По сравнению с встроенным, здесь запросы проходят напрямую по нужному адресу, что исключает лишний узел при вызове. Он интегрирован с механизмом Service Discovery.

А.Г. Шведов (УО «ГГУ имени Ф. Скорины», Гомель)
Науч. рук. **В.Д. Левчук**, канд. техн. наук, доцент

ДЕТАЛИ РЕАЛИЗАЦИИ МАСШТАБИРУЕМОСТИ В ПРОЕКТЕ СОЦИАЛЬНОЙ СЕТИ ДЛЯ ОБМЕНА ПУБЛИЧНЫМИ СООБЩЕНИЯМИ

При проектировании архитектуры социальной сети следует понимать, что маштабируемость – это одна из важнейших состовляющих. Конечно, создание микросервисного приложения требует больше усилий по проектированию, и связыванию отдельных компонентов, но в дальнейшем от этого только плюсы.

Проект разделён на отдельные сервисы: сервис для работы с юзером, сервис для работы с топиками-публикациями-коментариями, сервис для работы с приглашениями и сервис для работы с web-частью.

При создании микросервисного приложения существует проблема взаимодесйствия микросервисов друг с другом. Использование встроенного в Spring микросервис – eureka, который позволил обращаться к другим сервисам по имени, а не по ip.

Для добавления встроеного микросервиса следует дабавить аннотацию в каждом сервисе и добавть LoadBalancer.

```
@SpringBootApplication
@EnableEurekaClient
public class TopicServer {
    public static void main(String[] args) {
     SpringApplication.run(TopicServer.class,args);
    }
    @Bean
    @LoadBalanced
    RestTemplate restTemplate() {
     return new RestTemplate();
    }
}
```

Это позволило легко маштабировать проект горизонтально. А также возможность подмены одного сервиса другим без необходимости переустанавливать все приложение. Также это даёт высокую отказоустойчивость (при отказе какого-либо сервиса, можно вернуть прошлую работающую версию, не перезапуская его) и незавсисищее развёртывание (простые сервисы легче внедрять, небольшая вероятность отказа системы).