Fakultet informacijskih tehnologija

litTrack – Recommender sistem
Razvoj softvera II

Predmetni profesori:

Prof. dr. Elmir Babović

Prof. dr. Denis Mušić

Student:

Elmedina Mahinić , IB210078

Mostar, januar 2026

Contents

1. Uvod

U aplikaciji LitTrack implementiran je sistem preporuke s ciljem pružanja personaliziranog korisničkog iskustva prilikom kupovine knjiga. Osnovni cilj ovog sistema jeste olakšati korisnicima pronalazak knjiga koje odgovaraju njihovim interesima i čitateljskim navikama, te istovremeno unaprijediti ukupno korisničko iskustvo i podstaći aktivniju upotrebu aplikacije.

Implementiran je sistem preporuke zasnovan na **Collaborative filtering** pristupu, koji koristi zajedničke obrasce ponašanja korisnika prilikom kupovine knjiga kako bi generisao relevantne i personalizirane preporuke. Sistem analizira koje su knjige korisnici kupovali zajedno i na osnovu tih informacija predlaže dodatne knjige drugim korisnicima.

Implementacija sistema preporuke, kao i lokacija njegovog source code-a detaljno su predstavljeni u ostatku rada.

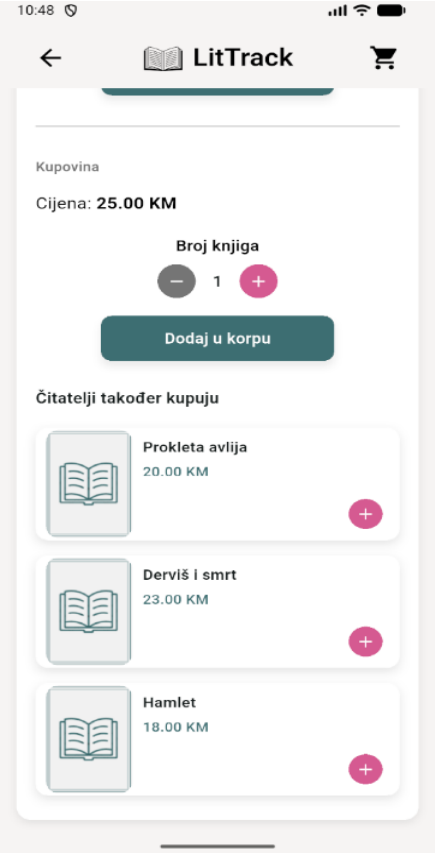
2. Collaborative filtering

Nakon što korisnik pristupi detaljima određene knjige, scroll-anjem ka dole, ispod dugmeta “Dodaj u korpu” može vidjeti sekciju “Čitatelji također kupuju” koja predstavlja dodatne knjige generisane putem sistema preporuke. Ove preporuke će biti generisane na osnovu zajedničkog ponašanja korisnika. Na primjer, kada korisnik naruči knjigu zajedno sa nekim drugim knjigama, ti podaci će se koristiti za kreiranje preporuka drugim korisnicima. Na taj način, korisnicima se nude knjige koje su drugi korisnici često naručivali sa trenutno posmatranom knjigom.

Da bi korisnik pristupio ovom dijelu aplikacije prvo mora biti prijavljen na svoj profil. Zatim treba da klikne na željenu knjigu i tima pristupa njenim detaljima. Scroll-anjem ka dole može vidjeti sekciju “Čitatelji također kupuju” gdje su prikazane druge knjige koje su kupci naručivali zajedno sa trenutno posmatranom knjigom (Slika 1).

Putanja gdje se prikazuju preporuke na frontendu je:

/LitTrack_RS2/litTrack/UI/littrack_mobile/lib/screens/knjiga_details_screen.dart

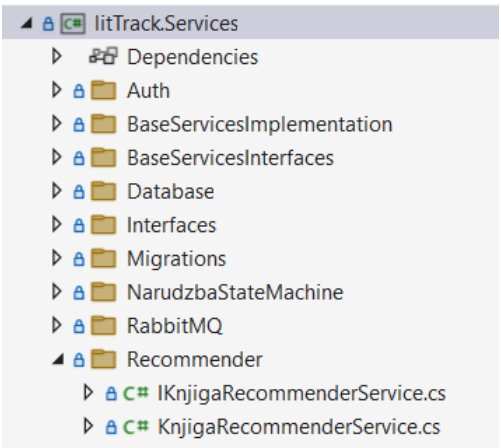


Slika 1.

2.1. Implementacija

Za pristup source code-u, koji će biti objašnjen u nastavku, potrebno je navigirati u:

/LitTrack_RS2/litTrack/litTrack.Services/Recommender/KnjigaRecommenderService.cs. (Slika 2)



Slika 2

2.1.1 Osnovna uloga servisa

Klasa **KnjigaRecommenderService** predstavlja centralnu komponentu sistema preporuke u aplikaciji LitTrack. Njena osnovna uloga je generisanje personaliziranih preporuka knjiga na osnovu historijskih podataka o kupovini korisnika. Servis koristi biblioteku **ML.NET** i algoritam **Matrix Factorization**, koji pripada grupi Collaborative Filtering metoda.

Servis je povezan sa bazom podataka putem Entity Framework konteksta _210078Context, dok se za mapiranje entiteta u DTO objekte koristi biblioteka **Mapster** (Slika 3).

```
public class KnjigaRecommenderService : IKnjigaRecommenderService
{
    private readonly IMapper _mapper;
    private readonly _210078Context _context;
    private static MLContext? mlContext;
    private static ITransformer? model;
    private static readonly object isLocked = new();
    private static Task? trainingTask;


    private const string ModelFilePath = "books-recommender-model.zip";

    0 references | ElmedinaMahinic, 18 days ago | 1 author, 1 change
    public KnjigaRecommenderService(_210078Context context, IMapper mapper)
    {
        _context = context;
        _mapper = mapper;
    }
```

Slika 3

2.1.2 Učitavanje i inicijalizacija modela

Model mašinskog učenja se inicijalizira samo jednom te se potom koristi za sve buduće zahtjeve.

MLContext i **ITransformer** model su definisani kao static, čime se osigurava da se model dijeli između svih instanci servisa. Korišten je mehanizam zaključavanja (lock) kako bi se spriječilo paralelno treniranje modela prilikom istovremenih zahtjeva korisnika.

Ako prethodno trenirani model postoji na disku (books-recommender-model.zip), on se učitava i odmah koristi. Ako model ne postoji, pokreće se proces treniranja metodom TrainData(). Na ovaj način sistem postiže visoku efikasnost i stabilnost bez ponovnog treniranja pri svakom zahtjevu (Slika 4).

```
public async Task<List<KnjigaDTO>> GetRecommendedBooks(int knjigaId)
{
    var exists = await _context.StavkaNarudzbes
        .AnyAsync(x => x.KnjigaId == knjigaId && !x.IsDeleted);

    if (!exists)
        return new List<KnjigaDTO>();

    if (mlContext == null)
    {
        bool needsTraining = false;

        lock (isLocked)
        {
            if (mlContext == null)
            {
                mlContext = new MLContext();

                if (File.Exists(ModelFilePath))
                {
                    using var stream = new FileStream(ModelFilePath, FileMode.Open, FileAccess.Read, FileShare.Read);
                    model = mlContext.Model.Load(stream, out _);
                }
                else
                {
                    needsTraining = true;
                }
            }
        }

        if (needsTraining)
        {
            lock (isLocked)
            {
                trainingTask ??= TrainData();
            }

            await trainingTask;
        }

    }
```

Slika 4

2.1.2 Generisanje preporuka

Generisanje preporuka se vrši kroz sljedeće korake:

1. Provjerava da li data knjiga ikada bila kupljena. Ako ne postoji nijedna stavka narudžbe za tu knjigu, metoda odmah vraća praznu listu, jer bez historije kupovine nije moguće generisati relevantne preporuke (Slika 4).
2. Nakon uspješne inicijalizacije modela, iz baze podataka se dohvaćaju sve knjige osim trenutno pregledavane, jer nema potrebe da se knjiga preporučuje sama sebi (Slika 5).
3. Za svaku knjigu koristi se trenirani model kako bi se izračunao stepen povezanosti sa trenutno pregledavanom knjigom. Predviđanja se vrše pomoću PredictionEngine mehanizma, koji za svaku kombinaciju knjiga generiše score vrijednost – procjenu vjerovatnoće da se te knjige kupuju zajedno (Slika 5).
4. Svaka knjiga i njen pripadajući score dodaju se u listu rezultata, kako bi se omogućila daljnja obrada i rangiranje preporuka (Slika 5)
5. Lista preporuka se sortira prema vrijednosti score-a (od najveće ka najmanjoj) (Slika 5)
6. Na kraju se izdvajaju tri knjige s najvišim score-om, odnosno one koje su najčešće kupljene zajedno s trenutno pregledavanom knjigom (Slika 5)

7. Dobijeni rezultati se mapiraju u KnjigaDTO objekte i vraćaju kao konačna lista preporučenih knjiga Slika 5)

```
    if (model == null)
        return new List<KnjigaDTO>();

    var books = await _context.Knjigas
        .Where(x => x.KnjigaId != knjigaId && !x.IsDeleted)
        .ToListAsync();

    using var predictionEngine = mlContext.Model.CreatePredictionEngine<BookEntry, CopurchasePrediction>(model);

    var predictions = new List<(Knjiga, float)>();

    foreach (var book in books)
    {
        var prediction = predictionEngine.Predict(new BookEntry
        {
            BookID = (uint)knjigaId,
            CoPurchaseBookID = (uint)book.KnjigaId
        });

        predictions.Add((book, prediction.Score));
    }

    var topBooks = predictions
        .OrderByDescending(x => x.Item2)
        .Take(3)
        .Select(x => x.Item1)
        .ToList();

    return _mapper.Map<List<KnjigaDTO>>(topBooks);
}
```

Slika 5

2.1.3 Priprema podataka za treniranje

Metoda **TrainData()** (Slika 6) prikuplja podatke iz baze:

1. Učitavaju se sve narudžbe i njihove stavke (Narudzba → StavkaNarudzbe).
2. Za svaku narudžbu formira se skup knjiga koje su kupljene zajedno.
3. Iz tog skupa generišu se parovi knjiga: Ako su knjige A i B kupljene u istoj narudžbi, kreiraju se zapisi (A,B) i (B,A).

Svaki takav par predstavlja jedan uzorak za treniranje i mapira se u klasu BookEntry, koja sadrži:

- **BookID** – osnovna knjiga
- **CoPurchaseBookID** – knjiga kupljena zajedno s njom
- **Label** – vrijednost 1 koja označava postojanje veze između knjiga

```
public async Task TrainData()
{
    if (mlContext == null)
        mlContext = new MLContext();

    var orders = await _context.Narudzbas
        .Include(o => o.StavkaNarudzbes)
        .Where(o => !o.IsDeleted)
        .ToListAsync();

    var data = new List<BookEntry>();

    foreach (var order in orders)
    {
        var itemIds = order.StavkaNarudzbes
            .Where(s => !s.IsDeleted)
            .Select(s => s.KnjigaId)
            .Distinct()
            .ToList();

        for (int i = 0; i < itemIds.Count; i++)
            for (int j = 0; j < itemIds.Count; j++)
            {
                if (i == j) continue;

                data.Add(new BookEntry
                {
                    BookID = (uint)itemIds[i],
                    CoPurchaseBookID = (uint)itemIds[j],
                    Label = 1f
                });
            }
    }

    if (!data.Any())
        return;

    var trainData = mlContext.Data.LoadFromEnumerable(data);
```

Slika 6

2.1.4 Treniranje modela

Za treniranje se koristi **Matrix Factorization** algoritam (Slika 7) iz ML.NET biblioteke i podešeni su sljedeći hiperparametri:

- **SquareLossOneClass** – funkcija gubitka pogodna za implicitne povratne informacije
- **NumberOfIterations** = 100
- **Alpha, Lambda, C** – parametri regularizacije i učenja

Model se trenira nad pripremljenim podacima i zatim se trajno sprema u fajl books-recommender-model.zip.

```
        var options = new MatrixFactorizationTrainer.Options
        {
            MatrixColumnIndexColumnName = nameof(BookEntry.BookID),
            MatrixRowIndexColumnName = nameof(BookEntry.CoPurchaseBookID),
            LabelColumnName = nameof(BookEntry.Label),
            LossFunction = MatrixFactorizationTrainer.LossFunctionType.SquareLossOneClass,
            Alpha = 0.01,
            Lambda = 0.005,
            NumberOfIterations = 100,
            C = 0.00001
        };

        var est = mlContext.Recommendation().Trainers.MatrixFactorization(options);
        model = est.Fit(trainData);

        using var stream = new FileStream(ModelFilePath, FileMode.Create, FileAccess.Write, FileShare.Write);
        mlContext.Model.Save(model, trainData.Schema, stream);
    }
}
```

Slika 7