Izrada aplikacije "Billy" za pregled i upravljanje osobnih računa korištenjem Flutter razvojnog okvira

Fajta, Ivan

Sveučilište u Rijeci – Fakultet informatike i digitalnih tehnologija

Jednopredmetna informatika

Ivan Fajta

Izrada aplikacije „Billy“ za pregled i upravljanje osobnih računa korištenjem Flutter razvojnog okvira

Završni rad

Mentor: izv. prof. dr. sc. Marija Brkić Bakarić

Rijeka, srpanj 2023.

Sveučilište u Rijeci
FIDIT Fakultet informatike i digitalnih tehnologija
www.inf.uniri.hr

Rijeka, 26.5.2023.

Zadatak za završni rad

Pristupnik: Ivan Fajta

Naziv završnog rada: Izrada aplikacije „Billy“ za pregled i upravljanje osobnih računa korištenjem Flutter razvojnog okvira

Naziv završnog rada na engleskom jeziku: Development of the Flutter application Billy for displaying and managing receipts

Sadržaj zadatka: Cilj rada je istražiti Flutter okvir za razvoj višeplatformskih aplikacija. U uvodnom dijelu prikazane su i ukratko objašnjene tehnologije za izradu višeplatformskih aplikacija, a potom je detaljnije prikazan razvojni okvir Flutter. U sklopu rada osmišljena je, razvijena i testirana aplikacija Billy za pregled i upravljanje osobnim računima te je prikazan proces izrade aplikacije. Poseban fokus stavljen je na primjenu NFC tehnologije. Na kraju rada izneseni su zaključci za usvajanje i korištenje razvojnog okvira Flutter.

Mentor

Izv. prof. dr. sc. Marija Brkić Bakarić

Voditelj za završne radove

Doc. dr. sc. Miran Pobar

Zadatak preuzet: 26.5.2023.

(potpis pristupnika)

Adresa: Radmile Matejčić 2
51000 Rijeka, Hrvatska
Tel: **+385(0)51 584 700**
E-mail: **ured@inf.uniri.hr**
OIB: **64218323816**
IBAN: **HR1524020061400006966**
uniri

SAŽETAK

Tema ovog završnog rada je izrada mobilne aplikacije za upravljanje osobnim računima čiji je cilj korisnicima omogućiti digitalno spremanje osobnih računa. U radu je prikazana izrada korisničke aplikacije i korištenje Firestore baze podataka koja služi kao poslužiteljski dio. Korisnički dio aplikacije implementiran je pomoću razvojnog okvira Flutter.

Ključne riječi: Firebase, Flutter, NFC tehnologija, widget

Abstract

The topic of this final thesis is the development of a mobile application for managing personal bills, the goal of which is to enable users saving their personal receipts in electronic form. This thesis describes the development process and the use of the Firestore database, which serves as the server part. The user part of the application is implemented using the Flutter development framework.

Keywords: Firebase, Flutter, NFC technology, widget

Sadržaj

1. UVOD

U današnje vrijeme potreba za povratom robe u trgovine je postala više od svakodnevice. Ali svi smo mi barem jednom u svom životu pretražili cijeli stan za tim jednim računom ili garancijom koja je završila u smeću. Istraživanje iz 2014 godine [1] navodi da je 10% otpada iz kućanstva zapravo bačeni račun. A danas u 2023 godini [2] Vam prijeti kazna za bacanje ili ne preuzimanje računa u trgovinama.
Tema završnog rada je " Mobilna aplikacija „Billy" za upravljanje osobnim računima ". Cilj je izraditi aplikaciju za mobilne uređaje kojima će biti omogućeno upravljanje osobnim računima korištenjem tehnologije Flutter. Flutter nam je paket koji koristimo za izradu aplikacija koje se mogu koristiti na više platforma (Android, IOS, Linux, Mac, Windows) kojeg je razvila firma pod imenom Google. Programski jezik koji koristimo za izradu Flutter aplikacija zove se Dart. Jezik Dart nam je objektno orijentirani programski jezik koji je zasnovan na klasama koje sadrže sintaksu u C stilu [1.].

U teorijskom djelu rada detaljnije će se proučiti način funkcioniranja Fluttera te kako se pomoću njega izrađuju aplikacije. Isto tako proučiti će se Firebase, odnosno baza podataka koja se koristiti kao sustav u pozadini „Billy" aplikacije. Firebase nam nudi veliki broj usluga za razno razne korisničke potrebe, od kojih će se koristiti prvenstveno registriranje i prijava korisnika odnosno pohranjivanje podataka korisnika, te pohrana korisnikovih računa.
U praktičnom dijelu rada cilj nam je pomoću Fluttera i Firebasea izraditi mobilnu aplikaciju za pohranu računa koja će korisniku omogućiti kreiranje korisničkog računa, prijavu u svoj korisnički račun, vlastoručno dodavanje osobnih računa, pregled svih dodanih računa, te komunikaciju između poslovnice i njegove aplikacije. Neke od dodatnih mogućnosti su pretraga računa po imenu firme, odnosno po datumu računa te promjena podataka za prijavu u korisnički račun.

2. OPIS KORIŠTENIH TEHNOLOGIJA

U ovom dijelu završnog rada naveli smo i opisali tehnologije koje smo koristili za izradu naše aplikacije. Prvo smo opisali programski jezik koji smo koristili, bazu podataka, zatim razvojni alat koji je korišten za izradu aplikacije za mobilne uređaje te uređivač koda koji smo koristili.

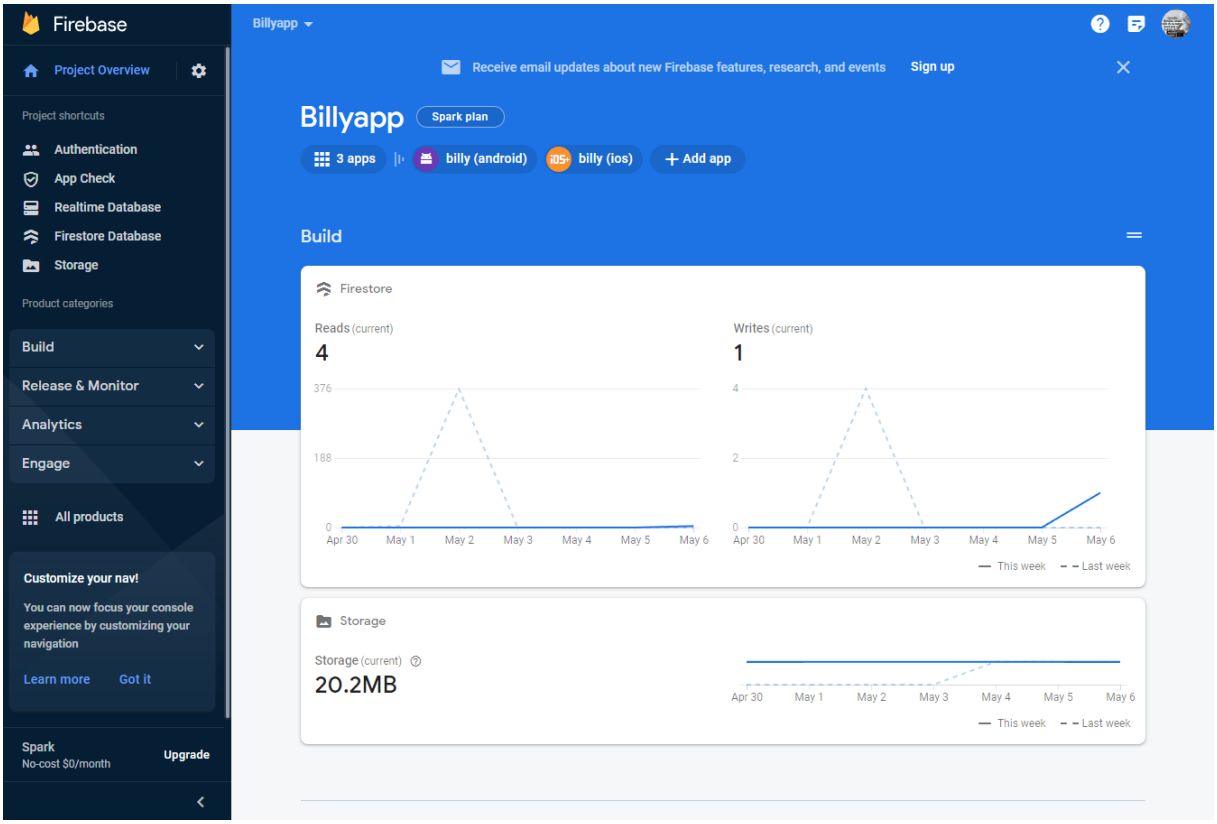
2.1. Dart

Dart je optimizirani programski jezik za klijenta za brze aplikacije na bilo kojoj platformi, koji je jednostavan za naučiti, s poznatom sintaksom. Taj jezik razvila je tvrtka pod imenom Google, koja ga koristi za izradu aplikacija za mobilne uređaje, aplikacija za računala za osobnu upotrebu, web aplikacija ili za kod koji će biti izvršen na poslužiteljima. Dart je svoju premijeru pred javnosti imao na GOTO konferenciji u Danskoj 2011. godine [2.]. Budući da Google većinu osobnih aplikacija stvara kao web-aplikacije, tvrtka je željela razviti programski jezik koji bi pomogao tim web-aplikacijama da rade i izgledaju još bolje. Tako je rođen Dart, a glavna namjena ovog programskog jezika bila je izrada web-aplikacija [3.]. Prva stabilna te prva kompletna verzija ovog programskog jezika objavljena je dvije godine nakon prvog predstavljanja, u studenom 2013.[4.]. Iako Dart nikada nije bio u mogućnosti da postigne puni uspjeh kao jezik za razvoj web aplikacija, Google ga je odlučio koristiti u druge svrhe.

Stoga je Dart kao programski jezik u kojem se izrađuju aplikacije postao jedna od glavnih prednosti Flutter platforme. Ovaj objektno orijentirani jezik ima sve značajke modernog programskog jezika koji programerima olakšava posao. Stoga im se lako prilagođava jer je njegova sintaksa koja je vrlo slična programskom jeziku C i bavi se „skupljanjem smeća“ (engl. Garbage collecting), podržava sučelja, apstraktne klase i mnogo više. 2015. godine Dart je izbacio 1.9 nadogradnju, koja je po nama u potpunosti odbacila sve stare ideje tog programskog jezika, a Dart dobio priliku prevesti kod u JavaScript. Stoga se počeo koristiti na platformi Flutter [4.]. Dart kao programski jezik još uvijek je u razvoju,2.0 verzija je objavljena 2018. koja je nudila audio podršku, a trenutna najnovija verzija je 2.19.6. u kojem smo dobili neka mala poboljšanja.

2.2. Firebase

Firebase je web i platforma za mobilne uređaje s alatima i dizajniranom infrastrukturom da pomogne razvojnim programerima u izradi aplikacija s visokom kvalitetom. Firebase nam se sastoji se osnovnih alata za razvoj web i aplikacija za mobilne uređaje koje programeri trebaju izgraditi na vlastitim poslužiteljima. Firebase (Slika 1) nam je u početku pružao uslugu baze podataka u stvarnom vremenu. Koristeći ovu bazu podataka, programeri bi mogli implementirati bazu podataka u svoje aplikacije, koje bi zatim mogli koristiti u isto vrijeme više korisnika. S vremenom je Firebase postupno dodavao inovativne alate i tako privukao pozornost Googlea koji je zatim 2014. godine i odlučio kupiti tvrtku [6.].



Slika 1 Početna stranica Firebase konzole

Nakon što je Google kupio tvrtku i nazvao je Firebase, Googleovi inženjeri počeli su razvijati platformu koja je postala sve popularnija među programerima. Neki od Firebase alata korištenih za izradu ove aplikacije su:

- Auth – Alat za autentifikaciju određenog korisnika. Korisnički podaci su zaštićeni i pohranjeni na Firebase poslužiteljima. Ovaj nam alat nudi i određenu autentifikaciju putem drugih servisa kao što su Google, Facebook, Apple i slično.
- Cloud Storage – Koristi se za pohranu podataka (slike, zvuk, tekst ili bilo koja druga vrsta datoteke).
- Cloud Firestore – Baza podataka sa svojstvom omogućavanja pristupa podacima putem pametnih telefona, web aplikacija i poslužitelja.

2.3. Flutter

Kao najbitnija osnova za razvoj i ideju ove aplikacije poslužila nam je platforma za razvoj hibridnih mobilnih aplikacija pod imenom Flutter. Flutter je razvio Google i predstavljen na Dart Developer Summitu 2015. Google je na ovoj konferenciji najavio Flutter kao inovativnu platformu koja će omogućiti korisnicima diljem svijeta pokretanje mobilnih aplikacija na Android operativnom sustavu pri 120 sličica u sekundi bez korištenja programskog jezika Java [7].Međutim, prva javna alfa verzija platforme objavljena je tek u svibnju 2017. [8]. Stoga je Google odlučio ovu platformu objaviti kao open source kako bi svi mogli doprinijeti njenom razvoju. Uz pomoć neprofesionalnih programera, platforma je razvijana godinama do prosinca 2018.objavljena je prva stabilna verzija [9]. Od tada, popularnost ove platforme raste iz dana u dan zbog svojih značajki i performansi koje nudi.

Kada govorimo o prednostima Fluttera u odnosu na druge platforme koje se također služe za razvoj hibridnih mobilnih aplikacija, često govorimo o Flutter engineu za prikaz koda na mobilnom telefonu. Launcher je napisan u C++ programskom jeziku i koristimo ga za prikaz 2D aplikacija. Osim toga, ovaj pokretač izravno komunicira s uređajem koji pokreće kod, što ne utječe na brzinu i učinkovitost aplikacije. Osim toga, pokretač podržava različite postavke operativnog sustava na kojem se pokreće, što čini da aplikacije izrađene pomoću Flutter platforme izgledaju i ponašaju se gotovo identično nativnim aplikacijama [10.].

Widgeti su temeljna komponenta svih Flutter aplikacija. Za razliku od drugih razvojnih platformi s različitim prikazima, kontrolerima pogleda, izgledima i drugim resursima, Flutter koristi samo widgete. S widgetima možemo raditi razne stvari, od dodavanja gumba ili izbornika, mijenjanja stila teksta ili boje fonta, dodavanja odjeljaka itd. Widgeti su

raspoređeni u strukturu stabla po aplikaciji, sa svakim novim widgetom koji dodamo u app add, prikazan na dnu stabla dok prikuplja sva svojstva widgeta izravno iznad njega [10.].Uz mali isječak programskog koda 1, iz završnog rada demonstrirat ćemo widget koji koristimo u Flutter platformi za prikaz određenog teksta na ekranu.

```
const Text(
  "Opps!..",
  style: TextStyle(
      fontSize: 40,
      fontWeight: FontWeight.bold),
),  // Text
```

Programski kod 1 Widget teksta

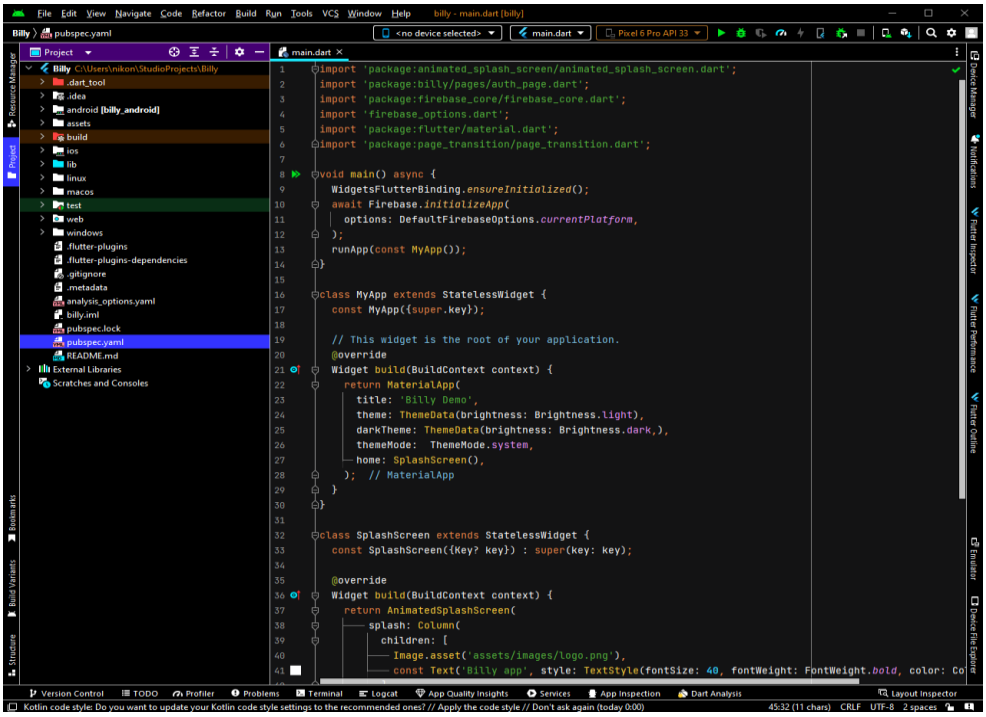
Taj widget prikazuje na ekranu poruku s tekstom „Opps!..“, veličine slova 40 te boldanim naglaskom.

Trenutačno se Flutter platforma koristi kod mnogih tvrtki ili pojedinaca, a svakim danom pridobije sve vise korisnika. Svemu tome je razlog upravo njena jednostavnost, brzina razvoja mobilnih aplikacija te mnoge njene ostale prednosti. Flutter platforma se razvija i dalje, svakim danom se dodaju novi widgeti, ispravljaju greške i to sve zahvaljujući Google-ovim inženjerima ili ostalim razvojnim programerima koji su smislili način kako nešto poboljšati. Trenutna verzija ove platforme u trenutku pisanja ovog rada je 3.7.12 koja je izašla 20. 04. 2023 [11.] te podržava razvoj aplikacija za mobilne uređaje sa Android i iOS operativnim sustavom, web aplikacija i aplikacija za osobna računala s Microsoft Windows, MacOS i Linux operativnim sustavima.

2.4. Android studio

Android Studio nam je službeno [12.] integrirano okruženje za razvoj (IDE), a operativni sustav Google Android, izgrađeno nam je na JetBrains IntelliJ IDEA i dizajnirano posebno za razvoj na Android uređajima . [13.] Može se preuzeti na uređajima s operativnim sustavom Windows, MacOS, ChromeOS i Linux. Android Studio najavljen je 16.svibnja 2013. na konferenciji Google I/O, a prva stabilna verzija za upotrebu izbačena je u prosincu 2014., počevši od verzije 1.0. [12.].Trenutna verzija Android studia za vrijeme pisanja ovog rada je

Flamingo (2022.2.1) koja je izašla u travnju 2023. [14.] a trenutni izgled stranice je prikazan na slici 2. Neki od jezika koje podržava su Java, C++, Kotlin [15.] i ostale koje sadrže proširenja.



Slika 2 Android studio, izgled početne stranice

Slika 3 Android Emulator

Isto tako jedna od bitnijih stvarni kod izrade aplikacije u Android studiju je i Android Emulator [16.]. Android Emulator simulira Android uređaje na računalu tako da možemo testirati svoju aplikaciju na različitim uređajima i razinama Android API-ja bez potrebe za svakim fizičkim uređajem. Korišteni android emulator u projektu prikazan je na slici 3 . Neke od prednosti su mu:

- Fleksibilnost: Osim što možemo simulirati razne uređaje i Android API razine, emulator nam dolazi s unaprijed određenim konfiguracijama za razno razne Android telefone, tablete, WearOS i Android TV uređaje.

- Brzina: Testiranje aplikacije na emulatoru je na neki način brže i lakše nego na fizičkom uređaju.
- Velika realnost uređaja: emulator nudi gotovo sve funkcije pravog Android uređaja. Možete simulirati dolazne telefonske pozive i tekstualne poruke, locirati svoj uređaj, simulirati različite brzine mreže, simulirati rotaciju i još mnogo toga.

2.5. Android operacijski sustav

Android je operativni sustav koji sadrži otvoreni koda za mobilne uređaje kao što su nam pametni telefoni i tableti, a razvila ga je američka tvrtka Google Inc[17]. Android je izvorno razvio Android Inc., koji je Google kupio 2005. [17.]. Android je prvi put svjetlo dana vidio 2007. godine, a prvi Android uređaji su na tržište stigli u rujnu 2008. [17].Android se temelji na modificiranoj verziji poznatog Linux kernela sustava i drugog softvera koji sadrži otvoreni kod koji je izvorno razvijen za uređaje koji imaju zaslon osjetljiv na dodir.

2.6. NFC tehnologija

NFC (engl. Near-Field Communication) zapravo opisuje niz komunikacijskih uređaja koji se mogu koristiti za bežičnu komunikaciju između uređaja na vrlo maloj udaljenost. Većinom jedan od uređaja dolazi bez ikakvog napajanja,primjerice kartice, naljepnice ili privjesci, ali onda te uređaje napaja drugi uređaj koji se koristi za NFC komunikaciju (Slika 4).

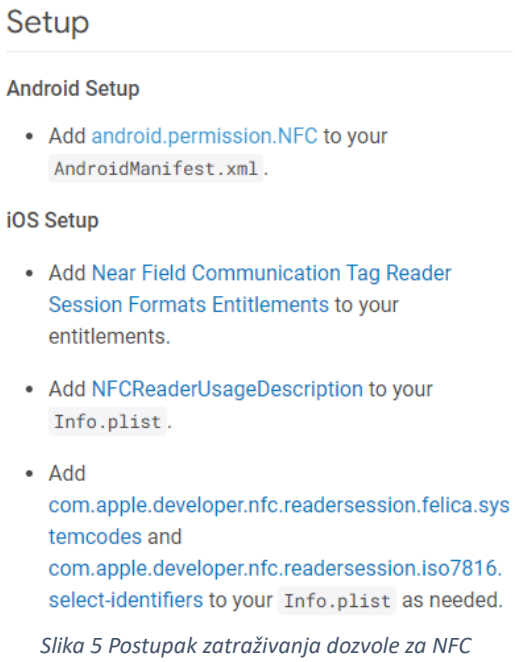
Slika 4 NFC uređaji

Uređaji dolaze u različitim veličinama pohrane, različitim udaljenostima prijenosa podataka, podršci samo za prijenos podataka, te ili podršci i za pisanje podataka i zaključavanje podataka na drugi uređaj. NFC Data Exchange Format ili poznatije NDEF je softver koji omogućava da svi kompatibilni uređaji razumiju strukturu i tip podataka koji su spremljeni na čip. Podaci koji se prenose mogu biti fotografije, videozapisi, dokumenti ili čak plaćanje usluga i proizvoda.

Prvi puta NFC tehnologija se pojavljuje na Android verziji v4.0 koja je izašla 2011 godine [19.] te donijela revoluciju u prijenosu podataka. S obzirom da svi mobilni uređaji ne sadrže NFC tehnologiju integriranu, potrebno je obratiti pozornost na oznaku „N“ na poleđini uređaja.

2.7. NFC manager

Veza između fluttera i NFC-a je veoma dobro pokrivena jer postoji više različitih paketa koji su povezani s NFC tehnologijom. Jedan od najbolje povezanih paketa je „nfc_menager“ koji podržava Android i iOS uređaje. Također taj paket se pokazao kao veoma pouzdan paket za korištenje. Naravno za korištenje tog paketa na uređaju bilo to Android ili iOS potrebno je prvo provjeriti da li uređaj sadrži NFC tehnologiju. Nakon toga ako smo ustanovili da uređaj sadrži tehnologiju, potrebno je zatražiti dozvolu od uređaja za korištenjem tehnologije. Postupak zatraživanja dozvole se razlikuje između Androida te iOS-a a postupak je prikazan na slici 5.


Setup

Android Setup

- Add android.permission.NFC to your `AndroidManifest.xml`.

iOS Setup

- Add Near Field Communication Tag Reader Session Formats Entitlements to your entitlements.
- Add NFCReaderUsageDescription to your `Info.plist`.
- Add com.apple.developer.nfc.readersession.felica.systemcodes and com.apple.developer.nfc.readersession.iso7816.select-identifiers to your `Info.plist` as needed.

Slika 5 Postupak zatraživanja dozvole za NFC

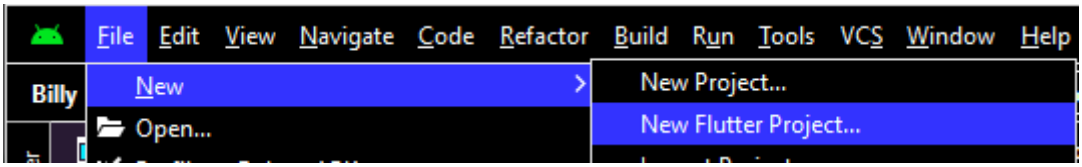
Postupak za pokretanje „NfcManager“ je veoma jednostavan. Prvo moramo provjeriti da li sustav podržava uopće NFC tehnologiju. Ako uređaj podržava onda sustav odrađuje ono šta je korisnik zamislio, a isječak programskog koda je prikazan pod brojem 2.

```
// Check availability
bool isAvailable = await NfcManager.instance.isAvailable();

// Start Session
NfcManager.instance.startSession(
  onDiscovered: (NfcTag tag) async {
    // Do something with an NfcTag instance.
  },
);

// Stop Session
NfcManager.instance.stopSession();
```

Programski kod 2 provjera dostupnosti NFC tehnologije

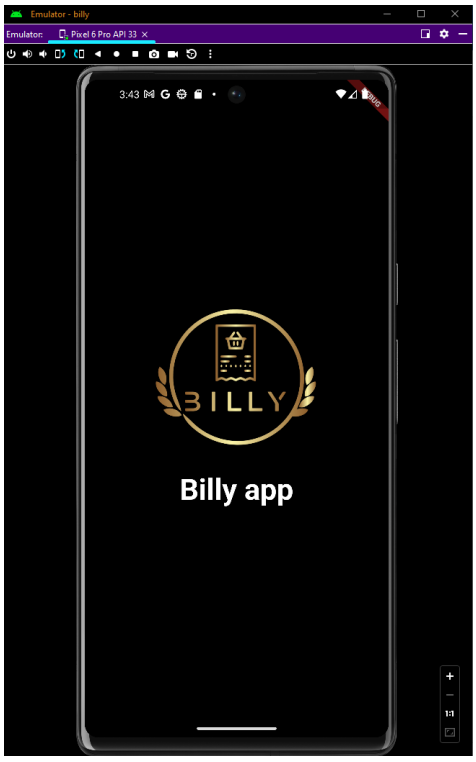
3. Razvoj aplikacije

U ovom poglavlju objasnit ćemo i opisati postupak razvoja aplikacije. Prije svega potrebnu je stvoriti novi projekt u Flutteru. Njega stvaramo pomoću programa Android studio u kojem je instaliran dodatak Flutter a jedan od primjera kako se stvori novi projekt je prikazan na slici 6 .



Slika 6 Stvaranje novog projekta

Aplikacija se sastoji od 12 zaslona te 3 pomoćne stranice koje nam služe za lakše stvaranje novih stanica s istim dizajnom. Cijela aplikacija počinje sa Splash zaslonom koji nam pokazuje logo aplikacije slika 7. Također aplikacija nam je dostupna i u dvije teme a to su „Dark mode“ i „Light mode“ koje se prilagođavaju temi mobitela odnosno kako je korisnik konfigurirao svoj mobitel što nam je prikazano na isječku programskog koda 3.



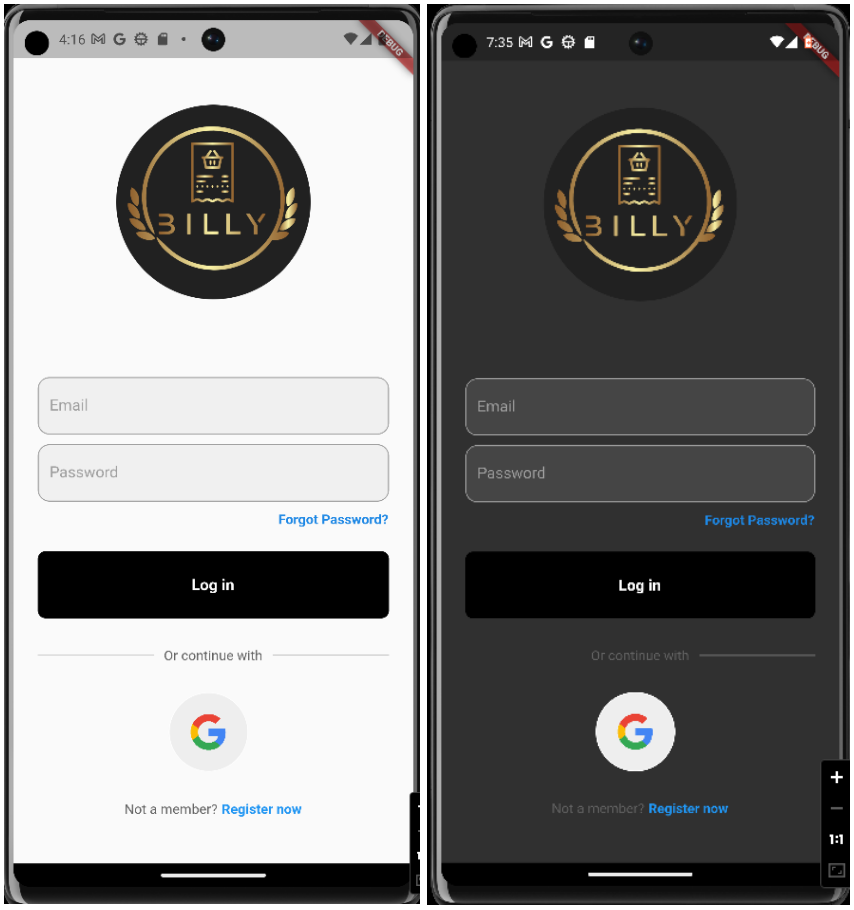
Slika 7 Izgled Splash zaslona

```
theme: ThemeData(brightness: Brightness.light),
darkTheme: ThemeData(brightness: Brightness.dark,),
themeMode:  ThemeMode.system,
home: SplashScreen(),
```

Programski kod 3 Konfiguriranje teme prema sistemu

3.1. Prijava korisnika

Nakon što splash screen odradi svoje korisnik se susreće sa zaslonom za prijavu korisnika prikazan na slici 8 s dva polja za unos informacija potrebnih za prijavu. Prije polja za unos prikazan nam je logo aplikacije u svojoj prepoznatljivoj kombinaciji. Prvo polje od korisnika traži da unese adresu e-pošte koju je koristio za registraciju u aplikaciji. U drugo polje potrebno je unijeti lozinku koju je korisnik postavio prilikom registracije u aplikaciju. Ako je korisnik zaboravio lozinku ima mogućnost stisnuti na gumb „Forgot Password“ koji će ga odvesti na poseban zaslon koji ćemo prikazati kasnije.



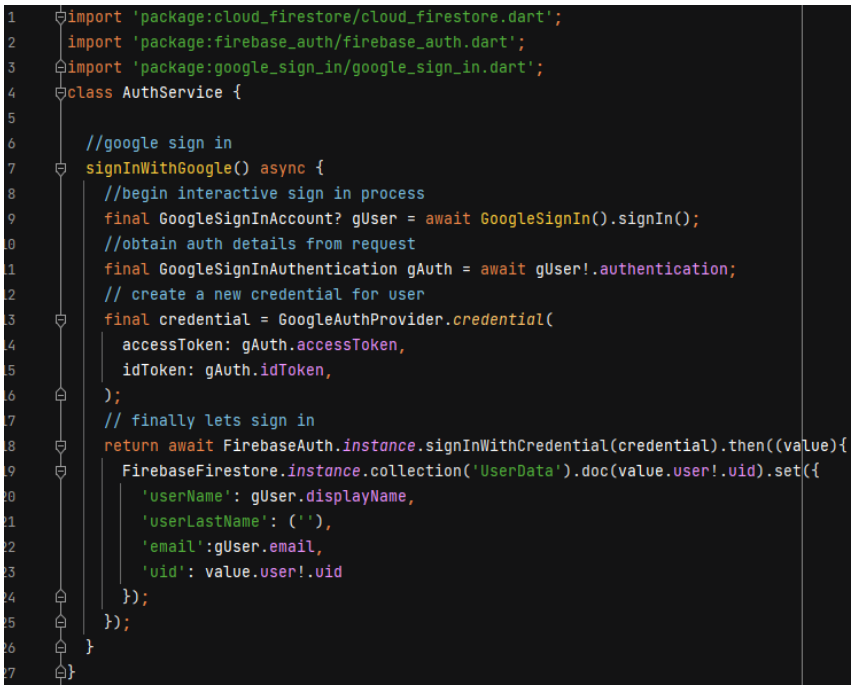
Slika 8 Zaslon za prijavu korisnika

Sljedeći nam je gumb „Log in“ koji ako su uneseni potrebni parametri poziva funkciju auth_page koja komunicira pomoću određenih metoda s Firebase servisom prikazane na isječku programskog koda 4 Pomoću te metode ćemo dobiti nazad podatke prijavljenog korisnika ili ako korisnik unese krive podatke bit će vraćen nazad na zaslon za prijavu korisnika. Kada se svi podaci provjere korisnik će biti proslijeđen na početni zaslon aplikacije.

```dart
import 'package:billy/pages/HomePage.dart';
import 'package:billy/pages/LoginOrRegisterPage.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:flutter/material.dart';

class AuthPage extends StatelessWidget {
  const AuthPage({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: StreamBuilder<User?>(
        stream: FirebaseAuth.instance.authStateChanges(),
        builder: (context, snapshot){
          //user is logged in
          if (snapshot.hasData){
            return HomePage();
          }
          //user is not logged
          else {
            return LoginOrRegisterPage();
          }
        },
      ),  // StreamBuilder
    );  // Scaffold
  }
}
```

Programski kod 4 Prikaz komunikacije s Firebase servisom za prijavu

Sljedeća metoda za prijavu je puno jednostavnija od unosa korisničkih podataka a to je prijava pomoću Google servisa koji prilikom pritiska na gumb s logom firme Google potražuje podatke od Google servisa te ih provjerava u funkciji auth_service koja komunicira pomoću određenih metoda s Google servisom koje su prikazane na isječku programskog koda 5. te ako korisnik nije prije bio prijavljen automatski ga registrira u aplikaciji te sprema njegove podatke u Firebase servis.



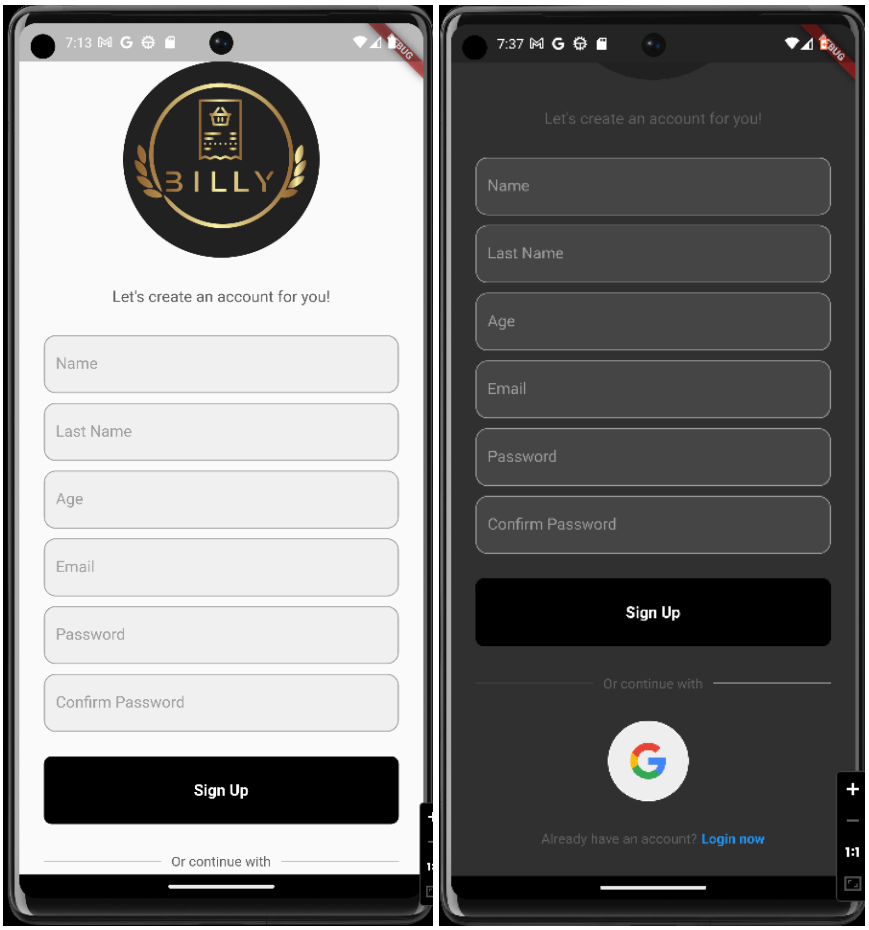
```dart
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:google_sign_in/google_sign_in.dart';
class AuthService {

  //google sign in
  signInWithGoogle() async {
    //begin interactive sign in process
    final GoogleSignInAccount? gUser = await GoogleSignIn().signIn();
    //obtain auth details from request
    final GoogleSignInAuthentication gAuth = await gUser!.authentication;
    // create a new credential for user
    final credential = GoogleAuthProvider.credential(
      accessToken: gAuth.accessToken,
      idToken: gAuth.idToken,
    );
    // finally lets sign in
    return await FirebaseAuth.instance.signInWithCredential(credential).then((value){
      FirebaseFirestore.instance.collection('UserData').doc(value.user!.uid).set({
        'userName': gUser.displayName,
        'userLastName': (''),
        'email':gUser.email,
        'uid': value.user!.uid
      });
    });
  }
}
```

Programski kod 5 Prikaz komunikacije s Google servisom za prijavu

Ako korisnik nema podatke za prijavu niti se želi prijaviti Google servisom ostaje mu mogućnost registracije u aplikaciji a to je omogućeno klikom na gumb „Register now“. Taj gumb će ga odvest na poseban zaslon koje korisnik mora ispuniti.

3.2 Registracija korisnika

Ako korisnik izabere postupak registracije pojavit će se novi zaslon koji je prikazan na slici 9. Na tom zaslonu korisnik je dužan unijeti sve potrebne osobne podatke. Ako korisnik izostavi neki od potrebnih podataka aplikacija će ga vratiti ponovo na zaslon za registraciju. Potrebni podaci su nam puno ime i prezime korisnika, njegova dob, email korisnika te lozinka za korisnički račun. Također korisnik mora ponoviti unos lozinke te ako se lozinke ne poklapaju korisniku neće biti omogućena registracija.



Slika 9 Zaslon registracije korisnika

Za unos potrebnih podataka smo koristili Textfield widget koji je prikazan u programskom kodu 6.

```
Textfield(
  controller: firstNameController,
  hintText: 'Name',
  obscureText: false,
),  // Textfield
```

Programski kod 6 Textfield

Kada korisnik unese sve potrebne podatke za prijavu klikom na gumb „Sign up“ se prvo provjerava ako se unesene lozinke podudaraju što je vidljivo na isječku iz programskog koda 7.

```dart
bool passwordConfirmed() {
  if (passwordController.text.trim() == confirmPasswordController.text.trim()) {
    return true;
  } else
    return false;
}
```

Programski kod 7 Provjera podudarnosti lozinki

Ako se lozinke podudaraju svi uneseni podaci spremaju se u Firebase servis te se korisnik prosljeđuje na početnu stranicu aplikacije što je vidljivo na isječku programskog koda 8, a ako se lozinke ne podudaraju korisniku se vraća na stranicu i traži se ponovni unos lozinke.

```dart
void signUserUp() async {
  //show loading circle
  showDialog(
    context: context,
    builder: (context) {
      return const Center(
        child: CircularProgressIndicator(),
      );  // Center
    },
  );
  // try creating user
  try{
    //check if password are same
    if (passwordConfirmed()) {
      //create user
       FirebaseAuth.instance.createUserWithEmailAndPassword(
        email: emailController.text,
        password: passwordController.text).then((value){
          FirebaseFirestore.instance.collection('UserData').doc(value.user!.uid).set({
            'userName': firstNameController.text.trim(),
            'userLastName': lastNameController.text.trim(),
            'email':emailController.text.trim(),
            'password': passwordController.text.trim(),
            'age': int.parse(ageController.text.trim()),
            'uid': value.user!.uid
          });
       });
    }

  } on FirebaseAuthException catch (e) {
    //pop loading circle
    Navigator.pop(context);
  }
}
```

Programski kod 8 Registracija korisnika na Firebase

Također korisnik se može registrirati i preko Google servisa klikom na logo firme Google, ili ako je shvatio da je već registriran može se vratiti ponovo na stranicu za prijavu u aplikaciju.

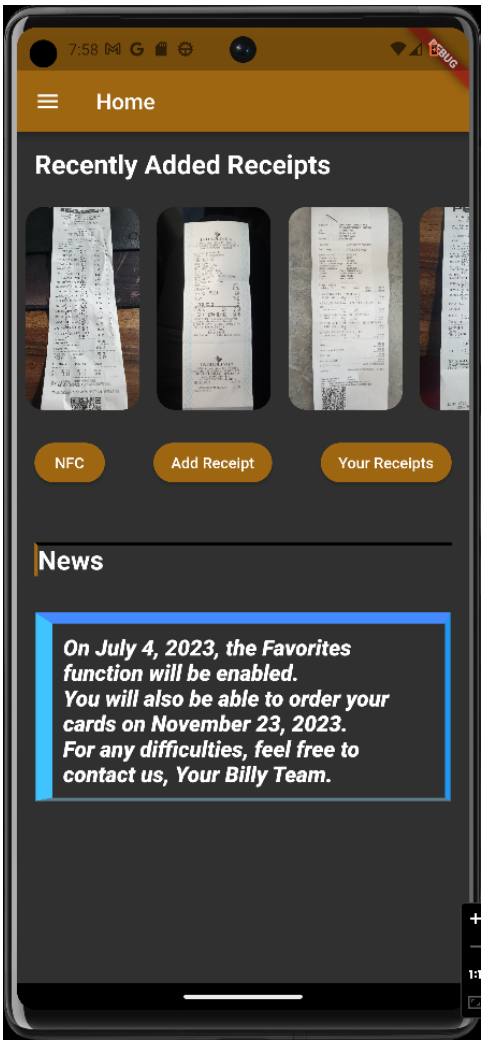
3.3 Početni zaslon „Home“

Početni zaslon je ujedno i glavni zaslon aplikacije iz kojeg se sve može bez imalo muke i pretraživanja odraditi. Na njemu u gornjem lijevom kutu imamo prikazan „Drawer meni“ odnosno izbornik koji nam je prisutan na svim zaslonima osim na prijavi i registraciji. Također u gornjem dijelu zaslona imamo prikazan naziv stranice koji je također prisutan na svim zaslonima. Prikaz početnog zaslona možemo pogledati na slici 10. Ispod widgeta „Appbar“ koji smo koristili za smještanje izbornika i naziva stranice koji možete pogledati na isječku programskog koda 9. postavljena je mini galerija posljednjih 5 računa koji su dodani u aplikaciju koji se mogu dodirom prsta pomicati lijevo desno radi otkrivanja preostalih posljednje dodatnih računa što vidimo na isječku programskog koda 10.

```
drawer: SideMenu(),
  appBar: AppBar(
    title: Text('Home'),backgroundColor: Color.fromRGBO(236, 144, 5, 100),),
```

Programski kod 9 Widget Appbar

Ispod posljednje dodanih računa imamo 3 gumba od kojih svaki ima različitu funkciju. Prvi gumb nam se naziva „NFC“ koji prilikom pritiska na taj gumb nas šalje na zaslon koji prilikom prislanjanja mobitela na čitač kod blagajne šalje potrebne podatke sustavu o korisniku koji je obavio kupnju u njihovoj trgovini. Sljedeći gumb je „Add Receipt“ odnosno dodavanje računa. Prilikom pritiska na taj gumb sustav nas šalje na novi zaslon na kojem imamo mogućnost dodati nove račune ručnim putem. Zadnji gumb koji sadrži ovaj zaslon je „Your Receipt“ koji će nas poslati na zaslon koji sadrži sve naše dodane račune gdje imamo više funkcija vezano za račune.



Slika 10 Snimka početnog zaslona

Sljedeće na toj stranici nam je prozor novosti u kojem developeri aplikacije mogu obavještavati korisnika o novostima u aplikaciji ili bilo kakvim bitnim promjenama vezano za njihov račun.

```dart
Container(
  height: 200.0,
  child: StreamBuilder<QuerySnapshot<Map<String, dynamic>>>(
    stream: _imagesStream,
    builder: (context, snapshot) {
      if (snapshot.hasData) {
        int itemCount = snapshot.data!.docs.length > 5 ? 5 : snapshot.data!.docs.length;
        return ListView.builder(
          scrollDirection: Axis.horizontal,
          itemCount: itemCount,
          itemBuilder: (context, index) {
            final data = snapshot.data!.docs[index].data();
            return Padding(
              padding: const EdgeInsets.all(8.0),
              child: ClipRRect(
                borderRadius: BorderRadius.circular(16.0),
                child: GestureDetector(
                  onTap: () {
                    Navigator.push(
                      context,
                      MaterialPageRoute(
                        builder: (context) =>
                            Scaffold(
                              appBar: AppBar(title: Text(data['firm']),),
                              body: Center(
                                child: Image.network(data['imageUrl']),
                              ),  // Center
                            ),  // Scaffold
                      ),  // MaterialPageRoute
                    );
                  },
                  child: Container(
                    child: Image.network(
                      data['imageUrl'],
                      fit: BoxFit.cover,
                    ),  // Image.network
                  ),  // Container
```

Programski kod 10 Posljednje dodani računi

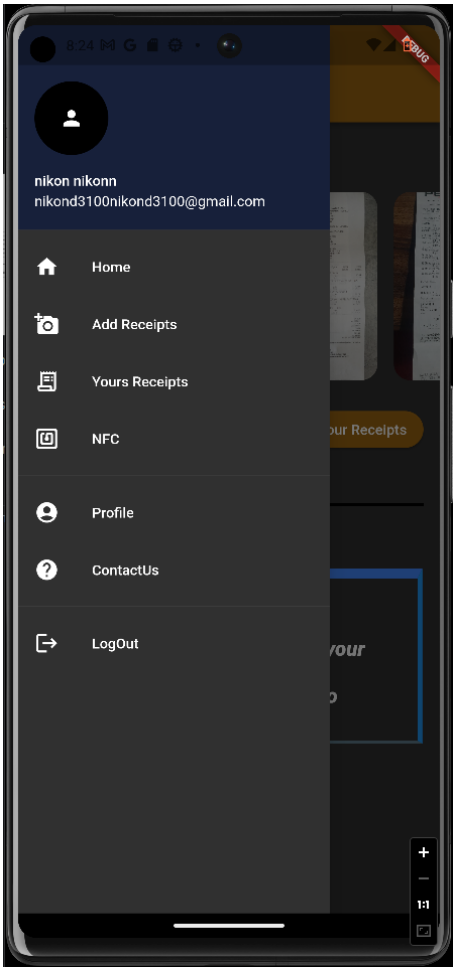
3.4 Izbornik

Izbornik je jedini dio u aplikaciji koji je na svakoj stranici isti i nema promjena. Izbornik možemo pogledati na slici 11. izbornik nam se sastoji od dva dijela. Prvi dio nam sadrži informacije o prijavljenom korisniku koje sustav povlači iz Firebase servisa što možemo vidjeti na isječku programskog koda 11, te drugog dijela u kojem imamo gumbe koji nas navode na određene zaslone.

```dart
class _SideMenuState extends State<SideMenu> {
  final user = FirebaseAuth.instance.currentUser!;
  String? name = '';
  String? lastname = '';
  Future _getDataFromDatabase() async{
    await FirebaseFirestore.instance.collection("UserData")
        .doc(user.uid)
        .get()
        .then((snapshot) async {
      if(snapshot.exists){
        setState(() {
          name = snapshot.data()!['userName'];
          lastname = snapshot.data()!['userLastName'];
        });
      }
    });
  }
```

Programski kod 11 Informacije o korisniku

Najbitnije informacije o korisniku u izborniku su njegovo ime i prezime te njegov email. Ispod toga sadržimo gumbe koji nas šalju na određene zaslone te sadržimo gumb „LogOut“ koji sluzi korisniku ako se želi odjaviti iz sustava te će ga se prilikom pritiska na taj gumb poslati nazad na zaslon za prijavu u aplikaciju.



Slika 11 Izbornik

3.5. Ručno dodavanje računa

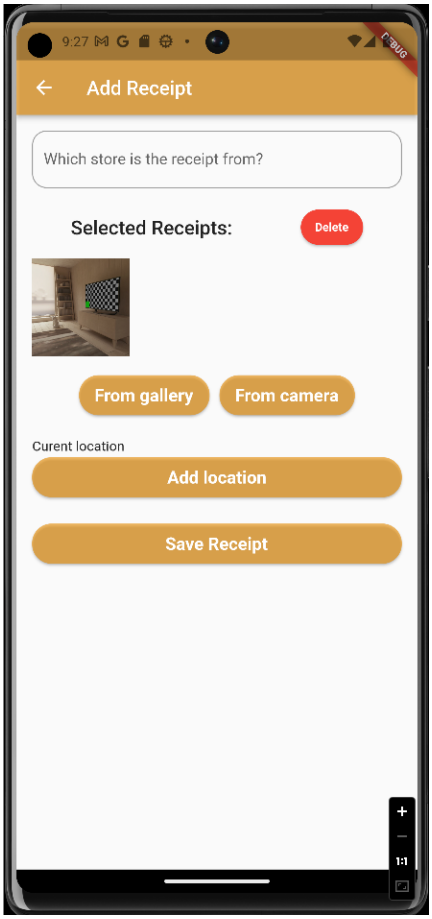
Ako korisnik želi dodati ručno neki novi račun to smo mu omogućili na dva načina. Prvi način je da korisnik može odabrati račun iz galerije ako ga je već prije slikao ili ga je imao pohranjenog. Te imamo mogućnost da ga korisnik sam uslika te da ga sustav sam prebaci u bazu podataka na Firebase servisu. Prilikom odabira računa iz galerije ili uslikanog računa ako nismo zadovoljni s izgledom ili smo krivoga odabrali imamo mogućnost i brisanja odabira što je vidljivo na isječku programskog koda 12. Izgled zaslona za ručno dodavanje računa

možemo pogledati na slici 12. Osim slike računa potrebni su još neki podaci koje korisnik mora sam unijeti ili prepustiti sustavu da odredi ih sam. Jedan od podataka koje korisnik mora sam unijeti je trgovina u kojoj je račun izdan.

```
void _deleteImage() {
  setState(() {
    _selectedImages = [];
  });
}
```

Programski kod 12 Brisanje odabranih slika

Jedan od bitnijih podataka je lokacija računa di je dodan odnosno lokacija trgovine u kojoj je račun izdan stoga taj podatak prepuštamo sustavu da sam odredi prilikom pritiska na gumb „Add location“.



Slika 12 Izgled zaslona za ručno dodavanje računa

Isječak programskog koda 13 prikazuje određivanje lokacije uređaja prilikom dodavanja novog računa što će nam poslije olakšati razvoj aplikacije.

```dart
ElevatedButton(
  onPressed: () {
    _addLocation().then((value) {
      lat = '${value.latitude}';
      long = '${value.longitude}';
      setState(() {
        locationMessage = 'Latitude $lat , Longitude: $long';
      });
    });
  },
  child: Text('Add location'),
  style: ElevatedButton.styleFrom(
    backgroundColor: Color.fromRGBO(236, 144, 5, 100), // text color
    shape: RoundedRectangleBorder(
      borderRadius: BorderRadius.circular(20), // rounded corners
    ),  // RoundedRectangleBorder
    padding: EdgeInsets.symmetric(horizontal: 16, vertical: 10), // button padding
    textStyle: TextStyle(
      fontSize: 18,
      fontWeight: FontWeight.bold,
    ), // button text style  // TextStyle
  ),
),  // ElevatedButton
```

Programski kod 13 Određivanje lokacije uređaja

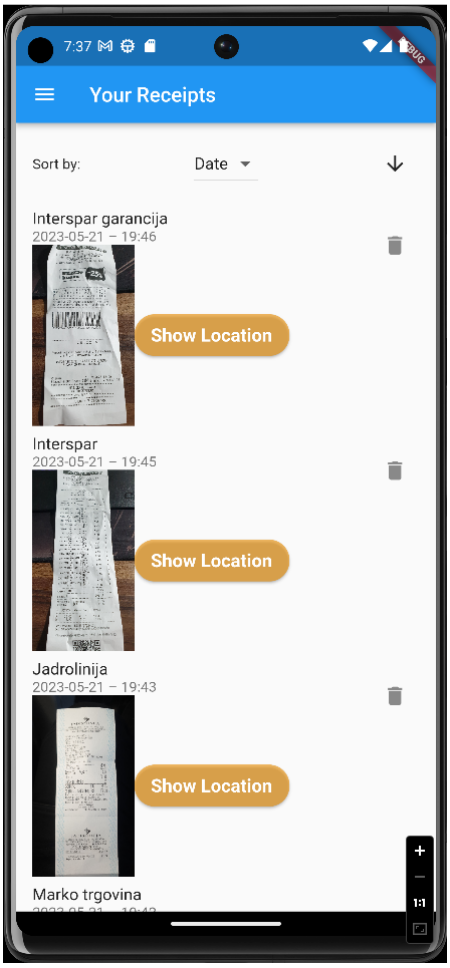
Prilikom ispune svih podataka vezanih za račun koji korisnik dodaje, korisnik je u mogućnosti da pritisne gumb „Save Receipt“ i spremi sve podatke na Firebase servis. Isječak programskog koda 14 prikazuje proces spremanja podataka u Firebase Storage i Firebase Database.

```dart
Future<void> _saveCollection() async {
  final String collectionName = _collectionNameController.text.trim();
  for (final imageFile in _selectedImages) {
    // Upload each image file to Firebase Storage
    final storageReference =
    FirebaseStorage.instance.ref().child('images/${Uuid().v4()}');
    final uploadTask = storageReference.putFile(imageFile);
    await uploadTask.whenComplete(() {});
    // Get the download URL for each uploaded image
    final downloadUrl = await storageReference.getDownloadURL();
    final metadata = {
      'firm': collectionName,
      'imageUrl': downloadUrl,
      'timestamp': FieldValue.serverTimestamp(),
      'useruid': user.uid,
      'Latitude': lat,
      'Longitude': long,
    };
    await FirebaseFirestore.instance.collection('Receipts').doc(user.uid)
        .collection('AllReceipts').doc(Uuid().v4())
        .set(metadata);
  }
  // Navigate back to the previous page
  Navigator.pop(context);
}
```

Programski kod 14 Spremanje ručno dodanih računa

3.6. Pregled računa

Ako korisnik se tek registrirao ili još nema unesenih računa ova stranica će biti prazna, ali ako je korisnik već unese svoje račune ili garancije onda će oni biti vidljivi na ovom zaslonu koji je prikazan na slici 13.



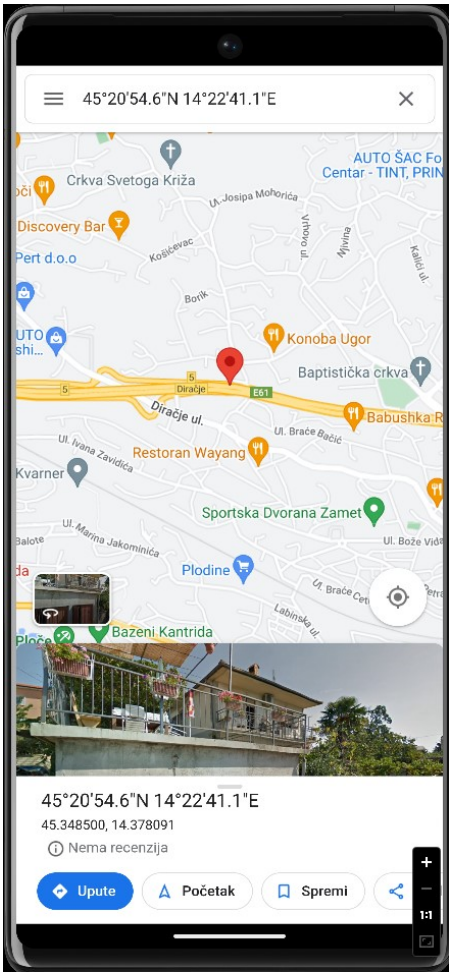
Slika 13 Zaslon pregleda računa

Prilikom otvaranja zaslona svi uneseni računi su sortirani po datumu dodavanja u sustav što možemo pogledati na isječku programskog koda 15. Ako korisnik poželi neki drugi način sortiranja ponudili smo i sortiranje prema imenu firme iz koje je račun. Također korisnik ima mogućnost i promijeniti da li želi prikazati uzlazno ili silazno klikom na strelicu u gornjem desnom kutu ekrana.

```dart
void _sortData(String field, bool isAscending) {
  setState(() {
    _sortBy = field;
    _isAscending = isAscending;
    _itemsStream = FirebaseFirestore.instance.collection('Receipts')
        .doc(user.uid).collection('AllReceipts')
        .orderBy(field, descending: !isAscending)
        .snapshots();
  });
}
```

Programski kod 15 Isječak sortiranja računa

Neki od ponuđenih podataka računa prilikom pregleda su imena firme iz kojeg su potekli, datum dodavanja u sustav te Google lokacija računa prilikom dodavanja u sustav. Ako korisnik želi pogledati lokaciju računa to može napraviti pritiskom na tipku „Show Location“. Nakon pritiska na tipku otvara se lokacija računa pomoću aplikacije „Google Maps“ te pokazuje korisniku točnu lokaciju dodavanja računa koja je prikazana na slici 14.

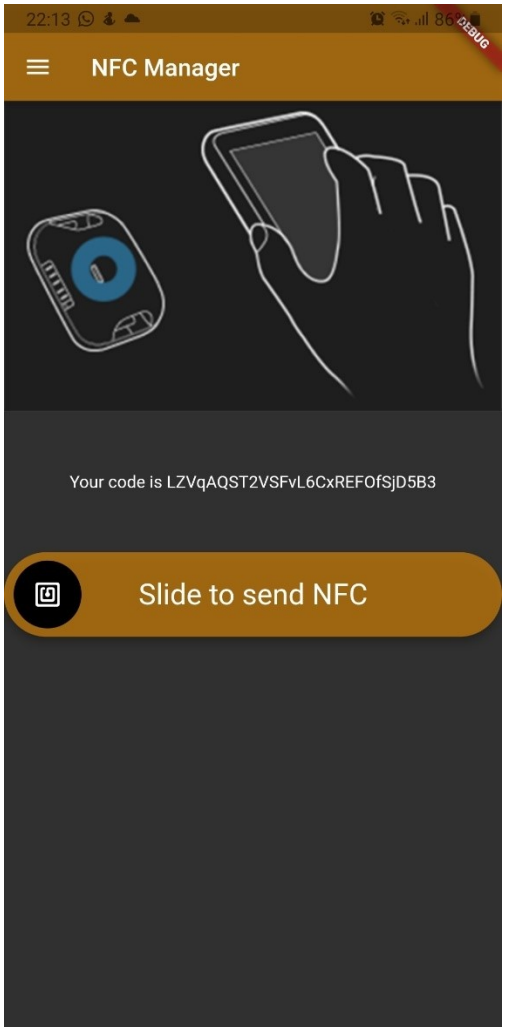


Slika 14 Prikaz lokacije dodavanja računa

Ako korisnik želi detaljnije pregledati račun to mu je omogućeno pritiskom na sliku računa di će se slika povećati preko cijelog ekrana. Također korisnik ima i mogućnost brisanja određenih računa pritiskom na ikonu „Trashbox“ koja se nalazi pored svakog računa koji je dodan u sustav.

3.7. NFC

Također omogućena je mogućnost i automatskog dodavanja računa u sustav direktno u trgovini. Za to imamo poseban zaslon koji se zove „NFC“ koji je prikazan na slici 15.



Slika 15 NFC zaslon

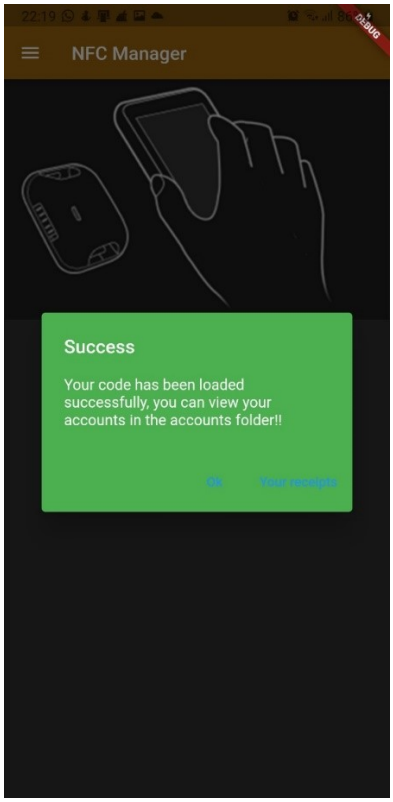
Ako firma ima ugovoreno s aplikacijom korisnik prilikom plaćanja prislanja mobitel na uređaj koji onda očitava kod i šalje direktno račun u sustav pod tim kodom korisnika, svaki korisnik ima svoj jedinstveni kod. Na vrhu zaslona sustav nam pomoću videozapisa prikazuje

kako se treba postupiti ako želimo preuzeti digitalni račun. Ispod videozapisa nalazi se jedinstveni kod ako je NFC uređaj zakazao kako bismo ga mogli unijeti ručno. Prije nego šta prislonimo mobitel na uređaj potrebno je pritisnuti i povući gumb i desnu stranu da bi se aktivirao NFC prijenos koda koji je omogućen isječkom programskog koda 16.

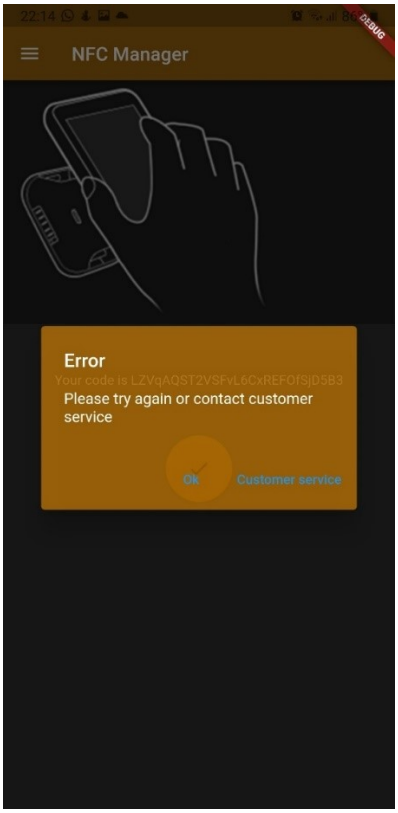
```
SlideAction(
  alignment: Alignment.bottomCenter,
  outerColor: Color.fromRGBO(236, 144, 5, 100),
  innerColor: Colors.black,
  sliderButtonIcon: const Icon(
    Icons.nfc_rounded,
    color: Colors.white,
  ),  // Icon
  text: 'Slide to send NFC',
  textStyle: const TextStyle(color: Colors.white, fontSize: 24,),
  onSubmit: () {
    NfcManager.instance.startSession(onDiscovered: (NfcTag tag) async {
```

Programski kod 16 Aktiviranje NFC uređaja

Ako je kod prenesen uspješno pojavi nam se „Pop up display“ s porukom o uspješnom prijenosu (Slika 16), a ako se dogodio neki problem prilikom prijenosa pojavi nam se ponovo „Pop up display“ ali ovaj puta s porukom o neuspješnom prijenosu (Slika 17).

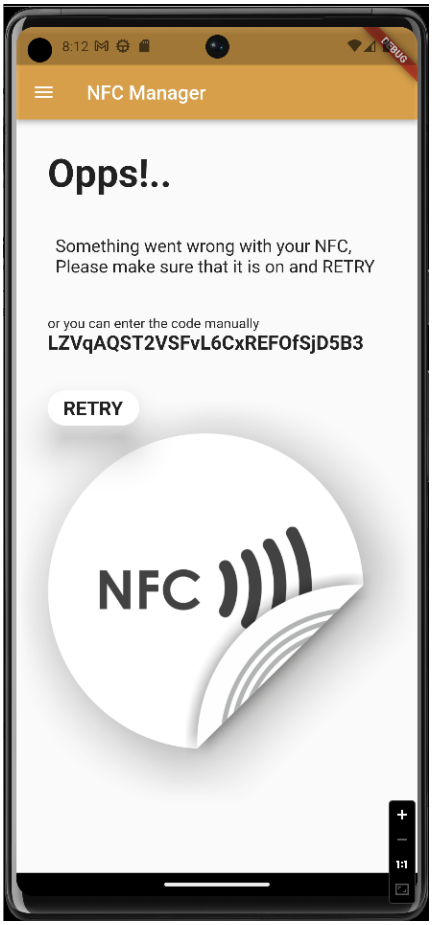


Slika 16 Poruka o uspješnom prijenosu koda



Slika 17 Poruka o neuspješnom prijenosu koda

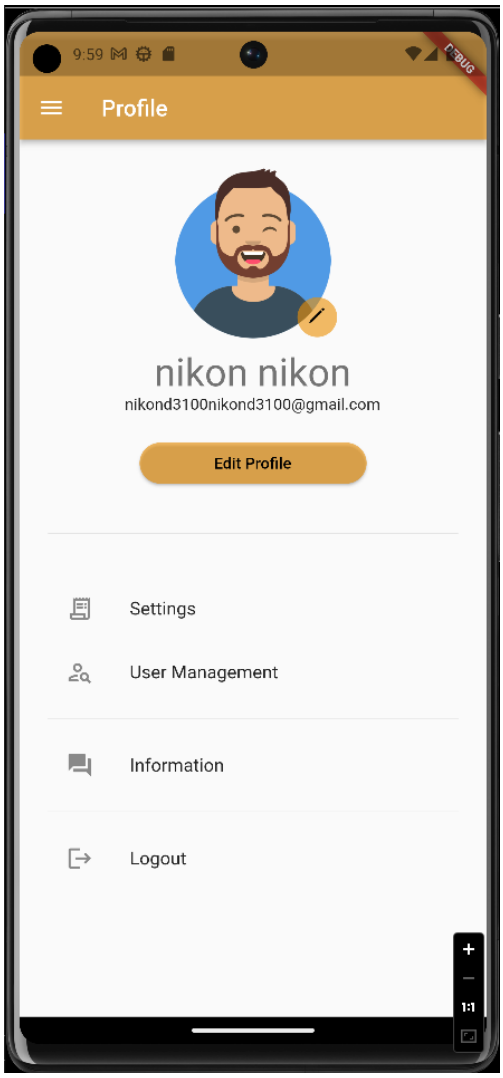
Isto tako moramo uzeti u obzir da neki uređaji nemaju NFC ili korisniku je ugašen. Ako se dogodi jedan od ta dva slučaja korisniku će prilikom pritiska na tipku „NFC“ izaći novi zaslon koji će ga upozoriti da nije moguće prenijeti kod putem NFC-a koji možemo pogledati na slici 18. Također smo mu omogućili da ručno može unijeti jedinstveni kod te prilikom pritiska na tipku „Retry“ korisnik osvježava stranicu i provjerava da li se problem riješio.



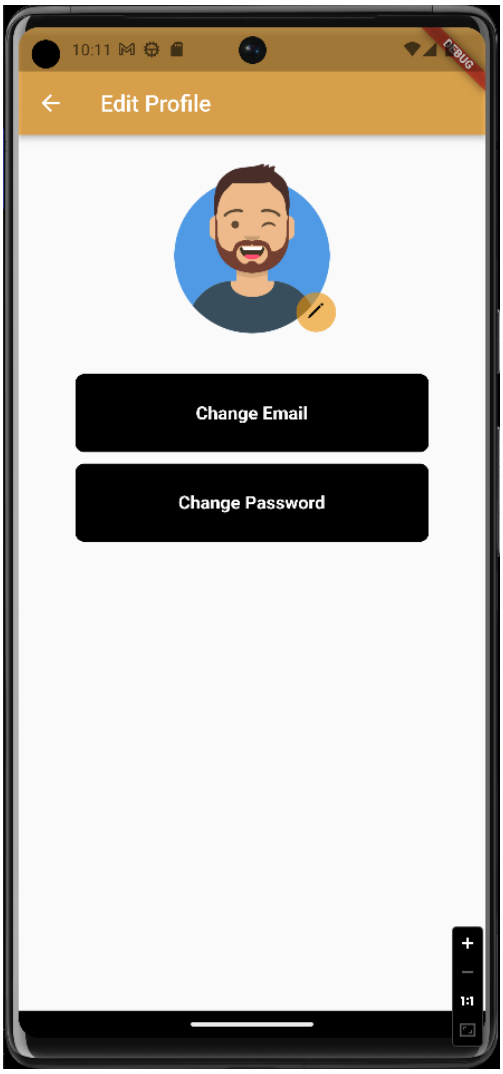
Slika 18 pristup NFC nije omogućen zaslon

3.8. Zaslon profila

Pošto naša aplikacija ima korisnike onda mora imati i stranicu za provjeru podataka ili izmjenu podataka korisnika. Kod nas to je „Profile“ zaslon koji nam omogućuje pristup osnovnim podacima korisnika te izmjenu nekih od tih parametara. Izgled stranice nam je prikazan na slici 19. Osim prikaza informacija omogućena nam je i promjena podataka kao što su lozinka ili email. Ako korisnik želi promijeniti neki od tih podataka mora pritisnuti tipku „Edit Profile“ koji će ga odvesti na novi zaslon di može birati koji podatak želi promijeniti. Izgled izbornika nam je prikazan na slici 20.



Slika 19 Zaslon profila



Slika 20 Izbornik promjene podataka

Prilikom promjene email adrese korisnik je dužan unijeti samo novu valjanu adresu, a za promjenu lozinke korisnik mora unijeti prvo staru lozinku onda mora unijeti novu lozinku te nakon toga mora ponovo unijeti novu lozinku za provjeru. Isječak programskog koda koji provjerava jesu li lozinke valjana prikazan nam je pod brojem 17. Te najbitniji dio u promjeni lozinke je spremanje lozinke koja nam je prikazana na isječku programskog koda 18.

```dart
TextFormField(
  controller: _confirmPasswordController,
  decoration: const InputDecoration(
    labelText: 'Confirm Password',
  ),  // InputDecoration
  obscureText: true,
  validator: (value) {
    if (value == null || value.isEmpty) {
      return 'Please confirm your new password.';
    }
    if (value != _newPasswordController.text) {
      return 'Passwords do not match.';
    }
    return null;
  },
),  // TextFormField
```

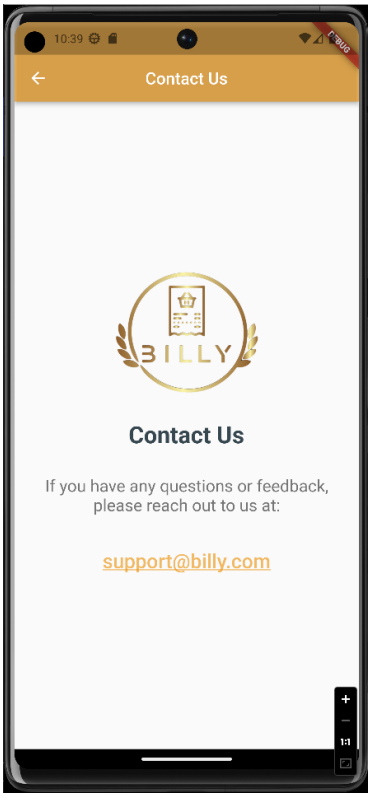
Programski kod 17 Provjera podudarnosti lozinke

```dart
try {
  final user = FirebaseAuth.instance.currentUser;
  final _firestore = FirebaseFirestore.instance;
  final credential = EmailAuthProvider.credential(
      email: user!.email!, password: _oldPasswordController.text);
  await user.reauthenticateWithCredential(credential);
  await user.updatePassword(_newPasswordController.text);
  await _firestore.collection('UserData').doc(user.uid).update({
    'password': _newPasswordController.text,
  });
```

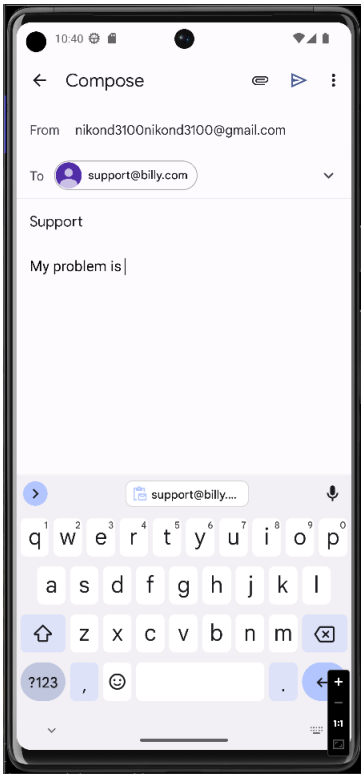
Programski kod 18 Spremanje nove lozinke

3.9 Pomoć

Ako korisnik zatreba bilo kakvu pomoć prilikom upotrebe aplikacije korisnička služba mu je uvijek dostupna na zaslonu „ContactUs“. Izgled zaslona nam je prikazan na slici 21. Stranica je prilično jednostavna te sadrži logo aplikacije, naziv stranice te widget „Button“ s tekstom support@billy.com.



Slika 21 ContactUs zaslon



Slika 22 Izgled predloška za slanje problema

Prilikom pritiska na taj widget korisnika se odmah šalje na predodređenu aplikaciju za slanje mailova. Aplikacija automatski popunjava mail na koji se šalje problem, naslov maila te početak poruke koju korisnik mora nastaviti sa svojim problemom. Izgled tog zaslona je prikazan na slici 22 a isječak programskog koda je prikazan na broju 19.

```
final Uri _url = Uri.parse('mailto:support@billy.com?subject=Support&body=My problem is ');
```

Programski kod 19 Slanje maila za pomoć

4. Flutter paketi

Svi korišteni Flutter paketi su preuzeti s njihove stranice [18.]. Popis svih Flutter paketa korištenih na ovoj aplikaciji te njihove verzije koje su korištene dostupan nam je na isječku programskog koda 20.

```
37    cupertino_icons: ^1.0.2
38    firebase_auth: ^4.3.0
39    cloud_firestore: ^4.1.0
40    firebase_core: ^2.8.0
41    firebase_storage: ^11.1.0
42    firebase_database: ^10.1.0
43    google_sign_in: ^6.0.2
44    animated_splash_screen: ^1.3.0
45    nfc_manager: ^3.2.0
46    slide_to_act: ^2.0.1
47    liquid_pull_to_refresh: ^3.0.1
48    image_picker: ^0.8.7+2
49    file_picker: ^5.2.7
50    uuid: ^3.0.7
51    flutter_staggered_grid_view: ^0.6.2
52    geolocator: ^9.0.2
53    url_launcher: ^6.1.11
```

Programski kod 20 Popis korištenih Flutter paketa

Također jedna od bitnih stavki je i popis dozvola koje aplikacija mora tražiti od korisnika za ispravno funkcioniranje, a taj popis nam je prikazan na isječku programskog koda 21.

```
34    <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
35    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
36    <uses-permission android:name="android.permission.NFC" />
37    <uses-feature android:name="android.hardware.nfc" android:required="true" />
38    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
39    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
40    <uses-permission android:name="android.permission.CAMERA" />
41    <uses-feature android:name="android.hardware.camera" />
42    <uses-feature android:name="android.hardware.camera.autofocus" />
43    <uses-permission android:name="android.permission.INTERNET"/>
```

Programski kod 21 Popis dozvola

5. Zaključak

Naša veza s pametnim telefonima u današnje vrijeme je postala sve čvršća i čvršća. Pogotovo jer pametne telefone koristimo za posao, zabavu, igru te još mnoge druge načine. A svemu tome pridonosi i stalni napredak u tehnologiji i umjetnoj inteligenciji. Nažalost jedna od glavnih tema samita u današnje vrijeme je klimatska promjena. Jedan od uzroka toga je i krčenje šuma radi dobivanja papira koji se koristi u razno raznim primjenama. Trenutno smo uzeli u primjer papir koji se koristi za ispis računa u firmama, trgovinama, bankama, tržnicama, autobusevima, itd. Velika većina tih računa završi ili u smeću ili izgubljeno negdje u stanu. Zbog toga je cilj ovog rada bio izraditi aplikaciju koja može pohraniti digitalno sve korisnikove račune te na taj način utjecati na smanjenje potrošnje papira.

S obzirom na to da je prikazanu aplikaciju samostalno izradio početnik, jasno je da je veza između Fluttera kao alata i Darta kao jezika veoma jaka. Iako je Flutter relativno nov alat koji je dostupan tek nekoliko godina [9.] veoma je lako pronašao svoje mjesto u raznim sektorima razvoja mobilnih aplikacija. Isto tako korišten je i Firebase radi uspostavljanja veze između aplikacije te potrebne nam baze podataka bez koje aplikacija ne bi mogla funkcionirati.

Aplikacija „Billy“ za digitalnu pohranu računa je jednostavna aplikacija za korištenje te može biti unaprijeđena dodavanjem raznih funkcionalnosti. Moguće je dodavanje kartice koja bi služila za prijenos koda za starije osobe koje se malo teže snalaze na mobitelima kada je žurba. Također još jedno poboljšanje bi moglo biti i više korisnika na jednom korisničkom računu tako da cijela obitelj čuva račune na jednom mjestu ali da svako od njih ima svoj kod.

6. Popis slika i programskih kodova

7. Literatura

[1.] Nepoznati autor, „Dart programming language“,
dostupno na: https://dart.dev/ [Pokušaj pristupa 07. svibnja 2023.]

[2.] Nepoznat autor, „Opening Keynote: Dart, a new programming language for structured web programming“,
dostupno na: Presentations -> Opening Keynote: Dart, a new programming language for structured web programming (gotocon.com) [Pokušaj pristupa 07. svibnja 2023.]

[3.] K. Walrath, „Inside a new language for building structured web apps“
dostupno na: http://radar.oreilly.com/2012/03/what-is-dart.html [Pokušaj pristupa 07. svibnja 2023.]

[4.] L. Bak, „A stable SDK for structured web apps“
dostupno na: https://news.dartlang.org/ [Pokušaj pristupa 07. svibnja 2023.]

[5.] A. T. Sandholm, „Annoucing Dart 2: Optimized for Client-Side Development: Dart's Core Tenets“,
dostupno na: https://medium.com/dartlang/announcingdart-2-80ba01f43b6 [Pokušaj pristupa 07. svibnja 2023.]

[6.] Nepoznati autor, „Firebase is Joining Google!“
dostupno na: https://firebase.blog/posts/2014/10/firebase-is-joining-google [Pokušaj pristupa 07. svibnja 2023.]

[7.] R. Amadeo, „Google's Dart language on Android aims for Java-free, 120 FPS apps“,
dostupno na: https://arstechnica.com/gadgets/2015/05/googles-dartlanguage-on-android-aims-for-java-free-120-fps-apps/ [Pokušaj pristupa 07. svibnja 2023.]

[8.] C. Bracken, „v0.0.6: Rev alpha branch version to 0.0.6, flutter 0.0.26“,
dostupno na: https://github.com/flutter/flutter/releases/tag/v0.0.6 [Pokušaj pristupa 07. svibnja 2023.]

[9.] K. Titong, „Speed Up Native Development As Google Flutter Comes Out Of Beta“,
dostupno na: https://appetiser.com.au/blog/speed-up-nativedevelopment-as-google-flutter-comes-out-of-beta/ [Pokušaj pristupa 07. svibnja 2023.]

[10.] T. Sneath, „Technical overview – Flutter“,
dostupno na: https://flutter.dev/docs/resources/technical-overview [Pokušaj pristupa 07. svibnja 2023.]

[11.] Nepoznat autor, „Flutter SDK archive“,
dostupno na: https://docs.flutter.dev/release/archive [Pokušaj pristupa 08. svibnja 2023.]

[12.] Nepoznat autor, „Android Studio“,
Dostupno na: https://developer.android.com/studio [Pokušaj pristupa 08. svibnja 2023.]

[13.] Ducrohet, „Xavier; Norbye, Tor; Chou, Katherine“,
dostupno na: https://android-developers.googleblog.com/2013/05/android-studio-ide-built-for-android.html [Pokušaj pristupa 15. svibnja 2023.]

[14.] Nepoznati autor, „Android Studio“
dostupno na: https://en.wikipedia.org/wiki/Android_Studio#cite_note-10 [Pokušaj pristupa 15. svibnja 2023.]
[15.] Nepoznati autor, „Develop Android apps with Kotlin“
dostupno na: https://developer.android.com/kotlin [Pokušaj pristupa 15. svibnja 2023.]
[16.] Nepoznati autor, „Run apps on the Android Emulator“
dostupno na: https://developer.android.com/studio/run/emulator [Pokušaj pristupa 15. svibnja 2023.]
[17.] Nepoznati autor, „Android(operacijski sustav)“,
Dostupno na: https://hr.wikipedia.org/wiki/Android_(operacijski_sustav) [Pokušaj pristupa 15. svibnja 2023.]
[18.] Nepoznati autor, „Dart packages“
Dostupno na: https://pub.dev/ [Pokušaj pristupa 15. svibnja 2023.]
[19.] Nepoznati autor, „Android Ice Cream Sandwich.“ Dostupno na: https://en.wikipedia.org/wiki/Android_Ice_Cream_Sandwich [Pokušaj pristupa 29. svibnja 2023.]