Дано:

Площадка open api https://dummyapi.io/docs

Задание:

Создайте в Postman коллекцию запросов, которая включает в себя:

- Получение списка пользователей;
- Создание пользователя;
- Получение пользователя по id;
- Обновление пользователя;
- Удаление пользователя;
- Создание негативных проверок на статус-коды: 400, 403, 404

Перейти на сайт: https://dummyapi.io/docs

Найти блок: Getting Started

Зарегистрироваться, например, через Google-аккаунт.

Получить API-ID Header.

В запросах используются следующие переменные коллекции:

{{URL}}: https://dummyapi.io/data/v1/user

{{app_id}}: 64edf0eb01c0218de1b489a5

{{user_id}}: 6512cf788224dd4f83a58759

1) Отправляем GET запрос на получение списка пользователей

URL запроса: {{URL}}/user

При этом в Header запроса вставляем api-id, полученные ранее.

2) POST запрос

URL запроса: {{URL}}/user/create

Тело запроса:

```
{
    "lastName": "{{$randomLastName}}",
    "firstName": "{{$randomFirstName}}",
    "email": "{{$randomEmail}}"
}
```

Тело ответа:

```
{
```

```
    "id": "6512cf788224dd4f83a58759",
    "firstName": "Kailee",
    "lastName": "Rice",
    "email": "destiney6@yahoo.com",
    "registerDate": "2023-09-26T12:32:56.187Z",
    "updatedDate": "2023-09-26T12:32:56.187Z"
}
```

Во вкладке Tests распарсим JSON и получим ID пользователя:

```
var jsonData = JSON.parse(responseBody);
console.log(jsonData);
var user_ID = jsonData.id;
console.log(user_ID);
```

Запишем id в переменную коллекции:

```
pm.collectionVariables.set("user_ID", user_ID);
```

Получили ID пользователя, записали в переменную.

Так же в этом запросе проверим совпадают ли Имя, фамилии и email пользователя в теле запроса с телом ответа:

```
var FirstName_req = JSON.parse(request.data).firstName;
var FirstName_resp = JSON.parse(responseBody).firstName;

pm.test("Проверить, что FirstName запроса равен FirstName из ответа", function () {
    pm.expect(FirstName_req).to.eql(FirstName_resp);
    console.log("FirstName_req = ", FirstName_req);
    console.log("FirstName_resp = ", FirstName_resp);
});

var LastName_req = JSON.parse(request.data).lastName;
var LastName_resp = JSON.parse(responseBody).lastName;

pm.test("Проверить, что LastName запроса равен LastName из ответа", function () {
    pm.expect(LastName_req).to.eql(LastName_resp);
    console.log("LastName_req = ", LastName_req);
    console.log("LastName_resp = ", LastName_resp);
});

var email_req = JSON.parse(request.data).email;
var email_resp = JSON.parse(responseBody).email;

pm.test("Проверить, что email запроса равен email из ответа", function () {
    pm.expect(email_req).to.eql(email_resp);
    console.log("email_req = ", email_req);
    console.log("email_resp = ", email_resp);
});
```

3) PUT запрос на обновление данных пользователя

URL запроса: {{URL}}/user/{{user_ID}}

Тело запроса:


```
{
    "firstName": "Emily"

}
```


Обновим имя

Тело ответа


```
{
    "id": "6512cf788224dd4f83a58759",
    "firstName": "Emily",
    "lastName": "Rice",
    "email": "destiney6@yahoo.com",
    "registerDate": "2023-09-26T12:32:56.187Z",
    "updatedDate": "2023-09-26T12:39:22.475Z"
}
```


4) DELETE запрос, удаление пользователя

URL запроса: {{URL}}/user/{{user_ID}}

Тело ответа: {


```
    "error": "RESOURCE_NOT_FOUND"
}
```


Негативные проверки

1) 403 – доступ запрещён

Метод POST, создаем пользователя

URL запроса: {{URL}}/user/create
В Headers убираем app-id

Тело ответа:


```
{
    "error": "APP_ID_MISSING"
}
```


2) 404 – пользователь не найден

Метод GET, выбираем пользователя с несуществующим id

URL запроса: {{URL}}/12345

Тело ответа:


```
{
    "error": "PATH_NOT_FOUND"
}
```


3) 400 – плохой запрос

Метод GET , создаем пользователя

URL запроса: {{URL}}/user/create

Тело ответа:


```
{
    "error": "PARAMS_NOT_VALID"
}
```