Analyse critique

Livrable remis :

Lorsque nous nous sommes vu remettre le livrable du binôme B3126, il semblait à première vue complet et correcte. Il n'y avait pas d'erreurs choquantes ou de choses qui manqueraient. Ce n'est qu'en lisant attentivement celui-ci que nous avons eu des doutes sur certains points. Il n'y avait pas d'erreurs, cependant un manque de clarté à des endroits, notamment pour l'aide à inspiration des mediums pendant les consultations. Même avec la démo nous avons mis un peu de temps à comprendre qu'il s'agissait d'un moyen pour les employés de trouver de l'inspiration car la fonction même de ce service n'était pas décrit clairement. Pour l'échelle de 1 à 4, devions nous choisir aléatoirement ? Était-ce le client qui donnait les chiffres ou bien le medium ? Au final, nous avons conclu grâce aux images fournis pour les IHM. Dans la globalité cependant le livrable était clair, et nous a bien guidés à travers la réalisation du front-end.

Lors de la démo cependant nous avons constatés quelques problèmes. Tout d'abord nous avons constatés une erreur dans la notification envoyée par rapport au medium choisi, en effet celui écrit dans la notification ne correspondait pas à celui effectivement choisit.

Ensuite, si l'on se mettait en *create* et non en *drop and create* pour la persistance, il était possible de créer plusieurs employés avec la même adresse mails. Cela nous a perturbés car nous n'avions pas remarqué au début que nous avions des doublons avec un ajout d'un 5 devant l'*id* du doublon pour le nouvel *id*.

Enfin, dans certains cas l'inscription d'un client échouait mais celui-ci apparaissait tout de même dans la base de données.

Après avoir fait nos tests nous avons analysés plus en profondeur les services mis à dispositions. Ceux-ci étaient dans la majorité bien fait, excepté le fait qu'il y ait des saisies et de l'affichage dans certains des services proposés. Nous nous sommes également arrêtés sur les 2 services *estClient()* et *estEmploye()* qui nous revenaient à appeler *authentifierClient()* ou *authentifierEmploye()* mais renvoyaient un booléen. Ces services ont été créés car il n'y avait qu'un bouton de connexion dans l'IHM demandée, il fallait donc savoir quel service d'authentification appeler. Nous avons utilisé le service *estClient()* car il était disponible, cependant cela revenait à faire 2 fois la même chose (un *findByMail()*) alors qu'il aurait suffi de traiter les cas où *authentifierClient()* et *authentifierEmploye()* renvoyaient *null*.

Enfin, il y a des services qui avait une signature en *void* c'est-à-dire qu'ils ne renvoyaient rien. Par exemple les services *accepterConsultation()* ou *validerConsultation()* ne renvoyaient rien, et il était plus difficile de tester si la fonction s'était exécuté sans problème ou non. Il aurait été bien d'avoir toujours au moins un booléen pouvant signaler le bon déroulement de l'appel au service, permettant une vérification simple et rapide dans le front-end.

Dernièrement, on peut conclure sur l'esthétique demandée qui était bien trop complexe à coder avec nos compétences actuelles ou sans l'utilisation de framework spécifiques. Notamment la page d'accueil que nous avons essayé de reproduire tant bien que mal avec des css. Nous rappelons cependant que ce n'était pas le but du projet de coller à l'esthétique du livrable et donc ce point n'est pas primordial. Il reste cependant intéressant de penser que lors du design d'un projet, on peut aussi réfléchir à ce qui est faisable, envisageable.

Auto-critique de notre livrable :

Après avoir travaillé sur la partie front-end, nous avons pu mieux comprendre les enjeux et besoins de cette partie du développement d'application. Avec ces nouvelles informations nous pouvons maintenant nous rendre compte de ce qu'il manquait à notre livrable et de ce qui a pu fauter au binôme à qui il a été remis.

Premièrement, le problème majeur détecté après relecture de notre livrable est qu'il manque le service *noterCours()* et sa description dans la liste des services. Cela a sans nul doute posé soucis au binôme l'ayant reçu car nous n'avons pas pu expliquer le lien entre l'esthétique de l'IHM et le service. En effet, dans notre IHM nous proposions 3 choix de compréhensions caractérisés par des smileys et des affirmations indiquant le niveau de compréhension. Il fallait intégrer ses choix avec des entiers de 1 à 3 qui correspondent à la note du cours, car le service *noterCours()* prend lui un entier en argument. Il y a également une incohérence avec le tableau ICAR de la Pop-up Auto-évaluation qui indique l'appelle au service *finVisioEleve()*, service qui n'existe pas. Ceci est dû à un changement lors de la programmation et la décision de diviser en 2 services distincts, celui qui n'en faisait qu'un. Nous n'avons cependant pas actualisé notre tableau ICAR en conséquence, ce qui a dû confondre le binôme. Bien qu'ils aient dû s'y retrouver en parcourant notre code, démo et tests, qui, par contre, étaient clairs.

Ensuite, il aurait sans doute été mieux de mettre les tableaux ICAR à côtés de leurs IHM respectives afin de visualiser directement et clairement les interactions et ce qu'elles font.

Nous avons nous aussi fait l'erreur pour deux services, *noterCours()* et *EnvoyerBilanCours()*, de mettre une signature en *void*. Là encore, il aurait été mieux de mettre un booléen ou de renvoyer respectivement la note ou le bilan ou bien *null* en cas de problème afin de simplifier la programmation du front. De même, nous avons utilisé une signature Object[] afin de renvoyer 2 informations de types différents, par exemple pour *checkDemandeSoutien* nous renvoyons un tableau contenant le cours actuel et l'élève associé. Il aurait été plus simple de renvoyer uniquement le cours actuel en sachant que l'on peut récupérer l'élève associé en utilisant un *get* à partir du cours. Cependant, l'utilisation de la classe Object nous semble pertinente pour les services statistiques (*statNbEtablissementParCommune()* et *statNbSoutienetDuree()*).

Enfin, nous avons opté pour un design relativement simple qui n'a du normalement pas poser de problème, excepté l'affichage des statistiques sur la carte

qui est plus complexe. Bien qu'encore une fois ce ne soit pas le but du projet, il reste intéressant de le mentionner.

Conclusion

Globalement, les deux livrables présentaient et contenaient les services nécessaires au bon fonctionnement des applications et permettaient l'implémentation de celles-ci. Ce travail nous a cependant permis de nous rendre compte de la difficulté de rédiger un dossier complet, cohérent et sans erreurs. Nous avons pu comprendre quels étaient les problèmes majeurs et comment les résoudre. Enfin, nous avons par la suite pu remettre en question notre livrable avec un œil nouveau et voir des problèmes que nous ne voyions pas avant afin de les corriger dans cette analyse critique.