Mobilna aplikacija za provjeru znanja pripravnika

Perić, Dalibor

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

Sveučilišni studij

MOBILNA APLIKACIJA ZA PROVJERU ZNANJA PRIPRAVNIKA

Diplomski rad

Dalibor Perić

Osijek, 2021.

ZAHVALA

Zahvaljujem se svojoj obitelji, supruzi Josipi i kćerki Lauri što su mi bili najveća podrška tijekom moga studija bez kojih uspjeh nebi bio potpun.

Zahvaljujem se i svojim roditeljima Miji i Kati, koji su mi kroz život omogućili školovanje, te također bili velika podrška tijekom moga studija. Zahvaljujem se i sestrama Dajani i Dariji na pruženoj podršci tijekom moga studija.

Takoder, zahvaljujem se svome mentoru Alfonsu na pruženoj pomoći i savjetima tijekom izrade ovoga rada, kao i svim prijateljima i kolegama koji su mi na neki način pomogli tijekom studija.

Ovaj rad posvećujem svojoj voljenoj kćerki Lauri.

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

Obrazac D1: Obrazac za imenovanje Povjerenstva za diplomski ispit

Osijek, 18.02.2021.

Odboru za završne i diplomske ispite

Imenovanje Povjerenstva za diplomski ispit

Ime i prezime studenta:	Dalibor Perić
Studij, smjer:	Diplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	D 856 R, 07.11.2016.
OIB studenta:	40428438090
Mentor:	Izv. prof. dr. sc. Alfonzo Baumgartner
Sumentor:	
Sumentor iz tvrtke:	
Predsjednik Povjerenstva:	Izv.prof.dr.sc. Tomislav Keser
Član Povjerenstva 1:	Izv. prof. dr. sc. Alfonzo Baumgartner
Član Povjerenstva 2:	Dr. sc. Tomislav Galba
Naslov diplomskog rada:	Mobilna aplikacija za provjeru znanja pripravnika
Znanstvena grana rada:	**Informacijski sustavi (zn. polje računarstvo)**
Zadatak diplomskog rada:	Potrebno je napraviti mobilnu aplikaciju koja će imati bazu podataka s pitanjima za provjeru znanja pripravnika iz područja elektrothenike. Posebno napraviti administratorsko sučelje u kojem se mogu zadavati pitanja i testovi, te pratiti rezultati svih korisnika. Korisničko sučelje treba imati izvršavanje testova i praćenje samo vlastitih rezultata. Po mogućnosti koristiti što suvremeniji programski jezik za mobilne aplikacije.
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:	18.02.2021.

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, 26.02.2021.

Ime i prezime studenta:	Dalibor Perić
Studij:	Diplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	D 856 R, 07.11.2016.
Turnitin podudaranje [%]:	10

Ovom izjavom izjavljujem da je rad pod nazivom**: Mobilna aplikacija za provjeru znanja pripravnika**

izrađen pod vodstvom mentora Izv. prof. dr. sc. Alfonzo Baumgartner

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:

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

IZJAVA

o odobrenju za pohranu i objavu ocjenskog rada

kojom ja __Dalibor Perić_____, OIB: __40428438090_____, student/ica Fakulteta elektrotehnike, računarstva i informacijskih tehnologija Osijek na studiju _Diplomski sveučilišni studij Računarstvo_, kao autor/ica ocjenskog rada pod naslovom: _Mobilna aplikacija za provjeru znanja pripravnika**,**

dajem odobrenje da se, bez naknade, trajno pohrani moj ocjenski rad u javno dostupnom digitalnom repozitoriju ustanove Fakulteta elektrotehnike, računarstva i informacijskih tehnologija Osijek i Sveučilišta te u javnoj internetskoj bazi radova Nacionalne i sveučilišne knjižnice u Zagrebu, sukladno obvezi iz odredbe članka 83. stavka 11. *Zakona o znanstvenoj djelatnosti i visokom obrazovanju* (NN 123/03, 198/03, 105/04, 174/04, 02/07, 46/07, 45/09, 63/11, 94/13, 139/13, 101/14, 60/15).

Potvrđujem da je za pohranu dostavljena završna verzija obranjenog i dovršenog ocjenskog rada. Ovom izjavom, kao autor/ica ocjenskog rada dajem odobrenje i da se moj ocjenski rad, bez naknade, trajno javno objavi i besplatno učini dostupnim:

a) široj javnosti

b) studentima/icama i djelatnicima/ama ustanove

c) široj javnosti, ali nakon proteka 6 / 12 / 24 mjeseci (zaokružite odgovarajući broj mjeseci).

**U slučaju potrebe dodatnog ograničavanja pristupa Vašem ocjenskom radu, podnosi se obrazloženi zahtjev nadležnom tijelu Ustanove.*

__Osijek, 26.02.2021._____ _____

(mjesto i datum) (vlastoručni potpis studenta/ice)

Sadržaj:

1. UVOD

Mobilna aplikacija je aplikacija koja se koristi na pametnim telefonima, teblet računalima ili nekakvim drugim mobilnim uređajima. Mobilne aplikacije mogu se koristiti na mobilnim uređajima koji rade Android i IOS sustavima. Sustavi za izradu mobilnih aplikacija konstantno napreduju i poboljšavaju se, stalno predstavljaju nove mogućnosti. Mogućnosti mobilne aplikacije ovise o svrsi za koju su izrađene. S obzirom na prirodu posla, dio posla, koji obuhvaća obuku i provjeru znanja pripravnika stvorio je ideju o izradi mobilne aplikacije koja će omogućiti bržu i jednostavniju provjeru naučenog znanja pripravnika.

Rad se sastoji od više poglavlja, s pripadajućim potpoglavljima. Glavni dio diplomskog rada sadrži tri poglavlja:

- Teorija i tehnologije, poglavlje koje opisuje korištene alate i programsku podršku,
- Izrada aplikacije, poglavlje koje sadrži detaljan pregled strukture rješenja, te opis kodova bitnijih dijelova mobilne aplikacije,
- Izgled aplikacije, poglavlje koje opisuje način korištenja mobilne aplikacije

1.1. Zadatak diplomskog rada

Zadatak ovog diplomskog rada je izraditi mobilnu aplikaciju koja će imati bazu podataka s pitanjima za provjeru znanja pripravnika iz područja elektrotehnike. Posebno napraviti administratorsko sučelje u kojem se mogu zadavati pitanja i testovi, te pratiti rezultati svih korisnika. Korisničko sučelje treba imati izvršavanje testova i praćenje samo vlastitih rezultata. Po mogućnosti koristiti što suvremeniji programski jezik za mobilne aplikacije.

2. TEORIJA TEHNOLOGIJA

U ovom poglavlju navedeni su i ukratko opisani korišteni alati i programska podrška potrebni za izradu mobilne aplikacije i pripadajuće baze podataka.

2.1. Alati i programska podrška

Za izradu mobilne aplikacije i pripadajuće baze podataka koriste se sljedeći alati i programska podrška.

MySQL

MySQL je RDBMS otvorenog koda. RDBMS punog naziva *Relational Database Managment System* u prijevodu znači relacijski sustav za upravljanje bazama podataka. MySQL su programi koji pružaju sustav baze podataka pomoću kojih se mogu povlačiti i spremati velike količine informacija te ih relacijski pohraniti. MySQL omogućava izradu baze podataka koje kontrolora SQL programski jezik. *Structured Query Language* u punom nazivu znači strukturni upitni jezik. SQL programski jezik pomoću kojeg se vrše radnje na MySQL relacijskim bazama podataka. [1]

Navicat Premium

Navicat Premium je alat za razvoj baza podataka koji istovremeno omogućuje povezivanje sa MySQL, MariaDB, MongoDB, SQL Server, Oracle, PostgreSQL, i SQLite bazama podataka iz jedne aplikacije. Omogućuje jednostavno stvaranje, upravljanje i održavanje kreirane baze podataka. [2]

XAMPP

XAMPP puni naziv je *Cross-platform apache mysql php perl*, a predstavlja serverski paket kojim se pokreće Apache, MySQL, PHP i phpMyAdmin koja korisniku daje potprograme i komponente potrebne za stvaranje lokalnog servera, testiranje i razvoj web okoline. [3]

Ionic

Ionic Framework je programski okvir pomoću kojeg se razvijaju hibridne aplikacije temeljene na HTML-u, CSS-u i JavaScrip-u. Pruža set gotovih funkcionalnosti koje su podržane na više platformi zbog čega i dolazi pridjev „hibridni“, odnosno nije potrebno razvijati posebnu aplikaciju za svaku platformu. [4]

Angular

Angular je strukturiran okvir napravljen u JavaScript okviru za izradu i razvoj klijentskih dinamičkih aplikacija. Angular je kompatibilan sa korištenjem HTML i CSS-a te ih nadograđuje proširujući njihove mogućnosti. Angular strukturni okvir napisan je u programskom jeziku TypeScript kojim su se nadomjestili nedostatci JavaScript-a. [5]

Node.js

Node.js je serverska JavaScript platforma koja služi za izradu aplikacija za Internet te pruža siguran i lak način izgradnje mrežnih aplikacija visoke performanse. [6]

JavaScript

JavaScript je skriptni jezik pomoću kojeg se u statičke HTML stanice uvode interaktivni elementi, odnosno on pruža korisniku interakciju, mogućnost promjene svojstva ili dinamičko stvaranje HTML sadržaja. On nosi naziv i „klijentski jezik“ jer se njegov kod interpretira na klijentskoj strani odnosno u pregledniku. [7]

TypeScript

TypeScript je objektno orijentiran jezik koji uvodi koncept objektno orijentiranog programiranja u neobjektno orijentirani jezik, JavaScript. TypeScript pruža mogućnost bogatijeg razvojnog okruženja te podržava pisanje dugog i dobro strukturiranog koda s mogućnošću prevođenja u običan JavaScript koji se može pokrenuti na bilo kojem operacijskom sustavu, poslužitelju ili pregledniku. [8]

HTML

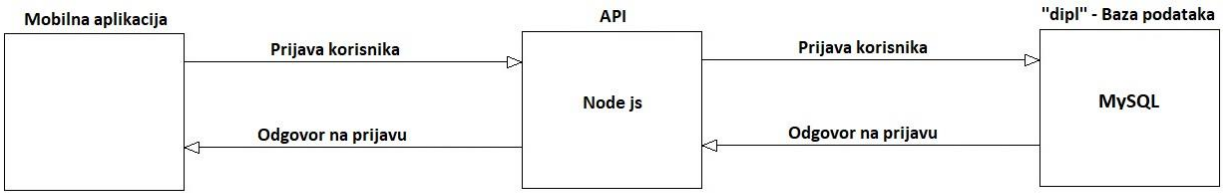
HTML je prezentacijski jezik, punog naziva HyperText Markup Language što u prijevodu znači prezentacijsko označavanje hipertekstualnih dokumenata. HTML je prije svega jednostavan jezik kojim se definira struktura i oblikuje sadržaj odnosno stvara hipertekst dokument na kojeg se stvaraju hiperveze dok prikaz hiperteksta omogućuje web preglednik. HTML je zapravo set prezentacijskih uputa usmjerenih web pregledniku kako određeni sadržaj treba izgledati bez obzira o kojoj vrsti preglednika govorimo. [9]

CSS

CSS je style sheet programski jezik, punog naziva *eng. Cascading Style Sheets*. Riječ *Cascading* označava kaskadnu primjenu CSS pravila dok se riječi *Style Sheets* odnose na datoteku koja definira stil, odnosno izgled web stranice. Osnovna temeljna tehnologija weba kakvog ga poznajemo danas je zapravo CSS koji služi za oblikovanje web stranica, odnosno on govori web pregledniku kako web stranica izgleda te kako će se detaljno formatirati dokument koji je napisan u markup programskom jeziku HTML s kojim se zapravo definira struktura i sadržaj. [10]

3. IZRADA APLIKACIJE

Ovo poglavlje opisuje tijek izrade Mobilne aplikacije, dizajn baze podataka koju aplikacija koristi te sam način izrade Mobilne aplikacije. Blok dijagram prikazan na slici 3.1. prikazuje relacijske veze između mobilne aplikacije i baze podataka kroz node server, a samim time i funkcionalnost njihovog zajedničkog rada.



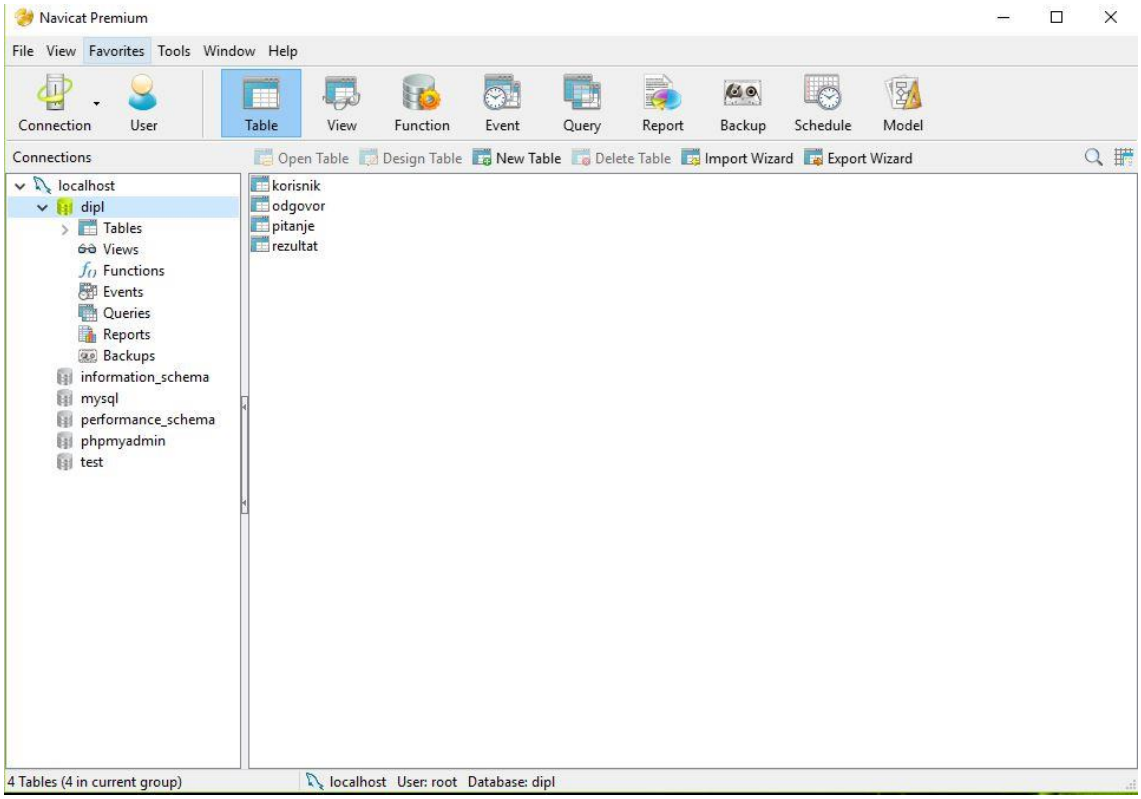
Slika 3.1. Blok dijagram mobilne aplikacije

Baza podataka i mobilna aplikacija napravljene su na sljedeći način:

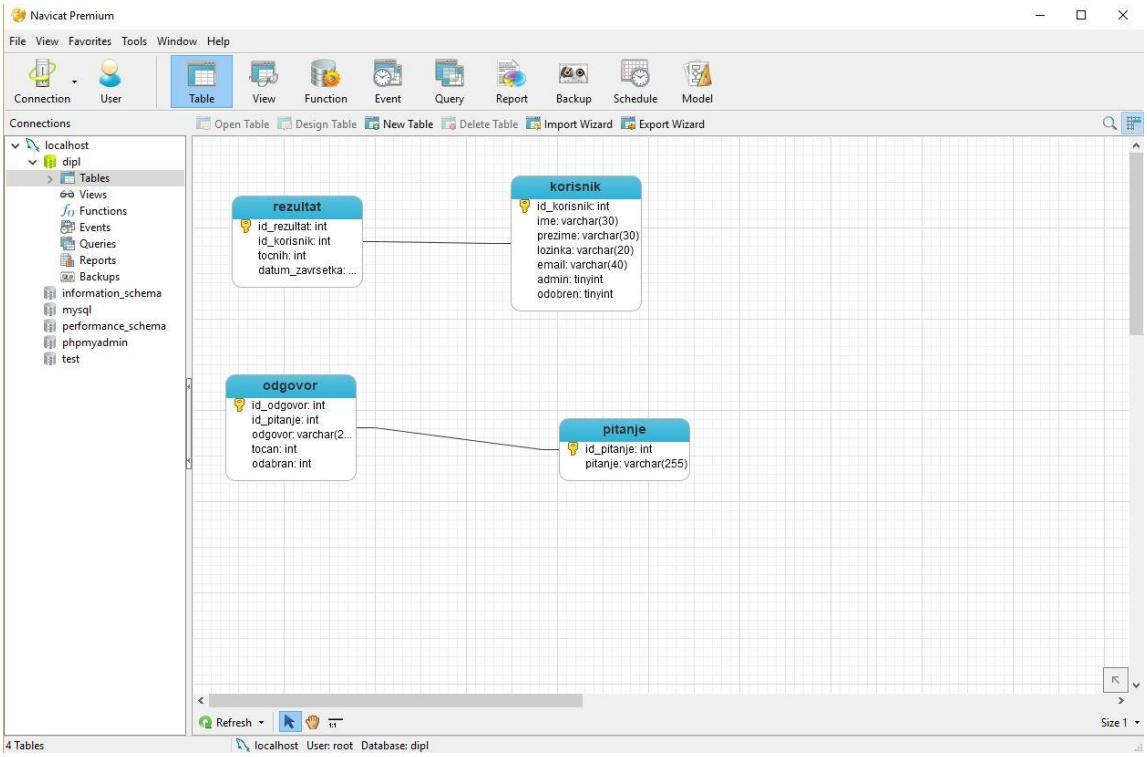
- Kreiran je server lokalno na računalu,
- Kreirana je baza podataka koja se sastoji od četiri tablice, tablica korisnik, tablica pitanje, tablica odgovor, tablica rezultati,
- Izrađena Mobilna aplikacija koja ima dva sučelje prijave, *"administrator"* i *"ispitanik"*,
- Sučelje *"administrator"*, ima mogućnost pregleda, unosa i uređivanja podataka s kojima aplikacija raspolaže,
- Sučelje *"ispitanik"*, ima samo jednu mogućnost, pisanje ispita, te nakon završetka prikaz vlastitih rezultata.

Baza podataka

Baza podataka je MySQL, kreirana je pomoću alata Navicat. Kako je baza napravljena lokalno na računalu, potrebno je pokrenuti XAMPP alat i u pokrenutom alatu pokrenuti MySQL. Server kreiran lokalno na računalu sadrži node server koji se mora pokrenuti pomoću alata Git Bash sa instrukcijom "node server", na serveru se nalaze instrukcije koje vrše potrebne operacije, kao što su unos, brisanje i uređivanje nad bazom podataka.

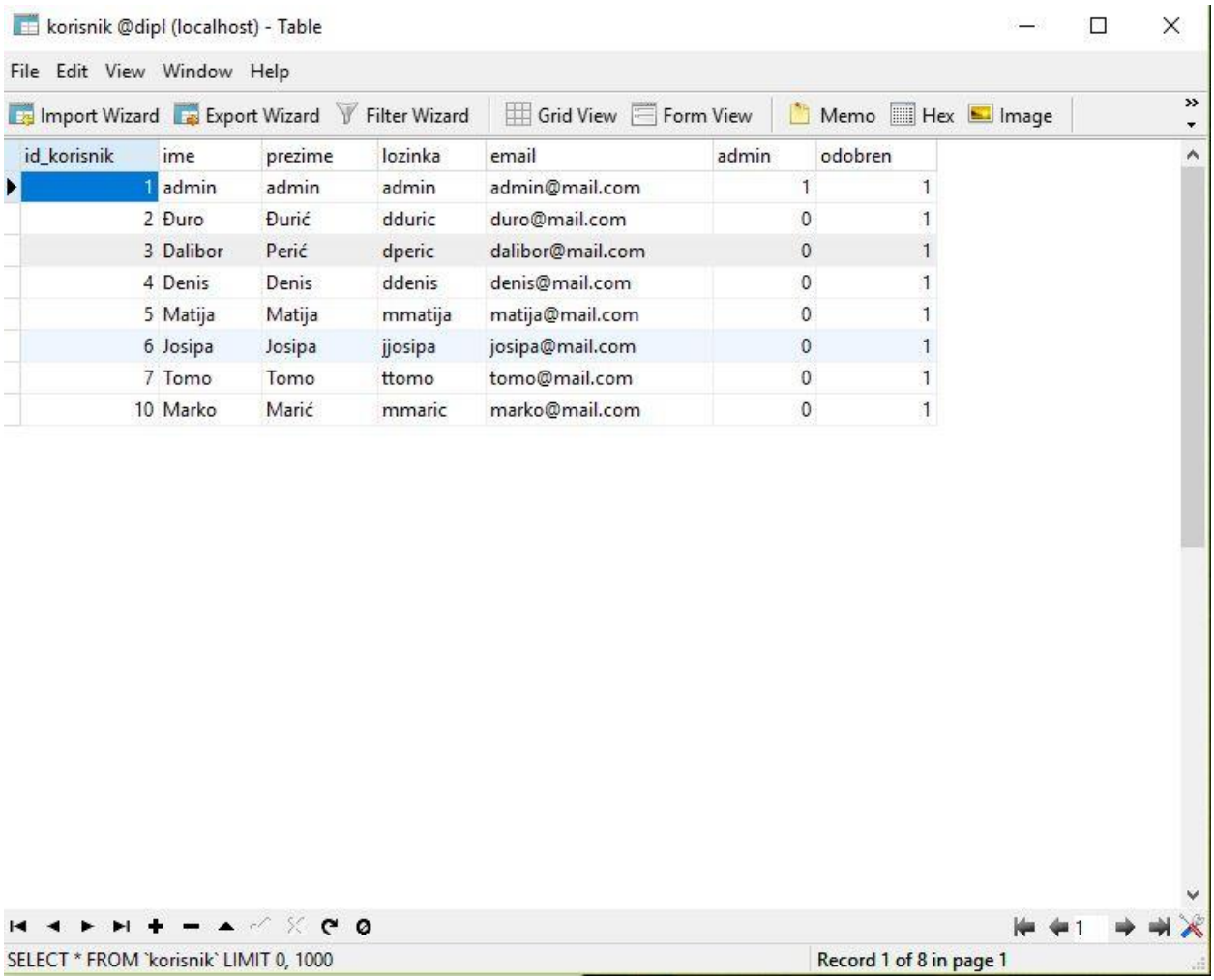


Slika 3.2. Navicat alat za izradu baze podataka (MySQL)



Slika 3.3. Dizajn MySQL baze podataka u Navicat alatu

Prema zahtjevima mobilne aplikacije, baza podataka napravljena je sa četiri tablice, tablice: *korisnik, pitanje, odgovor, rezultati*, kako je prikazano na slici 3.3. Tablica *"korisnik"* u koju se korisnici mogu dodavati unutar same baze podataka ili putem mobilne aplikacije, prikazana je na slici 3.4. Prilikom dodavanja korisnika, korisniku se može dodati status *"administrator"* ili *"korisnik"*. Putem mobilne aplikacije korisnika može dodati samo korisnik koji ujedno ima i status *"administrator"*.

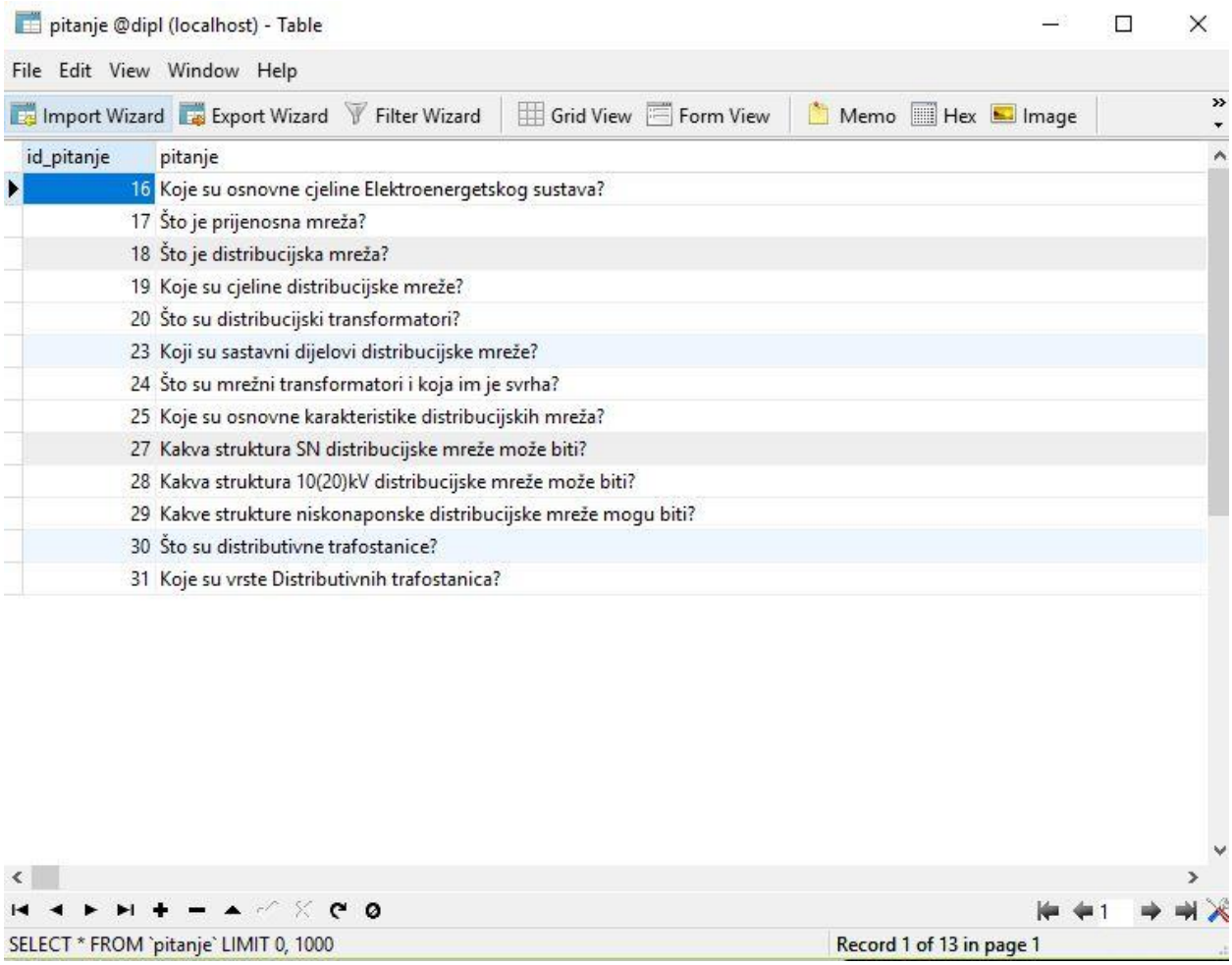


id_korisnik	ime	prezime	lozinka	email	admin	odobren
1	admin	admin	admin	admin@mail.com	1	1
2	Đuro	Đurić	dduric	duro@mail.com	0	1
3	Dalibor	Perić	dperic	dalibor@mail.com	0	1
4	Denis	Denis	ddenis	denis@mail.com	0	1
5	Matija	Matija	mmatija	matija@mail.com	0	1
6	Josipa	Josipa	jjosipa	josipa@mail.com	0	1
7	Tomo	Tomo	ttomo	tomo@mail.com	0	1
10	Marko	Marić	mmaric	marko@mail.com	0	1

Slika 3.4. Tablica *"korisnik"*

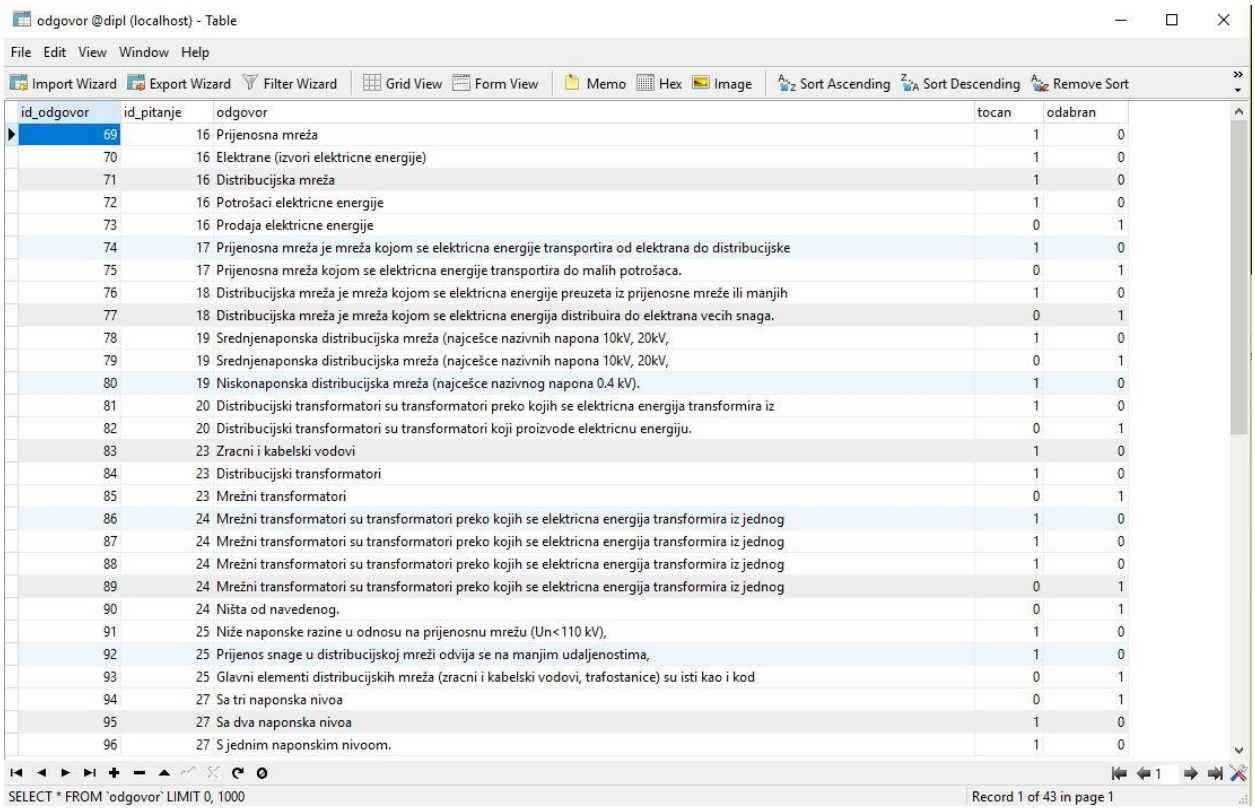
Tablica *"korisnik"* u sebi sadrži popis kreiranih korisnika s pripadajućim statusom. Svakom korisniku prilikom unosa mora biti dodijeljen identifikator **id_korisnik**, ovo se odnosi na ručni unos u samu bazu podataka, dok se kod unosa korisnika putem mobilne aplikacije identifikator **id_korisnik** automatski dodjeljuje. Kreiranje korisnika izvodi se na način da se upiše ime i

prezime korisnika, te email adresa i lozinka, koje će mobilna aplikacija koristiti kao korisničko ime i lozinku prilikom prijave u aplikaciju, također je obavezan unos statusa korisnika, tj. *"administrator"* ili *"korisnik"*.



Slika 3.5. Tablica *"pitanje"*

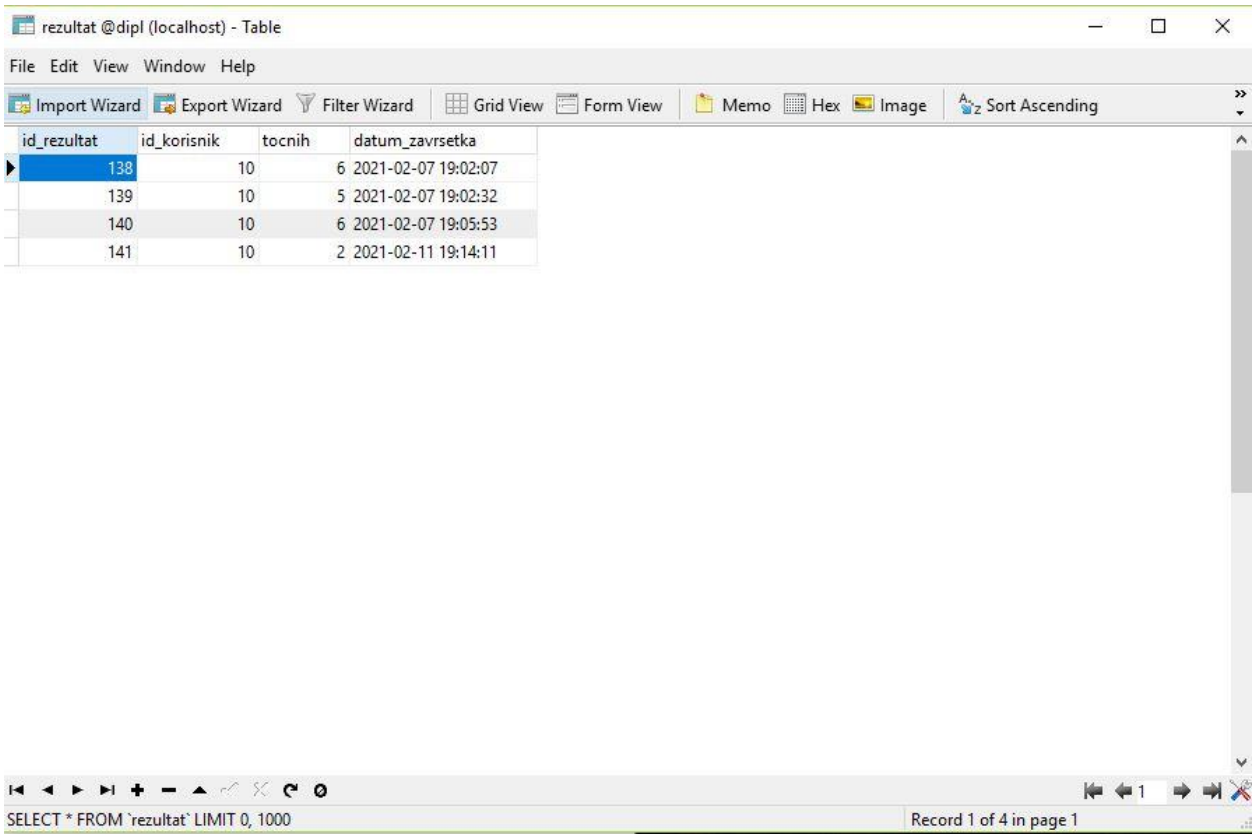
Tablica *"pitanje"* u sebi sadrži popis kreiranih pitanja. Svakom pitanju prilikom unosa mora biti dodijeljen identifikator **id_pitanje**, ovo se odnosi na ručni unos u samu bazu podataka, dok se kod unosa korisnika putem mobilne aplikacije identifikator **id_pitanje** automatski dodjeljuje. Kreiranje pitanja je jednostavno, izvodi se na način da se upiše pitanje te identifikator **id_pitanje** za pripadajuće pitanje.



id_odgovor	id_pitanje	odgovor	tocan	odabran
69	16	Prijenosna mreža	1	0
70	16	Elektrane (izvori elektricne energije)	1	0
71	16	Distribucijska mreža	1	0
72	16	Potrošaci elektricne energije	1	0
73	16	Prodaja elektricne energije	0	1
74	17	Prijenosna mreža je mreža kojom se elektricna energije transportira od elektrana do distribucijske	1	0
75	17	Prijenosna mreža kojom se elektricna energije transportira do malih potrošaca.	0	1
76	18	Distribucijska mreža je mreža kojom se elektricna energije preuzeta iz prijenosne mreže ili manjih	1	0
77	18	Distribucijska mreža je mreža kojom se elektricna energija distribuira do elektrana vecih snaga.	0	1
78	19	Srednjenaponska distribucijska mreža (najcešce nazivnih napona 10kV, 20kV,	1	0
79	19	Srednjenaponska distribucijska mreža (najcešce nazivnih napona 10kV, 20kV,	0	1
80	19	Niskonaponska distribucijska mreža (najcešce nazivnog napona 0.4 kV).	1	0
81	20	Distribucijski transformatori su transformatori preko kojih se elektricna energija transformira iz	1	0
82	20	Distribucijski transformatori su transformatori koji proizvode elektricnu energiju.	0	1
83	23	Zracni i kabelski vodovi	1	0
84	23	Distribucijski transformatori	1	0
85	23	Mrežni transformatori	0	1
86	24	Mrežni transformatori su transformatori preko kojih se elektricna energija transformira iz jednog	1	0
87	24	Mrežni transformatori su transformatori preko kojih se elektricna energija transformira iz jednog	1	0
88	24	Mrežni transformatori su transformatori preko kojih se elektricna energija transformira iz jednog	1	0
89	24	Mrežni transformatori su transformatori preko kojih se elektricna energija transformira iz jednog	0	1
90	24	Ništa od navedenog.	0	1
91	25	Niže naponske razine u odnosu na prijenosnu mrežu (Un<110 kV),	1	0
92	25	Prijenos snage u distribucijskoj mreži odvija se na manjim udaljenostima,	1	0
93	25	Glavni elementi distribucijskih mreža (zracni i kabelski vodovi, trafostanice) su isti kao i kod	0	1
94	27	Sa tri naponska nivoa	0	1
95	27	Sa dva naponska nivoa	1	0
96	27	S jednim naponskim nivoom.	1	0

Slika 3.6. Tablica *"odgovor"*

Tablica *"odgovor"* u sebi sadrži popis kreiranih odgovora. Svakom odgovoru prilikom unosa mora biti dodijeljen identifikator **id_odgovor**, također se iz padajućeg izbornika mora odabrati identifikator **id_pitanje**, kako bi se odgovor koji se unosi dodijelio pripadajućem pitanju, ovaj dio se odnosi na ručni unos u samu bazu podataka, dok se kod unosa korisnika putem mobilne aplikacije identifikator **id_odgovor** automatski dodjeljuje. Kreiranje odgovora izvodi se na način da se upiše odgovor, sa obavezanim unosom statusa odgovora, tj. *"točan"* ili *"netočan"*, upisati identifikator **id_odgovor, te iz padajućeg izbornika odabrati** identifikator **id_pitanje**.



Slika 3.7. Tablica *"rezultat"*

Tablica *"rezultat"* u sebi sadrži popis dobivenih rezultata nakon provjere znanja. Tablica *"rezultat"* prikazuje identifikator **id_rezultat**, korisnika čiji je pripadajući rezultat s njegovim identifikatorom **id_korisnik**, broj točnih odgovora pripadajućeg korisnika, te datum i vrijeme završetka provjere znanja.

Mobilna aplikacija za provjeru znanja pripravnika

Mobilna aplikacija je napravljena sa Ionic alatom koji u sebi sadrži prethodno spomenuti Angular okvir, sastoji se od dva sučelja, sučelje prijave i početno sučelje gdje su prikazani daljnji koraci koje aplikacija omogućava. Sučelja s prikazom mogućnosti aplikacije ovise o načinu prijave, *"administrator"* ili *"korisnik"*. Mobilna aplikacija preko node servera vrši operacije nad bazom podataka, šalje upite prema bazi podataka. Baza podataka odgovara mobilnoj aplikaciji tako što joj omogućava pristup traženim podacima, ovisno o načinu prijave, *"administrator"* ili *"korisnik"*.

Mobilna aplikacija - prijava

```
apiRoutes.post('/login', function(req, res){
    console.log(req.body);
    if (req.body.email && req.body.lozinka){
        pool.getConnection(function(err, connection) {
            if (err) {
                res.json({ success: false, message: 'Nešto je pošlo po zlu.', data:err });
            }else{
                var query = "SELECT * FROM korisnik WHERE BINARY email = ? AND BINARY lozinka =? AND odobren=1"; //api za prijavu - prima email i lozinku i ako postoji vraća sve podatk
                var table = [req.body.email, req.body.lozinka]; // parametri koje funkcija prima
                query = mysql.format(query,table);
                connection.query(query,function(err,rows){
                    connection.release();
                    if (err){
                        res.json({ success: false, message: 'Nešto je pošlo po zlu, SQL' });
                    }else{
                        if(rows.length > 0) {
                            res.json({success: true,message: 'Uspješna prijava!', data: rows});
                        } else {
                            res.json({success: false, message: 'Neuspješna prijava. Pogrešni podaci za prijavu ili je korisnik neaktivan' });
                        }
                    }
                });
            }
        });
    }else{
        res.json({ success: false, message: 'Podaci koje saljete nisu ispravni.' });
    }
});
```

Slika 3.8. Node server - prijava korisnika, JavaScript

```
prijava(loginpodaci){ //prijava, ako je uspješnja ide na početni ekran
  console.log(loginpodaci)
  this.api.prijava({"email": loginpodaci.email,"lozinka": loginpodaci.lozinka}).subscribe(res => {
    console.log(res)
    if (res.success == true) {
      console.log(res.data[0]);
      this.storage.set('korisnik', res.data[0]);
      this.loginpodaci.email = '';
      this.loginpodaci.lozinka = '';
      this.router.navigateByUrl('/pocetna');
    }else{
      this.greskaAlert();
    }
  })
}

async greskaAlert() { // poruka greške
  const alert = await this.alertController.create({
    cssClass: 'my-custom-class',
    header: 'Greška',
    message: 'Provjerite svoj e-mail i lozinku!',
    buttons: ['U redu']
  });
  await alert.present();
}
```

Slika 3.9. Mobilna aplikacija - prijava korisnika, TypeScript

Mobilna aplikacija kod prijave korisnika, kada se upišu korisničko ime i lozinka, poziva *"api"* koji se nalazi na node serveru, poziva ga funkcijom *this.api.prijava(). "Api"* provjerava podatke u bazi podataka, ukoliko su podaci točni povlači ih iz baze podataka funkcijom *pool.getConnection()*. Odgovara mobilnoj aplikaciji na poziv (upit) uspješnom prijavom

korisnika. Relacije i komunikacije između mobilne aplikacije i servera prikazane su na slikama 3.8., 3.9.

Mobilna aplikacija – dodaj korisnika

```
apiRoutes.post('/korisnikDodaj', function(req, res){
    console.log(req.body);
    if (req.body.ime && req.body.prezime && req.body.email && req.body.lozinka){
        pool.getConnection(function(err, connection) {
            if (err) {
                res.json({ success: false, message: 'Nešto je pošlo po zlu.', data:err });
            }else{
                var objKorsnik = {
                    ime :req.body.ime,
                    prezime :req.body.prezime,
                    email:req.body.email,
                    lozinka:req.body.lozinka,
                    admin:req.body.admin || 0, // ako ne dobije parametar postavlja 0
                    odobren:req.body.odobren || 0,
                }
                var query = "INSERT INTO korisnik SET ?";   // dodavanje korisnika
                var table = [objKorsnik];   // parametri
                query = mysql.format(query,table);
                connection.query(query,function(err,rows){
                    connection.release();
                    if (err){
                        res.json({ success: false, message: 'Nešto je pošlo po zlu, SQL' });
                    }else{
                        res.json({success: true,message: 'Uspješno!', data: rows});
                    }
                });
            }
        });
    }else{
        res.json({ success: false, message: 'Podaci koje saljete nisu ispravni.' });
    }
});
```

Slika 3.10. Node server - dodavanje korisnika, JavaScript

```
dodajIspitanika(){  // spremanje novog ispitanika
  this.api.korisnikDodaj({"email": this.email,"lozinka": this.lozinka, "admin": this.admin, "ime": this.ime, "prezime": this.prezime, "odobren": this.odobren}).subscribe(res => {
    console.log(res)
    if (res.success == true) {
      console.log(res.data[0]);
      this.storage.set('korisnik', res.data[0]);
      this.router.navigateByUrl('/popisispitanika');
    }else{
      this.greskaAlert();
    }
  })
}

odustani(){ // vraća ne prethodni ekran
  this.router.navigateByUrl('/popisispitanika');
}
```

```
async greskaAlert() { // greška alert
  const alert = await this.alertController.create({
    cssClass: 'my-custom-class',
    header: 'Greška',
    message: 'Došlo je do pogreške kod dodavanja ispitanika!',
    buttons: ['U redu']
  });
  await alert.present();
}
```

Slika 3.11. Mobilna aplikacija - dodavanje korisnika, TypeScript

Mobilna aplikacija kod dodavanja korisnika, kada se upišu traženi podaci za novoga, poziva *"api"* koji se nalazi na node serveru, poziva ga funkcijom *this.api.korisnikDodaj()*. *"Api"* izvršava unos podataka u bazu podataka, zatim funkcijom *pool.getConnection()* podatke povlači iz baze podataka. Odgovara mobilnoj aplikaciji na poziv (upit) o uspješnom dodavanju korisnika. Relacije i komunikacije između mobilne aplikacije i servera prikazane su na slikama 3.10., 3.11.

Mobilna aplikacija – brisanje korisnika

```
apiRoutes.post('/korisnikObrisi', function(req, res){
    console.log(req.body);
    if (req.body.id_korisnik){
        pool.getConnection(function(err, connection) {
            if (err) {
                res.json({ success: false, message: 'Nešto je pošlo po zlu.', data:err });
            }else{
                var query = "DELETE FROM korisnik where id_korisnik=?"; // brisanje korisnika
                var table = [req.body.id_korisnik];
                query = mysql.format(query,table);
                connection.query(query,function(err,rows){
                    connection.release();
                    if (err){
                        res.json({ success: false, message: 'Nešto je pošlo po zlu, SQL' });
                    }else{
                        res.json({success: true,message: 'Uspješno!', data: rows});
                    }
                });
            }
        });
    }else{
        res.json({ success: false, message: 'Podaci koje saljete nisu ispravni.' });
    }
});
```

Slika 3.12. Node server – brisanje korisnika, JavaScript

```
async obrisiIspitanika(id_korisnik){  // brisanje ispitanika iz baze
  const alert = await this.alertController.create({
    cssClass: 'my-custom-class',
    message: 'Jeste sigurni da želite obrisati ispitanika?',
    buttons: [
      {
       text: 'Obriši',
       handler: () => {
         this.api.korisnikObrisi({id_korisnik: id_korisnik}).subscribe(res => {
           console.log(res)
           if (res.success == true) {
             console.log(res.data);
             this.toastPoruka('Korisnik uspješno obrisan!');
             this.dohvatiIspitanike();
           }else{
             this.greskaAlert('Došlo je do greške kod brisanja ispitanika!');
           }
         })
       }
      },
      {
       text: 'Odustani',
       role: 'cancel',
       cssClass: 'secondary',
       handler: () => {

       }
      }
    ]
  });
  await alert.present();
}
```

Slika 3.13. Mobilna aplikacija – brisanje korisnika, TypeScript

Mobilna aplikacija kod brisanja korisnika, poziva *"api"* koji se nalazi na node serveru, poziva ga funkcijom *this.api.korisnikObrisi()*. *"Api"* izvršava brisanje podataka u bazi podataka, zatim funkcijom *pool.getConnection()* odgovara mobilnoj aplikaciji na poziv (upit) o uspješnom brisanju korisnika iz baze podataka. Relacije i komunikacije između mobilne aplikacije i servera prikazane su na slikama 3.12., 3.13.

Mobilna aplikacija – dodaj pitanje

```
apiRoutes.post('/pitanjeDodaj', function(req, res){
    console.log(req.body);
    if (req.body.pitanje && req.body.odgovor){
        if(req.body.odgovor.length > 0){
            pool.getConnection(function(err, connection) {
                if (err) {
                    res.json({ success: false, message: 'Nešto je pošlo po zlu.', data:err });
                }else{
                    var query = "INSERT INTO pitanje SET pitanje=?";    // dodavanje pitanja, sprema pitanje array s odgovorima
                    var table = [req.body.pitanje];
                    query = mysql.format(query,table);
                    connection.query(query,function(err,rows){
                        if (err){
                            console.log(err);
                            connection.release();
                            res.json({ success: false, message: 'Nešto je pošlo po zlu, SQL' });
                        }else{
                            if(rows.insertId > 0) {
                                var podaciupis = req.body.odgovor.map( function(el) { return mysql.format("(?,?,?)",
                                [rows.insertId, el.odgovor ||'', el.tocan || 0 ]) });
                                var query="insert into odgovor";    // spremanje odgovora
                                query+=" (id_pitanje, odgovor, tocan)";
                                query+=" VALUES "+podaciupis;
                                console.log(query)
                                connection.query(query,function(err,rows){
                                    connection.release();
                                    if (err){
                                        console.log(err);
                                        res.json({ success: false, message: 'Nešto je pošlo po zlu, SQL' });
                                    }else{
                                        res.json({success: true,message: 'Uspješno!', data: rows});
                                    }
                                });
                            } else {
                                connection.release();
                                res.json({success: false, message: 'Nema podataka' });
                            }
                        }
                    });
                }
            });
        }else{
            res.json({ success: false, message: 'Niste poslali odgovore.' });
        }

    }else{
        res.json({ success: false, message: 'Podaci koje saljete nisu ispravni.' });
    }
```

Slika 3.14. Node server - dodavanje pitanja i odgovora za pripadajuće pitanje, JavaScript

```
spremiPitanje(){  // spremanje pitanja zajedno sa odgovorima
  if(this.pitanje.length > 0){
    console.log(this.pitanje);
    console.log(this.odgovori);

    this.api.pitanjeDodaj({"pitanje": this.pitanje, "odgovor": this.odgovori}).subscribe(res => {
      console.log(res)
      if (res.success == true) {
        console.log(res.data);
        this.odgovor = '';
        this.toastPoruka("Uspješno spremljeno pitanje");
        this.router.navigateByUrl('/popispitanja');
      }else{
        this.greskaAlert('Došlo je do greške kod spremanja pitanja!');
      }
    })
  }else{
    this.greskaAlert('Niste unijeli pitanje!');
  }
}

async greskaAlert(poruka) { // greška alert
  const alert = await this.alertController.create({
    cssClass: 'my-custom-class',
    header: 'Greška',
    message: poruka,
    buttons: ['U redu']
  });
  await alert.present();
}

async toastPoruka(poruka) { //toast poruka od 2 sec, pojavi se nakon spremanja pitanja
  const toast = await this.toastController.create({
    message: poruka,
    duration: 2000
  });
  toast.present();
}
```

Slika 3.15. Mobilna aplikacija - dodavanje pitanja i odgovora za pripadajuće pitanje, TypeScript

Mobilna aplikacija kod dodavanja pitanja, kada se upišu traženi podaci za novo pitanje, poziva *"api"* koji se nalazi na node serveru, poziva ga funkcijom *this.api.pitanjeDodaj()*. *"Api"* izvršava unos podataka u bazu podataka, zatim funkcijom *pool.getConnection()* podatke povlači iz baze podataka. Odgovara mobilnoj aplikaciji na poziv (upit) o uspješnom dodavanju pitanja i pripadajućeg odgovora. Relacije i komunikacije između mobilne aplikacije i servera prikazane su na slikama 3.14., 3.15.

Mobilna aplikacija – brisanje pitanje

```
apiRoutes.post('/pitanjeObrisi', function(req, res){
    console.log(req.body);
    if (req.body.id_pitanje){
        pool.getConnection(function(err, connection) {
            if (err) {
                res.json({ success: false, message: 'Nešto je pošlo po zlu.', data:err });
            }else{
                var query = "DELETE FROM pitanje where id_pitanje=?";   // brisanje pitanja
                var table = [req.body.id_pitanje];
                query = mysql.format(query,table);
                connection.query(query,function(err,rows){
                    connection.release();
                    if (err){
                        res.json({ success: false, message: 'Nešto je pošlo po zlu, SQL' });
                    }else{
                        res.json({success: true,message: 'Uspješno!', data: rows});
                    }
                });
            }
        });
    }else{
        res.json({ success: false, message: 'Podaci koje saljete nisu ispravni.' });
    }
});
```

Slika 3.16. Node server – brisanje pitanja, JavaScript

```
async obrisiPitanje(){  // brisanje pitanja
  const alert = await this.alertController.create({
    cssClass: 'my-custom-class',
    message: 'Jeste sigurni da želite obrisati pitanje?',
    buttons: [
      {
        text: 'Obriši',
        handler: () => {
          this.api.pitanjeObrisi({id_pitanje: this.podacipitanja.id_pitanje}).subscribe(res => {
            console.log(res)
            if (res.success == true) {
              console.log(res.data);
              this.toastPoruka('Pitanje uspješno obrisano!');
              this.router.navigateByUrl('/popispitanja');
            }else{
              this.greskaAlert('Došlo je do greške kod brisanja pitanja!');
            }
          })
        }
      },
      {
        text: 'Odustani',
        role: 'cancel',
        cssClass: 'secondary',
        handler: () => {

        }
      }
    ]
  });
  await alert.present();
}
```

Slika 3.17. Mobilna aplikacija – brisanje pitanja, TypeScript

Mobilna aplikacija kod brisanja pitanja, poziva *"api"* koji se nalazi na node serveru, poziva ga funkcijom *this.api.pitanjeObrisi()*. *"Api"* izvršava brisanje podataka u bazi podataka, zatim funkcijom *pool.getConnection()* odgovara mobilnoj aplikaciji na poziv (upit) o uspješnom brisanju pitanja iz baze podataka. Relacije i komunikacije između mobilne aplikacije i servera prikazane su na slikama 3.16., 3.17.

Mobilna aplikacija – dodaj odgovor

```
apiRoutes.post('/odgovorDodaj', function(req, res){
    console.log(req.body);
    if (req.body.odgovor && req.body.id_pitanje){
        pool.getConnection(function(err, connection) {
            if (err) {
                res.json({ success: false, message: 'Nešto je pošlo po zlu.', data:err });
            }else{
                var objOdgovor = {
                    odgovor :req.body.odgovor,
                    id_pitanje :req.body.id_pitanje,
                    tocan: req.body.tocan || 0
                }
                var query = "INSERT INTO odgovor SET ?";    // dodavanje odgovora
                var table = [objOdgovor];
                query = mysql.format(query,table);
                connection.query(query,function(err,rows){
                    connection.release();
                    if (err){
                        res.json({ success: false, message: 'Nešto je pošlo po zlu, SQL' });
                    }else{
                        res.json({success: true,message: 'Uspješno!', data: rows});
                    }
                });
            }
        });
    }else{
        res.json({ success: false, message: 'Podaci koje saljete nisu ispravni.' });
    }
});
```

Slika 3.18. Node server – dodavanje odgovora, JavaScript

```
dodajOdgovor(odgovor){  // dodavanje odgovora ( veže se uz pitanje )
  if(odgovor.length > 0){
    this.api.odgovorDodaj({id_pitanje: this.podacipitanja.id_pitanje, odgovor: odgovor, tocan: this.tocan}).subscribe(res => {
      console.log(res)
      if (res.success == true) {
        console.log(res.data);
        this.odgovor = '';
        this.tocan = false;
        this.dohvatiOdgovore();
      }else{
        this.greskaAlert('Došlo je do greške kod dodavanja odgovora!');
      }
    })
  }else{
    this.greskaAlert('Niste unijeli odgovor!');
  }
}
```

Slika 3.19. Mobilna aplikacija – dodavanje odgovora, TypeScript

Mobilna aplikacija kod dodavanja odgovora, kada se upišu traženi podaci za novi odgovor, poziva *"api"* koji se nalazi na node serveru, poziva ga funkcijom *this.api.odgovorDodaj()*. *"Api"* izvršava unos podataka u bazu podataka, zatim funkcijom *pool.getConnection()* podatke povlači iz baze podataka. Odgovara mobilnoj aplikaciji na poziv (upit) o uspješnom dodavanju odgovora. Relacije i komunikacije između mobilne aplikacije i servera prikazane su na slikama 3.18., 3.19.

Mobilna aplikacija – brisanje odgovora

```
apiRoutes.post('/odgovorObrisi', function(req, res){
    console.log(req.body);
    if (req.body.id_odgovor){
        pool.getConnection(function(err, connection) {
            if (err) {
                res.json({ success: false, message: 'Nešto je pošlo po zlu.', data:err });
            }else{
                var query = "DELETE FROM odgovor where id_odgovor=?";   // brisanje odgovora
                var table = [req.body.id_odgovor];
                query = mysql.format(query,table);
                connection.query(query,function(err,rows){
                    connection.release();
                    if (err){
                        res.json({ success: false, message: 'Nešto je pošlo po zlu, SQL' });
                    }else{
                        res.json({success: true,message: 'Uspješno!', data: rows});
                    }
                });
            }
        });
    }else{
        res.json({ success: false, message: 'Podaci koje saljete nisu ispravni.' });
    }
});
```

Slika 3.20. Node server – brisanje odgovora, JavaScript

```
obrisiOdgovor(id_odgovor){  // brisanje odgovora
  this.api.odgovorObrisi({id_odgovor: id_odgovor}).subscribe(res => {
    console.log(res)
    if (res.success == true) {
      console.log(res.data);
      this.dohvatiOdgovore();
    }else{
      this.greskaAlert('Došlo je do greške kod brisanja odgovora!');
    }
  })
}

odustani(){ // vraća na prethodni ekran
  this.router.navigateByUrl('/popispitanja');
}
```

Slika 3.21. Mobilna aplikacija – brisanje odgovora, TypeScript

Mobilna aplikacija kod brisanja odgovora, poziva *"api"* koji se nalazi na node serveru, poziva ga funkcijom *this.api.odgovorObrisi(). "Api"* izvršava brisanje podataka u bazi podataka, zatim funkcijom *pool.getConnection()* odgovara mobilnoj aplikaciji na poziv (upit) o uspješnom brisanju odgovora iz baze podataka. Relacije i komunikacije između mobilne aplikacije i servera prikazane su na slikama 3.20., 3.21.

Mobilna aplikacija – popis rezultata

```
//////////////////////////////////rezultat/////////////////////////////////////////////
apiRoutes.post('/rezultatPopis', function(req, res){
    console.log(req.body);
    pool.getConnection(function(err, connection) {
        if (err) {
            res.json({ success: false, message: 'Nešto je pošlo po zlu.', data:err });  // dohvat rezultata
        }else{
            var query = "SELECT rezultat.*,DATE_FORMAT(rezultat.datum_zavrsetka, '%Y-%m-%d %H:%i:%s') as datum_zavrsetkaSQL, DATE_FORMAT(rezultat.datum_zavrsetka, '%d.%m.%Y %H:%i:%s')
            var table = [];
            query = mysql.format(query,table);
            connection.query(query,function(err,rows){
                connection.release();
                if (err){
                    res.json({ success: false, message: 'Nešto je pošlo po zlu, SQL' });
                }else{
                    if(rows.length > 0) {
                        res.json({success: true,message: 'Uspješno!', data: rows});
                    } else {
                        res.json({success: false, message: 'Nema podataka' });
                    }
                }
            });
        }
    });
});
```

Slika 3.22. Node server – popis rezultata, JavaScript

```
ionViewWillEnter(){ // funkcija za dohvat rezultata se poziva prije samog ulaska na ekran rezultata
 this.dohvatiRezultate();
}

dohvatiRezultate(){ // dohvat rezultata ipitanika
  this.api.rezultatPopis({}).subscribe(res => {
    console.log(res)
    if (res.success == true) {
      console.log(res.data);
      this.rezultati = [];
      this.rezultati = res.data;
    }else{
      this.rezultati = [];
    }
  })
}

async greskaAlert(poruka) { // greška alert
  const alert = await this.alertController.create({
    cssClass: 'my-custom-class',
    header: 'Greška',
    message: poruka,
    buttons: ['U redu']
  });
  await alert.present();
}
```

Slika 3.23. Mobilna aplikacija – popis rezultata, TypeScript

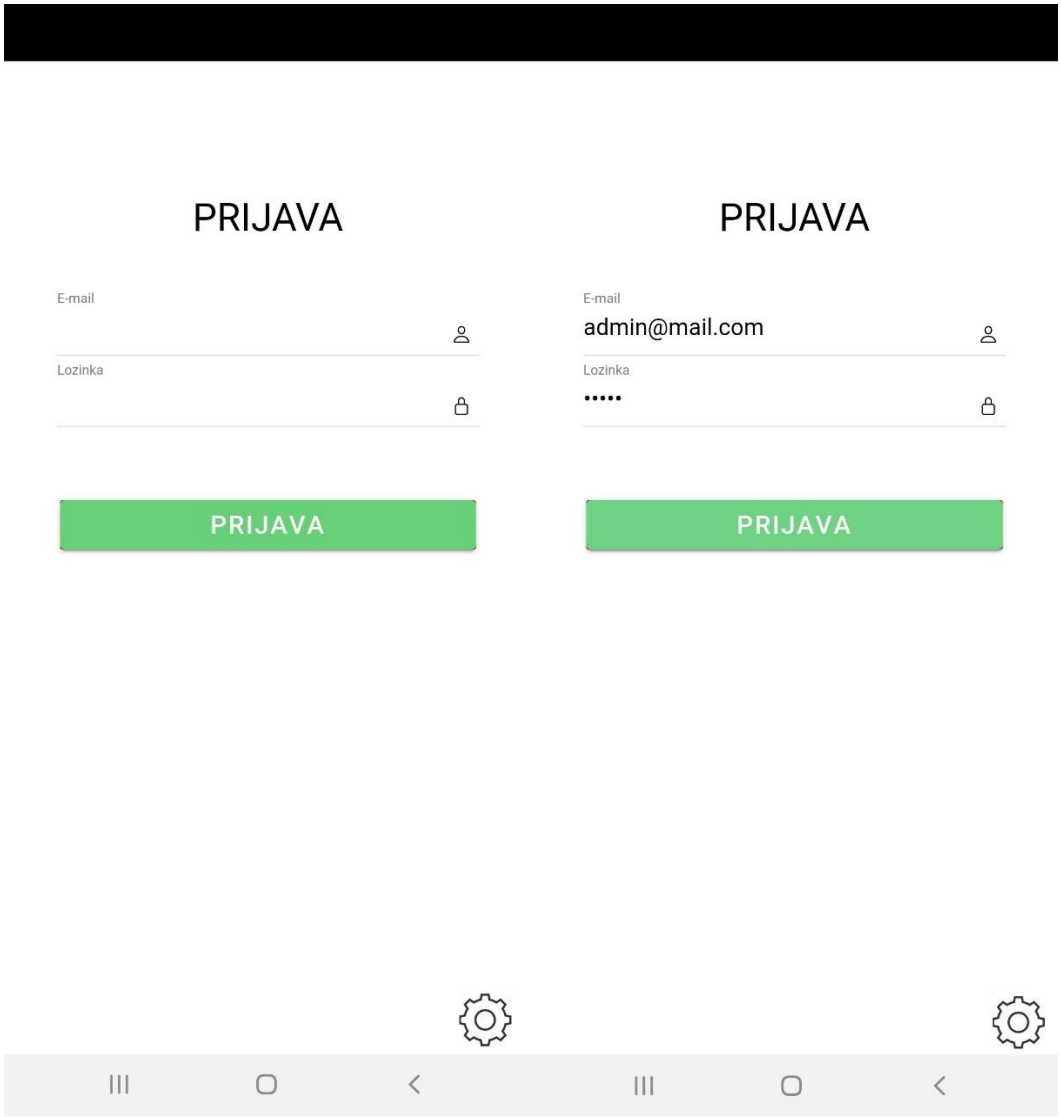
Mobilna aplikacija kod traženja popisa rezultata, poziva *"api"* koji se nalazi na node serveru, poziva ga funkcijom *this.api.rezultatPopis()*. *"Api"* provjerava podatke u bazi podataka, rezultate povlači iz baze podataka funkcijom *pool.getConnection()*. Odgovara mobilnoj aplikaciji na poziv (upit) popisom ispitanika. Relacije i komunikacije između mobilne aplikacije i servera prikazane su na slikama 3.22., 3.23.

4. IZGLED APLIKACIJE

U ovom poglavlju opisan je način korištenja Mobilne aplikacije. Kratke upute za upotrebu Mobilne aplikacije dane su u nastavku.

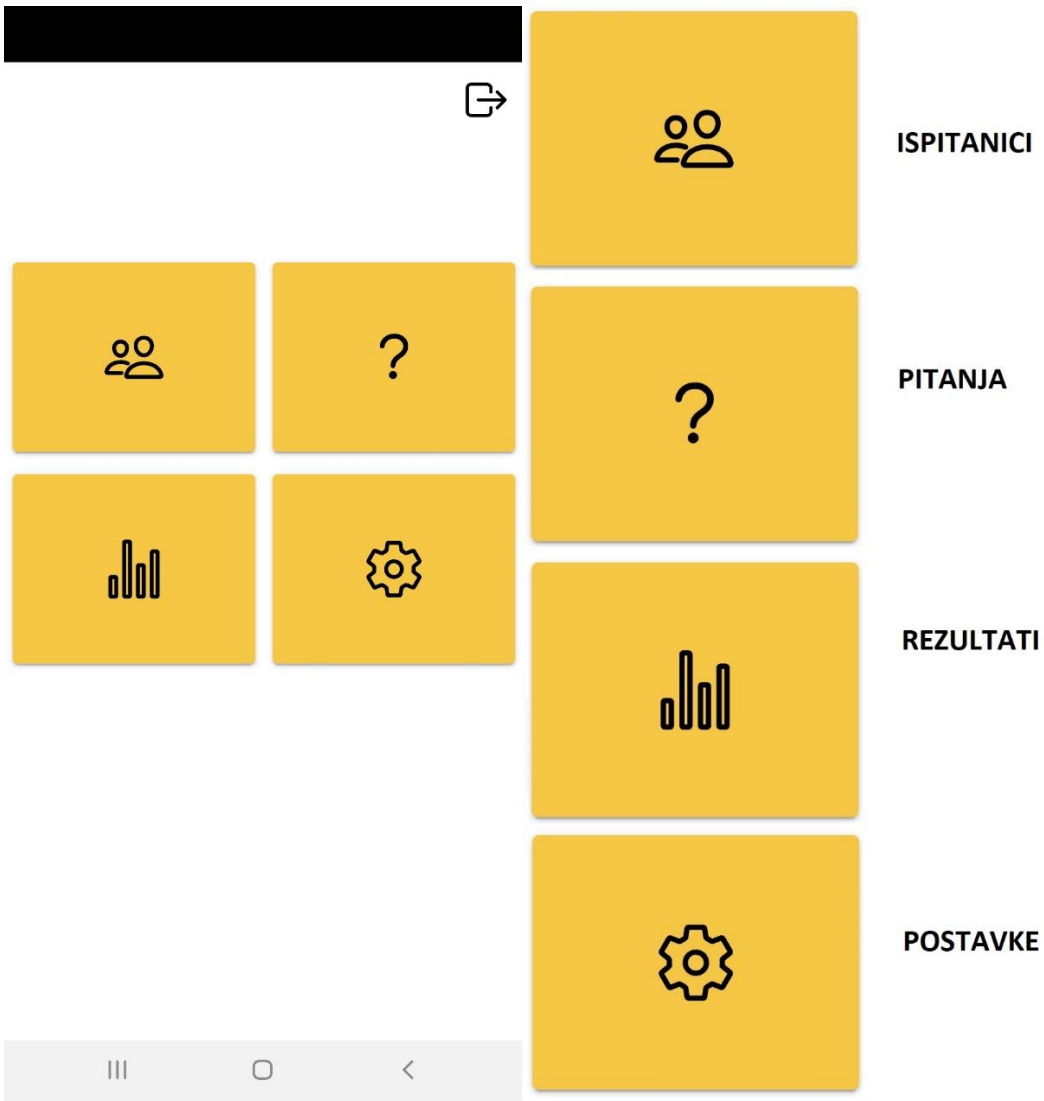
Instalacija i pokretanje Mobilne aplikacije na mobilnom uređaju

Potrebno je preuzeti instalacijsku datoteku Mobilne aplikacije ili ju prebaciti putem USB kabela na mobilni uređaj. Pokrenuti instalacijsku datoteku kako bi se aplikacija instalirala na mobilni uređaj. Nakon uspješne instalacije pokrenuti mobilnu aplikaciju, pokretanjem aplikacije prikazuje se početni zaslon, zaslon za prijavu, kako je prikazano na slici 4.1.



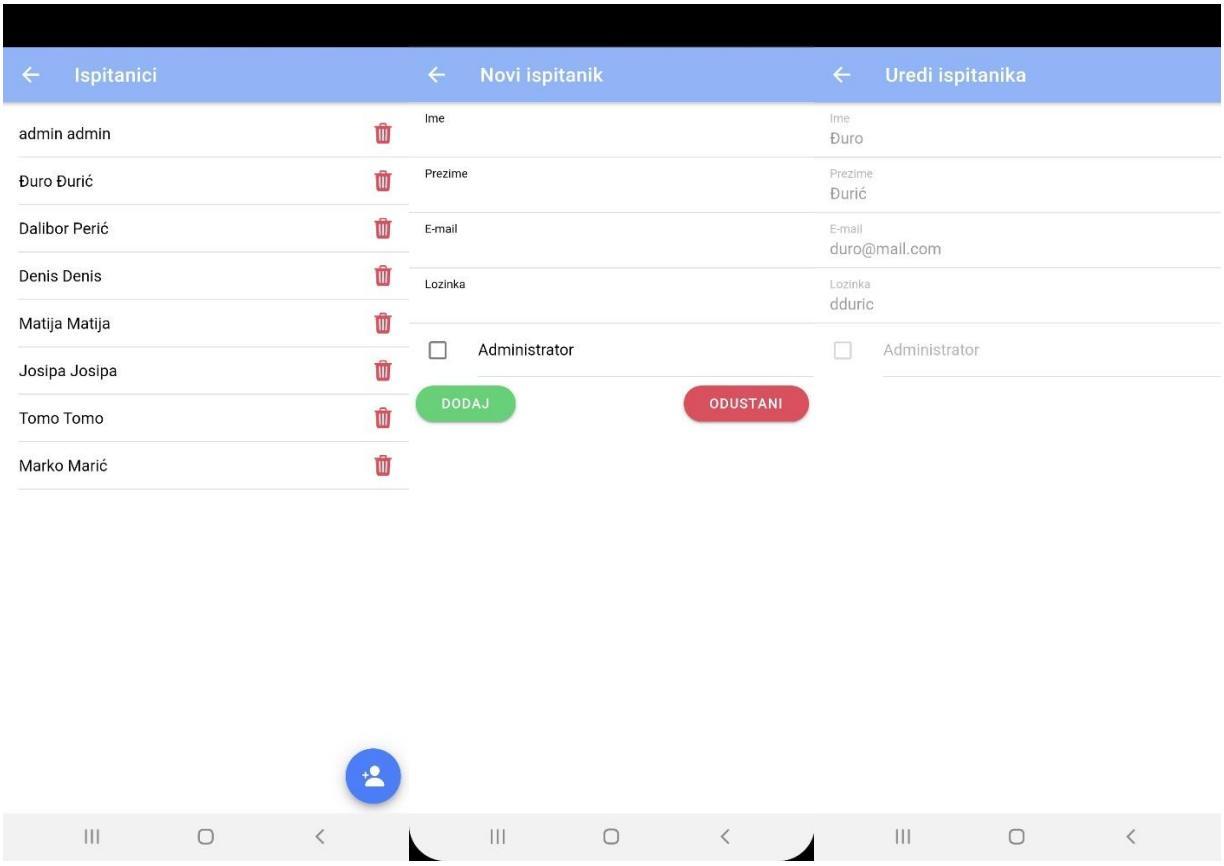
Slika 4.1. Mobilna aplikacija – sučelje za prijavu

U sučelje za prijavu, prikazanom na slici 4.1., potrebno je unijeti email adresu kao korisničko ime, te lozinku. Lozinku administrator dodjeljuje prilikom dodavanja korisnika. Administrator također kod unosa novog korisnika određuje status korisnika, novi korisnik može biti administrator ili ispitanik. Aplikacija može imati više administratora i ispitanika. Nakon unosa korisničkog imena i lozinke kliknuti na gumb *"PRIJAVA"*. Ukoliko prijava nije uspješna, pojaviti će se poruka: *"Greška – provjerite svoj e-mail i lozinku"*. Uspješnom prijavom otvara se novi zaslon koji ovisi o prijavljenom korisniku. Ukoliko se prijavio *"administrator"*, slika 4.2. prikazuje zaslon s njegovim mogućnostima u radu s Mobilnom aplikacijom.



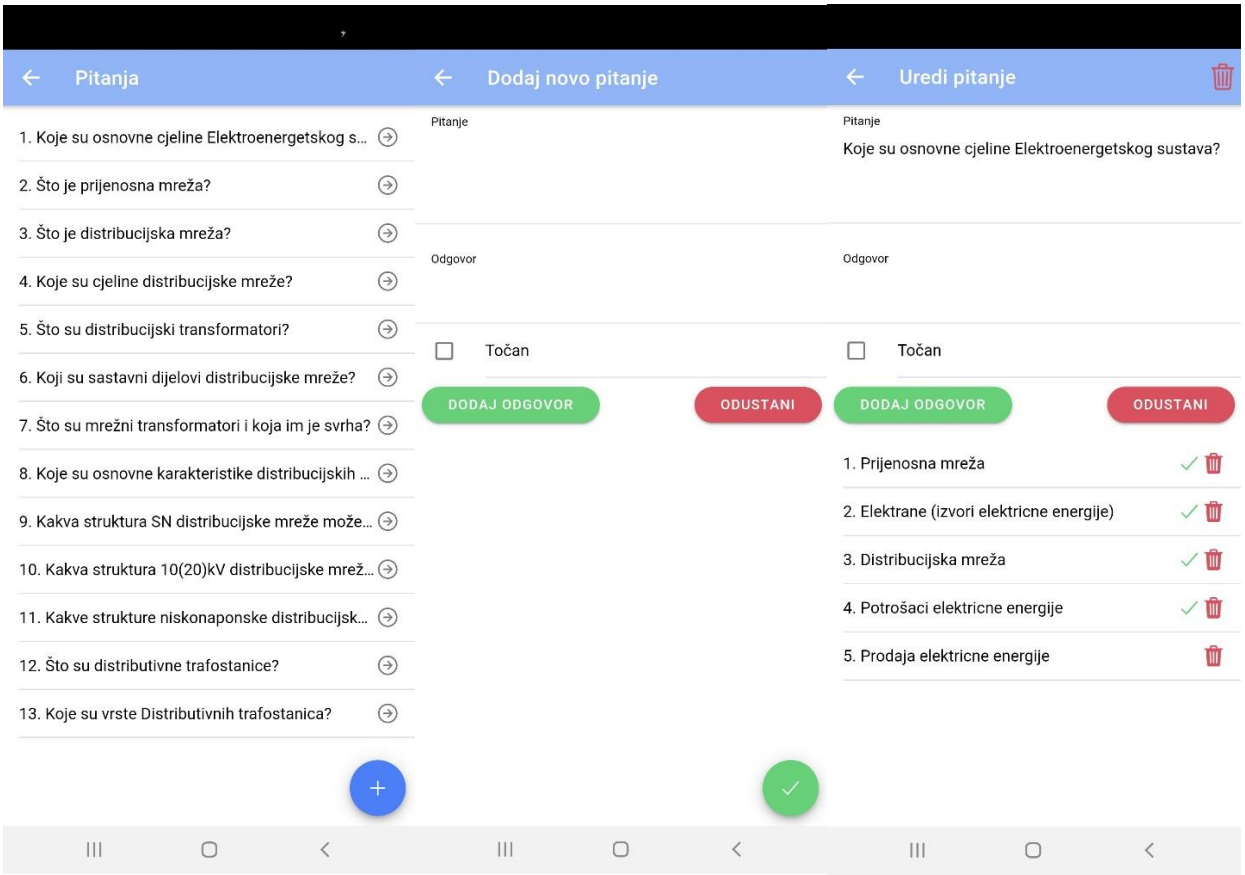
Slika 4.2. Mobilna aplikacija – sučelje *"administrator"*

Sučelje administratora prikazuje koje su sve njegove mogućnosti za rad u aplikaciji. Klikom na ikonu *"ispitanici"* otvara se popis postojećih korisnika. Popis ispitanika može se uređivati, što znači da se ispitanici mogu dodavati, brisati, te uređivati informacije o korisniku i njihovom statusu, *"administrator"* ili *"ispitanik"*, kako je prikazano na slici. 4.3.



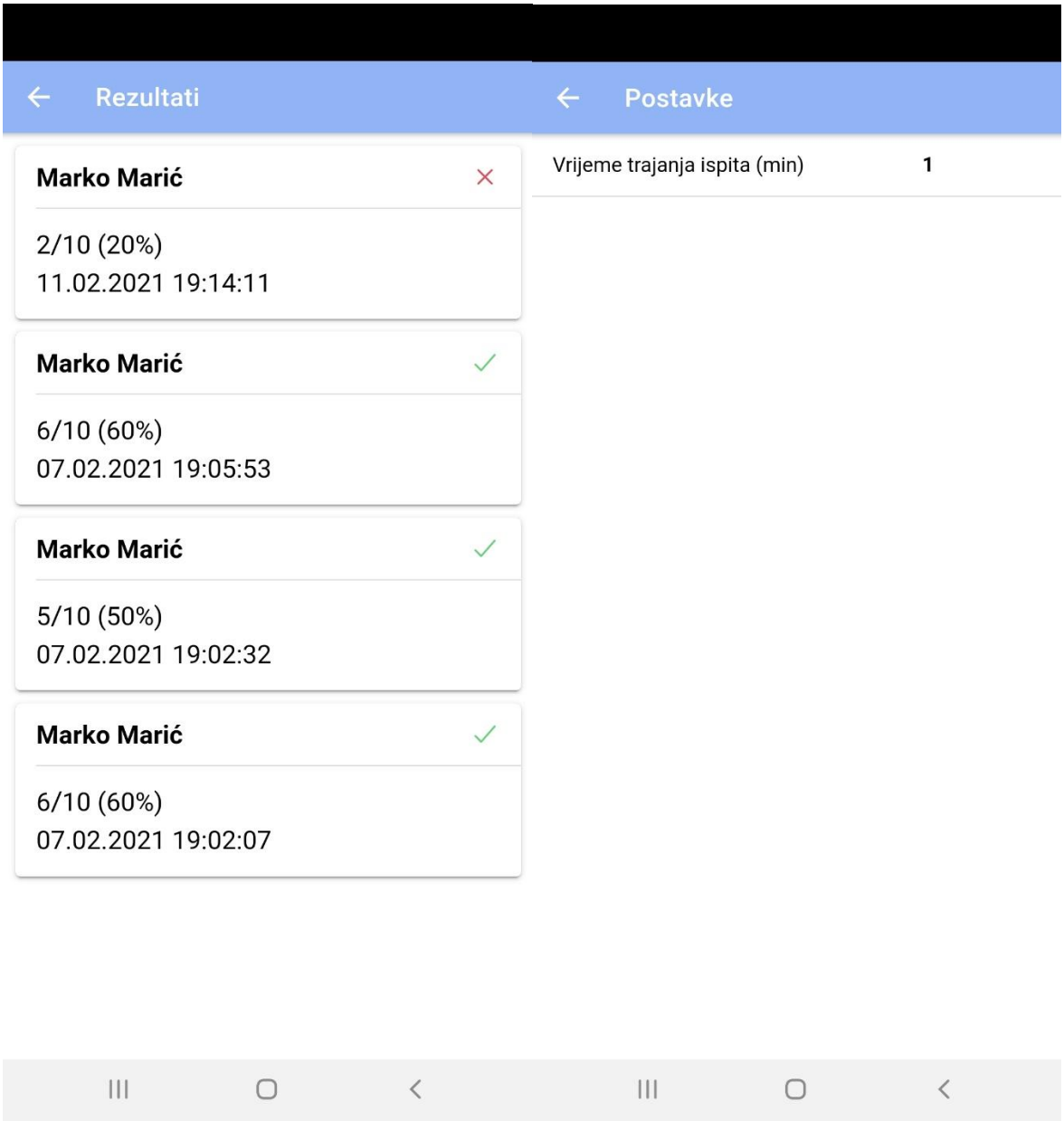
Slika 4.3. Mobilna aplikacija – sučelje *"administratora"*: *"Ispitanici"*

Klikom na ikonu *"pitanja"*, na početnom zaslonu, otvara se popis postojećih pitanja s pripadajućim odgovorima. Popis pitanja može se uređivati, što znači da se pitanja mogu dodavati, brisati, te dodatno uređivati, kako je prikazano na slici. 4.4.



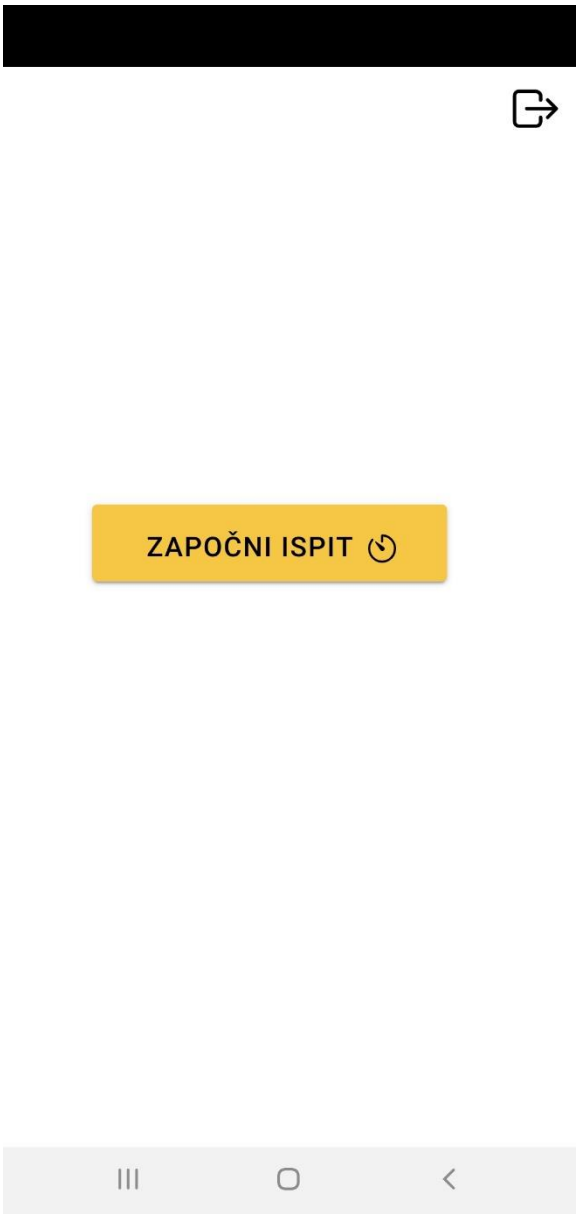
Slika 4.4. Mobilna aplikacija – sučelje *"administratora"*: *"Pitanja"*

Klikom na ikonu *"rezultati"* otvara se popis ispitanika s njihovim pripadajućim rezultatima, datumom i vremenom završetka provjere znanja, kako je prikazano na slici 4.5. Na slici 4.5.također su prikazane i postavke Mobilne aplikacije koje administrator može koristiti.



Slika 4.5. Mobilna aplikacija – sučelje *"administratora"*: *"Rezultati"* i postavke aplikacije

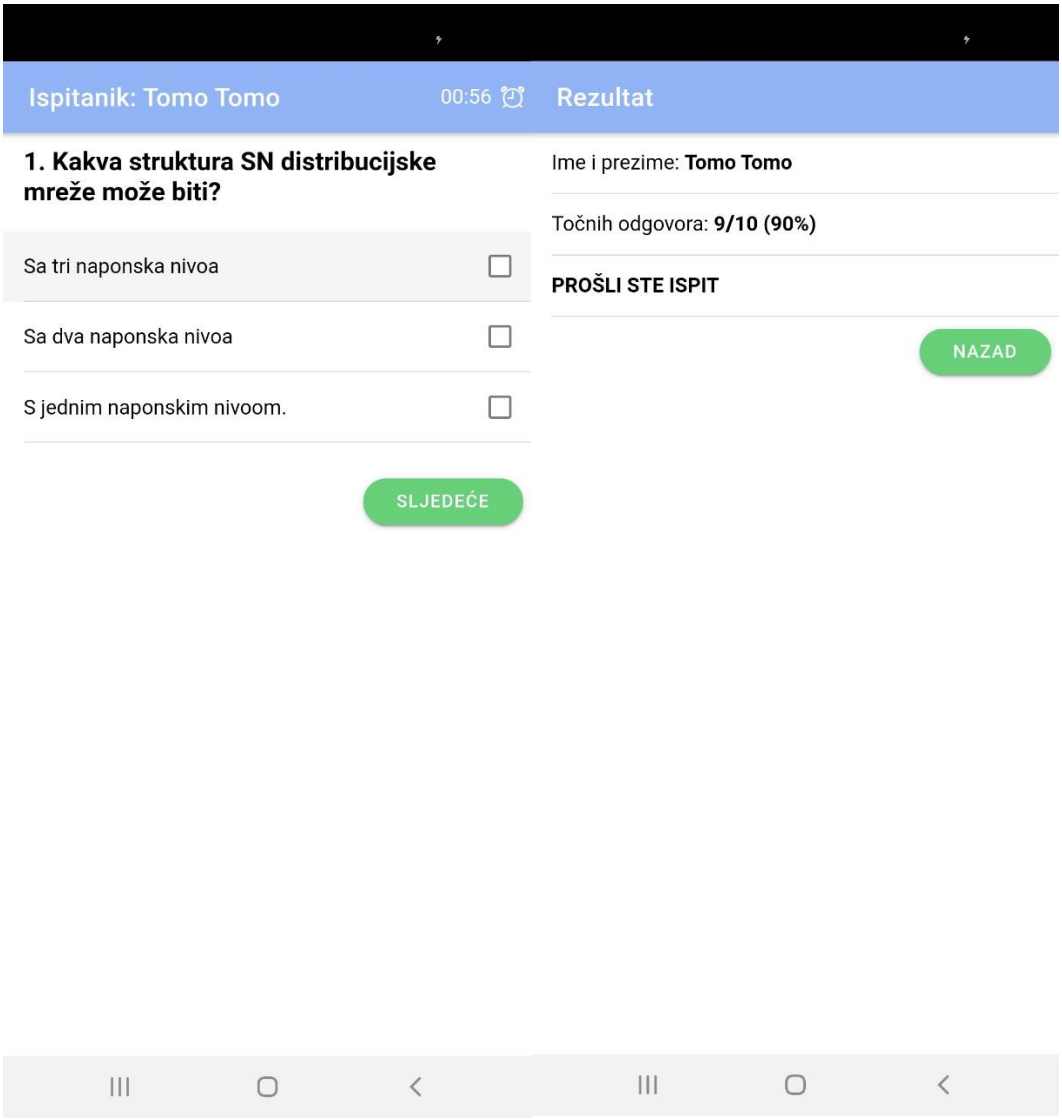
Ukoliko se prijavio *"Ispitanik"*, slika 4.6. prikazuje zaslon s njegovim mogućnostima u radu s Mobilnom aplikacijom.



Slika 4.6. Mobilna aplikacija – sučelje *"ispitanik"*

Sučelje ispitanika prikazuje koje su njegove mogućnosti. Ispitanik ima sljedeće mogućnosti: započeti ispit ili se odjaviti iz Mobilne aplikacije. Klikom na ikonu *"započni ispit"* otvara se zaslon s prvim pitanjem koje je nasumično odabrano iz popisa pitanja, kako je prikazano na slici 4.7. Ispitanik ima mogućnost odabira više odgovora, ali nema mogućnost povratka na prethodna pitanja. Ispitanik odgovara na pitanje potvrdnim okvirom (*engl. checkbox*), nema unosa teksta. Također je ispit vremenski ograničen, ovisno koliko je vremensko trajanje ispita predvidio administrator. Ukoliko ispitanik nije riješio sva pitanja, nakon isteka vremena aplikacija će

automatski završiti ispit, otvara se prikaz rezultata koji je ispitanik postigao, kao što je prikazano na slici 4.7.



Slika 4.7. Mobilna aplikacija – sučelje *"ispitanik"* rješavanje ispita, te postignuti rezultat

Nakon pregleda postignutih rezultata, ispitanik ima mogućnost završetka ispita i povratka na ponovnu provjeru znanja, tj. na početni zaslon, iz kojega se može odjaviti iz Mobilne aplikacije.

5. ZAKLJUČAK

Kako je već ranije u uvodnom dijelu spomenuto i opisano, zadatak rada bio je izrada mobilne aplikacije koja će imati bazu podataka s pitanjima za provjeru znanja pripravnika iz područja elektrotehnike. Mobilna aplikacija izrađena je kao hibridna mobilna aplikacija te se kao takva može implementirati na bilo koji mobilni uređaj, neovisno o platformi na kojoj mobilni uređaj radi. Izrađena je s dva sučelja, administratorsko sučelje u kojem se zadaju pitanja, te prate rezultati svih korisnika, Dok korisničko sučelje ima mogućnost izvršavanja testova i praćenje vlastitih rezultata. Aplikacija je zbog prirode posla i svrhe u kojoj će se koristiti bazirana na području elektrotehnike međutim, pogodna je za bilo koje drugo znanstveno područje. Kako bi se spomenuta tranzicija obavila, dovoljna je samo promjena setova pitanja i pripadajućih odgovora, što je jedna od najvećih prednosti same aplikacije. Pogodna je za brzu i kratku provjeru znanja. Uporaba mobilne aplikacije za provjeru znanja pripravnika praktična je i jednostavna, novi korisnici nakon prolaska kratkih uputa odmah se s razumijevanjem mogu služiti aplikacijom. Iz svega navedenog može se zaključiti kako mobilna aplikacija ima dosta prostora za napredak, od dizajna do kompleksnijih funkcijskih mogućnosti.

6. LITERATURA

[1] **MySQL** dostupno na: https://www.mysql.com [12.02.2021]

https://hr.wikipedia.org/wiki/SQL [12.02.2021]

https://znatko.com/2753/ [12.02.2021]

[2] **Navicat Premium** dostupno na: https://www.navicat.com/en/products/navicat-premium [12.02.2021]

[3] **XAMPP** dostupno na: https://www.apachefriends.org/index.html [12.02.2021]

https://www.lokalna.hr/edukacije/index.php?idr=1113 [12.02.2021]

[4] **Ionic** dostupno na: https://ionicframework.com/docs [12.02.2021]

[5] **Angular** dostupno na: https://angular.io [12.02.2021]

https://zir.nsk.hr/islandora/object/mathos:142/preview [12.02.2021]

[6] **Node.js** dostupno na: https://nodejs.org/en/about/ [12.02.2021]

https://repozitorij.etfos.hr/islandora/object/etfos%3A1267/datastream/PDF/view [12.02.2021]

[7] **JavaScript** dostupno na: https://www.w3schools.com/js/DEFAULT.asp [12.02.2021]

https://www.srce.unizg.hr/files/srce/docs/edu/osnovni-tecajevi/c501_polaznik.pdf [12.02.2021]

[8] **TypeScript** dostupno na: https://www.typescriptlang.org [12.02.2021]

https://core.ac.uk/download/pdf/198165579.pdf [12.02.2021]

[9] **HTML** dostupno na: https://hr.wikipedia.org/wiki/HTML [12.02.2021]

https://tesla.carnet.hr/mod/book/view.php?id=5430&chapterid=885 [12.02.2021]

[10] **CSS** dostupno na: https://www.srce.unizg.hr/files/srce/docs/edu/osnovni-tecajevi/c220_polaznik.pdf [12.02.2021]

SAŽETAK

Mobilna aplikacija za provjeru znanja pripravnika

Cilj ovog diplomskog rada bio je izraditi mobilnu aplikaciju za provjeru znanja pripravnika i njenu pripadajuću bazu podataka na lokalnom serveru. Za realizaciju rada opisane su korištene tehnologije: JavaScript, Ionic, Angular, SQL lite, MySQL, HTML/CSS. Ovako izgrađena mobilna aplikacija može se implementirati na bilo koju platformu mobilnih uređaja, jer je izrađena kao hibridna mobilna aplikacija. Nakon uvoda i opisa korištenih programskih jezika i alata, prikazana je izrada i izgled mobilne aplikacije.

Ključne riječi: mobilna aplikacija, pripravnik, baza podataka, JavaScript, Ionic, Angular, SQL lite, MySQL, HTML/CSS, platforma, mobilni uređaj, hibridna mobilna aplikacija

ABSTRACT

Mobile application for checking the knowledge of trainees

The aim of this thesis was to create a mobile application for testing the knowledge of trainees and its associated database on a local server. The technologies used for the realization of the work are described: JavaScript, Ionic, Angular, SQL lite, MySQL, HTML / CSS. A mobile application built in this way can be implemented on any mobile device platform, because it is designed as a hybrid mobile application. After the introduction and description of the used programming languages and tools, the design and layout of the mobile application are shown.

Keywords: mobile application, trainee, database, JavaScript, Ionic, Angular, SQL lite, MySQL, HTML / CSS, platform, mobile device, hybrid mobile application

ŽIVOTOPIS

Dalibor Perić rođen je 25. srpnja 1983. u Vinkovcima. Osnovnu školu "August Cesarec" završio je 1998. godine u Ivankovu, nakon čega upisuje srednju školu u Vinkovcima "Tehnička škola Ruđera Boškovića", koju završava 2002. godine. U razdoblju između srednje škole i upisa na fakultet odslužio je vojni rok, a potom je od 2003. do 2004. godine obavljao posao u poduzeću "Cestorad d.o.o". Poslije toga zapošljava se u poduzeću "Mobilis d.o.o.", gdje radi s telekomunikacijskim uređajima od 2006. do 2009. godine. Od 2011. do 2013. godine zaposlenik je poduzeća "Cras d.o.o.", gdje se kao mladi inženjer razvija u automatizacijskoj djelatnosti. Od 2014. godine zaposlen je u poduzeću "HEP ODS d.o.o.", gdje obavlja poslove samostalnog inženjera elektrotehnike. Odlično govori Engleski jezik, te ima položen vozački ispit. Uz sve poslove koje je radio možemo reći da se dobro snalazi u samostalnom radu, a isto tako i u timskom radu.

PRILOG

Na CD-u se nalazi prilog s cijelim kodom mobilne aplikacije, te sa bazom podataka.