САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Бек-энд разработка

Отчет
Лабораторная работа №3

Выполнил: Митурский Богдан Антонович

Группа: K33392

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: {
   dev: "./instances/hub/index.ts",
   prod: "./backend/instances/hub/index.js",
 },
 lobby: {
   dev: "./instances/lobby/index.ts",
   prod: "./backend/instances/lobby/index.js",
 },
 waitRoom: {
   dev: "./instances/waitRoom/index.ts",
   prod: "./backend/instances/waitRoom/index.js",
 },
 bot: {
   dev: "./instances/bot/index.ts",
   prod: "./backend/instances/bot/index.js",
 },
};

/** Количество инстансов */
const instances = process.env.INSTANCES || 1;

/** Режим запуска */
const IS_PRODUCTION = process.env.NODE_ENV === "production";

const ENV_DEV = {
 ...process.env,
 NODE_ENV: "development",
 MONGODB_URL: process.env.MONGODB_URL_DEV,
 SERVER_URL: process.env.SERVER_URL_DEV,
};

const ENV_PROD = {
 ...process.env,
 NODE_ENV: "production",
 MONGODB_URL: process.env.MONGODB_URL_PROD,
 SERVER_URL: process.env.SERVER_URL_PROD,
};

module.exports = {
 apps: [
   {
     name: "[HUB]",
     instances,
     exec_mode: "cluster_mode",
```

```
    interpreter: IS_PRODUCTION ? "" : "ts-node",
    script: IS_PRODUCTION ? scripts.hub.prod : scripts.hub.dev,
    node_args: "-r tsconfig-paths/register",
    env: {
      ...ENV_DEV,
      PORT: process.env.HUB_PORT || 8000,
    },
    env_production: {
      ...ENV_PROD,
      PORT: process.env.HUB_PORT || 8000,
    },
  },
  {
    name: "[WAIT_ROOM]",
    instances: 1,
    interpreter: IS_PRODUCTION ? "" : "ts-node",
    script: IS_PRODUCTION ? scripts.waitRoom.prod : scripts.waitRoom.dev,
    node_args: "-r tsconfig-paths/register",
    env: {
      ...ENV_DEV,
      PORT: process.env.WAIT_ROOM_PORT || 31000,
    },
    env_production: {
      ...ENV_PROD,
      PORT: process.env.WAIT_ROOM_PORT || 31000,
    },
  },
  {
    name: `[LOBBY]`,
    instances,
    interpreter: IS_PRODUCTION ? "" : "ts-node",
    script: IS_PRODUCTION ? scripts.lobby.prod : scripts.lobby.dev,
    node_args: "-r tsconfig-paths/register",
    env: {
      ...ENV_DEV,
      PORT: process.env.LOBBY_INITIAL_PORT || 32000,
    },
    env_production: {
      ...ENV_PROD,
      PORT: process.env.LOBBY_INITIAL_PORT || 32000,
    },
    increment_var: "PORT",
  },
  // Убрать при необходимости проверить ботов
```

```
    {
      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.

Bot 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.

id	name	namespace	version	mode	pid	uptime	↺	status	cpu	mem	user	watching
4	[BOT TG]	default	1.0.0	cluster	3085	0s	0	online	0%	39.1mb	mitursky	disabled
3	[BOT VK]	default	1.0.0	cluster	3077	0s	0	online	17%	110.5mb	mitursky	disabled
0	[HUB]	default	1.0.0	cluster	3074	0s	0	online	18%	115.8mb	mitursky	disabled
2	[LOBBY]	default	1.0.0	cluster	3076	0s	0	online	16%	111.4mb	mitursky	disabled
1	[WAIT_ROOM]	default	1.0.0	cluster	3075	0s	0	online	19%	116.2mb	mitursky	disabled

Рисунок 1 - запуск в 1 экземпляр

```
root@ubuntu-standard-4-4-10gb:/home/ubuntu# pm2 ls
```

id	name	mode	↺	status	cpu	memory
10	[BOT TG]	cluster	0	online	0%	101.5mb
9	[BOT VK]	cluster	0	online	0%	107.4mb
0	[HUB]	cluster	1	online	0%	150.5mb
2	[HUB]	cluster	1	online	0%	133.1mb
4	[HUB]	cluster	1	online	0%	123.4mb
6	[HUB]	cluster	1	online	0%	120.3mb
3	[LOBBY]	cluster	0	online	0%	184.8mb
5	[LOBBY]	cluster	0	online	0%	123.5mb
7	[LOBBY]	cluster	0	online	0%	125.6mb
8	[LOBBY]	cluster	0	online	0%	121.4mb
1	[WAIT_ROOM]	cluster	0	online	0%	106.8mb

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.