Univerzitet u Beogradu

Fakultet organizacionih nauka

Katedra za elektronsko poslovanje

Laravel

Domaći zadatak 1

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

Sadržaj

[1 Korisnički zahtev 3](#_Toc209801303)

[2 Implementacija 4](#_Toc209801304)

[3 REST API 16](#_Toc209801305)

[4 Korisničko uputstvo 27](#_Toc209801306)

# Korisnički zahtev

Aplikacija za upravljanje događajima razvijena je u okviru Laravel platforme kao REST-orijentisan backend koji isporučuje JSON odgovore putem API Resources. Primarna namena sistema je da omogući administraciji da kreira i organizuje događaje po kategorijama, dok registrovani korisnici mogu da pretražuju događaje i rezervišu (uzimaју) karte. Arhitektura je slojevita i oslanja se na Eloquent ORM za rad sa bazom, jasne kontrolere za poslovnu logiku i resurse za konzistentnu serijalizaciju odgovora.

Domen model čine četiri entiteta: User (sa poljem role – admin ili user), Category (naziv i opis), Event (naslov, opis, lokacija, početak/kraj i veza na kategoriju) i Ticket (veza na događaj i korisnika, status i opcionalna cena). Odnosi su: jedna kategorija ima više događaja, događaj ima više karata, a korisnik može imati više karata. Integritet podataka obezbeđen je spoljnim ključevima, jedinstvenim ograničenjem nad kombinacijom (event\_id, user\_id) kako bi jedan korisnik mogao imati najviše jednu kartu po događaju, kao i indeksima nad često filtriranim kolonama (npr. vreme početka događaja, status karte).

API sloj sadrži kontrolere i pravila pristupa: za Category i Event kreiranje, izmena i brisanje dozvoljeni su samo administratorima, dok je čitanje javno dostupno. Lista događaja podržava paginaciju i filter po kategoriji, čime se olakšava klijentska pretraga. Ticket kontroler sledi poslovna pravila: u index metodi korisnik vidi samo sopstvene karte (uz opcioni filter event\_id), dok administrator vidi sve; store za običnog korisnika uvek kreira kartu za njega sa statusom pending, dok administrator može kreirati kartu za bilo kog korisnika i odmah postaviti status/cenu; update (status i/ili cena) dozvoljen je isključivo administratoru; destroy može izvršiti ili vlasnik karte (za svoju kartu) ili administrator (za bilo koju). Povratni formati su ujednačeni korišćenjem CategoryResource, EventResource i TicketResource klasa.

# Implementacija

Kontroler za kategorije obezbeđuje kompletan CRUD nad entitetom Category uz jasno definisana pravila pristupa. Metoda index vraća JSON listu svih kategorija sortiranu po nazivu, proširenu brojačem povezanih događaja (events\_count); ako nema zapisa, vraća 404. show isporučuje pojedinačnu kategoriju sa brojem događaja. Svi izlazi su ujednačeni preko CategoryResource, čime je format odgovora stabilan za klijente.

Mutacione operacije su rezervisane za administratore: store kreira novu kategoriju uz validaciju jedinstvenog name i opcionog description, update ažurira postojeću (uz unique pravilo koje izuzima trenutni ID), a destroy briše kategoriju. Za sve ove rute kontroler eksplicitno proverava autentikaciju (Auth::check()) i ulogu (role === 'admin'), pri čemu greške vraća kao jasne JSON poruke i odgovarajuće HTTP kodove (401/403/422).

<?php

namespace App\Http\Controllers;

use App\Http\Resources\CategoryResource;

use App\Models\Category;

use Illuminate\Http\Request;

use Illuminate\Support\Facades\Auth;

class CategoryController extends Controller

{

    /\*\*

     \* Display a listing of the resource.

     \*/

    public function index()

    {

        $categories = Category::withCount('events')->orderBy('name')->get();

        if ($categories->isEmpty()) {

            return response()->json('No categories found.', 404);

        }

        return response()->json([

            'categories' => CategoryResource::collection($categories),

        ]);

    }

    /\*\*

     \* Show the form for creating a new resource.

     \*/

    public function create()

    {

        //

    }

    /\*\*

     \* Store a newly created resource in storage.

     \*/

    public function store(Request $request)

    {

        if (!Auth::check() || Auth::user()->role !== 'admin') {

            return response()->json(['error' => 'Only admins can create categories'], 403);

        }

        $validated = $request->validate([

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

            'description' => 'nullable|string',

        ]);

        $category = Category::create($validated);

        return response()->json([

            'message'  => 'Category created successfully',

            'category' => new CategoryResource($category),

        ]);

    }

    /\*\*

     \* Display the specified resource.

     \*/

    public function show(Category $category)

    {

        $category->loadCount('events');

        return response()->json([

            'category' => new CategoryResource($category),

        ]);

    }

    /\*\*

     \* Show the form for editing the specified resource.

     \*/

    public function edit(Category $category)

    {

        //

    }

    /\*\*

     \* Update the specified resource in storage.

     \*/

    public function update(Request $request, Category $category)

    {

        if (!Auth::check() || Auth::user()->role !== 'admin') {

            return response()->json(['error' => 'Only admins can update categories'], 403);

        }

        $validated = $request->validate([

            'name'        => 'sometimes|string|max:255|unique:categories,name,' . $category->id,

            'description' => 'sometimes|nullable|string',

        ]);

        $category->update($validated);

        return response()->json([

            'message'  => 'Category updated successfully',

            'category' => new CategoryResource($category),

        ]);

    }

    /\*\*

     \* Remove the specified resource from storage.

     \*/

    public function destroy(Category $category)

    {

        if (!Auth::check() || Auth::user()->role !== 'admin') {

            return response()->json(['error' => 'Only admins can delete categories'], 403);

        }

        $category->delete();

        return response()->json(['message' => 'Category deleted successfully']);

    }

}

Kontroler za događaje omogućava javno čitanje i administrativno upravljanje entitetom Event. index vraća straniciran spisak događaja sa eager-loaded kategorijom i brojem karata (tickets\_count), uz mogućnost filtriranja po category\_id. Na taj način klijent dobija efikasan i konzistentan prikaz sa metapodacima o paginaciji i linkovima za navigaciju. show vraća jedan događaj sa učitanom kategorijom i brojačem karata.

Kreiranje (store), izmena (update) i brisanje (destroy) dozvoljeni su samo administratorima; kontroler proverava autentikaciju i ulogu pre svake mutacije. Pri unosu i izmeni validiraju se obavezna polja i vremenski odnosi (end\_time posle ili jednako start\_time), a odgovori se serijalizuju preko EventResource. Time se obezbeđuje dosledan JSON izlaz, pravilno rukovanje greškama (422 za validaciju, 403 za nedozvoljene operacije) i jasno definisana poslovna pravila.

<?php

namespace App\Http\Controllers;

use App\Http\Resources\EventResource;

use App\Models\Event;

use Carbon\Carbon;

use Illuminate\Http\Request;

use Illuminate\Support\Facades\Auth;

class EventController extends Controller

{

    /\*\*

     \* Display a listing of the resource.

     \*/

    public function index(Request $request)

    {

        $perPage = max(1, (int) $request->input('per\_page', 10));

        $query = Event::with(['category'])

            ->withCount('tickets')

            ->when($request->filled('category\_id'), function ($q) use ($request) {

                $q->where('category\_id', (int) $request->input('category\_id'));

            })

            ->orderBy('start\_time');

        $events = $query->paginate($perPage);

        if ($events->total() === 0) {

            return response()->json('No events found.', 404);

        }

        return response()->json([

            'events' => EventResource::collection($events->items()),

            'meta' => [

                'current\_page' => $events->currentPage(),

                'last\_page'    => $events->lastPage(),

                'per\_page'     => $events->perPage(),

                'total'        => $events->total(),

            ],

            'links' => [

                'first' => $events->url(1),

                'last'  => $events->url($events->lastPage()),

                'prev'  => $events->previousPageUrl(),

                'next'  => $events->nextPageUrl(),

            ],

        ]);

    }

    /\*\*

     \* Show the form for creating a new resource.

     \*/

    public function create()

    {

        //

    }

    /\*\*

     \* Store a newly created resource in storage.

     \*/

    public function store(Request $request)

    {

        if (!Auth::check() || Auth::user()->role !== 'admin') {

            return response()->json(['error' => 'Only admins can create events'], 403);

        }

        $validated = $request->validate([

            'title'       => ['required', 'string', 'max:255'],

            'description' => ['nullable', 'string'],

            'location'    => ['nullable', 'string', 'max:255'],

            'start\_time'  => ['required', 'date'],

            'end\_time'    => ['nullable', 'date', 'after\_or\_equal:start\_time'],

            'category\_id' => ['nullable', 'integer', 'exists:categories,id'],

        ]);

        $event = Event::create($validated)->load(['category'])->loadCount('tickets');

        return response()->json([

            'message' => 'Event created successfully',

            'event'   => new EventResource($event),

        ], 201);

    }

    /\*\*

     \* Display the specified resource.

     \*/

    public function show(Event $event)

    {

        $event->load(['category'])->loadCount('tickets');

        return response()->json([

            'event' => new EventResource($event),

        ]);

    }

    /\*\*

     \* Show the form for editing the specified resource.

     \*/

    public function edit(Event $event)

    {

        //

    }

    /\*\*

     \* Update the specified resource in storage.

     \*/

    public function update(Request $request, Event $event)

    {

        if (!Auth::check() || Auth::user()->role !== 'admin') {

            return response()->json(['error' => 'Only admins can update events'], 403);

        }

        $data = $request->validate([

            'title'       => ['sometimes', 'string', 'max:255'],

            'description' => ['sometimes', 'nullable', 'string'],

            'location'    => ['sometimes', 'nullable', 'string', 'max:255'],

            'start\_time'  => ['sometimes', 'date'],

            'end\_time'    => ['sometimes', 'nullable', 'date'],

            'category\_id' => ['sometimes', 'nullable', 'integer', 'exists:categories,id'],

        ]);

        // Ako stiže samo end\_time (bez start\_time), ručno proveri odnos sa postojećim start\_time

        if (array\_key\_exists('end\_time', $data)) {

            $start = isset($data['start\_time']) ? new Carbon($data['start\_time']) : new Carbon($event->start\_time);

            $end   = $data['end\_time'] ? new Carbon($data['end\_time']) : null;

            if ($end && $end->lt($start)) {

                return response()->json([

                    'errors' => ['end\_time' => ['The end\_time must be a date after or equal to start\_time.']]

                ], 422);

            }

        }

        $event->update($data);

        $event->load(['category'])->loadCount('tickets');

        return response()->json([

            'message' => 'Event updated successfully',

            'event'   => new EventResource($event),

        ]);

    }

    /\*\*

     \* Remove the specified resource from storage.

     \*/

    public function destroy(Event $event)

    {

        if (!Auth::check() || Auth::user()->role !== 'admin') {

            return response()->json(['error' => 'Only admins can delete events'], 403);

        }

        $event->delete();

        return response()->json([

            'message' => 'Event deleted successfully',

        ]);

    }

}

Kontroler za karte sprovodi pravila koja razlikuju ponašanje korisnika i administratora. index vraća kolekciju karata bez paginacije; ako je ulogovan korisnik običan user, vidi isključivo sopstvene karte (uz opciono filtriranje event\_id), dok admin vidi sve karte i može ih filtrirati po događaju. Odgovor je konzistentan preko TicketResource, sa učitanim relacijama na događaj (i njegovu kategoriju) i korisnika.

Pri kreiranju (store) običan korisnik može izdati kartu samo sebi; status se automatski postavlja na pending, a slanje cene je eksplicitno zabranjeno. Administrator, nasuprot tome, može kreirati kartu za bilo kog korisnika i postaviti status i/ili cenu već pri unosu. Jedinstvenost kombinacije (event\_id, user\_id) se proverava na nivou validacije i baze. Ažuriranje (update) je rezervisano isključivo za administratore i služi za promenu statusa i/ili cene, dok brisanje (destroy) može izvršiti ili vlasnik sopstvene karte ili administrator nad bilo kojom kartom. Sve grane vraćaju jasne JSON poruke i odgovarajuće HTTP kodove (201 za uspešno kreiranje, 401/403/404/422 po potrebi).

<?php

namespace App\Http\Controllers;

use App\Http\Resources\TicketResource;

use App\Models\Ticket;

use Illuminate\Http\Request;

use Illuminate\Support\Facades\Auth;

use Illuminate\Validation\Rule;

class TicketController extends Controller

{

    /\*\*

     \* Display a listing of the resource.

     \*/

    public function index(Request $request)

    {

        if (!Auth::check()) {

            return response()->json(['error' => 'Unauthenticated'], 401);

        }

        $query = Ticket::query()

            ->with(['event.category', 'user'])

            ->orderByDesc('created\_at');

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

            $query->where('event\_id', (int) $request->input('event\_id'));

        }

        if (Auth::user()->role !== 'admin') {

            $query->where('user\_id', Auth::id());

        }

        $tickets = $query->get();

        if ($tickets->isEmpty()) {

            return response()->json('No tickets found.', 404);

        }

        return response()->json([

            'tickets' => TicketResource::collection($tickets),

        ]);

    }

    /\*\*

     \* Show the form for creating a new resource.

     \*/

    public function create()

    {

        //

    }

    /\*\*

     \* Store a newly created resource in storage.

     \*/

    public function store(Request $request)

    {

        if (!Auth::check()) {

            return response()->json(['error' => 'Unauthenticated'], 401);

        }

        $authUser = Auth::user();

        $allowedStatuses = ['pending', 'confirmed', 'cancelled'];

        // Odredi kome pravimo kartu

        $targetUserId = $authUser->role === 'admin'

            ? (int) $request->input('user\_id', $authUser->id)

            : $authUser->id;

        // Osnovna pravila (unique(event\_id,user\_id))

        $rules = [

            'event\_id' => [

                'required',

                'integer',

                'exists:events,id',

                Rule::unique('tickets')->where(

                    fn($q) => $q->where('event\_id', (int) $request->input('event\_id'))

                        ->where('user\_id',  $targetUserId)

                ),

            ],

        ];

        if ($authUser->role === 'admin') {

            // Admin sme user\_id, status i price

            $rules['user\_id'] = ['sometimes', 'required', 'integer', 'exists:users,id'];

            $rules['status']  = ['sometimes', 'required', Rule::in($allowedStatuses)];

            $rules['price']   = ['sometimes', 'nullable', 'numeric', 'min:0'];

        } else {

            // Useru eksplicitno ZABRANI slanje price (ako pošalje -> 422)

            $rules['price'] = ['prohibited'];

            // User ne može da postavi status – uvek pending

        }

        $validated = $request->validate($rules, [

            'event\_id.unique' => 'Korisnik već ima kartu za ovaj događaj.',

        ]);

        $status = $authUser->role === 'admin'

            ? ($request->input('status', 'pending'))

            : 'pending';

        $price = $authUser->role === 'admin'

            ? ($validated['price'] ?? null)

            : null;

        $ticket = Ticket::create([

            'event\_id' => (int) $validated['event\_id'],

            'user\_id'  => $targetUserId,

            'status'   => $status,

            'price'    => $price,

        ])->load(['event.category', 'user']);

        return response()->json([

            'message' => 'Ticket created successfully',

            'ticket'  => new TicketResource($ticket),

        ], 201);

    }

    /\*\*

     \* Display the specified resource.

     \*/

    public function show(Ticket $ticket)

    {

        //

    }

    /\*\*

     \* Show the form for editing the specified resource.

     \*/

    public function edit(Ticket $ticket)

    {

        //

    }

    /\*\*

     \* Update the specified resource in storage.

     \*/

    public function update(Request $request, Ticket $ticket)

    {

        if (!Auth::check() || Auth::user()->role !== 'admin') {

            return response()->json(['error' => 'Only admins can update tickets'], 403);

        }

        // Admin može menjati status i/ili cenu

        $data = $request->validate([

            'status' => ['sometimes', Rule::in(['pending', 'confirmed', 'cancelled'])],

            'price'  => ['sometimes', 'nullable', 'numeric', 'min:0'],

        ]);

        // Bar jedno polje mora doći

        if (!array\_key\_exists('status', $data) && !array\_key\_exists('price', $data)) {

            return response()->json([

                'errors' => ['update' => ['Provide at least one of: status or price.']]

            ], 422);

        }

        $ticket->update($data);

        $ticket->load(['event.category', 'user']);

        return response()->json([

            'message' => 'Ticket updated successfully',

            'ticket'  => new TicketResource($ticket),

        ]);

    }

    /\*\*

     \* Remove the specified resource from storage.

     \*/

    public function destroy(Ticket $ticket)

    {

        if (!Auth::check()) {

            return response()->json(['error' => 'Unauthenticated'], 401);

        }

        $user = Auth::user();

        if ($user->role !== 'admin' && $ticket->user\_id !== $user->id) {

            return response()->json(['error' => 'You can only delete your own tickets'], 403);

        }

        $ticket->delete();

        return response()->json([

            'message' => 'Ticket deleted successfully',

        ]);

    }

}

# REST API

|  |  |
| --- | --- |
| Opis funkcije | Register User |
| HTTP metoda | POST |
| URL | /api/register |
| URL parametri | (nema) |
| HTTP body parametri | {      "name": "Biljana",      "email": "biljana@mail.com",      "password": "password"  } |
| Format HTTP body parametara | JSON |
| Izlazni parametri | {      "data": {          "name": "Biljana",          "email": "biljana@mail.com",          "updated\_at": "2025-09-26T15:08:50.000000Z",          "created\_at": "2025-09-26T15:08:50.000000Z",          "id": 7      },      "access\_token": "1|d1usGMTaJ8IaaaW4eijniNQEeYOpDXwOW8gMZ7lN30532bee",      "token\_type": "Bearer"  } |
| Format izlaznih parametara | application/json |
| Opis funkcije | Login User |
| HTTP metoda | POST |
| URL | /api/login |
| URL parametri | (nema) |
| HTTP body parametri | {      "email": "biljana@mail.com",  // biljana@mail.com // admin@events.local      "password": "password" // admin123  } |
| Format HTTP body parametara | JSON |
| Izlazni parametri | {      "message": "Biljana logged in",      "access\_token": "2|0wuwtbw7c381dRXQJBHzEVa5ual2lrXxTtD9hXbe86013bad",      "token\_type": "Bearer"  } |
| Format izlaznih parametara | application/json |

|  |  |
| --- | --- |
| Opis funkcije | Logout User |
| HTTP metoda | POST |
| URL | /api/logout |
| URL parametri | (nema) |
| HTTP body parametri | (nema) |
| Format HTTP body parametara | (nema) |
| Izlazni parametri | {      "message": "You have successfully logged out."  } |
| Format izlaznih parametara | application/json |

|  |  |
| --- | --- |
| Opis funkcije | Get Categories |
| HTTP metoda | GET |
| URL | /api/categories |
| URL parametri | (nema) |
| HTTP body parametri | (nema) |
| Format HTTP body parametara | (nema) |
| Izlazni parametri | {      "categories": [          {              "id": 4,              "name": "Concert",              "description": "Muzicki događaji – solo nastupi i bendovi."          },          {              "id": 1,              "name": "Conference",              "description": "Višednevni i jednodnevni događaji sa predavanjima i panelima."          },          {              "id": 3,              "name": "Meetup",              "description": "Neformalna okupljanja zajednice uz kraća predavanja."          },          {              "id": 5,              "name": "Webinar",              "description": "Online predavanja i demonstracije alata/tehnologija."          },          {              "id": 2,              "name": "Workshop",              "description": "Praktične radionice sa manjim grupama i mentorisanjem."          }      ]  } |
| Format izlaznih parametara | application/json |

|  |  |
| --- | --- |
| Opis funkcije | Get Category |
| HTTP metoda | GET |
| URL | /api/categories/:category |
| URL parametri | category=1 |
| HTTP body parametri | (nema) |
| Format HTTP body parametara | (nema) |
| Izlazni parametri | {      "category": {          "id": 1,          "name": "Conference",          "description": "Višednevni i jednodnevni događaji sa predavanjima i panelima."      }  } |
| Format izlaznih parametara | application/json |

|  |  |
| --- | --- |
| Opis funkcije | Create Category |
| HTTP metoda | POST |
| URL | /api/categories |
| URL parametri | (nema) |
| HTTP body parametri | {    "name": "Hackathon"  } |
| Format HTTP body parametara | JSON |
| Izlazni parametri | {      "message": "Category created successfully",      "category": {          "id": 6,          "name": "Hackathon",          "description": **null**      }  } |
| Format izlaznih parametara | application/json |

|  |  |
| --- | --- |
| Opis funkcije | Update Category |
| HTTP metoda | PUT |
| URL | /api/categories/:category |
| URL parametri | category=6 |
| HTTP body parametri | {      "description": "Takmičarski događaj gde timovi u kratkom roku razvijaju prototipe i rešenja."  } |
| Format HTTP body parametara | JSON |
| Izlazni parametri | {      "message": "Category updated successfully",      "category": {          "id": 6,          "name": "Hackathon",          "description": "Takmičarski događaj gde timovi u kratkom roku razvijaju prototipe i rešenja."      }  } |
| Format izlaznih parametara | application/json |

|  |  |
| --- | --- |
| Opis funkcije | Delete Category |
| HTTP metoda | DELETE |
| URL | /api/categories/:category |
| URL parametri | category=6 |
| HTTP body parametri | (nema) |
| Format HTTP body parametara | (nema) |
| Izlazni parametri | {      "message": "Category deleted successfully"  } |
| Format izlaznih parametara | application/json |

|  |  |
| --- | --- |
| Opis funkcije | Get Events |
| HTTP metoda | GET |
| URL | /api/events |
| URL parametri | ?per\_page=15&category\_id=1 |
| HTTP body parametri | (nema) |
| Format HTTP body parametara | (nema) |
| Izlazni parametri | {      "events": [          {              "id": 3,              "title": "Tech Conference: Data & AI 2025",              "description": "Konferencija o primeni veštačke inteligencije i analitike podataka.",              "location": "Beograd, Sava Centar",              "start\_time": "2025-10-26T09:00:00.000000Z",              "end\_time": "2025-10-26T17:00:00.000000Z",              "category": {                  "id": 1,                  "name": "Conference",                  "description": "Višednevni i jednodnevni događaji sa predavanjima i panelima."              },              "tickets\_count": 2          }      ],      "meta": {          "current\_page": 1,          "last\_page": 1,          "per\_page": 15,          "total": 1      },      "links": {          "first": "http://localhost:8000/api/events?page=1",          "last": "http://localhost:8000/api/events?page=1",          "prev": **null**,          "next": **null**      }  } |
| Format izlaznih parametara | application/json |

|  |  |
| --- | --- |
| Opis funkcije | Get Event |
| HTTP metoda | GET |
| URL | /api/events/:event |
| URL parametri | event =1 |
| HTTP body parametri | (nema) |
| Format HTTP body parametara | (nema) |
| Izlazni parametri | {      "event": {          "id": 1,          "title": "Laravel Srbija Meetup",          "description": "Dva kratka predavanja i networking uz pizzu.",          "location": "Beograd, ICT Hub",          "start\_time": "2025-10-06T18:00:00.000000Z",          "end\_time": "2025-10-06T20:00:00.000000Z",          "category": {              "id": 3,              "name": "Meetup",              "description": "Neformalna okupljanja zajednice uz kraća predavanja."          },          "tickets\_count": 4      }  } |
| Format izlaznih parametara | application/json |

|  |  |
| --- | --- |
| Opis funkcije | Create Event |
| HTTP metoda | POST |
| URL | /api/events |
| URL parametri | (nema) |
| HTTP body parametri | {    "title": "React Advanced Workshop: Performance & Patterns",    "description": "Praktična radionica o optimizaciji performansi, server components i naprednim obrascima u React-u.",    "location": "Novi Sad, Startit Centar",    "start\_time": "2025-10-15T10:00:00+02:00",    "end\_time": "2025-10-15T14:00:00+02:00",    "category\_id": 2  } |
| Format HTTP body parametara | JSON |
| Izlazni parametri | {      "message": "Event created successfully",      "event": {          "id": 6,          "title": "React Advanced Workshop: Performance & Patterns",          "description": "Praktična radionica o optimizaciji performansi, server components i naprednim obrascima u React-u.",          "location": "Novi Sad, Startit Centar",          "start\_time": "2025-10-15T10:00:00.000000Z",          "end\_time": "2025-10-15T14:00:00.000000Z",          "category": {              "id": 2,              "name": "Workshop",              "description": "Praktične radionice sa manjim grupama i mentorisanjem."          },          "tickets\_count": 0      }  } |
| Format izlaznih parametara | application/json |

|  |  |
| --- | --- |
| Opis funkcije | Update Event |
| HTTP metoda | PUT |
| URL | /api/events/:event |
| URL parametri | event=6 |
| HTTP body parametri | {      "end\_time": "2025-10-15T18:00:00+02:00"  } |
| Format HTTP body parametara | JSON |
| Izlazni parametri | {      "message": "Event updated successfully",      "event": {          "id": 6,          "title": "React Advanced Workshop: Performance & Patterns",          "description": "Praktična radionica o optimizaciji performansi, server components i naprednim obrascima u React-u.",          "location": "Novi Sad, Startit Centar",          "start\_time": "2025-10-15T10:00:00.000000Z",          "end\_time": "2025-10-15T18:00:00.000000Z",          "category": {              "id": 2,              "name": "Workshop",              "description": "Praktične radionice sa manjim grupama i mentorisanjem."          },          "tickets\_count": 0      }  } |
| Format izlaznih parametara | application/json |

|  |  |
| --- | --- |
| Opis funkcije | Delete Event |
| HTTP metoda | DELETE |
| URL | /api/events/:event |
| URL parametri | event=6 |
| HTTP body parametri | (nema) |
| Format HTTP body parametara | (nema) |
| Izlazni parametri | {      "message": "Event deleted successfully"  } |
| Format izlaznih parametara | application/json |

|  |  |
| --- | --- |
| Opis funkcije | Request Ticket |
| HTTP metoda | DELETE |
| URL | /api/tickets/ |
| URL parametri | (nema) |
| HTTP body parametri | {      "event\_id": 5  } |
| Format HTTP body parametara | JSON |
| Izlazni parametri | {      "message": "Ticket created successfully",      "ticket": {          "id": 17,          "event\_id": 5,          "user\_id": 7,          "status": "pending",          "price": **null**,          "event": {              "id": 5,              "title": "Remote Webinar: Intro to Docker",              "description": "Osnove dockera, images, containers i best practices.",              "location": "Online",              "start\_time": "2025-10-03T18:00:00.000000Z",              "end\_time": "2025-10-03T19:30:00.000000Z",              "category": {                  "id": 5,                  "name": "Webinar",                  "description": "Online predavanja i demonstracije alata/tehnologija."              }          },          "user": {              "id": 7,              "name": "Biljana",              "email": "biljana@mail.com",              "role": "user"          },          "created\_at": "2025-09-26T15:39:29.000000Z",          "updated\_at": "2025-09-26T15:39:29.000000Z"      }  } |
| Format izlaznih parametara | application/json |

|  |  |
| --- | --- |
| Opis funkcije | Get Tickets |
| HTTP metoda | GET |
| URL | /api/tickets |
| URL parametri | ?event\_id=5 |
| HTTP body parametri | (nema) |
| Format HTTP body parametara | (nema) |
| Izlazni parametri | {      "tickets": [          {              "id": 17,              "event\_id": 5,              "user\_id": 7,              "status": "pending",              "price": **null**,              "event": {                  "id": 5,                  "title": "Remote Webinar: Intro to Docker",                  "description": "Osnove dockera, images, containers i best practices.",                  "location": "Online",                  "start\_time": "2025-10-03T18:00:00.000000Z",                  "end\_time": "2025-10-03T19:30:00.000000Z",                  "category": {                      "id": 5,                      "name": "Webinar",                      "description": "Online predavanja i demonstracije alata/tehnologija."                  }              },              "user": {                  "id": 7,                  "name": "Biljana",                  "email": "biljana@mail.com",                  "role": "user"              },              "created\_at": "2025-09-26T15:39:29.000000Z",              "updated\_at": "2025-09-26T15:39:29.000000Z"          }      ]  } |
| Format izlaznih parametara | application/json |

|  |  |
| --- | --- |
| Opis funkcije | Update Ticket |
| HTTP metoda | PUT |
| URL | /api/tickets/:ticket |
| URL parametri | ticket=17 |
| HTTP body parametri | {      "status": "confirmed",      "price": 10  } |
| Format HTTP body parametara | JSON |
| Izlazni parametri | {      "message": "Ticket updated successfully",      "ticket": {          "id": 17,          "event\_id": 5,          "user\_id": 7,          "status": "confirmed",          "price": 10,          "event": {              "id": 5,              "title": "Remote Webinar: Intro to Docker",              "description": "Osnove dockera, images, containers i best practices.",              "location": "Online",              "start\_time": "2025-10-03T18:00:00.000000Z",              "end\_time": "2025-10-03T19:30:00.000000Z",              "category": {                  "id": 5,                  "name": "Webinar",                  "description": "Online predavanja i demonstracije alata/tehnologija."              }          },          "user": {              "id": 7,              "name": "Biljana",              "email": "biljana@mail.com",              "role": "user"          },          "created\_at": "2025-09-26T15:39:29.000000Z",          "updated\_at": "2025-09-26T15:42:36.000000Z"      }  } |
| Format izlaznih parametara | application/json |

|  |  |
| --- | --- |
| Opis funkcije | Delete Ticket |
| HTTP metoda | DELETE |
| URL | /api/tickets/:ticket |
| URL parametri | ticket=17 |
| HTTP body parametri | (nema) |
| Format HTTP body parametara | (nema) |
| Izlazni parametri | {      "message": "Ticket deleted successfully"  } |
| Format izlaznih parametara | application/json |

# Korisničko uputstvo

A screenshot of a computer

AI-generated content may be incorrect.

Slika 1: Registracija korisnika

A screenshot of a computer

AI-generated content may be incorrect.

Slika 2: Prijava korisnika

A screenshot of a computer

AI-generated content may be incorrect.

Slika 3: Pregled kategorija

A screenshot of a computer

AI-generated content may be incorrect.

Slika 4: Pregled kategorije

A screenshot of a computer

AI-generated content may be incorrect.

Slika 5: Kreiranje kategorije

A screenshot of a computer

AI-generated content may be incorrect.

Slika 6: Ažuriranje kategorije

A screenshot of a computer

AI-generated content may be incorrect.

Slika 7: Brisanje kategorije

A screenshot of a computer

AI-generated content may be incorrect.

Slika 8: Pregled Eventova sa filterima i paginacijom

A screenshot of a computer

AI-generated content may be incorrect.

Slika 9: Pregled Eventa

A screenshot of a computer

AI-generated content may be incorrect.

Slika 10: Kreiranje eventa

A screenshot of a computer

AI-generated content may be incorrect.

Slika 11: Ažuriranje eventa

A screenshot of a computer

AI-generated content may be incorrect.

Slika 12: Brisanje eventa

A screenshot of a computer

AI-generated content may be incorrect.

Slika 13: Request karte

A screenshot of a computer

AI-generated content may be incorrect.

Slika 14: Pregled karata

A screenshot of a computer

AI-generated content may be incorrect.

Slika 15: Odobrenje karte