

Univerzitet u Novom Sadu   
Tehnički fakultet „Mihajlo Pupin“   
Zrenjanin

SEMINARSKI RAD

Predmet: Mobilne tehnologije i programiranje

**Evidencija zakazanih termina u mehaničarskoj radionici**

Predmetni nastavnik: Autor rada:

prof.dr Eleonora Brtka Stefan Zagorac SI 8/16

Predmetni asistent:

Prof.dr Igor Vecštejn

Zrenjanin 2022.

**Sadržaj:**

[**Predmetni cilj** 3](#_Toc121894696)

[**Uvod** 4](#_Toc121894697)

[**1.** **Korišćene tehnologije** 5](#_Toc121894698)

[1.1. Microsoft VS Code 5](#_Toc121894699)

[1.2. Android Studio 5](#_Toc121894700)

[**2.** **Baze podataka** 6](#_Toc121894701)

[**3.** **Jezičke tehnologije** 7](#_Toc121894702)

[3.1. Flutter 7](#_Toc121894703)

[3.2. Dart 7](#_Toc121894704)

[**4.** **UML dijagrami** 8](#_Toc121894705)

[**5.** **Aplikacija** 9](#_Toc121894706)

[5.1. Kreiranje UI login stranice 11](#_Toc121894707)

[5.2. Kreiranje UI stranice za registraciju 16](#_Toc121894708)

[5.3. Logika za login i registraciju 21](#_Toc121894709)

[5.4. UI i logika CRUD sistema za rad sa podacima Početne stranice 23](#_Toc121894710)

[5.5. UI i logika stranice za pretragu 34](#_Toc121894711)

[**6.** **Zaključak** 39](#_Toc121894712)

[**7.** **Literatura** 40](#_Toc121894713)

[**8.** **Izvori preuzetih ilustracija** 41](#_Toc121894714)

[Slika 1 41](#_Toc121894715)

[https://cdn.dribbble.com/users/528264/screenshots/3140440/firebase\_logo.png 41](#_Toc121894716)

## **Predmetni cilj**

Predmetni cilj predmeta Mobilne tehnologije i programiranje je sticanje osnovnih znanja i veština iz oblasti računarskog softvera i Interneta, kao suštinskih preduslova za dalje izučavanje informacionih i komunikacionih tehnologija i programiranja uopšte. Upoznavanje sa osnovama programiranja na mobilnim uređajima i osposobljavanje studenata da projektuju, kreiraju i održavaju aplikacije namenjene mobilnim platformama, sa posebnim akcentom na Android platformu [14]

Na predavanjima student se upoznaje sa osnovnim konceptima mobilnih aplikacija, istorijskim razvojem, tipovima mobilnih aplikacija, sigurnosti mobilnih aplikacija, korisničkim platformama i programerskim platformama. [14]

Na vežbama student savladava koncepte mobilnih aplikacija kroz tehnologiju flutter/dart i firebase. Kroz programerske primere student stiče znanja o korisničkoj i programerskoj platformi, osnovnim slojevima i servisima mobilnih aplikacija. [14]

## **Uvod**

Mobilna aplikacija je program koji je dizajniran da radi na mobilnom uređaju kao što je mobilni telefon, tablet ili wearable (pametni sat…). Te aplikacije imaju pristup senzorima i specifičnim funkcionalnostima uredjaja (kamera, vibracija i dr.). Razvijanje aplikacija za mobilne uređaje zahteva razmatranje karakteristika ovih uređaja i njihovih ograničenja (rade na bateriji i imaju slabije procesore i manje RAM memorije nego desktop računari). Programeri takođe moraju imati u vidu širok spektar: dimenzija ekrana, hardverske specifikacije i konfiguracije različitih uredjaja. Na većini mobilnih uredjaja se nalazi jedan od sledeća dva operativna sistema: Android i IOS.

Mobilne aplikacije se ponekad kategorizuju u zavisnosti od toga koji je pristup korišćen pri izradi aplikacije, pa stoga možemo razvoj mobilnih aplikacija podeliti na sledeće grupe:

* Izvorne (native) mobilne aplikacije:

iOS app development

Android app development

* Kvazi-izvorne mobilne aplikacije:

React Native

NativeScript

Xamarin

* Hybridne mobilne aplikacije

Ionic + Android.js

Quasar + Vue.js

Framework7 + React.js

Responsivne web aplikacije

## **Korišćene tehnologije**

U ovom seminarskom radu korišćene su tehnologije za razvoj mobilnih aplikacija i to su: Microsoft VS Code, Android Studio, Flutter i Dart ekstenzije za VS Code.

### 1.1. Microsoft VS Code

Microsoft VS Code je editor izvornog koda koji se može koristiti za razne programske jezike, kao što su Java, Javaskript, Gou, Node.js i C++.

Umeesto sistema projekata, VS Code omogućava korisniku da otvori jedan ili više direktorijuma, što zatim može biti sačuvano kao radni prostor za ponovnu upotrebu. Ovo mu omogućava da radi kao jezičko-agnostički uređivač koda za bilo koji jezik. Podržava brojne programske jezike i skup karakteristika koje se razlikuju od jezika do jezika. Neželjene datoteke i direktorijumi se mogu isključiti iz stabla projekta preko podešavanja. Mnoge karakteristike ovog uređivača nisu izložene u menijima ili u korisničkom interfejsu, već im se može pristupiti preko komandne palete.

### 1.2. Android Studio

Android Studio je razvojno okruženje zasnovano na IntelliJ IDEA softveru. Namenjen prvenstveno za razvoj Android mobilnih aplikacija, Android Studio je optimalno okruženje koje pruža sve mogućnosti trenutno dostupne u ovoj grani programiranja. Android aplikacije se mogu pisati i u drugim razvojnim okruženjima (na primer Eclipse IDE ili Visual Studio), ali sam razvoj je znatno otežan, jer druga okruženja nisu namenjena isključivo za Android i zahtevaju dodatna podešavanja u vidu dodataka specifičnih za Android projekte. Budući da je Android Studio zasnovan na IntelliJ editoru koda i razvojnim alatima istog, sasvim je opravdano reći da je okruženje koje omogućava kompletno iskustvo u razvoju Android projekata. Korisnički interfejs je kreiran tako da su najkorišćenije funkcionalnosti uvek dostupne na vidljivim mestima, a takođe i pruža mogućnost reorganizacije paleta alata onako kako korisniku odgovara. Pored ove dve karakteristike, Android Studio pruža i više nego dovoljno opcija za neometani razvoj Android aplikacija, uključujući podešavanja raličitih perspektiva za prikaz strukture projekta, integraciju sa nekoliko raličitih alata za kontrolu koda (engl. version control/source control) i mnoge druge.

## **Baze podataka**

Za ovaj seminarski rad korišćena je Firebase online baza podataka.

Firebase pruža usluge baze podataka za rad u realnom vremenu i bekend usluge. Servis programerima aplikacija pruža API koji omogućava sinhronizovanje aplikacionih podataka preko klijenata i njihovo skladištenje na oblak (eng. cloud).Kompanija pruža klijentske biblioteke koje omogućavaju integraciju sa Android, iOS, JavaScript, Java, Objective-C i Node.js aplikacijama. Bazi podataka se takođe može pristupiti preko REST API i veza za nekoliko Javaskript biblioteka kao što su AngularJS, React, VueJS, Ember.js i Backbone.js. REST API koristi serverski protokol za slanje događaja, što je u stvari API za kreiranje HTTP konekcija koje se koriste za primanje notifikacija sa servera. Programeri, korišćenjem baze podataka za rad u realnom vremenu, mogu da osiguraju svoje podatke korišćenjem propisanih bezbednosnih pravila koje pruža kompanijski server.



*Slika 1. Firebase logo.*

## **Jezičke tehnologije**

Za izadu ovog seminarskog rada korišćeni su programski jezik Dart i framework Flutter.

### Flutter

Flutter je jedan od modernijih višeplatformskih mobilnih razvojnih okruženja, nudi mnogo opcija koje omogućavaju razvoj aplikacija za Android i iOS na veoma brz i efikasan način.

Prednost Flutter-a jeste u tome što se kroz njega kreira aplikacija prilagođena svim platformama, na različitim uređajima, što značajno skraćuje vreme razvijanja dok se istovremeno smanjuje i trošak za klijenta. Flutter se ne razlikuje previše od razvoja aplikacija u nekom od drugih toolkit-ova. Pored toga što je jednostavniji, rad u njemu je i brži, ima mnogo plugin-ova koji rešavaju specifične probleme i brzo renderuje 2D objekte u aplikacijama. Samim tim, rast njegove popularnosti je trend koji će trajati i u godinama koje dolaze. Ne treba zaboraviti da iza ovog toolkit-a stoji Google koji daje sve od sebe da upravo ovaj framework postane alat koji će koristiti veliki broj developera.

### Dart

Dart je programski jezik opšte namene koji je originalno razvio Google, a kasnije prihvaćen kao standard od strane ECMA(ECMA-480). Koristi se za izradu veb, server aplikacija, aplikacija za mobilni telefon, kao i za Internet Stvari uređaje. Predstavlja softver otvorenog koda pod BSD licencom.

Dart se izvršava na četiri glavna načina, a to su: preveden kao JavaSkript, u Dartijum pretraživaču, samostalno, unapred komajliran (AOT - ahead of time).

## **UML dijagrami**

Unified Modeling Language (UML) je standardni jezik za izradu specifikacija,vizualizaciju, kreiranje i dokumentovanje delova softverskog sistema. Kreiran je od strane Object Management Group-e (OMG). Prva specifikacija UML 1.0 je izašla u januaru 1997. godine. On se razlikuje od drugih standardnih programskih jezika poput C++, Java, itd, jer predstavlja vizuelni jezik. Uglavnom UML se koristi za modelovanje softverskih sistema,ali on se može koristiti i za modelovanje drugih sistema.

UML nije tipičan programski jezik ali se može koristiti da se pomoću dijagrama generiše kod u različitim programskim jezicima. On je direktno povezan sa objektno orjentisanom analizom i dizajnom i dovoljno moćan da prikaže bilo koji koncept koji u njima postoji.

Jedni od najznačajnijih UML dijagrama su :

* Dijagram slučajeva korišćenja
* Dijagram aktivnosti
* Dijagram stanja
* Dijagram sekvenci
* Dijagram klasa.

U ovom radu su korišćeni: Dijagram slučajeva korišćenja (Use case dijagram), Dijagram Aktivnost i Dijagram sekvenci

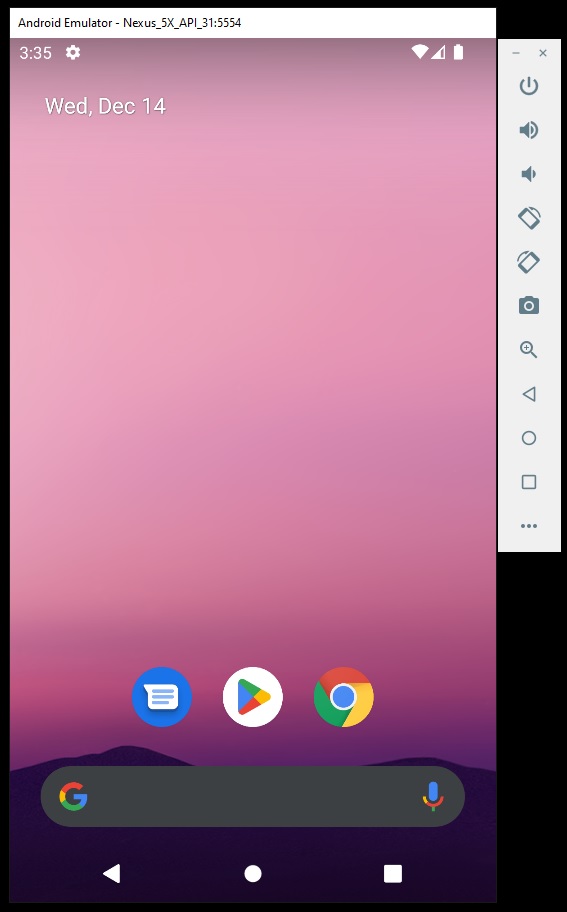
## **Aplikacija**

Kod izrade aplikacije najpre treba kreirati viruelni Android uređaj koristeći Android Studio iz tog razloga da bi imali bolji uvid u to šta se dešava svakim sledećim korakom koji napravimo.

Putanja za kreiranje virtuelnog android uredjaja:

***Android Studio > Virtual Device Manager > Create Device > Nexus 5X > R API 31***

Nakon preuzimanja i pokretanja virtuelnog uređaja dobijamo ovakav UI:



*Slika 2. UI virtuelnog android uređaja.*

Nakon toga potrebno je kreirati flutter projekat u Command promptu (CMD), tako što treba odabrati željeni folder komandom “cd” (choose directory), I uneti sledeću komandu:

***flutter create firstapp*** – firstapp je naziv Aplikacije(projekta) i može biti proizvoljan.

Naravno, podrazumeva se da je flutter instaliran globalno na računar.

Zatim u Microsoft VS Code-u otvara se folder gde je aplikacija smeštena:

***File > Open Folder > putanja do aplikacije***

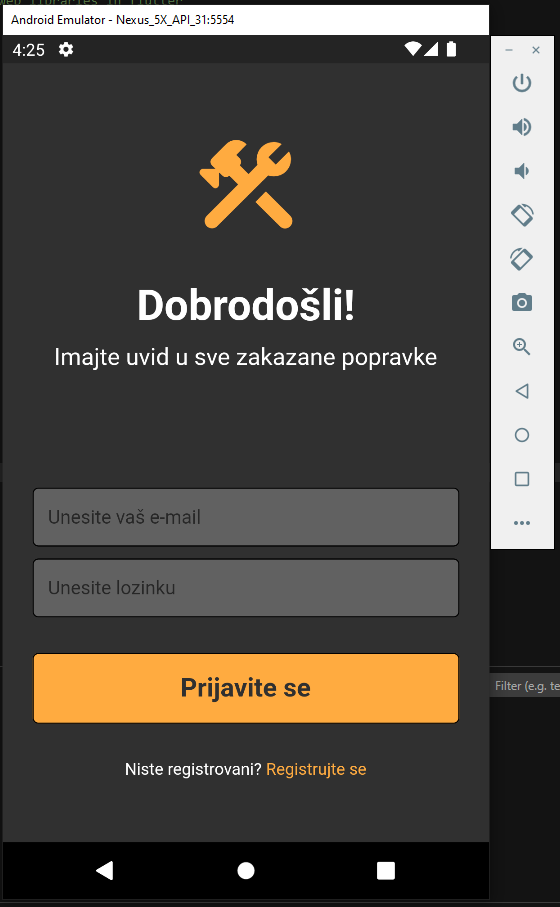
Da bi aplikacija bila funkcionalna sa backend strane, nužno neophodno je kreiranje firebase bazu podataka putem zvaničnog sajta firebase-a i povezivanje iste sa projektom (preteći jednostavne instrukcije sa sajta).

Nakon svega ovoga, potrebno je još i odabrati uređaj na kome će da se pokreće aplikacija koja se kreira, jednostavnim klikom na dugme “**devices**” u donjem desnom uglu Microsoft VS Code-a. Naravno bira se uređaj koji je prethodno kreiran u Android Studiu, tj. **Nexus 5X API 31.**

### Kreiranje UI login stranice

Kada je novi flutter projekat kreiran, automatski je kreirana krajnje jednostavna aplikacija koja broji klikove koji su aplicirani na **FloatingActionButton** klassu, za ovaj projekat je to suvišno tako da treba najpre obrisati sve u main.dart fajlu osim osnovne inicijalizacije i **build widget**-a koji vraća novi dart fajl u kome će se nalaziti provera da li je korisnik ulogovan ili ne, i na osnovu toga voditi ga na odgovarajuću stranicu (login ili register).

Nakon toga kreira se login stranica pod nazivom “**login\_stranica.dart**”. Zatim importuju se potrebne biblioteke i kreira se **StatefullWigdet** koji vraća **Scaffold** classu u kojoj se kreira UI login stranice. Korišćenjem klasa: **SafeArea, Conatainer, Column, Padding, Text, Icon**… i njihovih property-a kao što su: **child, color, size, padding**… Dolazimo do željenog UI-a login stranice.



*Slika 3. UI login stranice.*

Listing koda stranice: “**login\_stranica.dart**”:

// ignore\_for\_file: prefer\_const\_constructors, prefer\_const\_literals\_to\_create\_immutables, annotate\_overrides, unused\_import, depend\_on\_referenced\_packages

import 'package:firebase\_auth/firebase\_auth.dart';

import 'package:firebase\_core/firebase\_core.dart';

import 'package:firstapp/stranice/pocetna\_stranica.dart';

import 'package:flutter/material.dart';

class LoginStranica extends StatefulWidget {

  final VoidCallback prikaziStranicuZaRegistraciju;

  const LoginStranica({Key? key, required this.prikaziStranicuZaRegistraciju})

      : super(key: key);

  @override

  State<LoginStranica> createState() => \_LoginStranicaState();

}

class \_LoginStranicaState extends State<LoginStranica> {

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      backgroundColor: Colors.grey[850],

      body: SafeArea(

        child: Center(

          child: SingleChildScrollView(

            child:

                Column(mainAxisAlignment: MainAxisAlignment.center, children: [

              Icon(

                Icons.construction\_rounded,

                color: Colors.orangeAccent,

                size: 100,

              ),

              SizedBox(height: 30),

              Text(

                'Dobrodošli!',

                style: TextStyle(

                  fontWeight: FontWeight.bold,

                  fontSize: 36,

                  color: Colors.white,

                ),

              ),

              SizedBox(height: 10),

              Text(

                'Imajte uvid u sve zakazane popravke',

                style: TextStyle(

                  fontSize: 20,

                  color: Colors.white,

                ),

              ),

              SizedBox(height: 100),

              Padding(

                padding: const EdgeInsets.symmetric(horizontal: 25.0),

                child: Container(

                  decoration: BoxDecoration(

                    color: Colors.grey[700],

                    border: Border.all(color: Colors.black),

                    borderRadius: BorderRadius.circular(5),

                  ),

                  child: Padding(

                    padding: const EdgeInsets.only(left: 12.0),

                    child: TextField(

                      decoration: InputDecoration(

                          border: InputBorder.none,

                          hintText: 'Unesite vaš e-mail'),

                      cursorColor: Colors.white,

                      cursorHeight: 20,

                    ),

                  ),

                ),

              ),

              SizedBox(height: 10),

              Padding(

                padding: const EdgeInsets.symmetric(horizontal: 25.0),

                child: Container(

                  decoration: BoxDecoration(

                    color: Colors.grey[700],

                    border: Border.all(color: Colors.black),

                    borderRadius: BorderRadius.circular(5),

                  ),

                  child: Padding(

                    padding: const EdgeInsets.only(left: 12.0),

                    child: TextField(

                      obscureText: true,

                      decoration: InputDecoration(

                          border: InputBorder.none,

                          hintText: 'Unesite lozinku'),

                      cursorColor: Colors.white,

                      cursorHeight: 20,

                    ),

                  ),

                ),

              ),

              SizedBox(height: 30),

              Padding(

                padding: const EdgeInsets.symmetric(horizontal: 25.0),

                child: GestureDetector(

                  child: Container(

                    padding: EdgeInsets.all(16),

                    decoration: BoxDecoration(

                        color: Colors.orangeAccent,

                        border: Border.all(color: Colors.black),

                        borderRadius: BorderRadius.circular(5)),

                    child: Center(

                        child: Text(

                      'Prijavite se',

                      style: TextStyle(

                          color: Colors.grey[850],

                          fontSize: 22,

                          fontWeight: FontWeight.w700),

                    )),

                  ),

                ),

              ),

              SizedBox(height: 30),

              Row(

                mainAxisAlignment: MainAxisAlignment.center,

                children: [

                  Text(

                    'Niste registrovani?',

                    style: TextStyle(

                      color: Colors.white,

                    ),

                  ),

                  GestureDetector(

                    child: Text(

                      ' Registrujte se',

                      style: TextStyle(

                        color: Colors.orangeAccent,

                      ),

                    ),

                  ),

                ],

              ),

            ]),

          ),

        ),

      ),

    );

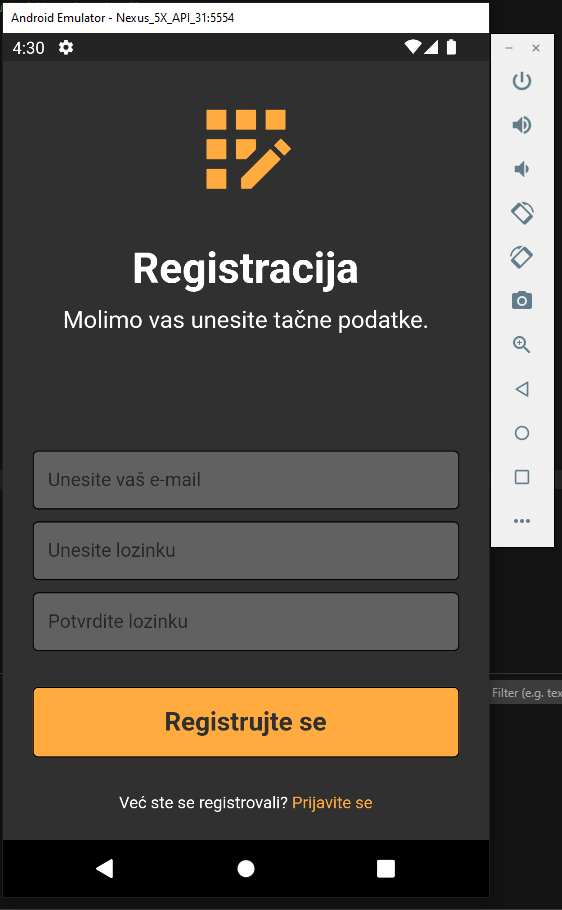
  }

}

*Listing 1. “login\_stranica.dart”.*

### Kreiranje UI stranice za registraciju

Najpre se kreira nova stranica pod nazivom “**registracija\_stranica.dart**” importuju potrebne biblioteke i stranice i kreira se **StatefulWidget** koji vraća Scaffold klasu u kojoj pravimo UI sličan (u temi) login stranice, s’tim da nam je potrebno tri polja za unos (**TextField** klasa), drugačija ikonica (**Icon** klasa), i drugačiji određeni tekstovi (**Text** klasa).



*Slika 4. UI stranice za registraciju.*

Listing koda stranice: “registracija\_stranica.dart”:

// ignore\_for\_file: prefer\_const\_constructors, unused\_import, depend\_on\_referenced\_packages

import 'package:firebase\_auth/firebase\_auth.dart';

import 'package:firebase\_core/firebase\_core.dart';

import 'package:firstapp/stranice/pocetna\_stranica.dart';

import 'package:flutter/material.dart';

class RegistracijaStranica extends StatefulWidget {

  final VoidCallback prikaziLoginStranicu;

  const RegistracijaStranica({

    Key? key,

    required this.prikaziLoginStranicu,

  }) : super(key: key);

  @override

  State<RegistracijaStranica> createState() => \_RegistracijaStranicaState();

}

class \_RegistracijaStranicaState extends State<RegistracijaStranica> {

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      backgroundColor: Colors.grey[850],

      body: SafeArea(

        child: Center(

          child: SingleChildScrollView(

            child:

                Column(mainAxisAlignment: MainAxisAlignment.center, children: [

              Icon(

                Icons.app\_registration\_sharp,

                color: Colors.orangeAccent,

                size: 100,

              ),

              SizedBox(height: 30),

              Text(

                'Registracija',

                style: TextStyle(

                  fontWeight: FontWeight.bold,

                  fontSize: 36,

                  color: Colors.white,

                ),

              ),

              SizedBox(height: 10),

              Text(

                'Molimo vas unesite tačne podatke.',

                style: TextStyle(

                  fontSize: 20,

                  color: Colors.white,

                ),

              ),

              SizedBox(height: 100),

              Padding(

                padding: const EdgeInsets.symmetric(horizontal: 25.0),

                child: Container(

                  decoration: BoxDecoration(

                    color: Colors.grey[700],

                    border: Border.all(color: Colors.black),

                    borderRadius: BorderRadius.circular(5),

                  ),

                  child: Padding(

                    padding: const EdgeInsets.only(left: 12.0),

                    child: TextField(

                      decoration: InputDecoration(

                          border: InputBorder.none,

                          hintText: 'Unesite vaš e-mail'),

                      cursorColor: Colors.white,

                      cursorHeight: 20,

                    ),

                  ),

                ),

              ),

              SizedBox(height: 10),

              Padding(

                padding: const EdgeInsets.symmetric(horizontal: 25.0),

                child: Container(

                  decoration: BoxDecoration(

                    color: Colors.grey[700],

                    border: Border.all(color: Colors.black),

                    borderRadius: BorderRadius.circular(5),

                  ),

                  child: Padding(

                    padding: const EdgeInsets.only(left: 12.0),

                    child: TextField(

                      obscureText: true,

                      decoration: InputDecoration(

                          border: InputBorder.none,

                          hintText: 'Unesite lozinku'),

                      cursorColor: Colors.white,

                      cursorHeight: 20,

                    ),

                  ),

                ),

              ),

              SizedBox(height: 10),

              Padding(

                padding: const EdgeInsets.symmetric(horizontal: 25.0),

                child: Container(

                  decoration: BoxDecoration(

                    color: Colors.grey[700],

                    border: Border.all(color: Colors.black),

                    borderRadius: BorderRadius.circular(5),

                  ),

                  child: Padding(

                    padding: const EdgeInsets.only(left: 12.0),

                    child: TextField(

                      obscureText: true,

                      decoration: InputDecoration(

                          border: InputBorder.none,

                          hintText: 'Potvrdite lozinku'),

                      cursorColor: Colors.white,

                      cursorHeight: 20,

                    ),

                  ),

                ),

              ),

              SizedBox(height: 30),

              Padding(

                padding: const EdgeInsets.symmetric(horizontal: 25.0),

                child: GestureDetector(

                  child: Container(

                    padding: EdgeInsets.all(16),

                    decoration: BoxDecoration(

                        color: Colors.orangeAccent,

                        border: Border.all(color: Colors.black),

                        borderRadius: BorderRadius.circular(5)),

                    child: Center(

                        child: Text(

                      'Registrujte se',

                      style: TextStyle(

                          color: Colors.grey[850],

                          fontSize: 22,

                          fontWeight: FontWeight.w700),

                    )),

                  ),

                ),

              ),

              SizedBox(height: 30),

              Row(

                mainAxisAlignment: MainAxisAlignment.center,

                children: [

                  Text(

                    'Već ste se registrovali?',

                    style: TextStyle(

                      color: Colors.white,

                    ),

                  ),

                  GestureDetector(

                    onTap: widget.prikaziLoginStranicu,

                    child: Text(

                      ' Prijavite se',

                      style: TextStyle(

                        color: Colors.orangeAccent,

                      ),

                    ),

                  ),

                ],

              ),

            ]),

          ),

        ),

      ),

    );

  }

}

*Listing 2. “registracija\_stranica.dart”.*

### Logika za login i registraciju

#### **Login stranica**

Da bi aplikacija bila funkcionalna tj. imala backend najpre je potrebno definisati dva text kontrolera na login stranici i funkciju koja se izvršava klikom na dugme “Prijavite se”.

 final \_emailKontroler = TextEditingController();

  final \_lozinkaKontroler = TextEditingController();

  Future prijava() async {

    await FirebaseAuth.instance.signInWithEmailAndPassword(

      email: \_emailKontroler.text.trim(),

      password: \_lozinkaKontroler.text.trim(),

    );

  }

  @override

  void dispose() {

    \_emailKontroler.dispose();

    \_lozinkaKontroler.dispose();

    super.dispose();

  }

*Listing 3. logika za login na “login\_stranica.dart”*

Zatim poljima za unos (**TextField** klasama) dodati property “**controller**” i vrednost property-u odgovarajući kontroler koji je prethodno definisan, kao i dugmetu koje služi za prijavu, tj. GestureDetector widgetu koji je obavijen oko dugmeta dodati property: „**onTap**“ i vrednos funkciju “**prijava()**” koja je takođe prethodno definisana,

Ovaj najčin login-a tj. prijave je moguć zbog korišćenja pre definisanih klasa i metoda koji su nam dostupni zbog korišćenja firebase baze podataka.

#### **Registraciona stranica**

Što se tiče stranice za registraciju logika za ovu stranicu je za nijansu komplikovanija jer imamo tri polja za unos (**TextField** klase), pa samim tim moramo imati tri kontrolera za unos i još jednu funkciju koja ima zadatak da proveri da li su unete lozinke iste, na osnovu povratne bool vrednosti poziva se funkcija koja kreira korisnički nalog.

final \_emailKontroler = TextEditingController();

  final \_lozinkaKontroler = TextEditingController();

  final \_lozinkaKontroler2 = TextEditingController();

  @override

  void dispose() {

    \_emailKontroler.dispose();

    \_lozinkaKontroler.dispose();

    \_lozinkaKontroler2.dispose();

    super.dispose();

  }

  Future registrujSe() async {

    if (potvrdjivacLozinke()) {

      await FirebaseAuth.instance.createUserWithEmailAndPassword(

          email: \_emailKontroler.text.trim(),

          password: \_lozinkaKontroler.text.trim());

    }

  }

  bool potvrdjivacLozinke() {

    if (\_lozinkaKontroler.text.trim() == \_lozinkaKontroler2.text.trim()) {

      return true;

    } else {

      return false;

    }

  }

*Listing 4. logika za registraciju na “registracija\_stranica.dart”.*

Kada je ovo završeno treba kreirati dve nove stranice: “**glavna\_stranica.dart**” i “**autentifikaciona\_stranica.dart**”.

Ove stranice služe da se utvrdi da li je korisnik ulogovan ili ne, kada se pokreće aplikacija i na osnovu toga se prikazuje određena stranica prilikom pokretanja aplikacije.

### UI i logika CRUD sistema za rad sa podacima Početne stranice

Najpre se kreira nova stranica “**pocetna\_stranica.dart**”, iz razloga lakše i bolje organizacije pravimo dva foldera “autentifikacija” i “stranice” pa na osnovu toga čemu .dart fajlovi služe vršimo razmeštaj.

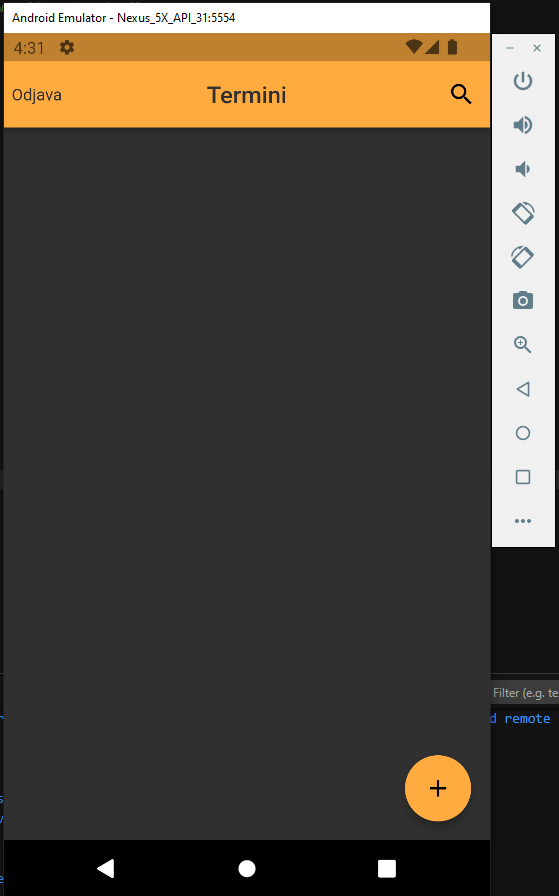
Nakon toga kreiramo **StatefulWidget** koji vraća **build Widget** koji vraća **Scaffold** klasu u kojoj kreiramo UI početne stranice koristeći klase: **AppBar, IconButton, Icon, GestureDetector, StreamBuilder, Card**…

Klasu **AppBar** koristimo jer je odgovarajuća za implementaciju ActionText-a koji služi za odjavu i dugmeta “search” koje će nas kasnije odvesti na stranicu za pretragu.

Klasu **StreamBuilder** koristimo iz razloga da bi se promene nad bazom dešavale u runtime-u.

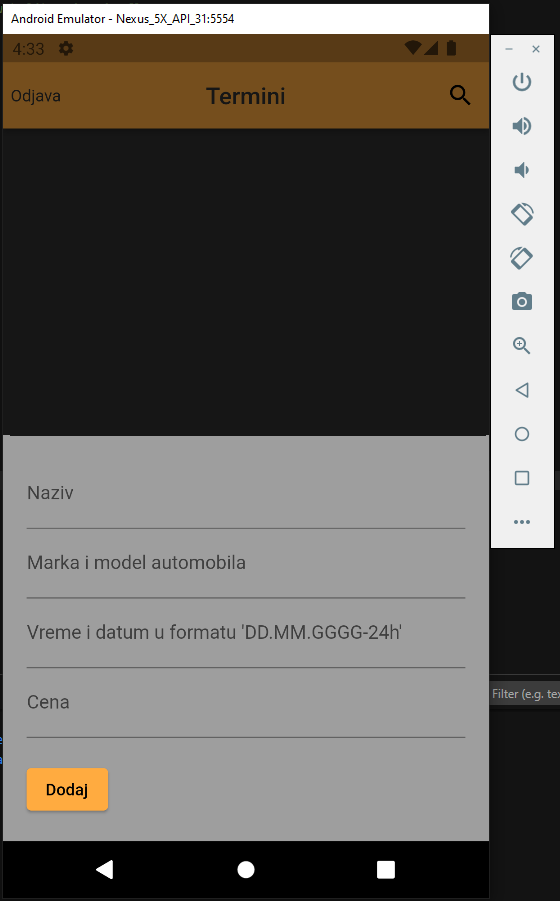
Unutar **StreamBuilder**-a imamo klasu **Card** u kojoj smeštamo podatke koji se nalaze u bazi, jedna klasa **Card** predstavlja jedan dokument u bazi i uz to sadrži dva IconButton-a koji pozivaju funkcije za ažuriranje podakata i brisanje dokumenta.

Srž ove stranice su tri funkcije (\_create, \_update, \_delete) koje kreiraju formu za unos (osim funkcije delete) i komunicaraju sa bazom na osnovu onoga što im je zadato da izvrše.



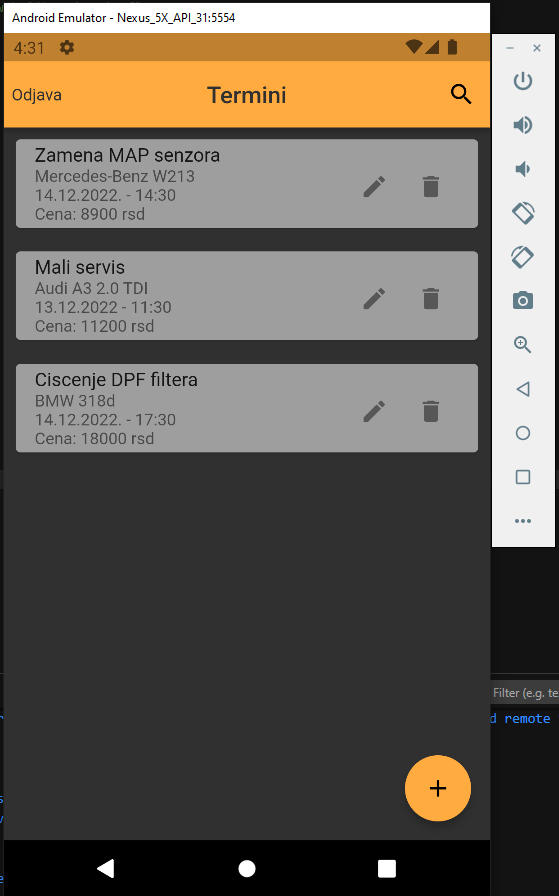
*Slika 5. UI početne stranice i praznom bazom*

Klikom na FloatingActionButton pozivamo funkciju **\_create** i dobijamo formu za unos:



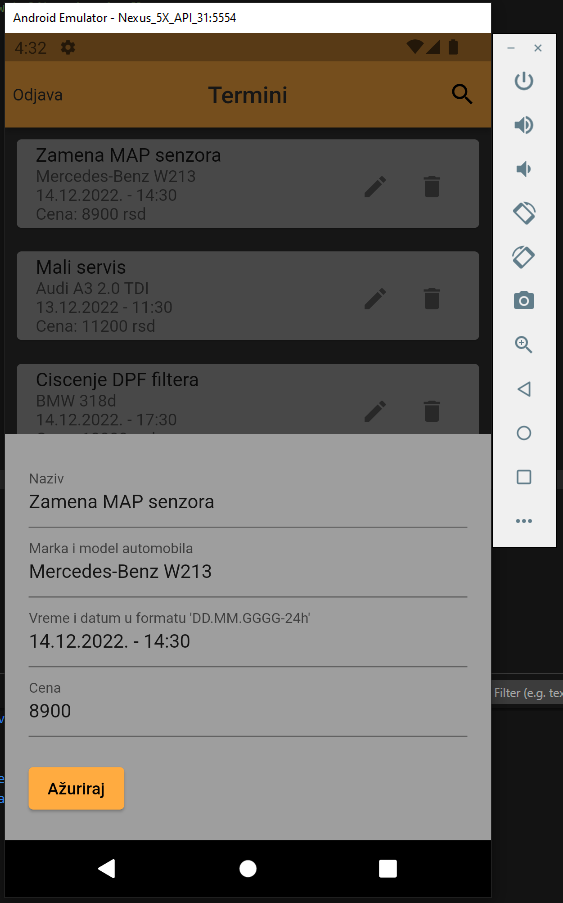
*Slika 6. Forma za kreiranje dokumenta.*

Nakon unosa podataka i klika na dugme “dodaj”, dobijamo zakazani termin prikazan na stranici kao i dva **IconButton**-a za ažuriranje i brisanje određenog termina.



*Slika 7. Izgled početne stranice nakon unosa termina.*

Klikom na dugme za ažurianje (**IconButton** sa ikonicom olovke) pozivamo funkciju **\_update** i dobijamo istu formu kao i kod **\_create** funkcije sa tim što ovde menjamo podatke i kontoroleri su podešeni na prethodnu vrednost.



*Slika 8. Izgled forme za ažuriranje termina.*

#### **Sortiranje**

Radi lakšeg pregleda pravimo **Query** gde instanci na kolekciju baze dodajemo metodu **.orderBy** sa kojom sortiramo termine po ceni on najniže do najviše.

// ignore\_for\_file: prefer\_interpolation\_to\_compose\_strings, prefer\_const\_constructors, unused\_import, depend\_on\_referenced\_packages, use\_build\_context\_synchronously, unused\_element, sort\_child\_properties\_last, avoid\_web\_libraries\_in\_flutter

import 'package:cloud\_firestore/cloud\_firestore.dart';

import 'package:firebase\_auth/firebase\_auth.dart';

import 'package:flutter/material.dart';

import 'package:firebase\_core/firebase\_core.dart';

import 'pretraga\_stranica.dart';

class PocetnaStranica extends StatefulWidget {

  const PocetnaStranica({super.key});

  @override

  State<PocetnaStranica> createState() => \_PocetnaStranicaState();

}

class \_PocetnaStranicaState extends State<PocetnaStranica> {

  final korisnik = FirebaseAuth.instance.currentUser!;

  final TextEditingController \_nameController = TextEditingController();

  final TextEditingController \_priceController = TextEditingController();

  final TextEditingController \_makeController = TextEditingController();

  final TextEditingController \_dateController = TextEditingController();

  final CollectionReference \_products =

      FirebaseFirestore.instance.collection('products');

  final Query \_productss =

      FirebaseFirestore.instance.collection("products").orderBy('price');

  Future<void> \_create([DocumentSnapshot? documentSnapshot]) async {

    \_nameController.text = '';

    \_priceController.text = '';

    \_makeController.text = '';

    \_dateController.text = '';

    await showModalBottomSheet(

        isScrollControlled: true,

        context: context,

        builder: (BuildContext ctx) {

          return Container(

            color: Colors.grey[500],

            child: Padding(

              padding: EdgeInsets.only(

                  top: 20,

                  left: 20,

                  right: 20,

                  bottom: MediaQuery.of(ctx).viewInsets.bottom + 20),

              child: Column(

                mainAxisSize: MainAxisSize.min,

                crossAxisAlignment: CrossAxisAlignment.start,

                children: [

                  TextField(

                    controller: \_nameController,

                    decoration: const InputDecoration(

                      labelText: 'Naziv',

                    ),

                  ),

                  TextField(

                    controller: \_makeController,

                    decoration: const InputDecoration(

                      labelText: 'Marka i model automobila',

                    ),

                  ),

                  TextField(

                    controller: \_dateController,

                    decoration: const InputDecoration(

                      labelText: "Vreme i datum u formatu 'DD.MM.GGGG-24h'",

                    ),

                  ),

                  TextField(

                    keyboardType:

                        const TextInputType.numberWithOptions(decimal: true),

                    controller: \_priceController,

                    decoration: const InputDecoration(

                      labelText: 'Cena',

                    ),

                  ),

                  const SizedBox(

                    height: 20,

                  ),

                  ElevatedButton(

                    style: ElevatedButton.styleFrom(

                        backgroundColor: Colors.orangeAccent),

                    child: const Text(

                      'Dodaj',

                      style: TextStyle(color: Colors.black),

                    ),

                    onPressed: () async {

                      final String name = \_nameController.text;

                      final int? price = int.tryParse(\_priceController.text);

                      final String make = \_makeController.text;

                      List searchList = [];

                      for (var i = 1; i <= make.length; i++) {

                        var nextEl = make.substring(0, i).toLowerCase();

                        searchList.add(nextEl);

                        nextEl = make.substring(0, i);

                        searchList.add(nextEl);

                      }

                      final String date = \_dateController.text;

                      if (price != null) {

                        await \_products.add({

                          "name": name,

                          "price": price,

                          "make": make,

                          "date": date,

                          'makeArr': searchList,

                        });

                        \_nameController.text = '';

                        \_priceController.text = '';

                        \_makeController.text = '';

                        \_dateController.text = '';

                        Navigator.of(context).pop();

                      }

                    },

                  )

                ],

              ),

            ),

          );

        });

  }

  Future<void> \_update([DocumentSnapshot? documentSnapshot]) async {

    if (documentSnapshot != null) {

      \_nameController.text = documentSnapshot['name'];

      \_priceController.text = documentSnapshot['price'].toString();

      \_makeController.text = documentSnapshot['make'];

      \_dateController.text = documentSnapshot['date'];

    }

    await showModalBottomSheet(

        isScrollControlled: true,

        context: context,

        builder: (BuildContext ctx) {

          return Container(

            color: Colors.grey[500],

            child: Padding(

              padding: EdgeInsets.only(

                  top: 20,

                  left: 20,

                  right: 20,

                  bottom: MediaQuery.of(ctx).viewInsets.bottom + 20),

              child: Column(

                mainAxisSize: MainAxisSize.min,

                crossAxisAlignment: CrossAxisAlignment.start,

                children: [

                  TextField(

                    controller: \_nameController,

                    decoration: const InputDecoration(labelText: 'Naziv'),

                  ),

                  TextField(

                    controller: \_makeController,

                    decoration: const InputDecoration(

                        labelText: 'Marka i model automobila'),

                  ),

                  TextField(

                    controller: \_dateController,

                    decoration: const InputDecoration(

                        labelText: "Vreme i datum u formatu 'DD.MM.GGGG-24h'"),

                  ),

                  TextField(

                    keyboardType:

                        const TextInputType.numberWithOptions(decimal: true),

                    controller: \_priceController,

                    decoration: const InputDecoration(

                      labelText: 'Cena',

                    ),

                  ),

                  const SizedBox(

                    height: 20,

                  ),

                  ElevatedButton(

                    style: ElevatedButton.styleFrom(

                        backgroundColor: Colors.orangeAccent),

                    child: const Text(

                      'Ažuriraj',

                      style: TextStyle(color: Colors.black),

                    ),

                    onPressed: () async {

                      final String name = \_nameController.text;

                      final int? price = int.tryParse(\_priceController.text);

                      final String make = \_makeController.text;

                      final String date = \_dateController.text;

                      List searchList = [];

                      for (var i = 1; i <= make.length; i++) {

                        var nextEl = make.substring(0, i).toLowerCase();

                        searchList.add(nextEl);

                        nextEl = make.substring(0, i);

                        searchList.add(nextEl);

                      }

                      if (price != null) {

                        await \_products.doc(documentSnapshot!.id).update({

                          "name": name,

                          "price": price,

                          "make": make,

                          "date": date,

                          "makeArr": searchList,

                        });

                        \_nameController.text = '';

                        \_priceController.text = '';

                        \_makeController.text = '';

                        \_dateController.text = '';

                        Navigator.of(context).pop();

                      }

                    },

                  )

                ],

              ),

            ),

          );

        });

  }

  Future<void> \_delete(String productId) async {

    await \_products.doc(productId).delete();

    ScaffoldMessenger.of(context).showSnackBar(

        const SnackBar(content: Text('Uspešno ste obrisali ovaj termin')));

  }

  @override

  Widget build(BuildContext context) {

    return Scaffold(

        appBar: AppBar(

          centerTitle: true,

          backgroundColor: Colors.orangeAccent,

          foregroundColor: Colors.grey[850],

          actions: <Widget>[

            IconButton(

              icon: Icon(

                Icons.search,

                color: Colors.black,

              ),

              onPressed: () {

                Navigator.push(

                  context,

                  MaterialPageRoute(

                      builder: (context) => const PretraziStranica()),

                );

              },

            )

          ],

          title: Text('Termini'),

          leading: GestureDetector(

            onTap: (() {

              FirebaseAuth.instance.signOut();

            }),

            child: Column(

              mainAxisAlignment: MainAxisAlignment.center,

              children: [

                Text(

                  'Odjava',

                  style: TextStyle(

                    color: Colors.grey[850],

                  ),

                ),

              ],

            ),

          ),

        ),

        body: Container(

          color: Colors.grey[850],

          child: StreamBuilder(

            stream: \_productss.snapshots(),

            builder: (context, AsyncSnapshot<QuerySnapshot> streamSnapshot) {

              if (streamSnapshot.hasData) {

                return ListView.builder(

                  itemCount: streamSnapshot.data!.docs.length,

                  itemBuilder: (context, index) {

                    final DocumentSnapshot documentSnapshot =

                        streamSnapshot.data!.docs[index];

                    return Card(

                      color: Colors.grey[500],

                      margin: const EdgeInsets.all(10),

                      child: ListTile(

                        title: Text(documentSnapshot['name']),

                        subtitle: Text(documentSnapshot['make'] +

                            '\n' +

                            documentSnapshot['date'] +

                            '\n' +

                            'Cena: ' +

                            documentSnapshot['price'].toString() +

                            ' rsd'),

                        trailing: SizedBox(

                          width: 96,

                          child: Row(

                            children: [

                              IconButton(

                                  icon: const Icon(Icons.edit),

                                  onPressed: () => \_update(documentSnapshot)),

                              IconButton(

                                  icon: const Icon(Icons.delete),

                                  onPressed: () =>

                                      \_delete(documentSnapshot.id)),

                            ],

                          ),

                        ),

                      ),

                    );

                  },

                );

              }

              return const Center(

                child: CircularProgressIndicator(),

              );

            },

          ),

        ),

        floatingActionButton: FloatingActionButton(

          onPressed: () => \_create(),

          child: const Icon(

            Icons.add,

            color: Colors.black,

          ),

          backgroundColor: Colors.orangeAccent,

        ),

        floatingActionButtonLocation: FloatingActionButtonLocation.endFloat);

  }

}

*Listing 5. listing “pocetna\_stranica.dart.”*

### UI i logika stranice za pretragu

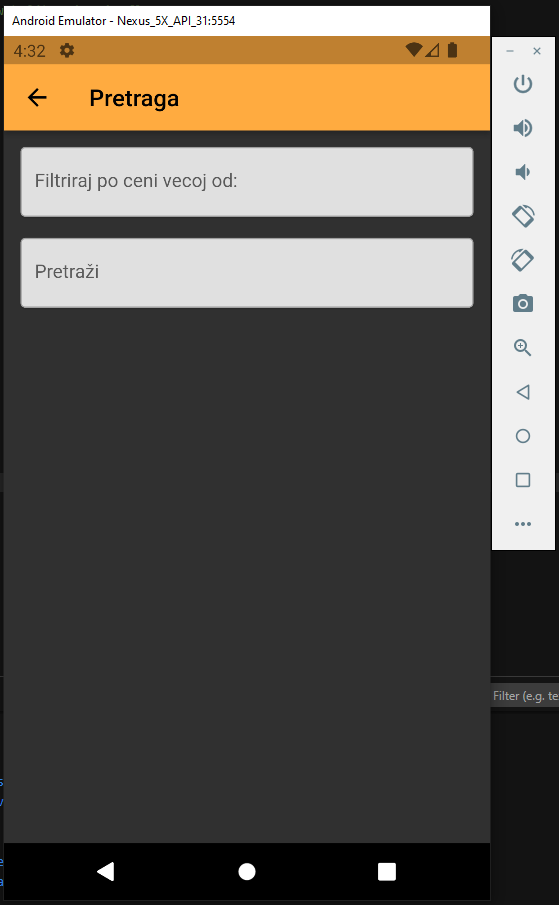
Pravimo stranicu “**pretraga\_stranica.dart**”, kao što sam naziv kaže ova stranica će nam služiti da pretražimo zakazane termine po nazivu modela koji treba da se popravlja ili servisira, međutim osim toga služiće nam i za filtriranje termina po ceni koja je veća od unete kao minimalni kriterijum.

UI ove stranice je sličan kao i UI početne stranice sa tim što na ovoj stranici imamo dva **TextField**-a u kojima unosimo kriterijume za pretragu i filtriranje.

Na svaku promenu koja se desi u ova dva **TextField**a-a pozivaju se dve funkcije u zavisnosti koje polje se menja.

Što se tiče polja za pretragu na čiju promenu se poziva funkcija **searchFromFirebase** koja uzma kao argument ono što je unešeno u TextField i upoređuje ga sa nizom iscepkanih stringova od stringa koji predstavlja model automobila. Taj niz se generiše u **\_create** i **\_update** funkciji koja se dešava pre **searchFromFirebase**-a.

Dok za funkciju **filter** nije potrebno generisanje niza već samo zadati parametar po kome uporedjuje cene termina sa njim.



*Slika 9. UI stranice za pretragu.*

// ignore\_for\_file: unused\_import, depend\_on\_referenced\_packages, prefer\_const\_constructors

import 'package:cloud\_firestore/cloud\_firestore.dart';

import 'package:firebase\_core/firebase\_core.dart';

import 'package:flutter/material.dart';

import 'dart:math';

class PretragaStranica extends StatelessWidget {

  const PretragaStranica({Key? key}) : super(key: key);

  @override

  Widget build(BuildContext context) {

    return MaterialApp(

      title: 'Pretraga',

      theme: ThemeData(

          primarySwatch: Colors.grey,

          // ignore: deprecated\_member\_use

          accentColor: Colors.grey),

      debugShowCheckedModeBanner: false,

      home: const PretraziStranica(),

    );

  }

}

class PretraziStranica extends StatefulWidget {

  const PretraziStranica({Key? key}) : super(key: key);

  @override

  State<PretraziStranica> createState() => \_PretraziStranicaState();

}

class \_PretraziStranicaState extends State<PretraziStranica> {

  List searchResult = [];

  void searchFromFirebase(String query) async {

    final result = await FirebaseFirestore.instance

        .collection('products')

        .where('makeArr', arrayContains: query)

        .get();

    setState(() {

      searchResult = result.docs.map((e) => e.data()).toList();

    });

  }

  List filterResult = [];

  void filter(String value) async {

    final int? number = int.tryParse(value);

    final result = await FirebaseFirestore.instance

        .collection('products')

        .where('price', isGreaterThan: number)

        .get();

    setState(() {

      searchResult = result.docs.map((e) => e.data()).toList();

    });

  }

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(

          backgroundColor: Colors.orangeAccent,

          title: const Text(

            "Pretraga",

            style: TextStyle(color: Colors.black),

          ),

          leading: IconButton(

            icon: Icon(

              Icons.arrow\_back,

              color: Colors.black,

            ),

            onPressed: () {

              Navigator.pop(context);

            },

          )),

      body: Container(

        color: Colors.grey[850],

        child: Column(

          children: [

            Padding(

              padding: const EdgeInsets.only(

                top: 10,

                left: 10,

                right: 10,

              ),

              child: Card(

                color: Colors.grey[300],

                child: TextField(

                  decoration: InputDecoration(

                    enabledBorder: OutlineInputBorder(

                        borderSide: BorderSide(color: Colors.grey)),

                    focusedBorder: OutlineInputBorder(

                        borderSide: BorderSide(color: Colors.grey)),

                    focusColor: Colors.orangeAccent,

                    hintText: 'Filtriraj po ceni vecoj od:',

                  ),

                  onChanged: (value) {

                    filter(value);

                  },

                ),

              ),

            ),

            Padding(

              padding: const EdgeInsets.all(10),

              child: Card(

                color: Colors.grey[300],

                child: TextField(

                  decoration: InputDecoration(

                    enabledBorder: OutlineInputBorder(

                        borderSide: BorderSide(color: Colors.grey)),

                    focusedBorder: OutlineInputBorder(

                        borderSide: BorderSide(color: Colors.grey)),

                    focusColor: Colors.orangeAccent,

                    hintText: 'Pretraži',

                  ),

                  onChanged: (query) {

                    searchFromFirebase(query);

                  },

                ),

              ),

            ),

            Expanded(

              child: ListView.builder(

                itemCount: searchResult.length,

                itemBuilder: (context, index) {

                  return Card(

                    color: Colors.grey[500],

                    margin: EdgeInsets.all(10),

                    child: ListTile(

                      textColor: Colors.grey[850],

                      title: Text(

                        searchResult[index]['name'],

                        style: TextStyle(color: Colors.black),

                      ),

                      subtitle: Text(searchResult[index]['make'] +

                          '\n' +

                          searchResult[index]['date'] +

                          '\n' +

                          'Cena: ' +

                          searchResult[index]['price'].toString() +

                          ' rsd'),

                    ),

                  );

                },

              ),

            ),

          ],

        ),

      ),

    );

  }

}

*Listing 6. listing “pretraga\_stranica.dart”*

## **Zaključak**

Mobilne tehnologije i programiranje nude pregršt korisnih opcija i mogućnosti koje se mogu koristiti svakodnevno u privatne i biznis shvrhe, primena ovih tehnologija je enormna u odnosu na svaku ostalu, gotovo 80% stanovništva koristi napredan mobilni uređaj na kome može za tren oka da instalira i koristi aplikaciju razvijenu ovim tehnologijama. Konkretno Flutter/dart tehnologija sa firebase-om je olakšala programerima još više i oslobodila ih zasebnih optimizacija za zasebne platforme.

Cilj ovog seminarskog rada je bio da se obuhvate osnove za kreiranje dobre mobilne aplikacije koja može imati široku primenu.

Dovoljno je samo biti upoznat sa ovakvom vrstom tehnologije i imati dobru ideju da bi se postigle velike stvari.

## **Literatura**

Literatura:

1. Marco L. Napoli - “Beginning Flutter: A Hands On Guide to App Development“, 2019

2. Eric Windmill - “Flutter in Action”, 2019

3.Gilad Bracha - “ The Dart Programming Language“, 2014.

Korišćeni web sajtovi:

1. <http://www.tfzr.uns.ac.rs/Predmet/mobilne-tehnologije-i-programiranje/sadrzaj-predmeta>

2. <https://docs.flutter.dev/>

3. <https://dart.dev/guides>

4. <https://code.visualstudio.com/docs>

5. <https://developer.android.com/docs>

## **Izvori preuzetih ilustracija**

|  |  |
| --- | --- |
| Slika 1 | https://cdn.dribbble.com/users/528264/screenshots/3140440/firebase\_logo.png |