Création d'un logiciel pour la gestion de réservation des salles d'université ou d'une entreprise

Régulièrement des mail sont envoyé au logiciel demandant l'attribution d'une ou plusieurs salles pendant une période de temps donnée. Celui-ci interroge la base de données pour vérifier la disponibilité des salle et en fonction du résultat effectue ou non la réservation.

Ce programme comporte quatre class ayant chacune un rôle bien défini :

- Une class *reservation* qui contient les données relative à une demande de réservation : le nom de la salle, la date de réservation, et l'horaire souhaité. Cette class sert au transport de l'information entre les différentes partie du programme.
- Une class *interface_mail* qui est capable de lire les mails (des fichiers texte) et de stocker les informations dans un objet de type *reservation.*
- Une *dao* (Data Access Object) qui gère la connexion à une base de données et l'envoie des requêtes.
- Enfin une class *moteur* qui fait le lien entre toutes les autres : elle récupère les données reçus par *interface_mail*, demande au *dao* les informations lui permettant lui permettant de vérifier si les demandes sont compatibles avec les réservations précédentes et si c'est le cas mes à jours la base.

Pour la class *reservation* :

On peut supposer pour simplifier que les emails ont une structure ressemblant aux exemple ci-dessous, c'est-à-dire que chaque demande de la réservation se présente de la façon suivante : mention de la salle, puis de la date (jour avant mois), puis mention du créneau horaire.

> *Mail1.txt :*
> *Bonjour,*
> *Nous souhaiterions utiliser la salle 736 le mardi 17 janvier entre 15 h et 19 h et la salle 140 le 18 janvier 9 h – 11 h.*
>
> *Mail2.txt :*
> *Salut,*
> *J'ai besoin de la salle 736 le 17 janvier entre 8 h et 10 h.*
>
> *Mail3.txt :*
> *Je peux avoir la salle 736 le mardi 17 janvier de 14 h à 16 h ?*
> *Chris.*

L'objet *reservation* doit pouvoir doit pouvoir contenir toutes ces informations, stockée sous forme de chaine de caractères, sauf les horaires qui seront des entiers pour pouvoir effectuer plus facilement des opérations.

Pour la class *interface_mail :*

La class *interface_mail* n'a qu'une seul méthode supplémentaire *read_mail()*, récupère le contenu du texte et cherche un motif commençant par le mot « salle » et alternant ensuite blocs de chiffres et boc d'autre caractères, censés être dans l'ordre : salle, date, horaire de début et horaire de fin. Le résultat est ensuite utilisé pour construire un objet *reservation*.

On peut stocker les information ainsi extraite dans une liste pérenne qui sera ensuite traité en temps voulu par le reste du programme. Ainsi l'interface et la base de données peuvent travailler de façon asynchrone. C'est-à-dire que l'interface peut recevoir des mail en continu toute la journée et la base de données, être mise à jour une fois par jour.

Pour la méthode *read_mail()*. Pour le moment on peut traiter les mails comme une chaine de caractère directement écrite dans le programme (Une interface utilisateur pourra être créer ultérieurement avec Flask).

Pour la class *dao* (Data Access Object)

Cette permettra de gérer l'interaction entre la base de données et l'appli afin d'enregistrer les réservation.

Elle sera idéalement muni de deux méthodes, *recuperer()* et *inserer()* effectuant respectivement l'insertion d'une ligne dans la table historisant les opérations et la récupération dans une liste de toute les ligne de la table correspondant à une salle donnée pour un jour donnée.

Chaque méthode consiste à se connecter à la base de données, puis à préparer la requête SQL avec les arguments appropriés et enfin envoyer cette requête à la base connectée.

Pour la class *moteur* :

Cette class fera le lien entre toute les autre class et sera idéalement équipé d'une méthode *chek_dispo()* qui utilisera la méthode *recuperer()*, d'une méthode *valider_reservation()* qui utilisera la méthode *inserer()* et d'une méthode *traiter_reservation()* qui utilisera les méthodes *a_traiter(), chek_dispo()* et *recuperer()*.