



UNIVERZITET U BIHAĆU
TEHNIČKI FAKULTET
ODSJEK: ELEKTROTEHNIKA
SMJER: RAČUNARSTVO I INFORMATIKA

Objektno orijentirane baze podataka

PROJEKTNI ZADATAK
TEMA: MOTO ŠKOLA

Profesor: Prof.dr Ramo Šendelj
Asistent: MA Zinaid Kapić

Student:
Elmin Midžić, 1173

Bihać, decembar 2023. godine

Sažetak

Većina postojećih okvira za razvoj web aplikacija, uključujući Laravel, koji je korišten za kreiranje ove stranice, su objektno orijentirani i izgrađeni korištenjem MVC dizajna. Laravel je PHP okvir za izgradnju objektno orijentiranih web aplikacija. Kroz ovu aplikaciju uvode se osnovne radne opcije sa Laravel-om, uključujući CRUD operacije na bazi podataka. Ova jednostavna aplikacija ima za cilj da pokaže kako se objektno-relacioni model koristi za rad sa relacionom bazom podataka čiji su podaci pohranjeni kao objekti. Predmet aplikacije je moto škola, a entiteti su konstruisani u skladu s tim pojmom. Odabir entiteta osigurava da se relevantne fraze za pretraživanje koriste za sadržaj moto škole. Program generiše objekte za svaku klasu, izvršava relevantne funkcije nad njima i prikazuje rad sa objektima kao prikaze podataka. Koristi se SQL baza podataka, a sama Laravel aplikacija se koristi za mapiranje entiteta baze podataka u objekte. Nekoliko pretraga koje izdvajaju podatke iz nekoliko objekata i filtriraju rezultate ovisno o odgovarajućim kriterijima daju različite opcije za rad na ovim stvarima.

Ključne riječi: Laravel, CRUD, PHP, Objektno-orijentirano, Baza podataka

Abstract

Most of the existing frameworks for web application development, including Laravel, which was utilized in creating this website, are grounded in an object-oriented approach and developed using the MVC design. Laravel, a PHP framework, is designed for constructing object-oriented web applications. Within this application, fundamental operations with Laravel are introduced, encompassing CRUD operations on the database. The purpose of this straightforward application is to showcase how the object-relational model is employed to interact with a relational database where data is stored as objects. The focal theme of the application is a motorcycle school, and entities are crafted in alignment with this concept. The selection of entities ensures the use of pertinent search phrases for the content of the motorcycle school. The program generates objects for each class, executes pertinent functions on them, and presents the handling of objects as data views. An SQL database is employed, and the Laravel application itself is utilized to map database entity objects. Several searches extract data from diverse objects and filter results based on relevant criteria, providing various options for working with these elements.

Keywords: Laravel, CRUD, PHP, Object-Oriented, Database

Sadržaj

1	Uvod	1
2	Modeliranje aplikacije	2
2.1	Opis aplikacije	2
2.2	Statički UML dijagrami	2
2.2.1	Klasni dijagram	3
2.3	Dinamički UML dijagrami	3
2.3.1	Dijagram slučajeva korištenja	4
2.3.2	Sekvencijalni dijagram	5
2.4	ER dijagram baze podataka	7
3	Implementacija	8
3.1	Tehnologija izrade aplikacije	8
3.1.1	Laravel	8
3.1.2	MySQL	8
3.2	MVC arhitektura	8
3.3	Objektno-relaciono mapiranje	9
3.4	REST Api	10
3.4.1	Implementacija REST API	10
4	Analiza rada aplikacije	17
4.1	Opis slučajeva korištenja	17
4.2	Testiranje rada baze podataka	29
4.3	Testiranje API-a	31
5	Zaključak	37

Popis slika

1	Klasni dijagram	3
2	Use-Case dijagram	4
3	Sekvencijalni dijagram	6
4	Entity Relationship dijagram	7
5	MVC arhitektura i komunikacija među komponentama	9
6	Kreirane akcije u Insomnia softveru	12
7	Prikaz instruktora (GET)	13
8	Dodavanje instruktora (POST)	14
9	Uređivanje instruktora (PUT)	15
10	Brisanje instruktora (DELETE)	16
11	Stranica za prijavu	17
12	Netačna prijava	18
13	Početna stranica	19
14	Stranica za pregled kandidata	20
15	Stranica za uređivanje podataka o kandidatu	21
16	Stranica za dodavanje kandidata	22
17	Brisanje kandidata	23
18	Stranica za pregled motora	24
19	Stranica za pregled instruktora	25
20	Stranica za dodjelu instruktora	26
21	Stranica za dodjelu instruktora kada je odbijen zahtjev	27
22	Stranica za prikaz statistike	28
23	Prikaz izlaza prethodno prikazanih implementiranih upita	30
24	Prikaz dodavanja instruktora sa ID 10	31
25	Prikaz dodavanja instruktora sa ID 10	32
26	Prikaz izmjene instruktora sa ID 10	33
27	Prikaz liste instruktora nakon izmjene	34
28	Prikaz brisanja instruktora sa ID 4	35
29	Prikaz svih instruktora nakon brisanja	36

1 Uvod

Za realizaciju ovog projekta koristit će se PHP programski jezik. PHP se smatra jednim od najpopularnijih programskih jezika za izradu web stranica. Suprotno većini popularnih jezika za razvoj web aplikacija koji grade prikaze podataka na strani klijenta, PHP se u potpunosti obrađuje na strani servera. Kroz godine, razni razvojni okviri su se razvijali kako bi olakšali izradu sve složenijih aplikacija, ubrzali njihovu implementaciju i organizirali kod. Laravel je jedan od najdražih PHP razvojnih okvira, a u implementaciji ovog projekta je angažiran Laravel.

Laravel se izdvaja ne samo po svojoj snažnoj arhitekturi, već i po mnogim mogućnostima koje olakšavaju razvoj web aplikacija. Bez obzira na tehnologiju koja se koristi za obradu podataka, baza podataka predstavlja organiziran skup podataka koji koriste jedna ili više aplikacija. Objektno-relacijsko mapiranje (ORM) koristi se kao tehnika programiranja za transformaciju podataka između objektno orijentiranih programskih jezika i relacijskih baza podataka (SQL tipa), čime se pojednostavljuje rad s podacima.

Modifikacija informacija kako bi se prilagodile bazi podataka može predstavljati izazov prilikom razvoja aplikacije povezane s bazom podataka. ORM omogućava apstrakciju od same baze podataka, značajno smanjujući količinu potrebnog izvornog koda. Prvi dio projektnog procesa obuhvaća modeliranje aplikacije i ključne informacije potrebne za uspješan razvoj. Zatim slijedi opis same aplikacije, uključujući njezino funkcioniranje i popis svih entiteta u sistemu.

Nadalje, pruženi su opisi statičkih i dinamičkih dijagrama koji vizualiziraju međusobne odnose između elemenata. Tehnologija korištena za izradu aplikacije detaljno je opisana u sljedećem poglavlju. Baza podataka je izgrađena pomoću PHP-a, Laravela i MySQL-a, a osnovne ideje iza Laravela su također objašnjene. Posljednje poglavlje sadrži studiju slučaja aplikacije, predstavlja aktere sistema te opisuje nekoliko scenarija upotrebe. Nakon toga, izvršeno je testiranje funkcionalnosti baze podataka kroz izvođenje unaprijed definiranih jednostavnih i zahtjevnih upita.

2 Modeliranje aplikacije

Da bi tema bila dobro shvaćena i realizovana, mora se pažljivo i precizno odrediti prilikom modeliranja aplikacije. Određeni dijagrami koji predstavljaju aplikaciju koriste se prije njene stvarne implementacije u određenom programskom jeziku kako bi se taj razvoj pojednostavio. Dijagrami, na primjer, pojednostavljaju razumijevanje aplikacije i olakšavaju implementaciju. Prilikom izrade tabele, nazivi tabele, odnosno entiteti, već su određeni u dijagramima. Zatim se određuju svojstva entiteta i njegove veze sa drugim entitetima u sistemu. Pet dijagrama je napravljeno prije stvarne implementacije, uključujući i statičke i dinamičke UML dijagrame.

2.1 Opis aplikacije

2.2 Statički UML dijagrami

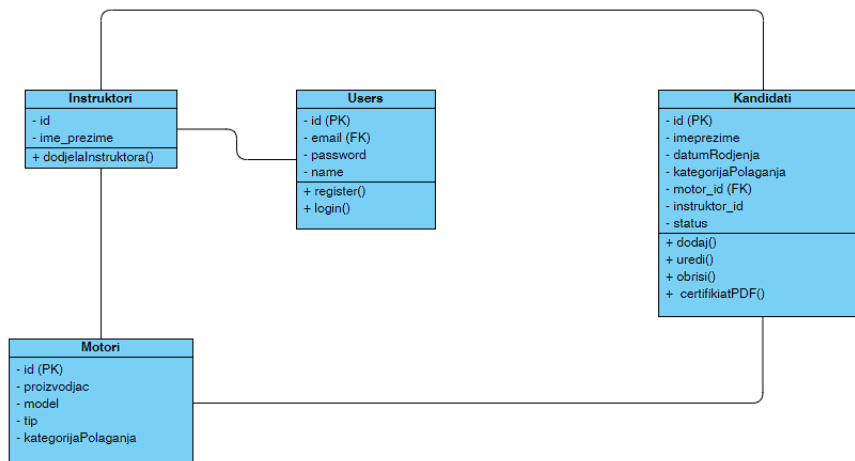
Statički UML (engl. *Unified Modelling Language*) dijagrami služe za opisivanje karakteristika ili dijela sistema [5]. Postoji 5 vrsta statičkih dijagrama:

- Dijagram strukture aplikacije
- Klasni dijagram
- Objektni dijagram
- Dijagram međusobnih odnosa objekata (kompozicioni dijagram)
- Dijagram pravila (ograničenja svakog dijela sistema)

Od navedenih statičkih dijagrama u nastavku je detaljnije opisan i prikazan klasni dijagram.

2.2.1 Klasni dijagram

Klasni dijagram opisuje klase koje su dio softverskog sistema. Termin klasa se koristi u objektno orijentiranom programiranju za definisanje strukture podataka koja se sastoji od svojstava (atributa) i funkcija (metoda). Klasni dijagram rađenog sistema je prikazan na slici 1.



Slika 1: Klasni dijagram

2.3 Dinamički UML dijagrami

Za modeliranje aplikacija, pored statičkih dijagrama, koriste se i dinamički dijagrami. Dinamički UML dijagrami opisuju ponašanje sistema, odnosno oni opisuju operacije, radnje i promjene koje se događaju u sistemu tokom rada. Postoji 5 vrsta dinamičkih dijagrama:

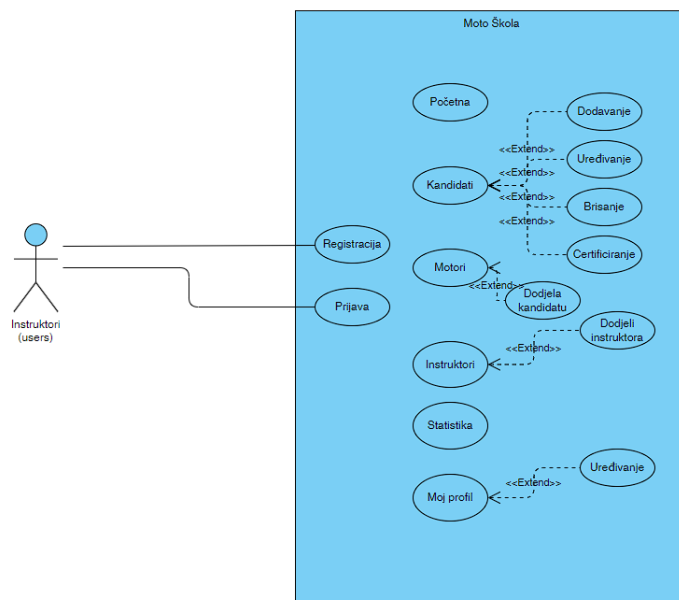
- Dijagram aktivnosti
- Dijagram stanja
- Dijagram slučajeva korištenja
- Sekvencijalni dijagram
- Komunikacioni dijagram

Od navedenih dinamičkih dijagrama u nastavku su detaljnije opisani i prikazani dijagram slučajeva korištenja, te sekvencijalni dijagram.

2.3.1 Dijagram slučajeva korištenja

Dijagram slučajeva korištenja (engl. *Use Case Diagram*) je osnovni dinamički dijagram koji opisuje sve mogućnosti korištenja aplikacije. Dijagrami slučajeva korištenja opisuju kako korisnik može koristiti sistem, kako komponente sistema mogu koristiti druge komponente sistema, kako i ko upravlja ponašanjem sistema. U slučaju ovog sistema postoji samo jedna vrsta korisnika, odnosno svi korisnici imaju jednaka prava na korištenje svih funkcija sistema. Funkcije registracije i prijave dođu unaprijed postavljene zajedno sa postavkom Laravel projekta. Pored ovih funkcija funkcija pristupa podacima korisničkog profila i njihovom uređivanju također dolazi postavljena u sklopu Laravel projekta. Entitet kandidati posjeduje mogućnosti brisanja, uređivanja, dodavanja. Entitet instruktori posjeduje mogućnost dodjele kandidatu. Glavna funkcija aplikacije jeste prikaz posebno dizajniranih upita, kojoj se pristupa preko linka sa početne stranice nakon prijave korisnika.

Opisani dijagram slučajeva korištenja (Use-case dijagram) rađenog sistema je prikazan na slici 2.



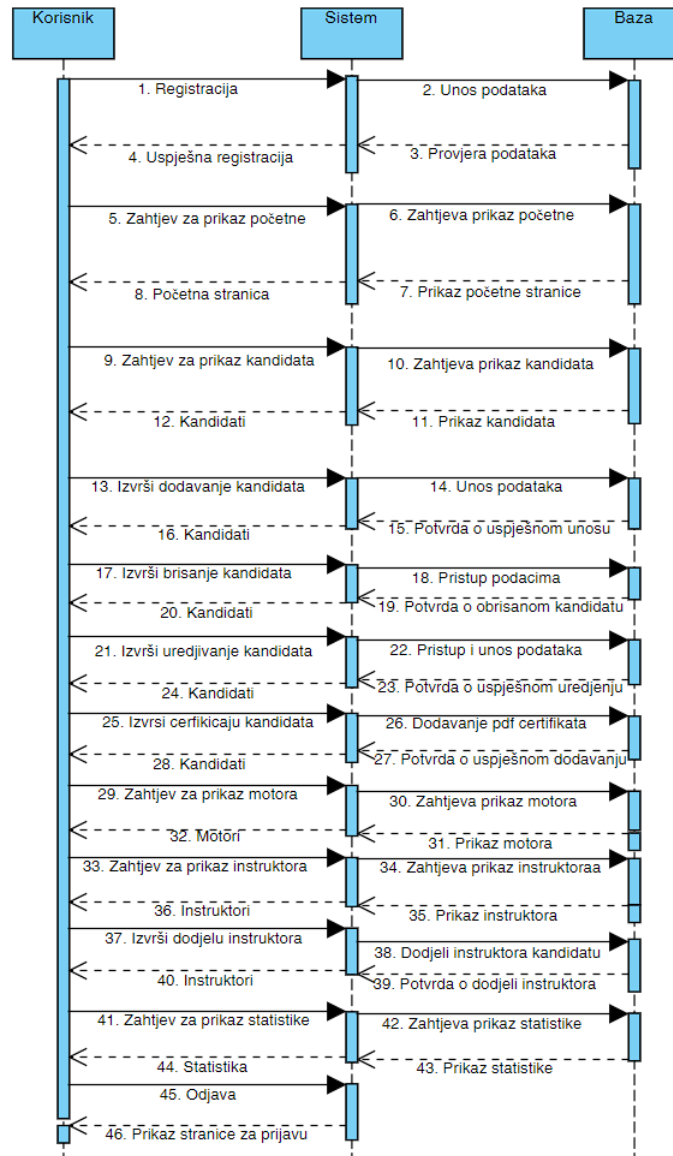
Slika 2: Use-Case dijagram

Kao što se može vidjeti iz priloženog Use-Case dijagrama, postoji samo jedna vrsta korisnika, sa velikim brojem opcija. Ukoliko korisnik posjeduje korisnički račun prijavljuje se na sistem, a ukoliko ne posjeduje, registruje se na sistem,

a zatim prijavljuje. Prijavom na sistem dobijaju se sve dostupne opcije na navigacionom meniju. Početne opcije su pregled svih unosa u bazi za odabrani entitet, a otvaranjem nekih opcija, pojavljuju se dodatne opcije za dodavanje, uređivanje i brisanje.

2.3.2 Sekvencijalni dijagram

Sekvencijalni dijagram spada u podgrupu dinamičkih dijagrama, a to su interakcijski dijagrami. Sekvencijalni dijagram uvodi vremensku komponentu za opisivanje ponašanja sistema. Koriste se za prikaz rada sistema u realnom vremenu. Sekvencijalni dijagram pokazuje i način na koji objekti surađuju u nekom određenom scenariju, a ta interakcija između objekata prikazuje se putem poruka. Kod ovog dijagrama se definišu: objekti, veze i poruke. Sekvencijalni dijagram daje naglasak na vrijeme, odnosno daje naglasak na redoslijed gdje se odvija međudjelovanje učesnika u nekom sistemu. Vrijeme na ovom dijagramu uvijek ide od vrha prema dnu. Prema tome korisnici i sistem predstavljaju vremenske pravce. Pored toga, postoji i horizontalna dimenzija koja određuje objekte. Sekvencijalni dijagrami sadrže i poruke koje razmjenjuju sami učesnici unutar sistema. One se razmjenjuju u određenom vremenu između učesnika u sistemu. Sekvencijalni dijagram predstavljen je na slici 3.

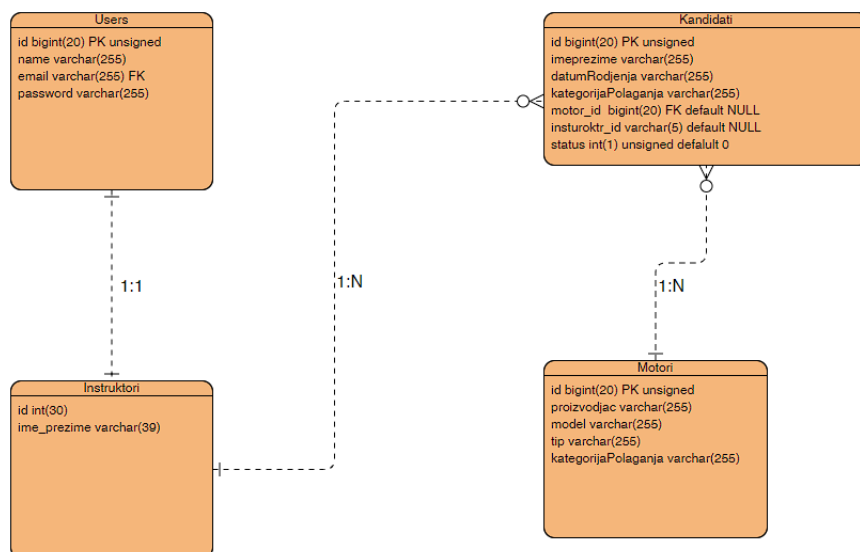


Slika 3: Sekvencijalni dijagram

Na slici 3. prikazan je sekvencijalni dijagram na kojem se nalazi vremenski prikaz radnji, tj. ponašanje sistema.

2.4 ER dijagram baze podataka

Baza podataka je opisana ER (engl. *Entity Relationship*) dijagramom. Ovaj dijagram sadrži entitete odnosno pojmove od interesa i njihove atribute, kao i veze među njima. Baza podataka korištena za realizaciju ovog sistema se sastoji od 4 tabele. Korištene tabele su: Users, Kandidati, Motori, Instruktori. Opisani ER dijagram radenog sistema je prikazan na slici 4.



Slika 4: Entity Relationship dijagram

Na slici 4. prikazan je ER dijagram na kojem je vidljivo da su između entiteta ostvarene veze 1 : N i 1:1.

3 Implementacija

3.1 Tehnologija izrade aplikacije

Aplikacija je implementirana u programskom jeziku PHP i koristeći Laravel framework. Laravel je baziran na MVC modelu (engl. *Model View Controller*) gdje je prezentacijska, funkcijska i podatkovna strana međusobno odvojena. DBMS (engl. *Database Management System*) sistem korišten za kreiranje baze podataka je MySQL.

3.1.1 Laravel

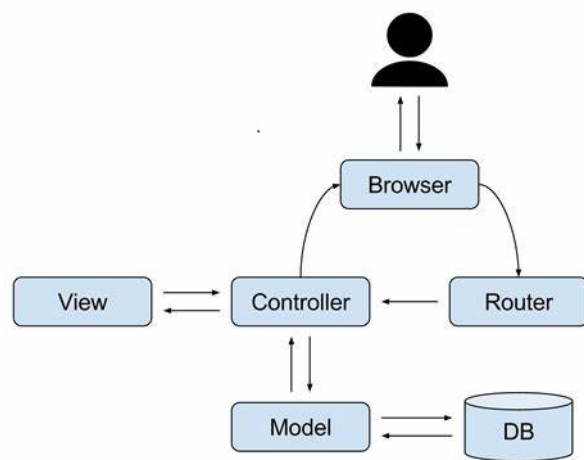
Laravel je open-source PHP okvir, koji je robustan i lako razumljiv. Laravel sa sobom donosi izrazito moćne web-predloške, implementaciju objektno-relacijskog preslikavanja (ORM) za komunikaciju sa bazom podataka, kao i brojne druge mogućnosti koje se zahtijevaju od moderne platforme za razvoj web-aplikacija. Laravel nudi bogat skup funkcionalnosti koji uključuje osnovne osobine PHP okvira poput CodeIgniter, Yii i drugih programskih jezika kao što su Ruby on Rails. Laravel ima vrlo bogat skup osobina koje znatno pomažu u poboljšanju brzine web razvoja.

3.1.2 MySQL

MySQL je relacionalna baza podataka. To znači da su podaci smješteni unutar strukture sposobni da prepoznaju odnose između sačuvanih informacija. Svaka baza podataka sadrži tabele, a svaka tabela (koja se također naziva i relacija) sadrži jednu ili više kategorija podataka pohranjenih u kolonama (koje se također nazivaju atributima). Svaki red sadrži jedinstven podatak (koji se inače naziva ključ) za kategorije definisane u kolonama. Predstavlja najpopularniji open source sistem za upravljanje bazama podataka i trenutno je u svijetu rangiran kao drugi najpopularniji sistem za upravljanje bazama podataka, poslije Oracle Database.

3.2 MVC arhitektura

Model-View-Controller je šablon softverske arhitekture. Koristi se u softverskom inženjeringu za odvajanje pojedinih dijelova aplikacije u komponente ovisno o njihovoj namjeni. Model se sastoji od podataka, poslovnih pravila, logike, i funkcija ugrađenih u programsku logiku. View odnosno pogled je bilo kakav prikaz podataka kao što je obrazac, tablica ili dijagram. Moguć je prikaz podataka kroz više različitih pogleda. Controller ili upravitelj prihvata ulazne podatke i zahtjeve (od korisnika prema sistemu) i pretvara ih u zahtjeve modelu ili pogledu. Ovakva arhitektura olakšava nezavisan razvoj, testiranje i održavanje određene aplikacije.



Slika 5: MVC arhitektura i komunikacija među komponentama

Na slici 5. prikazana je MVC arhitektura te komunikacija između komponenti, tačnije modela, pogleda i kontrolera.

3.3 Objektno-relaciono mapiranje

Objektno relaciono mapiranje jeste mehanizam koji omogućava rad sa klasama i objektima na aplikativnom nivou, uz automatski rad sa bazom podataka. Skup klasa/funkcija koje generišu odgovarajući SQL kod: dodavanje redova, veza, dovođenje podataka iz baze podataka (prosti i složeni upiti), te modifikovanje baze. ORM se koristi radi lakšeg testiranja koda poslovne logike (nezavisno od baze podataka). Moguća jednostavna promjena baze npr MySQL na PostgreSQL. ORM se najjednostavnije može predstaviti na praktičnom primjeru automobila koji ima niz senzora koji mjere vrijednost tlaka, temperature, brzine, ubrzanja, promjene brzine itd. Sve ove vrijednosti inženjerima su dostupne na njihovim računarima u stvarnom vremenu. Međutim kada sesija završi, inženjeri moraju proučiti i analizirati te podatke kako bi razumjeli kako poboljšati postavku, razviti automobil ili šta je uzrokovalo kvar. Da bi to bilo moguće, potrebno ih je izvesti u bazu podataka. ORM se koristi ovisno i programskom jeziku koji se koristi za implementaciju projekta. Ne može se koristiti bilo koji ORM, potrebno je koristiti ispravni. Npr. za Javu: Hibernate, MyBatis, iBatis, a za Python: Peewee, PonyORM, SQLAlchemy, te za PHP: Doktrina, Propel, Torpor i drugi.

3.4 REST Api

REST (Representational State Transfer) je arhitekturni stil koji se često koristi za projektovanje mrežnih aplikacija. REST API (Application Programming Interface) predstavlja interfejs koji omogućuje komunikaciju između različitih softverskih sistema putem HTTP protokola. Neke od značajki REST API su:

- **Resursi (Resources):**

U REST arhitekturi, sve se smatra resursom, a svaki resurs ima jedinstveni identifikator (URI - Uniform Resource Identifier). Resursi predstavljaju entitete kao što su podaci, servisi ili funkcionalnosti sistema.

- **HTTP Metode:**

REST API koristi HTTP metode za definisanje akcija koje se primenjuju na resurse. Četiri osnovne HTTP metode koje se često koriste u REST API su:

GET: Dobavljanje podataka sa resursa.

POST: Slanje podataka za kreiranje novog resursa.

PUT: Ažuriranje postojećeg resursa.

DELETE: Brisanje resursa.

- **Reprezentacija resursa:**

Podaci koji se prenose između klijenta i servera su u obliku reprezentacije resursa. To može biti JSON, XML ili neki drugi format podataka.

- **Statelessness (Bezstanje):**

REST API je bez stanja, što znači da svaki zahtjev od klijenta ka serveru mora sadržavati sve informacije potrebne za razumevanje i obradu tog zahtjeva. Server ne pamti stanje klijenta između zahtjeva.

- **HATEOAS (Hypermedia As The Engine Of Application State):**

Ideja da se kontrola nad aplikacijom nalazi u prenosivim hipertekstualnim vezama (linkovima) unutar reprezentacije resursa. To omogućava dinamičko otkrivanje i navigaciju kroz aplikaciju.

- **Cacheability (Keširanje):**

REST API može iskoristiti prednost HTTP keširanja za poboljšanje performansi i smanjenje opterećenja servera.

- **Uniform Interface (Jedinstveni interfejs):**

Klijent i server komuniciraju preko jednostavnog i unificiranog interfejsa, što olakšava razvoj, održavanje i skaliranje sistema.

REST API se koristi u razvoju web servisa zbog svoje jednostavnosti, skalabilnosti i lakoće integracije. Klijenti mogu pristupati i manipulirati resursima putem standardnih HTTP zahtjeva, čineći REST jednim od popularnih pristupa za izgradnju interoperabilnih i fleksibilnih softverskih sistema.

3.4.1 Implementacija REST API

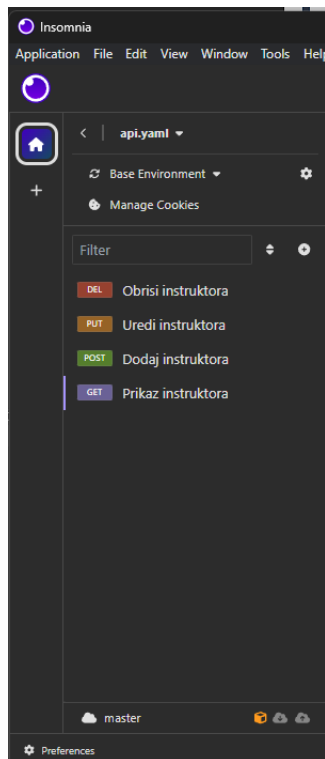
Za implementaciju REST Api u aplikaciju korišten je softver Insomnia. Odbran je model 'Instruktori' nad kojim će se odraditi sve 4 akcije: GET, POST,

PUT i DELETE. Prije same implementacije, potrebno je modifikovati model Instruktora dodavanjem sljedećih funkcija:

```
1 public function index()
2 {
3     $instruktori = DB::table('instruktori')->get();
4     $kandidati = DB::table('kandidati')->get();
5     if (request()->wantsJson()) {
6         return response()->json(['instruktori' => $instruktori
7     ]);
8 }
9
10 return view('instruktori.index', ['instruktori' =>
    $instruktori, 'kandidati' => $kandidati]);
11 }
12
13
14 public function store(Request $request)
15 {
16     error_log($request);
17     return Instruktori::create($request->all());
18 }
19
20 public function update(Request $request, $id)
21 {
22     $instruktor = Instruktori::find($id);
23     $instruktor->update($request->all());
24     return $instruktor;
25 }
26 public function destroy($id)
27 {
28     return Instruktori::destroy($id);
29 }
30 }
```

Kod 1: Modifikacija modela Instruktora za implementaciju REST API

Nakon implementacije, potrebno je u programu Insomnia kreirati novi projekat. U ovom slučaju kreiran je api.yaml. Nakon toga potrebno je kreirati sve 4 akcije GET, POST, PUT i DELETE kako je prikazano na sljedećoj slici.

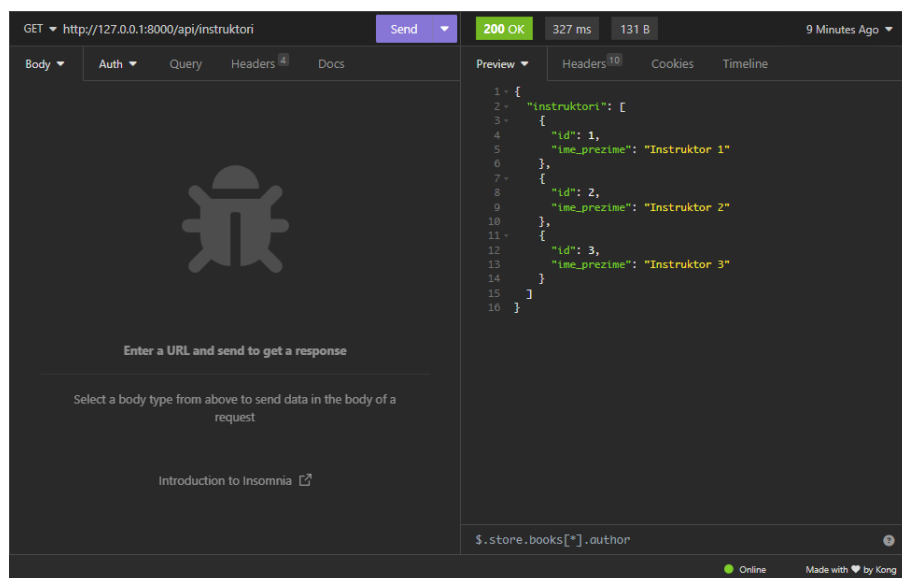


Slika 6: Kreirane akcije u Insomnia softveru

Na slici 6. prikazane su kreirane akcije:

- Prikaz instruktora (GET)
- Dodaj instruktora (POST)
- Obrisi instruktora (DELETE)
- Uredi instruktora (PUT)

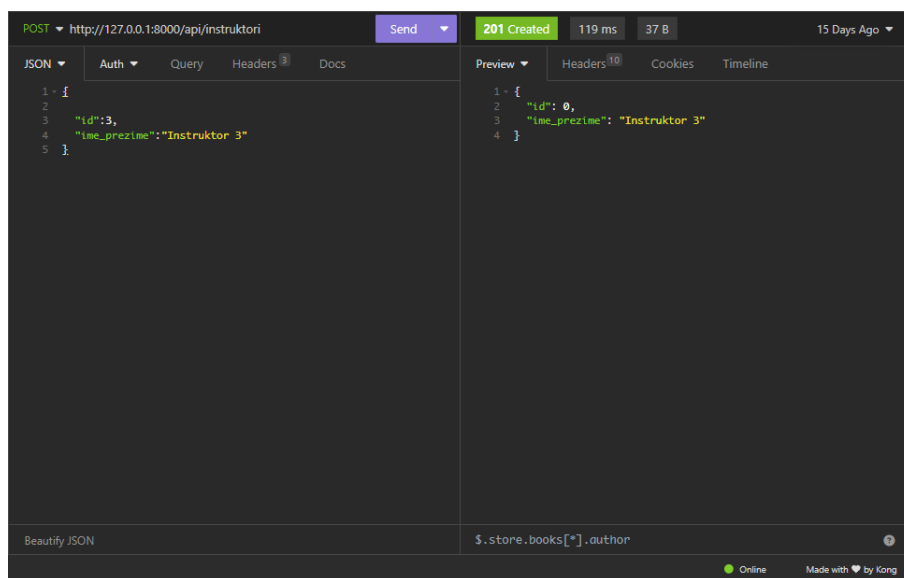
U nastavku će se prikazati implementacija sveke akcije. Prvo je kreirana akcija prikaza instruktora. Šalje se HTTP zahtjev koristeći GET metodu. Poslani zahtjev se šalje na lokalni server (127.0.0.1:8000), a u url se dodaje još '/api/instruktori'. URL koji se šalje izgleda ovako: `http://127.0.0.1:8000/api/instruktori`.



Slika 7: Prikaz instruktora (GET)

Na slici 7. prikazan je rezultat poslanog zahtjeva GET prema aplikaciji. Kao što se može vidjeti, aplikacija nam u JSON formatu vraća sve instruktore iz baze podataka.

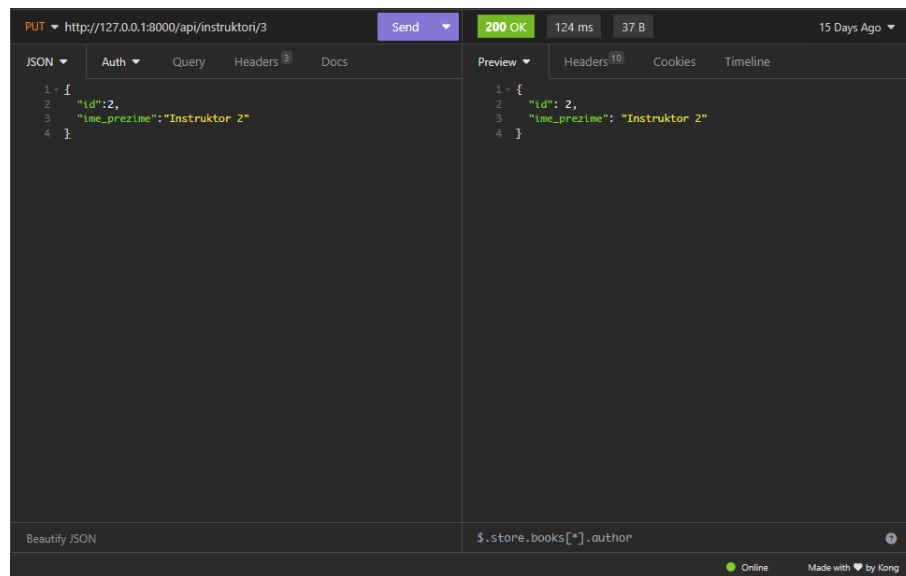
Nakon kreiranja akcije prikaza instruktora, kreira se akcija dodavanja instruktora. Šalje se HTTP zahtjev koristeći POST metodu. Poslani zahtjev se šalje na lokalni server (127.0.0.1:8000), a u url se dodaje još '/api/instruktori'. URL koji se šalje izgleda ovako: `http://127.0.0.1:8000/api/instruktori`.



Slika 8: Dodavanje instruktora (POST)

Na slici 8. prikazan je rezultat poslanog zahtjeva POST prema aplikaciji. Kao što se može vidjeti, kroz JSON fajl se unose podaci koji se žele poslati u bazu, a API vraća kod 201 koji označava da je instruktor uspješno spremljen u bazu podataka.

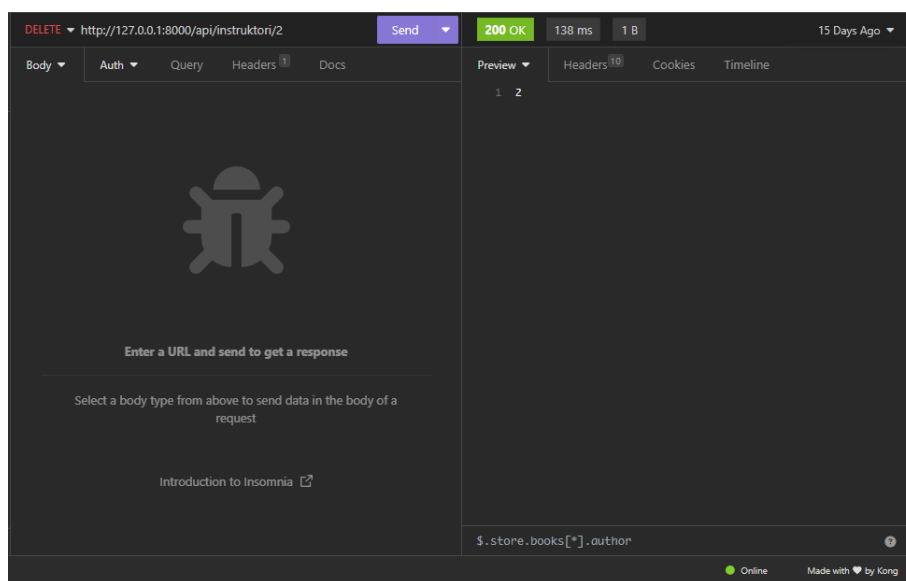
Nakon kreiranja akcije dodavanja instruktora, kreira se akcija uređivanja instruktora. Šalje se HTTP zahtjev koristeći PUT metodu. Poslani zahtjev se šalje na lokalni server (127.0.0.1:8000), a u url se dodja još '/api/instruktori/3', gdje broj 3 označava ID instruktora kojeg se želi urediti. URL koji se šalje izgleda ovako: <http://127.0.0.1:8000/api/instruktori/3>.



Slika 9: Uređivanje instruktora (PUT)

Na slici 9. prikazan je rezultat uređivanja instruktora čiji je id 3. Kao što se može vidjeti, kroz JSON fajl se unose novi podaci koji se žele zamjeniti sa trenutnim podacima u bazi, a API vraća kod 200 koji označava da je zahtjev prošao, te da su podaci instruktora sa id 3 izmjenjeni.

Na kraju, kreirana je završna akcija brisanja instruktora. Šalje se HTTP zahtjev koristeći DELETE metodu. Poslani zahtjev se šalje na lokalni server (127.0.0.1:8000), a u url se dodja još '/api/instruktori/2', gdje broj 2 označava ID instruktora kojeg se želi izbrisati.



Slika 10: Brisanje instruktora (DELETE)

Na slici 10. prikazan je rezultat brisanja instruktora čiji je id 2. Kao što se može vidjeti, URL je poslan id instruktora kojeg se želi obrisati iz baze, a API vraća kod 200 koji označava da je zahtjev prošao, te da je instruktor sa id 2 uspješno izbrisan iz baze.

U nastavku opisać će se sama analiza rada aplikacije, te pojedinačni slučajevi korištenja.

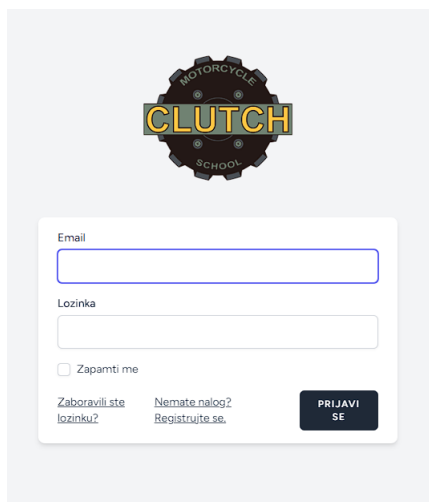
4 Analiza rada aplikacije

Analiza rada web aplikacije je izvršena prikazom pojedinačnih slučajeva korištenja. Prikazani su akteri određenog slučaja korištenja, tok procesa, te povratna informacija ili akcija koja može biti pozitivna ili negativna.

4.1 Opis slučajeva korištenja

Slučaj korištenja 1: Prijava na sistem

- Naziv SK: Prijava na sistem,
- Akteri SK: Korisnik,
- Učesnici SK: Akter i sistem,
- Preduslov: Postoji registrovan korisnik na sistem sa datim korisničkim računom,
- Osnovni scenarij SK:
 - Akter unosi svoju e-mail adresu i lozinku,
 - Sistem provjerava da li ima korisnik i provjerava šifru,
 - Ukoliko su podaci tačni sistem otvara početnu stranicu.



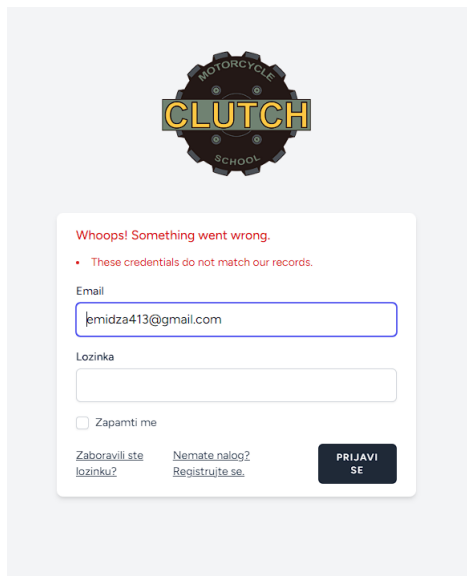
Slika 11: Stranica za prijavu

Kao što se može vidjeti na slici 11., u formi za prijavu je prikazan logo stranice, te polja za unos e-mail adrese i šifre.

- Sporedni scenarij SK:

- Uneseni podaci nisu registrovani u sistemu, sistem ispisuje poruku o netačnim podacima.

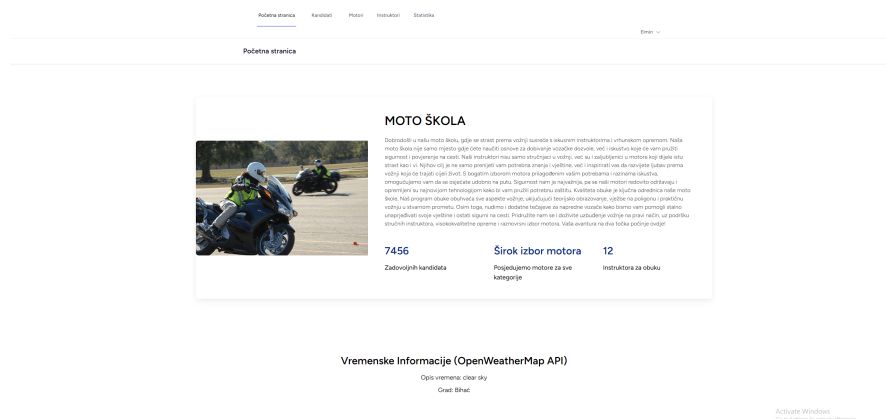
Prikaz stranice za prijavu na sistem u slučaju pogrešnog unosa podataka prikazana je na slici 12.



Slika 12: Netačna prijava

Kao što se može vidjeti na slici 12, na vrhu je ispisan poruka da se unešeni podaci ne poklapaju sa onima koji su spremljeni u bazi podataka.

Prikaz početne stranice, u slučaju uspješne prijave na sistem je predstavljen na slici 13.



Slika 13: Početna stranica

Kao što se može vidjeti na slici 13, na početnoj stranici su ispisani osnovni podaci o moto školi, te je izvršena interakcija sa weather API-jem, koji se nalazi na stranici "openweathermap.org".

Slučaj korištenja 2: Lista kandidata

- Naziv SK: Lista kandidata,
- Akteri SK: Korisnik,
- Učesnici SK: Akter i sistem,
- Preduslov: Akter je prijavljen,
- Osnovni scenarij SK:
 - Akter na navigacionom meniju odabire opciju za listu svih kandidata,
 - Sistem provjerava da je korisnik prijavljen (sesija nije istekla) i vodi ga na stranicu za listu kandidata

Prikaz stranice za pregled svih vježbi je predstavljen na slici 14.

[Početna stranica](#)
[Kandidati](#)
[Materijali](#)
[Instruktori](#)
[Statistika](#)

Eliminiraj

Kandidati

SPISAK KANDIDATA

Dodaj kandidata

Ime i prezime	Datum rođenja	Kategorija polaganja	Dodijeljen instruktor	Status	Akcije			
Zinaid Kapić	1996/05/01	A	Instruktor 1	ZAVRŠIO OBUKU!	Uredi	Obrisi	Choose File	No file chosen
Merjema Porić	2013-05-13	AM	Instruktor 1	ZAVRŠIO OBUKU!	Uredi	Obrisi	Choose File	No file chosen
Sara Cinac	2002-12-03	A1	Instruktor 1	ZAVRŠIO OBUKU!	Uredi	Obrisi	Choose File	No file chosen
Emin Kazferović	2002-05-03	A	Instruktor 2	ZAVRŠIO OBUKU!	Uredi	Obrisi	Choose File	No file chosen
Elmin Midžić	2001-03-04	A	Instruktor 2	NA OBUKU!	Uredi	Obrisi	Choose File	No file chosen
Amel Džanić	1985-12-04	A2	Instruktor 2	NA OBUKU!	Uredi	Obrisi	Choose File	No file chosen
Zahid Kapić	2001-05-10	A1	Instruktor 3	NA OBUKU!	Uredi	Obrisi	Choose File	No file chosen
Erin Tirić	1997-01-01	A2	Instruktor 3	NA OBUKU!	Uredi	Obrisi	Choose File	No file chosen
Elma Vojić	1994-08-10	A1	Instruktor 3	NA OBUKU!	Uredi	Obrisi	Choose File	No file chosen
Ms. Olga Effertz	06/10/1997	A2	NJUE DODJELJEN INSTRUKTOR!	ČEKA NA DODJELU INSTRUKTORA!	Uredi	Obrisi	Choose File	No file chosen

Showing 1 to 10 of 19 results

<

1

2

>

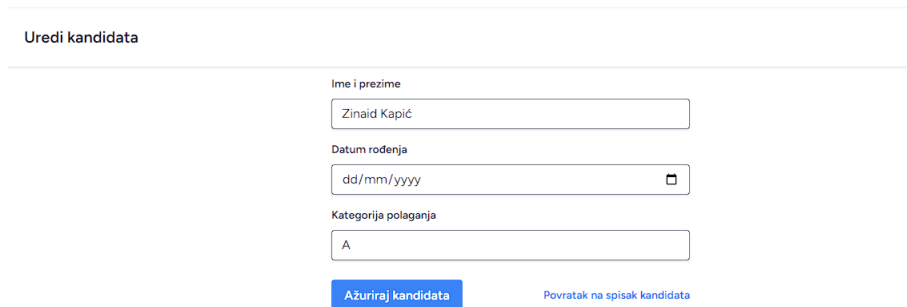
Slika 14: Stranica za pregled kandidata

- Sporedni scenarij SK:
 - Sesija je istekla i korisnik se vodi ponovno na stranicu za prijavu.

Slučaj korištenja 3: Uređivanje podataka o kandidatu

- Naziv SK: Uređivanje podataka o kandidatu,
- Akteri SK: Korisnik,
- Učesnici SK: Akter i sistem,
- Preduslov: Akter je prijavljen,
- Osnovni scenarij SK:
 - Akter pregledom liste kandidata odabire opciju za uređivanje kandidata,
 - Sistem provjerava da je korisnik prijavljen (sesija nije istekla) i vodi ga na stranicu za uređivanje kandidata

Prikaz stranice za uređivanje podataka o kandidatu je predstavljen na slici 15.



Uredi kandidata

Ime i prezime
Zinaid Kapić

Datum rođenja
dd/mm/yyyy

Kategorija polaganja
A

Ažuriraj kandidata [Povratak na spisak kandidata](#)

Slika 15: Stranica za uređivanje podataka o kandidatu

- Sporedni scenarij SK:
 - Sesija je istekla i korisnik se vodi ponovno na stranicu za prijavu.

Slučaj korištenja 4: Dodavanje kandidata

- Naziv SK: Dodavanje kandidata,
- Akteri SK: Korisnik,
- Učesnici SK: Akter i sistem,
- Preduslov: Akter je prijavljen,
- Osnovni scenarij SK:
 - Akter pregledom liste vježbi odabire opciju za dodavanje kandidata,
 - Sistem provjerava da je korisnik prijavljen (sesija nije istekla) i vodi ga na stranicu za dodavanje kandidata

Prikaz stranice za dodavanje kandidata je predstavljen na slici 16.

Dodaj kandidata

Ime i prezime

Datum rođenja

Kategorija polaganja

Dodaj kandidata

[Povratak na spisak kandidata](#)

Slika 16: Stranica za dodavanje kandidata

- Sporedni scenarij SK:
 - Sesija je istekla i korisnik se vodi ponovno na stranicu za prijavu.

Slučaj korištenja 5: Brisanje kandidata

- Naziv SK: Brisanje kandidata,
- Akteri SK: Korisnik,
- Učesnici SK: Akter i sistem,
- Preduslov: Akter je prijavljen,
- Osnovni scenarij SK:
 - Akter pregledom liste vježbi odabire opciju za brisanje kandidata,
 - Sistem provjerava da je korisnik prijavljen (sesija nije istekla) i briše odabranog kandidata iz baze, a zatim ponovno prikazuje ažuriranu stranicu za pregled kandidata

Prikaz brisanja kandidata predstavljen na slici 17.

Kandidati

Kandidat uspješno ubrisan!

SPISAK KANDIDATA

Dodaj kandidata

Ime i prezime	Datum rođenja	Kategorija natjecanja	Dodjeljen instruktor	Status	Akcije			
Zvezd Kapic	1996/05/01	A	Instruktor 1	ZAVRŠIO OBUKU	Uredi	Obriši	Choose File	No file chosen
Megima Ponic	2013-05-13	AM	Instruktor 1	ZAVRŠIO OBUKU	Uredi	Obriši	Choose File	No file chosen
Sara Orac	2002-12-03	A1	Instruktor 1	ZAVRŠIO OBUKU	Uredi	Obriši	Choose File	No file chosen
Emir Kapatirovic	2002-05-03	A	Instruktor 2	ZAVRŠIO OBUKU	Uredi	Obriši	Choose File	No file chosen
Elmir Hadzic	2001-03-04	A	Instruktor 2	NA OBUKU	Uredi	Obriši	Choose File	No file chosen
Amel Džanić	1985-12-04	A2	Instruktor 2	NA OBUKU	Uredi	Obriši	Choose File	No file chosen
Zahid Kapic	2001-05-10	A1	Instruktor 3	NA OBUKU	Uredi	Obriši	Choose File	No file chosen
Erin Tric	1997-01-01	A2	Instruktor 3	NA OBUKU	Uredi	Obriši	Choose File	No file chosen
Elma Vojic	1994-08-10	A1	Instruktor 3	NA OBUKU	Uredi	Obriši	Choose File	No file chosen
Ha. Olga Effertz	06/10/1997	A2	NJE DODJELJEN INSTRUKTOR	ČEKA NA DODJELU INSTRUKTORA	Uredi	Obriši	Choose File	No file chosen

Showing 1 to 10 of 18 results

[1](#)
[2](#)
[3](#)

Slika 17: Brisanje kandidata

- Sporedni scenarij SK:
 - Sesija je istekla i korisnik se vodi ponovno na stranicu za prijavu.

Slučaj korištenja 6: Lista motora

- Naziv SK: Lista motora,
- Akteri SK: Korisnik,
- Učesnici SK: Akter i sistem,
- Preduslov: Akter je prijavljen,
- Osnovni scenarij SK:
 - Akter na navigacionom meniju odabire opciju za listu svih motora,
 - Sistem provjerava da je korisnik prijavljen (sesija nije istekla) i vodi ga na stranicu za listu motora

Prikaz stranice za pregled svih motora je predstavljen na slici 18.

Motori

LISTA MOTORA			
Proizvođač	Model	Tip	Kategorije polaganja
Aprilia	SR 50	Moped (Skuter)	AM
Yamaha	MT-125	Naked	A1
Suzuki	SV650	Sportski Naked	A2
BMW	GS 1200	Sport Touring	A

Slika 18: Stranica za pregled motora

- Sporedni scenarij SK:
 - Sesija je istekla i korisnik se vodi ponovno na stranicu za prijavu.

Slučaj korištenja 7: Lista instruktora

- Naziv SK: Lista instruktora,
- Akteri SK: Korisnik,
- Učesnici SK: Akter i sistem,
- Preduslov: Akter je prijavljen,
- Osnovni scenarij SK:

- Akter na navigacionom meniju odabire opciju za listu svih instruktora,
- Sistem provjerava da je korisnik prijavljen (sesija nije istekla) i vodi ga na stranicu za listu instruktora

Prikaz stranice za pregled svih instruktora je predstavljen na slici 19.

Instruktori	
SPISAK INSTRUKTORA	
ID	Ime i prezime
1	Instruktor 1
2	Instruktor 2
3	Instruktor 3

Slika 19: Stranica za pregled instruktora

- Sporedni scenarij SK:
 - Sesija je istekla i korisnik se vodi ponovno na stranicu za prijavu.

Slučaj korištenja 8: Dodjela instruktora

- Naziv SK: Dodjela instruktora,
- Akteri SK: Korisnik,
- Učesnici SK: Akter i sistem,
- Preduslov: Akter je prijavljen,
- Osnovni scenarij SK:
 - Akter pregledom liste instruktora odabire opciju za dodjelu instruktora,
 - Sistem provjerava da je korisnik prijavljen (sesija nije istekla) i dodjeljuje instruktora kandidatu

Prikaz stranice dodjele instruktora kandidatu je predstavljen na slici 20.

DODJELA INSTRUKTORA

Instruktor:

Instruktor 1 ▾

Kandidat:

Zinaid Kapić ▾

Dodjeli Instruktora

Slika 20: Stranica za dodjelu instruktora

- Sporedni scenarij SK:
 - Sesija je istekla i korisnik se vodi ponovno na stranicu za prijavu.
 - Instruktor ima maksimalan broj kandidata doijeljen (3), te odbija se zahtjev i ispisuje odgovarajuća poruka.

Prikaz stranice dodjele instruktora kandidatu kada je zahtjev odbijen je predstavljen na slici 21.

Instruktori

Instruktor već ima maksimalan broj dodijeljenih kandidata.

SPISAK INSTRUKTORA

ID	Ime i prezime
1	Instruktor 1
2	Instruktor 2
3	Instruktor 3

DODJELA INSTRUKTORA

Instruktor:

Instruktor 1 ▾

Kandidat:

Zinaid Kapić ▾

Dodjeli Instruktora

Slika 21: Stranica za dodjelu instruktora kada je odbijen zahtjev

Na slici 21. može se vidjeti ispisana odgovarajuća poruka da instruktor ima maksimalan broj dodijeljenih kandidata, te da nije moguće obraditi zahtjev.

Slučaj korištenja 9: Prikaz statistike

- Naziv SK: Prikaz statistike,
- Akteri SK: Korisnik,
- Učesnici SK: Akter i sistem,
- Preduslov: Akter je prijavljen,
- Osnovni scenarij SK:
 - Akter na navigacionom meniju odabire opciju za prikaz statistike,
 - Sistem provjerava da je korisnik prijavljen (sesija nije istekla) i vodi ga na stranicu za prikaz statistike

Prikaz stranice statistike je predstavljen na slici 22.

Statistika		
Najstariji kandidat Zinaid Kapić, 1996/05/01	Najmlađi kandidat Merjema Porić, 2013-05-13	Kategorija sa najviše kandidata Kategorija A, 7 kandidata
Kategorija sa najmanje kandidata Kategorija AM, 2 kandidata	Najmanje korišten motor Aprilia, SR 50	Najviše korišten motor BMW, GS 1200

Slika 22: Stranica za prikaz statistike

- Sporedni scenarij SK:
 - Sesija je istekla i korisnik se vodi ponovno na stranicu za prijavu.

4.2 Testiranje rada baze podataka

U ovom dijelu opisan je način testiranja baze podataka. Odrađeni su upiti koji dokazuju rad baze podataka. Laravel omogućava korištenje seedera koji omogućavaju kreiranje testnih podataka. Korištenje factories i seedera znatno olakšavaju posao provjere i testiranja aplikacije. Prilikom kreiranja aplikacije, često je potrebno radi testiranja funkcionalnosti same aplikacije, izvršiti što brže popunjavanje tabela sa podacima. Factory i Seeder se koriste za generisanje testnih podataka za aplikaciju. Factory omogućava da se na vrlo jednostavn način mogu kreirati testni podaci bazirani na modelima. U factory omogućeno je i generisanje podataka koji se odnose na veze, dok u seederima to nije omogućeno. Prilikom realizacije projekta korišteni su testni podaci, prilikom čega su kreirani seederi i factories za svaki model, gdje putem opcije randomElement određujemo ispis određenih kategorija izabranih nekim nedefiniranim redoslijedom.

Testiranje aplikacije rađeno je na dva načina, prvo je napravljeno nekoliko ručnih unosa da bi se provjerila ispravnost formi za unos i uređivanje sadržaja, kao i podstranica za ispis svakog od entiteta. Zatim je koristeći "Database Seeding" funkcionalnost u Laravel-u napravljena funkcija za automatski unos podataka. Na ovaj način je baza podataka automatski ispunjena sa većim (unaprijed odabranim) brojem podataka, što je ubrzalo proces ispunjavanja baze i testiranja ispravnosti aplikacije.

Upit 1: Ispis najstarijeg kandidata

```
1 $najstarijiKandidat = DB::table('kandidati')
2     ->select('*')
3     ->orderBy(DB::raw("STR_TO_DATE(datumRodjenja, '%Y-%m-%d')"),
4         , 'asc')
5     ->limit(1)
6     ->get();
```

Kod 2: Ispis najstarijeg kandidata moto škole

Upit 2: Ispis najmlađeg kandidata

```
1 $najmladjiKandidat = DB::table('kandidati')
2     ->select('*')
3     ->orderBy(DB::raw("STR_TO_DATE(datumRodjenja, '%Y-%m-%d')"),
4         , 'desc')
5     ->limit(1)
6     ->get();
```

Kod 3: Ispis najmlađeg kandidata u moto školi

Upit 3: Ispis kategorija sa najviše kandidata

```
1 $najcescaKategorija = DB::table('kandidati')
2     ->select('kategorijaPolaganja', DB::raw('COUNT(
3         kategorijaPolaganja) AS brojPolaganja'))
4     ->groupBy('kategorijaPolaganja')
5     ->orderBy('brojPolaganja', 'desc')
6     ->limit(1)
```

```
6 ->get();
```

Kod 4: Ispis kategorije polaganja koja ima najveći broj kandidata

Upit 4: Ispis kategorije sa najmanje kandidata

```
1 $minKategorija = DB::table('kandidati')
2   ->select('kategorijaPolaganja', DB::raw('COUNT(
   kategorijaPolaganja) AS brojPolaganja'))
3   ->groupBy('kategorijaPolaganja')
4   ->orderBy('brojPolaganja', 'asc')
5   ->limit(1)
6   ->get();
```

Kod 5: Ispis kategorije polaganja koja ima najmanji broj kandidata

Upit 5: Ispis najviše korištenog motora za obuku

```
1 $kategorijaPolaganja = $najcescaKategorija->first()->
   kategorijaPolaganja;
2 $najcescemotori = DB::table('motori')
3   ->select('proizvodjac', 'model')
4   ->where('kategorijaPolaganja', $kategorijaPolaganja)
5   ->first();
```

Kod 6: Ispis motora koji se najviše koristi za obuku

Upit 6: Ispis najmanje korištenog motora za obuku

```
1 $kategorijaPolaganjaMin = $minKategorija->first()->
   kategorijaPolaganja;
2 $minmotori = DB::table('motori')
3   ->select('proizvodjac', 'model')
4   ->where('kategorijaPolaganja', $kategorijaPolaganjaMin)
5   ->first();
```

Kod 7: Ispis motora koji se najmanje koristi za obuku

Rezultati ovih upita su prikazani na stranici statistike, što je vidljivo na slici 23.

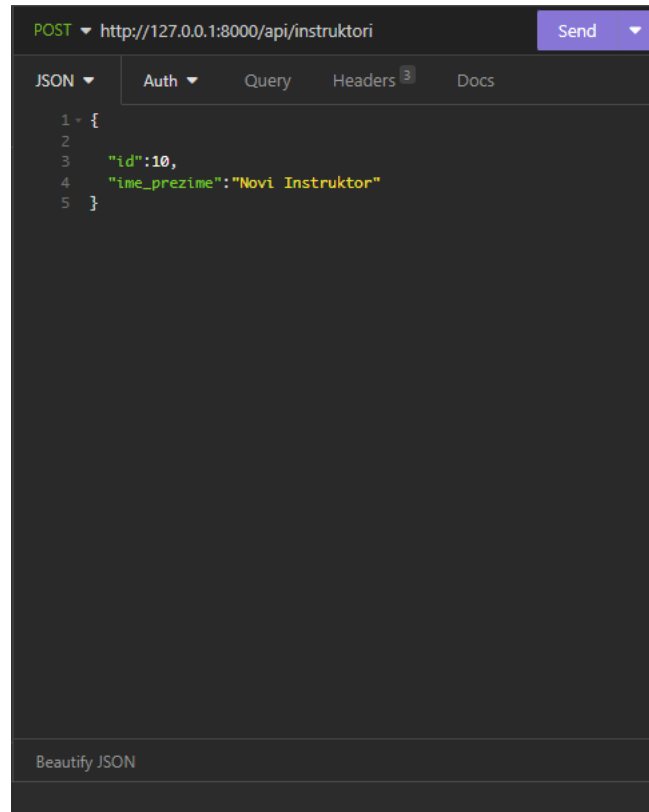
Statistika		
Najstariji kadnidat Zinaid Kapić, 1996/05/01	Najmadi kandidat Merjema Porčić, 2013-05-13	Kategorija sa najviše kandidata Kategorija A, 7 kandidata
Kategorija sa najmanje kandidata Kategorija AM, 2 kandidata	Najmanje korišten motor Aprilia, SR 50	Najviše korišten motor BMW, GS 1200

Slika 23: Prikaz izlaza prethodno prikazanih implementiranih upita

Kao što se može vidjeti na slici iznad, na osnovu prikazanih rezultata je moguće zaključiti da su realizovani upiti od onih najjednostavnijih, pa do nekih složenijih.

4.3 Testiranje API-a

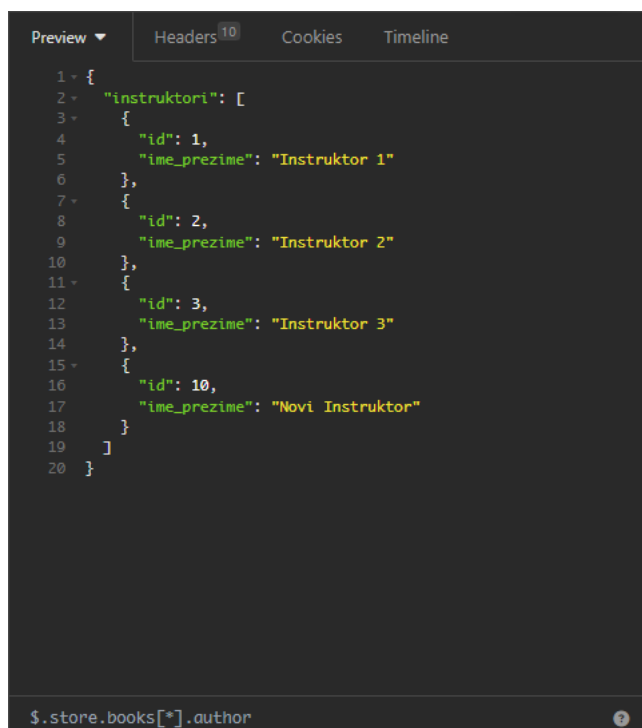
U narednom dijelu prikazat će se testovi kreiranog i implementiranog REST API. Za test, dodat će se novi instruktor sa ID 10 (POST). Nakon toga, uredit će se njegov ID na 4 (PUT). Nakon toga, izlistat će se svi instruktori (GET). Na kraju, obrisat će se instruktor sa ID 4 (DELETE), te ponovo izlistati svi instruktori.



Slika 24: Prikaz dodavanja instruktora sa ID 10

Na slici 24. prikazan je poslani zahtjev za dodavanje instruktora sa ID 10.

Sada će se pokrenuti zahtjev za prikaz svih instruktora, da se provjeri da li je instruktor sa ID 10 dodan u bazu.



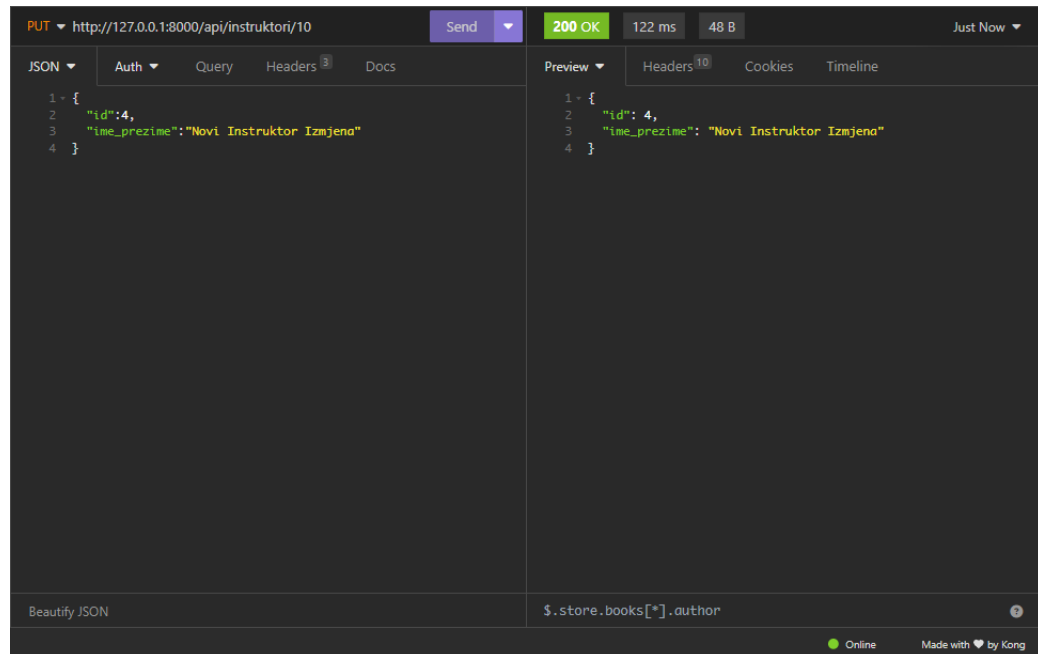
```
1 {
2   "instruktori": [
3     {
4       "id": 1,
5       "ime_prezime": "Instruktor 1"
6     },
7     {
8       "id": 2,
9       "ime_prezime": "Instruktor 2"
10    },
11    {
12      "id": 3,
13      "ime_prezime": "Instruktor 3"
14    },
15    {
16      "id": 10,
17      "ime_prezime": "Novi Instruktor"
18    }
19  ]
20 }
```

\$.store.books[*].author

Slika 25: Prikaz dodavanja instruktora sa ID 10

Na slici 25. prikazana je lista svih instruktora, te može se vidjeti da je na listi također i instruktor sa ID 10. Ovi testovi daju rezultat da POST i GET metoda rade ispravno.

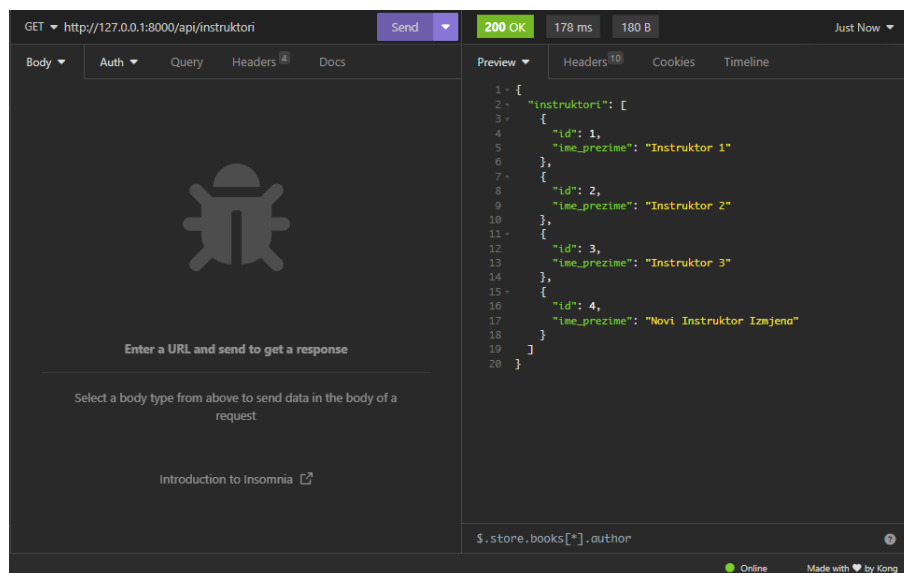
Sada će se odraditi izmjena instruktorovog ID-a na 4, te time testirati PUT metoda, a nakon toga će se izbrisati izmjenjeni instruktor kako bi se testirale preostale dvije metode PUT i DELETE.



Slika 26: Prikaz izmjene instruktora sa ID 10

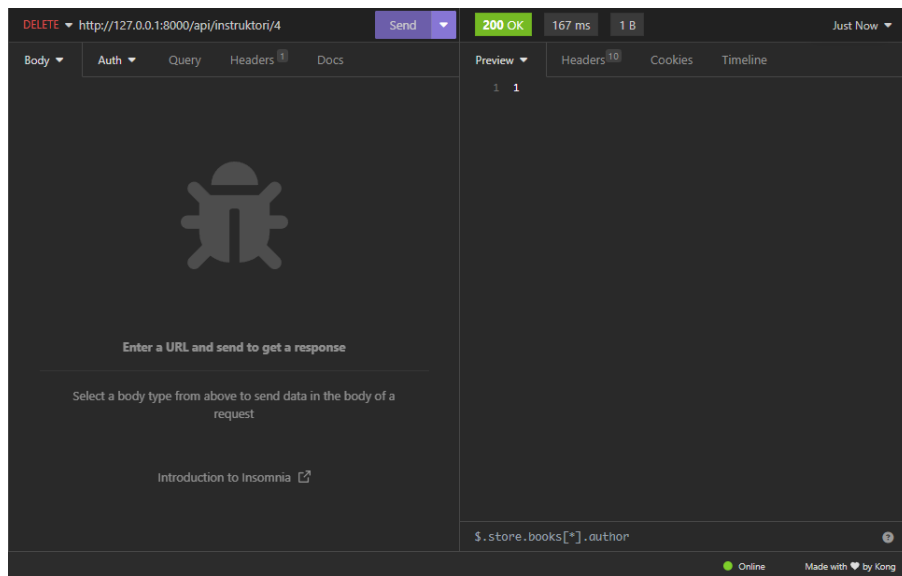
Na slici 26. prikazan je poslani zahtjev za izmjenu instruktora sa ID 10. U url se nalazi ID koji se želi promjeniti, a u JSON fajlu se šalju podaci koji će zamijeniti trenutna podatke u bazi podataka.

Da bi se testirala ispravnost, ponovo će se iskoristiti GET metoda, te prikazati svi instruktori.



Slika 27: Prikaz liste instruktora nakon izmjene

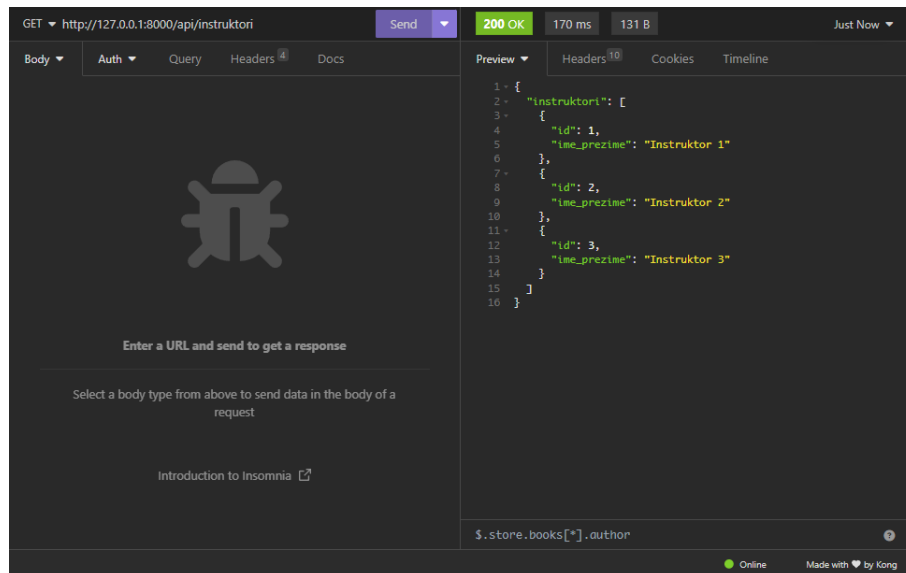
Na slici 27. prikazana je lista svih instruktora, te može se vidjeti da je izmjena instruktora uspješno odrađena. Za kraj, ostaje da se testira DELETE metoda.



Slika 28: Prikaz brisanja instruktora sa ID 4

Na slici 28. prikazan je poslani zahtjev za brisanje instruktora sa ID 4. U url se nalazi ID koji se želi promijeniti. API vraća kod 200 (OK), te vraća 1 kao potvrdu da je zahtjev uspješno odrađen.

Za kraj, ponovo će se izlistati lista svih instruktora.



Slika 29: Prikaz svih instruktora nakon brisanja

Na slici 29. prikazana je lista svih instruktora, te može se vidjeti da nema više instruktora sa ID 4, što znači da je DELETE metoda prošla test i da je zahtjev uspješno odrađen.

5 Zaključak

Zaključak projektnog zadatka moto škole ukazuje na važnost korištenja baza podataka u stvaranju i održavanju informacijskih sistema. Kroz primjenu objektno-relacijskog mapiranja (ORM) u Laravel okviru za izradu online aplikacija, ističe se olakšano modeliranje poslovnih procesa, što rezultira dobro organiziranim kodom i bolje dizajniranom bazom podataka.

Naglašava se ključna uloga MVC arhitekture, koja odvaja logiku, podatke i sloj prikaza, pružajući korisnicima samo neophodne informacije. Kroz UML dijagrame, poput ER i dijagrama slučaja korištenja, projekt ilustrira temelje svakog programa. Objektno-relacijsko mapiranje (ORM) također olakšava pisanje upita za rad s podacima relacijske baze, što rezultira smanjenjem koda i poboljšanom organizacijom.

Iz projekta se može zaključiti da suvremeno stvaranje web aplikacija postaje relativno jednostavno zahvaljujući raznim okvirima, poput Laravela, koji nude unaprijed napravljene funkcije. Kroz testiranje baze podataka prikazuju se prednosti ORM-a, uključujući jednostavnost rukovanja teškim upitima i poboljšanu organizaciju podataka.

Konačni zaključak naglašava da ORM značajno doprinosi razvoju koda, eliminirajući potrebu za ručnim pisanjem upita i stavljajući naglasak na rad s objektima. Rezultat je dobro organizirana web aplikacija koja je lakša za održavanje i ažuriranje. Također, stavlja se naglasak da su testovi API poziva (GET, PUT, POST, DELETE) i testovi složenijih upita uspješno izvršeni. Dodatno, implementiran je API za OpenWeather, što je također uspješno integrirano u projekt. Ovi rezultati potvrđuju funkcionalnost i pouzdanost API-ja unutar projektnog zadatka moto škole, pružajući dodatnu vrijednost korisnicima.

Literatura

- [1] Zagreb, Element.hr, Robert Manger, Baze podataka, 2014.
- [2] Indiana University Bloomington, Alan Dennis, Systems Analysis and Design: An Object-Oriented Approach with UML, 2020.
- [3] Rufus Stewart, Laravel: The Ultimate Beginner's Guide to Learn Laravel Step by Step, 2020.
- [4] Grady Booch, The Unified Modeling Language User Guide, 2005.
- [5] ORM (Object Relational Mapping): automatizira snimanje podataka, Dostupno na linku: <https://www.hwlibre.com/bs/relacijsko-mapiranje-objekta-orm/>
- [6] SQL queries (SQL upiti), Dostupno na linku: <https://edukacija.rs/it/baze-podataka/standardni-upitni-jezik-sql>