

API Testing: Postman and Rest Assured

Learn what an API is, how to test it by hand in Postman, and how to automate it with code.

1. What is an API?

API means Application Programming Interface. It is the way one software talks to another.

Restaurant example: you (the app screen) tell the waiter (API) what you want. The waiter goes to the kitchen (server and database) and brings the food (data) back. You never go to the kitchen yourself.

App / browser (client) → request → API (server) → query → Database

When you open an app and see your orders, the app has sent a **request** to an API and shown the **response**.

Why test APIs separately? They are faster than UI tests, more stable (no buttons that move), and they find bugs earlier, even before the screen is ready.

2. Parts of a request

PART	MEANING	EXAMPLE
URL / Endpoint	Address of the API.	<code>https://api.shop.com/users/2</code>
Method	What action you want (see below).	GET, POST
Headers	Extra information about the request.	<code>Content-Type: application/json</code>
Query param	Filter added in the URL after <code>?</code> .	<code>/users?page=2</code>
Path param	A value inside the URL path.	<code>/users/2</code>
Body	Data you send (for POST, PUT).	<code>{"name": "Ravi"}</code>
Authorization	Proof of who you are: API key, Bearer token, username-password.	<code>Authorization: Bearer abc123</code>

3. HTTP methods (CRUD)

METHOD	USE	CRUD
GET	Read data	Read
POST	Create new data	Create
PUT	Replace the whole record	Update
PATCH	Change only some fields	Update
DELETE	Remove data	Delete

4. Status codes (the server's reply number)

CODE	MEANING
200 OK	Request worked.
201 Created	New record created (after POST).
204 No Content	Worked, nothing to return (often after DELETE).
400 Bad Request	Your request is wrong (missing or invalid data).
401 Unauthorized	Not logged in or invalid token.
403 Forbidden	Logged in, but not allowed to do this.
404 Not Found	That record or URL does not exist.
500 Internal Server Error	Problem on the server. This is a bug to report.

Remember: **2xx** success, **4xx** client's mistake, **5xx** server's mistake.

5. JSON: the data format

```
{
  "id": 2,
  "name": "Ravi",
  "job": "QA",
  "skills": ["Selenium", "API"]
}
```

How to read it: data is in `"key": value` pairs. `{ }` is an object, `[ ]` is a list. Values can be text (in quotes), numbers, true/false, or null.

6. Manual API testing in Postman

Postman is a free app to send requests and see responses without writing code. Basic use: choose method → paste URL → add headers/body → click **Send** → check status, body and time.

POSTMAN WORD	MEANING
Collection	A folder of saved requests.
Environment	Saved variables such as <code>{{baseUrl}}</code> and <code>{{token}}</code> , so you can switch between QA and Staging easily.
Tests tab	Place to write small JavaScript checks that run after the response.
Collection Runner / Newman	Runs a whole collection at once. Newman does it from the command line, useful in automation pipelines.

```
pm.test("Status is 200", () => {
  pm.response.to.have.status(200);
});
const json = pm.response.json();
pm.expect(json.name).to.eql("Ravi");
pm.environment.set("token", json.token);
```

What it does: check the status code is 200 → read the response as JSON → check `name` equals "Ravi" → save the token in a variable so the next request can use it.

7. Automating APIs with Rest Assured (Java)

Rest Assured is a Java library to write API tests in a simple "given-when-then" style. Together with TestNG or JUnit, it becomes an automated suite.

given: setup (headers, params, body) → when: send request → then: check response

```
import static io.restassured.RestAssured.*;
import static org.hamcrest.Matchers.*;
```

What it does: lets you write `given()`, `equalTo()` and others directly, without long class names.

GET request

```
given().baseUrl("https://api.shop.com")
.when().get("/users/2")
.then().statusCode(200).body("name", equalTo("Ravi"));
```

What it does: set the base address → send GET to `/users/2` → check status is 200 → check the `name` field in the JSON is "Ravi".

POST request with body

```
given().header("Content-Type", "application/json")
.body("{\"name\":\"Ravi\",\"job\":\"QA\"}")
.when().post("/users")
.then().statusCode(201).body("job", equalTo("QA"));
```

What it does: tell the server we are sending JSON → send the new user data → expect 201 Created → check the response has job "QA".

Query parameter and token

```
given().queryParams("page", 2)
.header("Authorization", "Bearer " + token)
.when().get("/users")
.then().statusCode(200);
```

What it does: calls `/users?page=2` with a login token in the header.

Take a value from response for next call

```
String id = given().body(payload).when().post("/users")
    .then().extract().path("id");
```

What it does: creates a user and saves the new `id` from the response, so you can then GET or DELETE that same user. This is called chaining.

Response time and headers

```
.then().time(lessThan(2000L)).header("Content-Type", containsString("json"));
```

What it does: response must come in under 2000 ms and be JSON.

8. What to check in an API test

- **Status code** is correct (200, 201, 404...).
- **Response body:** correct fields and values, right data types, nothing missing.
- **Headers:** content type, caching, security headers.
- **Response time:** fast enough.
- **Negative cases:** missing required field → 400, wrong token → 401, unknown id → 404, wrong method.
- **Data check:** after POST, a GET returns the same data (and the database has it).
- **Security basics:** a user must not read another user's data.

9. More API knowledge

More status codes

CODE	MEANING
301 / 302	Redirect: the resource moved permanently / temporarily.
405 Method Not Allowed	Wrong method, such as DELETE on a read-only URL.
409 Conflict	Clash with existing data, such as an email that already exists.
415 Unsupported Media Type	Wrong <code>Content-Type</code> sent.

422 Unprocessable Entity	Data format is right but values fail the rules.
429 Too Many Requests	Rate limit crossed.
502 / 503 / 504	Bad gateway / service unavailable / gateway timeout. Server side problems.

Authentication types

TYPE	HOW IT WORKS
Basic Auth	Username and password sent (encoded) with every request.
API key	A secret key in the header or URL that identifies the app.
Bearer token (JWT)	Login once, get a token, send it as <code>Authorization: Bearer <token></code> . Tokens expire.
OAuth 2.0	"Login with Google" style. The user allows an app to access data, and the app receives an access token.

Authentication = who are you? **Authorization** = what are you allowed to do?

REST principles

- **Client-server:** client and server are separate.
- **Stateless:** each request carries everything needed. The server does not remember earlier requests.
- **Resources and URLs:** things are named as nouns (`/users/2`), the method says the action.
- **Uniform responses:** usually JSON with proper status codes.
- **Cacheable** responses where suitable.

Common headers

`Content-Type` (format of what you send), `Accept` (format you want back), `Authorization` (token), `Cache-Control`, `User-Agent`, `Set-Cookie` / `Cookie`.

Swagger / OpenAPI

A standard way to document an API. The Swagger page lists all endpoints, inputs and sample responses, and lets you try requests. Testers read it to write test cases.

Mock API

A fake API that returns fixed responses. It lets testing continue when the real API is not ready or is costly to call. Postman can create mock servers.

10. More Postman

- **Variables:** `{{baseUrl}}` style. Scopes: global, collection, environment, local.
- **Pre-request script:** JavaScript that runs **before** the request, for example to create a random email.
- **Chaining:** save a value from one response (token, id) in a variable and use it in the next request.
- **Data-driven run:** run a collection with a CSV or JSON file so each row is one test.
- **Newman:** command line runner for CI/CD.

```
pm.environment.set("email", "user" + Date.now() + "@test.com");
pm.test("Response time under 2s", () =>
pm.expect(pm.response.responseTime).to.be.below(2000));
pm.test("Has users", () =>
pm.expect(pm.response.json().data.length).to.be.above(0));
```

What it does: creates a new unique email for every run (a pre-request script) → checks the response came within 2 seconds → checks that the list of users is not empty.

```
newman run collection.json -e qa-environment.json
```

What it does: runs the whole collection from the terminal using the QA environment values.

11. More Rest Assured

```
Response res = given().when().get("/users/2");
int code = res.getStatusCode();
String name = res.jsonPath().getString("data.first_name");
long ms = res.getTime();
```

What it does: stores the whole response in a `Response` object, then reads the status code, one JSON value (using a path) and the time taken. Use this when you need to do several checks or use values later.

```
.then().body("data.id", hasItems(1, 2))
    .body("data.size()", greaterThan(0))
    .body(matchesJsonSchemaInClasspath("user-schema.json"));
```

What it does: the list of ids contains 1 and 2 → list is not empty → the whole response structure matches a saved JSON schema file (field names and types). Schema validation catches missing or renamed fields.

```
RequestSpecification spec = new RequestSpecBuilder()
    .setBaseUrl("https://api.shop.com")
    .addHeader("Authorization", "Bearer " + token).build();
given().spec(spec).when().get("/users").then().statusCode(200);
```

What it does: keeps common settings (URL, token) in one reusable object, so you do not repeat them in every test.

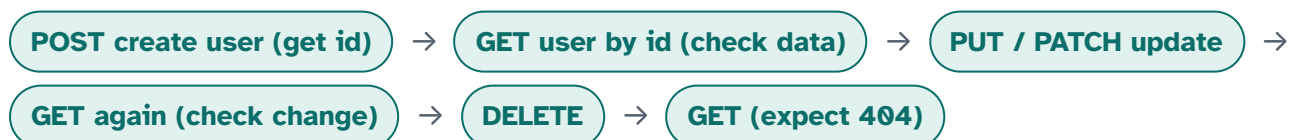
```
given().log().all().when().get("/users").then().log().all();
```

What it does: prints the full request and response, which is very helpful to debug a failing test.

POJO: instead of typing JSON text, create a Java class (`User` with name and job) and pass the object to `.body(user)`. Rest Assured converts it to JSON (serialization). The reverse, JSON to object, is deserialization.

12. One full CRUD flow

A typical end-to-end API test uses the result of one call in the next.



13. Test cases for one API (POST /users)

TYPE	WHAT TO TRY	EXPECTED
Valid	All fields correct	201, id returned

Missing field	No <code>name</code>	400 or 422 with clear message
Wrong data type	<code>"age": "abc"</code>	400 or 422
Boundary	Name with 0, 1, max and max+1 characters	Rules followed
Duplicate	Same email again	409 or 400
Extra fields	Unknown field in the body	Ignored or rejected as per rule
Wrong header	No or wrong <code>Content-Type</code>	415
Auth	No token, expired token, token of another user	401 or 403
Security	SQL or script text in values	Safe handling, no 500 error
Large body	Huge text or very big payload	Rejected cleanly
Wrong method	PUT on this URL	405

Common API bugs: wrong status code (200 for an error), 500 errors for bad input, missing validation, sensitive data (passwords, tokens) in the response, wrong data types, slow responses, one user able to read another user's data.

14. API testing vs UI testing

	API TESTING	UI TESTING
Speed	Very fast	Slower
Stability	Stable, no screens involved	Can break when the screen changes
Starts	Early, before the UI is ready	After the UI is built
Checks	Business logic and data	What the user sees and does

A good plan uses many API tests, fewer UI tests for key journeys, and manual testing for new and exploratory work. For speed under many users, a tool like **JMeter** is used (performance testing).

15. SQL basics a tester needs

After an API or UI action, testers often check the database to confirm the data is saved correctly.

QUERY	WHAT IT DOES
<code>SELECT * FROM users;</code>	Show all rows of the table <code>users</code> .
<code>SELECT name, email FROM users WHERE id = 2;</code>	Show chosen columns for one record.
<code>SELECT COUNT(*) FROM orders WHERE status = 'PAID';</code>	Count paid orders.
<code>SELECT * FROM users ORDER BY created_at DESC LIMIT 5;</code>	Latest 5 users.
<code>SELECT u.name, o.amount FROM users u JOIN orders o ON u.id = o.user_id;</code>	Join two tables to see each user with their orders.
<code>SELECT status, COUNT(*) FROM orders GROUP BY status;</code>	Count of orders for each status.

INSERT / UPDATE / DELETE

Add,
change or
remove
rows.
Take care:
always
use
WHERE
with
UPDATE
and
DELETE.

Keys: a **primary key** uniquely identifies a row (like `id`). A **foreign key** points to the primary key of another table. **DELETE** removes rows (can have WHERE), **TRUNCATE** removes all rows fast, **DROP** deletes the whole table.

16. Quick interview answers

- **What is an API?** A way for two applications to talk to each other using requests and responses.
- **PUT vs PATCH?** PUT replaces the whole record. PATCH changes only the given fields.
- **401 vs 403?** 401: not authenticated (who are you?). 403: authenticated but not allowed.
- **What is idempotent?** Sending the same request many times gives the same result. GET, PUT and DELETE are idempotent. POST is not (it creates a new record each time).
- **REST vs SOAP?** REST is lighter and usually uses JSON. SOAP is older, uses XML and strict rules.
- **What is a payload?** The data sent in the request body.
- **How do you handle auth in tests?** Get a token from the login API once, store it in a variable, and send it in the header of other requests.
- **Postman vs Rest Assured?** Postman: quick manual testing and small checks. Rest Assured: full code-based automation that fits in a CI/CD pipeline.
- **PUT vs POST?** POST creates a new resource. PUT replaces an existing one at a known URL. PUT is idempotent.
- **Authentication vs authorization?** Who you are vs what you may do.

- **What is JWT?** A signed token given after login and sent in the header. It has an expiry time.
- **What is Swagger?** A page that documents an API and lets you try it.
- **How do you validate the response structure?** JSON schema validation, or check each field and its data type.
- **What if the API returns 500 for wrong input?** It is a bug. The server should return a 4xx error with a message.
- **What is serialization?** Converting a Java object to JSON for the request. Deserialization is the reverse.
- **How do you test an API that is not ready?** Use a mock server with expected responses.
- **What is rate limiting?** The API blocks a user who sends too many requests in a time window (429).
- **DELETE vs TRUNCATE vs DROP?** Remove selected rows / remove all rows / remove the table itself.
- **Primary key vs foreign key?** Unique identity of a row vs link to another table's row.
- **INNER JOIN vs LEFT JOIN?** Inner: only rows that match in both tables. Left: all rows of the left table, with matches from the right if any.