Интернет магазин

Локальный адрес интернетмагазина: {base_url} = http://marketplace.local

Задание

Имеется web приложение (интернет магазин) которое состоит из следующих компонентов:

- Kafka
- Backend приложение
- Сервер БД MySQL
- Frontend приложение (SPA)

Описать процесс покупки товара (от корзины пользователя до момента оплаты и получения товара).

Для совершения покупок пользователи авторизуются.

У них есть личный кабинет где они могут посмотреть список купленных товаров.

Для обмена данных между клиентом и сервером, как мы вчера уточнили, используется REST.

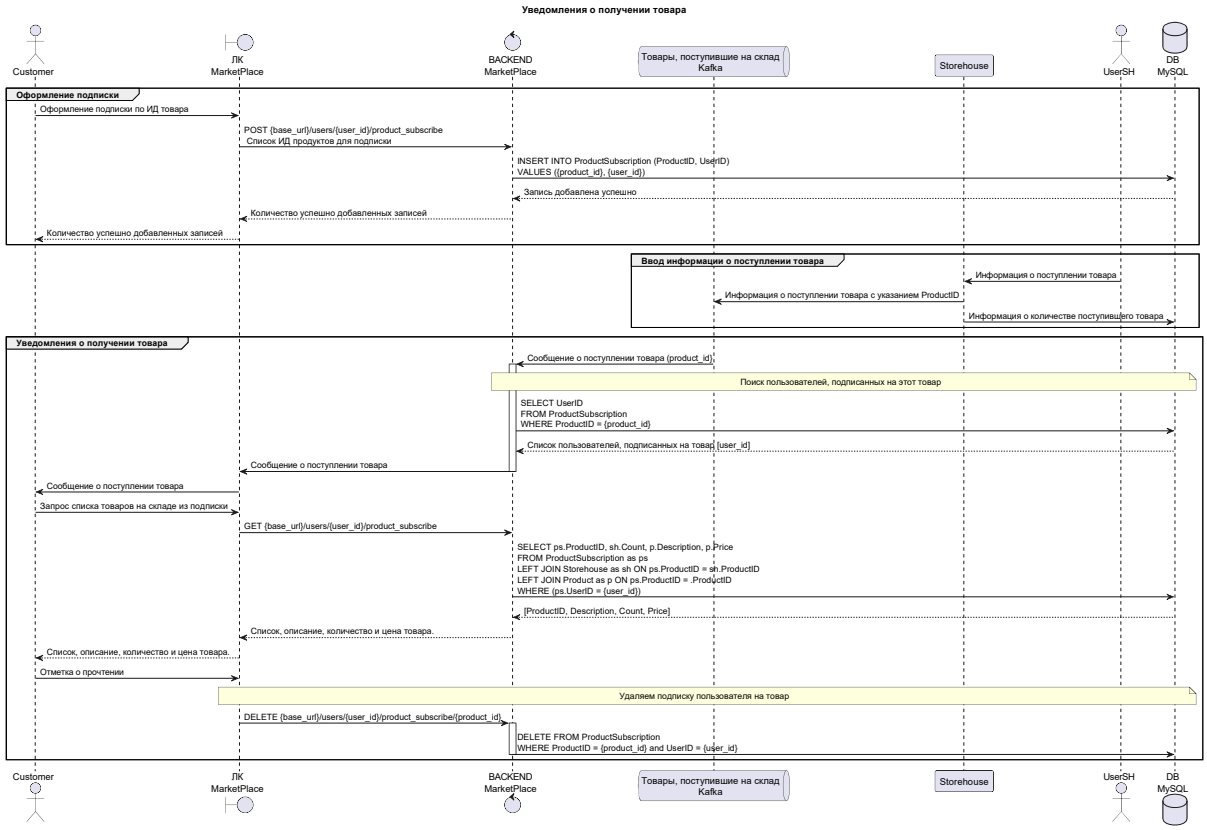
Допустим мы хотим расширить функционал данного магазина таким образом:

- Если пользователь выбирает товар для покупки а товара нет на складе мы должны давать ему возможность для подписки на событие поступления товара на склад.
- По факту поступления товара на склад пользователь должен получить уведомление в личном кабинете.
- После того как уведомление будет прочитано его статус должен переходить на прочитано (read).
- У пользователя должно быть список своих уведомлений.

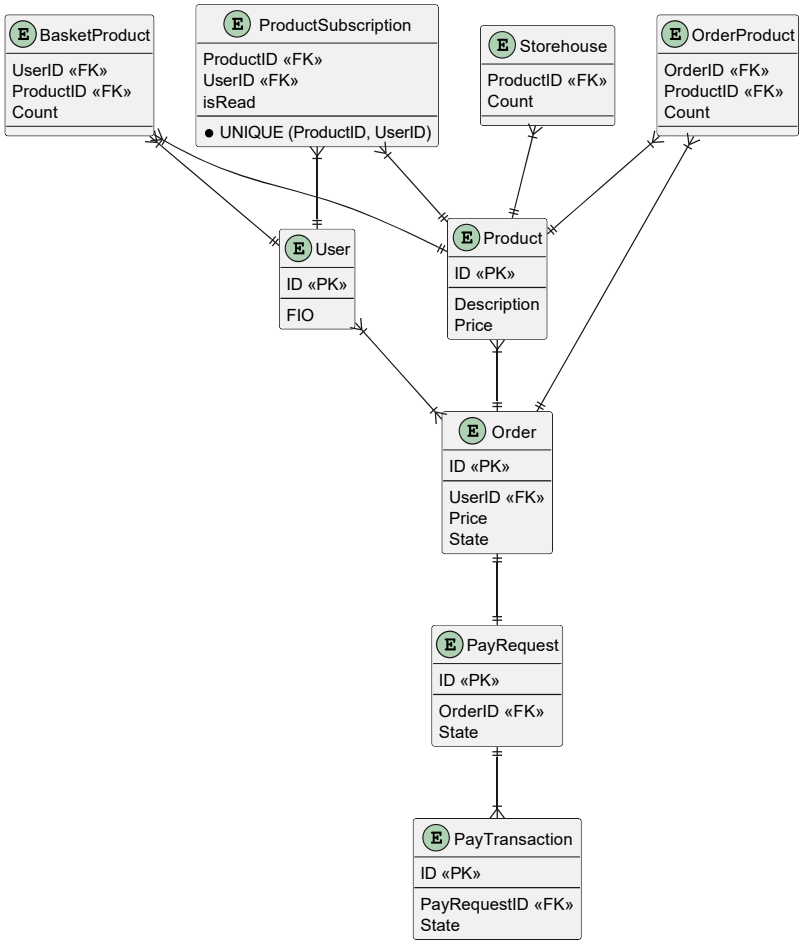
Требуется системный анализ для реализации данного функционала и по его итогам подготовить FS задачу для разработчика.

PS. Нужно подчеркнуть и описать все необходимые и важные процессы как мы вчера обсуждали во время собеседования.

Схема работы с уведомлениями о поступлении товара



ERD-схема таблиц БД



User (Пользователи)

Поле	Тип	Обязательно	Описание
ID	int	PK	ИД пользователя
FIO	string	+	ФИО пользователя

Product (Товары)

Поле	Тип	Обязательно	Описание
ID	int	PK	ИД товара
Price	double	+	Цена товара за ед. продукции в рублях
Description	string	+	Описание товара

ProductSubscription (Подписки пользователей на приход товара на склад)

Поле	Тип	Обязательно	Описание
ProductID	int	+	FK на таблицу **Product (Товары)**
UserID	int	+	FK на таблицу **User (Пользователи)**

В таблице должно действовать следующее ограничение (CONSTRAINT): `UNIQUE (ProductID, UserID)`.

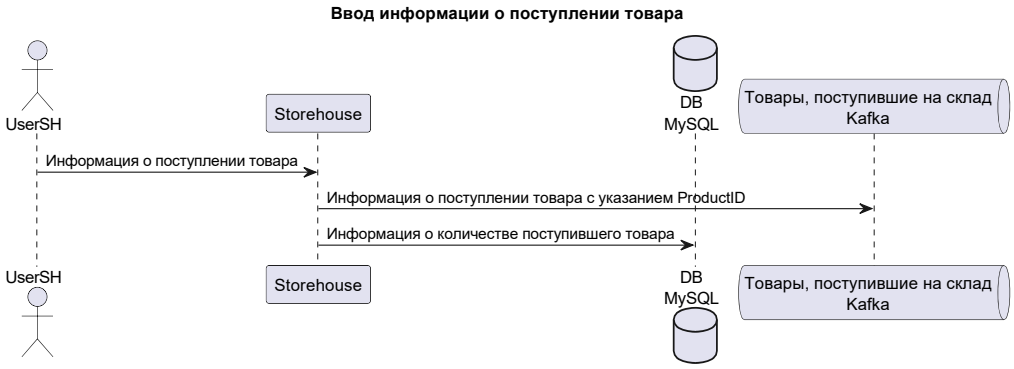
Storehouse (Склад товара)

Поле	Тип	Обязательно	Описание
ProductID	int	+	FK на таблицу **Product (Товары)**

Поле	Тип	Обязательно	Описание
Count	int	+	Количество товара на складе

Ввод информации о поступлении товара

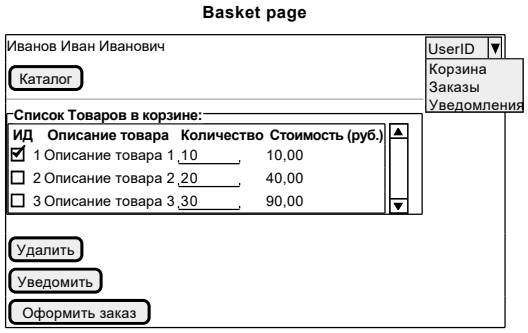
Ввод информации о поступлении товара осуществляется складским работником (на схеме `UserSH`) в собственной складской программе (на схеме `Storehouse`). При этом в топик `Товары, поступившие на склад` добавляется сообщение содержащее ProductID, а в таблице Storehouse (Склад товара) появляется запись с указанием количества.



Оформление подписки

Frontend: Страница Basket

Оформление подписки на поступление продуктов осуществляется пользователем в окне Страница Basket (адрес: `{base_url}/{user_id}/basket`). Путем выбора товара в таблице **Список товаров в корзине** и нажатия на кнопку **Уведомить**.



После нажатия на кнопку **Уведомить** система отправляет REST-запрос `POST {base_url}/users/{user_id}/product_subscribe` (метод: `addProductSubscribeForUser`).

REST API: POST /users/(user_id)/product_subscribe

`POST {base_url}/users/{user_id}/product_subscribe` - метод добавляет массив товаров в таблицу ProductSubscription (Подписки пользователей на приход товара на склад).

Описание BACKEND: addProductSubscribeForUser

Запрос

Поле	Тип	Обязательно	Описание
path			
user_id	int	+	ИД пользователя
body			
products	int[]	+	Массив ИД товаров для добавления в список отслеживаемых пользователем.

Ответ 200

Поле	Тип	Обязательно	Описание
body			

added_row_count	int	+	Количество успешно добавленных записей

Ответ 400

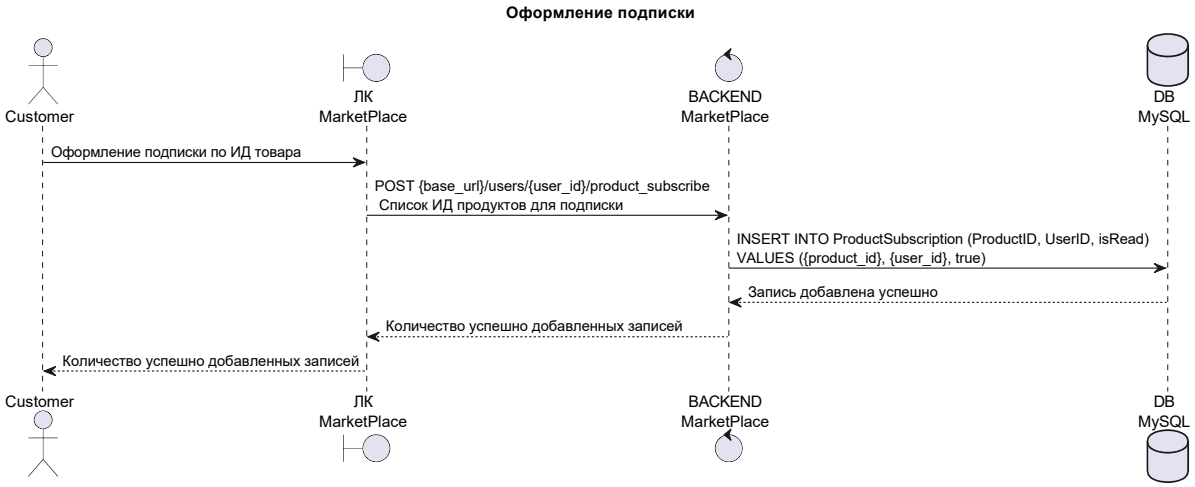
Ошибка входных параметров.

Backend: addProductSubscribeForUser

Входные параметры

Поле	Тип	Обязательно	Описание
user_id	int	+	ИД пользователя
products	int[]	+	Массив ИД товаров для добавления в список отслеживаемых пользователем.

Алгоритм



1. При получении запроса система проверяет корректность заполнения входных параметров (обязательность и тип). Если входные параметры не верны возвращается **ошибка 400**.
2. Если входные параметры верны, система для каждого элемента массива `products` входных параметров добавляет запись в таблицу ProductSubscription (Подписки пользователей на приход товара на склад), где поле `UserID` равно входному параметру `user_id`, а поле `ProductID` - очередному значению из массива `products`. SQL-запрос : `INSERT INTO ProductSubscription (ProductID, UserID, isRead) VALUES ({product_id}, {user_id}, true)`. Ошибки добавления записей, связанные с ограничением `UNIQUE (ProductID, UserID)` игнорируются. Количество успешно добавленных записей считается и возвращается в виде выходного параметра `added_row_count`.

Выходные параметры

Поле	Тип	Обязательно	Описание
added_row_count	int	+	Количество успешно добавленных записей

Уведомления о поступлении товара

Появление уведомлений в ЛК пользователей о появлении на складе интересующих их товаров инициируется появлением в топике `Товары, поступившие на склад` сервера Kafka.

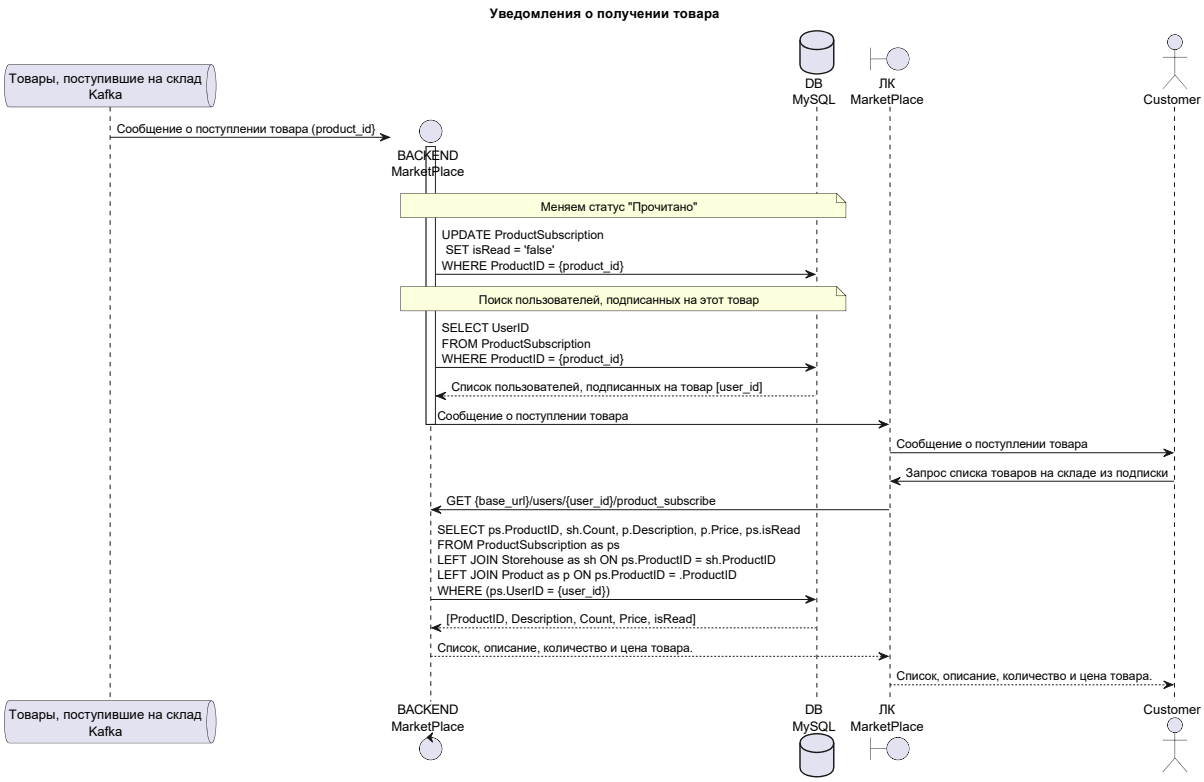
1. Получив сообщение система получает список пользователей, подписанных на поступление этого продукта выполнением метода `messageGenerator` с параметром `product_id`.
2. Для каждого полученного на предыдущем шаге пользователя инициируется активация знака поступивших уведомлений.

Backend: messageGenerator

Входные параметры

Поле	Тип	Обязательно	Описание
product_id	int	+	ИД товара поступившего на склад

Алгоритм

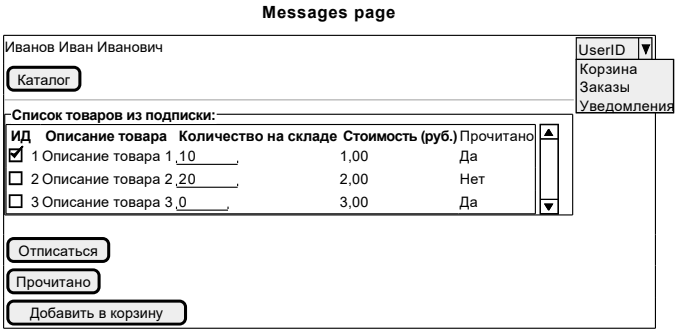


1. Система по `product_id` входных параметров выбирает записи из таблицы ProductSubscription (Подписки пользователей на приход товара на склад), где поле `ProductID` равно входному параметру `product_id` . SQL-запрос : `SELECT UserID FROM ProductSubscription WHERE ProductID = {product_id}` .
2. Для каждого найденного на предыдущем шаге UserID система генерирует уведомление в ЛК о поступелении товара.

Просмотр уведомлений о поступлении товара

Frontend: Страница Messages

Просмотр уведомлений о поступлении товара осуществляется пользователем в окне Страница Messages (адрес: `{base_url}/{user_id}/messages`).



При загрузке страницы система отправляет REST-запрос `GET {base_url}/users/{user_id}/product_subscribe` (метод: `getProductSubscribeForUser`). Результат работы метода отображается в таблице `Список товаров из подписки` .

При нажатии на кнопку `Добавить в корзину` система отправляет REST-запрос `POST {base_url}/users/{user_id}/basket_product` (метод: `addProductToUserBasket`) в качестве параметров передается массив `ProductID` записей, выделенных в таблице `Список товаров из подписки` .

REST API: GET

`GET {base_url}/users/{user_id}/product_subscribe` - метод получает описание товаров, указанных в таблице ProductSubscription (Подписки пользователей на приход товара на склад).

Описание BACKEND: getProductSubscribeForUser

Запрос

Поле	Тип	Обязательно	Описание
path			
user_id	int	+	ИД пользователя

Ответ 200

Поле	Тип	Обязательно	Описание
products	object[]	+	Массив с писанием товара
product_id	int	+	ИД Товара
description	string	+	Описание товара
count	int	+	Количество товара на складе
price	double	+	Цена товара
isRead	bolean	+	Статус (Не)прочитано

Ответ 400

Ошибка входных параметров.

REST API: POST

`POST {base_url}/users/{user_id}/basket_product` - метод добавляет в таблицу BasketProduct (Товары в корзине), отмеченные в окне записи.

Описание BACKEND: addProductToUserBasket

Запрос

Поле	Тип	Обязательно	Описание
path			
user_id	int	+	ИД пользователя
body			
products	object[]	+	Массив добавляемых товаров
product_id	int	+	ИД товара
count	int	+	Количество товара

Ответ 200

Поле	Тип	Обязательно	Описание
products	object[]	+	Массив с писанием товара
product_id	int	+	ИД Товара
description	string	+	Описание товара
count	int	+	Количество товара на складе
price	double	+	Цена товара
isRead	bolean	+	Статус (Не)прочитано

Ответ 400

Ошибка входных параметров.

Backend: getProductSubscribeForUser

Входные параметры

Поле	Тип	Обязательно	Описание
user_id	int	+	ИД пользователя

Алгоритм

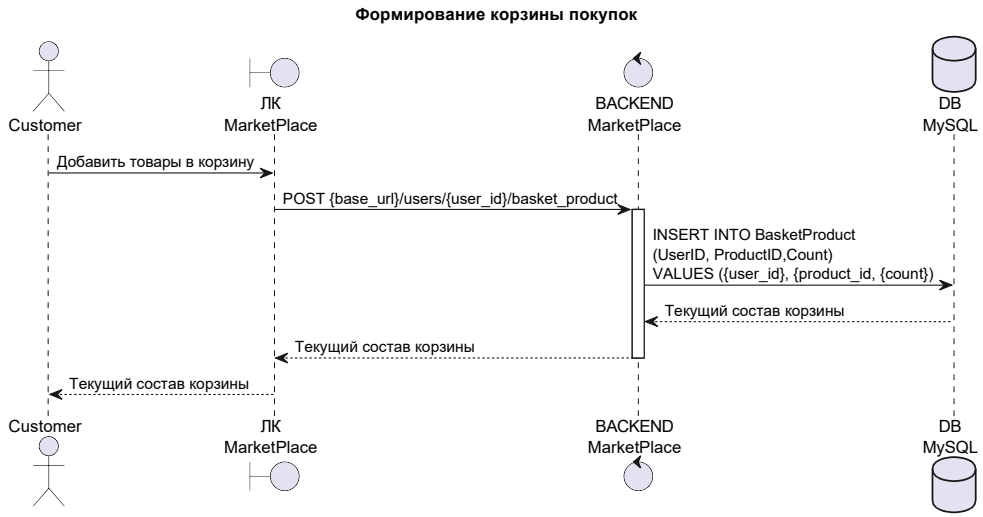
1. Система по `user_id` входных параметров выбирает записи и выводит в качестве выходных параметров из таблиц ProductSubscription (Подписки пользователей на приход товара на склад), Product (Товары), Storehouse (Склад товара), где поле `UserID` равно входному параметру `user_id`, а поля связанных таблиц `Storehouse.ProductID` и `Product.ID` полю `ProductID` таблицы `ProductSubscription`. SQL-запрос :

```
SELECT ps.ProductID, sh.Count, p.Description, p.Price, ps.isRead FROM ProductSubscription as ps LEFT JOIN Storehouse as sh (
```

Выходные параметры

Поле	Тип	Обязательно	Описание
products	object[]	+	Массив с писанием товара
product_id	int	+	ИД Товара
description	string	+	Описание товара
count	int	+	Количество товара на складе
price	double	+	Цена товара
isRead	bolean	+	Статус (Не)прочитано

Backend: addProductToUserBasket



Входные параметры

Поле	Тип	Обязательно	Описание
user_id	int	+	ИД пользователя
products	object[]	+	Массив добавляемых товаров
product_id	int	+	ИД товара
count	int	+	Количество товара

Алгоритм

1. При получении запроса система проверяет корректность заполнения входных параметров (обязательность и тип). Если входные параметры не верны возвращается **ошибка 400**.

2. Если входные параметры верны, система для каждого элемента массива `products` входных параметров добавляет запись в таблицу BasketProduct (Товары в корзине), где поле `UserID` равно входному параметру `user_id`, а поле `ProductID` - очередному значению из массива `products`. SQL-запрос :
   `INSERT INTO BasketProduct (ProductID, UserID, Count) VALUES ({product_id}, {user_id}, {count})`.

3. Система по `user_id` входных параметров выбирает записи и выводит в качестве выходных параметров из таблиц BasketProduct (Товары в корзине)), Product (Товары), Storehouse (Склад товара), где поле `UserID` равно входному параметру `user_id`, а поле связанной таблицы `Product.ID` полю `ProductID` таблицы `ProductSubscription`. SQL-запрос :

```
SELECT ps.ProductID, p.Description, p.Price, ps.Count FROM BasketProduct as ps LEFT JOIN Product as p ON ps.ProductID = p.P
```

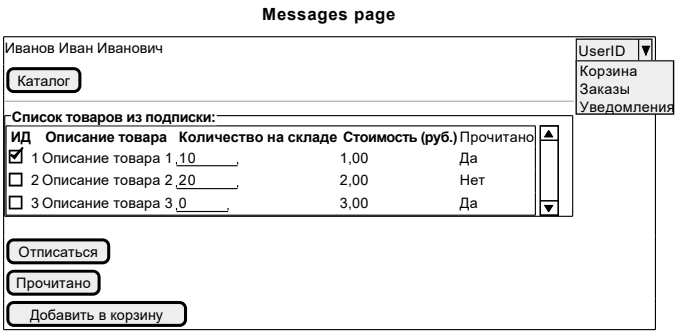
Выходные параметры

Поле	Тип	Обязательно	Описание
products	object[]	+	Массив с писанием товара
product_id	int	+	ИД Товара
description	string	+	Описание товара
count	int	+	Количество товара на складе
price	double	+	Цена товара
count	int	+	Количество

Отметка о прочтении уведомления

Frontend: Страница Messages

Отметка о прочтении уведомления осуществляется пользователем в окне Страница Messages (адрес: `{base_url}/{user_id}/messages`) нажатием кнопки в колонке `Прочитано` в таблице `Список товаров из подписки` в качестве параметров передается массив `ProductID`, соответствующеей записи в таблице `Список товаров из подписки` и новое значение (`true/false`) для поля isRead.



При нажатии на кнопку `Прочитано` система отправляет REST-запрос `PUT {base_url}/users/{user_id}/product_subscribe/{product_id}` (метод: `changeReadStatusUserSubscribe`) в качестве параметров передается массив `ProductID` записей, выделенных в таблице `Список товаров из подписки`.

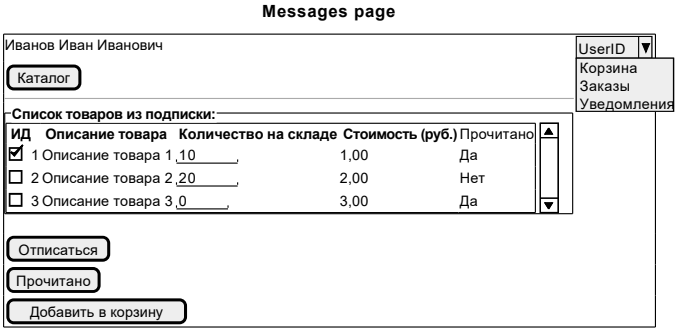
REST API: PUT

Backend: changeReadStatusUserSubscribe

Удаление подписки на товар

Frontend: Страница Messages

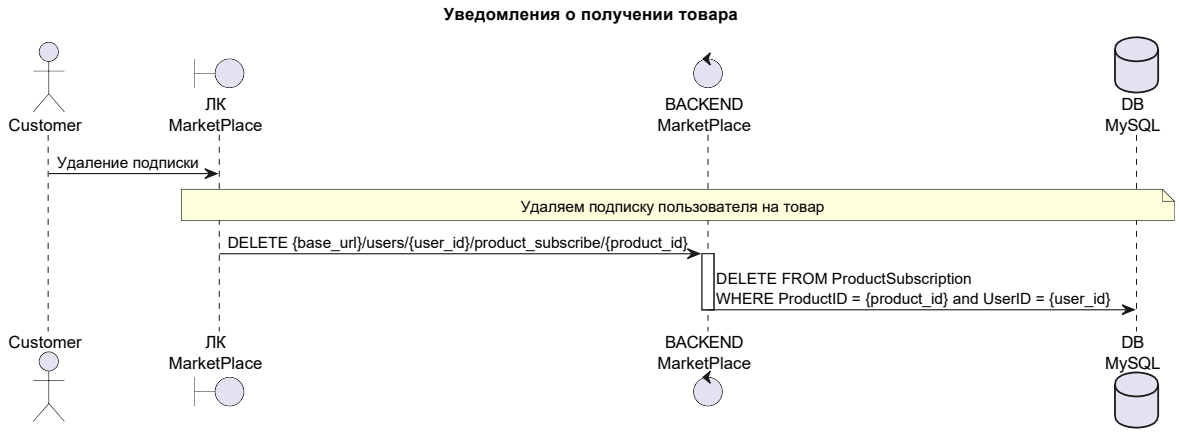
Удаление подписки на уведомления о поступлении товара осуществляется пользователем в окне Страница Messages (адрес: `{base_url}/{user_id}/messages`).



При нажатии на кнопку `Отписаться` система отправляет REST-запрос `DELETE {base_url}/users/{user_id}/product_subscribe` (метод: `deleteUserSubscribeByProductId`) в качестве параметров передается массив `ProductID` записей, выделенных в таблице `Список товаров из подписки`.

REST API: DELETE

Backend: deleteUserSubscribeByProductId



Фронт

Общие элементы управления

Нажатие кнопки ***Каталог*** отправляет пользователя на страницу каталога (**1.3 Страница Catalog**).

Выбор меню ***Корзина*** отправляет пользователя на страницу каталога (**1.2 Страница Basket**).

Выбор меню ***Заказы*** отправляет пользователя на страницу каталога (**1.4 Страница Orders**).

Выбор меню ***Уведомления*** отправляет пользователя на страницу каталога (**1.5 Страница Messages**).

Общие методы экранных форм

Получение реквизитов пользователя

Страница Wellcome

Адрес страницы: `{base_url}`

1.1.1 API страницы Wellcome

Используемые методы:

- Получение реквизитов пользователя: `GET {base_url}/users/{user_id}`
- Список предыдущих покупок пользователя: `GET {base_url}/users/{user_id}/products`

Элементы управления:

В таблицу ***Список предыдущих покупок*** загружаются результаты вызова метода `GET {base_url}/users/{user_id}/products`.

При загрузке страницы вызываются последовательно методы получения реквизитов пользователя (`GET {base_url}/users/{user_id}`) и список его предыдущих покупок (`GET {base_url}/users/{user_id}/products`).

Страница Basket

Адрес страницы: `{base_url}/{user_id}/basket`

Basket page

1.2.1 API страницы Basket

Используемые методы:

- Получение реквизитов пользователя: `GET {base_url}/users/{user_id}`
- Список товаров в корзине: `GET {base_url}/users/{user_id}/basket`
- Удалить выбранные позиции из корзины: `DELETE {base_url}/users/{user_id}/basket`
- Оформить заказ: `POST {base_url}/users/{user_id}/orders`

Элементы управления:

В таблицу ***Список товаров в корзине*** загружаются результаты вызова метода `GET {base_url}/users/{user_id}/basket`.

Нажатие кнопки ***Удалить*** вызывает метод `DELETE {base_url}/users/{user_id}/basket`, после чего происходит обновления страницы.

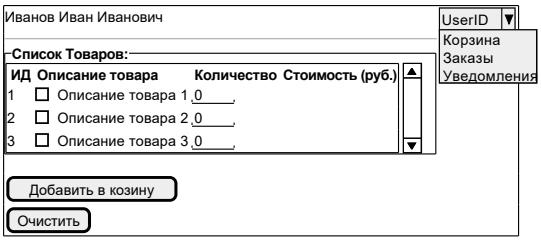
Нажатие кнопки ***Оформить заказ*** вызывает метод `POST {base_url}/users/{user_id}/orders`, после чего происходит переход на страницу каталога (**1.4 Страница Orders**).

1.2.2 BACKEND методов страницы Basket

При загрузке страницы вызываются последовательно методы получения реквизитов пользователя (`GET {base_url}/users/{user_id}`) и список товаров в его корзине (`GET {base_url}/users/{user_id}/basket`).

1.3 Страница Catalog

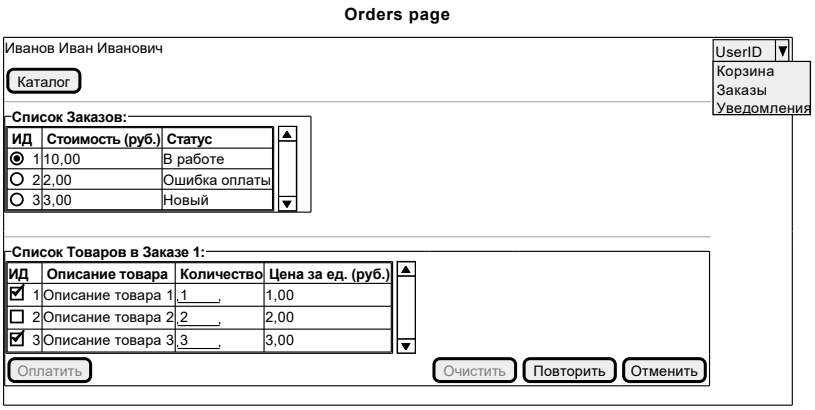
Catalog page



1.3.1 API страницы Catalog

1.3.2 BACKEND методов страницы Catalog

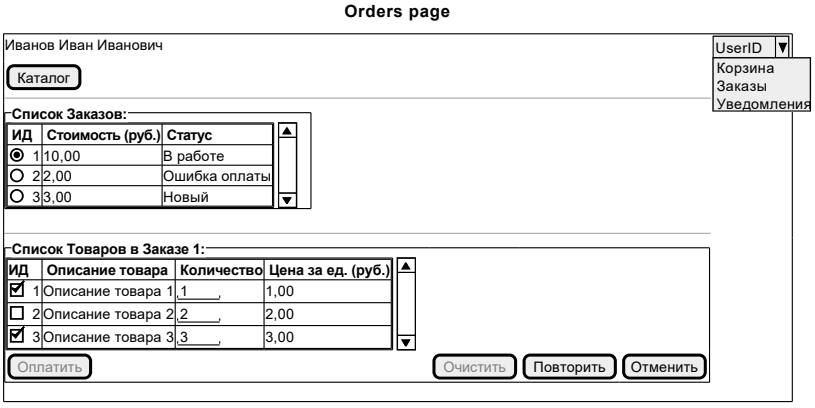
1.4 Страница Orders



1.4.1 API страницы Orders

1.4.2 BACKEND методов страницы Orders

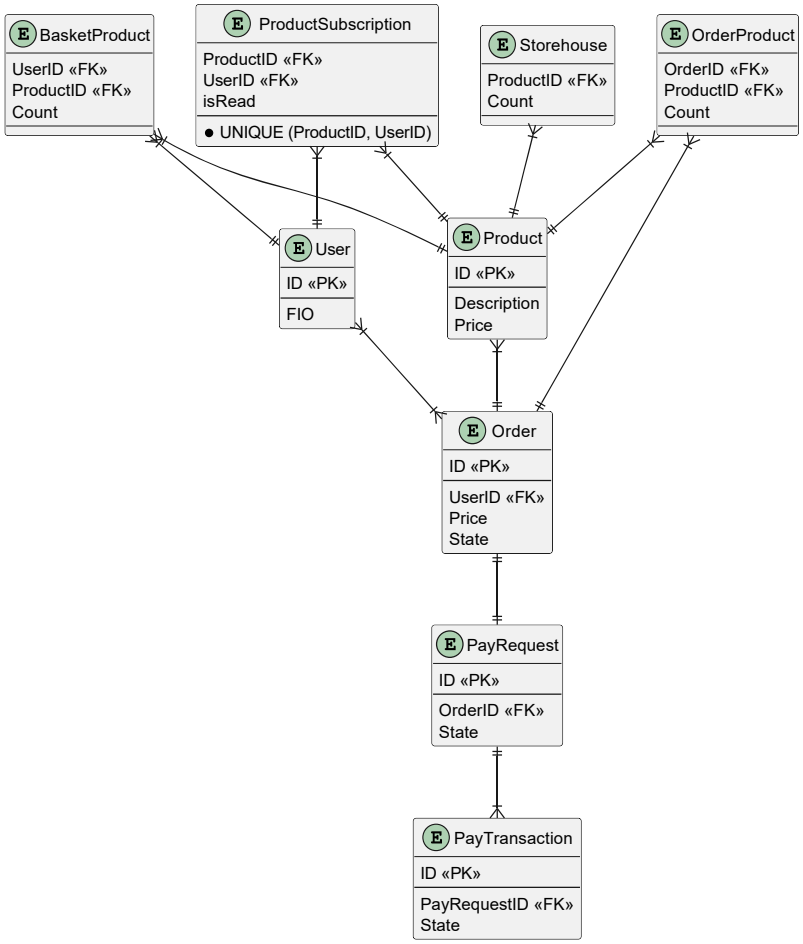
1.5 Страница Messages



1.5.1 API страницы Messages

1.5.2 BACKEND методов страницы Messages

ERD-схема таблиц БД



User (Пользователи)

Поле	Тип	Обязательно	Описание
ID	int	PK	ИД пользователя
FIO	string	+	ФИО пользователя

Order (Заказы)

Поле	Тип	Обязательно	Описание
ID	int	PK	ИД заказа
UserID	int	+	FK на таблицу **User (Пользователи)**
Price	double	+	Суммарная цена товара в заказе в рублях
State	string	+	Состояние заказа. Может принимать значения: • 'new' - Новый • 'checked' - Проверено • 'work' - В работе • 'reserved' - Товар зарезервирован • 'error' - Ошибка оплаты • 'paid' - Оплачено • 'canceled' - Отменено

Product (Товары)

Поле	Тип	Обязательно	Описание
ID	int	PK	ИД товара

Поле	Тип	Обязательно	Описание
Price	double	+	Цена товара за ед. продукции в рублях
Description	string	+	Описание товара

BasketProduct (Товары в корзине)

Поле	Тип	Обязательно	Описание
UserID	int	+	FK на таблицу **User (Пользователи)**
ProductID	int	+	FK на таблицу **Product (Товары)**
Count	int	+	Количество товара в корзине

OrderProduct (Товары в заказе)

Поле	Тип	Обязательно	Описание
OrderID	int	+	FK на таблицу **Order (Заказы)**
ProductID	int	+	FK на таблицу **Product (Товары)**
Count	int	+	Количество товара в корзине

PayRequest (Запросы на оплату)

Поле	Тип	Обязательно	Описание
ID	int	PK	ИД зароса на оплату
OrderID	int	+	FK на таблицу **Order (Заказы)**
State	string	+	Состояние запроса на оплату. Может принимать значения: • 'new' - Новый • 'checked' - Проверено • 'work' - В работе • 'reserved' - Товар зарезервирован • 'error' - Ошибка оплаты • 'paid' - Оплачено • 'canceled' - Отменено

PayTransaction (Транзакции оплат)

Поле	Тип	Обязательно	Описание
ID	int	PK	ИД Транзакции
T_UUID	uuid	-	UUID Транзакции, получаемый с платежного шлюза
PayRequestID	int	+	FK на таблицу **PayRequest (Запросы на оплату)**
State	string	+	Состояние запроса на оплату. Может принимать значения: • 'new' - Новый • 'checked' - Проверено • 'work' - В работе • 'reserved' - Товар зарезервирован • 'error' - Ошибка оплаты • 'paid' - Оплачено • 'canceled' - Отменено

Storehouse (Склад товара)

Поле	Тип	Обязательно	Описание
ProductID	int	+	FK на таблицу **Product (Товары)**
Count	int	+	Количество товара на складе

ProductSubscription (Подписки пользователей на приход товара на склад)

Поле	Тип	Обязательно	Описание
ProductID	int	+	FK на таблицу **Product (Товары)**
UserID	int	+	FK на таблицу **User (Пользователи)**

В таблице должно действовать следующее ограничение (CONSTRAINT): `UNIQUE (ProductID, UserID)` .

3 REST API

3.1 GET

GET /users/(user_id)

`GET /users/{user_id}` - Метод получает реквизиты пользователя по его ИД.

Описание BACKEND: getUserById

Запрос

Поле	Тип	Обязательно	Описание
path			
user_id	int	+	ИД пользователя

Ответ 200

Поле	Тип	Обязательно	Описание
user_id	int	+	ИД пользователя
fio	string	+	ФИО пользователя

Ответ 400

Не правильные входные параметы.

Ответ 404

Пользователь с ИД ***user_id*** не найден.

GET /users/(user_id)/products

`GET {base_url}/users/{user_id}/products` - Метод возвращает список предыдущих покупок пользователя

Описание BACKEND: getProductsByUserId

Метод получает реквизиты пользователя по его ИД.

Описание BACKEND: getUserById

Запрос

Поле	Тип	Обязательно	Описание
path			
user_id	int	+	ИД пользователя

Ответ 200

Поле	Тип	Обязательно	Описание
user_id	int	+	ИД пользователя
fio	string	+	ФИО пользователя

Ответ 400

Не правильные входные параметы.

GET /product

GET /product

Описание BACKEND: getProductCatalog

3.2 POST

POST users/(user_id)/basket_product

POST /users//basket_product

Описание BACKEND: addProductToUserBasket

POST /users/(user_id)/order

POST /users//order

Описание BACKEND: addOrderForUser

POST /orders/(order_id)/pay_request

POST /orders//pay_request

Описание BACKEND: addPayRequestForOrder

POST /users/(user_id)/product_subscribe

`POST {base_url}/users/{user_id}/product_subscribe` - метод добавляет массив товаров в таблицу отслеживамых пользователем продуктов.

Описание BACKEND: addProductSubscribeForUser

Запрос

Поле	Тип	Обязательно	Описание
path			
user_id	int	+	ИД пользователя
body			
products	int[]	+	Массив ИД товаров для добавления в список отслеживаемых пользователем.

Ответ 200

Поле	Тип	Обязательно	Описание
body			
added_row_count	int	+	Количество успешно добавленных записей

Ответ 400

Ошибка входных параметров.

3.3 DELETE

DELETE /orders/(order_id)

DELETE /orders/

Описание BACKEND: deleteOrderById

DELETE /users/(user_id)/product_subscribe/(product_id)

DELETE /users//product_subscribe/

Описание BACKEND: deleteUserSubscribeByProductId

4 BACKEND

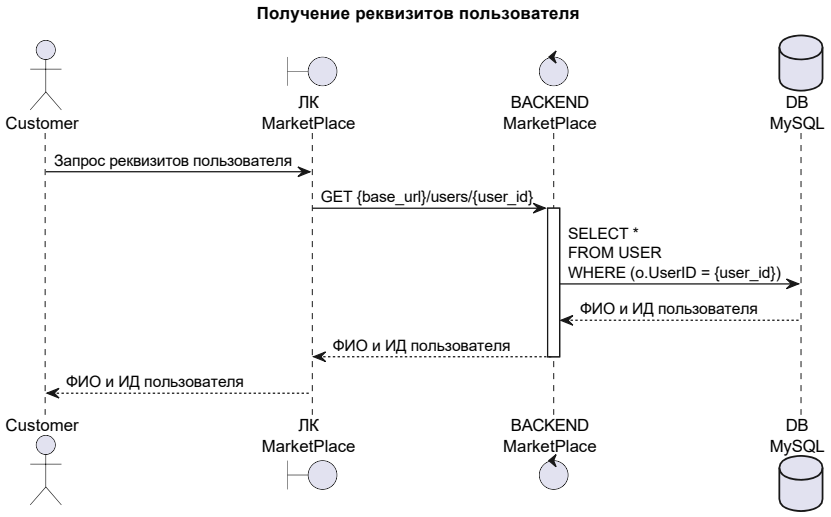
getUserById

Метод возвращает реквизиты пользователя по его ИД.

Входные параметры

Поле	Тип	Обязательно	Описание
user_id	int	+	ИД пользователя

Алгоритм



1. При получении запроса система проверяет корректность заполнения входных параметров (обязательность и тип). Если входные параметры не верны возвращается **ошибка 400**.
2. Если входные параметры верны, система запрашивает записи таблицы БД User (Пользователи) у которых `UserID` равен входному параметру `user_id`. SQL-запрос : `SELECT * FROM User WHERE UserID = {user_id}`
3. Если запрос не вернул ни одной записи возвращается **ошибка 404**.
4. Если запрос вернул запись с реквизитами пользователя система возвращает результат в виде ввыходных параметров.

Выходные параметры

Поле	Тип	Обязательно	Описание
user_id	int	+	ИД пользователя
fio	string	+	ФИО пользователя

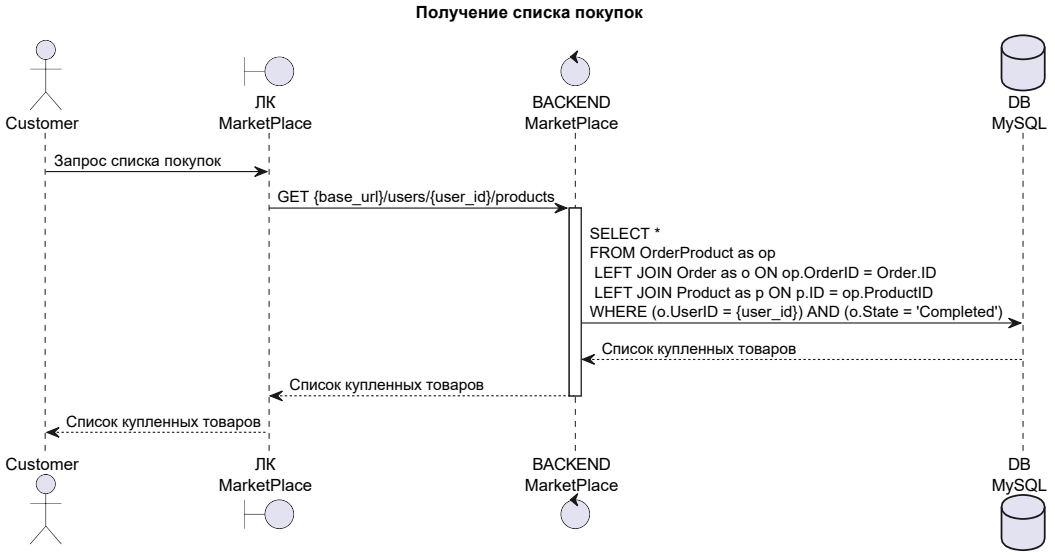
getProductsByUserId

Метод возвращает список купленных ранее продуктов пользователя по его ИД.

Входные параметры

Поле	Тип	Обязательно	Описание
user_id	int	+	ИД пользователя

Алгоритм

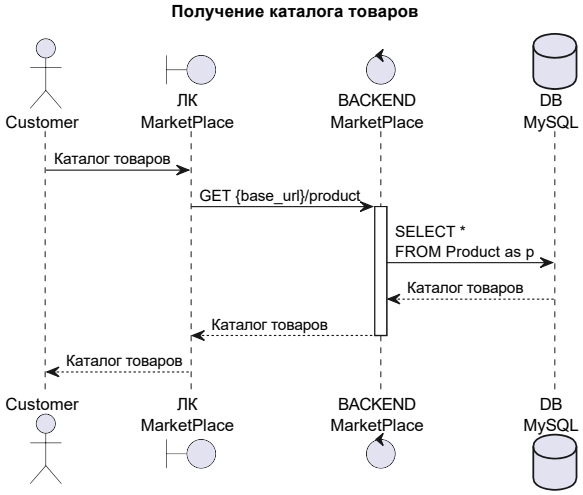


1. При получении запроса система проверяет корректность заполнения входных параметров (обязательность и тип). Если входные параметры не верны возвращается**ошибка 400**.
2. Если входные параметры верны, система запрашивает записи таблицы БД User (Пользователи) у которых `UserID` равен входному параметру `user_id`. SQL-запрос : `SELECT * FROM User WHERE UserID = {user_id}`
3. Если запрос не вернул ни одной записи возвращается **ошибка 404**.
4. Если запрос вернул запись с реквизитами пользователя система возвращает результат в виде ввыходных параметров.

Выходные параметры

Поле	Тип	Обязательно	Описание
user_id	int	+	ИД пользователя
fio	string	+	ФИО пользователя

getProductCatalog

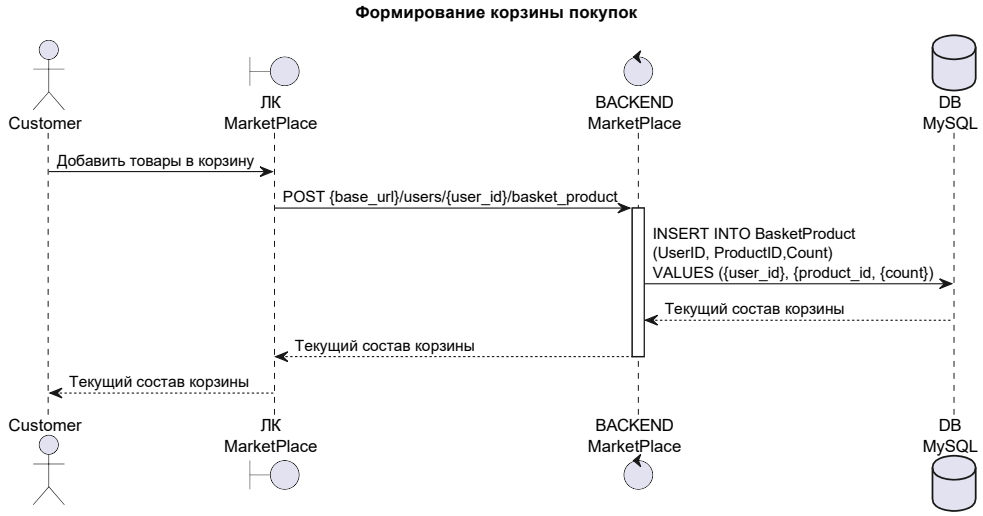


Входные параметры

Алгоритм

Выходные параметры

addProductToUserBasket

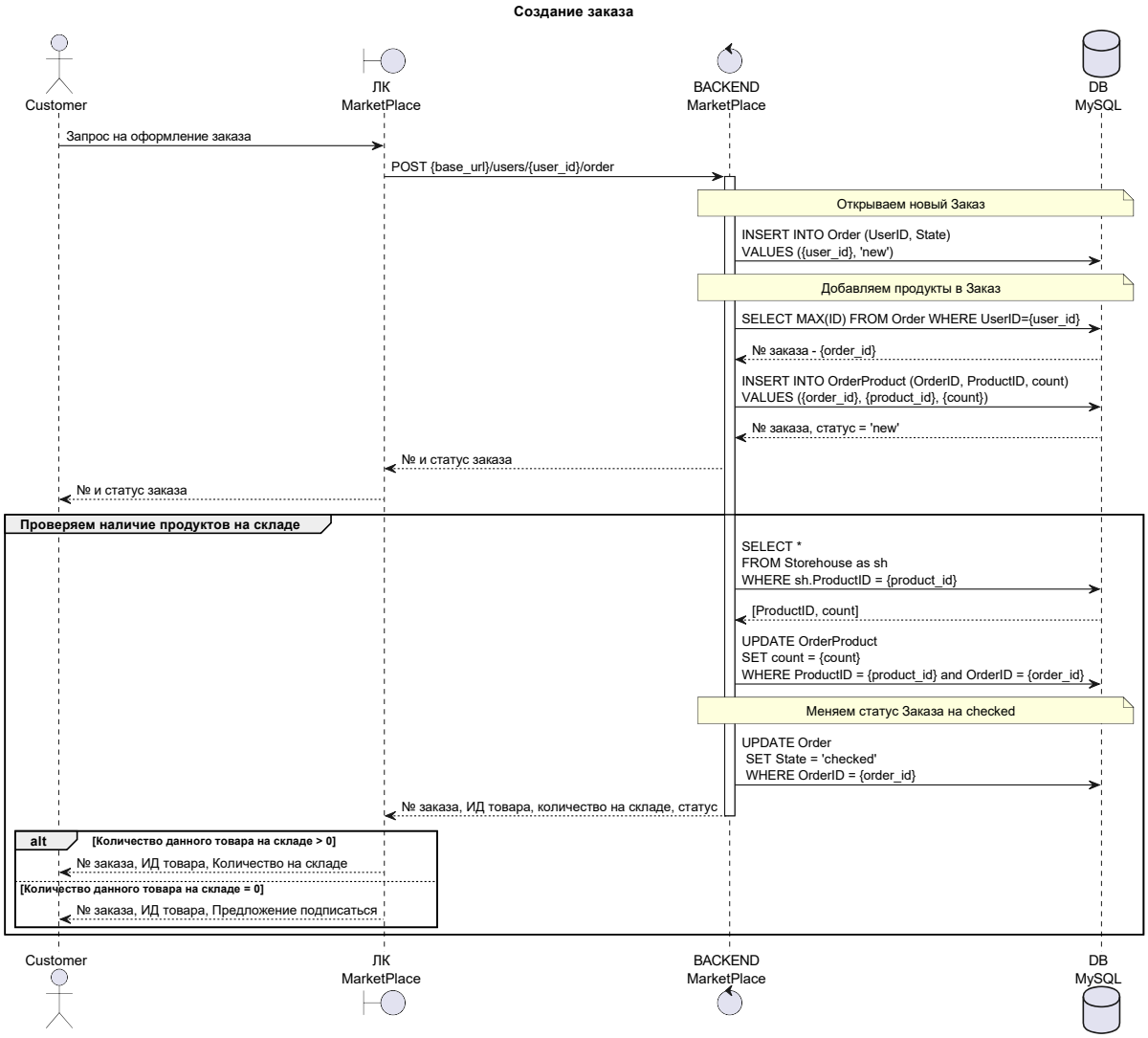


Входные параметры

Алгоритм

Выходные параметры

addOrderForUser

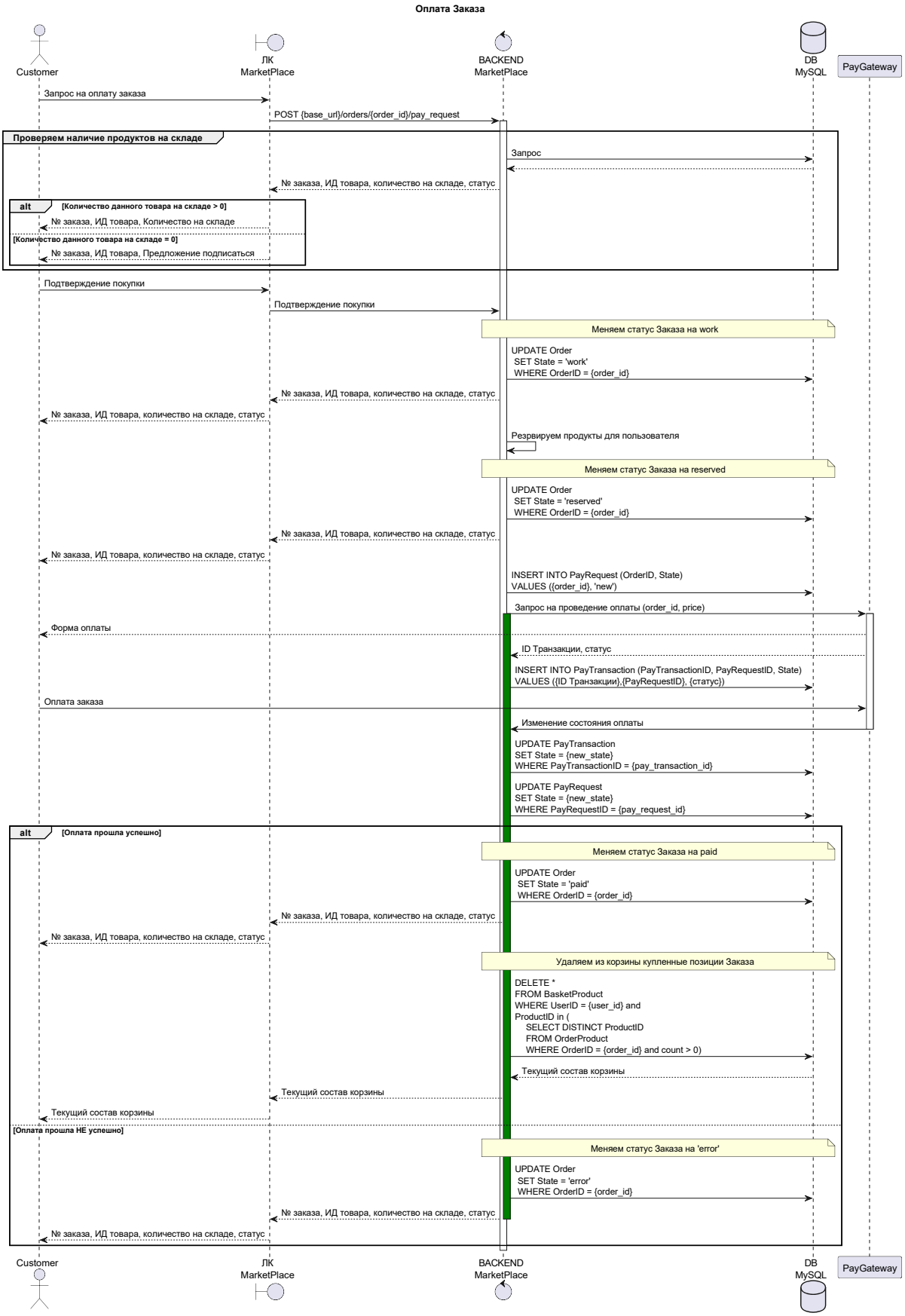


Входные параметры

Алгоритм

Выходные параметры

addPayRequestForOrder



Входные параметры

Алгоритм

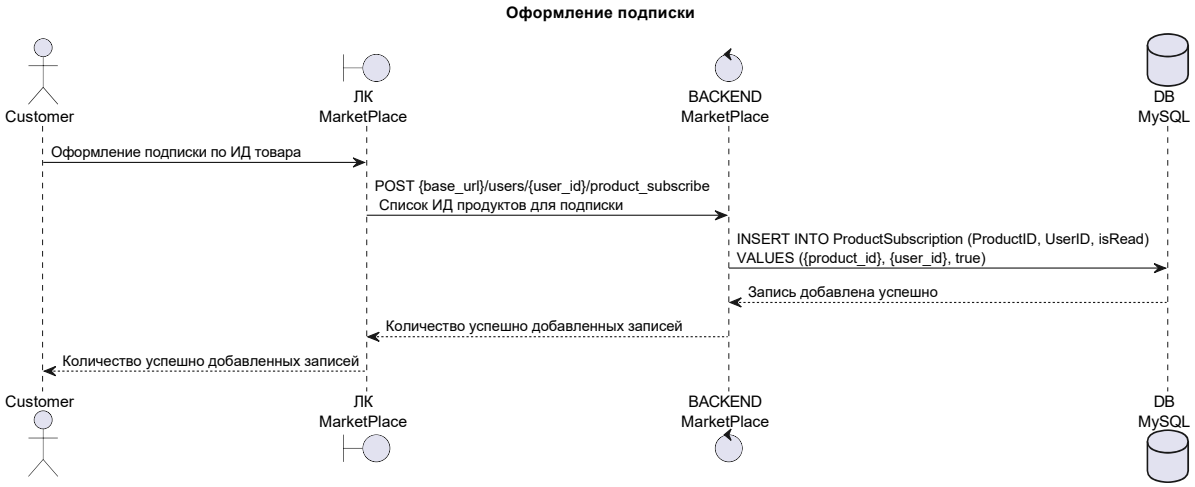
Выходные параметры

addProductSubscribeForUser

Входные параметры

Поле	Тип	Обязательно	Описание
user_id	int	+	ИД пользователя
products	int[]	+	Массив ИД товаров для добавления в список отслеживаемых пользователем.

Алгоритм

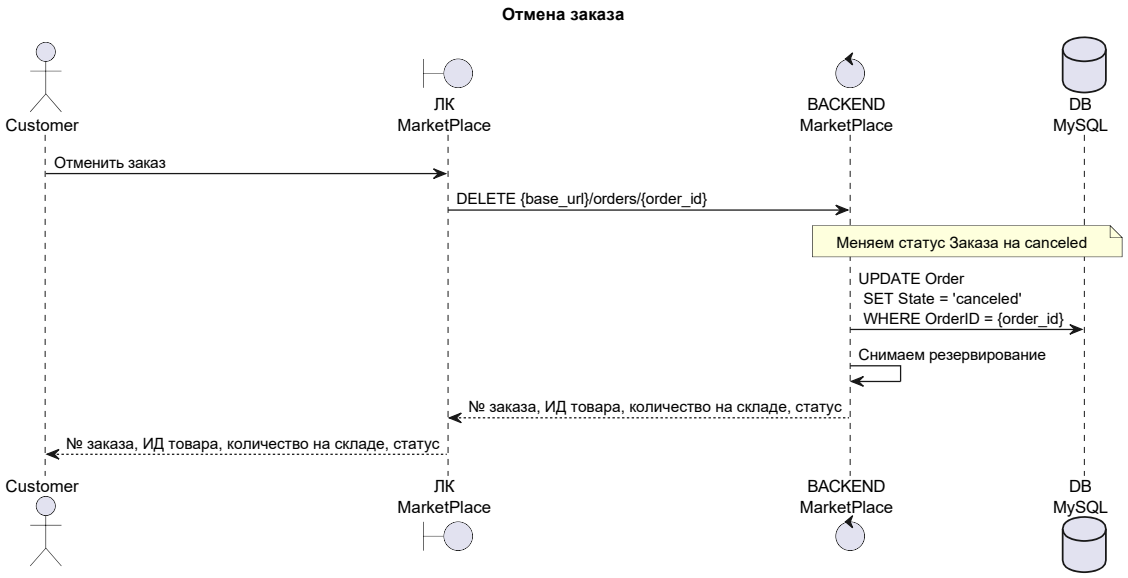


1. При получении запроса система проверяет корректность заполнения входных параметров (обязательность и тип). Если входные параметры не верны возвращается **ошибка 400**.
2. Если входные параметры верны, система для каждого элемента массива `products` входных параметров добавляет запись в таблицу ProductSubscription (Подписки пользователей на приход товара на склад), где поле `UserID` равно входному параметру `user_id`, а поле `ProductID` - очередному значению из массива `products`. SQL-запрос : `INSERT INTO ProductSubscription (ProductID, UserID) VALUES ({product_id}, {user_id})`. Ошибки добавления записей, связанные с ограничением `UNIQUE (ProductID, UserID)` игнорируются. Количество успешно добавленных записей считается и возвращается в виде выходного параметра `added_row_count`.

Выходные параметры

Поле	Тип	Обязательно	Описание
added_row_count	int	+	Количество успешно добавленных записей

deleteOrderById

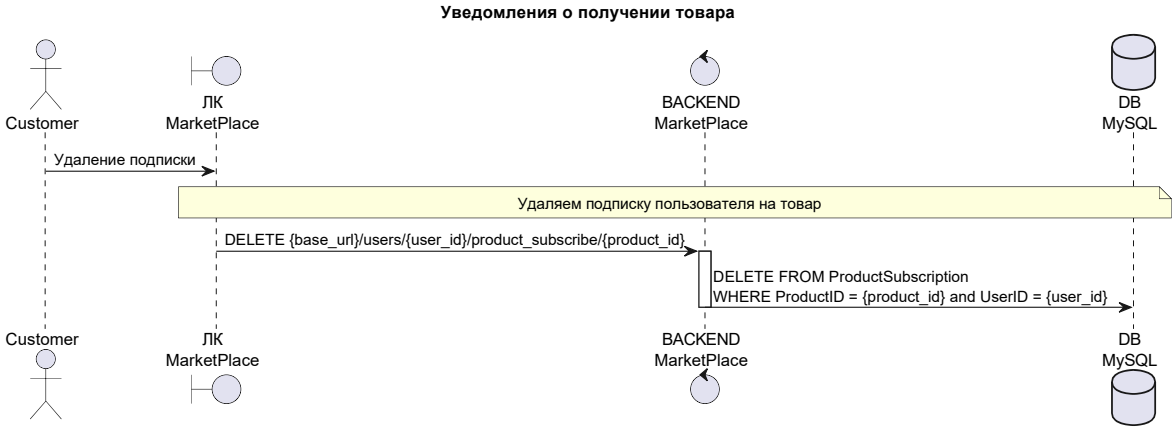


Входные параметры

Алгоритм

Выходные параметры

deleteUserSubscribeByProductId



Входные параметры

Алгоритм

Выходные параметры