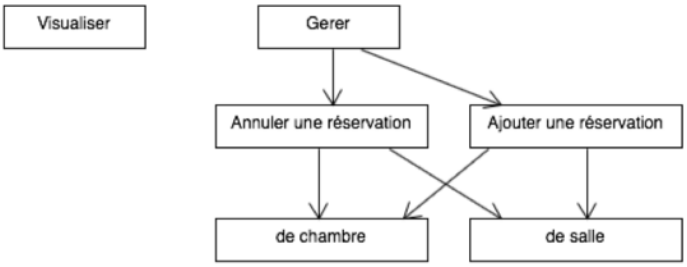
Le Berrigaud Axel
Payoux Manon

Gestion d'un hôtel

Le but de notre travail était de faire une application utilisable par les employés d'un hôtel pour ajouter des réservations, ajouter de nouveau clients, supprimer des réservations, gérer les comptes fidélités des clients et visualiser le planning des réservations.

I Organisation et fonctionnement des développements

Actions réalisables par l'employé :



Décomposition par étape lorsque l'employé veut *visualiser le planning des réservations.*

1)Le programme a besoin d'une date de début et d'une date de fin pour créer le planning.
La méthode listeDesDates() de la classe Réservation prend en paramètre ces deux dates et retourne une liste de toutes les dates entre ces deux dates, dates de début et de fin comprises.

2) Ces dates sont placées dans l'entête d'une JTable.

3)Les lignes du tableau sont construites en extrayant de la base de données le nombre de chambre simples, doubles et composées réservées à chacune des dates.
(voir partie 2 lors un exemple d'exécution de visualisation d'un planning.)

Décomposition par étape lorsque l'employé veut *créer une réservation.*

1) le programme regarde si le client est dans le base de données ou non.
Si il n'y ait pas il est ajouté. Le compte fidélité associé au client est également crée. Le client et son compte fidélité ont le même id.

2) le programme demande le nombre de personnes et le type de réservation (« chambre » ou « salle »).

Si le client veut réserver une salle de réunion	Si le client veut réserver une chambre
3)On demande au client des informations sur la réservation	3)On demande au client des informations sur la réservation
4) le programme regarde les salles indisponibles pour la réservation en fonction du nombre de personnes et du créneau horaire avec la fonction verifierSallesNonDispo() de la classe ReservationSalleReunion.	4) le programme regarde les chambres disponibles pour la réservation en fonction de la période et du nombre de chambre de chaque type demandé avec la fonction verifierChambreNonDispo() de la classe ReservationChambre.
5) le programme affecte une salle à la réservation si après l'étape précédente toutes les salles ne sont pas indisponibles. Le statut de la réservation passe en alors « validée ». Cela se fait avec la fonction affecterSalleToReser() de la classe ReservationSalleReunion.	5) le programme affecte des chambres à la réservation si le bon nombre de chambres est disponibles. Le statut de la réservation passe alors en « validée ». Cela se fait avec la fonction affecterChambreToReservation.
6) qu'elle soit validée ou non, la réservation est ajouté à la base de données. Si elle est validée, l'id de la salle réservée est ajouté à la table reservation_sr comme clé étrangère de la table salle_reunion. Méthode ajouterReservation() de la classe ReservationSalleReunion.	6) qu'elle soit validée ou non, la réservation est ajouté à la base de données. Si elle est validée, les id des chambres réservées sont inserés dans la table reservation_chambre sous forme d'une chaine de caractère. Une colonne pour les id de chaque type de chambre est prévue dans la base de données.
Si la réservation est validée 7) Le prix de la réservation est calculée en fonction du prix/heure de la réservation d'une salle (qui est le même pour toutes les salles) et du nombre de points de fidélité du client par les méthodes calculprixreservation (), calculremisefidelite() et calculprixTotal() de la classe FactureReservSalleReunion.	Si la réservation est validée 7) Le prix de la réservation est calculée en fonction du prix de chaque chambre, du nombre de jour et de la période de haute saison avec la méthode calculPrixReservation(), le prix des options est calculée par la méthode calculPrixOption(). La remise est calculée par la méthode calculRemiseFidelite(). Le prix total par calculPrixTotal(). Ces méthodes se trouvent dans la classe FactureReservChambre.
8)Les montants du prix de la réservation, de la remise fidélité et du prix total sont ajoutés dans la table facture. Méthode ajouterFactureBDD() de la classe FactureReservSalleReunion. L'id de la facture est le même que celui de la réservation.	8)Les montants du prix de la réservation, des options, de la remise fidélité et du prix total sont ajoutés dans la table facture par la méthode ajouterFactureBDD() de la classe FactureReservChambre. L'id de la facture est le même que celui de la réservation.
9) Le nombre de points fidélité utilisé pour la remise est retiré du nombre de points fidélité du client et le nombre de points gagné par le client avec sa facture est ajouté au nombre de points fidélité du client. Le nombre de points fidélité est mis à jour 2fois en utilisant les méthodes setNbPoints() de la classe CompteFIdelite et setPointFidelite() de la classe FactureReservSalleReunion.	9) Le nombre de points fidélité utilisé pour la remise est retiré du nombre de points fidélité du client et le nombre de points gagné par le client avec sa facture est ajouté au nombre de points fidélité du client. Le nombre de points fidélité est mis à jour 2fois en utilisant les méthodes setNbPoints() de la classe CompteFidelite et setPointFidelite() de la classe FactureReservChambre.

A noter : Dans un hôtel le prix est donné en fonction du nombre de nuit, ici on considèrera que la nombre de jour est égale au nombre de nuit.
Le client peut réserver qu'une seule salle de réunion à la fois et le créneau doit être sur un seul jour (ne pas chevaucher deux jours différents).

Décomposition par étapes lorsque l'employé veut *annuler une réservation*

1) on demande à l'employé si il veut supprimer une réservation de chambre ou de salle, cette donnée est stockée dans une variable type.

2) avant de savoir si l'employé veut réserver ou annuler le programme lui a déjà demandé le nom et le prénom du client. A partir de ces informations l'id du client va être retrouvé (méthode getId() de la classe Client).

3) La réservation de ce type du client et la facture sont supprimées de la base de données en faisant une requête sql sur la table réservation concerné et l'id du client. On se base sur l'hypothèse que la client a effectué une seule réservation de ce type (sinon toutes ses réservations sont supprimées).

II. Exécution du programme

Vous pouvez choisir de lancer 2 mains : celui de la classe MainHotel ou celui de la classe InterfaceGraphique.
Avant tout, il faut veiller à récupérer la base de données à partir du fichier.backup.
Attention dans la classe Connexion il faut rentrer le mot de passe pour accéder au serveur sur lequel vous avez mis la base de données.

Lors de l'éxécution du main de la classe MainHotel, une série de question est posée à l'employé dans la console.

Dans la console après une réservation de chambre faite pour un nouveau client :

```
Vous désirez visualiser l'état des réservations ou gérer les réservations ?
gerer
Quel est le nom du client ?
payoux
Quel est le prenom du client ?
manon
Quel est le tel du client ?
0634343434
Quel est le mail du client ?
manon@ensg.eu
le client n'est pas connu dans la BDD
le nouveau client a bien été ajouté à la base de données ainsi que son compte fidelité
le client a été retrouvé dans la base de données
le client veut annuler ou reserver?
reserver
Pour combien de personnes?
1
Quelle est le type de la reservation
chambre
A partir de quelle date?
2018-06-01
Jusqu'à quelle date?
2018-06-03
Combien de chambres simples?
1
```

```
Combien de chambres doubles?
0
Combien de chambres composees?
0
Le client veut-il des options?
oui
Combien de petit déjeuner dans la chambre?
2
Combien de petit dejeuner au restaurant?
1
Combien de dejeuner dans la chambre?
0
Combien de dejeuner au restaurant?
0
Combien de diner dans la chambre?
0
Combien de diner au restaurant?
0
Combien de séances de spa?
0
```

```
la demande de reservation a été validée !
Les chambres ont été attribuées à la reservation/les chambres que les clients vont occupées sont la/les chambre(s) Rose;
Le nouveau nombre de points fidelité a été mis à jour !
Le client va payé314.0euros
Le nouveau nombre de points fidelité a été mis à jour !
```

Dans la base de données :

Dans la table reservation_chambre

	datdebutsejour date	datefinsejour date	nbpersonnes integer	nbchambressimple integer	nbchambrescomposees integer	nbchambresdoubles integer	idclient integer
1	2018-06-01	2018-06-03	1	1	0	0	1

id integer	statut character varying	idchambressimplesreservees character varying	idchambresdoublesreservees character varying	idchambrescomposeesreservees character varying
1	validée	1;		

listoptions character varying
;petit déjeuner;petit ...

Dans la table facture

	id integer	idreserv integer	prix real	remise integer	prix_options real	total real
1	1	1	360	0	46	314

Dans la table client

	id integer	nom character varying	prenom character varying	telephone integer	adressemail character varying
1	1	payoux	manon	634343434	manon@ensg.eu

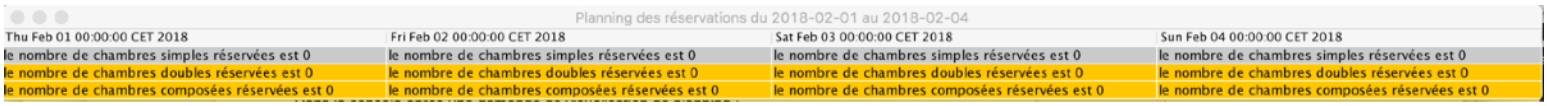
Dans la table compte_fidelite

	id integer	nbpointfidelite integer	idclient integer
1	1	31	1

Dans la console après une demande de visualisation de planning :

```
Vous désirez visualiser l'état des réservations ou gérer les réservations ?
visualiser
A partir de quelle date ?
2018-02-01
Juqu'à quelle date?
2018-02-04
```

Une fenêtre s'ouvre :
(la première ligne apparait grise car elle était sélectionnée)



Lors de l'exécution du main de la classe InterfaceGraphique, une fenêtre s'ouvre et l'employé peut cliquer sur des boutons et rentrer les informations demandées dans des cases.
Les fonctionnalités *visualiser, réserver* chambre fonctionnent correctement mais la fonctionnalité *réserver salle* n'est pas encore opérationnelle à cause d'un problème de format de dates (nous n'avons pas ce problème dans le main de MainHotel). Les informations de sorties (les strings dans les system.out.println()) du main de MainHotel ne sont pas toutes affichées sur l'interface.

III. Bilan du projet

Comment vous êtes vous organisés?

Chronologiquement , nous avons d'abord réfléchis aux fonctionnalités qu'on voulait que notre programme possède : réserver, annuler une réservation.. Puis, aux objets qui interviennent dans la gestion d'un hôtel et à leur lien. Ainsi nous avons commencé par créer le diagramme de cas d'utilisation et de classes.
Pour la gestion d'un hôtel, la base de données est primordiale afin de stocker les données et d'y avoir accès donc nous nous sommes vite penchés sur la manière dont nous allions stockés nos données et aux clés étrangères de chaque table pour faire des liens entre nos tables.
Nous avons ensuite commencés à coder les méthodes. A la fin nous avons créer des entrées pour les informations concernant la réservation devant être entrées par l'employé. Puis nous avons fait une interface graphique avec JFrame.

Bien que nous ayons codés les « principales » et « plus grosses » fonctions à deux, nous nous sommes beaucoup repartis les tâches pour les autres méthodes, lorsqu'on avait à modifier/ ajuster /remplir la base de données, continuer de faire l'interface graphique après avoir créer plusieurs fenêtres ensemble, ou bien commencer les rapports écrits.

Qu'a t-on réussi a faire et quels problèmes a t-on rencontrés?

Nous avons réussi à regarder si un client est déjà dans la base de données ou non et l'ajouté automatiquement sinon. Les fonctionnalités reserver, annuler, visualiser fonctionnent bien dans le main de la classe MainHotel.

Dès qu'on a commencé à coder on a eu des problèmes avec la convention des dates c'est pour cela qu'il y a beaucoup de fonctions permettant de convertir des dates dans la classe Reservation.

La fonction recupererDate() convertie une date Mon Jun 18 00:00:00 IST 2012 en date 2012-06-18 , elle n'est pas optimale car nous convertissons « a la main » les mois mais nous avons fait ça dans une soucis de temps car on n'a pas trouvé rapidement une autre méthode avec de la documentation.

Les méthodes qui nous ont demandés le plus de temps sont verifierChambreNonDispo() et verifierSalleNonDispo() car au début on a pas trouvé tout de suite comment bien gérer les cas de recoupement des périodes de réservations. La fonction verifierChambreNonDispo() est longue du fait qu'on ait 3 types de chambres.

Qu'aurions nous pu faire en plus ?

-On aurait pu exporter une facture dans un format csv par exemple.

-On a été pris par le temps pour faire l'interface d'identification de l'employé comme prévue dans l'analyse car nous aurions du créer une table en plus dans notre base de données avec le nom des employés et les mots de passe et ensuite créer la fenêtre mais nous n'avions plus de temps.

-Les salles de réunions ont le même prix à l'heure, on pourrait changer cela.

-Tout n'est pas fonctionnel dans l'interface, on peut essayer de l'améliorer en vue de l'oral.

-La classe Autres qui été prévue pour d'autres options est pas utilisé et opérationnelle dans le main.

-la période de haute saison est fixée et ne prend pas en compte le remplissage réel de l'hôtel.

-Enfin, on aurait également pu faire un planning un peu plus développer pour qu'il contienne plus d'informations.

Conclusion

Nous avons réussi à implémenter la plupart des fonctionnalités attendues pour la gestion d'un hôtel. Plusieurs types de chambres et d'options différentes sont proposées au client.

Ce projet en autonomie nous a permis de nous améliorer beaucoup. De revoir et d'appliquer la modélisation UML, de pratiquer la gestion d'une base de données (requêtes..) et programmer en orienté objet dans un but concret. Nous avons appris à utiliser les JTables et les JFrames que nous avions jamais utilisés avant.