Hemuppgift - Backend - Rock Paper Scissors (Sten sax påse)

Problembeskrivning

Skapa ett enkelt HTTP-API som låter utvecklare lösa sina meningsskiljaktigheter med hjälp av spelet Sten, sax, påse. Reglerna är enkla, bäst av 1 match vinner. Vi rekommenderar er att följa principerna för REST men avsteg kan göras om man kan motivera det på ett bra sätt.

Krav

- Ett oavgjort resultat räknas som ett resultat, det innebär att spelet inte behöver startas om vid oavgjort.
- Ingen persisteringsmekanism är tillåten. Hela tillståndet (statet) ska hållas i minnet.
- README ska finnas och innehålla exempel på hur applikationen kan köras via något lämpligt verktyg, tex:
    - curl
    - wget
    - Postman
    - HTTPie
- Implementationen ska vara skriven på någon av de angivna plattformarna.

I övrigt förväntar vi oss att implementationen är förvaltningsbar och följer "best practices".

Begränsningar

För att begränsa arbetsinsatsen som krävs så införs följande begränsningar:

- Applikationen behöver inte stödja returmatcher.
- Det behöver inte finnas någon klient-implementation. (Typ CLI, GUI).

Exempel på ett spelflöde

1. Spelare 1 skickar ett request för att skapa ett nytt spel och får tillbaka ett spel-ID från servern.

2. Spelare 1 skickar ID till spelare 2 via valfri kommunikationskanal. (t.ex., mail, slack eller telefax)
3. Spelare 2 ansluter sig till spelet med hjälp av ID.
4. Spelare 1 gör sitt drag (Sten).
5. Spelare 2 gör sitt drag (Sax).
6. Spelare 1 kollar tillståndet för spelet och upptäcker att hen vann.
7. Spelare 2 kollar tillståndet för spelet och upptäcker att hen förlorade.

Exempel API-design

Nedan visas ett exempel på vilka endpoints som API:et skulle kunna exponera för en klient:

```
POST /api/games
POST /api/games/{id}/join
POST /api/games/{id}/move
GET /api/games/{id}
```

där id kan vara ett UUID.

`GET /api/games/{id}`

Returnerar ett givet spels nuvarande tillstånd med ingående attribut. Tänk på vilka attribut som ska visas för vem och när.

`POST /api/games`

Skapar ett nytt spel. Ange spelarnamn i request-body:

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

`POST /api/games/{id}/join`

Ansluter till ett spel med givet ID. Ange spelarnamn i request-body:

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

`POST api/games/{id}/move`

Gör ett drag. Ange namn och drag i request-body:

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

Plattformar

Java

- Om du bygger en `Java`-lösning ska Apache Maven eller Gradle användas för att bygga och paketera API:et.
- API:et ska kunna startas direkt från Maven/Gradle (exempelvis genom att Jetty och `mvn jetty:run`) eller som en executable jar via exempelvis Spring Boot.
- Exempel på möjliga ramverk: SpringMVC (Spring Boot), Dropwizard, Jersey

.NET

- Om du bygger en `.NET`-lösning ska antingen .NET Core CLI 2.x eller senare eller en Visual Studio 2017-solution användas.
- Använder du en Visual Studio 2017-solution ska lösningen kunna startas direkt från Visual Studio.
- Använd gärna ASP.NET Core.
- Paketera lösningen *utan* `bin/` och `obj/`-kataloger i en zip-fil.

JavaScript - Node.js

- Om du bygger en `Node.js`-lösning skall den gå att starta via `npm` eller `yarn` genom exempelvis `yarn start` eller `npm run start`.
- Kika gärna på ramverk som Hapi, Koa eller Express
- Paketera lösningen *utan* `node_modules` i en zip-fil.