Implementacija višeplatformske mobilne aplikacije za zapošljavanje studenata koristeći razvojni okvir Flutter

Radočaj, Leo

Sveučilište Jurja Dobrile u Puli
Fakultet Informatike u Puli

Leo Radočaj

Implementacija višeplatformske mobilne aplikacije za zapošljavanje studenata koristeći razvojni okvir Flutter

Završni rad

Pula, rujan 2024.

Sveučilište Jurja Dobrile u Puli
Fakultet Informatike u Puli

Leo Radočaj

Implementacija višeplatformske mobilne aplikacije za zapošljavanje studenata koristeći razvojni okvir Flutter

Završni rad

JMBAG: 0303094823, redovni student
Studijski smjer: Sveučilišni preddiplomski studij informatika
Znanstveno područje: Društvene znanosti
Znanstveno polje: Informacijske znanosti
Znanstvena grana: Informacijski sustavi i informatologija
Kolegij: Programsko inženjerstvo
Mentor: doc. dr. sc. Nikola Tanković

Pula, rujan 2024.

Sažetak

Ovaj završni rad prikazuje razvoj višeplatformske mobilne aplikacije za zapošljavanje studenata koristeći razvojni okvir Flutter. Aplikacija omogućuje korisnicima pregled i prijavu na oglase za poslove prikazane u video formatu, uz mogućnost snimanja video životopisa. Za razvoj front-enda korišten je Flutter, dok je back-end izgrađen koristeći NestJS framework i PostgreSQL bazu podataka. Aplikacija nudi prilagođeno korisničko sučelje za studente i administratore, s mogućnošću registracije, prijave, pregleda oglasa te praćenja statusa prijava. Rad uključuje i analizu korištenih tehnologija te SWOT analizu aplikacije.

Ključne riječi

Flutter, mobilna aplikacija, front-end, back-end, zapošljavanje studenata, NestJS, PostgreSQL, video oglasi, video životopisi, video prijave

Abstract

This thesis presents the development of a cross-platform mobile application for student employment using the Flutter framework. The application allows users to browse and apply for job listings displayed in video format, with the option to record video resumes. The front-end was developed using Flutter, while the back-end was built with the NestJS framework and PostgreSQL database. The application offers a customized user interface for both students and administrators, enabling registration, login, job listing browsing, and tracking application status. The thesis also includes an analysis of the technologies used and a SWOT analysis of the application.

Keywords

Flutter, mobile application, student employment, video resumes, job listings, NestJS, PostgreSQL, front-end development, back-end development

Sadržaj

1. Uvod

Većina studenata traži poslove koji nude fleksibilne radne uvjete kako bi se uskladili s njihovim obvezama i rasporedima. Mobilne aplikacije koje koriste kratke videozapise kao glavni format sadržaja postale su vrlo popularne među studentskom populacijom. Ovaj završni rad će prikazati kako je izrađena mobila aplikacija za pronalaženje poslova gdje su poslovi i studentske prijave predstavljene u video obliku.

Razvoj mobilnih aplikacija za Android i iOS platformu je dugotrajan i zahtjevan proces kada se radi odvojeno, stoga ova aplikacija će biti izrađena koristeći Flutter razvojni okvir koji podržava izradu višeplatformnih aplikacija. Dostupan je kao otvoreni kod, te zajednica aktivno pridonosi razvoju i izradi dodatnih paketa dostupnih programerima. Flutter ima podršku za Material i Cupertino komponente, što dozvoljava razvojnim programerima da izrade aplikacije koje imaju autentičan izgled i osjećaj kao da su rađene za individualne platforme. Programer također može izrađivati svoje UI komponente ukoliko to odgovara zahtjevima, kao što je napravljeno u ovom završnom radu[1].

Obzirom da se razvojni okvir Flutter koristi za izradu *front-end*-a, potrebno je odabrati jezik i framework za izradu *back-end*-a. Kako bi razvoj bio što lakši, a da se ne gubi na kvaliteti, najbolje je odabrati isti ili što sličniji jezik Dart-u. Za izradu *back-end*-a koristimo TypeScript i framework NestJS. TypeScript je proširenje na jezik Javascript, te nam omogućava snažno tipiziranje kao što je to prisutno u Dartu. TypeScript je razvijen od strane Microsofta, dok je NestJS razvijen od strane zajednice uz snažnu podršku Microsofta, Valor Software i ostalih kompanija[3].

Kako bi nam razvoj bio što lakši, koristimo integrirano razvojno okruženje IntelliJ Idea (IDE), koji ima odličnu podršku za rad s Flutterom i TypeScript aplikacijama.

2. Analiza korisničkih zahtjeva

Student prilikom korištenja aplikacije mora se registrirati unosom studentske email adrese, informacije o studiju i godini, te odabirom kategorija poslova koji ga zanimaju. Dodatno, student ima mogućnost snimanja prijave za posao u video formatu, što predstavlja inovativan način predstavljanja potencijalnim poslodavcima.

Nakon uspješne prijave u aplikaciju, student može pregledavati poslove prikazane u video formatu. Klikom na određeni oglas, prikazuju se dodatne informacije o poslu, uključujući detaljan tekstualni opis i podatke o kompaniji koja je oglas objavila.

Aplikacija također sadrži stranicu na kojoj student može pratiti status svojih prijava, pregledavati poslove za koje se prijavio u prošlosti, kao i poslove koji su u tijeku. Na stranici kompanije, moguće je vidjeti sve oglase koje je određena kompanija objavila, kao i recenzije prethodnih radnika.

U postavkama aplikacije, student može dodati novi životopis, prilagoditi ga specifičnim vrstama poslova, promijeniti trenutnu lokaciju za koju želi vidjeti poslove, te ažurirati informacije o studiju.

Potrebno je imati neko sučelje gdje će se moći vršiti pregled kandidata, izrađivati profili kompanije te izrađivati poslovni oglasi.

2.1. SWOT Analiza

Snage

- **Interesantan format oglasa**
  - Poslovi su predstavljeni u obliku videa, što čini proces traženja posla zanimljivijim i privlačnijim studentima, osobito mlađim generacijama koje preferiraju vizualni sadržaj.
- **Lakoća i fleksibilnost razvoja**
  - Korištenje Flutter okvira omogućava brzu izradu aplikacije koja podržava više platformi (Android, iOS), čime se osigurava široka dostupnost studentima na različitim uređajima.
- **Personalizacija profila**
  - Mogućnost snimanja video životopisa i prilagođavanja profila prema osobnim interesima omogućava studentima da se bolje predstave poslodavcima i povećaju šanse za zapošljavanje.

Slabosti

- **Zahtjevnost snimanja video sadržaja**
  - Snimanje video životopisa i prilagodba videa za svaki posao može biti vremenski zahtjevna aktivnost za studente, osobito one koji nisu navikli na snimanje videa ili nemaju potrebne tehničke vještine. Također snimanje video oglasa može biti zahtjevno za poslodavce.
- **Ograničenost na studentsku populaciju**
  - Trenutna ciljna skupina su samo studenti, što smanjuje potencijal aplikacije da privuče širu bazu korisnika izvan studentskog okvira.

Prilike

- **Proširenje na širu populaciju**
    - Aplikacija se može proširiti kako bi obuhvatila ne samo studente već i druge korisnike koji traže poslove
- **Unapređenje video alata**
    - Dodavanje alata unutar aplikacije koji bi studentima olakšali snimanje i uređivanje video sadržaja moglo bi smanjiti barijeru korištenja videa kao glavnog formata.
- **Internacionalizacija**
    - Proširenje aplikacije na međunarodna tržišta omogućilo bi povezivanje studenata s poslodavcima izvan lokalnih granica, čime bi se povećale šanse za pronalazak posla.

Prijetnje

- **Konkurencija postojećih aplikacija**
    - Na tržištu već postoje poznate aplikacije za zapošljavanje koje imaju veliku bazu korisnika. Ako te platforme uvedu slične funkcionalnosti, poput video oglasa za poslove, aplikacija bi mogla izgubiti konkurentsku prednost.
- **Loša kvaliteta video sadržaja**
    - Loše snimljeni videi se neće svidjeti ni studentima ni poslodavcima. To može smanjiti angažman korisnika, ostaviti loš dojam o aplikaciji, te otežati povezivanje studenata s poslodavcima zbog neprofesionalnog predstavljanja i nemogućnosti prikazivanja ključnih informacija na kvalitetan način.

3. Korištene Tehnologije

Obzirom da izrađujemo mobilnu aplikaciju i back-end odabrani su slični jezici kako bi razvoj bio lakši.

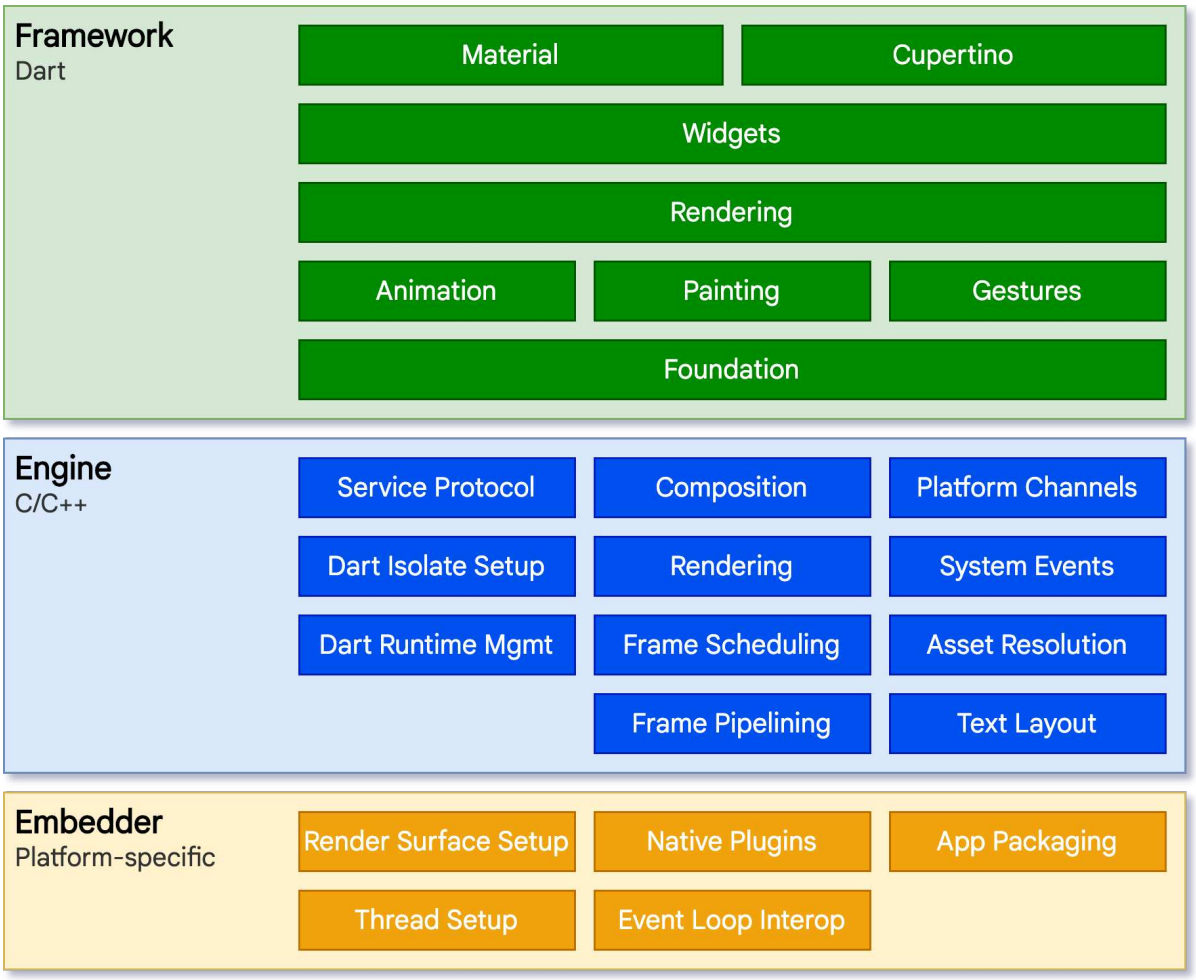
3.1. Dart

Dart je programski jezik razvijen od strane Google-a, razvijen je s ciljem da bude prigodan za izradu front-end aplikacija. Baš zbog toga je napravljen da je što sličniji JavaScriptu - kako bi se postojeći developeri brže naviknuli, ali uzimajući elemente iz Jave poput : objektno orijentiranosti i snažnog tipiziranja. Inicijalno je napravljen kao zamjena za Javascript, međutim zajednica nije bila zainteresirana za to, te je fokus prebačen u izradu SDK-a za izradu višeplatformnih aplikacija - Flutter.
Vrijedno je spomenuti da Dart ima odličnu podršku za izvršavanje asinkronog kod-a, te za ima podršku za *null safety* , što osigurava sigurnost pri izvršavanju koda jer smo sigurni da neke variable ne mogu imati *null* vrijednost, osim ako eksplicitno to ne definiramo.
Dart podržava dva načina prevođenja: Ahead-of-time (AOT) i Just-in-time (JIT). JIT nam je izuzetno koristan, jer nam omogućava da koristimo *hot-reload* , koji će primijeniti novo napisan kod, zadržavajući prethodno stanje aplikacije[2].

3.2. Flutter

Flutter je razvojni okvir otvorenog koda izrađen od strane Googla-a, sa snažnom kontribucijom od strane zajednice. Koristi se za izradu višeplatformnih aplikacija, te u ovom trenutku podržava Android, iOS, Web, Windows, Linux te MacOS platforme. Arhitektura Fluttera podijeljena je na nekoliko slojeva, pri čemu svaki sloj ima svoju ulogu u cjelokupnom sustavu[1].

1. **Sloj okvira (Framework Layer):** Ovo je najviši sloj u arhitekturi Fluttera i nudi bogat skup paketa koje omogućuju izgradnju korisničkih sučelja, upravljanje stanjima, te interakciju s uređajem. Programeri najčešće rade unutar ovog sloja. Ovaj sloj je napisan u Dart programskom jeziku i uključuje grafičke komponente (widgets), i animacije.
2. **Sloj renderiranja (Rendering Layer):** Ovaj sloj je odgovoran za pretvaranje grafičkih komponenti u niže razine prikaza koje se mogu iscrtati na ekranu. On prepoznaje strukturu korisničkog sučelja i obavlja proračune potrebne za prikaz elemenata na zaslonu. Renderiranje koristi engine sloj kako bi prikazao te elemente na uređaju.
3. **Engine sloj (Engine Layer):** Ovaj sloj je napisan u C/C++, te je odgovoran za iscrtavanje grafike i pokretanje Dart koda. Flutter engine koristi Impeller engine za prikaz grafike.
    a. **Platformski sloj (Platform Layer):** Ovaj sloj omogućava da Flutter aplikacija komunicira s platformom na kojoj se pokreće, poput pristupa kameri, senzorima i sl. U većini slučajeva, pakete koje koristimo već obavljaju ovu komunikaciju, tako da mi ne moramo sami raditi tu implementaciju[10].



Slika 1. Izvor: https://docs.flutter.dev/resources/architectural-overview

3.3. TypeScript

TypeScript je programski jezik često korišten u razvoju aplikacija, posebno kada se radi o većim i složenim projektima. Kao nadogradnja na JavaScript, TypeScript uvodi snažno tipiziranje, što omogućava ranije otkrivanje grešaka u kodu i olakšava održavanje i izradu aplikacija. Osim što se koristi na front-end-u, u kombinaciji s razvojnim okvirima poput: Angular, Vue.js i React, može se koristiti i na backendu sa Node.js, NestJS i ostalima. TypeScript se prevodi u JavaScript te se kao takav može koristiti u zamjenu JavaScriptu, odnosno može se kombinirati[3].

3.4. NestJS

NestJS je razvojni okvir temeljen na Node.js koji koristi TypeScript kao jezik, osmišljen je kako bi omogućio razvoj server-side aplikacija. Inspiriran je Angularom, kako bi se olakšao prelazak front-end programera na back-end. NestJS se temelji na modularnom dizajnu, omogućujući programerima da podijele aplikaciju na manje, izolirane module, što značajno poboljšava organizaciju i održavanje koda. Svaki modul sadrži više komponenti, poput: kontrolera, servisa i entiteta. NestJS se integrira s popularnim paketima i alatima kao što su TypeORM, Mongoose, GraphQL, Passport i drugi, omogućavajući brzi razvoj aplikacija s ugrađenim funkcionalnostima poput ORM-a, autentifikacije, autorizacije, i validacije podataka. Ovo značajno smanjuje vrijeme potrebno za razvoj složenih aplikacija[4].

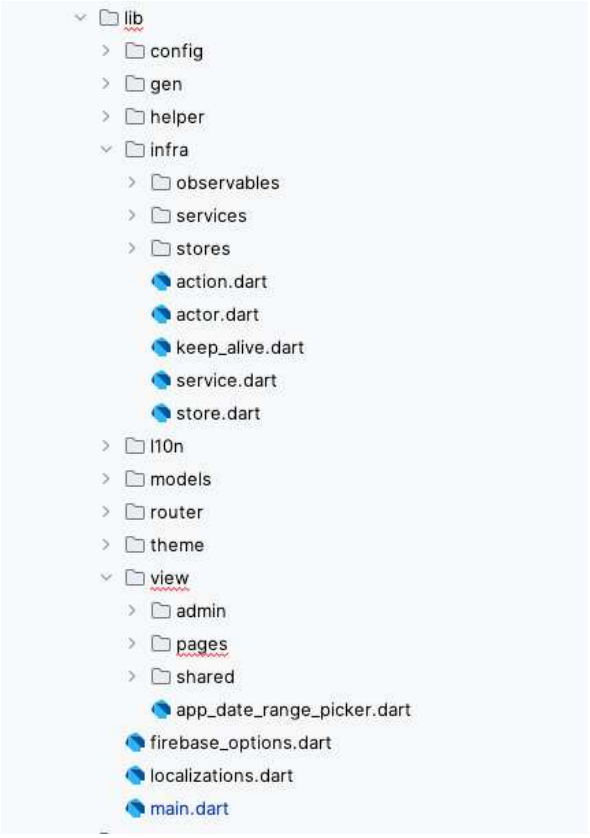
3.5. PostgreSQL

PostgreSQL je jedna od najpopularnijih relacijskih baza podataka, poznata po svojoj pouzdanosti, skalabilnosti i količini značajka. Kada se koristi u kombinaciji s NestJS-om, najčešće koristimo TypeORM, alat koji nam olakšava rad s bazama podataka. TypeORM omogućuje mapiranje klasa u TypeScriptu na tablice u relacijskoj bazi podataka[6]. Time se eliminira potreba za pisanjem složenih SQL upita, te imamo pristup tablicama kroz TypeScript jezik[5].

4. Implementacija aplikacije

Potrebno je definirati arhitekturu u smislu postavljanja direktorija, te odabrati način upravljanja stanjem unutar mobilne aplikacije. Dobro postavljen direktorij nam olakšava razvoj te buduće nadogradnje.

4.1. Struktura direktorija front-end



Slika 2. Prikaz direktorija unutar Flutter projekta

Objašnjenja za svaki direktorij i poddirektorij(Slika 2.):

1. **lib/** - Glavni direktorij za izvorni kod aplikacije.
2. **config/** - Sadrži konfiguraciju aplikacije.
3. **gen/** - Direktorij rezerviran za automatski generirani kod. U Flutter projektima, to su datoteke generirane od strane build_runner alata.
4. **helper/** - Pomoćni kod koji se koristi na više mjesta u aplikaciji
5. **infra/** - Sadrži infrastrukturu/arhitekturu aplikacije, odnosno svu poslovnu logiku.

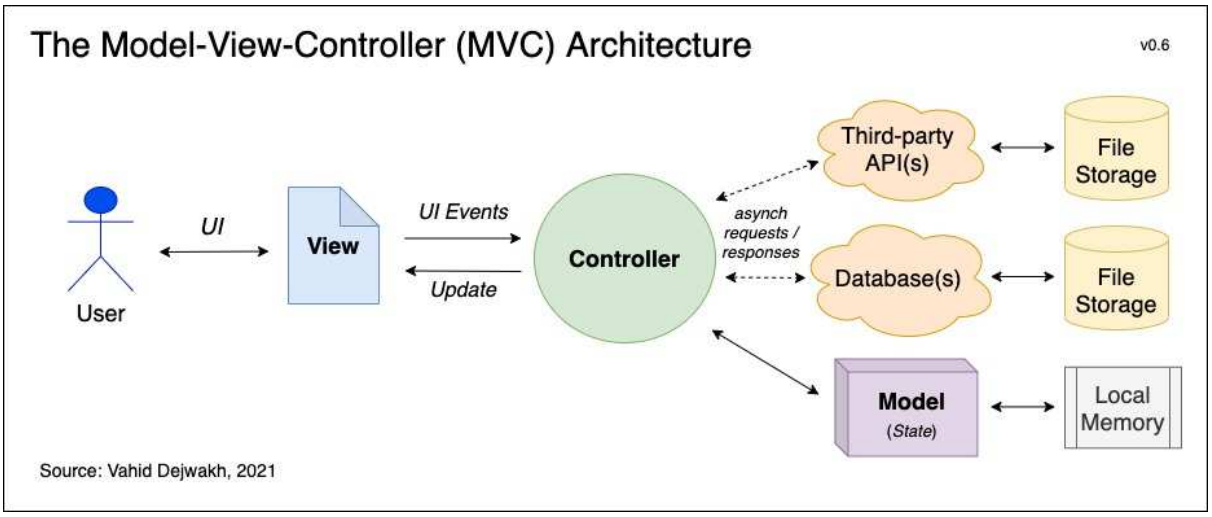
    - **observables/** - Sadrži klase koje su reprezentativne sadržaju koji će određeni ekran promatrati.
    - **services/** - Ovdje se nalaze servisi koji su potrebni za rad aplikacije, poput servisa za lokalnu pohranu podataka te servisa za komunikaciju s API-evima.
    - **stores/** - Sadrži klase koje upravljaju stanjem aplikacije.
6. **l10n/** - Sadrži datoteke i resurse potrebne za podršku više jezika. Korisimo .arb format za pisanje prijevoda.
7. **models/** - Sadrži modele podataka koje aplikacija koristi..
8. **router/** - Sadrži datoteke koje definiraju navigaciju kroz aplikaciju.
9. **theme/** - Ovdje se nalaze datoteke koje definiraju temu i stilizaciju aplikacije, poput boja, fontova i drugih vizualnih aspekata.
10. **view/** - Ovaj direktorij sadrži sve vizualne elemente aplikacije, uključujući UI komponente i stranice:
    - **pages/** - Ovdje se nalaze različite stranice aplikacije. Svaka stranica može biti zasebna datoteka ili poddirektorij.
    - **shared/** - Sadrži zajedničke UI komponente (widgets) koji se koriste na više stranica u aplikaciji.
11. **firebase_options.dart** -Sadrži konfiguraciju za integraciju s Firebase-om.
12. **main.dart** - Glavna ulazna točka aplikacije.

Ovako strukturirani direktoriji omogućuju organiziran i modularan razvoj aplikacije, gdje je svaki dio koda jasno odvojen prema svojoj funkcionalnosti. Time se olakšava razvoj, jer svaki direktorij ima specifičnu ulogu, čineći projekt preglednijim i lakšim za navigaciju.

4.2. Upravljanje stanjem unutar aplikacije

Za upravljanje stanjem unutar aplikacije koristimo MVC (Model-View-Controller) [8] arhitekturu, u kombinaciji s paketom **riverpod**. Riverpod omogućava jednostavno i efikasno upravljanje stanjem aplikacije na modularan način, neovisno o Widgetima. Zahvaljujući njegovom reaktivnom pristupu, svaka promjena stanja automatski ažurira korisničko sučelje bez potrebe za ručnim

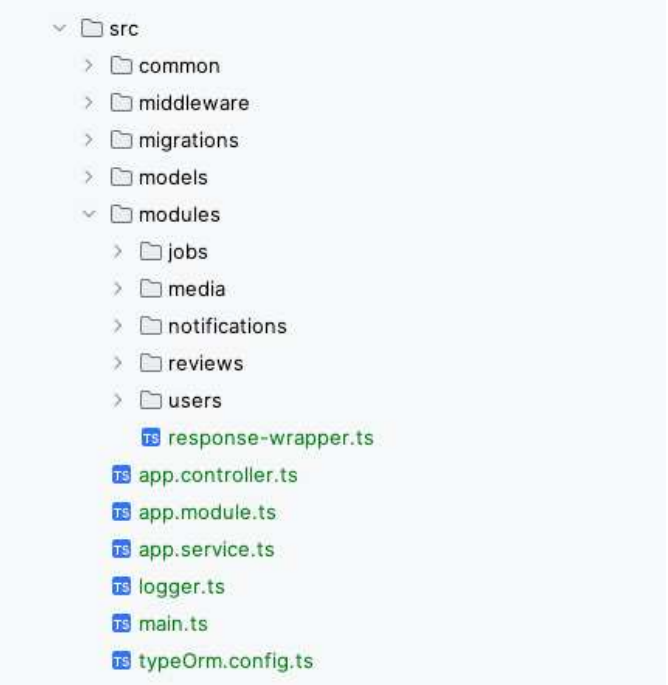
osvježavanjem ili eksplicitnim osvežavanjem sa strane programera. Ova kombinacija olakšava organizaciju koda i održavanje aplikacije [7].



Slika 3. Prikaz MVC Arhitekture[8]

- **Model**: Ova komponenta predstavlja podatke aplikacije. Model uključuje informacije poput poslova, statusa prijava, korisničkih podataka.
- **View**: View je korisničko sučelje s kojim student interagira. On je odgovoran za prikaz podataka koje kontroler šalje, ali ne sadrži nikakvu logiku osim prikazivanja tih podataka.
- **Controller**: Kada student napravi neku akciju (npr. prijavi se na posao, ažurira profil), kontroler prima te zahtjeve, a zatim vraća ažurirane podatke prema View-u za prikaz.

4.3. Struktura direktorija back-enda



Slika 4. Prikaz direktorija unutar NestJS projekta

1. **common**: Sadrži pomoćne funkcije i konstante koje se koriste u više dijelova aplikacije.
2. **middleware**: Ovdje se nalaze *middleware* funkcije koje presreću zahtjeve ili odgovore između klijenta i servera. *Middleware* se koristi za obavljanje zadataka poput autentifikacije, autorizacije, zapisivanje zahtjeva, ili modifikacije podataka[4].
3. **migrations**: Ovdje imamo migracije koje koristimo za početno punjenje baze s podacima.
4. **models**: Sadrži klase koje se mapiraju u tablice pomoću TypeORM-a.
5. **modules**: Ovaj direktorij sadrži različite funkcionalne module aplikacije. Svaki modul je organiziran u zasebnom poddirektoriju koji sadrži specifične resurse kao što su kontroleri, servisi, i modele za taj modul. Na primjer:
    - **jobs**: Modul za upravljanje poslovima.
    - **media**: Modul za upravljanje multimedijalnim sadržajem.
    - **notifications**: Modul za slanje i upravljanje obavijestima.
    - **reviews**: Modul za upravljanje recenzijama.

    - **users**: Modul za upravljanje korisnicima.
6. **main.ts**: Glavna ulazna točka aplikacije. Ovdje se nalazi kod za pokretanje aplikacije.
7. **typeOrm.config.ts**: Ova datoteka sadrži konfiguraciju za **TypeORM**, Ovdje su definirani podaci o konekciji, entiteti, migracije i drugi parametri povezani s bazom podataka.

4.4. Postavljanje baze podataka koristeći TypeORM

Pri pokretanju NestJS aplikacije, TypeORM će se automatski povezati na bazu podataka i mapirati klase definirane u kodu na odgovarajuće tablice u bazi. Nakon toga, TypeORM će izvršiti migracije kako bi ažurirao strukturu baze podataka. URL za povezivanje na bazu podataka pohranjen je u datoteci .env, a pristupa mu se putem ConfigService-a, što omogućava jednostavnu konfiguraciju.

```
TypeOrmModule.forRootAsync( options: {
    imports: [ConfigModule],
    useFactory: (configService: ConfigService) => {
        return {
            type: "postgres",
            url: configService.get("DATABASE_URL"),
            migrations: ["dist/migrations/*{.ts,.js}"],
            entities: ["dist/models/*.entity{.ts,.js}"],
            autoLoadEntities: true,
            logging: ["query", "error"],
            logger: "file",
        };
    },
    inject: [ConfigService],
}),
```

Slika 5. Prikaz konfiguracije TypeORM-a

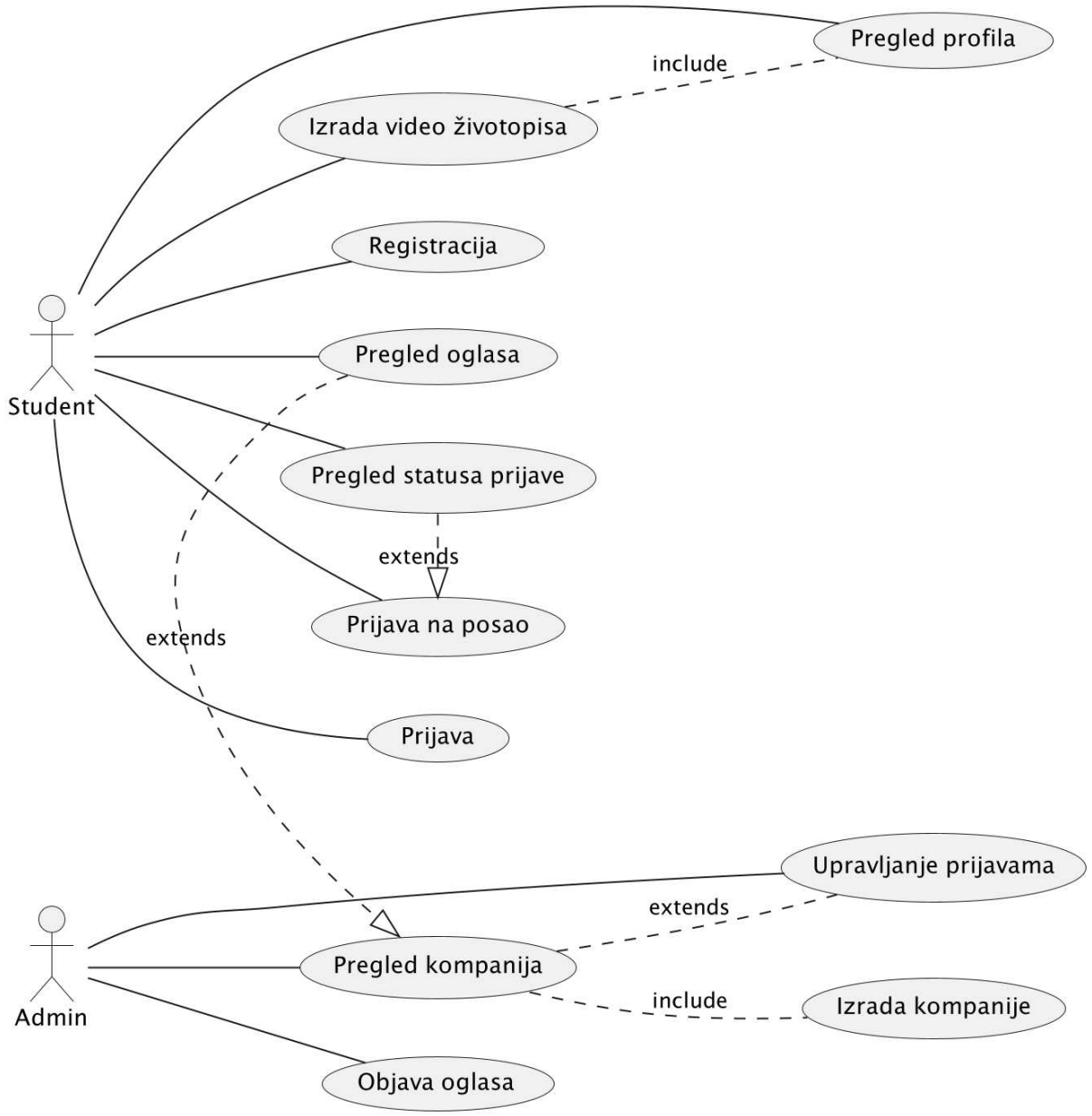
```
@Entity()  Show usages  new *
export class UserEntity extends BaseEntity {
    @PrimaryColumn()
    firebase_uid: string;


    @Column( options: {nullable:true})
    email: string;

    @Column( options: {
        type: "enum",
        enum: UserType,
        default: UserType.STUDENT,
    })
    type: UserType;

    @Exclude()
    @Column( options: {
        type: "text",
        array: true,
        default: [],
    })
    notification_tokens: string[];
}
```

Slika 6. Prikaz entiteta "UserEntity"

5. Korisničko sučelje

5.1. Mobilna Aplikacija

Use-case dijagram

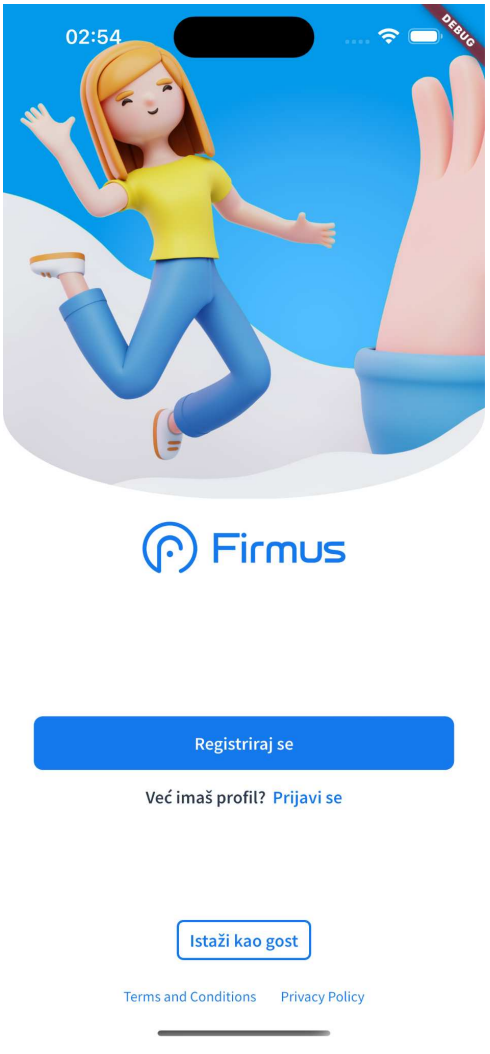


Slika 36. Use-case dijagram mobilne aplikacije

Stranica dobrodošlice

Korisnik može odabrati jednu od tri opcije: ulogirati se, registrirati novi profil ili pregledavati poslove kao gost. Ako odabere pregledavanje kao gost, imat će

ograničen pristup funkcionalnostima – moći će vidjeti oglase, ali neće imati mogućnost prijave na posao niti pristup drugim opcijama dostupnim registriranim studentima. Ova opcija omogućuje korisnicima da istraže aplikaciju prije nego se odluče za kreiranje profila.



Slika 7. Stranica dobrodošlice

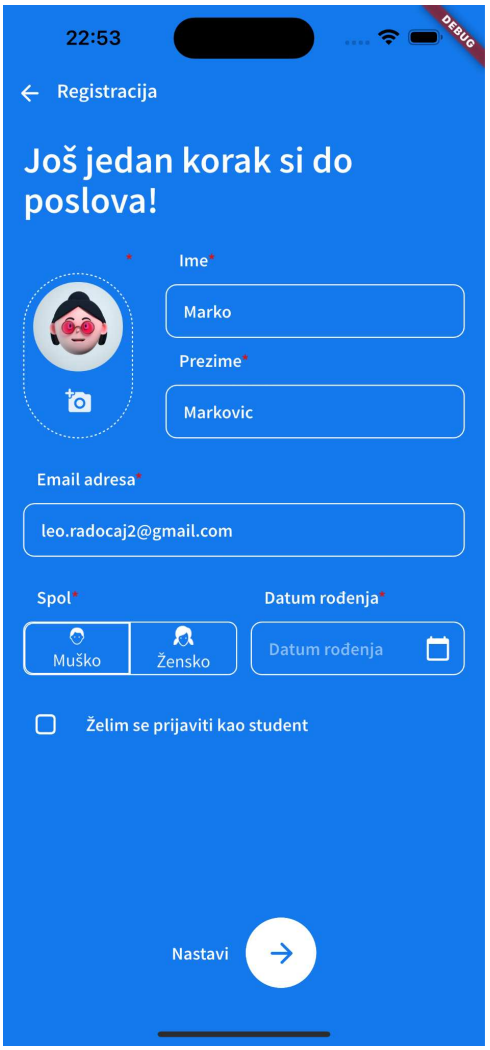
Registracija

Kako bi se ispunili svi poslovni zahtjevi, registracija korisnika izvedena je u nekoliko koraka, pri čemu korisnik ispunjava niz obrazaca s potrebnim podacima. Za implementaciju i validaciju ovih obrazaca koristi se paket **flutter_form_builder**[1], koji omogućuje jednostavno kreiranje formi uz ugrađene opcije za validaciju unosa.

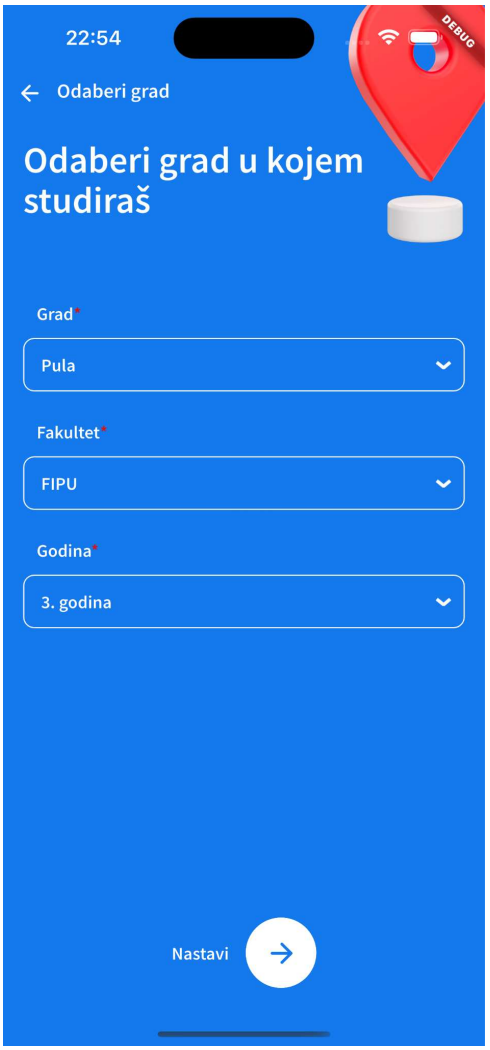
Korisnicima je također omogućeno dodavanje profilne slike, pri čemu se koristi paket **image_picker**[2] koji omogućava odabir slike iz galerije. Podaci o dostupnim pozicijama za rad, kao i popis fakulteta, dohvaćaju se izravno s našeg back-enda.

Nakon što korisnik završi proces registracije, na njegovu e-mail adresu bit će poslana pozivnica s linkom za potvrdu registracije. Klikom na taj link, korisnik će biti preusmjeren u aplikaciju, čime će se potvrditi njegova registracija. Za autentifikaciju korisnika koristi se **firebase_auth**[3], koji omogućava sigurnu i jednostavnu prijavu putem "magic linka". Na serverskoj strani, guardovi su pravilno postavljeni kako bi se osigurala zaštita i pravilna autorizacija korisnika tijekom korištenja aplikacije.

[1] https://pub.dev/packages/flutter_form_builder
[2] https://pub.dev/packages/image_picker
[3] https://pub.dev/packages/firebase_auth



Slika 8. Korak registracije – osobni podaci



Slika 9. Korak registracije – podaci o fakultetu

Slika 10. Korak registracije – odabir pozicija

Slika 11. Korak registracije – dodavanje poslovnog iskustva

```
export class UserController {
    async createStudent(
        @Body() createStudent: CreateStudentDTO,
        @Request() req,
        @UploadedFiles()
            files: {
            profile_picture: Express.Multer.File[];
            thumbnail?: Express.Multer.File[];
            video?: Express.Multer.File[];
        },
    ) {
        if (createStudent.anon) {
            const resp : StudentEntity  = await this.userService.createAnonymousStudent(
                req.user.user_id,
            );
            return {data: resp};
        }
        let resp : StudentEntity  = await this.userService.createStudent(
                createStudent as CreateStudentDTO,
                req.user.user_id,
                req.user.email,
                 files: {
                    profile_picture: files.profile_picture?.[0]?.buffer,
                    thumbnail: files.thumbnail?.[0]?.buffer,
                    video: files.video?.[0]?.buffer,
                },
            );

        return {data: resp};
    }
```

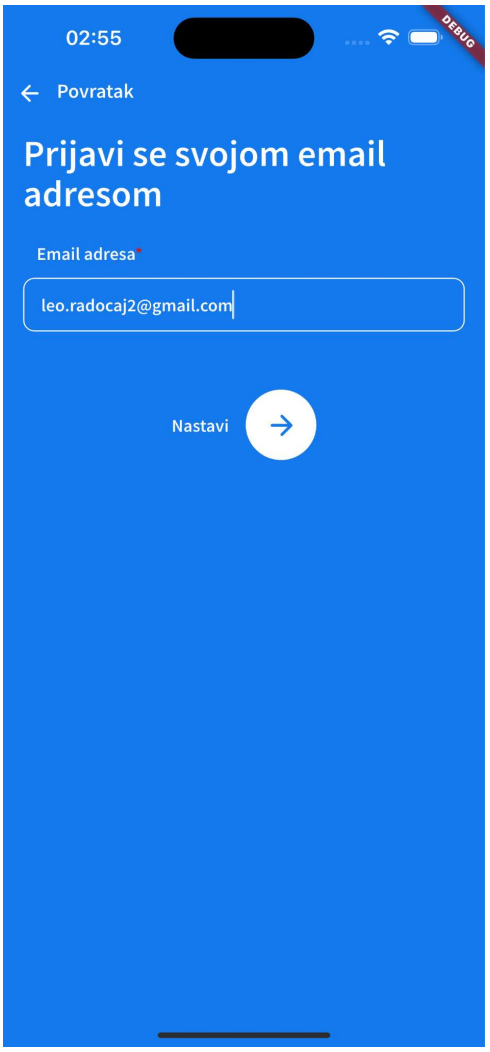
Slika 12. Ruta koja se poziva pri registraciji studenta

```
@Injectable()  Show usages  SempaiEcchi *
export class FirebaseEmailVerified implements CanActivate {
    canActivate(  no usages  SempaiEcchi *
        context: ExecutionContext,
    ): boolean | Promise<boolean> | Observable<boolean> {
        const request = context.switchToHttp().getRequest();
        if (!request.user.email_verified) {
            throw new UnauthorizedException( objectOrError: "Email not verified");
        }

        return true;
    }
}
```

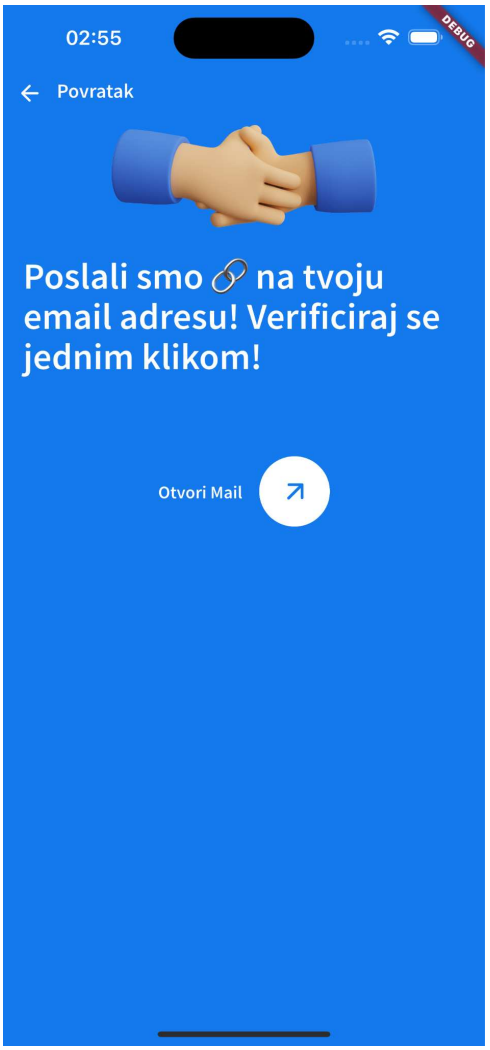
Slika 13. Guard koji provjerava da je student potvrdio mail

Prijava

Nakon što student unese svoju email adresu, biti će mu poslana poveznica putem koje će se moći prijaviti u aplikaciju. Klikom na tu poveznicu, student će biti autentificiran i preusmjeren u aplikaciju, bez potrebe za unosom lozinke, čime se pojednostavljuje proces prijave i povećava korisnička praktičnost.



Slika 14. Unos email adrese za prijavu



Slika 15. Potvrda poslane poveznice na email

```
Future<void> signInWithoutPassword([String? email]) {
  return FirebaseAuth.instance.sendSignInLinkToEmail(
      email: email! ,
      actionCodeSettings: ActionCodeSettings(
        iOSBundleId: "com.firmusapp.jobs",
        androidPackageName: "com.firmusapp.jobs",
        androidMinimumVersion: "1",
        handleCodeInApp: true,
        dynamicLinkDomain: "firmuswork.page.link",
        url: "https://firmuswork.page.link",
      ));
```

Slika 16. Kod koji poziva slanje poveznice na email

Oglasi

Klikom na određeni oglas, student može vidjeti detaljan opis posla(Slika 18.), prijaviti se na oglas te pristupiti informacijama o kompaniji koja je objavila taj oglas. Svi video oglasi dohvaćaju se s našeg back-enda, ovisno o preferencijama koje je student prethodno postavio u svom profilu.

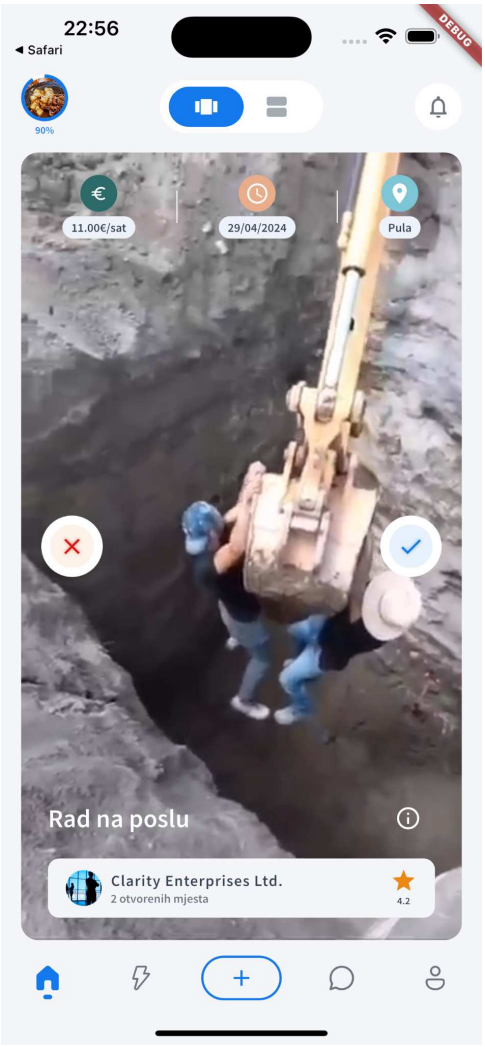
Student može birati između dva načina prikaza oglasa – kao videa ili kao lista. Za prikaz i reprodukciju videa unutar aplikacije koristi se paket **video_player**[4], koji omogućava pregledavanje videa sa poslužitelja.

Dodatno, kako bi se olakšala interakcija s oglasima – povlačenjem udesno student se prijavljuje na posao, dok povlačenjem ulijevo označava da nije zainteresiran za taj posao. Ova funkcionalnost omogućava brzo i intuitivno pregledavanje i upravljanje oglasima.

```
@override
Future<JobsResponse> fetchJobs(FetchJobsRequest jobsRequest) async {
  final context = navkey.currentContext!;

  final initial = await httpService.request(jobsRequest, converter: (data) {
    return JobsResponse.fromMap(data);
  });
  logger.info("got ${initial.offers.length} offers");

  final futures = <Future>[];
  for (var element in initial.offers) {
    final f2 = precacheImage(
        CachedNetworkImageProvider(element.company.logoUrl), context);

    futures.add(f2);

    if (!element.isVideo) {
      final f1 =
          precacheImage(CachedNetworkImageProvider(element.url), context);
      futures.add(f1);
    }
  }

  Future.wait(futures);
  return initial;
}
```

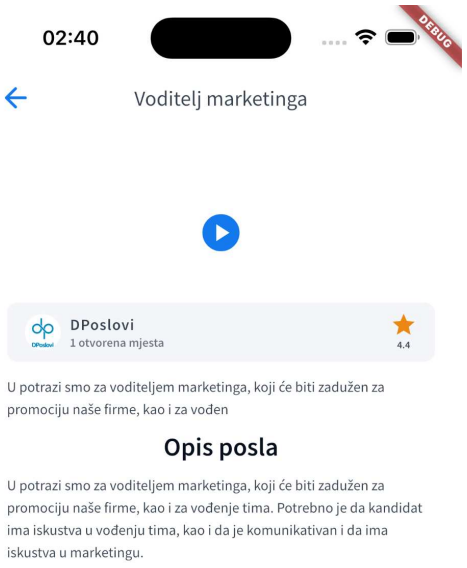
Slika 35. Kod za dohvaćanje oglasa sa poslužitelja

[4] https://pub.dev/packages/video_player



Slika 17. Prikaz video oglasa

Video oglas u aplikaciji (Slika 17.) pruža korisnicima detaljne informacije o dostupnom radnom mjestu kroz vizualno privlačan format. Na ekranu je prikazan posao uz ključne informacije poput plaće, koja iznosi 11,00 €/sat, roka za prijavu (29/04/2024), i lokacije obavljanja posla, u ovom slučaju Pula. Na dnu oglasa prikazana je kompanija koja je objavila oglas – *Clarity Enterprises Ltd.*, zajedno s ocjenom kompanije od 4,2 zvjezdice te brojem dostupnih radnih mjesta. Korisnik može jednostavno odlučiti želi li se prijaviti ili ne povlačenjem lijevo za odbijanje ili desno za prijavu.



Slika 18. Pregled detalja o oglasu

Detaljniji pregled oglasa pruža korisnicima dublji uvid u opis radne pozicije, uključujući specifične zadatke i odgovornosti koje posao zahtijeva. Ovdje korisnici mogu saznati više o prirodi posla, radnim uvjetima, očekivanjima od kandidata te dodatnim pogodnostima koje kompanija nudi(Slika 18.).

```
@Get( path: "/job-opportunities") no usages  ± SempaiEcchi *
@UseGuards(...basicAuthGuards, ResolveStudent)
async jobOpportunities(@Ip() ip, @Req() req) {
    const isAnon : boolean  = !req.student;

    const location : LocationEntity  = (req.student as StudentEntity).location;


    const jobs : JobOpportunityEntity[]  =
        await this.jobsService.getJobOpportunitiesForStudent(location?.latLng(), req.student);

    return {
        data: {
            job_opportunities: jobs ?? [],
        },
    };
}
```

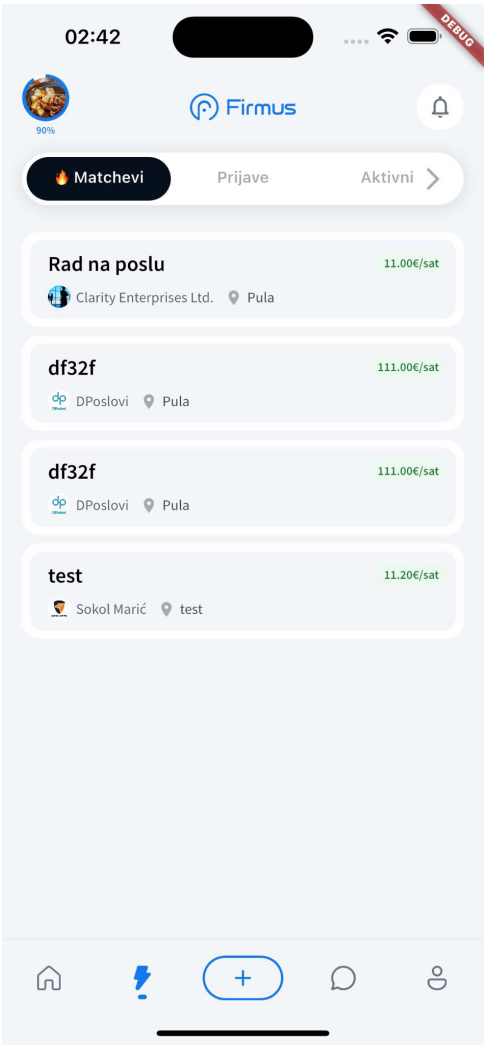
Slika 19. Ruta koja se poziva kada student otvori stranicu s oglasima

```
const lng : number  = location?.lng || 0;
const lat : number  = location?.lat || 0;

const kilometers : 30  = 30;
allJobs = await this.jobOpportunitiesRepository.createQueryBuilder( alias: "job")
    .where( where: "job.visible = true")
    .leftJoinAndSelect( property: "job.company", alias: "company").leftJoinAndSelect( property: "job.media", alias: "media")
    .leftJoinAndSelect( property: "company.logo", alias: "logo")
    .leftJoinAndSelect( property: "job.jobProfile", alias: "jobProfile")
    .leftJoin(
        JobApplicationEntity,
        alias: 'viewedJob',
        condition: 'viewedJob.id_job = job.id AND viewedJob.student_id = :student_id',
        parameters: {student_id}
    )
    .where( where: 'viewedJob.id IS NULL')
    .andWhere( where: 'job.job_profile_id IN (:...jobProfiles)', parameters: {jobProfiles})
    .andWhere( where: "ST_Distance_Sphere(ST_MakePoint(job.location.coordinates[0], job.location.coordinates[1])," +
        " ST_MakePoint(:lng, :lat)) < :kilometers * 1000", parameters: {lng, lat, kilometers})
    .orderBy( sort: `ARRAY_POSITION(:jobProfiles::int[], job.job_profile_id)`, order: 'ASC').setParameters({
        student_id,
        jobProfiles
    })
    .getMany();
```

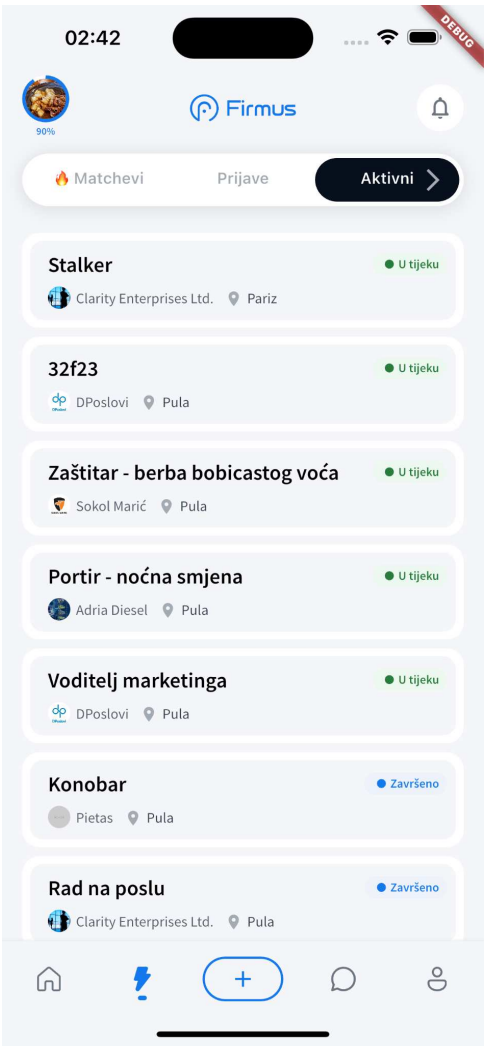
Slika 20. Upit na bazu koji će odabrati poslove koji se prikazuju studentu

Aktivni poslovi

Student može pregledati popis poslova koji su u tijeku (Slika 22.), poslove na koje se prijavio(Slika 23.), kao i poslove gdje je poslodavac odobrio njegovu prijavu, ali radni odnos još nije započeo(Slika 21.). Ova funkcionalnost omogućuje studentima da prate status svih svojih prijava, bilo da su u fazi razmatranja, u tijeku, ili čekaju finalizaciju radnog odnosa.



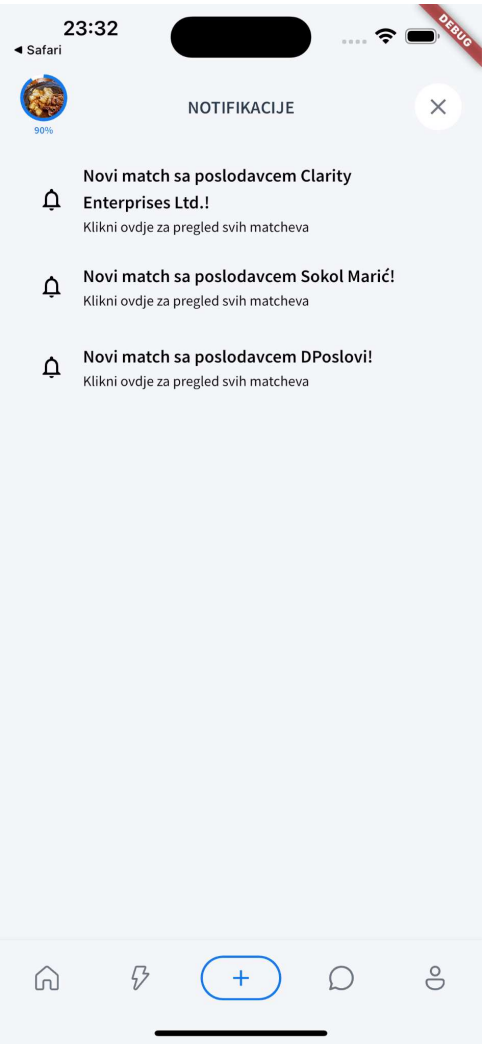
Slika 21. Pregled poslova gdje je poslodavac odobrio studenta



Slika 22. Pregled aktivnih poslova

Slika 23. Pregled poslova na koje se je student prijavio

Obavijesti



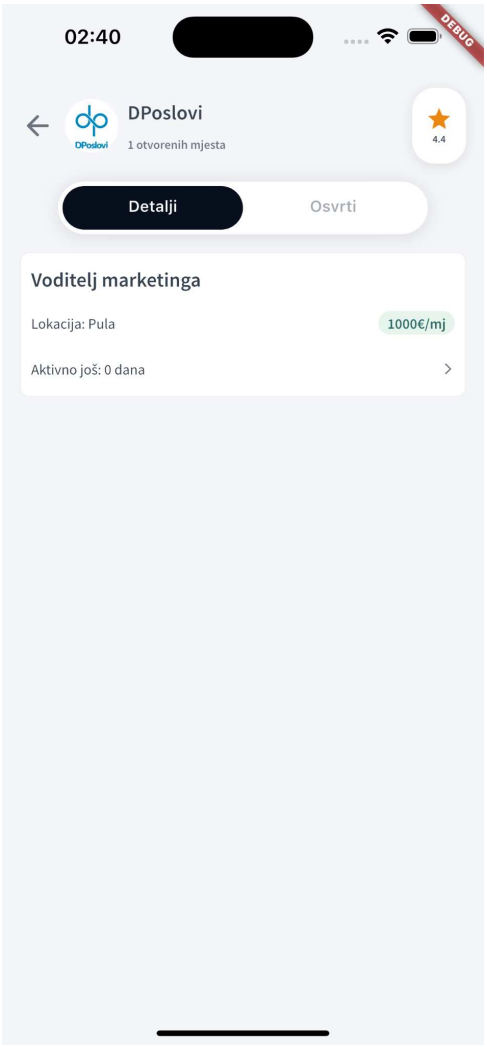
Slika 24. Prikaz notifikacija koje je student dobio

Student će dobiti obavijest svaki put kada poslodavac prihvati ili odbije njegovu prijavu. Te obavijesti prikazivat će se kao push notifikacija, za što se koristi paket **firebase_messaging** (Firebase Cloud Messaging)[5]. Osim trenutnih obavijesti, sve notifikacije bit će retroaktivno dostupne unutar aplikacije, gdje će student moći pregledati povijest svih obavijesti vezanih uz njegove prijave(Slika 24.).

[5] https://pub.dev/packages/firebase_messaging

Profil kompanije

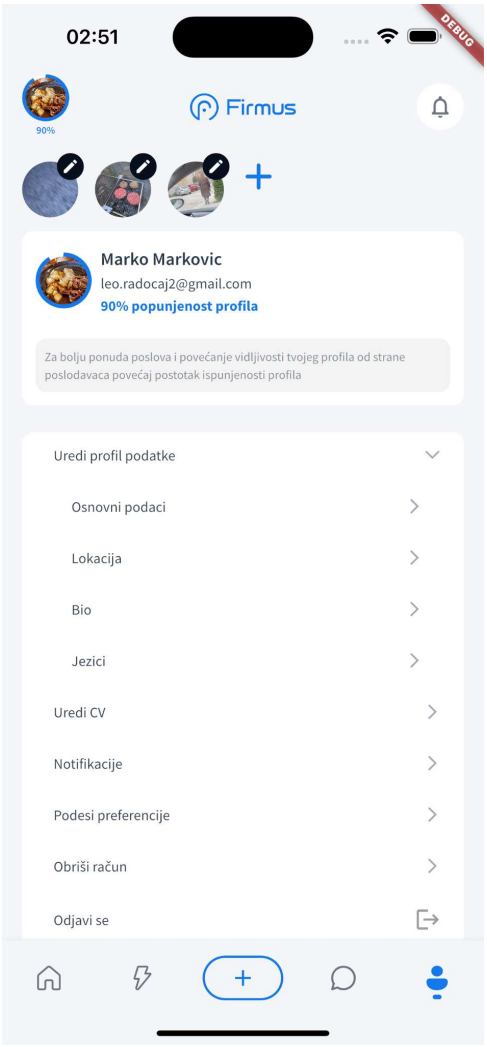
Ova stranica omogućava pregled svih oglasa(Slika 25.) koje je objavila određena kompanija, te uključuje recenzije poslova.



Slika 25. Prikaz profila kompanije

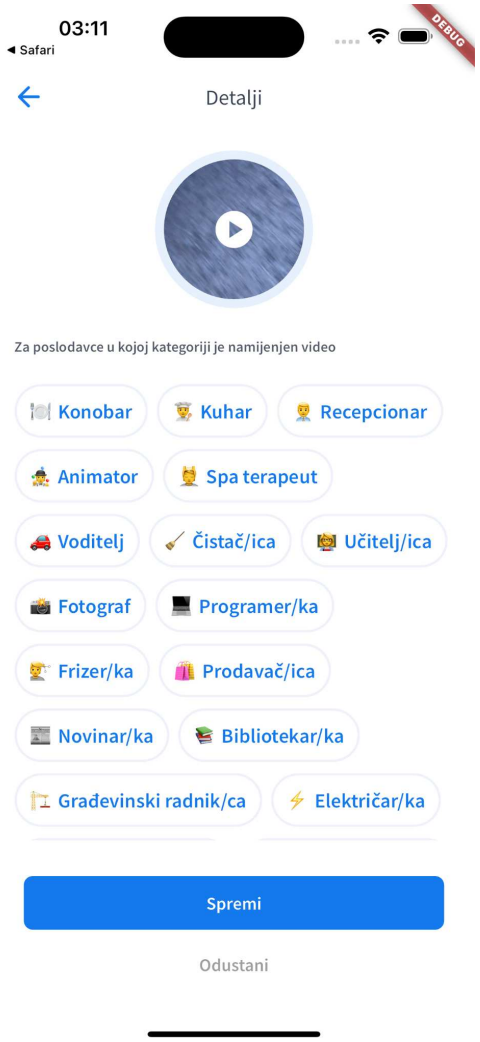
Profil studenta

Na ovoj stranici studenti mogu uređivati svoje osobne podatke, ažurirati svoj CV te promijeniti podatke o fakultetu i godini studija(Slika 26.).



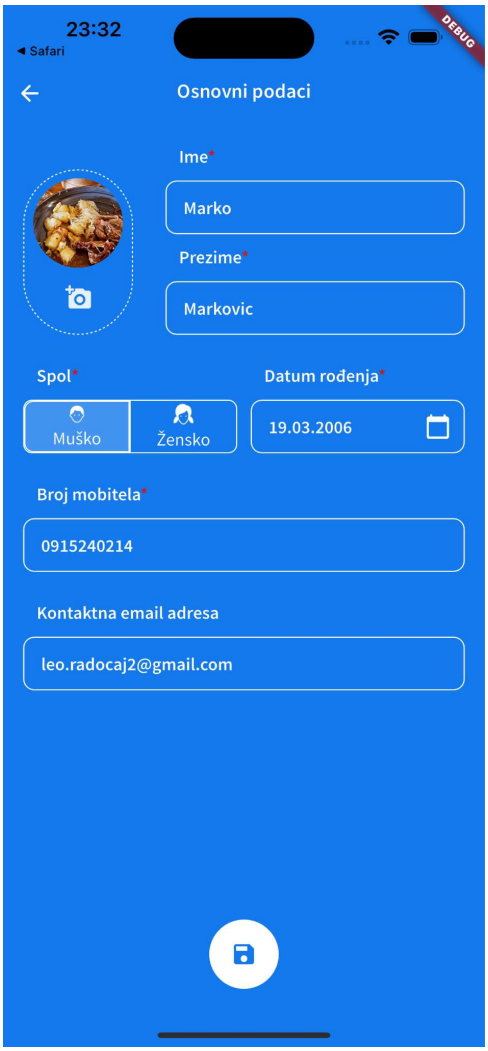
Slika 26. Profil studenta

Kod uređivanja video životopisa student može odabrati za koje poslove/pozicije je taj životopis namjenjen(Slika 27.).

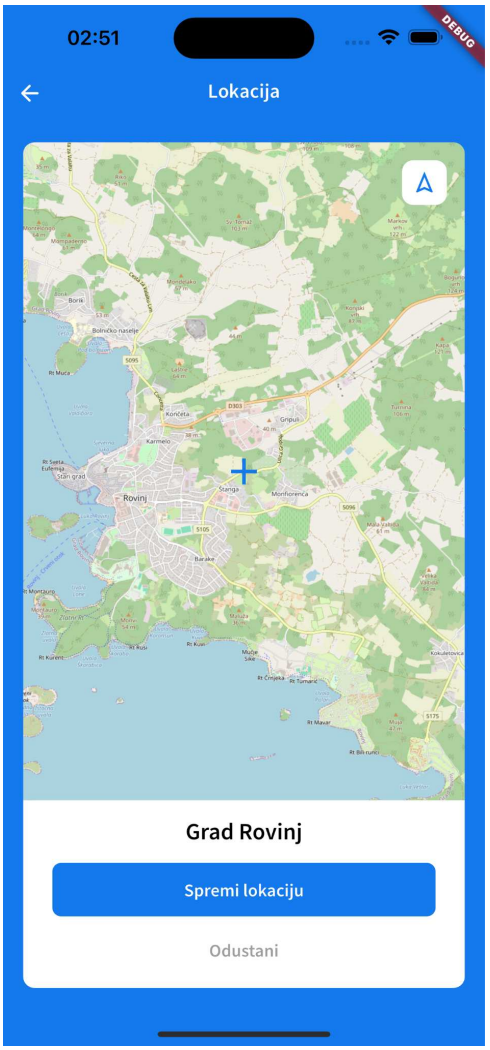


Slika 27. Odabir pozicija za koje je video životopis namjenjen

Student može promijeniti svoju profilnu sliku, email adresu na koju će ga se kontaktirati te broj mobitela (Slika 28.).



Slika 28. Uređivanje osobnih podataka



Slika 29. Odabir lokacije za koju se prikazuju poslovi

Za prikaz mape u aplikaciji(Slika 29.) koristimo OpenStreetMap u kombinaciji s paketom **flutter_map**[6].

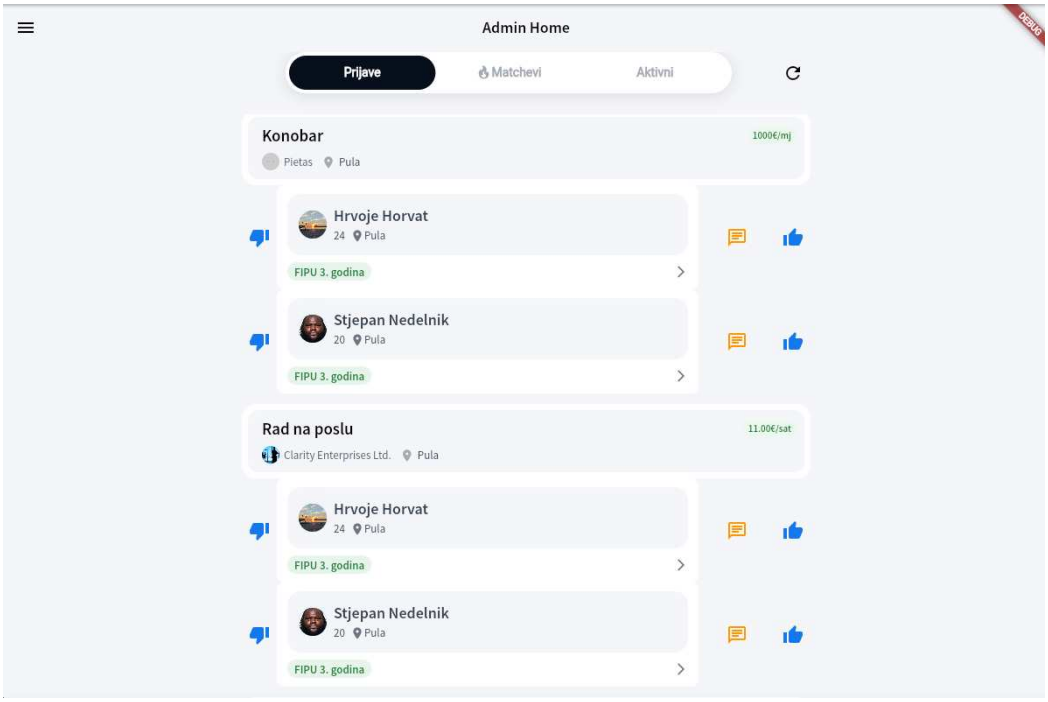
[6] https://pub.dev/packages/flutter_map

5.2. Admin sučelje

Admin sučelje je implementirano u Flutter-u unutar istog projekta kao i mobilna aplikacija, što omogućava ponovnu upotrebu postojećih modela, poslovne logike te komponenti. Time se izbjegava dupliciranje koda, čime se olakšava održavanje aplikacije i povećava efikasnost razvoja.

Pregled prijava

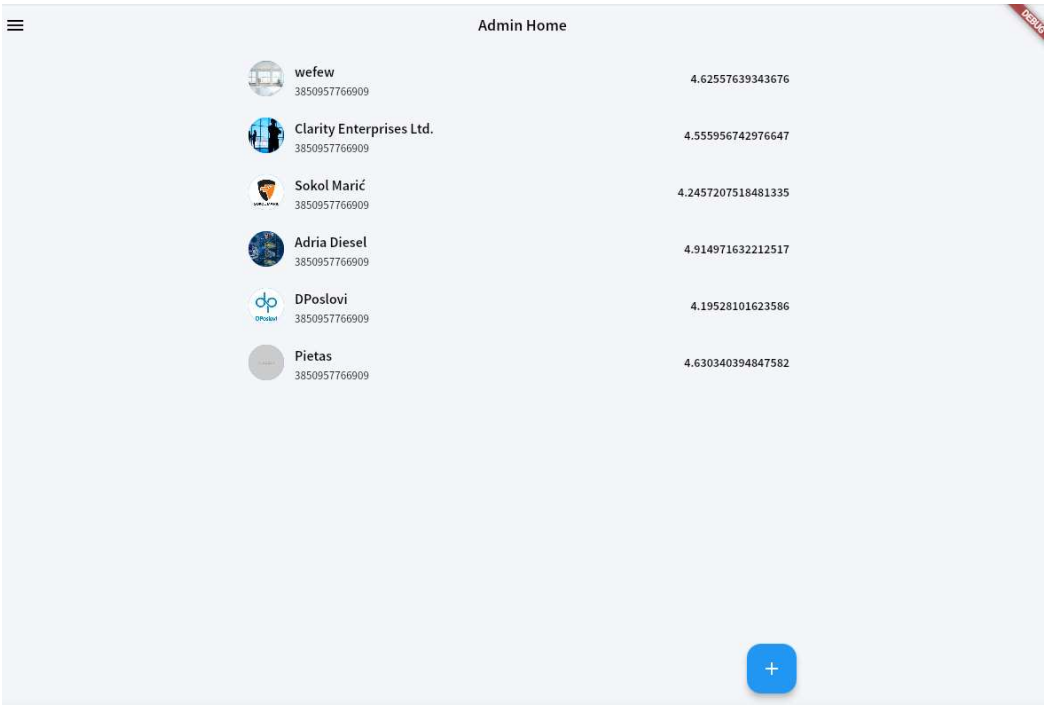
Admin ima pregledni panel u kojem može pratiti sve nove prijave za različite poslove, vidjeti popis studenata čije su prijave odobrene, te pratiti studente koji trenutno rade na određenom poslu.



Slika 30. Prikaz prijava na admin sučelju

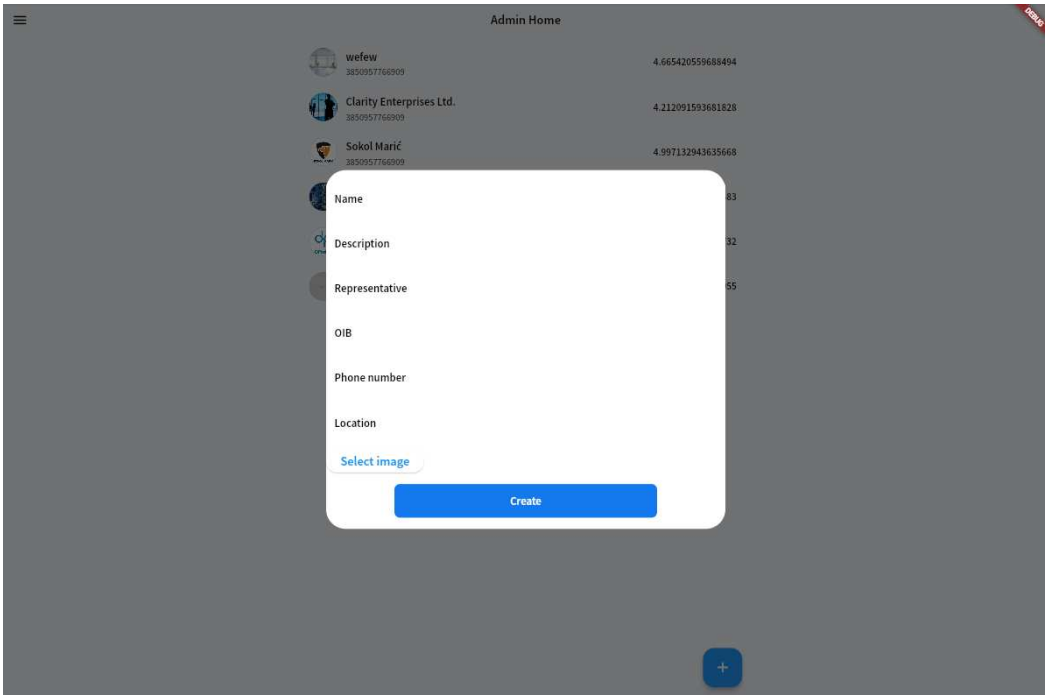
Pregled kompanija

Popis kompanija prikazuje njihove nazive, slike, kontakt brojeve i ocjene, omogućujući brz pregled osnovnih informacija o svakoj kompaniji.



Slika 31. Prikaz kompanija

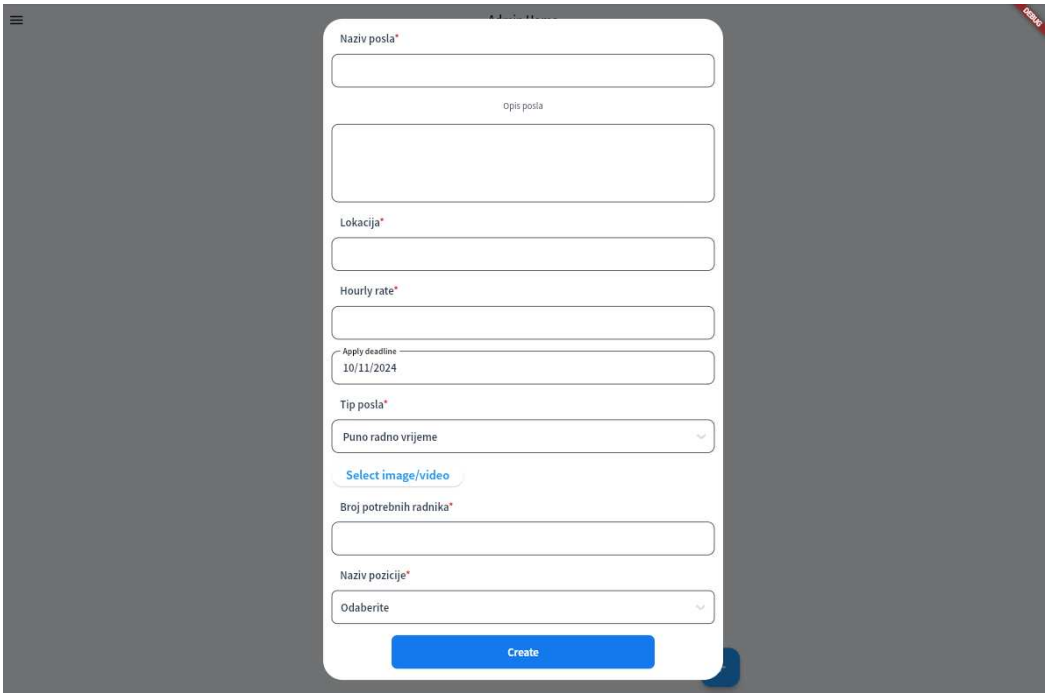
Izrada kompanije



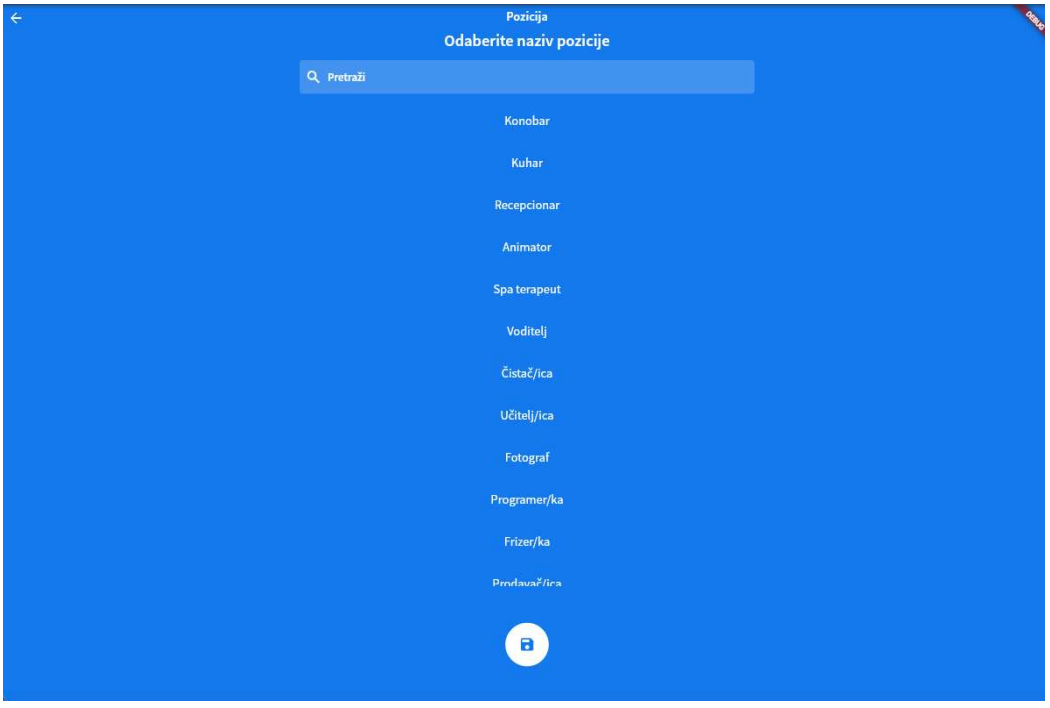
Slika 32. Forma za izradu kompanije

Izrada oglasa

Nakon što se odabere kompanija za koju se želi napraviti oglas, treba ispuniti formu gdje će upisati ključne informacije i prenijeti video.



Slika 33. Forma za izradu oglasa za posao



Slika 34. Odabir pozicije za poslovni oglas

6. Zaključak

Razvoj višeplatformske mobilne aplikacije za zapošljavanje studenata pokazao je kako korištenje Fluttera omogućava brzo razvijanje mobilnih aplikacija. Aplikacija inovativno prikazuje oglase u video formatu, čineći proces prijave zanimljivijim i interaktivnijim. Korištenje NestJS-a za backend omogućilo je izradu modularnog i lako održivog sustava, dok je TypeORM pojednostavio rad s bazom podatka PostgreSQL. Ova kombinacija čini sustav stabilnim i spremnim za budući razvoj i nadogradnju. Iako je primarno namijenjena studentima, potencijal za širenje funkcionalnosti je velik, uključujući podršku za širu populaciju i međunarodna tržišta.

Literatura

1. H. Majid , " Flutter Engineering", Hajian, 2024. Available: https://www.amazon.com/Flutter-Engineering-Majid-Hajian/dp/B0CSPN31J6. [Accessed: 08-10-2024].
2. S. Jonathan, "Dart Apprentice: Fundamentals", Sande, 2022. Available: https://www.kodeco.com/books/dart-apprentice-fundamentals. [Accessed: 08-10-2024].
3. G. Josh, "Learning TypeScript", Goldberg, 2022. Available: https://www.oreilly.com/library/view/learning-typescript/9781098110321/. [Accessed: 08-14-2024].
4. C. Daniel, L. Greg, " Practical Nest.js: Develop clean MVC web applications", Correa, 2022. Available: https://www.amazon.com/Practical-Nest-js-Develop-clean-applications/dp/B09RMBJDB3. [Accessed: 08-18-2024].
5. F. Dimitri, "The Art of PostgreSQL", Fontaine, 2019. Available: https://www.goodreads.com/book/show/52755483-the-art-of-postgresql. [Accessed: 08-18-2024].
6. H. David, "Quick Start to using Typescript and TypeORM on Node.js for CLI and web applications", May 23, 2019, . Available: https://www.amazon.com/Quick-Typescript-TypeORM-Node-js-applications-ebook/dp/B07S87X4ZK. [Accessed: 08-18-2024].
7. R. Remi , "Riverpod", Rousselet, 2024. Available: https://riverpod.dev/docs/introduction/why_riverpod. [Accessed: 08-18-2024].
8. D. Vahid, "Understanding the Model-View-Controller (MVC) Pattern", Dejwakh, 2024. Available: https://vahid.blog/post/2021-04-16-understanding-the-model-view-controller-mvc-pattern/. [Accessed: 08-2024].
9. R. Richard, "Flutter and Dart Cookbook: Developing Full-Stack Applications for the Cloud", Rose, 2023. Available: https://www.amazon.com/Flutter-Dart-Cookbook-Developing-Applications/dp/1098119517. [Accessed: 08-18-2024].

10. D. Paulo , "Flutter & Dart - Complete App Development Course", Dichone, 2024. Available: https://www.oreilly.com/library/view/flutter-dart/9781836203735/. [Accessed: 08-18-2024].

Popis slika

Programski kod

Flutter: https://github.com/SempaiEcchi/zavrsni-app

Backend: https://github.com/SempaiEcchi/zavrsni-backend