# Практическая работа. Лекция 16. Взаимодействие сервисов

В данной работе описано только взаимодействие сервисов в рамках процесса по отправке сообщения и получения его статуса. Данный процесс охватывает все типы взаимодействия, которые аналогичным образом применятся в системе между другими компонентами системы. Предполагается использовать микросервисную архитектуру.

## Взаимодействие сервисов message, adapter, SMSC

Компоненты взаимодействия:

1. Клиент – клиент СМС-сервиса
2. ApiGateway – шлюз, через который проходят все запросы. Перенаправляет запросы в сервис User для осуществления контроля доступа(аутентификация и авторизация);
3. Service Message – сервис по отправке сообщений;
4. Service User – сервис пользователей, которые управляют системой по отправке сообщений;
5. Service Adapter – сервис по взаимодействию с конкретным внешним провайдером по отправке смс-сообщений и получение его статуса;
6. Брокер message – являются промежуточным звеном взаимодействия между сервисами message, adapter;
7. SMSC – внешний провайдер.

Описание взаимодействие сервисов при отправке смс-сообщений и получение его статуса(Рис. 1):

1. Клиент отправляет запрос(содержит в header токен) по отправке сообщения в Систему СМС-сервис;
2. Запрос сперва попадает в шлюз Apigateway, который перенаправляет запрос в сервис user, чтобы аутентифицировать и авторизовать пользователя;
3. Если пользователь имеет действительный токен, то запрос далее передается в сервис message. Если токен оказался невалидным, то возвращается ошибка 401;
4. Сервис message отправляет сообщение в topic брокера message;
5. Сервис adapter читает сообщения из брокера message сообщения, трансформирует сообщения в необходимый формат(SMPP) и передает его во внешний сервис SMSC.

В каждом сервисе поддерживается идемпотентность, т.е. запрос обрабатывается один раз, уникальность сообщения



Рис. - Диаграмма последовательости взаимодействия центрального шлюза ApiGateway, сервиса User, сервиса message, сервиса adaper, внешнего сервиса SMSC при отправке сообщения(/notifier/sendmessage) и получения статуса отправленного сообщения(/notifier/msgstatus)

## Сценарии

**Сценарий “Обновить сервис adapter”.**

Применение брокера дает следующие преимущества:

Сервисы мало связаны, нет привязки к API сервиса, с котором необходимо взаимодействовать.

Атрибуты качества: гибкость(flexibility), совместимость(interoperability).

Критические характеристики:

* Адаптивность
* Модульность
* Слабая связанность

**Сценарий “Упал один из инстансов брокера message”.**

Кafka может работать в кластере, поддерживает репликацию, а также может автоматически перераспределять сообщения, в случае падения раздела или темы(topic).

Атрибуты качества: восстанавливаемость(recoverability), устойчивость(sustainability), надежность(reliability).

Критические характеристики:

* Зона поражения и потери бизнеса – сообщения, которые не были отправлены вовремя, могут потерять свою актуальность.
* Время на определение сбоя (MTTD – mean time to detect) – 15 сек.
* Время на исправление (MTTR – mean time to recovery) - 1-12 минуты
* Время между отказами (MTBF – mean time between failures) – 120 мин
* Процент доступности, время даунтайма системы за период – 99%
* Доля ошибок по отношению к доле успешных запросов – 95%
* RPO (recovery point objective) - допустимая потеря данных – 1%
* RTO (recovery time objective) - допустимое время восстановления данных – 20 мин.

**Сценарий “сервис adapter упал”.**

Брокер сообщений kafka дает возможность сохранять сообщения пока они не будут обработаны. После того, как будет поднят экземпляр сервиса adapter, он сможет продолжить всех необработанных запросов, скопившихся в брокере.

Атрибуты качества: восстанавливаемость(recoverability), устойчивость(sustainability), надежность(reliability).

Критические характеристики:

* Время на определение сбоя (MTTD – mean time to detect) – 30 сек.
* Время на исправление (MTTR – mean time to recovery) - 1-3 минуты
* Время между отказами (MTBF – mean time between failures) – 60 мин
* Процент доступности, время даунтайма системы за период – 99%
* Доля ошибок по отношению к доле успешных запросов – 95%
* Модель консистентности и производительность (latency) – 1сек
* RPO (recovery point objective) - допустимая потеря данных – 1%
* RTO (recovery time objective) - допустимое время восстановления данных – 1мин.

**Сценарий “экземпляр сервиса User упал”.**

Последовательность операций(Рис. 2):

1. Сервис ApiGateway отправляет запрос для аутентифкации и авторизации запроса;
2. Экземпляр 1 сервиса User отправляет ошибку 500;
3. Circuit Breaker перехватывает ошибку, отправляет повторно запрос(Retry), но сервис либо не отвечает и отваливается по таймауту, либо возвращает ошибку 500.
4. Circuit Breaker устанавливает данному инстансу сервиса статус CLOSED;
5. Circuit Breaker перенаправляет запрос другому инстансу, либо дает сигнал, что необходимо поднять новый инстанс сервиса User и перенаправляет запрос экземпляру 2;
6. Circuit Breaker получив ответ от экземпляра 2 сервиса User со статусом 200 или 2…, перенаправляет ответ в сервис ApiGateway;

Атрибуты качества: восстанавливаемость(recoverability), устойчивость(sustainability), надежность(reliability).

Критические характеристики:

* Время на определение сбоя (MTTD – mean time to detect) – 30 сек.
* Время на исправление (MTTR – mean time to recovery) - 1-3 минуты
* Время между отказами (MTBF – mean time between failures) – 60 мин
* Процент доступности, время даунтайма системы за период – 99%
* Доля ошибок по отношению к доле успешных запросов – 95%
* RPO (recovery point objective) - допустимая потеря данных – 1%
* RTO (recovery time objective) - допустимое время восстановления данных – 1мин.

**Сценарий “В сервис пришел запрос, который ранее уже приходил”.**

Идемпотентность позволяет с меньшими затратами мониторить ошибки, а также производить повторную обработку запроса не меняя полученный результат, зафиксированного при первой обработке запроса.

Атрибуты качества: повторяемость(repeatability), наблюдаемость(observability), прозрачность(transparency).

Критические характеристики:

* Повторная обработка запроса не влияет на состояние или результат.



Рис. - Диаграмма последовательности переключение запроса в случае отказа экземпляра сервиса

**Сценарий “Количество запросов от клиентов системы периодически увеличивается в 10-27 раз”**

Использование микросервисной архитектуры, платформы по управлению контейнерами kubernetes позволят отслеживать нагрузку и в зависимости от нее разворачивать и сворачивать дополнительные экземпляры каждого из сервисов в системе.

Атрибуты качества: масштабируемость(scalability), производительность.

Критические характеристики:

* Стоимость изменений при увеличении нагрузки. Применяется горизонтальное масштабирование. Затраты на количество потребляемых ресурсов провайдеру облачных систем, персонал, который осуществляет поддержку инфраструктуры.
* Деградация производительности при увеличении нагрузки. Узким местом может быть шлюз ApiGateway. Решение: подключение еще одного ЦОДа, кластера. Рекомендуемый характер зависимости времени отклика от нагрузки: от степенной(0 < n < 1) до логарифмического.