**МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ**

**Ордена Трудового Красного Знамени**

**Федеральное государственное бюджетное образовательное учреждение высшего образования**

**«Московский технический университет связи и информатики»**

Кафедра «Математическая Кибернетика и Информационные технологии»

**Лабораторная работа № 13. Основы работы с протоколом HTTP**

Выполнил: Студент группы

БПИ2401

Исламов Эмин

**Цель работы**

Получение практических навыков работы с Postman для тестирования и разработки API, изучение основных методов HTTP, создание и отправка запросов, анализ ответов сервера.

**Задание 1: Ознакомление с Postman**

1. Установка Postman:
   * Скачана последняя версия Postman с официального сайта.
   * Приложение успешно установлено и запущено.
2. Создание первого запроса:
   * Создан новый запрос.
   * URL: https://catfact.ninja/fact.
   * Метод: GET.
   * Ответ сервера:

json

{

"fact": "Cats and kittens should be acquired in pairs whenever possible as cat families interact best in pairs.",

"length": 102

}

**Задание 2: Работа с публичным API**

1. Выбранный API: REST Countries API (https://restcountries.com/v3.1/all).
2. Запрос:
   * Метод: GET.
   * Ответ сервера:

json

1. JSON-ответ REST Countries API имеет стандартизированную структуру, удобную для парсинга.

**Задание 3: Отправка данных с использованием POST**

1. Выбранный API: JSONPlaceholder (https://jsonplaceholder.typicode.com/posts).
2. Запрос:
   * Метод: POST.
   * Тело запроса (JSON):

json

{

"title": "Test Post",

"body": "This is a test post.",

"userId": 1

}

* + Ответ сервера:

json

{

"id": 101

}

Вывод:  
Ответ содержит: все отправленные данные (title, body, userId), новый id: 101 - идентификатор созданного поста.

**Задание 4: Модификация данных с использованием PUT**

1. Выбранный API: JSONPlaceholder (например, https://jsonplaceholder.typicode.com/posts/1).
2. Запрос:
   * Метод: PUT.
   * Тело запроса (JSON):

{

"id": 1,

"title": "Updated Post",

"body": "This post has been updated.",

"userId": 1

}

* + Ответ сервера:

json

{

"id": 1,

"title": "Updated Post",

"body": "This post has been updated.",

"userId": 1

}

Вывод:  
Ответ содержит полные данные обновлённого ресурса:

* id: 1 - идентификатор изменённого поста
* title и body - новые значения полей
* userId: 1 - владелец поста (не изменился)

**Задание 5: Удаление данных с использованием DELETE**

1. Выбранный API: JSONPlaceholder (например, https://jsonplaceholder.typicode.com/posts/1).
2. Запрос:
   * Метод: DELETE.
   * Ответ сервера:

json

{}

Вывод:  
Пустой JSON-объект ({}) — стандартный ответ при успешном удалении.

Подразумевает:

* Ресурс (например, пост с указанным ID) был удален.
* Сервер не возвращает данных, так как они больше не существуют.

**Выводы:**

1. Успешно отработаны основные HTTP-методы (GET, POST, PUT, DELETE) через Postman
2. Изучены форматы JSON-ответов для разных операций:
   * GET - детальные данные
   * POST - созданный ресурс
   * PUT - обновленные данные
   * DELETE - пустой ответ
3. Получены практические навыки работы с REST API:
   * Формирование запросов
   * Анализ ответов
   * Интерпретация статус-кодов
4. Подтверждено понимание принципов REST:
   * Единый интерфейс
   * Стандартные методы
   * Предсказуемые ответы

**Итог:** Работа демонстрирует готовность к взаимодействию с веб-API и может служить основой для более сложных интеграций.