Univerzitet u Beogradu

Fakultet organizacionih nauka

Katedra za elektronsko poslovanje

Laravel

Domaći zadatak 2

|  |  |  |
| --- | --- | --- |
| Ime | Prezime | Broj indeksa |
|  |  |  |
| Link ka Github-u |  | |

Sadržaj

[1 Korisnički zahtev 3](#_Toc154095131)

[2 Implementacija 4](#_Toc154095132)

[2.1 Model 4](#_Toc154095133)

[2.2 Migracije 4](#_Toc154095134)

[2.3 Kontroleri 6](#_Toc154095135)

[2.4 Api rute 7](#_Toc154095136)

[3 REST API 8](#_Toc154095137)

[4 Korisničko uputstvo 17](#_Toc154095138)

# 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 j e 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.

Implementacija

## Model

Model i njihove veze su prikazane na slici ispod

Slika na kojoj se nalazi dijagram, skeč, crtež

Opis je automatski generisan

Slika 1 pmov

## Migracije

U ovoj aplikaciji su kreirane sledeće migracije

1. **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.
2. **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 opciona, 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.

## 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 vrate 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 idex kako bi se obezbedila mogucnost kesiranja dogadjaja

    {

        /\* remember metoda pokušava da preuzme keširane podatke koristeći ključ 'dogadjaji\_cache\_key'.

        Ako podaci nisu dostupni u kešu, izvršava se anonimna 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::all();

        });

        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 azuiranja brisanja i dodavanja dogadjaja cemo brisati kes memoriju kako bi se osvezila prilikom sledeceg ucitavanja podataka

        return new DogadjajResource($dogadjaj);

    }

    public function show($id)

    {

        $dogadjaj = Dogadjaj::findOrFail($id);

        return new DogadjajResource($dogadjaj);

    }

    public function update(Request $request, $id)

    {

        $validator = Validator::make($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' => 'string|nullable',

            '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);

        }

        $dogadjaj = Dogadjaj::findOrFail($id);

        $dogadjaj->update($validator->validated());

        Cache::forget('dogadjaji\_cache\_key'); //nakon azuiranja brisanja i dodavanja dogadjaja cemo brisati kes memoriju kako bi se osvezila prilikom sledeceg ucitavanja podataka

        return new DogadjajResource($dogadjaj);

    }

    public function destroy($id)

    {

        $dogadjaj = Dogadjaj::findOrFail($id);

        $dogadjaj->delete();

        Cache::forget('dogadjaji\_cache\_key'); //nakon azuiranja brisanja i dodavanja dogadjaja cemo brisati kes memoriju kako bi se osvezila prilikom sledeceg ucitavanja podataka

        return response()->json(null, 204);

    }

    public function search(Request $request)

    {

        $user\_id = auth()->id();

        $query = Dogadjaj::where('user\_id', $user\_id);

        if ($request->filled('datum')) {

            $query->where('datum', $request->datum);

        }

        if ($request->filled('naziv')) {

            $query->where('naziv', 'like', '%' . $request->naziv . '%');

        }

        if ($request->filled('status')) {

            $query->where('status', $request->status);

        }

        if ($request->filled('kategorija\_id')) {

            $query->where('kategorija\_id', $request->kategorija\_id);

        }

        $dogadjaji = $query->get();

        return DogadjajResource::collection($dogadjaji);

    }

}

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.

<?php

namespace App\Http\Controllers;

use App\Http\Resources\UserResource;

use App\Models\User;

use Illuminate\Http\Request;

use Illuminate\Support\Facades\Auth;

use Illuminate\Support\Facades\Hash;

use Illuminate\Support\Facades\Validator;

use Laravel\Sanctum\PersonalAccessToken;

class AuthController extends Controller

{

    public function register(Request $request)

    {

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

            'name' => 'required|string|max:255',

            'email' => 'required|string|email|max:255|unique:users',

            'password' => 'required|string|min:8|confirmed',

        ]);

        if ($validator->fails()) {

            return response()->json($validator->errors(), 422);

        }

        $user = User::create([

            'name' => $request->name,

            'email' => $request->email,

            'password' => Hash::make($request->password),

        ]);

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

        return response()->json(['user' => new UserResource($user), 'token' => $token]);

    }

    public function login(Request $request)

    {

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

            'email' => 'required|string|email',

            'password' => 'required|string',

        ]);

        if ($validator->fails()) {

            return response()->json($validator->errors(), 422);

        }

        if (!Auth::attempt($request->only('email', 'password'))) {

            return response()->json(['message' => 'Unauthorized'], 401);

        }

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

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

        return response()->json(['user' => new UserResource($user), 'token' => $token]);

    }

    public function logout(Request $request)

    {

        $request->user()->tokens()->delete();

        return response()->json(['message' => 'Successfully logged out']);

    }

}

## 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 **dogadjaji** 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.

# 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 | {      "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"  } |
| 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 | {      "data": [          {              "id": 1,              "datum": "1986-01-05",              "vreme\_od": "15:02:07",              "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"          },  .. |
| 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 | {      "data": {          "id": 71,          "datum": "2024-03-26",          "vreme\_od": "11:29",          "vreme\_do": "14:16",          "naziv": "dogadjaj1",          "opis": "opis dogadjaja1",          "status": "odlozeno"      }  } |
| 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 dogadjaja1&status=odlozeno&kategorija\_id=1&user\_id=1 |
| HTTP body parametri | (nema) |
| Format HTTP body parametara | (nema) |
| Izlazni parametri | {      "data": {          "id": 71,          "datum": "2024-03-27",          "vreme\_od": "11:29",          "vreme\_do": "14:16",          "naziv": "dogadjaj1",          "opis": "opis dogadjaja1",          "status": "odlozeno"      }  } |
| 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 | {      "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."          },  .. |
| 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 | {      "data": {          "id": 351,          "reminder\_time": "2024-04-23 03:27:51",          "note": **null**      }  } |
| 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 | {      "data": {          "id": 351,          "reminder\_time": "2024-04-28 03:27:51",          "note": **null**      }  } |
| 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 |

# Korisničko uputstvo

Kako bi korisnici mogli da koriste našu aplikaciju, potrebno je da se prvo uloguju

Slika na kojoj se nalazi tekst, snimak ekrana, broj, linija

Opis je automatski generisan

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.

Slika na kojoj se nalazi tekst, snimak ekrana, Font

Opis je automatski generisan

Kada je korisnik ulogovan ima mogućnost prikaza svih svojih dogadjaja.

Slika na kojoj se nalazi tekst, snimak ekrana, broj, linija

Opis je automatski generisan

Korisnik ima I mogućnost izmene, brisanja I dodavanja događaja.

Slika na kojoj se nalazi tekst, snimak ekrana, broj, softver

Opis je automatski generisan

Slika na kojoj se nalazi tekst, snimak ekrana, broj, softver

Opis je automatski generisan

Slika na kojoj se nalazi tekst, linija, broj, softver

Opis je automatski generisan

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

Slika na kojoj se nalazi tekst, snimak ekrana, broj, dijagram

Opis je automatski generisan

Slika na kojoj se nalazi tekst, snimak ekrana, broj, softver

Opis je automatski generisan

Slika na kojoj se nalazi tekst, snimak ekrana, softver, broj

Opis je automatski generisan

Slika na kojoj se nalazi tekst, softver, broj, Ikona na računaru

Opis je automatski generisan

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

Slika na kojoj se nalazi tekst, softver, linija, broj

Opis je automatski generisan