Rapport Projet IHM

Introduction :

Le rapport suivant ainsi que les deux prototypes demandés ont été entièrement réalisés (mise à part l'API qui a été fournie par M. Madelaine) par notre binôme, et le projet a été rendu en retard en partie à cause d'un sabotage à deux reprises de notre base de données.
Notre binôme était composé de Adam MEDDAHI et Quentin LACOMBE.

Préparation du projet :

Avant de commencer à coder, nous avons commencé par réaliser des deux diagrammes de cas d'usage des prototypes ainsi que les diagrammes de classes correspondants, que voici :

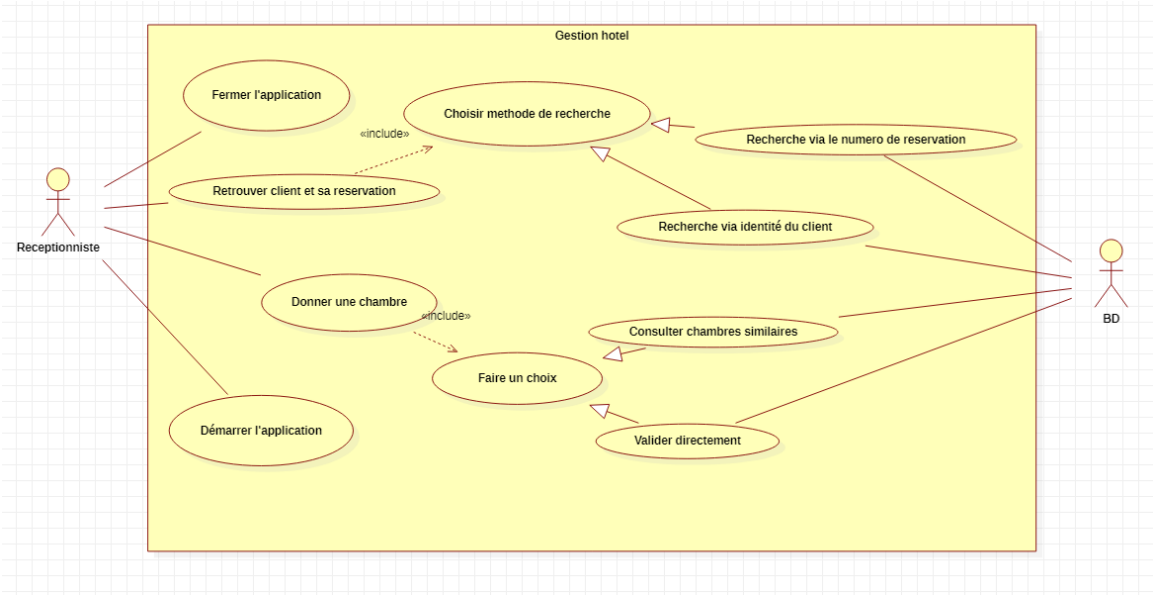

Diagramme de cas d'usage du premier prototype : l'allocation d'une chambre à un client

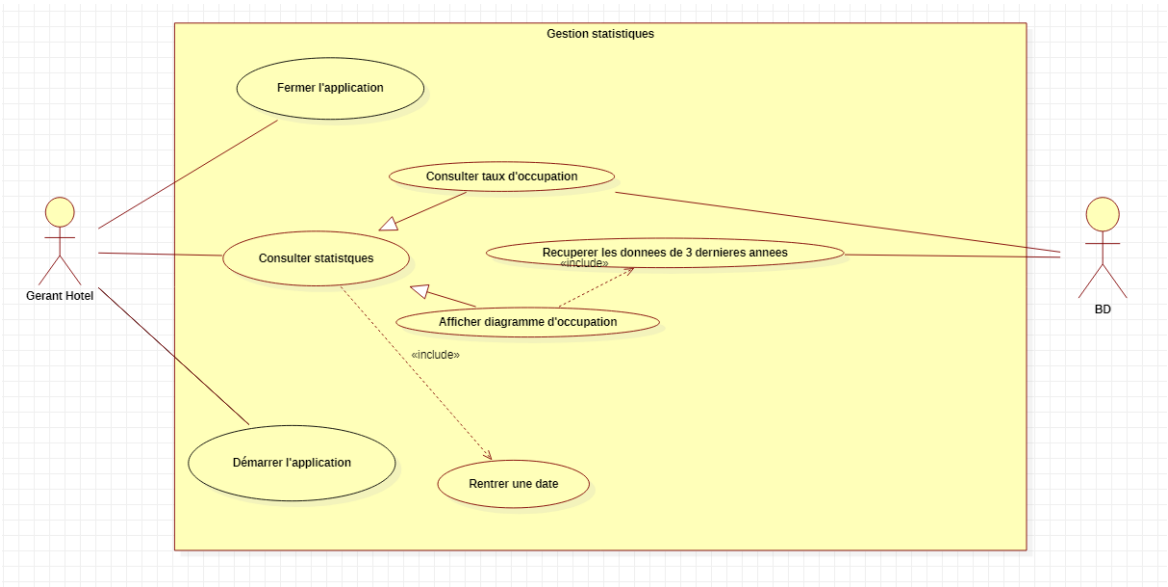

Diagramme de cas d'usage du second prototype : la consultation des statistiques

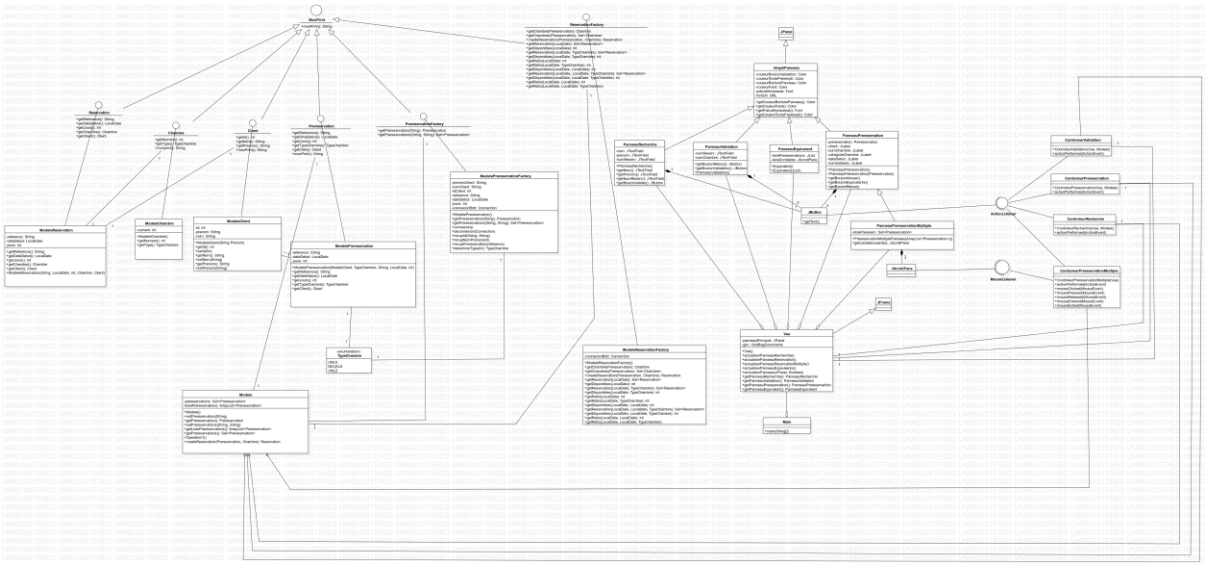

Diagramme de classes du premier prototype : la gestion des réservations

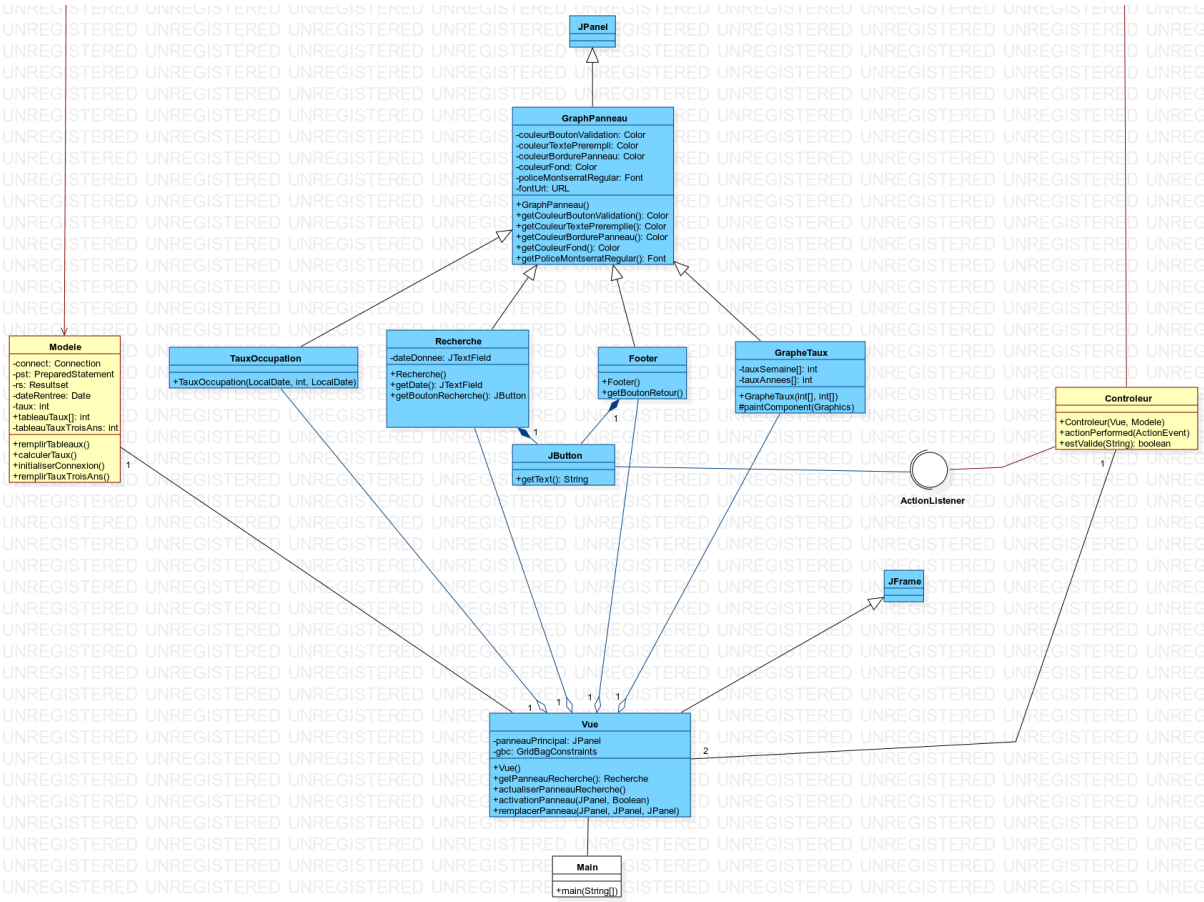

Diagramme de classes du second prototype : la consultation des statistiques

Matérialisation du projet :

Une fois que nous avions une bonne idée de ce que nous allions faire, nous avons dessiné un Wireflow sur le site UIzard.io, et nous nous sommes répartis les tâches et avons commencé l'élaboration du code du premier prototype.

Wireflow de notre premier prototype

Nous avons réalisé les vues sans trop de problèmes, en partie grâce à des JPanel, en initialisant nos conteneurs avec un GridBagLayout et en ajoutant les panneaux à l'intérieur avec des contraintes GridBagConstraints. Grâce à cette méthode, la majorité de notre application est dite « responsive », ce qui permet une expérience utilisateur plus agréable et une meilleure adaptation pour différentes tailles d'écrans. La fenêtre de l'application est divisée en quatre parties, qui sont les panneaux modélisant les fonctionnalités principales demandées, et entre lesquels on peut naviguer facilement grâce à un système de désactivation et d'activation des panneaux. De plus, nous avons utilisé une police différence à celle de base, la police Montserrat, ainsi que des couleurs significatives (comme le vert pour les boutons de validation et le rouge pour les boutons de retour), en essayant d'avoir la vue la plus « propre possible ».

Puis nous sommes passés à la réalisation des modèles, et nous avons bloqué pendant un petit temps car n'avions pas bien saisi l'utilité des usines abstraites, qui finalement servent de « coquilles » permettant la portabilité des données extraites des modèles (en tout cas, c'est comme cela que nous l'avons compris). Après avoir effectué un certain nombre de tests avec des requêtes simples pour être sûrs que nous maîtrisions bien la base de données, nous avons implémenté l'API en suivant les commentaires écrits dessus, et nous avons créé deux nouvelles tables dans la base de données : « AllocationChambres » et « Chambres », pour faciliter l'allocation de chambres selon les pré-réservations.

Le programme possède une classe permettant d'interagir avec la base de données et de retenir les données nécessaires au bon déroulement du programme. Nous avons sobrement appelés cette classe « Modele ». Cette dernière permet de pouvoir rechercher des prereservations et de retenir la prereservation courante ou l'ensemble de prereservations courant. Ce modèle permet aussi de créer une réservation ou un ensemble de réservations en se basant sur le ou les prereservation.s courant.es.

Dans l'état actuel du projet, il est possible de rechercher une préservation que ce soit via l'identité du client ou avec la référence de la prereservation.

Graphiquement, seulement la recherche via référence fonctionne.

Une fois les modèles et les vues réalisées, nous avons codé quatre contrôleurs, qui « écoutent » les boutons des trois premiers panneaux et les clics du deuxième. De ce côté-là, nous n'avons pas rencontré de problèmes particuliers, sauf lors de la vérification des premiers champs, pour dissocier la recherche par nom et prénom, et celle par référence. En effet, nous utilisions une comparaison de String avec « == », ce qui ne fonctionnait pas, jusqu'à ce que l'on trouve notre erreur sur le forum StackOverflow ; il fallait utiliser la méthode compareTo() de la classe String.

Enfin, en se basant sur la production de notre premier prototype, nous sommes passé au second, qui a été assez simple à effectuer car nous avons fait du recyclage des classes de la première partie. La tâche qui nous a pris le plus de temps a été la méthode pour chercher le taux d'occupation, car il fallait regarder les dates antérieures à celle entrée par l'utilisateur pour vérifier qu'elles n'étaient pas associées à des pré-réservations futures (à cause du nombre de nuits). De plus la conception du panneau de graphe a été assez chronophage, car nous avons dû dessiner les lignes et les rectangles plein en prenant en compte les dimensions de la fenêtre pour que tout soit bien responsive, et le calcul pour dessiner les barres de l'histogramme et partant du haut, l'a été d'autant plus car nous avons effectué les tests au pixel près.

Test à réaliser :

Dans le premier prototype :

-Rechercher une préréservation via une référence

-Rechercher les préréservations d'un client via son identité (Nom Prénom)

-Attribuer une réservation validant les critères d'une préréservation

-Afficher les autres réservations valables pour une préservation (équivalents)

Dans le second :

-Effectuer plusieurs recherches pour la même date de la semaine, et vous n'aurez jamais les mêmes valeurs pour les bâtons bleus, car il s'agit d'une simulation faite grâce à des nombres aléatoires (entre 12 et 65 pour que cela soit assez crédible).

Conclusion :

La conception et la réalisation de ce projet nous ont permis de nous familiariser avec les différents concepts d'IHM, ainsi qu'avec les nouvelles notions apprises en ACDA, comme l'exploitation d'une base de données, ou encore l'utilisation de Set<>. Nous avons pus nous familiariser avec les design pattern, notamment les factory, et avec le MVC. Même si cette expérience sur une longue période (plus d'un mois) nous a été très formatrice et enrichissante, elle nous laisse également un goût amer car nous avons été victime du sabotage, puis de la suppression intégrale de notre base de données (qui est en partie la cause du retard sur ce projet), ce qui nous apprendra à mieux sécuriser notre projet la prochaine fois. Enfin, nous avons également passé trop de temps sur la partie réflexion de notre projet, avec des diagrammes trop précis et un Wireflow également, qui aurait pu nous

porter préjudice, mais qui nous permet tout de même , en principe,d'avoir deux applications épurées et agréables à utiliser. Mais nous pensons tout de même que cette approche BDUF (Bif Design Up Front) ne s'inscrit pas dans un développement se voulant agile. Nous tirons donc de ce projet une expérience considérable, aussi bien technique (multiples technologies utilisées), qu'humaine (volonté de nuire en sabotant notre projet à un moment crucial).