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-22-06 **Мустафаева П.М.**

подпись

Москва, 2024 г.

ОГЛАВЛЕНИЕ

ЦЕЛЬ РАБОТЫ

Изучить возможности приложения «Postman», такие как создание рабочего пространства, коллекции, составлению и отправки запросов различных методов, создание папок, переменных, работа со скриптами и ответами от сервера.

ПРАКТИЧЕСКАЯ ЧАСТЬ

1.1. ЛАБОРАТОРНАЯ РАБОТА №2

Перед тем, как выполнить лабораторную работу было скачано приложение «Postman».

Регистрируем аккаунт в «Postman» и авторизуемся в нем (рис.1.1.1).

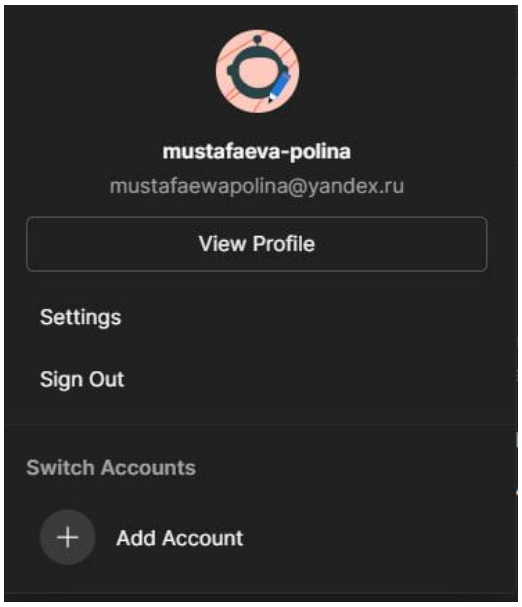

Рис. 1.1.1. Аккаунт в «Postman»

Создание рабочего пространства с названием дисциплины «URiNIS» и описанием (номер группы и подгруппы), а также создание коллекций для лабораторных работ представлены на рис. 1.1.2-1.1.3.

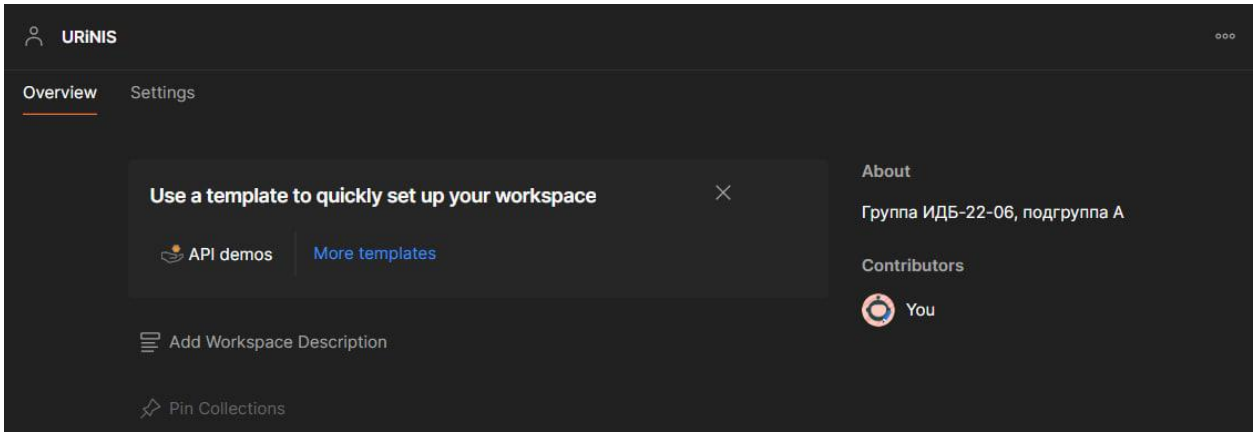

Рис. 1.1.2. Рабочее пространство в «Postman»

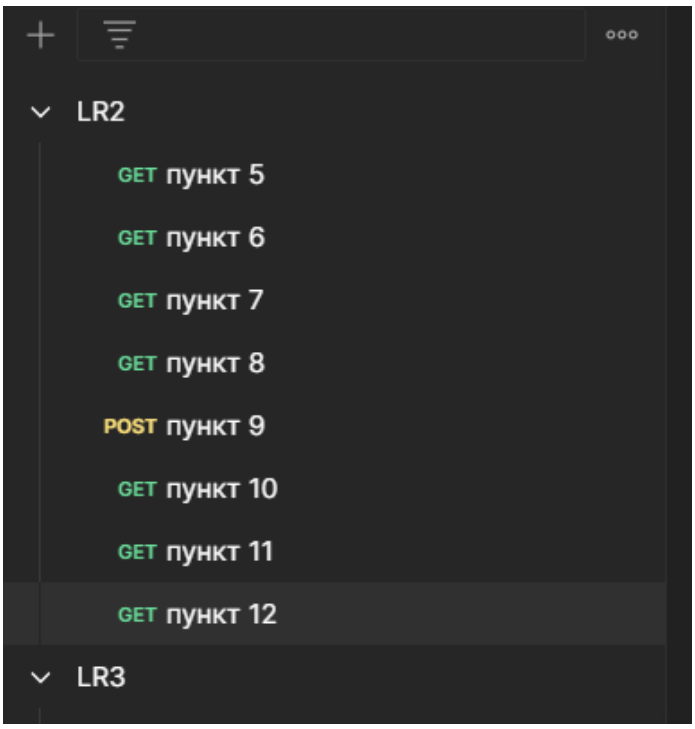

Рис. 1.1.3. Коллекции для лабораторных работ

Отправка GET запроса к сайту «API Swagger Petstore». Результат выполнения представлен на рис. 1.1.4.

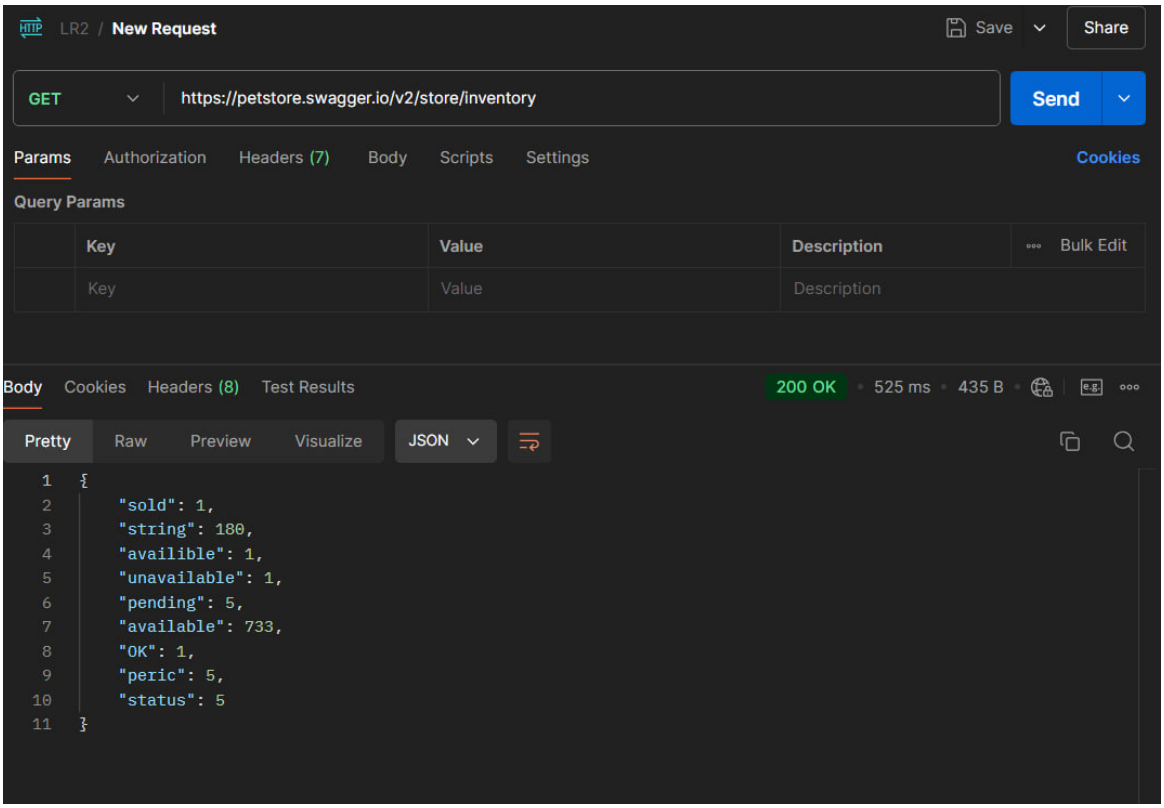

Рис. 1.1.4. GET запрос к сайту «API Swagger Petstore»

Отправка GET запроса с параметром для получения информации о домашнем питомце по его идентификатору в системе. С параметром «id = 150» питомец найден не был. Результат выполнения запроса представлен на рис. 1.1.5.

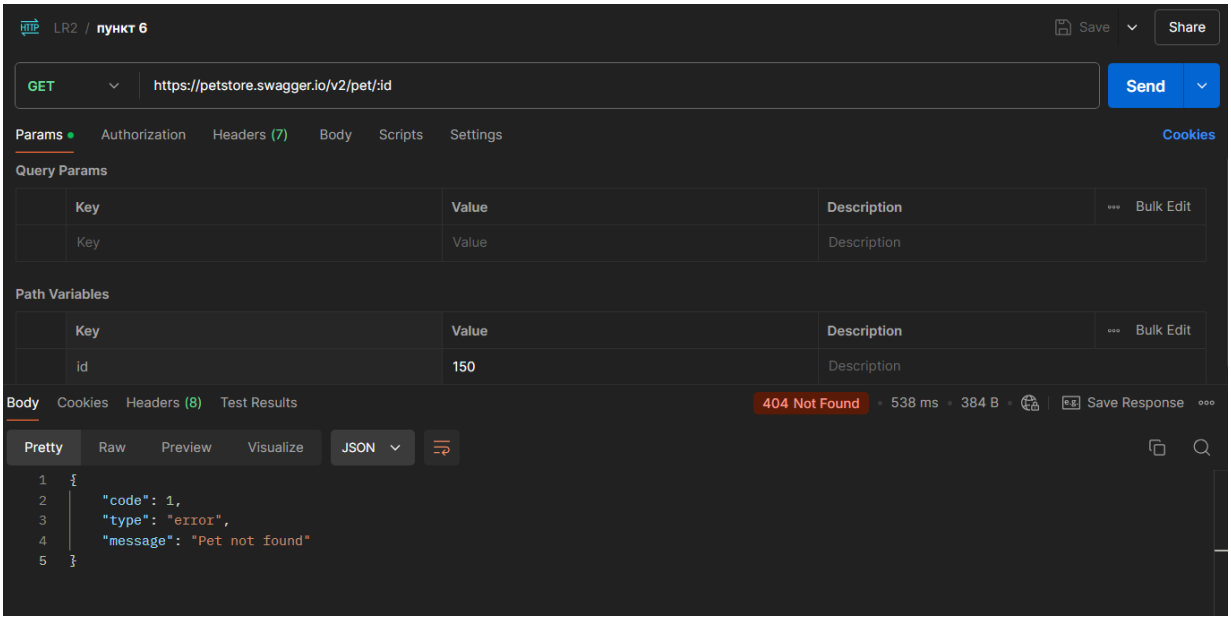

Рис. 1.1.5. GET запрос с параметром для получения информации о домашнем питомце по его id

Отправка GET запроса с query параметром, который позволяет получить информацию о домашних питомцах, отфильтрованных по одному из статусов. В данном примере был взят статус «pending». Результат выполнения запроса представлен на рис. 1.1.6.

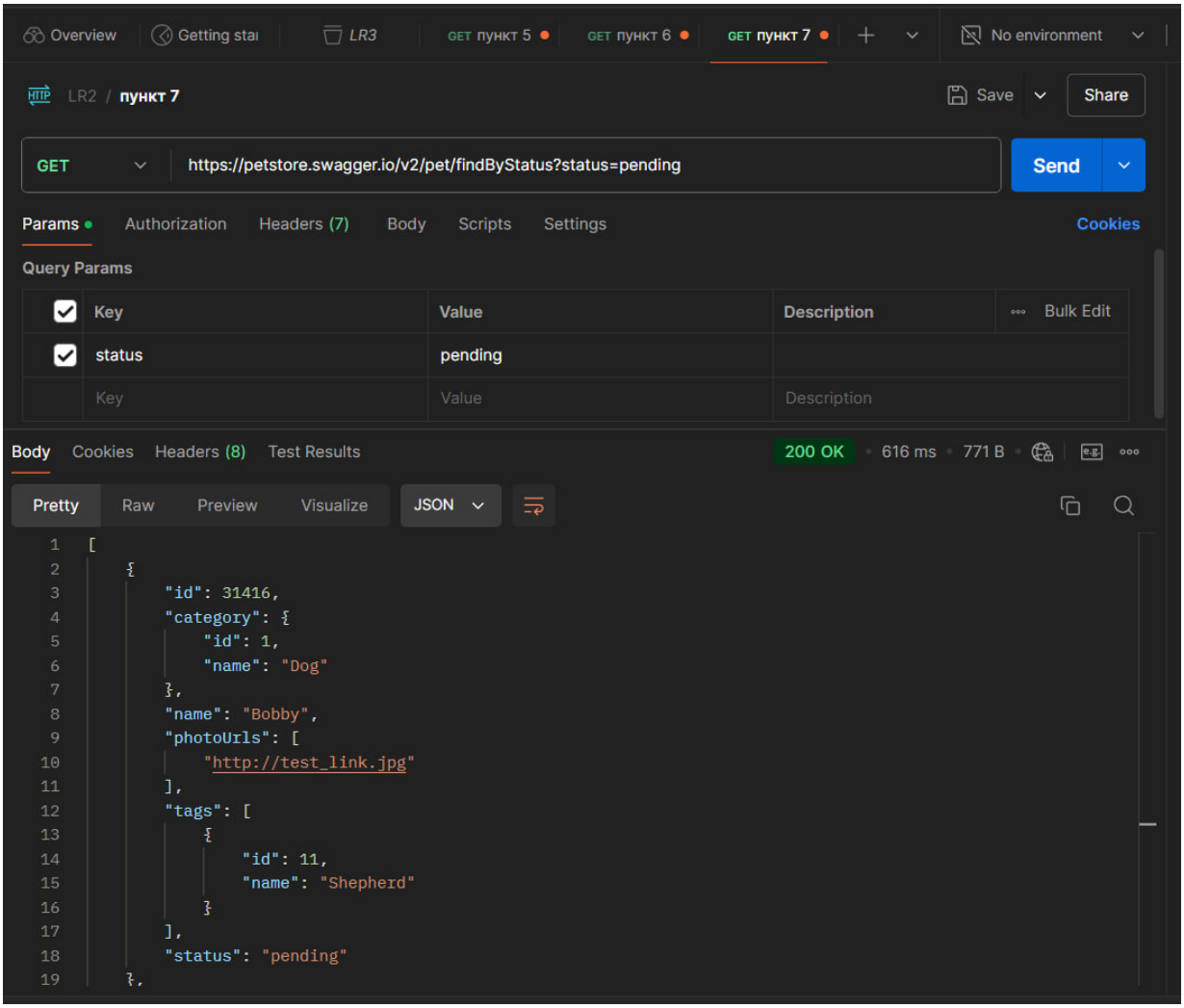

Рис. 1.1.6. GET запрос с query параметром

Отправка GET запроса с несколькими параметрами (имя пользователя и пароль). Для данного задания были взяты «username = Polina» и «password = 128812». Также при выполнении запроса была просмотрена вкладка «Bulk Edit» на которой были выведены значения имени и пароля в текстовом формате, которые можно было отредактировать. Результаты выполнения запроса представлены на рис. 1.1.7-1.1.8.

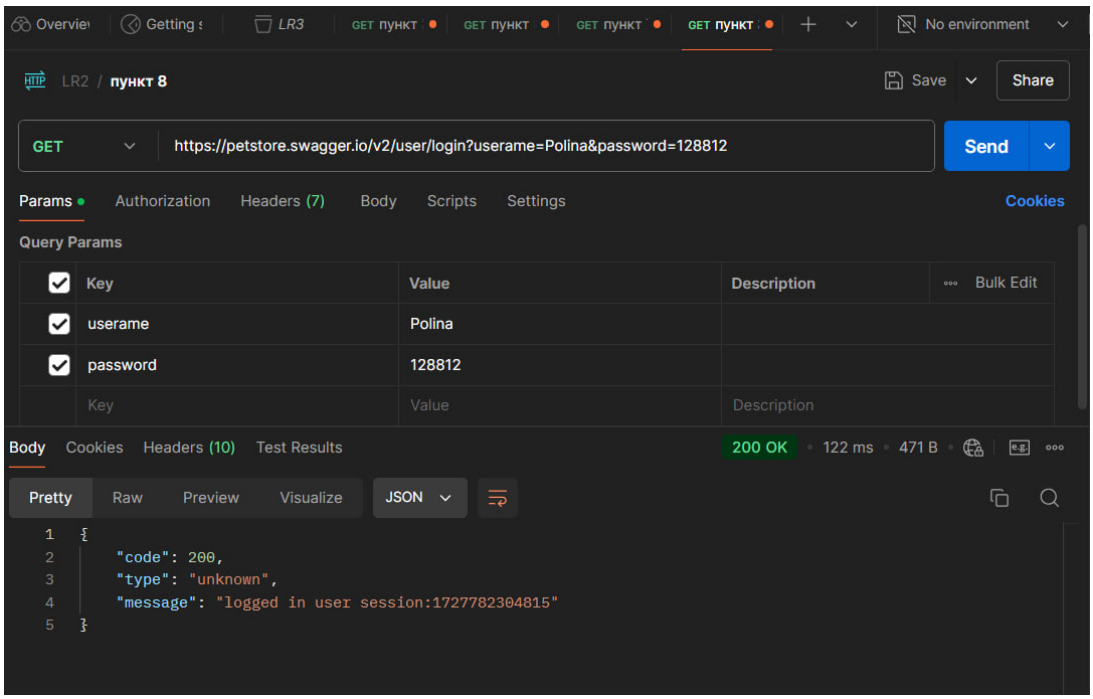

Рис. 1.1.7. GET запрос с несколькими параметрами

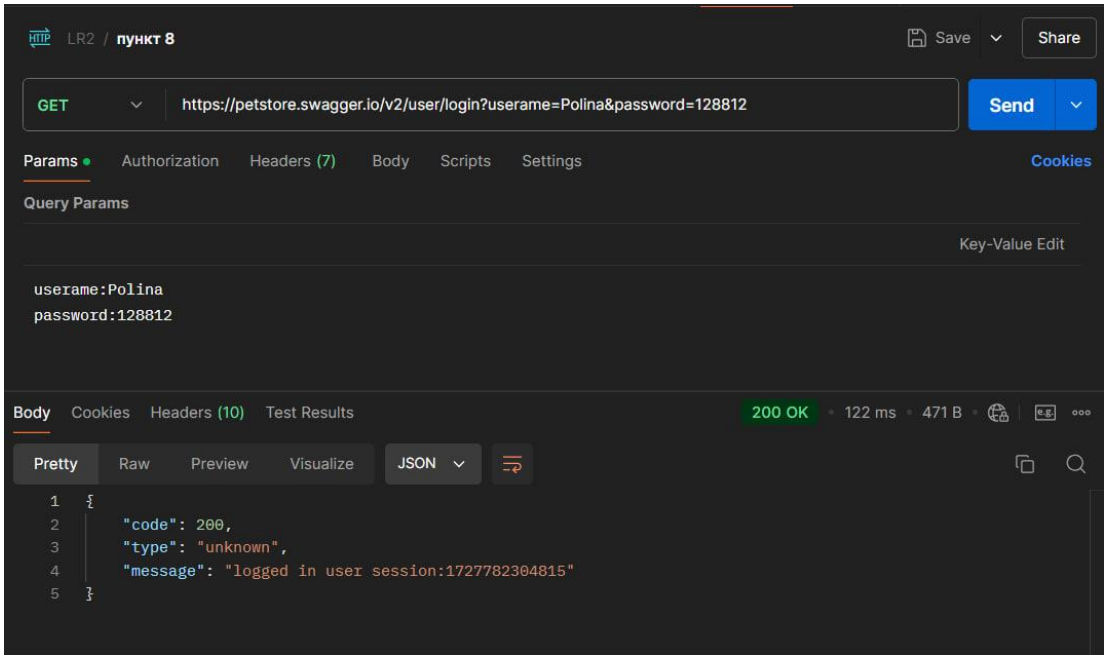

Рис. 1.1.8. GET запрос с несколькими параметрами на вкладке «Bulk Edit»

Создание POST запроса для создания записи о новом домашнем животном. Для данного запроса был выбран id соответствующий номеру группы «id=6», а также вид животного «name = dog» и его имя «name = Sunny». Результат выполнения запроса представлен на рис. 1.1.9.

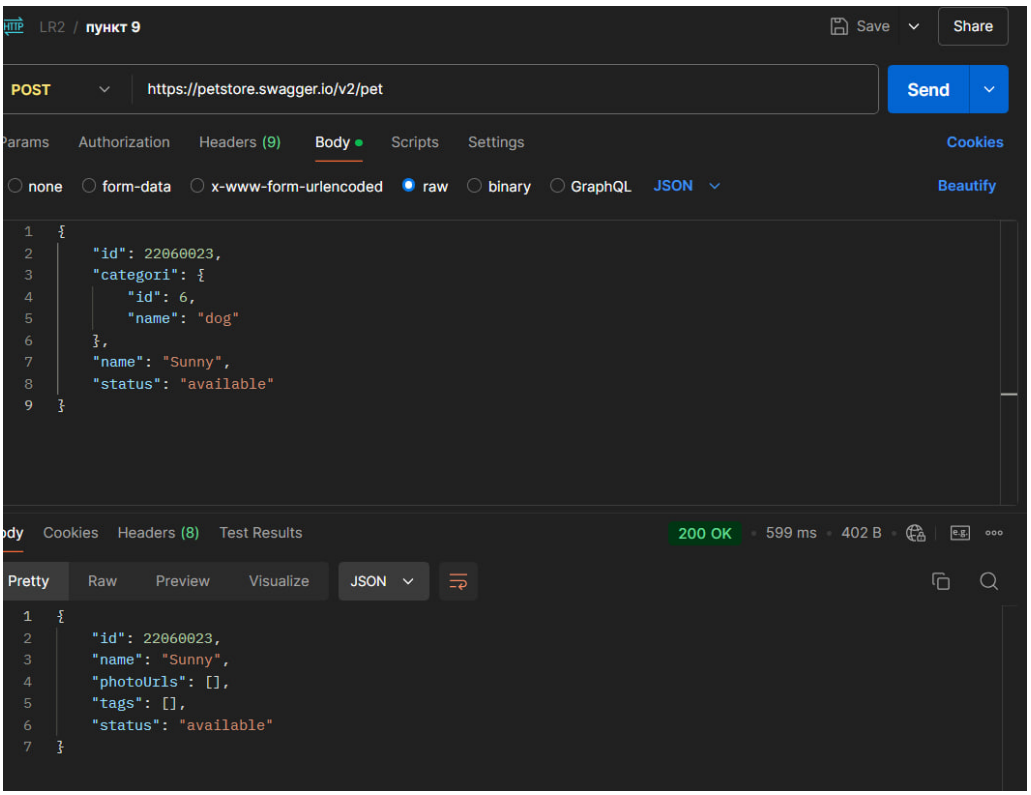

Рис. 1.1.9. POST запрос для создания записи о новом домашнем животном

Отправка PUT запроса для обновления данных о новом домашнем животном. Также было добавлено фото животного. Результат выполнения представлен на рис. 1.1.10-1.1.11.

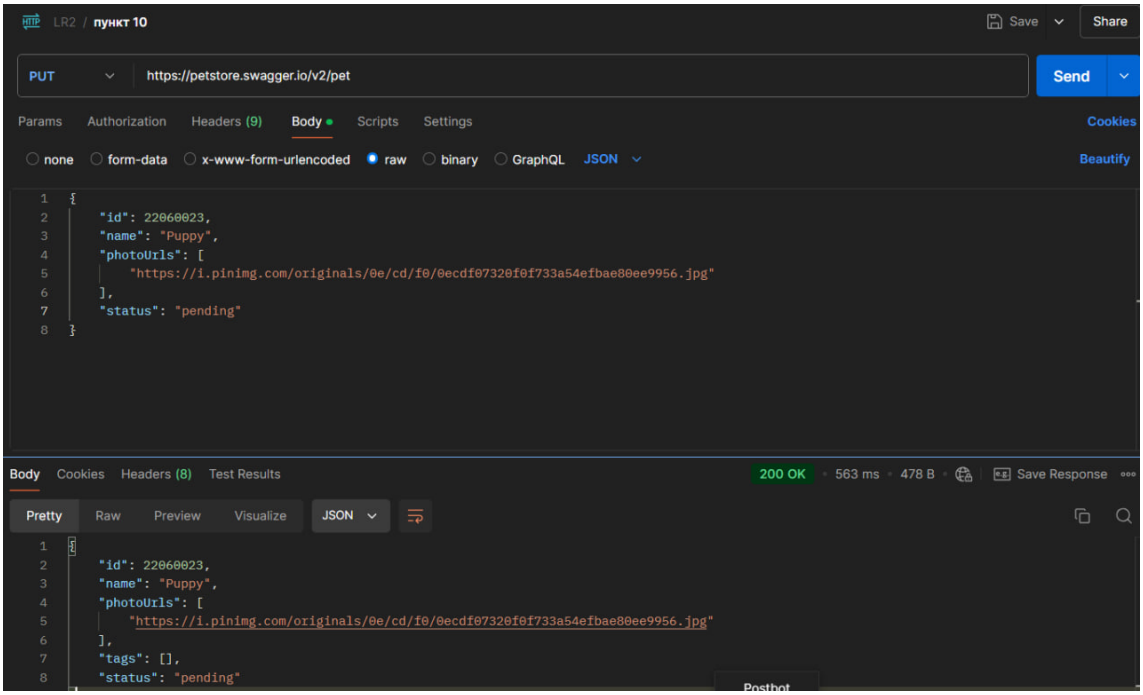

Рис. 1.1.10. PUT запрос на обновление данных о новом домашнем животном

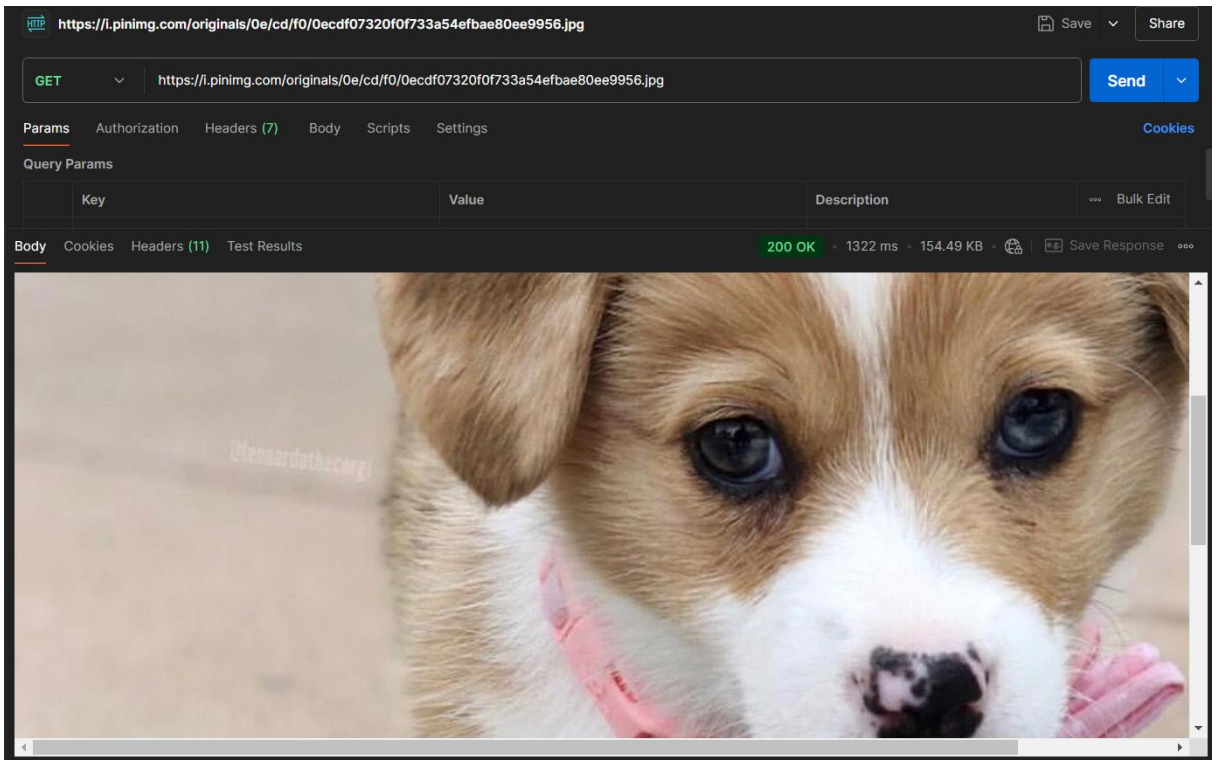

Рис. 1.1.11. GET запрос на открытие фотографии животного

Создание DELETE запроса для удаления созданной записи о новом домашнем животном. Результат выполнения запроса представлен на рис. 1.1.12-1.1.13.

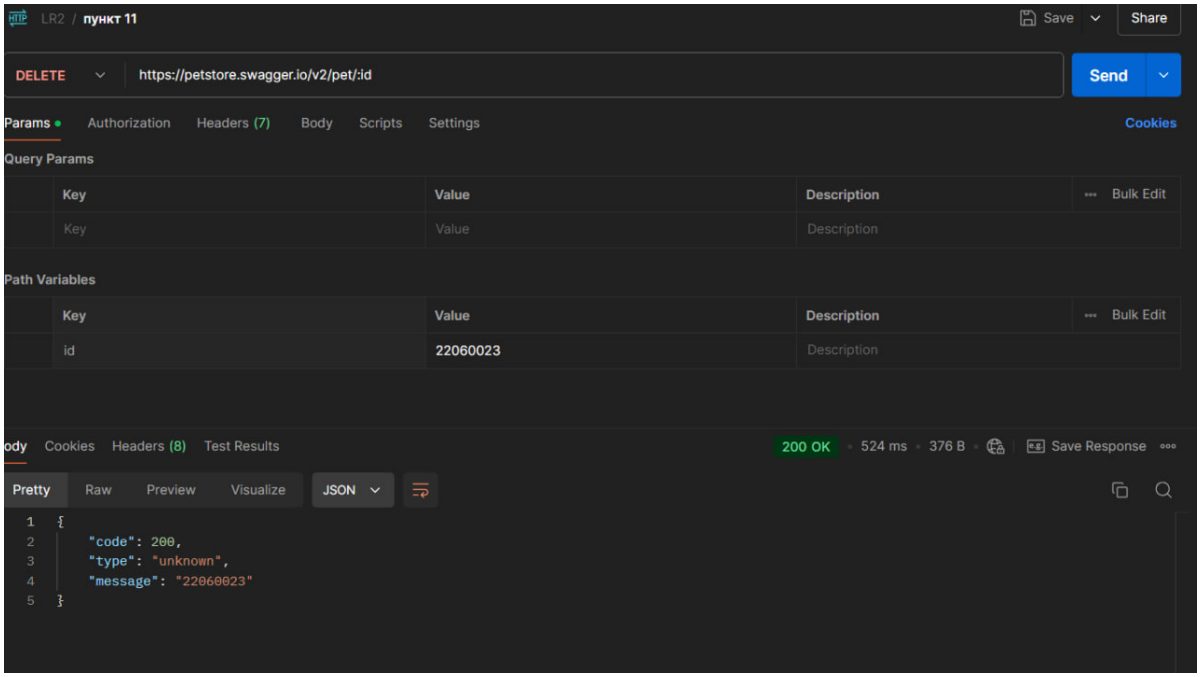

Рис. 1.1.12. DELETE запрос для удаления записи о новом домашнем животном

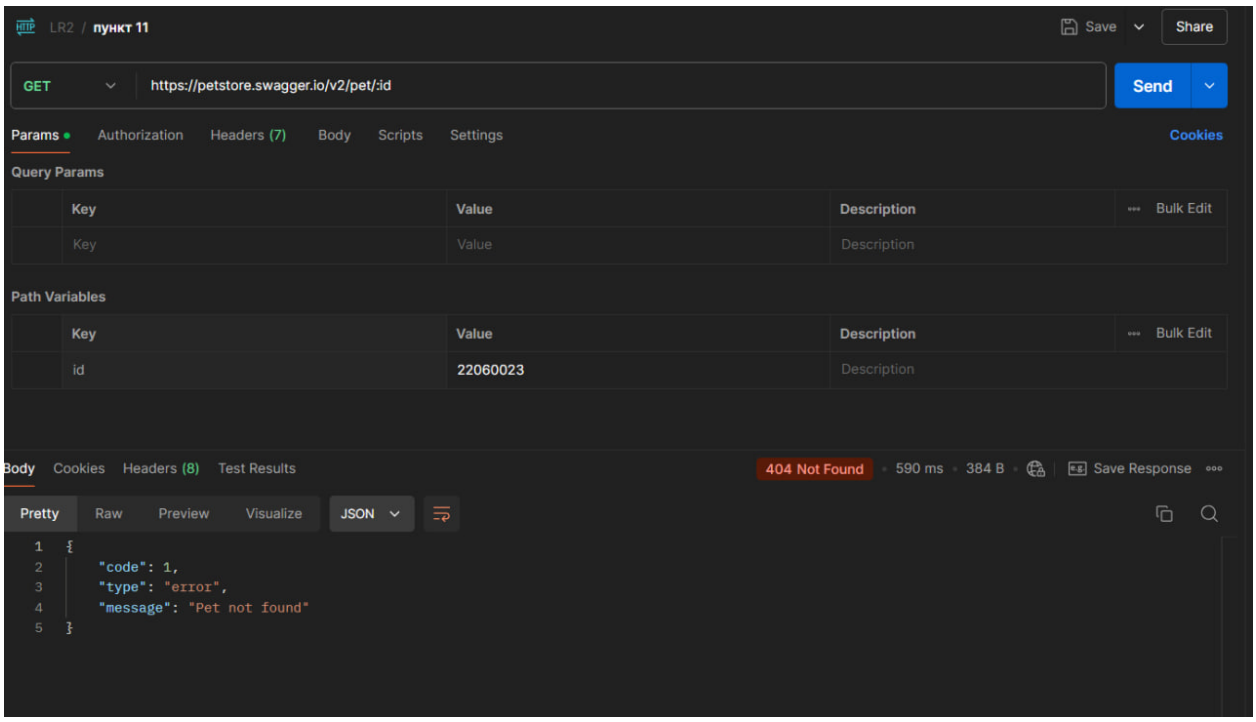

Рис. 1.1.13. GET запрос для проверки удаления записи о новом домашнем животном

Отправка POST запроса с эдпоинтом «https://postman-echo.com/post?name=Polina». Результат выполнения запроса в трех представлениях (Pretty, Raw, Prewiew) представлен на рис. 1.1.14-1.1.16.

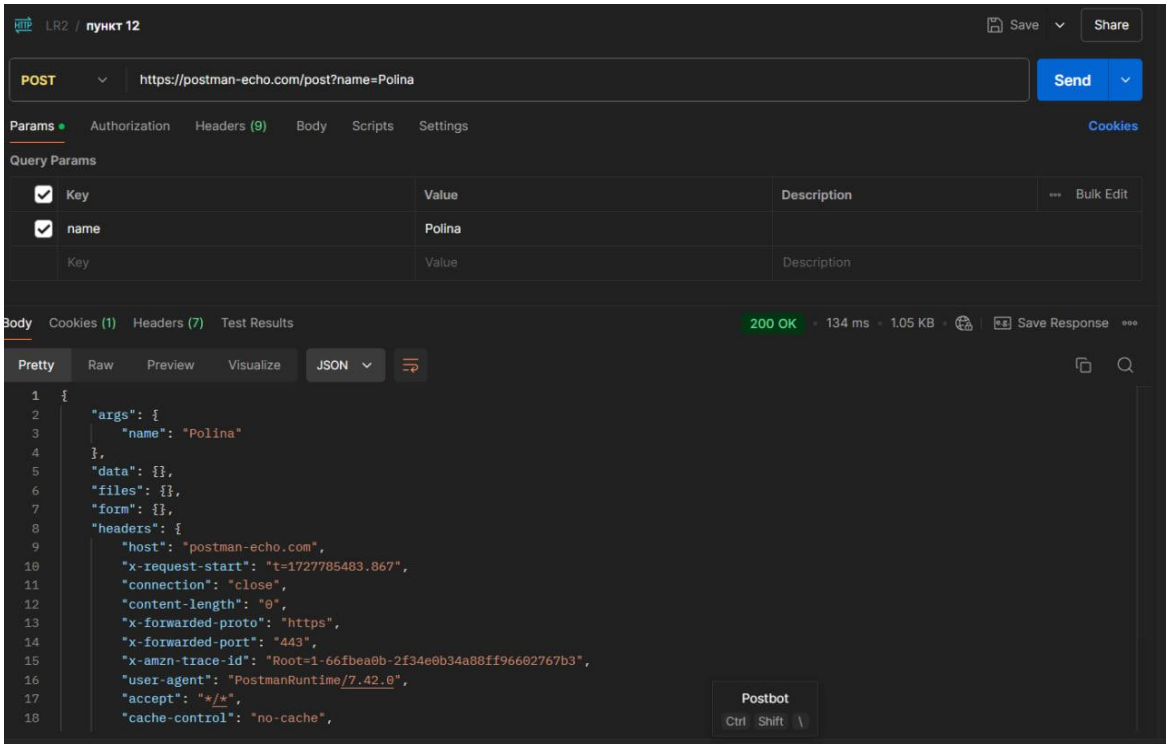

Рис. 1.1.14. POST запрос в представлении Pretty

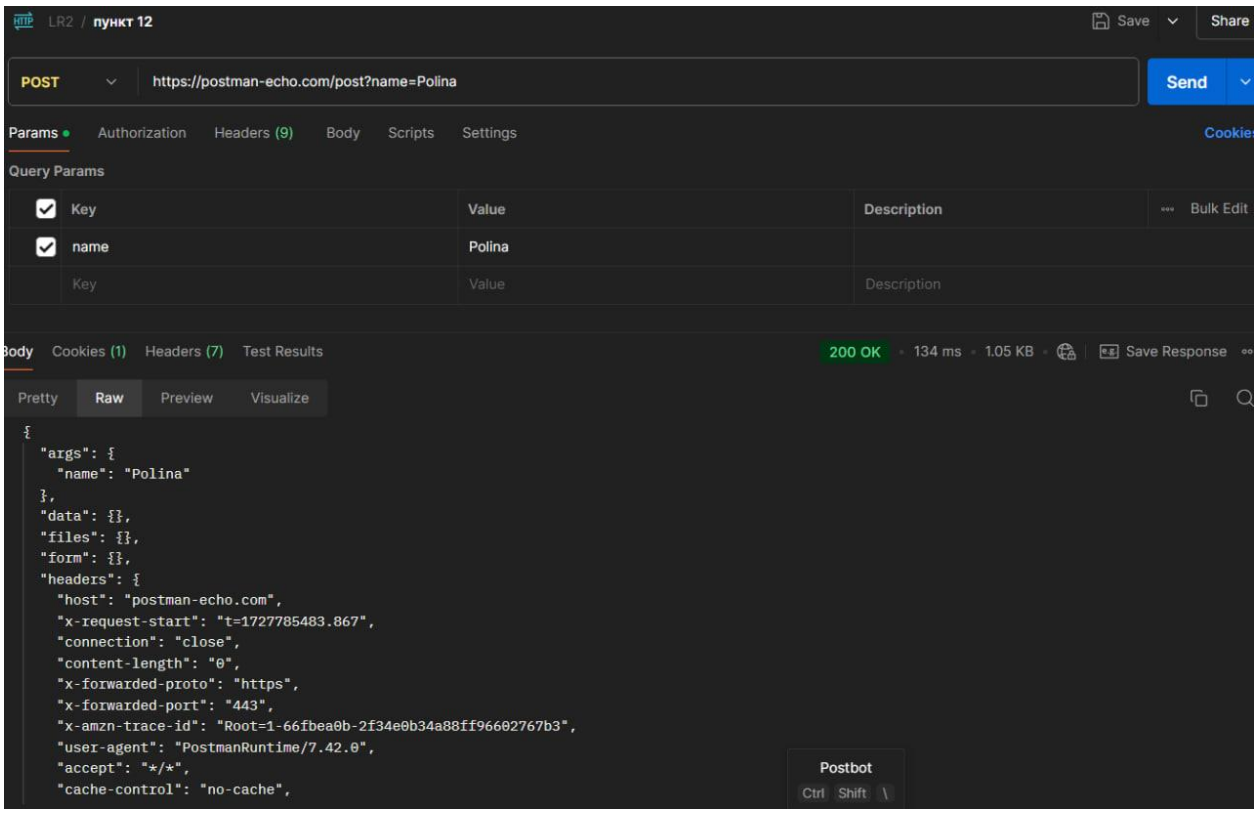

Рис. 1.1.15. POST запрос в представлении Raw

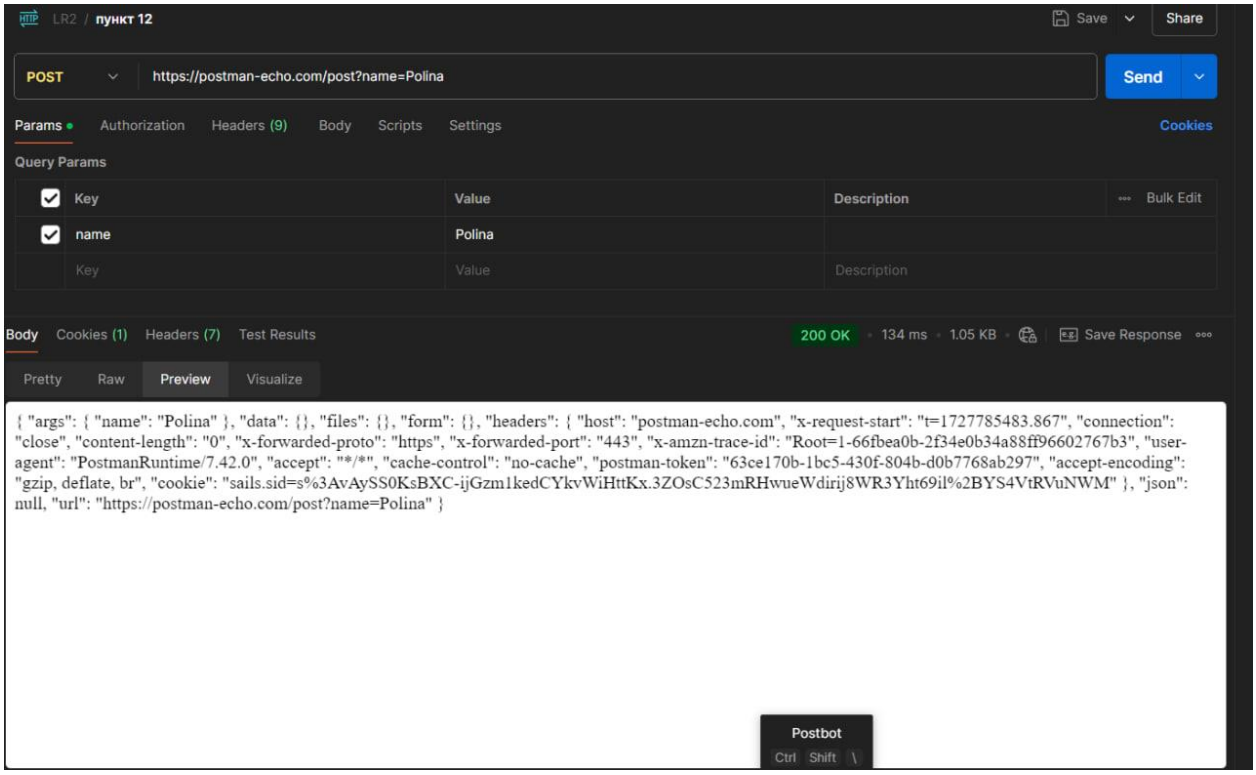

Рис. 1.1.16. POST запрос в представлении Prewiew

В запросе из предыдущего задания перейти на вкладку Headers. Результат выполнения задания представлен на рис. 1.1.17.

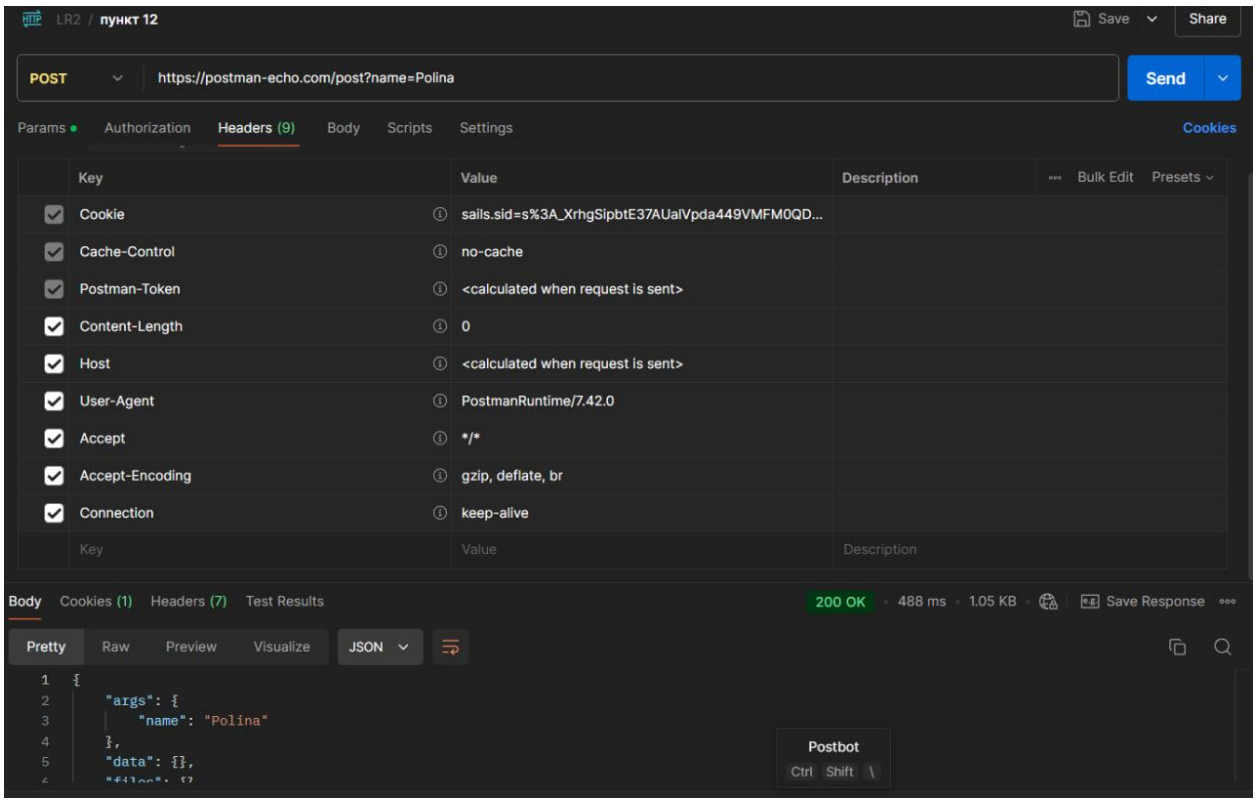

Рис. 1.1.17. Вкладка «Headers»

Дополнительная информация о предыдущем запросе (сетевая информация, код, время и размер ответа) представлена на рис. 1.1.18-1.1.21.

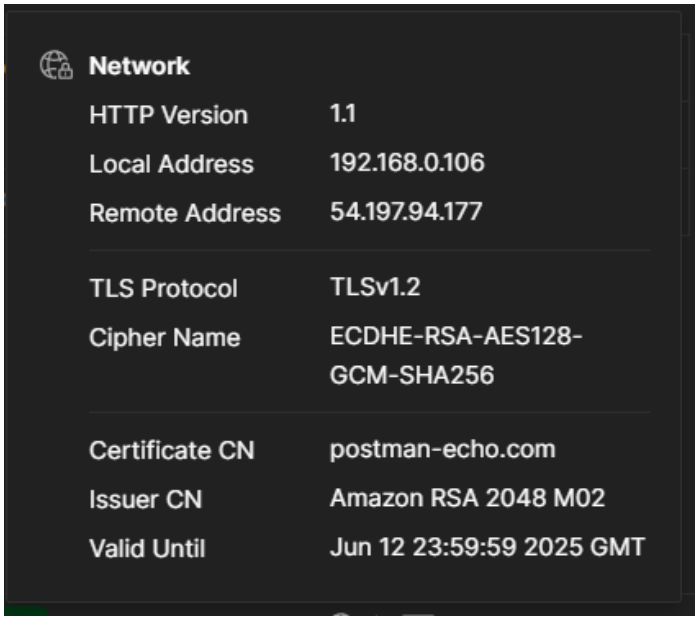

Рис. 1.1.18. Сетевая информация

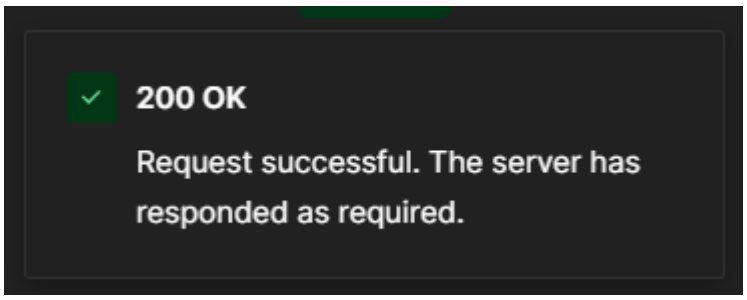

Рис. 1.1.19. Информация о коде выполнения

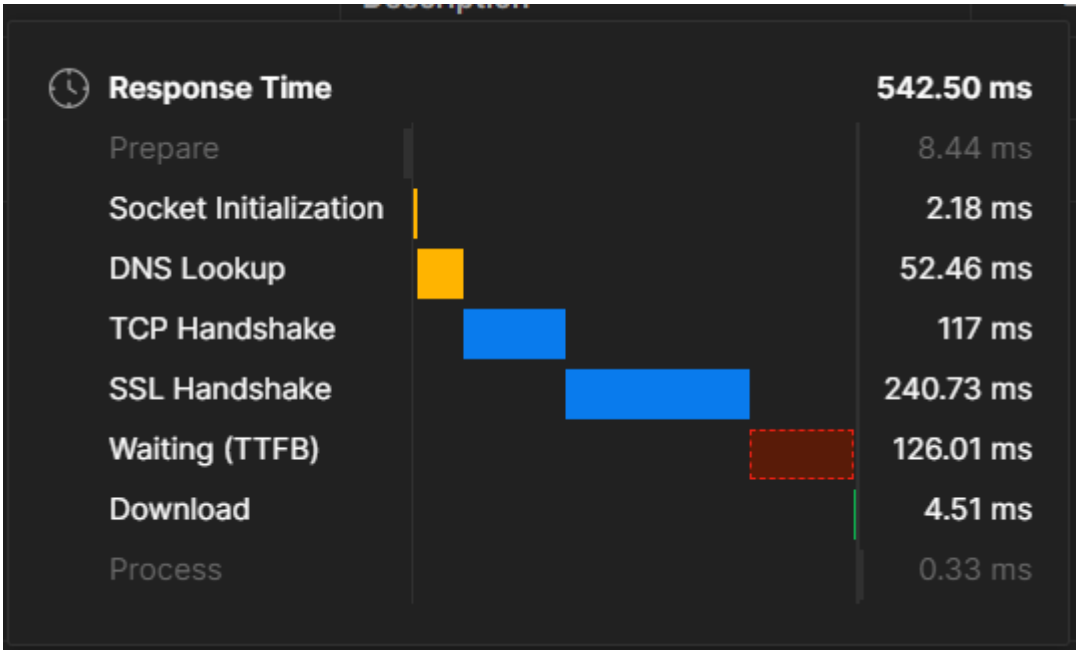

Рис. 1.1.20. Информация о времени выполнения

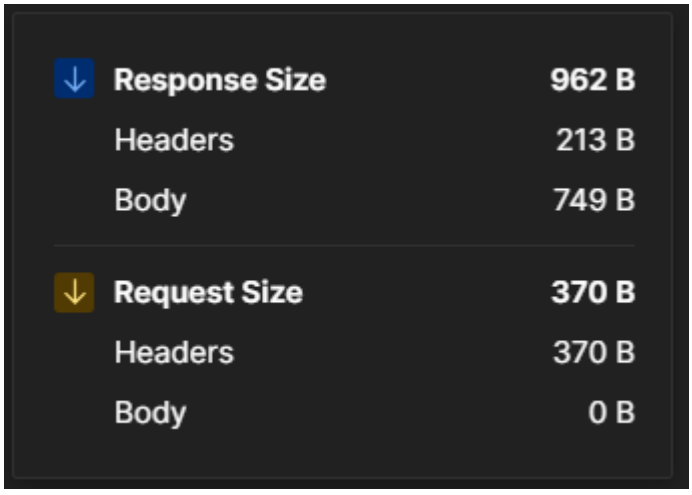

Рис. 1.1.21. Информация о размере

1.2. ЛАБОРАТОРНАЯ РАБОТА №3

Перед выполнением заданий была создана коллекция с названием «LR3».

Добавление в созданную коллекцию запрос с именем «pet_id», который будет получать информацию о питомце по его идентификатору в системе (значение данного path-параметра задайте таким же, как в 9 пункте ЛР № 2). Затем заменена значения path-параметра «id» на переменную «pet_id» (область видимости -глобальная). Результат выполнения запроса представлен на рис. 1.2.1.

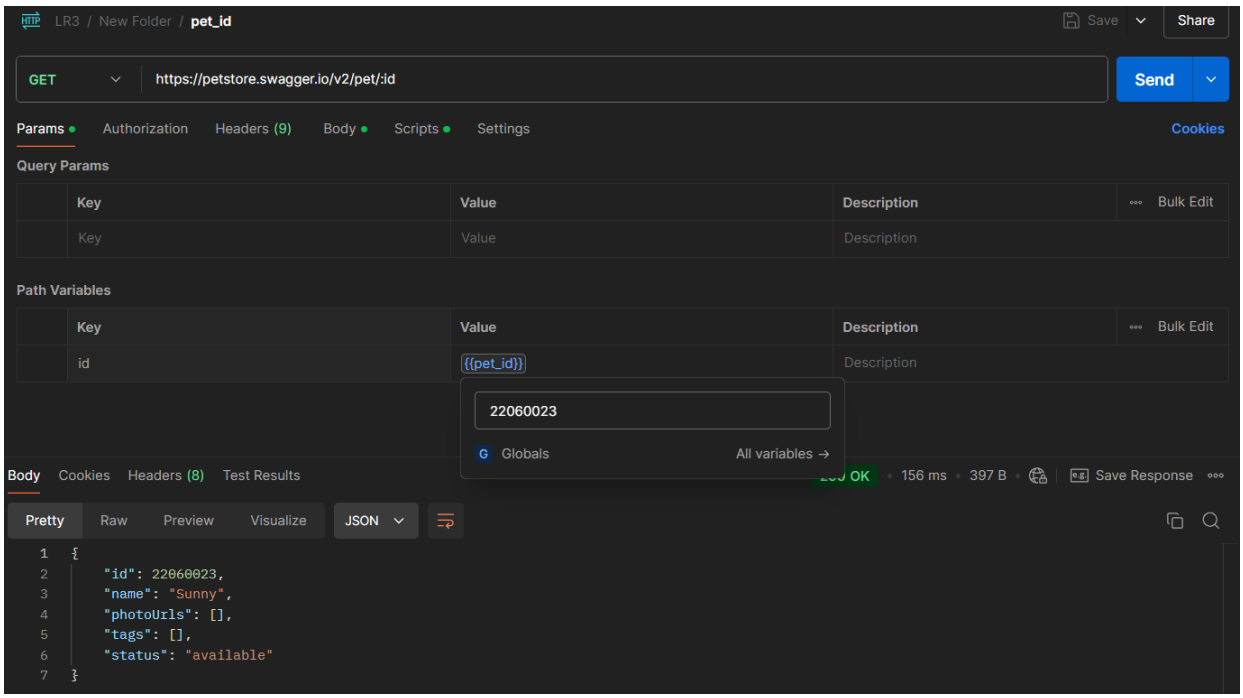

Рис. 1.2.1. GET запрос с глобальной переменной «pet_id»

Дублируем запрос из пункта 9 прошлой лабораторной работы в коллекцию текущей. В теле запроса значение параметра «id» задано через переменную «pet_id». Результат выполнения задания представлен на рис. 1.2.2.

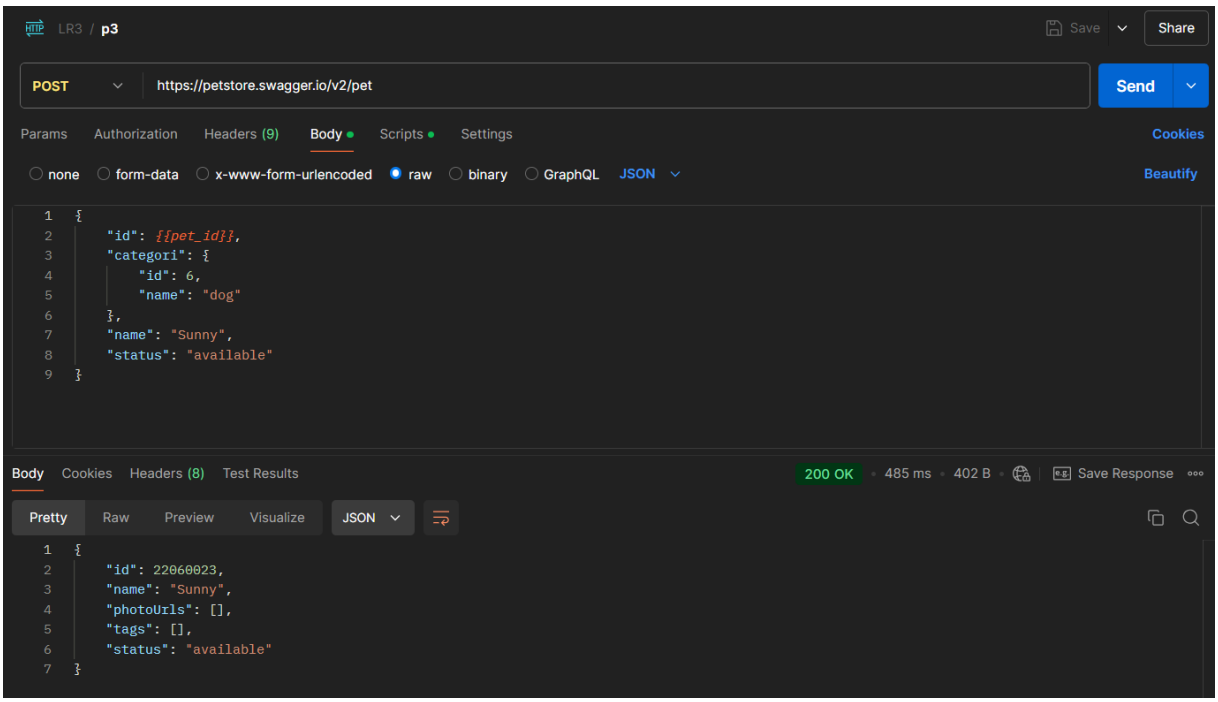

Рис. 1.2.2. POST запрос с переменной «pet_id» вместо «id»

Создаем глобальные переменные: «username» со значением = admin и «password», значение которой задаем как «123». Для переменной пароля задан секретный тип. Результат выполнения задания представлен на рис. 1.2.3.

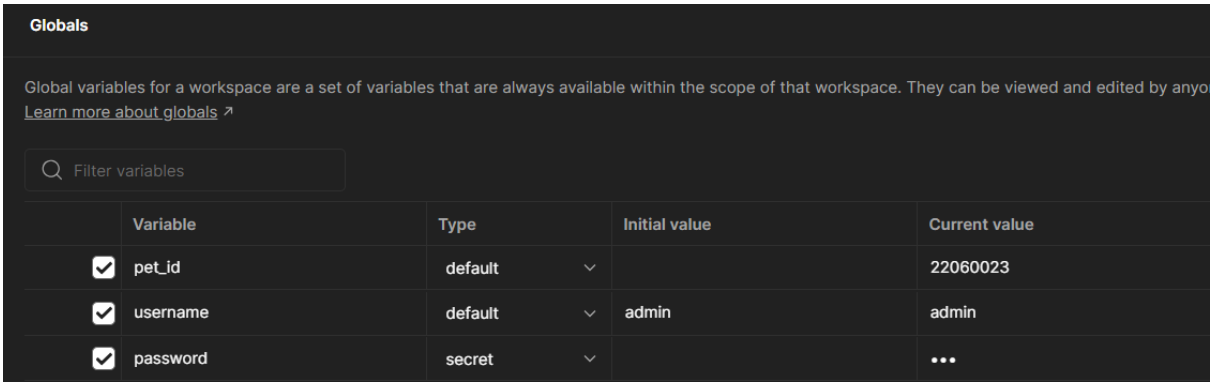

Рис. 1.2.3. Создание глобальных переменных «username» и «password»

Добавляем в коллекцию «LR-3» дубликат запроса из пункта 8, порядка выполнения лабораторной работы № 2. Вносим изменения в запрос так, чтобы в нем использовались переменные «username» и «password», созданные в предыдущем пункте. Результат выполнения задания представлен на рис. 1.2.4.

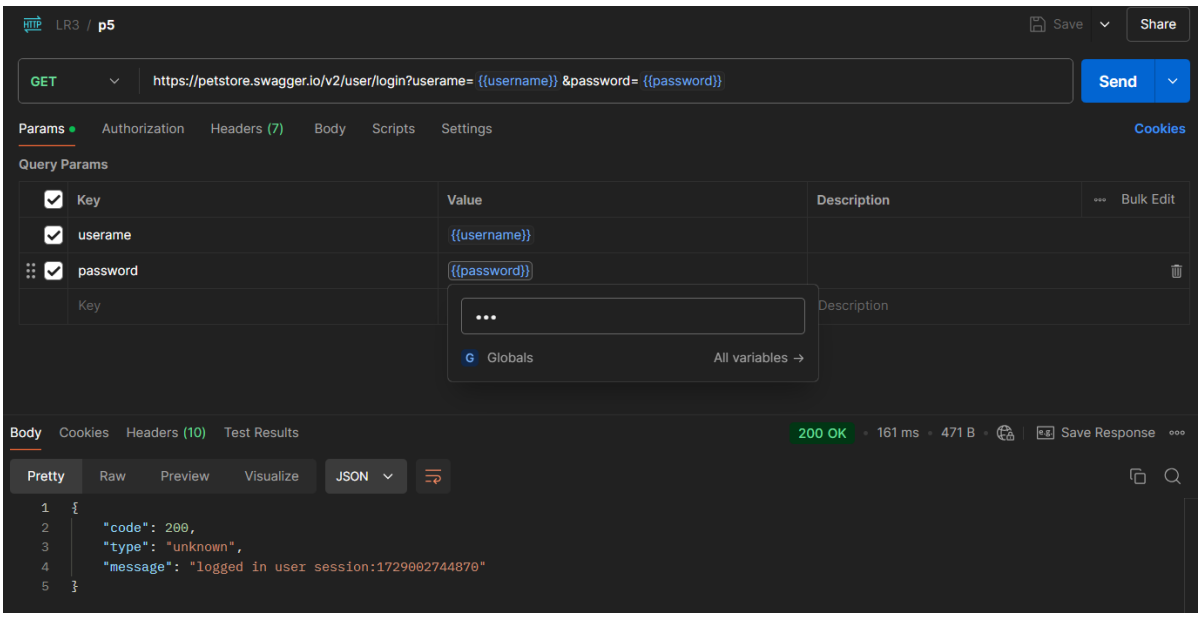

Рис. 1.2.4. GET запрос с использованием переменных «username» и «password»

Создаем POST запрос к сервису «postman-echo», в url которого указываем свое имя (Query Params «name»). В теле данного запроса прописываем параметры «Number», «FavoriteColor», «Honorific», «Position» и «Sphere», значения которых задаем через соответствующие динамические переменные. Результат выполнения задания представлен на рис. 1.2.5.

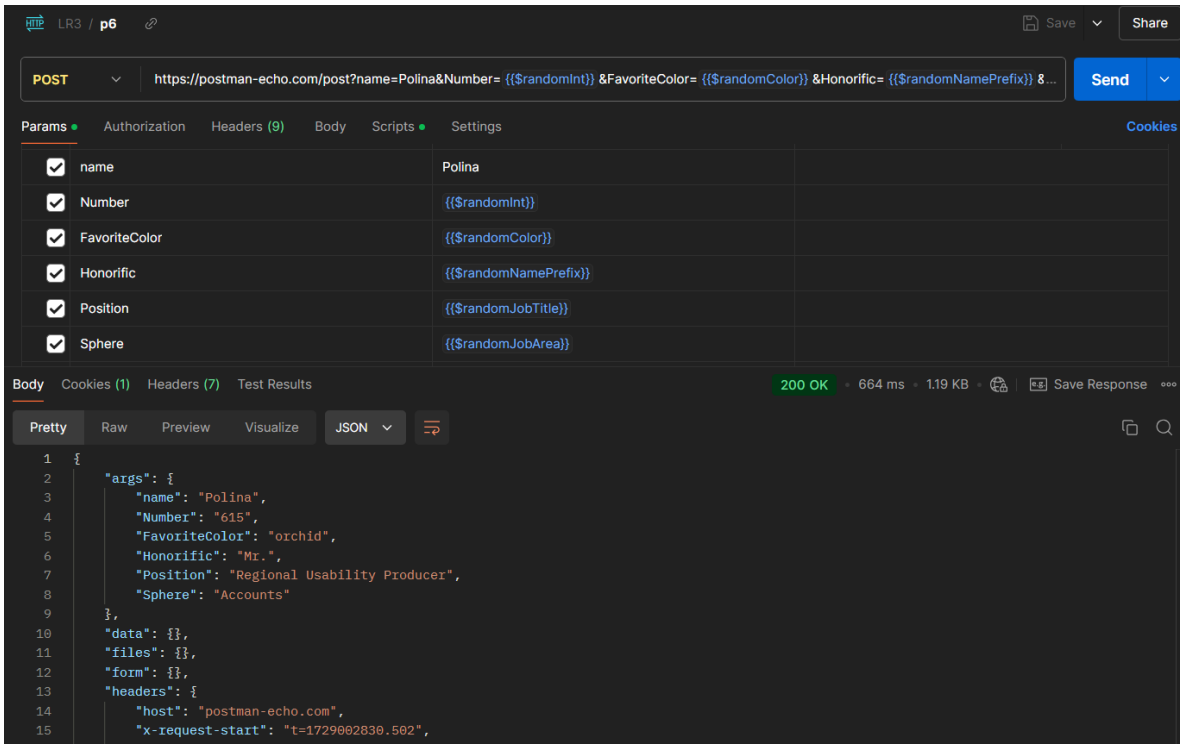

Рис. 1.2.5. POST запрос с динамическими переменными

Создаем папку, перемещаем в нее запрос «pet_id». При помощи команды вывода текста в консоль (console.log("Тип скрипта – уровень скрипта")) добавляем скрипты в каждом из блоков (Pre req / Tests) и на каждом из уровней скриптов (Collection / Folder / Request). Результат выполнения задания с проверкой скриптов на консоли представлен на рис. 1.2.6.

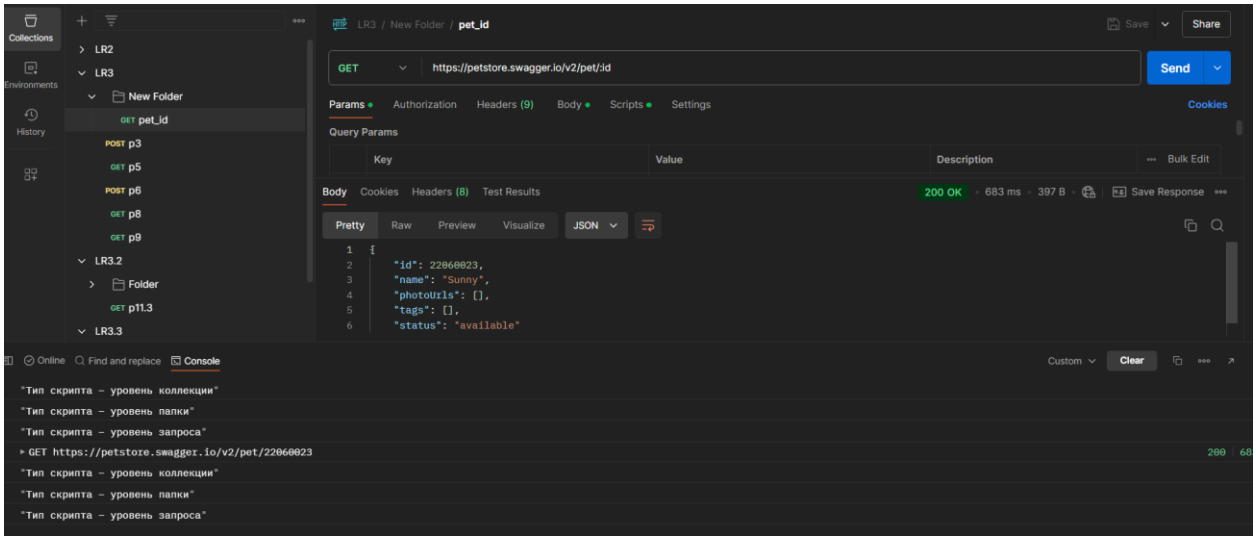

Рис. 1.2.6. Консоль после запуска GET запроса «pet_id» с добавленным скриптами на уровне коллекции, папки и запроса

Создаем переменную уровня коллекции с именем «param» и значением «default». На вкладке «Tests» POST запроса с Вашим именем (пункт № 6) вводим следующий код «pm.collectionVariables.set ("param", JSON.parse(responseBody).args.name)» для сохранения значения параметра из тела ответа в переменную «param». Не выполняя запрос, добавляем в коллекцию GET запрос с эдпоинтом «https://postman-echo.com/get?param={{param}}», после чего на вкладке «Pre-request script» этого же запроса добавляем код, который сделает преобразование значения и снова сохранит его в переменную. Результат выполнения задания представлен на рис. 1.2.7.

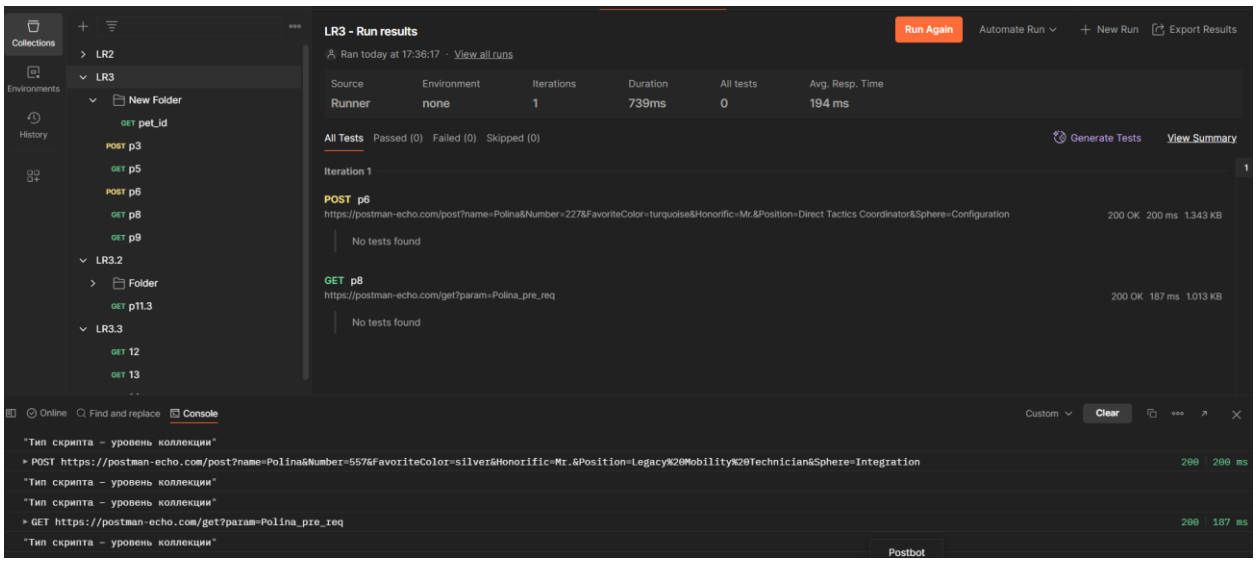

Рис. 1.2.7. Консоль после запуска коллекции с запросами GET и POST

Добавляем переменную (уровня коллекции) «delay», задаем ей значение 2. После чего создаем GET запрос с url «https://postman-echo.com/delay/:delay», в качестве path параметра используем созданную переменную. Далее на вкладке «Tests» необходимо прописать несколько проверок для полученного ответа, которые будут проверять код ответа и соответствие значения delay со значением переменной коллекции. Результат выполнения задания представлен на рис. 1.2.8.

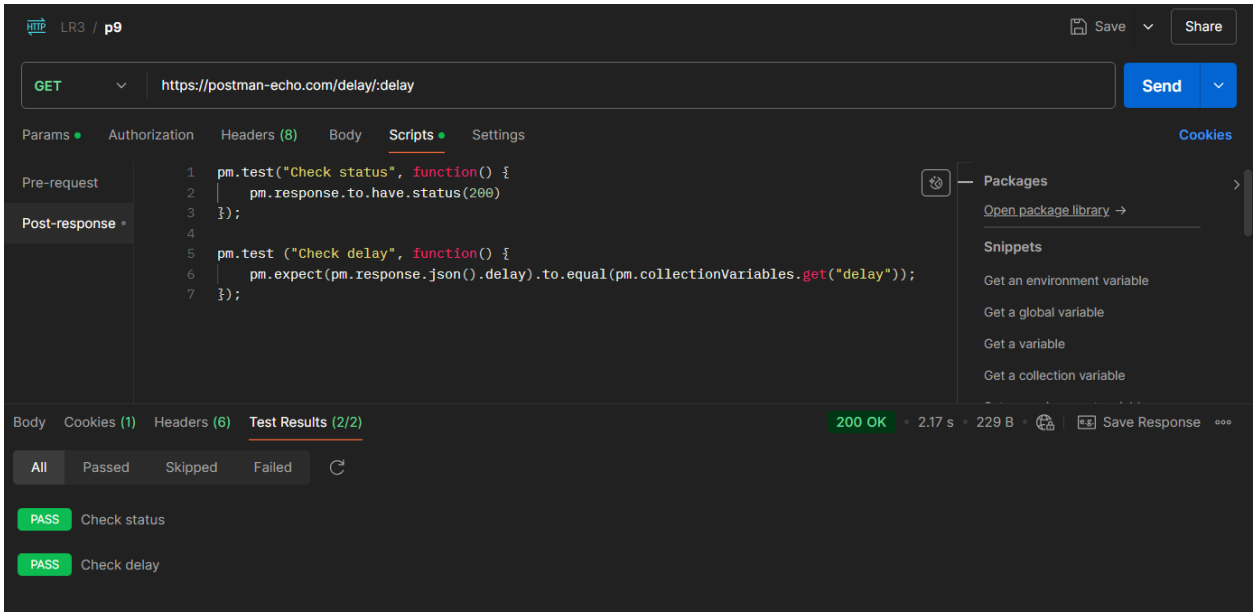

Рис. 1.2.8. GET запрос с переменной «delay»

Изменяем ожидаемое значение в первой проверке на 201 и снова выполняем запрос. Результат выполнения задания представлен на рис. 1.2.9.

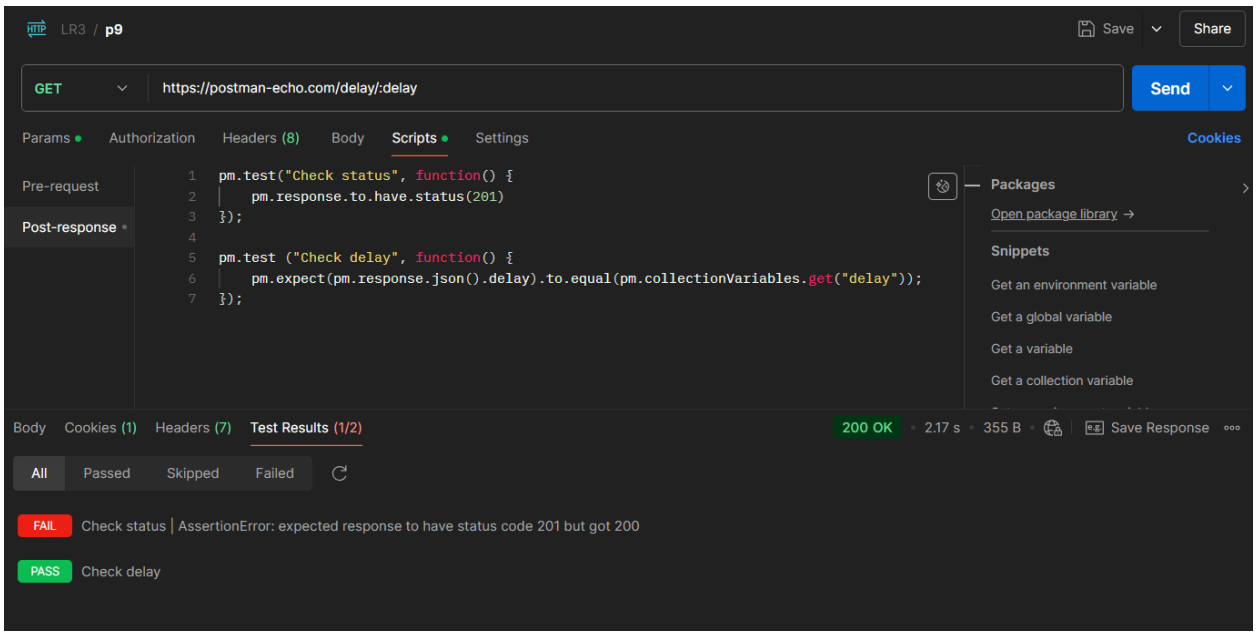

Рис. 1.2.9. GET запрос с переменной «delay» с кодом ответа 201

Создаем дополнительную коллекцию с названием «LR3.2», в описании которой прописываем текущую дату и собственное ФИО. В данной коллекции создаем папку с именем «Folder», 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 внутри коллекции добавляем еще один GET запрос «https://postman-echo.com/delay/:delay», параметр «delay» которого необходимо задавать из переменной коллекции с таким же именем. Для этого же запроса задаем скрипт как на рисунке 3.7. Затем добавьте тестовый скрипт на уровне папки (рис. 3.8). Последним пропишите скрипт уровня коллекции, код которого представлен на рисунке 3.9. Результат выполнения задания после запуска коллекции представлен на рис. 1.2.10.

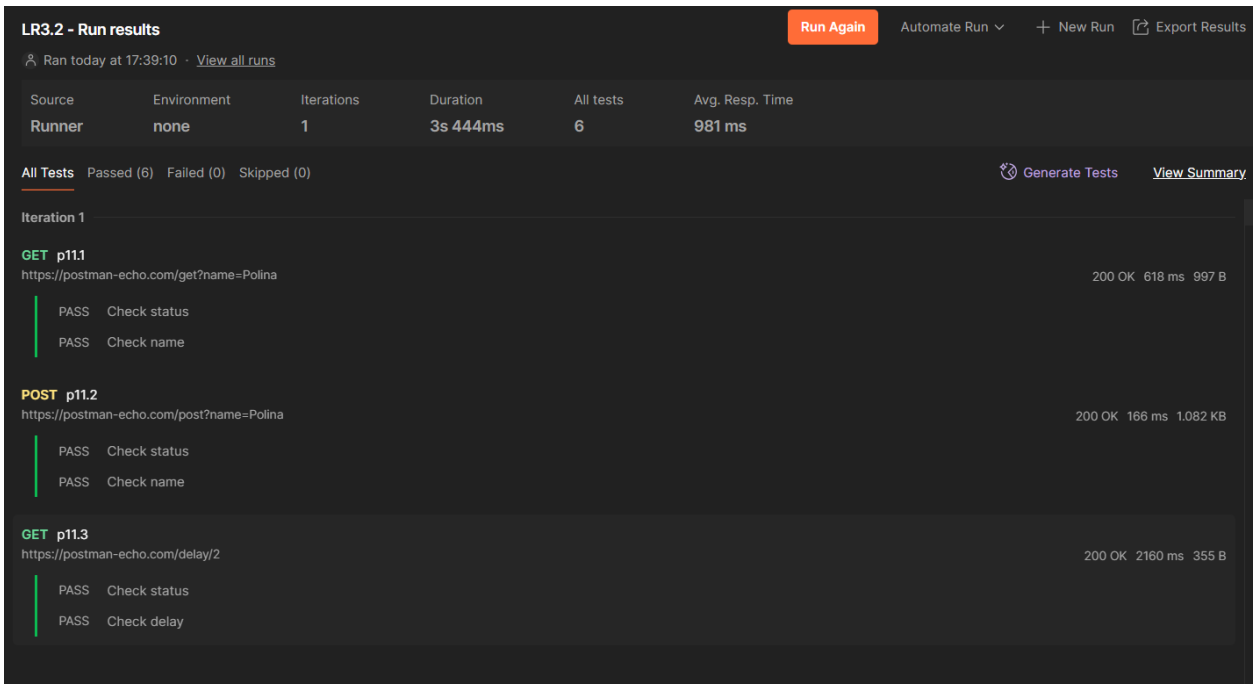

Рис. 1.2.10. Результат запуска коллекции «LR3.2» со всеми созданными в ней запросами

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».

Далее создаем дополнительную коллекцию с названием «LR3.3», создаем окружение «Local» (команда «Create new environment» на вкладке «Environments»). На всех уровнях (глобальном, коллекции, окружения) создаем переменную «count» со значениями: номер группы (без дефиса, например, 2205) для глобальной переменной, порядковый номер в списке группы (1, 2, 3 и т. д.) – для переменной коллекции и номер текущего месяца – для локальной переменной. В созданную коллекцию добавляем GET запрос с эдпоинтом «https://postman-echo.com/get?param={{count}}». Результат выполнения задания представлен на рис. 1.2.11.

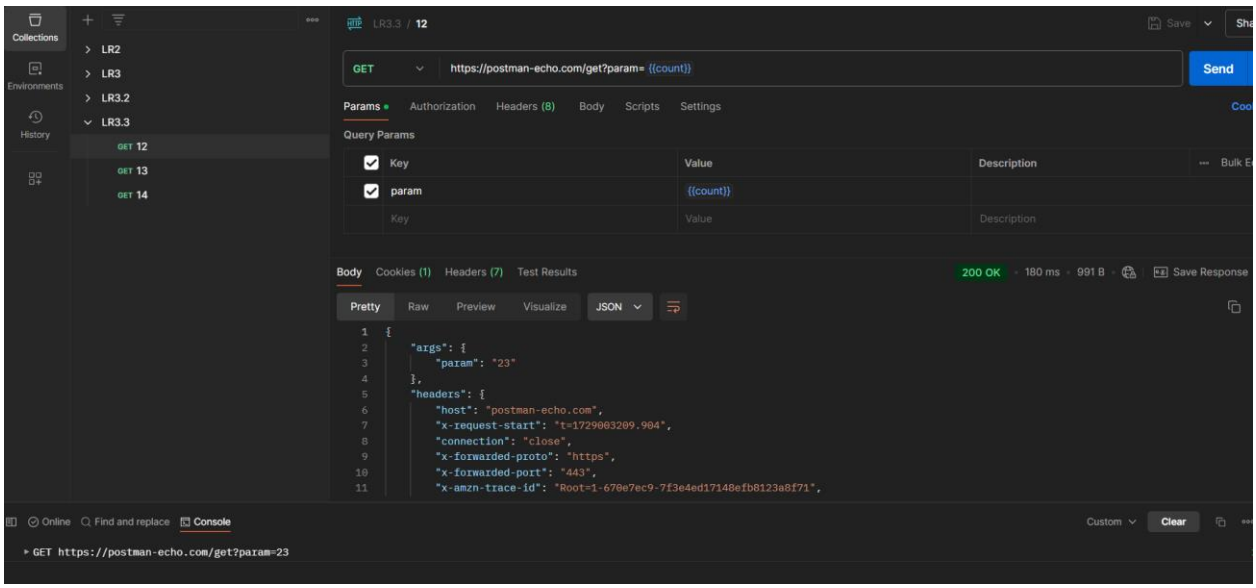

Рис. 1.2.11. GET запрос с переменной коллекции «count»

Проверяем существование глобальных переменных с именами «count» и «coun2» путем написания для каждой их них команды для тестового скрипта «console.log("Does count exist? : " + pm.globals.has("count"));». Результат выполнения задания представлен на рис. 1.2.12.

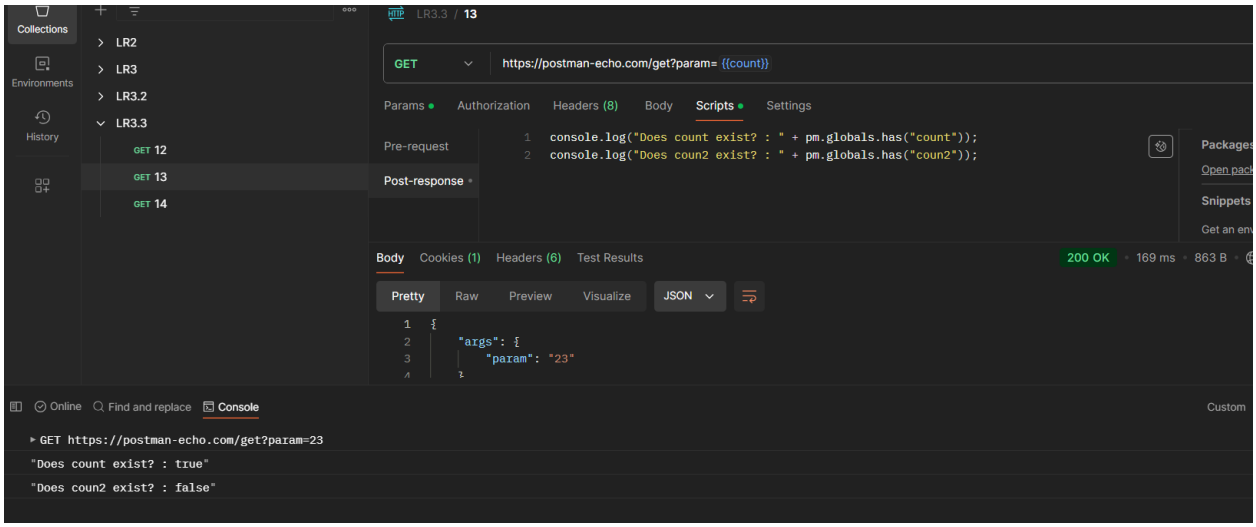

Рис. 1.2.12. Проверка на существование глобальных переменных «count» и «coun2»

Дублируем запрос 12 и измененяем в нем код так, чтобы в консоль выводились все заданные значения переменной «count». Результат выполнения задания представлен на рис. 1.2.13.

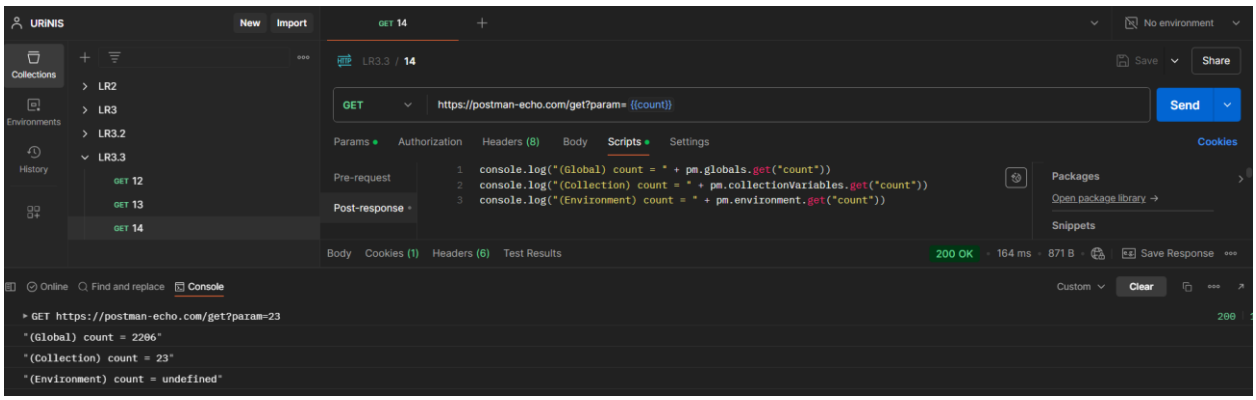

Рис. 1.2.13. GET запрос с выводом на консоль всех значений «count» в окружении «No Environment»

После прихода ответа от сервера, не отчищая консоль, меняем окружение с «No Environment» на «Local» и снова выполняем запрос. Результат выполнения задания представлен на рис. 1.2.14.

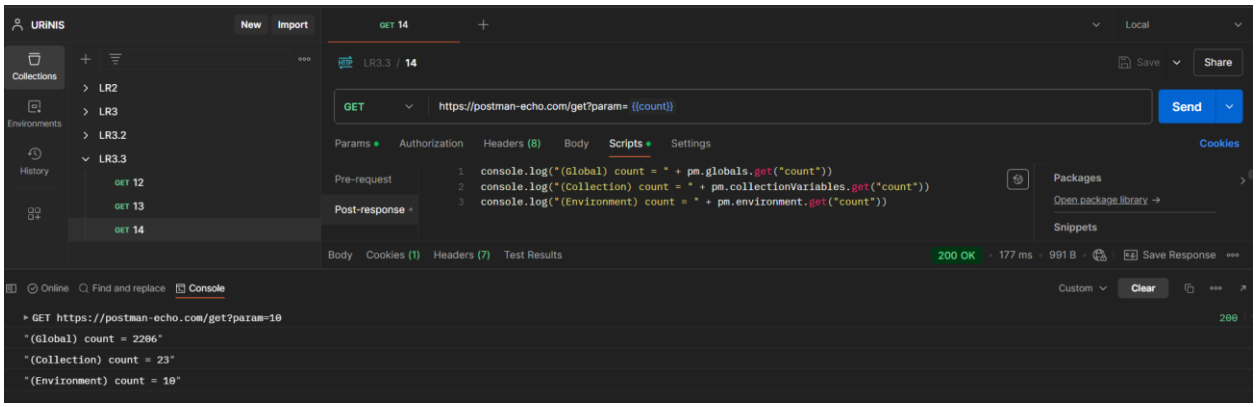

Рис. 1.2.14. GET запрос с выводом на консоль всех значений «count» в окружении «Local»

При смене окружения с «No Environment» на «Local» меняется переменная «(Environment) count», так как данная переменная создана в окружении «Local» и не видна при использовании другого окружения.

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.