Android aplikacija za auto školu

Safundžić, Tomislav

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

Sveučilišni studij

ANDROID APLIKACIJA ZA AUTOŠKOLU

Diplomski rad

Tomislav Safundžić

Osijek, 2018.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Obrazac D1: Obrazac za imenovanje Povjerenstva za obranu diplomskog rada

Osijek, 20.09.2018.

Odboru za završne i diplomske ispite

Imenovanje Povjerenstva za obranu diplomskog rada

Ime i prezime studenta:	Tomislav Safundžić
Studij, smjer:	Diplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	D 902 R, 22.09.2017.
OIB studenta:	39029575028
Mentor:	Izv. prof. dr. sc. Krešimir Nenadić
Sumentor:	
Sumentor iz tvrtke:	
Predsjednik Povjerenstva:	Izv. prof. dr. sc. Alfonzo Baumgartner
Član Povjerenstva:	Dr. sc. Tomislav Galba
Naslov diplomskog rada:	Android aplikacija za auto školu
Znanstvena grana rada:	**Programsko inženjerstvo (zn. polje računarstvo)**
Zadatak diplomskog rada:	Opisati način vođenja podataka polaznika u autoškoli. Predložiti način poboljšanja vođenja podataka polaznika. Izraditi i testirati Android aplikaciju koja omogućava lakše vođenje evidencije odrađenih sati vježbi vožnje polaznika uz prikaz prijeđenog puta vožnje i elemenata vožnje. Omogućiti instruktorima vožnje detaljan pregled podataka polaznika. Izraditi poslužiteljski dio za pohranu i dostavljanje podataka na zahtjev.
Prijedlog ocjene pismenog dijela ispita (diplomskog rada):	Izvrstan (5)
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: 3 bod/boda Postignuti rezultati u odnosu na složenost zadatka: 3 bod/boda Jasnoća pismenog izražavanja: 3 bod/boda Razina samostalnosti: 3 razina
Datum prijedloga ocjene mentora:	20.09.2018.

Potpis mentora za predaju konačne verzije rada u Studentsku službu pri završetku studija:	Potpis:
	Datum:

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

IZJAVA O ORIGINALNOSTI RADA

Osijek, 09.10.2018.

Ime i prezime studenta:	Tomislav Safundžić
Studij:	Diplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	D 902 R, 22.09.2017.
Ephorus podudaranje [%]:	6%

Ovom izjavom izjavljujem da je rad pod nazivom**: Android aplikacija za auto školu**

izrađen pod vodstvom mentora Izv. prof. dr. sc. Krešimir Nenadić

i sumentora

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis studenta:

SADRŽAJ

1. UVOD

Cilj ovog diplomskog rada je razraditi novo idejno rješenje koje bi omogućilo digitalizaciju rada autoškola. Izrađena Android aplikacija omogućuje lakše vođenje evidencije odrađenih sati vježbi vožnje polaznika uz prikaz prijeđenog puta vožnje i elemenata vožnje. Instruktorima je omogućen detaljan pregled podataka polaznika. Svi podaci su spremljeni na poslužitelj te su u svakom trenutku dostupni na uvid korisnicima – polaznicima vožnje i instruktorima. Android aplikacija je dio cijelog sustava *E – autoškole*. Sustav se sastoji još i od web dijela kojim upravljaju administratori samog sustava, te administratori pojedinih registriranih autoškola. Administratoru sustava je omogućena registracija novih autoškola u sustav te pripadnih administratora pojedinih autoškola. Administratorima autoškola je omogućena registracija novih kandidata ali i instruktora u njihovim autoškolama. Nakon takve registracije, korisnicima Android aplikacije – kandidatima i instruktorima je omogućena prijava s dodijeljenim korisničkim podacima.
U nastavku su opisani trenutni načini vođenja podataka polaznika u autoškolama i prednosti idejnog rješenja vođenja podataka polaznika. Zatim je dan detaljan uvid u idejno rješenje u vidu opisa funkcionalnosti i dijelova programskog koda. Potom slijedi faza testiranja Android aplikacije i prikaz rezultata.

1.1. Zadatak diplomskog rada

Opisati način vođenja podataka polaznika u auto školi. Predložiti način poboljšanja vođenja podataka polaznika. Izraditi Android aplikaciju koja omogućava lakše vođenje evidencije odrađenih sati vježbi vožnje polaznika uz prikaz prijeđenog puta vožnje i elemenata vožnje. Omogućiti instruktorima vožnje detaljan pregled podataka polaznika. Izraditi poslužiteljski dio za pohranu i dostavljanje podataka na zahtjev.

2. OPIS NAČINA RADA

U ovom poglavlju detaljno je opisan trenutni način vođenja podataka u autoškoli te je objašnjen način poboljšanja vođenja podataka korištenjem Android aplikacije.

2.1. Trenutni način vođenja podataka polaznika u autoškoli

Člankom 25. pravilnika o osposobljavanju kandidata za vozače definirani su podaci o kojima autoškola mora voditi evidenciju [1]:

1. Evidenciju o osposobljenim kandidatima za vozače – matičnu knjigu
2. Evidenciju o nastavi – dnevnik rada s imenikom

Autoškola je dužna trajno čuvati evidenciju o osposobljenim kandidatima, a ostalu evidenciju tri godine od dana završetka osposobljavanja kandidata za vozača.

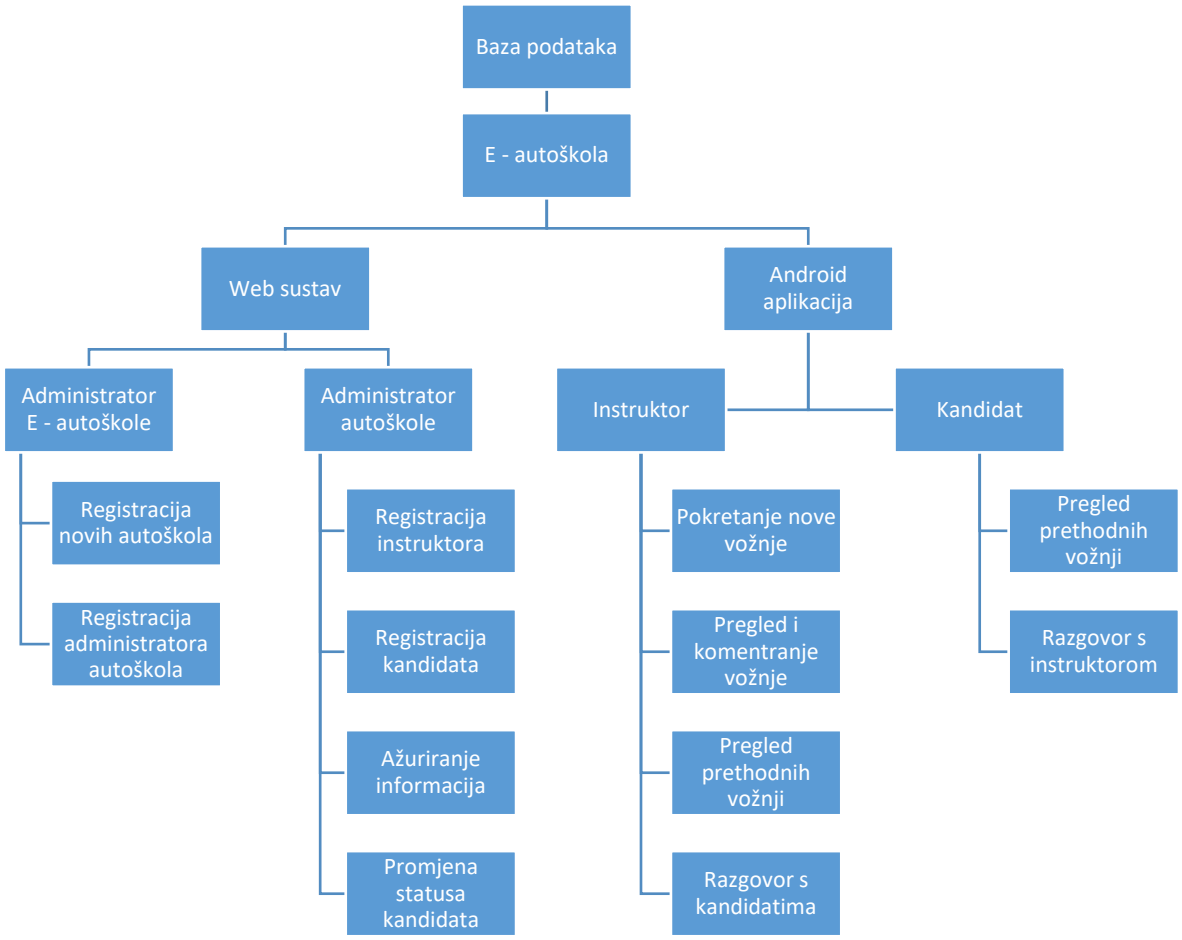
Autoškola Hertz Požega služi se obveznom matičnom knjigom od strane HAK-a. U njoj su svi osobni podaci o kandidatima, podaci o savladavanju ili ne savladavanju ispita iz prometnih propisa, ispita prve pomoći te ispita vožnje. Riječ je o digitalnom obliku matične knjige čiji je sustav izrađen od strane HAK-a. Sukladno zakonu, Autoškola Hertz vodi sve zapise iz matične knjige i u pisanom obliku, gdje se ručno pišu podaci istovjetni onima u digitalnoj matičnoj knjizi. Pri ručnom upisivanju podataka vodi se računa da se podaci istih identifikacijskih brojeva poklapaju i u digitalnoj i u ručno vođenoj matičnoj knjizi.

Uz matične knjige, Autoškola Hertz vodi i bazu podataka za vlastite potrebe. Sustav se vodi samo u digitalnom obliku. Prikupljaju se i spremaju podaci o kandidatima, instruktorima – mjesečni izvodi prijeđenih kilometara, potrošenog goriva i sl. Sustav generira i izdaje račune kandidatima u ovisnosti o unesenim postavkama i načinu plaćanja. Izdavanjem računa, vodi se evidencija plaćanja te administrator potvrđuje plaćanje kandidata pri dostavi potvrde o provedenom plaćanju. Knjižica kandidata za vozača popunjava instruktor s podacima o vožnji. Nakon toga, administrator vožnje potrebne podatke unosi u matičnu knjigu, odnosno u vlastiti izrađeni sustav. Ne postoji zajednička baza podatka u kojoj bi postojao jedinstveni unos podataka koji bi bili dostupni u svih navedenim sustavima.

Također, neke autoškole ne vode digitalni oblik baze podataka za vlastite potrebe, već samo ručno pisani oblik. Ručno izrađuju i popunjavaju potvrde o plaćanju i ostale potrebne dokumente u internom korištenju.

2.2. Opis poboljšanja trenutnog načina vođenja podataka

Rješenje koje nudi sustav *E – autoškola* omogućuje potpunu digitalizaciju podataka pojedinih autoškola. Također, nudi jednostavan i brz prikaz detalja vožnji pojedinih kandidata, omogućuje lak način dogovora slijedeće vožnje između kandidata i instruktora, jednostavno pokretanje nove vožnje skeniranjem QR koda kandidata, oporavak zaboravljene zaporke u dva klika.



Sl. 2.1. Hijerarhija sustava

Kako je vidljivo na slici 2.1 sustav *E – autoškole* se sastoji od dvije cjeline – web sustava i Android aplikacije. Web sustavu pristup imaju administratori sustava te administratori pojedinih autoškola. Administratorima sustava je omogućeno dodavanje novih autoškola s pripadajućim administratorima. Administratori autoškola registriraju nove instruktore i kandidate, ažuriraju njihove informacije te mijenjaju statuse kandidata u ovisnosti o njihovom trenutnom stanju u autoškoli.

Pristup sadržaju Android aplikacije imaju samo prethodno registrirani instruktori i kandidati. Android aplikacija je zapravo digitalizirana verzija Knjižice kandidata za vozača. Instruktorima je omogućeno pokretanje nove vožnje na način da skeniraju QR kod kandidata. Za vrijeme vožnje i pred sam kraj vožnje imaju mogućnost komentiranja netom obavljene vožnje. Također, instruktorima je omogućen i pregled prethodnih vožnji svih aktivnih kandidata, te razmjena poruka s njima. Kandidati imaju na uvid vlastite informacije, pregled prethodnih vožnji te razmjenu poruka s dodijeljenim instruktorom.

Cijeli sustav zapisuje i čita podatke iz baze podataka koja radi u realnom vremenu te na taj način omogućuje korisnicima trenutni uvid u željene podatke bez potrebe za osvježavanjem sadržaja. Baza podataka je jedinstvena i za web i za Android aplikaciju.

Izrađena baza podataka mogla bi se implementirati i za vlastiti sustav Autoškole Hertz, ali bi mogla služiti i za dohvaćanje podataka o vožnji, kandidatima i instruktorima u svrhu zapisa u matičnu knjigu od strane HAK-a. Na taj način bi se omogućilo rasterećenje administratora, koji iste podatke mora upisivati u dva sustava, te još dodatno voditi ručno pisanu bazu podataka.

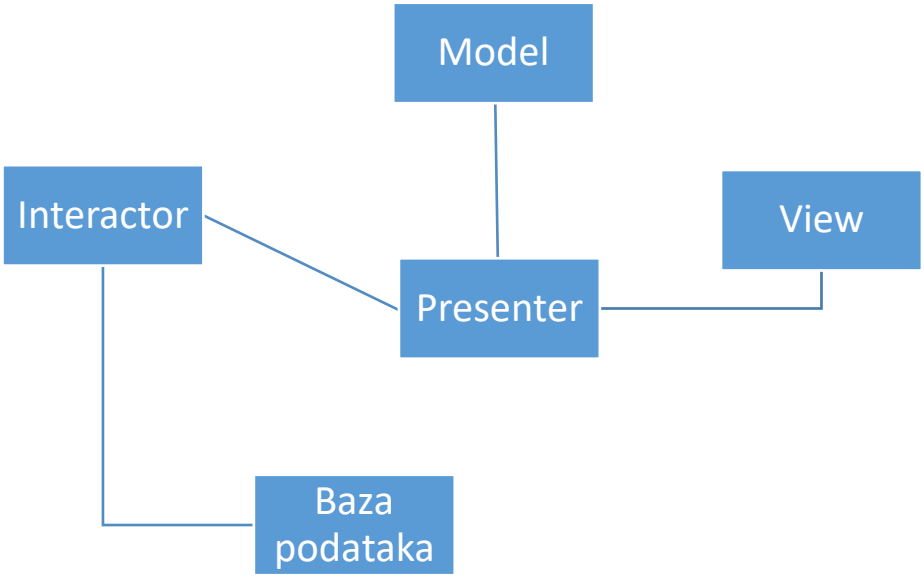
3. REALIZACIJA APLIKACIJE

U ovom poglavlju dan je na uvid detaljan način izrade svakog dijela aplikacije uz opisane korištene alate i tehnologiju.

3.1. Korišteni alati i tehnologije

Sustav *E-autoškole* je podijeljen na dva dijela – Android aplikacija te web dio. Za potrebe izrade Android aplikacije korišten je program *Android Studio*. Riječ je o službenom razvojnom okruženju za Google-ov Android operacijski sustav, neovisno o vrsti Android uređaja. Aplikacija je pisana u programskom jeziku *Kotlin*. *Kotlin* je statički tipizirani programski jezik čija sintaksa nije kompatibilna s *Java* kodom, no *Kotlin* je potpuno interoperabilan s *Java* programskim jezikom jer radi na *Java Virtual Machine (JVM)*.[2]

U Android aplikaciji korištena je MVP arhitektura. [3] Riječ je o *Model-View-Presenter* arhitekturi gdje se programski kod dijeli na tri osnovna dijela – *Model* su sve klase definirane unutar projekta, *View* je odgovoran za prikaz sadržaja na zaslonu uređaja te nema nikakve veze s logikom sustava. Definiran je uglavnom aktivnostima, fragmentima te ostalim sadržajem unutar aktivnosti, odnosno fragmenata. *Presenter* je zadužen za ostvarenje komunikacije između *Viewa* i *Modela* te *Interactora*. *Interactor* je zasebna klasa kojoj je zadatak rad s bazom podataka – na osnovu dobivene logike od strane *Presentera*, *Interactor* vrši zapis, čitanje, izmjenu ili brisanje podataka iz baze podataka. Dakle, MVP arhitektura je definirana na slijedeći način – bilo da se dogodi nekakav događaj u aktivnosti, odnosno fragmentu ili bilo da se stvara nova aktivnost, odnosno fragment, *View* zahtjeva sadržaj za prikaz (podatke) od *Presentera*. Ako je riječ o sadržaju nevezanog za bazu podatka, *Presenter* obavlja sav logički dio oko dohvata sadržaja i tako generirani sadržaj vraća *Viewu* te *View* vrši prikaz dohvaćenog sadržaja. Ako zatraženi sadržaj ima veze s bazom podataka, *Presenter* poziva *Interactora* od kojeg zahtjeva traženi sadržaj. *Interactor* obrađuje zahtjev *Presentera* na način da dohvaća podatke iz baze, briše ih, ili dodaje podatke u bazu podataka. U ovisnosti o obavljenoj aktivnosti, *Interactor* poziva *Presenter* te mu vraća tražene podatke, ili ako je bio slučaj brisanje ili dodavanja novih podataka u bazu, obavještava da je zapisivanje ili brisanje uspješno ili neuspješno obavljeno. Potom *Presenter* prosljeđuje *Viewu* dohvaćene podatke iz baze podatka ili obavijest o uspješnom ili neuspješnom brisanju ili dodavanju podataka u bazu podataka. *View* obavlja prikaz rezultata logike obavljene od strane *Presentera*.



Sl. 3.1. MVP arhitektura

Ostali dio sustava – web dio – je pisan u PHP-u. *Firebase* je korišten za bazu podataka. Riječ je o platformi od strane Google-a koja omogućuje čitanje i zapisivanje podataka, te interakciju s podacima u realnom vremenu. Zapisivanje podataka se vrši u JSON formatu. [4]

3.2. Baza podataka i autorizacija

Kako je već rečeno, *Firebase Database* se koristi za bazu podataka. Riječ je o bazi podataka u oblaku čiji se podaci spremaju u JSON format kao objekti. Svaki podatak spremljen u *Firebase Database* je tipa *String*. [5]

Prije svega, potrebno je kreirati novi *Firebase* projekt. Nakon definiranja traženih postavki pri kreiranju projekta, potrebno je implementirati *Firebase* u projekt – web dio sustava i Android aplikaciju. Za implementaciju *Firebasea* na web dio sustava potrebno je kopirati generiranu skriptu pri kreiranju *Firebase* projekta i zalijepiti ju u zaglavlje PHP dokumenta.

```
<script src="https://www.gstatic.com/firebasejs/5.3.1/firebase.js"></script>
<script>
  var config = {
    apiKey: "GENERIRANI_API_KEY",
    authDomain: "e-autoskola.firebaseapp.com",
    databaseURL: "https://e-autoskola.firebaseio.com",
    projectId: "e-autoskola",
    storageBucket: "e-autoskola.appspot.com",
    messagingSenderId: "961013229665"
  };
```

```
  firebase.initializeApp(config);
</script>
```

Programski kod 3.1. Implementacija Firebase projekta

Za implementaciju *Firebase* projekta u Android projekt potrebno je implementirati potrebne biblioteke u *build.gradle* datoteku Android projekta.

Gradle Dependency Line	Service
com.google.firebase:firebase-core:16.0.1	Analytics
com.google.firebase:firebase-database:16.0.1	Realtime Database
com.google.firebase:firebase-firestore:17.0.4	Cloud Firestore
com.google.firebase:firebase-storage:16.0.1	Storage
com.google.firebase:firebase-crash:16.0.1	Crash Reporting
com.google.firebase:firebase-auth:16.0.2	Authentication
com.google.firebase:firebase-messaging:17.1.0	Cloud Messaging
com.google.firebase:firebase-config:16.0.0	Remote Config
com.google.firebase:firebase-invites:16.0.1	Invites and Dynamic Links
com.google.firebase:firebase-ads:15.0.1	AdMob
com.google.firebase:firebase-appindexing:16.0.1	App Indexing
com.google.firebase:firebase-perf:16.0.0	Performance Monitoring
com.google.firebase:firebase-functions:16.1.0	Cloud Functions for Firebase Client SDK
com.google.firebase:firebase-ml-vision:16.0.0	ML Kit (Vision)
com.google.firebase:firebase-ml-model-interpreter:16.0.0	ML Kit (Custom Model)

Sl. 3.2. Lista dostupnih *Firebase* biblioteka [6]

Zbog sigurnosti i neovlaštenog čitanja i pisanja u bazu podataka potrebno je definirati pravila čitanja i pisanja podataka u i iz baze podataka. Omogućeno je da samo prijavljeni korisnici mogu pristupiti podacima koje se nalaze u bazi podataka.

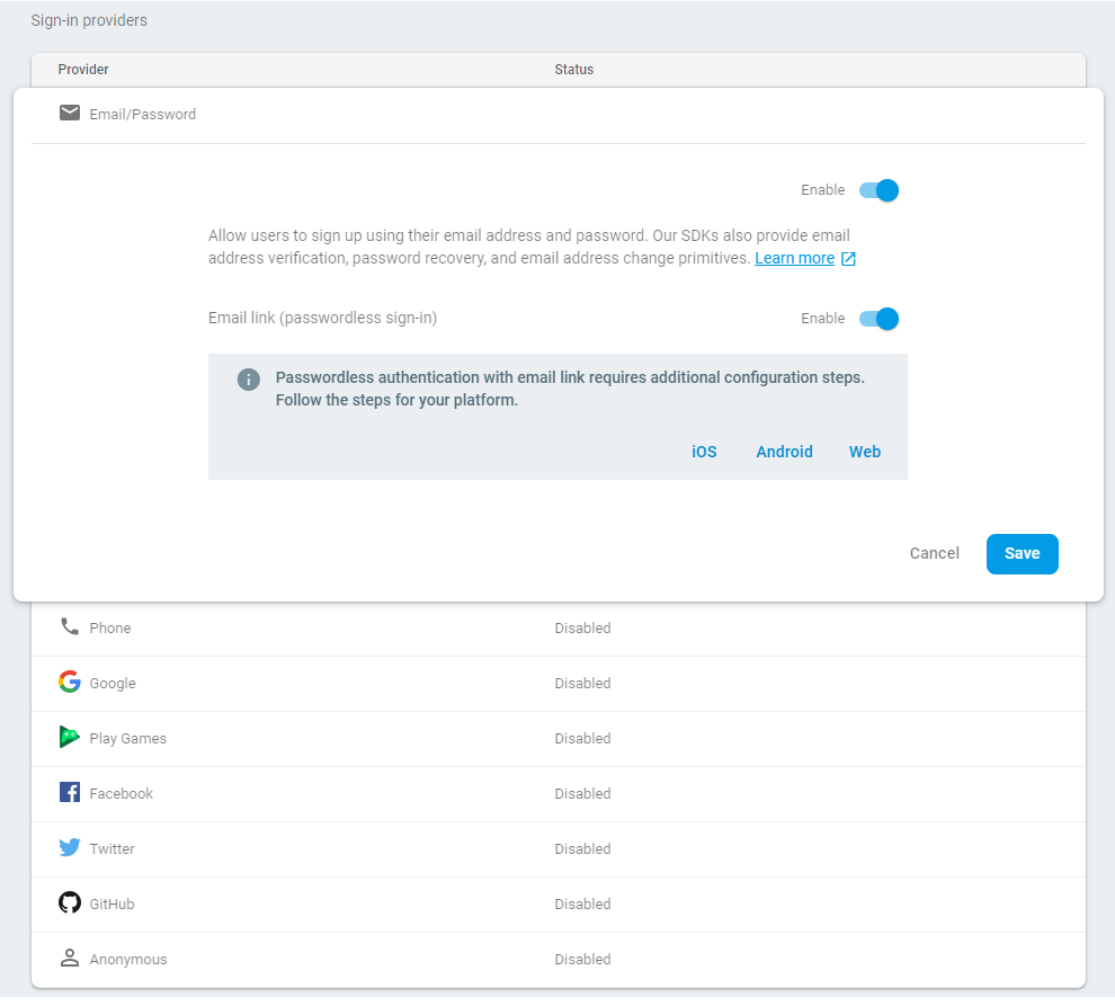
```
{
  "rules": {
    ".read":  "auth != null",
    ".write":  "auth != null"
  }
}
```

Programski kod 3.2. Definirana pravila čitanja i pisanja u bazi podataka

Sl. 3.3. Zapis podataka u bazi podataka

Baza se sastoji od dva glavna čvora – *schools* i *users*. Svaka registrirana škola je dodana kao dijete čvoru *schools* i to tako da je naziv čvora jedinstvena identifikacijska oznaka generirana pri registraciji. Na sličan način su generirana i djeca čvora *users*. Svaki naziv djeteta je zapravo jedinstveni *id* registriranog korisnika. Dodatno, kandidati imaju čvorove *chats* i *rides* gdje se sprema povijest razgovora, odnosno povijest vožnji. Svaki razgovor se sprema u čvor čiji je naziv vrijeme u milisekundama kad je poruka poslana. Svaka vožnja se sprema prema rednom broju vožnje, te se lokacija bilježi i sprema u bazu svakih 30 sekundi. Dodatno se još na kraju vožnje sprema i komentar instruktora.

Kako bi se mogla koristiti Android aplikacija, potrebna je prethodna prijava preko korisničkog računa dodijeljenog od strane autoškole. Autorizacija, odnosno prijava korisnika u aplikaciju omogućeno je pomoću *Firebase Authentication*. Omogućena je prijava samo putem emaila i lozinke. [7]



Sl. 3.4. Postavke autorizacije

Prijava u Android aplikaciji je izvedena na slijedeći način:

1. Unutar *MainActivity-a* prilikom klika na dugme poziva se *signIn* metoda u *Presenteru* koja ima dva elementa – upisani email i lozinku.

```
private fun startSignIn() {
    mProgressDialog.show()
    presenter.signIn(email_login.text.toString(),
password_login.text.toString())
}
```

Programski kod 3.3. Metoda *startSignIn* u *MainActivity*

2. Unutar *signIn* metode u *Presenteru* provjeravaju se upisani sadržaj u polje email i polje lozinka. U ovisnosti o ishodu provjere, pokreće se prijava, odnosno prikazuje se pogreška.

```
override fun signIn(email: String, password: String) {
    if (email.isValidEmail()) {
        if (password.isNotEmpty()) {
            userInteractor.performFirebaseLogin(email, password)
        } else {
            view.setPasswordError()
        }
    } else {
        view.setEmailError()
    }
}
```

Programski kod 3.4. Metoda *signIn* u *Presenteru*

3. U slučaju da uneseni podaci nisu valjani, poziva se *onFailure* metoda u *Presenteru* koja poziva metodu za prikaz greške unutar *Viewa*. U slučaju da se uneseni podaci valjani, poziva se metoda *performFirebaseLogin* unutar *UserInteractora*. Unutar metode *performFirebaseLogin* se provjerava postoji li korisnik s prethodno unesenim emailom i lozinkom. U ovisnosti o ishodu provjere, poziva se određena metoda u *Presenteru*.

```
fun performFirebaseLogin(email: String, password: String) {

    mAuth.signInWithEmailAndPassword(email, password)
            .addOnCompleteListener { task ->
                if (task.isSuccessful) {
                    val user = mAuth.currentUser
                    onLoginListener.onSuccess(user)
                } else {
                    onLoginListener.onFailure()
                }
            }
}
```

Programski kod 3.5. Metoda *performFirebaseLogin* unutar *UserInteractora*

4. U slučaju da uneseni podaci ne odgovaraju niti jednom korisniku, *Presenter* poziva metodu *errorWrongUserInformation* u *Viewu* koja prikazuje obavijest da uneseni podaci nisu

valjani. U slučaju da uneseni podaci odgovaraju registriranom korisniku, unutar *onSuccess* metode u *Presenteru* poziva se *getUserRole* metoda u *DatabaseInteractoru* te se provjerava uloga prijavljenog korisnika kako bi se prikazala valjana aktivnost u ovisnosti je li korisnik instruktor ili kandidat.

```
override fun getUserRole(user: FirebaseUser?) {

    val userRoleListener = object : ValueEventListener {
        override fun onDataChange(dataSnapshot: DataSnapshot) {
            val userRole = dataSnapshot.value.toString()
            when (userRole) {
                CANDIDATE -> databaseListener.setViewToCandidateMainActivity()
                INSTRUCTOR -> databaseListener.setViewToInstructorMainActivity()
                else -> databaseListener.setAuthorisationError()
            }
        }

        override fun onCancelled(databaseError: DatabaseError) {}
    }
    if (user != null) {
         myRef.child(USERS).child(user.uid).child(ROLE)
           .addListenerForSingleValueEvent(userRoleListener)
    }

}
```

Programski kod 3.6. Metoda *getUserRole* unutar *DatabaseInteractora*

5. U ovisnosti o vrsti uloge, poziva se odgovarajuća metoda u *Presenteru* koja otvara novu aktivnosti za pripadajućeg korisnika.

```
override fun setViewToCandidateMainActivity() {
    view.startCandidateMainActivity()
}
```

Programski kod 3.7. Metoda *setViewToCandidateMainActivity*

6. Unutar *Viewa* se definira pripadajuća aktivnost koja se poziva na slijedeći način:

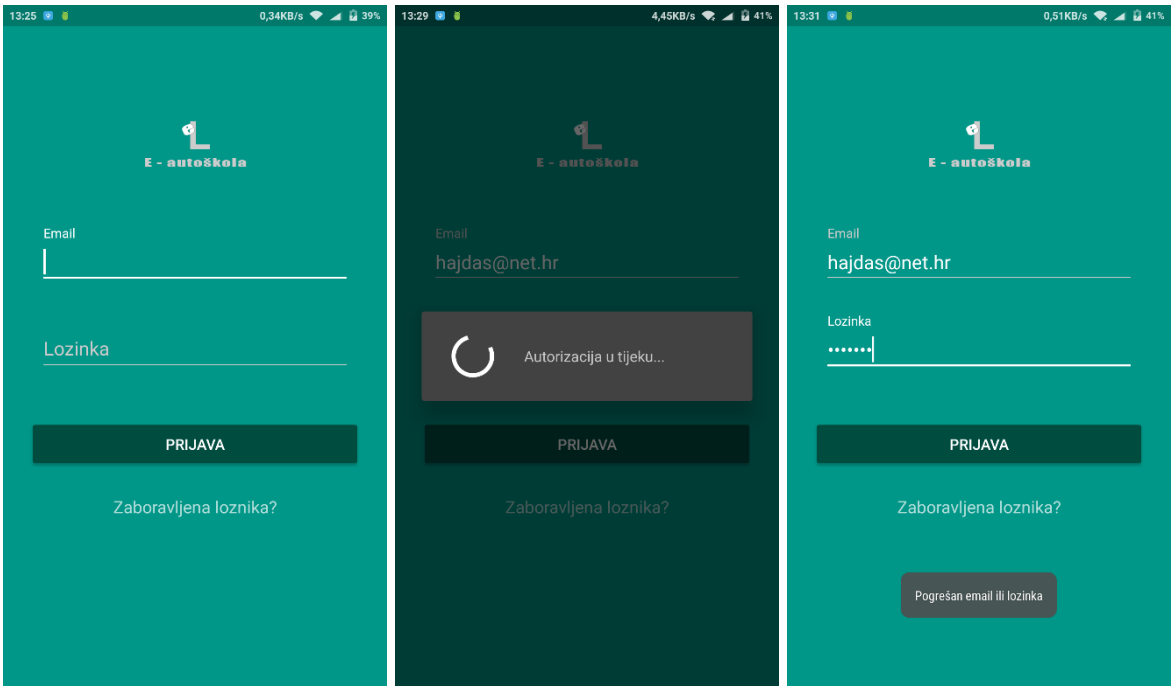
```
override fun startCandidateMainActivity() {
    startActivity(CandidateMainActivity.getLaunchIntent(this))
}
```

Programski kod 3.8. Metoda *startCandidateMainActivity*

7. Metoda *getLaunchIntent* je definiran u aktivnosti *CandidateMainActivitya* unutar *companion objecta*.

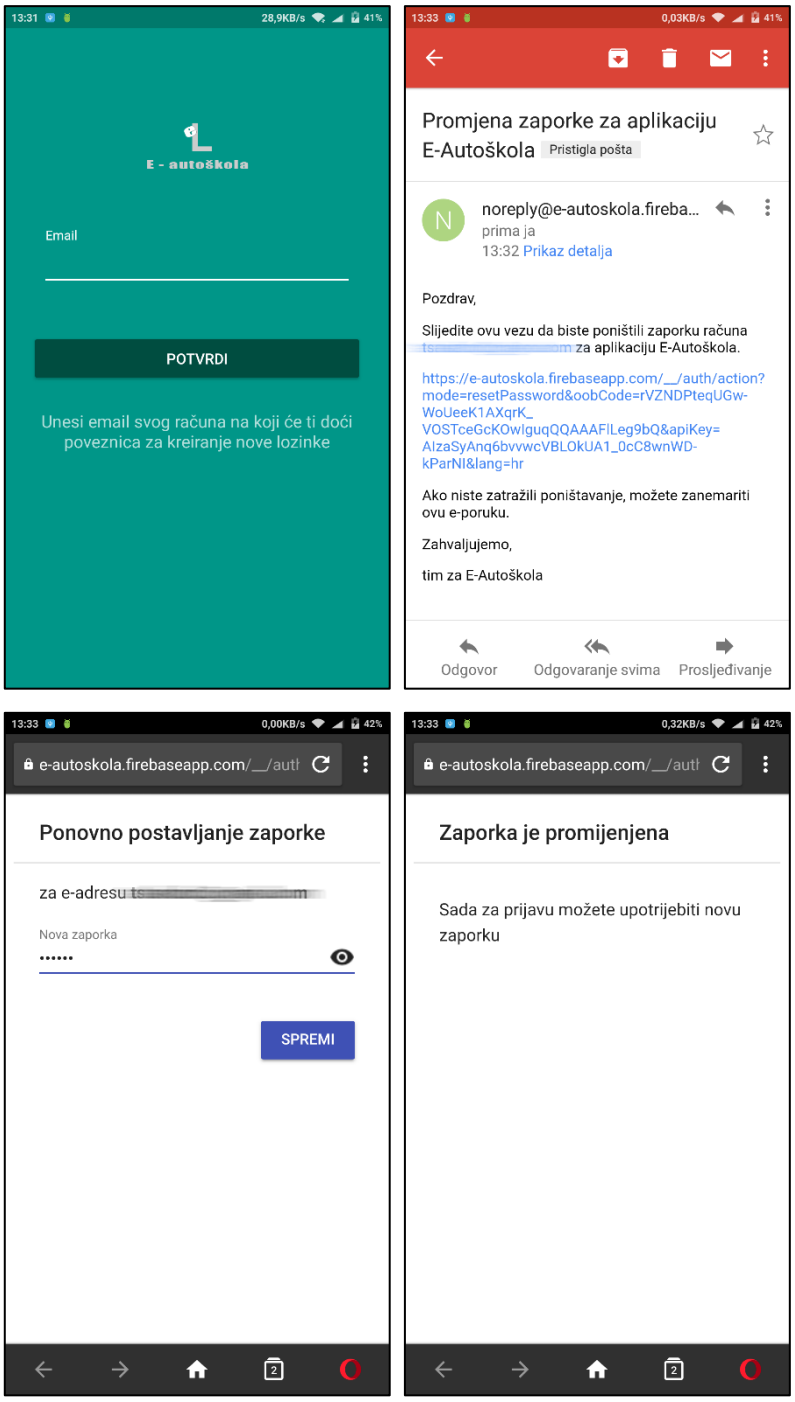
```
companion object {
    fun getLaunchIntent(from: Context) =
from.getIntent<CandidateMainActivity>().apply {}
}
```

Programski kod 3.9. Metoda *getLaunchIntent*



Sl. 3.5. Prikaz prijave korisnika u sustav

Omogućeno je i ponovno postavljanje lozinke u slučaju da je postojeća zaboravljena ili se želi promijeniti. Klikom na tekst *Zaboravljena lozinka?* pokreće se nova aktivnost *ForgottenPasswordActivity*. Od korisnika se zahtjeva upis emaila. Nakon toga klikom na potvrdi pokreće se metoda *checkEmail* u *Presenteru*. Metodom *sendPasswordResetEmail* se provjerava postoji li registrirani korisnik s upisanom email adresom. U slučaju da ne postoji pojavljuje se odgovarajuća obavijest. U slučaju da postoji korisnik čija je email adresa istovjetna s prethodno upisanom email adresom, pokreće se *onSucces* metoda te korisnik dobiva na email adresu poveznicu za promjenu lozinke.



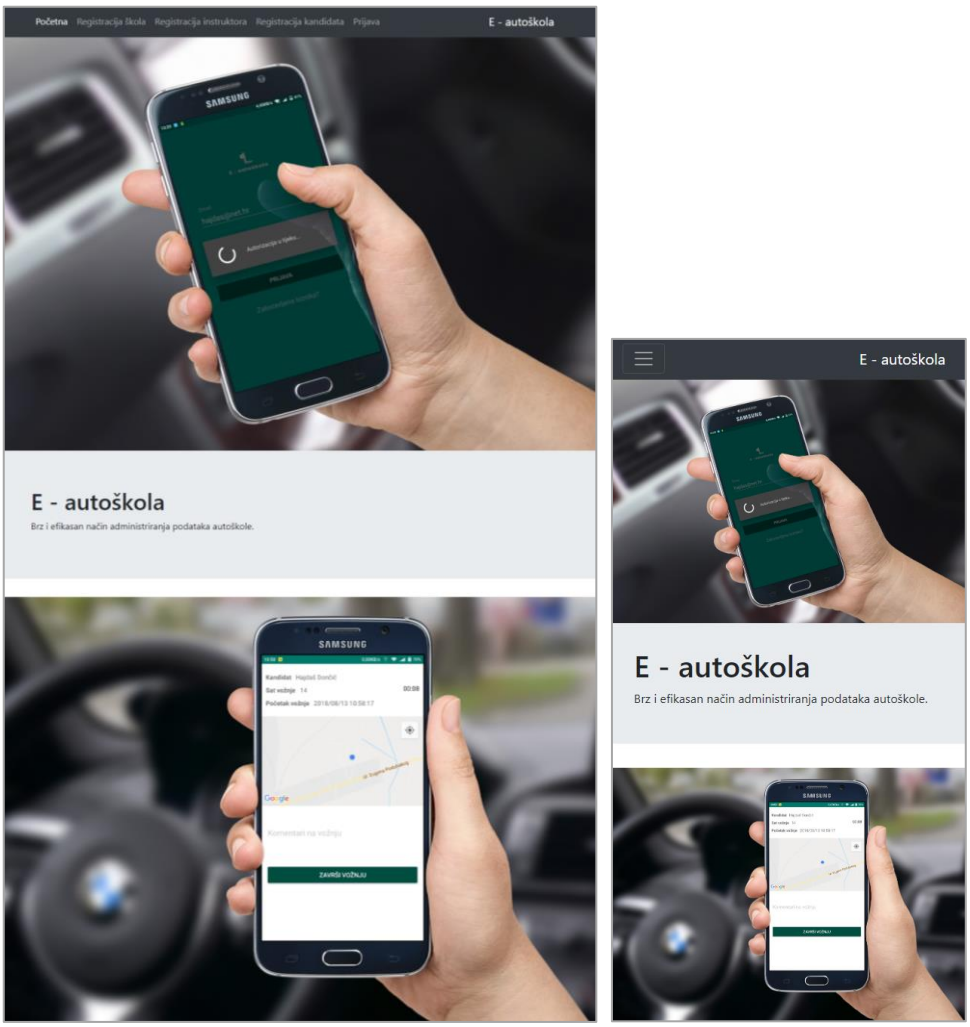
Sl. 3.6. Promjena lozinke

3.3. Web dio sustava

Web dio sustava koriste administratori sustava *E – autoškole* kako bi registrirali nove autoškole i pripadajuće administratore. Također, web dijelom sustava se služe i administratori pojedinih autoškola kako bi dodali nove kandidate i nove instruktore u bazu podataka vlastite

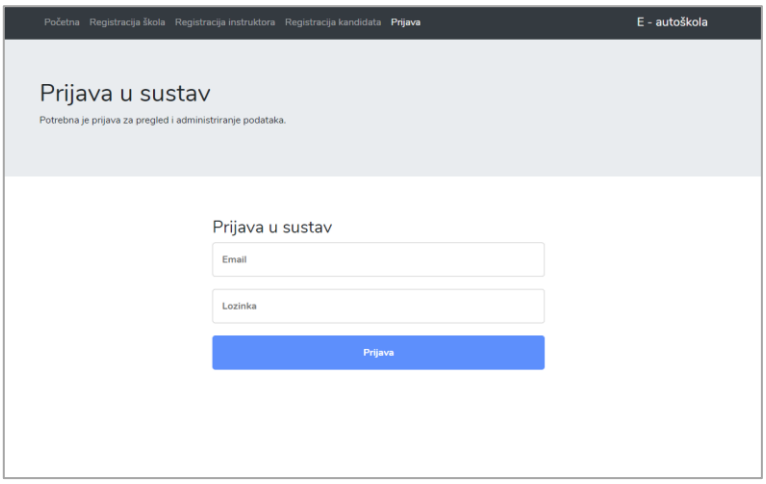
autoškole. Web dio sustava je u cijelosti napravljen u PHP-u te koristi istu bazu podataka kao i Android aplikacija. Na taj način je osiguran jedinstven prikaz podataka i jedinstvena sinkronizacija istih bez dodatne potrebe za kreiranjem međusloja između baza. Pristup administriranju je ograničen u ovisnosti o dodijeljenoj ulozi prilikom registriranje. Pristup registriranju novih autoškola i novih administratora autoškola imaju samo administratori sustava *E – autoškola*. Pristup registriranju novih instruktora i kandidata u pojedinu autoškolu imaju samo administratori tih autoškola. Na taj način osiguran je ispravan i kvalitetan pristup podacima i stvaranje novih podataka za pojedine autoškole bez mogućnosti manipulacije i ugrožavanja sigurnosti podataka.

Web dio je potpuno responzivan – korišten je *Bootstrap* – pa je osiguran kvalitetan i jedinstven prikaz kako na mobilnim uređajima, tako i na osobnim računalima, odnosno uređajima većih rezolucija. [8]

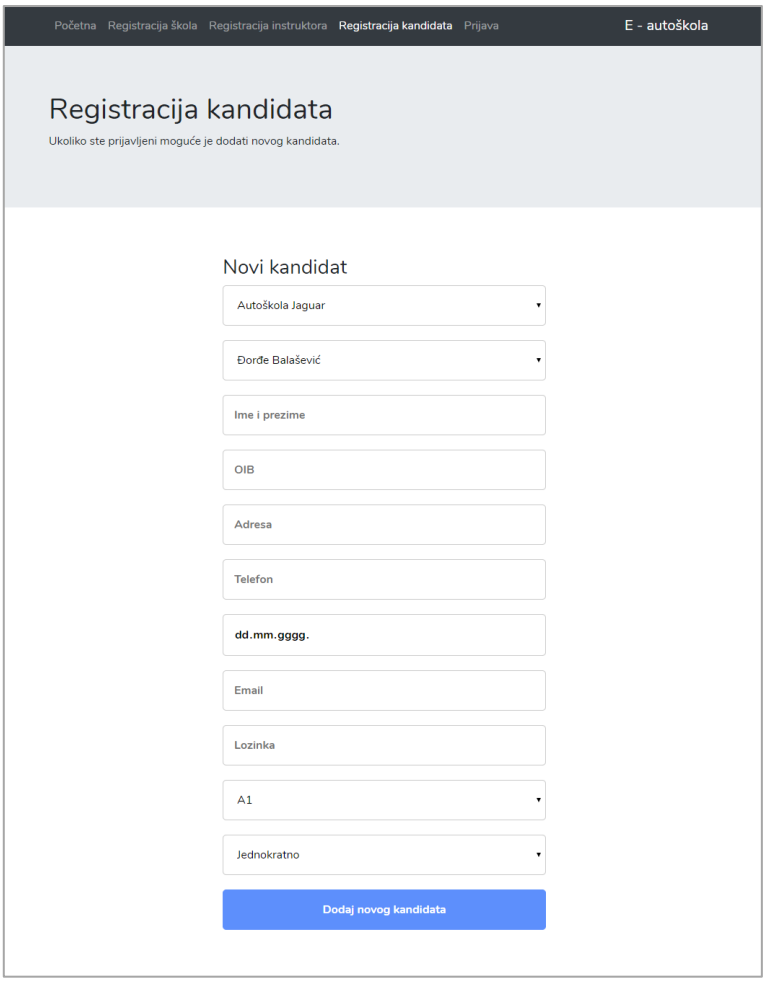


Sl. 3.7. Početna stranica web sustava

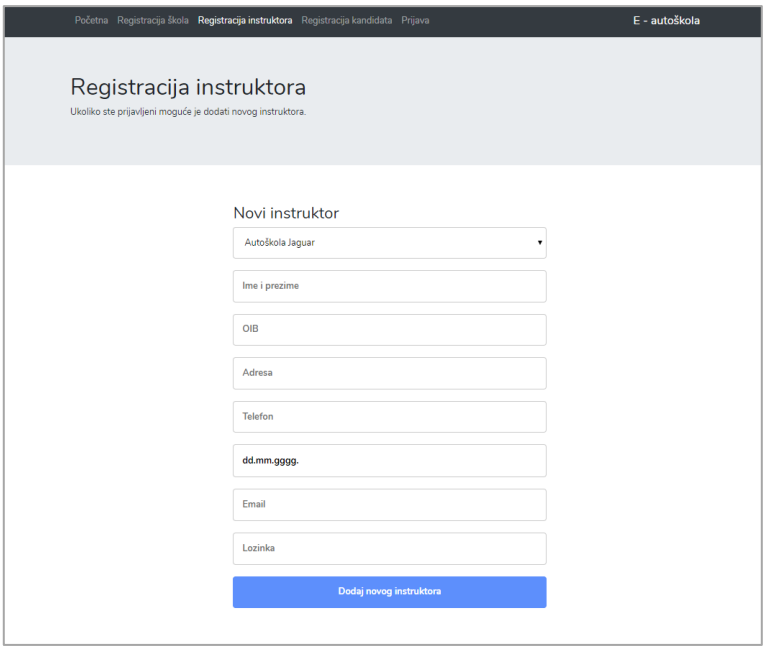
U slučaju da korisnik nije prijavljen, a želi administrirati podacima, sustav ga automatski preusmjerava na prijavu. Tek nakon prijave omogućeno je administriranje podacima u ovisnosti o dodijeljenoj ulozi u sustavu. Zasloni prijave, registracije novih kandidata i instruktora, te zaslon o obavijesti o nemogućnosti pristupa podacima zbog nedostatka privilegija nalaze se u nastavku.



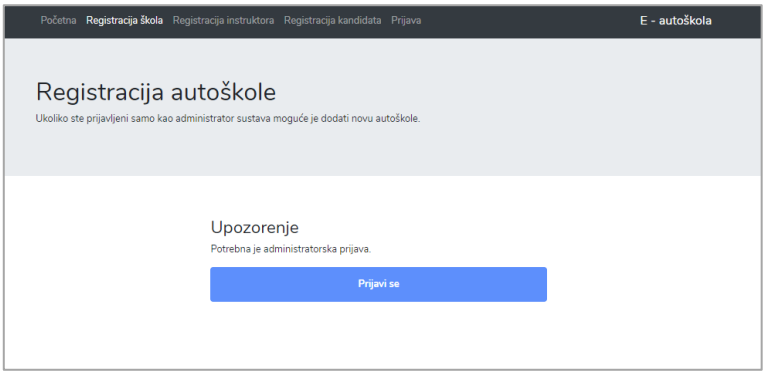
Sl. 3.8. Prijava u web sustav



Sl. 3.9. Registracija novog kandidata u web sustavu



Sl. 3.10. Registracija novog instruktora u web sustavu



Sl. 3.11. Obavijest o nedostatku privilegije nad pristupom podacima

3.4. Dio aplikacije namijenjen polaznicima vožnje

Polaznicima vožnje, odnosno kandidatima dostupan je prikaz vlastitih informacija, kao i informacija o upisanoj autoškoli te dodijeljenom instruktoru. Imaju uvid u detalje prethodno obavljenih vožnji te imaju mogućnost dogovora oko termina vožnji i ostalih potrebnih informacija s dodijeljenim instruktorom u obliku *chata*.

Definirana je glavna aktivnost za kandidate – *CandidateMainActivity* u kojem je definiran osnovni izgled i unutar kojega se pozivaju fragmenti u ovisnosti o odabiru korisnika iz izbornika. Izbornik je dizajniran pomoću *Bottom navigation* elementa u *Android Material Designu*. [9] Potrebno je definirati izgled i količinu objekata unutar *menu* izbornika. To je definirano unutar *.xml* datoteke:

```
<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto">
    <item
        android:id="@+id/accountInfo"
        android:enabled="true"
        android:icon="@drawable/ic_account"
        android:title="@string/account"
        app:showAsAction="ifRoom" />
    <item
        android:id="@+id/rideHistory"
        android:enabled="true"
        android:icon="@drawable/ic_ride_history"
        android:title="@string/rideHistory"
        app:showAsAction="ifRoom" />
    <item
        android:id="@+id/chat"
        android:enabled="true"
        android:icon="@drawable/ic_chat"
        android:title="@string/chat"
        app:showAsAction="ifRoom" />@
</menu>
```

Programski kod 3.10. Izbornik za *Bottom navigation*

Kako bi izbornik imao i funkcionalnost, unutar glavne aktivnosti kandidata potrebno je definirati *listener* za pojedine elemente *Bottom navitationa*. *Listener* osluškuje ponašanje korisnika i u ovisnosti o odabranom elementu izbornika mijenja postojeći fragment u zatraženi. Naravno, sva logika se odvija unutar metoda u *Presenteru,* koje se pozivaju iz *Viewa* na osnovu definiranog *listenera*.

```
override fun checkNewFragment(item: MenuItem, fragmentLayout: Int, activity:
Activity) {
    when (item.itemId) {
        R.id.accountInfo -> {
            changeFragments(fragmentLayout, CandidateAccountInfo(), activity)
        }
        R.id.rideHistory -> {
            changeFragments(fragmentLayout, CandidateRideHistory(), activity)
        }
        R.id.chat -> {
            changeFragments(fragmentLayout, ChatFragment(), activity)
        }
    }
}

override fun callFragmentChange(fragmentLayout: Int, fragment: Fragment,
activity: Activity) {
    changeFragments(fragmentLayout, fragment, activity)
}
```

Programski kod 3.11. Metoda *checkNewFragment*

Dohvaćanje i prikaz informacija u fragmentu *AccountInfo* je u cijelosti napravljen preko *DatabseInteractora* i *UserInteractora*. *Presenter* poziva *UserInteractora* na dohvaćanje jedinstvenog identifikatora trenutno prijavljenog korisnika.

```
var mAuth = FirebaseAuth.getInstance()

override fun getUserUid(): String {
    return mAuth.uid.toString()
}
```

Programski kod 3.12. Dohvaćanje jedinstvenog identifikatora korisnika

Nakon uspješno dohvaćenog jedinstvenog identifikatora, *Presenter* poziva *DatabaseInteractora* koji pretražuje korisnika prema dobivenom jedinstvenom identifikatoru. Unutar funkcije *onDataChange* dohvaćaju se svi podaci korisnika na osnovu prethodno definirane putanje *users/jedinstveni_identifikator*. Na sličan način, samo preko drugih putanja, se dobivaju i informacije o upisanoj autoškoli i dodijeljenom instruktoru. U fragmentu je dodatno implementiran i prikaz dodijeljenog QR koda kandidatu pri registraciji. Potrebno je pozvati *DatabaseInteractor* koji dohvaća QR kod u obliku slike tipa *.png*. Slika je spremljena pod nazivom istovjetnim kandidatovim jedinstvenim identifikatorom.

```
override fun getCandidateImage(uid: String) {

FirebaseStorage.getInstance().reference.child("$CANDIDATES/$uid").downloadUrl
            .addOnSuccessListener{ uri ->
                val imageURL: String = uri.toString()
                databaseListener.setUserImage(imageURL)

            }.addOnFailureListener{}
}
```

Programski kod 3.13. Metoda *getCandidateImage*

Aktivnost za slanje poruka – *MessageActivity* – služi za prikaz prethodnih poruka i omogućuje slanje novih poruka. Nove poruke automatski se pojavljuju pri upisu u bazu. Otvaranjem aktivnosti *Presenter* poziva *DatabaseListener* koji mu vraća prethodne poruke ako postoje i to tako da traži u bazi podataka razgovore unutar čvora koji nosi naziv *candidateUid_instructorUid*. Metodi se predaju potrebni podaci *candidateUid* i *instructorUid* prilikom njenog poziva. Svaka poruka se zapisuje u podčvor čiji je naziv zapravo trenutno vrijeme u milisekundama. Riječ je o zapisu vremena, odnosno vremenskoj oznaci UNIX – poznatoj i kao POXIS vremenska oznaka ili vrijeme epohe. Predstavlja trenutak definiran kao vrijeme u

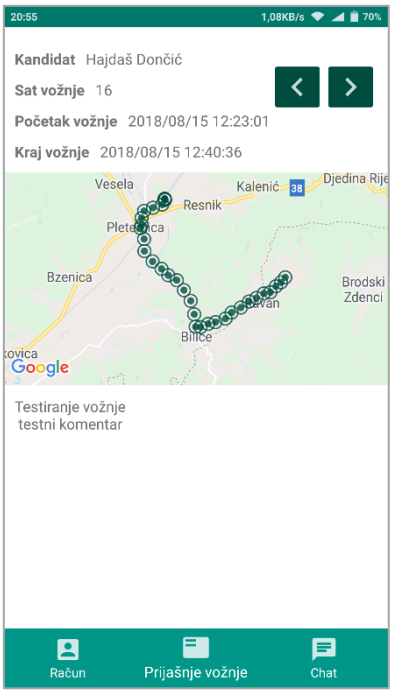
milisekundama koje je proteklo od reference poznate kao UNIX epoha – 00:00:00 01.01.1970. godine. [10]

Potrebno je i definirati i dva tipa poruka – poruke koje je poslao trenutno prijavljeni korisnik i poruke koje je poslao korisnik koji razgovara s prijavljenim korisnikom. To je napravljeno na način da se provjerava jedinstveni identifikator trenutno prijavljenog korisnika i jedinstveni identifikator pošiljatelja poruke. Ako su isti, definira se varijabla *VIEW_TYPE_ME* kojom se označava da trenutna poruka u procesu dohvaćanja poruka prijavljenog korisnika i potrebno ju je prikazati s desne strane. *LayoutInflater* je zadužen za prikaz – postavlja se prikaz koji je prethodno definiran unutar *layout* mape u *.xml* datoteci. Ako identifikator prijavljenog korisnika i identifikator pošiljatelja poruke nisu isti, definira se varijabla *VIEW_TYPE_OTHER* kojom se označava da trenutna poruka u procesu dohvaćanja nije poruka prijavljenog korisnika i potrebno ju je prikazati s lijeve strane s pripadajućom oznakom početnog slova imena.

Sl. 3.12. Razgovor instruktora i kandidata

Pri odabiru prikaza prethodno odvoženih sati, kao početni prikaz, prikazuju se podaci zadnje obavljene vožnje. Poziva se *Presenter* koji poziva *DatabaseInteractor* u kojem se provjerava zadnji podčvor unutar čvora *rides*. Dohvaćaju se podaci o vožnji i prikazuju unutar *Viewa*. Omogućeno je i kretanje između vožnji – postoji dva gumba kojima se korisnik kreće na slijedeću vožnju ili na prethodnu vožnju u ovisnosti o trenutnom prikazu obavljene vožnje.

Ukoliko ne postoje zapisi trenutno izabrane vožnje, korisnik se obavještava na način da nema podataka o početku i kraju vožnje, kao ni komentara vožnje ni lokacija na mapi.



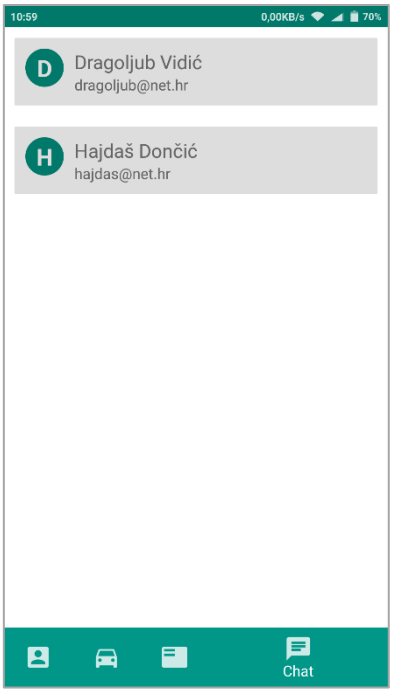
Sl. 3.13. Pregled prethodnih vožnji

3.5. Dio aplikacije namijenjen instruktorima

Instruktori u Android aplikaciji imaju pristup vlastitim informacijama, informacijama autoškole, pregledu prethodno završenih vožnji za pojedinog kandidata, razgovor s kandidatima koji još uvijek obavljaju satove vožnje i koji još uvijek nisu položili ispit vožnje. Također, imaju mogućnost započeti novu vožnju skeniranjem pripadajućeg QR koda kandidata. Dohvaćanje i prikaz informacija o instruktoru i autoškoli je isti kao i u dijelu aplikacije namijenjen polaznicima vožnje samo s izmijenjenim sadržajem. Također, na isti način kao i u dijelu aplikacije namijenjen polaznicima vožnje definiran je i izbornik *Bottom navigation*, ali s izmijenjenim elementima.

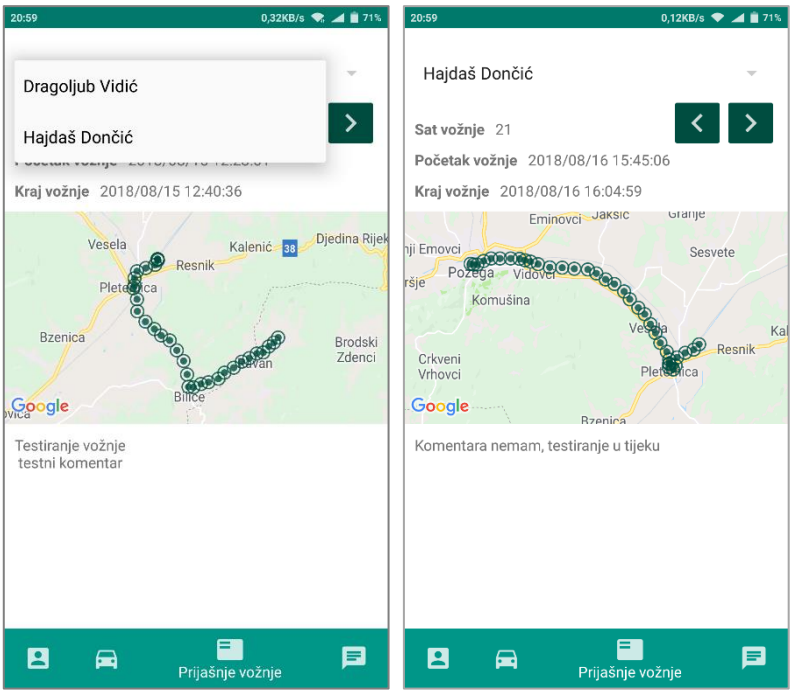
Kod odabira elementa za chat, prikazuje se lista svih kandidata prijavljenog instruktora. Napravljen je adapter pomoću kojega je definiran prikaz i sadržaj pojedinog elementa liste, odnosno *RecyclerViewa*. *DatabaseInteractor* obavlja posao dohvata pripadajućih kandidata iz baze podataka koji zadovoljavaju uvjete da im je dodijeljeni instruktor upravo instruktor koji je prijavljen u aplikaciju te da još uvijek nisu položili ispit vožnje, odnosno da im obuka vožnje još uvijek traje. Tako dobiveni kandidati se spremaju u *ArrayListu* te se svaki od njih preko definiranog adaptera raspoređuju kao elementi *RecyclerViewa*. (Slika 3.13.) Klikom na bilo koji

od kandidata, provjerava se njegova pozicija u *RecyclerViewu* te se na osnovu te pozicije pokreće *MessageActivity* (Slika 3.12.) koji unutar *companion objecta* ima funkciju *getLaunchIntent* čiji su elementi identifikacijski broj i naziv instruktora i kandidata.



Sl. 3.14. Lista kandidata

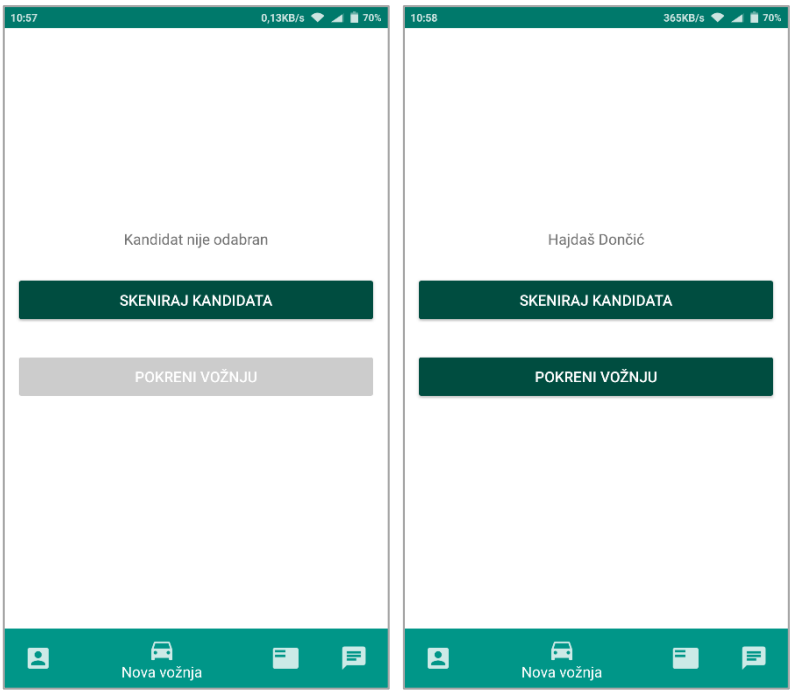
Pregled prethodnih vožnji je sličan kao i kod pregleda vožnji u dijelu aplikacije namijenjenom kandidatima. Promjena je u prikazu kandidata – instruktor ima nekoliko kandidata te mora postojati mogućnost uvida u detalje vožnje za svakoga od njih. Zbog toga je instruktoru dostupan padajući izbornik, odnosno *spinner*, u kojem su svi kandidati čiji je odabrani instruktor istovjetan trenutno prijavljenom instruktoru.



Sl. 3.15. Prikaz detalja prethodnih vožnji

DatabaseInteractor je zadužen za dohvaćanje kandidata iz baze podataka koji zadovoljavaju uvjete. Unutar *Presentera* je definiran *spinner* u kojeg se spremaju prethodno dohvaćeni kandidati, odnosno njihova imena. U ovisnosti o odabranom kandidatu, prikazuje se njegova zadnja upisana vožnja. Prethodne vožnje su dostupne klikom na gumb prethodno.

Instruktori imaju mogućnost pokretanja nove vožnje. Vožnju mogu pokrenuti samo nakon skeniranja pravovaljanog QR koda generiranog svakom kandidatu prilikom njihove registracije u sustav, odnosno pri njihovom upisu u autoškolu. Odabirom elementa *Nova vožnja* u izborniku aplikacije pokreće se fragment koji ima dva gumba – prvi služi za pokretanje kamere u svrhu skeniranja kandidata, a drugi služi za pokretanje vožnje. Gumb za pokretanje vožnje je onemogućen sve dok nije skeniran pravovaljani QR kod kandidata – kandidatov instruktor mora biti istovjetan trenutno prijavljenom instruktoru u sustavu, te zapis o trenutnom stanju i prolaznosti ispita vožnje kandidata mora biti negativan. U slučaju skeniranja nepravilnog QR koda, aplikacija javlja grešku.



Slika 3.16. Pokretanje nove vožnje

Za skeniranje QR kodova korišten je *Googleov* API alat *Barcode Scanning* iz skupine alata pod nazivom *Machine learning for mobile developers (ML Kit)*. [11] API za skeniranje barkodova omogućuje čitanje podataka kodiranih pomoću standardnih formata crtičnih kodova – QR, PDF417, CodeBar, Code 39, Aztec, EAN-8 i drugih.

Potrebno je dodati *ml-vision* biblioteku u *build.gradle* kako bi se API mogao koristiti. Također, potrebno je provjeriti je li korisnik dao dopuštenje aplikaciji za pristup i korištenje kameri. Ako korisnik nije dao dopuštenje aplikaciji za pristup i korištenje kamere uređaja, prikazuje se dijalog u kojem se korisnika obavještava i traži za davanje dopuštenja kako bi aplikacija mogla nesmetano raditi i biti funkcionalna u cijelosti. To je standardna procedura u svim aplikacijama koje zahtijevaju upotrebu kamere ili nekog drugog perifernog dijela uređaja. Nakon toga, može se pristupiti kreiranju čitača QR koda. QR kod unutar sustava *E – autoškole* sadrži jedinstveni identifikator korisnika – kandidata ili instruktora – koji je generiran pri registraciji korisnika u sustav. Prvo je potrebno definirati čitač, te u ovisnosti o skeniranom QR kodu poduzimaju se određene radnje.

```
private fun setReader() {
    barcodeDetector = BarcodeDetector.Builder(this)
            .setBarcodeFormats(Barcode.QR_CODE)
            .build()

    cameraSource = CameraSource.Builder(this, barcodeDetector)
            .setFacing(CameraSource.CAMERA_FACING_BACK)
```

```
            .setRequestedPreviewSize(1600, 1024)
            .setAutoFocusEnabled(true)
            .setRequestedFps(24.0f)
            .build()
    startReading()
}

private fun startReading() {
    cameraView.holder.addCallback(object : SurfaceHolder.Callback {
        override fun surfaceCreated(holder: SurfaceHolder) {
            try {
                cameraSource?.start(cameraView.holder)
            } catch (ie: IOException) {
                Log.e("CAMERA SOURCE", ie.toString())
            } catch (securityException: SecurityException) {
                Log.e("CAMERA SOURCE", securityException.message)
            }
        }

        override fun surfaceChanged(holder: SurfaceHolder,
               format: Int, width: Int, height: Int) {}

        override fun surfaceDestroyed(holder: SurfaceHolder) {
            cameraSource?.stop()
        }
    })
    handleDetections()
}

private fun handleDetections() {
    barcodeDetector?.setProcessor(object : Detector.Processor<Barcode> {

        override fun release() {}

        override fun receiveDetections(
         detections: Detector.Detections<Barcode>) {
            presenter.handleDetections(detections.detectedItems)
        }
    })
}
```

Programski kod 3.14. Definiranje čitaća QR koda

Unutar *Presentera* je definirana metoda *handleDetections*, koja u ovisnosti o prepoznatom QR kodu poziva *DatabaseInteractor* koji provjerava postoji li u bazi podataka korisnik čiji jedinstveni identifikator odgovara podatku iz QR koda.

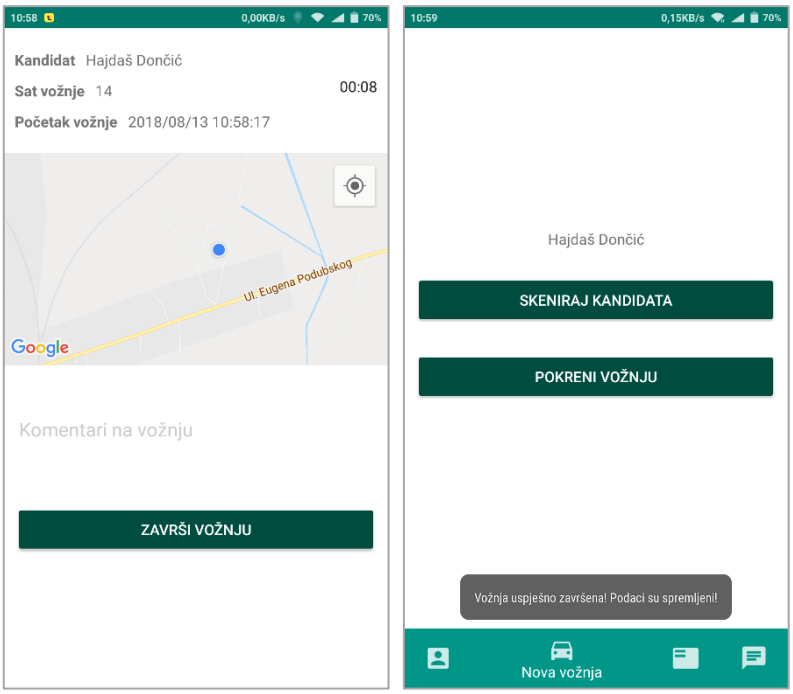
```
override fun handleDetections(barcodeDetector: SparseArray<Barcode>) {

    if (barcodeDetector.size() != 0) {
        view.stopCamera()
        databaseInteractor.getCandidateNAme(barcodeDetector.valueAt(0)
        .displayValue)
    }
}
```

Programski kod 3.15. Provjera skeniranog QR koda

```
override fun getCandidateNAme(candidateId: String) {

    if (candidateId.contains(".")) {
        databaseListener.returnDatabaseError()
    } else {
        val candidateData = object : ValueEventListener {
            override fun onDataChange(dataSnapshot: DataSnapshot) {

                val candidateName = dataSnapshot.child(NAME).value.toString()
                val candidateRole = dataSnapshot.child(ROLE).value.toString()
                if (candidateName != "null" && candidateRole == CANDIDATE) {
                    databaseListener.returnCandidateIdAndName(
                       candidateId, candidateName)
                } else {
                    databaseListener.returnDatabaseError()
                }
            }

            override fun onCancelled(databaseError: DatabaseError) {}
        }
        myRef.child(USERS).child(candidateId)
        .addListenerForSingleValueEvent(candidateData)
    }
}
```

Programski kod 3.16. Dohvaćanje podataka iz baze na osnovu skeniranog QR koda

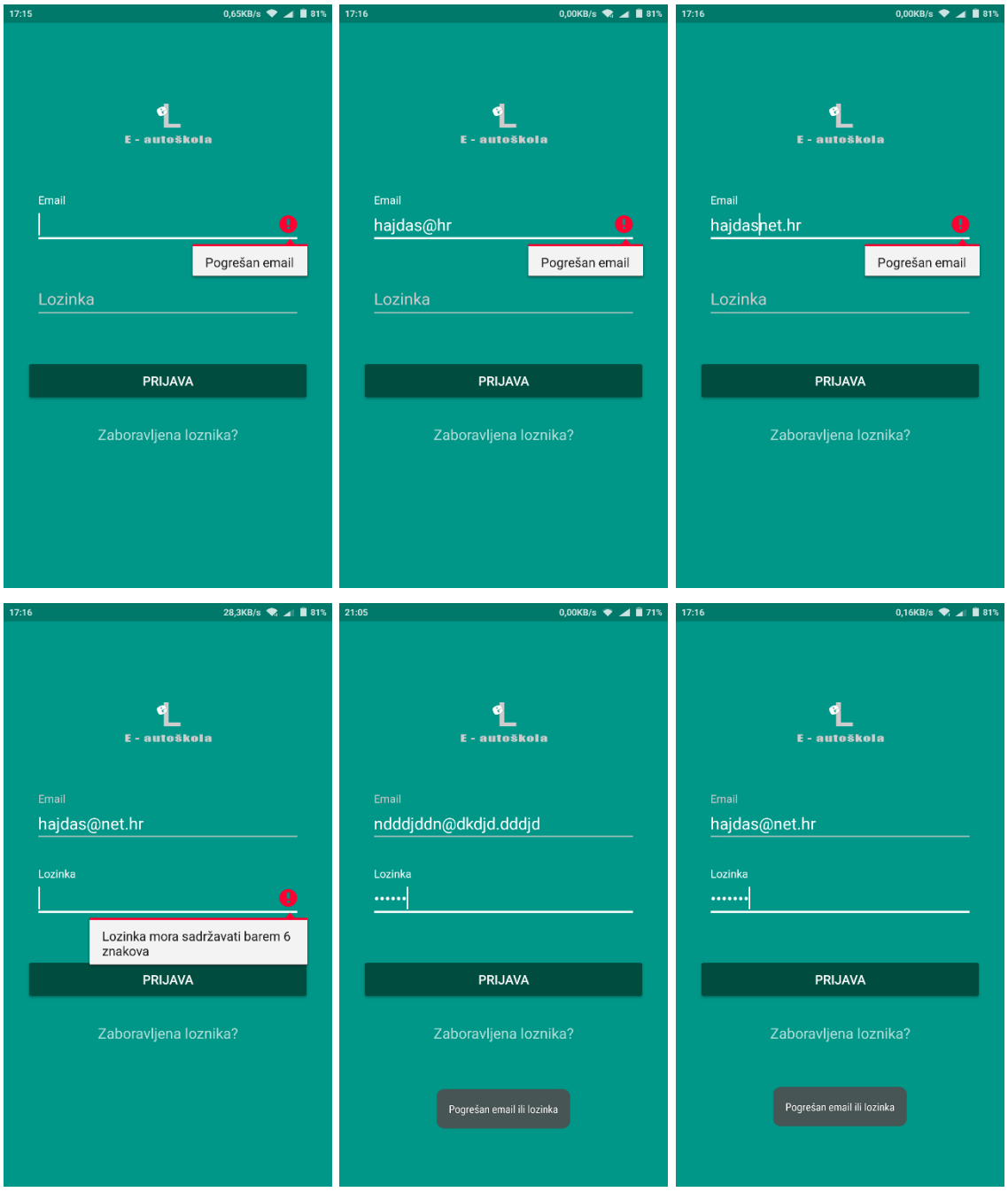
Ako je skenirani QR kod valjan i ako u bazi podataka postoji pripadajući kandidat, zatvara se kamera i skenirani podatak se prosljeđuje fragmentu za pokretanje nove vožnje. Prikazuje se ime skeniranog kandidata te gumb Pokreni vožnju postaje aktivan za odabir. Odabirom gumba za pokretanje vožnje pokreće se nova aktivnost – *RideActivity* kojoj se prosljeđuju podaci o skeniranom kandidatu. Pokretanjem aktivnosti, pokreće se i *Chronometer* koji prikazuje proteklo vrijeme od pokretanja vožnje. Prikazuju se i podaci o trenutnom satu vožnje, o vremenu početka vožnje te informacije o trenutnom položaju koje su prikazane na *Google Map* fragmentu. Trenutna lokacija se uzorkuje svakih 30 sekundi i sprema u bazu. Vrši se spremanje zemljopisne širine (*latitude*), zemljopisne dužine (*longitude*) te vrijeme uzorkovanja. U aktivnosti je onemogućen povratak unazad – *onBackPressed()* zbog toga što bi to značilo naprasno prekinutu vožnju. Omogućen je i unos komentara prije završetka vožnje, odnosno prije odabira gumba *Završi vožnju*. Komentar je obavezan kako bi se uopće mogla završiti vožnja.



Sl. 3.17. Pokrenuta vožnja i uspješno završena vožnja

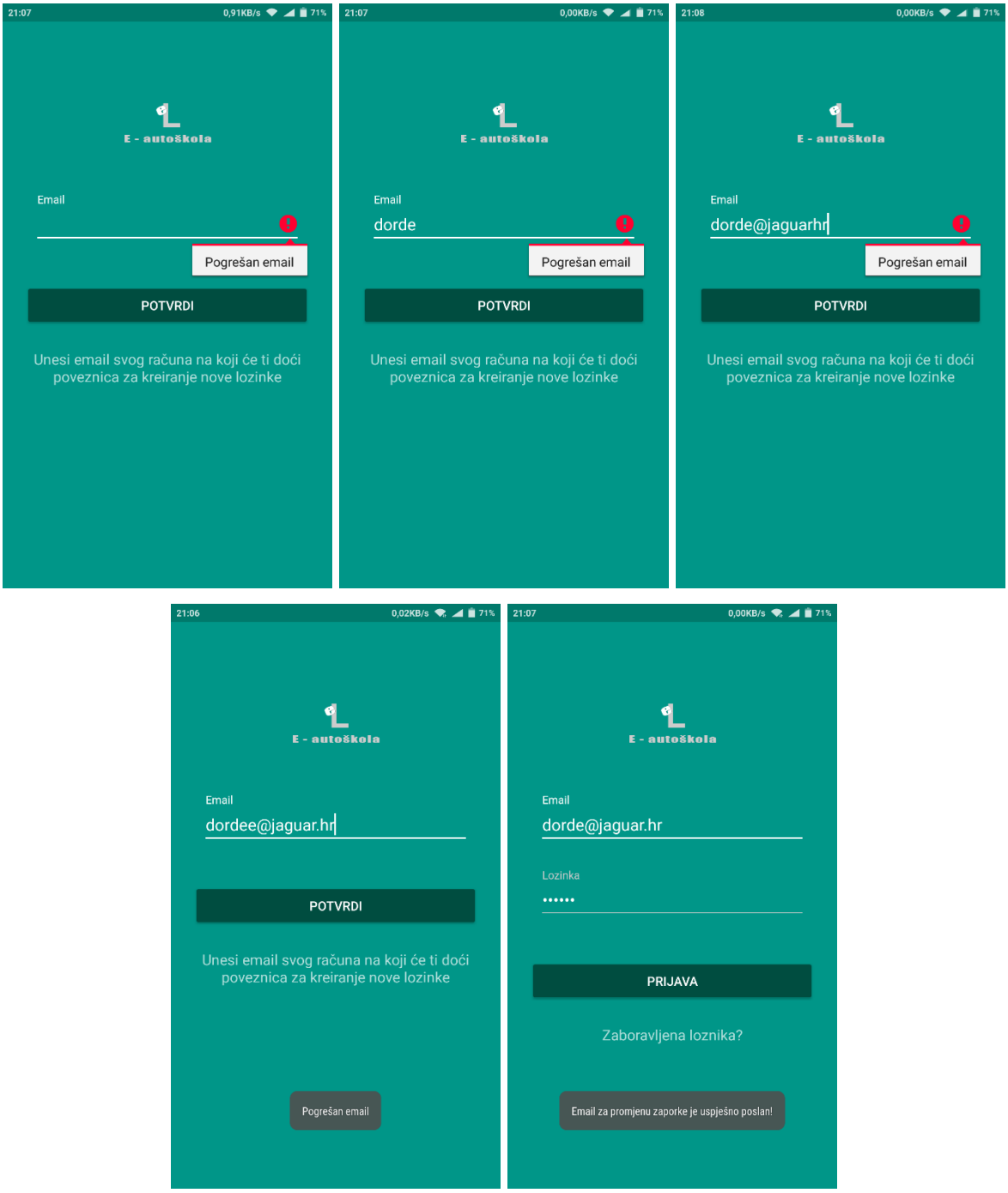
4. TESTIRANJE I REZULTATI

Testiranje aplikacije je obavljeno korištenjem same aplikacije u radnom okruženju pri različitim ponašanjima korisnika koje bi mogle dovesti do stvaranje greške u radu aplikacije i rušenja iste. Kod prijave u sustav provjerava se upisani email i lozinka. Aplikacija javlja pogrešku u slučaju nepravilno upisanog emaila ili lozinke, te u slučaju ostavljenih praznih polja ili nepravilno formiranog emaila. Osigurani su svi mogući slučajevi unosa nepravilnih podataka korisnika uz pripadajuće obavijesti o greškama.



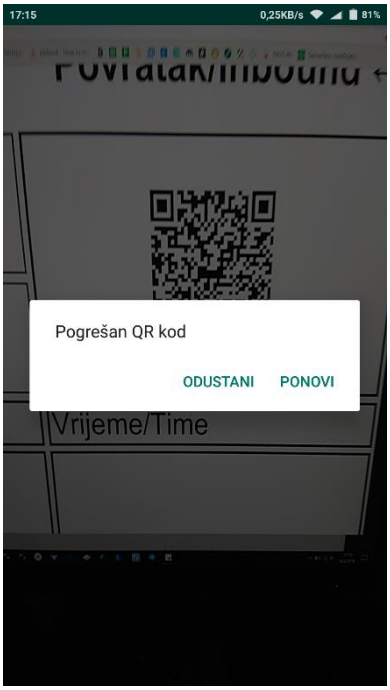
Sl. 4.1. Provjera upisa podataka prilikom prijave u sustav

U slučaju zaboravljene lozinke, od korisnika se zahtjeva unos emaila preko kojega mu je kreiran račun u sustavu. Ukoliko uneseni email ne zadovoljava formom, ukoliko je polje za unos prazno, ili ukoliko je uneseni email formom ispravan ali se ne nalazi u bazi registriranih korisnika, aplikacija obavještava korisnika o greški. Na taj način je osigurano pravilno djelovanje i rad u svim mogućim slučajevima unosa podataka korisnika.



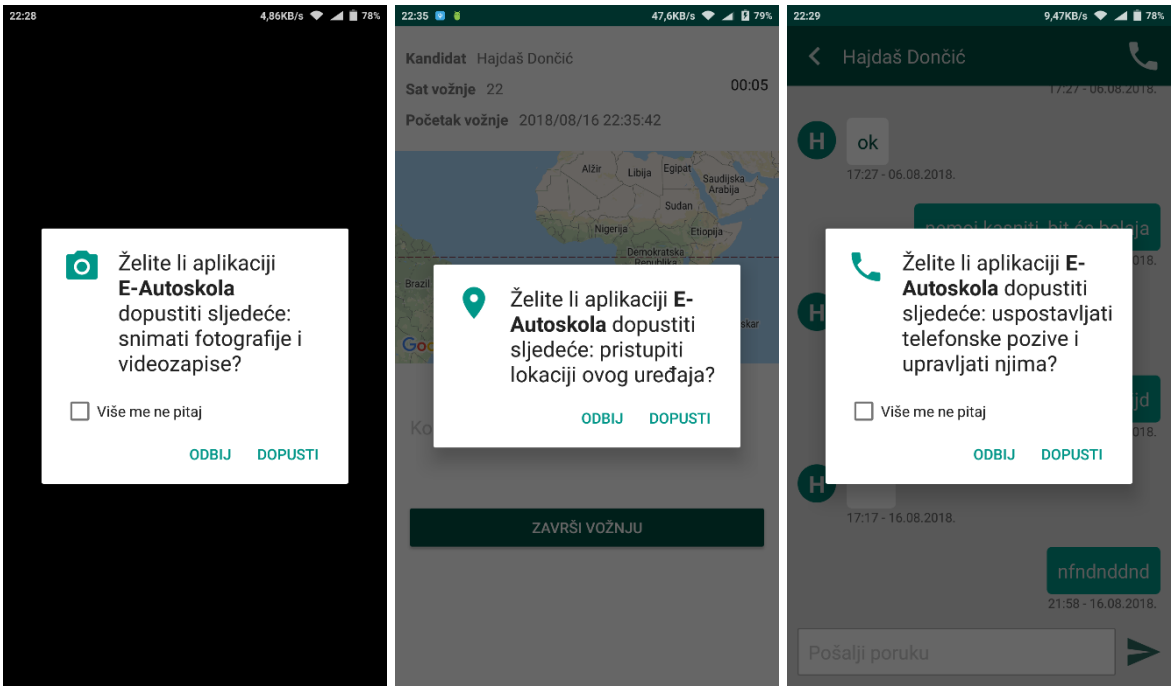
Sl. 4.2. Provjera upisa emaila za oporavak zaboravljene lozinke

Pri skeniranju QR koda provjerava se sadrži li pročitani podatak iz QR koda točku. Ako sadrži, znači da nije točan i da nije valjan za aplikaciju (u velikom broju slučajeva QR kodovi zapravo sadrže poveznicu na neku web stranicu). Ako ne sadrži točku, pristupa se traženju u bazi. Traže se podčvorovi čvora *users* i uspoređuju s dobivenim podatkom iz QR koda. Ukoliko ne postoji podčvor čiji je jedinstveni identifikator jednak dobivenom podatku iz QR koda, aplikacija javlja grešku o nepravilnom QR kodu. U slučaju pozitivnog rezultata usporedbe, aplikacija nastavlja daljnji rutinski rad.



Sl. 4.3. Skeniranje pogrešnog QR koda

Aplikacija traži od korisnika nekoliko dopuštenja – dopuštenje za pristup lokaciji uređaja, dopuštenje za pristup kameri uređaja te dopuštenje za pristup uspostavljanju telefonskih poziva i njihovim upravljanjem. U slučaju prijavljenog instruktora, bit će potrebna potvrda sva tri dopuštenja. A u slučaju prijavljenog kandidata bit će potrebna potvrda samo dopuštenja za pristup pozivima uređaja. Zahtjevi za određena dopuštanja se prikazuju netom prije potrebe korištenja određenih mogućnosti uređaja i samo u slučaju da prethodno korisnik nije dao dopuštenje za određenu mogućnost uređaja.



Sl. 4.4. Zahtjevi za dopuštenja aplikaciji

5. ZAKLJUČAK

Android aplikacija je dio cjelokupnog sustava *E – autoškola* koji se još sastoji od web dijela čiji su glavni korisnici administratori sustava te administratori pojedinih autoškola. Web dio sustava je početni korak kod registracije novog kandidata. Administratori sustava registriraju autoškole i pripadajuće administratore tih autoškola. Daljnji korak jest registracija instruktora i novih kandidata u sustav. Taj dio posla obavljaju administratori autoškola. Pri registraciji instruktora i kandidata, prikupljaju se svi potrebni podaci relevantni za pojedine tipove korisnika. Korištenje same Android aplikacije je omogućeno isključivo instruktorima vožnje i kandidatima. Kako bi se mogla koristiti Android aplikacija omogućena je prijava pomoću emaila i lozinke. Ukoliko korisnik zaboravi lozinku, omogućen je i oporavak, odnosno postavljanje nove lozinke.

Nakon uspješne prijave, u ovisnosti o vrsti korisničkog računa – instruktor ili kandidat – vrši se prikaz pripadajućih aktivnosti i izbornika. Kandidati imaju mogućnost pregleda njima relevantnih informacija – vlastite podatke, informacije upisane autoškole, te informacije o dodijeljenom instruktoru. Također, imaju mogućnost pregleda svih odvoženih sati vožnje. Prikazuju se podaci po rednim brojevima vožnje, odnosno, rednom broju sata vožnje. Kandidat dobiva informacije o početku vožnje, kraju vožnje, rednom broju vožnje, komentarima vožnje od strane instruktora te prikaz kretanja vozila za vrijeme vožnje. Kandidatima je omogućeno slanje poruka dodijeljenom instruktoru. Unutar aktivnosti za slanje poruka, jednim klikom je omogućen i poziv instruktoru na broj telefona koji je unesen u bazu podataka za tog instruktora.

U slučaju prijave preko korisničkog računa instruktora, prikazuju se informacije o instruktoru, te autoškoli u kojoj je instruktor zaposlen. Pri pokretanju nove vožnje, prvenstveno je potrebno skeniranje QR kod kandidata. U slučaju skeniranja valjanog QR koda omogućeno je pokretanje nove vožnje. Instruktori imaju uvid u sve prethodne vožnje dodijeljenih kandidata koji još nisu položili ispit vožnje, odnosno, koji još uvijek nisu dobili status vozača. Omogućen je odabir određenog kandidata i određenog sata vožnje kako bi se dobio uvid u tražene podatke. Također, kao i kandidati, instruktori imaju mogućnost razmjene poruka s dodijeljenim kandidatima, te pozive prema njima.

Aplikacija za autoškolu je aplikacija čija je svrha zamjena tradicionalne papirnate Knjižice kandidata za vozača. Omogućuje brz i jednostavan pristup podacima i potrebnim informacijama. Aplikacija kao takva ima potencijala za veliki broj autoškola, koje bi uporabom cijelog sustava *E*

– *autoškola* bile u mogućnosti rasteretiti administratore od višestrukog upisa istih podataka u odvojene baze podataka. Sustav bi riješio samo digitalni dio unosa, rukovanja i pristupa podacima, jer prema zakonu u Republici Hrvatskoj moraju se voditi i ručno pisane baze podataka. Postoji mogućnost implementiranja i sinkronizacije baze podataka sustava *E – autoškola* s bazom podataka HAK-a. Na taj način bi postojala jedinstvena baza podataka i jedinstveni pristup podacima. Neke autoškole imaju vlastite dizajnirane sustave, a neke autoškole koriste samo obveznu matičnu knjigu od strane HAK-a te ostale potrebne podatke vode samo u ručno pisanom obliku. Sustav *E – autoškola* bi koristio i jednim i drugim autoškolama jer doradom i implementacijom dodatnih funkcionalnosti postojala bi jedinstvena baza kandidata i osposobljenih vozača. U sustav je moguće implementirati podatke o prijeđenim kilometrima i potrošenom gorivu svakog instruktora, te ne bi bilo potrebno ručno upisivanje mjesečnih izvoda i detalja. Također, dodatnom implementacijom, bilo bi omogućen i pristup informacijama vožnje te komentarima instruktora i roditeljima kandidata jer u velikom broju slučajeva, oni upravo i financiraju vozački ispit. Uz sveprisutne najave digitalizacije svih grana sustava Republike Hrvatske, te moguću izmjenu zakona gdje ne bi bilo potrebe o ručnoj pisanoj bazi kandidata, sustav *E – autoškola* u kombinaciji s obveznom matičnom knjigom HAK-a nudi jednostavan, siguran i kvalitetan unos, pregled te izmjene podataka u autoškolama.

LITERATURA

[1] Pravilnik o osposobljavanju kandidata za vozače, https://narodne-novine.nn.hr/clanci/sluzbeni/2017_12_132_3037.html (14.08.2018.)

[2] Kotlin Programming Language, https://kotlinlang.org/ (08.08.2018.)

[3] Android Architecture Patterns – Model-View-Presenter, https://medium.com/upday-devs/android-architecture-patterns-part-2-model-view-presenter-_8a6faaae14a5 (08.08.2018.)

[4] Firebase, https://firebase.google.com/ (08.08.2018.)

[5] Firebase Realtime Database, https://firebase.google.com/products/realtime-database/ (08.08.2018.)

[6] Add Firebase to your Project, https://firebase.google.com/docs/android/setup (08.08.2018.)

[7] Firebase Authentication, https://firebase.google.com/docs/auth/ (09.08.2018.)

[8] Bootstrap, https://getbootstrap.com/ (12.08.2018.)

[9] Bottom navigation – Material Design, https://material.io/design/components/bottom-navigation.html (13.08.2018.)

[10] Current Millis – Milliseconds since Epoch, https://currentmillis.com/ (14.08.2018.)

[11] Barcode Scanning, https://firebase.google.com/docs/ml-kit/read-barcodes (14.08.2018.)

SAŽETAK

Android aplikacija za autoškolu namijenjena je autoškolama koje žele digitalizirati podatke o upisanim kandidatima te omogućiti jednostavan i brz pregled detalja vožnje i informacija o kandidatima, instruktorima i autoškoli. Namijenjena je instruktorima i kandidatima, te prvenstveno zamjenjuje Knjižicu kandidata za vozača. Uz detaljan prikaz podatka, aplikacija omogućuje čitanje i zapisivanje informacija o obavljenim vožnjama, njihov prikaz te slanje poruka između instruktora i kandidata . Android aplikacija za autoškolu dio je sustava *E – autoškola* koji se sastoji još od web sustava čiji su korisnici administratori sustava *E – autoškole* i administratori autoškola. Administratori sustava imaju mogućnost registrirati nove autoškole i nove administratore autoškola. Administratori autoškola imaju mogućnost registriranja instruktora i kandidata.

Ključne riječi: Android, Kotlin, Firebase, MVP, autoškola

ABSTRACT

The Android application for driving school is designed for driving schools that wants to digitize data about candidates and provide a quick and easy overview of driving details and information about candidates, instructors and school. The application is intended for instructors and candidates and replaces a booklet for new driver. With a detailed view of the data, the application allows chat between instructors and candidates. The application is part of the *E - Driving Schools* system, which consists of a web system whose users are system administrators of *E - Driving Schools* and administrators of each registered driving school. System administrators can register new driving schools and new administrators of driving school. Administrators of driving schools can register instructors and candidates.

Key words: Android, Kotlin, Firebase, MVP, driving school

ŽIVOTOPIS

Tomislav Safundžić rođen je 3. svibnja 1994. godine u Požegi. Od rođenja živi u Pleternici. Osnovnu školu završio je u Pleternici. Školovanje nastavlja upisom u Tehničku školu u Požegi, gdje 2013. godine stječe kvalifikaciju tehničara za računalstvo. Iste godine upisuje preddiplomski sveučilišni studij računarstva na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija u Osijeku. 2016. godine stječe akademski naziv Sveučilišni prvostupnik (*baccalaureus*) inženjer računarstva. Nastavlja obrazovanje upisom diplomskog studija računarstva na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija u Osijeku. 2018. obavlja stručnu praksu u tvrtki COBE, Hrvatske Republike 33, Osijek. Stručna praksa je sadržavala razvoj Android aplikacija korištenjem Jave i Kotlina, te upotrebu MVP arhitekture. U slobodno vrijeme aktivno se bavi fotografijom te radi kao *freelancer* fotograf i snimatelj.

_____

Tomislav Safundžić

PRILOZI

1. Git repozitorij cijelog projekta:
   https://github.com/tsafundzic/E_Autoskola.git