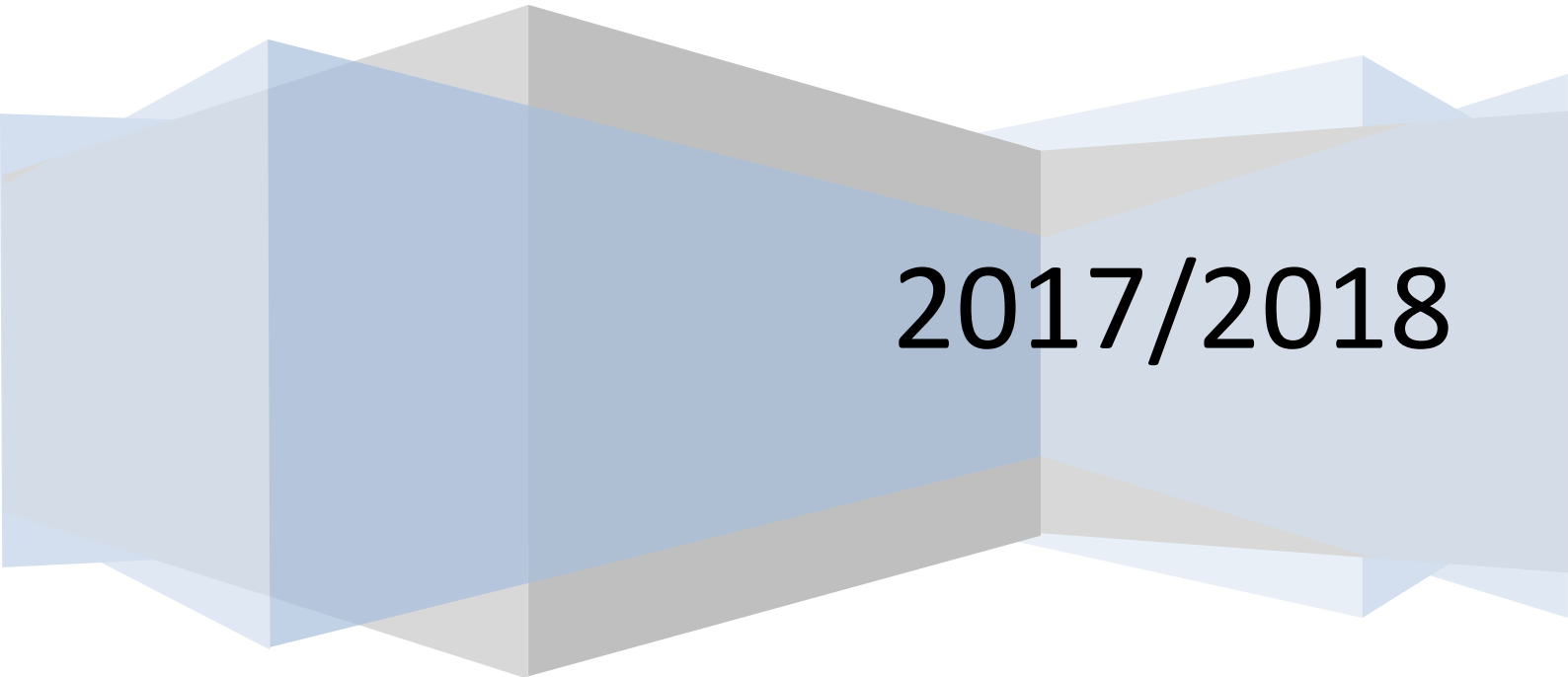


Rapport de synthèse

Gestion d'un Hôtel

17/05/2017



2017/2018

I. Introduction :

L'application gestion hôtel permet de gérer les réservations de chambres et de salle de réunions par un personnel de l'hôtel.

Notre programme java se compose de plusieurs package qui organisent mieux notre enchaînement. Tout d'abord, on a le package connexion qui nous permet de se connecter à la base de donnée de la même méthode vu en TP.

Le package 'fiches' contient plusieurs classes responsable de la création de fiche client et fiche d'entreprise lors de la réservation. Puis qui permet l'affichage de ces derniers dans l'interface. On a également une classe qui rajoute à chaque client des points de fidélités après chaque réservation.

Le package 'facture' qui permet de calculer la facture pour une chambre ou une salle de réunion et l'afficher, tout en prenant en compte les options choisies par le client.

On a séparé la réservation en deux panneaux complémentaires, chambre et salle, qui se rejoignent aux mêmes fonctionnalités.

Finalement, on a le package 'Calendrier' qui contient le panneau qui affiche notre calendrier, permet de sélectionner une date, et ouvrir un autre panneau où on affiche le jour sélectionné ainsi que les chambres occupées ce jour là.

II. Méthode de travail adoptée :

1. Analyse

En premier lieu, nous avons commencé par énumérer les objectifs de notre projet "Gestion d'un hôtel" ce qui a fait l'objet de notre premier rapport d'analyse.

Nous devons donc créer un programme permettant au personnel d'un hôtel de gérer les chambres de ce dernier.

Principalement le personnel doit pouvoir :

- Réserver des chambres tout en ajoutant les options choisies par le client,
- Voir sur le planning des réservations existantes, ajouter un client dans la base de données,
- Créer une facture de la réservation
- Comptabiliser les points de fidélités de chaque client qui aboutira à des réductions.

Puis, nous avons décidé de gérer également à la gestion des salles de réunion. Car celui-ci n'était pas très différent du premier.

Nous voulions pouvoir créer des réservations d'une salle de réunion par une entreprise, donc a priori créer des fiches entreprises dans la base de donnée et ainsi créer la facture associé à la réservation.

En vue des objectifs que nous nous étions fixés, nous pensions que faire une interface graphique serait bien malgré le fait que nous n'en n'avions jamais fait.

Nous avons commencé par faire le diagramme de cas d'utilisation qui était le diagramme le plus simple à faire. Puis on a enchaîné avec le diagramme de classe qui nous a pris plus de temps pour le réaliser du fait qu'il englobe toutes les fonctionnalités, toutes les variables et leurs types.

On a modifié ce diagramme plusieurs fois au cours du projet, on se rendait compte à chaque séance qu'il fallait modifier le diagramme en rajoutant et/ou supprimant une fonction et/ou variable.

Entre temps nous avons réalisé le diagramme d'activité et le diagramme de séquence tout en agrémentant notre rapport d'analyse.

On a réalisé tout les diagrammes sur papier, tout en s'assurant de l'exactitude des diagrammes par nos encadrant.

2. Développement

Après avoir fini la partie analyse de ce projet, on s'est attaqué à la partie développement. Nous avons commencé le mardi 10 avril. Nous nous sommes partagées le travail en deux pour qu'on soit plus efficace. L'une a commencé de créer la base de données sur Pgadmin tandis que l'autre a commencé la création des classes sur le logiciel UMLstar qu'on avait réalisé auparavant sur papier.

On s'est appuyé sur le diagramme de classe pour créer les tables de notre base de données, qui fut rapide, car notre diagramme était complet. Les tables sont exactement les mêmes que les classes sur le diagramme de classe.

Pour ce qui est des colonnes nous avons ajouté quelques unes qui ne sont pas dans le diagramme de classe, principalement pour simplifier notre codage.

Pour ce qui est des classes nous avons commencé par créer toutes les classes présentent dans notre diagramme ainsi que les constructeur et fonctions apparentés. Les fonctions étaient vident pour le moment.

On a commencé à créer le corps du programme en créant les classes (client , personnel, facture, points de fidélités, réservation chambre, réservation salle, chambre, salle de réunion ..) , et en important les fonctions associées à chaque classe. Et ainsi créer une classe spécifique pour connecter le programme à la base de données, comme ce qu'on avait vu pendant les séances du TP.

Puis on a commencé à se renseigner sur comment créer une interface graphique.

C'est à ce moment que nous nous sommes rendu compte de notre erreur.

Nos classes créant l'interface graphique ne font pas appel à celles des fonctions déjà créés. Elles appellent directement la base de donnée, crée et modifie celle-ci.

Nous avons donc abandonné notre première idée et avons décidé de tout faire à partir de notre interface graphique.

Nous avons du supprimer plusieurs classes et créer d'autres avec le nom qui comporte le nom des fonctions directement.

3. Interface graphique

La création de l'interface graphique s'est avéré très compliquée et nous as pris beaucoup de temps à réaliser, car en cours nous avons vu uniquement comment créer une fenêtre et éventuellement ajouter une image.

Premièrement nous voulions que le personnel s'identifie avant de pouvoir accéder au panneau principal. Nous avons donc créé deux identifiants avec leurs mots de passe respectifs. Ces identifiants sont accessible dans le fichier README associé à notre projet.

Nous avons appris comment créer une interface graphique avec des boutons, la complexité revient à créer chaque bouton séparément avec le texte qui sera affiché dessus et définir les paramètres du bouton, ainsi qu'à identifier la fonction qui devra être vérifiée.

La suite fut un peu plus facile à implémenter, à part la création du planning qui nous à poser beaucoup de difficulté car on devrait créer un calendrier contenant plusieurs boutons avec les jours, mois sur plusieurs années. On a eu également du mal à ajouter le prix des options lors de la facturation. Le programme nous affichait que le prix de la chambre sans rajouter le prix des options.

On a cherché un peu sur internet et on a trouvé beaucoup de programmes mais qui ne répondaient pas à notre besoin spécifique qui était de sélectionner une date précise et afficher les chambres qui ont été réservés à cette date, pour pouvoir finalement afficher les chambres prises ou libres à une date donnée. On s'est basée sur un programme qu'on a du modifier a plusieurs reprises pour aboutir à notre objectif.

On a également pu réaliser une réduction affiché dans la facture en cas de périodes creuses. On peut améliorer le design de l'interface graphique selon les goûts et les besoins. Finalement, on a abouti à l'interface ci-dessous :



Figure 1: Interface application 'Gestion Hotel'

III. Conclusion :

Au niveau de la modélisation UML, ce projet nous a permis de revoir tout les types de diagrammes, en s'attardant sur le diagramme de classe qui était le plus globale.

La partie développement de notre projet s'est révélée très enrichissante même si elle était plus compliquée que la première partie. Car on a du utiliser plusieurs notions qu'on a jamais vu auparavant. On s'est référé plusieurs fois à Internet pour s'inspirer.

Cette partie nous a permis de développer nos connaissances en programmation orientée objet. Et de mieux comprendre l'usage pratique de ce dernier. Notamment pour la création des interfaces.

On aurait pu créer une classe qui collecte et stock les données de la BDD afin de pouvoir les utiliser lors de l'exécution d'une commande et non faire appelle à la base de données directement. Ceci n'améliore pas le programme mais correspondrait plus aux objectifs demandé pour ce projet.

IV. Référence :

[1] : <https://docs.oracle.com/javase/tutorial/uiswing/layout/gridbag.html>

[2] : <https://openclassrooms.com/courses/apprenez-a-programmer-en-java>

[3] : https://www.jmdoudoux.fr/java/dej/chap-utilisation_dates.htm

[4] : <https://codes-sources.commentcamarche.net/source/101760-calendrier>