

# בדיקות ממשקים

# Web API Testing

# Using Postman



POSTMAN

# What is an API

API Stands for

A

→ Application

P

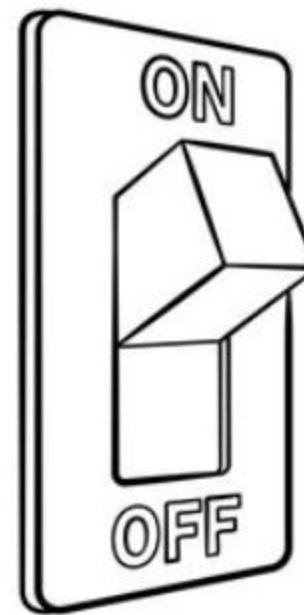
→ Programming

I

→ Interface

מה זה API ?

# What is an Interface



מה זה ממש ?

# What is an Interface

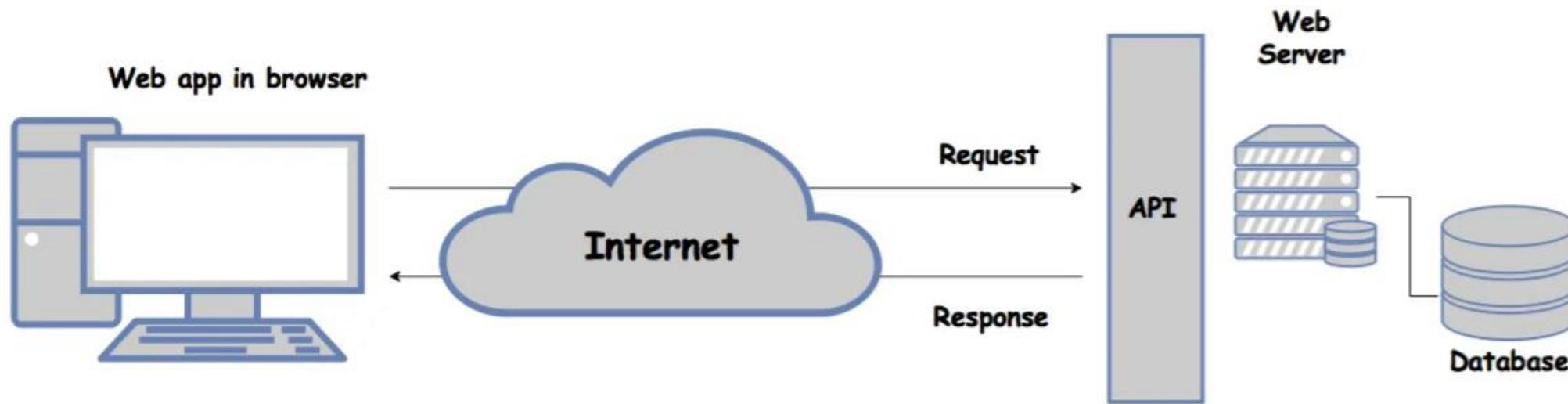


ATM Machine

מה זה ממש ?

# What is Web API

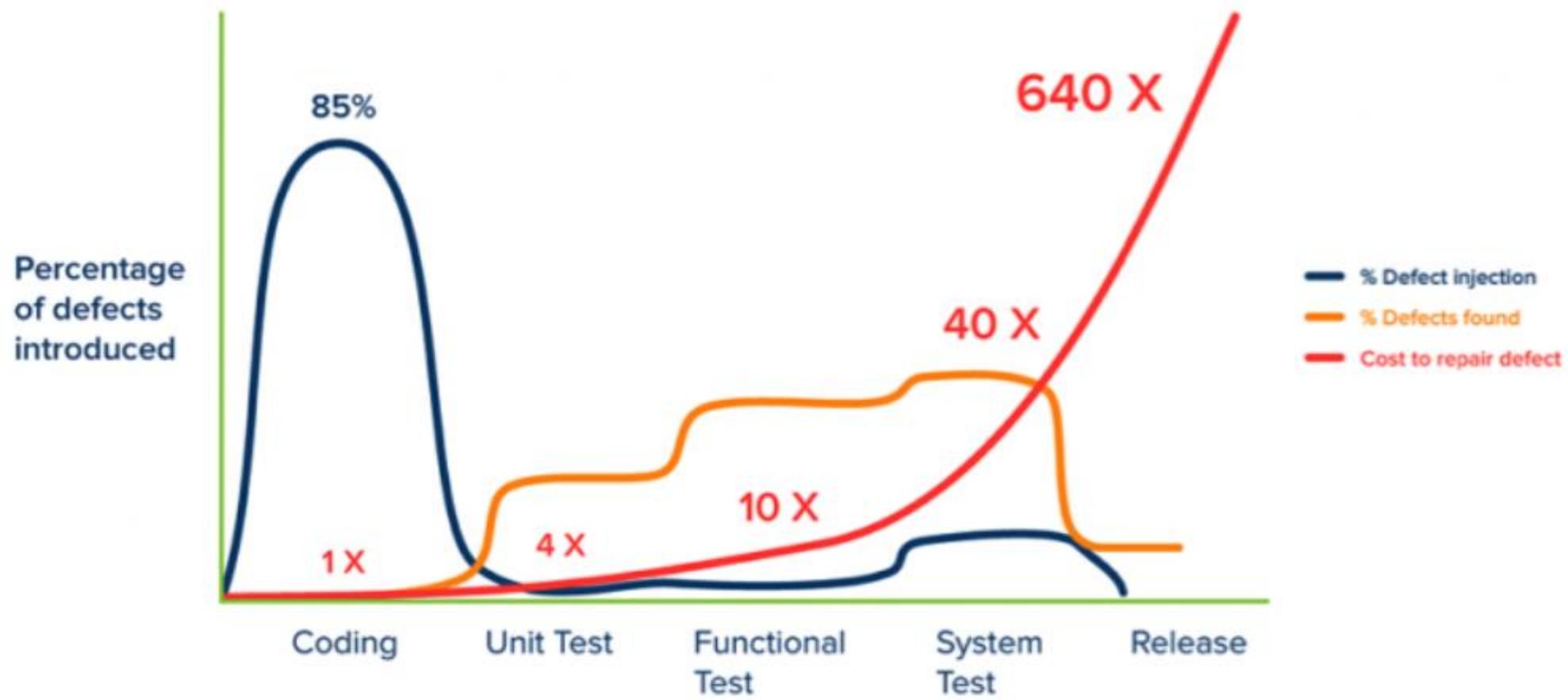
windows one can use ASP.NET Web API as backend. Hence, a web API is good for using with native applications which require web services but not



application development especially when it is an ASP.NET web application.

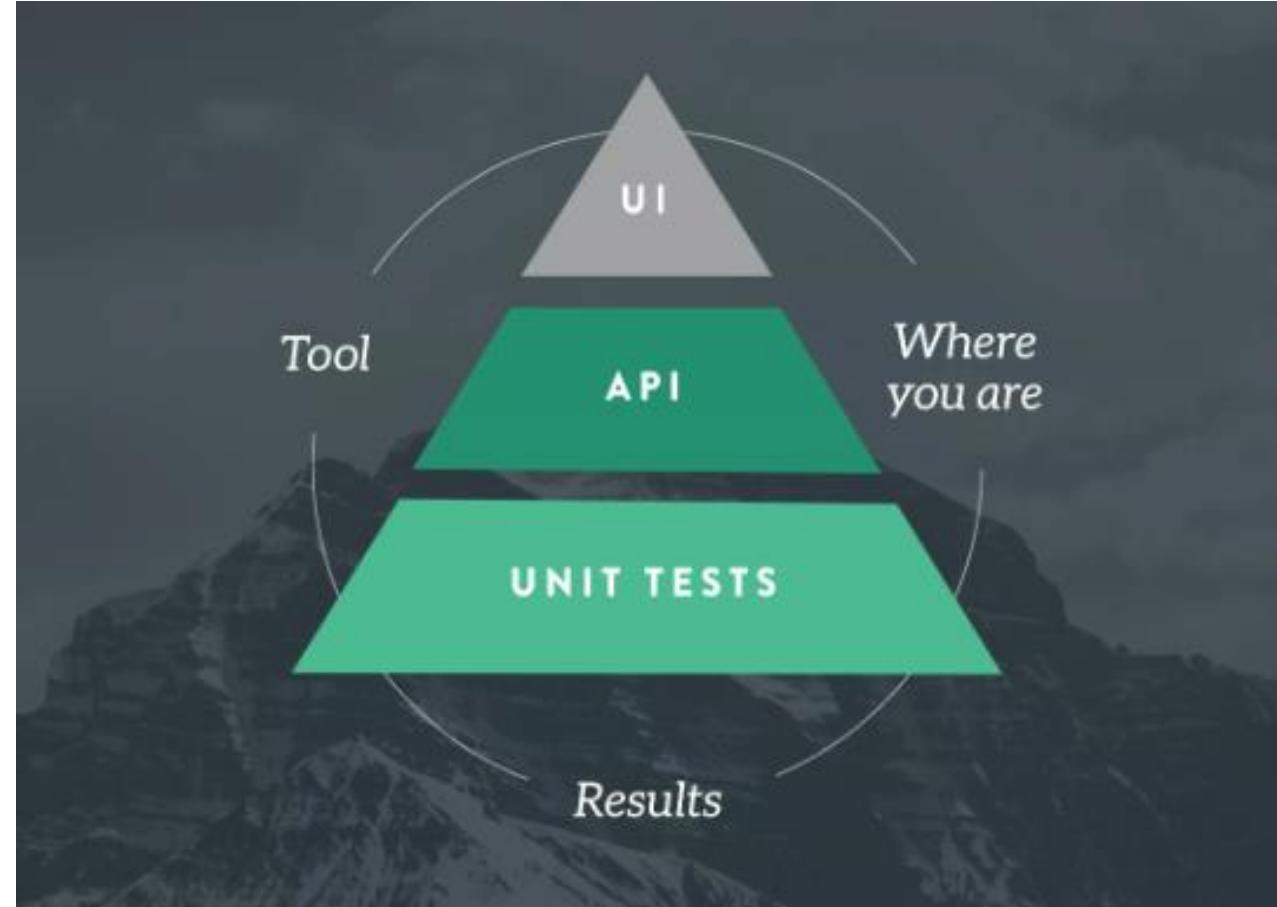
מה זה ממשק אינטרנט ?

# Why starting QA early in SDLC



פערם בעליות ובהשפעה של תקלות בשלבים שונים של המוצר ככל שנמצא את הבאגים רחוק יותר משלב כתיבת הקוד המחייב  
יהה גבוה יותר בצורה אקספוננציאלית

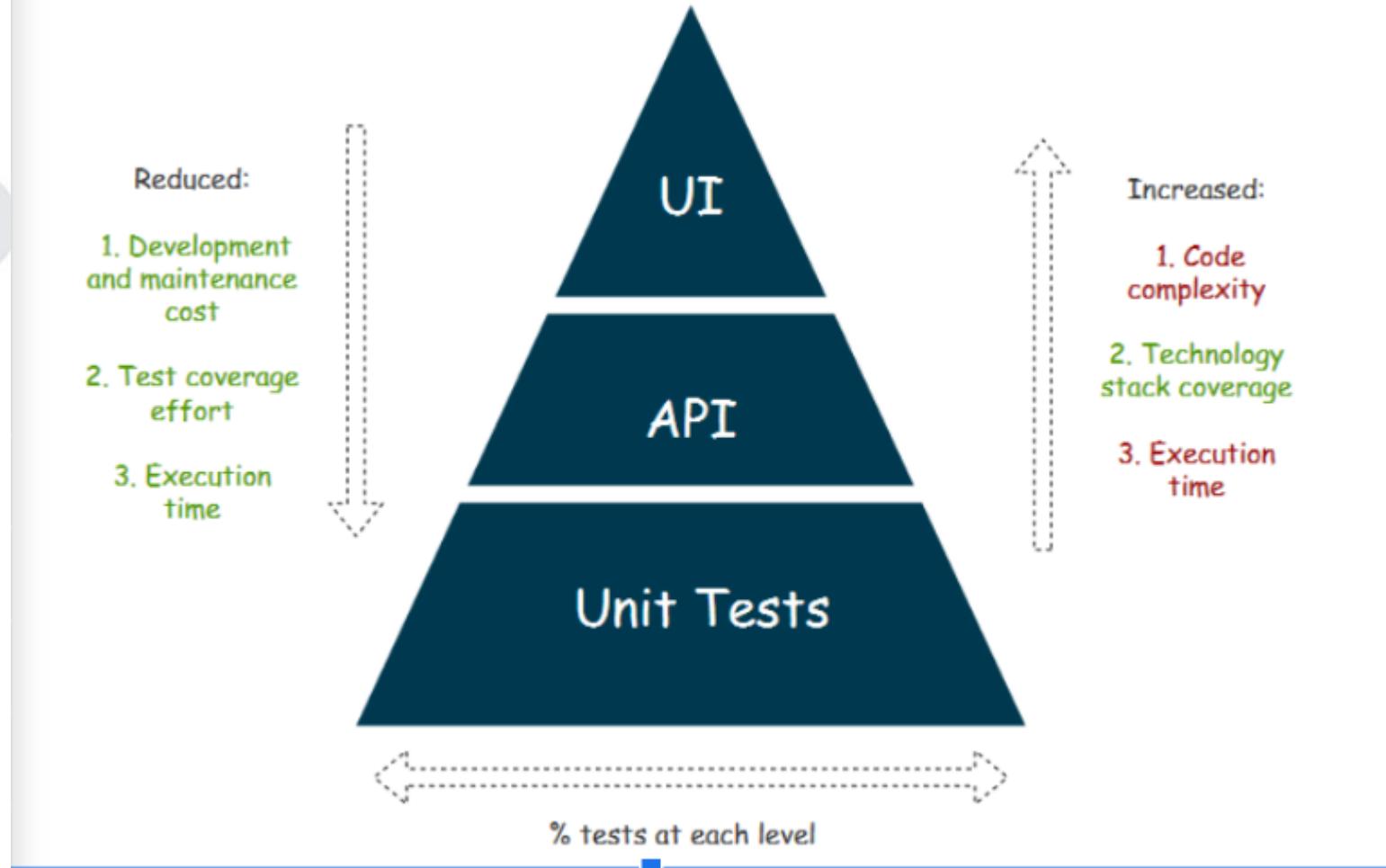
# Pyramid of testing



נכיר את מודל פירמידת הבדיקות שכל איש בדיקות צריך להכיר

# Why use API in Testing is better

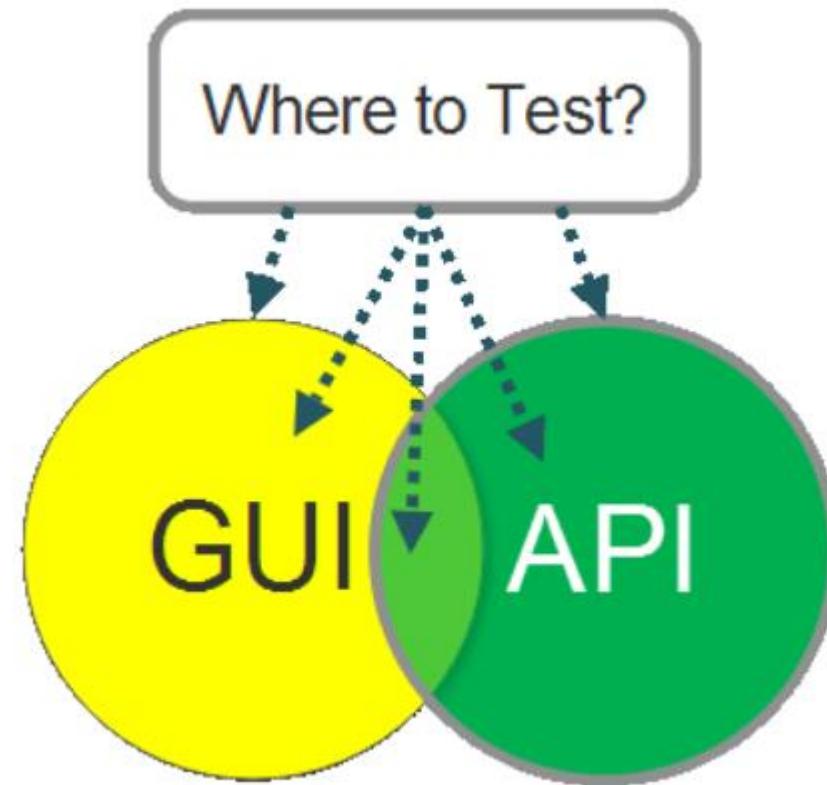
פירמידת הבדיקות: בקצה  
העליון בבדיקות קצח לסקא  
(E2E) בבדיקות יקרות בזמן  
הרצה וזמן יצירה, לעומת  
סוגי הבדיקות האחרות:  
בדיקות ממשקים מהירות  
וחולות יותר, כדי מסיע  
לכטיבת הבדיקות  
משגים, הוא הפוסטמן  
בדיקות יחידה הם הבדיקות  
הकצרות והיעילות ביותר,  
ארן מוגבלות מבחינה  
היקף הבדיקה.



# Why get into automation

Manual Testing	Automation Testing
The simplest low-level type, during each a QA runs all tests manually, without using any helpful software.	A QA uses a special program/tool for running tests, already existing or written specially for the project.
A time-consuming process.	Time-saving, particularly, because a QA can re-run the same tests numerous times
Can be repetitive and boring.	Helps to avoid repetitive tasks as a QA “delegates” them to a computer.
Is suitable for almost any software product.	Is suitable only for stable systems and used mainly for regression. Some types of testing (research, ad-hoc, etc.) cannot be automated.
Helps to define whether automation testing is possible and necessary.	100% automation is not possible.

# API vs. GUI Backend vs. Frontend



כאייש בדיקות נרצה להבין איך לכתוב ולהריץ בדיקות קצחה וגם בדיקות ממשקיים



# API testing

Compared by	API testing	UI testing
<i>Functionality</i>	The functionality of the business logic	The functionality of the UI and the business logic
<i>Fragility</i>	Somewhat fragile	Fragile
<i>Speed of execution</i>	Fast	Slow
<i>Maintenance cost</i>	Medium	High
<i>Who writes them</i>	Developers and testers	Testers

תרכנות וחסרונות של בדיקת ממשקים מול בדיקת ממשק גרפי, בדיקה של צד שרת מול בדיקת צד שרת הצד השני.

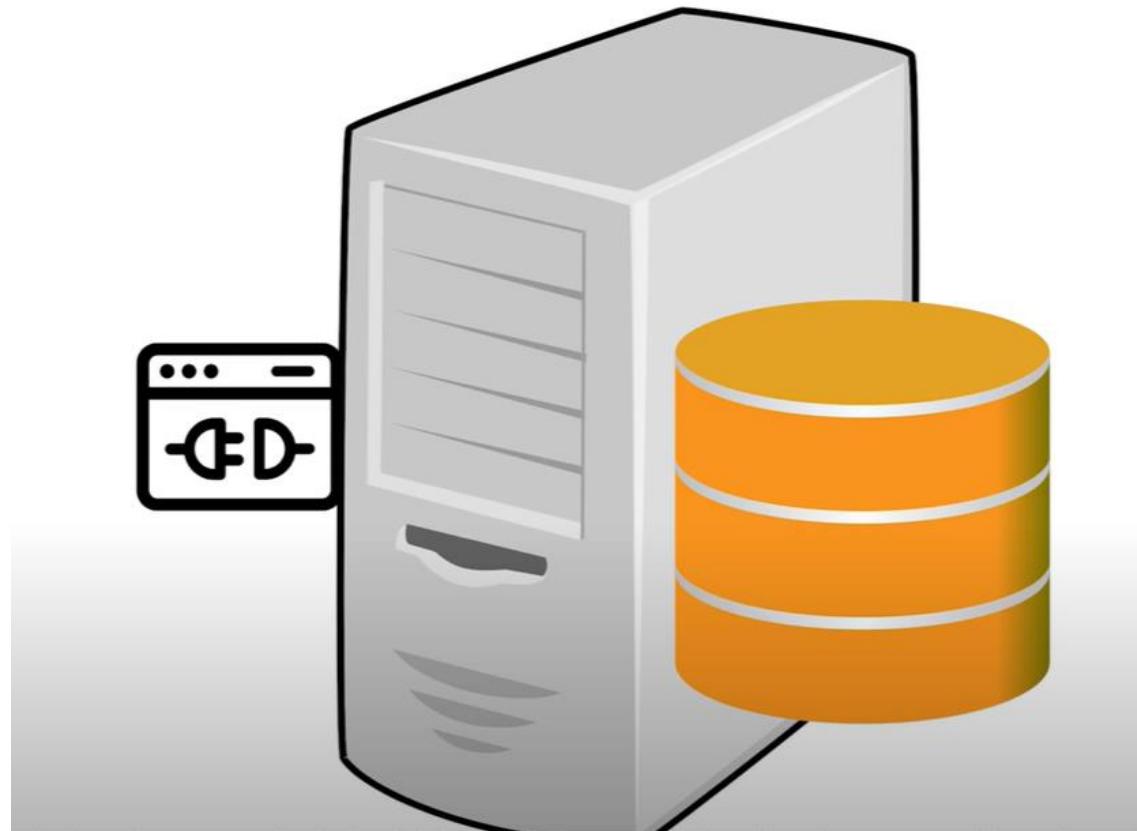
# Concept of API

**API** = **an interface**  
**to data**

- 👉 **they are designed to be used from programs**
- 👉 **no friendly interface**

בדיקות ממשקם הוא בדיקות של הממשק האינטראקטיבי (Web) עם הנתונים שיושבים לצד השרת (Backend)

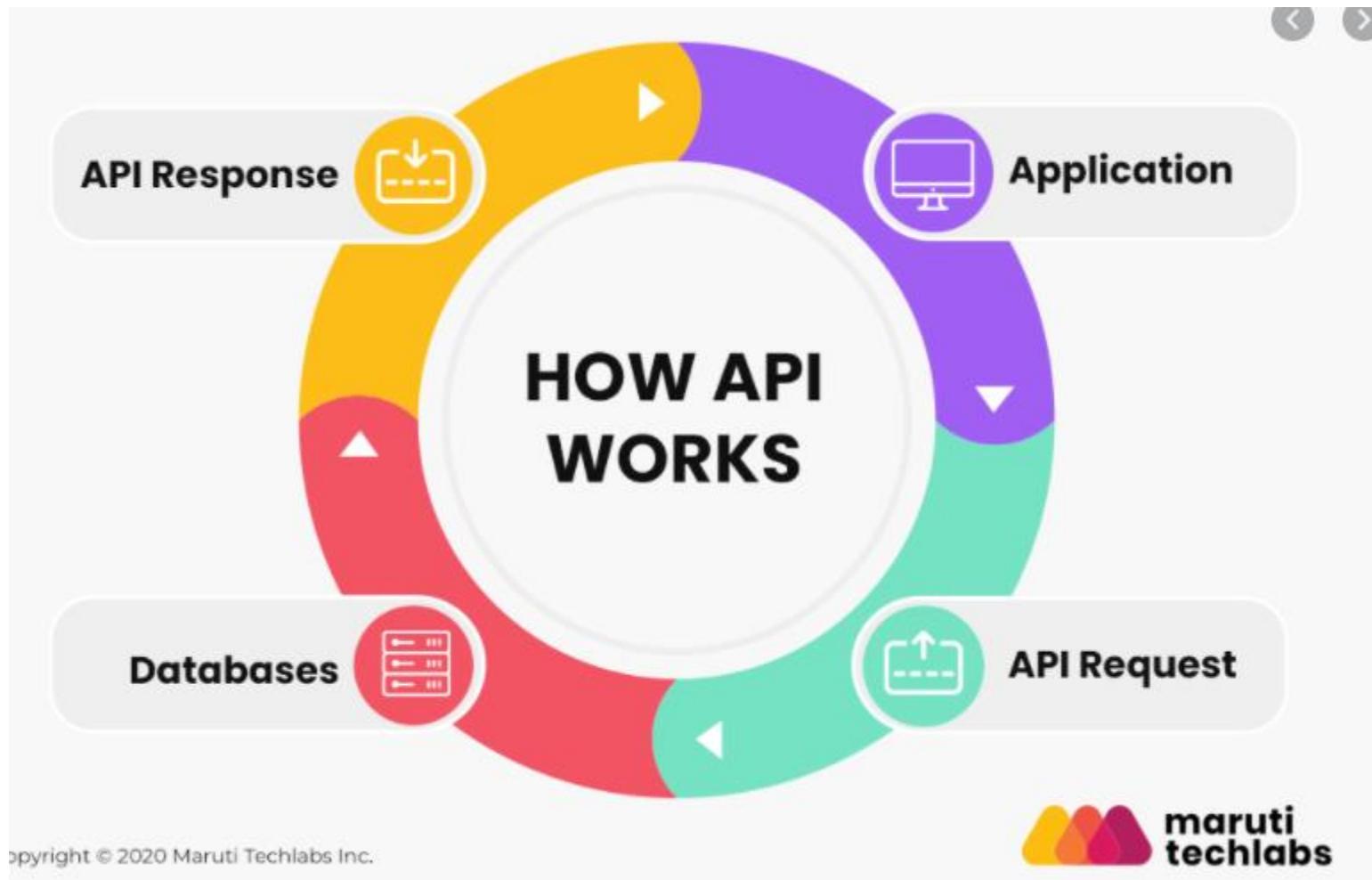
# API



הממשק האינטראקטיבי מאפשר לך לאפשר למשתמשים במכשיר, הממשק האינטראקטיבי הוא מערכת המפותחת על ידי מפתח צד שרת.  
חשוב להבין את המושג משקל, לפני שמסבירים את המושג API Web

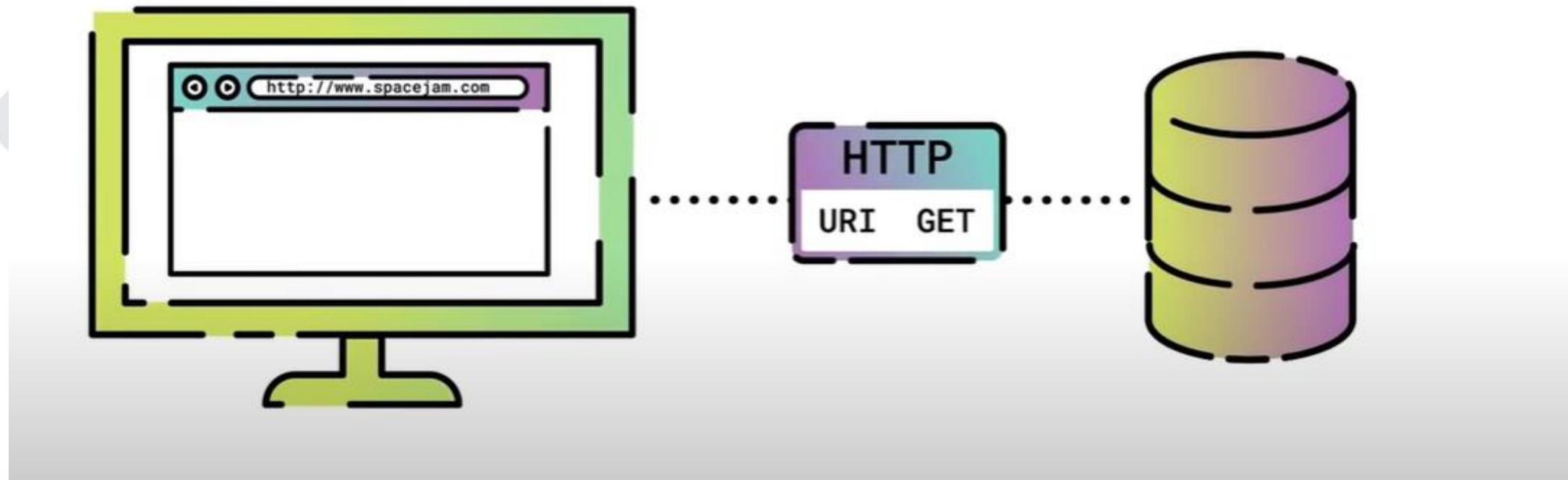


POSTMAN



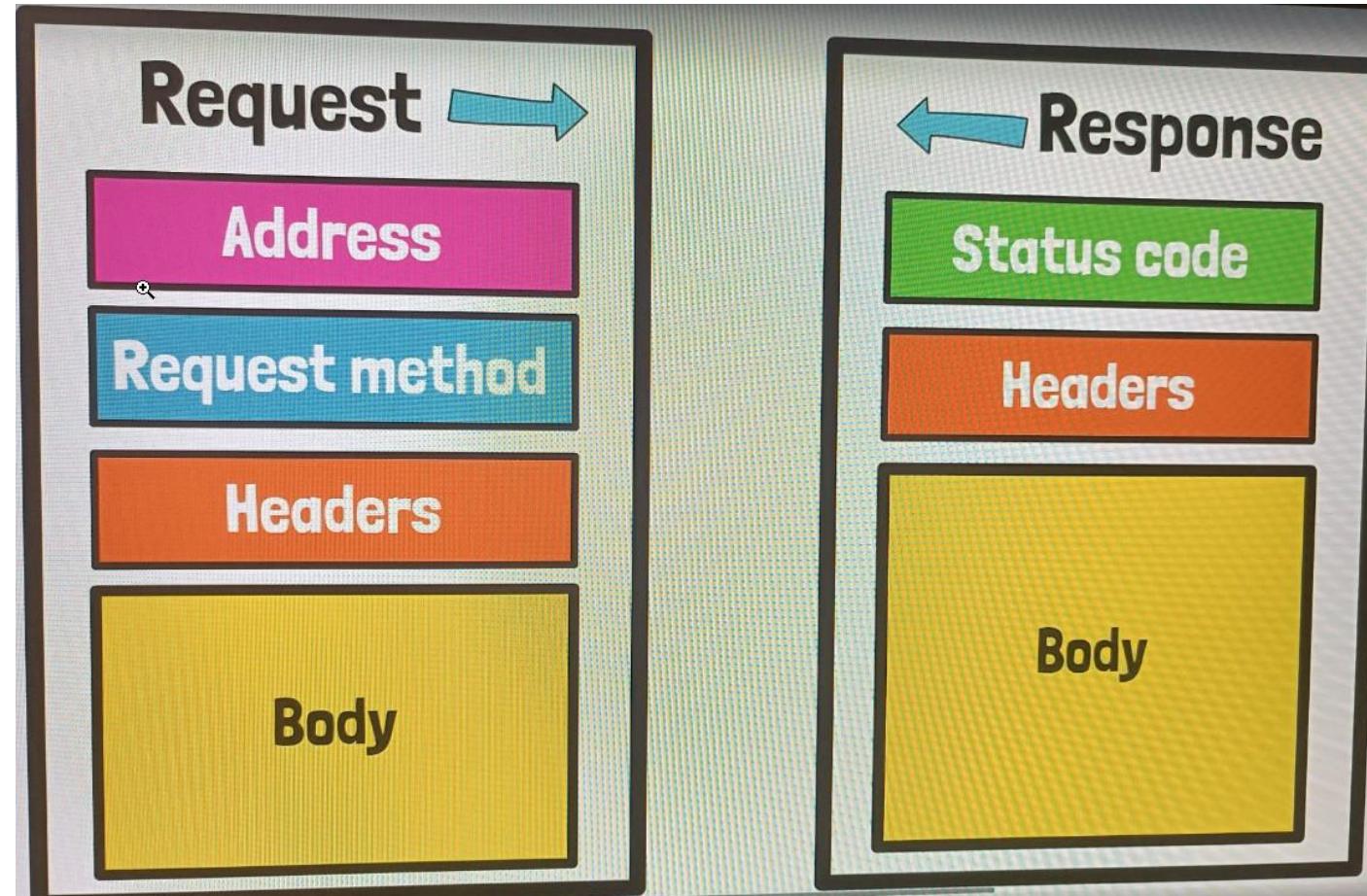
איך ממשק Web עובד: בקשה מצד השרת תענה בתשובה נתונים וקובציים מצד השרת :  
לדוגמה ייצור משתמש חדש במערכת של אתר e-commerce

# API



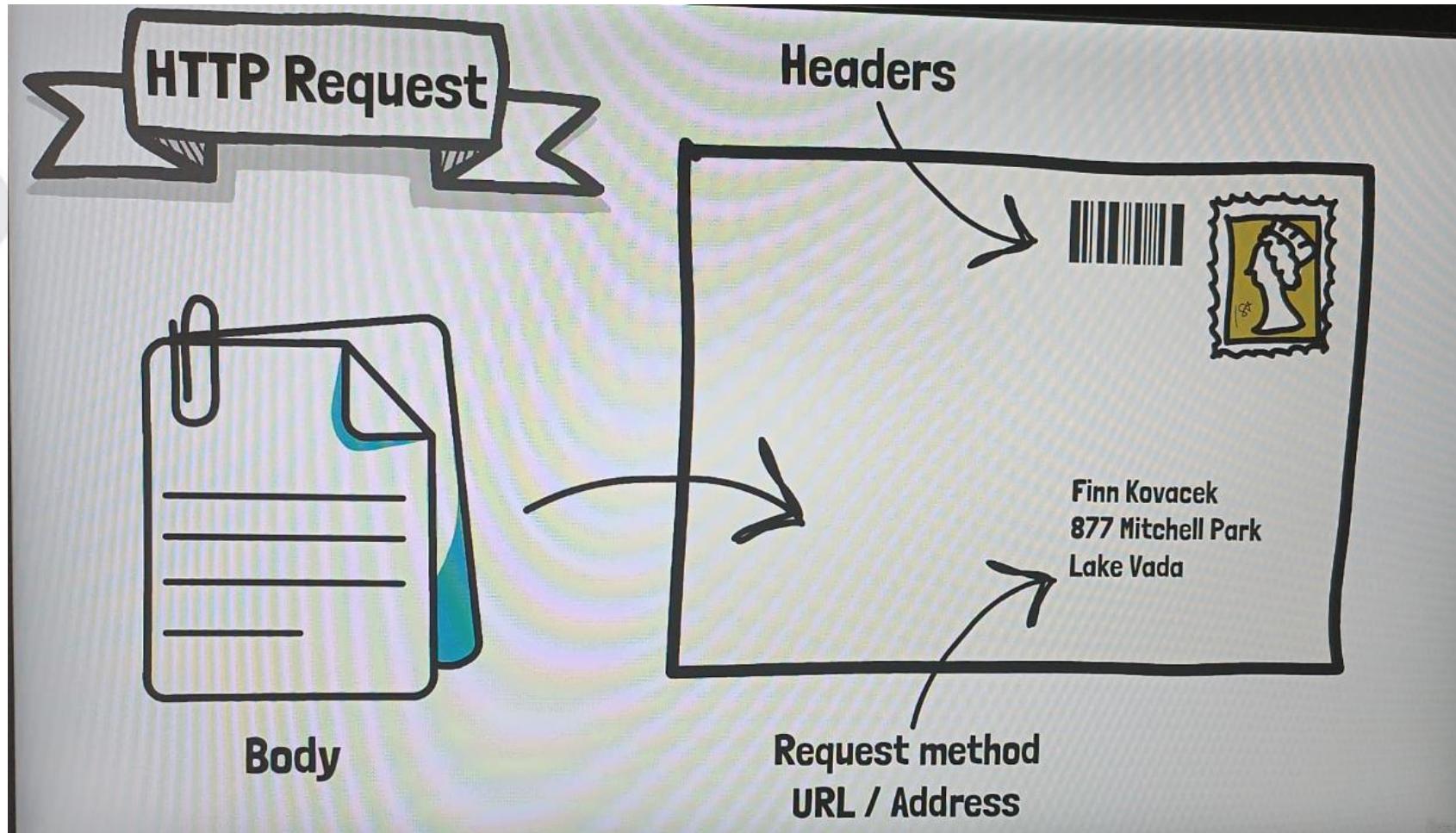
הלקוח שולח בקשה מסווג כלש/no error, כאשר רשות האינטרנט משתמש כ贊� or ובעוד השרת נכתבת כתובות תשתית התומכת בבקשת זו.

# REST HTTP Requests/Response



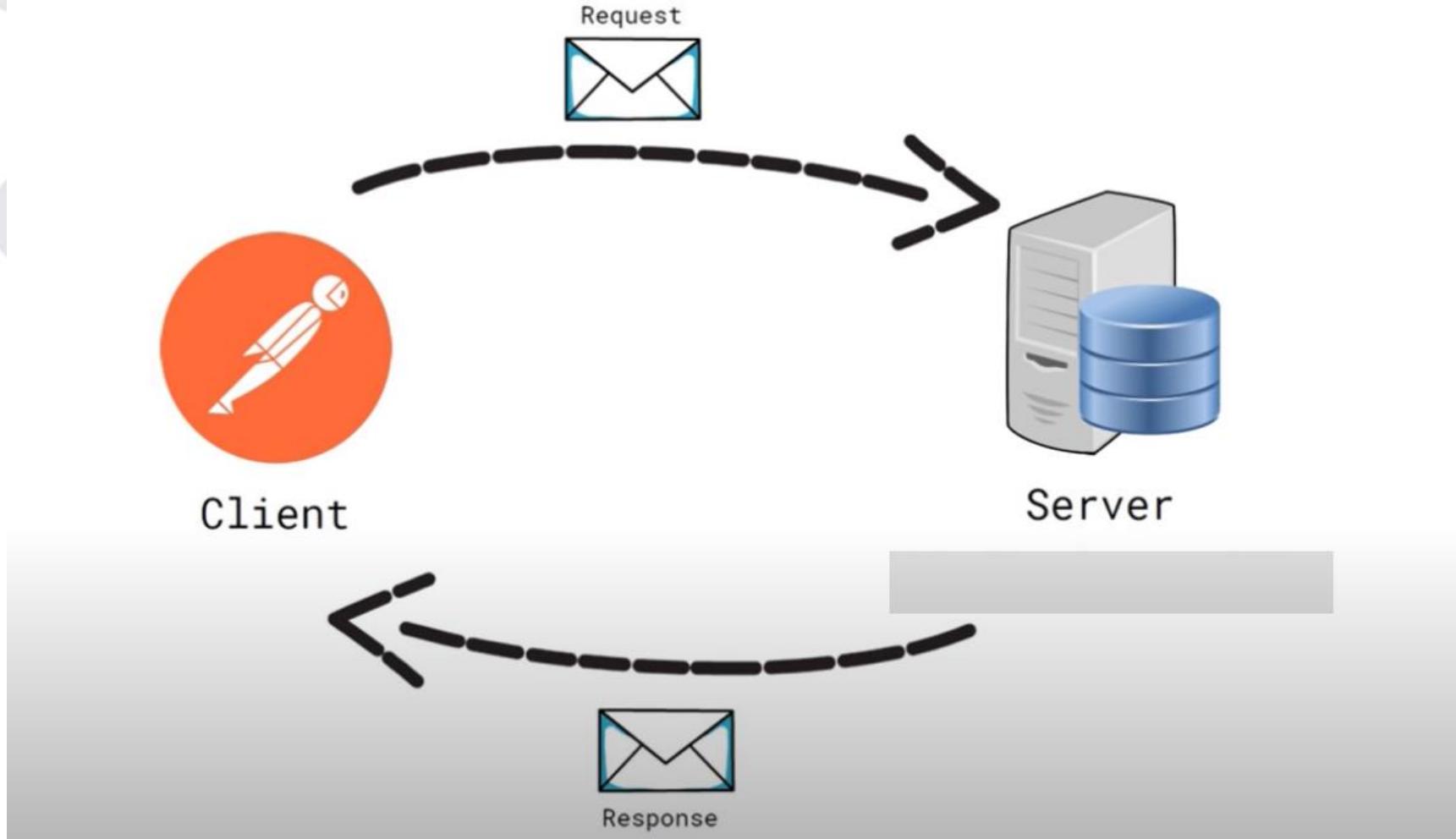
פרוטוקול REST שלרוב משמש אותנו, יש כלליים ומבנה קבועה גם לבקשת הצד הלקוח וגם לתשובות הצד השרת

# HTTP Requests/Response



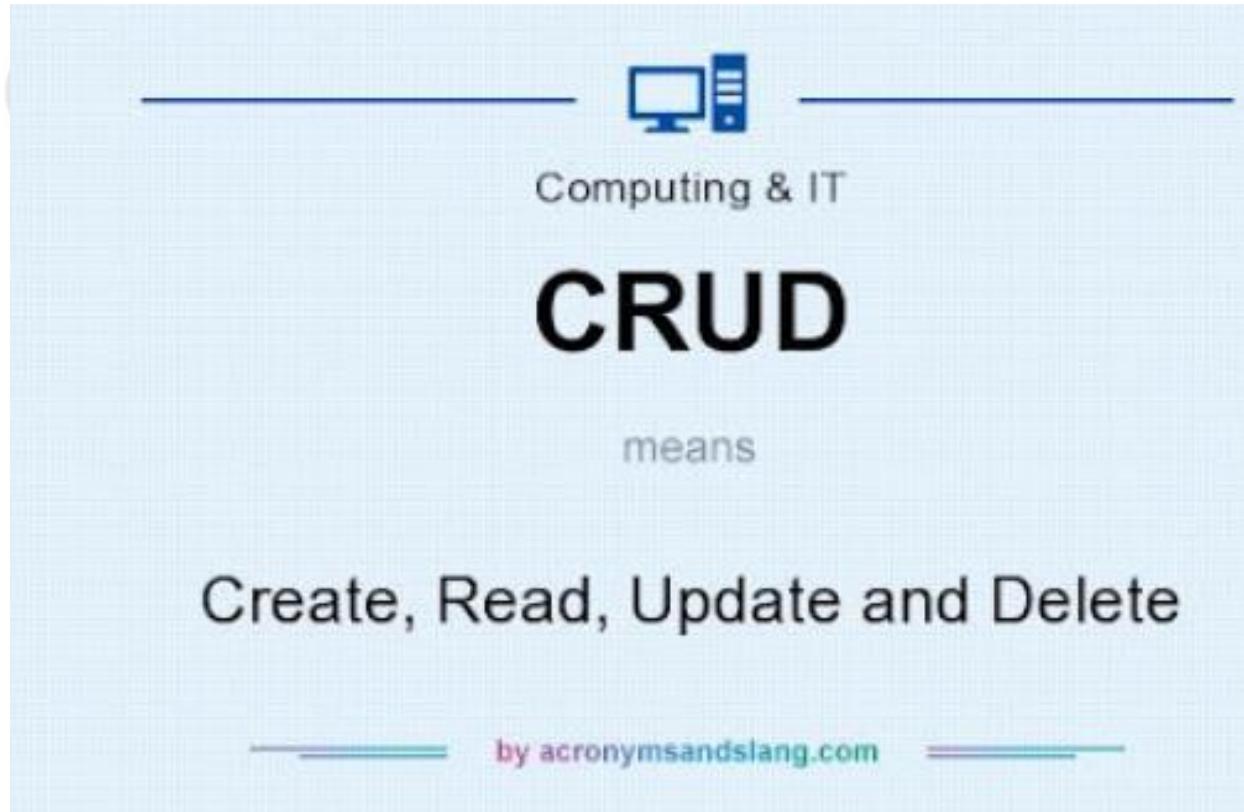
מה המשמעות של Headers: הגדנות מאפייני הבקשה: פורמט הבקשה ולאן לפנות מצד השרת

# Postman Practice



לפי סוג הבקשה תענה תשובה מצד השירות: דוגמאות: פורמט המידע החוזר JSON / XML /

# API



אפשר לקבל את סוג הבקשות של פרוטוקול RESTלסוגי הפעולות שהכרנו מעולם מושגி מסדי נתונים:SQL :  
Create / Update / Read / Delete

# HTTP request methods

## 1. GET:

- **Description:** The **GET** method requests a representation of a specific resource.
- **Purpose:** It is used to retrieve data from the server.
- **Example:** Fetching details of a user profile or retrieving a list of products.
- **Safety:** Safe (no side effects on the server).
- **Idempotent:** Yes (multiple identical requests yield the same result).

## 2. POST:

- **Description:** The **POST** method submits an entity (data) to a specified resource.
- **Purpose:** It often causes a change in state on the server (e.g., creating a new resource).
- **Example:** Creating a new user account or adding an item to a shopping cart.
- **Safety:** Not safe (may modify server state).
- **Idempotent:** No (multiple identical requests may create different resources).

# HTTP request methods

## 3. PUT:

- **Description:** The **PUT** method replaces all current representations of the target resource with the request payload.
- **Purpose:** Used for updating an existing resource.
- **Example:** Updating user profile information or modifying an article.
- **Safety:** Not safe (modifies server state).
- **Idempotent:** Yes (multiple identical requests yield the same result).

## 4. DELETE:

- **Description:** The **DELETE** method deletes the specified resource.
- **Purpose:** Removes a resource from the server.
- **Example:** Deleting a user account or removing a comment.
- **Safety:** Not safe (modifies server state).
- **Idempotent:** Yes (multiple identical requests yield the same result).

## 5. PATCH:

- **Description:** The **PATCH** method applies partial modifications to a resource.
- **Purpose:** Used for making specific updates to a resource.
- **Example:** Changing the status of an order or updating specific fields.
- **Safety:** Not safe (modifies server state).
- **Idempotent:** No (multiple identical requests may have different effects).

# Postman



Your personal and company data are protected

Postman is a powerful tool for testing and automating **API requests**. Let's dive into what it offers:

## 1. Automated API Testing with Postman:

- **Purpose:** Postman allows you to create test suites that can run repeatedly. These tests can include unit tests, functional tests, integration tests, end-to-end tests, regression tests, and more.
- **Benefits:**
  - **Streamline Development and QA:** Integrate automated testing into your CI/CD pipeline. Test every code push seamlessly.
  - **Scale Easily:** As your programs grow, automated testing helps maintain robustness and reduces the risk of breakage.
  - **Cost-Effective:** Spend less on manual QA, reduce lag time between development and QA, and minimize debugging in production.
- **How It Works:**
  - **Test Suites:** Organize your requests into Postman Collections. These collections allow you to run and automate a series of requests.
  - **Collaboration:** Postman Workspaces facilitate collaboration among teammates.



פוסטמן בرمת הגדרה: כל ניהול של ממשק ווב API: **REST API**: ועוד  
הוא כל מהאפשר כתיבה של ממשקים, ובנוסף להוסיף בדיקות מלאות כדי לוודא את ההתנהגות המוצופה.

# Postman

- **JavaScript Snippets:** Use built-in JavaScript snippets to test APIs without writing code.
- **Tools:**
  - **Postman App:** Manage and run your test workflows.
  - **Postman Monitoring:** Monitor APIs.
  - **Newman:** Command-line tool for running Postman collections.

## 2. Writing Tests in Postman:

- **Tests Tab:** Add tests to individual requests, collections, or folders. Tests execute after the request runs.
- **JavaScript:** Write test scripts in JavaScript. Validate responses, status codes, and more.
- **Example Test Script:**

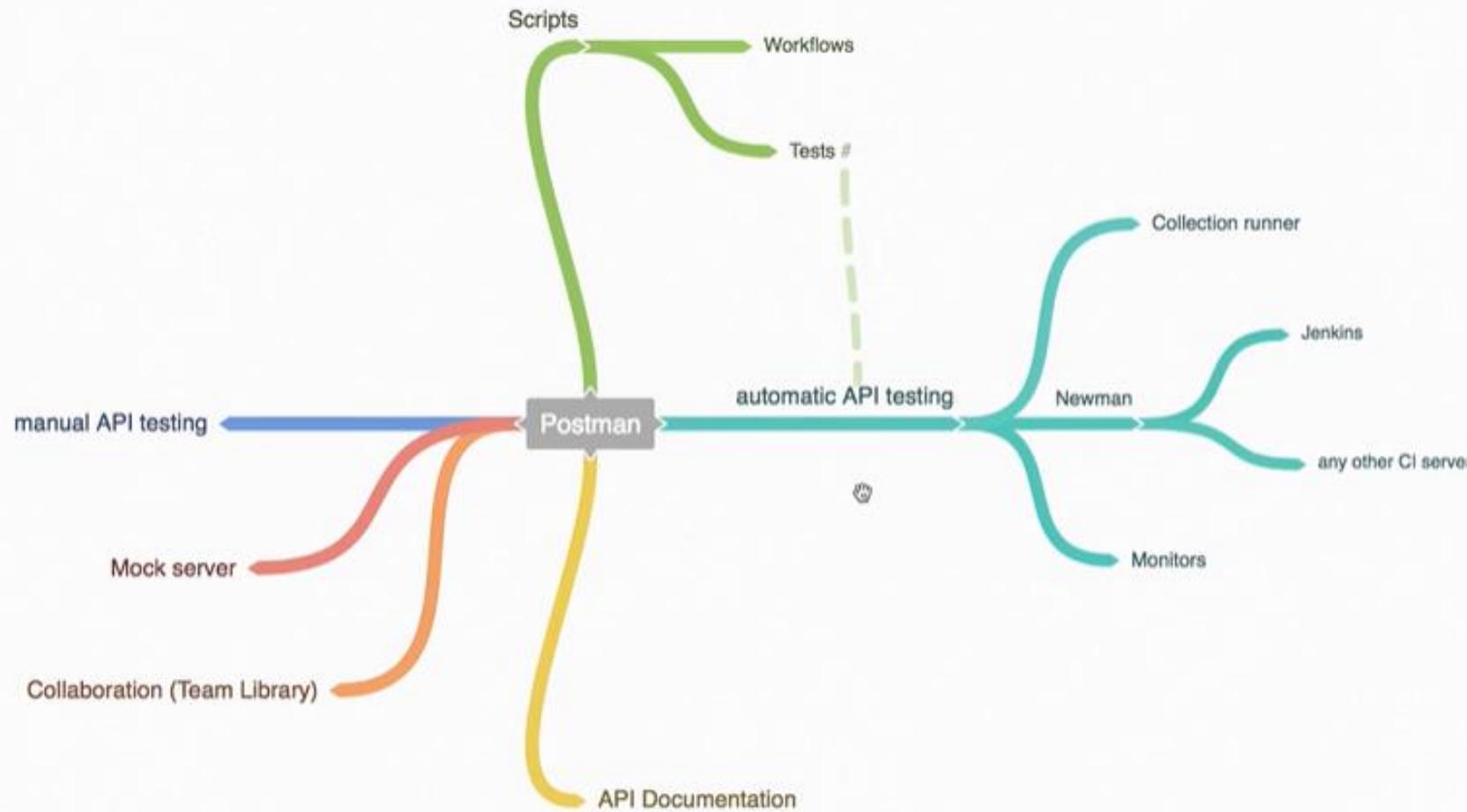
JavaScript



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

AI-generated code. Review and use carefully. [More info on FAQ](#).

# Postman - structure



# What is Postman?

## What is Postman?

Postman is an API platform for building and using APIs. Postman simplifies each step of the API lifecycle and streamlines collaboration so you can create better APIs—faster.

```
~/workspace ➔ curl --location --request POST 'https://simple-tool-rental-api.glitch.me/orders' --header 'Authorization: Bearer 204df14ba6ad5b12483bfc9acc947dd2fe5e9047c5bb2bf06face06d630b3690' --header 'Content-Type: application/json' --data-raw '{"toolId": 4643, "customerName": "John Doe"}'
```



MAN

# Postman SAAS platform

**SAAS : SOFTWARE AS A SERVICE**

- Universally Accessible From Any Platform
- No Need To Commute, You Can Work From Anyplace
- Excellent For Collaborative Working
- Vendor Provides Modest Software Tools
- Allows For Multi-Tenancy

PROS :

EcourseReview.com

# Postman: Advantages and disadvantages

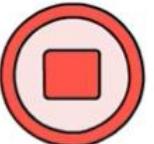


**Communicate with APIs** 



**User interaction** 



**Performance testing** 



**Security testing** 

# Practice Postman

GET `{{baseUrl}}/books?type=crime` Send

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

Query Params

KEY	VALUE	DESCRIPTION	...	Bulk E
<input checked="" type="checkbox"/> type	crime			
Key	Value	Description		

Body Cookies Headers (6) Test Results Status: 400 Bad Request Time: 265 ms Size: 313 B Save Response

Pretty Raw Preview Visualize JSON ↻

```
1 { "error": "Invalid value for query parameter 'type'. Must be one of: fiction, non-fiction." }
```

# Create Account for Postman

Enterprise user? [Sign in here](#)



**Sign In**

[Create Account instead?](#)

Email or Username

Password

Keep me signed in

[Trouble signing in?](#)

**Sign In**

or



**Sign in with Google**

Blocked from signing in? [Try on a web browser.](#)



Postman

# Your first request



Let's go ahead and start using an API in Postman.

# API Documentation

<https://github.com/vdespa/introduction-to-postman-course/blob/main/simple-books-api.md>

The screenshot shows a GitHub repository page for 'vdespa/introduction-to-postman-course'. The repository is public and has 196 stars, 137 forks, and 3 contributors. The 'Code' tab is selected, showing the file 'simple-books-api.md'. The file was last updated by vdespa on August 24, 2023. The content of the file is displayed below:

```
Simple Books API

This API allows you to reserve a book.

The API is available at https://simple-books-api.glitch.me

Endpoints

Status
```

# Download Postman

Download the app to quickly get started using the Postman API Platform. Or, if you prefer a browser experience, you can try the new web version of Postman.

## The Postman app

The ever-improving Postman app (a new release every two weeks) gives you a full-featured Postman experience.

[Download the App](#)

By downloading and using Postman, I agree to the [Privacy Policy](#) and [Terms](#).

Version 8.12.4 · [Release Notes](#) · [Product Roadmap](#)

Not your OS? Download for Mac ([macOS](#)) or Linux ([x64](#))

By clicking "Accept All Cookies", you agree to the storing of cookies on your device to enhance site navigation, analyze site usage, and assist in our marketing efforts.

[Accept All Cookies](#)

[Cookies Settings](#)

[Try the Web Version](#)

The screenshot shows the Postman web interface with the following details:

- Header:** Home, Workspaces, Reports, Explore
- Search Bar:** Search Postman
- Left Sidebar:** Collections, APIs, Environments, Mock Servers, Monitors, History
- Current Collection:** Twitter's Public Workspace > Twitter API v2 > Tweet Lookup > Single Tweet
- Request Details:** GET https://api.twitter.com/2/tweets/:id
- Query Params:** description, tweet.fields, expansions, media.fields, poll.fields, place.fields, user.fields
- Path Variables:** id (selected)
- Response Headers:** 200 OK, 468 ms, 734 B, Save
- Body:** (JSON response shown)

# Practice Postman

<https://reqres.in/>

The screenshot shows the Postman application interface. The address bar at the top displays the URL <https://reqres.in/>. The left sidebar lists various API endpoints categorized by method:

- GET**: LIST USERS, SINGLE USER, SINGLE USER NOT FOUND, LIST <RESOURCE>, SINGLE <RESOURCE>, SINGLE <RESOURCE> NOT FOUND.
- POST**: CREATE (highlighted with a yellow background).
- PUT**: UPDATE.
- PATCH**: UPDATE.
- DELETE**: DELETE.

On the right side, two JSON responses are shown in separate panes:

- The first pane shows a single user object:

```
{"name": "morpheus", "job": "leader"}
```
- The second pane shows a single user object with an ID and creation timestamp:

```
{"name": "morpheus", "job": "leader", "id": "434", "createdAt": "2024-03-17T18:50:39.747Z"}
```

A blue callout bubble with Hebrew text points to the POST method in the sidebar:

להוסיף את הבקשות הבאות, כולל  
בדיקות לפרויקט שיצרנו בכיתה



# API Documentation

<https://github.com/vdespa/Postman-Complete-Guide-API-Testing/blob/main/simple-grocery-store-api.md>

## Simple Grocery Store API

This API allows you to place a grocery order which will be ready for pick-up in the store.

The API is available at <https://simple-grocery-store-api.glitch.me>

### Endpoints

- Status
- Products
  - Get all products
  - Get a product
- Cart
  - Get a cart
  - Get cart items
  - Create a new cart
  - Add an item to cart
  - Modify an item in the cart
  - Replace an item in the cart



# My Workspace

The screenshot shows the Postman application interface. The left sidebar is titled "My Workspace" and contains a list of collections, environments, and other items. The main workspace shows a request for "FirstEP" in the "MyNewCollection\_temp" collection. The request is a GET method to "https://simple-books-api.glitch.me". The "Pre-request Script" tab is selected, showing a single line of code: "1". The "Body" tab shows a JSON response with a single key-value pair: "message": "Welcome to the Simple Books API.". The status bar at the bottom indicates a 200 OK status with a time of 663 ms and a size of 258 B.

Postman

File Edit View Help

Home Workspaces Reports Explore

Search Postman

My Workspace

New Import

MyNewCollection... GET FirstEP

MyNewCollection\_temp / FirstEP

GET https://simple-books-api.glitch.me

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

1

message: "Welcome to the Simple Books API."

Send

Cookies

Pre-request scripts are written in JavaScript, and are run before the request is sent.  
Learn more about pre-request scripts

SNIPPETS

Get an environment variable  
Get a global variable  
Get a variable  
Get a collection variable  
Set an environment variable  
Set a global variable  
Set a collection variable

Status: 200 OK Time: 663 ms Size: 258 B Save Response

# My Workspace

The screenshot shows the Postman application interface with a dark theme. The top navigation bar includes 'File', 'Edit', 'View', 'Help', 'Home', 'Workspaces', 'Reports', 'Explore', a search bar, and various icons for cloud storage, invites, and settings.

The main area is titled 'My Workspace' and contains the following sections:

- My Workspace**: A summary section with a title 'My Workspace' and a placeholder 'Add summary to briefly explain what this workspace is all about...'. It lists the following counts:
  - Collections: 14
  - APIs: 0
  - Environments: 12
  - Mock Servers: 3
  - Monitors: 0
- Get started**: A section with links to 'Create a request', 'Create a collection', 'Create an API', and 'Create an environment', along with a 'View More' link.
- Activity**: A timeline of recent events for the user 'Hagai Tregerman':
  - Edited the 'MyNewCollection\_testExample' collection (an hr ago)
  - Edited the 'MyTest1' environment (2 hrs ago)
  - Shared the 'MyTest1' environment to this workspace (2 hrs ago)
  - Added the 'MyTest1' environment (2 hrs ago)
  - Edited the 'MyNewCollection\_testExample' collection (View Changelog)
- Sharing**: A section titled 'Visibility' with options for 'Personal', 'Team', 'Private', and 'Public'.

# My Collections

The screenshot shows the Postman application interface. On the left, there's a sidebar with various sections: Home, Workspaces, Reports, Explore, My Workspace, New, Import, Collections (which is highlighted with a red box), APIs, Environments, Mock Servers, Monitors, and History. The main workspace shows a collection named "MyNewCollection\_temp" containing a single endpoint named "FirstEP". The "FirstEP" endpoint is a GET request to the URL <https://simple-books-api.glitch.me>. The "Pre-request Script" tab is selected, showing a single line of code: "1". To the right of the script, there's a snippet of code for "Pre-request scripts" which includes examples for getting environment variables, global variables, collection variables, and environment variables. Below the script, the "Body" tab is selected, showing the response body: "1", "2", "3", "message": "Welcome to the Simple Books API.". At the bottom right, the status bar indicates "Status: 200 OK Time: 663 ms Size: 258 B".

# Set Variables to Collection

The screenshot shows the Postman interface with a collection named "MyNewCollection\_testExample". The "Variables" tab is selected. A variable named "baseURL" is defined with an initial value of "https://simple-books-api.glitch.me". A tooltip indicates that this is "For all users". Another tooltip notes that this feature is "Only on my Postman". Below the variables, a "Query Params" table is shown for a specific request, where "baseURL" is listed with a value of "https://simple-books-api.glitch.me". The "Scope" dropdown shows it is limited to the current collection. A cursor is hovering over the "Set variable" button.

Variables

VARIABLE	INITIAL VALUE ⓘ	CURRENT VALUE ⓘ
baseURL	https://simple-books-api.glitch.me	https://simple-books-api.glitch.me

For all users

Only on my Postman

baseURL

KEY	Name	DESCRIP
Key	baseUrl	Description
Value	https://simple-books-api.glitch.me	
Scope	Collection: Simple ...	

Set variable

Body Cookies Headers (6) Test Results

Status: 200 OK Time: 310ms

# Shortcuts

The screenshot shows the Postman application interface. At the top, there is a navigation bar with links for Home, Workspaces, API Network, and Explore, along with a search bar labeled "Search Postman". Below the navigation bar, a message says "Hey there, night owl!" followed by "Pick up where you left off, catch up with your team's work." A sidebar on the left lists "red-capsule-721812" workspace details, including its URL and an "Invite" button, and links for Workspaces, Private API Network, API Governance, API Security, Integrations, and Reports.

A modal window titled "SETTINGS" is open, specifically the "Shortcuts" tab. This window contains a table of keyboard shortcuts categorized into Request, Sidebar, and other sections. The "Request" section includes shortcuts for Reopen Last Closed Tab (Ctrl + Shift + t), New Runner Tab (Ctrl + Shift + R), Request URL (Ctrl + L), Save Request (Ctrl + S), Save Request As (Ctrl + Shift + S), Send Request (Ctrl + Enter), Send And Download Request (Ctrl + Alt + Enter), Resize Request or Response Pane (Alt + scroll / Win + Alt + scroll), Scroll To Request (Ctrl + Alt + ↑), Scroll To Response (Ctrl + Alt + ↓), and Sidebar (Search Sidebar - Ctrl + F, Toggle Sidebar - Ctrl + \). Other sections include General, Themes, Data, Add-ons, Certificates, Proxy, Update, and About.

Below the sidebar, there are sections for "Explore popular APIs" featuring the Salesforce Platform and "Intro to writing tests" with a note about forks and watchers.

# REST Verbs

The screenshot shows the Postman application interface. On the left, there's a sidebar with navigation links: Home, Workspaces, Reports, Explore, My Workspace (selected), New, Import, Collections, APIs, Environments, Mock Servers, Monitors, and History. The main workspace shows a collection named "MyNewCollection\_temp / FirstEP Status". Inside this collection, there is a single request named "GET FirstEP Status". The request details pane shows the method "GET" highlighted, along with "Headers (7)", "Body", "Pre-request Script", "Tests", and "Settings". Below the request details, there's a table with columns "VALUE" and "DESCRIPTION". At the bottom of the request details pane, there are three status indicators: "Content-type is present" (green), "Content-Type header is application/json" (green), and "Response time is less than 200ms" (green). The status bar at the bottom right indicates "Status: 200 OK Time: 417 ms Size: 226 B Save Response". The top right of the screen has a toolbar with icons for invite, settings, notifications, and upgrade.

# HTTP Response

<https://developer.mozilla.org/en-US/docs/Web/HTTP>Status>

The screenshot shows the Postman application interface. At the top, there are several tabs: 'MyNewCollection\_temp' (highlighted), 'GET FirstEP' (highlighted), 'MyTest1' (highlighted), 'GET FirstEP Status' (highlighted), 'Overview' (disabled), 'Contacts List\_test...', and '+'. On the far right, it says 'No Environment'.

The main area shows a 'FirstEP Status' request. The method is 'GET' and the URL is '{{baseURL}}/status'. Below the URL, there are tabs for 'Params', 'Authorization', 'Headers (7)', 'Body', 'Pre-request Script', 'Tests', and 'Settings'. The 'Params' tab is selected, showing a table with one row:

KEY	VALUE	DESCRIPTION
Key	Value	Description

At the bottom left, there are tabs for 'Body' (highlighted), 'Cookies', 'Headers (6)', and 'Test Results (5/5)'. The 'Body' tab has sub-options: 'Pretty' (selected), 'Raw', 'Preview', 'Visualize', and 'JSON'. The JSON preview shows:1: {  
2: "status": "OK"  
3: }

At the bottom right, there is a status bar with 'Status: 200 OK', 'Time: 417 ms', 'Size: 226 B', and a 'Save Response' button. There are also icons for copy, search, and refresh.

# HTTP Response

<https://developer.mozilla.org/en-US/docs/Web/HTTP/Status>

## HTTP Status Codes

**Level 200 (Success)**

**200 : OK**

**201 : Created**

**203 : Non-Authoritative  
Information**

**204 : No Content**

**Level 400**

**400 : Bad Request**

**401 : Unauthorized**

**403 : Forbidden**

**404 : Not Found**

**409 : Conflict**

**Level 500**

**500 : Internal Server Error**

**503 : Service Unavailable**

**501 : Not Implemented**

**504 : Gateway Timeout**

**599 : Network timeout**

**502 : Bad Gateway**

# Response Errors

<https://developer.mozilla.org/en-US/docs/Web/HTTP/Status>

## HTTP response status codes

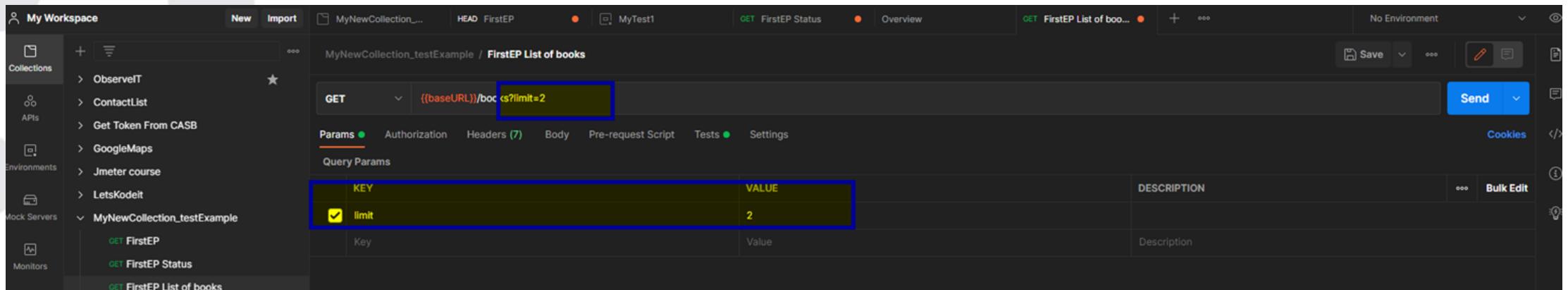
HTTP response status codes indicate whether a specific [HTTP](#) request has been successfully completed. Responses are grouped in five classes:

1. [Informational responses](#) ( 100 – 199 )
2. [Successful responses](#) ( 200 – 299 )
3. [Redirection messages](#) ( 300 – 399 )
4. [Client error responses](#) ( 400 – 499 )
5. [Server error responses](#) ( 500 – 599 )

# Query Param

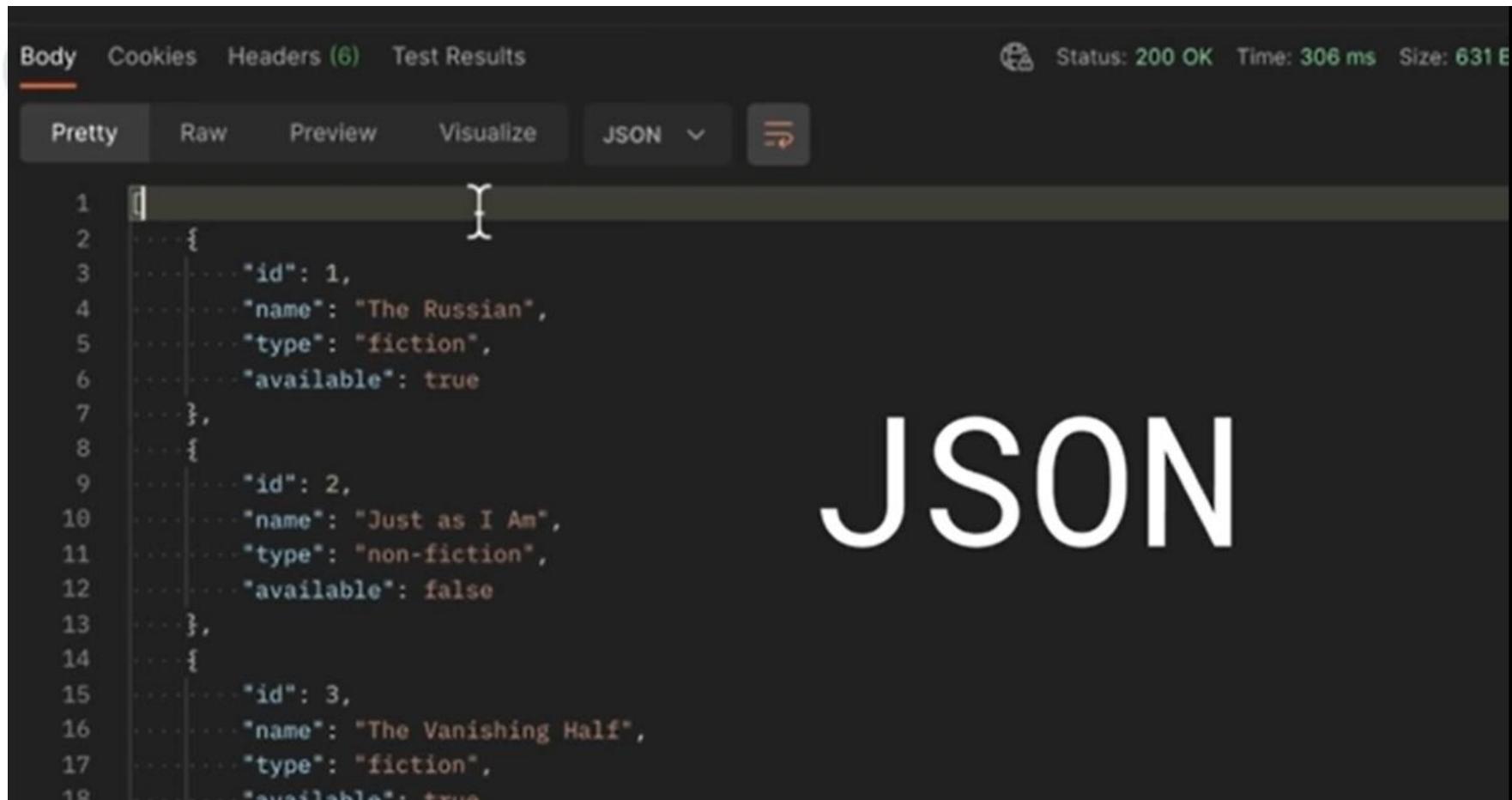
The screenshot shows the Postman application interface. On the left, there's a sidebar with sections for Collections, APIs, Environments, Mock Servers, and Monitors. The 'Collections' section is expanded, showing items like 'ObserveIT', 'ContactList', 'Get Token From CASB', 'GoogleMaps', 'Jmeter course', 'LetsKodeIt', and 'MyNewCollection\_testExample'. 'MyNewCollection\_testExample' is expanded, showing three test cases: 'FirstEP', 'FirstEP Status', and 'FirstEP List of books'. The main workspace is titled 'MyNewCollection\_testExample / FirstEP List of books'. It shows a GET request with the URL {{baseUrl}}/books?limit=2. Below the URL, under the 'Params' tab, is a table for 'Query Params' with one row: KEY limit and VALUE 2. There are tabs for Authorization, Headers (7), Body, Pre-request Script, Tests, and Settings. At the top right, there are buttons for Save, Send, and a gear icon. The status bar at the bottom indicates 'No Environment'.

# Query Param – Practice: type



The screenshot shows the Postman application interface. On the left, the 'My Workspace' sidebar lists various collections, APIs, environments, mock servers, and monitors. The main workspace displays a collection named 'MyNewCollection\_testExample' with a single endpoint named 'FirstEP List of books'. The endpoint is a GET request to the URL `/{{baseURL}}/books?limit=2`. In the 'Params' tab of the request details, there is a table for 'Query Params' with one entry: 'limit' (Key) with value '2' (Value). The entire URL and the 'limit' row in the table are highlighted with a blue selection box.

# Response



The screenshot shows a browser developer tools interface with a blue header bar. The header includes tabs for "Body", "Cookies", "Headers (6)", and "Test Results". On the right side of the header, it displays "Status: 200 OK", "Time: 306 ms", and "Size: 631 E". Below the header, there are several tabs: "Pretty" (which is selected), "Raw", "Preview", "Visualize", and "JSON". The main content area shows a JSON array with three objects, each representing a book. The JSON is displayed in a numbered, pretty-printed format.

```
1 [ ]  
2 { }  
3   "id": 1,  
4   "name": "The Russian",  
5   "type": "fiction",  
6   "available": true  
7 },  
8 { }  
9   "id": 2,  
10  "name": "Just as I Am",  
11  "type": "non-fiction",  
12  "available": false  
13 },  
14 { }  
15   "id": 3,  
16   "name": "The Vanishing Half",  
17   "type": "fiction",  
18   "available": true
```

# JSON

JSON הוא בסיס הכל פורמט להעברת נתונים בין שרת לבין סקריפט של JavaScript. ArrayBuffer מאפשר לשימוש בו להעברת נתונים בין שרת לשרת JSON. זה בסיס הכל Java API לא שזה חשוב לזכור את זה (ומזכיר באובייקט JavaScript לשיש לו תכונות שונות עם ערכי).

# Response Errors

<https://developer.mozilla.org/en-US/docs/Web/HTTP>Status>

The screenshot shows a POST request in Postman with the URL `{{baseUrl}}/books?type=crime`. The 'Params' tab is selected, showing a single query parameter `type` with the value `crime`. The 'Body' tab is selected, displaying the response. The response status is `400 Bad Request`, and the error message is `"error": "Invalid value for query parameter 'type'. Must be one of: fiction, non-fiction."`.

GET {{baseUrl}}/books?type=crime Send

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

Query Params

KEY	VALUE	DESCRIPTION	...	Bulk Ed
<input checked="" type="checkbox"/> type	crime			

Body Cookies Headers (6) Test Results

Pretty Raw Preview Visualize JSON ↻

```
1 [ { "error": "Invalid value for query parameter 'type'. Must be one of: fiction, non-fiction." } ]
```

# Path Variable

The screenshot shows the Postman application interface for a "Simple Book API / Get single book" request. The URL field contains `{baseUrl}/books/:bookId`, where `:bookId` is highlighted with a yellow box. Below the URL, there are tabs for Params, Authorization, Headers (5), Body, Pre-request Script, Tests, and Settings. Under the Params tab, there is a table for "Query Parameters". Under the Path Variables tab, there is a table:

KEY	VALUE	DESCRIPTION
bookId	Value	Description

# Post Requests

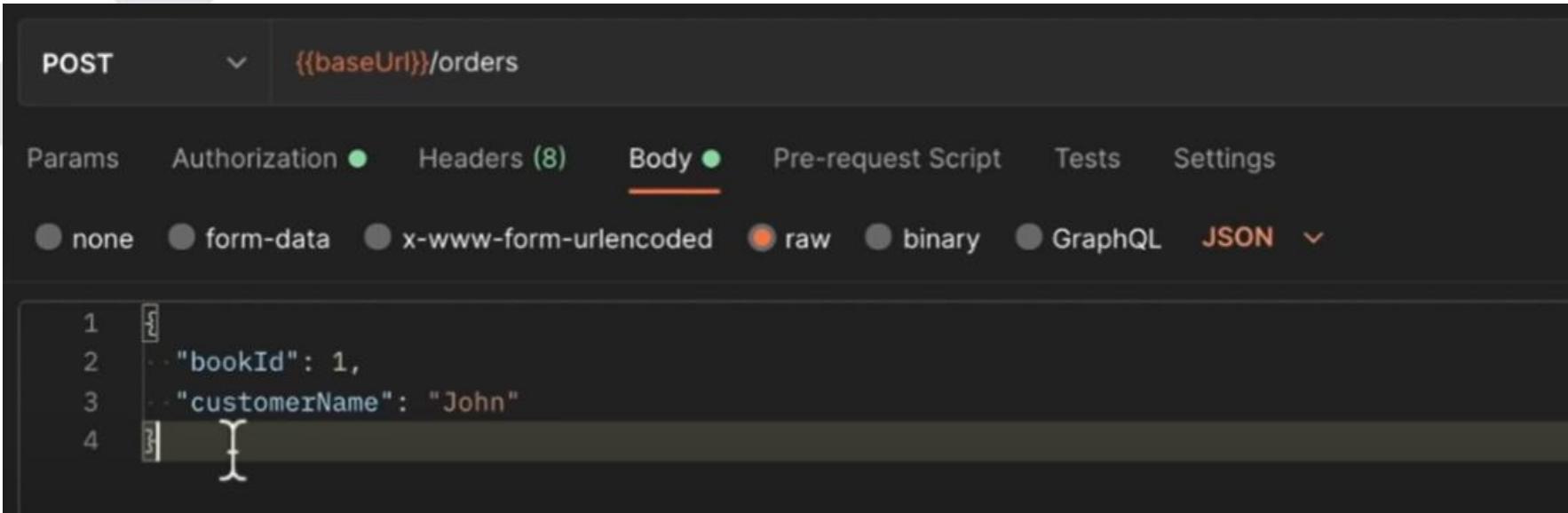
The screenshot shows the Postman application interface for making a POST request. The request details are as follows:

- Method:** POST
- URL:** `{{baseUrl}}/orders`
- Body:** The Body tab is selected, showing the following JSON payload:

```
1 {  
2   "bookId": 1,  
3   "customerName": "John"  
4 }
```

# Post Requests - Practice

Check error if insert invalid id for order



The screenshot shows the Postman interface with a POST request configuration. The method is set to POST and the URL is {{baseUrl}}/orders. The 'Body' tab is selected, showing a raw JSON payload:

```
1 {  
2   "bookId": 1,  
3   "customerName": "John"  
4 }
```

# Random Variables

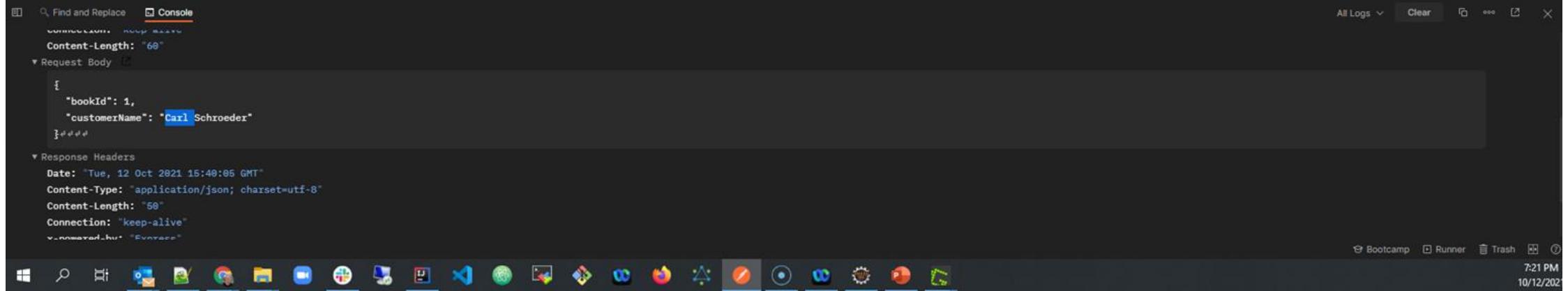


```
POST {{baseUrl}}/orders

Params Authorization Headers (10) Body Pre-request Script Tests Settings
none form-data x-www-form-urlencoded raw binary GraphQL JSON ▾

1
2   "bookId": 1,
3   "customerName": "{{randomFullName}}"
4
5
6
```

# Postman console

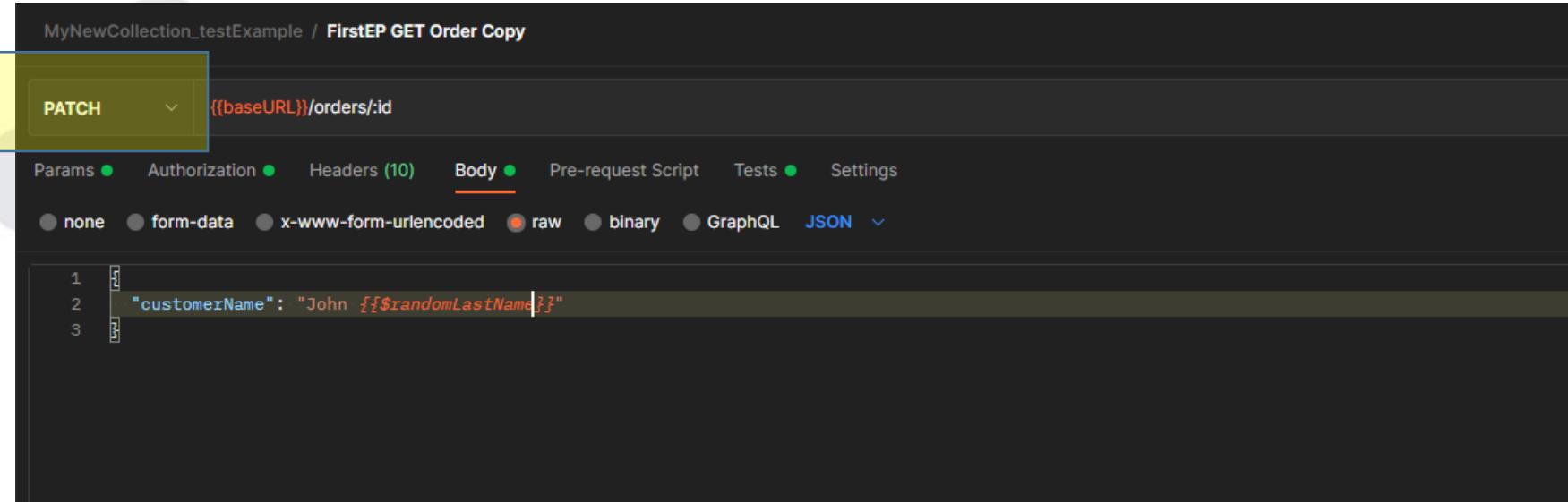


כל חשוב הוא הkonsole שマーה את הערכים שהמשתנים שלו מכילים, אנו יכולים להדפיס בעזרת פקודת `console.log()`.  
משתנים שאנו רוצים לראות את תוכנם.

# Practice

- \* Get all orders made
- \* Get specific order made

# Patch Action (Update)



# Test and Manipulation

The screenshot shows the Postman interface with a blue header bar containing the title "Test and Manipulation". Below the header is a search bar with placeholder text "{{baseURL}}/status". The main workspace is divided into several sections:

- Tests:** A tab labeled "Tests" is selected, highlighted with a yellow box. It contains a block of JavaScript code for testing the response status and body content.
- Body:** A tab labeled "Body" is selected, highlighted with a yellow box. It displays the JSON response body: {"status": "OK"}. Other tabs like "Cookies", "Headers", and "Test Results" are also visible.
- Response Status:** A status bar at the bottom right shows "Status: 200 OK", "Time: 873 ms", and "Size: 226 B".
- Snippets Panel:** A sidebar on the right is titled "Cookies" and contains a list of snippets for manipulating environment variables, global variables, and collection variables.

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

pm.test("Body matches string", function () {
    pm.expect(pm.response.text()).to.include('{\"status\":\"OK\"}');
});

pm.test("Content-Type is present", function () {
    pm.response.to.have.header("Content-Type");
});

pm.test("Content-Type header is application/json", () => {
    pm.expect(pm.response.headers.get('Content-Type')).to.contain('application/json');
});

pm.test("Response time is less than 200ms", function () {
    pm.expect(pm.response.responseTime).to.be.below(800);
});
```

# Test and Manipulation

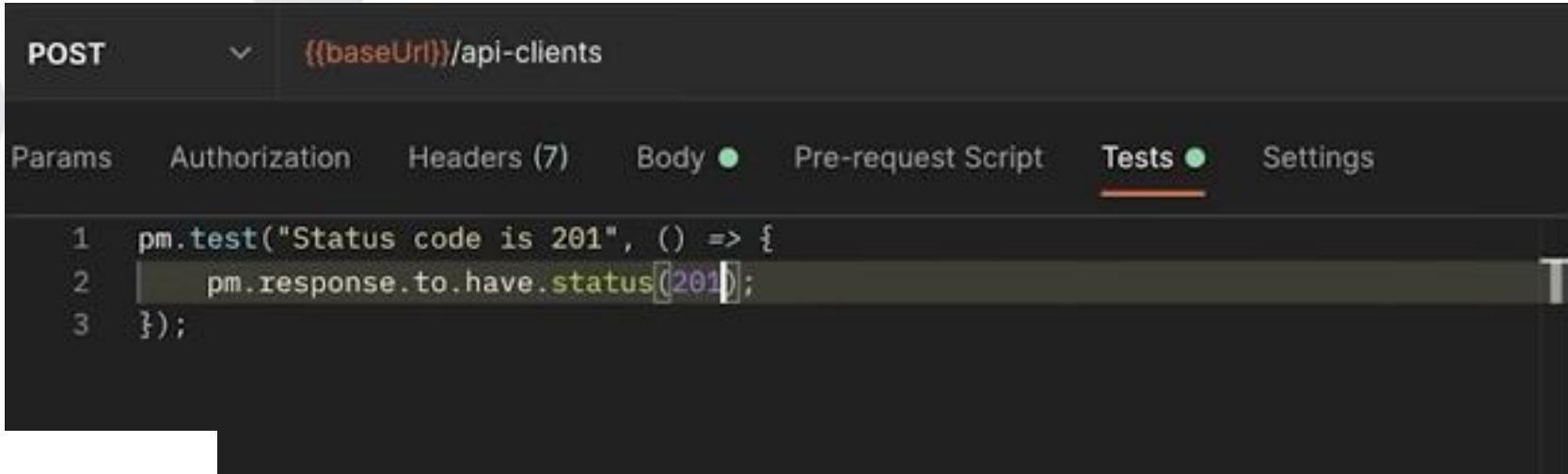
## Test Section

The screenshot shows the Postman interface with a dark theme. At the top, there is a navigation bar with a dropdown menu and a search bar containing the URL `{{baseUrl}}/books`. Below the navigation bar, there are tabs for `Params`, `Authorization`, `Headers (7)`, `Body`, `Pre-request Script`, `Tests` (which is currently selected), and `Settings`. The `Tests` tab contains the following script:

```
1 const response = pm.response.json();
2 console.log(response[1].id)
3 console.log(response[1].name)
4 console.log(response[1].type)
5
6
```

# Test and Manipulation

Practice: add at least one **test** for each request we made.



The screenshot shows the Postman interface with a dark theme. A POST request is being made to `{{baseUrl}}/api-clients`. The 'Tests' tab is active, containing the following PM Test script:

```
1 pm.test("Status code is 201", () => {
2   pm.response.to.have.status(201);
3});
```

# Star wars API

<https://swapi.dev/>

The screenshot shows a web browser window titled "SWAPI - The Star Wars API". The address bar contains the URL "https://swapi.dev". The main content area displays the following text:

All the Star Wars data you've ever wanted:  
Planets, Spaceships, Vehicles, People, Films and Species  
From all SEVEN Star Wars films  
Now with The Force Awakens data!

A "Try it now!" button is present, along with a text input field containing "https://swapi.dev/api/people/1" and a "request" button. Below this, a "Result:" section shows a JSON response for a person object:

```
height : "172",
"mass": "77",
"hair_color": "blond",
"skin_color": "fair",
"eye_color": "blue",
"birth_year": "19BBY",
"gender": "male",
"homeworld": "https://swapi.dev/api/planets/1/",
"films": [
    "https://swapi.dev/api/films/1/",
    "https://swapi.dev/api/films/2/",
    "https://swapi.dev/api/films/3/",
    "https://swapi.dev/api/films/6/"
],
"species": [],
"vehicles": [
    "https://swapi.dev/api/vehicles/14/",
    "https://swapi.dev/api/vehicles/30/"
```

At the bottom of the page, there are three sections: "What is this?", "How can I use it?", and "What happened with old swapi.co?".

**What is this?**  
The Star Wars API, or "swapi" (Swah-pee) is the world's first quantified and programmatically-accessible data source for all the data from the

**How can I use it?**  
All the data is accessible through our HTTP web API. Consult our [documentation](#) if you'd like to get started.

**What happened with old swapi.co?**  
swapi.co is no longer supported and maintained anymore. But since so many projects and tutorials used it as their educational playground, this is an

# Twilio.com API: Whatsapp / SMS

<https://console.twilio.com>

<https://www.twilio.com/en-us/blog/send-test-http-requests-twilio-sms-postman>

The use of Postman in this article will replace the code below:

Text

```
1 curl -X POST https://api.twilio.com/2010-04-01/Accounts/{{ACCOUNT_SID}}/Messages.json  
2 --data-urlencode "From=+<TWILIO_PHONE_NUMBER>" \  
3 --data-urlencode "Body=body" \  
4 --data-urlencode "To=+<YOUR_PHONE_NUMBER>" \  
5 -u your_account_sid:your_auth_token
```

```
$ curl wttr.in/London  
$ curl wttr.in/Moscow  
$ curl wttr.in/Salt+Lake+City
```

# Coverage

5	SOAP Requests using POSTMAN
	Multipart FileUpload using Postman
	Interceptor & Proxy to capture browser network traffic
	Mock Servers & Mock Requests in Postman
	Using Authentication in POSTMAN
	Data Driven testing using Postman
6	Scripting fundamentals in POSTMAN
	Practical Examples on Scripting in POSTMAN
	Newman for CLI execution
	Source control with Git, GitHub
	Jenkins,HTML Reports - Executing collections in Jenkins



# Make HTTP calls using the SOAP protocol

- <https://learning.postman.com/docs/sending-requests/soap/making-soap-requests/>

<https://copilot.microsoft.com/>:

- \* What is SOAP API ?
- \* which one is older SOAP or REST ?

	SOAP Representational State Transfer	REST Representational State Transfer
When	late 1990s.	early 2000s
Who	Microsoft	Dr. Roy Fielding
key principles	<p>Structured and Formalized:</p> <ul style="list-style-type: none"><li>● <b>Envelope:</b> Defines the overall structure of the message.</li><li>● <b>Encoding:</b> Specifies rules for expressing data types.</li><li>● <b>Requests:</b> Describes how each SOAP API request is structured.</li><li>● <b>Responses:</b> Outlines the structure of SOAP API responses.</li></ul>	<ul style="list-style-type: none"><li>● HTTP verbs (such as GET, PUT, POST, DELETE) and URIs (resources) for communication. Each request receives an HTTP <b>response</b> with a <b>status code</b> and a <b>body</b></li><li>● Stateless: Each request is self-descriptive, containing enough context for the server</li><li>● Client-Server:</li><li>● Cacheable</li></ul>
Language	XML	JSON

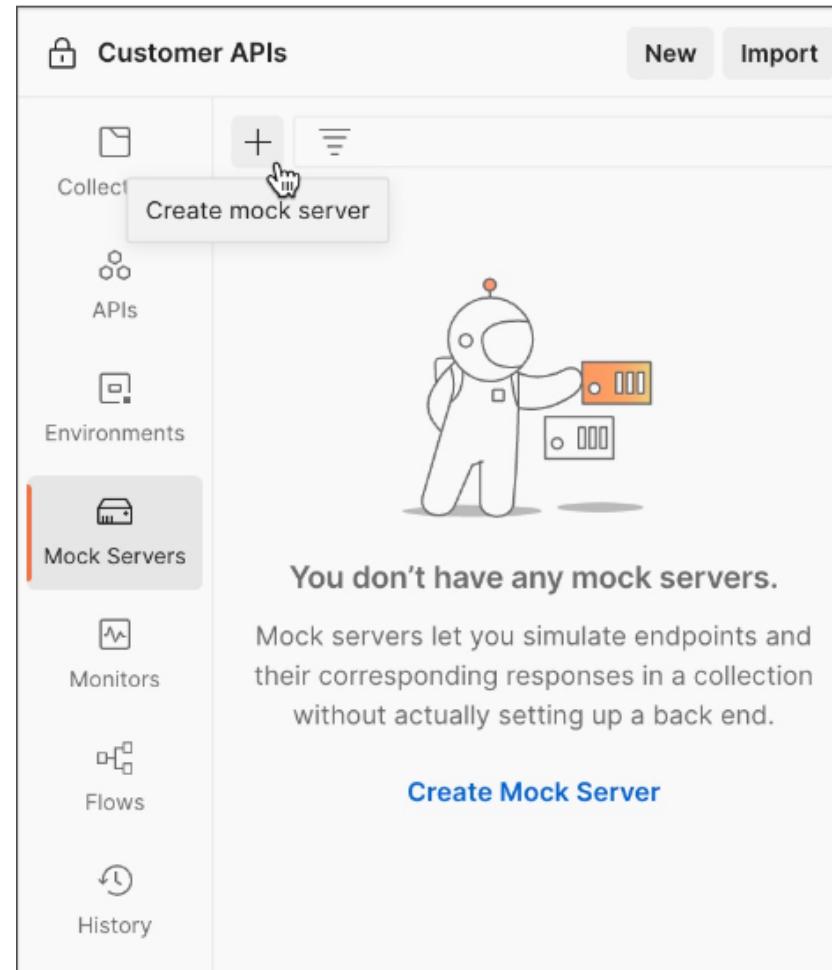
# Testing file uploads with Postman (multipart/form-data)

<https://apidog.com/blog/how-to-upload-a-file-and-json-data-in-postman/>

The screenshot shows the Postman application interface. At the top, there is a header bar with a 'POST Upload a file' button, a '+' icon, and an 'env' dropdown. To the right of the header is a dropdown for 'No Environment'. Below the header, the URL 'http://postman-echo.com/post' is entered into the address bar, preceded by an 'HTTP' icon and the text 'Upload a file via POST request / Upload a file'. To the right of the URL are 'Save' and 'Edit' buttons. The main area shows a 'POST' method selected in a red box, and the URL 'http://postman-echo.com/post'. Below this, tabs for 'Params', 'Authorization', 'Headers', 'Body' (which is also highlighted with a red box), 'Pre-request Script', 'Tests', and 'Settings' are visible. Under the 'Body' tab, a radio button for 'form-data' is selected, indicated by a red arrow pointing to it. Other options shown are 'none', 'x-www-form-urlencoded', 'raw', 'binary', and 'GraphQL'. At the bottom, a table has columns for 'Key', 'Value', and 'Description', with a 'Bulk Edit' button to the right.

# Configure and use a Postman mock server

<https://learning.postman.com/docs/designing-and-developing-your-api/mock-data/setting-up-mock/>



# Newman the cli companion for postman

<https://www.npmjs.com/package/newman>

Newman is a command-line collection runner for Postman. It allows you to effortlessly run and test a Postman collection directly from the command-line. It is built with extensibility in mind so that you can easily integrate it with your continuous integration servers and build systems.

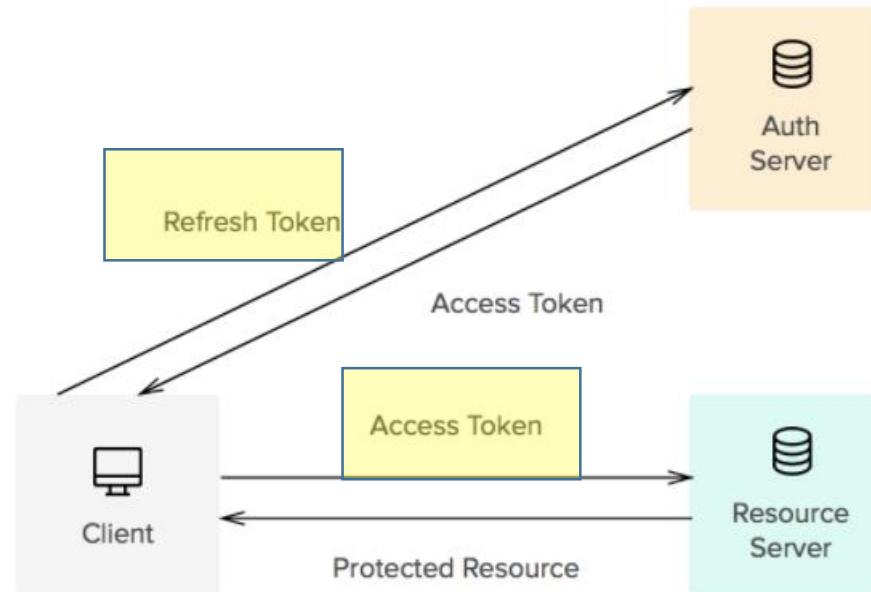
```
C:\Data\Newman>newman run dataaccess.com.postman_collection.json
newman

dataaccess.com

→ example_11
  POST https://www.dataaccess.com/webservicesserver/NumberConversion.wso [200 OK, 588B, 1499ms]
    ✓ Status code is 200
    ✓ Body matches string: NumberToWordsResult
```

	executed	failed
iterations	1	0
requests	1	0
test-scripts	2	0
prerequest-scripts	1	0
assertions	2	0
total run duration: 1583ms		
total data received: 323B (approx)		
average response time: 1499ms [min: 1499ms, max: 1499ms, s.d.: 0µs]		

# Access Token

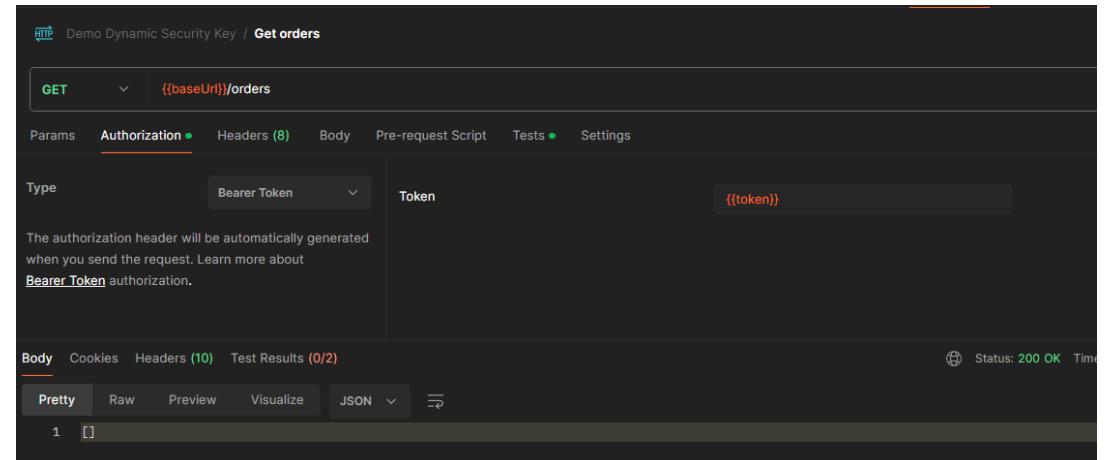


# API Authentication

<https://github.com/vdespa/Postman-Complete-Guide-API-Testing/blob/main/simple-grocery-store-api.md#API-Authentication>

Some endpoints may require authentication. To submit or view an order, you need to register your API client and obtain an access token.

The endpoints that require authentication expect a bearer token sent in the `Authorization` header.

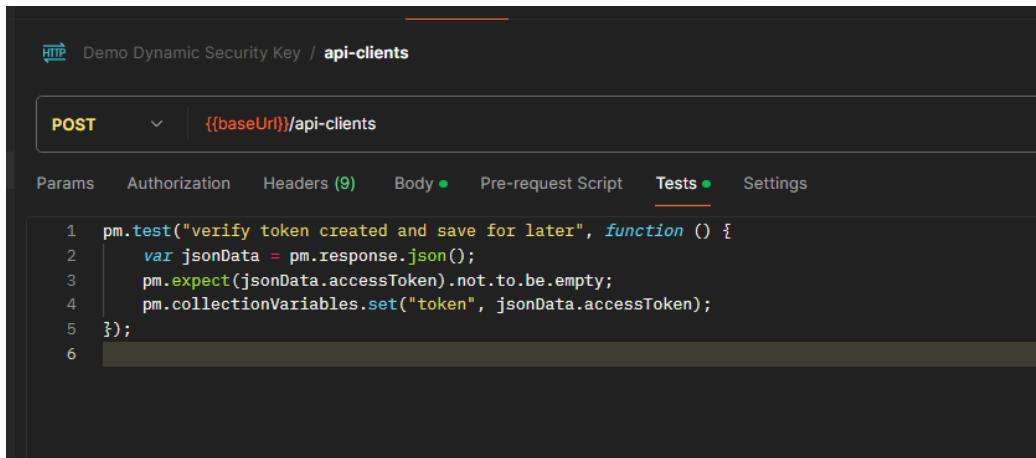


# API Authentication

<https://github.com/vdespa/Postman-Complete-Guide-API-Testing/blob/main/simple-grocery-store-api.md#API-Authentication>

Some endpoints may require authentication. To submit or view an order, you need to register your API client and obtain an access token.

The endpoints that require authentication expect a bearer token sent in the `Authorization` header.



The screenshot shows the Postman interface with a dark theme. A POST request is being made to `http://{{baseUrl}}/api-clients`. The `Tests` tab is active, containing the following JavaScript code:

```
1 pm.test("verify token created and save for later", function () {
2     var jsonData = pm.response.json();
3     pm.expect(jsonData.accessToken).not.to.be.empty;
4     pm.collectionVariables.set("token", jsonData.accessToken);
5 });
6
```

# Getting Data by Filter

Extracting data  
from the response

# Getting Data Dynamically

```
const response = pm.response.json();

const availableBook = response.filter((item) => item.available === true);

console.log(availableBook[0]);
```

# Getting Data Dynamically

```
pm.environment.set("bookId", availableBook[0].id);
console.log("Id of the ordered book is: " + pm.environment.get("bookId"));
```

GET ▼ `{baseUrl}/books?type=fiction`

Params ● Authorization Headers (7) Body Pre-request Script **Tests ●** Settings

```
2 const response = pm.response.json();
3 const availableBook = response.filter((item) => item.available === true);
4
5 if (availableBook[0]){
6   const book = availableBook[0];
7   pm.test("is book found", () => {
8     |   pm.expect(book.available).to.eql(true);
9
10 })
11
12 if (book.available){
13   |   pm.environment.set("bookId", availableBook[0].id);
```



סיכום ושאלות