МИНОБРНАУКИ РОССИИ

федеральное государственное бюджетное образовательное учреждение

высшего образования

«Московский государственный технологический университет «СТАНКИН»

(ФГБОУ ВО «МГТУ «СТАНКИН»)

Институт
информационных технологий

Кафедра
информационных систем

Основная образовательная программа 09.03.02
«Информационные системы и технологии»

Отчет по дисциплине «Управление рисками и надежностью информационных систем»

по лабораторной работе №2-3

Тема: «Postman»

Проверил
преподаватель **Петруша А.О.**

подпись

Выполнил
студент группы ИДБ-21-06 **Музафаров К.Р.**

подпись

Москва, 2023 г.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Вторая и третья лабораторная работа посвящена изучению возможностей приложения «Postman», таких как создание рабочего пространства, коллекции, составлению и отправки запросов с разными методами, а также изучению возможностей приложения «Postman», таких как создание папок, глобальных переменных и переменных уровня коллекции, работа со скриптами.

ГЛАВА 1. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ.

Основное назначение Postman – облегчить разработку, тестирование и документирование API. С его помощью можно отправить данные в запросе и проверить полученный ответ. Также у него есть много других интересных возможностей. Можно, например, сохранять запросы в папки и коллекции, удобно параметризировать запросы. Запускать коллекции с помощью Collection Runner и использовать их как автоматизированные тесты.

Тестировщики, с помощью Postman могут отправлять HTTP/HTTPS запросы к сервисам и получать ответы от них. С помощью такого подхода можно протестировать бэкенд сервисы и убедиться, что они работают корректно. С помощью Postman можно выполнять запросы к различным API, таким как REST, SOAP и GraphQL.

Основная сущность в Postman – рабочее пространство (Workspaces). В бесплатной версии доступно три вида рабочих пространств:

- личное (personal) – видно только владельцу;
- командное (team) – видно только членам команды, которые в нём работают;
- публичное (public) – доступно всем желающим.

В Pro-версии появляется доступ к приватным рабочим пространствам (private) – такие пространства доступны не всем членам команды, а только приглашённым. В Enterprise-версии добавляются партнёрские (partner) – они видны не только членам команды, но и приглашённым партнёрам.

Окно приложения после создания рабочего пространства состоит из:

- Верхнего меню.
- Нижнего меню.
- Бокового меню.
- Основной зоны работы.

Верхнее меню дает возможность создания рабочего пространства (Workspaces) и доступа к различным API (API Network). Также с его помощью

можно делать поиск. «Home» – содержит домашнюю страницу, на ней можно посмотреть недавнюю активность. С помощью «Workspaces» – можно создать новое рабочее пространство, выбрать одно из недавно посещенных, либо сделать поиск по существующим.

В боковом меню доступны основные функциональные элементы «Postman». В истории (History) можно увидеть сделанные ранее запросы. Для более удобного хранения, группировки и поиска таких запросов в «Postman» есть коллекции (Collections). Для выполнения данной лабораторной ознакомления с этими функциями достаточно. Другие вкладки будут рассмотрены позднее.

В «Postman» можно группировать запросы и примеры ответов в коллекции. Это поможет удобнее организовать рабочее пространство, создавать тестовую документацию и тесты для API. Для открытия коллекций нужно нажать на вкладку «Collections», расположенную в боковом меню.

Основная функциональность «Postman» – возможность создания и отправки запросов к API для проверки его функциональности и получения данных. Для этого не потребуется писать код или команды в терминале. В интерфейсе «Postman» создается запрос, нажимается кнопка отправить и получается ответ от API. API расшифровывается как Application Programming Interface или программный интерфейс приложения. С его помощью можно получить доступ к возможностям другого приложения и обмениваться с ним данными. Такое приложение называется API-сервером. Отправка запросов и получение ответов происходит через интернет с помощью протокола HTTP. Приложение, которое отправляет запрос, называется клиентом. Это может быть мобильное приложение, веб-сайт или другой сервис.

Запрос всегда содержит URL вызываемого эндпоинта API и HTTP-метод запроса. Эндпоинт – это URL. Он позволяет клиентским приложениям отправлять запросы на сервер и получить доступ к данным или выполнить определенные операции. Каждый эндпоинт обычно связан с определенным HTTP-методом, таким как, например, GET или POST, который определяет тип

операции, которую можно выполнить с помощью этого эндпоинта. Наиболее часто используются следующие методы:

- GET – для чтения данных.
- POST – для добавления новых данных.
- PUT – для обновления данных.
- PATCH – для частичного обновления данных.
- DELETE – для удаления данных.

«Postman» позволяет группировать запросы и примеры ответов не только в коллекции, но и в папки внутри этих коллекций. Это помогает удобнее организовать рабочее пространство, создавать тестовую документацию и тесты для API.

В «Postman» можно создавать переменные, позволяющие сохранить и повторно использовать различные значения. После создания переменной, ее значение можно использовать в коллекциях, окружениях, запросах или тестовых скриптах, ссылаясь с помощью имени этой переменной.

Переменные в «Postman» – это пары ключ-значение. Имя переменной является ключом, по которому к ней можно обратиться для получения доступа к ее значению. Также с помощью переменных можно передавать значения между запросами и тестами.

Таким образом в переменной можно хранить какие-либо данные. Добавление переменных бывает полезно, когда нужно использовать одинаковые данные в нескольких местах. Они позволяют делать запросы более читабельными и гибкими, поскольку появляется возможность задавать необходимые данные в одном месте.

При этом, в переменных можно хранить не только параметры, но и, например, часть url (в разных запросах может быть одинаковая или повторяющаяся часть адреса). Также может быть часть адреса, которая изменится в будущем. Ее удобно вынести в переменную, чтобы затем производить изменения только в одном месте.

Существует несколько способов добавления переменной. Для одного из них необходимо выделить уже имеющееся значение path-параметра запроса, после чего в всплывшей подсказке нажать «Set as variable», а затем выбрать «+ Set as new variable». В новом окне нужно будет указать имя переменной и ее область видимости. Стоит иметь в виду, что в данном способе добавления значение переменной уже задано и изменить его не получится.

После добавления переменной таким способом вместо значения параметра будет отображаться название созданной переменной в двойных фигурных скобках. Если навести на имя переменной появится всплывающая подсказка., в которой можно увидеть имя переменной, ее значения и область видимости.

Глобальные переменные доступны внутри всего рабочего пространства. Они имеют самую широкую область видимости из возможных в «Postman» (область видимости определяет, из какой части программы переменная будет доступна и может быть использована).

Переменные коллекции доступны для всех запросов внутри коллекции. Они не зависят от окружения, и их значения не меняются в зависимости от выбранного окружения. Такие переменные можно посмотреть на вкладке «Variables».

Помимо этого, в «Postman» есть динамические переменные, которые иногда бывает полезно использовать в своих запросах для создания случайных данных. Для каждого запроса будет создано новое значение, отличное от предыдущего. Воспользоваться ими можно указав перед названием знак доллара:

- Переменная {{$randomInt}} – возвращает случайное целочисленное значение в промежутке между 0 и 1000.
- {{$randomColor}} – возвращает случайный цвет.
- {{$randomFileName}} – возвращает случайно сгенерированное имя файла.
- И тому подобное.

ГЛАВА 2. ПРАКТИЧЕСКАЯ ЧАСТЬ

2.1. ЛАБОРАТОРНАЯ РАБОТА №2

Регистрируем аккаунт в «Postman» и входим в него (рис. 2.1.1)

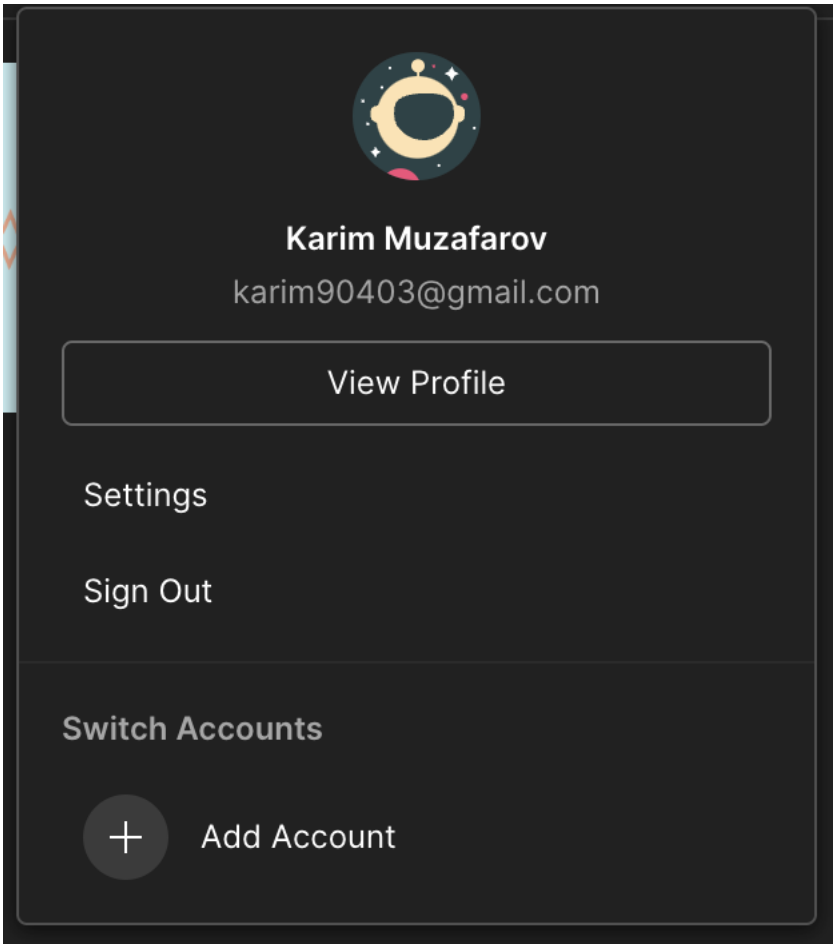

Рис. 2.1.1. Аккаунт в «Postman»

Создаем рабочее пространство и указываем номер группы и подгруппу в кратком описании, также создаем отдельные коллекция для лабораторных работ. Результат выполнения представлен на рис. 2.1.2-2.1.3

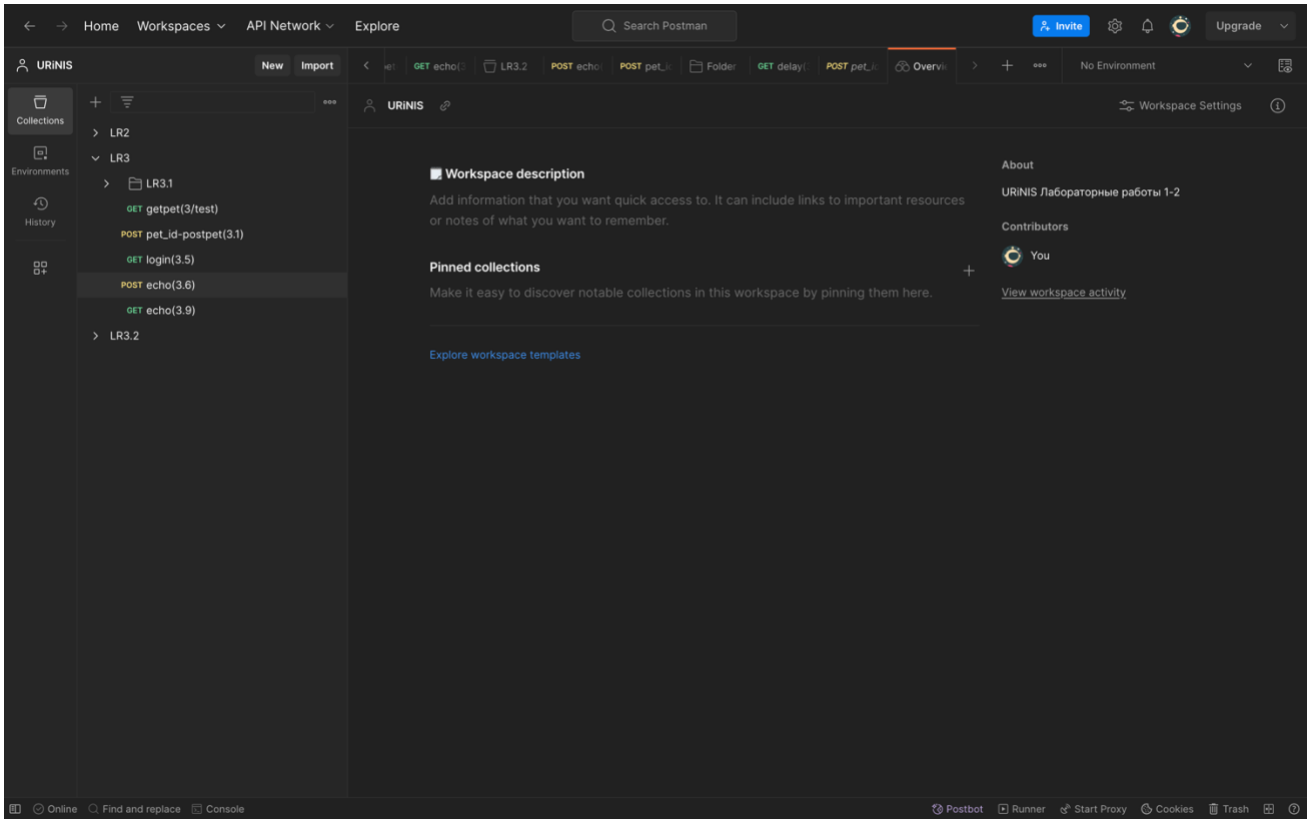

Рис. 2.1.2. Рабочее пространство в «Postman»

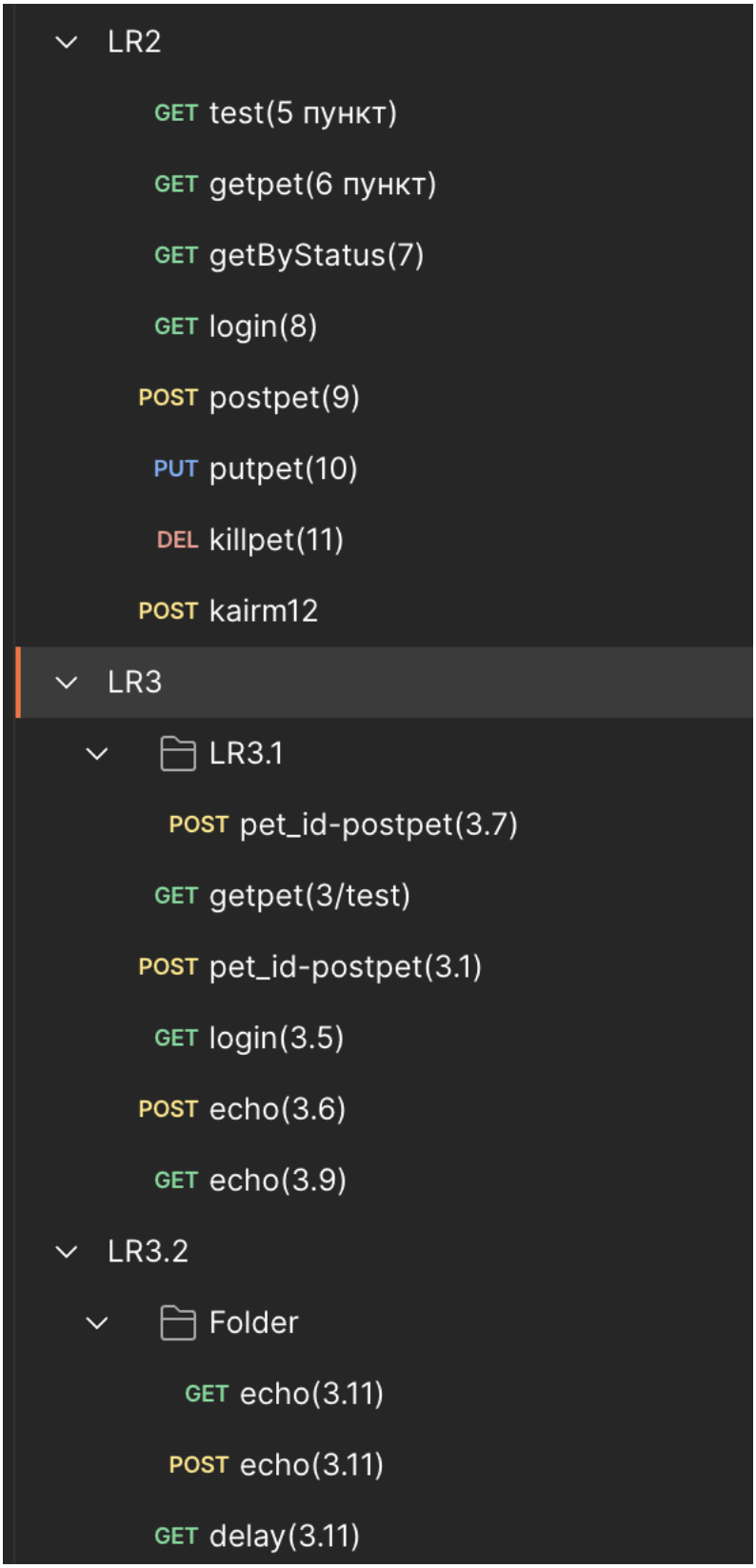

Рис. 2.1.3. Коллекции для лабораторных работ

Отправляем GET запрос к сайту «API Swagger Petstore». Результат выполнения представлен на рис. 2.1.4.

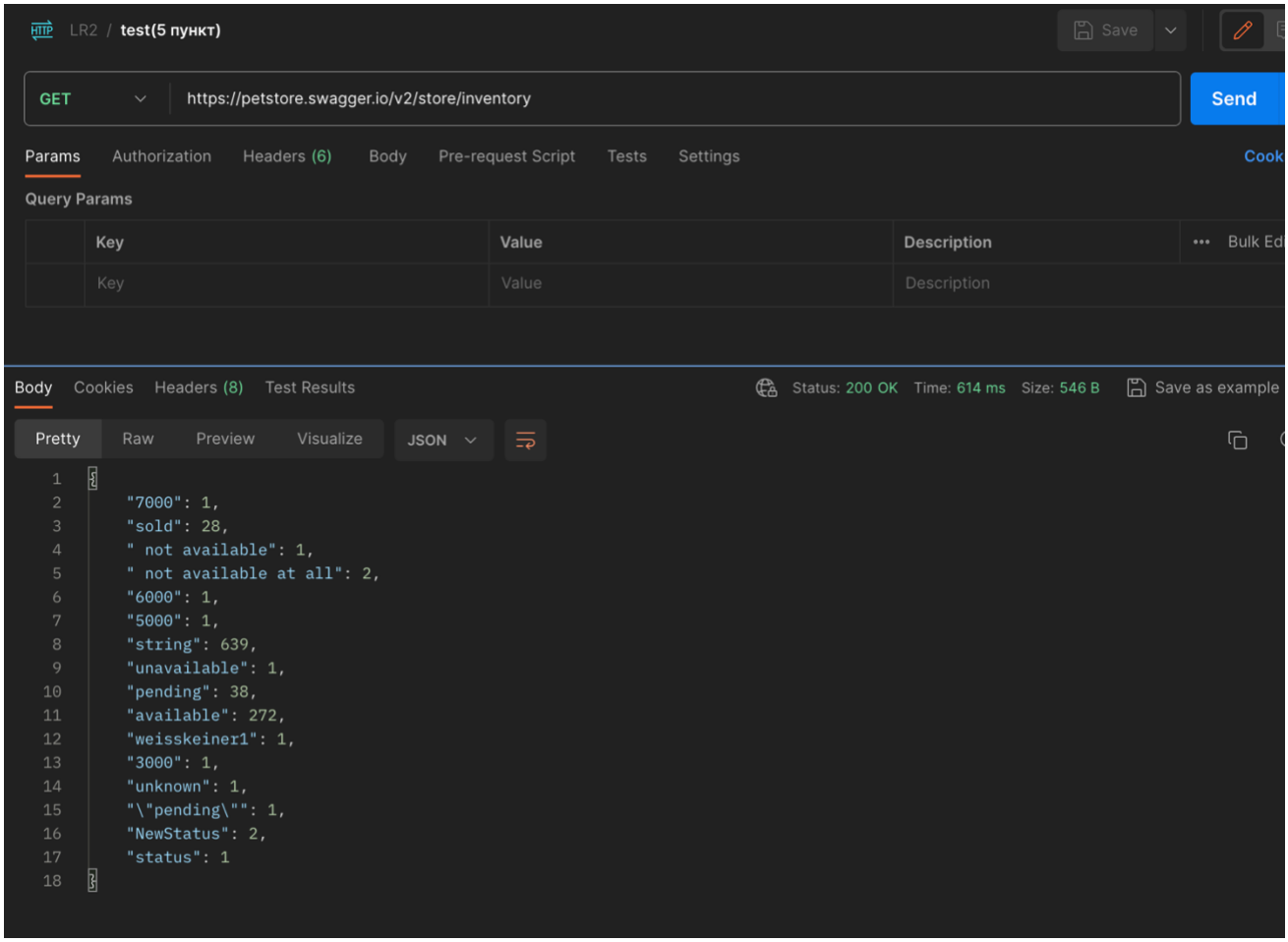

Рис. 2.1.4. Get запрос к сайту «API Swagger Petstore»

Отправляем GET запрос с path параметром для получения информации о домашнем питомце по его идентификатору в системе. Результат выполнения представлен на рис. 2.1.5.

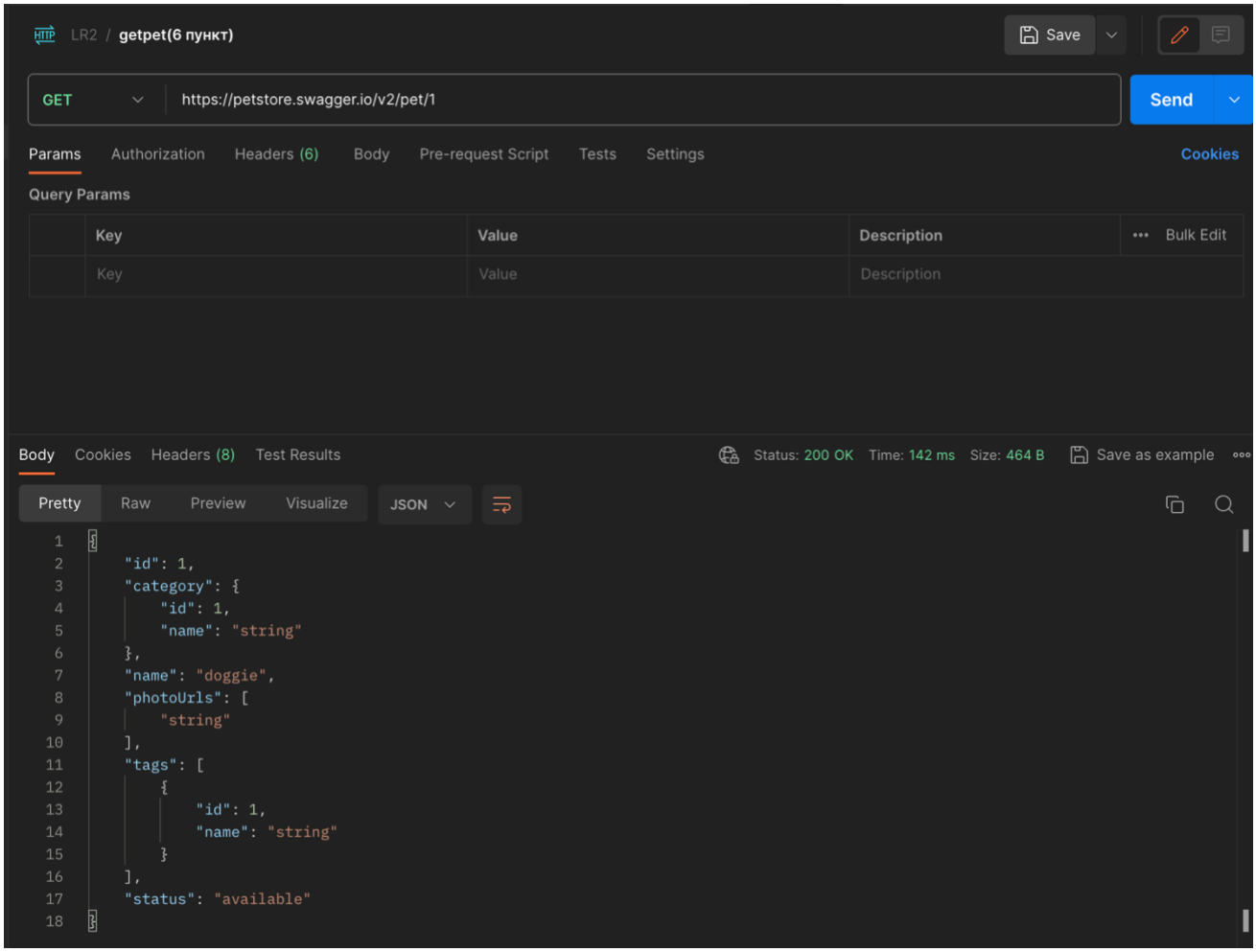

Рис. 2.1.5. GET запрос с параметром для получения информации о домашнем питомце по его идентификатору в системе

Отправляем GET запрос с query параметром, который позволяет получить информацию о домашних питомцах по одному из статусов. У параметра может быть одно из трех значений: «available», «pending» или «sold». Результат выполнения представлен на рис. 2.1.6.

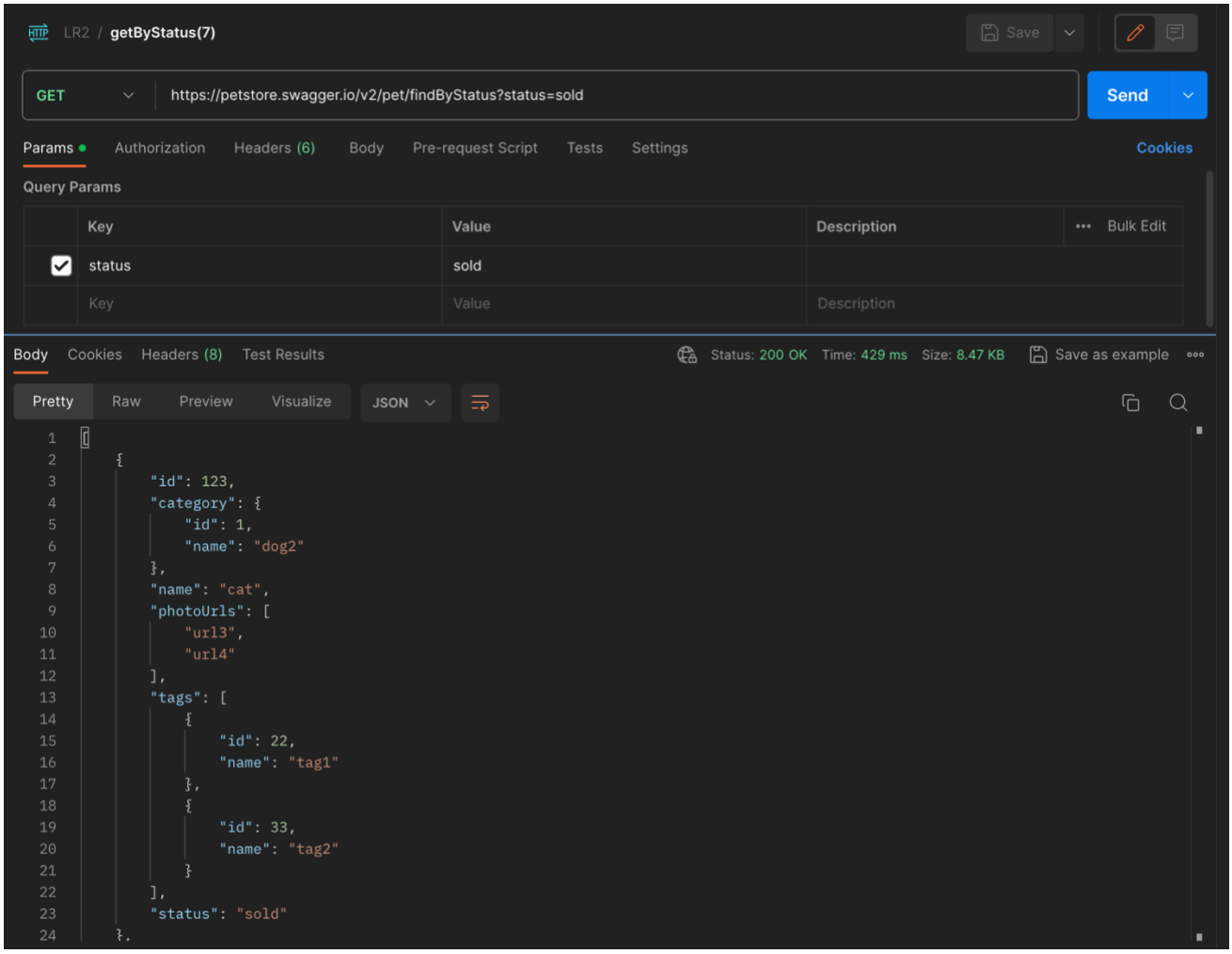

Рис. 2.1.6. GET запрос с query параметром, который позволяет получить информацию о домашних питомцах со статусом «sold».

Отправляем GET запрос с двумя параметрами (имя пользователя и пароль). Результат выполнения представлен на рис. 2.1.7.

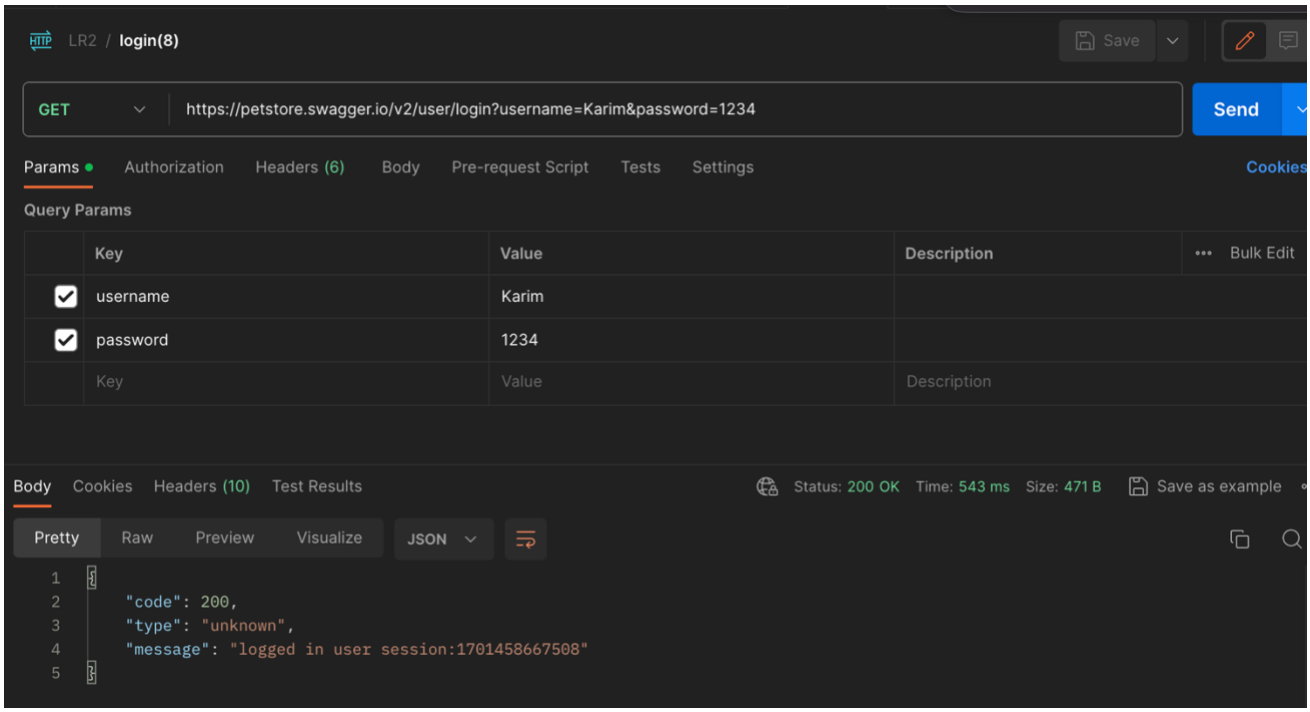

Рис. 2.1.7. GET запрос с двумя параметрами (имя пользователя и пароль)

Отправляем POST запрос для создания записи о новом домашнем питомце. Результат выполнения представлен на рис. 2.1.8.

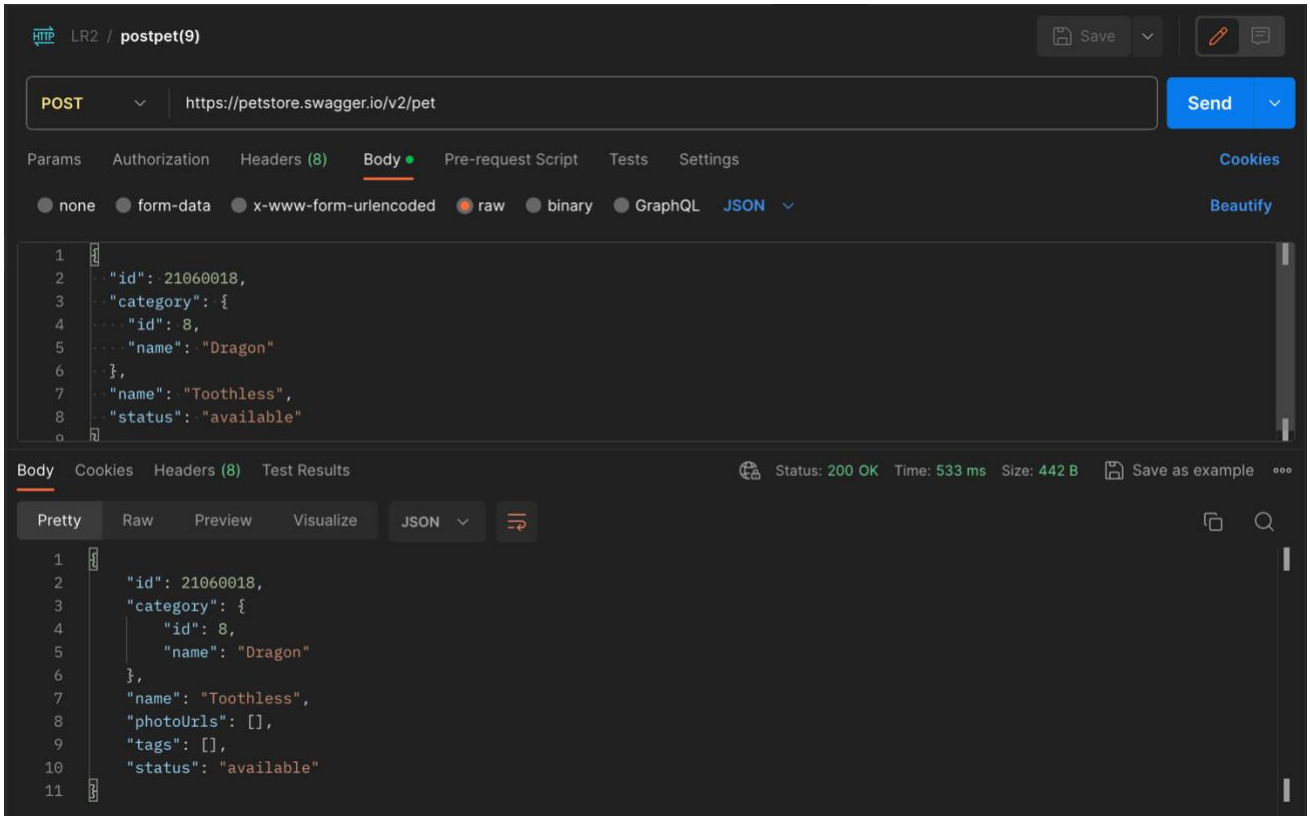

Рис. 2.1.8. POST запрос для создания записи о новом домашнем питомце

Отправляем PUT запрос для обновления данных из предыдущего запроса. Для этого необходимо повторить действия 9 пункта, изменения будут только в типе запроса и в коде его тела. В данном запросе необходимо поменять имя питомца, статус и добавить параметр «photoUrls», содержащий в себе ссылку на картинку и убрать категорию, указанную при создании. Результат выполнения представлен на рис. 2.1.9.

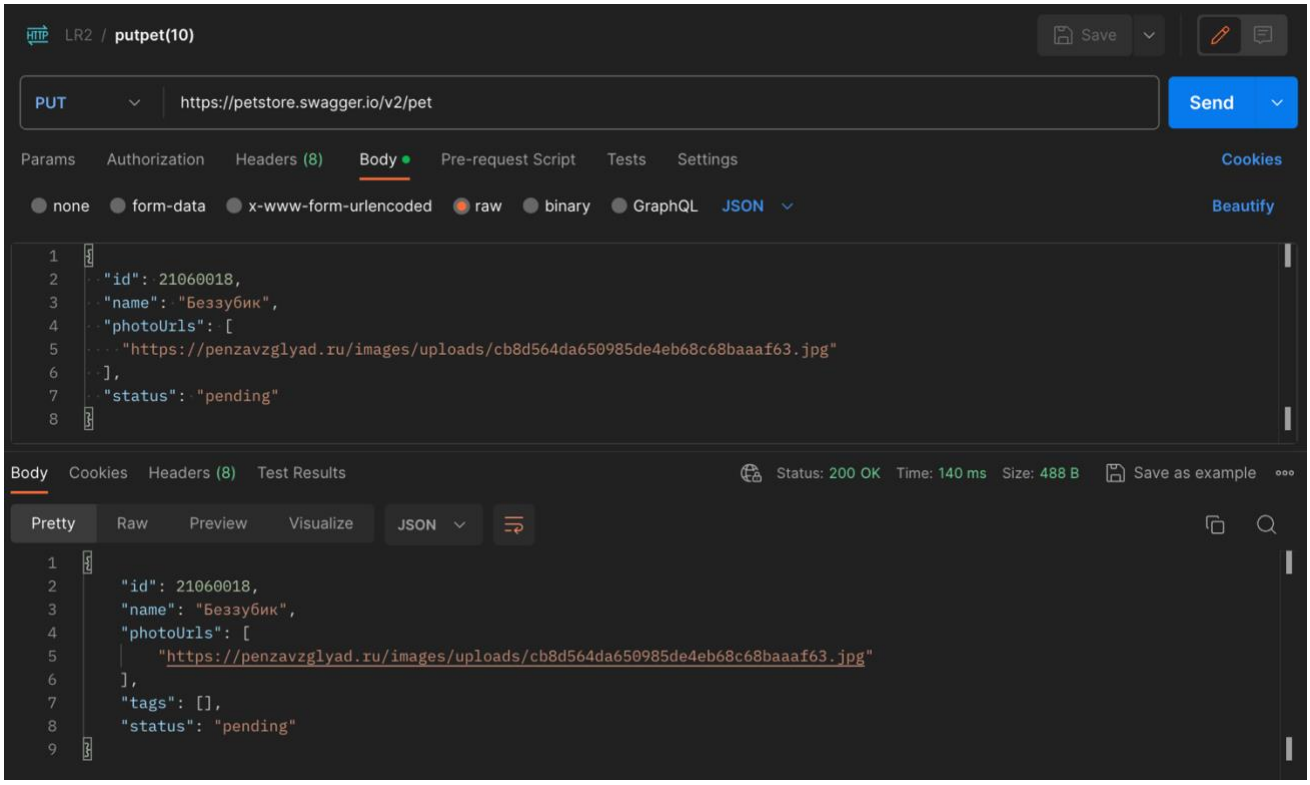

Рис. 2.1.9. PUT запрос для обновления данных из предыдущего запроса.

Удаляем созданную на рис. 2.1.8 запись с помощью DELETE запроса и проверяем факт удаления с помощью GET запроса. Результаты выполнения представлен на рис. 2.1.10-2.1.11.

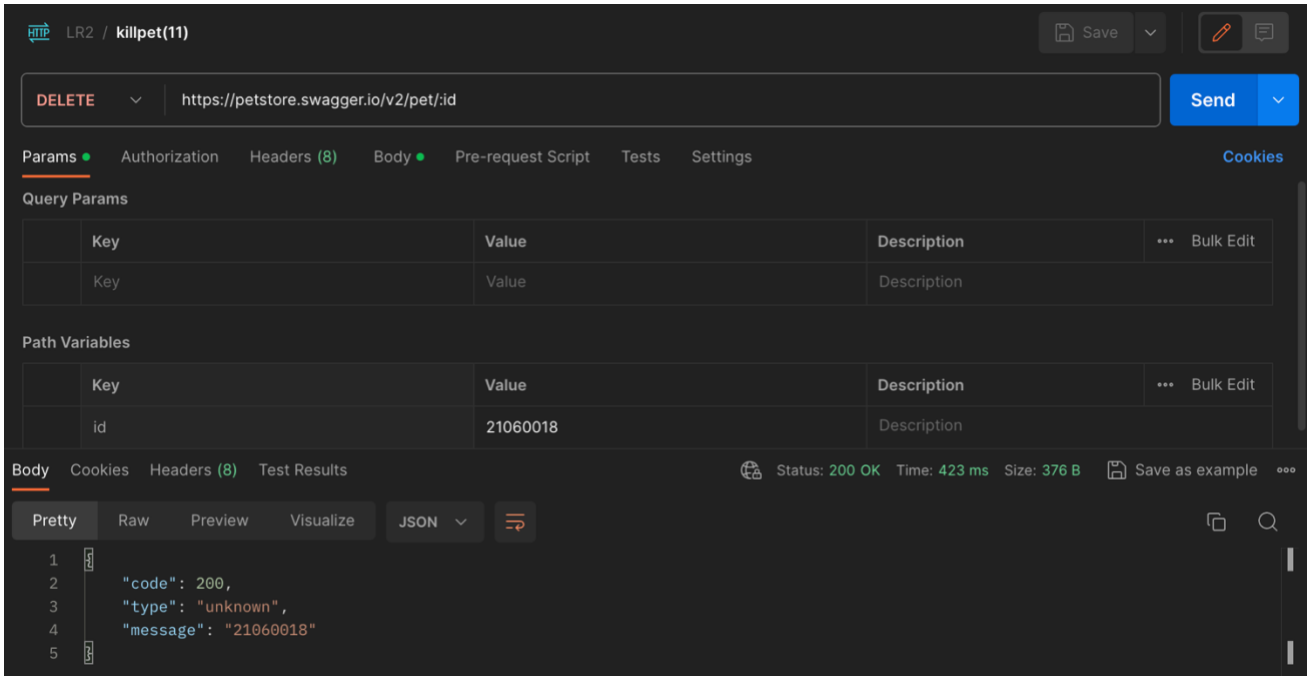

Рис. 2.1.10. DELETE запрос

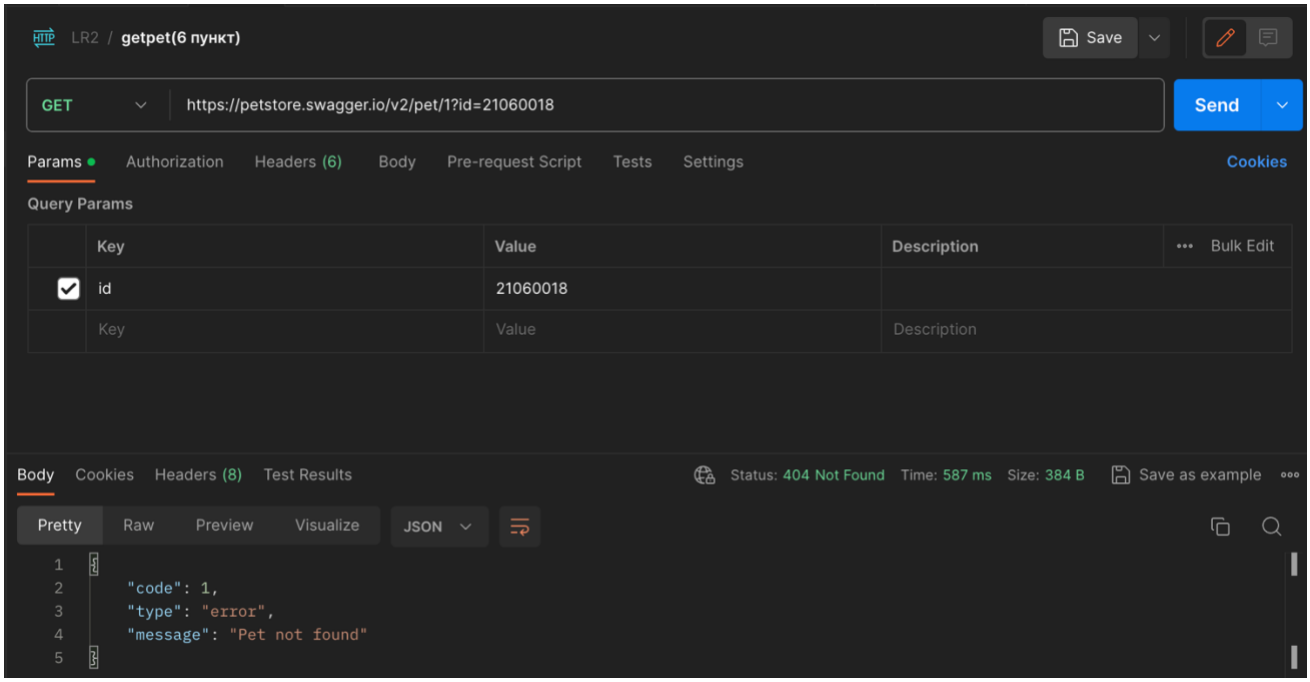

Рис. 2.1.11. Проверка удаления после применения DELETE запроса

Отправляем POST запрос с эндпоинтом «https://postman-echo.com/post?name=Karim». Результаты выполнения представлен на рис.2.1.12-2.1.14.

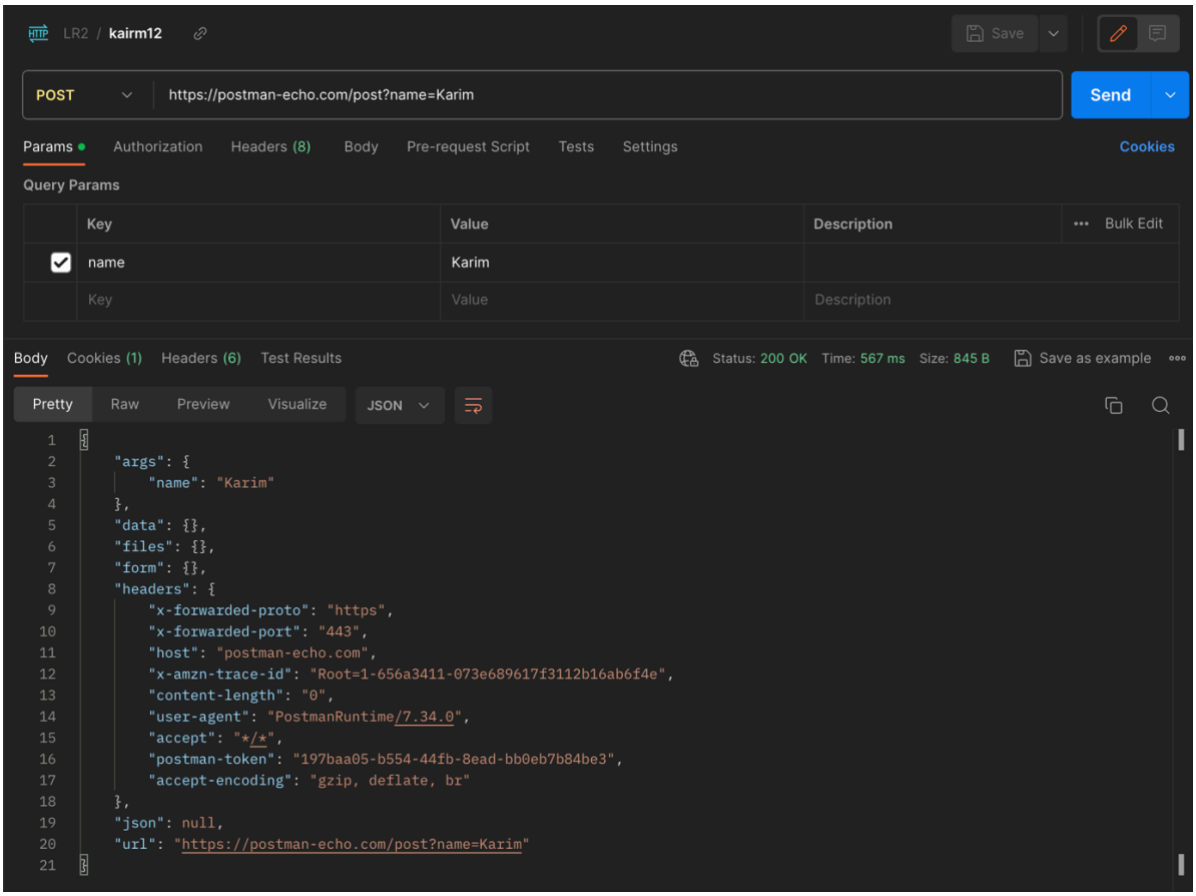

Рис. 2.1.12. ответ от сервера на запрос во вкладке «Pretty»

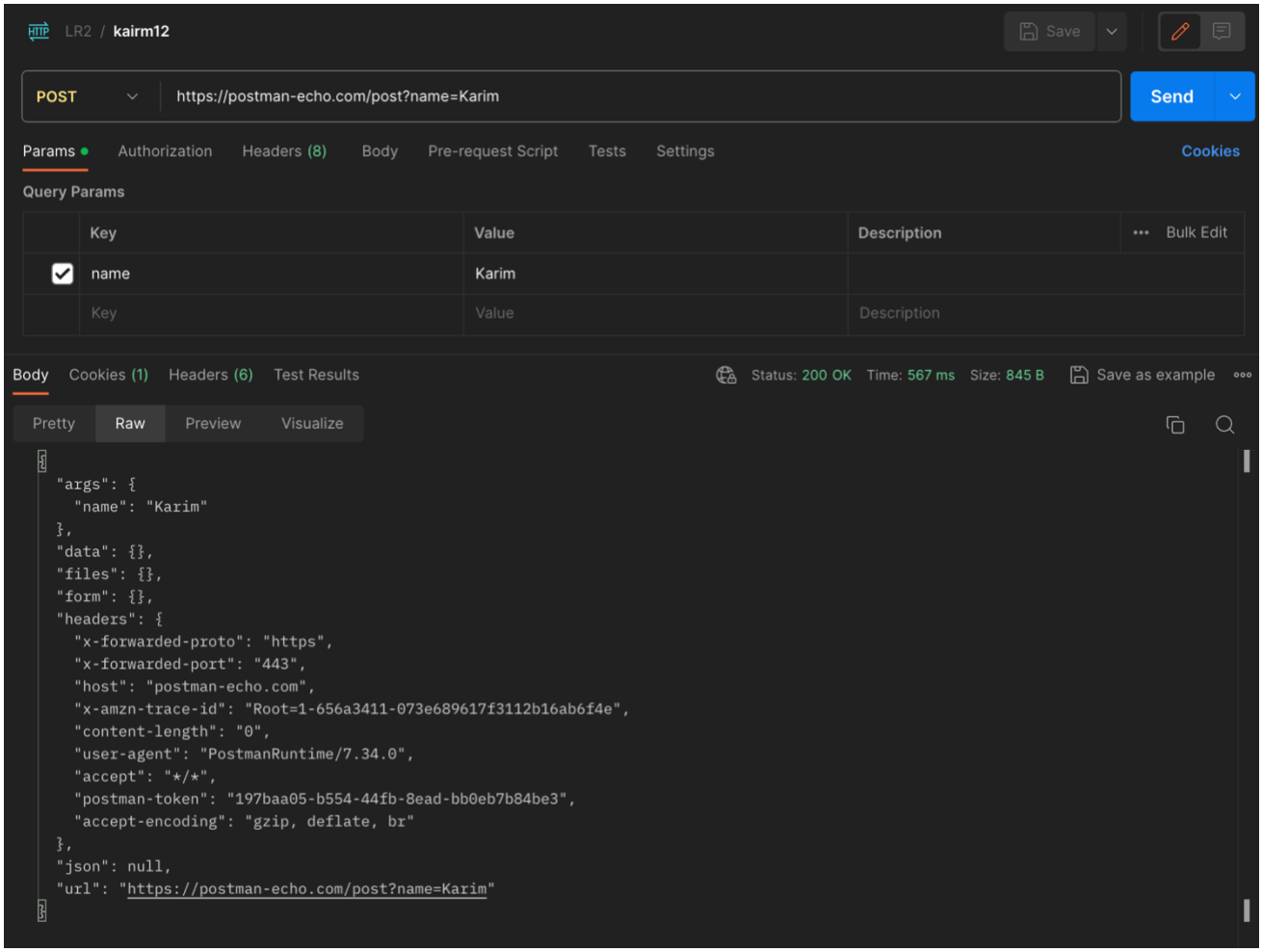

Рис. 2.1.13. ответ от сервера на запрос во вкладке «Raw»

Рис. 2.1.14. Ответа от сервера на данный запрос во вкладке «Preview»

В запросе из предыдущего задания перешли на вкладку «Headers». Результат выполнения представлен на рис. 2.1.15.

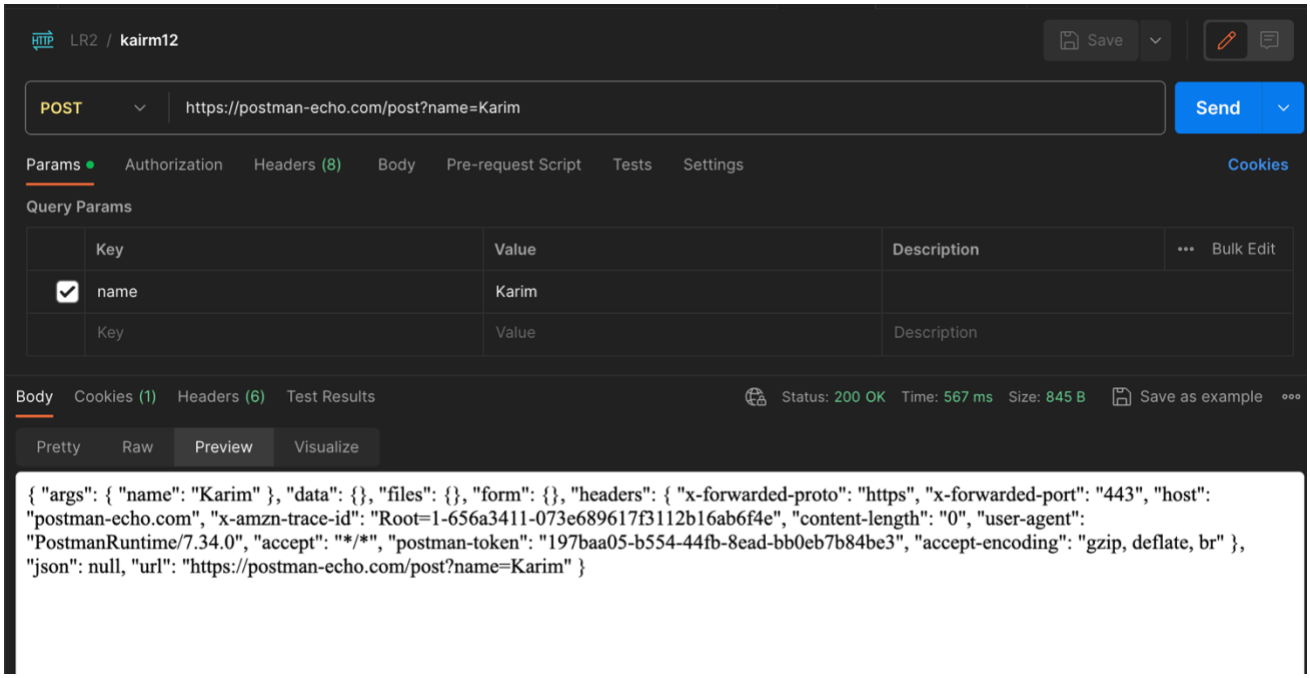

Рис. 2.1.15. вкладка «Headers»

Отобразили дополнительную информацию об ответе на тот же запрос (сетевую информацию, код, время и размер ответа). Результаты выполнения представлен на рис. 2.1.16-2.1.19.

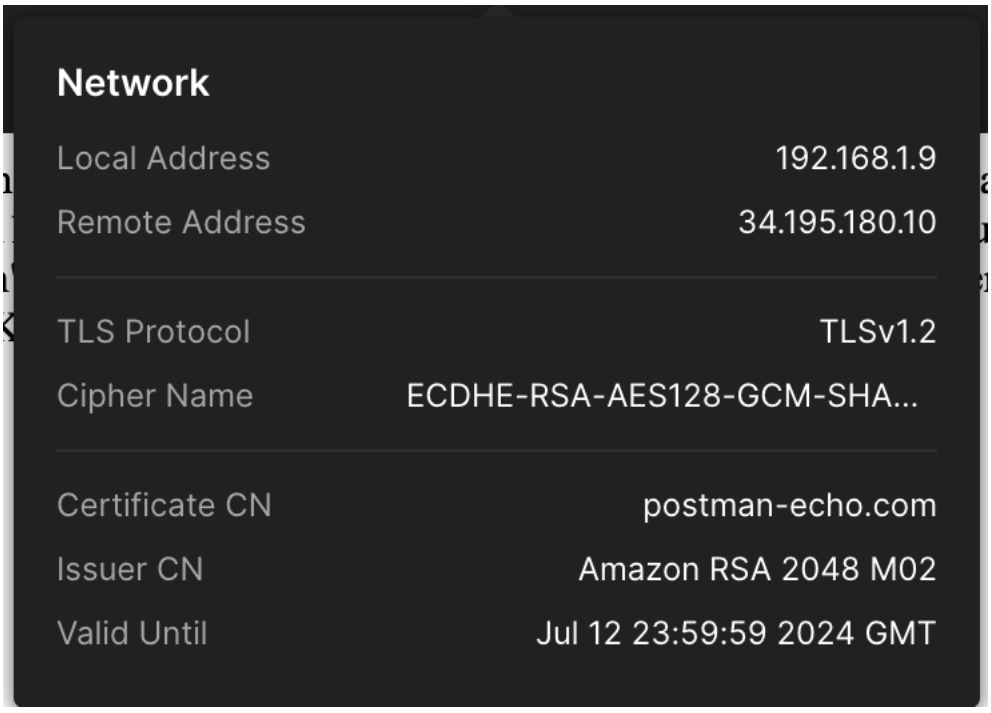

Рис. 2.1.16. Сетевая информация

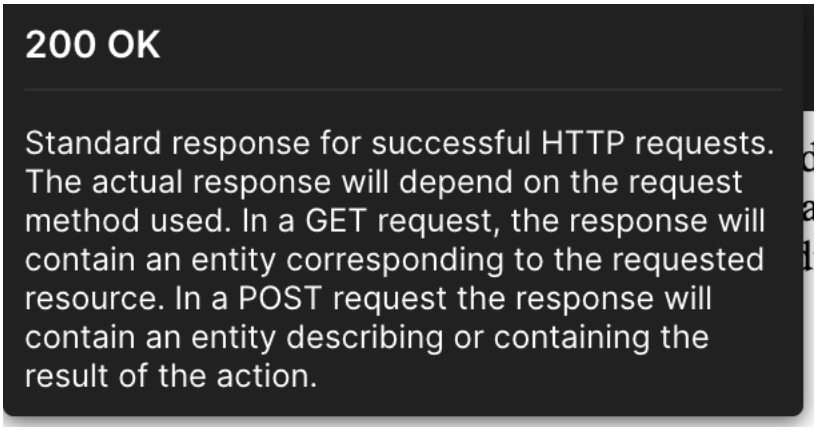

Рис. 2.1.17 . Информация о коде выполения

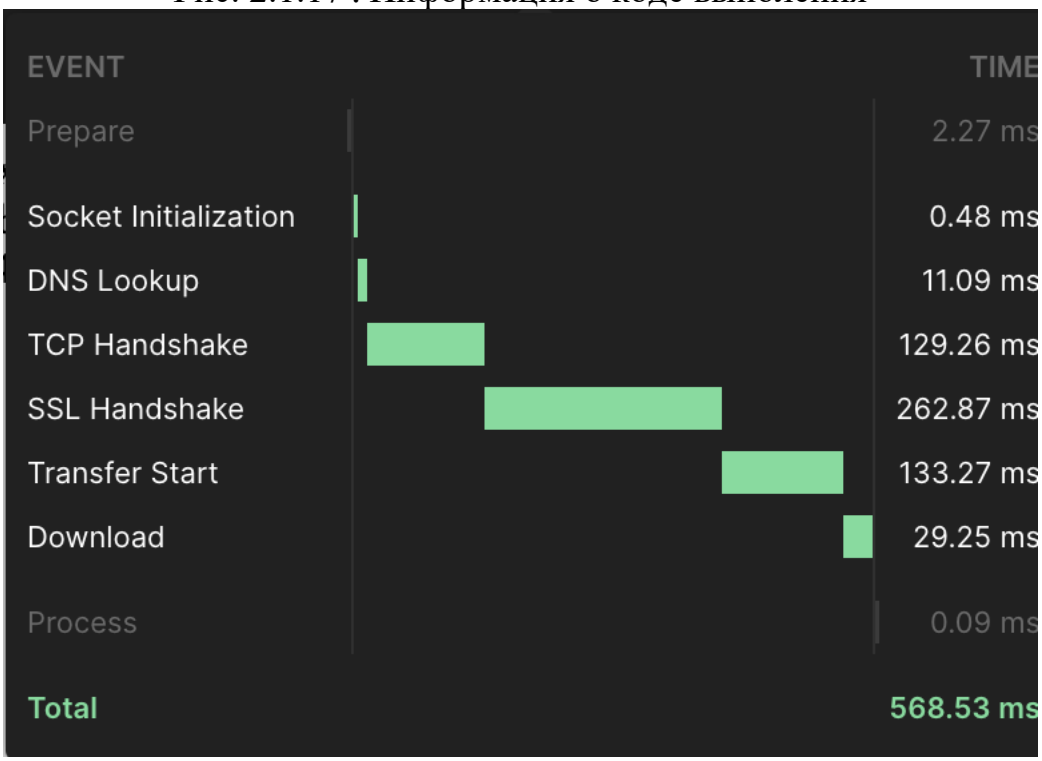

Рис. 2.1.18. Информация о времени выполнения

Рис. 2.1.19. Информация о размере

2.2. ЛАБОРАТОРНАЯ РАБОТА №3

Копируем запрос из пункта 9 лабораторной работы №2 в коллекцию текущей. В теле запроса значение параметра «id» задаем через новую переменную «pet_id». Результат выполнения представлен на рис. 2.2.1.

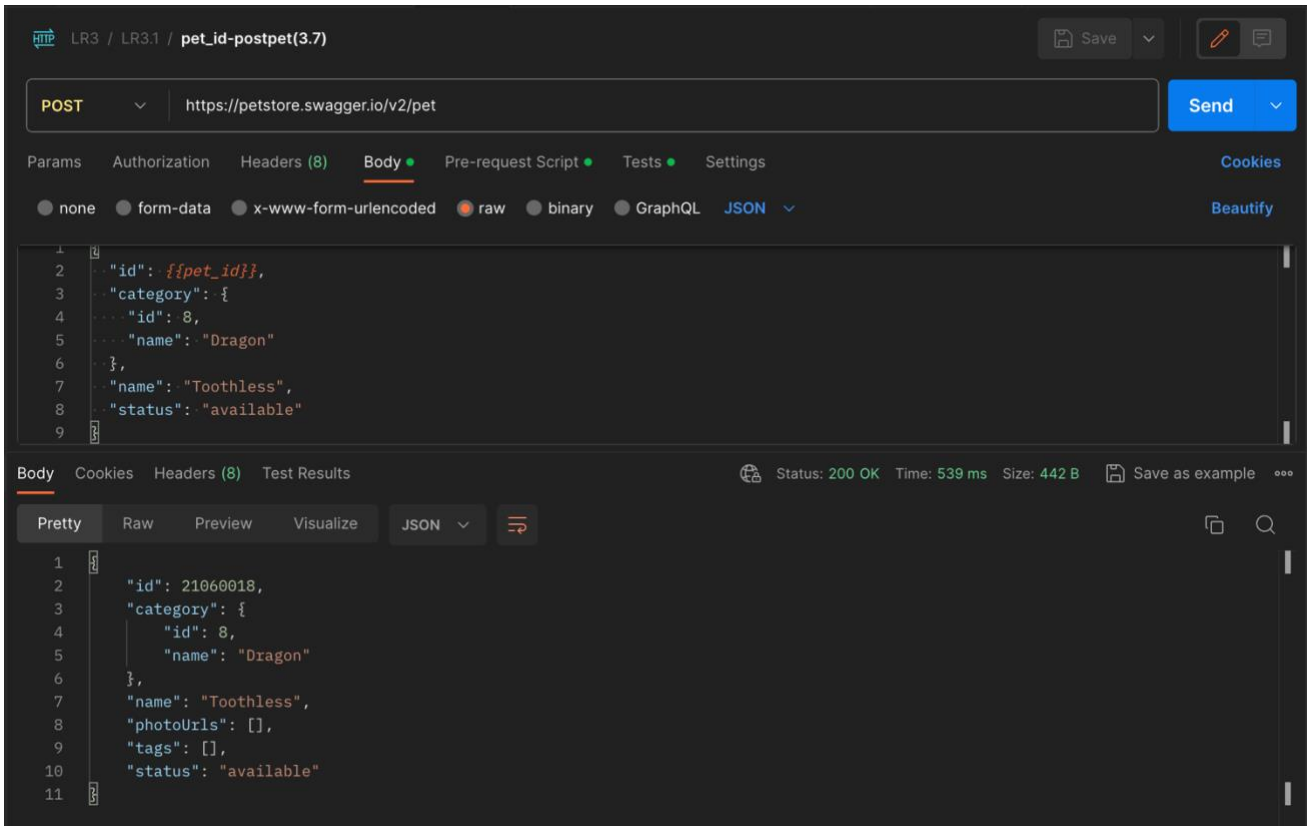

Рис. 2.2.1. Запрос с переменной «pet_id» вместо значения id

Создаем глобальные переменные: «username» со значением = admin и «password со значением = 123. Для переменной «password» задаем тип «secret». Результат выполнения представлен на рис. 2.2.2.

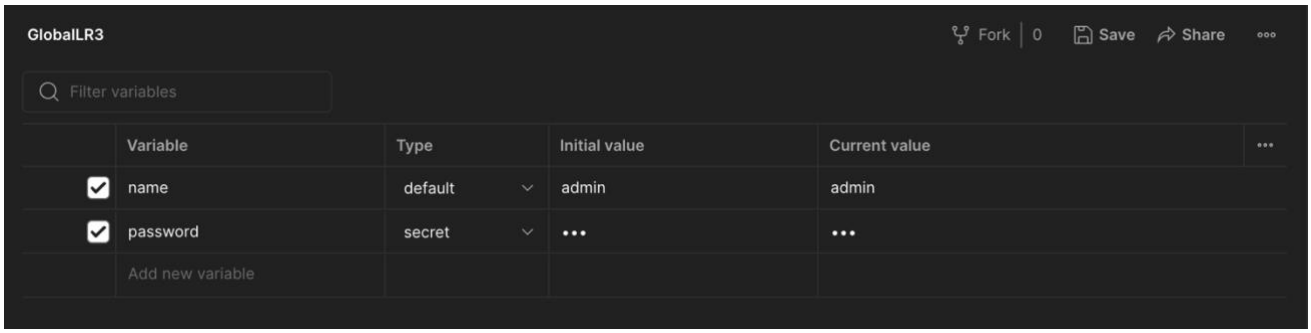

Рис. 2.2.2. Глобальные переменные «username» и «password»

Добавлем в коллекцию «LR-3» копию запроса из пункта 8 лабораторной

работы №2. Вносим изменения в запрос так, чтобы в нем использовались переменные «username» и «password», созданные в предыдущем пункте. Результат выполнения представлен на рис. 2.2.3.

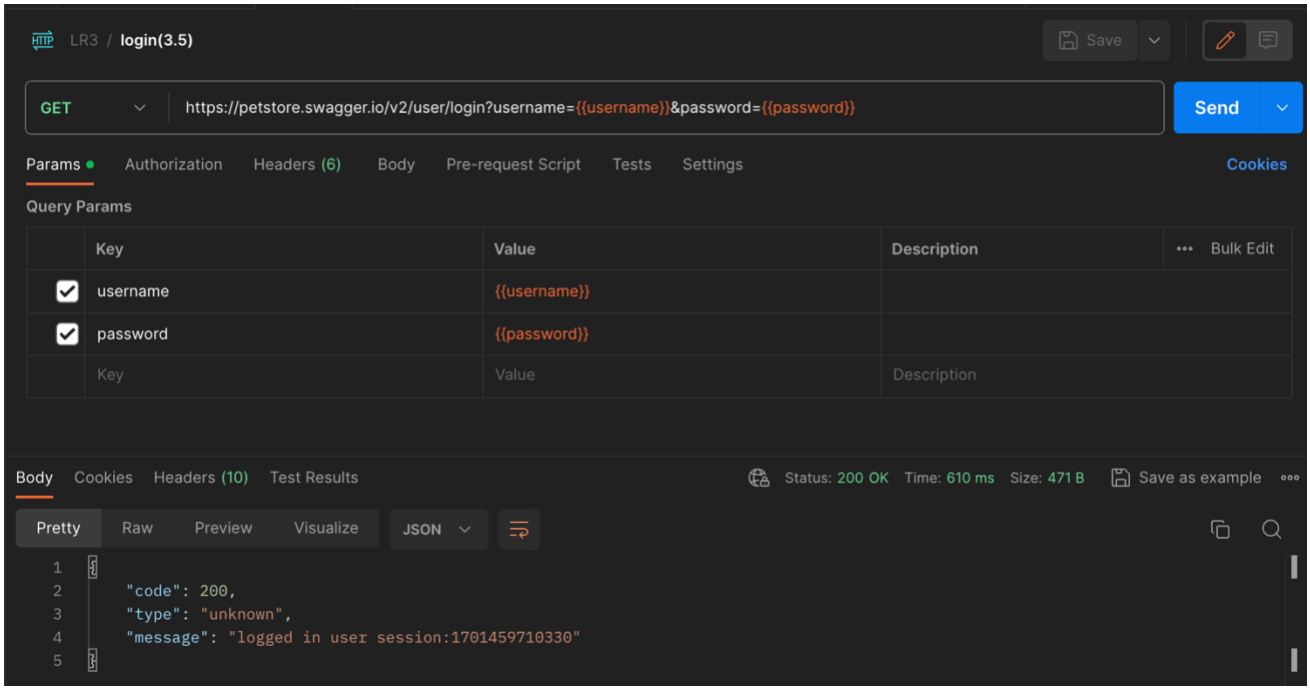

Рис. 2.2.3. Запрос с переменными «username» и «password», созданными в предыдущем пункте

Создаем POST запрос к сервису «postman-echo», параметры которого указали свое имя. В теле данного запроса прописали параметры «Number», «FavoriteColor», «Honorific», «Position» и «Sphere», значения которых задали через соответствующие динамические переменные. Результат выполнения представлен на рис. 2.2.4.

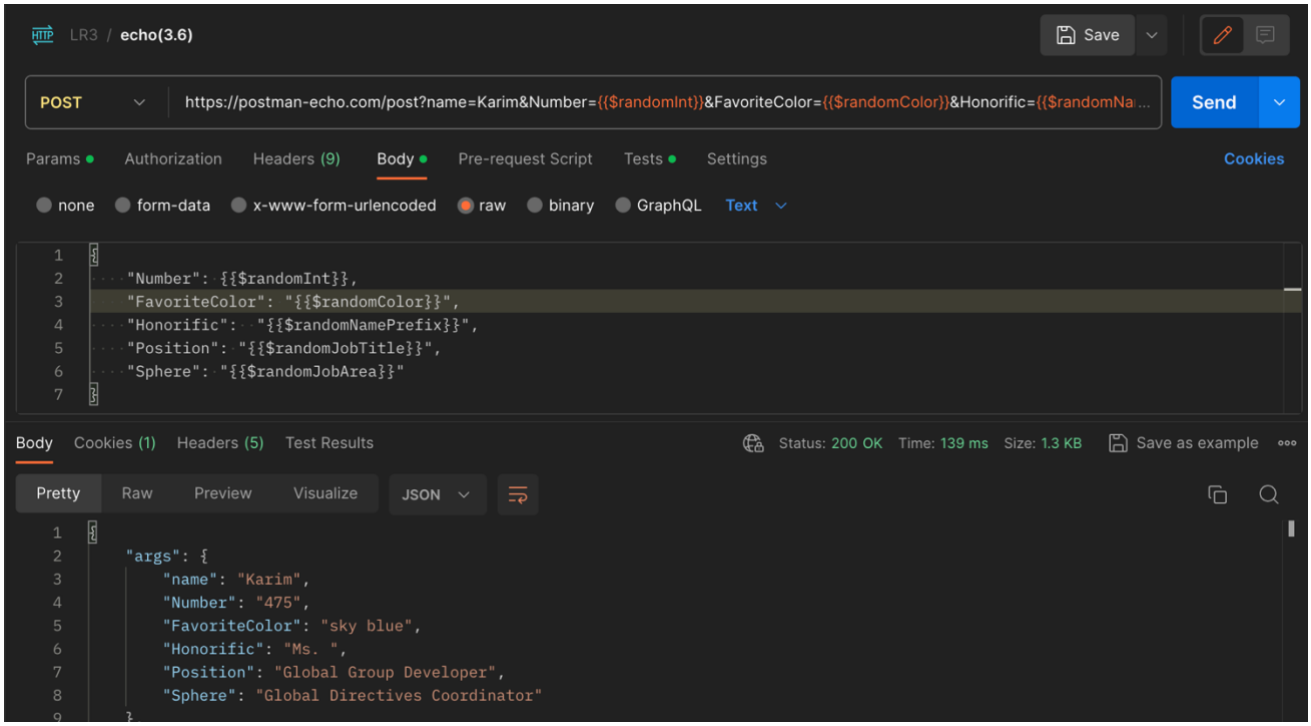

Рис. 2.2.4. POST запрос с динамическими переменными

Создаем папку, перемещаем в нее запрос с именем «pet_id». При помощи команды вывода текста в консоль «(console.log("Тип скрипта – уровень скрипта"))» добавляем скрипты в каждом из блоков: (console.log("pre- req – request")) на уровне запроса, (console.log("pre-req – folder")) на уровне папки и (console.log("pre-req – collection")) на уровне коллекции. Результат выполнения представлен на рис. 2.2.5.

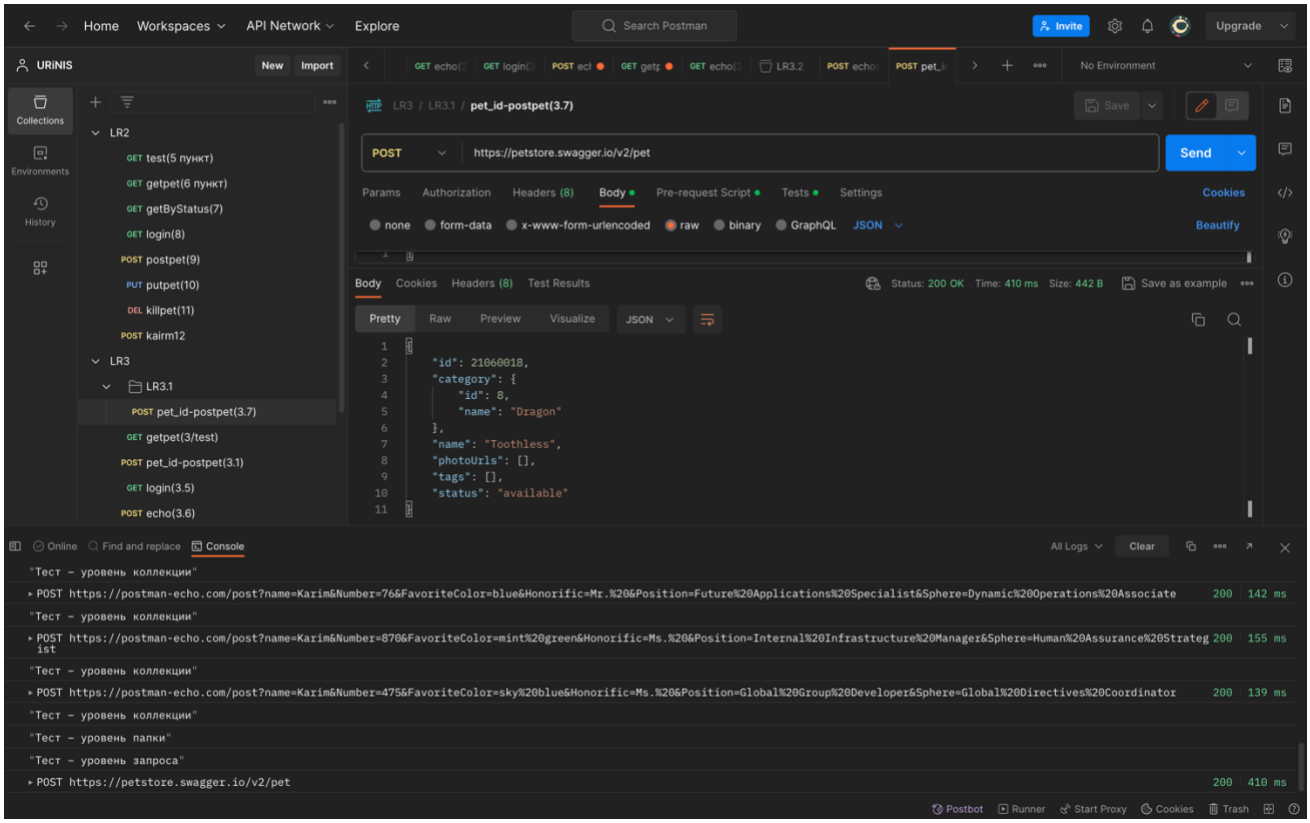

Рис. 2.2.5. Результат вывода консоли после задания скриптов на каждом уровне

Создаем переменную уровня коллекции с именем «param» и значением «default». На вкладке «Tests» POST запроса вставляем код «pm.collectionVariables.set("param", JSON.parse(responseBody).args.name)». Не выполняя запрос, добавляем в коллекцию GET запрос с эндпоинтом «https://postman-echo.com/get?param={{param}}», после чего на вкладке «Pre-request script» этого же запроса добавляем код, который возьмет значение этой переменной, заданной в скрипте POST запроса, и выведет его после преобразования. Результат выполнения представлен на рис. 2.2.6.

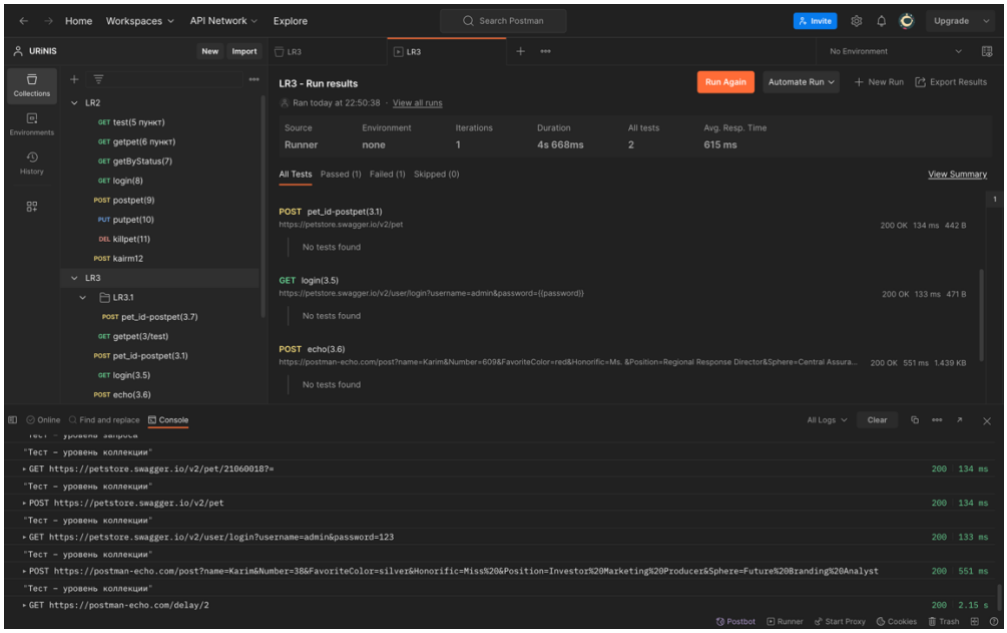

Рис. 2.2.6. Консоль после запуска коллекции с запросами POST и GET добавляем переменную уровня коллекции «delay», задаем ей значение.

После чего создаем GET запрос с эндпоинтом «https://postman-echo.com/delay/:delay», в качестве path параметра которого использовали созданную переменную. Результат выполнения представлен на рис. 2.2.7.

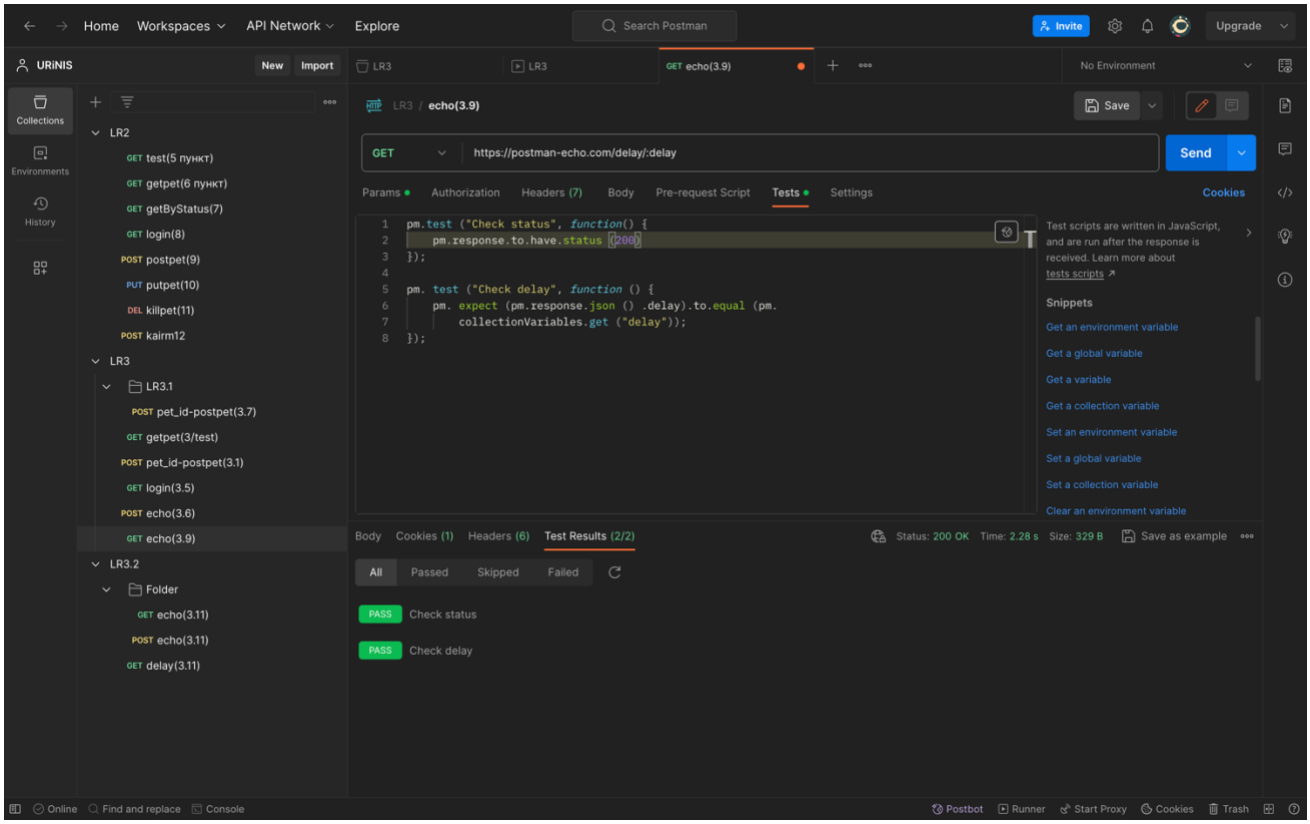

Рис. 2.2.7. GET запрос с переменной «delay»

Изменяем значение в проверке код ответа на 201 и снова выполняем запрос. Результат выполнения представлен на рис.2.2.8.

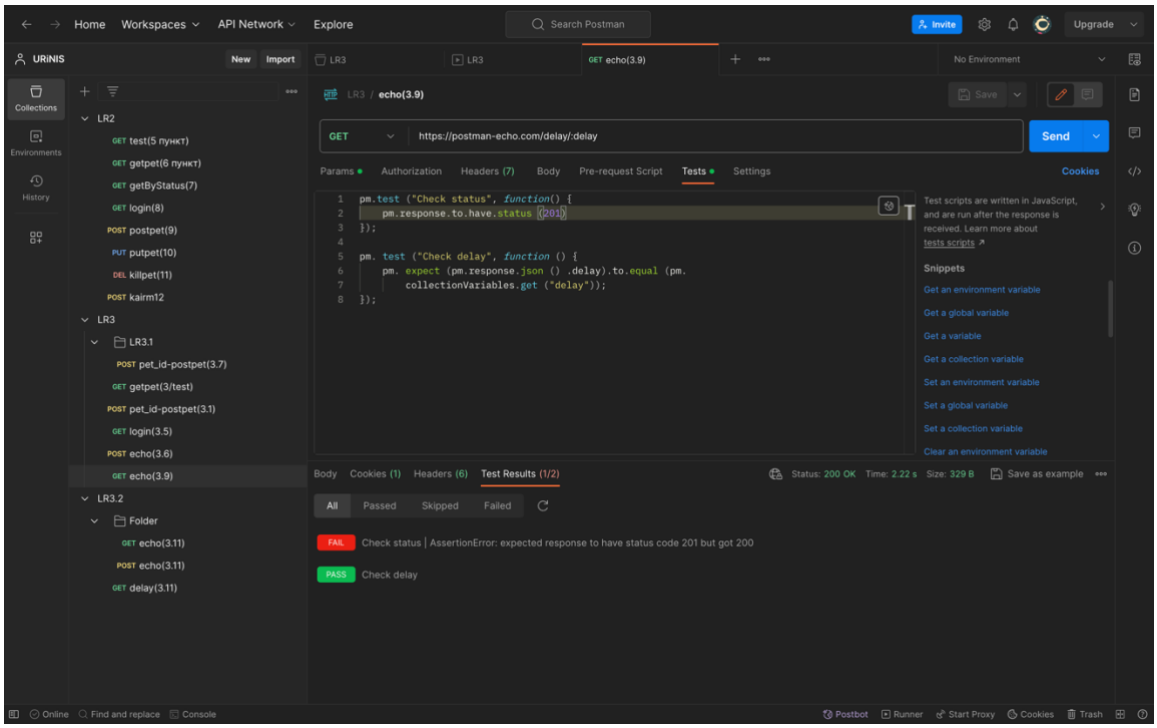

Рис. 2.2.8. GET запрос с переменной «delay» после замены значения код ответа на 201

Создаем дополнительную коллекцию с названием «LR-3.2», в описании которой прописываем текущую дату и собственное ФИО. В данной коллекции создаем папку, в которую добавляем два запроса: GET «https://postman-echo.com/get?name={{name}}» и POST «https://postman-echo.com/post?name={{name}}». Переменные «name» со значением своего имени и «delay» равную 2 создаем на уровне коллекции. После чего вне папки создаем еще один GET запрос «https://postman-echo.com/delay/:delay», в котором тоже задаем параметр «delay». Результат выполнения представлен на рис. 2.2.9.

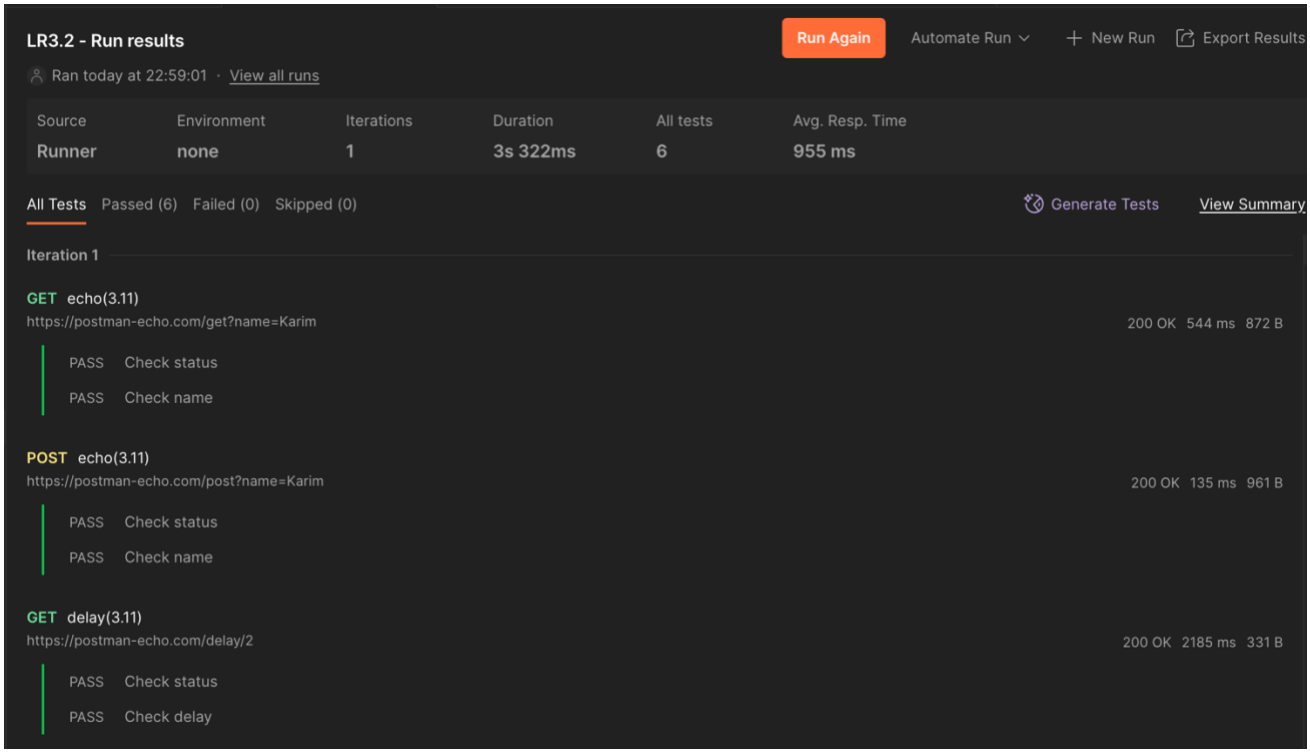

Рис. 2.2.9. Результат запуска коллекции LR-3.2 со всеми созданными в ней запросами

Значение Status проверяется на всех трёх уровнях, так как скрипт проверки «Status» находится на уровне коллекций. В запросах, находящихся в папке проверяется значения «Name», так как скрипт проверки «Name» находится на уровне папки. Значение «Delay» проверяется в последнем запросе, находящемся вне папки, так как скрипт проверки «Delay» был задан на уровне этого запроса, поэтому в нем не проверяется значение «Name».

ЗАКЛЮЧЕНИЕ

В данных лабораторных работах были изучены способы создания рабочего пространства, коллекций, составления и отправки запросов с различными методами в приложении «Postman». Также были изучены возможности приложения «Postman», такие как создание папок, переменных различного уровня и работа со скриптами.