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](#_Toc118213520)

[2 Implementacija 3](#_Toc118213521)

[3 REST API 3](#_Toc118213522)

[4 Korisničko uputstvo 3](#_Toc118213523)

# Korisnički zahtev

## Opis aplikacije

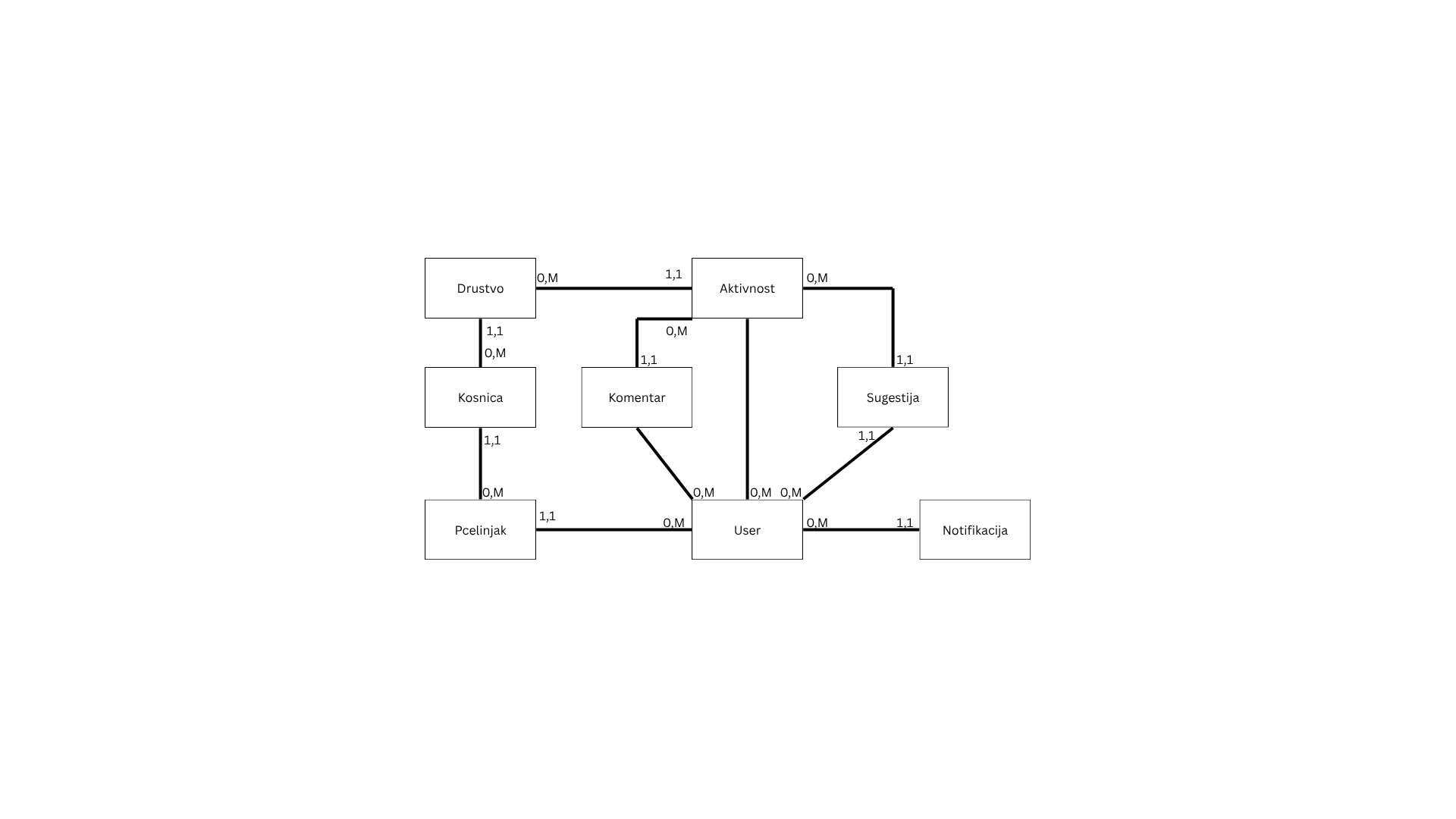
Ova aplikacija predstavlja sveobuhvatan alat za upravljanje pčelinjacima, košnicama i pčelinjim društvima. Korisnici, odnosno pčelari, mogu lako pratiti svoje pčelinjake i detaljno evidentirati svaku košnicu i društvo koje u njima živi. Aplikacija omogućava beleženje ključnih podataka kao što su starost matice, jačina društva i datum formiranja društva, što pomaže u praćenju zdravlja i produktivnosti pčela.

Kroz sistem aktivnosti korisnici planiraju, prate i dokumentuju različite radove i intervencije u pčelinjacima — kao što su pregledi, hranjenje, tretmani protiv bolesti i druge važne zadatke. Uz to, korisnici mogu dodavati komentare i sugestije vezane za svaku aktivnost, što olakšava komunikaciju i praćenje zapažanja i preporuka tokom vremena.

Aplikacija takođe šalje notifikacije i podsetnike kako bi pčelari bili pravovremeno obavešteni o važnim događajima ili obavezama. Na taj način se povećava efikasnost upravljanja i smanjuje mogućnost propusta u negovanju pčelinjih društava.

Ovaj sistem je namenjen da pomogne pčelarima svih nivoa da na jednostavan i organizovan način vode evidenciju i unaprede brigu o svojim pčelama, što može doprineti boljoj produktivnosti i zdravlju pčelinjaka.

Za potrebe kreiranja ove aplikacije fomulisan se konceptualni model, odnosno prošireni model objekti-veze, prikazan na slici ispod.



U nastavnu su objašnjeni svi koncepti modela:

1. User – predtsvalja korisnika sistema, tj. vlasnika pčelinjaka;
2. Pčelinjak – mesto gde se nalaze košnice, odnosno fizička lokacija koju poseduje korisnik;
3. Košnica – pojedinačna kutija, odnosno stanište u pčelinjaku u kojoj živi pčelinje društvo;
4. Društvo – odnosi se na pčelinje društvo, tj. zajednicu pčela koje nastanjuju košnicu, pri čemu svaka košnica može imati jedno ili više društava;
5. Aktivnost – predstavlja aktivnost vezanu za pčelinje drušvo, kao što je pregled košnice, hranjenje pčela, tretman protiv bolesti i slično;
6. Komentar – komentar koje korisnik ostavlja vezano za neku aktivnost, kao što su zapažanja, beleške i slično;
7. Sugestija – predlog, ideja ili preporuka vezana za neku aktivnost;
8. Notifikacija – obaveštenja koja korisnik može da prima, kao što su podsetnici za aktivnosti i slično.

Na osnovu konceptualnog modela moguće je kreirati relacioni model koji će kasnije poslužiti za projektovanje baze podataka. U nastavku je dat relacioni model.

* User(id, name, email, password)
* Pcelinjak(id, naziv, lokacija, user\_id)
* Kosnica(id, oznaka, tip, status, pcelinjak\_id)
* Drustvo(id, kosnica\_id, matica\_starost, jacina, datum\_formiranja)
* Aktivnost(id, naziv, opis, tip, pocetak, kraj, status, drustvo\_id, user\_id, notifikacija\_poslata)
* Komentar(id, sadrzaj, datum, aktivnost\_id, user\_id)
* Sugestija(id, poruka, datum\_kreiranja, user\_id, aktivnost\_id)
* Notifikacija(id, tekst, datum\_slanja, status, user\_id)

## Slučajevi korišćenja

Prilikom izrade aplikacije identifikovani slučajevi korišćenja prikazani u tabeli.

|  |  |  |  |
| --- | --- | --- | --- |
| Redni broj | Naziv | Preduslovi | Kriterijum pretrage |
| 1 | Registracija korisnika |  |  |
| 2 | Pregled korisnika | Ulogovani korisnik može pregledati svoje podatke. |  |
| 3 | Prijava korisnika |  |  |
| 4 | Ažuriranje korisnika | Ulogovani korisnik može izmeniti svoje podatke. |  |
| 5 | Kreranje pčelinjaka | Ulogovani korisnik može izvršiti kreiranje. |  |
| 6 | Pregled pčelinjaka | Ulogovani korisnik može pregledati svoje pčelinjake. |  |
| 7 | Brisanje pčelinjaka | Ulogovani korisnik može obrisati svoje pčelinjake. |  |
| 8 | Ažuriranje pčelinjaka | Ulogovani korisnik može izmeniti svoje pčelinjake. |  |
| 9 | Kreiranje košnice | Ulogovani korisnik može izvršiti kreiranje. |  |
| 10 | Pregled košnica | Ulogovani korisnik može pregledati svoje košnice. |  |
| 11 | Ažuriranje košnice | Ulogovani korisnik može izmeniti svoje košnice. |  |
| 12 | Brisanje košnice | Ulogovani korisnik može obrisati svoje košnice. |  |
| 13 | Kreiranje društva | Ulogovani korisnik može izvršiti kreiranje. |  |
| 14 | Pregled društva | Ulogovani korisnik može pregledati svoja društva. |  |
| 15 | Ažuriranje društva | Ulogovani korisnik može izmeniti svoja društva. |  |
| 16 | Brisanje društva | Ulogovani korisnik može obrisati svoja društva. |  |
| 17 | Kreiranje aktivnosti | Ulogovani korisnik može izvršiti kreiranje. |  |
| 18 | Pregled aktivnosti | Ulogovani korisnik može pregledati svoje aktivnosti. |  |
| 19 | Ažuriranje aktivnosti | Ulogovani korisnik može izmeniti svoje aktivnosti. |  |
| 20 | Brisanje aktivnosti | Ulogovani korisnik može obrisati svoje aktivnosti. |  |
| 21 | Kreiranje komentara | Ulogovani korisnik može izvršiti kreiranje. |  |
| 22 | Pregled komenrara | Ulogovani korisnik može pregledati svoje komentare. |  |
| 23 | Brisanje komentara | Ulogovani korisnik može obrisati svoje komentare. |  |
| 24 | Kreiranje sugestija | Ulogovani korisnik može izvršiti kreiranje. |  |
| 25 | Pregled sugestija | Ulogovani korisnik može pregledati svoje sugestije. |  |
| 26 | Ažuriranje sugestija | Ulogovani korisnik može izmeniti svoje sugestije. |  |
| 27 | Brisanje sugestija | Ulogovani korisnik može obrisati svoje sugestije. |  |

SK: Kreiranje aktivnosti

Naziv SK:  
Kreiranje aktivnosti

Aktori SK:  
Korisnik (User)

Učesnici SK:  
Korisnik i sistem

Preduslov:  
Sistem je uključen, a korisnik je uspešno ulogovan i ima pravo pristupa društvu kojem aktivnost pripada. Sistem prikazuje formu za kreiranje aktivnosti.

Osnovni scenario SK:

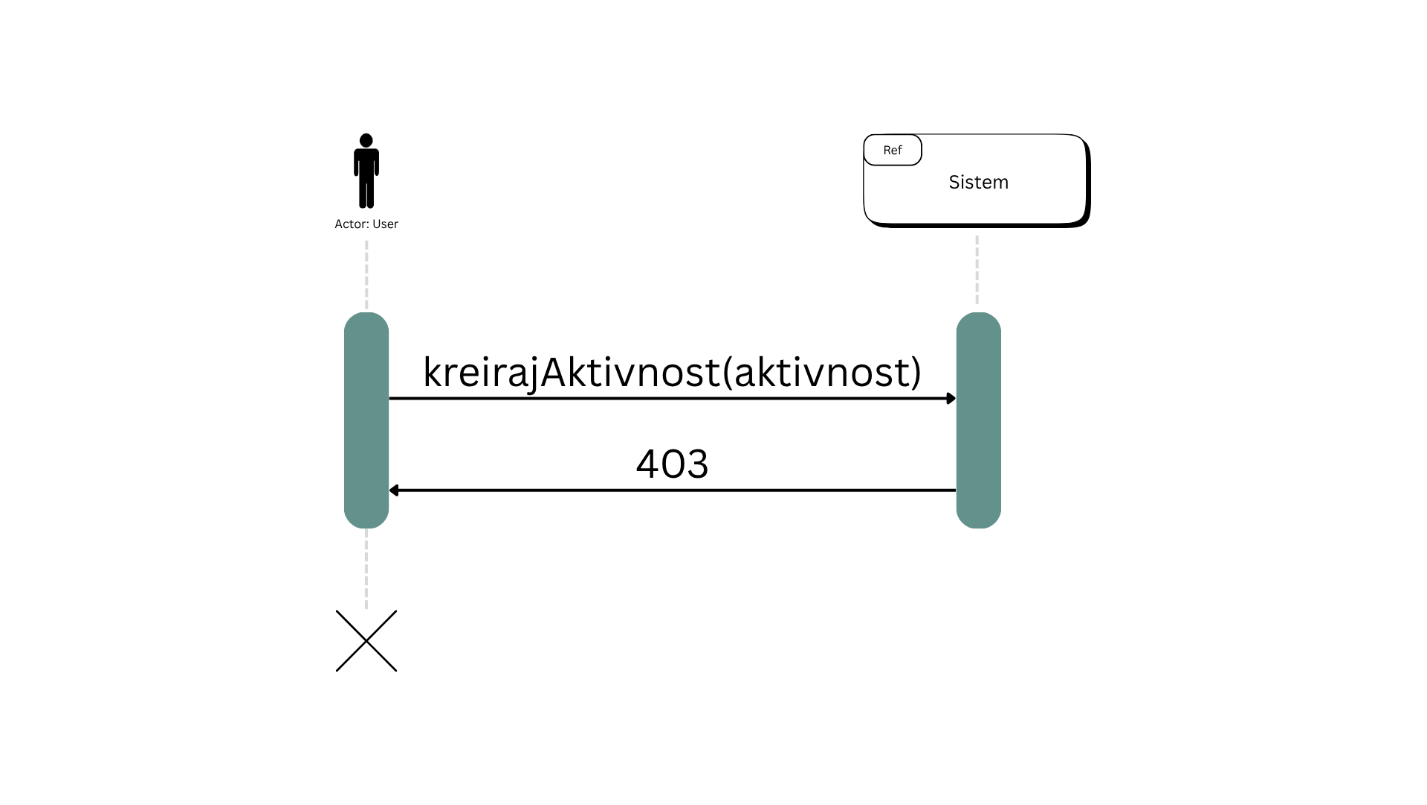
1. Korisnik unosi podatke o aktivnosti: naziv, opis, tip, početak i kraj, status i društvo kome aktivnost pripada. (APUSO)
2. Korisnik proverava da li su podaci ispravno uneti i po potrebi ih koriguje. (ANSO)
3. Korisnik šalje zahtev sistemu da sačuva novu aktivnost. (APSO)
4. Sistem validira podatke i pamti aktivnost u bazi. (SO)
5. Sistem prikazuje korisniku poruku: “Aktivnost uspešno kreirana” (IA)



Alternativni scenariji:  
3.1 Ukoliko sistem utvrdi da nisu popunjena sva obavezna polja ili da su podaci neispravni, prikazuje prilagođenu poruku. Prekida se izvršenje scenarija. (IA)



3.2 Ukoliko korisnik nema pravo da kreira aktivnost za dato društvo, sistem prikazuje 403 grešku i prekida izvršenje. (IA)



SK: Pregled aktivnosti

Naziv SK:  
Pregled aktivnost

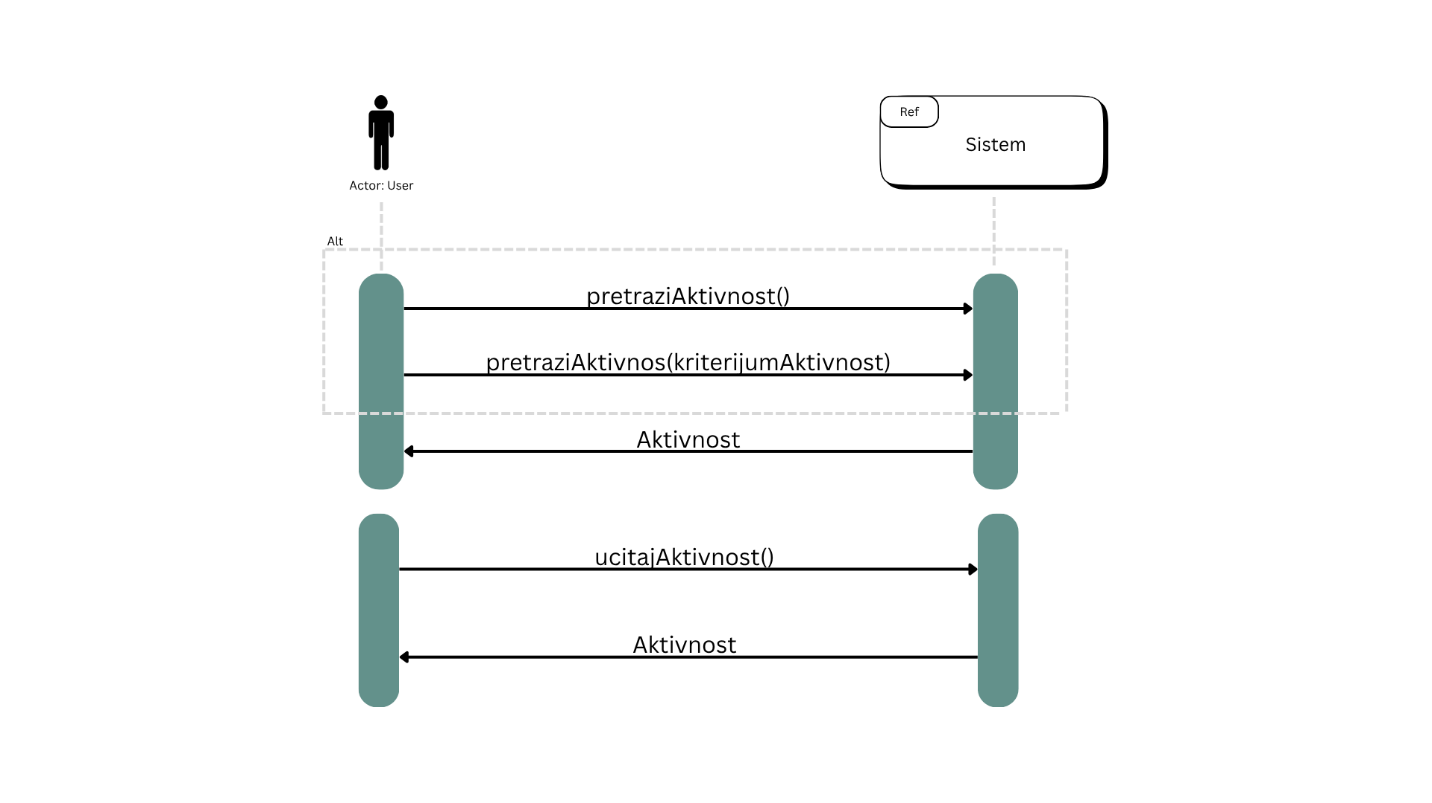
Aktori SK:  
Korisnik (User)

Učesnici SK:  
Korisnik i sistem

Preduslov:  
Sistem je uključen, korisnik je ulogovan i ima pravo pristupa aktivnostima. Sistem prikazuje formu za rad sa aktivnostima.

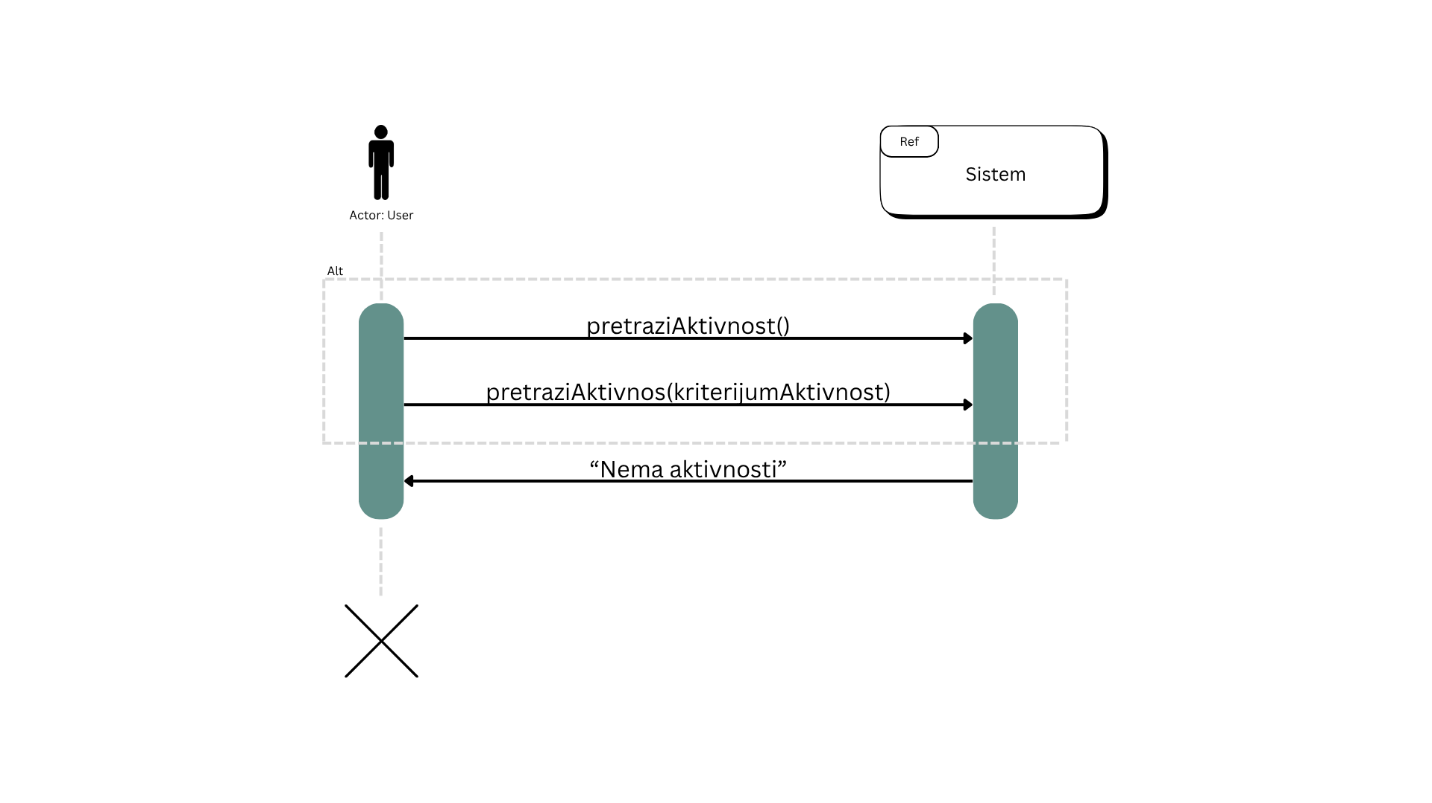
Osnovni scenario SK:

1. Korisnik unosi vrednost (kriterijum) po kojoj pretražuje aktivnosti, npr. naziv, status, datum početka ili kraja. (APUSO)
2. Korisnik poziva sistem da pronađe aktivnosti po zadatoj vrednosti. (APSO)
3. Sistem pretražuje aktivnosti u bazi na osnovu unetog kriterijuma. (SO)
4. Sistem prikazuje korisniku listu pronađenih aktivnosti. (IA)
5. Korisnik bira željenu aktivnost sa prikazane liste. (APUSO)
6. Korisnik traži od sistema da učita detalje odabrane aktivnosti. (APSO)
7. Sistem učitava podatke o odabranoj aktivnosti. (SO)
8. Sistem prikazuje korisniku detaljne podatke o aktivnosti. (IA)



Alternativni scenariji:

4.1 Ukoliko sistem ne može da pronađe aktivnosti po zadatom kriterijumu, prikazuje poruku: “Nema aktivnosti!”. Prekida se izvršenje scenarija. (IA)



SK: Ažuriranje aktivnosti

Naziv SK:  
Ažuriranje aktivnosti

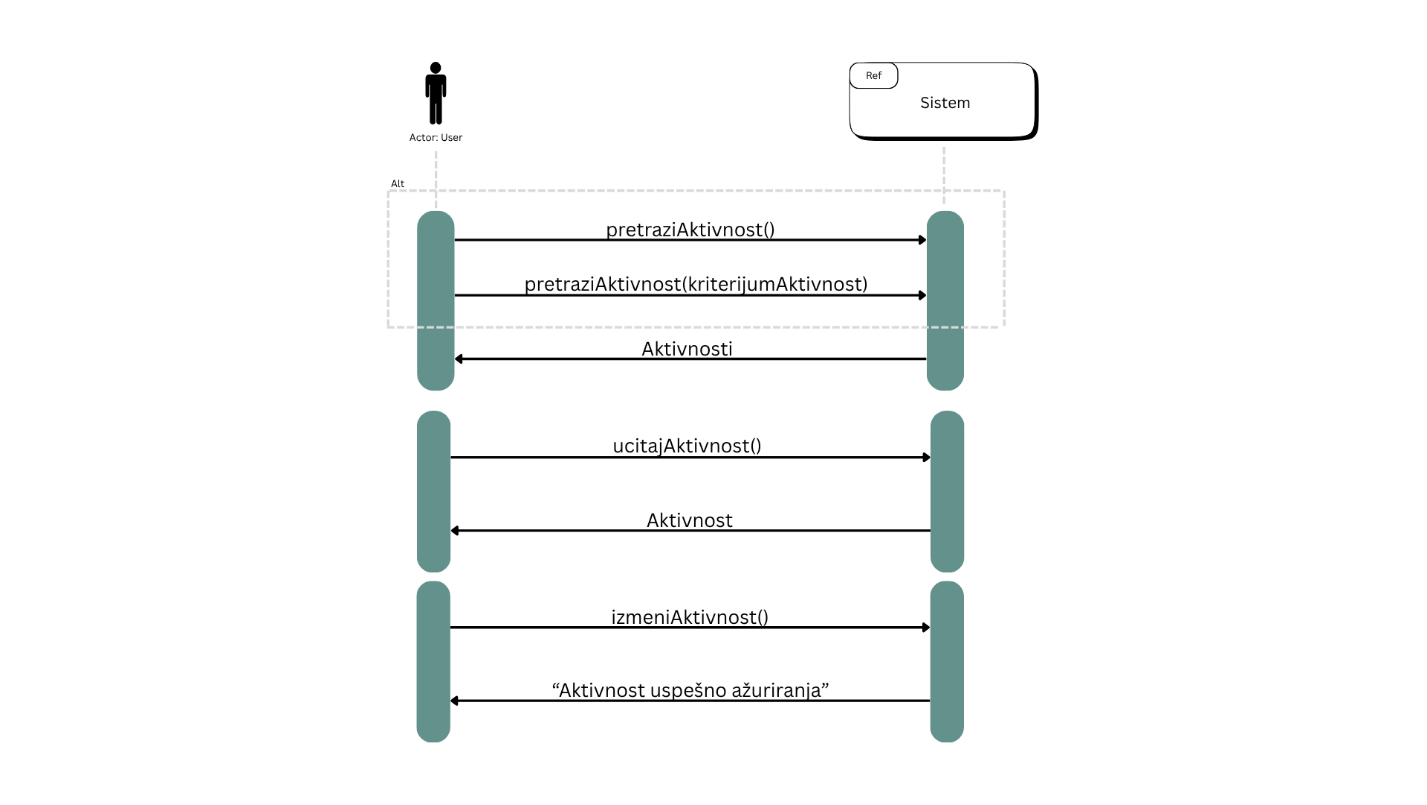
Aktori SK:  
Korisnik (User)

Učesnici SK:  
Korisnik i sistem

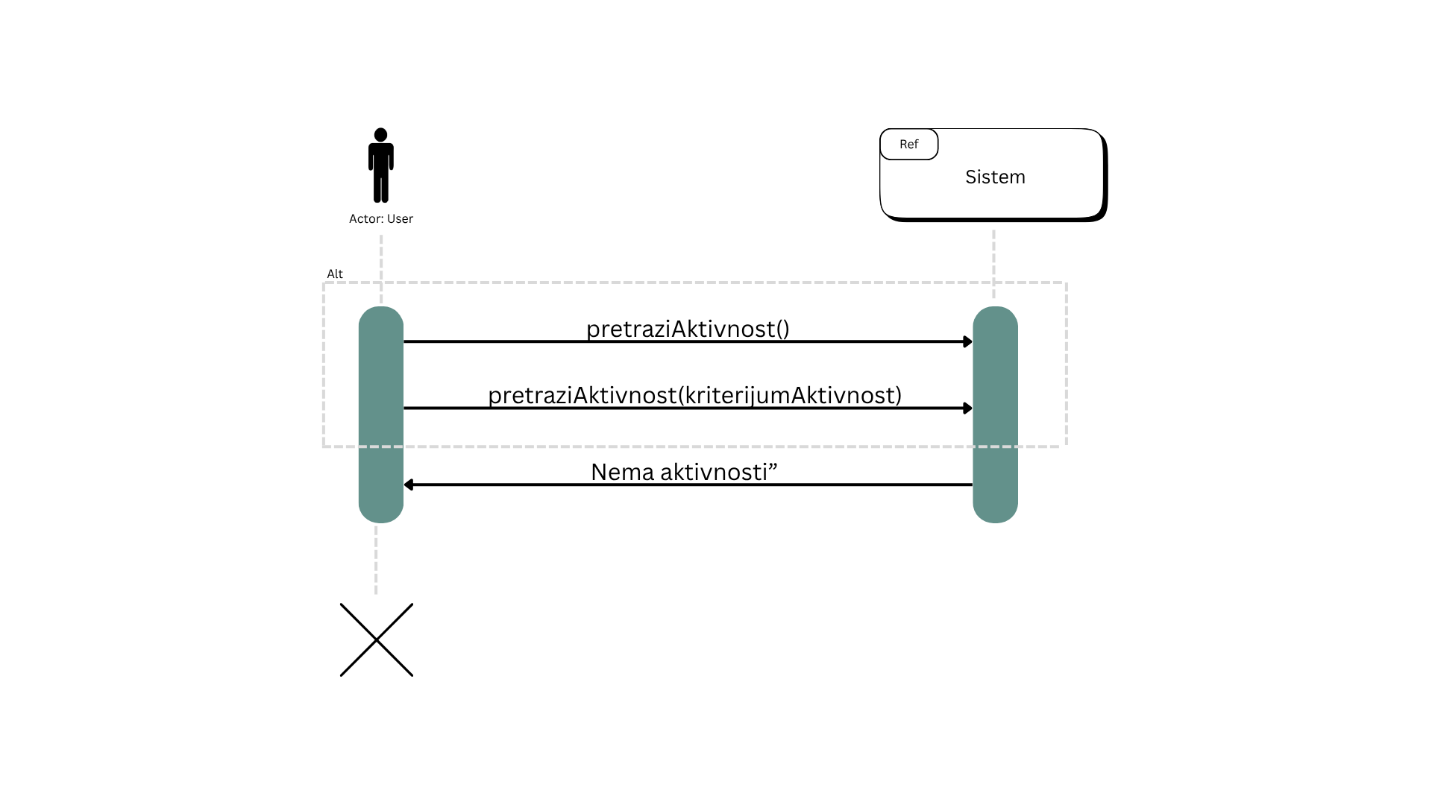
Preduslov:  
Sistem je uključen, korisnik je ulogovan i ima pravo pristupa aktivnostima. Sistem prikazuje formu za rad sa aktivnostima.

Osnovni scenario SK:

1. Korisnik unosi kriterijum za pretragu aktivnosti (npr. naziv, status, datum). (APUSO)
2. Korisnik poziva sistem da pronađe aktivnosti prema unetom kriterijumu. (APSO)
3. Sistem pretražuje bazu aktivnosti po zadatom kriterijumu. (SO)
4. Sistem prikazuje korisniku listu pronađenih aktivnosti. (IA)
5. Korisnik bira jednu aktivnost sa liste. (APUSO)
6. Korisnik traži od sistema da učita detalje odabrane aktivnosti. (APSO)
7. Sistem učitava podatke o aktivnosti iz baze. (SO)
8. Sistem prikazuje korisniku detalje o odabranoj aktivnosti. (IA)
9. Korisnik unosi izmene u podatke aktivnosti (npr. naziv, opis, status, datume). (APUSO)
10. Korisnik proverava da li su izmene ispravno unešene. (ANSO)
11. Korisnik šalje zahtev sistemu da sačuva izmene. (APSO)
12. Sistem ažurira podatke aktivnosti u bazi. (SO)
13. Sistem prikazuje korisniku poruku: “Aktivnost uspešno ažurirana!”. (IA)



4.1 Ukoliko sistem ne može da pronađe aktivnosti, prikazuje poruku: “Nema aktivnosti!”. Prekida se izvršenje scenarija. (IA)



SK: Brisanje aktivnosti

Naziv SK:  
Brisanje aktivnosti

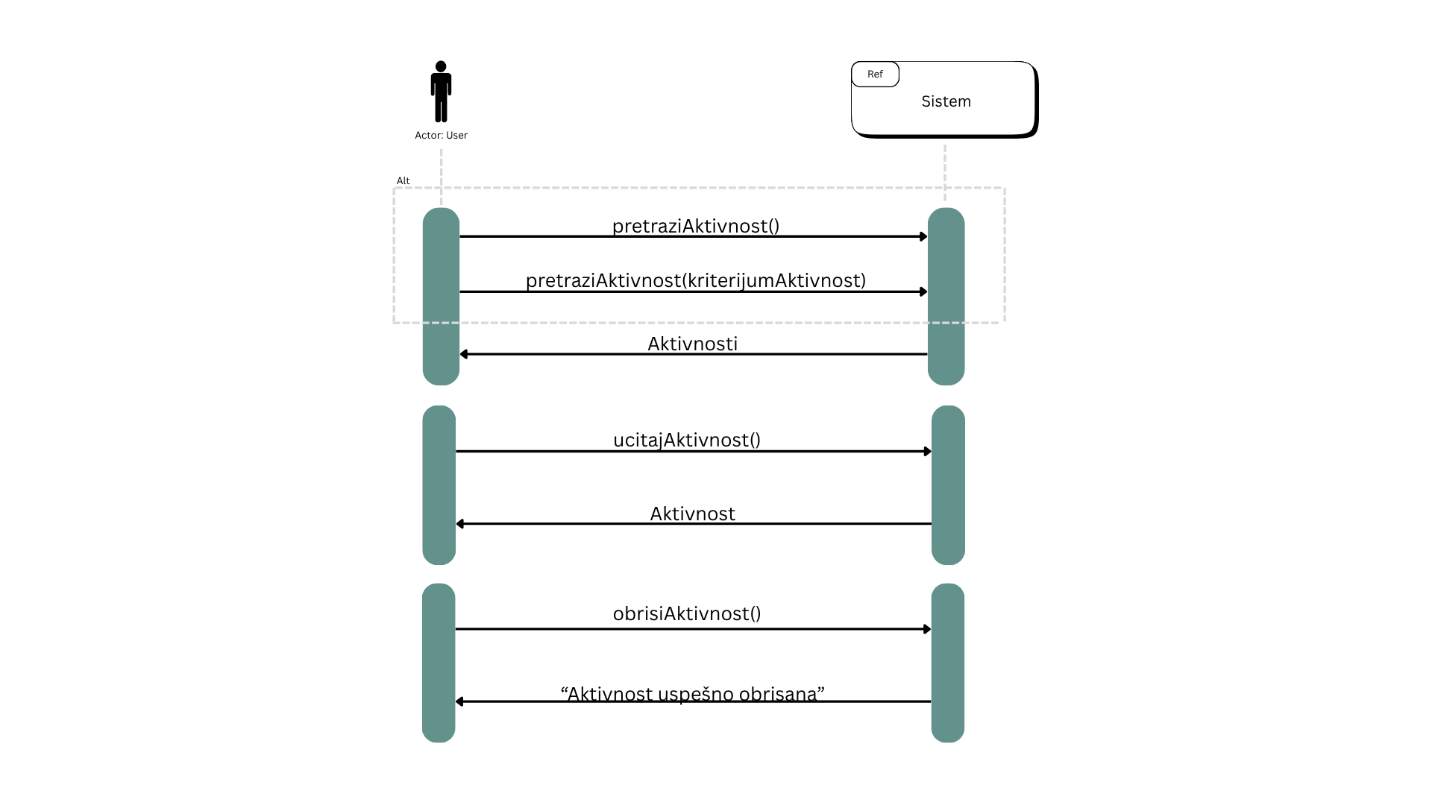
Aktori SK:  
Korisnik (User)

Učesnici SK:  
Korisnik i sistem

Preduslov:  
Sistem je uključen, korisnik je ulogovan i ima pravo pristupa aktivnostima. Sistem prikazuje formu za rad sa aktivnostima.

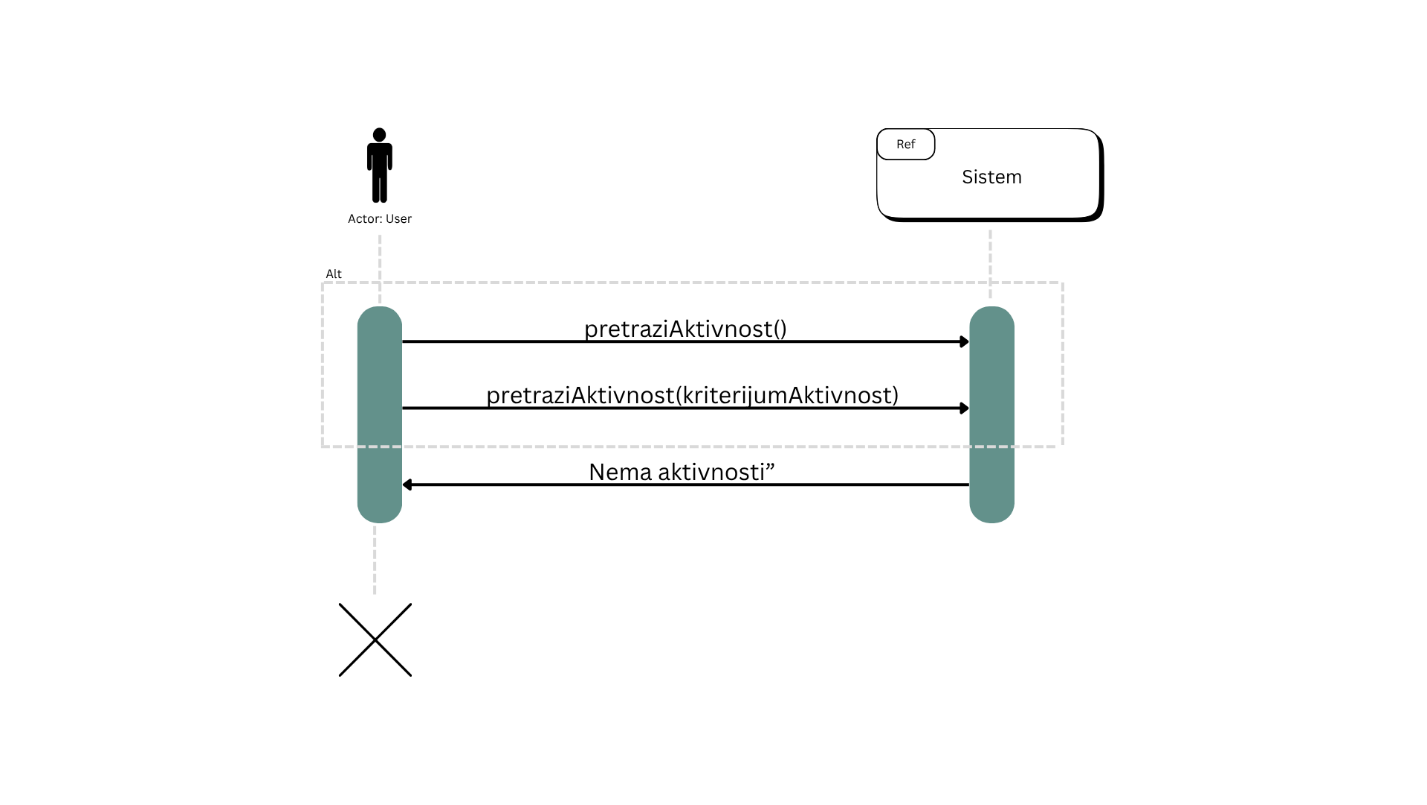
Osnovni scenario SK:

1. Korisnik bira kriterijum za pretragu aktivnosti (npr. naziv, status, datum). (APUSO)
2. Korisnik poziva sistem da pronađe aktivnosti prema unetom kriterijumu. (APSO)
3. Sistem traži aktivnosti po zadatoj vrednosti. (SO)
4. Sistem prikazuje korisniku listu pronađenih aktivnosti. (IA)
5. Korisnik bira jednu aktivnost sa liste. (APUSO)
6. Korisnik poziva sistem da učita podatke o odabranoj aktivnosti. (APSO)
7. Sistem učitava podatke o aktivnosti. (SO)
8. Sistem prikazuje korisniku podatke o aktivnosti. (IA)
9. Korisnik poziva sistem da obriše aktivnost. (APSO)
10. Sistem briše aktivnost zajedno sa povezanim komentarima i sugestijama (kaskadno). (SO)
11. Sistem prikazuje korisniku poruku: “Aktivnost uspešno obrisana”. (IA)

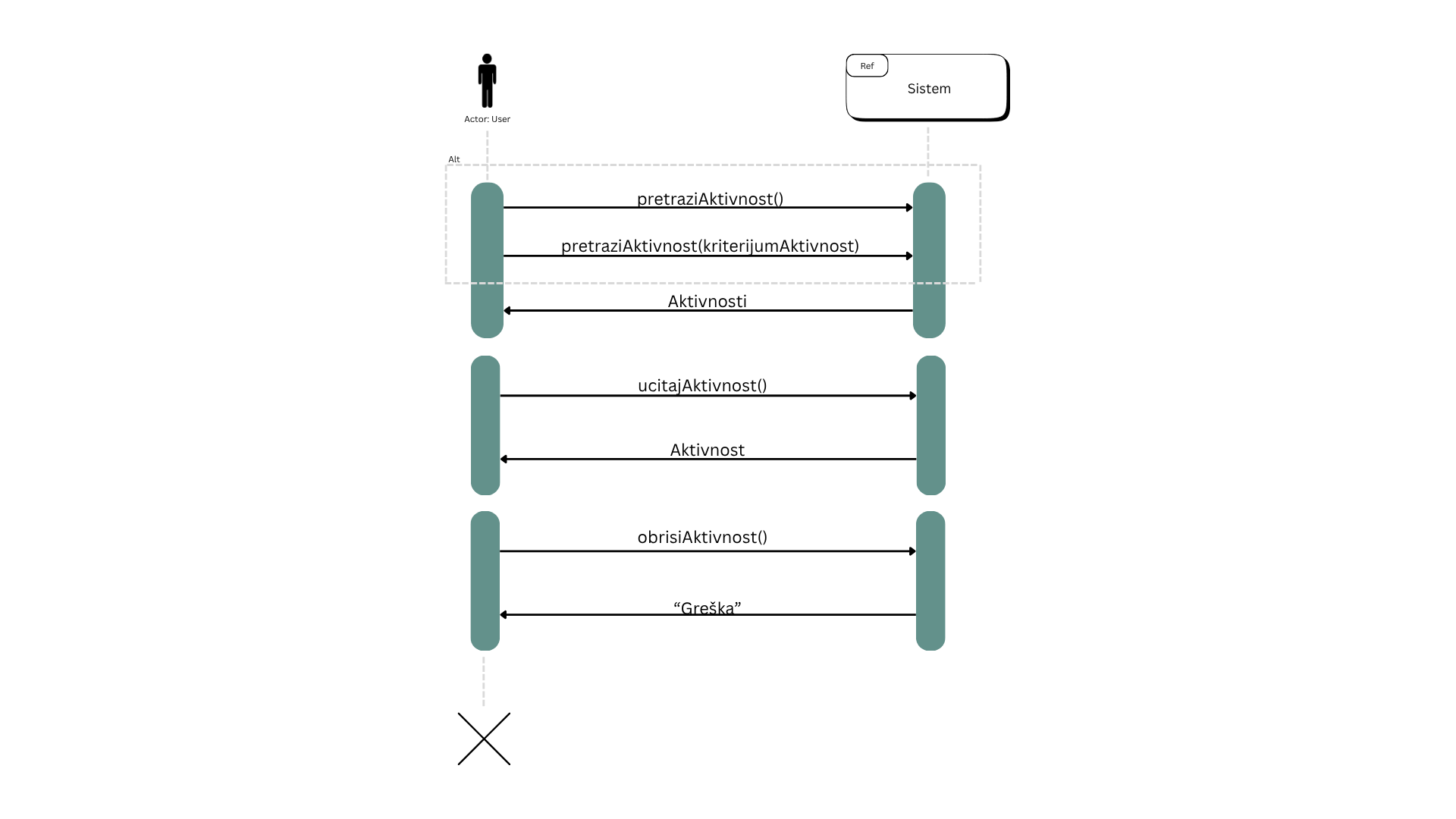


Alternativni scenariji:

4.1 Ukoliko sistem ne može da pronađe aktivnosti, prikazuje poruku: “Nema aktivnosti!”. Prekida se izvršenje scenarija. (IA)



11.1 Ukoliko sistem ne može da obriše aktivnost, prikazuje poruku: “Greška”. Prekida se izvršenje scenarija. (IA)



# Implementacija

## Opis modela

Na osnovu relacionog modela iz prethodnog poglavlja mogu se identifikovati potrebni atributi za svaki entitet, dok PMOV model omogućava uvid u odnose i njihove kardinalnosti.  
Da bismo ove veze prikazali u Laravel okruženju, koristimo Eloquent metode poput hasMany i belongsTo. Kada relacija između dva modela predstavlja odnos jedan prema više (1:0..\*), model koji pripada "više" strani sadržaće belongsTo metodu, dok će model sa "jedan" strane koristiti hasMany.  
Ovakav pristup omogućava Laravelu da automatski prepozna povezanost i strukturu odnosa između entiteta.

U daljem tekstu dati su primeri korišćenja Eloquent relacija, kao i njihova implementacija u samim modelima.

    public function drustvo(): *BelongsTo*

    {

        return *$this*->belongsTo(*Drustvo*::class);

    }

    public function komentars():*HasMany*{

        return *$this*->hasMany(*Komentar*::class);

    }

Kako svaka aktivnost pripada jednom društvu, potrebno je koristiti belongsTo relaciju, dok u slučaju predstavljanja odnosa klase aktivnost i klase komentar primenjuje se hasMany relacija kojom se označava da se za jednu pojavu instance aktivnosti može vezati više konkretnih instanci komentara.

Pomenute relacije definisane su u svim klasama oslikavajući kardinalnosti konceptualnog modela, odnosno PMOV-a.

## Opis migracija

Migracije u Laravelu predstavljaju način za verzionisano upravljanje strukturom baze podataka. Svaka migracija opisuje određenu promenu – poput kreiranja, izmene ili brisanja tabela i njihovih kolona. One omogućavaju da se promene u strukturi baze prate kroz vreme, a takođe olakšavaju timski rad jer omogućavaju lako deljenje i sinhronizaciju baze između razvojnih, testnih i produkcionih okruženja.

Kreiranje migracija realizuje se na jedan od dva načina:

1. Kreiranje migracije istovremeno sa modelom, dodavanjem flega -m, pri čemu je artisan komanda sledeća: php artisan make:model NazivModela -m;
2. Samostalno kreiranje migracije pozivom sledeće artisan komande: php artisan make:migration add\_deleted\_at\_to\_notifikacijas.

## Opis kontrolera

U Laravelu, kontroleri predstavljaju centralnu komponentu unutar MVC (Model-View-Controller) arhitekture. Njihova osnovna uloga je da obrađuju korisničke zahteve, pozivaju odgovarajuće metode iz modela i prosleđuju rezultate ka prikazima (views). Na taj način, kontroleri služe kao posrednici između sloja podataka (model) i korisničkog interfejsa (view), čime doprinose boljoj organizaciji i čitljivosti koda, kao i odvajanju poslovne logike od prezentacije.

U slučaju naše aplikacije, korišćeni su api kontroleri, odnosno kontroleri koji se bave isključivo obradom poslovne logike aplikacije, bez logike prikazivanja sadržaja. API kontroleri u Laravelu idealni su za razvoj backend dela aplikacije jer ne učestvuju u renderovanju prikaza (view-a). Umesto toga, njihov zadatak je da obrađuju zahteve i vraćaju odgovore u JSON formatu, što ih čini posebno pogodnim za rad sa frontend framework-ima poput React-a, kao i za testiranje putem alata kao što je Postman.

Za kreiranje api kontrolera koristi se naredba php artisan make:controller api/NazivModelaController –api. Pri izvršenju ove komande u okviru App\Http\Controllers\api kreira se NazivModelaController.php.

Metode api kontrolera orijentisane su prema CRUD operacijama nad modelima aplikacije. Kako bi se smanjila otperećenost kontrolera, obzirom na važnost njihove uloge u sistemu, korišćene su sledeće klase:

1. Resource – klase koje predtsvljaju sloj između JSON-a i modela, odnosno koje formatiraju odgovor koji server šalje prema klijentu;
2. Policy – klase koje obrađuju logiku autorizacije korisnika;
3. Form Request – klase koje pozivaju pojedinačne metode autorizacije, vrše validaciju i definišu specifične odgovore u slučaju narušavanja postavljenih pravila za validaciju.

Opis implementiranih:

1. Modela
2. Migracija
3. Kontrolera
4. Lista api ruta

Potrebno je objasniti specifične delove koda (npr. nije neophodno objašnjavati sve migracije, već jednu za kreiranje, jednu za dodavanje kolone, jednu za izmenu itd.).

# REST API

Kreirati API dokumentaciju.

Koristiti Slate za generisanje specifikacije <https://github.com/slatedocs/slate>

ili

Napraviti tabelu za svaku rutu kao u primeru ispod.

|  |  |
| --- | --- |
| Opis funkcije | Kreiranje novog članka |
| HTTP metoda | POST |
| URL | /api/write |
| URL parametri | (nema) |
| HTTP body parametri | {      "post\_id": "13",      "title": "Pasulj",      "desc": "Tekst članka",      "img": "16684321218food2.jpg",      "date": "2022-11-14 14:30:11",      "uid": 12      "cat": "food"  } |
| Format HTTP body parametara | JSON |
| Izlazni parametri | {      "msg": "Kreiran članak!"  } |
| Format izlaznih parametara | application/json |
| Opis funkcije | Prikaz svih članaka kategorije ’food’ |
| HTTP metoda | GET |
| URL | /api/products |
| URL parametri | ?cat=food |
| HTTP body parametri | (nema) |
| Format HTTP body parametara | (nema) |
| Izlazni parametri | {  "status": "success",      "result": 3,      "products": [     {      "post\_id": "13",      "title": "Pasulj",      "desc": "Tekst članka",      "img": "16684321218food2.jpg",      "date": "2022-11-14 14:30:11",      "uid": 12      "cat": "food"  },  {      "post\_id": "14",      "title": "Paprike",      "desc": "Tekst članka",      "img": "16692103921food3.jpg",      "date": "2022-11-14 14:35:31 ",      "uid": 12      "cat": "food"  },  {      "post\_id": "pro06",      "title": "Jabuke",      "desc": "Tekst članka",      "img": "16683125123food2.jpg",      "date": "2022-11-14 13:45:13",      "uid": 13      "cat": "food"  }  ]  } |
| Format izlaznih parametara | application/json |

# Korisničko uputstvo

Opis i screenshot-ovi aplikacije testirane u okviru Postman-a- (ili nekog drugog servisa za testiranje API putanja)