Sports Exercise Battle

Table of Contents

[1 Reading the specifications and laying the groundwork (2.5h) 1](#_Toc197146771)

[2 Planning the database (2.5h) 1](#_Toc197146772)

[3 Starting to code – creating the models (3h) 2](#_Toc197146773)

[4 Setting up the database and connecting it with the application (3.5h) 3](#_Toc197146774)

# Reading the specifications and laying the groundwork (2.5h)

The first thing I did was to carefully read through the specifications in order not to miss anything and risk having to scrap my ideas. After carefully reading through the text, I brainstormed and made thoughts about how the structure of the program and database should look like as well as what to implement as the additional feature. The reason I started planning the feature early is because implementing it after being finished would be harder, since I would have to dig through my code, find ways of implementing it and end up having to alter my code in a way that would allow the feature to function. By pre planning however, I can already write my code in a way that would make implementing that feature easier.

After then Initializing my repository, I started making prototypes for the database and also created another document in which I’m storing all my ideas.

# Planning the database (2.5h)

This is one of the first things I did. After reading the specifications I got to work and used a tool in order to visualize and plan my database. I ended up having to change the tables quite a lot, since at first, I wanted to go overboard, create everything as professional as possible, create tables to make retrieving data easier, to make implementing things later easier etc. At the end however, I decided on keeping things simple and only creating the tables that I actually needed.

This ended up being the Users and history table. At first, I created tables like session or tournament, but since those were not necessary according to the specifications, I ended up removing them and only kept the previously mentioned tables.

This is what the database ended up looking like:

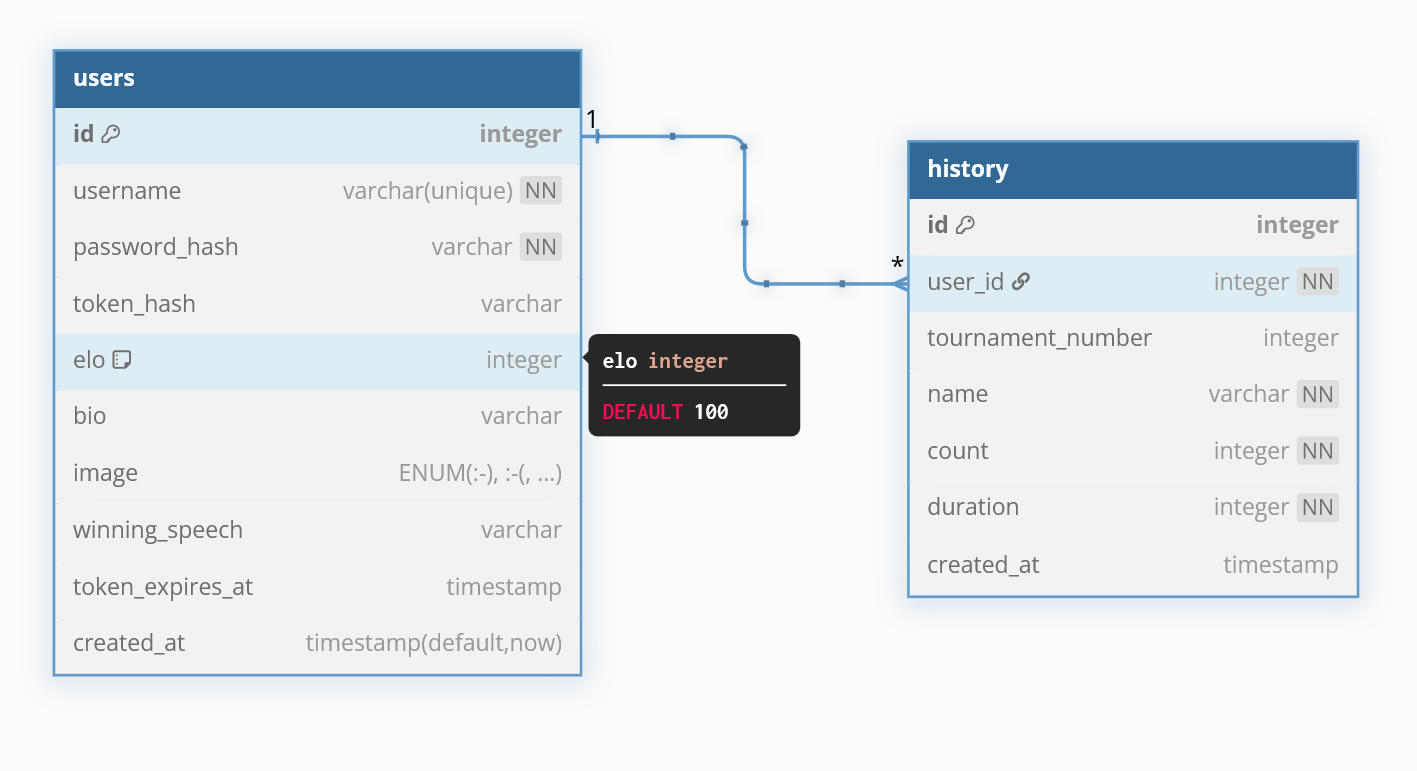


Figure 1: Vogue representation of database (https://www.dbdiagram.io/d)

# Starting to code – creating the models (3h)

My first challenge was that I haven’t coded using C# in almost 5 months, since the university I currently am studying abroad at only offers Java, the good news however, is that we have worked a lot with TCP/WebSocket’s as well as REST, hence I feel more than confident starting this project. My first action will be to quickly revise our Software Engineering course in order to get fluent in C# again.

After going over the basics, I created the models “User” and “PushUpRecord”, which contain all the data needed for the program.

These Models contain:

|  |  |
| --- | --- |
| **User** | **PushUpRecord** |
| Int Id | Int Id |
| string UserName | Int UserId |
| string PasswordHash | Int TournamentNumber |
| Int Elo | String Name |
| String Token | Int Count |
| String? Image  Enum UserImage  {  Happy, // :-)  Sad. // :-(  …  } | Int Duration |
| String? Bio |  |
| DateTime? TokenExpiresAt |  |

I also implemented functions like HashValue, which turns the Token and Password into a hash and also created the functions CreateAndSetToken, SetImage and SetPassword.

# Setting up the database and connecting it with the application (5.5h)

I used the database template I made above to create the database including the tables inside a docker container. After that I started researching, since I was not too sure on how to create a decent Database connection file. After hours of researching and refining, the finished product is able to establish a connection, close it and dispose of it, which gets rid of unmanaged resources. It also has functions to allow for CRUD operations. I forgot about CRUD when merging the branch, hence I was forced to return to the branch and create these functions before continuing with the http server.