Dev-Diary Sports Exercise Battle

## Setup

As a first step I wanted to understand the resources we were allowed to use for the project:

* Lecturer provided code for a functioning HTTP Server Demo (**\_08A3A4HttpServerDemo.sln**)
* **PostgresSQL** database run via **Docker**, accessed and modelled with **Datagrip (Jetbrains)**
* **Provided Curl Script** for Integration Tests
* **Swagger** and the MTCG- (previous project) as an API Specification Reference
* **Visual Studio** IDE

Further technologies used:

* **Draw.io** for database modeling
* **External C# libraries: Npgsql** for Unit-Tests, BCrypt.Net for Password Hashing

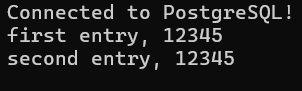
## Introduction

To have more oversight, I started with creating a flowchart to better understand the provided code **\_08A3A4HttpServerDemo.sln** which I provided in a separate file (TODO).

After installing a docker-container with a Postgres Image, I downloaded the Application Datagrip and connected it to my Postgres database. I created a new role “seb\_connection” which will be the login credentials for the SEB Datalayer. This role was provided with all privileges on the database and all tables and provided with a password. To test the connection via Datagrip, I created a test DML SQL file to create a table. Then I installed the library **Npgsql** via the nuget packet manager in Visual Studio.

## Database Access

To connect to the database via the application, I created a database connection object which creates a connection, and executes a parameterized query:

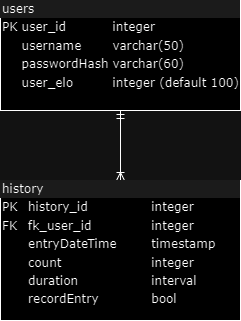


To prevent SQL Injections, all queries are executed as parameterized queries.

I modelled the users table in the database, consisting of a primary key that is automatically assigned, a username and a place to store the password. Passwords will be stored encrypted and will be encrypted in the business logic. A write into the database was also attempted.

## Database Modeling

To model the database, I used



## User creation

Next, I used the hashing algorithm provided in the Library BCrypt to hash the password. This will be used to store the passwords encrypted in the library and will also be important for the login function.

## Sessions

Using the provided curl-script as a guideline, I started to implement the endpoint “sessions”. I created a new Endpoint object and registered it in the HTTPServer object. Next, I created a Database-access- object that receives the “user”-Object loaded with all credentials as a parameter and builds a query to access the database based on the username. The verifying of the given password was very tricky, because at first, I thought I had to rehash the given password and then compare it to the one stored in the database, however this did not work, since the used method “BCrypt.HashPassword()” required a salting value. If none was provided a random value was generated. This made the compared hashes differ everytime. Reading the documentation of the BCrypt Library further, I found the correct method which was “Bcrypt.Verify()”. Using this the authentification via the given credentials worked immediately:

