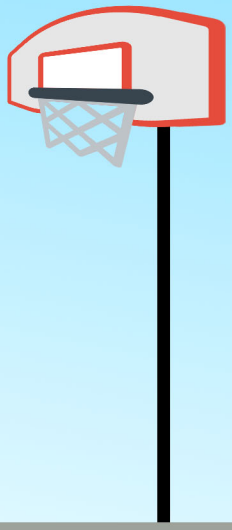
HOOP DREAMS



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, get basketball field by id and add a basketball field via HTTP. It would be recommended to use the **request** NPM package to implement that functionality. The rest of the data is associated with the MongoDB database. It is your job to navigate to **mLab** 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

- 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

  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
  - **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*
  - **allPickupGames** - *Should return a collection of all pickup games*
  - **allPlayers** - *Should return a collection of all players*
  - **basketballField** - *Should return a specific basketball field by id*
  - **pickupGame** - *Should return a specific pickup game by id*
  - **player** - *Should return a specific player by id*

- Mutations
  *All field arguments to mutations should be defined as input types, see **Schema** section*
  - **createPickupGame** - *Creates a pickup game*
  - **createPlayer** - *Create a player*
  - **updatePlayer** - *Updates a player*
  - **removePickupGame** - *Remove a pickup game*
  - **removePlayer** - *Remove a player*
  - **addPlayerToPickupGame** - *Adds a player to a specific pickup game*
  - **removePlayerFromPickupGame** - *Removes a player from a specific pickup game*

Folder structure

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/essentials/data.html#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

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

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

Penalties

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*).