

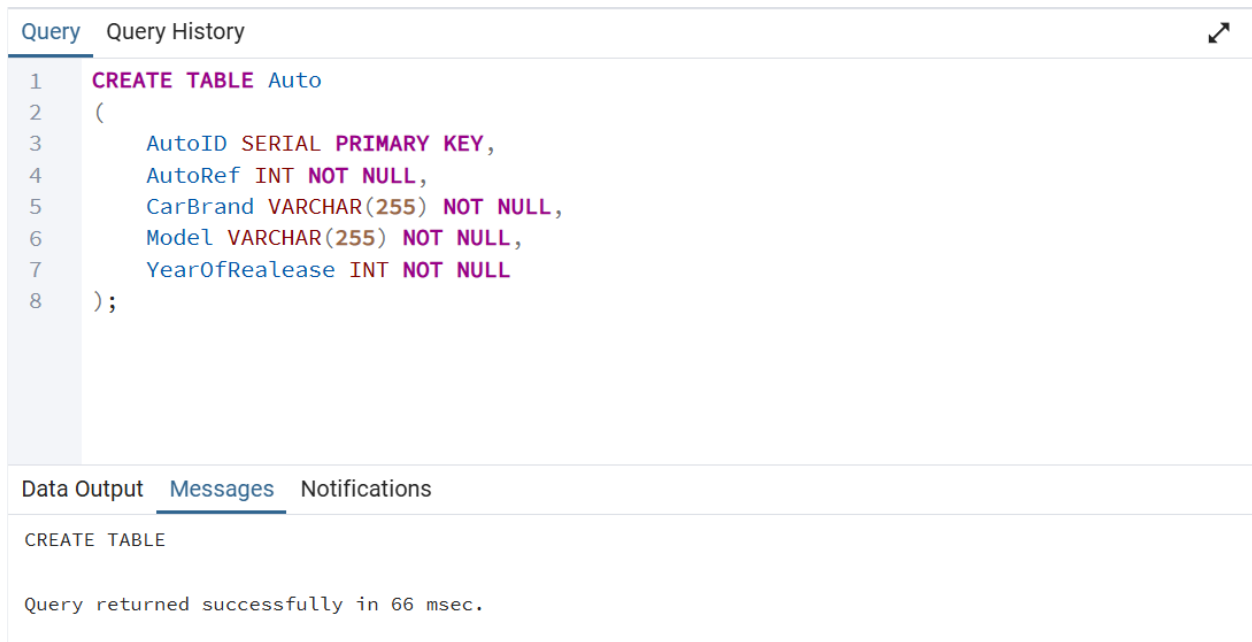
SQL

1. Создание таблицы Auto

Назначение: создание базовой структуры для хранения информации об автомобилях

Ключевые особенности:

- Организация данных о марках, моделях и годах выпуска
- Создание основной таблицы для дальнейшей работы
- Формирование структуры базы данных



The screenshot shows a SQL query editor interface. At the top, there are tabs for 'Query' and 'Query History'. The 'Query' tab is active, displaying a SQL statement to create a table named 'Auto'. The statement is as follows:

```
1 CREATE TABLE Auto
2 (
3     AutoID SERIAL PRIMARY KEY,
4     AutoRef INT NOT NULL,
5     CarBrand VARCHAR(255) NOT NULL,
6     Model VARCHAR(255) NOT NULL,
7     YearOfRelease INT NOT NULL
8 );
```

Below the query editor, there are tabs for 'Data Output', 'Messages', and 'Notifications'. The 'Messages' tab is active, showing the following output:

```
CREATE TABLE

Query returned successfully in 66 msec.
```

2. Создание таблицы AutoPrice

Назначение: формирование структуры для хранения ценовой информации

Ключевые особенности:

- Связывание с основной таблицей автомобилей
- Хранение актуальных цен
- Создание связи между таблицами

Query

Query History

1

2

3

4

5

6

CREATE TABLE AutoPrice

(

AutoPriceID SERIAL PRIMARY KEY,

AutoPriceRef INT NOT NULL,

Price INT NOT NULL

);

Data Output

Messages

Notifications

CREATE TABLE

Query returned successfully in 30 msec.

3-4. Заполнение таблиц данными

Назначение: инициализация базы данных тестовой информацией

Ключевые особенности:

- Добавление 10 записей в каждую таблицу
- Демонстрация работы с массовыми вставками
- Формирование рабочей выборки данных

Query

Query History

1

2

3

4

5

6

7

8

9

10

11

12

13

INSERT INTO Auto

(AutoRef, CarBrand, Model, YearOfRelease)

VALUES

(1, 'Toyota', 'Camry', 2020),

(2, 'Toyota', 'RAV4', 2023),

(3, 'Honda', 'Accord', 2012),

(4, 'Honda', 'Civic', 2008),

(5, 'Audi', 'A6', 2020),

(6, 'Porsche', 'Cayenne', 2010),

(7, 'Hyundai', 'Solaris', 2013),

(8, 'Hyundai', 'Sonata', 2017),

(9, 'Lexus', 'LX', 2024),

(10, 'Mazda', '6', 2016);

Data Output

Messages

Notifications

INSERT 0 10

Query returned successfully in 47 msec.

```
Query  Query History  ↗
1  INSERT INTO AutoPrice
2      (AutoPriceRef, Price)
3  VALUES
4      (1, 1779000),
5      (2, 2800000),
6      (3, 1200000),
7      (4, 945000),
8      (5, 3350000),
9      (6, 3100000),
10     (7, 990000),
11     (8, 1700000),
12     (9, 15990000),
13     (10, 19000000);
```

Data Output Messages Notifications

```
INSERT 0 10
```

Query returned successfully in 38 msec.

5. Выборка первых 3 авто с сортировкой

Назначение: демонстрация получения первых записей с упорядочиванием

Ключевые особенности:

- Применение сортировки по возрастанию (ASC)
- Комбинация LIMIT и ORDER BY
- Получение упорядоченных результатов
- Работа с базовыми методами сортировки

Query

Query History

1

SELECT *

2

FROM Auto

3

ORDER BY AutoID ASC

4

LIMIT 3;

Data Output

Messages

Notifications

+

📄

▼

📋

▼

🗑️

🗄️

⬇️

📈

SQL

Showing rows: 1 to 3

🖋️

Page

	autoid [PK] integer	autoref integer	carbrand character varying (255)	model character varying (255)	yearofrelease integer
1	1	1	Toyota	Camry	2020
2	2	2	Toyota	RAV4	2023
3	3	3	Honda	Accord	2012

6. Выборка последних 5 авто

Назначение: получение последних добавленных записей

Ключевые особенности:

- Сортировка данных в обратном порядке
- Комбинация ORDER BY и LIMIT
- Работа с последними записями

Query

Query History

1

SELECT *

2

FROM Auto

3

ORDER BY AutoID DESC

4

LIMIT 5;

Data Output

Messages

Notifications

≡

📄

▼

📋

▼

🗑️

🗄️

⬇️

📈

SQL

Showing rows: 1 to 5

✎

Page

	autoid [PK] integer ✎	autoref integer ✎	carbrand character varying (255) ✎	model character varying (255) ✎	yearofrelease integer ✎
1	10	10	Mazda	6	2016
2	9	9	Lexus	LX	2024
3	8	8	Hyundai	Sonata	2017
4	7	7	Hyundai	Solaris	2013
5	6	6	Porche	Cayenne	2010

7. Поиск самого молодого авто

Назначение: определение автомобиля с максимальным годом выпуска

Ключевые особенности:

- Использование функции MAX для поиска максимального значения
- Получение одной записи с самым новым автомобилем
- Работа с числовыми данными через агрегатные функции

Query

Query History

1

SELECT *

2

FROM Auto

3

WHERE YearOfRelease = (SELECT MAX(YearOfRelease) FROM Auto);

Data Output

Messages

Notifications

Showing rows: 1 to 1

Page

autoid

[PK] integer

autoref

integer

carbrand

character varying (255)

model

character varying (255)

yearofrelease

integer

1

9

9

Lexus

LX

2024

8. Поиск самого старого авто

Назначение: определение автомобиля с минимальным годом выпуска

Ключевые особенности:

- Применение функции MIN для поиска минимального значения
- Получение единственной записи со старым автомобилем
- Работа с числовыми данными через агрегатные функции

Query

Query History

1

2

3

SELECT *

FROM Auto

WHERE YearOfRelease = (SELECT MIN(YearOfRelease) FROM Auto);

Data Output

Messages

Notifications

<

9. Объединение таблиц через INNER JOIN

Ключевые особенности:

- Query

Query History

1

2

3

4

SELECT AutoRef, CarBrand, Model, YearOfRealease, Price
FROM Auto
INNER JOIN AutoPrice ON AutoRef = AutoPriceRef;

Data Output

Messages

Notifications

≡

📄

▼

📋

▼

🗑️

🗄️

⬇️

📈

SQL

Showing rows: 1 to 10

✎

	autoref integer	carbrand character varying (255)	model character varying (255)	yearofrealease integer	price integer
1	1	Toyota	Camry	2020	1779000
2	2	Toyota	RAV4	2023	2800000
3	3	Honda	Accord	2012	1200000
4	4	Honda	Civic	2008	945000
5	5	Audi	A6	2020	3350000
6	6	Porche	Cayenne	2010	3100000

Total rows: 10

Query complete 00:00:00.069

10. Обновление данных с условием

Ключевые особенности:

- Использование оператора AND для комбинирования условий
- Обновление конкретных записей по сложным критериям
- Работа с числовыми и строковыми данными одновременно
- Демонстрация точного обновления данных

Query

Query History

1

UPDATE Auto

2

SET YearOfRelease = 2025

3

WHERE CarBrand = 'Toyota' AND Model = 'Camry';

4

Data Output

Messages

Notifications

UPDATE 1

Query returned successfully in 44 msec.

11. Удаление записи с условием

Назначение: удаление данных с применением множественных условий

Ключевые особенности:

- Применение оператора AND для фильтрации
- Безопасное удаление записей по нескольким критериям
- Работа с составными условиями
- Демонстрация ответственного подхода к удалению данных

Query

Query History

1

DELETE

2

FROM Auto

3

WHERE CarBrand = 'Toyota' AND Model = 'RAV4';

4

Data Output

Messages

Notifications

DELETE 1

Query returned successfully in 30 msec.

12. Сложный запрос с фильтрацией

Ключевые особенности:

- Применение нескольких условий одновременно
- Использование NOT IN и LIKE
- Работа со строковыми данными
- Фильтрация по нескольким критериям

Query

Query History

1

SELECT *

2

FROM Auto

3

WHERE CarBrand NOT IN('Toyota','Honda') AND Model LIKE 'S%';

+

📄

▼

📋

▼

🗑️

🗄️

⬇️

📈

SQL

Showing rows: 1 to 2

Page

	autoid [PK] integer	autoref integer	carbrand character varying (255)	model character varying (255)	yearofrelease integer
1	7	7	Hyundai	Solaris	2013
2	8	8	Hyundai	Sonata	2017

13. Удаление таблицы

Назначение: управление структурой базы данных

Ключевые особенности:

- Демонстрация DDL операций
- Работа с командами управления структурой
- Понимание последствий удаления данных



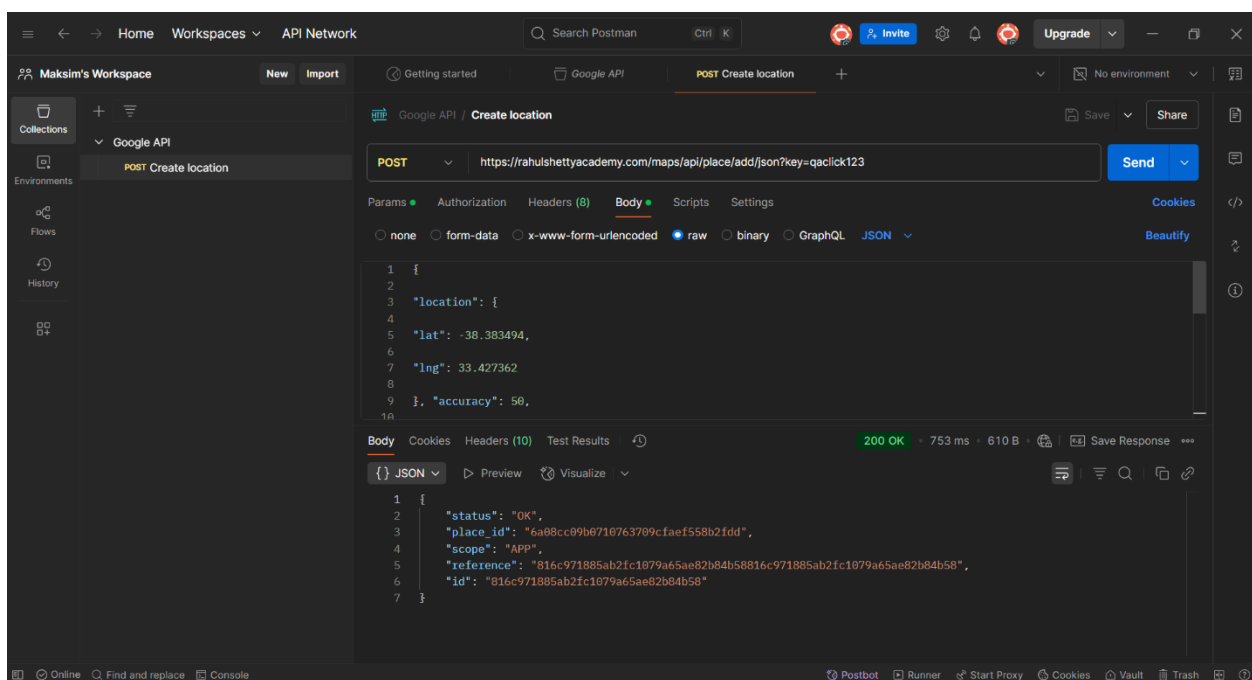
Postman

1. POST-запрос создания ресурса

Назначение: демонстрация создания нового объекта через API

Ключевые особенности:

- Отправка данных для создания нового ресурса
- Получение уникального идентификатора
- Проверка успешного создания (статус 201)
- Валидация обязательных полей

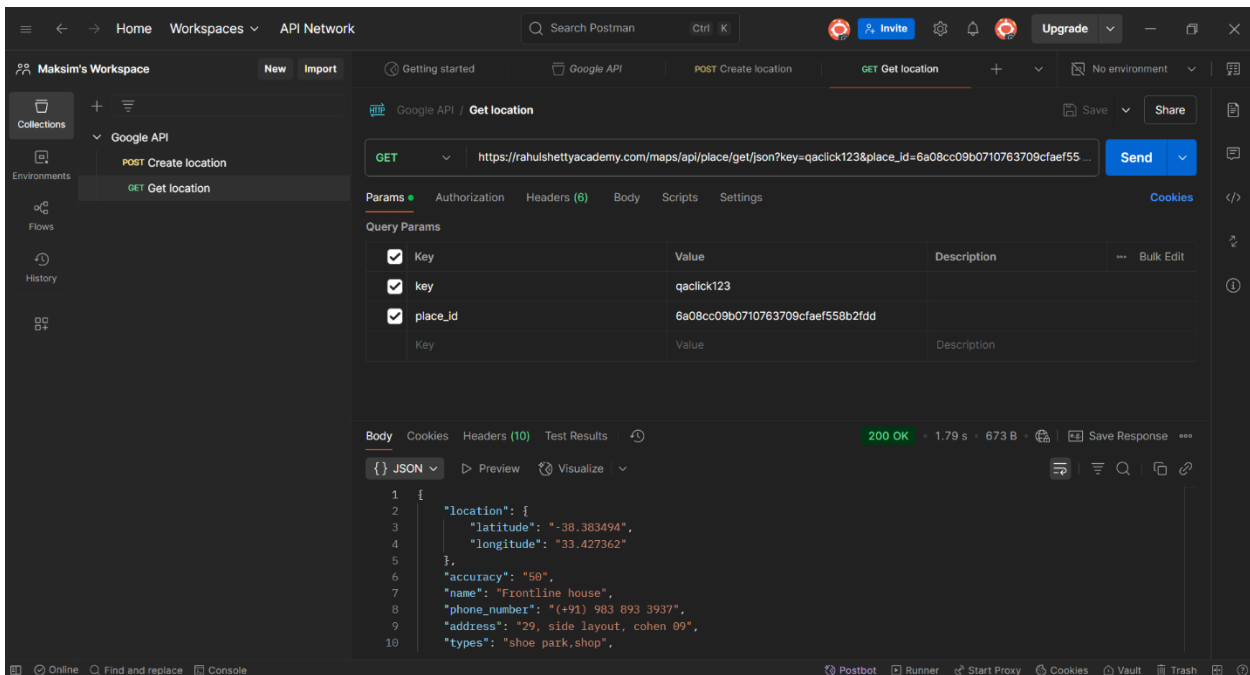


2. GET-запрос с валидным ID

Назначение: получение данных существующего объекта

Ключевые особенности:

- Использование корректного place_id
- Проверка успешного ответа (статус 200)
- Верификация полученных данных
- Демонстрация корректной работы API

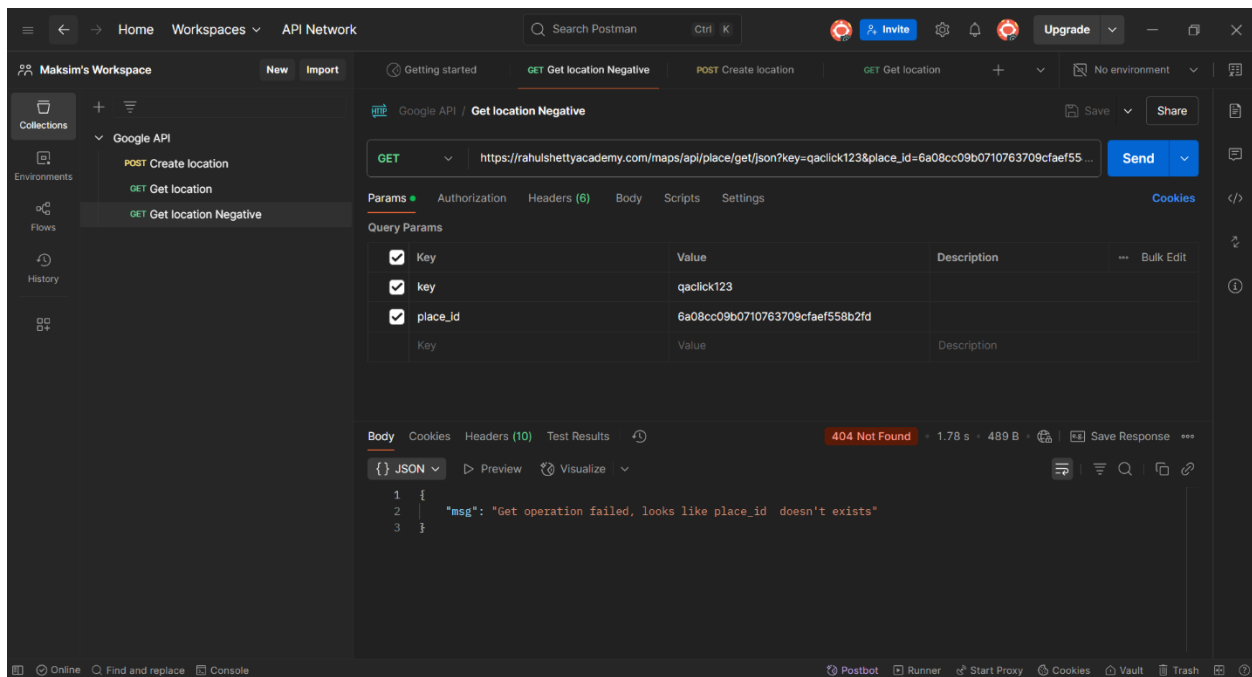


3. GET-запрос с некорректным ID

Назначение: проверка обработки ошибок

Ключевые особенности:

- Использование несуществующего place_id
- Проверка ошибки 404
- Верификация сообщения об ошибке
- Демонстрация обработки некорректных данных

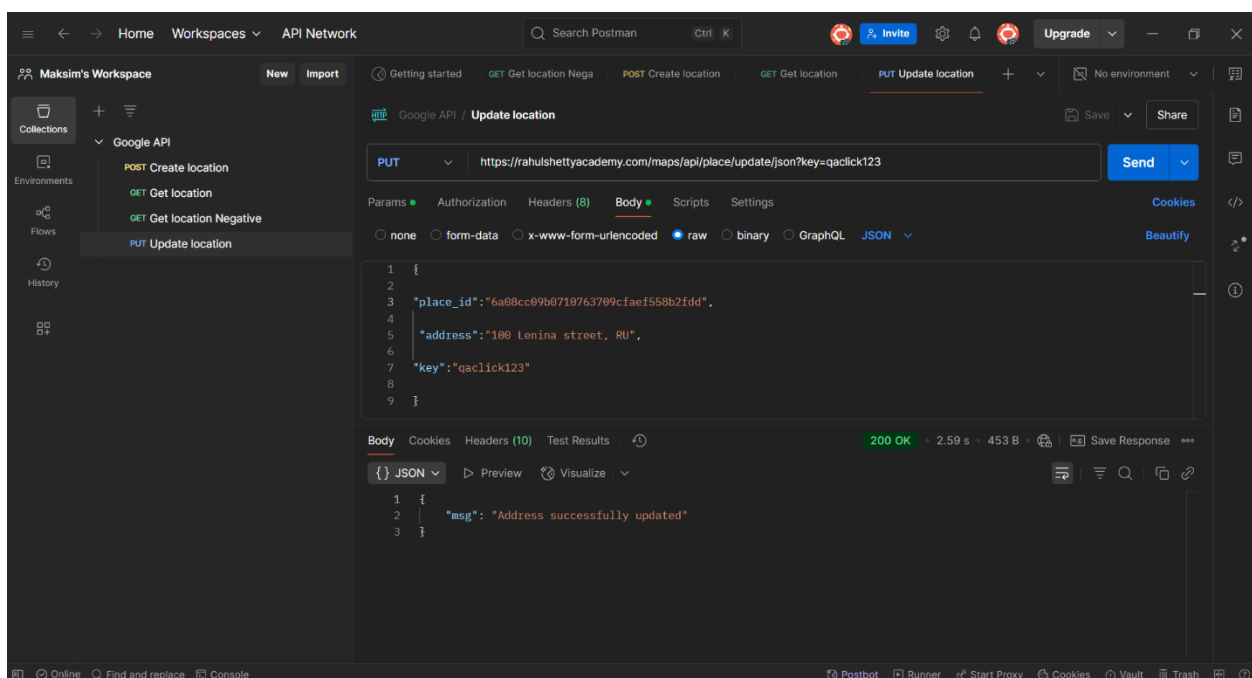


4. PUT-запрос обновления

Назначение: модификация существующего объекта

Ключевые особенности:

- Отправка обновленных данных
- Использование валидного place_id
- Проверка успешного обновления (статус 200)
- Верификация измененных полей

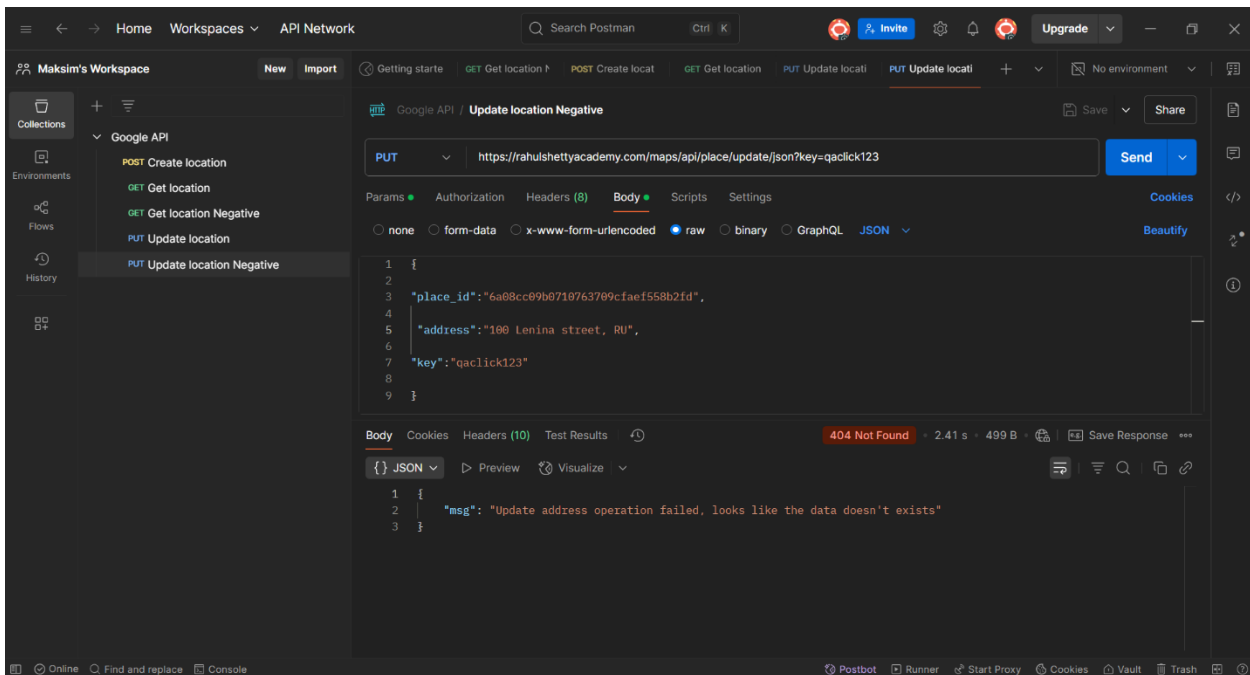


5. PUT-запрос с некорректным ID

Назначение: тестирование обработки ошибок обновления

Ключевые особенности:

- Попытка обновления несуществующего объекта
- Проверка ошибки 404
- Верификация сообщения об ошибке
- Демонстрация корректной обработки

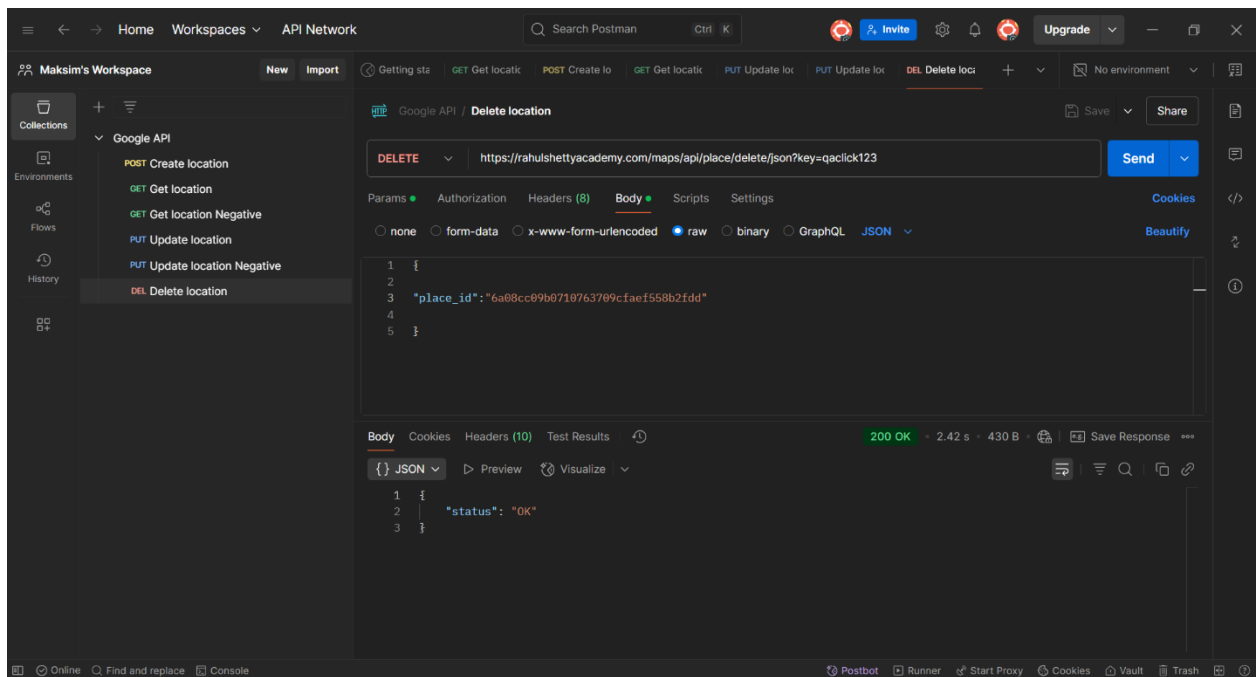


6. DELETE-запрос удаления

Назначение: удаление существующего объекта

Ключевые особенности:

- Использование валидного place_id
- Проверка успешного удаления (статус 200)
- Верификация отсутствия данных после удаления

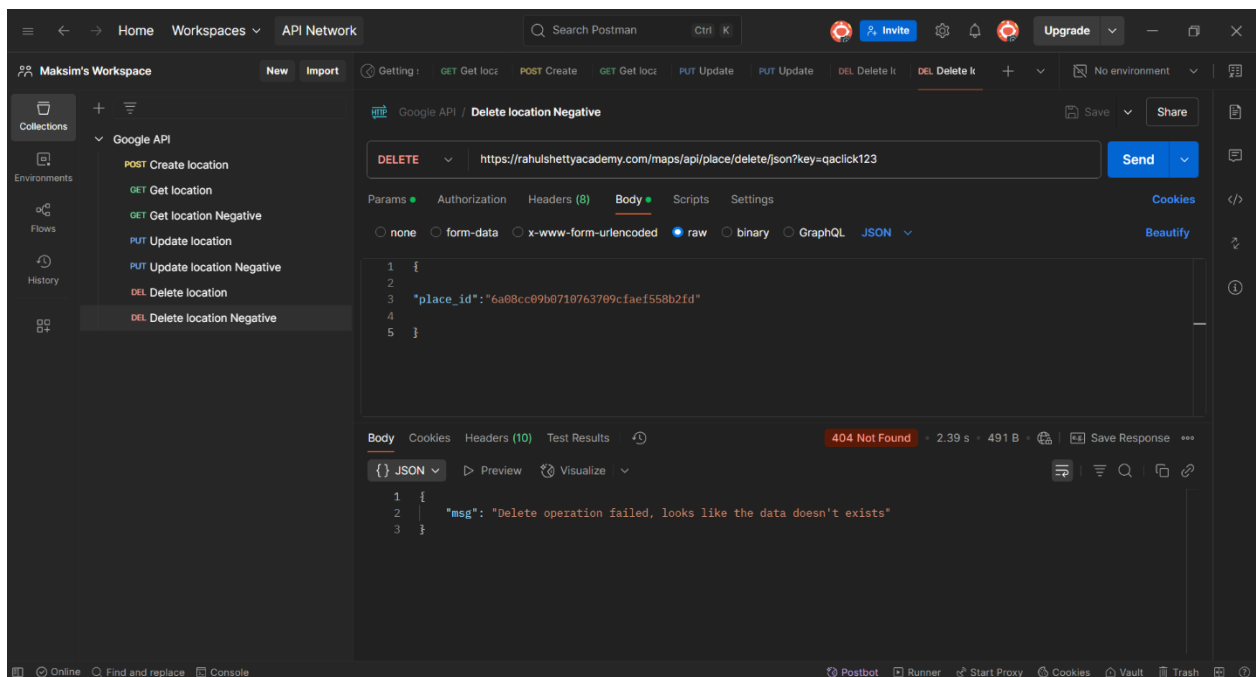


7. DELETE-запрос с некорректным ID

Назначение: проверка обработки ошибок удаления

Ключевые особенности:

- Попытка удаления несуществующего объекта
- Проверка ошибки 404
- Верификация сообщения об ошибке
- Демонстрация корректной обработки



Проект нагрузочного тестирования веб-сайта КиноПоиск

1. Введение

Цель тестирования: оценка производительности веб-сайта КиноПоиск при базовой нагрузке с учетом автоматической авторизации.

Период проведения: 14 июля 2025 года

Общая длительность теста: 31 секунда

Окружение тестирования:

- JMeter версии 5.6.3
- Операционная система: Windows 11
- Тестирование проводится на реальном сайте
- Реализация авторизации через встроенные механизмы

2. Методология тестирования

Используемые инструменты:

- Apache JMeter 5.6.3
- HTTP Request Sampler
- Header Manager
- View Results Tree
- Summary Report

3. Тестовые сценарии

Основные сценарии тестирования:

- Сценарий 1: тестирование главной страницы
- Сценарий 2: проверка раздела «Фильмы»
- Сценарий 3: анализ раздела «Топ-250»
- Сценарий 4: тестирование страницы фильма

Параметры выполнения теста:

- Ramp-up period: 1 секунда
- Loop Count: 1 цикл
- Время выполнения всего теста: 31 секунда

4. Конфигурация тестирования

Параметры нагрузки:

- Количество виртуальных пользователей: 1 (базовый сценарий)
- Интервал между запросами: 10 секунд
- Настройки Header Manager: включены необходимые заголовки

5. Результаты тестирования

Анализ результатов:

- Успешное выполнение всех сценариев за 31 секунду
- Стабильное время отклика страниц
- Корректная работа системы

6. Метрики производительности

Ключевые показатели:

- Общая длительность теста: 31 секунда
- Среднее время ответа сервера
- Процент успешных запросов: 100%
- Стабильность работы системы

7. Выводы и рекомендации

Результаты анализа:

- Успешное прохождение всех сценариев
- Система показала стабильную работу
- Тест выполнен в установленные временные рамки

Рекомендации:

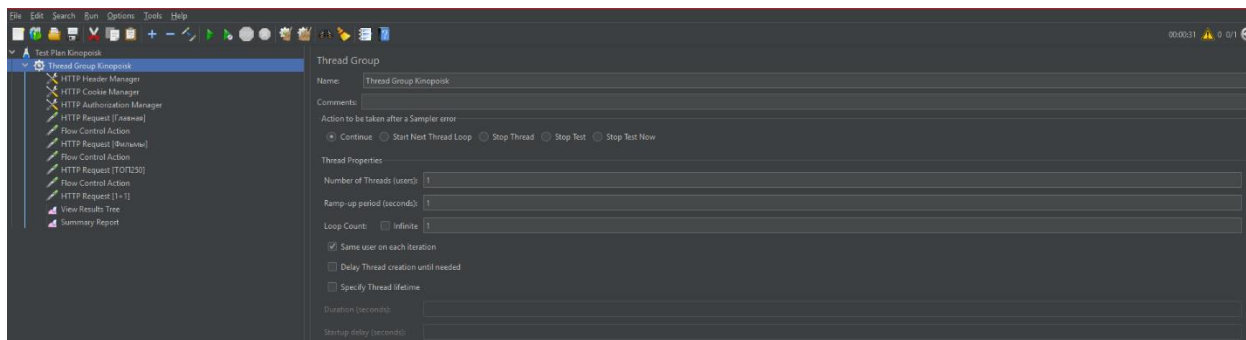
- Провести тестирование при увеличенной нагрузке
- Добавить мониторинг времени отдельных запросов
- Рассмотреть расширение тестовых сценариев

8. Приложения

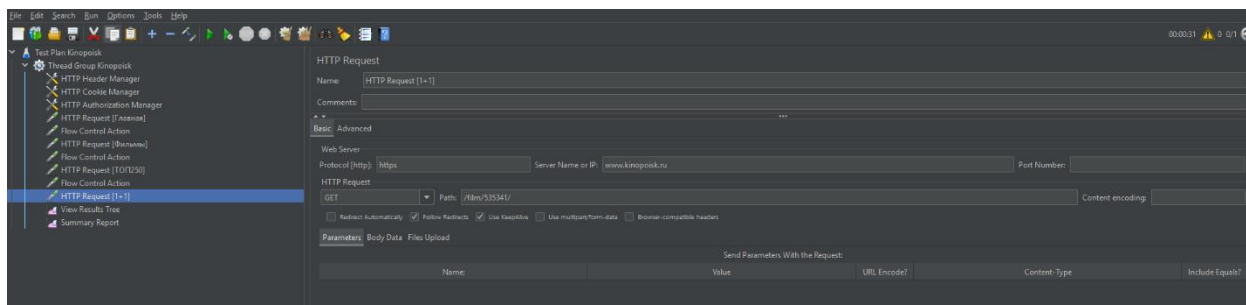
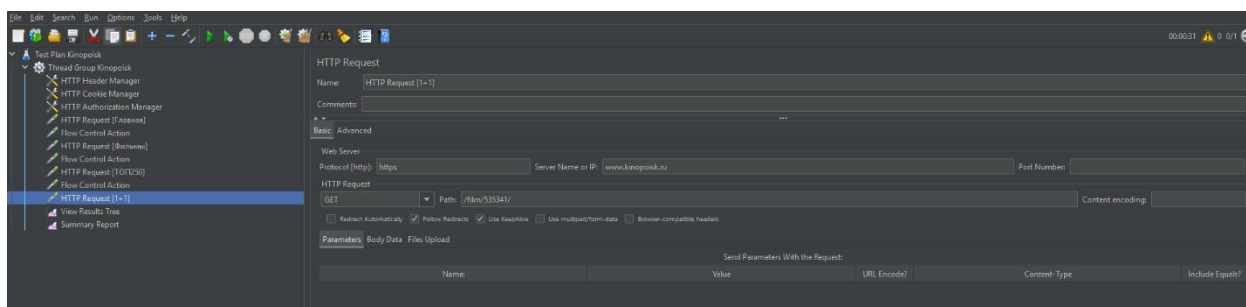
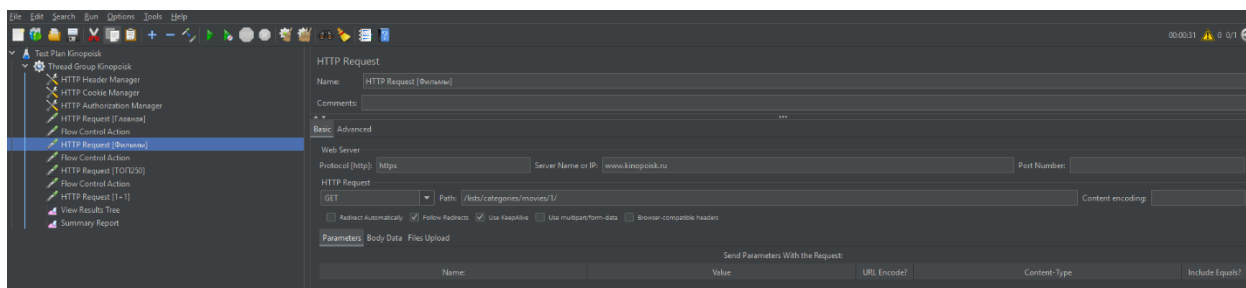
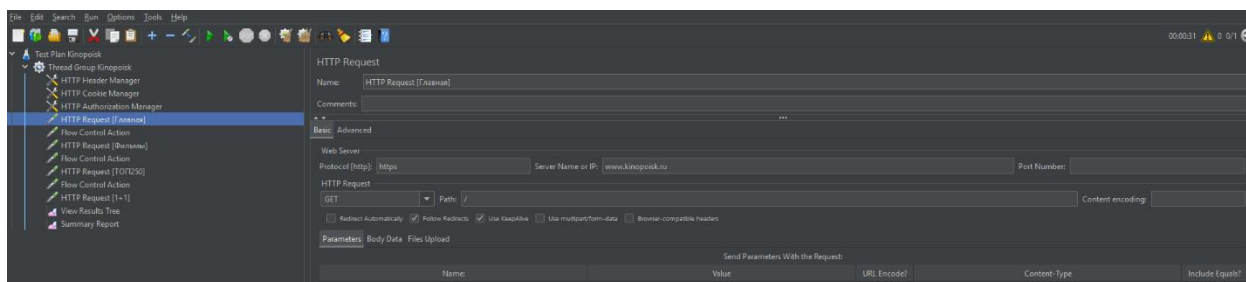
Визуальная документация:

Скриншоты конфигурации (исключая авторизацию):

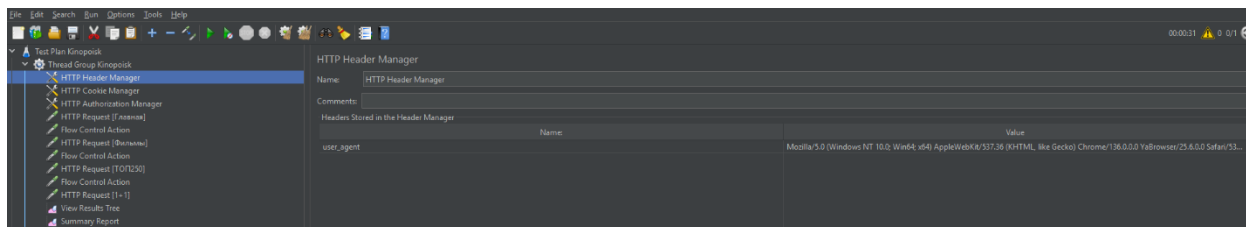
Скрин 1: общая конфигурация тест-плана



Скрин 2: настройки HTTP Request

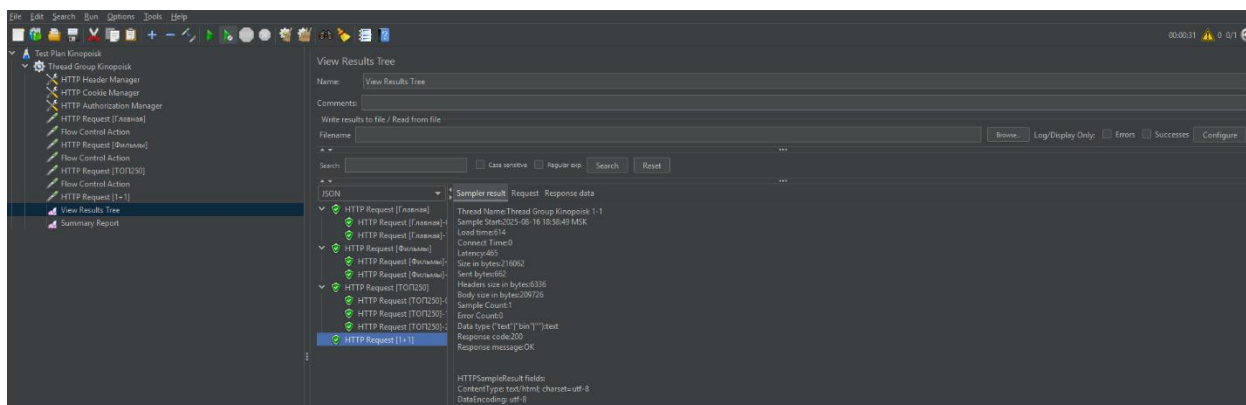


Скрин 3: настройки Header Manager

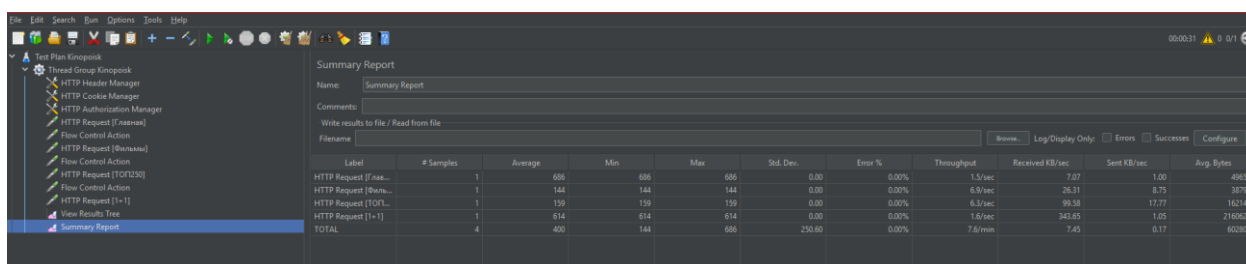


Результаты выполнения:

Скрин 4: View Results Tree



Скрин 5: Summary Report



9. Заключение

Достигнутые результаты:

- Успешное проведение нагрузочного тестирования
- Проверка работоспособности системы
- Сбор базовых метрик производительности

Особенности проекта:

- Компактное время выполнения теста
- Использование продуктивной среды
- Применение современных инструментов тестирования

10. План дальнейшего развития

Предстоящие задачи:

- Увеличение количества виртуальных пользователей
- Добавление стресс-тестов
- Внедрение дополнительных метрик мониторинга
- Автоматизация процесса тестирования

Примечание: Конфигурация авторизации не представлена в связи с требованиями конфиденциальности.