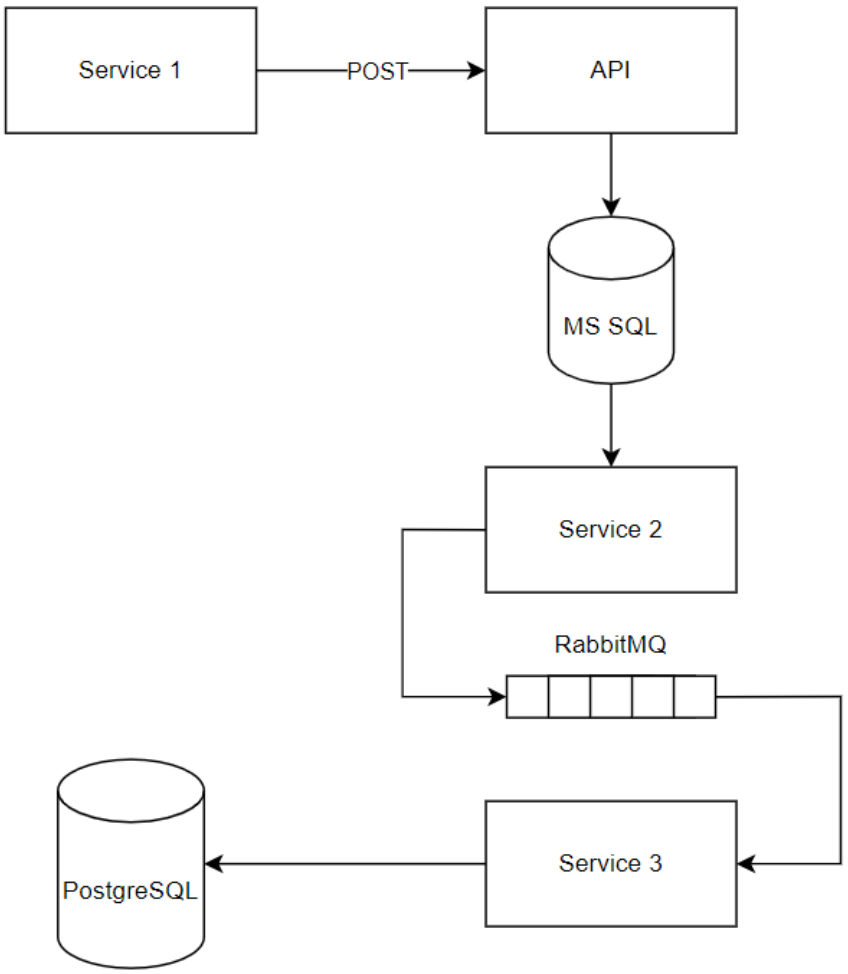
Задание на стажировку
Реализация системы репликации спортивных данных.

Задание: Необходимо реализовать систему репликации спортивных данных. Общая схема системы представлена на рисунке.



Service1

Сервис должен генерировать хаотичные спортивные данные и передавать их в API. В сервисе должно быть ограничено кол-во данных которые он генерирует.
Для генерации данных необходимо использовать библиотеку Bogus (https://github.com/bchavez/Bogus).
Для взаимодействия с API необходимо использовать библиотеку Flurl (https://github.com/tmenier/Flurl).

API

Апи должна принимать спортивные данные и сохранять их в БД MS SQL.
Апи должна быть реализована в стиле Minimal API.
Для взаимодействия с БД необходимо использовать библиотеку Dapper (https://github.com/DapperLib/Dapper).

Пример входящих сообщений.
Events:

```
{
    "SportName" : "Soccer",
    "CategoryName" : "England",
    "ChampionshipName" : "Premier League",
    "EventName" : "Arsenal vs. Tottenham",
    "EventDate" : "2023-07-20T17:30:00"
}
```

Markets:

```
{
    "EventName" : "Arsenal vs. Tottenham",
    "Markets" : [
        {
            "MarketName" : "1x2",
            "Outcomes" : [
                {
                    "OutcomeName" : "1",
                    "Price" : 1.2
                },
                {
                    "OutcomeName" : "X",
                    "Price" : 1.4
                },
                {
                    "OutcomeName" : "2",
                    "Price" : 1.6
                }
            ]
        },
        {
            "MarketName" : "Double Chance",
            "Outcomes" : [
                {
                    "OutcomeName" : "1X",
                    "Price" : 1.2
                },
                {
                    "OutcomeName" : "12",
                    "Price" : 1.4
                },
```

```
                {
                    "OutcomeName" : "X2",
                    "Price" : 1.6
                }
            ]
        }
    ]
}
```

Service 2

Сервис должен по таймеру (например, раз в 5 минут) вычитывать только новые данные из БД MS SQL и отправлять их в очередь RabbitMQ. Данные должны отправляться с идентификаторами.
Для взаимодействия с БД необходимо использовать библиотеку Dapper (https://github.com/DapperLib/Dapper).
Для работы с очередью необходимо использовать библиотеку RabbitMQ.Client (https://github.com/rabbitmq/rabbitmq-dotnet-client).

Service 3

Сервис должен принимать данные из очереди RabbitMQ и сохранять в БД PostgreSQL. Обработка сообщений должна проходить параллельно в 5 потоках. При этом сообщения с одинаковым идентификатором должны быть обработаны последовательно.
Для взаимодействия с БД необходимо использовать библиотеку Dapper (https://github.com/DapperLib/Dapper).
Для работы с очередью необходимо использовать библиотеку RabbitMQ.Client (https://github.com/rabbitmq/rabbitmq-dotnet-client).

Пример входящих сообщений:

```
{
    "SportId" : 1,
    "SportName" : "Soccer",
    "CategoryId" : 2,
    "CategoryName" : "England",
    "ChampionshipId" : 3,
    "ChampionshipName" : "Premier League",
    "EventId" : 1234,
    "EventName" : "Arsenal vs. Tottenham",
    "EventDate" : "2023-07-20T17:30:00",
    "Markets" : [
        {
            "MarketId" : 1,
            "MarketName" : "1x2",
```

```
            "Outcomes" : [
                {
                    "OutcomeId" : 1,
                    "OutcomeName" : "1",
                    "Price" : 1.2
                },
                {
                    "OutcomeId" : 2,
                    "OutcomeName" : "X",
                    "Price" : 1.4
                },
                {
                    "OutcomeId" : 3,
                    "OutcomeName" : "2",
                    "Price" : 1.6
                }
            ]
        },
        {
            "MarketId" : 10,
            "MarketName" : "Double Chance",
            "Outcomes" : [
                {
                    "OutcomeId" : 1,
                    "OutcomeName" : "1X",
                    "Price" : 1.2
                },
                {
                    "OutcomeId" : 2,
                    "OutcomeName" : "12",
                    "Price" : 1.4
                },
                {
                    "OutcomeId" : 3,
                    "OutcomeName" : "X2",
                    "Price" : 1.6
                }
            ]
        }
    ]
}
```

Индивидуальные задания.

Команда.

1. Завести закрытый репозиторий на GitHub.
2. Спроектировать схемы БД. БД должны соответствовать **третьей** нормальной форме.

Ермолаев Данила.

1. Реализовать Minimal API.

Емельянов Валентин.

1. Реализовать скрипты создания БД PostgreSQL.
2. Реализовать Service 1.

Хабибулов Сергей.

1. Реализовать скрипты создания БД MS SQL.
2. Реализовать Service 2.

Безрядина Алла.

1. Реализовать Service 3.