Mobilna aplikacija za učenje Brailleovog pisma

Hlupić, Dominik

Sveučilište Jurja Dobrile u Puli

Fakultet Informatike

DOMINIK HLUPIĆ

MOBILNA APLIKACIJA ZA UČENJE BRAILLOVOG PISMA

MOBILE APPLICATION FOR LEARNING BRAILLE

Završni rad

Sveučilište Jurja Dobrile u Puli

Fakultet Informatike

DOMINIK HLUPIĆ

MOBILNA APLIKACIJA ZA UČENJE BRAILLOVOG PISMA

MOBILE APPLICATION FOR LEARNING BRAILLE

Završni rad

JMBAG: 0303069045, redovni student

Studijski smjer: Informatika

Znanstveno područje: Društvene znanosti

Znanstveno polje: Informacijske i komunikacijske znanosti

Znanstvena grana: Informacijski sustavi i informatologija

Kolegij: Programsko inženjerstvo

Mentor: doc.dr.sc. Nikola Tanković

IZJAVA O AKADEMSKOJ ČESTITOSTI

Ja, dolje potpisani *Dominik Hlupić*, ovime izjavljujem da je ovaj završni rad rezultat isključivo mojega vlastitog rada, da se temelji na mojim istraživanjima te da se oslanja na objavljenu literaturu kao što to pokazuju korištene bilješke i bibliografija.
Izjavljujem da niti jedan dio završnog rada nije napisan na nedozvoljen način, odnosno da je prepisan iz kojega necitiranog rada, te da ikoji dio rada krši bilo čija autorska prava. Izjavljujem, također, da nijedan dio rada nije iskorišten za koji drugi rad pri bilo kojoj drugoj visokoškolskoj, znanstvenoj ili radnoj ustanovi.

Student

Dominik Hlupić

U Puli, Rujan, 2022.godine

IZJAVA O KORIŠTENJU AUTORSKOG DIJELA

Ja, *Dominik Hlupić* dajem odobrenje Sveučilištu Jurja Dobrile u Puli, kao nositelj u prava iskorištavanja, da moj završni rad pod nazivom *Mobilna aplikacija za učenje Braillovog pisma* koristi na način da gore navedeno autorsko djelo, kao cjeloviti tekst trajno objavi u javnoj internetskoj bazi Sveučilišne knjižnice Sveučilišta Jurja Dobrile u Puli te kopira u javnu internetsku bazu završnih radova Nacionalne i sveučilišne knjižnice (stavljanje na raspolaganje javnosti), sve u skladu s Zakonom o autorskom pravu i drugim srodnim pravima i dobrom akademskom praksom, a radi promicanja otvorenoga, slobodnoga pristupa znanstvenim informacijama. Za korištenje autorskog djela na gore navedeni način ne potražujem naknadu.

Student

Dominik Hlupić

U Puli, Rujan, 2022. godine

SADRŽAJ

1. Uvod.....1

2. Motivacija.....2

2. Braille input- implementacija.....4

3. Flutter Widgets.....6

4.Firebase.....8

5. Dijagrami aplikacije.....10

6. Dodatni Widgeti.....15

7. Zaključak.....18

1. Uvod

Kako ljudi kroz vrijeme, provode sve više vremena za mobilnim telefonom, zaiskrila je znatiželja za razvojem mobilnih aplikacija. Google Flutter koji je korišten za završni rad, koristi jednu kodnu bazu za iOS, Android, Web, Windows, MacOS, Linux itd., imajući slobodan izbor paketa koji nam pomažu izgraditi sjajne mobilne aplikacije.
Za pokretanje aplikacije na računalu potrebne su instalacije koje se mogu svesti u nekoliko koraka: Preuzimanje i instalacija Flutter SDK(Software Development Kit), Android Studio i njegov SDK i postavljanje uređaja na kojem će se aplikacija pokrenuti.

Kroz razvoj aplikacije koja je izrađena u sklopu završnog rada – učenje brajice kroz igru, korišten je IDE(*Integrated development enviroment* ili u prijevodu *Integrirano razvojno okruženje*) Visual Studio Code(u nastavku VS Code). VS Code što nudi odličnu podršku za Flutter, gdje osnovno proširenje pokriva efektivno uređivanje, pokretanje i ponovno pokretanje aplikacija. Za pokretanje aplikacije u VS Code-u potrebno je odabrati *Run>Start Debugging* u glavnom prozoru. [1]

Temelj Fluttera je kodna baza, odnosno programski jezik Dart. Dart je *Type-Safe,* što ukratko znači radi provjeru da li vrijednost varijable odgovara tipu podataka. S druge strane, nudi upotrebu *dynamic* tipa podataka gdje je korisno tijekom eksperimentiranja.[2] Također nudi *Sound null safety* koji kod pristupa varijabli postavljenje na null zaustavlja grešku. Generalno, ako imamo metodu koja kao parametar prima cijeli broj, a dobije null, doći će do pogreške tijekom izvođenja aplikacije. *Sound null safety* označava u kodu kada god varijabla koja nije null nije inicijalizirana s vrijednošću koja nije null ili joj je dodjeljen null, što omogućuje ispravak pogrešaka prije implementacije aplikacije.[3]

U cijelini, Dart je odgovoran za razvoj brzih *front-end* aplikacija na bilo kojoj platformi, pritom koristeći brojne zbirke, bile matematičke, asinkrone ili HTTP.[2]

Link git repozitorija: https://github.com/Dhlupic-Stingray6/Braille_aplikacija.git

2.Motivacija

Braillevo pismo ili brajica je sustav pisma za slijepe osobe. Stvorio ga je Louis Braille 1829. godine. Pojedini znak koristi šest izbočenih točaka u schemi po tri točke u 2 okomita reda, s čime možemo dobiti 63 znaka. Od tih 63 znaka, 32 znaka su samostalno prepoznatljiva dok ostali 31 se mogu prepoznati samo uz predznak. Brojevi i velika slova se dobivaju s pomoću predznaka. Braille je također stvorio glazbene znakove. Brajica se čita pipajući prstima obiju ruku slijeva na desno, a pišu na tvrđem papiru s desna na lijevo. Braille koji je bio sam slijep od četvrte godine života, je djelovao kao učitelj slijepe djece i kao glazbenik. Tako je poznavao potrebe i mogućnosti slijepih i njegov sustav je prihvaćen u cijelome svijetu. Prvi Hrvatski udžbenik za slijepe je izradio Vinko Bek 1889. godine. [4]

Navedene točke za potrebe izrade aplikacije mogu se zabilježiti brojevima od 1 do 6, što će biti spomenuto u idućim poglavljima.

Danas slijepe i slabovidne osobe koriste mobilne uređaje koristeći TalkBack funkciju koja izgovara stavke na zaslonu. Korisnik se može kretati kroz uređaj pomoću TalkBack-a prstima ulijevo i udesno za pomicanje među stavkama, dvostrukim dodirom za odabir ili aktivaciju stavke i povlačenjem dvama prstima za pomicanje.
Odabir za govor je dodatna usluga koja dodirom određene stavke izgovara na glas ili pritiskom gumba za odlušavanje cijelog zaslona.

Krajni cilj aplikacije je potaknuti korisnike na upoznavanje sustava brajice i olakšati njeno savladavanje. Treba biti prilagođena za svakog korisnika, čak i za zdrave osobe koje žele pomagati drugim ljudima u upoznavanju brajice.

Danas je dosta mali postotak poznaje i koristi Braillevo pismo, kao najčešći razlog je nedostatak materijala za učenje i rehabilitatora. Učenje i korištenje brajice ima pozitivan učinak na komunikaciju slijepih osoba. Savladavanjem brajice omogućuje slijepim osobama samostalnost i snalažljivost u svakodnevnom životu.

Prije planiranja izrade aplikacije potrebno je postaviti pitanja : Treba li aplikaciju prilagoditi mobilnim uslugama kao što su Google Talkback ili izraditi aplikaciju koja bi slova na zaslonu izgovarala Text to speech? Podržava li Flutter Text to speech pakete koji će izgovarati sadržaj na hrvatskom jeziku?

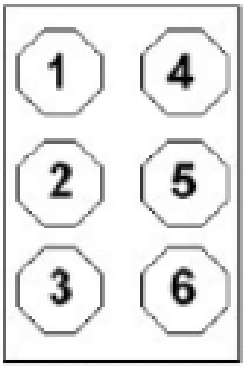
S druge strane, ako bi se motiviralo i zdrave osobe, javila bi se potreba za izradom kvalitetnog korisničkog sučelja u kontekstu razvoja igara, koji bi koristio prilagođene assets-e(sličice, zvukove, animacije) i interaktivni razvoj u igri. Pri tome se govori o osvajanju bodova, otključavanju novih razina i na kraju – pobjedi. Flutter nudi git repozitorij koji se koristi kao početni šablon u izradi flutter igara.

Sljedeća mogućnost nam može biti primjer strojnog učenja tj. klasifikacije. U slučaju da slijepa osoba naiđe na opipljiv dio teksta u brajici, fotografira taj tekst i od slike se pročita tekst koji se izgovori na glas te osoba sazna prijevod teksta. To bi u teoriji funkcioniralo kada bi se istrenirao training model u Tensorflow-u, konventirao u Tensorflow lite i integrirao u Flutter aplikaciju.

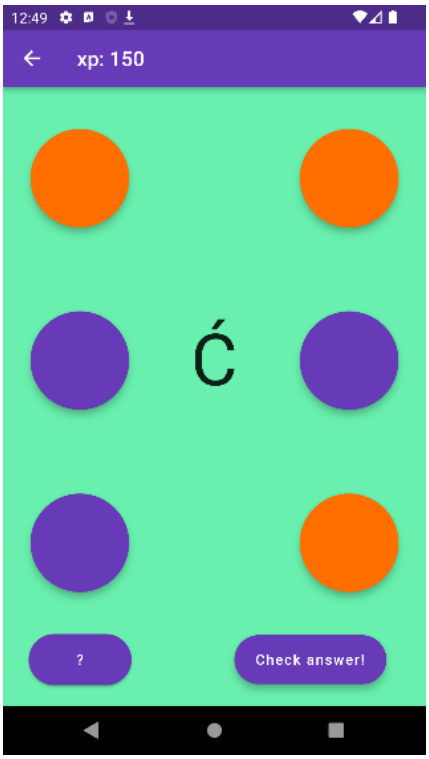
3. Implementacija

Ćelije jednog znaka brajice možemo indeskirati po brojevima od 1 do 6(slika 2), a stanje izbočenosti/udubljenja s 0 (udubljenje) i 1 (izbočenje) možemo izvući jedinstveno polje za svako slovo npr. List[1,0,0,1,0,0] bilo bi u prijevodu slovo C. [5]

Treba imati na umu da prvih 10 slova (a, b, c, d, e, f, g, h, i, j) dijele isti znak sa brojevima 0 do 9.[6]



Slika 1 -Braille ćelija[4]



Slika 2 - Isječak Braille aplikacije

Sljedeći korak nam je izraditi 2 polja. Prvo polje je id , sadrži 34 šesteroznamenkasta broja znakova 0 ili 1 tipa String, a drugo polje je znak slova,koji sadrži 34 slova kao također tipa String. Ta 2 polja ćemo povezati u jednu mapu u odnosu ključ-vrijednost, tj. id-slovo.

Prema potrebi, iz te mape možemo u pojedinu varijablu(*element*) inicijalizirati ključ i vrijednost traženog slova slučajnim odabirom, gdje možemo u isto vrijeme ispisati slovo na zaslon i koristiti njegov id za provjeru ulaza. Ulaz od 6 gumba(točaka) generira polje *list* od 6 članova brojeva 0 ili 1 tipa Integer. Pritiskom na gumb za provjeru to polje od šest članova se pretvori u jedan šesteroznamenkasti broj. Zatim kroz *for* iteraciju radimo usporedbu kroz if selekciju ulaza sa poljem indeksa iz mape:

```
var currentNum = list.join('');
        myLetter = '';
for (var i = 0; i < letters.map.length; i++) {
          if (currentNum == letters.id[i]) {
            myLetter = letters.letter[i];
          }
        }
```

For iteracija za formiranja inputa koji će se spremiti u varijablu myLetter

Nakon iteracije vrši se završna selekcija gdje se ustvrđuje da li je korisnik upisao točnu kombinaciju tj. znak ili nije:

```
if (element == myLetter) {

          _incrementPoints();
          rand(); //funkcija koja nam generira element-
                  // nakon uspješnog odgovora
                  // izvlači se novo slovo
          }
```

If selekcija element sa myLetter – ako je true, pozivaju se funkcija rand() za generiranje novog elementa. Funkcija _incrementPoints() će se opisati u sljedećim poglavljima

Ukratko, gumbi kao točke imaju pristup zasebnom polju od 6 brojeva. Svaki gumb mijenja zasebni indeks polja (ljubičasta predstavlja nulu, a naranđasta jedinicu) te pritiskom gumba za provjeru radi se usporedba prvoga polja s poljem od ponuđenog slova kao zadatak.

4. Flutter Widgets

„*Widgeti su središnja hijerarhija klasa u Flutter okviru.*“[6]

Widgeti su centralna hijerarhija klasa u Flutter aplikacijskom okviru. Oni daju opis elementima koji se prikazuju na zaslonu

Postoje 3 tipa klase widgeta – StatelessWidget, StatefulWidget&State i Inherited. U aplikaciji su korišteni Stateless i Stateful widget.

StatelessWidget

Stateless widget je widget koji nam opisuje dio korisničkog sučelja izgradnjom skupa drugih widgeta daju konkretnije opise. Stateless widget ili Widgeti bez stanja su korisni u slučaju da korisničko sučelje ovisi samo o vlastitim konfiguracijama. Ako imamo promjenjivu kompoziciju, bolje je koristiti StatefulWidget. [8]

```
class MyApp extends StatelessWidget {
  const MyApp({
    super.key,
    /*int xp, int level*/
  });

  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      debugShowCheckedModeBanner: false,
      navigatorKey: navigatorKey,
      theme: ThemeData(
          primarySwatch: Colors.deepPurple,
          primaryColorDark: Colors.amber[900]),
      home: MainPage(),
    );
  }
}
```

Primjer iz ovog završnog rada - MyApp StatelessWidget, koji je korijen aplikacije

StatefulWidget

Stateful wiget ima promjenjivo stanje - *state*. State se koristi informacija koja se učitava u isto vrijeme kada se gradi widget i također je promjenjiv tijekom životnog vijeka(*lifecycle*) widgeta. Kada se promjeni stanje u aplikaciji, pozove se setState() koja obavještava okvir da se widget opet izgradi. Pojednostavljeno, StatefulWidget se koristi kada postoji neka promjena te je potrebno promjeniti izgled. [9]

```
class Login extends StatefulWidget {
  final VoidCallback onClickedLogin;

  const Login({
    Key? key,
    required this.onClickedLogin,
  }) : super(key: key);

  @override
  State<Login> createState() => _LoginState();
}

class _LoginState extends State<Login> {
  final emailController = TextEditingController();
  final passwordController = TextEditingController();

...
```

Isječak iz koda, gdje je Login klasa StatefulWidget.

Inherited

Inherited widget se koristi kod prijenosa informacija kroz stabla. Kada se referenciraju, natjerat će svog potrošača da se ponovno izgradi kada i sam promjeni stanje. [10]

5. Firebase

Firebase je Googleova platforma za razvoj aplikacija, Backend as a service(BaaS).

Za potrebe završnog rada koristi se Firebase kao backend i baza podataka aplikacije.

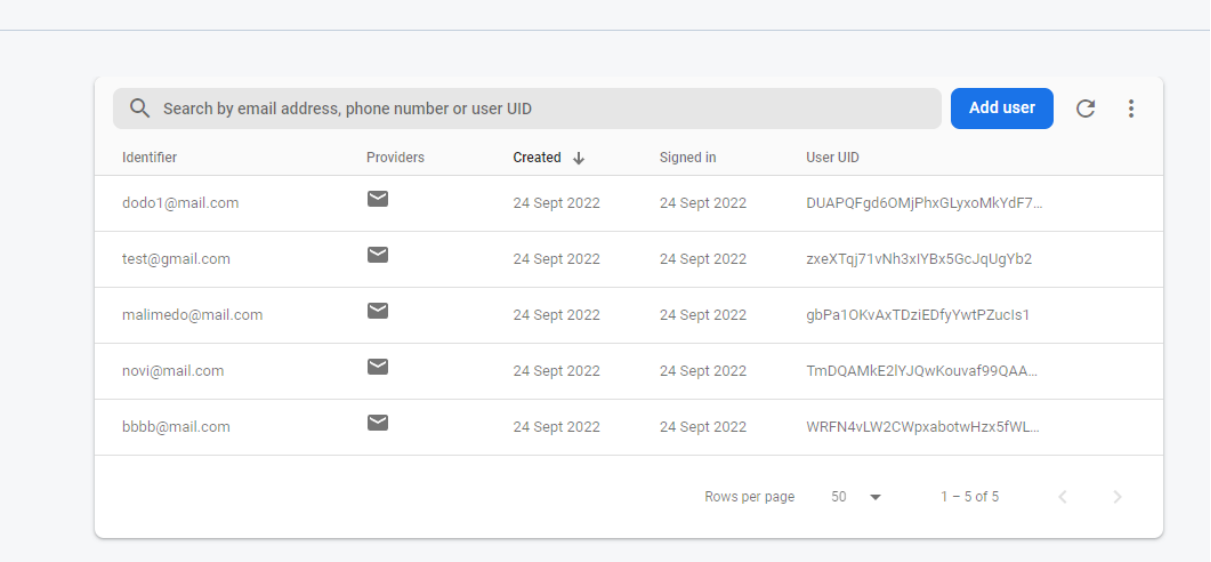
U svrhu aplikacije koriste se proizvodi Firebase authentication kao autentifikacija i Realtime Database kao baza podataka.

Firebase autentifikacija

Kada aplikacija poznaje identitet korisnika, to joj omogućuje spremanje korisničkih podataka u oblak ili bazu podataka. Firebase Authentication podržava autentifikaciju korištenjem zaporki, tel. brojeva ili nekih od popularnih pružatelja identiteta kao što su Google, Facebook i Twitter.

Da bi se korisnik prijavio u aplikaciju, potrebni su nam vjerodajnice. One mogu biti e-mail adresa i zaporka ili može biti token od pružatelja identiteta. Nakon uspješne se može pristupiti osnovnim korisničkim podacima te kontrolirati njegov pristup aplikaciji.

Ti korisnički podaci su email adresa kao identifikator, oznaka davatelja usluga(u slučaju slike je email adresa), datum kreiranja i datum prijave te njegov jedinstven UID.



Slika 3 – prikaz korisnika

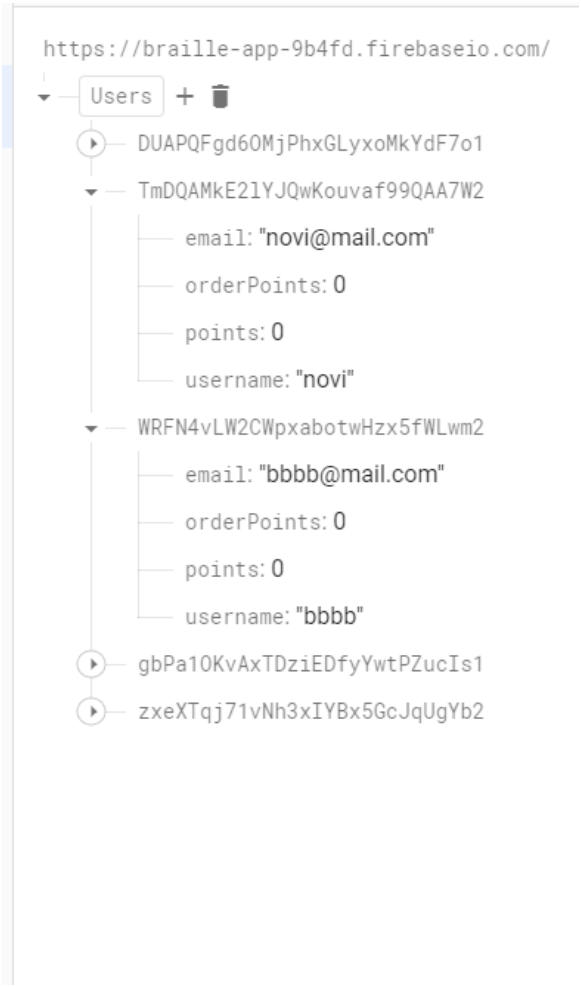
Osim što Firebase nudi mogućnost izrade novog korisnika preko konzole, moguće je svakog korisnika obrisati, resetirati šifru i onesposobiti račun.[11]

Firebase baza podataka

Firebase Realtime Database NoSQL baza podataka u oblaku. Podaci su spremljeni u obilku JSON-a (JavaScript Object Notation) i sinkronizirana je svakom klijentu bez obzira na vrijeme. Kada se izrade Cross-platform aplikacije, svaki kljent ima pristup toj istoj bazi i njezinim ažuriranjima

Pristup i manipuliranje podacima je moguć pomoću refenci JSON podataka kao što je „user/{UID}/email“.

Osim upravljanjem podacima, moguće je uspostaviti pravila pristupa pisanja i čitanja, omogućiti stvaranje sigurnosnih kopija i cijelokupno korištenje baze podataka u prikazu grafikona.[12]



Slika 4 – prikaz Firebase BP

Osim Realtime Database baze podataka, Firebase nudi i Cloud Firestore, također nonSQL baza podataka u oblaku. Cloud Firestore omogućuje preciznije i kvalitetnije upite, pohranu složenijih objekata i skalabilnost.[13]

6. Dijagrami aplikacije

U ovom poglavlju su slike dijagrama aplikacije rađenih u web alatu Lucidchart

6.1 Use-case dijagram

Use-case dijagram

Kratak opis dijagrama

Ovaj Use-case dijagram prikazuje moguće radnje koje korisnik može napraviti u aplikaciji.

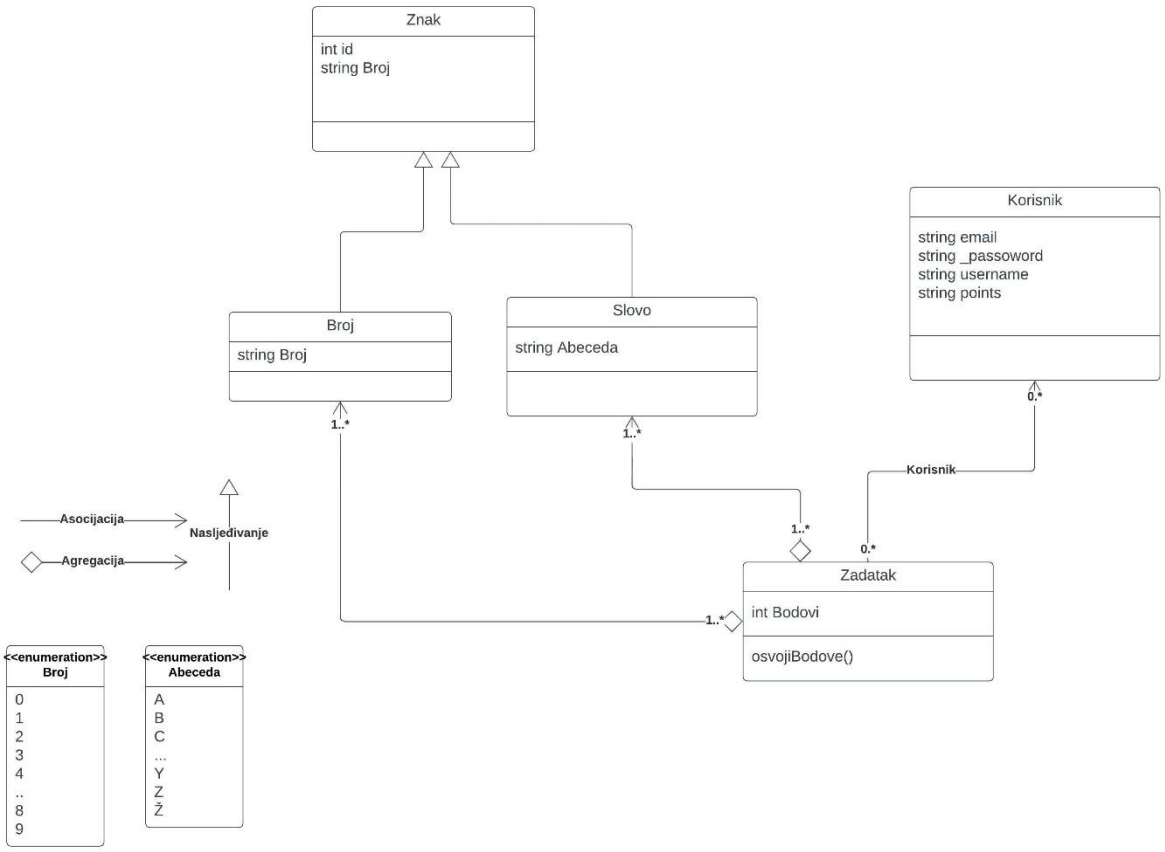
Prvo, korisnik kreira novi korisnički račun, upisuje email adresu, username kao nadimak, lozinku i potvrdu lozinke. Aplikacija može spriječiti kreiranje novog računa u slučajevima: korisnik nije upisao ispravan e-mail, lozinka ima manje od 6 znakova i ako se lozinke ne slažu tj. nisu identične. No, ako ima račun, samo se prijavljuje u aplikaciju sa emailom i lozinkom.

Nakon uspješne autentifikacije korisnik u izborniku može odabrati pregled ljestvicu i riječnika, postavke gdje se nalaze podaci o korisničkom računu te može odabrati drugi izbornik sa razinama igre.

Veza <<*extend*>> označava radnje koje korisnik može, ali i ne mora izvršiti. Na primjer u postavkama korisnik ima uvid u podatke i opcije brisanja bodova i računa, promjena nadimka(username) i odjave.

Veza <<*include*>> označava radnje koje uključuju u radnju. Na primjer kada korisnik rješi zadatak točnim, osvoji određen iznos bodova. U drugom slučaju korisnik kada obriše račun, automatski se odjavi iz aplikacije.

6.2 Class dijagram

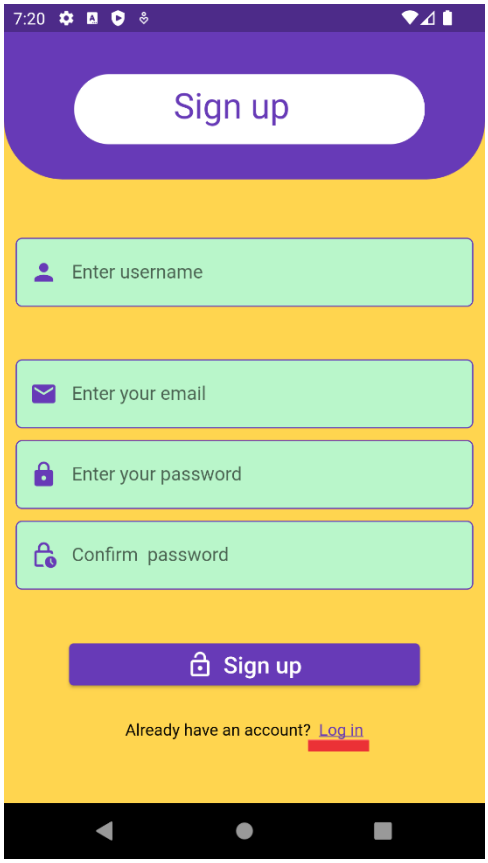


Opis dijagrama

Dijagram gore naveden predstavlja okrvirnu sliku widgeta. Kod pokretanja aplikacije klasa Mainpage u prati stanje instance FirebaseAuth koji djeluje kao ulazna točka. Ako korisnik nije prijavljen u aplikaciju, ona vraća klasu Login ili Signup kao početnu točku. Login ili signup to ovisi o Auth klasi tj. njezinom bool atributu. To znači da korisnik na Signup stranici može odabrati opciju da je već postojeći korisnik i želi se samo prijaviti u aplikaciji i obratno za Login stranicu.(Slika 4)

Zatim imamo drugu veću klasu *Settings* gdje imamo pristup funkcijama za brisanje korisničkog računa, izmjena korisničkog imena, postavljanje osvojenih bodova na nulu i odjava iz aplikacije.

Izbornik razina(*StartLearn*) prikazuje iznos bodova koje je korisnik osvojio(kod kreiranja korisničkog računa broj bodova je 0). Na temelju osvojenih bodova korisniku su zaključane/otključane razine tj. pristup klasama GameNumber i GameReverse.



Slika 7 – prikaz Signup stranice

Te tri klase (GameLetter, GameNumber i GameReverse) dijele zajedničku funkciju incrementPoints() za osvajanje bodova. Kod osvajanja bodova, korisnik kao klijent konstanto mjenja iznos bodova u Firebase Realtime Database bazu podataka. Kako bi se bodovi ispravno zbrajali, u svrhu aplikacije će se koristiti Transackija umjesto klasične promjene podataka.

```
Future<void> _incrementPoints() async {

    final ref = FirebaseDatabase(
            databaseURL: "https://braille-app-9b4fd.firebaseio.com/")
        .ref("Users/${user.currentUser!.uid}");

    TransactionResult result = await ref.runTransaction((Object? user) {
      if (user == null) {
        return Transaction.abort();
```

```
      }
      //
      Map<String, dynamic> _user = Map<String, dynamic>.from(user as Map);
      _user['points'] = (_user['points'] ?? 0) + 150;
      _user['orderPoints'] = (_user['orderPoints'] ?? 0) - 150;

      return Transaction.success(_user);
    });
  }
```

Funkcija _incrementPoints()

Prvo, inicijalizira se referenca baze od databaseURL-a i Trasankcija tj. TransactionHandler. Kod pokretanja transakcije ona uzima trenutno stanje objekta korisnika. Zatim trenutno stanje sprema u novu mapu, koju šalje u bazu podataka. O varijabli orderPoints se spominje u sljedećem poglavlju.

Prikaz bodova u aplikaciji:

Zamisao da se kroz svaku promjenu vrijednosti bodova u bazi, odmah ažurira na zaslonu, omogućuje nam Stream. Stream daje mogućnost čitanja podataka u stvarnom vremenu. Metoda onValue se pokreće kada dođe do promjene podatka, dok .listen omogućuje pretplatu na taj Stream. U navedenom primjeru pretplate, svaki put kada korisnik osvoji bodove, pretplata će pozvati setState funkciju, prepisati u varijablu, koja se kasnije koristi za prikaz:

```
showPoints() {
    final ref = FirebaseDatabase(
            databaseURL: "https://braille-app-9b4fd.firebaseio.com/")
        .ref("Users/${user.currentUser!.uid}/points");

    Stream<DatabaseEvent> points_stream = ref.onValue;

    points_stream.listen((DatabaseEvent event) {
      if (mounted) {
        setState(() {
          points = event.snapshot.value.toString();
        });
      }
    });
  }
```

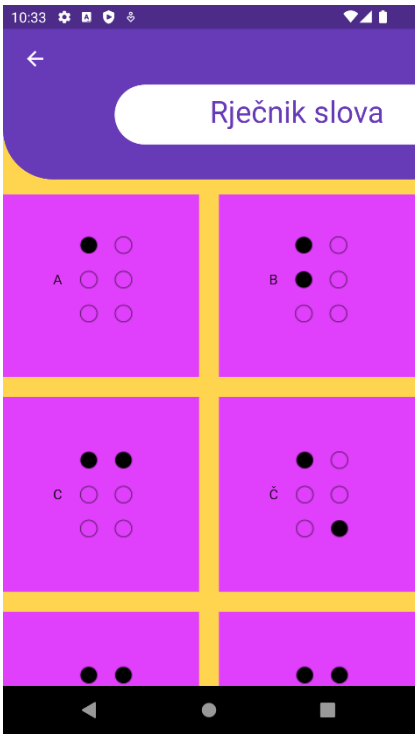
Funkcija za prikaz bodova u aplikaciji.

7. Dodatni widgeti

U ovom poglavlju su navedeni izrađeni Leaderboard i Dictionary widgeti.

Rječnik

Rječnik slova je jednostavan widget koji prikazuje slovo i njegov pripadajući znak u obliku grid tablice. Korišten je SilverGrid widget koji prikazuje mapu slova

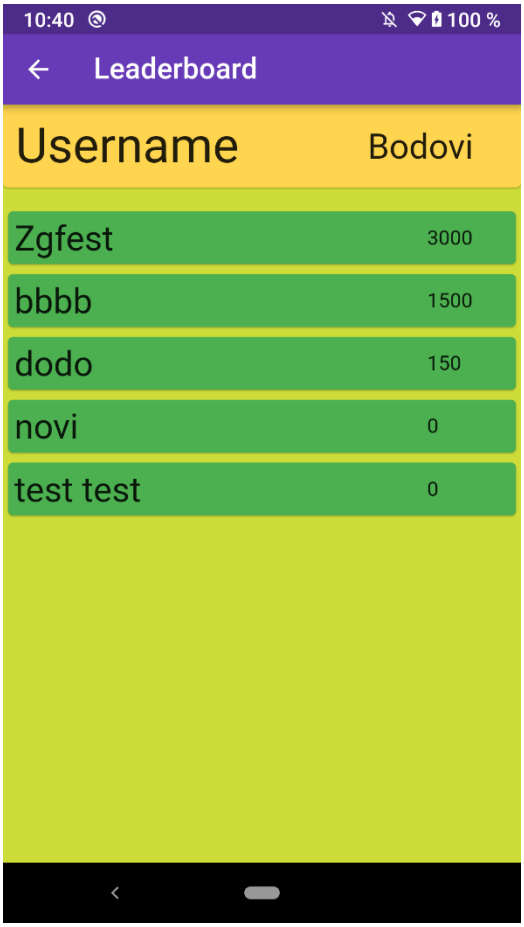


Slika 8 Isječak zaslona Rječnik Slova

Leaderboard

Leaderboard widget je ljestvica koja se prikazuje kao lista korisnika sortirano prema osvojenim bodovima pomoću upita na referencu baze podataka orderByChild().

Kako Firebase ne nudi sortiranje silazno, bilo je potrebno postaviti dodatnu varijablu koja je negativan broj osvojenih bodova *orderPoints*. To znači, ako korisnik u primjeru ima osvojenih 1500 bodova, *orderPoints* će bit -1500, gdje će biti prema ljestivici drugi po redu.



Slika 9 Isječak zaslona Leaderboard.

Settings

Widget postavke sadrži nekoliko funkcionalnosti: pregled podataka o korisniku, koje je spremljeno u bazu podataka, izmjena nadimka tj. username korisnika, odjava korisnika. Kod brisanja korisničkog računa i brisanja osvojenih bodova aplikacija će u AlertDialog prozoru pitati korisnika jel stvarno želi učiniti željenu radnju ili prekini zahtjev(cancel).

Kako bi kod brisanja korisničkog računa, bili sigurno obrisani podaci o korisniku u Firebase autentifikaciji i bazi podataka, a ne ovise jedno o drugome, javila se potreba za instalacijom Firebase proširenja Delete User Data.

Delete User Data radi na način da brisanjem korisničkog računa u firebase autentifikaciji se i obrišu njegovi podaci u bazi podataka preko njegovog UID-a. Da bi sve to radilo, potrebno je konfigurirati proširenje. Zahvaljući tom proširenju, omogućeno je jednostavno brisanje korisnika i njegovih podataka pozivom instance FirebaseAuth i funkcije deleteData().

```
final user = FirebaseAuth.instance;

TextButton(
  onPressed: () {
      user.currentUser!.delete();
      }

);
```
Primjer inicalizranja FirebaseAuth instance i njenog poziva u TextButton widgetu kao event handler

Zaključak

Ovaj završni rad se bavio učenjem brajice kroz igru u mobilnoj aplikaciji. Aplikacija se radila u Flutter aplikacijskom okruženju sa Dart programskim jezikom. Za pohranu podataka o korisniku se koristi baza podataka u oblaku Firebase servisa. Firebase isto tako vodi brigu o autentifikaciji korisnika. Na kraju se to zaokružuje u neku cijelinu gdje aplikacija ima svoju ulogu, bez previše bugova tj. neželjenih radnji u tijekom pokretanja koda. Testirane aplikacije nije provedeno, što zapravo daje smjernicu za njeni daljni plan razvoja.

Izvori:

[1] Stranica Flutter: https://flutter.dev/,

[2] Stranica Dart https://dart.dev/overview;

[3] Dart Sound null safety https://dart.dev/null-safety

[4] Brailleovo pismo. *Hrvatska enciklopedija, mrežno izdanje.* Leksikografski zavod Miroslav Krleža, 2021. Pristupljeno 25. 9. 2022.

https://www.enciklopedija.hr/natuknica.aspx?id=9210

[5] BRAILLEAPP: Educational Mobile Application to Assist in the Learning of Braille Language

https://www.researchgate.net/publication/279189920_BRAILLEAPP_Educational_Mobile_Application_to_Assist_in_the_Learning_of_Braille_Language

[6] Widget class

https://api.flutter.dev/flutter/widgets/Widget-class.html

[7] Flutter by Example

https://flutterbyexample.com/lesson/the-widget-tree

[8] StatelessWidget

https://api.flutter.dev/flutter/widgets/StatelessWidget-class.html

[9] StatefulWidget

https://api.flutter.dev/flutter/widgets/StatefulWidget-class.html

[10] InheritedWidget

https://api.flutter.dev/flutter/widgets/InheritedWidget-class.html

[11] Firebase dokumentacija

https://firebase.google.com/firebase/authentication

[12] Realtime Database

https://firebase.google.com/docs/database

[13] Firestore

https://firebase.google.com/docs/firestore

Popis slika:

Slika 1 - Primjer prikaza ćelije jednog Braille znaka:

Slika 2 – screenshot Braille aplikacije kod gameLetter widgeta

Slika 3 – prikaz popisa korisnika u Firebase Auth buildu

Slika 4 - prikaz Firebase BP

Slika 5 – use case dijagram aplikacije

Slika 6 – klasni dijagram aplikacije

Slika 7 – prikaz sign up zaslona

Slika 8 - Isječak zaslona Rječnik Slova

Slika 9 - Isječak zaslona Leaderboard.

Link projekta github:

https://github.com/Dhlupic-Stingray6/Braille_aplikacija.git