

Федеральное государственное автономное образовательное  
учреждение высшего образования

Университет ИТМО

Дисциплина: Сервис-ориентированная архитектура

## **Лабораторная работа 3**

Вариант 8521729

**Выполнили:**

Кривоносов Егор Дмитриевич

**Группа:** Р34111

**Преподаватель:**

Райла Мартин

2023 г.

Санкт-Петербург

# Оглавление

<b>Задание</b>	<b>3</b>
<b>Выполнение</b>	<b>3</b>
Исходный код	3
Consul:	4
Nargoхy:	5
Настройка (то что нужно было добавить в конфиг):	5
Создание сертификата в формате .pem для Nargoхy:	6

# Задание

## Лабораторная работа #3

Введите вариант:

### Внимание! У разных вариантов разный 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" 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 Nginx.
- 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 Nginx.

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


# Выполнение

## Исходный код

<https://github.com/RedGry/SOA>

Лабораторная работа на **JAVA 11**, если попытаться запустить на более старой версии - будут проблемы с wildfly.

# Consul:

 dc1 **Services** Nodes Key/Value ACL Intentions

[← All Services](#)

## first-service

[Instances](#) [Intentions](#) [Tags](#)

**first-service-45012**

✓ All service checks passing

✓ All node checks passing

soa

192.168.0.121:45012

contextPath=/api/v1

**first-service-45022**

✓ All service checks passing

✓ All node checks passing

soa

192.168.0.121:45022

contextPath=/api/v1

## Haproxy:

stats												
	Queue			Session rate			Sessions					
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last
Frontend				0	2	-	2	2	262 122	8		6
Backend	0	0		0	2		0	1	26 213	4	0	0s 6

haproxy_as_api_gateway												
	Queue			Session rate			Sessions					
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last
Frontend				0	1	-	1	3	262 122	3		3

load_balancer												
	Queue			Session rate			Sessions					
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last
first_service_1	0	0	-	0	1		0	1	-	3	3	12s
first_service_2	0	0	-	0	1		0	1	-	3	3	8s
Backend	0	0		0	2		0	1	26 213	6	6	8s

haproxy_as_api_gateway_2												
	Queue			Session rate			Sessions					
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last
Frontend				0	1	-	0	2	262 122	1		7s

load_balancer_2												
	Queue			Session rate			Sessions					
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last
second-service-1	0	0	-	0	1		0	1	-	1	1	
second-service-2	0	0	-	0	0		0	0	-	0	0	
Backend	0	0		0	1		0	1	26 213	1	1	

Настройка (то что нужно было добавить в конфиг):

```
listen stats
    bind 127.0.0.1:8404
    stats enable
    stats uri /stats
    stats refresh 10s

frontend haproxy_as_api_gateway
    bind 127.0.0.1:4500 ssl crt /etc/haproxy/ssl/mydomain.pem
    default_backend load_balancer
```

```
backend load_balancer
    server first_service_1 127.0.0.1:45012 check ssl verify none
    server first_service_2 127.0.0.1:45022 check ssl verify none

frontend haproxy_as_api_gateway_2
    bind 127.0.0.1:4600 ssl crt /etc/haproxy/ssl/mydomain.pem
    default_backend load_balancer_2

backend load_balancer_2
    server second-service-1 127.0.0.1:46012 check ssl verify none
    server second-service-2 127.0.0.1:46022 check ssl verify none
```

## Создание сертификата в формате .pem для Нароху:

1. `keytool -export -alias localhost -file server.der -keystore server.keystore`  
P.S.: (как делать keystore можно посмотреть в отчете по 2 лабе)
2. `openssl x509 -inform der -in server.der -out server.pem`
3. `keytool -importkeystore -srckeystore server.keystore -destkeystore server.keystore.p12 -deststoretype PKCS12`
4. `openssl pkcs12 -in server.keystore.p12 -nodes -nocerts -out server.key`
5. `cat server.key >> mydomain.pem`
6. `cat server.pem >> mydomain.pem`