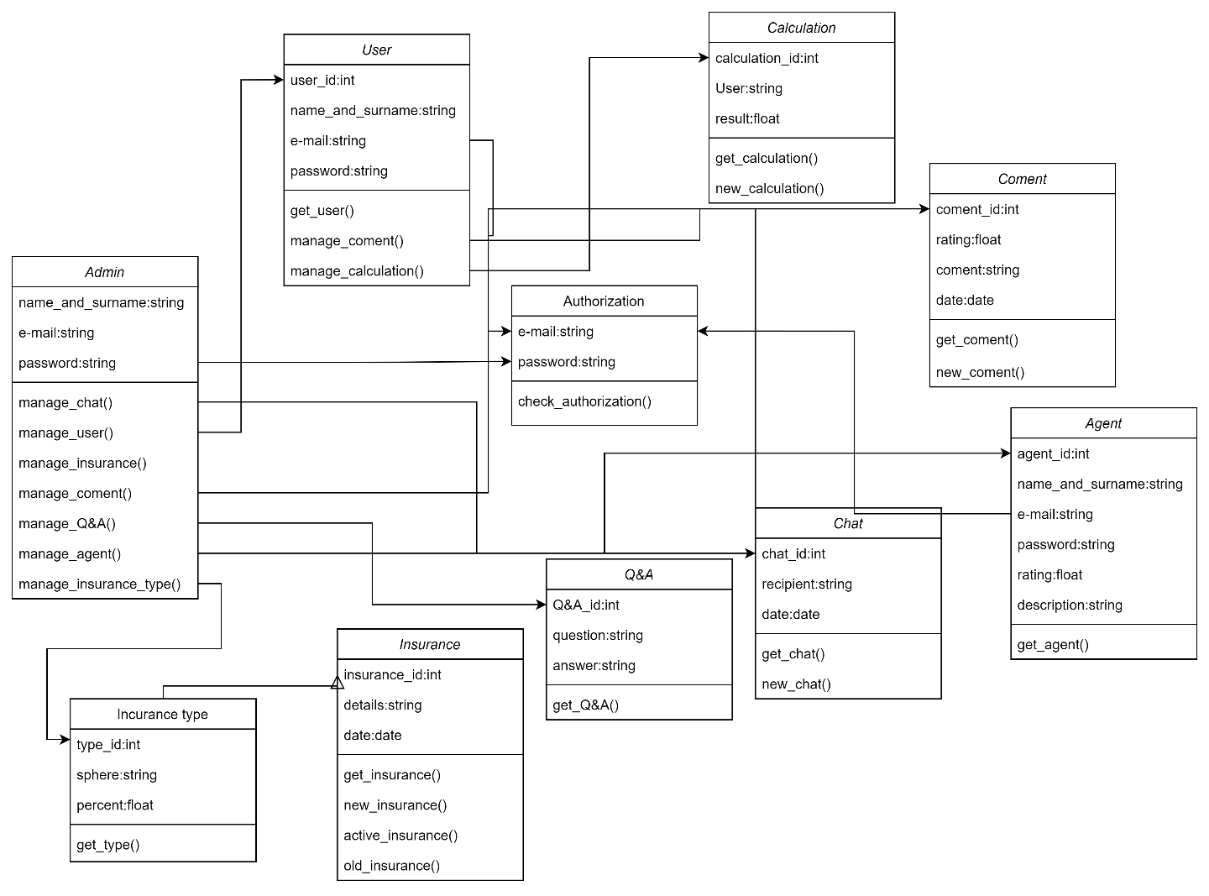
****

**Патерни, які я вважаю доцільно використати:**

**Фабричний метод, для типів страхування**

**Фабричний метод** — це породжувальний патерн проектування, який визначає загальний інтерфейс для створення об’єктів у суперкласі, дозволяючи підкласам змінювати тип створюваних об’єктів.

Оскільки типи страхування є практично одим і тим же – відсоток страхування для певного об'єкту, то як на мене повністю доцільним буде використання патерну Фабричний метод.

Изображение выглядит как текст

Автоматически созданное описание

**Спостерігач для адмінів**

**Спостерігач**— це поведінковий патерн проектування, який створює механізм підписки, що дає змогу одним об’єктам стежити й реагувати на події, які відбуваються в інших об’єктах.

Використання даного патерну для адміністраторів дозволить їм оперативно реагувати на запити на листування та оновлення сайту, а також без проблем керувати користувачами і інформацією на сайті

Изображение выглядит как текст

Автоматически созданное описание

**Ланцюжок обов'язків для чат боту**

**Ланцюжок обов’язків**— це поведінковий патерн проектування, що дає змогу передавати запити послідовно ланцюжком обробників. Кожен наступний обробник вирішує, чи може він обробити запит сам і чи варто передавати запит далі ланцюжком.

З означення чітко випливає пряме завдання чат-бота – вирішити питання, або передати далі, тому цей патерн буде ідеальним для цієї реалізації

Изображение выглядит как текст

Автоматически созданное описание

**Знімок для редагування профіля користувача**

**Знімок** — це поведінковий патерн проектування, що дає змогу зберігати та відновлювати минулий стан об’єктів, не розкриваючи подробиць їхньої реалізації.

Оскільки користувач має змогу редагувати профіль і може передумати, то для уникнення втрати його «старих» даних доцільно використовувати патерн знімок

Изображение выглядит как текст

Автоматически созданное описание

**Одинак для бази даних**

**Одинак** — це породжувальний патерн проектування, який гарантує, що клас має лише один екземпляр, та надає глобальну точку доступу до нього.

Оскільки база даних у нас одна то, як на мене досить хорошим рішення буде використати патерн одинак для відображення бази даних в режимі адміністратора, тоді адміністратори зможуть керувати базою паралельно і відразу бачити зміни, зроблені іншими адміністраторами

Изображение выглядит как текст

Автоматически созданное описание