Mobilna aplikacija za naručivanje i praćenje dostave

Forjan, Krešimir

Undergraduate thesis / Završni rad

2020

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **Josip Juraj Strossmayer University of Osijek, Faculty of Electrical Engineering, Computer Science and Information Technology Osijek / Sveučilište Josipa Jurja Strossmayera u Osijeku, Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:200:607512

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-04-24**

Repository / Repozitorij:

Faculty of Electrical Engineering, Computer Science and Information Technology Osijek

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Sveučilišni preddiplomski studij Računarstva

MOBILNA APLIKACIJA ZA NARUČIVANJE I PRAĆENJE DOSTAVE

Završni rad

Krešimir Forjan

Osijek, 2020.

SADRŽAJ

1. UVOD 1

1.1. Zadatak završnog rada 1

2. PREGLED POSTOJEĆIH RJEŠENJA 2

2.1. Glovo – Naruči bilo što. Dostava hrane i ostalo 2

2.2. Wolt: Dostava hrane 3

3. OPIS KORIŠTENIH TEHNOLOGIJA 4

3.1. Dart 4

3.2. Flutter 4

3.3. Firebase 5

3.4. Visual Studio Code 6

4. RAZVOJ APLIKACIJE 8

4.1. Početni zaslon 8

4.2. Zaslon za stvaranje narudžbe 9

4.3. Zaslon za praćenje dostave 11

4.4. Zaslon za prikaz postojećih narudžbi 13

4.5. Zaslon s informacijama pojedine narudžbe 15

5. KORIŠTENJE APLIKACIJE 17

5.1. Način rada za naručitelja 18

5.2. Način rada za dostavljača 19

6. ZAKLJUČAK 22

LITERATURA 23

SAŽETAK 25

ABSTRACT 26

ŽIVOTOPIS 27

PRILOZI..... 28

1. UVOD

Kupovina unutar aplikacija postala je dio naše svakodnevnice. Takav način prodaje povećava potrebu za praćenjem lokacije pošiljke, a korisnicima je najpoželjnije praćenje u stvarnom vremenu zbog točnog uvida u stvarno stanje.

U ovom radu prikazuje se proces izrade mobilne aplikacije za naručivanje proizvoda i praćenje njihove dostave u stvarnom vremenu. Aplikacija će omogućiti dostavljaču uvid u lokaciju naručitelja, a naručitelj će imati mogućnost praćenja pošiljke.

Za razvoj aplikacije korištene su tehnologije: programski jezik Dart, set razvojnih alata za programsku podršku Flutter, platforma za razvoj aplikacija Firebase i uređivač koda Visual Studio Code.

Aplikacija se sastoji od dva dijela, dio za kupce i dio za prodavače. Mogućnosti koje aplikacija pruža kupcima su registracija korisnika pomoću elektroničke pošte i lozinke te odabir i praćenje željene pošiljke. Prodavač time ima uvid u sve zatražene pošiljke i njihovu lokaciju.

U drugom poglavlju bit će navedena i prikazana postojeća rješenja na servisu Google Play. Treće poglavlje opisuje korištene tehnologije koje su korištene za razvoj aplikacije. U četvrtom poglavlju je detaljno opisan postupak razvoja aplikacije. Peto poglavlje daje upute o načinu korištenja aplikacije i njenu svrhu. U posljednjem poglavlju je zaključak.

1.1. Zadatak završnog rada

Potrebno je napraviti mobilnu aplikaciju za naručivanje proizvoda i praćenje njihove dostave u realnom vremenu. Potrebno je slati obavijesti korisniku o prelasku iz jednog stadija u drugi i dati mu prikaza trenutne lokacije dostavljača.

2. PREGLED POSTOJEĆIH RJEŠENJA

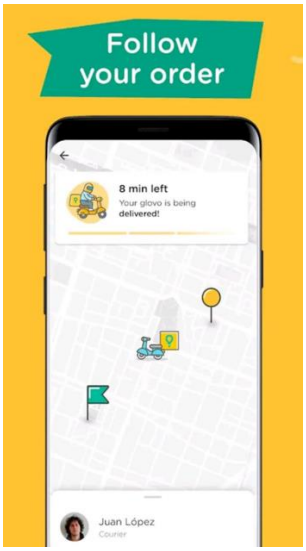
Postoje aplikacije koje omogućuju korisnicima praćenje pošiljke u stvarnom vremenu. Takve aplikacije pokazale su se kao vrlo uspješni projekti u posljednjih nekoliko godina. Neke od najuspješnijih aplikacija na servisu Google Play bit će prikazane u nastavku ovog poglavlja.

2.1. Glovo – Naruči bilo što. Dostava hrane i ostalo

Glovo je španjolska tehnološka kompanija pokrenuta 2015. godine [1]. Aplikacija Glovo omogućuje kupovinu, preuzimanje i slanje bilo kojeg proizvoda unutar jednog grada. Aplikacija pruža mogućnost odabira proizvoda iz široke ponude i obavještava dostavljače o zatraženoj narudžbi. Kupac ima mogućnost praćenja lokacije pošiljke u stvarnome vremenu, a dostavljač ima pristup lokaciji kupca. To znatno olakšava dostavu pošiljke dostavljaču, a kupcu štedi vrijeme koje bi potrošio na potraživanje i kupovinu proizvoda. Izbornik ponude koju nudi Glovo prikazan je na slici 2.1.

Slika 2.1. Početna stranica aplikacije Glovo – Naruči bilo što. Dostava hrane i ostalo

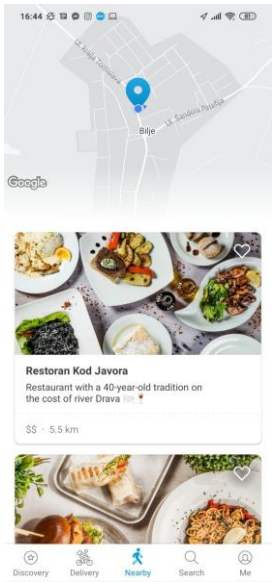
Usluga praćenja lokacije u stvarnom vremenu omogućena je zahvaljujući GPS-u i mobilnim podatcima. GPS je kratica globalni položajni sustav (engl. *Global positioning system*). GPS je mreža satelita pomoću kojih je omogućeno precizno određivanje položaja na Zemlji. [2] Slika 2.2. sadrži prikaz lokacije u stvarnom vremenu unutar aplikacije Glovo.



Slika 2.2. Praćenje lokacije unutar aplikacije Glovo – Naruči bilo što. Dostava hrane i ostalo

2.2. Wolt: Dostava hrane

Wolt je finska tehnološka kompanija koja dostavlja hranu iz restorana putem mobilne i internetske platforme. To je bio prvi međunarodni servis s mogućnošću praćenja narudžbe na ovom području [3]. Wolt pruža vrlo slične usluge kao već spomenuti Glovo, ali je njegovo područje ograničeno na proizvode iz restorana.



Slika 2.3. Izgled aplikacije Wolt: Dostava hrane

3. OPIS KORIŠTENIH TEHNOLOGIJA

U ovom poglavlju navedene su i opisane tehnologije koje su korištene za izradu aplikacije. Prvo je opisan korišteni programski jezik, zatim korišteni razvojni alat za izradu mobilnih aplikacija, baza podataka te korišteni uređivač koda.

3.1. Dart

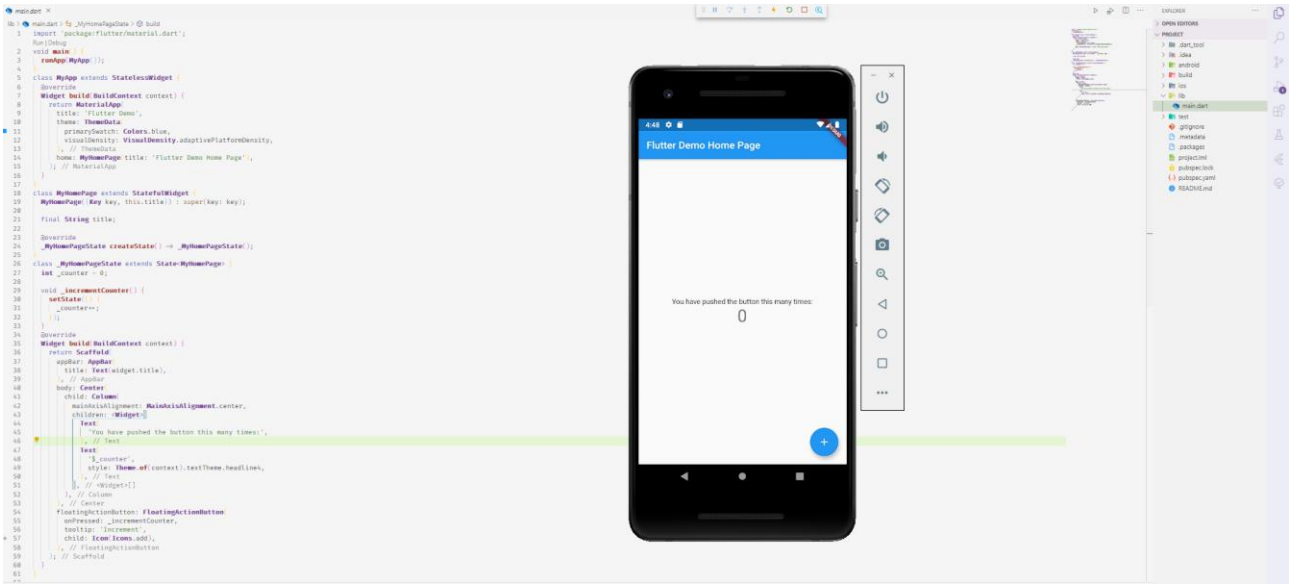
Za razvoj aplikacije korišten je programski jezik Dart. Dart je objektno orijentirani programski jezik sa sintaksom sličnom C-u [4]. Razvija ga Google te omogućuje razvoj mobilnih, računalnih i mrežnih aplikacija. Dart kod može biti preveden u strojni kod ili JavaScript, ovisno o primjeni [5]. Dart je programski jezik koji se koristi za razvoj aplikacija pomoću razvojnog alata za izradu mobilnih aplikacija Flutter.

```
Run | Debug
void main() {
  for (int i = 0; i < 5; i++) {
    print('hello ${i + 1}');
  }
}
```

Slika 3.1. Dart kod

3.2. Flutter

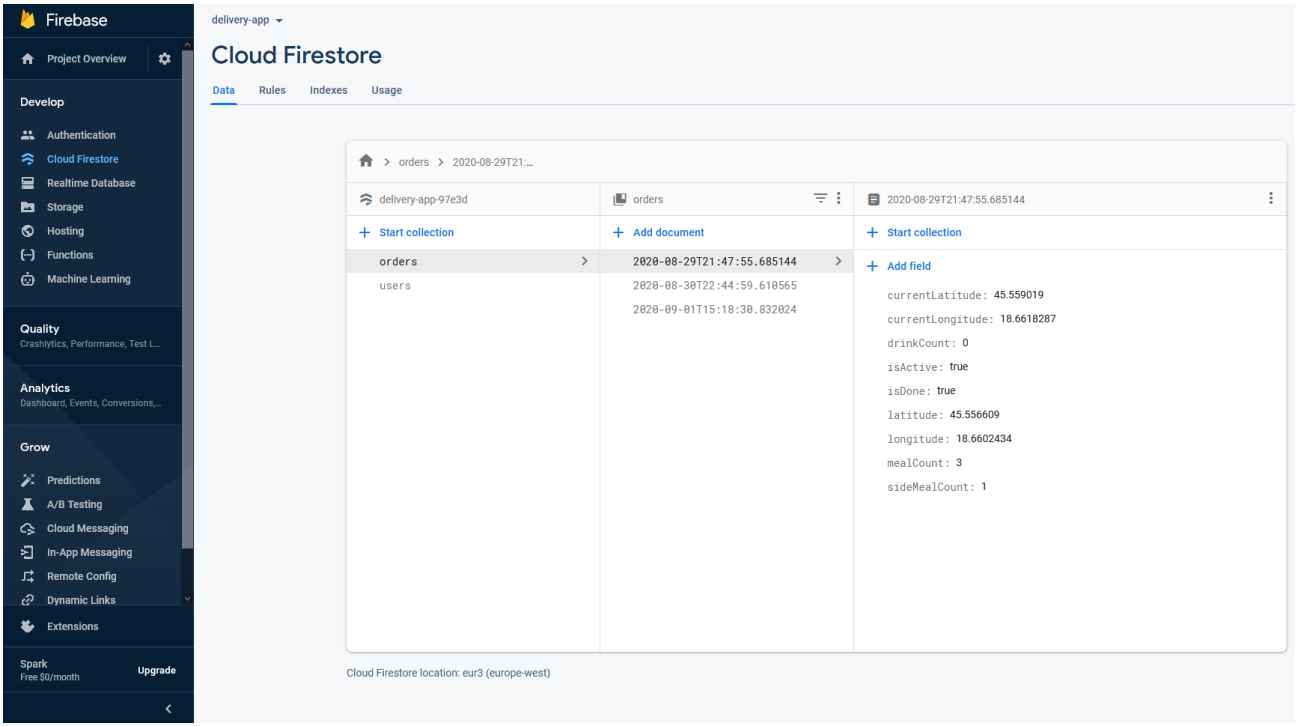
Flutter je razvojni alat otvorenog koda koji je razvija Google od 2017 godine. Služi za razvoj aplikacija na raznim platformama kao što su: Android, iOS, Linux, Windows i web stranice [6]. Aplikacije izrađene u Flutteru su višeplatformske te se s jednim izvornim kodom mogu pokretati na različitim platformama. Prva verzija, poznata kao „*Sky*“, predstavljena je prvi puta 2015. godine s ciljem da pokreće mobilne aplikacije s konstantnih 120 slika po sekundi [7]. 4. prosinca 2018. objavljen je Flutter 1.0 kao prva stabilna verzija [8]. Flutter je vrlo efikasan jer korisnik može vidjeti promjene u izgledu aplikacije u stvarnome vremenu. To je moguće zahvaljujući dinamičkom prevođenju koje Flutter podržava. Rad u Flutteru je dodatno olakšan brojnim bibliotekama koje pruža razvojni alat. Biblioteke sadrže velik broj gotovih funkcija za prikaz korisničkog sučelja koje se vrlo lako koriste. Slika 3.2. prikazuje početni projekt stvoren razvojnim alatom Flutter.



Slika 3.2. Početni Flutter projekt

3.3. Firebase

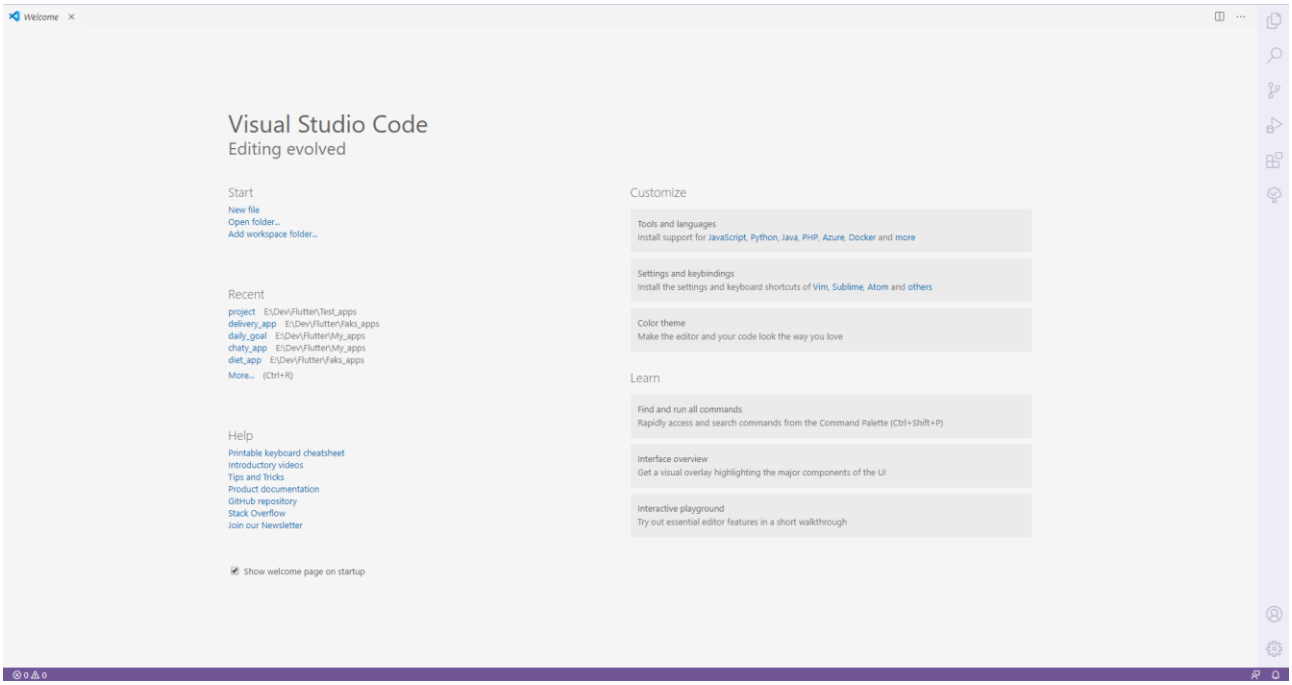
Firebase je platforma za stvaranje mobilnih i web aplikacija koju je Google 2014. godine otkupio od *start-up* tvrtke Envolve [9]. Firebase nudi razne usluge kao što su baza podataka u stvarnome vremenu, autentifikacija korisnika, analiza korisničkih podataka i obrada korisničkih podataka. Većina usluga se naplaćuje samo u slučaju da broj zahtjeva prema Firebaseu pređe granicu dozvoljenih zahtjeva koju određuje Google. Ovakav model baze podataka je vrlo povoljan korisnicima jer mogu imati bazu podataka i obrađivati podatke bez fizičkog servera. Slika 3.3. prikazuje Firebase konzolu pomoću koje korisnik upravlja bazom podataka.



Slika 3.3. Prikaz Firebase konzole

3.4. Visual Studio Code

Visual Studio Code je besplatni uređivač koda za Windows, Linux i macOS kojeg je razvio Microsoft. [10] 2019. godine Visual Studio Code pokazao se kao najpopularniji uređivač koda u anketi web stranice StackOverflow. [11] Visual Studio Code korisnici mogu vrlo lako prilagoditi svojim potrebama koristeći razne dodatke koje razvijaju drugi korisnici i Microsoft. Visual Studio Code sadrži mnogo alata kao što su alati za otklanjanje pogrešaka, alati za refaktoriranje koda i alati za automatsko dovršavanje sintakse. Visual Studio Code prikazan je na slici 3.4.



Slika 3.4. Visual Studio Code

4. RAZVOJ APLIKACIJE

U ovom poglavlju opisan je postupak razvoja aplikacije. Potrebno je stvoriti projekt kojeg je moguće nadograđivati novim funkcionalnostima. Flutter projekt stvara se naredbom *„flutter create naziv_projekta“*. Navedena naredba stvara projekt sa svim datotekama potrebnim za razvoj mobilne aplikacije, u direktoriju u kojem se korisnik trenutno nalazi. Aplikacija se sastoji od pet zaslona. Prvo je prikazan postupak razvoja zaslona zaduženih za stvaranje narudžbe, a zatim zaslona zaduženih za dostavu narudžbe.

4.1. Početni zaslon

Početni zaslon je prvi zaslon koji se prikazuje pri pokretanju aplikacije. Ako je korisnik u prijašnjoj sesiji obavio proces odabira vrste korisnika, početni zaslon se ne prikazuje, već se prikazuje zaslon koji odgovara vrsti korisnika. Kod koji omogućuje ovakvo ponašanje prikazan je na slici 4.1. Kod na slici sadrži dvije funkcije „*StreamBuilder*“ koje mijenjaju prikaz ovisno o postojanju korisnika u bazi podataka. Navedene podatke dohvaća pomoću objekata „*Stream*“ koji pružaju stanje s baze podataka u stvarnome vremenu. Funkcijom „*InkWell*“ na početnom zaslonu su prikazane dvije tipke za odabir željene funkcionalnosti. Pritiskom na jednu od tipki podatci o odabiru šalju se u bazu podataka te se pri svakom novom pokretanju aplikacije učitavaju zbog provjere postojeće sesije. Podatak o sesiji sprema se u obliku tipa podatka koji ima dva stanja: „*deliver*“ ili „*order*“.

```dart
return StreamBuilder(
  stream: _auth.onAuthChanged,
  builder: (context, userSnapshot) {
    if (userSnapshot.connectionState == ConnectionState.active) {
      User _user = userSnapshot.data;

      if (_user == null) {
        return RoleSelectionPage.create(context);
      } else {
        return StreamBuilder(
          stream: _databse.getUserTypeStream(_user.userId),
          builder: (context, typeSnapshot) {
            if (typeSnapshot.data == UserType.deliver) {
              return DeliverPage();
            } else if (typeSnapshot.data == UserType.order) {
              return OrderPage();
            } else {
              return _buildLoadingScreen(context);
            }
          },
        ); // StreamBuilder
      }
    } else {
      return _buildLoadingScreen(context);
    }
  },
); // StreamBuilder
```

Slika 4.1. Dart kod za provjeru postojeće sesije

4.2. Zaslon za stvaranje narudžbe

Zaslon za stvaranje narudžbe je prvi zaslon koji će se prikazati ako korisnik na početnom zaslonu odabere da aplikaciju želi koristiti za naručivanje hrane. Ovaj zaslon zadužen je za slanje podataka o narudžbi u bazu podataka. Tipka za potvrdu narudžbe postaje dostupna tek kada korisnik unese sve podatke koji su potrebni za uspješnu dostavu zbog provjere ispravnosti unosa koja je prikazana na slici 4.4. Za lociranje korisnika aplikacija koristi razvojni paket „*Geolocator*“ koji pomoću satelitskog radionavigacijskog sustava pruža trenutnu lokaciju telefona. Klasa „*Geolocator*“ sadržana u istoimenom razvojnom paketu pruža funkciju „*getCurrentPosition*“ koja ima povratni tip „*Position*“ s koordinatama uređaja. Slika 4.2. prikazuje kod zadužen za lociranje korisnika, a slika 4.3. kod zadužen za unos veličine narudžbe. Za unos veličine narudžbe koristi se razvojni paket „*Numberpicker*“ koji na zaslonu prikazuje stupac s ponuđenim vrijednostima cijelih brojeva. Promjenom vrijednosti na izborniku vrijednosti podatci se spremaju u unutarnje stanje aplikacije. S unutarnjeg stanja aplikacije podatci se šalju u bazu podataka posebnom funkcijom koju pruža Firebase

razvojni paket. Funkcija za slanje podataka u bazu podataka je „*setState*“ i kao ulazni podatak prima objekt tipa „*Map*“. Ugrađenim alatima „*Map*“ se pretvara u JSON format te se sprema u bazi podataka. JSON format je standardni format za prenošenje podataka na principu povezivanja ključa s prikladnim podatkom. Navedena funkcija Firebase razvojnog paketa prikazana je na slici 4.5.

```
Future<void> updateLocation() async {
  Position position = await Geolocator()
      .getCurrentPosition(desiredAccuracy: LocationAccuracy.high);
  _updateWith(position: position);
  toggleLocatedStatus();
}
```

Slika 4.2. Dart kod za lociranje korisnika

```
NumberPicker.integer(
  infiniteLoop: true,
  listViewWidth: mediaQuery.size.width * 0.15,
  initialValue: _currentValue,
  minValue: 0,
  maxValue: 10,
  onChanged: (value) {
    setState(() {
      _currentValue = value;
      widget.changeHandler(value);
    });
  },
), // NumberPicker.integer
```

Slika 4.3. Dart kod za unos

```
bool get isOrderable {
  return located == true &&
      (mealCount != 0 || sideMealCount != 0 || drinkCount != 0);
}
```

Slika 4.4. Dart kod za provjeru valjanosti unosa

```dart
class FirestoreDatabase implements Database {
  @override
  Future<void> addOrder(Order order) async {
    final instance = Firestore.instance.document(ApiPath.order(order.id));
    instance.setData(order.toMap());
  }
```

Slika 4.5. Dart kod za slanje podataka o narudžbi na bazu podataka

4.3. Zaslon za praćenje dostave

Nakon što korisnik potvrdi željenu narudžbu prikazuje se zaslon za praćenje dostave. Praćenje lokacije u stvarnom vremenu omogućuje funkcija „*StreamBuilder*“ koja mijenja prikaz u aplikaciji svaki puta kada dođe do promijene u bazi podataka koja je prouzročena promjenom lokacije dostavljača. U funkciju „*StramBuilder*“ ugniježđene su ostale funkcije za prikaz korisničkog sučelja koje se pozivaju svaki puta kada je potrebno izmijeniti prikaz aplikacije. Funkcija „*StreamBuilder*“ prikazana je na slici 4.6. Na slici 4.6. su također prikazane funkcije za prikaz korisničkog sučelja kao što su funkcija „*Column*“ za prikaz elemenata korisničkog sučelja u stupcima i funkcija „*Contaienr*“ koja služi kao spremnik za druge elemente korisničkog sučelja s opcionalnim svojstvima koja utječu na elemente unutar spremnika. Funkcija koja pruža podatke iz baze podataka u stvarnom vremenu stvara objekt tipa „*Stream*“ koji je potreban funkciji „*StreamBuilder*“ da pravilno obnavlja prikaz aplikacije. Ta funkcija prikazana je na slici 4.7. Za prikaz karte korišten je paket „*flutter_map*“ čija je svrha prikaz lokacije na karti pomoću danih koordinata. U slučaju da je stanje narudžbe neaktivno ili je narudžba završila, zaslon za praćenje dostave mijenja svoj izgled kako bi se prilagodio trenutnom stanju narudžbe. Završetkom dostave aplikacija prima podatak o uspješnoj dostavi narudžbe te nudi naručitelju tipku za završetak procesa koja će poslati zahtjev za brisanje podataka o narudžbi iz baze podataka.

```
child: StreamBuilder(
  stream: Provider.of<Database>(context, listen: false)
      .getOrder(this.orderId),
  builder: (cotnext, snapshot) {
    if (snapshot.connectionState == ConnectionState.active &&
        snapshot.data != null) {
      Order order = snapshot.data;
      return Column(
        children: [
          (order.isActive &&
                  order.currentLongitude != null &&
                  order.currentLatitude != null)
              ? SafeArea(
                  child: Column(
                    children: [
                      SizedBox(
                        height: 20,
                      ), // SizedBox
                      Container(
                        width: double.infinity,
                        height: 400,
                        child: _buildMap(···
                      ), // Container
                      SizedBox(  // SizedBox
                      Text(  // Text
                    ],
                  ), // Column
                ) // SafeArea
              : _buildPlaceHolder(context, order),
        ],
      ); // Column
    } else {···
    }
  },
), // StreamBuilder
```

Slika 4.6. Dart kod funkcije „StreamBuilder“

```
Stream<Order> getOrder(String orderId) {
  final orderSnapshot = Firestore.instance
      .collection(ApiPath.orders())
      .document(orderId)
      .snapshots();
  return orderSnapshot
      .map((event) => Order.fromMap(event.data, event.documentID));
}
```

Slika 4.7. Dart kod za dohvaćanje podataka o narudžbi iz baze podataka

```
Widget _buildMap(BuildContext context, Position position) {
  return FlutterMap(
    options: MapOptions(
      center: LatLng(position.latitude, position.longitude),
      zoom: 13.5,
    ), // MapOptions
    layers: [ ···
  ); // FlutterMap
}
}
```

Slika 4.8. Dart kod za prikaz karte

4.4. Zaslon za prikaz postojećih narudžbi

U slučaju da korisnik na početnom zaslonu odabere da aplikaciju želi koristiti kao dostavljač, prikazat će se zaslon za prikaz postojećih narudžbi. Nakon što korisniku prikaže ovaj zaslon, počinje kontinuirano praćenje lokacije dostavljača unutar aplikacije. Postojeće narudžbe s baze podataka se dohvaćaju na sličan način kao i lokacija u slučaju praćenja lokacije. Funkcija za dohvaćanje podataka stvara objekt tipa „*Stream*“ koji pri svakoj promjeni podataka obavijesti „*StreamBuilder*“ da je potrebno promijeniti prikaz aplikacije. Lista trenutno dostupnih narudžbi prikazana je funkcijom „*ListView.builder*“, koja je vrlo pogodna za prikaz lista koje nemaju poznatu duljinu, jer u memoriji zadržava samo one elemente koji su trenutno vidljivi na zaslonu. Element liste dostupnih narudžbi je tipka koja otvara zaslon s detaljima o narudžbi, a svaka tipka sadržava informaciju o udaljenosti narudžbe od trenutne lokacije dostavljača. Udaljenost dostavljača i lokacije narudžbe računa se iz pozicija prikazanih u geografskoj dužini i širini korištenjem funkcija razvojnog paketa „*Geodesy*“. Slika 4.9. prikazuje funkciju za dohvaćanje podataka o postojećim narudžbama u stvarnom vremenu. Slika 4.10. prikazuje funkciju za stvaranje tipke odabira narudžbe sa svim podatcima o pojedinoj narudžbi. Funkcija „*Navigator.push*“ sa slike 4.10. služi za promjenu trenutnog prikaza aplikacije te se njenim pozivom stvara prikaz koji joj je predan kao argument. Na slici 4.11. prikazan je kod funkcije „*ListView.builder*“ za stvaranje liste narudžbi. „*ListView.builder*“ sa slike iscrtava elemente korisničkog sučelja za svaki od elemenata liste „*orders*“.

```dart
Stream<List<Order>> getOrdersStream() {
  final ordersSnapshot =
      Firestore.instance.collection(ApiPath.orders()).snapshots();
  return ordersSnapshot.map(
    (event) => event.documents
        .map((e) => Order.fromMap(e.data, e.documentID))
        .toList(),
  );
}
```

Slika 4.9. Dart kod za dohvaćanje postojećih narudžbi iz baze podataka

```dart
ListTile(
  leading: Icon(
    Icons.location_on,
    color: getActiveColor(theme),
  ), // Icon
  title: Text( // Text
  onTap: () {
    Navigator.push(
      context,
      MaterialPageRoute(
        builder: (context) => ConfirmDeliveryScreen(
          model: model,
          order: order,
        ), // ConfirmDeliveryScreen
      ), // MaterialPageRoute
    );
  },
  subtitle: Text(
    'Main meals: ${order.mealCount} Side meals: ${order.sideMealCount} Drinks: ${order.drinkCount}',
    style: TextStyle(color: theme.textTheme.bodyText2.color),
    overflow: TextOverflow.fade,
  ), // Text
  trailing: Icon(
    Icons.arrow_forward_ios,
    color: theme.canvasColor,
  ), // Icon
), // ListTile
```

Slika 4.10. Dart kod za stvaranje tipke odabira narudžbe

```dart
Widget _buildListView(List<Order> orders) {
  return RefreshIndicator(
    onRefresh: () async {
      setState(() {});
    },
    child: ListView.builder(
      itemCount: orders.length,
      itemBuilder: (BuildContext context, int index) {
        return OrderTile(
          model: widget.model,
          order: orders[index],
        ); // OrderTile
      },
    ), // ListView.builder
```

Slika 4.11. Dart kod koji stvara „ListView.builder“

4.5. Zaslon s informacijama pojedine narudžbe

Odabirom aktivne narudžbe iz liste narudžbi prikazuje se zaslon s informacija o odabranoj narudžbi. Uz prikaz broja naručenih namirnica, ovaj zaslon sadrži tipku za aktivaciju narudžbe ili tipku za završetak dostave, ovisno o trenutnom stanju narudžbe. Aktivnost narudžbe zapisuje se u unutarnje stanje aplikacije što onemogućuje daljnju aktivaciju novih narudžbi. U slučaju da postoji već aktivna narudžba na zaslonu će biti ispisana obavijest korisniku da već postoji aktivna narudžba. Slika 4.13. prikazuje kod za ispisivanje obavijesti o već postojećoj aktivnosti narudžbe. Ako korisnik aktivira neku od narudžbi, boja teksta aktivirane narudžbe bit će promijenjena na zaslonu za prikaz postojećih narudžbi. Slika 4.12. prikazuje kod koji mijenja funkcionalnost tipke ovisno o stanju narudžbe. U slučaju da je narudžba aktivna bit će prikazana tipka za poziv funkcije „*finishDelivery*“ koja u bazu podataka šalje informaciju o završetku dostave i funkcija „*Navigator.pop*“ koja korisnika vraća na početni zaslon, a u slučaju da narudžba nije aktivna bit će prikazana tipka za poziv funkcija „*updateDeliveryStatus*“ i „*updateCurrentOrderLocation*“ koje u bazu podataka šalju informaciju o početku dostave te trenutnu lokaciju uređaja.

```
return widget.order.isActive
    ? RaisedButton(
        color: Theme.of(context).hintColor,
        child: Text( // Text
        onPressed: () async {
          await Provider.of<Database>(context, listen: false)
              .finishDelivery(widget.order);
          widget.model.toggleIsActiveDelivery();
          widget.model.updateOrder(null);
          Navigator.pop(context);
        },
      ) // RaisedButton
    : RaisedButton(
        color: Theme.of(context).primaryColor,
        child: Text( // Text
        onPressed: () async {
          setState(() {
            widget.order.isActive = true;
            widget.model.updateOrder(widget.order);
            widget.model.toggleIsActiveDelivery();
          });
          await Provider.of<Database>(context, listen: false)
              .updateDeliveryStatus(widget.order);
          await Provider.of<Database>(context, listen: false)
              .updateCurrentOrderLocation(
            widget.order,
            widget.model.position,
          );
        },
      ); // RaisedButton
```

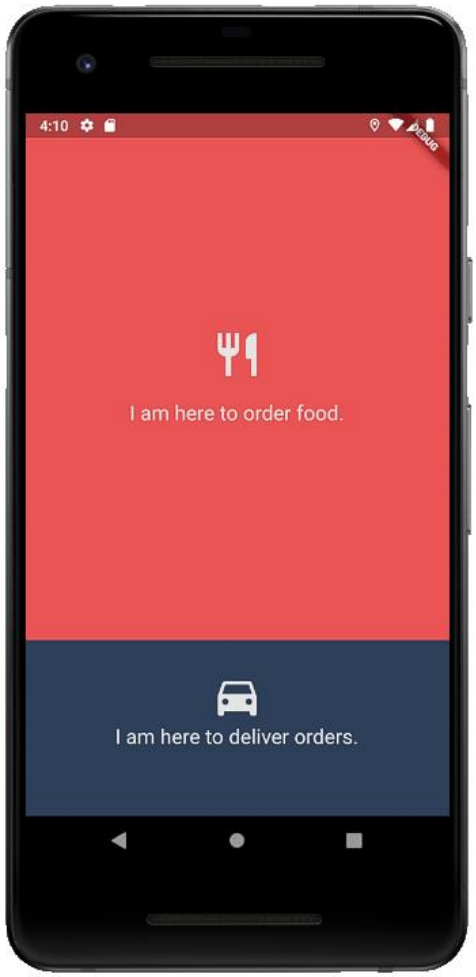
Slika 4.12. Dart kod za prikaz tipke ovisno o stanju narudžbe

```
if (widget.model.order != null) {
  if (widget.model.order.id != widget.order.id &&
      widget.model.isActiveDelivery) {
    return Text('You alredy started another delivery.');
  }
}
```

Slika 4.13. Dart kod za ispis obavijesti o aktivnoj narudžbi

5. KORIŠTENJE APLIKACIJE

Kod prvog pokretanja aplikacije neće postojati informacija o prijašnjoj sesiji pa će korisnik imati izbor između dva načina rada: naručitelj ili dostavljač. Odabir se vrši pritiskom na jednu od dvije ponuđene tipke. Izbornik načina rada prikazan je na slici 5.1.

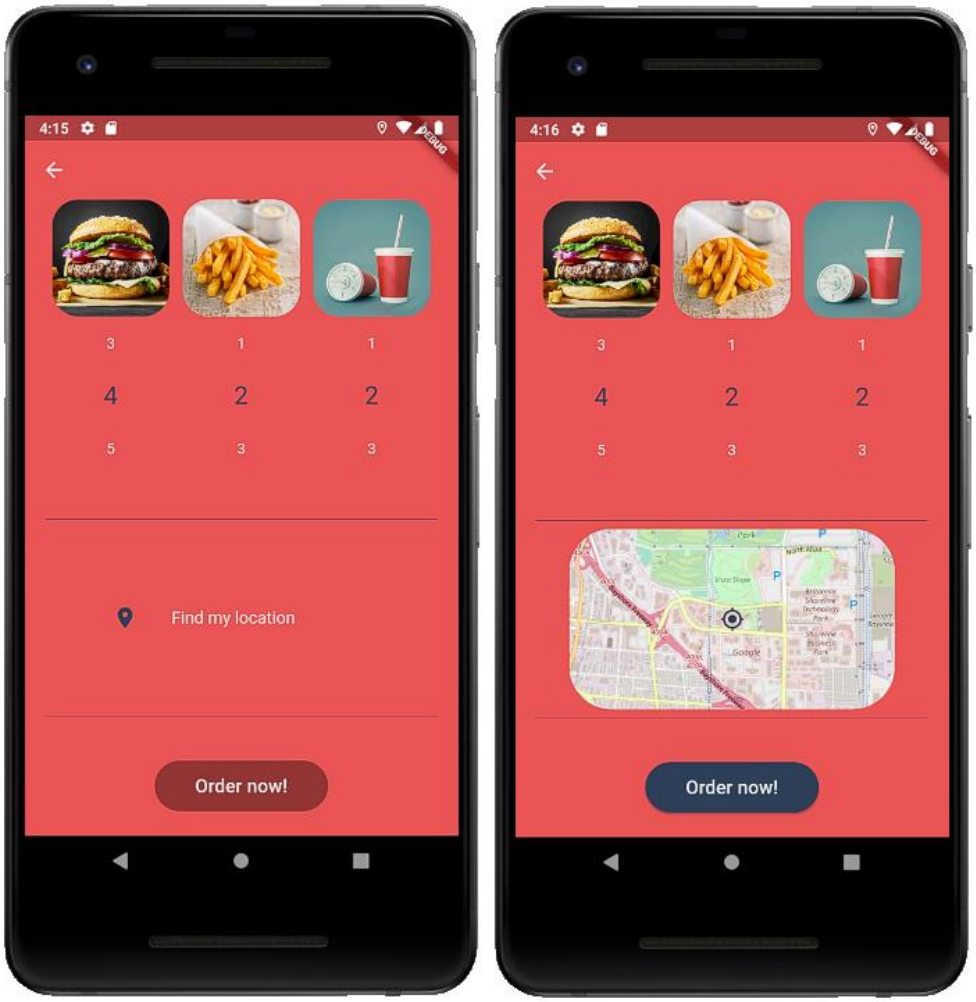


Slika 5.1. Prikaz izbornika načina rada

Odabirom jednog od ponuđenih načina rada aplikacija preusmjerava korisnika na novi zaslon prikladan njegovu odabiru. Izbor korisnika sprema se u bazu podataka pa će pri svakom idućem korištenju aplikacija automatski pokrenuti način rada koji je korisnik odabrao u prošloj sesiji.

5.1. Način rada za naručitelja

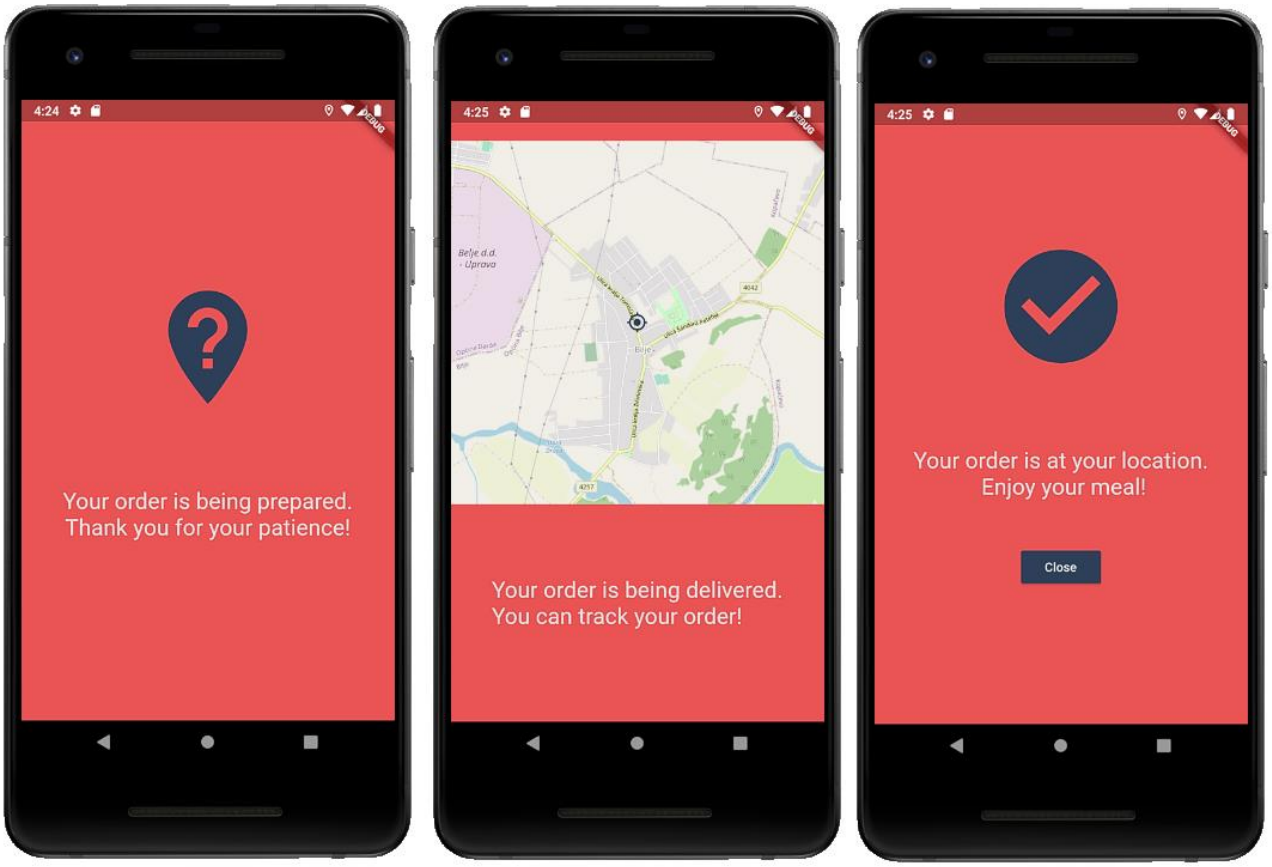
Ako korisnik odabere način rada za naručitelja, bit će preusmjeren na zaslon za naručivanje. Narudžba za naručivanje sadrži ponudu hrane koja može biti naručena, polje za unos količine željene hrane i tipku za pronalazak lokacije korisnika. Kako bi narudžba mogla biti ostvarena potrebno je postaviti količinu barem jednog proizvoda na više od jedan i pronaći lokaciju korisnika pritiskom na tipku „*Find my location*“. Slika 5.2. prikazuje zaslon za naručivanje prije i poslije pronalaska lokacije korisnika. U slučaju da je korisnik uspješno lociran, na zaslonu će se prikazati karta s njegovom trenutnom lokacijom.



Slika 5.2. Prikaz zaslona za naručivanje prije i nakon lociranja korisnika

Kada korisnik započne narudžbu pritiskom na tipku „*Order now!*“ podatci o narudžbi se prikazuju svim aktivnim dostavljačima, a korisnik je preusmjeren na zaslon za čekanje. Nakon što

dostavljač aktivira narudžbu i time potvrdi da je narudžba spremna za dostavu, korisniku se prikazuje karta na kojoj može pratiti trenutnu lokaciju dostavljača u stvarnom vremenu. Svako kretanje dostavljača vidljivo je na karti do završetka dostave. Pri završetku dostave, korisnik je obaviješten o uspješnosti dostave i ima mogućnost zatvaranja narudžbe. Glavna tri stadija narudžbe, priprema, dostava i kraj, prikazani su na slici 5.3.

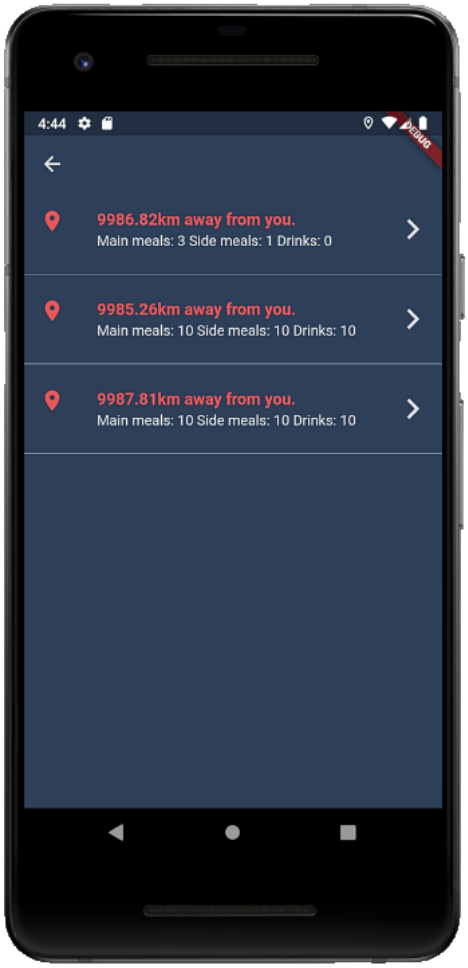


Slika 5.3. Prikaz zaslona za praćenje narudžbe

5.2. Način rada za dostavljača

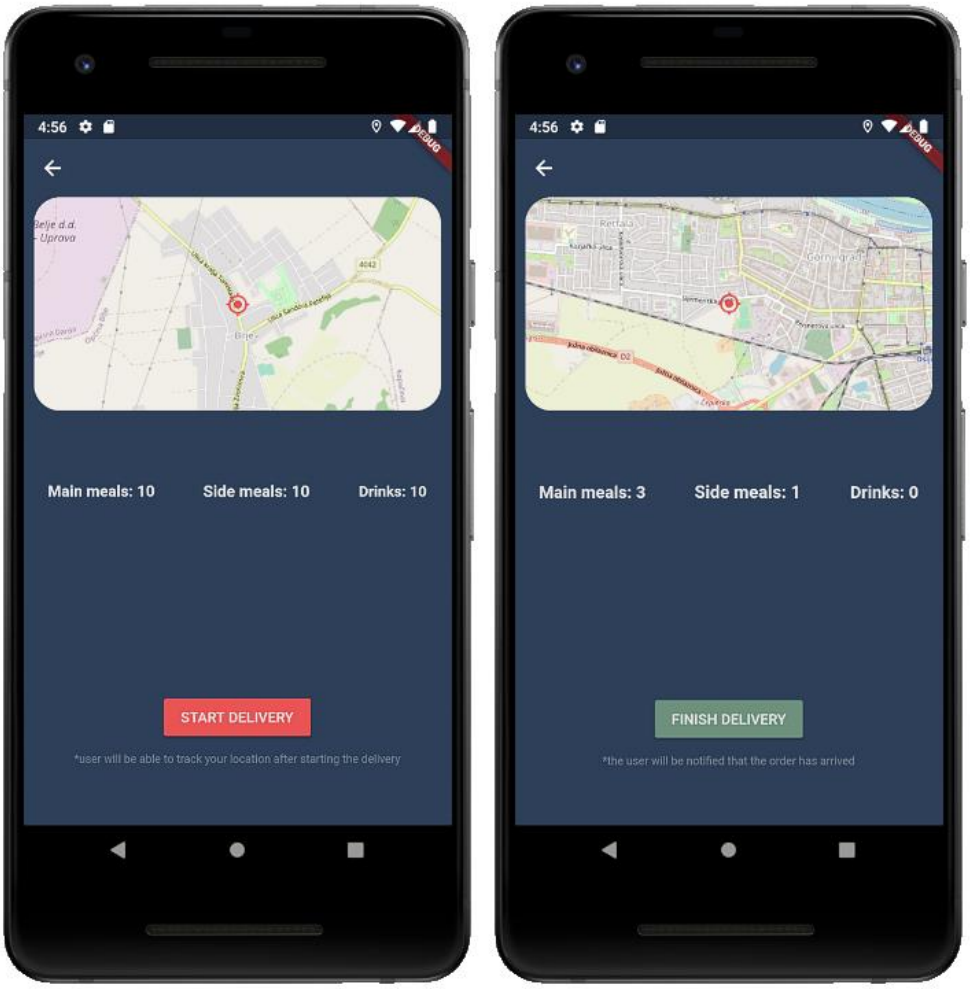
U slučaju da je korisnik odabrao način rada za dostavljača, bit će preusmjeren na zaslon s popisom svih narudžbi koje u tom trenutku čekaju dostavu. Svaka narudžba uz sebe ima trenutnu udaljenost od dostavljača izraženu u kilometrima i veličinu narudžbe koju je korisnik zatražio. Ako ne postoji niti jedna narudžba koja čeka dostavu, dostavljaču je ispisana obavijest o nepostojanju narudžbi. Iako se lista ažurira automatski u slučaju pojavljivanja novih narudžbi, listu je moguće

osvježiti povlačenjem članova liste prema dolje. Izgled zaslona s popisom narudžbi vidljiv je na slici 5.4.



Slika 5.4. Prikaz zaslona s trenutno dostupnim narudžbama

Odabirom neke od dostupnih narudžbi prikazuje se zaslon sa slike 5.5. Na tom zaslonu dostavljač ima pregled lokacije na koju je potrebno dostaviti narudžbu. Narudžbu je moguće aktivirati pritiskom na tipku „*START DELIVERY*“. Nakon aktivacije započinje dijeljenje lokacije s korisnikom koji je stvorio narudžbu. Dostavljaču se nakon aktivacije omogućuje funkcija „*FINISH DELIVERY*“ koja završava dostavu i zaustavlja dijeljenje lokacije u stvarnom vremenu.



Slika 5.5. Prikaz zaslona s informacijama o narudžbi

Dostavljač je automatski preusmjeren na zaslon za prikaz postojećih narudžbi pri završetku narudžbe i ima mogućnost aktiviranja nove narudžbe. U slučaju da nema više niti jedna dostupna narudžba, dostavljaču će biti ispisana obavijest o nepostojanju narudžbi.

6. ZAKLJUČAK

Aplikacije s mogućnošću praćenja lokacije u stvarnom vremenu su sve češća pojava u ljudskim životima. Aplikacije za transport putnika, dostavu hrane, dostavu paketa i mnoge druge oslanjaju se na GPS usluge pametnih telefona zbog jednostavnosti pronalaska željene lokacije. Korisnicima je iskustvo korištenja aplikacije znatno poboljšano jer mogu dobiti konkretnu informaciju o lokaciji svoje narudžbe ili prijevoza. Zbog toga je cilj ovog rada bio izraditi aplikaciju koja će imati mogućnost lociranja korisnika i pružati im mogućnost praćenja promjene lokacije.

Aplikacije je izrađena u razvojnom alatu za mobilne aplikacije Flutter. Flutter znatno olakšava razvoj aplikacija jer se s jednim izvornim kodom aplikacija može pokrenuti na različitim platformama kao što su Android i iOS. Kao platforma za spremanje podataka korišten je Firebase. Firebase olakšava razvoj aplikacija zbog jednostavnog pristupa mrežnoj bazi podataka bez fizičkog servera.

Aplikacija može biti još unaprijeđena dodavanjem raznih funkcionalnosti. Moguće je dodavanje većeg izbora hrane ili namirnica u izborniku za narudžbu kako bi zahtjevi korisnika bili što lakše ispunjeni. Još jedno od mogućih poboljšanja je dodavanje korisničkih računa i povijesti narudžbi. To bi omogućilo korisnicima sinkroniziranje podataka na više uređaja i uvid u prijašnje narudžbe. Postoji i mogućnost poboljšanja vizualnih dijelova aplikacije. Neka vizualna poboljšanja su dodavanje novih animacija i mogućnost odabira tema.

LITERATURA

[1] »Glovo,« [Mrežno]. Available: https://about.glovoapp.com/en. [Pokušaj pristupa 3 rujan 2020.].

[2] M. L. D. T. Miljenko Lapine, »Kartografija,« [Mrežno]. Available: http://www.kartografija.hr/old_hkd/obrazovanje/prirucnici/gpspoc/gpspoc.htm. [Pokušaj pristupa 3 rujan 2020.].

[3] »Jabuka.tv,« 29 listopad 2019.. [Mrežno]. Available: https://www.jabuka.tv/dvije-globalne-aplikacije-za-dostavu-hrane-digle-su-prodaju-nekim-restoranima-u-zagrebu-za-100/. [Pokušaj pristupa 3 rujan 2020.].

[4] Dart, »Important concepts,« [Mrežno]. Available: https://dart.dev/guides/language/language-tour#important-concepts. [Pokušaj pristupa 2. rujan 2020.].

[5] Dart, »The Dart type system,« [Mrežno]. Available: https://dart.dev/guides/language/type-system. [Pokušaj pristupa 2. rujan 2020.].

[6] M. Patel, »Concettolabs,« 29 travanj 2019.. [Mrežno]. Available: https://www.concettolabs.com/blog/flutter-single-codebase-application-for-ios-and-android/. [Pokušaj pristupa 2 rujan 2020.].

[7] R. Amadeo, »Arstechnica,« 2015. svibanj 2. [Mrežno]. Available: https://arstechnica.com/gadgets/2015/05/googles-dart-language-on-android-aims-for-java-free-120-fps-apps/. [Pokušaj pristupa 2 rujan 2020.].

[8] E. Protalinski, »Venturebeat,« 4 prosinac 2018.. [Mrežno]. Available: https://venturebeat.com/2018/12/04/google-launches-flutter-1-0-its-android-and-ios-mobile-app-sdk/. [Pokušaj pristupa 2. rujan 2020.].

[9] F. Lardinois, »Techcrunch,« 21 listopad 2014.. [Mrežno]. Available: https://techcrunch.com/2014/10/21/google-acquires-firebase-to-help-developers-build-better-

realtime-apps/?guccounter=1&guce_referrer=aHR0cHM6Ly9lbi53aWtpcGVkaWEub3JnLw&guce_referrer_sig=AQAAAJyacAPBMun0GzpLc_TknfoAiP1dmfmqPj7riEJsBZ_gIXev9U8996fj48d7kkZKO. [Pokušaj pristupa 2 rujan 2020.].

[10] F. Lardinois, »Techcrunch,« 29 travanj 2015. [Mrežno]. Available: https://techcrunch.com/2015/04/29/microsoft-shocks-the-world-with-visual-studio-code-a-free-code-editor-for-os-x-linux-and-windows/. [Pokušaj pristupa 2 rujan 2020.].

[11] »Stack Overflow,« [Mrežno]. Available: https://insights.stackoverflow.com/survey/2019#development-environments-and-tools. [Pokušaj pristupa 2 rujan 2020.].

SAŽETAK

Cilj ovog završnog rada bio je izraditi mobilnu aplikaciju za naručivanje proizvoda i praćenje njihove dostave u realnom vremenu. Korisnik ima mogućnost pokrenuti aplikaciju kao naručitelj ili dostavljač. Ovisno o odabiru, pruža se usluga stvaranja narudžbe ili popis već postojećih narudžbi koje čekaju dostavu. Korisnik koji naručuje može pratiti stanje u kojem je njegova narudžba i pratiti lokaciju dostavljača koji dostavlja zatraženu narudžbu, a dostavljač ima uvid u lokaciju naručitelja. Izvorni kod pisan je u programskom jeziku Dart. Pisan je i uređen u programu Visual Studio Code. Flutter je korišteni alat za razvoj mobilnih aplikacija koji omogućuje pokretanje aplikacije na više platformi. Podatci potrebni za rad aplikacije spremaju se na Firebase platformu. Rezultat ovog rada je aplikacija koju je moguće pokrenuti na mobilnom uređaju sa svim navedenim mogućnostima.

Ključne riječi: Android aplikacija, Dart, dostava, Firebase, Flutter

ABSTRACT

Mobile application for ordering and tracking delivery

The goal of this bachelor's thesis was to create a mobile application for ordering products and monitoring their delivery in real time. The user has the option to run the application as a client or deliverer. Depending on the choice, an order creation service or a list of existing orders awaiting delivery is provided. The ordering user can monitor the state of his order and track the location of the delivery person who delivers the requested order. The delivery person has an insight into the customer's location. The source code is written in the Dart programming language. It is written and edited in Visual Studio Code. Flutter is a software development kit used to develop mobile applications that allows applications to run on multiple platforms. The data required for the operation of the application is stored on the Firebase platform. The result of this work is an application that can be run on a mobile device with all the above requirements.

Key words: Android application, Dart, delivery, Firebase, Flutter

ŽIVOTOPIS

Krešimir Forjan rođen je 28. lipnja 1998. godine u Osijeku, Hrvatska. Pohađao je Osnovnu školu Bilje u Bilju. Tijekom osnovnoškolskog obrazovanja sudjelovao je na natjecanjima iz matematike osvojivši treće mjesto u županiji. Nakon završene osnovne škole upisuje III. gimnaziju Osijek. Za vrijeme srednjoškolskog obrazovanja bavi se veslanjem i sudjeluje na juniorskom svjetskom i europskom prvenstvu te je 2016. i 2017. godine bio dvostruki državni prvak. Maturirao je 2017. godine. Iste godine upisuje preddiplomski studij Računarstva na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija u Osijeku.

Krešimir Forjan

PRILOZI

DVD

- izvorni kod aplikacije
- videozapis s prikazom rada aplikacije
- pisani rad u .docx i .pdf formatu