ДЗ №2 Postman

1. Необходимо залогиниться
POST
http://162.55.220.72:5005/**login**
login : str (кроме /)
password : str

Приходящий токен необходимо передать во все остальные запросы.

```
{
    "token": "/s34lfgbj/str/jjd909/41022kjkWpqc516None102857evny"
}
```

AUTO TEST

```
//1. Статус код 200
pm.test("Test 1_Check status code 200", function () {
    pm.response.to.have.status(200);
});

//2.Создание переменной "token" и добавление в environment
pm.test("Test 2_Set variable with name token and add it to environmen", function () {
    var jsonData = pm.response.json();
    console.log('This is jsonData', jsonData)
    pm.environment.set("token", jsonData.token);
});
```

2.
http://162.55.220.72:5005/user_info
req.
POST
age: int
salary: int
name: str
auth_token

resp.

```
{'start_qa_salary':salary,
 'qa_salary_after_6_months': salary * 2,
 'qa_salary_after_12_months': salary * 2.9,
 'person': {'u_name':[user_name, salary, age],
                        'u_age':age,

'u_salary_1.5_year': salary * 4}
                        }
```

Тесты:
1) Статус код 200
2) Проверка структуры json в ответе.
3) В ответе указаны коэффициенты умножения salary, напишите тесты по проверке правильности результата перемножения на коэффициент.
4) Достать значение из поля 'u_salary_1.5_year' и передать в поле salary запроса

AUTO TEST

```
//1. Статус код 200
pm.test("Test 1_Check status code 200", function () {
    pm.response.to.have.status(200);
});

//2. Проверка структуры json в ответе.

let person = JSON.parse(responseBody);
console.log('Object', person);

var schema = {
  "type": "object",
  "properties": {
    "person": {
      "type": "object",
      "properties": {
        "u_age": {
          "type": "integer"
        },
        "u_name": {
          "type": "array",
          "items": [
            {
              "type": "string"
            },
            {
              "type": "integer"
            },
            {
              "type": "integer"
```

http://~~116.203.27.46:5002~~ (http://~~188.130.138.105:5001~~/new_data)/get_test_user
```
            }
          ]
        },
        "u_salary_1_5_year": {
          "type": "integer"
        }
      },
      "required": [
        "u_age",
        "u_name",
        "u_salary_1_5_year"
      ]
    },
    "qa_salary_after_12_months": {
      "type": "number"
    },
    "qa_salary_after_6_months": {
      "type": "integer"
    },
    "start_qa_salary": {
      "type": "integer"
    }
  },
  "required": [
    "person",
    "qa_salary_after_12_months",
    "qa_salary_after_6_months",
    "start_qa_salary"
  ]
}
        pm.test('Test 2_Check jsonSchema in response is valid', function() {
        pm.response.to.have.jsonSchema(schema);
});


//3.   Проверка правильности вычислений с учётом указанных коэффициентов.

// "Использование "SNIPPETS: Response body JSON value  check
//pm.test("Your test name", function () {
   // var jsonData = pm.response.json();
   // pm.expect(jsonData.value).to.eql(100);
//});
//3.1. Проверка коэффициента 2.9 для qa_salary_after_12_month
pm.test("Test 3_Check index for qa_salary_after_12_months", function () {
var total = pm.response.json();
var total1 = JSON.parse(request.data);
console.log(total1);
var total2 = total1.salary * 2.9;
    pm.expect(total.qa_salary_after_12_months).to.eql(total2)
});
```
 |

	//3.2. Проверка коэффициента 2.0 for qa_salary_after_6_months pm.**test**("Test 4_Check index for qa_salary_after_6_months", **function** () { var total3 **=** pm.response.json(); pm.expect(total3.qa_salary_after_6_months).to.eql(20000); }); **//3.3. Проверка коэффициента 4 для u_salary_1_5_year** pm.**test**("Test 5_Check index for u_salary_1_5_year", **function** () { var total4 **=** pm.response.json(); console.**log**(total4); var total5 **=** total4.start_qa_salary ***4; console.**log**(total5); pm.expect(total4.person.u_salary_1_5_year).to.eql(total5); }); **//4. Достать значение из поля "start_qa_salary" и передать в поле salary запроса http://162.55.220.72:5005/new_data)/get_test_user** pm.**test**("Test 6_Set variable with name salary_base and add it to en vironmen", **function** () { var jsonData **=** pm.response.json(); console.**log**('This is jsonData', jsonData) var salary_base **=** Number(jsonData.start_qa_salary); console.**log**('value of salary_base', salary_base); pm.environment.**set**("salary_base", salary_base); }); //Создание переменной "name" и добавление в environment pm.**test**("Test 7_Set variable age and add it to environment", **function** () { var jsonData **=** pm.response.json(); console.**log**('This is jsonData_new', jsonData) var age **=** Number(jsonData.person.u_age); console.**log**('value of age', age); pm.environment.**set**("age", age); }); //Создание переменной "name" и добавление в environment pm.**test**("Test 8_Set variable name and add it to environment", **function** () { var jsonData **=** pm.response.json(); console.**log**('This is jsonData_name', jsonData) var name **=** (jsonData.person.u_name[0]); console.**log**('value of name', name); pm.environment.**set**("name", name); });

	//Создание переменной "weight" и добавление в environment pm.**test**("Test 9_Set variable weight and add it to environment, **function** () { var weight **=** 53; pm.environment.**set**("weight", weight); console.**log**('value of weight', weight); });
3. http://162.55.220.72:5005/**new_data** req. POST age: int salary: int name: str auth_token Resp. {'name':name, 'age': int(age), 'salary': [salary, str(salary*2), str(salary*3)]} Тесты: 1) Статус код 200 2) Проверка структуры json в ответе. 3) В ответе указаны коэффициенты умножения salary, напишите тесты по проверке правильности результата перемножения на коэффициент. 4) проверить, что 2-й элемент массива salary больше 1-го и 0-го	**AUTO TEST** **//1. Статус код 200** pm.**test**("Test 1_Check status code 200", **function** () { pm.response.to.have.status(200); }); **//2. Проверка структуры json в ответе.** let person **=** JSON.**parse**(responseBody); console.**log**('Object', person); var schema **=** { "type" : "object", "properties" : { "age" : {'description' : 'Age in year',"type" : "integer"}, "name" : {"type" : "string"}, "salary" : { "type" : "array", "items": [{"description" : 'Salary in Euro', "type": "integer" }, { 'description' : 'Salary*2 in Euro', "type": "string" }, { 'description' : 'Salary*3 in Euro', "type": "string" }] } }, "required" : ["age","name","salary"], }; pm.**test**('Test 2_Check jsonSchema in response', **function**() { pm.response.to.have.jsonSchema(schema) }); **//3. В ответе указаны коэффициенты умножения salary, напишите тесты по проверке правильности результата перемножения на коэффициент.** //3.1. Проверка коэффициента правильности вычислений salary*2 pm.**test**("Test 3_Check value salary at index salary*2", **function** () { var salary_0 **=** pm.response.json(); console.**log**('This is array of data', salary_0); var salary1 **=** salary_0.salary[1]; console.**log**('Value of calculation salary*2 from array', salary1);

	pm.expect(salary_0.salary[1]).to.eql(salary1); console.**log**('Check calculation salary*2', salary_0.salary[1]); **//3.2. Проверка коэффициента правильности вычислений salary*3** pm.**test**("Test 4_Check value salary at index salary*3", **function** () { var salary_1 **=** pm.response.json(); console.**log**('This is array of data', salary_1); var salary3 **=** salary_1.salary[2]; console.**log**('Value of calculation salary*3 from array', salary3); pm.expect(salary_1.salary[2]).to.eql(salary3); console.**log**('Check calculation salary*3', salary_1.salary[2]); **//4. Проверить, что 2-й элемент массива salary больше 1-го и 0-го** pm.**test**('Test 5_Verifying that an array [2] is larger then [1]', **function** () { pm.expect(**parseInt**(salary_1.salary[2])).to.be.above(**parseInt**(salary_0.salary[1])) pm.**test**('Test 6_Verifying that an array [2] is larger then [0]', **function** () { var salary4 **=** pm.response.json(); console.**log**('array', salary4); var salary5 **=** salary4.salary[0]; console.**log**('Value from array [0]', salary5); pm.expect(**parseInt**(salary_1.salary[2])).to.be.above(**parseInt**(salary4.salary[0])) console.**log**('Check value[0] in array', salary4.salary[0] }) }) }) });
4. http://162.55.220.72:5005**/test_pet_info** req. POST age: int weight: int name: str auth_token Resp. {'name': name, 'age': age, 'daily_food':weight * 0.012, 'daily_sleep': weight * 2.5}	**AUTO TEST** **//1. Статус код 200** pm.**test**("Test 1_Check status code 200", **function** () { pm.response.to.have.status(200); }); **//2. Проверка структуры json в ответе.** var schema **=** { "type": "object", "properties": { "age" : {'description' : 'Age in year',"type" : "integer"}, "daily_food" : {"description" :"Daily food in kg", "type" : "number"}, "daily_sleep": {'description' : "Daily sleep in minute","type" : "number"},

Тесты: 1) Статус код 200 2) Проверка структуры json в ответе. 3) В ответе указаны коэффициенты умножения weight, напишите тесты по проверке правильности результата перемножения на коэффициент.	"name" : {"type" : "string"} }, "required" : ["age","daily_food","daily_sleep","name"], }; pm.**test**('Test 2_Check jsonSchema in response', **function**() { pm.response.to.have.jsonSchema(schema) }); **//3. В ответе указаны коэффициенты умножения weight, напишите тесты по проверке правильности результата перемножения на коэффициент.** //3.1. Проверка результате умножения 'daily_food':weight * 0.012 pm.**test**("Test 3_Verifying calculation daily_food", **function** () { var weight2 **=** pm.response.json(); console.**log**('Value output', weight2); var weight3 **=** weight2.daily_food; console.**log**('Output value of the daily_food', weight3); //получение переменной 'weight'из окружения pm.environment.**get**('weight'); console.**log**(pm.environment.**get**('weight')); var weight1 **=** pm.environment.**get**('weight') ***** 0.012; console.**log**('The value had calculated by formula', weight1); pm.expect(weight3).to.eql(weight1) }); **//3.2. Проверка результате умножения 'daily_sleep':weight * 2.5** pm.**test**("Test 4_Verifying calculation daily_sleep", **function** () { var weight2 **=** pm.response.json(); console.**log**('Value output', weight2); var weight3 **=** weight2.daily_sleep; console.**log**('Output value of the daily_sleep', weight3); //получение переменной 'weight'из окружения pm.environment.**get**('weight'); console.**log**('Value from environment is', pm.environment.**get**('weight')); var weight4 **=** pm.environment.**get**('weight') ***** 2.5; console.**log**('The value had calculated by formula', weight4); pm.expect(weight3).to.eql(weight4) });
5) http://162.55.220.72:5005/**get_test_user** req. POST age: int salary: int	**AUTO TEST** **//1. Статус код 200** pm.**test**("Test 1_Check status code 200", **function** () { pm.response.to.have.status(200); });

name: str
auth_token

Resp.
{'name': name,
'age':age,
'salary': salary,
'family':{'children':[['Alex', 24],['Kate', 12]],
'u_salary_1.5_year': salary * 4}
}

Тесты:
1) Статус код 200
2) Проверка структуры json в ответе.
3) Проверить что значение поля name = значению переменной name из окружения
4) Проверить что значение поля age в ответе соответсвует отправленному в запросе значению поля age

//2. Проверка структуры json в ответе.

```
//описание схемы модели
var schema = {
    //указание типа объекта
          "type" : "object",
          //описание свойств
            "properties" : {
         "age" : {'description' : 'Age in year',"type" : "string"},
          // вложенный объект с массивом данных children
"family": {
   "type" : "object",
       "properties": {
      "children": {"type" : "array",
       "items": [
          {
           "type": "array",
            "items":  [
            {"description" : 'name_son', "type": "string"},
              {"description" : "age_son", "type": "integer"}
            ]
          },
          {
            "type": "array",
            "items": [
                {"description" : 'name_daughter', "type": "string"},
             {"description" : "age_daughter", "type": "integer"}
            ]
          }
       ]
      },
         "u_salary_1_5_year" : {  'description' : 'Salary in Euro', "type" :
 "integer"}
       },
       //обязательные свойства
      "required" : ["children","u_salary_1_5_year"]
},
          "name" : {"type" : "string"},
         "salary" : {  'description' : 'Salary in Euro', "type" : "integer"}
            },
//обязательные свойства
       "required" : ["name","age","salary", "family"]

 };

pm.test('Test 2_Check jsonSchema in response', function() {
    pm.response.to.have.jsonSchema(schema)
  });
```

//3) Проверить что значение поля name = значению переменной name из окружения

```
pm.test("Test 3_Verifying value “name” = value “name” from environment", function
```

	() { var jsonData = pm.response.json(); console.log('This is jsonData_name', jsonData); var name1 = jsonData.name; console.log('Value name1 is', name1); //получение переменной 'name' из окружения var name2 = pm.environment.get('name'); console.log('The value name2 from enveronment is', name2); pm.expect(name1).to.eql(name2) }); **//4) Проверить что значение поля age в ответе соответствуе т отправленному в запросе значению поля age** pm.test("Test 4_Verifying value age_response = value age_request ", function () { var jsonData = pm.response.json(); console.log('This is jsonData', jsonData); var age1 = jsonData.age; console.log('Value age_response is', age1); //получение переменной 'age_request' из form-data var age2 = request.data["age"]; console.log('The value age_request is', age2); pm.expect(age1).to.eql(age2); });
6. http://162.55.220.72:5005/**currency** req. POST auth_token Resp. Передаётся список массив объектов. [{"Cur_Abbreviation": str, "Cur_ID": int, "Cur_Name": str } … {"Cur_Abbreviation": str, "Cur_ID": int, "Cur_Name": str }] Тесты: 1) Можете взять любой объект из присланного списка, используйте js random.	**AUTO TEST** //1) Взять любой объект из присланного списка, с помощью js random. В объекте забрать Cur_ID и передать через окружение в следующий запрос. //Получение тела ответа let jsonData = JSON.parse(responseBody); console.log('Object_list', jsonData); //Определяем длину массива, и какой элемент берём для дальнейшей передачи его в окружение console.log('Object length', jsonData.length); var i = jsonData.length; console.log('1=', i); //функция Random возвращает значение от 0 до 1, а так как в массиве больше 1, то используется следующая функция (https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Global_Objects/Math/random) function getRandomInt(i) { return Math.floor(Math.random() * i); }

В объекте возьмите Cur_ID и передать через окружение в следующий запрос.	//Выбор произвольного объекта и извлечение из него значения Cur_ID console.log('Select random values: =', jsonData[getRandomInt(i)]); console.log('This is value Cur_ID random =', jsonData[getRandomInt(i)].Cur_ID); //Запись произвольного значения Cur_ID в переменую Curr_code и передача её в окружение pm.environment.set("Curr_code", jsonData[getRandomInt(i)].Cur_ID); //Создание пустого списка let curr_list = []; console.log('Array curr_list', curr_list); //Использование цикла для добавление в пустой список curr_list значения Cur_Id for (n = 0; n < jsonData.length; n++){ console.log('Reciving value currency from Object', jsonData[n]); curr_list.push(jsonData[n].Cur_ID); console.log('Add Cur_ID to the array curr_list', curr_list); }; //длина массива console.log('Length curr_list', jsonData.length);
7. http://162.55.220.72:5005/curr_byn req. POST auth_token curr_code: int Resp. { "Cur_Abbreviation": str "Cur_ID": int, "Cur_Name": str, "Cur_OfficialRate": float, "Cur_Scale": int, "Date": str } Тесты: 1) Статус код 200 2) Проверка структуры json в ответе.	**AUTO TEST** //1. Статус код 200 //pm.test("Test 1_Check status code 200", function () { //pm.response.to.have.status(200); //}); //2. Проверка структуры json в ответе. let profile = JSON.parse(responseBody); console.log('Object', profile); //описание схемы модели var schema = { //тип объекта "type" : "object", //описание свойств "properties" : { " Cur_Abbreviation " : { 'description' : ' Currency code ', "type" : "string"}, " Cur_ID " : { 'description' : ' Currency code ', "type" : "integer"}, "Cur_Name" : { 'description' : 'Currency Name', "type" : "string"}, "Cur_OfficialRate" : { 'description' : 'Official Rate of a currency', "type" : "number"}, "Cur_Scale" : { "type" : "integer"}, "Date" : { 'description' : ' current data in the response', "type" : "string"} }, //обязательные свойства

	"required" : ["Cur_Abbreviation","Cur_ID","Cur_Name","Cur_OfficialRate","Cur_Scale","Date"], }; pm.**test**('Test 2_Check jsonSchema in response is valid', **function**() { pm.response.to.have.jsonSchema(schema); });