Университет ИТМО

Факультет программной инженерии и компьютерной техники

Лабораторная работа №3

По дисциплине «Сервис-ориентированная архитектура»

Вариант № 2002.14.05

Студент:
Михайлова Алена Андреевна

Группа:
P34112

Преподаватель:
Каюков Иван Алексеевич

Санкт-Петербург

2021 г.

1. **Текст задания.**

Переработать веб-сервисы из лабораторной работы #2 таким образом, чтобы они реализовывали основные концепции микросервисной архитектуры. Для этого внести в оба сервиса - "вызываемый" (из лабораторной работы #1) и "вызывающий" (добавленный в лабораторной работе #2) 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.

Оба веб-сервиса и клиентское приложение должны сохранить полную совместимость с API, реализованными в рамках предыдущих лабораторных работ.

2. **Ссылка на репозиторий с исходным кодом**

https://github.com/amikhaylova/soa-lab-3

3. **Конфигурация Haproxy**

Базовый сервис (Spring Boot)

```
frontend base_frontend
bind 127.0.0.1:8081
option tcplog
mode tcp
default_backend base_backend

backend base_backend
mode tcp
option ssl-hello-chk
server-template mywebapp 2 _application._tcp.service.consul resolvers consul resolve-
opts allow-dup-ip resolve-prefer ipv4 check
```

```
resolvers consul
nameserver consul 127.0.0.1:8600
nameserver dns 192.168.0.1:53
accepted_payload_size 8192
hold valid 10s
```

Вызывающий сервис (JAX-RS + EJB)

```
frontend caller_frontend
bind 127.0.0.1:8383
option tcplog
mode tcp
default_backend caller_backend

backend caller_backend
mode tcp
option ssl-hello-chk
server server1 127.0.0.1:28181
server server2 127.0.0.1:28182
```

4. Конфигурация пула EJB

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE glassfish-ejb-jar PUBLIC "-//GlassFish.org//DTD GlassFish Application Server
3.1 EJB 3.1//EN" "http://glassfish.org/dtds/glassfish-ejb-jar_3_1-1.dtd">
<glassfish-ejb-jar>
    <enterprise-beans>
        <ejb>
            <ejb-name>OscarServiceImpl</ejb-name>
            <bean-pool>
                <max-pool-size>20</max-pool-size>
                <steady-pool-size>1</steady-pool-size>
            </bean-pool>
        </ejb>
    </enterprise-beans>
</glassfish-ejb-jar>
```

5. Выводы по работе

В ходе данной лабораторной работы было еще более грустно, чем в ходе предыдущей лабораторной работы.

Очень долго возилась с настройкой сертификатов безопасности в Payara. Ошибка была в том, что я почему-то решила делать это через ui административной консоли. Нужно было делать сразу через .xml конфиги и не страдать.

Самым большим открытием стала связка Haproxy и Consul. Это очень здорово, что Haproxy может формировать пул бэкенд сервисов, делая запросы к Consul, и что не нужно хардкодить в конфигах Haproxy ip-адреса и порты.

Самой безболезненной частью работы неожиданно стали ejb.

А еще я нашла баг в коде с прошлой лабы в одной из функций бизнес-логики, из-за которого очень долго выполнялся запрос.