САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Бек-энд разработка

Отчет Лабораторная работа №3

Выполнил: Митурский Богдан Антонович

Группа: К33392

Проверил: Добряков Д. И.

Санкт-Петербург

2024 г.

Задача

Необходимо реализовать отдельный микросервис, выполняющий какую-либо содержательную функцию из всего арсенала функций вашего приложения.

Ход работы

В рамках работы реализуем микросервисную структуру проекта, а также подготовим конфигурирующий файл для запуска и масштабирования микросервисов. Для управления микросервисами будем использовать Project Manager 2, который позволит удобно управлять кластерами в Node TS.

Подготовим структуру:

- Bot (Микросервис чат-бота работающего отдельно от проекта)
- Hub (Микросервис для основного Rest API на express возвращающего данные пользователя по запросу)
- Lobby (Микросервис для обработки игрового процесса в реальном времени посредством socket.io)
- WaitRoom (Микросервис ожидающий подключения двух игроков и перенаправляющий их в конкретное игровое лобби посредством socket.io)



Каждый из микросервисов - полноценный сервер умеющий запускаться самостоятельно и содержащий внутри свою логику. Подготовим ecosystem.config.js который будет конфигурировать pm2 для запуска всех микросервисов.

```
require("dotenv").config();
const scripts = {
```

```
hub: {
 lobby: {
};
const instances = process.env.INSTANCES || 1;
const IS PRODUCTION = process.env.NODE ENV === "production";
};
const ENV PROD = {
module.exports = {
 apps: [
    instances,
```

```
interpreter: IS_PRODUCTION ? "" : "ts-node",
script: IS PRODUCTION ? scripts.hub.prod : scripts.hub.dev,
node args: "-r tsconfig-paths/register",
instances: 1,
node args: "-r tsconfig-paths/register",
 PORT: process.env.WAIT ROOM PORT || 31000,
instances,
script: IS_PRODUCTION ? scripts.lobby.prod : scripts.lobby.dev,
env: {
```

```
{
   name: `[BOT VK]`,
   instances: 1,
   script: IS_PRODUCTION ? scripts.bot.prod : scripts.bot.dev,
   node_args: "-r tsconfig-paths/register",
   env: { ...ENV_DEV, SOCIAL: "vk" },
   env_production: { ...ENV_PROD, SOCIAL: "vk" },
   interpreter: IS_PRODUCTION ? "" : "ts-node",
},

{
   name: `[BOT TG]`,
   instances: 1,
   script: IS_PRODUCTION ? scripts.bot.prod : scripts.bot.dev,
   node_args: "-r tsconfig-paths/register",
   env: { ...ENV_DEV, SOCIAL: "tg" },
   env_production: { ...ENV_PROD, SOCIAL: "tg" },
   interpreter: IS_PRODUCTION ? "" : "ts-node",
},
],
};
```

Hub будут запускаться на одном порту. Т.к. это обычное REST Api, PM2 из коробки будет распределять запросы и балансировать нагрузку.

Lobby будут запускаться на разных портах, для этого, в конфиге указывается increment_var: "PORT". Каждое лобби будет работать на сокетах и располагаться на отдельном порту, куда игроки будут подключаться на 3х минутные матчи.

Wait Room будет запускаться в единственном экземпляре, т.к. не подразумевает большую нагрузку. Все пользователи будут подключаться к нему, чтобы найти игровой матч.

Вот будет запускаться в двух экземплярах, но, каждый раз с разными параметрами. Один - для работы в тг. Один - для работы в ВК.

В результате, мы настроили масштабируемую микросервисную структуру, в которой микросервисы взаимодействуют друг с другом по большей части через базы данных и иногда с помощью запросов в локальной сети, а в остальном независимы и легко масштабируемы. На скриншотах ниже предоставлен результат запуска микросервисов в 1 и в 4 экземпляра. Как видим, масштабируются только заложенные нами Hub и Lobby, остальные микросервисы запускаются в едином экземпляре.

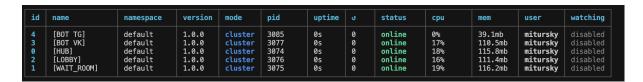


Рисунок 1 - запуск в 1 экземпляр

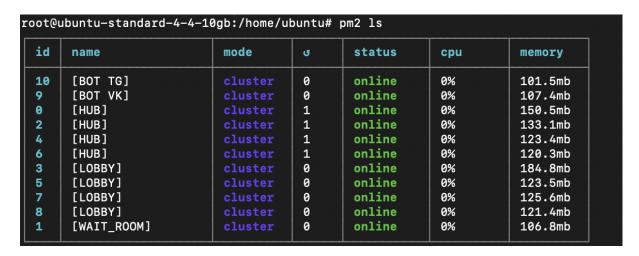


Рисунок 2 - запуск в 4 экземпляра

Вывод

В ходе работы была разработана микросервисная архитектура и настроены 4 отдельных микросервиса. Они могут независимо масштабироваться и коммуницировать между собой. Так, у проекта появилась масштабируемая и удобная архитектура.