UNIVERSIDADE FEDERAL DE SÃO CARLOS

Centro de Ciências Exatas e Tecnologia

Departamento de Computação

Bacharelado em Ciência da Computação

DevOps

Manual T1

Professor:

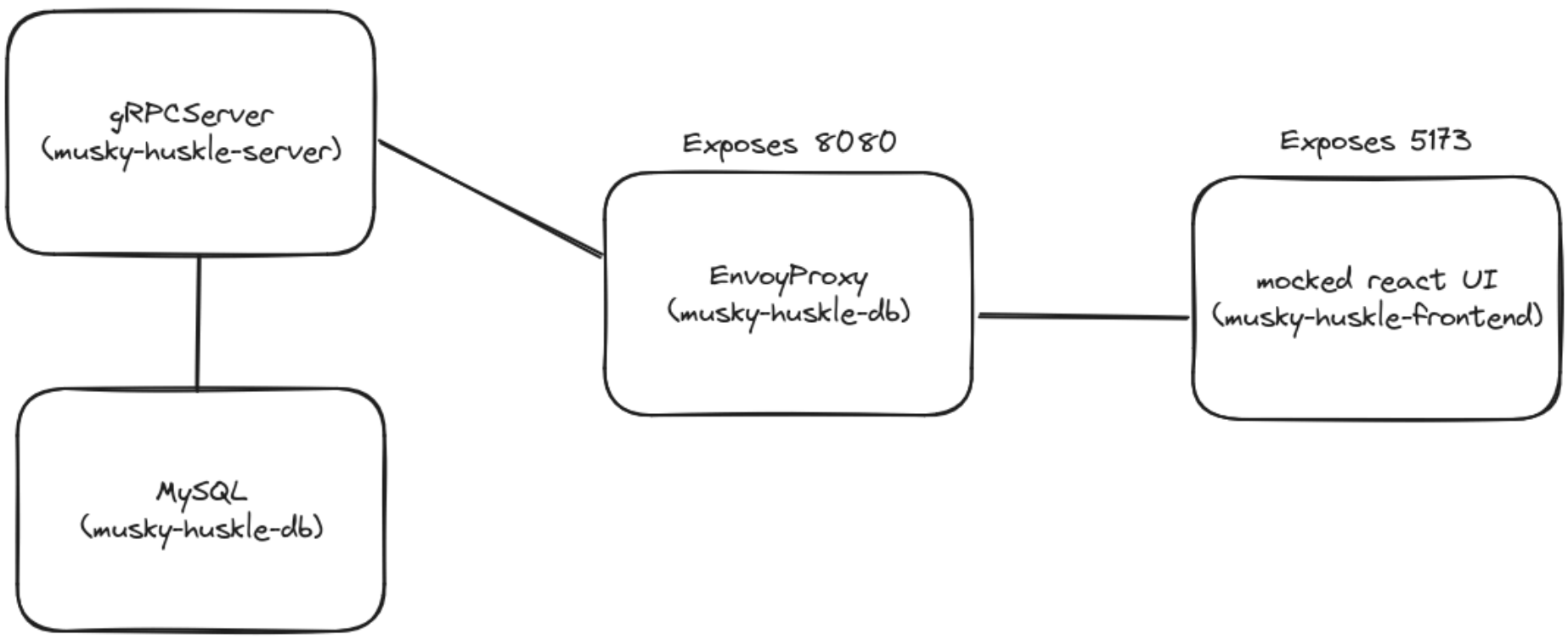
Delano Beder

Integrantes da equipe:

Daniel Kenichi Tiago Tateishi - RA: 790837

Rodrigo Pavão Coffani Nunes - RA: 800345

1. Arquitetura do projeto



2. Sobre a aplicação

O muskyhuskle é uma aplicação de um jogo simples baseado em jogos como Loldle, Wordle e Termooo. A aplicação foi criada para o aprendizado prático da utilização de tecnologias como docker e gRPC. O jogo é composto de 4 componentes:

1. Uma interface criada em react (ainda apenas um mock-up, sem interação com o backend);
2. Um backend em gRPC responsável pela lógica do jogo e pelo CRUD de membros,;
3. Um banco de dados MySQL; e
4. Um Envoy Proxy para encaminhar chamadas HTTP 1.1 do frontend para o backend em gRPC (que apenas aceita chamadas HTTP 2).

   OBS: O Frontend está apenas mockado, então o backend não está acessível por ele. Dessa forma, está sendo exposta a porta do Envoy Proxy para serem realizadas as chamadas gRPC diretamente por ele para interação com o backend e o banco de dados.

3. Iniciando a aplicação

Clonando o repositório

- Comando: `git clone https://github.com/parmenashp/musky-huskle.git`
- Resultado esperado:

Configurando ambiente

- Entre na nova pasta: `cd musky-huskle`
- Utilizando o template `.env.template`, crie um arquivo `.env` , ele será o responsável pela configuração das variáveis de ambiente utilizadas na aplicação.
- Rode o comando: `cp .env.template .env`

Subindo containers:

- Rode o comando: `docker compose up -d`
- Resultado esperado:

```
 => => exporting layers
 => => writing image sha256:5b61a267a2fff41ebed87cf60e60d6d5cc0bddbb4f1ec716957727e47a20560c
 => => naming to docker.io/library/musky-huskle-backend
[+] Running 5/5
 ✔ Network musky-huskle_default      Created
 ✔ Container musky-huskle-db         Started
 ✔ Container musky-huskle-proxy      Started
 ✔ Container musky-huskle-interface  Started
 ✔ Container musky-huskle-server     Started
```

-

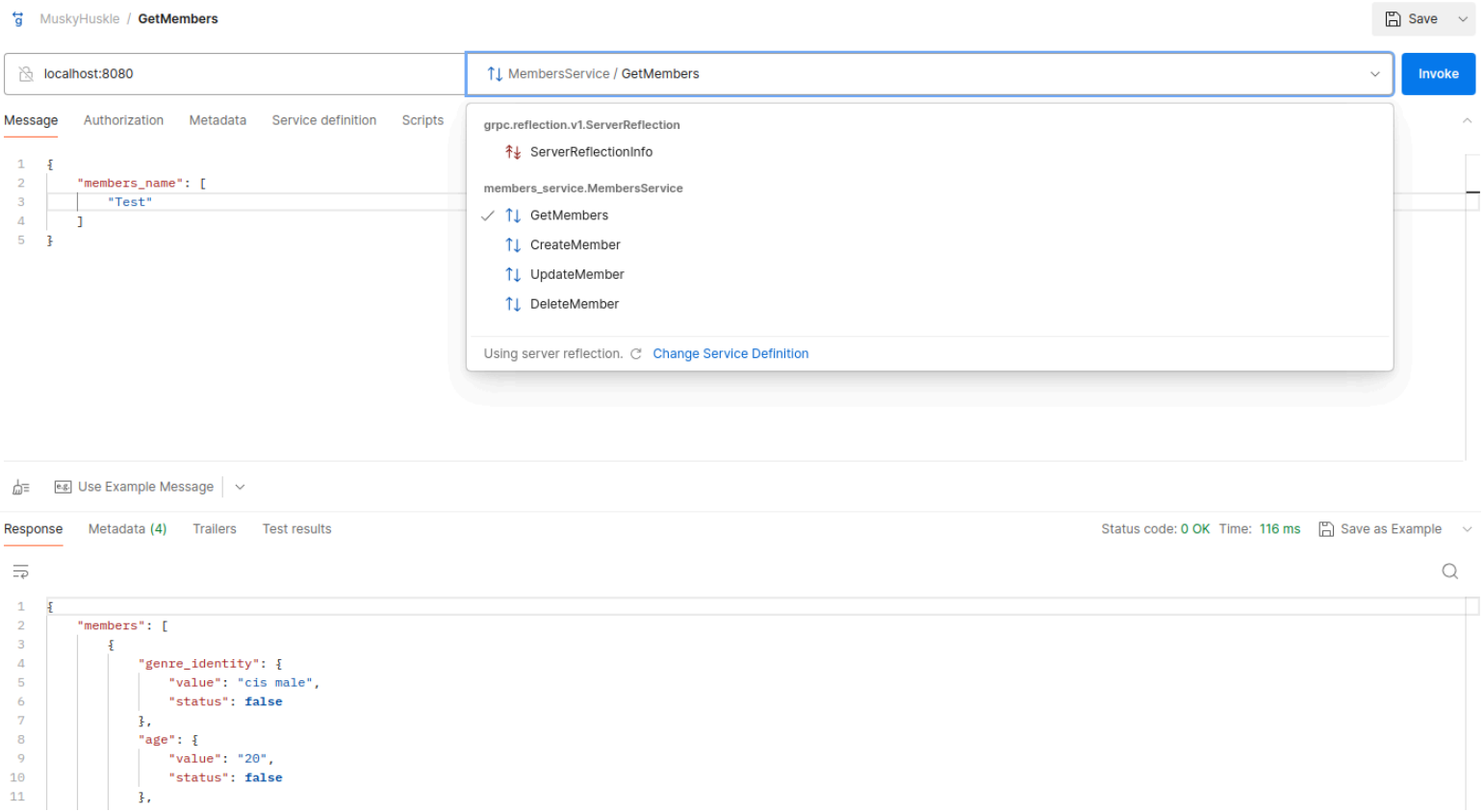
Verificando inicialização correta dos containers:

- Comando: `docker ps`
- Resultado esperado:

```
docker ps
CONTAINER ID   IMAGE                          COMMAND                  CREATED         STATUS         PORTS                                                                NAMES
09cfe052b420   musky-huskle-backend           "/go/bin/muskyhuskle"    2 minutes ago   Up 2 minutes   0.0.0.0:9621->9621/tcp, :::9621->9621/tcp                            musky-huskle-server
f2cbc5096498   musky-huskle-frontend          "docker-entrypoint.s…"   2 minutes ago   Up 2 minutes   0.0.0.0:5173->5173/tcp, :::5173->5173/tcp                            musky-huskle-interface
695b021330d7   mysql:latest                   "docker-entrypoint.s…"   2 minutes ago   Up 2 minutes   0.0.0.0:3306->3306/tcp, :::3306->3306/tcp, 33060/tcp                 musky-huskle-db
d3ffe12a56cb   envoyproxy/envoy:v1.30-latest  "/docker-entrypoint.…"   2 minutes ago   Up 2 minutes   0.0.0.0:8080->8080/tcp, :::8080->8080/tcp, 10000/tcp                 musky-huskle-proxy
```

Testando interação entre os containers:

- O container do frontend ainda não está interagindo com os outros containers por ter apenas uma interface mock-up. A interação ocorre somente entre os outros 3 containers.
- Para testar a aplicação, será necessário a utilização de um API platform como o Postman ou o Insomnia, uma vez que o frontend ainda não se comunica com o backend.
- Crie uma request gRPC para `‘localhost:8080’` na sua API platform, e ative o *server reflection*, que disponibilizará os métodos servidos pela API do gRPC server.
- Gere uma *sample message* para o método `CreateMembers` e execute a chamada. Note que a chamada deve retornar um `OK 200`.
- Para validar a criação do membro de teste no banco de dados, pode ser feita uma chamada em `GetMembers` com o nome do membro criado, que deverá retornar as informações desse membro.



- Para validar que de fato as chamadas estão passando pelo proxy para chegar no server, pode ser feito a inspeção dos logs no container do proxy (que deve estar executando em modo debug), utilizando `docker logs -f musky-huskle-proxy`
  - Dentro dos logs, deve se encontrar algo como:
    `[2024-07-11 18:31:35.604][23][debug][router] [source/common/router/router.cc:738] [Tags: "ConnectionId":"0","StreamId":"18173009301836948497"] router decoding headers:`
    `':authority', 'localhost:8080'`
    `':method', 'POST'`
    `':path', '/members_service.MembersService/CreateMember'`
    `':scheme', 'http'`
    `'grpc-accept-encoding', 'identity,deflate,gzip'`
    `'accept-encoding', 'identity'`
    `'user-agent', 'grpc-node-js/1.8.10'`
    `'content-type', 'application/grpc'`
    `'te', 'trailers'`
    `'x-forwarded-proto', 'http'`
    `'x-request-id', '834c3041-a1ee-4711-9fe6-11d57eadd8e8’`
  - E também:
    `[2024-07-11 18:56:06.215][22][debug][http] [source/common/http/conn_manager_impl.cc:1838] [Tags: "ConnectionId":"4","StreamId":"17163497840655509902"] encoding headers via codec (end_stream=false):`
    `':status', '200'`
    `'content-type', 'application/grpc'`
    `'x-envoy-upstream-service-time', '1'`

'date', 'Thu, 11 Jul 2024 18:56:06 GMT'
'server', 'envoy'