Atelier de gestion de projet

Rapport de projet : COO

~

Système intelligent de création d'offres de voyage
(Les Petites Antilles)

Auteurs

Valentin BELYN
Vincent ARCHAMBAULT
Arnaud SERY
Benoit CONS
Thomas RE
Loïc TRAMIS

Université de Cergy-Pontoise
Master 1 IISC 2018-2019

Remise du rapport le 18/01/2019
UE : AGP

SOMMAIRE

TABLE DES ILLUSTRATIONS

1. Introduction

Dans le cadre de l'atelier de gestion de projet organisé lors de la semaine du 14 au 18 janvier 2019, nous devions concevoir et implémenter un système permettant de générer des offres de voyages en fonction de paramètres fournis par l'utilisateur.

Cette rapport traite de la conception des couches métier (business), présentation et contrôleur MVC. La partie liée à la couche persistance des données est annexée dans un second compte-rendu (BDA).

2. Fonctionnalités

Le système dispose de deux fonctionnalités principales pour l'utilisateur :

- La recherche d'informations simples sur des lieux en fonction de critères comme le type de lieu ou encore des mots clés.
- La construction automatique d'offres de séjour en fonction de critères communiqués par le client. Les critères sont : le type de lieux (activité ou lieu historique, ou les deux), des mots clés présents dans la description des lieux, la prix minimum et maximum, le nombre de jour de voyage, la fréquence des excursions (tous les jours, tous les deux jours, etc.) et le temps maximal de transport entre chaque visite. Une fois la demande traitée, la couche métier (business) renvoie à la couche vue (web/JSF) les trois offres les plus cohérentes avec les critères de l'utilisateur.

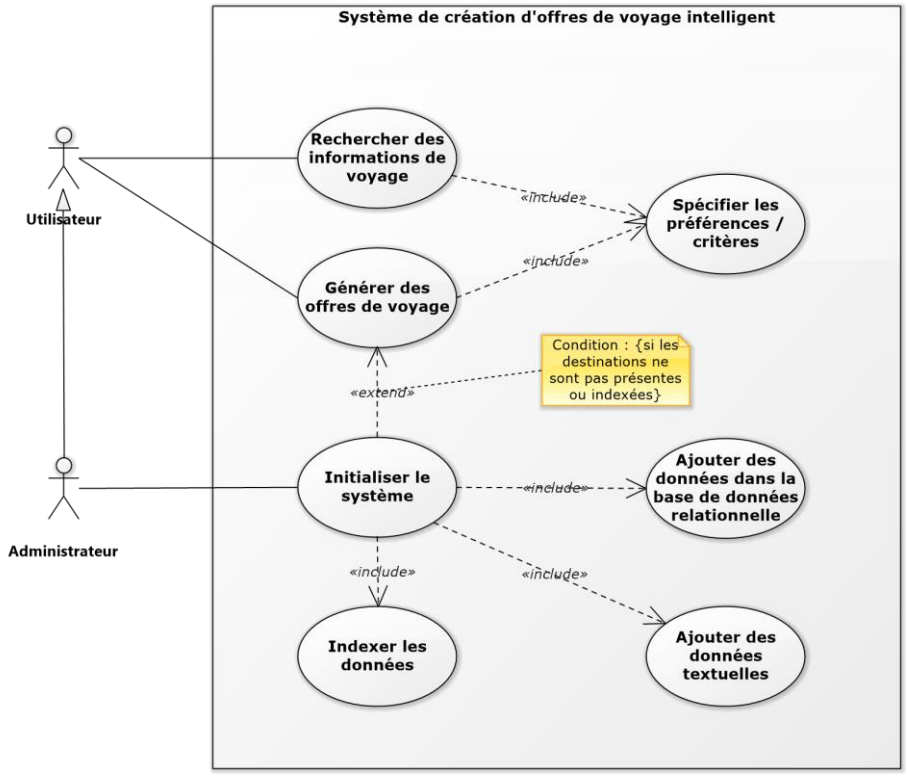

Figure 1 Diagramme de cas d'utilisation du système

3. Fonctionnement de l'application

3.1 Diagramme de classes

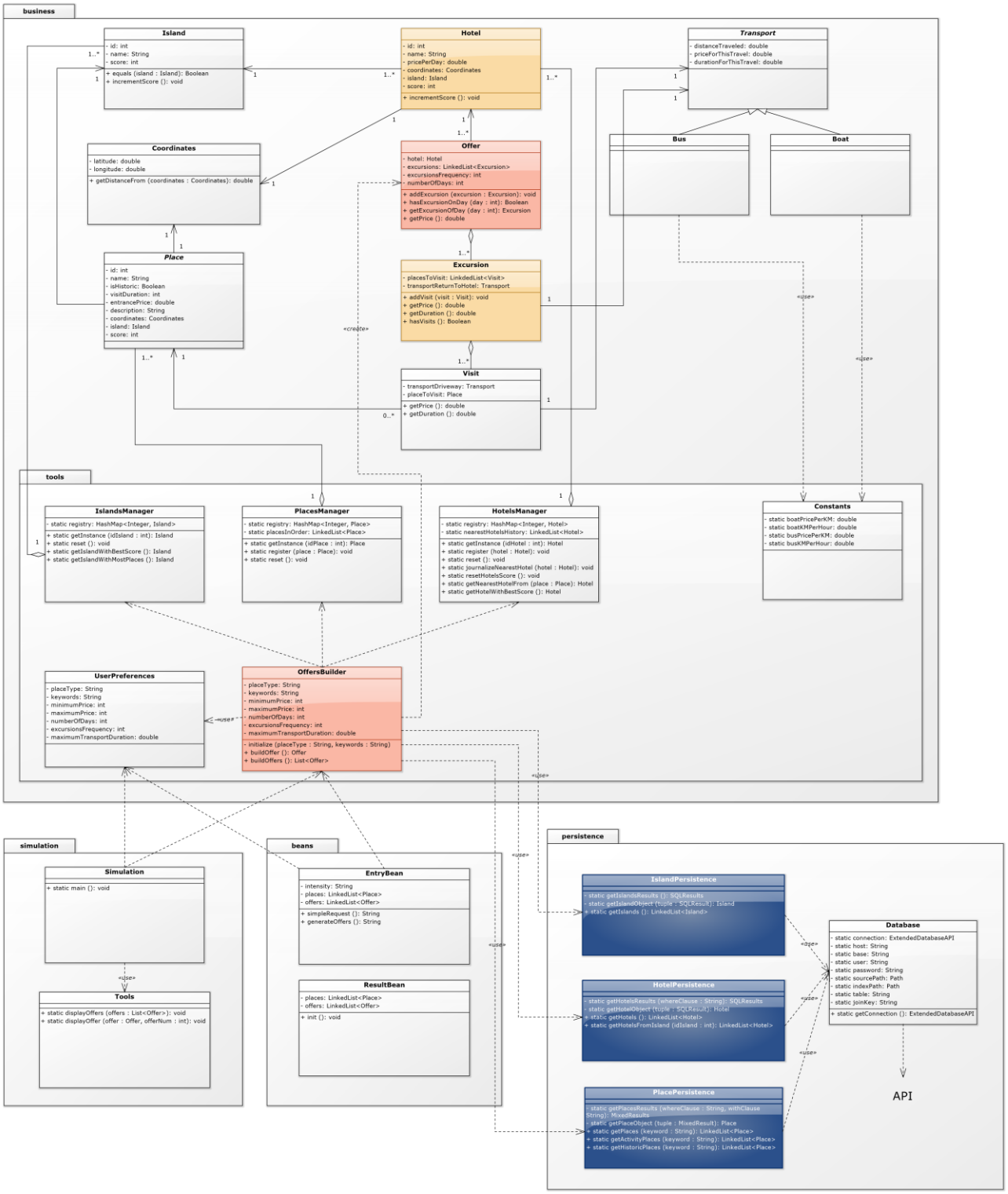

Figure 2 Diagramme de classes de l'application

La couche persistance a été simplifiée pour ce rapport. Elle est plus détaillée dans le rapport BDA.

Notre application est composée des couches/packages suivants :

- business : contient nos classes métier.
- persistence : contient les classes permettant d'interroger les bases de données et de retourner des objets de la couche métier prêts à être utilisés.
- beans : contient les classes utilisées comme contrôleur dans le modèle MVC. Elle fait le lien entre la vue (présentation) et la couche métier.
- présentation : contient les pages web JSF.
- simulation : contient les classes utilisées pour générer les offres en mode console.
- tests : contient les tests JUnit et non JUnit.

3.2 Design patterns

Dans le but de centraliser la gestion des données, les classes IslandsManager, PlacesManager et HotelsManager utilisent le pattern Singleton en mode register. Ces disposent d'opérations pour calculer les scores des différents points d'intérêt.

Des patrons Façade sont employés au niveau de la partie persistence (ExtendedDatabaseAPI, Searcher, SQLSearcher). Les classes de la couche persistance utilisent aussi des patterns Singleton en mode Lazy.

Le patron Builder est utilisé pour la génération des offres. Il s'agit de la classe OffersBuilder.

Enfin, le pattern Strategy est employé pour ajuster le prix et la durée des transports (au niveau de la classe Transport et de ses sous-classes).

3.3 Processus de générations des offres

Dans le but de trouver les meilleurs lieux à visiter pour l'utilisateur en fonction de ses préférences, nous avons mis un système de score. Chaque île, hôtel et lieu a un score.

Dans un premier temps, nous recherchons tous les lieux correspondant aux critères du client. Nous analysons ensuite la répartition de ces lieux au sein des différentes îles. L'île « gagnante » est celle ayant le plus de lieux.

Nous calculons, pour chacun des hôtels de cette île, les distances qui séparent les hôtels des lieux. L'hôtel étant le mieux placé pour accéder aux destinations est élu.

Lors de la génération des offres, nous trouvons les places les plus pertinentes grâce à leur score donné par Lucene. Si le lieu touristique ayant le score le plus élevé est trop éloigné de la destination courante (qui est l'hôtel au début de l'excursion), une autre destination est prise (en respectant l'ordre des scores).

Un budget est défini pour chaque jour. Cela est donné par un calcul avec les valeurs renseignées par l'utilisateur. Si un jour il reste de l'argent non dépensé, il est reporté au lendemain. Il se cumulera ainsi au budget défini le jour suivant. Nous avons ainsi une bonne gestion de l'argent du client.

Les offres varient entre-elles par le fait qu'elles n'ont pas le même hôtel et par conséquent, les visites varient d'une offre à l'autre car les distances évoluent.

Pour tous les calculs, nous prenons de plus en compte la vitesse des transports et leur prix afin d'obtenir des offres plus précises. Le transport est choisi en fonction de la situation. Par exemple, s'il faut changer d'île pour accéder à une destination, seul un bateau pourra être pris.

4. Tests unitaires

Les tests unitaires ont été faits sur JUnit 5 et ont un total de recouvrement de 97.7% sur la partie COO. Ils ont tous été faits au fur et à mesure de la conception durant les 4 premiers jours de conception de la partie COO.

Après chaque commit, des tests ont été faits pour s'assurer du bon fonctionnement de chaque partie du projet afin de détecter au plus tôt toutes les anomalies possibles qui ralentiraient notre avancement. Tous les tests ont donc été faits en white box.

5. Annexes

5.1 Diagramme de classe (version complète)

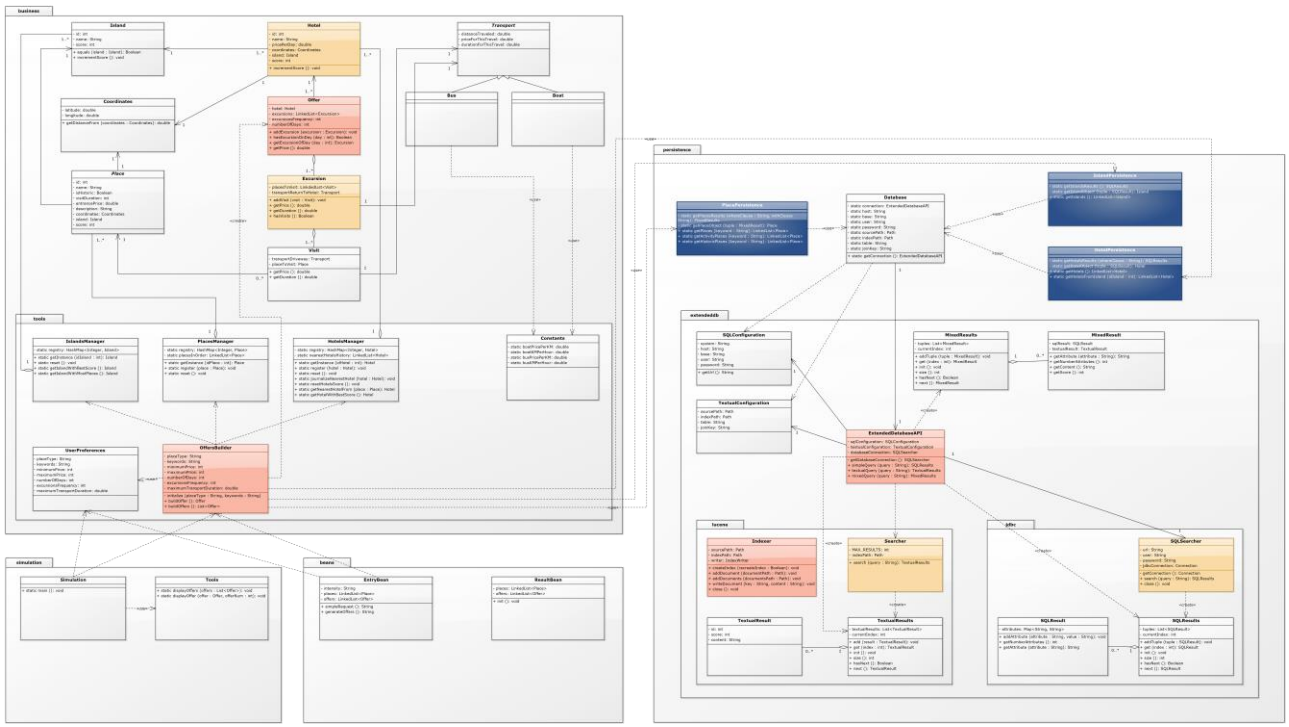

Figure 3 Diagramme de classes (version complète)