Plan de tests

Projet MATUI : un outil d'aide à la décision de méthode pour recueillir des données auprès d'utilisateurs.

PAULINE TEOULLE
CORENTIN ROY
UNIVERSITÉ GRENOBLE-ALPES

M1 WIC
2020-2021

Sommaire

Sommaire 2

1. Introduction 3

1.1. Objectifs et méthodes 3

1.2. Documents de référence 3

2. Guide de lecture 3

2.1. Maîtrise d'œuvre 3

2.2. Maîtrise d'ouvrage 4

3. Concepts de base 4

4. Tests fonctionnels de l'API 4

4.1. Identification 4

4.2. Description 5

4.3. Contraintes 5

4.4. Dépendances 6

4.5. Procédure de test 6

5. Tests fonctionnels React 14

5.1. Identification 14

5.2. Description 14

5.3. Dépendances 15

5.4. Procédure de test 15

6. Vérification de la documentation 18

7. Annexes 19

8. Glossaire 19

9. Références 20

10. Index 21

1. Introduction

Ce document sert à expliquer et lister les différents tests réalisés pour l'application MATUI.
L'objectif est de tester le code et de vérifier le comportement de celui-ci. Cela permet d'identifier les comportements à risques de l'application.
Les tests ont été réalisés sous Postman pour vérifier que l'API fonctionne correctement et Jest pour vérifier que les composants React construisent le DOM de la bonne façon.

Les tests sont disponibles sous le format JSON pour l'API et les snapshots SNAP pour React.

1.1. Objectifs et méthodes

L'objectif du projet est de développer une application web afin d'aider les chercheurs en informatique à choisir une méthode ou un outil qui les aidera à récolter des informations auprès des utilisateurs. L'objectif est de faciliter l'utilisation du diagramme MATUI et de le diffuser plus largement.

Le projet doit informatiser le diagramme MATUI sous deux aspects :

- Éditer le diagramme (Une personne peut ajouter ou supprimer une partie , une association de documents à une méthode ou un outil, etc ...)

- Utiliser le diagramme (Une personne peut l'utiliser pour accéder au résultat d'une méthode particulière et aux documents aidant à l'application de celle-ci).

Les tests nous permettent de vérifier que la modification de code (ajout d'une fonctionnalité) ne va pas impacter de manière négative les fonctionnalités déjà existantes. Toutes les fonctionnalités à tester sont listées et ont un scénario associé, ce qui permet de tester le bon comportement de l'application et éventuellement les failles du code.

Les tests de l'API ont été réalisés et exécutés grâce à Postman, qui permet de paramétrer un environnement de test et d'écrire des tests en JavaScript pour chaque requête.

Les tests React sont réalisés en Jest qui est un framework JavaScript et qui permet de créer de fausses données pour vérifier les rendus.

1.2. Documents de référence

Ce présent document a été élaboré grâce aux documents tels que le cahier des charges et le cahier de recette.

2. Guide de lecture

2.1. Maîtrise d’œuvre

Le personnel technique de la maîtrise d'œuvre doit avoir accès à l’ensemble du document. Ce document permet d’expliquer les différents tests réalisés, et de les lister pour une évolution future.

2.2. Maîtrise d’ouvrage

Les responsables de la maîtrise d’ouvrage sont Nadine MANDRAN et Sophie DUPUY. Elles ont besoin du document dans son entièreté. Ce document permet de garantir la conformité du produit vis-à-vis du cahier des charges et du cahier de recette.

3. Concepts de base

Les tests de l’API sont réalisés comme des tests fonctionnels qui permettent de tester les requêtes HTTP (GET, POST, PUT, DELETE). Chaque test d’API nécessite l’utilisation d’une base de données (il y a donc une dépendance). Il faut donc s’assurer d’avoir un état de base de données qui correspond à ce que l’on peut avoir en situation réelle.

Chaque test est divisé en plusieurs parties: la première est le test du code de la réponse HTTP, la seconde est le test du format de retour (ici JSON), la troisième est le test de la longueur du JSON renvoyé, et enfin, la dernière est le test des caractéristiques des éléments contenus dans le JSON (string, array…).

4. Tests fonctionnels de l’API

4.1. Identification

Numéro	Scénario
1	Créer un compte utilisateur
2	Se connecter
3	Récupérer les informations de MATUI
4	Enregistrer l’arbre MATUI
5	Consulter la liste de tous les membres.
6	Supprimer un membre.
7	Changer le rôle d’un membre.
8	Récupérer la liste de documents associé à une méthode

9	Lire le contenu de la page d'accueil
10	Modifier le contenu de la page d'accueil
11	Lire tous les documents
12	Supprimer un document
13	Lire le message de fin de questionnaire

4.2. Description

Scénario 1: Ce scénario est composé de 2 tests qui consistent à simuler la création d'un compte.

- 1 qui vérifie la création de compte valide (avec des informations correctes)
- 1 qui vérifie la création de compte invalide (avec des informations incorrectes)

Scénario 2: Ce scénario est composé de 2 tests qui consistent à simuler la connexion au site par un utilisateur :

- 1 qui vérifie la connexion avec des identifiants corrects
- 1 qui vérifie la connexion avec des identifiants incorrects

Scénario 3: Ce scénario comprend 1 test qui vérifie la récupération des informations de l'arbre MATUI sous un format JSON.

Scénario 4: Ce scénario comprend 1 test qui simule la suppression des données actuelles et qui insère les nouvelles.

Scénario 5: Ce scénario comprend 1 test qui vérifie la récupération des informations des comptes (id, pseudo, mail, rôle) sous un format JSON.

Scénario 6: Ce scénario comprend 1 test qui simule la suppression d'un compte.

Scénario 7: Ce scénario comprend 1 test qui simule la modification de rôle d'un compte.

Scénario 8: Ce scénario comprend 1 test qui simule la récupération des documents associés à une méthode.

Scénario 9: Ce scénario comprend 1 test qui simule la récupération du contenu de la page d'accueil.

Scénario 10: Ce scénario comprend 1 test qui simule la modification du contenu de la page d'accueil.

Scénario 11: Ce scénario comprend 1 test qui simule la récupération de tous les documents présents dans la base de données.

Scénario 12: Ce scénario comprend 2 tests qui simulent la suppression d'un document dans la base de données.

- 1 qui vérifie la suppression lorsque le document existe dans le répertoire
- 1 qui vérifie la suppression lorsque le document n'existe pas

Scénario 13: Ce scénario comprend 1 test qui simule la récupération du message de fin de questionnaire.

4.3. Contraintes

Une des contraintes est d'avoir une base de données utilisable et correspondant à une situation réelle. Il faut donc vérifier que les clés étrangères existent dans les tables utilisées.

L'environnement de test est local : http://{{domainname}}/Projet_TER/API/Controllers avec une variable d'environnement {{domainname}} qui correspond au serveur local (ici, localhost:80).

Le principe des tests d'API est le suivant : on vérifie le code réponse renvoyé, s'il y a bien un JSON renvoyé, si le corps du JSON est correct, si la structure du JSON est celle attendue, le nombre d'éléments composant le JSON...

4.4. Dépendances

Tests	Test à mener au préalable
1.1	
1.2	1.1
2.1	1.1
2.2	1.1
3	4
4	
5	1.1
6	1.1
7	1.1
8	
9	
10	9
11	
12	
13	

4.5. Procédure de test

Un exemple de test d'API est disponible en Annexe 1.

Scénario	Test	Données entrée	Résultats attendus	Critère de validation

1.1	Création valide de compte	{
		"mail":"superAdmin@superAdmin.fr"
		,
		"pseudo" : "superAdmin",
		"mot_de_passe" : "1234"
		}
			{
			"Message": "Success"
			}
				- HTTP_Status = 200
				- Réponse JSON
				- Body JSON = "Message"
				- Longueur JSON = 1
				- "Message" de type string

1.2	Création invalide de compte	{
		"mail":"superAdmin@superAdmin.fr"
		,
		"pseudo" : "superAdmin",
		"mot_de_passe" : "1234"
		}
			{
			"ErrorPseudo": "This username corresponds to an existing account, please enter another username."
			}
				- HTTP_Status = 200
				- Réponse JSON
				- Body JSON = "ErrorPseudo"
				- Longueur JSON = 1
				- "ErrorPseudo" de type string

2.1	Connexion valide	{
		"username" : "superAdmin",
		"mot_de_passe" : "1234"
		}
			{
			"token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ1c2VyX2lkIjoiNCIsInVzZXJfdXNlcm5hbWUiOiJzdXBlckFkbWluIiwidXNlcl9yb2xlIjoidXNlciIsInVzZXJfbWFpbCI6InN1cGVyQWRtaW5Ac3VwZXJBZG1pbi5mciIsImV4cCI6MTYyMzg1MTQwNX0.cT4BzG2gMTVKembgMc_gucH2_NkFTkf3t7jQm9xyGE8"
			}
				- HTTP_Status = 200
				- Réponse JSON
				- Body JSON = "token"
				- Longueur JSON = 1
				- "token" de type string

2.2	Connexion invalide	{
		"username" : "superAdmin",
		"mot_de_passe" : "A!"
		}
			{
			"ErrorPassword": "Invalid Password"
			}
				- HTTP Status = 404
				- Réponse JSON
				- Body JSON = "ErrorPassword"

				- Longueur JSON = 1 - “ErrorPassword” de type string - “ErrorPassword” = “Invalid Password”

3	Récupération MATUI		{
			"entree": [{
			"ID_Entree": "5",
			"Date": "2021-06-11",
			"ID_Critere": "1",
			"x": "496",
			"y": "62"
			}],
			"criteres": [{
			"ID_Critere": "1",
			"Libelle": "Le composant activable existe-t-il ?",
			"Informations": **null**,
			"x": "353",
			"y": "193"
			},{
			"ID_Critere": "2",
			"Libelle": "Voulez-vous recommencer ?",
			"Informations": **null**,
			"x": "335",
			"y": "1793"
			}],
			"decisions": [{
			"ID_Decision": "1",
			"Libelle": "Non",
			"ID_Critere_entrant": "1",
			"ID_Critere_sortant": **null**
			}, {
			"ID_Decision": "2",
			"Libelle": "Non",
			"ID_Critere_entrant": "1",
			"ID_Critere_sortant": “2”
			}],
			"methodes": [{
			"ID_Methode": "1",
				- HTTP_Status = 200
				- Réponse JSON
				- Body JSON contient “entree”, “”criteres”, “decisions”, “methodes”, “methodesRessources”, “ressources”, “sortie”
				- Longueur JSON = 7
				- “entree” de type array
				- “criteres” de type array
				- “decisions” de type array
				- “methodes” de type array
				- “methodesRessources” de type array
				- “ressources” de type array
				- “sortie” de type array

			"Libelle": "M1",
			"Description": "description",
			"Effectif_preconise":"effectif"
			"Donnees_produites":"données",
			"Type_analyse": "analyse",
			"Type_methode": "methode",
			"Exemple": "exemple",
			"ID_Decision": "13",
			"x": "1533",
			"y": "527"
			}],
			"methodesRessources": [],
			"ressources": [{
			"ID_Ressource": "3",
			"Nom": "UserTest.pdf",
			"Fichier":
			"../../../src/public/documents Ressources/UserTest.pdf"
			}],
			"sortie": [{
			"ID_Sortie": "1",
			"Message": "azeazea",
			"ID_Decision": "4",
			"x": "324",
			"y": "1918"
			}]
			}	

4	Création MATUI	{
		"entree": [
		{
		"ID_Entree": "1",
		"Date": "2021-05-07",
		"ID_Critere": "1",
		"x": 17,
		"y" : 15
		}
		],
		"criteres": [
		{
		"ID_Critere": "1",
		"Libelle": "Does the activatable component exist ?",
		"Informations": null,
		"x": 17,
		"y" : 15
		},
		{
		"ID_Critere": "2",
			{
			"Criteres": "Success CRITERE",
			"Decisions": "Success DECISION",
			"Methodes": "Success METHODE",
			"MethodesRessources": "Success METHODE RESSOURCE",
			"Entree": "Success ENTREE",
			"Sortie": "Success SORTIE"
				- HTTP_Status = 200
				- Réponse JSON
				- Body JSON contient “Criteres”, “”Decisions”, “Methodes”, “MethodesRessources”, “Entree”, “Sortie”
				- Longueur JSON = 6
				- “Criteres” de type string
				- “Decisions” de type

		"Libelle": "Is the activatable component dynamic or static ?",
		"Informations": "Dynamic components product data whereas static component doesn't.",
		"x": 17,
		"y" : 15
		},
		{
		"ID_Critere": "3",
		"Libelle": "Do you want to exchange with a single user ?",
		"Informations": null,
		"x": 17,
		"y" : 15
		}
		],
		"decisions": [
		{
		"ID_Decision": "1",
		"Libelle": "Yes",
		"ID_Critere_entrant": "1",
		"ID_Critere_sortant": "2"
		},
		{
		"ID_Decision": "2",
		"Libelle": "No",
		"ID_Critere_entrant": "2",
		"ID_Critere_sortant": "3"
		},
		{
		"ID_Decision": "3",
		"Libelle": "No",
		"ID_Critere_entrant": "2",
		"ID_Critere_sortant": "3"
		},
		{
		"ID_Decision": "4",
		"Libelle": "No",
		"ID_Critere_entrant": "3",
		"ID_Critere_sortant": "1"
		},
		{
		"ID_Decision": "5",
		"Libelle": "No",
		"ID_Critere_entrant": "1",
		"ID_Critere_sortant": "3"
		},
		{
		"ID_Decision": "6",
		"Libelle": "No",
		"ID_Critere_entrant": "3",
		"ID_Critere_sortant": null
		}
		],
		"methodes": [
		{
		"ID_Methode": "1",
		"Libelle": "M1: Social Probes, technical probes",
		"Description": "Description here",
		"Effectif_preconise": "Between 6 and 20 people of different profiles",
		"Donnees_produites": "Audio, video, field documents, logbook",
		"Type_analyse": "Annotations,
			}	string
				- “Methodes” de type string
				- “MethodesRessources” de type string
				- “Entree” de type string
				- “Sortie” de type string

		thematic analysis",
		"Type_methode": "Mixed : quantitative and qualitative",
		"Exemple": "Quantify whether the professional practices of gardeners are frequent or not",
		"ID_Decision": "3",
		"x": 17,
		"y" : 15
		}
		],
		"methodesRessources": [
		{
		"ID_MethodeRessource": "1",
		"ID_Methode": "1",
		"ID_Ressource": "3"
		},
		{
		"ID_MethodeRessource": "2",
		"ID_Methode": "1",
		"ID_Ressource": "4"
		}
		],
		"sortie": [
		{
		"ID_Sortie": "1",
		"message": "Message de fin !",
		"ID_Decision": "6",
		"x": 17,
		"y" : 15
		}
		]
		}		

5	Récupération des comptes		[
			{
			"ID_Utilisateur": "1",
			"Mail": "superAdmin@superAdmin.fra",
			"Pseudo": "superAdmin",
			"Role": "user"
			}
			]
				- HTTP_Status = 200
				- Réponse JSON
				- Body JSON contient “ID_Utilisateur”, “Mail”, “Pseudo”, “Role

6	Suppression de compte	{
		"id" : 4
		}
			{
			"Message": "Success"
			}
				- HTTP_Status = 200
				- Réponse JSON
				- Body JSON contient “Message”
				- Longueur JSON = 1
				- “Message” =

				“Success”

7	Modification de rôle	{
		"id_utilisateur": "4",
		"role" : "super-admin"
		}
			{
			"0": [
			{
			"ID_Utilisateur": "4",
			"Mail": "superAdmin@superAdmin.fr",
			"Pseudo": "superAdmin",
			"Role": "super-admin"
			}],
			"Message": "Success"
			}
				- HTTP_Status = 200
				- Réponse JSON
				- Body JSON contient “0” et “Message”
				- Longueur JSON = 2
				- “0” de type array et contient “ID_Utilisateur”, “Mail”, “Pseudo”, “Role”
				- “Message” = “Success”

8	Récupérer la liste de documents associé à une méthode	{
		"id_methode" : "1"
		}
			[
			{
			"ID_Ressource": "3",
			"Nom": "UserTest.pdf",
			"Fichier": "../../../src/public/documentsRessources/UserTest.pdf"
			}
			]
				- HTTP_Status = 200
				- Réponse JSON
				- Body JSON contient “ID_Ressource”, “Nom”, “Fichier”

9	Lire le contenu de la page d’accueil		{
			"description": "Description here"
			}
				- HTTP_Status = 200
				- Réponse JSON
				- Body JSON contient “description”
				- Longueur JSON = 1
				- “description” de type string

10	Modifier le contenu de la page d’accueil	{
		"description" : "Test pour changer le contenu de l’accueil"
		}
			{
			"description": "Test pour changer le contenu de l’accueil"
			}
				- HTTP_Status = 200
				- Réponse JSON
				- Body JSON contient “description”
				- Longueur JSON = 1
				- “description” de type

				string

11	Lire tous les documents		{
			"ressources": [
			{
			"ID_Ressource": "1",
			"Nom": "UserTest.pdf",
			"Fichier": "../../../src/public/documentsRessources/UserTest.pdf"
			}
			]
			}
				- HTTP_Status = 200
				- Réponse JSON
				- Body JSON contient “ressources”
				- “ressources” de type array
				- Éléments de “ressources” contiennent “ID_Ressource”, “Nom”, “Fichier”

12	Supprimer un document existant dans un dossier	{
		"id" : "3"
		}
			{
			"Directory": "Success DIRECTORY",
			"DataBase": "Success DB"
			}
				- HTTP_Status = 200
				- Réponse JSON
				- Body JSON contient “Directory” et “DataBase”
				- “Directory” de type string
				- “DataBase de type string

12	Supprimer un document inexistant dans un dossier	{
		"id" : "3"
		}
			{
			"Error": "Error, file doesn't exist"
			}
				- HTTP_Status = 404
				- Réponse JSON
				- Body JSON contient “Error”
				- “Error” de type string

13	Lire le message de fin de questionnaire		{
			"message": "Message de fin !"
			}
				- HTTP_Status = 200
				- Réponse JSON
				- Body JSON contient “message”
				- “message” de type string

5. Tests fonctionnels React

5.1. Identification

Numéro	Scénario
1	Chargement du composant login
2	Chargement des inputs
3	Chargement du bouton d’action
4	Prise en compte des valeurs des inputs
5	Chargement des méthodes retenues
6	Chargement du nombre de méthode retenu
7	La valeur d’un composant méthode est égale à la valeur du tableau des méthodes correspondant à son id
8	La vue Summary est correctement rendue
9	La vue Home est correctement rendue
10	La vue Login est correctement rendue
11	La vue Register est correctement rendue
12	La vue Issues est correctement rendue

5.2. Description

Scénario 1: Ce scénario consiste à simuler la création de la vue Login dans un DOM fictif.

Scénario 2: Ce scénario consiste à vérifier que la vue Login chargée dans le Scénario 1 possède les éléments inputs correspondants à l'id “username” et “password”.

Scénario 3: Ce scénario consiste à vérifier que la vue Login chargée dans le Scénario 1 possède l’éléments button permettant d’envoyer les informations de connexion.

Scénario 4: Ce scénario consiste à vérifier que les éléments input de la vue Login chargée dans le Scénario prennent en compte les informations entrées par l’utilisateur.

Scénario 5: Ce scénario consiste à vérifier que les méthodes affichées dans la vue Summary sont bien affichées dans leur intégralité.

Scénario 6: Ce scénario consiste à vérifier la cohérence des informations affichées lors du chargement de la vue Summary.

Scénario 7: Ce scénario consiste à vérifier que les informations affichées lors du chargement de la vue Summary par l'intermédiaire du composant MethodCard sont cohérentes avec les informations données au composant Summary.

Scénario 8: Ce scénario consiste à vérifier la structure et le bon chargement de la vue Summary par rapport aux anciennes snapshots.

Scénario 9: Ce scénario consiste à vérifier la structure et le bon chargement de la vue Home par rapport aux anciennes snapshots.

Scénario 10: Ce scénario consiste à vérifier la structure et le bon chargement de la vue Login par rapport aux anciennes snapshots.

Scénario 11: Ce scénario consiste à vérifier la structure et le bon chargement de la vue Register par rapport aux anciennes snapshots.

Scénario 12: Ce scénario consiste à vérifier la structure et le bon chargement de la vue Issues par rapport aux anciennes snapshots.

5.3. Dépendances

Scénario	Test à mener au préalable
1	
2	1
3	1
4	1, 2, 3
5	
6	5
7	5, 6
8	
9	
10	
11	
12	

5.4. Procédure de test

Scénario	Test	Données entrée	Résultats attendus	Critère de validation
1	Chargement du composant login	<Login/>	true	Composant défini dans le DOM
2	Chargement des inputs	<Login/>	true	Input défini dans le DOM
3	Chargement du bouton d'action	<Login/>	true	Button défini dans le DOM
4	Prise en compte des valeurs des inputs	<Login/> { value: "supercoco" } { value: "test" }	"supercoco" "test"	Inputs récupèrent les bonnes valeurs

5	Chargement des méthodes retenues	<Summary/>
		retainedMethods = [{
		id: 1,
		Libelle: "M1: Social Probes, technical probes",
		Description: "desc",
		Effectif_preconise: "10",
		Donnees_produites: "Audio, video",
		Type_methode: "Annotations",
		Type_analyse: "quantitative",
		Exemple: "M1 Exemple"
		},
		{
		id: 2,
		Libelle: "M2: Observations in situ",
		Description: "desc",
		Effectif_preconise: "6",
		Donnees_produites: "Audio, video",
		Type_methode: "Annotations",
		Type_analyse: "qualitative",
		Exemple: "M2 Exemple"
		}]
			true	Le nombre de méthode affiché est égal au nombre de méthode retenues

6	Chargement du nombre de méthode retenu	<title/> <number/>	true	Le nombre indiqué dans le title est égal à la longueur du tableau des méthodes

7	La valeur d'un composant méthode est égale à la valeur du tableau des méthodes correspondant à son id	<methodCard/>
		retainedMethods = [{
		id: 1,
		Libelle: "M1: Social Probes, technical probes",
		Description: "desc",
		Effectif_preconise: "10",
		Donnees_produites: "Audio, video",
		Type_methode: "Annotations",
		Type_analyse: "quantitative",
		Exemple: "M1 Exemple"
		},
		{
		id: 2,
		Libelle: "M2: Observations in situ",
		Description: "desc",
		Effectif_preconise: "6",
		Donnees_produites: "Audio, video",
		Type_methode: "Annotations",
		Type_analyse: "qualitative",
		Exemple: "M2 Exemple"
		}]
			true	le libellé de la méthode est égal au libellé de la méthode du tableau [retainedMethods] correspondant à son id

8	La vue Summary est correctement rendue	<Summary/>
		retainedMethods = [{
		id: 1,
		Libelle: "M1: Social Probes, technical probes",
		Description: "desc",
		Effectif_preconise: "10",
		Donnees_produites: "Audio, video",
		Type_methode: "Annotations",
		Type_analyse: "quantitative",
		Exemple: "M1 Exemple"
		},
		{
		id: 2,
		Libelle: "M2: Observations in situ",
		Description: "desc",
		Effectif_preconise: "6",
		Donnees_produites: "Audio, video",
		Type_methode: "Annotations",
		Type_analyse: "qualitative",
		Exemple: "M2 Exemple"
		}]
			cf : __snapshots__	le rendu est cohérent avec la précédente snapshot

9	La vue Home est correctement rendue	<Home/>	*cf : __snapshots__*	le rendu est cohérent avec la précédente snapshot

10	La vue Login est correctement rendue	<Login/>	*cf :* *__snapshots__*	le rendu est cohérent avec la précédente snapshot
11	La vue Register est correctement rendue	Register/>	*cf :* *__snapshots__*	le rendu est cohérent avec la précédente snapshot

12	La vue Issues est correctement rendue	<Issues/>
		issue = {
		id: 1,
		Libelle: "Activable component ?",
		Informations : "more informations"
		}
		
		decisions = [
		{
		ID_Decision: "1",
		ID_Critere_sortant: "1",
		ID_Critere_sortant: "2",
		Libelle: "yes"
		},
		{
		ID_Decision: "1",
		ID_Critere_sortant: "1",
		ID_Critere_sortant: "4",
		Libelle: "no"
		}
		]
			cf : *__snapshots__*	le rendu est cohérent avec la précédente snapshot

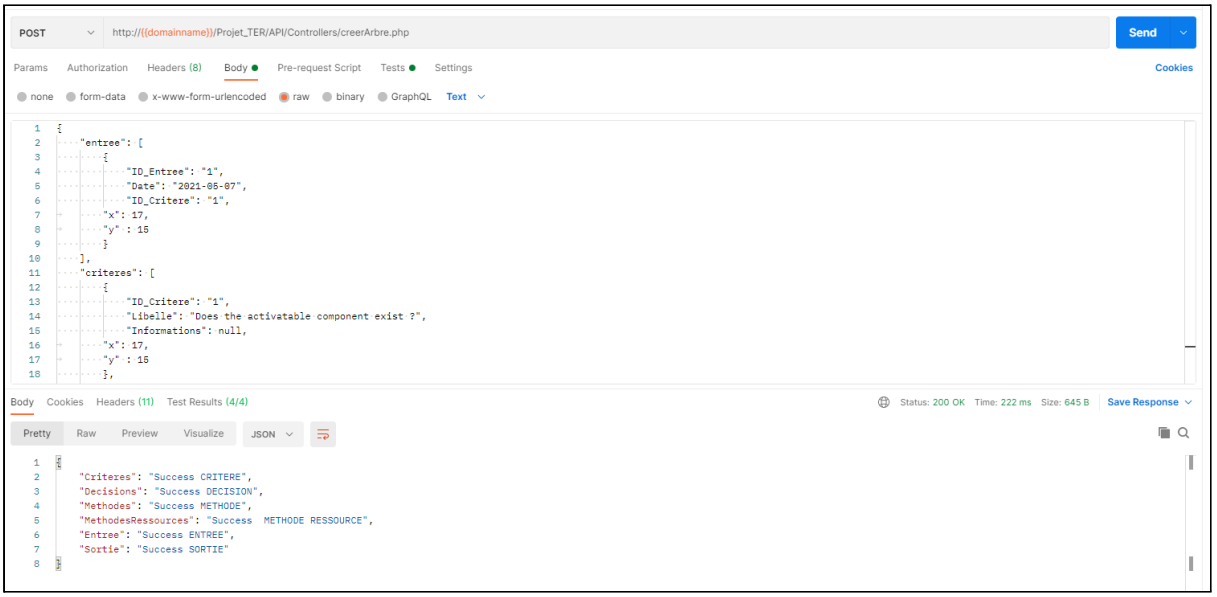
6. Vérification de la documentation

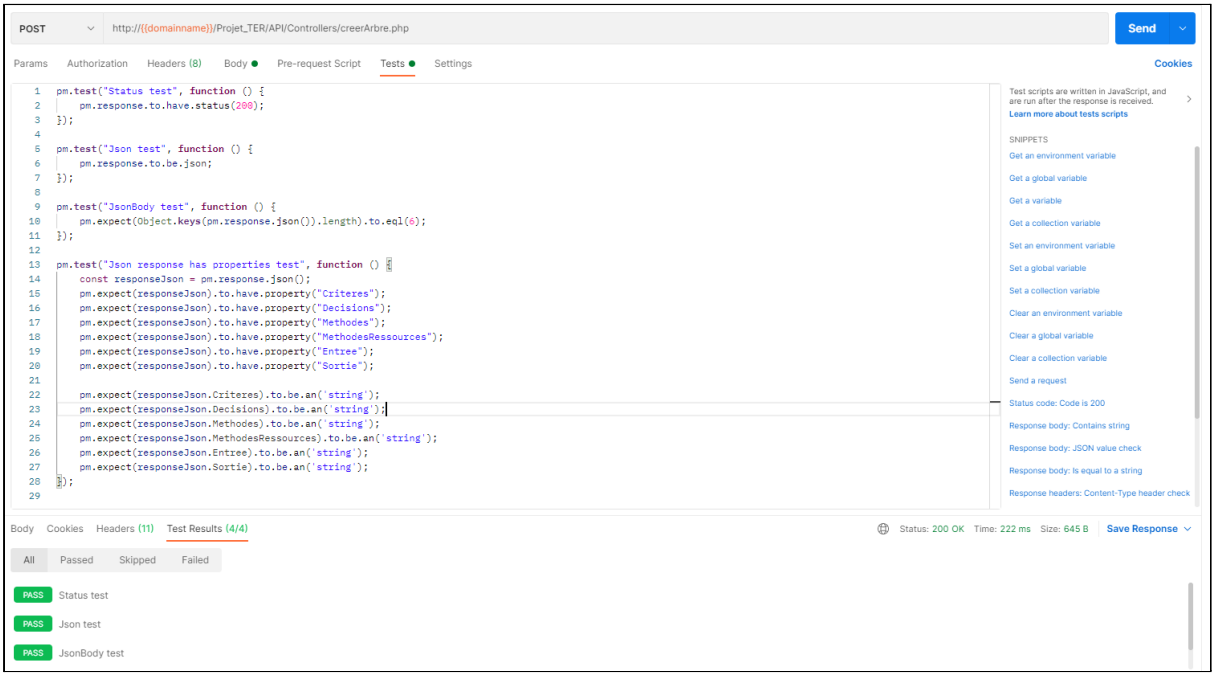
La documentation recense tous les tests réalisés jusqu'à présent. Ils ont été développés pendant le développement et utilisés tout au long du projet.

Les tests assurent que l'on récupère les données correctes selon les contextes d'appel à l'API et que les requêtes sont bien exécutées (grâce aux échanges avec la base de données). Ils permettent aussi de vérifier que le DOM renvoie bien ce qu'il faut sur la page du navigateur web en fonction des conditions de rendu.

Lors de la comparaison avec le cahier de recette (qui sert de référence aux tests), on observe que plusieurs scénarios de tests ont été ajoutés. Cela vient du fait que de nouvelles fonctionnalités, et donc de nouveaux besoins, sont apparus au fur et à mesure du développement du projet. En effet, certaines contraintes n'avaient pas été envisagées lors de la phase de conception.

7. Annexes





Annexe 1: Exemple de tests d'API pour la création de l'arbre MATUI (Scénario 4)

8. Glossaire

Postman - Environnement de développement d'API. Permet de concevoir, simuler, tester les API.

Jest - Framework de test JavaScript conçu pour écrire des tests de façon accessible et polyvalente.

DOM - Interface de de programmation normalisée par le W3C, qui permet à des scripts d'examiner et de modifier le contenu du navigateur web. Par le DOM, la composition d'un document HTML ou XML est représentée sous forme d'un jeu d'objets.

Diagramme MATUI - Diagramme produit par Nadine MANDRAN et Sophie DUPUY qui est un logigramme permettant d'avoir des méthodes et outils pertinents pour récolter des besoins utilisateurs.

Logigramme - Schémas qui représente un processus, un système ou un algorithme informatique. Utilisé dans de multiples domaines pour documenter, étudier, planifier, améliorer et faire partager des processus souvent complexes ainsi transposés dans des schémas clairs et faciles à comprendre. Ils utilisent des rectangles, des ellipses, des losanges et potentiellement de nombreuses autres formes pour définir le type d'étape, ainsi que des flèches de connexion pour définir le flux et la séquence.

THEDRE - La "recherche sur la conception d'expériences humaines traçables" est une approche qui se concentre sur la recherche en informatique centrée sur l'homme. Il aide les chercheurs à intégrer des méthodes de production de données issues des sciences humaines et sociales pour l'informatique. Nom du site sur lequel le projet sera disponible.

API - Ensemble normalisé de classes, de méthodes, de fonctions et de constantes qui sert de façade par laquelle un logiciel offre des services à d'autres logiciels.

JavaScript - Langage de programmation de scripts principalement employé dans les pages web interactives et à ce titre est une partie essentielle des applications web.

Tests fonctionnels - Imitation du comportement d'un utilisateur lorsque celui-ci teste une fonctionnalité.

Requête HTTP - Protocole de communication client-serveur. Action spécifique demandée au serveur.

Base de données - Conteneur qui permet de stocker et de retrouver des données structurées, semi-structurées ou des données brutes (information).

JSON - Représentation de données structurées.

Clés étrangères - Contrainte qui garantit l'intégrité référentielle entre deux tables. Identifie une colonne ou un ensemble de colonnes d'une table comme référençant une colonne ou un ensemble de colonnes d'une autre table (la table référencée).

Snapshot - Fichier de référence stocké à côté d'un test pour pouvoir comparer les deux. Utile pour voir un changement inattendu.

9. Références

Lien de documentation pour les tests Postman de l'API:
https://www.postman.com/automated-testing/ (visité en juin 2021)
https://learning.postman.com/docs/writing-scripts/test-scripts/ (visité en juin 2021)
https://learning.postman.com/docs/writing-scripts/script-references/test-examples/ (visité en juin 2021)

Lien vers les tests Postman de l'API:
https://www.getpostman.com/collections/14bfd31a3bb516a684e6

10. Index

Postman - Partie Introduction - page 4
Jest - Partie Introduction - page 4
DOM - Partie Introduction - page 4
Diagramme MATUI - Partie Objectifs et méthodes- page 4
Logigramme - Partie Objectifs et méthodes- page 4
JavaScript - Partie Objectifs et méthodes- page 4
API - Partie Tests fonctionnels de l'API - page 5
Tests fonctionnels - Partie Tests fonctionnels - page 5 et 16
Requête HTTP - Partie Concept de base - page 5
Base de données - Partie Concept de base - page 5
JSON - Partie Concept de base - page 5
Clés étrangères - Partie Contraintes - page 7
Snapshot - Partie Description - page 17