RESTful web servisi i rad s JSON-om u Javi

Žegerc, Dora

Undergraduate thesis / Završni rad

2022

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Zagreb, Faculty of Organization and Informatics / Sveučilište u Zagrebu, Fakultet organizacije i informatike**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:211:070899

Rights / Prava: Attribution 3.0 Unported/Imenovanje 3.0

Download date / Datum preuzimanja: **2024-03-29**

SVEUČILIŠTE U ZAGREBU
FAKULTET ORGANIZACIJE I INFORMATIKE
VARAŽDIN

Repository / Repozitorij:

Faculty of Organization and Informatics - Digital Repository

zir.nsk.hr

DIGITALNI AKADEMSKI ARHIVI I REPOZITORIJI

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

VARAŽDIN

Dora Žegerc

RESTful web servisi i rad s JSON-om u Javi

ZAVRŠNI RAD

Varaždin, 2022.

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

VARAŽDIN

Dora Žegerc

Matični broj: 0016140587

Studij: Primjena informacijske tehnologije u poslovanju

RESTful web servisi i rad s JSON-om u Javi

ZAVRŠNI RAD

Mentor/Mentorica:

Doc. dr. sc. Andročec Darko

Varaždin, 2022.

Dora Žegerc

Izjava o izvornosti

Izjavljujem da je moj završni/diplomski rad izvorni rezultat mojeg rada te da se u izradi istoga nisam koristio drugim izvorima osim onima koji su u njemu navedeni. Za izradu rada su korištene etički prikladne i prihvatljive metode i tehnike rada.

Autor/Autorica potvrdio/potvrdila prihvaćanjem odredbi u sustavu FOI-radovi

_____

Sažetak

U ovom radu će fokus biti na RESTful web servisima. U početku će biti objašnjene web aplikacije, te prednosti i nedostatci web aplikacija. Zatim će biti predstavljeni web servisi općenito i vrste web servisa. Osvrt će biti i na prikaz razlike između SOAP i REST servisa. Nadalje, detaljnije će biti analizirani RESTful web servisi, kao i ograničenja koja postoje. Analiza i primjer HTTP metoda koje se koriste u RESTful web servisima. Također će biti detaljnije analiziran i objašnjen JSON format za razmjenu podataka. Bit će objašnjen i programski jezik Java kao i MySQL baza podataka u kojemu će biti implementiran praktični dio ovog rada. Na kraju će biti opis i postupci razvoja aplikacije MyGym u NetBeans razvojnom okruženju.

Ključne riječi: json, java, restful, mysql

Sadržaj

1. Uvod 1
2. Web aplikacije 2
3. Web servisi 4
3.1. Vrste web servisa 5
2. RESTful web servisi 7
2.1. Ograničenja 8
2.1.1. Klijent – poslužitelj 8
2.1.2. Nepostojanje stanja 8
2.1.3. Pred memorija 8
2.1.4. Jedinstveno sučelje 9
2.1.5. Slojeviti sustav 9
2.1.6. Kod na zahtjev 9
2.3. HTTP metode RESTful web servisa 9
3. JSON (JavaScript Object Notation) 11
3.1. Sintaksa podataka 11
3.2. Vrste podataka 11
3.3. Obrada podataka 12
3.4. JSON Shema 12
3. Programski jezik Java 14
3.1. Java i RESTful web usluge 16
3.2. Java i JSON 16
4. MySQL baze podataka 18
4.1. MySQL Workbench 18
4.2. ERA dijagram 19
5. Razvoj aplikacije "MyGym" 21
5.1. Stvaranje RESTful web servisa 21
6. Zaključak 28
7. Literatura 29

1. Uvod

Web aplikacije temelje se na nekom web pregledniku i koriste razne tehnologije za obavljanje zadataka na webu. Neke aplikacije koriste skripte samo na poslužiteljskoj ili samo na klijentskoj aplikaciji, dok neke koriste oboje u isto vrijeme. Tu dolazi važnost web servisa, pomoću kojeg komuniciraju klijentska i poslužiteljska aplikacija. Tako poslužiteljska aplikacija upravlja resursima i odgovara na zahtjeve klijenta od strane klijentske aplikacije putem odgovora web servisa. Kada ja riječ o web servisima, govori se o SOAP i REST web servisima. Pomoću oba servisa se mogu izraditi web usluge, no postoje neke razlike između korištenja svakoga od njih. Jedna važna razlika je ta da SOAP ne može koristiti REST iz razloga što je SOAP protokol, dok je kod REST usluga to moguće iz razloga što je to koncept, koji također može koristiti i druge protokole.

RESTful web servisi sadrže nekoliko načela koja ih čine jednostavnim laganim i brzim. Neki od tih načela su: identificiranje resursa putem URI-a, samoopisne poruke, jedinstveno sučelje i interakcija sa stanjem putem hiperveza, detaljniji opis svakog od načela slijedi u nastavku. U RESTful servisima postoji specifičan skup ograničenja koja su primijenjena na elemente unutar arhitekture. Temelje se na HTTP protokolu i njegovim metodama. Metode koje se koriste su get, post, put, i delete. Da bi klijent koji stvara REST servise i poslužitelj koji ih koristi mogli komunicirati, moraju koristiti određeni format za razmjenu podataka. Najčešće korišteni format za razmjenu podataka je JSON, iz razloga što se temelji na sintaksi doslovnih vrijednosti JavaScript objekata. Uz to, neovisan je o programskom jeziku i računalnoj platformi što predstavlja jednu od velikih prednosti njegova korištenja.

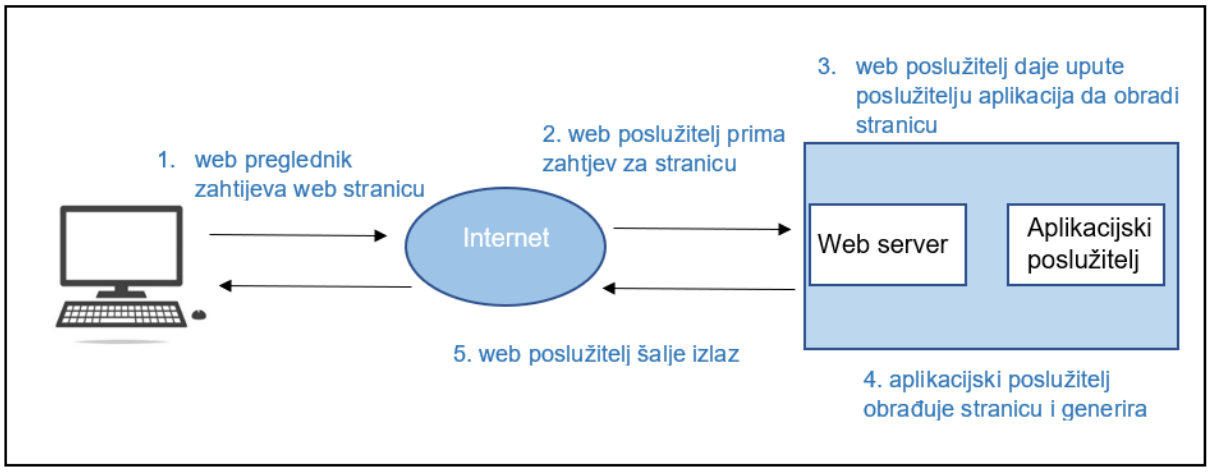
2. Web aplikacije

Web aplikacije su vrsta računalnog programa koji najčešće radi uz pomoć nekog web preglednika (Chrome, Mozilla Firefox, Internet Explorer i sl.), te koriste razne web tehnologije za obavljanje raznih zadataka na internetu. Najčešće koriste kombinaciju skripti na poslužiteljskoj strani (PHP, ASP i sl.) za upravljanje informacijama te pohranom i dohvaćanjem podataka. Neki od njih koriste i skripte na klijentskoj strani (JavaScript, HTML i sl.) za prikazivanje podataka i informacija korisnicima, dok neke od web aplikacija koriste i klijentsku i poslužiteljsku stranu u isto vrijeme [10].

Najčešće su kodirane u jezicima koje podržavaju gotovo svi web preglednici. Razlikujemo statične i dinamičke web aplikacije. Statične web aplikacije prikazuju isti sadržaj za sve korisnike i ne zahtijevaju nikakvu obradu na poslužitelju, te se taj sadržaj ne mijenja od korisnika do korisnika. Dinamičke web aplikacije zahtijevaju obradu na strani poslužitelja i prikazuju sadržaj koji se može mijenjati od korisnika do korisnika ili s vremena na vrijeme (primjer: Gmail, Amazon, Yahoo i sl.)[10].

Dakle, za rad web aplikacija potreban je neki web poslužitelj te aplikacijski poslužitelj. Aplikacijski poslužitelj služi za obavljanje zadataka od strane korisnika, koji ponekad može trebati i bazu podataka koja služi za pohranu podataka. Neke od tehnologija koje koristi aplikacijski poslužitelj su: PHP, JSP, ASP.NET, ASP i sl. Većina dostupnih web aplikacija na internetu je napisana korištenjem jezika HTML (HyperText Markup Language), CSS (Cascading Style Sheets) i Javascript koji se koriste na klijentskoj strani (frontend). Dok se na strani poslužitelja (backend) koriste jezici Java, Python, PHP i Ruby [10].

Na slici 1. prikazan je jedan tipičan tijek web aplikacije. Najčešće, korisnik je taj koji šalje zahtjev web poslužitelju koristeći neke od web preglednika kao što su Google Chrome, Firefox, Microsoft Edge, Internet Explorer i sl. Web poslužitelj tada prosljeđuje zahtjev odgovarajućem aplikacijskom poslužitelju. Zatim aplikacijski web poslužitelj izvršava zadatke koji su mu zadani, kao što su obrada baze podataka, postavljanje upita nad bazama podataka i sl., te na temelju toga proizvodi rezultat traženih podataka. Nakon obavljenih zadataka, aplikacijski poslužitelj šalje web poslužitelju dobiveni rezultat zajedno sa traženim podacima ili obrađenim podacima. Na kraju web poslužitelj odgovara korisniku traženim ili obrađenim podacima te daje rezultat na korisnikov zahtjev [10].



Slika 1: Tijek web aplikacije (Prema: https://www.javatpoint.com/web-application)

Prednosti web aplikacija:

- Fleksibilnost, korisnik može koristiti web aplikaciju neovisno o tome gdje se nalazi sve dok ima ispravnu internetsku vezu
- Dostupne su i mogu se pokrenuti na bilo kojem operacijskom sustavu
- Svi korisnici su u mogućnosti pristupiti istoj verziji aplikacije, tako da se smanjuje problem s kompatibilnošću
- Eliminiraju problem uz ograničenje prostora zbog toga što ih nije potrebno instalirati lokalno
- Smanjenje troškova za krajnje korisnike zbog znatno manjih troškova za održavanje
- Jednostavno stvaranje novog korisnika, potrebno je korisničko ime, lozinka i URL
- Mogućnost programiranja u više programskih jezika kako bi radile na velikom broju operacijskih sustava [10]

Nedostatci web aplikacija:

- Za pristup je neophodna internetska veza
- Prilikom izrade bitno je uzeti u obzir da aplikacija mora podržavati nekoliko web preglednika kao i njihove starije i novije verzije
- Kod programiranja potrebno je napraviti da se mogu koristiti na bilo kojem operacijskom sustavu [10]

3. Web servisi

Web servisi se mogu definirati na više načina, no najčešće predstavljaju proizvod u obliku funkcije za višekratnu upotrebu koju je izradila jedna tvrtka i koji je postavljen unutar mreže kako bi se i drugi mogli njime koristiti. Najveća prednost web servisa je ta što su neovisni o platformi [7, str. 73].

Osnovni cilj web servisa na visokoj razini je unapređenje distribuiranog računalstva (u kojem je logika aplikacije podijeljena na logičke dijelove i radi na više računala). Praktični razlog izgradnje web servisa je taj da se omogući međusobno korištenje računalnih metoda koje su samo opisujuće, te se mogu ponovno koristiti. Web servis je sam po sebi dio koda kojeg mogu pozvati druge aplikacije ili udaljeni procesi. A praktični cilj im je omogućiti tvrtkama da svoje poslovanje fokusiraju na osnovne poslovne potrebe, te da Web servise pozivaju kako bi nadopunili svoje osnovnu djelatnost [7, str. 74].

Postoji razlika između web stranice i web usluge, a to je upotreba XML-a koji se koristi za definiranje i kontrolu podatka koji se šalju na web servis ili se dobivaju od web servisa. Web servisi su definirani zbirkom alata i specifikacija koje programeri koriste za njihovu izgradnju, a sve funkcionira iz razloga što gotovo svi koriste isti skup standardnih specifikacija [7, str. 74].

Kada je riječ o stvaranju web servisa, ono uključuje pisanje, izlaganje i registraciju za sve ovlaštene korisnike kako bi ih vidjeli i mogli koristiti, a samo korištenje servisa podrazumijeva pronalaženje i povezivanje s web servisom na predvidljiv i stabilan način. Također, za bolje razumijevanje web servisa potrebno je definirati pojmove klijent i poslužitelj. Poslužitelj predstavlja uređaj koji ima mogućnost upravljanja resursima, odnosno uređaj koji odgovara na zahtjeve koje dobije. Kontrolira izlaz informacija putem odgovora web servisa. S druge strane, klijent je svaka aplikacija koja se oslanja na poslužitelja kako bi izveo određene operacije [7, str. 74-75].

3.1. Vrste web servisa

Postoje dvije vrste web servisa, to su SOAP, što predstavlja akronim od Simple Object Access Protocol i REST, što predstavlja akronim od Representational State Transfer.

SOAP je protokol koji je temeljen na XML formatu zapisa putem kojeg se pristupa web uslugama. Neke od prednosti koje ovaj protokol ima su prije svega sigurnost, zato što sam definira vlastitu sigurnost, te je neovisan o platformi na kojoj se koristi i programskom jeziku u kojemu je pisan [6].

REST je arhitektonski stil za pružanje standarda između računalnih sustava na webu, te olakšava međusobnu komunikaciju sustava. Servisi koji su usklađeni s REST-om se najčešće nazivaju RESTful web servisi, o njima će više biti riječ u sljedećim poglavljima.
U tablici su prikazane neke važne razlike između SOAP i REST web servisa:

Tablica 1: Razlika između REST i SOAP servisa

SOAP	**REST**
Protokol	Arhitektonski stil
SOAP ne može koristiti REST zato što je to protokol	REST može koristiti SOAP zato što je to koncept, te također može koristiti i druge protokole
SOAP koristi servisno sučelje da bi predstavio poslovnu logiku	REST koristi URI da bi predstavio poslovnu logiku
SOAP definira standarde koje je potrebno strogo slijediti	REST ne definira previše standarda
SOAP zahtjeva više propusnosti i resursa	REST zahtjeva manje propusnosti i resursa
SOAP definira vlastiti sigurnosni sloj	REST nasljeđuje sigurnosne mjere od temeljenog transporta
SOAP dopušta samo XML format podataka	REST dopušta različite formate podataka

(Izrađeno prema: https://www.javatpoint.com/soap-vs-rest-web-services)

Servisno orijentirana arhitektura (SOA) definirana je kao obrazac dizajna koji je dizajniran s ciljem izgradnje distribuiranih sustava koji drugim aplikacijama isporučuju usluge putem protokola. Ovakav koncept nije ovisan o platformi na kojoj se koristi kao ni o programskom jeziku. Usluga je samostalna funkcija koja predstavlja jedinicu funkcionalnosti i u mogućnosti je razmjenjivati informacije s druge usluge, bez da ovisi o stanju te druge usluge [6].

Slika prikazuje servisno orijentiranu arhitekturu. Korisnik neke usluge šalje zahtjev za tu uslugu davatelju usluge, a davatelj te usluge šalje odgovor na uslugu.

Slika 2: Ilustracija SOA-e (Izvor: https://www.javatpoint.com/service-oriented-architecture)

Neke od prednosti servisno orijentirane arhitekture [6]:

- Jednostavna je za integraciju
- Zbog specifikacije usluge, složenosti se izoliraju, a samim time integracija postaje lakša za upravljanje, te se tako može upravljati složenošću
- Usluge su neovisne o platformi i tako mogu komunicirati s drugim aplikacijama preko zajedničkog jezika
- Olakšava implementaciju usluga bez da time utječe na druge aplikacije ili usluge
- Omogućuje paralelni razvoj, jer SOA slijedi arhitekturu temeljenu na slojevima
- Laka dostupnost svih SOA usluga svakom podnositelju zahtjeva
- Male veličine usluga doprinose lakšem testiranju i samim time otklanjanju pogrešaka

2. RESTful web servisi

REST predstavlja akronim za Representational state of transfer, odnosno reprezentativno stanje prijenosa. Reprezentativni prijenos je arhitektonski stil koji određuje ograničenja, kao npr. jednoobrazno sučelje, koje ako se primjenjuju na web uslugu induciraju poželjna svojstva, kao što su izvedbe, skalabilnosti i prilagodljivosti, koja uslugama omogućuju bolji rad na web-u. Podaci i funkcionalnosti se u REST arhitektonskom stilu smatraju resursima i pristupa im se pomoću jedinstvenih identifikatora resursa (u daljnjem tekstu URI). Klijenti i poslužitelji u stilu arhitekture REST, razmjenjuju prikaze resursa pomoću standardiziranog sučelja i protokola [1, str. 482].

Postoji nekoliko načela koje potiču RESTful aplikacije da budu jednostavne, lagane i brze [5]:

- Identifikacija resursa putem URI-a: RESTful web usluge izlažu skup resursa koji identificiraju ciljeve interakcije sa svojim klijentima, a resurse identificiraju jedinstveni identifikatori resursa (URI), koji pružaju prostor za otkrivanje resursa i usluga.
- Jedinstveno sučelje: pomoću operacija PUT, GET, POST i DELETE se manipulira resursima. PUT se koristi za stvaranje nekog novog resursa koji se zatim može izbrisati s DELETE. GET dobiva trenutno stanje resursa u nekom predstavljanju, dok POST prenosi novo stanje na resurs.
- Samoopisne poruke: sadržaju resursa se može pristupiti u različitim formatima, a to su HTML, XML, tekst, PDF, JPEG, JSON i još mnogi drugi. Meta podaci o resursu su dostupni i koriste se za kontrolu pred memoriranja, otkrivanje pogrešaka u prijenosu, dogovaranje odgovarajućeg formata prikazivanja i izvođenje provjere autentičnosti ili kontrole pristupa.
- Interakcije sa stanjem putem hiperveza: interakcije sa stanjem se temelje na konceptu eksplicitnog prijenosa stanja. Zahtjevi su bez stanja. Postoji i nekoliko tehnika za razmjenu stanja, kao što su prepisivanje URI-ja, kolačića i polja skrivenih obrazaca. Stanja se mogu ugraditi u poruke odgovora tako da bi ukazale na valjana buduća stanja interakcije.

Fain [1, str. 482] definira resurs kao svaki objekt kojem se može pristupiti korištenjem hiperveze, te svaki taj resurs ima svoj URI. Također navodi da REST resursi moraju podržavati standardne HTTP (Hypertext Transfer Protocol) zahtjeve bez stanja. Standardne HTTP metode su get, post, put i delete.

2.1. Ograničenja

Arhitektonski stil se sastoji od skupa ograničenja koja su primijenjena na elemente unutar arhitekture. Ispitivanjem svakog od ograničenja mogu se identificirati svojstva inducirana ograničenjima web-a. Na taj način se mogu primijeniti dodatna ograničenja kojima bi se oblikovao novi arhitektonski stil koji bolje održava željena svojstva moderne web arhitekture. U nastavku će biti objašnjena ograničenja koja su specifična za REST stil [2].

2.1.1. Klijent – poslužitelj

Komponente klijent – poslužitelj možemo objasniti na način da klijent predstavlja okidački proces, a poslužitelj predstavlja reaktivni proces. Što znači da klijent upućuje na zahtjeve koji izazivaju reakcije kod poslužitelja, poslužitelj čeka na izradu zahtjeva, a nakon toga reagira na njih. Poslužitelj je zapravo proces koji se ne završava, te pruža usluge više od jednog klijenta [2].

Fielding [2] navodi kako je razdvajanje problema princip koji stoji iza ograničenja klijent – poslužitelj. Razdvajanje problema korisničkog sučelja od zabrinutosti za pohranu podataka, dovodi do poboljšavanja skalabilnosti pojednostavljujući tako poslužiteljske komponente. Najvažnija prednost razdvajanja za web je ta što omogućuje da se komponente razvijaju neovisno, podržavajući tako zahtjev za internetskim razmjerima više organizacijskih domena.

2.1.2. Nepostojanje stanja

Ograničenje nepostojanje stanja (eng. Stateless) se dodaje interakciji klijent – poslužitelj, to bi izgledalo kao klijent – bez poslužitelja, što znači da svaki zahtjev klijenta za poslužitelja mora sadržavati sve potrebne podatke za razumijevanje zahtjeva. Tako je stanje sesije u potpunosti na klijentu. Nepostojanje stanja inducira svojstva vidljivosti, pouzdanosti i skalabilnosti. Sustav praćenja ne mora gledati dalje od jednog zahtjeva da bi se utvrdila puna priroda zahtjeva i tako je poboljšana vidljivost. Pouzdanost je poboljšana tako što olakšava oporavak od djelomičnih kvarova. Nepotrebno pohranjivanje stanja između zahtjeva omogućuje brzo oslobađanje komponente poslužitelja resursa, te tako poboljšava skalabilnost i još dodatno pojednostavljuje implementaciju jer poslužitelj ne mora upravljati uporabom resursa u svim zahtjevima. Nedostatak ovog ograničenja je taj što može smanjiti mrežne performanse povećanjem podataka koji se ponavljaju poslani u nekom nizu zahtjeva [2].

2.1.3. Pred memorija

Ograničenje pred memorije poboljšava mrežnu učinkovitost, te zahtijeva da podaci unutar odgovora na zahtjev budu implicitno ili eksplicitno označeni kao pred memorirani ili kao oni koje nije moguće pred memorirati. Ako je situacija takva da je odgovor moguće pred memorirati, onda se na klijentskoj pred memoriji daje pravo ponovne upotrebe tih podataka

odgovora za kasnije, ekvivalentne zahtjeve. Prednost korištenja pred memorije je u tome što imaju potencijal djelomičnog ili potpunog uklanjanja nekih interakcija, poboljšavajući tako učinkovitost, skalabilnost kao i izvedbu koju percipiraju korisnici smanjenjem prosječne latencije niza interakcija [2].

2.1.4. Jedinstveno sučelje

Značajka koja razlikuje REST arhitektonski stil od drugih mrežnih stilova je jedinstveno sučelje. Naglasak se stavlja na ujednačeno sučelje između komponenata. Načela općenitosti softverskog inženjerstva koja se primjenjuju na sučelje komponenata, pojednostavljuju ukupnu arhitekturu sustava i poboljšavaju vidljivost interakcija. Nedostatak jedinstvenog sučelja je taj što pogoršava učinkovitost, iz razloga što se informacije prenose u standardiziranom obliku, a trebale bi se prenositi u obliku koji je specifičan za potrebe određene aplikacije. Za postizanje jedinstvenog sučelja potrebna su višestruka arhitektonska ograničenja [2].

2.1.5. Slojeviti sustav

Prema Fielding [2] slojevita ograničenja sustava se dodaju kako bi se dodatno poboljšalo ponašanje zahtjeva na Internet. Slojevitim stilom se omogućava da se arhitektura sastoji od hijerarhijskih slojeva ograničavajući tako ponašanje komponenti da svaka komponenta ne može „vidjeti“ izvan svog neposrednog sloja s kojim međusobno djeluje. Prednost slojevitih sustava je ta što dodaju latentnost obradi podataka, te time smanjuju performanse koje uočavaju korisnici. Za sustav koji je temeljen na mreži, te koji podržava ograničenja pred memorije, to se može poništiti pomoću zajedničkog pred memoriranja kod posrednika.

2.1.6. Kod na zahtjev

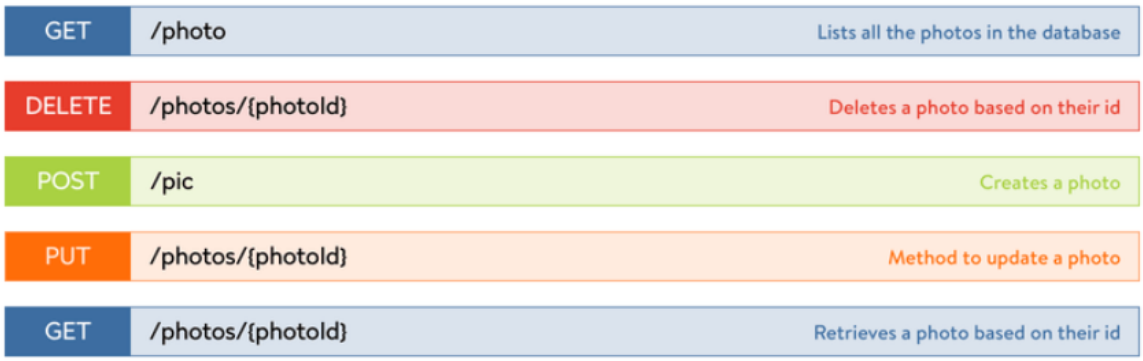
Kod na zahtjev predstavlja ograničenje koje omogućava da se funkcija klijenta proširi za preuzimanje i izvršavanje koda u obliku aplikacija ili skripti. To rezultira smanjivanjem broja značajki koje je potrebno prethodno implementirati. Ograničenje je opcionalno zato što omogućuje dizajniranje arhitekture koja podržava željeno ponašanje u općem slučaju, uz razumijevanje da ista može biti onemogućena unutar nekog konteksta [2].

2.3. HTTP metode RESTful web servisa

Purushothaman [3] definira HTTP (eng. Hypertext transfer protocol) kao protokol koji definira način na koji se oblikuju, obrađuju i prenose poruke putem interneta. Dok Rodriguez [4] navodi da se RESTful web usluge temelje na HTTP protokolu i njegovim metodama. Osnovno načelo REST dizajna uspostavlja pojedinačno mapiranje između CRUD operacija

(eng. Create, read, update, delete) i HTTP metoda. Prema tom mapiranju, HTTP metode se koriste na sljedeći način:

- POST se upotrebljava da bi stvorili resurs na poslužitelju
- GET se upotrebljava da bi dohvatili taj resurs, odnosno da bi ga prikazali, ova metoda se koristi samo za čitanje podataka
- PUT se upotrebljava da bi promijenili stanje resursa ili da bi ga ažurirali
- DELETE se upotrebljava kada se želi ukloniti ili izbrisati određeni resurs



Slika 3: Primjer glavnih HTTP metoda (Izvor: https://yalantis.com/blog/how-to-create-a-restful-api/)

3. JSON (JavaScript Object Notation)

JSON (eng. JavaScript Object Notation) je jednostavan format za razmjenu podataka, koji je čitljiv i razumljiv ljudima. Format JSON se temelji na sintaksi doslovnih vrijednosti JavaScript objekata, ali ga se smatra jezično neovisnim oblikom zbog njegovog širokog usvajanja u svim programskim jezicima. Stoga možemo reći kako je JSON format neovisan o programskom jeziku i računalnoj platformi [3].

3.1. Sintaksa podataka

Kao što je prethodno navedeno, JSON format je vrlo jednostavan format za razmjenu podataka, pa je tako jednostavan i po dizajnu, te ga predstavljaju sljedeće dvije strukture podataka [3]:

- Neuređena zbirka parova naziva i vrijednosti (predstavljaju objekt) podrazumijeva atribute objekta te njihove vrijednosti koji su prikazani u obliku para naziva i vrijednosti. Dvotočkom (:) su odvojeni ime i vrijednosti u paru, a nazivi u objektu su nizovi, te njihove vrijednosti mogu biti bilo koja JSON vrsta podataka. Cijeli objekt je okružen vitičastim zagradama.
- Uređena zbirka vrijednosti (predstavljaju polja): polja su okružena uglatim zagradama ([ ]), dok su njihove vrijednosti odvojene zarezom (,). Svaka od vrijednosti može biti drugačije vrste, bilo da je polje ili objekt.

3.2. Vrste podataka

U ovom poglavlju će biti objašnjeni osnovni tipovi podataka JSON formata koje navodi Purushothaman [3].

- Broj: ova vrsta podataka se koristi za pohranu cijelih ili decimalnih brojeva, decimalni brojevi su odvojeni točkom
- Tekst (eng. String): predstavlja niz od nula ili više znakova, okružen je dvostrukim navodnicima i podržava eng. backslash
- Logički tip podataka (eng. Boolean): vrsta podataka koja predstavlja istinitu ili lažnu vrijednost, ova vrsta podataka se koristi za prikaz je li kondicija istinita ili lažna.
- Polje (eng. Array): vrsta podataka koja predstavlja popis vrijednosti koje mogu biti bilo koje vrste podataka. Vrijednosti su odvojene zarezom i okružene su uglatim zagradama
- Objekt (eng. object): vrsta podataka koja sadrži neuređenu zbirku parova vrijednosti atributa koji su odvojeni zarezom i okruženi su vitičastim zagradama. Svi atributi moraju biti nizovi, te se moraju međusobno razlikovati unutar istog objekta.

- Null: vrsta podataka koja označava praznu vrijednost

3.3. Obrada podataka

Obrada JSON podataka podrazumijeva čitanje, pisanje, ispitivanje i modificiranje istih. Purushothaman [3] objašnjava dva široko usvojena programska modela za obradu JSON podataka:

1. Model objekta (eng. Object model): model u kojem se svi JSON podaci čitaju u memoriju u obliku stabla. Stablo je moguće obraditi, analizirati ili modificirati s odgovarajućim API-jima. Prednost ovog modela je ta što daje veću fleksibilnost kod manipuliranja sadržajem.
2. Model streaminga : pojam streaming u ovom kontekstu znači da se podaci mogu čitati ili pisati u blokovima, stoga, ovaj model ne čita cijeli sadržaj JSON-a u memoriju, nego čita jedan po jedan element. Za svako čitanje tokena, parser generira potrebne događaje koji ukazuju na vrstu tokena.

3.4. JSON Shema

JSON shema predstavlja specifikaciju za format koji je zasnovan na JSON-u za definiranje strukture JSON podataka. Koristi se za kompletnu validaciju konstrukcije, koristan je i za automatizirano ispitivanje, kompletnu provjeru strukture valjanosti podataka koji su dostavljeni od strane klijenta, te se još koristi za definiranje dokumentacije koja je čitljiva ljudima i korisnicima. Primjer JSON sheme prema Tutorialspoint [5]:

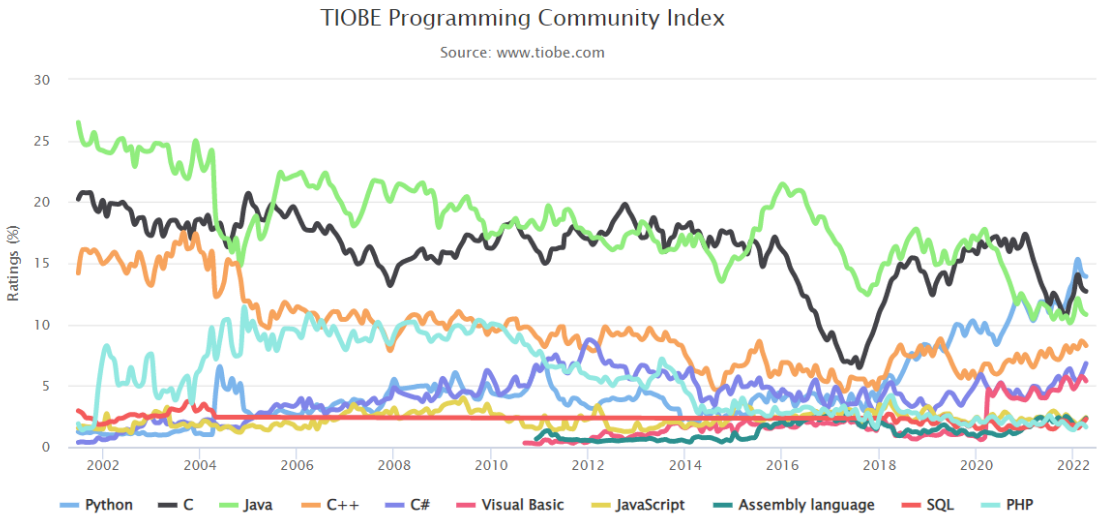
```
{
   "$schema": "http://json-schema.org/draft-04/schema#",
   "title": "Proizvod",
   "description": "Proizvod iz Acemovog kataloga",
   "type": "object",
"properties": {
      "id": {
         "type": "integer"
      },
      "ime": {
         "description": "Ime proizvoda",
         "type": "string"
      },
      "cijena": {
         "type": "number",
```

```
            "minimum": 0,
            "exclusiveMinimum": true
        }
    },
    "required": ["id", "ime", "cijena"]
}
```

Prema ovoj shemi svaki dokument koji će prema njoj provjeravati valjanost mora sadržavati objekt proizvod, te taj objekt mora obavezno sadržavati svoj ID, ime i cijenu. U shemi se nalaze i neke ključne riječi, kao što je *$shema,* ona navodi da je shema napisana prema nacrtu v4 specifikacije. *Title i description* označavaju da shemi dodajemo naslov i mali opis. Ključna riječ *Type* predstavlja prvo ograničenje, a to je da mora biti JSON objekt. *Properties* definira razne ključeve i njihove vrijednosti koje će se kasnije koristiti u JSON datoteci, a u ovom slučaju id ima ograničenje da mora sadržavati cijeli broj, za naziv je stavljeno ograničenje da mora sadržavati niz znakova, dok je za cijenu ograničenje da mora biti broj. Uz ograničenje broj, nalazi se i ograničenje *minimum,* to je ograničenje koje predstavlja minimalnu prihvatljivu vrijednost, što je u ovom slučaju 0. Kod cijene se nalazi još i ograničenje *exclusiveMinimum* koje je vezano za ograničenje *minimum*, a funkcionira tako da, ako *exclusiveMinimum* ima logičku vrijednost *true,* ta instanca je valjana ako je strogo veća od vrijednosti koja je definirana za ograničenje *minimum* [5].

3. Programski jezik Java

Fain [1, str. 1] navodi kako je programski jezik Java u posljednja dva desetljeća jedan od najpopularnijih i najkorištenijih jezika u raznim područjima, od programiranja video igrica do programiranja složenijih i zahtjevnijih aplikacija poput onih za trgovanje na burzama ili za upravljanje svemirskim vozilima.



Slika 4: Graf popularnosti programskih jezika za travanj 2022
(Izvor: https://www.tiobe.com/tiobe-index/)

travnja 2022	travnja 2021	Promijeniti	Programski jezik	Ocjene	Promijeniti
1	3	^	Piton	13,92%	+2,88%
2	1	v	C	12,71%	-1,61%
3	2	v	Java	10,82%	-0,41%
4	4		C++	8,28%	+1,14%
5	5		C#	6,82%	+1,91%
6	6		Visual Basic	5,40%	+0,85%
7	7		JavaScript	2,41%	-0,03%
8	8		Asemblerski jezik	2,35%	+0,03%
9	10	^	SQL	2,28%	+0,45%
10	9	v	PHP	1,64%	-0,19%

Slika 5: Tablica popularnosti programskih jezika
(Izvor: https://www.tiobe.com/tiobe-index/)

Prema podacima indeksa programske zajednice TIOBE (The Software Quality Company) koja je pokazatelj popularnosti programskih jezika i čiji se indeks ažurira svakog mjeseca,

vidljivo je kako se u zadnje vrijeme popularnost Jave smanjuje, no i dalje se nalazi pri vrhu ljestvice, što znači da je i dalje zastupljena. Konkurenstki jezici prema popularnosti su Python i C.

James Gosling je 1995. godine dizajnirao programski jezik Java, radeći u tvrtci Sun Microsystems koju je kasnije kupila tvrtka Oracle Corporation. James-ov cilj prilikom dizajniranja Jave bio je pružiti jednostavniju alternativu jeziku C++ koja će biti neovisna o platformi. „Java je programski jezik opće namjene koji se koristi u svim industrijama i za gotovo sve vrste aplikacija.“ [1, str.1]. Također, Java je i objektno orijentirani jezik, što daje mogućnost povezivanja programske konstrukcije s objektima iz stvarnog svijeta. Java programi se izvršavaju pomoću Java virtualnog stroja (eng. Java Virtual Machine, JVM), koji je isti na svim platformama [1, str. 1-2].

U nekim programskim jezicima kada napišemo izvorni kôd, možemo ga odmah pokrenuti. Takvi jezici se nazivaju interpreterskim jezicima (npr. JavaScript), dok programski jezik Java zahtijeva kompajliranje izvornog kôda prije izvršavanja. Što znači da se izvorni kôd pretvara u kôd koji je specifičan za procesor ili bajtkod kojeg izvršava JVM i koji jedan dio tog bajtkoda može pretvoriti u strojni kôd koji je specifičan za platformu i koristi JIT (Just-In-Time) kompajler [1, str. 3]

Java platforma predstavlja okruženje u kojem se izvode Java aplikacije i postoje četiri takve platforme, a to su Java standardno izdanje (eng. Java Standard Edition, Java SE), Java Enterprise Edition (Java EE), Java mikro izdanje (Java ME) i JavaFX [12].

- **Java SE** definira sve od osnovnih tipova i objekata pa sve do klasa visoke razine koje se koriste za umrežavanje, sigurnost, pristup bazi podataka i sl., pruža osnovnu funkcionalnost programskog jezika Java. Java SE sadrži osnovni API (), a uz njega i virtualni stroj, razvojne alate, tehnologije za implementaciju, te drugih biblioteka klasa i alata koji se općenito koriste u Java aplikacijama [12].
- „**Java EE** platforma pruža API i runtime okruženje za razvoj i pokretanje velikih, višeslojnih, skalabilnih, pouzdanih i sigurnih mrežnih aplikacija.“ [12]
- **Java Me** pruža API i virtualni stroj malog otiska za pokretanje aplikacija napisanih u Javi na manjim uređajima kao što je mobilni telefon. Najčešće, Java ME aplikacije su klijenti usluga Java EE platforme [12].
- **Java FX** predstavlja platformu za stvaranje bogatih internetskih aplikacija pomoći API-ja korisničkog sučelja. Koriste hardverski ubrzane grafičke i medijske mehanizme kako bi iskoristili prednosti klijenata viših performansi i modernog izgleda kao i korištenje AP-ja visoke razine za povezivanje s mrežnim izvorima podataka, a uz to mogu biti i klijenti usluga Java EE platforme [12].

Java platforme se sastoje od JVM-a i API-ja. JVM predstavlja program za određenu hardversku i softversku platformu za pokretanje Java aplikacija, dok API predstavlja skup softverskih komponenti koje je moguće koristiti za stvaranje drugih softverskih komponenti ili aplikacija [12].

3.1. Java i RESTful web usluge

Devedesetih godina prošlog stoljeća web se počeo masovno koristiti, tako su milijuni ljudi širom svijeta pristupali web aplikacijama koje su bile zastarjele, te su bile napisane u raznim programskim jezicima i instalirane na brojnim vrstama računala. Zbog toga je postajala potreba da se podaci tvrtke izlože široj javnosti, što je rezultiralo izradom standardnog sučelja preko kojeg se pristupa podacima preko web-a. Web usluge bi se trebale promatrati kao mehanizam za izlaganje korisnih podataka koje je moguće preuzimati preko HTTP protokola. Tako, tisuće javno dostupnih web usluga nude mnoštvo različitih API-ja za uzimanje različitih vrsta podataka. Činjenica je da velika poduzeća koriste privatne web usluge za interne potrebe [1, str.481].

Simple Object Access Protocol (SOAP) je bio prvi standard za objavljivanje i konzumiranje web usluga, gdje bi web klijenti kreirali HTTP zahtjeve, te primali odgovore koristeći SOAP sintaksu. Da bi klijenti mogli koristiti uslugu moraju znati popis usluga koje organizacija nudi, imena koja se nude i adresu krajnje točke na koju se moraju spojiti. SOAP web usluge su doslovne, ali i dalje se često koriste kao jednostavan način za integraciju sa softverskim proizvodima nekih trećih strana. Neke od njih su i javno dostupne. Web usluge temeljene na REST načelima nazivaju se RESTful web usluge. Pravilo je da REST resursi moraju sadržavati standardne HTTP zahtjeve bez stanja (get, put, update, delete) [1, str. 481- 482].

3.2. Java i JSON

Danas sve veću popularnost zauzima razvoj SPA (Single Page web Application), odnosno jednostranične web aplikacije. One se odnose na aplikacije koje ne ažuriraju cijelu web stranicu, nego samo neke dijelove prilikom preuzimanja podataka ili sl. jednostranične web aplikacije trebaju koristiti određeni format za razmjenu podataka između klijenta i poslužitelja. Zbog popularnosti korištenja JSON formata i Java EE specifikacije standardizira njegovu obradu u dokumentu JSR 353 (Java Specification Requests). Aplikacijski poslužitelji u Javi najčešće implementiraju specifikaciju obrade JSON formata u posebnim JAR-ovima (JavaARchive) koji se temelje na popularnom formatu ZIP datoteke i koji se koriste za združivanje mnogih datoteka u jednu [13], kako bi se mogli koristiti sa samostalnim Java SE aplikacijama [1, str. 484].

javax.json paket sadrži klase koje podržavaju dva načina za formiranje JSON podataka iz Jave:

1. **Object model API**: Java kod izrađuje stablo objekata koje predstavlja JSON podatke u memoriji, te ga šalje u I/O tok.
2. **Streaming API**: generira rezultat u zadani tok, njime upravljaju događaji, te on šalje događaje kad naiđe na početak ili kraj JSON objekta [1, str. 484].

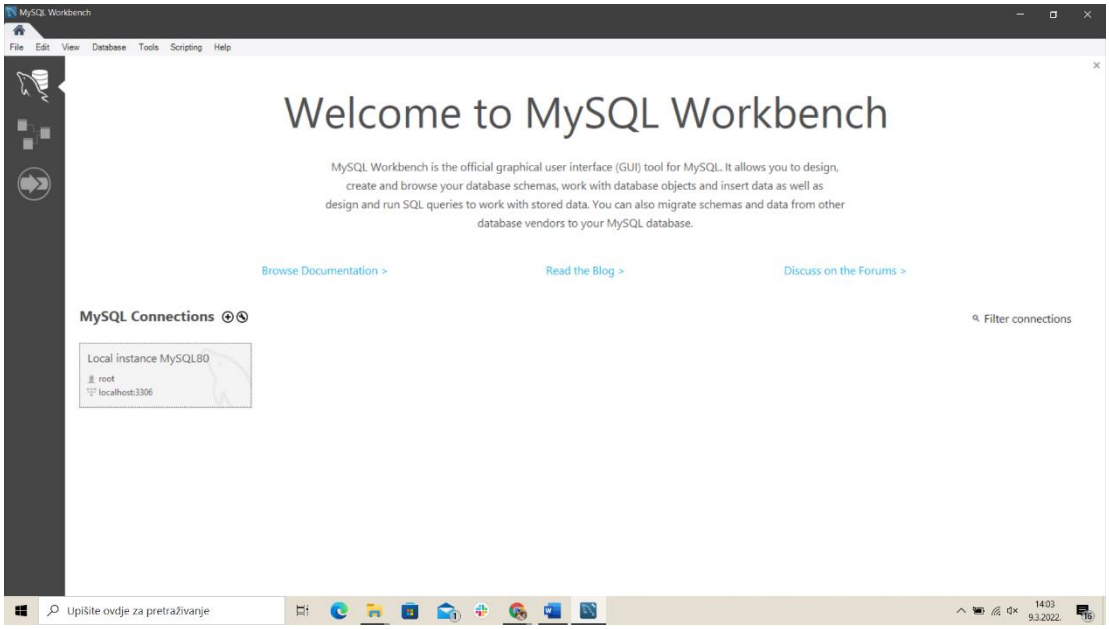
4. MySQL baze podataka

MySQL je najpopularniji i najrašireniji sustav otvorenog koda za upravljanje bazom podataka kojeg je izdala Švedska tvrtka MySQL AB 1995. godine. Jedan je od najpopularnijih sustava iz razloga što je kako i sama definicija kaže, otvorenog koda, što znači da ga svatko može koristiti, uz to dostupan je na većini platformi, te samim time njegovo preuzimanje i instalacija su vrlo jednostavni [8].

Radi kao pozadinski proces na sustavu, te kao većina baza podataka podržava strukturni upitni jezik (SQL, eng. Structured Query Language). SQL se može koristiti za stvaranje baze podataka i objekata, tada se koristi pod jezik DDL (eng. Data Definition Language) koji služi za izgradnju i modificiranje strukture i shema baze podataka, te DML (eng. Definition Manipulation Language) koji služi za manipulaciju nad podacima, a za to koristi SQL naredbe [8].

4.1. MySQL Workbench

MySQL workbench predstavlja grafički alat za dizajniranje baza podataka. Podržava MySQL verziju poslužitelja 5.6 i novije, stoga sve starije verzije je potrebno nadograditi, u suprotnom neće biti kompatibilne s MySQL workbench-om. U svom radu koristila sam najnoviju verziju, a to je 8.0.28 [9].



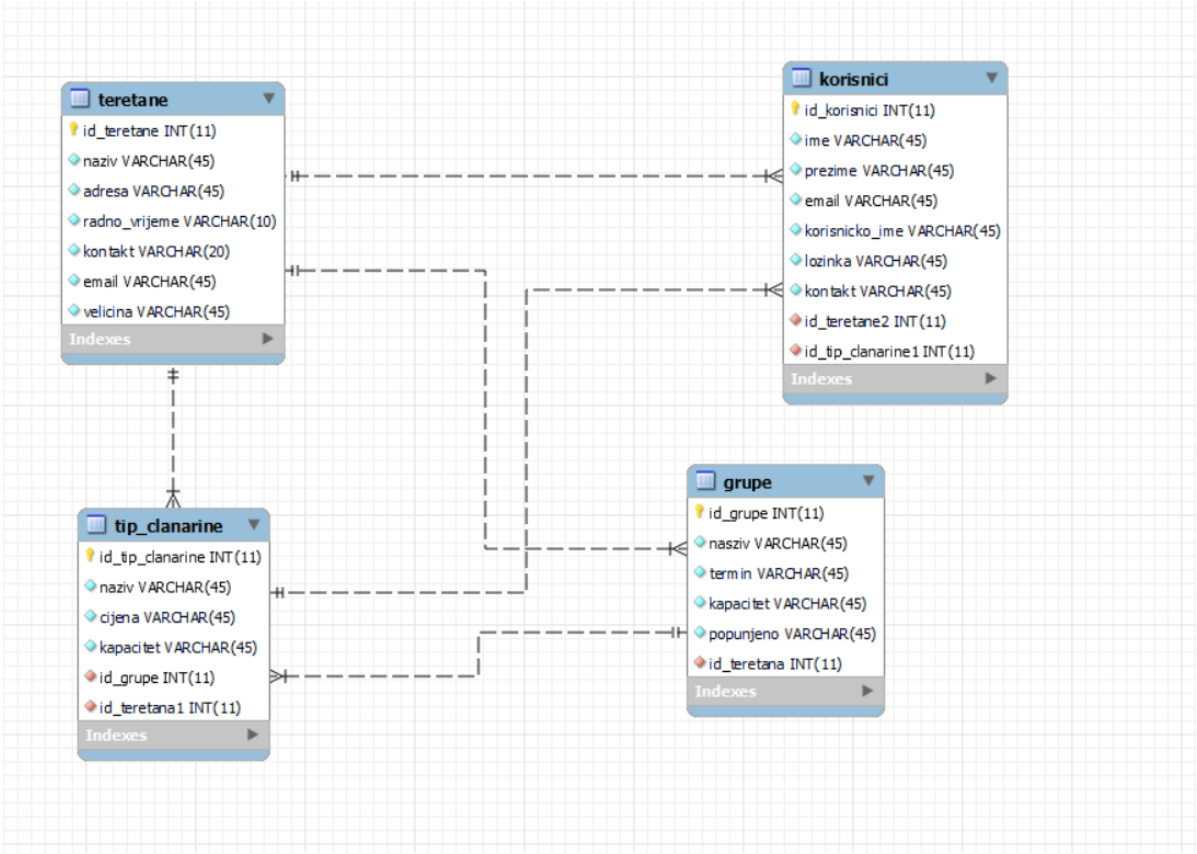
Slika 6: MySQL workbench sučelje (Izvor: vlastiti izvor)

Funkcionalnosti MySQL workbench-a:

- SQL razvoj: omogućuje stvaranje i upravljanje vezama s poslužiteljem baze podataka., te izvršavanje SQL upita nad vezama baze podataka.
- Modeliranje podataka: grafičko stvaranje modela sheme baze podataka, a uz pomoć uređivača tablica moguće vršiti promjene svih aspekata baze podataka
- Administracija poslužitelja: administracija instanci MySQL poslužitelja administriranjem korisnika, izvođenjem sigurnosnog kopiranja i oporavka, te praćenjem performansi MySQL poslužitelja
- Migracija podataka: migracija s Microsoft SQL servera, Microsoft Accessa, Sybase ASE, SQLite, SQL Anywhere, PostreSQL i drugih RDBMS tablica objekata i podataka na MySQL.

MySQL workbench je dostupan u dva izdanja: Community Edition i Commercial Edition, Community edition dostupan je besplatno, te sam taj koristila i u svome radu [9].

4.2. ERA dijagram



Slika 7: ERA dijagram baze podataka (Izvor: izrada autora)

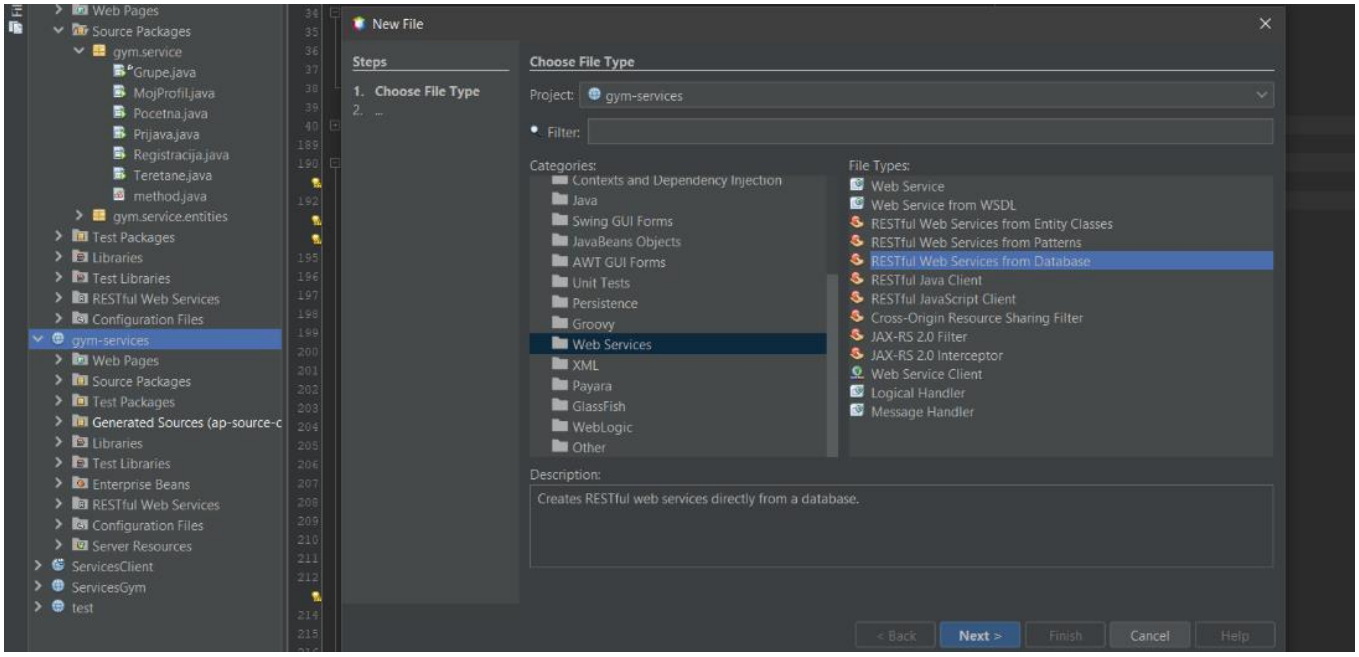
Na slici 7. se nalazi ERA dijagram baze podataka koja se koristi u aplikaciji. Baza se sastoji od četiri tablice. Tablica 'teretane' sadrži podatke o svim teretanama u gradu kao što su ime teretane, radno vrijeme adresa, veličina, email i kontakt. Tablica 'tip_clanarine' sadrži podatke o članarinama u svim teretanama i povezana je sa tablicom 'teretane' zato što jedna teretana može ponuditi više tipova članarina. Tablica 'grupe' sadrži podatke o grupnim treninzima kao što su naziv treninga, termin u kojem se održava, broj osoba koji može sudjelovati, popunjenost za određeni trening i teretanu u kojoj se određeni grupni trening održava, iz tog razloga je tablica 'grupe' povezana sa tablicom 'teretane' jer jedna teretana može ponuditi više grupnih treninga. Tablica 'korisnici' sadrži informacije o korisnicima koji se prijavljuju/registriraju u sustav, to su podaci vezani za ime i prezime korisnika, njegov email, korisničko ime i lozinku, kontakt broj, te informacijama o teretani u koju ide i koju vrstu članarine ima. Tablica 'korisnici' je povezana sa tablicama 'teretane' i 'tip_clanarine' iz razloga što u svaku teretanu može ići više korisnika i jednu vrstu članarine može imati više korisnika. Isto tako svaki korisnik može ići u jednu teretanu i imati jednu vrstu članarine.

5. Razvoj aplikacije “MyGym“

Za razvoj aplikacije sam koristila NetBeans razvojno okruženje za Javu. On omogućava razvoj aplikacija iz skupa modularnih softverskih komponenti koje se nazivaju moduli. NetBeans zahtijeva i korištenje Java Development Kit-a (JDK). JDK je razvojno okruženje za izradu aplikacija i komponenti pomoću Java programskog jezika, uključuje razne korisne alate za razvoj i testiranje programa.

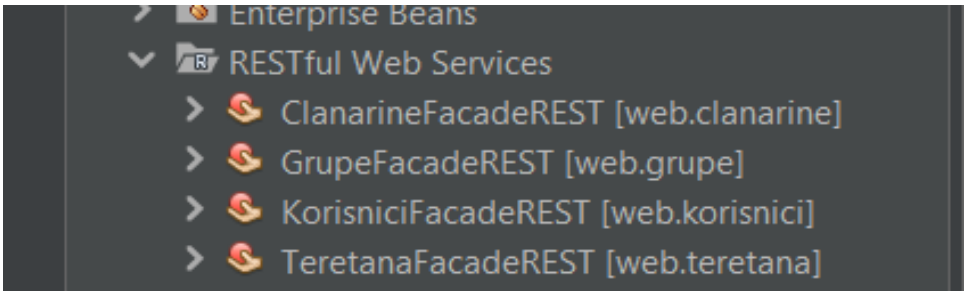
5.1. Stvaranje RESTful web servisa

U NetBeans razvojnom okruženju stvaramo novi projekt pod nazivom gym-services, to je ujedno i poslužiteljska aplikacija u kojoj će biti kreirani RESTful web servisi. Poslužitelj na kojem se aplikacija izvršava naziva se GlassFish server. U NetBeans-u postoji opcija stvaranja RESTful web usluga iz baze podataka (Slika 8).



Slika 8: Funkcija stvaranja RESTful web servisa iz baze podataka
(Vlastira izrada)

Prilikom kreiranja servisa potrebno je napraviti vezu sa bazom podataka pomoću MySQL (Connector/J driver). Nakon toga odabiremo koje tablice želimo koristiti i klikom na finish generiraju se klase entiteta i usluga za svaku odabranu tablicu iz baze podataka. U ovom slučaju odabrane su četiri tablice, a generirani RESTful webservisi su prikazani na sljedećoj slici (Slika 9):



Slika 9: Stvoreni RESTful web servisi iz baze podataka (Vlastita izrada)

5.1.1. RESTful web servis KorisniciFacadeREST

Prilikom kreiranja RESTful web servisa, NetBeans sam generira neke od metoda koje se mogu koristit, dok se mogu kreirati i proizvoljne metode koje želimo pozivati. Na slici (Slika 10) su prikazane metode koje je kreirao NetBeans. Kreirani servisi koriste metode GET, PUT, POST i DELETE. Metodama je moguće pristupiti putem URI-a http://localhost:8080/gym-services/webresources/web.korisnici.

```
@Stateless
@Path("web.korisnici")
public class KorisniciFacadeREST extends AbstractFacade<Korisnici> {

    @PersistenceContext(unitName = "gym-servicesPU")
    private EntityManager em;

    public KorisniciFacadeREST() {
        super(Korisnici.class);
    }

    @POST
    @Override
    @Consumes({MediaType.APPLICATION_JSON})
    public void create(Korisnici entity) {
        super.create(entity);
    }

    @PUT
    @Path("{id}")
    @Consumes({MediaType.APPLICATION_JSON})
    public void edit(@PathParam("id") Integer id, Korisnici entity) {
        super.edit(entity);
    }

    @DELETE
    @Path("{id}")
    public void remove(@PathParam("id") Integer id) {
        super.remove(super.find(id));
    }

    @GET
    @Path("{id}")
    @Produces({MediaType.APPLICATION_JSON})
    public Korisnici find(@PathParam("id") Integer id) {
        return super.find(id);
```

Slika 10: Metode u RESTful servisu Korisnici (Vlastita izrada)

Korištenje metode za pretraživanje korisnika po id-u je moguće na način da se putem HTTP GET metode na URI http://localhost:8080/gym-services/webresources/web.korisnici pošalje GET parametar id. To je metoda @GET @Path({id}) (Slika 10). GET metodi je potrebno proslijediti parametar id, kako bi dobili tražene podatke. Prilikom pozivanja metode, parametar "id" se zamjenjuje vrijednošću koja mu je zadana. Podaci koji se šalju resursu i rezultat koji se vraća klijentu navedeni su kao vrsta MIME medija u zaglavlju HTTP zahtjeva ili odgovora. Tu je korištena bilješka @Produces({MediaType_APPLICATION_JSON}) koja specificira da klasa proizvodi poruke koristeći JSON vrstu medija MIME [14].

Za dobivanje podataka za korisnika pod id-em "2" HTTTP zahtjevu koji koristi GET metodu prosljeđujemo parametar 2. http://localhost:8080/gym-services/webresources/web.korisnici/2 a rezultat RESTful web servisa je dobiven u JSON formatu i izgleda ovako (Slika 11):


```
{
    "email": "aanic@gmail.com",
    "idKorisnik": 2,
    "ime": "Anja",
    "kontakt": "0912357896",
    "korisnickoIme": "aanic",
    "lozinka": "0001",
    "prezime": "Anić",
    "teretana": "Studio For Life",
    "tipClanarine": "Bootybuilding"
},
```

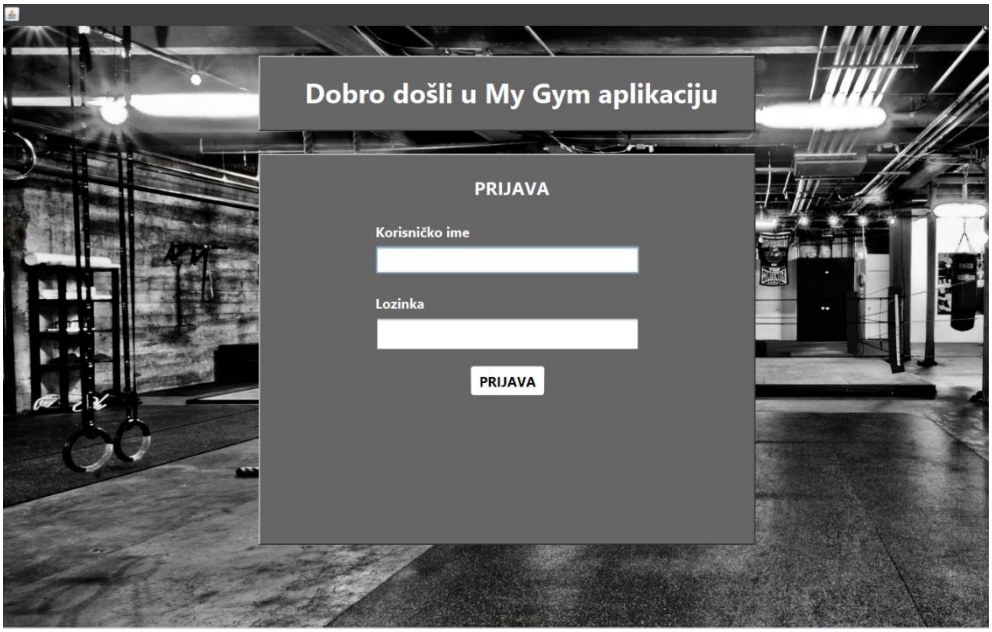

Slika 11: Rezultat RESTful web servisa za dobivanje podataka putem id-a (Vlastita izrada)

Za korištenje metode za prijavu korisnika potrebno je pristupiti URI-u: http://localhost:8080/gym-services/webresources/web.korisnici/prijava . U ovoj metodi se provjerava ispravnost unešenih podataka. Ako korisnik unese ispravne podatke vraća mu se rezultat "1", dok se kod krivo unešenih podataka vraća rezultat "0" (Slika 12).

```
@GET
@Path("prijava")
@Produces({MediaType.APPLICATION_XML, MediaType.APPLICATION_JSON})
  public String login(@QueryParam("korisnickoIme") String korisnickoIme,@QueryParam("lozinka") String lozinka){

    for(Korisnici k: findAll()){
        if(k.getKorisnickoIme().equals(korisnickoIme) && k.getLozinka().equals(lozinka)){
           return "1";
        } else {
            return "0";
        }
    }
    return null;
}
```

Slika 12: RESTful web servis sa metodom za prijavu (Vlastita izrada)

Slika 13 prikazuje formu za prijavu u aplikaciju. Uspješnom prijavom u aplikaciju se korisniku omogućava pregled svih teretani koje se nalaze u gradu Bjelovaru, zatim može pristupiti i popisu grupnih treninga kao i pregledu svih vrsta članarina za izraženom cijenom i ostalim potrebnim informacijama.



Slika 13: Forma za prijavu u aplikaciju (Vlastita izrada)

Forma za prijavu napravljena je na način da sadrži vezu na bazu podataka, tu je potrebno uvesti paket 'java.sql.Connection', zatim je potrebno napraviti vezu sa MySQL bazom podataka, koristeći port 3306, tablicu gym i root korisnika za povezivanje na REST API. Također, potrebno je inicijalizirati varijable Connection, PreparedStatement i ResultSet na null, te se spojiti na JDBC SQL pupravljački program ' com.mysql.cj.jdbc.Driver'. Nakon toga, provjeravaju se uneseni podaci u polja za unos koja se nalaze na formi. Ukoliko su podaci

ispravno uneseni, tada se korisnika šalje na početnu stranicu aplikacije (Slika 14), ukoliko podaci nisu ispravno uneseni, korisniku se pojavljuje poruka o unosu neispravnih podataka.

```
private void prijavaActionPerformed(java.awt.event.ActionEvent evt) {
    Connection veza = null;
       PreparedStatement prep = null;
       ResultSet rezultat = null;
   try {
       Class.forName("com.mysql.cj.jdbc.Driver");
       veza = DriverManager.getConnection("jdbc:mysql://localhost:3306/gym", "root", "");
       prep=veza.prepareStatement("SELECT * FROM korisnici WHERE korisnicko_ime=? AND lozinka=?");

       prep.setString(1, korIme.getText());
       prep.setString(2, lozinka.getText());

       rezultat=prep.executeQuery();

       if(rezultat.next()){
           Pocetna pocetna = new Pocetna();
           pocetna.setVisible(true);
           dispose();
       } else {
           JOptionPane.showMessageDialog(this, "Korisničko ime ili lozina nisu ispravni!");
       }

   } catch (ClassNotFoundException ex) {
       Logger.getLogger(Prijava.class.getName()).log(Level.SEVERE, null, ex);
   } catch (SQLException ex) {
       Logger.getLogger(Prijava.class.getName()).log(Level.SEVERE, null, ex);
   }

}
```

Slika 14: Kod forme za prijavu u aplikaciju (Vlastita izrada)

Za prikaz popisa svih teretani u JSON formatu kreirana je java klasa 'TeretanePopis' (Slika 15). Za dohvaćanje popisa svih teretani pozvali smo RESTful web servis koji se nalazi na URI-u: http://localhost:8080/gym-services/webresources/web.teretana. Potrebno je pozvati metodu openConnection() na gore navedeni URL koji vraća instancu HttpURLConnection. Zatim se postavlja metoda zahtjeva GET, te se poziva metoda setRequestMethod gdje postavljamo MIME tip podataka na JSON. Pomoću JSONObject-a prikazujemo podatke u JSON formatu (Slika 16).

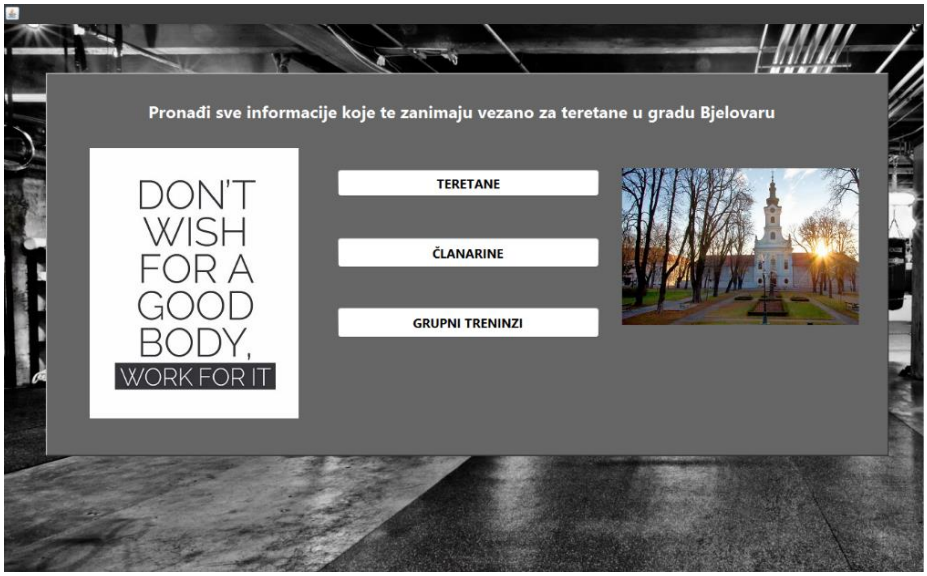
```
public class TeretanePopis {
    public static void main(String[] args) throws JSONException, IOException {
      try {
          String restService = "http://localhost:8080/gym-service/webresources/web.teretane";
          URL url = new URL(restService);
          HttpURLConnection conn = (HttpURLConnection) url.openConnection();
          conn.setRequestMethod("GET");
          conn.setRequestProperty("CONTENT-TYPE", "application/json");
          conn.setRequestProperty("ACCEPT", "application/json");

          if (conn.getResponseCode() == 200) {
              BufferedReader br = new BufferedReader(new InputStreamReader((conn.getInputStream())));

              String output;
              System.out.println("");

              while ((output = br.readLine()) != null) {
                  System.out.println(output);

                  JSONObject json = new JSONObject(output);
                  int id = json.getInt("id_teretana");
                  String naziv = json.getString("naziv");
                  String adresa = json.getString("adresa");
                  String radno_vrijeme = json.getString("radno_vrijeme");
                  String email = json.getString("email");
                  String velicina = json.getString("velicina");
                  String opis = json.getString("opis");

                  System.out.println(id); System.out.println(naziv); System.out.println(adresa); System.out.println(radno_vrijeme);
                  System.out.println(email); System.out.println(velicina); System.out.println(opis);
              }
              conn.disconnect();
          }
      } catch (MalformedURLException ex) {
          Logger.getLogger(TeretanePopis.class.getName()).log(Level.SEVERE, null, ex);
      } catch (ProtocolException ex) {
          Logger.getLogger(TeretanePopis.class.getName()).log(Level.SEVERE, null, ex);
      }
```

Slika 15: Rezultat u JSON formatu (Vlastita izrada)

{"adresa":"Ul. Ferde Rusana 10, 43 000 Bjelovar","email":"bra.fitnes@gmail.com","idTeretana":1,"kontakt":"099 471 5288","naziv":"Force fitness centar","opis":"Zbog čega se naš fitness centar razlikuje od ostalih? Upravo zbog visoke stručnosti
ikolovanih trenera sa Kineziološkog Fakulteta u Zagrebu, Fakulteta za sport i fizičko vaspitanje iz Beograda i Zdravstvenog Veleučilišta (studij fizioterapije) u Zagrebu, te zbog prisustva trenera na mnogobrojnim naučim i stručnim tečajevima,
dukacijama i skupovima i konstantnim usavršavanjem.","radnoVrijeme":"Ponedjeljak - Petak 09:00 - 22:00/Subota 10:00 - 17:00/Nedjelja-Zatvoreno","velicina":"320m2"},{"adresa":"Trg Stjepana Radića 24, 43000
jelovar","email":"myway.bjelovar@gmail.com","idTeretana":2,"kontakt":"091 927 9449","naziv":"My Way","opis":"Sportsko rekreativni centar MY WAY s radom je započeo u siječnju 2019. godine. Osnovali su ga supružnici Sokolović, Vera i
enato. U kratkom roku napredovali su do najvećeg i najopremljenijeg Sportsko rekreativnog centra u Bjelovaru uz vlastitu viziju, volju, upornost i učenje, ali i uz podršku najbližih, prijatelja, kumova, članova i obitelji Vuglač te stručnih suradnika,
ineziologa i licenciranih trenera.","radnoVrijeme":"Ponedjeljak - Petak 07:00 - 21:00 / Subota 08:00 - 14:00 / Nedjelja - Zatvoreno","velicina":"500m2"},{"adresa":"Ul. Ante Trumbića 4, 43 000
jelovar","email":"ffitnessfforllife@gmail.com","idTeretana":3,"kontakt":"095 897 9754","naziv":"Fitness For Life","opis":"Fitness centar koji je otvoren prije tri godine u ulici Ante Trumbića 4 s ciljem da pomogne što većem broju ljudi da poboljšaju
voje zdravlje i skinu suvišne kilograme, trenutno ima dvokratno radno vrijeme no dugoročni cilj im je da rade od 0-24 h. S dva trenera i trenericom, kao i ponudom najmodernijih tehnologija i sprava nastoje pružiti najbolju ponudu svojim
lijentima.","radnoVrijeme":"Ponedjeljak - Petak 08:00 - 12:00 i 15:00 - 22:00 / Subota 09:00 - 13:00 / Nedjelja - zatvoreno","velicina":"400m2"},{"adresa":"Ul. Antuna Mihanovića 21a, 43000 Bjelovar","email":"info@fitness-
tep.hr","idTeretana":4,"kontakt":"099 340 6711","naziv":"Fitness Step","opis":"Fitness centar Step osnovan je u listopadu 2016.godine. Vrlo brzo nakon osnivanja postaje jedan od vodećih fitness centara u Bjelovaru. Brojimo oko 1000 članova,od
ojih je 300-tinjak redovno aktivnih. Centar je opremljen novim i modernim spravama,dvoranom za pilates, aerobic,trx/girje,cardio dvoranom,te fitness dijelom koji se prostire na 500-tinjak m2. Uz učestalo nadopunjavanje sadržaja,pokušavamo
ašim članovima omogućiti najkvalitetnije vježbanje. Usmjereni smo na rekreaciju, rehabilitaciju, kondicijske treninge te grupne treninge.","radnoVrijeme":"Ponedjeljak - Petak 09:00 - 22:00 / Subota 10:00 - 21:00 / Nedjelja 17:00 -
1:00","velicina":"250m2"},{"adresa":"Ul. Petra Zrinskog 8a, 43 000 Bjelovar","email":"info@omegagym.com","idTeretana":5,"kontakt":"098 700 765","naziv":"Omega gym","opis":"Omega gym se nalazi u mjestu Bjelovar. Omega gym posluje u
ljedećim kategorijama Fitnes centri. U teretani imamo sve potrebno za vrhunski trening za sve tipove sportaša, od rekreativaca do onih koji teze najvišim rezultatimal","radnoVrijeme":"Ponedjeljak - Nedjelja 10:00 - 22:00","velicina":"180m2"},
"adresa":"Ul. Andrije Hebranga 29, 43 000 Bjelovar","email":"info@titangym.com","idTeretana":6,"kontakt":"098 161 9881","naziv":"Tritan gym","opis":"TITAN GYM Bjelovar – najopremljeniji fitness centar u Bjelovaru, s novim tehnologijama ISO
.ATARAL sprava i opreme Vasil, Hammer Strength, Precor, Woodway, Life fitness, Greenmaster – Momentum, Kodin, Vibro Gym i dr. Vježbanje s rekreativcima, fitness i bodybuilding, pripreme natjecatelja, fitness shop i savjetovanje o
upplementima i prehrani za rekreativce i sportaše. Trenirajte s najboljima – TRAIN WITH THE BEST. Fitness centar "Titangym" Bjelovar ujedno je i izložbeni salon fitness sprava i opreme Metalcraft-a, gdje projektiramo i proizvodimo pod ISO
ertifikatima KVALITETAN hrvatski brand fitness sprava.","radnoVrijeme":"Ponedjeljak - Petak 09:00 - 12:00 i 16:00 - 22:00 / Subota 10:00 - 13:00 i 16:00 - 19:00 / Nedjelja - zatvoreno","velicina":"120m2"}]

Slika 16: Prikaz popisa teretani u JSON formatu (Vlastita izrada)

Na početnoj stranici korisnik može odabrati pregled teretani, članarina ili grupnih treninga. Klikom na određeni gumb prikazuje se tablica sa svim podacima iz baze podataka.

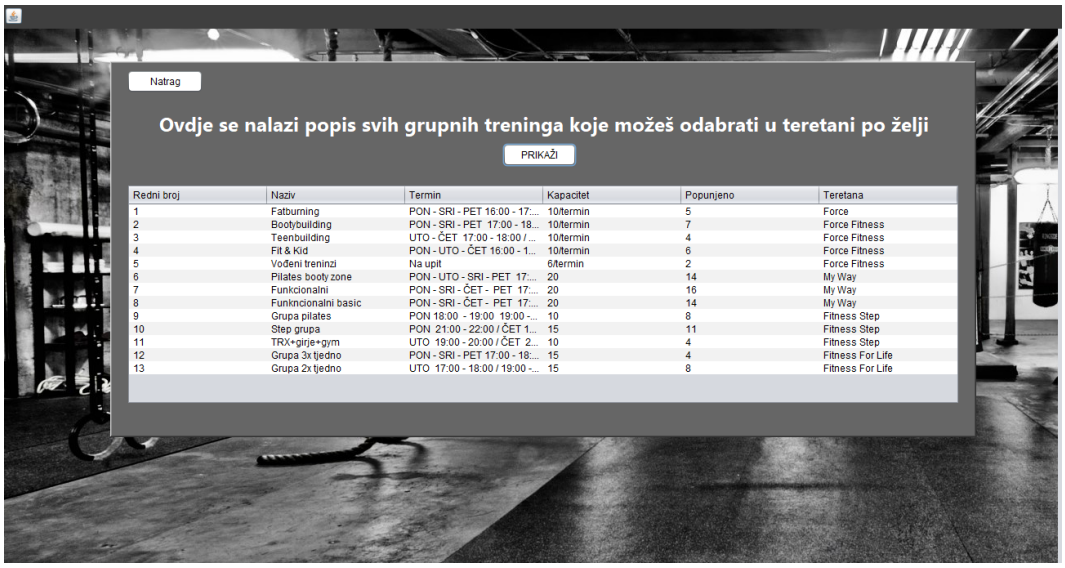


Slika 17: Početna stranica aplikacije (Vlastita izrada)

Slika 18 prikazuje kod za button 'PRIKAŽI' koji vraća podatke iz baze podataka u tablicu (Slika 19). Kao i u formi za prijavu, potrebno je napraviti vezu na bazu, inicijalizirati Connection, PreparedStatement i ResultSet na null. Zatim pomoću getString() funkcije dohvaćamo podatke iz baze pod određenim argumentom.

```
private void grupeActionPerformed(java.awt.event.ActionEvent evt) {
    try {
        Class.forName("com.mysql.cj.jdbc.Driver");
        Connection veza = DriverManager.getConnection("jdbc:mysql://localhost:3306/gym", "root", "");

        Statement stat = veza.createStatement();
        String upit = "SELECT * FROM grupe";
        ResultSet rezultat = stat.executeQuery(upit);

        while(rezultat.next()){
            String id = String.valueOf(rezultat.getInt("idgrupe"));
            String naziv = rezultat.getString("naziv");
            String termin = rezultat.getString("termin");
            String kapacitet = rezultat.getString("kapacitet");
            String popunjeno = rezultat.getString("popunjeno");
            String teretana = rezultat.getString("teretana");

            String grupeTable[] = {id,naziv,termin, kapacitet, popunjeno, teretana};
            DefaultTableModel table = (DefaultTableModel)grupa.getModel();
```

Slika 18: Prikaz podataka za grupne treninge iz baze podataka (Vlastita izrada)



Slika 19: Rezultat button-a za prikaz podataka iz baze podataka (Vlastita izrada)

Svakom od RESTful web servisa se može pristupiti putem URI-a. RESTful web servisu koji prikazuje sve tipove članarina u JSON formatu pristupamo putem URI-a: http://localhost:8080/gym-services/webresources/web.clanairine , također možemo mu proslijediti i id parametar za pretraživanje teretane po id-u. Putem URI-a: http://localhost:8080/gym-services/webresources/web.teretana dobijemo popis svih teretani u Bjelovaru u JSON formatu i također možemo pretraživati teretane po id-u. URI za prikaz i pretraživanje podataka po id-u za grupne treninge: http://localhost:8080/gym-services/webresources/web.grupe . NetBeans nudi mogućnost testiranja kreiranih RESTful web servisa u klijentskoj aplikaciji. Stoga se svaki od kreiranih servisa tako može testirati korištenjem gore navedenih URI-a i parametara.

6. Zaključak

Sve većom uporabom weba i sve većim i rasprostranjenijim pristupom web aplikacijama javila se i potreba za korištenjem web servisa. Web servisi se mogu gledati kao mehanizmi za izlaganje podataka preko HTTP protokola. Postoji mnoštvo javnih web servisa koji nude različite API-je za uzimanje određene vrste podataka. Prvi standard za objavljivanje i konzumiranje web usluga bio je SOAP, u radu je opisan SOAP protokol te uspoređen sa REST-om, na kojem se bazira ovaj rad. Usporedbom se može vidjeti kako REST jednostavniji od SOAP-a što i rezultira njegovom čestom upotrebom.

Kako bi podaci u aplikacijama bili razumljivi i onima koji ih koriste i onima koji rade s njima, potrebno je odrediti format za razmjenu podataka. Jedan od najčešće korištenih je JSON format. Obradom JSON podataka podrazumijevaju se operacije kao što su čitanje, pisanje, ispitivanje i modificiranje, te samim time ovaj format omogućava sve potrebne operacije za rad sa podacima.

Java je jedan od najkorištenijih programskih jezika u različitim područjima, koristit se za programiranje video igrica, a isto tako se i koristi za programiranje složenijih aplikacija. Pregledom raznih stranica, knjiga i tutoriala na internetu, ustanovljeno je kako se i za sam razvoj RESTful web servisa koristi Java programski jezik. Točnije, Java Enterprise Edition platforma koja je vrlo pouzdana i skalabilna platforma.

Tako su za potrebe ovog rada izrađene su dvije web aplikacije u razvojnom okruženju NetBeans IDE programirane u Java EE jeziku. Aplikacija gym-service predstavlja poslužiteljsku aplikaciju u kojoj su kreirani RESTful web servisi iz baze podataka. Dok gym-service-client predstavlja klijentsku aplikaciju koja radi sa podacima iz baze podataka.

7. Literatura

[1] Y. Fain, Programiranje Java. Zagreb: Dobar plan. 2015.

[2] R. T. Fielding, Architectural Styles and the Design of Network-based Software Architectures [Doktorska disertacija]. University of California, Irvine, 2000, Dostupno: https://www.ics.uci.edu/~fielding/pubs/dissertation/fielding_dissertation.pdf / [pristupano: 18.7.2021.]

[3] " RESTful Java Web Services" (septembar, 2015) J. Purushothaman [Na internetu]. Dostupno: https://github.com/eko55/bij/blob/master/RESTful%20Java%20Web%20Services%20-%20Second%20Edition%20%5BeBook%5D.pdf/ [pristupano: 19.7.2021.]

[4] "Introduction to RESTful Web services" (05.11.2008.) A. Rodriguez [Na internetu]. Dostupno: https://developer.ibm.com/articles/ws-restful/ [pristupano: 19.7.2021.]

[5] "What Are RESTful Web Services?" (bez dat.) Java Oracle [Na internetu]. Dostupno: https://docs.oracle.com/javaee/6/tutorial/doc/gijqy.html [pristupano: 20.07.2021.]

[6] "SOAP vs REST Web Services" (bez dat.) JavaTpoint [Na internetu]. Dostupno: https://www.javatpoint.com/soap-vs-rest-web-services [pristupano: 20.02.2022.]

[7] C. Kochmer i E. Frandsen, "JSP™ and XML, Integrating XML and Web Services in Your JSP Application", 2002. [Na internetu]. Dostupno: https://books.google.co.vi/books?id=dFc27ogrYg8C&printsec=copyright#v=onepage&q–&f=false/ [pristupano: 22.02.2022.]

[8] "What is MySQL? Everything You Need to Know" (bez dat.) Talend [Na internetu]. Dostupno: https://www.talend.com/resources/what-is-mysql/ [pristupano: 01.03.2022.]

[9] " Chapter 1 General Information" (bez dat.) MySQL™ [Na internetu]. Dostupno: https://dev.mysql.com/doc/workbench/en/wb-intro.html [pristupano: 01.03.2022.]

[10] "What is web application" (bez dat.) JavaTpoint [Na internetu]. Dostupno: https://www.javatpoint.com/web-application [pristupano: 04.04.2022.]

[11] "TIOBE Index for April 2022" (10.04.2022.) TIOBE [Na internetu]. Dostupno: https://www.tiobe.com/tiobe-index/ [pristupano: 12.04.2022.]

[12] "Understanding Java Platform, Enterprise Edition" (april, 2012.) Ian Evans [Na internetu] Dostupno: https://docs.oracle.com/javaee/6/firstcup/doc/gcrlo.html [pristupano: 12.04.2022.]

[13] "JAR File Overview" (bez dat.) Oracle Java Documentation [Na internetu]. Dostupno: https://docs.oracle.com/javase/8/docs/technotes/guides/jar/jarGuide.html [pristupano: 13.04.2022.]

[14] “ Fusion Middleware Developing RESTful Web Services for Oracle WebLogic Server“ (bez dat.) Oracle Help Center [Na internetu]. Dostupno: https://docs.oracle.com/cd/E24329_01/web.1211/e24983/develop.htm#RESTF113 [pristupano: 12.09.2022.]

Popis slika

Slika 1: Tijek web aplikacije (Prema: https://www.javatpoint.com/web-application) 3
Slika 2: Ilustracija SOA-e (Izvor: https://www.javatpoint.com/service-oriented-architecture)...6
Slika 3: Primjer glavnih HTTP metoda (Izvor: https://yalantis.com/blog/how-to-create-a-restful-api/).....10
Slika 4: Graf popularnosti programskih jezika za travanj 2022 (Izvor: https://www.tiobe.com/tiobe-index/).....14
Slika 5: Tablica popularnosti programskih jezika (Izvor: https://www.tiobe.com/tiobe-index/) 14
Slika 6: MySQL workbench sučelje (Izvor: vlastiti izvor) 18
Slika 7: ERA dijagram baze podataka (Izvor: izrada autora) 19
Slika 8: Funkcija stvaranja RESTful web servisa iz baze podataka (Vlastira izrada) 21
Slika 9: Stvoreni RESTful web servisi iz baze podataka (Vlastita izrada).....22
Slika 10: Metode u RESTful servisu Korisnici (Vlastita izrada).....22
Slika 11: Rezultat RESTful web servisa za dobivanje podataka putem id-a (Vlastita izrada).23
Slika 12: RESTful web servis sa metodom za prijavu (Vlastita izrada).....24
Slika 13: Forma za prijavu u aplikaciju (Vlastita izrada) 24
Slika 14: Kod forme za prijavu u aplikaciju (Vlastita izrada).....25
Slika 15: Rezultat u JSON formatu (Vlastita izrada) 25
Slika 16: Prikaz popisa teretani u JSON formatu (Vlastita izrada).....26
Slika 17: Početna stranica aplikacije (Vlastita izrada) 26
Slika 18: Prikaz podataka za grupne treninge iz baze podataka (Vlastita izrada) 26
Slika 19: Rezultat button-a za prikaz podataka iz baze podataka (Vlastita izrada).....27

Popis tablica

Tablica 1: Razlika između REST i SOAP servisa 5