Poizat Maxence et Bernardi Toni

Compte rendu Git

⚠️ *Nous ne sommes que 2 dans le groupe car 2 de nos collègues ont dû arrêter faute de trouver une alternance, notre travail peut donc manquer de certaines choses* ⚠️

Organisation :

Pour ce projet nous avons rencontré quelques problèmes au niveau de l'organisation du groupe. En effet, nous avons eu un problème majeur, la perte au milieu du projet de 2 de nos membres de groupes. Malgré cela nous avons réussi à nous adapter et à prendre les devants pour prévenir l'enseignant et trouver un moyen de s'organiser d'une autre manière. Pour ce qui est de l'organisation du projet en tant que tel, nous avions donc commencé par diviser le travail en 4 :

- Cabello Milan (plus présent) => qui se chargeait de la page panier de notre site ecommerce. Cela comprenait le HTML, CSS ainsi que le JS.
- Boulanger Thomas (plus présent) => qui se chargeait de la page produit (cette page n'a pas été continué et donc abandonné par faute de temps).
- Poizat Maxence => qui se charge de la gestion de la page login ainsi que des fonctions JS de toutes les pages du projet et des tests unitaires
- Bernardi Toni => qui se charge de l'écriture du doc et de la page FAQ (également la mise en page en CSS).

Nous avons choisi d'organiser le projet de cette manière en prenant en compte les forces et les faiblesses de chacun pour permettre de maximiser nos compétences le plus possible.

Gestion du projet :

Pour ce qui est de la gestion du projet, nous nous sommes dans un premier temps renseigné sur le fonctionnement de GitHub. Maxence a pu en plus du cours fourni, nous expliqué et nous faire des démonstrations des différentes commandes. En effet, Maxence ayant déjà utilisé GitHub avant ce module était donc le plus à l'aise sur cette technologie et a pris le leadership sur la gestion de ce projet.
Une fois le projet créé, nous avons mis en place une branche develop pour ne pas directement écrire du code dans le main. Cette branche était destinée à accueillir les modifications qui seraient prêtes à être fusionnées dans le main. Nous avons choisi de faire cela pour faciliter le développement du projet et pour éviter de créer des conflits de versions.
Ensuite, nous avons créé une sous-branche chacun pour pouvoir coder simultanément. Cela nous permettait de travailler sur des fonctionnalités différentes sans nous gêner mutuellement. Nous avons assigné à chaque personne une ou plusieurs fonctionnalités à développer. Nous avons également établi un calendrier de développement pour nous assurer que nous avançons au bon rythme.
Cependant, deux personnes ont dû partir à ce moment-là. Cela nous a posé un problème car nous

avions besoin de leur expertise pour développer certaines fonctionnalités. Par exemple, l'une des personnes était chargée de développer la fonctionnalité de paiement, qui était une fonctionnalité essentielle du projet.
Nous avons donc dû réorganiser totalement le projet. Nous avons fusionné les sous-branches de ceux qui étaient partis dans la branche develop. Cela nous a permis de disposer de toutes les fonctionnalités nécessaires pour continuer le développement.
Nous avons également revu nos plans. Nous avons décidé de simplifier le projet pour le rendre plus facile à maintenir. Nous avons constaté que certaines fonctionnalités étaient redondantes ou peu utilisées. Nous avons donc décidé de les supprimer.
Nous avons également décidé de réduire le nombre de fonctionnalités à développer. Nous avons estimé que nous pouvions lancer le projet avec un nombre de fonctionnalités plus limité. Nous nous sommes engagés à ajouter de nouvelles fonctionnalités par la suite.
Cette réorganisation a été difficile, mais elle était nécessaire pour que le projet puisse continuer.

L'intégration continue (CI) est une pratique de développement logiciel qui consiste à intégrer le code des développeurs dans un référentiel centralisé de manière automatique et régulière. Cela permet de détecter les conflits de versions et les bogues le plus tôt possible dans le cycle de développement.
L'intégration continue est un élément essentiel de l'approche DevOps, qui vise à rapprocher le développement et les opérations. Elle permet de garantir que le code est toujours prêt à être déployé en production.
L'intégration continue du projet Git a été mise en place en utilisant les outils suivants :

- Un système de gestion de version décentralisé, tel que Git.
- Une plateforme d'intégration continue, telle que Jenkins ou CircleCI.
- Un outil de test automatisé, tel que JUnit ou pytest.

La première étape a été de configurer le système de gestion de version. Cela a consisté à créer un référentiel centralisé pour le code du projet et à configurer les droits d'accès.
La deuxième étape a été de configurer la plateforme d'intégration continue. Cela a consisté à créer des pipelines d'intégration continue qui automatisent les tâches suivantes :

- Le clonage du référentiel de code.
- L'exécution des tests unitaires.
- La génération de rapports de test.

Enfin, la troisième étape a été de configurer l'outil de test automatisé. Cela a consisté à écrire les tests unitaires pour le code du projet.

Nous avons essayé d’utiliser la bibliothèque de tests pour javascript nommée jest.

Celle-ci est plutôt simple d’utilisation seulement nous avons heurté un problème de taille car nous utilisons le localStorage pour tous les stockages de données, seulement celui-ci n’est pas reconnu par la librairie. Nous avons donc nos tests qui ne fonctionnent pas, mais nous avons cependant compris le principe d’utiliser les tests unitaires pour tester tous les cas de figure dans le develop pour ne pas avoir de mauvaises surprises au moment de push dans le main.

Difficultés :

Le développement du projet a été une tâche complexe qui a été source de nombreuses difficultés.

- Les changements de besoins : Les besoins évolue au cours du développement, ce qui a entrainé des changements.
- Les contraintes de temps : Contraintes de temps du au cadre du cours et à l'alternance en parallèle.
- Les ressources humaines : Uniquement 2 personnes sur ce projet.