

Les Aventuriers Archi Perdus

LG3 – Ingénierie des tests : Les outils

Livrable à l'attention de monsieur Metangmo Davy
dans le cadre de la licence professionnelle CDAD

MARION LABBÉ
PAUL DAMPFHOEFFER
STÉPHANE LITT
WILLIAM TRAHAY

2021/2022

IUT	Robert Schuman	
Institut universitaire de technologie		
	Université de Strasbourg	

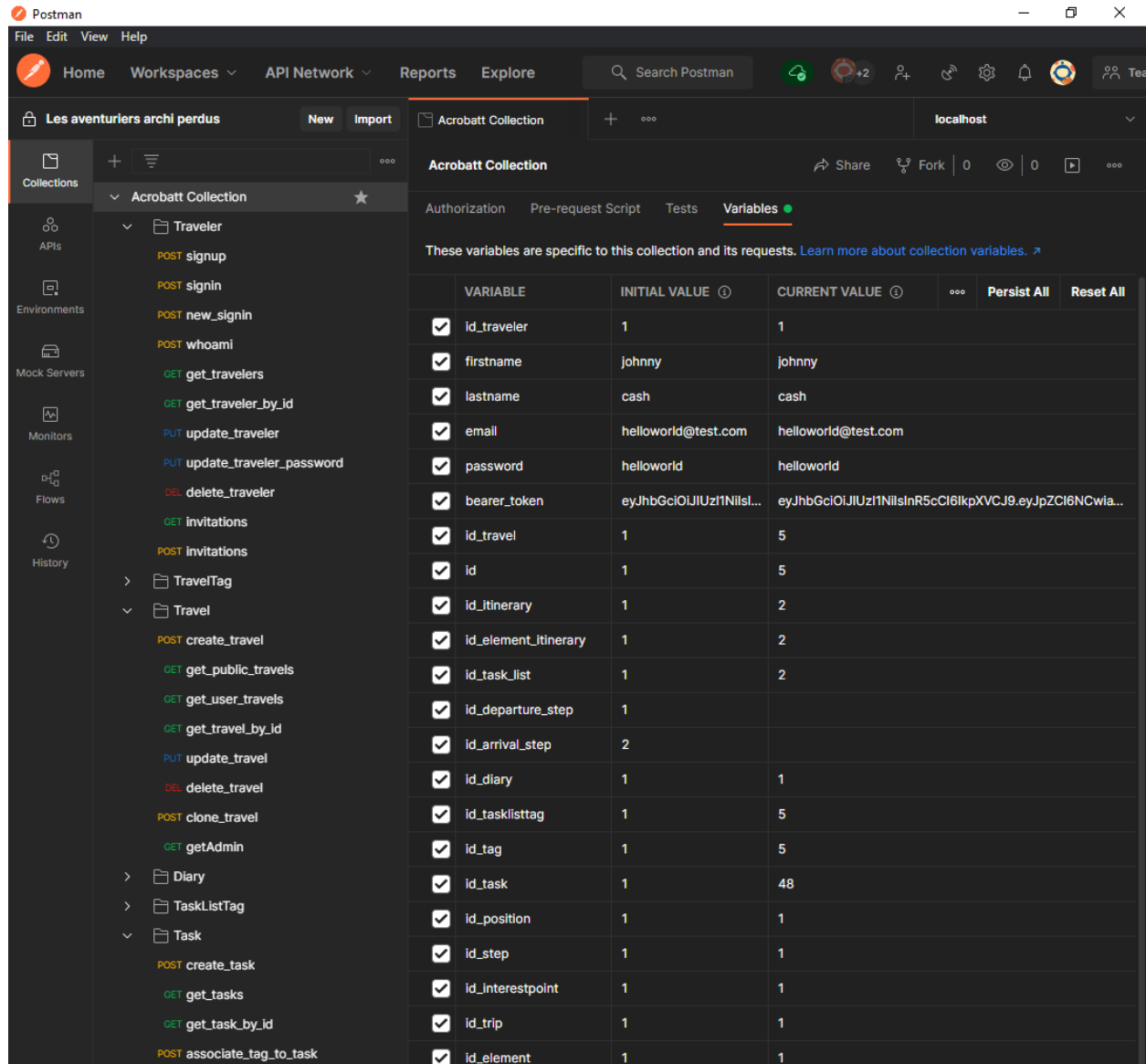
TABLE DES MATIERES

PARTIE 1 – API.....	2
POSTMAN	2
NEWMAN.....	3
SEEDERS	7
PROCESSUS	9
INTEGRATION CONTINUE	11
RESSOURCES	13
PARTIE 2 – IHM Web.....	13
INTRODUCTION	13
TESTS MIS EN PLACE	13
DIFFICULTÉS ET RETOURS D'EXPÉRIENCE	16

PARTIE 1 – API

POSTMAN

Dans la continuité de notre apprentissage lors des cours sur les tests, nous avons utilisé la plateforme Postman pour tester notre API.



The screenshot shows the Postman application interface. On the left sidebar, the 'Collections' section is expanded, showing a tree structure of collections. The 'Acrobatt Collection' is selected, and its contents are displayed in the main panel. The 'Variables' tab is active, showing a table of variables specific to this collection.

	VARIABLE	INITIAL VALUE	CURRENT VALUE		Persist All	Reset All
<input checked="" type="checkbox"/>	id_traveler	1	1			
<input checked="" type="checkbox"/>	firstname	johnny	johnny			
<input checked="" type="checkbox"/>	lastname	cash	cash			
<input checked="" type="checkbox"/>	email	helloworld@test.com	helloworld@test.com			
<input checked="" type="checkbox"/>	password	helloworld	helloworld			
<input checked="" type="checkbox"/>	bearer_token	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6ImNw...	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6ImNw...			
<input checked="" type="checkbox"/>	id_travel	1	5			
<input checked="" type="checkbox"/>	id	1	5			
<input checked="" type="checkbox"/>	id_itinerary	1	2			
<input checked="" type="checkbox"/>	id_element_itinerary	1	2			
<input checked="" type="checkbox"/>	id_task_list	1	2			
<input checked="" type="checkbox"/>	id_departure_step	1				
<input checked="" type="checkbox"/>	id_arrival_step	2				
<input checked="" type="checkbox"/>	id_diary	1	1			
<input checked="" type="checkbox"/>	id_tasklisttag	1	5			
<input checked="" type="checkbox"/>	id_tag	1	5			
<input checked="" type="checkbox"/>	id_task	1	48			
<input checked="" type="checkbox"/>	id_position	1	1			
<input checked="" type="checkbox"/>	id_step	1	1			
<input checked="" type="checkbox"/>	id_interestpoint	1	1			
<input checked="" type="checkbox"/>	id_trip	1	1			
<input checked="" type="checkbox"/>	id_element	1	1			

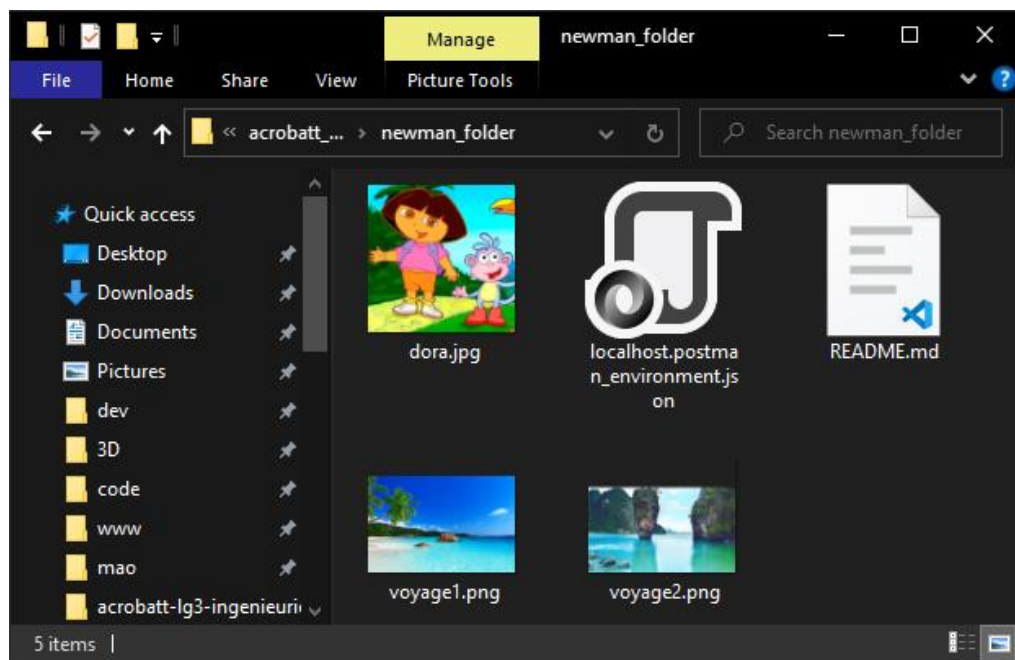
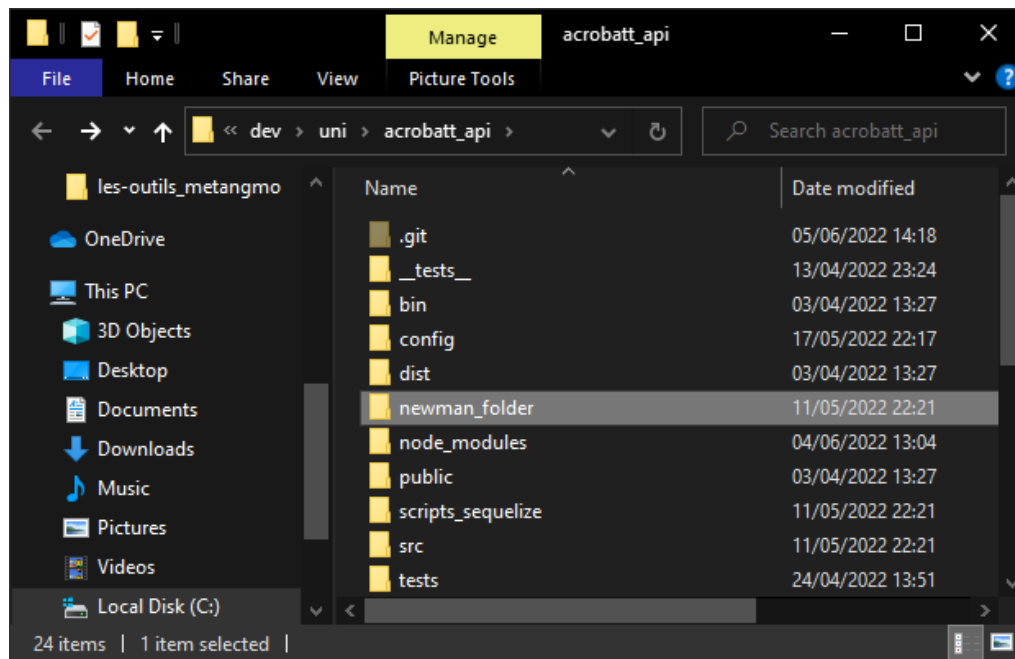
POST associate_tags_to_task	<input checked="" type="checkbox"/>	id_document	1	1
POST remove_tag_from_task	<input checked="" type="checkbox"/>	id_photo	1	1
PUT update_task	<input checked="" type="checkbox"/>	type	Point	Point
DEL delete_task	<input checked="" type="checkbox"/>	lng	7.7	7.7
TravelReview	<input checked="" type="checkbox"/>	lat	48.2	48.2
POST create_travelreview	<input checked="" type="checkbox"/>	content	New content	New content
GET get_travelreview	<input checked="" type="checkbox"/>	is_in_album	true	true
GET get_travelreview_by_id	<input checked="" type="checkbox"/>	commentary	this is a commentary	this is a commentary
PUT update_travelreview	<input checked="" type="checkbox"/>	is_finished	false	false
DEL delete_travelreview	<input checked="" type="checkbox"/>	is_public	false	false
Position	<input checked="" type="checkbox"/>	name	Untitled	Untitled
Step	<input checked="" type="checkbox"/>	dateonly	2022-01-05	2022-01-05
POST create_step	<input checked="" type="checkbox"/>	is_terminated	false	false
GET get_step	<input checked="" type="checkbox"/>	order	1	1
GET get_step_by_id	<input checked="" type="checkbox"/>	duration	1	1
PUT update_step	<input checked="" type="checkbox"/>	tagName	Nature	Nature
GET get_all_associated_interestpo...	<input checked="" type="checkbox"/>	count	1	1
DEL delete_step	<input checked="" type="checkbox"/>	description	descriptionwithnospa...	descriptionwithnospace
PUT reorder_steps	<input checked="" type="checkbox"/>	id_transportmode	1	1
InterestPoint	<input checked="" type="checkbox"/>	id_traveltag	1	1
Trip	<input checked="" type="checkbox"/>	review	5	5
TransportMode	<input checked="" type="checkbox"/>	id_travelreview	1	1
Document	<input checked="" type="checkbox"/>	travel_name	Super voyage !	Super voyage !
Photo	<input checked="" type="checkbox"/>	bearer_token_wrong	jzajdozaj	jzajdozaj
Participant	<input checked="" type="checkbox"/>	updated_firstname	test	test
Anonymous Traveler				
Archives				
Restful_Booker_CRUD_Collection				

NEWMAN

Comme nous avons pu le voir en cours, l'utilitaire en ligne de commande Newman permet d'exécuter des collections Postman et a facilité ce processus tout au long de notre développement.

Son utilisation permet de s'assurer que tout continue à fonctionner lors de l'implémentation de nouvelles fonctionnalités ou pour tester la mise à jour d'une partie de notre code.

Nous avons donc mis en place un dossier dédié appelé « newman_folder » dans le repos du projet Gitlab de l'API, afin d'avoir facilement accès à l'exécution de ces tests.



En effet, une des problématiques rencontrées fut l'exécution de tests sur une route prenant en entrée un formulaire (FormData) pour une fonctionnalité de mise en ligne de photos. La contrainte étant que sur Postman, il faut pointer sur un fichier avec un chemin en absolu, et cette technique ne permet pas aux utilisateurs de garder le même fichier, ou le même chemin de fichier, entre chaque changement d'environnement.

Nous avons donc opté pour ce dossier centralisé comprenant les fichiers concernés et l'exécution de la commande CLI Newman à sa racine.

Ce dossier est donc versionné et inclus dans notre repo Gitlab du projet de l'API, avec un fichier README pour que les membres de l'équipe y ai facilement accès.

README :

```
### Running Tests with Newman

#### For windows run the following command in this directory:
newman run https://www.getpostman.com/collections/efae735f77861799640 -e .\localhost.postman_environment.json
#### For unix run the following command in this directory:
newman run https://www.getpostman.com/collections/efae735f77861799640 -e /localhost.postman_environment.json
```

Fichier d'environnement

```
{
  "id": "dfefd99c-84c0-4c3f-9807-4b82ac9c5923",
  "name": "localhost",
  "values": [
    {
      "key": "baseUrl",
      "value": "http://localhost:8080",
      "type": "default",
      "enabled": true
    }
  ],
  "_postman_variable_scope": "environment",
  "_postman_exported_at": "2022-04-07T14:59:51.909Z",
  "_postman_exported_using": "Postman/9.14.14"
}
```

Une autre difficulté a été l'exécution multiple de requêtes, nécessaires pour créer suffisamment de données exploitables pour les prochaines exécutions.

(Exemple : il faut deux étapes pour générer un trajet, et il faut pouvoir exploiter l'ordre des étapes pour tester la requête de réarrangement d'ordre)

L'utilisation de variables pour stocker le nombre d'itérations a permis de résoudre cette problématique.

POST

{{baseUrl}}/travel/:idTravel/step

Send

Params ● Auth ● Headers (9) Body ● Pre-req. Tests ● Settings

Cookies

```
1  var jsonData = pm.response.json();
2
3  pm.collectionVariables.get("printResponse") && console.log(jsonData);
4
5  var currentCount = pm.collectionVariables.get("count")
6
7  console.log(currentCount);
8
9  if (currentCount < 3) {
10     pm.collectionVariables.set("order", currentCount);
11     pm.collectionVariables.set("count", currentCount + 1);
12     pm.test("Status code is 201", function () {
13         pm.response.to.have.status(201);
14     });
15     console.log(currentCount);
16
17     postman.setNextRequest("create_step");
18 } else {
19     pm.test("Status code is 201", function () {
20         pm.response.to.have.status(201);
21     });
22     postman.setNextRequest("get_step");
23 }
24
25 var jsonData = pm.response.json();
26 console.log(jsonData.id)
27 console.log(jsonData.element.id)
28
29 pm.collectionVariables.set("id_step", jsonData.id);
30
31 console.log("id_step = " + pm.collectionVariables.get("id_step"))
32
33 pm.collectionVariables.set("id_element", jsonData.element.id);
34
35
```

SEEDERS

Certaines données de test dépendaient de données préexistantes, nous avons donc intégré au process de lancement de notre API des seeders, permettant de générer des jeux de données au préalable.

```
'use strict';

// William Trahay, last month * transportmode and trip assoc done. transportmode ...

module.exports = {
  async up(queryInterface, Sequelize) {
    await queryInterface.bulkInsert('TransportModes', [
      {
        name: "avion",
        createdAt: new Date(),
        updatedAt: new Date()
      },
      {
        name: "velo",
        createdAt: new Date(),
        updatedAt: new Date()
      },
      {
        name: "bus",
        createdAt: new Date(),
        updatedAt: new Date()
      },
      {
        name: "marche",
        createdAt: new Date(),
        updatedAt: new Date()
      },
      {
        name: "voiture",
        createdAt: new Date(),
        updatedAt: new Date()
      },
      {
        name: "hoverboard",
        createdAt: new Date(),
        updatedAt: new Date()
      }
    ], {});
  }
};
```


'use strict'; marion.labbe, 2 weeks ago • ajout seeders

```
module.exports = {
  async up (queryInterface, Sequelize) {
    const secret = process.env.TOKEN_SECRET;
    const bcrypt = require("bcryptjs");

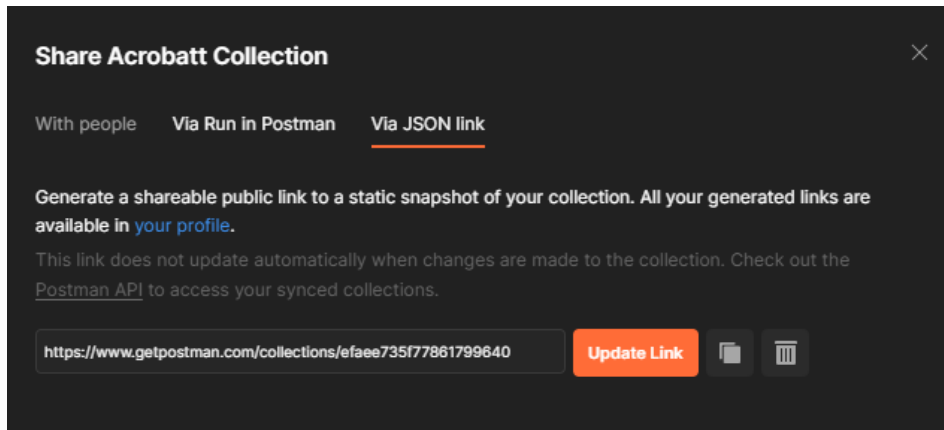
    const salt = bcrypt.genSaltSync(10, secret);
    const encrypt_pass = (password) => bcrypt.hashSync(password, salt);

    await queryInterface.bulkInsert('Travelers', [{
      firstname: "John",
      lastname: "Doe",
      email: "john@doe.com",
      password: encrypt_pass("12345"),
      createdAt: new Date(),
      updatedAt: new Date()
    },
    {
      firstname: "Dora",
      lastname: "L'exploratrice",
      email: "dora@mapadora.fr",
      password: encrypt_pass("dora"),
      createdAt: new Date(),
      updatedAt: new Date()
    },
    {
      firstname: "Jean",
      lastname: "Dupont",
      email: "jean@dupont.fr",
      password: encrypt_pass("jean"),
      createdAt: new Date(),
      updatedAt: new Date()
    }
  ], {});
}
```

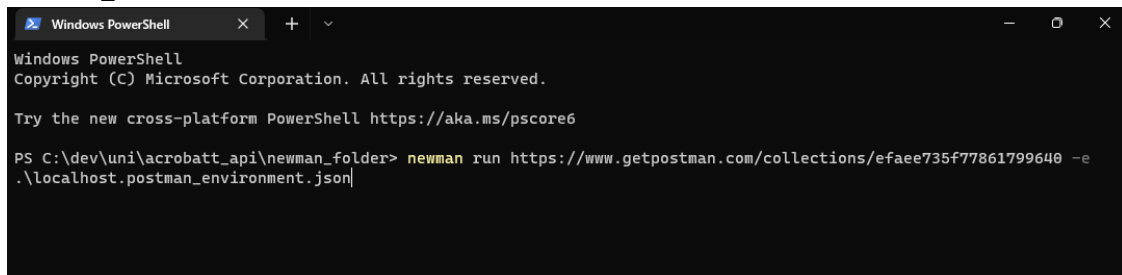
PROCESSUS

Le processus pour lancer une série de tests à l'aide de l'utilitaire Newman est le suivant :

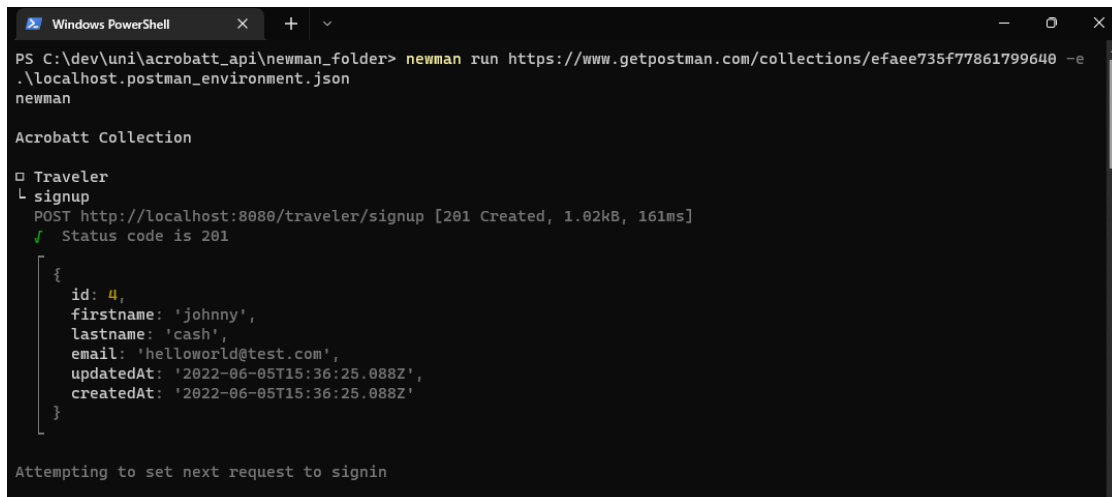
- S'assurer que la collection est à jour



- Préparer la commande en utilisant le lien vers la collection et en se positionnant dans le dossier newman_folder.



- Exécuter la commande et consulter les logs.



```
L signin
POST http://localhost:8080/traveler/signin [200 OK, 1.18kB, 107ms]
✓ Status code is 200

{
  user: {
    id: 4,
    firstname: 'johnny',
    lastname: 'cash',
    email: 'helloworld@test.com',
    createdAt: '2022-06-05T15:36:25.088Z',
    updatedAt: '2022-06-05T15:36:25.088Z'
  },
  token: 'eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6NCwiaWF0IjoNjU0NDQzMzg1LCJleHAiOjE2NTQ0ODY1ODV9.pJfaItYvTpHVSTh4wvZNayB2cquCcEGrG63UodSsgxs'
}
```

Attempting to set next request to whoami

```
L whoami
POST http://localhost:8080/traveler/whoami [200 OK, 1.02kB, 12ms]
✓ Status code is 200

{
  id: 4,
  firstname: 'johnny',
  lastname: 'cash',
  email: 'helloworld@test.com',
  createdAt: '2022-06-05T15:36:25.088Z',
  updatedAt: '2022-06-05T15:36:25.088Z'
}
```

Attempting to set next request to get_travelers

```
L get_travelers
GET http://localhost:8080/traveler [200 OK, 1.48kB, 6ms]
✓ Status code is 200

[
  {
    id: 1,
    firstname: 'John',
    lastname: 'Doe',
    email: 'john@doe.com',
    createdAt: '2022-06-05T15:35:58.552Z',
    updatedAt: '2022-06-05T15:35:58.552Z'
  },
  {
    id: 2,
    firstname: 'Dora',
    lastname: 'L\'exploratrice',
    email: 'dora@mapadora.fr',
    createdAt: '2022-06-05T15:35:58.628Z',
    updatedAt: '2022-06-05T15:35:58.628Z'
  },
  {
    id: 3,
    firstname: 'Jean',
    lastname: 'Dupont',
    email: 'jean@dupont.fr',
    createdAt: '2022-06-05T15:35:58.707Z',
    updatedAt: '2022-06-05T15:35:58.707Z'
  },
  {
    id: 4,
    firstname: 'johnny',
    lastname: 'cash',
    email: 'helloworld@test.com',
    createdAt: '2022-06-05T15:36:25.088Z',
    updatedAt: '2022-06-05T15:36:25.088Z'
  }
]
```

- Observer les résultats des tests.

```
Windows PowerShell X + v

L

Attempting to set next request to delete_traveler

□ Traveler
  L delete_traveler
    DELETE http://localhost:8080/traveler/4 [200 OK, 879B, 9ms]
    ✓ Status code is 200
    [
      'Object deleted'
      '***** COLLECTION VARIABLES RESET *****'
    ]



|                    | executed | failed |
|--------------------|----------|--------|
| iterations         | 1        | 0      |
| requests           | 73       | 0      |
| test-scripts       | 150      | 0      |
| prerequest-scripts | 81       | 0      |
| assertions         | 73       | 0      |



total run duration: 8.2s
total data received: 16.84kB (approx)
average response time: 17ms [min: 5ms, max: 248ms, s.d.: 31ms]

PS C:\dev\uni\acrobatt_api\newman_folder>
```

INTEGRATION CONTINUE

Un aspect intéressant de notre projet est l'intégration continue. Nous avons paramétré sur notre environnement de production (VPS avec Debian installé) un Gitlab runner, et créé un fichier « .gitlab-ci.yml » pour mettre automatiquement en production notre branche master lorsqu'une merge request est validée sur celle-ci.

En ajoutant la commande Newman, qui pointant sur le dossier de l'API et sur la collection Postman, nous avons la possibilité de réaliser des tests facilement.

```
stages:
  - build
  - test
  - deploy

cache:
  paths:
    - node_modules/

install_dependencies:
  stage: build
  script:
    - echo "running npm install"
    - pwd
    - node -v
    - npm install
  artifacts:
    paths:
      - node_modules/
  only:
    - develop

test:
  stage: test
  script:
    - newman run https://www.getpostman.com/collections/efaae735f77861799640 -e
      /localhost.postman_environment.json

deploy:
  stage: deploy
  script:
    - cp ../.env .env 2> /dev/null || true
    - cp ../config.json ./config/config.json 2> /dev/null || true
    - curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.39.1/install.sh |
      bash
    - ". ~/.nvm/nvm.sh"
    - nvm use 16.14.2
    - pm2 delete acrobatt_api 2> /dev/null || true
    - pm2 start app.js --name acrobatt_api
    - sleep 5
  only:
    - develop
    - master
```

RESSOURCES

Le code de nos tests est disponible dans le fichier « 05062022-Acrobatt Collection.postman_collection.json », export de notre collection.

Pour accéder à notre collection Postman, veuillez suivre le lien suivant :

(sous réserve de validité)

<https://go.postman.co/workspace/Les-aventuriers-archi-perdus~ecb75991-3026-42b8-bf40-b1d696a4bcef/collection/14350605-0e0e8f5c-4a8f-4a81-8620-afb5d8b0fbd8?action=share&creator=14605222>

PARTIE 2 – IHM WEB

INTRODUCTION

Pour donner suite aux différents cours portant sur les outils de tests, nous avons mis en place quelques tests réalisés avec l'outil Cypress. Ce dernier permet de tester l'affichage d'éléments via des sélecteurs particuliers, mais aussi de réaliser des tests unitaires ou des tests d'API via des requêtes (comme Postman).

Nous nous sommes limités à des tests d'IHM, bien que nous ayons écrit quelques requêtes comme celle qui permet de se logger pour ne pas passer systématiquement par l'interface.

TESTS MIS EN PLACE

Nous avons testé trois fonctionnalités, bien que l'application présente de nombreux éléments à tester : la création de compte, le login d'un utilisateur et l'ajout d'un participant à un voyage.

La progression des cours nous a amenés à affiner les patterns des tests, ce qui nous a également permis de faire évoluer l'organisation des fichiers. Le refactoring des tests ayant pris un certain temps, nous nous sommes concentrés sur ces trois fonctionnalités en tentant d'appliquer les bonnes pratiques mentionnées notamment dans la documentation de Cypress.

PRÉ-REQUIS

Nous avons créé plusieurs factories ainsi que des seeders pour remplir la base de données avec des éléments. En effet, les différents tests nécessitaient d'avoir des données en base ou des factories d'objet (notamment dans le cadre du page-object pattern) :

- Pour le test de la création de compte : une factory qui crée un utilisateurs (voir dans le dossier Model pour les objets)
- Pour le test du login : un objet Credentials (login/password) ainsi que l'utilisateur existant en base.
- Pour le test de l'ajout d'un participant ne possédant pas de compte sur l'application : deux utilisateurs et les Credentials associés, l'un ayant déjà un compte et l'autre non. Le premier utilisateur a un déjà créé un voyage qui apparaît dans sa liste de voyages.

PATTERN

Les trois fonctionnalités testées font appel à plusieurs écrans. Chaque écran correspond à une classe page-object. On trouve dans le dossier page-objects du projet :

- HomePage : c'est la racine du site
- ParticipantPage : c'est la page à l'intérieur du dashboard utilisateur qui permet de consulter les participants à un voyage et d'en inviter des nouveaux.
- ProfilEditionPage : c'est la page qui permet d'afficher les informations de compte de l'utilisateur
- SigninPage : la page de connexion
- SignupPage : la page de création de compte

Chaque page comprend les méthodes et un test qui permet de confirmer l'affichage attendu. En parallèle de cela, des sélecteurs particuliers `_[data-cy = ""]_` ont été définis dans les balises des éléments à tester. Cela facilite la sélection de l'objet pour pouvoir ensuite écrire les assertions.

Le dossier `_api_` contient les appels à l'API avec la fonction Cypress `cy.request` qui permet de faire une requête sans passer par l'interface. Ces appels sont utilisés lors du test d'ajout de participant.

Le dossier `_integration_` comprend les tests des fonctionnalités `signin`, `signup` et `addParticipant` faisant appel aux `PageObject`.

SCÉNARIOS DE TEST

.Signup

L'utilisateur entre sur la page d'accueil du site et clique sur "Créer un compte". Il entre ensuite ses informations. La redirection sur la page Signin indique que le compte a bien été créé.

.Signin

Un utilisateur déjà enregistré en base tente de se connecter. Il clique sur "Me Connecter", entre ses informations. Une redirection sur le dashboard utilisateur confirme le test.

.addParticipant

Un utilisateur possédant déjà un compte et ayant déjà créé un voyage est connecté à son compte (requête POST via cy.request). Il accède à la page des participants depuis le dashboard.

Dans un premier temps, il invite un participant sans adresse mail, ce qui ajoute directement le nom de la personne dans la liste des participants. Cet ajout n'a aucune autre implication. L'apparition du nom du participant dans la liste "Participants ajoutés" confirme cet ajout. Le participant est ensuite supprimé de la liste. Sa disparition de la liste confirme le test.

L'administrateur du voyage ajoute ensuite un participant avec une adresse mail. Cette personne n'a pas encore de compte sur l'application. De la même manière que précédemment, il rentre les informations nom + email dans le formulaire et clique sur le bouton ajouter. L'apparition du nom et du mail de cette personne dans "Participants en attente" confirme qu'une invitation a été envoyée. Ce participant se déconnecte de l'application.

La personne ajoutée crée un compte (via une requête POST comme précédemment) avec le même email que celui renseigné par l'administrateur du voyage. Le nouvel utilisateur arrive ensuite sur son dashboard, et doit constater que l'onglet notification a un badge avec les nombres de notifications arrivées (un petit 1 ici). Cet élément confirme une première partie de test. L'utilisateur clique sur cet onglet notification et doit constater qu'il est invité à rejoindre le voyage de l'administrateur. Il clique sur accepter l'invitation et le voyage s'ajoute à sa liste de voyages. On rebascule ensuite sur le compte de l'administrateur du système pour constater dans la page participants que l'utilisateur invité est passé dans la catégorie des participants au voyage.

DIFFICULTÉS ET RETOURS D'EXPÉRIENCE

Nous avons pu nous rendre compte que les tests d'IHM sont assez chronophages et sont très dépendants de la base de données, le lancement de l'API et du client web, ainsi que des performances du poste (temps de chargement des pages etc.).

Cet outil permet cependant de faire des tests de scénarios utilisateurs complet et de se rendre compte de la manière dont s'affichent les composants. L'un des avantages de Cypress est de pouvoir proposer en un seul outil plusieurs catégories de tests : interface, api, unitaires, etc. et de les intégrer au pipeline de déploiement afin de les automatiser.