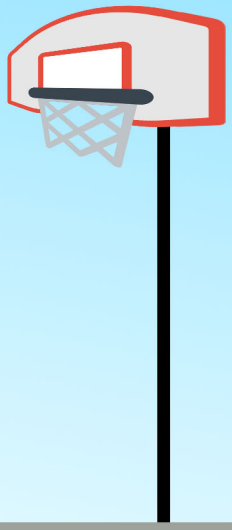
HOOP DREAMS



Hoop Dreams

A group of enthusiastic basketball fans regularly play basketball with each other. After years of always having to call everyone to play a basketball game of pickup they are beginning to find this method rather annoying. This group has come to us and want us to create a **GraphQL API** which can deliver data to make it easier to find and collect players to a basketball pickup game. This is where we come in! Because of our exceptional **GraphQL** skills and attention to detail we are the right ones for the job!

Template

The assignment comes with a very basic template which includes:

- **hoop_dreams/**
  - **index.js** (*the apollo-server setup*)
  - **errors.js** (*contains several error classes which can be imported in other modules - you may add more error classes, if you find these lacking*)
  - **package.json**

Data source

There are two data source, where we collect our data: *a web service* and *a MongoDB database*. The web service will be accessible via this url https://basketball-fields.herokuapp.com/ and the documentation for this web service can be found here: https://basketball-fields.herokuapp.com/docs. This web service will hold onto all data associated with the basketball fields. You can get all basketball fields and get a basketball field by id via HTTP. It would be recommended to use the **request** or **request-promise** NPM package to implement that functionality. The rest of the data is associated with the MongoDB database. It is your job to navigate to **MongoDb Atlas** and create a new database, setup the schema and provide the data. You are given a lot of freedom on your MongoDB database design, use it wisely! You can also choose if you are going to use **Mongoose** or the **MongoDB** NodeJS driver.

Rules (10%)

- Pickup games cannot be added to a basketball field which has a status of closed
- Pickup games cannot overlap if they are being played in the same basketball field
- Players cannot be added to pickup games that have already passed
- Players cannot be added to pickup games, if the maximum capacity has been reached for that basketball field
- Players cannot be removed from pickup games that have already passed
- A query or mutation which accepts an id as a field argument must check whether the resource with the provided id exists
- All input types which contain ids that point to certain resources should validate the ids
- Players which are registered as hosts to pickup games should automatically be added as a registered player
- If a player which is a host in a pickup game is deleted, the first (*in alphabetical order*) registered player should be assigned as the new host or if the pickup game has no registered players the pickup game should be marked as deleted
- Players cannot be registered more than once to the same pickup game
- Players cannot be registered to two pickup games that overlap
- Players cannot be removed from a pickup game they are not currently registered in
- Pickup games cannot be created with start and end date that has already passed
- Pickup games that have an end date which comes before the start date cannot be created
- Pickup games can be at max 2 hours, but a minimum of 5 minutes

If any of the rules status above are broken, an error which fits the purpose within errors.js should be thrown

Assignment description

Below is a description of the functionality the **GraphQL API** should contain. For all reference on how the schema should be setup, look at the **Schema** section below.

- Queries **(30%)**
  All queries should exclude resources which are marked as deleted
  - **(5%) allBasketballFields** - *Should return a collection of all basketball fields. Contains a field argument called **status** which is of type BasketballFieldStatus (enum) and should be used to filter the data based on the status of the basketball field*
  - **(5%) allPickupGames** - *Should return a collection of all pickup games*
  - **(5%) allPlayers** - *Should return a collection of all players*
  - **(5%) basketballField** - *Should return a specific basketball field by id*
  - **(5%) pickupGame** - *Should return a specific pickup game by id*
  - **(5%) player** - *Should return a specific player by id*

- Mutations **(35%)**
  *All field arguments to mutations should be defined as input types, see **Schema** section*
  - **(5%) createPickupGame** - *Creates a pickup game and returns the newly created pickup game matching the PickupGame type*
  - **(5%) createPlayer** - *Create a player and returns the newly created player matching the Player type*
  - *(5%)* **updatePlayer** - *Updates a player by id and returns the updated player matching the Player type*
  - **(5%) removePickupGame** - *Marks a pickup game as deleted and returns either true or an error if something happened*
  - **(5%) removePlayer** - *Marks a player as deleted and returns either true or an error if something happened*
  - **(5%) addPlayerToPickupGame** - *Adds a player to a specific pickup game and returns the pickup game matching the PickupGame type*
  - **(5%) removePlayerFromPickupGame** - *Removes a player from a specific pickup game returns either true or an error if something happened*

Folder structure (5%)

The folder structure of the project should be as the following:

- **hoop_dreams/**
  - **schema/**
    - **input/** - *Contains all definitions of input types*
    - **mutations/** - *Contains all declarations of available mutations*
    - **queries/** - *Contains all declarations of available queries*
    - **scalar/** - *Contains all definitions of scalar types*
    - **types/** - *Contains all definitions of schema types*
    - **enums/** - *Contains all definitions of enum types*
    - **index.js** - *Should export a root object which is composed of all the declared and defined types*
  - **resolvers/**
    - **basketballFieldResolver.js** - *Should resolve a subset of the GraphQL schema for the basketball field*
    - **pickupGameResolver.js** - *Should resolve a subset of the GraphQL schema for the pickup game*
    - **playerResolver.js** - *Should resolve a subset of the GraphQL schema for the player*
    - **index.js** - *Should export a root object which declares all resolving functionality*
  - **data/**
    - **db.js** - *The connection to the MongoDB database. The database should be accessible via the context (***https://www.apollographql.com/docs/apollo-server/security/authentication/#putting-user-info-on-the-context***)*

- **services/**
  - **basketballFieldService.js** - *Should contain logic to connect to the web service and return data, either by using promises or providing callbacks*
- **index.js** - *Should start the apollo server*

Schema (20%)

Here below is a description of each types within the schema:

- Types
  - BasketballField **(5%)**
    - id : ID*
    - name : string*
    - capacity : int*
    - yearOfCreation : Moment*
    - pickupGames An array of PickupGame (*where the array cannot be null nor the items within the array*)
    - status: BasketballFieldStatus*
  - PickupGame **(2.5%)**
    - id : ID*
    - start : Moment*
    - end : Moment*
    - location : BasketballField*
    - registeredPlayers : An array of Player (*where the array cannot be null nor the items within the array)*
    - host : Player*
  - Player **(2.5%)**
    - id : ID*
    - name : string*
    - playedGames : An array of PickupGame (*where the array cannot be null nor the items within the array*)
- Input types
  - PickupGameInput **(2.5%)**
    - start : Moment*
    - end : Moment*
    - basketballFieldId : string*
    - hostId : string*
  - PlayerInput **(2.5%)**
    - name : string*
  - SignupPlayerInput **(2.5%)**
    - playerId : string*
    - pickupGameId : string*
- Custom scalar types
  - Moment **(1%)**
    - Should be resolved using the NPM package **moment** and should have an Icelandic locale using the 'llll' format
- Enum types
  - BasketballFieldStatus **(1.5%)**
    - OPEN
    - CLOSED

Penalties (-10%)

There will be penalties for not removing **node_modules** from the solution before submitting. The penalty will be a subtraction of 1. For your own sake, don't forget this.

Submission

Submit a single compressed file (***.zip, *.rar**) in **Canvas**. Don't forget to exclude node_modules. If you are working in groups, please remember to comment the names of the group members (*excluding the one who is submitting the file*).