База данных и созданное API находятся на GitHub: https://github.com/K4ch0k/greenatom

После восстановления БД из файла «Greenatom.bak» необходимо обеспечить доступ API к ней. Для этого в файле «Web.config», находящегося в корневом каталоге проекта, нужно изменить 69 строчку кода (поменять название сервера и при необходимости добавить данные для авторизации).

Проверка работоспособности API выполнялась в программе Postman. Если API принимает в себя объект, то на вкладке «Headers» добавлялся «Key» = «Content-Type» с «Value» = «application/json».

Для каждого API имеется документация. Попасть на нее можно через браузер после запуска отладки API в открывшейся странице перейти на вкладку «API», далее выбрать интересующее API.

База данных.

Выполнение задания началось с разработки БД в СУБД Microsoft SQL Server. На рисунке 1 представлена диаграмма итогового варианта БД. Сущности «Orders» - Заказы; «ProductsInOrder» - Предметы, входящие в заказ; «Products» - Предметы являются обязательными. Сущности «Status» - Статус оформления продукта в заказе и «Users» - Пользователи, осуществляющие заказы являются необязательными для выполнения задания, но я решил их добавить, посчитав необходимым для «Бизнес-логики».

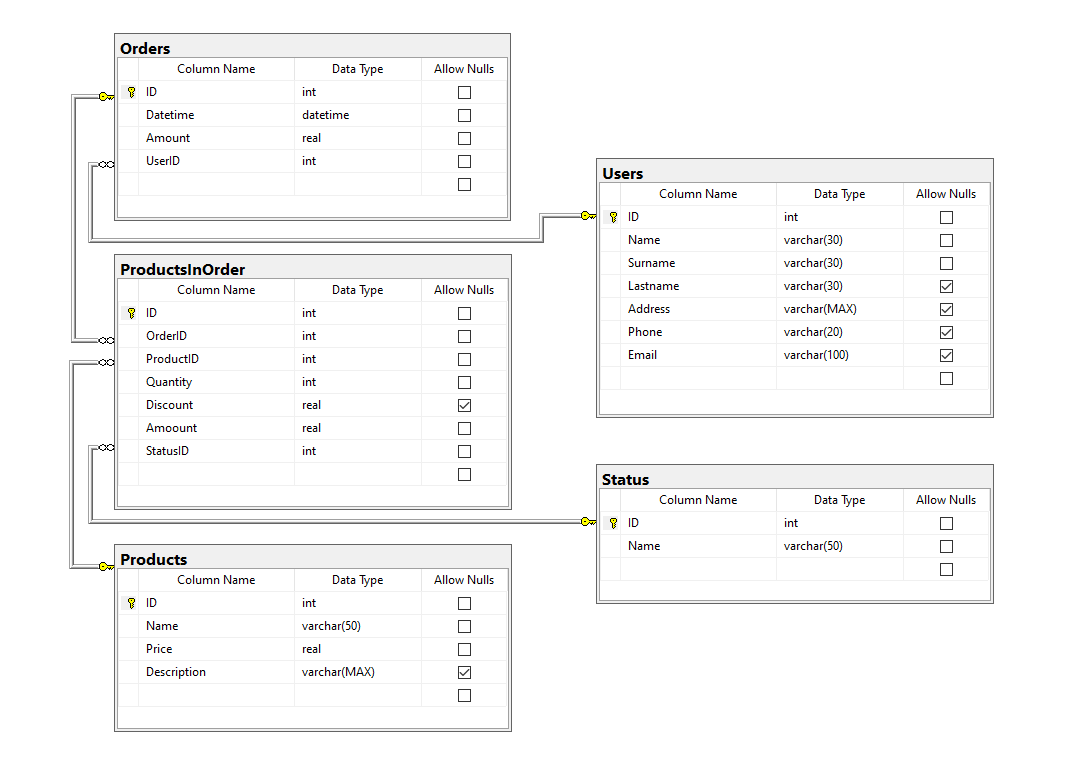


Рисунок 1. Диаграмма созданной БД

После создания базы данных начался этап разработки API, который разбит на несколько «шагов» для выполнения поставленной задачи.

1. Редактирование API, созданной по умолчанию;
2. Реализация контроллера API для таблицы «Products»;
3. Реализация контроллера API для таблицы «Orders»;
4. Реализация контроллера API для таблицы «ProductsInOrder».

Редактирование API, созданного по умолчанию.

Сразу была добавлена технология работы с данными ADO.NET Entity Framework(Рисунок 2) для удобного взаимодействия с данными из БД.

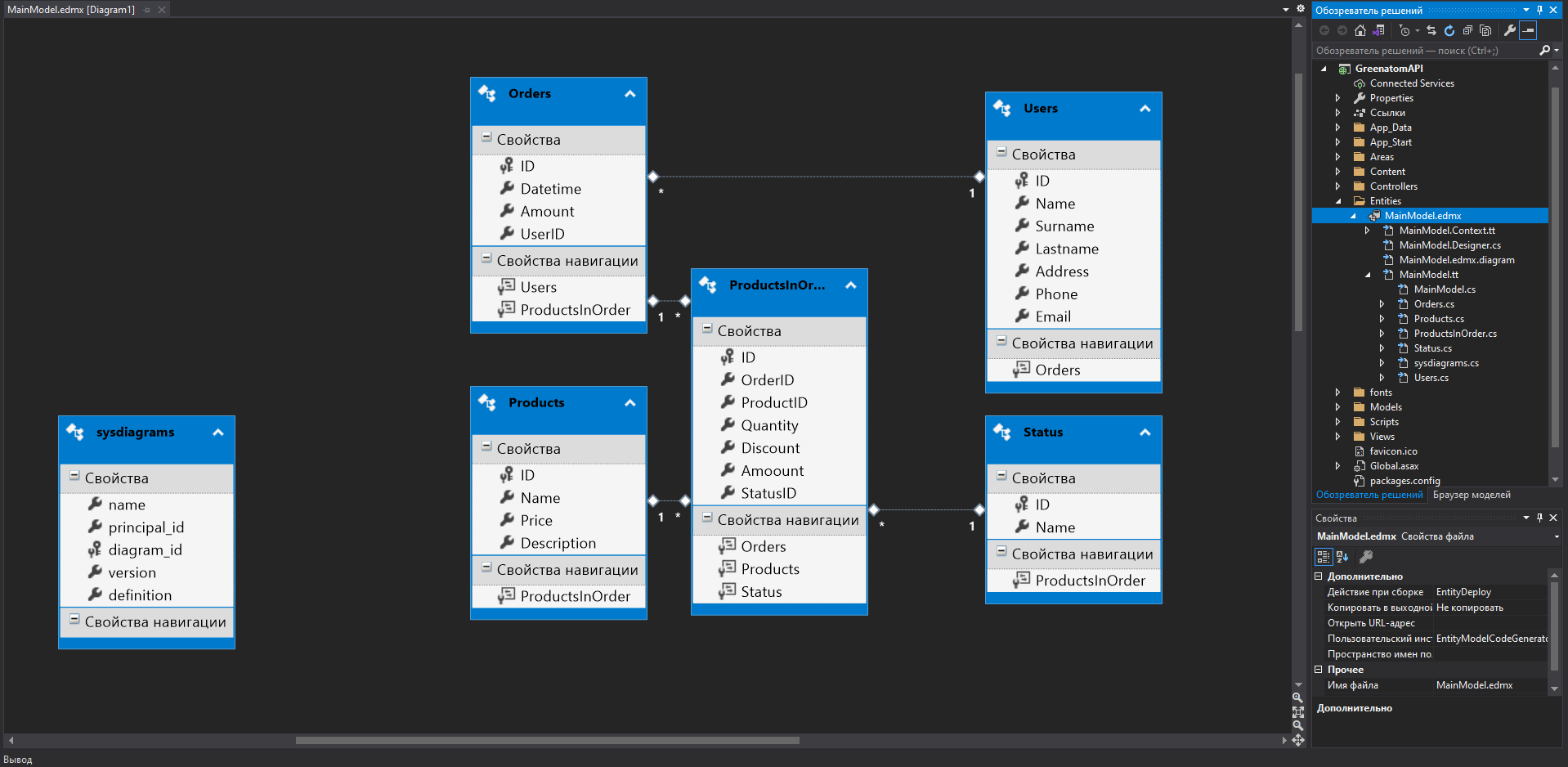


Рисунок 2. Созданная модель данных на основе БД

В файле «WebApiConfig.cs», который находится в каталоге App\_Start были добавлены следующие строчки кода для настройки возвращаемого формата json:

config.Formatters.Clear();

config.Formatters.Add(new JsonMediaTypeFormatter());

Итоговый вариант файла «WebApiConfig.cs» представлен на рисунке 3.

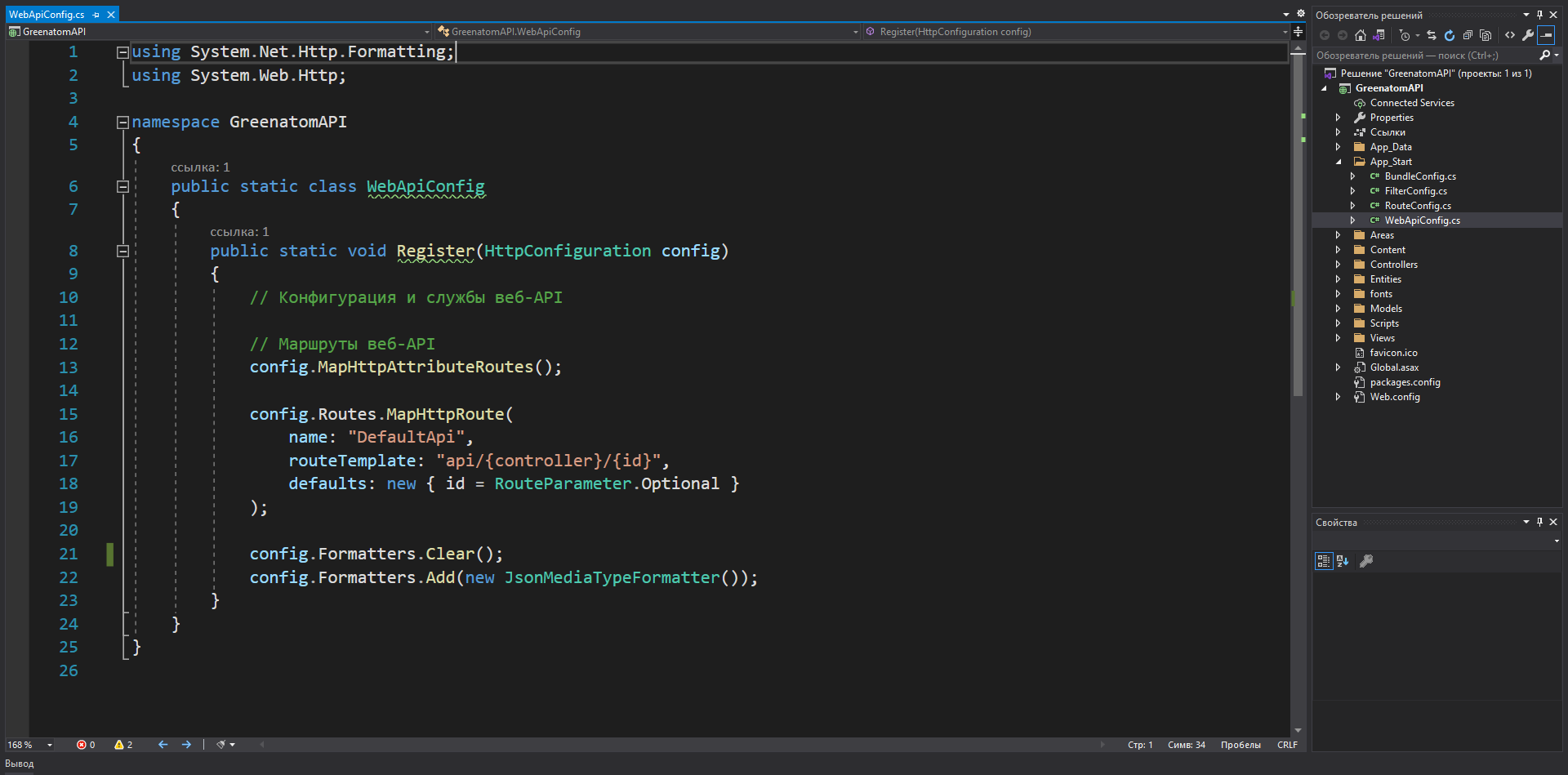


Рисунок 3. Измененный файл «WebApiConfig.cs»

Последнее изменение, выполненное в этом шаге произведено для автоматического создания описания документации на основе XML-комментариев. Для этого из каталога проекта нужно перейти по следующему пути «\Areas\HelpPage\App\_Start\HelpPageConfig.cs» и раскомментировать 37 строчку этого файла(Рисунок 4).

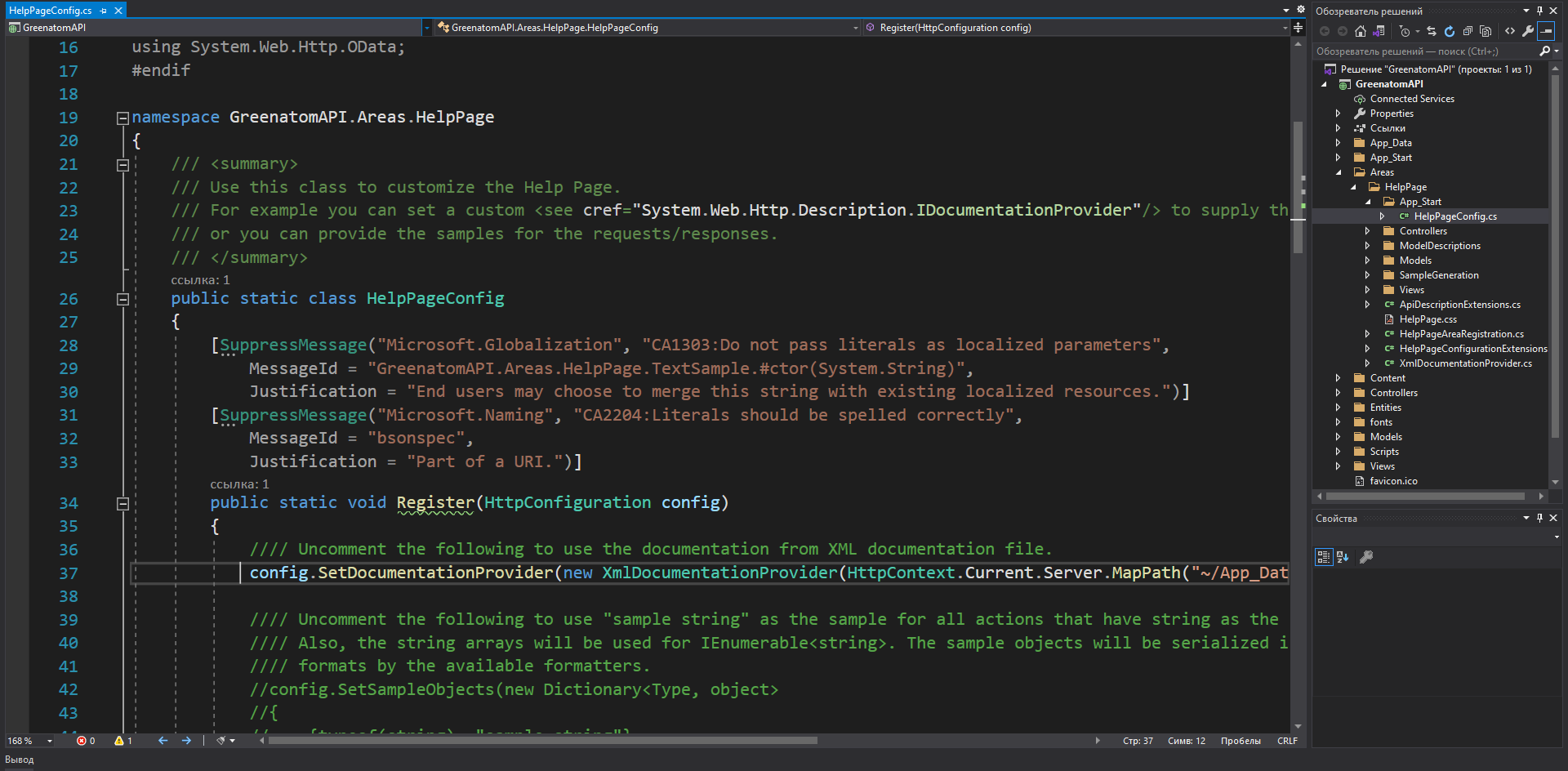


Рисунок 4. Код, позволяющий автоматически генерировать документацию на основе XML-комментариев

Реализация контроллера API для таблицы «Products».

Для реализации CRUD-операций необходимо создать 4 метода (Create – Создание; Read – Чтение; Update – Модификация; Delete - Удаление).

На чтение было создано 2 метода для просмотра всех записей и одной записи об объекте. На рисунке 5 изображен код для просмотра всех продуктов. Таблица «Products» имеет связь с таблицей «ProductsInOrder», поэтому при обращении к элементу таблицы «Products» будут показаны все связанные записи из таблицы «ProductsInOrder». Однако текущий метод не подразумевает показ всех связей, он должен возвращать информацию из таблицы «Products». Для этого был создан класс (Рисунок 6), конвертирующий информацию из БД в запись, которую уже можно корректно отобразить. Подобные модели были созданы и для других методов, взаимодействующих с сущностями БД.

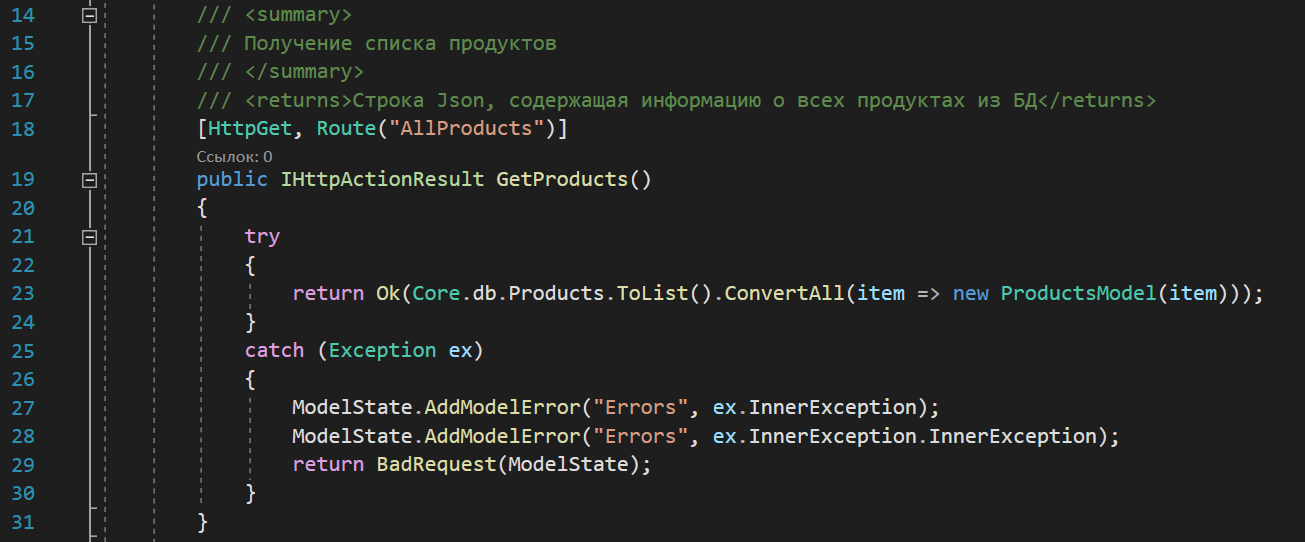


Рисунок 5. Метод, возвращающий Json строку из списка продуктов

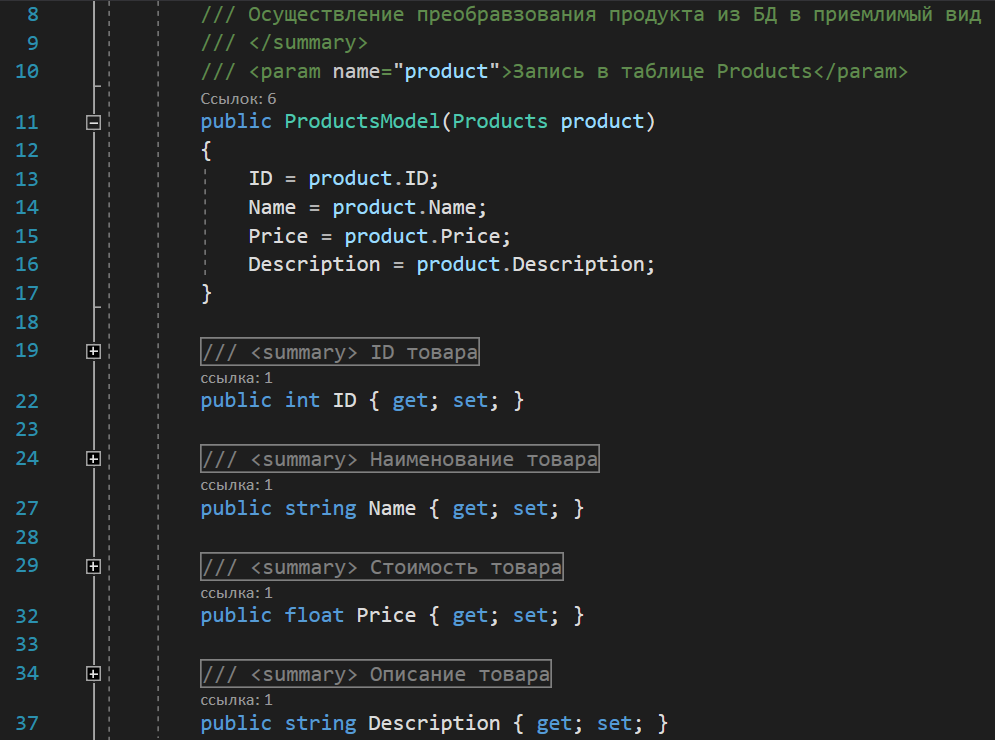


Рисунок 6. Модель для конвертации информации о продукте

Результат, полученный при обращении к этому API показан на рисунке 7.

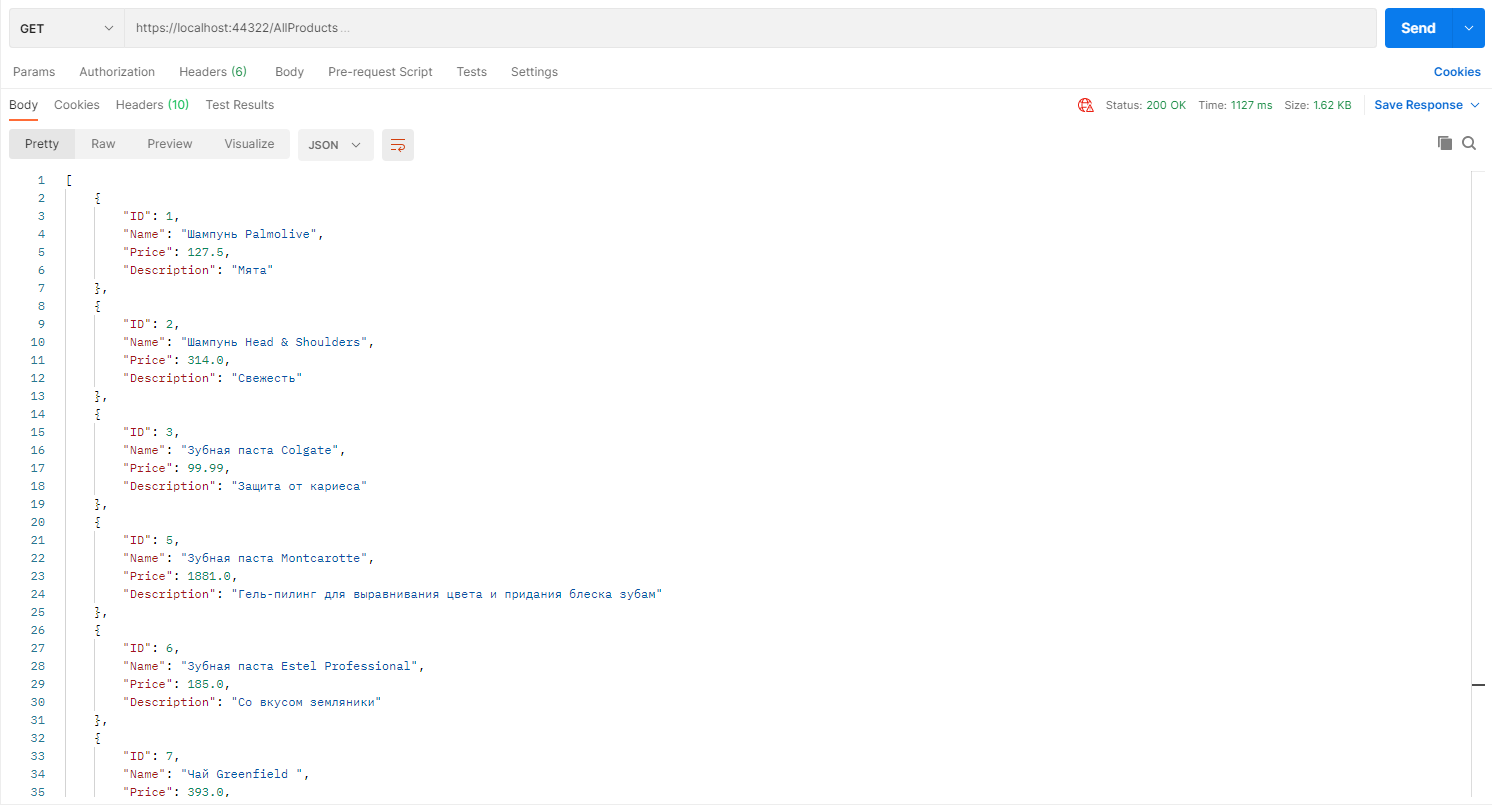


Рисунок 7. Ответ API AllProducts

Для получения информации об одном товаре, необходимо передать в API аргумент ID – целочисленное число, обозначающее идентификатор продукта из БД. Код данного метода продемонстрирован на рисунке 8.

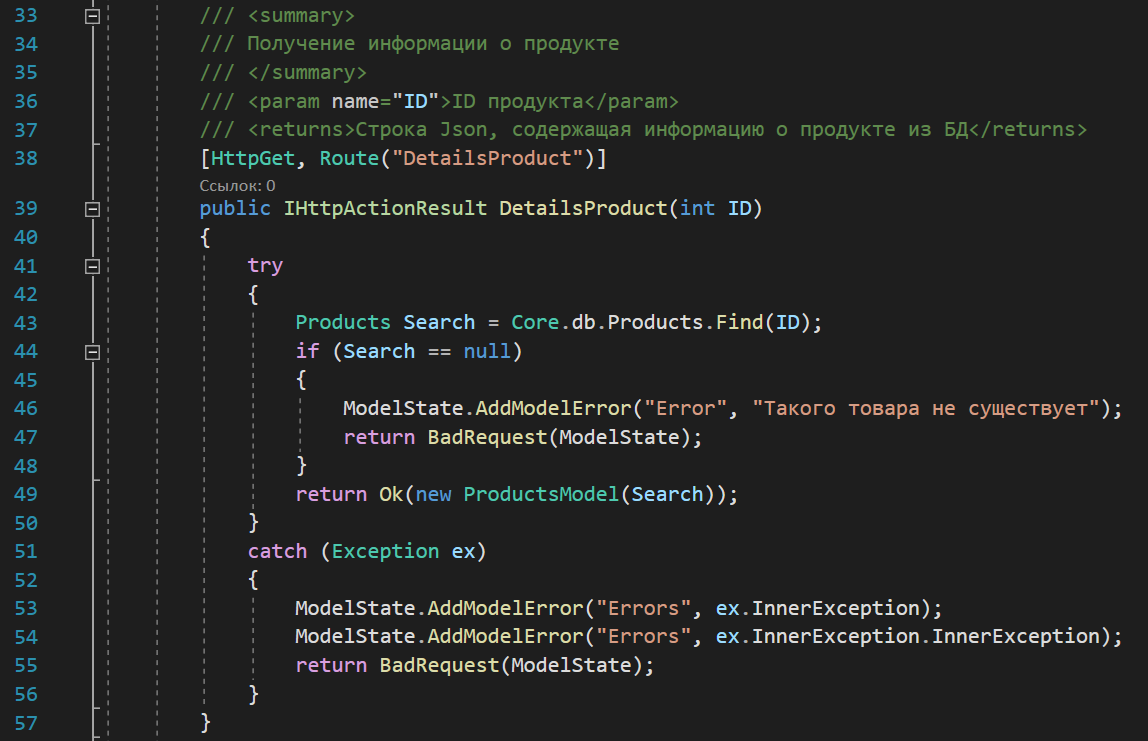


Рисунок 8. Метод, реализующий поиск продукта по заданному ID

Результат, полученный при обращении к этому API показан на рисунке 9.

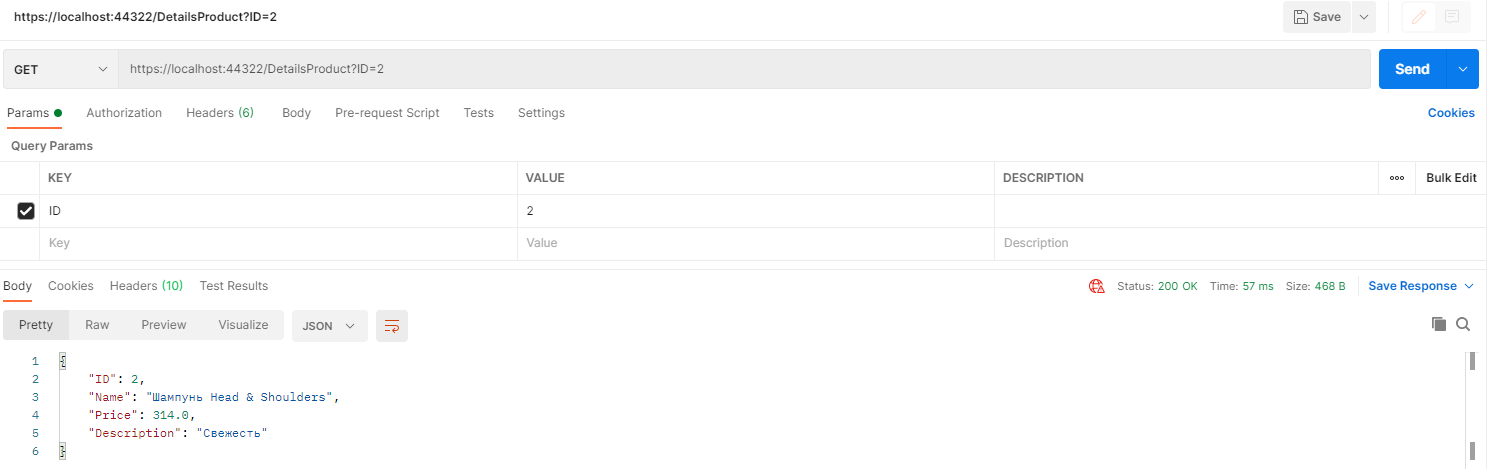


Рисунок 9. Ответ API DetailsProduct?ID={ID}

Следующий метод (Рисунки 10,11) осуществляет создание записи о продукте и добавляет экземпляр в БД, а обращающийся к API клиент увидит созданный объект (Рисунок 12). Данный код осуществляет проверку поля «Name» - наименование продукта на пустоту и поля «Price» - стоимость продукта на отрицательное значение. Если один из параметров не пройдет проверку, то добавление данных в БД не будет выполнено, а метод вернет ошибку (Рисунок 13).

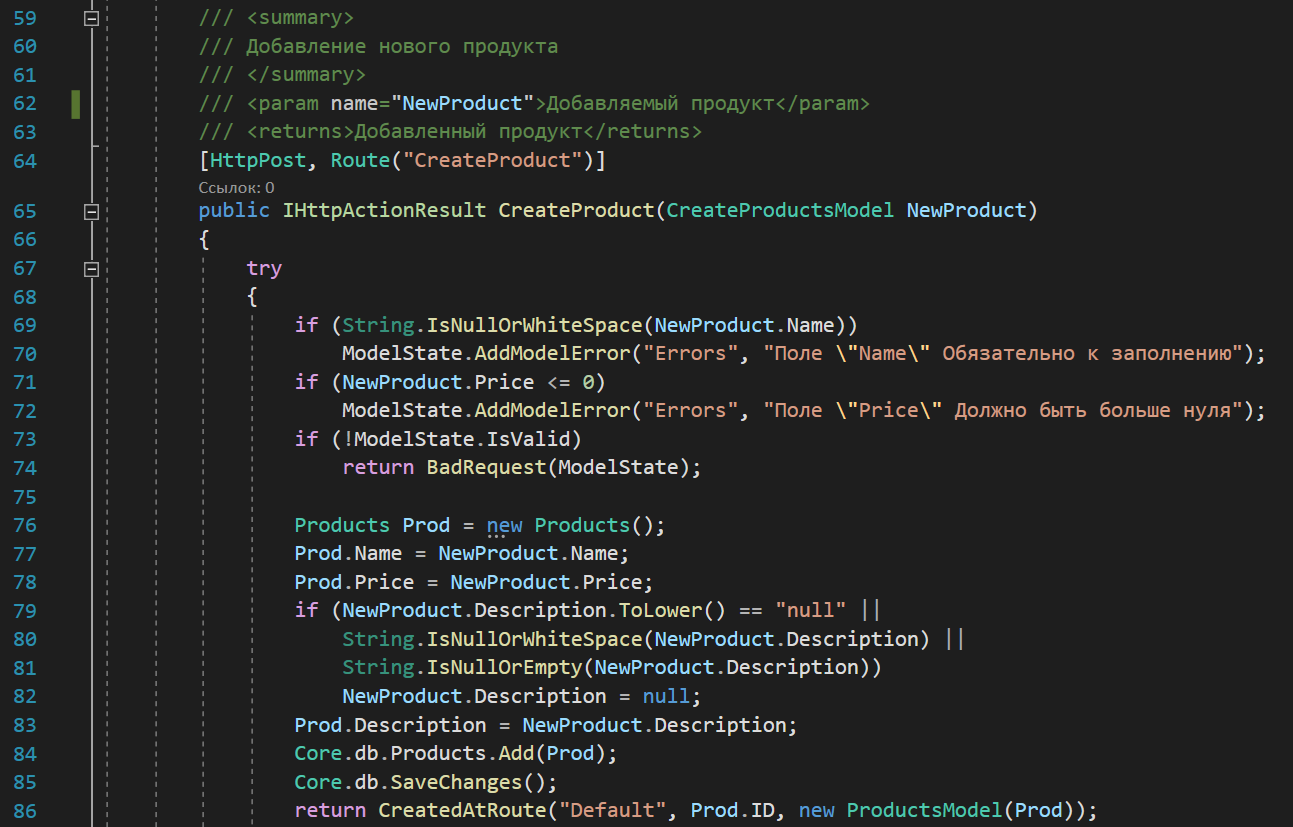


Рисунок 10. Метод создания нового продукта (часть 1)

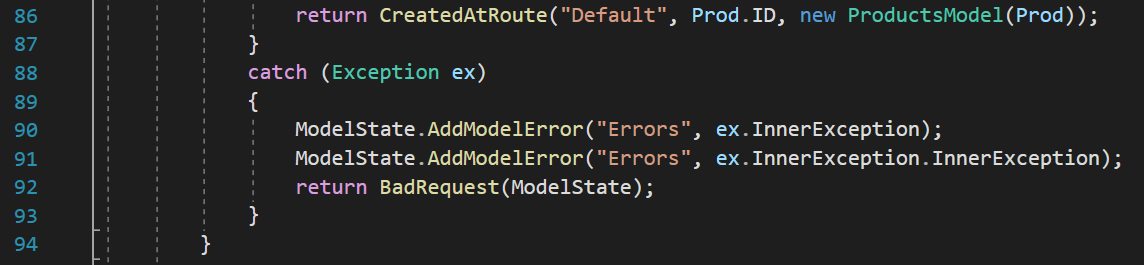


Рисунок 11. Метод создания нового продукта (часть 2)

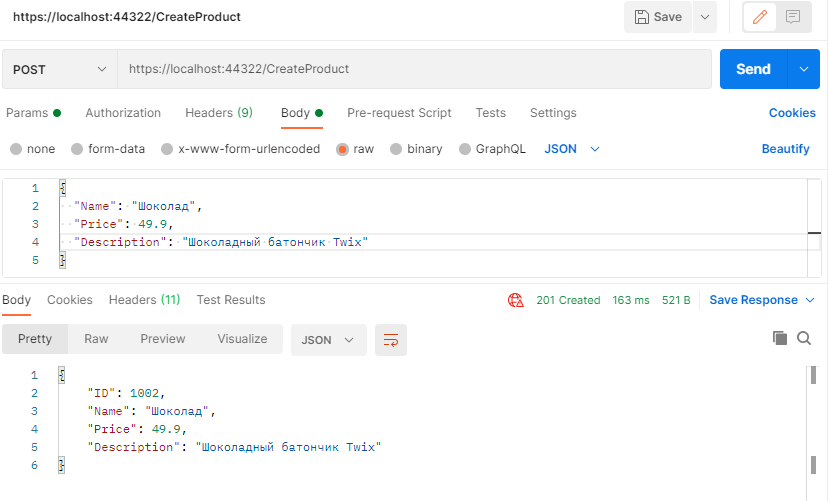


Рисунок 12. Ответ от API CreateProduct при успешном добавлении данных

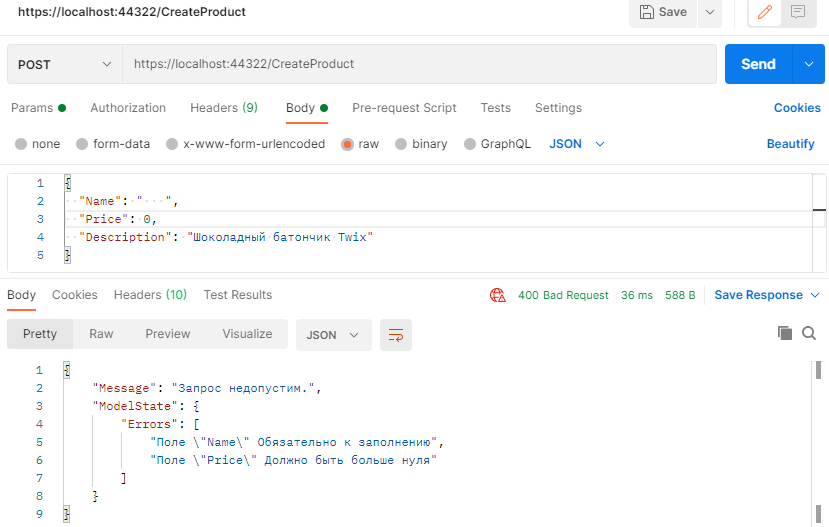


Рисунок 13. Ответ от API при отправке некорректных данных

Для возможности удаления записи из БД был создан метод DeleteProduct (Рисунок 14), принимающий аргумент ID – целочисленное число, обозначающее идентификатор продукта из БД. API возвращает удаленный экземпляр продукта (Рисунок 15).

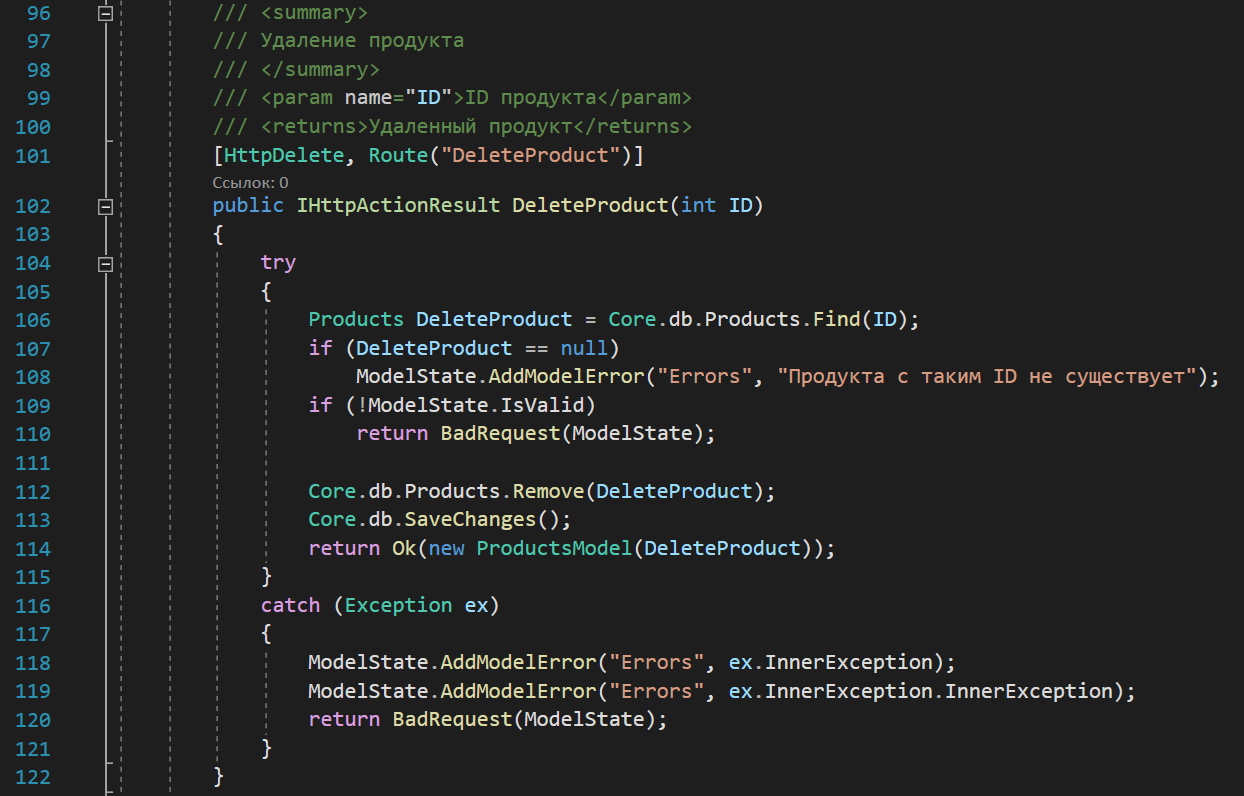


Рисунок 14. Метод для удаления записи о продукте из БД

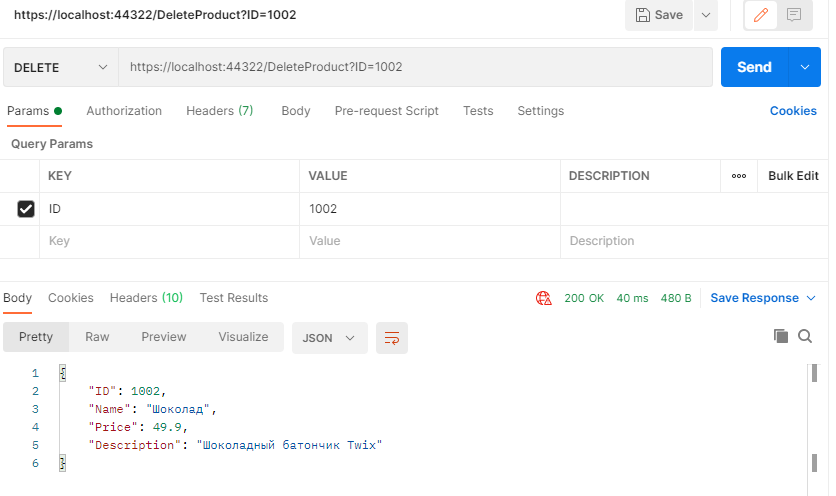


Рисунок 15. Ответ от API при удалении продукта из БД

Последний метод, позволяющий изменять существующий объект продукта – EditProduct (Рисунок 16) принимает параметр идентификатора продукта и измененный объект. Клиент, отправивший запрос на API, в случае успеха получит измененный экземпляр объекта, а в случае если одна из проверок добавит ошибку, то увидит массив ошибок, представленный в формате Json.

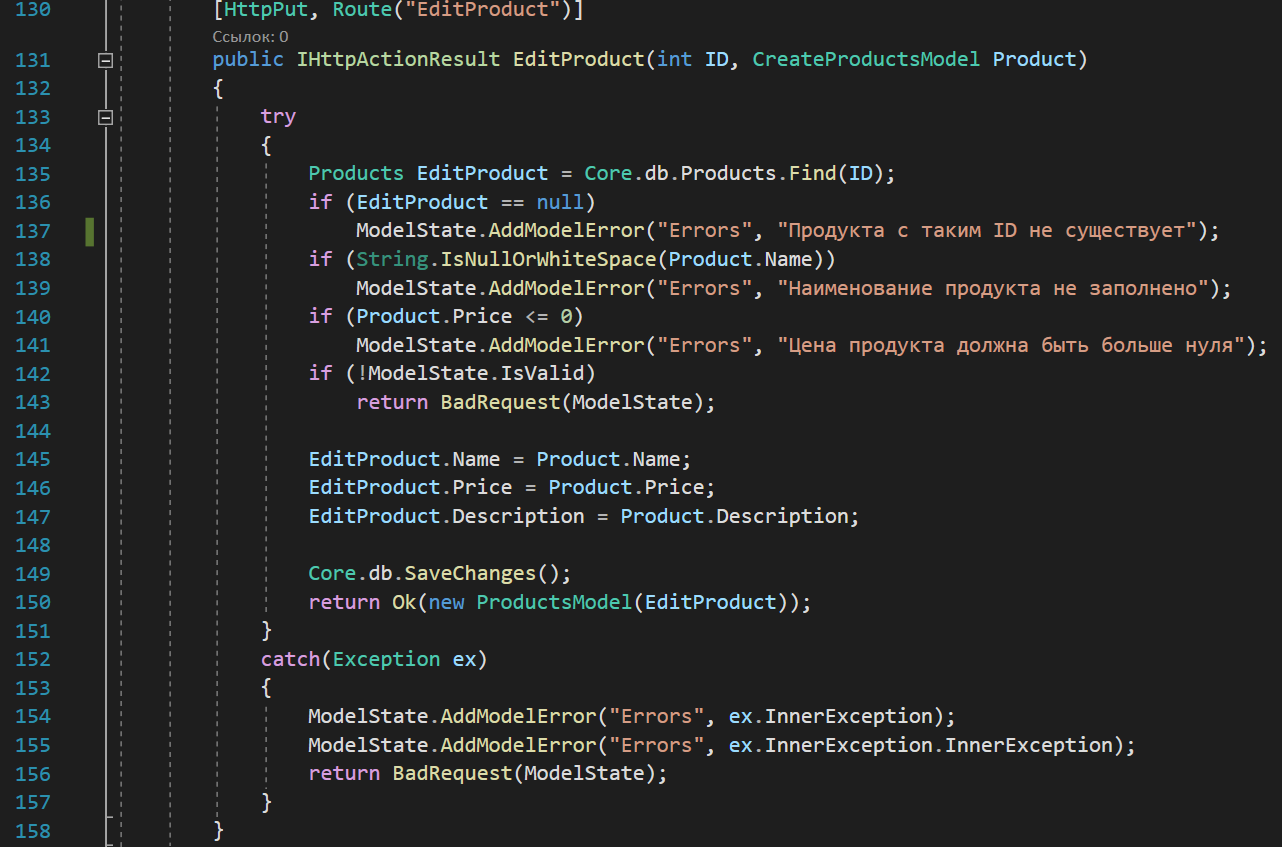


Рисунок 16. Метод изменения записи о продукте

Реализация контроллера API для таблицы «Orders».

Код для API таблицы «Orders» был выполнен по подобию контроллера «Products» и содержит в себе те же самые 4 метода для CRUD-операций.

Реализация контроллера API для таблицы «ProductsInOrder».

Метод для получения списка всех продуктов из заказа (Рисунок 17) не сильно отличается от предыдущих двух API.

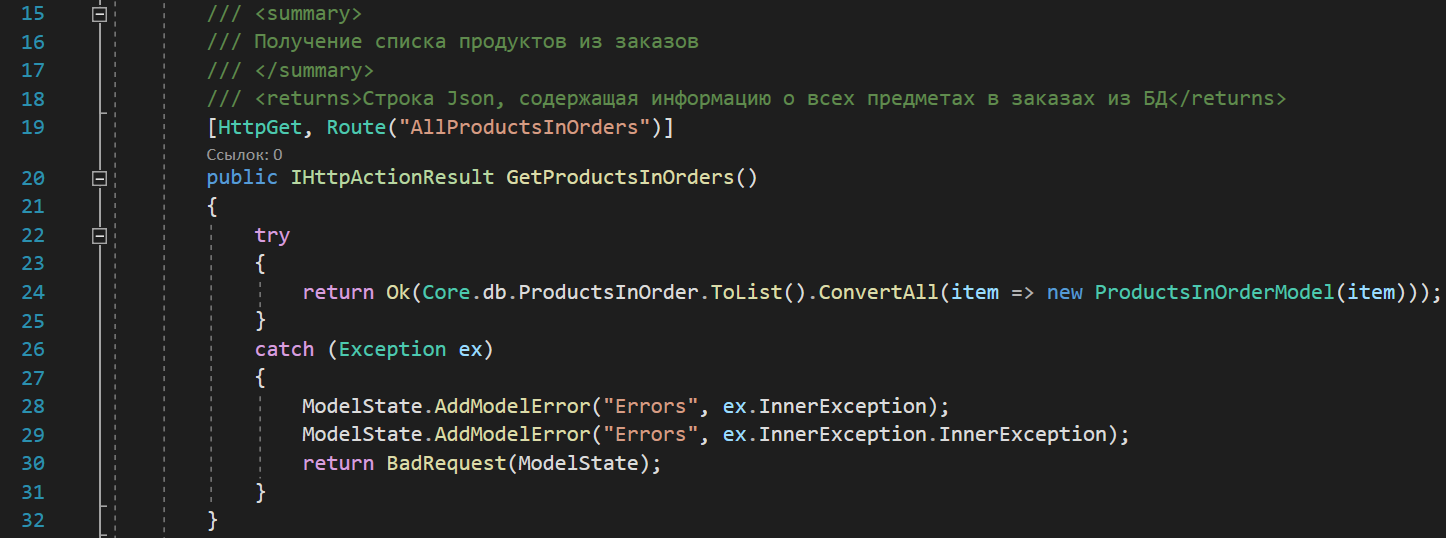


Рисунок 17. Получение списка продуктов, находящихся в заказах

Пример Json строки, которую возвращает данный метод показан на рисунке 18.

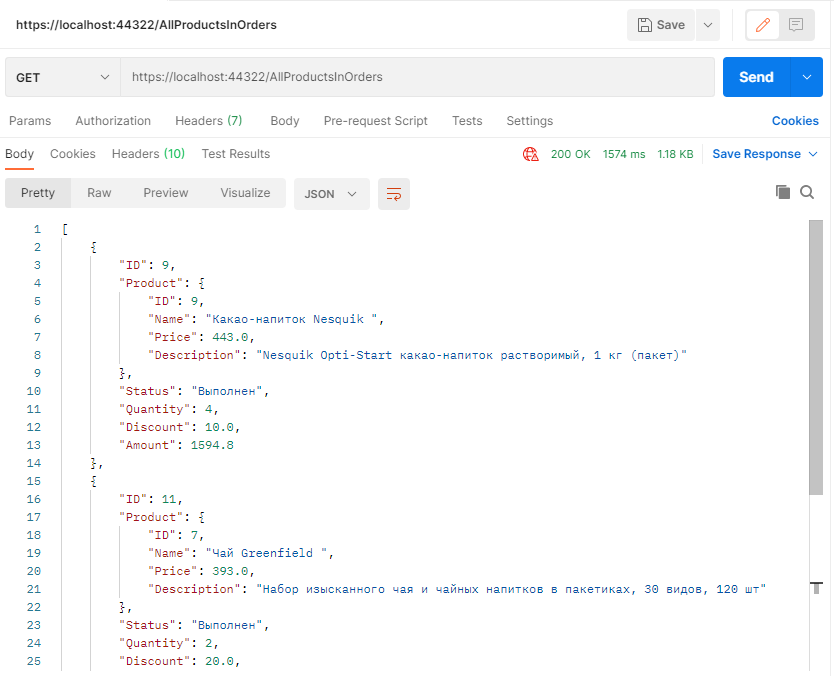


Рисунок 18. Пример ответа от API «AllProductsInOrders»

API «DetailsProductsInOrders» (Рисунок 19) возвращает информацию о конкретной записи из заказа с соответствующим ID, если такая запись действительно существует.

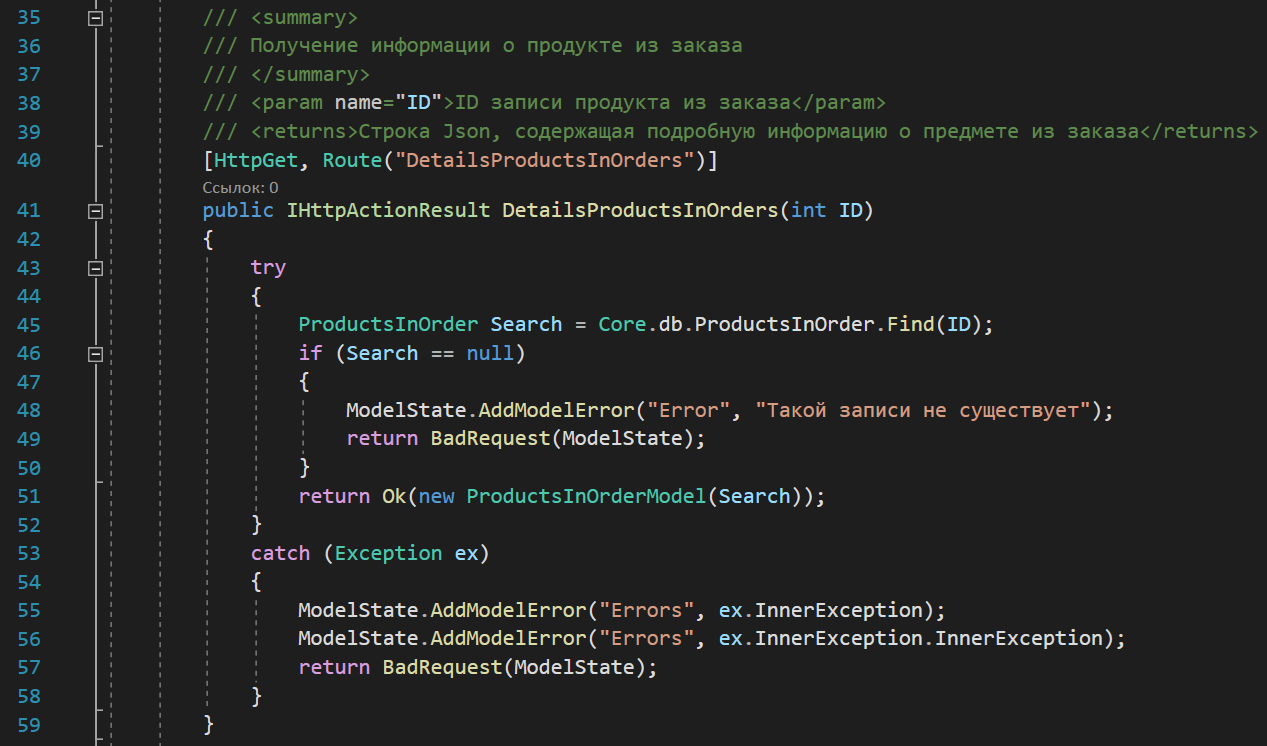


Рисунок 19. Метод для получения информации по продукту из заказа

Результат, возвращаемый этим методом показан на рисунке 20.

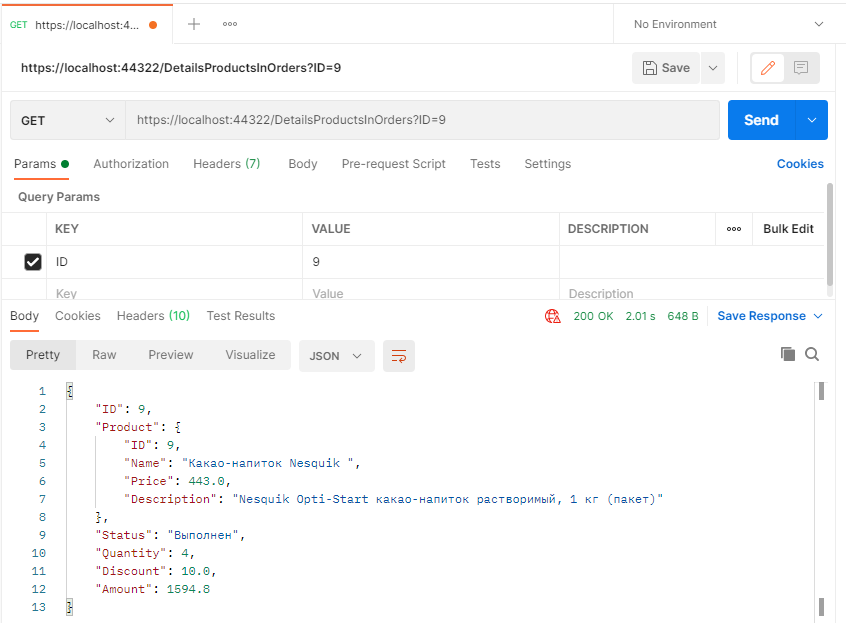


Рисунок 20. Строка Json, возвращаемая при обращении к «DetailsProductsInOrders»

Следующий метод, представленный на рисунках 21 и 22, осуществляет проверку переданных данных и если все условия удовлетворены, то создает новый заказ с добавлением в него продукта.

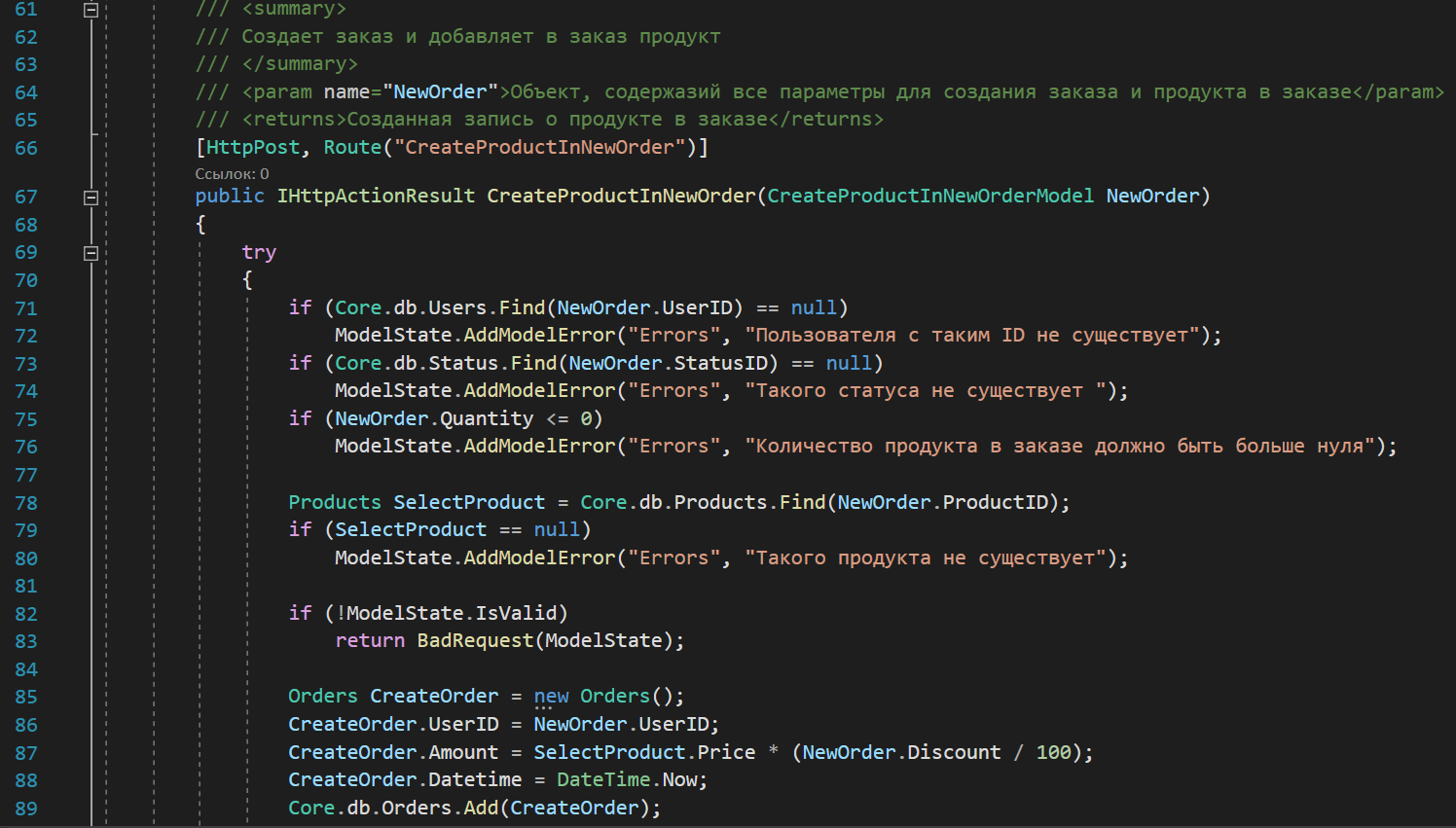


Рисунок 21. Добавление продукта в новый заказ (часть 1)

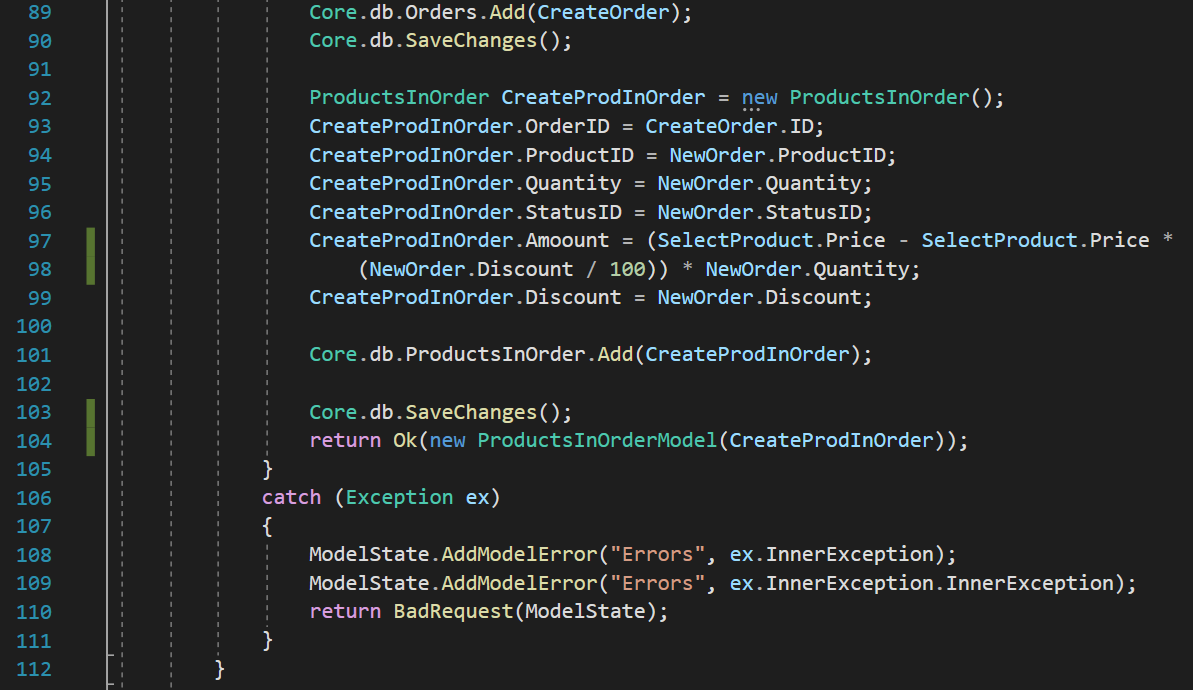


Рисунок 22. Добавление продукта в новый заказ (часть 2)

Пример запроса к этому API и ответа от него при передаче корректной Json строки, показан на рисунке 23.

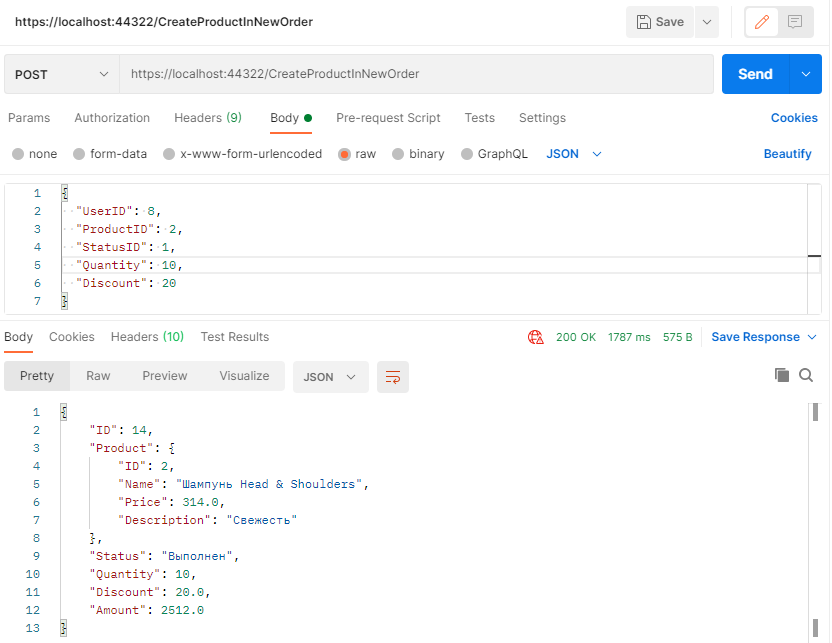


Рисунок 23. Ответ от API CreateProductInNewOrder

Также я посчитал, что должна быть возможность добавлять несколько продуктов в один заказ, а для этого был написан еще один метод (Рисунок 24 и 25), выполняющий функцию добавления одного товара в заказ при соблюдении четырех условий.

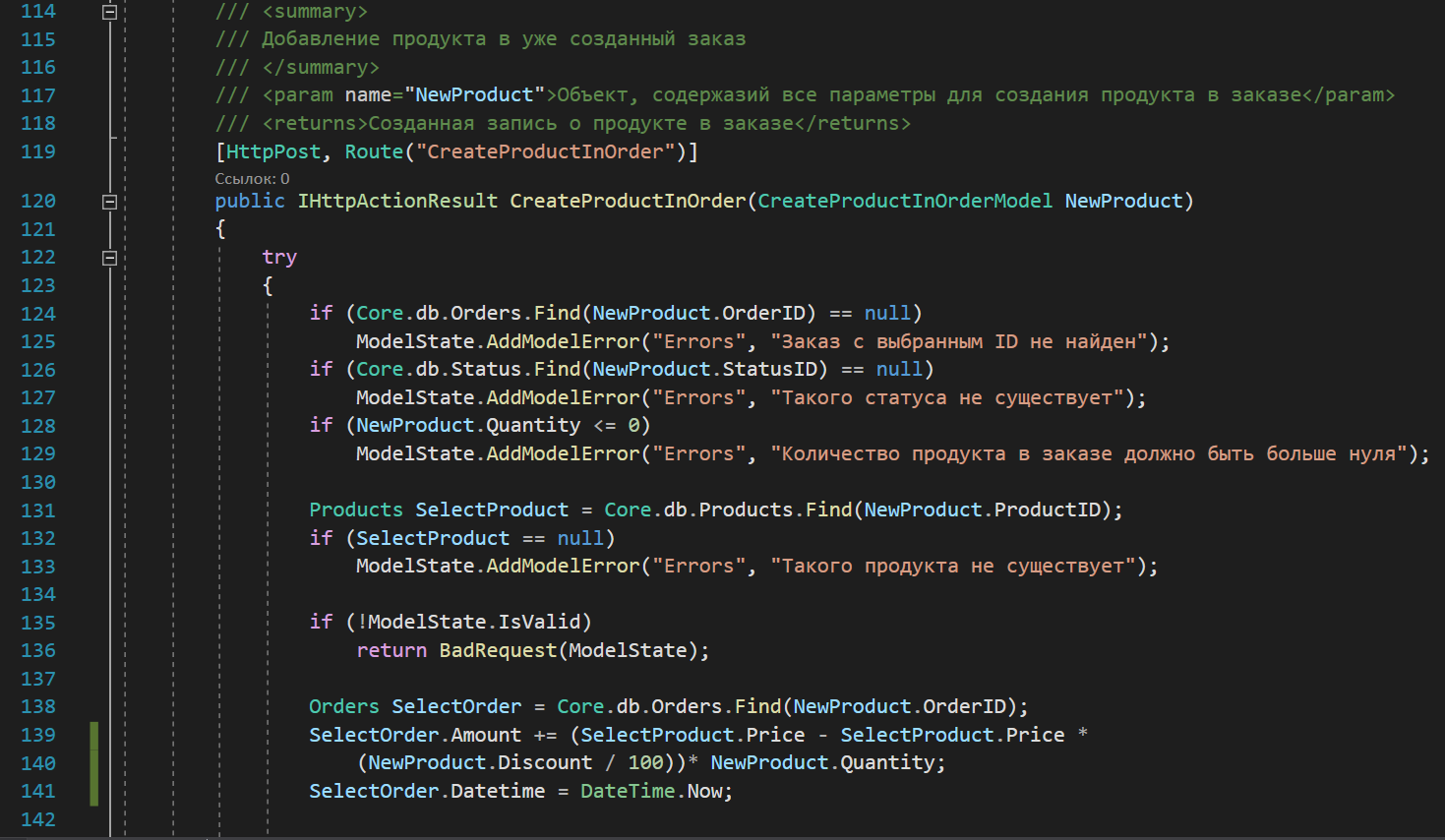


Рисунок 24. Метод добавления продукта в существующий заказ (часть 1)

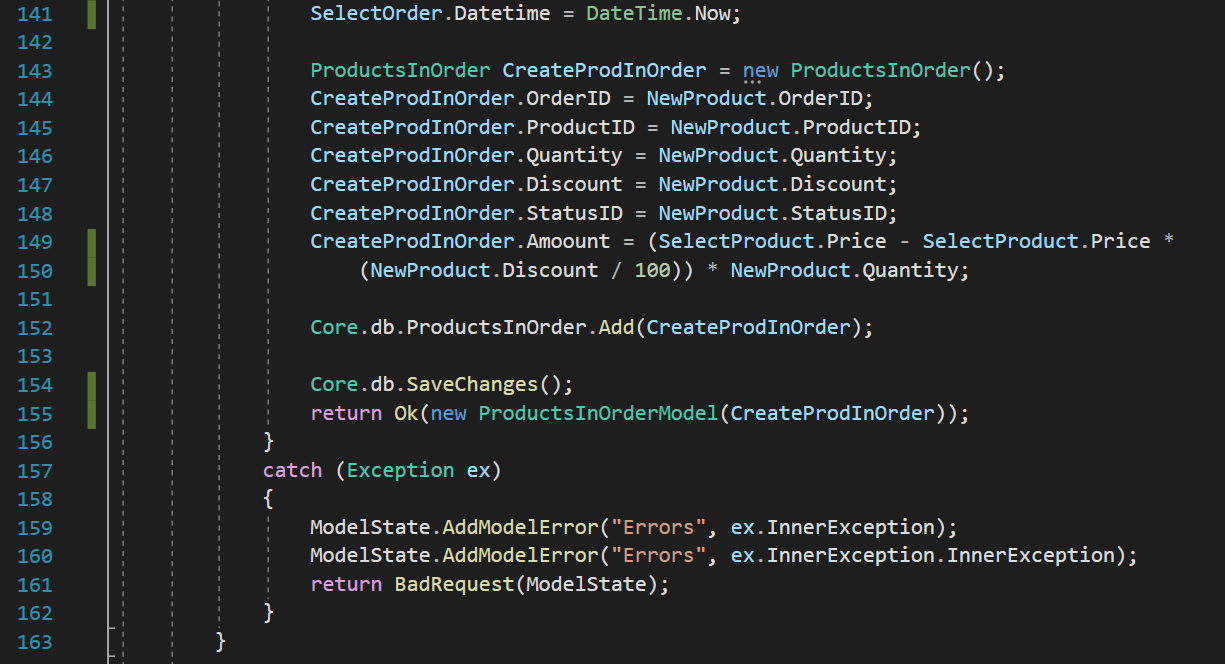


Рисунок 25. Метод добавления продукта в существующий заказ (часть 2)

Удаление продукта из заказа выполняет следующий метод контроллера для сущности «ProductsInOrder» (Рисунок 26). При возможности удаления итоговая стоимость заказа пересчитывается.



Рисунок 26. API, удаляющее продукт из заказа

Последний реализованный метод позволяет изменять существующую запись о продукте, находящегося в заказе, показан на рисунках 27 и 28.

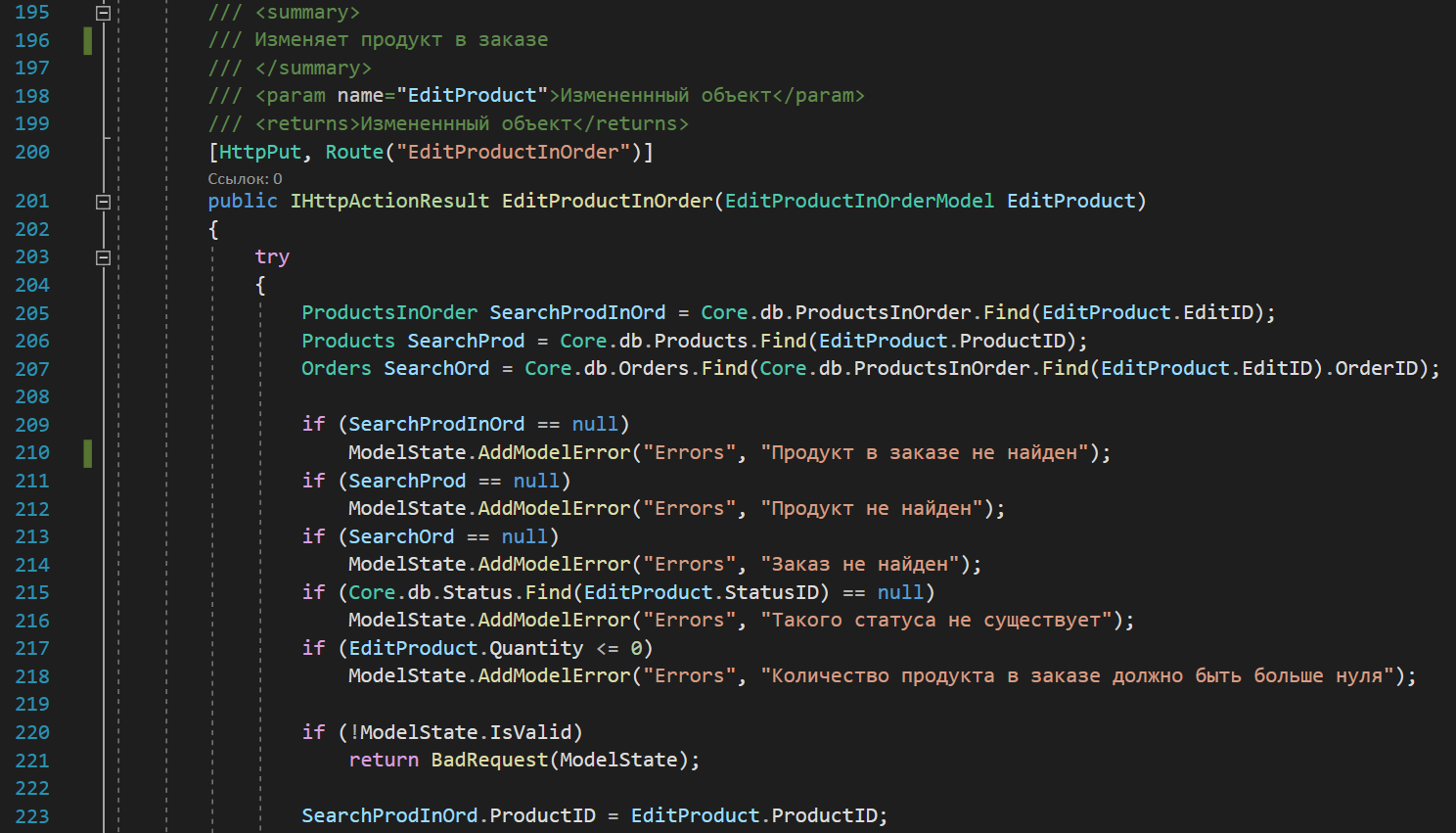


Рисунок 27. API для изменения информации о продукте в заказе (часть 1)

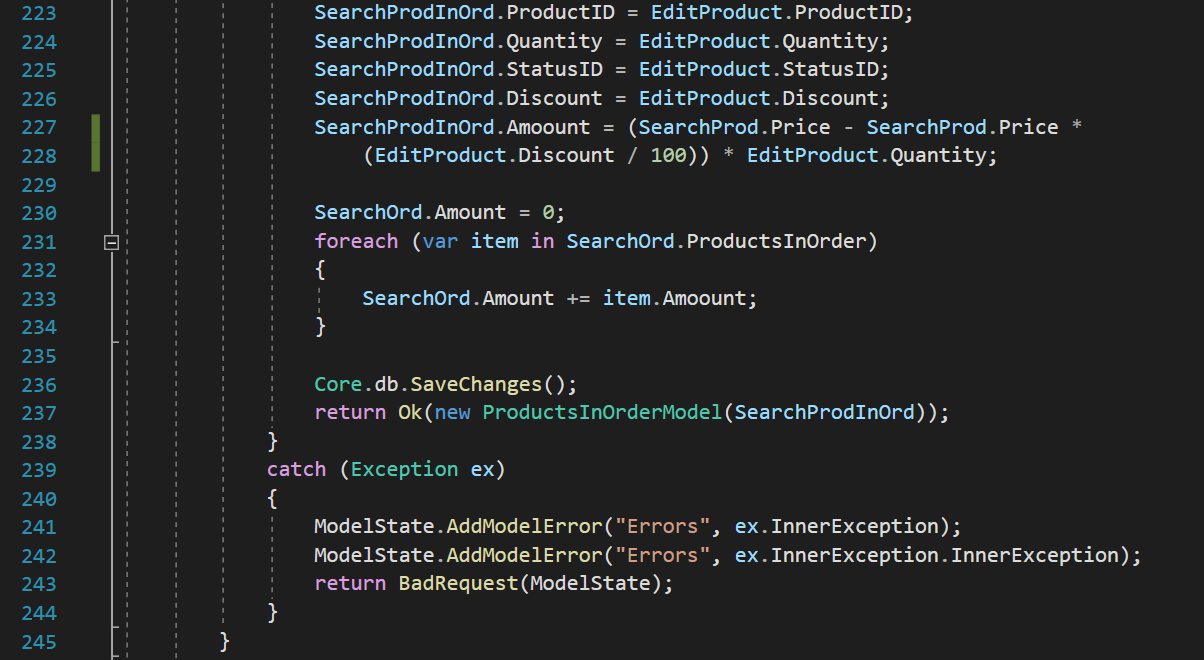


Рисунок 28. API для изменения информации о продукте в заказе (часть 2)

Пример корректного запроса к этому методу показан на рисунке 29.

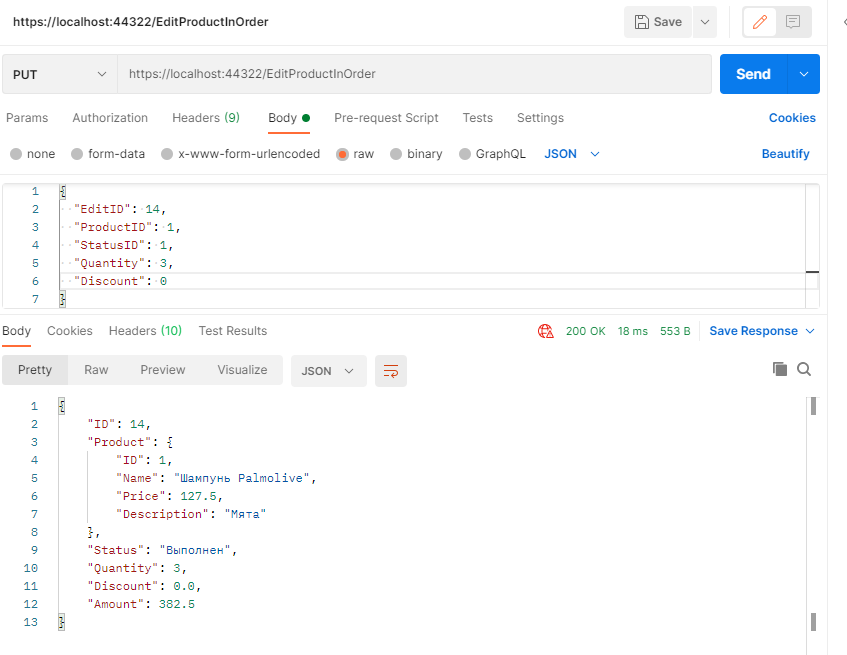


Рисунок 29. Изменение записи о продукте в заказе

На этом создание API в рамках задания можно считать выполненным.