Coding Test

OBS

Introduction

The following coding test is provided to you as a step through your application to Olympic Channel development department as part of the Full Stack JS Developer.

The exercise consists on the development of a small sample application that will illustrate your skills on the different areas involved in design, coding, good practices and patterns.

Please do read the instructions carefully before beginning coding and do not hesitate to reach your point of contact in case you have any doubts or questions.

What you should have received

As part of this assignment you should have received the following files:

- OCS_CodingTest_FullStack.pdf: This document
- Resources.zip: A zip file containing some images you can use to beautify the application. Please note you're not required to use them and you can use your own resources.
- OCS_Athletes.DB: An SQLite database with sample data for athletes.

What you are expected to produce and how long you'll have

You'll have a full week (7 days) to complete the assignment once you receive it.

The test shouldn't take more than from 4 to 8 hours to develop, the week is provided so that you can adapt the completion of the test to your schedule and timetable as you like. You're of course free to dedicate as much extra time as you want. Time of development will not be measured, only quality of code and good practices will.

The result of the test should consist on:

- A backend service in node.js which provides the backend interface against the provided sqlite database which contains data for the athletes. You can produce whatever backend you like (REST, GraphQL, etc)
- A SPA (Singe-page application) using the framework of your choice which consumes the API and fulfils the requirements specified in the next section.
- Instructions on how to run your results. Note we expect the solution to work correctly in both Chrome and Firefox as well as in mobile devices in both portrait and landscape mode.
- (Optional) A react-native application which consumes the API and fulfils the requirements specified in the next section.
- (Optional) An AMP implementation for mobile using SSR

Please focus on producing production ready code, that is, code that does not only work but would be what you consider to be ready to be put in production, this includes automated tests, good styling and structure, appropriate comments, etc.

Among others, the following aspects of your solution will be valued:

- Good coding practices and patterns
- Good design of the solution
- Quality of the solution and quality assurance
- Cleanliness and responsiveness of the UI design

Extra characteristics like using source control, containers, etc would be considered extra points but are not required for the solution.

The problem.

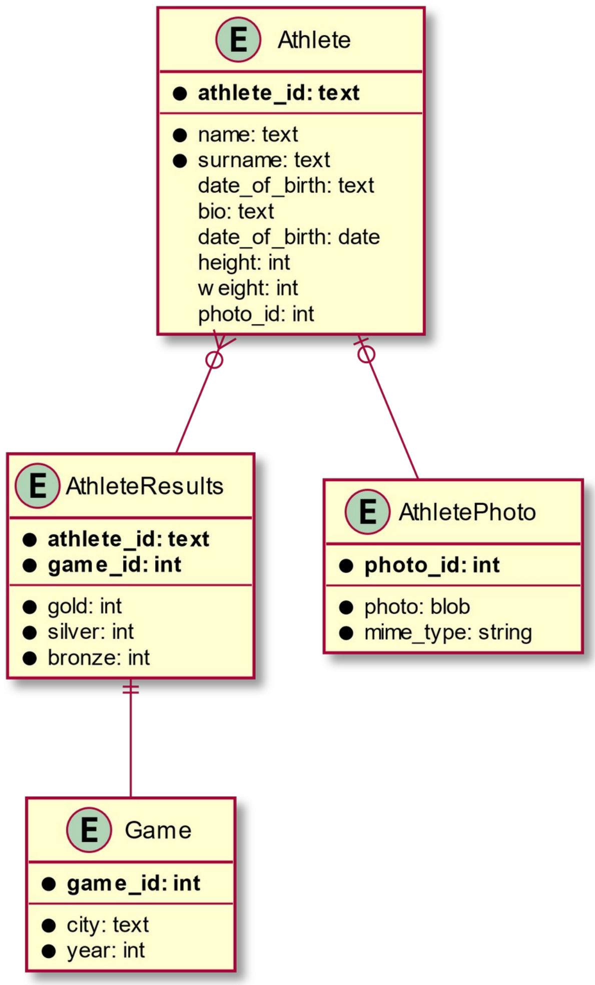
As interest grows in the Olympics, people are getting more and more interested on knowing everything there is to know about the athletes that compete on each of the games.

As such, the Olympic Channel would like to provide a new application that allow the public to get to know them through their mobile devices or through the webpage.

All the information required is stored in an SqLite database which contains all the information for the athletes. For each athlete the database contains the following information:

- Id of the athlete. The unique id of the athlete
- Name. The athlete given name
- Surname: The athlete family name
- Date of Birth. The athlete date of birth
- Height. The athlete height in cm
- Weight. The athlete weight in kg.
- Photo. The athlete archive photo. This will always have a square size.
- Bio. A long piece of markdown formatted text with the bio of the athlete
- Results. The list of results for all the different games the athlete has participated on
    - Game: London, Tokyo, Rio, etc
    - Gold. Number of gold medals won
    - Silver. Number of silver medals won.
    - Bronce. Number of bronce medals won.

The following diagram illustrates the database model:



Requirements

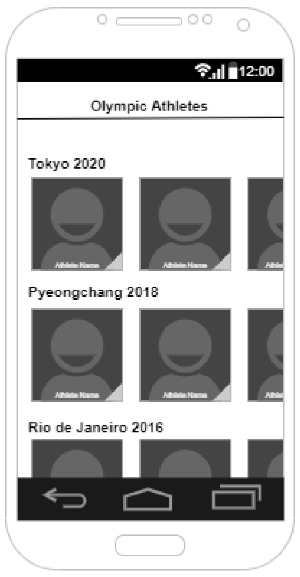
1. Upon opening the application, it shall display a list of all the athletes in the database (it can be paginated or infinite scroll) grouped by game from more recent games to older games.
2. Note an athlete may appear on several games in different orders depending on the result he/she obtained on the particular game.
3. Each "group" must be titled by the name of the game and the year of the game.
4. In each group there must be a horizontal scroll or carrousel with all the athletes ordered by global score in the current game with higher score first.
5. The global score of an athlete is calculated as follow:
    a. Gold Medal: 5 points
    b. Silver Medal: 3 points
    c. Bronze Medal: 1 point
6. The title of the application main screen shall be: "Olympic Athletes"
7. Each entry in the carrousel must contain
    a. A thumbnail of the athlete photo

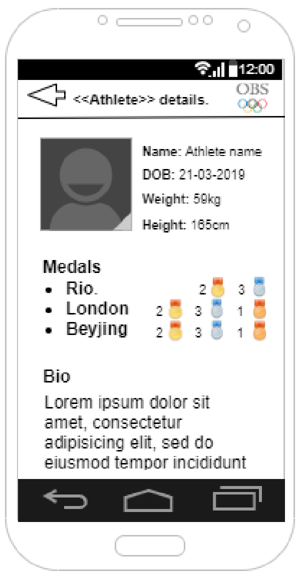
b. The athlete name and surname on the bottom part of the image

8. Upon clicking on the image of a given athlete the application will switch to a detail view of the athlete.
9. The title of the detail view shall be “<<Athlete name>> details” where athlete name corresponds to the name and surname of the athlete.
10. The detail view shall display the picture of the athlete.
11. The detail view shall display the information of the athlete (name, date of birth, weight and height) to the side of the photo.
12. The detail view shall display the list of all medals in all games the athelete has achieved.
13. The detail view shall display the markdown formatted bio of the athlete under the medals section.

For reference for the web design please see: https://www.olympicchannel.com/en/athletes (use it only as a sample, expand or modify to fulfil requirements above)

The following is a sample mock up of the design for mobile web view or application, you can use it as a reference and improve it to make it as aesthetically pleasing as possible.





How to deliver the results

In order to deliver the results please answer to the email with either:

- The solution attached, including instruction on how to execute and test it. Please make sure you don’t include elements susceptible of being intercepted by the antivirus (i.e. executable binaries). Ideally the solution should be able to be run in the simulator by npm run or npm start.
- A link to a private GitHub repository (or similar) with enough information in it to execute and test your solution (i.e. via de usual readme.md format on GitHub)