Atelier de Gestion de Projet

Janvier 2025

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 sont formées pour pratiquer **le travail d'équipe « technique »**. Cet atelier est le premier atelier du Master IISC, qui sera suivi par des ateliers similaires en Master IISC Pro 2e année.

Les étudiants ont une semaine (5 jours) pour **gérer et réaliser** ce projet de A à Z. L'approche de gestion de projet est une adaptation de la méthode agile très connue : XP (*Extreme Programming*) vue en cours de Gestion de Projet. 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 des recherches d'informations de voyage et construire des 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. N'hésitez pas en ajouter d'autres.

Chaque équipe de projet réalisera le système en se basant sur une destination différente. Les 9 destinations à attribuer aux différentes 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 dans le cadre 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 pendant l'atelier et le rapport de gestion de projet.

Note COO dans le cadre 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 le rapport BDA (partie conception) et la démonstration du produit final.

Note BDA dans le cadre 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, le code source du projet, la soutenance BDA.

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

2. Exigences fonctionnelles

2.1 Organisation des informations 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 des lieux où on peut faire une activité de loisir particulière.
- ✓ 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.
- ✓ Pour l'hébergement, comme les destinations sont des îles, on s'intéresse uniquement aux hôtels au bord de la mer. Donc chaque hôtel inclut systématiquement une plage.
- ✓ On ne va pas seulement rester allongé sur la plage… Il faut organiser les excursions pour aller aux sites touristiques (historiques ou activités) pendant le séjour. Chaque excursion est composée d'une série de sites touristiques et de transports (hôtel → site, (plusieurs fois et 3 sites au maximum par jour) site → site, site → hôtel).
- ✓ On suppose qu'il y a une seule excursion au maximum par jour. Pour certains jours, on peut ne pas planifier d'excursion, si le client souhaite un séjour moins intensif.
- ✓ Les hôtels ont différentes gammes donc différents prix. Les clients pourraient se loger à plusieurs hôtels pendant le séjour, donc il faut aussi inclure les trajets entre hôtels dans le contenu de l'offre.
- ✓ On prend en considération les distances entre hôtel → site, site → site et site → hôtel. Ces distances peuvent être soit calculées, soit prédéfinies à l'avance.
- ✓ 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. Les moyens de transport possibles pour les excursions planifiées (hôtel → site, (plusieurs fois) site → site et site → hôtel) sont à pied, l'autobus et le bateau.
- ✓ Les transports ont des prix prédéfinis qui seront pris en compte dans le calcul du prix total des offres.

2.2 Recherche simple des informations de voyage

Le système fournit des **fonctionnalités de simples recherches** des informations de voyage (surtout sites et hôtels) mentionnées dans **2.1**. Par exemple, trouver des sites dont la description contient un mot clé donné. On peut effectuer des recherches avec des critères pour lesquels on peut préciser les valeurs voulues. Cette fonctionnalité à développer vous facilitera les tests unitaires et sera utile pour développer **2.3**.

2.3 Construction automatique des offres de séjour

La fonctionnalité la plus importante du système est de construire automatiquement (intelligemment) des offres, à partir des critères spécifiés par l'utilisateur. Le système doit pouvoir traiter n'importe quel sous-ensemble des critères suivants (au moins ceux-là) :

- ✓ 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 par l'utilisateur et puis le système fait le traitement « intelligent » pour construire **plusieurs offres de séjour possibles**.

Chaque offre de séjour contiendra un ensemble d'excursions planifiées et le (les) hôtel(s) proposé(s) pour le séjour. Le système affichera les informations complètes des offres construites sur l'interface graphique (pages Web).

3. Exigences techniques

Les exigences techniques du projet concernent les deux unités d'enseignement COO et BDA. Concrètement, l'évaluation de la partie COO inclut la conception et implémentation globale du projet, et celle de la partie BDA inclut les exigences spécifiques de la réalisation de la partie « base de données relationnelle étendue » du système.

3.1 Partie Conception Orientée Objet (COO)

Architecture de l'application

Il s'agit d'une application en architecture de 5-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.

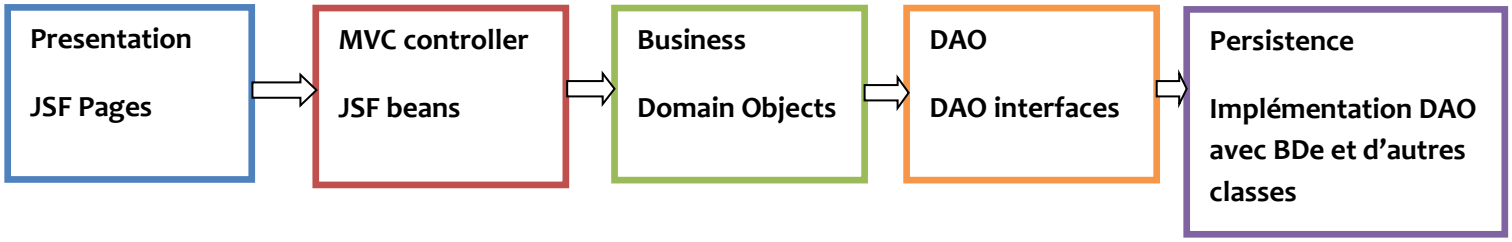

Figure 1 Architecture en 5-tiers

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 *Managed Beans*) 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).

Attention, inévitablement, certains *Domain objects* sont nécessairement utilisées dans les autres couches, donc leur conception a besoin de grands soins.

Couche DAO

Dans cette couche, vous définissez un ensemble d'interface contenant des méthodes abstraites. Ces méthodes répondent aux besoins d'accès aux données (y compris différentes opérations, calculs sur ces données). Elles sont utilisées par la couche métier et sont implémentées par la couche persistance.

Couche persistance

Cette couche contient la Base de Données étendue (BDe) que vous allez concevoir et réaliser (cf. la partie BDA), et d'autres classes nécessaires pour implémenter les interfaces DAO.

A faire

- ✓ Modéliser et implémenter les classes métiers (*Business Objects*)
- ✓ Conception des interfaces DAO
- ✓ Conception de la Couche persistance avec la partie BDA
- ✓ Implémentation (programmation) de votre conception
- ✓ Utiliser Spring **IoC** pour gérer les objets des classes importantes et complexes.
- ✓ Réaliser les tests unitaires (automatisés ou non). Vous devez parfois « simuler » (*mock*) temporairement les entrées/sorties pour les API quand les parties concernées ne sont pas encore développées par vos collègues
- ✓ Préparer les données / fichiers (réelles ou fictives) pour les différents tests du système
- ✓ Intégration des différentes couches du système
- ✓ Conception des pages JSF (Couche présentation) avec les beans (Couche contrôleur MVC)

Bonus

- ✓ L'utilisation de Spring AOP n'est pas obligatoire mais serait un bonus en cas d'une solution appropriée et intéressante d'AOP.
- ✓ **Indépendamment** du travail demandé de la partie BDA, vous pourrez réaliser une sauvegarde avec Hibernate des historiques des offres construites correspondantes aux critères utilisés.

3.2 Partie 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 la description de cette syntaxe ici :https://lucene.apache.org/core/10_1_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 en principe plus pertinent qu'un texte 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*. Comme le lien entre le résultat SQL et le résultat textuel se fait à travers la clé de T, il faudra donc rajouter la clé de T dans la clause **select** de la partie SQL de la requête si elle n'y est pas déjà. Les résultats finaux 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*** (donc dans l'ordre fourni par la requête textuelle).

   Hypothèses : Nous considérons que les résultats d'une requête SQL sont potentiellement trop volumineux pour être gardés en mémoire, tandis que ceux des requêtes textuelles sont de taille raisonnable (ensembles de couples clé-score) et peuvent être stockés en mémoire. Nous considérons également que les résultats finaux des requêtes mixtes peuvent être stockés en mémoire.

   Remarque : La composition des résultats des deux requêtes dans le sens inverse (en prenant un par un les résultats de la requête textuelle et en vérifiant si la clé apparait dans le résultat de la partie SQL) aurait l'avantage de donner les résultats dans le bon ordre, mais nécessiterait la ré-exécution de la partie SQL pour chaque résultat de la partie textuelle (les résultats SQL ne pouvant pas être gardés en mémoire, conformément à l'hypothèse ci-dessus).

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 les opérateurs suivants :

- ✓ 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 de pertinence, et sont produits en ordre décroissant du score.
- ✓ Un opérateur de jointure entre les résultats SQL et textuels pour le premier type de plan d'exécution.
- ✓ Tout autre opérateur que vous jugerez nécessaire pour ce plan d'exécution.

Le plan d'exécution est un arbre d'opérateurs, dont la racine produit les résultats de la requête. Chaque opérateur qui peut fonctionner en mode « pipleline » sera implémenté sous la forme d'un itérateur qui fournit deux méthodes : *init*() et *next*(). Il faut également que le plan d'exécution offre les mêmes deux opérateurs, *init*() et *next*(), qui permettent de produire le résultat finale élément par élément.

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

- ✓ Le fonctionnement détaillé des opérateurs des deux types de plans.
- ✓ Un comparatif des temps 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. Les hypothèses supplémentaires suivantes sont considérées :

- Le temps d'exécution d'une requête SQL dans le premier plan est à peu près le même que le coût d'une requête textuelle, car cette dernière utilise un index construit sur disque.
- Le temps d'exécution d'une requête SQL du deuxième plan est *k* fois plus faible que celui de la requête SQL du premier plan, car elle peut utiliser un index sur la clé.

✓ Toute discussion argumentée sur des améliorations possibles dans les plans d'exécution pour diminuer le temps d'exécution dans les différents cas de figure.

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 le squelette de programme fourni et la documentation générale https://lucene.apache.org/core/10_1_0/index.html)
✓ L'étude comparative des deux types de plans

4. Exigences organisationnelles

Pendant l'atelier, la présence de tous les membres de l'équipe est bien obligatoire.

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

	Lundi 27/01	Mardi 28/01	Mercredi 29/01	Jeudi 30/01	Vendredi 31/01
Matin	**Intro projet (TL) 9H30** **Intro projet (DV) 10H**	**Autonomie**	**Autonomie**	**Autonomie**	**Autonomie**
Après-midi	**Autonomie** **Q / R BDA (DV) à 16H**	**Autonomie**	**Point d'avancement (TL) 13H** **Autonomie**	**Autonomie**	**Rendu final du projet** **Présentation BDA et Démo finale, (TL + DV)**

Tableau 2 : Emploi du temps des jours de l'atelier

Légende

✓ **Intro projet** : Explication des différentes parties du projet et réponses aux questions.
✓ **Q / R BDA** Une séance de questions / réponses sur la partie BDA
✓ **Point d'avancement** Il y aura un ordre de passage pour chaque équipe.
✓ **Autonomie** : Séances auxquelles les étudiants travaillent entièrement en autonomie, vous pouvez poser des questions à TL **via Discord** en cas de blocage, qui va vous répondre dans les plus brefs délais.

5. Description des livraisons attendues

Tableau 1 résume les livraisons attendues.

Livraison	Destinataires	Date et heure	Format & Moyen
Point d'avancement	TL	Mercredi 29 janvier 2025 13H00- Bureau A485	Démo des fonctionnalités réalisées (chef de projet)
Rapport BDA	DV & TL	Vendredi 31 janvier 2025 à 12H00	PDF (email)
Rapport COO	TL	Vendredi 31 janvier 2025 à 12H00	PDF (email)
Slides présentation BDA	DV & TL	Vendredi 31 janvier 2025 à 12H00	PDF (email)
Code source (release finale)	DV & TL	Vendredi 31 janvier 2025 à 12H00	Fichier compressé .zip (email)
Présentation BDA (exposé)	DV & TL	Vendredi 31 janvier 2025 13H00 -	Exposé avec slides (4-5 min)
Démonstration finale	DV & TL	Vendredi 31 janvier 2025 13H00 -	Démonstration produit (5 min)
Rapport GP	TL	Vendredi 31 janvier 2025 à **17H**	PDF (email)

Tableau 1 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 taille de police du corps (sauf les titres) de vos documents doit être 11.
- ✓ Les marges des pages du corps du document doivent être 2 cm exacte pour les quatre côtés.
- ✓ Chaque schéma / tableau dans les documents doit être accompagné par de l'explication textuelle.

Rapport BDA : Ce rapport de 4-5 pages documente la conception de la partie BDA du système, API, diagramme de classes, modèles de données, etc., ainsi que l'étude comparative des deux types de plan.

Rapport COO : Ce rapport de 4-5 pages documente la conception globale et de chaque partie du système, sauf la partie BDA. Vous devrez également bien expliquer l'algorithme de construction des offres. Vous devez utiliser les digrammes UML pour modéliser et documenter votre solution. Vous pouvez également utiliser d'autres formats de schéma, mais l'utilisation des digrammes UML est préconisée.

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 / appliquer correctement les aspects techniques vus en cours / TD COO.

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

Présentation BDA (exposé) : La présentation avec **slides** de la **partie BDA**. Cette présentation ne devra pas dépasser 5 minutes et pourrait être réalisée uniquement par les membres de l'équipe qui auront directement travaillé sur le développement de la partie BDA.

Démonstration finale : Cette démonstration du produit final ne doit pas dépasser 5 minutes. **Pas de manuel utilisateur à fournir.**

Rapport GP : Ce rapport de 4-5 pages 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 d'une meilleure gestion de votre projet de synthèse.

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/
- ✓ Documentation officielle des tags JSF
  http://docs.oracle.com/javaee/6/javaserverfaces/2.1/docs/vdldocs/facelets/
- ✓ Démonstrations PrimeFaces
  https://www.primefaces.org/showcase/
- ✓ Site officiel de Lucene
  http://lucene.apache.org/
- ✓ l'API Lucene
  http://lucene.apache.org/core/10_1_0/index.html