Analyse post projet – Predict’If

## Damien GALLET – Thomas GUEGAN

## B3128

Ce document commente le code et le dossier que nous avons reçu pour la seconde partie du TP Développement Application Système d’Information : le projet Predict’If, une application de voyance 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, ainsi que sur notre propre travail que nous avons rendu à l’issue de la première partie.

Notons que nous n’allons pas nous concentrer sur chaque problème ou défaut de l’IHM, le dossier annoté est destiné à cet usage, mais nous allons voir les aspects plus généraux du rendu que l’on nous a distribué.

## Points positifs :

### Des services qui marchent :

Cela peut paraître étrange pour un code redistribuable de relever ce point, mais les services étaient tous fonctionnels, comme ils devaient l’être. De plus : chaque service présent dans le dossier de conception était codé et fonctionnel. Cela apporte une grande cohérence dans le projet : nous savions parfaitement quels services étaient utilisés pour quels IHM, et nous pouvions simplement les appeler dans les servlets pour avoir un fonctionnement impeccable.

Nous n’avons donc pas à réécrire des services, à part peut-être quelques rajouts de valeurs de retour, plus par confort que par nécessité.

### Un remplissage automatique de la base de données :

L’application nécessite une base de données pour son fonctionnement, ainsi que des tuples présents au lancement du projet pour son fonctionnement (des Mediums, des Predictions, ...). Sur ce point, le projet que nous avons reçu diffère du notre : il opte pour une entrée écrite en java via des services, et donc appelable par méthode, là où nous utilisons des fichiers sources à entrer hors du code. Et le point positif de cette entrée par méthode, c’est que le remplissage de la base de données est donc automatisé : à l’inverse de notre projet, il n’y a pas de pénible tâche d’entrée manuelle. Une simple méthode suffit, ce qui nous permet de changer la stratégie de persistance en « drop and create », ce qui est bien plus confortable pour effectuer des tests.

### Pas de travail inutile :

Nous relevons également un point très pratique du projet reçu : tout est utile. Le code comprend tous les services utiles, ni plus, ni moins, et le modèle comporte tous les attributs utiles. Ainsi, il est impossible de s’écarter sur des fonctionnalités non-demandées, inattendues. Tout le contenu du code est à utiliser à un moment ou à un autre.

## Points négatifs :

### Une conception incomplète :

Peut-être l’erreur la plus gênante du projet reçu : le dossier a été très clairement conçu pour être donné avec le sujet. De ce fait, le dossier est complètement insuffisant pour comprendre le travail demandé. La première raison : des 3 IHM demandées, il n’est jamais spécifié laquelle ne sera pas à réaliser ! Nous avons tenté de comprendre au mieux, mais nous avons été contraints de demander rapidement quel était précisément le travail demandé.

### Des IHM étranges :

Nous pensons que ce point mérite d’être relevé dans cette analyse plus que dans le dossier annoté : les 2 IHM que nous devons réaliser sont destinés au Web. Les maquettes que nous avons reçu n’ont, à notre sens, absolument pas été pensé de cette manière. Le design est compact, très ressemblant à une application fenêtré classique. De plus, l’IHM employé, proposant des recherches et des entrées d’informations, raisonne par interactions de boutons qui génèrent de nouvelles fenêtres, solution très étrange pour du web, puisque cela demande d’utiliser des pop-ups incessants.

### Un modèle de donnée difficile à utiliser :

Enfin, nous relevons un problème qui n’impacte pas la partie service (qui rappelons-le, est parfaitement fonctionnelle), mais la partie modèle : les entités utilisées pour la base de données ne comportent que le strict minimum d’informations. Cela implique que des données sont manquantes, et il serait plus que commode de les afficher dans les données (Exemple pour l’employé : seul le mail est utilisé dans les services, mais il est important pour la suite de conserver le nom, le login et le mot de passe !). Aussi, nous avons dû revoir l’architecture en conséquence.

Mais cela a engendré un problème : la méthode d’initialisation prévue pour remplir la base (permettant le remplissage automatique, voir points positifs n°3), ne suffit donc plus, puisqu’on ne peut pas alimenter les attributs « inutiles » avec les services. Aussi, nous avons dû de peupler la base avec les fichiers sources : et donc, modifier leur modèle. Précisons au passage que cette solution n’est pas complète, puisque la distinction en type des prédictions (Santé, Amour et Travail) ne gère pas cette entrée manuelle : les tables concernant les prédictions et ses sous-tables ne sont donc pas cohérentes.

En conclusion, le modèle a été pensé pour ne garder que ce qui été utilisé par les services, mais les limites de ce choix se sont clairement fait ressentir maintenant que nous voulons faire fonctionner ces derniers dans des IHM finies.