Analyse

http://dummy.restapiexample.com/

L’api en quelques mots

L’api est une porte d’entrée vers des services de tiers. Le but est de consommer des informations comme une recherche internet mais de manière automatique.

Les api permettent d’avoir une unité technique dans la recherche d’information.

C’est également un moyen sécurisé d’aller chercher l’information (ou de fournir de l’info à) des partenaires. Seules les informations qui ont intéressantes à récupérer sont disponibles via les api ; Il est possible aussi de vendre par ce biais des morceaux d’information. C’est d’ailleurs ce qui se passe tous les jours avec les grands groupe comme Google qui fournissent de l’information via api.

L’api est plus flexible et simplifie le développement d’applications pour les développeurs car peu importe le langage informatique utilisé derrière l’api, il est facile de communiquer avec des tiers car le résultat de la recherche suit un protocole réutilisable.

Au sein d’une même entreprise des langages informatiques innovants peut émerger tant que le partage d’information entre les autres équipes se fait via api.

Test strategy

De manière générale on va tester :
-le status de la reponse (200 par exemple en cas de succès ou forcer une 400 avec de mauvais paramètres)
-les clefs de la réponse / la structure d'un GET
-la rapidité
-La bonne insertion / modification d'un POST / PUT / DELETE
-La version de l'api. Vérifier que la V1 retourne bien la bonne structure par exemple

Exemple de Gherkin pour tester l'api :

```
Feature: Test Api Users
As a developer
I want to make sure CRUD REST API works fine

Scenario: Get a contact

Given an contact id
When I send GET request to "https://reqres.in/api/users/2"
Then I get response code 200
```

Voir le Github avec les fichiers .features
Chaque fichier feature sera un scénario de test pour l'api

Framework Choice for Testing API
(pros)

Le choix va se porter sur l'association **Cucumber / Axios / NodeJs**

Cucumber car :
-il embarque la syntaxe BDD de Gherkin.
-il est compatible avec beaucoup de langages
-compatibilité Jira
-Reutilisation du Gherkin en code de test fonctionnel / Assertions

Axios car :
-protection xsrf
-mature /réputé / facile
-utilisation des credentials si besoin
-léger à l'installation
-compatible cucumber js

NodeJs car :
-la plupart des développeurs chez Believe ont des connaissances js ou php
-Le javascript servira dans d'autres situations que le test API et ce la évite de multiplier les langages de testing
-Pas besoin de logiciel spécifique. Un simple éditeur de texte fonctionne

Les trois outils présentés sont réputés et ont faits leurs preuves

QA report Template

Exemple with Api Users

CRUD api users	DEV	RC
GET Structure	PASS	ERROR
GET Status	PASS	PASS
GET fast enough	PASS	PASS
POST Structure	PASS	PASS
POST Status	ERROR	PASS
POST Effective	PASS	PASS
POST fast enough	PASS	PASS
PUT Structure		
PUT Status	…	…
…		
…		
DELETE …		

Click here to have details

Details errors :

-Get structure error :
When I send GET request to "users/2"
Then …

-Post status error :
When ..
Then …

Tutorial for test an api with BDD / Cucumber / Axios (1/3)

For our project we will use Cucumber Tool in order to launch some functionals tests
Cucumber will be use with NodeJs server and Axios library

How install

Once NodeJs and Git are installed on your machine :
1) Create an empty repository and go inside
2) Clone this github remote repo inside : https://github.com/xxx
3) Launch the command « npm test» to verify the good working of the tool
4)Write your own tests inside « features » repository

How Write a test

Each feature to test is composed by a couple of files.
myFeature.feature
/step_definitions/myFeature.js

a .feature file is a Gherkin definition (acceptance definition) with examples
a .js file is the script which respect the Gherkin interface to create a real unit test (call an api or database or …)

Tutorial for test an api with BDD / Cucumber / Axios (2/3)

Exemple : verify a GET structure data

➔*The .feature file*

The search_api.feature file is like that (the structure is really important) :

```
Feature: Test Api Users
As a developer
I want to make sure CRUD REST API works fine

Scenario: Get a contact

Given an id for api
When I send GET request to "https://reqres.in/api/users/2"
Then I get response code 200
```

The structure is a yaml structure (key => value)
The steps « Given / When / Then » are essentials in your next js file because these steps are the keys of the js methods

Tutorial for test an api with BDD / Cucumber / Axios (3/3)

Exemple : verify a GET structure data

➔*The .js file*

The steps Given / When / Then are essentials in your next js file because these steps are the keys of the js methods

```
const assert = require('assert');
const { Given, When, Then } = require('@cucumber/cucumber');
const axios = require('axios');

Given('an id for api', function () {
    this.test = '';
});

When('I send GET request to {string}', async function (path) {

    const response = await axios({
    url: path,
    method: 'get'
  })

  console.log(response.data);
});

Then('I get response code {int}', function (statuss) {
    assert.equal(this.test, statuss);
});
```

….to complete