

Univerzitet u Beogradu
Fakultet organizacionih nauka
Katedra za elektronsko poslovanje

Laravel

Domaći zadatak 1

Aleksandra Milenković 2020/0195

Jana Milošević 2020/0064

Link ka Github-u: https://github.com/elab-development/internet-tehnologije-projekat-googlekalendar_2020_0195.git

Beograd, 2022

Sadržaj

1.	Korisnički zahtev	3
1.1	Model	4
1.2	Migracije	4
1.3	Kontroleri	5
1.4	Api rute	9
2.	REST API	11
3.	Korisničko uputstvo.....	17

1. Korisnički zahtev

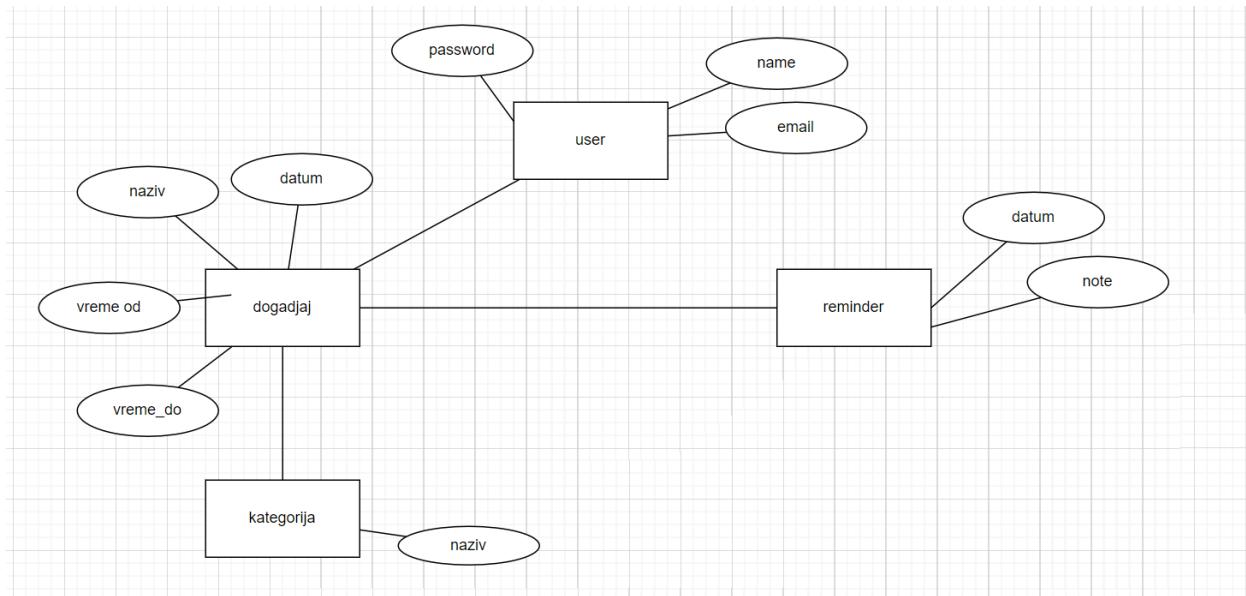
Potrebno je kreirati aplikaciju za planiranje svakodnevnog života po uzoru na GoogleCalendar. Korisnicima aplikacije je potrebno omogućiti registraciju, prijavu i odjavu na sistem. Korisnici koji su prijavljeni imaju mogućnost kreiranja događaja, za određen datum i određeni vremenski interval. Za svaki događaj se pamte osnovne informacije – naziv, opis, tip dogadjaja...

Kako bi korisnicima kalendar bio pregledniji potrebno je podeliti događaje po kategorijama (posao, škola, fakultet, hobi, zabava, porodica, prijatelji...). Korisnici imaju mogućnost modifikovanja kategorija po svom načinu života tj. moguće je dodavati, brisati i menjati kategorije.

Korisnici takođe imaju mogućnost i kreiranja podsetnika za događaje koji su im važni, kako bi se osigurali da ih ne propuste.

1.1 Model

PMOV model aplikacije i veze prikazane su na slici ispod.



Slika 1- Model aplikacije

1.2 Migracije

U ovoj aplikaciji su kreirane sledeće migracije:

- Migracija za kreiranje tabele korisnika (create_users_table)**: Ova migracija kreira tabelu **users** u bazi podataka sa kolonama za ID, ime, email (sa jedinstvenim ograničenjem), vreme verifikacije email-a, lozinku, i dva timestamp-a za praćenje kreiranja i ažuriranja zapisa korisnika.
- Migracija za kreiranje tabele događaja (create_dogadjajs_table)**: Kreira tabelu **dogadjajs** sa kolonama za ID, datum, vreme početka (**vreme_od**), vreme završetka (**vreme_do**), naziv, opis, status, ID kategorije (**kategorija_id**), ID korisnika (**user_id**), i timestamp-ove za kreiranje i ažuriranje zapisa.

3. **Migracija za kreiranje tabele kategorija (create_kategorijas_table)**: Ova migracija kreira tabelu **kategorijas** koja sadrži kolone za ID, naziv kategorije, i timestamp-ove za praćenje kreiranja i ažuriranja zapisa kategorija.
4. **Migracija za kreiranje tabele podsetnika (create_reminders_table)**: Kreira tabelu **reminders** sa kolonama za ID, naslov podsetnika, opis, datum i vreme kada podsetnik treba da se aktivira, ID korisnika (**user_id**), i timestamp-ove za kreiranje i ažuriranje zapisa.
5. **Migracija za dodavanje stranih ključeva tabeli događaja (add_foreign_keys_to_dogadjaji_table)**: Ova migracija dodaje strane ključeve **kategorija_id** i **user_id** u tabeli **dogadjajs**, povezujući je sa tabelama **kategorijas** i **users** respektivno.
6. **Migracija za dodavanje dodatnih kolona tabeli korisnika (add_extra_columns_users_table)**: Dodaje nove kolone tabeli **users**, koje mogu uključivati dodatne informacije o korisnicima, kao što su adresa, broj telefona ili bilo koji drugi relevantni podaci (detalji kolona nisu navedeni u pregledu).
7. **Migracija za činjenje kolone opis opcionom u tabeli događaja (make_opis_nullable_in_dogadjajs_table)**: Menja kolonu **opis** u tabeli **dogadjajs** tako da postane opcionalna, omogućavajući unos zapisa bez obavezne vrednosti za **opis**.
8. **Migracija za uklanjanje kolone detalji iz tabele događaja (remove_detalji_column_from_dogadjajs_table)**: Uklanja kolonu **detalji** iz tabele **dogadjajs**, što može biti korak u procesu normalizacije ili kao rezultat promene zahteva aplikacije.

1.3 Kontroleri

DogadjajController je ključni deo naše aplikacije. U njemu su implementirane sledeće metode:

- **Index** - Implementira keširanje za poboljšanje performansi prilikom učitavanja svih događaja. Upotrebom Cache::remember, metoda proverava da li su podaci već keširani pod ključem **dogadjaji_cache_key**. Ako nisu, izvršava upit u bazi podataka da dohvati sve događaje, kešira ih na 60 minuta, i vraća ih korisniku.
- **Store** - Omogućava kreiranje novog događaja validacijom zahteva koristeći Laravel Validator. Nakon uspešne validacije, događaj se kreira i odmah briše keš pod ključem **dogadjaji_cache_key** kako bi se osiguralo da se pri sledećem pozivu index metode vrati ažurirani podaci.

- **Show** - Vraća detalje specifičnog događaja na osnovu ID-a. Ako događaj sa prosleđenim ID-om postoji, vraća se kao odgovor; inače, izaziva se izuzetak.
- **Update** - Ažurira postojeći događaj validacijom podataka iz zahteva, a zatim ažurira događaj u bazi. Kao i u store metodi, keš se briše nakon ažuriranja kako bi se osvežili podaci.
- **Destroy** - Briše događaj na osnovu ID-a. Nakon brisanja, takođe briše keš kako bi se osigurao povratak ažurirane liste događaja pri sledećem pozivu index metode.
- **Search** - Omogućava pretragu događaja na osnovu više kriterijuma kao što su datum, naziv, status, i kategorija. Koristi dinamički upit zasnovan na prisutnosti parametara u zahtevu. Rezultati pretrage se ne keširaju u ovoj verziji kontrolera.

```

<?php

namespace App\Http\Controllers;

use App\Http\Resources\DogadjajResource;
use App\Models\Dogadjaj;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Validator;
use Illuminate\Support\Facades\Cache;

class DogadjajController extends Controller
{
    public function index() //prepravljena funkcija index kako bi se obezbila mogućnost kesiranja dogadjaja
    {

        /* remember metoda pokušava da preuzeme keširane podatke koristeći ključ 'dogadjaji_cache_key'.
        Ako podaci nisu dostupni u kešu, izvršava se anonima funkcija koja vraća sve događaje iz baze.
        Rezultat ove funkcije se kešira na 60 minuta (60*60 sekundi). */
        $dogadjaji = Cache::remember('dogadjaji_cache_key', 60*60, function () {
            return Dogadjaj::where('user_id', auth()->id())->get();
        });

        return DogadjajResource::collection($dogadjaji);
    }

    public function store(Request $request)
    {
        $validator = Validator::make($request->all(), [
            'datum' => 'required|date',
            'vreme_od' => 'required|date_format:H:i',
            'vreme_do' => 'required|date_format:H:i|after:vreme_od',
            'naziv' => 'required|string|max:255',
            'opis' => 'required|string',
            'status' => 'required|in:zavrseno,odlozeno,otkazano,u_toku,zakazano',
            'kategorija_id' => 'required|exists:kategorijas,id',
            'user_id' => 'required|exists:users,id',
        ]);

        if ($validator->fails()) {
            return response()->json($validator->errors(), 422);
        }

        $dogadjaj = Dogadjaj::create($validator->validated());
        Cache::forget('dogadjaji_cache_key'); //nakon azuriranja brisanja i dodavanja dogadjaja ćemo brisati keš

        return new DogadjajResource($dogadjaj);
    }

    public function show($id)
    {
        $dogadjaj = Dogadjaj::findOrFail($id);
        return new DogadjajResource($dogadjaj);
    }

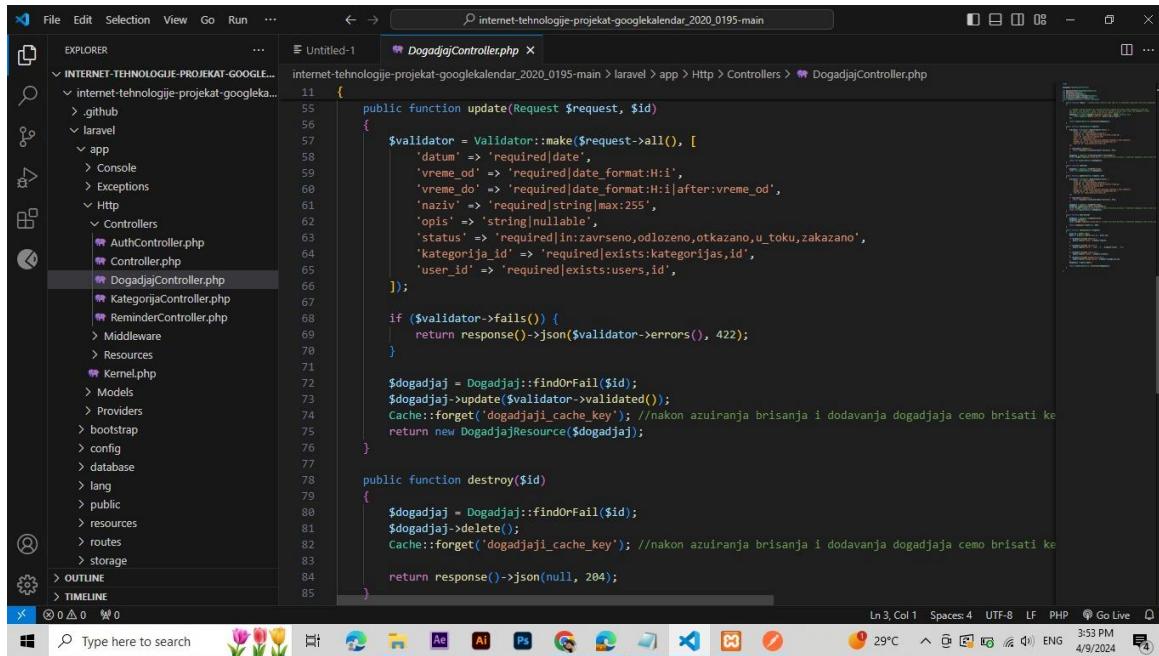
    public function update(Request $request, $id)
    {
    }
}

```

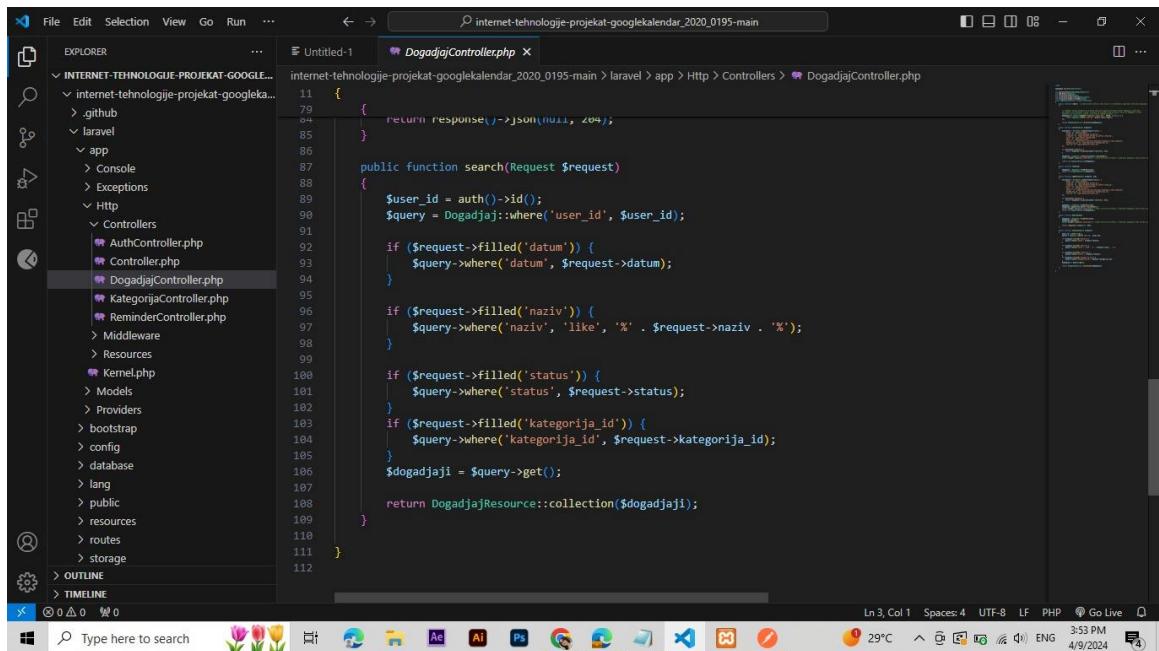
```

11 {
26     public function store(Request $request)
27     {
28         $validator = Validator::make($request->all(), [
29             'datum' => 'required|date',
30             'vreme_od' => 'required|date_format:H:i',
31             'vreme_do' => 'required|date_format:H:i|after:vreme_od',
32             'naziv' => 'required|string|max:255',
33             'opis' => 'required|string',
34             'status' => 'required|in:zavrseno,odlozeno,otkazano,u_toku,zakazano',
35             'kategorija_id' => 'required|exists:kategorijas,id',
36             'user_id' => 'required|exists:users,id',
37         ]);
38
39         if ($validator->fails()) {
40             return response()->json($validator->errors(), 422);
41         }
42
43         $dogadjaj = Dogadjaj::create($validator->validated());
44         Cache::forget('dogadjaji_cache_key'); //nakon azuriranja brisanja i dodavanja dogadjaja ćemo brisati keš
45
46         return new DogadjajResource($dogadjaj);
47     }
48
49     public function show($id)
50     {
51         $dogadjaj = Dogadjaj::findOrFail($id);
52         return new DogadjajResource($dogadjaj);
53     }
54
55     public function update(Request $request, $id)
56     {
57     }
}

```



```
11 {
12     public function update(Request $request, $id)
13     {
14         $validator = Validator::make($request->all(), [
15             'datum' => 'required|date',
16             'vreme_od' => 'required|date_format:H:i',
17             'vreme_do' => 'required|date_format:H:i|after:vreme_od',
18             'naziv' => 'required|string|max:255',
19             'opis' => 'string|nullable',
20             'status' => 'required|in:zavrseno,odlozeno,u_toku,zakazano',
21             'kategorija_id' => 'required|exists:kategorijas,id',
22             'user_id' => 'required|exists:users,id',
23         ]);
24
25         if ($validator->fails())
26         {
27             return response()->json($validator->errors(), 422);
28         }
29
30         $dogadjaj = Dogadjaj::findOrFail($id);
31         $dogadjaj->update($validator->validated());
32         Cache::forget('dogadjaji_cache_key'); //nakon azuriranja brisanja i dodavanja dogadjaja cemo brisati key
33         return new DogadjajResource($dogadjaj);
34     }
35
36     public function destroy($id)
37     {
38         $dogadjaj = Dogadjaj::findOrFail($id);
39         $dogadjaj->delete();
40         Cache::forget('dogadjaji_cache_key'); //nakon azuriranja brisanja i dodavanja dogadjaja cemo brisati key
41
42         return response()->json(null, 204);
43     }
44 }
```



```
11 {
12     public function search(Request $request)
13     {
14         $user_id = auth()->id();
15         $query = Dogadjaj::where('user_id', $user_id);
16
17         if ($request->filled('datum')) {
18             $query->where('datum', $request->datum);
19         }
20
21         if ($request->filled('naziv')) {
22             $query->where('naziv', 'like', '%' . $request->naziv . '%');
23         }
24
25         if ($request->filled('status')) {
26             $query->where('status', $request->status);
27         }
28         if ($request->filled('kategorija_id')) {
29             $query->where('kategorija_id', $request->kategorija_id);
30         }
31         $dogadjaji = $query->get();
32
33         return DogadjajResource::collection($dogadjaji);
34     }
35 }
```

Na sličan način su implementirani i ostali kontroleri. Izdvojićemo još i kontroler zadužen za **autentifikaciju**. U njemu su kreirane sledeće metode:

- **Register:** Omogućava registraciju novih korisnika. Prima podatke korisnika (ime, email, lozinka) preko HTTP zahteva, vrši validaciju ovih podataka, a zatim kreira novog korisnika u bazi podataka sa zahashovanom lozinkom. Nakon uspešne registracije, korisniku se izdaje personalizovani token (putem **createToken** metode), koji se koristi za autentifikaciju u budućim zahtevima. Odgovor uključuje podatke o korisniku i izdati token.

- **Login:** Procesira prijavu korisnika. Validira email i lozinku, a zatim koristi **Auth::attempt** za pokušaj prijave sa prosleđenim kredencijalima. Ako su kredencijali ispravni, korisniku se izdaje novi token i vraćaju se korisnički podaci. U slučaju neuspešne prijave, vraća se greška sa status kodom 401, označavajući neautorizovan pristup.
- **Logout:** Omogućava korisniku da se odjavi. Ova metoda briše sve token-e koje je korisnik generisao, efektivno ga odjavljujući sa svih uređaja ili sesija gde je bio autentifikovan. Vraća poruku o uspešnoj odjavi.

The screenshot shows two identical instances of a code editor (VS Code) side-by-side, both displaying the same PHP code for the `AuthController.php` file. The code handles user registration, login, and logout operations using Laravel's authentication facade.

```

1 <?php
2
3 namespace App\Http\Controllers;
4
5 use App\Http\Resources\UserResource;
6 use App\Models\User;
7 use Illuminate\Http\Request;
8 use Illuminate\Support\Facades\Auth;
9 use Illuminate\Support\Facades\Hash;
10 use Illuminate\Support\Facades\Validator;
11 use Laravel\Sanctum\PersonalAccessToken;
12
13 class AuthController extends Controller
14 {
15     public function register(Request $request)
16     {
17         $validator = Validator::make($request->all(), [
18             'name' => 'required|string|max:255',
19             'email' => 'required|string|email|max:255|unique:users',
20             'password' => 'required|string|min:8',
21         ]);
22
23         if ($validator->fails()) {
24             return response()->json($validator->errors(), 422);
25         }
26
27         $user = User::create([
28             'name' => $request->name,
29             'email' => $request->email,
30             'password' => Hash::make($request->password),
31         ]);
32
33     }
34
35     public function login(Request $request)
36     {
37         $validator = Validator::make($request->all(), [
38             'email' => 'required|string|email',
39             'password' => 'required|string',
40         ]);
41
42         if ($validator->fails()) {
43             return response()->json($validator->errors(), 422);
44         }
45
46         if (!Auth::attempt(['email' => $request->email, 'password' => $request->password])) {
47             return response()->json(['message' => 'Unauthorized'], 401);
48         }
49
50         $user = User::where('email', $request->email)->firstOrFail();
51         $token = $user->createToken('authToken')->plainTextToken;
52
53         return response()->json(['user' => new UserResource($user), 'token' => $token, 'message' => 'Registration successful']);
54
55     }
56
57     public function logout(Request $request)
58     {
59
60     }
61

```

The code editor interface includes a sidebar with project navigation, a central code editor area, and a bottom status bar showing file details like line and column numbers, encoding, and file type.

```
14
15     if ($validator->fails()) {
16         return response()->json(['message' => 'Unauthorized'], 401);
17     }
18
19     $user = User::where('email', $request->email)->firstOrFail();
20     $token = $user->createToken('authToken')->plainTextToken();
21
22     return response()->json(['user' => new UserResource($user), 'token' => $token]);
23 }
24
25 public function logout(Request $request)
26 {
27     $request->user()->tokens()->delete();
28     return response()->json(['message' => 'Uspešno izlogovano']);
29 }
```

1.4 Api rute

1. Registracija i prijava korisnika:

- **POST /register** ruta koristi **AuthController** klasu i njenu **register** metodu za kreiranje novog korisnika. Očekuje podatke kao što su ime, email, lozinka itd., koje korisnik treba da pošalje kako bi se registrovao.
- **POST /login** ruta takođe koristi **AuthController**, ali ovog puta za prijavljivanje korisnika. Korisnik šalje svoje kredencijale (email i lozinka), a sistem, ako su kredencijali ispravni, vraća token za autentifikaciju.

2. Rad sa događajima:

- Rute vezane za **događaji** su grupisane pod **auth:sanctum** middleware, što znači da je za pristup potrebno biti autentifikovan. **apiResource** pruža standardne RESTful rute (**index, store, show, update, destroy**) za rad sa događajima koristeći **DogadjajController**.

3. Odjava:

- **POST /logout** ruta koristi **AuthController** i njegovu **logout** metodu za odjavljivanje korisnika. Briše token koji se koristi za autentifikaciju sesije.

4. Upravljanje podsetnicima (reminders):

- Unutar **auth:sanctum** middleware grupe, definisane su rute za rad sa podsetnicima koristeći **ReminderController**. Ovo uključuje listanje svih podsetnika (**GET /reminders**), prikaz, kreiranje, ažuriranje i brisanje podsetnika po ID-u. Ove operacije zahtevaju autentifikaciju.

5. Upravljanje kategorijama:

- **apiResource za kategorije** omogućava rad sa kategorijama kroz standardne RESTful rute, koristeći **KategorijaController** unutar **auth:sanctum** grupe. To znači da je za pristup ovim rutama potrebno biti autentifikovan.

6. Pretraga događaja:

- Dodatna funkcionalnost za pretragu događaja omogućena je kroz **GET /dogadjaji/search** rutu koja koristi **DogadjajController** i njegovu **search** metodu. Ova ruta omogućava filtriranje i pretragu događaja na osnovu različitih parametara i takođe zahteva autentifikaciju korisnika.

2. REST API

Opis funkcije	Prijava korisnika
HTTP metoda	POST
URL	/api/login
URL parametri	email=jana@gmail.com&password=jana
HTTP body parametri	(nema)
Format HTTP body parametara	(nema)
Izlazni parametri	<pre>{ "user": { "id": 1, "name": "Jana", "email": "jana@gmail.com", "email_verified_at": null, "time_zone": "UTC", "language": "en", "calendar_color_theme": null, "start_week_on": "Monday", "created_at": "2024-03-22T14:00:20.000000Z", "updated_at": "2024-03-22T14:00:20.000000Z" }, "token": "1 VRHpNKOZXqdx8WvW2NgiT5yaQ8ZHmohW44I4YOFXb3f66ce1" }</pre>
Format izlaznih parametara	application/json
Opis funkcije	Prikaz svih dogadjaja korsinika
HTTP metoda	GET
URL	/api/dogadjaji
URL parametri	nema
HTTP body parametri	nema
Format HTTP body parametara	JSON
Izlazni parametri	<pre>{ "data": [{ "id": 1, "datum": "1986-01-05", "vreme_od": "15:02:07", "vreme_do": "15:02:07" }] }</pre>

	<pre> "vreme_do": "07:20:56", "naziv": "Voluptatem non perferendis non vero ipsam doloribus.", "opis": "Harum voluptas quaerat possimus quam nulla eveniet. Labore a voluptates quia eos perspiciatis ex fugiat. Modi vitae ducimus repellat doloremque omnis doloremque odio. Deleniti error dolorem molestias iste quam ipsum sint. Aspernatur illo enim magnam tenetur a.", "status": "u_toku" }, { "id": 2, "datum": "1970-04-26", "vreme_od": "11:29:16", "vreme_do": "14:16:08", "naziv": "Officia dolore quibusdam dolores a ea.", "opis": "Mollitia et voluptatem consequuntur quas hic omnis soluta ut. Ipsam optio veniam sed aut debitis distinctio. Ut sed nulla corporis neque.", "status": "odlozeno" }, .. </pre>
Format izlaznih parametara	Application/json
Opis funkcije	Kreiranje dogadjaja
HTTP metoda	POST
URL	/api/dogadjaji
URL parametri	datum=2024-03-26&vreme_od=11:29&vreme_do=14:16&naziv=dogadjaj1&opis=opis_dogadjaja1&status=odlozeno&kategorija_id=1&user_id=1
HTTP body parametri	(nema)
Format HTTP body parametara	(nema)
Izlazni parametri	<pre> { "data": { "id": 71, "datum": "2024-03-26", "vreme_od": "11:29", "vreme_do": "14:16", "naziv": "dogadjaj1", "opis": "opis dogadjaja1", </pre>

	<pre> "status": "odlozeno" } } </pre>
Format izlaznih parametara	application/json

Opis funkcije	Ažuriranje dogadjaja
HTTP metoda	PUT
URL	/api/ dogadjaji
URL parametri	71?datum=2024-03-27&vreme_od=11:29&vreme_do=14:16&naziv=dogadjaj1&opis=opis_dogadjaj1&status=odlozeno&kategorija_id=1&user_id=1
HTTP body parametri	(nema)
Format HTTP body parametara	(nema)
Izlazni parametri	<pre> { "data": { "id": 71, "datum": "2024-03-27", "vreme_od": "11:29", "vreme_do": "14:16", "naziv": "dogadjaj1", "opis": "opis dogadjaj1", "status": "odlozeno" } } </pre>
Format izlaznih parametara	application/json

Opis funkcije	Brisanje dogadjaja
HTTP metoda	DELETE
URL	/api/ dogadjaji
URL parametri	71
HTTP body parametri	(nema)
Format HTTP body parametara	(nema)
Izlazni parametri	-no content-
Format izlaznih parametara	application/json

Opis funkcije	Prikaz svih podsetnika
HTTP metoda	GET
URL	/api/ reminders
URL parametri	nema
HTTP body parametri	nema
Format HTTP body parametara	JSON
Izlazni parametri	<pre>{ "data": [{ "id": 1, "reminder_time": "2024-04-03 03:27:51", "note": "Rerum vitae ea ut maiores aut consequatur." }, { "id": 2, "reminder_time": "2024-03-05 03:09:05", "note": "Quis aliquam et est ab voluptas." }, ...] }</pre>
Format izlaznih parametara	Application/json
Opis funkcije	Kreiranje podsetnika
HTTP metoda	POST

URL	/api/ reminders
URL parametri	dogadjaj_id=1&reminder_time=2024-04-23 03:27:51
HTTP body parametri	(nema)
Format HTTP body parametara	(nema)
Izlazni parametri	<pre>{ "data": { "id": 351, "reminder_time": "2024-04-23 03:27:51", "note": null } }</pre>
Format izlaznih parametara	application/json

Opis funkcije	Ažuriranje podsetnika
HTTP metoda	PUT
URL	/api/ reminders
URL parametri	351?dogadjaj_id=1&reminder_time=2024-04-28 03:27:51
HTTP body parametri	(nema)
Format HTTP body parametara	(nema)
Izlazni parametri	<pre>{ "data": { "id": 351, "reminder_time": "2024-04-28 03:27:51", "note": null } }</pre>
Format izlaznih parametara	application/json

Opis funkcije	Brisanje podsetnika
HTTP metoda	DELETE
URL	/api/ reminders
URL parametri	351
HTTP body parametri	(nema)
Format HTTP body parametara	(nema)
Izlazni parametri	-no content-
Format izlaznih parametara	application/json

Opis funkcije	Odjava korisnika
HTTP metoda	POST
URL	/api/logout
URL parametri	nema
HTTP body parametri	(nema)
Format HTTP body parametara	(nema)
Izlazni parametri	{ "message": "Successfully logged out" }
Format izlaznih parametara	application/json

3. Korisničko uputstvo

Kako bi korisnici mogli da koriste našu aplikaciju, potrebno je da se prvo registruju, a zatim uloguju.

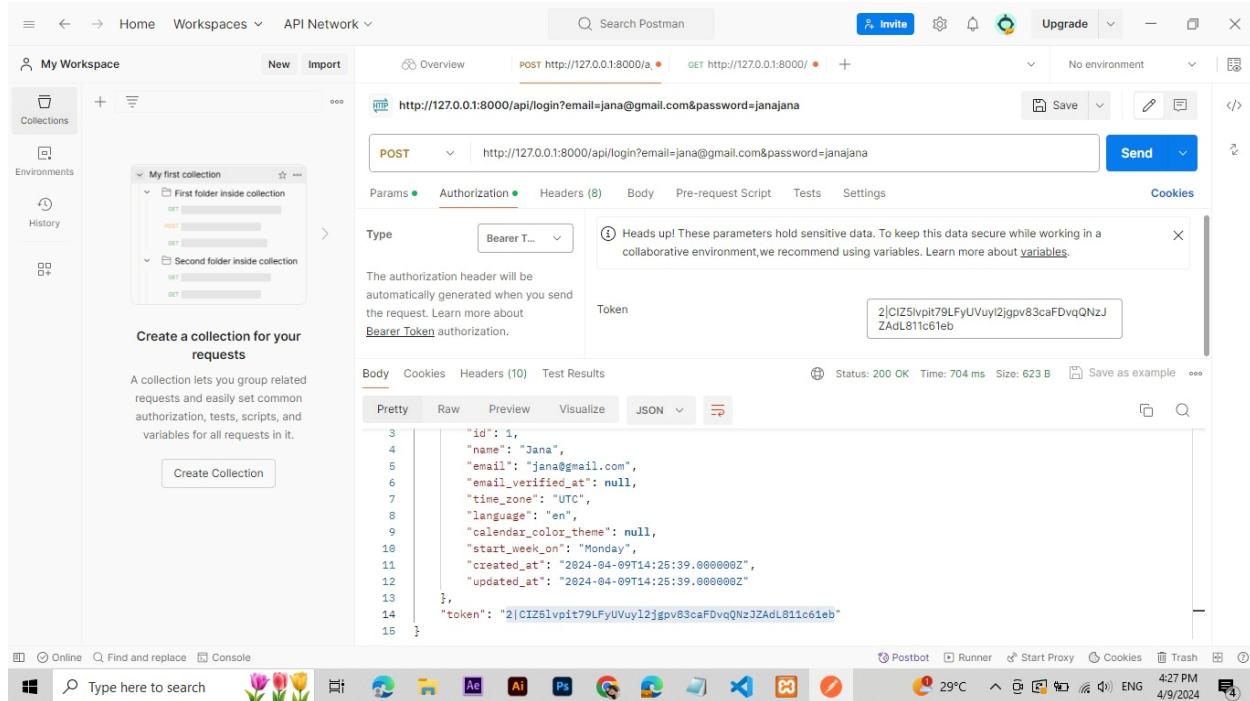
The screenshot shows the Postman interface with a collection named "My first collection". Inside this collection, there is a folder named "First folder inside collection" containing a single POST request. Another folder named "Second folder inside collection" also contains a single POST request. The main request is for the URL `http://127.0.0.1:8000/api/register?name=Jana&email=jana@gmail.com&password=janajana`. The "Params" tab shows four parameters: "name" (value: "Jana"), "email" (value: "jana@gmail.com"), and "password" (value: "janajana"). The response status is 200 OK, and the response body is a JSON object:

```
4
5   "name": "Jana",
6   "email": "jana@gmail.com",
7   "email_verified_at": null,
8   "time_zone": null,
9   "language": null,
10  "calendar_color_theme": null,
11  "start_week_on": null,
12  "created_at": "2024-04-09T14:25:39.000000Z",
13  "updated_at": "2024-04-09T14:25:39.000000Z"
14 },
15  "token": "1|6WHMLQR65saDC4dctUziCybcq2DsUAH5YYV7SsS54d32176",
16  "message": "Registracija je uspesna"
```

The screenshot shows the Postman interface with the same "My first collection" and its sub-folders. A new POST request has been added to the "First folder inside collection" for the URL `http://127.0.0.1:8000/api/login?email=jana@gmail.com&password=janajana`. The "Params" tab shows two parameters: "email" (value: "jana@gmail.com") and "password" (value: "janajana"). The response status is 200 OK, and the response body is a JSON object:

```
3
4   "id": 1,
5   "name": "Jana",
6   "email": "jana@gmail.com",
7   "email_verified_at": null,
8   "time_zone": "UTC",
9   "language": "en",
10  "calendar_color_theme": null,
11  "start_week_on": "Monday",
12  "created_at": "2024-04-09T14:25:39.000000Z",
13  "updated_at": "2024-04-09T14:25:39.000000Z"
14 },
15  "token": "2|CIZ61vpit79LFyUVuy12jgpv83caFDvqQNzJZAdL811c61eb"
```

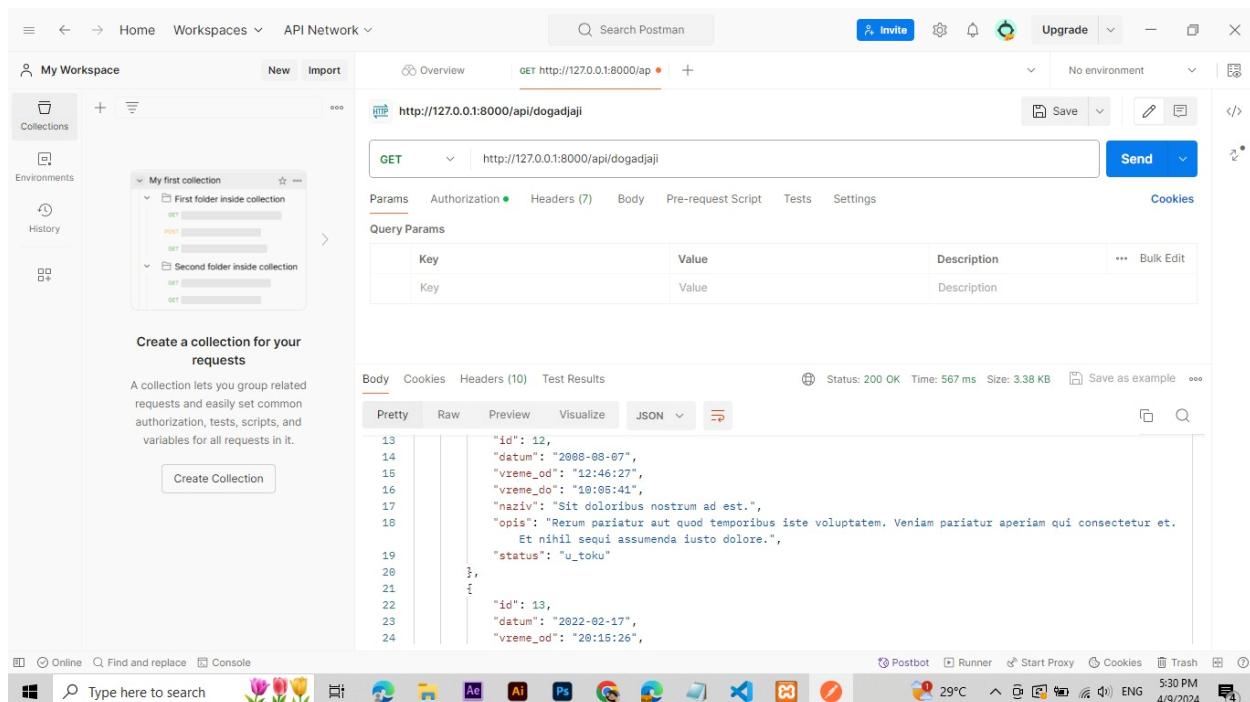
Ukoliko su uneti parametri uspešni korisnici će dobiti token, potrebno je taj token kopirati i smestiti u karticu za autorizaciju, kako bismo mogli da izvršavamo sledeće operacije.



The screenshot shows the Postman interface with a collection named "My first collection". Inside this collection are two folders: "First folder inside collection" and "Second folder inside collection", each containing a single GET request. The main workspace shows a POST request to `http://127.0.0.1:8000/api/login?email=jana@gmail.com&password=janajana`. The "Authorization" tab is selected, showing a "Bearer Token" input field with the value `2|CIZ5lvpit79LFyUVuy12|gpv83caFDvqQNzJAdL811c61eb`. The "Body" tab shows the JSON response:

```
3 |   "id": 1,
4 |   "name": "Jana",
5 |   "email": "jana@gmail.com",
6 |   "email_verified_at": null,
7 |   "time_zone": "UTC",
8 |   "language": "en",
9 |   "calendar_color_theme": null,
10 |  "start_week_on": "Monday",
11 |  "created_at": "2024-04-09T14:25:39.000000Z",
12 |  "updated_at": "2024-04-09T14:25:39.000000Z"
13 | },
14 | "token": "2|CIZ5lvpit79LFyUVuy12|gpv83caFDvqQNzJAdL811c61eb"
```

Kada je korisnik ulogovan ima mogućnost prikaza svih svojih dogadjaja.



The screenshot shows the Postman interface with the same "My first collection" structure. The main workspace shows a GET request to `http://127.0.0.1:8000/api/dogadjaji`. The "Headers" tab is selected, showing "Content-Type: application/json". The "Body" tab shows the JSON response:

```
13 |   "id": 12,
14 |   "datum": "2008-08-07",
15 |   "vreme_od": "12:46:27",
16 |   "vreme_do": "18:05:41",
17 |   "naziv": "Sit doloribus nostrum ad est.",
18 |   "opis": "Rerum pariatur aut quod temporibus iste voluptatem. Veniam pariatur aperiam qui consequetur et.
19 |   Et nihil sequi assumenda iusto dolore.",
20 |   "status": "u_toku"
21 |
22 |
23 |
24 |   "id": 13,
25 |   "datum": "2022-02-17",
26 |   "vreme_od": "20:15:26",
```

Korisnik ima i mogućnost izmene, brisanja i dodavanja događaja.

The screenshot shows the Postman interface with a POST request to `http://127.0.0.1:8000/api/dogadjaji`. The request parameters are:

- datum: 2025-05-04
- vreme_od: 17:30
- vreme_do: 19:15
- naziv: Rodjendan
- opis: janin rodjendan

The response status is 201 Created, and the JSON body is:

```
1 {
2     "data": {
3         "id": 75,
4         "datum": "2025-05-04",
5         "vreme_od": "17:30",
6         "vreme_do": "19:15",
7         "naziv": "Rodjendan",
8         "opis": "janin rodjendan",
9         "status": "zakazano"
10    }
11 }
```

The screenshot shows the Postman interface with a PUT request to `http://127.0.0.1:8000/api/dogadjaji/75`. The request parameters are:

- datum: 2025-05-04
- vreme_od: 17:15
- vreme_do: 19:15
- naziv: Rodjendan
- opis: janin rodjendan

The response status is 200 OK, and the JSON body is:

```
1 {
2     "data": {
3         "id": 75,
4         "datum": "2025-05-04",
5         "vreme_od": "17:15",
6         "vreme_do": "19:15",
7         "naziv": "Rodjendan",
8         "opis": "janin rodjendan",
9         "status": "zakazano"
10    }
11 }
```

The screenshot shows the Postman interface with a collection named "Your collection". A specific request is being tested: a DELETE operation on the URL `http://127.0.0.1:8000/api/dogadjaji/75`. The request includes the following parameters:

- datum: 2025-05-04
- vreme_od: 17:15
- vreme_do: 19:15
- naziv: Rodjendan
- opis: janin rodjendan

The response status is 204 No Content, with a time of 465 ms and a size of 284 B.

Na sličan način, korisnik ima mogućnost kreiranja, brisanja, prikaza i ažuriranja kategorija i podsetnika. Prikazaćemo na slikama CRUD operacije za podsetnik.

The screenshot shows the Postman interface with a collection named "Your collection". A specific request is being tested: a GET operation on the URL `http://127.0.0.1:8000/api/reminders`. The response is a JSON object with the key "data" containing two reminder entries:

```

1: {
  "data": [
    {
      "id": 51,
      "reminded_time": "2024-03-15 04:28:21",
      "note": "Placeat repellat est praesentium impedit praesentium sit quo."
    },
    {
      "id": 52,
      "reminded_time": "2024-05-03 00:51:28",
      "note": "Ut voluptas soluta optio pariatur."
    }
  ]
}

```

The response status is 200 OK, with a time of 553 ms and a size of 5.35 KB.

My Workspace

Overview POST http://127.0.0.1:8000/a.

http://127.0.0.1:8000/api/reminders?dogadaj_id=1&reminder_time=2024-05-05 12:00:00

POST http://127.0.0.1:8000/api/reminders?dogadaj_id=1&reminder_time=2024-05-05 12:00:00

Send

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies

Query Params

Key	Value	Description
dogadaj_id	1	
reminder_time	2024-05-05 12:00:00	

Body Cookies Headers (10) Test Results Status: 201 Created Time: 561 ms Size: 382 B Save as example

Pretty Raw Preview Visualize JSON

```
1 {  
2   "data": {  
3     "id": 351,  
4     "reminder_time": "2024-05-05 12:00:00",  
5     "note": null  
6   }  
7 }
```

Online Find and replace Console Postbot Runner Start Proxy Cookies Trash

Type here to search

My Workspace

Overview DEL http://127.0.0.1:8000/a.

http://127.0.0.1:8000/api/reminders/351?dogadaj_id=1&reminder_time=2024-05-05 12:00:00

DELETE http://127.0.0.1:8000/api/reminders/351?dogadaj_id=1&reminder_time=2024-05-05 12:00:00

Send

Params Authorization Headers (7) Body Pre-request Script Tests Settings Cookies

Query Params

Key	Value	Description
dogadaj_id	1	
reminder_time	2024-05-05 12:00:00	

Body Cookies Headers (9) Test Results Status: 204 No Content Time: 498 ms Size: 284 B Save as example

Pretty Raw Preview Visualize Text

```
1
```

Online Find and replace Console Postbot Runner Start Proxy Cookies Trash

Type here to search

Nakon što je korisnik završio sa radom u aplikaciji, potrebno je da se odjavi.

The screenshot shows the Postman application interface. On the left, there's a sidebar with 'My Workspace' containing 'Collections', 'Environments', and 'History'. A 'Create a collection for your requests' section is present. The main area shows a POST request to 'http://127.0.0.1:8000/api/logout'. The 'Authorization' tab is selected, showing an API Key. The 'Headers' tab lists 8 items. The 'Body' tab is selected, showing a JSON response with the message: "message": "Successfully logged out". Below the body, the status is 200 OK, time is 538 ms, and size is 345 B. The bottom of the screen shows the Windows taskbar with various pinned icons like File Explorer, Microsoft Edge, Adobe Photoshop, and others, along with system status icons for battery level, signal strength, and temperature.