Web portal za edukacije

Čačija, Ana

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU

FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Sveučilišni studij

WEB PORTAL ZA EDUKACIJE

Diplomski rad

Ana Čačija

Osijek, 2023.

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

Obrazac D1: Obrazac za imenovanje Povjerenstva za diplomski ispit

Osijek, 08.07.2023.

Odboru za završne i diplomske ispite

Imenovanje Povjerenstva za diplomski ispit

Ime i prezime Pristupnika:	Ana Čačija
Studij, smjer:	Diplomski sveučilišni studij Računarstvo
Mat. br. Pristupnika, godina upisa:	D-1044R, 06.10.2019.
OIB studenta:	30610059925
Mentor:	prof. dr. sc. Krešimir Nenadić
Sumentor:	,
Sumentor iz tvrtke:	
Predsjednik Povjerenstva:	izv. prof. dr. sc. Zdravko Krpić
Član Povjerenstva 1:	prof. dr. sc. Krešimir Nenadić
Član Povjerenstva 2:	doc. dr. sc. Krešimir Romić
Naslov diplomskog rada:	Web portal za edukacije
Znanstvena grana diplomskog rada:	**Programsko inženjerstvo (zn. polje računarstvo)**
Zadatak diplomskog rada:	Potrebno je modelirati i izraditi bazu podataka koja će se koristiti za potrebe web portala za edukacije. Omogućiti registraciju korisnika koji će imati pristup organiziranim materijalima za potrebe raznovrsnih edukacija. Opisati postupak izrade portala, načine korištenja pojedinih web tehnologija u postupku izrade te način funkcioniranja web portala. Tema rezervirana: Ana Čačija
Prijedlog ocjene pismenog dijela ispita (diplomskog rada):	Izvrstan (5)
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: 3 bod/boda Postignuti rezultati u odnosu na složenost zadatka: 3 bod/boda Jasnoća pismenog izražavanja: 3 bod/boda Razina samostalnosti: 3 razina
Datum prijedloga ocjene od strane mentora:	08.07.2023.

Potvrda mentora o predaji konačne verzije rada:	*Mentor elektronički potpisao predaju konačne verzije.*
	Datum:

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

IZJAVA O ORIGINALNOSTI RADA

Osijek, 18.07.2023.

Ime i prezime studenta:	Ana Čačija
Studij:	Diplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	D-1044R, 06.10.2019.
Turnitin podudaranje [%]:	4

Ovom izjavom izjavljujem da je rad pod nazivom **: Web portal za edukacije**

izrađen pod vodstvom mentora prof. dr. sc. Krešimir Nenadić

i sumentora ,

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis studenta:

Sadržaj

1. UVOD

U današnjem, tehnološki naprednom svijetu, stjecanje znanja vrlo je brzo postalo izrazito važno za sveukupni uspjeh i napredovanje pojedinca. Pravilno usmjerena edukacija ključni je temelj za istraživanje novih mogućnosti, te stjecanje dubljeg razumijevanja različitih područja interesa. Ubrzanim razvojem digitalnih proizvoda i tehnologije, javlja se potreba za što pristupačnijim načinima učenja kako bi se obrazovno iskustvo pojedinca prilagodilo njegovim osobnim potrebama, a razina znanja povećala sukladno obrazovnim sadržajima. Uzimajući važnost obrazovanja u obzir, izrada aplikacije koja korisnicima omogućava pristup raznovrsnim tečajevima, te doticaj s različitim područjima znanja, osigurava razvoj višestrukih intelektualnih vještina pojedinca, te primjenom stečenih vještina omogućava unaprjeđenje kritičkog razmišljanja i rješavanja kompleksnih problema. Web portal razvijen u ovome radu, prethodno napisano predstavlja interakcijom između ključnih uloga predavača, studenata i administratora, koji različitim odgovornostima doprinose osiguranju kvalitetnog sadržaja, te jednostavnog stjecanja znanja. Koristeći svoje stručno iskustvo, predavači kreiranjem sadržaja lekcija unutar tečajeva upravljaju njegovom relevantnošću pružajući studentima smjernice za razumijevanje složenih koncepata i tema. Nakon uspješno položenih završnih ispita, tečajevi studentima osiguravaju službeni dokaz potvrde stečenih znanja u obliku certifikata, koji se u budućnosti mogu primijeniti u svrhu validacije akademskog postignuća pojedinca.

Poglavlja koja slijede nakon uvoda, detaljnije opisuju važnost obrazovanja putem interneta, te omogućuju pregled sličnih postojećih rješenja (drugo poglavlje). Uz to, trećim poglavljem opisani su alati i tehnologije korišteni prilikom izrade ove aplikacije, a pregledom četvrtog poglavlja temeljito je opisan postupak izrade platforme uključujući faze dizajna, te implementacije funkcionalnosti potrebnih za učinkovitu izvedbu aplikacije. Peto poglavlje sadrži prikaz korisničkih zaslona te opisa za svaku od stranica dostupnu u aplikaciji. Zadnje poglavlje, zaključak, uz pregled postignutih rezultata ovog rada, navodi i mogućnosti daljnjeg razvoja aplikacije u smislu funkcionalnosti koje bi, ukoliko implementirane, omogućile značajan razvoj aplikacije u budućnosti.

1.1 Zadatak diplomskog rada

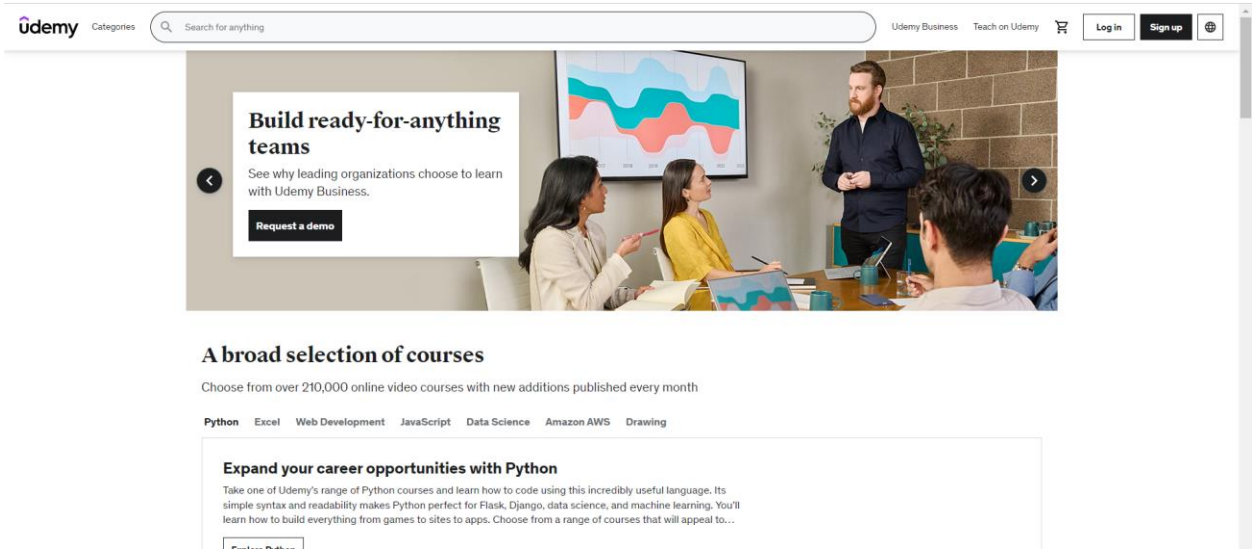
Potrebno je modelirati i izraditi bazu podataka koja će se koristiti za potrebe web portala za edukacije. Omogućiti registraciju korisnika koji će imati pristup organiziranim materijalima za potrebe raznovrsnih edukacija. Opisati postupak izrade portala, načine korištenja pojedinih web tehnologija u postupku izrade te način funkcioniranja web portala.

2. PREGLED PODRUČJA

Suvremeno digitalno doba omogućilo je edukaciju putem interneta kao neizostavno sredstvo u procesu usvajanja znanja i vještina. Brojne aplikacije nude mogućnosti sudjelovanja u raznovrsnim tečajevima pružajući korisnicima različite metode učenja te proširenu domenu tematskih područja, a istovremeno prilagođavajući se rasporedu korisnika. Ovaj rad, uz prethodno navedene stavke, pruža intuitivno korisničko sučelje koje čini sveukupno korisničko iskustvo ugodnim, a korisnika fokusiranog isključivo na glavni cilj – edukaciju. U nastavku se mogu pronaći već postojeća slična rješenja s tržišta edukacijskih aplikacija, te načini na koji su ista pristupila zadatku obrazovanja korisnika putem interneta.

2.1. Udemy

Udemy, jedna od vodećih platformi za tečajeve putem interneta, nudi mnogobrojne tečajeve iz različitih područja znanja. Ova platforma pruža mogućnosti poput učenja programiranja, poslovnih vještina, jezičnog učenja, i slično. Omogućava korisnicima efikasno pretraživanje tečajeva, pristup video lekcijama, te sudjelovanje u interaktivnim aktivnostima. Početna stranica platforme Udemy prikazana je na slici 2.1..

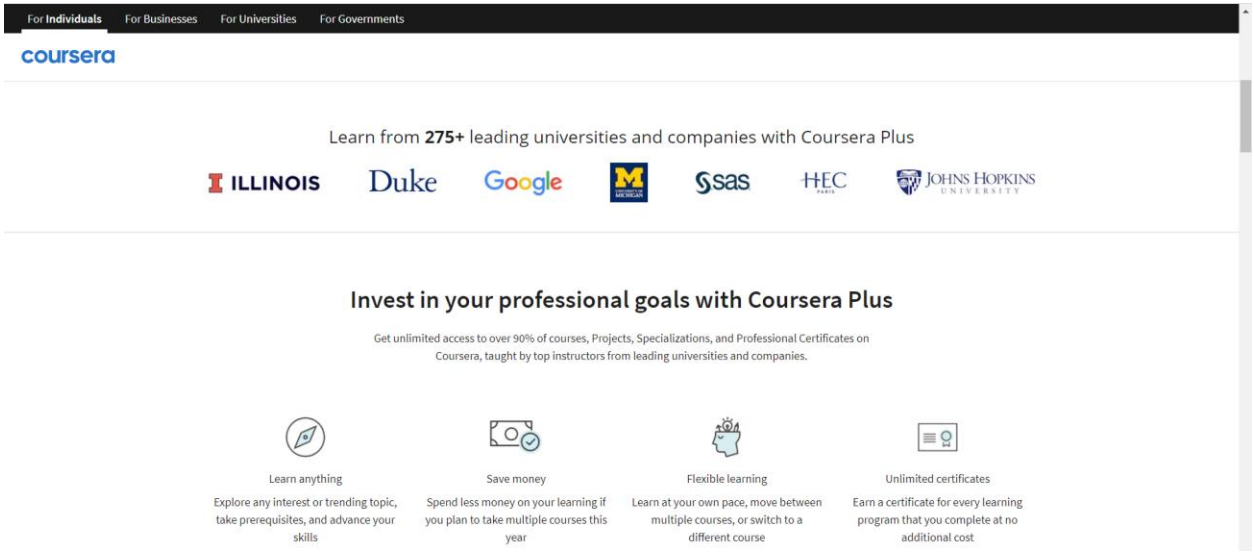


Slika 2.1. Stranica Udemy, izvor: [1]

2.2. Coursera

Surađujući s vodećim institucijama i sveučilištima diljem svijeta, Coursera korisnicima osigurava visokokvalitetne sadržaje tečajeva, uz akademska priznanja poput certifikata i diploma za daljni profesionalni uspjeh i razvoj. Tečajevi Coursere često su strukturirani u modulima uključujući video lekcije te provjere znanja. Poput prethodno opisanog Udemy-ja, Coursera nudi korisnicima

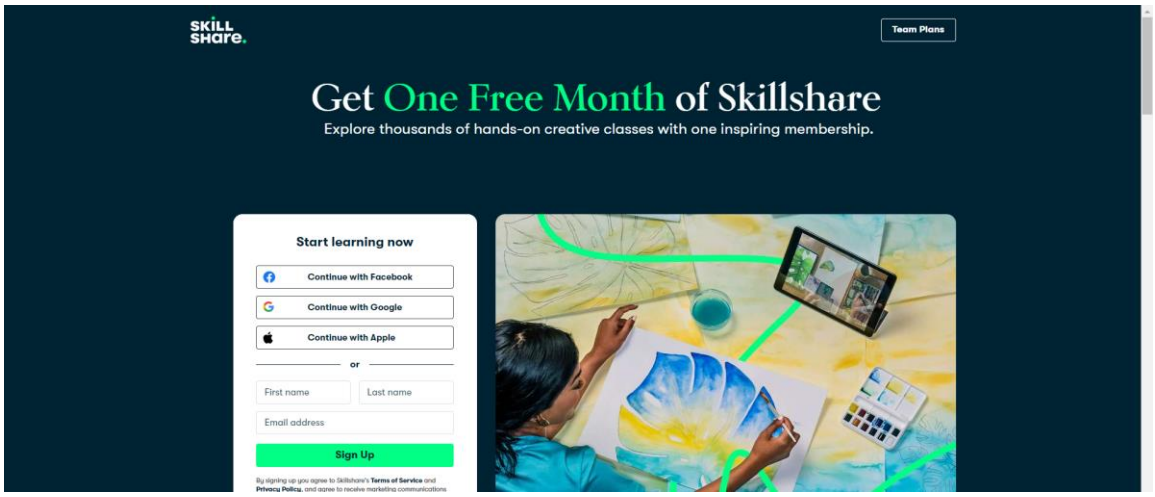
mogućnost interakcije kroz međusobno ocjenjivanje i komentiranje predanih radova, te na taj način poboljšava razvoj vještina svih sudionika. Početna stranica platforme Coursera prikazana je slikom 2.2..



Slika 2.2. Stranica Coursera, izvor: [2]

2.3. Skillshare

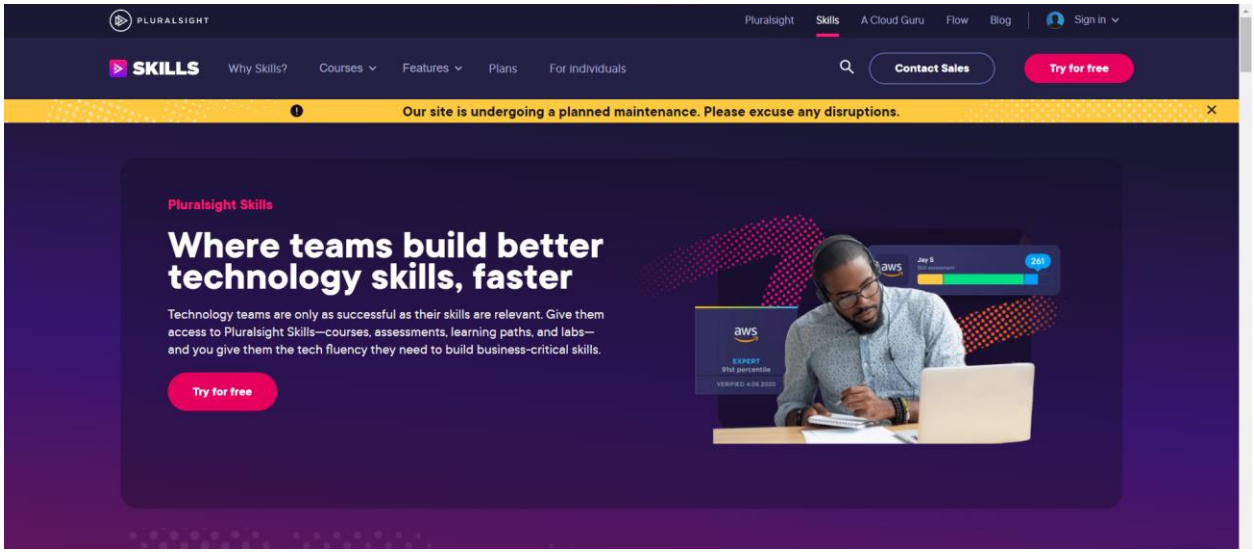
Stavljajući fokus na kreativne discipline, Skillshare korisnicima omogućuje pristup raznim tečajevima iz područja kao što su grafički dizajn i ilustracija, fotografija, marketing te ostale slične vještine. Uz pretplatničke modele za studente i profesore, Skillshare nudi i proširene mogućnosti za timove, koje omogućuju dodatne pogodnosti svim članovima tima. Na slici 2.3. prikazana je početna stranica platforme Skillshare.



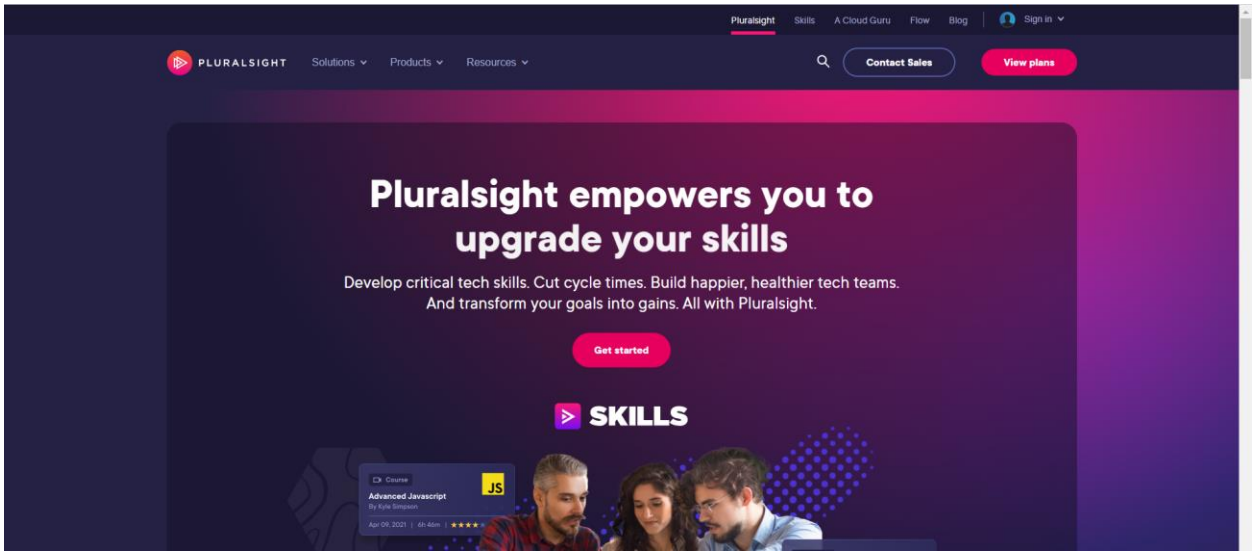
Slika 2.3. Stranica Skillshare, izvor: [3]

2.4. Pluralsight

Pluralsight je napredna platforma za učenje primarno fokusirana na razvoj tehnoloških vještina pojedinca. Područja poput programiranja, razvoja programskih rješenja i sigurnosti podataka neka su od tehnoloških disciplina u širokoj ponudi Pluralsighta. Proizvod Pluralsighta naziva Skills, prikazan slikom 2.4., pruža korisnicima dodatne mogućnosti poput detaljnih instrukcija, izrade projekata i praktičnih zadataka kako bi korisnici kroz primjenu znanja ubrzali i unaprijedili svoj proces učenja. Slika 2.5. prikazuje početnu stranicu platforme Pluralsight.



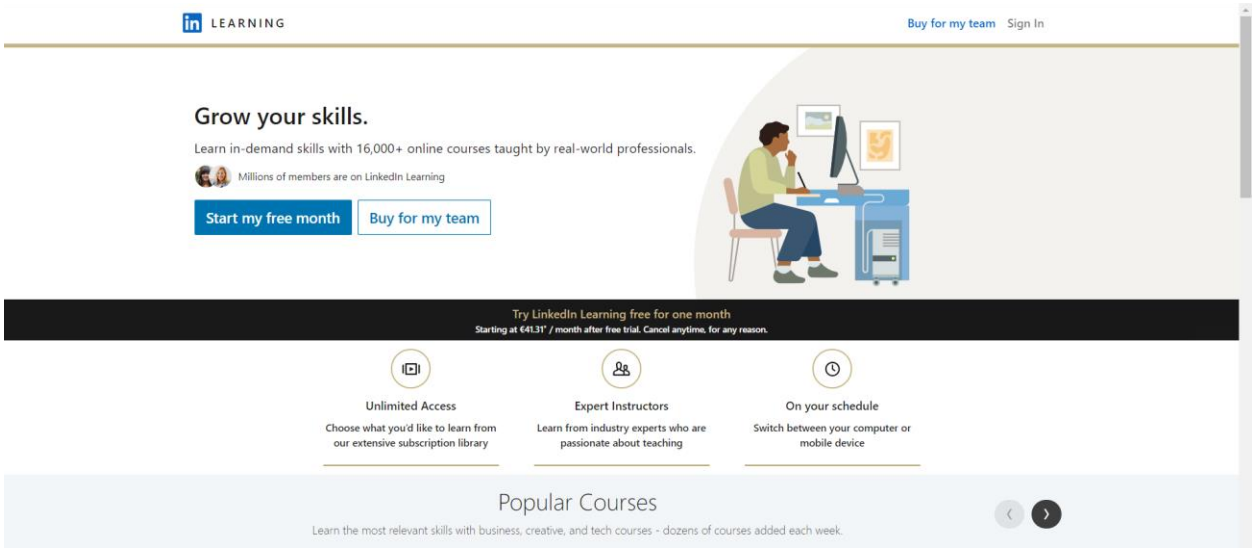
Slika 2.4. Početna stranica platforme Skills, izvor: [4]



Slika 2.5. Početna stranica platforme Pluralsight, izvor: [5]

2.5. LinkedIn Learning

Kao vodeća profesionalna društvena mreža koja povezuje profesionalce iz različitih industrija omogućujući korisnicima izgradnju profesionalne mreže te pronalaženje poslovnih prilika, LinkedIn u svojoj ponudi ističe i alate poput LinkedIn Learninga u svrhu daljnjeg obrazovanja i razvoja vještina korisnika. LinkedIn Learning, uz osiguravanje visokokvalitetnih tečajeva iz raznih područja poput tehnologije, kreativnosti, te poslovnih vještina, nudi jednostavnu integraciju s LinkedIn platformom. Na taj način, korisnicima je omogućen prikaz novo stečenih vještina na profilima što osigurava otvaranje novih mogućnosti povezivanja korisnika s profesionalcima iz iste industrije. Početna stranica platforme LinkedIn Learning prikazana je slikom 2.6..



Slika 2.6. Početna stranica platforme LinkedIn Learning, izvor: [6]

3. TEHNOLOGIJE KORIŠTENE U IZRADI RADA

Pri izradi ove web aplikacije, korištene su razne tehnologije i alati, čije će se ključne karakteristike analizirati i opisati u ovom poglavlju, te obrazložiti razlozi odabira istih prilikom razvoja ove aplikacije. Poseban naglasak u ovom poglavlju stavljen je na pružanje detaljnog objašnjenja načina na koji svaka od korištenih tehnologija doprinosi korisnim funkcionalnostima i unapređenju performansi web aplikacija. Svako od potpoglavlja pruža konkretne primjere strukture i sintakse određenog programskog jezika s ciljem pružanja jasnog uvida u praktičnu primjenu spomenutih tehnologija, osiguravajući bolje razumijevanje primjene korištenih tehnologija u praksi.

3.1. Opisni jezik HTML

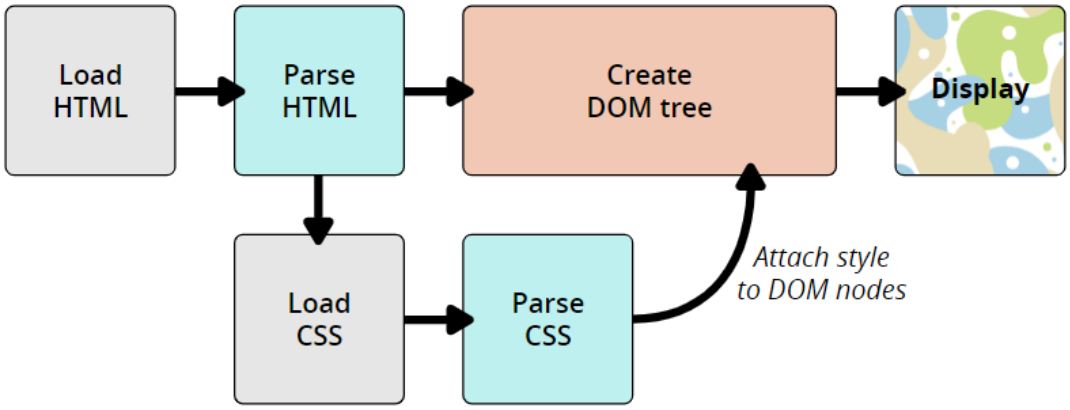
HTML (skraćeno od *Hypertext Markup Language*), opisni je jezik koji definira strukturu web stranica. Prema [7], riječ „Hypertext“ označava tekst, često s uključenim podacima poput slika, organiziran na način koji povezuje slične elemente. Riječ „Markup“ označava vodič za stil tiskanja u formatu tiskane ili digitalne kopije, a riječ „Language“ predstavlja jezik koji računalni sustav razumije te koristi kako bi protumačio naredbe. HTML datoteke (.htm ili .html) sastoje se od HTML elemenata, koji sadrže oznake što internetskom pregledniku označava početak i kraj elementa. Prilikom korištenja oznaka, razlikuju se početne i zatvarajuće oznake, dok se sadržaj elementa nalazi između početne i završne oznake. Jedan od najbitnijih tagova, <html>, predstavlja korijen svakog HTML dokumenta, te služi kako bi se u njega upisivali svi ostali HTML elementi [8]. Uz oznaku <html>, bitno je istaknuti i oznake <head> i <body>, koje također imaju vrlo bitne funkcije u HTML dokumentima. Naime, oznaka <head> služi kako bi se unutar nje upisali metapodaci, iako se isti ne prikazuju na web stranici, ali se koriste od strane internetskih preglednika, te pretraživača. Prema [9], uobičajeni elementi uključeni unutar oznake <head> sadrže naslov, poveznice vanjskih korištenih resursa, metapodatke poput ključnih riječi te autora, te eventualni programski kod koji se treba izvesti na stranici. Oznaka <body> definira tijelo dokumenta te sadrži sav sadržaj HTML dokumenta, kao što su zaglavlja, paragrafi, fotografije i tablice, i slično [10]. U HTML dokumentu može se nalaziti samo po jedna od svake spomenute oznake. Slikom 3.1. moguće je vidjeti HTML dokumenta zajedno s njegovim osnovnim oznakama.

```html
<html>
    <head>
        <meta charset="utf-8">
        <meta name="viewport" content="width=device-width, initial-scale=1">
        <title>Courses</title>
        <link href="http://fonts.cdnfonts.com/css/sofia-pro" rel="stylesheet">
       <script>
        (function(){
          window.Laravel = {
            csrfToken: '{{ csrf_token() }}'
          };
        })();
       </script>
    </head>
  <body>
    <h1>Page title</h1>
      <section class="c-section--primary">
        <h2 class="c-title">Section title</h2>
        <div class="c-card">
            <h3 class="c-card__title">Title</h3>
            <p class="c-card__description">Description</p>
        </div>
        <div class="c-card">
            <h3 class="c-card__title">Title</h3>
            <p class="c-card__description">Description</p>
        </div>
      </section>
      <section class="c-section--primary">
        <h2 class="c-title">Section title</h2>
        <div class="c-card c-card--filled">
            <h3 class="c-card__title">Title</h3>
            <p class="c-card__description">Description</p>
        </div>
      </section>
  </body>
</html>
```

Slika 3.1. HTML dokument, izvor: *izrada autora*

3.2. Stilski jezik CSS

CSS (skraćeno od *Cascading Style Sheets*), omogućuje korisnicima kreiranje stilskog izgleda web stranica, odnosno specificiranje načina na koji se stranice prezentiraju korisnicima. Prema [11], CSS se može koristiti u raznim stilskim kontekstima, kao što su mijenjanje boje i veličine elemenata, kreiranje specifičnog rasporeda elemenata, te prikaz različitih animacija elemenata. Slikom 3.2. prikazan je način prezentiranja elemenata te učitavanje CSS-a u internetskom pregledniku.



Slika 3.2. Prikaz HTML elemenata i CSS-a u pregledniku, izvor: [12]

Sintaksa pisanja CSS-a sastoji se od korištenja jednog ili više selektora (odvojenih zarezom), te deklaracije napisane unutar vitičastih zagrada. Prema [13], selektori ukazuju na HTML element koji se želi stilizirati CSS-om, a deklaracijski blok sadrži jednu ili više deklaracija odvojenih točkom sa zarezom. Svaka deklaracija sadrži CSS obilježje i njegovu vrijednost, odvojene dvotočjem. Prikaz jednostavnog CSS-a može se vidjeti na slici 3.3.

```
.c-button {
    &--primary {
        background-color: #0e58f9;
        color: #fff;
    }
    &--outline {
        border: 1px solid #0e58f9;
        color: #0e58f9;
    }
    &--sml {
        padding: 8px 6px;
    }
    &--base {
        padding: 12px 6px;
    }
    &__icon {
        width: 12px;
        height: 12px;
    }
}
```

Slika 3.3. Korištenje CSS-a, izvor: *izrada autora*

Slikom 3.3., uz korištenje CSS-a, prikazano je i korištenje posebne metodologije naziva BEM (skraćeno od *Block Element Modifier*). BEM metodologija koristi se u svrhu organiziranja CSS klasa kako bi se održala jednostavnost strukture te na jednostavan način omogućilo održavanje CSS koda. U nazivu *Block Element Modifier*, riječ „*Block*“ označava samostalni objekt stranice, poput gumba ili navigacijskog bloka. „*Element*“ predstavlja komponentu unutar bloka s jedinstvenom funkcijom – poput ikone unutar gumba prikazane slikom 3.3.. Elementi se označavaju posebnom vrstom oznake prije naziva („__“), koja omogućava njihovu prepoznatljivost te pravilno interpretiranje koda. Prema [14], „*Modifier*“ označava jednu od varijanti prikaza bloka, kao što je njegova veličina (na slici 3.3., veličine gumba definirane su nazivima „*sml*“ i „*base*“). Poput elemenata, *modifier*-i se razlikuju od ostatka CSS-a koristeći posebnu oznaku prije naziva, što je u ovom slučaju „—„. Korištenjem BEM metodologije osigurava se točno raspoznavanje svrhe elemenata i blokova u kojima se isti pojavljuju, a imena klasa čitka su i intuitivna za sve koji pristupe HTML-u i CSS-u stranice.

3.4. Programski jezik JavaScript

Prema [15], JavaScript je programski jezik uobičajeno korišten prilikom kreiranja dinamičkih interakcija u razvoju internetskih stranica, aplikacija, poslužitelja i ostalo. JavaScript datoteke moguće je uključiti u HTML dokument unutar oznake <script> te na taj način omogućiti upotrebu ovog jezika na stranici, ili ga integrirati iz vanjskog izvora. Uz upotrebu u svrhu razvoja internet i mobilnih aplikacija, JavaScript pruža izvrsne mogućnosti prilikom kreiranja internetskih poslužitelja te poslužiteljskih aplikacija. Prema [15], mogućnosti JavaScripta sežu od prikaza animacija, promjene svojstava i sadržaja HTML elemenata, do kreiranja i čitanja kolačića i upotrebe kompleksnih funkcija i podatkovnih oblika. Jedna od vrlo čestih primjena JavaScripta u izradi aplikacija je prikaz podataka primljenih s poslužiteljske strane u posebnom formatu naziva JSON (*JavaScript Object Notation*). Na taj način se osigurava razmjena podataka između klijentske i poslužiteljske strane aplikacije, a sam JSON format podataka prikazan je slikom 3.4.

```
"questions": [
    {
        "title":"First Question",
        "points": "3",
        "quiz_id": "1"
    },
    {
        "title":"Second Question",
        "points": "0",
        "quiz_id": "1"
    },
],
```

Slika 3.4. JSON format podataka, izvor: *izrada autora*

Uz brojne prednosti JavaScripta, JavaScript okviri (engl. *frameworks*) koriste se u razvoju modernih aplikacija pružajući programerima set predefiniranih biblioteka, funkcionalnosti i komponenti olakšavajući pri tome proces razvoja. Okviri pružaju strukturu i omogućavaju brže i efikasnije razvijanje kompleksnih web aplikacija uz razvijanje visoko funkcionalnih i skalabilnih aplikacija. Neki od JavaScript okvira su React, Angular i Vue, detaljnije opisan u nastavku. Na slici 3.5. prikazan je jednostavan primjer koda napisan koristeći JavaScript.

```
function buttonClick(){
    let buttons = document.getElementsByClassName("c-button");
    buttons.forEach(button, idx => {
        button.addEventListener("click", function(){
            alert("Hi!");
        })
    });
}
```

Slika 3.5. Primjer JavaScript koda, izvor: *izrada autora*

3.5. Razvojni okvir Vue.js

Prema [16], Vue je moderno JavaScript razvojno okruženje koje pruža napredne mogućnosti za progresivna poboljšanja te unaprjeđenje iskustva i performansi prilikom kreiranja kompleksnih aplikacija. Jedna od bitnih karakteristika Vue-a je njegova komponentna arhitektura. Kreiranjem modularnih komponenti, programerima je omogućeno ponovno korištenje i jednostavno kombiniranje istih u procesu izrade aplikacija. Na taj način omogućuje se organiziranost koda, poboljšana skalabilnost te olakšano održavanje koda. Sve .vue datoteke sadrže jednaku osnovnu

strukturu, koja se sastoji od oznake <template>, oznake <script>, te oznake <style>. Oznaka <template> sadrži cjelokupnu *markup* strukturu uključujući HTML, te uz to i određene sintakse specifične za Vue. Primjer korištenja jedne od spomenutih jednostavnih Vue sintaksi, korištenje „*v-show*“ smjernice, te osnovna struktura Vue dokumenta, prikazani su slikom 3.6..

```
<template>
    <div>
        <Nav />
        <div :v-show="alert">Alert is set!</div>
    </div>
</template>

<script>
import Nav from "../components/Nav";
export default {
    data() {
        return {
            alert: false,
        };
    },
    components: {
        Nav,
    },
    methods: {
        setAlert() {
            if (!this.alert) this.alert == true;
            else this.alert == false;
        },
    },

};
</script>

<style lang="scss">
</style>
```

Slika 3.6. Primjer Vue dokumenta, izvor: *izrada autora*

Također, jedna od specifičnosti sintakse Vue dokumenata, to je da svaka <template> oznaka smije sadržavati isključivo jednu pod-oznaku (u primjeru sa slike 3.6., ta oznaka je unutar kojeg se nalaze sve ostale potrebne oznake). Vue omogućava korištenje predefinirane funkcije *data*, u kojoj se definira kolekcija logike i pohranjuje unutar komponente, a kojoj se može pristupiti u cijeloj .vue datoteci. Prema [17], spomenuta funkcija vraća reaktivno stanje instance komponente.

Prilikom kreiranja funkcija, potrebno ih je definirati unutar *methods* objekta. Nakon definiranja funkcija na takav način, one se mogu, nadalje, koristiti za obavljanje raznih akcija unutar datoteke.

3.6. Skriptni jezik PHP

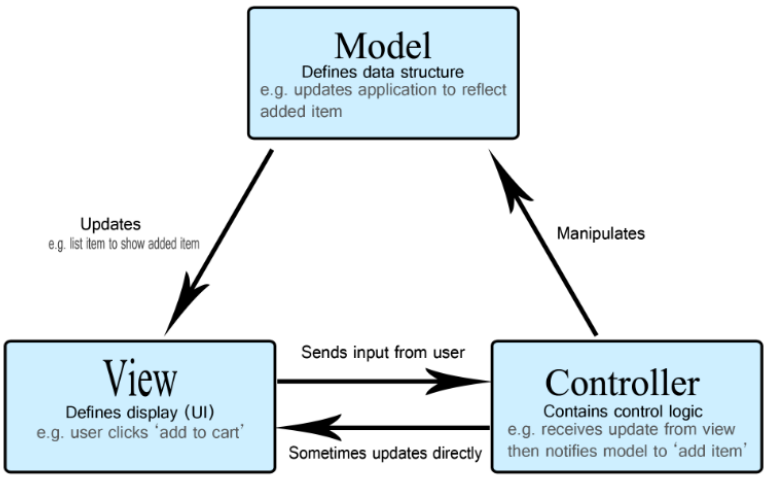
Prema [18], PHP (skraćeno od *PHP: Hypertext Preprocessor*) je skriptni jezik visoke razine otvorenog koda dizajniran za obradu podataka na strani poslužitelja. Kao jedni od bitnijih razloga njegove široke primjene u razvoju web stranica i aplikacija, ističu se njegova efikasnost, jednostavnost i fleksibilnost. Također, PHP ne ovisi niti o jednoj platformi, što omogućuje pojedincima korištenje PHP-a neovisno o operacijskom sustavu kojeg koriste. PHP se često koristi u kombinaciji s Apache web poslužiteljem te MySQL bazom podataka, no, prema [18], ne postoji ograničenje s obzirom na korištenje baze podataka. Zbog navedenih stavki, te svoje velike fleksibilnosti, u današnjem svijetu skoro 80% web stranica koriste PHP [18]. Jedna od brojnih prednosti PHP-a je mogućnost ugradnje PHP skripte bilo gdje u HTML dokumentu. PHP skripta sadrži početnu oznaku <?php te završnu oznaku ?>. Između navedenih oznaka, može se pisati PHP kod, koji sadrži varijable napisane s početnim znakom „$“ (dolar), a završetak svakog iskaza (engl. *statement*) završava oznakom „;“. Kada je riječ o osjetljivosti na velika i mala slova (engl. *case-sensitivity*), sve ključne riječi, klase i funkcije nisu osjetljive na isto. Ipak, imena varijabli jesu osjetljiva, zbog čega je potrebno obratiti pažnju prilikom njihovog definiranja. Slika 3.7. prikazuje primjer korištenja PHP koda.

```php
public function getCoursesLessons()
{
    return Course::with('lessons')->get();
}

public function getCourses()
{
    return Course::all();
}

public function getUserCourses(Request $request)
{
    return Course::where('user_id', $request->id)->get();
}

public function getCourse(Request $request)
{
    return Course::where('id', $request->id)->get();
}
```

Slika 3.7. Primjer PHP koda, izvor: *izrada autora*

3.7. Razvojni okvir Laravel

Jedno od vodećih PHP razvojnih okruženja, Laravel, okvir je web aplikacija koji nudi bogat skup funkcionalnosti ubrzavajući brzinu razvoja programskih rješenja. Koristeći MVC (engl. *Model-View-Controller*) obrazac arhitekture, Laravel pomaže svojim korisnicima u održavanju konzistentnosti i strukture programskog koda. Također, prema [19], MVC pristup olakšava razvoj manjih i većih web aplikacija. Ključna karakteristika MVC-ja je odvajanje aplikacije u tri glavne logičke komponente – model, pogled (engl. *View*), i upravljač (engl. *Controller*). Svaka od navedenih komponenti izgrađena je na način da sudjeluje u određenim aspektima razvoja aplikacije. Komponenta model odgovara svoj logici povezanoj s podacima s kojom korisnik radi. Pogled je komponenta s ciljem korištenja sve logike povezane s korisničkim sučeljem aplikacije. Prema [20], upravljači (engl. *Controllers*) se koriste kao sučelje između modela i pogleda kako bi se uspjela obraditi poslovna logika te nadolazeći zahtjevi, manipulirati podacima koristeći komponentu modela, te omogućiti prikaz krajnjeg rezultata interakcijom s komponentom pogleda. Slika 3.8. prikazuje arhitekturu MVC, dok slika 3.9. prikazuje strukturu upravljača u Laravelu.



Slika 3.8. MVC arhitektura, izvor: [21]

```php
<?php

namespace App\Http\Controllers;

use App\Models\Option;
use Illuminate\Http\Request;

class OptionController extends Controller
{
    public function getOptions()
    {
        return Option::all();
    }

    public function getOption(Request $request)
    {
        return Option::where('id', $request->id)->get();
    }

    public function saveOption(Request $request)
    {
        $Option = Option::create([
            'text' => $request->text,
            'points' => $request->points,
            'question_id' => $request->question_id,
        ]);

        return $Option;
    }
}
```

Slika 3.9. Struktura upravljača u Laravelu, izvor: *izrada autora*

Uz spomenutu MVC arhitekturu, Laravel pruža i brojne druge mogućnosti s ciljem olakšavanja razvoja aplikacija, poput sučelja naredbenog retka Artisan integriranog s Laravelom. Prema [22], ovo sučelje korisnicima nudi širok spektar naredbi stvorenih kako bi olakšale migriranje podataka te upravljanje bazama podataka te stvaranje upravljača i modela, čime osigurava da se programeri što više fokusiraju na kreiranje same logike aplikacije. Primjer korištenja naredbe Artisana prikazana je slikom 3.10., a njena svrha je kreiranje migracije koja će nakon toga postati dostupna unutar *database/migrations* direktorija. Laravel u daljnjim koracima izrade aplikacije koristi naziv migracije kako bi protumačio naziv tablice, te odlučio hoće li migracija kreirati novu tablicu. Primjer strukture jednostavne migracije prikazan je na slici 3.11..

```
php artisan make:migration subjects_table
```

Slika 3.10. Korištenje naredbe Artisana, izvor: *izrada autora*

```php
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class SubjectsTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::connection($this->connection)->create('subjects', function (Blueprint $table) {
            $table->id();
            $table->string('title');
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::connection($this->connection)->dropIfExists('subjects');
    }
}
```

Slika 3.11. Primjer strukture migracije, izvor: *izrada autora*

3.8. MySQL sustav upravljanja relacijskim bazama podataka

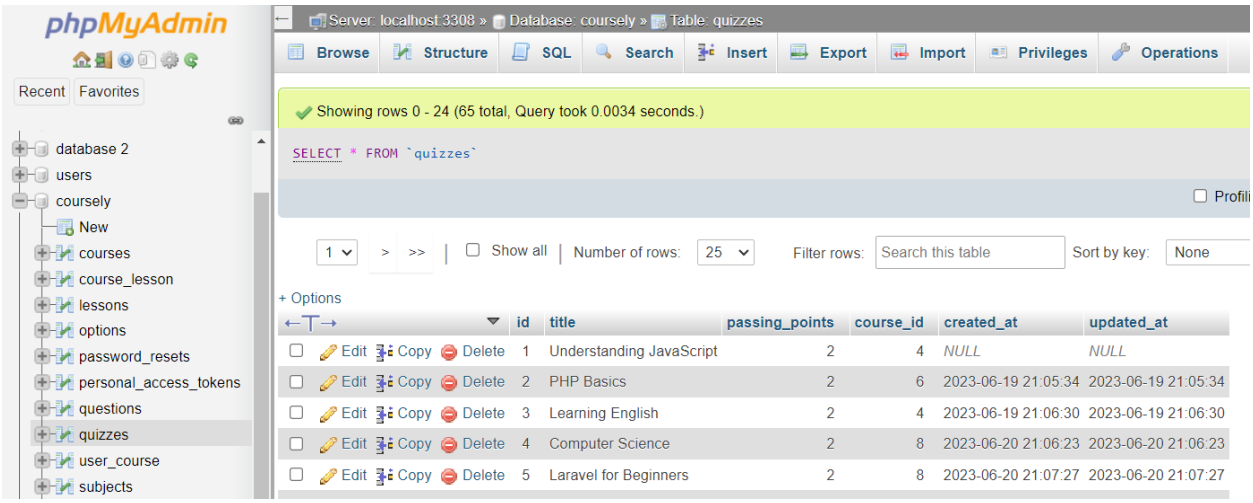
Baziran na strukturiranom upitnom jeziku (engl. *Structured Query Language* – SQL), prema [23], MySQL služi pristupanju, dodavanju te obradi podataka pohranjenih u računalnoj bazi podataka. Razlikujući dvije vrste baza podataka, relacijske i ne-relacijske baze, MySQL pripada kategoriji relacijskih baza podataka, implementirajući ključne karakteristike poput spremanja podataka u višestruke, odvojene tablice i retke, te dodjeljivanje jedinstvenih oznaka elementima tablice. Kao što je spomenuto u [24], relacijske baze podataka poput MySQL-a nude brojne prednosti, uključujući visoku razinu sigurnosti prilikom prijenosa podataka, napredne mogućnosti rada sa

strukturiranim podacima, pisanje kompleksnih upita korištenih prilikom analize podataka, i slično. Prethodno navedene karakteristike osiguravaju optimiziranje aktivnosti poput dohvaćanja podataka i ažuriranje informacija, čineći relacijske baze podataka iznimno prihvaćenima i često upotrebljenima. Na slici 3.12. prikazan je primjer MySQL upita, koji u ovom slučaju predstavlja naredbu za kreiranje tablice „employee“ te njenih stupaca.

```
CREATE TABLE  employee(
   'id' INTEGER  NOT NULL AUTO_INCREMENT,
   'name' VARCHAR(30) NOT NULL,
   'profile' VARCHAR(40) DEFAULT 'engineer',
   PRIMARY KEY ('id')
);
```

Slika 3.12. MySQL upit, izvor: [25]

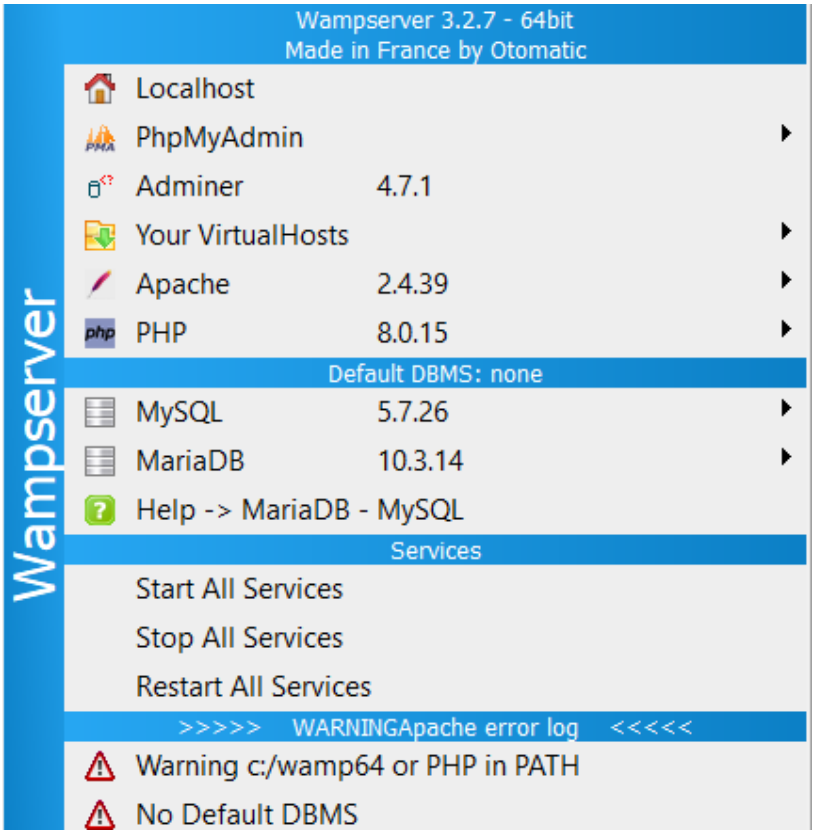
Jedna od najpopularnijih platformi za upravljanje MySQL bazama podataka, phpMyAdmin, alat je otvorenog koda (engl. *open-source*) napisan u PHP-u, koji podržava različite tipove operacija na MySQL-u [26]. Neke od spomenutih operacija su upravljanje tablicama i stupcima, dopuštenjima, poveznicama, ali i kreiranje, ažuriranje, izmjenu te uklanjanje navedenih objekata i slično. Slika 3.13. prikazuje sučelje phpMyAdmina, s prikazom tablice „quizzes“ relacijske baze podataka.



Slika 3.13. Prikaz sučelja phpMyAdmin-a, izvor: *izrada autora*

3.9. WAMP poslužitelj

WAMPServer (WAMP skraćeno od *Windows, Apache, MySQL, and PHP*), čijom se instalacijom na operacijski sustav korisnika također instaliraju i internetski poslužitelj Apache, sustav upravljanja bazama podataka MySql te skriptni jezik PHP [27], besplatno je internetsko razvojno okruženje namijenjeno za operacijski sustav Windows [28]. Jednostavno grafičko sučelje prikazano slikom 3.14., omogućuje korisnicima lako izvođenje akcija i testiranje programskog koda na lokalnom internetskom poslužitelju. WAMP omogućava brz pristup prethodno opisanom alatu PhpMyAdmin čime ubrzava proces izrade aplikacija.



Slika 3.14. Prikaz Wampserver-a, izvor: *izrada autora*

4. POSTUPAK IZRADE APLIKACIJE

Prilikom izrade ove web aplikacije, naglasak je stavljen na provođenje cjelokupnog procesa razvoja proizvoda, koji započinje definiranjem arhitekture aplikacije, utvrđivanjem funkcionalnih zahtjeva na sustav te cjelokupnim dizajnom korisničkog sučelja i iskustva aplikacije. Nakon završetka dizajna, proces implementacije odvijao se koristeći razvojno okruženje Visual Studio Code, koje pruža sve potrebne resurse za razvoj i testiranje aplikacije, te njeno održavanje.

4.1. Dizajn aplikacije

Kako bi se osigurala jedinstvena pozicija proizvoda na tržištu već dostupnih konkurentskih aplikacija, potrebno je definirati što specifičniju granu industrije u kojoj će se proizvod nalaziti. U poglavlju 2., kroz trenutno dostupne konkurentske aplikacije, mogu se primijetiti različitosti u pogledu njihovih primarnih funkcija, ciljane publike, te sveukupne uske tematike na koju se fokusiraju. Sukladno tome, svaka od navedenih platformi zadovoljavat će potrebe različitih skupina korisnika. Aplikacija Coursely namijenjena je za nekoliko glavnih skupina korisnika poput studenata, adolescenata, te mladih osoba s ciljem promjene trenutne karijere. S obzirom na to da se velik broj funkcionalnosti, a također i vizualni identitet aplikacije, kreiraju prema potrebama grupe pojedinaca koji će biti većinski korisnici aplikacije, definirana su dva glavna predstavnika ciljanih skupina u obliku korisničkih osoba (engl. *user personas*). Korisničke osobe čest su alat korišten u dizajnu s ciljem stvaranja jasnog i detaljnog profila potencijalnih korisnika aplikacije. One obuhvaćaju karakteristike, ciljeve, motivacije, potrebe i ponašanja ciljanih korisnika, te na taj način omogućuju razumijevanje stvarnih potreba korisnika na dubljoj razini u svrhu kreiranja izuzetnog korisničkog iskustva usmjeravanjem strategije razvoja proizvoda. Prikazom s tablica 4.1. i 4.2., raspoznaju se dvije različite korisničke osobe koje su predstavnici ciljane publike ove platforme.

Tablica 4.1. Korisnička osoba Marija, izvor: *izrada autora*

Ime	Marija
Dob	21
Lokacija	Osijek, Hrvatska
Obitelj	/
Karijera	Student
Ciljevi	1. Mogućnost učenja više različitih predmeta na jednom mjestu 2. Edukacija na daljinu (engl. *remote*)

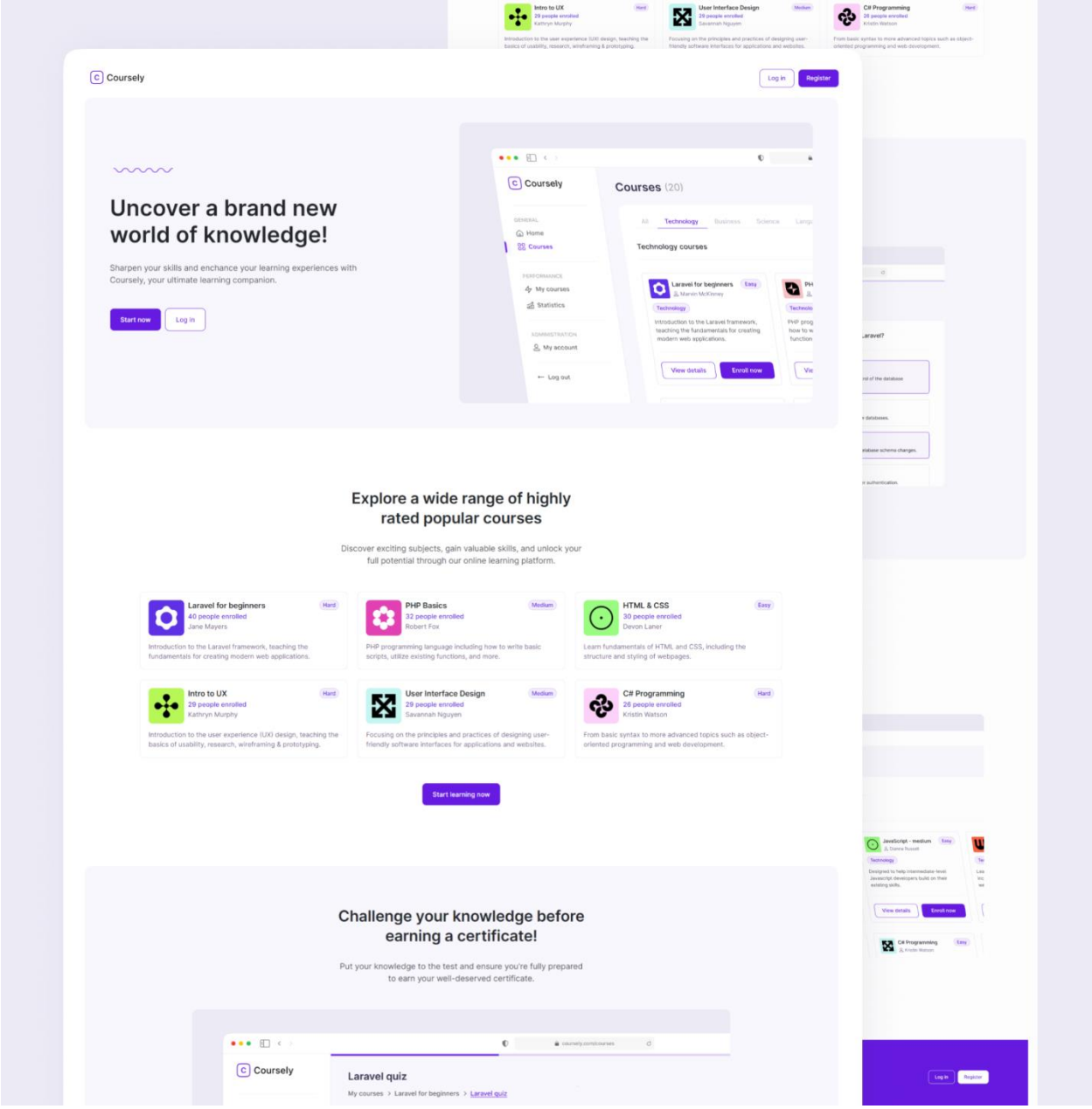
	3. Dostupnost aplikacije na mobilnim uređajima
Frustracije	1. Cjenovne karakteristike prethodno upisanih tečajeva 2. Nedovoljan broj tečajeva dostupan na zasebnoj platformi 3. Vrlo slaba optimizacija aplikacija za rad na mobilnim uređajima
Motivacije	1. Pregled statistike završenih i trenutno upisanih tečajeva u smislu praćenja napretka 2. Dobivanje pozitivnog povratnog odgovora od strane profesora 3. Primanje certifikata nakon uspješnog završetka tečaja
Osobnost	Introvert
Tehnološka spremnost	Marija vrlo dobro poznaje i koristi razne tehnologije na svakodnevnoj bazi, ali preferira korištenje mobilnih uređaja. Također koristi i prijenosno računalo zbog studentskih obveza, ali većinu privatnih interesa obavlja putem mobilnog uređaja.

Tablica 4.2. Korisnička osoba Luka, izvor: *izrada autora*

Ime	Luka
Dob	30
Lokacija	Zadar, Hrvatska
Obitelj	Oženjen
Karijera	Vlasnik poduzeća za proizvodnju elektromaterijala
Ciljevi	1. Proširenje poslovanja u drugu državu

	2. Pristup tečajevima jezika za početnike 3. Mogućnost preuzimanja materijala za učenje, kako bi se, po potrebi, i supruga paralelno educirala
Frustracije	1. Pretjerano detaljizirani opisi tema trenutno dostupnih materijala tečajeva, otežavajući učenje na početničkoj razini 2. Nedovoljan broj tečajeva dostupan na zasebnoj platformi
Motivacije	1. Učenje osnovnih elemenata stranog jezika 2. Postizanje osnovne razine komunikacije na stranom jeziku 3. Primanje certifikata nakon uspješnog završetka tečaja
Osobnost	Ekstrovert
Tehnološka spremnost	Luka izbjegava korištenje mobilnih uređaja, osim u svrhe bitnih tekstualnih poruka i poziva. Najčešće koristi tablet, ali posjeduje i veliku količinu znanja o računalima.

Prema prethodno opisanim korisnicima, uz detaljnije razumijevanje njihovih potreba, povećava se vjerojatnost da će aplikacija u stvarnosti zadovoljavati spomenute potrebe. Kako bi se kreirao vizualni identitet aplikacije te detaljni dizajn korisničkog sučelja (engl. *User Interface Design*), u obzir se kao bitan parametar uzima dob ciljane skupine korisnika, te sukladno tome određuje smjer u kojem se dizajn u nastavku razvoja proizvoda treba kretati. Izbor tipografije, primarnih i sekundarnih boja aplikacije, stilskih izražaja poput stila fotografija i potencijalnih vizuala koji će se nalaziti na stranici, također dijelom ovise o ciljanoj skupini prema kojoj se kreira aplikacija. Slikom 4.3. prikazan je dizajn korisničkog sučelja početne stranice (engl. *Homepage*) portala Coursely.

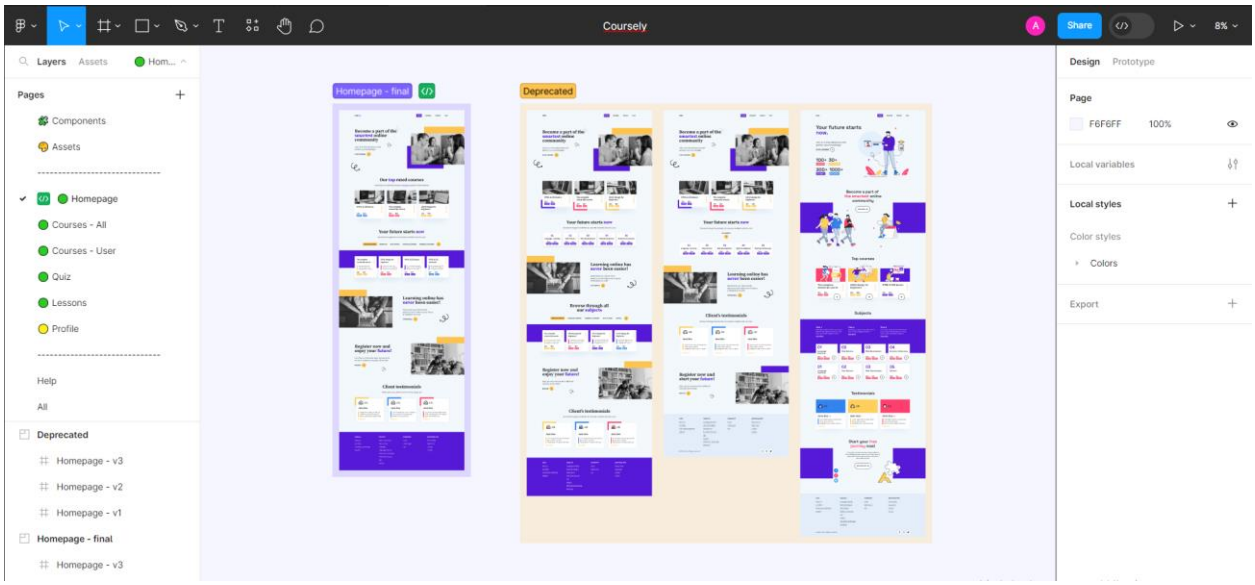


Slika 4.3. Dizajn korisničkog sučelja početne stranice portala, izvor: *izrada autora*

Početna stranica portala omogućuje korisnicima pregled općenitih funkcionalnosti aplikacije, kao i registraciju novih korisnika te prijavu već postojećih. Više o registraciji i prijavi korisnika može se pročitati u ulomku 4.2..

Alat korišten prilikom izrade dizajna, Figma, preglednički je baziran (engl. *browser-based*) alat čije mogućnosti uključuju podršku prilikom vizualnog dizajna, izradu kompleksnih prototipa te korisničkih interakcija, organiziranje i strukturiranje datoteka i resursa potrebnih tijekom procesa definiranja dizajna, i slično. Jedna od prednosti alata Figma je mogućnost korištenja proširenja

(engl. *plugins*) omogućujući korisnicima brojne pogodnosti te unapređenje produktivnosti. Prikaz sučelja Figma dostupan je na slici 4.4..

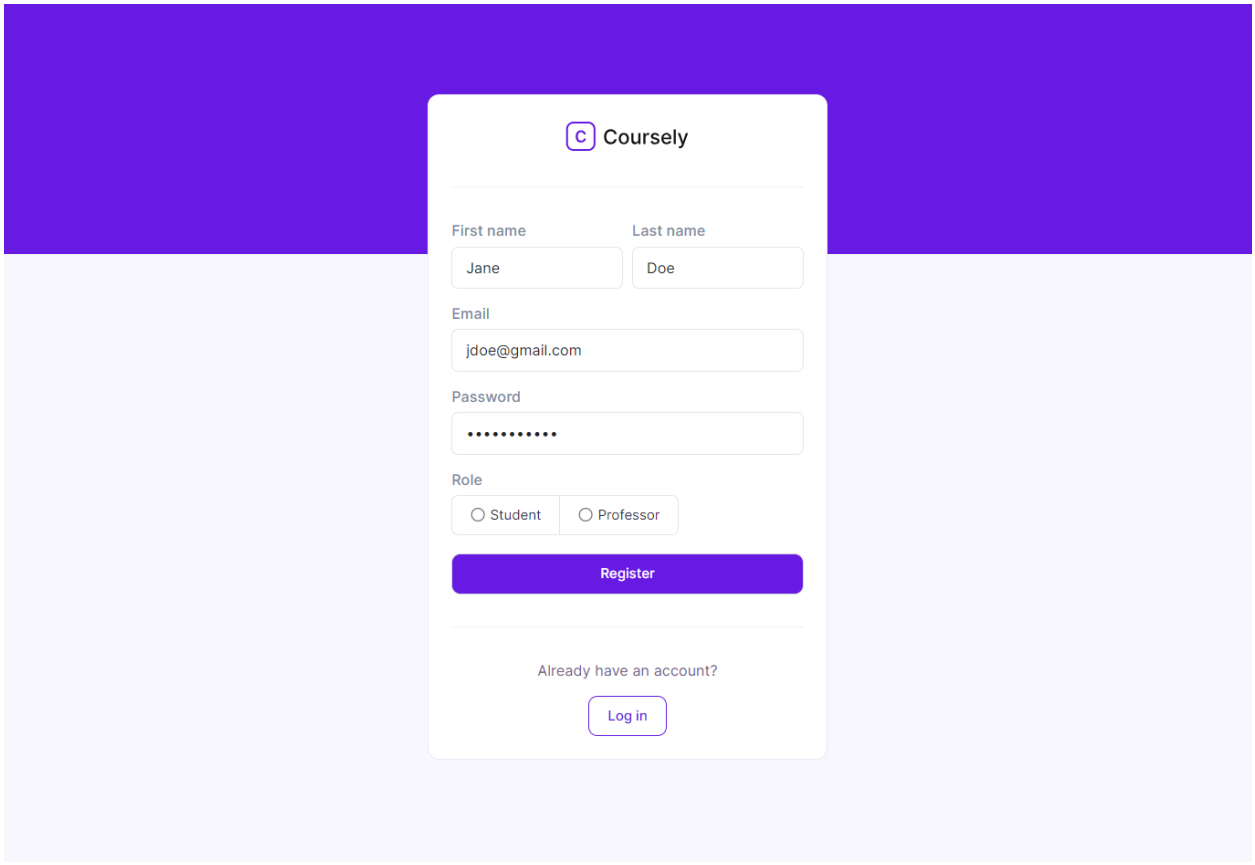


Slika 4.4. Prikaz sučelja Figma, izvor: *izrada autora*

Od nekoliko glavnih elemenata sučelja Figme, razlikuju se gornja alatna traka s mogućnostima poput kreiranja novih vektora, teksta, oblika i slično, lijevi panel koji sadrži grupirane stranice i sve slojeve (engl. *layers*) dizajna svake stranice, te desni panel u kojemu se nalaze mogućnosti kreiranja prototipa i popis globalnih stilova. Jedna od naprednih mogućnosti Figme uključuje definiranje ekrana koji predstavljaju dovršen dizajn spreman za implementaciju, omogućujući različitim članovima razvojnih timova jednostavnu suradnju. Takve stranice označene su dodatnom zelenom ikonom, te se nakon prebacivanja u razvojni način (engl. *development mode*) rada Figme, programerima omogućuju brojne dodatne funkcionalnosti poput pripremljenih dijelova koda za pojedini element dizajna, olakšano provjeravanje (engl. *inspect*) svojstava komponenti, i slično.

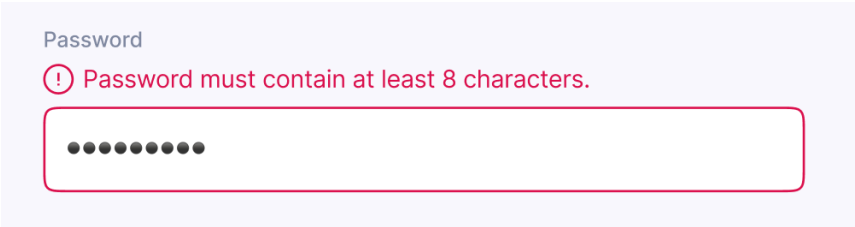
4.2. Registracija i prijava korisnika u sustav

Kako bi se korisnicima osiguralo iskustvo praćenja napretka upisanih tečajeva te različitim ulogama korisnika omogućile odgovornosti vezane isključivo za te uloge, potrebno je izvršiti registraciju prije početka korištenja mogućnosti aplikacije. Registracija od korisnika zahtijeva unos adrese elektroničke pošte (engl. *email address*), kreiranje korisničke zaporke te odabir uloge korisnika. Slika 4.5. prikazuje zaslon registracije korisnika u sustav.



Slika 4.5. Zaslon registracije korisnika, izvor: *izrada autora*

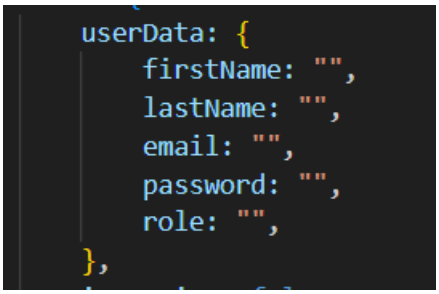
Slikom 4.5., također su prikazana različita stanja u kojima se polja za unos elemenata mogu nalaziti. Osim prikazanih stanja, postoji i stanje prikazano u slučaju da korisnik unese pogrešni format podatka unutar određenog polja, prikazan slikom 4.6..



Slika 4.6. Prikaz poruke greške prilikom unosa podataka u polje, izvor: *izrada autora*

Registracija korisnika odvija se posjetom putanje */register* portala. Zaslon prikazan na spomenutoj putanji, Vue komponenta imena *Register*, podatke koje korisnik unese, nakon pritiska gumba za registraciju, proslijeđuje Laravel upravljaču *UserController*. Taj upravljač nadalje kreira novog korisnika spremanjem proslijeđenih podataka u tablicu *users*. Vue komponenta *Register* sastoji se od nekoliko glavnih dijelova povezanih na način kako bi se uspješno ostvario proces kreiranja

korisnika. Slikom 4.7. prikazan je format objekta korisnika koji će se proslijediti upravljaču, a na slici 4.8. prikazana je funkcija kojom se odvija spomenuto prosljeđivanje.



Slika 4.7. Format podatka proslijeđen UserController-u, izvor: *izrada autora*

```
methods: {
    async register() {
        this.isLogging = true;
        const res = await this.callApi("post", "/register", this.userData);
        if (res.status === 200) {
            this.success("Registered successfully");
            setTimeout(function () {
                window.location = "/";
            }, 1500);
        } else {
            if (res.status === 422) {
                if (res.data.errors.firstName) {
                    this.error(res.data.errors.firstName[0]);
                }
                if (res.data.errors.lastName) {
                    this.error(res.data.errors.lastName[0]);
                }
                if (res.data.errors.email) {
                    this.error(res.data.errors.email[0]);
                }
                if (res.data.errors.password) {
                    this.error(res.data.errors.password[0]);
                }
            } else {
                this.swr(res.data.message);
            }
        }
        this.isLogging = false;
    },
},
```

Slika 4.8. Funkcija registriranja korisnika, izvor: *izrada autora*

Na slici 4.8., prikazana je funkcija *callApi,* koja prima 3 parametra – metodu, putanju, i podatke. Riječ je o funkciji koja koristeći Axios posjeduje mogućnost slanja proizvoljnih zahtjeva prema

poslužiteljskoj strani. Prema [29], Axios je temeljena na obećanju (engl. *promise-based*) HTTP biblioteka koja nudi razne načine slanja zahtjeva poput *get, post, put/patch* ili *delete*. Jedini parametar zahtjevan od strane Axiosa je URL putanja, ali uz to postoji i mogućnost konfiguriranja parametara prema tipu zahtjeva koji se želi poslati. Slika 4.9. prikazuje definiciju funkcije *callApi*, kojoj se prosljeđuju metoda (*get, post, put/patch, delete, head, request, options* [29]), putanja, te podaci.

```
async callApi(method, url, data) {
    try {
        return await axios({
            method: method,
            url: url,
            data: data,
        });
    } catch (error) {
        return error.response;
    }
},
```

Slika 4.9. Axios funkcija, izvor: *izrada autora*

Laravel putanje (engl. *routes*), definirane unutar *web.php* datoteke, povezuju određenu putanju s vrstom zahtjeva i upravljačkom funkcijom zaduženom za pozivanje u slučaju pristupa specifičnoj putanji. Slika 4.10. prikazuje tri jednostavne putanje unutar *web.php* datoteke, od kojih sve pozivaju različite funkcije upravljača *UserController*.

```
Route::post('/register', [UserController::class, 'register']);
Route::post('/login', [UserController::class, 'login']);
Route::get('/logout', [UserController::class, 'logout']);
```

Slika 4.10. Putanje unutar web.php, izvor: *izrada autora*

Prethodno prikazane putanje koriste se prilikom registracije i prijave već postojećeg korisnika u sustav, te prilikom odjave korisnika iz sustava. Prva linija prikazana na slici 4.10. poziva funkciju *register* upravljača *UserController*. U toj funkciji odvija se provjera ograničenja proslijeđenih podataka, šifriranje zaporke, te kreiranje korisnika u bazi podataka. Provjera ograničenja proslijeđenih podataka može se napraviti na dva načina: korištenjem funkcije *validate* Laravela, ili ručnim pozivanjem metode *make* klase *Validator*. Oba prethodno spomenuta načina obavit će jednak zadatak, provjeru pravila zadanih kao ograničenja podataka. Prema [30], u slučaju da

provjera utvrdi kako zadana pravila nisu ispunjena, vraćanjem odgovora sa statusom *422*, pregledniku će se vratiti JSON reprezentacija utvrđenih pogrešaka. Slika 4.11. prikazuje funkciju *register* upravljača *UserController*, u kojoj se nalaze zadana pravila za podatke koje korisnik treba unijeti. Na primjer, zaporka mora imati minimalnu duljinu od 8 znakova, dok ime korisnika ne smije sadržavati više od 30 znakova.

```php
use App\Models\User;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;

class UserController extends Controller
{
    public function register(Request $request)
    {
        $this->validate($request, [
            'firstName' => 'required|max:30',
            'lastName' => 'required|max:30',
            'email' => 'bail|required|email|unique:users|max:30',
            'password' => 'bail|required|min:8',
            'role' => 'required',
        ]);

        $password = bcrypt($request->password);
        User::create([
            'firstName' => $request->firstName,
            'lastName' => $request->lastName,
            'email' => $request->email,
            'password' => $password,
            'role' => $request->role
        ]);
    }
```

Slika 4.11. Funkcija upravljača za registriranje korisnika, izvor: *izrada autora*

Način spremanja korisničke zaporke u bazu podataka potencijalno može predstavljati velik sigurnosni rizik prilikom izrade aplikacija. Ukoliko se zaporka sprema u jednakom formatu u kojem je unesena od strane korisnika, postoji vrlo velika mogućnost dolaska do sigurnosnih problema u budućnosti života aplikacije. Iz tog razloga, Laravel nudi mogućnost jednostavnog spremanja zaporke u bazu podataka korištenjem funkcije *bcrypt*. Nakon što se zaporka promijeni korištenjem navedene funkcije, dovoljno ju je samo spremiti u tablicu te na taj način korisnicima omogućiti sigurno registriranje u aplikaciju.

Korisnička prijava u sustav, iako dizajnom vrlo slična zaslonu registracije, zahtijeva korištenje nekoliko različitih bitnih funkcionalnosti kako Laravela, tako i okvira Vue. Kao što je prikazano na prethodnoj slici (4.10.), prilikom prijave već registriranog korisnika u sustav, poziva se funkcija *login* upravljača *UserController*. Ta funkcija, za razliku od *register* funkcije, ne posjeduje svrhu kreiranja novog korisnika, nego provjeru postoji li validan registrirani korisnik s podacima unesenim u formu za prijavu. Laravel nudi opsežne mogućnosti kada je riječ o autentikaciji korisnika, pa se tako korištenjem statičkog sučelja (engl. *interface*) Auth može pristupati raznim metodama namijenjenima korištenju upravo u spomenute svrhe. Prema [31], sva statička sučelja Laravela (također naziva *Facades*), definirana su u prostorima imena (engl. *namespace*) *Illuminate\Support\Facades* što im omogućuje brz pristup te jednostavno korištenje. Koristeći metodu *attempt*, koja prihvaća parove ključ – vrijednost (engl. *key – value pairs*) kao prvi argument, pokušat će pronaći korisnika u bazi podataka na temelju proslijeđenih parametara. Prilikom prosljeđivanja korisničke zaporke, koristeći ovu metodu nije potrebno koristiti funkcije za kriptiranje zaporke, jer će to Laravel samostalno napraviti prilikom usporedbe zaporki. Ukoliko je autentikacija uspješna, *attempt* metoda će vratiti vrijednost *true*. Na slici 4.12. prikazana je *login* funkcija kojom se odvijaju sve provjere prilikom prijave korisnika u sustav.

```php
public function login(Request $request)
{
    $this->validate($request, [
        'email' => 'bail|required|email',
        'password' => 'required'
    ]);

    if (Auth::attempt(['email' => $request->email, 'password' => $request->password])) {
        $user = Auth::user();

        return response()->json([
            'msg' => 'You are logged in',
            'user' => $user,
        ]);
    } else {
        return response()->json([
            'msg' => 'Unknown user'
        ], 401);
    }
}
```

Slika 4.12. Funkcija za prijavu korisnika, izvor: *izrada autora*

Uz korištenje metode *attempt* te sučelja *Auth*, bitan dio funkcije login uključuje i prosljeđivanje objekta korisnika u Vue, gdje se pohranjuje trenutno stanje aplikacije sadržavajući korisnika kao trenutno prijavljenog u aplikaciji. Na taj način omogućeno je dohvaćanje potrebnih podataka

trenutno prijavljenog korisnika, te eventualnu izmjenu trenutnog stanja (engl. *state*) aplikacije. Navedeno je omogućeno korištenjem biblioteke sustava za upravljanje stanjem (engl. *state management library*) aplikacije Vuex. Vuex-ov *store* kolekcija je različitih metoda i podataka, od kojih neke metode mogu obrađivati podatke, ili ažurirati svojstva *store*-a s određenim vrijednostima [32]. Stanje predstavlja kolekciju podataka u određenom vremenskom trenutku, koja može biti promijenjena korisničkim interakcijama koje prosljeđuju podatke mutacijama (engl. *mutations*) te na taj način ažuriranju stanje objekta. Slika 4.13. prikazuje kreiranje stanja za *store* ove aplikacije.

```
import Vuex from "vuex";

Vue.use(Vuex);

const store = new Vuex.Store({
    state: {
        user: {
            status: false,
            username: "",
            role: "",
            id: "",
        },
    },
    getters:{},
    actions: {},
    mutations: {
        setUserLoginStatus(state, data) {
            state.user.status = data;
            if (data.firstName) {
                state.user.username = data.firstName;
                state.user.role = data.role;
                state.user.id = data.id;
            }
        },
    },

});
```

Slika 4.13. Korištenje Vuex store-a, izvor: *izrada autora*

Prilikom prijave korisnika u sustav, nakon uspješne autentikacije, funkcija *login* upravljača *UserController* kao jedno od svojstava (engl. *property*) postavlja objekt korisnika (prikazano slikom 4.12.). Postavljena svojstva dohvaćaju se prilikom kreiranja Vue aplikacije, nakon čega se pozivanjem mutacije *setUserLoginStatus* sa slike 4.13., mijenja stanje statusa trenutno prijavljenog korisnika te na taj način osigurava raspoznavanje prijavljenih korisnika u danome

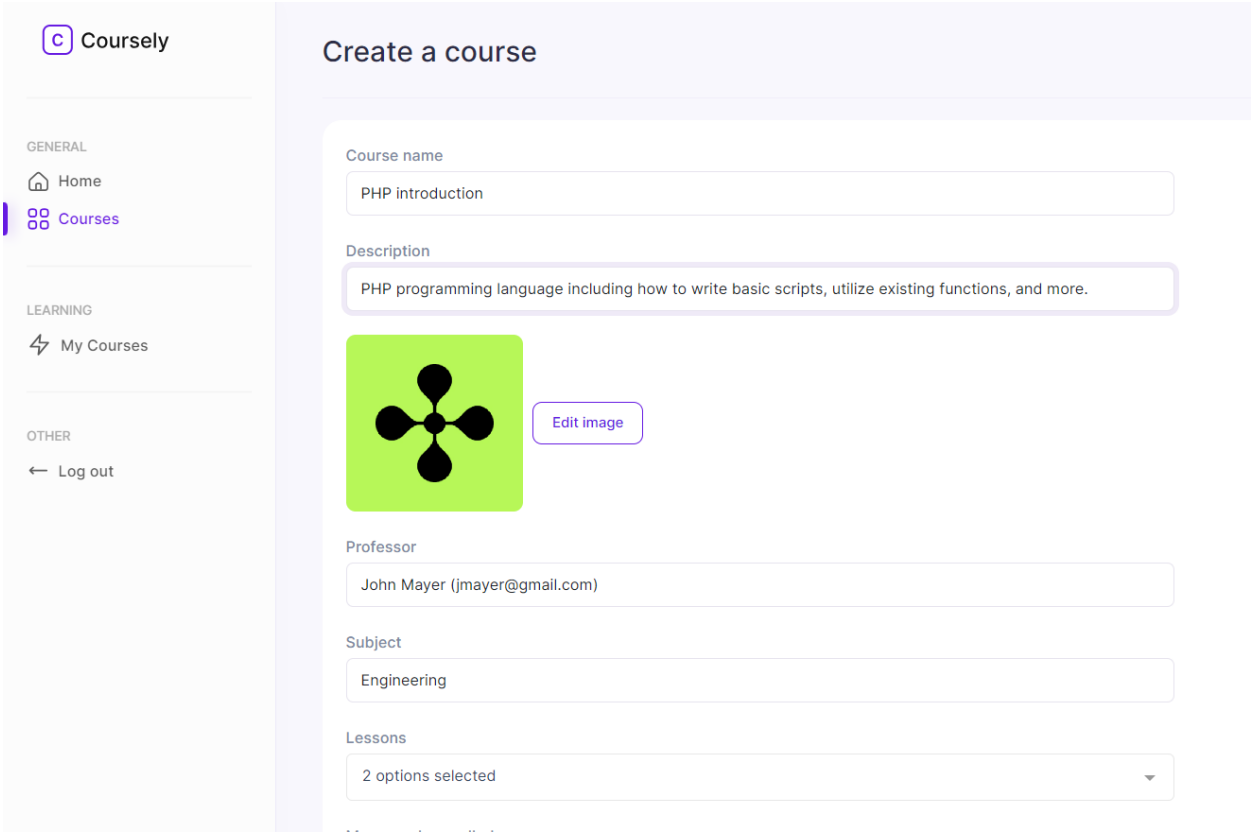
trenutku. Primjer dohvaćanja svojstva proslijeđenog iz Laravela u Vue, te postavljanje prijavljenog korisnika koristeći Vuex *store* prikazano je slikom 4.14., na kojoj se vidi dio sadržaja komponente *App.vue*.

```
export default {
  props: ["user"],

  created() {
    this.$store.commit("setUserLoginStatus", this.user);
    AOS.init();
  }
};
```

Slika 4.14. Dio sadržaja komponente App.vue, izvor: *izrada autora*

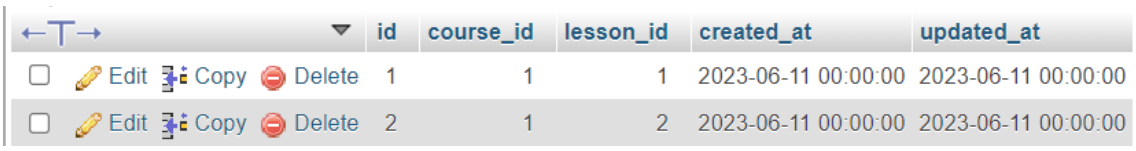
4.3. Kreiranje i prikaz tečajeva

Mogućnost kreiranja tečajeva dostupna je isključivo korisnicima s pripadajućom ulogom administratora sustava. Korisnici s ulogom predavača (odnosno profesora), ograničeni su na kreiranje i dodavanje novih lekcija u tečajeve, te kreiranje završnih provjera znanja. Navedena ograničenja prisutna su kako bi bilo moguće kontrolirati ponudu tečajeva prisutnih na platformi. Pritiskom na gumb *Create course*, administrator posjećuje stranicu na putanji */new-course*. Kako bi se uspješno kreirao tečaj, potrebno je unijeti podatke naziva i opisa tečaja, tematike na koju se odnosi (engl. *subject*), odabrati proizvoljan broj lekcija koje će biti dodijeljene tečaju (u slučaju da su spomenute lekcije već prethodno kreirane te dodijeljene jednom od ostalih tečajeva), maksimalan broj korisnika koji smiju biti upisani u pojedini tečaj, razinu kompleksnosti, te predavača koji će biti dodijeljen tečaju. Na slici 4.15. prikazana je forma na zaslonu stranice pomoću koje se odvija kreiranja novog tečaja.



Slika 4.15. Forma kreiranja novog tečaja, izvor: *izrada autora*

Prilikom spremanja tečaja u bazu podataka, odnosno unutar tablice *courses*, potrebno je dodijeliti *id* tečaja odabranim pripadajućim lekcijama. Kako jedna lekcija može pripadati mnogobrojnim različitim tečajevima, a svaki od tih tečaja sadrži jednu ili više lekcija, kreirana je dodatna tablica unutar baze *course_lesson*. Navedeni pristup koristi se u slučajevima postojanja veze više na više (engl. *many-to-many relationship*) između objekata baze podataka, prilikom čega je potrebno kreirati novu tablicu kako bi se objekti uspješno povezali. Tablica *course_lesson* sadrži strane ključeve (engl. *foreign keys*) *course_id* te *lesson_id* kao reference na postojeće primarne ključeve tečajeva i lekcija. Na taj način osigurava se povezanost određene lekcije sa svim tečajevima kojima ta lekcija pripada. Primjerom na slici 4.16. prikazana je pripadnost različitih lekcija jednome kvizu, odnosno prikaz spomenutoga u tablici *course_lesson*.



	id	course_id	lesson_id	created_at	updated_at
Edit Copy Delete	1	1	1	2023-06-11 00:00:00	2023-06-11 00:00:00
Edit Copy Delete	2	1	2	2023-06-11 00:00:00	2023-06-11 00:00:00

Slika 4.16. Prikaz tablice course_lesson, izvor: *izrada autora*

Tablica *course_lesson* kreirana je u Laravelu koristeći opciju *make:migration* Artisana, a kako bi se omogućilo dodavanje i ispis lekcija određenog tečaja, bilo je potrebno unutar modela tečaja dodati novu funkciju *lessons*, koja naznačuje pripadnost više lekcija. Unutar modela lekcije, također je bilo potrebno dodati funkciju koja naznačuje jednaku funkcionalnost, ali naziva *courses*. Na slici 4.17. prikazana je funkcija *lessons* modela *Course*, kojom se u nastavku pristupa prikazu svih lekcija određenog tečaja.

```
public function lessons()
{
    return $this->belongsToMany(Lesson::class, 'course_lesson');
}
```

Slika 4.17. Prikaz funkcije lessons, izvor: *izrada autora*

Za uspješno spremanje tečaja u bazu podataka, prilikom spremanja potrebno je, osim podataka vezanih za tečaj, spremiti poveznice između tečaja i dodijeljenih lekcija u tablicu *course_lesson*. To se može obaviti koristeći metodu *attach*, koja polju pripadajućih lekcija tečaja dodaje nove unose i na taj način pohranjuje podatke u željenu tablicu. Slikom 4.18. prikazan je način pohranjivanja lekcija tečaja u tablicu *course_lesson.*

```
$course = Course::create([
    'title' => $request->title,
    'numOfPeople' => $request->numOfPeople,
    'description' => $request->description,
    'user_id' => $request->user_id,
    'professor_id' => $request->professor_id,
    'difficulty' => $request->difficulty,
    'subject' => $request->subject
]);


for ($i = 0; $i < count($request->lessons); $i++) {
    $course->lessons()->attach($request->lessons[$i]);
}
```

Slika 4.18. Prikaz kreiranja tečaja i dodjela lekcija tečaju, izvor: *izrada autora*

S obzirom na to da korisnik ima mogućnost odabira više lekcija odjednom, potrebno je koristiti petlju *for* koja za broj lekcija provodi funkcionalnost dodjele pojedine lekcije tečaju. Prilikom

prikaza postojećih lekcija, postoji ograničenje koje dozvoljava prikaz isključivo lekcija sa jednakom tematikom kao što je odabrana tematika na tečaju. Na primjer, ako je tematika tečaja programiranje, bit će moguće dodati samo lekcije s tematikom programiranja na taj tečaj. Navedeno je postignuto izmjenom upravljača lekcija *LessonCotroller*, tako što je dodana nova funkcija koja kao povratni podatak sadrži samo one lekcije koje za *subject* imaju tematiku odabranu prilikom kreiranja tečaja. Tako dohvaćene lekcije u Vue dalje se prikazuju u multiselect bloku, te nakon odabira pohranjuju u polje i prosljeđuju kao podatak tečaja. Funkcija kojom se unutar Vue sprema tečaj, *saveCourse*, prikazana je slikom 4.19., gdje se vidi dodavanje polja lekcija podacima tečaja koji se prosljeđuju upravljaču.

```
async saveCourse() {
    this.isCreating = true;
    this.courseData.professor_id = JSON.stringify(
        this.selectedProfessor.id
    );
    this.courseData.subject = this.selectedSubject.title;

    this.courseData.lessons = this.selectedLessons;

    const res = await this.callApi(
        "post",
        "course/save-course",
        this.courseData
    );

    if (res.status === 201) {
        this.success("Course created successfully!");
        this.$router.push({ path: "/courses" });
    } else {
        console.log(res);
        this.swr();
    }
    this.isCreating = false;
},
```

Slika 4.19. Prikaz funkcije saveCourse unutar Vue, izvor: *izrada autora*

Svi tečajevi dodani u sustav prikazani su posjetom putanje */courses*, a pojedini tečaj detaljnije se može pogledati posjetom putanje */course/id*.

Ukoliko je trenutni korisnik sustava onaj s ulogom administratora, uz ostale mogućnosti, takvom se korisniku nudi i mogućnost uređivanja tečaja. Uređivanjem tečaja, administrator može

promijeniti dodijeljenog predavača, ime tečaja, uključene lekcije, te sve ostale podatke koji su potrebni i prilikom primarnog unosa tečaja. Posjetom putanje za uređivanje tečaja, forma se prilikom kreiranja komponente popunjava podacima tečaja čiji je *id* proslijeđen preko putanje, te se na takav način osigurava da korisnik zna što je prethodno bilo unešeno kako bi mogao odlučiti koje će informacije promijeniti. Prilikom uređivanja tečaja, ukoliko se promijeni odabir lekcija dodanih na tečaj, potrebno je ukloniti poveznice tečaja s prethodno odabranim lekcijama. Kako bi se osiguralo da se spomenute poveznice uklone samo ukoliko prethodne lekcije nisu ponovno odabrane za tečaj, umjesto metode *attach* korištene prilikom kreiranja tečaja, potrebno je koristiti metodu *sync*. Ukoliko se u ove svrhe koristi metoda *attach*, ne bi postojala mogućnost uklanjanja lekcija koje su prethodno odabrane, a prilikom uređivanja tečaja izostavljene. U slučaju da korisnik nije odabrao nove lekcije uređujući tečaj, poveznice prethodno odabranih lekcija s tečajem ostaju jednake. Slikom 4.20. prikazan je dio funkcije uređivanja tečaja upravljača *CourseController*, kojom se upravlja tablicom *course_lesson* u slučaju mijenjanja lekcija.

```php
$course = Course::where('id', $request->id)->first();

if (count($request->lessons) > 0) {
    $course->lessons()->sync($request->lesson_ids);
}
```

Slika 4.20. Usklađivanje odabranih lekcija, izvor: *izrada autora*

4.4. Kreiranje novih lekcija i dodavanje tema

U svrhu što bolje preglednosti obrazovnih tema, sadržaj tečajeva raspodijeljen je u manje cjeline koje se nazivaju lekcije. Svaka od lekcija sadrži naziv, podatke unesene od strane administratora ili predavača, te tematiku na koju se odnosi. Odabir tematike, kao i prilikom kreiranja tečaja, odvija se odabirom jedne od ponuđenih opcija svih postojećih tema. Uzimajući u obzir to da će korisnici samostalno unoseći nazive tema možda istu temu nazvati različitim imenima, što će rezultirati pogrešnim lekcijama ponuđenim za odabir prilikom kreiranja tečaja, u bazu podataka dodana je nova tablica *subjects*, koja definira sve teme ponuđene u sustavu. Na taj način izbjegava se korisnička greška te osigurava pravilan prikaz podataka na svim mjestima u sustavu. Prikaz tablice *subjects* nalazi se na slici 4.21., a tablica sadrži *id* teme, te njen naziv.

←T→	id	title	created_at	updated_at
Edit Copy Delete	1	Engineering	2023-06-11 00:00:00	2023-06-11 00:00:00
Edit Copy Delete	2	Business	2023-06-11 00:00:00	2023-06-11 00:00:00
Edit Copy Delete	3	Art	2023-06-11 00:00:00	2023-06-11 00:00:00

Slika 4.21. Prikaz tablice subjects, izvor: *izrada autora*

U svrhu kontroliranja tema koje portal nudi za edukaciju, dodavanje tema ograničeno je samo na korisnike s ulogom administratora.

Slično poput komponente za kreiranje tečaja, prilikom kreiranja komponente za dodavanje nove lekcije, potrebno je koristeći funkciju *callApi* dohvatiti listu postojećih tema. Vue to omogućava pružanjem metode *created*, koja se poziva automatski prilikom kreiranja Vue komponente. Metoda *created* te dohvaćanje tema prikazani su slikom 4.22..

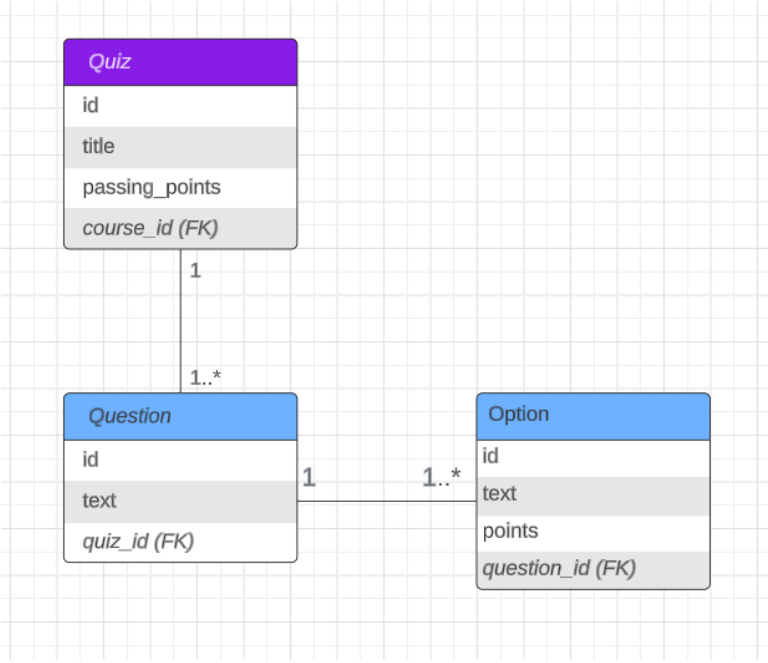
```
async created() {
    this.token = window.Laravel.csrfToken;

    const res = await this.callApi("get", "/get-subjects");
    if (res.status === 200) {
        this.subjectsList = res.data;
    }
},
```

Slika 4.22. Metoda created Vue komponente, izvor: *izrada autora*

Osim mogućnosti dodavanja tema, ključnu ulogu prilikom kreiranja tečaja predstavlja odabir jedne ili više lekcija koje će prilikom kreiranja biti pridružene tečaju. Spomenutu funkcionalnost može se koristiti ukoliko su lekcije prethodno dodane u sustav, što se može postići posjetom putanje *new-lesson*, te unosom svih podataka potrebnih za kreiranje nove lekcije. Prilikom kreiranja lekcije, moguće je priložiti dodatnu literaturu ili dokumentaciju odabirom pdf dokumenta, koji se nadalje pohranjuje u bazi podataka skupa s ostatkom informacija o lekciji. Također, dostupna je i mogućnost kreiranja lekcije za određeni tečaj ukoliko je trenutni korisnik onaj s ulogom administratora, ili predavač dodijeljen tečaju za koji želi kreirati lekciju. Prilikom kreiranja lekcije za određeni tečaj, postoji ograničenje koje predavaču ne dozvoljava promjenu teme lekcije, zbog toga što tečajevi smiju sadržavati isključivo lekcije s temom jednakom temi tečaja.

4.5. Završne provjere znanja

Uzimajući u obzir potrebu potvrde uspješnog završetka dodatne edukacije, ova platforma nudi opciju dodjeljivanja certifikata korisnicima koji uspješno završe upisane tečajeve. Kako bi se moglo odlučiti koji korisnici su usvojili naučeno u dovoljnoj mjeri da bi im ono pomoglo i koristilo u budućnosti, aplikacija nudi završne provjere znanja prilikom završetka tečaja. Svaka od provjera znanja vremenski je ograničena i može joj se pristupiti samo jednom, kako bi se osiguralo da korisnici ne ispunjavaju provjere znanja s ciljem dobivanja potvrde o završenom tečaju, bez prethodnog usvajanja znanja učenjem sadržaja lekcija. Završne provjere znanja, odnosno kvizovi, utemeljene su na proizvoljnom broju pitanja prikazanih korisniku, od kojih svako pitanje sadrži proizvoljan broj opcija koje korisnik može odabrati prilikom odgovaranja na pitanje. Svaka od ponuđenih opcija nosi određeni broj bodova, a svaki od kvizova nosi ukupan minimalni broj bodova koje korisnik treba ostvariti kako bi se njegova provjera znanja smatrala uspješno položenom. Prije same implementacije tablica potrebnih za ostvarenje provjera znanja, napravljen je dijagram veza entiteta (engl. *Entity Relationship Diagram*, ili skraćeno *ER Diagram*) u svrhu detaljnijeg razumijevanja veza između objekata. Dijagram veza entiteta, prema [33], tip je dijagrama koji vizualno prikazuje način na koji su ljudi, objekti ili koncepti međusobno povezani unutar sustava. Takvi dijagrami koriste prethodno definirani set oblika i simbola kako bi prikazali željenu strukturu sustava i atribute objekata. ER dijagram završnih provjera znanja ove aplikacije prikazan je slikom 4.23..



Slika 4.23. ER dijagram završnih provjera znanja, izvor: *izrada autora*

Prethodnom slikom (4.23.), prikazane su veze između tablica provjere znanja (*Quiz*), ponuđenih pitanja unutar kviza (*Question*), te svih ponuđenih odgovora pitanja (*Option*). Svaki od završnih ispita nudi proizvoljan broj pitanja, od kojih svako pitanje nudi proizvoljan broj odgovora. Svaki odgovor sadrži određeni broj bodova, a završni ispit kao jedan od podataka sadrži minimalni broj bodova potrebnih kako bi se ispit uspješno položio. Bodovi se zbrajaju korisnikovim odabirom ponuđenih odgovora prilikom rješavanja ispita, čime se postiže validacija da je korisnik zadovoljio uvjete i položio završni ispit. Kako bi se kviz uspješno kreirao, korisnik je dužan unijeti naziv kviza, minimalan broj bodova za uspješno polaganje ispita, te odabrati tečaj na kojeg se završni ispit odnosi. Kreiranje kviza unutar baze podataka odvija se pozivom funkcije *saveQuiz* upravljača *QuizController*, prikazane slikom 4.24..

```
21      public function saveQuiz(Request $request)
22      {
23          $quiz = Quiz::create([
24              'title' => $request->title,
25              'passing_points' => $request->passing_points,
26              'course_id' => $request->course_id,
27          ])->id;
28
29          return $quiz;
30      }
31
```

Slika 4.24. Pohrana kviza u bazu podataka, izvor: *izrada autora*

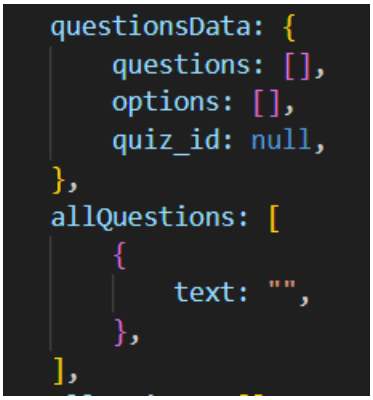
Osim podataka obveznih kako bi se kviz uspješno kreirao, korisnik ima mogućnost dodavanja proizvoljnog broja pitanja, te proizvoljno mnogo odgovora za svako od dodanih pitanja na istom zaslonu. U slučaju da se korisnik odluči dodati više pitanja na ispit, s obzirom na to da se kreiranje kviza odvija u isto vrijeme kao i kreiranje pitanja dodanih na isti, prilikom kreiranja završnog ispita potrebno je pohraniti *id* novo kreiranog objekta kviza, te isti nakon toga dohvatiti unutar Vue komponente kako bi se mogao pridružiti pitanjima korisnika. Ukoliko se spomenuti korak izostavi, neće postojati mogućnost istovremenog kreiranja novih pitanja, jer je za njihovo kreiranje (kao što je prikazano ER dijagramom na slici 4.23.) obvezno odabrati kojem završnom ispitu pripadaju. Navedeno je postignuto koristeći metodu *id* prikazanu linijom 27 slike 4.24.. Povratni podatak iz Laravela, varijabla *quiz* (linija 29, slika 4.24.) predstavlja upravo *id* prethodno kreiranog kviza. U sljedećem koraku, unutar Vue komponente kreiranja kviza (*NewQuiz.vue*), iz povratnih podataka odabire se proslijeđeni *id*, te dodjeljuje objektu novo dodanih pitanja, kako bi se pitanja uspješno spremila u bazi podataka. Slika 4.25. prikazuje prethodno opisanu akciju linijom 205, uz

što se istovremeno može vidjeti i spremanje, odnosno prosljeđivanje svih podataka završnog ispita iz Vue komponente upravljaču *QuizController*.

```
192        async saveQuiz() {
193            this.isCreating = true;
194            this.quizData.course_id = this.selectedCourse.id;
195
196            const res = await this.callApi(
197                "post",
198                "quiz/save-quiz",
199                this.quizData
200            );
201
202            if (res.status === 201 || res.status === 200) {
203                this.info("Saving...");
204
205                this.questionsData.quiz_id = res.data;
206                this.saveQuestions();
207
208            } else {
209                console.log(res);
210                this.swr();
211            }
212
213            this.isCreating = false;
214        },
```

Slika 4.25. Slanje podataka kviza upravljaču QuizController, izvor: *izrada autora*

Dodavanje proizvoljnog broja pitanja postignuto je korištenjem *v-for* smjernice (engl. *directive*) dostupne korištenjem okvira Vue pomoću koje se osigurava prikaz svih trenutno dodanih pitanja, uz istovremenu mogućnost dodavanja prazne opcije objekta pitanja unutar polja prikazanog *v-for* smjernicom. Na taj se način osigurava prikaz svih pitanja s već ispunjenim podacima, ali i najnovijeg pitanja čiji se podaci tek trebaju definirati. Svaka novo kreirana opcija pitanja kao jedini podatak prima samo tekst pitanja, dok se podatak *quiz_id* nalazi se u objektu koji sadrži listu svih novo kreiranih pitanja. Prethodno spomenuta lista, naziva *allQuestions*, posebna je lista unutar Vue komponente koja služi isključivo za dodavanje novih pitanja (zbog pojednostavljenja čitljivosti i jasnoće programskog koda), a netom prije prosljeđivanja objekta podataka pitanja (*questionsData*) upravljaču, ta se lista sprema unutar navedenog objekta, te na taj način prosljeđuje upravljaču *QuestionController*. Slika 4.26. prikazuje formate opisanih podataka.



Slika 4.26. Prikaz formata podataka za pohranjivanje pitanja, izvor: *izrada autora*

Dodavanje novog pitanja u listu pitanja prikazanih na sučelju, omogućeno je funkcijom *addQuestion*, kojom se listi *allQuestions* dodaje nova opcija pitanja korištenjem metode *push*. Također, postoji mogućnost brisanja dodanih pitanja metodom *splice*. Spomenute metode postoje kao dio JavaScripta, te se koriste za dodavanje elemenata na kraj polja, odnosno promjenu sadržaja polja brisanjem ili premještanjem sadržaja na određeno mjesto u polju. U svrhu brisanja određenog elementa, metodi *splice* potrebno je proslijediti redni broj elementa koji se treba obrisati. Funkcije *addQuestion* i *removeQuestion* prikazane su na slici 4.27..

```
removeQuestion(index) {
    this.allQuestions.splice(index, 1);
},

addQuestion() {
    let questions = [
        {
            text: "",
        },
    ];
    this.allQuestions.push(questions);
},
```

Slika 4.27. Funkcije addQuestion i removeQuestion, izvor: *izrada autora*

Slično načinu dodavanja pitanja, kreiranje i brisanje proizvoljnog broj opcija dodijeljenih svakom pitanju, odvija se funkcijama *addOption*, te *removeOption*. Razlika između funkcije za dodavanje pitanja *addQuestion,* te funkcije za dodavanja opcije *addOption*, parametar je rednog broja koji se treba proslijediti prilikom pozivanja funkcije *addOption* i kreiranja nove opcije. Kako bi se u bazi podataka uspješno kreirala nova opcija (novi ponuđeni odgovor), ona mora kao podatak nositi *id*

pitanja (*question_id*) kojemu pripada. S obzirom na to da u ovom slučaju, prilikom kreiranja opcija, pitanja još nisu pohranjena u bazi podataka, a podatak *id* kreira se tek prilikom pohrane podatka u bazu, potrebno je na neki način odrediti koje opcije će pripadati kojem pitanju. Iz tog razloga, prilikom kreiranja nove opcije, prosljeđuje se redni broj pitanja za koje je ta opcija dodana. Taj redni broj može se dobiti iz *v-for* smjernice korištene za prikaz svih pitanja, te on ne označava stvarni *id* pitanja, već se koristi u svrhu pohrane opcija prilikom spremanja pitanja unutar *QuestionController*-a. Slika 4.28. prikazuje funkcije *addOption* te *removeOption*.

```
removeOption(idx) {
    this.allOptions.splice(idx, 1);
},

addOption(index) {
    let options = [
        {
            text: "",
            points: "",
            quest_id: index,
        },
    ];
    this.allOptions.push(...options);
},
```

Slika 4.28. Funkcije addOption i removeOption, izvor: *izrada autora*

Sve dodane opcije, kao i sva dodana pitanja, na korisničkom sučelju prikazuju se korištenjem *v-for* smjernice. Međutim, dodavanjem funkcionalnosti prikaza svih opcija unutar već postojeće smjernice *v-for* (korištene za prikaz svih pitanja), dolazi do problema u kojem se sve dodane opcije prikazuju za svako od dodanih pitanja. Kako bi se funkcionalnost dodavanja i prikaza opcija vezanih isključivo za određeno pitanje mogla uspješno izvesti, na način da se sve novo dodane opcije prikazuju samo unutar pitanja za koje su namijenjene, te nigdje dalje na zaslonu, unutar smjernice *v-for* prikaza svih opcija korištena je i dodatna smjernica *v-show*, kojom se u ovisnosti o određenom uvjetu element može prikazati ili sakriti sa sučelja. U ovom slučaju, uvjet potreban kako bi navedeno ponašanje pravilno funkcioniralo, bio je uspoređivanje rednog broja proslijeđenog kao parametra funkciji *addOption*, s trenutnim rednim brojem elementa kojeg petlja prikazuje. Ukoliko su redni brojevi jednaki, sve opcije namijenjene pitanju s tim rednim brojem bit će prikazane, a u slučaju prikaza pitanja s nekim drugim rednim brojem, opcije će biti skrivene (odnosno, bit će prikazane samo opcije vezane za to određeno pitanje). Slike 4.29. i 4.30. prikazuju

dvije petlje kojima se prikazuju sva dodana pitanja, te svi dodani odgovori. Također, navedene slike prikazuju korištenje prethodno spomenutih funkcija (*addQuestion*, *addOption*, *removeQuestion*, *removeOption*) te Vue smjernica.

```
<Button class="c-btn--primary c-btn--base" @click="addQuestion()"
  >Add new question</Button
>
<b-row
  v-for="(item, index) in allQuestions"
  v-bind:item="item"
  v-bind:index="index"
  v-bind:key="item.id"
  class="c-question"
>
  <Button
    class="c-btn--secondary c-btn--base"
    @click="addOption(index)"
    >Add option</Button
  >

  <b-col cols="1">
    <label><b style="u-type--text">Question Text</b></label>
    <b-form-input v-model="item.text"></b-form-input>
  </b-col>

  <b-row
    v-for="(it, idx) in allOptions"
    v-bind:item="it"
    v-bind:index="idx"
    v-bind:key="it.id"
    class="c-option"
    v-show="it.quest_id == index"
  >
    <b-col cols="1">
      <label>
        <b style="u-type--text"> Option Text </b>
      </label>
      <b-form-input v-model="it.text"></b-form-input>
```

Slika 4.29. Prikaz dodanih pitanja i odgovora, izvor: *izrada autora*

```
    <b-col cols="1">
      <label>
        <b style="u-type--text"> Option Text </b>
      </label>
      <b-form-input v-model="it.text"></b-form-input>

      <label><b style="u-type--text">Option Points</b></label>
      <b-form-input v-model="it.points" type="number"></b-form-input>
    </b-col>

    <Button
      v-if="idx > 0"
      class="c-btn--outline c-btn--base"
      @click="removeOption(idx)"
      >Remove option</Button
    >
  </b-row>

  <Button
    v-if="index > 0"
    class="c-btn--outline c-btn--base"
    @click="removeQuestion(index)"
    >Remove question</Button
  >
</b-row>
```

Slika 4.30. Prikaz dodanih pitanja i odgovora, izvor: *izrada autora*

Nakon što korisnik doda željena pitanja i odgovore provjeri znanja, oni se pritiskom gumba *Save*, te pozivanjem funkcije *saveQuestions*, prosljeđuju *QuestionController* upravljaču, iz kojega se u nastavku pohranjuju u tablice *question*, odnosno *option* baze podataka. Slika 4.31. prikazuje funkciju *saveQuestions* Vue komponente.

```
  async saveQuestions() {
    const res3 = await this.callApi(
      "post",
      "question/save-multiple-questions",
      this.questionsData
    );

    if (res3.status === 201 || res3.status === 200) {
      this.success("Quiz created successfully");
    }
  },
},
```

Slika 4.31. Prosljeđivanje pitanja i odgovora upravljaču, izvor: *izrada autora*

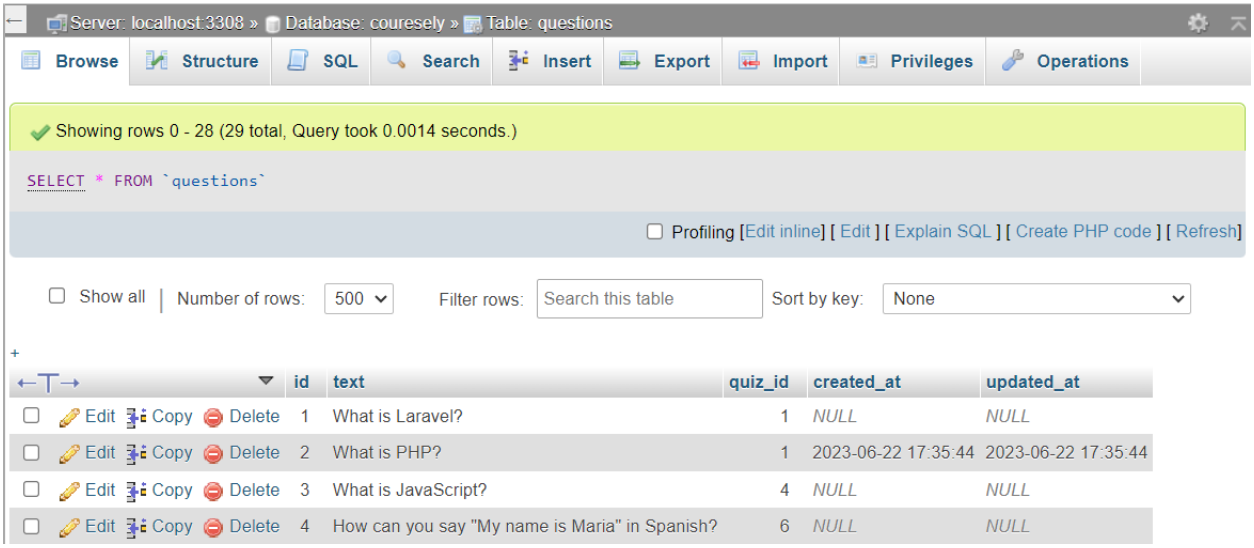
Funkcija koja se poziva putanjom *question/save-multiple-questions* funkcijom *callApi*, zaslužna je za pohranu više od jednog pitanja u bazu podataka. Toj funkciji predaje se parametar *questionsData* koji sadrži sva pitanja koja je korisnik dodao, sve dodane opcije te njihove pripadajuće redne brojeve pitanja kojemu su dodijeljene, te *id* upravo kreirane provjere znanja. Kako bi se u bazu podataka uspješno pohranili svi elementi polja, potrebno je, iterirajući kroz *for* petlju, pohranjivati jedno po jedno pitanje. S obzirom na to da opcije koje trebaju istovremeno biti pohranjene moraju sadržavati stvarni *id* pripadajućeg pitanja (onaj koji će tek biti kreiran unutar *for* petlje, te pohranjen u tablici pitanja), a trenutno sadrže redni broj pitanja za koje su dodane, uvođenjem ugniježđene (engl. *nested*) *for* petlje, dobiva se mogućnost usporedbe rednog broja pitanja trenutne iteracije *for* petlje, sa rednim brojem koje sadrži svaka od opcija. Ukoliko je taj redni broj jednak, opciji se dodjeljuje *question_id* jednak upravo kreiranom *id*-ju pitanja, te se opcija pohranjuje u bazi podataka. Slika 4.32. prikazuje opisani postupak pohranjivanja pitanja i opcija u bazu podataka.

```
30      public function saveMultipleQuestions(Request $request)
31      {
32          for ($i = 0; $i < count($request->questions); $i++) {
33              $question = Question::insertGetId([
34                  'text' => $request->questions[$i]['text'],
35                  'quiz_id' => $request->quiz_id,
36              ]);
37
38              for ($j = 0; $j < count($request->options); $j++) {
39                  if ($request->options[$j]['quest_id'] == $i) {
40                      Option::insert([
41                          'text' => $request->options[$j]['text'],
42                          'points' => $request->options[$j]['points'],
43                          'question_id' => $question,
44                      ]);
45                  }
46              }
47          }
48
49          return $question;
50      }
```

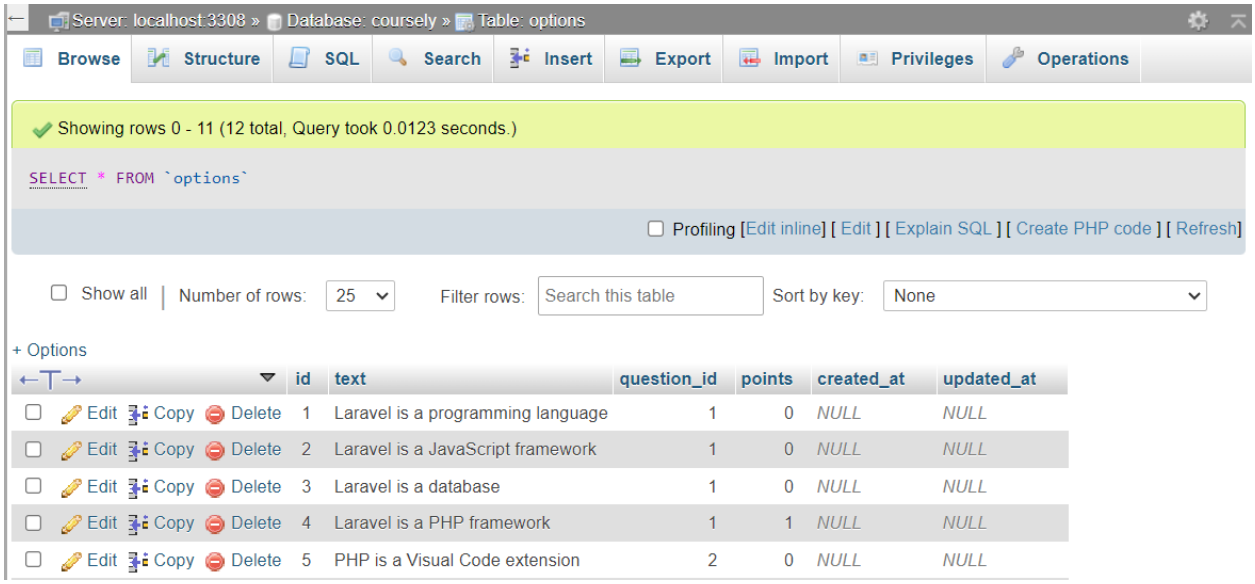
Slika 4.32. Pohranjivanje proizvoljnog broja pitanja i odgovora u bazu podataka, izvor: *izrada autora*

Linija 32 slike 4.32., prikazuje roditeljsku *for* petlju kojom se iterira kroz dobiveno polje pitanja. Redni brojevi pridruženi svakom od elemenata polja kreću se od 0, pa sve do kraja duljine polja

pitanja. Podatak duljine polja pitanja dobiva se korištenjem metode *count*. Prilikom umetanja pitanja u bazu podataka, koristi se funkcija *insertGetId*, koja varijabli *question* daje povratni podatak *id*-ja posljednje kreiranog pitanja. Na taj je način omogućeno pridruživanje tog *id*-ja podatku *question_id* potrebnom za pohranu opcije u bazu podataka. Linijom 38, prikazana je ugniježđena (engl. *nested*) *for* petlja zaslužna za pohranjivanje opcija. Slično roditeljskoj petlji, ona se kreće od 0 pa sve do kraja duljine polja opcija. Na liniji 39 iste slike, odvija se prethodno opisana provjera rednog broja opcije i pitanja. Kako i unutar Vue komponente redni brojevi kreću od 0 pa sve do kraja duljine polja pitanja, opcijama pridruženi redni brojevi (*quest_id*) jednaki su rednim brojevima roditeljske *for* petlje (varijabla *i*), odnosno redni brojevi opcija vezani za određeno pitanje (*quest_id* podatak svake opcije), jednaki su pitanjima s istim rednim brojem roditeljske *for* petlje (pitanje pod rednim brojem varijable *i*). Ukoliko su dva spomenuta redna broja jednaka, znači da određena opcija ugniježđene petlje pripada pitanju trenutne iteracije roditeljske petlje. Nakon što se prethodno uspješno utvrdi, opciju se pohranjuje u bazu podataka uz pripadajući podatak *question_id* kojeg zapravo predstavlja prethodno dobiven povratni podatak varijable *question*. Tablice pitanja (*questions*) i odgovora (*options*) iz baze podataka prikazane su slikama 4.33., te 4.34..

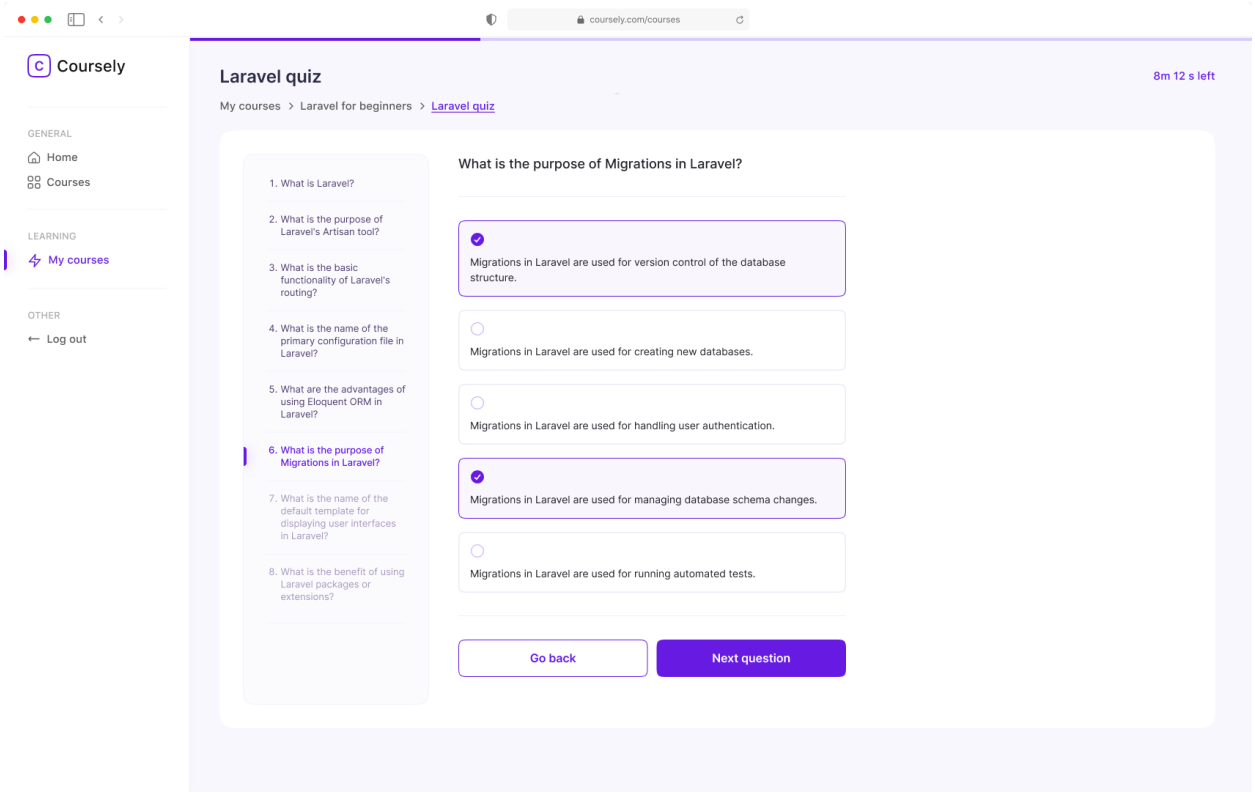


Slika 4.33. Tablica pitanja, questions, izvor: *izrada autora*



Slika 4.34. Tablica odgovora, options, izvor: *izrada autora*

Slikom 4.35. prikazano je korisničko sučelje završnog ispita s pripadajućim pitanjima, odgovorima, te prethodno navedenim ograničenjima (vremensko ograničenje te sveukupni broj pristupa ispitu).



Slika 4.35. Zaslon završnog ispita, izvor: *izrada autora*

4.7. Polaganje i završetak tečaja pojedinog korisnika

Registracija s ulogom studenta korisnicima donosi mogućnosti poput polaganja tečaja, preuzimanje objavljenih materijala lekcija, te u konačnici preuzimanje službenog certifikata ukoliko je tečaj uspješno završen. Posjetom putanje *courses*, studentu je prikazana lista svih postojećih tečajeva, od kojih svaki sadrži opciju upisivanja. Odabirom spomenute opcije, kreira se novi redak unutar tablice *user_course*, koja od podataka pohranjuje *id* korisnika, te *id* odabranog tečaja. Nakon uspješnog upisa u tečaj, korisnik je preusmjeren na putanju *my-courses*, gdje se nalazi popis svih tečajeva u koje je korisnik upisan. Korisnik s ulogom predavača u sustavu, posjetom putanje *my-courses* vidi sve tečajeve na kojima je on dodijeljen kao predavač. Prilikom kreiranja novog unosa u tablicu, potrebno je pozvati funkciju *enrollInCourse* upravljača *UserController*, prikazanu slikom 4.36..

```php
public function enrollInCourse(Request $request)
{
    $course = Course::where('id', $request->course_id)->first();
    $user = User::where('id', $request->user_id)->first();

    $user->user_courses()->attach($course->id);
}
```

Slika 4.36. Upis korisnika u tečaj, izvor: *izrada autora*

Posjetom liste upisanih tečajeva, korisniku se nudi mogućnost filtriranja tečajeva na temelju tema koje postoje u sustavu. Ukoliko se tečaj odnosi na odabranu temu, korisnik s lakoćom može pristupiti odabranom tečaju. Na slici 4.37. prikazana je funkcija koja omogućuje filtriranje na temelju odabranih tema. Uz promjenu prikaza tečajeva izmjenom podatka *courses*, potrebno je i označiti koji od odjeljaka je trenutno odabran, odnosno sve ostale prikazati neaktivnim.

```javascript
async getCoursesForSubject(subject) {
    const res = await this.callApi(
        "get",
        "/get-courses-for-subject/" + subject.title
    );
    if (res.status === 200) this.courses = res.data;

    this.selectedSubject = subject.title;
    this.activeTab = this.selectedSubject;
},
```

Slika 4.37. Filtriranje tečajeva, izvor: *izrada autora*

Posjetom pojedinog tečaja, korisnik ima mogućnost preuzeti dokumente priložene lekcijama u svrhu što veće prilagodljivosti prilikom edukacije. Navedeno je omogućeno korištenjem *downloadPdf* funkcije, koja posjetom putanji dokumenta omogućuje brzo preuzimanje datoteke. Slika 4.38. prikazuje *downloadPdf* funkciju, koja za parametar prima putanju do odabranog dokumenta. Prilikom preuzimanja dokumenta, potrebno je kreirati putanju *fileURL* koja predstavlja objekt proslijeđen u parametru. U ovom slučaju taj objekt je tipa *Blob*, često korišten u svrhe pohranjivanja datoteka ili multimedijskih sadržaja.

```
methods: {
    downloadPdf(lessonPdf) {
        axios({
            url: "http://localhost:8000" + lessonPdf,
            method: "GET",
            responseType: "blob",
        }).then((response) => {
            var fileURL = window.URL.createObjectURL(
                new Blob([response.data])
            );
            var fileLink = document.createElement("a");

            fileLink.href = fileURL;
            fileLink.setAttribute("download", "lesson-pdf.pdf");
            document.body.appendChild(fileLink);

            fileLink.click();
        });
    },
```

Slika 4.38. Preuzimanje dokumenta, izvor: *izrada autora*

Također, posjetom pojedinog tečaja u koji je korisnik upisan, korisniku se odobrava mogućnost polaganja završne provjere znanja. S obzirom na to da je provjeru znanja moguće obaviti samo jedan puta, ova funkcionalnost bit će onemogućena nakon što korisnik prvi puta završi ispit, no njegovi rezultati ostat će dostupni i nakon završetka ispita. Uzimajući u obzir broj završnih ispita koje pojedini korisnik može polagati, te broj korisnika koji može polagati pojedini završni ispit, utvrđena je veza između tih dviju uloga zbog koje je potrebno kreirati novu tablicu *user_quiz*, prikazanu slikom 4.39..

	id	user_id	quiz_id	total_points	result	created_at	updated_at
Edit Copy Delete	1	1	1	2	pass	NULL	NULL
Edit Copy Delete	2	2	1	2	pass	NULL	NULL
Edit Copy Delete	3	3	1	2	pass	NULL	NULL
Edit Copy Delete	4	7	65	4	pass	NULL	NULL

Slika 4.39. Tablica rezultata korisnika, izvor: *izrada autora*

Ova tablica, osim korisnikovog *id*-ja, te *id*-ja pojedinog završenog ispita, sadrži i dodatna polja *total_points*, te *result*. Polje *result* koristi se prilikom preusmjeravanja korisnika na zaslon rezultata nakon polaganja ispita, gdje se u ovisnosti o rezultatu (*pass* ili *fail*) korisniku određenim bojama prikazuje je li uspješno položio završni ispit. Slika 4.40. prikazuje izlistanje opcija odabranog pitanja na zaslonu korisnika korištenjem *v-for* smjernice, te mogućnost odabira pojedine opcije ukoliko korisnik smatra da je ona točan odgovor na pitanje. Metodom *selectOption*, opcija se dodaje polju *selectedOptions*, a ukupni rezultat korisnika se povećava za broj bodova dodijeljen odabranoj opciji. Ukoliko opcija već postoji u polju, ona se briše iz polja te se ukupni broj bodova smanjuje za broj bodova opcije. U ovisnosti o tome postoji li opcija u spomenutom polju, kartici koja na korisničkom zaslonu predstavlja opciju, dodjeljuju se različite klase s ciljem vizualnog označavanja odabranih (ili neodabranih) opcija.

```
<div
    class="c-container--cards"
    v-if="this.questionContent"
>
    <div
        v-for="(questionOption, ind) in questionOptions"
        :key="ind"
        @click="selectOption(questionOption)"
        class="c-card c-card--base c-card--secondary"
        :class="
            selectedOptions.includes(questionOption)
                ? 'c-card--active'
                : ''
        "
    >
        <div class="c-card__check"></div>
        <p>{{ questionOption.text }}</p>
    </div>
</div>
```

Slika 4.40. Odabir opcije, izvor: *izrada autora*

Nakon što je korisnik dao odgovore na ponuđena pitanja (ili nakon što istekne vrijeme dozvoljeno za rješavanje ispita), kreira se objekt koji sadrži podatke korisnikovog *id*-ja, sveukupno ostvarenog broja bodova, *id*-ja tečaja, te broja bodova potrebnih za uspješno polaganje tečaja. Ti podaci se, nadalje, proslijeđuju upravljaču *UserController*, koji provodi dodatne operacije nad podacima, te ih naposljetku sprema u tablicu *user_quiz*. Slika 4.41. prikazuje funkciju *saveUserResults* spomenutog upravljača, te način pohranjivanja podataka u tablicu koja sadrži podatke osim *id*-ja korisnika i ispita.

```php
public function saveUserResults(Request $request)
{
    $user = User::where('id', $request->user_id)->first();

    if ($request->total >= $request->passing) {
        $res = "pass";
    } else $res = "fail";

    $user->user_quizzes()->attach($request->quiz_id, ['total_points' => $request->total, 'result' => $res]);

    return $user;
}
```

Slika 4.41. Pohrana rezultata korisnika, izvor: *izrada autora*

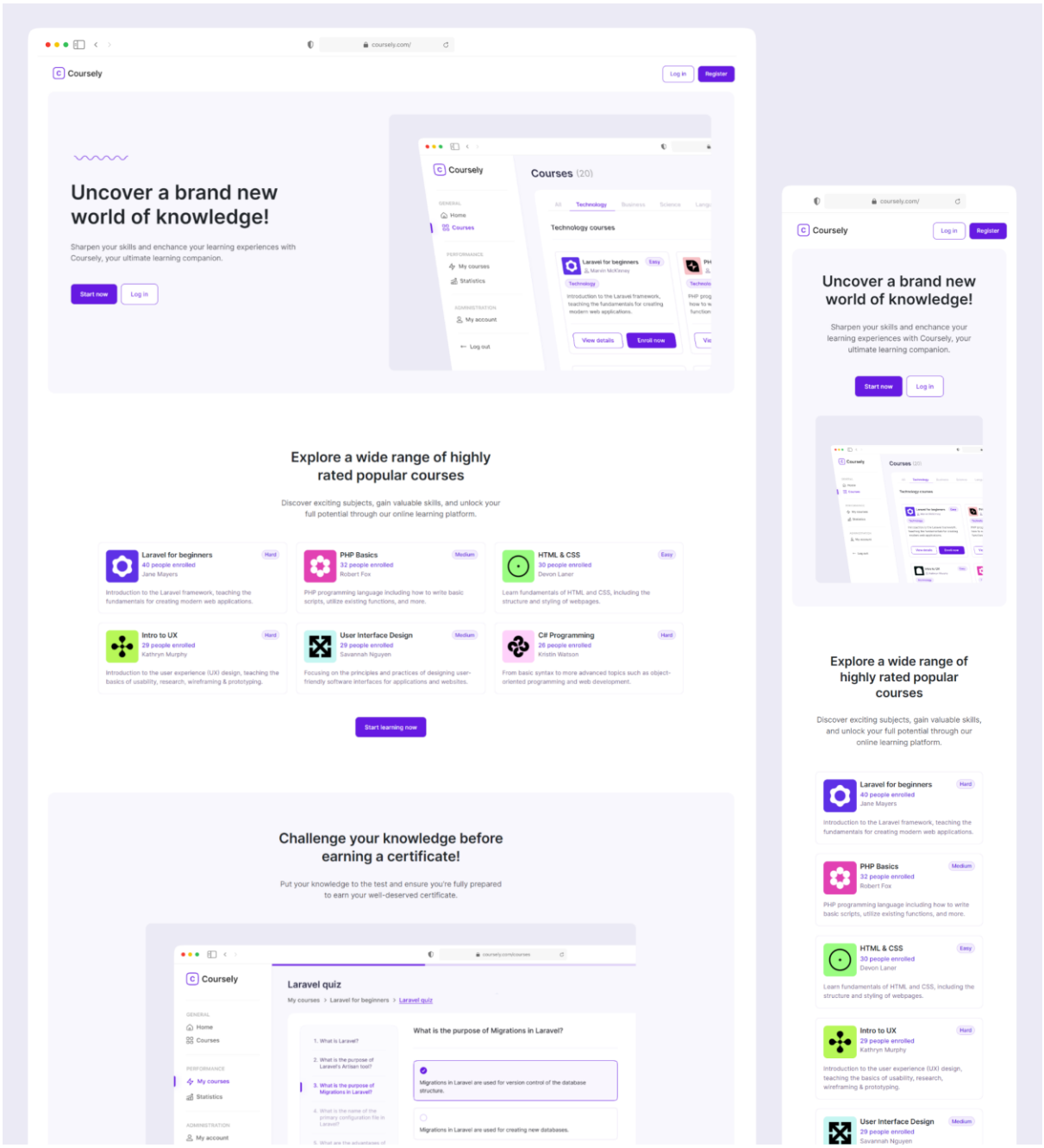
Pohranom rezultata korisnika, korisnik je preusmjeren na stranicu rezultata koja mu prikazuje je li položio završni ispit ili ne. Ukoliko završni ispit nije položen, podaci će se prikazati u crvenim nijansama, a ilustracija prikazana na zaslonu također će biti crvena. U slučaju da je korisnik položio ispit, navedene boje zamijenit će se nijansama primarne boje korištene u sustavu. Ukoliko je korisnik uspješno položio završni ispit, omogućeno mu je preuzimanje dokumenta certifikata s njegovim podacima, te podacima o položenom tečaju. U suprotnom, korisnik se može vratiti na stranicu tečajeva. Spomenuti korisnički zaslon prikazan je poglavljem 5, uz ostale zaslone aplikacije.

5. PRIKAZ RADA APLIKACIJE

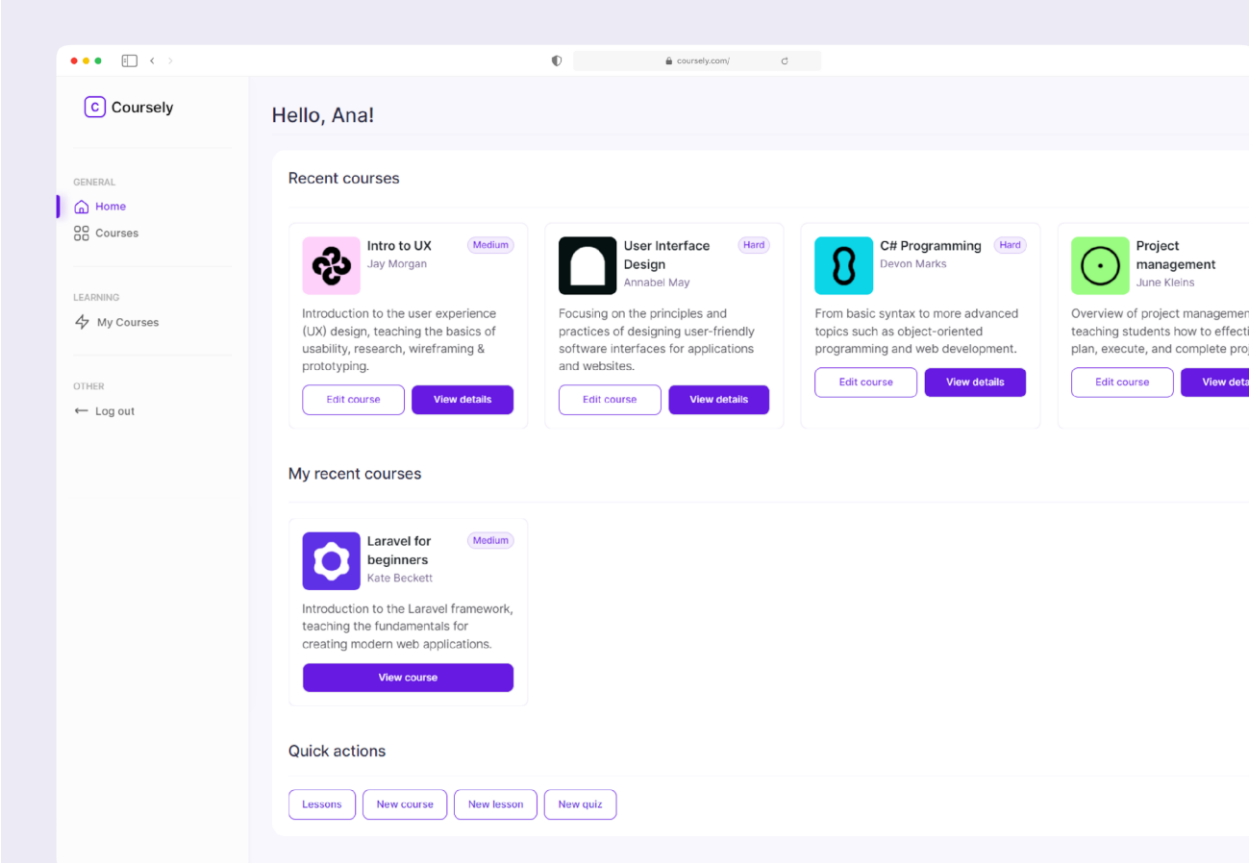
U svrhu strukturiranja i prikaza podataka na način na koji je korisnicima pažnja prvenstveno usmjerena na tečajeve i lekcije, odnosno na glavni sadržaj aplikacije, korisničko je sučelje kreirano s ciljem postizanja jednostavnosti te lake snalažljivosti korisnika. U potpoglavljima u nastavku prikazani su korisnički zasloni nekih od stranica aplikacije, uz dodatna pojašnjenja ciljeva.

5.1. Početna stranica

U sustavu u kojem posjetitelji stranice mogu biti korisnici koji nemaju napravljen račun u aplikaciji ili oni koji imaju napravljen račun ali nisu prijavljeni, te trenutno prijavljeni korisnici, potrebno je osigurati različit izgled početne stranice za svaku od tih opcija. Ukoliko korisnik nije prijavljen u sustav, on ima mogućnost pregleda početne stranice koja prikazuje općenite podatke o aplikaciji. U slučaju da je korisnik prijavljen u aplikaciju, njegova početna stranica (u navigaciji *Home*) sadržavat će skup nedavno dodanih tečajeva, te nekolicinu tečajeva u koje se korisnik nedavno upisao. Također, ovisno o ulozi korisnika u sustavu, administratori imaju ponuđenu opciju brzo dostupnih akcija koje uključuju prečace za dodavanje lekcija, tečajeva, kvizova i slično. Slika 5.1. prikazuje početnu stranicu korisnika koji nije prijavljen u sustav, dok slika 5.2. prikazuje početnu stranicu korisnika s ulogom administratora koji je trenutno prijavljen u sustav.



Slika 5.1. Početna stranica neprijavljenog korisnika, izvor: *izrada autora*

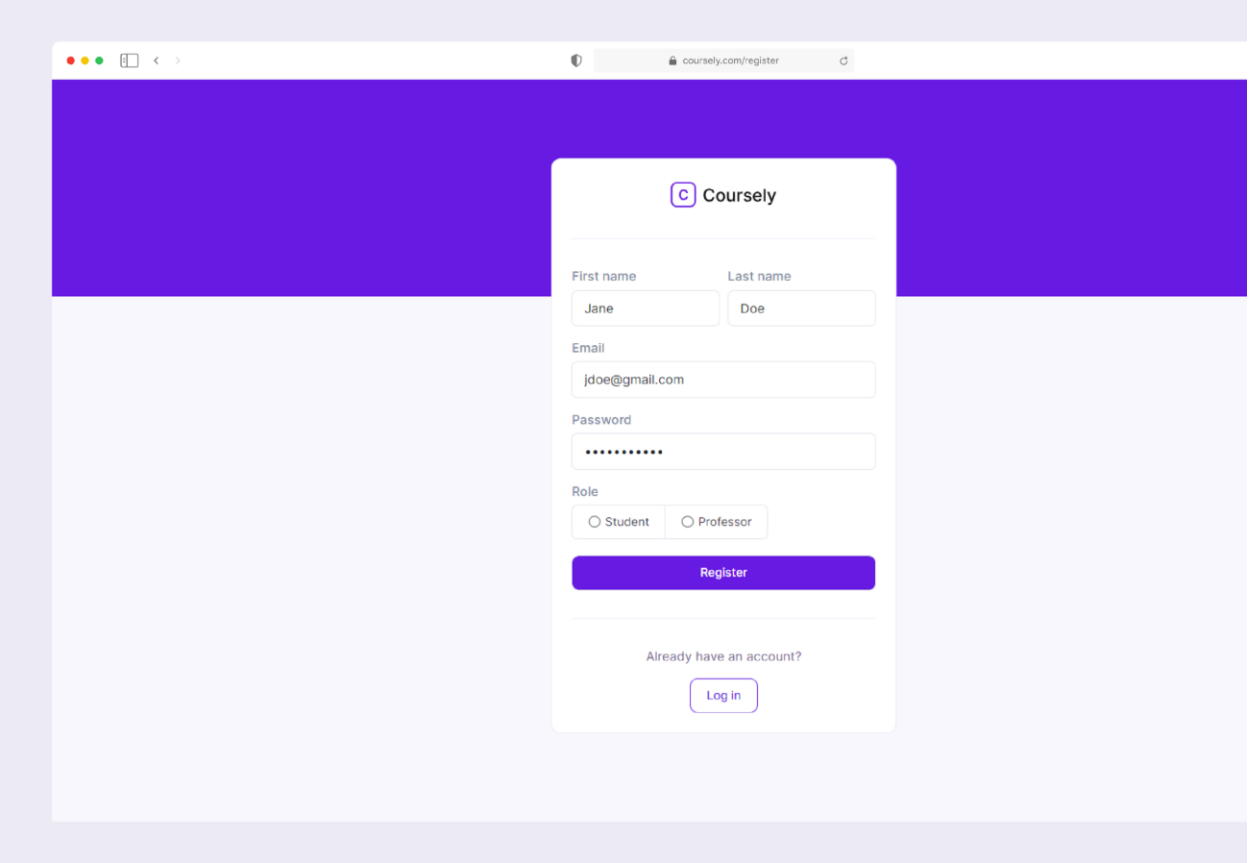


Slika 5.2. Početna stranica prijavljenog korisnika, izvor: *izrada autora*

Kako bi iskustvo bilo omogućeno i na manjim ekranima, korištene su metode prilagodbe elemenata zaslona i za manje uređaje, omogućujući pristup početnim stranicama za raznovrsne rezolucije uređaja.

5.2. Registracija korisnika

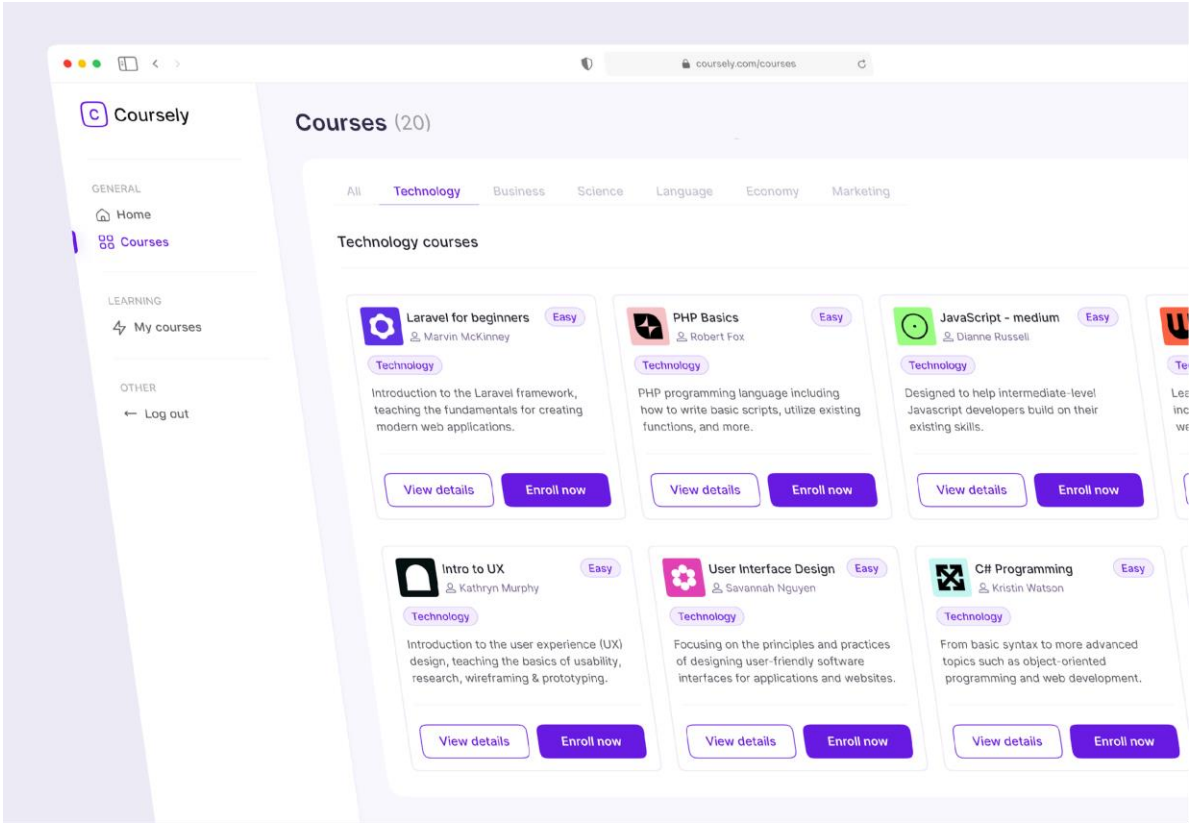
Za kreiranje računa u sustavu, potrebno je stisnuti na gumb *Register* na početnoj stranici, ili na gumb *Log in*, te naknadno sa stranice za prijavu posjetiti lokaciju za kreiranje računa. Zaslon za registraciju korisnika prikazan je slikom 5.3., dok je zaslon za prijavu korisnika vizualno jednaka forma, s podacima za unos adrese elektroničke pošte i lozinke.



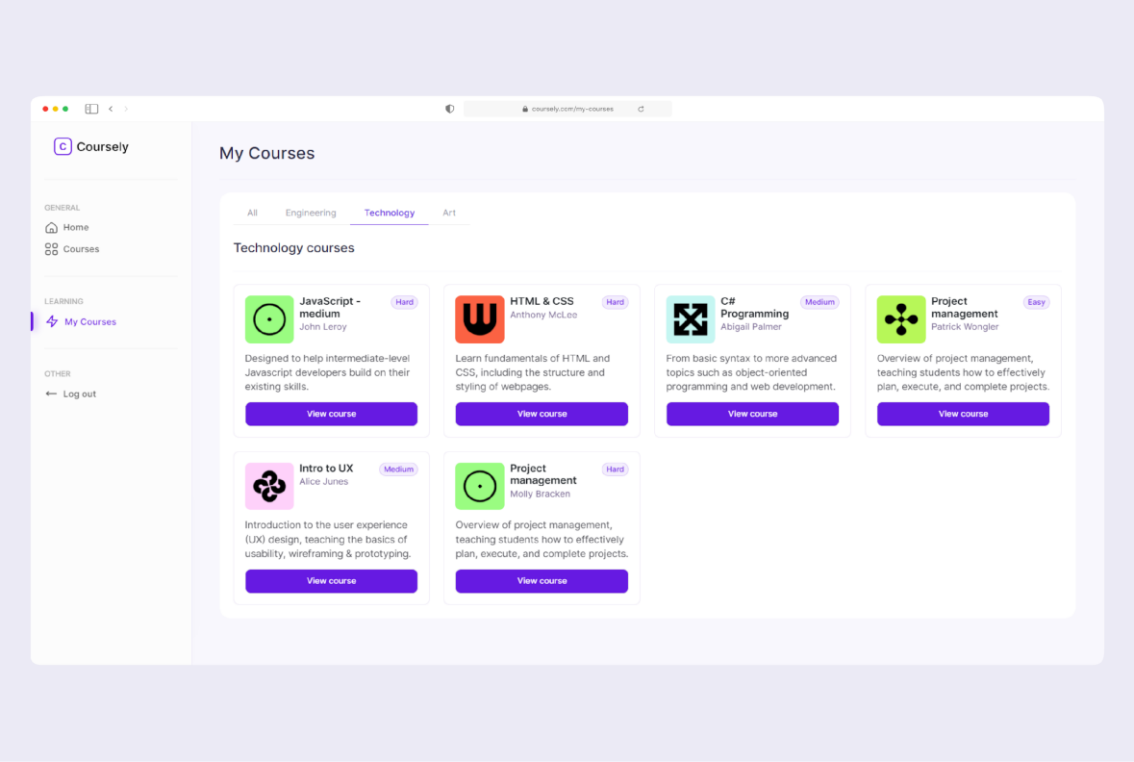
Slika 5.3. Registracija korisnika, izvor: *izrada autora*

5.3. Zasloni tečajeva

Upisom korisnika u tečaj, nudi se mogućnost pristupa stranici *My courses*. Slikama 5.4. i 5.5., prikazani su tečajevi stranice *Courses*, kao i stranice *My courses*. Korisnik prijavljen prilikom kreiranja navedenih slika zaslona, koristi ulogu studenta.

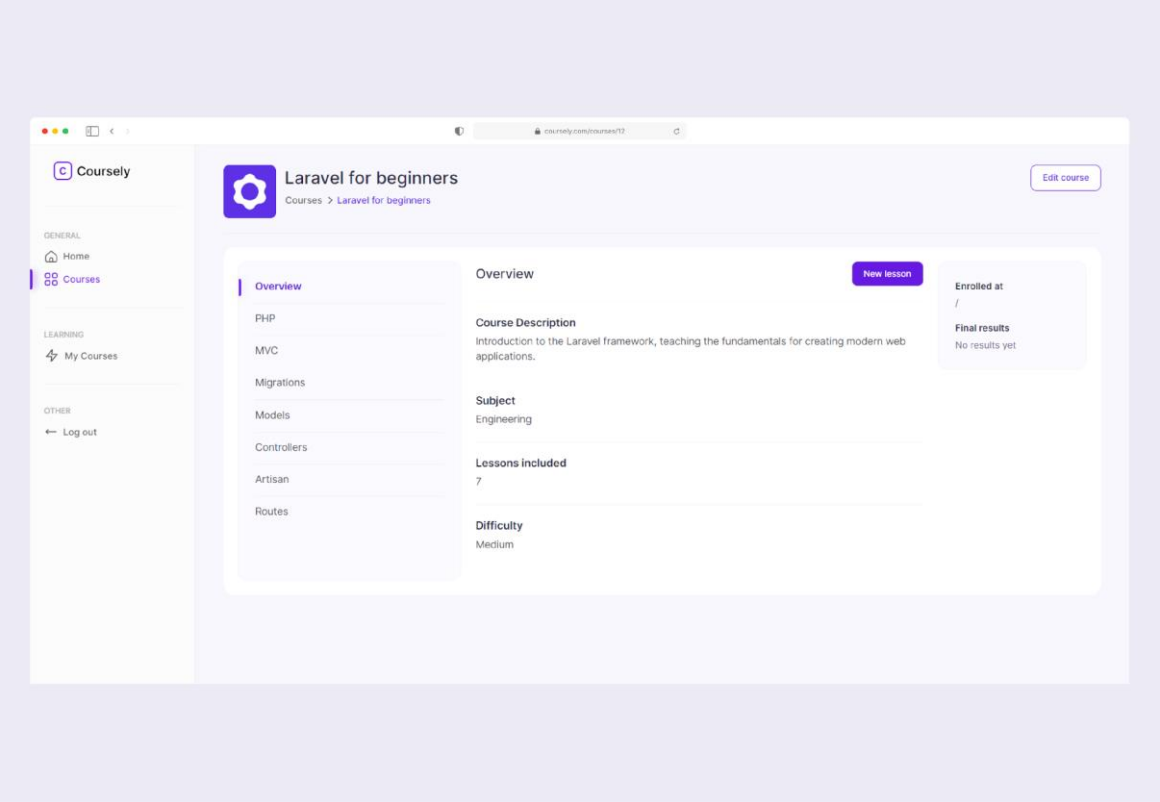


Slika 5.4. Courses stranica studenta, izvor: *izrada autora*



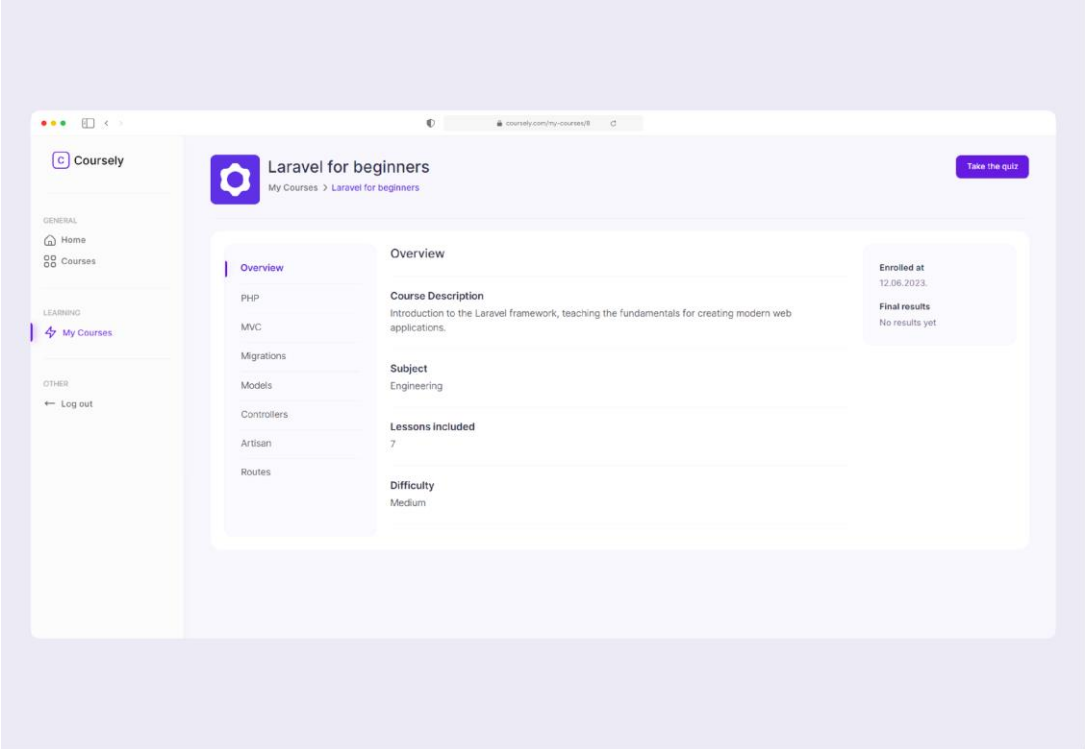
Slika 5.5. My courses stranica studenta, izvor: *izrada autora*

Uzimajući u obzir različite uloge korisnika u sustavu, zaslon pojedinog tečaja sadrži različitosti u ovisnosti o ulozi prijavljenog korisnika. Slika 5.6. prikazuje zaslon tečaja ukoliko je prijavljeni korisnik administrator, a u slučaju da je prijavljeni korisnik student, njemu prikazana mogućnost bila bi upisivanje u tečaj, bez mogućnosti uređivanja tečaja i dodavanja lekcija. Također, ako je prijavljeni korisnik predavač, te se odluči za uređivanje tečaja, neke od mogućnosti neće mu biti dostupne. Primjerice, jedna od nedostupnih mogućnosti je promjena teme tečaja, kao i promjena profesora dodijeljenog tečaju. Samo administrator ima ovlasti provesti spomenute akcije.



Slika 5.6. Zaslon tečaja administratora, izvor: *izrada autora*

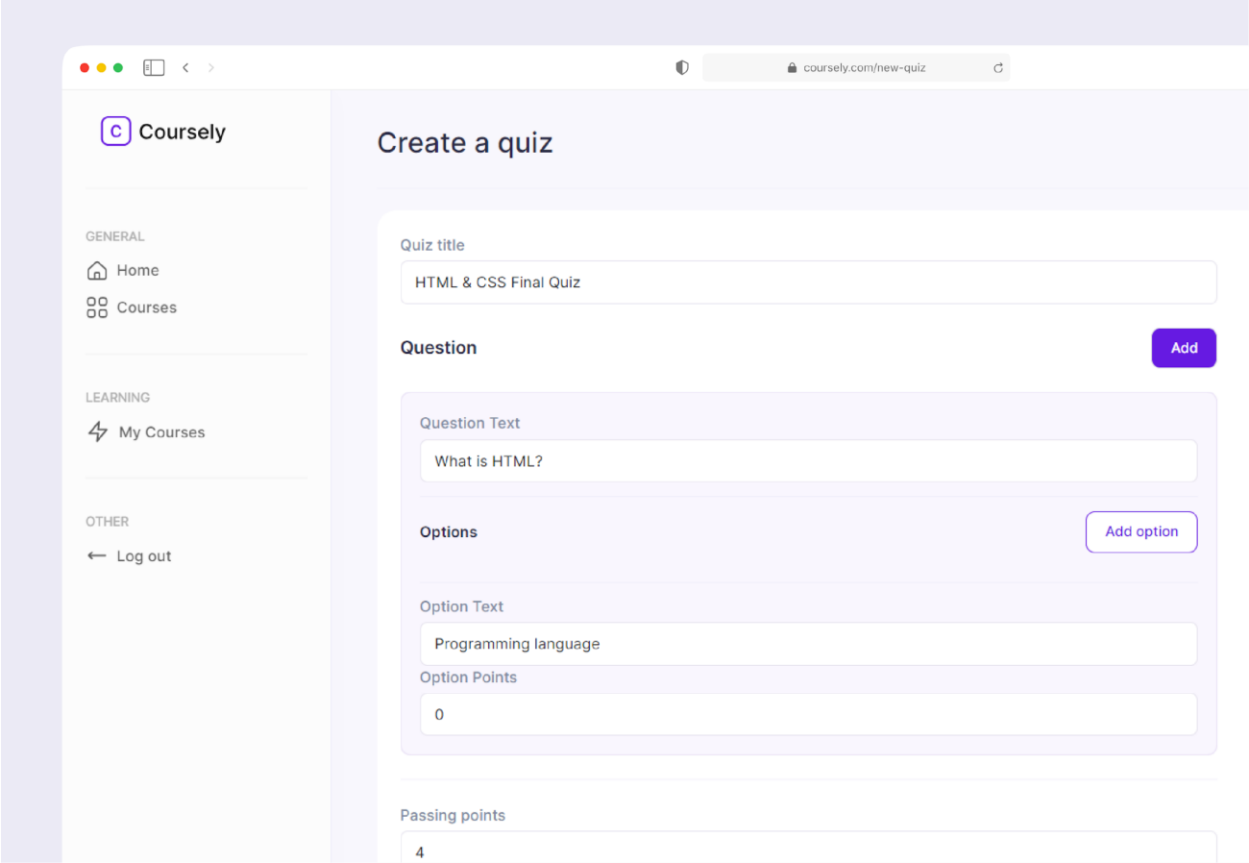
Ukoliko je student već upisan u tečaj, on će vidjeti zaslon prikazan na slici 5.7..



Slika 5.7. Zaslon tečaja upisanog studenta, izvor: *izrada autora*

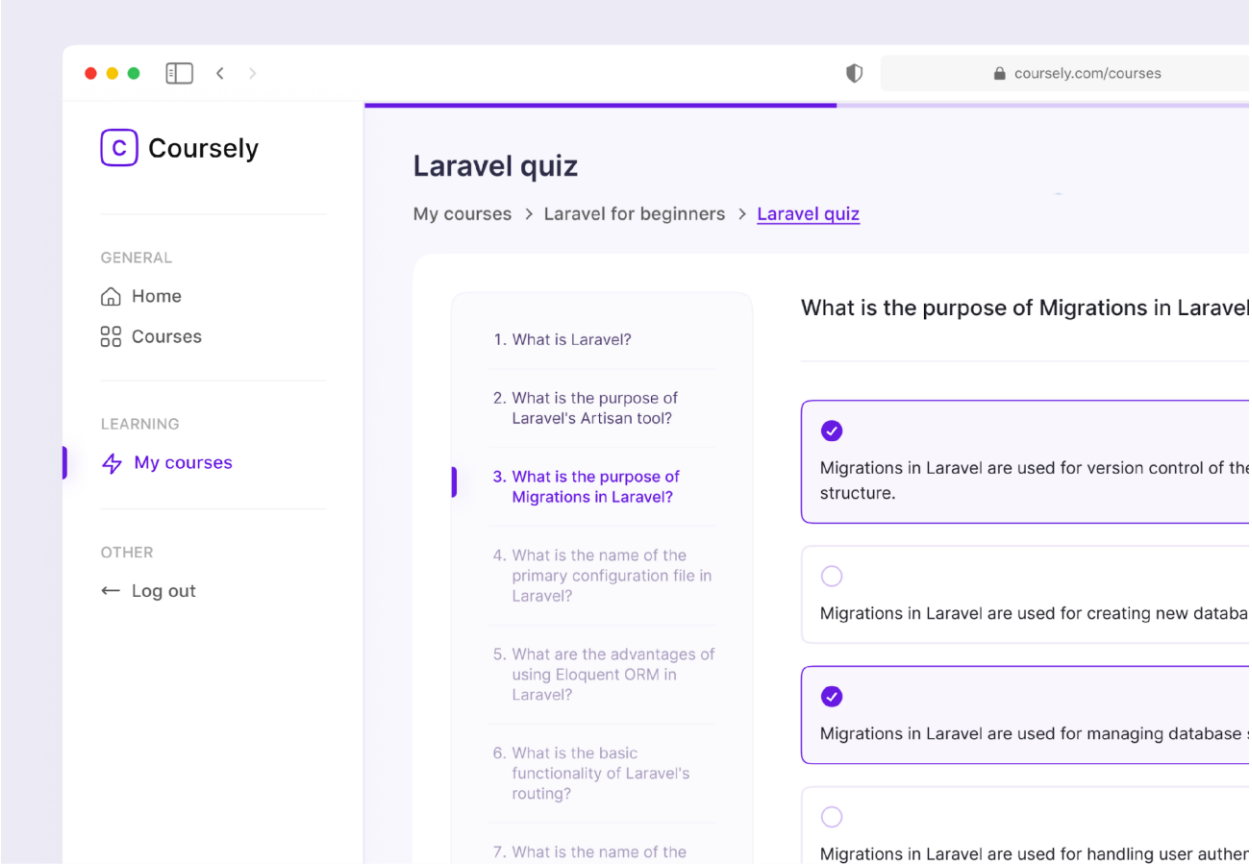
5.4. Završni ispiti

Dodavanje završnog ispita moguće je obaviti posjetom stranice određenog tečaja te klikom na gumb Create quiz, koji će se prikazati ukoliko je prijavljeni korisnik onaj korisnik koji je dodan kao predavač na tečaj, te ukoliko odabrani tečaj nije prethodno povezan s nekim od kvizova. Slika 5.8. prikazuje zaslon kreiranja kviza, na kojem je moguće dodati pitanja te opcije za svako od ponuđenih pitanja. Na spomenutoj slici, kreirani su jedno pitanje te jedna opcija za to pitanje. Klikom na gumb *Add* u ravnini s pitanjima, moguće je dodati proizvoljan broj novih pitanja, a na sličan način, klikom na gumb *Add option*, korisnik može dodati više opcija na svako od pitanja.



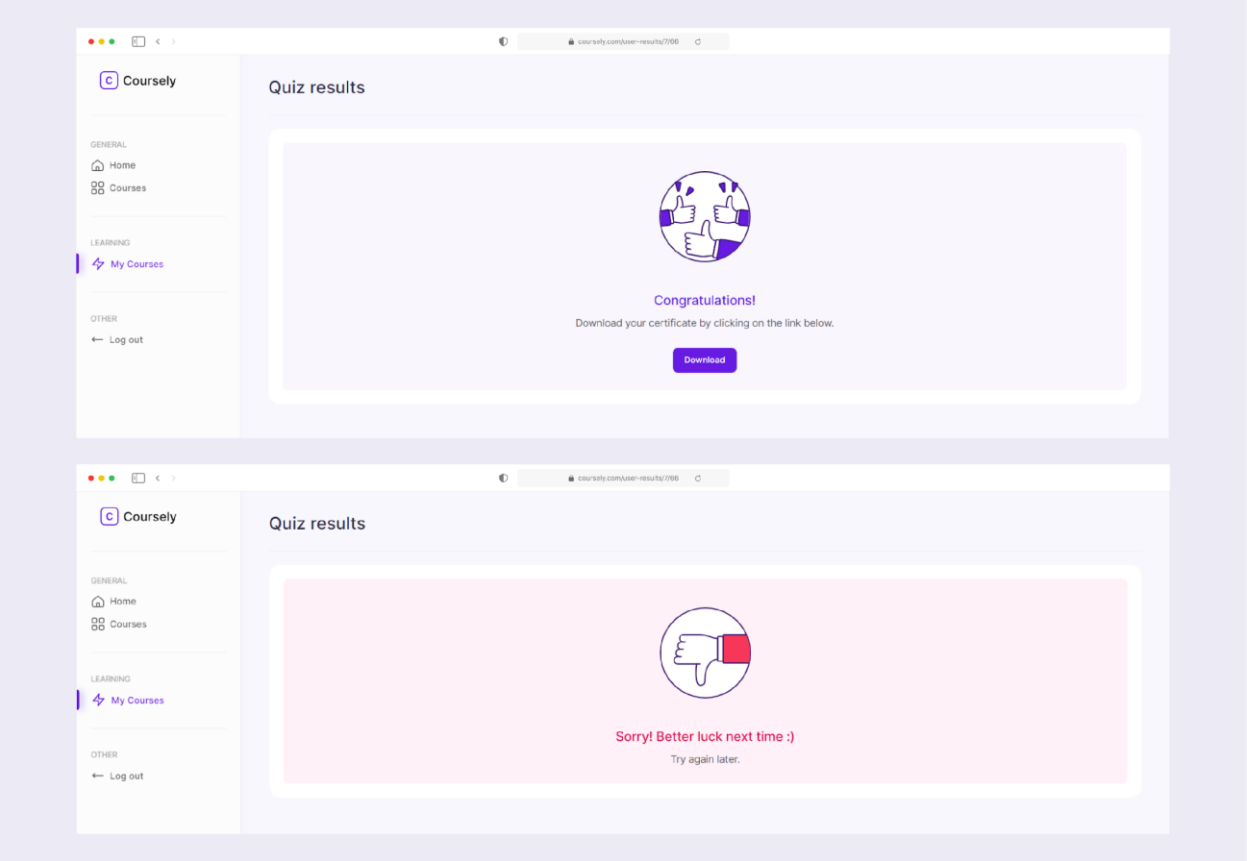
Slika 5.8. Kreiranje završnog ispita, izvor: *izrada autora*

Nakon upisa korisnika u tečaj, ukoliko je kviz dostupan za taj tečaj, postoji mogućnost rješavanja kviza pritiskom na gumb *Take the quiz*. Slika 5.9. prikazuje zaslon završnog ispita u tijeku. Pitanja su pozicionirana s lijeve strane glavnog dijela zaslona, dok su opcije za odabir ponuđene u glavnom dijelu zaslona. Vremensko ograničenje korisniku je vidljivo cijelo vrijeme linijom prikazanom na vrhu zaslona, koja sukladno isteknutom vremenu mijenja svoju duljinu.



Slika 5.9. Rješavanje završnog ispita, izvor: *izrada autora*

Završetkom kviza, korisnika se preusmjerava na stranicu s rezultatima. Prikaz dvije stranice s rezultatima može se vidjeti na slici 5.10., a razlika između spomenutih zaslona korisnikov je krajnji rezultat. Na drugom zaslonu crvenom je bojom označeno da korisnik nije zadovoljio uvjete potrebne kako bi uspješno položio ispit.



Slika 5.10. Zaslon rezultata, izvor: *izrada autora*

6. ZAKLJUČAK

Kao ključni dio ovog rada, istražena je važnost edukacije putem interneta u današnjem svijetu. Proučeni su razlozi iz kojih pojedinci, odnosno korisnici spomenutih sustava, donose odluku usavršiti ili unaprijediti svoje obrazovanje koristeći platforme namijenjene učenju putem interneta, te na koje načine spomenuti sustavi doprinose životu pojedinca. Iz postojećih rješenja, analizirane su ključne stavke web aplikacija kako bi se uspješno definirali zahtjevi s ciljem uspostave sustava s kvalitetnim korisničkim iskustvom za određenu skupinu ljudi. Prema spomenutim zahtjevima, kreirana je platforma za edukacije putem interneta koristeći tehnologije okvira Laravel i VueJS. Sve tehnologije upotrebljene prilikom izrade rada, detaljno su opisane skupa s razlozima iz kojih su iste odabrane i korištene. U radu su također detaljno opisane sve funkcionalnosti aplikacije uz popratne slike koda napisanog kako bi određena funkcionalnost bila uspješno provedena. Uz slike koda, rad sadrži slike tablica iz baze podataka, te prikaz korisničkih zaslona pojedinih uloga korisnika u sustavu.

Uzimajući u obzir daljnje unapređenje ove aplikacije, postoji nekoliko opcija koje bi osigurale značajan razvoj platforme ukoliko bi bile implementirane. Mogućnost komunikacije profesora sa studentima u obliku slanja poruka, koje bi osim tekstualnih poruka mogle sadržavati i datoteke poput fotografija ili dokumenata, zasigurno bi pridonijela kvaliteti obrazovanja te omogućila korisnicima s ulogom studenta bolje shvaćanje tematike u slučaju postojanja nejasnoća. Također, sjajna nadogradnja ovog sustava bila bi omogućavanje prijenosa lekcija uživo (engl. *live*), kako bi predavači imali mogućnost educirati korisnike upisanih tečajeva u stvarnom vremenu, te na taj način osigurali izvrsnost sadržaja lekcija tečaja.

LITERATURA

[1] Udemy, [Online] Dostupno na: https://www.udemy.com/ [Pristup: 06.06.2023.]

[2] Coursera, [Online] Dostupno na: https://www.coursera.org/ [Pristup: 06.06.2023.]

[3] Skillshare, [Online] Dostupno na: https://www.skillshare.com/ [Pristup: 06.06.2023.]

[4] Pluralsight Skills, [Online] Dostupno na: https://www.pluralsight.com/product/skills [Pristup: 06.06.2023.]

[5] Pluralsight, [Online] Dostupno na: https://www.pluralsight.com/ [Pristup: 06.06.2023.]

[6] LinkedIn Learning, [Online] Dostupno na: https://www.linkedin.com/learning/ [Pristup: 06.06.2023.]

[7] C. Kolade, „What is HTML – Definition and Meaning of Hypertext Markup Language“, FreeCodeCamp, [Online] Dostupno na: https://www.freecodecamp.org/news/what-is-html-definition-and-meaning/ [Pristup: 07.06.2023.]

[8] „HTML <html> Tag“, W3Schools, [Online] Dostupno na: https://www.w3schools.com/tags/tag_html.asp [Pristup: 07.06.2023.]

[9] „Head Tag (in HTML)“, RankRanger, [Online] Dostupno na: https://www.rankranger.com/seo-glossary/head-tag [Pristup: 07.06.2023.]

[10] „HTML <body> Tag“, W3Schools, [Online] Dostupno na: https://www.w3schools.com/tags/tag_body.asp [Pristup: 07.06.2023.]

[11] „What is CSS?“, MDN Web Docs, [Online] Dostupno na: https://developer.mozilla.org/en-US/docs/Learn/CSS/First_steps/What_is_CSS [Pristup: 07.06.2023.]

[12] „How CSS works“, MDN Web Docs, [Online] Dostupno na: https://developer.mozilla.org/en-US/docs/Learn/CSS/First_steps/How_CSS_works [Pristup: 07.06.2023.]

[13] „CSS Syntax“, W3Schools, [Online] Dostupno na: https://www.w3schools.com/css/css_syntax.asp [Pristup: 07.06.2023.]

[14] T. Matijević, „Introduction to BEM Methodology“, Toptal, [Online] Dostupno na: https://www.toptal.com/css/introduction-to-bem-methodology [Pristup: 07.06.2023.]

[15] A. Jordana, „What Is JavaScript? A Basic Introduction to JS for Beginners“, Hostinger, [Online] Dostupno na: https://www.hostinger.com/tutorials/what-is-javascript [Pristup: 07.06.2023.]

[16] „Getting started with Vue“, MDN Web Docs, [Online] Dostupno na: https://developer.mozilla.org/en-US/docs/Learn/Tools_and_testing/Client-side_JavaScript_frameworks/Vue_getting_started [Pristup: 08.06.2023.]

[17] „Options: State“, VueJS, [Online] Dostupno na: https://vuejs.org/api/options-state.html [Pristup: 08.06.2023.]

[18] C. Kolade, „What is PHP? The PHP Programming Language Meaning Explained“, FreeCodeCamp, [Online] Dostupno na: https://www.freecodecamp.org/news/what-is-php-the-php-programming-language-meaning-explained/ [Pristup: 08.06.2023.]

[19] A. Jalli, „What is Laravel“, Builtin, [Online] Dostupno na: https://builtin.com/software-engineering-perspectives/laravel [Pristup: 08.06.2023.]

[20] „MVC Framework – Introduction“, TutorialsPoint, [Online] Dostupno na: https://www.tutorialspoint.com/mvc_framework/mvc_framework_introduction.htm [Pristup: 08.06.2023.]

[21] „MVC“, MDN Web Docs, [Online] Dostupno na: https://developer.mozilla.org/en-US/docs/Glossary/MVC [Pristup: 08.06.2023.]

[22] „Artisan Console“, Laravel, [Online] Dostupno na: https://laravel.com/docs/9.x/artisan [Pristup: 08.06.2023.]

[23] „1.2.1 What is MySQL?“, MySQL, [Online] Dostupno na: https://dev.mysql.com/doc/refman/8.0/en/what-is-mysql.html [Pristup: 08.06.2023.]

[24] T. Pattinson, „Relational vs. Non-Relational Databases“, Pluralsight, [Online] Dostupno na: https://www.pluralsight.com/blog/software-development/relational-vs-non-relational-databases [Pristup: 08.06.2023.]

[25] geetanjali16, „MySQL | Common MySQL Queries“, GeeksForGeeks, [Online] Dostupno na: https://www.geeksforgeeks.org/mysql-common-mysql-queries/ [Pristup: 08.06.2023.]

[26] „phpMyAdmin“, Javatpoint, [Online] Dostupno na: https://www.javatpoint.com/phpmyadmin [Pristup: 08.06.2023.]

[27] Y. Emma, „What Is WAMP – a Friendly Guide for Beginners“, Hostinger, [Online] Dostupno na: https://www.hostinger.com/tutorials/what-is-wamp [Pristup: 08.06.2023.]

[28] „WampServer“, WampServer, [Online] Dostupno na: https://www.wampserver.com/en/ [Pristup: 08.06.2023.]

[29] W. Mwaura, „Making HTTP requests with Axios“, Circleci, [Online] Dostupno na: https://circleci.com/blog/making-http-requests-with-axios/ [Pristup: 12.06.2023.]

[30] „Validation“, Laravel, [Online] Dostupno na: https://laravel.com/docs/5.0/validation [Pristup: 12.06.2023.]

[31] „Facades“, Laravel, [Online] Dostupno na: https://laravel.com/docs/10.x/facades [Pristup: 12.06.2023.]

[32] D. Berning, „How To Manage State in a Vue.js Application with Vuex“, DigitalOcean, [Online] Dostupno na: https://www.digitalocean.com/community/tutorials/how-to-manage-state-in-a-vue-js-application-with-vuex [Pristup: 16.06.2023.]

[33] Lucidchart, [Online] Dostupno na: https://www.lucidchart.com/pages/er-diagrams [Pristup: 23.06.2023.]

SAŽETAK

U suvremenom, digitalno razvijenom svijetu, kao ključno sredstvo usvajanja znanja predstavlja se obrazovanje putem interneta. Edukacija prilagođena individualnim potrebama korisnika, uz istraživanje novih područja znanja, omogućuje temeljito razvijanje interesa na jednostavan način. Kako bi se korisnicima omogućio pristup raznovrsnim područjima interesa na jednom mjestu, kao rezultat ovog rada razvijena je web aplikacija koja korisnicima pruža intuitivno korisničko sučelje s brojnim funkcionalnostima, istovremeno osiguravajući visoku kvalitetu sadržaja tečajeva. Korisnici s različitim ulogama unutar sustava različitim odgovornostima doprinose osiguranju kakvoće sadržaja te jednostavnog stjecanja znanja. Aplikacija je izrađena u Laravel i Vue.js razvojnim okruženjima, a od ostalih tehnologija koristi HTML, CSS, JavaScript, PHP, MySQL, te WAMP. Ovaj portal omogućava prilagodbu obrazovnog iskustva osobnim potrebama korisnika, te na taj način unapređuje iskustvo u stjecanju znanja korisnika.

Ključne riječi: edukacija, Laravel, obrazovanje, profesor, student, učenje, Vue, web aplikacija

ABSTRACT

Web portal for education

Education through the internet has emerged as a key method for acquiring knowledge in the modern, digitally advanced world. Individualized education, tailored to each of the students' needs, fosters a deep understanding of new areas of knowledge through an intuitive approach facilitating comprehensive knowledge exploration. As a result, this web application was developed in order to allow the users to access diverse areas of interest in one place, ensuring high-quality course content and intuitive user interface. Through their respective responsibilities, users with different roles within the system contribute to the seamless acquisition of knowledge and the quality of content. The application was built using Laravel and Vue.js frameworks, while also using other technologies: HTML, CSS, JavaScript, PHP, MySQL and WAMP. Providing the ability to customize their educational experiences, this platform enhances the users' educational journeys.

Key words: education, Laravel, learning, knowledge, professor, student, Vue, web application

ŽIVOTOPIS

Ana Čačija, rođena 7.4.1998. godine u Osijeku, Hrvatska, 2004. godine upisuje se u Osnovnu školu „Tin Ujević“ u Osijeku. Godine 2012. završava svoje osnovnoškolsko obrazovanje, te nakon toga upisuje Prirodoslovno-matematičku gimnaziju u Osijeku. 2016. godine, nakon mature, upisuje se na Fakultet Elektrotehnike, Računarstva i Informacijskih Tehnologija u Osijeku, gdje 2019. godine završava preddiplomski sveučilišni studij Računarstva, te upisuje diplomski studij Računarstvo, smjer Programsko inženjerstvo.

PRILOZI

Prilog 1: Dokumentacija diplomskog rada na CD-u

Prilog 2: Kod aplikacije dostupan na Github-u: https://github.com/anacc12/coursely