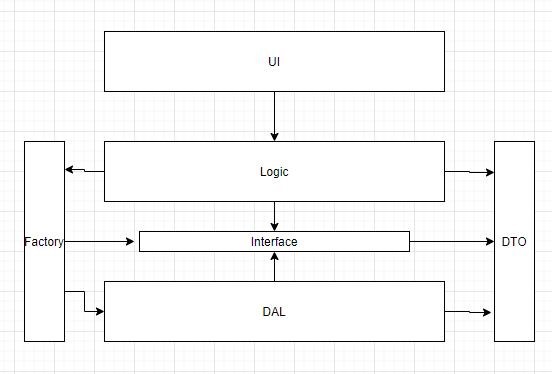
Ontwerp



Gemaakt door: Kevin Mout

Klas: DB02U.

# Architectuur(5 lagen)

Mijn architectuur bestaat uit de volgende compenenten met hun afhankelijkheden.

* De UI (user interface), ook wel de presentatie laag of view genoemd.

Dit is de verbinding tussen de gebruiker en de applicatie. Via deze UI kan de gebruiker gegevens:

* Invoeren,
* Lezen,
* Aanpassen,
* Verwijderen.

In het Engels is dat create, read, update en read (CRUD).

De UI is bijvoorbeeld een webpagina, het is een informatie vertaler voor de gebruiker van de laag genaamd logic.

* De logic, ook wel de business laag genoemd.

Deze laag is het hart van de applicatie. Deze laag vervoert de opdrachten uit van de UI, neemt logische beslissingen, evalueert en voert berekeningen uit. Het verplaatst en verwerkt ook gegevens tussen de twee omringende lagen. Deze zijn:

* UI,
* En DAL (Data acces layer).

Tussen de Dal en de logic bevinden zich de:

* Factory,
* Interface,
* DTO (data transfer object).
* De Factory, ook wel Factory Method Pattern genoemd.

De Factory wordt in de logic met een interface aangeroepen. De Factory implementeert deze regels in de 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.

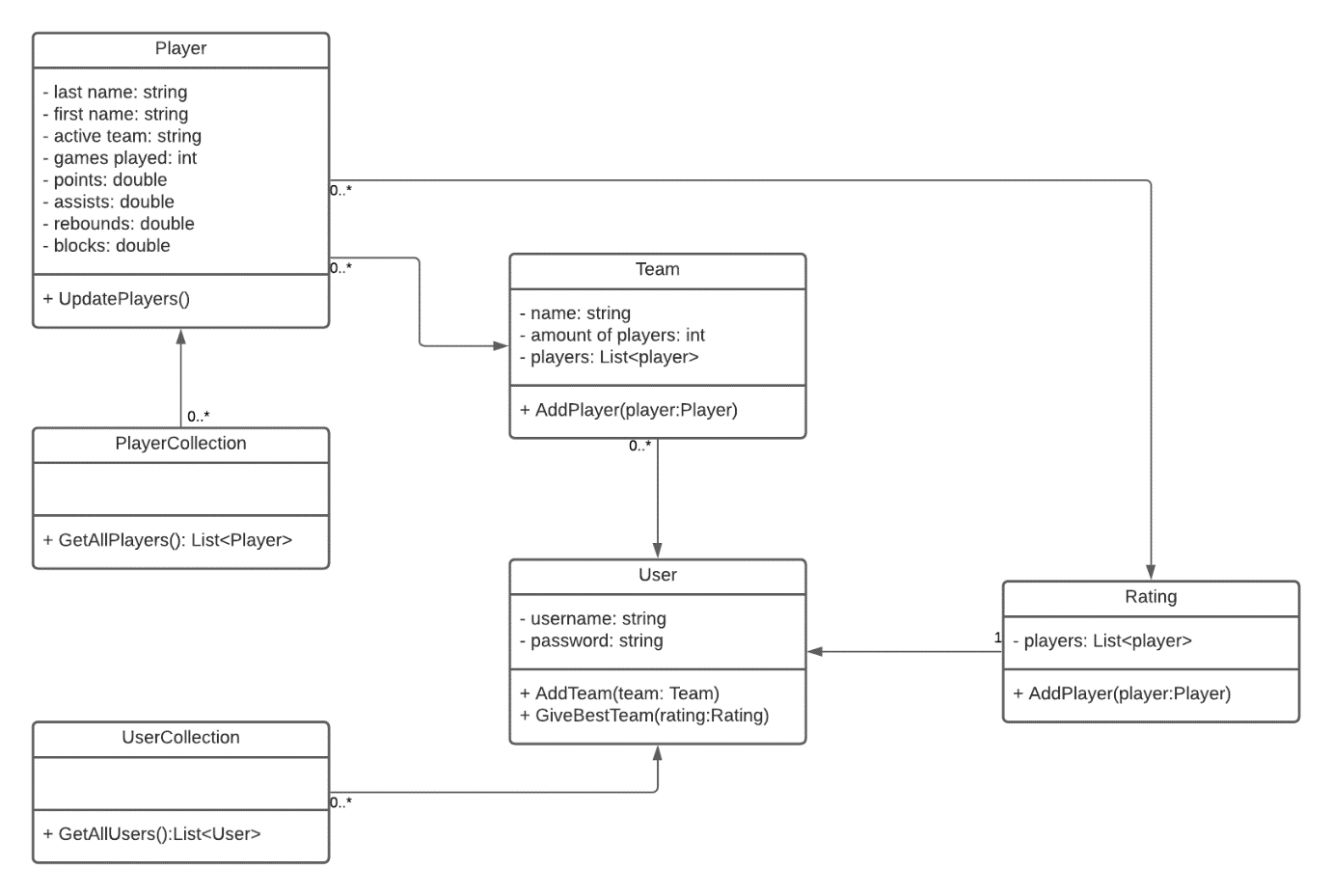
Dit is een object die data meeneemt tussen processen en opslaat. Het is “een containertje” met data.

* De Dal, ook wel de persistentie.

In de Dal vindt de connectiviteit tussen de applicatie en gekozen data opslag plaats. Hier roep je bijvoorbeeld een database aan. Een database is een digitale opslag van gegevens in een voorgedefinieerde vorm.

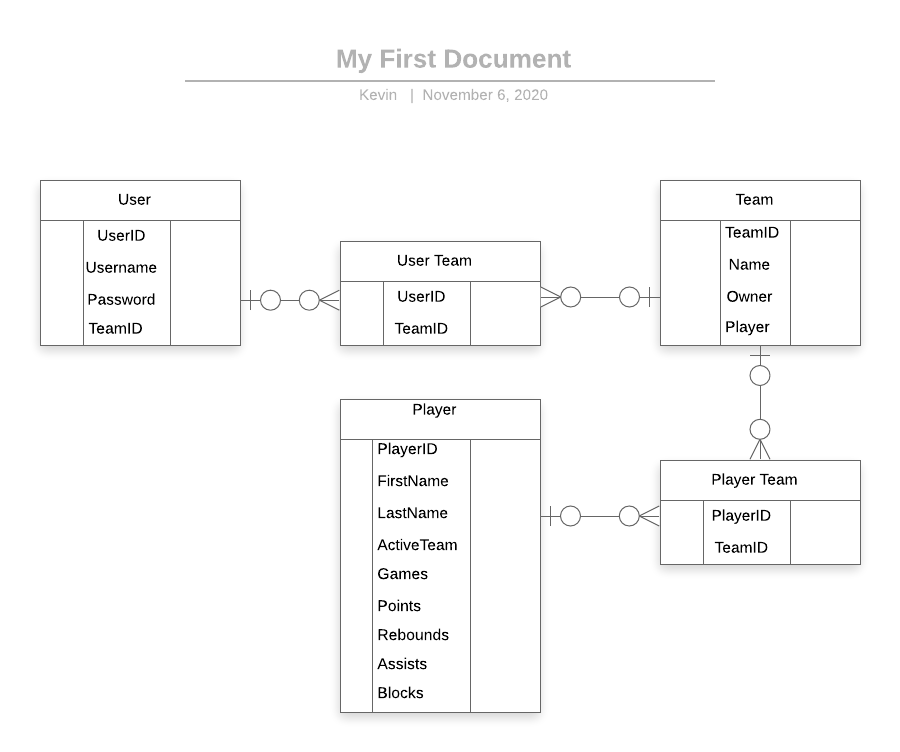
# Klassendiagram

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.