Analyse post-projet – If’Routard

## Damien GALLET – Thomas GUEGAN

## B3128

Ce document commente le code, ainsi que le dossier que nous avons rendu à l’issue de la première partie du TP Développement Application Système d’Information. Le sujet qui nous était attribué était If’Routard, une application d’agence de voyage en ligne.

Nous tenterons d’effectuer l’analyse la plus objective possible, dans l’objectif de ressortir 3 aspects positifs ainsi que 3 aspects négatifs de notre rendu final. Cette dernière se basera sur notre ressentiment à la réception du dossier que nous avons récupéré au début de la seconde partie (pour plus d’informations à ce sujet, voir l’analyse post-projet sur Predictif), ainsi que sur notre remise en question effectuée plusieurs semaines en aval du rendu, où nous avons eu un meilleur aperçu de notre travail.

## Points positifs :

### Une base de données bien formée.

En tant qu’application SI, If’Routard se devait de manipuler une base de données contenant des informations relativement importantes, et complètes. Pour cela, le département nous a remis des fichiers de données sources, destinés à remplir ces mêmes bases. Nous avons donc réalisés nos tables en adéquation avec ces fichiers sources : ainsi, ces derniers peuvent être exécutés à l’initialisation de la base pour la remplir avec d’importantes données : Cela va non seulement garantir l’utilisation d’informations cohérentes, mais en plus mettre à l’épreuve notre mapping d’éléments, ainsi que nos services de recherche qui ont pu être testées sur des environnements réalistes.

Ces fichiers, présents dans le dossier data du projet, sont donc essentiels à l’initialisation du projet : la syntaxe des éléments du modèle est alors identique à celle demandée (à quelques différences de vocabulaire près, qui ont été prises en compte dans les fichiers sources). Il est donc très simple de réutiliser nos données.

Enfin, un fichier détaillant les informations de connexion à la base de données par défaut est également présent.

### Un cas d’usage détaillé.

Il y a eu confusion quant à savoir si l’énoncé du projet devait être transmis en plus du code et de la conception pour la deuxième partie. Ainsi, tous les binômes n’ont pas forcément détaillé le travail attendu, et le cas d’usage associé à l’IHM demandé.

Ce n’est cependant pas notre cas : nous avons tenus à spécifier la manière d’utiliser l’IHM souhaité, et même celle qui n’est pas à implémenter, pour une meilleure compréhension. Ainsi, le lecteur de notre dossier comprend alors parfaitement le cas d’usage qu’il doit implémenter dans l’IHM client. Il peut également mieux appréhender notre vue de démonstration que nous avons employé à l’issue de la première partie puisqu’elle reprend exactement ce cas d’usage. Il n’y a donc pas ou peu de confusions possibles, et il est donc tout à fait envisageable de réaliser l’IHM sans utiliser le sujet (et nous confirmons que nous n’avons pas eu besoin ni de le transmettre, ni d’expliquer le fonctionnement du cas d’usage).

Commentaire qui peut paraître évident, mais qui a révélé son importance quand nous avons reçu le dossier : nous avons spécifié quelle IHM des deux présentées était à concevoir !

### Des services précisés et utilisables, et utilisés.

L’IHM client va utiliser un certain nombre de services : ces derniers, spécifiés dans le document de conception, ont tous été créés, utilisés dans la vue de test en tant qu’exemple (permettant de clarifier le fonctionnement), et utilisable car fonctionnels. De plus, nous n’avons pas créé une quantité incohérente de services, mais simplement ceux qui s’avéraient nécessaires au fonctionnement du cas d’usage.

De plus, comme le cas d’usage est repris à l’identique dans la vue de présentation, le fonctionnement ordonné de l’application est donc identifiable dans cette vue. Les créateurs d’IHM savent donc exactement par quelles étapes leur production doit passer pour répondre au cas d’usage demandé.

## Points négatifs

### Un excès de zèle sur l’IHM

Nous avons été assez exigeant concernant les IHM, et notamment l’IHM Web qui est à réaliser : sa forme est similaire à une interface Web de voyage moderne, mais il peut être difficile de la suivre. Mais surtout, le problème est que certaines informations sont présentes sur l’IHM sans être spécifiées dans les services, voire parfois sans même exister dans le modèle ! Nous avons eu des retours à ce sujet, et c’est effectivement un problème dérangeant.

### Un service parfois capricieux

En suivant rigoureusement la procédure d’inscription de client, tel que nous avons pu le montrer pendant la présentation, il arrive qu’un devis ne s’enregistre pas comme il le faut dans la base de données. Cette erreur arrive une fois sur deux environ, et nous ne l’avons pas corrigé avant de soumettre le code. Aussi, une erreur de fonctionnement qui n’est pas du ressort des créateurs d’IHM peut arriver, et c’est très problématique.

### Une base de données solide et pleine... si on la remplit !

Comme dit précédemment notre base de données est complète et adapté à If’Routard, et se remplit via des fichiers sources. Cependant, nous n’avons pas implémentés de scripts pour automatiser cette entrée : Autrement dit, à chaque connexion, il faut remplir la base manuellement, et c’est assez fastidieux. Encore pire, cela force l’unité de persistance à adopter la stratégie « create », pour ne pas perdre les informations que l’on a rentré à l’exécution de l’application. Cela signifie qu’on doit tester sur l’intégralité des données, et que le moindre tuple partiellement complet ajouté par essai ou test ne peut être supprimé que par réinitialisation de la base de données... entraînant ainsi un nouveau remplissage.

Nous avons bien sûr pensé à automatiser cette action, mais cela s’est avéré plus compliqué que prévu. Par manque de temps, et préférant nous concentrer sur les services importants, nous avons préféré garder cette solution.