**Федеральное государственное бюджетное образовательное учреждение высшего образования**

**«Белгородский государственный технологический университет им. В.Г.Шухова»**

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

**Методические указания к лабораторной работе № 4. «Скрипты для автоматизации тестирования REST API в Postman»**

по дисциплине «Тестирование программных систем»

Выполнил: ст. группы ПВ-202

Буйвало А.А.

Проверили:

Лебединская Анастасия Александровна

Бабенко Анастасия Александровна

Белгород, 2023

**Лабораторная работа №4**

**«Скрипты для автоматизации тестирования REST API в Postman»**

**Цель работы:**

* 1. Рассмотреть типы скриптов и порядок их запуска;
  2. Рассмотреть скрипты, запускаемые перед запросом, их назначение и основные задачи, решаемые с их помощью;
  3. Рассмотреть скрипты, запускаемые после получения ответа;
  4. Составить коллекцию и обработать ответ от сервера с помощью простых скриптов.

**Задания**

Необходимо создать коллекцию и присвоить ей наименование по следующему шаблону Группа\_ФИО\_ЛабораторнаяРабота\_4\_Задание(номер задания). Коллекции без правильного наименования приняты не будут.

Необходимо выполнить скрипты по следующим заданиям, т.е. должно быть три коллекции:

1. Задание № 1 или № 2;
2. Задание № 3 или № 4;
3. В задании № 5 какой-то один метод, кроме запроса токена

Необходимо оформить краткий отчет о лабораторной работе исходя из ее целей. К отчету прикрепить скриншоты вызовов с открытой вкладкой Tests.

**Выполнение работы**

1. Проверить ,что статус код ответа = 200 ;

pm.test("Status code is 200", *function* () {

pm.response.to.have.status(200);

});

1. Присвоить тело ответа переменной response\_JSON ;

*var* response\_JSON = pm.response.json();

1. Проверить, что name в ответе равно name request (ввести вручную);

pm.test("Проверить, что name в ответе равно name request (ввести вручную);", *function* () {

pm.expect(response\_JSON.name).to.eql("Nastya");

});

1. Проверить, что age в ответе равно age request (ввести вручную);

pm.test("Проверить, что age в ответе равно age request (ввести вручную);", *function* () {

pm.expect(response\_JSON.age).to.eql("20");

});

1. Проверить, что salary в ответе равно salary request (ввести вручную);

pm.test("Проверить, что salary в ответе равно salary request (ввести вручную);", *function* () {

pm.expect(response\_JSON.salary).to.eql(5);

});

1. Присвоить тело запроса переменной request\_JSON ;

*var* request\_JSON = request.data

1. Проверить, что name в ответе равно name request (взять из тела запроса);

pm.test("Проверить, что name в ответе равно name request (взять из тела запроса);", *function* () {

pm.expect(response\_JSON.name).to.eql(request\_JSON.name);

});

1. Проверить, что age в ответе равно age request (взять из тела запроса);

pm.test("Проверить, что age в ответе равно age request (взять из тела запроса);", *function* () {

pm.expect(response\_JSON.age).to.eql(request\_JSON.age);

});

1. Проверить, что salary в ответе равно salary request (взять из тела запроса);

pm.test("Проверить, что salary в ответе равно salary request (взять из тела запроса);", *function* () {

pm.expect(response\_JSON.salary).to.eql(parseInt(request\_JSON.salary));

});

1. Вывести в консоль параметр family из response;

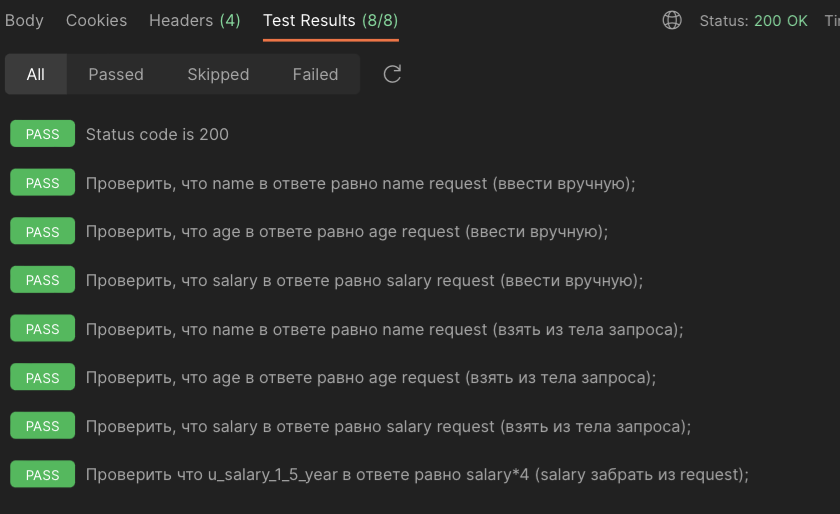
*console*.log(response\_JSON.family)

1. Проверить что u\_salary\_1\_5\_year в ответе равно salary\*4 (salary забрать из request)

pm.test("Проверить, что age в ответе равно age request (ввести вручную);", *function* () {

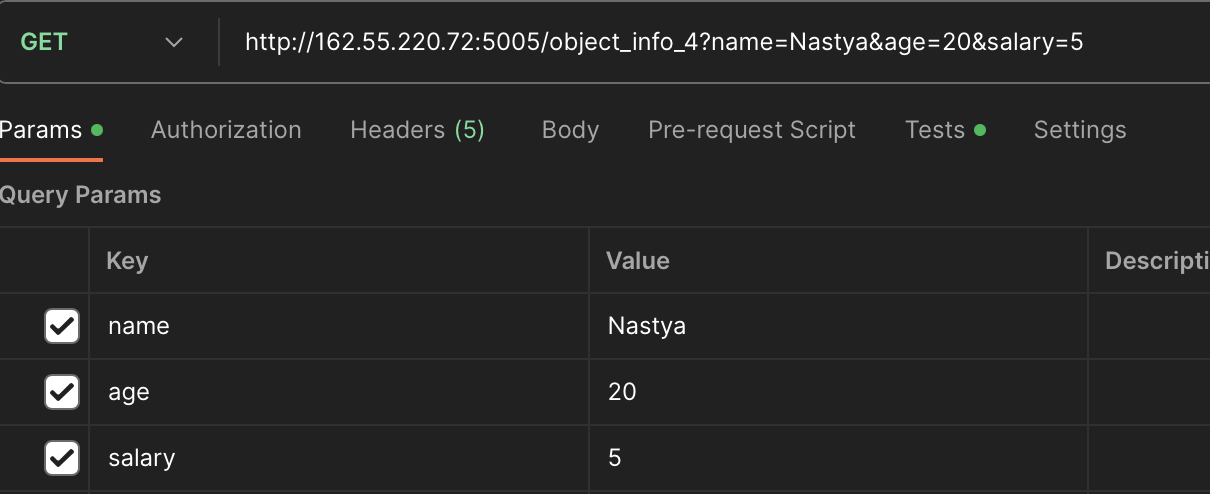
pm.expect(response\_JSON.family.u\_salary\_1\_5\_year).to.eql(request\_JSON.salary\*4);

});



**Задание №3**

Запрос методом GET <https://162.55.220.72:5005/user_info_4> параметры запроса name, age, salary присвоить самостоятельно



1. Проверить, что статус код ответа = 200

pm.test("Status code is 200", *function* () {

pm.response.to.have.status(200);

});

2. Присвоить тело ответа переменной response\_JSON ;

*var* response\_JSON = pm.response.json()

3. Присвоить тело запроса переменной request\_JSON

*var* request\_JSON = pm.request.url.query.toObject();

4. Проверить, что name в ответе равно name s request (name забрать из request.)

pm.test("Test response name with request data", () *=>* {

pm.expect(response\_JSON.name).to.equal(request\_JSON.name);

})

5. Проверить, что age в ответе равно age из request (age забрать из request.)

pm.test("Test response age with request data", () *=>* {

pm.expect(response\_JSON.age).to.equal(parseInt(request\_JSON.age));

})

6. Вывести в консоль параметр salary из request

*console*.log("request salary =", request\_JSON.salary);

7. Вывести в консоль параметр salary из response

*console*.log("response salary =", response\_JSON.salary)

8. Вывести в консоль 0-й элемент параметра salary из response

9. Вывести в консоль 1-й элемент параметра salary параметр salary из response

10. Вывести в консоль 2-й элемент параметра salary параметр salary из response

*console*.log("salary[0] =", response\_JSON.salary[0])

*console*.log("salary[1] =",response\_JSON.salary[1])

*console*.log("salary[2] =",response\_JSON.salary[2])

11. Проверить, что 0-й элемент параметра salary равен salary из request (salary

забрать из request.)

12. Проверить, что 1-й элемент параметра salary равен salary\*2 из request (salary забрать из request.)

13. Проверить, что 2-й элемент параметра salary равен salary\*3 из request (salary забрать из request.)

pm.test("Test response salary[0]", () *=>* {

pm.expect(parseInt(response\_JSON.salary[0])).to.eql(parseInt(request\_JSON.salary));

})

pm.test("Test response salary[1]", () *=>* {

pm.expect(parseInt(response\_JSON.salary[1])).to.eql(parseInt(request\_JSON.salary \* 2));

})

pm.test("Test response salary[2]", () *=>* {

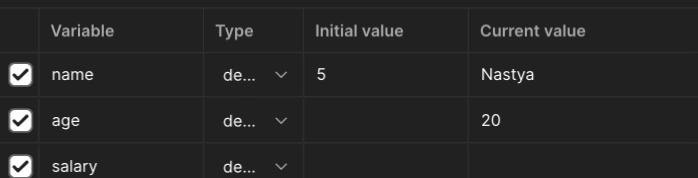
pm.expect(parseInt(response\_JSON.salary[2])).to.eql(parseInt(request\_JSON.salary \* 3));

})

14. Создать в окружении переменную name

15. Создать в окружении переменную age

16. Создать в окружении переменную salary



17. Передать в окружение переменную name

18. Передать в окружение переменную age

19. Передать в окружение переменную salary

pm.environment.set("name", request\_JSON.name)

pm.environment.set("age", request\_JSON.age)

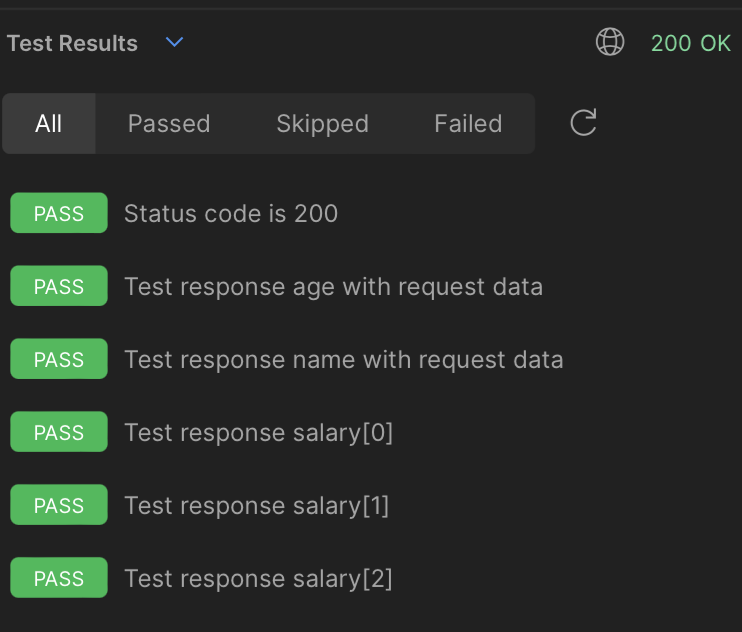
pm.environment.set("salary", request\_JSON.salary)

20. Написать цикл который выведет в консоль по порядку элементы списка из

параметра salary

for(*let* i = 0; i < 3; i++)

*console*.log(response\_JSON.salary[i]);



**Задание №5**

Запрос методом POST <https://162.55.220.72:5005/login>. Необходимо залогиниться login : str , password : str . Полученный токен необходимо передавать в последующие запросы. Используйте окружение.

Отправить запрос методом POST http://162.55.220.72:5005/test\_pet\_info тело запроса отправить RAW JSON { “age” : int , “weight” : int , “name” : “string” , auth\_token

: “string” }. В ответ получено тело { “name”: “string”, “age” : int, “daily\_food” : weight \*

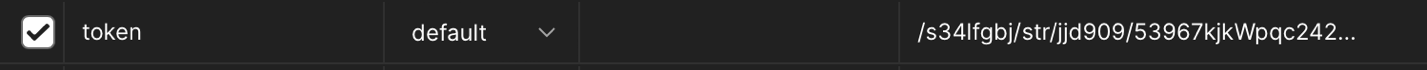
0.012 , “daily\_sleep”: weight \* 2.5 }. Написать тесты :

1. Проверить что статус код ответа = 200 ;

2. Проверка структуры json в ответе ;

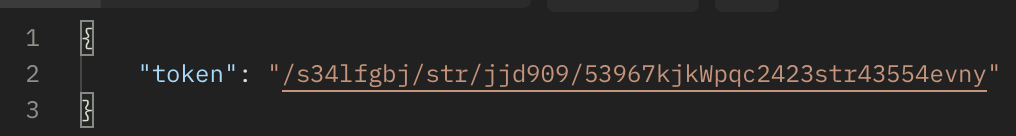
3. В ответе указаны коэффициенты умножения weight, напишите тесты по

проверке правильности результата перемножения на коэффициент.



*var* response\_JSON = pm.response.json()

pm.environment.set("token", response\_JSON.token)



pm.test("Status code is 200", *function* () {

pm.response.to.have.status(200);

});

*const* response\_JSON = pm.response.json();

*const* request\_FORMDATA = pm.request.body.formdata;

*const* schema = {

"items": {

"age": {

"type": "string"

},

"daily\_food": {

"type": "number"

},

"daily\_sleep": {

"type": "number"

},

"name": {

"type": "string"

}

}

}

pm.test("Scheme validation", () *=>* {

pm.expect(tv4.validate(response\_JSON, schema)).to.be.true;

})

*var* request\_JSON = request.data

pm.test("daily\_food", () *=>* {

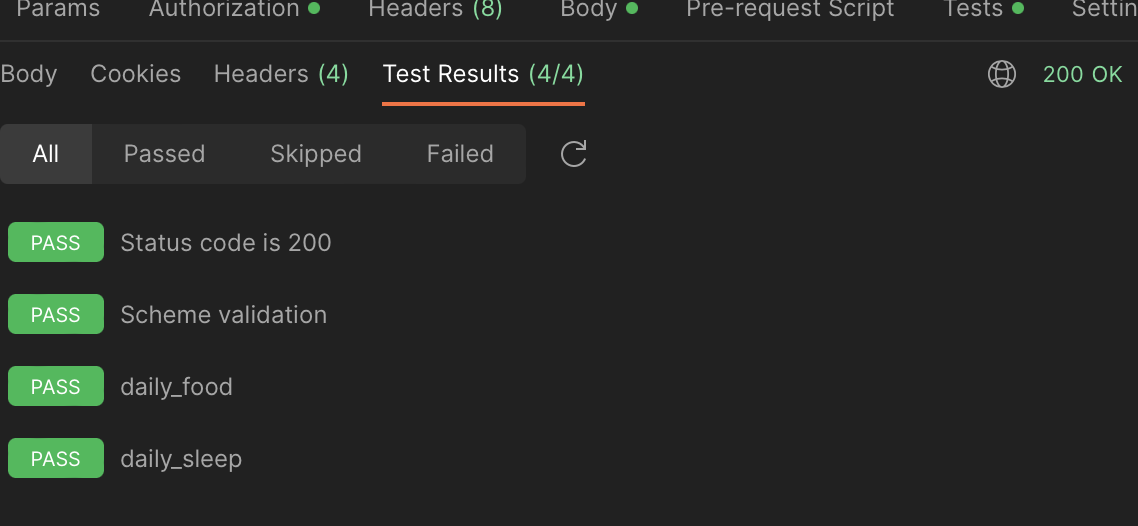
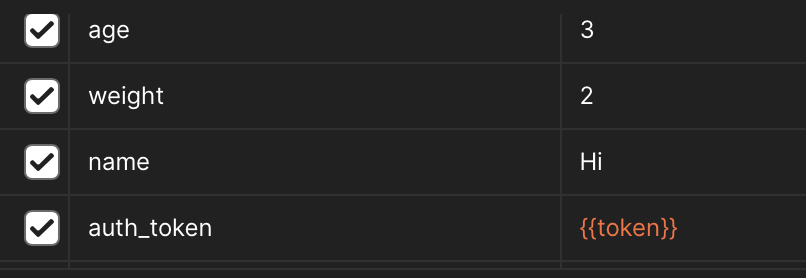
pm.expect(response\_JSON.daily\_food).to.eql(request\_JSON.weight \* 0.012);

})

pm.test("daily\_sleep", () *=>* {

pm.expect(response\_JSON.daily\_sleep).to.eql(request\_JSON.weight \* 2.5);

})



**Вывод:** в ходе выполнения данной лабораторной работы мною были изучены возможности использования скриптов в сценариях тестирования Postman. Рассмотрены скрипты перед запросом, скрипты после получения ответа.