

SApplauz – sistem preporuka

SApplauz sistem preporuka je implementiran kao content-based recommendation system koji koristi žanrove predstava i korisničke preferencije za generisanje personalizovanih preporuka. Sistem je dizajniran da rješava problem "cold start" scenarija i pruža relevantne preporuke korisnicima na osnovu njihove historije kupovine karata i recenzija.

Sistem preporuka se sastoji od sljedećih komponenti:

1. Backend Service ("RecommendationService.cs")
- Lokacija: "backend/SApplauz.Application/Services/RecommendationService.cs"
2. API Controller ("RecommendationsController.cs")
- Lokacija: "backend/SApplauz.API/Controllers/RecommendationsController.cs"
3. Frontend Screen ("recommendations_screen.dart")
- Lokacija: "mobile/lib/screens/recommendations/recommendations_screen.dart"
4. Cache Service ("recommendations_cache_service.dart")
- Lokacija: "mobile/lib/services/recommendations_cache_service.dart"
5. Database Entity ("RecommendationProfile.cs")
- Lokacija: "backend/SApplauz.Domain/Entities/RecommendationProfile.cs"

Kao što sam prethodno navela, sistem koristi content-based filtering pristup koji se bazira na:

1. Korisničkim preferencijama - Žanrovi predstava koje je korisnik gledao/kupljene karte
2. Žanrovima predstava - Svaka predstava ima jedan ili više žanrova
3. Ocjenama predstava - Prosječna ocjena predstave utiče na score
4. Popularnosti - Broj recenzija i ocjena

U nastavku ću preciznije objasniti po koracima.

- Korak 1: Učitavanje korisničkih preferencija

```
private async Task<Dictionary<int, double>> GetUserPreferencesAsync(string userId)
```

```
{
```

```
    var profile = await _dbContext.RecommendationProfiles
```

```
        .FirstOrDefaultAsync(p => p.UserId == userId);
```

```
    if (profile == null || string.IsNullOrEmpty(profile.PreferredGenresJson))
```

```

{
    return new Dictionary<int, double>();
}

var preferences = JsonSerializer.Deserialize<Dictionary<int, double>>(profile.PreferredGenresJson);

return preferences ?? new Dictionary<int, double>();
}

```

Ova funkcija učitava RecommendationProfile za korisnika. JSON string se deserializuje u Dictionary<int, double>, gdje je ključ GenreId, a vrijednost Weight (težina preferencije od 0 do 10). Ako korisnik nema definisane preferencije, vraća se prazan dictionary (cold start slučaj).

- Korak 2: Cold Start Handling

```

if (userPreferences.Count == 0)
{
    var popularShows = await GetPopularShowsAsync(count);

    return popularShows;
}

```

Ako korisnik nema definisane preferencije (cold start), sistem vraća popularne predstave. Popularnost se određuje sortiranjem prvo po broju recenzija (opadajuće), a zatim po prosječnoj ocjeni (opadajuće).

- Korak 3: Filtriranje predstava

```

var userShowIds = await GetUserShowIdsAsync(userId);

var scoredShows = allShows
    .Where(s => !userShowIds.Contains(s.Id))
    .Select(show => {
        var showGenres = new List<int> { show.GenreId };

        var score = CalculateScore(userPreferences, showGenres, show);

        return new { Show = show, Score = score };
    });

```

```

    })
    .Where(x => x.Score > 0)
    .OrderByDescending(x => x.Score)
    .Take(count)
    .ToList();

```

Sistem prvo isključuje predstave koje korisnik već ima (kupljene karte ili ostavljene recenzije). Za svaku preostalu predstavu izračunava se score, zadržavaju se samo one sa pozitivnim score-om, a zatim se sortiraju po score-u (opadajuće) i uzima se top “count” predstava.

- Korak 4: Izračun Score-a

```

private double CalculateScore(Dictionary<int, double> userPreferences, List<int> showGenres,
Show show)
{
    if (showGenres.Count == 0)
    {
        return 0;
    }

    double totalScore = 0;
    int matchingGenres = 0;
    foreach (var genreId in showGenres)
    {
        if (userPreferences.ContainsKey(genreId))
        {
            totalScore += userPreferences[genreId];
            matchingGenres++;
        }
    }
}

```

```

if (matchingGenres == 0)
{
    return 0;
}

var baseScore = totalScore / (userPreferences.Values.Sum() + 1);

var genreMatchBonus = (double)matchingGenres / showGenres.Count;

var avgRating = show.Reviews.Any()
    ? show.Reviews.Average(r => r.Rating) / 5.0
    : 0.5;

var ratingBonus = avgRating * 0.2;

var finalScore = (baseScore * 0.5) + (genreMatchBonus * 0.3) + (ratingBonus * 0.2);

return Math.Min(1.0, finalScore);
}

```

Base Score (50%) predstavlja normalizovanu sumu korisničkih preferencija za podudarne žanrove. Genre Match Bonus (30%) zavisi od udjela žanrova predstave koji se poklapaju s korisničkim preferencijama. Rating Bonus (20%) je prosječna ocjena predstave, normalizovana na raspon 0–1 (gdje je 5.0 = 1.0). Konačni rezultat je kombinovani score u rasponu od 0.0 do 1.0.

Formula:

$$finalScore = (baseScore \times 0.5) + (genreMatchBonus \times 0.3) + (ratingBonus \times 0.2)$$

- Korak 5: Generisanje Razloga

```

private string GenerateReason(Dictionary<int, double> userPreferences, List<string>
showGenres, double score)
{
    if (showGenres.Any())
    {
        if (score > 0.7)
        {

```

```

        return $"Visoko preporučujemo! Žanrovi: {string.Join(", ", showGenres)}";
    }

    if (score > 0.4)
    {
        return $"Preporučujemo na osnovu žanrova: {string.Join(", ", showGenres)}";
    }
}

if (score > 0.3)
{
    return "Možda će vam se svidjeti";
}

return "Preporučujemo";
}

```

Sistem generiše ljudski-čitljiv razlog za preporuku na osnovu vrijednosti score-a. Ako je score veći od 0.7, prikazuje se poruka „Visoko preporučujemo!“. Za score veći od 0.4 koristi se poruka „Preporučujemo na osnovu žanrova“, a za score veći od 0.3 „Možda će vam se svidjeti“. U svim ostalim slučajevima prikazuje se poruka „Preporučujemo“.

- Ažuriranje preferencija

Preferencije se ažuriraju automatski kada:

1. Korisnik kupi kartu - Dodaje se žanr predstave u preferencije sa težinom 1.0
2. Korisnik ostavi recenziju - Dodaje se žanr predstave sa težinom = “rating / 5.0” (npr. rating 5 = težina 1.0, rating 3 = težina 0.6)

U nastavku je prikazana implementacija:

```

public async Task UpdateUserPreferencesAsync(string userId, int showId, int? rating = null)
{
    var show = await _dbContext.Shows

```

```
.Include(s => s.Genre)

.FirstOrDefaultAsync(s => s.Id == showId);

if (show == null)
{
    return;
}

var profile = await _dbContext.RecommendationProfiles
    .FirstOrDefaultAsync(p => p.UserId == userId);

if (profile == null)
{
    profile = new RecommendationProfile
    {
        UserId = userId,
        LastUpdated = DateTime.UtcNow
    };
    _dbContext.RecommendationProfiles.Add(profile);
}

var preferences = await GetUserPreferencesAsync(userId);
var genreId = show.GenreId;
var weight = rating.HasValue ? rating.Value / 5.0 : 1.0;

if (preferences.ContainsKey(genreId))
{
```

```

        preferences[genreId] += weight;
    }
    else
    {
        preferences[genreId] = weight;
    }

    var maxPreference = preferences.Values.Max();
    if (maxPreference > 10)
    {
        var normalizationFactor = 10.0 / maxPreference;
        foreach (var key in preferences.Keys.ToList())
        {
            preferences[key] *= normalizationFactor;
        }
    }

    var preferencesJson = JsonSerializer.Serialize(preferences);
    profile.PreferredGenresJson = preferencesJson;
    profile.LastUpdated = DateTime.UtcNow;

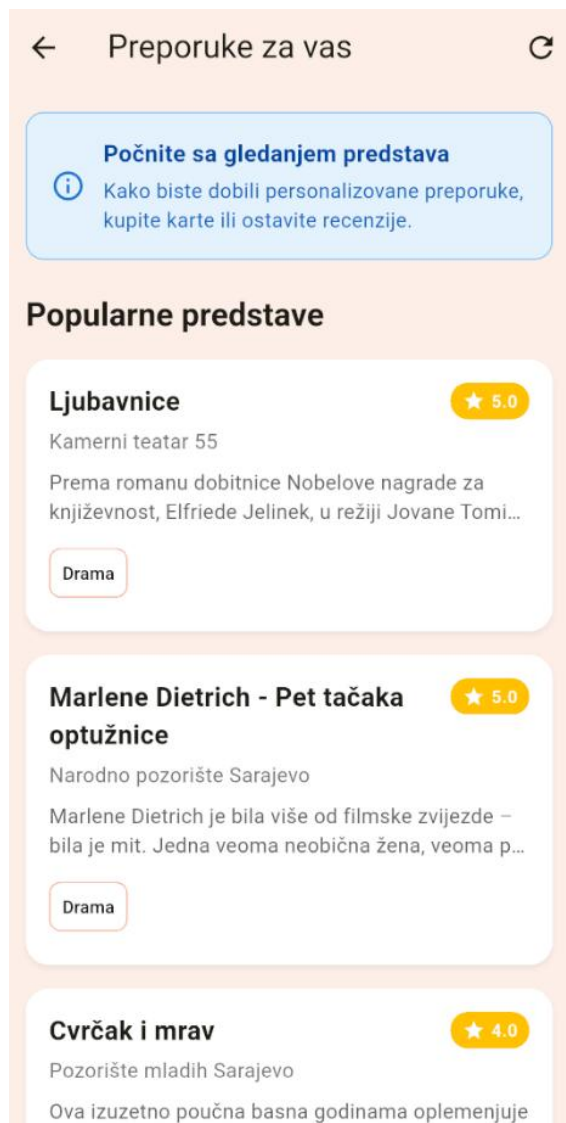
    await _dbContext.SaveChangesAsync();
}

```

Ako korisnik već ima preferenciju za određeni žanr, njegova težina se povećava. Ako preferencija ne postoji, dodaje se nova. Na kraju se radi normalizacija, tako da se sve težine proporcionalno skaliraju ako neka vrijednost pređe 10.

Sapplauz sistem preporuka omogućava korisnicima da brzo dobiju relevantne i personalizovane preporuke predstava. Sistem koristi žanrove za prilagođavanje sadržaja interesovanjima korisnika, dok se za nove korisnike bez istorije (cold start) prikazuju popularne predstave. Preporuke se keširaju radi bržeg učitavanja, a korisničke preferencije se automatski ažuriraju na osnovu kupovine karata i recenzija. Dodatno, sistem generiše jasne i razumljive razloge za svaku preporuku. Generalno gledano, sistem je efikasan, skalabilan i praktičan, te pruža kvalitetne preporuke koje unapređuju korisničko iskustvo aplikacije.

Cold start:



Primjer preporuke korisniku na osnovu žanra::

