

```

1 // WardrobeOS.Data/AppDbContext.cs
2 using System;
3 using Microsoft.EntityFrameworkCore;
4 using System.ComponentModel.DataAnnotations;
5 using System.ComponentModel.DataAnnotations.Schema;
6
7 namespace WardrobeOS.Data;
8
9 public sealed class AppDbContext : DbContext
10 {
11     //--- DbSets (Этап 1)
12     public DbSet<User> Users => Set<User>();
13     public DbSet<Avatar> Avatars => Set<Avatar>();
14     public DbSet<AvatarCapture> AvatarCaptures => Set<AvatarCapture>();
15
16     public DbSet<Brand> Brands => Set<Brand>();
17     public DbSet<Category> Categories => Set<Category>();
18     public DbSet<Condition> Conditions => Set<Condition>();
19     public DbSet<Fit> Fits => Set<Fit>();
20     public DbSet<Color> Colors => Set<Color>();
21     public DbSet<Pattern> Patterns => Set<Pattern>();
22     public DbSet<Material> Materials => Set<Material>();
23     public DbSet<Season> Seasons => Set<Season>();
24     public DbSet<Currency> Currencies => Set<Currency>();
25
26     public DbSet<Wardrobe> Wardrobes => Set<Wardrobe>();
27     public DbSet<Garment> Garments => Set<Garment>();
28     public DbSet<GarmentImage> GarmentImages => Set<GarmentImage>();
29     public DbSet<GarmentColor> GarmentColors => Set<GarmentColor>();
30     public DbSet<GarmentPattern> GarmentPatterns => Set<GarmentPattern>();
31     public DbSet<GarmentMaterial> GarmentMaterials => Set<GarmentMaterial>();
32     public DbSet<GarmentSeason> GarmentSeasons => Set<GarmentSeason>();
33     public DbSet<GarmentTag> GarmentTags => Set<GarmentTag>();
34     public DbSet<GarmentCapture> GarmentCaptures => Set<GarmentCapture>();
35     public DbSet<GarmentVector> GarmentVectors => Set<GarmentVector>();
36     public DbSet<MediaIngestJob> MediaIngestJobs => Set<MediaIngestJob>();
37
38     //--- DbSets (Этап 2)
39     public DbSet<Outfit> Outfits => Set<Outfit>();
40     public DbSet<OutfitItem> OutfitItems => Set<OutfitItem>();
41     public DbSet<Follow> Follows => Set<Follow>();
42     public DbSet<UserOutfitReaction> UserOutfitReactions => Set<UserOutfitReaction>();
43
44     public DbSet<SearchIndexState> SearchIndexStates => Set<SearchIndexState>();
45     public DbSet<ContentLabel> ContentLabels => Set<ContentLabel>();
46     public DbSet<AuditLog> AuditLogs => Set<AuditLog>();
47
48     //--- DbSets (Этап 3)
49     public DbSet<Listing> Listings => Set<Listing>();
50     public DbSet<Interest> Interests => Set<Interest>();
51     public DbSet<InterestMessage> InterestMessages => Set<InterestMessage>();
52     public DbSet<SwapItem> SwapItems => Set<SwapItem>();
53     public DbSet<PeerReview> PeerReviews => Set<PeerReview>();
54
55     //----- USERS
56     protected override void OnModelCreating(ModelBuilder b)
57     {
58         //----- USERS
59         b.Entity<User>(e =>
60             {
61                 e.ToTable("users");
62                 e.HasKey(x => x.Id).IsClustered(false);
63                 e.Property(x => x.Id).HasDefaultValueSql("NEWSEQUENTIALID()");
64                 e.Property(x => x.Email).HasMaxLength(256).IsRequired();
65                 e.Property(x => x.EmailNorm)
66                     .HasComputedColumnSql("LOWER([email])", stored: true);
67                 e.HasIndex(x => x.EmailNorm).IsUnique().HasDatabaseName("UX_users_email_norm");
68                 e.Property(x => x.Name).HasMaxLength(128);
69                 e.Property(x => x.Role).HasMaxLength(20).HasDefaultValue("user");
70                 e.Property(x => x.Status).HasMaxLength(20).HasDefaultValue("active");
71                 e.Property(x => x.CreatedAt).HasDefaultValueSql("sysutcdatetime()");
72                 e.Property(x => x.RowVersion).IsRowVersion();
73                 e.HasIndex(x => x.CreatedAt).IsClustered().HasDatabaseName("CI_users_created")
74                     .IsDescending(true);
75             });
76
77
78         //----- AVATAR
79         b.Entity<Avatar>(e =>
80             {
81                 e.ToTable("avatars");
82                 e.HasKey(x => x.UserId);
83                 e.HasOne(x => x.User).WithOne(x => x.Avatar).HasForeignKey<Avatar>(x => x.UserId).OnDelete(DeleteBehavior.NoAction);
84             });
85                 e.Property(x => x.Sex).HasMaxLength(12);
86                 e.Property(x => x.SkinTone).HasMaxLength(24);
87                 e.Property(x => x.HairColor).HasMaxLength(24);
88                 e.Property(x => x.BodyType).HasMaxLength(24);
89                 e.Property(x => x.PxPerMm).HasPrecision(9, 6);
90                 e.Property(x => x.UpdatedAt).HasDefaultValueSql("sysutcdatetime()");
91
92         b.Entity<AvatarCapture>(e =>
93             {
94                 e.ToTable("avatar_captures");
95                 e.HasKey(x => x.Id);
96                 e.Property(x => x.Id).HasDefaultValueSql("NEWSEQUENTIALID()");
97                 e.Property(x => x.View).HasMaxLength(16).IsRequired();
98                 e.Property(x => x.Calibrator).HasMaxLength(16);
99                 e.Property(x => x.PxPerMm).HasPrecision(9, 6);
100                e.Property(x => x.Device).HasMaxLength(64);
101                e.Property(x => x.CreatedAt).HasDefaultValueSql("sysutcdatetime()");
102                e.HasOne<User>().WithMany().HasForeignKey(x => x.UserId).OnDelete(DeleteBehavior.NoAction);
103                e.HasIndex(x => new { x.UserId, x.CreatedAt }).HasDatabaseName("IX_avatar_captures_user").IsDescending(true);
104

```

```

105    →→→→} );
106
107
108 →→→→// ----- LOOKUPS
109 →→→→// Somnitelno, mozno i potom
110 →→→→b.Entity<Brand>(e =>
111 →→→→{
112 →→→→→e.ToTable("brands");
113 →→→→→e.HasKey(x => x.Id);
114 →→→→→e.Property(x => x.Name).HasMaxLength(128).IsRequired();
115 →→→→→e.HasIndex(x => x.Name).IsUnique();
116 →→→→);
117
118 →→→→b.Entity<Category>(e =>
119 →→→→{
120 →→→→→e.ToTable("categories");
121 →→→→→e.HasKey(x => x.Id);
122 →→→→→e.Property(x => x.Code).HasMaxLength(32).IsRequired();
123 →→→→→e.Property(x => x.Name).HasMaxLength(64).IsRequired();
124 →→→→→e.HasIndex(x => x.Code).IsUnique();
125 →→→→);
126
127 →→→→b.Entity<Condition>(e =>
128 →→→→{
129 →→→→→e.ToTable("conditions");
130 →→→→→e.HasKey(x => x.Id);
131 →→→→→e.Property(x => x.Code).HasMaxLength(16).IsRequired();
132 →→→→→e.Property(x => x.Name).HasMaxLength(64).IsRequired();
133 →→→→→e.HasIndex(x => x.Code).IsUnique();
134 →→→→);
135
136 →→→→b.Entity<Fit>(e =>
137 →→→→{
138 →→→→→e.ToTable("fits");
139 →→→→→e.HasKey(x => x.Id);
140 →→→→→e.Property(x => x.Code).HasMaxLength(16).IsRequired();
141 →→→→→e.Property(x => x.Name).HasMaxLength(64).IsRequired();
142 →→→→→e.HasIndex(x => x.Code).IsUnique();
143 →→→→);
144
145 →→→→b.Entity<Color>(e =>
146 →→→→{
147 →→→→→e.ToTable("colors", ·tb =>
148 →→→→→{
149 →→→→→→tb.HasCheckConstraint("CK_colors_hex", ·"hex LIKE
150 →→→→→→→tb.HasCheckConstraint("CK_colors_hex", ·"# [0-9A-Fa-f] [0-9A-Fa-f] [0-9A-Fa-f] [0-9A-Fa-f] [0-9A-Fa-f] [0-9A-Fa-f] [0-9A-Fa-f] [0-9A-Fa-f] ");
151 →→→→→→→tb.HasCheckConstraint("CK_colors_lab_l", ·"[l] BETWEEN 0 AND 100");
152 →→→→→→→tb.HasCheckConstraint("CK_colors_lab_a", ·"[a] BETWEEN -128 AND 127");
153 →→→→→→→tb.HasCheckConstraint("CK_colors_lab_b", ·"[b] BETWEEN -128 AND 127");
154 →→→→→→);
155 →→→→→→→e.HasKey(x => x.Id);
156 →→→→→→→e.Property(x => x.Name).HasMaxLength(32).IsRequired();
157 →→→→→→→e.HasIndex(x => x.Name).IsUnique();
158 →→→→→→→e.Property(x => x.Hex).HasColumnName("hex").HasMaxLength(7).IsRequired();
159 →→→→→→→e.HasIndex(x => x.Hex).IsUnique();
160 →→→→→→);
161 →→→→→→b.Entity<Pattern>(e =>
162 →→→→→→{
163 →→→→→→→e.ToTable("patterns");
164 →→→→→→→e.HasKey(x => x.Id);
165 →→→→→→→e.Property(x => x.Name).HasMaxLength(32).IsRequired();
166 →→→→→→→e.HasIndex(x => x.Name).IsUnique();
167 →→→→→→);
168
169 →→→→b.Entity<Material>(e =>
170 →→→→{
171 →→→→→e.ToTable("materials");
172 →→→→→e.HasKey(x => x.Id);
173 →→→→→e.Property(x => x.Name).HasMaxLength(32).IsRequired();
174 →→→→→e.HasIndex(x => x.Name).IsUnique();
175 →→→→);
176
177 →→→→b.Entity<Season>(e =>
178 →→→→{
179 →→→→→e.ToTable("seasons");
180 →→→→→e.HasKey(x => x.Code);
181 →→→→→e.Property(x => x.Code).HasMaxLength(16);
182 →→→→→e.Property(x => x.Name).HasMaxLength(32).IsRequired();
183 →→→→);
184
185 →→→→b.Entity<Currency>(e =>
186 →→→→{
187 →→→→→e.ToTable("currencies");
188 →→→→→e.HasKey(x => x.Code);
189 →→→→→e.Property(x => x.Code).HasMaxLength(3);
190 →→→→→e.Property(x => x.Name).HasMaxLength(64).IsRequired();
191 →→→→);
192
193
194 →→→→// ----- WARDROBES
195 →→→→b.Entity<Wardrobe>(e =>
196 →→→→{
197 →→→→→e.ToTable("wardrobes");
198 →→→→→e.HasKey(x => x.Id);
199 →→→→→e.Property(x => x.Id).HasDefaultValueSql("NEWSEQUENTIALID()");
200 →→→→→e.HasOne(x => x.Owner).WithMany(x => x.Wardrobes).HasForeignKey(x => x.OwnerId).OnDelete(DeleteBehavior.NoAction);
201 →→→→→e.Property(x => x.Title).HasMaxLength(120).IsRequired();
202 →→→→→e.Property(x => x.Description).HasMaxLength(500);
203 →→→→→e.Property(x => x.Visibility).HasMaxLength(7).HasDefaultValue("private");
204 →→→→→e.Property(x => x.IsDefault).HasDefaultValue(false);
205 →→→→→e.Property(x => x.CreatedAt).HasDefaultValueSql("sysutcdatetime()");
206 →→→→→e.HasIndex(x => new { x.OwnerId, x.CreatedAt }).IsClustered().IsDescending(true).HasDatabaseName(
207 →→→→→→→"CI_wardrobes_owner_created");
→→→→→→→e.HasIndex(x => x.OwnerId).HasFilter("[is_default] = 1").IsUnique().HasDatabaseName("UX_wardrobes_default");

```

```

208 →→→→});  

209  

210  

211 →→→→// ----- GARMENTS  

212 →→b.Entity<Garment>(e =>  

213 →→{  

214 →→→→e.ToTable("garments", tb =>  

215 →→→→→→tb.HasCheckConstraint("CK_garments_sale_price",  

216 →→→→→→→→"([is_for_sale] = 0 AND [price_minor] IS NULL) OR ([is_for_sale] = 1 AND [price_minor] IS NOT NULL)");  

217 →→→→→→});  

218 →→→→e.HasKey(x => x.Id).IsClustered(false);  

219 →→→→e.Property(x => x.Id).HasDefaultValueSql("NEWSEQUENTIALID()");  

220 →→→→e.HasOne(x => x.Owner).WithMany(x => x.Garments).HasForeignKey(x => x.OwnerId).OnDelete(DeleteBehavior.NoAction);  

221 →→→→e.HasOne(x => x.Wardrobe).WithMany(x => x.Garments).HasForeignKey(x => x.WardrobeId).OnDelete(DeleteBehavior.  

222 →→→→→→NoAction);  

223 →→→→e.Property(x => x.Title).HasMaxLength(160).IsRequired();  

224 →→→→e.Property(x => x.Size).HasMaxLength(32);  

225 →→→→e.Property(x => x.Visibility).HasMaxLength(7).HasDefaultValue("private");  

226 →→→→e.Property(x => x.IsForSale).HasDefaultValue(false);  

227 →→→→e.Property(x => x.CurrencyCode).HasMaxLength(3).HasDefaultValue("EUR");  

228 →→→→e.Property(x => x.PxPerMm).HasPrecision(9, 6);  

229 →→→→e.Property(x => x.PhashPrefix)  

230 →→→→→→.HasComputedColumnSql("CONVERT(binary(4), SUBSTRING([phash], (1), (4)))", stored:true);  

231 →→→→e.Property(x => x.isHidden).HasDefaultValue(false);  

232 →→→→e.Property(x => x.ModerationStatus).HasMaxLength(16).HasDefaultValue("visible");  

233 →→→→e.Property(x => x.CreatedAt).HasDefaultValueSql("sysutcdatetime()");  

234 →→→→e.Property(x => x.RowVersion).IsRowVersion();  

235  

236 →→→→e.HasIndex(x => new { x.OwnerId, x.CreatedAt }).IsClustered().IsDescending(true)  

237 →→→→→→.HasDatabaseName("CI_garments_owner_created");  

238 →→→→e.HasIndex(x => new { x.WardrobeId, x.CreatedAt }).HasDatabaseName("IX_garments_wardrobe").IsDescending(true);  

239 →→→→e.HasIndex(x => x.CategoryId).HasDatabaseName("IX_garments_category");  

240 →→→→e.HasIndex(x => x.BrandId).HasDatabaseName("IX_garments_brand");  

241 →→→→e.HasIndex(x => x.ConditionId).HasDatabaseName("IX_garments_condition");  

242 →→→→e.HasIndex(x => x.Phash).HasDatabaseName("IX_garments_phash");  

243 →→→→e.HasIndex(x => x.Visibility)  

244 →→→→→→.HasFilter("[visibility] = 'public'")  

245 →→→→→→.IncludeProperties(x => new { x.OwnerId, x.WardrobeId, x.CategoryId })  

246 →→→→→→.HasDatabaseName("IX_garments_public");  

247 →→→→e.HasIndex(x => new { x.IsForSale, x.CreatedAt })  

248 →→→→→→.HasFilter("[is_for_sale] = 1")  

249 →→→→→→.IncludeProperties(x => new { x.PriceMinor, x.CurrencyCode, x.OwnerId, x.CategoryId })  

250 →→→→→→.HasDatabaseName("IX_garments_sale");  

251  

252 →→→→e.HasIndex(x => new { x.OwnerId, x.Phash })  

253 →→→→→→.HasFilter("[phash] IS NOT NULL")  

254 →→→→→→.IsUnique()  

255 →→→→→→.HasDatabaseName("UX_garments_owner_phash");  

256 →→});  

257  

258 →→→→b.Entity<GarmentImage>(e =>  

259 →→→→{  

260 →→→→→→e.ToTable("garment_images");  

261 →→→→→→e.HasKey(x => x.Id);  

262 →→→→→→e.Property(x => x.Id).HasDefaultValueSql("NEWSEQUENTIALID()");  

263 →→→→→→e.Property(x => x.Url).HasMaxLength(512).IsRequired();  

264 →→→→→→e.Property(x => x.IsPrimary).HasDefaultValue(false);  

265 →→→→→→e.Property(x => x.CreatedAt).HasDefaultValueSql("sysutcdatetime()");  

266 →→→→→→e.HasOne(x => x.Garment).WithMany(x => x.Images).HasForeignKey(x => x.GarmentId).OnDelete(DeleteBehavior.Cascade);  

267 →→→→→→e.HasIndex(x => x.GarmentId).HasFilter("[is_primary] = 1").IsUnique().HasDatabaseName("UX_garment_images_primary");  

268 →→→→→→// (правило "нельзя оставаться без primary" реализуется триггером/доменной логикой)  

269 →→});  

270  

271 →→→→b.Entity<GarmentColor>(e =>  

272 →→→→{  

273 →→→→→→e.ToTable("garment_colors");  

274 →→→→→→e.HasKey(x => new { x.GarmentId, x.ColorId });  

275 →→→→→→e.Property(x => x.IsPrimary).HasDefaultValue(false);  

276 →→→→→→e.HasIndex(x => new { x.ColorId, x.GarmentId }).HasDatabaseName("IX_color_to_garments");  

277 →→});  

278  

279 →→→→b.Entity<GarmentPattern>(e =>  

280 →→→→{  

281 →→→→→→e.ToTable("garment_patterns");  

282 →→→→→→e.HasKey(x => new { x.GarmentId, x.PatternId });  

283 →→→→→→e.HasIndex(x => new { x.PatternId, x.GarmentId }).HasDatabaseName("IX_pattern_to_garments");  

284 →→});  

285  

286 →→→→b.Entity<GarmentMaterial>(e =>  

287 →→→→{  

288 →→→→→→e.ToTable("garment_materials");  

289 →→→→→→e.HasKey(x => new { x.GarmentId, x.MaterialId });  

290 →→→→→→e.HasIndex(x => new { x.MaterialId, x.GarmentId }).HasDatabaseName("IX_material_to_garments");  

291 →→});  

292  

293 →→→→b.Entity<GarmentSeason>(e =>  

294 →→→→{  

295 →→→→→→e.ToTable("garment_seasons");  

296 →→→→→→e.HasKey(x => new { x.GarmentId, x.SeasonCode });  

297 →→→→→→e.HasIndex(x => new { x.SeasonCode, x.GarmentId }).HasDatabaseName("IX_season_to_garments");  

298 →→→→→→});  

299 →→});  

300  

301 →→→→b.Entity<GarmentTag>(e =>  

302 →→→→{  

303 →→→→→→e.ToTable("garment_tags");  

304 →→→→→→e.HasKey(x => new { x.GarmentId, x.Tag });  

305 →→→→→→e.Property(x => x.Tag).HasMaxLength(64).IsRequired();  

306 →→→→→→e.HasIndex(x => new { x.Tag, x.GarmentId }).HasDatabaseName("IX_tag_to_garments");  

307 →→});  

308  

309 →→→→b.Entity<GarmentCapture>(e =>  

310 →→→→{  

311 →→→→→→e.ToTable("garment_captures");

```

```

312     →→→→e.HasKey(x => x.Id);
313     →→→→e.Property(x => x.Id).HasDefaultValueSql("NEWSEQUENTIALID()");
314     →→→→e.Property(x => x.PxPerMm).HasPrecision(9,6);
315     →→→→e.Property(x => x.Calibrator).HasMaxLength(16);
316     →→→→e.Property(x => x.Device).HasMaxLength(64);
317     →→→→e.Property(x => x.Notes).HasMaxLength(256);
318     →→→→e.Property(x => x.CreatedAt).HasDefaultValueSql("sysutcdatetime()");
319   →→→} );
320
321   →→b.Entity<GarmentVector>(e =>
322   →→{
323     →→→e.ToTable("garment_vectors", tb =>
324     →→→{
325       →→→tb.HasCheckConstraint("CK_garment_vectors_len", "DATALENGTH([vector]) = CONVERT(int, [dims]) * 4");
326     } );
327     →→→e.HasKey(x => x.GarmentId);
328     →→→e.Property(x => x.Model).HasMaxLength(64).IsRequired();
329     →→→e.Property(x => x.Dims).HasColumnType("smallint");
330     →→→e.Property(x => x.IsL2Normalized).HasDefaultValue(true);
331     →→→e.Property(x => x.CreatedAt).HasDefaultValueSql("sysutcdatetime()");
332   →→→} );
333
334   →→b.Entity<MediaIngestJob>(e =>
335   →→{
336     →→→e.ToTable("media_ingest_jobs");
337     →→→e.HasKey(x => x.Id);
338     →→→e.Property(x => x.Id).HasDefaultValueSql("NEWSEQUENTIALID()");
339     →→→e.Property(x => x.TargetType).HasMaxLength(16).IsRequired();
340     →→→e.Property(x => x.Status).HasMaxLength(16).HasDefaultValue("queued");
341     →→→e.Property(x => x.ErrorMessage).HasMaxLength(1000);
342     →→→e.Property(x => x.CreatedAt).HasDefaultValueSql("sysutcdatetime()");
343     →→→e.HasIndex(x => new { x.OwnerId, x.CreatedAt }).HasDatabaseName("IX_ingest_owner_created").IsDescending(true);
344   →→→} );
345
346
347   →→→// ----- OUTFITS (этап 2)
348   →→b.Entity<Outfit>(e =>
349   →→{
350     →→→e.ToTable("outfits");
351     →→→e.HasKey(x => x.Id).IsClustered(false);
352     →→→e.Property(x => x.Id).HasDefaultValueSql("NEWSEQUENTIALID()");
353     →→→e.Property(x => x.Title).HasMaxLength(160);
354     →→→e.Property(x => x.Note).HasMaxLength(1000);
355     →→→e.Property(x => x.Visibility).HasMaxLength(7).HasDefaultValue("private");
356     →→→e.Property(x => x.isHidden).HasDefaultValue(false);
357     →→→e.Property(x => x.ModerationStatus).HasMaxLength(16).HasDefaultValue("visible");
358     →→→e.Property(x => x.LikeCount).HasDefaultValue(0);
359     →→→e.Property(x => x.SaveCount).HasDefaultValue(0);
360     →→→e.Property(x => x.CreatedAt).HasDefaultValueSql("sysutcdatetime()");
361     →→→e.Property(x => x.RowVersion).IsRowVersion();
362     →→→e.HasOne(x => x.Owner).WithMany(x => x.Outfits).HasForeignKey(x => x.OwnerId).onDelete(DeleteBehavior.NoAction);
363     →→→e.HasOne(x => x.Wardrobe).WithMany().HasForeignKey(x => x.WardrobeId).onDelete(DeleteBehavior.NoAction);
364     →→→e.HasIndex(x => new { x.OwnerId, x.CreatedAt }).IsClustered().IsDescending(true).HasDatabaseName(
365       "CI_outfits_owner_created");
366     →→→e.HasIndex(x => x.Visibility)
367       →→→.HasFilter("[visibility] = 'public'");
368     →→→e.IncludeProperties(x => new { x.OwnerId, x.LikeCount, x.SaveCount });
369   →→→} );
370
371   →→b.Entity<OutfitItem>(e =>
372   →→{
373     →→→e.ToTable("outfit_items");
374     →→→e.HasKey(x => new { x.OutfitId, x.GarmentId });
375     →→→e.Property(x => x.RotationDeg).HasPrecision(6,2);
376     →→→e.Property(x => x.Scale).HasPrecision(9,4);
377     →→→e.Property(x => x.X).HasPrecision(9,4);
378     →→→e.Property(x => x.Y).HasPrecision(9,4);
379     →→→e.HasOne(x => x.Outfit).WithMany(x => x.Items).HasForeignKey(x => x.OutfitId).onDelete(DeleteBehavior.Cascade);
380     →→→e.HasOne(x => x.Garment).WithMany().HasForeignKey(x => x.GarmentId).onDelete(DeleteBehavior.NoAction);
381     →→→e.HasIndex(x => new { x.GarmentId, x.OutfitId }).HasDatabaseName("IX_item_garment_to_outfits");
382   →→→} );
383
384   →→b.Entity<Follow>(e =>
385   →→{
386     →→→e.ToTable("follows");
387     →→→e.HasKey(x => new { x.FollowerId, x.FolloweeId });
388     →→→e.Property(x => x.CreatedAt).HasDefaultValueSql("sysutcdatetime()");
389     →→→e.HasOne(x => x.Follower).WithMany().HasForeignKey(x => x.FollowerId).onDelete(DeleteBehavior.NoAction);
390     →→→e.HasOne(x => x.Followee).WithMany().HasForeignKey(x => x.FolloweeId).onDelete(DeleteBehavior.NoAction);
391     →→→e.HasIndex(x => new { x.FolloweeId, x.CreatedAt }).HasDatabaseName("IX_followee").IsDescending(true);
392   →→→} );
393
394   →→b.Entity<UserOutfitReaction>(e =>
395   →→{
396     →→→e.ToTable("user_outfit_reactions");
397     →→→e.HasKey(x => new { x.UserId, x.OutfitId, x.Type });
398     →→→e.Property(x => x.Type).HasMaxLength(8).IsRequired();
399     →→→e.Property(x => x.CreatedAt).HasDefaultValueSql("sysutcdatetime()");
400     →→→e.HasIndex(x => new { x.OutfitId, x.Type }).HasDatabaseName("IX_reactions_outfit");
401   →→→} );
402
403
404   →→→// ----- TECH / MODERATION / AUDIT
405   →→b.Entity<SearchIndexState>(e =>
406   →→{
407     →→→e.ToTable("search_index_state");
408     →→→e.HasKey(x => new { x.EntityType, x.EntityId, x.IndexName });
409     →→→e.Property(x => x.EntityType).HasMaxLength(16).IsRequired();
410     →→→e.Property(x => x.IndexName).HasMaxLength(64).IsRequired();
411     →→→e.Property(x => x.IndexVersion).HasDefaultValue(1);
412     →→→e.Property(x => x.VectorStatus).HasMaxLength(16).HasDefaultValue("pending");
413   →→→} );
414
415   →→b.Entity<ContentLabel>(e =>

```

```

416    →→→→→e.ToTable("content_labels");
417    →→→→→e.HasKey(x => x.Id);
418    →→→→→e.Property(x => x.TargetType).HasMaxLength(16).IsRequired();
419    →→→→→e.Property(x => x.Label).HasMaxLength(32).IsRequired();
420    →→→→→e.Property(x => x.Score).HasPrecision(5,4).IsRequired();
421    →→→→→e.Property(x => x.CreatedAt).HasDefaultValueSql("sysutcdatetime()");
422    →→→→→e.HasIndex(x => new { x.TargetType, x.TargetId }).HasDatabaseName("IX_labels_target");
423    →→→→→} );
424    →→→→→} );
425
426    →→→b.Entity<AuditLog>(e =>
427    →→{
428    →→→→→e.ToTable("audit_log");
429    →→→→→e.HasKey(x => x.Id);
430    →→→→→e.Property(x => x.Action).HasMaxLength(32).IsRequired();
431    →→→→→e.Property(x => x.Payload);
432    →→→→→e.Property(x => x.CreatedAt).HasDefaultValueSql("sysutcdatetime()");
433    →→→→→e.HasIndex(x => new { x.EntityType, x.EntityId, x.CreatedAt }).HasDatabaseName("IX_audit_entity").IsDescending(true);
434    →→→→→} );
435
436
437    →→→→→// ----- LISTINGS / INTERESTS (этап 3)
438    →→→b.Entity<Listing>(e =>
439    →→{
440    →→→→→e.ToTable("listings");
441    →→→→→e.HasKey(x => x.Id);
442    →→→→→e.Property(x => x.Id).HasDefaultValueSql("NEWSEQUENTIALID()");
443    →→→→→e.Property(x => x.Type).HasMaxLength(12).HasDefaultValue("gift"); // gift|swap|sell
444    →→→→→e.Property(x => x.Note).HasMaxLength(500);
445    →→→→→e.Property(x => x.Status).HasMaxLength(12).HasDefaultValue("active");
446    →→→→→e.Property(x => x.PriceMinor);
447    →→→→→e.Property(x => x.CurrencyCode).HasMaxLength(3).HasDefaultValue("EUR");
448    →→→→→e.Property(x => x.CreatedAt).HasDefaultValueSql("sysutcdatetime()");
449    →→→→→e.Property(x => x.UpdatedAt);
450
451    →→→→→ehasOne(x => x.Owner).WithMany().HasForeignKey(x => x.OwnerId).onDelete(DeleteBehavior.NoAction);
452    →→→→→ehasOne(x => x.Garment).WithMany().HasForeignKey(x => x.GarmentId).onDelete(DeleteBehavior.NoAction);
453    →→→→→ehasOne(x => x.Outfit).WithMany().HasForeignKey(x => x.OutfitId).onDelete(DeleteBehavior.NoAction);
454
455    →→→→→// XOR check
456    →→→→→e.ToTable(tb => tb.HasCheckConstraint("CK_listings_xor",
457    →→→→→→→"(CASE WHEN [garment_id] IS NULL THEN 0 ELSE 1 END + CASE WHEN [outfit_id] IS NULL THEN 0 ELSE 1 END) = 1"));
458
459    →→→→→e.HasIndex(x => new { x.Status, x.CreatedAt });
460    →→→→→e.HasFilter("[status] = 'active'");
461    →→→→→e.IncludeProperties(x => new { x.OwnerId, x.Type, x.PriceMinor, x.CurrencyCode });
462    →→→→→e.HasDatabaseName("IX_listings_active");
463    →→→→→e.HasIndex(x => new { x.OwnerId, x.CreatedAt }).HasDatabaseName("IX_listings_owner").IsDescending(true);
464    →→→→→} );
465
466    →→→b.Entity<Interest>(e =>
467    →→{
468    →→→→→e.ToTable("interests");
469    →→→→→e.HasKey(x => x.Id);
470    →→→→→e.Property(x => x.Id).HasDefaultValueSql("NEWSEQUENTIALID()");
471    →→→→→e.Property(x => x.Kind).HasMaxLength(12).HasDefaultValue("inquiry");
472    →→→→→e.Property(x => x.OfferCurrency).HasMaxLength(3).HasDefaultValue("EUR");
473    →→→→→e.Property(x => x.Status).HasMaxLength(16).HasDefaultValue("open");
474    →→→→→e.Property(x => x.CreatedAt).HasDefaultValueSql("sysutcdatetime()");
475    →→→→→e.Property(x => x.UpdatedAt);
476
477    →→→→→ehasOne(x => x.Listing).WithMany().HasForeignKey(x => x.ListingId).onDelete(DeleteBehavior.NoAction);
478    →→→→→ehasOne(x => x.Initiator).WithMany().HasForeignKey(x => x.InitiatorId).onDelete(DeleteBehavior.NoAction);
479    →→→→→ehasOne(x => x.Owner).WithMany().HasForeignKey(x => x.OwnerId).onDelete(DeleteBehavior.NoAction);
480
481    →→→→→e.HasIndex(x => new { x.OwnerId, x.Status, x.CreatedAt }).HasDatabaseName("IX_interests_owner_status").IsDescending(true);
482    →→→→→e.HasIndex(x => new { x.InitiatorId, x.CreatedAt }).HasDatabaseName("IX_interests_initiator").IsDescending(true);
483    →→→→→} );
484
485    →→→b.Entity<InterestMessage>(e =>
486    →→{
487    →→→→→e.ToTable("interest_messages");
488    →→→→→e.HasKey(x => x.Id);
489    →→→→→e.Property(x => x.Id).HasDefaultValueSql("NEWSEQUENTIALID()");
490    →→→→→e.Property(x => x.Body).HasMaxLength(2000).IsRequired();
491    →→→→→e.Property(x => x.CreatedAt).HasDefaultValueSql("sysutcdatetime()");
492    →→→→→ehasOne(x => x.Interest).WithMany().HasForeignKey(x => x.InterestId).onDelete(DeleteBehavior.Cascade);
493    →→→→→ehasOne(x => x.Sender).WithMany().HasForeignKey(x => x.SenderId).onDelete(DeleteBehavior.NoAction);
494    →→→→→e.HasIndex(x => new { x.InterestId, x.CreatedAt }).HasDatabaseName("IX_interest_messages");
495    →→→→→} );
496
497    →→→b.Entity<SwapItem>(e =>
498    →→{
499    →→→→→e.ToTable("swap_items");
500    →→→→→e.HasKey(x => new { x.InterestId, x.GarmentId, x.Role });
501    →→→→→e.Property(x => x.Role).HasMaxLength(8).HasDefaultValue("offer"); // offer|request
502    →→→→→ehasOne(x => x.Garment).WithMany().HasForeignKey(x => x.GarmentId).onDelete(DeleteBehavior.NoAction);
503    →→→→→e.HasIndex(x => x.InterestId).HasDatabaseName("IX_swap_items_interest");
504    →→→→→} );
505
506    →→→b.Entity<PeerReview>(e =>
507    →→{
508    →→→→→e.ToTable("peer_reviews");
509    →→→→→e.HasKey(x => x.Id);
510    →→→→→e.Property(x => x.Id).HasDefaultValueSql("NEWSEQUENTIALID()");
511    →→→→→e.Property(x => x.Rating).HasColumnType("tinyint");
512    →→→→→e.Property(x => x.Comment).HasMaxLength(500);
513    →→→→→e.Property(x => x.CreatedAt).HasDefaultValueSql("sysutcdatetime()");
514    →→→→→e.HasIndex(x => new { x.RevieweeId, x.CreatedAt }).HasDatabaseName("IX_reviews_user").IsDescending(true);
515    →→→→→e.HasIndex(x => new { x.InterestId, x.ReviewerId }).IsUnique().HasDatabaseName("UX_peer_reviews_once");
516    →→→→→} );
517
518    →→→} );

```

| | KEY | VALUE |
|-----|-------------|---|
| 1. | Id: | Pagrindinis raktas (GUID). |
| 2. | Email: | Vartotojo el. paštas (unikalus). |
| 3. | Name: | Rodomas vardas (nebūtinės). |
| 4. | Role: | Rolė: user moderator admin. |
| 5. | Status: | Būsena: active blocked deleted (soft delete). |
| 6. | CreatedAt: | Sukūrimo UTC laikas. |
| 7. | EmailNorm: | Normalizuotas el. paštas (lowercase, tik skaitymui). |
| 8. | RowVersion: | Konkurencingumo žyma (SQL Server rowversion) optimistinėi blokavimui kontrolei. |
| 9. | Avatar: | 1:N ryšys su vartotojo avataru. |
| 10. | Wardrobes: | 1:N vartotojo spintos. |
| 11. | Garments: | 1:N visos vartotojo drabužių įrašai. |
| 12. | Outfits: | 1:N vartotojo sukurti deriniai. |

```

542 public class User
543 {
544     public Guid Id { get; set; } // Nado
545     public string Email { get; set; } = default!; // Nado
546     public string? Name { get; set; } // Nado
547     public string Role { get; set; } = "user"; // Nado
548     public string Status { get; set; } = "active"; // Nado
549     public DateTime CreatedAt { get; set; } // Nado
550     public string? EmailNorm { get; private set; } // computed (podumaju 4to eto)
551     public byte[] RowVersion { get; set; } = default!; // 4to eto?
552
553     public Avatar? Avatar { get; set; } // Sozdadim v 3D?
554     public ICollection<Wardrobe> Wardrobes { get; set; } = new List<Wardrobe>(); // Skafy?
555     public ICollection<Garment> Garments { get; set; } = new List<Garment>(); // Sety?
556     public ICollection<Outfit> Outfits { get; set; } = new List<Outfit>(); // Odezda?
557 }
558
559
560
561

```

| | KEY | VALUE |
|----|---|--|
| 1. | UserId: | PK ir FK → User.Id. |
| 2. | User: | Navigacija į vartotoja. |
| 3. | Sex: | Lytis (laisvas tekstas; rekomenduojama male female other). |
| 4. | HeightCm / WeightKg: | Ūgis / svoris (cm / kg). |
| 5. | ChestCm / WaistCm / HipsCm / FootCm / ShoulderCm: | Kūno apimtys (mm arba cm – čia cm). |
| 6. | SkinTone / HairColor / BodyType: | Išvaizdos atributai (laisvi). |
| 7. | PxPerMm: | Kalibravimas (pikseliai per mm) pagal A4/kortele. |
| 8. | UpdatedAt: | Paskutinio atnaujinimo UTC. |

```

562
563
564
565
566
567
568
569
570
571
572
573
574
575 public class Avatar
576 {
577     [Key] public Guid UserId { get; set; }
578     public User User { get; set; } = default!;
579
580     public string? Sex { get; set; }
581     public int HeightCm { get; set; }
582     public int WeightKg { get; set; }
583     public int? ChestCm { get; set; }
584     public int? WaistCm { get; set; }
585     public int? HipsCm { get; set; }
586     public int? FootCm { get; set; }
587     public int? ShoulderCm { get; set; }
588     public string? SkinTone { get; set; }
589     public string? HairColor { get; set; }
590     public string? BodyType { get; set; }
591     public decimal? PxPerMm { get; set; }
592     public DateTime UpdatedAt { get; set; }
593 }
594
595 public class AvatarCapture
596 {
597     public Guid Id { get; set; }
598     public Guid UserId { get; set; }
599     public string View { get; set; } = default!;
600     public string? Calibrator { get; set; }
601     public decimal? PxPerMm { get; set; }
602     public string? Device { get; set; }
603     public DateTime CreatedAt { get; set; }
604 }
605
606 // Lookups
607 public class Brand
608 {
609     public int Id { get; set; }
610     public string Name { get; set; } = default!;
611 }
612 public class Category
613 {
614     public int Id { get; set; }
615     public string Code { get; set; } = default!;
616     public string Name { get; set; } = default!;
617 }
618
619 public class Condition
620 {
621     public int Id { get; set; }
622     public string Code { get; set; } = default!;
623     public string Name { get; set; } = default!;

```

```

624 }
625
626 public class Fit
627 {
628     →→public int Id { get; set; }
629     →→public string Code { get; set; } = default!;
630     →→public string Name { get; set; } = default!;
631 }
632
633 public class Color
634 {
635     →→public int Id { get; set; }
636     →→public string Name { get; set; } = default!;
637     →→public string Hex { get; set; } = default!;
638     →→public double? L { get; set; }
639     →→public double? A { get; set; }
640     →→public double? B { get; set; }
641 }
642
643 public class Pattern
644 {
645     →→public int Id { get; set; }
646     →→public string Name { get; set; } = default!;
647 }
648
649 public class Material
650 {
651     →→public int Id { get; set; }
652     →→public string Name { get; set; } = default!;
653 }
654
655 public class Season
656 {
657     →→[Key] public string Code { get; set; } = default!;
658     →→public string Name { get; set; } = default!;
659 }
660
661 public class Currency
662 {
663     →→[Key] public string Code { get; set; } = default!;
664     →→public string Name { get; set; } = default!;
665 }
666
667 public class Wardrobe
668 {
669     →→public Guid Id { get; set; }
670     →→public Guid OwnerId { get; set; }
671     →→public User Owner { get; set; } = default!;
672     →→public string Title { get; set; } = default!;
673     →→public string? Description { get; set; }
674     →→public string Visibility { get; set; } = "private";
675     →→public bool IsDefault { get; set; }
676     →→public DateTime CreatedAt { get; set; }
677     →→public DateTime? UpdatedAt { get; set; }
678
679     →→public ICollection<Garment> Garments { get; set; } = new List<Garment>();
680 }
681
682 public class Garment
683 {
684     →→public Guid Id { get; set; }
685     →→public Guid OwnerId { get; set; }
686     →→public User Owner { get; set; } = default!;
687     →→public Guid WardrobeId { get; set; }
688     →→public Wardrobe Wardrobe { get; set; } = default!;
689
690     →→public string Title { get; set; } = default!;
691     →→public int? BrandId { get; set; }
692     →→public Brand? Brand { get; set; }
693     →→public int? CategoryId { get; set; }
694     →→public Category Category { get; set; } = default!;
695     →→public string? Size { get; set; }
696     →→public int? FitId { get; set; }
697     →→public Fit? Fit { get; set; }
698     →→public int? ConditionId { get; set; }
699     →→public Condition? Condition { get; set; }
700     →→public string Visibility { get; set; } = "private";
701     →→public bool IsForSale { get; set; }
702     →→public int? PriceMinor { get; set; }
703     →→public string? CurrencyCode { get; set; } = "EUR";
704     →→public decimal? PxPerMm { get; set; }
705     →→public byte[]? Phash { get; set; }
706     →→public byte[]? PhashPrefix { get; private set; } // computed
707     →→public bool IsHidden { get; set; }
708     →→public string ModerationStatus { get; set; } = "visible";
709     →→public DateTime CreatedAt { get; set; }
710     →→public DateTime? UpdatedAt { get; set; }
711     →→public byte[] RowVersion { get; set; } = default!;
712
713     →→public ICollection<GarmentImage> Images { get; set; } = new List<GarmentImage>();
714     →→public ICollection<GarmentColor> Colors { get; set; } = new List<GarmentColor>();
715     →→public ICollection<GarmentPattern> Patterns { get; set; } = new List<GarmentPattern>();
716     →→public ICollection<GarmentMaterial> Materials { get; set; } = new List<GarmentMaterial>();
717     →→public ICollection<GarmentSeason> Seasons { get; set; } = new List<GarmentSeason>();
718     →→public ICollection<GarmentTag> Tags { get; set; } = new List<GarmentTag>();
719 }
720
721 public class GarmentImage
722 {
723     →→public Guid Id { get; set; }
724     →→public Guid GarmentId { get; set; }
725     →→public Garment Garment { get; set; } = default!;
726     →→public string Url { get; set; } = default!;
727     →→public int? Width { get; set; }
728     →→public int? Height { get; set; }

```

```
729     →public bool IsPrimary { get; set; }
730     →public DateTime CreatedAt { get; set; }
731 }
732
733 public class GarmentColor
734 {
735     →public Guid GarmentId { get; set; }
736     →public int ColorId { get; set; }
737     →public bool IsPrimary { get; set; }
738 }
739
740 public class GarmentPattern
741 {
742     →public Guid GarmentId { get; set; }
743     →public int PatternId { get; set; }
744 }
745
746 public class GarmentMaterial
747 {
748     →public Guid GarmentId { get; set; }
749     →public int MaterialId { get; set; }
750 }
751
752 public class GarmentSeason
753 {
754     →public Guid GarmentId { get; set; }
755     →public string SeasonCode { get; set; } = default!;
756 }
757
758 public class GarmentTag
759 {
760     →public Guid GarmentId { get; set; }
761     →public string Tag { get; set; } = default!;
762 }
763
764 public class GarmentCapture
765 {
766     →public Guid Id { get; set; }
767     →public Guid GarmentId { get; set; }
768     →public decimal? PxPerMm { get; set; }
769     →public string? Calibrator { get; set; }
770     →public string? Device { get; set; }
771     →public string? Notes { get; set; }
772     →public DateTime CreatedAt { get; set; }
773 }
774
775 public class GarmentVector
776 {
777     →[Key] →public Guid GarmentId { get; set; }
778     →public string Model { get; set; } = default!;
779     →public short Dims { get; set; }
780     →public bool IsL2Normalized { get; set; } = true;
781     →public byte[] Vector { get; set; } = default!;
782     →public DateTime CreatedAt { get; set; }
783 }
784
785 public class MediaIngestJob
786 {
787     →public Guid Id { get; set; }
788     →public Guid OwnerId { get; set; }
789     →public string TargetType { get; set; } = default!; // garment/avatar
790     →public Guid? TargetId { get; set; }
791     →public string Status { get; set; } = "queued";
792     →public string? ErrorMessage { get; set; }
793     →public DateTime CreatedAt { get; set; }
794 }
795
796 // Этап 2
797 public class Outfit
798 {
799     →public Guid Id { get; set; }
800     →public Guid OwnerId { get; set; }
801     →public User Owner { get; set; } = default!;
802     →public Guid? WardrobeId { get; set; }
803     →public Wardrobe? Wardrobe { get; set; }
804
805     →public string? Title { get; set; }
806     →public string? Note { get; set; }
807     →public string Visibility { get; set; } = "private";
808     →public bool IsHidden { get; set; }
809     →public string ModerationStatus { get; set; } = "visible";
810     →public int LikeCount { get; set; }
811     →public int SaveCount { get; set; }
812     →public DateTime CreatedAt { get; set; }
813     →public DateTime? UpdatedAt { get; set; }
814     →public byte[] RowVersion { get; set; } = default!;
815
816     →public ICollection<OutfitItem> Items { get; set; } = new List<OutfitItem>();
817 }
818
819 public class OutfitItem
820 {
821     →public Guid OutfitId { get; set; }
822     →public Outfit Outfit { get; set; } = default!;
823     →public Guid GarmentId { get; set; }
824     →public Garment Garment { get; set; } = default!;
825     →public int? ZIndex { get; set; }
826     →public decimal? X { get; set; }
827     →public decimal? Y { get; set; }
828     →public decimal? Scale { get; set; }
829     →public decimal? RotationDeg { get; set; }
830 }
831
832 public class Follow
833 {
```

```

834     →public Guid FollowerId { get; set; }
835     →public User Follower { get; set; } = default!;
836     →public Guid FolloweeId { get; set; }
837     →public User Followee { get; set; } = default!;
838     →public DateTime CreatedAt { get; set; }
839 }
840
841 public class UserOutfitReaction
842 {
843     →public Guid UserId { get; set; }
844     →public Guid OutfitId { get; set; }
845     →public string Type { get; set; } = default!; // like|save
846     →public DateTime CreatedAt { get; set; }
847 }
848
849 // Tech / moderation / audit
850 public class SearchIndexState
851 {
852     →[MaxLength(16)] public string EntityType { get; set; } = default!; // garment|outfit
853     →public Guid EntityId { get; set; }
854     →[MaxLength(64)] public string IndexName { get; set; } = default!;
855     →public int IndexVersion { get; set; } = 1;
856     →[MaxLength(16)] public string VectorStatus { get; set; } = "pending";
857     →public DateTime? LastIndexedAt { get; set; }
858     →public string? ErrorMessage { get; set; }
859 }
860 public class ContentLabel
861 {
862     →public long Id { get; set; }
863     →public string TargetType { get; set; } = default!;
864     →public Guid TargetId { get; set; }
865     →public string Label { get; set; } = default!;
866     →public decimal Score { get; set; }
867     →public DateTime CreatedAt { get; set; }
868 }
869 public class AuditLog
870 {
871     →public long Id { get; set; }
872     →public string EntityType { get; set; } = default!;
873     →public Guid EntityId { get; set; }
874     →public string Action { get; set; } = default!;
875     →public Guid? ActorId { get; set; }
876     →public string? Payload { get; set; }
877     →public DateTime CreatedAt { get; set; }
878 }
879
880 // Этап 3
881 public class Listing
882 {
883     →public Guid Id { get; set; }
884     →public Guid OwnerId { get; set; }
885     →public User Owner { get; set; } = default!;
886     →public Guid? GarmentId { get; set; }
887     →public Garment? Garment { get; set; }
888     →public Guid? OutfitId { get; set; }
889     →public Outfit? Outfit { get; set; }
890
891     →public string Type { get; set; } = "gift"; // gift|swap|sell
892     →public string? Note { get; set; }
893     →public string Status { get; set; } = "active"; // active|paused|closed
894     →public int? PriceMinor { get; set; } // только для sell (деньги вне платформы)
895     →public string? CurrencyCode { get; set; } = "EUR";
896     →public DateTime CreatedAt { get; set; }
897     →public DateTime? UpdatedAt { get; set; }
898 }
899
900 public class Interest
901 {
902     →public Guid Id { get; set; }
903     →public Guid ListingId { get; set; }
904     →public Listing Listing { get; set; } = default!;
905     →public Guid InitiatorId { get; set; }
906     →public User Initiator { get; set; } = default!;
907     →public Guid OwnerId { get; set; } // денормализация владельца листинга
908     →public User Owner { get; set; } = default!;
909
910     →public string Kind { get; set; } = "inquiry"; // inquiry|offer|swap_offer
911     →public int? OfferPriceMinor { get; set; }
912     →public string? OfferCurrency { get; set; } = "EUR";
913     →public string Status { get; set; } = "open"; // open|accepted|declined|closed
914     →public DateTime CreatedAt { get; set; }
915     →public DateTime? UpdatedAt { get; set; }
916 }
917
918 public class InterestMessage
919 {
920     →public Guid Id { get; set; }
921     →public Guid InterestId { get; set; }
922     →public Interest Interest { get; set; } = default!;
923     →public Guid SenderId { get; set; }
924     →public User Sender { get; set; } = default!;
925     →public string Body { get; set; } = default!;
926     →public DateTime CreatedAt { get; set; }
927     →public DateTime? ReadAt { get; set; }
928 }
929
930 public class SwapItem
931 {
932     →public Guid InterestId { get; set; }
933     →public Interest Interest { get; set; } = default!;
934     →public Guid GarmentId { get; set; }
935     →public Garment Garment { get; set; } = default!;
936     →public string Role { get; set; } = "offer"; // offer|request
937 }
938

```

```
939 public class PeerReview
940 {
941     →public Guid Id { get; set; }
942     →public Guid InterestId { get; set; }
943     →public Guid ReviewerId { get; set; }
944     →public Guid RevieweeId { get; set; }
945     →public byte Rating { get; set; } // 1..5
946     →public string? Comment { get; set; }
947     →public DateTime CreatedAt { get; set; }
948 }
949
```