Obrazovna interaktivna aplikacija za poučavanje podržanog učenja

Ivančić, Hana

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 1618

OBRAZOVNA INTERAKTIVNA APLIKACIJA ZA POUČAVANJE PODRŽANOG UČENJA

Hana Ivančić

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 1618

OBRAZOVNA INTERAKTIVNA APLIKACIJA ZA POUČAVANJE PODRŽANOG UČENJA

Hana Ivančić

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 4. ožujka 2024.

ZAVRŠNI ZADATAK br. 1618

Pristupnica:	**Hana Ivančić (0036538570)**
Studij:	Elektrotehnika i informacijska tehnologija i Računarstvo
Modul:	Računarstvo
Mentor:	doc. dr. sc. Tomislav Jagušt

Zadatak: **Obrazovna interaktivna aplikacija za poučavanje podržanog učenja**

Opis zadatka:

Podržano učenje (eng. reinforcement learning) je grana strojnog učenja u kojoj agent uči optimalno ponašanje u određenoj okolini putem iskustva, iterativno izvršavajući akcije radi maksimizacije nagrade. U sklopu ovog završnog rada potrebno je osmisliti i izraditi obrazovnu aplikaciju koja će učenicima pomoći u razumijevanju podržanog učenja, na primjeru robota koji pronalaze put iz zadanog labirinta. Potrebno je istražiti postojeće alate i metode poučavanja podržanog učenja te analizirati mogućnosti njihove primjene u sklopu razvijene obrazovne aplikacije. Nadalje, potrebno je osmisliti način na koji se teorija podržanog učenja može prenijeti u interaktivno iskustvo kroz koje će se istrenirati agent i zorno prikazati dobiveni rezultati. Konačno, potrebno je ostvariti interaktivnu aplikaciju koja omogućava treniranje agenta, varijaciju parametara i analizu dobivenih rezultata.

Rok za predaju rada: 14. lipnja 2024.

Zahvaljujem doc. dr. sc. Tomislavu Jaguštu na nesebičnoj podršci, organizaciji, inovativnosti, spremnosti na prihvaćanje mojih ideja, suradnji u svakom trenutku i razumijevanju za izazovan i dinamičan proces razvoja ovog rada.

Sadržaj

1. Uvod

Podržano učenje (engl. Reinforcement Learning) je ključno područje strojnog učenja koje omogućava agentima učenje optimalnog ponašanja kroz interakciju s okolinom. Podržano učenje temelji se na principu da agent dobiva povratne informacije iz okoline putem nagrada i kazni, ovisno o izvršenim akcijama. Kroz proces eksperimentiranja i iterativnog učenja, agent postupno optimizira svoje ponašanje kako bi maksimizirao ukupnu nagradu.

Ova metoda učenja primjenjuje se u raznim područjima, uključujući robotiku, igre, autonomna vozila i sustave za preporuke. Na primjer, u robotici, agent može naučiti kako se kretati kroz nepoznati prostor izbjegavajući prepreke, dok u sustavima za preporuke može naučiti predlagati korisnicima proizvode na temelju njihovih prethodnih izbora.

Cilj ovog završnog rada je izraditi obrazovnu aplikaciju koja će olakšati razumijevanje podržanog učenja učenicima kroz praktičan i interaktivan pristup. Tradicionalni pristupi učenju podržanog učenja često se oslanjaju na teorijske koncepte i matematičke modele koji mogu biti apstraktni i teški za razumijevanje. Međutim, pružanjem interaktivne platforme, učenici mogu vizualizirati kako agent uči iz svojih iskustava i prilagođava svoje akcije kako bi postigao bolje rezultate.

Korištenjem modernih tehnologija kao što su React, Spring i PostgreSQL, razvijena je aplikacija koja omogućava treniranje agenta u labirintu, varijaciju parametara i analizu dobivenih rezultata. U aplikaciji, korisnici mogu postaviti različite parametre simulacije, kao što su kazne za pogrešne korake i nagrade za postizanje cilja, kako bi vidjeli kako te promjene utječu na ponašanje agenta. Simulacija također uključuje vizualne prikaze napretka agenta kroz labirint, što korisnicima omogućava da jasno

vide proces učenja i prilagodbe. Analiza dobivenih rezultata omogućava korisnicima da kvantificiraju učinkovitost različitih strategija učenja i bolje razumiju kako različiti faktori utječu na proces učenja.

Ovaj interaktivni pristup ne samo da olakšava razumijevanje teorijskih koncepata podržanog učenja, već također potiče aktivno učenje i angažman korisnika. Učenici mogu eksperimentirati s različitim scenarijima i promatrati izravne posljedice svojih odluka, što doprinosi dubljem razumijevanju i primjeni podržanog učenja u stvarnim situacijama.

2. Podržano učenje

Podržano učenje je metoda strojnog učenja u kojoj agent uči optimalno ponašanje u određenoj okolini putem pokušaja i pogrešaka. Agent prima povratne informacije iz okoline putem nagrada i kazni, ovisno o izvršenim akcijama. Kroz iterativni proces, agent postupno uči koje akcije vode do najboljih rezultata.

2.1. Osnovni pojmovi podržanog učenja

Podržano učenje temelji se na nekoliko ključnih komponenti:

- **Agent**: Entitet koji donosi odluke i izvršava akcije u određenoj okolini.
- **Okolina (Environment)**: Svijet u kojem agent djeluje i iz kojeg prima povratne informacije.
- **Stanja (States)**: Različite situacije u kojima se agent može nalaziti unutar okoline.
- **Akcije (Actions)**: Skup mogućih odluka koje agent može donijeti u bilo kojem stanju.
- **Nagrada (Reward)**: Povratna informacija koju agent dobiva nakon izvršene akcije, a koja mu pomaže u procjeni uspješnosti te akcije.

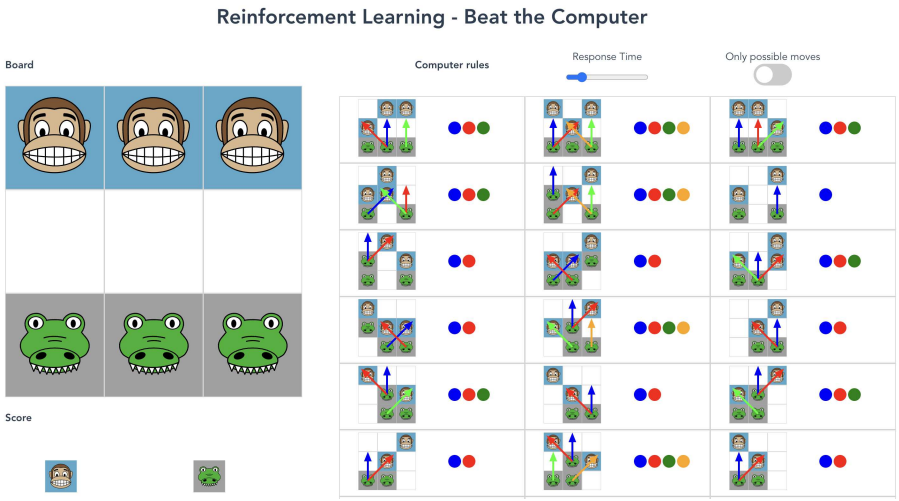
Glavni cilj agenta je naučiti politiku (policy) koja maksimizira ukupnu nagradu tijekom vremena. Politika definira pravila prema kojima agent odabire akcije u različitim stanjima.

2.2. Primjeri poučavanja podržanog učenja

Postoji nekoliko metoda kojima se podržano učenje može približiti učenicima na jednostavan i razumljiv način:

Kutije šibica za igru križić-kružić [16]: Jedan od kreativnih načina poučavanja podržanog učenja je kroz korištenje kutija šibica u igri križić-kružić. Svaka kutija šibica predstavlja jedno stanje na ploči za igru. Unutar svake kutije nalaze se šibice koje predstavljaju moguće poteze. Kada agent (igrač) napravi potez, dobiva povratnu informaciju o tome je li potez bio dobar ili loš. Na temelju ove povratne informacije, šibice koje predstavljaju dobre poteze se nagrađuju dodavanjem više šibica, dok se loši potezi kažnjavaju uklanjanjem šibica. Kroz ponavljane igre, agent uči koje poteze treba odabrati kako bi pobijedio u igri tako što se povećava broj šibica koje predstavljaju dobre poteze, a šibice koje predstavljaju loše poteze s vremenom sve nestanu.

Schlag das Krokodil [17]: Ova metoda koristi jednostavnu igru u kojoj učenici uče kako agenti koriste podržano učenje za pobjedu nad krokodilom u igri. Učenici preuzimaju ulogu agenta koji pokušava pobijediti krokodila kroz niz pokušaja i pogrešaka, primajući nagrade za uspješne poteze i kazne za neuspješne. Kroz ovu igru, učenici uče osnovne principe podržanog učenja na zabavan i interaktivan način. Igra se sastoji od pomicanja figura na ploči i donošenja odluka na temelju povratnih informacija.



Slika 2.1. Primjer igre "Schlag das Krokodil"

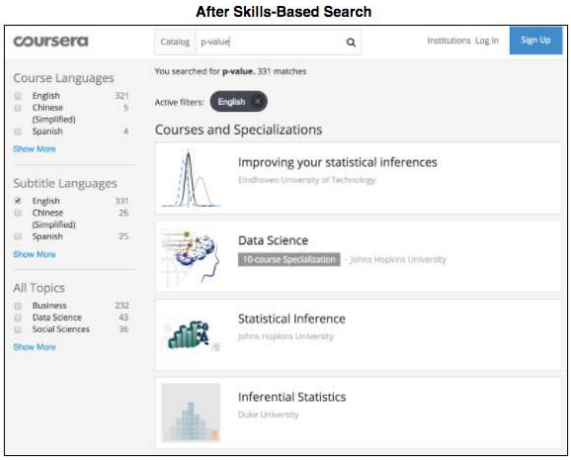
Korištenjem ovih metoda, učenici mogu steći dublje razumijevanje kako agenti uče i prilagođavaju se svojoj okolini. Praktični primjeri i aktivnosti pomažu u približavanju složenih koncepata podržanog učenja i omogućuju učenicima da primijene teorijska znanja u stvarnim situacijama.

3. Slične obrazovne aplikacije

U današnje vrijeme, podržano učenje sve se više koristi u raznim obrazovnim aplikacijama koje omogućuju učenicima interaktivno i praktično učenje kroz simulacije i igranje uloga. Ove aplikacije koriste algoritme strojnog učenja za prilagodbu sadržaja na temelju interakcija korisnika, čime se povećava učinkovitost učenja. U nastavku su izdvojene neke od najznačajnijih aplikacija koje koriste podržano učenje za obrazovne svrhe.

3.1. Coursera

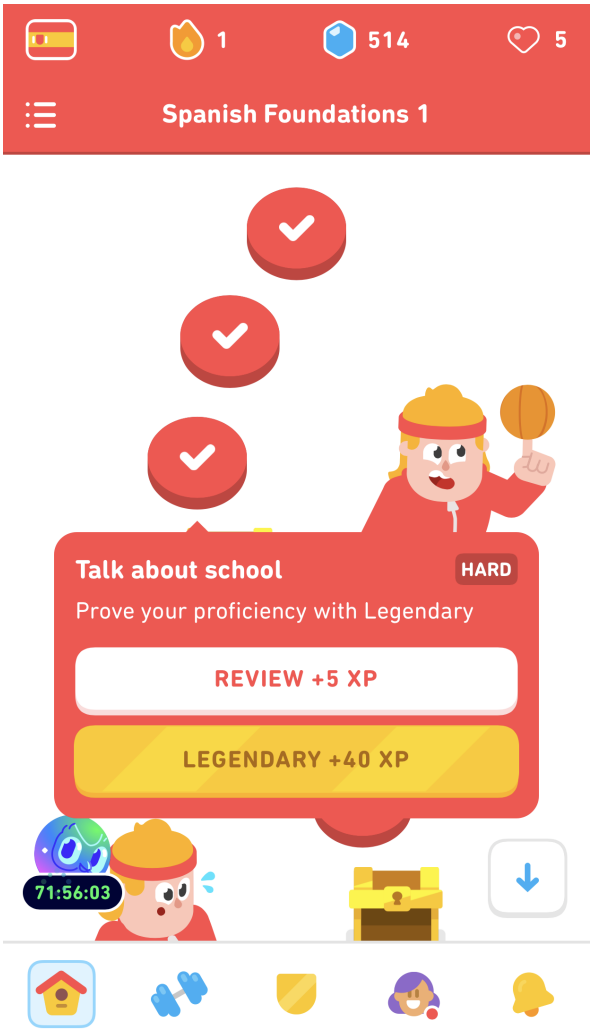
Coursera nudi online tečajeve na različitim razinama obrazovanja, uključujući podržano učenje. Platforma koristi napredne algoritme za analizu interakcija korisnika s materijalima i prilagođava tečajeve kako bi odgovarali njihovim potrebama. Coursera također omogućuje učenicima da prate svoj napredak i postignuća kroz interaktivne zadatke i kvizove.



Slika 3.1. Aplikacija Coursera

3.2. Duolingo

Duolingo je jedna od najpopularnijih aplikacija za učenje jezika koja koristi elemente podržanog učenja kako bi prilagodila tečajeve i zadatke prema individualnom napretku korisnika. Korisnici započinju s jednostavnim zadacima i, kako napreduju, zadaci postaju složeniji. Algoritmi analiziraju odgovore korisnika i prilagođavaju težinu i vrstu zadataka kako bi osigurali optimalan napredak u učenju. Duolingo također koristi igrifikaciju kako bi zadržao motivaciju korisnika kroz nagrade i bodove.

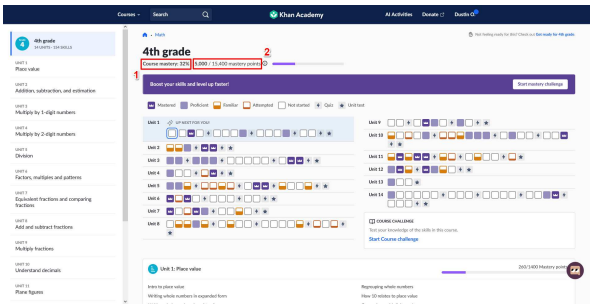


Slika 3.2. Aplikacija Duolingo

3.3. Khan Academy

Khan Academy pruža širok spektar obrazovnih materijala i koristi algoritme podržanog učenja za personalizaciju iskustva učenja. Aplikacija prati napredak učenika i prilagođava

težinu zadataka kako bi se osiguralo da učenici dobiju odgovarajuće izazove. Khan Academy također nudi povratne informacije u stvarnom vremenu, pomažući učenicima da isprave greške i razumiju koncepte na dublji način.



Slika 3.3. Aplikacija Khan Academy

3.4. Mogućnosti nadogradnje aplikacije

Iako ova aplikacija za poučavanje podržanog učenja već pruža korisnicima vrijedne alate i resurse, postoji nekoliko mogućnosti za daljnje poboljšanje temeljem analize postojećih sličnih obrazovnih aplikacija.

- **Igrifikacija (Gamification) kao u Duolingu**: Uvođenjem elemenata igrifikacije, kao što su bodovi, značke i nagrade za postignuća, možemo povećati motivaciju i angažman korisnika. Ova aplikacija bi mogla uključivati izazove, razine težine i virtualne nagrade koje bi učenici mogli osvojiti za uspješno završavanje zadataka.
- **Personalizirani sadržaj kao u Khan Academy**: Koristeći napredne algoritme za praćenje napretka korisnika, ova aplikacija bi mogla prilagoditi težinu i vrstu zadataka prema individualnim potrebama svakog učenika. To bi uključivalo analizu rezultata korisnika i prilagodbu sadržaja kako bi se osiguralo da svaki učenik dobije odgovarajuće izazove.
- **Interaktivni zadaci i kvizovi kao u Courseri**: Uvođenje interaktivnih zadataka i kvizova koji se prilagođavaju prema napretku korisnika može dodatno poboljšati iskustvo učenja. Kvizovi mogu uključivati različite formate pitanja, od višestrukih izbora do interaktivnih simulacija, što bi korisnicima omogućilo bolje razumijevanje i primjenu teorijskih znanja.

Integriranjem ovih elemenata u ovu aplikaciju, mogli bismo značajno poboljšati

iskustvo korisnika, povećati angažman i motivaciju te osigurati bolje rezultate u učenju podržanog učenja. Ove nadogradnje bi omogućile korisnicima da dobiju sveobuhvatno i prilagođeno iskustvo učenja, slično onome što nude vodeće obrazovne platforme danas.

4. Korištene tehnologije i alati

Aplikacija je izrađena koristeći sljedeće tehnologije:

- **React** - za frontend aplikacije
- **Spring** - za backend aplikacije
- **PostgreSQL** - za bazu podataka

4.1. React

React je popularna JavaScript biblioteka za izradu korisničkih sučelja. Omogućava izradu dinamičnih i responzivnih web aplikacija. U ovoj aplikaciji React je korišten za izradu komponenti sučelja koje omogućuju interakciju s korisnicima.

React omogućuje izgradnju složenih korisničkih sučelja iz manjih, izoliranih dijelova zvanih komponente. Svaka komponenta ima svoj životni ciklus koji uključuje faze kao što su kreiranje, ažuriranje i uništavanje. React koristi virtualni DOM, što omogućuje brže i efikasnije ažuriranje sučelja. Komponente se mogu ponovno koristiti, što smanjuje količinu koda i olakšava održavanje.

4.2. Spring

Spring je okvir za izradu enterprise aplikacija u Javi. Koristi se za izradu robusnih i skalabilnih backend sustava. Aplikacija koristi Spring za rukovanje poslovnom logikom, autentifikaciju korisnika i komunikaciju s bazom podataka.

Spring omogućuje razvoj složenih aplikacija koristeći principe kao što su inverzija kontrole, aspektno orijentirano programiranje i deklarativno programiranje. Pruža

infrastrukturu za kreiranje transakcija, sigurnosti, i upravljanje podacima. Spring Boot, dodatak Spring Frameworku, pojednostavljuje konfiguraciju i implementaciju aplikacija.

4.3. PostgreSQL

PostgreSQL je open-source sustav za upravljanje relacijskim bazama podataka. U aplikaciji, PostgreSQL je korišten za pohranu podataka o korisnicima, lekcijama i rezultatima.

PostgreSQL podržava složene upite, transakcije i napredne tipove podataka. Također podržava proširenja koja omogućuju dodatne funkcionalnosti, kao što su postgeospatial podaci i puni tekst pretraga. PostgreSQL osigurava ACID (Atomicity, Consistency, Isolation, Durability) svojstva, što ga čini pouzdanim rješenjem za upravljanje podacima.

4.4. Render

Render je platforma koja omogućava jednostavan deployment aplikacija i baza podataka. U aplikaciji, Render je korišten za deploy frontenda, backenda i PostgreSQL baze podataka.

Render pruža jednostavan način za deploy web aplikacija, API-ja i baza podataka s podrškom za CI/CD (Continuous Integration and Continuous Deployment). Platforma podržava automatsko skaliranje, upravljanje SSL certifikatima, nadzor aplikacija, i rollback funkcionalnosti. Render također omogućava korisnicima da lako postave svoje aplikacije koristeći Docker ili Git repozitorije.

4.5. GitHub

GitHub je platforma za verzioniranje koda i suradnju koja koristi Git za kontrolu verzija. U aplikaciji, GitHub je korišten za pohranu i upravljanje izvornih kodom za frontend, backend i infrastrukturne konfiguracije.

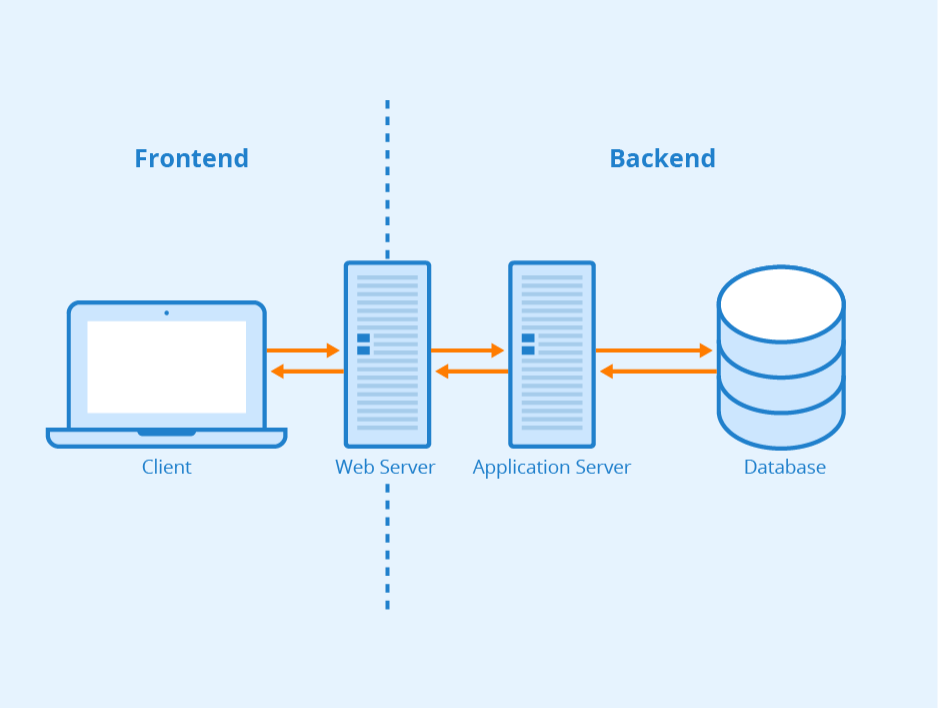
GitHub omogućava programerima da surađuju na projektima kroz alate za kontrolu

verzija, issue tracking, code review i CI/CD integracije. Neke od glavnih značajki GitHub-a uključuju:

- **Repository**: Pohrana projekata i upravljanje verzijama koda.
- **Branches**: Omogućava razvoj novih značajki i popravak grešaka u izoliranim granama.
- **Pull Requests**: Alat za pregled i spajanje promjena koda iz jedne grane u drugu.
- **Actions**: CI/CD alat za automatizaciju tijeka rada kao što su testiranje i deploy.
- **Issues**: Praćenje bugova, zadataka i novih značajki.

5. Arhitektura sustava

Sustav je podijeljen na frontend, backend i bazu podataka. Svaki dio sustava ima specifične funkcije i zaduženja.



Slika 5.1. Arhitektura sustava

5.1. Frontentd - React

Primjer React komponente (pitanja u lekcijama) prikazan je u nastavku:

```
import React, { useState, useEffect } from 'react';
import { Button, Form } from 'react-bootstrap';

function Question({ question }) {
```

```
const currentUser = JSON.parse(sessionStorage.getItem('currentUser'));
const [selectedAnswer, setSelectedAnswer] = useState(null);
const [selectedQuestion, setSelectedQuestion] = useState(null);
const [correctAnswer, setCorrectAnswer] = useState(null);
const [answeredQuestion, setAnsweredQuestion] = useState(null);

useEffect(() => {
  setCorrectAnswer(null);
  setAnsweredQuestion(null);
}, [question]);

function submitAnswer() {
  const payload = { answerId: selectedAnswer, userId: currentUser.id };
  fetch('https://mazerunnerbackend.onrender.com/answered', {
    method: 'POST',
    headers: { 'Content-Type': 'application/json' },
    body: JSON.stringify(payload),
  })
  .then(response => response.json())
  .then(() => {
   selectedQuestion.answers.map(answer => {
      if (answer.isCorrect) {
        setCorrectAnswer(answer);
        setAnsweredQuestion(selectedQuestion);
      }
    })
  })
  .catch(error => console.error('Error submitting answer:', error));
}

return (
    <div>
```

```
{question && (
    <div>
    <p>{question.question}</p>
    <Form>
        {question.answers.map((answer, aIndex) => (
        <Form.Check
            key={aIndex}
            type="radio"
            id={`answer-${aIndex}`}
            label={answer.answerText}
            checked={selectedAnswer === answer.answerId}
            onChange={() => {setSelectedAnswer(answer.answerId);
            setSelectedQuestion(question)}}
        />
        ))}
    </Form>
    {question && (
        <Button
        onClick={submitAnswer}
        variant="success"
        disabled={!selectedAnswer || correctAnswer}
        style={{ backgroundColor: '#FF028D', borderColor: '#FF028F' }}
        >
        Predaj
        </Button>
    )}
    {correctAnswer && answeredQuestion &&
    answeredQuestion.id === question.id && (
        <div>
        <p>Točan odgovor: {correctAnswer.answerText}</p>
        </div>
    )}
```

```
        </div>
        )}
    </div>
  );
}

export default Question;
```

5.2. Backend - Spring

U aplikaciji, Spring se koristi za rukovanje poslovnom logikom. U nastavku je prikazan dio koda za kontroler koji upravlja registracijom korisnika:

```
package app.controllers;

import app.dtos.UsersDTO;
import app.models.UsersModel;
import app.repositories.UsersRepository;
import app.services.UsersService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.*;

@RestController
@Controller
public class UsersController {

    @Autowired
    private final UsersService usersService;
```

```
    @Autowired
    UsersRepository repo;

    public UsersController(UsersService usersService) {
        this.usersService = usersService;
    }

    @PostMapping("/addUser")
    public void addUser(@RequestBody UsersModel user) {
        repo.save(user);
    }

    @PostMapping(path = "/register", consumes = "multipart/form-data")
    public void registerNewUser(@RequestParam("username") String username,
    @RequestParam("surname") String lastName, @RequestParam("password")
    String password, @RequestParam("name") String name) {
        usersService.registerUser(username, password, name, lastName);
    }

    @PostMapping(path = "/login", consumes = "multipart/form-data")
    @ResponseBody
    public ResponseEntity<UsersDTO> login(@RequestParam("username")
    String username, @RequestParam("password") String password) {
        UsersModel authenticatedUser = usersService.authenticate
        (username, password);
        ResponseEntity<UsersDTO> response;
        response = authenticatedUser == null ? new ResponseEntity<>
        (HttpStatus.UNAUTHORIZED) : new ResponseEntity<>
        (new UsersDTO(authenticatedUser), HttpStatus.OK);
        return response;
    }
}
```

5.3. Baza podataka - PostgreSQL

U aplikaciji, PostgreSQL se koristi za pohranu informacija o korisnicima, lekcijama i rezultatima. Struktura baze podataka omogućava učinkovito spremanje i dohvaćanje podataka potrebnih za funkcioniranje aplikacije. Baza podataka se sastoji od nekoliko ključnih tablica koje su međusobno povezane kroz relacije.

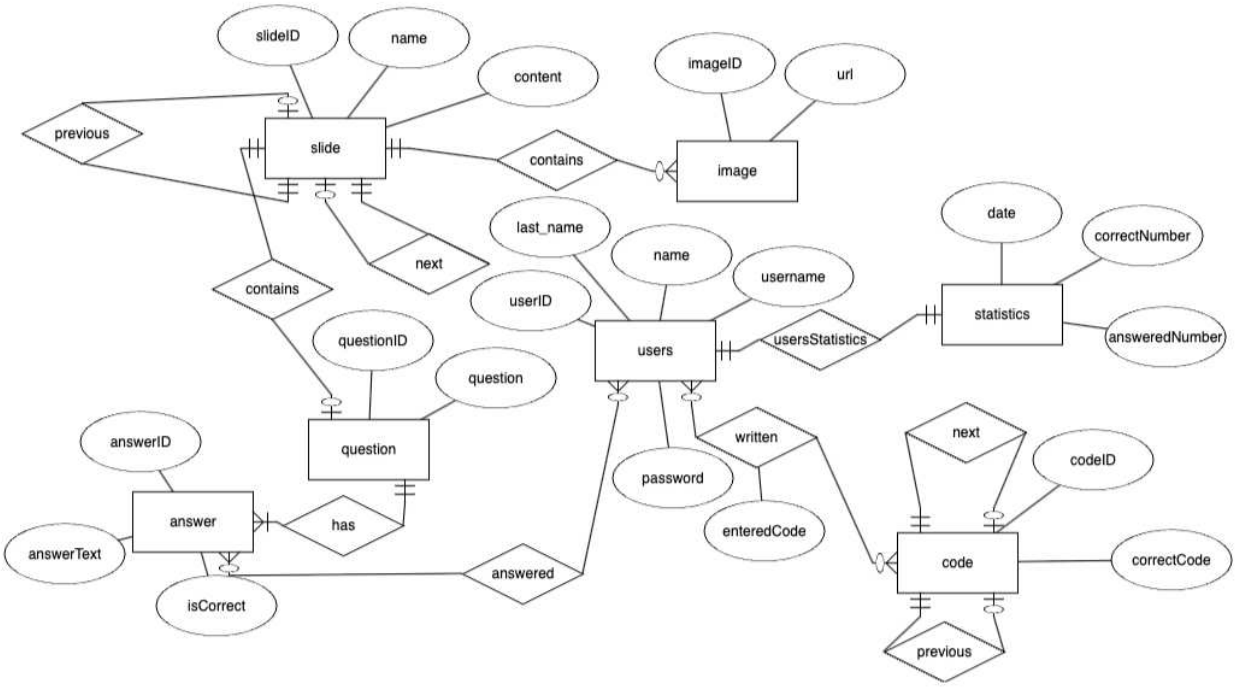
Baza podataka se sastoji od sljedećih tablica:

- **users**: Tablica koja sadrži informacije o korisnicima. Sadrži stupce *userID* (primarni ključ), *username*, *name*, *last_name* i *password*.
- **statistics**: Tablica koja pohranjuje statistiku odgovora korisnika. Sadrži stupce *id* (primarni ključ), *date*, *correctNumber*, *answeredNumber* i *userID* (vanjski ključ koji se odnosi na *userID* u tablici *user*).
- **slide**: Tablica koja sadrži informacije o lekcijama. Sadrži stupce *slideName*, *slideContent*, *slideID* (primarni ključ), *nextSlide* i *previousSlide* (vanjski ključevi koji se odnose na *slideID* u istoj tablici).
- **image**: Tablica koja pohranjuje URL-ove slika povezanih s lekcijama. Sadrži stupce *url*, *imageID* (primarni ključ) i *slideID* (vanjski ključ koji se odnosi na *slideID* u tablici *slide*).
- **code**: Tablica koja sadrži kodove koji su povezani s lekcijama. Sadrži stupce *correctCode*, *codeID* (primarni ključ), *nextCode* i *previousCode* (vanjski ključevi koji se odnose na *codeID* u istoj tablici).
- **question**: Tablica koja pohranjuje pitanja vezana uz lekcije. Sadrži stupce *questionID* (primarni ključ), *question* i *slideID* (vanjski ključ koji se odnosi na *slideID* u tablici *slide*).
- **written**: Tablica koja pohranjuje unose koda od strane korisnika. Sadrži stupce *enteredCode*, *codeID* (vanjski ključ koji se odnosi na *codeID* u tablici *code*) i *userID* (vanjski ključ koji se odnosi na *userID* u tablici *user*).
- **answer**: Tablica koja sadrži moguće odgovore na pitanja. Sadrži stupce *answerID*

(primarni ključ), *answerText*, *isCorrect* i *questionID* (vanjski ključ koji se odnosi na *questionID* u tablici *question*).

- **answered**: Tablica koja bilježi koje odgovore su korisnici dali. Sadrži stupce *answerID* (vanjski ključ koji se odnosi na *answerID* u tablici *answer*) i *userID* (vanjski ključ koji se odnosi na *userID* u tablici *user*).

Ovakva struktura baze podataka omogućava učinkovito upravljanje informacijama o korisnicima, lekcijama i njihovim rezultatima, te pruža fleksibilnost u dodavanju novih funkcionalnosti i proširenju aplikacije. Struktura baze podataka prikazana je na slici 5.2.



Slika 5.2. ERD model baze podataka

5.4. Deploy - Render

U aplikaciji, Render se koristi za deploy frontenda, backenda i PostgreSQL baze podataka. Svaki dio aplikacije je zasebno deployan i konfiguriran:

- **Frontend**: Frontend aplikacija je deployana kao web servis sa zadanim naredbama za pokretanje.

- **Backend**: Backend aplikacija je deployana kao web servis koristeći Docker. Render omogućava jednostavno postavljanje okruženja i varijabli za backend aplikacije.
- **Baza podataka**: PostgreSQL baza podataka je deployana kao managed database service. Render upravlja sigurnosnim postavkama, backup-om i skaliranjem baze podataka.

5.5. Upravljanje kodom - GitHub

U aplikaciji, GitHub se koristi za upravljanje kodom i konfiguracijom za frontend, backend i infrastrukturu:

- **Frontend**: Kôd za frontend aplikaciju, napisan u Reactu, pohranjen je u posebnom direktoriju repozitorija. Svaka promjena se prati kroz commitove i pull requestove.
- **Backend**: Backend aplikacija, napisana u Spring Bootu, također ima svoj direktorij u repozitoriju. GitHub omogućava kontinuirano integriranje promjena u backend aplikaciji.
- **Infrastruktura**: Konfiguracijske datoteke za deployment i infrastrukturu, kao što je Docker konfiguracija, pohranjene su u repozitoriju.

6. Opis aplikacije

Aplikacija je podijeljena u nekoliko glavnih modula:

- **Registracija i prijava korisnika** - omogućava korisnicima stvaranje računa i prijavu u sustav.
- **Lekcije** - sastoje se od tekstualnog materijala, slika i pitanja koja pomažu korisnicima u razumijevanju podržanog učenja.
- **Simulacija Q-learninga** - omogućava korisnicima interakciju s virtualnim agentom koji uči kako pronaći put kroz labirint.

6.1. Korisničke upute

Ovo poglavlje pruža detaljne upute za korištenje aplikacije.

6.1.1. Registracija i prijava

- Otvorite aplikaciju i kliknite na "Registriraj se".
- Unesite svoje ime i prezime, željeno korisničko ime i lozinku.
- Kliknite na "Registracija" kako biste stvorili novi račun.
- Nakon registracije, prijavite se unosom svog korisničkog imena i lozinke.

6.1.2. Navigacija kroz lekcije

- Nakon prijave, kliknite na "Lekcije" u glavnom izborniku.

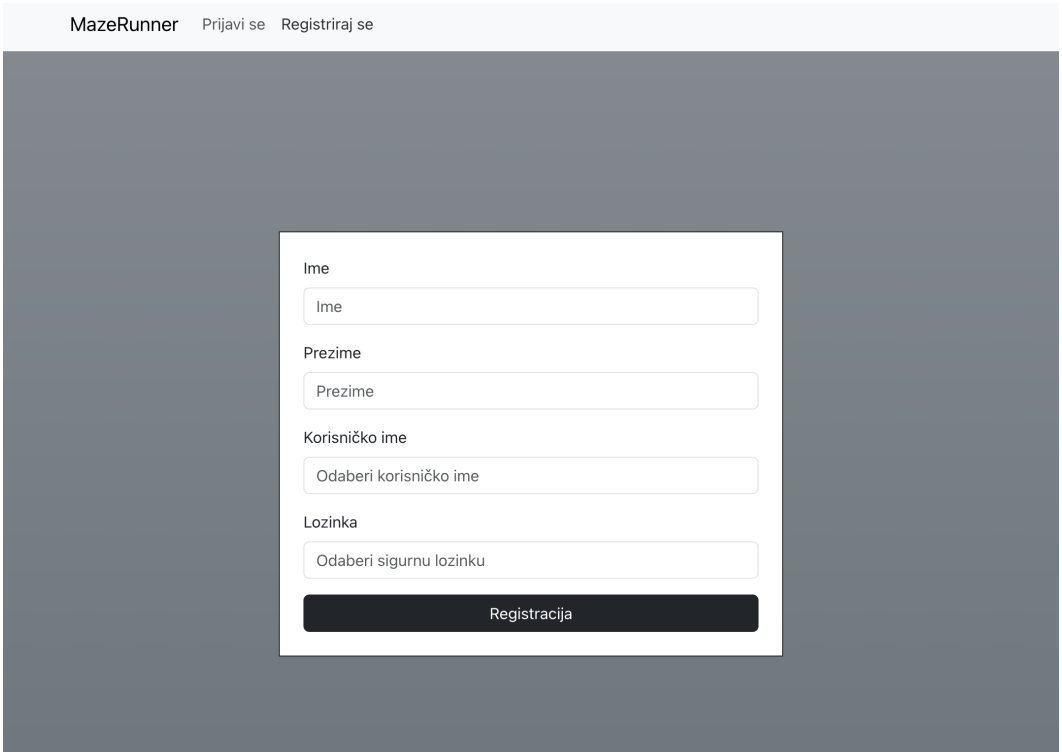
- Proučite tekstualne materijale i slike, te odgovorite na pitanja za provjeru znanja odabirom odgovora i pritiskom na gumb "Predaj", nakon što ste proučili točne odgovore na pitanja pređite na sljedeći dio lekcije tako da kliknete na gumb "Sljedeći".

6.1.3. Pokretanje simulacije Q-learninga

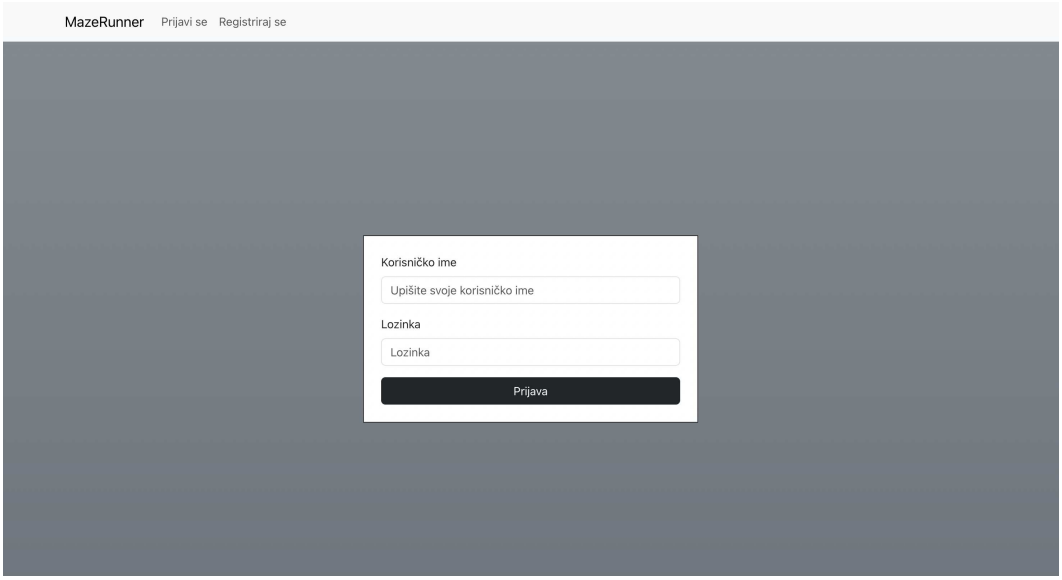
- Kliknite na "Simulacija" u glavnom izborniku.
- Postavite parametre simulacije koristeći slider-e.
- Kliknite na "Pokreni" kako biste započeli simulaciju učenja i mogli pratiti napredak agenta kroz labirint ili na gumb "Test" kako biste vidjeli koje je rješenje agent pronašao sa postavljenim parametrima.

6.2. Registracija i prijava korisnika

Korisnici se mogu registrirati pomoću svog imena i prezimena, korisničkog imena i lozinke. Nakon registracije, korisnici mogu pristupiti svom profilu i pratiti svoj napredak. Slika 6.1. prikazuje formu za registraciju korisnika, a slika 6.2. prikazuje formu za prijavu korisnika.



Slika 6.1. Forma za registraciju korisnika



Slika 6.2. Forma za prijavu korisnika

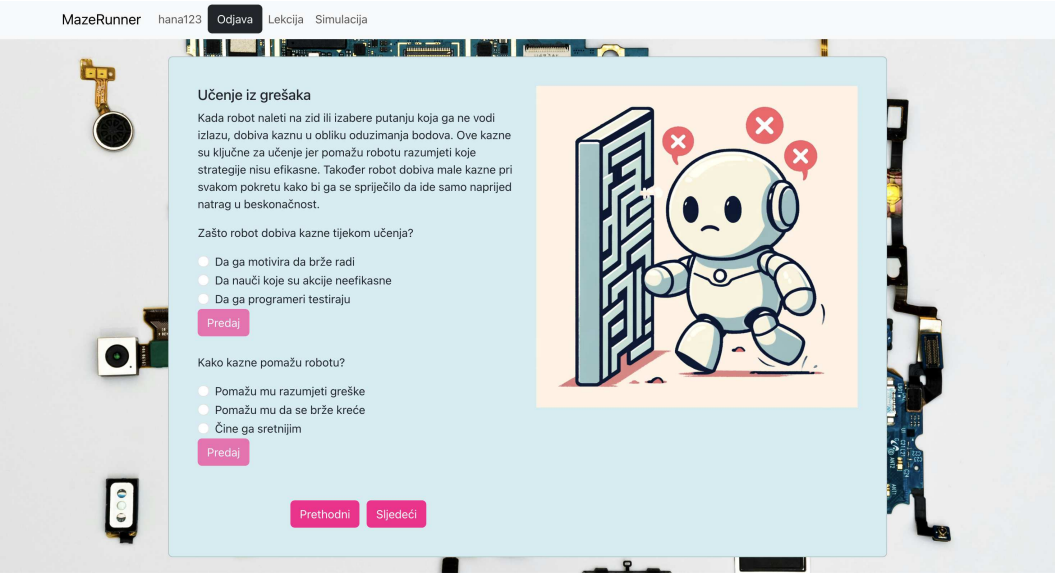
6.3. Statistika korisnika

Na početnoj stranici nakon prijave korisnici mogu vidjeti stupičasti dijagram svoje statistike koji prikazuje točnost njihovih odgovora u postotku po danu. Primjer takvog dijagrama je na slici 6.3.

Slika 6.3. Stupičasti dijagram statistike

6.4. Lekcije

Lekcije su glavni obrazovni sadržaj aplikacije. Svaka lekcija sadrži objašnjenja dijela gradiva, slike i pitanja za provjeru znanja. Primjer lekcije prikazan je na slici 6.4.

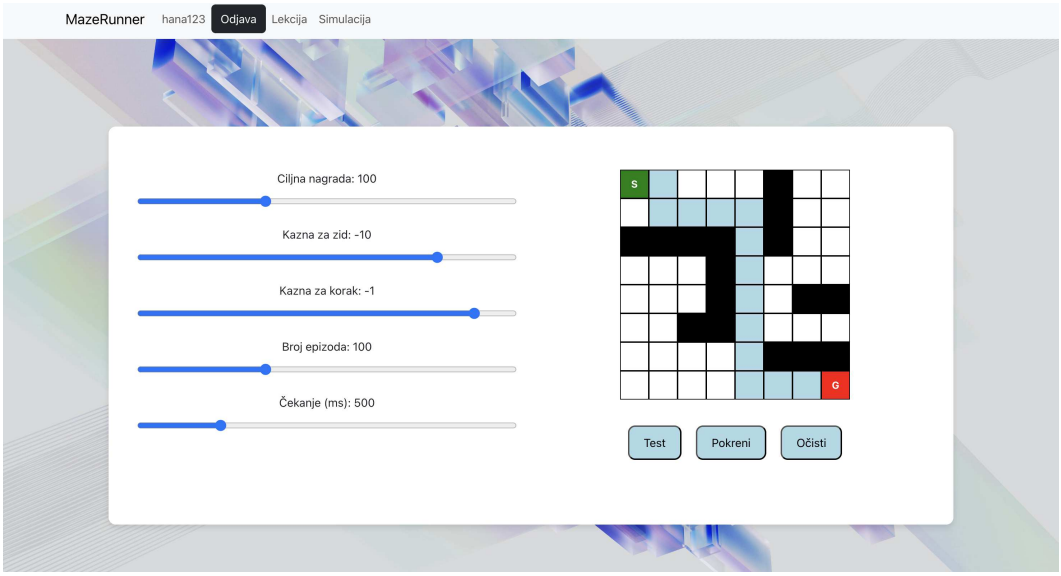


Slika 6.4. Primjer lekcije

6.5. Simulacija Q-learninga

Q-learning je algoritam podržanog učenja koji agent koristi za učenje optimalnog puta kroz labirint. Korisnici mogu pokrenuti simulaciju, mijenjati parametre i pratiti

napredak agenta. Slika 6.5. prikazuje sučelje za simulaciju Q-learninga.



Slika 6.5. Sučelje za simulaciju Q-learninga

6.5.1. Detalji simulacije Q-learninga

Simulacija uključuje nekoliko ključnih komponenti:

- **Labirint** - labirint se sastoji od mreže 8x8 ćelija, gdje korisnici mogu postaviti zidove.
- **Parametri** - korisnici mogu postaviti parametre kao što su wall penalty, step penalty i goal reward, te broj epizoda treniranja koje će agent odraditi prije testiranja naučenog.
- **Gumbi** - simulacija uključuje gumbe za testiranje i resetiranje simulacije kao i gumb za pokretanje koji pokazuje kretanje agenta kroz labirint tijekom prve epizode treniranja dok još ništa nije naučio o svojoj okolini.

6.5.2. Korišteni algoritam: Q-learning

Algoritam korišten u ovoj simulaciji je Q-learning, algoritam podržanog učenja koji agentu omogućuje učenje optimalne politike djelovanja kroz interakciju s okolinom. Q-learning koristi funkciju vrijednosti Q za procjenu koristi pojedine akcije u određenom stanju. Funkcija Q ažurira se pomoću Bellmanove jednadžbe, čime agent postupno poboljšava svoje odluke na temelju nagrada dobivenih iz okoline.

Opis Q-learning algoritma

Q-learning algoritam se sastoji od sljedećih koraka:

1. **Inicijalizacija**: Q-tablica se inicijalizira na nulte vrijednosti. Q-tablica je trodimenzionalna matrica koja sadrži Q-vrijednosti za svaki par stanja i akcije, u ovom primjeru labirinta svako polje ima 4 moguće agcije - pomak gore, dolje, lijevo ili desno pa Q-tablica ima oblik 8x8x4.

2. **Izbor akcije**: U svakom stanju agent bira akciju koristeći politiku izbora akcije koja balansira između istraživanja (eng. exploration) i iskorištavanja (eng. exploitation). U ovoj implementaciji koristi se epsilon-greedy metoda. U epsilon-greedy metodi, agent donosi odluke o tome koju akciju poduzeti na temelju dva pristupa:

   - **Eksploatacija (Exploitation)**: Agent odabire najbolju poznatu akciju na temelju trenutnih Q-vrijednosti. Ovo je korisno jer koristi prikupljeno znanje za maksimizaciju trenutne nagrade.

   - **Istraživanje (Exploration)**: Agent nasumično odabire akciju bez obzira na trenutne Q-vrijednosti. Ovo je važno jer omogućava agentu da otkrije nove akcije koje mogu dovesti do većih nagrada u budućnosti.

   ### Algoritam

   Epsilon-greedy metoda koristi parametar ϵ koji određuje vjerojatnost s kojom će agent odabrati istraživanje umjesto eksploatacije:

   - S vjerojatnošću ϵ, agent nasumično bira akciju (istraživanje).

   - S vjerojatnošću $1 - \epsilon$, agent bira najbolju poznatu akciju (eksploatacija).

   Vrijednost ϵ se često smanjuje tijekom vremena kako agent prikuplja više znanja o okolini, čime se postupno prelazi s istraživanja na eksploataciju.

3. **Izvršenje akcije**: Nakon izbora akcije, agent izvršava akciju i prelazi u novo stanje. Dobiva se nagrada za izvršenu akciju.

4. **Ažuriranje Q-vrijednosti**: Q-vrijednosti se ažuriraju pomoću Bellmanove jednadžbe:

$$Q(s,a) = Q(s,a) + \alpha(r + \gamma \max_{a'} Q(s',a') - Q(s,a))$$

gdje je:

- $Q(s,a)$ - trenutna Q-vrijednost za stanje s i akciju a
- α - stopa učenja
- r - nagrada dobivena za akciju a
- γ - faktor diskontiranja budućih nagrada
- $\max_{a'} Q(s',a')$ - maksimalna Q-vrijednost za novo stanje s'

5. **Ponavljanje**: Proces se ponavlja za definirani broj epizoda ili dok agent ne nauči optimalnu politiku.

Primjeri koda

Sljedeći kod prikazuje ključne dijelove implementacije algoritma Q-learning:

```
public int getAction(int[] state, int currentEpisode) {
    double explorationRate = getExplorationRate(currentEpisode);
    if (rand.nextDouble() < explorationRate) {
        return rand.nextInt(4);
    } else {
        double[] qValues = qTable[state[0]][state[1]];
        int bestAction = 0;
        for (int i = 1; i < qValues.length; i++) {
            if (qValues[i] > qValues[bestAction]) {
                bestAction = i;
            }
        }
        return bestAction;
    }
```

```
}

public void updateQTable(int[] state, int action, int[] nextState,
double reward) {
    double[] qValues = qTable[nextState[0]][nextState[1]];
    double bestNextActionValue = Arrays.stream(qValues).max().getAsDouble();
    double currentQValue = qTable[state[0]][state[1]][action];
    qTable[state[0]][state[1]][action] = currentQValue + learningRate *
    (reward + discountFactor * bestNextActionValue - currentQValue);
}

public static String trainAgent(QLearningAgent agent, Maze maze,
int numEpisodes) {
    List<Integer> episodeRewards = new ArrayList<>();
    List<Integer> episodeSteps = new ArrayList<>();

    for (int episode = 0; episode < numEpisodes; episode++) {
        EpisodeResult result = finishEpisode(agent, maze, episode, true);
        episodeRewards.add((int) result.episodeReward);
        episodeSteps.add(result.episodeStep);
    }

    int averageReward = episodeRewards.stream().mapToInt(Integer::intValue)
    .sum() / episodeRewards.size();
    int averageSteps = episodeSteps.stream().mapToInt(Integer::intValue)
    .sum() / episodeSteps.size();

    StringBuilder sb = new StringBuilder();
    sb.append("The average reward is: " + averageReward + "\n");
    sb.append("The average steps is: " + averageSteps + "\n");
    return sb.toString();
}
```

```
public static String testAgent(QLearningAgent agent, Maze maze,
int numEpisodes) {
    lastEpisodeResult = finishEpisode(agent, maze, numEpisodes, false);
    int episodeReward = (int) lastEpisodeResult.episodeReward;
    int episodeStep = lastEpisodeResult.episodeStep;
    List<int[]> path = lastEpisodeResult.path;
    StringBuilder sb = new StringBuilder();
    sb.append("Learned Path:");
    for (int[] position : path) {
        sb.append("(" + position[0] + ", " + position[1] + ")-> ");
    }
    sb.append("\n");
    sb.append("Goal!\n");
    sb.append("Number of steps: " + episodeStep + "\n");
    sb.append("Total reward: " + episodeReward + "\n");
    return sb.toString();
}

public static EpisodeResult finishEpisode(QLearningAgent agent, Maze maze,
int currentEpisode, boolean train) {
    int[] currentState = maze.startPosition;
    boolean isDone = false;
    int episodeReward = 0;
    int episodeStep = 0;
    List<int[]> path = new ArrayList<>();
    path.add(Arrays.copyOf(currentState, currentState.length));

    while (!isDone) {
        int action = agent.getAction(currentState, currentEpisode);
        int[] nextState = { currentState[0] + ACTIONS[action][0],
        currentState[1] + ACTIONS[action][1] };
```

```
            if (nextState[0] < 0 || nextState[0] >= maze.mazeHeight ||
            nextState[1] < 0 || nextState[1] >= maze.mazeWidth ||
            maze.maze[nextState[0]][nextState[1]] == 1) {
                nextState = currentState;
                episodeReward += WALL_PENALTY;
            } else if (Arrays.equals(nextState, maze.goalPosition)) {
                path.add(Arrays.copyOf(nextState, nextState.length));
                episodeReward += GOAL_REWARD;
                isDone = true;
            } else {
                path.add(Arrays.copyOf(nextState, nextState.length));
                episodeReward += STEP_PENALTY;
            }

            if (train) {
                agent.updateQTable(currentState, action, nextState, episodeReward);
            }

            currentState = nextState;
            episodeStep++;
        }

        lastEpisodeResult = new EpisodeResult(episodeReward, episodeStep, path,
         agent.qTable);
        return lastEpisodeResult;
    }
```

6.6. Implementacija

6.6.1. Frontend implementacija

Frontend aplikacije je implementiran koristeći React. U nastavku je prikazan dio koda za komponentu lekcija:

```
import React, { useState, useEffect } from 'react';
import { Button, Card, Row, Col } from 'react-bootstrap';
import backgroundImage from '../assets/lessionBackground.jpg';
import Question from './Questions';
import { useNavigate } from 'react-router-dom';

function SlideComponent({ initialSlideId = 1 }) {
  const [slideId, setSlideId] = useState(initialSlideId);
  const [slide, setSlide] = useState(null);
  const [loading, setLoading] = useState(true);
  const [error, setError] = useState(null);
  const navigate = useNavigate();

  useEffect(() => {
    const fetchSlide = async () => {
      try {
        const response = await fetch(`https://mazerunnerbackend.onrender.com/
        slides/${slideId}`);
        if (!response.ok) {
          throw new Error(`HTTP error! status: ${response.status}`);
        }
        const data = await response.json();
        setSlide(data);
        setLoading(false);
      } catch (e) {
        console.error(e);
```

```
      setError(e.message);
      setLoading(false);
    }
  };

  fetchSlide();
}, [slideId]);

if (loading) return <div>Učitavanje...</div>;
if (error) return <div>Greška: {error}</div>;
if (!slide) return <div>Lekcija nije pronađena</div>;

return (
  <div className="slide-component-wrapper" style={{
    backgroundImage: `url(${backgroundImage})`,
    backgroundSize: 'cover',
    backgroundPosition: 'center',
    backgroundAttachment: 'fixed',
    overflow: 'hidden',
    minHeight: '100vh',
  }}>
    <div className="d-flex justify-content-center align-items-center
    flex-column p-4">
      <Card className="w-60 p-4 mb-4" style={{ width: '1000px',
      backgroundColor: '#D1EDF2' }}>
        <Card.Body>
          <Row>
            <Col xs={6}>
              {slide.slideName && <Card.Title>{slide.slideName}</Card.Title>}
              {slide.slideContent && <Card.Text>{slide.slideContent}
              </Card.Text>}
              {slide.questions && slide.questions.map((question, qIndex) => (
```

```
        <div key={qIndex} className="mb-4">
          <Question question={question} />
        </div>
      ))}
    </Col>
    <Col xs={6}>
      {slide.images && slide.images.map((image, index) => (
        <img
          key={index}
          src={image.url}
          alt="Slide"
          className="img-fluid mb-3"
          style={{ maxHeight: '90%', width: 'auto' }}
        />
      ))}
    </Col>
  </Row>
  <Row className="mt-4">
    <Col xs={6} className="d-flex align-items-center
    justify-content-center">
      <Button
        onClick={() => slide.previousSlide &&
        setSlideId(slide.previousSlide.slideId)}
        variant="primary"
        disabled={!slide.previousSlide}
        style={{ backgroundColor: '#FF028D', borderColor: '#FF028F',
        marginRight: '10px' }}
      >
        Prethodni
      </Button>
      <Button
        onClick={() => {if (slide.nextSlide){
```

```
                    setSlideId(slide.nextSlide.slideId)} else
                    {navigate("/home")}}}
                  variant="primary"
                  style={{ backgroundColor: '#FF028D',
                  borderColor: '#FF028F' }}
                >
                  Sljedeći
                </Button>
              </Col>
            </Row>
          </Card.Body>
        </Card>
      </div>
    </div>
  );
}

export default SlideComponent;
```

6.6.2. Backend implementacija

Backend aplikacije je implementiran koristeći Spring Boot. U nastavku je prikazan dio koda za kontroler koji upravlja prikazivanjem lekcija:

```
package app.controllers;

import app.models.SlideModel;
import app.repositories.SlideRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;

import java.util.Optional;

@RestController
```

```
@RequestMapping("/slides")
public class SlideController {

    @Autowired
    private SlideRepository slideRepository;

    @GetMapping("/{id}")
    public SlideModel getSlideById(@PathVariable int id) {
        Optional<SlideModel> slide = slideRepository.findBySlideId(id);
        return slide.orElse(null);
    }
}
```

7. Testiranje

7.1. Primjeri scenarija korištenja

U ovom poglavlju opisat ćemo nekoliko primjera scenarija korištenja aplikacije.

7.1.1. Scenarij 1: Učenje osnova podržanog učenja

Učenik se prijavljuje u aplikaciju i pristupa lekcijama koje objašnjavaju osnovne koncepte podržanog učenja. Nakon proučavanja svakog dijela lekcije, učenik rješava pitanja za provjeru usvojenosti znanja za taj dio lekcije te zatim prelazi na sljedeći dio.

7.1.2. Scenarij 2: Treniranje agenta u labirintu

Učenik pokreće simulaciju Q-learninga, postavlja parametre simulacije i prati kako agent uči optimalan put kroz labirint. Učenik može mijenjati parametre i analizirati rezultate kako bi bolje razumio utjecaj različitih faktora na proces učenja.

7.1.3. Scenarij 3: Praćenje napretka

Učenik pristupa svom profilu kako bi pratio svoj napredak kroz lekcije i simulacije. Statistika korisnika prikazuje se u obliku stupičastog dijagrama koji prikazuje postotak točno riješenih zadataka po danima.

7.2. Testiranje aplikacije

Aplikacija je testirana s grupom učenika osnovne škole(N=24) kako bi se procijenila njezina učinkovitost i korisničko iskustvo.

7.2.1. Metodologija testiranja

Učenici su pozvani da koriste aplikaciju tijekom jednog školskog sata. Testiranje je obuhvatilo sljedeće aktivnosti:

- Prijava u aplikaciju i pregled lekcija.
- Rješavanje pitanja za provjeru znanja nakon svake lekcije.
- Pokretanje simulacije Q-learninga i eksperimentiranje s različitim parametrima.
- Praćenje osobnog napretka kroz prikaz statistike u obliku dijagrama.

7.2.2. Rezultati testiranja

Rezultati testiranja pokazali su da su učenici uspješno koristili aplikaciju i ostvarili značajan napredak u razumijevanju podržanog učenja. Prosjek točnih odgovora na pitanja za provjeru znanja iznosio je 75.5%. Učenici su također pokazali velik interes za simulaciju Q-learninga i aktivno su eksperimentirali s parametrima kako bi vidjeli kako to utječe na ponašanje agenta.

7.2.3. Zaključak testiranja

Testiranje je pokazalo da je aplikacija učinkovita u poučavanju podržanog učenja i da pruža korisničko iskustvo koje je prilagođeno potrebama učenika osnovne škole(Slika 7.1.).

Slika 7.1. Učenici osnovne škole koriste aplikaciju za podržano učenje

8. Zaključak

Razvoj ove obrazovne aplikacije pokazao je važnost interaktivnog pristupa u poučavanju složenih koncepata kao što je podržano učenje. Korištenjem modernih tehnologija kao što su React, Spring i PostgreSQL, stvorili smo alat koji omogućava učenicima da na praktičan način nauče ključne pojmove podržanog učenja. Budući rad uključuje dodatno proširenje funkcionalnosti aplikacije i testiranje u stvarnim obrazovnim okruženjima kako bi se poboljšala učinkovitost i korisničko iskustvo.

Literatura

[1] React Documentation, "React - A JavaScript library for building user interfaces", dostupno na: https://reactjs.org/docs/getting-started.html, zadnje pristupljeno: lipanj 2024.

[2] Spring Framework Documentation, "Spring Framework", dostupno na: https://spring.io/projects/spring-framework, zadnje pristupljeno: lipanj 2024.

[3] PostgreSQL Documentation, "PostgreSQL Documentation", dostupno na: https://www.postgresql.org/docs/, zadnje pristupljeno: lipanj 2024.

[4] Science Buddies Project Ideas, "Machine Learning Maze", dostupno na: https://www.sciencebuddies.org/science-fair-projects/project-ideas/ArtificialIntelli p008/artificial-intelligence/machine-learning-maze, zadnje pristupljeno: lipanj 2024.

[5] React Course - Beginner's Tutorial for React JavaScript Library [Video]. *React Crash Course 2021*. Dostupno na: https://www.youtube.com/watch?v=QUNM-QyM5PA&list=LL&index=16&t=1s, zadnje pristupljeno: lipanj 2024.

[6] Spring Boot Tutorial for Beginners [Video]. *Spring Boot Quick Start*. Dostupno na: https://www.youtube.com/watch?v=x_nfnVU0wAI&list=LL&index=14, zadnje pristupljeno: lipanj 2024.

[7] PostgreSQL Tutorial for Beginners [Video]. *PostgreSQL Crash Course*. Dostupno na: https://www.youtube.com/watch?v=ltvRsnka7Mo&list=LL&index=24, zadnje pristupljeno: lipanj 2024.

[8] Q-learning Algorithm Explained [Video]. *Reinforcement Learning - Q-learning*.

Dostupno na: https://www.youtube.com/watch?v=j942wKiXFu8&list=PL4cUxeGkcC9gZD-Tvwfod2gaISzfRiP9d, zadnje pristupljeno: lipanj 2024.

[9] Introduction to Reinforcement Learning [Video]. *Intro to Reinforcement Learning*. Dostupno na: https://www.youtube.com/watch?v=9SGDpanrc8U&t=5s, zadnje pristupljeno 13. lipnja 2024.

[10] M. Čupić, "Umjetna inteligencija - Podržano učenje", dostupno na: http://java.zemris.fer.hr/nastava/ui/rl/rl-20200401.pdf, zadnje pristupljeno: lipanj 2024.

[11] CSS Introduction, W3schools, "Introduction to CSS", dostupno na: https://www.w3schools.com/css/css_intro.asp, zadnje pristupljeno: lipanj 2024.

[12] What is JavaScript?, W3schools, "Introduction to JavaScript", dostupno na: https://www.w3schools.com/whatis/whatis_js.asp, zadnje pristupljeno: lipanj 2024.

[13] Introduction to Bootstrap, "Bootstrap Introduction", dostupno na: https://getbootstrap.com/docs/5.3/getting-started/introduction/, zadnje pristupljeno: lipanj 2024.

[14] What is Q-learning?, Simplilearn, "Q-learning Tutorial", dostupno na: https://www.simplilearn.com/tutorials/machine-learning-tutorial/what-is-q-learning, zadnje pristupljeno 13. lipnja 2024.

[15] Q-learning Definition, TechTarget, "Definition of Q-learning", dostupno na: https://www.techtarget.com/searchenterpriseai/definition/Q-learning, zadnje pristupljeno: lipanj 2024.

[16] T. Jagušt, V. Šturlić-Tupek i M. A. Rogošić, "AI iz kutije šibica", prezentacija, zadnje pristupljeno: lipanj 2024.

[17] S. Seegerer, "Schlag das Krokodil", dostupno na: https://www.stefanseegerer.de/schlag-das-krokodil/, zadnje pristupljeno: lipanj 2024.

Popis slika

Sažetak

Obrazovna interaktivna aplikacija za poučavanje podržanog učenja

Hana Ivančić

Uvođenjem novih tehnologija u obrazovni sustav omogućuje se modernizacija i digitalizacija tradicionalnih metoda poučavanja. Ovaj završni rad bavi se razvojem obrazovne interaktivne aplikacije za poučavanje podržanog učenja, koristeći suvremene tehnologije kao što su React, Spring i PostgreSQL. Aplikacija omogućuje učenicima da kroz simulacije i praktične primjere bolje razumiju principe podržanog učenja, posebice Q-learning algoritam. Pomoću aplikacije, učenici mogu postavljati različite parametre simulacije i pratiti kako agent uči put kroz labirint, što doprinosi dubljem razumijevanju i praktičnoj primjeni teorijskih znanja.

Ključne riječi: Podržano učenje, Q-learning, React, Spring, PostgreSQL

Abstract

Educational Interactive Application for Teaching Reinforcement Learning

Hana Ivančić

The integration of new technologies into the educational system facilitates the modernization and digitalization of traditional teaching methods. This thesis focuses on the development of an educational interactive application for teaching reinforcement learning, utilizing modern technologies such as React, Spring, and PostgreSQL. The application enables students to better understand the principles of reinforcement learning, particularly the Q-learning algorithm, through simulations and practical examples. With the application, students can set various simulation parameters and observe how an agent learns a path through a maze, contributing to a deeper understanding and practical application of theoretical knowledge.

Keywords: Reinforcement learning, Q-learning, React, Spring, PostgreSQL