Automation Testing

Client-Server Basics

A Day in the Life of an Automation Test Engineer

Thomas now understands the value of REST APIs in the implementation of his restaurant website.

He would like to create a REST API that receives HTTP requests from his website frontend. The API should also be able to send HTTP responses back to the frontend.

To achieve this, John will make use of GET, POST, PUT, and DELETE HTTP methods to communicate with his REST API.

Learning Objectives

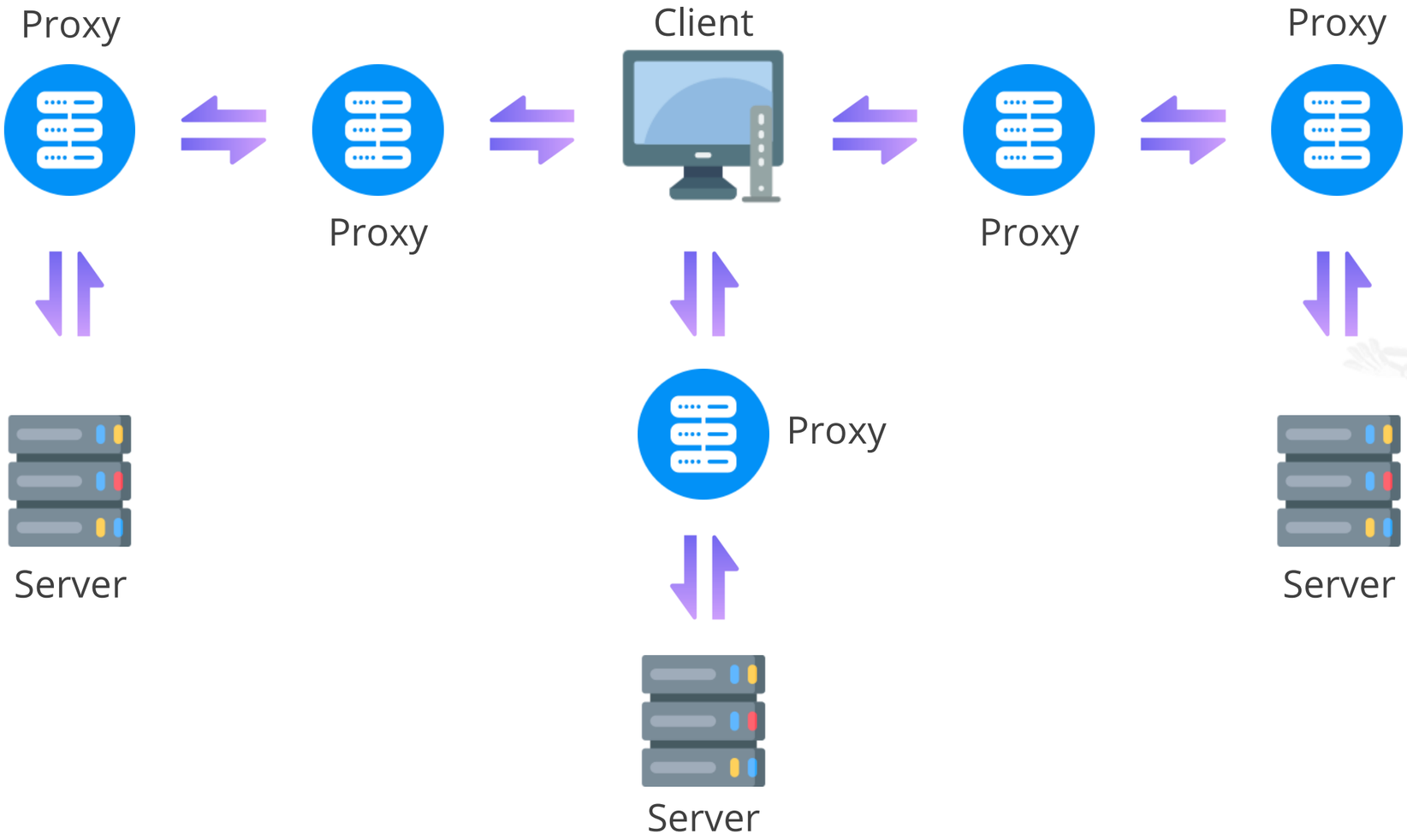
By the end of this lesson, you will be able to:

- Visualize a client-server architecture which includes REST APIs
- Identify GET, POST, PUT, and DELETE HTTP methods
- List the elements of GET, POST, PUT, and DELETE HTTP requests
- Analyze HTTP responses and response codes

Fundamentals of Client-Server Architecture

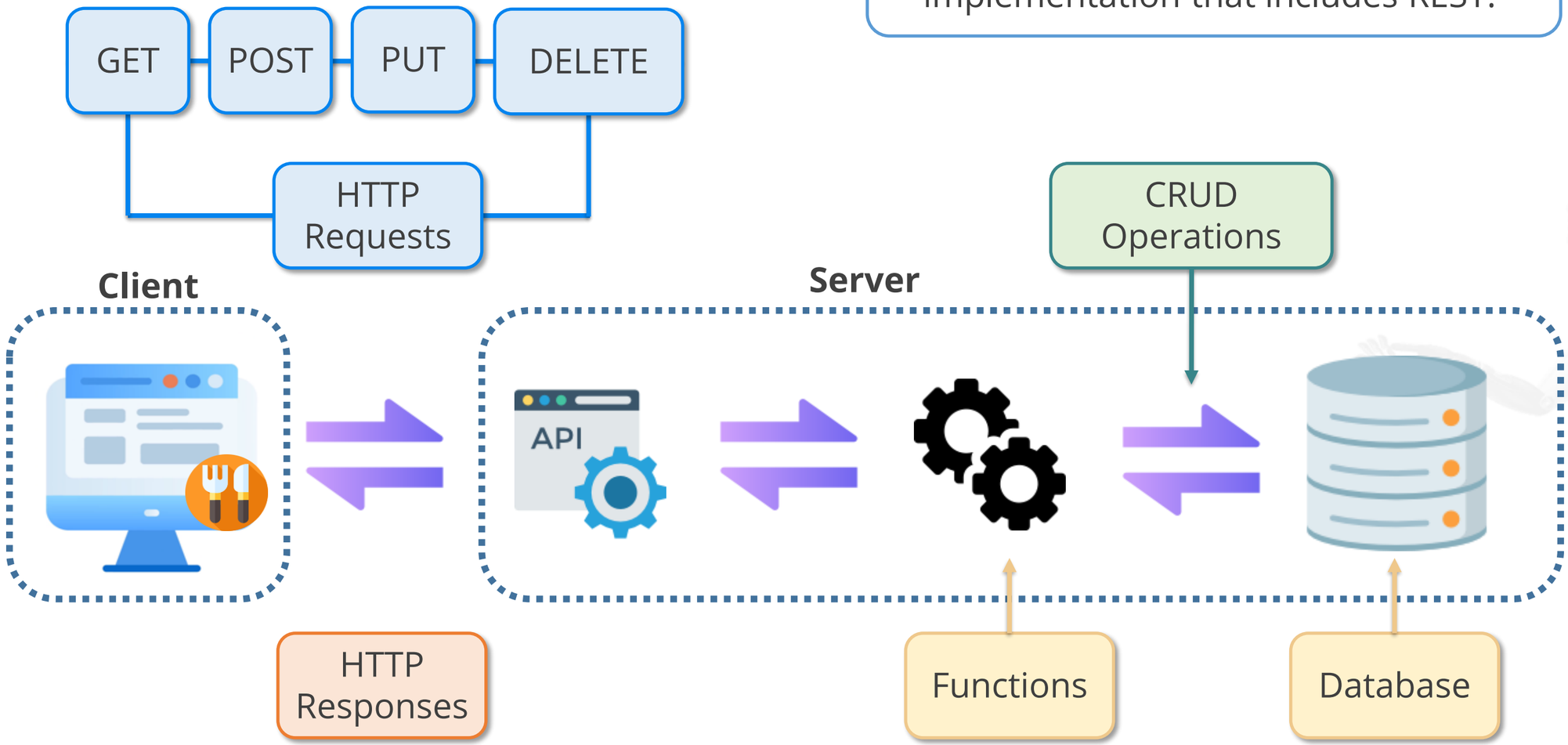
Client-Server Architecture

Here is an example of a simple client-server system with a client, proxies, and servers:



Client-Server Architecture

Here is an illustration of a client-server implementation that includes REST:



Hypertext Transfer Protocol (HTTP)

Distributed application system with a clear demarcation between the requesting system and the system servicing the request

HTTP is a client-server protocol used to access and fetch resources from the Web.

Rules that govern the exchange of data between systems

Web resources include HTML documents, images, text, videos, and so on

HTTP Requests

Components of a HTTP Request

The three parts of a HTTP request are:

1. Request Line

```
DELETE /booking/delete HTTP/1.1
```

2. Header

```
Host: www.johnrestaurant.com
Content-Type: application/json
```

3. Payload

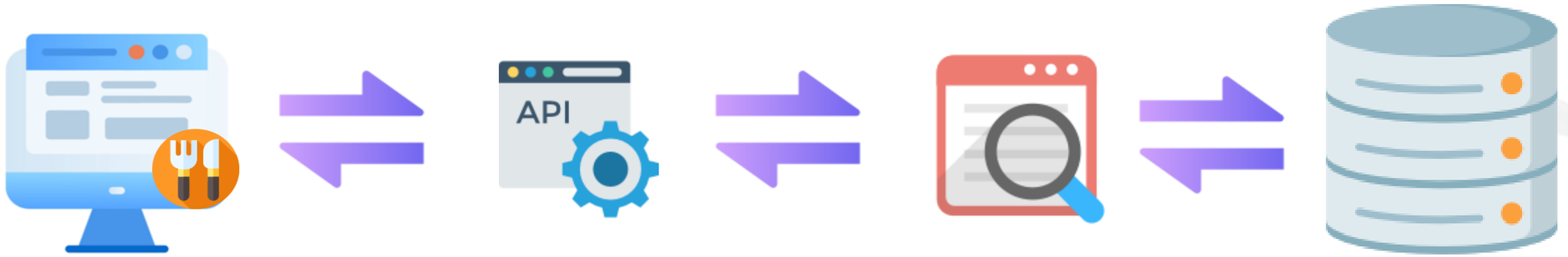
```
{
        “booking_id” : 12112,
}
```

GET HTTP Method

The GET HTTP method is used to retrieve data from the server.

The GET HTTP request is sent to API with ID to be retrieved.

API calls retrieveBooking() function, which looks for record in database.



API parses information in JSON and returns via HTTP response.

If the record is found, the information is returned to API.

GET HTTP Request

Header

application/json

Method

GET

Endpoint

https://www.johnrestaurant.com/booking/retrieve?booking_id=12112

Parameters

Payload

POST HTTP Method

The POST HTTP method is used to create a fresh datapoint in the server.

The POST HTTP request is sent to API with payload.

API calls a function that writes data into database.

API returns the status of record creation and reference ID via HTTP response (in JSON).

POST HTTP Request

Header

application/json

Method

POST

Endpoint

https://www.johnrestaurant.com/booking/create

Parameters

Payload

JSON

```
{"booking": {
      "name" : "Jane Doe", "phone" : "4774722213"
      "table_num" : "4"
      "date" : "04-12-2022", "time" : "1330"}
}
```

PUT HTTP Method

The PUT HTTP method will update the existing datapoint on the server.

The PUT HTTP request is sent to API with payload.

API calls a function that updates existing data with payload data.

API returns the status of record update via HTTP response (in JSON).

PUT HTTP Request

Header

application/json

Method

PUT

Endpoint

https://www.johnrestaurant.com/booking/update

Parameters

Payload

JSON

```
{
      “booking_id” : 12112,
      “time” : “1430”}
}
```

DELETE HTTP Method

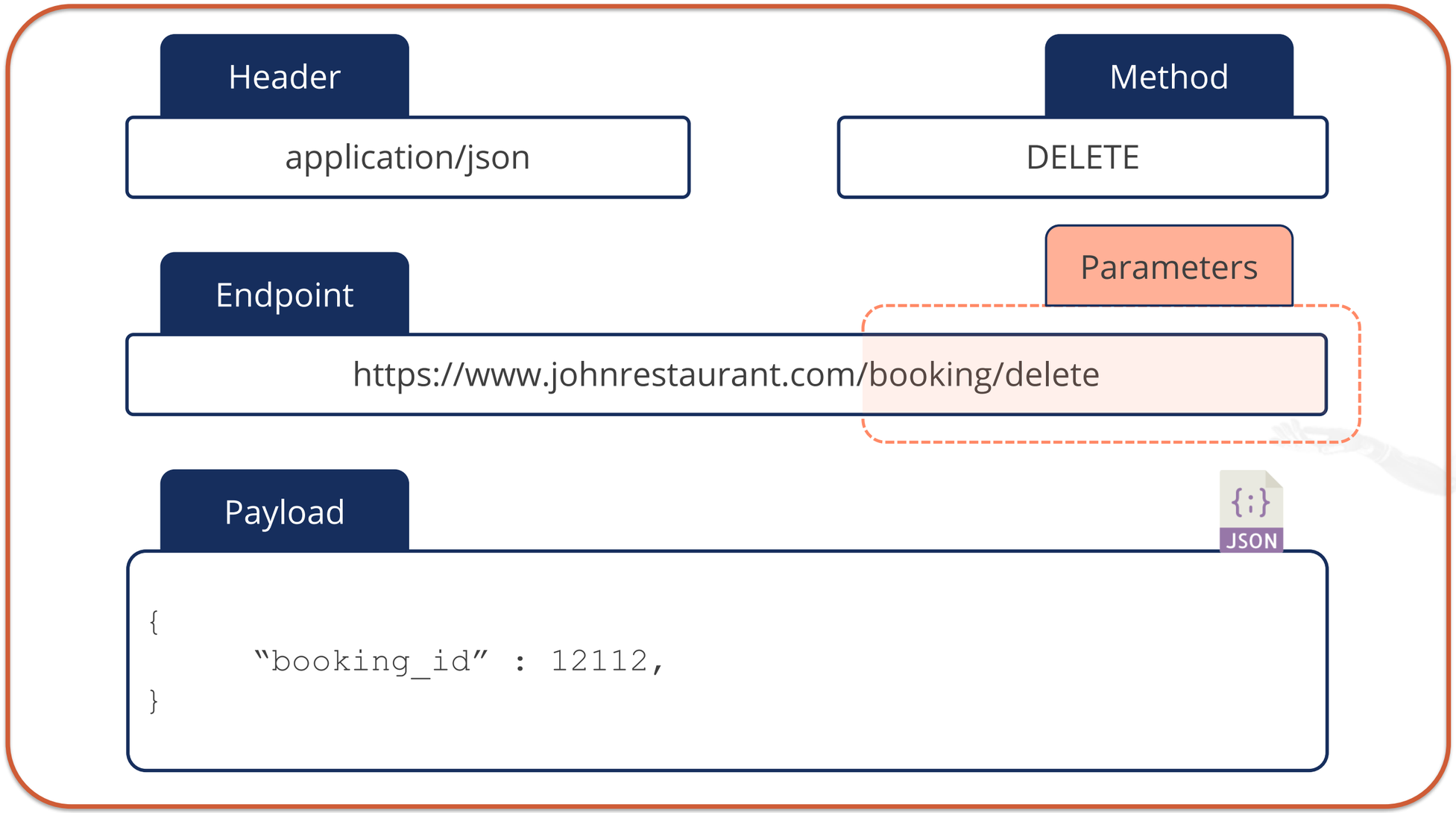
The DELETE HTTP method will delete an existing datapoint on the server.

The DELETE HTTP request is sent to API with payload.

API calls a function that deletes existing datapoint.

API returns the status of record deletion via HTTP response (in JSON).

DELETE HTTP Request



HTTP Response

Components of a HTTP Response

An HTTP response has a similar set of components as an HTTP request.

1. Status Line

```
HTTP/1.1 200 OK
```

2. Header

```
Date: Thu, 11 Aug 2022 14:36:41
GMT
Server: Apache/2.4.41 (Ubuntu)
Content-Length: 15
Content-Type:
application/json;charset=UTF-8
```

3. Body

```
{
        “status” : “SUCCESS”
}
```

HTTP Response Codes

HTTP response codes specify whether an HTTP request was successful.

1xx	Informational	A request is received and processed.

2xx	Successful	A request is received, understood, and accepted.

3xx	Redirection	Further action is required to service a request.

4xx	Client Error	A request has syntax errors or cannot be fulfilled.

5xx	Server Error	The server failed to fulfill a valid request.

Key Takeaways

- Client-server systems that make use of REST APIs and HTTP methods are a common and effective form of distributed computing.
- HTTP is a client-server protocol used to access and fetch resources from the Web.
- HTTP requests comprise a request line, header, and payload.
- HTTP responses are split across five classes depending on the nature of the response.