

GIT GITHUB TESTING

Table des matières

Liste des compétences travaillées dans le module.....	2
1 Les différents types de test :	3
1.1 Introduction	3
1.2 Test end to end :	3
1.2.1 Exemple	3
1.2.2 Outils et environnement de test (End to End).....	5
1.2.3 Structure du projet :	6
1.2.4 Environnement de test :	6
1.2.5 Créer un test end to end avec Cypress	7
1.2.6 Créer un test end to end avancé avec Cypress :	10
1.2.7 Exercices pratiques test end to end Cypress :	11
1.2.8 Tests Cypress automatique :	12
2 Intégration Cypress GitHub Action :	14
2.1 Définition Workflow CI/CD :	14
2.2 Environnement GITHUB ACTIONS :	14
2.2.1 Configuration du repository GitHub :	15
2.2.2 intégrer les tests Cypress à Github Actions :	16
2.2.3 Exercices pratiques test end to end Cypress / GitHub Actions :	16

Auteur :

Mathieu MITHRIDATE

Date création :

02 / 05 / 2023

Relu, validé & visé par :

☒ Jérôme CHRETIENNE
☒ Sophie POULAKOS
☒ Mathieu PARIS

Date révision :

xx / xx / 20xx



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

GIT GITHUB TESTING

Liste des compétences travaillées dans le module :

Etre capable de comprendre les enjeux liés au testing

Etre capable de réaliser des tests unitaires

Etre capable de concevoir et d'exécuter des scénarios de tests end-to-end

Etre capable de tester les appels à une API

Etre capable d'utiliser un Framework de testing

Auteur :

Mathieu MITHRIDATE

Date création :

02 / 05 / 2023

Relu, validé & visé par :

☒ Jérôme CHRETIENNE
☒ Sophie POULAKOS
☒ Mathieu PARIS

Date révision :

xx / xx / 20xx



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

GIT GITHUB TESTING

1 Les différents types de test :

1.1 Introduction :

Dans cette partie nous allons voir les différents types de test, la théorie et les outils utilisés.

Afin de tester une logiciel, site web ou application, nous devons mettre en place des tests à chaque étape du développement.

Les tests sont un enjeu majeur dans un développement applicatif, ils nous permettrons de livrer un produit qui sera le plus stable et sécurisé possible.

1.2 Test end to end :

Les tests end to end, en français test de bout en bout (e2e), sont une approche qui permet d'évaluer le fonctionnement d'un produit (logiciel, site web, application etc...). Ces tests permettent de vérifier si tous les composants du projet sont capables de fonctionner de manière optimale dans des situations réelles.

C'est quoi un test end to end ?

Les tests end to end sont une méthodologie qui évalue l'ordre de travail d'un produit complexe, du début jusqu'à la fin. Ce processus consiste à vérifier que tous les composants du projet fonctionnent de manière optimale dans des situations de production réelles.

1.2.1 Exemple :

Dans un projet d'un site internet, nous avons un formulaire d'inscription, celui-ci nous permet d'ajouter un utilisateur en base de données (méthode **addUser**). Pour nous aider à identifier ce que nous devons tester nous allons partir d'un **diagramme d'activité UML**.

Auteur :

Mathieu MITHRIDATE

Date création :

02 / 05 / 2023

Relu, validé & visé par :

☒ Jérôme CHRETIENNE
☒ Sophie POULAKOS
☒ Mathieu PARIS

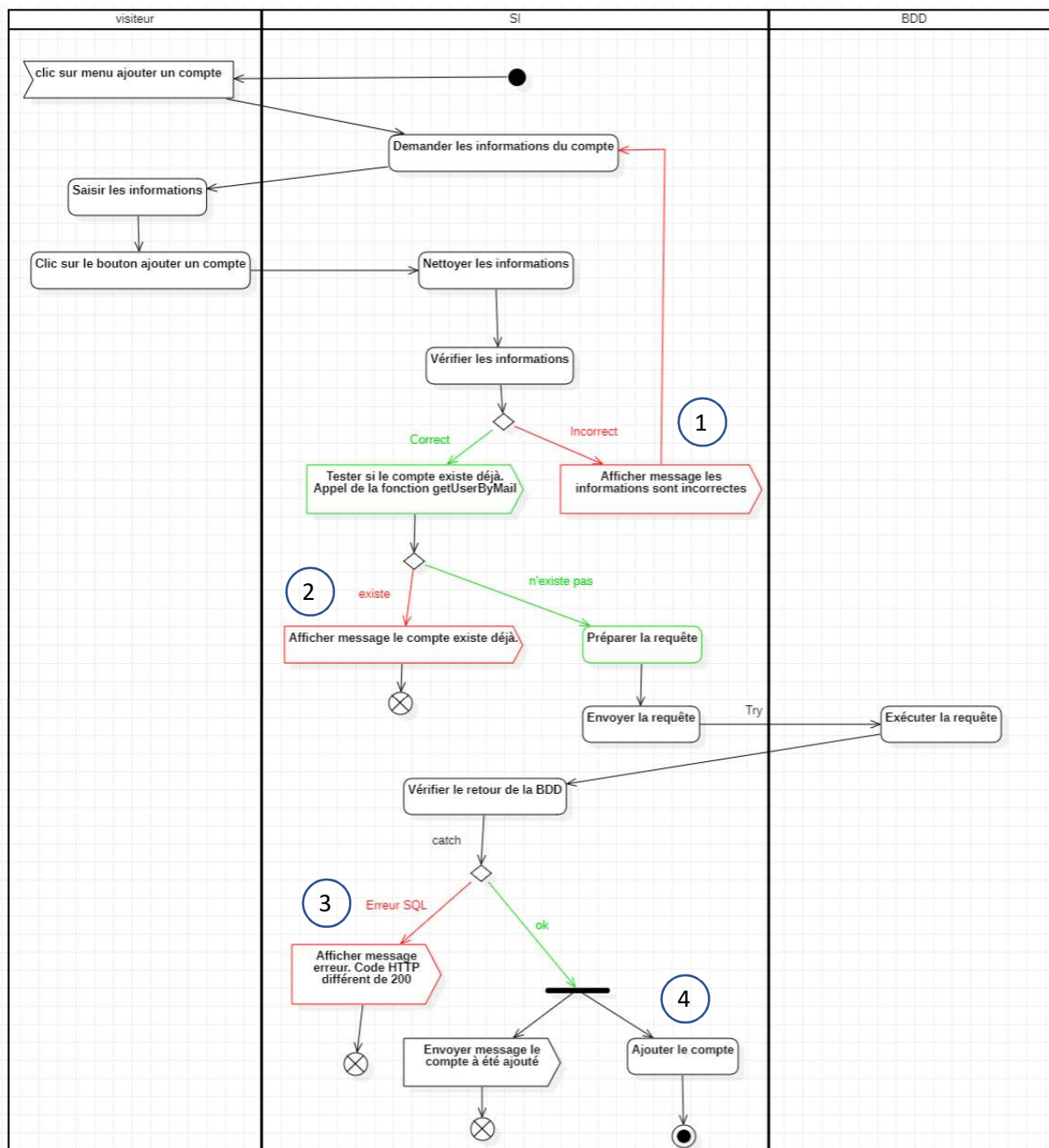
Date révision :

xx / xx / 20xx



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

GIT GITHUB TESTING



Auteur :

Mathieu MITHRIDATE

Date création :

02 / 05 / 2023

Relu, validé & visé par :

☒ Jérôme CHRETIENNE
☒ Sophie POULAKOS
☒ Mathieu PARIS

Date révision :

xx / xx / 20xx

GIT GITHUB TESTING

Pour pouvoir valider la méthode **addUser** nous devons tester les sorties suivantes :

- 1 Erreur les informations sont incorrectes (code HTTP 400 Bad request),
- 2 Erreur le compte existe déjà (code HTTP 400 Bad request),
- 3 Erreur d'enregistrement BDD (code HTTP 500 erreur interne serveur),
- 4 Le compte a été ajouté en BDD (code HTTP 200 Ok).

1.2.2 Outils et environnement de test (End to End) :

Le projet sera développé avec les langages, technologies, outils suivants :

- HTML, CSS, JavaScript (front-end),
- PHP (back-end),
- SQL (Base de données),
- Serveur web apache,
- Serveur Base de données (MariaDB),
- Node JS (npm gestionnaire de package),
- Cypress (Framework de test End To End),
- Navigateur web Chrome,
- Gestionnaire de version GIT (repository GITHUB),

Auteur :

Mathieu MITHRIDATE

Date création :

02 / 05 / 2023

Relu, validé & visé par :

- ☒ Jérôme CHRETIENNE
- ☒ Sophie POULAKOS
- ☒ Mathieu PARIS

Date révision :

xx / xx / 20xx



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

GIT GITHUB TESTING

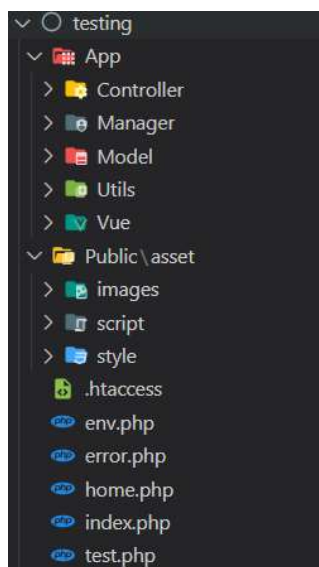
1.2.3 Structure du projet :

Le projet sera développé en structure MVC (**Model, Vue, Controller**).

Pour vous aider à construire le projet vous pouvez utiliser le **script bash** suivant :

<https://github.com/mithridatem/script>

Voir fichier **readme** pour exécuter le script bash



1.2.4 Environnement de test :

Pour exécuter nos tests end to end nous allons utiliser Node JS (npm) et le Framework Cypress.

Installation de Node JS :

<https://nodejs.org/en/download>

Initialisation de Node JS dans le projet :

Ouvrir un terminal dans le projet (**racine du projet**) et exécuter la ligne de commande suivante :

```
npm init -y
```

Installation de l'environnement de test Cypress :

Exécuter dans le terminal la ligne de commande suivante (cela va télécharger et configurer Cypress) :

```
npm install -D cypress
```

Auteur :

Mathieu MITHRIDATE

Date création :

02 / 05 / 2023

Relu, validé & visé par :

☒ Jérôme CHRETIENNE
☒ Sophie POULAKOS
☒ Mathieu PARIS

Date révision :

xx / xx / 20xx



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

GIT GITHUB TESTING

1.2.5 Créer un test end to end avec Cypress :

Lancer l'interface de test :

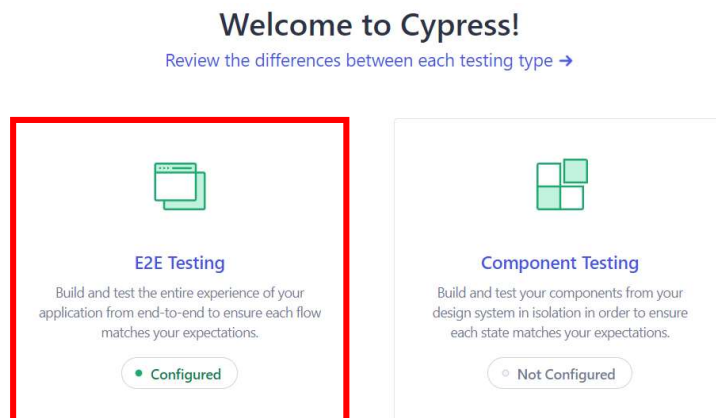
1 Nous allons éditer le fichier **package.json** à la racine du projet et remplacer le contenu de **test** par :

```
"scripts": {  
  "test": "cypress open"  
}
```

2 Pour lancer ce test (il va lancer l'interface de **Cypress**) saisir la commande suivante :

```
npm run test
```

3 Au lancement **Cypress** nous demande ce que l'on souhaite effectuer comme type de test, nous allons choisir : **E2E Testing**.



Auteur :

Mathieu MITHRIDATE

Date création :

02 / 05 / 2023

Relu, validé & visé par :

☒ Jérôme CHRETIENNE
☒ Sophie POULAKOS
☒ Mathieu PARIS

Date révision :

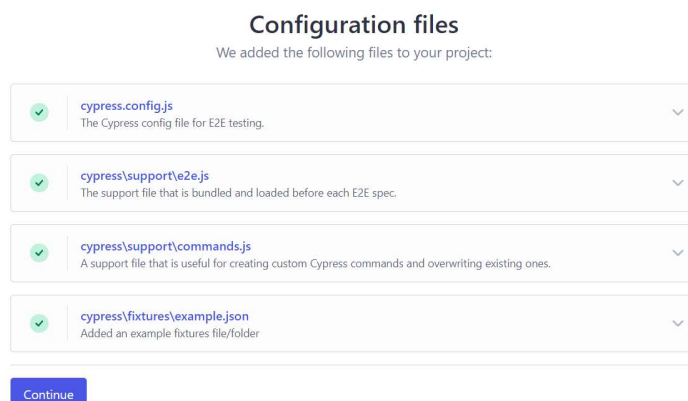
xx / xx / 20xx



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

GIT GITHUB TESTING

4 Puis on clique sur **Continue**



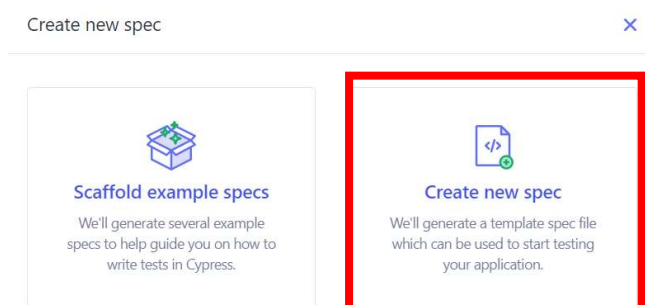
Cypress va nous ajouter un répertoire cypress dans le projet qui va contenir les éléments suivants :

Fixtures (des jeux de données factices, comme avec Symfony)

Support (pour des commandes et modules supplémentaires)

e2e (qui va contenir nos différents tests end to end)

5 Nous allons créer un nouveau **spec** (test) en cliquant sur **new spec** :



Auteur :

Mathieu MITHRIDATE

Date création :

02 / 05 / 2023

Relu, validé & visé par :

☒ Jérôme CHRETIENNE
☒ Sophie POULAKOS
☒ Mathieu PARIS

Date révision :

xx / xx / 20xx






Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

GIT GITHUB TESTING

6 Nommer le test comme ci-dessous (**nom.cy.js**) :

Enter the path for your new spec ×

 cypress/e2e/nouveau_spec.cy.js

NB : le test se retrouvera dans le répertoire **cypress/e2e/**

L'outil va nous créer un **template de test** comme ci-dessous :

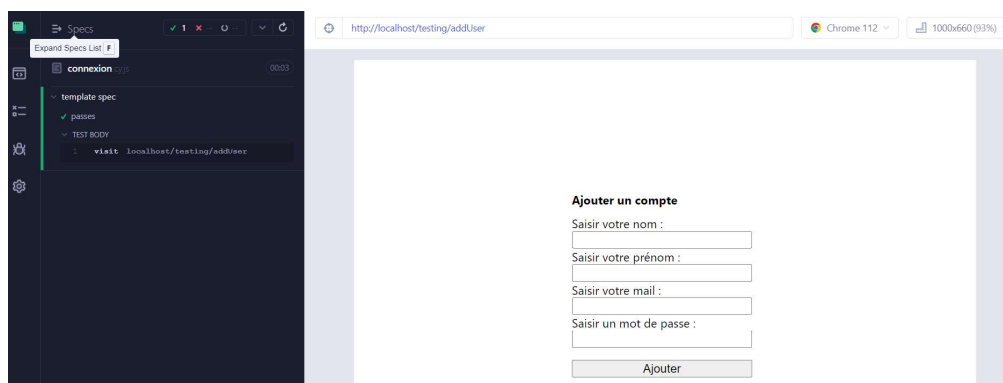
```
describe('template spec', () => {
  it('passes', () => {
    cy.visit('https://example.cypress.io')
  })
})
```

Ce test va lancer grâce à la fonction **visit** le site <https://example.cypress.io>.

7 Nous allons remplacer le site par :

<http://localhost/testing/addUser>

8 Vérifier si le formulaire d'inscription apparaît bien à l'écran (capture ci-dessous)



Auteur :

Mathieu MITHRIDATE

Date création :

02 / 05 / 2023

Relu, validé & visé par :

☒ Jérôme CHRETIENNE
☒ Sophie POULAKOS
☒ Mathieu PARIS

Date révision :

xx / xx / 20xx

GIT GITHUB TESTING

1.2.6 Créer un test end to end avancé avec Cypress :

Dans cette partie nous allons voir comment créer un test Cypress avancé. Celui-ci va nous permettre de vérifier le retour d'un formulaire d'inscription (gestion des erreurs) et enregistrer grâce à une api son exécution.

Cypress possède une **api** pour piloter un navigateur internet :

<https://docs.cypress.io/api/table-of-contents>

1 importer la base de projet :

Récupérer la base de projet depuis le repository ci-dessous :

https://github.com/mithridatem/test_cypress

2 Configurer **Cypress** dans le projet :

Ouvrir un terminal et saisir les commandes ci-dessous :

```
npm init -y
```

```
npm install -D cypress
```

3 Editer le fichier **package.json** comme ci-dessous :

```
"scripts": {  
  "test": "cypress open"  
},
```

4 Lancer **Cypress** :

```
npm run test
```

5 Créer un test **end to end** :

Comme vu plus haut dans ce support cliquer sur **new spec** Cypress va créer un nouveau fichier de test dans le répertoire **cypress/e2e/** à la racine de votre projet.

6 Description du test partie 1 :

Le test va effectuer les actions suivantes :

- Ouvrir la page du site <http://localhost/testing/addUser> (formulaire d'ajout de compte)
- Remplir les champs mail et password. (Avec un mail et un mot de passe)
- Cliquer sur le bouton submit.

Auteur :

Mathieu MITHRIDATE

Date création :

02 / 05 / 2023

Relu, validé & visé par :

- ☒ Jérôme CHRETIENNE
- ☒ Sophie POULAKOS
- ☒ Mathieu PARIS

Date révision :

xx / xx / 20xx



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

GIT GITHUB TESTING

Le test s'écrit dans la méthode **it** dans un **callback** (fonction).

Vous allez utiliser les fonctions **visit** (lancer une url d'un site), **get** (sélecteur, elle fonctionne comme **querySelector** en JS), **click** (cliquer sur un élément HTML), **type** (saisir du texte dans un **input HTML**)

7 Lancer votre test et vérifier le résultat :

Pour lancer le test le choisir dans la liste des **specs** en cliquant dessus, **Cypress** va l'exécuter automatiquement.

Si le test est exécuté vous devriez avoir un affichage comme ci-dessous :

```
nouveau_spec.cypress.js

Mes tests
  ✓ formulaire
  TEST BODY
    1 visit localhost/testing/addUser
    2 get :input[name='mail']
    3 -type mathieu@test.com
    4 get :input[name='mdp']
    5 -type 1234
    6 get :input[name='submit']
    7 -click
      (form sub) --submitting form--
      (page load) --page loaded--
```

Nous pouvons voir dans la fenêtre de résultat toutes les fonctions qui ont été ajouté au test et leur exécution.

NB : Nous avons la possibilité de lancer plusieurs tests les uns après les autres en ajoutant d'autre méthode **it** dans la fonction **describe**.

1.2.7 Exercices pratiques test end to end Cypress :

1 Description du test n°1 à réaliser :

Depuis la base de projet, ajouter un test qui va réaliser les actions suivantes :

- Se connecter sur la page <http://localhost/testing/addUser>
- Récupérer les 4 champs inputs (nom, prénom, mail, mot de passe),
- Compléter les informations dans les 4 champs inputs pour créer un compte,
- Vérifier que le message d'ajout est bien : **Le compte a été ajouté en BDD**

2 Créer le test qui correspond à la consigne.

Auteur :

Mathieu MITHRIDATE

Date création :

02 / 05 / 2023

Relu, validé & visé par :

- ☑ Jérôme CHRETIENNE
- ☑ Sophie POULAKOS
- ☑ Mathieu PARIS

Date révision :

xx / xx / 20xx



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

GIT GITHUB TESTING

3 Description du test n°2 à réaliser :

Depuis la base de projet, ajouter un test qui va réaliser les actions suivantes :

- Se connecter sur la page <http://localhost/testing/addUser>
- Récupérer les 4 champs inputs (nom, prénom, mail, mot de passe),
- Compléter les informations dans les 4 champs inputs pour créer un compte,
- Vérifier que le message d'ajout est bien : **Le compte a été ajouté en BDD**,
- Faire une requête API (**request**) sur l'url **http://localhost/testing/api/addTest** , pour ajouter un enregistrement BDD du test effectué (*nom du test, date, son état : valid 1 **passé**, valid 0 **échoué***)

4 Créer le test qui correspond à la consigne test n°2.

5 Réaliser un nouveau test sur la fonctionnalité de votre choix.

1.2.8 Tests Cypress automatique :

Cypress permet également de lancer les tests de façon totalement automatique en utilisant la commande suivante :

```
./node_modules/.bin/cypress run --browser chrome
```

Cette commande va lancer **cypress** dans le terminal et afficher dans celui-ci un compte rendu :

```
Running: connexion.cy.js (1 of 2)

Mes tests
  ✓ connexion (2941ms)

1 passing (5s)

(Results)

Tests:      1
Passing:    1
Failing:    0
Pending:    0
Skipped:    0
Screenshots: 0
Video:      true
Duration:   4 seconds
Spec Ran:   connexion.cy.js
```

A noter une fonction très intéressante de Cypress, il va vous enregistrer des vidéos au format mp4 dans le répertoire **cypress/videos** à la racine de votre projet.

Ajouter la ligne : **video: true**, dans le fichier **cypress.config.js**

Auteur :

Mathieu MITHRIDATE

Date création :

02 / 05 / 2023

Relu, validé & visé par :

☒ Jérôme CHRETIENNE
☒ Sophie POULAKOS
☒ Mathieu PARIS

Date révision :

xx / xx / 20xx



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

GIT GITHUB TESTING

Nous pouvons également customiser le fichier package.json à la racine du projet en ajoutant la ligne suivante :

```
"scripts": {  
  "test": "cypress open",  
  "auto_test": "cypress run --browser chrome"  
},
```

Nous pourrions lancer l'ensemble de nos tests en saisissant dans le terminal la commande :
`npm run auto_test`

Auteur :

Mathieu MITHRIDATE

Date création :

02 / 05 / 2023

Relu, validé & visé par :

☒ Jérôme CHRETIENNE
☒ Sophie POULAKOS
☒ Mathieu PARIS

Date révision :

xx / xx / 20xx



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

GIT GITHUB TESTING

2 Intégration Cypress GitHub Action :

Dans cette partie du module nous allons voir comment utiliser de façon combiné **Cypress** et **GitHub**. En ajoutant un **workflow** qui lance nos tests à chaque **pull** sur le **repository**.

2.1 Définition Workflow CI/CD :

CI/CD c'est quoi ?

CI/CD est un terme générique couvrant plusieurs phases **DevOps**. **CI (intégration continue)** est la pratique consistant à intégrer des modifications de code dans un repository. **CD (déploiement continu)** : Déploiement continu publie automatiquement les nouvelles versions.

Effectuer des tests **CI/CD** réduit les **erreurs** de code et les **problèmes**. Cela nous permet d'intégrer les modifications ou ajouts de code dans un **repository** à la condition que ceux-ci passent les **tests** mis en place (si les tests sont validés, on autorise le push des mises à jour).

2.2 Environnement GITHUB ACTIONS :

A la manière d'un **GitLab**, **GitHub** a intégré des automatisations pilotés par les **actions** sur un **repository**.

Exemple : Workflow nouvelle fonctionnalité (ajouter un article)

- 1 Coder la fonctionnalité,
- 2 Etablir les tests qui vont permettre de vérifier si la fonctionnalité est opérationnelle.
- 3 Ajouter la nouvelle fonction au repository (add et commit)
- 4 lancement du plan de test au moment du push sur le repository distant. **GitHub** va monter un container **docker (Ubuntu + cypress)** et va exécuter le contenu du fichier **package.json** (généré grâce à la commande **npm init**) dans lequel nous avons défini des actions à réaliser (**tests**).

Auteur :

Mathieu MITHRIDATE

Date création :

02 / 05 / 2023

Relu, validé & visé par :

☒ Jérôme CHRETIENNE
☒ Sophie POULAKOS
☒ Mathieu PARIS

Date révision :

xx / xx / 20xx



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

GIT GITHUB TESTING

2.2.1 Configuration du repository GitHub :

Nous allons nous rendre sur le repository **GitHub** et ajouter dans l'onglet actions un fichier de workflow **Cypress**.

Get started with GitHub Actions

Build, test, and deploy your code. Make code reviews, branch management, and issue triaging work the way you want. Select a workflow to get started.

Skip this and [set up a workflow yourself](#) →

Nous allons créer un fichier **main.yml** dans le répertoire : **.github/workflows** (à la racine du **repository**) comme ci-dessous :

```
name: End-to-end tests
on: push
jobs:
  cypress-run:
    runs-on: ubuntu-22.04
    steps:
      - name: Checkout
        uses: actions/checkout@v3
      # Install npm dependencies, cache them correctly
      # and run all Cypress tests
      - name: Cypress run
        uses: cypress-io/github-action@v5
```

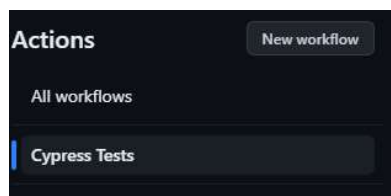
Après avoir créé le fichier, il faut le commit (site **github**)

Source de fichier de configuration, le site suivant nous donne des templates de configuration :

<https://github.com/cypress-io/github-action>

NB : il faut faire attention aux tabulations (c'est un fichier **YML**)

Nous allons nous retrouver avec un nouveau workflow comme ci-dessous :



A chaque push il va s'exécuter -> lancer une VM Ubuntu et les actions définies. (main.yml)

Auteur :

Mathieu MITHRIDATE

Date création :

02 / 05 / 2023

Relu, validé & visé par :

☒ Jérôme CHRETIENNE
☒ Sophie POULAKOS
☒ Mathieu PARIS

Date révision :

xx / xx / 20xx



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

GIT GITHUB TESTING

2.2.2 intégrer les tests Cypress à Github Actions :

Pour terminer la configuration nous allons écrire un test simple et vérifier ce qu'il se passe.

Ajouter un fichier dans cypress/e2e qui va contenir une simple commande (lancer une url) comme ci-dessous :

```
describe('Mes tests', () => {  
  it('lancer le serveur', () => {  
    cy.visit('https://google.com')  
  })  
})
```

Si tout se déroule comme prévu au moment du push, **GitHub** va exécuter le test écrit ci-dessus et afficher un rapport comme celui-ci :

```
Mes tests  
✓ lancer le serveur (1986ms)  
  
1 passing (4s)  
  
(Results)  
  
| Tests:      1  
| Passing:    1  
| Failing:    0  
| Pending:    0  
| Skipped:    0  
| Screenshots: 0  
| Video:      true  
| Duration:   4 seconds  
| Spec Ran:   test.cy.js
```

NB : Attention nous ne pouvons pas utiliser un site web qui tourne en local sur notre machine (ex : <http://localhost/test/index.php>) avec **GitHub Actions**. Si nous voulons l'utiliser nous allons devoir déployer le site en ligne. On peut par contre l'utiliser avec **cypress** en local (sans l'intégration **GitHub Actions**).

2.2.3 Exercices pratiques test end to end Cypress / Github Actions :

Rédiger les tests suivants :

-Tester l'ajout d'un utilisateur en BDD réussi.

-Tester l'ajout d'un utilisateur en BDD échec.

Ecrire les différentes étapes qui valide l'ajout (message enregistrement réussi), et l'échec (message enregistrement échec).

Auteur :

Mathieu MITHRIDATE

Date création :

02 / 05 / 2023

Relu, validé & visé par :

☒ Jérôme CHRETIENNE
☒ Sophie POULAKOS
☒ Mathieu PARIS

Date révision :

xx / xx / 20xx



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.