



TECH PRO ED

PROFESSIONAL TECHNOLOGY EDUCATION

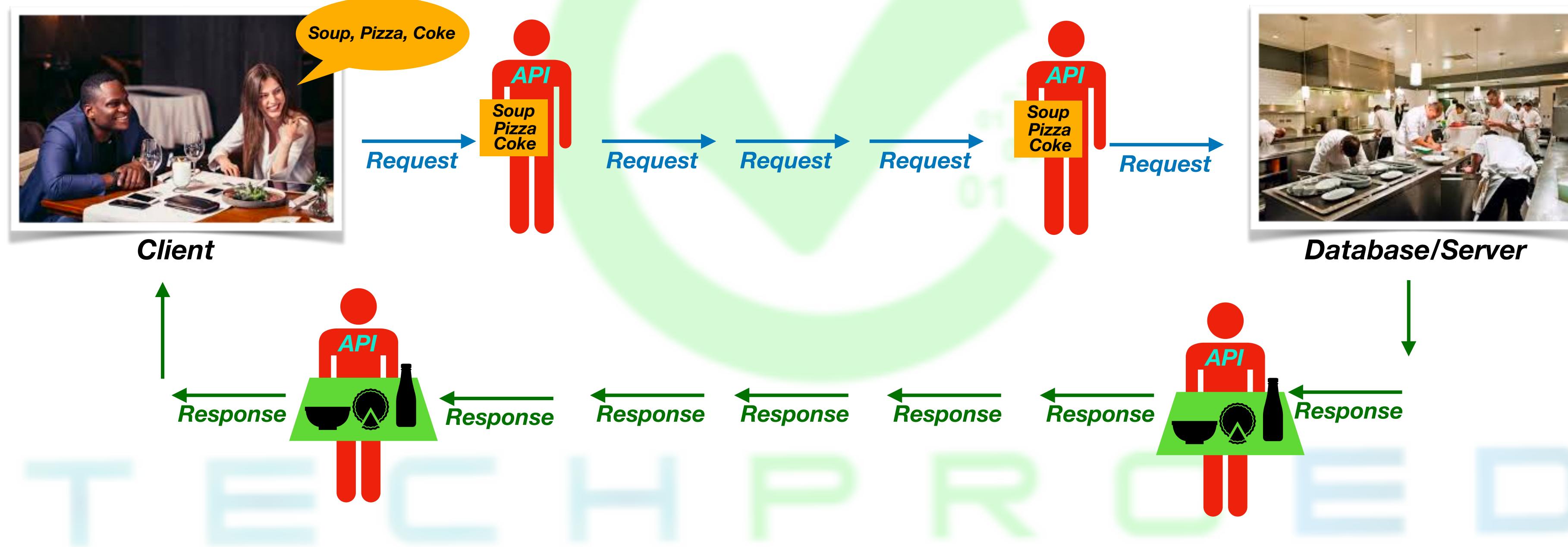
API TESTING COURSE SPRING 2020

What is API ?

API stands for **Application Programming Interface**.

API is an application **without User Interface(UI)**.

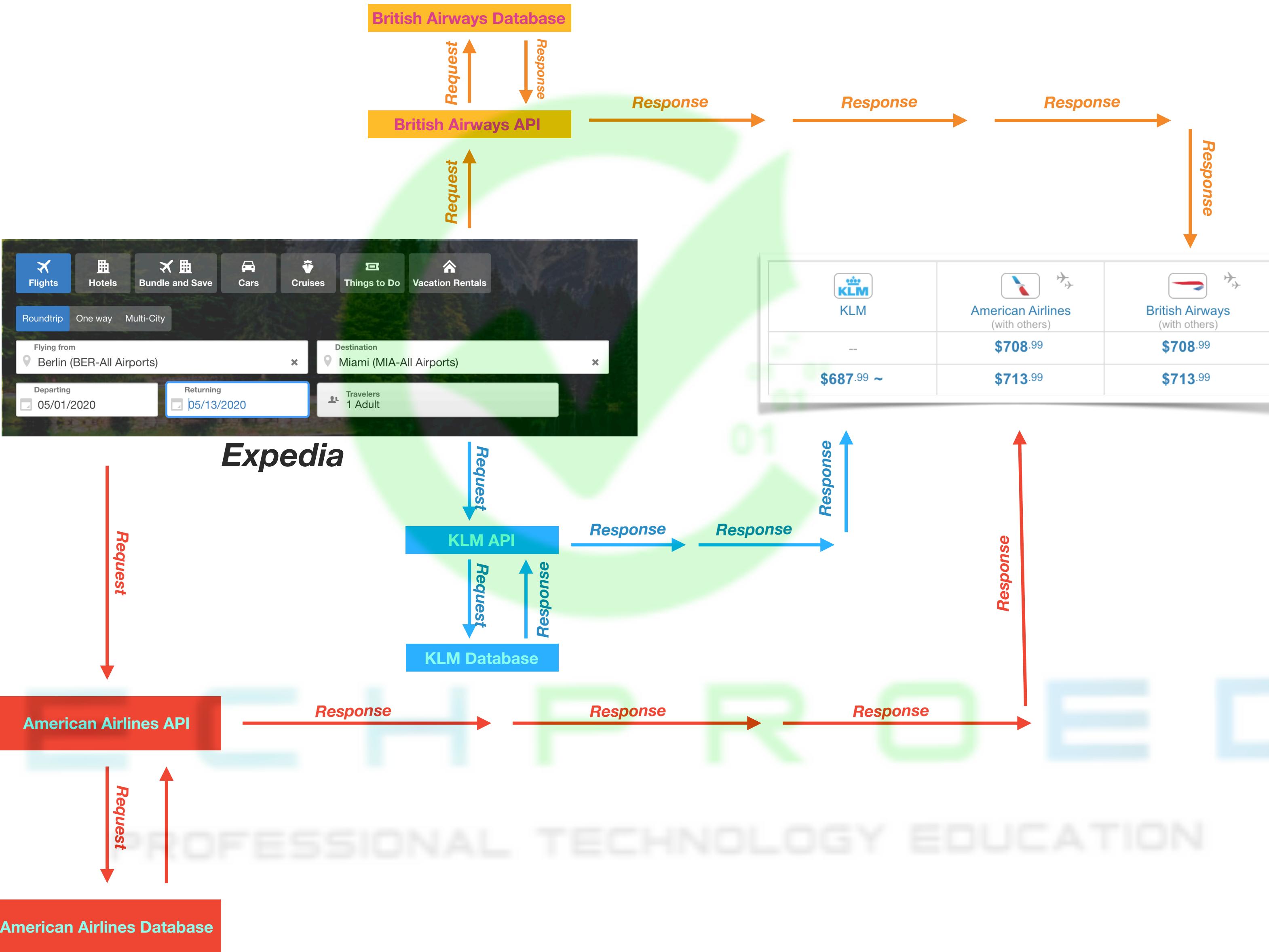
APIs specify **how** one software program should **interact with other** software programs.



Assume an **API** as a **Waiter** at a **Restaurant**.

As a waiter, the **API takes a request** from a client, takes that request to the database, fetches the requested data from the database and **returns a response** to the client.

How API Works ?



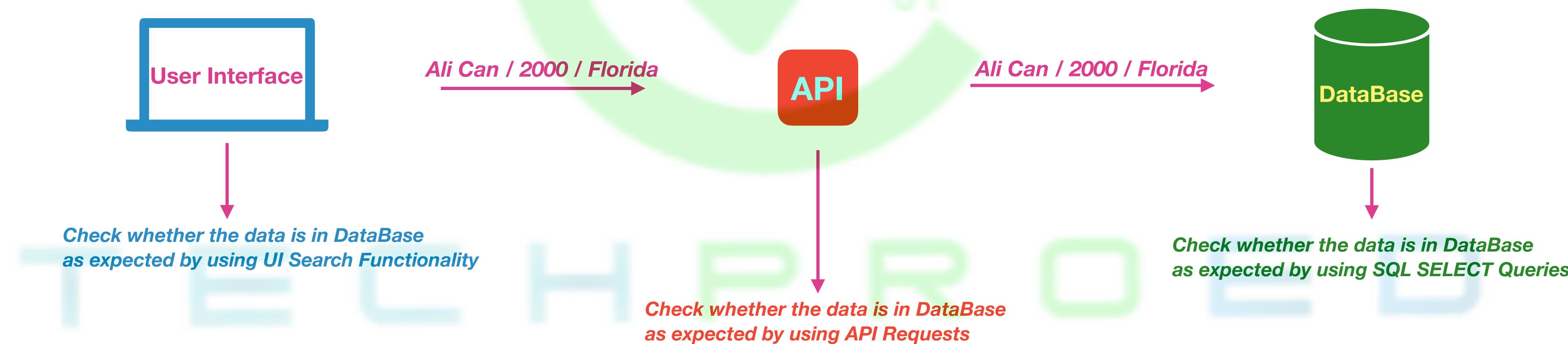
What is API Testing ?

API testing is done to check whether the **API** meets expectations in terms of **functionality, reliability, performance, and security** of an application.

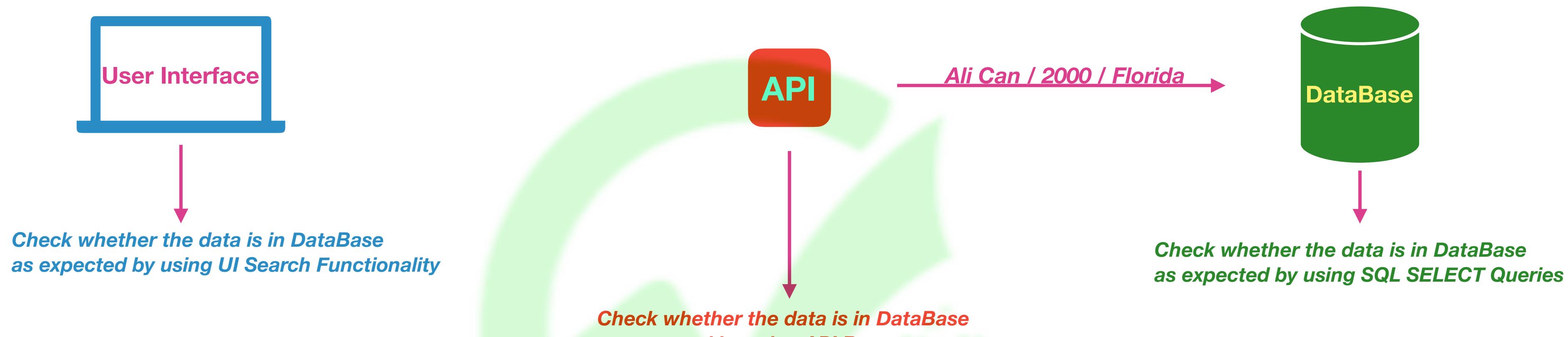
API testing won't concentrate on look of the application.

End to End (E2E) Scenarios:

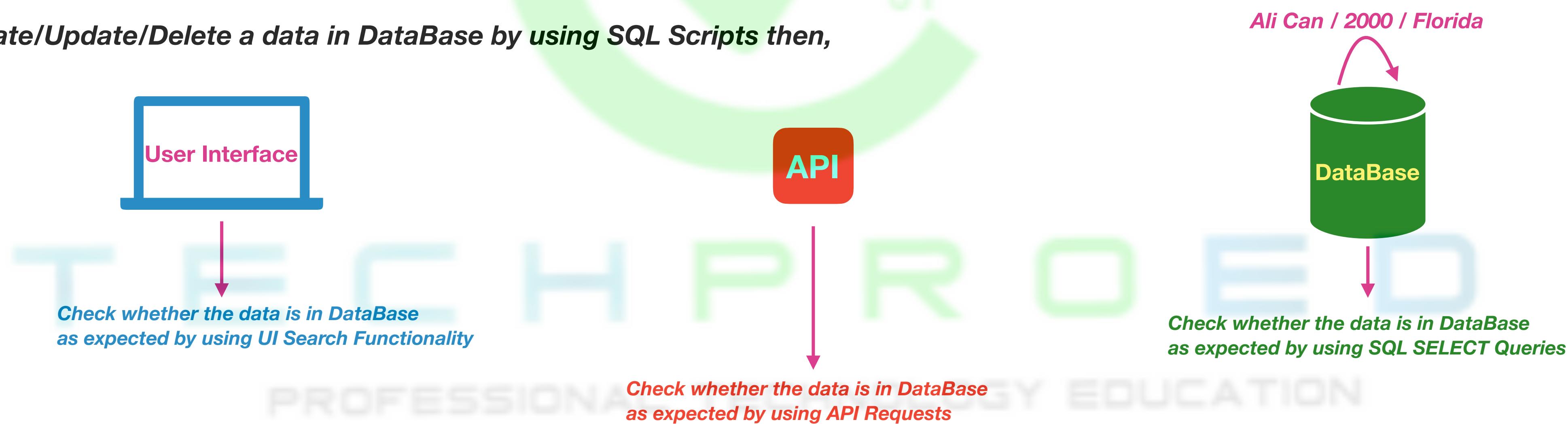
1) Create/Update/Delete a data in DataBase by using User Interface then,



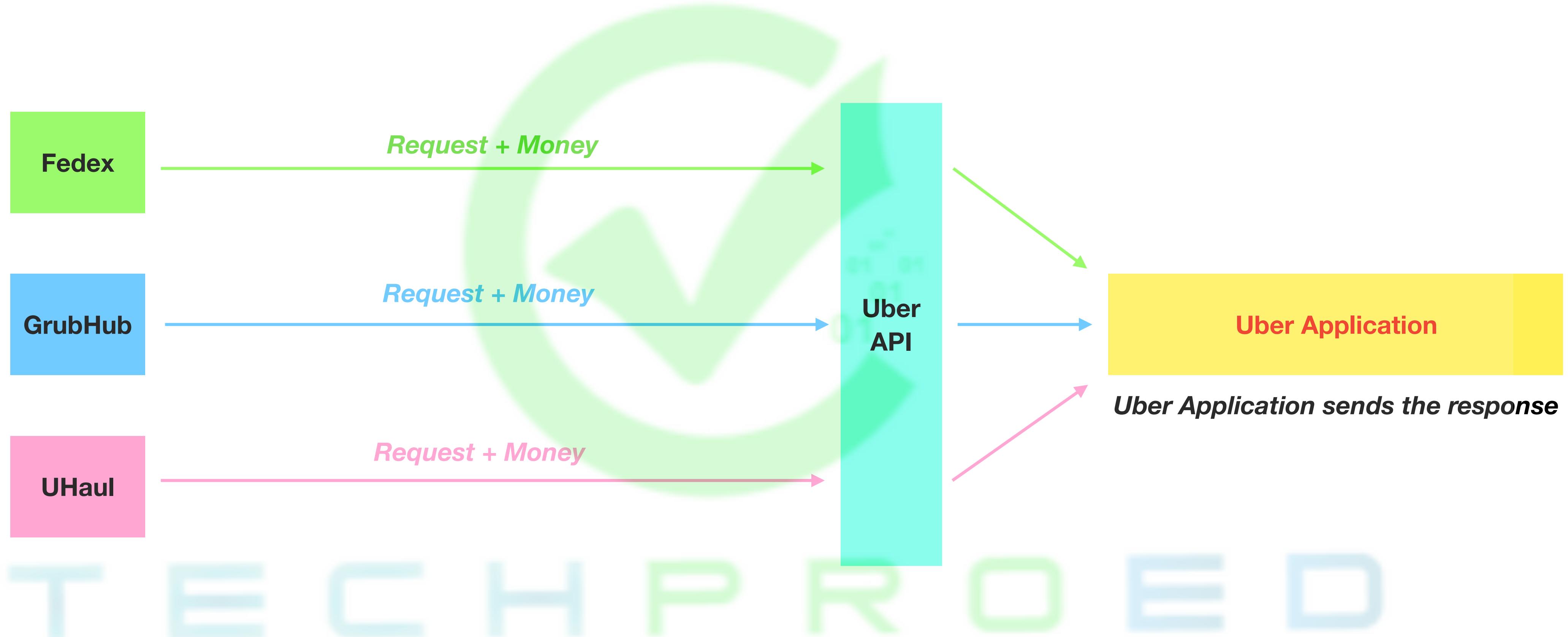
2) Create/Update/Delete a data in DataBase by using API then,



3) Create/Update/Delete a data in DataBase by using SQL Scripts then,



Note: Every company creates their own APIs and earns money by selling the services



TECH PRO E D

PROFESSIONAL TECHNOLOGY EDUCATION

What is the difference between API and Web Services ?

API and Web Service create communication between applications.

Web Service

The only difference is that a Web Service creates interaction between two applications through internet.

For example, Expedia uses KLM Airlines DataBase through internet.

So Expedia is using **KLM Web Service**.

API

Desktop applications such as Microsoft Word use their own APIs which don't need internet.

So when we use Microsoft Word, we use **Microsoft Word API**

Note: All Web Services are APIs but all APIs are not Web Services.

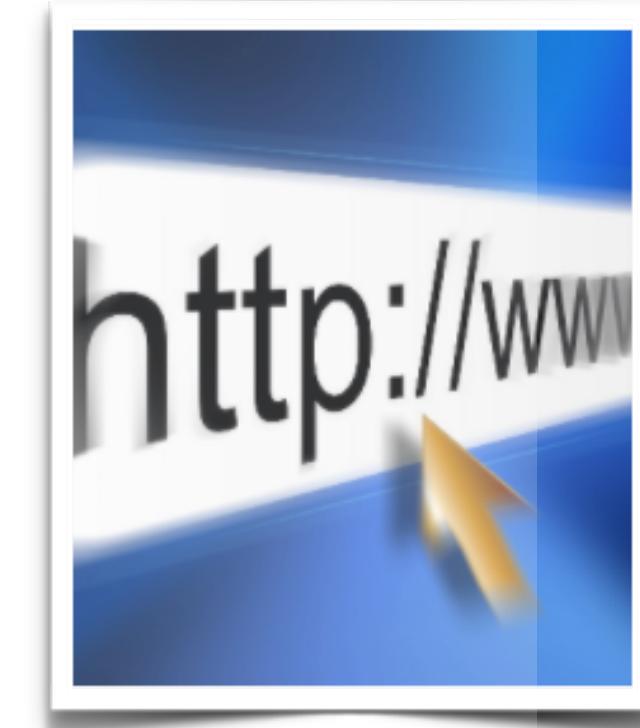
What is HTTP?

HTTP stands for **Hyper Text Transfer Protocol**

Communication between **client** computers and **web servers** is done by sending **HTTP Requests** and receiving **HTTP Responses**

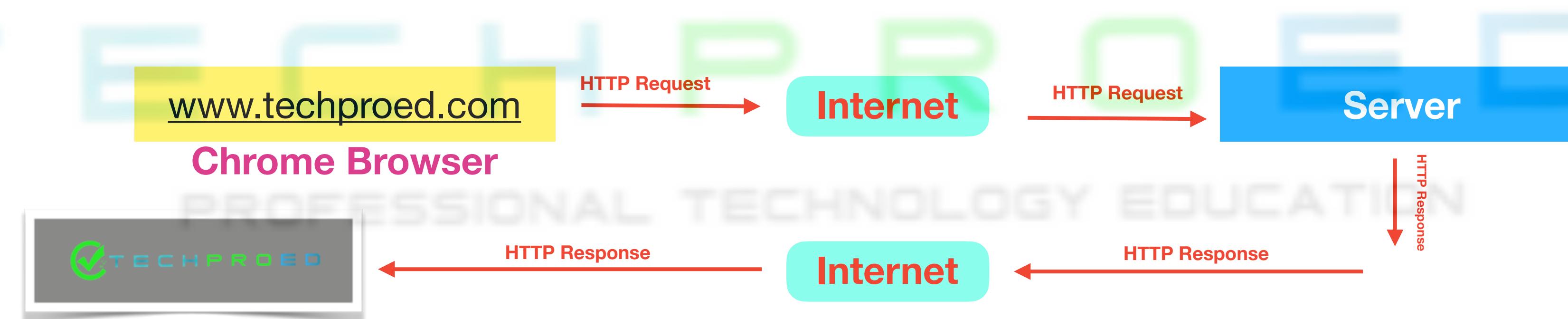
Clients are often **browsers** (Chrome, Edge, Safari), but they can be any type of program or device.

Servers are most often computers in the cloud.



Communication between **clients** and **servers** is done by requests and responses:

1. A client (a browser) sends an **HTTP Request** to the web
2. A **Web Server** receives the request
3. The server runs an application to process the request
4. The server returns an **HTTP Response** (output) to the browser
5. The client (the browser) receives the response

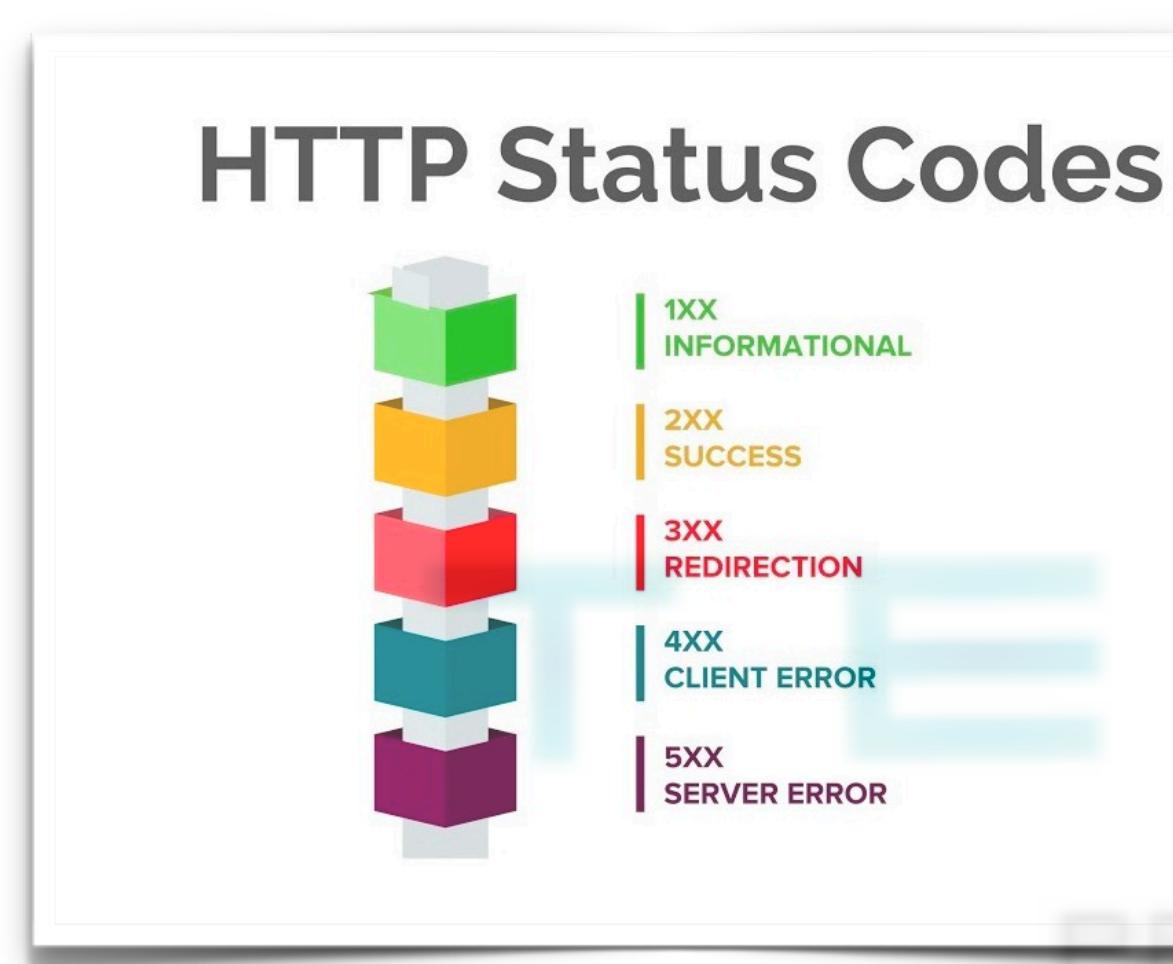


What exactly needs to be verified in API Testing?

On **API Testing**, we send a request to the API with the known data and we analyze the response.

1) HTTP Status Codes

2) Data Accuracy



The screenshot shows the Postman API testing interface. At the top, there are tabs for Params, Authorization, Headers, Body, Pre-request Script, Tests, Cookies, Code, and Comments (0). Below these are sections for Query Params, Body, Cookies, Headers (22), and Test Results. A green circular progress bar is overlaid on the interface. In the bottom right corner of the main window, there is a red box highlighting the status message "Status: 200 OK" and the time "Time: 227 ms". A red number "1)" is placed above this box. In the bottom left corner of the JSON response pane, there is another red box highlighting the JSON data, and a red number "2)" is placed above it. The JSON response is as follows:

```
1 {  
2   "type": "FeatureCollection",  
3   "metadata": {  
4     "generated": 1555900401000,  
5     "url": "https://earthquake.usgs.gov/earthquakes/feed/v1.0/summary/4.5_hour.geojson",  
6     "title": "USGS Magnitude 4.5+ Earthquakes, Past Hour",  
7     "status": 200,  
8     "api": "1.8.1",  
9     "count": 1  
10    },  
11    "features": [  
12      {  
13        "type": "Feature",  
14        "properties": {  
15          "mag": 4.6,  
16          "place": "92km WNW of Neiafu, Tonga",  
17          "time": 1555898797799,  
18          "updated": 1555900126040,  
19          "tz": -720,  
20          "url": "https://earthquake.usgs.gov/earthquakes/eventpage/us70003a3x",  
21          "id": "us70003a3x"  
22        }  
23      }  
24    ]  
25  }  
26 }
```

Common HTTP Status Codes

- 1) 200 (OK) ==> This is the standard response for successful HTTP requests.
- 2) 201 (CREATED) ==> This is the standard response for an HTTP request that resulted in an item being successfully created.
- 3) 204 (NO CONTENT) ==> This is the standard response for successful HTTP requests, where nothing is being returned in the response body.
- 4) 400 (BAD REQUEST) ==> The request cannot be processed because of bad request syntax, excessive size, or another client error.
- 5) 403 (FORBIDDEN) ==> The client does not have permission to access this resource.
- 6) 404 (NOT FOUND) ==> The resource could not be found at this time. It is possible it was deleted, or does not exist yet.
- 7) 500 (INTERNAL SERVER ERROR) ==> The generic answer for an unexpected failure if there is no more specific information available.

TECHPROED

PROFESSIONAL TECHNOLOGY EDUCATION

Tools Used for API Testing?

1) Postman

Postman is a popular **API Tool** that makes it easy for developers to **create, share, “test”** and **document APIs**.



2) SOAP UI

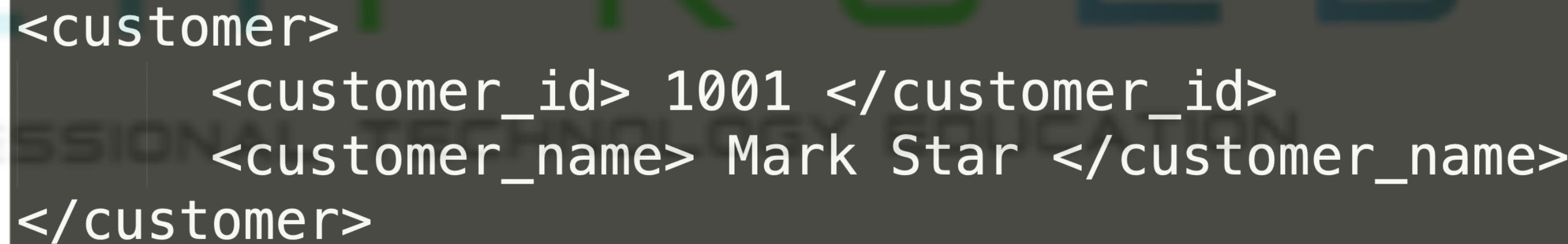
SOAP stands for **Simple Object Access Protocol**.

It is important for web applications to be able to communicate over the Internet.

The best way to communicate between applications is over **HTTP**, because **HTTP** is supported by all internet browsers and servers. **SOAP** was created to accomplish this.

SOAP provides a **way or format** to communicate between applications running on different operating systems, with different technologies and programming languages. **SOAP** is **platform independent**

SOAP is based on **XML Format**.



A dark grey rectangular box containing white XML code. The code defines a single customer object with attributes customer_id and customer_name.

```
<customer>
    <customer_id> 1001 </customer_id>
    <customer_name> Mark Star </customer_name>
</customer>
```

3) REST / Restful

REST stands for REpresentational State Transfer

REST is making easier for systems to communicate with each other.

In the **REST** architecture, clients **send requests to retrieve or modify resources**, and servers **send responses** to these requests.

REST is able to use **XML** and **JSON (Java Script Object Notation)** Format; therefore, it is more common than **SOAP**

```
<customer>
    <customer_id> 1001 </customer_id>
    <customer_name> Mark Star </customer_name>
</customer>
```

XML Format

```
{
    "customer_id": 1001,
    "customer_name": "Mark Star"
}
```

JSON Format (key- value like maps)

What does a REST request generally consist of ?

1) An HTTP method defines what kind of operation to perform

- A) GET is for *reading* data ==>To search hotels from www.hotels.com
- B) POST is for *creating* a new data ==>To make hotel reservation from www.hotels.com
- C) PUT is for *updating* data completely ==>To change all reservation details from www.hotels.com
- D) PATCH is for *updating* data partially ==>To change just reservation date from www.hotels.com
- E) DELETE is for *removing* data ==>To cancel reservation from www.hotels.com

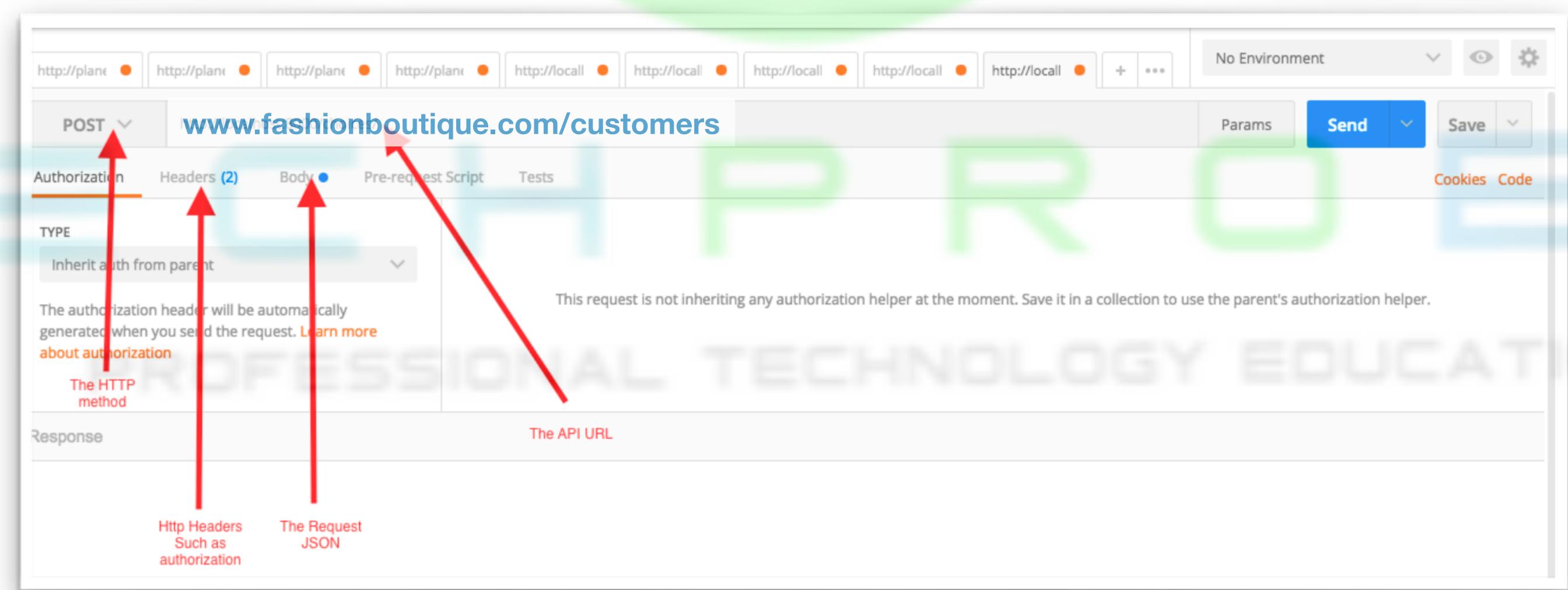
2) In HEADER, the client sends the type of content which will be received from the server.

Header ensures that the server does not send data that cannot be understood or processed by the client.

3) A PATH (Endpoint/URL/URI) to a resource:

Requests must contain a path to a resource that the operation should be performed on.

Endpoint is like www.fashionboutique.com/customers



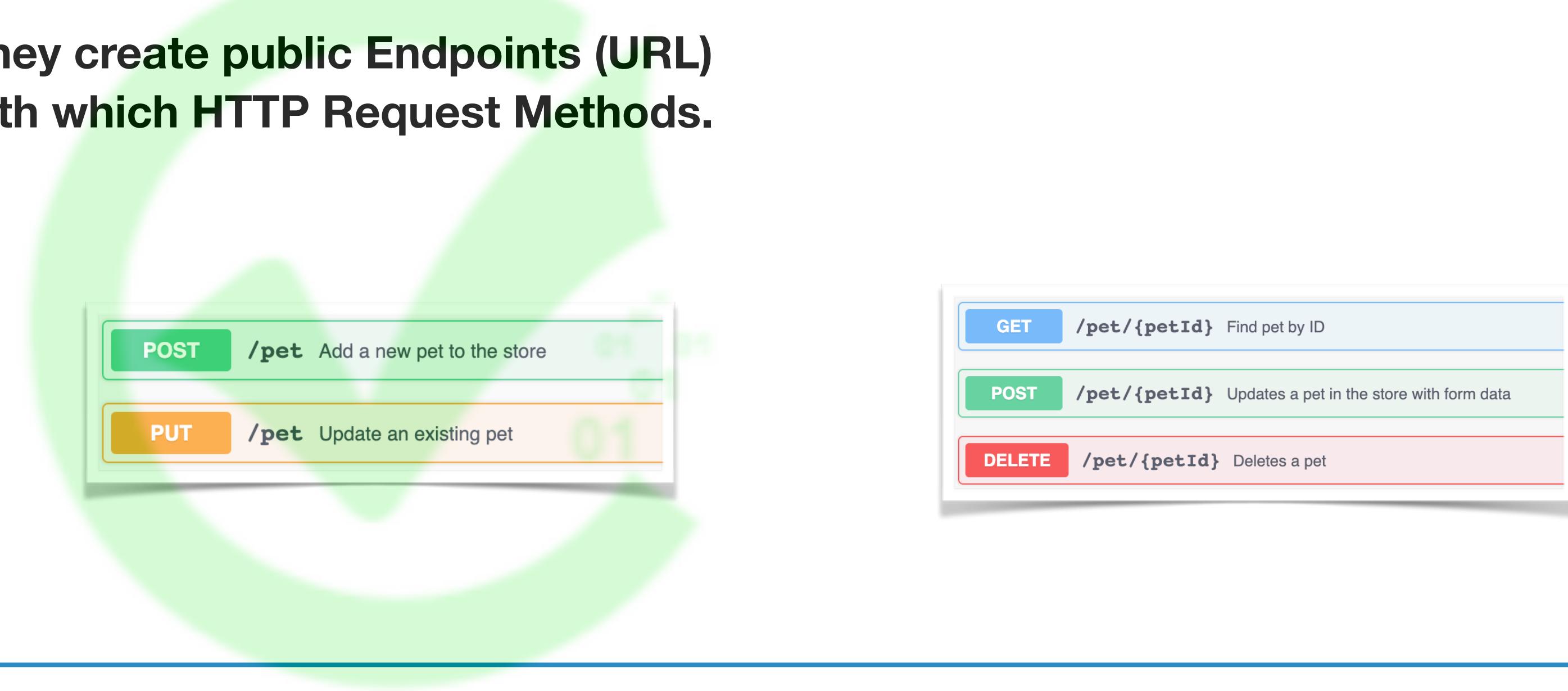
What is an API Endpoint?

The place that APIs send requests and where the resource lives, is called an endpoint.

When API developers create an API, they create public Endpoints (URL) and decide which Endpoint will run with which HTTP Request Methods.

<http://petstore.swagger.io/>

Base URL / Method Name



Testers test every endpoints if they work as expected by using **GET, POST, PUT, DELETE** HTTP Request methods.

TECHPROED

PROFESSIONAL TECHNOLOGY EDUCATION

What is Swagger?

Swagger is a framework for describing your API by using a common language that is easy to read and understand by developers and testers, even they have weak source code knowledge.

You can think of it as a **blueprint** for a house.

You can use whatever building materials you like, but you can't step outside the parameters of the blueprint.

I used **SWAGGER** for API documentation.
SWAGGER gives me API Endpoints and informs me about how to use them.

TECH PRO E

PROFESSIONAL TECHNOLOGY EDUCATION

Postman Panel

You can create collection
to store all requests for organizing

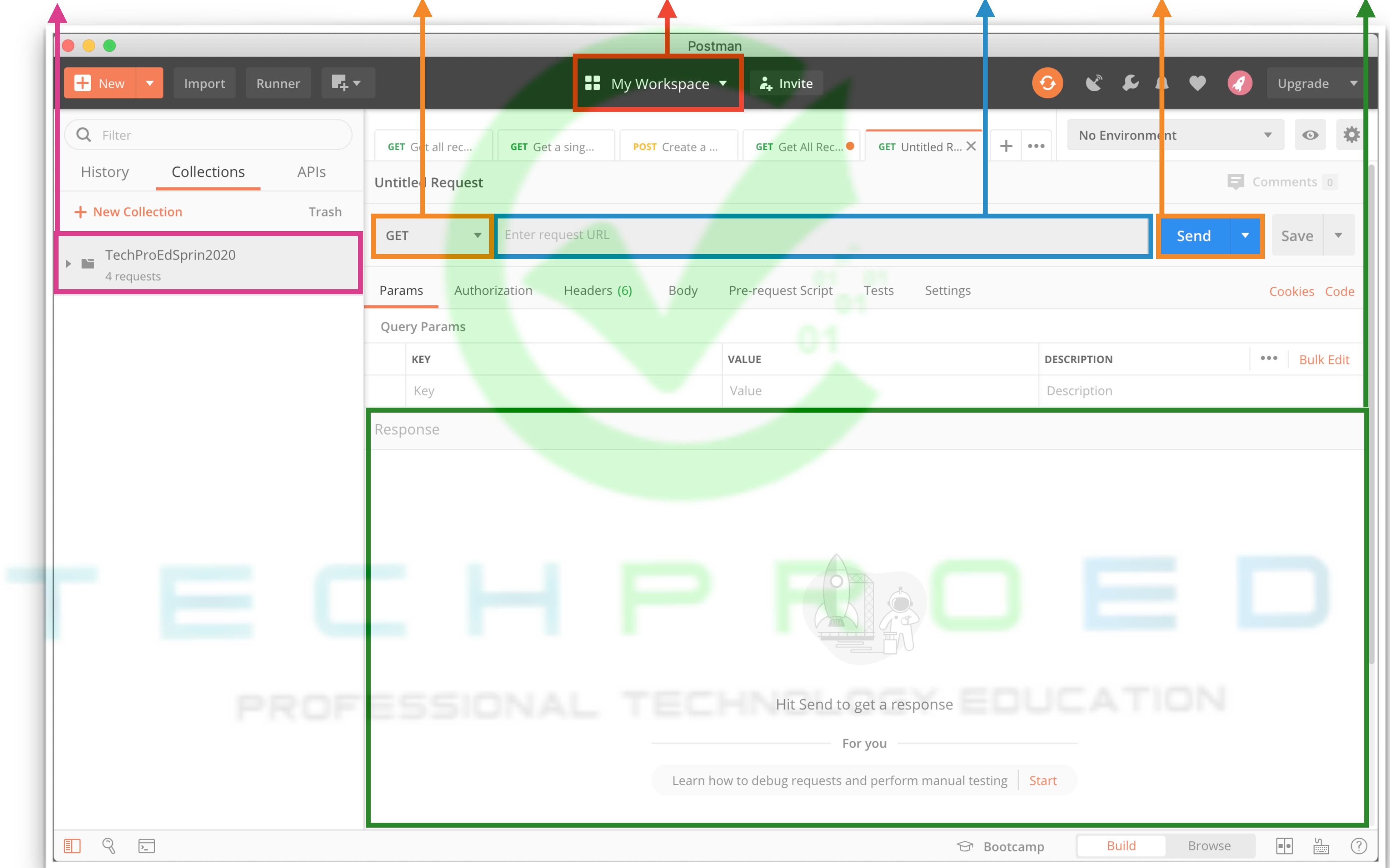
Request Method

Name of the workspace

Request URL bar

Send button

Response window



How to work with GET Request ?

Postman is used for backend testing where we enter the end-point URL, it sends the request to the server and receives the response back from the server.

Base URL	Base URL	Method	Endpoints	Description
https://restful-booker.herokuapp.com	JSON	GET	/booking	Get all employee data
		GET	/booking/1	Get a single employee data
		POST	/booking? firstname=Suleyman&lastname=Alptekin&totalprice=123&depositpaid=true&additionalneeds=Wifi	Create new record in database
		PATCH	/booking/12	Update an employee record partially
		PUT	/booking/11	Update an employee record
		DELETE	/booking/11	Delete an employee record

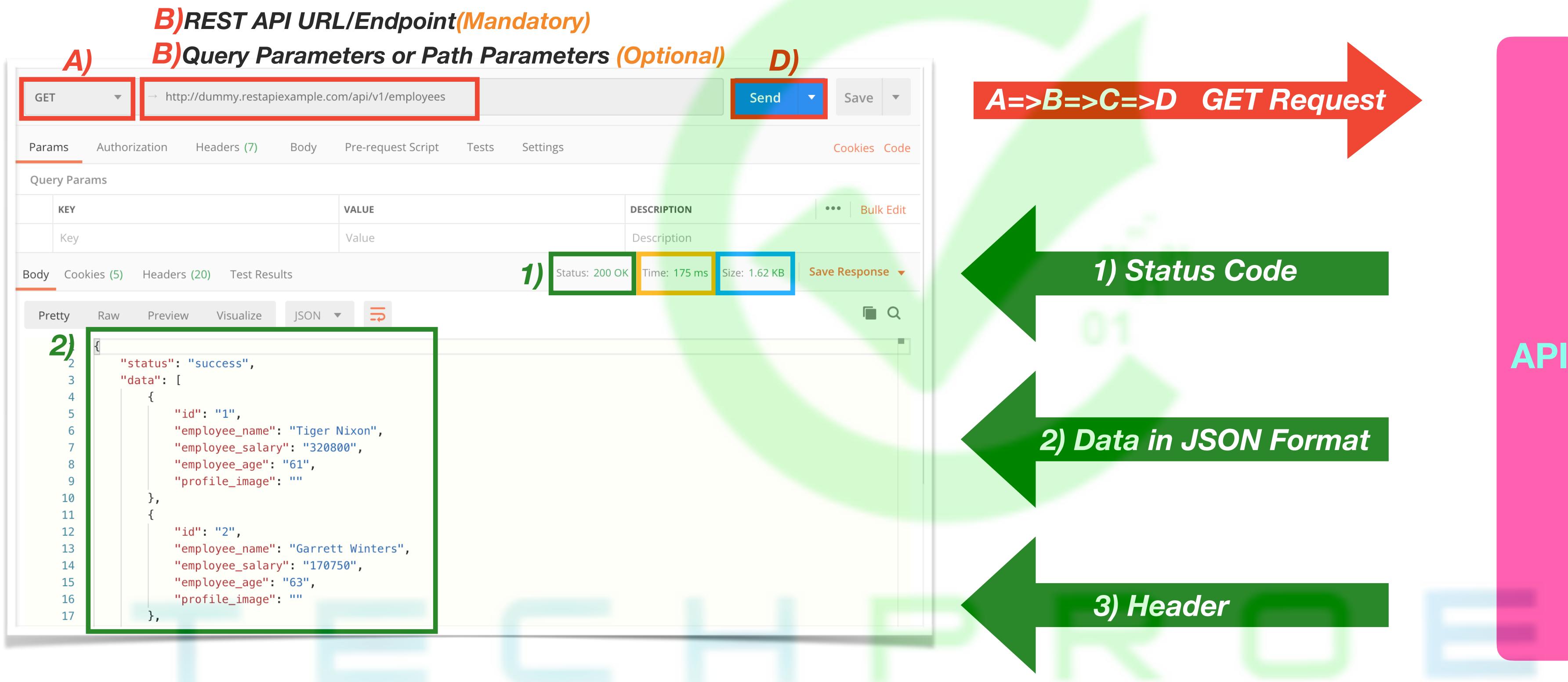
TECHPROED

PROFESSIONAL TECHNOLOGY EDUCATION

How to work with GET Request ?

C) In Headers use “Accept” to select response format.

The format can be XML or JSON (Optional)



Note: After getting response we check **status code** first

Note: **Accept** indicates **what kind of response from the server** the client can accept.

But API should support the expected format, otherwise API sends the response in default format

Note: You will need Authorization to GET Request

How to work with POST Request ?

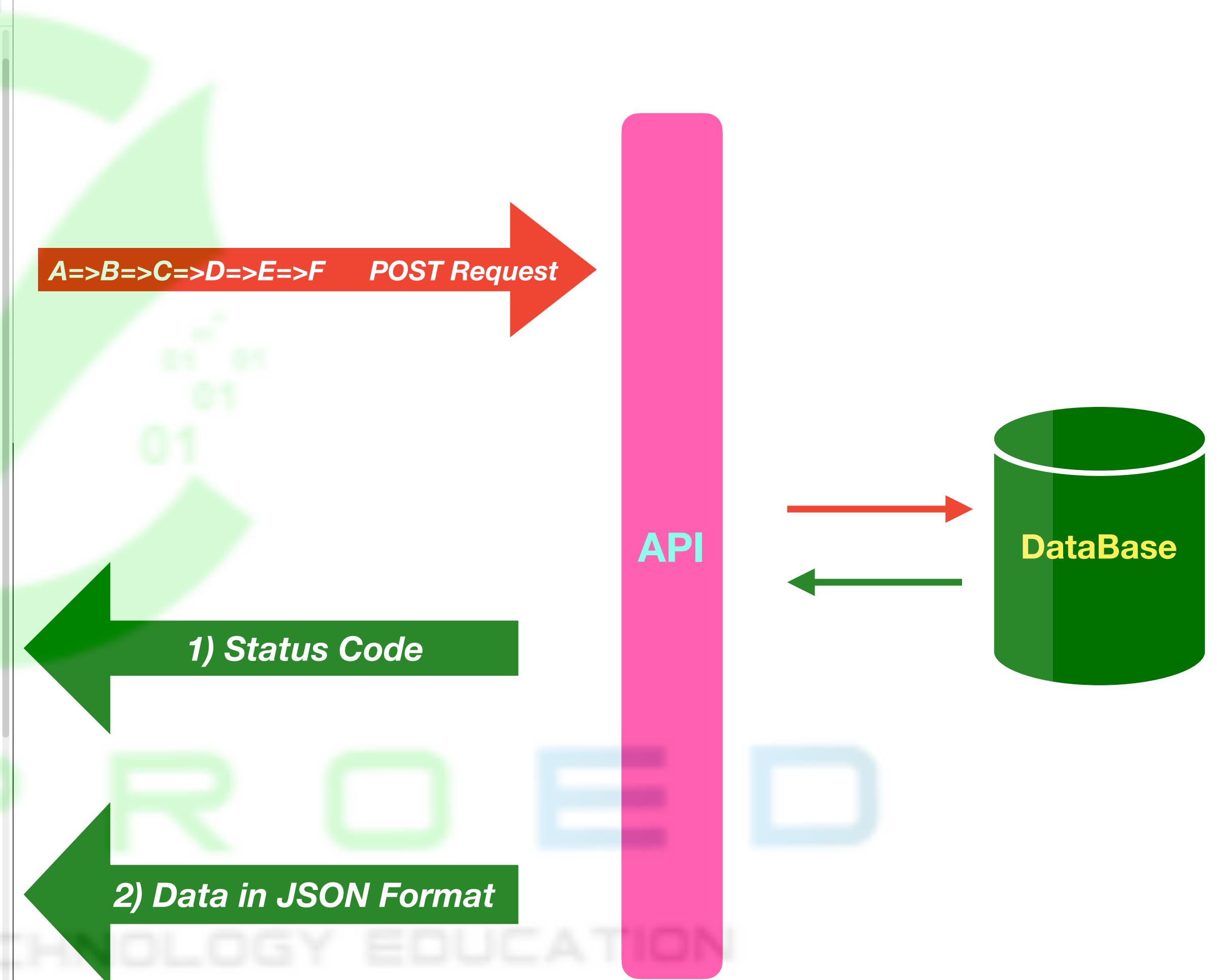
The screenshot shows the Postman interface with the following details:

- A)** Method dropdown set to POST.
- B)** URL input field: https://jsonplaceholder.typicode.com/users.
- C)** Body tab selected.
- D)** Content type dropdown set to JSON.
- E)** Request body (Pretty) JSON:

```
1 {  
2   "id": 11,  
3   "name": "Suleyman Alptekin",  
4   "username": "SALp",  
5   "email": "selam73@yahoo.com",  
6   "address": {  
7     "street": "Walk street",  
8     "suite": "Apt. 123",  
9     "city": "Hialeah",  
10    "zipcode": "33018-3874",  
11    "geo": {  
12      "lat": "-37.3159",  
13      "lng": "81.1496"  
14    },  
15  },  
16  "phone": "1-770-123-8031 x56442",  
17  "website": "slptkn73.org",  
18  "company": {  
19    "name": "TechProEd LLC",  
20    "catchPhrase": "Full Stack Automation Tester Course",  
21    "bs": "Real-time e-school"  
22  }  
23 }
```

- F)** Send button.
- 1)** Status: 201 Created.
- 2)** Response body (Pretty) JSON:

```
1 {  
2   "id": 11,  
3   "name": "Suleyman Alptekin",  
4   "username": "SALp",  
5   "email": "selam73@yahoo.com",  
6   "address": {  
7     "street": "Walk street",  
8     "suite": "Apt. 123",  
9     "city": "Hialeah",  
10    "zipcode": "33018-3874",  
11    "geo": {  
12      "lat": "-37.3159",  
13      "lng": "81.1496"  
14    },  
15  },  
16  "phone": "1-770-123-8031 x56442",  
17  "website": "slptkn73.org",  
18  "company": {  
19    "name": "TechProEd LLC",  
20    "catchPhrase": "Full Stack Automation Tester Course",  
21    "bs": "Real-time e-school"  
22  }  
23 }
```

Note: After getting response we check **status code** first

Note: You will need Authorization to POST Request

How to work with PUT Request ?

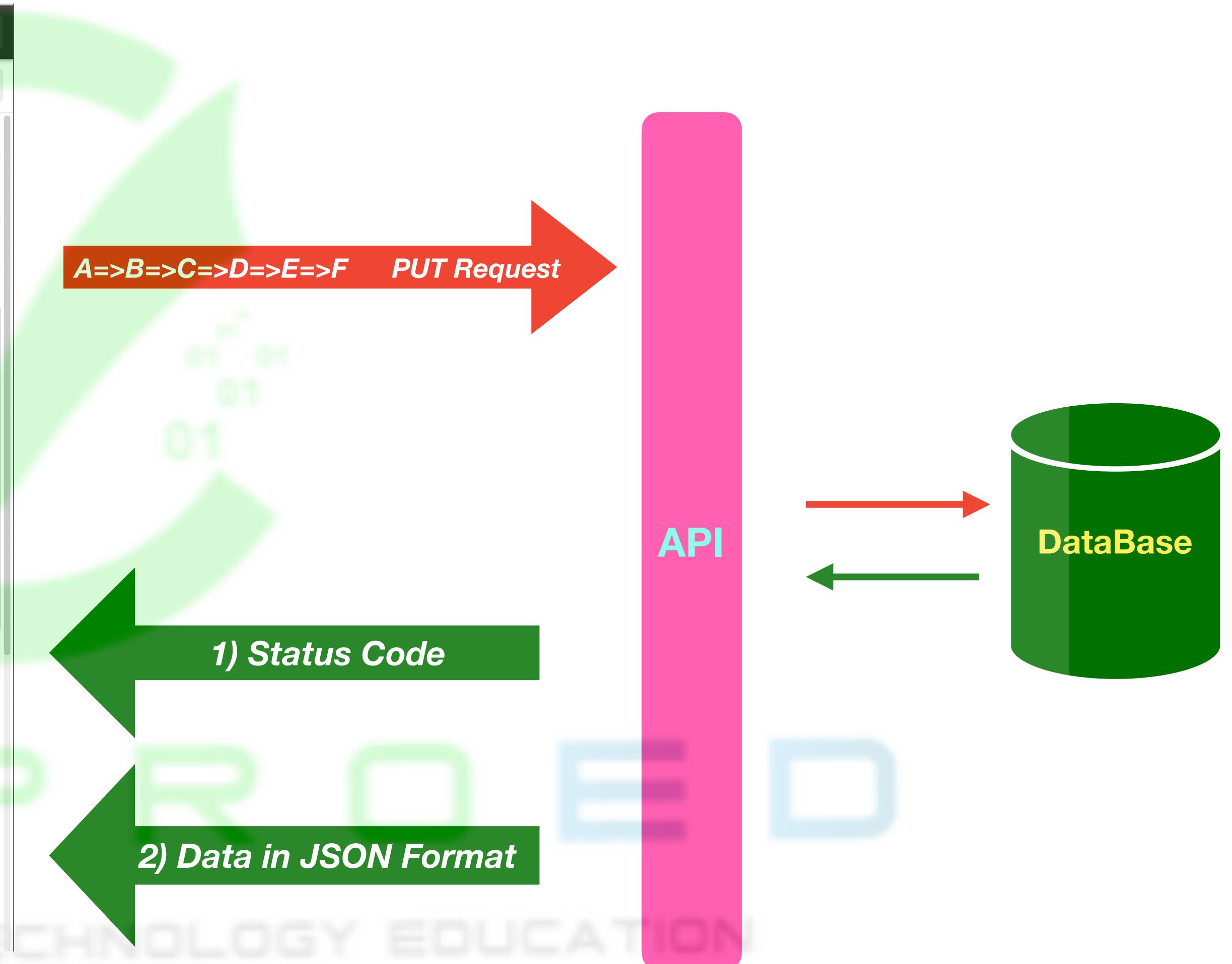
The screenshot shows the Postman interface with the following details:

- A)** Method: PUT
- B)** URL: https://restful-booker.herokuapp.com/booking/11
- C)** Headers: Content-Type: application/json
- D)** Body: Raw JSON
- E)** Request Body (Raw JSON):

```
1 {  
2   "firstname": "Suleyman",  
3   "lastname": "Alptekin",  
4   "totalprice": 321,  
5   "depositpaid": true,  
6   "bookingdates": {  
7     "checkin": "2020-05-02",  
8     "checkout": "2020-05-05"  
9   },  
10  "additionalneeds": "Breakfast"  
11 }  
12 
```

- F)** Response Status: 200 OK
- 1)** Response Body (Raw JSON):

```
1 {  
2   "firstname": "Suleyman",  
3   "lastname": "Alptekin",  
4   "totalprice": 321,  
5   "depositpaid": true,  
6   "bookingdates": {  
7     "checkin": "2020-05-02",  
8     "checkout": "2020-05-05"  
9   },  
10  "additionalneeds": "Breakfast"  
11 }  
12 
```

Note: After getting response we check **status code** first

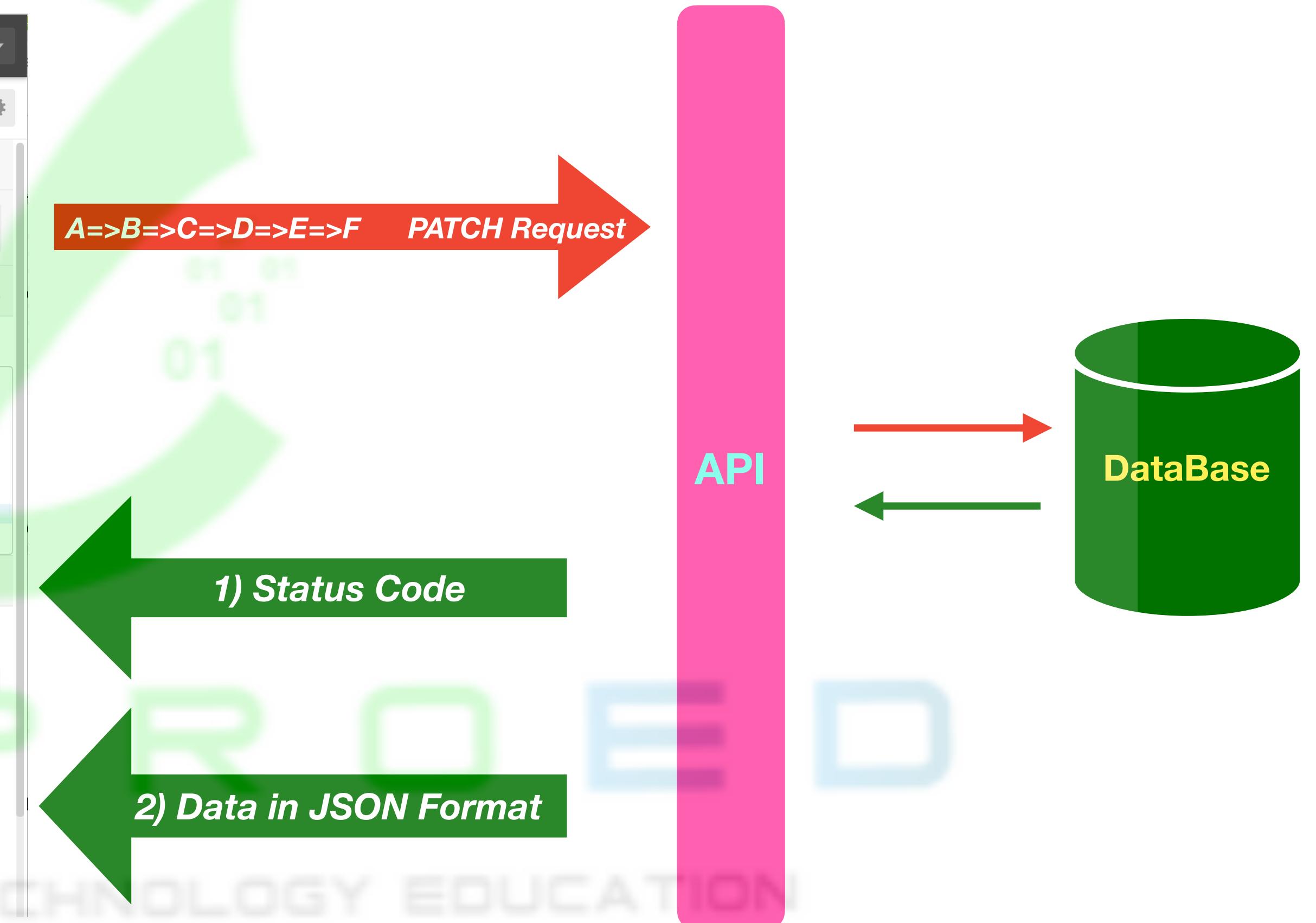
Note: You will need Authorization to PUT Request

How to work with PATCH Request ?

The screenshot shows the Postman interface with a PATCH request to `https://restful-booker.herokuapp.com/booking/13`. The request path is highlighted in red. The JSON body is also highlighted in red. The response status code 200 OK is highlighted in green.

Annotations:

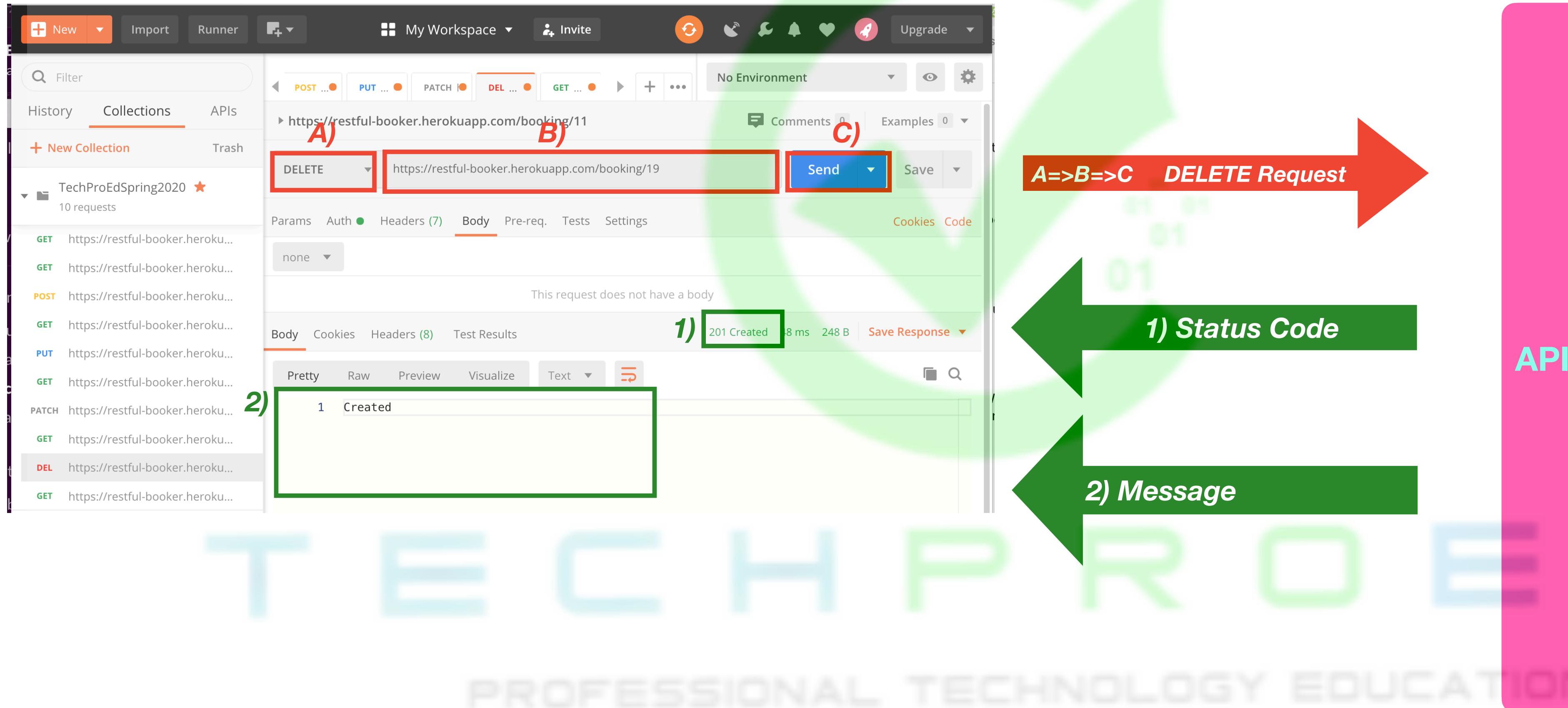
- A) History
- B) Collections
- C) APIs
- D) New Collection
- E) Trash
- F) TechProEdSpring2020
- G) Filter
- H) My Workspace
- I) Invite
- J) Upgrade
- K) Launch...
- L) POST https://...
- M) PUT https://...
- N) PATCH https://...
- O) +
- P) ...
- Q) No Environment
- R) Comments
- S) Examples 0
- T) Send
- U) Save
- V) Params
- W) Auth
- X) Headers (9)
- Y) Body
- Z) Pre-req.
- A1) Tests
- A2) Settings
- B1) Cookies
- B2) Code
- C1) Beautify
- D1) raw
- D2) JSON
- E1) {
E2) "firstname": "Turkan",
E3) "bookingdates": {
E4) "checkin": "2020-05-03"
E5) },
E6) "additionalneeds": "Dinner"
E7) }
E8)
- F1) 1) 200 OK
F2) 515 ms
F3) 417 B
F4) Save Response
- G1) Body
G2) Cookies
G3) Headers (8)
G4) Test Results
- H1) Pretty
H2) Raw
H3) Preview
H4) Visualize
H5) JSON
- I1) 1) {
I2) "firstname": "Turkan",
I3) "lastname": "Bennett",
I4) "totalprice": 4719,
I5) "depositpaid": false,
I6) "bookingdates": {
I7) "checkin": "2020-05-03",
I8) "checkout": "2020-05-04"
I9) },
I10) "additionalneeds": "Dinner"
I11) }



Note: After getting response we check **status code** first

Note: You will need Authorization to PATCH Request

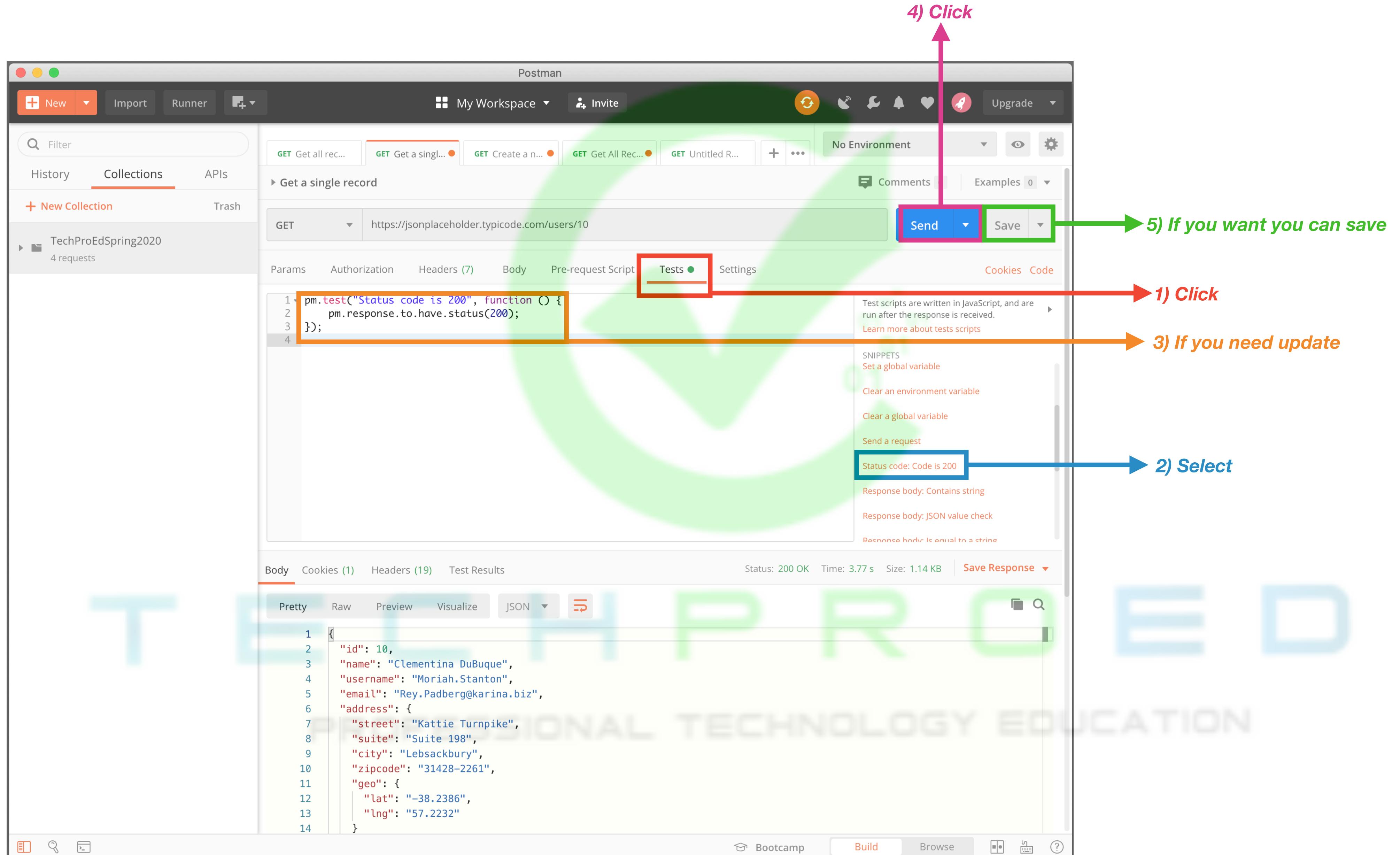
How to work with *DELETE* Request ?



Note: After getting response we check **status code** first

Note: You will need Authorization to **DELETE Request**

How to Create Tests on Postman ?



How to Create API Test Cases ?

Test the following test cases by using Postman (Manually)

- 1) When I send a **GET** request to REST API URL <https://restful-booker.herokuapp.com/booking>
And **Accept** type is “application/JSON”
Then **HTTP Status Code** should be 200
And **Response format type** should be "application/JSON"

- 2) When I send a **GET** request to REST API URL <https://restful-booker.herokuapp.com/booking/5>
And **Accept** type is “application/JSON”
Then **HTTP Status Code** should be 200
And **first name** should be “Mary”
And **total price** should be 814

- 3) When I send a **GET** request to REST API URL <http://dummy.restapiexample.com/api/v1/employees>
And **Accept** type is “application/JSON”
Then **HTTP Status Code** should be 200
And “Gerret Winters” should be displayed among data

- 4) When I send a **GET** request to REST API URL <http://dummy.restapiexample.com/api/v1/employee/1>
And **Accept** type is “application/JSON”
Then **HTTP Status Code** should be 200
And **employee name** should be “Tiger Nixon”
And **employee salary** should be “320800”
And **employee age** should be “61”

Query and Path Parameters ?

Note 1: <https://www.google.com/search?q=selenium>

The meaning of "?" is URL finished.

After the "?", we type **keys** and **values**, they are called **query(request) parameters**.

In REST, there are 2 types of parameters

1)Query Parameters: BaseURL/users?occupation=programmer ==> Fetch a list of programmer user

It is not the part of the URL

It uses key-value format

Note 2: Those parameters must be defined by developers

2)Path Parameters: BaseURL/users/123 ==> Fetch a user whose id is 123

It is a part of the URL

BaseURL/users/:id ==> In here ":id" is path parameter.

When you enter id you get all data about the user who has the id

Note 3: If you want to **sort** or **filter** items, then you should use **Query Parameter**.

Note 4: If you want to **identify a resource**, you should use **Path Parameter**.

What is JsonPath ?

Note: We use JsonPath to navigate and manipulate data in Json Format

- 1) \$ will return all the nodes inside the JSON document.
- 2) In order to get children of a given node, we can use the dot (.) operator with \$ use \$.Countries
- 3) If you want to display the Data nodes of all the countries use \$.Countries[*].Data
- 4) You can use the Array Index [i, j, k...] to identify an entry at a particular index like \$.Countries[0]
To get the last element use -1 as index like \$.Countries[-1]
- 5) You can extract multiple items from the array at different indexes like \$.Countries[0,3]
It will return the countries whose indexes are 0 and 3

To Practice JsonPath Queries: <https://jsonpath.herokuapp.com/>

PROFESSIONAL TECHNOLOGY EDUCATION

How to create JsonPath Object ?

We can create JsonPath object in different ways.

- 1) JsonPath json = response.jsonPath();**
- 2) JsonPath json = new JsonPath(responseAsString());**
- 3) JsonPath json = new JsonPath(filePath.json);**



Interview Questions

1) Explain what is a “Resource” in REST?

Answer: REST architecture treats every **content** as a **resource**.

These resources can be either text files, HTML pages, images, videos or dynamic business data.

REST Server provides access to resources and REST client accesses and modifies these resources.

Here each **resource** is identified by **URIs**.

2) What is the most popular way to represent a resource in REST?

Answer: REST uses different representations to define a resource like **TEXT**, **JSON**, and **XML**.

XML and **JSON** are the most popular representations of resources.

3) Which protocol is used by RESTful web services?

Answer: RESTful web services make use of **HTTP** protocol as a medium of communication between client and server.

4) State the core components of an HTTP Request?

Answer: Each HTTP request includes five key elements.

1. The Verb which indicates HTTP methods such as **GET**, **PUT**, **POST**, **DELETE**.
2. **URI** stands for Uniform Resource Identifier (**URI**). It is the identifier for the resource on the server.
3. **HTTP Version** which indicates HTTP version, for example-**HTTP v1.1**.
4. **Request Header** carries metadata (as key-value pairs) for the HTTP Request message. Metadata could be a client (or browser) type, the format that the client supports, message body format, and cache settings.
5. **Request Body** indicates the message content or resource representation.

5) State the core components of an HTTP response?

Answer: Every HTTP response includes four key elements.

1. Status/Response Code – Indicates Server status for the resource present in the HTTP request.

For example, 404 means resource not found, and 200 means response is ok.

2. HTTP Version – Indicates HTTP version, for example-HTTP v1.1.

3. Response Header – Contains metadata for the HTTP response message stored in the form of key-value pairs.

For example, content length, content type, response date, and server type.

4. Response Body – Indicates response message content or resource representation.

6) Name the most commonly used HTTP Request Methods supported by REST?

Answer: There are a few HTTP methods in REST which are more popular.

1. GET -It requests a resource at the request-URL. It should not contain a request body as it will get discarded.

Maybe it can be cached locally or on the server.

2. POST – It submits information to the service for processing; it should typically return the modified or new resource.

3. PUT – At the request URL it updates the resource.

4. DELETE – It removes the resource at the request-URL.

5. PATCH - At the request URL it updates the resource partially.

TECHPROED

PROFESSIONAL TECHNOLOGY EDUCATION

7) Is there any difference between PUT and POST operations? Explain it.

Answer: PUT and POST operation are almost the same. The only difference between the two is in terms of the result generated by them. A PUT operation is idempotent while the POST operation can give a different result.

Let's take an example.

1. PUT puts a file or resource at a particular URI and precisely at that URI.

If the resource already exists, then PUT updates it. If it's a first-time request, then PUT creates one.

2. POST sends data to a particular URI and expects the resource at that URI to deal with the request.

The web server at this point can decide what to do with the data in the context of the specified resource.

8) What do you understand by payload in RESTful web service?

Answer: Request body of every HTTP message includes request data called as Payload.

This part of the message is of interest to the recipient.

We can say that we send the payload in the POST method but not in <GET> and <DELETE> methods.

TECHPROED

PROFESSIONAL TECHNOLOGY EDUCATION