

Opis implementacije recommender sistema

Tip sistema preporuke

Implementiran je **Content-Based Filtering** sistem preporuke koji koristi karakteristike ponuda za preporučivanje korisnika kojima bi ponuda mogla biti zanimljiva.

Glavna logika sistema

Sistem radi po principu pronalaska korisnika čiji su interesi usklađeni sa karakteristikama nove ponude. Algoritam koristi sledeće korake:

1. **Ekstraktovanje karakteristika ponude:** Za svaku ponudu se izdvajaju ključni identifikatori:
 - boardTypeld - tip pansiona
 - hotelld - identifikator hotela
 - cityld - identifikator grada
 - countryld - identifikator države
2. **Pronalaženje relevantnih korisnika:** Pretražuju se korisnici čiji tagovi odgovaraju bilo kojoj karakteristici ponude
3. **Kreiranje korisničkih profila:** Na osnovu historije tagova (posljednja godina) kreira se profil interesovanja za svakog korisnika
4. **Računanje sličnosti:** Za svakog korisnika računa se broj poklapanja sa karakteristikama ponude
5. **Rangiranje i filtriranje:** Korisnici se sortiraju po broju poklapanja i uzima se top 100

Ključne karakteristike implementacije

- **Vremenska komponenta:** Koriste se samo tagovi iz poslednje godine
- **Aktivnost korisnika:** Uključuju se samo aktivni korisnici
- **Hijerarhijska struktura:** Sistem prepoznaje hijerarhiju lokacije (Hotel → Grad → Zemlja)

- **Scoring algoritam:** Korisnici se rangiraju na osnovu broja poklapanja tagova
- **Ograničavanje rezultata:** Vraća se maksimalno 100 najrelevantnijih korisnika

Source kod

- Putanja do servisa: ***eTouristAgencyAPI /eTouristAgencyAPI.Services/ OfferContentBasedService.cs***
- Putanja do interfejsa: ***eTouristAgencyAPI/eTouristAgencyAPI.Services/Interfaces/IOfferContentBased Service.cs***
- Screenshot servisa

```
using eTouristAgencyAPI.Services.Database;
using eTouristAgencyAPI.Services.Database.Models;
using eTouristAgencyAPI.Services.Interfaces;
using Microsoft.EntityFrameworkCore;

namespace eTouristAgencyAPI.Services
{
    2 references | Hamza Bikić, 18 hours ago | 1 author, 2 changes
    public class OfferContentBasedService : IOfferContentBasedService
    {
        private readonly eTouristAgencyDbContext _dbContext;

        0 references | Hamza Bikić, 1 day ago | 1 author, 1 change
        public OfferContentBasedService(eTouristAgencyDbContext dbContext)
        {
            _dbContext = dbContext;
        }

        2 references | Hamza Bikić, 18 hours ago | 1 author, 2 changes
        public async Task<List<User>> GetUsersForOfferAsync(Offer offer)
        {
            var boardTypeId = offer.BoardTypeId.ToString();
            var hotelId = offer.HotelId.ToString();
            var cityId = offer.Hotel.CityId.ToString();
            var countryId = offer.Hotel.City.CountryId.ToString();

            var userTags = await _dbContext.UserTags.Include(x => x.User).Where(x => x.User.IsActive &&
                                                                 x.CreatedOn > DateTime.Now.AddYears(-1) &&
                                                                 (x.Tag == boardTypeId ||
                                                                 x.Tag == hotelId ||
                                                                 x.Tag == cityId ||
                                                                 x.Tag == countryId)).ToListAsync();

            var users = userTags.DistinctBy(x => x.UserId).Select(x => x.User).ToList();
            List<KeyValuePair<User, int>> countOfMutualTagsPerUser = new List<KeyValuePair<User, int>>();

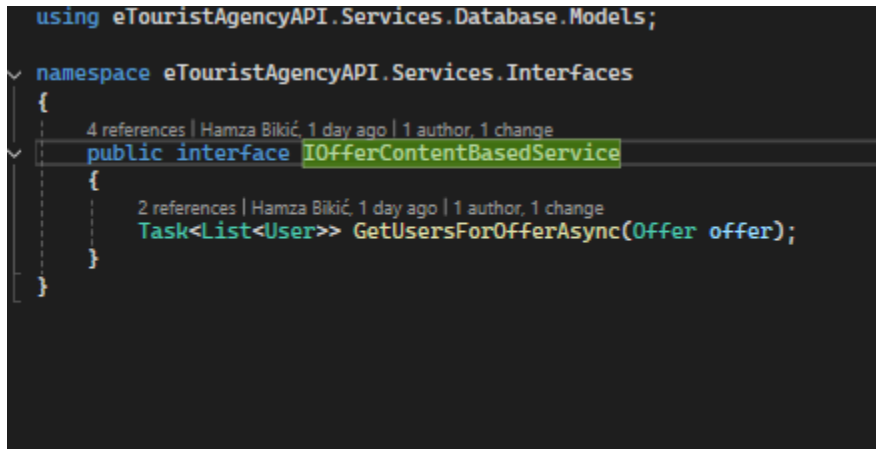
            foreach (var user in users)
            {
                int countOfMutualTags = 0;

                foreach (var userTag in userTags.Where(x => x.UserId == user.Id))
                {
                    if (userTag.Tag == boardTypeId ||
                        userTag.Tag == hotelId ||
                        userTag.Tag == cityId ||
                        userTag.Tag == countryId)
                    {
                        countOfMutualTags++;
                    }
                }

                countOfMutualTagsPerUser.Add(new KeyValuePair<User, int>(user, countOfMutualTags));
            }

            return countOfMutualTagsPerUser.OrderByDescending(x => x.Value).Take(100).Select(x => x.Key).ToList();
        }
    }
}
```

- **Screenshot intefejsa**



```
using eTouristAgencyAPI.Services.Database.Models;

namespace eTouristAgencyAPI.Services.Interfaces
{
    4 references | Hamza Bikić, 1 day ago | 1 author, 1 change
    public interface IOfferContentBasedService
    {
        2 references | Hamza Bikić, 1 day ago | 1 author, 1 change
        Task<List<User>> GetUsersForOfferAsync(Offer offer);
    }
}
```

Kako koristiti sistem preporuke?

Sistem preporuke se aktivira automatski kroz proces upravljanja ponudama u aplikaciji. Evo detaljnog opisa toka:

Scenario aktivacije sistema preporuke

Korak 1: Kreiranje draft ponude

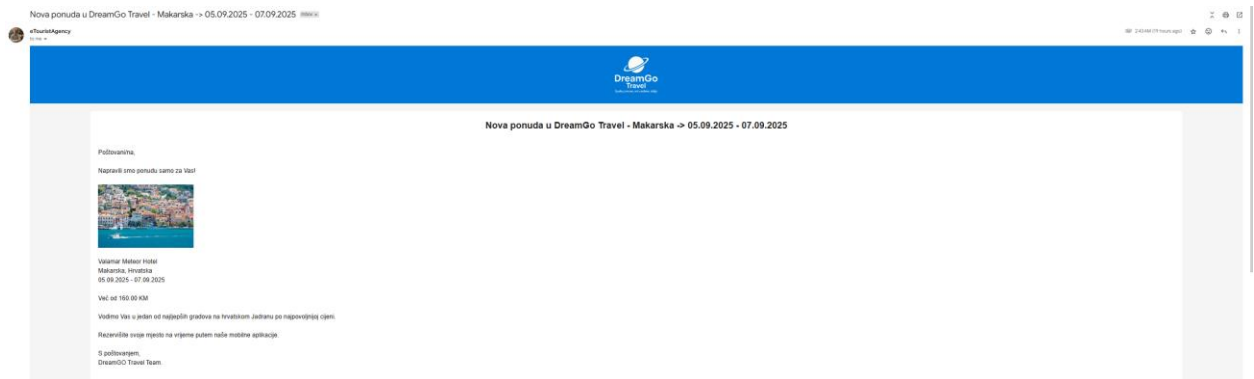
Prvo se kreira ponuda sa statusom "Draft" koja još uvek nije dostupna korisnicima.

Korak 2: Aktivacija ponude - Trigger sistema preporuke

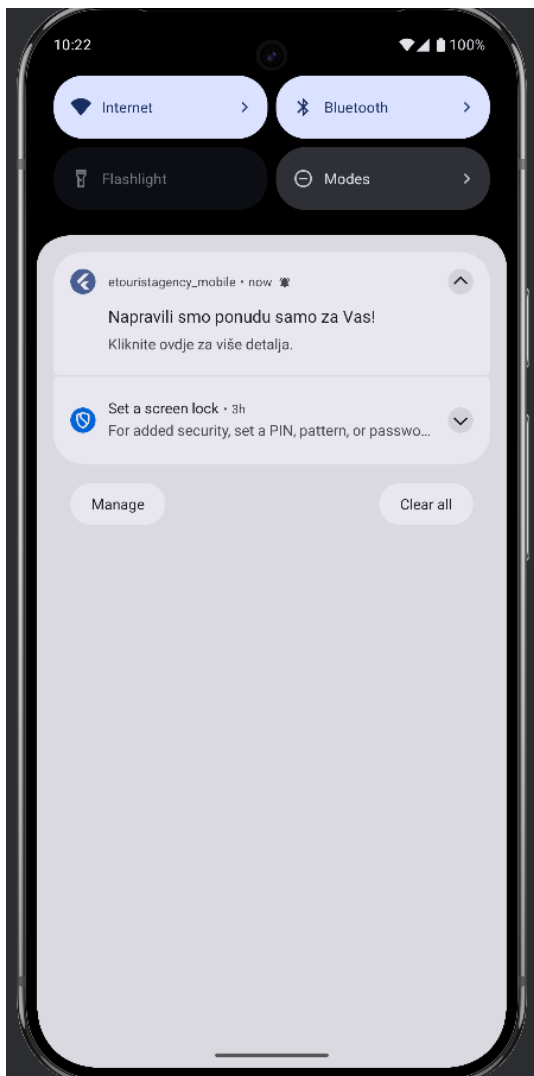
Kada administrator odluči da aktivira ponudu, dešava se sledeće:

1. Status update: Ponuda se menja iz "Draft" u "Aktivna"
2. Automatski poziv algoritma: Sistem automatski poziva OfferContentBasedService.GetUsersForOfferAsync() metodu
3. Generisanje liste preporučenih korisnika: Algoritam vraća do 100 najrelevantnijih korisnika na osnovu njihovih historijskih interesovanja
4. Kreiranje personalizovanih email-ova: Za svakog preporučenog korisnika generiše se prilagođen email sa detaljima ponude, te se isti šalje RabbitMQ-u.
5. Kreiranje personalizovanih push notifikacija: Za svakog preporučenog korisnika generiše se sadržaj push notifikacije, te se isti šalje RabbitMQ-u.

- Primjer e-mail notifikacije



- Primjer push notifikacije



Klikom na push notifikaciju, korisnik će biti automatski preusmjeren na screen o detaljima ponude na kojem može započeti proces kreiranja rezervacije.

