## Football Betting

Your task is to create a database for a **FootballBookmakerSystem**, using the **Code First** approach. It should look like this:



### Constraints

Your **namespaces** should be:

* P02\_FootballBetting – for your **Startup** class, if you have one
* P02\_FootballBetting.Data – for your **DbContext**
* P02\_FootballBetting.Data.Models – for your models

Your models should be:

* **FootballBettingContext** – your DbContext
* **Team** – **TeamId**, **Name**, **LogoUrl**, **Initials** (JUV, LIV, ARS…), **Budget**, **PrimaryKitColorId**, **SecondaryKitColorId**, **TownId**
* **Color** – **ColorId**, **Name**
* **Town** – **TownId**, **Name**, **CountryId**
* **Country** – **CountryId**, **Name**
* **Player** – **PlayerId**, **Name**, **SquadNumber**, **IsInjured**, **PositionId** , **TeamId**, **TownId**
* **Position** – **PositionId**, **Name**
* **PlayerStatistic** – **GameId**, **PlayerId**, **ScoredGoals**, **Assists**, **MinutesPlayed**
* **Game** – **GameId**, **HomeTeamId**, **AwayTeamId**, **HomeTeamGoals**, **AwayTeamGoals**, **HomeTeamBetRate**, **AwayTeamBetRate**, **DrawBetRate**, **DateTime**, **Result**
* **Bet** – **BetId**, **Amount**, **Prediction**, **DateTime**, **UserId**, **GameId**
* **User** – **UserId**, **Username**, **Name**, **Password**, **Email**, **Balance**

Table relationships:

* **A Team** has one **PrimaryKitColor** and one **SecondaryKitColor**
* **A Color** has **many PrimaryKitTeams** and **many SecondaryKitTeams**
* **A Team** residents in one **Town**
* **A Town** can host **several** **Teams**
* **A Game** has one **HomeTeam** and one **AwayTeam** and a **Team** can have **many** **HomeGames** and **many** **AwayGames**
* **A Town** can be placed in **one** **Country** and a **Country** can have many **Towns**
* **A Player** can play for **one** **Team** and **one** **Team** can have many **Players**
* **A Player** can play at one **Position** and one **Position** can be played by **many** **Players**
* **One** **Player** can play in **many** **Games** and in each **Game**, **many** **Players** take part (both collections must be named **PlayersStatistics**)
* **Many** **Bets** can be placed on **one** **Game**, but **a** **Bet** can be only on **one** **Game**
* Each bet for given game must have **Prediction** result
* **A Bet** can be placed by only **one** **User** and one **User** can place many **Bets**

Separate the **models**, **data** and **client** into **different layers** (projects).

---------------------------------------------------------------------------------------------------------------

public class Bet

{

public int BetId { get; set; }

public decimal Amount { get; set; }

public Prediction Prediction { get; set; }

public DateTime DateTime { get; set; }

public int UserId { get; set; }

[ForeignKey(nameof(UserId))]

public virtual User User { get; set; } = null!;

public int GameId { get; set; }

[ForeignKey(nameof(GameId))]

public virtual Game Game { get; set; } = null!;

}

public class Country

{

public Country()

{

Towns = new HashSet<Town>();

}

[Key]

public int CountryId { get; set; }

[MaxLength(50)]

public string Name { get; set; } = null!;

public virtual ICollection<Town> Towns { get; set; } = null!;

}

-------------------------------------------------------------------------------------

public class Color

{

public Color()

{

PrimaryKitTeams = new HashSet<Team>();

SecondaryKitTeams = new HashSet<Team>();

}

public int ColorId { get; set; }

[MaxLength(20)]

public string Name { get; set; } = null!;

[InverseProperty(nameof(Team.PrimaryKitColor))]

public virtual ICollection<Team> PrimaryKitTeams { get; set; } = null!;

[InverseProperty(nameof(Team.SecondaryKitColor))]

public virtual ICollection<Team> SecondaryKitTeams { get; set; } = null!;

}

public class Position

{

public Position()

{

Players = new HashSet<Player>();

}

public int PositionId { get; set; }

[MaxLength(20)]

public string Name { get; set; } = null!;

public virtual ICollection<Player> Players { get; set; }

}

public class Game

{

public Game()

{

PlayersStatistics = new HashSet<PlayerStatistic>();

Bets = new HashSet<Bet>();

}

public int GameId { get; set; }

public int HomeTeamId { get; set; }

[ForeignKey(nameof(HomeTeamId))]

public virtual Team HomeTeam { get; set; } = null!;

public int AwayTeamId { get; set; }

[ForeignKey(nameof(AwayTeamId))]

public virtual Team AwayTeam { get; set; } = null!;

public byte HomeTeamGoals { get; set; }

public byte AwayTeamGoals { get; set; }

public double HomeTeamBetRate { get; set; }

public double AwayTeamBetRate { get; set; }

public double DrawBetRate { get; set; }

[Required]

public DateTime DateTime { get; set; }

[MaxLength(10)]

public string Result { get; set; } = null!;

public virtual ICollection<Bet> Bets { get; set; } = null!;

public virtual ICollection<PlayerStatistic> PlayersStatistics { get; set; } = null!;

}

public class Player

{

public Player()

{

PlayersStatistics = new HashSet<PlayerStatistic>();

}

public int PlayerId { get; set; }

[MaxLength(60)]

public string Name { get; set; } = null!;

public int SquadNumber { get; set; }

public bool IsInjured { get; set; }

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

public int PositionId { get; set; }

[ForeignKey(nameof(PositionId))]

public virtual Position Position { get; set; } = null!;

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

public int TeamId { get; set; }

[ForeignKey(nameof(TeamId))]

public virtual Team Team { get; set; } = null!;

//#####################################################

public int TownId { get; set; }

[ForeignKey(nameof(TownId))]

public virtual Town Town { get; set; } = null!;

//#####################################################

public virtual ICollection<PlayerStatistic> PlayersStatistics { get; set; } = null!;

}

public class Team

{

public Team()

{

Players = new HashSet<Player>();

HomeGames = new HashSet<Game>();

AwayGames = new HashSet<Game>();

}

public int TeamId { get; set; }

[MaxLength(50)]

public string Name { get; set; } = null!;

public string? LogoUrl { get; set; }

[MinLength(3)]

[MaxLength(3)]

public string Initials { get; set; } = null!;

public decimal Budget { get; set; }

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*!!!!!!!!!!!!!!!!

public int PrimaryKitColorId { get; set; }

[ForeignKey(nameof(PrimaryKitColorId))]

public virtual Color PrimaryKitColor { get; set; } = null!;

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

public int SecondaryKitColorId { get; set; }

[ForeignKey(nameof(SecondaryKitColorId))]

public virtual Color SecondaryKitColor { get; set; } = null!;

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

public int TownId { get; set; }

[ForeignKey(nameof(TownId))]

public virtual Town Town { get; set; } = null!;

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*!!!!!!!!!!!!!!!!

public virtual ICollection<Player> Players { get; set; } = null!;

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

[InverseProperty(nameof(Game.HomeTeam))]

public virtual ICollection<Game> HomeGames { get; set; } = null!;

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

[InverseProperty(nameof(Game.AwayTeam))]

public virtual ICollection<Game> AwayGames { get; set; } = null!;

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

}

public class PlayerStatistic

{

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

public int GameId { get; set; }

[ForeignKey(nameof(GameId))]

public virtual Game Game { get; set; } = null!;

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

public virtual int PlayerId { get; set; }

[ForeignKey(nameof(PlayerId))]

public virtual Player Player { get; set; } = null!;

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

public byte ScoredGoals { get; set; }

public byte Assists { get; set; }

public byte MinutesPlayed { get; set; }

}

public class Town

{

public Town()

{

Teams = new HashSet<Team>();

Players = new HashSet<Player>();

}

public int TownId { get; set; }

[MaxLength(50)]

public string Name { get; set; } = null!;

public int CountryId { get; set; }

[ForeignKey(nameof(CountryId))]

public virtual Country Country { get; set; } = null!;

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

public virtual ICollection<Player> Players { get; set; } = null!;

}

public class User

{

public User()

{

Bets = new HashSet<Bet>();

}

public int UserId { get; set; }

[MaxLength(80)]

public string Username { get; set; } = null!;

[MaxLength(80)]

public string Name { get; set; } = null!;

[MaxLength(50)]

public string Password { get; set; } = null!;

public string? Email { get; set; }

public decimal Balance { get; set; }

public virtual ICollection<Bet> Bets { get; set; } = null!;

}

public enum Prediction

{

Win,

Lose,

Draw

}

public class FootballBettingContext : DbContext

{

string connectionString = "Server=192.168.88.40 ,1434; Database = FootballBookmakerSystem; User Id = sa; Password = password;";

public DbSet<Team> Teams { get; set; } = null!;

public DbSet<Color> Colors { get; set; } = null!;

public DbSet<Town> Towns { get; set; } = null!;

public DbSet<Country> Countries { get; set; } = null!;

public DbSet<Player> Players { get; set; } = null!;

public DbSet<Position> Positions { get; set; } = null!;

public DbSet<PlayerStatistic> PlayersStatistics { get; set; } = null!;

public DbSet<Game> Games { get; set; } = null!;

public DbSet<Bet> Bets { get; set; } = null!;

public DbSet<User> Users { get; set; } = null!;

//-------------This is for JUDJE-------------------------------//////

public FootballBettingContext(DbContextOptions dbcontextoptions) : base(dbcontextoptions)

{

}

//protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)

//{

// optionsBuilder.UseSqlServer(connectionString);

//}

protected override void OnModelCreating(ModelBuilder modelBuilder)

{

modelBuilder.Entity<PlayerStatistic>(entity =>

{

entity

.HasKey(ps => new { ps.PlayerId, ps.GameId });

});

modelBuilder.Entity<Team>(entity =>

{

entity

.HasOne(t => t.PrimaryKitColor)

.WithMany(c => c.PrimaryKitTeams)

.HasForeignKey(t => t.PrimaryKitColorId)

.OnDelete(DeleteBehavior.NoAction);

});

modelBuilder.Entity<Team>(entity =>

{

entity

.HasOne(t => t.SecondaryKitColor)

.WithMany(c => c.SecondaryKitTeams)

.HasForeignKey(t => t.SecondaryKitColorId)

.OnDelete(DeleteBehavior.NoAction);

});

modelBuilder.Entity<Game>(entity =>

{

entity

.HasOne(g => g.HomeTeam)

.WithMany(t => t.HomeGames)

.HasForeignKey(g => g.HomeTeamId)

.OnDelete(DeleteBehavior.NoAction);

});

modelBuilder.Entity<Game>(entity =>

{

entity.HasOne(g => g.AwayTeam)

.WithMany(t => t.AwayGames)

.HasForeignKey(g => g.AwayTeamId)

.OnDelete(DeleteBehavior.NoAction);

});

modelBuilder.Entity<Player>(entity =>

{

entity

.HasOne(p => p.Position)

.WithMany(p => p.Players)

.HasForeignKey(p => p.PositionId)

.OnDelete(DeleteBehavior.NoAction);

});

modelBuilder.Entity<Player>(entity =>

{

entity

.HasOne(p => p.Team)

.WithMany(t => t.Players)

.HasForeignKey(p => p.TeamId)

.OnDelete(DeleteBehavior.NoAction);

});

modelBuilder.Entity<Player>(entity =>

{

entity.HasOne(p => p.Town)

.WithMany(t => t.Players)

.HasForeignKey(p => p.TownId)

.OnDelete(DeleteBehavior.NoAction);

});

}

}