**Сайт для заказа верхней одежды “Tostyle”**

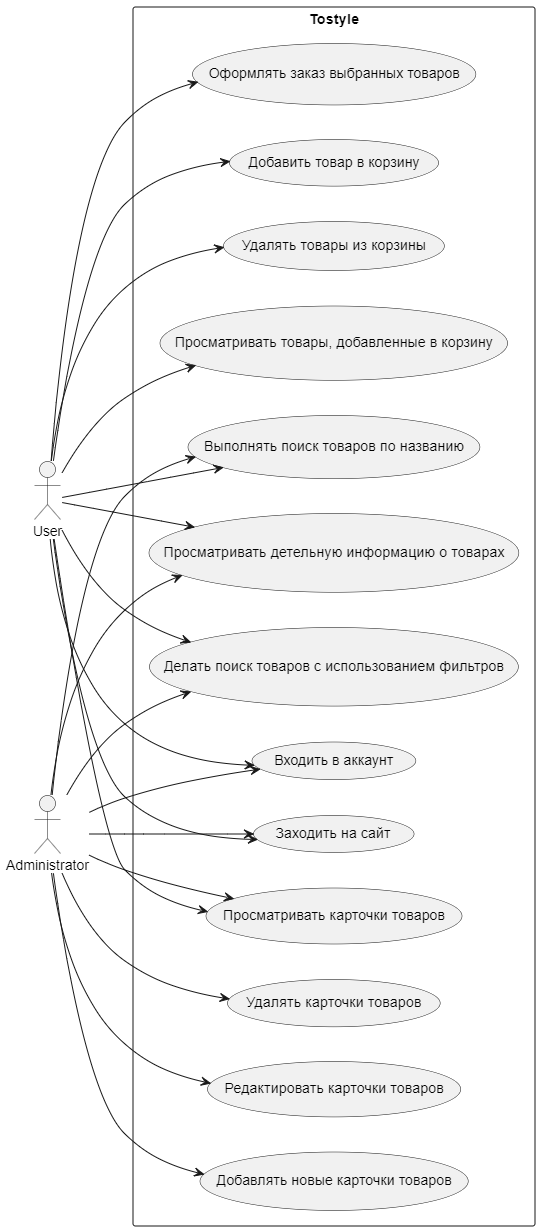
**Userstory для пользователя:**

1. Я как пользователь хочу заходить на сайт через Google Chrome, чтобы иметь доступ к функционалу сайта.
2. Я как пользователь хочу просматривать карточки товаров, чтобы ознакомиться с ассортиментом.
3. Я как пользователь хочу выполнять поиск по названию товаров, чтобы быстро найти нужный товар.
4. Я как пользователь хочу просматривать детальную информацию о товарах, чтобы узнать больше о продукте перед покупкой.
5. Я как пользователь хочу делать поиск товаров с использованием фильтров (размер, бренд, цвет, цена), чтобы найти товары, соответствующие моим предпочтениям.
6. Я как пользователь хочу добавлять товар в корзину, чтобы сохранить выбранные товары для покупки.
7. Я как пользователь хочу просматривать товары, добавленные в корзину, чтобы проверить выбранные товары перед оформлением заказа.
8. Я как пользователь хочу удалять товары из корзины, чтобы корректировать список покупок.
9. Я как пользователь хочу оформлять заказ выбранных товаров, чтобы завершить процесс покупки.
10. Я как пользователь хочу входить в аккаунт, чтобы получить доступ к персональным данным и истории заказов.

**Userstory для администратора:**

1. Я как администратор хочу заходить на сайт, чтобы управлять контентом и данными.
2. Я как администратор хочу просматривать карточки товаров, чтобы контролировать ассортимент.
3. Я как администратор хочу выполнять поиск по названию товаров, чтобы быстро находить нужные товары для управления.
4. Я как администратор хочу просматривать детальную информацию о товарах, чтобы проверять актуальность данных.
5. Я как администратор хочу делать поиск товаров с использованием фильтров (размер, бренд, цвет, цена), чтобы анализировать ассортимент.
6. Я как администратор хочу добавлять новые карточки товаров, чтобы расширять ассортимент.
7. Я как администратор хочу удалять карточки товаров, чтобы убирать устаревшие или недоступные товары.
8. Я как администратор хочу редактировать карточки товаров, чтобы обновлять информацию о товарах.
9. Я как администратор хочу входить в аккаунт, чтобы получить доступ к административным функциям.

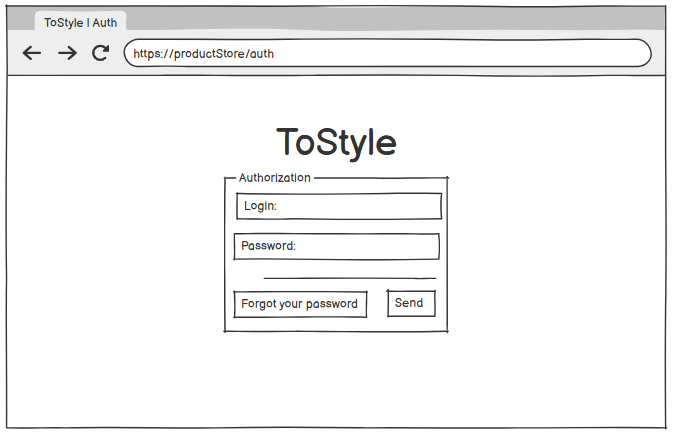
**Диаграмма usecases:**

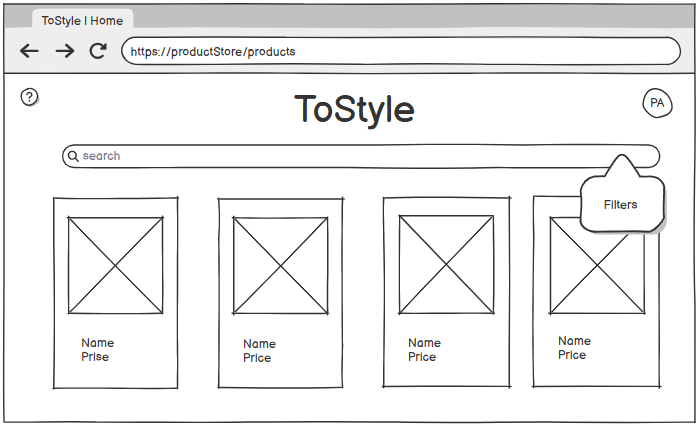


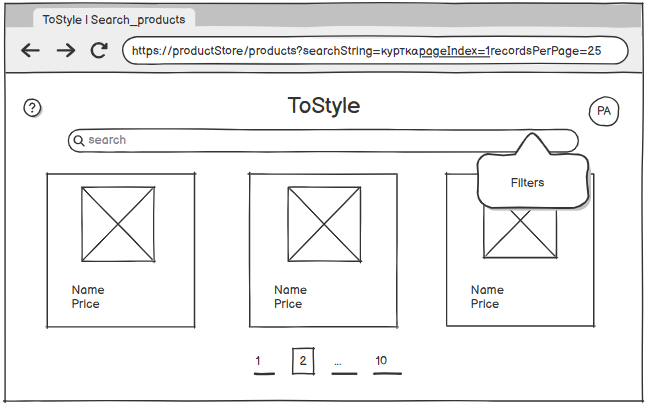
**Функциональные требования:**

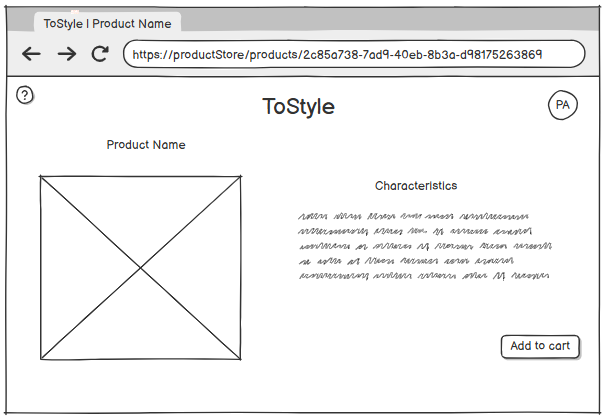
1. Система должна позволять пользователю заходить на сайт.
2. Система должна позволять пользователю входить в аккаунт.
3. Система должна предоставлять пользователю возможность открывать карточки товаров.
4. Система должна позволять пользователю просматривать детальную информацию о товарах.
5. Система должна позволять пользователю выполнять поиск товаров по названию и фильтрам.
6. Система должна позволять пользователю просматривать все карточки товаров, которые были добавлены в Корзину.
7. Система должна позволять пользователю добавлять товар в Корзину.
8. Система должна позволять пользователю удалять товары из Корзины.
9. Система должна позволять пользователю со страницы Корзина возвращаться на страницу Home.
10. Система должна позволять администратору заходить на сайт.
11. Система должна предоставлять администратору возможность открывать карточки товаров.
12. Система должна позволять администратору выполнять поиск товаров по названию и фильтрам.
13. Система должна позволять администратору просматривать детальную информацию о товарах.
14. Система должна позволять администратору просматривать все карточки товаров, которые были добавлены в Корзину.
15. Система должна позволять администратору добавлять новые карточки товаров.
16. Система должна позволять администратору удалять карточки товаров.
17. Система должна позволять администратору редактировать карточки товаров.
18. Система должна позволять администратору со страницы Корзина возвращаться на страницу Home.
19. Система должна позволять администратору входить в аккаунт.

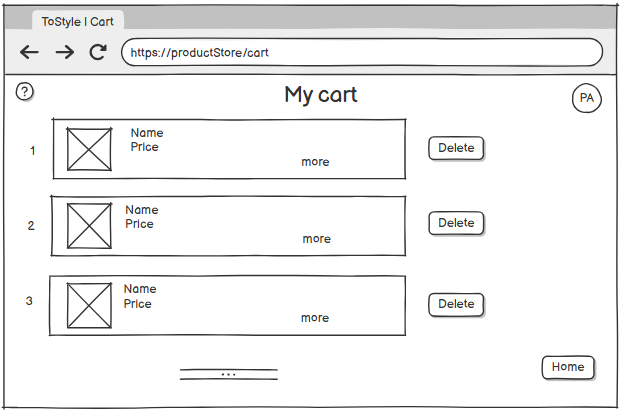
**Макеты пользовательского интерфейса:**











**Нефункциональные требования:**

1. Bход должен осуществляться по логину и паролю.
2. На одной странице поиска товаров должно находится не более 25 товаров.
3. Внизу каждой страницы поиска товаров должны находиться номера страниц и их количество.
4. Нагрузочные характеристики: в системе одновременно должны работать до 50 пользователей.

**Usecases**

**Usecase 1**

**Цель:** авторизация на сайте.

**Предусловие:** неавторизованный пользователь зашел на сайт.

**Триггер:** неавторизованный пользователь нажал на кнопку “Login”.

**Участники:**

1. Неавторизованный пользователь
2. Frontend
3. Backend
4. Database.

1. Неавторизованный пользователь заходит на сайт.

2. Неавторизованный пользователь вводит Login и Password и нажимает кнопку Send.

3. Frontend отправляет запрос Backend, передавая учетные данные.

4. Backend отправляет запрос в Database с целью поиска пользователя.

5. Database возвращает ответ Backend.

6. Пользователь найден.

7. Backend отправляет запрос в Database с целью проверки пароля.

8. Пароль верный.

9. Backend выпускает JWT Token.

10. Backend передаёт Frontend JWT Token в соответствии с ролью пользователя.

11. Frontend переадресует пользователя на главную страницу.

**Результат**: Пользователь успешно прошел аутентификацию и получил токен, содержащий его роль и позволяющий ему совершать дальнейшие запросы.

**Расширение сценария: Неправильный логин**

6.1. Пользователь не найден.

6.2. Backend передает ответ Frontend.

6.3. Frontend отражает пользователю следующий текст: Неверный логин.

**Завершение сценария. Пользователь не прошел идентификацию.**

**Расширения сценария: Неверный пароль**

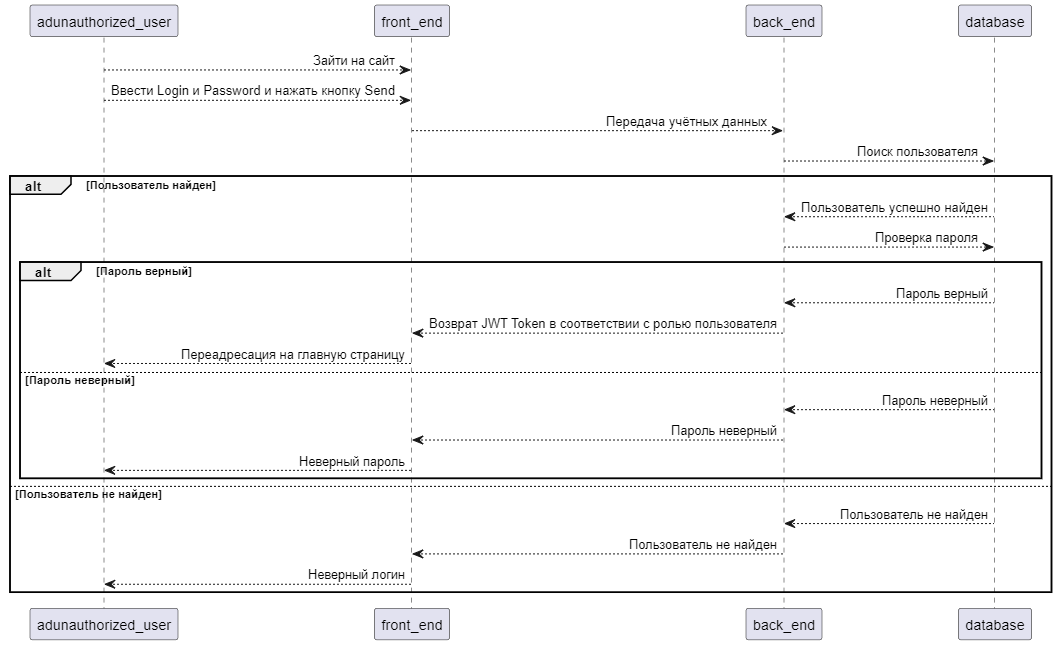
8.1. Пароль неверный.

8.2 Backend передает ответ Frontend.

8.3 Frontend отражает пользователю следующий текст: Неверный пароль.

**Завершение сценария. Пользователь не прошел аутентификацию.**

**Диаграмма последовательности:**

****

**Usecase 2**

**Цель:** выполнить поиск товаров.

**Предусловие:** пользователь зашёл на главную страницу сайта.

**Триггер:** пользователь нажал на строку “Search”.

**Участники:**

1. Неавторизованный пользователь/пользователь/администратор
2. Frontend
3. Backend
4. Database.

1.Пользователь зашел на главную страницу сайта.

2. Пользователь нажал на строку “Search”.

3. Пользователь ввёл название товара и нажал Enter.

4. Frontend отправляет запрос:

**GET https://productStore/api/v1/products?searchString={user\_input}&pageIndex=0&recordsPerPage=25**,

в Backend передавая название товара в query параметре searchString. PageIndex и recordsPerPage необходимы для пагинации.

5.Backend отправляет запрос в Database в таблицу Products в поле Name с целью поиска товаров подходящих под запрос пользователя.

6. Database возвращает ответ в Backend.

7. Товары найдены.

8. Backend передает информацию о товарах Frontend (см. пример объекта ниже).

9. Frontend отрисовывает пользователю карточки найденных товаров.

**Результат:** Пользователь успешно выполнил поиск товаров по названию и получил карточки товаров.

**Расширение сценария: Поиск товара по фильтрам**

3.1. Пользователь нажал на всплывающее окно “Filters”.

3.2. Пользователь применил один или более фильтров и нажал Enter.

3.3. Frontend отправляет запрос:

**GET https://productStore/api/v1/products?searchString={user\_input}&pageIndex=0&recordsPerPage=25**

в Backend передавая указанную в фильтрах информацию о товаре в query параметрах size, brand, color, price.

3.4. Backend отправляет запрос на поиск в Database в таблицу Products в соответствующие фильтрам поля (Name, size, brand, color, price).

3.5. Database возвращает ответ в Backend.

3.6. Товары найдены.

3.7. Backend передает информацию о товарах Frontend (см. пример объекта ниже).

3.8. Frontend отрисовывает пользователю карточки найденных товаров.

**Завершение сценария:** Пользователь успешно выполнил поиск товаров с использованием фильтров и получил карточки товаров.

**Расширение сценария: Товары не найдены.**

7.1. Товары не найдены.

7.2. Backend передает ошибку 404 во Frontend.

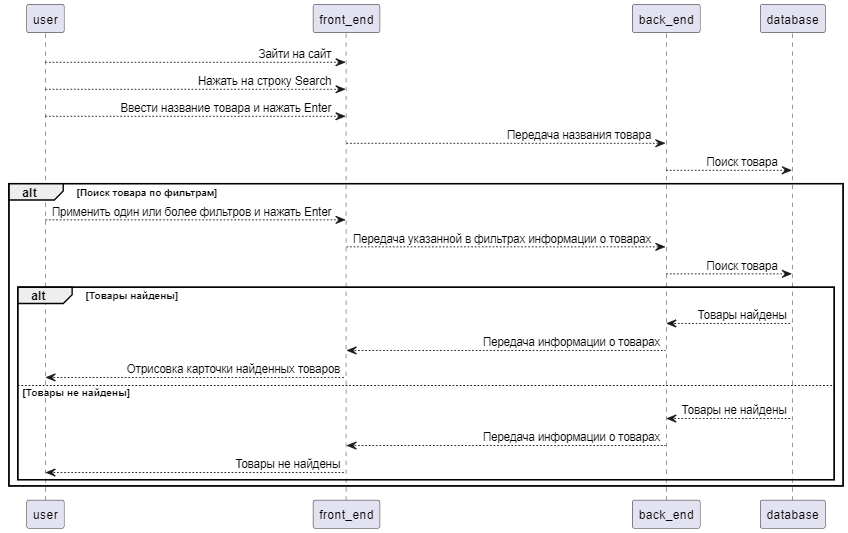
7.3. Frontend отражает пользователю следующий текст: Товары не найдены.

**Завершение сценария:** Товары, запрашиваемые пользователем, не найдены.

**Пример объекта ответа:**

****

**Диаграмма последовательности:**



**Usecase 3**

**Цель:** добавить новый товар.

**Предусловие:** пользователь прошел авторизацию и получил роль “Администратор”.

**Триггер:** администратор нажал на кнопку “Добавить новый товар”.

**Участники:**

1. Администратор
2. Frontend
3. Backend
4. Database.

1.Администратор нажал на кнопку “Добавить новый товар”.

2. Администратор указал все необходимые характеристики товара и нажал на кнопку “Создать”.

3.Frontend отправляет запрос:

**POST https://productStore/api/v1/product?query**

в Backend передавая характеристики товара в json-объекте (см. пример объекта ниже).

4.Backend отправляет запрос на создание товара в Database в таблицу Products в соответствующие фильтрам поля (Name, size, brand, color, price, id, product, description, picture\_id).

5.Database сохраняет полученные данные о товаре и возвращает ответ в Backend.

6.Backend передает информацию во Frontend: 201 Created.

7.Frontend отрисовывает администратору новую карточку созданного товара.

**Результат:** Администратор успешно создал новую карточку товара в магазине.

**Расширение сценария: Администратор заполнил не все необходимые поля.**

3.1.Frontend возвращает ответ пользователю, содержащий следующий текст: Нужно заполнить все указанные поля.

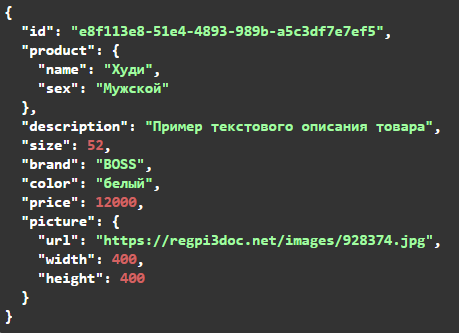
**Результат:** Новая карточка товара не создана.

**Расширение сценария: Администратор указал некорректные символы.**

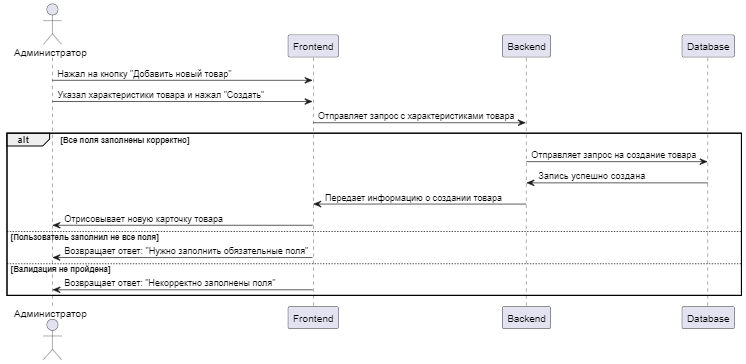
3.1. Frontend возвращает ответ пользователю, содержащий следующий текст: Некорректно заполнены поля.

**Результат:** Новая карточка товара не создана.

**Пример объекта запроса:**



**Диаграмма последовательности:**



**Usecase 4**

**Цель:** обновить существующий товар по id.

**Предусловие:** пользователь прошел авторизацию и получил роль “Администратор”.

**Триггер:** администратор нажал на кнопку “Редактировать товар”.

**Участники:**

1. Администратор
2. Frontend
3. Backend
4. Database.

1.Администратор нажал на кнопку “Редактировать товар”.

2. Администратор внес в характеристики товара необходимые правки и нажал отправить.

3. Frontend отправляет запрос:

**PUT https://productStore/api/v1/product?query**

в Backend передавая новые характеристики товара (см. пример json-объекта ниже).

4. Backend отправляет запрос с целью обновления указанного товара в Database в таблицу Products в соответствующие фильтрам поля (Name, size, brand, color, price, product, description, picture\_id)..

5. Database сохраняет полученные данные о товаре и возвращает ответ в Backend.

6. Backend передает информацию во Frontend: 200 ОК.

7.Frontend отрисовывает администратору обновленную карточку товара и возвращает ответ, содержащий следующий текст: Товар обновлен.

**Результат:** Администратор успешно обновил товар.

**Расширение сценария: Создание нового товара.**

4.1. Вackend проверяет id товара. id не найден.

4.2. Вackend создает новую запись с новым id в таблице Products и отправляет запрос на создание товара в Database.

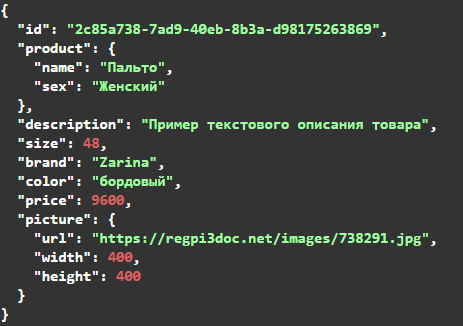
4.3. Database сохраняет полученные данные о товаре и возвращает ответ в Backend.

4.4.Backend передает информацию во Frontend: 201 Created.

4.5. Frontend возвращает администратору ответ, содержащий следующий текст: Создан новый товар.

**Результат:** Создание нового товар по id.

**Пример объекта запроса:**

****

**Usecase 5**

**Цель:** удалить товар по id.

**Предусловие:** пользователь прошел авторизацию и получил роль “Администратор”.

**Триггер:** администратор нажал на кнопку “Удалить товар”.

**Участники:**

1. Администратор
2. Frontend
3. Backend
4. Database.

1.Администратор нажал на кнопку “Удалить товар”.

2. Frontend отправляет запрос:

**DELETE https://productStore/api/v1/product/{id}**

в Backend передавая id товара.

3. Backend отправляет запрос с целью удаления указанного товара в Database в таблицу Products в поле id.

4. Database возвращает ответ в Backend.

5. Backend передает информацию во Frontend: 200 ОК.

6. Frontend возвращает администратору ответ, содержащий следующий текст: Товар удален.

**Результат:** Администратор успешно удалил товар.

**Usecase 6**

**Цель:** обновить цену товара.

**Предусловие:** пользователь прошел авторизацию и получил роль “Администратор”.

**Триггер:** администратор нажал на кнопку “Редактировать карточку товара”.

**Участники:**

1. Администратор
2. Frontend
3. Backend
4. Database.

1.Администратор нажал на кнопку “Редактировать карточку товара”.

2. Администратор обновил цену товара и нажал отправить.

3. Frontend отправляет запрос:

**PATCH https://productStore/api/v1/product/{id}**

в Backend передавая новую цену товара в json-объекте.

4. Backend отправляет запрос в Database в таблицу Products в поле Price.

5. Database возвращает ответ в Backend.

6. Backend передает информацию во Frontend: 200 ОК (см. пример json-объекта ниже).

7. Frontend возвращает администратору ответ, содержащий следующий текст: Цена товара обновлена.

**Результат:** Администратор успешно обновил цену товара.

**Расширение сценария: Товар не найден.**

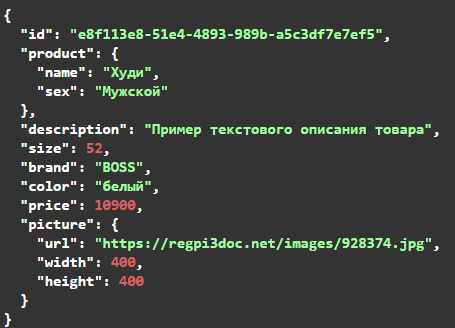
4.1. Вackend проверяет id товара. id не найден.

4.2. Backend передает информацию во Frontend.

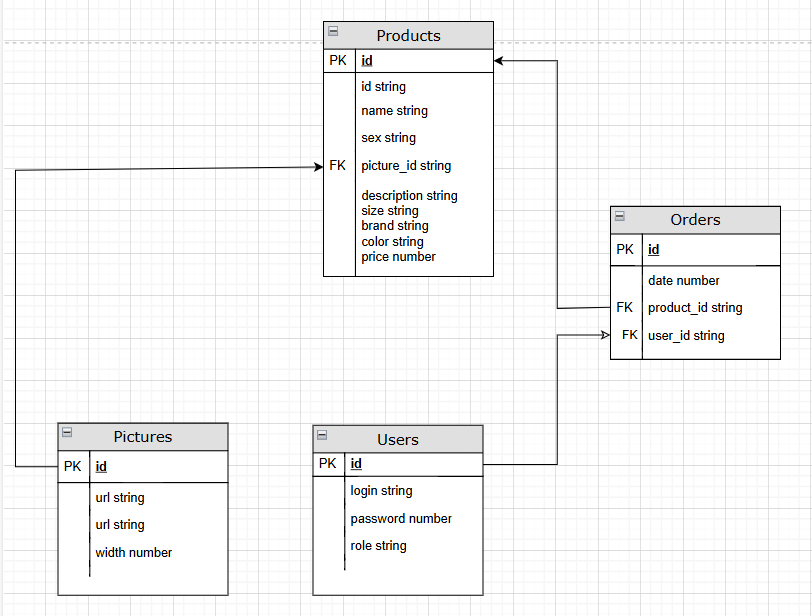
4.3. Frontend возвращает администратору ответ, содержащий следующий текст: Данного товара не существует.

**Результат:** Цена товара не обновлена.

**Пример объекта ответа:**



**Модель данных:**



**OpenAPI specification доступна по ссылке:** <https://drive.google.com/file/d/16_HOrT1_hcOwV3u4HJYFBOO4HFGh2Cmy/view?usp=sharing>

**Эту спецификацию, json и диаграммы можно посмотреть на гугл диске:** <https://drive.google.com/drive/folders/16gdAj0nsWu4kM7zffg-BTDAg4ufBF4nn?usp=sharing>