Design patterns utilisés

Observer

Le design pattern Observer permet de modéliser l'abonnement à un salon de discussion et la notification des utilisateurs du salon à chaque nouveau message.

Command

Le design pattern Command permet de centraliser le comportement de l'envoi de message afin de pouvoir s'en servir à plusieurs moments (en appuyant sur la touche entrée, le clique d'un bouton…).

Proxy

Le design pattern Proxy permet de ne charger qu'en cas de besoin les messages antérieurs d'un salon pour éviter de charger de nombreux objets en une seule fois, tout en étant transparent dans son utilisation.

Visitor

Le design pattern Visitor permet d'ajouter du comportement de formatage de texte selon plusieurs formats (gras, italique, liens, couleurs) avant l'affichage, sans avoir à modifier les classes de base.

State

Le design pattern State permet de gérer le comportement de l'utilisateur en fonction de son état actif ou inactif. Cela nous permet d'afficher instantanément les messages qu'il reçoit ou de les mettre en attente.

Strategy

Le design pattern Strategy permet de formater le contenu de messages selon plusieurs algorithmes sans les connaître à l'avance.

Template method

Le design pattern Template Method permet de définir le comportement général des éléments dans un message qui peut changer en fonction des cas.

Singleton

Le design pattern Singleton permet d'instancier une seule fois les différents états possibles des utilisateurs plutôt que d'instancier un nouvel état à chaque changement d'état d'un utilisateur. Il permet également d'instancier une seule fois les différents formateurs de message plutôt que de les instancier à chaque nouveau message. Il permettrait également d'instancier une seule fois la connexion à la base de données.

Composite

Le design pattern Composite permet d'imbriquer plusieurs éléments composant un message (texte, lien, couleur) en un élément composite et de récupérer le contenu de l'élément de base comme celui de l'élément composite de la même façon.

Prototype

Le design pattern Prototype permet de cloner une instance d'un IMessage afin de formater son contenu, sans connaître à l'avance son type concret.

Diagramme de classes