## DWELLZ

## Опис

Dwellz - це застосунок для зручної оренди житла. Користувачі можуть створити своє оголошення з фотографіями та описом своєї квартири чи будинку для здачі в оренду. Також в застосунку можна знайти житло з різними потребами та фільтрами, щоб швидко знайти найбільш підходящий варіант. Dwellz пропонує безпечну та зручну платформу для здійснення оренди без посередників.

## Функціональні вимоги до ПЗ

*Реєстрація та авторизація:* Застосунок повинен мати можливість реєстрації нових користувачів і авторизації вже зареєстрованих користувачів для доступу до функцій перегляду оголошень, публікації нових оголошень, комунікації між потенційним орендарем та орендодавцем.

*Пошук оголошень:* Застосунок повинен мати список популярних та рекламних оголошень на головній сторінці. Також окремою є сторінка всіх оголошень, що задовільняють введені користувачем параметри пошуку.

*Керування обраними оголошеннями:* Користувач повинен мати можливість додавати оголошення до обраних та, за потреби, видаляти його зі списку.

*Додавання оголошення:* Застосунок повинен забезпечувати можливість створення оголошення з детальним описом, фото та відео з подальшою опцією просування, редагування та видалення. Усі оголошення проходять через модерацію працівниками компанії.

*Чат між орендарем та орендодавцем:* Застосунок повинен мати можливість створювати діалоги між орендарем та орендодавцем стосовно конкретного оголошення.

## Нефункціональні вимоги до ПЗ

*Інтерфейс:* Застосунок повинен мати інтуїтивно зрозумілий та привабливий інтерфейс, що дозволяє легко керувати оголошеннями та здійснювати їх пошук відповідно до заданих фільтрів.

*Швидкість та продуктивність:* Застосунок повинен працювати швидко та ефективно, забезпечуючи миттєве завантаження оголошень на інтерактивній мапі та іншого функціоналу застосунку.

*Сумісність:* Застосунок повинен бути сумісним з усіма актуальними версіями мобільних операційних систем Android та iOS та бути адаптованим під різні діагоналі екрану телефонів та планшетів.

*Безпека та відсутність посередників:* Застосунок повинен надсилати кожне оголошення на модерацію на перевірку коректності опису, зображень, відео та адреси. Орендодавець, а відповідно і його оголошення, може бути верифікованим за допомогою селфі та фотографії паспорту чи водійського посвідчення задля перевірки чи дана особа не є посередником між оголошенням та власниками житла.

**Архітектура ПЗ**

Архітектура визначає загальну структуру та організацію компонентів системи, їх взаємозв'язок та функціональність.

Для застосунку з пошуку та здачі житла під оренду буде використанотрирівневу архітектуру: клієнтський рівень (представлення), серверний рівень (логіка застосунку) та рівень бази даних (зберігання та доступ до даних).

На клієнтському рівні розташовується інтерфейс користувача, через який відбувається взаємодія з проєктом за допомогою мобільного застосунку. Цей рівень відповідає за відображення даних, взаємодію з користувачем та передачу запитів до сервера. Клієнтський рівень буде реалізовано за допомогою фреймворку для кросплатформеної розробки Flutter.

За серверний рівень відповідатиме фреймворк для створення вебзастосунків та API-інтерфейсів ASP.NET Core MVC, який реалізує шаблон Model-view-controller. Серверний рівень містить логіку застосунку, яка обробляє запити користувача, валідує дані, здійснює розрахунки та взаємодію з базою даних. Тут розташовуються контролери, сервіси та інші компоненти, необхідні для коректної роботи застосунку.

Рівень бази даних включає систему керування базами даних (MS SQL Server) та схему бази даних, яка забезпечує зберігання та доступ до даних.

Така архітектура дозволяє чітко розділити функціональність застосунку, спрощує розробку та підтримку, а також полегшує масштабування та розширення системи.

## Взаємодія між компонентами ПЗ

Взаємодія між компонентами ПЗ відбувається з використанням API. Клієнтський рівень взаємодіє з серверним, надсилаючи запити на сервер та отримуючи відповіді. Це може бути реалізовано за допомогою RESTful API, який використовує HTTP протокол для передачі даних.

Серверний рівень обробляє запити, виконує необхідні операції та взаємодіє з базою даних для отримання необхідної інформації. Він також може взаємодіяти з іншими зовнішніми сервісами або API, які можуть знадобитися для додаткової функціональності.

Така архітектура дозволяє забезпечити зручну та безпечну взаємодію між користувачем та системою.

*RapidAPI*, найбільший у світі API-хаб, використовується понад трьома мільйонами розробників для пошуку, тестування та підключення до тисяч API - і все це за допомогою єдиного API-ключа та інформаційної панелі.

*Provider* - це інструмент управління станом у Flutter, який дозволяє віджетам отримувати доступ до загальних даних і перебудовуватися при їх зміні. Він використовує концепцію ін'єкції залежностей, надаючи дані там, де вони потрібні в дереві віджетів, і перебудовуючи лише ті віджети, яким важлива ця частина даних.

*Riverpod* - це бібліотека управління станом для Flutter, яка покликана покращити Provider. Riverpod забезпечує більш безпечний, тестований та гнучкий спосіб керування станом та його спільного використання в застосунку.

## Діаграма варіантів використання

Use-case діаграми використання UML є основною формою опису поведінки системи з точки зору користувача і визначають очікувану поведінку, а не конкретний спосіб реалізації. Вони дозволяють нам проектувати систему з орієнтацією на кінцевого користувача та передавати поведінку системи шляхом визначення зовнішніх взаємодій.

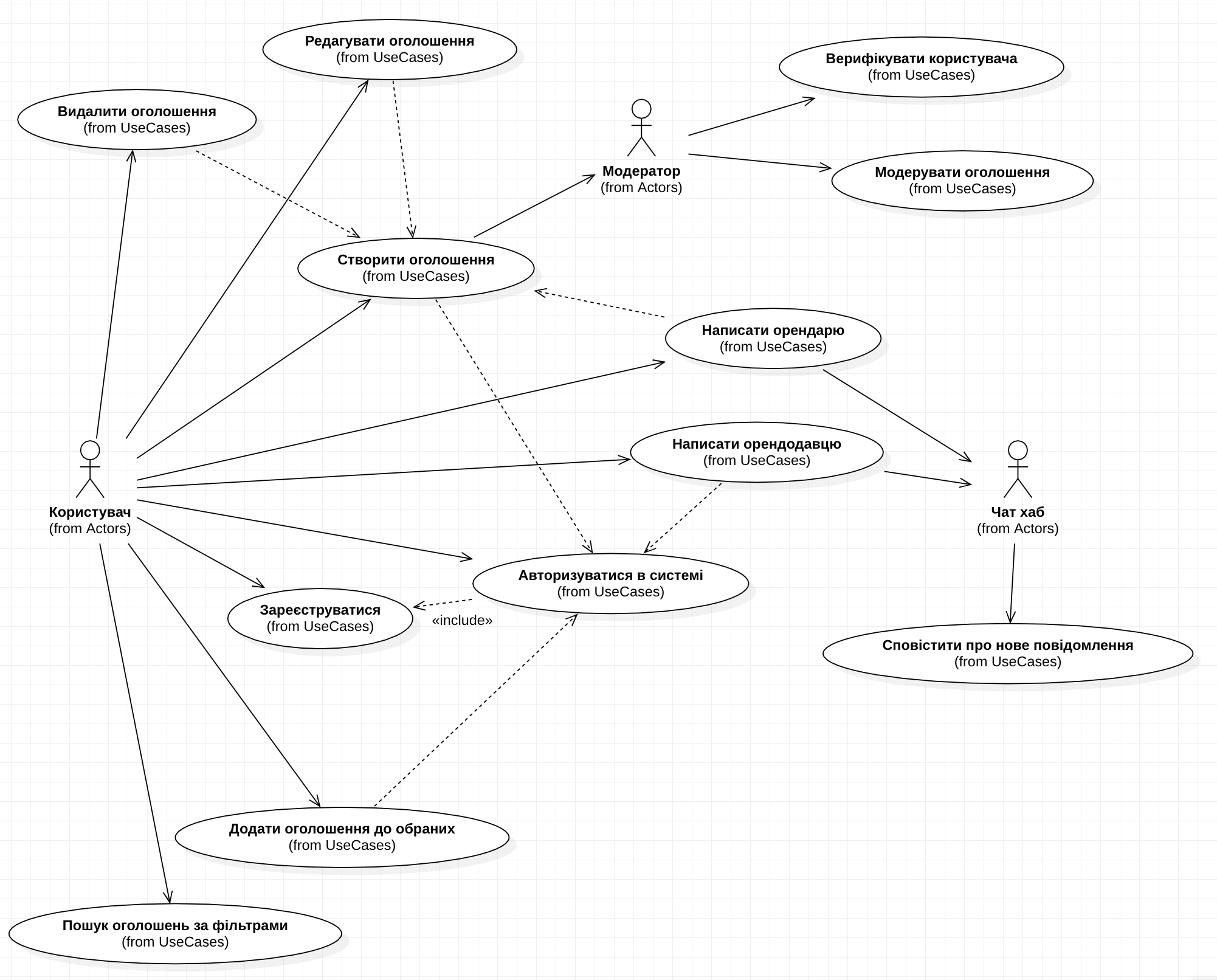


Рис. 3.1 (Use-case діаграма)

## Діаграми послідовностей

Діаграми послідовності в UML є важливими діаграмами для моделювання взаємодії об'єктів і деталей виконання операцій. Вони зосереджені на послідовності взаємодій та передають цю інформацію за допомогою вертикальної осі діаграми, відображаючи моменти надсилання повідомлень.

Діаграми послідовності широко використовуються для уточнення діаграм прецедентів, що дозволяють детально описати поведінкові аспекти системи. Вони передають інформацію у формі повідомлень, а взаємодія об'єктів показується через обмін цими повідомленнями в межах сценарію.

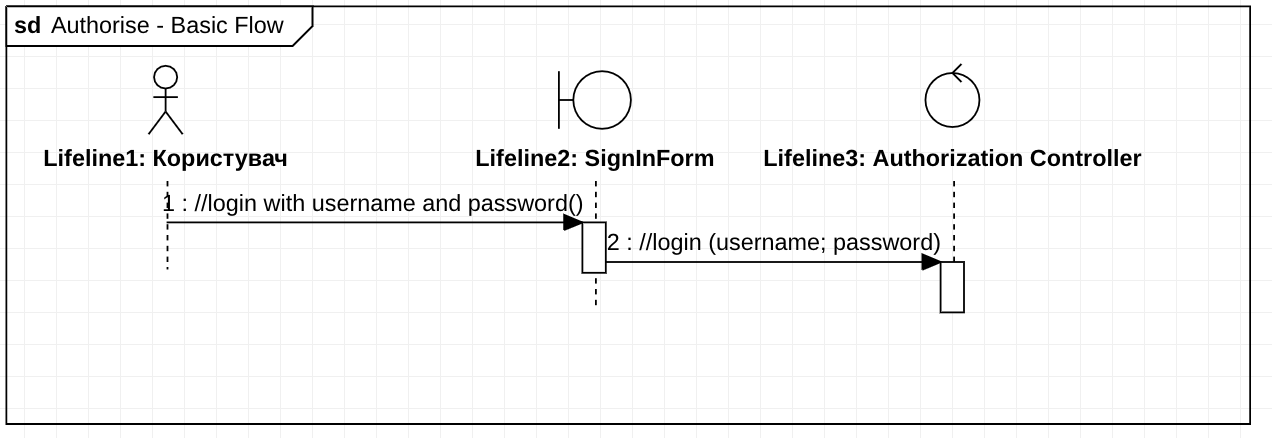


Рис. 3.3 (Діаграма послідовності авторизації)

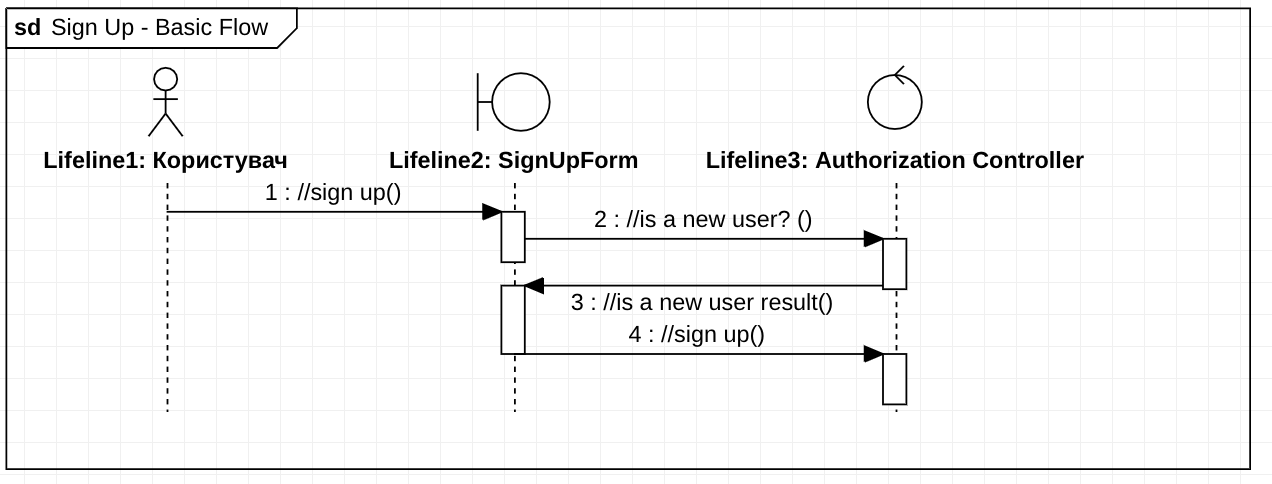


Рис. 3.4 (Діаграма послідовності реєстрації)

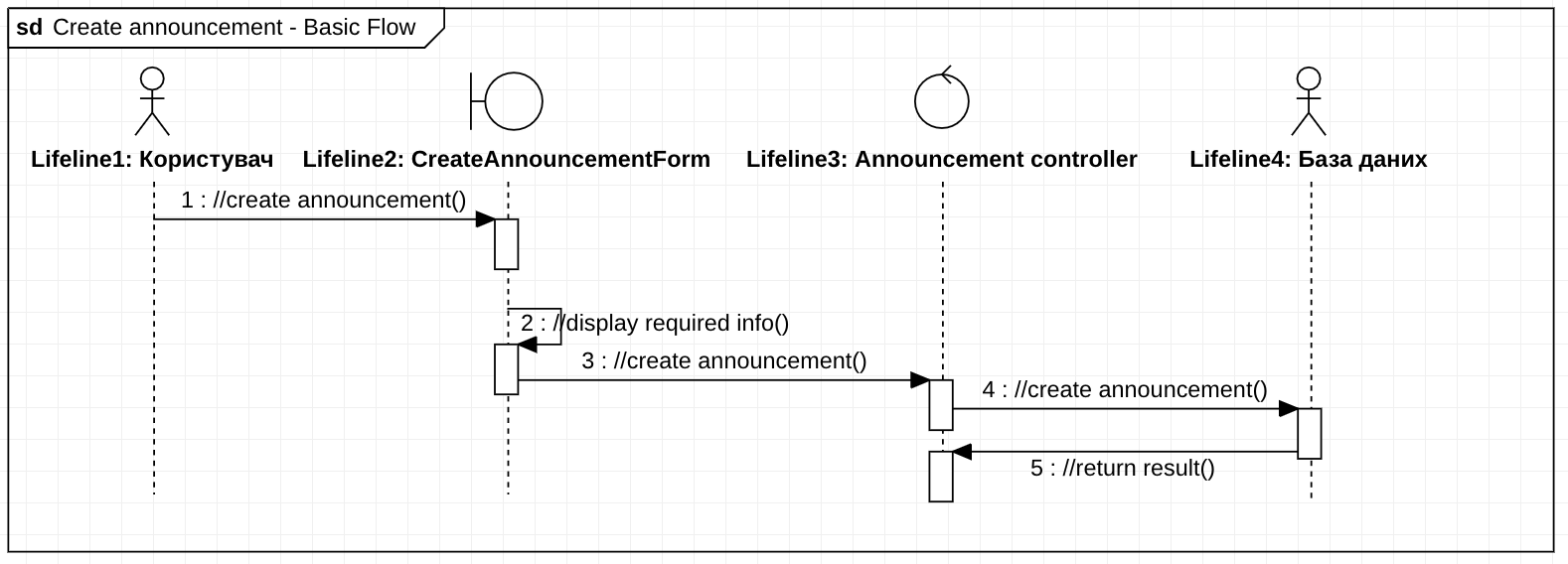


Рис. 3.5 (Діаграма послідовності створення оголошення)

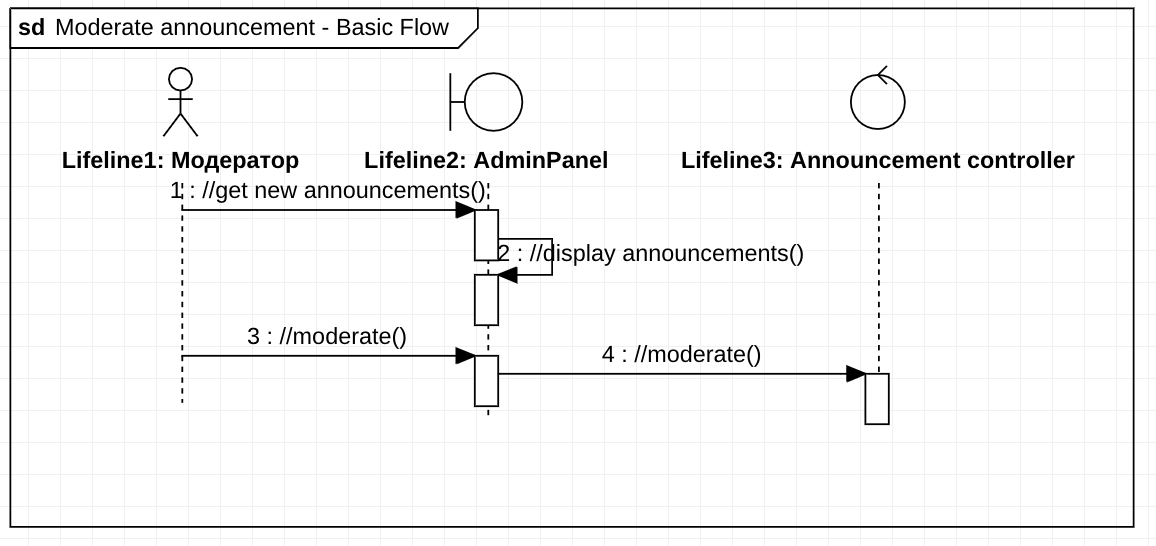


Рис. 3.6 (Діаграма послідовності модерації оголошення)

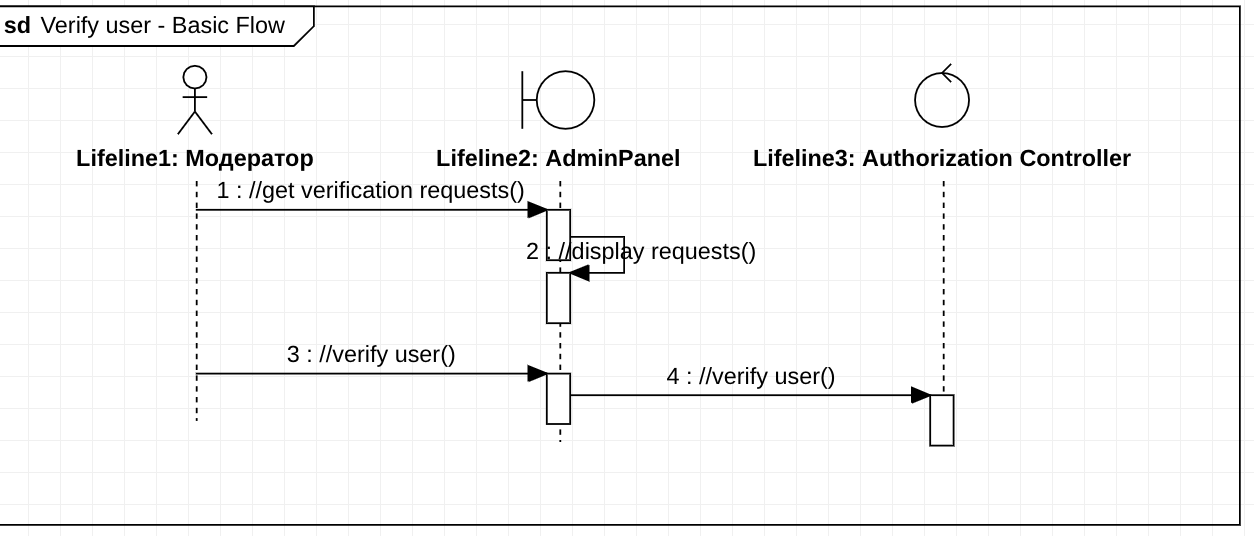


Рис. 3.7 (Діаграма послідовності верифікації користувача)

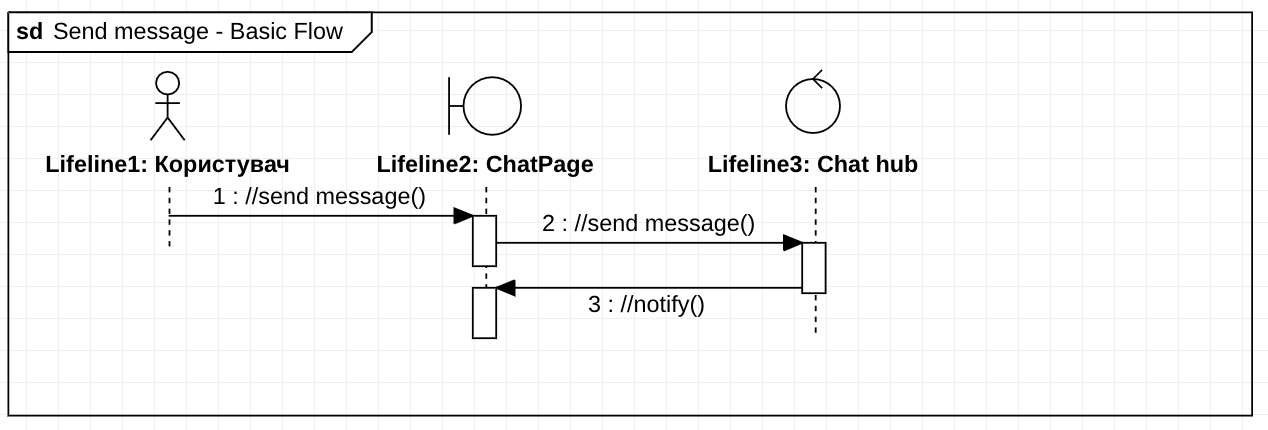


Рис. 3.8 (Діаграма послідовності надсилання повідомлення)

## Діаграма класів

Діаграма класів - це діаграма, яка моделює структуру системи шляхом відображення класів, їх атрибутів та методів, а також взаємозв'язків між класами.

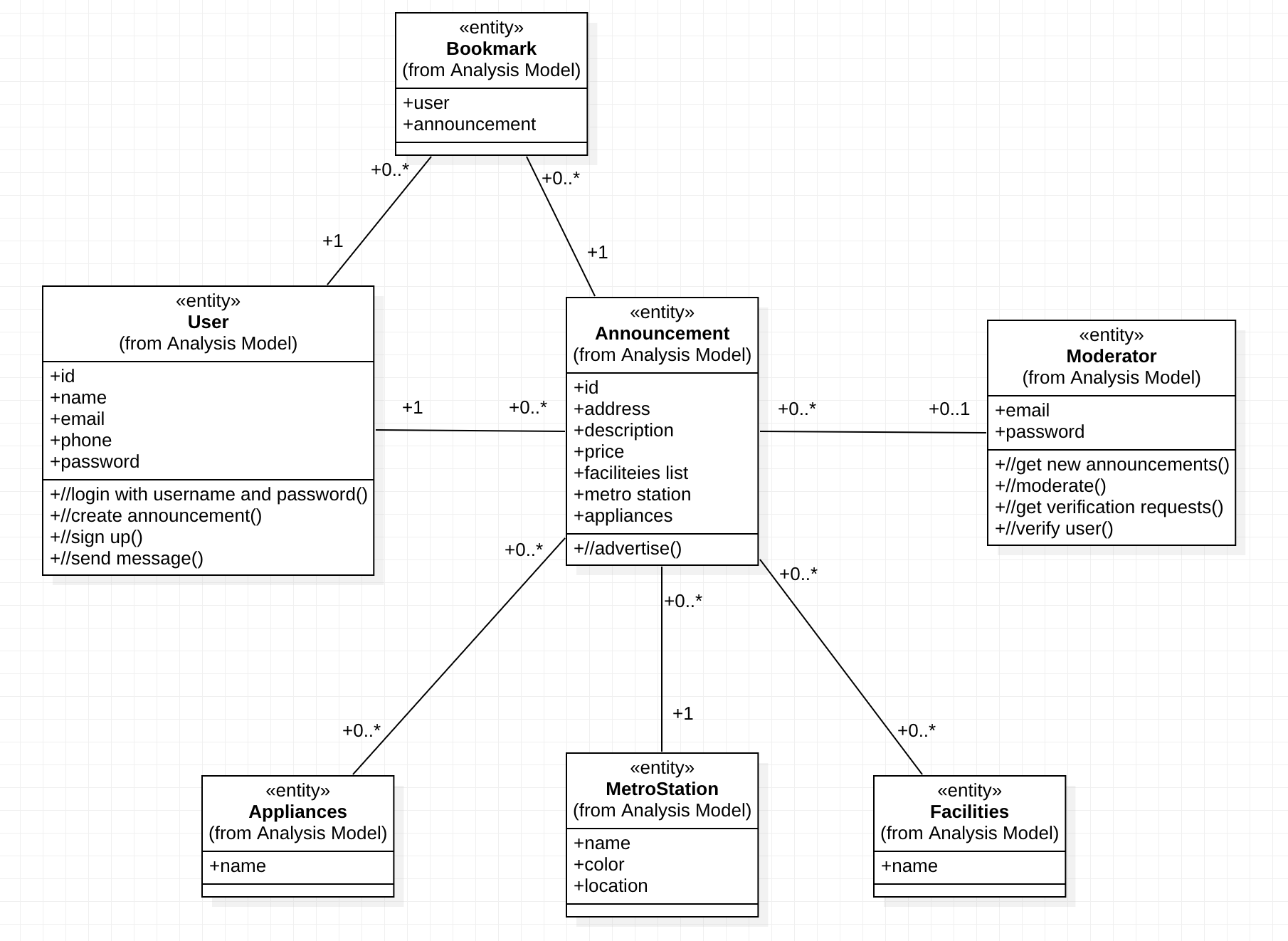


Рис. 3.9 (Діаграма класів)

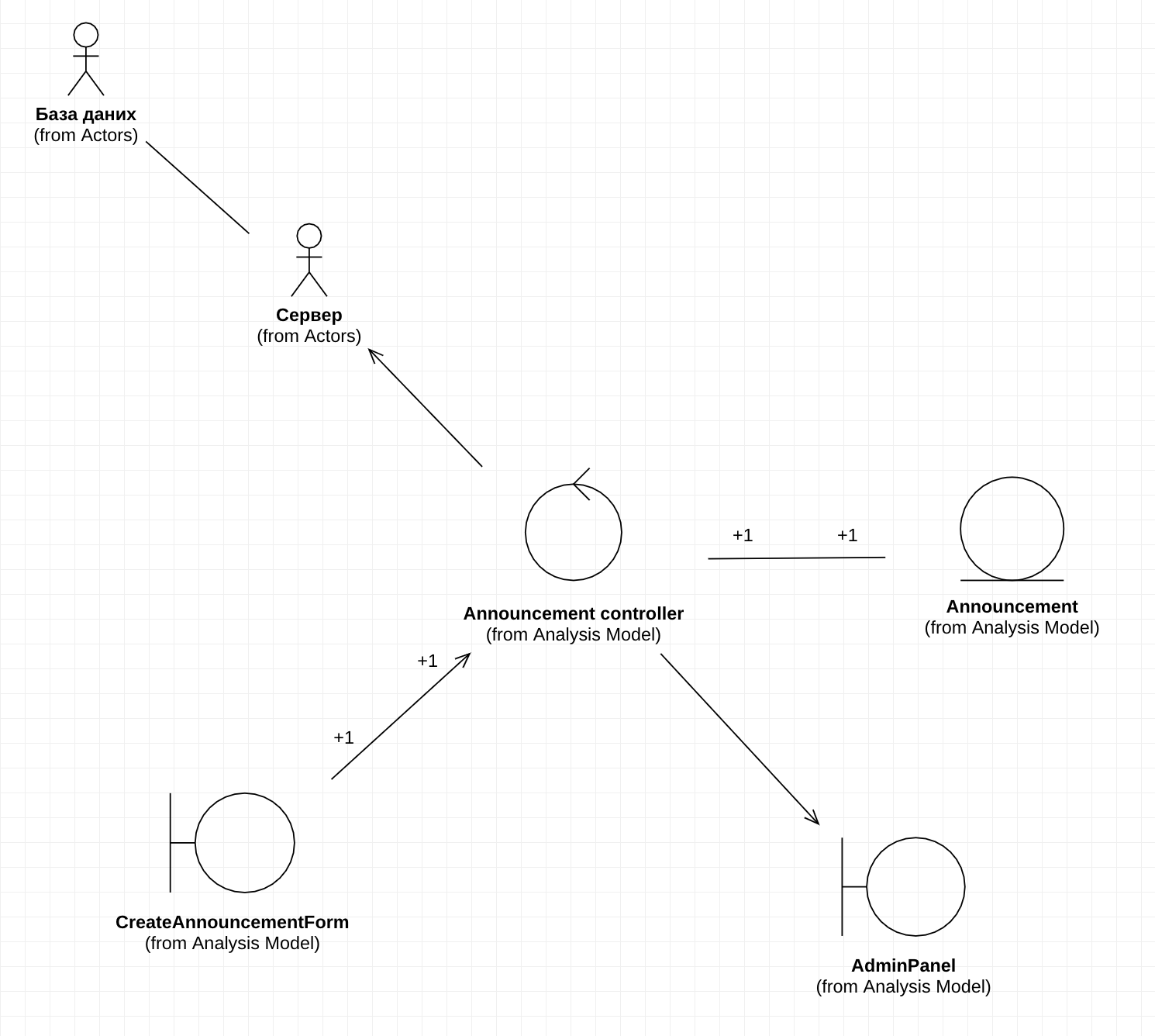


Рис. 3.10 (Діаграма класів VOPC (full))

## Діаграма розгортання

Діаграма розгортання (Deployment diagram) - це діаграма, яка моделює фізичне розташування апаратного та програмного забезпечення системи. Вона відображає структуру та конфігурацію апаратних компонентів (серверів, комп'ютерів, мережевих пристроїв) та програмних компонентів (програм, модулів, бібліотек) у системі. Діаграма розгортання показує зв'язки між цими компонентами та мережами, а також визначає способи комунікації між ними.

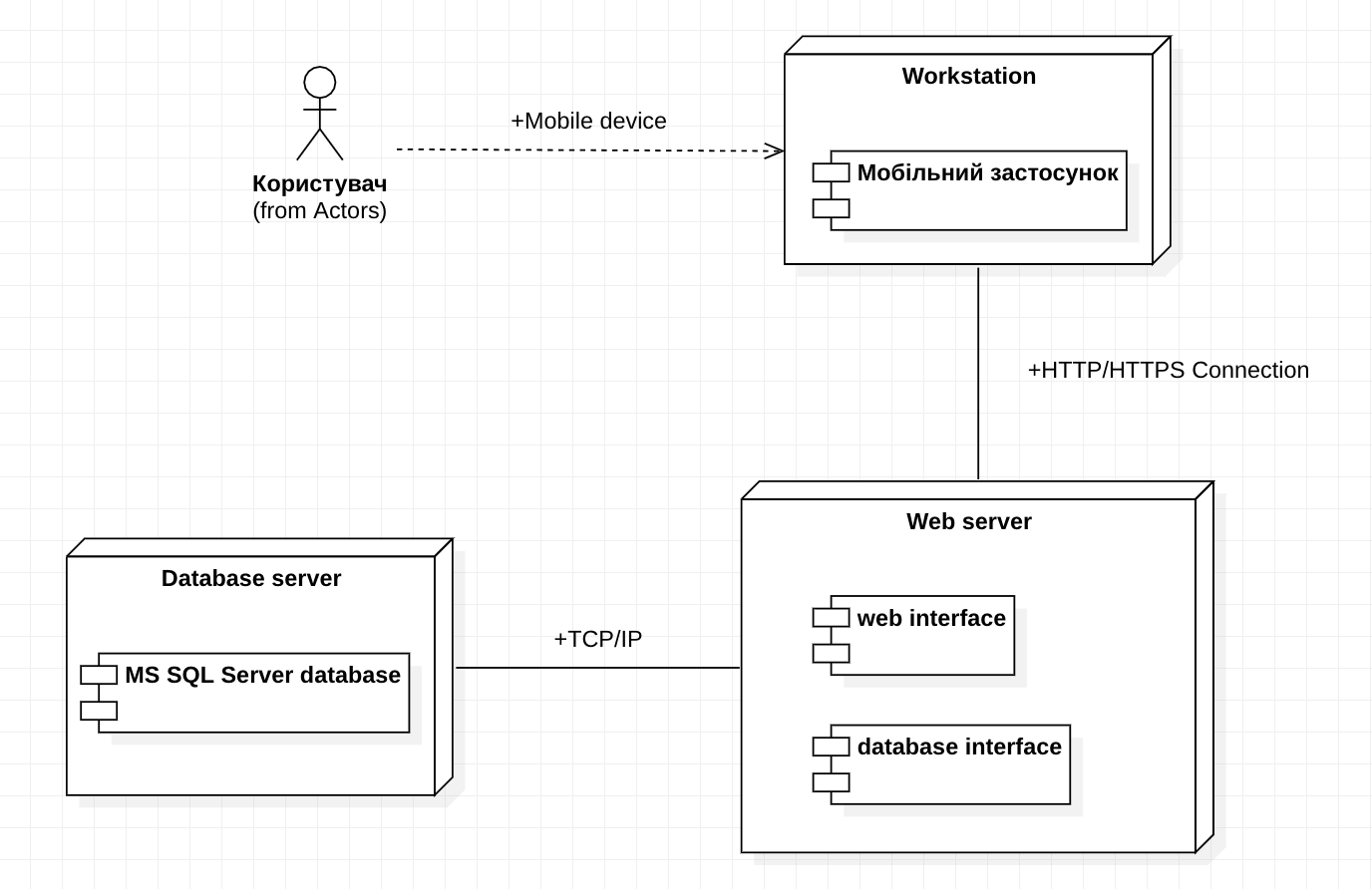


Рис. 3.11 (Діаграма розгортання)