Test exercise: cats & dogs

The goal of this test is to create simple voting application. Voting app contains 2 functions: answering question “Who do you like most?” Cats or Dogs and seeing the results who is winning and with how many votes. UI should look clean and nice.

Despite of the fact that task is simple, the architecture of the application is microservices oriented. Architecture of the application is shown in the Fig 1. Technological stack is described below.

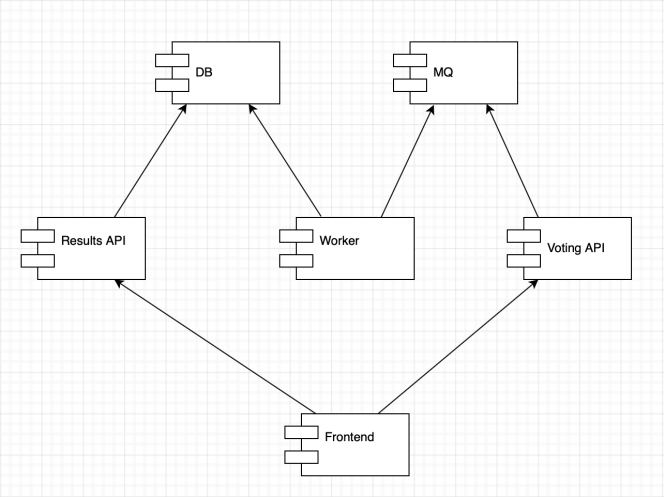

Figure 1. Component diagram of the application

1.1 Application technological stack

- Frontend should be implemented in Angular or React. Component is responsible of the user interface of the both functions.
- Voting API should be implemented in Spring Boot. Component is responsible of the voting part of the app. Interface for communicating to frontend should be REST/JSON. When vote is submitted it is passed to message queue.
- Results API should be implemented in NodeJS. Component is responsible of the results part of the app. Interface for communicating to frontend should be WebSockets.
- Worker is agent that subscribes to message queue and persists voting data to database. Component should be implemented in Spring Boot.
- DB is freely chosen database.
- MQ is ActiveMQ or RabbitMQ.

1.2 Infrastructure requirements

- Custom components should have their own GIT repository.
- For GIT repository GitHub is preferred.
- Every component should run in Docker.
- Reuse existing images from DockerHub.
- Whole application can be started with docker-compose.
- Database should create initial schema with Flyway.

1.3 Results

For results please provide GIT repositories.