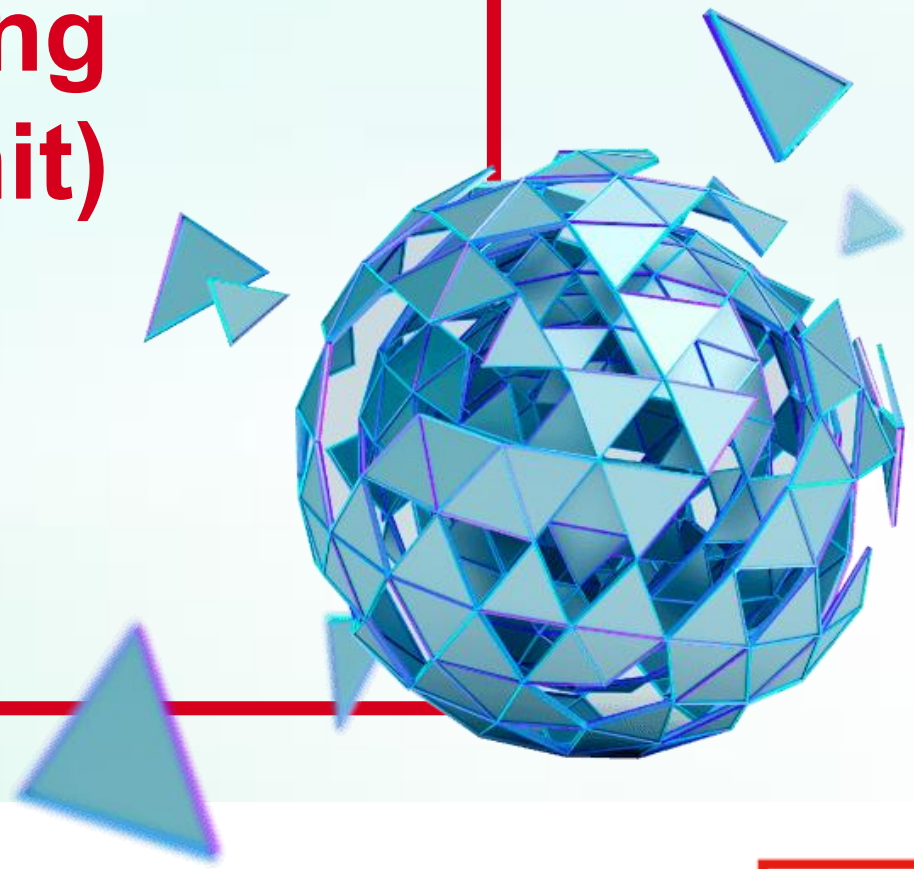
API Automated Testing with RestSharp (NUnit)



Truong Nguyen/NashTech

Nash Tech.

Agenda

1. What is API Testing
2. Why we do API tests
3. API Testing with RestSharp
4. Validate JSON Schema
5. Steps for building an automation API testing framework
6. Exercises

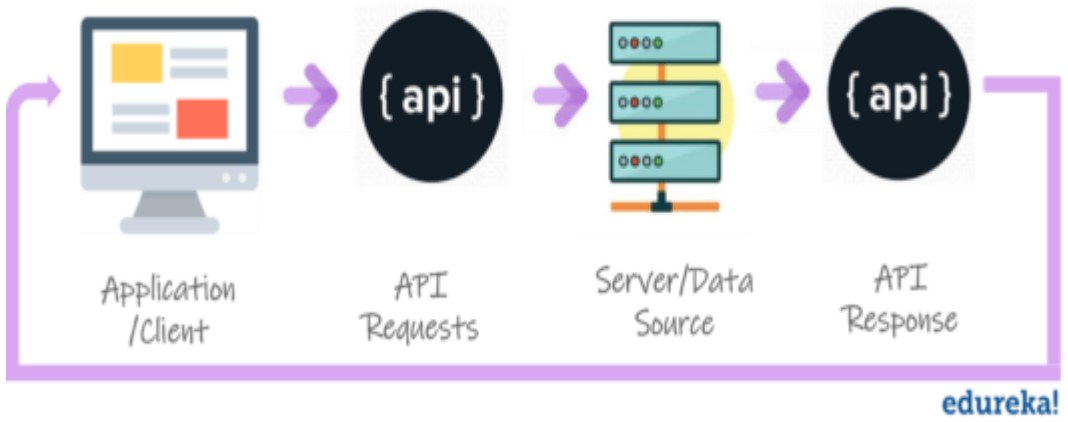
What is an API?

- API stands for “**A**pplication **P**rogramming **I**nterface.”
- API is an interface to an application designed for other computer systems to use.
- APIs can be used on web-based systems, operating systems, database systems and computer hardware.

What Is API & API Testing

- API enables communication and data exchange between two separate software system.

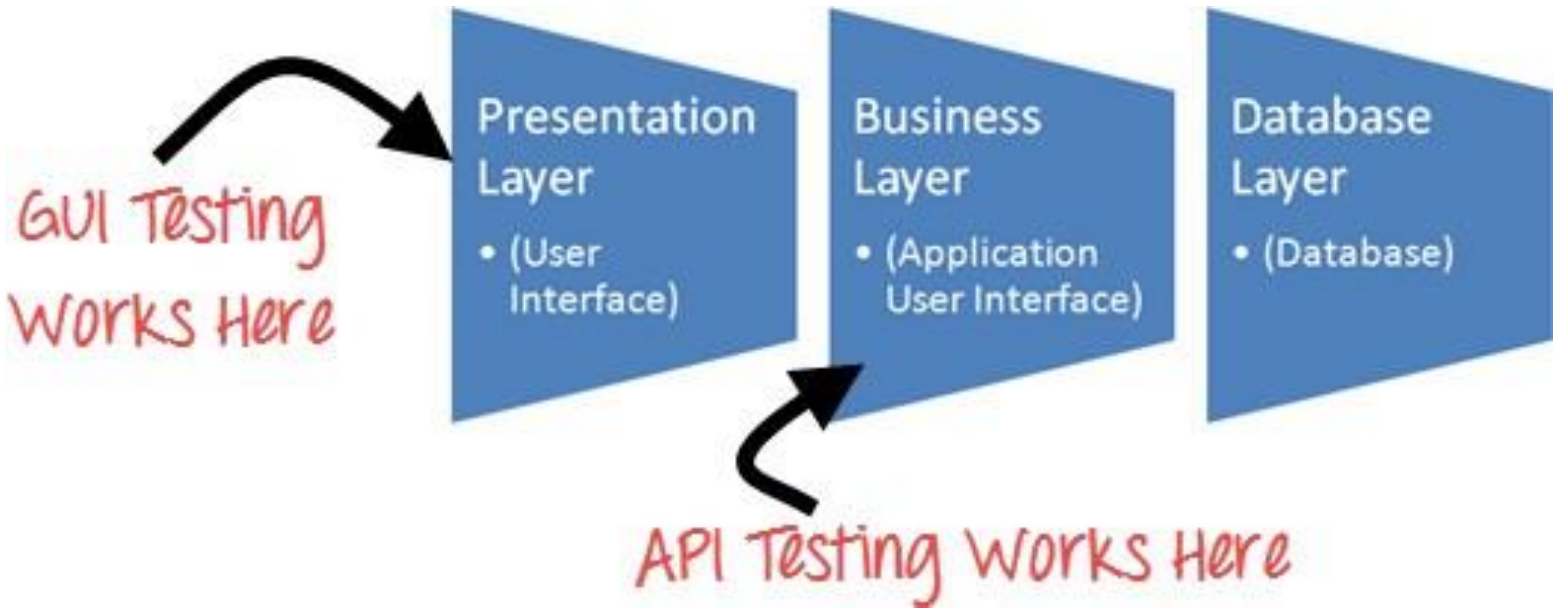
- The purpose of API Testing is to check the functionality, reliability, performance, and security of the programming interfaces.



What is API Testing?

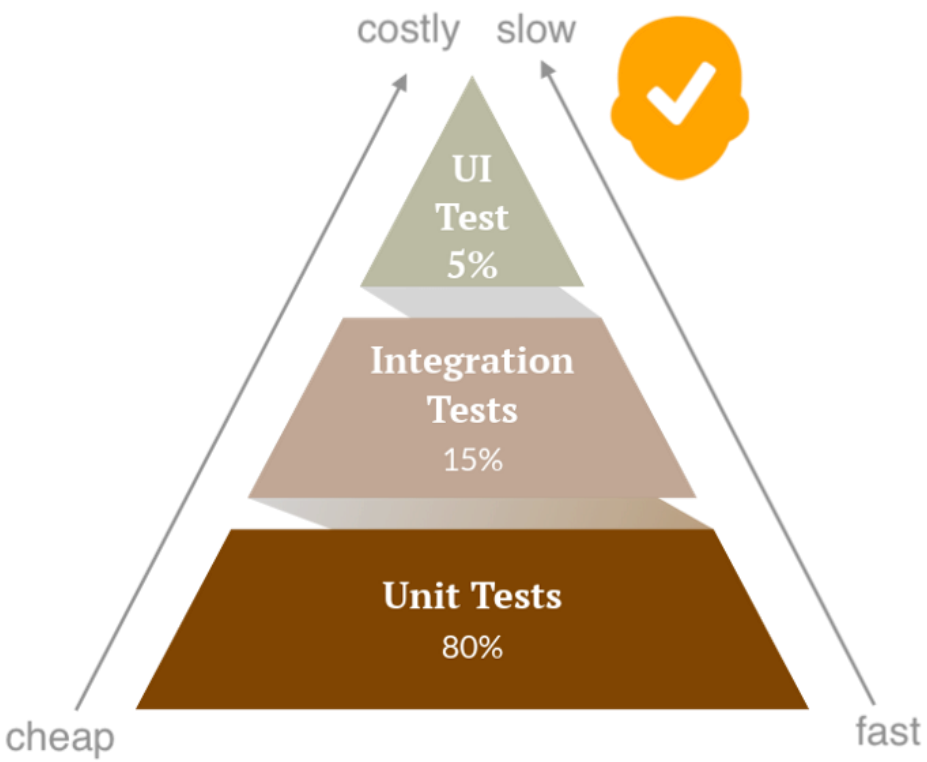
- It is a type of software testing that involves testing APIs directly.
- It is to check whether the API meets expectations in terms of functionality, reliability, performance, and security of an application
- In API Testing our main focus will be on a Business logic layer of the software architecture

API Testing


GUI Testing
Works Here
Presentation Layer
• (User Interface)
Business Layer
• (Application User Interface)
Database Layer
• (Database)
API Testing Works Here

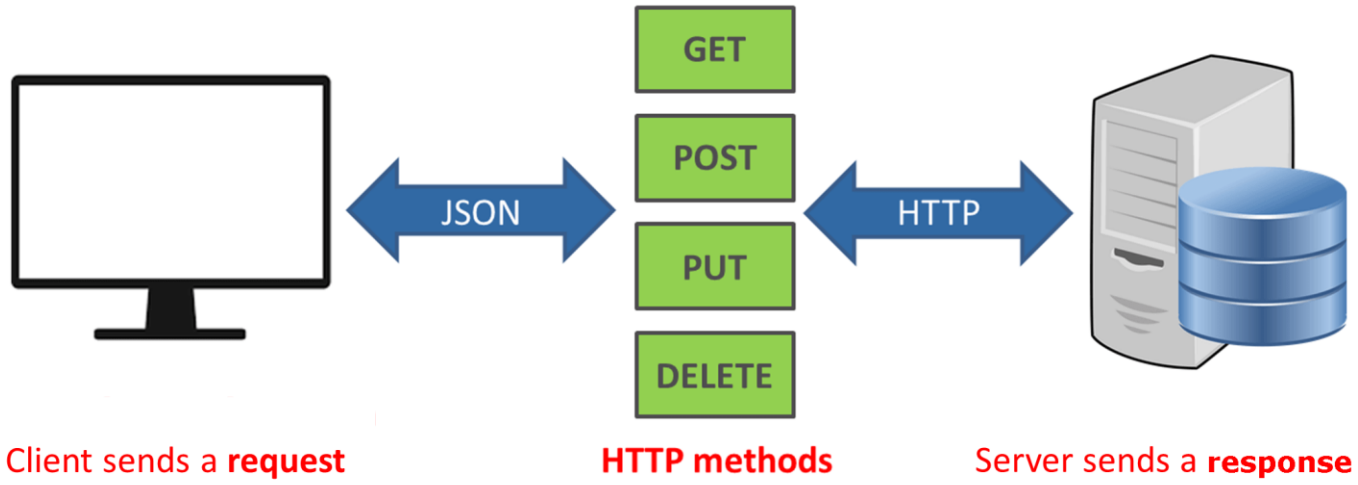
Why We Do API Tests

- Identifies bugs before it goes to UI
- Effective testing at a lower level over high-level
- Reduces future efforts to fix defects
- Time-saving



What is REST API

- REST is acronym for **RE**presentational **S**tate **T**ransfer
- It referred to as RESTful API or RESTful web service
- It is an application program interface (API) that uses HTTP requests to GET, PUT, POST and DELETE data.



HTTP Basic Concepts

- HTTP stands for Hypertext Transfer Protocol
- ***HTTP Request*** is a packet of information that one computer sends to another computer to communicate something

HTTP Basic Concepts

- **HTTP Request Structure**
  - HTTP Verbs : GET, POST, PUT, DELETE …
  - A set of HTTP Headers
    - type of Browser
    - type of content
    - what type of response is accepted in return
  - A body or payload
    - POST and PUT can have payloads, GET and DELETE can not.

HTTP Basic Concepts

- ***HTTP Response*** is the packet of information sent by *Server* to the *Client* in response to an earlier *Request* made by *Client*

HTTP Basic Concepts

- Example of HTTP Response

```
HTTP/1.1 200 OK
Date: Fri, 30 Jun 2017 13:50:11 GMT
Connection: close
Content-Type: application/json
```


```
{
  "projects": {
    "project": [
      {
        "id":  1,
        "name": "A New Project",
        "position": 0,
        "description": "",
        "state": "active",
        "created-at": "2017-06-27T12:25:26+01:00",
        "updated-at": "2017-06-27T12:25:26+01:00"
      }
    ]
  }
}
```

HTTP Basic Concepts

- **HTTP Status Codes**
  - *1xx - Informational*
  - *2xx - Success e.g. 200 Success*
  - *3xx - Redirection e.g. 302 Temporary Redirect*
  - *4xx - Client Error e.g. 400 Bad Request, 404 Not Found*
  - *5xx - Server Error e.g. 500 Internal Server Error*
- ***Payloads**: JSON or XML*

HTTP Basic Concepts

- Public Request (no auth)
- Authentication
  - Basic Authentication Headers
  - Bearer Authentication Headers
  - Custom Headers
  - Session Cookies

```
POST http://www.compendiumdev.co.uk/apps/mocktracks/reflect.php HTTP/1.1
Authorization: Basic Ym9iOmRvYmJz
```

```
POST http://www.compendiumdev.co.uk/apps/mocktracks/reflect.php HTTP/1.1
X-APPLICATION_KEY: asds-234j-werw
```

How to create a C# Nunit Project

1. Install .Net Core SDK
2. Install Visual Studio Code (add extensions C#, C# Extensions)
3. Create a Nunit project by CMD: dotnet new nunit -o <projectName> -f net6.0
4. Restore / Build project by CMD: dotnet restore / dotnet build
5. Add project reference
   - dotnet add <currentProject.csproj> reference <pathOtherProejct.csproj>
6. Run test by CMD
   - dotnet test
   - dotnet test --filter Name~<testCaseName>
   - dotnet test --filter "TestCategory=<categoryName>"

API test case example in RestSharp (NUnit)

```
<PackageReference Include="RestSharp" Version="110.2.0" />
```

```
[Test]
| 0 references
public async Task GetBooksSuccessfulWhenExistData()
{
    RestClient restClient = new RestClient("https://demoqa.com");
    RestRequest request = new RestRequest("BookStore/v1/Books")
        .AddHeader("Accept", "application/json");

    var response = await restClient.ExecuteGetAsync(request);
    response.StatusCode.Should().Be(HttpStatusCode.OK);
    response.Content.Should().NotBeNull();
}
```

The response

After all RestSharp yields the response as an object literal containing properties such as status, body, headers, duration.

Name	Value
response	"StatusCode: OK, Content-Type: application/json, Content-Length: 4514)"
Content	"{\"books\":[{\"isbn\":\"9781449325862\",\"title\":\"Git Pocket Guide\",\"subTi...
ContentEncoding	{}
ContentHeaders	Count = 2
ContentLength	4514
ContentType	"application/json"
Cookies	{System.Net.CookieCollection}
ErrorException	null
ErrorMessage	null
Headers	Count = 6
IsSuccessful	true
RawBytes	{byte[4514]}
Request	{RestSharp.RestRequest}
ResponseStatus	Completed
ResponseUri	{https://demoqa.com/BookStore/v1/Books}
RootElement	null
Server	"nginx/1.17.10 (Ubuntu)"
StatusCode	OK
StatusDescription	"OK"
Version	{1.1}

Assertions

Assertions are statements you create that check one aspect of the HTTP response. You can create multiple assertions for one check that assert various aspects of a response, for example:

- HTTP response status equals 200.
- HTTP response body equals the text “success”.
- HTTP response time is lower than 2000 milliseconds.
- HTTP response header “X-Custom-Header” equals “SomeValue”.
- HTTP response JSON object has a key called “projectName” with a value “Project”

Methods

Method: tells the server what kind of action the client wants the server to take in. Some most commonly seen in API's are:

- **GET** - Asks the server to retrieve a resource.
- **POST** - Asks the server to create a new resource.
- **PUT** - Asks the server to edit/update an existing resource.
- **DELETE** - Asks the server to delete a resource.
- **PATCH** - Asks the server to update partial modifications an existing resource.

BookStore

GET	/BookStore/v1/Books
POST	/BookStore/v1/Books
DELETE	/BookStore/v1/Books
GET	/BookStore/v1/Book
DELETE	/BookStore/v1/Book
PUT	/BookStore/v1/Books/{ISBN}

GET Method With Query String

GET /BookStore/v1/Book

Parameters

Try it out

Name	Description
ISBN * required string (query)	ISBN

Responses

Response content type application/json

Code	Description
200	Success

{{demoqa}}/BookStore/v1/Book?ISBN=9781593275846

GET Method

```
[Test]
| 0 references
public async Task GetDetailBookSuccessfullWithValidId()
{
    var book = new BookResponseDto
    {
        Isbn = "9781593275846",
        Title = "Eloquent JavaScript, Second Edition"
    };

    RestClient restClient = new RestClient("https://demoqa.com");
    RestRequest request = new RestRequest("BookStore/v1/Book")
        .AddHeader("Accept", "application/json")
        .AddParameter("ISBN", book.Isbn);

    var response = await restClient.ExecuteGetAsync(request);
    var actualBook = JsonConvert.DeserializeObject<BookResponseDto>(response.Content);

    response.StatusCode.Should().Be(HttpStatusCode.OK);
    actualBook.Should().BeEquivalentTo(book);
}
```

POST: Generate Token (API document)

POST `/login` Realizar login

A duração do token retornado em authorization é de 600 segundos (10 minutos). Caso esteja expirado irá receber status code 401 (Unauthorized).

Parameters — Try it out

Name	Description
user * required object *(body)*	Example Value | Model `{ "email": "fulano@qa.com", "password": "teste" }` Parameter content type application/json

Responses — Response content type: application/json

Code	Description
200	Login realizado com sucesso Example Value | Model `{ "message": "Login realizado com sucesso", "authorization": "Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1bWFpbCI6ImZ1bGFub0BxYS5jb20iLCJwYXNzd29yZCI6InR1c3R1IiwiaWF0IjoxNTg5NzU4NzQ2LCJ1eHAiOjE1ODk3Njg3NDZ9.B6TASHV8k9xBerz4NSeFB1AZGSDhZ1qESt767M0567I" }`
400	E-mail e/ou senha inválidos

https://serverest.dev/#/Login/post_login

Exercise

1. Login to ‘https://demoqa.com’
2. Get token (print)
3. Verify the token is not null

POST: Generate Token (Rest Sharp)

```
[Test]
0 references
public async Task GetTokenSuccessfulWithValidAccount()
{
    RestClient restClient = new RestClient("https://serverest.dev");
    var user = new UserRequestDto
    {
        Email = "fulano@qa.com",
        Password = "teste"
    };

    var request = new RestRequest("login")
        .AddHeader("Accept", "application/json")
        .AddBody(user, ContentType.Json);

    var response = await restClient.ExecutePostAsync(request);
    var result = (dynamic)JsonConvert.DeserializeObject(response.Content);

    response.StatusCode.Should().Be(HttpStatusCode.OK);
    ((string)result["authorization"]).Should().NotBeNull();
```

POST: Basic/Bearer Authorization(API document)

POST /BookStore/v1/Books

Parameters

Cancel

Name	Description
addListOfBooks * required object (body)	Edit Value | Model


```
{
  "userId": "15868515-1e4a-4166-b2aa-22e754c04b8e",
  "collectionOfIsbns": [
    {
      "isbn": "9781449365035"
    }
  ]
}
```


Cancel

Parameter content type

application/json

Execute | Clear

https://demoqa.com/swagger/#/BookStore/BookStoreV1BooksPost

Json Schema

JSON Schema is a vocabulary that allows you to **annotate** and **validate** JSON documents.

Benefits

- Describes your existing data format(s).
- Provides clear human- and machine- readable documentation.
- Validates data which is useful for:
  - Automated testing.
  - Ensuring quality of client submitted data.

Example

JSON Object:

```
[
{
"name": "John",
"age": 10
}
]
```

JSON Schema Example:

```
{
"type": "array",
"items":
{
"type": "object",
"Properties":
{
"name": {"type": "string"},
"age": {"type": "integer"}
}
}
}
```

API Response Example

A body of the API response example

```
Yielded:                                                    cypress_runner.js:191069
▼Object i
  ▼body:
    ▼books: Array(8)
      ▼0:
          author: "Richard E. Silverman"
          description: "This pocket guide is the perfect on-the-job companion to Git, the distributed ver…
          isbn: "9781449325862"
          pages: 234
          publish_date: "2020-06-04T08:48:39.000Z"
          publisher: "O'Reilly Media"
          subTitle: "A Working Introduction"
          title: "Git Pocket Guide"
          website: "http://chimera.labs.oreilly.com/books/1230000000561/index.html"
        ▶[[Prototype]]: Object
      ▶1: {isbn: '9781449331818', title: 'Learning JavaScript Design Patterns', subTitle: "A JavaScript …
      ▶2: {isbn: '9781449337711', title: 'Designing Evolvable Web APIs with ASP.NET', subTitle: 'Harness…
      ▶3: {isbn: '9781449365035', title: 'Speaking JavaScript', subTitle: 'An In-Depth Guide for Program…
      ▶4: {isbn: '9781491904244', title: "You Don't Know JS", subTitle: 'ES6 & Beyond', author: 'Kyle Si…
      ▶5: {isbn: '9781491950296', title: 'Programming JavaScript Applications', subTitle: 'Robust Web Ar…
      ▶6: {isbn: '9781593275846', title: 'Eloquent JavaScript, Second Edition', subTitle: 'A Modern Intr…
      ▶7: {isbn: '9781593277574', title: 'Understanding ECMAScript 6', subTitle: 'The Definitive Guide f…
       length: 8
      ▶[[Prototype]]: Array(0)
    ▶[[Prototype]]: Object
    duration: 1463
  ▶headers: {server: 'nginx/1.17.10 (Ubuntu)', date: 'Wed, 17 Nov 2021 16:25:59 GMT', content-type: 'app…
    status: 200
  ▶[[Prototype]]: Object
```

Create JSON Schema from API response

1. Open online tool: https://www.liquid-technologies.com/online-json-to-schema-converter
2. Copy json object from response and paste to the online tool to convert.

Sample JSON Document


```
{
    "books": [
        {
            "isbn": "9781449325862",
            "title": "Git Pocket Guide",
            "subTitle": "A Working Introduction",
            "author": "Richard E. Silverman",
            "publish_date": "2020-06-04T08:48:39.000Z",
            "publisher": "O'Reilly Media",
            "pages": 234,
            "description": "This pocket guide is the perfect on-the-job companion to Git, the distributed version control system. It provides a compact, readable introduction to Git for new users, as well as a reference
            "website": "http://chimera.labs.oreilly.com/books/1230000000561/index.html"
```


Options

Generate Schema

Infered JSON Schema


```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "type": "object",
  "properties": {
    "books": {
      "type": "array",
      "items": [
        {
          "type": "object",
          "properties": {
            "isbn": {
              "type": "string"
            },
            "title": {
              "type": "string"
            },
            "subTitle": {
              "type": "string"
            },
```

JSON Schema file

serverest_getallusers_schema.json

Unsplash.APITesting > Resources > Schema > serverest_getallusers_schema.json > ...

```
1  {
2    "$schema": "https://json-schema.org/draft/2019-09/schema",
3    "$id": "http://example.com/example.json",
4    "type": "object",
5    "default": {},
6    "title": "Root Schema",
7    "required": [
8      "quantidade",
9      "usuarios"
10   ],
11   "properties": {
12     "quantidade": {
13       "type": "integer",
14       "default": 0,
15       "title": "The quantidade Schema",
16       "examples": [
17         14
18       ]
19     },
20     "usuarios": {
21       "type": "array",
22       "default": [],
23       "title": "The usuarios Schema",
24 >     "items": {…
66       }
67     }
68   }
69 }
```

https://serverest.dev/#/Usu%C3%A1rios/get_usuarios

Validate Schema (Rest Sharp)

```
public const string GetAllUserSchema = @"Resources\Schema\serverest_getallusers_schema.json";

[Test]
0 references
public async Task GetUsersSuccessfulWhenExistData()
{
    RestClient restClient = new RestClient("https://serverest.dev");
    var request = new RestRequest("usuarios")
        .AddHeader("Accept", "application/json");

    var response = await restClient.ExecuteGetAsync(request);
    var schema = await JsonSchema.FromJsonAsync(JsonFileUtility.ReadJsonFile(GetAllUserSchema));
    var result = JsonConvert.DeserializeObject<GetAllUserResponseDto>(response.Content);

    response.StatusCode.Should().Be(HttpStatusCode.OK);
    schema.Validate(response.Content).Should().BeEmpty();
```

Demo

Scenario 2

1. Use the POST method in ’https://gorest.co.in/’ to create a new user.
2. Validate status code.
3. Validate the content of the response body.

Steps for building an automation framework

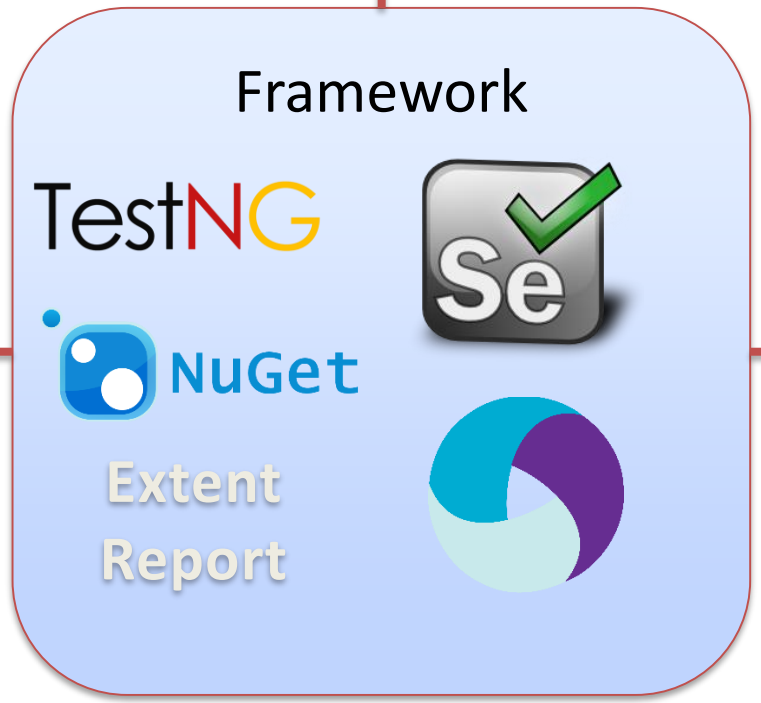
1. Framework Architecture
2. Build Framework Core Component
3. Build Framework Test Component

Framework Architecture

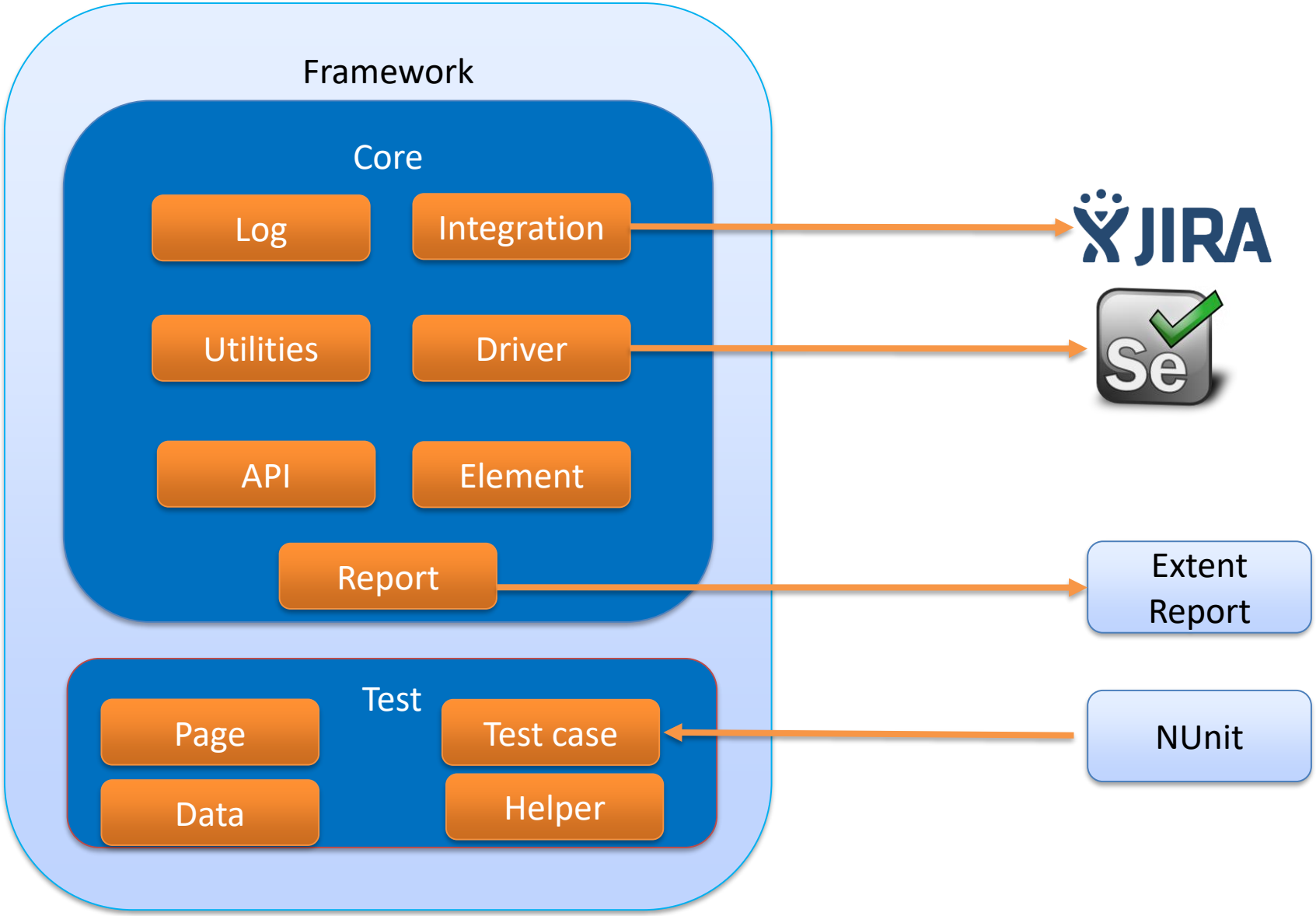
Framework Architecture

Choose Toolset:

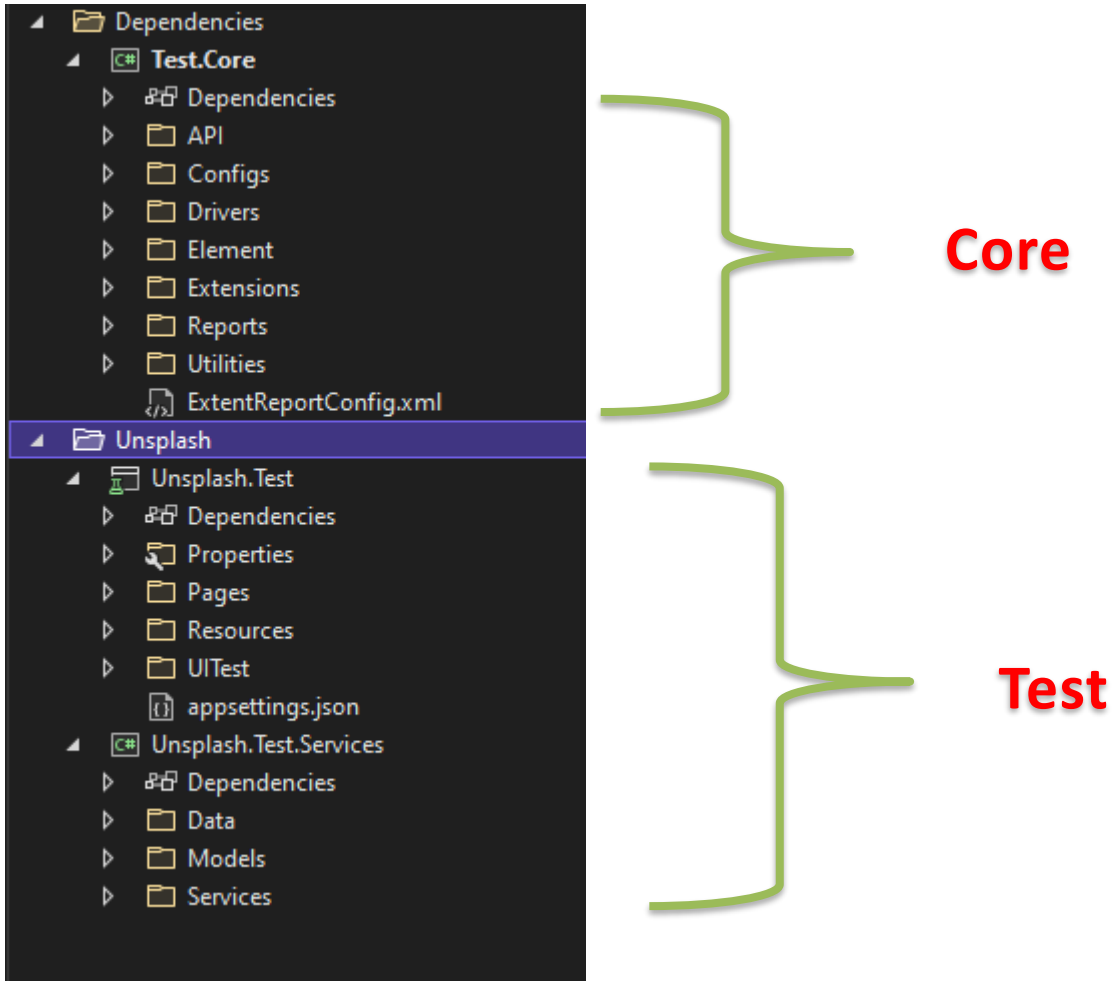
- Test Runner
- Automation Tool
- Build Tool
- Report Tool



Framework Architecture



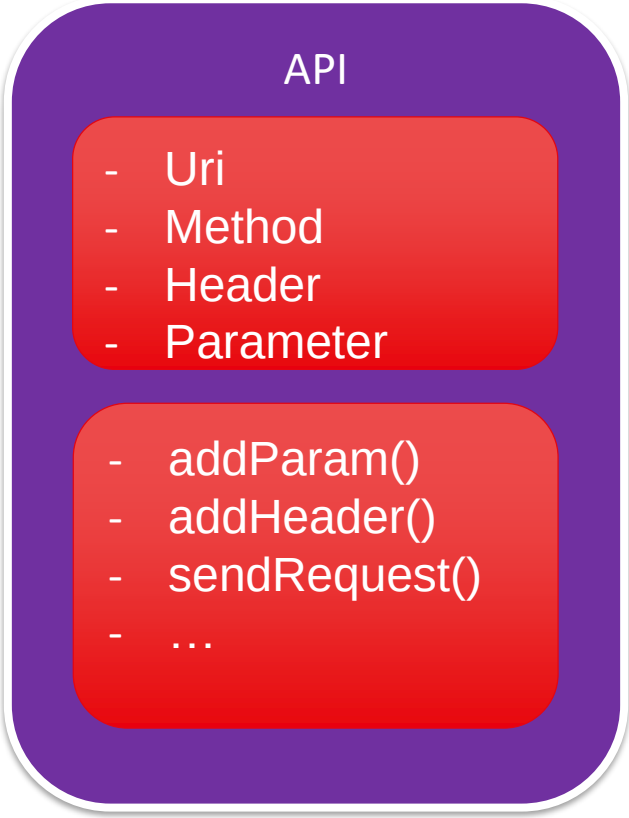
Framework Architecture



Build framework core component

1. Driver (Skip)
2. Element (Skip)
3. API
4. Report

API


API
- Uri
- Method
- Header
- Parameter
- addParam()
- addHeader()
- sendRequest()
- …

Report

API Testing for DemoQA | 13/11/2022 5:50:19 CH

Tests

Automation.DemoQA.APITesting.TestCases.GetDetailUserTest 17:50:19 CH
0h:0m:2s+146ms

Automation.DemoQA.APITesting.TestCases.GetDetailUserTest

13/11/2022 5:50:19 CH | 13/11/2022 5:50:22 CH | 0h:0m:2s+146ms

GetDetailUserUnsuccessfulWithUserIdNonExisted("user_1")

STATUS	TIMESTAMP	DETAILS
i	17:50:19	1. Get Token
i	17:50:21	2. Send request to get detail user with id 333e5b7c-6113-465f-a20b-9f8cb517def6
i	17:50:21	3. Verify unauthorized message
×	17:50:22	Message: Expected message to be "User not authorized!" with a length of 20, but "User not found!" has a length of 15, differs near "fou" (index 9).
×	17:50:22	Test ended with Fail at FluentAssertions.Execution.LateBoundTestFramework.Throw(String message) at FluentAssertions.Execution.TestFrameworkProvider.Throw (String message) at FluentAssertions.Execution.DefaultAssertionStrategy.HandleFailure(String message) at FluentAssertions.Execution.AssertionScope.FailWith(Func`1 failReasonFunc) at FluentAssertions.Execution.AssertionScope.FailWith(Func`1 failReasonFunc) at FluentAssertions.Primitives.StringEqualityValidator.ValidateAgainstLengthDifferences() at FluentAssertions.Primitives.StringValidator.Validate() at FluentAssertions.Primitives.StringAssertions`1.Be(String expected, String because, Object[] becauseArgs) at Automation.DemoQA.APITesting.TestCases.GetDetailUserTest.GetDetailUserUnsuccessfulWithUserIdNonExisted(String accountKey) in D:\My_Study\4. Mentor Automation\3. T-Sharp Automation Testing\SourceCode\C#\Non BDD\Automation.DemoQA.APITesting\TestCases\GetDetailUserTest.cs:line 77 at NUnit.Framework.Internal.TaskAwaitAdapter.GenericAdapter`1.BlockUntilCompleted() at NUnit.Framework.Internal.MessagePumpStrategy.NoMessagePumpStrategy.WaitForCompletion(AwaitAdapter awaitable) at NUnit.Framework.Internal.AsyncToSyncAdapter.Await(Func`1 invoke) at NUnit.Framework.Internal.Commands.TestMethodCommand.RunTestMethod(TestExecutionContext context) at NUnit.Framework.Internal.Commands.TestMethodCommand.Execute(TestExecutionContext context) at NUnit.Framework.Internal.Commands.BeforeAndAfterTestCommand.<>c__DisplayClass1_0.b__0() at NUnit.Framework.Internal.Commands.BeforeAndAfterTestCommand.RunTestMethodInThreadAbortSafeZone(TestExecutionContext context, Action action)

https://www.ecanarys.com/Blogs/ArticleID/305/Extent-Reports-in-Selenium-CSharp-C

Build framework test component

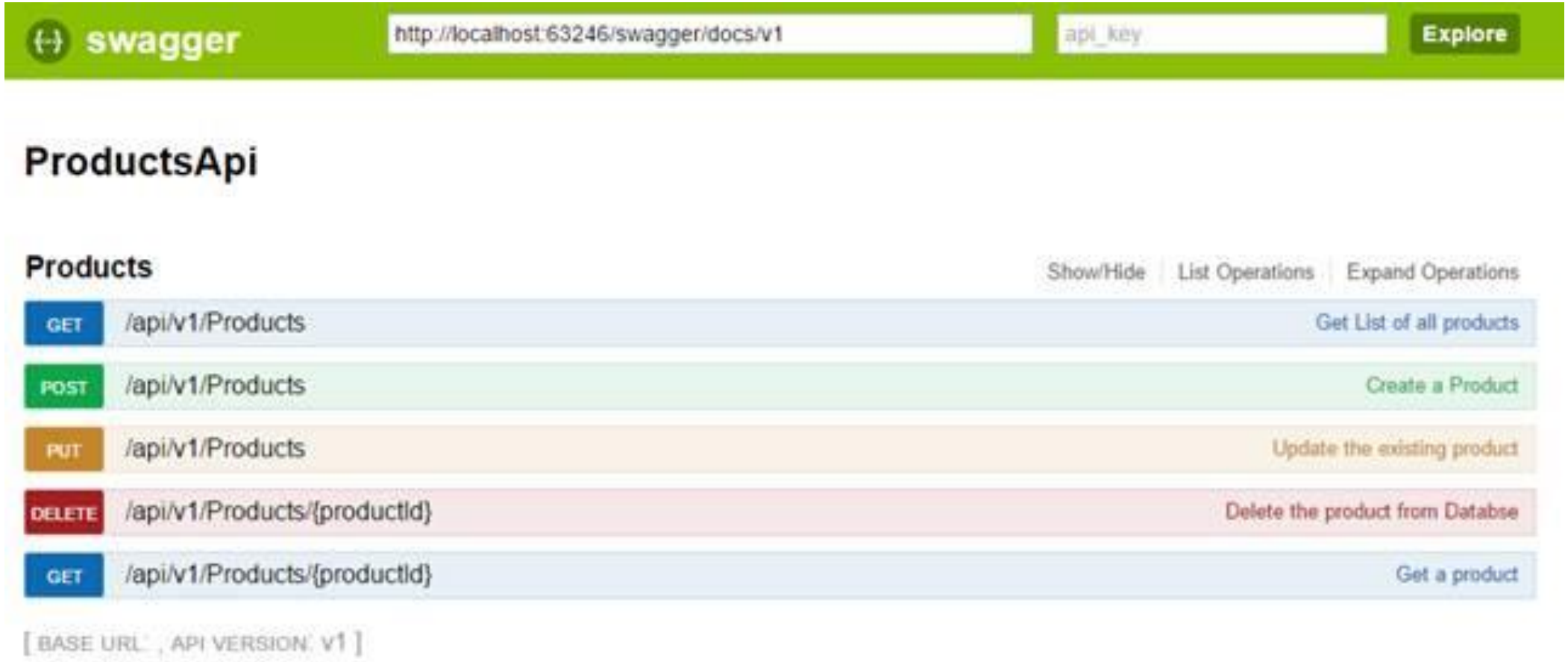
1. Test
2. Page (skip)
3. API Services

Test

1. Create TestBase Class
2. Create test classes to contain test case
    - Add all test case related to a component to a class
    - Create sub package if necessary
    - Set priority/group for test case

API Services

Create API class based on component



Exercise

1. Build an Automation API Testing framework
2. Test API for "Add book to collection" in DemoQA web


Q
&
A

References

Request:

https://restsharp.dev/intro.html#getting-started

Reporter:
https://www.extentreports.com/docs/versions/4/net/index.html

JSON Schema:

https://www.jsonschema.net/app/schemas/0

http://json-schema.org/understanding-json-schema/

Postman:

https://learning.postman.com/docs/getting-started/introduction/

Fluent Assertion:

https://methodpoet.com/fluent-assertions/

Thank you

Nash Tech.