Sample of work - Backend - Rock Paper Scissors

Problem description

Build a REST API that lets developers resolve their differences using the game Rock, Paper, Scissors. The rules are simple, the best of 1 match wins.

In general

Your solution should be prepared for production. If there are areas of improvement - i.e. error handling or something similar please document that in the readme.

We want you to deliver your solution with clear instructions on how to build, install and run it.

Consider your work a proof of concept where Cygni is the customer that you are currently consulting at. Your solution should be the starting point for a larger project that others should be able to continue working on. Your selection of tech (language, frameworks and libraries) should be motivated within the context of the assignment and this might be discussed on a later technical interview.

Your solution will, upon delivery, be reviewed by two of our consultants. They will not have any other information about you as a person besides your experience level. Therefore we ask you to not include any personal identifiers within your code base.

Requirements

General

- The code should follow best practices
- The code should be maintainable

- A README must be included and contain examples of how the application is built, installed, run and used
- The implementation must be written on one of the specified platforms below.

The API

- Expose a REST endpoint that responds with JSON.
- A game that ends in a tie is a completed game. This means that the game does not need to be restarted in the event of a tie.
- No persistence mechanism is allowed. The whole state must be retained in memory. No database is allowed and all data should be lost when the server restarts.

Restrictions

In order to limit the amount of work required, the following restrictions are introduced:

- The application should not support rematches.
- There should not be any client implementation (like a CLI, GUI or web client).

Example of a game flow

1. Player 1 sends a request to create a new game and gets a game ID back from the server.
2. Player 1 sends the ID to player 2 via any communication channel (e.g., mail, slack or fax).
3. Player 2 joins the game using the ID.
4. Player 1 makes a move (Rock).
5. Player 2 makes a move (Scissors).
6. Player 1 checks the state of the game and checks who won.
7. Player 2 checks the state of the game and checks who won.

Example API design

Below is an example of which endpoints the API **COULD** expose to a client. Remember to consider what HTTP verb is suitable for each situation:

```
[POST | PUT | PATCH] /api/games
[POST | PUT | PATCH] /api/games/{id}/join
[POST | PUT | PATCH] /api/games/{id}/move
GET /api/games/{id}
```

where the id is the identifier for a specific game.

`GET /api/games/{id}`

Returns the current state of a given game with included attributes. Think about what attributes to display and when.

```
{
    "id": "some-game-id",
    ... other game attributes ...
}
```

`[POST | PUT | PATCH] /api/games`

Creates a new game. Enter player name in the request-body:

```
{
  "name": "Lisa"
}
```

`[POST | PUT | PATCH] /api/games/{id}/join`

Connects to a game with a given ID. Enter player name in the request-body:

```
{
  "name": "Pelle"
}
```

`[POST | PUT | PATCH] /api/games/{id}/move`

Make a move. Enter name and move in the request-body:

```
{
  "name": "Lisa",
  "move": "Rock"
}
```

Language and Platforms

Java

- If you are building a Java solution, Apache Maven or Gradle should be used to build and package the API.
- You should be able to start the API directly from Maven/Gradle (for example by using Jetty and `mvn jetty:run`) or as an executable jar via for example Spring Boot.
- Examples of possible frameworks: SpringMVC (Spring Boot), Dropwizard, Jersey.
- Pack the solution *without* any build artifacts in a zip file

C# - .NET

- If you are building a `.NET` solution please use a currently supported version of .NET
- Pack the solution *without* `bin/` and `obj/` directories in a zip file.

Js/Ts - Node.js

- If you are building a `Node.js` solution, it should be possible to start it via `npm` or `yarn` for example `yarn start` or `npm run start`.
- If a specific node version is needed please document that
- Feel free to look at frameworks such as Hapi, Koa or Express.
- Pack the solution *without* `node_modules` in a zip file.