

## DOKUMENTACIJA SISTEMA PREPORUKE za aplikaciju SeriLovers

Predmet: Razvoj softvera II  
Fakultet informacijskih tehnologija

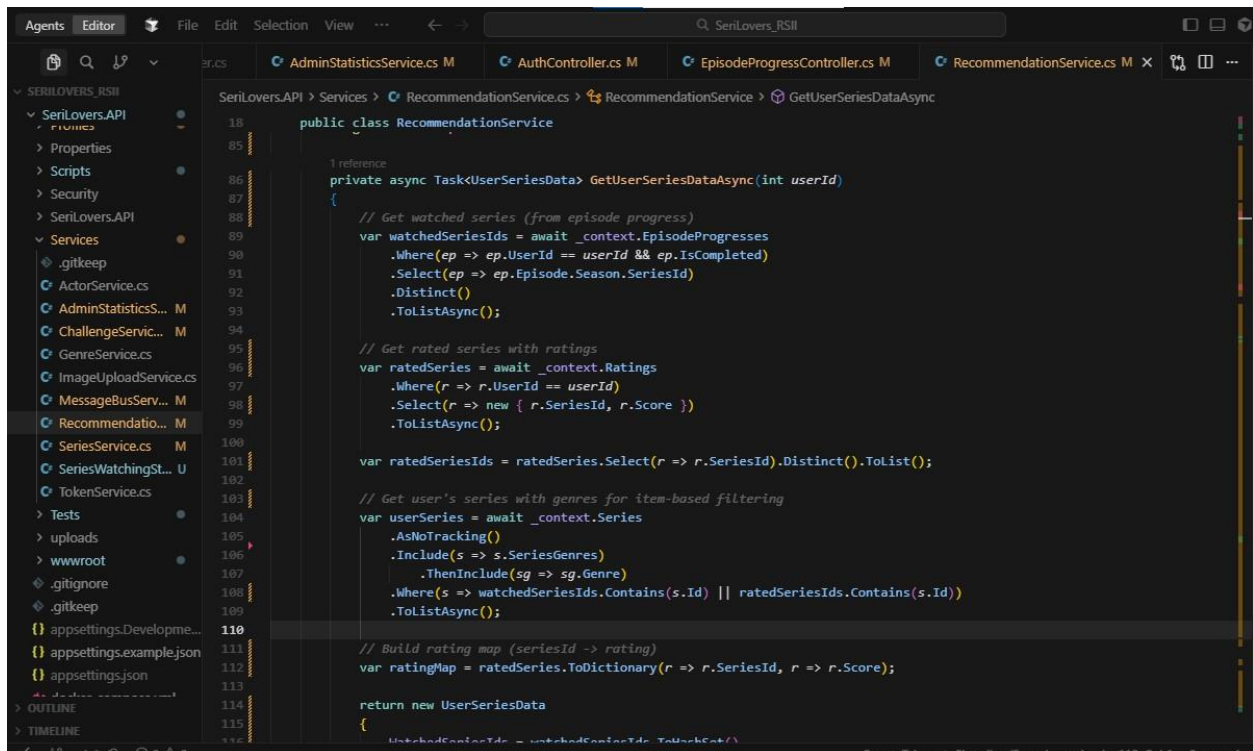
Studentica: Elmedina Marić  
Akademska godina: 2025/2026

### 1. Uvod

Ovaj dokument predstavlja detaljan opis implementacije sistema preporuke u aplikaciji **SeriLovers**. Glavni cilj sistema preporuke je unaprijediti korisničko iskustvo kroz personalizovane prijedloge TV serija, prilagođene interesima i prethodnom ponašanju korisnika.

Sistem preporuke omogućava korisnicima da brže i lakše pronađu relevantan sadržaj, bez potrebe za dugotrajnim i ručnim pretraživanjem baze podataka. Time se povećava zadovoljstvo korisnika i ukupna vrijednost aplikacije.

Implementirani sistem preporuke je **hibridni recommender sistem**, koji kombinuje sadržajno filtriranje (*item-based filtering*) i kolaborativno filtriranje zasnovano na korisnicima (*user-based collaborative filtering*). Kombinovanjem ova dva pristupa ostvaruje se veća preciznost, relevantnost i pouzdanost preporuka u odnosu na korištenje samo jednog modela.



```
18 public class RecommendationService
19 {
20     // Get watched series (from episode progress)
21     private async Task<UserSeriesData> GetUserSeriesDataAsync(int userId)
22     {
23         var watchedSeriesIds = await _context.EpisodeProgresses
24             .Where(ep => ep.UserId == userId && ep.IsCompleted)
25             .Select(ep => ep.Episode.Season.SeriesId)
26             .Distinct()
27             .ToListAsync();
28
29         // Get rated series with ratings
30         var ratedSeries = await _context.Ratings
31             .Where(r => r.UserId == userId)
32             .Select(r => new { r.SeriesId, r.Score })
33             .ToListAsync();
34
35         var ratedSeriesIds = ratedSeries.Select(r => r.SeriesId).Distinct().ToListAsync();
36
37         // Get user's series with genres for item-based filtering
38         var userSeries = await _context.Series
39             .AsNoTracking()
40             .Include(s => s.SeriesGenres)
41             .ThenInclude(sg => sg.Genre)
42             .Where(s => watchedSeriesIds.Contains(s.Id) || ratedSeriesIds.Contains(s.Id))
43             .ToListAsync();
44
45         // Build rating map (seriesId -> rating)
46         var ratingMap = ratedSeries.ToDictionary(r => r.SeriesId, r => r.Score);
47
48         return new UserSeriesData
49         {
50             WatchedSeriesIds = watchedSeriesIds,
51             RatedSeriesIds = ratedSeriesIds,
52             UserSeries = userSeries,
53             RatingMap = ratingMap
54         };
55     }
56 }
```

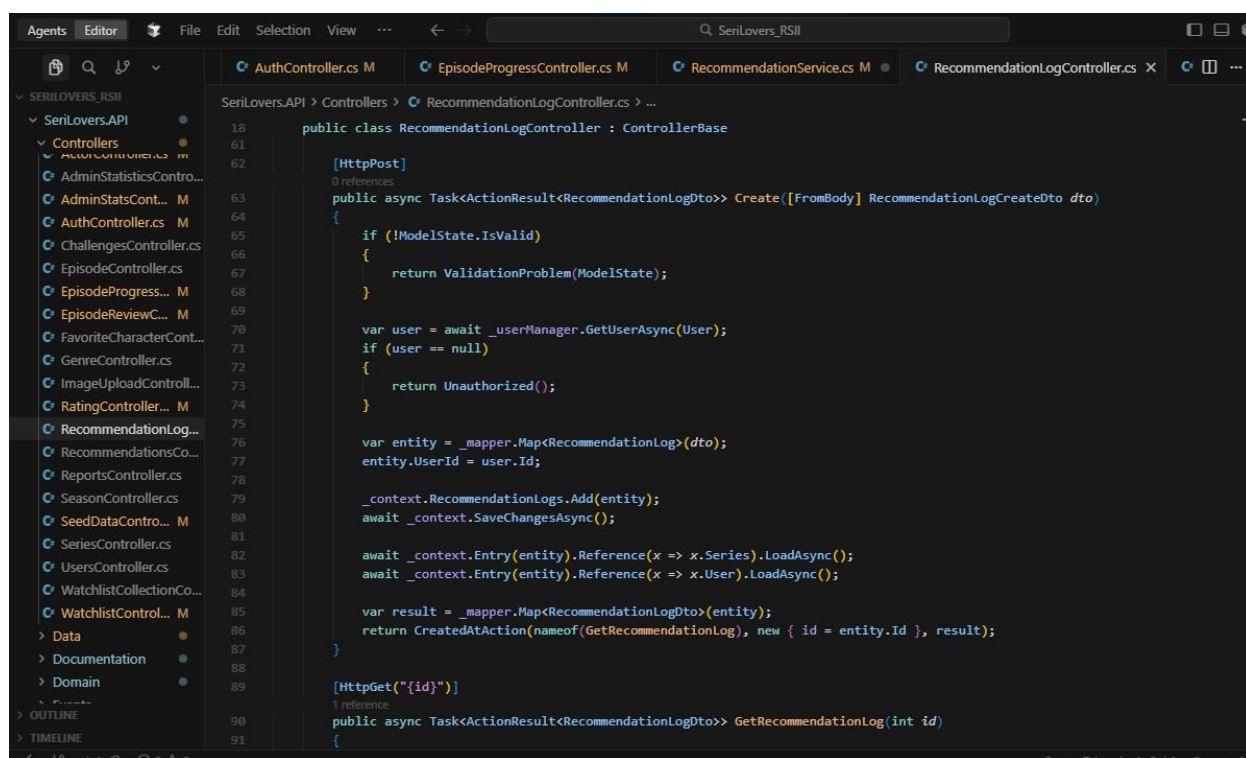
Slika 1 -SeriLovers.API/Services/RecommendationService.cs

## 2. Arhitektura sistema preporuke

Sistem preporuke je implementiran u backend dijelu aplikacije koristeći **ASP.NET Core Web API** arhitekturu. Centralni dio sistema predstavlja servis **RecommendationService**, koji sadrži kompletnu poslovnu logiku za generisanje preporuka.

Kontroler **RecommendationsController** služi kao posrednik između frontend aplikacije i sistema preporuke, izlažući REST API rute koje frontend koristi za dohvat preporuka. Ovakva arhitektura omogućava jasnu separaciju odgovornosti, lakše održavanje sistema i jednostavno proširenje funkcionalnosti u budućnosti.

Frontend dio aplikacije, razvijen u **Flutter** tehnologiji, komunicira sa backendom putem HTTP zahtjeva i prikazuje rezultate sistema preporuke krajnjim korisnicima.



```
Agents Editor File Edit Selection View ... SerLovers_RSII
SeriLovers.API > Controllers > RecommendationLogController.cs > ...
18 public class RecommendationLogController : ControllerBase
61
62 [HttpPost]
0 references
63 public async Task<ActionResult<RecommendationLogDto>> Create([FromBody] RecommendationLogCreatedDto dto)
64 {
65     if (!ModelState.IsValid)
66     {
67         return ValidationProblem(ModelState);
68     }
69
70     var user = await _userManager.GetUserAsync(User);
71     if (user == null)
72     {
73         return Unauthorized();
74     }
75
76     var entity = _mapper.Map<RecommendationLog>(dto);
77     entity.UserId = user.Id;
78
79     _context.RecommendationLogs.Add(entity);
80     await _context.SaveChangesAsync();
81
82     await _context.Entry(entity).Reference(x => x.Series).LoadAsync();
83     await _context.Entry(entity).Reference(x => x.User).LoadAsync();
84
85     var result = _mapper.Map<RecommendationLogDto>(entity);
86     return CreatedAtAction(nameof(GetRecommendationLog), new { id = entity.Id }, result);
87 }
88
89 [HttpGet("{id}")]
1 reference
90 public async Task<ActionResult<RecommendationLogDto>> GetRecommendationLog(int id)
91 {
```

Slika 2 - SeriLovers.API/Controllers/RecommendationsController.cs

### 3. Korišteni podaci za generisanje preporuka

Sistem preporuke koristi više izvora podataka kako bi formirao potpun korisnički profil. Analiziraju se serije koje je korisnik gledao, serije koje je ocijenio, kao i ocjene i ponašanje drugih korisnika unutar sistema.

Posebna pažnja posvećena je žanrovima serija, jer oni predstavljaju važan indikator korisničkih preferencija. Također se uzima u obzir popularnost serija, broj ocjena i prosječna ocjena, što omogućava sistemu da donese kvalitetnije odluke prilikom preporučivanja sadržaja.

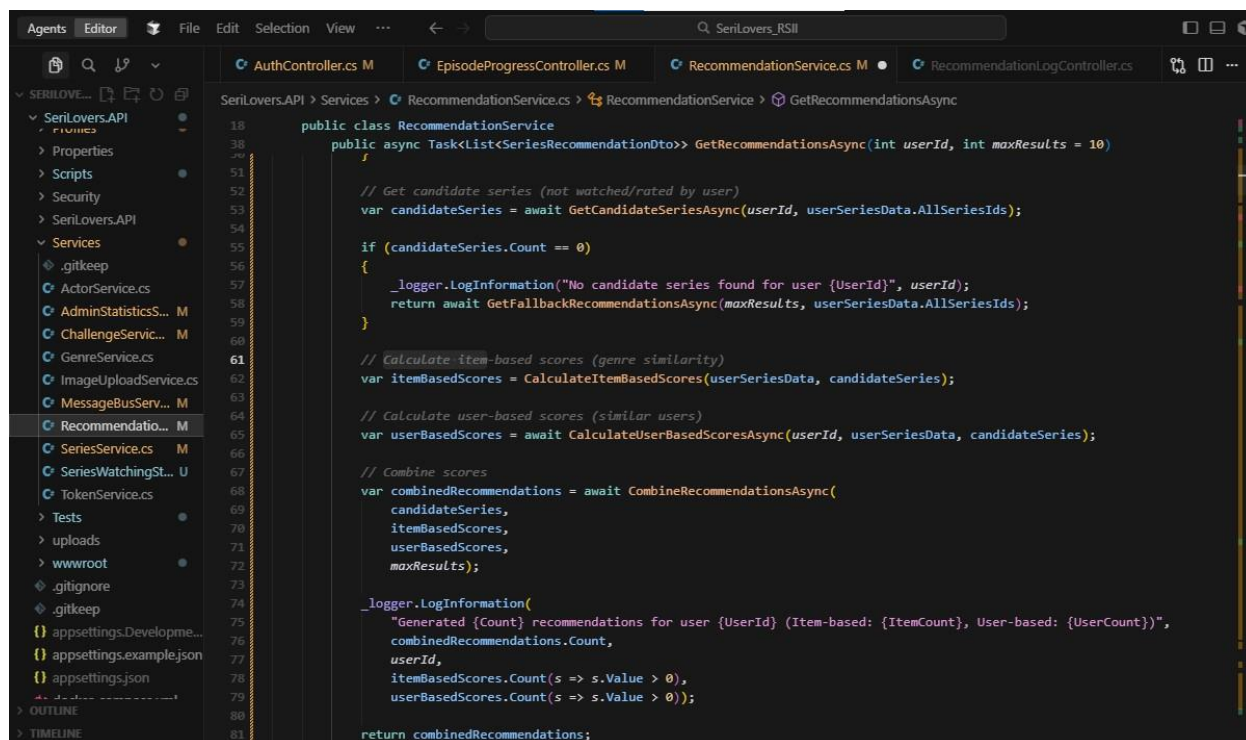
Kombinovanjem eksplicitnih podataka (ocjene) i implicitnih podataka (gledanje sadržaja), sistem dobija realističniji uvid u interesovanja korisnika.

### 4. Item-based filtering (sadržajno filtriranje)

Item-based filtering se zasniva na pretpostavci da će se korisniku svidjeti serije koje su slične onima koje je već gledao ili visoko ocijenio. U ovom sistemu sličnost između serija se prvenstveno određuje na osnovu žanrova.

Za svaku seriju koju je korisnik gledao ili ocijenio, izdvajaju se pripadajući žanrovi, koji se zatim ponderišu prema korisničkoj ocjeni. Više ocijenjene serije imaju veći uticaj na formiranje korisničkog profila.

Sličnost između korisnikovog profila i kandidatskih serija računa se pomoću **ponderisane Jaccard sličnosti**, čime se dobija numerički skor koji označava stepen podudaranja sadržaja. Ovaj pristup omogućava da preporuke budu tematski usklađene sa interesima korisnika.



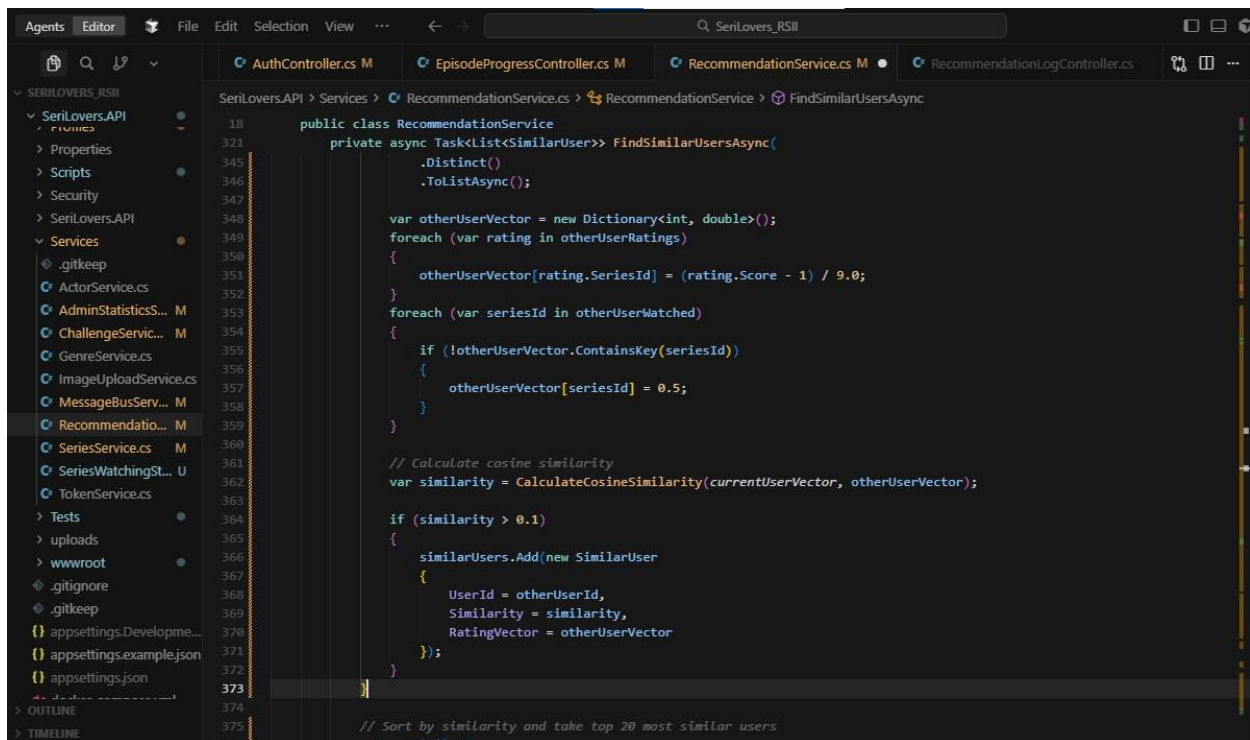
Slika 3 -CalculateItemBasedScores

## 5. User-based collaborative filtering

User-based collaborative filtering se oslanja na ideju da korisnici sa sličnim ukusom u prošlosti imaju tendenciju da se slažu i u budućnosti. U ovom sistemu sličnost između korisnika se računa na osnovu njihovih ocjena i gledanog sadržaja.

Za svakog korisnika se formira vektor ocjena, pri čemu se eksplicitne ocjene normalizuju, a gledane serije tretiraju kao implicitne pozitivne interakcije. Na osnovu tih vektora računa se sličnost između korisnika koristeći **kosinusnu sličnost**.

Korisnici sa najvećim stepenom sličnosti smatraju se relevantnim, a njihov ukus se koristi za preporuku novih serija. Na ovaj način sistem koristi kolektivno znanje zajednice korisnika.



Slika 4 - CalculateCosineSimilarity

## 6. Kombinovanje rezultata (hibridni pristup)

Kako bi se postigla optimalna ravnoteža između sadržajne sličnosti i ponašanja drugih korisnika, rezultati item-based i user-based filtriranja se kombinuju u jedinstveni skor.

U implementaciji je primijenjen ponderisani pristup, gdje item-based filtering ima težinu od **60%**, dok user-based filtering učestvuje sa **40%** u konačnom rezultatu. Ovakva raspodjela omogućava da se korisničke lične preferencije blago favorizuju u odnosu na kolektivne preporuke.

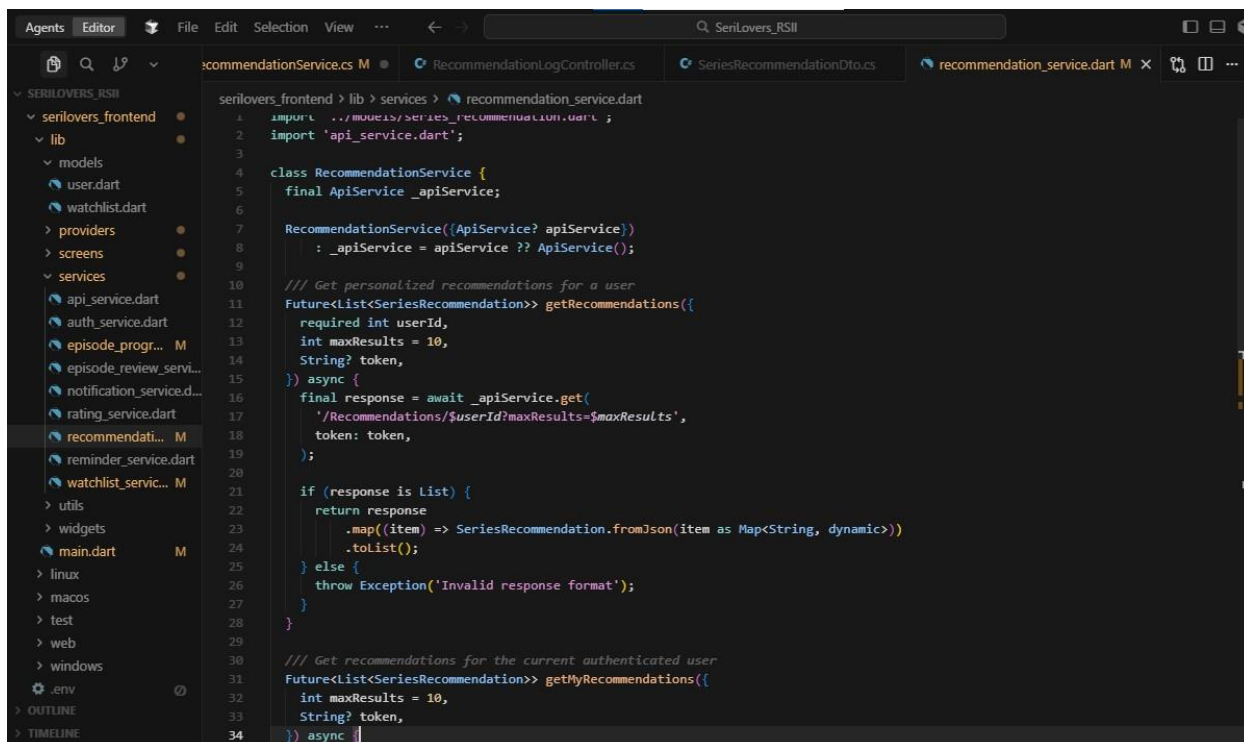
Finalni skor se koristi za rangiranje serija, nakon čega se korisniku prikazuju najrelevantnije preporuke.

## 7. Prikaz preporuka u aplikaciji

Preporuke se prikazuju u frontend dijelu aplikacije putem posebnog servisa RecommendationService. Frontend šalje zahtjev prema API ruti /api/Recommendations/me, nakon čega backend vraća listu preporučenih serija.

Svaka preporuka sadrži dodatne informacije, uključujući naziv serije, listu žanrova, prosječnu ocjenu i razlog preporuke. Prikaz razloga preporuke povećava transparentnost sistema i pomaže korisniku da razumije zašto je određena serija predložena.

Na ovaj način sistem preporuke ne djeluje kao „crna kutija“, već pruža objašnjenja za svoje odluke.



Slika 5 - lib/services/recommendation\_service.dart

## 8. Zaključak

Implementirani sistem preporuke u aplikaciji SeriLovers predstavlja robustno i fleksibilno rješenje za personalizaciju sadržaja. Korištenjem hibridnog pristupa postiže se veća tačnost preporuka i bolje korisničko iskustvo.

Sistem je dizajniran tako da se može lako proširivati dodavanjem novih faktora, kao što su glumci, režiseri ili vremenski trendovi, što ga čini pogodnim za dalji razvoj i unapređenje aplikacije.