

Вступ до Postman та тестування API для початківців

Зміст

Розділ I. ВСТУП ДО POSTMAN.....	3
1.1 Як встановити postman.....	3
1.2 Як оновити postman.....	3
1.3 Вступ до API.....	4
1.4 Протокол HTTP.....	5
Розділ 2. ФОРМУВАННЯ ЗАПИТІВ В POSTMAN.....	7
2.1 Колекції postman.....	7
2.2 Параметри Query.....	9
2.3 Змінні Path.....	11
2.4 Авторизація API.....	12
2.5 Проблема статус-кодів HTTP.....	13
2.6 HTTP заголовки.....	14
2.7 JSON формат.....	14
2.8 GET vs POST.....	15
2.9 В яких випадках Postman не використовується.....	16
2.10 Метод запитів PATCH.....	16
2.11 Метод запитів DELETE.....	16
Розділ 3. ПІДГОТОВКА ДО АВТОМАТИЗАЦІЇ.....	17
3.1 Основи автоматизації.....	17
3.2 Перший API тест.....	17
3.3 Змінні postman.....	19
3.4 Робота зі змінними Postman зі скриптів.....	21
Розділ 4. ЗАПУСК АВТОМАТИЗОВАНОГО ЗБОРУ.....	22
4.1 Collection Runner.....	22
4.2 Моніторинг Postman.....	23
4.3 Newman – інструмент CLI в Postman.....	25
БІБЛІОГРАФІЯ:.....	27

Розділ I. ВСТУП ДО POSTMAN

1.1 Як встановити postman

Postman є безкоштовним для використання. Є два способи, як ним користуватися:

- Перейти в браузері за посиланням [postman.com](https://www.postman.com)
- Як автономний додаток, який вам треба завантажити на свій комп'ютер(доступний для Windows, macOS, або Linux)

Не використовуйте застарілий додаток Google Chrome

Перейдіть на [postman.com](https://www.postman.com) та створіть свій акаунт.(Рис. 1.1)

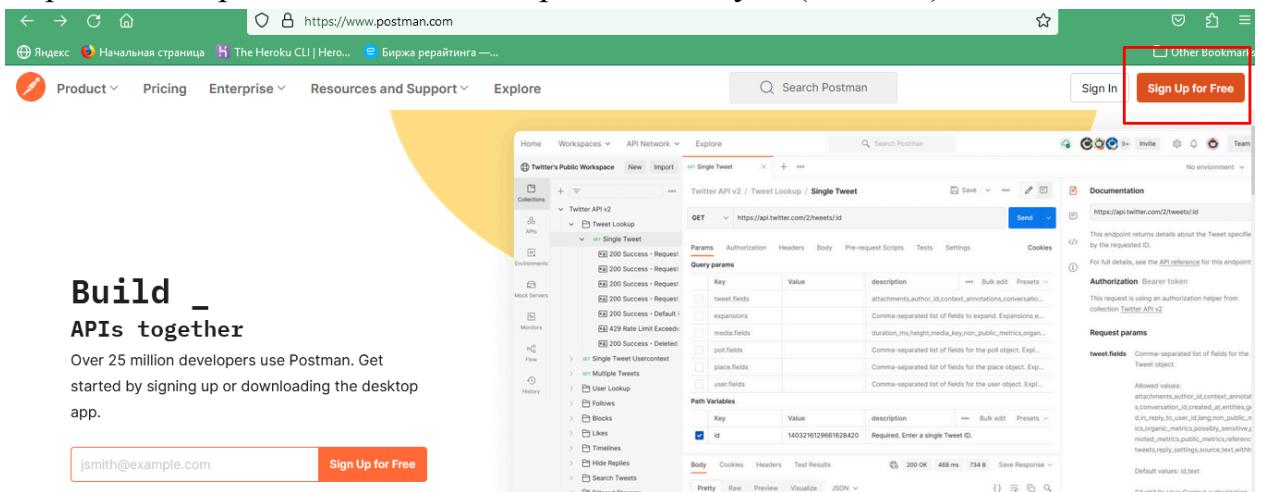


Рис. 1.1 Реєстрація на сайті [postman.com](https://www.postman.com)

Після реєстрації переходимо до графи workspaces та обираємо my workspace. (Рис. 1.2). В цьому робочому середовищі будемо працювати.

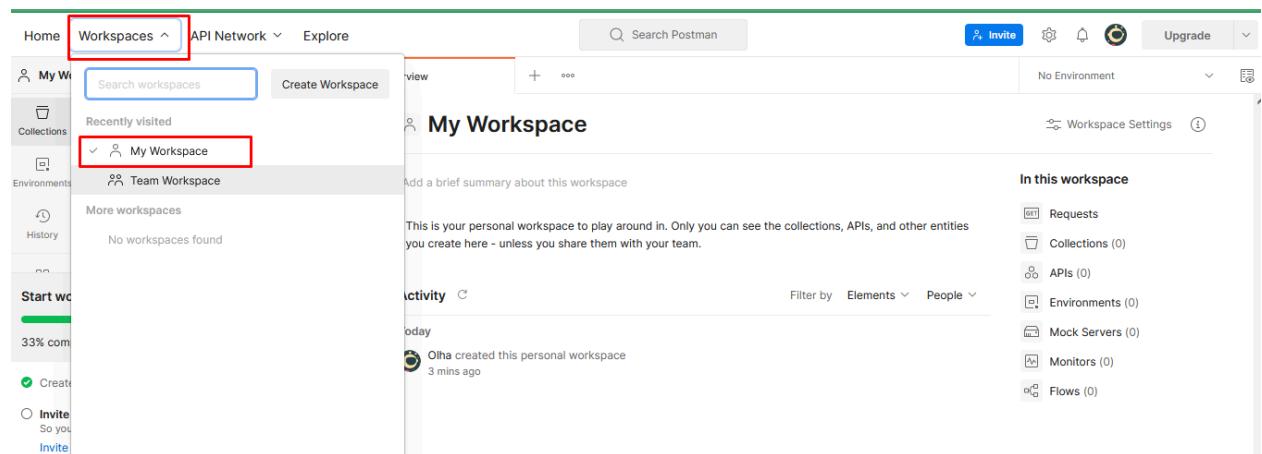


Рис.1.2 Робоче середовище

1.2 Як оновити postman

Оновлення postman надзвичайно важливо, так як нова версія може включати виправлення помилок, нові функції, або важливі оновлення безпеки.

Вам прийде повідомлення, коли вийде нова версія, але ви також можете перевірити наявність нової версії в меню додатку. (Рис.1.3)

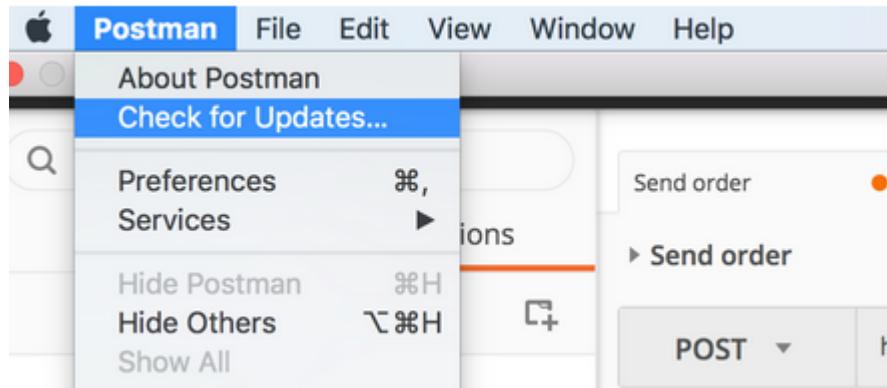


Рис. 1.3 Перевірка оновлених версій

1.3 Вступ до API

Інтерфейс прикладного програмування або API це готові конструкції мови програмування, що дозволяють розробнику будувати складну функціональність із меншими зусиллями. Вони "приховують" складніший код від програміста, забезпечуючи простоту використання.

Додати API можна в розділі My Workspace. (Рис.1.4)

A screenshot of the Postman 'My Workspace' section. The left side shows a summary area with a placeholder 'Add a brief summary about this workspace'. Below it is an 'Activity' feed showing recent events: 'Olha deleted the New API API just now', 'Olha added the New API API 5 mins ago', and 'Olha created this personal workspace 29 mins ago'. The right side is titled 'In this workspace' and lists various entities: Requests, Collections (0), APIs (0) (which is highlighted with a red box), Environments (0), Mock Servers (0), Monitors (0), and Flows (0). A 'Create new API' button is located at the top right of the list. There are also '+' and '-' buttons for managing collections.

Рис. 1.4 Створити API

У вікні, яке відкриється додати посилання на запит. Для цього натисніть на плюс. (Рис.1.5)



Рис. 1.5

Оберіть метод GET та вставте посилання. Рис. 1.6

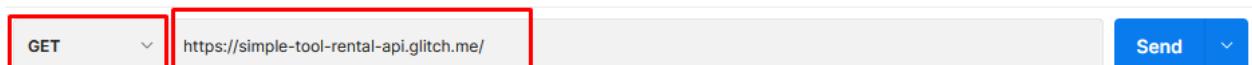


Рис. 1.6

Для того, щоб повернути статус API додайте **/status** до посилання на запит та натисніть «відправити»(Рис. 7)



Рис. 1.7

За кілька секунд ви отримаєте відповідь. 'UP' означає, що API працює правильно та ми можемо працювати з цим API. (Рис 1.8)



Рис. 1.8

Рис.5 Додання API посилання

1.4 Протокол HTTP

Розглянемо, що відбулося, коли ми отримали статус API. Postman відправив запит на сервер. Сервер отримав запит та відправив дані postman. Для того, аби така комунікація відбулась, ми використали протокол HTTP. HTTP – це протокол, який дає можливість клієнту(в даному випадку Postman) та серверу, який виконує API, комунікувати. HTTPS – це захищена та зашифрована версія HTTP, тому краще використовувати HTTPS. (Рис. 1.9)

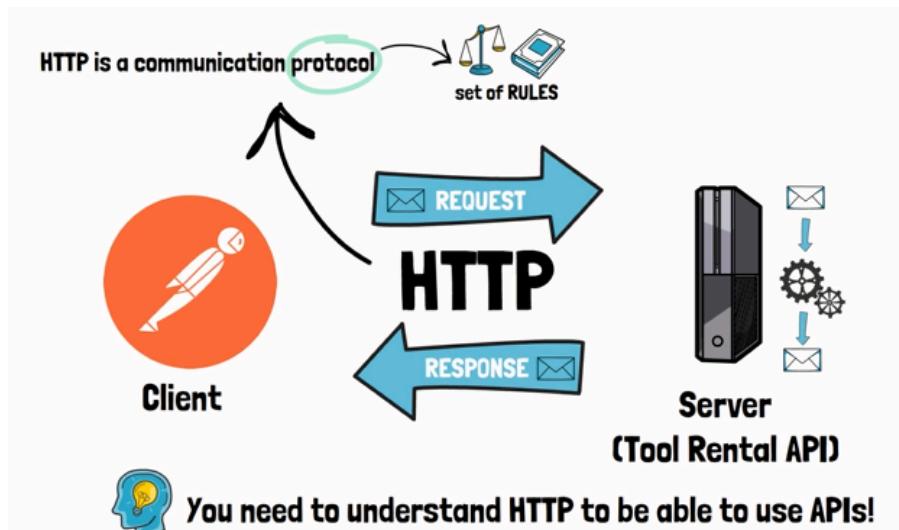


Рис. 1.9

HTTP запит містить:

- Метод запиту
- Адресу / URL
- Хедери
- Баді

HTTP відповідь містить:

- Статус коду
- Хедери
- Баді

Розділ 2. ФОРМУВАННЯ ЗАПИТІВ В POSTMAN

2.1 Колекції postman

Колекція – це як папка, яка зберігає всі запити, які зазвичай відносяться до API або якось пов’язані один з одним. Для того, щоб створити колекцію, треба зберегти API, а у вікні, яке відкриється вибрати «нова колекція» або «створити колекцію». (Рис. 2.1)

SAVE REQUEST

Request name

`https://simple-tool-rental-api.glitch.me/status`

Add description

Save to Select a collection/folder

Search for collection or folder

New API

New Collection

Save

Cancel

Рис. 2.1

Назвемо колекцію Tool Rental API та натиснемо «створити» та зберігаємо запит. (Рис. 2.2, Рис.2.3)

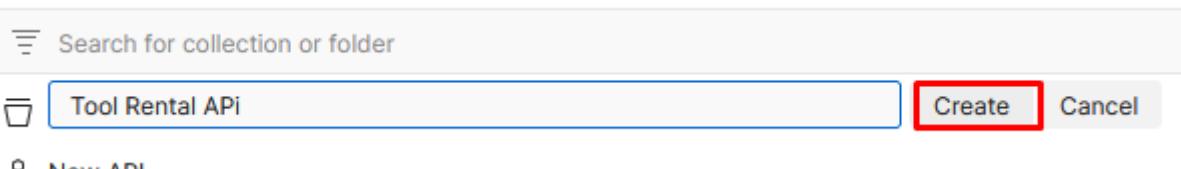


Рис. 2.2

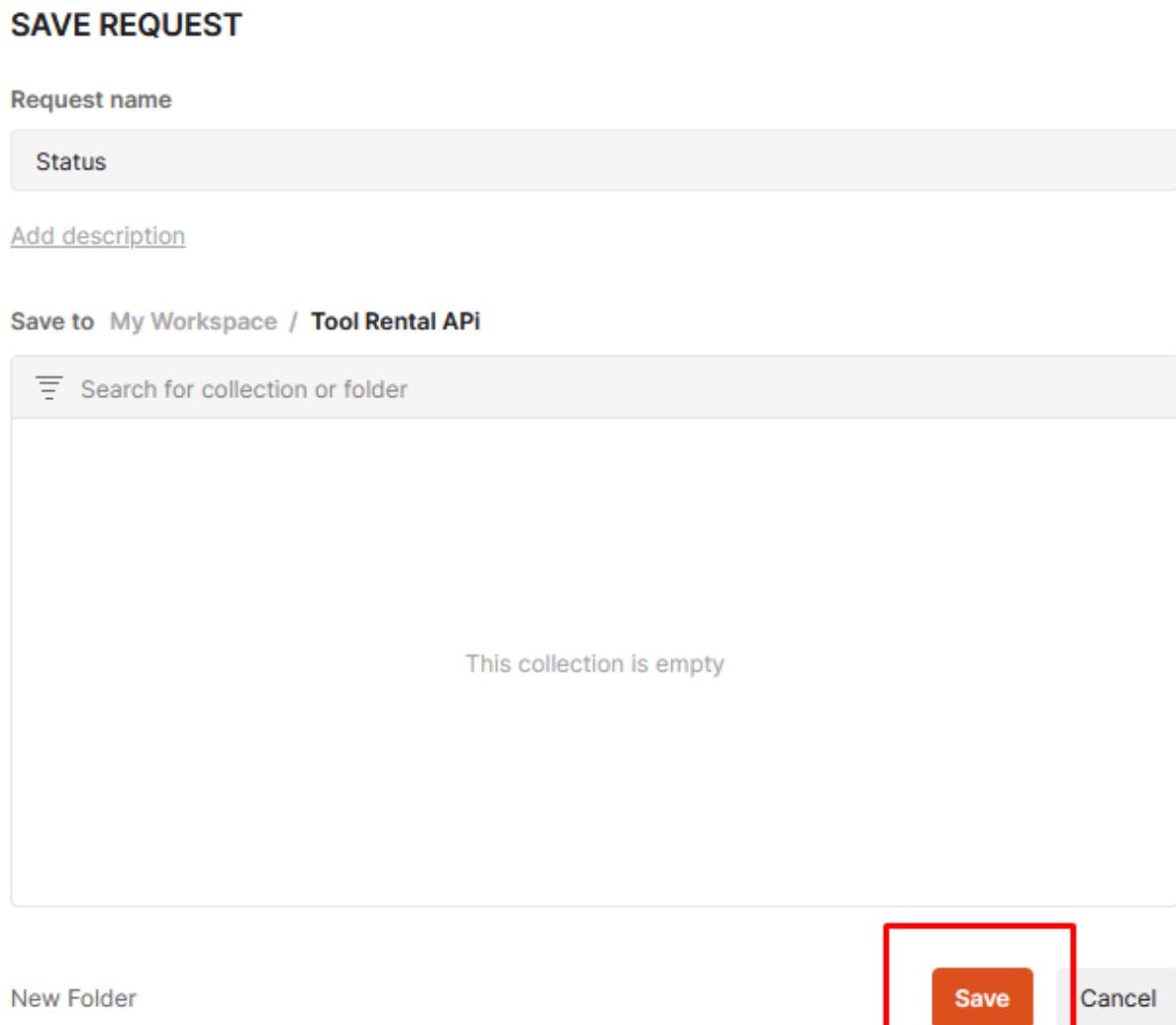


Рис. 2.3

Працюючи з запитами Postman, можна використати концепцію базової URL, що допоможе зробити запит конфігураційним. Для цього помістимо посилання в змінну та дамо змінній ім'я, а в графі «область видимості» обираємо нашу колекцію. Створити змінну можна лише, якщо запит збережено.(Рис. 2.4)

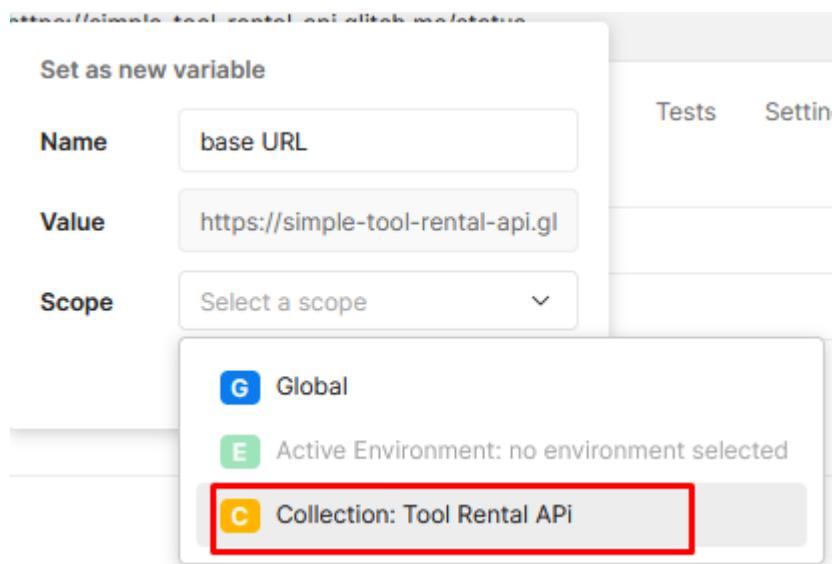


Рис. 2.4

Після цього адреса зміниться. Імя змінної буде знаходитися між двома фігурними дужками. Це синтакс для змінної в Postman . Запит буде працювати правильно. (Рис. 2.5)

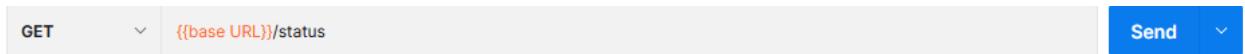


Рис. 2.5

Якщо перейти до вкладки «змінні», ми побачимо, що в кожної змінної є початкове та поточне значення. (Рис. 2.6)

	Variable	Initial value	Current value
<input checked="" type="checkbox"/>	base URL	https://simple-tool-rental-api.glitch.me	https://simple-tool-rental-api.glitch.me
Add new variable			

Рис. 2.6

Початкове значення:

- Використовується Postman при підтверджені запиту
- Приватне для вас / вашого акаунту Postman
- Немає можливості поділитися з іншими

Поточне значення:

- Можна змінювати, коли ділитися колекцією
- Не використовується застосунком Postman

2.2 Параметри Query

Параметри Query – один із способів відправити дані в API. Вони можуть бути обовязковими або опціональними (згідно з документацією). Якщо змінна не

збережена, вона підсвічується червоним кольором та при наведенні видно, що змінна не вирішена. В такому випадку відправити дані неможливо.. (Рис. 2.7)

GET <{{base URL}}/tools>

Send 

Params Authorization Headers (5) Body Pre-request Script Tests Settings Cookies

Query Params

Key	Value	Description	...	Bulk Edit
Key	Value	Description		

Response



Could not send request

Рис. 2.7

Щоб вирішити проблему, достатньо зберегти змінну. Змініть назву та помістіть її в існуючу колекцію, або створіть нову. Дані прийдуть в форматі JSON. (Рис. 2.8)

Body Cookies Headers (6) Test Results

Status: 200 OK Time: 815 ms Size: 2.04 KB

Save as Example

Pretty Raw Preview Visualize JSON

```
1 {  
2     "id": 4643,  
3     "category": "plumbing",  
4     "name": "PEX Clamp Tool",  
5     "inStock": true  
6 },  
7 {  
8     "id": 1225,  
9     "category": "power-tools",  
10    "name": "1/2 in. Brushless Hammer Drill",  
11    "inStock": true  
12 },  
13 {  
14     "id": 2177,  
15     "category": "ladders"  
16 }
```

Рис. 2.8

Щоб додати параметри Query, впишіть ключ та значення з JSON файлу, які Ви отримали у відповідному місці. (Рис. 2.9)

The screenshot shows the Postman interface with the following details:

- Params** tab is selected.
- Query Params** section:

Key	Value	Description	Bulk Edit
<input checked="" type="checkbox"/> category	plumbing		
- Body** tab is selected.
- JSON** response view:


```

1 [
2   {
3     "id": 4643,
4     "category": "plumbing",
5     "name": "PEX Clamp Tool",
6     "inStock": true
7   },

```
- Status bar: Status: 200 OK, Time: 2.21 s, Size: 531 B, Save as Example.

Рис. 2.9

Якщо значення вписати інше, ніж прийшло, буде помилка 400 Bad Request. (Рис. 2.10)

The screenshot shows the Postman interface with the following details:

- Params** tab is selected.
- Query Params** section:

Key	Value	Description	Bulk Edit
<input checked="" type="checkbox"/> category	fooba		
- Body** tab is selected.
- JSON** response view:


```

1 [
2   {
3     "error": "Invalid value for query parameter 'category'. Must be one of: ladders, plumbing, power-tools, trailers,
4           electric-generators, lawn-care"
5   }

```
- Status bar: Status: 400 Bad Request, Time: 861 ms, Size: 368 B, Save as Example.

Рис. 2.10

2.3 Змінні Path

Змінні Path в Postman це всього на всього placeholder. На відміну від параметрів Query, змінні Path повертають 1 набір даних в форматі JSON. (Рис. 2.11)

The screenshot shows the Postman application interface. At the top, there is a header with 'GET' selected, a URL field containing '({base URL})/tools/:toolId', and a 'Send' button. Below the header, there are tabs for 'Params', 'Authorization', 'Headers (5)', 'Body', 'Pre-request Script', 'Tests', and 'Settings'. A 'Cookies' tab is also present. Under the 'Params' tab, there is a table with one row: 'Key' 'Value' 'Description' with the value '4643' and the description 'Description'. Below the table, there are tabs for 'Body', 'Cookies', 'Headers (6)', and 'Test Results'. The 'Body' tab is selected, showing a JSON response with the following content:

```

1
2   "id": 4643,
3   "category": "plumbing",
4   "name": "PEX Clamp Tool",
5   "manufacturer": "JWGJW",
6   "price": 3.45,
7   "current-stock": 14,
8   "inStock": true
9

```

At the bottom right of the interface, there are buttons for 'Cookies', 'Auto-select agent', 'Runner', and 'Trash'.

Рис. 2.11

Для відслідкування запитів та відповідей, можна використовувати консоль.
Рис. (2.12)

The screenshot shows a browser's developer tools console tab. It displays two recent API requests:

- ▶ GET https://simple-tool-rental-api.glitch.me/tools/:toolId
- ▶ GET https://simple-tool-rental-api.glitch.me/tools/4643

Рис. 2.12

2.4 Авторизація API

Деякі ендпоїнти можуть вимагати авторизації. Для того, щоб підтвердити або подивитися ордери необхідно зареєструвати. Для цього продублюємо минулий запит, змінемо імя та метод на POST. А до змінної додамо /api-clients.

Перейдемо в боді, оберемо raw та формат JSON. Отримаємо accessToken
(Рис. 2.13)

POST {{base URL}}/api-clients

Body (7)

Params Authorization Headers (7) Body Pre-request Script Tests Settings Cookies Beautify

Body (1)

```

1 ...
2 ... "clientName": "Postman",
3 ... "clientEmail": "kingamoseley@example.com"
4 ...

```

Status: 201 Created Time: 316 ms Size: 299 B Save as Example

Pretty Raw Preview Visualize JSON

```

1 ...
2 ... "accessToken": "b6cb0cda787cc58a6394c8b6ef83206aab99a38ce13267259ec2760aac5bf7d3"
3 ...

```

Рис. 2.13

Переходимо у вкладку авторизації, та обираємо токен та тип (Рис. 2.14)

POST {{base URL}}/api-clients

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies

Type Bearer T... (1) Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment, we recommend using variables. Learn more about variables X

The authorization header automatically generates the token. See [authorization](#)

Inherit auth from...
No Auth
API Key
Bearer Token
JWT Bearer
Basic Auth
Digest Auth
OAuth 1.0
OAuth 2.0

Status: 409 Conflict Time: 375 ms Size: 283 B Save as Example

Pretty Raw JSON

"error": "Email is already registered. Try a different email."

Cookies Auto-select agent Runner Trash

Рис. 2.14

2.5 Проблема статус-кодів HTTP

Типічні помилки:

404 – перевірте URL або метод запитів HTTP

400 – перевірте request body та переконайтесь що правильний формат JSON

409 – клієнт зареєстрований

2.6 HTTP заголовки

HTTP заголовки можна знайти в запиті та у відповіді.

Для перегляду заголовків перейдіть в розділ «заголовки» та натисніть на «скріті». Відкриється перелік. (Рис. 2.15)

The screenshot shows the Postman interface with the 'Headers' tab selected, indicated by a red box around the 'Headers (8)' label. Below it, another red box highlights the '8 hidden' link. The table lists two visible headers under the 'Key' column: 'Content-Type' and 'User-Agent'. The 'Content-Type' row has a note: 'These headers will be automatically added and sent with the request. Click to view and modify these headers.' The 'User-Agent' row also has a 'Description' column. At the top right, there are 'Cookies', 'Params', 'Authorization', 'Body', 'Pre-request Script', 'Tests', and 'Settings' tabs. On the far right, there are 'Bulk Edit' and 'Presets' buttons.

Рис. 2.15

До типових заголовків відносять:

- Content Type – передає API інформацію про те, що request body в форматі JSON.
- Authorization – містить інформацію про автентифікацію.

До типічних хедерів відповіді відносять:

- Content Type – говорить клієнту (в даному випадку програмі Postman) що request body в форматі JSON.

2.7 JSON формат

Ми використовуємо JSON для передачі даних. (Рис. 2.16)



Рис. 2.16

JSON представлений у форматі «ключ-значення». Ключі та значення пишуться в середині фігурних скобок в лапках Значення в цифрах та булінах пишуться без лапок. (Рис.2.17)

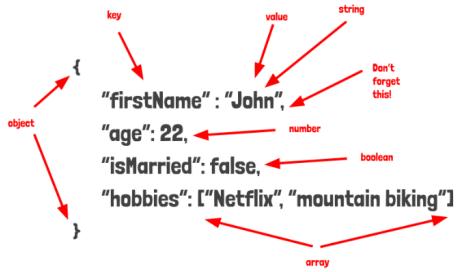


Рис.2.17

Якщо формат буде написаний неправильно, буде помилка, як на Рис. 2.18

Params Authorization ● Headers (8) **Body** ● Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL Text

1 John wants to order a bosch generator

Body Cookies Headers (6) Test Results Status: 400 Bad Request Time: 1321 ms Size: 264 B

Pretty Raw Preview Visualize JSON

```

1
2   "error": "Invalid or missing client name."
3
  
```

Рис. 2.18

Типічні помилки формату JSON включають наступні:

- Відсутні лапки для ключів та значень
- Відсутні коми після рядків
- Кома після останнього рядка

2.8 GET vs POST

Метод GET:

- Лише отримує дані
- Можна викликати кілька раз без побічних даних, тобто нічого не зміниться.
- Немає request body
- Можна додавати параметри в хедерах та URL

Метод POST:

- Створює нові дані

- При кожному виклику даного методу, будуть створюватися нові дані.
- Має request body
- Можна додавати параметри в баді, хедерах та URL

2.9 В яких випадках Postman не використовується

- Немає взаємодії з користувачем, такої як заповнення форм або кліки на кнопки.
- Краще не використовувати при тестуванні продуктивності, або будь-яких інших видах тестування в яких Ви здійснюєте багато запитів за короткий час.

2.10 Метод запитів PATCH

Метод запитів PATCH дозволяє змінювати існуючі дані.

2.11 Метод запитів DELETE

Метод запитів DELETE використовується для видалення даних. Зазвичай, request body не потрібний. Для того, щоб перевірити чи видалення пройшло успішно, використовуйте запит GET.

Розділ 3. ПІДГОТОВКА ДО АВТОМАТИЗАЦІЇ

3.1 Основи автоматизації.

Тестувати API вручну – великий обсяг роботи. Коли хтось вносить зміни до API нам треба вручну протестувати всі ендпоїнти та параметри та перевірити чи API працює як і раніше.

Postman дозволяє нам тестувати API за допомогою написання API тестів.

Автоматизація це процес за якого Postman виконує тестування, а тестувальник вмішується лише якщо щось іде не так.

3.2 Перший API тест

Postman має вбудовану функцію, яка дозволяє писати тести.

Зазвичай ми пишемо тести щоб перевірити чи буде відповідь така, як ми очікуєм, чи ні. Для написання тестів Postman використовує JavaScript.

Напишемо базовий тест для перевірки статусу коду. Для цього перейдемо у вкладку «тести» та справа під сніпетами знайдемо «статус коду: код 200».

Після вибору в нас зявиться скрипт для тесту. Натискаємо «відправити» та перевіряємо тест на успішність у вкладці «результати тестів». (Рис.3.1)

The screenshot shows the Postman interface for a 'Tool Rental API / Status' request. The 'Tests' tab is highlighted with a red box. The test script is:

```
1 pm.test("Status code is 200", function () {  
2     pm.response.to.have.status(200);  
3 });
```

Below the tests, the 'Test Results' section is highlighted with a red arrow pointing to it. It shows one result: 'PASS Status code is 200'. To the right, a status summary indicates 'Status: 200 OK'.

Рис.3.1

В даному скрипті ми зазначили, що очікуємо статус 200. Якщо ж в очікуваному результаті ми вкажемо інший статус – висвітиться помилка (Рис. 3.2), в якій говориться, що в результаті отримали статус 200, а очікували 400.

The screenshot shows the Postman interface with a failing test. The URL is `(({base URL}))/status`. The 'Tests' tab contains the following script:

```
1 pm.test("Status code is 400", function () {  
2     pm.response.to.have.status(400);  
3 });
```

The 'Test Results' section shows 0/1 failed. The failure message is: "FAIL Status code is 400 | Assertion Error: expected response to have status code 400 but got 200".

Рис. 3.2

Якщо в посиланні допустити помилку, тести не пройдуть. (Рис. 3.3). Нам буде повідомлено, що замість статусу 200, ми отримали статус 404.

The screenshot shows the Postman interface with a failing test. The URL is `(({base URL}))/status4`. The 'Tests' tab contains the following script:

```
1 pm.test("Status code is 200", function () {  
2     pm.response.to.have.status(200);  
3 });
```

The 'Test Results' section shows 0/1 failed. The failure message is: "FAIL Status code is 200 | Assertion Error: expected response to have status code 200 but got 404".

Рис. 3.3

А що, як в нас буде статус “down” замість ‘up’ в body? Напишемо ще один тест, щоб ви краще розуміли структуру. Ми завжди починаємо писати тест з функції `pm.test()`, яка приймає 2 параметри. Перший параметр – імя тесту, другий – функція колбек. В середині фігурних скобок пишемо твердження.(Рис. 3.4)

The screenshot shows the Postman interface with a test script in the code editor:

```
4 pm.test("Status is up", () => {  
5     pm.expect(1).to.eql(1);  
6 });
```

Below the code, the "Test Results" tab is selected, showing 2/2 passed tests:

- PASS Status code is 200
- PASS Status is up

At the top right, the status is 200 OK, time is 1067 ms, and size is 226 B.

Рис. 3.4

Будьте обережними при написанні тесту, адже якщо допустити синтаксичну помилку (наприклад, закрити функцію раніше), тести будуть завжди проходили.(Рис. 3.5)

The screenshot shows the Postman interface with an invalid test script in the code editor:

```
4 pm.test("Status is up"), () => {}  
5     pm.expect(1).to.eql(2);  
6 };
```

Below the code, the "Test Results" tab is selected, showing 2/2 passed tests, but the second test is highlighted with a red border:

- PASS Status code is 200
- PASS Status is up

At the top right, the status is 200 OK.

Рис. 3.5

3.3 Змінні postman

Для того, що зберегти час та уникнути копіювання та вставлення змінних з одного запиту в інший, Postman дозволяє створювати різні типи змінних:

1. Змінні колекції – доступні лише для колекції.(Рис. 3.6)

	Variable	Initial value	Current value	...
<input checked="" type="checkbox"/>	base URL	https://simple...	https://simple-tool-rental-a...	

Рис. 3.6

2. Змінні середовища:

- Доступні лише для середовища
- Корисні, коли ви хочете перевикористати одну й ту ж саму колекцію для різних серверів, таких як localhost, тестування та продакшн.

3. Глобальні змінні – доступні для всього робочого середовища.(Рис. 3.7)

No active Environment

An environment is a set of variables that allow you to switch the context of your requests.

No global variables

Global variables are a set of variables that are always available in a workspace.

Use variables to reuse values and protect sensitive data

Store sensitive data in variable type secret to keep its values masked on the screen. Learn more about [variable type](#)

Work with the current value of a variable to prevent sharing sensitive values with your team. Learn more about [variable values](#)

Рис. 3.7

Глобальні змінні дуже зручні у використанні(Рис. 3.8)

The screenshot shows the Postman interface with a GET request. The URL is {{base URL}}/orders/:orderId. In the 'Params' section, there is a row with 'Key' as 'orderId' and 'Value' as '{{orderId}}'. This value is highlighted with a red box.

Рис. 3.8

3.4 Робота зі змінними Postman зі скриптів

За допомогою скриптів можна створити нову або знайти існуючу змінну. Для цього використайте наступний скрипт: pm.collectionVariables.get("apiToken"). Якщо ввести змінну, якої не існує і подивитися в консоль, виведе undefined(Рис. 3.9)

The screenshot shows the Postman interface with a collection variable 'baseUrl' checked in the 'Console' tab. The output shows 'undefined'.

```
7 console.log(pm.collectionVariables.get("baseUrl"))
```

Body Cookies Headers (6) Test Results (1/1) Status: 33% Next: Invite at least 1 person. Invite

Рис. 3.9

Для створення нової змінної колекції, використовуйте наступний синтаксис: pm.collectionVariables.set("firstName", "John"). Крім того, за допомогою цього синтаксису можна змінити значення у змінної (Рис. 3.10)

The screenshot shows a script block with the following code:

```
1 pm.test("Status code is 401", function () {
2 | pm.response.to.have.status(401);
3 });
4
5 pm.globals.set("orderId", "uOUmzQONs5FBQwzPM5ejm")
```

On the right, a 'Globals' table shows the variable 'orderId' with an initial value of 'MwAizGyrOxKIVVeTjDT0Z' and a current value of 'uOUmzQONs5FBQwzPM5ejm'.

Рис. 3.10

Розділ 4. ЗАПУСК АВТОМАТИЗОВАНОГО ЗБОРУ

4.1 Collection Runner

Collection Runner – інструмент postman, який дозволяє запускати всю колекцію одним кліком, замість того, щоб проходити по кожному запиту. Є два шляхи, щоб здійснити запуск колекції:

Натиснути на «runner» внизу сторінки або за допомогою налаштувань колекції. (Рис. 4.1)

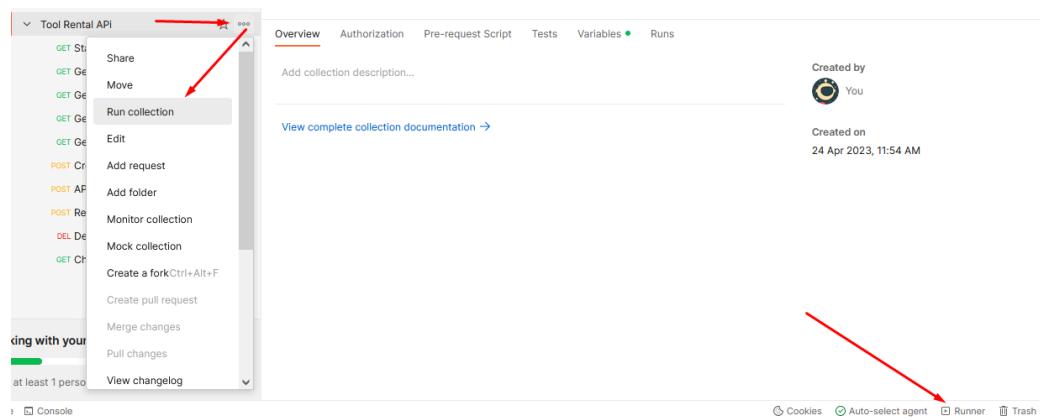


Рис. 4.1

Для того, щоб запити почали виконуватися, оберіть «запустити» та назву колекції(Рис. 4.2)

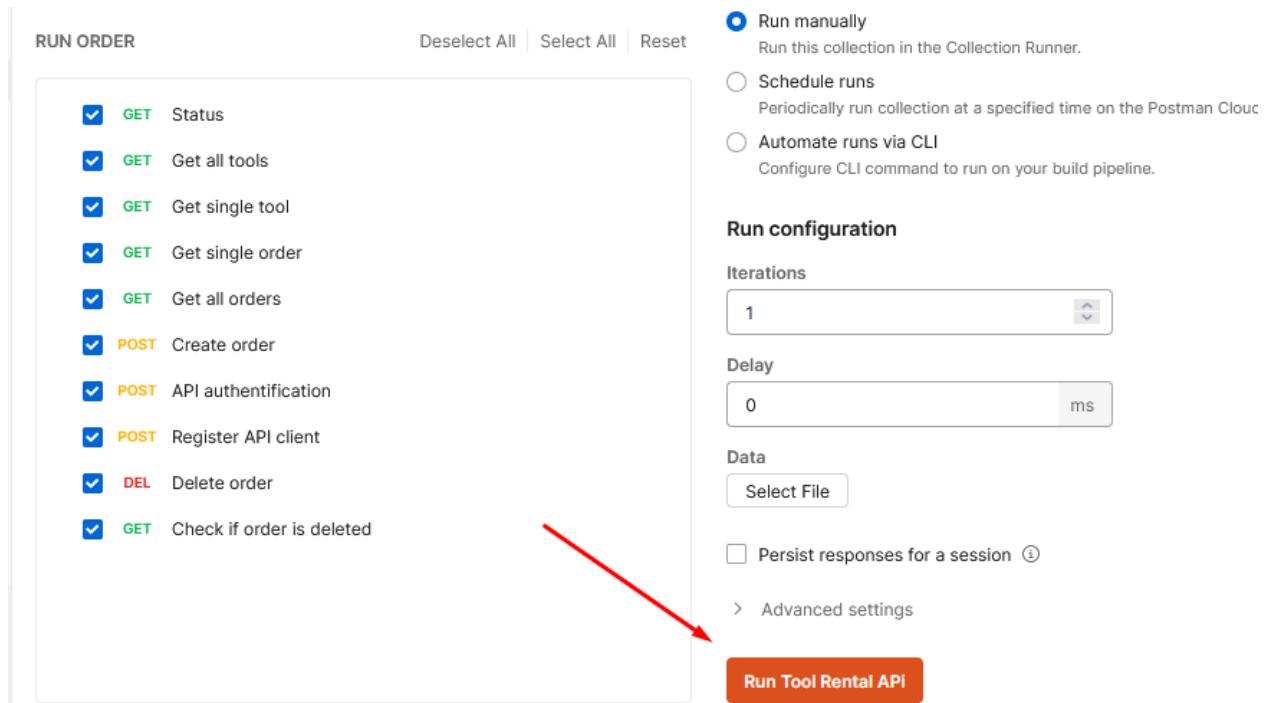


Рис. 4.2

Після виконання ви зможете побачити всю інформацію з виконання запитів, включаючи тести, які пройшли, та тести, які впали. (Рис. 4.3)

All Tests	Passed (5)	Failed (3)	Skipped (0)	View Summary
Iteration 1				
GET Status				
https://simple-tool-rental-api.glitch.me/status				200 OK 362 ms 226 B
PASS Status code is 200				
PASS Status is up				
GET Get all tools				
https://simple-tool-rental-api.glitch.me/tools?category=electric-generators				200 OK 227 ms 724 B
PASS Status code is 200				
GET Get single tool				
https://simple-tool-rental-api.glitch.me/tools/4643				200 OK 212 ms 340 B

Рис. 4.3

4.2 Моніторинг Postman.

Автоматизувати запуск колекції можна за допомогою моніторингу postman.
Для створення перейдіть до своєї колекції(Рис. 4.4)

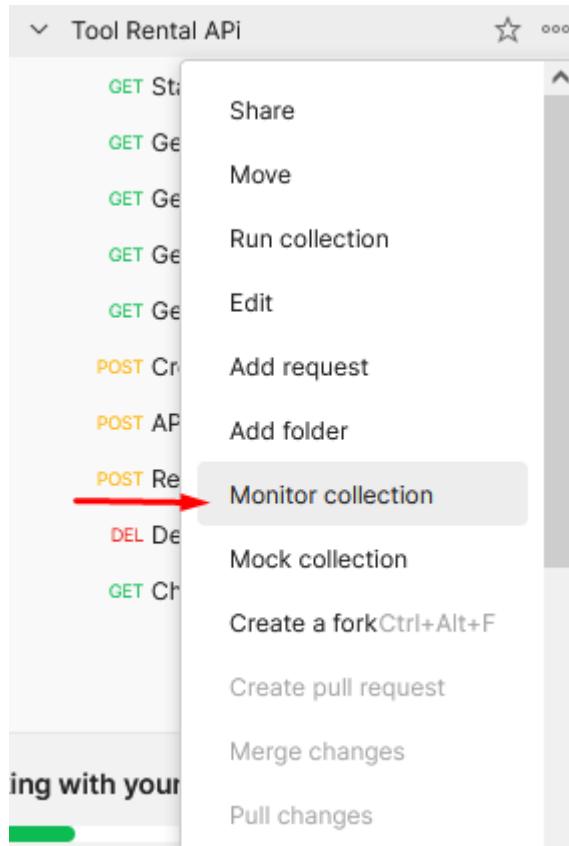


Рис. 4.4

Далі вписуєте імя, а дефолтні налаштування можна залишити без змін. Та натискаєте «створити моніторинг» (Рис. 4.5)

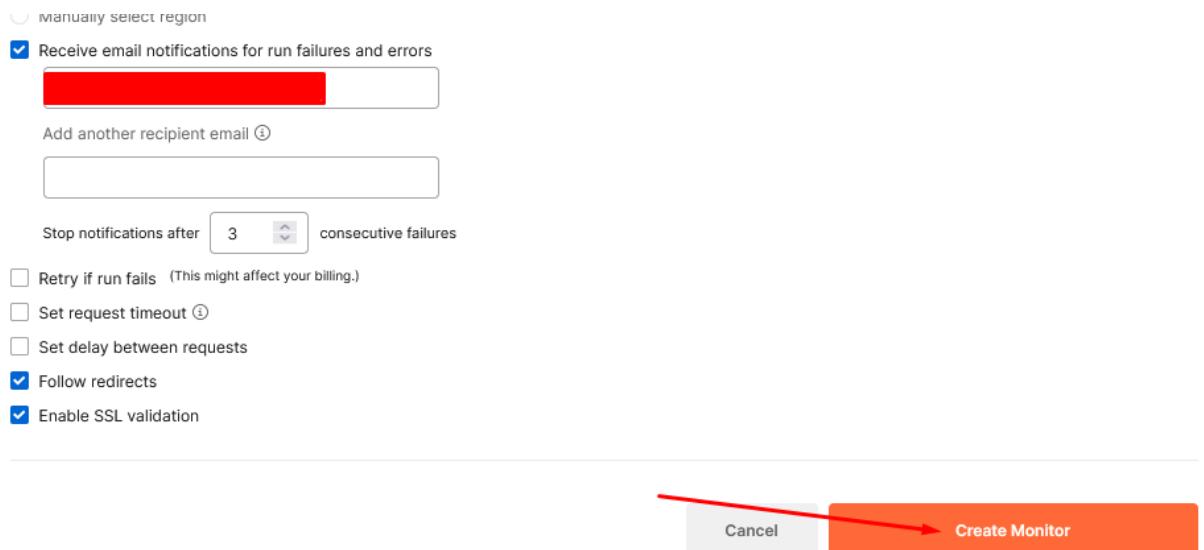


Рис. 4.5

Після виконання, якщо ви бачите “unhealthy”, значить десь є помилка. Відстежити її можна за допомогою інформації. (Рис. 4.6)

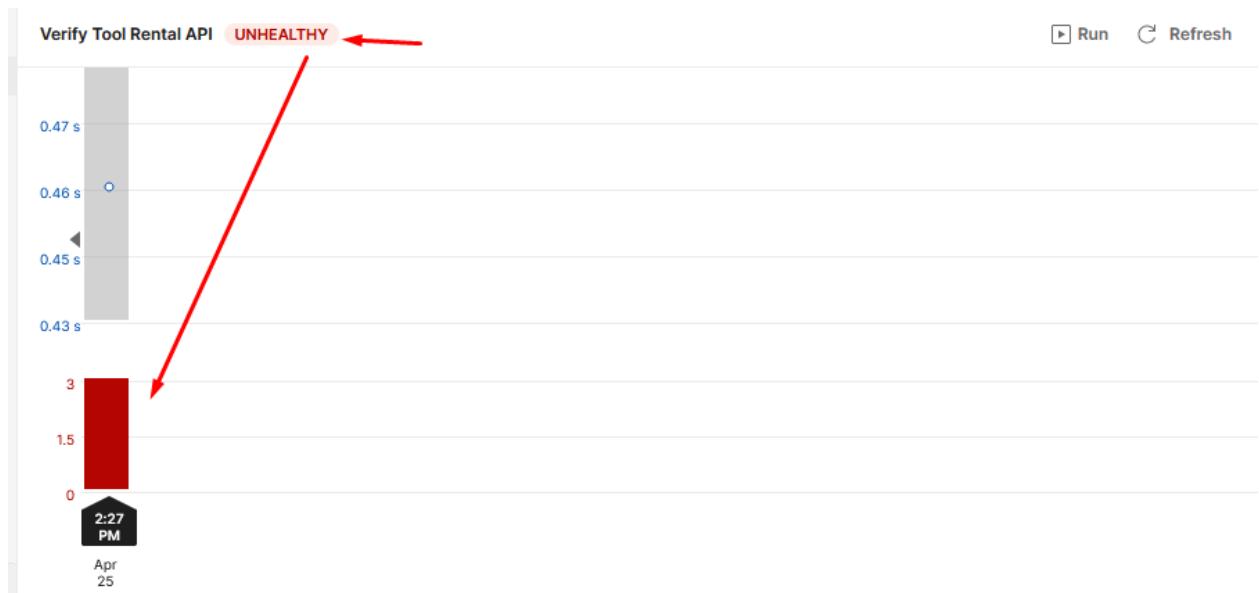


Рис.4.6

Нижче можна побачити кількість тестів, які пройшли, та тих – які впали(Рис. 4.7)

All Tests	Passed	Failed
✓ PASS Status is up		
GET Get all tools https://simple-tool-rental-api.glitch.me/tools?category=electric-generators	200 OK 16 ms 510 B	
✓ PASS Status code is 200		
GET Get single tool https://simple-tool-rental-api.glitch.me/tools/4643	200 OK 23 ms 127 B	
✓ PASS Status code is 200		
GET Get single order https://simple-tool-rental-api.glitch.me/orders/%7B%7BorderId%7D%7D	401 Unauthorized 21 ms 41 B	
✗ FAIL Status code is 200		
<input checked="" type="checkbox"/> Cookies <input checked="" type="checkbox"/> Auto-select agent <input type="checkbox"/> Runner		

Рис. 4.7

4.3 Newman – інструмент CLI в Postman

Newman інструмент CLI в Postman, за допомогою якого можна запускати колекцію postman з терміналу. Для того, щоб використовувати цей інструмент має бути встановлений Node.js. Встановити newman можна вводячи в терміналі наступну команду: `npm install -g newman`

Запустити newman можна за допомогою команди `newman run` + посилання на Вашу колекцію. (Рис. 4.8)

```
~/workspace/tool-rental-api ➔ newman run https://www.getpostman.com/collections/db6aae5712d7b8ae6f2e
newman

Tool Rental API

→ Status
  GET https://simple-tool-rental-api.glitch.me/status [200 OK, 226B, 594ms]
    ✓ Status code is 200
    [
      'UP',
      'UP'
    ]
    ✓ Status is UP
```

Рис. 4.8

Важливо! Якщо ви запускаєте за допомогою посилання на колекцію, при зміні в запитах вам треба буде знову брати посилання та вставляти його в newman.

Після запуску буде показано результати виконання запитів, а також можна буде подивитися кількість успішних тестів та кількість тестів, що впали (Рис. 4.9)

	executed	failed
iterations	1	0
requests	9	0
test-scripts	18	0
prerequest-scripts	9	0
assertions	12	0
total run duration: 2.7s		
total data received: 3.27kB (approx)		
average response time: 277ms [min: 131ms, max: 1155ms, s.d.: 311ms]		

Рис. 4.9

БІБЛІОГРАФІЯ:

1. <https://simple-tool-rental-api.glitch.me/>
2. <https://github.com/vdespa/quick-introduction-to-postman/blob/main/simple-tool-rental-api.md#Get-all-tools>
3. Despa V. Quick Introduction to Postman and API Testing for Beginners [Електронний ресурс] / Valentin Despa. – 2023. – Режим доступу до ресурсу:
file:///C:/Users/Professional/Downloads/Introduction+to+Postman+and+API+v2023-1+(course+notes)-1.pdf.
4. <https://www.postman.com/downloads/>
5. <https://www.postman.com/release-notes/>
6. <https://github.com/postmanlabs/newman>

