Atelier de Gestion de Projet

Janvier 2017

Cahier des charges

Tianxiao LIU & Dan VODISLAV

Table des matières

1. Introduction

1.1 Contexte

L'objectif d'apprentissage de l'atelier est d'approfondir et de pratiquer les technologies de la gestion et de la réalisation d'un projet technique transversal. Les compétences visées dans cet atelier sont celles exigées par les trois unités d'enseignement (UE) : GP, COO et BDA. Les équipes de projet (mêmes équipes qu'en projet de synthèse) sont formées pour pratiquer le travail d'équipe « **technique** ». Cet atelier est le premier atelier du Master SIC, qui sera suivi par cinq ateliers en Master SIC Pro 2e année.

Les étudiants ont une semaine pour **gérer la réalisation** de ce projet de A à Z. L'approche de gestion de projet est une adaptation de la méthode agile très connue : XP (*Extreme Programming*). Pendant le projet, les documents et versions intermédiaires du produit seront livrés, permettant l'évaluation et le suivi du travail, avec une soutenance et une démonstration à la fin du projet.

1.2 Objectif du projet

Il s'agit d'un mini système intelligent permettant de construire automatiquement les offres de séjour pour une destination de voyage.

Grâce au système, on pourra gérer les données liées à la destination, effectuer les recherches d'informations de voyage et construire les offres de séjour (avec les excursions organisées) selon les critères du client. Les critères peuvent être, par exemple, la variation thématique, coût financier, confort, etc.

Chaque équipe de projet réalisera le système en se basant sur une destination différente. Les 9 destinations à attribuer aux 9 équipes de projet sont :

- ✓ Iles Crète et Santorin
- ✓ Iles Canaries
- ✓ Iles des Petites Antilles
- ✓ Iles des Seychelles
- ✓ Iles d'Hawaï
- ✓ Ile de Tahiti
- ✓ Ile de Bali
- ✓ Ile de la Réunion
- ✓ Iles Maldives

1.3 Modalités de contrôle de connaissances (MCC)

Trois notes différentes seront faites à l'issue du projet :

Note AGP qui représente **33%** de l'UE GP. L'évaluation sera basée essentiellement sur la performance d'équipe / individuelle sur la gestion / réalisation du projet pendant la semaine d'atelier. Les matières permettant de noter seront le suivi du projet et le rapport de gestion de projet (cf. 5. Description des livraisons attendues).

Note COO qui représente **30%**de l'UE COO. L'évaluation sera basée essentiellement sur la qualité de la conception du projet, la qualité du programme, ainsi que la qualité du produit final. Les matières permettant de noter seront le code source du projet, le rapport COO et la démonstration du produit (cf. 5. Description des livraisons attendues).

Note BDA qui représente **25%** de l'UE BDA. L'évaluation sera basée sur la qualité de la conception et réalisation de la partie BDA. Les matières permettant de noter seront le rapport BDA, la soutenance du projet et la démonstration du produit final. (cf. 5. Description des livraisons attendues).

Les contributions et efforts **individuels** de chaque membre de l'équipe, à la conception, au développement et à la gestion du projet seront bien pris en compte dans la notation.

2. Exigences fonctionnelles

2.1 Organisation des données du système

Pour chaque destination, les données (informations) de voyage sont organisées comme suit :

- ✓ Une destination est composée des **sites touristiques** qui sont des lieux historiques ou lieux où on peut faire une activité de loisir particulière.
- ✓ Pour l'hébergement, comme les destinations sont des îles, on s'intéresse uniquement aux hôtels au bord de la mer, où il y a la plage intégrée dans l'hôtel.
- ✓ Il faut organiser les **excursions** pour aller aux sites touristiques (historiques ou activités) pendant le séjour. En tout cas, on ne va pas seulement rester allongé sur la plage !
- ✓ Les hôtels ont différentes gammes donc différents prix.
- ✓ Pour simplifier, on ne considère que les transports **locaux**. Les vols aller-retour pour la destination ne sont pas pris en compte par le système. On suppose que les voyageurs ne louent pas de voitures pour se déplacer librement. Ainsi, toutes les excursions sont organisées. Les deux moyens de transport possibles pour les excursions sont l'autobus et le bateau.
- ✓ Pour **chaque site** touristique, une **description textuelle complète** doit être gérée par le système, permettant de prendre en compte des mots clés spécifiques utilisés par les recherches et la construction des offres de séjour.

2.2 Recherche des informations de voyage

Le système fournit des fonctionnalités de simples recherches des informations de voyage mentionnées dans **2.1**. Via l'interface Web du système, on peut effectuer des recherches avec des critères pour lesquels on peut préciser les valeurs voulues.

2.3 Construction automatique des offres de séjour

Le système peut construire automatiquement les offres avec des critères spécifiés. Le système doit pouvoir traiter n'importe quel sous-ensemble des critères suivants :

- ✓ Les sites touristiques souhaités dans les excursions, pour lesquels on précise les mots clés présents dans la description des sites, ex. nom propre, activité, type, etc.
- ✓ Fourchette du prix total de l'offre
- ✓ Le confort (rythme) souhaité (il faut définir un moyen pour mesure le confort, ex. cela peut dépendre de la fréquence et du type des excursions, les moyens et durées des transports locaux, etc.)

Ces informations (ou un sous-ensemble) de critère de recherche seront précisées sur l'interface Web et puis le système fait le traitement « intelligent » pour construire les offres de séjour possibles. Chaque offre de séjour contiendra une série d'excursions planifiées pendant le séjour. Le système affichera les informations complètes des offres construites.

3. Exigences techniques

Les exigences techniques du projet concernent les deux unités d'enseignement COO et BDA. Concrètement, la partie COO concerne la conception globale du projet et la partie BDA concerne les exigences spécifiques de la réalisation de la partie « base de données relationnelle étendue » du système. Dans cette section, on décrit également les outils pour réaliser et gérer le projet.

3. 1 Conception Orientée Objet (COO)

Architecture de l'application

Il s'agit d'une application en architecture de 4-tiers (***Figure 1***). L'objectif est de bien définir les couches (tiers) pour produire un système composé des modules faiblement couplés, afin d'avoir son extensibilité et sa réutilisabilité. Les dépendances entre les couches (modules) sont définies avec une flèche.

Figure 1 Architecture en 4-tiers

Couche présentation

Cette couche contient essentiellement les pages Web JSF (*Java Server Faces*) : éléments HTML + les composants JSF. La mise en page est assurée par les feuilles de style CSS. La configuration de l'environnement Web est assurée par les fichiers XML.

Couche contrôleur MVC

Cette couche contient les classes utilisées comme contrôleur dans le modèle MVC (Modèle-Vue-Contrôleur). Le framework JSF est lui-même de nature MVC, c'est-à-dire que l'on utilise les classes JavaBean (JSF *ManagedBeans*) pour répondre aux requêtes venant des pages JSF (Vue), en appelant les services (méthodes) fournis par la Couche métier (Modèle). Cette couche intermédiaire assure un découplage entre la Couche présentation et la Couche métier. Les classes (JSF *ManagedBeans*) dans cette couche doivent avoir un accès aux objets (classes) de domaine (*Domain Object*) définis dans la couche métier (cf. Couche métier).

Couche métier

Cette couche s'occupe de la partie fonctionnelle et logique (le métier) de l'application. La couche peut s'organiser en plusieurs sous-modules fonctionnels permettant de réaliser les services (*Business Interfaces*) accessibles par les MVC contrôleurs. Pour implémenter ces services, on doit avoir un accès aux services d'accès et de traitement de données.

Dans cette couche, on conçoit un ensemble de classes de domaine (*Domain ou Business Objects*). Ces classes s'occupe des calculs / traitements logiques (ex. construction des offres de séjour) et de la manipulation des données « logiques ». Pour ces dernières, Le principe est d'avoir un découplage entre le modèle « logique » de données (Couches métier et présentation) et le modèle « physique » de données (Couche persistance).

Couche persistance

Cette couche sera la Base de Données étendue (BDe) que vous allez concevoir et réaliser (cf. 3.2 Bases de Données Avancées (BDA)).

A faire

- ✓ Conception des pages JSF (Couche présentation) avec les beans (Couche contrôleur MVC).
- ✓ Conception des API métier (selon les besoins de la Couche contrôleur MVC).
- ✓ Modéliser et implémenter les classes métiers (*Business Objects*).
- ✓ Conception de la Couche persistance avec la partie BDA (cf. 3.2 Bases de Données Avancées (BDA)).
- ✓ Implémentation (programmation) de votre conception.
- ✓ Réaliser les tests unitaires. Vous devez parfois « simuler » (*mock*) temporairement les entrées/sorties pour les API quand les parties concernées ne sont pas encore développées par vos collègues.
- ✓ Intégration des différentes couches du système.

3.2 Bases de Données Avancées (BDA)

Une base de données relationnelle étendue

Pour la partie BDA de votre projet, l'objectif est de construire une extension d'une BD relationnelle, qui, en plus de répondre à des requêtes SQL, puisse aussi traiter des requêtes portant sur du contenu textuel. Cette extension doit s'appuyer sur une BD relationnelle à votre choix (MySQL, Oracle, etc.) et sur le moteur d'indexation textuelle Lucene (http://lucene.apache.org/).

Votre application doit utiliser cette base de données étendue (BDe), qui sera implémentée sous la forme d'une API, à concevoir. Pour simplifier, on va considérer que seulement une des tables de la base (appelons-la *T*) peut avoir une information textuelle associée à chaque ligne. Cette information ne sera pas stockée dans la table, mais dans des fichiers texte placés dans un répertoire *R* du disque. Plus précisément, pour une ligne de clé *c* de la table *T* à laquelle on associe un texte *t*, on créera dans le répertoire *R* un fichier de nom *c* contenant le texte *t*.

L'API de la BDe devra supporter au moins les fonctionnalités suivantes :

- ✓ Déclarer le nom de la table *T* et de son attribut clé, ainsi que le nom du répertoire *R* où seront placés les fichiers texte.
- ✓ Ajouter un texte *t* à la ligne de clé *c* de la table *T*, en créant le fichier associé dans *R*.
- ✓ Créer sur disque l'index textuel Lucene sur les documents du répertoire *R*.
- ✓ Répondre à des requêtes sur la BDe, en adoptant le style JDBC (itération dans les résultats).

Types de requêtes

Les requêtes sur la BDe peuvent être de deux types :

- ✓ Requêtes SQL « normales », sans partie textuelle. Elles seront traitées de façon classique, à travers JDBC, sur la base de données relationnelle.
- ✓ Requêtes mixtes SQL-texte, contenant une clause supplémentaire **with**, qui décrit la partie textuelle de la requête. Elles seront traitées en combinant l'action de la BD relationnelle et du moteur d'indexation textuelle Lucene.

Par exemple, dans le contexte d'une application touristique, la requête

```
select nom from Site
where type='historique'
with sculpture Renaissance
```

demande le nom des destinations touristiques de type historique, dont la description textuelle parle de sculpture et/ou de Renaissance.

La requête textuelle de la clause **with** respecte la syntaxe de Lucene (voir par exemple http://lucene.apache.org/core/6_3_0/queryparser/org/apache/lucene/queryparser/classic/package-summary.html#package.description). A la différence des requêtes SQL classiques, à chaque résultat est associé un score de pertinence. Les résultats des requêtes textuelles sont triés par ordre décroissant de la pertinence, par exemple un texte contenant les deux mots recherchés sera plus pertinent qu'un qui ne contient qu'un seul des deux mots. Une requête mixte devra retourner les résultats triés par ordre décroissant de la pertinence textuelle et donner accès au score de chaque résultat.

Traitement des requêtes mixtes

Une requête est une chaine de caractères ; la requête est mixte si l'on trouve une clause **with** dans cette chaine. La requête mixte doit être décomposée en une partie SQL et une partie textuelle.

On considérera deux types de plans d'exécution possibles :

1. On exécute séparément la partie SQL et la partie textuelle de la requête, ensuite on combine les deux résultats en les joignant sur la clé de la table *T*. Il faudra donc inclure la clé de T dans la clause **select** de la partie SQL de la requête. Les résultats doivent être retournés dans l'ordre donné par la requête textuelle. Plus précisément, pour chaque résultat de la partie SQL (incluant donc une clé c de T) on parcourt les résultats de la requête textuelle pour chercher la clé c (si elle existe) et son score. Les résultats de la partie SQL qui retrouvent la clé dans les résultats de la requête textuelle, sont retournés en ordre décroissant du score de pertinence textuelle.
2. On exécute d'abord la requête textuelle et pour chaque clé retournée par celle-ci on vérifie par une requête SQL si elle fait partie du résultat final ou non (si elle respecte la partie SQL de la requête). La partie SQL de la requête doit donc être modifiée pour inclure une condition sur la valeur de la clé de T.

Il est demandé d'implémenter dans l'API de la BDe l'exécution de requêtes mixtes, en utilisant le premier type de plan, ensuite de réaliser une étude comparative entre les deux types de plans d'exécution.

Pour implémenter dans la BDe le premier type de plan d'exécution, il faudra définir dans l'API 3 opérateurs :

- ✓ Un opérateur SQL, qui reçoit une requête SQL et l'exécute sur la BD relationnelle en utilisant JDBC.
- ✓ Un opérateur textuel, qui reçoit une requête textuelle et l'exécute sur les documents du répertoire *R*. Les résultats contiennent le nom du document (clé de *T*) et son score et sont produits en ordre décroissant du score.
- ✓ Un opérateur qui combine les résultats SQL et textuels pour le premier type de plan d'exécution.

Chaque opérateur sera implémenté sous la forme d'un itérateur qui fournit deux méthodes : *init*() et *next*(). Un plan d'exécution est un arbre d'opérateurs, dont la racine produit les résultats de la requête.

L'étude comparative des deux types de plans doit décrire :

- ✓ Le fonctionnement détaillé des opérateurs des deux types de plans.
- ✓ Un comparatif des coûts d'exécution des deux types de plan, en fonction du temps d'exécution et du nombre de résultats des parties SQL, respectivement textuelle de la requête. On considère que le coût de l'exécution d'une requête SQL est le même lorsqu'on y rajoute les éléments supplémentaires dans les deux plans. On peut aussi considérer que le temps d'exécution de la partie SQL est beaucoup plus grand que celui de la partie textuelle (d'un ou deux ordres de grandeur).
- ✓ Des propositions argumentées d'amélioration du temps d'exécution des deux types de plan.

A faire

- ✓ Spécifier l'API de la BDe : paquetages, classes, méthodes, etc.
- ✓ Implémenter les opérations de l'API à l'aide de JDBC et de l'API Lucene (voir http://lucene.apache.org/core/6_3_0/index.html).
- ✓ L'étude comparative des deux types de plans.

3.3 Les outils pour la réalisation et la gestion du projet

Tableau 1 résume les outils à utiliser pour différents aspects du projet.

Aspect	Outils
Programmation	✓ Java SE **1.6** (**Ne pas prendre 1.7 ou plus** : pas compatibles avec tous les frameworks) ✓ JSF ✓ JDBC ✓ Lucene
IDE	Eclipse pour Java EE (Vous pouvez utiliser autres IDE si vous gérez bien la compatibilité de configuration pour tous les membres de l'équipe)
Contrôle version	Git (GitLab ou GitHub, **ajoutez l'utilisateur « tliu2 » dans votre projet**)
Test unitaires	JUnit (fourni)
Système Log	Log4j (fourni)
Conception / Modélisation	Software Ideas Modeler (Vous pouvez utiliser autres outils mais il faut bien garantir la qualité (rigueur) des digrammes / schémas.)
Rédaction	Word ou OpenOffice ou LaTex ou les outils en ligne

Tableau 1 : Outils pour la réalisation et la gestion du projet

4. Exigences organisationnelles

Tableau 2 illustre l'organisation de la semaine de l'atelier pour tous les participants.

<table>
<tr><th></th><th>Lun. 16/01</th><th>Mar. 17/01</th><th>Mer. 18/01</th><th>Jeu. 19/01</th><th>Ven. 20/01</th></tr>
<tr><td rowspan="4">Matin</td><td>Intro projet (TL)</td><td rowspan="3">Encadr. GP (TL)</td><td rowspan="4">Autonomie</td><td rowspan="4">Autonomie</td><td rowspan="4">Encadr. GP (TL)</td></tr>
<tr><td>Intro projet (DV)</td></tr>
<tr><td>Cours GP (TL)</td></tr>
<tr><td>Encadr. GP (TL)</td><td>Cours GP (TL)</td></tr>
<tr><td rowspan="4">Après-midi</td><td rowspan="2">Cours GP (TL)</td><td rowspan="2">Autonomie</td><td rowspan="4">Autonomie</td><td rowspan="4">Autonomie</td><td rowspan="2">Autonomie</td></tr>
<tr><td rowspan="3">Encadr. GP (TL)</td></tr>
<tr><td>Encadr. BDA (DV)</td><td rowspan="5">Soutenance & Démo</td></tr>
<tr><td></td><td></td><td>Release 1</td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td></tr>
</table>

Tableau 2 : Emploi du temps de la semaine d'atelier

- ✓ **Intro projet :** Explication des différentes parties du projet et réponses aux questions par DV et TL
- ✓ **Cours GP :** Séances de cours magistraux (en anglais), données par TL
- ✓ **Encadr. GP :** Séances auxquelles les étudiants sont en travail autonome, mais avec la présence de TL pour l'encadrement technique du projet et de la gestion du projet
- ✓ **Encadr. BDA** Séances auxquelles les étudiants sont en travail autonome, mais avec la présence de DV pour l'encadrement technique de la partie BDA
- ✓ **Release 1 :** cf. 5. Description des livraisons attendues
- ✓ **Autonomie :** Séances auxquelles les étudiants travaillent entièrement en autonomie
- ✓ **Soutenance Démo. :** Soutenance du projet (la partie BDA) avec la démonstration du produit final (DV et TL)

5. Description des livraisons attendues

Tableau 2 résume les livraisons attendues.

Livraison	Destinataire	Date et heure	Format & Moyen
Rapport COO	TL	**Mardi 17 janvier, 23H**	PDF (email)
Release 1 (version prototype)	TL	**Mercredi 18 janvier, 15H45**	Démo. informelle par chef
Rapport GP	TL	**Jeudi 19 janvier, 23H**	PDF (email)
Rapport BDA	DV & TL	**Vendredi 20 janvier à 15H00**	PDF (email)
Présentation BDA (exposé)	DV & TL	**Vendredi 20 janvier 15H30 -**	Exposé avec slides (5 min)
Release finale (démo)	DV & TL	**Vendredi 20 janvier 15h30 -**	Démonstration produit (5 min)
Slides présentation BDA	DV &TL	**Vendredi 20 janvier 18H00**	PDF ou PPTX ou ODT (email)
Code source (release finale)	DV &TL	**Vendredi 20 janvier 18H00**	Fichier compressé .zip (email)

Tableau 2 Livraisons attendues

Pour tous les documents rendus :

- ✓ **Il ne faut surtout pas recopier inutilement les informations du présent cahier des charges dans vos documents à rendre.**
- ✓ **La police doit être « Candara » (disponible sous Word ou OpenOffice), de taille 11 (sauf les titres).**
- ✓ **Les marges des pages du corps du document doivent être 1,27 cm pour les quatre côtés.**
- ✓ **Chaque schéma / tableau dans le document doit être expliqué avec du texte.**

Rapport COO : Ce rapport documente la conception globale et de chaque partie du système, sauf la partie BDA. Vous devez utiliser les digrammes UML pour modéliser et documenter votre solution. Vous pouvez également utiliser d'autres formats de schéma « non UML », mais l'utilisation des digrammes UML doit être priorisée.

Release 1 (version prototype) : Cette release est informelle mais elle sera une évaluation certificative. Il n'y a pas de remise de code source. Le chef de projet présentera à l'enseignant la version BETA du système, avec des fonctionnalités restreintes.

Rapport GP : Ce rapport documente le déroulement et la gestion de votre projet. Il doit préciser les méthodes et techniques de gestion de projet que vous aurez pratiquées pour garantir la performance et la vélocité de l'équipe. Ce rapport doit aussi inclure vos analyses et rétrospectives approfondies de votre gestion de projet, les points à améliorer en vue de votre projet de synthèse.

Rapport BDA : Ce rapport documente la conception de la partie BDA du système, ainsi que l'étude comparative des deux types de plan.

Présentation BDA (exposé) : La présentation avec **slides** ne concerne que **la partie BDA.** Cette présentation ne devra pas dépasser 5 minutes et sera faite par les membres de l'équipe qui auront directement travaillé sur le développement de la partie BDA. Elle sera suivie par 5-10 minutes de questions / réponses sur la partie BDA.

Release finale (démo) : Les autres membres de l'équipe feront la démonstration du **produit final**. Cette démonstration ne doit pas dépasser 5 minutes. **Pas de manuel utilisateur à fournir.**

Slides présentation BDA : Les slides à envoyer par email aux enseignants.

Code source (release finale) : Le code source à envoyer par email aux enseignants doit respecter la convention du codage de Java d'Oracle. Le programme doit respecter au maximum les principes de base de COO vus en cours. Pour réduire la taille du fichier compressé du projet, vous ne mettrez pas les librairies externes. En cas de nécessité, ces dernières doivent être fournies dans votre email de remise du projet par un lien de téléchargement en ligne (partage de fichier).

Annexe : Ressources utiles pour le projet

- ✓ Support du cours Gestion de Projet
  https://depinfo.u-cergy.fr/~tliu/gp.php
- ✓ Support du cours Conception Orientée Objet
  https://depinfo.u-cergy.fr/~tliu/coo.php
- ✓ Support du cours Bases de Données Avancées
  https://depinfo.u-cergy.fr/~vodislav/Master/BDA/
- ✓ Un bon site de tutoriels pour bien débuter les frameworks JSF, avec beaucoup d'exemples et slides
  http://www.coreservlets.com/
- ✓ Documentation officielle des tags JSF
  http://docs.oracle.com/javaee/6/javaserverfaces/2.1/docs/vdldocs/facelets/
- ✓ Site officiel de Lucene
  http://lucene.apache.org/