# Swimming Championship

# Spring Application

This is the documentation of a Spring Application for a Men's Long Course Swimming Competition developed as part of the Web Development with Java course.

## Model

The main entities of the application are:

- Swimmer
- Event
- Session
- Race
- User
- Ticket
- Sponsor

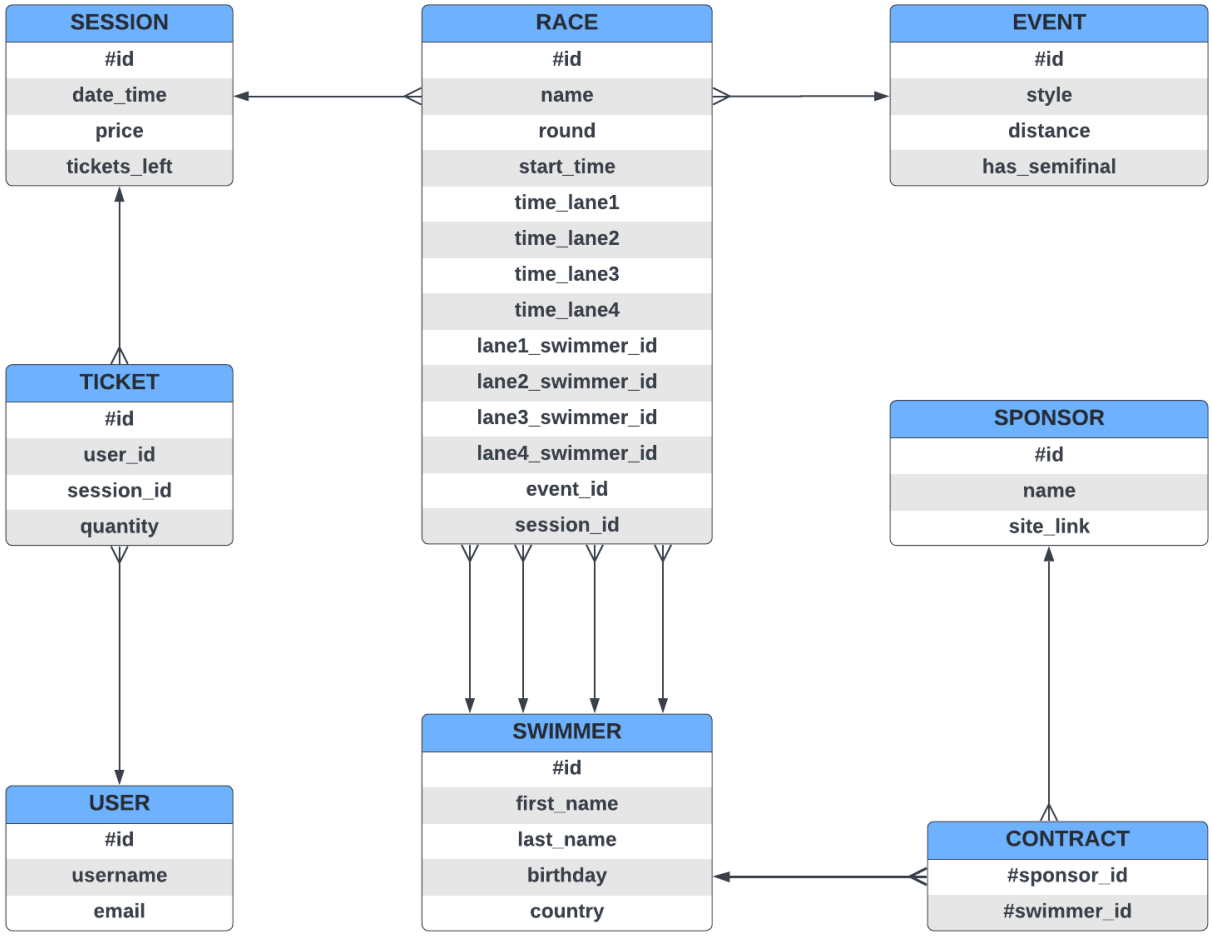
Each event has multiple races which belong to one of three possible rounds (Heat, Semifinal, Final).

Each race takes place during a session for which users of the app can buy tickets.

As this championship takes place in a 4 lane swimming pool, up to four athletes can compete in a race. Some events don't have a semifinal round due to the lower number of swimmers who registered for the qualifying rounds.

A swimmer can be sponsored by multiple companies, and a company can sponsor multiple swimmers.

By adding the fields, foreign keys and transforming the many-to-many relationship between swimmer and sponsor into an associative table we get the following conceptual diagram.




## Business requirements

The following units of business logic have been implemented and in a future development would be authorized depending on the user roles:

- add a swimmer;
- show the schedule of a session;
- show the companies together with the athletes they sponsor;
- create a new user;
- as a user, buy tickets for a session;
- show a certain user and their tickets;
- show swimmers (optionally filtered by country);
- set the times for a race;
- add a new heat (qualifying round);
- schedule a final stage (semifinal/final);

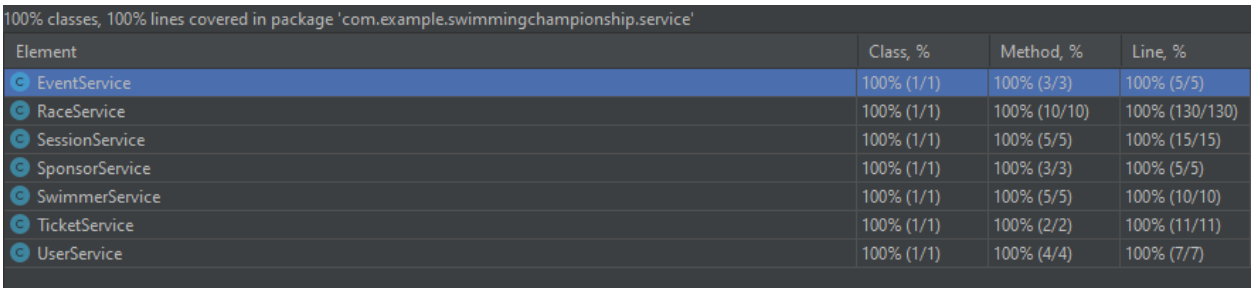
- place the swimmers who came out of the heats on the lanes in the semifinals/final (depending on whether the event has a semifinal round);
- place the swimmers who came out of the semifinals on the lanes in the final;
- replace a swimmer who withdrew from a final stage.

## Main features

- as a user, buy one or more tickets to a session;
- place the swimmers on the corresponding lanes of a final stage based on the ranking of the previous round;
- replace a swimmer who withdrew from the semifinals/final with the one having the next best time in the previous round;
- add a new heat providing the swimmers;
- schedule a final stage;
- show swimmers (optionally filtered by country);
- set the times of a race.

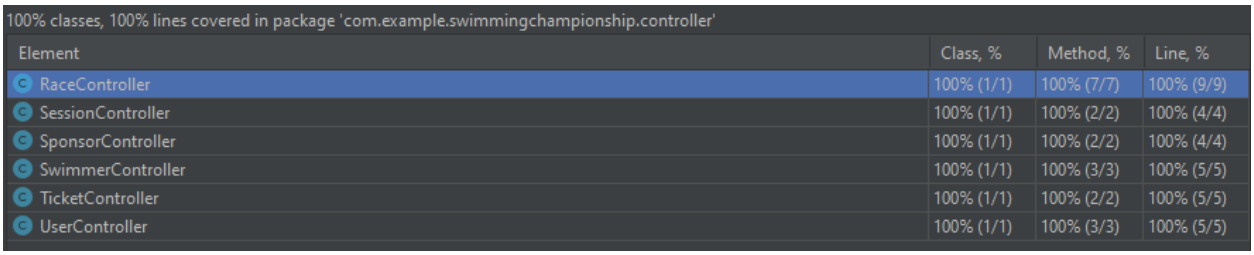
## Unit tests

With the implemented unit tests we achieved a 100% line coverage on the service and controller layers.



100% classes, 100% lines covered in package 'com.example.swimmingchampionship.service'

| Element | Class, % | Method, % | Line, % |
|---|---|---|---|
| EventService | 100% (1/1) | 100% (3/3) | 100% (5/5) |
| RaceService | 100% (1/1) | 100% (10/10) | 100% (130/130) |
| SessionService | 100% (1/1) | 100% (5/5) | 100% (15/15) |
| SponsorService | 100% (1/1) | 100% (3/3) | 100% (5/5) |
| SwimmerService | 100% (1/1) | 100% (5/5) | 100% (10/10) |
| TicketService | 100% (1/1) | 100% (2/2) | 100% (11/11) |
| UserService | 100% (1/1) | 100% (4/4) | 100% (7/7) |



100% classes, 100% lines covered in package 'com.example.swimmingchampionship.controller'

| Element | Class, % | Method, % | Line, % |
|---|---|---|---|
| RaceController | 100% (1/1) | 100% (7/7) | 100% (9/9) |
| SessionController | 100% (1/1) | 100% (2/2) | 100% (4/4) |
| SponsorController | 100% (1/1) | 100% (2/2) | 100% (4/4) |
| SwimmerController | 100% (1/1) | 100% (3/3) | 100% (5/5) |
| TicketController | 100% (1/1) | 100% (2/2) | 100% (5/5) |
| UserController | 100% (1/1) | 100% (3/3) | 100% (5/5) |

## Postman collection

To try out the application, a Postman collection can be found here.

## Swagger documentation

To view the Swagger documentation run the project and access this link, or open the .html file attached to the project.

## Source code

The source code can also be found on GitHub.