Ontwerp



Gemaakt door: Kevin Mout

Klas: DB02U.

# Architectuur(eerste instantie)

Voor het maken van mijn webapplicatie heb ik eerst 3 lagen gebruikt, omdat ik nog nooit met een architectuur heb gewerkt. Daarom wil ik het eerst zo eenvoudig mogelijk houden. Mijn architectuur bestaat daarom uit de view, logic en DAL laag.

Voor de view laag heb ik gebruik gemaakt van ASP.NET framework. Dit heb ik gekozen omdat ik dan makkelijk de data kan weergeven uit de DAL laag. In principe kan ik ook een andere User Interface gebruiken, omdat de logic laag niet afhankelijk is van de view laag.

In de logic laag komen natuurlijk al mijn klassen. Dit heb ik uitgewerkt in een klassendiagram op de volgende pagina.

In de DAL laag roep ik mijn database aan. De database is in dit geval mysql. Nogmaals de rede is om het nog zo eenvoudig mogelijk te houden. In de toekomst komt er natuurlijk nog een factory laag en interfaces waardoor het niet uitmaakt welke DAL laag en database ik gebruik.

View

Logic

DAL

# Architectuur(5 lagen)

Mijn architectuur bestaat uit deze componenten.

* De UI, ook wel de presentatie laag of view genoemd.

Dit is de connectiviteit tussen de gebruiker en de applicatie. Deze laag krijgt de gegevens door vanuit de logic laag, vertaalt dit en geeft de gegevens weer in bijvoorbeeld een web browser.

* De logic, ook wel de business laag genoemd.

Dit bevat classes, properties en functies om data via een factory en interface(s) data uit de DAL te halen.

* De Factory Design Pattern.

Het enige wat een factory doet is nieuwe objecten maken. In mijn applicatie haalt de factory, met een specifieke interface, data uit mijn DAL.

* Interfaces.

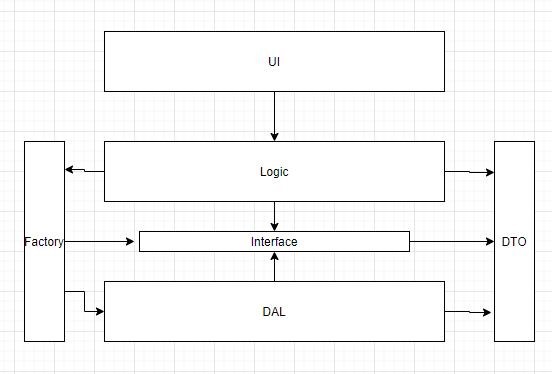
In mijn applicatie maak ik gebruik van interfaces. Dit zijn contracten of regels die kan meegeven aan bijvoorbeeld een factory. Een interface voert zelf niks uit. Hij zorgt er alleen voor dat bijvoorbeeld de juiste functies worden aangeroepen.

* DTO’s.

DTO staat voor data-transfer-object. Dit is een object dat data meeneemt tussen processen. Het verschil tussen DTO objects en logic objects, is dat een DTO behalve dingen met opslag geen functies heeft. Het is echt gewoon een containertje met data.

* De Dal, ook wel de presistentie.

In de Dal vindt de connectiviteit tussen de applicatie en gekozen data opslag plaats. Hier roep je in feite een database aan.

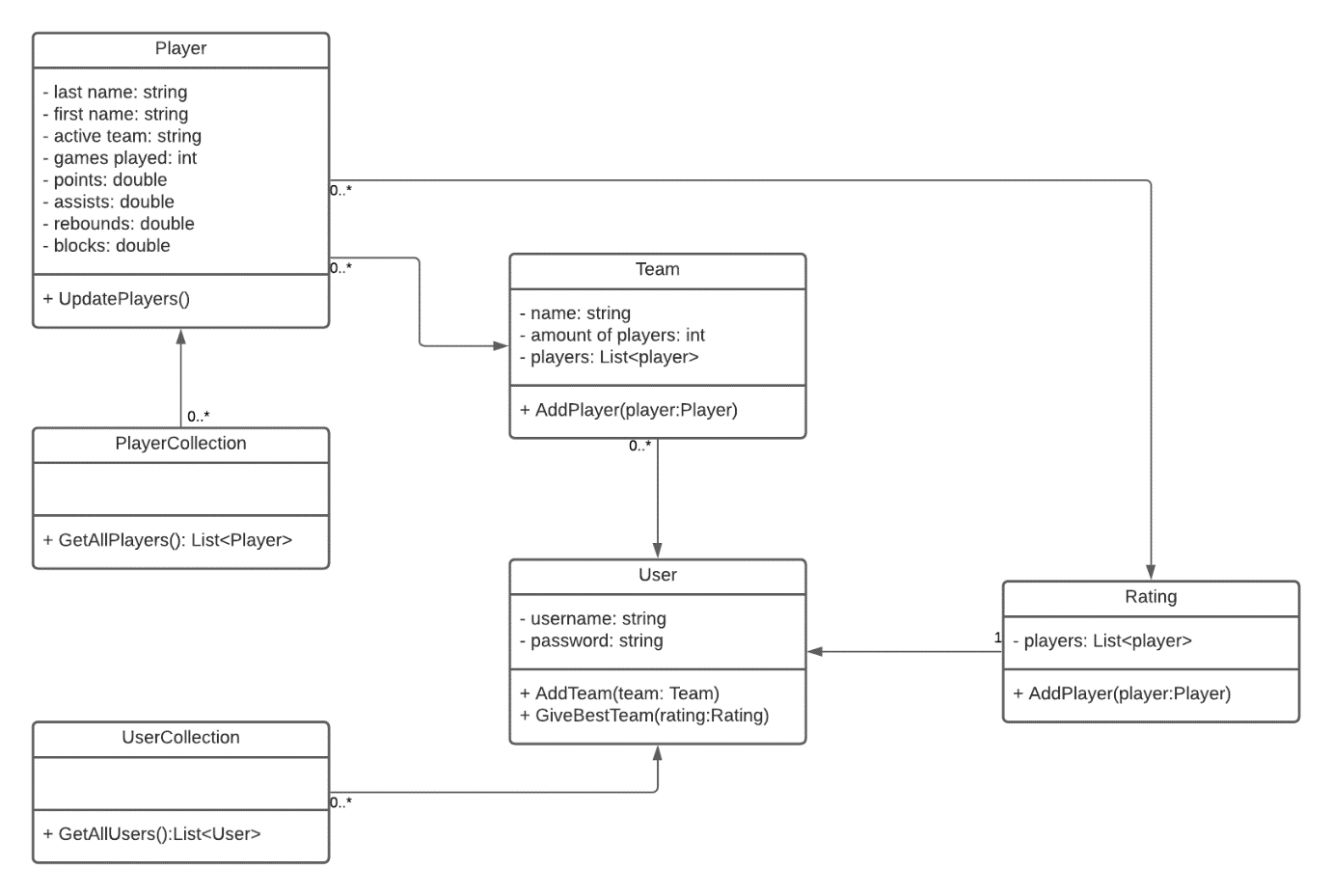
Hieronder staat een foto met de dependencies.

Deze vorm noem je dependency inversion. Het punt van dependency inversion is dat je bijvoorbeeld een andere UI erop kan zetten en dat je verder in de logic niks hoeft aan te passen. Bij een andere database hoef je alleen maar de Dal aan te passen. Dit komt dus doordat er tussen 2 belangrijke classes een interface zit.

De rede dat ik

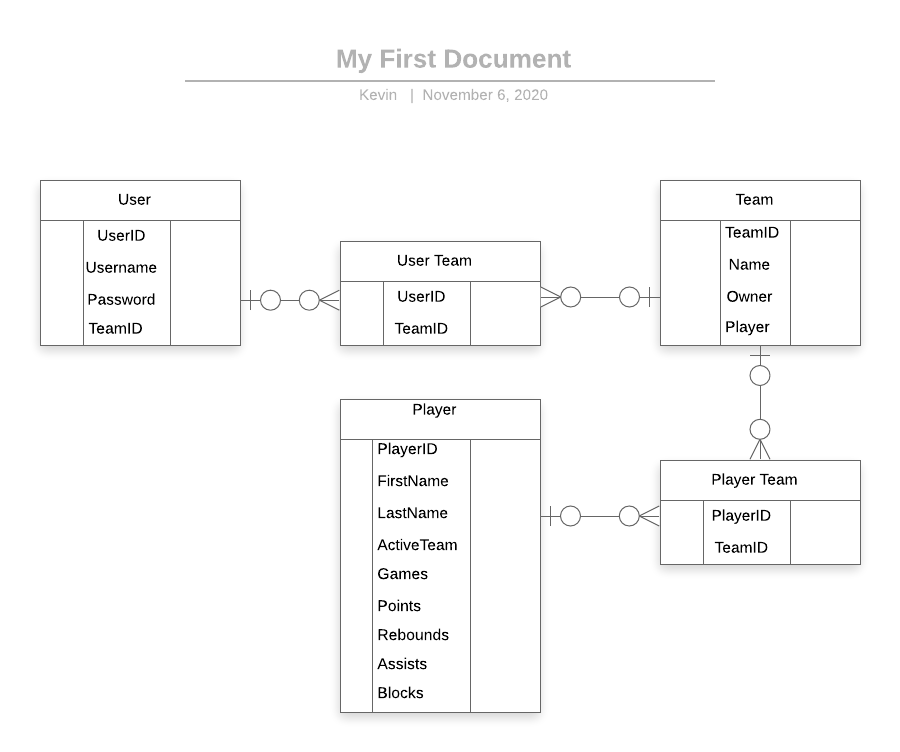
# Klassendiagram

Dit is het klassendiagram. Je hebt de playercollection waarbij ik de data van de database uit haal via de GetAllPlayers() methode. In de player class zorg ik ervoor dat de gegevens van de database geïnstantieerd worden. In de team en rating class kan je spelers toevoegen. In de user class heb je een gebruikersnaam en wachtwoord. Je kunt in deze class een team aanmaken en spelers toevoegen in de team class. Je kunt ook, wat volgens jou het beste team is, maken.



# Databaseontwerp

Dit is het databaseontwerp. Ik heb 5 tabellen nodig. User, team, player, userteam en playerteam tabel. In de user tabel sla ik alle gebruikersnamen en wachtwoorden van de desbetreffende users in op. Iedere user heeft een unieke id. Dan zeg ik dat meerdere users meerdere teams kunnen hebben. Hiervoor gebruik ik een tussen tabel user team. In de team tabel staat de gebruikersnaam van de eigenaar van het team, de naam van het team en de spelers in het team. Meerdere teams hebben meerdere spelers dus ik maak ook hier gebruik van een tussen tabel player team. In de player tabel heb je de naam, het actieve team, aantal wedstrijden, punten, rebounds, assists en blocks. Ook player heeft een unieke id nummer.



De afgelopen dagen heb ik CRUD functies toegevoegd aan mijn applicatie. Je kan een speler en een team, maken, aanpassen, verwijderen en lezen. Timo zei dat dit er een heel stuk beter uitziet dan wat ik opleverde in sprint 5. Hij had echter wel een paar opmerkingen met betrekking tot OO-principes. Bijvoorbeeld dat ik Iplayerdal dal = playerfactory.getplayerDal(); in elke methode had staan in mijn class. Je kunt in plaats daarvan zeggem private readonly Iplayerdal dal; en in de constructor zeggen dat dal = playerfactory.getplayerdal();. Dan kan je dal in alle methodes gebruiken maar dan wel alleen in die class.

De interface is eigenlijk een contract met regels. De factory zegt met die regels: haal de informatie uit de dal.