

UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Domen Koščak

Mobilna aplikacija za pomoč pri nakupovanju

DIPLOMSKO DELO

VISOKOŠOLSKI STROKOVNI ŠTUDIJSKI PROGRAM
PRVE STOPNJE
RAČUNALNIŠTVO IN INFORMATIKA

MENTOR: prof. dr. Rok Rupnik

Ljubljana, 2023

To delo je ponujeno pod licenco *Creative Commons Priznanje avtorstva-Deljenje pod enakimi pogoji 2.5 Slovenija* (ali novejšo različico). To pomeni, da se tako besedilo, slike, grafi in druge sestavine dela kot tudi rezultati diplomskega dela lahko prosto distribuirajo, reproducirajo, uporabljajo, priobčujejo javnosti in predelujejo, pod pogojem, da se jasno in vidno navede avtorja in naslov tega dela in da se v primeru spremembe, preoblikovanja ali uporabe tega dela v svojem delu, lahko distribuira predelava le pod licenco, ki je enaka tej. Podrobnosti licence so dostopne na spletni strani creativecommons.si ali na Inštitutu za intelektualno lastnino, Streliška 1, 1000 Ljubljana.



Izvorna koda diplomskega dela, njeni rezultati in v ta namen razvita programska oprema je ponujena pod licenco GNU General Public License, različica 3 (ali novejša). To pomeni, da se lahko prosto distribuira in/ali predeluje pod njenimi pogoji. Podrobnosti licence so dostopne na spletni strani <http://www.gnu.org/licenses/>.

Besedilo je oblikovano z urejevalnikom besedil L^AT_EX.

Kandidat: Domen Koščak

Naslov: Mobilna aplikacija za pomoč pri nakupovanju

Vrsta naloge: Diplomaska naloga na visokošolskem programu prve stopnje
Računalništvo in informatika

Mentor: prof. dr. Rok Rupnik

Opis:

Zasnуйте mobilno aplikacijo, ki omogoča obvladovanje seznamov za pomoč pri nakupovanju. Aplikacija naj omogoča tudi deljenje seznamov. Nato aplikacijo razvijte z uporabo tehnologij po lastnem izboru.

Title: Mobile Application to Support Shopping

Description:

Design a mobile app to manage lists to help with shopping. The application should also allow sharing of lists. Then develop the application using technologies of your choice.

Zahvalil bi se svoji družini in prijateljem, da so me skozi celoten študij spodbujali in bodrili ter mentorju dr. Roku Rupniku za odlično mentorstvo.

Kazalo

Povzetek

Abstract

| | | |
|----------|---------------------------------------|----------|
| 1 | Uvod | 1 |
| 1.1 | Predstavitev ideje | 1 |
| 1.2 | Cilj diplomske naloge | 1 |
| 1.3 | Struktura diplomske naloge | 2 |
| 2 | Tehnologije, orodja in ogrodja | 3 |
| 2.1 | Flutter | 3 |
| 2.1.1 | Zgodovina | 3 |
| 2.1.2 | Projekti | 4 |
| 2.1.3 | Programska orodja | 5 |
| 2.2 | Python | 5 |
| 2.2.1 | Zgodovina | 5 |
| 2.2.2 | Uporabljene knjižnice | 6 |
| 2.2.3 | Projekti | 8 |
| 2.2.4 | Programska orodja | 8 |
| 2.3 | MariaDB | 8 |
| 2.3.1 | Zgodovina | 8 |
| 2.3.2 | Projekti | 9 |
| 2.3.3 | Programska orodja | 9 |

| | | |
|----------|--|-----------|
| 3 | Analiza | 11 |
| 3.1 | Diagram primerov uporabe | 11 |
| 3.2 | Podatkovni model | 14 |
| 3.2.1 | Uporabniki | 15 |
| 3.2.2 | Družina | 15 |
| 3.2.3 | Izdelek | 16 |
| 3.2.4 | Seznam | 16 |
| 3.2.5 | Povezovalne tabele | 17 |
| 3.2.6 | Uporabniška gesla | 17 |
| 4 | Načrtovanje | 19 |
| 4.1 | Izbira arhitekture | 19 |
| 4.2 | Uporabniški vmesnik | 20 |
| 4.3 | Direktorijska zgradba aplikacije | 22 |
| 4.4 | Varnost | 24 |
| 5 | Predstavitev aplikacije | 25 |
| 5.1 | Prijava in registracija | 25 |
| 5.2 | Pridružitve družini | 28 |
| 5.3 | Domač zaslon | 29 |
| 5.4 | Nastavitve in dodajanje novih izdelkov | 30 |
| 6 | Zaključek | 33 |
| 6.1 | Nadaljnji razvoj | 33 |
| | Celotna literatura | 35 |

Seznam uporabljenih kratic

| kratica | angleško | slovensko |
|--------------|---------------------------------------|---|
| RDBMS | relational database management system | sistem za upravljanje relacijskih podatkovnih baz |
| API | application programming interface | programski vmesnik |
| SDK | software development kit | oprema za razvoj programske opreme |
| IDE | integrated development environment | integrirano razvojno okolje |
| ORM | object-relational mapping | objektno-relacijsko mapiranje |

Povzetek

Naslov: Mobilna aplikacija za pomoč pri nakupovanju

Avtor: Domen Koščak

Na trgu je že kar nekaj aplikacij, ki poskušajo rešiti problem pomnjenja izdelkov, ki jih je potrebno kupiti. Obstajajo enostavne, kamor lahko pišemo izdelke kot na preprost list papirja in tudi zahtevnejše, ki omogočajo sortiranje izdelkov po trgovinah in ustvarjanje več seznamov. Želel sem ustvariti svojo, ki bi imela neko dodano vrednost v primerjavi z ostalimi. Aplikacija *Koopy* je narejena za uporabnike mobilnih naprav z operacijskim sistemom iOS in Android, saj tako doseže praktično vse uporabnike. Omogoča jim shranjevanje izdelkov v seznamih, deljenje seznamov z ostalimi uporabniki in ustvarjanje skupin uporabnikov imenovanih družina. Razvita je bila z ogrodjem Flutter v urejevalniku Visual Studio Code. Uporablja podatkovno bazo MariaDB, kjer so shranjeni vsi podatki za delovanje aplikacije.

Ključne besede: računalnik, aplikacija, flutter, mobilni, telefon.

Abstract

Title: Mobile Application to Support Shopping

Author: Domen Koščak

There are a lot of applications on the market that try to solve the problem of remembering what products to buy. There are some simple ones that offer just listing the products like on the paper in physical form as well as more complex ones, where you can sort the products depending in which store you should buy it and offer creating more lists. I wanted to create my own with added functionality against existing ones. Application *Koopy* is made for iOS and Android users, covering almost all mobile phone users. It offers saving products in lists, sharing lists with other users and creating groups of users called families. It was developed using Flutter in Visual Studio Code. It uses MariaDB database where all data is saved.

Keywords: computer, application, flutter, mobile, phone.

Poglavje 1

Uvod

Danes se brez mobilnega telefona ne odpravimo praktično nikamor, za v trgovino je skoraj obvezen pripomoček. Iz lastnih izkušenj vem, da ko se odpravim v trgovino, skoraj vedno pozabim na kakšno stvar. Starejši takšne težave rešujejo z uporabo papirja in svičnika, mlajši se pogosto poslužujejo tekstovnih sporočil.

1.1 Predstavitev ideje

Ker nas mobilni telefon spremlja ves čas, sem pomislil, zakaj ga ne bi aktivno uporabljali tudi v trgovini. Tako sem prišel do ideje, da bi lahko naredil aplikacijo, ki bi nadomestila nakupovalni listek in tekstovna sporočila. Omogočala bi dodajanje in odstranjevanje stvari iz seznamov, imela bi več seznamov za različne trgovine in bi omogočala deliljenje seznamov z ostalimi uporabniki.

1.2 Cilj diplomske naloge

Cilj diplomske naloge je mobilna aplikacija za mobilne naprave z operacijskim sistemom Android in iOS, ki bo olajšala nakupovanje in izboljšala nakupovalno izkušnjo. Aplikacija bo podprta s strežnikom v ozadju, ki bo

skrbel za shranjevanje podatkov in medsebojno delovanje znotraj aplikacije.

1.3 Struktura diplomske naloge

Diplomska naloga je sestavljena iz uvoda, treh večjih poglavij in zaključka. V prvem poglavju so predstavljene uporabljene tehnologije in orodja, ki sestavljajo aplikacijo. V drugem poglavju je predstavljena podatkovna zbirka in orodje za njeno uporabljanje, v tretjem pa je predstavljena sama aplikacija in njeno delovanje. V zaključku so končne ugotovitve in ideje ter izboljšave za nadaljni razvoj.

Poglavje 2

Tehnologije, orodja in ogrodja

V tem poglavju so predstavljene uporabljene tehnologije in orodja, ki so omogočile razvoj mobilne aplikacije. Na kratko je predstavljena zgodovina, nekateri od projektov in podprta programska orodja, ki so ustvarjeni s to tehnologijo ali orodjem.

2.1 Flutter

Flutter[3] je ogrodje za razvoj programske opreme (SDK), ki omogoča razvijanje aplikacij za Android, iOS, Windows, macOS, Linux in pa spletne brskalnike z uporabo ene programske kode, brez potrebe po pisanju svoje aplikacije za vsakega izmed naštetih operacijskih sistemov.

Za programiranje se uporablja programski jezik Dart, ki ga je razvil Google in je zelo podoben programskemu jeziku Java, s par razlikami.

2.1.1 Zgodovina

Prva različica Flutterja je bila predstavljena leta 2015 na Dartovi konferenci in je podpirala zgolj operacijski sistem Android. Poudarek je takrat bil na hitrem prikazovanju na ekranu (120 sličic na sekundo).

Naslednja pomembnejša različica je bila predstavljena leta 2018 na Google-ovih dneh razvijalcev (Google Developer Days) v Shanghaju, kjer so predstavili predogled nove verzije, ki so jo konec leta spustili v javnost. To je bila tudi prva stabilna različica ogrodja Flutter.

Leta 2020 je prišla verzija 1.17.0 ki je vključevala podporo za Metal API, ki je izboljšal izvajanje na napravah z operacijskim sistemom iOS za 50%.

Flutter verzija 2 je bila predstavljena leta 2021, ki je vsebovala podporo za razvijanje aplikacij za spletne brskalnike. Poleg brskalnikov je vsebovala zgodnji dostop do ustvarjanja aplikacij za Windows, Linux in macOS operacijske sisteme. Dodali so "null-safety" funkcionalnost, ki je povzročila veliko težav z zunanjimi paketi.

Konec leta 2021 je Google izdal različico 2.5 ki je prinesla podporo za zadnjo verzijo Google-ovega Material dizajna. Prav tako je bila podpora za Apple Silicon naprave označena kot stabilna.

Trenutno stabilna zadnja različica je Flutter 3.0.2.

2.1.2 Projekti

Flutter postaja vedno bolj popularen in uporabljan. Veliko aplikacij je že narejenih z njegovo uporabo in to so nekatere izmed njih.

- BMW aplikacija
- Google Pay
- eBay aplikacija
- iRobot aplikacija

2.1.3 Programska orodja

Flutter je ogrodje za razvoj programske opreme in ne pride z integriranim razvojnim orodjem. Za to je potrebno uporabiti okolja (IDE), kot so Visual Studio Code, Android Studio, IntelliJ Idea in podobni.

Za uporabo Flutter SDK-ja ga je potrebno prenesti iz njihove spletne strani, ga razpakirati v neko mapo v računalniku in dodati pot do mape med sistemske spremenljivke. V IDEjih nato Flutter dodamo preko vtičnikov, ki omogočajo njegovo uporabo.

2.2 Python

Python[7] je en izmed najpopularnejših programskih jezikov, ki je uporabniku prijazen in zelo podoben človeškemu jeziku. Uporablja se ga praktično povsod, skupnost pa je široka, zato lahko odgovore na vprašanja najdemo povsod in z lahkoto.

2.2.1 Zgodovina

Python je bil prvič predstavljen leta 1980 s strani Guida van Rossuma na Nizozemskem, leta 1989 pa se je pričela njegova implementacija.

Leta 2000 je bila predstavljena različica 2.0, ki je vsebovala veliko novih funkcionalnosti. Čez osem let so izdali Python različico 3.0, katere funkcionalnosti so vključili tudi v verzijah 2.6.x in 2.7.x. Ob izdaji verzije 3.0 so izdali tudi orodje "2to3", ki je omogočalo prehod iz stare verzije na novo.

Leta 2015 je bila predvidena ukinitvev Python verzije 2.7, ki pa se je prestavila na leto 2020, saj so se pojavile skrbi glede migracije obstoječe kode veliko projektov na novo verzijo. Od ukinitve verzije 2.7 so uradno podprte samo še verzije od 3.6 dalje.

Leta 2021 so bile izdane verzije 3.8.8 in 3.9.2, za starejše verzije pa so ugotovili, da imajo varnostna tveganja, ki omogočajo oddaljeno izvajanje ukazov in okuževanje podatkov v predpomnilniku.

Trenutno zadnja stabilna verzija je 3.10.5, verzija 3.9.13 pa je zadna verzija Pythona 3.9, ki bo prejela samo še varnostne popravke.

2.2.2 Uporabljene knjižnice

V Pythonu so zelo popularne knjižnice, ki omogočajo lažji razvoj programov. Jaz sem uporabil knjižnice Flask, Peewee in Flasgger.

Flask

Flask[2] je knjižnica za ustvarjanje spletnega strežnika. Omogoča enostavno in hitro postavitvev strežnika, ki lahko služi kot spletna stran ali API. Uporabil sem ga za oboje, API je potreben za povezavo aplikacije s podatkovno bazo, na spletni strani pa je dokumentacija za API.

Poleg Flaska sem uporabil še Flask Restful knjižnico, ki omogoča ustvarjanje REST končnih točk v APIju. Kratica REST pomeni "Representational State Transfer" in narekuje, kako naj bi API deloval. Da je API označen kot REST mora izpolnjevati naslednje pogoje:

- Promet poteka preko HTTP/HTTPS
- Vsak zahtevek ("request") mora biti individualen in nepovezan z ostalimi
- Odziv ("response") mora vsebovati pomenljivo sporočilo

Peewee

Peewee[6] je knjižnica, ki omogoča interakcijo s podatkovno bazo preko objektno-relacijskega mapiranja (ORM). ORM nam omogoča upravljanje podatkovne baze na način objektno orientiranega programiranja, kjer je vsaka tabela predstavljena s svojim razredom, stolpci pa so razredne spremenljivke.

S pomočjo Peeweeja skrajšamo SQL iskalne poizvedbe, saj jih knjižnica generira sama. Na ta način je koda bolj pregledna in lažje berljiva.

Flasgger

Flasgger[1] je knjižnica, ki omogoča uporabo Swaggerja v Flasku. Flasgger omogoča avtomatsko generiranje API dokumentacije. V datotekah s končnico ".yaml" povemo glavne lastnosti API končnih točk, nato pa knjižnica sama generira spletno stran, z vsemi končnimi točkami, ki smo jih vnesli.

Primer ".yaml" konfiguracijske datoteke izgleda tako:

```
Opis končne točke in kaj počne
---
parameters:
  - in: body
    schema:
      properties:
        id:
          type: integer
responses:
  200:
    description: Uspešno izveden zahtevek
    schema:
      properties:
        message:
          type: string
```

2.2.3 Projekti

Ker je Python med popularnejšimi programskimi jeziki je veliko projektov napisanih v njem. Naštel bom par projektov, ki so razviti z uporabo Python programskega jezika.

- Instagram
- Google
- Spotify
- Netflix

2.2.4 Programska orodja

Python je programski jezik, ki pride v paketu s svojim razvojnim okoljem IDLE. Uporabniki niso omejeni zgolj na njegovo uporabo, ampak lahko uporabljajo poljuben tekstovni urejevalnik. Pomembna je samo končnica datoteke, v katero shranijo svojo programsko kodo, ki mora biti ".py". Datoteko lahko nato zaženemo preko ukazne vrstice.

2.3 MariaDB

MariaDB[5] je sistem za upravljanje relacijskih podatkovnih baz (RDBMS), ki ga je razvila skupnost. Razvoj so v glavnem prevzeli nekateri izmed originalnih razvijalcev MySQL. Mišljen je, da ohrani visoko kompatibilnost z MySQL, tako z ukazi kot s knjižnicami.

2.3.1 Zgodovina

MariaDB je nastala iz MySQLa. Ker je bil MySQL kupljen s strani Sun Microsystems v letu 2008, leta 2010 pa ga je kupil Oracle. Ustanovitelj

Michael Monty Widenius kopiral izvorno kodo MySQLa in ustvaril MariaDB.

Leta 2012 so ustanovili podjetje MariaDB foundation, da bi se izognili enakemu scenariju kot se je zgodil z Oraclom.

2.3.2 Projekti

Mnogi so mnenja, da je MariaDB boljši RDBMS kot Oracle MySQL, zato vedno več podjetij prehaja na MariaDB.

- Google
- Red Hat
- Fedora
- Wikipedia

2.3.3 Programska orodja

MariaDB ne pride s svojim integriranim programskim okoljem, zato moramo zanj poskrbeti sami. Uporabimo lahko različne programe ali pa kar spletne vmesnike.

Programi

Najpogosteje se za urejanje MariaDB podatkovne baze uporablja MySQL Workbench. To je program, ki ga je razvil Oracle in ponuja vizualni urejevalnik, omogoča enostavno migracijo baz ter vsebuje vizualni prikazovalnik optimizacije poizvedb.

Drugi program je DBeaver. To je odprtokodni program, ki podpira več vrst podatkovnih baz, na primer MariaDB, MySQL, PostgreSQL in ostale. Prednosti so uporabniku prijazen vmesnik in poenostavljeno urejanje poizvedb z samodejnim dokončevanjem in formatiranjem. Priročen pa je tudi prikaz vsebine tabele.

Spletni vmesniki

Prvi izmed spletnih vmesnikov je Adminer. Podobno kot DBeaver podpira različne tipe podatkovnih baz, omogoča pa tudi kreiranje popolnoma nove podatkovne baze. Uporabniški vmesnik je intuitiven, brez nepotrebnih ikon ali druge vsebine.

Med uporabniki je priljubljen tudi phpMyAdmin, ki je, kot ime pove, napisan v php programskem jeziku. Uporabniku omogoča iskanje po podatkovnih bazah, ima pregleden vmesnik z možnostjo spreminjanja tem in omogoča označevanje SQL poizvedb.

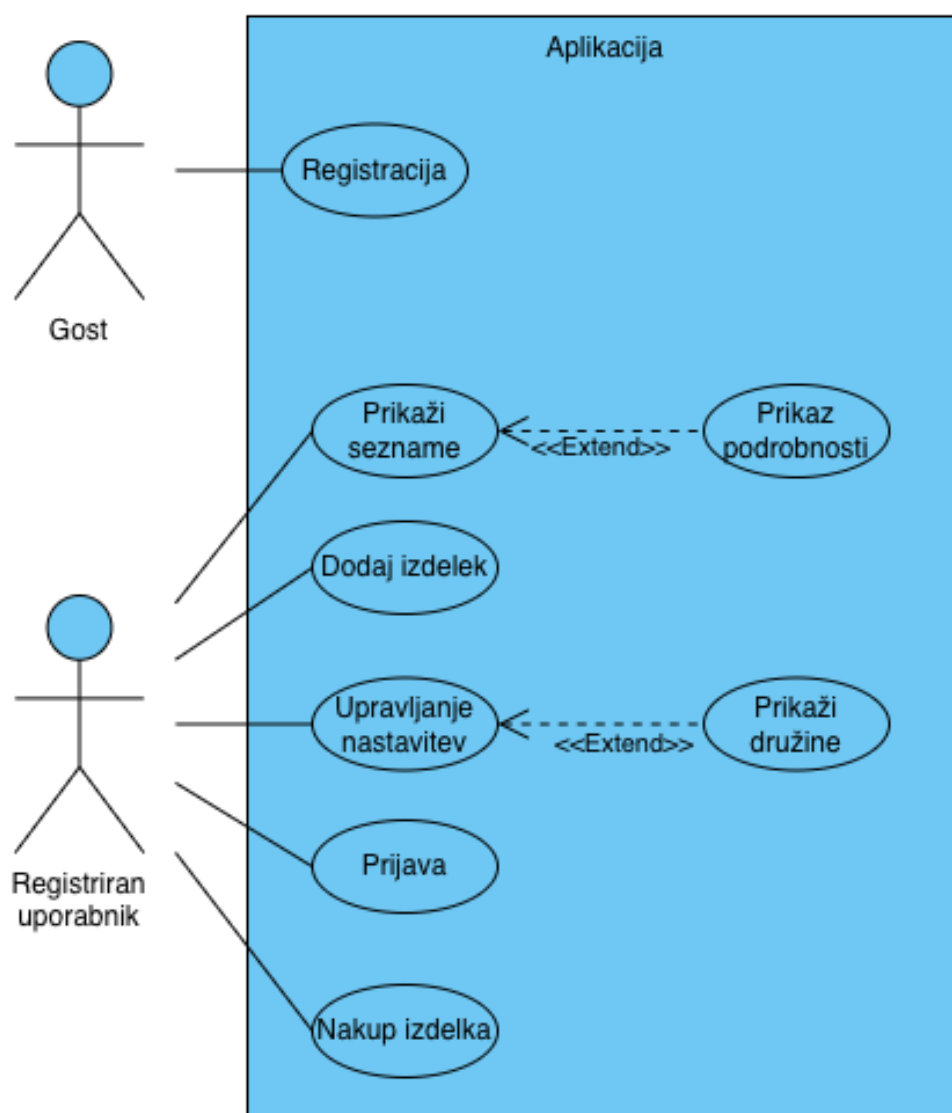
Poglavje 3

Analiza

3.1 Diagram primerov uporabe

Za lažje določanje funkcionalnosti, ki jih bo aplikacija vsebovala sem ustvaril diagram primerov uporabe, ki je predstavljen spodaj. Predstavljeni so primeri uporabe, ki so opisani v spodnji tabeli.

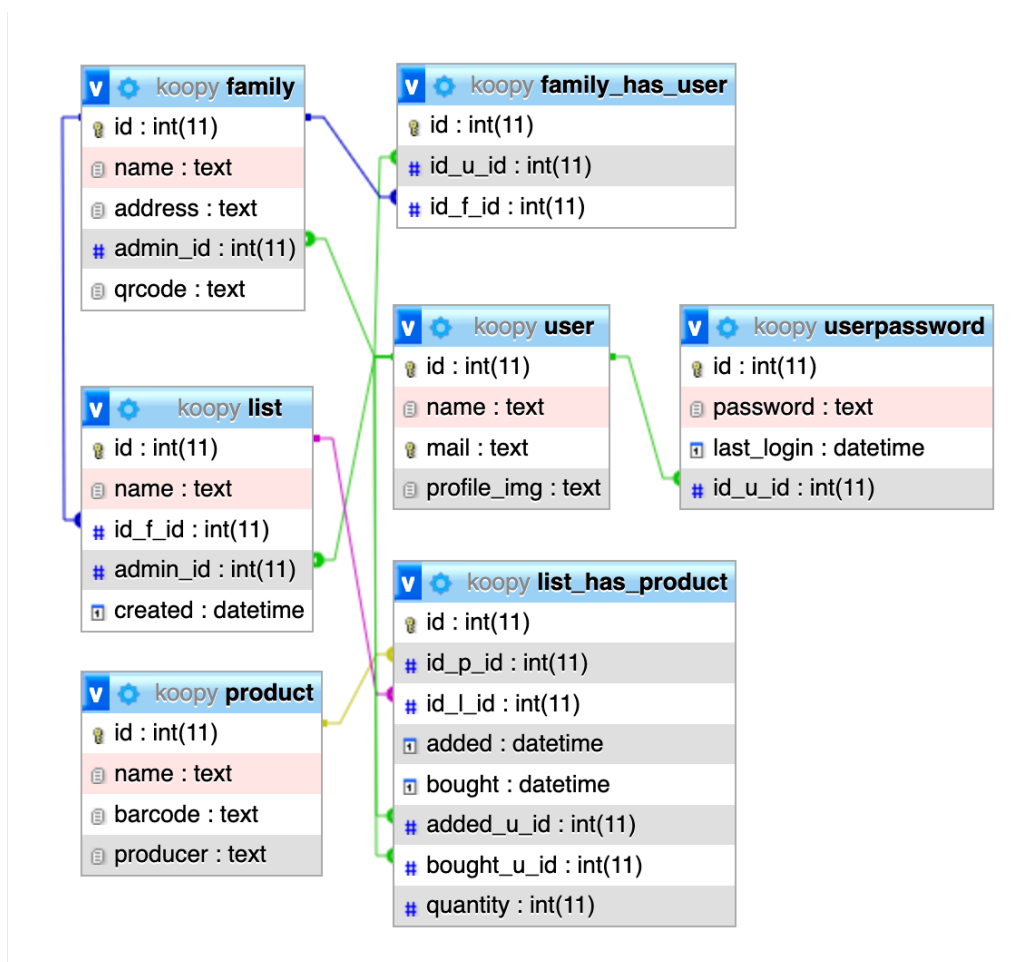
| Primer uporabe | Opis |
|-----------------------|--|
| Prijava | Uporabnik se prijavi in lahko dostopa do vsebine aplikacije |
| Registracija | Uporabnik se registrira in lahko dostopa do vsebine aplikacije |
| Prikaz seznamov | Uporabniku se prikažejo ustvarjeni seznam |
| Prikaz družin | Uporabniku se prikaže seznam družin, katerih je član in med katerimi lahko izbira |
| Prikaz nastavitev | Uporabniku se prikažejo nastavitve, kjer lahko ustvarja družine in se jim pridružuje |
| Prikaz podrobnosti | Uporabniku se prikažejo podrobnosti izdelka na seznamu |
| Nakup izdelka | Uporabnik označi izdelek kot kupljen |
| Dodajanje izdelka | Uporabnik na seznam doda izdelek |
| Brisanje seznama | Uporabnik izbriše seznam |
| Brisanje izdelka | Uporabnik izbriše izdelek iz seznama |



Slika 3.1: Diagram primerov uporabe

3.2 Podatkovni model

Podatkovna baza je sestavljena iz več entitet, ki so med seboj povezane z uporabo tujih ključev ali vmesnih entitet. Glavne so *user*, *family*, *product* in *list*. V nadaljevanju bom predstavil vsako od entitet in njihove atribute. Na spodnji sliki je konceptualni model z vsemi entitetami in njihovimi atributi.



Slika 3.2: Konceptualni model podatkovne baze

3.2.1 Uporabniki

V entiteti *user* so shranjeni podatki o uporabnikih. Ko se uporabnik prvič prijavi oziroma registrira se zanj ustvari zapis v tej entiteti. Sestavljena je iz štirih atributov. To so *id*, *name*, *mail* in *profile_img*.

Atribut *id* predstavlja enolični identifikator za vsakega uporabnika; tako jih lahko ločimo in dobimo samo enega, če to potrebujemo in ga uporabimo kot tuji ključ za povezovanje tabel. *id* se samodejno generira ob registraciji. Atributa *name* in *mail* uporabnik zapolni s svojim imenom in priimkom ter svojim e-poštnim naslovom, ki mora biti edinstven; ni se mogoče registrirati dvakrat z istim e-poštnim naslovom. Atribut *profile_img* pa je atribut, v katerem je shranjena pot do profilne slike, ki jo uporabniku naključno določi aplikacija ob registraciji.

↩

T

→

▼

id

name

mail

profile_img

□

✎

Edit

📄

Copy

🗑

Delete

1

Domen Koščak

domen.koscak@gmail.com

pig.jpg

Slika 3.3: Primer entitete *user*

3.2.2 Družina

V entiteti *family* so shranjeni podatki o družinah oziroma skupinah uporabnikov. Sestavljena je iz atributov *id*, *name*, *address*, *admin_id* in *qrcode*.

Ponovno se pojavi atribut *id*, ki predstavlja enako kot v entiteti *user*. V atributu *name* uporabnik določi ime družine, kateri v atributu *address* pove tudi naslov. Na enem naslovu je lahko samo ena družina z imenom: na istem naslovu ne moreta biti dve družini z enakim imenom. V entiteti je tuji ključ *admin_id*, ki predstavlja uporabnika, ki je družino ustvaril in je edini, ki jo lahko zbriše. Zadnji atribut, *qrcode*, je atribut, v katerem je shranjeno enolično besedilo, ki se v aplikaciji pretvori v qr kodo in olajša dodajanje članov v družino.

| ← T → | | | | | |
|--------------------------|----|--------|------------------------|----------|--------------------------------------|
| | id | name | address | admin_id | qrcode |
| <input type="checkbox"/> | 1 | Koščak | Šentvid pri Stični 27a | 1 | aa235133-f9c7-5bdd-a0d6-f8fd098f602d |

Slika 3.4: Primer entitete *family*

3.2.3 Izdelek

V tej entiteti so shranjeni vsi podatki o izdelkih. Sem se shranjujejo izdelki, ki jih uporabniki dodajajo na svoje sezname. V kolikor izdelek že obstaja v zbirki, se nov ne naredi, ampak se uporabi že obstoječi.

Sestavljena je iz atributov *id*, *name* in *producer*. Atribut *name* predstavlja ime izdelka, atribut *producer* pa predstavlja proizvajalca tega izdelka.




| ← T → | | | |
|--------------------------|----|---------------|---------------|
| | id | name | producer |
| <input type="checkbox"/> | 4 | Kisla smetana | Zelene doline |

Slika 3.5: Primer entitete *product*

3.2.4 Seznam

V entiteti *list* so shranjeni vsi seznam po družinah. Ko nekdo v družini naredi nov seznam za nakupovanje, se ta shrani v to entiteto s podatki *id*, *name*, *created*, *admin_id* in *id_f_id*.

Atribut *name* predstavlja ime seznama, v družini pa ne more biti več seznamov z istim imenom. V atributu *created* je shranjen datum z uro, kdaj je bil seznam ustvarjen. Atribut *admin_id* je enak kot pri družini in predstavlja osebo, ki lahko seznam zbriše. Zadnji atribut pa je tuj ključ na družino, v katero spada seznam.




| ← T → | | | | | | |
|--------------------------|--|--|--|---------|---------------------|---------|
| | | id | name | id_f_id | admin_id | created |
| <input type="checkbox"/> |  Edit |  Copy |  Delete | 4 | Hofer | 1 |
| | | | | 3 | 2022-10-29 18:33:53 | |

Slika 3.6: Primer entitete *list*

3.2.5 Povezovalne tabele




V podatkovni zbirki sta tudi dve povezovalni entiteti, ki povezujeta uporabnike z družinami (*family_has_user*) in izdelke s seznamami (*list_has_product*).

Prva entiteta vsebuje podatke, kdo spada v katero družino. Vsebuje tri attribute, en je enolični identifikator, druga dva pa sta tuja ključa. Tuja ključa se vežeta na uporabnika in na družino. Na ta način sta entiteti povezani in lahko ugotovimo, kdo spada v katero družino.

| ← T → | | | |
|--------------------------|--|--|--|
| | id | id_u_id | id_f_id |
| <input type="checkbox"/> |  Edit |  Copy |  Delete |
| | 1 | 1 | 1 |

Slika 3.7: Primer entitete *family_has_user*

Druga entiteta pa vsebuje podatke, kateri izdelek spada na kater seznam. Za razliko od druge ima ta več atributov, ki določajo, ali je bil izdelek kupljen, koliko tega izdelka potrebujemo, kdaj je bil dodan in kdo ga je dodal ter kdo ga je kupil.

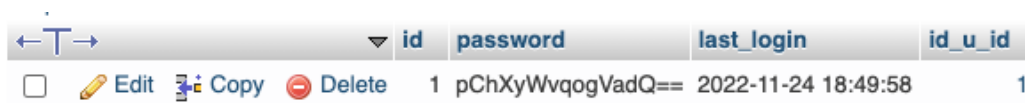
| ← T → | | | | | | | | | |
|--------------------------|--|--|--|-------|--------|------------|---------------------|---------------------|---|
| | id | id_p_id | id_l_id | added | bought | added_u_id | bought_u_id | quantity | |
| <input type="checkbox"/> |  Edit |  Copy |  Delete | 17 | 15 | 7 | 2022-11-17 21:13:38 | 2022-11-17 21:13:51 | 7 |
| | | | | | | | | | 7 |
| | | | | | | | | | 1 |

Slika 3.8: Primer entitete *list_has_product*

3.2.6 Uporabniška gesla

Posebej imam določeno še entiteto v kateri so shranjena uporabniška gesla. Razlog za to je, da v primeru, ko API dostopa do baze za potrebe

podatkov, ki so preko tujega ključa povezani z uporabnikom, baza vrne tudi geslo. Ker se mi to ni zdelo varno, sem težavo rešil z novo entiteto, v kateri so uporabniška gesla, datum in ura zadnje prijave ter identifikator uporabnika.



| | id | password | last_login | id_u_id |
|---|----|------------------|---------------------|---------|
| <input type="checkbox"/> Edit Copy Delete | 1 | pChXyWvqogVadQ== | 2022-11-24 18:49:58 | 1 |

Slika 3.9: Primer entitete *userpassword*

Poglavje 4

Načrtovanje

V tem poglavju je predstavljeno, kako sem se lotil ustvarjanja mobilne aplikacije in kako je zgrajena.

Prvi korak v razvoju aplikacije je bil načrtovanje. V tem delu sem zbral zamisli, kaj vse želim implementirati v aplikaciji in katere tehnologije bi za to uporabil.

Začel sem z analizo potreb, kaj vse bi si uporabniki želeli, da jim aplikacija nudi. Odločil sem se, da bo aplikacija služila kot nadomestilo za fizični papirnati listek z dodano vrednostjo. Ta dodana vrednost je možnost deljenja seznama z ostalimi uporabniki, ves čas dostopen seznam in sledenje kupljenih izdelkov.

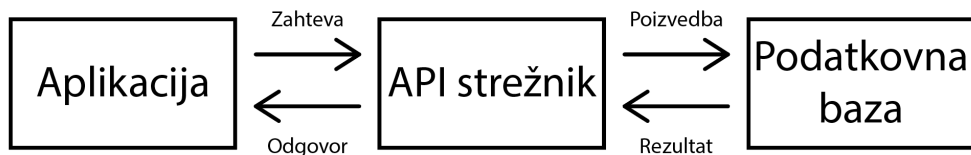
4.1 Izbira arhitekture

Za potrebe aplikacije sem na lastnem strežniku postavil novo virtualno okolje, v katerem sta podatkovna baza in API strežnik.

Ko sem imel dodelano zamisel, kaj bo aplikacija ponujala, sem začel s pregledovanjem različnih možnosti za implementiranje vseh funkcij aplikacije. Tako sem prišel do zaključka, da bom uporabil ogrodje Flutter.

Ker ima veliko podporo skupnosti je tudi lahek za uporabo in ima veliko dodatkov, ki olajšajo delo. Izmed veliko možnih sem izbral dodatke, ki mi

omogočajo generiranje in branje QR kod, olajšajo delo z animacijami in omogočajo lažje nadzorovanje aktivnega stanja (state manager).



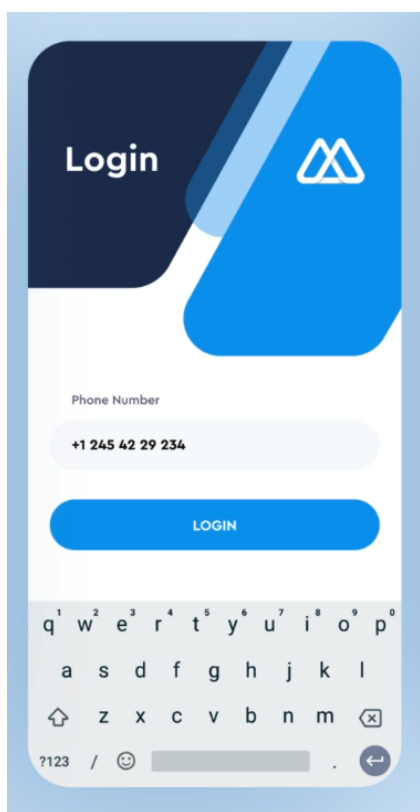
Slika 4.1: Arhitekturna shema

4.2 Uporabniški vmesnik

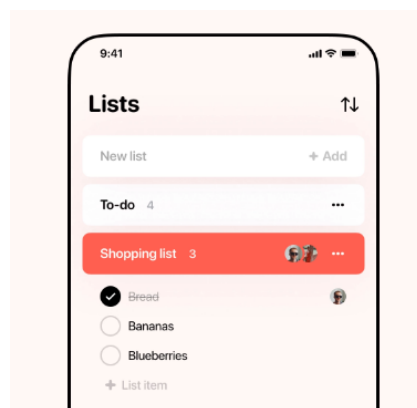
Sledilo je načrtovanje uporabniškega vmesnika aplikacije. Inspiracijo za to sem iskal na spletnem mestu *dribbble*. To je spletna stran, na katero ustvarjalci nalagajo svoje ideje, predloge in animacije. Vmesnik sem zasnoval na podlagi nekaterih idej, ki sem jih prilagodil in v programu Adobe XD ustvaril celotno podobo aplikacije.

V programu Adobe Illustrator sem ustvaril ikono aplikacije, ki sem jo s pomočjo spletnega programa za animacije *Rive* tudi animiral.

Tako sem dobil celotno podobo aplikacije, kar mi je olajšalo nadaljnje delo.



Slika 4.2: Ideja za prijavni zaslon



Slika 4.3: Ideja za domač zaslon

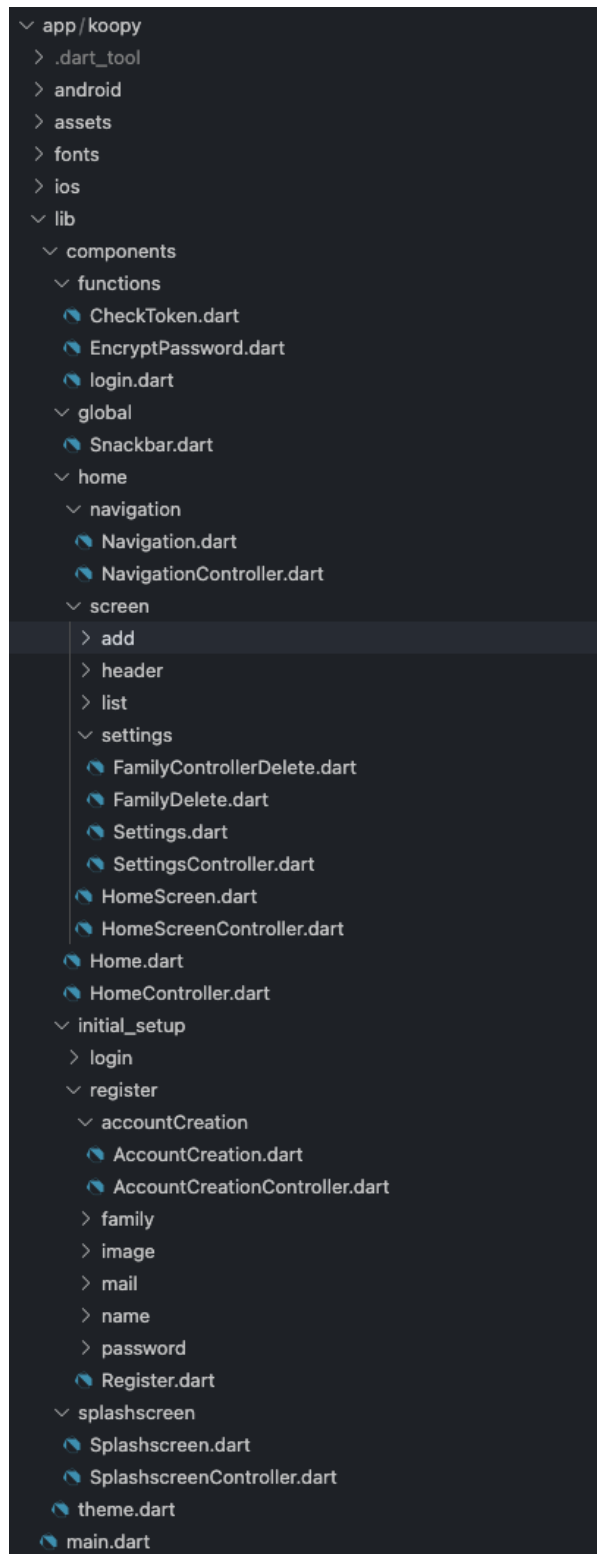
4.3 Direktorijška zgradba aplikacije

Aplikacija je razdeljena na več direktorijev, da omogoča lažje branje programske kode in kasnejše vzdrževanje.

V korenskem (root directory) lahko vidimo tri pomembnejše direktorije. To so *assets*, *fonts* in *lib*. Prva dva služita za shranjevanje slik, animacij in pisav. Direktorij *assets* je razdeljen še na dva manjša direktorija *animations* in *images*. V prvem sta shranjeni dve animaciji, ena za odpiranje aplikacije in ena za ustvarjanje novega računa. V drugi pa so različice ikone aplikacije.

Glavni direktorij, v katerem se nahaja vsa programska koda, pa je *lib*. V njem lahko najdemo datoteko *main.dart*, ki se zažene, ko odpremo aplikacijo. Ta potem pokliče druge elemente, ki so shranjeni v direktoriju *components* znotraj direktorija *lib*.

V direktoriju *components* pa so shranjene komponente aplikacije, na primer domača stran, dialog, ki se pojavi ob uspešno izvedeni operaciji ali ob napaki.



Slika 4.4: Direktorijska zgradba aplikacije

4.4 Varnost

Ker je aplikacija prosto dostopna vsem, sem moral poskrbeti tudi za varnost le te. Za to sem poskrbel z uporabo JSON Web Token (JWT) žetonov. To so žetoni, ki so sestavljeni iz treh delov. Prvi del predstavlja glavo, v kateri sta zapisana algoritem kodiranja in tip žetona. Drugi del so podatki, ki jih nosi. Tretji del pa je kontrolni del in skrbi, da žeton ni pokvarjen oziroma ga ne more nekdo naključno spremeniti.

Encoded

PASTE A TOKEN HERE

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0IjoxNTE2MjM5MDIyfQ.SflKxwRJSMeKKF2QT4fwpMeJf36P0k6yJV_adQssw5c
```

Decoded

EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE

```
{  "alg": "HS256",  "typ": "JWT"}
```

PAYLOAD: DATA

```
{  "sub": "1234567890",  "name": "John Doe",  "iat": 1516239022}
```

VERIFY SIGNATURE

```
HMACSHA256(  base64UrlEncode(header) + "." +  base64UrlEncode(payload),  your-256-bit-secret)
```

☐ secret base64 encoded

Slika 4.5: Primer JSON Web Token-a [4]

Za namen varnosti pri prijavi in prestrezanju podatkov, pa sem geslo zakodiral z uporabo *Salsa20* šifre. Za to kodiranje sem določil naključen ključ, s katerim se geslo v obliki niza znakov zakodira in ga ni mogoče razvozlati brez tega ključa. Na ta način sem zagotovil, da tudi v primeru prestrezanja podatkov, geslo uporabnika ni enostavno berljivo.

Poglavje 5

Predstavitev aplikacije

V tem poglavju je predstavljena sama aplikacija, z določenimi posnetki zaslona. Predstavljeni so pomembnejši zasloni, opisano je njihovo delovanje in kakšno vlogo imajo za uporabnika. Vsi posnetki zaslona so zajeti v simulatorju *iPhone 14 Pro*.

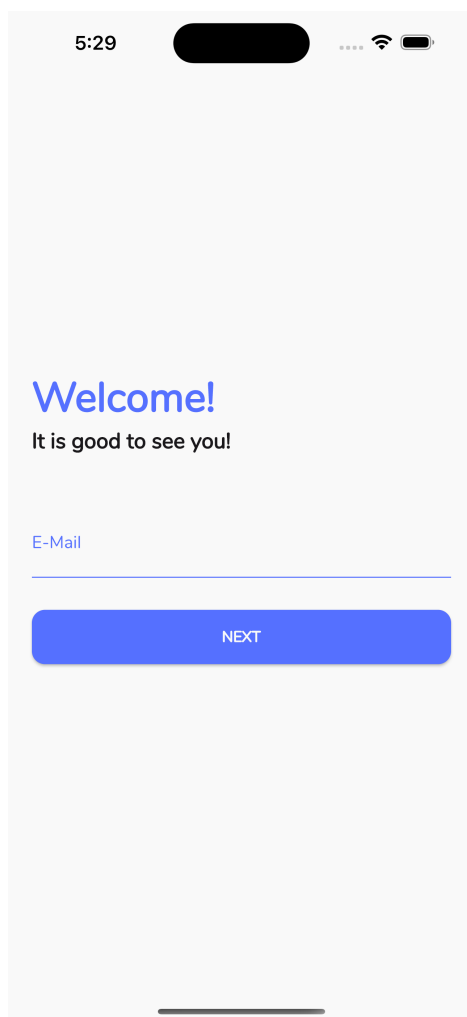
5.1 Prijava in registracija

Prvi zaslon, ki pozdravi uporabnika, je animirana ikona aplikacije. Animacija ikone služi kot distrakcija, ko aplikacija prenaša zahtevane podatke z interneta. Ko se podatki prenesejo in je aplikacija pripravljena na uporabo, se animacija konča in s spodnje strani se na ekranu pokaže gumb za nadaljevanje.

Na naslednjem zaslonu uporabnik vnese svoj e-poštni naslov. Aplikacija preveri, ali naslov že obstaja med aktivnimi uporabniki. V kolikor uporabnik obstaja, ga preusmeri na zaslon za vpis gesla, drugače pa nadaljuje z vpisovanjem imena.

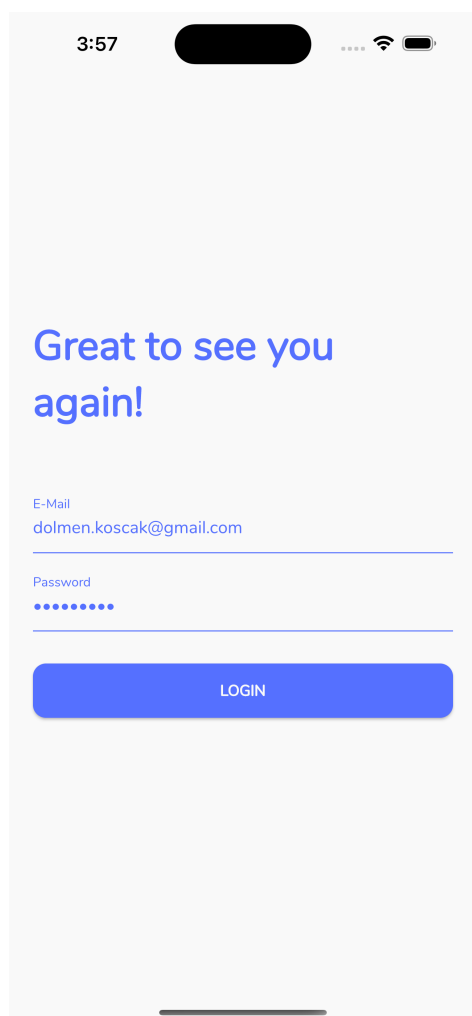


Slika 5.1: Pozdravni zaslon

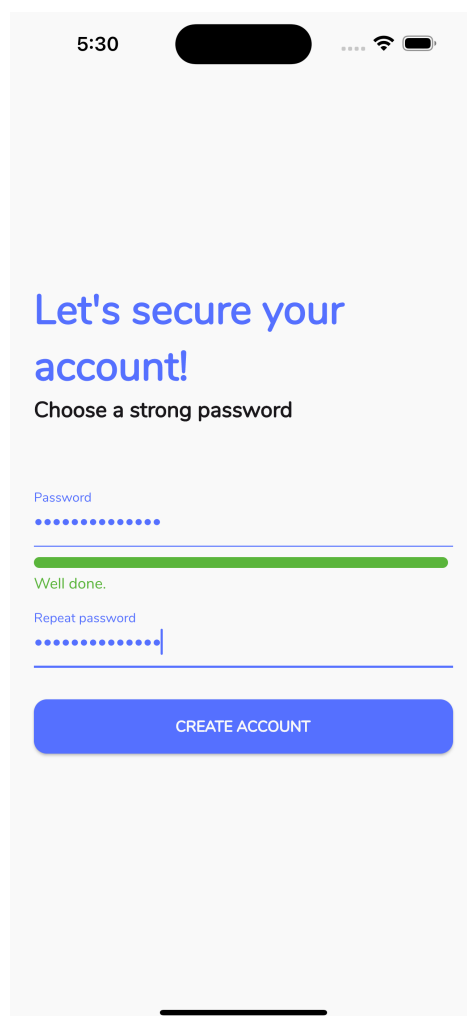


Slika 5.2: Vnos e-poštnega naslova

V kolikor e-poštni naslov obstaja, se uporabniku prikaže dialog, ki ga vpraša, ali se želi prijaviti. S klikom nanj uporabnika preusmeri na zaslon za prijavo. Če e-poštni naslov v podatkovni zbirki še ne obstaja, je od uporabnika zahtevan vnos imena in gesla, ki ga bo uporabljal za prijavo v aplikacijo.



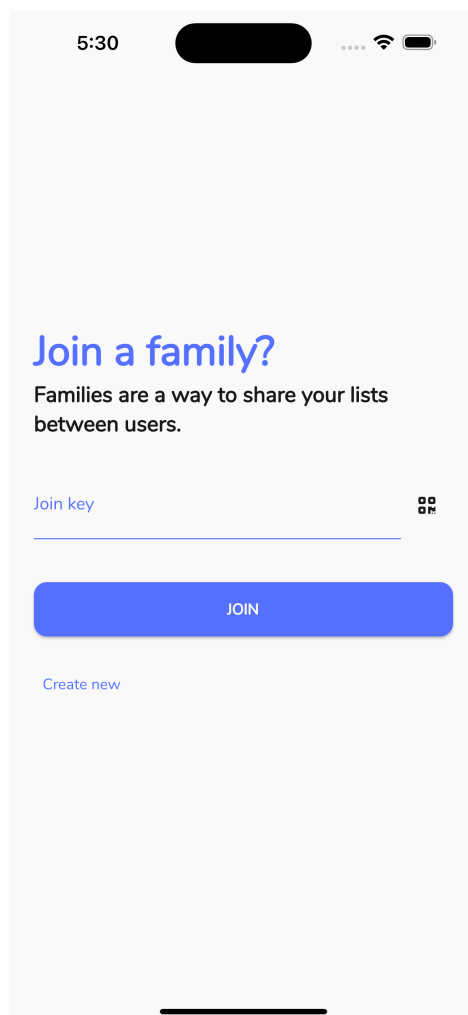
Slika 5.3: Prijavni zaslon



Slika 5.4: Določitev gesla

5.2 Pridružitev družini

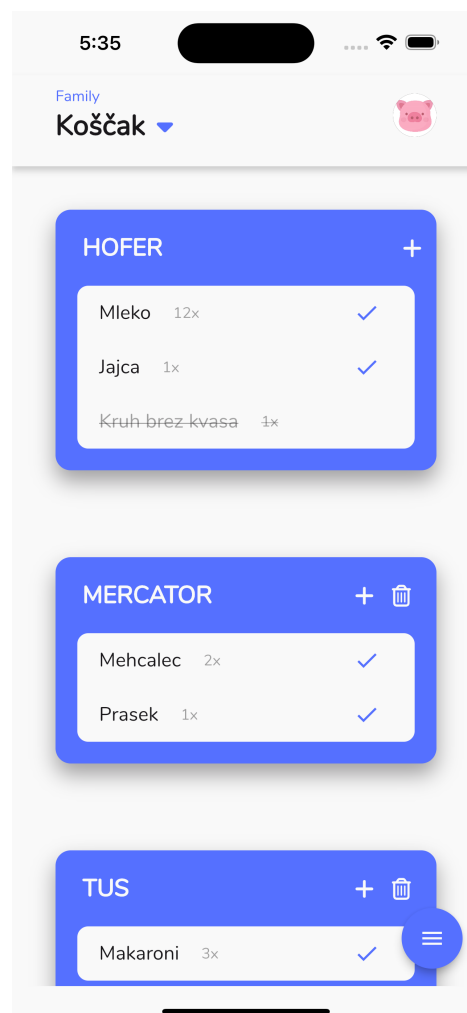
Če uporabnika še ni v bazi, ga aplikacija po uspešni registraciji preusmeri na zaslon za pridružitev družini. Tu lahko uporabnik ustvari novo družino, ali pa se pridruži že obstoječi.



Slika 5.5: Pridružitev družini

5.3 Domač zaslon

V kolikor uporabnik obstaja in se je v prejšnjih korakih uspešno prijavil, se mu pokaže domač zaslon, na katerem vidi sezname z izdelki. Od tu naprej lahko ustvarja nove sezname, dodaja izdelke in ureja nastavitve.

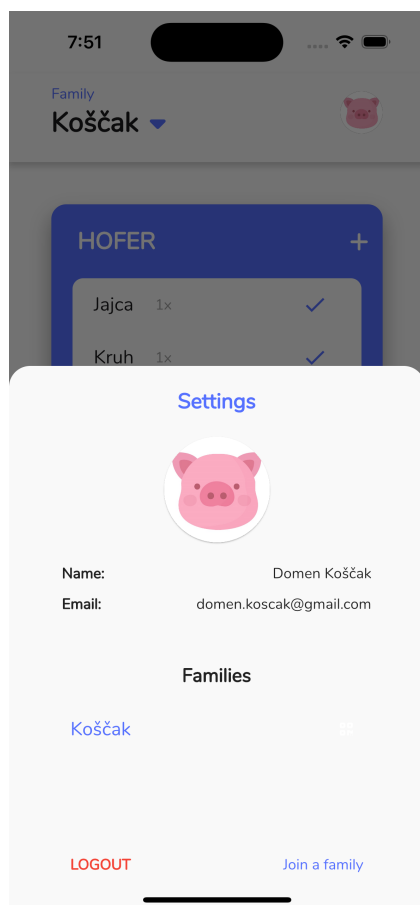


Slika 5.6: Domač zaslon

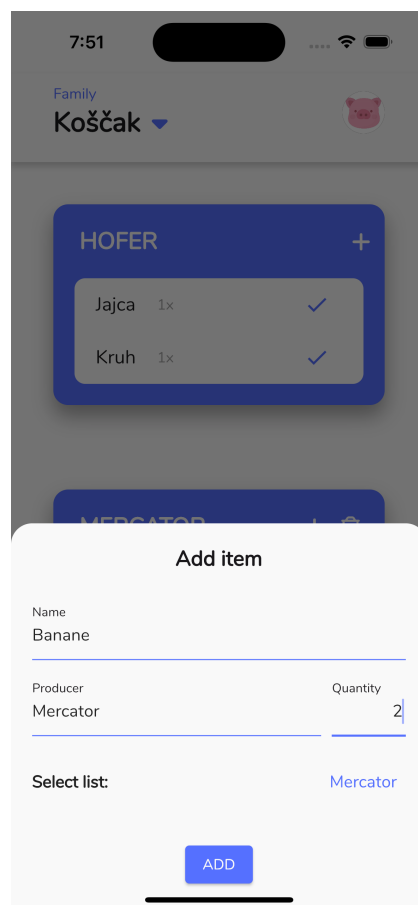
5.4 Nastavitve in dodajanje novih izdelkov

Pomembnejši sta še dve okni, ki sta poravnani na spodnji rob in ob odprtju priletita na zaslon. To sta okno za nastavitve in okno za dodajanje novih izdelkov na seznam. Prvi služi kot pregled podatkov, pridružitve v novo družino ali prikazovanje QR kode za pridružitve, in pa za odjavo iz aplikacije.

Drugo okno pa služi za dodajanje izdelkov na seznam. Tu so zahtevani podatki ime izdelka, proizvajalec, količina in seznam, na katerega dodajamo izdelek. Po dodajanju se okno zapre in izdelek se pokaže na izbranem seznamu.



Slika 5.7: Nastavitve



Slika 5.8: Dodajanje izdelka

Poglavje 6

Zaključek

Z ustvarjanjem aplikacije sem se naučil veliko novega in spoznal nove tehnologije. Flutter sem v osnovi poznal že prej, nikoli pa ga nisem uporabil za namen večjega projekta kot je bila aplikacija Koopy. Spopadel sem se tudi z načrtovanjem uporabniškega vmesnika, ikon in animacij. Kot pričakovano, sem se moral, v procesu ustvarjanja, spopasti tudi z nepričakovanimi težavami in napakami tako v programski kodi kot tudi v samem načrtu aplikacije. S potrpežljivostjo in pomočjo spletnih forumov sem vse težave premagal in ustvaril aplikacijo, ki jo lahko uporabniki uporabljajo in si olajšajo nakupovanje ter pripomorejo k čistejšemu okolju.

Razvoj aplikacije in pisanje diplomskega dela je bil iziv, ki je zahteval veliko časa, truda in potrpežljivosti. Na trenutke je zmagal manjko motivacije, ki pa ga je bilo potrebno premagati. Na koncu se je vztrajnost izplačala, saj je nastal lep in uporaben produkt.

6.1 Nadaljnji razvoj

V fazi testiranja in dnevne uporabe so se pojavile tudi nekatere ideje, ki bi si jih želel v prihodnosti implementirati v aplikaciji. Nekatere izmed njih so bile dodajanje možnosti preverjanja izdelka. To pomeni, da bi poleg že zahtevanih podatkov izdelek shranjeval tudi črtno kodo. Na ta način, bi v

trgovini lahko s kamero telefona prebrali črtno kodo in bi vedeli, da gre za pravi izdelek. Tako bi se izognili dvomom, v kolikor proizvajalec spremeni izgled produkta. Druga ideja pa je bila, da bi ustvaril tudi spletni vmesnik. Tako bi lahko do aplikacije dostopali tudi iz računalnika ali katere nepodprte naprave, saj bi bila na voljo v spletu.

V kolikor bi aplikacija postala zelo zanimiva in bi število uporabnikov hitro raslo, pa bi bilo potrebno nadgraditi varnost aplikacije in izboljšati poizvedbe ter podatkovno bazo.

Celotna literatura

- [1] *Flasgger - Easy Swagger UI for your Flask API*. <https://github.com/flasgger/flasgger>. Dostopano: 30.6.2022.
- [2] *Flask (web framework)*. [https://en.wikipedia.org/wiki/Flask_\(web_framework\)](https://en.wikipedia.org/wiki/Flask_(web_framework)). Dostopano: 30.6.2022.
- [3] *Flutter (software)*. [https://en.wikipedia.org/wiki/Flutter_\(software\)](https://en.wikipedia.org/wiki/Flutter_(software)). Dostopano: 30.6.2022.
- [4] *Java Web Token*. <https://jwt.io>. Dostopano: 30.6.2022.
- [5] *MariaDB*. <https://en.wikipedia.org/wiki/MariaDB>. Dostopano: 30.6.2022.
- [6] *PeeWee documentation*. <http://docs.peewee-orm.com/en/latest/>. Dostopano: 30.6.2022.
- [7] *Python (programming language)*. [https://en.wikipedia.org/wiki/Python_\(programming_language\)](https://en.wikipedia.org/wiki/Python_(programming_language)). Dostopano: 30.6.2022.