Микросервисная архитектура интернет-магазина (ограниченный контекст - управление заказами)

В рамках ограниченного контекста “управления заказами” выделим четыре сервиса, руководствуясь бизнес-возможностями: “заказ”, “оплата”, “склад”, “доставка”.

1. Пользовательские сценарии

1. **Когда** пользователь создаст заказ
   Тогда в списке заказов отобразится новый заказ
   И заказ будет иметь статус “Создан”

2. **Когда** будет произведена оплата заказа
   Тогда статус заказа обновится на “Оплачен”
   И заказ будет зарезервирован на складе

3. **Когда** заказ будет зарезервирован на складе
   Тогда статус заказа обновится на “Зарезервирован”
   И заказ будет передан на доставку

4. **Когда** заказ будет передан на доставку
   Тогда статус заказа обновится на “Доставляется”

5. **Когда** заказ будет доставлен
   Тогда статус заказа обновится на “Выполнен”

6. **Когда** пользователь отменит заказ
   И оплата будет уже произведена
   Тогда статус заказа обновится на “Отменен”
   И будет произведен возврат оплаты

7. **Когда** пользователь отменит заказ
   И заказ будет уже передан на доставку
   Тогда статус заказа обновится на “Отменен”
   И будет произведен возврат оплаты

И будет создано поручение в службу доставки на возврат заказа на склад

8. **Когда** произойдет ошибка в процессе оплаты
   Тогда статус заказа обновится на “Отменен”

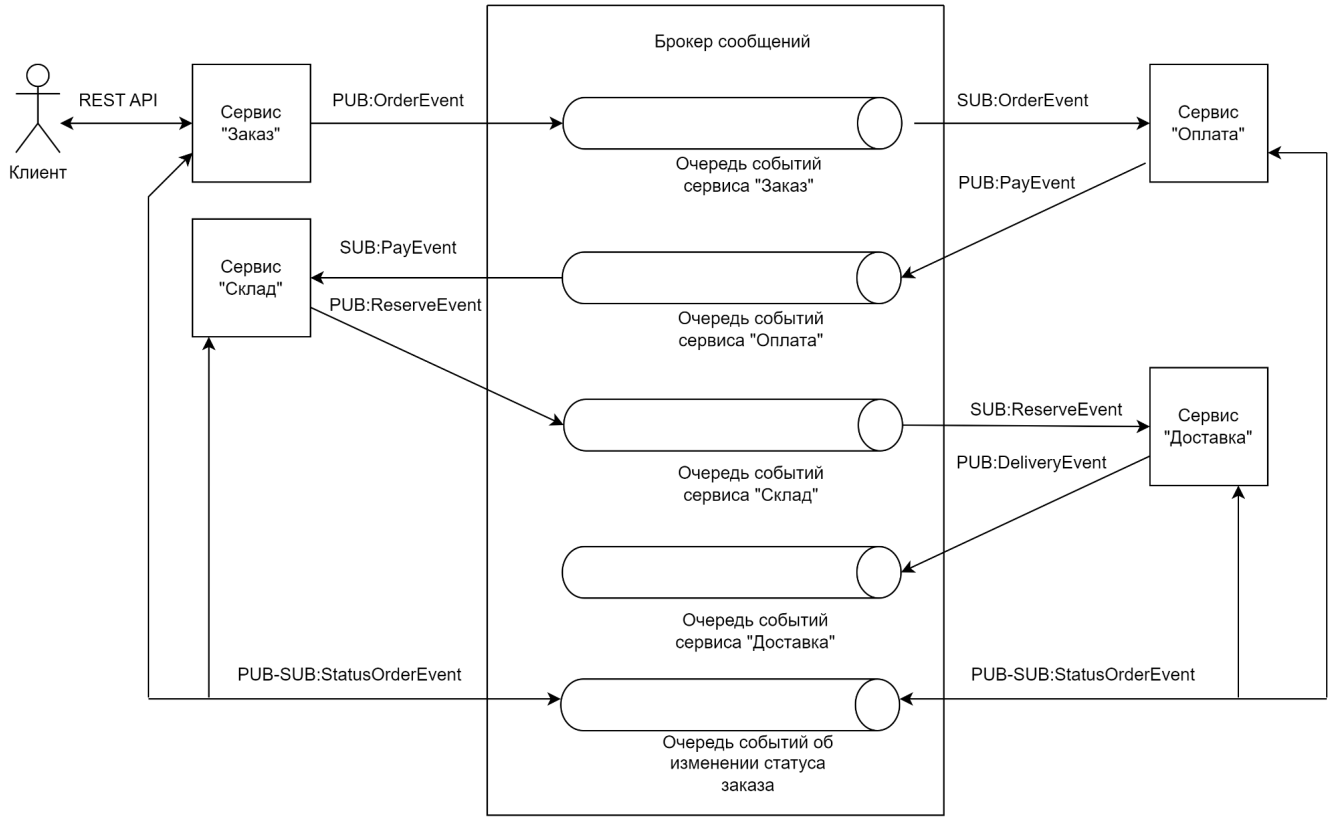
9. **Когда** произойдет ошибка в процессе резервирования на складе
   Тогда статус заказа обновится на “Отменен”
   И будет произведен возврат оплаты

10. **Когда** произойдет ошибка в процессе доставки
    Тогда статус заказа обновится на “Отменен”
    И будет произведен возврат оплаты
    И будет создано поручение в службу доставки на возврат заказа на склад

2. Схема взаимодействия сервисов

Взаимодействие сервисов будем реализовывать используя событийно-ориентированный подход. Каждый сервис будет обмениваться данными(событиями) с каждым через брокер сообщений. Это дает следующие преимущества:

1. Простота горизонтального масштабирования. Если нагрузка на один из сервисов будет слишком велика, то всегда можно добавить еще один экземпляр сервиса, при этом никаких изменений не понадобиться, просто большее количество сервисов будут вычитывать сообщений из очереди, что ускорит обработку событий.
2. Сервисы ничего не знают друг о друге, общаются посредством брокера сообщений, как следствие - нет необходимости в реализации механизма обнаружения сервисов(service discovery)
3. Нет единого центра и как следствие единой точки отказа (за исключением самого брокера сообщений, но мы полагаемся на его стабильность)
4. Простота оповещения всех сервисов о необходимости отката своих действий в случае каких-либо ошибок выполнения (реализация паттерна saga)



Как видно из рисунка у каждого сервиса есть своя очередь сообщений, в которую он публикует данные. После выполнения определенных действий каждый сервис пишет в свою очередь результат выполнения этих действий. Другие сервисы подписываются на конкретные очереди и слушают/ждут события из этой очереди. Так же есть одна общая очередь сообщений “Очередь событий об изменении статуса заказа” в неё пишут все сервисы и на неё подписаны все сервисы. Это позволяет своевременно оповестить все сервисы об изменении статуса заказа. Например если в процессе создания заказа зарезервировать товар на складе не удалось (по причине его отсутствия), то сервис “Склад” меняет статус заказа на “заказ отменен” при этом каждый сервис получает соответствующее событие об изменении статуса заказа и выполняет соответствующее компенсирующее изменение (например сервис “Оплата” выполняет возврат денежных средств клиенту совершившему заказ)

3. Зона ответственности и назначение сервисов (Microservices Canvas)

Название	Заказ	
Описание	Создание, отмена заказа, получение списка заказов. Удаление заказа не предполагается.	
API		
Команды	**Запросы**	**События**
Синхронные: - POST api/v1/orders/create - POST api/v1/orders/cancel	- GET api/v1/orders/list	- OrderEvent - StatusOrderEvent
Зависимости		
Вызывает	**Подписан на**	
MQ Orders: - OrderEvent MQ StatusOrder: - StatusOrderEvent	MQ StatusOrder: - StatusOrderEvent	

Название	Оплата	
Описание	Выполнение оплаты. При успешном выполнении меняет статус заказа на “оплачен”, в противном случае на “отменен”	
API		
Команды	**Запросы**	**События**
		- PayEvent - StatusOrderEvent
Зависимости		
Вызывает	**Подписан на**	
MQ Pay: - PayEvent MQ StatusOrder: - StatusOrderEvent	MQ Orders: - OrderEvent MQ StatusOrder: - StatusOrderEvent	

Название	Склад	
Описание	Выполнение резервирования товаров из заказа на складе. При успешном выполнении меняет статус заказа на “зарезервирован”, в противном случае на “отменен”.	
API		
Команды	**Запросы**	**События**
		- ReserveEvent - StatusOrderEvent
Зависимости		
Вызывает	**Подписан на**	
MQ Stock: - ReserveEvent MQ StatusOrder: - StatusOrderEvent	MQ Pay: - PayEvent MQ StatusOrder: - StatusOrderEvent	

Название	Доставка	
Описание	Выполнение создания поручения на доставку. При успешном выполнении меняет статус заказа на “передан в доставку”/”доставлен”, в противном случае на “отменен”.	
API		
Команды	**Запросы**	**События**
		- DeliveryEvent - StatusOrderEvent
Зависимости		
Вызывает	**Подписан на**	
MQ StatusOrder: - StatusOrderEvent	MQ Stock: - ReserveEvent MQ StatusOrder: - StatusOrderEvent	

4. Контракты (REST API сервиса заказов)

Ссылка на Swagger API - https://github.com/julinserg/otus-microservice-hw/blob/main/hw07_docs/openapi.yaml

POST /create Создать заказ

Parameters | Try it out

Name	Description
currentUserId * required string *(query)*	ID пользователя, который выполняет запрос. currentUserId

Request body | application/json

Example Value | Schema


```
{
  "id": "579182b1-6537-48c9-b1b6-8105ec1ceb39",
  "products": {
    "option": [
      {
        "id": "579182b1-6537-48c9-b1b6-8105ec1ceb39",
        "name": "Товар1",
        "price": 123
      }
    ]
  },
  "shippingTo": "ул. Ленина, д. 12, кв. 51",
  "cardParams": "1234 5678 1234 5678; 345; Ivan Ivanov",
  "status": "CREATED"
}
```


Responses

Code	Description	Links
200	Операция выполнена	*No links*

POST `/cancel` Отменить заказ

Parameters | Try it out

Name	Description
currentUserId * required `string` *(query)*	ID пользователя, который выполняет запрос. currentUserId
orderID * required `string` *(query)*	ID заказа. orderID

Responses

Code	Description	Links
200	Операция выполнена	*No links*
400	Некорректные параметры запроса	*No links*

GET `/list` Получить список заказов

Parameters | Try it out

Name	Description
currentUserId * required `string` *(query)*	ID пользователя, который выполняет запрос. currentUserId

Responses

Code	Description	Links
200	Операция выполнена Media type application/json Controls Accept header. Example Value | Schema	*No links*


```
[
  {
    "id": "579182b1-6537-48c9-b1b6-8105ec1ceb39",
    "products": {
      "option": [
        {
          "id": "579182b1-6537-48c9-b1b6-8105ec1ceb39",
          "name": "Товар1",
          "price": 123
        }
      ]
    },
    "shippingTo": "ул. Ленина, д. 12, кв. 51",
    "cardParams": "1234 5678 1234 5678; 345; Ivan Ivanov",
    "status": "CREATED"
  }
]
```