|  |
| --- |
| Marion Labbé Paul Dampfhoeffer Stéphane Litt William Trahay  2021/2022 |

|  |
| --- |
| 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 |

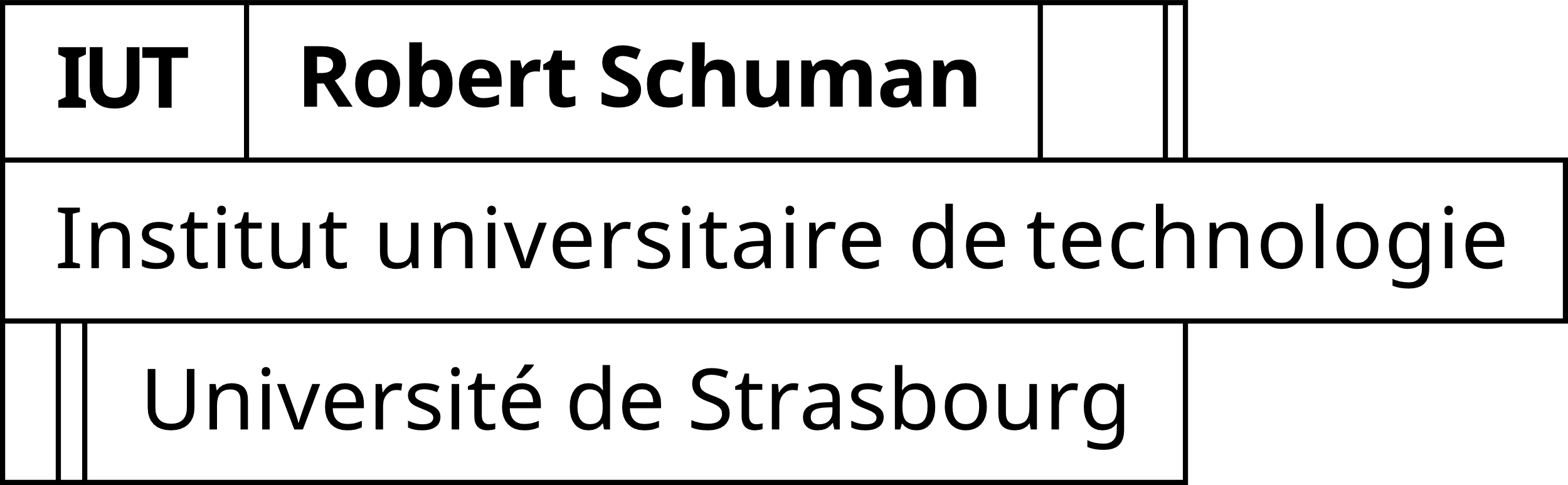


Table des matières

[PARTIE 1 – API 2](#_Toc106283139)

[POSTMAN 2](#_Toc106283140)

[NEWMAN 3](#_Toc106283141)

[SEEDERS 7](#_Toc106283142)

[PROCESSUS 9](#_Toc106283143)

[INTEGRATION CONTINUE 11](#_Toc106283144)

[RESSOURCES 13](#_Toc106283145)

[PARTIE 2 – IHM Web 13](#_Toc106283146)

[INTRODUCTION 13](#_Toc106283147)

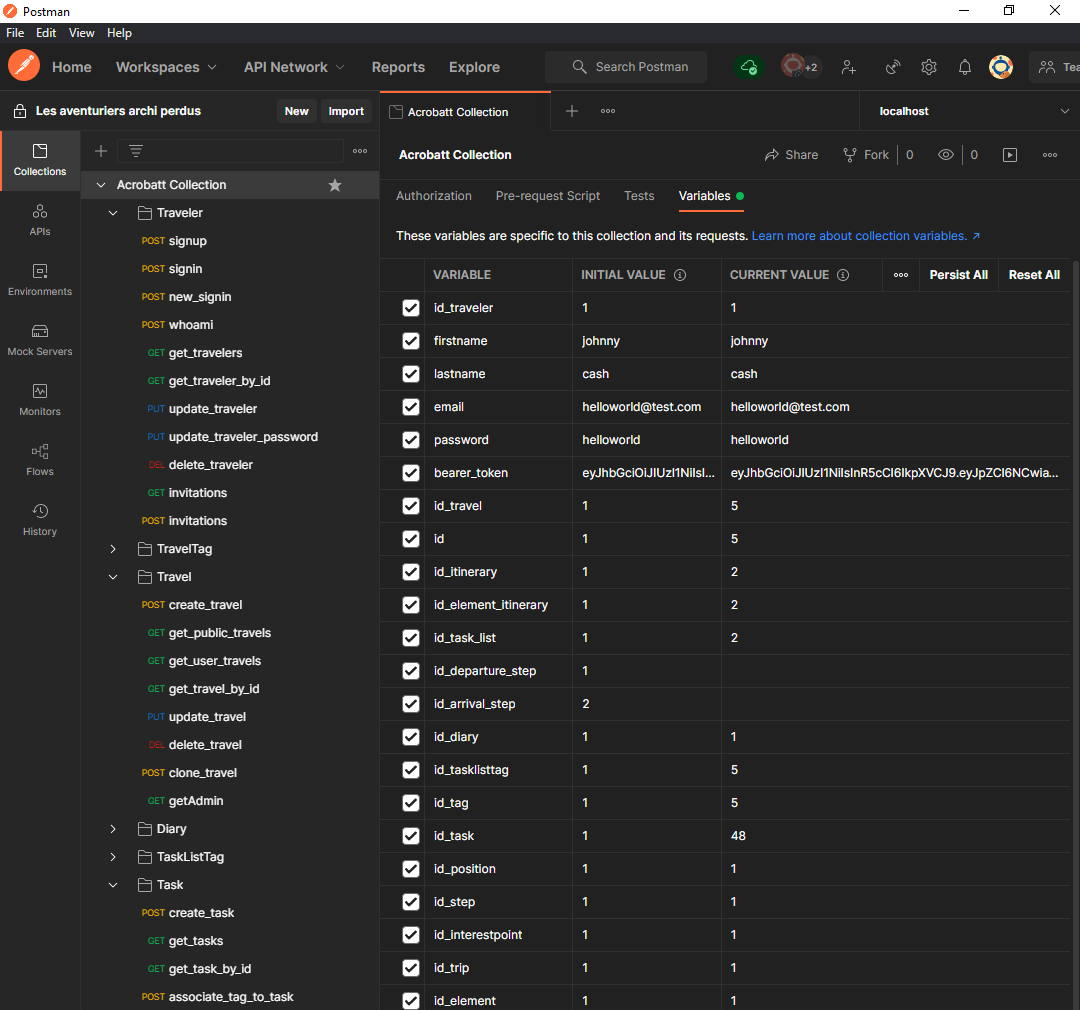
[TESTS MIS EN PLACE 13](#_Toc106283148)

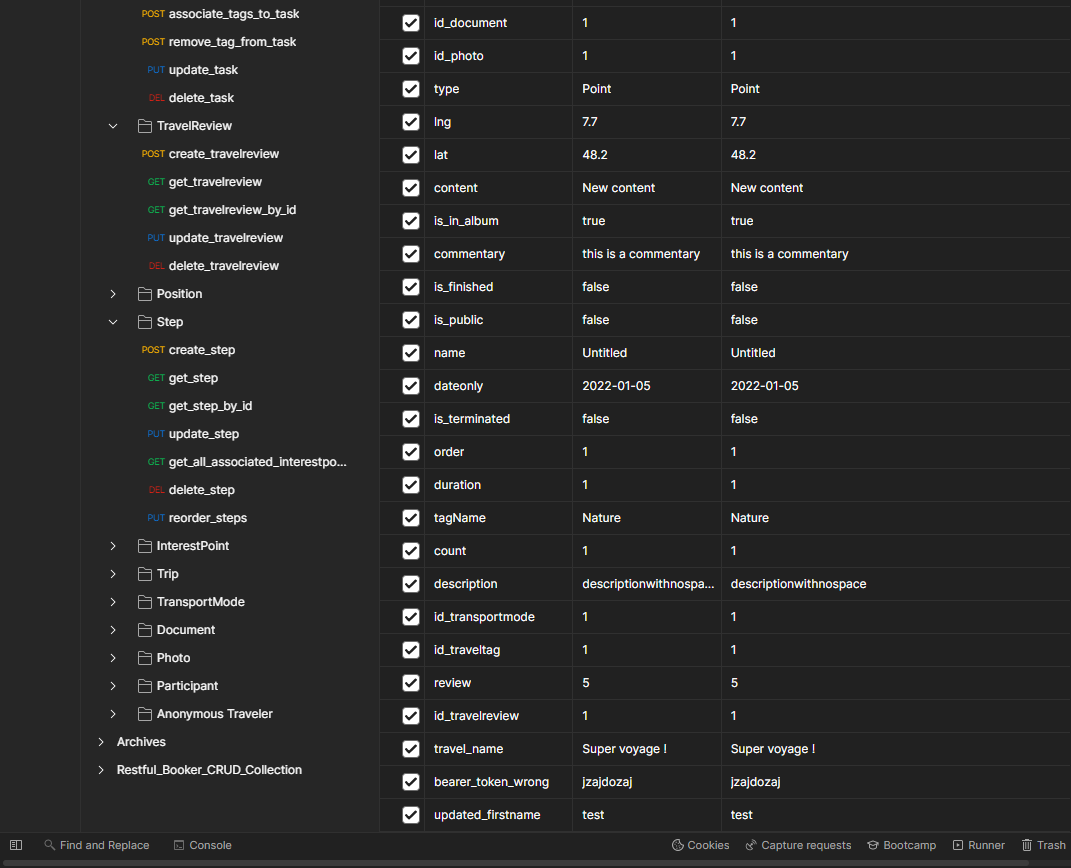
[DIFFICULTÉS ET RETOURS D’EXPÉRIENCE 16](#_Toc106283149)

## 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.





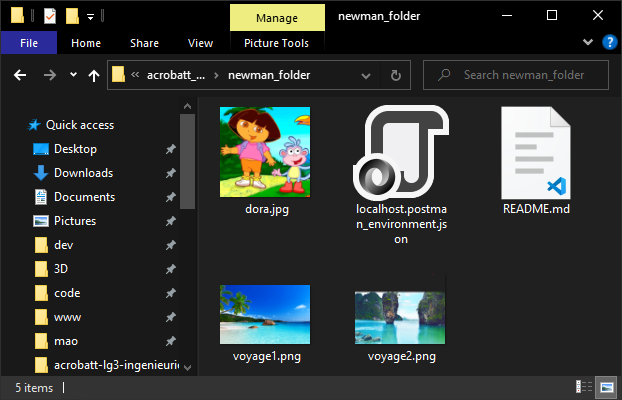
### NEWMAN

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.

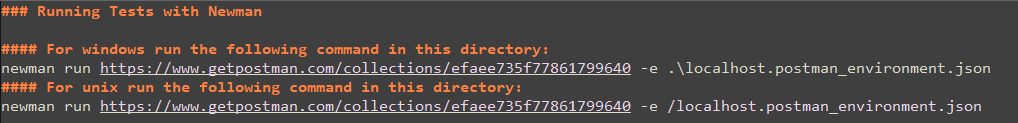
Une image contenant texte, capture d’écran, moniteur, noir

Description générée automatiquement



En effet, une des problématiques rencontrées fut l’exécution de tests sur une route prenant en entré 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 :



Fichier d’environnement

Une image contenant texte

Description générée automatiquement

Une autre difficulté a été l’exécution multiple de requêtes, nécessaires pour créer suffisamment de données exploitables pour les prochaines exécutés.  
(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.  
  
Une image contenant texte

Description générée automatiquement

### 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 donné au préalable.

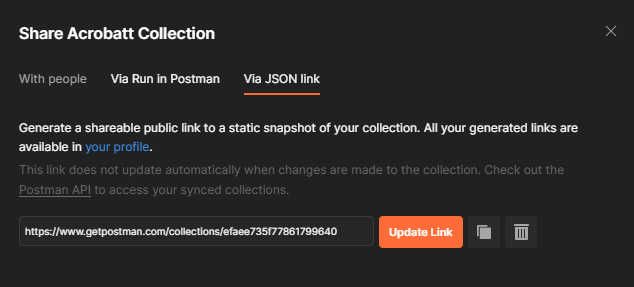
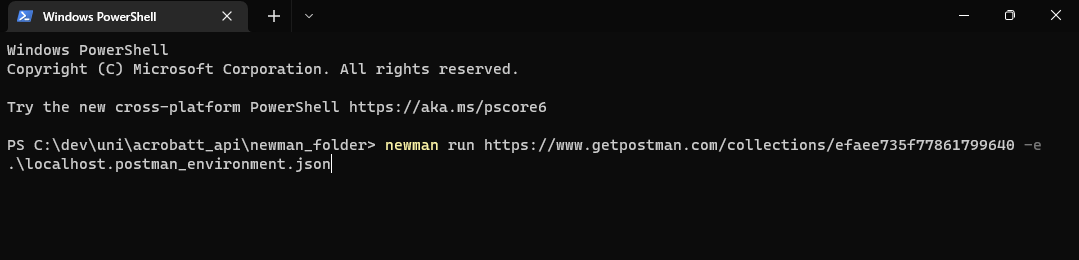
Une image contenant texte

Description générée automatiquement



### PROCESSUS

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

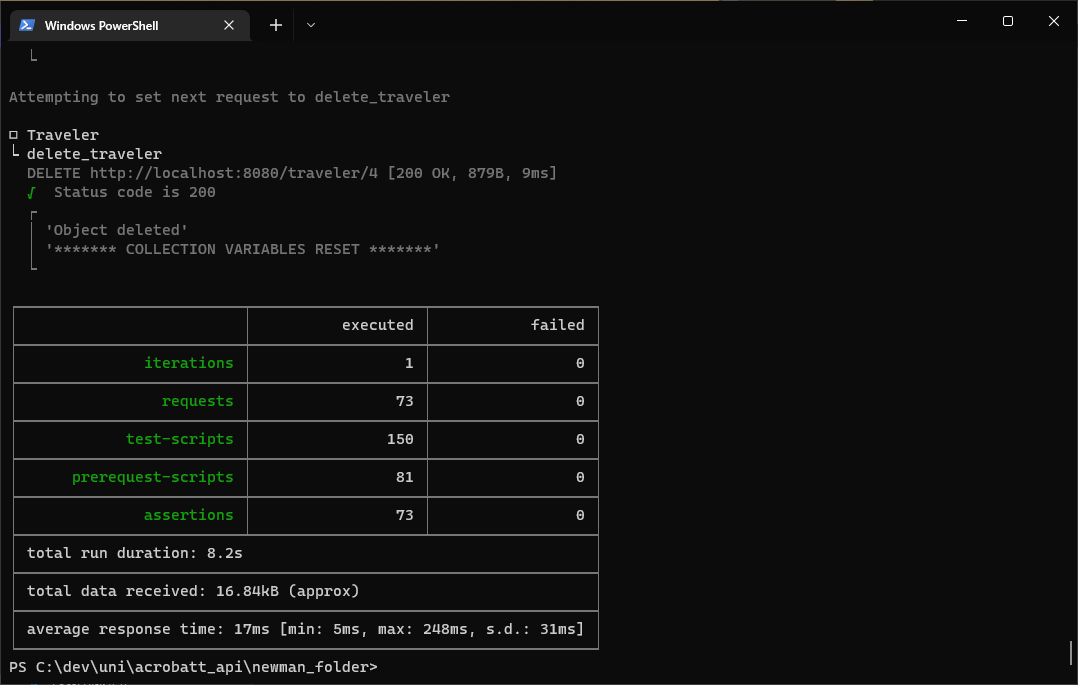
* S’assurer que la collection est à jour
* Préparer la commande en utilisant le lien vers la collection et en se positionnant dans le dossier newman\_folder.  
  
* Exécuter la commande et consulter les logs.  
  Une image contenant texte

  Description générée automatiquement

Une image contenant texte

Description générée automatiquement

* Observer les résultats des tests.



### 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/efaee735f77861799640 -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 Cypess. 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 créé 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 formualaire et clique sur le boutton 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 créé 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.