

Dokumentacija sistema preporuke

Korisnici Melo aplikacije imaju mogućnost vidjeti preporučeni 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

```

1  using Melo.Models;
2  using Melo.Services.Interfaces;
3  using Microsoft.EntityFrameworkCore;
4  using Microsoft.ML;
5
6  namespace Melo.Services
7  {
8      2 references | Anes Hrvačić, 269 days ago | 1 author, 1 change
9      public class ModelTrainingService : IModelTrainingService
10     {
11         private readonly ApplicationDbContext _context;
12         private readonly MLContext _mlContext;
13         private readonly string _modelDirectory = Path.Combine(Directory.GetCurrentDirectory(), "Recommendations", "Models");
14
15         0 references | Anes Hrvačić, 269 days ago | 1 author, 1 change
16         public ModelTrainingService(ApplicationDbContext context)
17         {
18             _context = context;
19             _mlContext = new MLContext();
20
21         7 references | Anes Hrvačić, 269 days ago | 1 author, 1 change
22         public async Task TrainAndSaveModel(string entityType)
23         {
24             IEnumerable<RecommendationData> interactions = [];
25
26             switch (entityType)
27             {
28                 case "song":
29                     interactions = await GetUserSongInteractions();
30                     break;
31                 case "album":
32                     interactions = await GetUserAlbumInteractions();
33                     break;
34                 case "artist":
35                     interactions = await GetUserArtistInteractions();
36                     break;
37                 default:
38                     throw new Exception("Invalid entity type");
39             }
40
41             var model = TrainModel(interactions);
42             SaveModel(model, $"{entityType}Model.zip");
43         }
44
45         1 reference | Anes Hrvačić, 269 days ago | 1 author, 1 change
46         private async Task<IEnumerable<RecommendationData>> GetUserSongInteractions()
47         {
48             var userSongInteractions = await _context.UserSongLikes.Select(ul => new RecommendationData
49             {
50                 UserId = (uint)ul.UserId,
51                 EntityId = (uint)ul.SongId,
52                 InteractionScore = 2
53             })
54             .Union(_context.UserSongViews.Select(uv => new RecommendationData
55             {
56                 UserId = (uint)uv.UserId,
57                 EntityId = (uint)uv.SongId,
58                 InteractionScore = (float)(0.1 * uv.Count)
59             })
60             ).ToListAsync();
61
62             return userSongInteractions;
63         }
64
65         1 reference | Anes Hrvačić, 269 days ago | 1 author, 1 change
66         private async Task<IEnumerable<RecommendationData>> GetUserArtistInteractions()

```

```

58     return userSongInteractions;
59 }
60
61 1 reference | Anes Hrvačić, 269 days ago | 1 author, 1 change
62 private async Task<IEnumerable<RecommendationData>> GetUserArtistInteractions()
63 {
64     var userArtistInteractions = await _context.UserArtistLikes.Select(ul => new RecommendationData
65     {
66         UserId = (uint)ul.UserId,
67         EntityId = (uint)ul.ArtistId,
68         InteractionScore = 2
69     })
70     .Union(_context.UserArtistViews.Select(uv => new RecommendationData
71     {
72         UserId = (uint)uv.UserId,
73         EntityId = (uint)uv.ArtistId,
74         InteractionScore = (float)(0.1 * uv.Count)
75     })
76     ).ToListAsync();
77     return userArtistInteractions;
78 }
79
80
81 private async Task<IEnumerable<RecommendationData>> GetUserAlbumInteractions()
82 {
83     var userAlbumInteractions = await _context.UserAlbumLikes.Select(ul => new RecommendationData
84     {
85         UserId = (uint)ul.UserId,
86         EntityId = (uint)ul.AlbumId,
87         InteractionScore = 2
88     })
89     .Union(_context.UserAlbumViews.Select(uv => new RecommendationData
90     {
91         UserId = (uint)uv.UserId,
92         EntityId = (uint)uv.AlbumId,
93         InteractionScore = (float)(0.1 * uv.Count)
94     })
95     ).ToListAsync();
96     return userAlbumInteractions;
97 }
98
99
100 private ITransformer TrainModel(IEnumerable<RecommendationData> interactions)
101 {
102     var dataView = _mlContext.Data.LoadFromEnumerable(interactions);
103     var pipeline = _mlContext.Recommendation().Trainers.MatrixFactorization(
104         labelColumnName: "InteractionScore",
105         matrixColumnIndexColumnName: "UserId",
106         matrixRowIndexColumnName: "EntityId",
107         numberOfIterations: 100,
108         learningRate: 0.2f
109     );
110     return pipeline.Fit(dataView);
111 }
112
113 private void SaveModel(ITransformer model, string modelName)
114 {
115     var modelPath = Path.Combine(_modelDirectory, modelName);
116     if (!Directory.Exists(_modelDirectory))
117     {
118         Directory.CreateDirectory(_modelDirectory);
119     }
120 }

```

```

80 namespace Melo.Services
81 private async Task<IEnumerable<RecommendationData>> GetUserAlbumInteractions()
82 {
83     var userAlbumInteractions = await _context.UserAlbumLikes.Select(ul => new RecommendationData
84     {
85         UserId = (uint)ul.UserId,
86         EntityId = (uint)ul.AlbumId,
87         InteractionScore = 2
88     })
89     .Union(_context.UserAlbumViews.Select(uv => new RecommendationData
90     {
91         UserId = (uint)uv.UserId,
92         EntityId = (uint)uv.AlbumId,
93         InteractionScore = (float)(0.1 * uv.Count)
94     })
95     ).ToListAsync();
96     return userAlbumInteractions;
97 }
98
99
100 1 reference | Anes Hrvačić, 269 days ago | 1 author, 1 change
101 private ITransformer TrainModel(IEnumerable<RecommendationData> interactions)
102 {
103     var dataView = _mlContext.Data.LoadFromEnumerable(interactions);
104     var pipeline = _mlContext.Recommendation().Trainers.MatrixFactorization(
105         labelColumnName: "InteractionScore",
106         matrixColumnIndexColumnName: "UserId",
107         matrixRowIndexColumnName: "EntityId",
108         numberOfIterations: 100,
109         learningRate: 0.2f
110     );
111     return pipeline.Fit(dataView);
112 }
113
114 1 reference | Anes Hrvačić, 269 days ago | 1 author, 1 change
115 private void SaveModel(ITransformer model, string modelName)
116 {
117     var modelPath = Path.Combine(_modelDirectory, modelName);
118     if (!Directory.Exists(_modelDirectory))
119     {
120         Directory.CreateDirectory(_modelDirectory);
121     }
122     _mlContext.Model.Save(model, null, modelPath);
123 }
124 }

```

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

```
1  using Melo.Models;
2  using Melo.Services.Interfaces;
3  using Microsoft.AspNetCore.Authorization;
4  using Microsoft.AspNetCore.Mvc;
5
6  namespace Melo.API.Controllers
7  {
8      3 references | Anes Hrvacic, 138 days ago | 1 author, 5 changes
9      public class RecommendationsController : CustomControllerBase
10     {
11         private readonly ILogger<RecommendationsController> _logger;
12         private readonly IRecommendationService _recommendationService;
13         private readonly IModelTrainingService _modelTrainingService;
14
15         0 references | Anes Hrvacic, 269 days ago | 1 author, 1 change
16         public RecommendationsController(ILogger<RecommendationsController> logger, IRecommendationService recommendationService, IModelTrainingService modelTrainingService)
17         {
18             _logger = logger;
19             _recommendationService = recommendationService;
20             _modelTrainingService = modelTrainingService;
21         }
22
23         [Authorize(Policy = "SubscribedUser")]
24         [HttpGet("Get-Recommendations")]
25         0 references | Anes Hrvacic, 164 days ago | 1 author, 2 changes
26         public async Task<ActionResult> GetRecommendations([FromQuery] int size = 20)
27         {
28             var userHasSongInteractions = await _recommendationService.UserHasSongInteractions();
29             var songRecommendations = !userHasSongInteractions ? await _recommendationService.GetPopularSongs(size) : await _recommendationService.GetSongRecommendations(size);
30
31             var userHasArtistInteractions = await _recommendationService.UserHasArtistInteractions();
32             var artistRecommendations = !userHasArtistInteractions ? await _recommendationService.GetPopularArtists(size) : await _recommendationService.GetArtistRecommendations(size);
33
34             var userHasAlbumInteractions = await _recommendationService.UserHasAlbumInteractions();
35             var albumRecommendations = !userHasAlbumInteractions ? await _recommendationService.GetPopularAlbums(size) : await _recommendationService.GetAlbumRecommendations(size);
36
37             return Ok(new
38             {
39                 Songs = songRecommendations,
40                 Artists = artistRecommendations,
41                 Albums = albumRecommendations
42             });
43         }
44
45         [Authorize(Policy = "Admin")]
46         [HttpPost("Train-Models")]
47         0 references | Anes Hrvacic, 138 days ago | 1 author, 4 changes
48         public async Task<ActionResult> TrainModels()
49         {
50             try
51             {
52                 await _modelTrainingService.TrainAndSaveModel("song");
53                 await _modelTrainingService.TrainAndSaveModel("artist");
54                 await _modelTrainingService.TrainAndSaveModel("album");
55                 _logger.LogInformation($"Models for recommender system trained at {DateTime.Now} (manual)");
56                 return Ok(new MessageResponse() { Success = true, Message = "Models trained and saved successfully" });
57             }
58             catch (Exception ex)
59             {
60                 _logger.LogError(ex, "Error training models manually");
61                 return StatusCode(500, new MessageResponse() { Success = false, Message = "Not enough data for model training" });
62             }
63         }
64     }
65 }
```

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

```

1  using AutoMapper;
2  using Melo.Models;
3  using Melo.Services.Entities;
4  using Melo.Services.Interfaces;
5  using Microsoft.EntityFrameworkCore;
6  using Microsoft.ML;
7
8  namespace Melo.Services
9  {
10     2 references | Anes Hrvacic, 269 days ago | 1 author, 1 change
11     public class RecommendationService : IRecommendationService
12     {
13         private readonly string _modelDirectory = Path.Combine(Directory.GetCurrentDirectory(), "Recommendations", "Models");
14
15         private readonly ApplicationDbContext _context;
16         private readonly MLContext _mlContext;
17         private readonly IAuthService _authService;
18         private readonly IMapper _mapper;
19
20         private ITransformer _cachedSongModel;
21         private ITransformer _cachedArtistModel;
22         private ITransformer _cachedAlbumModel;
23
24         0 references | Anes Hrvacic, 269 days ago | 1 author, 1 change
25         public RecommendationService(ApplicationDbContext context, IAuthService authService, IMapper mapper)
26         {
27             _context = context;
28             _mlContext = new MLContext();
29             _authService = authService;
30             _mapper = mapper;
31         }
32
33         2 references | Anes Hrvacic, 269 days ago | 1 author, 1 change
34         public async Task<List<SongResponse>> GetSongRecommendations(int size)
35         {
36             var userId = _authService.GetUserId();
37             var model = LoadSongModel();
38             return await GetRecommendations<Song, SongResponse>(userId, size, model, "song");
39         }
40
41         2 references | Anes Hrvacic, 269 days ago | 1 author, 1 change
42         public async Task<List<ArtistResponse>> GetArtistRecommendations(int size)
43         {
44             var userId = _authService.GetUserId();
45             var model = LoadArtistModel();
46             return await GetRecommendations<Artist, ArtistResponse>(userId, size, model, "artist");
47         }
48
49         2 references | Anes Hrvacic, 269 days ago | 1 author, 1 change
50         public async Task<List<AlbumResponse>> GetAlbumRecommendations(int size)
51         {
52             var userId = _authService.GetUserId();
53             var model = LoadAlbumModel();
54             return await GetRecommendations<Album, AlbumResponse>(userId, size, model, "album");
55         }
56
57         1 reference | Anes Hrvacic, 269 days ago | 1 author, 1 change
58         private ITransformer LoadSongModel()
59         {
60             if (_cachedSongModel == null)
61             {
62                 var modelPath = Path.Combine(_modelDirectory, "songModel.zip");
63                 if (File.Exists(modelPath))
64                 {
65                     _cachedSongModel = _mlContext.Model.Load(modelPath, out var modelInputSchema);
66                 }
67             }
68             return _cachedSongModel;
69         }
70
71         1 reference | Anes Hrvacic, 269 days ago | 1 author, 1 change
72         private ITransformer LoadArtistModel()
73         {
74             if (_cachedArtistModel == null)
75             {
76                 var modelPath = Path.Combine(_modelDirectory, "artistModel.zip");
77                 if (File.Exists(modelPath))
78                 {
79                     _cachedArtistModel = _mlContext.Model.Load(modelPath, out var modelInputSchema);
80                 }
81             }
82             return _cachedArtistModel;
83         }
84
85         1 reference | Anes Hrvacic, 269 days ago | 1 author, 1 change
86         private ITransformer LoadAlbumModel()
87         {
88             if (_cachedAlbumModel == null)
89             {
90                 var modelPath = Path.Combine(_modelDirectory, "albumModel.zip");
91                 if (File.Exists(modelPath))
92                 {
93                     _cachedAlbumModel = _mlContext.Model.Load(modelPath, out var modelInputSchema);
94                 }
95             }
96             return _cachedAlbumModel;
97         }
98
99         3 references | Anes Hrvacic, 269 days ago | 1 author, 1 change
100        private async Task<List<TResponse>> GetRecommendations<TEntity, TResponse>(int userId, int size, ITransformer model, string entityType)
101        {
102            var predictionEngine = _mlContext.Model.CreatePredictionEngine<RecommendationData, Prediction>(model);
103            var allEntities = await GetAllEntities<TEntity, TResponse>(entityType);
104            var predictions = new List<(TResponse entity, float score)>();
105            foreach (var entity in allEntities)
106            {
107                var prediction = predictionEngine.Predict(new RecommendationData
108                {
109                    UserId = (uint)userId,
110                    EntityId = GetEntityId(entity)
111                });
112                predictions.Add((entity, prediction.Score));
113            }
114            var topPredictions = predictions
115                .OrderByDescending(p => p.score)
116                .Take(size)
117                .Select(p => p.entity)
118                .ToList();
119            return topPredictions.Select(entity => entity.ToResponse()).ToList();
120        }
121    }
122 }

```

```

117     }
118
119     1 reference | Anes Hrvaić, 269 days ago | 1 author, 1 change
120     private async Task<List<TResponse>> GetAllEntities<TEntity, TResponse>(string entityType)
121     {
122         List<TEntity> entities = new List<TEntity>();
123
124         switch (entityType)
125         {
126             case "song":
127                 entities = await _context.Songs.Include(s => s.SongGenres)
128                     .ThenInclude(sg => sg.Genre)
129                     .Include(s => s.SongArtists)
130                     .ThenInclude(sa => sa.Artist)
131                     .Cast<TEntity>().ToListAsync();
132                 break;
133             case "artist":
134                 entities = await _context.Artists.Include(a => a.ArtistGenres)
135                     .ThenInclude(ag => ag.Genre)
136                     .Cast<TEntity>().ToListAsync();
137                 break;
138             case "album":
139                 entities = await _context.Albums.Include(a => a.AlbumGenres)
140                     .ThenInclude(ag => ag.Genre)
141                     .Include(a => a.AlbumArtists)
142                     .ThenInclude(aa => aa.Artist)
143                     .Cast<TEntity>().ToListAsync();
144                 break;
145             default:
146                 throw new Exception("Invalid entity type");
147         }
148
149         return _mapper.Map<List<TResponse>>(entities);
150     }
151
152     1 reference | Anes Hrvaić, 269 days ago | 1 author, 1 change
153     private uint GetEntityId<T>(T entity)
154     {
155         if (entity is SongResponse song) return (uint)song.Id;
156         if (entity is ArtistResponse artist) return (uint)artist.Id;
157         if (entity is AlbumResponse album) return (uint)album.Id;
158         throw new Exception("Entity does not have a valid ID");
159     }
160
161     2 references | Anes Hrvaić, 269 days ago | 1 author, 1 change
162     public async Task<bool> UserHasSongInteractions()
163     {
164         var userId = _authService.GetUserId();
165         var userSongLikes = await _context.UserSongLikes.Where(usl => usl.UserId == userId).ToListAsync();
166         var userSongViews = await _context.UserSongViews.Where(usv => usv.UserId == userId).ToListAsync();
167
168         return userSongLikes.Any() || userSongViews.Any();
169     }
170
171     2 references | Anes Hrvaić, 269 days ago | 1 author, 1 change
172     public async Task<bool> UserHasArtistInteractions()
173     {
174         var userId = _authService.GetUserId();
175         var userArtistLikes = await _context.UserArtistLikes.Where(ual => ual.UserId == userId).ToListAsync();
176         var userArtistViews = await _context.UserArtistViews.Where(uav => uav.UserId == userId).ToListAsync();
177
178         return userArtistLikes.Any() || userArtistViews.Any();
179     }
180
181     2 references | Anes Hrvaić, 269 days ago | 1 author, 1 change
182     public async Task<bool> UserHasAlbumInteractions()
183     {
184         var userId = _authService.GetUserId();
185         var userAlbumLikes = await _context.UserAlbumLikes.Where(ual => ual.UserId == userId).ToListAsync();
186         var userAlbumViews = await _context.UserAlbumViews.Where(uav => uav.UserId == userId).ToListAsync();
187
188         return userAlbumLikes.Any() || userAlbumViews.Any();
189     }

```

```

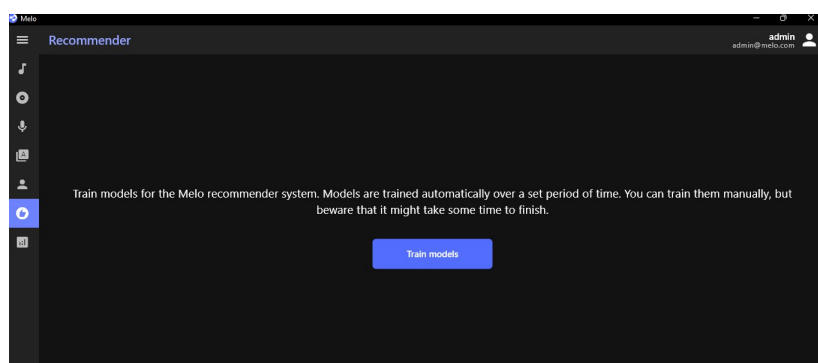
155     if (entity is AlbumResponse album) return (uint)album.Id;
156     throw new Exception("Entity does not have a valid ID");
157 }
158
159 public async Task<bool> UserHasSongInteractions()
160 {
161     var userId = _authService.GetUserId();
162     var userSongLikes = await _context.UserSongLikes.Where(ysl => ysl.UserId == userId).ToListAsync();
163     var userSongViews = await _context.UserSongViews.Where(ysv => ysv.UserId == userId).ToListAsync();
164
165     return userSongLikes.Any() || userSongViews.Any();
166 }
167
168 public async Task<bool> UserHasArtistInteractions()
169 {
170     var userId = _authService.GetUserId();
171     var userArtistLikes = await _context.UserArtistLikes.Where(ual => ual.UserId == userId).ToListAsync();
172     var userArtistViews = await _context.UserArtistViews.Where(uav => uav.UserId == userId).ToListAsync();
173
174     return userArtistLikes.Any() || userArtistViews.Any();
175 }
176
177 public async Task<bool> UserHasAlbumInteractions()
178 {
179     var userId = _authService.GetUserId();
180     var userAlbumLikes = await _context.UserAlbumLikes.Where(ual => ual.UserId == userId).ToListAsync();
181     var userAlbumViews = await _context.UserAlbumViews.Where(uav => uav.UserId == userId).ToListAsync();
182
183     return userAlbumLikes.Any() || userAlbumViews.Any();
184 }
185
186 public async Task<List<SongResponse>> GetPopularSongs(int size)
187 {
188     var songs = await _context.Songs.Include(s => s.SongGenres)
189                                     .ThenInclude(sg => sg.Genre)
190                                     .Include(s => s.SongArtists)
191                                     .ThenInclude(sa => sa.Artist)
192                                     .OrderByDescending(s => s.LikeCount).Take(size).ToListAsync();
193     return _mapper.Map<List<SongResponse>>(songs);
194 }
195
196 public async Task<List<ArtistResponse>> GetPopularArtists(int size)
197 {
198     var artists = await _context.Artists.Include(a => a.ArtistGenres)
199                                         .ThenInclude(ag => ag.Genre)
200                                         .OrderByDescending(a => a.LikeCount).Take(size).ToListAsync();
201     return _mapper.Map<List<ArtistResponse>>(artists);
202 }
203
204 public async Task<List<AlbumResponse>> GetPopularAlbums(int size)
205 {
206     var albums = await _context.Albums.Include(a => a.AlbumGenres)
207                                         .ThenInclude(ag => ag.Genre)
208                                         .Include(a => a.AlbumArtists)
209                                         .ThenInclude(aa => aa.Artist)
210                                         .OrderByDescending(a => a.LikeCount).Take(size).ToListAsync();
211     return _mapper.Map<List<AlbumResponse>>(albums);
212 }
213 }
214 }

```

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



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

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

```
1 using Melo.Services.Interfaces;
2 using Microsoft.Extensions.Configuration;
3 using Microsoft.Extensions.DependencyInjection;
4 using Microsoft.Extensions.Hosting;
5 using Microsoft.Extensions.Logging;
6
7 namespace Melo.Services
8 {
9     4 references | Anes Hrvacic, 231 days ago | 1 author, 3 changes
10     public class ModelTrainingBackgroundService : BackgroundService
11     {
12         private readonly IServiceProvider _serviceProvider;
13         private readonly ILogger<ModelTrainingBackgroundService> _logger;
14         private readonly IConfiguration _configuration;
15
16         0 references | Anes Hrvacic, 233 days ago | 1 author, 1 change
17         public ModelTrainingBackgroundService(IServiceProvider serviceProvider, ILogger<ModelTrainingBackgroundService> logger, IConfiguration configuration)
18         {
19             _serviceProvider = serviceProvider;
20             _logger = logger;
21             _configuration = configuration;
22         }
23
24         0 references | Anes Hrvacic, 233 days ago | 1 author, 2 changes
25         protected override async Task ExecuteAsync(CancellationToken stoppingToken)
26         {
27             await TrainModelsAsync(stoppingToken);
28
29             string modelTrainingFrequencyHours = Environment.GetEnvironmentVariable("RECOMMENDER_MODEL_TRAINING_FREQUENCY_HOURS") ?? _configuration["Recommender:ModelTrainingFrequencyHours"];
30
31             while (!stoppingToken.IsCancellationRequested)
32             {
33                 await Task.Delay(TimeSpan.FromHours(Convert.ToDouble(modelTrainingFrequencyHours)), stoppingToken);
34                 await TrainModelsAsync(stoppingToken);
35             }
36         }
37
38         2 references | Anes Hrvacic, 231 days ago | 1 author, 2 changes
39         private async Task TrainModelsAsync(CancellationToken cancellationToken)
40         {
41             try
42             {
43                 using var scope = _serviceProvider.CreateScope();
44                 var modelTrainingService = scope.ServiceProvider.GetRequiredService<IModelTrainingService>();
45                 await modelTrainingService.TrainAndSaveModel("song");
46                 await modelTrainingService.TrainAndSaveModel("artist");
47                 await modelTrainingService.TrainAndSaveModel("album");
48
49                 _logger.LogInformation($"Models for recommender system trained at {DateTime.Now} (scheduled)");
50             }
51             catch (Exception ex)
52             {
53                 _logger.LogError(ex, "Error training models automatically");
54             }
55         }
56     }
57 }
```

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

