Zahvaljujem mentoru doc. dr. sc. Tomislavu Jaguštu na vođenju i savjetima prilikom izrade ovog rada

Sadržaj

Uvod

Jedan od problema današnjeg učenja u školama je motivacija i zainteresiranost učenika, bilo za neki predmet u cjelini ili za određeno gradivo nekog predmeta. Tako je i u slučaju učenja, računanja i prepoznavanja novčanica i kovanica, te razumijevanja da novac, bilo papir ili kovanica, ima određenu vrijednost. Također, nastavnici često mogu imati poteškoća u motiviranju učenika da obrate pažnju na sadržaj koji se obrađuje na satu. Uz problem zainteresiranosti učenika, nastavnici se često suočavaju i s problemom neadekvatnog uvida u trenutno znanje pojedinog učenika, tj. s kašnjenjem povratne informacije. Običnim pismenim testom je prilično teško detaljno pratiti znanje svakog pojedinog učenika, ili izmjeriti koliko dugo je trajao učenikov proces razmišljanja i zaključivanja na pojedinom pitanju. Za razliku od vježbi i testova na papiru, tehnologijom podržano učenje, tj. obrazovne web aplikacije nude rješenja na neke od ovih izazova, npr. zainteresiranosti učenika za vježbanje i lakše praćenje i analizu učenikovih odgovora. Cilj ovog rada je implementirati web aplikaciju koja će učenicima pomoći da savladaju nastavno gradivo vezano uz učenje o novcu i iznosima pojedinih novčanica i kovanica, a istovremeno nastavnicima omogućiti bolji nadzor nad procesom učenja, praćenje napretka svakog pojedinog učenika, olakšati kreiranje, ponovno korištenje i dijeljenje zadataka.

Web aplikaciju koriste i nastavnici i učenici. Nastavnici kroz sučelje dodaju nove zadatke za vježbu. Postoji nekoliko različitih vrsta zadataka – pitanja s ponuđenim višestrukim odgovorima, pitanja s unosom, pitanja sa slikovnim odabirom i sl. Uz zadatke, mogu se definirati i vježbe za rješavanje u koje se dodaju ranije napravljeni zadaci, te dodatne informacije o vježbi kao što je npr. valuta. Ovdje se vidi još jedna prednost implementacije – omogućeno je učenje i vježbanje s novčanicama i kovanicama različitih valuta. Pri rješavanju vježbe, učenici unose odgovore sve dok ne unesu točan odgovor, a kad unesu netočan odgovor ponudit će im se pomoć u obliku savjeta (en. hint). Mogućnosti interakcije s aplikacijom, slikovni zadaci s različitim novčanicama i kovanicama učiniti će učeniku dotično gradivo zanimljivijim i jasnijim.

Pri svakom unosu odgovora, bilo točnog ili netočnog, podaci o odgovoru se spremaju u bazu podataka. Ti podaci se mogu koristiti za daljnju analizu i statistiku koja je kroz aplikaciju vidljiva nastavniku. Na taj način, nastavnici mogu dobiti brzu povratnu informaciju o učenikovom znanju. Analizom podataka lakše će se detektirati učenici koji imaju problema sa svladavanjem gradiva, te će im nastavnici moći posvetiti više vremena.

U ovom radu će biti opisan način izrade web aplikacije za vježbanje i prepoznavanje novčanica i kovanica kao i njene funkcionalnosti. Usporedit će se sa sličnim dostupnim aplikacijama, prikazat će se tehnologije i alati koji su se koristili pri izradi sustava, modeli podataka, te arhitektura i opis sustava krajnjeg proizvoda.

1. Slične aplikacije/igre

1.1. UK Coin Calculator

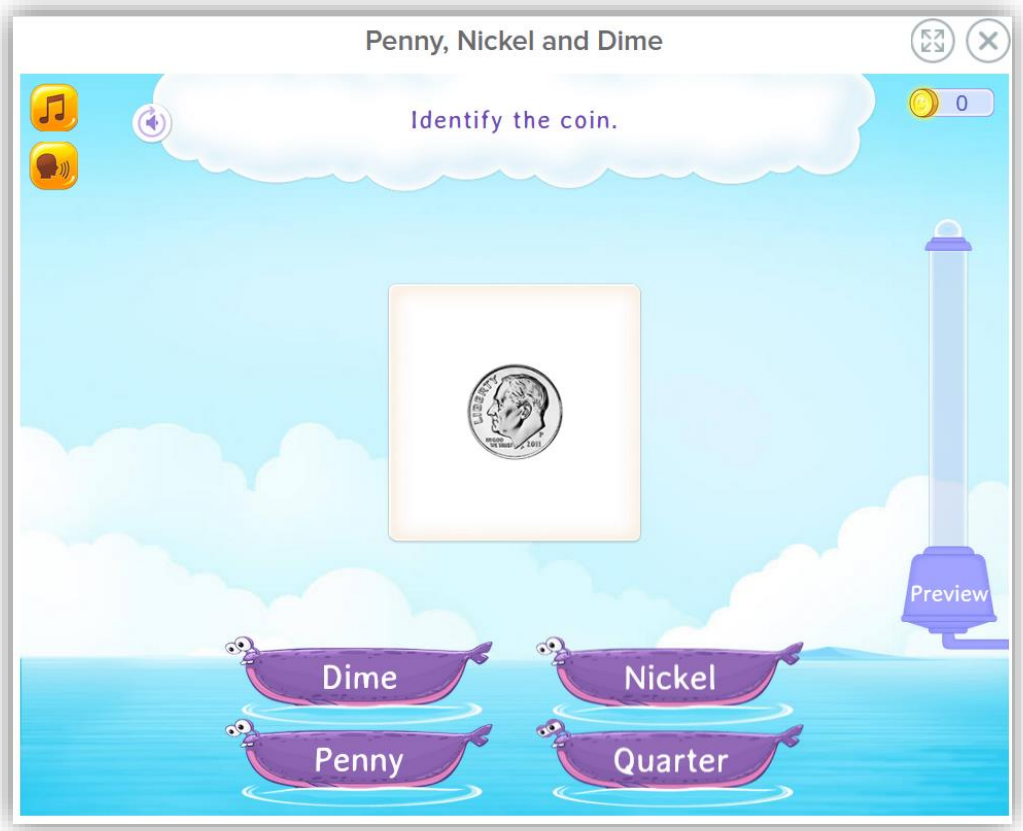
UK Coin Calculator[1] radi kao kalkulator, ali svi su ulazi samo kovanice. Ovaj kalkulator novca je izvrstan za djecu koja uče kako zbrajati novac. Pritisne se bilo koji od novčića, a zatim gumb + ili – da se doda ili oduzme taj iznos od svog ukupnog iznosa. Pomaže u podučavanju i učenju računanja s novcem. Također moguće je mijenjati valute: američke dolare, funte, eure, kanadske dolare i australske dolare. Nedostaci aplikacije su što učenici trebaju unaprijed imati znanja s novcem i znati kako koja kovanica izgleda, te moraju imati znanje o vrijednosti kovanica. Aplikacija nema administratorsko sučelje (za nastavnika) pa tako ni mogućnost praćenja i analize odgovora i samo služi za vježbanje zbrajanja i oduzimanja kovanica.

Slika 1.1 Sučelje za oduzimanje i zbrajanje kovanica

[1] https://apps.apple.com/gb/app/uk-coin-calculator/id501983186

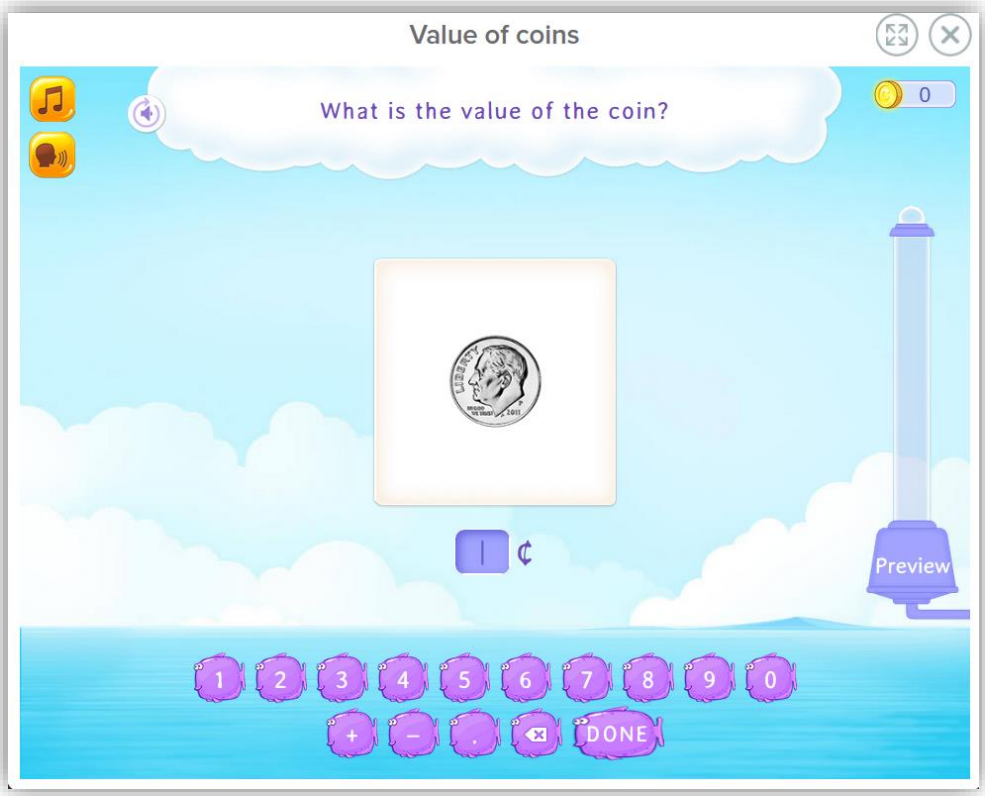
1.2. SplashLearn

SplashLearn[2] je besplatna aplikacija za roditelje i nastavnike. Aplikacija ima i matematičke i novčane aktivnosti za djecu. Ima odjeljak za novac koji će učenicima pomoći da nauče: brojati kovanice *Penny*, *Nickels* i *Dimes*, vrijednost kovanica, brojanje novca, uspoređivanje iznosa, oduzimanje i dodavanje novca, itd. Pri pokretanju vježbe prikaže se prozor sa zadatkom te ovisno o vrsti vježbe traži od korisnika potrebnu akciju, npr. za prikazani novac treba odabrati gumb koji je novac prikazan. Nema mogućnost dodavanja zadataka i vježbi niti analizu urađenih vježbi i učenikovih odgovora.



Slika 1.2 Zadatak s prepoznavanjem kovanica

[2] https://www.splashlearn.com/math-skills/second-grade



Slika 1.3 Zadatak s učenjem vrijednosti kovanica

Slika 1.4 Zadatak s izračunom ostatka

1.3. Counting coins

Counting coins[3] je jednostavna, besplatna aplikacija za djecu koja uči kako brojati novčiće. Učeniku se daju različite kombinacije novčića i od njega se traži da uskladi vrijednost koristeći drugu kombinaciju novčića. Od njih se također traži da unesu vrijednost kovanica prikazanu na ekranu, kao i da koriste novčiće kako bi pogodili iznos koji je dat. Isto kao i kod prethodne aplikacije nedostatak je to što nema sučelje za nastavnike preko kojeg bi mogli dodavati zadatke i grupe zadataka, te pratiti kako su djeca rješavala vježbe i testove, te kasnije raditi analizu.

Slika 1.5 Zadatak s prepoznavanjem vrijednosti i zbrajanjem kovanica

[3] https://apps.apple.com/us/app/counting-coins/id374981504

2. Ideja rada i opis dijelova sustava

Potrebno je razviti sustav koji će iskoristiti prednosti prethodno navedenih aplikacija, ali i nadomjestiti njihove nedostatke. Nastavnici bi mogli dodavati nove učenike, zadatke i vježbe, uređivati sve navedeno i zadati koja vježba će se rješavati. Učenici bi zatim rješavali zadanu vježbu na školskim tabletima u web-preglednicima. Nakon što učenici riješe vježbu, nastavnik će imati mogućnost uvida u statistiku rješavanja vježbe za svakog svog učenika.

Prva stvar koju treba odraditi pri korištenju sustava je napraviti zadatke i vježbe za rješavanje. Za to, nastavnici se moraju prvo prijaviti s korisničkim imenom i lozinkom. Nakon uspješne prijave nastavnik je preusmjeren na svoju profilnu stranicu na kojoj su mu prikazani dosad napravljeni zadaci i vježbe, te njegova škola i učenici iz njegovih razreda. Također, ima opcije za dodavanje novih zadataka, vježbi i studenata kao i njihovo uređivanje. Vrste zadataka koje može napraviti su – zadaci s tekstualnim unosom, zadaci s višestrukim ponuđenim odgovorima, slikovni zadaci s tekstualnim unosom i slikovni zadaci s odabirom novčanica i/ili kovanica. Nakon definiranja zadataka potrebno je definirati i grupe zadataka odnosno vježbe. Pri njihovom definiranju potrebno je dodati novo napravljene ili ranije definirane zadatke koji će se rješavati, valutu koja će se ispitivati u dotičnoj vježbi, te moraju odabrati opciju hoće li se ta vježba rješavati ili ne. Ako je već neka vježba prije bila postavljena za rješavanje, neće biti moguće postaviti i novu vježbu za rješavanje.

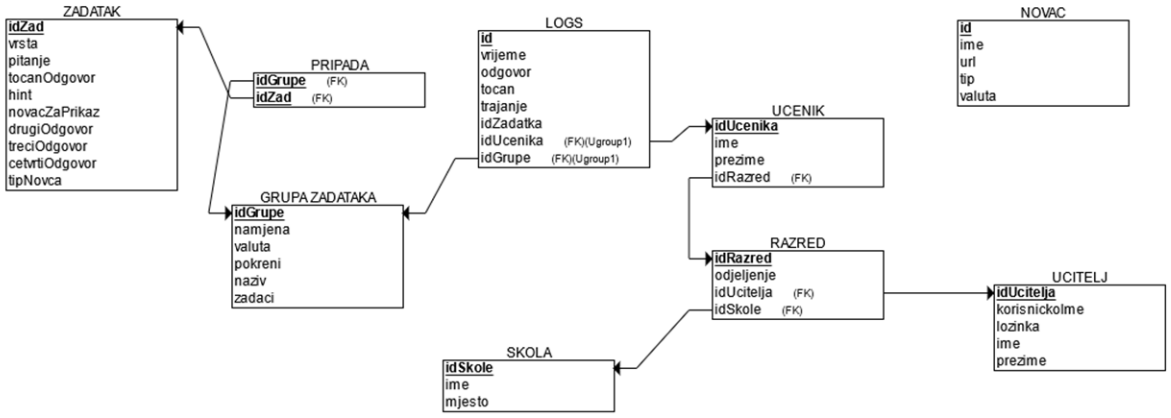
Kada je postavljeno koja će se vježba rješavati, učenici će na početnoj stranici kliknuti na gumb „*START*“. Klikom na gumb sustav će ih odvesti na prijavu gdje će morati odabrati koju školu i razred pohađaju, te svoje ime. Nakon uspješne prijave sustav učenicima prikazuje zadatke za rješavanje i to jednog po jednog. Svaki učenik može vježbu riješiti samo jednom, a redoslijed zadataka u vježbama je uvijek takav da se učenicima prvo prikazuju zadaci koji će ih učiti prepoznavanje novčanica i kovanica, a tek nakon uspješno odrađenih zadataka s prepoznavanjem prikazivati će im se zadaci s računanjem, bilo slikovni ili tekstualni. Ako učenicima bude potrebna pomoć, zadaci će imati mogućnost prikaza pomoći pri rješavanju. Pri unosu odgovora, bilo točnog ili netočnog, sustav ga pohranjuje zajedno sa vremenom njegovog rješavanja.

Nakon što je točno odgovorio na sva pitanja učenik završava vježbu, a nastavnik na svome sučelju može vidjeti statistiku urađene vježbe za svakog svog učenika koji ju je rješavao. U statistici su mu prikazani točni i netočni odgovori učenika, vrijeme rješavanja svakog

zadatka kao i ukupno vrijeme rješavanja vježbe.

3. Model podataka

3.1. Relacijski model podataka



Slika 3.1 Relacijski model baze podataka

3.2. Opis relacijskog modela podataka

Baza podataka se sastoji od devet entiteta među kojima su važniji entiteti *zadatak* i *grupa zadataka*. Nastavnik kreira zadatke koji se mogu koristiti u više različitih grupa zadataka, odnosno vježbi. Standardni atributi svakog zadatka su atribut *vrsta* po kojem je zadatak specijaliziran, atribut *hint* koji služi kao pomoć učeniku koji se muči s rješavanjem zadatka, te atributi u kojima se nalaze pitanje i točan odgovor za dotični zadatak. Pomoću atributa *tipNovca* mogu se prikazati slike novčanica i/ili kovanica u zadacima s odabirom novca. Ovisno o tom atributu, u zadatku se prikazuju slike i novčanica i kovanica ili slike novčanica i kovanica pojedinačno. Kod zadataka sa slikovnim unosom, od velike važnosti je atribut *novacZaPrikaz* pomoću kojeg se učeniku prikazuju novčanice i/ili kovanice koje je nastavnik eksplicitno predodredio za određeni zadatak.

Entitet *grupa zadataka* je zapravo vježba koju učenici rješavaju. U tom entitetu je bitan

atribut *pokreni* koji služi kao pojedinačni bit koji može biti postavljen samo za jednu vježbu. Pomoću njega nastavnik postavlja koju će vježbu učenici rješavati. Valuta koja se koristi u vježbi je određena istoimenim atributom.

Pošto se svaki zadatak može nalaziti u više grupa zadataka, koristimo entitet *pripada* pomoću kojeg možemo saznati kojoj sve vježbi pripada određeni zadatak, te od kojih se sve zadataka vježba sastoji.

Kad učenik unese odgovor, podaci se spremaju u tablicu *logs*. Bilježi se odgovor koji je određeni učenik unio za određeni zadatak i vježbu. Također se pamti točan odgovor za određeni zadatak, točno vrijeme kada je odgovor unesen, te koliko dugo vremena je bilo potrebno učeniku za unos odgovora.

Za korisnike se koriste dva entiteta kako bi se mogle odvojiti uloge i prava pristupa. Za učenike se koristi entitet *ucenik* u kojem se nalaze učenikove osnovne informacije: ime, prezime, te šifra razreda kojeg pohađa. Entitet *ucitelj* se koristi za spremanje informacija o nastavniku, od kojih se za prijavu u sustav koriste korisničko ime i lozinka. Učenici mogu pripadati jednom razredu koji pripada jednoj školi, dok nastavnik može voditi više razreda u jednoj školi.

Entitet *novac* se koristi za pohranu novčanica i kovanica koji se koriste u zadacima. Atribut *url* predstavlja URL slike za određeni novac, a pomoću atributa *tip* se pohranjuje informacija o tome radi li se o novčanici ili kovanici. Pomoću tipa novca nastavnik može pri sastavljanju zadatka odrediti koji tip novca će se koristiti za prikaz.

3.3 Opisi entiteta

Zadatak

- **idZad** – šifra zadatka (PK)
- vrsta – vrsta zadatka (unos, odabir, slikovno s unosom, slikovno s odabirom)
- pitanje – tekst pitanja zadatka
- tocanOdgovor – tekst točnog odgovora
- hint – pomoć pri rješavanju zadatka
- novacZaPrikaz – imena novčanica i kovanica za prikaz u zadacima sa slikovnim

odabirom

- drugiOdgovor – ponuđeni odgovor u zadacima s višestrukim izborom
- treciOdgovor – ponuđeni odgovor u zadacima s višestrukim izborom
- cetvrtiOdgovor – ponuđeni odgovor u zadacima s višestrukim izborom
- tipNovca – tip novca za prikaz (novčanica, kovanica, oboje)

Grupa zadataka

- **idGrupe –** šifra grupe zadataka (PK)
- namjena – podatak koji označava koristi li se vježba za ispitivanje ili vježbanje
- valuta – valuta koja se koristi u vježbi
- pokreni – zastavica koja označava rješava li se vježba ili ne
- naziv – naziv vježbe
- zadaci – šifre zadataka koji se koriste u vježbi

Pripada

- **idGrupe** – šifra grupe zadataka (PK)
- **idZad** – šifra zadatka (PK)

Logs

- **id** – šifra loga (PK)
- vrijeme – vrijeme(timestamp) kada je učenik unio odgovor
- odgovor – tekst odgovora kojeg je unio učenik za određeno pitanje
- tocan – tekst točnog odgovora za određeno pitanje
- trajanje – vrijeme koje je bilo potrebno učeniku da unese odgovor
- idZadatka – šifra zadatka na kojeg se log odnosi
- idUcenika – šifra učenika na kojeg se log odnosi
- idGrupe – šifra vježbe na koju se log odnosi

Ucenik

- **idUcenika** – šifra učenika (PK)
- ime – ime učenika
- prezime – prezime učenika
- idRazred – šifra razreda kojem učenik pripada

Razred

- **idRazred** – šifra razreda (PK)
- odjeljenje – naziv odjeljenja
- idUcitelja – šifra nastavnika koji predaje tom razredu
- idSkole – šifra škole kojoj razred pripada

Skola

- **idSkole** – šifra škole (PK)
- ime – naziv škole
- mjesto – mjesto u kojem se škola nalazi

Ucitelj

- **idUcitelja** – šifra nastavnika (PK)
- korisnickoIme – korisničko ime za prijavu
- lozinka – lozinka za prijavu
- ime – ime nastavika
- prezime – prezime nastavnika

Novac

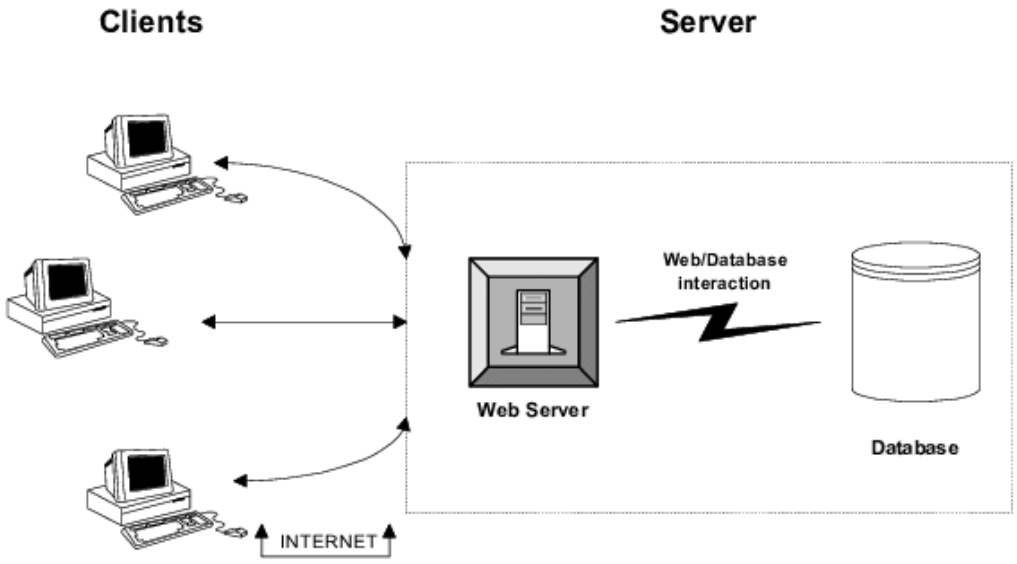
- **id** – šifra novca (PK)
- ime – naziv novca

- url – URL slike novca
- tip – tip novca (novčanica, kovanica)

4. Korištene tehnologije i alati

Arhitektura sustava se može podijeliti na tri dijela:

- Klijent – program koji pristupa poslužitelju (ili više njih) tražeći uslugu [1]
- Web poslužitelj – program koji dostavlja uslugu drugim programima koji su spojeni na njega preko komunikacijskog kanala [1]
- Baza podataka



Slika 4.1 Arhitektura klijent - poslužitelj

Arhitektura klijent-poslužitelj je arhitektura u kojoj mnogi klijenti traže i primaju uslugu od centraliziranog poslužitelja (host računala). Klijentska računala pružaju sučelje koje omogućuje korisniku računala da zatraži usluge poslužitelja i da prikaže rezultate koje poslužitelj vraća. Poslužitelji čekaju da stignu zahtjevi klijenata i zatim odgovaraju na njih.

U idealnom slučaju, poslužitelj klijentima pruža standardizirano transparentno sučelje tako da klijenti ne moraju biti svjesni specifičnosti sustava (tj. hardvera i softvera) koji pruža uslugu. Klijenti se često nalaze na radnim stanicama ili na osobnim računalima, dok se poslužitelji nalaze drugdje u mreži, obično na snažnijim računalima. Klijent i poslužitelj komuniciraju pomoću HTTP protokola. To je komunikacijski protokol koji se koristi za povezivanje klijenata s web poslužiteljima na Internetu ili na lokalnoj mreži (intranet) [2]. Primarna funkcija HTTP-a je uspostaviti vezu s poslužiteljem i poslati HTML stranice natrag u korisnikov preglednik [2]. Također se koristi za preuzimanje podataka s poslužitelja u preglednik ili u bilo koju aplikaciju koja koristi HTTP. To je protokol bez očuvanja stanja što znači da održava vezu između klijenta i poslužitelja samo za trenutni zahtjev, tj. ne čuva stanje između dva ciklusa zahtjev-odgovor s istim klijentom.

4.1 PostgreSQL

Upotrebljavan pri izradi i korištenju baze podataka, PostgreSQL je napredna relacijska baza podataka otvorenog koda koja podržava i relacijski (SQL) i nerelacijski (JSON) upit [3]. PostgreSQL se koristi kao primarna pohrana podataka ili skladište podataka za mnoge web, mobilne, geoprostorne i analitičke aplikacije. Sustav pri izvođenu transakcija poštuje ACID svojstva (atomičnost, konzistentnost, izolacija i izdržljivost) koja osiguravaju valjanost podataka, čak i pri padu sustava. Jedna ključna razlika između PostgreSQL-a i standardnih sustava relacijskih baza podataka je u tome što PostgreSQL pohranjuje mnogo više informacija u svojim katalozima: ne samo informacije o tablicama i stupcima, već i informacije o vrstama podataka, funkcijama, metodama pristupa itd. Te tablice može mijenjati korisnik, a budući da PostgreSQL svoj rad temelji na tim tablicama, to znači da korisnici mogu proširiti PostgreSQL.

4.2 React.js

React.js je JavaScript biblioteka otvorenog koda koja se koristi za izgradnju korisničkih sučelja, posebno za aplikacije na jednoj stranici i koja je korištena za izradu *front end* dijela ovog sustava. Omogućuje stvaranje velikih web aplikacija koje mogu mijenjati podatke bez ponovnog učitavanja stranice. Također omogućuje stvaranje višekratnih UI komponenti. Glavna svrha *React-a* je da bude brz, skalabilan i jednostavan. U *React*-u se za predložak

umjesto uobičajenog JavaScripta koristi JSX. JSX je jednostavan JavaScript koji omogućuje navođenje HTML-a i koristi sintaksu HTML oznaka za renderiranje višekratnih UI komponenti [4]. Prednost *React-a* je što stvara „in-memory“ strukture podataka u cache memoriji koja izračunava napravljene promjene i zatim ažurira preglednik [4]. To omogućuje posebnu značajku kao da se cijela stranica prikazuje pri svakoj promjeni, dok *React* generira samo komponente koje su se stvarno promijenile. Da bi se komponenta uopće promijenila mora se promijeniti jedno njeno stanje, tj. ugrađeni *React* objekt koji se koristi za sadržavanje podataka ili informacija o komponenti. Stanje se može ažurirati na odgovor poslužitelja, potom kao odgovor na rukovanje događajima ili na promjene *props*-a. Ažuriranje se radi pomoću metode setState() koja stavlja u red sva ažuriranja izvršena u stanju komponente i daje upute *React-u* da ponovno generira komponentu i njezine potomke s ažuriranim stanjem. Primjer inicijalizacije stanja komponente (Kôd 4.1) i njenog ažuriranja je dan u nastavku (Kôd 4.2). Da bismo uopće postavili komponentu na stranicu, potrebno je rezervirano mjesto gdje će se ta komponenta učitati. To rezervirano mjesto je *div* element s identifikatorom *root* unutar HTML dokumenta *index.html* (Kôd 4.3).

```
constructor(props) {
        super(props)
        this.state = {
            pitanje: this.props.pitanje,
            tocan: this.props.tocan,
            hint: this.props.hint,
            novacZaPrikaz: this.props.novacZaPrikaz,
            idzadatka: this.props.idzadatka,
            valuta: "",
            odgovorKorisnika: "",
            novac: [],
            showHint: false,
            taskError: false,
            errorText: "",
            taskSucces: false
        }
    }
```

Kôd 4.1 Incijalizacija stanja komponente

```
  handleChange(event) {
      const { name, value } = event.target
      this.setState({
          [name]: value,
      })
  }
```

Kôd 4.2 Ažuriranje stanja komponente metodom setState()

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <link rel="icon" href="./images/moneyIcon.png" />
    <meta name="viewport" content="width=device-width,
initial-scale=1" />
    <link rel="stylesheet" href="/./styles/main.css">
    <title>Vježbalica</title>
  </head>

  <body>
    <div id="root" style="height:max-content"></div>
  </body>
</html>
```

Kôd 4.3 Mjesto za smještanje komponenti unutar index.html

4.2.1. React axios

Axios je biblioteka koja služi za stvaranje HTTP zahtjeva. U *React* aplikacijama je nekad potrebno doći do podataka iz vanjskih izvora kako bi se stranice mogle normalno prikazati. S *axios*-om je prilično lako doći do takvih podataka. *Axios* dakle, pomaže u dohvaćanju podataka dodajući ih u stanje kako bi se olakšala aplikacija kada god se pojavi zahtjev [5]. Glavna svrha korištenja *axios*-a je dobiti podršku za presretanje zahtjeva i odgovora, pretvorbu podataka u JSON format i njihovu transformaciju. *Axios* se temelji na

obećanjima, što daje mogućnost da se iskoriste prednosti JavaScript asinkronosti (*async* i *await*), što rezultira čitljivijim asinkronim kodom [5]. Dodatno, *axios* je vrlo lako modificirati i prilično je lagan. Primjer slanja jednog zahtjeva u aplikaciji pomoću *axios*-a dan je u nastavku (Kôd 4.4).

```
axios.get(URL + '/loginStudent/getExercise', { withCredentials: false })
      .then(response => {
            if (response.data.err !== undefined) {
                  this.setState({
                        err: true,
                        errorText: response.data.exerciseError
                  })
            } else {
                  const vjezba = {
                        "idvjezbe": response.data.exercise[0].idgrupe,
                        "zadaci": response.data.tasks,
                        "valuta": response.data.exercise[0].valuta,
                        "naziv": response.data.exercise[0].naziv,
                        "brojRijesenih": 0
                  }
                  localStorage.setItem('vjezba2', JSON.stringify(vjezba))
                  this.setState({
                        err: false,
                  })
            }
      })
      .catch(error => {
            console.log(error)
      })
```

Kôd 4.4 Slanje GET zahtjeva pomoću axios

4.3 Express.js

Express.js je radni okvir napisan u JavaScript-u koji se koristi za izradu API-ja i web aplikacija i koji je korišten za izradu *back end* dijela ovog sustava. *Express* pruža mehanizme za rukovanje HTTP zahtjevima s različitim URL putovima (rutama), postavlja uobičajene postavke web aplikacije kao što su port koji će se koristiti za povezivanje, mjesto predložaka

koji se koriste za renderiranje odgovora itd. [6]. Bitan faktor u *Express-u* je usmjeravanje koje se odnosi na određivanje načina na koji aplikacija odgovara na zahtjev klijenta na određenu krajnju točku, tj. na URI (put) i specifičnu metodu HTTP zahtjeva (GET, POST). Svaka ruta može imati jednu ili više funkcija za rukovanje, koje se izvršavaju kada se ruta podudara. Primjer jedne takve funkcije za rukovanje u aplikaciji je dan u nastavku (Kôd 4.5).

```
router.post('/addTask', async function (req, res) {
    let tasks = await getTasks()
    let ima = false
    for (i = 0; i < tasks.length; i++) {
        if (tasks[i].idzad == req.body.id)
            ima = true
        break
    }
    let add = false
    if (!ima) {
        add = await insertTask(req.body.id, req.body.vrsta,
req.body.pitanje, req.body.tocanOdg, req.body.hint,
req.body.tipNovca, req.body.odgovor2, req.body.odgovor3,
req.body.odgovor4, req.body.novacZaPrikaz)
    }
    if (add)
        res.json({
            addingError: undefined,
        })
    else
        res.json({
            addingError: 'Već postoji zadatak s takvim ID-em'
        })

})
```

Kôd 4.5 Funkcija za obradu POST zahtjeva pri dodavanju novog zadatka

Za komunikaciju s PostgreSQL bazom podataka se koristi biblioteka node-postgres. Da bi se mogli izvršavati SQL upiti nad bazom podataka potrebno je prvo definirati postavke za pristup programskim konfiguriranjem bazena (en. *pool*) ili klijenta s informacijama o vezi (Kôd 4.6).

```
const {Pool} = require('pg');
const pool = new Pool({
    user: 'dbuser',
    host: 'dbhost',
    database: 'dbname',
    password: 'dbpassword',
    port: 5432
});
```

Kôd 4.6 Definiranje postavki za pristup bazi podataka

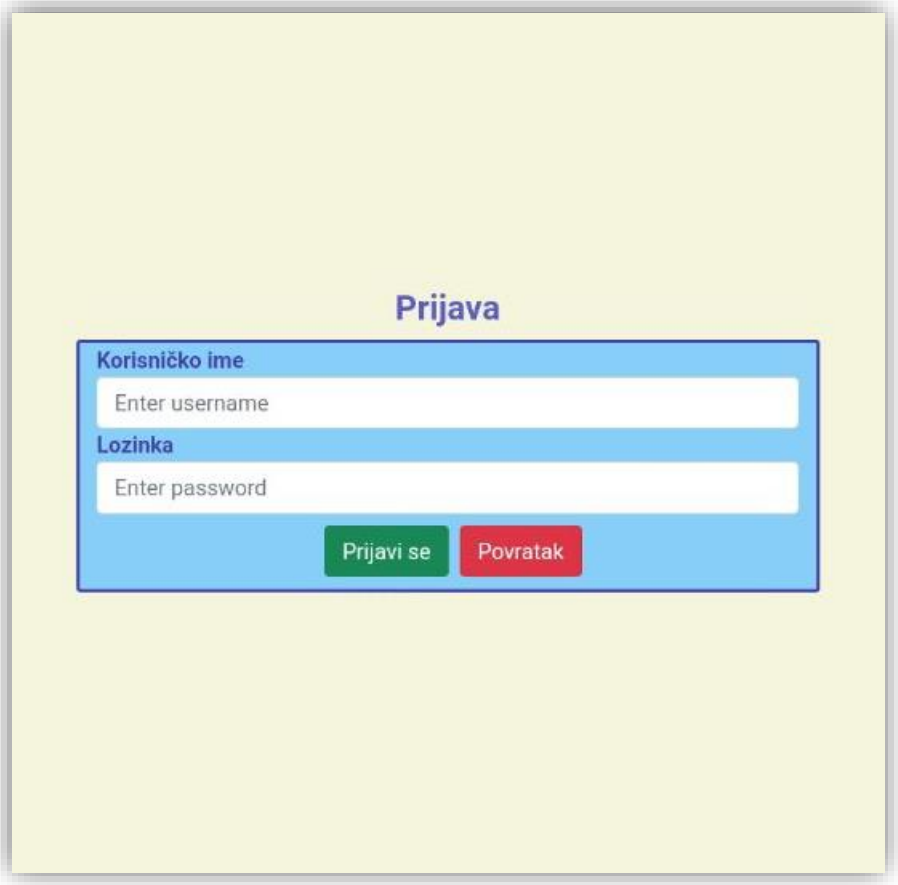
4.4 Git i Github

Git je distribuirani sustav za praćenje promjena u bilo kojem skupu datoteka, koji radi na principu grananja i koji se obično koristi za koordinaciju rada između programera koji zajednički razvijaju izvorni kod [7]. Sastoji se od udaljenog repozitorija koji se najčešće nalazi na nekoj *Git* platformi u oblaku i lokalnih kopija tog repozitorija na računalima programera [7]. Jedna od takvih *Git* platformi u oblaku je i *GitHub* koji je korišten za svrhe ove aplikacije. Nudi funkcionalnost distribuirane kontrole verzija i upravljanja izvornog koda. Omogućuje kontrolu pristupa i nekoliko značajki suradnje kao što su praćenje bug-ova, zahtjevi za značajkama, upravljanje zadacima itd.

5. Opis aplikacije

Prije korištenja aplikacije potrebno je u bazu podataka unijeti podatke o školama, razredima i nastavnicima koji u njima rade kako bi se nastavnici mogli prijavljivati u sustav i zatim ga normalno koristiti. Također je potrebno unijeti i sav novac određene valute koja će se koristiti pri rješavanju vježbi kako bi sustav u zadacima mogao prikazivati slike novčanica i kovanica ako to bude potrebno. Na početnoj stranici ispod gumba „*START*“ se nalazi link za nastavnike za prijavu u sustav (Slika 5.1).

Slika 5.1 Početna stranica



Slika 5.2 Forma za prijavu nastavnika

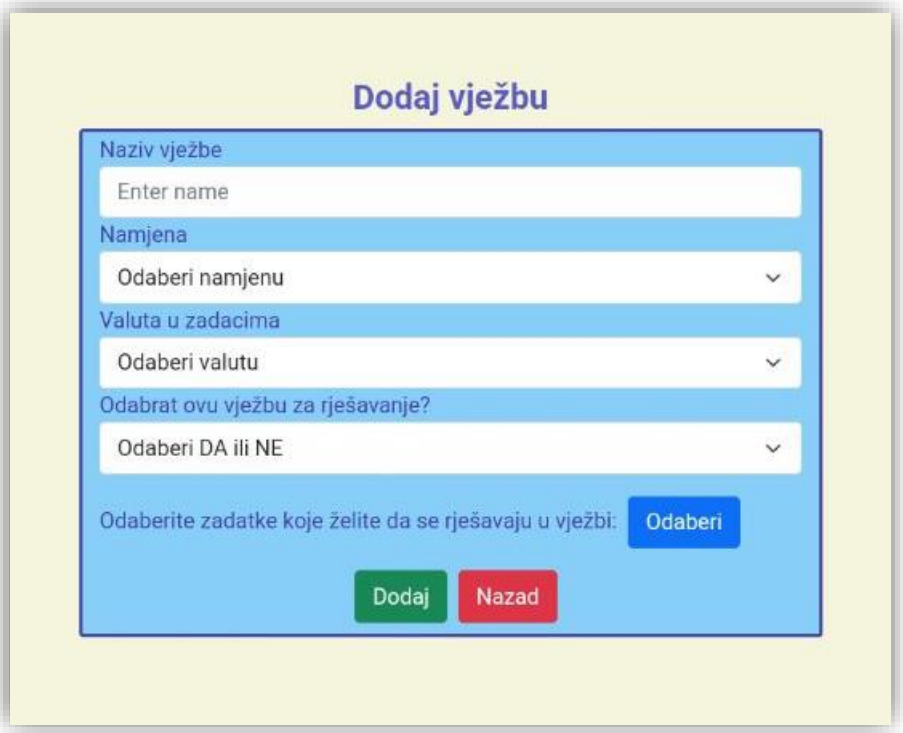
Nakon prijave u sustav nastavniku se prikazuje njegova profilna stranica (Slika 5.3). Na njoj su mu prikazani podaci o kreiranim zadacima i vježbama, te popis učenika iz njegovih razreda. Na vrhu profila se nalazi naziv škole u kojoj nastavnik radi te gumb „*OPCIJE*“. Klikom na taj gumb, nastavniku se nude opcije za dodavanje novih učenika, zadataka, vježbi, te mogućnost odjave.

Slika 5.3 Nastavnikova profilna stranica

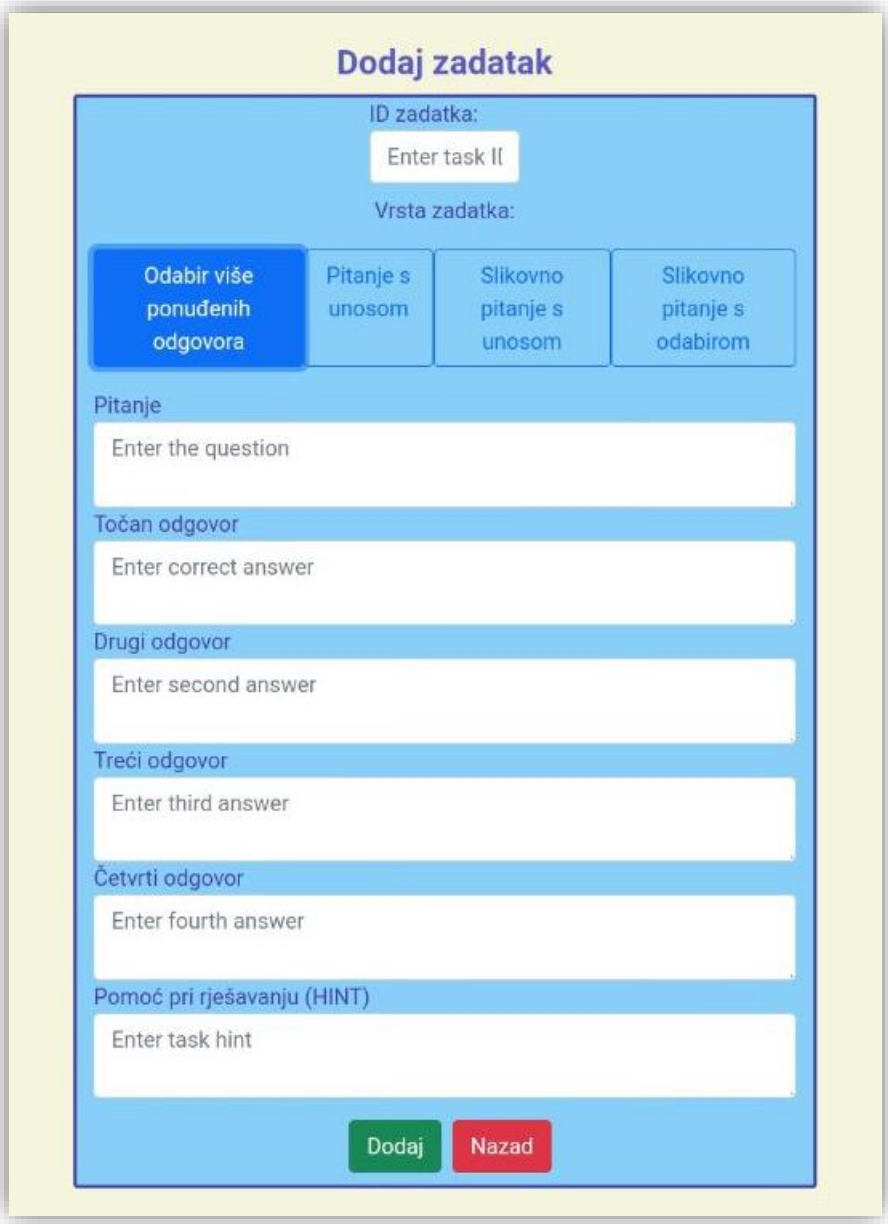
Kod dodavanja učenika (Slika 5.4), uz ime i prezime, potrebno je u padajućem izborniku odabrati i razred u koji učenik ide. Na stranici za dodavanje novih zadataka (Slika 5.5) je potrebno odabrati vrstu zadatka, te se ovisno o odabranoj vrsti prikazuju različita polja za unos. Potrebno je ispuniti sva polja za unos, te šifra zadatka mora biti različita od šifri već postojećih zadataka ili inače neće biti moguće dodati novi zadatak. Vrste zadataka koje nastavnik može dodati su: unos, slikovnoUnos, odabir, slikovnoOdabir. Što se tiče dodavanja novih vježbi za rješavanje (Slika 5.6), bitno je pomoću padajućeg izbornika izabrati valutu koja će se koristiti u vježbi i otvaranjem pop-up prozora pomoću gumba „*Odaberi*“ odabrati zadatke za rješavanje. Također, pomoću padajućeg izbornika za pokretanje vježbe se odabire hoće li se vježba pokrenuti učenicima za vježbanje ili ne. Potrebno je odabrati i namjenu vježbe, odnosno koristi li se za ispitivanje ili samo za vježbanje. Trenutno se ta stavka ne koristi, no s njom je moguće nadograditi sustav tako da

se vode različite statistike za ispite i vježbe, te da se vježbe mogu rješavati proizvoljan broj puta, dok ispiti imaju samo jedan pokušaj rješavanja.

Slika 5.4 Forma za dodavanje novih učenika

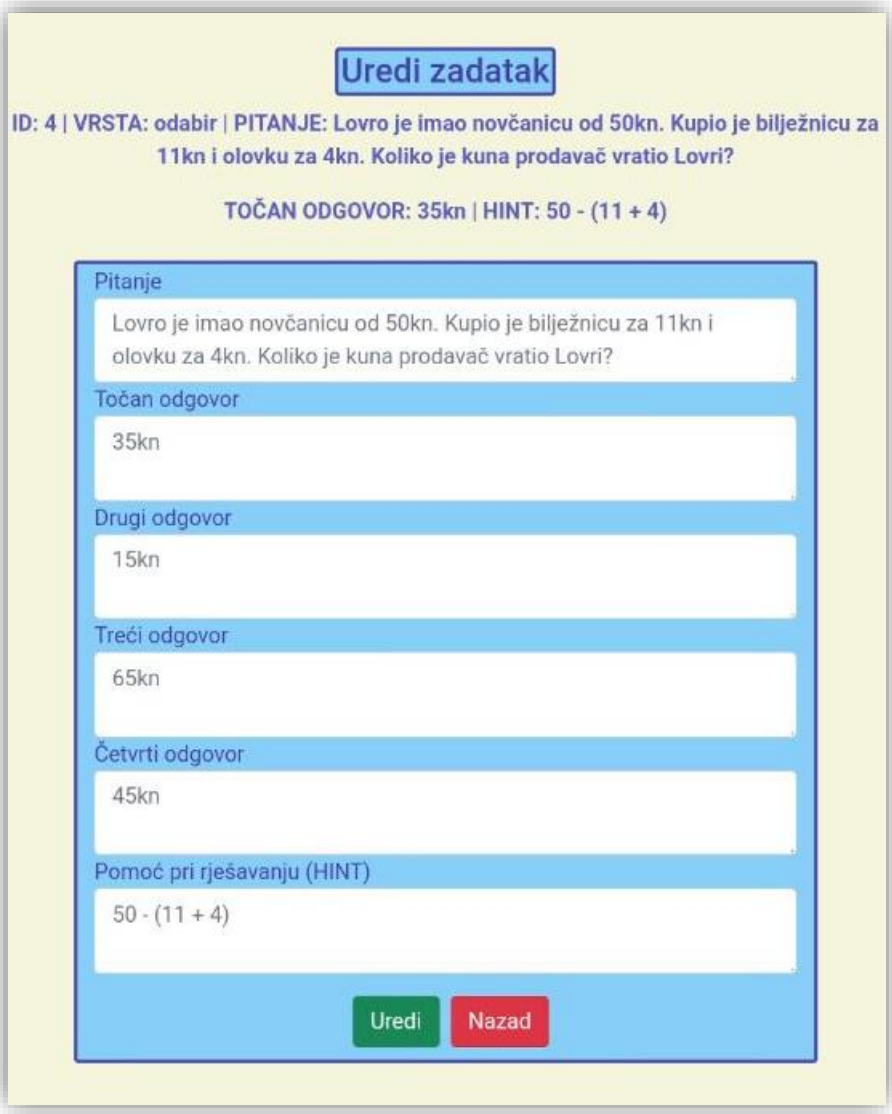


Slika 5.5 Forma za dodavanje nove vježbe



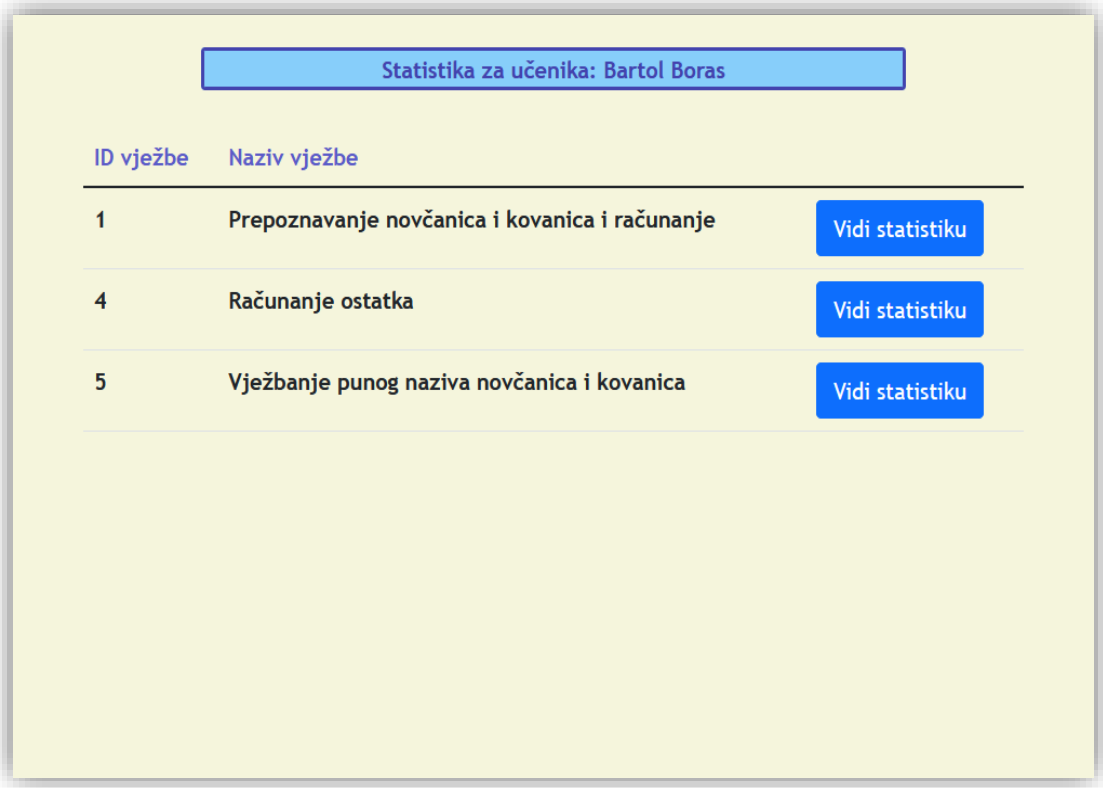
Slika 5.6 Forma za dodavanje novog zadatka vrste odabir

Svi podaci o vježbi, zadacima i učenicima se mogu uređivati. To je omogućeno gumbom „*Uredi*“ koji se nalazi na nastavnikovoj profilnoj stranici u svakom retku uz dotičnu vježbu, zadatak ili učenika. Klik na taj gumb vodi na stranicu za uređivanje (Slika 5.7) gdje se ispod naslova i u poljima za unos nalaze prijašnje informacije o odabranoj stavki. Potrebno je ispuniti sva polja kako bi se stavka uredila. Prilikom uređivanja vježbi neće biti moguće postaviti vježbu da se pokrene za rješavanje ako je već prije bila postavljena neka druga vježba za rješavanje.



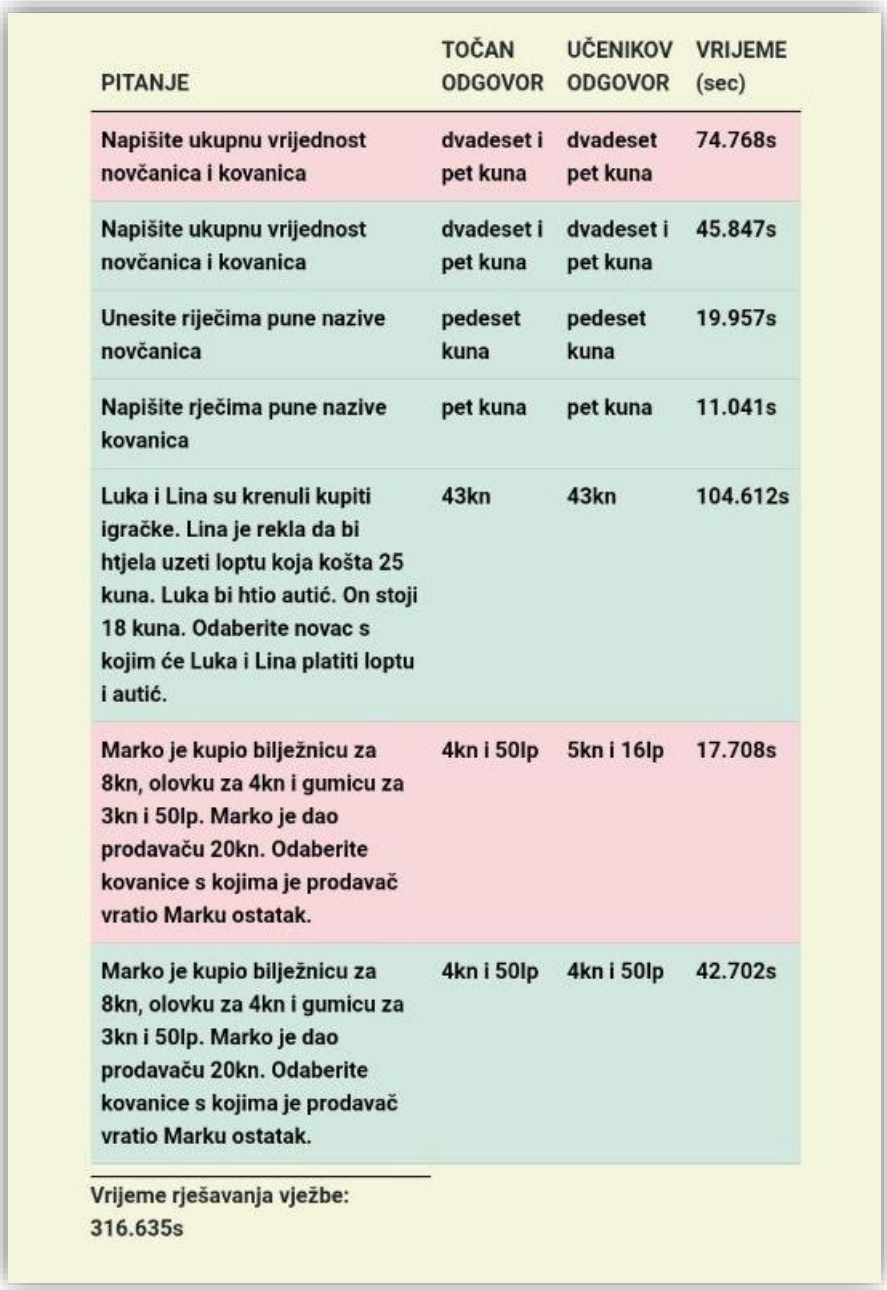
Slika 5.7 Forma za uređivanje zadatka

Na nastavnikovom popisu učenika se u svakom retku nalazi još i gumb „*Statistika*“. Klik na taj gumb vodi na listu statistika svih vježbi koje je odabrani učenik u tom trenutku odradio (Slika 5.8). Ako učenik do tada nije uradio nijednu vježbu, umjesto liste statistika biti će prikazana poruka koja obavještava nastavnika o istome.



Slika 5.8 Popis statistika svih vježbi koje je odabrani učenik uradio

Klik na gumb „*Vidi statistiku*“ vodi nastavnika na novu stranicu na kojoj mu je prikazana statistika za odabranu vježbu (Slika 5.9). Statistiku čine svi odgovori koje je učenik unio u dotičnoj vježbi, bilo točni ili netočni. Točni odgovori su označeni zelenom bojom, a netočni crvenom bojom. Osim popisa odgovora, statistiku čini i vrijeme rješavanja svakog zadatka izraženo u sekundama, odnosno vrijeme koje je bilo potrebno učeniku da unese odgovor. Na kraju statistike se nalazi ukupno vrijeme rješavanja vježbe, također izraženo u sekundama.



PITANJE	TOČAN ODGOVOR	UČENIKOV ODGOVOR	VRIJEME (sec)
Napišite ukupnu vrijednost novčanica i kovanica	dvadeset i pet kuna	dvadeset pet kuna	74.768s
Napišite ukupnu vrijednost novčanica i kovanica	dvadeset i pet kuna	dvadeset i pet kuna	45.847s
Unesite riječima pune nazive novčanica	pedeset kuna	pedeset kuna	19.957s
Napišite rječima pune nazive kovanica	pet kuna	pet kuna	11.041s
Luka i Lina su krenuli kupiti igračke. Lina je rekla da bi htjela uzeti loptu koja košta 25 kuna. Luka bi htio autić. On stoji 18 kuna. Odaberite novac s kojim će Luka i Lina platiti loptu i autić.	43kn	43kn	104.612s
Marko je kupio bilježnicu za 8kn, olovku za 4kn i gumicu za 3kn i 50lp. Marko je dao prodavaču 20kn. Odaberite kovanice s kojima je prodavač vratio Marku ostatak.	4kn i 50lp	5kn i 16lp	17.708s
Marko je kupio bilježnicu za 8kn, olovku za 4kn i gumicu za 3kn i 50lp. Marko je dao prodavaču 20kn. Odaberite kovanice s kojima je prodavač vratio Marku ostatak.	4kn i 50lp	4kn i 50lp	42.702s

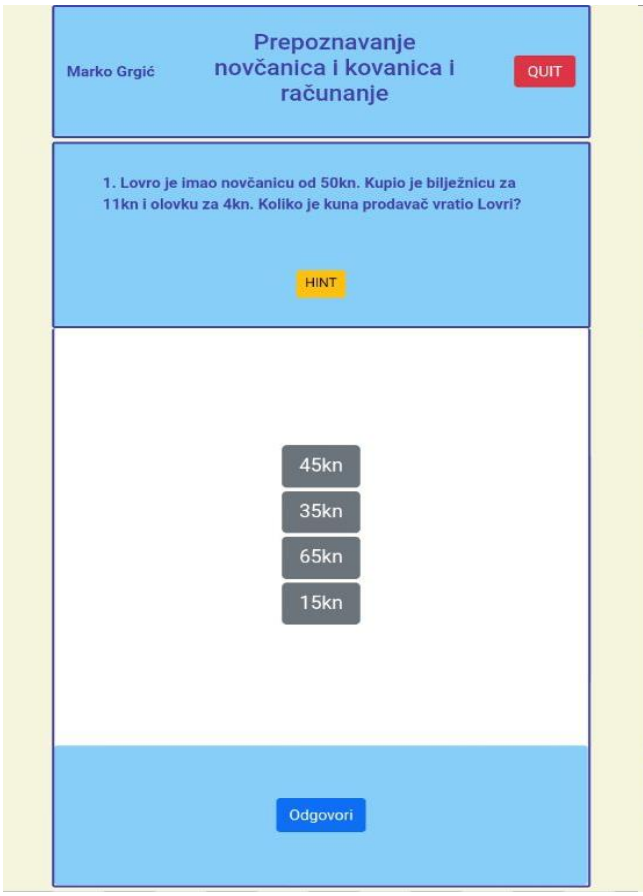
Vrijeme rješavanja vježbe:
316.635s

Slika 5.9 Statistika vježbe

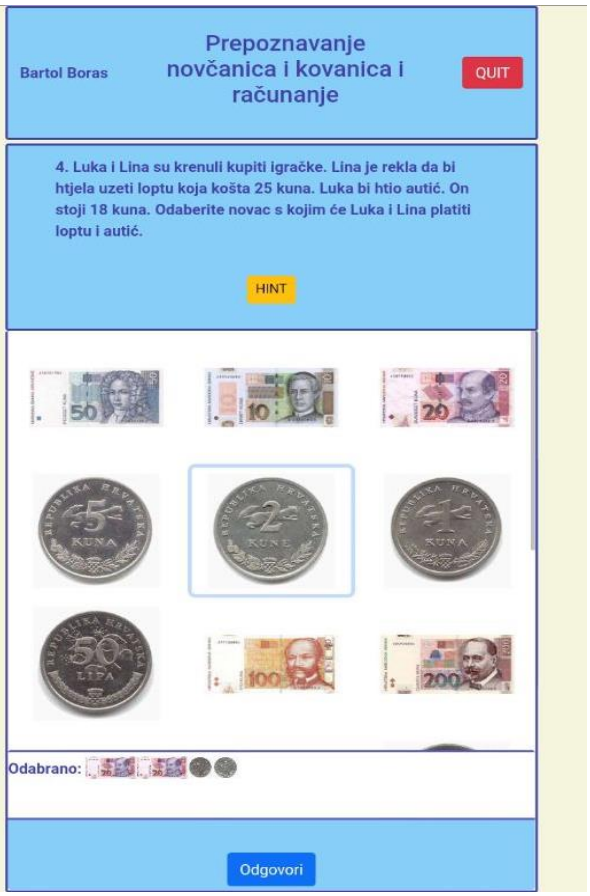
Nakon što je nastavnik unio sve podatke potrebne za rješavanje vježbe, učenici mogu započeti rješavanje vježbe klikom na gumb „*START*“ koji se nalazi na početnoj stranici. No, prije početka rješavanja moraju odabrati svoju školu, razred i ime kako bi se vježba mogla povezati s učenikom. Odabir je omogućen u obliku forme s padajućim izbornicima (Slika 5.10).

Slika 5.10 Forma za prijavu učenika prije početka vježbe

Nakon što je odabrao svoje podatke, učenik može započeti rješavanje vježbe klikom na gumb „*KRENI*“. Pri tome mu se otvara nova stranica na kojoj mu je prikazan prvi od onoliko zadataka koliko ih vježba sadrži. Redoslijed zadataka u svakoj vježbi je takav da se prvo pojavljuju zadaci u kojima učenik mora naučiti prepoznati novčanice i kovanice te naučiti njihove nazive, a tek nakon što točno riješi takve vrste zadataka pojavljivat će mu se zadaci s računanjem i odabirom novčanica i kovanica. Kako bi uspješno odradio vježbu, učenik ne može prijeći na sljedeće pitanje sve dok ne odgovori točno na trenutno pitanje. Ako učenik ima problema s rješavanjem zadatka, pomoć može potražiti ispod pitanja klikom na žuti gumb „*HINT*“. Također može u bilo kojem trenutku napustiti vježbu klikom na gumb „*QUIT*“, međutim tada će se vježba računati kao riješena i neće joj moći ponovo pristupiti. Primjeri rješavanja nekih zadataka dani su na slikama (Slika 5.11, Slika 5.12).



Slika 5.12 Zadatak s ponuđenim odgovorima



Slika 5.11 Zadatak s odabirom novčanica i kovanica

Nakon što unese odgovor i pritisne na gumb „*Odgovori*“, šalje se zapis na poslužitelj i sprema se u bazu podataka u tablicu *logs*. Zapis se sastoji od pitanja, učenikovog odgovora, točnog odgovora za taj zadatak, šifre vježbe i trajanja koje je bilo potrebno da se unese odgovor. Istovremeno se učeniku prikaže kratka poruka koja ga obavijesti je li točno ili netočno odgovorio. Riješivši točno sva pitanja, vježba se završava i učenik biva preusmjeren na početnu stranicu.

Zaključak

Tehnologijom podržano učenje je vrlo aktualno i aktivno područje, a tehnologija se sve više primjenjuje u raznim obrazovnim procesima. Primjena tehnologije u učenju je donijela mnoge benefite, pa tako i povećanje zainteresiranosti učenika za predmet i lakše praćenje učenikovog znanja, što je iskorišteno i u ovom završnom radu.

Cilj ovog rada je bio implementirati sustav koji će učenicima učenje i vježbanje novčanica i kovanica učiniti lakšim i zanimljivijim, a nastavnicima omogućiti lakše i adekvatnije praćenje učenikovih znanja i sposobnosti. Početne ideje, poput mogućnosti interaktivne vježbe više vrsta zadataka, korištenja više valuta i slikovnih prikaza, vođenje statistike za svakog učenika i vježbe koje je obavio su uspješno implementirane.

U radu postoji prostor za dodatna proširenja. Uz vježbe se može implementirati i mogućnost sastavljanja i pisanja ispita, dodavanje više vrsta zadataka, poboljšanje statistike uspoređivanjem rezultata s ostalim učenicima itd. Na razvojnoj strani se može poboljšati sigurnost lozinki pri prijenosu na poslužitelj računanjem sažetaka, te koristiti neka druga biblioteka za upite nad bazom podataka kako bi se smanjila ranjivost na *SQL injection* napade.

Literatura

[1] Predavanja s kolegija Programsko inženjerstvo – Fakultet elektrotehnike i računarstva, Sveučilište u Zagrebu

[2] Predavanja s kolegija Razvoj programske potpore za web – Fakultet elektrotehnike i računarstva, Sveučilište u Zagrebu

[3] PostgreSQL

poveznica – https://www.postgresqltutorial.com/postgresql-getting-started/what-is-postgresql/

[4] React

poveznica – https://www.c-sharpcorner.com/article/what-and-why-reactjs/

[5] React axios

poveznica – https://www.javatpoint.com/react-axios

[6] Express.js

poveznica – https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express_Nodejs/Introduction

[7] Git

poveznica – https://docs.github.com/en/get-started/using-git/about-git

Sažetak

Naslov

Web aplikacija za vježbanje prepoznavanja i upotrebe novčanica i kovanica

Sažetak

Tehnologijom podržano učenje nudi različiti oblik stjecanja znanja od tradicionalnog, ali vodi k boljem i manje napornom razumijevanju gradiva kao i učinkovitijoj i jednostavnijoj analizi učenikovih znanja. Iskorištene su tehnološke prednosti kako bi se osigurala veća fleksibilnost za nastavnika i učenika, te kako bi se postigli ciljevi učenja primjereni potrebama pojedinih učenika.

Sve te prednosti, tematski vezano u gradivo učenja novčanica i kovanica, je objedinila napravljena web aplikacija za učenje i vježbanje novčanica i kovanica. Pomoću tehnologija *React.js* i *Express.js* je napravljena izrada i rješavanje vježbi u aplikaciji, te vođenje statistike, a za spremanje i obradu podataka je korišten *PostgreSQL*.

Ključne riječi

React, Express, PostgreSQL, Novčanice i kovanice, Vježba, Učenje

Summary

Title

Web application for practicing the recognition and use of banknotes and coins

Abstract

Technology-supported learning offers a different form of knowledge acquisition than traditional but leads to a better and less strenuous understanding of the material as well as a more efficient and simple analysis of students' knowledge. Technological advantages have been used to ensure greater flexibility for teachers and students, and to achieve learning goals appropriate to the needs of individual students. All these advantages, thematically related to the material of learning banknotes and coins, were combined by a web application for learning and practicing banknotes and coins. With the help of React.js and Express.js technologies, the creation and solving of exercises in the application as well as keeping statistics, was done. PostgreSQL was used to save and process data.

Key words

React, Express, PostgreSQL, Banknotes and coins, Exercise, Learning