PROJET PRO E-COMMERCE

Introduction

La réalisation d'un site complet d'e-commerce à partir d'un cahier des charges fonctionnelles (CDCF), en suivant une méthodologie agile vous permettra, indépendamment de votre expérience en entreprise, de suivre en théorie comme en pratique un parcours de développement idéal.

Il vous servira ainsi de jauge en entreprise pour améliorer vos processus et workflow existants, ou s'ils sont déjà de qualité, d'en avoir la confirmation, tout comme produire un renforcement ; acquérir de bonnes habitudes est un processus qui requiert une répétition sur le temps.

La deuxième vertu de cet exercice est le travail en équipe, que certains ne connaissent malheureusement pas en entreprise. Vous trouverez ici la démonstration des bienfaits qu'un bon workflow peut amener à une équipe, et notamment une synergie extraordinaire (le fameux "1 + 1 = 3").

En troisième est dernier point se trouve l'architecture d'une application d'envergure qui dépasse le cas de l'éponyme "Todo list" dont les limites tutorialesques sont rapidement atteintes. Ici, vous serez amenés à construire une application résiliente et riche de fonctionnalités, avec des contraintes de sécurité qui dépassent le verrouillage d'un message par mot de passe.

Étapes de la mise en place du projet

1. Présentation du projet
2. Lecture du CDCF
3. Mise en place des équipes
4. Mise en place du planning
5. Développement itératif
6. Rendu de la présentation finale (slides)
7. Rendu du code
8. Présentation finale (oraux)

Étapes de la mise en place du projet

Prenez le temps de bien lire le CDCF, vous serez notés suivant la fidélité de votre rendu par rapport à celui-ci.

Si un élément n’est pas clair, vous pouvez vous adresser aux représentants “clients” (les personnes qui jouent le rôle du client).

Plusieurs sessions de suivi peuvent être mises en place, mais ce sera à *votre* initiative (i.e suivant la méthode agile).

Équipes

Vous travaillerez par équipe de 3. Chacun devra travailler sur tous les aspects du projet : frontend web, app mobile, backend et organisation du travail (méthodologie agile). Afin de garantir cela, chaque élève devra réaliser au moins 3 user stories par aspect (frontend web, app mobile et backend) et participer activement à l'organisation du projet (ce sera mesuré, soyez-en certains).

Comme cela arrive souvent dans la vie professionnelle, vous n'aurez pas forcément à choisir les gens avec qui vous travaillerez.

Suivi du travail

Étant donné que la note comprend l'application de la méthodologie agile (Scrum, XP, etc., c'est à votre choix), vous devrez informer les responsables de l'évolution de votre travail à la fin de chaque sprint, qui représentera généralement d'ailleurs votre rythme d'alternance.

Vous devrez vous référer au document de référence pour le format de ce rapport de fin de sprint.

Si un membre ne travaille pas (assez), cela se verra immédiatement dans le nombre de ticket non réalisé qui se cumule. Le membre en question sera immédiatement pénalisé (-5 pts par sprint).

Rendus

Il sera attendu de chaque équipe :

- un code pour chaque aspect de l'app, tous regroupés dans *un seul repository git* ;
- l'ensemble des rapports de fin de sprint ;
- le document de présentation en PDF ;
- les slides de présentation du projet au format PDF.

Code

Notation

Les éléments de notation sont les suivants :

- qualité de la conception ;
- rigueur de la syntaxe, la (bonne) application des conventions (coding style) ainsi que du savoir-faire propre à chaque environnement ;
- tests unitaires (et en bonus si possible, test d'intégration) ;
- le workflow avec git (petits commits précis, message correct, etc.) ;
- le workflow avec GitHub (PR, review de PR, etc.) ;
- documentation (cf. ci-dessous) ;
- élégance du code (oui, l'élégance).

Tout code trop “sale” ne sera même pas corrigé et sera directement noté avec un 0. Nous ne sommes plus à l’âge de pierre de la programmation, vous devez tous savoir au moins utiliser un linter et un formatter propre à vos langages de programmation.

Il ne vous est pas demandé d’héberger en ligne votre projet, mais il ne vous est pas interdit de le faire non plus :)

Documentation technique

Vous devrez documenter votre code **à l’extérieur** de celui-ci. C’est-à-dire qu’il est inutile de mettre de // we send the list of products au- dessus d’une ligne telle que res.send(products). *Les commentaires inutiles vous enlèveront des points.* Au lieu de cela, il vous faudra présenter les éléments suivants :

- un fichier README.md à la racine du repo qui explique rapidement le contexte et la mise en place du code en général ;
- un fichier README.md à la racine de chaque dossier de chaque aspect (front/mobile/back) qui explique comment bootstrap le projet, lancer les tests, etc. ;
- un seed pour démarrer le projet en développement (uniquement pour le backend) ;
- une documentation *claire et précise* de votre API backend, correspondant au format choisi (REST/GraphQL/etc.) ;
- tous formats de documentation qui vous semblent pertinents.

Choix des technologies

Le choix des langages et frameworks est libre MAIS il est doit être validé par les responsables du ce projet. Vous devrez soumettre vos demandes *AVANT* de démarrer le projet. Aucune exception ne sera faite.

Document de présentation

C'est un joli (et nous insisterons sur le "joli") document qui présente la réalisation de votre projet, en expliquant les éléments suivants :

- la première page comportera le titre, la date, la classe et pour chaque élève le nom et prénom, l'username utilisé sur le service de versioning en ligne choisi (e.g : GitHub, GitLab, etc.) ainsi que l'e-mail pour les joindre ;
- la seconde page contiendra la totalité des informations permettant d'accéder au code, c'est-à-dire lien du repo git, etc., mais aucune information de mise en place du code ne doit y apparaitre ici (cf. documentation technique pour cela) ;
- une introduction n'expliquant pas le projet puisqu'il vous est donné, mais expliquant rapidement le cheminement et le contexte de la réalisation du projet ;
- choix des technologies et pourquoi ;
- choix de la variante de la méthodologie agile et pourquoi (e.g : XP vs Scrum) ;
- choix de l'architecture et pourquoi (e.g : MVC vs MVVM vs MVI, REST vs GraphQL) ;
- tous les diagrammes nécessaires pour comprendre votre architecture (e.g : Diagramme de classes & Diagramme de séquence UML, MCD & MPD Merise, etc.) ;

- les difficultés rencontrées, *si* difficultés il y a eu ;
- les remerciements ne sont pas obligatoire.

NE JOIGNEZ AUCUN SCREENSHOT DE VOTRE ***CODE***.

Le document doit faire entre 15 et 20 pages, et contiendra *quelques* screenshot du frontend web et de l'app mobile, PAS DE CODE.

Slides de présentation

Il ne doit pas être un copier-coller plus mince de votre document de présentation. Il reprend de nombreux éléments de ce dernier mais doit être au format paysage, comprendre les informations nécessaires à la compréhension *rapide* de l'ensemble du déroulement du projet, et doit présenter les qualités d'un document préparé pour l'oral
:

- phrases courtes ;
- pas de paragraphes ;
- moins de 5 bullets points par slides ;
- plus d'images (diagramme, etc.) que de textes, etc.

Pro tips

- ne vous précipitez pas pour écrire le code, commencez par la conception, notamment les schémas de modélisation de données, tels que le MCD, etc. ;
- soyez régulier, ne faites pas tout d’un bloc (et surtout pas au dernier moment) ;
- ne vous battez pas, ni soyez indifférents les uns envers les autres : s’il y a un problème, parlez-en aux responsables ;
- n’hésitez pas à faire du pair programming ;
- posez des questions.