CAHIER DE RECETTES
GESTION DE SERVICES

Membres :
- GHERNAOUT Mohamed Wassim
- KAHOUADJI Lydia
- LAADIDAOUI Omar
- MEDJOUB Naoual
- MEGHOUCHE Rafik
- MERZEG Ramzi Ameur
- MOKRANE Islam
- SALAS Gauthier

Client :
Pr. Djelloul ZIADI

Master 1 GIL Promotion : 2021 / 2022

Historique

Version	**Date**	**Modifications réalisés**	**Rédigé par**	**Relu par**
0.1	15/11/2021 15/12/2021	Rédaction	MOKRANE Islam, Wassim GHERNAOUT, Omar Laadidaoui, Naoual MEDJOUB	
0.1	24/12/2021	Vérification		MOKRANE Islam, Wassim GHERNAOUT, Omar Laadidaoui, Naoual MEDJOUB

Table des matières

Cahier de recettes

1.1 Objet

Ce document regroupe les différents éléments du plan de test que l'on compte appliquer au long du développement.

L'application qui devra être testé a pour but premier de gérer la gestion des services horaires des enseignants.

Les tests seront effectués avec 3 configurations utilisateurs : enseignant, responsable de formation, administrateur.

La phase de test enseignant testera les fonctionnalités de la partie consultation et demande de programme horaire.

La phase de test responsable formation testera les fonctionnalités des parties Gestion des programmes horaires.

La phase de test administrateur testera les fonctionnalités des parties Gestion de formations, gestions des enseignants, gestion des sessions de choix.

Dans chaque phase de test, il y aura deux types de tests, les test front-end et les test back-end. Pour les tests front-end, on pense se diriger vers Jest qui est le framework de test unitaire le plus utilisé adapté à ReactJS.

Du côté back-end, plusieurs framework s'offrent à nous tels que encore Jest ou Macho pour Nodejs.

1.2 Documents applicables et de référence

- Spécification Techniques des Besoins (STB)

- Document d'Analyse Technique (DAT)

1.3 Environnement de test

Les tests seront effectués dans une VM diposant de l'environnement Nodejs, Apache, MySql et une base de données remplie au préalable.

Ces tests seront effectués avec les navigateur Firefox, Chrome dans leur versions les plus stables.

1.4 Responsabilités

En ce qui concerne l'organisation de la création des tests, nous effectuerons des tests unitaires. Chaque composant devra être testé dès lors que son développement est achevé afin d'éviter au maximum tout problème (ou au moins les détecter le plus tôt possible). Les tests seront écrits par un autre membre de l'équipe que celui qui développe le composant pour éviter les tests biaisés.

1.5 Stratégie de tests

Nous avons décidé d'appliquer une stratégie TDD (Test Driven Development). Avant le développement d'une fonctionnalité, un membre de l'équipe (autre que celui qui doit développer ladite fonctionnalité) écrira un test et le lancera avec des données erronées pour vérifier si celui-ci échoue. Une fois la fonctionnalité finie, le composant devra passer le test qui a été créé pour lui. Le fait de faire écrire le test par un autre membre de l'équipe permet d'éviter d'écrire un test qui serait biaisé par la vision que le développeur a de son futur composant. Pour effectuer les tests dans la partie back-end nous utiliserons Mocha.

En ce qui concerne l'équipe front-end nous effectuons principalement Jest en suivant des procédures de test prédéfinis. A la fin de chaque semaine, les fonctionnalités développées durant cette semaine seront testées. Ensuite à la fin de chaque sprint tous les tests devront être refaits.

1.6 Gestion des anomalies

Quand une anomalie est découverte, elle sera signalée à l'ensemble de l'équipe via une carte trello. Il sera précisé sur la carte trello si le problème est au niveau du Backend ou du front-end ainsi que une courte description de l'erreur. La personne ayant développé le composant sera assignée à la tâche "correction du bug". De plus, on ouvrira une issue sur gitLab du projet pour avoir un suivi des différents problèmes rencontrés.

En fonction de la gravité de l'anomalie, celle-ci devra être traitée en priorité. Si une anomalie survient sur une fonctionnalité qui a déjà été présentée au client, elle devra être réglée avant la prochaine présentation.

1.7 Différentes aspects de test d'un système

Pour avoir une bonne phase de test, un système doit être testé de manière fonctionnelle, tester la robustesse, sécurité, performance et maintenance ainsi que les exigences de ce dernier.

1.8 Procedures de test

Pour s'assurer du bon fonctionnement de l'application, les cas d'utilisations seront testés en Front en suivant les procédures décrites après. Tous ces tests seront répertoriés dans un tableau "suivi_des_test.xls" fourni en annexe.

1.8.1 Aspect fonctionnel

1.8.1.1 Authentification

Objet à tester : Authentification		
Action	**Résultat attendus**	**Test réussi : Oui / Non**
Cliquer sur le bouton s'authentifier	Affichage de l'interface d'authentification	
Remplir le formulaire d'authentification	Le format de l'émail et la taille du mot de passe sont corrects	
Valider l'authentification	l'utilisateur est authentifié	

Table 1.1 – Procedure de test de l'authentification

1.8.1.2 Demande de programme horaire

Objet à tester : Demande de programme horaire		
Action	**Résultat attendus**	**Test réussi : Oui / Non**
Cliquer sur une formation	Affichage de l'interface de la formation sélectionnée	
Cliquer sur un module	Affichage d'un formulaire	
Remplir le formulaire et valider	La demande est enregistrée	

Table 1.2 – Procedure de test de la demande de programme horaire

1.8.1.3 Consultation du programme horaire

Objet à tester : Consultation du programme horaire		
Action	**Résultat attendus**	**Test réussi : Oui / Non**
Cliquer sur le bouton consulter un programme horaire	Affichage du programme horaire	

TABLE 1.3 – Procedure de test de la consultation de programme horaire

1.8.1.4 Gestion des programmes horaires

Objet à tester : Gestion des programmes horaires		
Action	**Résultat attendus**	**Test réussi : Oui / Non**
Partie 1 : Consulter un programme horaire		
Cliquer sur une formation	Affichage de l'interface de la formation sélectionnée	
Partie 2 : Valider un programme horaire		
Cliquer sur le bouton valider un programme horaire	Un box de confirmation s'affiche	
Confirmer la validation	Le programme horaire est validé	
Partie 3 : Modifier un programme horaire		
Cliquer sur le bouton modifier le programme horaire	Un stepper s'ouvre à l'étape 1 avec les informations déjà rempli	
Modifier le formulaire de l'étape 1 et valider	Affichage du récapitulatif du programme	
Cliquer sur le bouton valider	Le Le programme est modifié avec tous les changements	

TABLE 1.4 – Procedure de test de la gestion de programme horaire

1.8.1.5 Gestion des enseignants

Objet à tester : Gestion des enseignants		
Action	**Résultat attendus**	**Test réussi : Oui / Non**
Partie 1 : Affichage des enseignants		
Cliquer sur consulter la liste des enseignants	Affichage de l'interface de la liste des enseignants	
Partie 2 : Ajouter un enseignant		
Cliquer sur le bouton ajouter un nouvel enseignant	L'interface d'ajout d'un enseignant s'affiche	
Remplir les informations de l'enseignant et valider	Affichage du récapitulatif de l'enseignant	
Cliquer sur le bouton valider	Le nouvel enseignant est enregistré	
Partie 3 : Modifier un enseignant		
Choisir un enseignant	l'interface de l'enseignant s'affiche	
Cliquer sur modifier l'enseignant	Un stepper s'ouvre à l'étape 1 avec les informations déjà rempli	
Modifier le formulaire de l'étape 1 et valider	Affichage du récapitulatif de l'enseignant	
Cliquer sur le bouton valider	Les informations de l'enseignant sont modifiés avec tous les changements	
Partie 3 : Supprimer un enseignant		
Choisir un enseignant	l'interface de l'enseignant s'affiche	
Cliquer sur supprimer l'enseignant	Un box de confirmation s'affiche	
Confirmer la suppression de l'enseignant	l'enseignant est supprimé	

Table 1.5 – Procedure de test de la gestion des enseignants

1.8.1.6 Gestion des formations

Objet à tester : Gestion des formations		
Action	**Résultat attendus**	**Test réussi : Oui / Non**
Partie 1 : Affichage des formations		
Cliquer sur consulter la liste des formations	Affichage de l'interface de la liste des formations	
Partie 2 : Ajouter une formation		
Cliquer sur le bouton ajouter une nouvelle formation	L'interface d'ajout d'une formation s'affiche	
Remplir les informations de la formation et valider	Affichage du récapitulatif de la formation	
Cliquer sur le bouton valider	Le nouvelle formation est enregistrée	
Partie 3 : Modifier une formation		
Choisir une formation	l'interface de la formation s'affiche	
Cliquer sur modifier la formation	Un stepper s'ouvre à l'étape 1 avec les informations déjà remplis	
Modifier le formulaire de l'étape 1 et valider	Affichage du récapitulatif de la formation	
Cliquer sur le bouton valider	Les informations de la formation sont modifiés avec tous les changements	
Partie 3 : Supprimer une formation		
Choisir une formation	l'interface de la formation s'affiche	
Cliquer sur supprimer la formation	Un box de confirmation s'affiche	
Confirmer la suppression de la formation	la formation est supprimée	

Table 1.6 – Procedure de test de la gestion des formations

Objet à tester : Gestion des modules		
Action	**Résultat attendus**	**Test réussi : Oui / Non**
Partie 1 : Affichage des modules		
Cliquer sur une formations	Affichage de l'interface de la formation	
Cliquer sur consulter la liste des modules	Affichage de l'interface de la liste des modules	
Partie 2 : Ajouter un module		
Cliquer sur le bouton ajouter un nouveau module	L'interface d'ajout d'un module s'affiche	
Remplir les informations de le module et valider	Affichage du récapitulatif du module	
Cliquer sur le bouton valider	Le nouveau module est enregistré	
Partie 3 : Modifier un module		
Choisir un module	l'interface du module s'affiche	
Cliquer sur modifier le module	Un stepper s'ouvre à l'étape 1 avec les informations déjà remplis	
Modifier le formulaire de l'étape 1 et valider	Affichage du récapitulatif du module	
Cliquer sur le bouton valider	Les informations du module sont modifiés avec tous les changements	
Partie 3 : Supprimer un module		
Choisir un module	l'interface du module s'affiche	
Cliquer sur supprimer le module	Un box de confirmation s'affiche	
Confirmer la suppression du module	le module est supprimé	

TABLE 1.7 – Procedure de test de la gestion des modules

1.8.1.7 Gestion des sessions de choix

Objet à tester : Gestion des sessions de choix		
Action	**Résultat attendus**	**Test réussi : Oui / Non**
Partie 1 : Ouvrir une session de choix		
Cliquer sur le bouton ouvrir une session de choix	Session ouverte, les enseignants ont accès aux demandes	
Partie 2 : Fermer une session de choix		
Cliquer sur le bouton fermer une session de choix	Session fermée, les enseignants n'ont plus accès aux demandes	

TABLE 1.8 – Procedure de test de la gestion des sessions de choix

1.8.2 Exigences Non-Fonctionnelles

1.8.2.1 Exigences de réalisation

ID	Spécification	Méthode de Test
STB-REA-01	Le back-end de la plate-forme doit être développé avec Nodejs, ExpressJs	Le code du back-end de l'application est en NodeJs
STB-REA-02	Le front-end de la plate-forme doit être développé avec ReactJS	Le code du front-end de l'application est développé avec ReactJS
STB-REA-03	La gestion de la base de données doit être faite par MySQL	Le code de la base de données est en MySQL

1.8.2.2 Exigences opérationnelles

ID	Spécification	Méthode de Test
STB-OP-01	L'application devra tourner sur serveur récent ex : Ubuntu LTS 18.04	Déploiement de l'application dans un conteneur Docker Ubuntu LTS 18.04

1.8.2.3 Exigences de qualité

ID	Spécification	Méthode de Test
STB-QUA-01	Les erreurs/exceptions doivent être traitées et générer un log	Mise en place des scénarios provoquant des erreurs (data erronée ...) et Vérification des logs et de l'état du système avant et après.
STB-QUA-02	L'application doit permettre de générer/exporter un document du programme horaire. (librairie jspdf-react)	Générer le document à la main et vérifier son contenu.

1.8.2.4 Exigences d'interface

ID	Spécification	Méthode de Test
STB-INT-01	L'application doit être ergonomique et intuitive pour ses utilisateurs	Suivi de l'application et s'assurer qu'elle respecte les critères ergonomiques des IHM On peut utiliser par exemple : Critères Ergonomiquespour l'Évaluation-d'Interfaces Utilisateurs
STB-INT-02	La règles des 3 clics doit être respectée	Parcourir et tester l'application à la main

1.8.3 Aspect Robustesse

1.8.3.1 Pourquoi tester

Lorsque la robustesse des tests logiciels est évoquée, cela signifie généralement que le système déployé ou encore en cours de développement fonctionne bien dans des conditions normales ou ordinaires. Les tests robustes consistent à améliorer la fiabilité et à trouver ces cas critiques en entrant des données qui imitent les conditions environnementales extrêmes pour aider à déterminer si le système est suffisamment robuste pour fonctionner.

1.8.3.2 Méthodes de test garantissent la robustesse

Bien qu'il existe de nombreuses techniques et outils de test de robustesse, le fuzz est probablement la méthode de test la plus largement utilisée car elle existe depuis des décennies. Les tests Fuzz se sont avérés très efficaces et c'est une méthode relativement simple où vous créez des cas de test avec de multiples variations d'entrées inattendues et surveillez les exceptions.

Après des tests exhaustifs, s'il ne plante pas, n'échoue pas aux assertions de code intégrées ou n'a pas de fuites de mémoire potentielles, alors vous avez atteint un degré élevé de robustesse logicielle.

Nous proposons pour cela de tester les règles de gestion :

1.8.3.3 Authentification

Objet à tester : Authentification		
Action	**Résultat attendus**	**Test réussi : Oui / Non**
Partie 01 : Saisie d'email		
Saisir plusieurs formats d'e-mail non-valides	Affichage d'un message d'erreur indiquant que l'email n'est pas valide	
Saisir plusieurs d'email valides mais qui n'existent pas dans la base de données	Affichage d'un message d'erreur indiquant que l'e-mail n'est pas lié à un compte	
Partie 02 : Saisie du mot de passe		
Entrez un mot de passe de moins de 6 caractères	Affichage d'un message d'erreur indiquant que le mot de passe est court	
Partie 03 : Combinaison e-mail et mot de passe		
Saisir un faux e-mail et un mot de passe	Affichage d'un message d'erreur indiquant que l'e-mail ou le mot de passe est incorrectt	
Saisir un e-mail existant et un mot de passe incorrect	Affichage d'un message d'erreur indiquant que le mot de passe est incorrect	
Cliquer sur le bouton connexion sans remplir les champs e-mail et mot de passe	Affichage d'un message d'erreur indiquant que l'utilisateur doit saisir un e-mail et un mot de passe	

TABLE 1.9 – Procedure de test des règles de gestion de l'authentification

1.8.3.4 Demande du programme horaire

Objet à tester : Demande du programme horaire		
Action	**Résultat attendus**	**Test réussi : Oui / Non**
Partie 01 : Validation a blanc		
Cliquer sur enregistrer sans remplir le formulaire de demande ou remplir tous les champs a 0	La demande ne sera pas enregistrée	
Partie 02 : Saisie de la valeur du cours		
Saisir des valeurs autres que 1 et 0	La saisie d'une valeur autre que 1 ou 0 n'est pas autorisé	
Saisir des lettres et des symboles	La saisie d'une valeur autre que 1 ou 0 n'est pas autorisé	
Partie 03 : Saisie du nombre de groupe		
Saisir des valeurs inférieures ou supérieures que le nombre de groupes	La saisie d'une valeur inférieure ou supérieure que le nombre de groupes n'est pas autorisé	

TABLE 1.10 – Procedure de test des règles de gestion de la demande du programme horaire

1.8.3.5 Gestion des enseignants

Objet à tester : Gestion des enseignants		
Action	**Résultat attendus**	**Test réussi : Oui / Non**
Partie 01 : Ajout d'un enseignant		
Ajouter un enseignant avec un e- mail déjà utilisé	Affichage d'un message d'erreur indiquant que l'e-mail est déjà utilisé	
Partie 02 : Modification d'un enseignant		
Modifier l'e-mail d'un enseignant avec un e-mail déjà utilisé	Affichage d'un message d'erreur indiquant que l'e-mail est déjà utilisé	

TABLE 1.11 – Procedure de test des règles de gestion de la gestion des enseignants

1.8.3.6 Gestion des formations

Objet à tester : Gestion des formations		
Action	**Résultat attendus**	**Test réussi : Oui / Non**
Partie 01 : Ajouter une formation		
Ajouter une formation qui existe déjà	Afficher un message d'erreur indiquant que la formation existe déjà dans l'université	
Partie 02 : Modification d'une formation		
Modifier une formation en une formation qui existe déjà	Afficher un message d'erreur indiquant que la formation existe déjà dans l'université	
Partie 03 : Ajouter un module		
Ajouter un module qui existe déjà dans une même formation	Afficher un message d'erreur indiquant que le module existe dans cette formation	
Partie 04 : Modification d'un module		
Modifier un module en un module qui existe déjà dans une même formation	Afficher un message d'erreur indiquant que le module existe dans cette formation	

TABLE 1.12 – Procedure de test des règles de gestion de la gestion des formations