ŠOLSKI CENTER CELJE
SŠ KER

SOCIALNA APLIKACIJA TEGGJŪ

Mentor:
dipl. inž. BOŠTJAN LUBEJ

Avtorji:
DOMEN LAZNIK, R-4. B
TRISTAN GAJŠEK, R - 4. B
ŽIGA KLEPEJ, R - 4. B

CELJE, 1. 2. 2022
2021 / 2022

1. 2. 2022

IZJAVA*

Mentor Boštjan Lubej v skladu z 20. členom Pravilnika o organizaciji mladinske raziskovalne dejavnosti »Mladi za Celje« Mestne občine Celje, zagotavljam, da je v raziskovalni nalogi z naslovom Socialna aplikacija Teggjū, katere avtorica je/so Domen Laznik, Tristan Gajšek in Žiga Klepej:

- besedilo v tiskani in elektronski obliki istovetno,
- pri raziskovanju uporabljeno gradivo navedeno v seznamu uporabljene literature,
- da je za objavo fotografij v nalogi pridobljeno avtorjevo dovoljenje in je hranjeno v šolskem arhivu,
- da sme Osrednja knjižnica Celje objaviti raziskovalno nalogo v polnem besedilu na knjižničnih portalih z navedbo, da je raziskovalna naloga nastala v okviru projekta Mladi za Celje,
- da je raziskovalno nalogo dovoljeno uporabiti za izobraževalne in raziskovalne namene s povzemanjem misli, idej, konceptov oziroma besedil iz naloge ob upoštevanju avtorstva in korektnem citiranju,
- da smo seznanjeni z razpisni pogoji projekta Mladi za Celje.

Celje, 12. 4. 2022

žig šole

ŠOLSKI CENTER CELJE
SREDNJA ŠOLA ZA KEMIJO, ELEKTROTEHNIKO IN RAČUNALNIŠTVO
CELJE, Pot na Lavo 22

Podpis mentorja

Podpis odgovorne osebe

*

POJASNILO

V skladu z 20. členom Pravilnika raziskovalne dejavnosti »Mladi za Celje« Mestne občine Celje je potrebno podpisano izjavo mentorja (-ice) in odgovorne osebe šole vključiti v izvod za knjižnico, dovoljenje za objavo avtorja (-ice) fotografskega gradiva, katerega ni avtor (-ica) raziskovalne naloge, pa hrani šola v svojem arhivu.

ZAHVALA

Zahvaljujemo se vsem, ki so kakorkoli pomagali pri izdelavi raziskovalne naloge. Brez pomoči naloga ne bi nastala, pa naj je šlo le za spodbudne besede, majhno idejo ali nasvete in kritike pri izdelovanju izdelka.

Zahvaljujemo se mentorju prof. dipl. inž. Boštjanu Lubeju za ves trud, čas, podporo in vztrajnost, ki ga je vložil v izdelovanje raziskovalne naloge, ter za tehnični pregled le-te.

Zahvala gre tudi Nataši Gajšek za slovnični pregled, ravnateljici Aniti Laznik, ki podpira raziskovalno dejavnost na šoli in prof. mag. Boštjanu Resinoviču za napotke, informacije ter podporo.

KAZALO VSEBINE

KAZALO SLIK

KAZALO GRAFOV

Povzetek

Z našo raziskovalno nalogo smo želeli ugotoviti, kakšne socialne aplikacije so danes najbolj priljubljene, kaj je pri njih uporabnikom najpomembneje, kaj si želijo videti, katerih lastnosti ne marajo in kakšen jim je najljubši izgled. Predvidevali smo, da je uporabnikom všeč preprost videz, slikovna in video vsebina, prilagodljiv videz ter zabava in komuniciranje kot namen uporabe. Mislili smo tudi, da uporabnikom ni všeč, ko aplikacije zbirajo njihove podatke, četudi so ti uporabljeni za prilagajanje priporočene vsebine.

Pri raziskovanju smo dobili te informacije iz spletne ankete, na katero so odgovarjali dijaki naše šole in še ostali uporabniki socialnih aplikacij. Ugotovili smo, da so se anketiranci strinjali z večino naših predvidevanj, razen z zbiranjem podatkov aplikacij, kar nas je presenetilo. Pustili so nam še tudi mnenja in predloge, kako bi lahko izdelali čim boljšo takšno socialno aplikacijo.

Izdelali smo mobilno socialno aplikacijo na podlagi informacij, ki smo jih prejeli od ankete. Aplikacijo smo poimenovali Teggjū in je predvsem namenjena ogledovanju slikovnih vsebin uporabnikov, ki jim slediš, všečkanju teh vsebin, komuniciranju s prijatelji ter objavljanju lastne vsebine.

Ključne besede: Android, podatkovne baze, React-Native, Socialna aplikacija, virtualna naprava

Summary

With our research project, we wanted to find out which social applications are the most popular today, what is most important to users, what they want to see, which features they do not like and what is their favorite look. We anticipated that users would like simple looks, pictorial and video content, flexible looks, and entertainment and communication as the purpose of use. We also thought users didn't like it when apps collected their data, even if it was used to customize recommended content.

In the research, we obtained this information from an online survey, which was answered by students of our school and other users of social applications. We found that respondents agreed with most of our assumptions, except for the collection of application data, which surprised us. They also left us opinions and suggestions on how we could create the best possible social application.

We created a mobile social application based on the information we received from the survey named Teggjū. The application is primarily intended for viewing the image content of the users you follow, liking this content, communicating with friends and publishing your own content.

Keywords: Android, databases, React-Native, Social application, virtual device

Kratice in okrajšave

CSS – Cascading Style Sheets (Kaskadni slogovni listi)

JSON – JavaScript Object Notation (JavaScript objektni zapis)

SQL – Structured Query Language (Strukturiran jezik za povpraševanje)

1 UVOD

Iz dneva v dan se srečamo z aplikacijami, s katerimi se lahko povežemo. Uporabljamo jih, vendar ne vse; temveč pa tiste, ki so nam najbolj praktične ali pa so nam najbolj všeč. Nikoli se ne vprašamo, kako so te aplikacije narejene in kako težko jih je narediti. S takšnimi vprašanji smo se srečali v našem projektu in nanj poskušali odgovoriti, medtem ko smo ustvarili svojo aplikacijo za socialna omrežja.

1.1 Opredelitev problema

Zaradi hitrega razvoja tehnologije do te mere, da ima danes vsak lahko v žepu zelo zmogljiv računalnik, ni presenečenje, da si želi veliko ljudi hitrega dostopa do informacij, zabave, novic ter komuniciranja s komer koli, kadar koli in kjer koli.
Na spletu je na voljo že veliko socialnih aplikacij, ki nudijo takšne storitve na prvi pogled zastonj. V resnici praktično vsaka aplikacija pridobiva prihodke od oglaševanja ali pa s prodajo osebnih informacij uporabnikov, ne da bi se ti sami tega zavedali.
Za našo aplikacijo je vsak imel svoje ideje, kako bi funkcionirala, izgledala in kakšne storitve bi nudila. Zato smo se obrnili na več drugih uporabnikov popularnih socialnih aplikacij, da bi s pomočjo ankete ugotovili, kaj imajo najraje, kaj jim največ pomeni in kaj bi si želeli uporabljati.

1.2 Metode raziskovanja

Za raziskovalno metodo smo uporabili spletno anketo. Tako smo lahko v kratkem času dobili veliko podatkov od velikega števila ljudi. Naša ciljna skupina so bili naši sovrstniki, ostali najstniki in ostali uporabniki socialnih aplikacij. Pri reševanju ankete je sodelovalo 105 anketirancev, ki so odgovarjali na različna vprašanja o izgledu, delovanju, uporabi in lastnih preferencah socialnih aplikacij. Z rezultati ankete smo tudi preverjali zastavljene hipoteze in jih na podlagi tega potrdili ali pa zavrgli.

1.3 Hipoteze in cilji

Cilj raziskovalne naloge je pridobiti informacije o različnih lastnostih socialnih aplikacij in ugotoviti, česa si uporabniki najbolj želijo, te lastnosti pa implementirati v našo lastno aplikacijo.

V raziskovalni nalogi smo si postavili naslednje hipoteze:

1. Večina ljudi uporablja socialne aplikacije za komuniciranje s prijatelji in zabavo z uporabo videoposnetkov in slik.
2. Mnogim je najljubše, da aplikacije nimajo kompleksnega videza in kompleksne uporabe, pomemben jim je prilagodljiv videz, kot je na primer nočni način.
3. Uporabniki si ne želijo, da bi aplikacije zbirale njihove podatke, čeprav zgolj za usmerjeno oglaševanje in priporočanje podobnih vsebin.

2 SOCIALNA APLIKACIJA TEGGJŪ

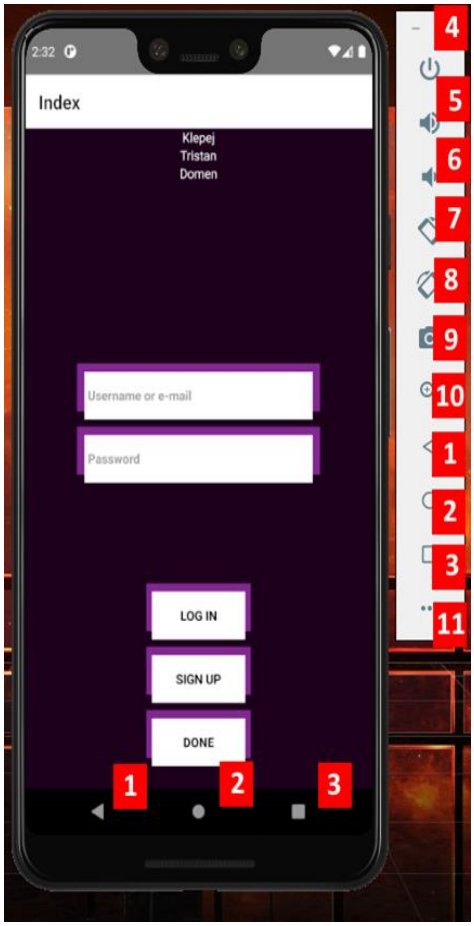
Da se lahko spopadamo z vprašanji, ki smo jih postavili pred izdelavo, smo morali ustvariti svojo aplikacijo za socialna omrežja; srečali smo se s prvim vprašanjem: katero programsko okolje bi uporabili za prav to?

2.1 Izbira okolja in drugih programskih oprem

Za izdelavo naše socialne aplikacije smo imeli mnogo opcij, a smo se na koncu odločili za React-Native. To je odprtokodno ogrodje za izdelavo grafičnega vmesnika na sistemih Android in iOS. Ker smo potrebovali nekakšen način, da vidimo rezultate našega dela, smo potrebovali emulator, saj bi objavljanje kode na naše mobilne naprave potrebovalo preveč časa. Za ta namen smo uporabili Android Studio, saj ima vgrajen emulator, ki ga je zelo lahko uporabiti.

2.1.1 Emulator

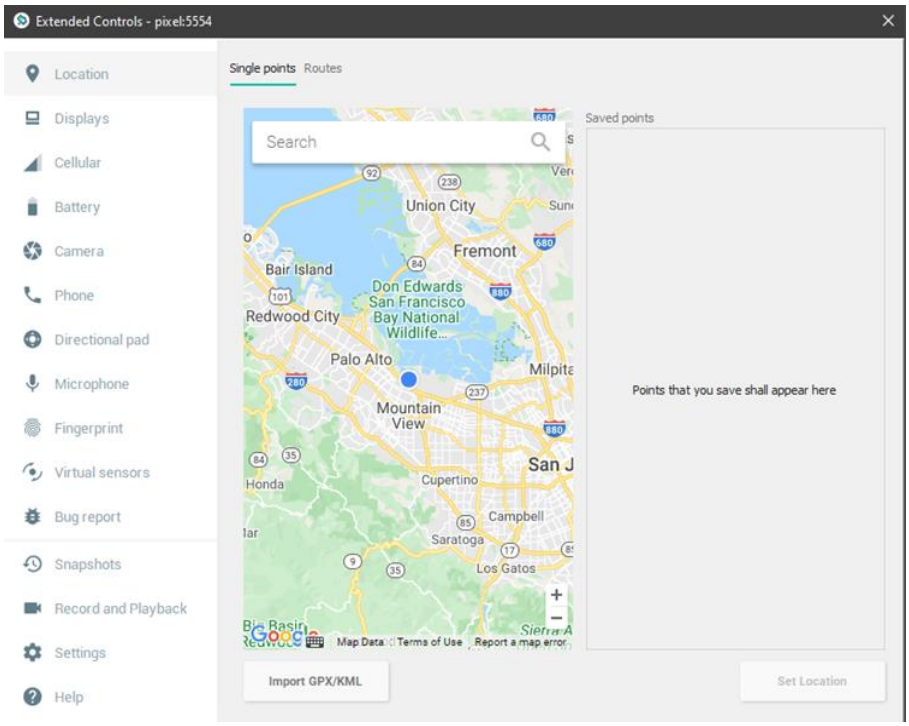
Emulator deluje tako kot navaden Pixel telefon in ima mnogo možnosti za lažje programiranje.



Slika 1 - Emulator in njegove funkcije

1. Gumb za vrnitev na prejšnjo stran,
2. Gumb za vrnitev na osnovni zaslon,
3. Gumb za pogled zavihkov,
4. Gumb za zaustavitev emulatorja,
5. Gumb za nastavljanje glasnosti,
6. Gumb za popolno utišanje naprave,
7. Gumb za obrat emulatorja na levo za 90°,
8. Gumb za obrat emulatorja na desno za 90°,
9. Gumb za zajem slike zaslona,
10. Gumb za povečanje zaslona,
11. Dodatne nastavitve

V dodatnih nastavitvah so še ostale možnosti za nastavitev naprave.



Slika 2 - Dodatne nastavitve emulatorja

Tukaj lahko spreminjamo še dodatne nastavitve naprave, kot so lokacija. Nam konec koncev to pride prav? Odgovor je da, saj nam jezik React-Native tudi omogoča na primer dostop lokacije naprave, kar pride prav tudi v socialnih aplikacijah, kjer lahko vidimo novice, ki so aktualne v naši bližini.

2.2 Teoretične osnove

Da smo lahko razumeli postopek izdelovanja aplikacij za Android operacijski sistem v React-Native okolju, smo najprej potrebovali predznanje za jezik, ki smo ga uporabljali. React-Native je do neke mere zelo neortodoksen jezik, saj ima nenavadno sintakso in v določenih primerih deluje na nepričakovan način.

2.2.1 Import

Prva stvar, ki smo jo morali poznati, je bila ključna beseda “import”. Ta nam omogoča, da v našo kodo implementiramo že obstoječo kodo, ki jo lahko napišemo mi, ali pa nekdo drug.

```
import React, { useState, useEffect, Component } from "react";
import {
  View,
  Text,
  Pressable,
  StyleSheet,
  Image,
  ScrollView
} from "react-native";
```

Slika 3 - Koda za uvoz knjižnic in ostalih datotek

V zgornjem primeru lahko vidimo kodo iz datoteke »account.js«, ki v naš program vnese že napisano kodo iz »react« in »react-native« knjižnic, ki nam omogoči, da uporabljamo metode, komponente in elemente, kot so »userState«, »Component«, »View« itd.

2.2.2 Pogostejši elementi v programiranju

React-Native ima veliko elementov, ki so prisotni v več jezikih in so zato lažji za razumevanje v primeru, da ima program že izkušnje z drugimi jeziki, predvsem JavaScript, saj je veliko sintakse zelo podobne kodi v njem.

2.2.2.1 Komentarji

Preden začnemo z elementi, ki dodajo nekakšno funkcionalnost našemu programu, bi bilo dobro, da vemo, kaj so komentarji. Kot v večini jezikov poznamo dve vrsti komentarjev: enovrstični in večvrstični in, kot po navadi, so tukaj tudi enovrstični označeni z dvema poševnicama (“//”), medtem ko se večvrstični začnejo s poševnico in zvezdico (“/*”), končajo pa z zvezdico in poševnico (“*/”).

V te komentarje lahko damo kar koli hočemo, saj so komentarji v procesu prevajanja kode v program ignorirani. Zaradi tega so najpogosteje uporabljeni za označevanje določenih delov kode in opisovanje, kako ta koda dela.

2.2.2.2 Spremenljivke

Najosnovnejši elementi v React-Native, ki imajo nekakšno funkcionalnost, so spremenljivke, ki so v tem primeru deklarirane s ključno besedo "var" ali pa "let". React-Native deluje na dinamičen način, ko pride do spremenljivk. To pomeni, da lahko spremenljivkam spremenimo podatkovni tip po tem, ko jim je vrednost že bila določena. Za primer tega bi lahko dali spremenljivko, v katero shranjujemo podatke v obliki celih števil. V to spremenljivko lahko kasneje shranjujemo vrednosti, kot so nizi, decimalna števila itd.

2.2.2.3 Funkcije in metode

V programiranju so funkcije način, kako lahko večje število stavkov damo v eno zbirko in je ta nato lahko poklicana. To nam omogoči, da lahko več vrstic kode izvedemo z le eno vrstico. Metode so iste kot funkcije, z razliko, da so del razreda. Te so lahko statične in objektne, kar nam določa, če potrebujemo poseben objekt za uporabo metode, ali pa jo lahko pokličemo kar s pomočjo razreda.

Ključna beseda "function" je kot v JavaScript namenjena za definicijo funkcij. Kot skoraj v vseh jezikih imajo te funkcije ime, da jih sploh lahko pokličemo, in parametre, ki jih lahko uporabimo za spreminjanje načina, na katerem bo funkcija delovala. Lahko bi na primer imeli dva parametra, ki bi bila števili, ki sta nato sešteti. Vsem tem sledi telo funkcije, kjer je koda, ki bo izvedena. Na koncu imamo "return" stavek, ki je v funkcijah in metodah uporabljen, da vrne vrednost po klicu funkcije. Ta vrednost je lahko nato dana določeni spremenljivki.

Če hočemo poklicati funkcijo, ki smo jo na zgornji način definirali, moramo napisati najprej ime funkcije, nato pa oklepaje, v katerih lahko damo vrednosti, če ima dana funkcija parametre. Poleg funkcij in metod še poznamo anonimne funkcije, ki so redkejše, a so vseeno lahko uporabne v določenih primerih. Definicija teh funkcij je zelo podobna definiciji normalnih funkcij, a zdaj moramo uporabiti znak za enakost pred oklepaji s parametri in puščico za njimi. V primeru, da nočemo funkcije shraniti v neko spremenljivko, preko katere bi bila poklicana, lahko izpustimo prvi del.

V našem programu smo anonimne funkcije predvsem uporabljali v primerih, kjer smo morali kot parameter funkciji dati še eno funkcijo, a je nismo hoteli definirati kot navadne funkcije, saj je v drugih primerih ne bi uporabljali in smo zato naredili anonimno funkcijo.

2.2.2.4 Razredi

Ker je React-Native bolj ali manj objektno usmerjen jezik, ima razrede in objekte. Ti so lahko uporabljeni za poenostavitev kode, da nam je bolj razumljiva. To omogočimo s tem, da damo več spremenljivk in metod v en razred, ki bi predstavljal vse, kar potrebuje primerek nečesa, kar se v tem primeru imenuje objekt. Zato imamo lahko na primer en razred, ki predstavlja zgradbo uporabnika. To so njegove lastnosti, ki jih predstavljajo spremenljivke in njegovo delovanje, ki predstavljajo metode.

Navadno imamo pri objektno-usmerjenem programiranju tudi dedovanje, kar pomeni, da lahko določenemu razredu določimo, da bo imel vse elemente razreda, od katerega deduje. Poznamo tudi primere dedovanja, ki zahtevajo, da v razredu, ki deduje od nekega drugega razreda, moramo definirati določen element, navadno metodo, ker to razred, od katerega dedujemo zahteva.

2.2.3 JSON format

"JSON" je kratica za "JavaScript Object Notation". To je format za zapisovanje podatkov, ki je pogosto uporabljen v mnogo objektno-usmerjenih jezikih in tudi v našem programu. Kot je iz imena razvidno, je ta jezik uporabljen za ustvarjanje zgradbe objektov.

V tem jeziku imamo tudi več podatkovnih tipov, a moramo sprva določiti, na katero spremenljivko bodo vrednosti določene, kar je zapisano na levi strani v narekovajih. V določenih primerih JSON-a je tudi možno, da nimamo narekovajev, a je to popolnoma odvisno od našega okolja.

Prvi podatkovni tip, ki ga lahko omenimo, je število. To število je lahko celo število ali pa tudi decimalno. Po tem imamo "boolean" tip, ki ga predstavljajo vrednosti "true" in "false". Ti tipi po navadi uporabijo najmanj spomina v računalniku in so uporabljeni v primerih, kjer moramo vedeti, ali je nekaj v enem stanju ali drugem. Temu sledi niz, ki je le seznam znakov, kot so črke, števila, itd. Nato imamo seznam, ki je zapisan z uporabo oglatih oklepajev in lahko vsebuje več vrednosti vseh možnih podatkovnih tipov. Na koncu še imamo objekt, ki je definiran z zavitimi oklepaji in ima lahko, kot seznami, več vrednosti raznih tipov, a morajo te vrednosti zdaj imeti svojo spremenljivko.

Zadnja stvar, vredna omembe, je to, da se datoteka v JSON formatu navadno začne z zavitimi oklepaji. To je zato, ker celotna datoteka predstavlja en objekt, a bi lahko tudi začeli z oglatimi oklepaji, če bi hoteli definirati seznam, v katerega bi nato lahko dali več vrednosti, med katerimi bi lahko bili tudi objekti, kar pomeni, da bi lahko imeli seznam objektov.

2.3 Postopek priprave in izdelave

Najprej smo si vsi namestili programsko okolje in poskrbeli, da je ustrezno povezano z Andoid Studijem. Za vse različne zaslone skozi katere navigira uporabnik smo morali prav te povezati. To smo naredili s “stack navigatorjem”, ki poskrbi, da lahko za vsak zaslon napišemo kodo v posebej datoteko. Ustvarili smo nekaj JavaScript datotek, vsako za svoj zaslon. Vanje smo najprej uvozili vse potrebne knjižnice, ki smo jih potrebovali. Zatem so sledili uvozi komponent in globalnih spremenljivk iz knjižnic ali globalne JSON datoteke. Od te točke naprej je sledil »markup«, ki predstavlja elemente na zaslonu. Na koncu je bil še CSS-u podoben seznam slogov za oblikovanje in urejanje elementov. Podatke moramo seveda nekam shranjevati, zato smo dodali še knjižnico Realm, s katero smo kreirali podatkovno bazo, v katero bomo lahko zapisovali podatke. Funkcije in metode za interaktivnost uporabnika z aplikacijo smo napisali znotraj razreda posameznega zaslona. Prav tako bi z njimi brali in vpisovali podatke iz in v podatkovno bazo.

2.4 Namestitev okolja

Najprej smo morali namestiti program NodeJS. To je programska oprema, ki nam omogoča gradnjo aplikacij na osnovi jezika JavaScript. Za tem smo namestili aplikacijo Android Studio, ki nam omogoča programiranje in pogled na emulator. Za konec pa smo še namestili Visual Studio Code, saj je dostop do konzole s tem programom lažji in koda ima preprostejši videz.

2.4.1 Namestitev knjižnice

Zdaj, ko smo imeli vse te programe nameščene, smo morali omogočiti, da bo naš sistem podpiral programiranje v jeziku React-Native, zato smo namestili Chocolatey, ki je najboljši upravitelj paketov za operacijski sistem Windows. To smo naredili tako, da smo odprli Administrator Windows PowerShell in smo tam napisali sledeče:

```
Set-ExecutionPolicy Bypass -Scope Process -Force;
 [System.Net.ServicePointManager]::SecurityProtocol =
[System.Net.ServicePointManager]::SecurityProtocol -bor 3072; iex ((New-Object
System.Net.WebClient).DownloadString('https://community.chocolatey.org/install.ps1'))
```

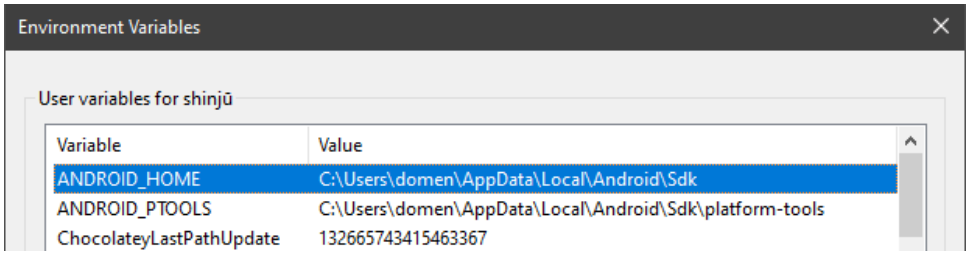
Tako se je paket samodejno namestil. Za tem smo še morali napisati:

```
Choco install –y nodejs.install openjdk8
```

Ta koda nam je omogočila, da programiramo s pomočjo JavaScript.

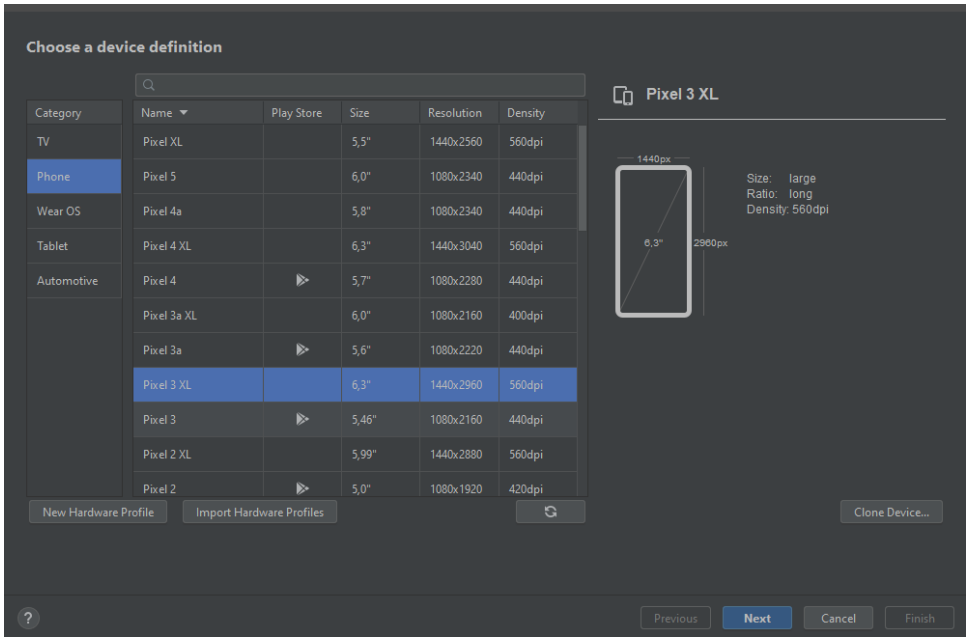
2.4.2 Namestitev emulatorja

Prej smo že namestili program Android Studio, vendar smo ga za tem morali nastaviti. V Windows nastavitvah za spremenljivke okolja smo morali dodati dve novi spremenljivki za ANDROID_HOME in ANDROID_PTOOLS, ki sta nam omogočili uporabo emulatorja na svojem sistemu.



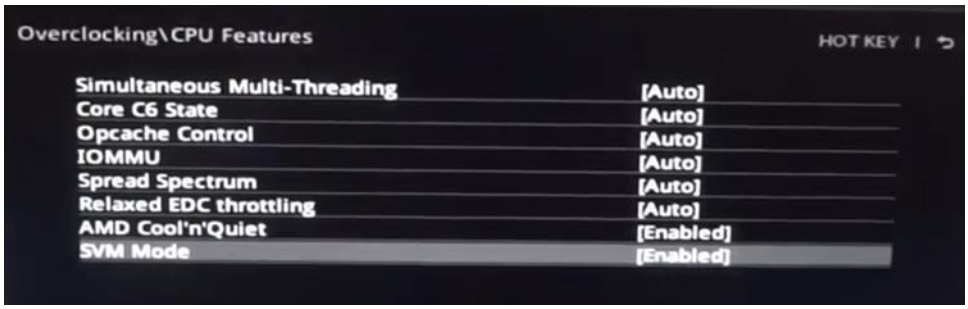
Slika 4 - Dodajanje spremenljivk v Windows

Po tem smo se lahko odločili za virtualno napravo, na kateri bomo programirali svojo aplikacijo. Android Studio ima široko platno za takšne naprave, ki so različice Pixel naprav podjetja Google. Na koncu smo se odločili, da bomo uporabili Pixel 3 XL.



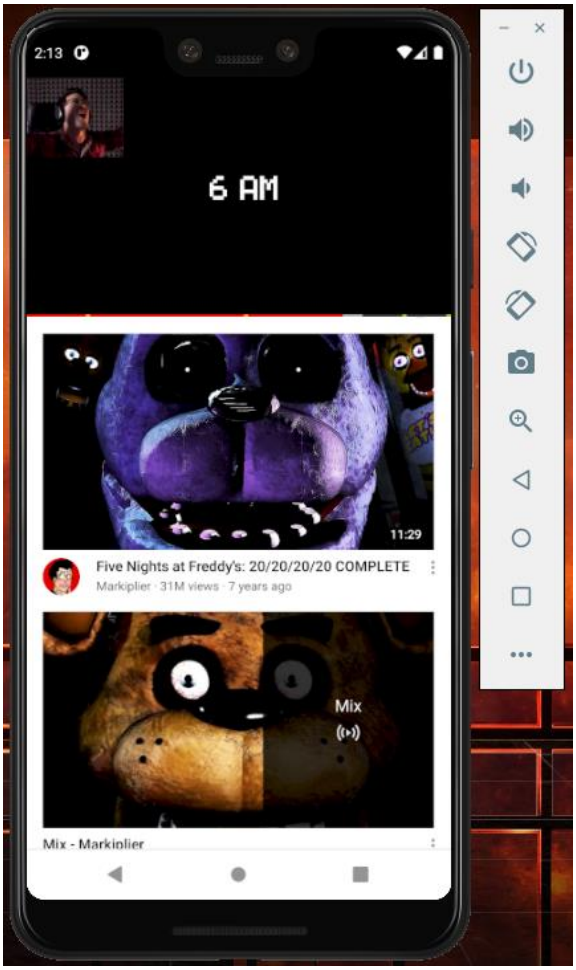
Slika 5 - Izbiranje naprave za emulacijo

V primeru uporabe Intel procesorja smo zdaj končali namestitev emulatorja, vendar smo se uporabniki Ryzen procesorja srečali z novo težavo, ki nas je vzela kar nekaj časa za ugotovitev rešitve. Procesor nam ni omogočal uporabo emulatorja, zato smo morali spremeniti njegove nastavitve, do katerih smo lahko dostopali samo v BIOS-u. Tam smo morali vklopiti SVM, kar nam omogoča virtualizacijo procesorja.



Slika 6 - Omogočanje virtualizacije v BIOS-u za Ryzen procesor

Za tem smo lahko končno zagnali svoj emulator, ki deluje tako kot navadna mobilna naprava.



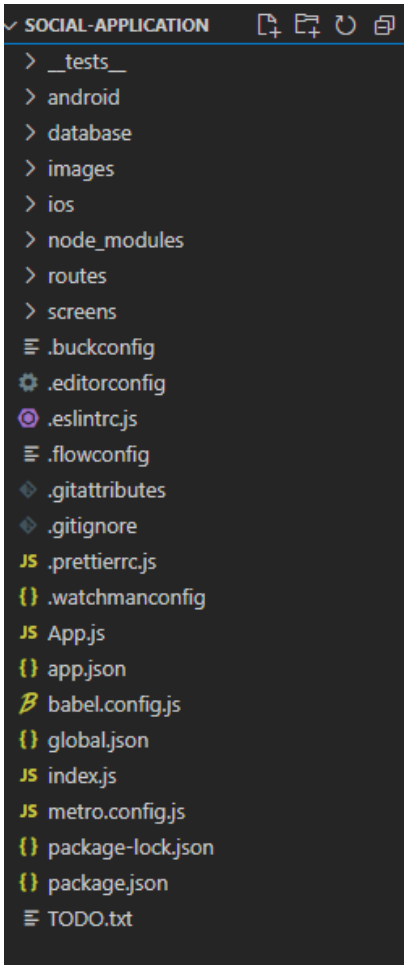
Slika 7 - Delujoč emulator

2.4.3 Osnovne datoteke

Po tem smo zagnali aplikacijo Visual Studio Code in izbrali lokacijo mape, kjer želimo imeti svojo aplikacijo nameščeno. Izbrali smo terminal in napisali naslednjo kodo:

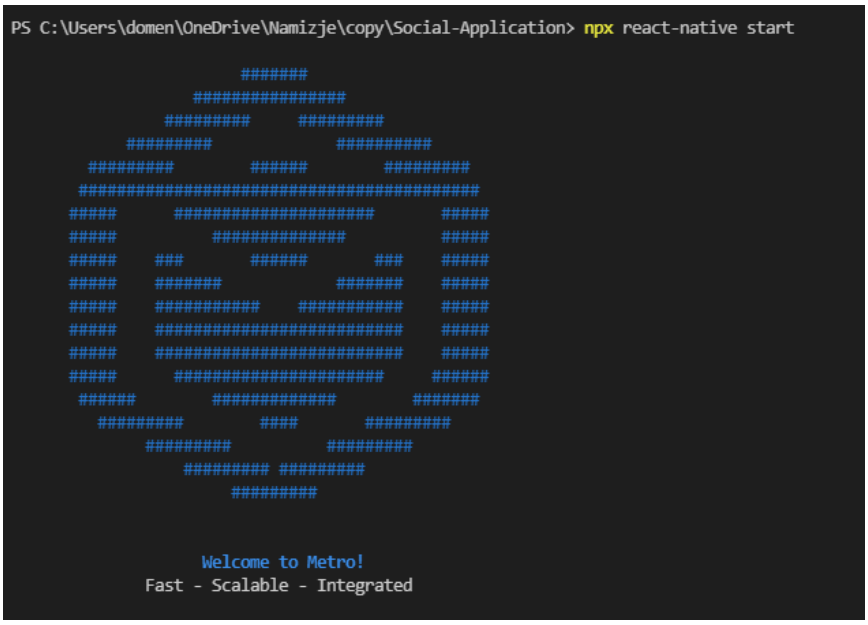
`npx react-native init imeProjekta`

S tem smo namestili osnovne datoteke projekta in tudi poimenovali svoj projekt.



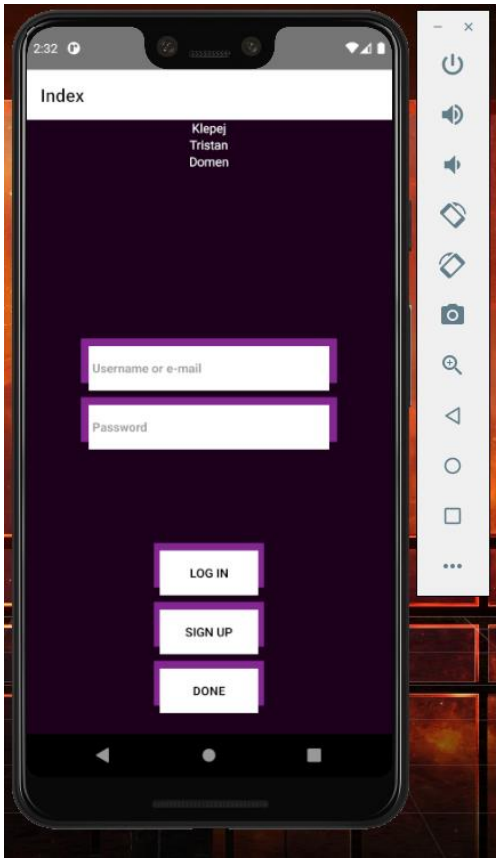
Slika 8 - Datoteke in podmape projekta

Zdaj smo lahko tudi začeli programirati svojo kodo. V terminal smo lahko napisali zdaj kodo `npx react-native start` in s tem se je začel izvajati “metro,” tj. projekt.



Slika 9 - Uspešen zagon projekta

Če smo odprli novi terminal in napisali npx react-native run-android, se nam je tudi prikazal projekt na emulatorju.



Slika 10 - Projekt na emulatorju

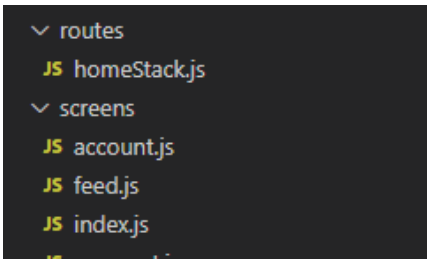
2.5 Postopek izdelave

Zdaj, ko smo imeli pripravljeno naše okolje in smo lahko z le enem ukazom odprli naš emulator, smo začeli pisati kodo. Začeli smo z implementacijo Stack-Navigation, ki nam omogoča, da se premikamo iz enega zaslona na drugega.

2.5.1 Stack-Navigation

Za uporabo Stack-Navigation smo morali najprej namestiti knjižnico, ki nam prav to omogoča. To smo naredili z ukazom `npx react-native install ime_knjižnice`.

Morali smo narediti datoteko s končnico ».js«, ki je namenjena za prehod med zasloni in podobne datoteke za prav te zaslone. V »routes« mapi smo naredili datoteko, v kateri je napisana koda, ki nam definira nekaj konstantnih vrednosti, te nam pa povejo, kje so shranjeni naši zasloni.



Slika 11 - Prikaz datotek in map v Visual Studio raziskovalcu datotek

Najprej smo implementirali knjižnico in zaslone, ki jih bomo uporabljali. Naredili smo »const screens«, ki je konstantna spremenljivka, v kateri so zapisani zasloni in njihova imena. Uporabili smo tudi »const HomeStack«. To je metoda iz knjižnice, ki nam omogoča prehod med zasloni. Na koncu smo še pa izvozili navigacijo, ki smo jo lahko uporabili na ostalih zaslonih.

Ker se bo »App.js« datoteka vedno ob zagonu izvedla, je potrebno navigacijo implementirati v prav to datoteko.

Tako bo zdaj vedno prikazano tisto, kar je navedeno v »homestack.js«. Ker je na prvem mestu »index«, se bo najprej prikazal »index«, za tem se pa lahko sklicujemo na ostale zaslone in jih prikažemo z »onPress« metodami ali pa podobno.

3 OPIS REŠITVE

V React-Native jeziku stvari deluje tako, da imamo več “zaslonov” in vsak od teh predstavlja eno stran naše aplikacije. Poleg tega še imamo lahko več datotek, ki dodajo drugo funkcionalnost, kot je na primer povezanost s podatkovno bazo.

3.1 Tehnični pogled

3.1.1 Podatkovna baza

V osnovi React-Native nima dostop do podatkovnih baz in so zato, kot za veliko drugih funkcij, potrebne knjižnice. Teh ima za podatkovne baze jezik veliko, kot so SQLite, PouchDB, Async Storage, itd. Med vsemi temi smo se mi odločili za Realm, ker ima unikaten način dostopa do podatkovnih baz in je tudi bil videti zaradi tega malo lažji od drugih.

3.1.2 Ustvarjanje baze

Prva stvar, ki jo moramo narediti, je to, da sploh omogočimo uporabo Realm-a. Da naredimo to, napišemo »import« stavek in se s tem v bistvu že ustvari podatkovna baza. Edina stvar, ki nam zdaj še ostane, je to, da ustvarimo naše tabele in z njimi komuniciramo preko ukazov, ki jih lahko sproži uporabnik.

3.1.3 Delo s tabelami

Kot je že bilo omenjeno, ima Realm unikaten način, kako se pristopa do podatkovnih baz. To je razvidno iz tega, da moramo najprej ustvariti podatkovno strukturo v JSON formatu, ki se imenuje shema, ta pa predstavlja, kakšni podatki bodo v naši tabeli.
Recimo, da bi hoteli ustvariti tabelo »User« za uporabnike. Sprva moramo narediti razred, ki podeduje od razreda »Realm.Object«, saj bo tako pridobil vse lastnosti, ki jih potrebuje, da lahko deluje kot tabela. Ko to naredimo, definiramo shemo tabele, tako da nastavimo spremenljivko »User.schema« na JSON objekt. V tem objektu najprej definiramo spremenljivko »name«, ki nam pove ime tabele, nato pa določimo, kaj bo njen primarni ključ s spremenljivko »primaryKey«, ki je v tem primeru nastavljen na “id”. Zadnja stvar, ki je nujna, je spremenljivka »properties«, ki programu našteje vse podatke, ki bodo prisotni v tabeli in njihove podatkovne tipe. V primeru spremenljivke »username«, ki je uporabljena za uporabniško ime uporabnikov, imamo pri njej določeno besedo »string«, kar pomeni, da bo ta stolpec v tabeli imel podatkovni tip niza. Zadnja stvar, vredna omembe, je to, da je prva

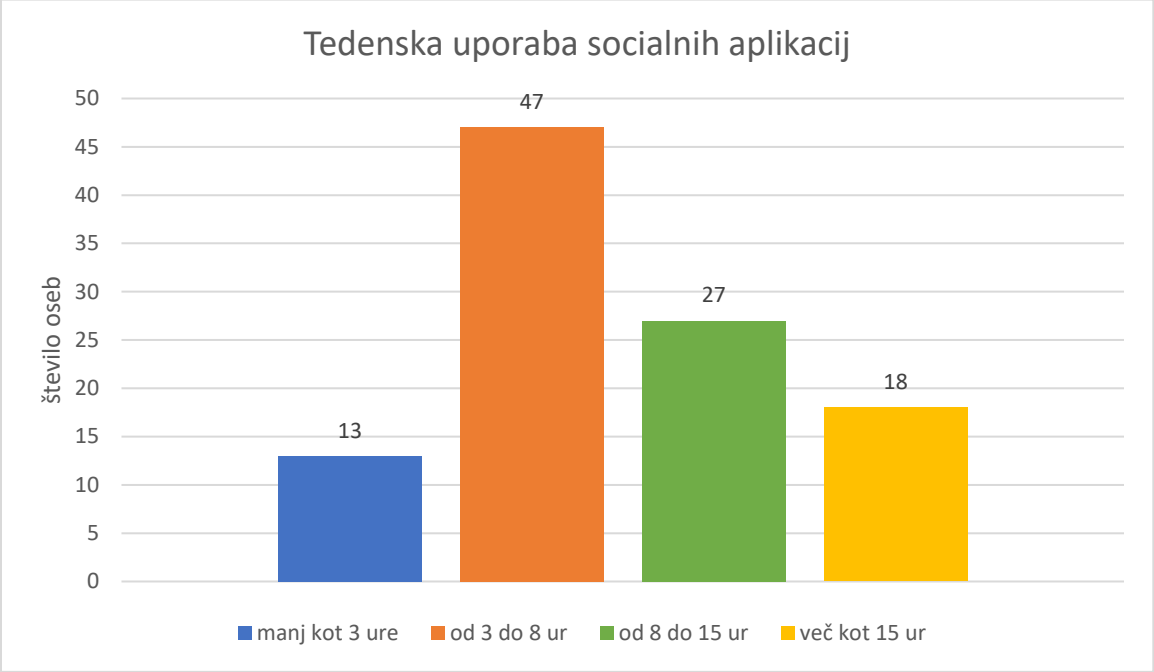
spremenljivka v »properties« poimenovana »id«, kar pomeni, da bo to primarni ključ te tabele, saj je bilo tako določeno v drugi vrsti sheme.

Zdaj, ko smo naše tabele in njihove sheme ustvarili, je čas, da vanje začnemo vnašati podatke in da začnemo te podatke tudi brati. Način, kako je to delo videti, je lahko zelo znano tistim, ki že imajo izkušnje z jezikom SQL, saj Realm deluje na zelo podoben način, a namesto, da pišeš kodo v obliki SQL ukazov, samo pokličeš metode, ki so namenjene funkciji, ki jo hočeš opraviti. Za primer bi lahko dali »write« metodo, ki je uporabljena za zapisovanje podatkov v tabelo.

4 ANALIZA ANKETE

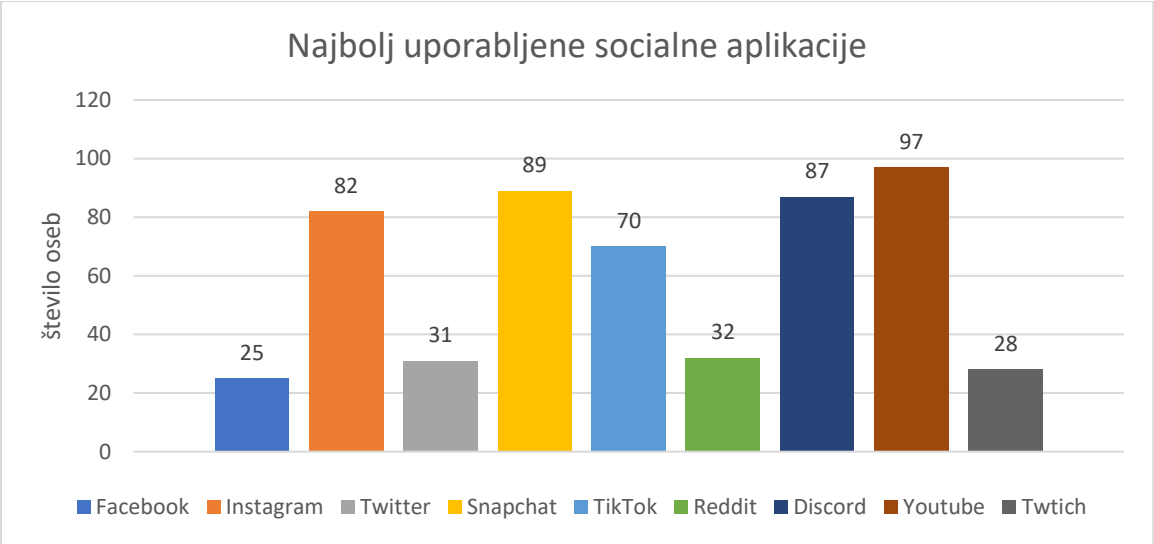
Za vrsto raziskovalne naloge smo uporabili anketo narejeno na spletni strani 1KA. Anketo je rešilo 105 anketirancev, večina od teh je bila srednješolcev. Odgovarjali so na vprašanja o izgledu, uporabi, delovanju in pomembnosti lastnosti socialnih aplikacij.

Najprej smo anketirance vprašali, koliko časa na teden uporabljajo socialne aplikacije. Rezultati nam povejo, da večina ljudi uporablja socialne aplikacije od 3 do 8 ur na teden, veliko pa tudi od 8 do 15 ur na teden in več kot 15 ur na teden. S temi rezultati vidimo, da uporabniki porabijo veliko časa zanje in jim je zato pomembno, da so jim aplikacije čim bolj všeč.



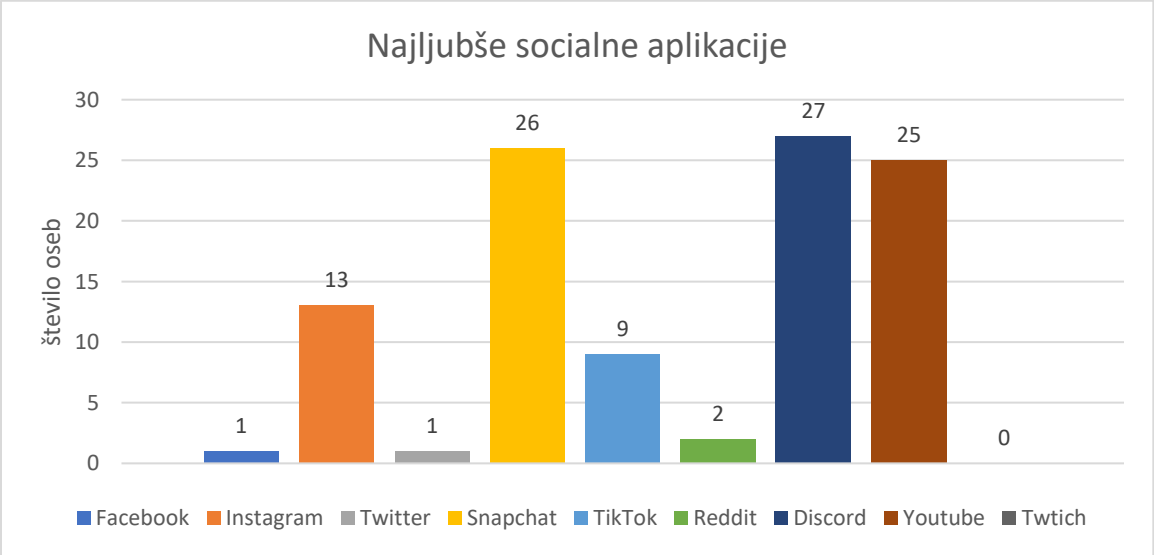
Graf 1 - Tedenska uporaba socialnih aplikacij

Pri vprašanju, katere socialne aplikacije izmed naštetih najbolj popularnih uporabljajo, je večina izbrala aplikacije s preprostim izgledom, primarnim namenom deljenja slik in videoposnetkov ter pogovarjanja s prijatelji. V tej smeri smo tudi sami razmišljali, da bi naredili aplikacijo s takšnimi lastnostmi.



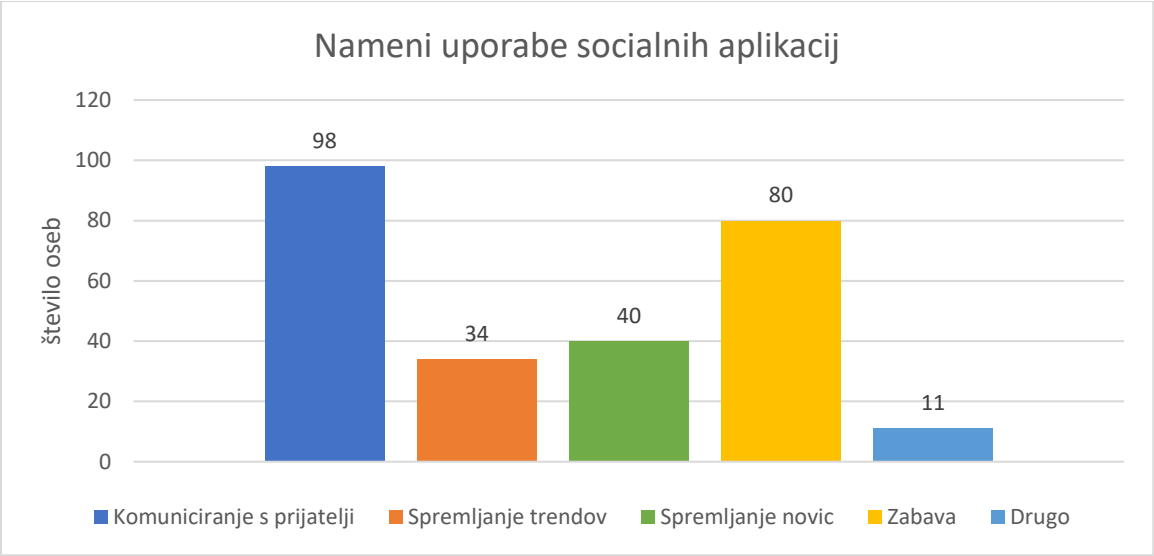
Graf 2 - Najbolj uporabljene socialne aplikacije

Najljubše socialne aplikacije anketirancev se ujemajo z odgovori na prejšnje vprašanje ankete, kjer smo videli, da najbolj uporabljajo Snapchat, Discord in YouTube. Instagram ter TikTok sta pri tem vprašanju nekoliko manj izbrana, zato lahko sklepamo, da so nekoliko več uporabljeni z namenom sledenja internetnim vplivnežem ali prijateljem.



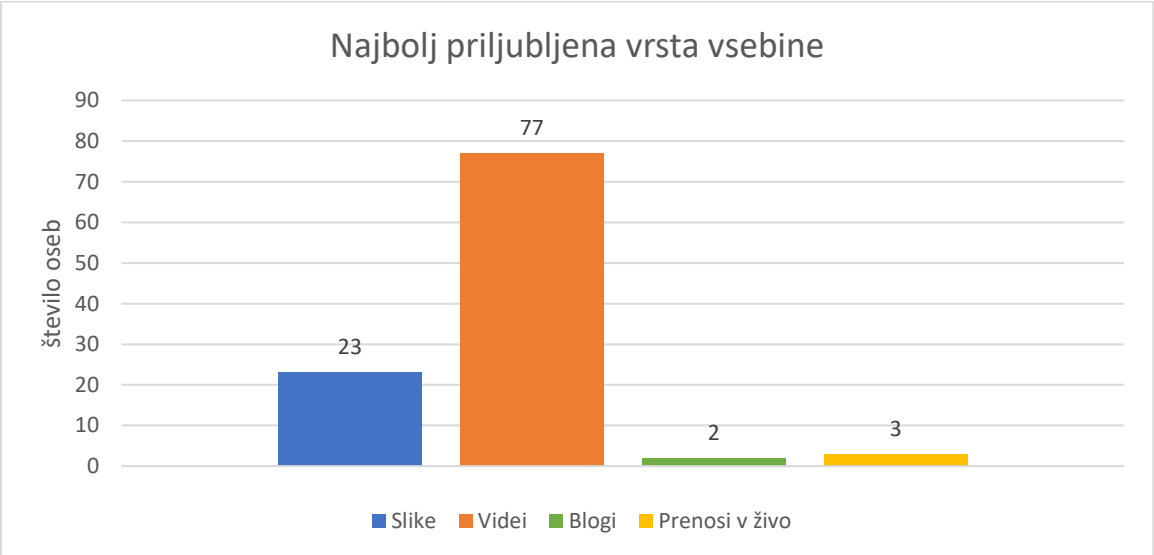
Graf 3 - Najljubše socialne aplikacije

Komuniciranje s prijatelji ter zabava sta se izkazala za najpogostejša namena uporabe aplikacij, spremljanje trendov ter novic pa nekoliko manj. Nekateri anketiranci so pod drugo napisali dopisovanje z dekleti kot namen uporabe, kar lahko v našem primeru štejemo kot dopisovanje s prijatelji.



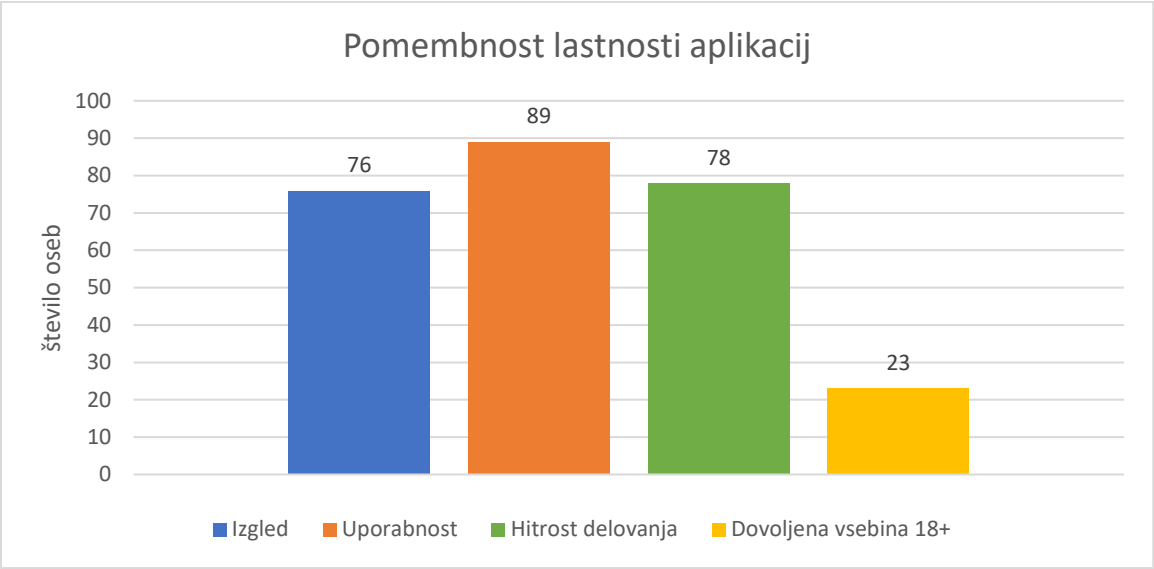
Graf 4 - Nameni uporabe socialnih aplikacij

Za najbolj priljubljeno vrsto vsebine so se izkazali videi, nekaj manj pa še slike. Blogi ter prenosi v živo so najmanj priljubljene vrste, kar smo pričakovali.



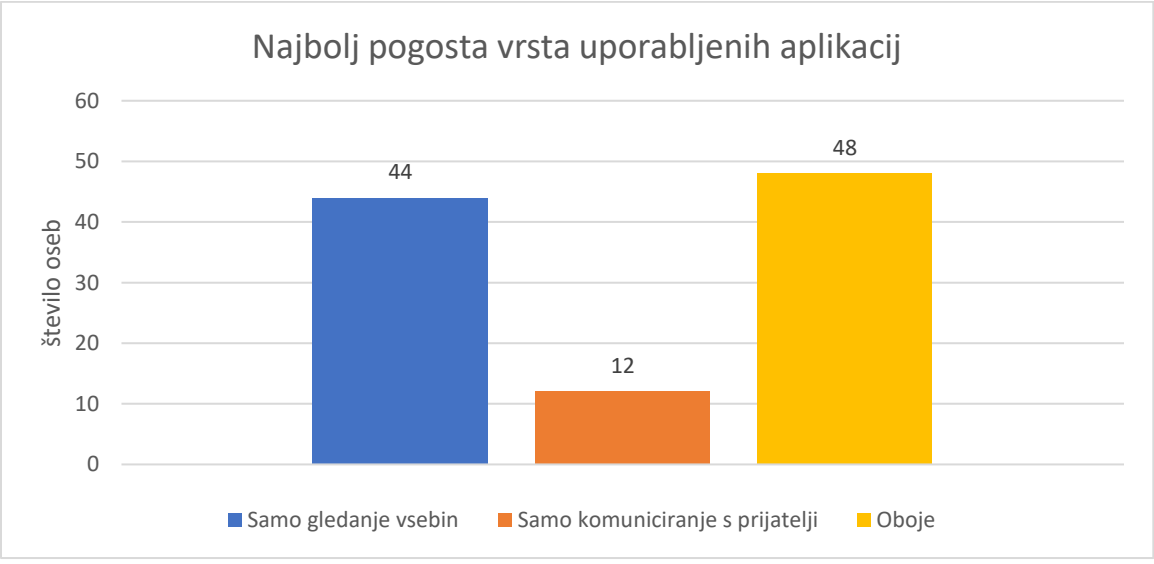
Graf 5 - Najbolj priljubljena vrsta vsebine

Pri vprašanju, katere lastnosti aplikacije so vam pomembne, je večina izbrala izgled, uporabnost in hitrost delovanja. Manjši skupini ljudi je pomembno, da je na aplikaciji dovoljena odrasla vsebina.



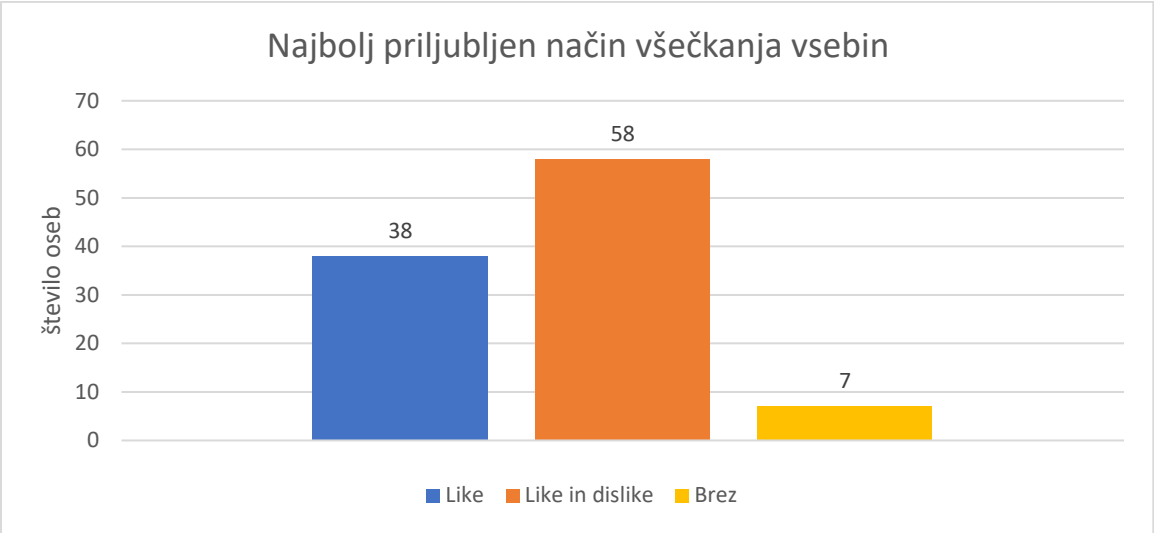
Graf 6 - Pomembnost lastnosti aplikacij

Aplikacije, ki omogočajo gledanje vsebin ter komuniciranje s prijatelji, in aplikacije, ki omogočajo samo gledanje vsebin, so se izkazale za najbolj uporabljene pri anketirancih. Manj so priljubljene aplikacije, s katerimi lahko samo komuniciramo s prijatelji in nimajo možnosti gledanja javnih vsebin.



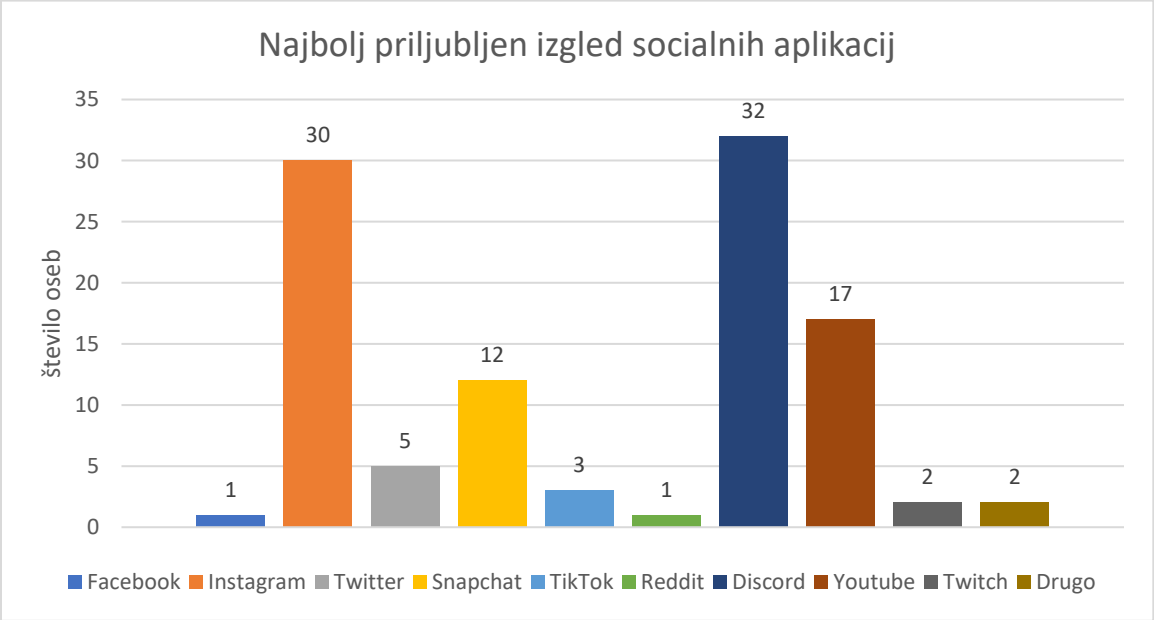
Graf 7 - Najbolj pogosta vrsta uporabljenih aplikacij

Najljubši aplikaciji za vrsto všečkanja vsebin bi bili na primer YouTube in Reddit z možnostjo »like« in »dislike«. Nekoliko manj pa Twitter ter Instagram z izključno možnostjo »Like«. Sama izbira všečkanja je zaželena med anketiranci, zato Twitch in 4chan ne bi bila izbrana.



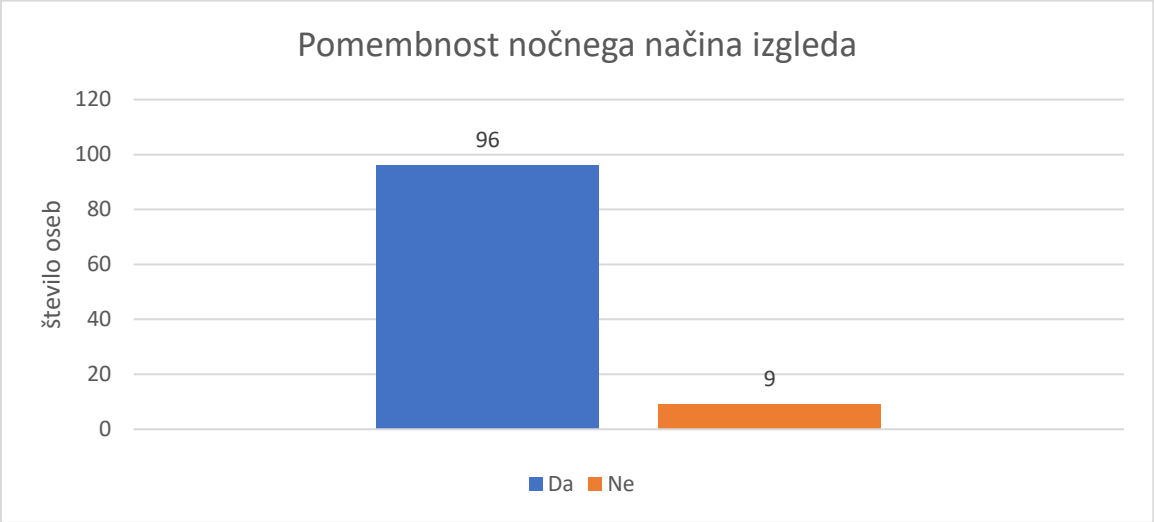
Graf 8 - Najbolj priljubljen način všečkanja vsebin

Za najbolj priljubljeno socialno aplikacijo glede na sam izgled se je izkazal Discord, takoj za njim pa Instagram. YouTube in Snapchat sta tudi bila nekoliko večkrat izbrana od ostalih aplikacij, kar spet potrjuje, da je uporabnikom všeč preprost izgled.



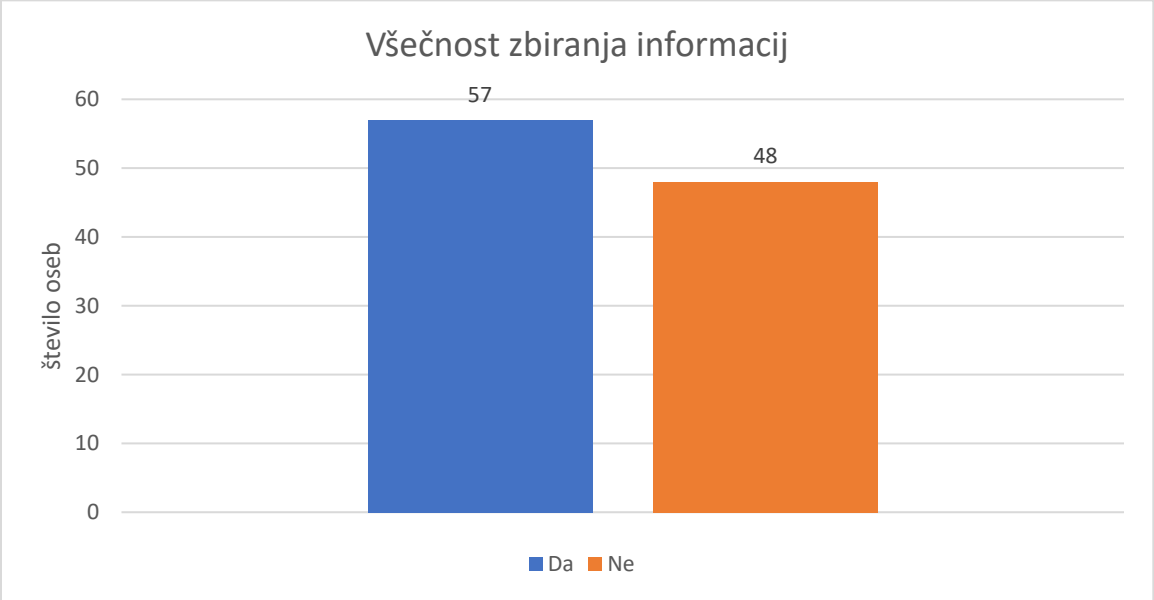
Graf 9 - Najbolj priljubljen izgled socialnih aplikacij

Z vprašanjem, ali je uporabnikom pomembno, da ima aplikacija nočni način, se je pokazalo, da je veliki večini pomembna ta možnost izgleda, zato smo jo implementirali v našo aplikacijo.



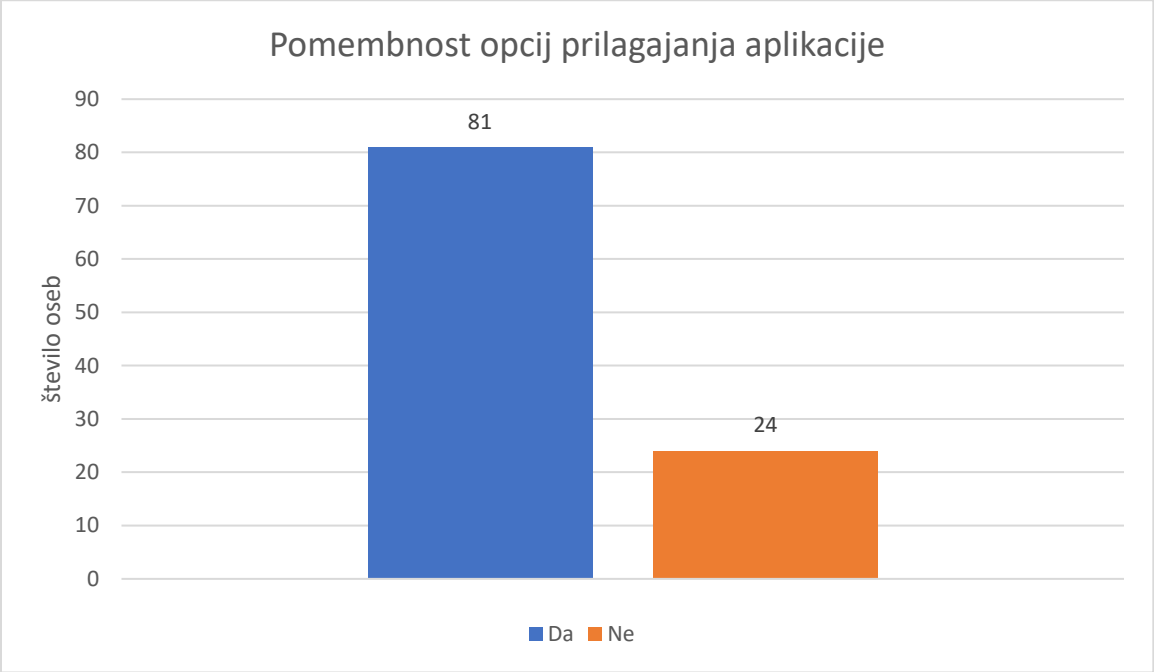
Graf 10 - Pomembnost nočnega načina izgleda

Pri naslednjem vprašanju smo ugotovili, da ima večina raje, da aplikacija zbira podatke uporabnika, s katerimi lahko predlaga podobne vsebine. Ta rezultat nas je še posebej presenetil, saj smo bili prepričani, da večini to ni všeč in se temu izogibajo, če se lahko.



Graf 11 - Všečnost zbiranja informacij

Pri zadnjem vprašanju smo povprašali še po pomembnosti izbir prilagajanja aplikacij in večina je odgovorila, da jim je pomembno, kar se ujema z rezultati vprašanja o pomembnosti nočnega načina.



Graf 12 - Pomembnost izbir prilagajanja aplikacije

Na koncu smo dali anketirancem še možnost, da nam sporočijo še predloge ali mnenja, ki bi pripomogla k naši izdelavi socialne aplikacije. Poudarjeni so bili predlogi o minimalističnem izgledu aplikacije ter implementaciji nočnega načina. Omenjena sta tudi bila predloga o uporabniškem nagrajevanju vsebin drugih ter optimizacija aplikacije za okolja s slabšim ali omejenim dostopom do interneta.

4.1 Analiza hipotez

Glede na odgovore, ki smo jih pridobili s spletno anketo, smo ugotovili, da je bilo večina naših hipotez potrjenih.

Prvo hipotezo, da večina ljudi uporablja socialne aplikacije za komuniciranje s prijatelji in zabavo z uporabo videoposnetkov in slik, lahko potrdimo z odgovori na peto in sedmo vprašanje naše ankete, kjer se rezultati ujemajo z našimi pričakovanji.

Drugo hipotezo, da je večini najljubše, da nimajo aplikacije kompleksnega videza in uporabe, pomemben jim je prilagodljiv videz, kot je na primer nočni način, lahko prav tako potrdimo, saj se rezultati odgovorov na tretje, šesto, deseto in dvanajsto vprašanje spet ujemajo z našimi predvidevanji, prav tako pa so to potrdili anketiranci z dodatnimi predlogi in mnenji na koncu ankete.

Tretjo in zadnjo hipotezo, da si uporabniki ne želijo, da bi aplikacije zbirale njihove podatke, čeprav zgolj za usmerjeno oglaševanje in priporočanje podobnih vsebin, pa moramo zavreči, saj se je izkazalo, da bi bilo večini bolj všeč, če bi aplikacija zbirala njihove podatke, če bi to pomenilo, da jim je ponujena podobna vsebina glede na njihova prejšnja iskanja vsebin.

5 ZAKLJUČEK

V prvem delu naloge smo opisali okolje, v katerem smo izdelali aplikacijo, programsko opremo, ki smo jo uporabili za pisanje kode in emulator, s katerim smo istočasno videli razlike izgleda in funkcionalnosti aplikacije med programiranjem. Prav tako smo opisali namestitev okolja, knjižnic, emulatorja ter osnovnih datotek. Nadaljevali smo s postopkom izdelave in »stack navigation«, zatem je pa sledil opis rešitve s tehničnim pogledom v izdelavo aplikacije.

V drugem delu sledi raziskovalni del naloge, kjer smo anketirali 105 uporabnikov socialnih aplikacij. Z grafi smo predstavili rezultate in jih komentirali. Iz pridobljenih informacij smo izvedeli kakšne preferenca imajo anketiranci pri uporabi, izbiri in všečnosti aplikacij samih, njihovih različnih lastnostih s poudarkom na izgledu ter všečnostjo zbiranja podatkov uporabnikov aplikacij. Z rezultati smo potrdili dve hipotezi in eno zavrgli, ugotovitve, predloge in mnenja pa uporabili pri oblikovanju in izdelavi naše aplikacije.

Naučili smo se izdelave večjega projekta v okolju React Native ter povezovanje podatkovne baze z aplikacijo ter izdelavo poročila takšne naloge.

Manjše težave smo imeli z namestitvijo okolja in povezovanjem podatkovne baze, saj ima namestitev veliko različnih korakov, takšne podatkovne baze pa še nikoli prej nismo uporabili pri izdelavi mobilnih aplikacij.

V prihodnosti bi lahko projekt še razširili z izdelavo spletne aplikacije za računalnike in dodajanjem različnih funkcij, storitev, varnosti ter ostalih elementov aplikaciji sami.

6 VIRI

1ka.si. (12. februar 2022). Pridobljeno 20. marec 2022 iz https://www.1ka.si/a/57395467&preview=on

furulevi. (29. oktober 2019). *youtube.com*. Pridobljeno 20. marec 2022 iz https://www.youtube.com/watch?v=kSTNuYD0o5w

Mash, P. w. (24. november 2020). *youtube.com*. Pridobljeno 20. marec 2022 iz https://www.youtube.com/watch?v=LiHkAGyNSJU&list=PL8kfZyp--gEXs4YsSLtB3KqDtdOFHMjWZ&index=4

Native, R. (19. januar 2022). *reactnative.dev*. Pridobljeno 20. marec 2022 iz https://reactnative.dev/docs/environment-setup

wikipedia.org. (12. marec 2022). Pridobljeno 20. marec 2022 iz https://en.wikipedia.org/wiki/React_Native