УНИВЕРСИТЕТ ИТМО

Факультет программной инженерии и компьютерной техники

Направление подготовки 09.03.04 Программная инженерия

Дисциплина «Сервис-Ориентированная архитектура»

**Лабораторная работа №3**

Вариант 1229

Студент

*Митрофанов Е. Ю.*

*Любкин А. С.*

*P34101*

Преподаватель

*Цопа Е. А.*

Санкт-Петербург, 2022 г.

Задание лабораторной работы

Переработать веб-сервисы из лабораторной работы #2 таким образом, чтобы они реализовывали основные концепции микросервисной архитектуры. Для этого внести в оба сервиса -- "вызываемый" и "вызывающий" перечисленные ниже изменения.

**Изменения в "вызываемом" сервисе:**

* Сконфигурировать окружение для работы сервиса на платформе Spring Boot.
* Запустить второй экземпляр сервиса на другом порту. Реализовать балансировку нагрузки между экземплярами с помощью Haproxy.
* Реализовать механизм Service Discovery. Для этого установить Consul и интегрировать свой сервис с ним, автоматически регистрируя в момент запуска.

**Изменения в "вызывающем" сервисе:**

* Разделить приложение на два модуля -- веб-приложение с веб-сервисом и EJB-jar с бизнес-компонентами.
* Переместить всю логику из класса сервиса в Stateless EJB. В классе сервиса оставить только обращение к методам бизнес-интерфейса. EJB-компонент должен быть доступен удалённо (иметь Remote-интерфейс).
* Сформировать на уровне сервера приложений пул компонентов EJB настраиваемой мощности, динамически расширяемый при увеличении нагрузки.
* Настроить второй экземпляр сервера приложений на другом порту, "поднять" на нём вторую копию веб-сервиса и пула EJB.
* Настроить балансировку нагрузки на оба запущенных узла через Haproxy.

Спецификация REST API

Спецификация доступна по ссылке: <https://se.ifmo.ru/~s285583/soa/>

Ссылки на исходный код

Серверное приложение: <https://github.com/EgorMIt/SOA-Lab2>

Клиентское приложение: <https://github.com/EgorMIt/SOA-Lab2-frontend>

Контейнеризация сервисов

Изображение выглядит как текст

Автоматически созданное описание

Вывод

Во время выполнения лабораторной работы мы переработали ранее разработанное приложение в соответствии с основными концепциями микросервисной архитектуры. Мы изучили работу балансировщиков нагрузки и использовали Haproxy для балансировки нескольких экземпляров двух сервисов приложения, а также использовали механизм Service Discovery для автоматической регистрации сервисов приложения. Для развертывания всех сервисов приложения мы использовали технологии контейнеризации Docker.