SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 170

WEB-APLIKACIJA ZA RJEŠAVANJE PROGRAMSKIH ZADATAKA S ELEMENTIMA DRUŠTVENE MREŽE

Nikola Kešćec

Zagreb, lipanj 2021.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 170

WEB-APLIKACIJA ZA RJEŠAVANJE PROGRAMSKIH ZADATAKA S ELEMENTIMA DRUŠTVENE MREŽE

Nikola Kešćec

Zagreb, lipanj 2021.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 12. ožujka 2021.

ZAVRŠNI ZADATAK br. 170

Pristupnik:	**Nikola Kešćec (0036515301)**
Studij:	Elektrotehnika i informacijska tehnologija i Računarstvo
Modul:	Računarstvo
Mentor:	izv. prof. dr. sc. Igor Mekterović

Zadatak: **Web-aplikacija za rješavanje programskih zadataka s elementima društvene mreže**

Opis zadatka:

Napraviti pregled web-aplikacija koje omogućuju rješavanje programskih zadataka (npr. TopCoder, CoderByte, CodeWars, itd.) u rekreativne, obrazovne ili natjecateljske svrhe i uočiti tipične elemente i obrasce. Napraviti web-aplikaciju s elementima društvene mreže koja će omogućiti zadavanje i rješavanje programskih zadataka. Aplikacija mora omogućiti stvaranje korisničkih računa, nakon čega korisnici mogu definirati vlastite programske zadatke gdje uz teksta zadatka moraju definirati i ulazno-izlazne parove znakovnih nizova koji će poslužiti za testiranje ispravnosti rješenja (dinamička analiza kôda). Zadatke rješavaju drugi korisnici, pri čemu je potrebno ostvariti nekakav sustav rangiranja rješenja, npr. na temelju brzine rješavanja ili brzine obavljanja ili na temelju glasova korisnika itd. Osim toga, ostvariti i neke elemente društvenih mreža: omogućiti povezivanje korisnika odnosno omogućiti korisnicima da prate druge korisnike, glasaju, komentiraju te potencijalno ugraditi sustav reputacije. Donijeti ocjenu ostvarenog pristupa te smjernice za budući razvoj.

Rok za predaju rada: 11. lipnja 2021.

Srdačno se zahvaljujem svojem mentoru, prof. dr. sc. Igoru Mekterovi ću na cijenjenoj pomoći, savjetovanju i stručnom usmjeravanju. Također se zahvaljujem i Hermanu-Zvonimiru Došilovi ću na pomoći i uputama oko Judge0a bez kojeg ovaj završni rad ne bi bio moguć.

SADRŽAJ

1. Uvod

U zadnjem desetljeću sveprisutnost računalnih sustava uveliko je doprinijela većem porastu zanimanja populacije prema svemu što je povezano s računalima. Naravno, to podrazumijeva i **programiranje**. Programiranje je proces osmišljanja skupa instrukcija koje govore računalu kako izvoditi određeni zadatak.

Iako su već početkom tisućljeća računalni sustavi bili poprilično sveprisutni, njihova je zastupljenost u zadnjih dva desetljeća još porasla. Porastom njihove zastupljenosti porasla je potražnja za računalnom sklopovskom podrškom (engl. *hardware*), a s njom i potreba za kvalitetnom programskom podrškom (engl. *software*). Kvalitetna programska podrška rezultat je godina učenja, razmatranja i pisanja programskog koda, a kao i svaku drugu vještinu potrebno ju je usavršavati. Usavršavanje programiranja je moguće na mnogo načina, a jedan od popularnijih načina jest rješavanje programskih zadataka. Naravno, sva rješenja programskih zadataka nisu jednako kvalitetna stoga se nerijetko rješenja rangiraju po kvaliteti sa svrhom isticanja znanja autora kvalitetnog rješenja. Događanja na kojima se upravo takav pristup prakticira i cijeni su programerska natjecanja (engl. *competitive programming competitions*).

Makar su takva događanja izrazito bogat izvor programerskog znanja, neiskusne populacije zainteresirane programiranjem teško da će svojevoljno ući na opisana događanja. Razlog tome je razlika u količini znanja između njih i regularnih sudionika jer dobar dio tih sudionika već je vrlo vješt. Stoga je postojanje internetskih stranica koje pružaju uvid u programiranje i rješavanje programskih zadataka dobrodošla novost. Ipak, često riječ vrlo iskusnog programera neiskusnom programeru može biti neprocjenjiva. Iz tog razloga zanimljiv je spoj društvenih mreža (engl. *social networks*) i stranica koje pružaju mogućnost rješavanja programskih zadataka. Osobi koju pisanje programskog koda interesira stranica pruža nekompetitivnu mogućnost rješavanja zadataka, a iskusnom programeru mogućnost da kroz rješavanje i objašnjavanje svojih rješenja produbi i utemelji svoja znanja. Upravo je **Codeflow**, internetska aplikacija opisanih svojstava, tema ovog završnog rada.

2. Korisnički zahtjevi

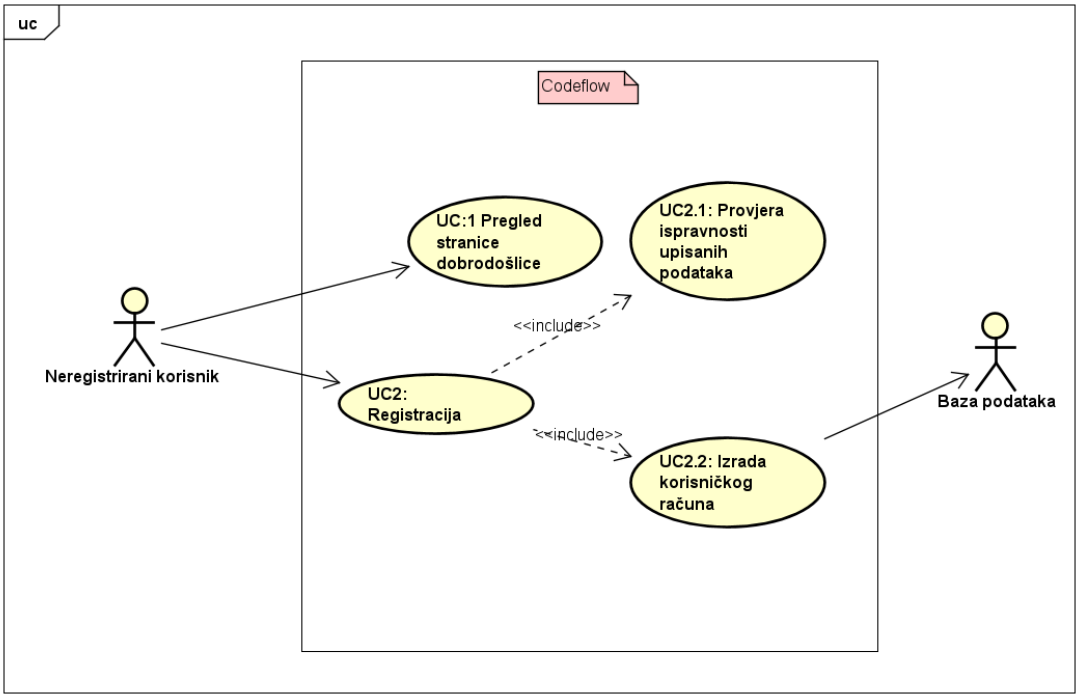
Društvene mreže i stranice koje podržavaju zadavanje i rješavanje programskih zadataka imaju različite **funkcionalne** i **nefunkcionalne** zahtjeve. Internetska aplikacija koja je tema ovog završnog rada ispunjava dio funkcionalnih i nefunkcionalnih zahtjeva stranica navedenih tipova.

2.1. Funkcionalni zahtjevi

Funkcionalni zahtjevi podrazumijevaju usluge koje sustav mora pružati te kako će reagirati na različite ulazne poticaje. Drugim riječima, funkcionalni zahtjevi definiraju što internetska stranica može učiniti za određenog korisnika i što određeni korisnici mogu učiniti na njoj. Većina modernih društvenih mreža traže od svojih korisnika da prvo naprave korisnički račun prije nego što im dopuste pristup svojim glavnim funkcionalnostima. Iz tog razloga na korisnika treba gledati iz dvije različite perspektive: kao na **neregistriranog korisnika** i **registriranog korisnika**.

2.1.1. Zahtjevi neregistriranog korisnika

Neregistriran korisnik, prema uzoru na mnoge moderne društvene mreže, ima mogućnost pristupanja stranici dobrodošlice te pravo registriranja. Neregistrirani korisnik tijekom registracije upisuje određene podatke s kojima će se, ako su ispravni, registrirati. Grafički prikaz zahtjeva prikazan je na slici 2.1.



Slika 2.1: Zahtjevi neregistriranog korisnika

2.1.2. Zahtjevi registriranog korisnika

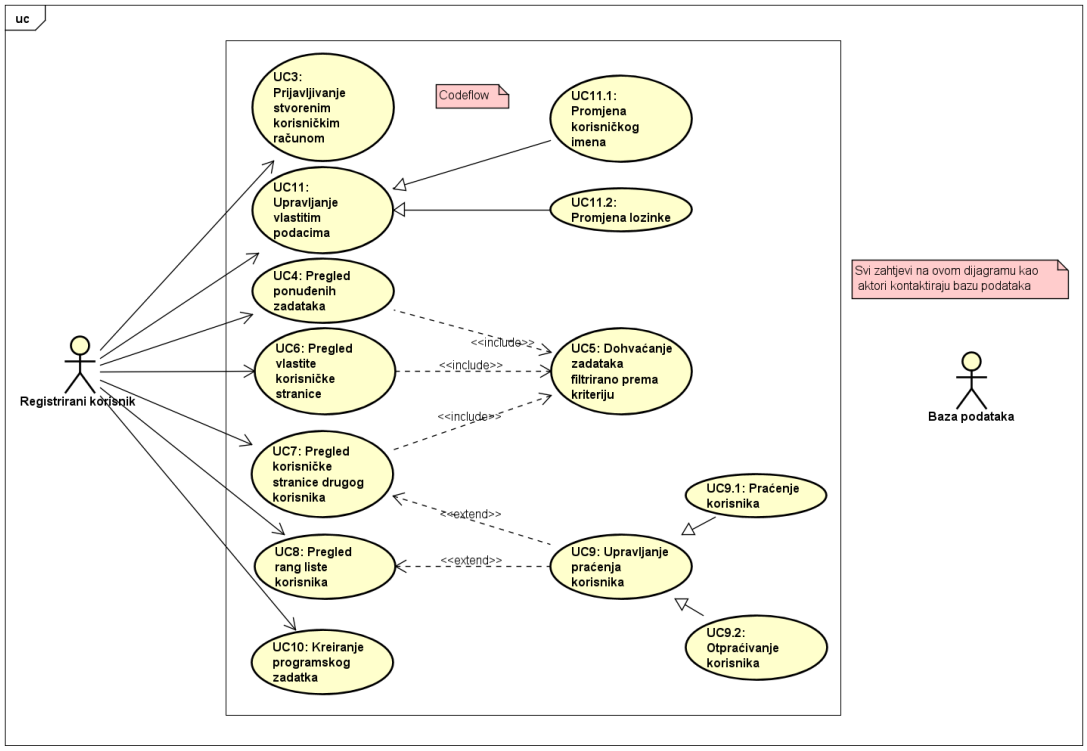
Nakon što se korisnik registrira postaju mu dostupne dodatne funkcionalnosti. Registriranom korisniku dostupna je mogućnost prijave s podacima koje je koristio za registraciju. Pošto se prijavi korisnik ima mogućnost pregledavanja ponuđenih zadataka i rang liste. Ponuđene zadatke može filtrirati na neki od dostupnih načina i dostupan mu je detaljniji pregled jednog od filtriranih zadataka. Prema podržanim funkcionalnostima modernih društvenih mreža, korisniku je predstavljena opcija filtriranja zadataka prema sljedećim kriterijima:

- preporučeni zadaci
- zadaci slijeđenih korisnika
- najnoviji zadaci

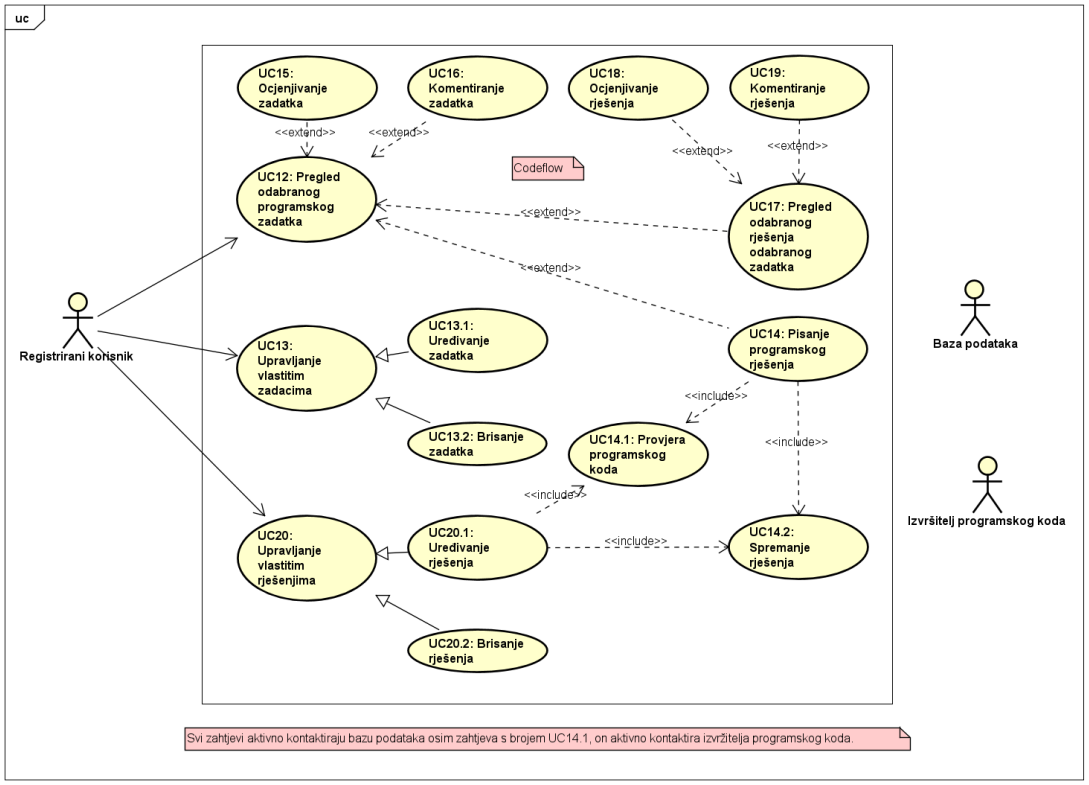
Tijekom detaljnijeg pregleda odabranog zadatka prijavljenom su korisniku ponuđene opcije ocjenjivanja i komentiranja zadatka. Uz sam detaljniji opis zadatka, korisniku je moguć odabir nekog od rješenja gledanog zadatka. Odabrano rješenje registrirani korisnik može komentirati, a mogućnost ocjenjivanja zadatka omogućena mu je samo nakon rješavanja zadatka. Budući da je rješavanje zadatka preduvjet jednom od ostalih zahtjeva, ta je funkcionalnost također omogućena korisniku. Korisničko rješenje eva-

luirano je pomoću vanjskog izvršitelja programskog koda te nakon evaluacije spremljeno. Uz stvaranje rješenja programskih zadataka, prijavljenom korisniku ponuđena je opcija stvaranja programskog zadatka. Zahtjevi navedeni uz pregledavanja zadataka, rješavanje zadataka i stvaranje zadataka jezgreni su zahtjevi aplikacije.

Prijavljenom korisniku dostupna je mogućnost potpunog upravljanja vlastitim zadacima, rješenjima, komentarima i ocjenama. Pod tim se podrazumijevaju njihovo uređivanje i brisanje. Uz navedenu mogućnost upravljanja vlastitim zadacima, rješenjima, komentarima i ocjenama, prijavljenom korisniku omogućeno je mijenjanje vlastitih podataka. Po uzoru na određene mogućnosti društvenih mreža, prijavljenom korisniku su dostupne opcije pregleda profila drugih registriranih korisnika. Isto tako, korisnik može druge korisnike pratiti ili otpratiti. Osim mogućnosti pregleda tuđih korisničkih stranica, prijavljeni korisnik može vidjeti svoju korisničku stranicu. Detaljniji grafički prikazi ovih korisničkih zahtjeva vidljivi su na slikama 2.2 i 2.3.



Slika 2.2: Zahtjevi registriranog korisnika, prvi dio.



Slika 2.3: Zahtjevi registriranog korisnika, drugi dio.

2.2. Nefunkcionalni zahtjevi

Nefunkcionalni zahtjevi, za razliku od funkcionalnih, ne opisuju koje mogućnosti sustav treba pružati nego koje karakteristike on treba imati. U slučaju promatrane internetske stranice, nefunkcionalni zahtjevi navedeni su unutar sljedeće liste:

- korisničke lozinke trebaju biti pravilno spremljene
- veza s bazom podataka i izvršiteljem programskog koda treba biti stabilna
- korisničko sučelje treba biti intuitivno za korištenje
- ostvarivanje evaluacije korisničkih rješenja treba biti unutar dvominutnog intervala

3. Postojeća programska rješenja i funkcionalnosti društvenih mreža

Poznata programska rješenja koja podržavaju funkcionalnost rješavanja programskih zadataka su:

- Leetcode[13]
- Edgar[8]
- Codewars[12]
- HackerRank[9]

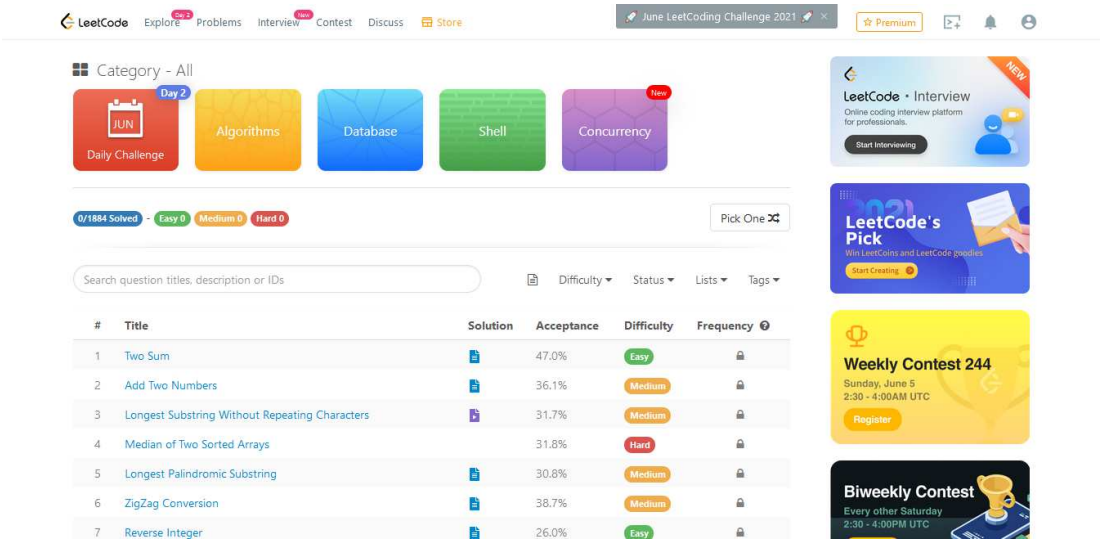
3.1. Leetcode

Leetcode je vjerojatno najveća i najpoznatija internetska aplikacija koja omogućava rješavanje programskih zadataka. Osnovana 2015. godine u Silicijskoj dolini, od samog početka bilježila je značajan rast i već 2017. godine imala preko milijun registriranih korisnika.

Svrha i namjera Leetcode aplikacije jest priprema korisnika aplikacije na razgovore za posao pomoću specifično dizajniranih programskih zadataka. Pripremljeni zadaci slične su prirode kao i oni koje najpoznatije tehnološke firme predstavljaju svojim pristupnicima na razgovorima za posao. Preduvjet pristupu ponuđenim zadacima je registracija, a registrirani korisnici uz mogućnost rješavanja zadataka (koje mogu filtrirati po kriterijima) također mogu provoditi rasprave unutar foruma. Rasprave su često povezane s temom pitanja koje velike tehnološke tvrtke postavljaju. Bitno je napomenuti da Leetcode podržava programska rješenja pisana unutar četrnaest različitih programskih jezika. Rješenja zadataka mogu biti predložena unutar rasprave te dodatno komentirana od drugih korisnika.

Ipak, postoje i neke funkcionalnosti koje nisu podržane unutar Leetcodea. Leetcode ne dopušta svojim registriranim korisnicima da sami dizajniraju svoje zadatke ili da prate

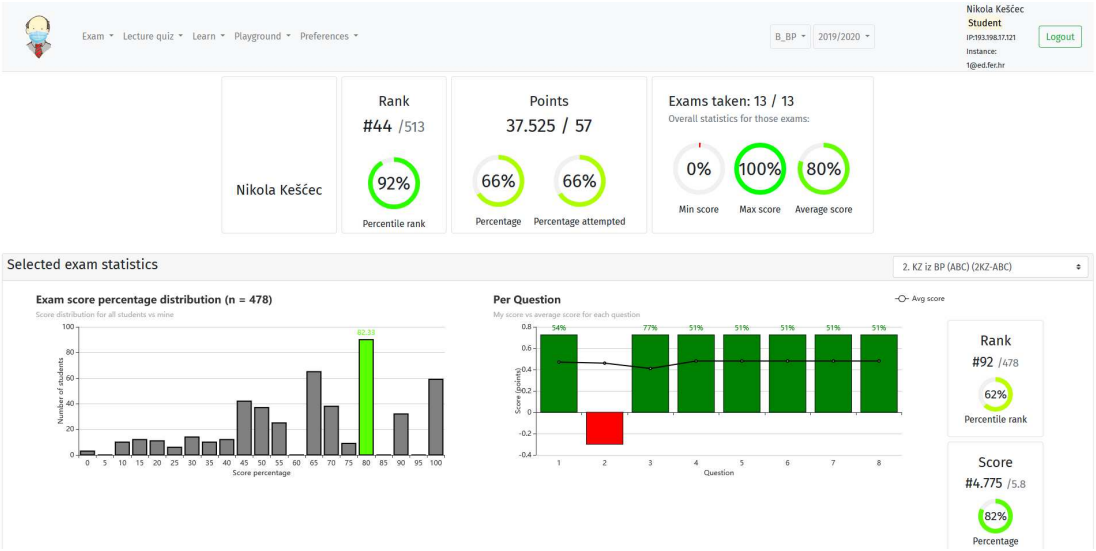
korisnike koji su im se dojmili unutar diskusijskog foruma.
Ukratko, Leetcode internetska aplikacija je veoma impresivna po broju funkcionalnosti koje nudi svojim registriranim korisnicima. Navedene funkcionalnosti napisane unutar ove sekcije najosnovnije su mogućnosti koje Leetcode nudi.



Slika 3.1: Prikaz stranice Leetcode

3.2. Edgar

Edgar je internetska aplikacija koju Fakultet elektrotehnike i računarstva koristi u edukacijske svrhe, primarno kao sustav provođenja provjera vezanih uz temu programiranja. Na Fakultetu elektrotehnike i računarstva se intenzivno koristi zbog svojih mogućnosti definiranja i provođenja testova.
Pristup internetskoj aplikaciji moguć je samo ovlaštenim osobama pomoću njihovih korisničkih računa. Korisnici koji imaju ulogu predavača mogu definirati testove koje onda ostali studenti rješavaju nakon što se prijave sa svojim korisničkim računima. Edgar podržava mnoge programske jezike te tijekom rješavanja testnih zadataka vraća povratne informacije korisniku. Također, Edgar podržava stvaranja edukacijskih materijala poput edukacijskih vježbi. Isto tako, korisniku je na raspolaganju i pisanje proizvoljnog koda unutar "Code sandbox" funkcionalnosti stranice. Korisnici Edgara ne mogu ući u interakciju jedni s drugima.
Internetska stranica Edgar primarno pruža funkcionalnost provjere programskog koda. Ne fokusira se na interakciju korisnika te sadržava poveći broj funkcionalnosti od kojih su ovdje navedene samo osnovne.



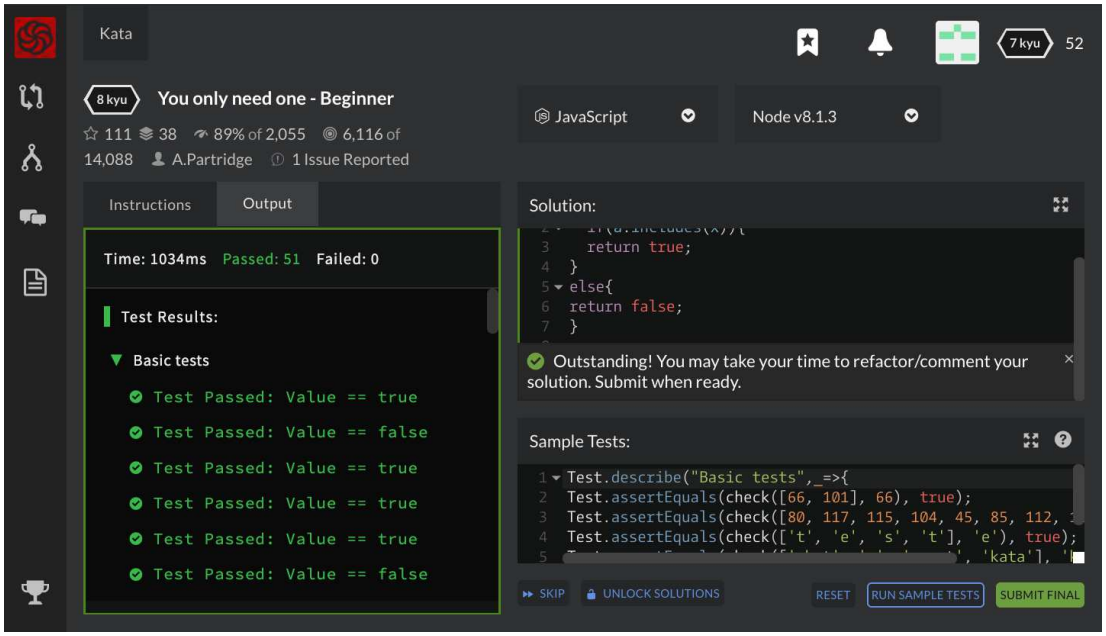
Slika 3.2: Prikaz stranice Edgar

3.3. Codewars

Codewars internetska stranica po funkcionalnosti slična je Leetcodeu, ali uvodi dodatne mogućnosti rangiranja korisnika prema težini zadataka (unutar stranice zvani *kata*) koje rješavaju. Osnovana je 2012. godine, a njezina namjera usavršavanje je znanja programskih jezika i proširenje znanja drugih jezika.

Kate su zadaci koji nose određenu količinu bodova, a definiraju ih korisnici Codewarsa. Svaka kata nosi određen broj bodova, zvani *kyu*. Teži zadaci nose više bodova, a rješenja zadataka moguće je komentirati s ostalim korisnicima. Također, isto kao i sve već spomenute internetske stranice, Codewars pruža mogućnost rješavanja zadataka u različitim jezicima. Bitno je napomenuti da Codewars implementira sve popularniji proces "igrifikacije" (engl. *gamification*). Taj proces uvodi razine i određene nagrade koje korisnike motiviraju za daljnje rješavanje sve težih zadataka.

Tijekom registracije, korisnici trebaju proći inicijalizacijski test koji testira njihova osnovna znanja jednog od ponuđenih jezika i to je minimalno predznanje glavan je preduvjet budućeg korištenja stranice.



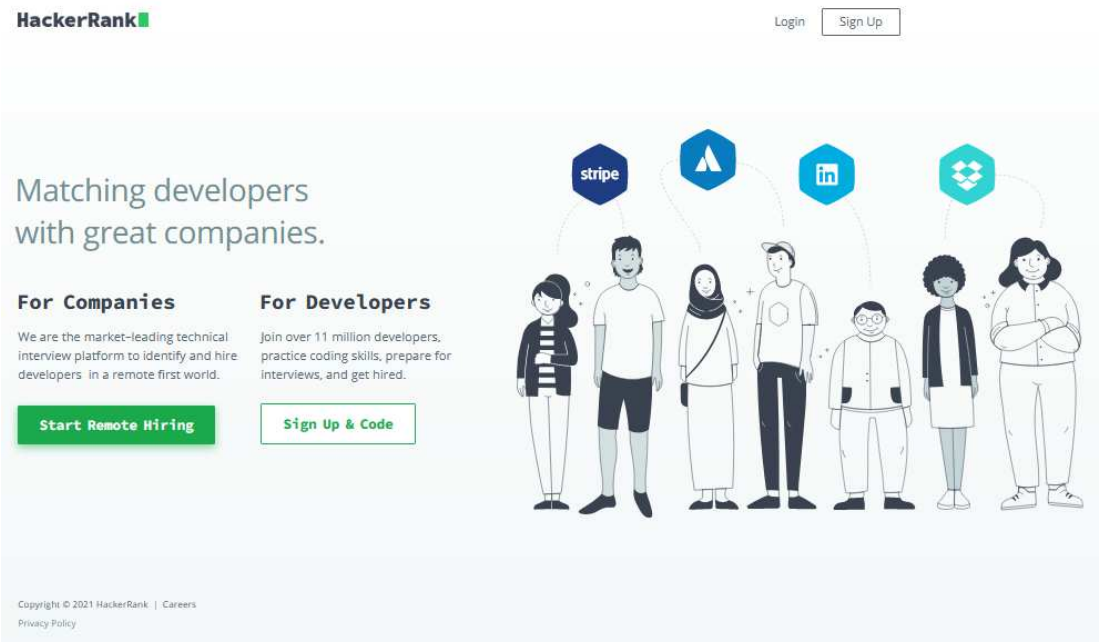
Slika 3.3: Prikaz rješavanja jedne *kate*

3.4. HackerRank

Internetska stranica HackerRank pruža drugačije usluge od već navedenih stranica. Namjera stranice je povezivanje programera s tvrtkama koje bi bile zainteresirane za njih, ali i učenje novih jezika, programskih algoritama te paradigmi.

Stranica pruža mogućnost stvaranja računa korisnicima, ali i tvrtkama koje interesira HackerRank. Mogućnost rješavanja programskih zadataka omogućena je za korisnike i tvrtke, a nerijetko stranica organizira i natjecanja zvana "CodeSprints". Na tim natjecanjima korisnici rješavaju iste programske zadatke. HackerRank stranica predana rješenja rangira prema njihovoj točnosti i kvaliteti. Autore tih rješenja također rangira na HackerRank ljestvici korisnika, te, kao i Codewars, provodi proces "igrifikacije".

Mogućnosti HackerRank stranice raspodijeljene su između različitog tipa korisnika: regularnog korisnika i tvrtke. Korisnicima pruža mogućnosti rješavanja zadataka u svrhu promocije te onim korisnicima koji traže posao potencijalno može pomoći u tomu.



Slika 3.4: Prikaz stranice dobrodošlice HackerRank internetske stranice

3.5. Zajedničke funkcionalnosti i dodatak funkcionalnosti društvenih mreža

Vidljivo je iz bližeg razmatranja navedenih stranica da svaka od stranica ima različitu namjeru te u različite svrhe koristi funkcionalnost zadavanja i rješavanja zadataka. Stranice Codewars, Leetcode i HackerRank donekle su slične po funkcionalnostima koje pružaju, ali ipak razlikuju se podosta u namjeri. Edgar je različit od navedenih stranica jer njegova je svrha prvenstveno akademske i edukacijske prirode. Ipak, navedene stranice imaju nekoliko dijeljenih funkcionalnosti:

- stvaranje zadataka (Edgar i Codewars)
- rješavanje zadataka
- provjera rješenja zadataka
- komentiranje rješenja (Codewars, Leetcode, HackerRank)

Bitno je također uvidjeti da dio stranica pruža određene mogućnosti društvenih mreža, kao što su već spomenuto komentiranje rješenja te mogućnost određene interakcije s korisnicima. Ipak, mogućnosti praćenja korisnika kod velikog dijela stranica nisu moguće, a isto tako direktno ocjenjivanje rješenja nije podržano na svim stranicama. Nerijetko su i određene funkcionalnosti stranica moguće samo korisnicima koji plaćaju

određenu pretplatu (poput pregleda službenog rješenja nekog programskog zadatka). Ono što Codeflow stranica sadržava unutar sebe jest spoj navedenih najčešćih funkcionalnosti promotrenih stranica i jezgrenih funkcionalnosti modernih društvenih mreža. Te jezgrene funkcionalnosti su:

- praćenje korisnika
- komentiranje na objavljene sadržaje drugih korisnika (uključujući vlastite)
- ocjenjivanje sadržaja drugih korisnika
- pregledavanje profila drugih korisnika

4. Korištene tehnologije

Arhitektura modernih internetskih aplikacija najčešće se sastoji od tri sloja:

- korisničkog ili prezentacijskog sloja
- poslovnog sloja
- podatkovnog sloja

Tijekom razvoja internetske aplikacije Codeflow korištene su tehnologije koje se uglavnom mogu smjestiti unutar jednog od ta tri sloja te su one po slojevima dodatno pojedinačno razmotrene.

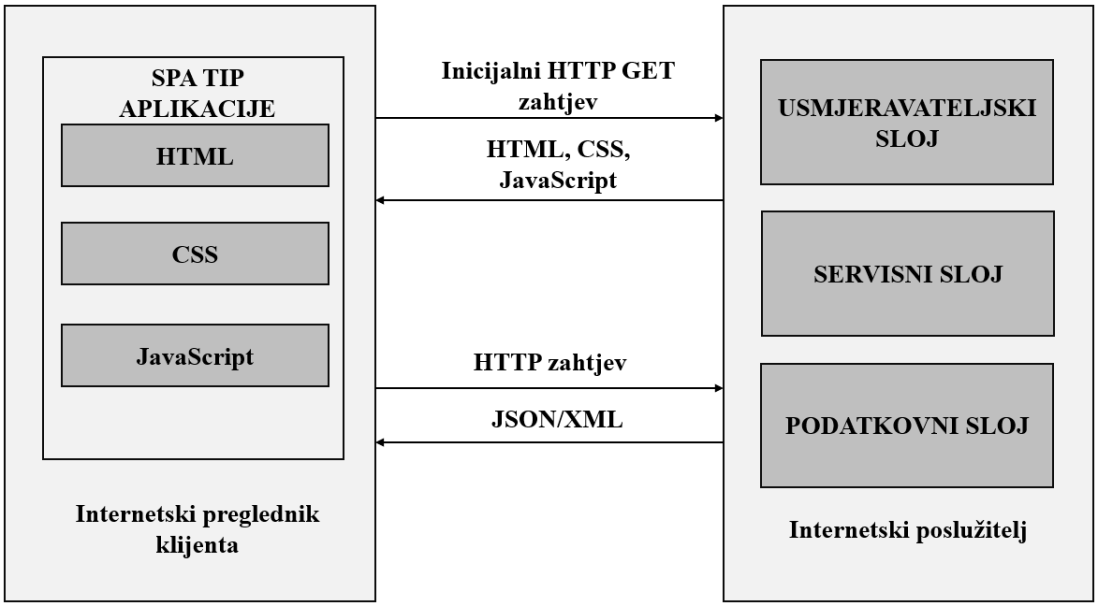
4.1. Tehnologije prezentacijskog sloja

Korištene tehnologije prezentacijskog sloja pisane su JavaScript[6] programskim jezikom jer on je jedini programski jezik kojeg internetski preglednici podržavaju. Ipak, razvoj JavaScript aplikacija ne događa se isključivo unutar internetskih preglednika, a podršku za razvoj i izvođenje izvan spomenutih preglednika omogućava Node.js[19] platforma. Node.js platforma je cjelokupno pokretačko okruženje koje pomoću komponente *V8 JavaScript engine*[27] izvodi JavaScript programski kod izvan internetskog preglednika. Node.js također unutar sebe sadržava upravitelja paketima, NPM[18] (Node Package Manager), koji pruža mogućnost korištenja velikog broja različitih dodataka i biblioteka (engl. *libraries*) pisanih jezikom JavaScript.

Single Page Application tip stranica

SPA[17] (engl. *Single page Application*) tip internetskih aplikacija pri promjeni stranice ne dohvaćaju HTML datoteku sa sadržajem nove stranice već dinamički prepisuju sadržaj trenutačne stranice uz pomoć JavaScript programskog jezika te podataka dobivenih s internetskog poslužitelja. Cilj *Single Page Application* tipa internetske aplikacije jest doživljaj korištenja aplikacije učiniti sličnim onomu kojeg imaju aplikacije pokrenute na računalu.

Budući da se stranice mijenjaju isključivo manipulacijom elemenata DOM-a (engl. *Document Object Model*), tijekom korištenja aplikacije učitavanje stranice potrebno je obaviti samo jednom tijekom prvog pristupa stranici. Glavni nedostaci SPA internetskih aplikacija jesu sporo prvo učitavanje internetske aplikacije i loša optimizacija zastupljenosti tijekom pretrage internetskim pretraživačima. Sporo učitavanje aplikacije uzrokovano je potrebom za učitavanjem cjelokupne internetske aplikacije tijekom prvog pristupa stranici, a loša optimizacija zastupljenosti tijekom pretrage inicijalnim nedostatkom HTML-a prilikom početnog učitavanja. Ipak, brzina odaziva na korisničke zahtjeve uveliko unaprjeđuje korisničko iskustvo, pa je upravo korisničko sučelje Codeflow aplikacije SPA tip internetske aplikacije. Kratak pregled interakcije i arhitekture SPA tipa internetske aplikacije i internetskog poslužitelja prikazan je na slici 4.1.



Slika 4.1: Prikaz općenite interakcija i arhitekture SPA tipa internetskih aplikacija

4.1.1. React i React Router

React[7] je biblioteka kojoj je svrha izrada korisničkih sučelja (engl. *user interface*). Otvorenog je koda te ga razvijaju i održavaju različite tvrtke (od kojih je najbitnija Facebook jer ona ga je zapravo i začela) i različiti individualni programeri. Prva verzija Reacta na tržište je izašla 2013. godine, a od tada njegova popularnost kontinuirano raste. Reactov fokus jest razvoj dinamičnih i lagano nadogradivih korisničkih sučelja. React laganu nadogradivost ostvaruje putem komponentno usmjerenog razvoja, što znači da svaka razvijana funkcionalnost enkapsulira unutar sebe svu logiku i izgled

koji su potrebni za njezin rad. Sve komponente mogu unutar sebe sadržavati druge komponente, tako ostvarujući kompoziciju (engl. *composition*) komponenata, a njenim korištenjem razvijanje složenih korisničkih sučelja postaje jasno i strukturirano. React biblioteka spomenute komponente koristi za izgradnju konkretnog dokumenta objektnog modela koji kasnije biva prikazivan korisniku. Kako korisnička interakcija ne bi bila spora, React interno održava posebnu strukturu podataka, virtualni dokument objekt model, koja pamti stanje komponenata. Ako se ikoja komponenta promijeni, React samo promijenjenu komponentu ažurira unutar stvarnog modela objekta dokumenta. Tim pristupom održava responzivnost korisničkog sučelja.

```
import { Container } from "react-bootstrap";
import Footer from "./Footer";
import TopBar from "./TopBar";

const Layout = ({ children, path }) => {
  return (
    <Container
      fluid
      className="d-flex justify-content-between column pattern p-0 full-height"
    >
      <TopBar path={path} />
      <Container
        fluid
        className="flex-grow-1 d-flex column justify-content-between"
      >
        {children}
      </Container>
      <Footer />
    </Container>
  );
};

export default Layout;
```

Slika 4.2: Primjer jednostavnije kompozicije komponenata unutar Reacta

React Router[21] dodatak je React biblioteci koja služi za usmjeravanje između različitih dostupnih stranica razvijenih pomoću React komponenata. Pomoću njegovih funkcionalnosti ostvaruju se značajke SPA tipa aplikacije jer React Router preusmjeravanje stranica obavlja isključivo unutar internetskog preglednika klijenta. Deklariranjem specifičnih URL-a (engl. *Uniform Resource Locator*) te korištenjem specijalnih *Link* React komponenata klijentu je omogućeno mijenjanje stranica.

```
import React from "react";
import { BrowserRouter as Router, Switch } from "react-router-dom";
import Renderer from "./renderer";
import routes from "./routes";

const CodeFlowRouter = () => {
  return (
    <Router>
      <Switch>
        {routes.map((route, index) => (
          <Renderer {...route} key={index} />
        ))}
      </Switch>
    </Router>
  );
};
```

Slika 4.3: Primjer korištenja React Routera.

4.1.2. Bootstrap

Bootstrap[4] je radni okvir (engl. *framework*) otvorenog koda namijenjen dizajnu i prilagođavanju responzivnih internetskih stranica s mnoštvom definiranih komponenata, izgleda i predložaka. Temeljen je na CSS stilskom jeziku te, uz pomoć JavaScript programskog jezika osigurava jednakost izgleda neovisno o platformi. Održava ga tvrtka Twitter te mnogi individualni programeri. Bootstrap, osim što omogućava dizajn stranice, podržava i lagano raspoređivanje elemenata stranice korištenjem ugrađenih svojstava CSS-a. Boostrap se koristi pomoću ugrađenih klasa, a postoje i određeni prilagodni radni okviri koji Bootstrap omogućavaju u drugim radnim okvirima ili bibliotekama (dobar primjer je React Bootstrap za React).

```
<Row className="d-flex h-100">
  <Col
    xs={12}
    md={9}
    className="bg-wine-darker p-0 border-lg border-right border-rich-black flex-grow-1 d-flex column"
  >
```

Slika 4.4: Primjer korištenja React-Bootstrapa i Bootstrap klasa unutar Reacta.

4.1.3. Material-UI

Material-UI[15] je radni okvir otvorenog koda koji omogućava korištenje gotovih dijelova korisničkog sučelja koja prate načela *material designa*. Specijaliziran je za React biblioteku te se bilo koja Material-UI komponenta može koristiti kao obična React komponenta. Podržava i potpunu prilagodbu raspoloživih komponenata.

```
import { withStyles } from "@material-ui/core";
import Rating from "@material-ui/lab/Rating";
import { BsStar } from "react-icons/bs";

const StyledRating = withStyles({
  iconFilled: {
    color: "#f7f7ff",
  },
  iconHover: {
    color: "#ff3d47",
  },
})(Rating);

const Grade = ({ grade }) => {
  return (
    <StyledRating
      name="readOnlyGrade"
      value={grade}
      emptyIcon={<BsStar />}
      readOnly
    ></StyledRating>
  );
};

export default Grade;
```

Slika 4.5: Primjer korištenja Material-UI komponenata unutar Reacta.

4.1.4. Ace

Ace[1] je uređivač programskog koda kojeg je lagano integrirati unutar internetske stranice ili JavaScript aplikacije. Otvorena je koda te podržava sve funkcionalnosti koje uobičajen uređivač teksta podržava. Budući da je prvotno namijenjen pisanju programskog koda, podržava mnoštvo različitih jezika. Ima mogućnost naglašavanja sintakse odabranog programskog jezika, a također implementira rudimentarno nadopunjavanje programskog koda. Uz sve spomenute mogućnosti, sadržava različite teme te se one mogu mijenjati tijekom korištenja uređivača.

```
<AceEditor
  value={code}
  width={"100%"}
  mode={mode}
  theme="vibrant_ink"
  name="ACE_EDITOR_DIV"
  readOnly={view}
  highlightActiveLine={false}
  showPrintMargin={false}
  setOptions={{
    showLineNumbers: true,
    tabSize: 2,
  }}
></AceEditor>
```

Slika 4.6: Primjer korištenja Ace-Editor React komponente.

4.2. Tehnologije poslovnog sloja

Korištene tehnologije poslovnog sloja temeljene su na programskom jeziku Javi[20]. Uz Javu korišten je i Maven[2], alat za upravljanje projektom koji nudi mogućnosti upravljanja Java ovisnostima (engl. *dependencies*). Java ovisnosti su već napisani Java paketi koji proširuju osnovnu Javinu funkcionalnost.

4.2.1. Spring, Spring Boot i povezane tehnologije

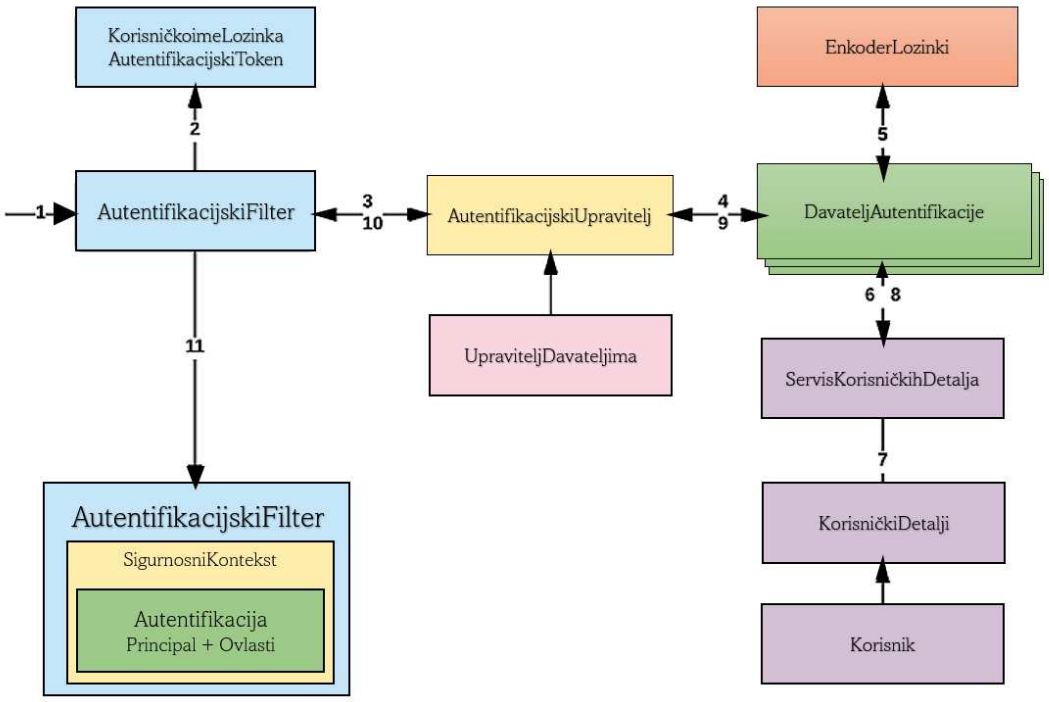
Spring[24] je radni okvir za različite programske jezike, od kojih je jedan Java. Otvorena je koda te podliježe svim zahtjevima modernih radnih okvira, a namjera mu je da olakša razvoj modernih poslovnih aplikacija. Razvoj olakšava pružanjem implementacija često korištenih funkcionalnosti i pružanjem mogućnosti uključivanja novih. Temeljna tehnika kojom Spring omogućuje uključivanje novih funkcionalnosti jest inverzija upravljanja (engl. *inversion of control*) koja omogućava radnom okviru da transparentno poziva klijentski kod korištenjem različitih dobro poznatih oblikovnih obrazaca (engl. *design patterns*). Inverziju upravljanja Spring radni okvir postiže korištenjem injekcije ovisnosti (engl. *dependency injection*), a ona je zapravo mogućnost radnog okvira da zadovolji tražene objektne ovisnosti. Drugim riječima, radni okvir Spring sam stvara potrebne objekte te ih "injektira" tamo gdje su traženi (za razliku od tradicionalnog pristupa gdje korisnik sam stvara tražene objekte). Sama implementacija funkcionalnosti injekcije ovisnosti unutar Springa veoma je složena, ali uporaba funkcionalnosti izrazito je jednostavna. Tražene objekte potrebno je anotirati s posebnom anotacijom *@Autowired*. Objekte koje treba stvoriti potrebno je anotirati s *@Bean* anotacijom. Pomoću tih anotacija, Spring može sam stvoriti i injektirati objekt. Inverzija ovisnosti i injekcija ovisnosti omogućava Springu prilagodljiv i lagano nadogradiv razvoj, što ga čini veoma pogodnim za razvoj internetskih aplikacija. Dodatno, da bi održao kvalitetnu strukturu razvijane aplikacije, Spring preporuča internu raspodjelu Spring aplikacije u tri sloja: "**Controller**" sloj (definira interakciju s klijentskim zahtjevima), "**Service**" sloj (ostvaruje temeljne funkcionalnosti) i "**Repository**" sloj (ostvaruje interakcije s podatkovnim slojem sveukupne aplikacije).
Spring radni okvir podržava velik broj dodataka koji zadovoljavaju različite korisničke potrebe, a od njih su korištene Spring Security[25] i Spring Data JPA[26].

```java
@Autowired
private JwtAuthenticationPoint jwtAuthenticationPoint;

@Value("${frontend.link}")
private String url;

@Override
protected void configure(AuthenticationManagerBuilder auth) throws Exception {
    auth.userDetailsService(myUserDetailsService);
}

@Override
protected void configure(HttpSecurity http) throws Exception {
    http.cors().and().csrf().disable().authorizeRequests()
            .antMatchers("/").permitAll()
            .antMatchers("/authenticate").permitAll()
            .antMatchers("/deauthenticate").permitAll()
            .antMatchers("/refresh").permitAll()
            .antMatchers("/programmer/register").permitAll()
            .antMatchers("/h2-console/**").permitAll()
            .anyRequest().authenticated()
            .and().exceptionHandling().authenticationEntryPoint(jwtAuthenticationPoint)
            .and().sessionManagement().sessionCreationPolicy(SessionCreationPolicy.STATELESS);
    http.headers().frameOptions().disable();
    http.addFilterBefore(jwtRequestFilter, UsernamePasswordAuthenticationFilter.class);
}

@Bean
public WebMvcConfigurer corsConfigurer() {
    You, 4 days ago | 1 author (You)
    return new WebMvcConfigurer() {
        @Override
        public void addCorsMappings(CorsRegistry registry) {
            registry.addMapping("/**").allowCredentials(true).allowedOrigins(url).allowedMethods("GET", "PUT", "POST", "DELETE");
        }
    };
}
```

Slika 4.7: Primjer injekcije ovisnosti te @Autowired i @Bean anotacija unutar Springa.

Spring Security

Spring Security je dodatak Spring radnom okviru koji transparentno omogućava autentifikaciju (engl. *authentication*) (proces je prikazan na slici 4.8) i autorizaciju (engl. *authorization*) dijelovima razvijene aplikacije . Autentifikacija je proces provjere identiteta korisnika, a autorizacija je provjera prava korištenja resursa kojeg korisnik traži. Spring Security te funkcionalnosti ostvaruje korištenjem lanaca filtera (engl. *filter chains*) od kojih svaki provjerava određene dijelove korisničkog zahtjeva.



Slika 4.8: Prikaz procesa autentifikacije unutar Spring Securityja.

Spring Data JPA

Spring Data JPA proširuje Spring radni okvir i njegov je cilj smanjenje količine programskog koda potrebnog za interakciju s podatkovnim slojem. Kao što mu i u imenu piše, Spring Data JPA podliježe JPA (skraćenica za Jakarta Persistence) specifikaciji. JPA nije ništa drugo doli specifikacija za upravljanje relacijskim podacima te upitnim jezikom (engl. *query language*) JPQL-om. Spring Data JPA za implementaciju specifikacije koristi Hibernate ORM (4.2.2).

Spring Boot

Naposlijetku, Spring Boot modul je Spring radnog okvira koji smanjuje broj konfiguracija Springa potrebnih za rad. Također uključuje početne ovisnosti, poput integriranog poslužitelja.

4.2.2. Hibernate

Hibernate[22] je radni okvir koji pruža automatsko preslikavanje razreda definiranih u programskog kodu u odgovarajuće relacije relacijske baze podataka. Takav tip radnih okvira znan je još kao ORM (engl. *object-relational mapping*) radni okviri. Implemen-

tiranje JPA specifikacije čini ga kompatibilnim sa Spring Data JPA proširenjem (4.2.1), a njegova se funkcija može shvatiti kao most između baze podataka (engl. *database*) i podatkovnog sloja aplikacije. Funkcionalnost preslikavanja između različitih oblika podataka (programskih razreda i relacija) ostvaruje generiranjem i izvršavanjem SQL upita.

4.3. Tehnologije podatkovnog sloja

Unutar podatkovnog sloja koristi se relacijska baza podataka. Interakcija s njom odvija se korištenjem SQL-a (engl. *Structured Query Language*).

4.3.1. PostgreSQL

PostgreSQL[28] je sustav za upravljanje bazom podataka otvorena koda. Dizajniran je za obradu i pohranu velike količine podataka. Popularan je i zbog toga ga podržavaju mnogi ORM radni okviri poput Hibernatea.

4.4. Pomoćne tehnologije

Tehnologije koje se same po sebi ne mogu svrstati ni u jedan od navedenih glavnih slojeva spomenute su u ovom potpoglavlju.

4.4.1. Axios

Axios[5] je lagan HTTP (engl. *Hypertext Transfer Protocol*) klijent pisan JavaScriptom. Podržava asinkrono slanje zahtjeva, funkcionalnost presretanja i modificiranje zahtjeva i odgovora.

```
axios
  .post(
    "/programmer/register",
    {
      username,
      email,
      password,
    },
    {
      baseURL: process.env.REACT_APP_BACKEND_SERVER_URL,
      withCredentials: true,
    }
  )
```

Slika 4.9: Primjer slanja POST zahtjeva pomoću Axiosa.

4.4.2. Judge0

Judge0[10] je izvršitelj programskoga koda otvorena koda kojeg Codeflow aplikacija koristi za provjeravanje rješenja pisanih za korisničke programske zadatke. Skalabilne je arhitekture te podržava preko 60 programskih jezika. Uz podršku prevođenja i izvođenja jednodatotečnih rješenja, Judge0 podržava prevođenje te izvođenje programskih rješenja koja se sastoje od više datoteka. Podržava konfiguriranje postavki korištenih prevoditelja, postavljanje argumenata komandne linije te još mnogo drugih funkcionalnosti.

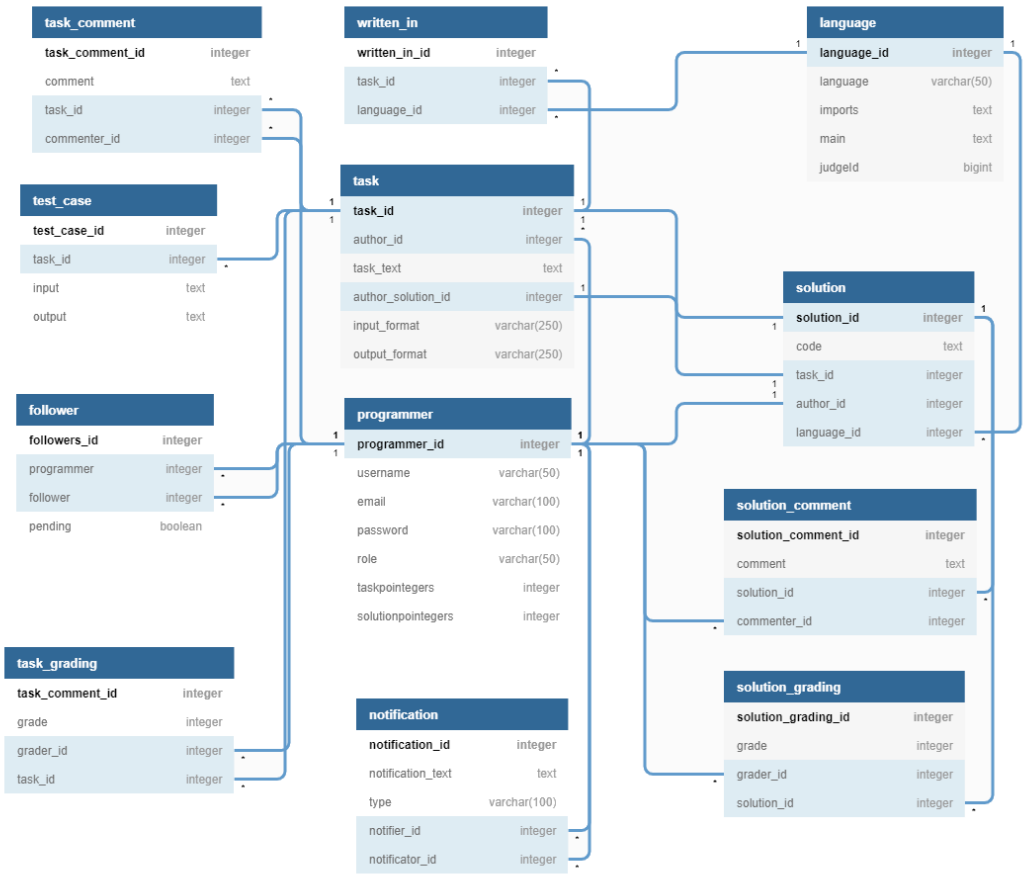
4.4.3. Git

Git[11] je sustav za upravljanje izvornim kodom distribuiranog tipa otvorena koda (engl. *distributed version control*). Upravljanje izvornim kodom vrši se pohranjivanjem koda unutar Git repozitorija koji mogu biti udaljeni ili lokalni. Udaljeni repozitoriji (engl. *remote*) najčešće su smješteni unutar nekog oblaka (engl. *cloud*) na nekoj Git platformi (primjer takve platforme je GitHub). Budući da je sustav distribuiran, lokalni repozitoriji sadržavaju svu povijest programskog koda koju i udaljeni. Inicijalne promjene izvornog koda događaju se unutar lokalnog repozitorija te tek nakon što se spreme unutar lokalnog repozitorija mogu se poslati na udaljeni repozitorij. Korištenje Gita osigurava strukturirano i kvalitetno praćenje te spremanje projekta.

5. Arhitektura rješenja

5.1. Domena rješenja

Domena rješenja ostvarena je definicijom relacijske baze podataka pomoću PostgreSQL sustava za upravljanje bazama. Iako korišteni ORM Hibernate podržava *code-first* pristup definiranja baze podataka (tablice se definiraju programskim razredima, a ne programski razredi tablicama) ipak je odabran pristup gdje se prvo definira shema korištene baze podataka (engl. *database-first*). Iz tog je razloga shema baze podataka koja opisuje domenu problema Codeflow aplikacije sa svim tablicama i njihovim međusobnim vezama prikazana na slici 5.1.



Slika 5.1: Kompletan prikaz sheme Codeflow baze podataka.

Valja napomenuti da sve tablice dijele atribute ts_created, ts_modified, user_created, user_modified radi lakšeg praćenja stvaranja i promjena zapisa unutar tablica. Zbog toga što su dijeljeni neće biti dodatno spomenuti tijekom detaljnijeg opisa svake tablice. Same tablice oblikovane su prema korisničkim zahtjevima (definirani u poglavlju 2).

Tablica 5.1: Dijeljeni atributi

Ime atributa	Tip	Opis
ts_created	timestamp with timezone	vrijeme nastanka
ts_modified	timestamp with timezone	vrijeme modifikacije
user_created	varchar(50)	korisnik koji je napravio zapis
user_created	varchar(50)	korisnik koji je zadnji modificirao zapis

Tablica **programer** opisuje korisnika aplikacije Codeflow te predstavlja centralnu tablicu. Stvara se tijekom registracije korisnika te pohranjuje sve korisničke informacije. Vlastiti atributi tablice programer su programmer_id, username, email, password, role, taskpoints i solutionpoints. Role atribut je dodan, ali unutar ove verzije različite korisničke uloge nisu implementirane.

Tablica 5.2: Tablica programmer

Ime atributa	Tip	Opis
programmer_id	integer	surogatni primarni ključ
username	varchar(50)	korisnikovo korisničko ime, mora biti jedinstveno
email	varchar(100)	korisnikov e-mail, mora biti jedinstven
password	varchar(100)	*hash* korisnikove lozinke
role	varchar(50)	korisnikova uloga
taskpoints	integer	bodovi koje je korisnik dobio zbog kvalitete svojih zadataka
solutionpoints	integer	bodovi koje je korisnik dobio zbog kvalitete svojih rješenja

Tablica **task** opisuje zadatak kojeg je korisnik napravio. Tijekom stvaranja zadatka spremaju se novi podaci unutar tablice task, ali i unutar tablice test_case jer ti podaci su neophodni za dinamičku provjeru ispravnosti zadatka. Vlastiti atributi tablice task su task_id, author_id, task_text, author_solution_id, input_format i output_format.

Tablica 5.3: Tablica task

Ime atributa	Tip	Opis
task_id	integer	surogatni primarni ključ
author_id	integer	strani ključ autora zadatka
task_text	text	tekst zadatka
author_solution_id	integer	autorovo rješenje (nije nužno)
input_format	varchar(250)	primjer ulaza
output_format	varchar(250)	primjer izlaza

Tablica **solution** opisuje rješenje kojeg je korisnik napravio. Vlastiti atributi tablice solution su solution_id, code, task_id, author_id, task_text i language_id.

Tablica 5.4: Tablica solution

Ime atributa	Tip	Opis
solution_id	integer	surogatni primarni ključ
code	text	kod rješenja
task_id	integer	strani ključ zadatka rješenja
author_id	integer	strani ključ autora zadatka
language_id	integer	strani ključ jezika u kojem je rješenje pisano

Tablica **notification** opisuje obavijest koja biva poslana korisniku tijekom nekog događaja unutar aplikacije. Vlastiti atributi tablice notification su notification_id, notification_text, type, notifier_id, notificator_id. Prate se tipovi obavijesti jer ovisno o njima vizualno se razlikuju prikazane notifikacije.

Tablica 5.5: Tablica notification

Ime atributa	Tip	Opis
notification_id	integer	surogatni primarni ključ
notification_text	text	sadržaj obavijesti
type	varchar(100)	tip obavijesti
notifier_id	integer	strani ključ korisnika koji je uzrokovao obavijest
notificator_id	integer	strani ključ korisnika koji treba vidjeti obavijest

Tablica **language** opisuje podržani jezik koji je korisniku dostupan tijekom stvaranja zadataka. Korisniku koji rješava predstavlja, ako je ponuđen unutar opisa zadatka, mogući jezik rješenja. Vlastiti atributi tablice language su language_id, language, imports, main, judgeId. Atribut judgeId koristi se tijekom interakcije aplikacije s Judge0 izvršiteljem računskog koda.

Tablica 5.6: Tablica language

Ime atributa	Tip	Opis
language_id	integer	surogatni primarni ključ
language	varchar(50)	ime programskog jezika
imports	text	uvezeni dijelovi koda
main	text	početni programski kod
judgeId	bigint	jedinstveni identifikator programskog jezika unutar sustava Judge0

Tablica **test_case** opisuje testni slučaj koji korisnik zadaje tijekom izrade zadatka, a koji se koristi kod dinamičke provjere zadatka. Vlastiti atributi tablice test_case su test_case_id, task_id, input i output.

Tablica 5.7: Tablica test_case

Ime atributa	Tip	Opis
test_case_id	integer	surogatni primarni ključ
task_id	integer	strani ključ zadatka kojem testni slučaj pripada
input	text	ulaz testnog slučaja
output	text	izlaz testnog slučaja

Tablica **written_in** je spojna relacija kojom se ostvaruje NN odnos između tablica language i task. Vlastiti atributi tablice written_in su written_in_id, task_id i language_id.

Tablica 5.8: Tablica written_in

Ime atributa	Tip	Opis
written_in_id	integer	surogatni primarni ključ
task_id	integer	strani ključ zadatka
language_id	integer	strani ključ jezika

Tablica **solution_grading** predstavlja ocjene vezane uz jedno programsko rješenje. Vlastiti atributi tablice solution_grading su solution_grading_id, grade, grader_id i solution_id.

Tablica 5.9: Tablica solution_grading

Ime atributa	Tip	Opis
solution_grading_id	integer	surogatni primarni ključ
grade	integer	ocjena
grader_id	integer	strani ključ korisnika ocjenjivača
solution_id	integer	strani ključ ocjenjenog rješenja

Tablica **task_grading** predstavlja ocjene vezane uz jedan programski zadatak. Vlastiti atributi tablice task_grading su task_grading_id, grade, grader_id i task_id.

Tablica 5.10: Tablica task_grading

Ime atributa	Tip	Opis
task_grading_id	integer	surogatni primarni ključ
grade	integer	ocjena
grader_id	integer	strani ključ korisnika ocjenjivača
task_id	integer	strani ključ ocjenjenog zadatka

Tablica **solution_comment** predstavlja komentare vezane uz jedno programsko rješenje. Vlastiti atributi tablice solution_comment su solution_comment_id, comment, commenter_id i task_id.

Tablica 5.11: Tablica solution_comment

Ime atributa	Tip	Opis
solution_comment_id	integer	surogatni primarni ključ
comment	text	komentar
solution_id	integer	strani ključ komentiranog rješenja
commenter_id	integer	strani ključ korisnika komentatora

Tablica **task_comment** predstavlja komentare vezane uz jedan programski zadatak. Vlastiti atributi tablice task_comment su task_comment_id, comment, grader_id i

commenter_id.

Tablica 5.12: Tablica task_comment

Ime atributa	Tip	Opis
task_comment_id	integer	surogatni primarni ključ
comment	text	komentar
task_id	integer	strani ključ komentiranog zadatka
commenter_id	integer	strani ključ korisnika komentatora

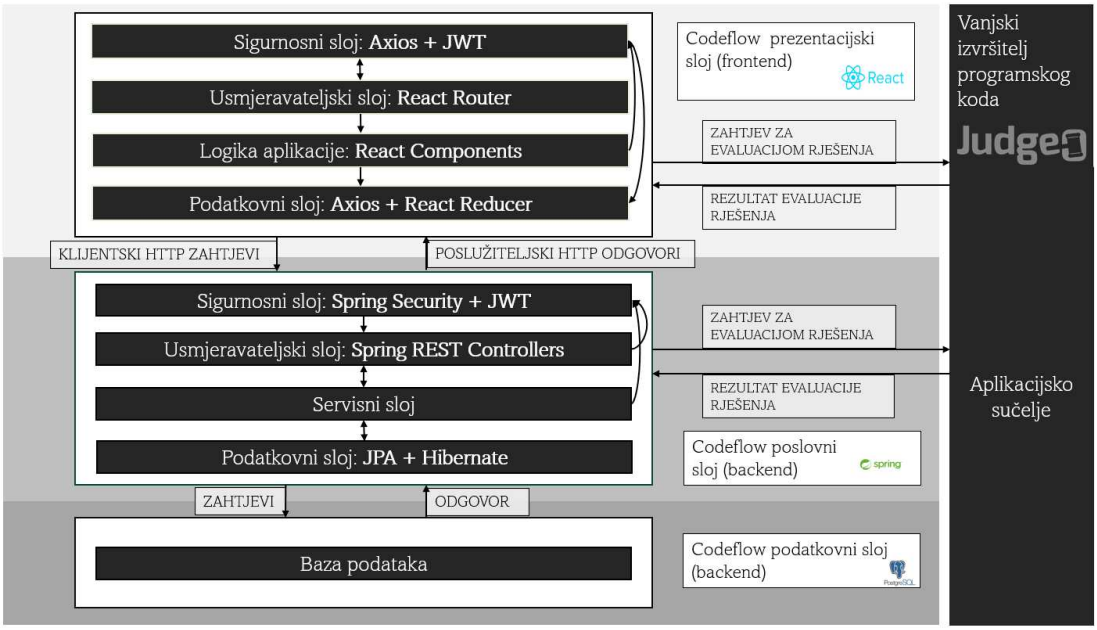
Tablica **follower** opisuje pratiteljstva između korisnika. Vlastiti atributi tablice follower su followers_id, follower i pending. Pending atribut označava je li prihvaćeno pratiteljstvo. Tek ako je pending atribut postavljen na *false* pratiteljstvo se smatra punopravnim.

Tablica 5.13: Tablica follower

Ime atributa	Tip	Opis
followers_id	integer	surogatni primarni ključ
programmer	text	strani ključ praćenog korisnika
follower	integer	strani ključ pratitelja
pending	boolean	provjera punopravnosti pratiteljstva

5.2. Implementacija rješenja

Implementacija Codeflow aplikacije strukturirana je prema modernoj troslojnoj arhitekturi (spomenuta tijekom početka poglavlja 4). Prezentacijski sloj (engl. *frontend*), poslovni sloj i podatkovni sloj (engl. *backend*) istaknuti su na slici 5.2 arhitekture aplikacije.



Slika 5.2: Arhitektura Codeflow aplikacije s naglašenim bitnijim tehnologijama i interakcijama svakog sloja.

Implementacija prezentacijskog sloja

Prezentacijski sloj definira korisničko sučelje aplikacije te je za njegovu izradu korišten React. React aplikacija prezentacijskog sloja sastoji se od trinaest stranica od kojih svaka predstavlja određeni pogled definiran korisničkim zahtjevima, a same stranice koriste i dijele manje komponente. Radi bolje strukturiranog razvoja, React aplikacija dodatno je podijeljena je u četiri interna sloja: **sigurnosni sloj**, **usmjeravateljski sloj**, **sloj logike aplikacije** i **podatkovni sloj**.

Korisnički zahtjevi razlikuju prijavljene i neprijavljene korisnike pa tako korisničko sučelje razlikuje i definira akcije dopuštene prijavljenim i neprijavljenim korisnicima. Tijekom uporabe, neprijavljeni korisnici isprva pristupaju **sigurnosnom sloju**. Sigurnosni sloj nakon uspješne prijave od poslovnog sloja dobiva dva *http-only* kolačića (engl. *cookie*). Oba sadržavaju *JSON Web Tokens*[3] (skraćeno JWT) i unutar sebe opisuju osnovne korisnikove podatke. Njihovim primitkom React aplikacija sigurna je da je korisnik uspješno ovjerio svoj identitet i da smije spremiti i koristiti njegove podatke. *JSON Web Tokens* podijeljeni su u tri dijela: *header* dio koji opisuje tip tokena i korišteni kriptografski algoritam, *payload* dio koji sadržava korisničke informacije i dio s potpisom koji je kriptirani sažetak prva dva dijela. Prvi kolačić šalje se uz svaki prijavljeni korisnički zahtjev, a drugi kolačić koristi se za obnovu prvog kolačića jer vremensko trajanje prvog kolačića vrlo je kratko (dvije minute). Odbijenost ko-

risničkog zahtjeva zbog vremenskog isteka prvog kolačića ispravlja se pomoću *Axios Response Interceptor* funkcionalnosti koja šalje novi zahtjev za obnovu kolačića. Novi zahtjev sadržava drugi kolačić. U slučaju odbijanja produljenja JWT kolačića korisnik se odjavljuje dok se u slučaju produljenja originalno odbijen zahtjev ponovno šalje. Provjera ispravnosti poslanih kolačića obrađuje se unutar poslovnog sloja. Kombinacija JWT kolačića i provjere spremljenih podataka unutar lokalnog spremnika internetskog preglednika tijekom promjene stranica osigurava ispravno raspoznavanje prijavljenih korisnika i neprijavljenih korisnika.

```
axiosInstance.interceptors.response.use(
  (response) => {
    return response;
  },
  async function (error) {
    const originalRequest = error.config;

    if (error.response === undefined) {
      dispatch({ type: authActions.LOGOUT });
      history.push("/");
    }
    if (error.response.status === 401 && !originalRequest._retry) {
      originalRequest._retry = true;
      await axiosInstance.get("/refresh");
      return axiosInstance(originalRequest);
    }
    if (error.response.status === 401) {
      console.log(error);
    } else if (error.response.status === 403) {
      await axiosInstance.get("/deauthenticate");
      if (dispatch) {
        dispatch({ type: authActions.LOGOUT });
      } else {
      }
      if (history) {
        history.push("/");
      } else {
        window.location = "/";
      }
    } else {
      dispatch({
        type: authActions.ERROR,
        payload: error.response ? error.response.data : "COULD NOT CONNECT",
      });
    }
  }
);
```

Slika 5.3: Primjer korištenja Axios Interceptor funkcionalnosti.

Usmjeravateljski sloj koristi React Router tehnologiju koja (uz povratnu informaciju sigurnosnom sloju) preusmjerava korisnike na tražene stranice.
Poslovna logika aplikacije definirana korisničkim zahtjevima obrađena je unutar **sloja logike aplikacije**. Sloj logike aplikacije uvelike koristi enkapsuliranu prirodu React komponenata. Svaka komponenta interno pohranjuje svoj izgled i funkcionalnost, a one variraju od posebnih gumbiju pa sve do tablica s rješenjima. Unutar ovog sloja zahtjevi usmjereni prema Judge0 izvršitelju slani su tijekom provjere korisničkog rje-

šenja zadatka.
U konačnici, **podatkovni sloj** koristi Axios biblioteku za slanje zahtjeva te React Reducer funkcionalnost Reacta za strukturirano spremanje i upravljanje podacima unutar React aplikacije.

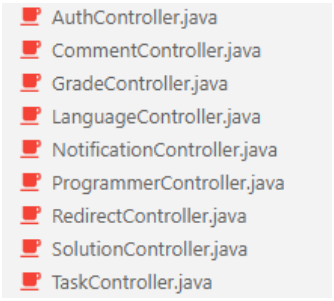
Implementacija poslovnog sloja

Poslovni sloj Codeflow aplikacije sastoji se od internetskog poslužitelja pisanog Java programskim jezikom uz pomoć Spring radnog okvira. Arhitekturalni stil internetskog poslužitelja prilagođen je REST (engl. *representational state transfer*) arhitekturalnom stilu. Unutar REST arhitekturalnog stila, podaci i funkcionalnosti se smatraju resursima i pristupa im se pomoću URI-ja (engl. *Uniform Resource Identifiers*). Sadržaj resursa može se oblikovati u različite medijske tipove, često zvane MIME (engl. *Multipurpose Internet Mail Extensions*). Najčešći medijski tipovi sadržaja resursa su JSON (engl. *Javascript Object Notation*), XML (engl. *Extensible Markup Language*) i HTML. REST također propisuje točno određene metode za pristup i modifikaciju resursa[23].
Internetski poslužitelj, analogno prezentacijskom sloju, podijeljen je na interne slojeve radi kvalitetnije strukture. Slojevi po redu su: **sigurnosni sloj**, **usmjeravateljski sloj**, **servisni sloj** i **podatkovni sloj**. Semantički gledano, navedeni slojevi (izuzev sigurnosnog sloja) podliježu raspodjeli koju predlaže Spring radni okvir.
Unutar **sigurnosnog sloja** dodan je poseban *JwtFilter* koji služi za provjeru JWT kolačića spomenutih unutar implementacije prezentacijskog sloja. U slučaju neispravnog JWT kolačića unutar zahtjeva koji pristupa zaštićenom resursu, zahtjev biva odbijen i ne prosljeđuje se na usmjeravateljski sloj. JWT kolačić neispravan je ako je vremenski istekao ili ako se nanovo izračunati kriptografski sažetak, koji se računa iz prva dva dijela tokena, ne podudara sa starim kriptografskim sažetkom zapisanim unutar dobivenog tokena. *JwtFilter* ignorira zahtjeve za prijavu, registraciju i obnovu tokena, a uspješna prijava i obnova tokena generira nov par JWT kolačića pomoću **jjwt**[14] maven ovisnosti. Ista maven ovisnost koristi se i za provjeru tokena.

```
Cookie[] cookies = httpServletRequest.getCookies();
String jwt = null;
String username = null;


if(!(httpServletRequest.getRequestURI().equals("/authenticate")
        || httpServletRequest.getRequestURI().equals("/refresh")
        || httpServletRequest.getRequestURI().equals("/deauthenticate")
        || httpServletRequest.getRequestURI().equals("/user/register") )) {
    try {
        if(cookies != null) {
            jwt = CookieUtils.extractFieldFromCookie(cookies, "Access-token");
            if(jwt != null)
                username = jwtUtil.extractUsername(jwt);
        }
    } catch (JwtException ex) {
        {
            jwtAuthenticationPoint.commence(httpServletRequest, httpServletResponse, new BadCredentialsException("Bad authorization."));
            return;
        }
    }
}


if(username != null && SecurityContextHolder.getContext().getAuthentication() == null) {
    UserDetails userDetails = userDetailsService.loadUserByUsername(username);
    if(jwtUtil.validateToken(jwt, userDetails)) {
        UsernamePasswordAuthenticationToken usernamePasswordAuthenticationToken =
                new UsernamePasswordAuthenticationToken(userDetails, null, userDetails.getAuthorities());
        usernamePasswordAuthenticationToken.setDetails(new WebAuthenticationDetailsSource().buildDetails(httpServletRequest));
        SecurityContextHolder.getContext().setAuthentication(usernamePasswordAuthenticationToken);
    }
}

filterChain.doFilter(httpServletRequest, httpServletResponse);
```

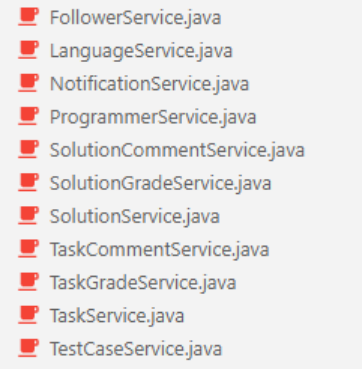
Slika 5.4: Programski kod JwtFiltera.

Ako zahtjev uspješno prođe sigurnosni sloj, stiže na **usmjeravateljski sloj** koji se sastoji od specijalnih REST Spring kontrolera koji Spring radnom okviru nalažu da vrijednost prilagodi JSON formatu i pospremi unutar tijela HTTP odgovora. Usmjeravateljski sloj sadržava kontroler za svaki od glavnih entiteta spomenutih unutar domene rješenja (5.1) te dodatni kontroler posvećen sigurnosti aplikacije. Kontroleri ne provode poslovnu logiku te pozivaju servisni sloj za daljnju obradu podataka.



Slika 5.5: Prikaz svih REST kontrolera usmjeravateljskog sloja.

Servisni sloj sadržava servise unutar kojih se provodi poslovna logika diktirana korisničkim zahtjevima.



Slika 5.6: Prikaz svih servisa servisnog sloja.

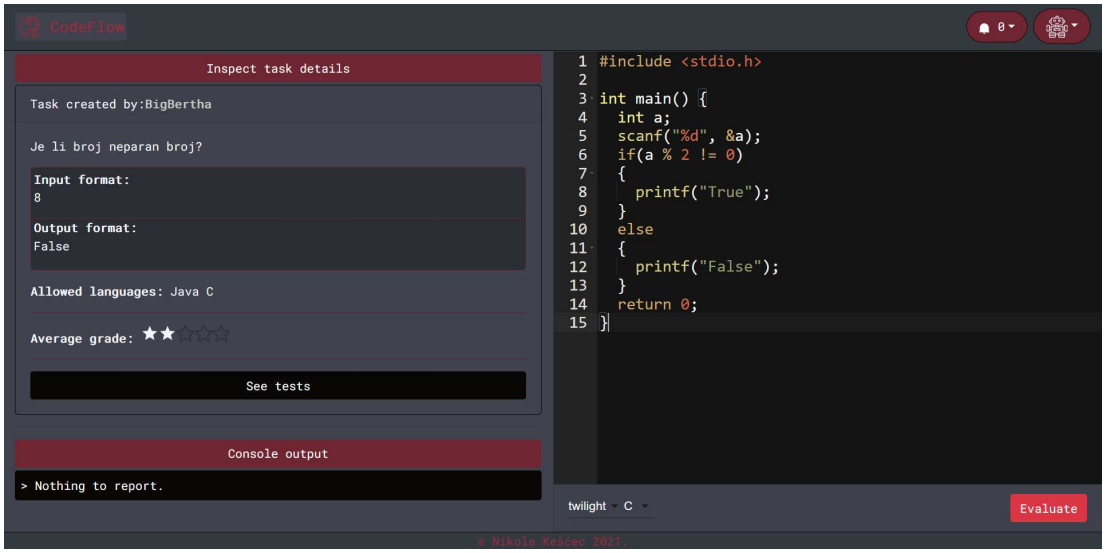
Podatkovni sloj sastoji se od programskih klasa (znane još kao i model) koje opisuju entitete baze podataka te JPA repozitorijska (engl. *repositories*) sučelja. Repozitorijska sučelja interno unutar svojih implementacija provode prevođenje programskih naredbi u SQL naredbe te redaka dobivenih SQL upitima u objekte definiranih prigodnim modelom. Interna implementacija uvelike ovisi o Hibernate ORM radnom okviru. Podatkovni sloj poslovnog sloja jedini je sloj koji šalje zahtjeve bazi podataka.

Implementacija podatkovnog sloja

Podatkovni sloj sastoji se od PostgreSQL sustava za upravljanje bazama podataka. Omogućuje pohranjivanje i izmjenu podataka.

6. Evaluacija i rangiranje rješenja programskog zadatka

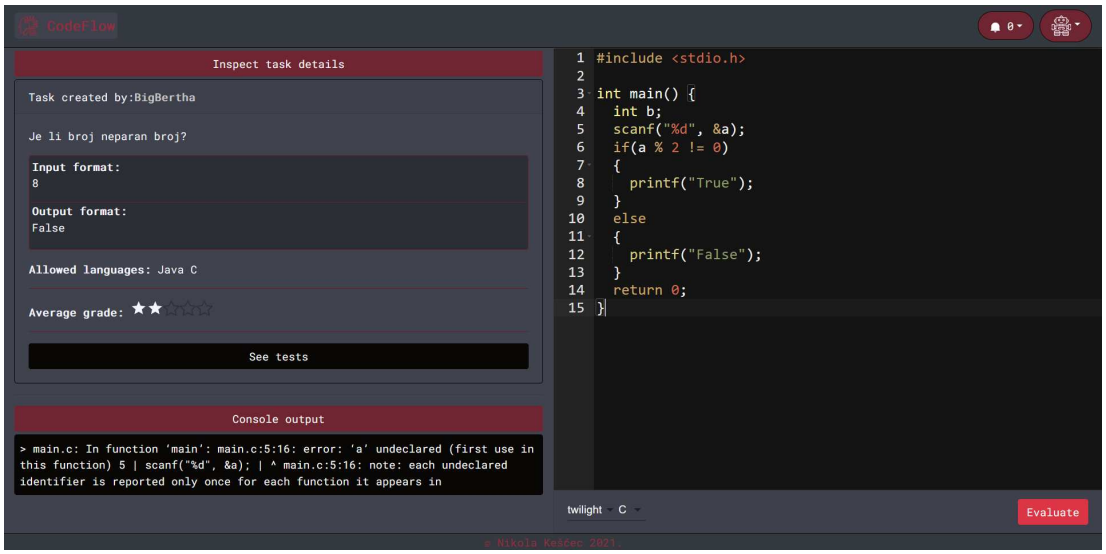
Provjeru točnosti rješenja programskog zadatka definiraju autori zadataka putem proizvoljnog broja ulazno-izlaznih parova. Za svaki par se provjerava vraća li program odgovarajući izlaz za zadani ulaz. Testni slučajevi unutar Codeflow aplikacije definirani su kao par ulaznih i izlaznih vrijednosti koji nikad ne mogu biti prazni. Sama provjera dobivene izlazne vrijednosti i očekivane vrijednosti delegirana je Judge0 izvršitelju programskog koda.



Slika 6.1: Prikaz rješenja zadatka spremnog za evaluaciju. S lijeve strane unutar opisa zadatka vidljiv je primjer ulaznih i izlaznih testnih vrijednosti. Ispod detaljnog prikaza smješteno je područje koje ispisuje rezultate evaluacije.

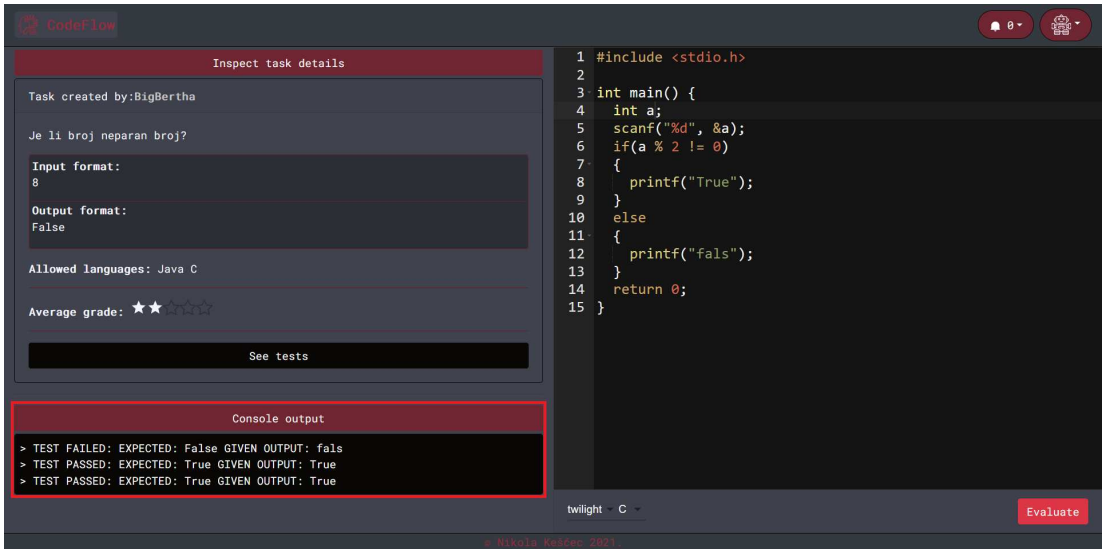
Prvi korak evaluacije zadatka sastoji se od pripreme zahtjeva koji sadrži programski kod rješenja, identifikacijski broj programskog jezika rješenja, dio testnog slučaja koji definira ulaz te dio testnog slučaja koji definira očekivani izlaz rješenja. Pošto je zah-

tjev pripremljen, šalje se Judge0 izvršitelju programskog koda. Ako Judge0 uspješno prevede i izvede rješenje uspoređuje izlaz dobiven kao rezultat izvođenja programskog koda s očekivanim izlazom unutar tijela zahtjeva, inače vraća grešku. Nakon što aplikacija Codeflow zaprimi odgovor, pregledava tijelo odgovora. Ako tijelo odgovora sadržava obavijesti o bilo kakvim greškama nastalim tijekom prevođenja ili izvođenja, aplikacija grešku prikazuje korisniku te prestaje s daljnjom evaluacijom zadatka (prikazano na slici 6.2).



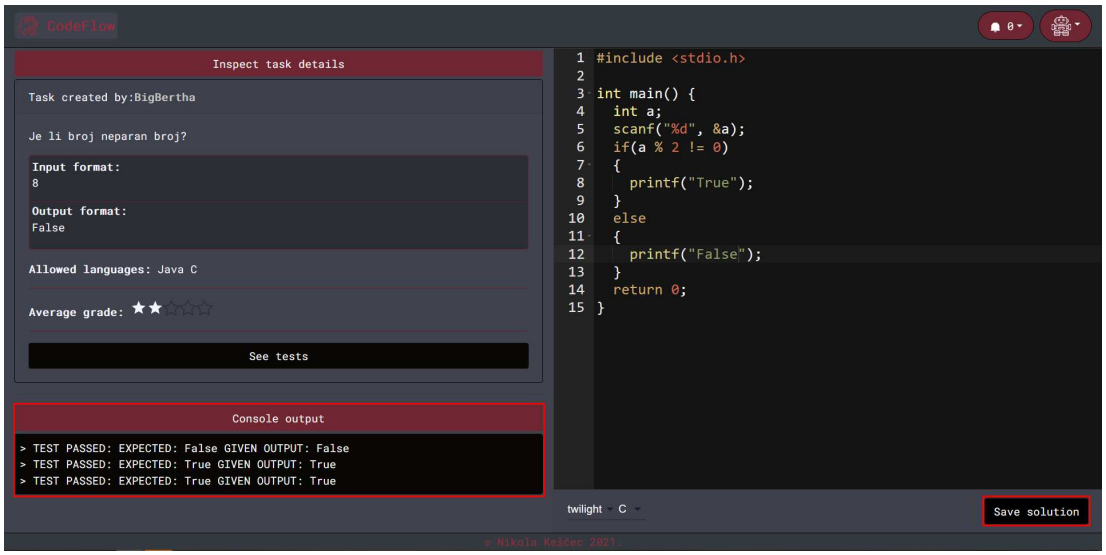
Slika 6.2: Prikaz rezultata evaluacije rješenja koje sadržava sintaksnu pogrešku.

Nadalje, ako se izlaz prevedenog i izvedenog rješenja ne podudara s očekivanim izlazom evaluacija se ne prekida, već se izvodi za sve dane ispitne slučajeve (nepodudarnosti se interno prate). Tek nakon izvođenja svakog od testnog slučaja bez grešaka korisnik biva obavješten o rezultatima podudarnosti dobivenih izlaza i očekivanih izlaza (prikaz je vidljiv na slici 6.3).



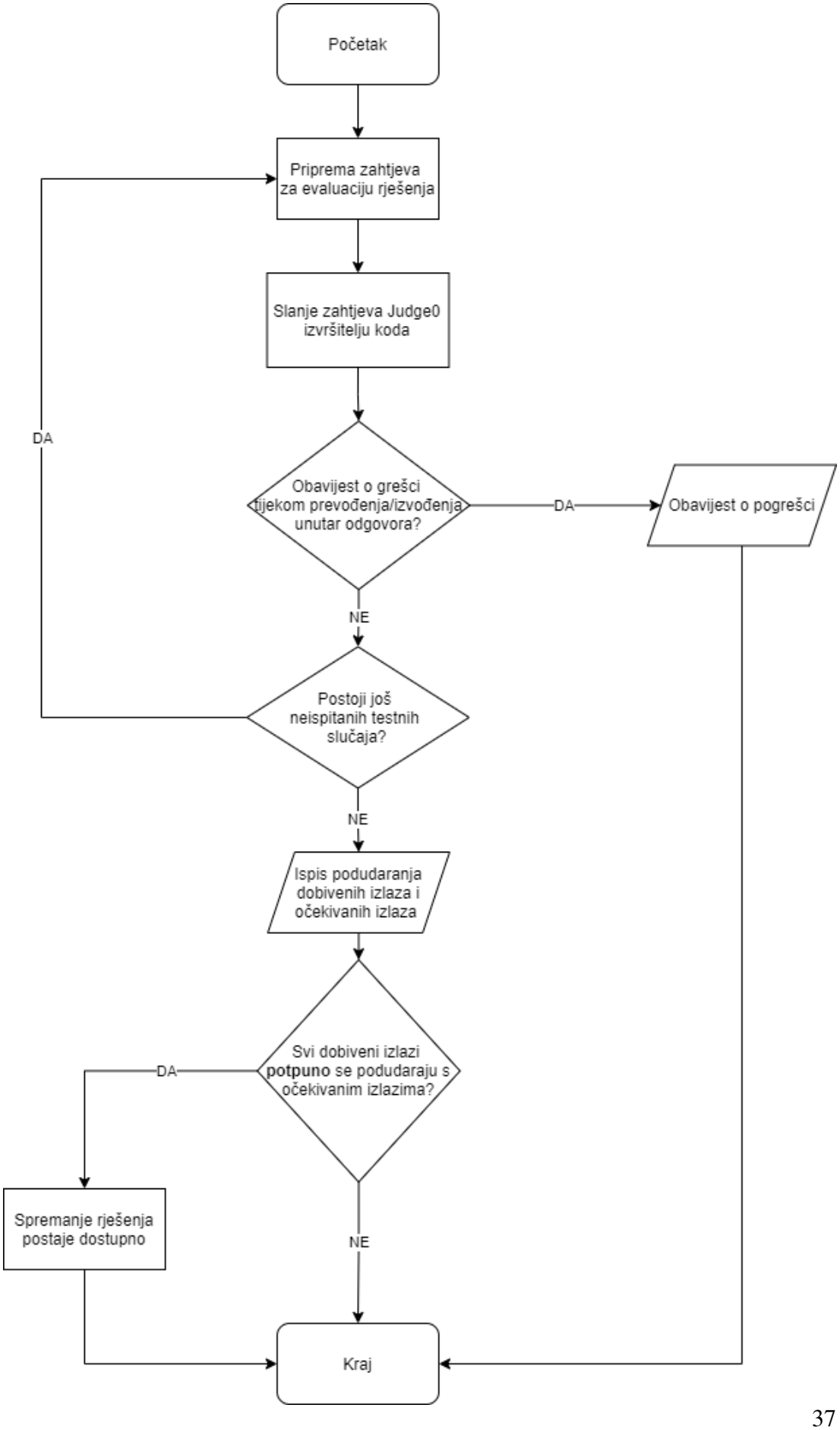
Slika 6.3: Prikaz evaluacije s nepodudaranjima. Naglašen je prostor rezultata usporedbe.

Bitna napomena ovog sustava evaluacije jest da mogućnost spremanja rješenja programskog zadatka postaje dostupna samo u slučaju **potpune** podudarnosti svih dobivenih izlaza i očekivanih izlaza navedenih unutar testnih slučajeva. Prikaz uspješne evaluacije vidljiv je na slici 6.4.



Slika 6.4: Prikaz evaluacije s potpunim podudaranjem. Naglašen je prostor rezultata usporedbe i omogućeno spremanje rješenja zadatka.

Grafički prikaz izvođenja evaluacije rješenja programskog zadatka unutar Codeflow aplikacije prikazan je na slici 6.5.



Slika 6.5: Prikaz algoritma evaluacije rješenja zadatka unutar Codeflow aplikacije.

Rangiranje rješenja provedeno je rangiranjem prosječne ocjene rješenja. Korisnici aplikacije ocjenjuju rješenja te se na temelju njihovih ocjena izračunava prosječna ocjena. Korisnici rješenje mogu ocijeniti samo jednom, a ako promijene svoje mišljenje u vezi dane ocjene, mogu ju izbrisati ili promijeniti. Pristup rangiranja prosječnom ocjenom odabran je zbog klijentske interakcije. Alternativan pristup rangiranja korisničkih rješenja bio bi na temelju trajanja izvršavanja rješenja, ali on nije odabran zbog nemogućnosti osiguravanja potpuno identičnih uvjeta tijekom izvođenja svakog korisničkog rješenja.

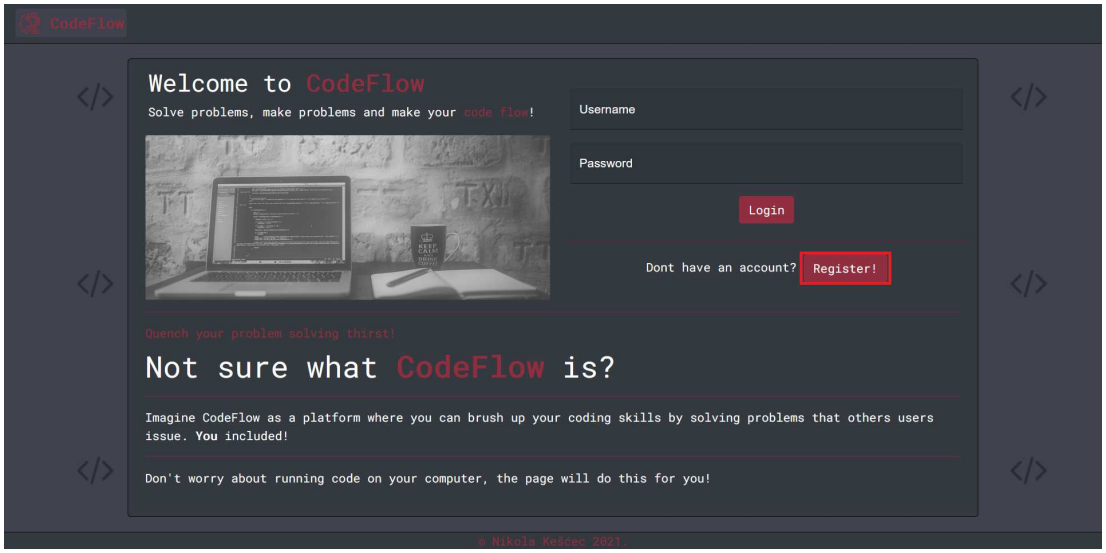
7. Primjer karakteristične korisničke uporabe

Ovo poglavlje posvećeno je razmatranju uporabe i ostvarenih funkcionalnosti Codeflow internetske aplikacije iz perspektive novog korisnika. Očekivani radni tok novog korisnika sastoji se od:

1. pregleda stranice dobrodošlice
2. registracije
3. prijave
4. pregleda sažetaka zadataka
5. praćenja korisnika
6. prihvaćanja ili odbijanja zahtjeva za pratiteljstvo
7. detaljnog pregleda zadatka (uz opcionalno komentiranje i uvjetovano ocjenjivanje)
8. pregleda rješenja zadatka (uz opcionalno komentiranje i ocjenjivanje)
9. rješavanja odabranog zadatka
10. stvaranja zadatka

7.1. Pregled stranice dobrodošlice

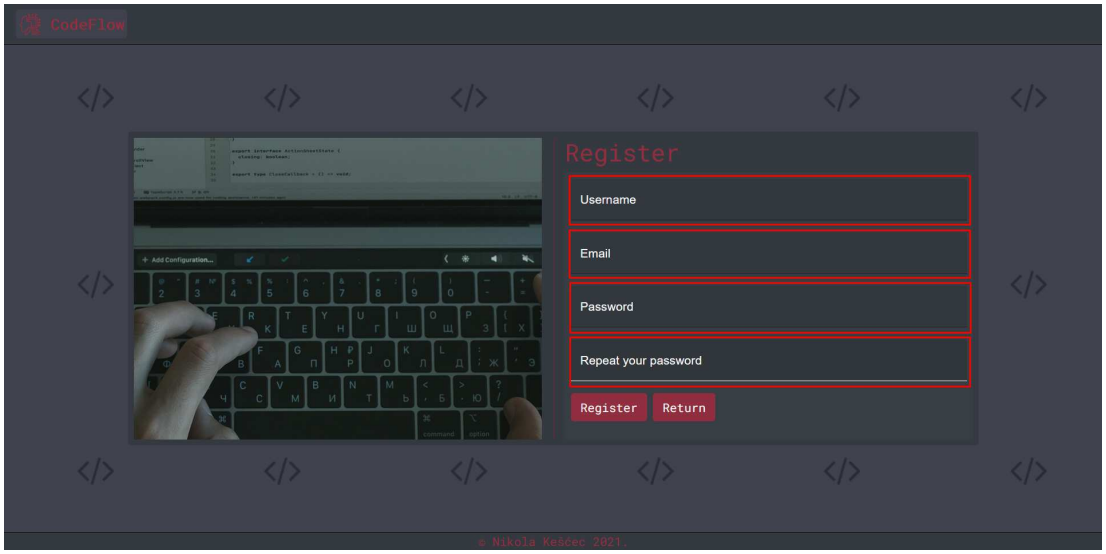
Korisnici prvim pristupom Codeflow aplikaciji pregledavaju stranicu dobrodošlice. Ako nemaju korisnički račun, pritiskom na gumb *Register* preusmjerava ih se na stranicu za registraciju.



Slika 7.1: Prikaz stranice dobrodošlice s naglašenim gumbom koji vodi na stranicu za registraciju.

7.2. Registracija

Dolaskom na stranicu posvećenu registraciji korisnika neregistrirani korisnici moraju upisati podatke koji će se koristiti za stvaranje njihovog korisničkog računa. Polja za ispunu označena na slici 7.2 obavezna su te svako od njih sadržava dodatne uvjete. *Username* i *Email* polja moraju biti jedinstvena, *Password* polje mora sadržavati lozinku dulju od 8 znakova te sadržaj *Repeat your password* polja mora biti jednak *Password* polju. Tek kada su navedeni uvjeti polja ispunjeni, korisnici pritiskom gumba *Register* stvaraju novi korisnički račun s kojim se mogu prijaviti.



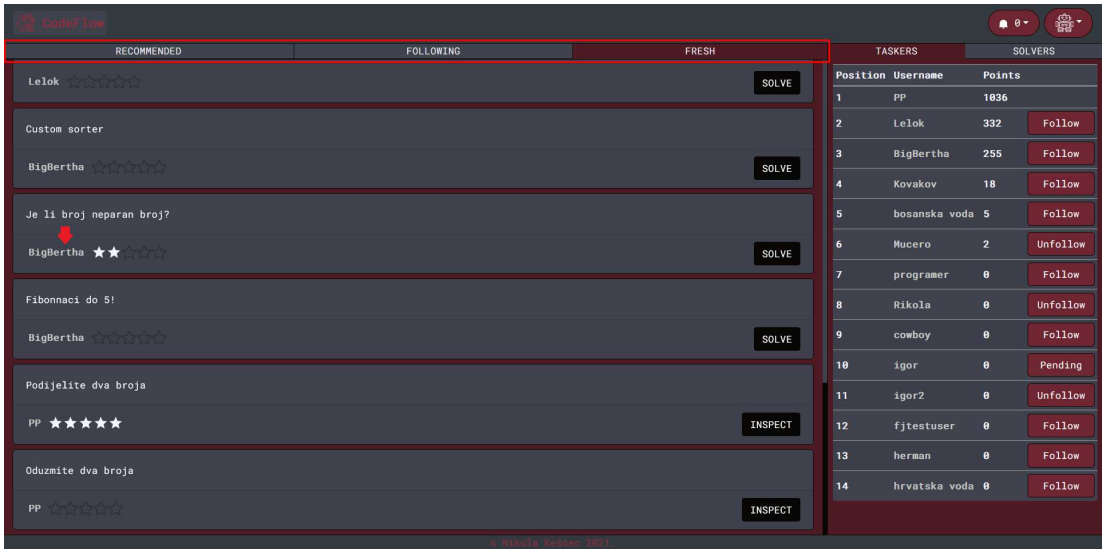
Slika 7.2: Prikaz stranice registracije s označenim elementima forme za prijavu.

7.3. Prijava

Registrirani korisnici mogu se prijaviti pomoću podataka koje su tijekom registracije definirali. Podaci koji su relevantni za prijavu su *Username* i *Password* (vidljivi na slici 7.1). Ako tijekom prijave bilo koji od tih podataka nije ispravan, Codeflow aplikacija će obavijestiti korisnika o pogrešci, ali neće naglasiti koji od podataka nije točan (iz sigurnosnih razloga).

7.4. Pregled sažetaka zadataka

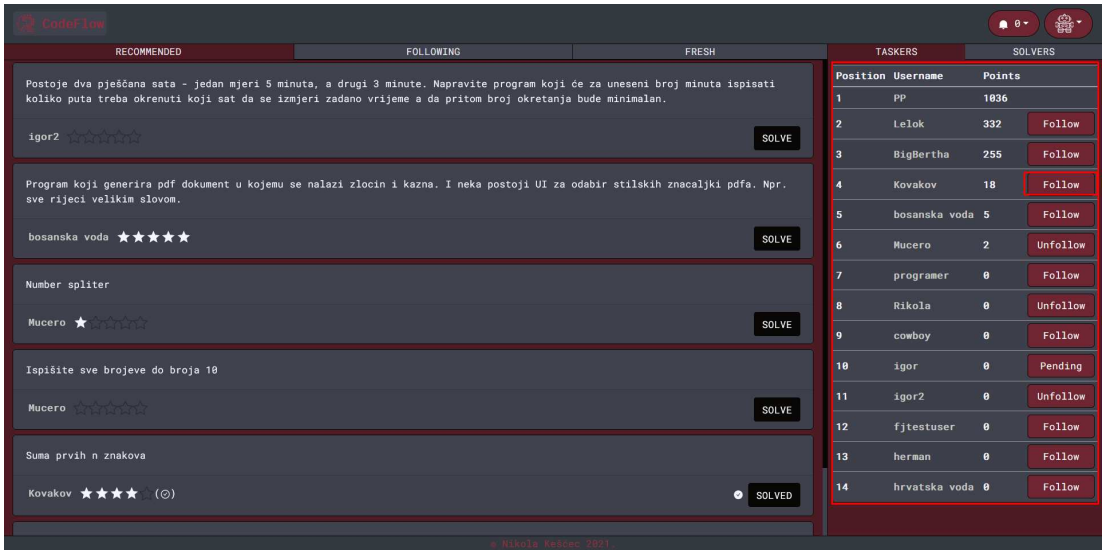
Nakon prijave, korisnici mogu pregledavati ponuđene sažetke zadataka. Sažetak zadatka sadržava opis zadatka, korisničko ime autora zadatka i prosječnu ocjenu zadatka. Ponuđene sažetke korisnici mogu filtrirati prema *Recommended*, *Following* i *Fresh* opcijama. *Recommended* je prva opcija i ona predlaže korisnicima samo sažetke zadataka čiji su autori korisnici koje prijavljeni korisnici prate te sažetke zadataka s prosječnom ocjenom većom od određenog praga. Druga, *Following* opcija, isto tako prikazuje isključivo sažetke zadataka čiji su autori korisnici koje prijavljeni korisnici prate. Treća, *Fresh* opcija, prikazuje sažetke zadataka prema njihovoj starosti. Stranica s filtriranim sažetcima zadataka nije jedina stranica gdje je moguće vidjeti sažetke zadataka jer korisnici mogu pregledavati sažetke zadataka pojedinačnih korisnika na njihovim korisničkim stranicama. Do pojedinačnih korisničkih stranica dolazi se pritiskom na odabrano korisničko ime (strelicom naznačeno na slici 7.3).



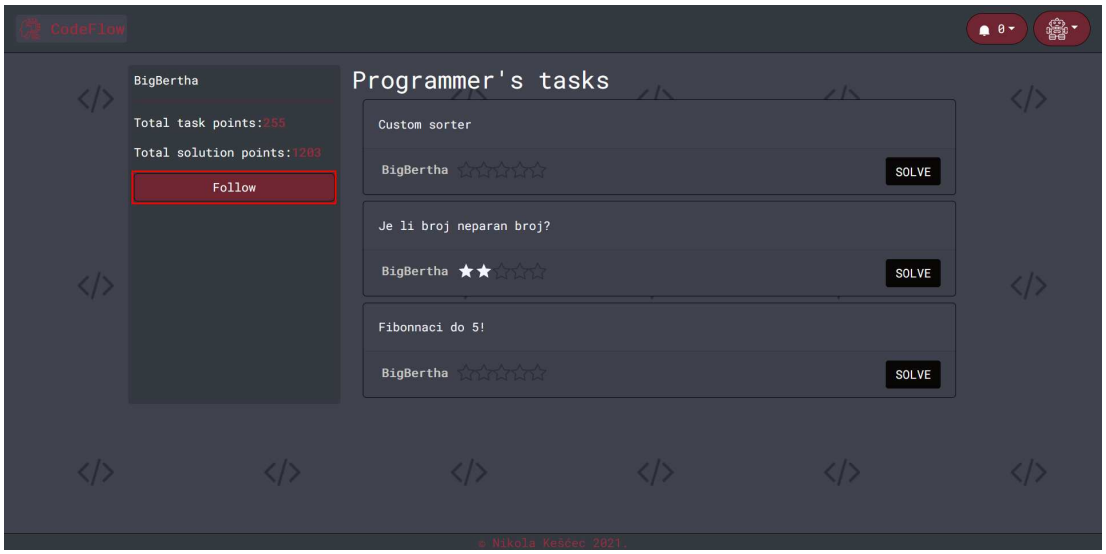
Slika 7.3: Prikaz stranice s mogućnošću pregleda i filtriranja ponuđenih sažetaka zadataka te naznačenim imenom korisnika koje vodi do korisničke stranice tog korisnika.

7.5. Praćenje korisnika

Korisnici Codeflow aplikacije imaju mogućnost praćenja drugih korisnika. Zahtjev za praćenjem može se predati na dva načina: direktnim pritiskom na gumb *Follow* unutar rang tablice korisnika (kao što je označeno na slici 7.4) ili direktnim odlaskom na korisničku stranicu određenog korisnika (istim mehanizmom kao i kod pregleda sažetaka zadataka pojedinog korisnika). Na korisničkoj stranici pritiskom gumba *Follow* ostvaruje se ista funkcionalnost kao i u prvom navedenom načinu. Klikom na gumb Follow samo se šalje zahtjev za pratiteljstvo koje onda korisnik kojem je usmjereno pratiteljstvo mora odobriti ili odbaciti.



Slika 7.4: Prikaz stranice s prvim naglašenim načinom praćenja korisnika.

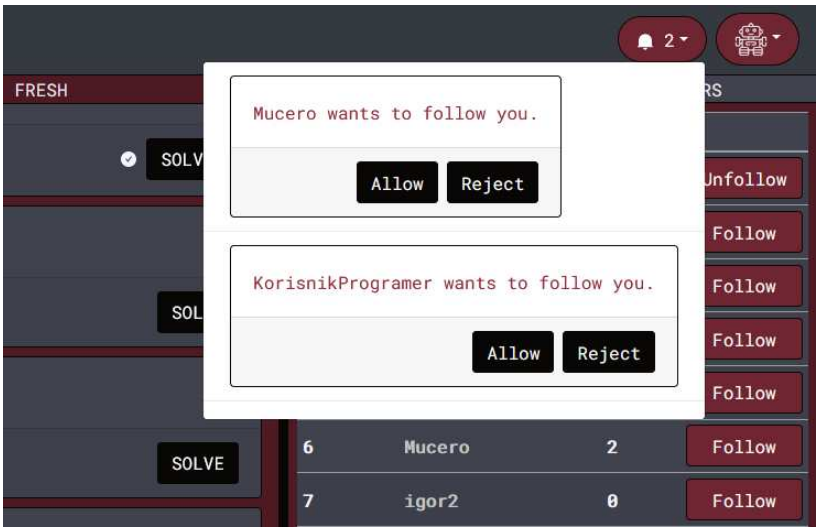


Slika 7.5: Prikaz stranice drugog korisnika s naglašenim gumbom za praćenje.

7.6. Prihvaćanje ili odbijanje zahtjeva za pratiteljstvo

Zahtjev za prijateljstvo korisnici mogu prihvatiti ili odbiti. Svaki od zahtjeva formiran je u obliku obavijesti. Sve obavijesti korisnici u bilo kojem trenutku korištenja Codeflow aplikacije (ako postoje) mogu pregledati klikom na *dropdown* gumb označen na slici 7.6. Pritiskom na taj gumb sve dostupne obavijesti korisnicima su prikazane te korisnici unutar obavijesti koje sadržavaju zahtjeve za pratiteljstvo imaju pravo navedeni zahtjev prihvatiti ili odbiti.

Slika 7.6: Prikaz izgleda *dropdown* gumba s obavijestima.

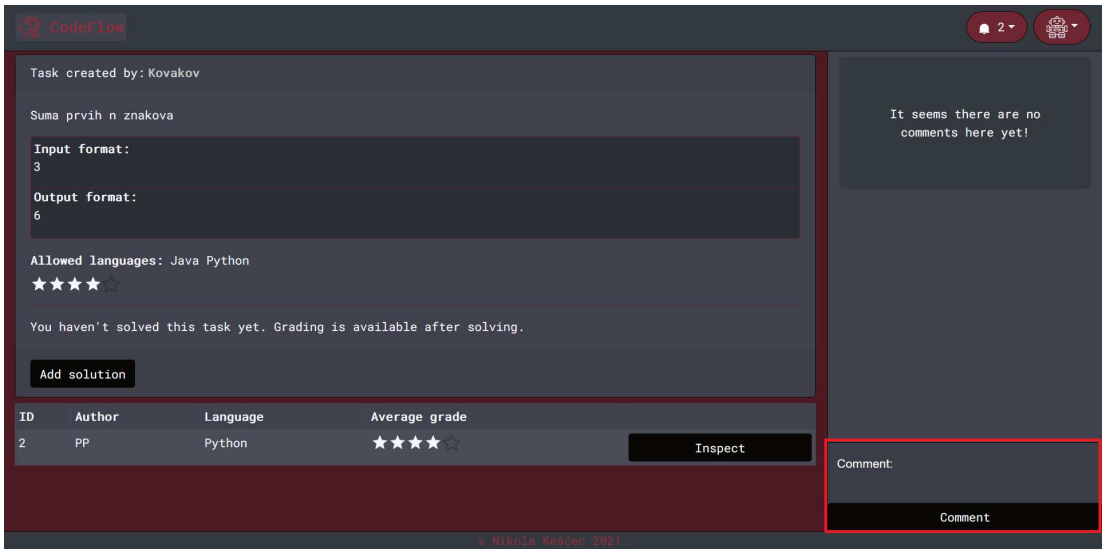


Slika 7.7: Prikaz obavijesti koje sadržavaju zahtjeve za pratiteljstvo.

7.7. Detaljan pregled zadatka (uz opcionalno komentiranje i ocjenjivanje)

Detaljan pregled zadatka korisnici mogu obaviti pritiskom gumbiju *SOLVE* (ako nije već riješio), *SOLVED* (ako je već riješio) ili *INSPECT* (ako je zadatak njegov). Gumb koji vodi na detaljan pregled zadatka nalazi se na sažetku zadatka spomenutog tijekom objašnjavanja pregleda sažetaka zadataka (7.4). Stranica koja prikazuje detaljan pregled zadatka sadržava ime autora, opis zadatka, primjer upisa i ispisa, dopuštene

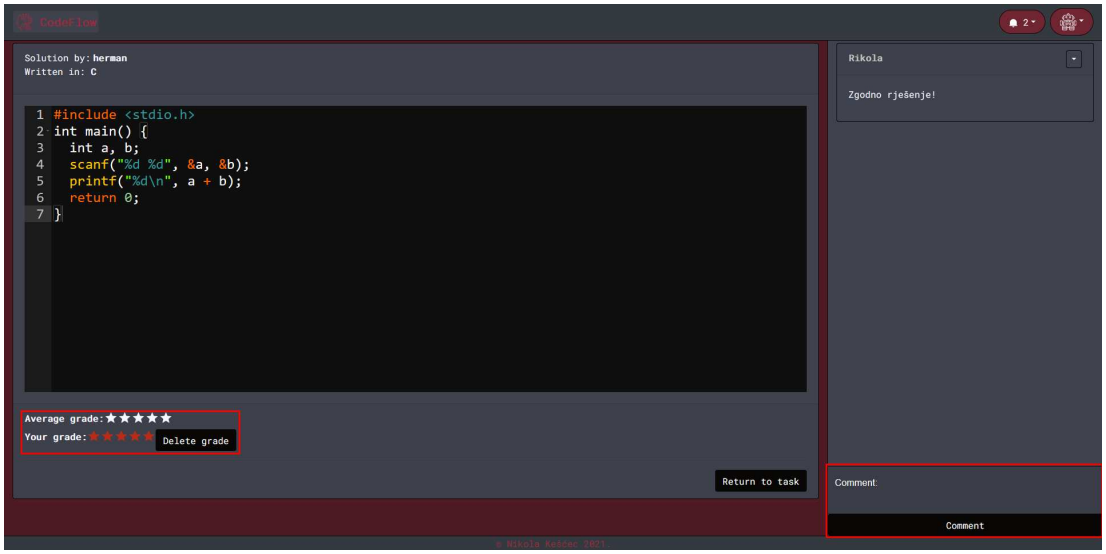
programske jezike rješenja zadataka te prosječnu ocjenu. Opcionalno, korisnici mogu komentirati zadatak pisanjem komentara unutar polja za pisanje komentara (naznačeno na slici 7.8). Ocjenjivanje zadatka korisnicima je moguće tek nakon rješavanja zadatka.



Slika 7.8: Prikaz detaljnog pregleda zadatka s naglašenim prostorom za unos opcionalnog komentara.

7.8. Pregled rješenja zadatka (uz opcionalno komentiranje i ocjenjivanje)

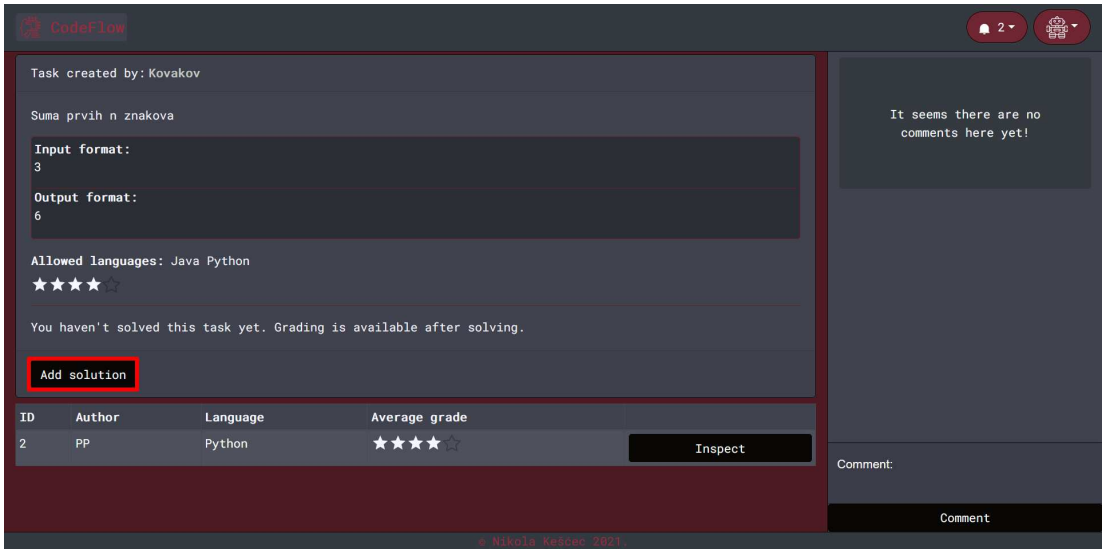
Detaljan pregled rješenja zadatka korisnici mogu obaviti pritiskom gumba *Inspect* unutar tablice rješenja koja se nalazi na detaljnom pregledu zadatka. Stranica koja prikazuje detaljan pregled rješenja sadržava ime autora, programski jezik rješenja, programski kod rješenja te prosječnu ocjenu. Opcionalno, korisnici mogu komentirati rješenje pisanjem komentara unutar polja za pisanje komentara (postupak je identičan kao i kod komentiranja zadatka). Također, korisnici uz opcionalno komentiranje rješenja mogu i ocijeniti rješenje ocjenom od 1 do 5.



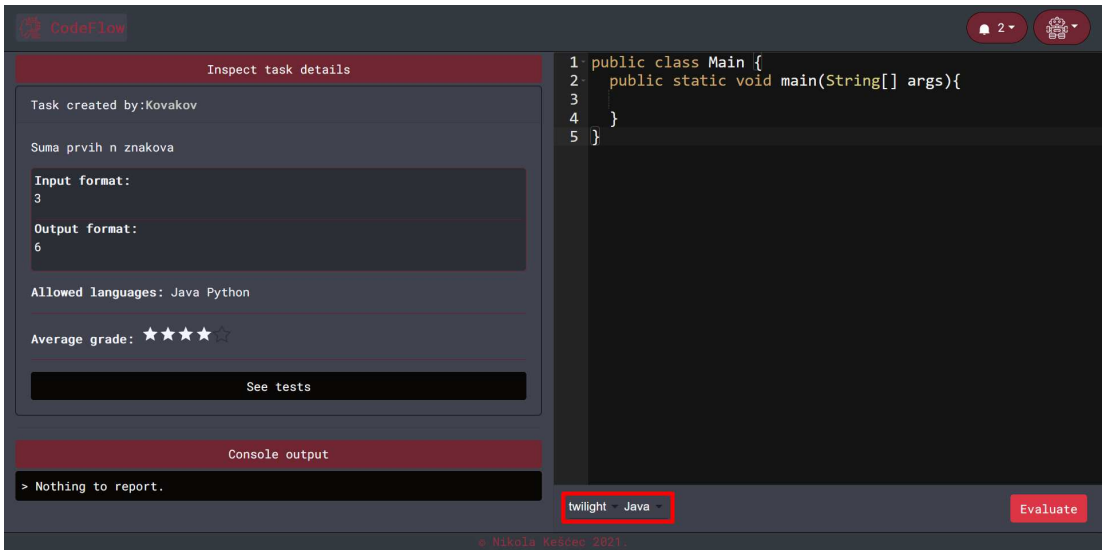
Slika 7.9: Prikaz pregleda rješenja zadatka uz naglašene elemente opcionalnog ocjenjivanja te komentiranja.

7.9. Rješavanje odabranog zadatka

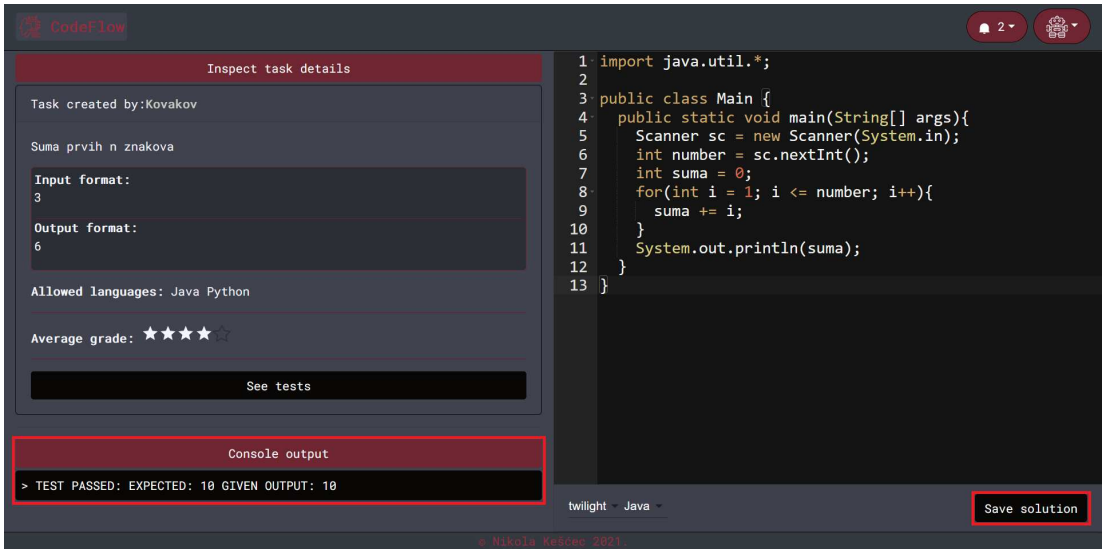
Tijekom detaljnog pregleda zadatka korisnici imaju mogućnost riješiti zadatak pritiskom na gumb *Add solution*. U slučaju da je korisnik već riješio zadatak gumb *Inspect my solution* bit će prikazan umjesto *Add solution* gumba te će taj gumb korisnika voditi na stranicu pregleda njegovog rješenja. Pošto korisnici pritisnu gumb *Add solution* bit će odvedeni na stranicu koja sadržava detalje zadatka, izlaz konzole i uređivač programskog koda. Detalji zadatka navedeni su istim redom kao i na detaljnom prikazu zadatka uz nadodanu mogućnost pregledavanja testova. Tijekom rješavanja zadatka korisnici mogu mijenjati temu uređivača programskog koda te programski jezik u kojem pišu rješenje (svaki zadatak definira dopuštene programske jezike). Pošto korisnici napišu rješenje, pritisak na *Evaluate* gumb prvo će rješenje poslati na provjeru Judge0 izvršitelju programskog koda, a tek nakon uspješne usporedbe izlaza programskog rješenja s testovima zadatka bit će omogućeno spremanje rješenja zadatka pritiskom na gumb *Save solution*. Nakon spremanja rješenja korisnici će biti preusmjereni na pregled njihovog rješenja zadatka.



Slika 7.10: Prikaz detaljnog prikaza zadatka s naglašenim gumbom za dodavanje zadatka.



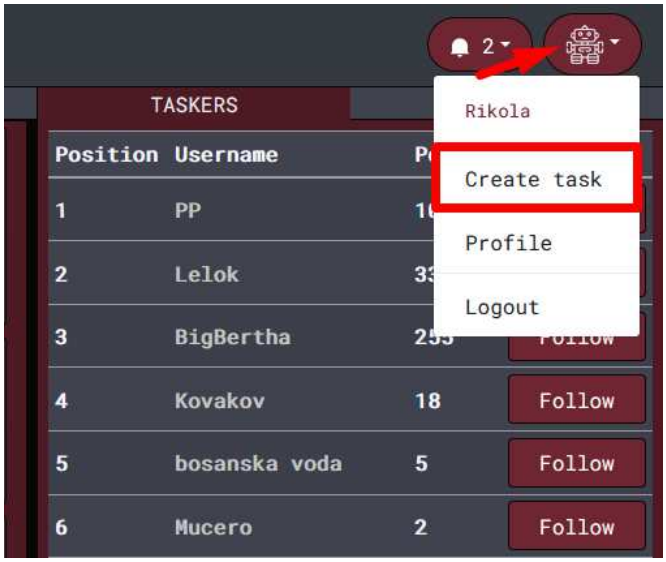
Slika 7.11: Prikaz stranice koja sadržava uređivač programskog koda s naglašenim opcijama za promjenu teme i programskog jezika.



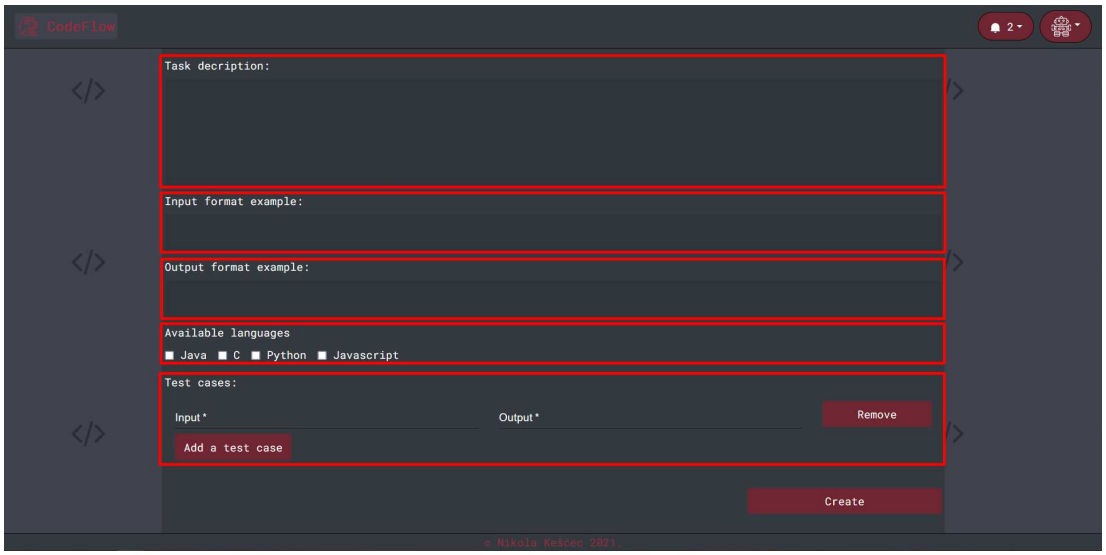
Slika 7.12: Prikaz stranice koja sadržava uređivač programskog koda s evaluiranim rješenjem spremnim za pohranu. Naglašeni su rezultati evaluacije rješenja te gumb za spremanje rješenja.

7.10. Stvaranje zadatka

Uz pregledavanje zadataka i rješenja te pisanja rješenja za odabrani zadatak, korisnici imaju pravo u bilo kojem trenutku napraviti zadatak. Poveznica koja vodi do stranice koja sadržava formu za stvaranje zadatka nalazi se unutar *dropdown* gumba naglašenom na slici 7.13. Pritiskom na nju korisnici su odvedeni na stranicu na kojoj mogu definirati svoj zadatak. Stranica sadržava formu koja deklarira potrebne podatke za stvaranje programskog zadatka. Ti podaci redom su: opis zadatka, primjer ulaza i izlaza, dopušteni jezici te lista testnih slučajeva koji zadržavaju par ulaznih i izlaznih vrijednosti (naglašeni na slici 7.14). Svaki od traženih podataka obavezan je te pojedinačno definiraju dodatne uvjete. Pojedinačni uvjeti su da opis zadatka ne smije biti kraći od 25 riječi, bar jedan programski jezik mora biti odabran te bar jedan testni zadatak mora biti definiran s popunjenim parom ulaznih i izlaznih vrijednosti. Tek nakon što korisnici zadovolje tražene uvjete mogu stvoriti novi zadatak.



Slika 7.13: Prikaz poveznice koja vodi na stranicu za stvaranje zadatka. Prikazivanje poveznice ostvaruje se pritiskom na označeni gumb.



Slika 7.14: Prikaz forme za stvaranje zadatka s označenim traženim vrijednostima.

8. Budući razvoj

Codeflow aplikacija sadržava potencijal za dodavanje funkcionalnosti i unaprjeđenje postojećih. Dizajnirana je na način da dodavanje novih funkcionalnosti ne mijenja funkcionalnost već ostvarenih funkcionalnosti te je time otvorena za nadogradnju. Dakle, budući razvitak podijeljen je na kategoriju **nadogradnji** i **unaprjeđenja**.

Nadogradnje

Prva preporučena nadogradnja aplikacija bilo bi uvođenje sistema "igrifikacije" (spomenuto unutar potpoglavlja 3.3). Uvođenje znački, kategorija i povremenih natjecanja pozitivno bi utjecalo na sveukupno korisničko iskustvo.
Nedostatak trenutačnog sustava komentiranja i obavijesti u usporedbi s ostalim sličnim stranicama jest potreba za promjenom stranice ili, u najgorem slučaju, ponovnim učitavanjem aplikacije radi učitavanja promjena unutar komentara ili obavijesti. Ovaj nedostatak rješiv je korištenjem **WebSocket** protokola. WebSocket[16] protokol omogućuje potpuno dvosmjernu komunikaciju između klijenta i poslužitelja u stvarnom vremenu. Makar različit od tipičnog HTTP protokola, dizajniran je da radi na istim vratima (engl. *ports*) (443 i 80) kao i HTTP protokol, što ga čini veoma prihvatljivim za normalnu internetsku interakciju između klijenta i poslužitelja. Podržan je na svim modernim internetskim preglednicima i njegovo bi svojstvo dvosmjerne komunikacije omogućilo internetskom pregledniku da ažuriranje komentara i obavijesti istovremeno dojavi i React aplikaciji na prezentacijskom sloju. React aplikacija bi potom ažurirala svoje korisničko sučelje da ispravno prati stanje podataka unutar podatkovnog sloja. Takav sistem bio bi prigodniji trenutačnom te bi kao nadogradnja potpuno zamijenio trenutačni.

Unaprjeđenja

Najvažnije unaprjeđenje bilo bi dodatna prilagodba sustava evaluacije korisničkih rješenja. Sustav u trenutnom stanju šalje rješenje pojedinačno za svaki testni slučaj tije-

kom evaluacije, što nije održivo za veća rješenja i veći broj testnih slučajeva. Unaprjeđenje bi sustav prilagodilo da umjesto pojedinačnog slanja pripremi takozvanu skupnu (engl. *batched*) pošiljku i obavi sve evaluacije u okviru jednog zahtjeva.
Unaprjeđenje sustava spremanja rješenja koje bi bilo pogodno implementirati jest provjera plagijata tijekom dodavanja novog rješenja. Trenutačna verzija aplikacije ne provjerava sličnost rješenja te korisnici mogu veoma jednostavno u potpunosti kopirati već postojeća rješenja drugih korisnika. Sustav spremanja također ne pruža mogućnost spremanja nedovršenog rješenja programskog zadatka. Dodavanje te mogućnosti uveliko bi poboljšalo korisničko iskustvo te je još jedno od poželjnih unaprjeđenja.

9. Zaključak

Velik porast zanimanja za programiranjem te sve veća potreba za održavanjem programerskog znanja rezultirala je nastankom stranica koje omogućuju vježbanje i razvijanje programerskih vještina. Neke od stranica usvojile su elemente društvenih mreža, poput dodavanja mogućnosti raspravljanja i komentiranja zadataka i rješenja, a sve češća je i implementacija sustava "igrifikacije". Doduše, učestali zajednički čimbenik navedenih stranica jest nemogućnost da prosječni korisnik zada zadatak. Ipak, zbog podržanih funkcionalnosti rješavanja zadataka, određenih socijalnih elemenata i implementacije navedenih sustava "igrifikacije" popularnost tih stranica visoka je.

Tema ovog završnog rada bila je razvijanje i implementiranje aplikacije koja podržava učestale značajke popularnih stranica koje omogućavaju rješavanje programskih zadataka. Dodatne funkcionalnosti ove aplikacije su podržavanje najbitnijih elemenata socijalnih mreža i mogućnost zadavanja zadataka od strane bilo kojeg korisnika.

Izazovan element izrade aplikacije koja zadovoljava spomenute zahtjeve bio je organizacija i strukturiranje sustava za stvaranje zadataka i pripremanje okvirnih predložaka za rješenja stvorenih zadataka. Jednako je izazovan element razvoja aplikacije ovoga specifičnoga tipa bio podržavanje provjeravanja ispravnosti korisničkog programskog koda. Ipak, uz pomoć Judge0 izvršitelja programskog koda i ta funkcionalnost uspješno je podržana.

Odabir Spring radnog okvira i React JavaScript biblioteke bio je kvalitetan jer obje spomenute tehnologije posjeduju bogatu podršku te veoma kvalitetnu dokumentaciju. Također, spomenute tehnologije kvalitetno podržavaju izradu višeslojnih internetskih aplikacija te su uz kombinaciju PostgreSQL sustava za upravljanje bazama podataka potpuno zadovoljile tehnološke potrebe ovog završnog rada.

LITERATURA

[1] *Ace*. Ace, 10.6.2021. URL https://ace.c9.io/#nav=about.

[2] *Maven*. Apache, 9.6.2021. URL https://maven.apache.org/.

[3] *JWT*. Auth0, 10.6.2021. URL https://jwt.io/.

[4] *Bootstrap, Getting Started*. Bootstrap Team, 10.6.2021. URL https://getbootstrap.com/docs/5.0/getting-started/introduction/.

[5] *Axios*. Community Project, 10.6.2021. URL https://axios-http.com/docs/intro.

[6] *JavaScript*. ECMAScript, 10.6.2021. URL https://www.javascript.com/.

[7] *React, Getting Started*. Facebook, 10.6.2021. URL https://reactjs.org/docs/getting-started.html.

[8] *Edgar*. Fakultet elektrotehnike i računarstva, 10.6.2021. URL https://edgar.fer.hr/.

[9] *HackerRank*. HackerRank, 10.6.2021. URL https://www.hackerrank.com/.

[10] *Judge0*. Herman-Zvonimir Došilović, 10.6.2021. URL https://judge0.com/.

[11] *Git*. Junio Hamano and others, 10.6.2021. URL https://axios-http.com/docs/intro.

[12] *Codewars*. LeetCode, 10.6.2021. URL https://leetcode.com/.

[13] *Leetcode*. LeetCode, 10.6.2021. URL https://leetcode.com/.

[14] *jjwt*. Les Hazlewood, 10.6.2021. URL https://github.com/jwtk/jjwt.

[15] *Material-UI, Getting Started*. Material-UI, 10.6.2021. URL https://material-ui.com/getting-started/installation/.

[16] *WebSocket*. Mozilla, 10.6.2021. URL https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API.

[17] *Single Page Application*. Mozzila, 9.6.2021. URL https://developer.mozilla.org/en-US/docs/Glossary/SPA.

[18] *NPM*. npm, Inc., 10.6.2021. URL https://www.npmjs.com/.

[19] *Node.js*. OpenJS Foundation, 10.6.2021. URL https://nodejs.org/en/.

[20] *Java*. Oracle, 9.6.2021. URL https://www.oracle.com/java/technologies/.

[21] *React Router*. ReactTraining, 10.6.2021. URL https://reactrouter.com/web/guides/quick-start.

[22] *Hibernate*. Red Hat, 10.6.2021. URL https://hibernate.org/.

[23] *Representational state transfer (REST)*. restfulapi.net, 10.6.2021. URL https://restfulapi.net/.

[24] *Spring Framework*. Spring, 7.6.2021. URL https://spring.io/projects/spring-framework.

[25] *Spring Security*. Spring, 8.6.2021. URL https://spring.io/projects/spring-security.

[26] *Spring Data JPA*. Spring, 9.6.2021. URL https://spring.io/projects/spring-data-jpa.

[27] *Node.js*. The Chromium Project, 10.6.2021. URL https://v8.dev/.

[28] *PostgreSql*. The PostgreSQL Global Development Group, 10.6.2021. URL https://www.postgresql.org/.

Web-aplikacija za rješavanje programskih zadataka s elementima društvene mreže

Sažetak

Završni rad razmatra izradu web-aplikacije za rješavanje programskih zadataka s elementima društvene mreže. Funkcionalnosti razvijene web-aplikacije dobivene su promatranjem čestih funkcionalnosti već postojećih programskih rješenja te dodavanjem temeljnih funkcionalnosti društvene mreže. Korisnici razvijene web-aplikacije mogu zadavati i rješavati programske zadatke, a provjera rješenja ostvarena je korištenjem Judge0 izvršitelja programskog koda. Tijekom zadavanja zadataka korisnici biraju programske jezike koji će biti dostupni tijekom rješavanja te navode testne slučaje koji određuju ispravnost rješenja. Podržane funkcionalnosti društvene mreže unutar web-aplikacije su praćenje korisnika, ocjenjivanje i komentiranje sadržaja te pregledavanje korisničkih profila drugih korisnika. Razvijena web-aplikacija sastoji se od React internetske aplikacije, Spring internetskog poslužitelja i PostgreSQL baze podataka.

Ključne riječi: React, SPA, Spring, REST, JWT, PostgreSQL, zadavanje programskih zadataka, Judge0.

Web application for solving programming tasks with social network elements

Abstract

Final work considers the development of web-application for solving programming tasks with social network elements. Functionalities of developed web-application have been obtained by observation of functionalities of already existing programming solutions and additional adding of fundamental social network functionalities. Users of developed web-application can assign and solve programming tasks. Solution evaluation is supported with usage of Judge0 code execution engine. While assigning programming task Users choose programming languages that will be available for usage in task solving and they also state test cases that will be used to determine solution correctness. Supported social network functionalities of developed web-application are user following, content grading and commenting and user profile page examination. Developed web-application is consisted of React web-application, Spring web server and PostgreSQL database.

Keywords: React, SPA, Spring, REST, JWT, PostgreSQL, programing problems assingment, Judge0.