine kao dio „Android L Developer Preview“ programa. Na navedenoj konferenciji 2021. godine predstavljena je iduća generacija Material Design-a naziva „MaterialYou“ koji donosi evoluciju samoga sustava, no za vrijeme pisanja završnog rada nisu objavljeni detalji.

Material Design nadahnut je stvarnim svijetom proučavanjem interakcije tinte i papira. Temelji se na metaforičkom materijalu od kojeg su svi elementi korisničkog sučelja sagrađeni. Površine i rubovi istog pružaju vizualne znakove za interakciju utemeljene u stvarnosti koji se ponašaju u skladu s pravilima fizike. Fleksibilnost materijala stvara nove mogućnosti koje nadmašuju one stvarnog svijeta bez kršenja pravila fizike.

Temeljni elementi poput tipografije, rešetki, prostora, boja, mjerila i upotreba slika ne služe samo kako bi bili oku ugodni već čine mnogo više. Oni stvaraju hijerarhiju i značenje te ostvaruju glavni vizualni dojam. Pažljivi odabir boja, slika, fontova te namjerno ostavljanje praznog prostora stvara sučelje koje uranja korisnika u iskustvo.

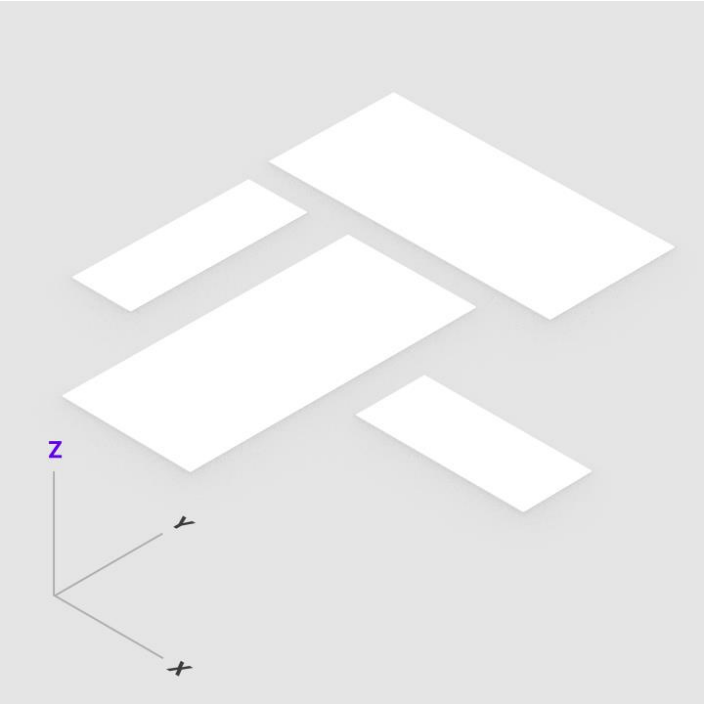
Kretnje (animacije) naglašavaju korisniku kako je on glavni pokretač radnji. On ih pokreće transformirajući cjelokupni dizajn. Objekti se korisniku predstavljaju bez narušavanja iskustva korištenja čak i dok se transformiraju ili reorganiziraju. Kretnje moraju biti smislene i prikladne, a služe usmjeravanju pozornosti. Povratne informacije moraju biti suptilne, a jasne.

2.1. Okruženje

U stvarnome svijetu objekti mogu biti u međusobnom odnosu. Mogu se nalaziti jedan na drugome ili jedan pokraj drugoga, ali dva objekta ne mogu u isto vrijeme zauzeti isti prostor. Ova se svojstva odražavaju i u Material Design-u. Površine i njihove kretnje nalikuju na onima u stvarnom svijetu.

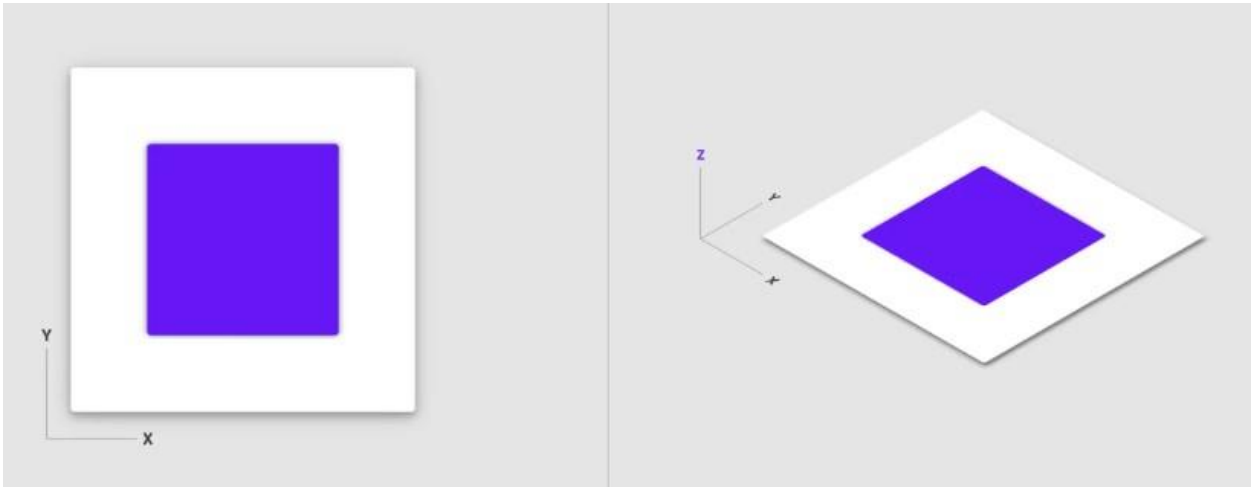
Korisničko sučelje koje primjenjuje Material Design standard prikazuje se u okruženju koje izražava trodimenzionalni prostor pomoću svjetla, površina i bačenih sjena. Elementi se mogu pomicati vodoravno, okomito te na različitim dubinama po z-osi.

Element može imati promjenjivu dužinu i širinu, ali svaki element mora imati debljinu koja iznosi točno 1 dp, gdje dp predstavlja gustoću piksela (engl. *density pixel*) koja govori koliko se piksela po inču nalazi na zaslonu uređaja. Sve navedeno može se uočiti na slici 2.1.

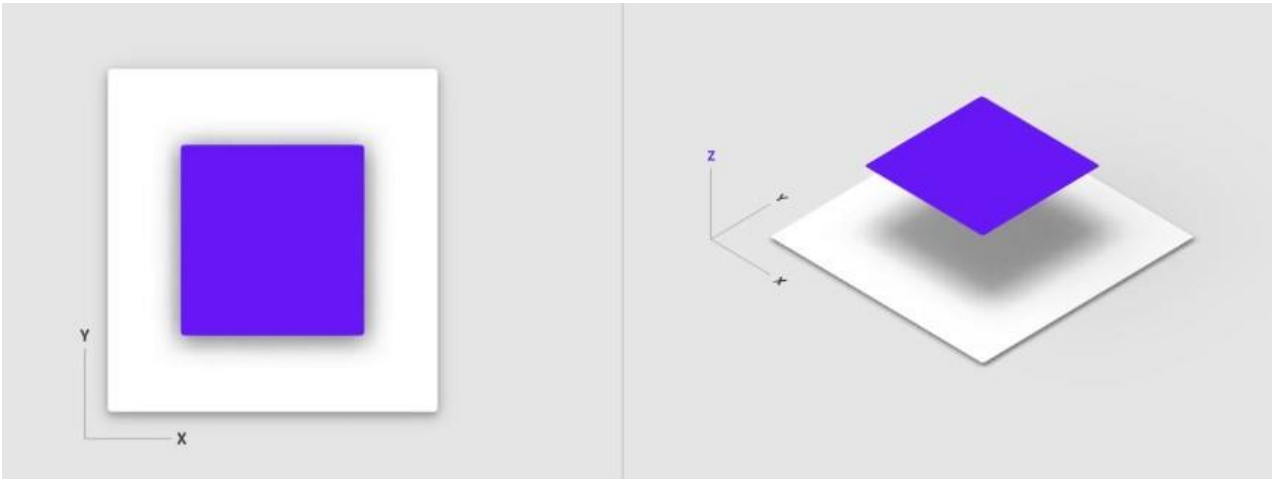


Sl. 2.1. Pravilne dimenzije elemenata Material Design-a

Materijal, točnije, elementi izrađeni od njega, na različitim uzvišenjima bacaju drugačiju sjenu koja izražava razinu uzvišenja. Na slikama 2.2. i 2.3. mogu se vidjeti pogledi odozgo te pogled iz tri dimenzije koji prikazuju način na koji bi objekti trebali bacati sjenu u ovisnosti o razini njihovog uzvišenja.

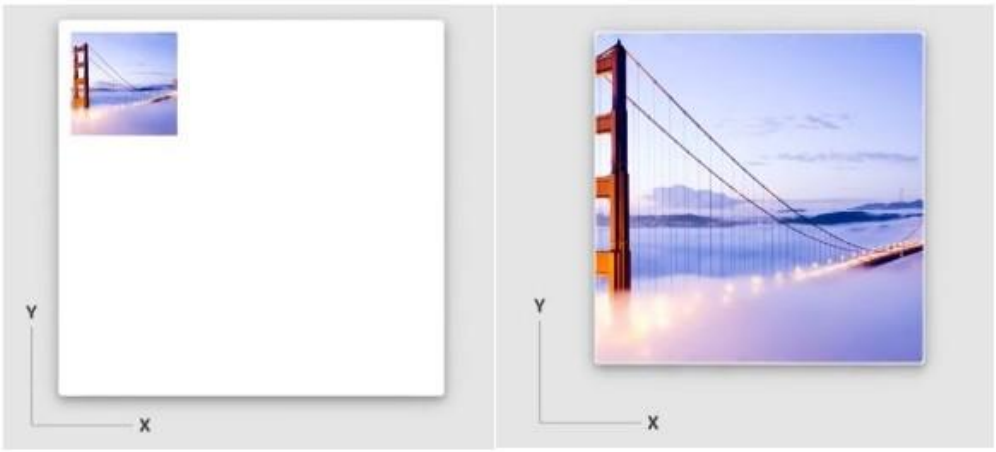


Sl. 2.2. Pravilno bacanje sjena elementa – bez uzvišenja

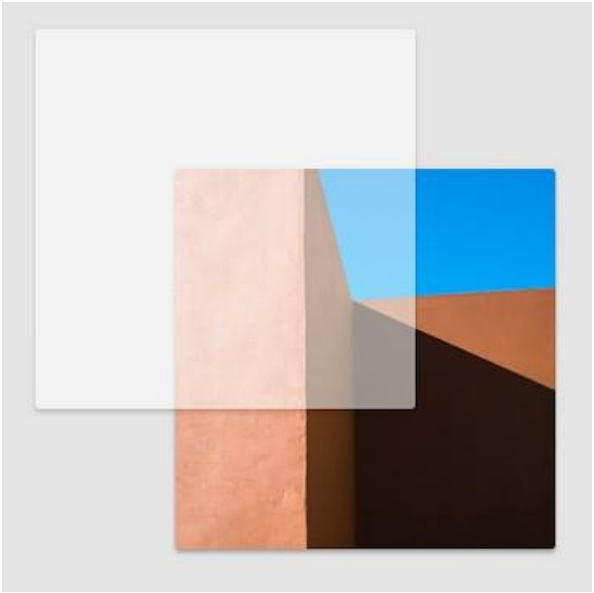


Sl. 2.3. Pravilno bacanje sjena elementa – izraženo uzvišenje

Sadržaj se može prikazati na elementu u bilo kojem obliku ili boji te se mora izraziti kao jedinstveni sloj. Može se ponašati neovisno ili ovisno o elementu, ali mora biti unutar njegovih granica. Naravno, prilikom prikaza sadržaja, element može mijenjati prozirnost bilo preko cijele svoje površine ili samo njezinog dijela. Sve navedeno prikazano je na slikama 2.4. i 2.5.



Sl. 2.4. Ponašanje sadržaja unutar elementa



Sl. 2.5. Prozirnost elementa pri prikazu sadržaja

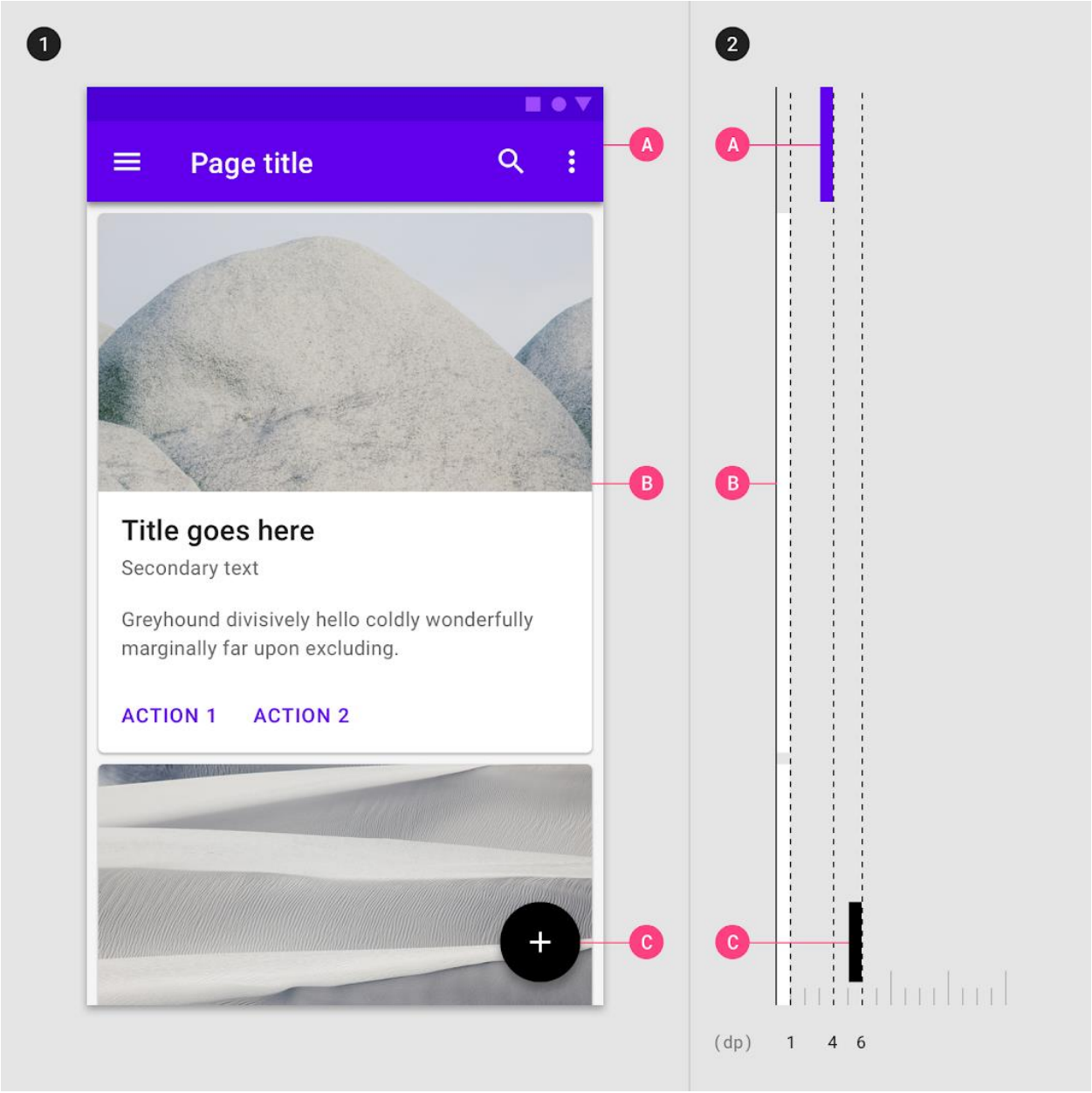
Površine elemenata Material Design-a mogu se ponašati na jedan od navedenih načina:

- Ponašanje poput čvrste površine koja ostaje nepromijenjenih dimenzija za vrijeme interakcija s korisnikom.
- Ponašanje poput rastezljive površine koja se duž jedne dimenzije (širine ili dužine) može proširiti ili smanjiti za vrijeme interakcije s korisnikom (kako bi se prikazao sadržaj) sve do određene granice, kada se počne ponašati poput čvrste površine.
- Ponašanje poput površine kojom se sadržaj može pomicati bez mijenjanja njezinih dimenzija, sve dok ne dođe do kraja kada se površina u tom smjeru počinje ponašati poput čvrste.

Sl. 2.6. Ponašanje površine poput čvrste površine

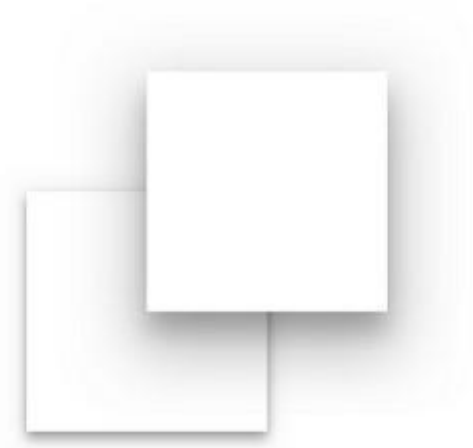
Uzvišenje kao dio Material Design-a određuje se kao udaljenost na z-osi jedne površine materijala od druge, a izražava se pomoću sjena. Sve Material Design površine i komponente imaju određene vrijednosti uzvišenja, a ono pruža brojne pogodnosti. Na primjer, uzvišenje površinama omogućuje pomicanje ispred ili iza drugih površina. Također, uzvišenje odražava prostorne odnose, poput plutajućeg akcijskog gumba koji nam pomoću vlastite sjene daje do znanja da je odvojen od druge površine. Pozornost pridajemo najvišem uzvišenju, poput dijaloškog okvira koji se privremeno pojavljuje ispred svih ostalih površina kako bi nam ukazao na potrebnu akciju.

Komponente imaju zadanu vrijednost uzvišenja, a ona se mijenja kao odgovor na interakciju s korisnikom ili događaj u aplikaciji. Svaka komponenta ima jednaku vrijednost uzvišenja. Na primjer, sve korištene kartice imaju jednako uzvišenje kao druge kartice. Zadana vrijednost razlikuje se od platforme do platforme, pa čak ovisi i o aplikaciji. Na mobilnim uređajima one pomoću sjena ukazuju na moguću interakciju s komponentom, dok, primjerice, na desktop uređajima zadane vrijednosti su manje zbog samog broja ostalih elemenata koji mogu ukazati na navedenu interakciju.

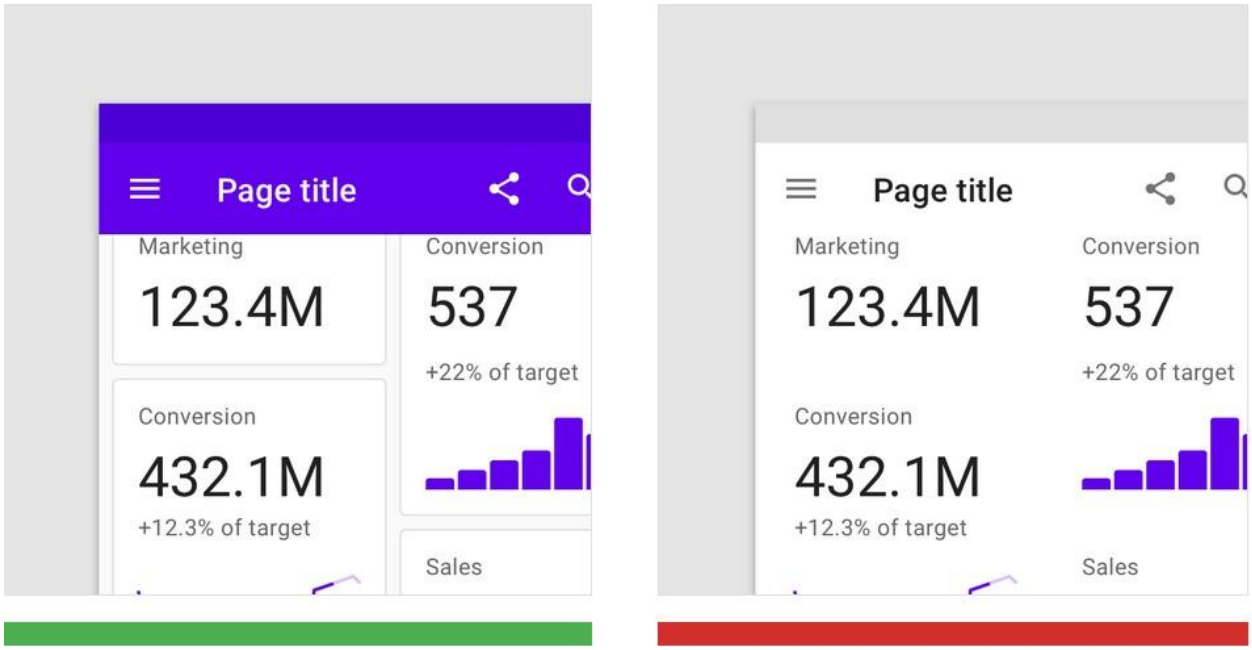


Sl. 2.7. Prikaz zadanih uzvišenja na mobilnom uređaju s prednje i bočne strane

Djelomično ili potpuno preklapanje površina nam ukazuje da imaju različite vrijednosti uzvišenja. One s višom vrijednosti djeluju kao da su ispred onih s nižom vrijednosti. Preklapanje se može dogoditi kao rezultat kretnje koja je promijenila poziciju površine u korisničkom sučelju. Primjer preklapanja vidljiv je na slici 2.8.



Sl. 2.8. Sjene naglašavaju rubove površine, preklapanje i razinu uzvišenja



Sl. 2.9. Primjer dobre i loše upotrebe uzvišenja i preklapanja

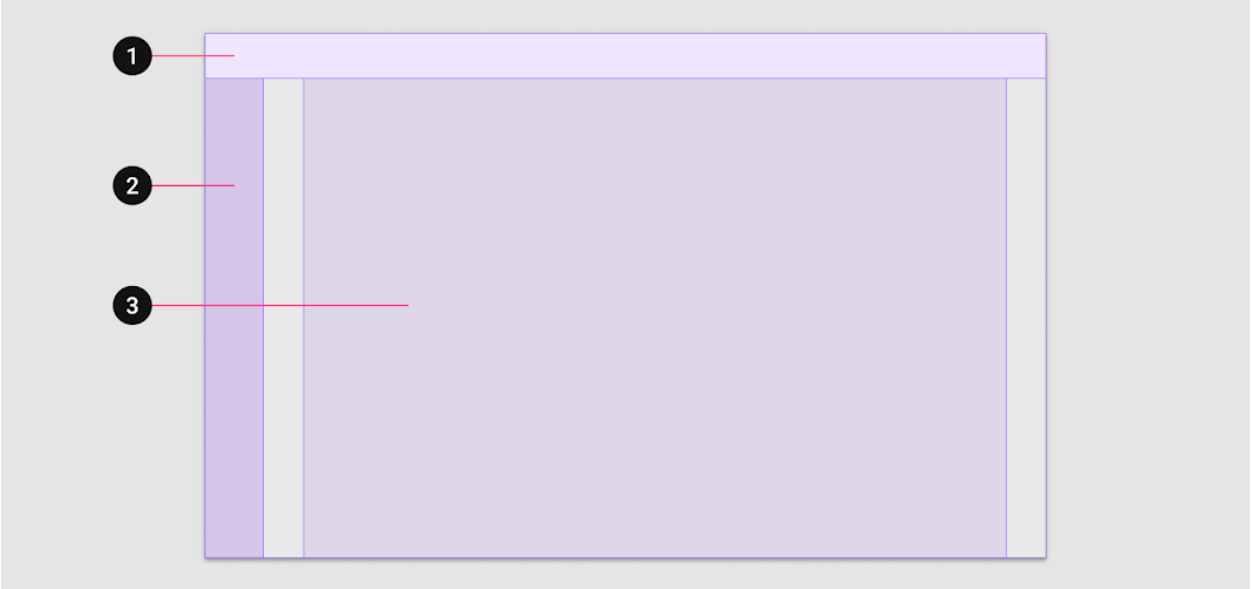
Na slici 2.9. na lijevom dijelu slike prikazano je kako se gornja traka preklapa s karticama i time jasno ukazuje da je iznad njih. Na desnom dijelu slike vidimo kako u potpunosti nedostaju rubovi i preklapanja. Temeljem toga ne može se odrediti koliko površina se na zaslonu nalazi niti njihove razine uzvišenja.

2.2. Raspored

Raspored u Material Design-u služi se ujednačenim elementima i razmacima kako bi se potaknula dosljednost na različitim platformama i veličinama zaslona. Osnovne niti vodilje su:

- Predvidljivost: korištenje intuitivnih, dosljednih rasporeda i prostorne organizacije
- Dosljednost: korištenje sustava rešetki, ključnih linija i ispuna
- Responzivnost: korištenje rasporeda prilagodljivih korisniku i uređaju

Osnovu rasporeda čine određene regije koje se sastoje od komponenti i elemenata slične funkcije koje mogu sadržavati manje regije. Glavne regije čine: aplikacijska traka, navigacijska regija i tijelo. One su vidljive na slici 2.10.



Sl. 2.10. 1-aplikacijska traka; 2-navigacijska regija; 3-tijelo

O aplikacijskoj traci i navigaciji više će riječi biti kasnije.

Tijelo se koristi za prikaz većine sadržaja aplikacije te najčešće sadrži komponente poput kartica, gumbova, slika i slično. Ono ima svoje vodoravne i okomite dimenzije, broj stupaca i rubove (margine). Tijelo također ima određene točke prijeloma pomoću kojih se određuje prikaz sadržaja i rubova. Na izrazito malim točkama prijeloma rubovi su vrijednosti 16 dp. Kako se raspored povećava tako se i tijelo širi u odnosu na širinu zaslona.

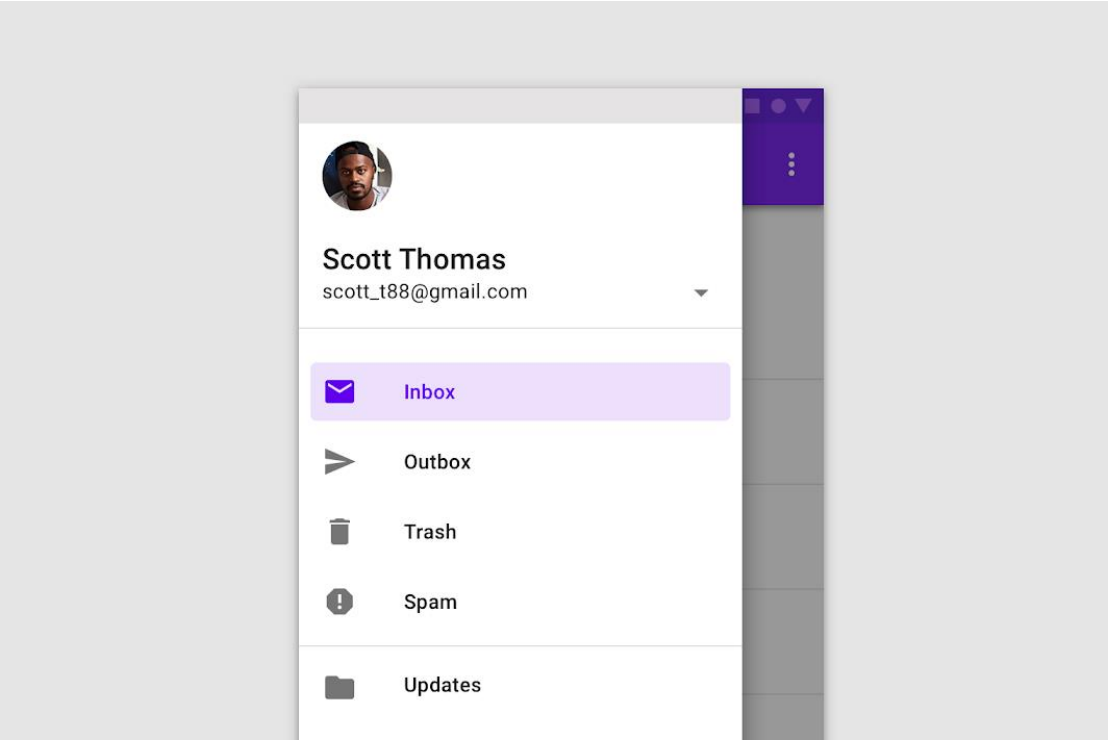
2.3. Navigacija

Navigaciju bismo mogli definirati kao čin prelaženja ili premještanja između zaslona aplikacije s ciljem obavljanja određene radnje. Korisnik se može pomicati u jednom od tri navedena smjera:

- Bočna navigacija odnosi se na kretanje između zaslona na istoj razini hijerarhije
- Navigacija prema naprijed odnosi se na kretanje između zaslona na uzastopnim razinama hijerarhije
- Navigacija prema natrag odnosi se na kretanje unatrag kroz zaslone kronološki ili hijerarhijski.

2.3.1. Bočna navigacija

Bočnu navigaciju mogu upotrijebiti aplikacije s nekoliko najviših razina hijerarhije. Ona se može ostvariti putem navigacijske ladice ukoliko aplikacija ima pet ili više odredišta, donje navigacijske trake ukoliko aplikacija ima između tri i pet odredišta te tabova ukoliko ih ima dva.



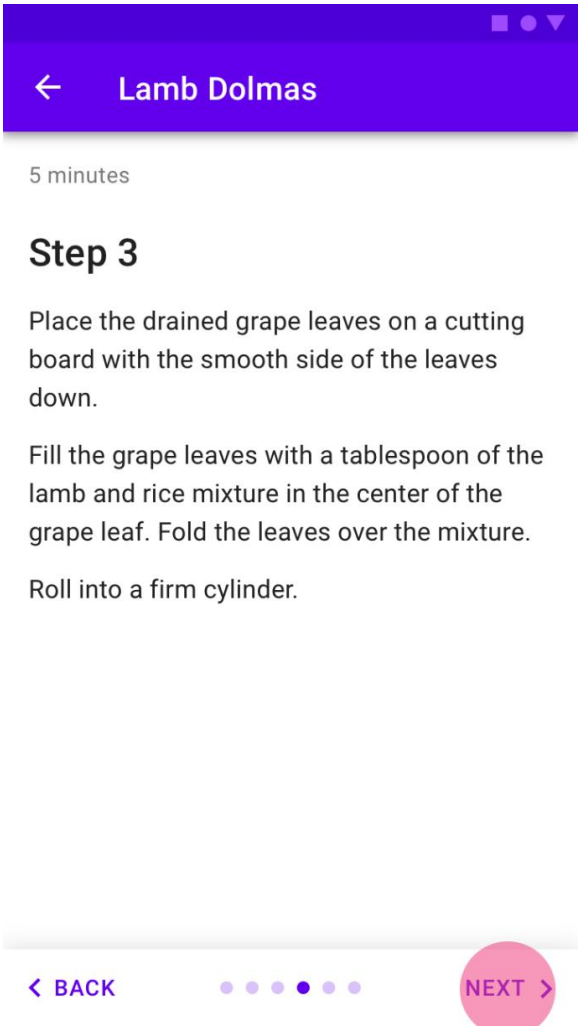
Sl. 2.11. Primjer navigacijske ladice s pet odredišta

2.3.2. Navigacija prema naprijed

Navigacija prema naprijed odnosi se na jednu od tri vrste kretnje. Kretnja prema dolje u hijerarhiji kako bi korisnik pristupio sadržaju na zaslonu djeteta iz zaslona roditelja. Slijedno kretanje (sekvencijalno) označava kretanje slijedeći tok zaslona, poput procesa registracije - izravna kretnja s jednog zaslona na drugi.

Za razliku od bočne navigacije koja koristi komponente, navigacija prema naprijed najčešće se odvija prilikom interakcije sa sadržajem. Može ze implementirati pomoću kartica, gumbova koje vode korisnika na novi zaslon, gumba za pretraživanje u jednom ili više zaslona te poveznica unutar sadržaja.

Na slici 2.12. prikazan je primjer slijednog kretanja. Možemo vidjeti kako nam gumb svojom pozicijom i tekstom daje do znanja da ćemo klikom na njega nastaviti kretnju tokom.



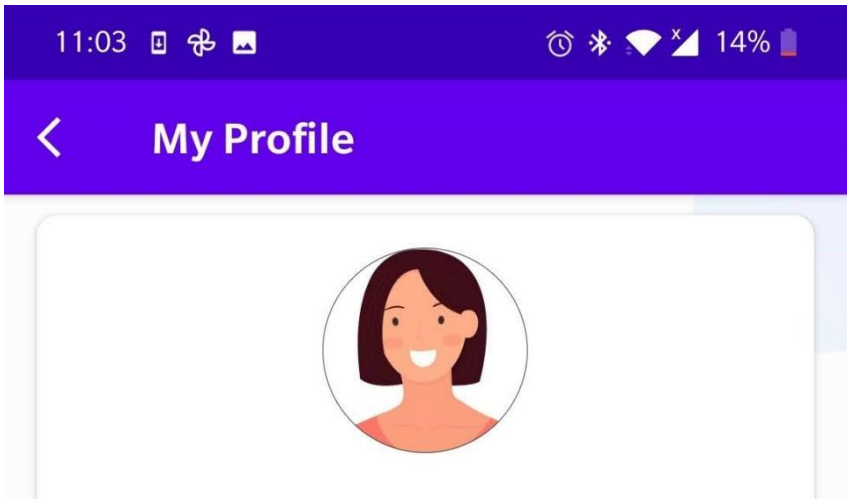
Sl. 2.12. Primjer slijednog kretanja

2.3.3. Navigacija prema natrag

Navigacija prema natrag odnosi se na kretanje unatrag između zaslona kronološki ili prema gore kroz hijerarhiju.

Kretanje unatrag kronološki odnosi se na navigaciju u suprotnom redoslijedu kroz korisnikove nedavno posjećene zaslone. Moguće je i kretanje iz aplikacije u aplikaciju. Ovu vrstu navigacije uglavnom prema zadanom podržava operacijski sustav ili platforma te oni definiraju ponašanje i način na koji mu korisnik može pristupiti.

Kretanje prema gore omogućuje korisniku pristup zaslonu iznad u hijerarhiji sve dok se ne dođe do najviše razine aplikacije. Ova vrsta trebala bi biti implementirana u svim zaslonima djeteta aplikacije te slijediti smjernice operacijskog sustava ili platforme. Na slici 2.13 prikazan je ovaj način navigacije.

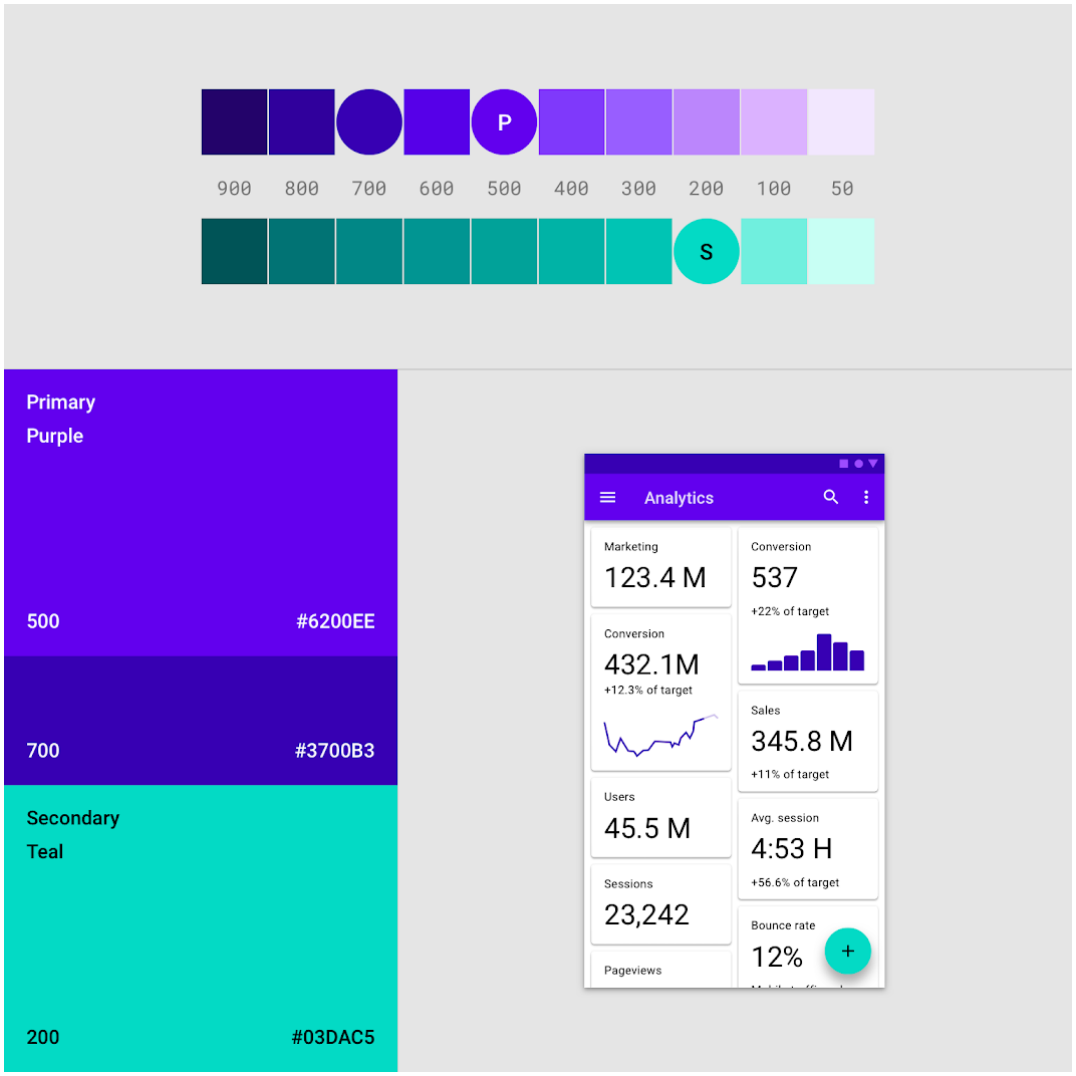


Sl. 2.13. Primjer kretanja prema gore

2.4. Boje

Boje omogućuju stvaranje jedinstvenih aplikacija koje odražavaju potreban stil ili brend, a da pri tom i dalje budu dijelom većeg sustava Material Design-a. Njegov sustav boja pomaže pri njihovoj primjeni na smislen način odabirom primarne i sekundarne boje koje odražavaju stil te njihovih svjetlijih i tamnijih varijanti. Teme u boji dizajnirane su kako bi bile skladne, omogućile razlikovanje elemenata i površina sučelja te osigurale pristupačnost teksta.

Boje ukazuju na interaktivne elemente i njihov odnos s drugim elementima. Važni bi elementi trebali biti istaknuti bojom, a da pri tom zadovoljavaju sve standarde čitljivosti. Material Design dolazi sa zadanom temom boja koja se može odmah koristiti, a ona uključuje primarne i sekundarne boje, njihove tamnije i svjetlije varijante te dodatne boje za ukazivanje grešaka ili primjenu na elementima poput pozadina.



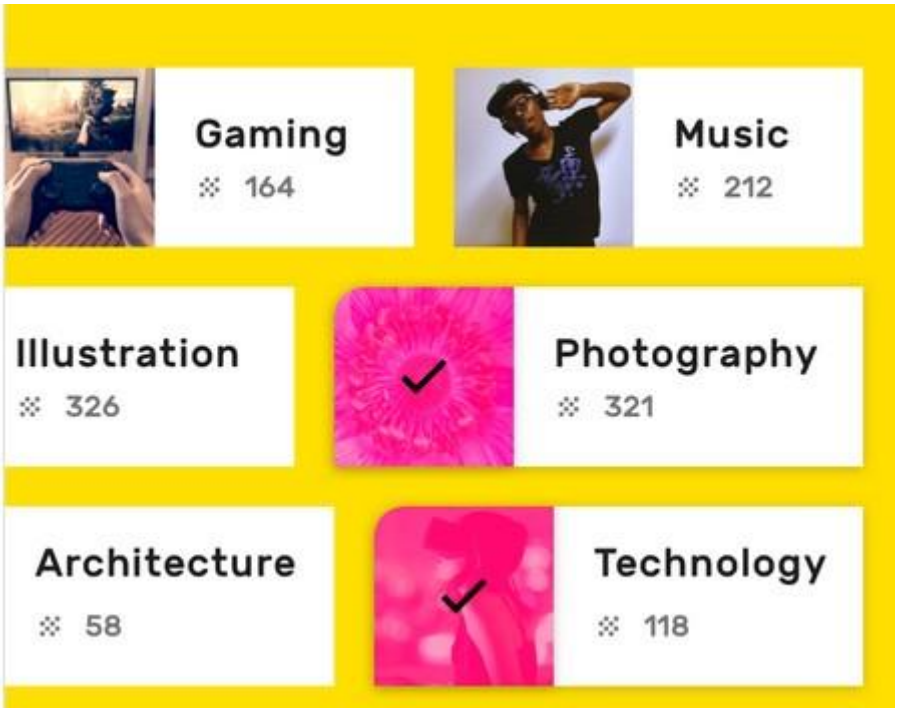
Sl. 2.14. Primjer korištenja primarne i sekundarne boje u aplikaciji

Primarna boja je ona najčešće prikazivana na komponentama i zaslonima aplikacije. Pomoću njezinih svijetlih i tamnijih varijanti moguće je kreirati kontrast između elemenata korisničkog sučelja poput statusne i navigacijske trake te razlikovanje elemenata unutar komponenata.

Sekundarna boja pruža dodatne načine za naglašavanje aplikacije. Ona nije nužna te njezino korištenje treba biti pažljivo primijenjeno samo na određene dijelove aplikacije poput plutajućeg akcijskog gumba, trake koja prikazuje napredak ili klizača i sklopki. Poput primarne boje, sekundarna boja također može imat svoje svjetlije i tamnije varijante.

U Material Design-u boja privlači pažnju određenim elementima na zaslonu. Kada je element boje koja kontrastira onoj na kojoj se nalazi, govori korisniku kako je on važan. Zbog postojanja različitih tema boja postoje razni načini na koje se može korisniku dati do znanja da element ima određenu važnost, kao što, primjerice, višebojni element iskače iz jednobojne pozadine.

Boja i oblik materijala mogu se nadopunjavati kako bi naglasili određene akcije poput davanja do znanja korisniku da je element odabran, kao iz primjera na slici 2.15.



Sl. 2.15. Primjer suradnje boje i oblika materijala

Boje prenose značenje različitih elemenata korisničkog sučelja. Primjerice, vremenska aplikacija može prikazivati boje koje označavaju trenutne vremenske uvjete, dok aplikacija za navigaciju može bojama prikazati stanje u prometu.

Korištenje boja treba biti dosljedno na način da određene boje uvijek imaju isto značenje čak i ako se kontekst promijeni. Također treba obratiti pažnju na boje s lokalnim ili kulturnim značenjem. Na primjer, crvena u brojnim kulturama označava boju upozorenja, ali u nekima ne mora biti tako.

Boje korisniku govore o trenutnom stanju u aplikaciji, njezinim komponentama ili elementima te o promjeni istog. Prilikom promjene stanja boja bi trebala biti uočljiva, jer blage promjene boje mogu proći neprimijećeno. Iz tog je razloga na promjenu stanja, uz promjenu boje, najbolje ukazati i pokretom, točnije, promjenom položaja elementa. Kao što je spomenuto, boja ukazuje korisniku na interakciju s elementima te njihov odabir.

2.5. Tipografija

Poput boja, tipografija se koristi kako bi se dizajn i sadržaj predstavili što učinkovitije i jasnije. Material Design podržava široki raspon stilova kojima se mogu izraziti i zadovoljiti sve potrebe proizvoda i njegovog sadržaja. Za izražavanje veličine fonta u upotrebi je nekoliko jedinica na više platformi. Na operacijskom sustavu android koristi se jedinica sp (engl. *scaleable pixels* ili *scale-independent pixels*), operacijski sustav iOS koristi pt (engl. *points*), a internet preglednici rem (engl. *root em size*). Tablica 2.1. prikazuje pretvorbu između navedenih jedinica.

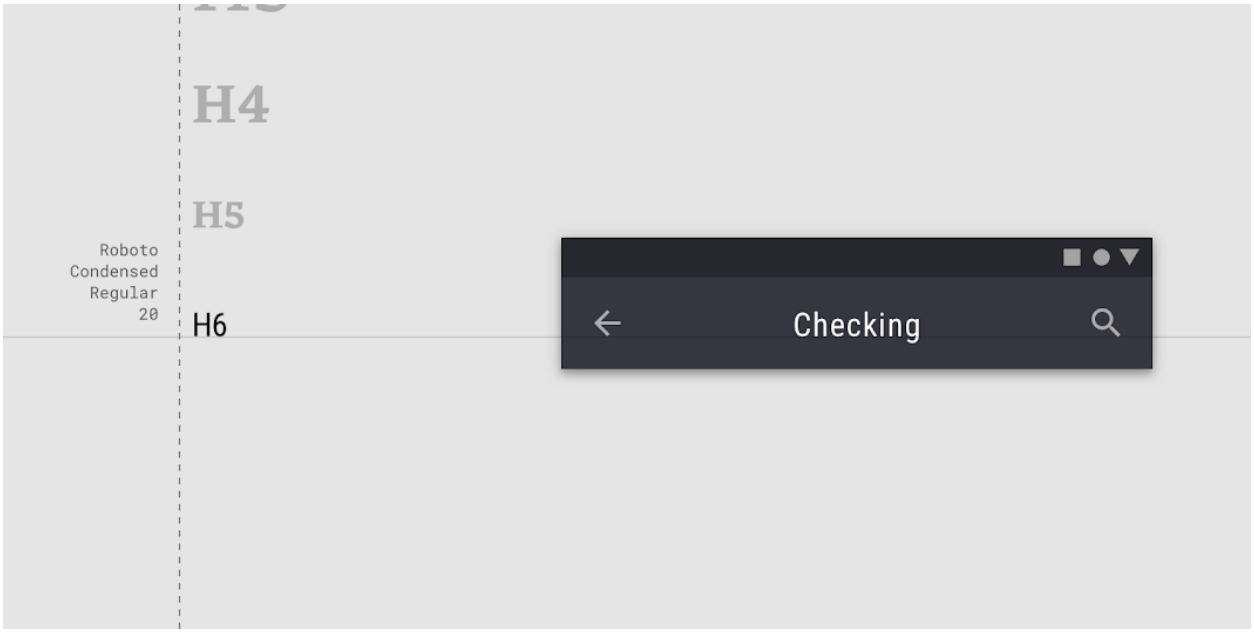
Tab 2.1. Pretvorba jedinica između platformi

Android	iOS	Web
10 sp	10 pt	0.625 rem
12 sp	12 pt	0.75 rem
24 sp	24 pt	1.5 rem
60 sp	60 pt	3.75 rem

Internet preglednici računaju vrijednost rem-a prema veličini korijenskog elementa. Za moderne preglednike zadana vrijednost iznosi 16 piksela, stoga se pretvorba računa prema formuli 2-1:

$$rem = \frac{SP\ veličina}{16} \qquad (2-1)$$

Naslovi predstavljaju najveći tekst na zaslonu te njegove veličine mogu biti u rasponu od jedan do šest. Za naslove moguće je odabrati font izražajnog ili nekonvencionalnog stila i time se privući korisniku pozornost.



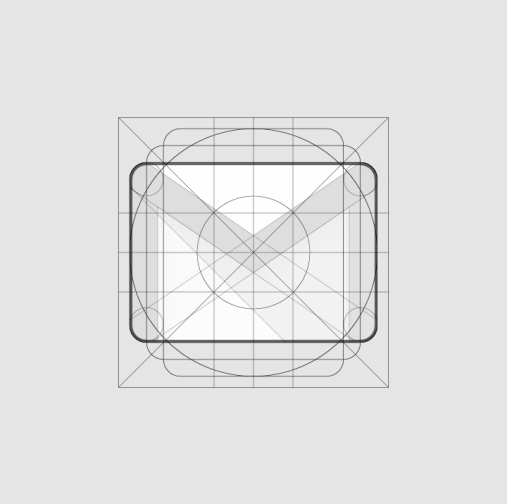
Sl. 2.16. Primjer sans fonta veličine šest

2.6. Ikonografija

Ikone aplikacije su vizualni izraz produkta i njegovih usluga. One na jednostavan način korisniku prenose namjeru aplikacije. Iako su ikone vizualno različite, one trebaju slijediti iste koncepte. Nadahnutost stvarnim svijetom vidljiva je i u ikonama. Svaka je ikona izrezana, presavijena i osvijetljena poput papira, ali je predstavljena jednostavnim grafičkim elementima.

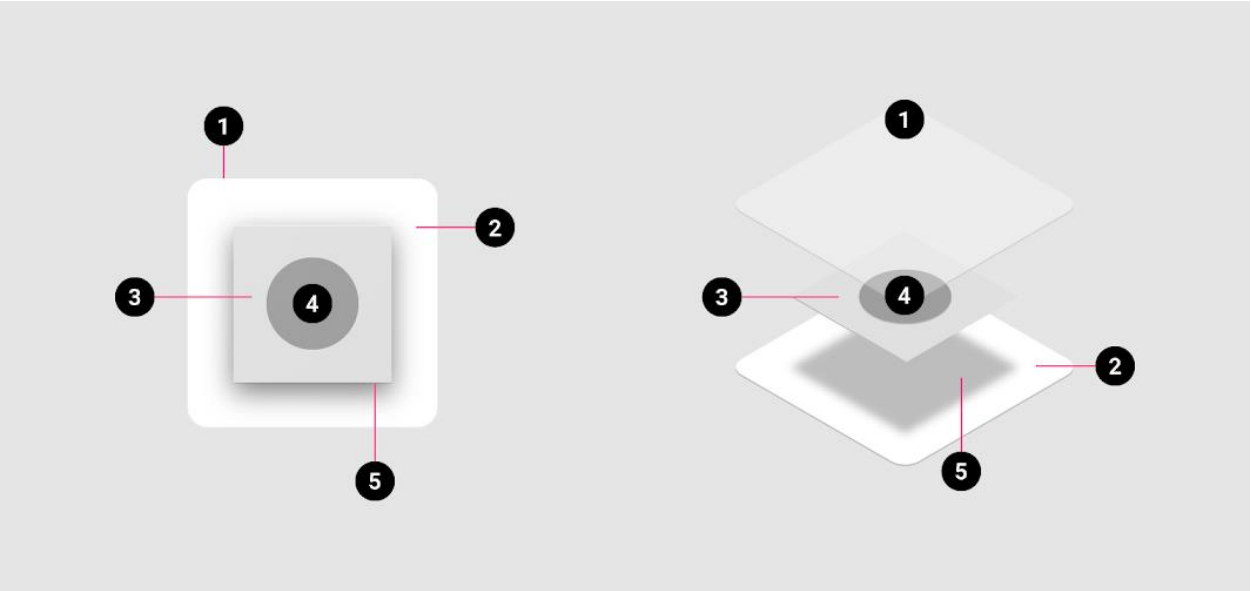
Prilikom stvaranja ikone aplikacije, pregled i uređivanje trebaju se odvijati na 400%, točnije 192x192 dp pri čemu će rubovi biti prikazani sa 4 dp. Korištenjem ove veličine bilo koje promjene će biti proporcionalno skalirane što znači da će rubovi ostati dobro obrađeni, a sama ikona poravnata prilikom vraćanja na 100%, odnosno 48 dp.

Izrada ikona temeljena je na sustavu rešetke. Korištenjem osnovnih linija i oblika za bazu vrlo je jednostavno postići dosljednost. Osnovne linije i oblici su krug, kocka, pravokutnici i dijagonale.



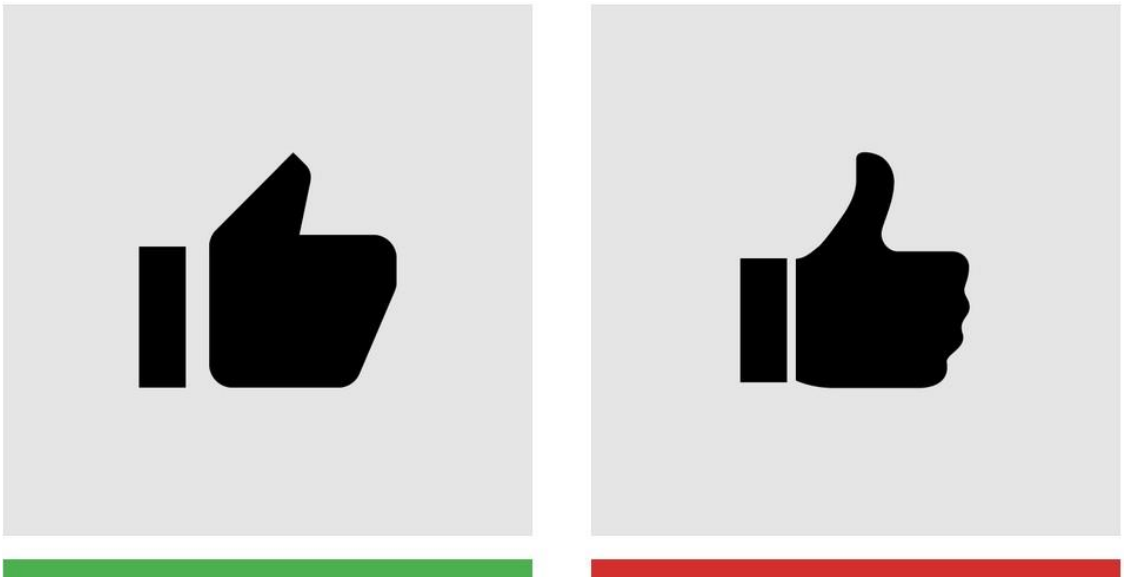
Sl. 2.17. Osnovne linije i oblici ikona

Osnovna struktura ikone temelji se na slaganju svakog elementa ispred prethodnog tako da su svaki logotip i ikonica dizajnirani odozdo prema gore. Struktura slaganja ikone prikazana je na slici 2.18. Material pozadina je najudaljeniji sloj. Material prvi plan je sloj koji baca sjenu na pozadinu. Boja je primijenjena samo na jednom elementu ili na njegovom dijelu. Krajnji sloj je onaj na kojem se može pojaviti efekt osvjetljenja ukoliko se koristi.



Sl. 2.18. Struktura ikone: 1-krajnji sloj; 2-material pozadina; 3-material prvi plan; 4-boja; 5-sjene

Osim ikona aplikacije, u ikonografiju pripadaju i sustavske ikone koje predstavljaju učestale akcije, sustavske datoteke i slično. Njihov dizajn treba biti jednostavan, moderan, a u nekim slučajevima i neobičan. Svaka ikona predstavljena je svojim minimalnim oblikom koji izražava samo osnovne karakteristike. Poput stvaranja ikona aplikacije, i kod sustavskih ikoni koristi se sustav rešetke. Sustavske ikone trebaju biti podebljane i koristiti geometrijske oblike, a ne prirodne. Primjer je prikazan na slici 2.19.



Sl. 2.19. Primjer dobre i loše sustavske ikone

3. KOMPONENTE

Komponente su gradivni blokovi Material Design-a za stvaranje korisničkog sučelja. One su stvorene s ciljem rješavanja određenih zahtjeva korisničkog sučelja poput:

- prikaza: postavljanje i organiziranje sadržaja
- navigacije: omogućavanje kretanja korisnicima kroz aplikaciju
- akcije: omogućavanje obavljanja radnji
- korisničkog unosa informacija
- komunikacije s korisnikom

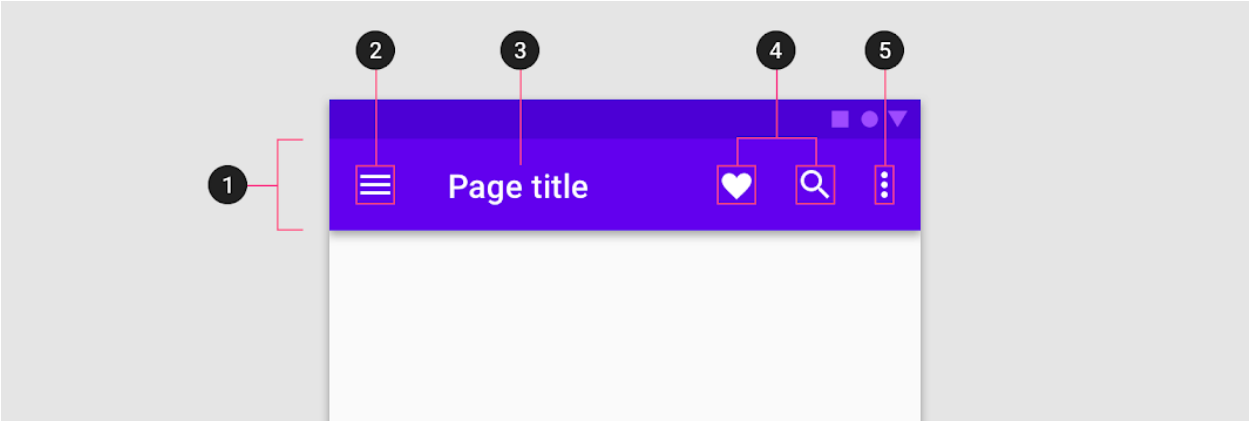
Biblioteke s komponentama dostupne su za Android, iOS, Flutter i web. U ovom završnom radu obrađene su komponente korištene za Android. Za početak korištenja potrebno je na razini aplikacije u Gradle datoteci uključiti Material Components for Android biblioteku.

```
implementation 'com.google.android.material:material:1.3.0'
```

Sl. 3.1. Gradle implementacija

3.1. Gornja aplikacijska traka

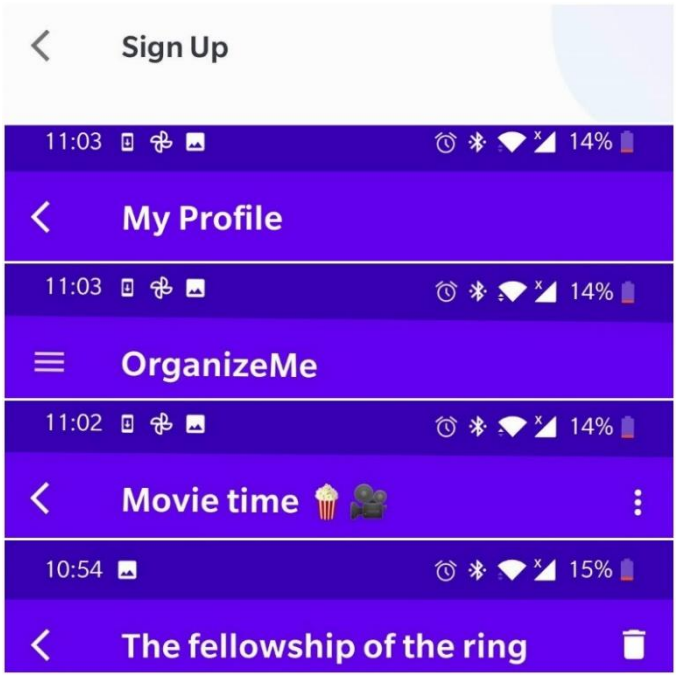
Gornja aplikacijska traka omogućuje navigaciju i prikaz ključnih radnji na vrhu zaslona mobilnog uređaja. Po potrebi, pri pomicanju sadržaja, može se sakriti ili pretvoriti u kontekstualnu traku. Gornje aplikacijske trake pružaju pouzdano vođenje korisnika kroz aplikaciju. Preporučeni raspored elemenata gornje aplikacijske trake jest s lijeve strane navigacija, a desno od nje naslov, ukoliko je potreban. Pokraj naslova gumbovi s radnjama, a ukoliko za time postoji potreba, krajnje desno može se iskoristiti gumb za otvaranje dodatnog izbornika.



Sl. 3.2. Gornja aplikacijska traka: 1-spremnik; 2-navigacijski izbornik; 3-naslov; 4-ikone određenih radnji; 5-dodatni izbornik

Ukoliko se koristi, navigacijska ikona treba biti poravnata s lijeve strane gornje aplikacijske trake te imati jedan od idućih oblika: ikona izbornika koji otvara navigacijsku ladicu, strjelicu prema gore koja označava kretanje prema gore u hijerarhiji ili strjelicu prema natrag koja označava povratak na prethodni zaslon. Naslov, ukoliko se koristi, treba biti pokraj navigacijske ikone te, ili opisati zaslon ili dio aplikacije na kojem se korisnik trenutno nalazi, ili pisati naziv aplikacije.

Navedena komponenta jedna je od češće korištenih u aplikacijama te je iskorištena i u OrganizeMe aplikaciji na nekoliko načina što je vidljivo na slici 3.3.



Sl. 3.3. Načini implementacije gornje aplikacijske trake u OrganizeMe aplikaciji

Na slici 3.4. prikazan je korišteni layout u XML datotekama.

```
<com.google.android.material.appbar.AppBarLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:theme="@style/AppTheme.AppBarOverlay"
    >

    <androidx.appcompat.widget.Toolbar
        android:id="@+id/toolbar_my_profile_activity"
        android:layout_width="match_parent"
        android:layout_height="?attr/actionBarSize"
        android:background="?attr/colorPrimary"
        app:popupTheme="@style/AppTheme.PopupOverlay"
        />

</com.google.android.material.appbar.AppBarLayout>
```

Sl. 3.4. Korišteni layout u XML datotekama za prikaz gornje aplikacijske trake

```
<style name="AppTheme.AppBarOverlay" parent="ThemeOverlay.AppCompat.Dark.ActionBar"></style>

<style name="AppTheme.PopupOverlay" parent="ThemeOverlay.AppCompat.Light"></style>
```

Sl. 3.5. Stilovi iskorišteni za prikaz gornje aplikacijske trake

U programskom dijelu korištena je funkcija setUpActionBar() čiji je kod vidljiv na slici 3.6. i koja se poziva u onCreate() funkciji. Unutar navedene funkcije pomoću View Binding-a povezuju se XML elementi i Kotlin kod, postavlja se action bar te potrebna ikona za navigaciju.

```
private fun setUpActionBar(){
    setSupportActionBar(myProfileBinding.toolbarMyProfileActivity)

    myProfileBinding.toolbarMyProfileActivity.setNavigationIcon(R.drawable.ic_action_navigation_menu)

    val actionBar = supportActionBar;
    if(actionBar != null){
        actionBar.setDisplayHomeAsUpEnabled(true);
        actionBar.setHomeAsUpIndicator(R.drawable.ic_white_color_back_24dp);
        actionBar.title = "My Profile";
    }
    myProfileBinding.toolbarMyProfileActivity.setNavigationOnClickListener{ it: View!
        onBackPressed();
    }
}
```

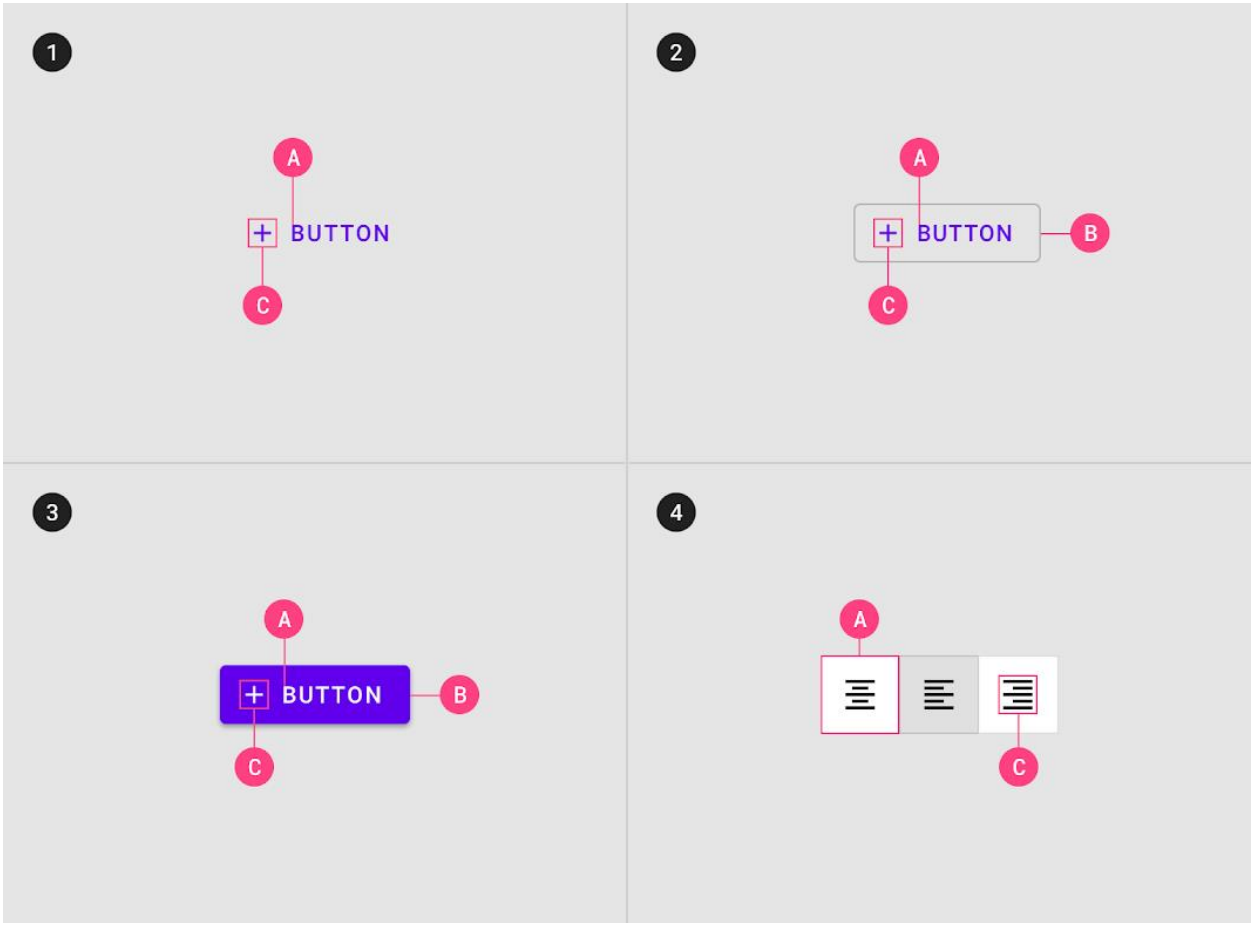
Sl. 3.6. Implementacija setUpActionBar() funkcije

3.2. Gumbovi

Jednostavnim dodirom omogućuju korisnicima obavljanje željenih radnji. Često su dijelom korisničkog sučelja i koriste se na mjestima poput dijaloških okvira, obrazaca, kartica i slično. Trebaju biti lako uočljivi i korisniku dati do znanja da pomoću njih pokreću radnju.

Kao što je vidljivo na slici 3.7. postoji nekoliko vrsta gumbova i njihovi dijelovi:

- Tekstualni gumb (1)
- Obrisni gumb (2)
- Puni gumb (3)
- Preklopni gumb (4)



Sl. 3.7. Gumbovi: A-oznaka teksta; B-spremnik; C-ikona (proizvoljno)

Tekstualni i obrisni gumbovi koriste tekstualne oznake koje opisuju radnju koja će se izvršiti klikom na gumb. Po zadanom, Material Design koristi velika slova unutar gumbova kako bi se lakše razlikovao od ostatka sadržaja.

Tekstualni gumbovi najčešće su korišteni za manje važne radnje u dijaloškim okvirima ili karticama. Tekstualna oznaka najvažniji je element tog gumba i mora biti izražena od ostatka, ali tekst mora ostati jasan i sažet. Dijaloški okviri koriste navedene gumbove jer su povoljni za objedinjavanje teksta okvira i radnje koju gumbovi pokreću.

Obrisni gumbovi najčešće su korišteni za radnje od srednje važnosti, odnosno radnje koje su važne, ali nisu primarne radnje u aplikaciji. Obris oko tekstualne oznake može biti prikazan na način da širina gumba bude jednaka širini tekstualne oznake s ispunom od 16 dp. Obrisni gumbovi mogu biti korišteni na raznim pozadinama jer je njegov spremnik proziran sve do trenutka interakcije.

Puni gumbovi razlikuju se od ostalih uzvišenjem i ispunjenjem. Korišteni su za primarne radnje trenutnog zaslona, ali i aplikacije. Spremnik gumba može biti prikazan na način da širina gumba bude jednaka širini tekstualne oznake s ispunom od 16 dp. Unutar ove vrste gumba može se koristiti i ikona koja jasno govori koju vrstu radnje će gumb izvršiti.

Preklopni gumbovi mogu biti korišteni za grupiranje srodnih radnji, a samo jedna radnja može biti odabrana u trenutku.

Govoreći o hijerarhiji važnosti gumbova, raspored zaslona bi trebao sadržavati jedan istaknuti gumb koji privlači pozornost i jasno daje do znanja kako su svi ostali gumbovi manje važni od njega. Može biti popraćen jednim ili više gumbova manje važnosti.

U OrganizeMe aplikaciji korišteno je nekoliko vrsta gumbova. Puni i obrisni gumbovi vidljivi su na slici 3.8.

Sl. 3.8. Puni i obrisni gumb

Puni gumb korišten je za primarne radnje i korišten je na nekoliko mjesta u aplikaciji. Za oba gumba izrađena je nova „layout resource“ datoteka u kojoj su definirana svojstva gumba. Kako bi se razbila monotonost korišten je gradijent s lijeva na desno boja heksadekadskih vrijednosti #FF3700B3 i #7E3FF2. U istoj je datoteci definiran kut zakrivljenosti rubova gumba. Navedena se datoteka koristi u XML datoteci kao pozadina gumba.

```xml
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="rectangle">

    <gradient
        android:angle="360"
        android:startColor="@color/purple_700"
        android:endColor="@color/shape_button_rounded_colorEnd2"
        android:type="linear"
        >

    </gradient>

    <corners android:radius="10sp"/>

</shape>
```

Sl. 3.9. Datoteka shape_button_rounded.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="rectangle">

    <stroke android:color="@color/purple_700"
        android:width="1dp"/>

    <solid  android:color="@color/white"/>
    <corners android:radius="10sp"/>

</shape>
```

Sl. 3.10. Datoteka shape_button_white_border.xml

```
<androidx.appcompat.widget.AppCompatButton
    android:id="@+id/btn_sign_in_intro"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_gravity="center"
    android:layout_marginStart="20dp"
    android:layout_marginTop="50dp"
    android:layout_marginEnd="20dp"
    android:background="@drawable/shape_button_rounded"
    android:foreground="?attr/selectableItemBackground"
    android:gravity="center"
    android:paddingTop="8dp"
    android:paddingBottom="8dp"
    android:text="Sign In"
    android:textColor="@color/white"
    android:textSize="18sp"

    />
```

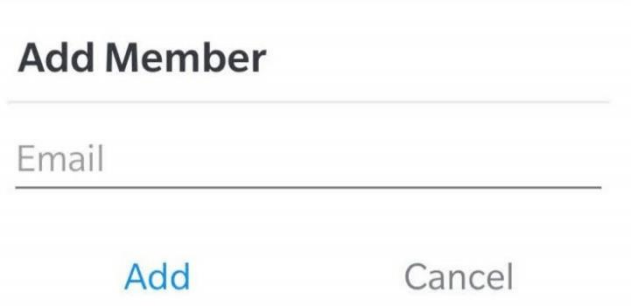
Sl. 3.11. Primjer korištenja stvorenih datoteka u activity_intro.xml

Primjer programskog koda jednog od gumbova nalazi se na slici 3.12. gdje se u onCreate() funkciji postavlja slušatelj koji klikom na gumb pokreće zadani kod, u ovom slučaju funkciju za registraciju korisnika.

```
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    binding = ActivitySignUpBinding.inflate(layoutInflater);
    setContentView(binding.root);

    setUpActionBar();

    binding.btnSignUpSignUpActivity.setOnClickListener{ it: View!
        registerUser();
    }

}
```

Sl. 3.12. Primjer programske implementacije jednog od gumbova

U aplikaciji je za određene radnje korišten i tekstualni gumb vidljiv na slici 3.13. Programski je implementiran kao TextView koji ima slušatelja i time se ponaša kao gumb za pokretanje željene radnje.



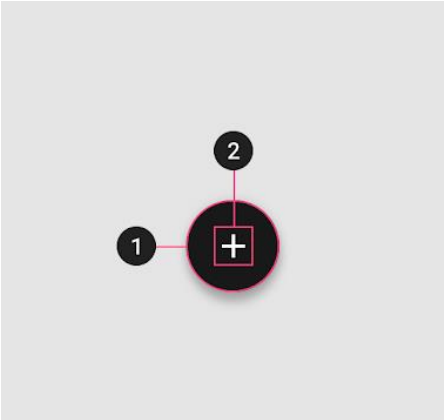
Sl. 3.13. Primjer tekstualnog gumba

```
<TextView
    android:id="@+id/tv_add_member"
    android:layout_width="0dp"
    android:layout_height="match_parent"
    android:layout_weight="1"
    android:background="?attr/selectableItemBackground"
    android:gravity="center"
    android:padding="10dp"
    android:text="Add"
    android:textColor="@color/colorAccent"
    android:textSize="16sp"
    />

<TextView
    android:id="@+id/tv_cancel"
    android:layout_width="0dp"
    android:layout_height="match_parent"
    android:layout_weight="1"
    android:background="?attr/selectableItemBackground"
    android:gravity="center"
    android:padding="10dp"
    android:text="Cancel"
    android:textColor="@color/secondary_text_color"
    android:textSize="16sp"
    />
```

Sl. 3.14. Implementacija tekstualnog gumba u XML datoteci

3.3. Plutajući akcijski gumb

Plutajući akcijski gumb predstavlja glavnu ili najčešće korištenu radnju na zaslonu, a što su uglavnom radnje kreiranja sadržaja ili pokretanja procesa. Prikazuje se u donjem desnom kutu ispred sveukupnog sadržaja, najčešće kao krug s ikonom u sredini.

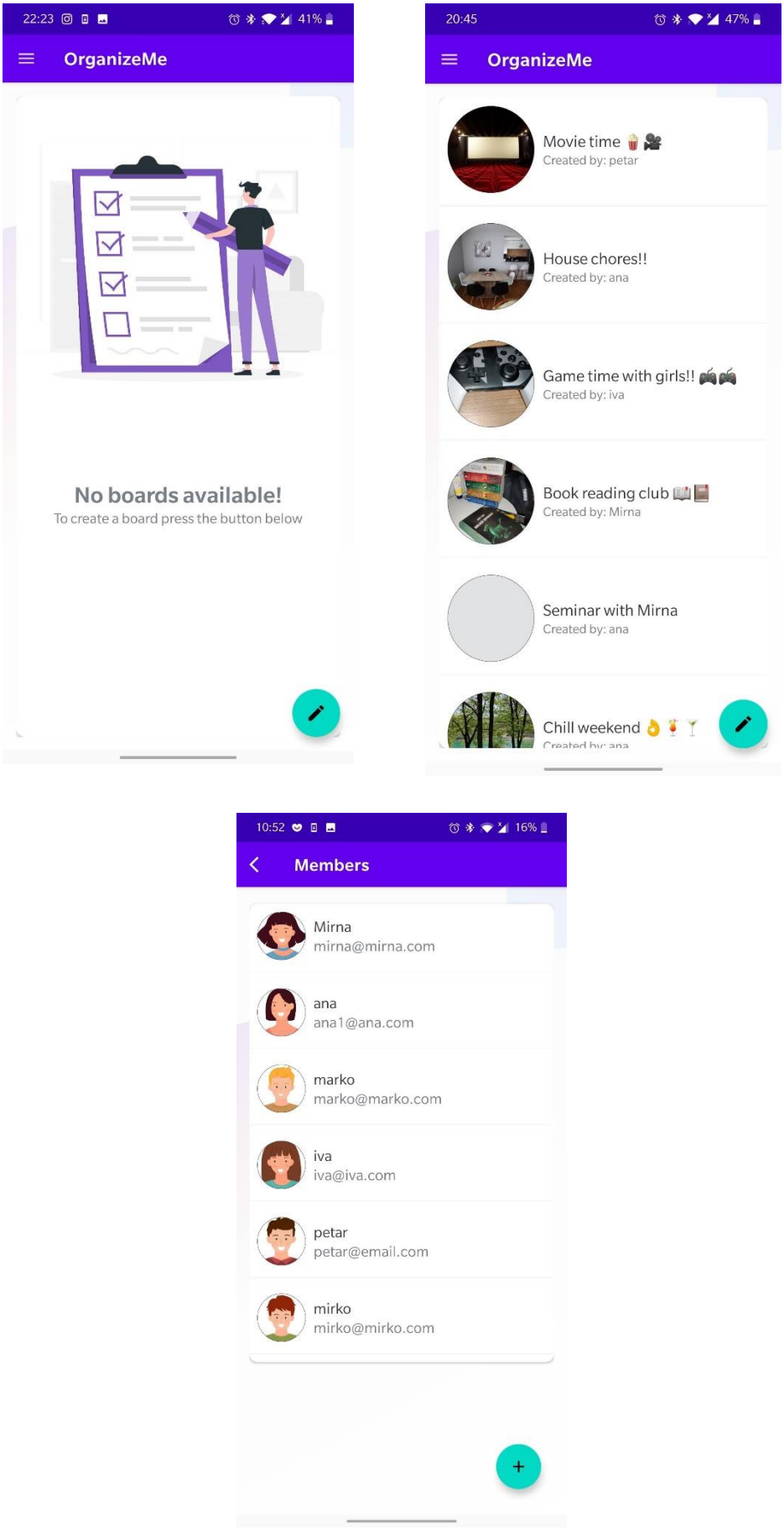


Sl. 3.15. Plutajući akcijski gumb: 1-spremnik; 2-ikona

Postoje manja i veća verzija ovoga gumba koje bi se trebale koristiti ovisno o veličini korištenog zaslona. Može koristiti primarnu ili sekundarnu boju aplikacije ovisno o preferencijama, no ako se koristi sekundarna boja treba biti pažljiv da ona ne preuzme ukoliko na zaslonu ima još elemenata te boje.

Ikona plutajućeg akcijskog gumba treba jasno dati do znanja o radnji koja će se klikom na gumb izvršiti. Također, tekst unutar ovog gumba treba izbjegavati. Plutajući akcijski gumb ne smije obavljati destruktivne radnje poput arhiviranja ili brisanja.

Navedna komponenta korištena je i unutar izrađene aplikacije. Jedan plutajući akcijski gumb na glavnom zaslonu čijim klikom se otvara zaslon za kreiranje ploče. Klikom na drugi gumb pojavljuje se dijaloški okvir za dodavanje članova u ploču. Korištena je sekundarna boja.



Sl. 3.16. Primjena plutajućeg akcijskog gumba u aplikaciji OrganizeMe

```
<com.google.android.material.floatingactionbutton.FloatingActionButton
    android:id="@+id/fab_create_board"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="bottom|end"
    android:layout_margin="16dp"
    app:srcCompat = "@drawable/ic_baseline_create_24"
    />
```

Sl. 3.17. Plutajući akcijski gumb u xml datoteci

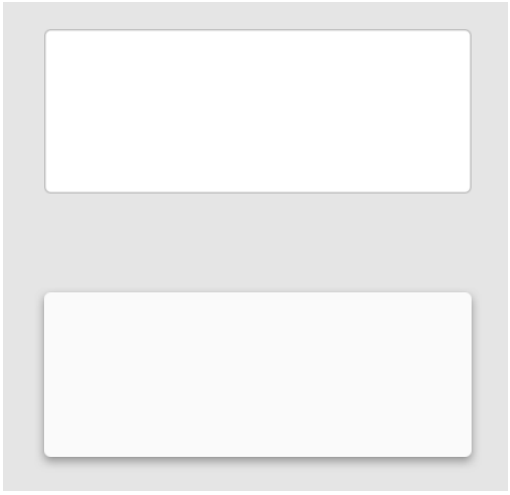
```
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    mainActivityBinding = ActivityMainBinding.inflate(layoutInflater)
    setContentView(mainActivityBinding.root)

    setUpActionBar();
    mainActivityBinding.navView.setNavigationItemSelectedListener(this);

    Firestore().loadUserData( activity: this, readBoardsList: true);

    appBarMainBinding.fabCreateBoard.setOnClickListener{ it: View!
        val intent = Intent( packageContext: this,CreateBoardActivity::class.java);
        intent.putExtra(Constants.NAME, mUserName);

        startActivityForResult(intent, CREATE_BOARD_REQUEST_CODE);
    }

}
```

Sl. 3.18. Programski dio plutajućeg akcijskog gumba

3.4. Kartica

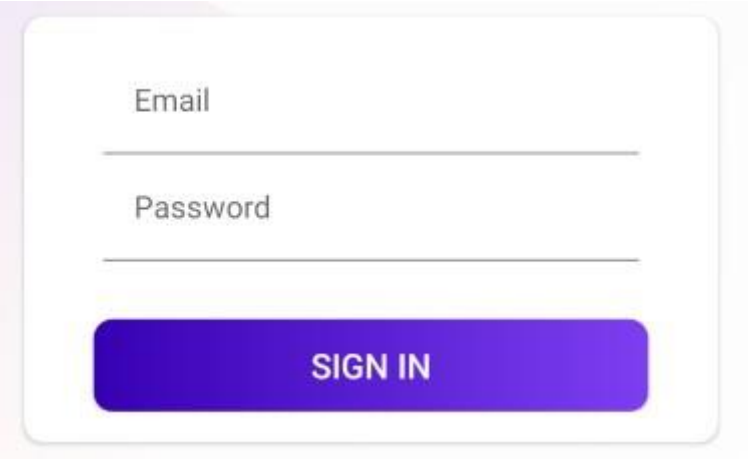
Kartice su površine na kojima se prikazuje sadržaj i radnje za istog. Trebaju na prvi pogled pružiti potrebne informacije, a elementi poput slika i teksta trebaju biti posloženi tako da hijerarhija važnosti bude lako uočljiva. Kartice trebaju biti samostalne i ne mogu se spojiti s drugima ili podijeliti u više njih. Razina uzvišenja kartice je najčešće 1 dp, ali po potrebi može biti viša.

Sama kartica ponaša se poput spremnika za sadržaj i ona je sama po sebi dovoljan element. Raspored sadržaja mijenja se ovisno o sadržaju, a može uključivati velike ili male slike, naslove, ikone ili gumbove. Po potrebi može se i koristiti razdjelnik sadržaja.



Sl. 3.19. Primjer dvaju kartica različitih uzvišenja

Kartice su korištene na više mjesta unutar izrađene aplikacije. Služe samo kao spremnik sadržaja i nemaju programsku implementaciju. Primjerice, na zaslonu za prijavu i registraciju korisnika korištene su kartice u kojima se nalaze podaci s jasno vidljivom hijerarhijom koje je potrebno popuniti i gumbom koji naglašava radnju. Primjer takve kartice i njene implementacije vidljiv je na slikama u nastavku.

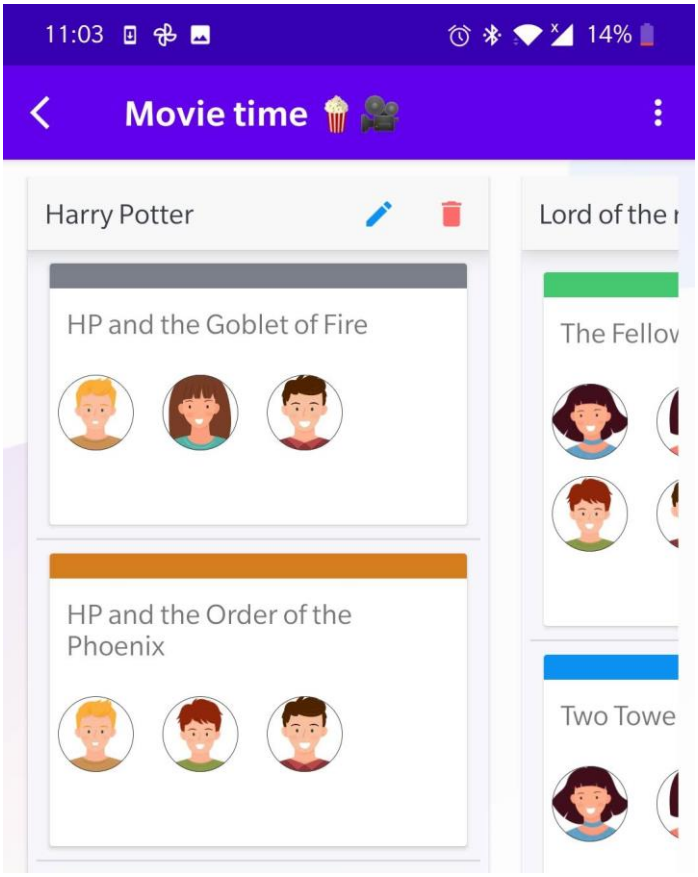


Sl. 3.20. Primjer kartice na zaslonu za prijavu korisnika

```
49      <androidx.cardview.widget.CardView
50          android:layout_width="match_parent"
51          android:layout_height="wrap_content"
52          android:layout_marginTop="60dp"
53          android:layout_marginStart="25dp"
54          android:layout_marginEnd="25dp"
55          android:elevation="10dp"
56          app:cardCornerRadius="10dp"
57          >
58
59          <LinearLayout
60              android:layout_width="match_parent"
61              android:layout_height="wrap_content"
62              android:orientation="vertical"
63              android:padding="16dp"
64              >
65
66              <com.google.android.material.textfield.TextInputLayout...>
84
85              <com.google.android.material.textfield.TextInputLayout...>
103
104             <androidx.appcompat.widget.AppCompatButton...>
122
123         </LinearLayout>
124
125     </androidx.cardview.widget.CardView>
```

Sl. 3.21. Implementacija kartice za prijavu korisnika

Također, kartica je korištena i za prikaz RecyclerView-a na glavnom zaslonu, ali i za prikaz pojedinih kartica zadataka korisnika.



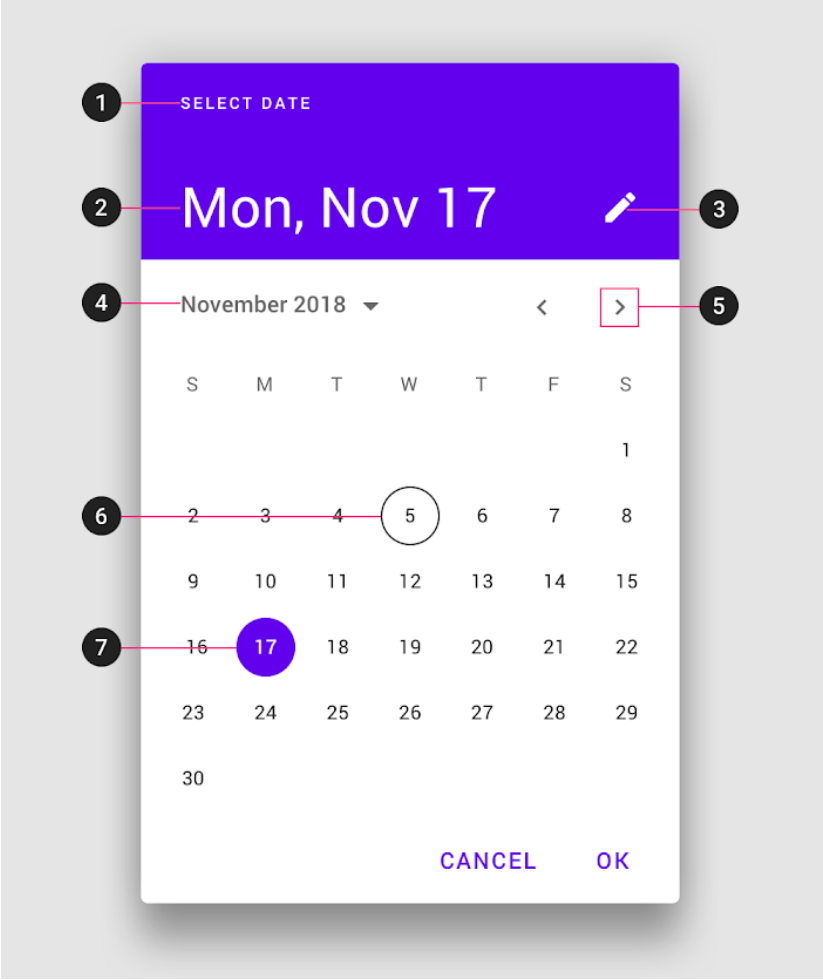
Sl. 3.22. Primjer kartice za prikaz zadataka i njihovih kartica

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.cardview.widget.CardView xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_margin="8dp"
    android:background="?attr/selectableItemBackground"
    android:elevation="5dp"
    android:orientation="vertical"
    >

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        >

        <View...>

        <TextView...>

        <androidx.recyclerview.widget.RecyclerView...>

        <TextView...>

    </LinearLayout>

</androidx.cardview.widget.CardView>
```

Sl. 3.23. Implementacija kartice za prikaz kartica zadataka – item_card.xml

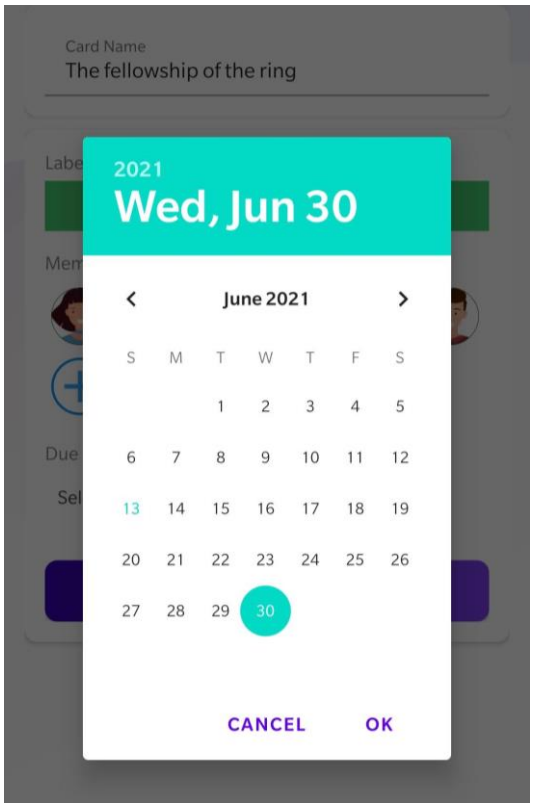
3.5. Birač datuma

Samo ime komponente govori njezinu namjenu, a to jest omogućavanje odabira datuma iz kalendara. Korisnik klikom na strjelice ili jednostavnim potezima na zaslonu može se kretati kroz mjesece. Ovisno o kontekstu, mogu prikazati buduće, sadašnje, ali i prošle datume te trebaju jasno naznačiti važne datume poput trenutno odabranih. To ne znači da se trebaju koristiti za svaku namjenu. Primjerice, birači datuma nisu prikladni za odabir datuma u dalekoj prošlosti ili budućnosti, poput unosa datuma rođenja.



Sl. 3.24. Birač datuma: 1-naslov; 2,7-odabrani datum; 3-proizvoljni gumb za prijelaz na unos tipkovnicom; 4-odabir godine; 5-stranice s mjesecima; 6-trenutni datum

U OrganizeMe aplikaciji klikom na pojedinu karticu sa zadatkom otvara se zaslon s više detalja za istu. Unutar njega moguće je postaviti krajnji rok izvršenja kartice klikom na tekstualni gumb. Implementacija u XML datoteci u ovom slučaju nije korištena, već isključivo programska.



Sl. 3.25. Prikaz birača datuma

```
81          mSelectedDueDateMilis = mBoardDetails
82              .taskList[mTaskListPosition]
83              .cards[mCardPosition]
84              .dueDate;
85
86          if(mSelectedDueDateMilis > 0 ){
87              val simpleDateFormat = SimpleDateFormat( pattern: "dd/MM/yyyy", Locale.ENGLISH);
88              val selectedDate = simpleDateFormat.format(Date(mSelectedDueDateMilis));
89              activityCardDetailsBinding.tvSelectDueDate.text = selectedDate;
90
91          }
92
93          activityCardDetailsBinding.tvSelectDueDate.setOnClickListener {  it: View!
94              showDatePicker();
95          }
```

Sl. 3.26. Implementacija unutar onCreate() funkcije u datoteci CardDetailsActivity.kt

```kotlin
private fun showDatePicker(){
    val calendar = Calendar.getInstance();
    val year = calendar.get(Calendar.YEAR);
    val month = calendar.get(Calendar.MONTH);
    val day = calendar.get(Calendar.DAY_OF_MONTH);

    val datePickerDialog = DatePickerDialog(
        context: this,
        DatePickerDialog.OnDateSetListener{ view, year, monthOfYear, dayOfMonth ->
            val sDayOfMonth = if(dayOfMonth < 1) "0$dayOfMonth" else "$dayOfMonth";
            val sMonthOfYear = if((monthOfYear + 1) < 10) "0${monthOfYear + 1}" else "${monthOfYear + 1}";

            val selectedDate = "$sDayOfMonth/$sMonthOfYear/$year";
            activityCardDetailsBinding.tvSelectDueDate.text = selectedDate;

            val sdf = SimpleDateFormat( pattern: "dd/MM/yyy", Locale.ENGLISH);
            val theDate = sdf.parse(selectedDate);

            mSelectedDueDateMilis = theDate!!.time;
    },
        year,
        month,
        day
    );

    datePickerDialog.show();

}
```

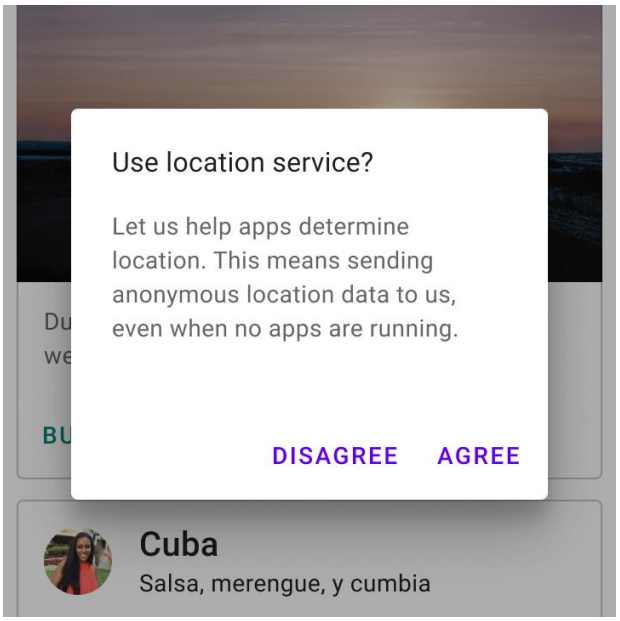
Sl. 3.27. Implementacija funkcije showDatePicker() u datoteci CardDetailsActivity.kt

U OnCreate() funkciji na slici 3.26. ukoliko je odabran datum, on se postavlja u TextView te je postavljen slušatelj na tekstualni gumb čijim klikom se poziva funkcija showDatePicker() prikazana na slici 3.27. U njoj se iz instance kalendara dohvaća datum te se stvara instanca DatePickerDialog-a u kojem se definira način ponašanja. Navedenu instancu potrebno je i prikazati na zaslonu.

3.6. Dijaloški okvir

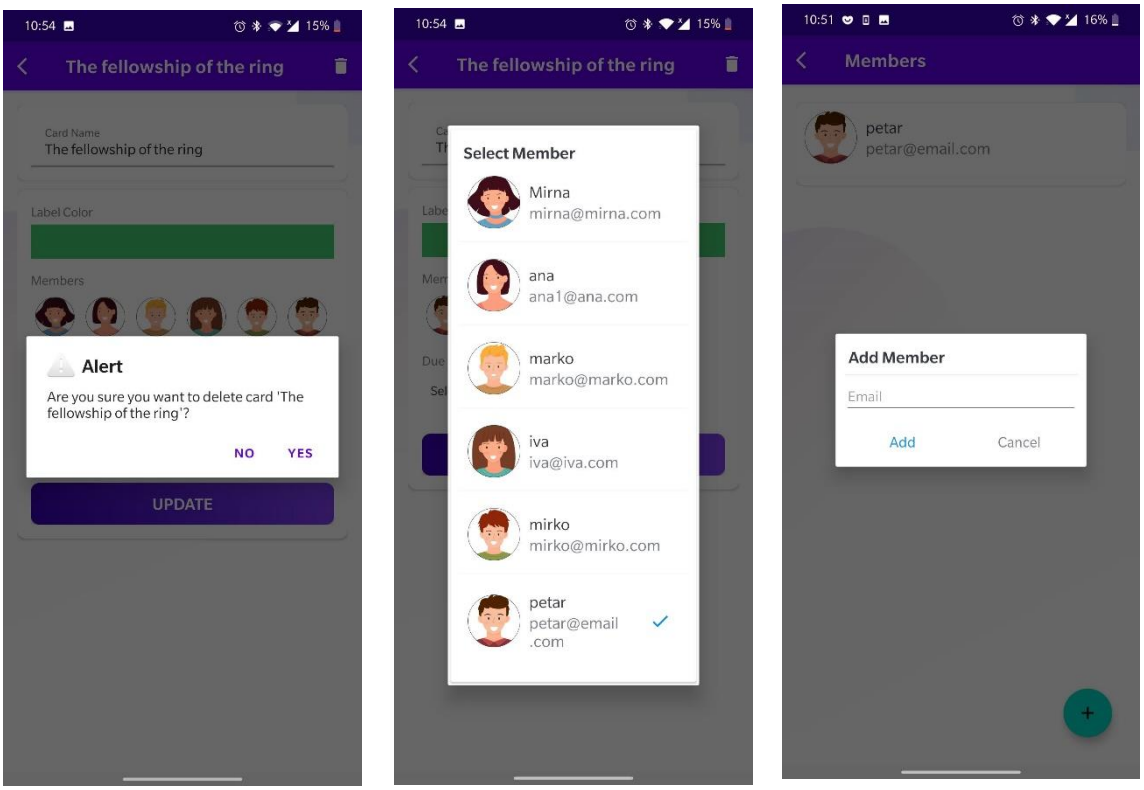
Dijaloški okvir može se zamisliti poput vrste prozora koji se pojavljuje ispred svog sadržaja aplikacije kako bi pružio važne informacije najčešće zahtijevajući odluku. Pri pojavljivanju oni onemogućuju sve funkcije aplikacije sve dok se ne poduzme potrebna radnja. Zbog svoje, iako namjerne, nametljive prirode, treba biti pažljiv s njihovom uporabom.

Dijaloški okvir trebao bi se koristiti kada se pojavi greška koja ometa normalan rad aplikacije ili pri pojavi kritičnih informacija koje zahtijevaju određenu radnju. Ima visok prioritet u odnosu na snackbar. Postoji nekoliko vrsta poput jednostavnih dijaloških okvira i okvira upozorenja.



Sl. 3.28. Primjer jednostavnog dijaloškog okvira

Dijaloški okviri, točnije jednostavni i okviri upozorenja, korišteni su na više mjesta unutar OrganizeMe aplikacije. Njihova primjena vidljiva je na slici 3.29.



Sl. 3.29. Primjer korištenih dijaloških okvira u OrganizeMe aplikaciji

Klikom na gumb za brisanje kartice pojavljuje se okvir upozorenja, a na slici 3.30. prikazana je njegova implementacija. Pomoću graditelja stvara se AlertDialog te mu se postavlja naslov, poruka i ikona. Definira se ponašanje za pozitivni i negativni gumb odluke te se na kraju AlertDialog prikazuje.

```
295     private fun alertDialogForDeleteCard(cardName: String){
296         val builder = AlertDialog.Builder( context: this);
297         builder.setTitle("Alert");
298         builder.setMessage("Are you sure you want to delete card '{cardName}'?")
299         builder.setIcon(android.R.drawable.ic_dialog_alert);
300
301         builder.setPositiveButton("Yes"){dialogInterface, which ->
302             dialogInterface.dismiss();
303             deleteCard();
304
305         }
306
307         builder.setNegativeButton("No"){dialogInterface, which ->
308             dialogInterface.dismiss();
309         }
310
311         val alertDialog: AlertDialog = builder.create();
312         alertDialog.setCancelable(false);
313         alertDialog.show();
314
315     }
```

Sl. 3.30. Implementacija funkcije alertDialogForDeleteCard() za prikaz dijaloškog okvira upozorenja

Dijaloški okvir za prikaz članova koristi recyclerView, a programska i XML implementacija okvira za dodavanje članova nalazi se na slikama 3.31. i 3.32. Napuhuje se XMl datoteka te se stvara instanca dijaloškog okvira gdje se implementiraju pozitivni i negativni tekstualni gumb.

```
99  private fun dialogAddSearchMember(){
100
101     val dialog = Dialog( context: this);
102
103     var dialogBinding: DialogAddSearchMemberBinding = DialogAddSearchMemberBinding.inflate(layoutInflater);
104     dialog.setContentView(dialogBinding.root);
105
106     dialogBinding.tvAddMember.setOnClickListener {  it: View!
107         val email = dialogBinding.etEmailSearchMember.text.toString();
108
109         if(email.isNotEmpty()){
110             dialog.dismiss();
111             showProgressDialog("Please wait");
112
113             Firestore().getMemberDetails( activity: this, email);
114
115         }
116         else{
117             Toast.makeText( context: this,  text: "Please enter members' email address",Toast.LENGTH_SHORT).show();
118         }
119     }
120     dialogBinding.tvCancel.setOnClickListener {  it: View!
121         dialog.dismiss();
122     }
123
124     dialog.show();
125
126 }
```

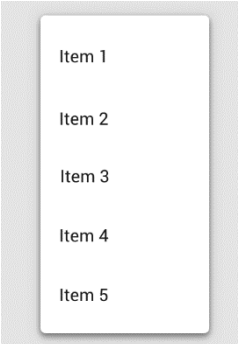
Sl. 3.31. Programska implementacija dijaloškog okvira za dodavanje članova

```
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <androidx.cardview.widget.CardView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:background="@drawable/shape_dialog_rounded"
        app:cardCornerRadius="3dp"
        app:cardElevation="3dp"
        >

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:gravity="center"
            android:orientation="vertical"
            android:padding="10dp"
            >

            <TextView...>

            <View...>


            <androidx.appcompat.widget.AppCompatEditText...>

            <LinearLayout...>
        </LinearLayout>
    </androidx.cardview.widget.CardView>
```

Sl. 3.32. Implementacija dijaloškog okvira za dodavanje članova u XML datoteci

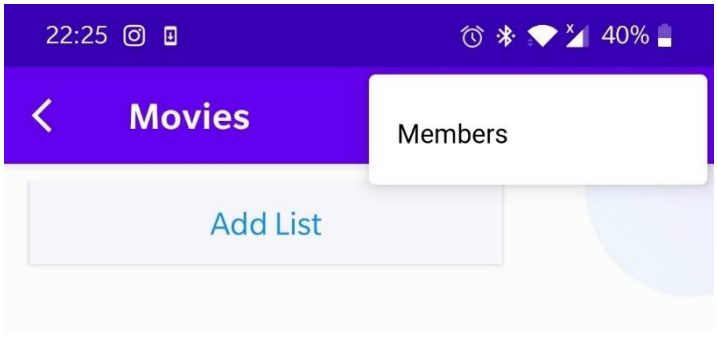
3.7. Izbornik

Izbornici na privremenoj površini prikazuju popis mogućih izbora, a pojavljuju se pri interakciji korisnika s gumbom ili sličnim komponentama. Oni trebaju biti jednostavni za interakciju, a jedna od prednosti im je što se ne ističu te zauzimaju manje prostora od drugih kontrola odabira poput grupe gumbova. U sebi, osim teksta, mogu sadržavati i odgovarajuću ikonu te razdjelnik.



Sl. 3.33. Primjer izbornika

Klikom na gumb s tri točkice sa slike 3.3. otvara se izbornik vidljiv na slici 3.34. implementacije sa slike 3.36. U korištenom zaslonu potrebno je prepisati funkcije onCreateOptionsMenu() u kojoj se napuhuje izgled izbornika te onOptionsItemSelected() u kojoj se definira ponašanje elemenata izbornika.



Sl. 3.34. Primjer izbornika u aplikaciji OrganizeMe

```xml
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto">
    <item
        android:id="@+id/action_members"
        android:orderInCategory="100"
        android:title="Members"
        app:showAsAction="never"
        />
</menu>
```

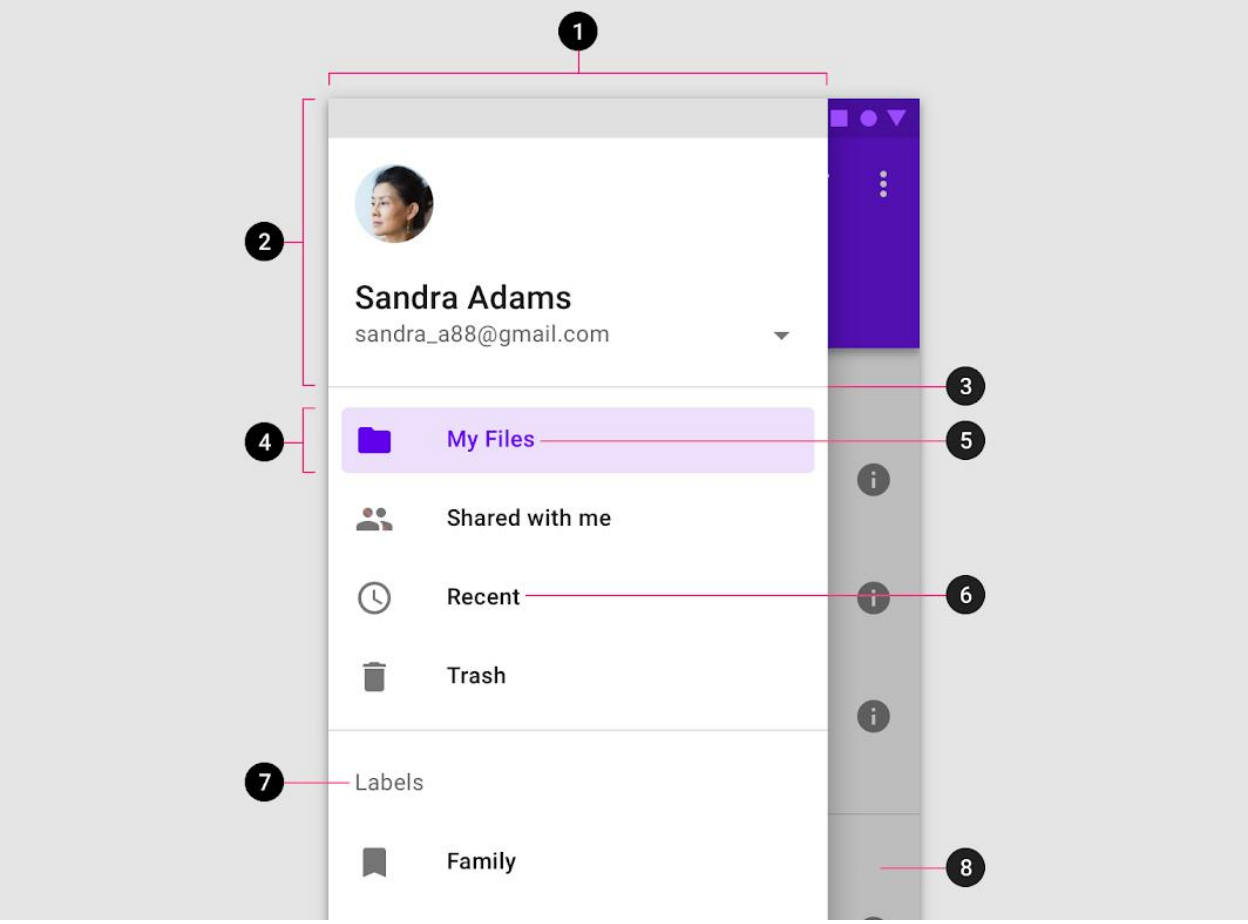
Sl. 3.35. Implementacija izbornika u aplikaciji OrganizeMe - menu_members.xml

```
override fun onCreateOptionsMenu(menu: Menu?): Boolean {
    menuInflater.inflate(R.menu.menu_members,menu);
    return super.onCreateOptionsMenu(menu)
}

override fun onOptionsItemSelected(item: MenuItem): Boolean {
    when(item.itemId){
        R.id.action_members ->{
            val intent = Intent( packageContext: this, MembersActivity::class.java);
            intent.putExtra(Constants.BOARD_DETAIL, mBoardDetails);
            startActivityForResult(intent, MEMBERS_REQUEST_CODE);
            return true;
        }
    }

    return super.onOptionsItemSelected(item)
}
```

Sl. 3.36. Programska implementacija izbornika u aplikaciji OrganizeMe

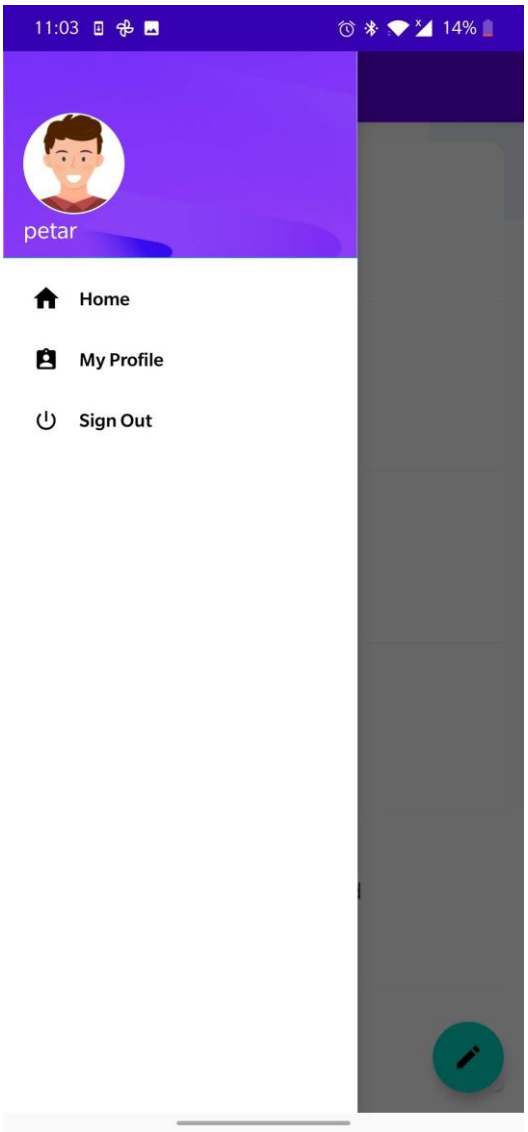
3.8. Navigacijska ladica

Navigacijska ladica pruža pristup glavnim odredištima aplikacije. Može na ekranu biti trajno ili se kontrolirati pomoću navigacijske ikone. Navigacijska ladica treba biti jasno organizirana na način da položaj elemenata i njihovo grupiranje daje do znanja kako se pomoću njih pristupa željenom odredištu. Odredišta uz tekst mogu biti prikazana i odgovarajućom ikonom. Također, ladice mogu imati i zaglavlje na kojem se prikazuju ili korisničke informacije ili informacije o aplikaciji ili brendu.



Sl. 3.37. Navigacijska ladica: 1-spremnik; 2- proizvoljno zaglavlje; 3-proizvoljni razdjelnik; 4,5-aktivni element i njegov tekst; 6-neaktivan element, 7-podnaslov; 8-pozadina

Kao glavni oblik navigacije ladica je korištena i u aplikaciji OrganizeMe, a pristupa joj se pomoću navigacijske ikone vidljive na slici 3.3. ili potezom od ruba zaslona udesno. Navigacijska ladica sastoji se od zaglavlja u kojem je prikazana korisnikova profilna slika te njegovo korisničko ime. Zaglavlje ima i pozadinu te je tankim razdjelnikom odvojeno od navigacijskih elemenata koji označavaju glavne destinacije i radnje u aplikaciji, a to su odlazak na početni zaslon, odlazak na zaslon o korisnikovom profilu te odjavu korisnika.

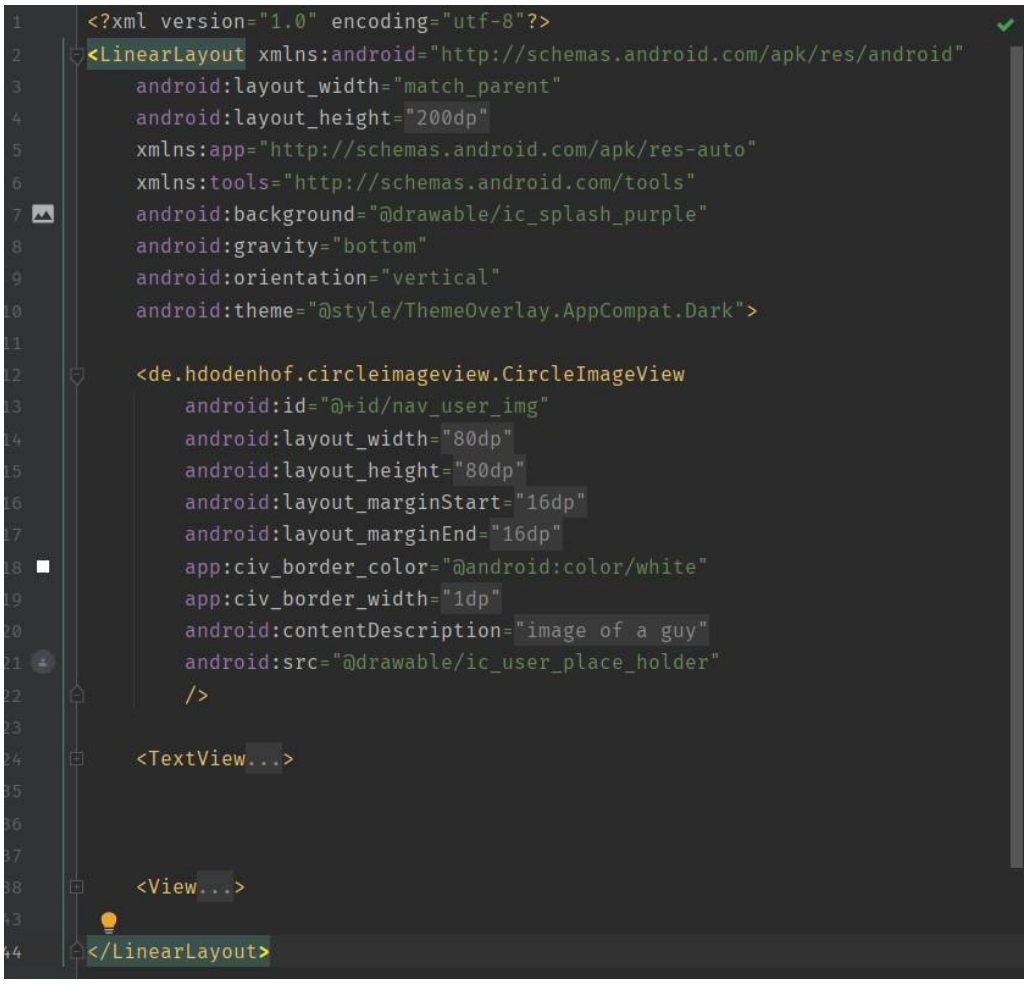


Sl. 3.38. Primjer navigacijske ladice u aplikaciji OrganizeMe

Navigacijska ladica implementirana je na način da se XML datoteka glavnog zaslona ponaša poput ladice, odnosno korišten je DrawerLayout, a ostali sadržaj uključen je u nju. Sama ladica podijeljena je na dva dijela, zaglavlje i NavigationView.

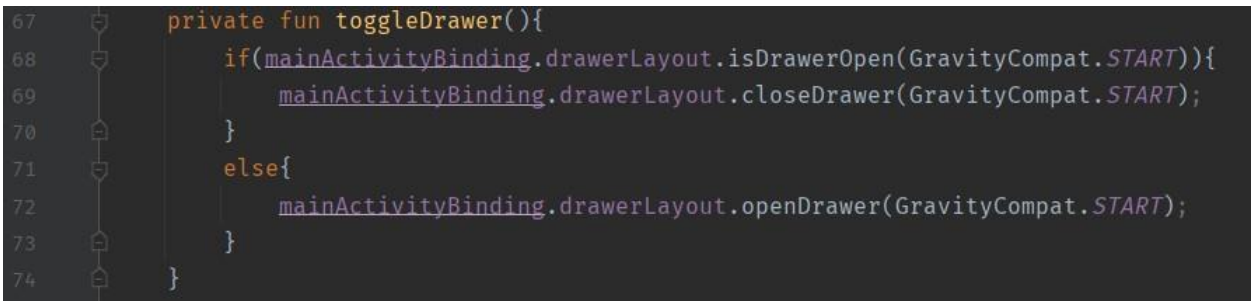
```
<androidx.drawerlayout.widget.DrawerLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/drawer_layout"
    android:layout_height="match_parent"
    android:layout_width="match_parent"
    android:fitsSystemWindows="true"
    tools:openDrawer="start"
    >
    <include
        android:id="@+id/appBarMain_included"
        layout="@layout/app_bar_main"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        />


    <com.google.android.material.navigation.NavigationView
        android:id="@+id/nav_view"
        android:layout_width="wrap_content"
        android:layout_height="match_parent"
        android:layout_gravity="start"
        android:fitsSystemWindows="true"
        app:headerLayout="@layout/nav_header_main"
        app:menu="@menu/activity_main_drawer"
        />


</androidx.drawerlayout.widget.DrawerLayout>
```

Sl. 3.39. Korištenje DrawerLayouta u activity_main.xml



Sl. 3.40. Zaglavlje ladice, datoteka nav_header_main.xml

Funkcija toggleDrawer vidljiva na slici 3.41. otvara i zatvara navigacijsku ladicu.



Sl. 3.41. Implementacija funkcije toggleDrawer()

Funkcija updateNavigationUserDetails() ažurira korisničko ime i sliku u zaglavlju dok funkcija onNavigationItemSelected() implementira prebacivanje na potrebne zaslone.

```
fun updateNavigationUserDetails(loggedInUser: com.plenart.organizeme.models.User, readBoardsList: Boolean) {

    val nav_user_img: CircleImageView = findViewById(R.id.nav_user_img);
    val tv_username: TextView = findViewById(R.id.tv_username);

    mUserName = loggedInUser.name;

    Glide.with( activity: this)
        .load(loggedInUser.image)
        .centerCrop()
        .placeholder(R.drawable.ic_user_place_holder)
        .into(nav_user_img);

    tv_username.text = loggedInUser.name;

    if(readBoardsList){
        showProgressDialog("Please wait");
        Firestore().getBoardsList( activity: this);
    }

}
```

Sl. 3.42. Implementacija funkcije updateNavigationUserDetails()

```
override fun onNavigationItemSelected(item: MenuItem): Boolean {
    when(item.itemId){
        R.id.nav_home ->{
            toggleDrawer();
        }
        R.id.nav_my_profile -> {
            startActivityForResult(Intent( packageContext: this, MyProfileActivity::class.java),
                MY_PROFILE_REQUEST_CODE);
        }
        R.id.nav_sign_out ->{
            FirebaseAuth.getInstance().signOut();
            val intent = Intent( packageContext: this, IntroActivity::class.java);
            intent.addFlags( flags: Intent.FLAG_ACTIVITY_CLEAR_TOP or Intent.FLAG_ACTIVITY_NEW_TASK);
            startActivity(intent);
            finish();
        }

    }
        mainActivityBinding.drawerLayout.closeDrawer(GravityCompat.START);

    return true;
}
```

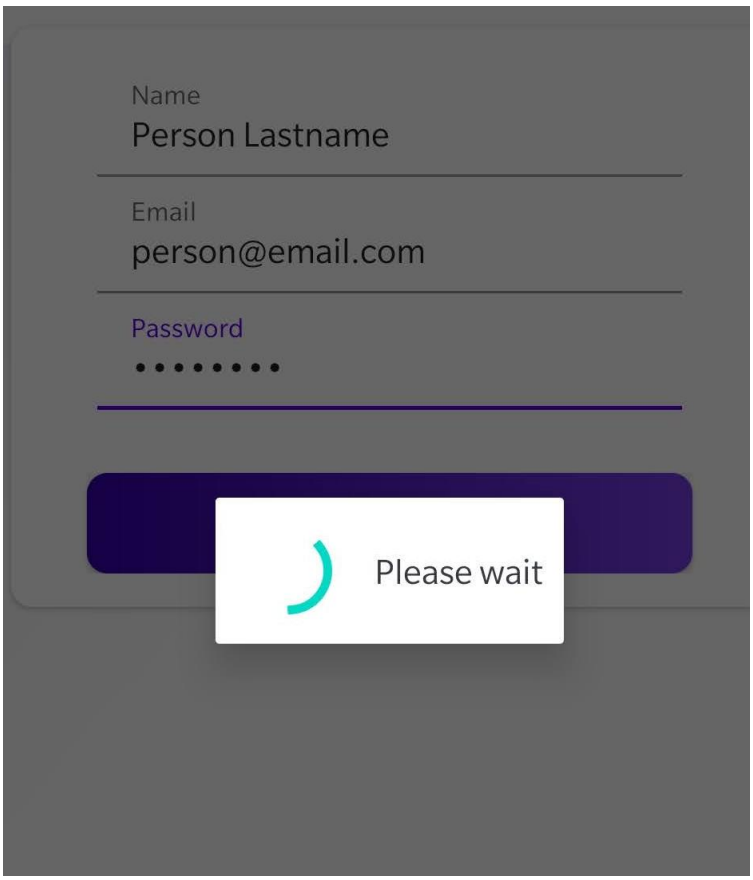
Sl. 3.43. Implementacija funkcije onNavigationItemSelected()

3.9. Pokazatelj napretka

Pokazatelj napretka informira korisnika o statusu procesa koji se trenutno izvode u pozadini poput učitavanja aplikacije, ažuriranja i slično. Oni su animirani kako bi privukli pozornost, ali nikada ne smiju biti čisto dekorativni.

Postoje dvije vrste pokazatelja, a to su linearni i kružni. Isto tako, pokazatelji napretka mogu biti određeni i neodređeni, gdje se potonji koriste kada nije potrebno naznačiti trajanje postupka. S druge strane, određeni pokazatelji napretka prikazuju stopu završetka procesa.

U aplikaciji OrganizeMe prikazivan je kružni neodređeni pokazatelj napretka u dijaloškom okviru. Korišten je na više mjesta aplikacije poput za vrijeme registriranja ili prijave korisnika, za vrijeme učitavanja ploča ili popisa zadataka na zaslon te nakon premještanja pozicije kartica zadataka.



Sl. 3.44. Prikaz neodređenog kružnog pokazatelja napretka

Funkcija showProgressDialog() vidljiva na slici 3.45. stvara dijaloški okvir, napuhuje XML datoteku sa slike 3.47. i postavlja potreban tekst. Funkcija hideProgressDialog() uklanja dijaloški okvir. Na slici 3.46. vidljivi su pozivi navedenih funkcija pri učitavanju ploča na ekran.

```
fun showProgressDialog(text: String){
    mProgressDialog = Dialog( context: this);
    dialogProgressBinding = DialogProgressBinding.inflate(layoutInflater)
    mProgressDialog.setContentView(dialogProgressBinding.root);

    dialogProgressBinding.tvProgressText.text = text;

    mProgressDialog.show();
}

fun hideProgressDialog(){
    mProgressDialog.dismiss();
}
```

Sl. 3.45. Implementacija funkcija showProgressDialog() i hideProgressDialog()

```
createBoardBinding.btnCreateCreateBoardActivity.setOnClickListener{ it: View!
    Log.i( tag: "create brd btn", msg: "click create brd btn");
    if(mSelectedImageFileUri != null){
        uploadBoardImage();
    }
    else{
        showProgressDialog("Please wait");
        createBoard();
    }

}

}

fun boardCreatedSuccessfully(){
    hideProgressDialog();
    setResult(Activity.RESULT_OK);
    finish();
}
```

Sl. 3.46. Implementacija neodređenog kružnog pokazatelja napretka

```
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:gravity="center"
    android:orientation="horizontal"
    android:padding="10dp">

    <ProgressBar
        android:id="@+id/progressBar"
        android:layout_width="50dp"
        android:layout_height="50dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toStartOf="@+id/tv_progress_text"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        />

    <TextView
        android:id="@+id/tv_progress_text"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="16dp"
        android:text="Please wait"
        android:textColor="@color/primary_text_color"
        android:textSize="16sp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintStart_toEndOf="@+id/progressBar"
        app:layout_constraintTop_toTopOf="parent"
        />
```

Sl. 3.47. Implementacija neodređenog kružnog pokazatelja napretka u XML datoteci

3.10. Snackbar

Snackbar se privremeno pojavljuje na dnu zaslona kako bi informirao korisnika o postupku kojeg je aplikacija izvršila ili će ga izvršiti. Ne smiju narušavati korisničko iskustvo i nije potrebna interakcija korisnika kako bi se on sa zaslona povukao. Mogu sadržavati jednu radnju koja će ponovno pokrenuti postupak ili koja će maknuti snackbar sa zaslona.

Sl. 3.48. Snackbar: 1-tekstualna oznaka; 2-spremnik; 3-gumb s radnjom

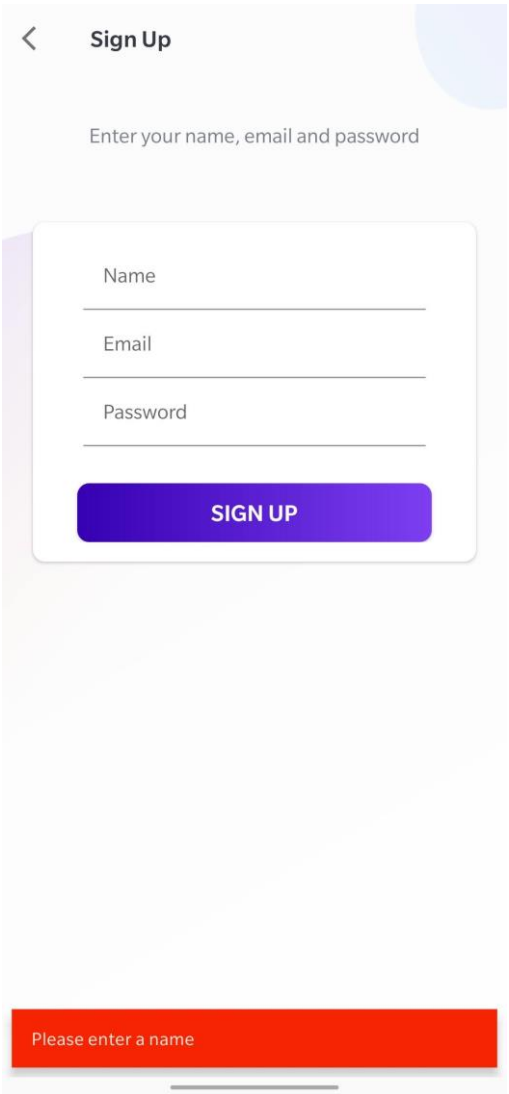
Navedena komponenta korištena je u aplikaciji kako bi informirala korisnika o neuspjelom postupku registracije ili prijave zbog praznog ili krivo popunjenog tekstualnog polja. Koristi crvenu boju kako bi ukazala korisniku na pogrešku pri izvođenju radnje. Funkcija showErrorSnackBar() vidljiva na slici 3.49. stvara snackbar instancu, postavlja boju te vidljivost. Na slici 3.50. vidimo primjer poziva navedene funkcije.

```
61    fun showErrorSnackBar(message: String){
62        val snackBar = Snackbar.make(findViewById(android.R.id.content),message,Snackbar.LENGTH_LONG);
63        val snackBarView = snackBar.view;
64        snackBarView.setBackgroundColor(ContextCompat.getColor( context: this,R.color.snackbar_error_color));
65        snackBar.show();
66
67    }
```

Sl. 3.49. Implementacija snackbara u funkciji showErrorSnackBar()

```
73      private fun validateForm(email: String,password: String): Boolean {
74          return when{
75              TextUtils.isEmpty(email)->{
76                  showErrorSnackBar("Please enter email");
77                  false;
78              }
79              TextUtils.isEmpty(password)->{
80                  showErrorSnackBar("Please enter a password");
81                  false;
82              }
83              else -> {true}
84          }
85
86      }
```

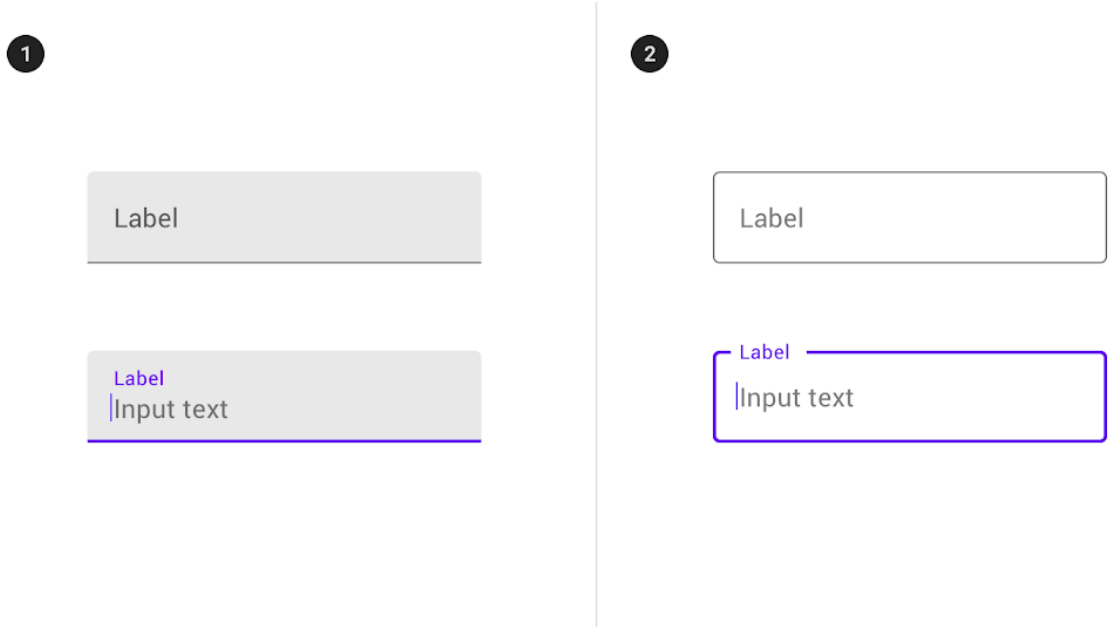
Sl. 3.50. Poziv funkcije showErrorSnackBar() u kodu



Sl. 3.51. Primjer snackabara u aplikaciji OrganizeMe

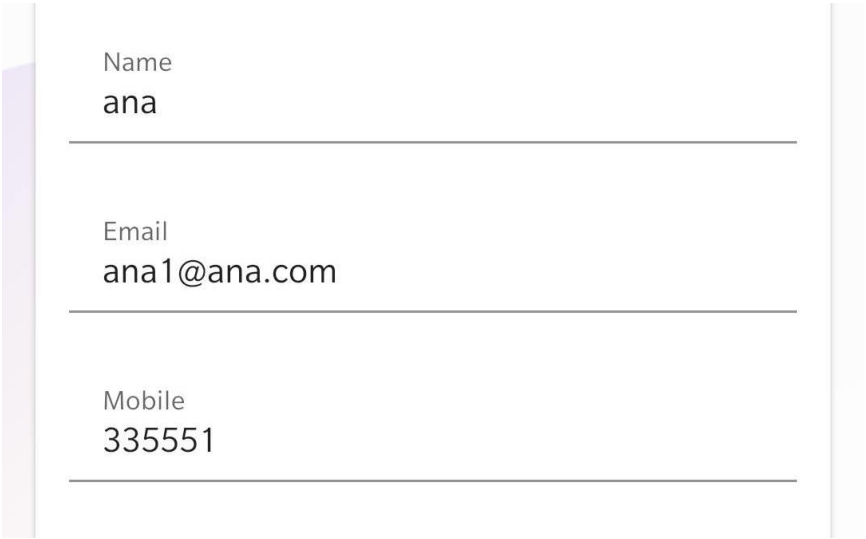
3.11. Tekstualno polje

Kako im i samo ime govori, tekstualna polja omogućuju korisnicima unos teksta u korisničko sučelje koje se prenosi u pozadinske radnje. Najčešće se pojavljuju u obliku obrazaca ili dijaloških okvira, a trebali bi biti istaknuti i ukazati korisniku da se u njih mogu unijeti informacije. Postoji dva tipa tekstualnih polja, a to su ispunjena i ocrtana tekstualna polja. Obje vrste se nalaze u spremniku koji naglašava mogućnost interakcije.



Sl. 3.52. Tekstualna polja: 1-ispunjena tekstualna polja; 2-ocrtana tekstualna polja

Tekstualna polja korištena su u OrganizeMe aplikaciji na svim mjestima gdje je potreban korisnikov unos podataka poput podataka za registraciju korisnika, njegovu prijavu, popunjavanje informacija o korisničkom profilu i slično. Odabrana su ispunjena tekstualna polja, ali s prozirnom pozadinom iz razloga što bi ocrtana tekstualna polja svojim obrisom narušavala izgled kartice na kojoj se nalaze.



Sl. 3.53. Primjer korištenih tekstualnih polja u aplikaciji OrganizeMe

```
66          <com.google.android.material.textfield.TextInputLayout
67              android:layout_width="match_parent"
68              android:layout_height="wrap_content"
69              android:layout_marginStart="25dp"
70              android:layout_marginEnd="25dp"
71              app:boxBackgroundColor="@android:color/transparent"
72              android:background="@android:color/transparent"
73              >
74
75              <com.google.android.material.textfield.TextInputEditText
76                  android:id="@+id/et_email_sign_in_activity"
77                  android:layout_width="match_parent"
78                  android:layout_height="wrap_content"
79                  android:hint="Email"
80                  android:inputType="textEmailAddress"
81                  android:textSize="16sp"
82                  />
83          </com.google.android.material.textfield.TextInputLayout>
```

Sl. 3.54. XML implementacija tekstualnih polja u aplikaciji OrganizeMe

```
43      private fun singInUser(){
44
45          auth = FirebaseAuth.getInstance();
46
47          val email: String = binding.etEmailSignInActivity.text.toString().trim();
48          val password: String = binding.etPasswordSignInActivity.text.toString();
```

Sl. 3.55. Primjer dohvaćanja teksta iz tekstualnih polja u aplikaciji OrganizeMe

4. ZAKLJUČAK

Material Design je sustav osmišljen s ciljem stvaranja dosljednog i responzivnog dizajn jezika neovisnog o platformi. Inspiriran je stvarnim svijetom, točnije, interakcijom tinte i papira. Elementi korisničkog sučelja izrađeni su od zamišljenog materijala koji slijedi pravila fizike. Imaju razne oblike, uzvišenje očitovano sjenama te boje odražavaju brend, a ujedno su i dijelom većeg sustava Material Design-a. Komponente navedenog sustava dostupne su na više platformi, a čine gradivne blokove od kojih se sastoji korisničko sučelje. Komponente rješavaju određene probleme poput prikaza ili navigacije. Ideja ovog završnog rada je izrada Android aplikacije koristeći smjernice i komponente Material Design sustava. U aplikaciji je iskorišteno jedanaest od dvadesetak dostupnih komponenti za Android sustav. Iskorištene komponente uključuju gornju aplikacijsku traku, gumbove, navigacijsku ladicu te tekstualna polja kao jedne od često korištenih komponenata u brojnim aplikacijama. Korištene su još i kartice, birači datuma i snackbar kao komponente za posebnu namjenu. Sve korišteno u budućnosti se može dodatno proširiti, kako u funkcionalnosti, tako i novim Material Design elementima poput animacija koje zbog prirode završnog rada nisu korištene.

LITERATURA

[1] Material Design [online], dostupno na: https://material.io/design/introduction#principles [12.6.2021.]

[2] Material Design [online], dostupno na: https://material.io/design/environment [12.6.2021.]

[3] Material Design [online], dostupno na: https://material.io/design/layout [13.6.2021.]

[4] Material Design [online], dostupno na: https://material.io/design/navigation [14.6.2021.]

[5] Material Design [online], dostupno na: https://material.io/design/color [14.6.2021.]

[6] Material Design [online], dostupno na: https://material.io/design/typography [14.6.2021.]

[7] Material Design [online], dostupno na: https://material.io/design/iconography [14.6.2021.]

[8] Material Design komponente [online], dostupno na: https://material.io/components?platform=android [16.6.2021.]

[9] Material Design komponente [online], dostupno na: https://material.io/components/app-bars-top [16.6.2021.]

[10] Material Design komponente [online], dostupno na: https://material.io/components/buttons [16.6.2021.]

[11] Material Design komponente [online], dostupno na: https://material.io/components/buttons-floating-action-button [17.6.2021.]

[12] Material Design komponente [online], dostupno na: https://material.io/components/cards [17.6.2021.]

[13] Material Design komponente [online], dostupno na: https://material.io/components/date-pickers [17.6.2021.]

[14] Material Design komponente [online], dostupno na: https://material.io/components/dialogs [18.6.2021.]

[15] Material Design komponente [online], dostupno na:

https://material.io/components/menus [18.6.2021.]

[16] Material Design komponente [online], dostupno na: https://material.io/components/navigation-drawer [19.6.2021.]

[17] Material Design komponente [online], dostupno na: https://material.io/components/progress-indicators [21.6.2021.]

[18] Material Design komponente [online], dostupno na: https://material.io/components/snackbars [22.6.2021.]

[19] Material Design komponente [online], dostupno na: https://material.io/components/text-fields [23.6.2021.]

SAŽETAK

U završnom radu opisan je Material Design standard, njegove smjernice i komponente. Izrađena je Android aplikacija koristeći dostupne komponente za navedeni operacijski sustav. Prikazana je njihova primjena u dizajnu korisničkog sučelja te implementacija u programskom jeziku Kotlin.

Ključne riječi: Android, komponente, Material Design

ABSTRACT

Title: Application of Material Design in user interface design

The goal of this final paper was an application of Material Design and it's components. An android application was developed using the available Material Design components for Android. Their application in user interface and implementation in Kotlin programming language are presented.

Key words: Android, components, Material Design

ŽIVOTOPIS

Petar Lenart rođen je u Osijeku 9.6.1999. Osnovnu školu završio je u Orahovici 2014. godine. Nakon osnovne škole upisuje Opću gimnaziju u Orahovici. Nakon završene srednje škole, upisuje se na preddiplomski studij Računarstva na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija u Osijeku.

PRILOZI

Na priloženom optičkom disku nalaze se .doc i .pdf verzija završnog rada te kôd aplikacije.