🏆 TourneyPlanner 🏆

Syntax valg:
C#:

- Classes, enums and methods => **PascalCase**
- Private properties => **_leadingUnderscoreCamelCase**
- Constant properties => **ALL_CAPS**
- Interfaces start with uppercase I(i) => **IPascalCase**

Dart:

- Classes, enums, typedefs, and extensions => **UpperCamelCase**
- Libraries, packages, directories, and source files => **snake_case***(lowercase_with_underscores)*
- Variables, constants, parameters, and named parameters => **lowerCamelCase**

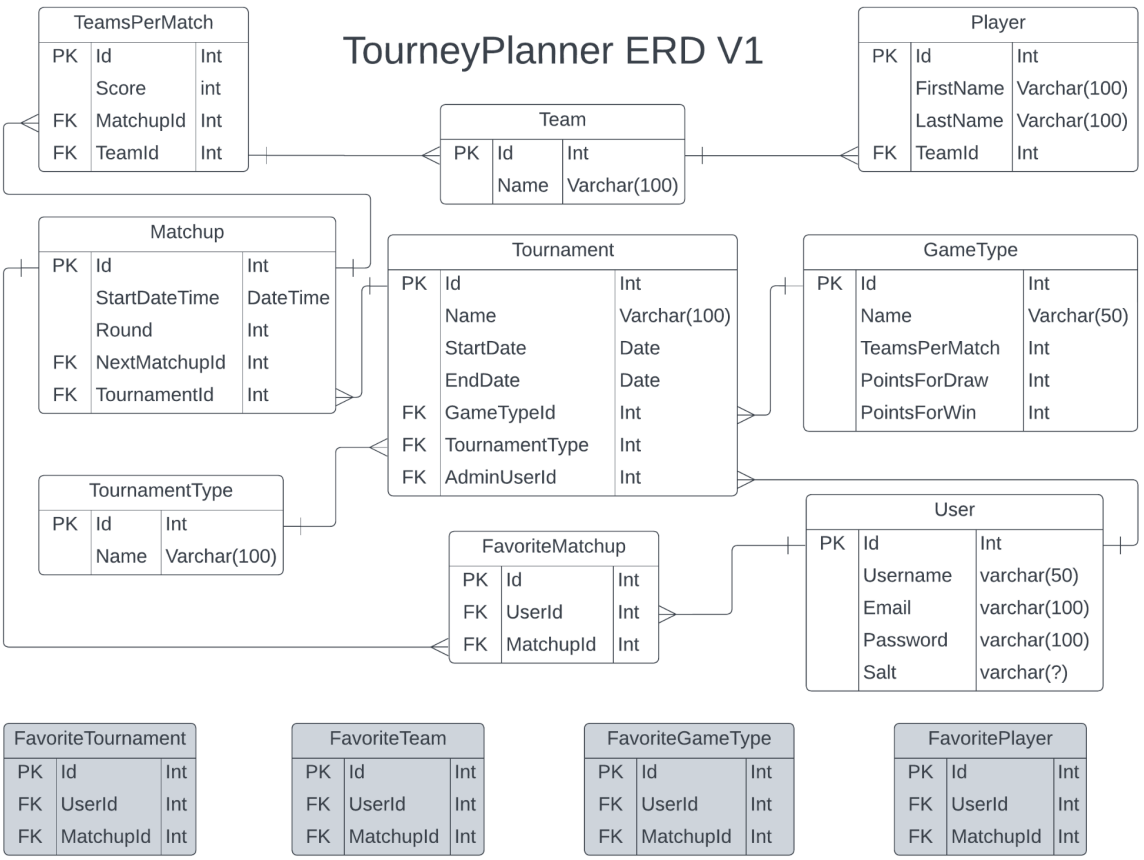
Problemformulering:
Hvordan kan man udvikle en effektiv administrativ løsning til at registrere og opretholde en oversigt over resultaterne af en turnering, som brugere kan følge med i på tværs af platforme?

Server konfiguration:
RAID type:

Virtualisering:

Andre extensions:

Databasestruktur:
ER-diagram:



Entiteter/attributter:
User: Id, Username, Email, Password, Salt
Team: Id, Name
TeamsPerMatch: TeamsPerMatchId, Score, MatchupId, TeamId
Tournament: Id, StartDate, EndDate, GameTypeId, TournamentTypeId, AdminUserId
TournamentType: Id, Name
GameType: Id, Name, TeamsPerMatch, PointsForDraw, PointsForWin
Matchup: Id, Rounds, TournamentId
Player: Id, TeamId, FirstName, LastName
FavoriteMatchup: Id, UserId, MatchupId

Mulige udvidelser:

- FavoriteTournament
- FavoriteTeam
- FavoriteGameType
- FavoritePlayer

Notifikationer:
Kommer til udtryk igennem hvad en bruger vælger at sætte som favorit. Hvis en kamp sættes som favorit, vil brugeren få en notifikation når kampen er færdig.

ORM:
Entity Framework - Database first

Certifikat:
Server Certificate:

- mkcert localhost 127.0.0.1 10.0.0.2 ::1

Client Certificate:

- mkcert -client -pkcs12 localhost 127.0.0.1 10.0.0.2 ::1
- default password = changeit

Input validering:
Front-end:
Implementeres via en form, så der ikke tillades at sende data til backend, uden at datatyper og minimumskrav som eksempelvis længde overholdes.

Back-end:
Implementeres ved at tjekke for null værdier, og at datatyperne er i det forventede format, inden der laves indsættelser i databasen.

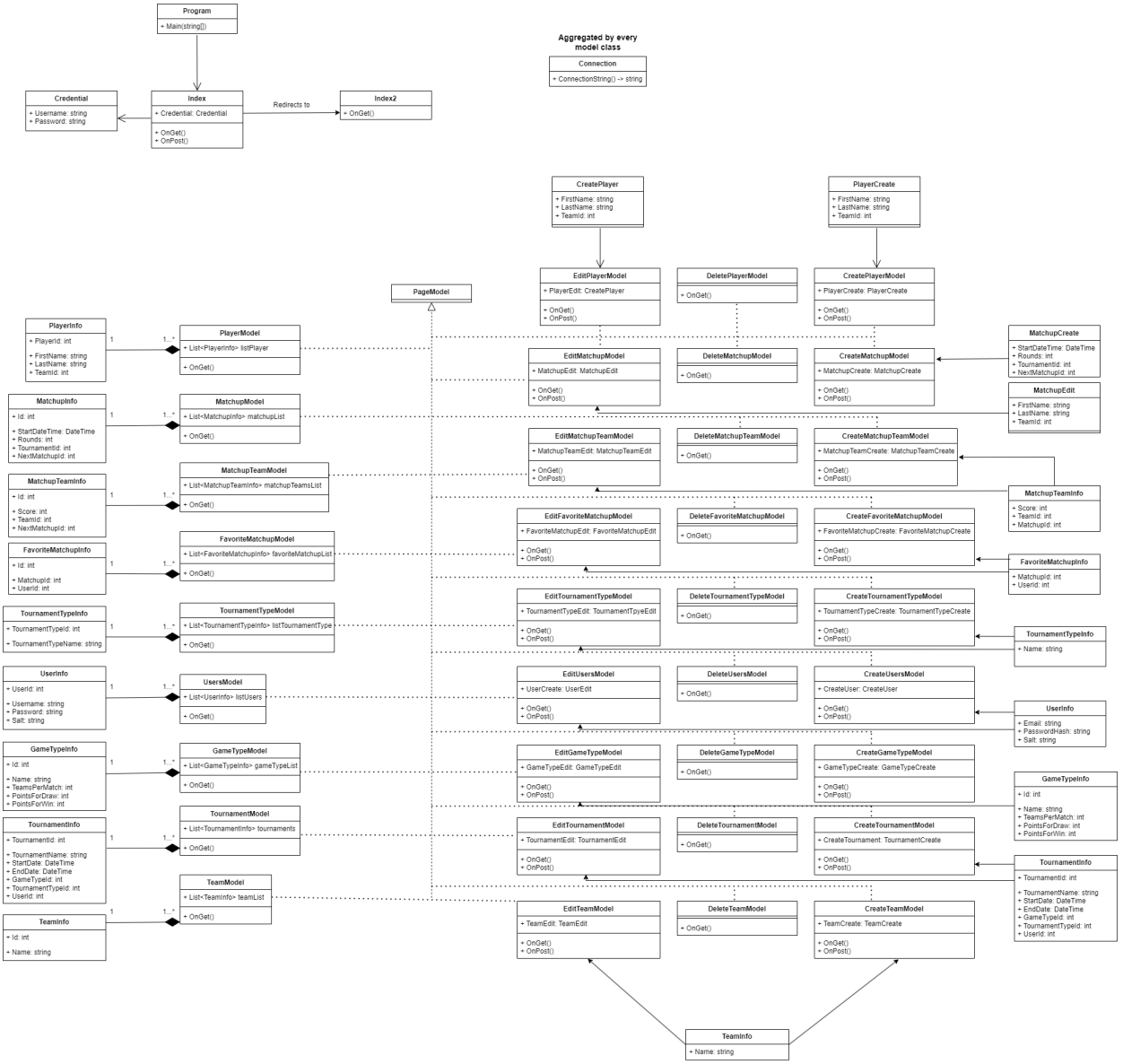
Logging/Exception handling:
Alle exceptions wrappes i en try-catch blok, hvor der i catch delen vil være et kald til en global exception handler som logger hvilke exceptions der sker, og hvornår. Så der nemt kan debugges på et senere tidspunkt.

Unit test:
C#:

- Controllers
- Repositories
- Services

Klassediagram API:

Klassediagram Admin:



Konklusion: