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

Ova aplikacija je dizajnirana kao platforma za upravljanje dokumentima, pružajući funkcionalnosti za kreiranje, čuvanje, ažuriranje i brisanje dokumenata. Korisnici mogu pretraživati dokumente koristeći različite filtere poput naslova, sadržaja, autora, kategorija i tagova, a takođe imaju mogućnost preuzimanja dokumenata. Pored upravljanja dokumentima, aplikacija podržava kreiranje i upravljanje korisničkim nalogom, uključujući registraciju, prijavljivanje, odjavljivanje i resetovanje lozinke. Dodatno, korisnici mogu da komentarišu dokumente, a postoji i sistem kategorizacije i tagovanja dokumenata za bolju organizaciju i pretragu.

# Implementacija

## Model

U modelu Category.php, kategorija sadrži polje name koje čuva ime kategorije, i ima vezu mnogo-na-mnogo sa modelom Document, što znači da svaka kategorija može biti povezana sa više dokumenata, a svaki dokument može pripadati više kategorija.

Model Comment.php uključuje polja content, document\_id i user\_id, gde svaki komentar sadrži tekstualni sadržaj, referencu na dokument na koji se odnosi i korisnika koji ga je kreirao. Ovaj model ima veze mnogo-na-jedan sa modelima Document i User, što ukazuje da je svaki komentar povezan sa jednim dokumentom i jednim korisnikom.

U Document.php, dokument ima polja kao što su title, content, file\_path, author\_id, downloads, is\_public, category\_id i tags. Ovaj model ima mnogo-na-jedan vezu sa User (preko author\_id), mnogo-na-jedan vezu sa Category i mnogo-na-mnogo vezu sa Tag, što omogućava dokumentima da imaju više tagova i da pripadaju više kategorija.

Model Tag.php sadrži polja name, color i slug, i ima mnogo-na-mnogo vezu sa Document, omogućavajući svakom tagu da bude povezan sa više dokumenata.

Na kraju, model User.php obuhvata polja kao što su name, email, password, date\_of\_birth i bio, i ima veze jedan-na-mnogo sa Document i Comment, što znači da svaki korisnik može kreirati više dokumenata i komentara.

## Migracije

U ovom radu imamo sledece migracije:

Slika na kojoj se nalazi tekst, snimak ekrana, Font

Opis je automatski generisan

Slika 1 prikaz svih migracija

Migracija create\_documents\_table kreira tabelu documents sa kolonama za title, content, file\_path, author\_id, downloads, is\_public, category\_id, i tags.

<?php

use Illuminate\Database\Migrations\Migration;

use Illuminate\Database\Schema\Blueprint;

use Illuminate\Support\Facades\Schema;

return new class extends Migration

{

    /\*\*

     \* Run the migrations.

     \*

     \* @return void

     \*/

    public function up()

    {

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

            $table->id();

            $table->string('title');

            $table->text('content');

            $table->string('file\_path')->nullable();

            $table->unsignedBigInteger('author\_id');

            $table->unsignedBigInteger('category\_id');

            $table->integer('downloads')->default(0);

            $table->boolean('is\_public')->default(true);

            $table->text('tags');

            $table->timestamps();

            $table->softDeletes();

        });

    }

    /\*\*

     \* Reverse the migrations.

     \*

     \* @return void

     \*/

    public function down()

    {

        Schema::dropIfExists('documents');

    }

};

Migracija add\_foreign\_key\_to\_documents\_table dodaje strani ključ category\_id u tabelu documents, povezujući je sa tabelom categories.

<?php

use Illuminate\Database\Migrations\Migration;

use Illuminate\Database\Schema\Blueprint;

use Illuminate\Support\Facades\Schema;

return new class extends Migration

{

    /\*\*

     \* Run the migrations.

     \*

     \* @return void

     \*/

    public function up()

    {

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

            $table->foreign('author\_id')->references('id')->on('users');

            $table->foreign('category\_id')->references('id')->on('categories');

        });

    }

    /\*\*

     \* Reverse the migrations.

     \*

     \* @return void

     \*/

    public function down()

    {

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

            $table->dropForeign('author\_id');

        });

    }

};

Migracija add\_columns\_to\_tags\_table dodaje nove kolone color i slug u tabelu tags.

<?php

use Illuminate\Database\Migrations\Migration;

use Illuminate\Database\Schema\Blueprint;

use Illuminate\Support\Facades\Schema;

return new class extends Migration

{

    /\*\*

     \* Run the migrations.

     \*

     \* @return void

     \*/

    public function up()

    {

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

            $table->string('slug')->unique(); // Polje za jedinstveni slug

        });

    }

    /\*\*

     \* Reverse the migrations.

     \*

     \* @return void

     \*/

    public function down()

    {

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

            $table->dropColumn('slug');

        });

    }

};

Migracija add\_foreign\_key\_to\_comments\_table uključuje dodavanje stranih ključeva document\_id i user\_id u tabelu comments, povezujući je sa tabelama documents i users.

<?php

use Illuminate\Database\Migrations\Migration;

use Illuminate\Database\Schema\Blueprint;

use Illuminate\Support\Facades\Schema;

return new class extends Migration

{

    /\*\*

     \* Run the migrations.

     \*

     \* @return void

     \*/

    public function up()

    {

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

            $table->foreign('document\_id')->references('id')->on('documents');

            $table->foreign('user\_id')->references('id')->on('users');

        });

    }

    /\*\*

     \* Reverse the migrations.

     \*

     \* @return void

     \*/

    public function down()

    {

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

            $table->dropForeign('document\_id');

            $table->dropForeign('document\_id');

        });

    }

};

Migracija drop\_description\_column\_from\_categories\_table uklanja kolonu description iz tabele categories.

<?php

use Illuminate\Database\Migrations\Migration;

use Illuminate\Database\Schema\Blueprint;

use Illuminate\Support\Facades\Schema;

return new class extends Migration

{

    /\*\*

     \* Run the migrations.

     \*

     \* @return void

     \*/

    public function up()

    {

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

            $table->dropColumn('description');

        });

    }

    /\*\*

     \* Reverse the migrations.

     \*

     \* @return void

     \*/

    public function down()

    {

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

            $table->text('description');

        });

    }

};

Migracija add\_columns\_to\_users\_table dodaje kolone date\_of\_birth i bio u tabelu users.

<?php

use Illuminate\Database\Migrations\Migration;

use Illuminate\Database\Schema\Blueprint;

use Illuminate\Support\Facades\Schema;

return new class extends Migration

{

    /\*\*

     \* Run the migrations.

     \*

     \* @return void

     \*/

    public function up()

    {

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

            $table->date('date\_of\_birth');

            $table->text('bio');

        });

    }

    /\*\*

     \* Reverse the migrations.

     \*

     \* @return void

     \*/

    public function down()

    {

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

            $table->dropColumn('date\_of\_birth');

            $table->dropColumn('bio');

        });

    }

};

Migracija rename\_columns\_in\_comments\_table menja imena nekih kolona u tabeli comments.

<?php

use Illuminate\Database\Migrations\Migration;

use Illuminate\Database\Schema\Blueprint;

use Illuminate\Support\Facades\Schema;

return new class extends Migration

{

    /\*\*

     \* Run the migrations.

     \*

     \* @return void

     \*/

    public function up()

    {

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

            $table->renameColumn('tekst','content');

        });

    }

    /\*\*

     \* Reverse the migrations.

     \*

     \* @return void

     \*/

    public function down()

    {

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

            $table->renameColumn('content','tekst');

        });

    }

};

## Kontroleri

U ovom radu su kreirani sledeci kontroleri

Slika na kojoj se nalazi tekst, snimak ekrana, Font

Opis je automatski generisan

Slika 2 prikaz kreiranih kontrolera

U ovom AuthControlleru, imamo tri glavne metode za rad sa korisnicima: register, login i logout.

* Register: Ova metoda prima zahtev za registraciju korisnika. zahtev mora sadrzati korisnikovo ime, email, lozinku, datum rođenja i biografiju. Metoda koristi validaciju da bude sigurna da su svi potrebni podaci uneti i da su validni. Ako validacija ne prodje, vraca se odgovor sa greškama. Ako validacija prodje, korisnik se kreira u bazi podataka sa unetim podacima.
* Login: Ova metoda prima zahtev za prijavu korisnika koji mora sadrzati email i lozinku. Metoda koristi validaciju da bude sigurna da su svi potrebni podaci uneti i da su validni. Ako validacija ne prodje, vraca se odgovor sa greškama. Ako validacija prodje, korisnik se prijavljuje u aplikaciju. Ako prijava ne uspe, vraca se odgovor sa greškom. Ako prijava uspe, kreira se token za autentifikaciju i vraca se zajedno sa korisnikovim podacima.
* Logout: Ova metoda brise token za autentifikaciju trenutno prijavljenog korisnika. Nakon brisanja tokena, korisnik se automatski odjavljuje iz aplikacije.

<?php

namespace App\Http\Controllers;

use App\Models\User;

use Illuminate\Http\Request;

use Illuminate\Support\Facades\Auth;

use Illuminate\Support\Facades\Hash;

use Illuminate\Support\Facades\Validator;

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',

            'date\_of\_birth' => 'required|date',

            'bio' => 'required|string'

        ]);

        if ($validator->fails()) {

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

        }

        $user = User::create([

            'name' => $request->name,

            'email' => $request->email,

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

            'date\_of\_birth' => $request->date\_of\_birth,

            'bio' => $request->bio

        ]);

        return response()->json(['message' => 'User successfully registered', 'user' => $user], 201);

    }

    public function login(Request $request)

    {

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

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

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

        ]);

        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('auth\_token')->plainTextToken;

        return response()->json(['message' => 'Login successful', 'token' => $token,'user'=> $user]);

    }

    public function logout(Request $request)

    {

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

        return response()->json(['message' => 'Logout successful']);

    }

    public function changePassword(Request $request)

    {

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

            'current\_password' => 'required|string|min:8',

            'new\_password' => 'required|string|min:8|confirmed',

        ]);

        if ($validator->fails()) {

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

        }

        if (!Hash::check($request->current\_password, $request->user()->password)) {

            return response()->json(['message' => 'The current password is incorrect.'], 403);

        }

        $request->user()->update([

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

        ]);

        return response()->json(['message' => 'Password successfully changed.']);

    }

}

Dodatno, u ovom AuthControlleru imamo i metodu za promenu lozinke. Ova metoda prima zahtev za promenu lozinke.

Kontroler PasswordResetController ima dve funkcije koje omogućavaju korisnicima da resetuju svoju lozinku. Prva funkcija sendResetLinkEmail služi za slanje linka za resetovanje lozinke na email koji korisnik unese. Ova funkcija prvo validira email korisnika, a zatim koristi Laravelovu funkciju Password::sendResetLink za slanje linka. U zavisnosti od rezultata, vraća odgovarajući JSON odgovor sa porukom uspeha ili greške. Druga funkcija reset omogućava korisniku da postavi novu lozinku koristeći token dobijen putem emaila. Nakon validacije ulaznih podataka, koristi se Password::reset za promenu lozinke. Ako je resetovanje uspešno, korisnikov remember\_token se ažurira, a lozinka se čuva u hashiranom obliku u bazi podataka, nakon čega se vraća odgovor uspeha ili greške.

<?php

namespace App\Http\Controllers;

use App\Models\User;

use Carbon\Carbon;

use Illuminate\Auth\Events\PasswordReset;

use Illuminate\Http\Request;

use Illuminate\Support\Facades\DB;

use Illuminate\Support\Facades\Hash;

use Illuminate\Support\Facades\Password;

use Illuminate\Support\Str;

class PasswordResetController extends Controller

{

    public function sendResetLinkEmail(Request $request)

    {

      $request->validate(['email' => 'required|email']);

        $status = Password::sendResetLink(

            $request->only('email')

        );

        return $status === Password::RESET\_LINK\_SENT

                    ? response()->json(['message' => \_\_($status)])

                    : response()->json(['email' => \_\_($status)], 422);

    }

    public function reset(Request $request)

    {

        $request->validate([

            'token' => 'required',

            'email' => 'required|email',

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

        ]);

        $status = Password::reset(

            $request->only('email', 'password', 'password\_confirmation', 'token'),

            function ($user, $password) {

                $user->forceFill([

                    'password' => Hash::make($password)

                ])->setRememberToken(Str::random(60));

                $user->save();

                event(new PasswordReset($user));

            }

        );

        return $status === Password::PASSWORD\_RESET

                    ? response()->json(['message' => \_\_($status)])

                    : response()->json(['email' => \_\_($status)], 422);

    }

}

DocumentController sadrži nekoliko metoda za upravljanje dokumentima. Metoda index vraća sve dokumente iz baze podataka. show vraća određeni dokument na osnovu ID-a. store služi za kreiranje novog dokumenta, gde se prvo vrši validacija podataka, a zatim čuva dokument u bazi sa opcionalnim uploadom fajla. Metoda update omogućava ažuriranje postojećeg dokumenta, uključujući i zamenu fajla ukoliko je novi fajl poslat. destroy briše dokument, uz proveru da li je dokument već soft-deleted. search omogućava filtriranje dokumenata po različitim kriterijumima kao što su naslov, sadržaj, autor, kategorija i tagovi. download omogućava preuzimanje fajla dokumenta, uz proveru da li fajl postoji. Takođe, postoji pomoćna funkcija getMimeType koja vraća MIME tip fajla na osnovu njegove ekstenzije.

<?php

namespace App\Http\Controllers;

use App\Models\Document;

use Illuminate\Http\Request;

use Illuminate\Support\Facades\Storage;

use Illuminate\Support\Facades\Validator;

class DocumentController extends Controller

{

    public function index()

    {

        $documents = Document::all();

        return response()->json($documents);

    }

    public function show($id)

    {

        $document = Document::findOrFail($id);

        return response()->json($document);

    }

    public function store(Request $request)

    {

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

            'title' => 'required|max:255',

            'content' => 'required',

            'author\_id' => 'required|exists:users,id',

            'category\_id' => 'required|exists:categories,id',

            'tags' => 'required|array',      //Ovo pravilo zahteva da polje 'tags' bude prisutno u zahtevu (ne smije biti prazno ili null) i da bude niz (array).

            'tags.\*' => 'distinct|exists:tags,id',  //Ovo pravilo se primenjuje na svaki element niza 'tags'.  //distinct' proverava da svaki tag u nizu bude jedinstven, tj. da se ne ponavlja više puta.

            'file' => 'required|file', // Pravilo za validaciju uploadovanog fajla

         //   'is\_public' => 'required',

            'downloads' => 'required|numeric',

        ]);

        if ($validator->fails()) {

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

        }

        $documentData = $validator->validated();

        // Obrada uploada fajla

        if ($request->hasFile('file')) {

            $file = $request->file('file');

            $filePath = $file->store('documents', 'public'); // Sačuvaj fajl i vrati putanju

            $documentData['file\_path'] = $filePath;

        }

        $documentData['is\_public'] = 0;

        // Kreiraj dokument i postavi kategoriju

        $document = Document::create($documentData);

        // Kako su tagovi predstavljeni kao niz, treba ih konvertovati u string pre snimanja

       // $document->tags()->sync($request->tags);   //Ova linija koda se koristi za uspostavljanje veze many-to-many između Document i Tag. Metoda sync prima niz ID-ova tagova i ažurira odgovarajuću pivot tabelu, uklanjajući sve postojeće veze i dodajući nove. S obzirom na vašu strukturu, ovo je ispravno za rad sa tagovima.

        return response()->json($document, 201);

    }

    public function update(Request $request, $id)

    {

        $document = Document::findOrFail($id);

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

            'title' => 'sometimes|required|max:255',

            'content' => 'sometimes|required',

            'author\_id' => 'sometimes|required|exists:users,id',

            'category\_id' => 'sometimes|required|exists:categories,id',

            'tags' => 'sometimes|required|array',

            'tags.\*' => 'distinct|exists:tags,id',

            'file' => 'sometimes|file',

            'is\_public' => 'sometimes|boolean',

            'downloads' => 'sometimes|numeric',

        ]);

        if ($validator->fails()) {

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

        }

        $documentData = $validator->validated();

        // Obrada uploada fajla

        if ($request->hasFile('file')) {

            // Pre nego što sačuvamo novi fajl, trebalo bi da obrisemo stari

            Storage::delete($document->file\_path);

            $file = $request->file('file');

            $filePath = $file->store('documents', 'public'); // Sačuvaj fajl i vrati putanju

            $documentData['file\_path'] = $filePath;

        }

        // Konvertovanje niza tagova u string pre snimanja, ako su tagovi prosleđeni

        if (isset($documentData['tags'])) {

            $documentData['tags'] = implode(',', $request->tags);

        }

        $document->update($documentData);

        return response()->json($document);

    }

    public function destroy($id)  //posto nas model koristi softDeletes, onda metoda destroy mora da bude implementirana na malo drugaciji nacin

    {

        $document = Document::withTrashed()->findOrFail($id);

        if ($document->trashed()) {

            //  dokument je već obrisan

            return response()->json(['message' => 'Document has already been deleted.'], 404);

        } else {

            $document->delete();

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

        }

    }

    public function search(Request $request)

{

    // Počinjemo sa kreiranjem osnovnog upita

    $query = Document::query();

    // Filtriramo po naslovu ako je prosleđen kao parametar

    if ($request->has('title')) {

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

    }

    // Filtriramo po sadržaju ako je prosleđen kao parametar

    if ($request->has('content')) {

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

    }

    // Filtriramo po autoru ako je prosleđen kao parametar

    if ($request->has('author\_id')) {

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

    }

    // Filtriramo po kategoriji ako je prosleđen kao parametar

    if ($request->has('category\_id')) {

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

    }

    // Filtriramo po tagovima ako su prosleđeni kao parametar

    if ($request->has('tags')) {

        $tags = $request->tags;

        $query->whereHas('tags', function ($query) use ($tags) {

            $query->whereIn('id', $tags);

        });

    }

    // Filtriramo po dostupnosti ako je prosleđen kao parametar

    if ($request->has('is\_public')) {

        $query->where('is\_public', $request->is\_public);

    }

    // Dobijamo filtrirane dokumente

    $documents = $query->get();

    // Vraćamo dokumente kao JSON odgovor

    return response()->json($documents);

    }

    public function download($id)

    {

        $document = Document::findOrFail($id);

        $filePath = storage\_path('app/public/' . $document->file\_path);

        if (!file\_exists($filePath)) {

            return response()->json(['message' => 'File not found.'], 404);

        }

        $fileName = basename($filePath);

        $fileExtension = pathinfo($filePath, PATHINFO\_EXTENSION);

        // Određivanje MIME tipa na osnovu ekstenzije fajla

        $mimeType = $this->getMimeType($fileExtension);

        // Preuzimanje fajla sa ispravnim MIME tipom

        return response()->download($filePath, $fileName, [

            'Content-Type' => $mimeType,

            'Content-Disposition' => 'attachment; filename="' . $fileName . '"'

        ]);

    }

    // Pomoćna funkcija za dobijanje MIME tipa

    private function getMimeType($extension)

    {

        $mimeTypes = [

            'pdf' => 'application/pdf',

            'jpg' => 'image/jpeg',

            'jpeg' => 'image/jpeg',

            'png' => 'image/png',

            'txt' => 'text/plain',

        ];

        return $mimeTypes[$extension] ?? 'application/octet-stream';

    }

}

## Api rute

Prva ruta je GET /api/documents/search, koja koristi DocumentController za pretragu dokumenata po kriterijumima kao što su naslov, kategorija i oznake. Druga ruta, GET /api/documents/{document}/download, omogućava preuzimanje dokumenta i takođe koristi DocumentController.

Za resetovanje lozinke postoje dve rute: POST /api/forgot-password i POST /api/reset-password, koje koriste PasswordResetController za slanje linka za resetovanje lozinke i sam proces resetovanja lozinke.

Takođe, postoje rute za autentifikaciju korisnika: POST /api/register i POST /api/login koje koriste AuthController za registraciju novih korisnika i prijavljivanje postojećih korisnika.

Unutar grupe koja zahteva autentifikaciju korisnika (auth:sanctum), nalaze se dodatne rute. Postoji ruta POST /api/logout za odjavljivanje korisnika, kao i više ruta za upravljanje dokumentima, komentarima, oznakama i kategorijama kroz odgovarajuće kontrolere (DocumentController, CommentController, TagController, CategoryController). Na kraju, postoji ruta POST /api/changePassword za promenu korisničke lozinke. Ove rute omogućavaju kreiranje, čitanje, ažuriranje i brisanje (CRUD operacije) nad pomenutim resursima.

<?php

use App\Http\Controllers\AuthController;

use App\Http\Controllers\CategoryController;

use App\Http\Controllers\CommentController;

use App\Http\Controllers\DocumentController;

use App\Http\Controllers\PasswordResetController;

use App\Http\Controllers\TagController;

use Illuminate\Http\Request;

use Illuminate\Support\Facades\Route;

/\*

|--------------------------------------------------------------------------

| API Routes

|--------------------------------------------------------------------------

|

| Here is where you can register API routes for your application. These

| routes are loaded by the RouteServiceProvider within a group which

| is assigned the "api" middleware group. Enjoy building your API!

|

\*/

Route::get('/documents/search', [DocumentController::class, 'search']);  //GET /api/documents/search?title=izveštaj&category\_id=3&tags[]=1&tags[]=5&is\_public=1

Route::get('/documents/{document}/download', [DocumentController::class, 'download']);

Route::post('/forgot-password', [PasswordResetController::class, 'sendResetLinkEmail']);

Route::post('/reset-password', [PasswordResetController::class, 'reset']);

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

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

Route::middleware('auth:sanctum')->group(function () {

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

    Route::apiResource('documents', DocumentController::class);

        Route::apiResource('comments', CommentController::class);

        Route::apiResource('tags', TagController::class);

        Route::apiResource('categories', CategoryController::class);

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

});

# REST API

|  |  |
| --- | --- |
| Opis funkcije | Prikaz svih dokumenata |
| HTTP metoda | GET |
| URL | /api/ documents |
| URL parametri | nema |
| HTTP body parametri | nema |
| Format HTTP body parametara | JSON |
| Izlazni parametri | [      {          "id": 4,          "title": "dasda",          "content": "dsada",          "file\_path": "documents/zJxbzLWt4le7K34az25zryHyoGwAw3657r7FsYdD.png",          "author\_id": 1,          "category\_id": 1,          "downloads": 5,          "is\_public": 0,          "tags": [              "1"          ],          "created\_at": "2024-01-26T21:56:30.000000Z",          "updated\_at": "2024-01-26T21:56:30.000000Z",          "deleted\_at": **null**      },      {          "id": 6,          "title": "das",          "content": "da",          "file\_path": "documents/IabI7QfnAkOO6SL4z4UcQxnuvM4olVTB4mVeFcBB.png",          "author\_id": 5,          "category\_id": 1,          "downloads": 0,          "is\_public": 0,          "tags": [              "1",              "2"          ],          "created\_at": "2024-01-29T20:42:43.000000Z",          "updated\_at": "2024-01-29T20:42:43.000000Z",          "deleted\_at": **null**      },  …  **}** |
| Format izlaznih parametara | Application/json |
|  |  |
| Opis funkcije | Kreiranje dokumenta |
| HTTP metoda | POST |
| URL | Api/documents |
| URL parametri | ?title=naslov&content=sadrzaj&author\_id=1&category\_id=1&tags[]=[1]&downloads=0 |
| HTTP body parametri |  |
| Format HTTP body parametara | nema |
| Izlazni parametri | {      "title": "naslov",      "content": "sadrzaj",      "author\_id": "1",      "category\_id": "1",      "downloads": "0",      "tags": [          "1"      ],      "file\_path": "documents/dBHGUnp58sDP0Ve2TSOWG7xq13yyC8PcrHlF4qMb.png",      "is\_public": 0,      "updated\_at": "2024-01-30T13:55:12.000000Z",      "created\_at": "2024-01-30T13:55:12.000000Z",      "id": 12  } |
| Format izlaznih parametara | application/json |

|  |  |
| --- | --- |
| Opis funkcije | Ažuriranje dokumenta |
| HTTP metoda | PUT |
| URL | /api/documents |
| URL parametri | 12?title=naslov&content=sadrzaj&author\_id=1&category\_id=1&downloads=0 |
| HTTP body parametri | (nema) |
| Format HTTP body parametara | (nema) |
| Izlazni parametri | {      "id": 12,      "title": "naslov",      "content": "sadrzaj",      "file\_path": "documents/dBHGUnp58sDP0Ve2TSOWG7xq13yyC8PcrHlF4qMb.png",      "author\_id": "1",      "category\_id": "1",      "downloads": "0",      "is\_public": 0,      "tags": [          "1"      ],      "created\_at": "2024-01-30T13:55:12.000000Z",      "updated\_at": "2024-01-30T13:55:12.000000Z",      "deleted\_at": **null**  } |
| Format izlaznih parametara | application/json |

|  |  |
| --- | --- |
| Opis funkcije | Brisanje dokumenta |
| HTTP metoda | DELETE |
| URL | /api/documents |
| URL parametri | 12 |
| HTTP body parametri | (nema) |
| Format HTTP body parametara | (nema) |
| Izlazni parametri | -No Content- |
| Format izlaznih parametara | application/json |
| Opis funkcije | Pretraga dokumenata |
| HTTP metoda | GET |
| URL | /api/documents |
| URL parametri | ?author\_id=1 |
| HTTP body parametri | nema |
| Format HTTP body parametara | JSON |
| Izlazni parametri | [      {          "id": 4,          "title": "dasda",          "content": "dsada",          "file\_path": "documents/zJxbzLWt4le7K34az25zryHyoGwAw3657r7FsYdD.png",          "author\_id": 1,          "category\_id": 1,          "downloads": 5,          "is\_public": 0,          "tags": [              "1"          ],          "created\_at": "2024-01-26T21:56:30.000000Z",          "updated\_at": "2024-01-26T21:56:30.000000Z",          "deleted\_at": **null**      }  ] |
| Format izlaznih parametara | Application/json |
|  |  |
| Opis funkcije | Kreiranje komenatara |
| HTTP metoda | POST |
| URL | /api/comments |
| URL parametri | content=sadrzaj&user\_id=1&document\_id=11 |
| HTTP body parametri | (nema) |
| Format HTTP body parametara | (nema) |
| Izlazni parametri | {      "content": "sadrzaj",      "document\_id": "11",      "user\_id": "1",      "updated\_at": "2024-01-30T14:03:48.000000Z",      "created\_at": "2024-01-30T14:03:48.000000Z",      "id": 4  } |
| Format izlaznih parametara | application/json |

|  |  |
| --- | --- |
| Opis funkcije | Ažuriranje komentara |
| HTTP metoda | PUT |
| URL | /api/ comments |
| URL parametri | 4?content=sadrzaj&user\_id=1&document\_id=11 |
| HTTP body parametri | (nema) |
| Format HTTP body parametara | (nema) |
| Izlazni parametri | {      "id": 4,      "content": "sadrzaj",      "document\_id": "11",      "user\_id": "1",      "created\_at": "2024-01-30T14:03:48.000000Z",      "updated\_at": "2024-01-30T14:03:48.000000Z"  } |
| Format izlaznih parametara | application/json |

|  |  |
| --- | --- |
| Opis funkcije | Brisanje komenatara |
| HTTP metoda | DELETE |
| URL | /api/ comments |
| URL parametri | 4 |
| HTTP body parametri | (nema) |
| Format HTTP body parametara | (nema) |
| Izlazni parametri | --No Content-- |
| Format izlaznih parametara | application/json |

|  |  |
| --- | --- |
| Opis funkcije | Registracija korisnika |
| HTTP metoda | POST |
| URL | /api/register |
| URL parametri | ?name=korisnik&email=korisnik@gmail.com&password=password&date\_of\_birth=2024-01-01&bio=biografija&password\_confirmation=password |
| HTTP body parametri | (nema) |
| Format HTTP body parametara | (nema) |
| Izlazni parametri | {      "message": "User successfully registered",      "user": {          "name": "korisnik",          "email": "korisnik@gmail.com",          "date\_of\_birth": "2024-01-01",          "bio": "biografija",          "updated\_at": "2024-01-30T14:08:06.000000Z",          "created\_at": "2024-01-30T14:08:06.000000Z",          "id": 6      }  } |
| Format izlaznih parametara | application/json |

|  |  |
| --- | --- |
| Opis funkcije | Prijava korisnika |
| HTTP metoda | POST |
| URL | /api/login |
| URL parametri | ?email=korisnik@gmail.com&password=password |
| HTTP body parametri | (nema) |
| Format HTTP body parametara | (nema) |
| Izlazni parametri | {      "message": "Login successful",      "token": "10|isCOm8BedOz0ykBCehxNxzgtDH27RGuJyLLClqCo48f05a3d",      "user": {          "id": 6,          "name": "korisnik",          "email": "korisnik@gmail.com",          "email\_verified\_at": **null**,          "created\_at": "2024-01-30T14:08:06.000000Z",          "updated\_at": "2024-01-30T14:08:06.000000Z",          "date\_of\_birth": "2024-01-01",          "bio": "biografija"      }  } |
| 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": "Logout successful"  } |
| Format izlaznih parametara | application/json |

# Korisničko uputstvo

Registracija korisnika

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

Opis je automatski generisan

Login korisnika

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

Opis je automatski generisan

Token koji smo dobili kopiramo i smestamo u karticu za autorizaciju

Slika na kojoj se nalazi tekst, snimak ekrana, Font

Opis je automatski generisan

Prikaz svih dokumenata

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

Opis je automatski generisan

Kreiranje dokumenta

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

Opis je automatski generisan

Azuriranje dokumenta

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

Opis je automatski generisan

Brisanje dokumenta

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

Opis je automatski generisan

Pretraga dokumenata

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

Opis je automatski generisan

Preuzimanje dokumenata

^Slika na kojoj se nalazi tekst, snimak ekrana, dizajn

Opis je automatski generisan

Prikaz svih komentara

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

Opis je automatski generisan

Kreiranje komentara

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

Opis je automatski generisan

Azuriranje komentara

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

Opis je automatski generisan

Brisanje komentara

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

Opis je automatski generisan

Odjava

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

Opis je automatski generisan