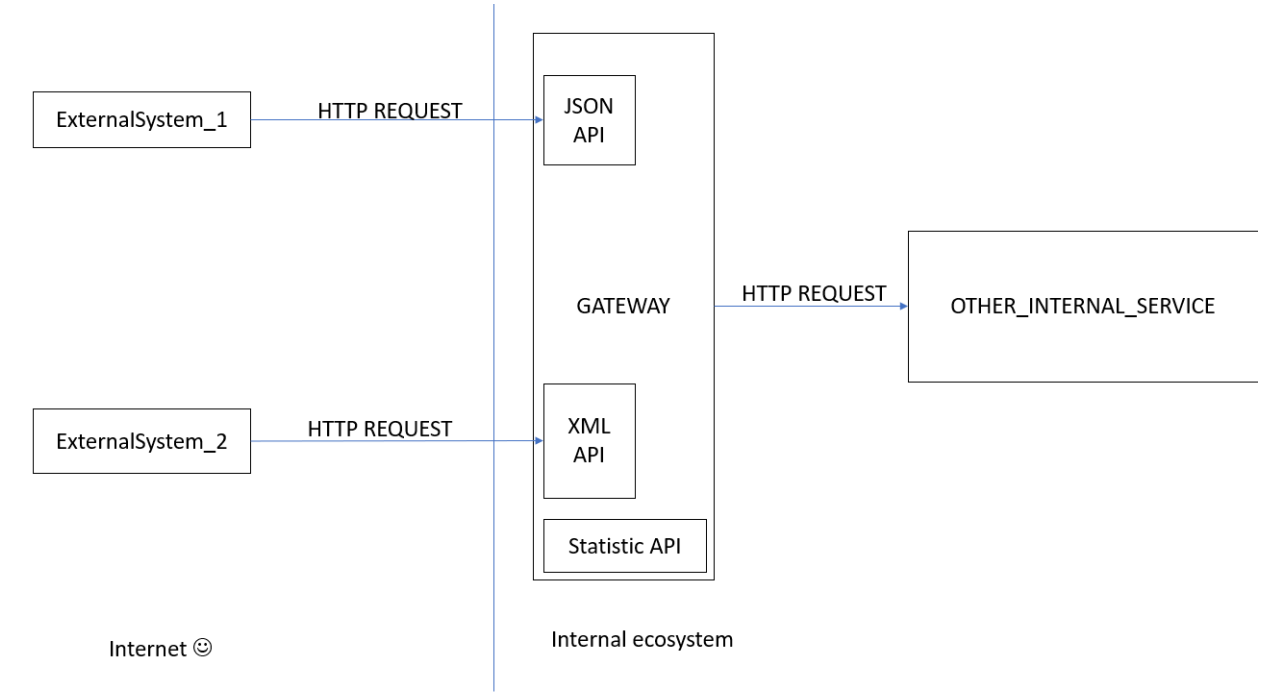
Цел на заданието: Да се създаде приложение фасада (наричано GATEWAY), което

1. да интерпретира и след това унифицира данни идващи от външни системи
2. прехвърля данните към друго приложение, което поема последваща обработка

Следната диаграма показва най-общо възможностите на приложението



Приложението трябва да поддържа два вида API за външните системи.

JSON API: Поддържа два ендпойнта, които приемат следните POST рикуести:

/json_api/insert

```
{
  "requestId": "b89577fe-8c37-4962-8af3-7cb89a245160",
  "timestamp": 1586335186721,
  "producerId": "1234",
  "sessionId": 47966003032113150
}
```

Обработката на този рикуест изисква проверка за съществуване на сесия с даденото id и евентуално създаването на такава. Към сесията трябва да се съхрани requestId и в случай на

повторение да върне грешка в респонса. При нормално изпълнение на рикуеста, респонса е OK и без контент

/json_api/find

```
{
 "requestId": "b89577fe-8c37-4962-8af3-7cb89a24q909",
 "sessionId": 47966003032113150
}
```

Този рикуест изисква отново проверка за съществуване на сесията и създаването ѝ при необходимост. Респонса трябва да върне json списък от всички събрани requestId

XML API: Предоставя единствен ендпойнт за пост рикуести, който може да приема следните данни:

/xml_api/command

```
<command id="1234" >
        <enter session="13617162" >
                <timestamp>1586335186721</timestamp>
                <player>238485</player>
        </enter>
</command>
<command id="1234-8785" >
        <get session="13617162" />
</command>
```

Логиката се повтаря относно съхранението на данните, като се има предвид, че ид-то на командата играе аналогична роля на requestId в json api. Също така player и producerId са аналогични понятия, идентифициращи крайния потребител.

Статистическо API: С цел статистика, приложението трябва да предоставя GET ендпойнт, който по зададено id на потребител, връща в json формат лист от id на негови сесии.

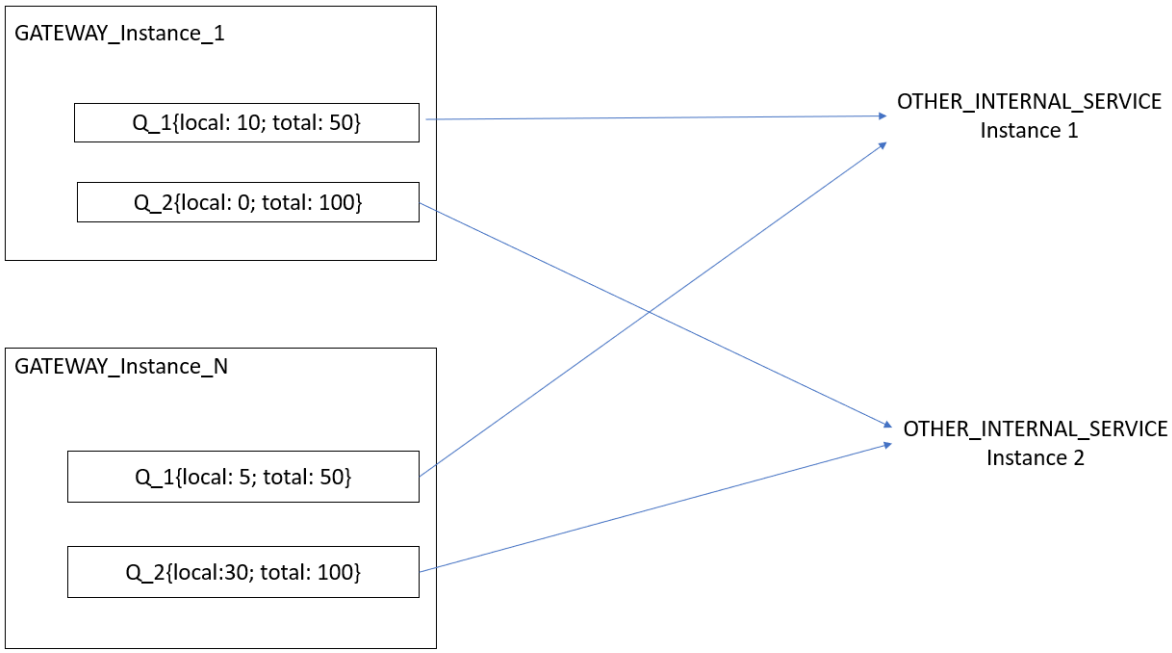
Архитектура

Като част от обработката на рикуестите от външните системи, трябва да се включи асинхронен REST API call към друг сервиз (OTHER_INTERNAL_SERVICE). Предлагам да използваш exo сървиса на Постман. Респонса от този сървис, трябва да бъде запазен от приложението. В екосистемата има н броя инстанции на OTHER_INTERNAL_SERVICE, които са фиксирани и съответно конфигурирани в GATEWAY. Приложението GATEWAY също има множество инстанции.

Вътрешно обръщението към OTHER_INTERNAL_SERVICE се случват посредством „опашка“, в която рикуеста се добавя, преди да се върне респонс към външната система. В GATEWAY има по една опашка за всяка инстанция на OTHER_INTERNAL_SERVICE. Какъв вид „опашки“ ще използваш е изцяло твое решение.

Частта от приложението, която ще чете от опашка, обработва по един рикуест на предефиниран интервал от време. Целта тук е да се симулира натоварване и някакъв вид изчакване.

Инстанциите на GATEWAY, трябва да осигуряват равномерно натоварване на инстанциите на OTHER_INTERNAL_SERVICE. Това означава, че GATEWAY инстанциите трябва да имат бърз механизъм за синхронизация. Във всеки един момент, всяка инстанция на GATEWAY, трябва да знае приблизително колко общо изчакващи рикуести има към всяка инстанция на OTHER_INTERNAL_SERVICE



Нека кажем, че в текущия момент, системите изглеждат както на диаграмата. Постъпва рикуест в инстанция 1 на GATEWAY. От гледна точка на локалните опашки, Q_2 няма натоварване и би следвало там да се разпредели рикуеста. Благодарение на описаната синхронизация, рикуеста в действителност трябва да иде на Q_1.

Технологичния стек, с който разполагаш е .NET Core 3.1, EF/EF Core, Redis, RabbitMQ, Postgres, Docker.

Имплементацията трябва да се направи с презумпцията, че това е силно натоварена система откъм брой рикуести за единица време.

Огромно преимущество ще е добавянето на тестове към проекта.

Изисква се да осигуриш достъп до git репозитори по твой избор.