Soccer

Contents

[1 Entity Relationship Diagram 2](#_Toc477774822)

[2 Models 3](#_Toc477774823)

[2.1 Date 3](#_Toc477774824)

[2.2 Group 3](#_Toc477774825)

[2.3 GroupUser 3](#_Toc477774826)

[2.4 League 4](#_Toc477774827)

[2.5 Match 4](#_Toc477774828)

[2.6 Prediction 5](#_Toc477774829)

[2.7 Status 6](#_Toc477774830)

[2.8 Team 6](#_Toc477774831)

[2.9 Tournament 7](#_Toc477774832)

[2.10 TournamentGroup 7](#_Toc477774833)

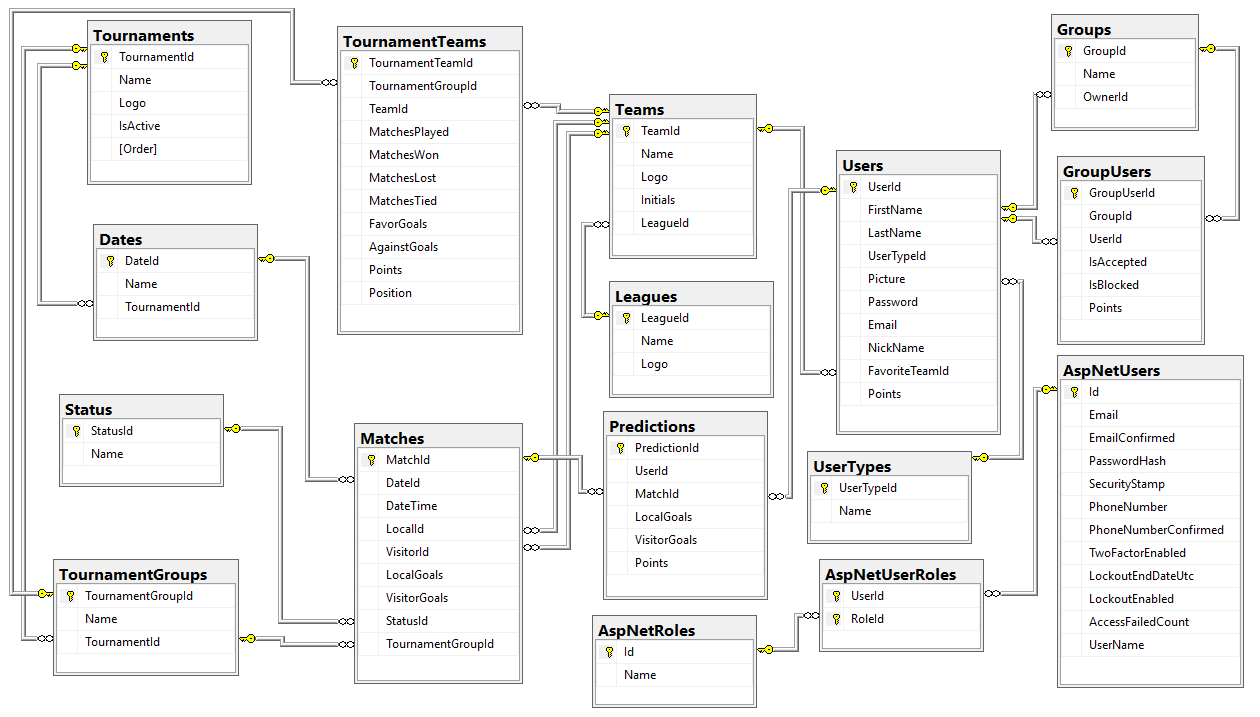
[2.11 TournamentTeam 8](#_Toc477774834)

[2.12 User 8](#_Toc477774835)

[2.13 UserType 9](#_Toc477774836)

[3 Special Relationships 10](#_Toc477774837)

# Entity Relationship Diagram



# Models

## Date

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

namespace Domain

{

public class Date

{

[Key]

public int DateId { get; set; }

[Required(ErrorMessage = "The field {0} is required")]

[MaxLength(50, ErrorMessage = "The maximun length for field {0} is {1} characters")]

[Index("Date\_Name\_TournamentId\_Index", IsUnique = true, Order = 1)]

[Display(Name = "Date")]

public string Name { get; set; }

[Display(Name = "Tournament")]

[Index("Date\_Name\_TournamentId\_Index", IsUnique = true, Order = 2)]

public int TournamentId { get; set; }

public virtual Tournament Tournament { get; set; }

public virtual ICollection<Match> Matches { get; set; }

}

}

## Group

using Newtonsoft.Json;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

namespace Domain

{

public class Group

{

[Key]

public int GroupId { get; set; }

[Required(ErrorMessage = "The field {0} is required")]

[MaxLength(50, ErrorMessage = "The maximun length for field {0} is {1} characters")]

[Index("Group\_Name\_Index", IsUnique = true)]

[Display(Name = "Group")]

public string Name { get; set; }

[DataType(DataType.ImageUrl)]

public string Logo { get; set; }

[Display(Name = "User")]

public int OwnerId { get; set; }

public virtual User Owner { get; set; }

public virtual ICollection<GroupUser> GroupUsers { get; set; }

}

}

## GroupUser

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

namespace Domain

{

public class GroupUser

{

[Key]

public int GroupUserId { get; set; }

[Index("GroupUser\_GroupId\_UserId\_Index", IsUnique = true, Order = 1)]

[Display(Name = "Group")]

public int GroupId { get; set; }

[Index("GroupUser\_GroupId\_UserId\_Index", IsUnique = true, Order = 2)]

[Display(Name = "User")]

public int UserId { get; set; }

[Display(Name = "Is accepted?")]

public bool IsAccepted { get; set; }

[Display(Name = "Is blocked?")]

public bool IsBlocked { get; set; }

public int Points { get; set; }

public virtual Group Group { get; set; }

public virtual User User { get; set; }

}

}

## League

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

namespace Domain

{

public class League

{

[Key]

public int LeagueId { get; set; }

[Required(ErrorMessage = "The field {0} is required")]

[MaxLength(50, ErrorMessage = "The maximun length for field {0} is {1} characters")]

[Index("League\_Name\_Index", IsUnique = true)]

[Display(Name = "League")]

public string Name { get; set; }

[DataType(DataType.ImageUrl)]

public string Logo { get; set; }

public virtual ICollection<Team> Teams { get; set; }

}

}

## Match

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

namespace Domain

{

public class Match

{

[Key]

public int MatchId { get; set; }

[Display(Name = "Date")]

public int DateId { get; set; }

[Display(Name = "Date time")]

[DataType(DataType.DateTime)]

public DateTime DateTime { get; set; }

[Display(Name = "Local")]

public int LocalId { get; set; }

[Display(Name = "Visitor")]

public int VisitorId { get; set; }

[Display(Name = "Local goals")]

public int LocalGoals { get; set; }

[Display(Name = "Visitor goals")]

public int VisitorGoals { get; set; }

[Display(Name = "Status")]

public int StatusId { get; set; }

[Display(Name = "Group")]

public int TournamentGroupId { get; set; }

public virtual Date Date { get; set; }

public virtual Team Local { get; set; }

public virtual Team Visitor { get; set; }

public virtual Status Status { get; set; }

public virtual TournamentGroup TournamentGroup { get; set; }

public virtual ICollection<Prediction> Predictions { get; set; }

}

}

## Prediction

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

namespace Domain

{

public class Prediction

{

[Key]

public int PredictionId { get; set; }

[Index("Prediction\_UserId\_MatchId\_Index", IsUnique = true, Order = 1)]

[Display(Name = "User")]

public int UserId { get; set; }

[Index("Prediction\_UserId\_MatchId\_Index", IsUnique = true, Order = 2)]

[Display(Name = "Match")]

public int MatchId { get; set; }

[Display(Name = "Local goals")]

public int LocalGoals { get; set; }

[Display(Name = "Visitor goals")]

public int VisitorGoals { get; set; }

public int Points { get; set; }

public virtual User User { get; set; }

public virtual Match Match { get; set; }

}

}

## Status

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

namespace Domain

{

public class Status

{

[Key]

public int StatusId { get; set; }

[Required(ErrorMessage = "The field {0} is required")]

[MaxLength(50, ErrorMessage = "The maximun length for field {0} is {1} characters")]

[Index("Status\_Name\_Index", IsUnique = true)]

[Display(Name = "Status")]

public string Name { get; set; }

public virtual ICollection<Match> Matches { get; set; }

}

}

## Team

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

namespace Domain

{

public class Team

{

[Key]

public int TeamId { get; set; }

[Required(ErrorMessage = "The field {0} is required")]

[MaxLength(50, ErrorMessage = "The maximun length for field {0} is {1} characters")]

[Index("Team\_Name\_LeagueId\_Index", IsUnique = true, Order = 1)]

[Display(Name = "Team")]

public string Name { get; set; }

[DataType(DataType.ImageUrl)]

public string Logo { get; set; }

[Required(ErrorMessage = "The field {0} is required")]

[StringLength(3, ErrorMessage = "The length for field {0} must be {1} characters", MinimumLength = 3)]

public string Initials { get; set; }

[Display(Name = "League")]

[Index("Team\_Name\_LeagueId\_Index", IsUnique = true, Order = 2)]

public int LeagueId { get; set; }

public virtual League League { get; set; }

public virtual ICollection<User> Fans { get; set; }

public virtual ICollection<Match> Locals { get; set; }

public virtual ICollection<Match> Visitors { get; set; }

public virtual ICollection<TournamentTeam> TournamentTeams { get; set; }

}

}

## Tournament

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

namespace Domain

{

public class Tournament

{

[Key]

public int TournamentId { get; set; }

[Required(ErrorMessage = "The field {0} is required")]

[MaxLength(50, ErrorMessage = "The maximun length for field {0} is {1} characters")]

[Index("Tournament\_Name\_Index", IsUnique = true)]

[Display(Name = "Tournament")]

public string Name { get; set; }

[DataType(DataType.ImageUrl)]

public string Logo { get; set; }

[Display(Name = "Is Active?")]

public bool IsActive { get; set; }

[Display(Name = "Order")]

public int Order { get; set; }

public virtual ICollection<TournamentGroup> TournamentGroups { get; set; }

public virtual ICollection<Date> Dates { get; set; }

}

}

## TournamentGroup

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

namespace Domain

{

public class TournamentGroup

{

[Key]

public int TournamentGroupId { get; set; }

[Required(ErrorMessage = "The field {0} is required")]

[MaxLength(50, ErrorMessage = "The maximun length for field {0} is {1} characters")]

[Index("TournamentGroup\_Name\_TournamentId\_Index", IsUnique = true, Order = 1)]

[Display(Name = "Group")]

public string Name { get; set; }

[Display(Name = "Tournament")]

[Index("TournamentGroup\_Name\_TournamentId\_Index", IsUnique = true, Order = 2)]

public int TournamentId { get; set; }

public virtual Tournament Tournament { get; set; }

public virtual ICollection<Match> Matches { get; set; }

public virtual ICollection<TournamentTeam> TournamentTeams { get; set; }

}

}

## TournamentTeam

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

namespace Domain

{

public class TournamentTeam

{

[Key]

public int TournamentTeamId { get; set; }

[Index("TournamentTeam\_TournamentGroupId\_TeamId\_Index", IsUnique = true, Order = 1)]

[Display(Name = "Group")]

public int TournamentGroupId { get; set; }

[Index("TournamentTeam\_TournamentGroupId\_TeamId\_Index", IsUnique = true, Order = 2)]

[Display(Name = "Team")]

public int TeamId { get; set; }

[Display(Name = "Matches played")]

public int MatchesPlayed { get; set; }

[Display(Name = "Matches won")]

public int MatchesWon { get; set; }

[Display(Name = "Matches lost")]

public int MatchesLost { get; set; }

[Display(Name = "Matches tied")]

public int MatchesTied { get; set; }

[Display(Name = "Favor goals")]

public int FavorGoals { get; set; }

[Display(Name = "Against goals")]

public int AgainstGoals { get; set; }

public int Points { get; set; }

public int Position { get; set; }

public virtual TournamentGroup TournamentGroup { get; set; }

public virtual Team Team { get; set; }

}

}

## User

using Newtonsoft.Json;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

namespace Domain

{

public class User

{

[Key]

public int UserId { get; set; }

[Required(ErrorMessage = "The field {0} is required")]

[MaxLength(50, ErrorMessage = "The maximun length for field {0} is {1} characters")]

[Display(Name = "First name")]

public string FirstName { get; set; }

[Required(ErrorMessage = "The field {0} is required")]

[MaxLength(50, ErrorMessage = "The maximun length for field {0} is {1} characters")]

[Display(Name = "Last name")]

public string LastName { get; set; }

[Display(Name = "User type")]

public int UserTypeId { get; set; }

public virtual UserType UserType { get; set; }

[DataType(DataType.ImageUrl)]

public string Picture { get; set; }

[Required(ErrorMessage = "The field {0} is required")]

[MaxLength(100, ErrorMessage = "The maximun length for field {0} is {1} characters")]

[DataType(DataType.EmailAddress)]

[Index("User\_Email\_Index", IsUnique = true)]

public string Email { get; set; }

[Required(ErrorMessage = "The field {0} is required")]

[MaxLength(20, ErrorMessage = "The maximun length for field {0} is {1} characters")]

[Index("User\_NickName\_Index", IsUnique = true)]

[Display(Name = "Nick name")]

public string NickName { get; set; }

[Display(Name = "Favorite team")]

public int FavoriteTeamId { get; set; }

public int Points { get; set; }

public virtual Team FavoriteTeam { get; set; }

public virtual ICollection<Group> UserGroups { get; set; }

public virtual ICollection<GroupUser> GroupUsers { get; set; }

public virtual ICollection<Prediction> Predictions { get; set; }

}

}

## UserType

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

namespace Domain

{

public class UserType

{

[Key]

public int UserTypeId { get; set; }

[Required(ErrorMessage = "The field {0} is required")]

[MaxLength(50, ErrorMessage = "The maximun length for field {0} is {1} characters")]

[Index("UserType\_Name\_Index", IsUnique = true)]

[Display(Name = "User type")]

public string Name { get; set; }

public virtual ICollection<User> Users { get; set; }

}

}

# Special Relationships

protected override void OnModelCreating(DbModelBuilder modelBuilder)

{

modelBuilder.Conventions.Remove<OneToManyCascadeDeleteConvention>();

modelBuilder.Configurations.Add(new MatchesMap());

modelBuilder.Configurations.Add(new GroupsMap());

modelBuilder.Configurations.Add(new UsersMap());

}

using Domain;

using System.Data.Entity.ModelConfiguration;

namespace Backend.Models

{

internal class MatchesMap : EntityTypeConfiguration<Match>

{

public MatchesMap()

{

HasRequired(o => o.Local)

.WithMany(m => m.Locals)

.HasForeignKey(m => m.LocalId);

HasRequired(o => o.Visitor)

.WithMany(m => m.Visitors)

.HasForeignKey(m => m.VisitorId);

}

}

}

using Domain;

using System.Data.Entity.ModelConfiguration;

namespace Backend.Models

{

internal class GroupsMap : EntityTypeConfiguration<Group>

{

public GroupsMap()

{

HasRequired(o => o.Owner)

.WithMany(m => m.UserGroups)

.HasForeignKey(m => m.OwnerId);

}

}

}

using Domain;

using System.Data.Entity.ModelConfiguration;

namespace Backend.Models

{

internal class UsersMap : EntityTypeConfiguration<User>

{

public UsersMap()

{

HasRequired(o => o.FavoriteTeam)

.WithMany(m => m.Fans)

.HasForeignKey(m => m.FavoriteTeamId);

}

}

}