Univerzitet u Beogradu

Fakultet organizacionih nauka

Katedra za elektronsko poslovanje

Laravel

Domaći zadatak 1

Ime	Prezime	Broj indeksa
Marko	Ivanović	20150406
Link ka Github-u	https://github.com/elab-development/internet-tehnologije-projekat-ekocrowdsourcing_2015_0406	

Beograd, 2024

Sadržaj

1 Korisnički zahtev

Ovaj rad predstavlja backend deo crowdfunding platforme za ekološke projekte, izrađen u PHP framework-u Laravel, koja omogućava korisnicima da podrže različite ekološke inicijative putem donacija. Platforma sadrži četiri glavne tabele: Korisnici, Donacije, Projekti i Tipovi projekata.

Na početnoj strani, posetioci mogu da vide najnovije projekte, dok se lista svih projekata i donacija nalazi na njima odgovarajućim stranicama.

Ključne funkcionalnosti aplikacije uključuju:

- Prikaz tri najnovija ekološka projekta na početnoj strani.
- Prikaz projekata i donacija na odgovarajućim stranicama.
- Mogućnost kreiranja novih projekata za ulogovane korisnike (korisnici, moderatori i administratori).
- Doniranje koje je dostupno svim korisnicima, uključujući i neregistrovane posetioce.
- Administracija i moderacija sadržaja zavisno od uloge korisnika (korisnik, moderator, administrator).

Struktura podataka:

- Projekti: Naziv, opis, lokacija, tip (iz tabele Tipovi projekata) i korisnik koji je napravio projekat.
- Korisnici: Ime, email adresa, šifra i uloga korisnika (korisnik, moderator, administrator).
- Donacije: Email adresa donatora, neobavezni opis, iznos donacije i projekat za koji se donira.
- Tipovi projekata: Naziv tipa projekta kao što su npr. Deponija, Vazduh, Pošumljavanje, Korita.

Slučajevi korišćenja

1. Pregled početne strane

Akter: Posetilac
Opis: Posetilac pristupa početnoj strani i vidi tri najnovija ekološka projekta.
Preduslov: Sistem je uključen.
Tok radnje:

- Posetilac otvara početnu stranicu.
- Sistem prikazuje tri najnovija projekta.

2. Pregled liste projekata

Akter: Posetilac
Opis: Posetilac može da pregleda listu svih dostupnih projekata.
Preduslov: Sistem je uključen.
Tok radnje:

- Posetilac otvara stranicu sa listom projekata.
- Sistem prikazuje sve projekte sa njihovim detaljima.

3. Pregled liste donacija

Akter: Posetilac
Opis: Posetilac može da pregleda listu svih donacija.
Preduslov: Sistem je uključen.
Tok radnje:

- Posetilac otvara stranicu sa listom donacija.
- Sistem prikazuje sve donacije sa njihovim detaljima.

4. Doniranje

Akter: Posetilac
Opis: Posetilac može donirati za projekat čak i ako nije registrovan.
Preduslov: Sistem je uključen.
Tok radnje:

- Posetilac otvara stranicu projekta za koji želi da donira.
- Posetilac unosi email adresu, iznos donacije i neobavezni opis.
- Posetilac potvrđuje donaciju.
- Sistem beleži donaciju i prikazuje potvrdu.

5. Kreiranje novog projekta

Akter: Ulogovani korisnik (korisnik, moderator, administrator)
Opis: Ulogovani korisnik može kreirati novi ekološki projekat.
Preduslov: Sistem je uključen. Korisnik je ulogovan.
Tok radnje:

- Ulogovani korisnik otvara stranicu za kreiranje projekta.
- Korisnik unosi naziv, opis, lokaciju i tip projekta.
- Korisnik potvrđuje kreiranje projekta.
- Sistem beleži novi projekat i povezuje ga sa korisnikom.

6. Uređivanje i brisanje projekata

Akter: Moderator ili administrator
Opis: Moderator ili administrator može urediti ili obrisati postojeći projekat.
Preduslov: Sistem je uključen. Korisnik je moderator ili administrator.
Tok radnje:

- Moderator/administrator otvara stranicu za uređivanje projekta.
- Moderator/administrator menja podatke projekta ili bira opciju za brisanje.
- Sistem beleži promene ili briše projekat.

7. Upravljanje tipovima projekata

Akter: Administrator
Opis: Administrator može dodati ili obrisati tipove projekata.
Preduslov: Sistem je uključen. Korisnik je administrator.
Tok radnje:

- Administrator otvara stranicu za upravljanje tipovima projekata.
- Administrator unosi novi tip projekta ili bira tip za brisanje.
- Sistem beleži promene

2 Implementacija

Modeli

- **Korisnik (User)**

Model Korisnik sadrži informacije o korisnicima aplikacije, kao i logiku za autentifikaciju i autorizaciju.

Atributi jednog korisnika su:

- id
- name
- email
- password
- type
- created_at
- updated_at

Relacije:

- Korisnik može imati više projekata (one-to-many relacija sa modelom Project).

```
1. class User extends Authenticatable
2. {
3.     use Notifiable;
4. 
5.     protected $fillable = [
6.         'name', 'email', 'password', 'type',
7.     ];
8. 
```

```
    protected $hidden = [
        'password', 'remember_token',
    ];

    protected $casts = [
        'email_verified_at' => 'datetime',
    ];

    public function projects()
    {
        return $this->hasMany(Project::class);
    }
}
```

➢ Projekat (Project)

Model Projekat predstavlja ekološki projekat na platformi.

Atributi jednog projekta su:

- id
- name
- description
- location
- type_id (foreign key za tabelu types)
- user_id (foreign key za tabelu users)
- created_at
- updated_at

Relacije:

- Projekat pripada korisniku (many-to-one relacija sa modelom User).
- Projekat pripada tipu (many-to-one relacija sa modelom Type).
- Projekat može imati više donacija (one-to-many relacija sa modelom Donation).

```
class Project extends Model
{
    protected $fillable = [
        'name', 'description', 'location', 'type_id', 'user_id',
    ];

    public function user()
    {
        return $this->belongsTo(User::class);
    }

    public function type()
    {
        return $this->belongsTo(Type::class);
    }

    public function donations()
    {
        return $this->hasMany(Donation::class);
    }
}
```

➢ Donacija (Donation)

Model Donacija predstavlja donaciju za određeni projekat.

Atributi:

- id
- email
- amount
- description (neobavezno)
- project_id (foreign key za tabelu projects)
- created_at
- updated_at

Relacije:

- Donacija pripada projektu (many-to-one relacija sa modelom Project).

```php
class Donation extends Model
{
    protected $fillable = [
        'email', 'amount', 'description', 'project_id',
    ];

    public function project()
    {
        return $this->belongsTo(Project::class);
    }
}
```

➢ Tip (Type)

Model Tip predstavlja tip projekta.

Atributi:

- id
- name
- created_at
- updated_at

Relacije:

- Tip može imati više projekata (one-to-many relacija sa modelom Project).

```php
class Type extends Model
{
    protected $fillable = [
        'name',
    ];

    public function projects()
    {
        return $this->hasMany(Project::class);
    }
}
```

Migracije

Migracije definišu strukturu tabela u bazi podataka. U ovom projektu nalaze se migracije za kreiranje i ažuriranje tabela, dodavanje spoljnih ključeva, kao i brisanje ograničenja i kolona u tabelama. U „up“

delu migracije dodajemo nove tabele, kolone, ograničenja, dok u „down“ delu pišemo operacije koje će poništiti „up“ delovanje, odnosno vratiti tabelu u stanje pre izvršavanja.

Migracija za kreiranje tabele korisnika (create_users_table)

Definiše tabelu **users** sa atributima id, name, email, password, type, timestamps.

```
    public function up(): void
    {
        Schema::create('users', function (Blueprint $table) {
            $table->id();
            $table->string('name');
            $table->string('lastname');
            $table->string('email')->unique();
            $table->string('password');
            $table->string('type');
            $table->timestamp('email_verified_at')->nullable();
            $table->rememberToken();
            $table->timestamps();
        });
    }
    public function down(): void
    {
        Schema::dropIfExists('users');
    }
```

Migracija za kreiranje tabele projekata (create_projects_table)

Definiše tabelu **projects** sa atributima id, name, description, location, type_id, user_id, timestamps. type_id i user_id su spoljni ključevi koji referenciraju tabele types i users.

```
    public function up(): void
    {
        Schema::create('projects', function (Blueprint $table) {
            $table->id();
            $table->string('name');
            $table->string('type');
            $table->string('location');
            $table->string('description')->nullable();
            $table->timestamps();
        });
    }
    public function down(): void
    {
        Schema::dropIfExists('projects');
    }
```

Migracija za kreiranje tabele donacija (create_donations_table)

Definiše tabelu **donations** sa atributima id, email, amount, description, project_id, timestamps, gde je project_id je spoljni ključ koji referencira tabelu projects.

```
    public function up(): void
    {
        Schema::create('donations', function (Blueprint $table) {
            $table->id();
            $table->string('email');
            $table->string('amount');
            $table->foreignId('project_id');
            $table->timestamps();
        });
    }
    public function down(): void
    {
        Schema::dropIfExists('donations');
    }
```

Migracija za kreiranje tabele tipova (create_types_table)

Definiše tabelu **types** sa atributima id, name, timestamps.

```
        Schema::create('types', function (Blueprint $table) {
            $table->id();
            $table->string('name')->unique();
            $table->timestamps();
        });
```

Migracije za ažuriranje postojećih tabela:

Promena vrednosti atributa „type“ za Korisnika iz string u enum sa mogućim vrednostima: admin, mod i user.

```
    Public function up(): void
    {
         Schema::table('users', function (Blueprint $table) {
            $table->enum('type', ['admin', 'mod', 'user'])->default('user')-
>change();
        });
    }
    public function down(): void
    {
         Schema::table('users', function (Blueprint $table) {
            $table->string('type')->change();
        });
    }
```

Povezivanje tabela Projekti i Korisnici postavljanjem atributa „user_id“ kao spoljni ključ. U „down“ delu prvo skidamo ograničenje spoljnog ključa, a zatim brišemo kolonu „user_id“.

```
    public function up(): void
    {
        Schema::table('projects', function (Blueprint $table) {
            $table->foreignId('user_id');
        });
    }
    public function down(): void
    {
        Schema::table('projects', function (Blueprint $table) {
            $table->dropForeign(['user_id']);
            $table->dropColumn('user_id');
        });
    }
```

Dodavanje opcionog atributa „description“ za tabelu Donacije.

```
    public function up(): void
    {
        Schema::table('donations', function (Blueprint $table) {
            $table->string('description')->nullable();
        });
    }
    public function down(): void
    {
        Schema::table('donations', function (Blueprint $table) {
            $table->dropColumn('description');
        });
    }
```

Povezivanje tabela Projekti i Tipovi postavljanjem atributa „type_id“ kao spoljni ključ.

```
    public function up(): void
    {
        Schema::table('projects', function (Blueprint $table) {
            $table->foreignId('type_id')->nullable()->constrained('types');
        });
    }
    public function down(): void
    {
        Schema::table('projects', function (Blueprint $table) {
            $table->dropForeign(['type_id']);
            $table->dropColumn('type_id');
        });
    }
```

Migracije za brisanje kolone iz tabele:

Brisanje kolone „lastname“ iz tabele Korisnici.

```
        Schema::table('users', function (Blueprint $table) {
            $table->dropColumn('lastname');
        });
```

Brisanje kolone „type“, iz tabele Projekti, čiju smo svrhu prethodno zamenili spoljnim ključem „type_id“.

```
    public function up(): void
    {
        Schema::table('projects', function (Blueprint $table) {
            $table->dropColumn('type');
        });
    }
    public function down(): void
    {
        Schema::table('projects', function (Blueprint $table) {
            $table->string('type');
        });
    }
```

Kontroleri

Kontroleri koje imamo su:

- UserController – sadrži funkcije za prikaz i obradu podataka Korisnika.
- ProjectController – sadrži funkcije za prikaz i obradu podataka Projekata.
- DonationController – sadrži funkcije za prikaz i obradu podataka Donacija.
- TypeController – sadrži funkcije za prikaz i obradu podataka Tipova projekata.
- UserProjectController – sadrži funkcije za prikaz projekata po korisniku.
- ProjectDonationController – sadrži funkciju za prikaz donacija po projektu.
- AuthController – sadrži funkcije za registraciju, prijavu i odjavu korisnika.

Funkcije UserController-a:

- Index: Prikazuje sve korisnike.
- Show: Prikazuje pojedinačnog korisnika.
- store: Kreira novog korisnika.
- update: Ažurira nekog postojećeg korisnika.
- updateMe: Ažurira trenutnog korisnika.
- destroy: Briše korisnika.

```
class UserController extends Controller
{
    public function index()
    {
        $users = User::all();
```

```php

        return new UserCollection($users);
    }
    public function store(Request $request)
    {
        $validatedData = $request->validate([
            'name' => 'required|string|max:255',
            'email' => 'required|email|unique:users|max:255',
            'password' => 'required|min:8',
        ]);

        $validatedData['type'] = 'user';

        $user = User::create($validatedData);

        return response()->json(['User created successfully', new
UserResource($user)]);
    }

    protected function failedValidation(Validator $validator)
    {
        throw new HttpResponseException(response()->json(['Validation
error' => $validator->errors()], 422));
    }
    public function show(User $user)
    {
        return new UserResource($user);
    }

    public function showCurrent()
    {
        $user = Auth::user();

        if ($user) {
            return new UserResource($user);
        } else {
            return response()->json(['message' => 'Unauthorized'], 401);
        }
    }

     public function updateMe(Request $request)
     {
        $user = Auth::user();
        $updateUser = User::find($user->id);

```

```
          $validatedData = $request->validate([
               'name' => 'sometimes|string|max:255',
               'email' => 'sometimes|email|max:255',
               'password' => 'sometimes|min:8'
           ]);

           $updateUser->update($validatedData);
           return response()->json(['message' => 'User updated
   successfully', 'User:' => new UserResource($updateUser)]);
       }

      public function update(Request $request, $id)
      {
          $authUser = Auth::user();

           if ($authUser->type!=='admin') {
              return response()->json(['message' => 'Unauthorized.'], 403);
          }

          $user = User::find($id);

          if (!$user) {
              return response()->json(['message' => 'User not found'], 404);
          }

          $validatedData = $request->validate([
              'name' => 'sometimes|string|max:255',
              'email' => 'sometimes|email|max:255',
              'password' => 'sometimes|min:8',
              'type' => 'sometimes|string|max:255'
          ]);

          $user->update($validatedData);

          return response()->json(['message' => 'User updated successfully',
   'User:' => new UserResource($user)]);
      }

      public function destroy(User $user)
      {
          if (!$user) {
              return response()->json(['message' => 'User not found'], 404);
          }
          $user->delete();
```

```
        return response()->json(['message' => 'User deleted
successfully']);
    }
}
```

ProjectController:

- index: Prikazuje sve projekte, sa opcijama za filtriranje prema tipu i lokaciji.
- show: Prikazuje pojedinačni projekat.
- latestProjects – Prikazuje 3 najnovija projekta.
- store: Kreira novi projekat.
- update: Ažurira postojeći projekat.
- destroy: Briše projekat.

```
class ProjectController extends Controller
{
    public function index(Request $request)
    {
         $query = Project::query();

         if ($request->has('type_id')) {
             $query->type($request->input('type_id'));
         }

         if ($request->has('location')) {
             $query->location($request->input('location'));
         }
        $projects = $query->paginate(10);

        return new ProjectCollection($projects);
    }

    public function latestProjects(){
        $latestProjects = Project::orderBy('created_at', 'desc')->take(3)-
>get();
        return response()->json(['3 latest projects:' =>
$latestProjects]);
    }
    public function store(Request $request)
    {
        $user_id = Auth::id();

        $validatedData = $request->validate([
            'name' => 'required|string|max:255',
            'type_id' => 'required|exists:types,id',
```

```
            'location' => 'required|min:2',
            'description' => 'sometimes|min:4',
        ]);

        $validatedData['user_id'] = $user_id;

        $project = Project::create($validatedData);

        return new ProjectResource($project);
    }
    public function show(Project $project)
    {
       return new ProjectResource($project);
    }
    public function update(Request $request, $id)
    {
        $project = Project::find($id);
        if (!$project) {
            return response()->json(['message' => 'Project not found'], 404);
        }
        $validatedData = $request->validate([
            'name' => 'sometimes|string|max:255',
            'type_id' => 'sometimes|exists:types,id',
            'description'=>'sometimes',
            'location' => 'sometimes|min:4'
        ]);

        $project->update($validatedData);
        return response()->json(['message' => 'Project updated successfully', 'Project:' => new ProjectResource($project)]);

    }
    public function destroy($id)
    {
        $project = Project::find($id);
        if (!$project) {
            return response()->json(['message' => 'Project not found'], 404);
        }
        $project->delete();

        return response()->json(['message' => 'Project deleted successfully']);
    }}
```

DonationController:

- index: Prikazuje sve donacije, sa opcijama za filtriranje prema projektu email adresi donatora.
- show: Prikazuje pojedinačnu donaciju.
- store: Kreira novu donaciju.
- update: Ažurira postojeću donaciju.
- destroy: Briše donaciju.

```php
class DonationController extends Controller
{
    public function index(Request $request)
    {
        $query = Donation::query();

        if ($request->has('email')) {
            $query->email($request->input('email'));
        }

        if ($request->has('project_id')) {
            $query->projectId($request->input('project_id'));
        }

        $donations = $query->paginate(10);

        return new DonationCollection($donations);
    }
    public function store(Request $request)
    {
        $validatedData = $request->validate([
            'email' => 'required|string|max:255',
            'amount' => 'required|numeric',
            'description' => 'nullable|string',
            'project_id' => 'required'
        ]);

        $donation = Donation::create($validatedData);

        return new DonationResource($donation);
    }
    public function show(Donation $donation)
    {
        return new DonationResource($donation);
    }
    public function update(Request $request, $id)
    {
```

```
        $donation = Donation::find($id);
        if (!$donation) {
            return response()->json(['message' => 'Donation not found'], 404);
        }

        $validatedData = $request->validate([
            'email' => 'sometimes|string|max:255',
            'amount' => 'sometimes|numeric',
            'description' => 'sometimes|string',
            'project_id' => 'sometimes'
        ]);

        $donation->update($validatedData);

        return response()->json(['message' => 'Donation updated successfully', 'Donation:' => new DonationResource($donation)]);
    }
    public function destroy($id)
    {
        $donation = Donation::find($id);
        if (!$donation) {
            return response()->json(['message' => 'Donation not found'], 404);
        }
        $donation->delete();

        return response()->json(['message' => 'Donation deleted successfully']);
    }
}
```

TypeController:

- index: Prikazuje sve tipove projekata.
- show: Prikazuje pojedinačni tip projekta.
- store: Kreira nov tip projekta.
- destroy: Briše tip projekta.

```
class TypeController extends Controller
{
    public function index(Request $request)
    {
        $types = Type::all();

        return new TypeCollection($types);
```

```
    }
    public function store(Request $request)
    {
        $validatedData = $request->validate([
            'name' => 'required|string|max:255|unique:types,name',
        ]);

        $type = Type::create($validatedData);

        return response()->json(['message' => 'Type created successfully',
'type' => $type], 201);
    }
    public function show(Type $type)
    {
        return new TypeResource($type);
    }
    public function destroy(Type $type)
    {
        $type->delete();

        return response()->json(['message' => 'Type deleted
successfully'], 200);
    }
}

```

ProjectDonationController:

- index: Prikazuje sve donacije određenog projekta.

```
class ProjectDonationController extends Controller
{
    public function index($project_id){

        $donations = Donation::where('project_id',$project_id)->get();
        if($donations->isEmpty()){
            return response()->json('No donations', 404);
        }
        return new DonationCollection($donations);
    }
}
```

UserProjectController:

- index: Prikazuje sve projekte određenog korisnika.

```
class UserProjectController extends Controller
{
    public function index($user_id){
```

```
        $projects = Project::where('user_id',$user_id)->get();
        if($projects->isEmpty()){
            return response()->json('No projects', 404);
        }
        return new ProjectCollection($projects);
    }
}
```

AuthController:

- register: Prikazuje sve donacije određenog projekta.
- login: Prijavljuje korisnika na sistem.
- logout: Odjavljuje korisnika sa sistema.

```
class AuthController extends Controller
{
    public function register(Request $request){

        $data = $request->all();
        if ($data['type']!='user' && (Auth::check() && Auth::user()->type
!= 'admin')) {
            $data['type']='user';
        }

        $validator = Validator::make($data,[
            'name' => 'required|string|max:255',
            'email' => 'required|email|unique:users|max:255',
            'password' => 'required|min:8',
        ]);
        $data['type'] = 'user';

        if($validator->fails()){
            return response()->json($validator->errors());
        }

        $user = User::create($data);

        $token = $user->createToken('auth_token')->plainTextToken;

        return response()->json(['User created successfully', new
UserResource($user)]);
    }

    public function login(Request $request){
```

```
        if ($request->bearerToken()) {
            return response()->json(['message' => 'Already logged in'],
200);
        }

        if (!Auth::attempt($request->only('email', 'password'))) {
            return response()->json(['sucess'=>false]);
        }

        $user = User::where('email', $request['email'])->firstOrFail();

        Auth::login($user);

        $token = $user->createToken('auth_token')->plainTextToken;

        return response()->json(['sucess'=>true,
            'access_token' => $token,
            'token_type' => 'Bearer',
        ]);
    }

    public function logout(Request $request){
        $request->user()->currentAccessToken()->delete();

        return response()->json(['message' => 'User logged out
successfully']);
    }
```

Takođe su definisani i middleware-i za kontrolisanje pristupa rutama na osnovu tipa korisnika. To su middleware admin, admin-or-mod, mod. Osim toga imamo i Sanctum preko kojeg dodeljujemo tokene za pristup ulovoganim korisnicima.

U nastavku se nalazi lista **API ruta** sortiranih po ograničenjima pristupa.

Rute kojima mogu da pristupe i gosti i registrovani korisnici:

```
Route::resource('donations', DonationController::class)->only(['show','index']);
Route::resource('projects.donations', ProjectDonationController::class)-
>only(['index']);
Route::resource('projects', ProjectController::class)->only(['show','index']);
Route::post('/register', [AuthController::class, 'register']);
Route::post('/login', [AuthController::class, 'login']);
```

Rute kojima mogu da pristupe samo registrovani korisnici:

```
Route::group(['middleware' => ['auth:sanctum']], function () {
    Route::patch('/update-user', [UserController::class, 'updateMe']);
    Route::get('/profile', [UserController::class, 'showCurrent']);
    Route::post('/logout', [AuthController::class, 'logout']);
    Route::post('/create-donation', [DonationController::class, 'store']);
    Route::resource('users.projects', UserProjectController::class)-
>only(['show','index']);
    Route::resource('projects', ProjectController::class)->only(['store']);
});
```

Rute kojima mogu da pristupe korisnici tipa Admin ili Moderator:

```
Route::group(['middleware' => ['auth:sanctum','admin-or-mod']], function ()
{
    Route::resource('users', UserController::class)->only(['show','index']);
    Route::resource('projects', ProjectController::class);
    Route::delete('/delete-project/{id}', [ProjectController::class,'destroy']);
});
```

Rute za manipulaciju kojima mogu da pristupe samo Admin korisnici:

```
Route::group(['middleware' => ['auth:sanctum', 'admin']], function () {
        Route::patch('/update-user/{id}', [UserController::class, 'update']);
        Route::resource('users', UserController::class);
        Route::resource('donations', DonationController::class);
        Route::post('types', [TypeController::class, 'store']);
        Route::delete('types/{type}', [TypeController::class, 'destroy']);
        Route::get('/admin-dashboard', function () {
        return 'Welcome to the admin dashboard!';
    });
});
```

3 REST API

Opis funkcije	Registrovanje korisnika
HTTP metoda	POST
URL	/api/register
URL parametri	(nema)

HTTP body parametri	{
	"name": "test korisnik",
	"email": "korisnik@test.com",
	"password": "12345678",
	}

Format HTTP body parametara	JSON

Izlazni parametri	{
	"User created successfully",
	{
	"name": "test korisnik",
	"email": "korisnik@test.com",
	"type": "user"
	}
	
	}

Format izlaznih parametara	application/json
Opis funkcije	Prijavljivanje na platformu
HTTP metoda	POST
URL	/api/login
URL parametri	(nema)

HTTP body parametri	{
	"email": "korisnik@test.com",
	"password": "12345678",
	}

Format HTTP body parametara	JSON

Izlazni parametri	{
	"sucess": true,
	"access_token":
	"4|KRNULMXcerEXwClCg0eA5VZ3DnW6F7169qYU8sx90ab4ec89",
	"token_type": "Bearer"
	}

Format izlaznih parametara	application/json
Opis funkcije	Pregled postojecih projekata
HTTP metoda	GET
URL	/api/projects
URL parametri	(nema)
HTTP body parametri	(nema)
Format HTTP body parametara	JSON

Izlazni parametri	{
	"Projects": [Lista projekata]
	}

Format izlaznih parametara	application/json

Opis funkcije	Prikaz jednog projekta na osnovu njegovog id-a
HTTP metoda	GET
URL	/api/projects/
URL parametri	{id}
HTTP body parametri	(nema)
Format HTTP body parametara	JSON

Izlazni parametri	{
	"Project": {
	"name": "McDermott",
	"type": "Korita",
	"location": "Lake Kory",
	"project description": "Praesentium molestias odio voluptatum sit velit eum et voluptas.",
	"user": "Miss Delphia Walker DVM"
	}
	}

Format izlaznih parametara	application/json
Opis funkcije	Prikaz svih donacija
HTTP metoda	GET
URL	/api/donations
URL parametri	(nema)
HTTP body parametri	(nema)
Format HTTP body parametara	JSON

Izlazni parametri	{
	"Donations": [
	{
	"email": "jedan@jedan.com",
	"amount": "111",
	"donation message": "opis1",
	"project": {
	"name": "McDermott",
	"type": "Korita",
	"location": "Lake Kory",
	"project description": "Praesentium molestias odio voluptatum sit velit eum et voluptas.",
	"user": "Miss Delphia Walker DVM"
	}
	},
	]

Format izlaznih parametara	application/json
Opis funkcije	Pregled svih donacija za odredjen projekat

HTTP metoda	GET
URL	/api/projects/{id}/donation
URL parametri	Id projekta čije donacije prikazujemo
HTTP body parametri	(nema)
Format HTTP body parametara	JSON

Izlazni parametri	{
	"Donations": [
	]
	}

Format izlaznih parametara	application/json
Opis funkcije	Kreiranje novog projekta
HTTP metoda	POST
URL	/api/projects
URL parametri	(nema)

HTTP body parametri	{
	"name": "test projekat",
	"type_id": "1",
	"location": "test lokacija",
	"description": "test opis",
	}

Format HTTP body parametara	JSON

Izlazni parametri	{
	"Project": {
	"name": "test projekat",
	"type": "Deponija",
	"location": "test lokacija",
	"project description": "test opis",
	"user": "Admin"
	}
	}

Format izlaznih parametara	application/json
Opis funkcije	Prikaz projekata određenog korisnika
HTTP metoda	GET
URL	/api/users/1/projects
URL parametri	1 – id korisnika
HTTP body parametri	(nema)
Format HTTP body parametara	JSON
Izlazni parametri	{ "Projects": [

	<pre>{
	"name": "123",
	"type": "Vazduh",
	"location": "test",
	"project description": "test",
	"user": "Admin"
	}
	]
	}</pre>

Format izlaznih parametara	application/json
Opis funkcije	Kreiranje nove donacije
HTTP metoda	POST
URL	/api/create-donation
URL parametri	(nema)

HTTP body parametri	<pre>{
	"email": "test@email.com",
	"amount": "13.12",
	"description":"opisopis",
	"project_id": "1"
	}</pre>

Format HTTP body parametara	JSON

Izlazni parametri	<pre>{
	"Donation": {
	"email": "test@email.com",
	"amount": "13.12",
	"donation message": "opisopis",
	"project": {
	"name": "McDermott",
	"type": "Korita",
	"location": "Lake Kory",
	"project description": "Praesentium molestias odio voluptatum sit velit eum et voluptas.",
	"user": "Miss Delphia Walker DVM"
	}
	}
	}</pre>

Format izlaznih parametara	application/json
Opis funkcije	Odjava korisnika sa sistema
HTTP metoda	POST
URL	/api/logout
URL parametri	(nema)

HTTP body parametri	(nema)
Format HTTP body parametara	JSON

Izlazni parametri	<pre>{
	  "message": "User logged out successfully"
	}</pre>

Format izlaznih parametara	application/json
Opis funkcije	Pregled profila trenutnog korisnika
HTTP metoda	POST
URL	/api/profile
URL parametri	(nema)
HTTP body parametri	(nema)
Format HTTP body parametara	JSON

Izlazni parametri	<pre>{
	  "User": {
	  "name": "Admin",
	  "email": "admin@admin.com",
	  "type": "admin"
	  }
	}</pre>

Format izlaznih parametara	application/json
Opis funkcije	Ažuriranje sopstvenog profila
HTTP metoda	PATCH
URL	/api/update-user
URL parametri	(nema)

HTTP body parametri	<pre>{
	  "name": "test",
	  "email": "test@email.com",
	}</pre>

Format HTTP body parametara	JSON

Izlazni parametri	<pre>{
	  "message": "User updated successfully",
	  "User:": {
	  "name": "test",
	  "email": "test@email.com",
	  "type": "admin"
	  }
	}</pre>

Format izlaznih parametara	application/json
Opis funkcije	Brisanje projekta
HTTP metoda	DELETE
URL	/api/delete-project/
URL parametri	{id}
HTTP body parametri	(nema)
Format HTTP body parametara	JSON

Izlazni parametri	<pre>{
	"message": "Project deleted successfully"
	}</pre>

Format izlaznih parametara	application/json
Opis funkcije	Ažuriranje projekta
HTTP metoda	PATCH
URL	/api/projects/
URL parametri	{id}

HTTP body parametri	<pre>{
	"name": "test name",
	"type_id": "2",
	"location": "projekat123",
	"description":"11111"
	}</pre>

Format HTTP body parametara	JSON

Izlazni parametri	<pre>{
	"message": "Project updated successfully",
	"Project:": {
	"name": "test name",
	"type": "Vazduh",
	"location": "projekat123",
	"project description": "11111",
	"user": "Miss Delphia Walker DVM"
	}
	}</pre>

Format izlaznih parametara	application/json
Opis funkcije	Prikaz liste korisnika
HTTP metoda	GET
URL	/api/users
URL parametri	(nema)
HTTP body parametri	(nema)

Format HTTP body parametara	JSON

Izlazni parametri	<pre>{
	"Users": [
	{
	"name": "Admin",
	"email": "admin@admin.com",
	"type": "admin"
	},
	{
	"name": "Mod",
	"email": "mod@mod.com",
	"type": "mod"
	},
	{
	"name": "User",
	"email": "User@user.com",
	"type": "user"
	}]
	}</pre>

Format izlaznih parametara	application/json
Opis funkcije	Prikaz tipova projekata
HTTP metoda	GET
URL	/api/types
URL parametri	(nema)
HTTP body parametri	(nema)
Format HTTP body parametara	JSON

Izlazni parametri	<pre>{
	"Types:": [
	{
	"name": "Deponija"
	},
	{
	"name": "Vazduh"
	},
	{
	"name": "Posumljavanje"
	},
	{
	"name": "Korita"
	}
	]
	}</pre>

Format izlaznih parametara	json
Opis funkcije	Prikaz određenog tipa projekta
HTTP metoda	GET
URL	/api/types
URL parametri	{id}
HTTP body parametri	(nema)
Format HTTP body parametara	JSON

Izlazni parametri	{
	"Project type:": {
	"name": "Deponija"
	}
	}

Format izlaznih parametara	json

Opis funkcije	Brisanje tipa projekta
HTTP metoda	POST
URL	/api/type/
URL parametri	{id}
HTTP body parametri	(nema)
Format HTTP body parametara	JSON

Izlazni parametri	{
	"message": "Type deleted successfully"
	}

Format izlaznih parametara	json
Opis funkcije	Kreiranje novog tipa
HTTP metoda	POST
URL	/api/types
URL parametri	(nema)

HTTP body parametri	{
	"name": "test tip",
	}

Format HTTP body parametara	JSON

Izlazni parametri	{
	"message": "Type created successfully",
	"type": {
	"name": "test type",
	"updated_at": "2024-05-28T14:54:20.000000Z",

	<pre>"created_at": "2024-05-28T14:54:20.000000Z",
	"id": 7
	}
	}</pre>

Format izlaznih parametara	application/json
Opis funkcije	Ažuriranje donacije
HTTP metoda	PATCH
URL	/api/donations
URL parametri	{id}

HTTP body parametri	<pre>{
	"amount": "99.99",
	"description":"UPDATE",
	"project_id": "2"
	}</pre>

Format HTTP body parametara	JSON

Izlazni parametri	<pre>{
	"message": "Donation updated successfully",
	"Donation:": {
	"email": "UPDATE@email.com",
	"amount": "99.99",
	"donation message": "UPDATE",
	"project": {
	"name": "Beer",
	"type": "Korita",
	"location": "Boganshire",
	"project description": "Corrupti corporis et quia sequi recusandae.",
	"user": "Bryce Bruen DVM"
	}
	}
	}</pre>

Format izlaznih parametara	application/json
Opis funkcije	Brisanje donacije
HTTP metoda	DELETE
URL	/api/donations/
URL parametri	{id}
HTTP body parametri	(nema)
Format HTTP body parametara	JSON
Izlazni parametri	{

	<pre>{
	    "message": "Donation deleted successfully"
	}</pre>

Format izlaznih parametara	application/json
Opis funkcije	Ažuriranje bilo kog korisnika
HTTP metoda	PUT
URL	/api/update-users/
URL parametri	{id}

HTTP body parametri	<pre>{
	  "name": "USER update",
	  "email": "test@update.com",
	  "password": "newpassword1234",
	  "type": "mod"
	}</pre>

Format HTTP body parametara	JSON

Izlazni parametri	<pre>{
	    "message": "User updated successfully",
	    "User:": {
	        "name": "USER update",
	        "email": "test@update.com",
	        "type": "mod"
	    }
	}</pre>

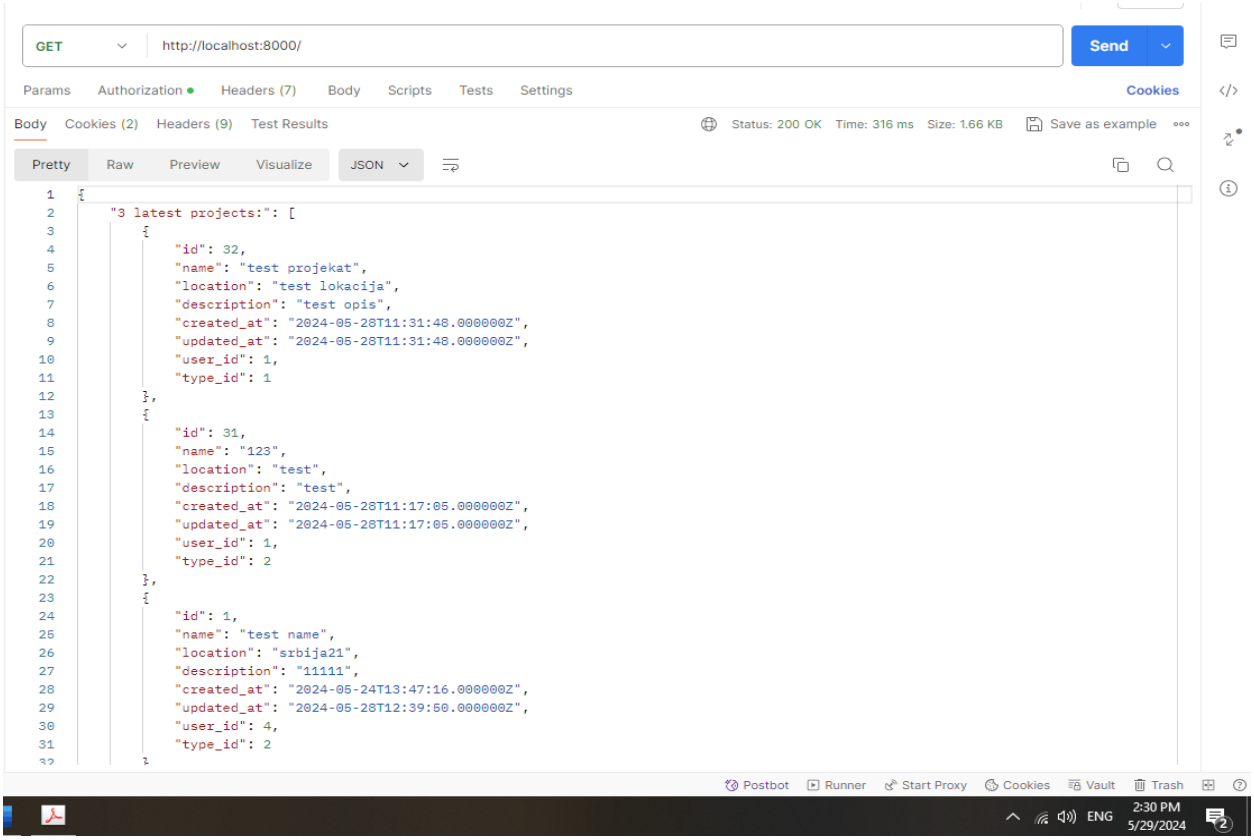
Format izlaznih parametara	application/json
Opis funkcije	Brisanje korisnika
HTTP metoda	DELETE
URL	/api/users/
URL parametri	{id}
HTTP body parametri	(nema)
Format HTTP body parametara	JSON

Izlazni parametri	<pre>{
	    "message": "User deleted successfully"
	}</pre>

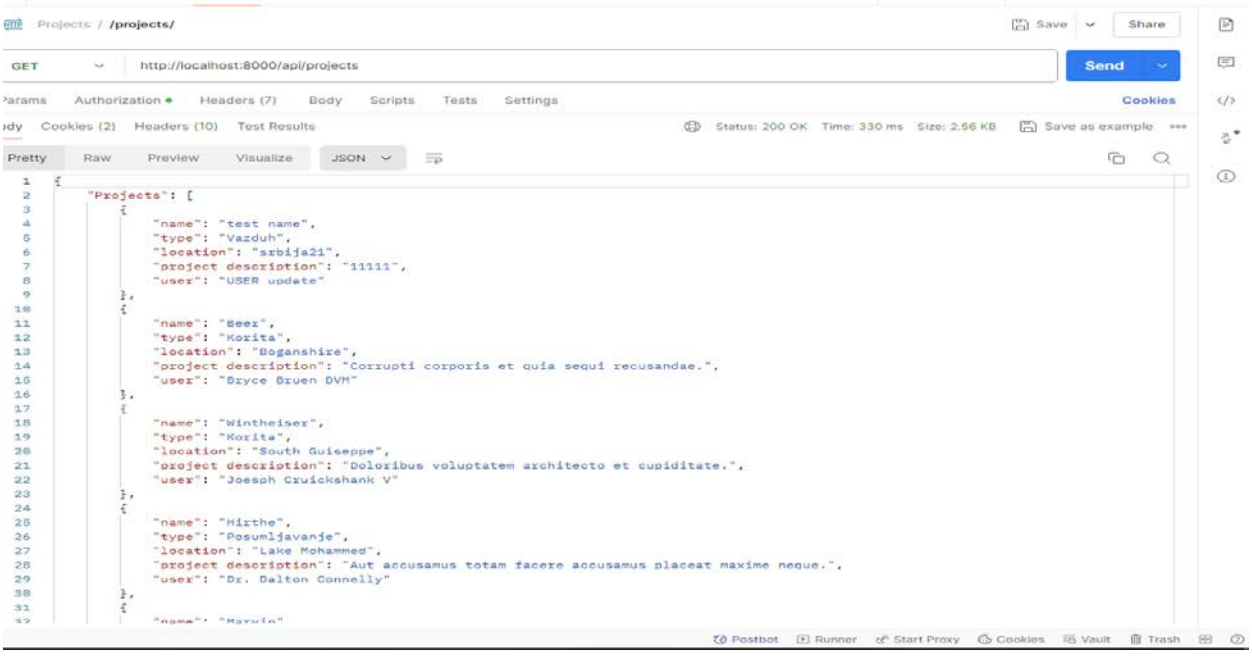
Format izlaznih parametara	json

4 Korisničko uputstvo

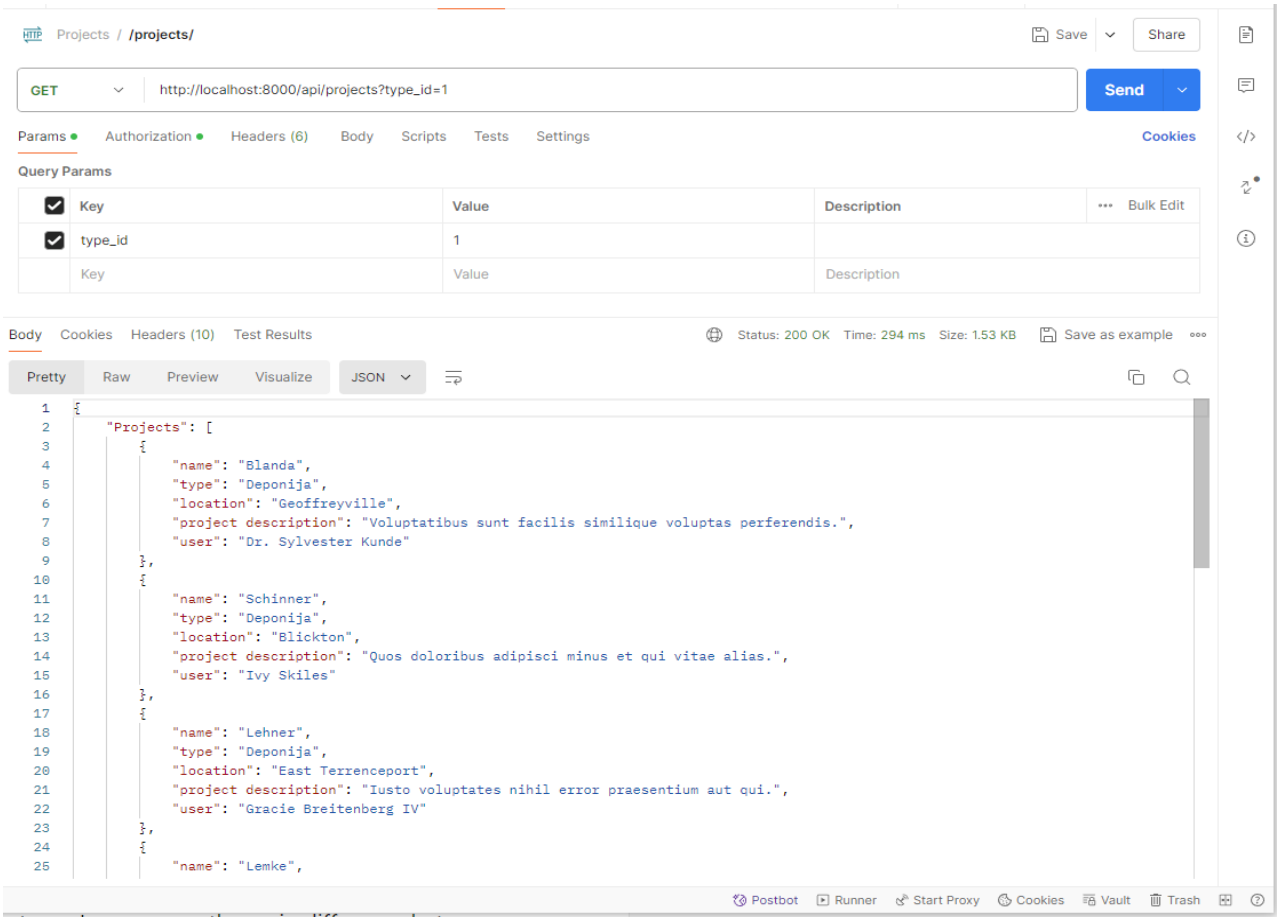
1. Korisnik odlazi na početnu stranicu gde se prikazuju 3 najnovija projekta.

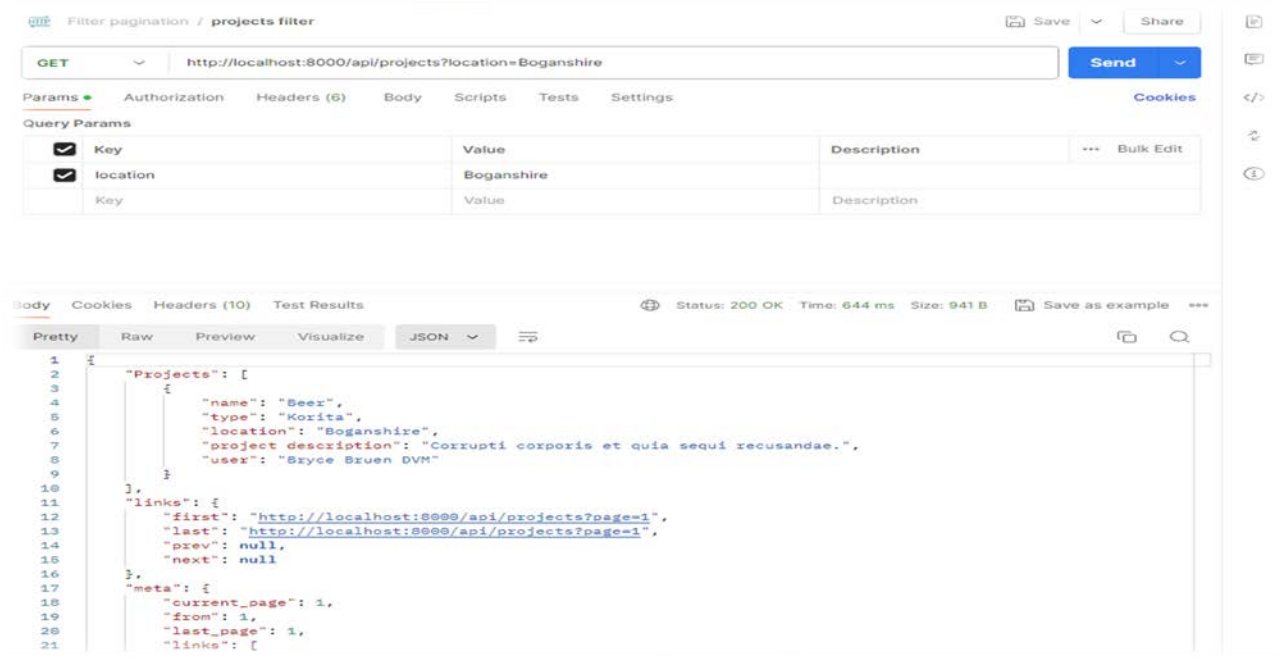


2. Korisnik šalje get zahtev ruti /api/projects koja mu ispisuje listu projekata. Projekti su prikazani po stranicama, gde se na svakoj stranici nalazi 10 projekata.

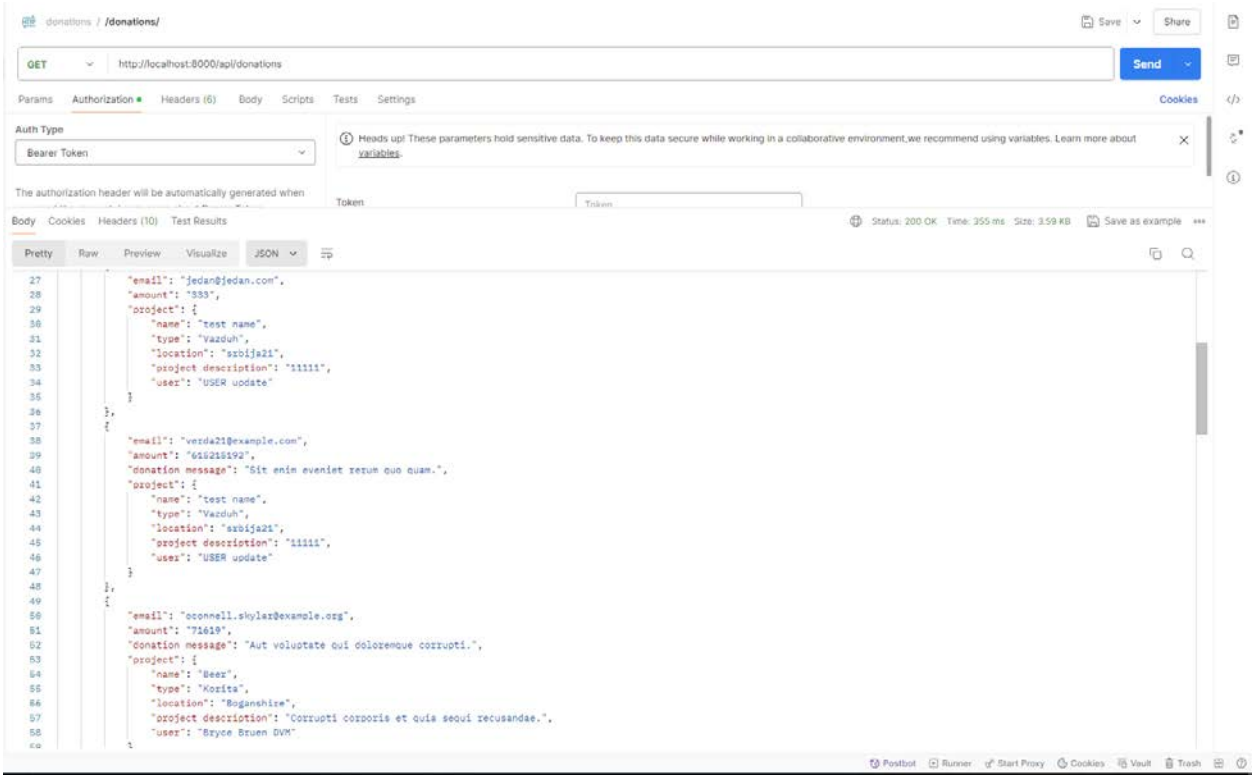


- Lista projekata se može filtrirati na osnovu tipa (type_id=X), lokacije (location=Y) ili kombinacije tipa i lokacije (type_id=X&location=Y).

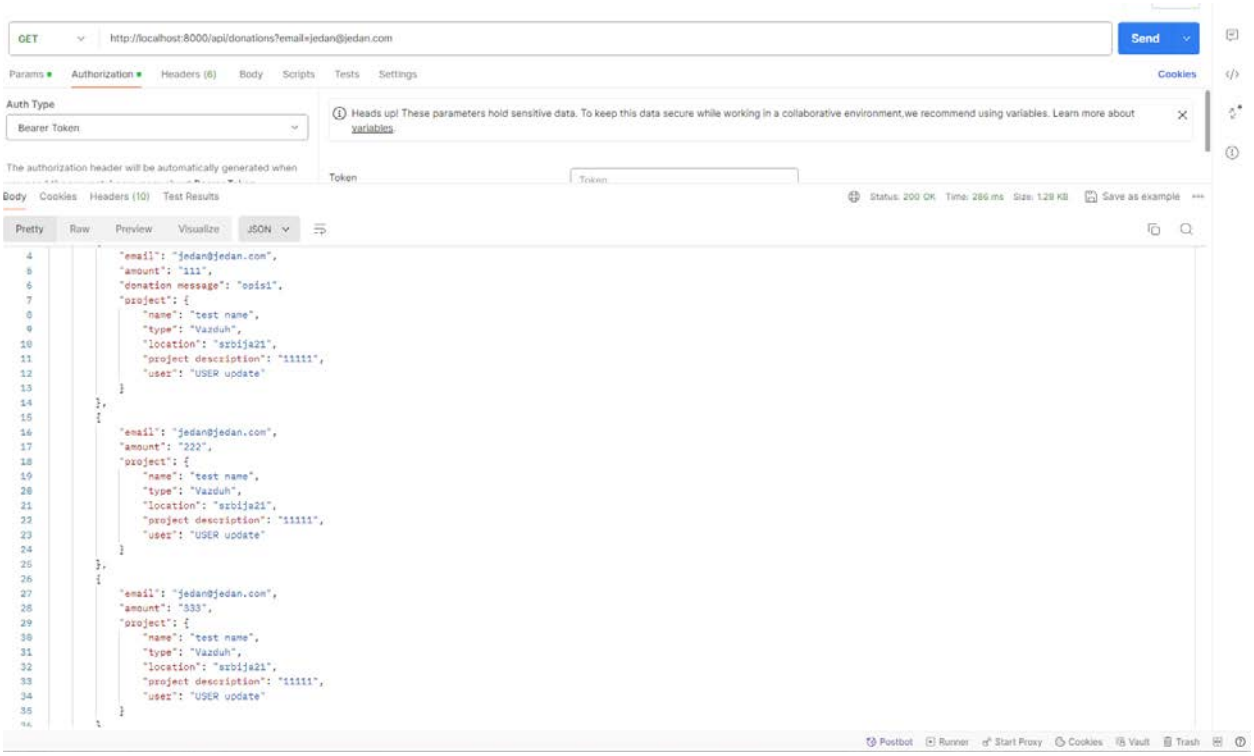




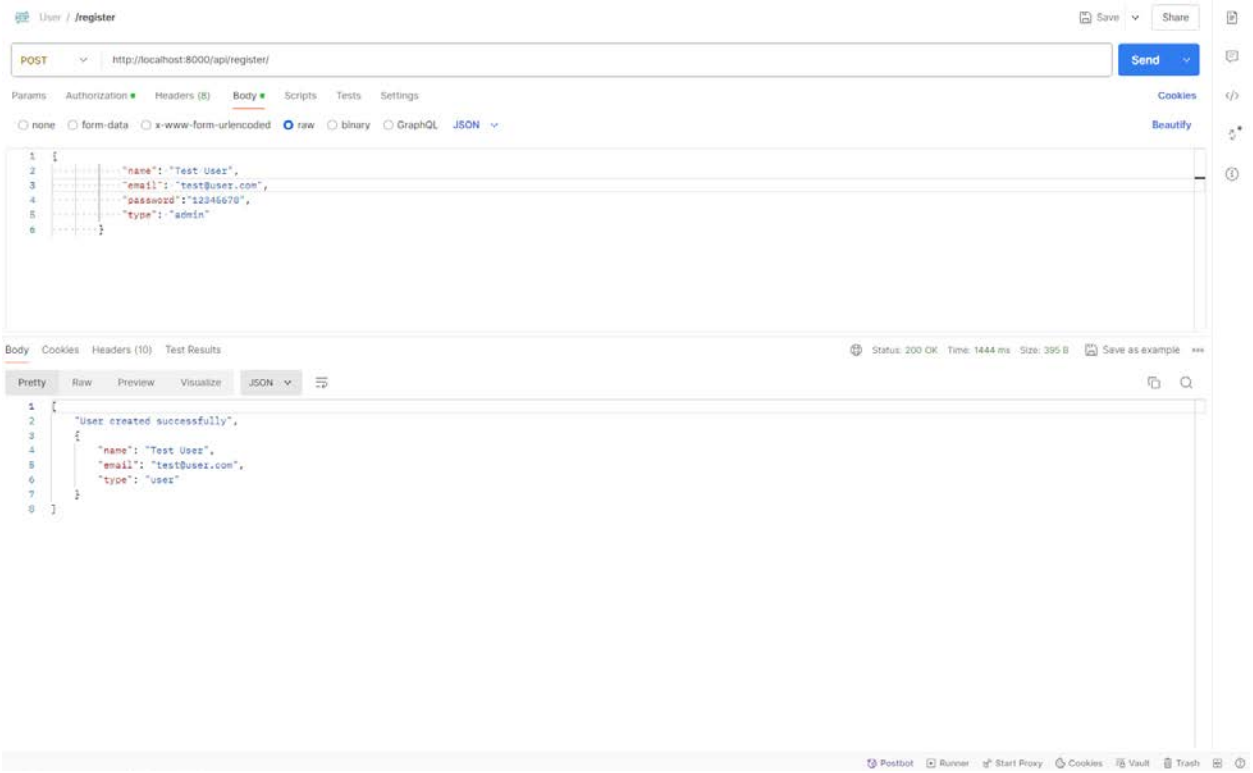
3. Posetilac pristupa ruti /api/donations koja mu ispisuje listu donacija. Donacije su prikazane po stranicama, gde se na svakoj stranici nalazi 10 projekata.



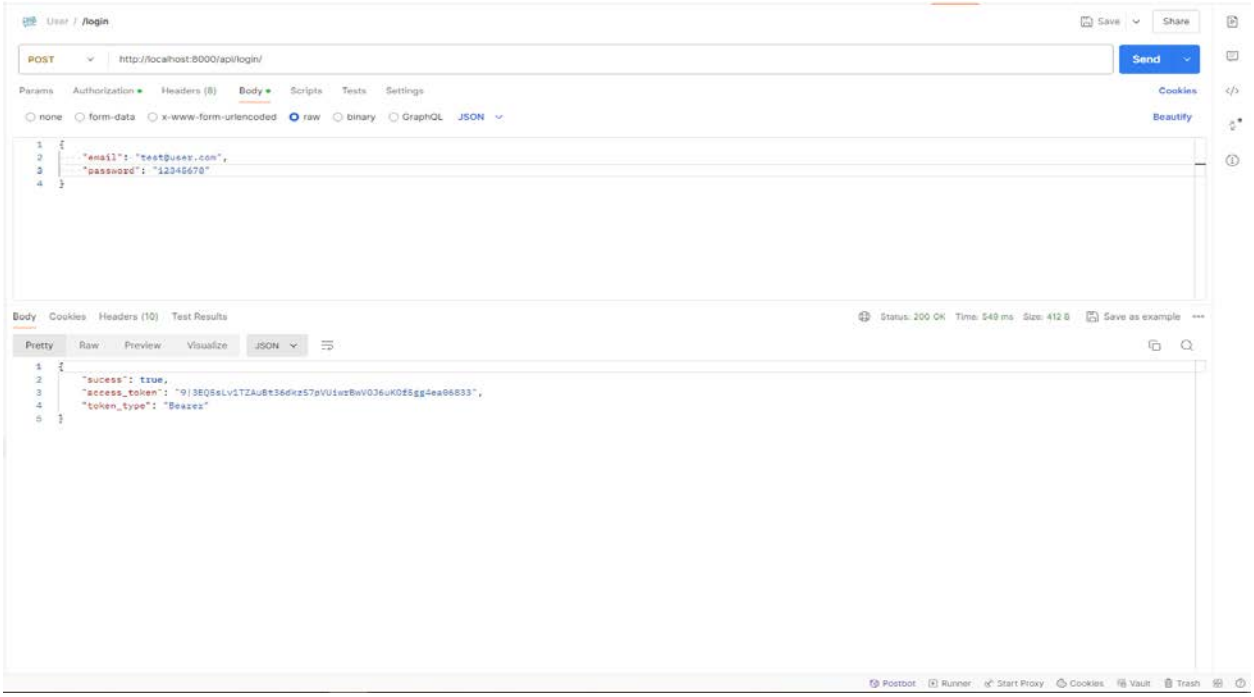
- Kao i kod projekata lista donacija se može filtrirati i to na osnovu email adrese donatora (email), projekta (project_id) ili kombinacije ova dva parametra.



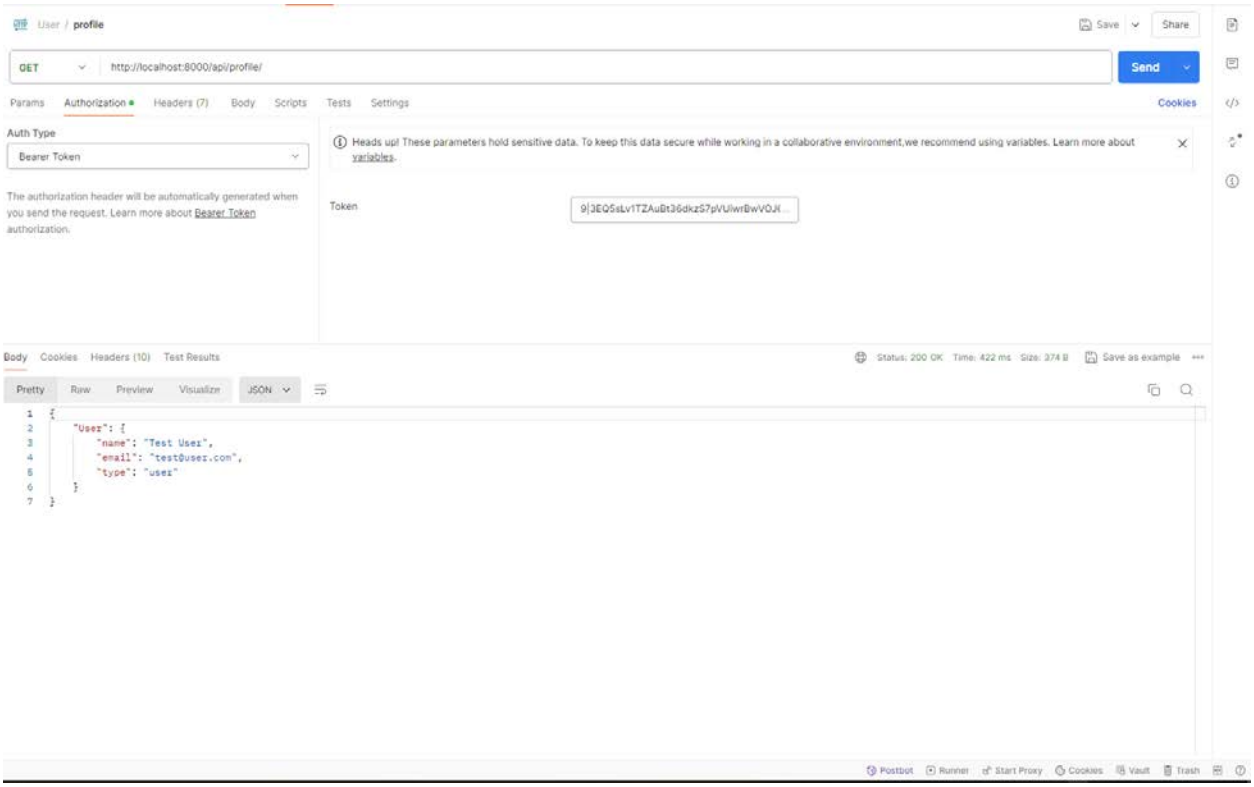
4. Kako bismo napravili novog korisnika potrebno je ruti /api/register poslati parametre novog korisnika. Parametri su: ime (name), imejl adresa (email) i šifra (password).
   Svaki korisnik na registrovan preko ove rute biće napravljen kao korisnik tipa “user” bez obzira da li je neki tip prosleđen kao parameter ili ne.



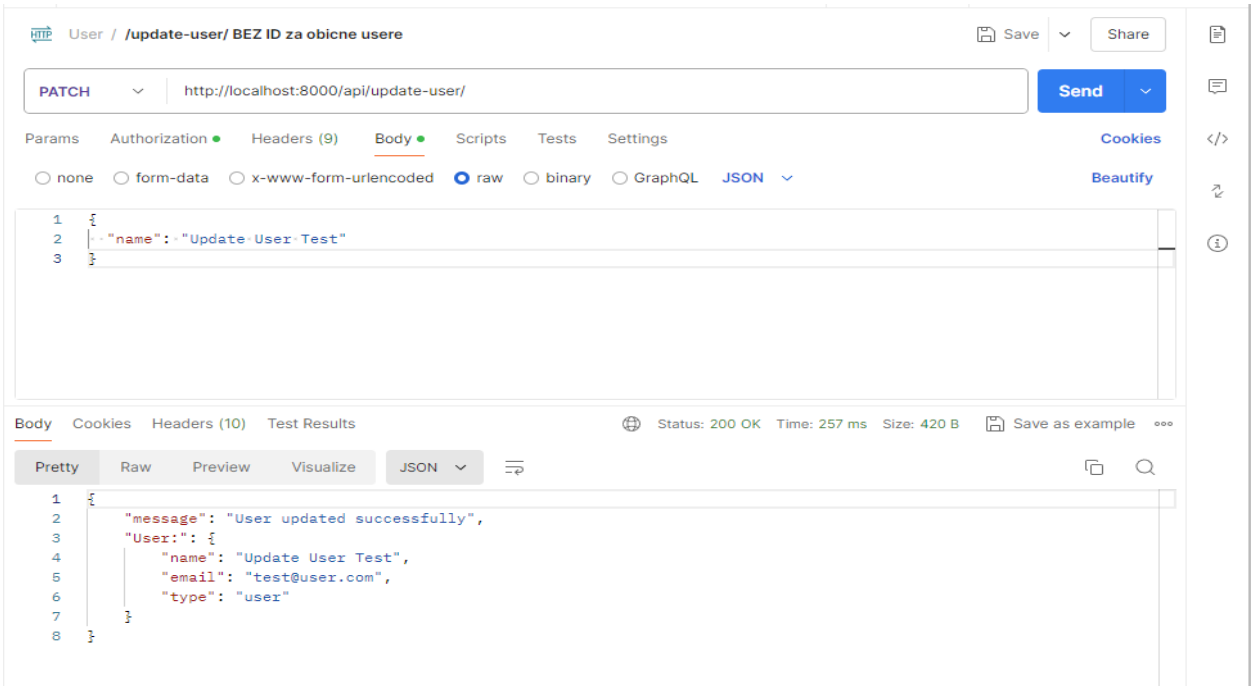
5. Da bismo se prijavili na sistem potrebno je poslati parametre imejl i šifru korisnika ruti /api/login. U slučaju uspešne prijave kao povratnu informaciju dobijamo odgovarajući token.



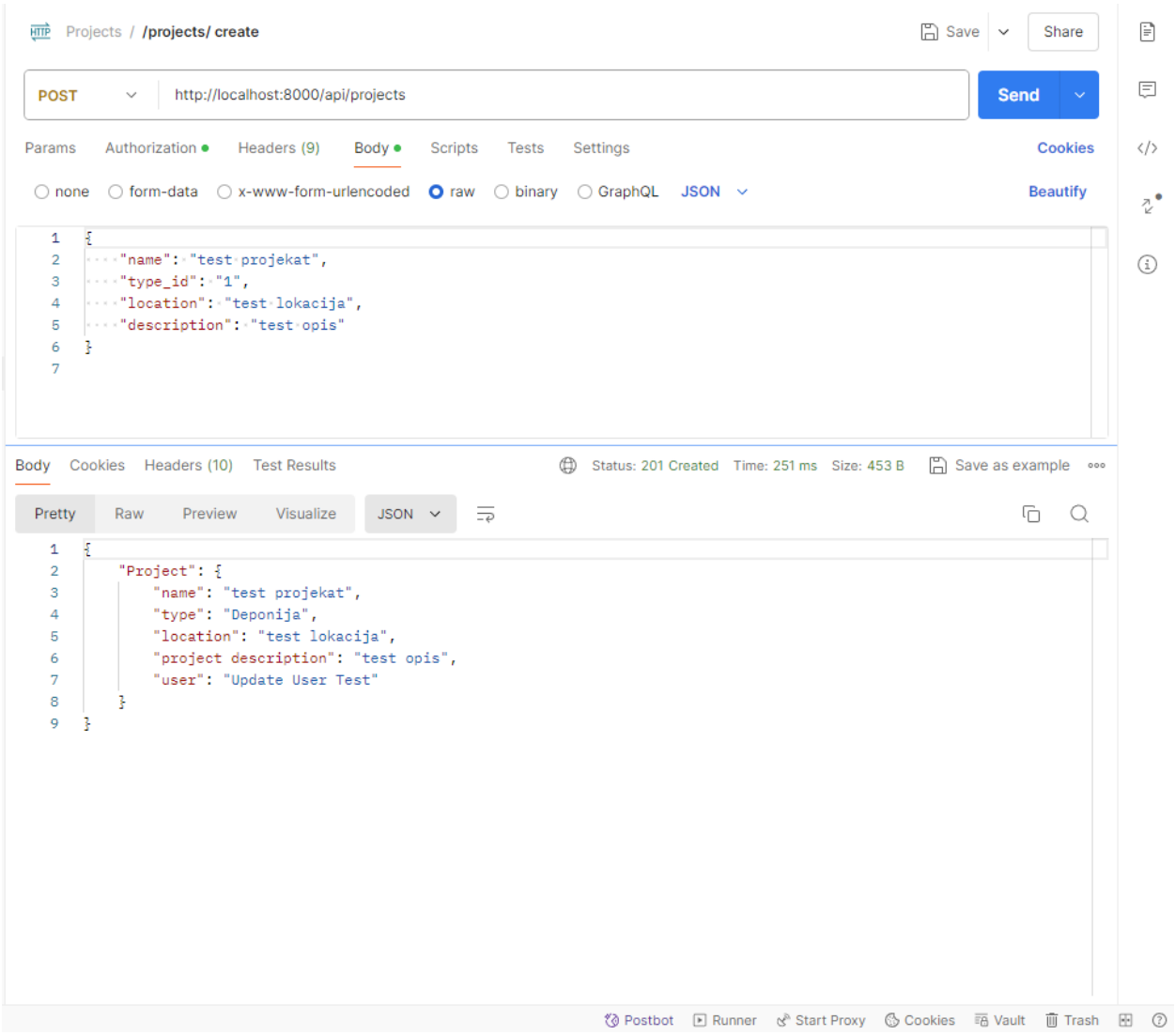
6. U koliko ruti /api/profile prosledimo token dobijen prijavom biće nam ispisana tri atributa trenutnog korisnika: ime, imejl adresa i tip korisnika.



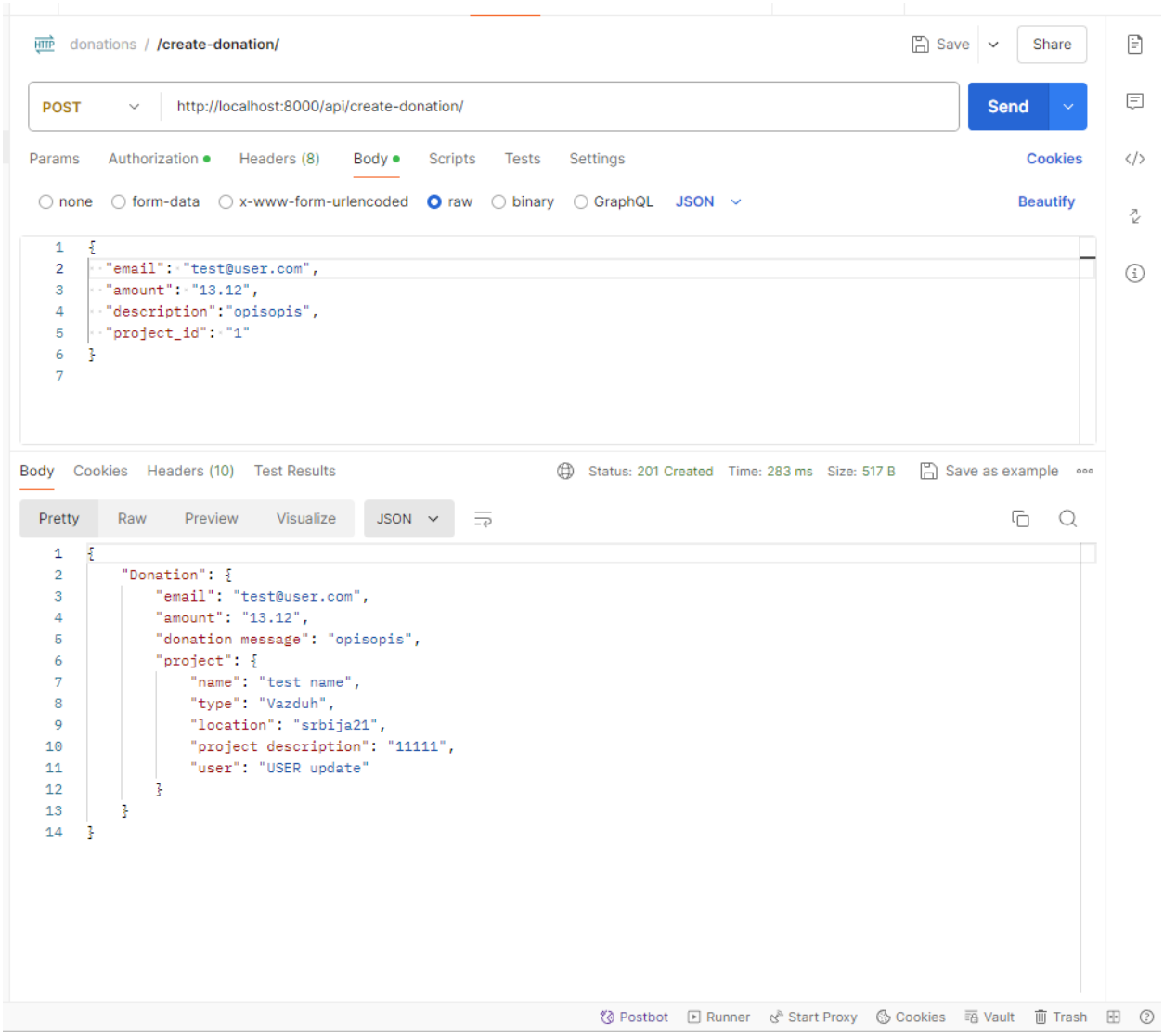
7. Na ruti /api/update-user/ možemo ažurirati određene atribute trenutnog korisnika slanjem patch zahteva. Da bismo to uradili potrebno je kao parametre poslati željene atribute i njihove ažurirane vrednosti.



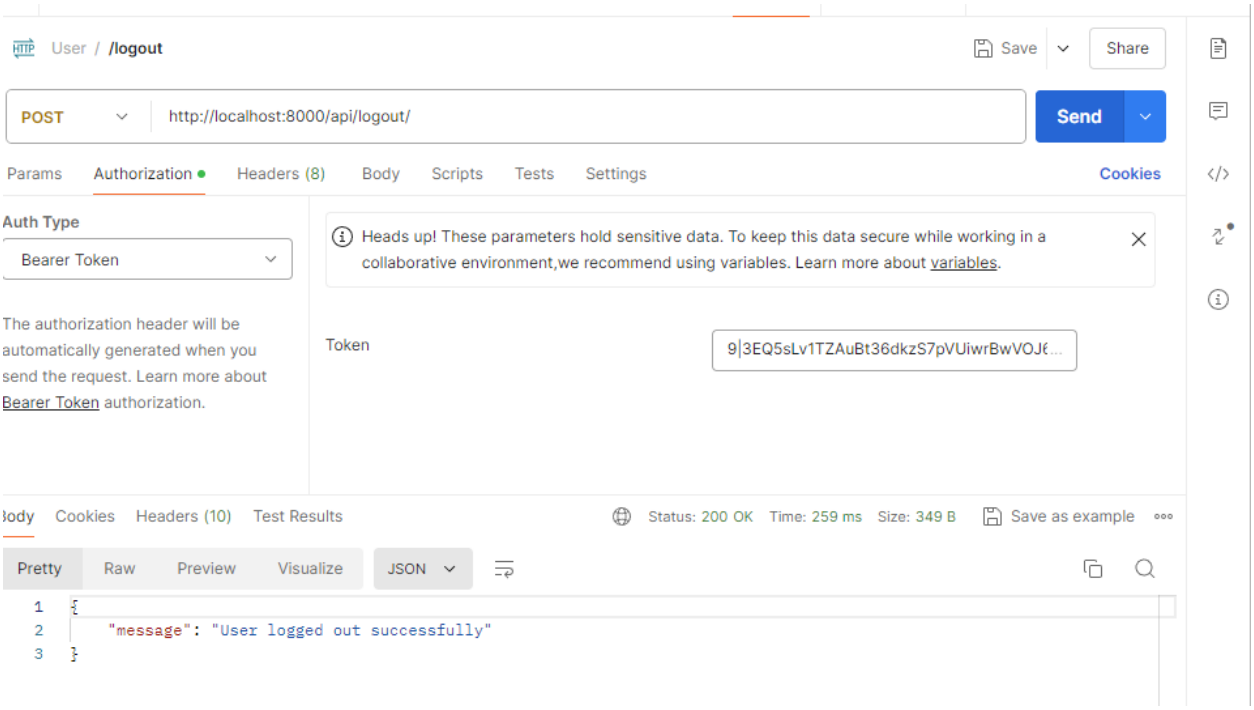
8. Kako bismo napravili novi projekat potrebno je ruti /api/projects poslati parametre novog projekta, kao i token dobijen prijavom. Parametri koji se šalju su ime, id tipa, lokacija i opcioni atribut opis. Pomoću tokena koji smo prosledili se utvrđuje ko je korisnik koji je napravio novi projekat.



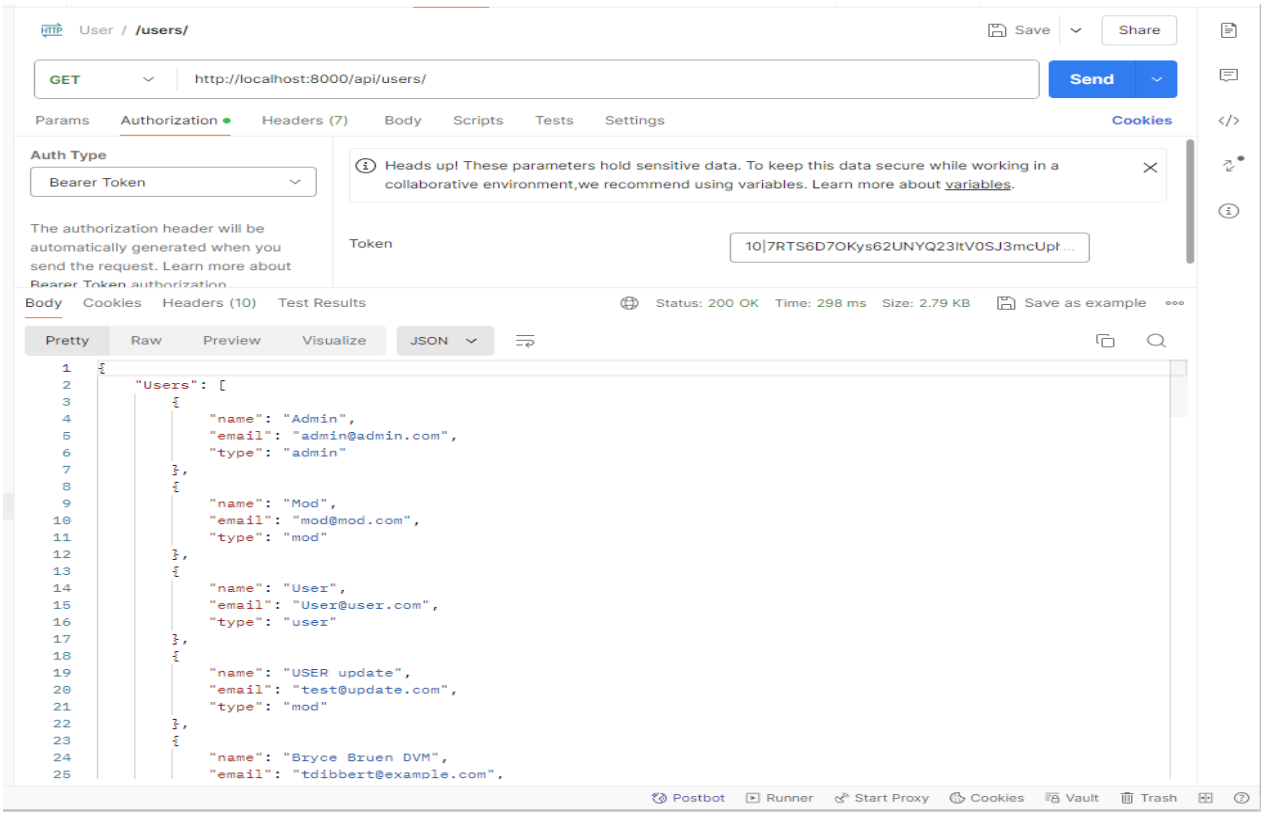
9. Da bismo napravili novu donaciju potrebno je ruti /api/create-donation proslediti parametre nove donacije. Za razliku od projekata, za kreiranje novih donacija nije potrebno biti prijavljen na sistem. Parametri koji se šalju su : imejl adresa donatora, iznos donacije, opis (opciono) i id projekta.



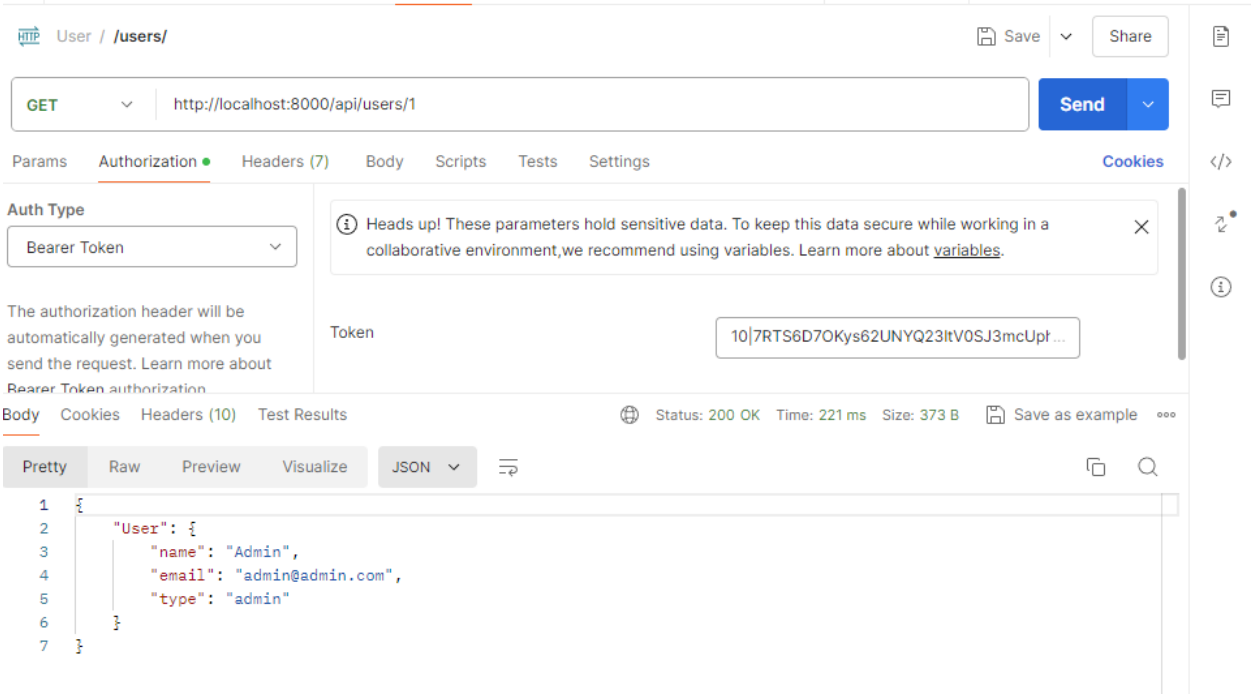
10. Slanjem zahteva, koji sadrži token, ruti /api/logout korisnik se odjavljuje sa sistema.



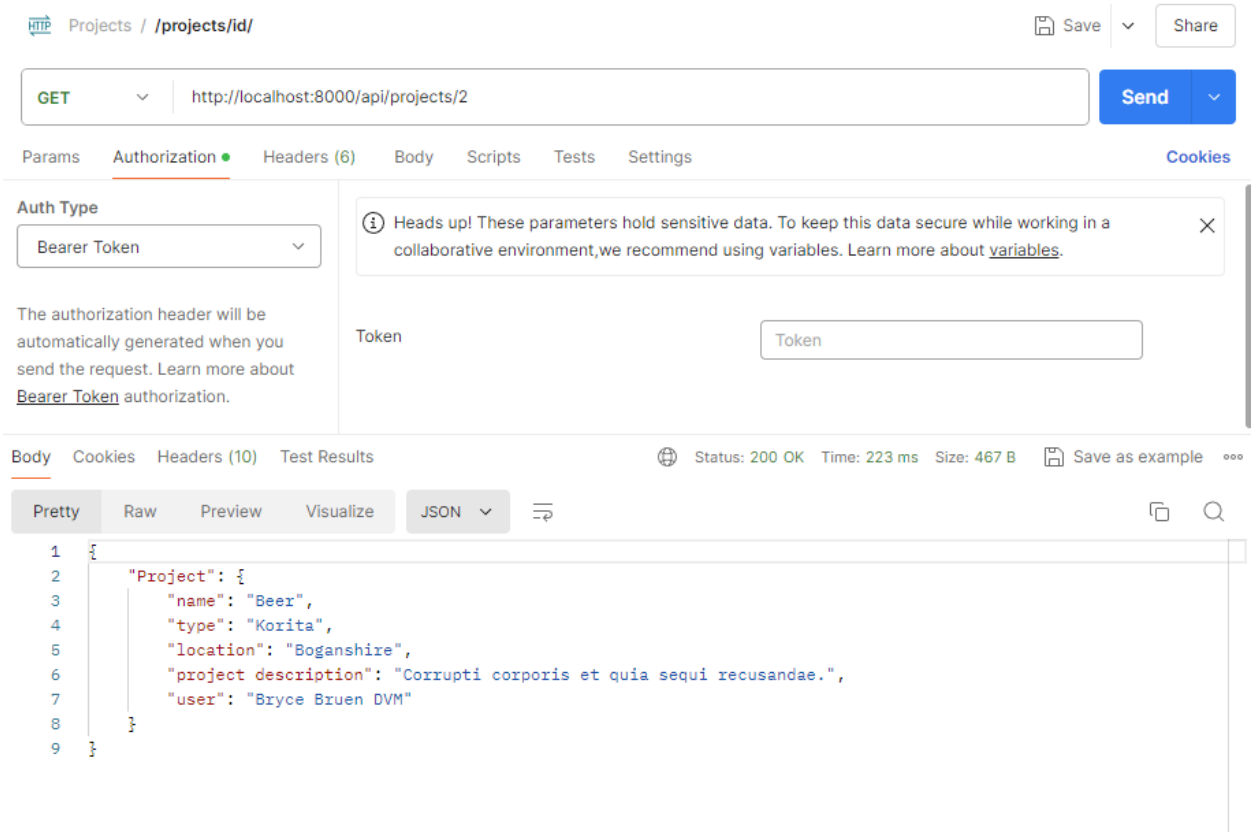
11. Pristupom ruti /api/users dobijamo listu svih korisnika. Ovoj ruti mogu pristupiti samo korisnici tipa “admin” ili “mod”. U zahtevu je potrebno proslediti i token za autorizaciju.



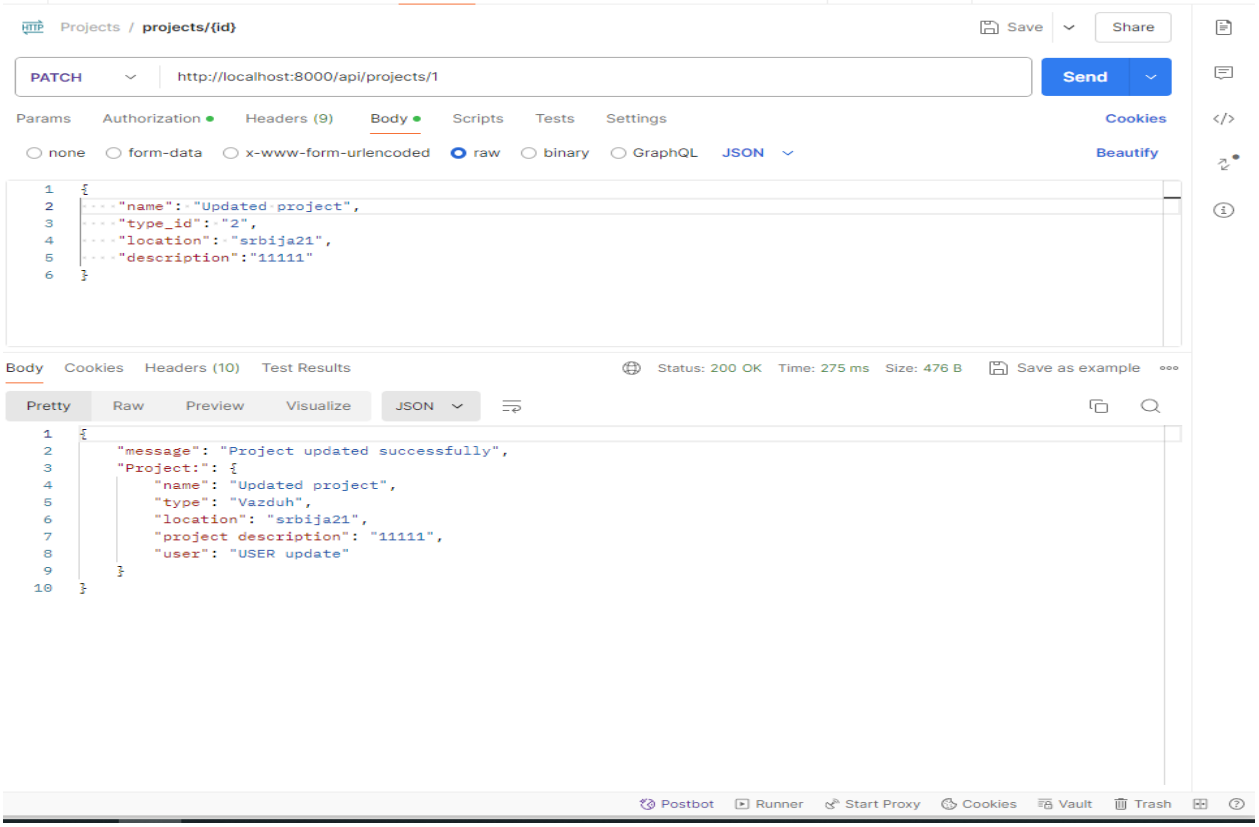
12. Prikazu pojedinačnog korisnika možemo pristupiti dodavanjem njegovog id-a na rutu /api/users/. Takođe, potrebno je da korisnik koji šalje zahtev bude tipa „admin“ ili „mod“.



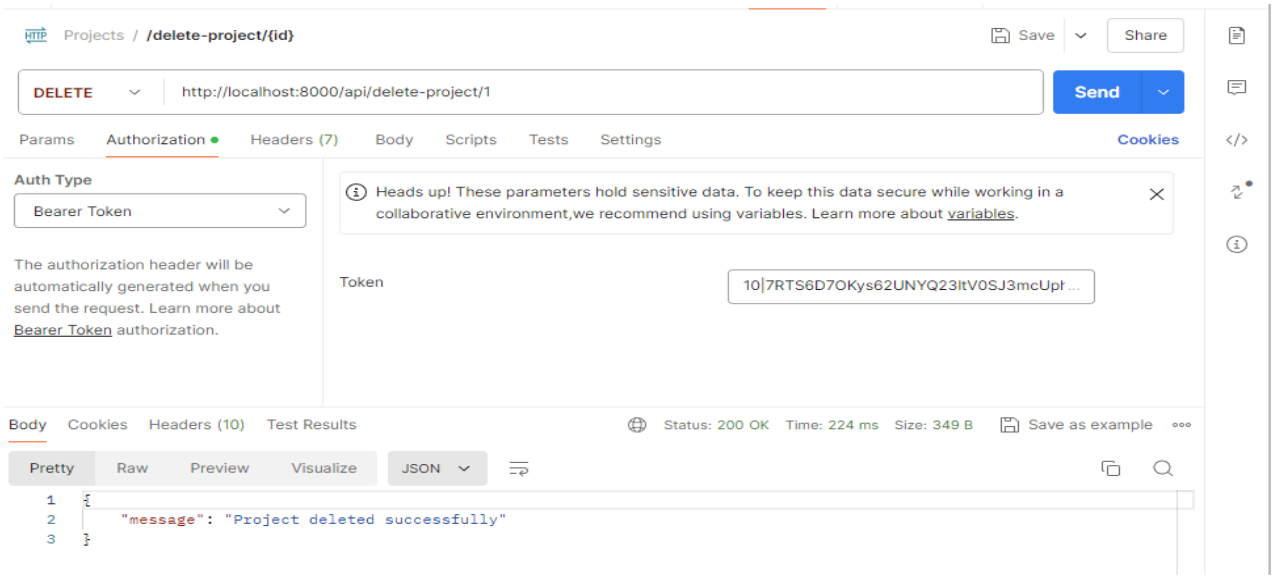
13. Prikazu pojedinačnog projekta mogu svi pristupiti dodavanjem id-a na kraj rute /api/projects.



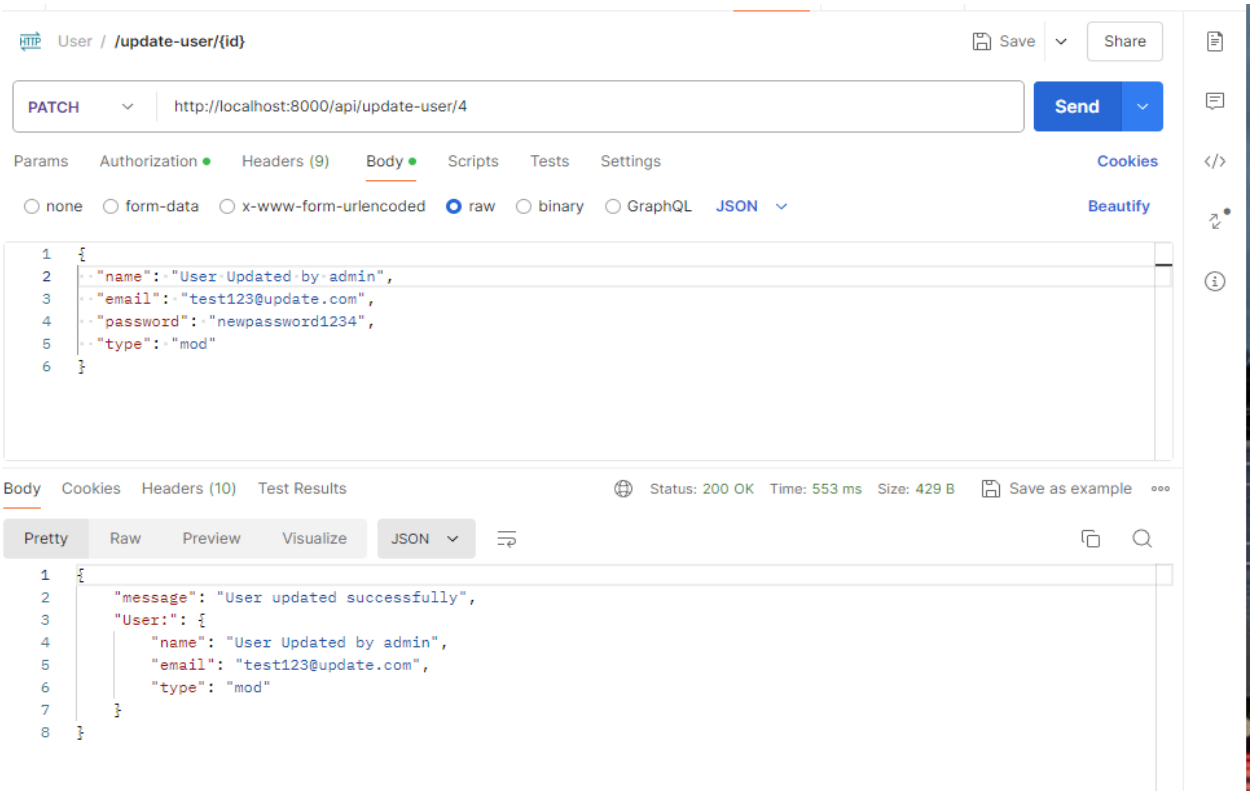
14. Ažuriranje atributa određenog projekta može se uraditi slanjem patch zahteva ruti /api/projects/ kojoj dodajemo id projekta koji ažuriramo.
    Kao parametre zahteva potrebno je poslati atribute koje ažuriramo, kao i njihove vrednosti.
    Preduslov za uspešno ažuriranje su da projekat sa prosleđenim id-em postoji, kao i da je korisnik koji šalje zahtev tipa „admin“ ili „mod“.



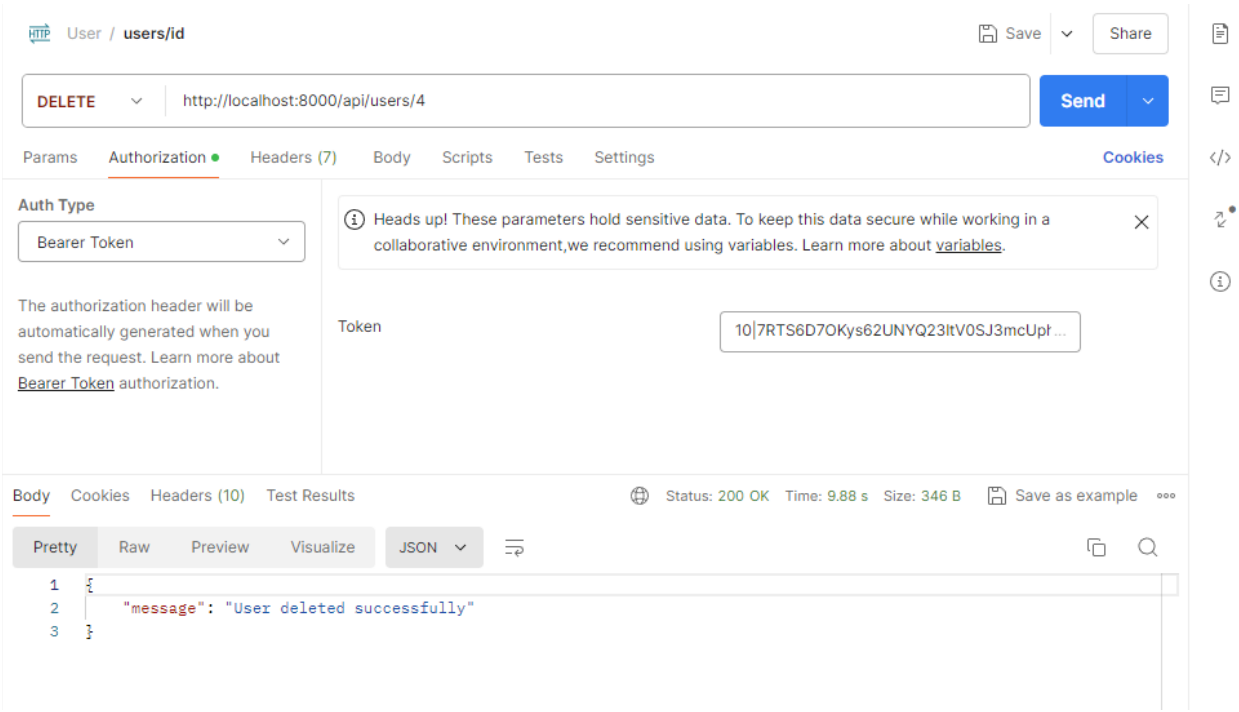
15. Slanjem delete zahteva ruti /api/delete-project kojoj je dodat id-a projekta korisnik briše odgovarajući projekat. Da bi uspešno obrisao projekat potrebno je da korisnik bude tipa „admin“ ili „mod“, kao i da postoji projekat sa tim id-em.



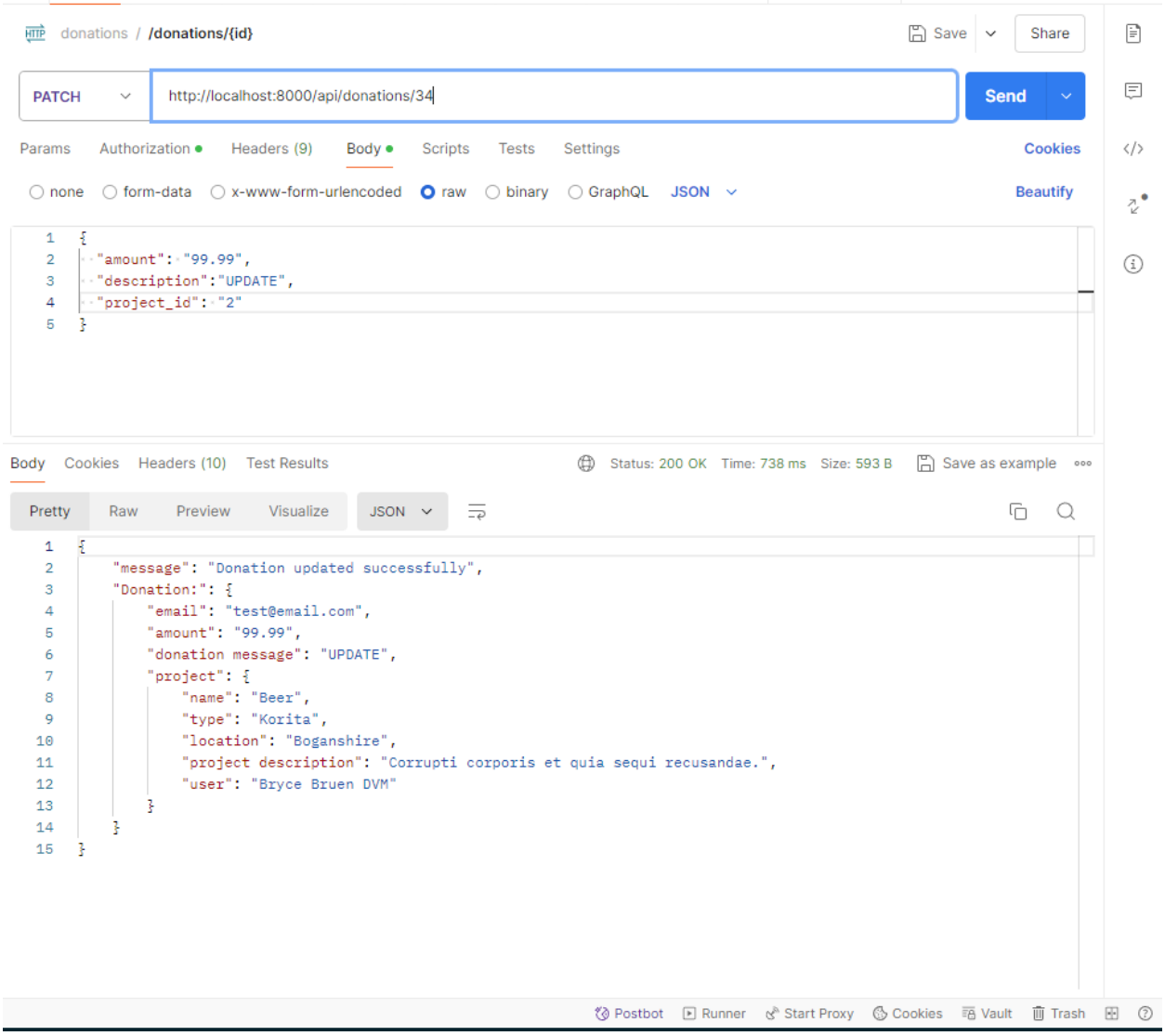
16. „Admin“ korisnici imaju mogućnost da ažuriraju atribute drugih korisnika slanjem patch zahteva. To se može uraditi slanjem atributa i vrednosti koje želimo da ažuriramo kao parametre ruti /api/update-user/ kojoj na kraj dodajemo id korisnika koji se ažurira.



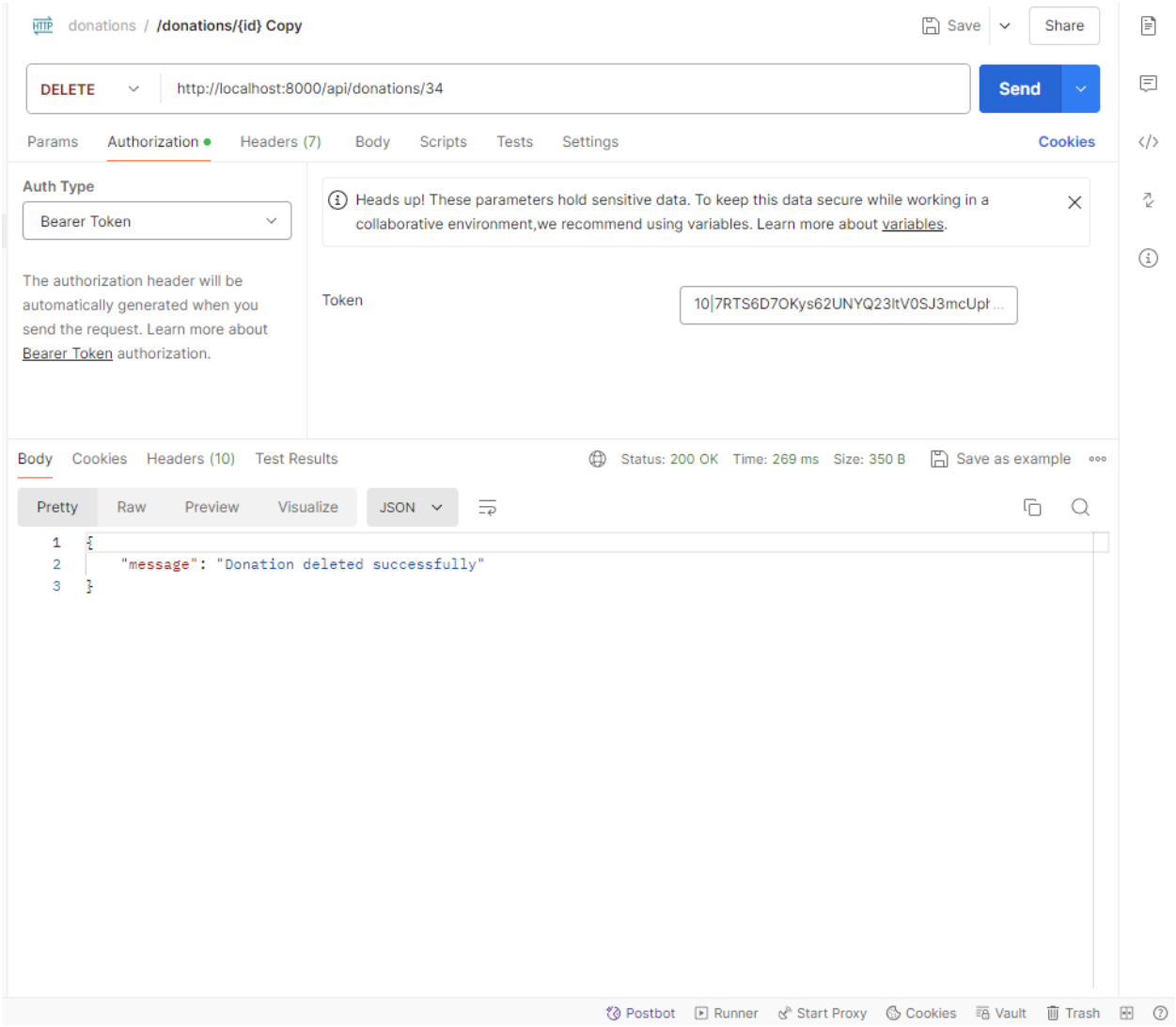
17. „Admin“ korisnici imaju mogućnost da obrišu korisnike slanjem delete zahteva ruti /api/users na koju se dodaje id korisnika kojeg želimo da obrišemo.



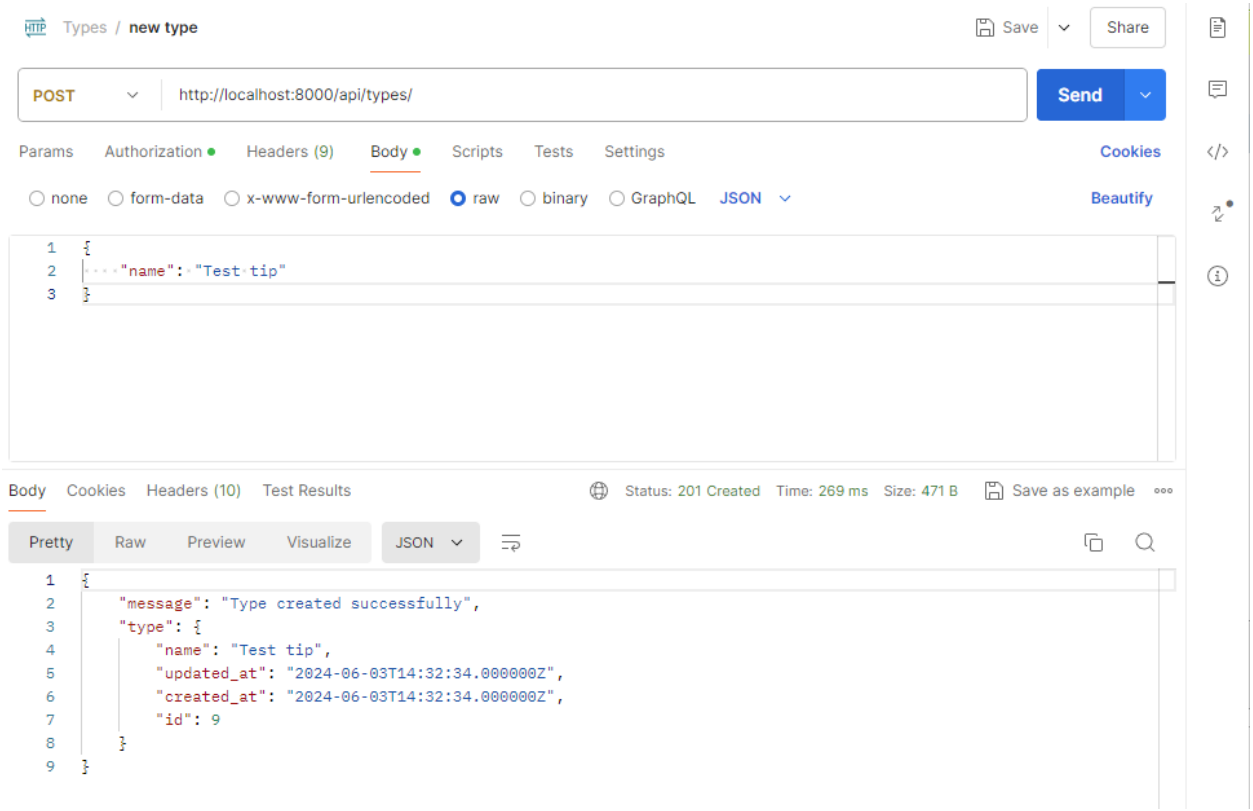
18. „Admin“ korisnici imaju mogućnost ažuriranja donacija dodavanjem njihovog id-a na kraj rute /api/donations. Kao parametri se šalju željeni atributi i njihove ažurirane vrednosti. Osim „admin“ tipa korisnika, preduslov je i da postoji donacija sa tim id-em.



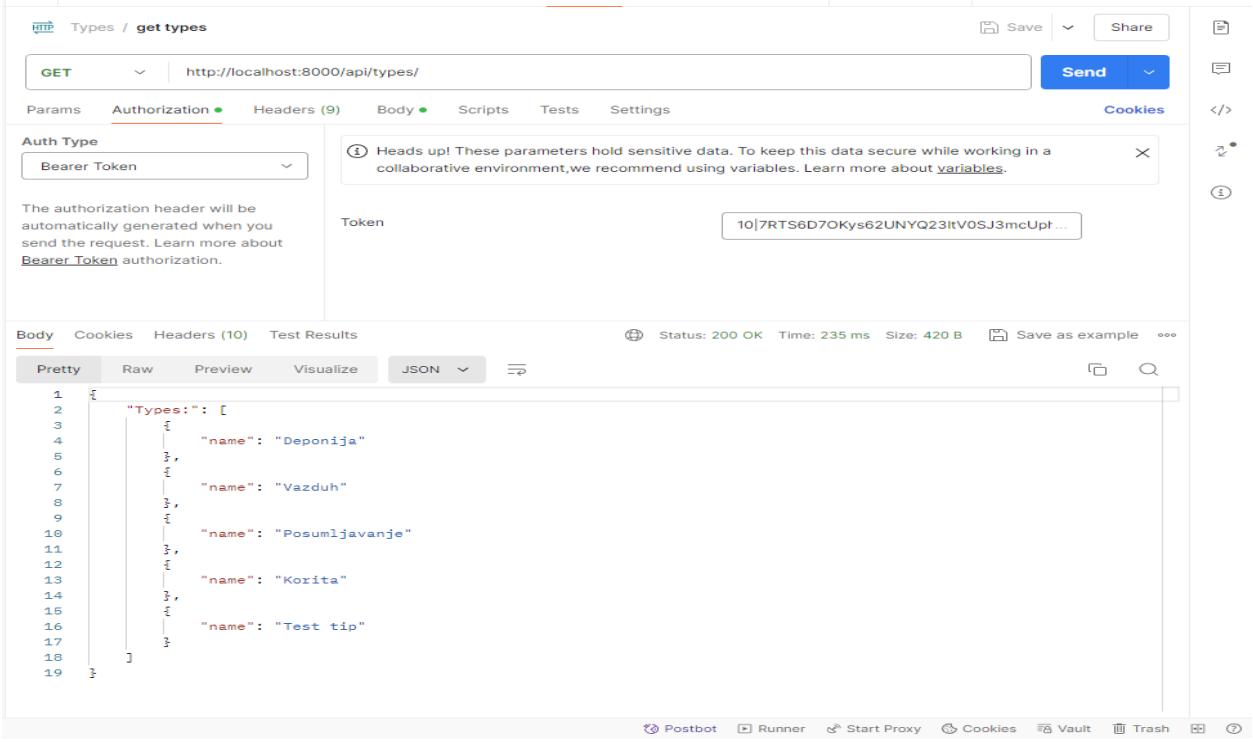
19. „Admin korisnici takođe imaju mogućnost brisanja donacija slanjem delete zahteva ruti /api/donations/ na čiji kraj je dodat id donacije koja se briše.
Preduslov je da donacija sa tim id-em postoji.



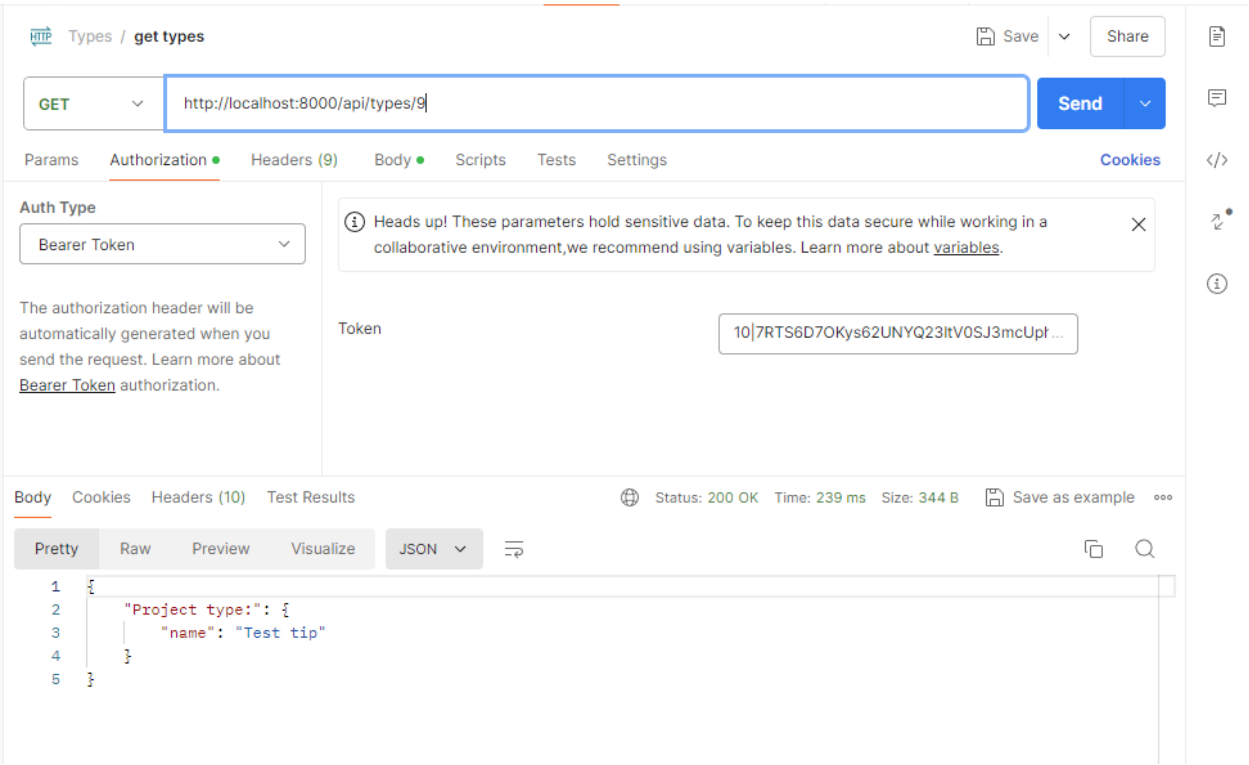
20. Kreiranje novog tipa projekta mogu da izvrše samo „Admin“ korisnici slanje post zahteva ruti /api/types. Od parametara je potrebno proslediti atribut „name“ i njegovu vrednost.



21. „Admin“ korisnici slanjem get zahteva ruti /api/types kao povratnu informaciju dobijaju listu mogućih tipova projekata.



22. „Admin“ korisnici slanjem get zahteva ruti /api/types/ kojoj je dodat id određenog tipa dobijaju pojedinačni prikaz tipa sa tim id-em.



23. „Admin“ korisnici slanjem delete zahteva ruti /api/types/ koja sadrži id određenog tipa, brišu odgovarajući tip projekta. Preduslov za to je da postoji tip sa tim id-em, kao i da ni jedan projekat nije povezan sa tim tipom.