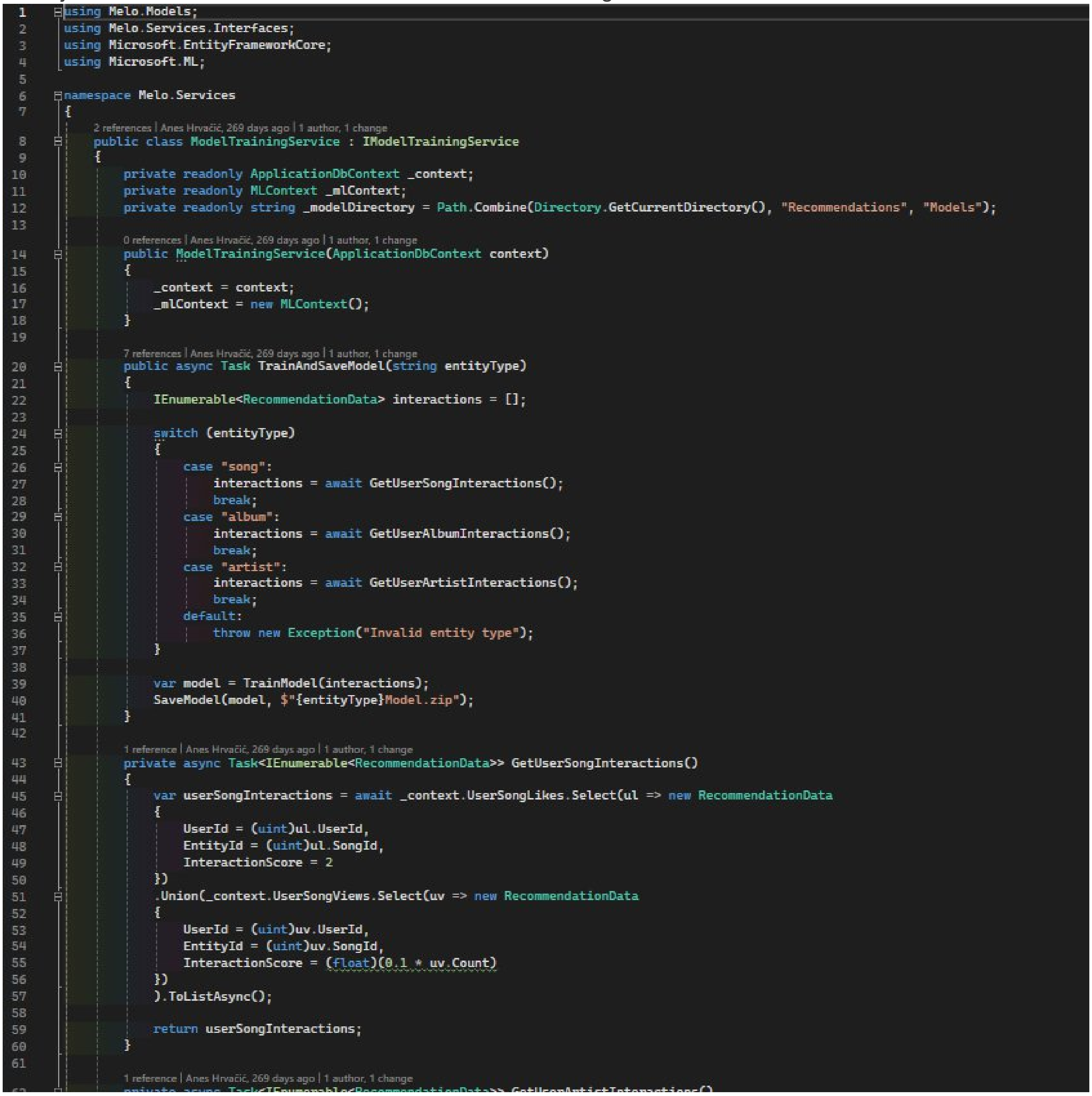
Melo

Dokumentacija sistema preporuke

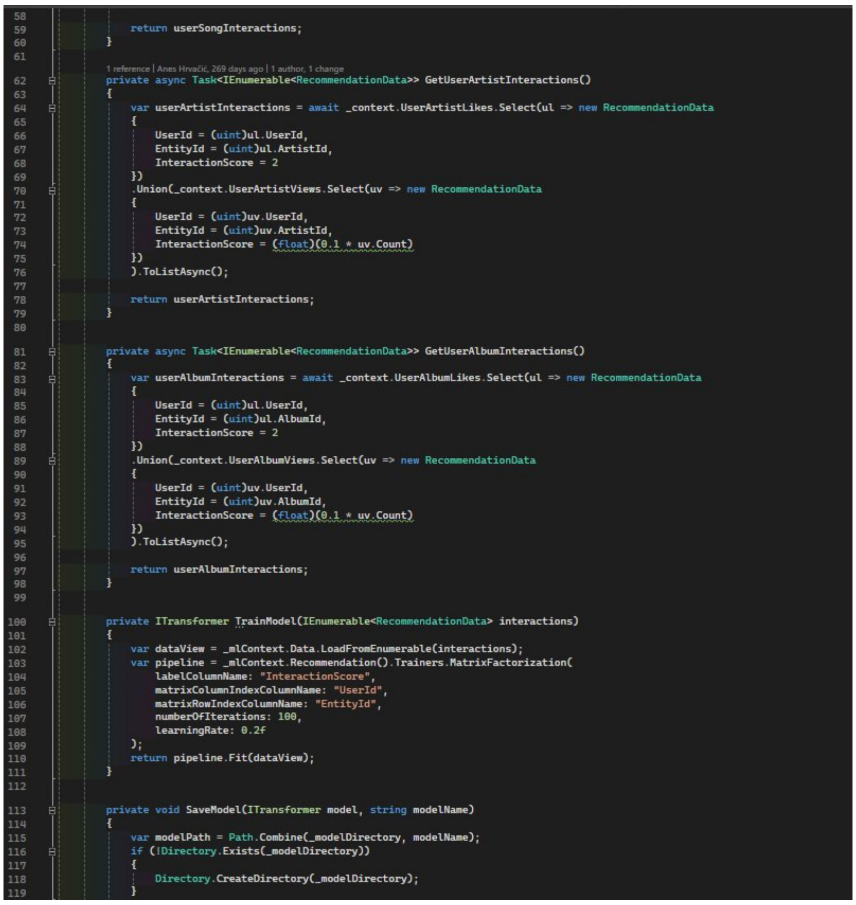
Korisnici Melo aplikacije imaju mogućnost vidjeti preporučen sadržaj na početnoj stranici, odnosno preporučene pjesme, albume i umjetnike. Sistem preporuke je napravljen na principu *„user based collaborative filtering“.*

Prvi korak cijelog procesa jeste treniranje modela:

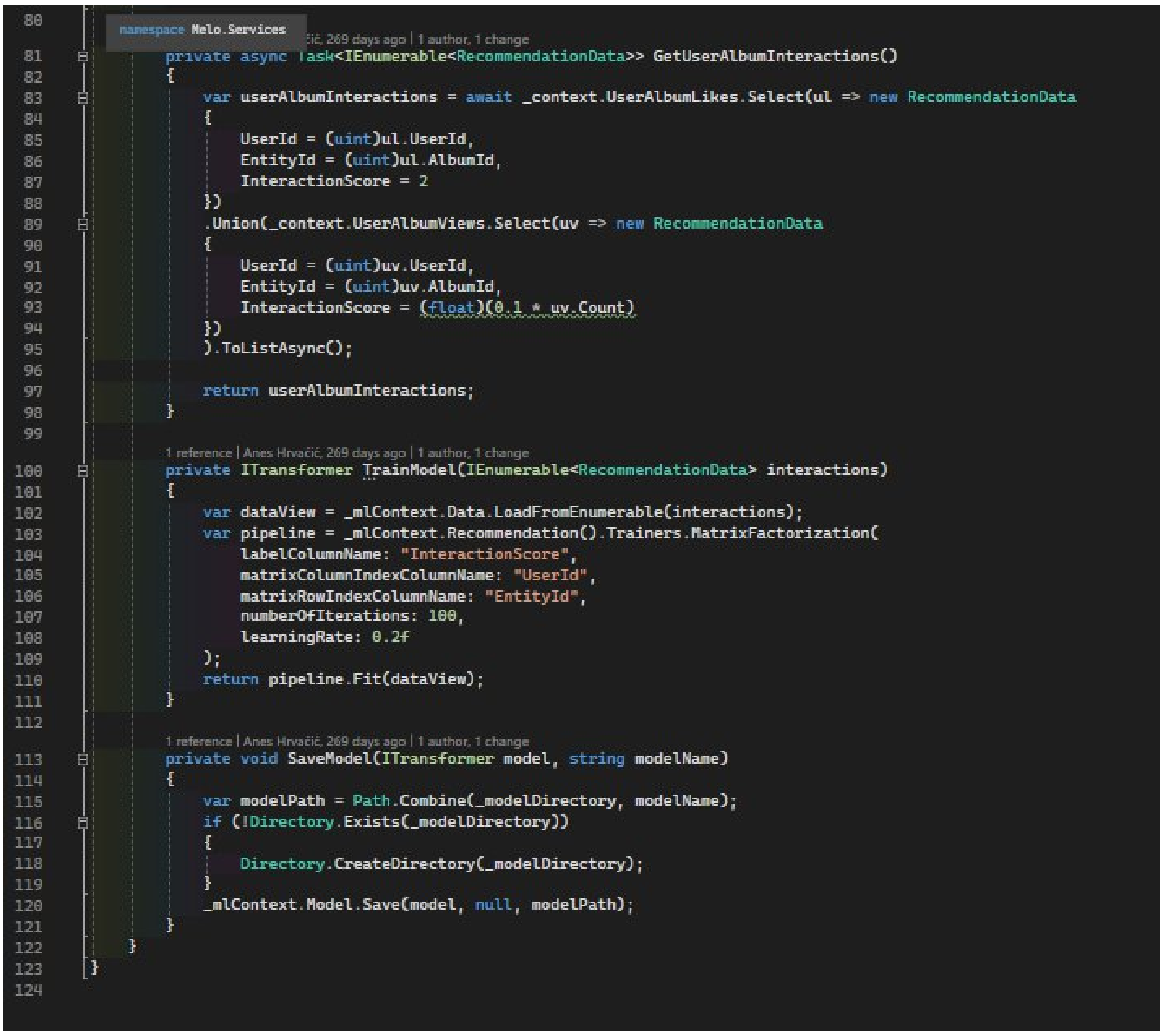
Putanja do koda: Melo\Melo.Services\Services\ModelTrainingService.cs



```csharp
using Melo.Models;
using Melo.Services.Interfaces;
using Microsoft.EntityFrameworkCore;
using Microsoft.ML;

namespace Melo.Services
{
    2 references | Anes Hrvačić, 269 days ago | 1 author, 1 change
    public class ModelTrainingService : IModelTrainingService
    {
        private readonly ApplicationDbContext _context;
        private readonly MLContext _mlContext;
        private readonly string _modelDirectory = Path.Combine(Directory.GetCurrentDirectory(), "Recommendations", "Models");

        0 references | Anes Hrvačić, 269 days ago | 1 author, 1 change
        public ModelTrainingService(ApplicationDbContext context)
        {
            _context = context;
            _mlContext = new MLContext();
        }

        7 references | Anes Hrvačić, 269 days ago | 1 author, 1 change
        public async Task TrainAndSaveModel(string entityType)
        {
            IEnumerable<RecommendationData> interactions = [];

            switch (entityType)
            {
                case "song":
                    interactions = await GetUserSongInteractions();
                    break;
                case "album":
                    interactions = await GetUserAlbumInteractions();
                    break;
                case "artist":
                    interactions = await GetUserArtistInteractions();
                    break;
                default:
                    throw new Exception("Invalid entity type");
            }

            var model = TrainModel(interactions);
            SaveModel(model, $"{entityType}Model.zip");
        }

        1 reference | Anes Hrvačić, 269 days ago | 1 author, 1 change
        private async Task<IEnumerable<RecommendationData>> GetUserSongInteractions()
        {
            var userSongInteractions = await _context.UserSongLikes.Select(ul => new RecommendationData
            {
                UserId = (uint)ul.UserId,
                EntityId = (uint)ul.SongId,
                InteractionScore = 2
            })
            .Union(_context.UserSongViews.Select(uv => new RecommendationData
            {
                UserId = (uint)uv.UserId,
                EntityId = (uint)uv.SongId,
                InteractionScore = (float)(0.1 * uv.Count)
            })
            ).ToListAsync();

            return userSongInteractions;
        }

        1 reference | Anes Hrvačić, 269 days ago | 1 author, 1 change
        private async Task<IEnumerable<RecommendationData>> GetUserArtistInteractions()
```



```
                return userSongInteractions;
            }

            1 reference | Anes Hrvačić, 269 days ago | 1 author, 1 change
            private async Task<IEnumerable<RecommendationData>> GetUserArtistInteractions()
            {
                var userArtistInteractions = await _context.UserArtistLikes.Select(ul => new RecommendationData
                {
                    UserId = (uint)ul.UserId,
                    EntityId = (uint)ul.ArtistId,
                    InteractionScore = 2
                })
                .Union(_context.UserArtistViews.Select(uv => new RecommendationData
                {
                    UserId = (uint)uv.UserId,
                    EntityId = (uint)uv.ArtistId,
                    InteractionScore = (float)(0.1 * uv.Count)
                })
                ).ToListAsync();

                return userArtistInteractions;
            }


            private async Task<IEnumerable<RecommendationData>> GetUserAlbumInteractions()
            {
                var userAlbumInteractions = await _context.UserAlbumLikes.Select(ul => new RecommendationData
                {
                    UserId = (uint)ul.UserId,
                    EntityId = (uint)ul.AlbumId,
                    InteractionScore = 2
                })
                .Union(_context.UserAlbumViews.Select(uv => new RecommendationData
                {
                    UserId = (uint)uv.UserId,
                    EntityId = (uint)uv.AlbumId,
                    InteractionScore = (float)(0.1 * uv.Count)
                })
                ).ToListAsync();

                return userAlbumInteractions;
            }


            private ITransformer TrainModel(IEnumerable<RecommendationData> interactions)
            {
                var dataView = _mlContext.Data.LoadFromEnumerable(interactions);
                var pipeline = _mlContext.Recommendation().Trainers.MatrixFactorization(
                    labelColumnName: "InteractionScore",
                    matrixColumnIndexColumnName: "UserId",
                    matrixRowIndexColumnName: "EntityId",
                    numberOfIterations: 100,
                    learningRate: 0.2f
                );
                return pipeline.Fit(dataView);
            }


            private void SaveModel(ITransformer model, string modelName)
            {
                var modelPath = Path.Combine(_modelDirectory, modelName);
                if (!Directory.Exists(_modelDirectory))
                {
                    Directory.CreateDirectory(_modelDirectory);
                }
```



namespace Melo.Services

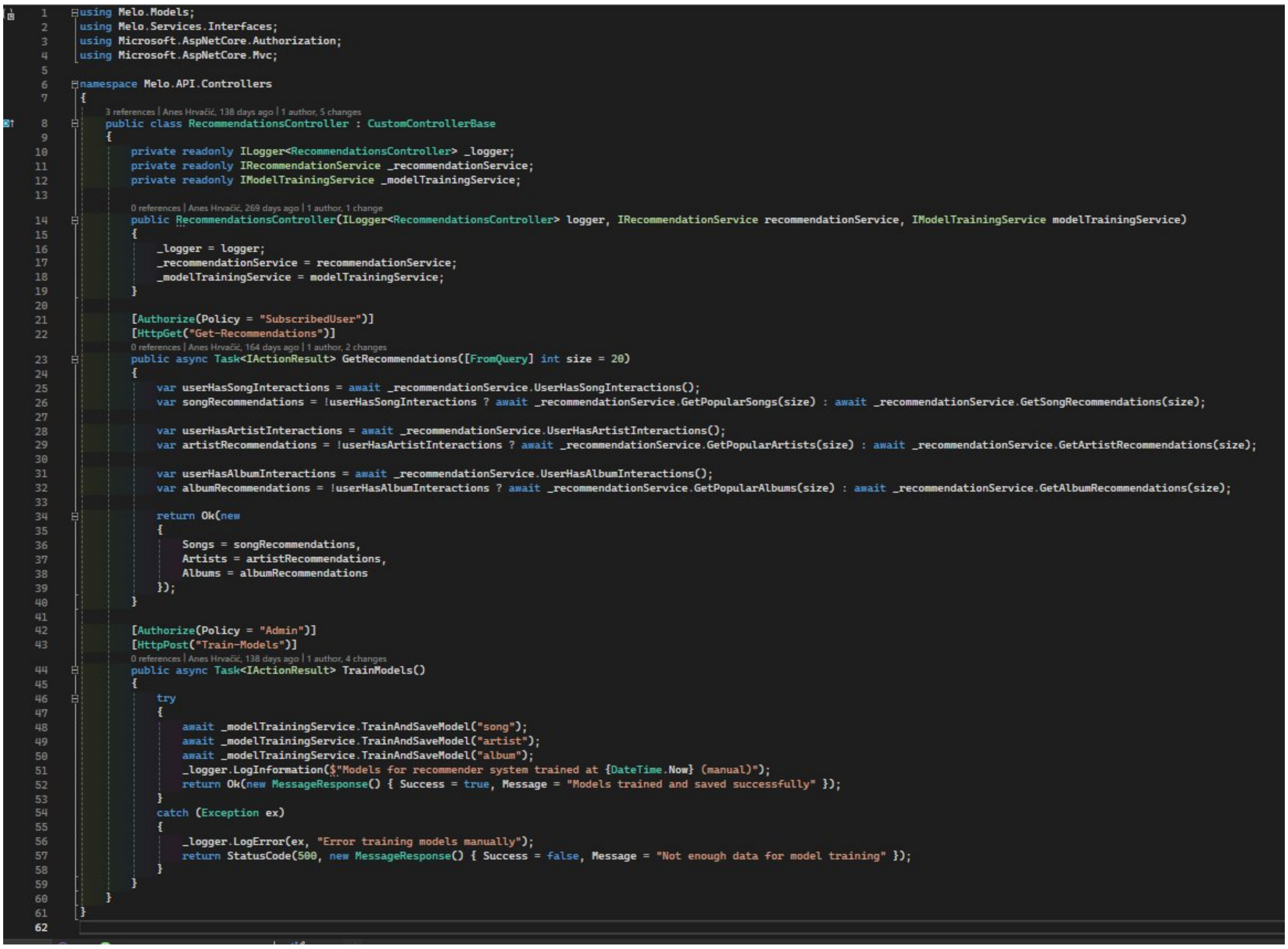
```
            1 reference | Anes Hrvačić, 269 days ago | 1 author, 1 change
            private async Task<IEnumerable<RecommendationData>> GetUserAlbumInteractions()
            {
                var userAlbumInteractions = await _context.UserAlbumLikes.Select(ul => new RecommendationData
                {
                    UserId = (uint)ul.UserId,
                    EntityId = (uint)ul.AlbumId,
                    InteractionScore = 2
                })
                .Union(_context.UserAlbumViews.Select(uv => new RecommendationData
                {
                    UserId = (uint)uv.UserId,
                    EntityId = (uint)uv.AlbumId,
                    InteractionScore = (float)(0.1 * uv.Count)
                })
                ).ToListAsync();

                return userAlbumInteractions;
            }

            1 reference | Anes Hrvačić, 269 days ago | 1 author, 1 change
            private ITransformer TrainModel(IEnumerable<RecommendationData> interactions)
            {
                var dataView = _mlContext.Data.LoadFromEnumerable(interactions);
                var pipeline = _mlContext.Recommendation().Trainers.MatrixFactorization(
                    labelColumnName: "InteractionScore",
                    matrixColumnIndexColumnName: "UserId",
                    matrixRowIndexColumnName: "EntityId",
                    numberOfIterations: 100,
                    learningRate: 0.2f
                );
                return pipeline.Fit(dataView);
            }

            1 reference | Anes Hrvačić, 269 days ago | 1 author, 1 change
            private void SaveModel(ITransformer model, string modelName)
            {
                var modelPath = Path.Combine(_modelDirectory, modelName);
                if (!Directory.Exists(_modelDirectory))
                {
                    Directory.CreateDirectory(_modelDirectory);
                }
```

```
                _mlContext.Model.Save(model, null, modelPath);
            }
        }
    }
```

Iz baze se povlače interakcije korisnika sa entitetima (pjesme, albumi, umjetnici) te se računa **InteractionScore**. InteractionScore se računa tako što lajk vrijedi ***2***, a pregledi vrijede ***brojPregleda * 0.1***. Sakupljene interakcije se unose u *Matrix Factorization Trainer* i dobijaju modeli koji se čuvaju u file system.

Sljedeći korak cijelog procesa jeste dobijanje preporuka:

Putanja do koda: Melo\Melo.API\Controllers\RecommendationController.cs

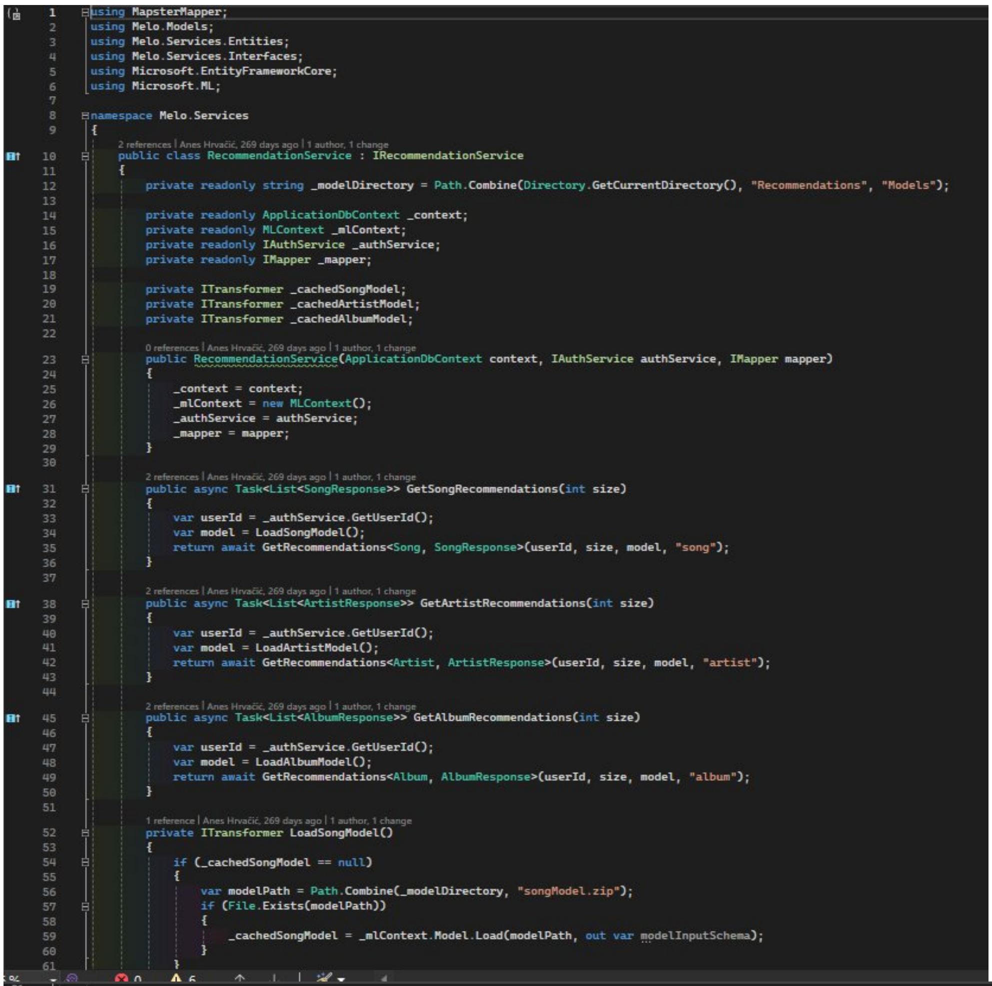


```csharp
using Melo.Models;
using Melo.Services.Interfaces;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;

namespace Melo.API.Controllers
{
    3 references | Anes Hrvačić, 138 days ago | 1 author, 5 changes
    public class RecommendationsController : CustomControllerBase
    {
        private readonly ILogger<RecommendationsController> _logger;
        private readonly IRecommendationService _recommendationService;
        private readonly IModelTrainingService _modelTrainingService;

        0 references | Anes Hrvačić, 269 days ago | 1 author, 1 change
        public RecommendationsController(ILogger<RecommendationsController> logger, IRecommendationService recommendationService, IModelTrainingService modelTrainingService)
        {
            _logger = logger;
            _recommendationService = recommendationService;
            _modelTrainingService = modelTrainingService;
        }

        [Authorize(Policy = "SubscribedUser")]
        [HttpGet("Get-Recommendations")]
        0 references | Anes Hrvačić, 164 days ago | 1 author, 2 changes
        public async Task<IActionResult> GetRecommendations([FromQuery] int size = 20)
        {
            var userHasSongInteractions = await _recommendationService.UserHasSongInteractions();
            var songRecommendations = !userHasSongInteractions ? await _recommendationService.GetPopularSongs(size) : await _recommendationService.GetSongRecommendations(size);

            var userHasArtistInteractions = await _recommendationService.UserHasArtistInteractions();
            var artistRecommendations = !userHasArtistInteractions ? await _recommendationService.GetPopularArtists(size) : await _recommendationService.GetArtistRecommendations(size);

            var userHasAlbumInteractions = await _recommendationService.UserHasAlbumInteractions();
            var albumRecommendations = !userHasAlbumInteractions ? await _recommendationService.GetPopularAlbums(size) : await _recommendationService.GetAlbumRecommendations(size);

            return Ok(new
            {
                Songs = songRecommendations,
                Artists = artistRecommendations,
                Albums = albumRecommendations
            });
        }

        [Authorize(Policy = "Admin")]
        [HttpPost("Train-Models")]
        0 references | Anes Hrvačić, 138 days ago | 1 author, 4 changes
        public async Task<IActionResult> TrainModels()
        {
            try
            {
                await _modelTrainingService.TrainAndSaveModel("song");
                await _modelTrainingService.TrainAndSaveModel("artist");
                await _modelTrainingService.TrainAndSaveModel("album");
                _logger.LogInformation($"Models for recommender system trained at {DateTime.Now} (manual)");
                return Ok(new MessageResponse() { Success = true, Message = "Models trained and saved successfully" });
            }
            catch (Exception ex)
            {
                _logger.LogError(ex, "Error training models manually");
                return StatusCode(500, new MessageResponse() { Success = false, Message = "Not enough data for model training" });
            }
        }
    }
}
```

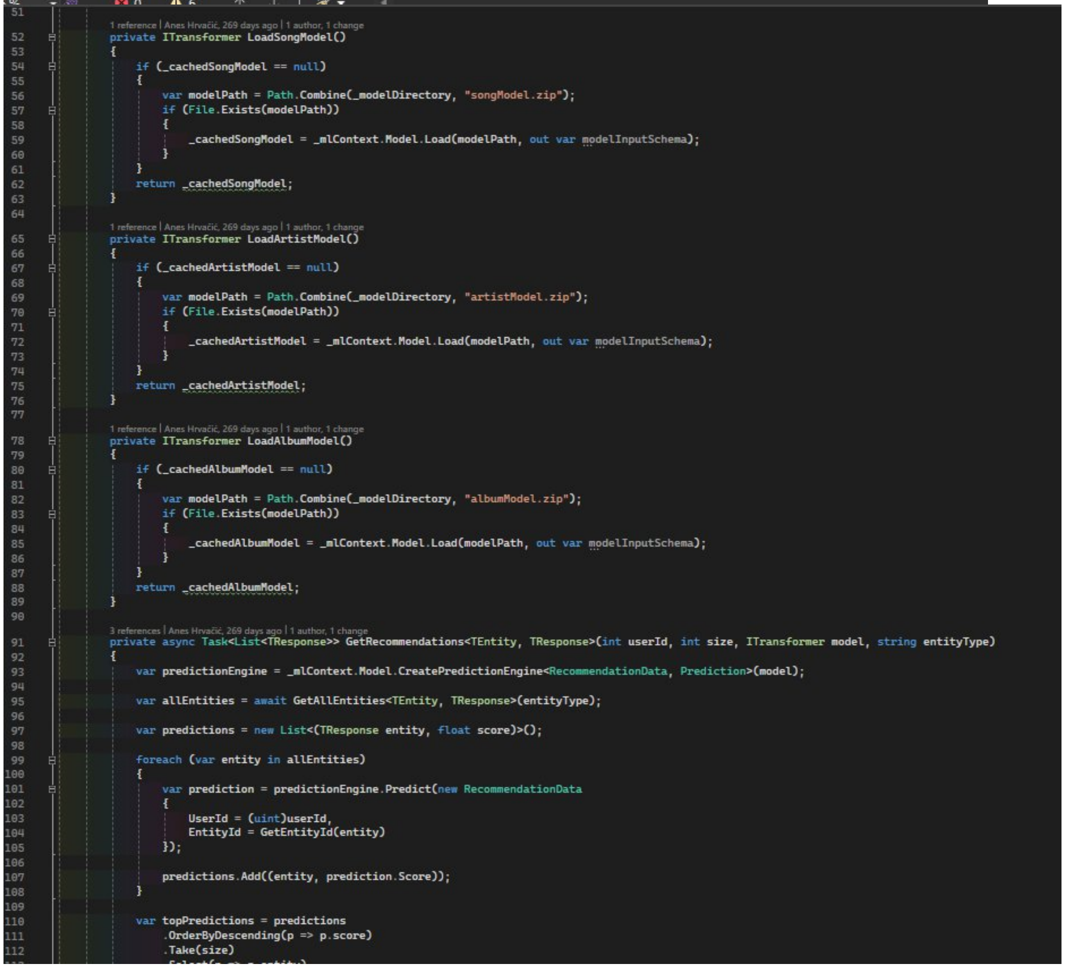
U slučaju da korisnik nema dovoljno interakcija da mu se preporuči sadržaj, dobit će listu najpopularnijih entiteta.

Na slici također vidimo i API endpoint za treniranje modela, koji je dostupan administratorima kao način manuelnog treniranja modela (više o ovome u nastavku).

Putanja do koda: Melo\Melo.Services\Services\RecommendationService.cs

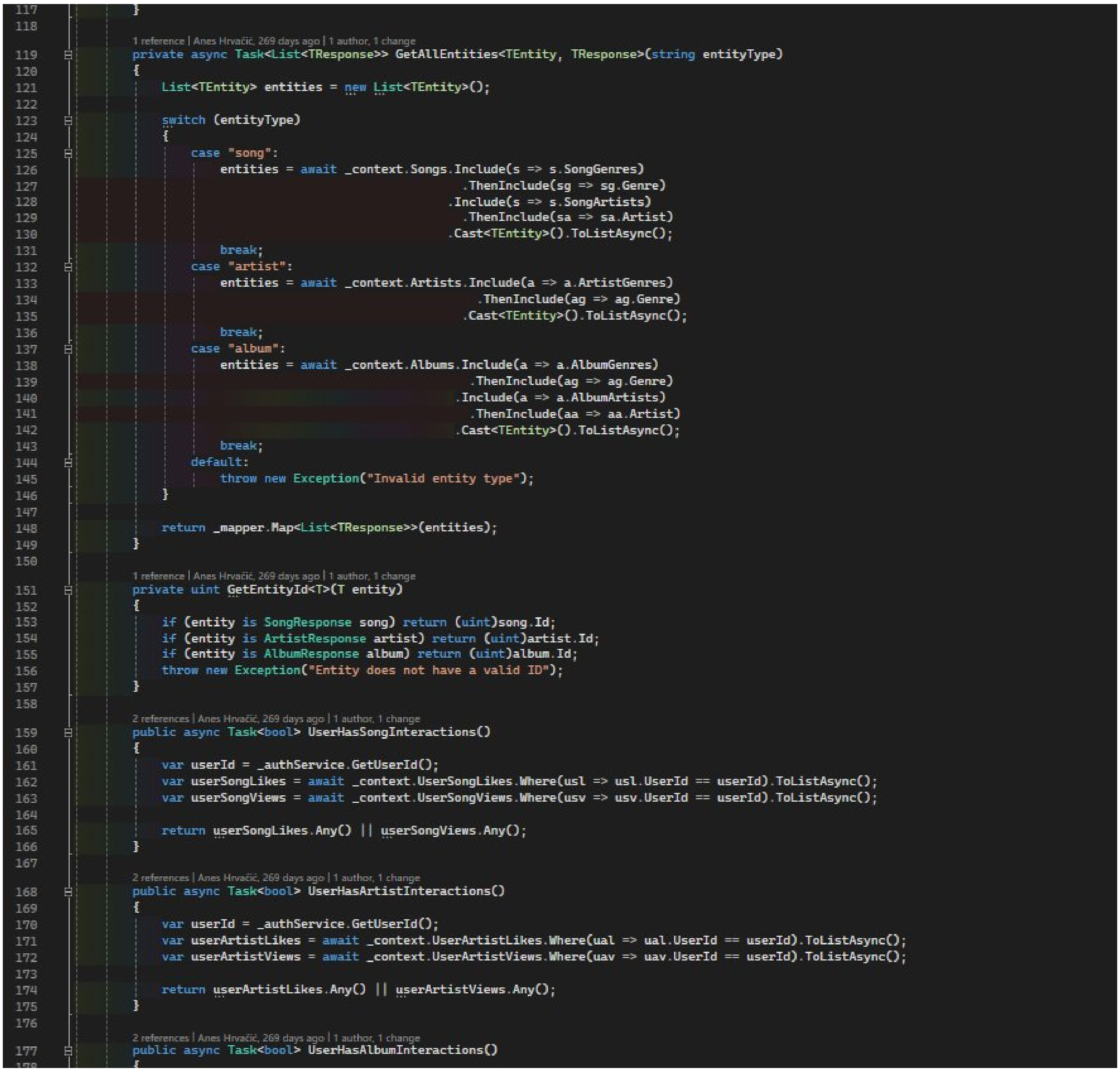


```csharp
using MapsterMapper;
using Melo.Models;
using Melo.Services.Entities;
using Melo.Services.Interfaces;
using Microsoft.EntityFrameworkCore;
using Microsoft.ML;

namespace Melo.Services
{
    2 references | Anes Hrvačić, 269 days ago | 1 author, 1 change
    public class RecommendationService : IRecommendationService
    {
        private readonly string _modelDirectory = Path.Combine(Directory.GetCurrentDirectory(), "Recommendations", "Models");

        private readonly ApplicationDbContext _context;
        private readonly MLContext _mlContext;
        private readonly IAuthService _authService;
        private readonly IMapper _mapper;

        private ITransformer _cachedSongModel;
        private ITransformer _cachedArtistModel;
        private ITransformer _cachedAlbumModel;

        0 references | Anes Hrvačić, 269 days ago | 1 author, 1 change
        public RecommendationService(ApplicationDbContext context, IAuthService authService, IMapper mapper)
        {
            _context = context;
            _mlContext = new MLContext();
            _authService = authService;
            _mapper = mapper;
        }

        2 references | Anes Hrvačić, 269 days ago | 1 author, 1 change
        public async Task<List<SongResponse>> GetSongRecommendations(int size)
        {
            var userId = _authService.GetUserId();
            var model = LoadSongModel();
            return await GetRecommendations<Song, SongResponse>(userId, size, model, "song");
        }

        2 references | Anes Hrvačić, 269 days ago | 1 author, 1 change
        public async Task<List<ArtistResponse>> GetArtistRecommendations(int size)
        {
            var userId = _authService.GetUserId();
            var model = LoadArtistModel();
            return await GetRecommendations<Artist, ArtistResponse>(userId, size, model, "artist");
        }

        2 references | Anes Hrvačić, 269 days ago | 1 author, 1 change
        public async Task<List<AlbumResponse>> GetAlbumRecommendations(int size)
        {
            var userId = _authService.GetUserId();
            var model = LoadAlbumModel();
            return await GetRecommendations<Album, AlbumResponse>(userId, size, model, "album");
        }

        1 reference | Anes Hrvačić, 269 days ago | 1 author, 1 change
        private ITransformer LoadSongModel()
        {
            if (_cachedSongModel == null)
            {
                var modelPath = Path.Combine(_modelDirectory, "songModel.zip");
                if (File.Exists(modelPath))
                {
                    _cachedSongModel = _mlContext.Model.Load(modelPath, out var modelInputSchema);
                }
            }
```

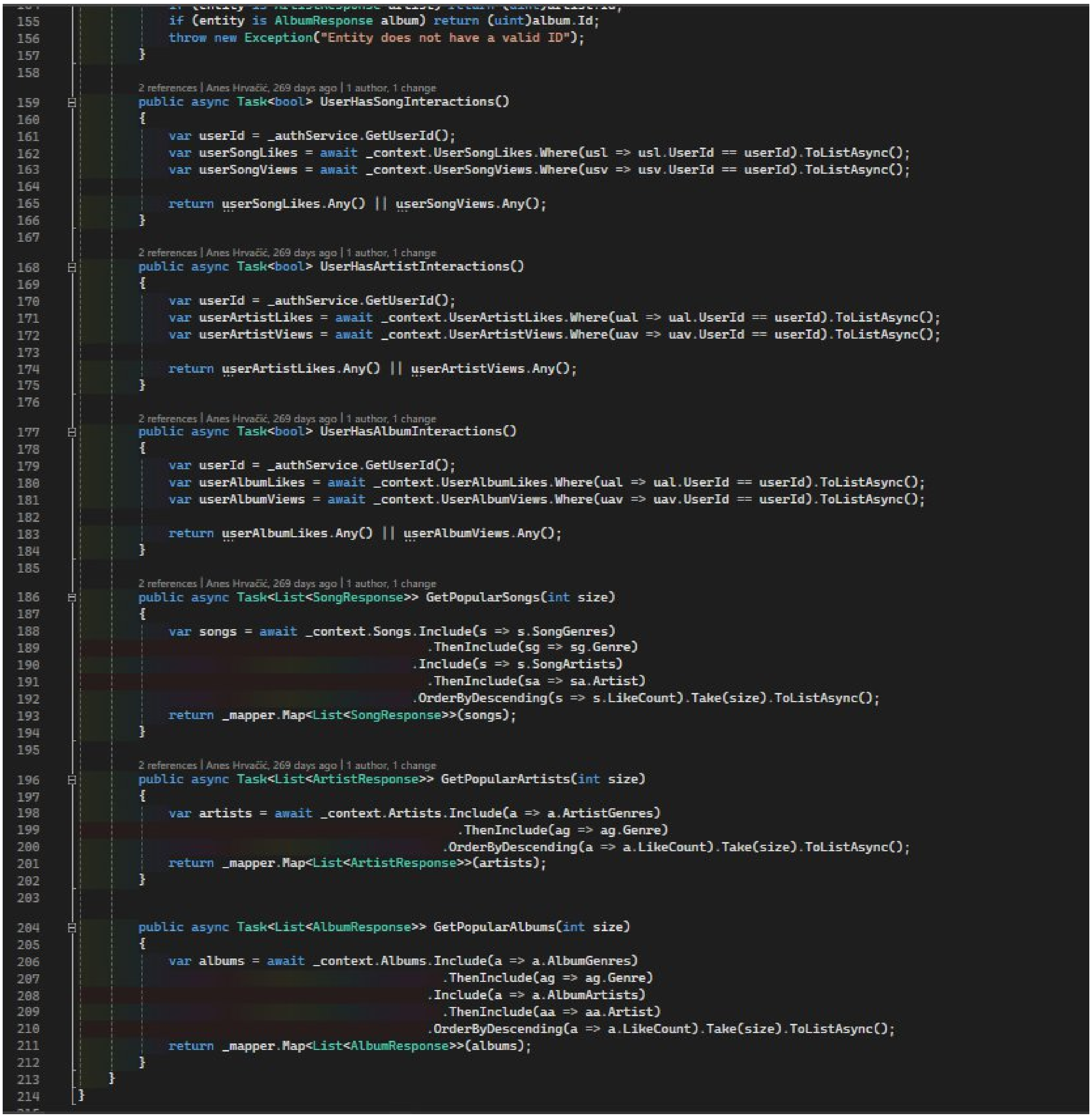


```csharp
        1 reference | Anes Hrvačić, 269 days ago | 1 author, 1 change
        private ITransformer LoadSongModel()
        {
            if (_cachedSongModel == null)
            {
                var modelPath = Path.Combine(_modelDirectory, "songModel.zip");
                if (File.Exists(modelPath))
                {
                    _cachedSongModel = _mlContext.Model.Load(modelPath, out var modelInputSchema);
                }
            }
```

```csharp
            return _cachedSongModel;
        }

        1 reference | Anes Hrvačić, 269 days ago | 1 author, 1 change
        private ITransformer LoadArtistModel()
        {
            if (_cachedArtistModel == null)
            {
                var modelPath = Path.Combine(_modelDirectory, "artistModel.zip");
                if (File.Exists(modelPath))
                {
                    _cachedArtistModel = _mlContext.Model.Load(modelPath, out var modelInputSchema);
                }
            }
            return _cachedArtistModel;
        }

        1 reference | Anes Hrvačić, 269 days ago | 1 author, 1 change
        private ITransformer LoadAlbumModel()
        {
            if (_cachedAlbumModel == null)
            {
                var modelPath = Path.Combine(_modelDirectory, "albumModel.zip");
                if (File.Exists(modelPath))
                {
                    _cachedAlbumModel = _mlContext.Model.Load(modelPath, out var modelInputSchema);
                }
            }
            return _cachedAlbumModel;
        }

        3 references | Anes Hrvačić, 269 days ago | 1 author, 1 change
        private async Task<List<TResponse>> GetRecommendations<TEntity, TResponse>(int userId, int size, ITransformer model, string entityType)
        {
            var predictionEngine = _mlContext.Model.CreatePredictionEngine<RecommendationData, Prediction>(model);

            var allEntities = await GetAllEntities<TEntity, TResponse>(entityType);

            var predictions = new List<(TResponse entity, float score)>();

            foreach (var entity in allEntities)
            {
                var prediction = predictionEngine.Predict(new RecommendationData
                {
                    UserId = (uint)userId,
                    EntityId = GetEntityId(entity)
                });

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

            var topPredictions = predictions
                .OrderByDescending(p => p.score)
                .Take(size)
```



```csharp
        }

        1 reference | Anes Hrvačić, 269 days ago | 1 author, 1 change
        private async Task<List<TResponse>> GetAllEntities<TEntity, TResponse>(string entityType)
        {
            List<TEntity> entities = new List<TEntity>();

            switch (entityType)
            {
                case "song":
                    entities = await _context.Songs.Include(s => s.SongGenres)
                                                  .ThenInclude(sg => sg.Genre)
                                                .Include(s => s.SongArtists)
                                                  .ThenInclude(sa => sa.Artist)
                                                .Cast<TEntity>().ToListAsync();
                    break;
                case "artist":
                    entities = await _context.Artists.Include(a => a.ArtistGenres)
                                                    .ThenInclude(ag => ag.Genre)
                                                  .Cast<TEntity>().ToListAsync();
                    break;
                case "album":
                    entities = await _context.Albums.Include(a => a.AlbumGenres)
                                                   .ThenInclude(ag => ag.Genre)
                                                 .Include(a => a.AlbumArtists)
                                                   .ThenInclude(aa => aa.Artist)
                                                 .Cast<TEntity>().ToListAsync();
                    break;
                default:
                    throw new Exception("Invalid entity type");
            }

            return _mapper.Map<List<TResponse>>(entities);
        }

        1 reference | Anes Hrvačić, 269 days ago | 1 author, 1 change
        private uint GetEntityId<T>(T entity)
        {
            if (entity is SongResponse song) return (uint)song.Id;
            if (entity is ArtistResponse artist) return (uint)artist.Id;
            if (entity is AlbumResponse album) return (uint)album.Id;
            throw new Exception("Entity does not have a valid ID");
        }

        2 references | Anes Hrvačić, 269 days ago | 1 author, 1 change
        public async Task<bool> UserHasSongInteractions()
        {
            var userId = _authService.GetUserId();
            var userSongLikes = await _context.UserSongLikes.Where(usl => usl.UserId == userId).ToListAsync();
            var userSongViews = await _context.UserSongViews.Where(usv => usv.UserId == userId).ToListAsync();

            return userSongLikes.Any() || userSongViews.Any();
        }

        2 references | Anes Hrvačić, 269 days ago | 1 author, 1 change
        public async Task<bool> UserHasArtistInteractions()
        {
            var userId = _authService.GetUserId();
            var userArtistLikes = await _context.UserArtistLikes.Where(ual => ual.UserId == userId).ToListAsync();
            var userArtistViews = await _context.UserArtistViews.Where(uav => uav.UserId == userId).ToListAsync();

            return userArtistLikes.Any() || userArtistViews.Any();
        }

        2 references | Anes Hrvačić, 269 days ago | 1 author, 1 change
        public async Task<bool> UserHasAlbumInteractions()
        {
```

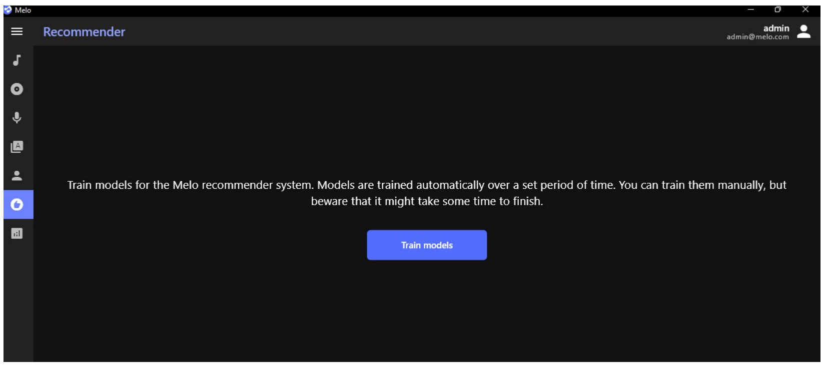


```
            if (entity is AlbumResponse album) return (uint)album.Id;
            throw new Exception("Entity does not have a valid ID");
        }

        2 references | Anes Hrvačić, 269 days ago | 1 author, 1 change
        public async Task<bool> UserHasSongInteractions()
        {
            var userId = _authService.GetUserId();
            var userSongLikes = await _context.UserSongLikes.Where(usl => usl.UserId == userId).ToListAsync();
            var userSongViews = await _context.UserSongViews.Where(usv => usv.UserId == userId).ToListAsync();

            return userSongLikes.Any() || userSongViews.Any();
        }

        2 references | Anes Hrvačić, 269 days ago | 1 author, 1 change
        public async Task<bool> UserHasArtistInteractions()
        {
            var userId = _authService.GetUserId();
            var userArtistLikes = await _context.UserArtistLikes.Where(ual => ual.UserId == userId).ToListAsync();
            var userArtistViews = await _context.UserArtistViews.Where(uav => uav.UserId == userId).ToListAsync();

            return userArtistLikes.Any() || userArtistViews.Any();
        }

        2 references | Anes Hrvačić, 269 days ago | 1 author, 1 change
        public async Task<bool> UserHasAlbumInteractions()
        {
            var userId = _authService.GetUserId();
            var userAlbumLikes = await _context.UserAlbumLikes.Where(ual => ual.UserId == userId).ToListAsync();
            var userAlbumViews = await _context.UserAlbumViews.Where(uav => uav.UserId == userId).ToListAsync();

            return userAlbumLikes.Any() || userAlbumViews.Any();
        }

        2 references | Anes Hrvačić, 269 days ago | 1 author, 1 change
        public async Task<List<SongResponse>> GetPopularSongs(int size)
        {
            var songs = await _context.Songs.Include(s => s.SongGenres)
                                              .ThenInclude(sg => sg.Genre)
                                            .Include(s => s.SongArtists)
                                              .ThenInclude(sa => sa.Artist)
                                            .OrderByDescending(s => s.LikeCount).Take(size).ToListAsync();
            return _mapper.Map<List<SongResponse>>(songs);
        }

        2 references | Anes Hrvačić, 269 days ago | 1 author, 1 change
        public async Task<List<ArtistResponse>> GetPopularArtists(int size)
        {
            var artists = await _context.Artists.Include(a => a.ArtistGenres)
                                                  .ThenInclude(ag => ag.Genre)
                                                .OrderByDescending(a => a.LikeCount).Take(size).ToListAsync();
            return _mapper.Map<List<ArtistResponse>>(artists);
        }

        public async Task<List<AlbumResponse>> GetPopularAlbums(int size)
        {
            var albums = await _context.Albums.Include(a => a.AlbumGenres)
                                                .ThenInclude(ag => ag.Genre)
                                              .Include(a => a.AlbumArtists)
                                                .ThenInclude(aa => aa.Artist)
                                              .OrderByDescending(a => a.LikeCount).Take(size).ToListAsync();
            return _mapper.Map<List<AlbumResponse>>(albums);
        }
    }
}
```

Na osnovu sačuvanih modela, korisnik će dobiti preporučeni sadržaj.

Modeli se mogu trenirati manuelno od strane administratora:

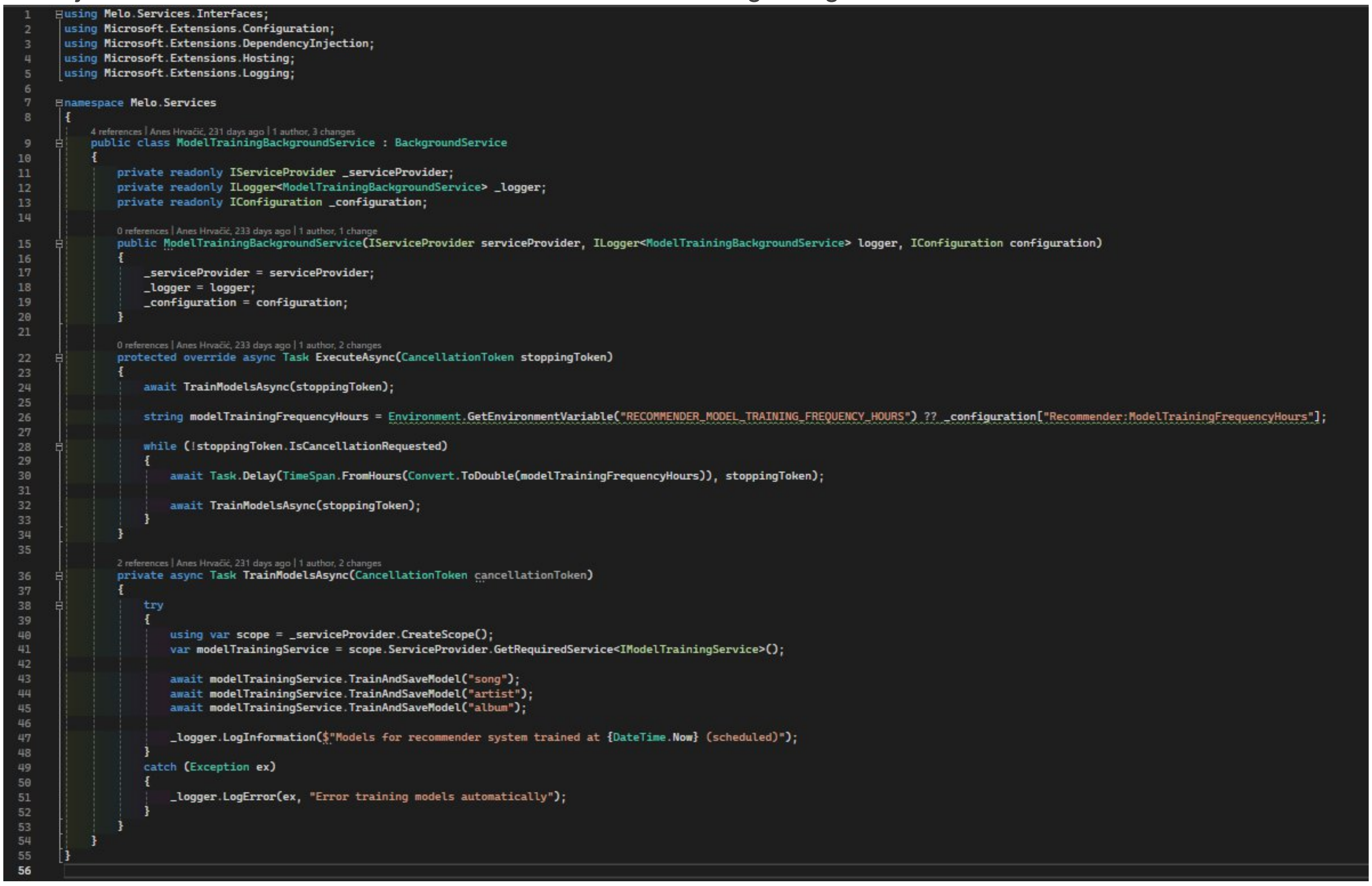
Putanja do koda: Melo\Melo.UI\melo_desktop\lib\pages\admin_recommender_page.dart



Train models for the Melo recommender system. Models are trained automatically over a set period of time. You can train them manually, but beware that it might take some time to finish.

Svakako, modeli se treniraju i automatski, pri čemu se logira vrijeme treniranja modela. Vrijeme manuelnog treniranja modela se također logira, sa jasnom naznakom da je u pitanju manuelno treniranje. Interval automatskog treniranja modela je konfigurabilan:

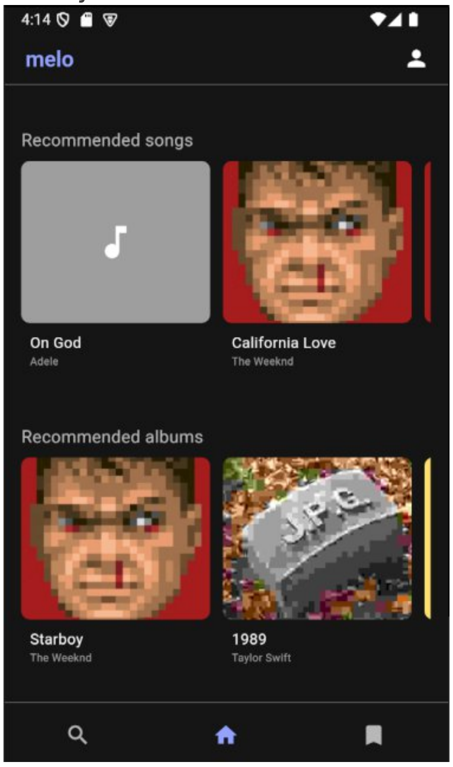
Putanja do koda: Melo\Melo.Services\Services\ModelTrainingBackgroundService.cs



```
using Melo.Services.Interfaces;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;
using Microsoft.Extensions.Logging;

namespace Melo.Services
{
    4 references | Anes Hrvačić, 231 days ago | 1 author, 3 changes
    public class ModelTrainingBackgroundService : BackgroundService
    {
        private readonly IServiceProvider _serviceProvider;
        private readonly ILogger<ModelTrainingBackgroundService> _logger;
        private readonly IConfiguration _configuration;

        0 references | Anes Hrvačić, 233 days ago | 1 author, 1 change
        public ModelTrainingBackgroundService(IServiceProvider serviceProvider, ILogger<ModelTrainingBackgroundService> logger, IConfiguration configuration)
        {
            _serviceProvider = serviceProvider;
            _logger = logger;
            _configuration = configuration;
        }

        0 references | Anes Hrvačić, 233 days ago | 1 author, 2 changes
        protected override async Task ExecuteAsync(CancellationToken stoppingToken)
        {
            await TrainModelsAsync(stoppingToken);

            string modelTrainingFrequencyHours = Environment.GetEnvironmentVariable("RECOMMENDER_MODEL_TRAINING_FREQUENCY_HOURS") ?? _configuration["Recommender:ModelTrainingFrequencyHours"];

            while (!stoppingToken.IsCancellationRequested)
            {
                await Task.Delay(TimeSpan.FromHours(Convert.ToDouble(modelTrainingFrequencyHours)), stoppingToken);

                await TrainModelsAsync(stoppingToken);
            }
        }

        2 references | Anes Hrvačić, 231 days ago | 1 author, 2 changes
        private async Task TrainModelsAsync(CancellationToken cancellationToken)
        {
            try
            {
                using var scope = _serviceProvider.CreateScope();
                var modelTrainingService = scope.ServiceProvider.GetRequiredService<IModelTrainingService>();

                await modelTrainingService.TrainAndSaveModel("song");
                await modelTrainingService.TrainAndSaveModel("artist");
                await modelTrainingService.TrainAndSaveModel("album");

                _logger.LogInformation($"Models for recommender system trained at {DateTime.Now} (scheduled)");
            }
            catch (Exception ex)
            {
                _logger.LogError(ex, "Error training models automatically");
            }
        }
    }
}
```

Kao što sam naveo na početku, preporučeni sadržaj se nalazi na početnoj stranici aplikacije, u vidu *„carousel“* listi:

Putanja do koda: Melo\Melo.UI\melo_mobile\lib\pages\home_page.dart



4:14

melo

Recommended songs

On God
Adele

California Love
The Weeknd

Recommended albums

Starboy
The Weeknd

1989
Taylor Swift