Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій

Лабораторна робота №6

з дисципліни <<Технології розробки програмного забезпечення>>

Тема <<Шаблони «Abstract Factory», «Factory Method», «Memento», «Observer», «Decorator»>>

Виконав:
студент ІА-23
Содолиснький Вадим

Перевірив:
Мягкий М. Ю.

Київ 2024

Тема: Шаблони «Abstract Factory», «Factory Method», «Memento», «Observer», «Decorator»

Хід роботи

Патерн Наглядач (Observer)

Визначає залежність типу «один до багатьох» між об’єктами таким чином, що при зміні стану одного об'єкта всіх залежних від нього сповіщають про цю подію.

Переваги

На зміну об’єкта реагують всі елементи, які його використовують, що значно зменшує обсяг коду

Проблема

Для офісного комунікатора, який працює в реальному часі потрібно реалізувати функціонал, який дозволяє відстежувати зміни об'єктів таких як: чат, учасники групи і тд.

Рішення

Даний функціонал реалізовано за допомогою патерну Observer та технології SignalR.

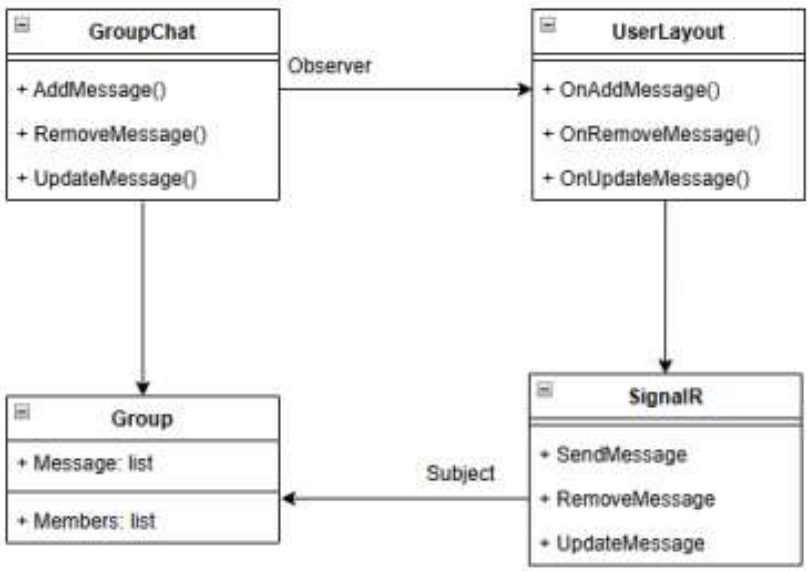
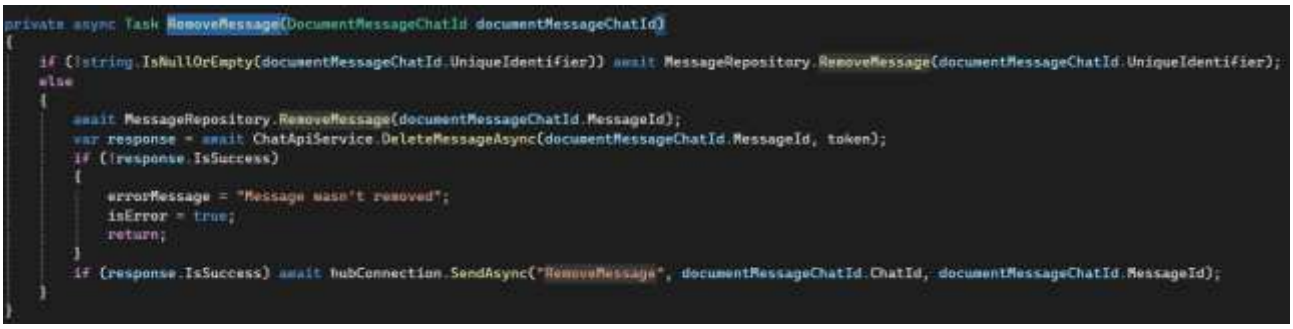


Рис. 1 – Реалізація прослуховування об’єкту Group

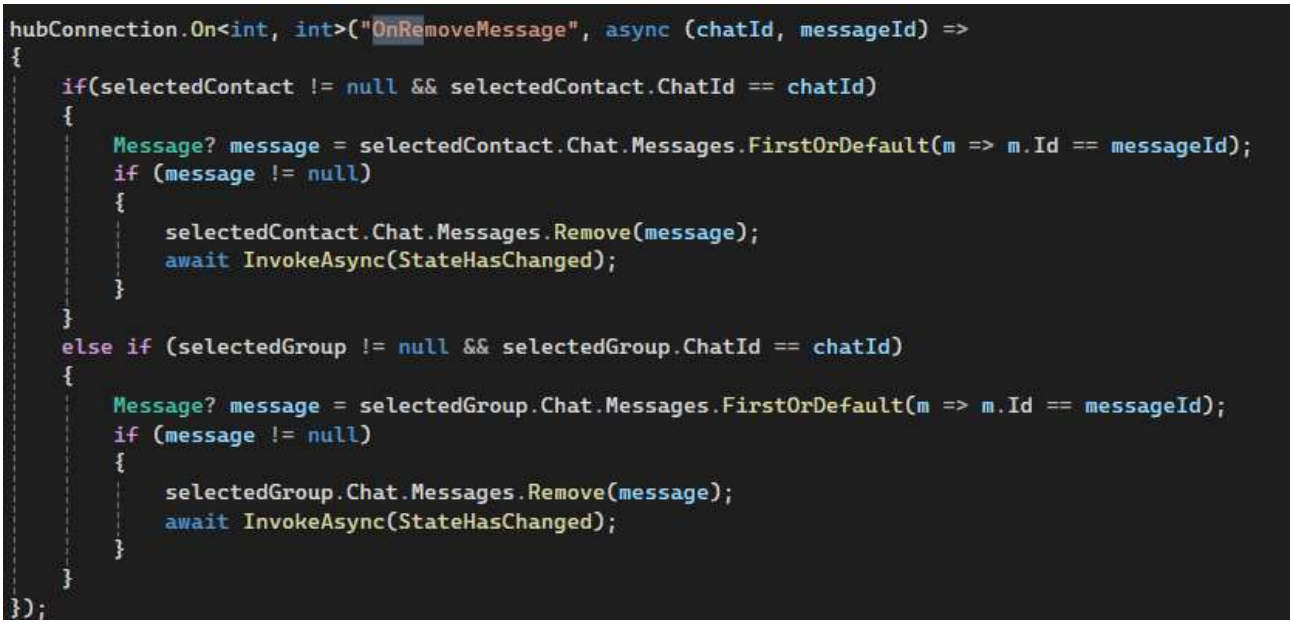
```
private async Task DeleteMessage(int messageId)
{
    Message? message = Group.Chat.Messages.FirstOrDefault(m => m.Id == messageId);
    if (message == null) return;
    Group.Chat.Messages.Remove(message);
    await OnDeleteMessage.InvokeAsync(new DocumentMessageChatId(0, messageId, Group.ChatId, message.UniqueIdentifier));

}
```

Рис. 2 – Виклик методи видалення повідомлення



```
private async Task RemoveMessage(DocumentMessageChatId documentMessageChatId)
{
    if (!string.IsNullOrEmpty(documentMessageChatId.UniqueIdentifier)) await MessageRepository.RemoveMessage(documentMessageChatId.UniqueIdentifier);
    else
    {
        await MessageRepository.RemoveMessage(documentMessageChatId.MessageId);
        var response = await ChatApiService.DeleteMessageAsync(documentMessageChatId.MessageId, token);
        if (!response.IsSuccess)
        {
            errorMessage = "Message wasn't removed";
            isError = true;
            return;
        }
        if (response.IsSuccess) await hubConnection.SendAsync("RemoveMessage", documentMessageChatId.ChatId, documentMessageChatId.MessageId);
    }
}
```

Рис. 3 – Видалення повідомлення



```
hubConnection.On<int, int>("OnRemoveMessage", async (chatId, messageId) =>
{
    if(selectedContact != null && selectedContact.ChatId == chatId)
    {
        Message? message = selectedContact.Chat.Messages.FirstOrDefault(m => m.Id == messageId);
        if (message != null)
        {
            selectedContact.Chat.Messages.Remove(message);
            await InvokeAsync(StateHasChanged);
        }
    }
    else if (selectedGroup != null && selectedGroup.ChatId == chatId)
    {
        Message? message = selectedGroup.Chat.Messages.FirstOrDefault(m => m.Id == messageId);
        if (message != null)
        {
            selectedGroup.Chat.Messages.Remove(message);
            await InvokeAsync(StateHasChanged);
        }
    }
});
```

Рис. 4 – Прослуховування на видалення повідомлення

Сторінка ChatGroup, яка містить об'єкт класу Group може виконувати різні дії над станом об'єкту. Ці зміни стану прослуховуються UserLayout, який викликає методи класу SignalR для відображення цих змін усіх учасників групи. Це дозволяє всім користувачам отримувати релевантний стан чату не перезавантажуючи сторінку та покращує продуктивність застосунка.

Висновок: я використав патерн Observer для розробки застосунка, який працює в реальному часі.