Univerzitet u Beogradu

Fakultet organizacionih nauka

Katedra za elektronsko poslovanje

Laravel

Domaći zadatak 2

|  |  |  |
| --- | --- | --- |
| Ime | Prezime | Broj indeksa |
| Žarko | Colić | 253/17 |
| Link ka Github-u | https://github.com/torrent98/LaravelFinal/tree/master/fast-services | |

Sadržaj

[1 Korisnički zahtev 3](#_Toc134522629)

[2 Implementacija 4](#_Toc134522630)

[2.1 Model 4](#_Toc134522631)

[2.2 Migracija 8](#_Toc134522632)

[2.3 Kontroler 11](#_Toc134522633)

[2.4 Lista api ruta 25](#_Toc134522634)

[3 REST API 27](#_Toc134522635)

[4 Korisničko uputstvo 36](#_Toc134522636)

# Korisnički zahtev

Fast-Services API predstavlja API koji je izrađen korišćenjem PHP okvira koji se naziva Laravel. Svrha ove aplikacije jeste da pruži drugim aplikacijama (korisničkim, web aplikacijama itd.) servise i funkcionalnosti nad uslugama ***auto servisa***. Ovaj API sadrži rute koje nude kompletan CRUD za korisnike uključujući i registraciju i prijavu na sistem, kao i autentifikovan pristup različitim servisima API-a. Pored korisnika, podržane su i operacije nad servisima.

Slučajevi korišćenja:

1. *Registracija*
2. *Login*
3. *Prikaz korisnika*
4. *Izmena korisnika*
5. *Brisanje korisnika*
6. *Logout*
7. **Kreiranje Rating-a**
8. **Prikaz Rating-a**
9. **Izmena Rating-a**
10. **Brisanje Rating-a**
11. *Kreiranje mehanicara*
12. *Prikaz mehanicara*
13. *Izmena mehanicara*
14. *Brisanje mehanicara*
15. **Kreiranje usluge**
16. **Prikaz svih usluga**
17. **Izmena usluge**
18. **Brisanje usluge**

# Implementacija

## Model

Ovaj API sadrži 4 modela i to su: Rating, Mechanic, Service i User. Svi su smešteni u App->Models. User model je model koji se automatski kreira. Pristup rutama u okviru API-a je zasnovan na autentifikaciji korisnika. Atributi User modela su: name, email, role i password. Sama autentifikacija je zasnovana na mejlu i šifri korisnika. Ispod je dat izgled klase **User**.

class User extends Authenticatable

{

    use HasApiTokens, HasFactory, Notifiable;

    /\*\*

     \* The attributes that are mass assignable.

     \*

     \* @var array<int, string>

     \*/

    protected $fillable = [

        'name',

        'email',

        'password',

        'role'

    ];

    /\*\*

     \* The attributes that should be hidden for serialization.

     \*

     \* @var array<int, string>

     \*/

    protected $hidden = [

        'password',

        'remember\_token',

    ];

Polje ***fillable***ukazuje na atribute modela User koje je moguće masovno dodeljivati kroz API dok polje *hidden* ukazuje na atribute koji se ne serijalizuju prilikom serijalizacije User modela. Na primer prilikom vraćanja User-a ovi atribute se ne vraćaju u okviru JSON fajla.

 public function rating() {

        return $this->hasMany(Rating::class);

    }

Funkcija ***rating****()* koja se zapravo koristi i kojoj se pristupa kao polje ***Rating***ukazuje na to da svaki korisnik može imati više rating-a i ovo polje nam omogućava da vratimo sve ocene datog korisnika.

public function isAdmin() {

        return $this->role === 'admin';

    }

    public function isUser() {

        return $this->role === 'user';

    }

U okviru ovog modela takođe imamo funkcije za proveru uloge korisnika tj da li je korisnik admin ili user.

Naredni model **je Rating** model.

protected $fillable = [

        'date\_and\_time',

        'user',

        'service',

        'rating',

        'note',

        'mechanic'

    ];

Svaki Rating sadrži datum i vreme, id usluge, id mehanicara, id korisnika koji je uneo ocenu, belešku/komentar i samu ocenu. Sva ova polja su protected kako ne bi bila javno vidljiva svim korisnicima, već samo adminu.

   use HasFactory;

    protected $table = 'ratings';

    public $primaryKey = 'id';

Atribut table skladišti sve ocene u jedan niz/tabelu. Primarni ključ ocena je njihov id.

public function userkey() {

        return $this->belongsTo(User::class, 'user');

    }

    public function servicekey() {

        return $this->belongsTo(Service::class, 'service');

    }

    public function mechanickey() {

        return $this->belongsTo(Mechanic::class, 'mechanic');

    }

*Rating* je povezan sa 3 druga modela (*User, Service, Mechanic*), i sa svakim od tih modela fomira veza kardinalnosti 1-M. Pored toga ovaj model sadrži i spoljni ključ ka User modelu, kako bismo u svakom trenutku znali koju ocenu je dao koji korisnik. Takođe imamo i spoljne ključeve prema provajderu i usluzi.

Sledeći model je **Mechanic**.

use HasFactory;

    protected $table = 'mechanics';

    public $primaryKey = 'id';

    public function rating() {

        return $this->hasMany(Rating::class);

    }

    protected $fillable = [

        'name',

        'phone\_number',

        'years\_of\_experience',

        'email'

    ];

Kreirana table promenljiva za skladištenje i primarni ključ, kao i atributi: ime, email, broj telefona, godine iskustva.

Funkcija *hasMany()* nam omogućava da jedan pružalac usluge može da ima više ocena I da ta ocena ima spoljni ključ po mechanic\_id.

Četvrti model je **Service**.

class Service extends Model

{

    use HasFactory;

    protected $table = 'services';

    public $primaryKey = 'id';

    public function rating() {

        return $this->hasMany(Rating::class);

    }

    protected $fillable = [

        'name',

    ];

}

On takođe ima svoj table atribut, primarni ključ, spoljni ključ ka ocenama i svoj atribut za ime usluge.

## Migracija

Pored migracija koje se po default-u kreiraju prilikom kreiranja Laravel aplikacije u okviru ove aplikacije kreirane su i dodatne migracije koje su navedene dalje u tekstu.

Važno je naglasiti da svaka migracija podrazumeva *up()* i *down()* funkciju. *Up()* funkcija koja se aktivira prilikom pokretanja same migracije, dok *down()* funkcija treba da proizvede inverzan efekat efektu *up()* funkcije i pokreće se prilikom rollback-a migracije.

*2023\_05\_05\_193349\_create\_mechanics\_table*

Migracija koja kreira tabelu Mehanicara – Mechanic-a koja odgovara gore navedenom Mechanic modelu. Navedeni su nazivi i ograničenja svih potrebnih kolona Mechanic tabele u bazi.

public function up(): void

    {

        Schema::create('mechanics', function (Blueprint $table) {

            $table->id();

            $table->string('name');

            $table->string('phone\_number')->uniqe();

            $table->integer('years\_of\_experience');

            $table->string('email');

        });

    }

    /\*\*

     \* Reverse the migrations.

     \*/

    public function down(): void

    {

        Schema::dropIfExists('mechanics');

    }

*2023\_05\_05\_193938\_create\_ratings\_table*

public function up(): void

    {

        Schema::create('ratings', function (Blueprint $table) {

            $table->id();

            $table->timestamp('date\_and\_time');

            $table->foreignId('user');

            $table->foreignId('service');

            $table->integer('rating');

            $table->text('note')->nullable();

        });

    }

    /\*\*

     \* Reverse the migrations.

     \*/

    public function down(): void

    {

        Schema::dropIfExists('ratings');

    }

*2023\_01\_18\_195716\_create\_services\_table*

Migracija koja kreira tabelu Services koja odgovara gore navedenom Services modelu. Navedeni su nazivi i ograničenja svih potrebnih kolona Services tabele u bazi.

public function up()

    {

        Schema::create('services', function (Blueprint $table) {

            $table->id();

            $table->string('name');

        });

    }

    public function down()

    {

        Schema::dropIfExists('services');

    }

*2023\_01\_18\_195142\_add\_column\_mechanic\_to \_ratings\_table*

Migracija koja tabeli Ratings koja prvobitno nije imala spoljni ključ ka Mechanic tabeli, dodaje upravo taj spoljni ključ (u vidu nove kolone).

public function up(): void

    {

        Schema::table('ratings', function (Blueprint $table) {

            $table->foreignId('mechanic');

        });

    }

    /\*\*

     \* Reverse the migrations.

     \*/

    public function down(): void

    {

        Schema::table('ratings', function (Blueprint $table) {

            $table->dropForeign('mechanic');

        });

    }

*2023\_01\_20\_092058\_add\_column\_role\_to\_users\_table*

Migracija koja tabeli Users dodaje kolonu za role koja prethodno nije napravljena.

public function up(): void

    {

        Schema::table('users', function (Blueprint $table) {

            $table->enum('role', ['user', 'admin'])->default('user');

        });

    }

    /\*\*

     \* Reverse the migrations.

     \*/

    public function down(): void

    {

        Schema::table('users', function (Blueprint $table) {

            $table->dropColumn('role');

        });

    }

## Kontroler

U okviru ove aplikacije kreirano je osam kontrolera zaduženih za ispunjenje zahteva korisnika nad User, Rating, Service, Mechanic modelom. Pored ovih osnovnih kontrolera dodati su i kontroleri: *UserRatingController, ServiceRatingrController, MechanicRatingController* omogućavaju da se prikažu ocene po izabranim tabelama. Takođe, kreiran je i AuthController u okviru API foldera koji nam služi za funkcionisanje reigstracije, login i logout rute.

*UserController*

class UserController extends Controller

{

    /\*\*

     \* Display a listing of the resource.

     \*/

    public function index()

    {

        return new UserCollection(User::all());

    }

    /\*\*

     \* Show the form for creating a new resource.

     \*/

    public function create()

    {

        //

    }

    /\*\*

     \* Store a newly created resource in storage.

     \*/

    public function store(Request $request)

    {

        //

    }

    /\*\*

     \* Display the specified resource.

     \*/

    public function show(User $user)

    {

        return new UserResource($user);

    }

    /\*\*

     \* Show the form for editing the specified resource.

     \*/

    public function edit(User $user)

    {

        //

    }

    /\*\*

     \* Update the specified resource in storage.

     \* @param  int  $id

     \* @return \Illuminate\Http\Response

     \*/

    public function update(Request $request, User $user)

    {

        if(auth()->user()->id == $user->id || auth()->user()->isAdmin()) {

            $validator = Validator::make($request->all(), [

                'name' => 'required|string|max:100',

                'email' => 'required|max:50|email|unique:users,email,'.$user->id,

                'password' => 'required|string|regex:"^(?=.\*[A-Za-z])(?=.\*\d)(?=.\*[@$!%\*#?&])[A-Za-z\d@$!%\*#?&]{8,}$"'

            ]);

            if ($validator->fails())

                return response()->json($validator->errors());

            $user->email\_verified\_at = now();

            $user->name =  $request->name;

            $user->email = $request->email;

            $user->password = Hash::make($request->password);

            $user->save();

            return response()->json(['user' => $user, 'access\_token' => $user->remember\_token, 'token\_type' => 'Bearer']);

        }

        return response()->json('You are not authorized to update someone');

    }

    /\*\*

     \* Remove the specified resource from storage.

     \* @param  int  $id

     \* @return \Illuminate\Http\Response

     \*/

    public function destroy(User $user)

    {

        if($user->id != auth()->user()->id && auth()->user()->isAdmin()) {

            $user->delete();

            return response()->json('User is deleted successfully.');

        }

        return response()->json('You do not have the privilege to delete user if you are not admin, nor to delete your own account.');

    }

*ServiceController*

class ServiceController extends Controller

{

    /\*\*

     \* Display a listing of the resource.

     \*/

    public function index()

    {

        return new ServiceCollcetion(Service::all());

    }

    /\*\*

     \* Show the form for creating a new resource.

     \*/

    public function create()

    {

        //

    }

    /\*\*

     \* Store a newly created resource in storage.

     \*/

    public function store(Request $request)

    {

        $validator = Validator::make($request->all(), [

            'name' => 'required|string|max:150|unique:services',

        ]);

        if ($validator->fails())

            return response()->json($validator->errors());

        if(auth()->user()->isUser())

            return response()->json('You are not authorized to create new services.');

        $service = Service::create([

            'name' => $request->name,

        ]);

        return response()->json(['Service is created successfully.', new ServiceResource($service)]);

    }

    /\*\*

     \* Display the specified resource.

     \*/

    public function show(Service $service)

    {

        return new ServiceResource($service);

    }

    /\*\*

     \* Show the form for editing the specified resource.

     \*/

    public function edit(Service $service)

    {

        //

    }

    /\*\*

     \* Update the specified resource in storage.

     \* @param  \Illuminate\Http\Request  $request

     \* @param  int  $id

     \* @return \Illuminate\Http\Response

     \*/

    public function update(Request $request, $id)

    {

        $validator = Validator::make($request->all(), [

            'name' => 'required|string|max:150|unique:services,name,' .$id,

        ]);

        if ($validator->fails())

            return response()->json($validator->errors());

        if(auth()->user()->isUser())

            return response()->json('You are not authorized to update services.');

        $service->name = $request->name;

        $service->save();

        return response()->json(['Service is updated successfully.', new ServiceResource($service)]);

    }

    /\*\*

     \* Remove the specified resource from storage.

     \* @param  int  $id

     \* @return \Illuminate\Http\Response

     \*/

    public function destroy(Service $service)

    {

        if(auth()->user()->isUser())

        return response()->json('You are not authorized to delete services.');

        $rating = Rating::get()->where('service', $service->id);

        if (count($rating) > 0)

            return response()->json('You cannot delete services that have appointment ratings.');

        $service->delete();

        return response()->json('Service is deleted successfully.');

    }

}

*MechanicController*

class MechanicController extends Controller

{

    /\*\*

     \* Display a listing of the resource.

     \*/

    public function index()

    {

        $mechanics = Mechanic::all();

        return new MechanicCollection($mechanics);

    }

    /\*\*

     \* Show the form for creating a new resource.

     \*/

    public function create()

    {

        //

    }

    /\*\*

     \* Store a newly created resource in storage.

     \*/

    public function store(Request $request)

    {

        $validator = Validator::make($request->all(), [

            'name' => 'required|string|max:150',

            'phone\_number' => 'required|string|max:150|unique:mechanics',

            'years\_of\_experience' => 'required|numeric|lte:30|gte:1',

            'email' => 'required|email|unique:mechanics',

        ]);

        if ($validator->fails())

            return response()->json($validator->errors());

        if(auth()->user()->isUser())

            return response()->json('You are not authorized to create new providers.');

        $mechanics = Mechanic::create([

            'name' => $request->name,

            'phone\_number' => $request->phone\_number,

            'years\_of\_experience' => $request->years\_of\_experience,

            'email' => $request->email,

        ]);

        return response()->json(['Mechanic is created successfully.', new MechanicResource($mechanics)]);

    }

    /\*\*

     \* Display the specified resource.

     \*/

    public function show(Mechanic $mechanic)

    {

        return new MechanicResource($mechanic);

    }

    /\*\*

     \* Show the form for editing the specified resource.

     \*/

    public function edit(Mechanic $mechanic)

    {

        //

    }

    /\*\*

     \* Update the specified resource in storage.

     \* @param  int  $id

     \* @return \Illuminate\Http\Response

     \*/

    public function update(Request $request, Mechanic $mechanics, $id)

    {

        $validator = Validator::make($request->all(), [

            'name' => 'required|string|max:150',

            'phone\_number' => 'required|string|max:150|unique:mechanics,phone\_number,'.$mechanics->id,

            'years\_of\_experience' => 'required|numeric|lte:30|gte:1',

            'email' => 'required|email|unique:mechanics,email,'.$mechanics->id,

        ]);

        if ($validator->fails())

            return response()->json($validator->errors());

        if(auth()->user()->isUser())

            return response()->json('You are not authorized to update mechanics.');

        $mechanics->name = $request->name;

        $mechanics->phone\_number = $request->phone\_number;

        $mechanics->years\_of\_experience = $request->years\_of\_experience;

        $mechanics->email = $request->email;

        $mechanics->save();

        return response()->json(['Mechanic is updated successfully.', new MechanicResource($mechanics)]);

    }

    /\*\*

     \* Remove the specified resource from storage.

     \* @param  int  $id

     \* @return \Illuminate\Http\Response

     \*/

    public function destroy(Mechanic $mechanic)

    {

        if(auth()->user()->isUser())

            return response()->json('You are not authorized to delete mechanics.');

        $rating = Rating::get()->where('mechanic', $mechanic->id);

        if (count($rating) > 0)

            return response()->json('You cannot delete mechanics that have ratings.');

        $mechanic->delete();

        return response()->json('Mechanic is deleted successfully.');

    }

}

*RatingController*

class RatingController extends Controller

{

    /\*\*

     \* Display a listing of the resource.

     \*/

    public function index()

    {

        return new RatingCollection(Rating::all());

    }

    /\*\*

     \* Show the form for creating a new resource.

     \*/

    public function create()

    {

        //

    }

    /\*\*

     \* Store a newly created resource in storage.

     \*/

    public function store(Request $request)

    {

        $validator = Validator::make($request->all(), [

            'date\_and\_time' => 'required|date',

            'service' => 'required|numeric|gte:1|lte:5',

            'rating' => 'required|numeric|lte:5|gte:1',

            'note' => 'required|string|min:20',

            'mechanic' => 'required|numeric|gte:1|lte:10',

        ]);

        if ($validator->fails())

            return response()->json($validator->errors());

        if(auth()->user()->isAdmin())

            return response()->json('You are not authorized to create new ratings.');

        $rating = Rating::create([

            'date\_and\_time' => $request->date\_and\_time,

            'user' => auth()->user()->id,

            'service' => $request->service,

            'rating' => $request->rating,

            'note' => $request->note,

            'mechanic' => $request->mechanic,

        ]);

        return response()->json(['Rating is created successfully.', new RatingResource($rating)]);

    }

    /\*\*

     \* Display the specified resource.

     \*/

    public function show(Rating $rating)

    {

        return new RatingResource($rating);

    }

    /\*\*

     \* Show the form for editing the specified resource.

     \*/

    public function edit(Rating $rating)

    {

        //

    }

    /\*\*

     \* Update the specified resource in storage.

     \* @param  int  $id

     \* @return \Illuminate\Http\Response

     \*/

    public function update(Request $request, Rating $rating)

    {

        $validator = Validator::make($request->all(), [

            'date\_and\_time' => 'required|date',

            'user' => 'required|numeric|digits\_between:1,5',

            'service' => 'required|numeric|gte:1|lte:5',

            'rating' => 'required|numeric|lte:5|gte:1',

            'note' => 'required|string|min:20',

            'mechanic' => 'required|numeric|gte:1|lte:10',

        ]);

        if ($validator->fails())

            return response()->json($validator->errors());

        if(auth()->user()->isAdmin())

            return response()->json('You are not authorized to update ratings.');

        if(auth()->user()->id != $rating->user)

            return response()->json('You are not authorized to update someone elses ratings.');

        $rating->date\_and\_time = $request->date\_and\_time;

        $rating->user = auth()->user()->id;

        $rating->service = $request->service;

        $rating->rating = $request->rating;

        $rating->note = $request->note;

        $rating->mechanic = $request->mechanic;

        $rating->save();

        return response()->json(['Appointment rating is updated successfully.', new RatingResource($rating)]);

    }

    /\*\*

     \* Remove the specified resource from storage.

     \*/

    public function destroy(Rating $rating)

    {

        if(auth()->user()->isAdmin())

            return response()->json('You are not authorized to delete ratings.');

        if(auth()->user()->id != $rating->user)

            return response()->json('You are not authorized to delete someone elses ratings.');

        $rating->delete();

        return response()->json('Rating is deleted successfully.');

    }

}

*MechanicRatingController*

class MechanicRatingController extends Controller

{

    public function index($mechanic\_id)

    {

        $rating = Rating::get()->where('mechanic', $mechanic\_id);

        if (count($rating) == 0)

            return response()->json('Data not found', 404);

        return new RatingCollection($rating);

    }

}

*ServiceRatingController*

class ServiceRatingController extends Controller

{

    public function index($service\_id)

    {

        $rating = Rating::get()->where('service', $service\_id);

        if (count($rating) == 0)

            return response()->json('Data not found', 404);

        return new RatingCollection($apprat);

    }

}

*UserRatingController*

class UserRatingController extends Controller

{

    public function index($user\_id)

    {

        $rating = Rating::get()->where('user', $user\_id);

        if (count($rating) == 0)

            return response()->json('Data not found', 404);

        return new RatingCollection($rating);

    }

    public function myrating()

    {

        if(auth()->user()->isAdmin())

            return response()->json('You are not allowed to have ratings.');

        $rating = Rating::get()->where('user', auth()->user()->id);

        if (count($rating) == 0)

            return response()->json('Data not found', 404);

        return new RatingCollection($rating);

    }

}

## Lista api ruta

Na ovoj slici možemo videti sve rute koje ima naša aplikacija:

Route::group(['middleware' => ['auth:sanctum']], function () {

    //admin

    Route::resource('services', ServiceController::class)->only(['store', 'update', 'destroy']); //radi store, PUT cudan, delete radi

    Route::resource('mechanics', MechanicController::class)->only(['store', 'update', 'destroy']);  //radi store, PUT cudan, delete radi

    Route::resource('users', UserController::class)->only(['destroy']);  //radi

    Route::post('/register', [AuthController::class, 'register']); //radi

    Route::resource('ratings', RatingController::class)->only(['store', 'update', 'destroy']); //radi store, PUT cudan, delete radi

    Route::post('/logout', [AuthController::class, 'logout']);  //radi

    //ulogovan KORISNIK

    Route::get('/myrating', [UserRatingController::class, 'myrating']); //

    Route::resource('users', UserController::class)->only(['update']);  //PUT radi cudno

});

Route::post('/login', [AuthController::class, 'login']); //radi

//javne

//servis

Route::resource('services', ServiceController::class)->only(['index', 'show']); //radi

//mehanicari

Route::resource('mechanics', MechanicController::class)->only(['index', 'show']); //radi

//ocene

Route::resource('ratings', RatingController::class)->only(['index', 'show']); //radi

//ocene po korisniku, mehanicaru i servisu

Route::get('/users/{id}/ratings', [UserRatingController::class, 'index']); //radi

Route::get('/mechanics/{id}/ratings', [MechanicRatingController::class, 'index']); //radi

Route::get('/services/{id}/ratings', [ServiceRatingController::class, 'index']); //radi

Route::resource('users', UserController::class)->only(['index', 'show']); //radi

# REST API

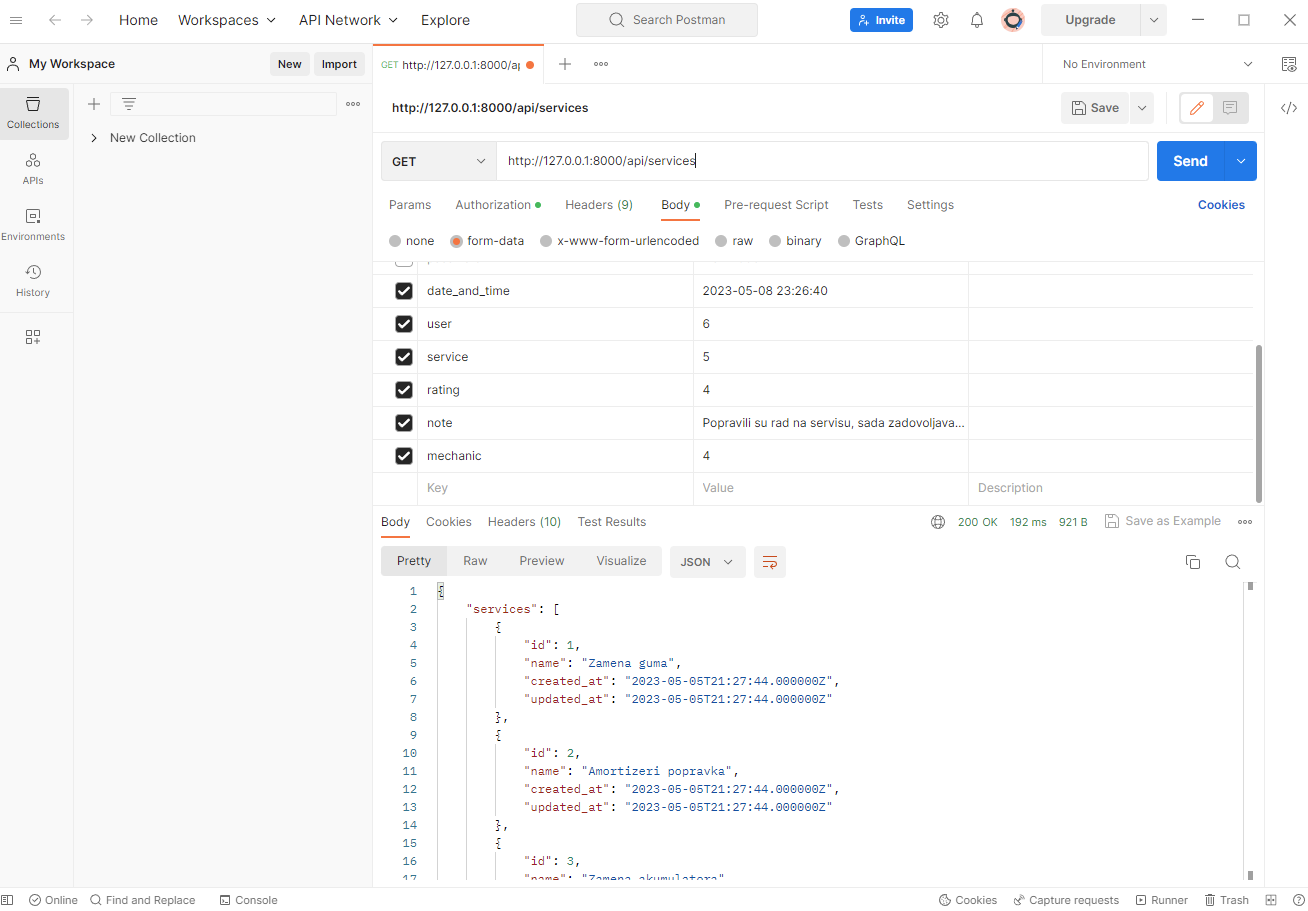
Kreirati API dokumentaciju.

|  |  |
| --- | --- |
| Opis funkcije | Prikaz svih korisnika |
| HTTP metoda | GET |
| URL | http://localhost:8000/api/users |
| URL parametri | (nema) |
| HTTP body parametri | (nema) |
| Format HTTP body parametara | Application/json |
| Izlazni parametri | {      "users": [          {              "name": "Zarko1",              "email": "zarko1@gmail.com",              "role": "admin"          },          {              "name": "Srdjan",              "email": "srdjan03@gmail.com",              "role": "user"          },          {              "name": "Milan",              "email": "komsa@gmail.com",              "role": "user"          },          {              "name": "Damjan",              "email": "damjan@gmail.com",              "role": "user"          }      ]  } |
| Format izlaznih parametara | application/json |
| Opis funkcije | Prikaz određenog korisnika iz tabele Users |
| HTTP metoda | GET |
| URL | http://localhost:8000/api/users/{id} |
| URL parametri | $user\_id=id |
| HTTP body parametri | (nema) |
| Format HTTP body parametara | Application/json |
| Izlazni parametri | {      "user": {          "name": "Milan",          "email": "komsa@gmail.com",          "role": "user"      }  } |
| Format izlaznih parametara | application/json |
| Opis funkcije | Prikaz svih ocena |
| HTTP metoda | GET |
| URL | http://localhost:8000/api/ratings |
| URL parametri | (nema) |
| HTTP body parametri | (nema) |
| Format HTTP body parametara | Application/json |
| Izlazni parametri | {      "ratings": [          {              "date\_and\_time": "2023-05-05 21:27:44",              "user": {                  "name": "Srdjan",                  "email": "srdjan03@gmail.com",                  "role": "user"              },              "service": {                  "name": "Zamena guma"              },              "rating": 5,              "note": "Odlicna zamena, vrlo efikasno!",              "mechanic": {                  "name": "Izaiah Fritsch",                  "phone\_number": "+1 (916) 224-7339",                  "years\_of\_experience": 15,                  "email": "kstiedemann@example.net"              }          },          {              "date\_and\_time": "2023-05-08 23:26:36",              "user": {                  "name": "Zarko1",                  "email": "zarko1@gmail.com",                  "role": "admin"              },              "service": {                  "name": "Zamena guma"              },              "rating": 5,              "note": "Servo zamena po prihvatljivoj ceni uz dosta profesionalizma.",              "mechanic": {                  "name": "Mr. Douglas Kuhlman DDS",                  "phone\_number": "571.994.2099",                  "years\_of\_experience": 22,                  "email": "arne.gerlach@example.net"              }          }      ]  } |
| Format izlaznih parametara | application/json |
| Opis funkcije | Prikaz određene ocene iz tabele Ratings |
| HTTP metoda | GET |
| URL | http://localhost:8000/api/ratings/{id} |
| URL parametri | $ rating\_id=id |
| HTTP body parametri | (nema) |
| Format HTTP body parametara | Application/json |
| Izlazni parametri | {      "date\_and\_time": "2023-05-05 21:27:44",      "user": {          "name": "Srdjan",          "email": "srdjan03@gmail.com",          "role": "user"      },      "service": {          "name": "Zamena guma"      },      "rating": 5,      "note": "Odlicna zamena, vrlo efikasno!",      "mechanic": {          "name": "Izaiah Fritsch",          "phone\_number": "+1 (916) 224-7339",          "years\_of\_experience": 15,          "email": "kstiedemann@example.net"      }  } |
| Format izlaznih parametara | application/json |
| Opis funkcije | Prikaz svih mehanicara |
| HTTP metoda | GET |
| URL | http://localhost:8000/api/mechanics |
| URL parametri | (nema) |
| HTTP body parametri | (nema) |
| Format HTTP body parametara | Application/json |
| Izlazni parametri | {      "mechanics": [          {              "name": "Izaiah Fritsch",              "phone\_number": "+1 (916) 224-7339",              "years\_of\_experience": 15,              "email": "kstiedemann@example.net"          },          {              "name": "Mr. Douglas Kuhlman DDS",              "phone\_number": "571.994.2099",              "years\_of\_experience": 22,              "email": "arne.gerlach@example.net"          },          {              "name": "Terrance Aufderhar",              "phone\_number": "667-562-5135",              "years\_of\_experience": 6,              "email": "hudson.paolo@example.net"          },          {              "name": "Dagmar Collins",              "phone\_number": "747.349.5690",              "years\_of\_experience": 25,              "email": "tleuschke@example.com"          },          {              "name": "Maggie Von",              "phone\_number": "(817) 904-9848",              "years\_of\_experience": 10,              "email": "durgan.fausto@example.net"          }      ]  } |
| Format izlaznih parametara | application/json |
| Opis funkcije | Prikaz određenog mehanicara iz tabele Mechanics |
| HTTP metoda | GET |
| URL | http://localhost:8000/api/mechanics/{id} |
| URL parametri | $mechanic\_id=id |
| HTTP body parametri | (nema) |
| Format HTTP body parametara | Application/json |
| Izlazni parametri | {      "mechanic": {          "name": "Maggie Von",          "phone\_number": "(817) 904-9848",          "years\_of\_experience": 10,          "email": "durgan.fausto@example.net"      }  } |
| Format izlaznih parametara | application/json |
| Opis funkcije | Prikaz svih usluga |
| HTTP metoda | GET |
| URL | http://localhost:8000/api/services |
| URL parametri | (nema) |
| HTTP body parametri | (nema) |
| Format HTTP body parametara | Application/json |
| Izlazni parametri | {      "services": [          {              "id": 1,              "name": "Zamena guma",              "created\_at": "2023-05-05T21:27:44.000000Z",              "updated\_at": "2023-05-05T21:27:44.000000Z"          },          {              "id": 2,              "name": "Amortizeri popravka",              "created\_at": "2023-05-05T21:27:44.000000Z",              "updated\_at": "2023-05-05T21:27:44.000000Z"          },          {              "id": 3,              "name": "Zamena akumulatora",              "created\_at": "2023-05-05T21:27:44.000000Z",              "updated\_at": "2023-05-05T21:27:44.000000Z"          },          {              "id": 4,              "name": "Servo popravka",              "created\_at": "2023-05-05T21:27:44.000000Z",              "updated\_at": "2023-05-05T21:27:44.000000Z"          },          {              "id": 5,              "name": "Veliki servis",              "created\_at": "2023-05-05T21:27:44.000000Z",              "updated\_at": "2023-05-05T21:27:44.000000Z"          }      ]  } |
| Format izlaznih parametara | application/json |
| Opis funkcije | Prikaz određene usluge iz tabele Services |
| HTTP metoda | GET |
| URL | http://localhost:8000/api/services/{id} |
| URL parametri | $service\_id=$id |
| HTTP body parametri | (nema) |
| Format HTTP body parametara | Application/json |
| Izlazni parametri | {      "service": {          "name": "Veliki servis"      }  } |
| Format izlaznih parametara | application/json |
| Opis funkcije | Registracija korisnika |
| HTTP metoda | POST |
| URL | http://localhost:8000/api/register |
| URL parametri | (nema) |
| HTTP body parametri | Name: Ana  Email: [a@gmail.com](mailto:a@gmail.com)  Password: AnaLozinka2023! |
| Format HTTP body parametara | Application/json |
| Izlazni parametri | {      "user": {          "name": "Ivan",          "email": "ivan1@gmail.com",          "role": "user",          "updated\_at": "2023-05-09T08:41:20.000000Z",          "created\_at": "2023-05-09T08:41:20.000000Z",          "id": 6      },      "access\_token": "12|7GP9NLLQV2bjYtawvZSis9qXOSziZuT7Bi161Kk9",      "token\_type": "Bearer"  } |
| Format izlaznih parametara | application/json |
| Opis funkcije | Logoutovanje korisnika |
| HTTP metoda | POST |
| URL | http://localhost:8000/api/logout |
| URL parametri | (nema) |
| HTTP body parametri | (nema) |
| Format HTTP body parametara | Application/json |
| Izlazni parametri | {  "message": "You have successfully logged out and the token was successfully deleted"  } |
| Format izlaznih parametara | application/json |
| Opis funkcije | Loginovanje korisnika |
| HTTP metoda | POST |
| URL | http://localhost:8000/api/login |
| URL parametri | (nema) |
| HTTP body parametri | Email: [zarko1@gmail.com](mailto:zarko1@gmail.com)  Password: Zarko\*1998 |
| Format HTTP body parametara | Application/json |
| Izlazni parametri | {      "message": "Hi Ivan, welcome ",      "access\_token": "13|rG4KJo9MWHDg1uDjIYmpr0463MRvAAm92yHuKw2e",      "token\_type": "Bearer"  } |
| Format izlaznih parametara | application/json |
| Opis funkcije | Kreiranje nove ocene |
| HTTP metoda | POST |
| URL | http://localhost:8000/api/ratings |
| URL parametri | (nema) |
| HTTP body parametri | Date\_and\_time: 2023-01-20 11:41:07  Service: 5  Rating: 3  Note: Prosecan servis, nisam odusevljen uslugom  Mechanic: 5 |
| Format HTTP body parametara | Application/json |
| Izlazni parametri | [      "Rating is created successfully.",      {          "date\_and\_time": "2023-05-08 23:26:30",          "user": {              "name": "Ivan",              "email": "ivan1@gmail.com",              "role": "user"          },          "service": {              "name": "Veliki servis"          },          "rating": "3",          "note": "Prosecan servis, nisam odusevljen uslugom",          "mechanic": {              "name": "Maggie Von",              "phone\_number": "(817) 904-9848",              "years\_of\_experience": 10,              "email": "durgan.fausto@example.net"          }      }  ] |
| Format izlaznih parametara | application/json |
| Opis funkcije | Apdejtovanje ocene |
| HTTP metoda | PUT |
| URL | http://localhost:8000/api/apprat |
| URL parametri | (nema) |
| HTTP body parametri | Date\_and\_time: 2023-01-20 11:41:07  Service: 1  User: 6  Rating: 4  Note: Popravili su rad na servisu, sada zadovoljava standarde  Mechanic: 4 |
| Format HTTP body parametara | Application/json |
| Izlazni parametri | {      "date\_and\_time": [          "The date and time field is required."      ],      "user": [          "The user field is required."      ],      "service": [          "The service field is required."      ],      "rating": [          "The rating field is required."      ],      "note": [          "The note field is required."      ],      "mechanic": [          "The mechanic field is required."      ]  } |
| Format izlaznih parametara | application/json |

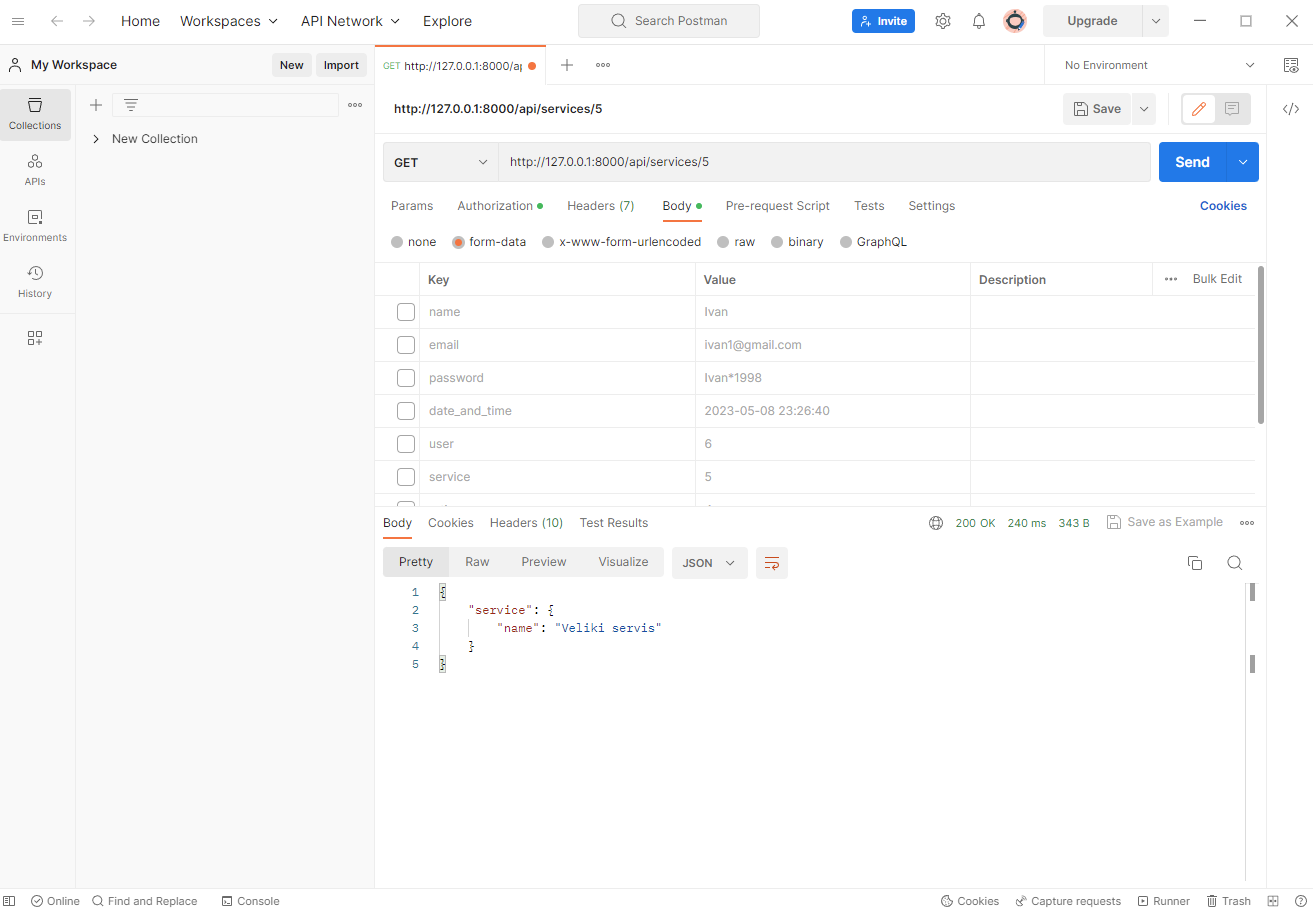
Postoji još mnogo mogućnosti za PUT, POST, DELETE metode, neke od njih može koristiti običan user a neke samo ’admin’ user.

# Korisničko uputstvo

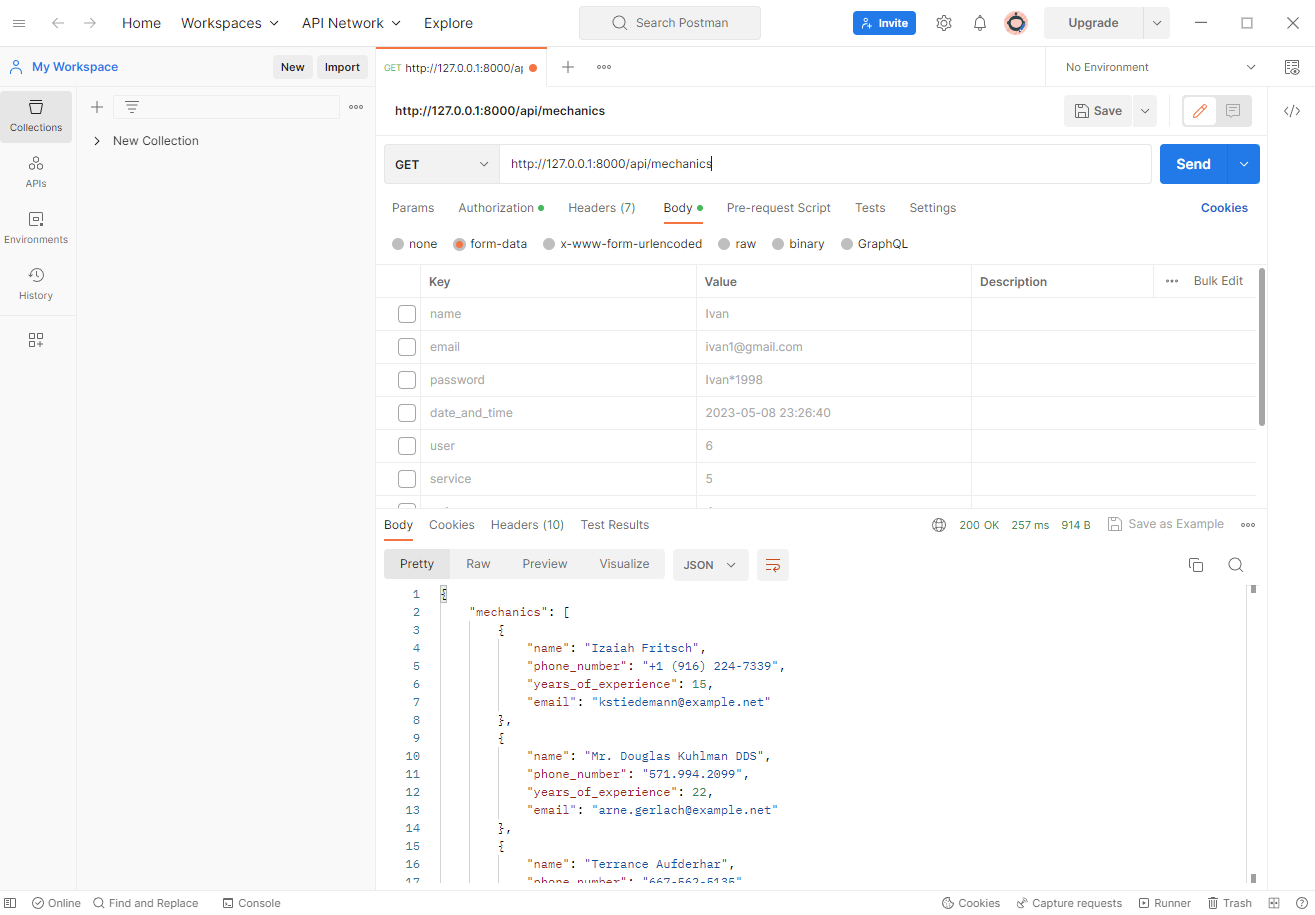
1. Zahtev: Pristup uslugama iz tabele Services preko GET metode



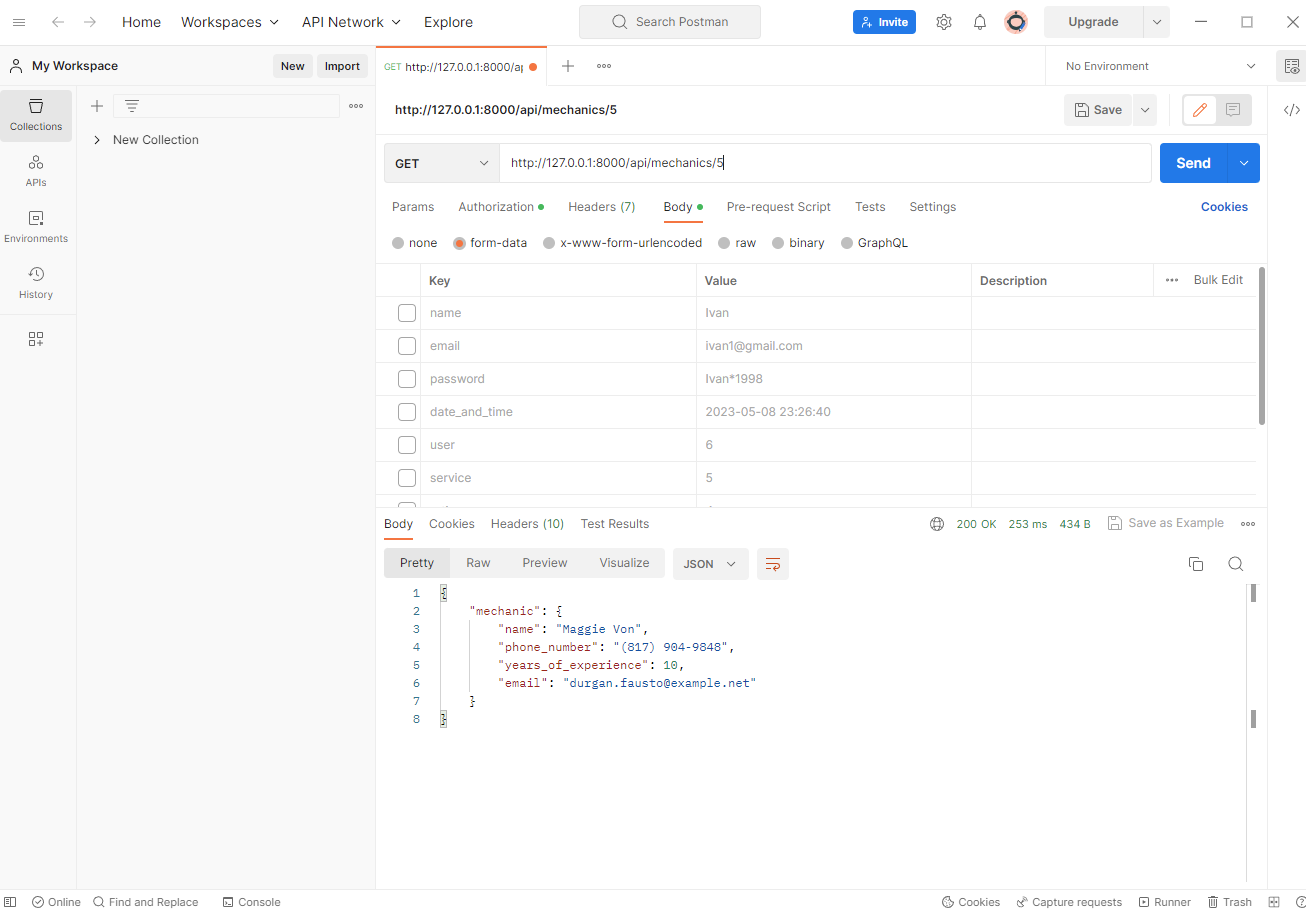
2. Zahtev: Pristup konkretnoj usluzi iz tabele Service preko id-a preko GET metod



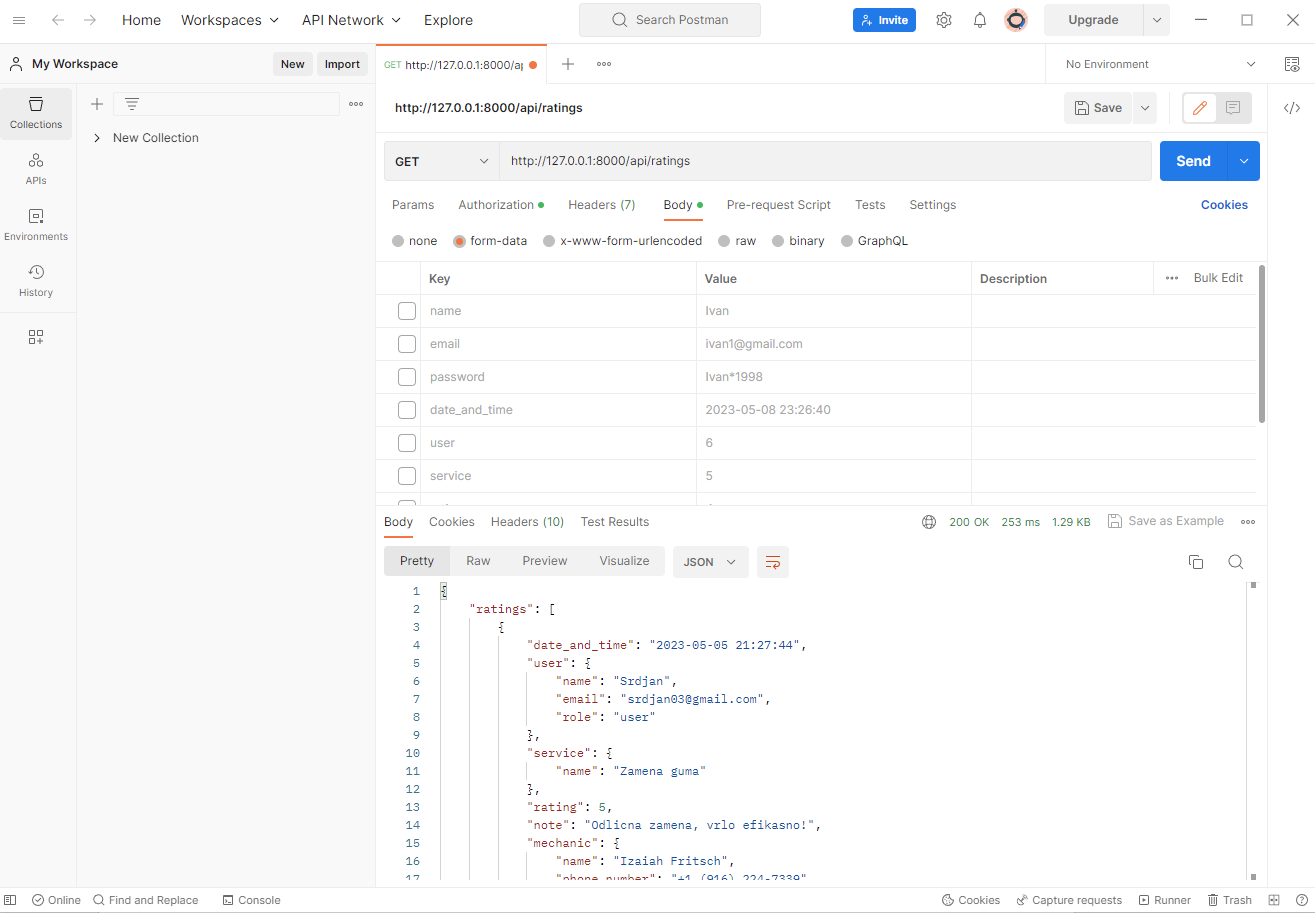
3. Zahtev: Pristup mehanicarima iz tabele Mechanics preko GET metode



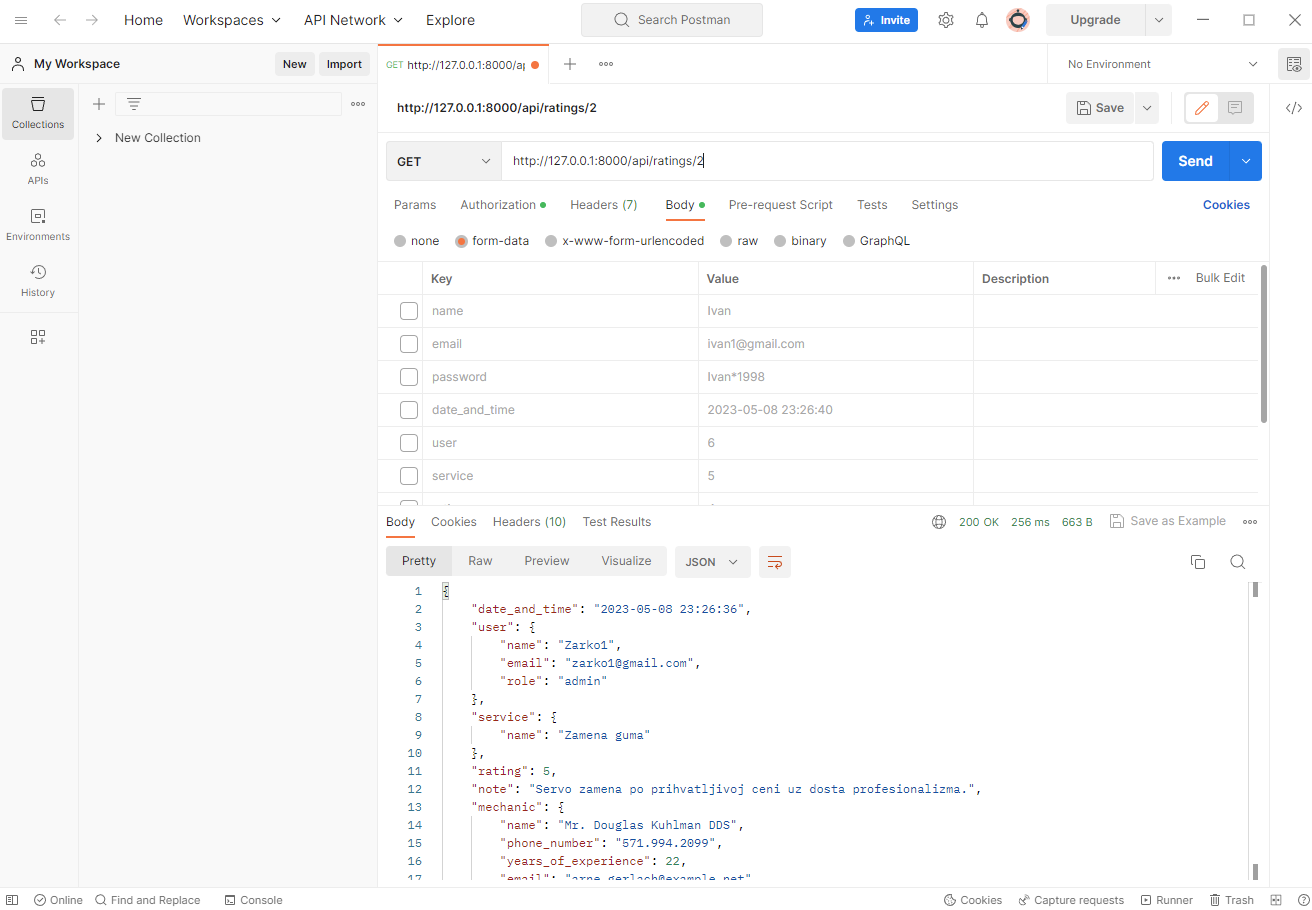
4. Zahtev: pristup konkretnom mehanicaru iz tabele Mechanics preko id-a preko GET metode



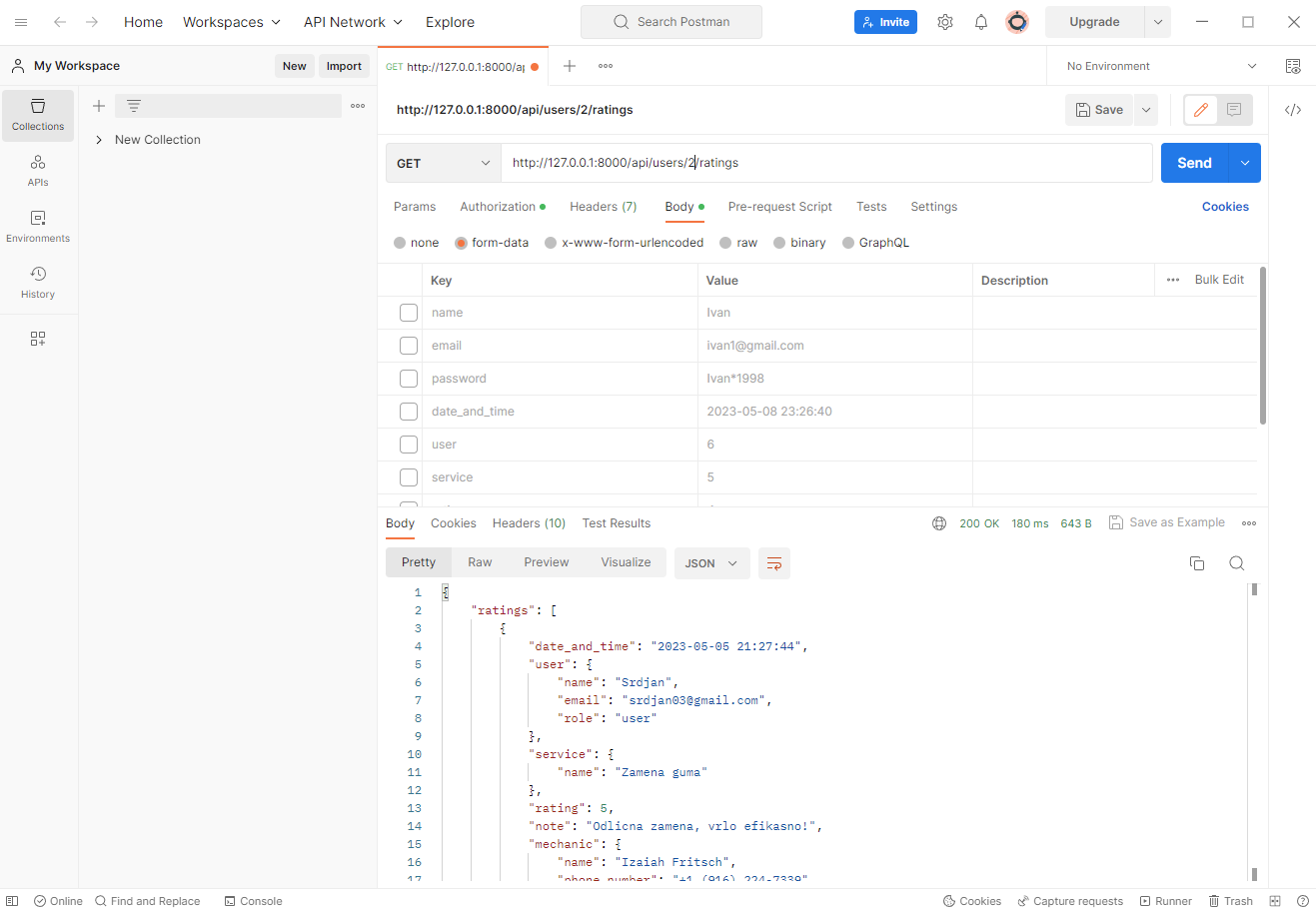
5. Zahtev: Prikaz svih ocena iz tabele Ratings preko GET metode



6. Zahtev: Prikaz konkretne ocene iz tabele Ratings preko id-a preko GET metode



7. Zahtev: primer GET metode – prikaz ocene koju je dao konkretan korisnik – navodimo users I {id} korisnika kako bi izvukli ocene koje je korisnik sa tim Idem dao



Kako ne bismo slikali sve moguće slučajeve, videli smo gde se menja sam tip metode: mi smo sve radili sa get i post metodom, ali na tom mestu se metoda može promeniti u PUT, PATCH, DELETE…

Neke metode ne može da izvrši korisnik koji nema autorizovan pristup preko AuthController-a, odnosno korisnik čija uloga nije tipa ‘admin’. Za takve usluge potrebno je da se obezbedi Bearer token, koji se dobija prilikom login-ovanja korisnika. Taj token se onda kopira u Authorization tab i nakon toga je omogućen autorizovan pristup zaštićenim metodama.