Университет ИТМО

Факультет программной инженерии и компьютерной техники

Лабораторная работа №3
Сервис-ориентированная архитектура

Выполнил: Кривошейкин Сергей

Группа № P34122

Преподаватель: Усков И. В.

г. Санкт-Петербург

2021

Задание

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:

- 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.
- Установить ПО Consul и настроить Service Discovery с его помощью. Сервис должен регистрироваться в Service Discovery 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.

Код: https://github.com/Serzh721/SOA_lab3

Haproxy.cfg:

```
defaults
    timeout connect 5000ms
    timeout server 50000ms
    timeout client 50000ms
    timeout check 50000ms

frontend soa
    mode http
    bind *:443 ssl crt C:\soa.pem
    http-request set-header X-Forwarded-For %[src]
    use_backend soa-2

backend soa-2
    mode http
    http-request set-header X-Forwarded-For %[src]
    balance roundrobin
    server instance_1 127.0.0.1:10444 ssl check verify none
    server instance_2 127.0.0.1:10445 ssl check verify none

listen stats
    bind 127.0.0.1:1111
    mode http
    stats uri /
    stats auth soa:soa
```

Вывод: в ходе выполнения данной лабораторной работы я разделил оба веб-сервиса на несколько модулей, помучался с сериализацией даты и запуском одного и того же war 2 раза параллельно, а также познакомился с Haproxy и Consul.