Chessinator
Ontwerpdocument

Siem Lucassen – S2-DB01

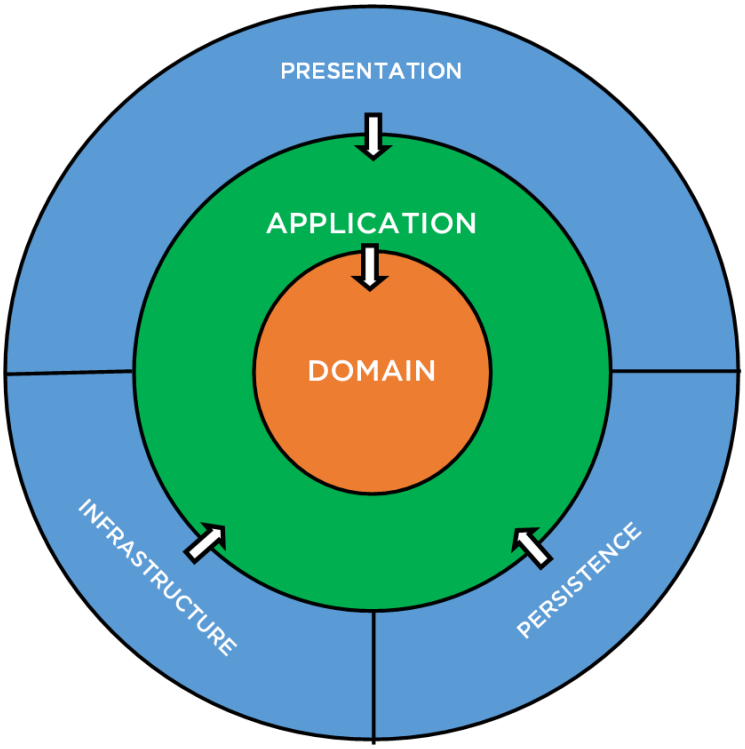
Versie	Datum	Veranderingen	Auteur
0.1	09-03-2021	Klassendiagram	Siem Lucassen
0.2	16-03-2021	Database ontwerp	Siem Lucassen
0.3	23-03-2021	Architectuurlagen	Siem Lucassen
0.4	25-03-2021	Architectuurdiagram	Siem Lucassen
1	26-03-2021	Feedback verwerken	Siem Lucassen

Inhoud

Architectuur

Architectuurlagen

In de video [1] over Clean Architecture door *Jason Taylor*, ben ik erachter gekomen hoe ik binnen mijn project de architectuurlagen kan toepassen en begrijp ik waarom het zo kan. Ook krijg ik een inzicht van experts die hun mening geven.



Domain –

Entities (Datamodellen), Value Objects, Enumerations, Logic(domeinlogica), Exceptions

Initialiseer alle collections en gebruik private setters, Creëren van custom Domain exceptions

Application –

Interfaces, Models, Logic (Servicelogica), Commands/Queries, Validators, Execptions

Ik heb gezien hoe CQRS en de library MediatR je design vereenvoudigd en zal dit onthouden voor een volgend, meer uitgebreid project.
Staat onafhankelijk van Infrastructure en Data access

Persistence –

DbContext, Migrations, Configurations, Seeding, Abstractions

EF Core heeft manieren voor unit testen zonder repositories

Onafhankelijk van de database,
Gebruik conventions over configuration,
Gebruik API configuraties over Data Annotations,
Gebruik extensie die automatisch alle entity type configureert.

Infrastructuur –

Implementations (voor buiten de applicatie), API Clients, File System, Email /SMS, System Clock, Alles wat extern is/ Communicatie buiten de applicatie…

Heeft klassen voor externe resources,
Implementeren van service interfaces van de Application laag die buiten de applicatie communiceren,
Er is geen laag die afhankelijk is van het infrastructuur laag.

Presentation –

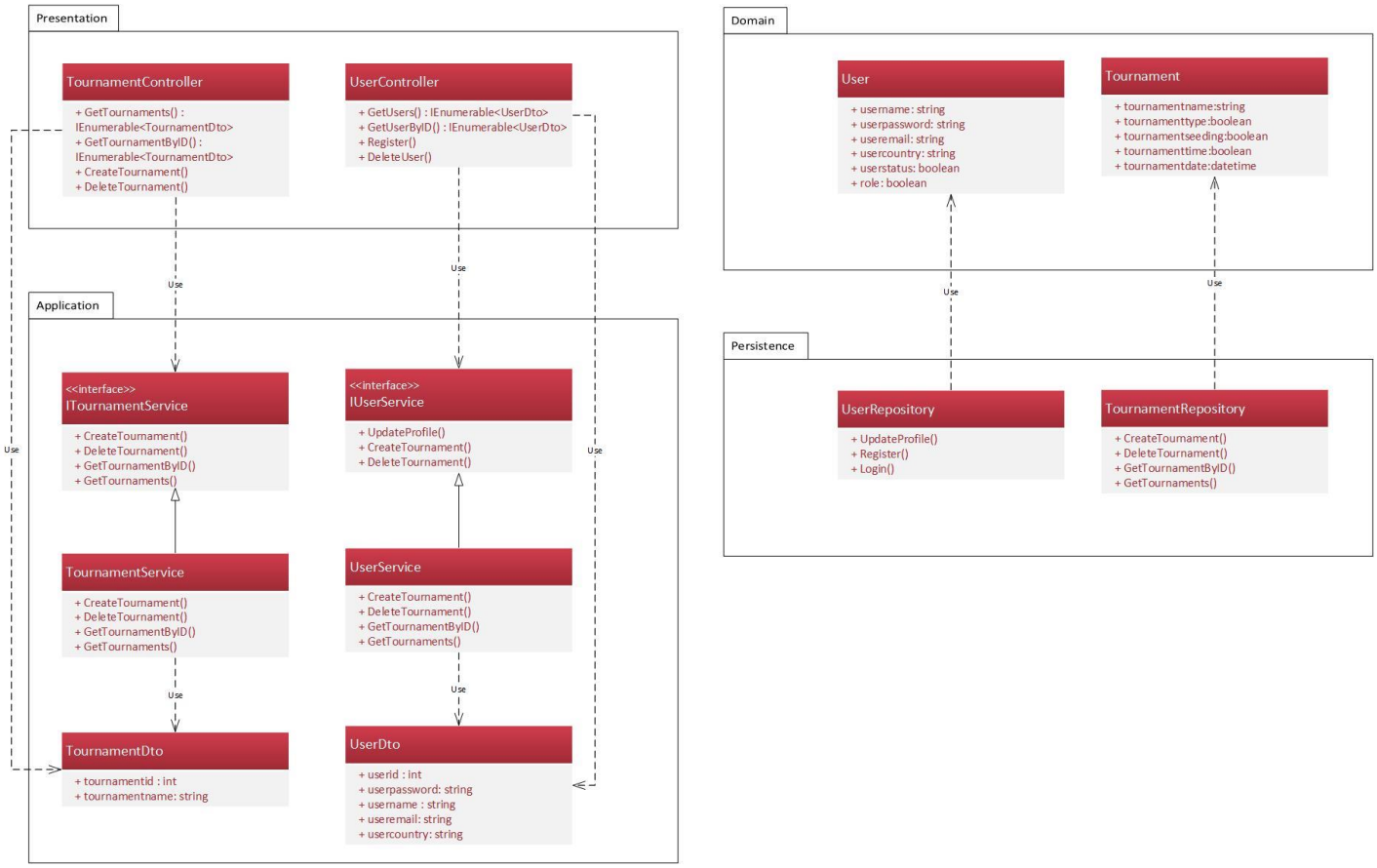
SPA – Angular of React, WEB API, Razor Pages, MVC, Web Form

Controllers moet geen Application logica bevatten,
Het creëren van gedefinieerde viewmodels,
Het gebruik maken van Open API bridges/ het besluit maken tussen frontend en backend).

Architectuurdiagram

In dit architectuurdiagram laat ik zien welke entiteiten behoren tot welke laag, dit is belangrijk omdat sommige lagen afhankelijk of juist onafhankelijk van elkaar moeten zijn. Door het houden aan de architectuurlagen, heb ik al enige SOLID principes in mijn project zitten.

**In dit diagram laat ik mijn eerste flow zien van hoe ik denk dat alleen de entiteit Tournament en User gaan lopen, aangezien ik nog geen feedback hierop heb kunnen krijgen. Ik heb geen infrastructure-laag omdat ik nog geen externe functies heb die communiceren buiten de applicatie.*

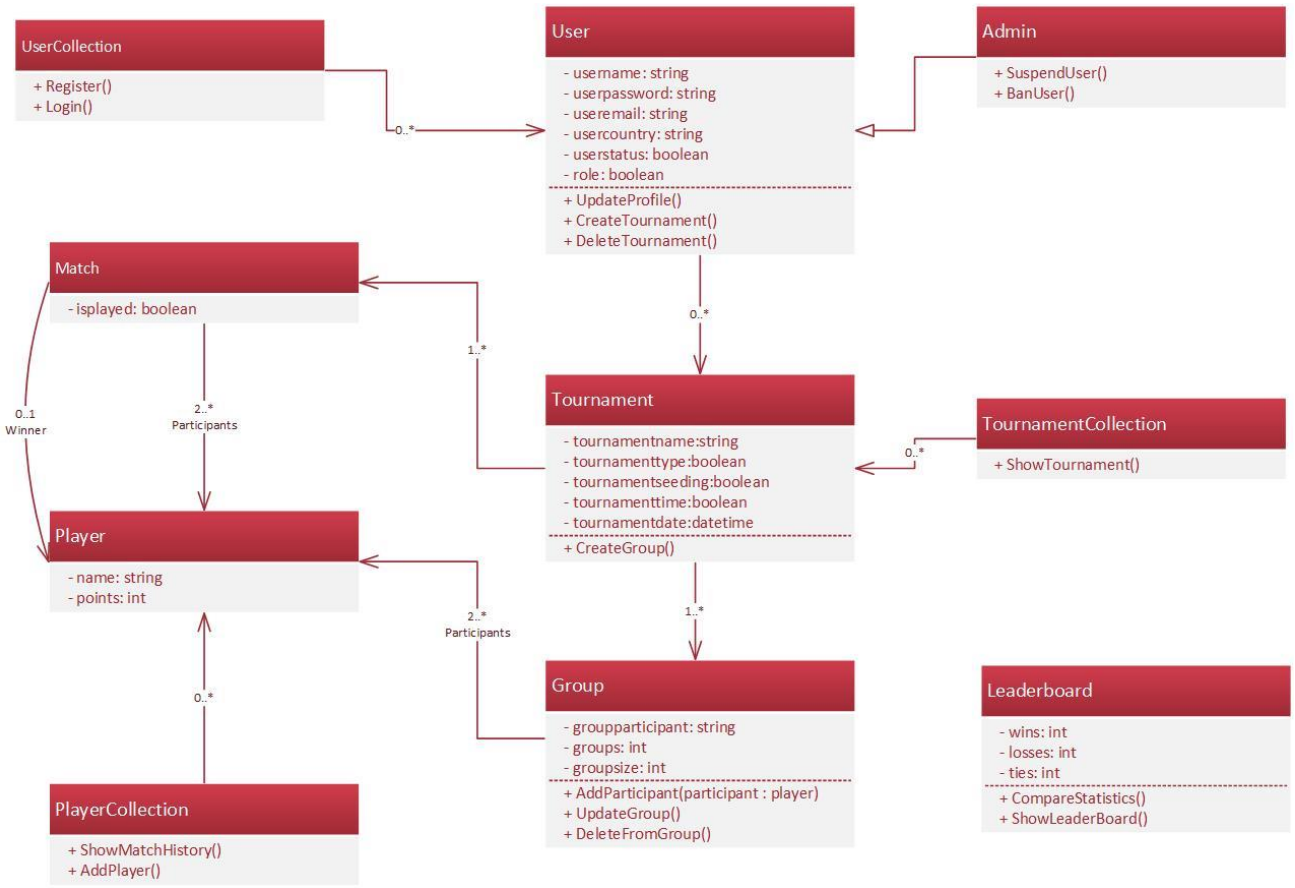


Klassendiagram

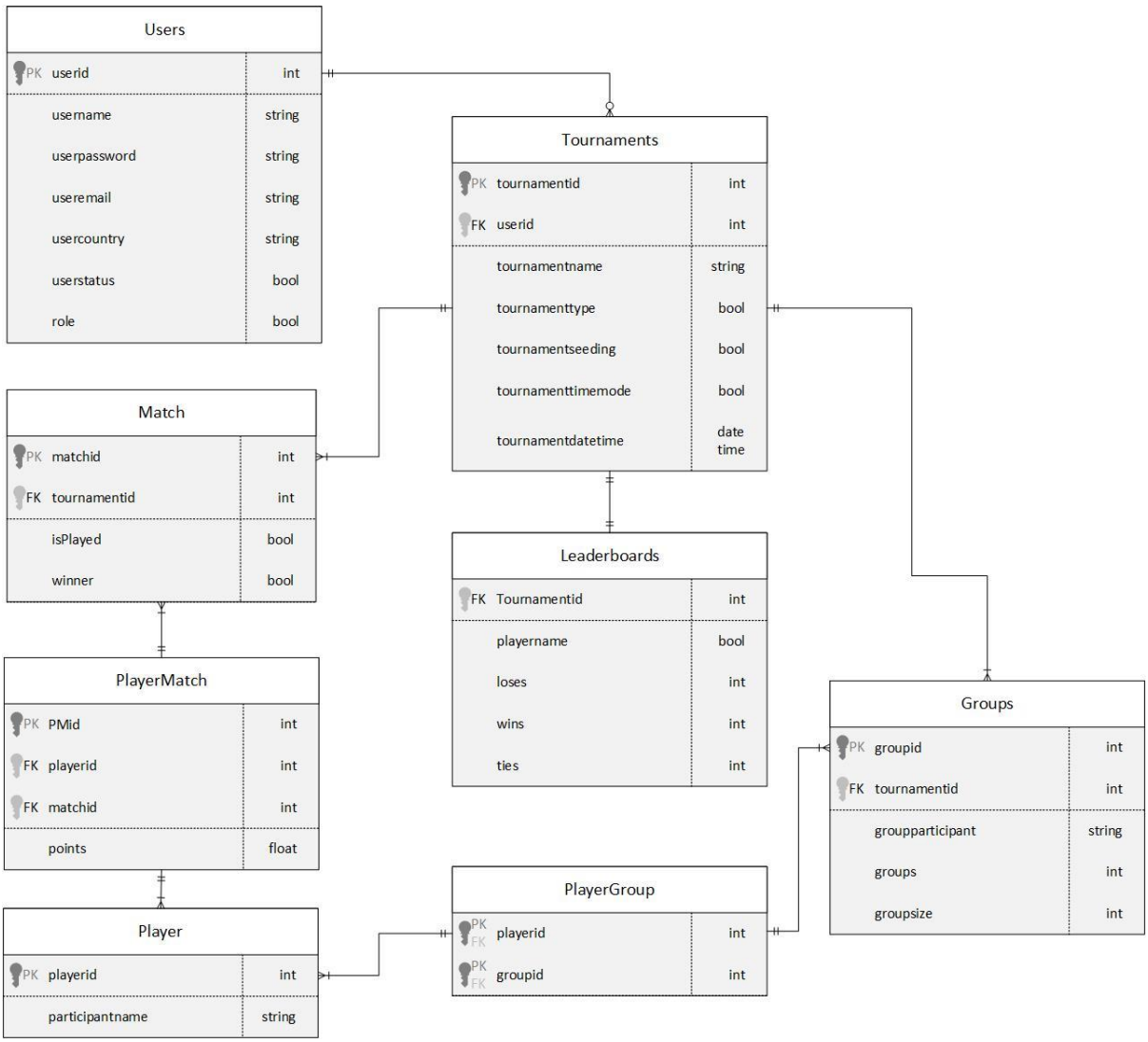
In dit klassendiagram is te zien wat de relaties zijn tussen de klassen binnen de applicatie.

De **Collection** klassen zijn voor het beheren van toegewezen de klassen.

- De **Admin** *inherit* de user om zo redundante gegevens te voorkomen.
- Een **User** kan meerdere **Tournaments** aanmaken
- Een **Tournament** heeft 1 of meerdere **Groups** of **Matches**
- Een **Groep** en **Match** hebben 2 of meer **Participants (Player)**.
- Een **Match** kan een winner hebben of een gelijkspel.



Database ontwerp



Bibliografie

[1] b. d. SSW TV | Videos for developers, „Clean Architecture with ASP.NET Core 2.1 | Jason Taylor,” Youtube.com, 19 oktober 2018. [Online]. Available: https://www.youtube.com/watch?v=_lwCVE_XgqI.