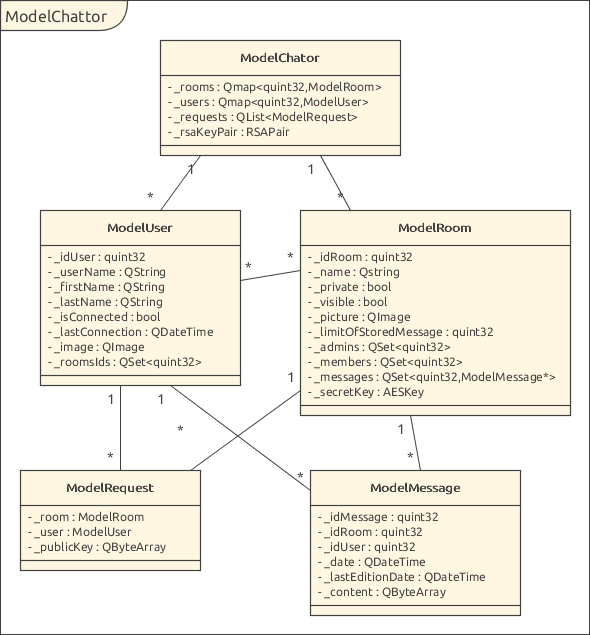
Modèle Chator

Afin qu’une application utilisateur puisse stocker toutes les informations relatives au chat, nous avons décidé d’utilisé un modèle issu du patron MVC.

# Schéma



# Rôle du Modèle

Le rôle de notre modèle et des modèles en général est de stocker des informations. Ces dernières sont stockées séparée du reste de l’application afin de permettre une certain réutilisabilité et modularité. A noter que les informations contenues chez un utilisateur ne sont pas forcément les mêmes que pour un autre utilisateur car ce dernier ne stockera seulement les informations utiles pour lui-même.

Nous utilisons différentes classes pour effectuer ce stockage :

* ModelChator
* ModelRoom
* ModelMessage
* ModelUser
* ModelRequest

Chacune de ces classes proposent des méthodes permettant le contrôledes données (lecture, ajout, modification et suppression) implémentée entre autres par des constructeurs, accesseurs et modificateurs.

# ModelChator

Cette classe contient la liste des utilisateurs ayant des salles en commun avec l’utilisateur de l’application. Elle contient également les salles dont l’utilisateur fait partie ainsi que les salles visibles au public.

C’est par cette classe que toute demande d’information auprès du modèle doit transiter. ModeChator est ensuite en mesure de faire suivre les requêtes vers ModelRoom.

# ModelRoom

Ce modèle décrit une salle de discussion, il contient la liste de tous les membres (simple admin ou utilisateur) ainsi que diverses informations sur la salle. Les messages sont aussi référencés dans ce modèle. Ainsi toutes les requêtes concernant un message (ajout, suppression, modification) arrivent au ModelRoom depuis le ModelChator et sont finalement traitées. Le même cheminement est appliqué pour les opérations relatives aux utilisateurs : ajout et nomination d’admin.

# ModelMessage

Ce modèle est le plus basique, il se situe tout en bas de la hiérarchie. Il stocke uniquement un message et en permet sa modification.

# ModelUser

A même titre que le modèle message, on a accès à différent modificateurs et accesseurs sur les champs privés consistant en plusieurs informations sur l’utilisateur y compris si il est connecté ou non.

# ModelRequest

Permet de stocker les demandes d'adhésion à une salle privée.

# Interaction avec la vue et le contrôleur

Il n’y a pas d’interaction à proprement parler depuis le modèle vers le contrôleur et la vue. C’est le contrôleur qui est chargé de faire le lien entre le modèle et la vue. C’est-à-dire d’afficher les informations du modèle dans la vue ou à l’inverse reçoit une interaction de l’utilisateur sur la vue et répercute ces actions sur le modèles. Ces interactions sont faites via les méthodes publiques mise à dispositions par les différents éléments du patron MVC. Ainsi chaque élément n’a pas connaissance des champs privés de ses voisins.