Міністерство освіти і науки України

Харківський національний університет імені В.Н. Каразіна

Навчально-науковий інститут комп'ютерних наук та штучного інтелекту

Рекомендовано до захисту:

протокол засідання кафедри

№ ___ від ____.____.2025 р.

Завідувач кафедри _____

_____ _____

(підпис) (прізвище та ініціали)

Пояснювальна записка

до кваліфікаційної роботи бакалавра

на тему_____Створення універсальної платформи для ОТГ, ОСББ_____

Захищено на засіданні ЕК № _____

протокол № ____ від ____.____.2019 р.

Оцінка _____ / _____

Голова ЕК

_____ _____

(підпис) (прізвище та ініціали)

Виконав:

студент 4 курсу, групи КС- 43

Спеціальності:122-Комп’ютерні науки

(шифр і назва спеціальності підготовки)

Воловецький Єгор Михайлович

(прізвище, ім’я та по батькові, підпис)

Керівник старший викладач Рало О.М.

(ступень, звання, прізвище та ініціали, підпис)

Рецензент к.т.н., доцент Маций О.Б.

(ступень, звання, прізвище та ініціали, підпис)

Харків – 2025

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Харківський національний університет імені В. Н. Каразіна

Навчально-науковий інститут комп'ютерних наук та штучного інтелекту

Кафедра Інтелектуальних програмних систем і технологій

Рівень вищої освіти (освітньо-кваліфікаційний рівень) бакалавр

Галузь знань: 12 Інформаційні технології

Спеціальність: 122 Комп'ютерні науки

Освітня програма: 122 Комп'ютерні науки

ЗАТВЕРДЖУЮ

В.О. завідувача кафедри

Олег ОЛЕШКО

«____» _____20___ року

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ (ПРОЕКТ)

_____Воловецький Єгор Михайлович_____

(прізвище, ім'я, по батькові студента)

1. Тема роботи_Створення універсальної платформи для ОТГ, ОСББ

Керівник роботи_Рало Олександр Миколайович_____,

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від "16"_квітня_____2025року №_№ 4101-5/962

2. Строк подання студентом роботи_____2 червня_____

3. Перелік питань, які потрібно розробити

Проаналізувати існуючі варіанти доступних камер відеонагляду, вирішити які інструменти використовувати для комфортної розробки, розробити та реалізувати веб-додаток для ОТГ та ОСББ, що включає в себе контроль

камерами відеонагляду та іентегрований чат для жильців спільноти_____

4. План роботи

№ з/п	Назви етапів роботи	Термін виконання етапів роботи
1	Аналіз та постановка задачі	січень
2	Дослідження камер та інструментів розробки	початок лютого
3	Розробка архітектури платформи	лютий-початок березня
4	Розробка програмного забезпечення	березень - квітень
5	Тестування	кінець квітня
6	Підготовка пояснювальної записки	травень
7	Підготовка доповіді та презентації	травень

5. Дата видачі завдання_____19 січня 2025_____

Студент ____Воловецький Є.М.___ _____ ____

ініціали, прізвище підпис

Керівник роботи ____Рало О.М.____ _____

ініціали, прізвище підпис

АНОТАЦІЯ

Дипломна робота складається з пояснювальної записки обсягом 44 сторінки, яка містить 11 рисунків, 6 додатків та перелік 22 джерел. Робота присвячена розробці універсальної платформи для об’єднань територіальних громад (ОТГ) та об’єднань співвласників багатоквартирних будинків (ОСББ), яка інтегрує функції відеоспостереження, комунікації та управління.

Метою роботи є створення цифрового рішення для підвищення ефективності управління житловою інфраструктурою, забезпечення безпеки та покращення комунікації між мешканцями та адміністрацією. Для реалізації платформи використано сучасні технології: Java Spring Boot (бекенд), React.js (фронтенд), PostgreSQL (база даних), FFmpeg (обробка відеопотоків) та WebSocket (чат у реальному часі).

Результатом роботи є функціональна платформа, яка поєднує модулі відеоспостереження, чату та адміністрування. Наукова новизна полягає в інтеграції цих функцій у єдиний інтерфейс, адаптований під потреби українських ОТГ та ОСББ. Рекомендовано використовувати розробку для автоматизації управління житловою інфраструктурою, підвищення прозорості процесів та покращення комунікації між користувачами.

Ключові слова: ОТГ, ОСББ, ВІДЕОСПОСТЕРЕЖЕННЯ, ЧАТ, УПРАВЛІННЯ, SPRING BOOT, REACT, FFMPEG, WEB SOCKET, БЕЗПЕКА, ІНТЕГРАЦІЯ.

ABSTRACT

The work consists of an explanatory note of 44 pages, containing 11 figures, 6 appendices, and a list of 22 references. The work is dedicated to the development of a universal platform for territorial communities (ОТГ) and condominium associations (ОСББ), integrating video surveillance, communication, and management functions.

The aim of the work is to create a digital solution to improve the efficiency of housing infrastructure management, ensure security, and enhance communication between residents and administrators. Modern technologies were used for implementation: Java Spring Boot (backend), React.js (frontend), PostgreSQL (database), FFmpeg (video stream processing), and WebSocket (real-time chat).

The result is a functional platform combining video surveillance, chat, and administration modules. The scientific novelty lies in the integration of these functions into a single interface tailored to the needs of Ukrainian ОТГ and ОСББ. The platform is recommended for automating housing management, increasing process transparency, and improving user communication.

Keywords: ОТГ, ОСББ, VIDEO SURVEILLANCE, CHAT, MANAGEMENT, SPRING BOOT, REACT, FFMPEG, WEB SOCKET, SECURITY, INTEGRATION.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ СКОРОЧЕНЬ І ТЕРМІНІВ, ОДИНИЦЬ ВИМІРУ

ОТГ – об’єднання територіальних громад

ОСББ – об’єднання співвласників багатоквартирних будинків

БД – база даних

ВСТУП

У XXI столітті інформаційні технології стали невіддільною частиною майже всіх сфер людської діяльності. Системи управління житлово-комунальним господарством також не залишаються осторонь цифрових трансформацій. Потреба у безпеці, прозорості управління, ефективному комунікуванні між мешканцями та органами місцевого самоврядування постійно зростає. У зв'язку з цим усе більше громад, об'єднань співвласників багатоквартирних будинків (ОСББ), територіальних громад (ОТГ) та керуючих компаній звертають увагу на впровадження сучасних цифрових рішень, які здатні оптимізувати процеси управління, забезпечити безпечне житлове середовище для мешканців та підвищити рівень їхньої обізнаності й залученості.

Особливої актуальності набувають мобільні та веб-додатки, які дозволяють здійснювати контроль за станом будинку, прибудинкової території та взаємодіяти з іншими мешканцями в режимі реального часу. Зокрема, інтерес викликають рішення, які поєднують можливості відеонагляду з елементами соціальної взаємодії — такими як чати, сповіщення, опитування, звернення до адміністрації тощо. Упровадження таких технологій є важливим кроком на шляху до створення «розумного» житлового середовища.

У межах даної дипломної роботи розглядається концепція та реалізація програмного продукту — веб-додатку, призначеного для використання мешканцями ОСББ та представниками місцевого самоврядування. Основними функціональними модулями додатку є:

- Система відеоспостереження з можливістю перегляду зображень з камер у режимі реального часу;
- Функція чату для оперативного обміну повідомленнями між мешканцями, а також з адміністрацією або відповідальними особами;

- Механізми авторизації та контролю доступу, що дозволяють гарантувати безпеку та конфіденційність обміну інформацією.

Розробка таких додатків потребує комплексного підходу: врахування технічних можливостей (інтеграція з камерами відеоспостереження, збереження відеоархіву), вимог до захисту персональних даних, зручності інтерфейсу для різних категорій користувачів, а також забезпечення стабільної роботи сервісу навіть при великій кількості одночасних підключень.

Об’єктом дослідження є процес організації цифрової взаємодії між мешканцями житлових будинків, органами місцевого самоврядування та адміністрацією ОСББ/ОТГ з використанням інформаційно-комунікаційних технологій.

Предметом дослідження виступають методи, програмні засоби та архітектурні рішення для розробки веб-додатку, що забезпечує контроль за системою відеоспостереження та підтримує функціональну взаємодію між користувачами шляхом обміну повідомленнями (чатом).

Метою даної роботи є дослідження та реалізація інструментів для організації цифрової взаємодії в межах житлової інфраструктури з одночасним впровадженням механізмів моніторингу об’єктів у реальному часі. У процесі дослідження буде проведено аналіз існуючих рішень, визначено основні вимоги до архітектури додатку, спроектовано та реалізовано прототип, а також оцінено ефективність та потенційні шляхи масштабування розробки.

Наукова новизна полягає у поєднанні функціоналу системи відеоспостереження та комунікаційної платформи в одному додатку, адаптованому під потреби українських ОТГ та ОСББ. Таке рішення дозволяє створити єдине інформаційне середовище для взаємодії мешканців, підвищити рівень безпеки та оперативності прийняття рішень у межах багатоквартирних будинків і місцевих громад.

1. ОБ'ЄКТНА І ПРЕДМЕТНА ОБЛАСТІ ДОСЛІДЖЕННЯ. ВИКОРИСТАНІ ТЕХНОЛОГІЇ

1.1 Об'єктна область дослідження

Об'єктом дослідження є сучасні цифрові платформи для автоматизації управління комунальною інфраструктурою, зокрема для об'єднаних територіальних громад та об'єднань співвласників багатоквартирних будинків. У рамках дослідження розглядаються процеси проектування, розробки та впровадження програмного рішення, яке інтегрує:

- систему відеоспостереження (включно з обробкою потокового відео та архівуванням даних);
- механізми комунікації (чат);
- систему контролю доступу (розмежування ролей, автентифікація).

Акцент робиться на універсальності платформи, що дозволяє адаптувати її під різні типи користувачів (адміністратори ОТГ, управителі ОСББ, жильці) та масштабувати функціонал залежно від потреб.

1.2 Предметна область дослідження

Предметом дослідження є програмно-архітектурні рішення, що забезпечують:

1. Ефективну взаємодію модулів:
   - Інтеграція відеоспостереження з чатом.
   - Синхронізація даних між frontend (React) та backend (Spring Boot) через REST API та WebSocket.
2. Безпеку даних:
   - Шифрування конфіденційної інформації (особисті дані, відеозаписи).

- Рольова модель доступу (наприклад, обмеження прав перегляду камер для звичайних жильців).

3. Масштабованість:

- Використання мікросервісної архітектури для розподіленого оброблення відеопотоків.
- Оптимізація запитів до бази даних для швидкої роботи з великими обсягами даних (журнали подій, архіви відео).

1.3 Використані інструменти та технології

Для реалізації платформи застосовано сучасний стек технологій, а саме:

3.3.1. Backend-розробка (Java Spring)

- Spring Boot: Основа серверної частини, що забезпечує швидке розгортання, авто-конфігурацію та вбудований сервер (Tomcat).
- Spring Security:
  - JWT-аутентифікація для захисту API.
- Spring Data JPA/Hibernate: Для ORM-маппінгу та роботи з реляційними базами даних (PostgreSQL).
- Spring WebSocket: Реалізація чату у реальному часі.
- FFmpeg: Бібліотека для обробки відеопотоків (перетворення rtsp-потоку у hls так ретрансляція).
- Lombok: Скорочення коду (getters/setters).

3.3.2. Frontend-розробка (React)

- React.js: Бібліотека для побудови динамічного інтерфейсу з використанням компонентного підходу.
- Redux Toolkit: Управління глобальним станом додатка (наприклад, зберігання даних про поточного користувача).

- Axios: HTTP-клієнт для взаємодії з бекендом.

3.3.3. Інфраструктура та DevOps

- PostgreSQL: Основна база даних для зберігання структурованих даних (користувачі, чати, документи).
- Git/GitHub Actions: Система контролю версій та CI/CD для автоматичного тестування й деплою.

3.3.4. Додаткові інструменти

- JUnit: Модульне тестування бекенду.

1.4. Висновок

Обраний стек технологій (Java Spring + React) дозволить створити продуктивну, безпечну та масштабовану платформу, яка охоплює всі ключові потреби ОТГ та ОСББ. Використання сучасних інструментів (наприклад, WebSocket для чату) забезпечує гнучкість подальшого розвитку системи.

2. АКТУАЛЬНІСТЬ ТЕМИ ДОСЛІДЖЕННЯ ТА АНАЛІЗ ІСНУЮЧИХ АНАЛОГІВ

2.1. Актуальність теми

Сучасні об’єднання співвласників багатоквартирних будинків (ОСББ) та об’єднані територіальні громади (ОТГ) стикаються з низкою проблем, пов’язаних з управлінням комунальною інфраструктурою, комунікацією між жильцями та забезпеченням безпеки. Традиційні методи управління (паперовий документообіг, розсилка оголошень, фізичні збори) неефективні в умовах цифровізації.

Ключові проблеми, що обумовлюють актуальність розробки універсальної платформи:

1. Недостатня інформаційна прозорість – жильці часто не мають доступу до актуальних рішень ОСББ/ОТГ, стану комунальних платежів або архівів відеоспостереження.

2. Відсутність єдиного комунікаційного простору – обмін інформацією відбувається через соцмережі, чати у месенджерах (Telegram, Viber), що ускладнює контроль і зберігання даних.

3. Складність управління безпекою – системи відеоспостереження часто ізольовані, не інтегровані з платформами для жильців.

4. Низький рівень цифрової грамотності – деякі користувачі (особливо літні люди) потребують простого інтерфейсу для взаємодії.

5. Відсутність стандартизованих рішень – багато ОСББ/ОТГ використовують розрізнені програми для різних задач (наприклад, окремо чат, окремо система обліку платежів).

Розробка універсальної платформи, яка поєднує відеоспостереження, комунікаційні інструменти та управлінські функції, дозволить:

- Підвищити ефективність управління (швидке прийняття рішень, онлайн-голосування).
- Забезпечити прозорість (доступ до документів, протоколів, відеоархівів).
- Покращити безпеку (інтеграція камер з чатом для оперативного реагування).
- Спростити комунікацію (єдиний простір для обговорень, скарг, оголошень).

2.2. Аналіз існуючих аналогів

На ринку існують різні програмні рішення для ОСББ та ОТГ, проте більшість з них мають обмежений функціонал або орієнтовані на окремі задачі.

2.2.1. Відомі аналоги для ОСББ

1. "Будинок Online" (Україна)
   - Функціонал: Управління платіжами, чат, документообіг.
   - Недоліки: Відсутність модуля відеоспостереження, закрита API.
   - Відмінність моєї платформи: Інтеграція з камерами, відкрита архітектура.
2. "BuildingLink" (США/Канада)
   - Функціонал: Заявки на ремонт, резервування місць, оголошення.
   - Недоліки: Високі тарифи, немає української локалізації.
   - Відмінність моєї платформи: Адаптація під українські реалії, lower-cost рішення.
3. "Condominium" (Польща)
   - Функціонал: Фінансовий облік, голосування.

- Недоліки: Немає чату та інтеграції з камерами.
- Відмінність моєї платформи: Система контролю камерами та інтеграція з чатом для жильців.

2.2.2. Рішення для ОТГ (міст і громад)

1. "Smart City Hub" (Європа)
   - Функціонал: Управління міською інфраструктурою, IoT-датчики.
   - Недоліки: Складність впровадження, орієнтованість на великі міста.
2. "Дія.Місто" (Україна)
   - Функціонал: Електронні послуги для громадян.
   - Недоліки: Не охоплює внутрішньобудинкові процеси ОСББ.
3. "Київ Digital"
   - Функціонал: Відеоспостереження, звітність.
   - Недоліки: Немає інструментів для комунікації жильців.

2.2.3. Спеціалізовані системи відеоспостереження

1. iVideon
   - Функціонал: Хмарне відеоспостереження.
   - Недоліки: Немає інтеграції з комунікаційними платформами.
2. Milestone XProtect
   - Функціонал: Професійне відеоаналітичне ПЗ.
   - Недоліки: Висока вартість, складність налаштування.

2.3. Порожнини ринку та переваги моєї платформи

Проведений аналіз показав, що існуючі рішення мають такі проблеми:

- Фрагментація: Жильці змушені використовувати кілька додатків для різних потреб.
- Відсутність інтеграції: Відеоспостереження, чати та управлінські інструменти працюють ізольовано.
- Високі витрати: Іноземні аналоги часто недоступні через ціну або відсутність локалізації.

Наша платформа заповнює ці порожнини завдяки:

- Єдиному інтерфейсу для всіх функцій (чат, камери).
- Відкритій API-архітектурі, що дозволяє підключати додаткові сервіси.
- Адаптивності до потреб малих ОСББ та великих ОТГ.
- Локалізації під українські законодавчі вимоги (наприклад, згідно із законом про ОСББ).

2.4. Висновки

Актуальність розробки підтверджується гострою потребою ОСББ та ОТГ в інтегрованих цифрових рішеннях, які поєднують управління, безпеку та комунікацію. Існуючі аналоги не задовольняють усі потреби через роз’єднаність функцій або високу вартість. Запропонована платформа на базі Java Spring + React пропонує унікальний підхід, що відкриває нові можливості для автоматизації життя громад.

Перспективи розвитку:

- Додавання AI-аналітики для відео (детекція підозрілих об’єктів).
- Інтеграція з державними е-сервісами (наприклад, "Дія").
- Розширення для "розумних будинків" (IoT-датчики води, світла, газу тощо).
- Додавання платіжних систем для оплати комунальних послуг.

3. ПОЧАТОК РОБОТИ ТА РЕАЛІЗАЦІЯ BACKEND

3.1 Вибір технологій та програмного забезпечення для реалізації проекту.

На етапі планування розробки універсальної платформи для ОТГ та ОСББ було проведено комплексний аналіз сучасних технологій. В результаті для реалізації проекту обрано наступний стек:

- Backend: Java Spring Boot
- Frontend: React.js

Цей вибір обґрунтований низкою технічних та практичних факторів.

3.1.2. Обґрунтування вибору Java Spring для backend

Переваги Java як мови програмування:

- Крос-платформенність
- Надійність і безпека
- Висока продуктивність
- Багата екосистема

Переваги Spring Boot:

- Швидке розгортання
- Модульність
- Потужна підтримка безпеки
- Інтеграція з базами даних
- REST API support
- Тестування

Відповідність вимогам проекту:

- Обробка великої кількості запитів від користувачів
- Необхідність стабільної роботи з відеопотоками
- Складні бізнес-процеси управління ОСББ/ОТГ
- Висока важливість безпеки даних

3.1.3. Обґрунтування вибору React для frontend

Ключові переваги React:

- Компонентний підхід
- Віртуальний DOM
- Одностороннє зв'язування даних
- Багата екосистема
- Широка спільнота

Відповідність вимогам проекту:

- Необхідність динамічного інтерфейсу з багатьма інтерактивними елементами
- Важливість швидкої реакції на дії користувача
- Потреба в адаптивному дизайні для різних пристроїв
- Можливість легкого масштабування інтерфейсу

3.1.4. Порівняння з альтернативними технологіями

Альтернативи для backend:

- Node.js: менш підходить для складних бізнес-процесів, проблеми з продуктивністю CPU-bound операцій
- Python/Django: нижча продуктивність, менш підходить для високонавантажених систем

- PHP/Laravel: менш сучасна архітектура, обмежені можливості для складних систем

Альтернативи для frontend:

- Angular: порівняно з React важчий в освоєнні.
- Vue.js: менша екосистема, хоча й хороша альтернатива
- JavaScript: значно збільшує час розробки, немає переваг компонентного підходу

3.1.5. Висновки

Вибір Java Spring Boot для backend та React.js для frontend є оптимальним рішенням для даного проекту, оскільки:

1. Забезпечує високу продуктивність і масштабованість
2. Дозволяє ефективно реалізувати складну бізнес-логіку
3. Надає всі необхідні інструменти для розробки безпечного та надійного рішення
4. Має велику спільноту та підтримку
5. Дозволяє швидко адаптуватися до змінних вимог

Цей технологічний стек особливо добре підходить для:

- Систем з високими вимогами до безпеки
- Проектів зі складними бізнес-процесами
- Рішень, що потребують стабільної обробки великих обсягів даних
- Додатків з багатим інтерактивним інтерфейсом

У подальшому такий стек технологій дозволить дуже просто розширювати функціонал платформи та інтегрувати нові модулі.

3.2 Початкове налаштування проекту

1. Архітектура проекту на базі Maven

Проект розроблено з використанням Maven як інструменту для збірки та управління залежностями. Це дозволило:

- Стандартизувати структуру проекту згідно з Maven Convention
- Автоматизувати процес збирання, тестування та розгортання
- Ефективно керувати залежностями через централізований pom.xml
- Підтримувати модульну архітектуру (при необхідності розбиття на submodules)

Ключові переваги обраної архітектури:

- Чітке розділення на рівні (controllers, services, repositories)
- Легка підтримка та масштабування коду
- Можливість підключення додаткових Maven-плагінів (Lombok)

Усі залежності проекту додані у додаток А.

Як основну базу даних обрано PostgreSQL через:

- Надійність та ACID-сумісність
- Підтримку JSON для зберігання конфігурацій камер
- Масштабованість (від малих ОСББ до великих ОТГ)

Саме підключення до БД реалізовано у файлі application.properties та наведено нижче на Рис. 3.2.1

```
spring.application.name=Cameras

spring.datasource.url=jdbc:postgresql://localhost:5432/video_surveillance
spring.datasource.username=postgres
spring.datasource.password=root
spring.jpa.hibernate.ddl-auto=update

spring.mvc.cors.allowed-origins=http://localhost:3000
spring.mvc.cors.allowed-methods=GET,POST,PUT,PATCH,DELETE
spring.mvc.cors.allowed-headers=*

hls.output-directory=./media/streams
hls.segment-time=2
```

Рис. 3.2.1 – налаштування застосунку

Для уникнення CORS-конфліктів було створено файл CorsConfig.java та для коректної роботи чату через web-socket реалізовано клас WebSocketConfig.java, лістинги обох конфігураційних файлів додано у додаток Б.

Обрана архітектура на базі Maven + Spring Boot + PostgreSQL дозволила:

- Чітко організувати структуру проекту
- Ефективно керувати залежностями
- Забезпечити надійне зберігання даних про камери
- Легко масштабувати систему при необхідності

Блок-схеми з логіками для камер та чату наведені нижче на рис. 3.2.1 та 3.2.2.

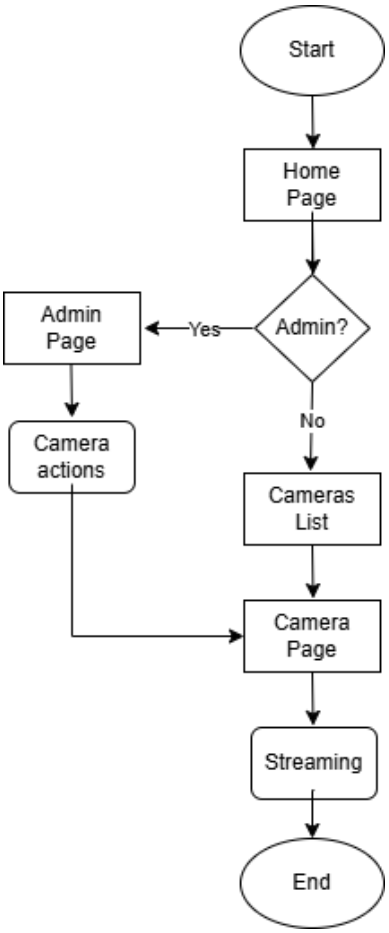

Рис. 3.2.1 – Блок-схема логіки камер

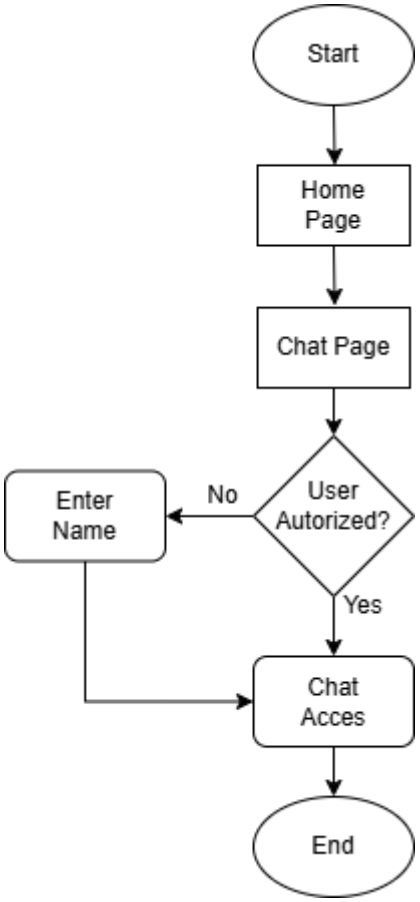

Рис. 3.2.2 – Блок-схема логіки чату

3.3 Створення усіх директорій

Backend частина проекту має такі основні частини:

- Controller (класи для контролю різними частинами застосунку наприклад камерами, автентифікацією, чатом та hls ресурсами)
  - CameraController
  - ChatController
  - ChatAuthController
  - HlsResourceController
- Models (класи з моделями елементів, що зберігаються у БД)
  - Camera
  - ChatMassage
- Repository (інтерфейси для спрощення роботи з БД)
  - CameraRepository

- ChatMassageRepository

- Service (класи що допомогають контролерам використовувати інтерфейси та містять основну логіку роботи програми)
  - CameraService
  - ChatHandler
  - ChatService
  - StreamService

На рис. 3.3.1 наведено повну архітектуру backend. У додаток В винесено діаграми класів для елементів камери та чату.

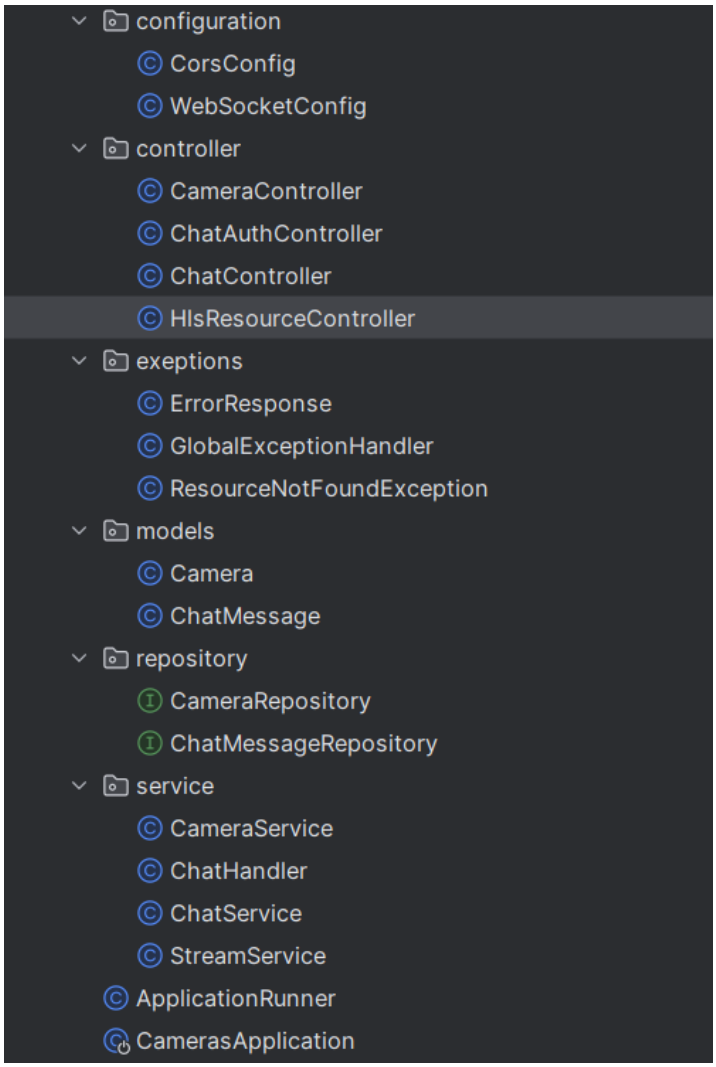

Рис. 3.3.1 – архітектура backend

3.4 Реалізація роботи з камерами

Основна задача даної роботи це створити застосунок для контролю камерами, тому треба успішно це реалізувати. Я вирішив використовувати RTSP-камери, тому що:

1. Відкритий стандарт
   - RTSP — це відкритий протокол, що підтримується багатьма пристроями та ПЗ (наприклад, VLC, Milestone, Blue Iris, ZoneMinder).
   - Не залежить від виробника, на відміну від пропрієтарних протоколів (наприклад, деякі камери Hikvision чи Dahua мають власні API).
2. Низька затримка (лаг)
   - Оптимізований для потокового відео в реальному часі, тому затримка часто нижча, ніж у HTTP/MJPEG-потоків.
3. Підтримка мультипотоковості
   - Може передавати кілька потоків (наприклад, основний у високій роздільній здатності та субпотік у низькій).
   - Дозволяє економити пропускну здатність.
4. Сумісність з NVR та VMS
   - Легко інтегрується в існуючі системи.
5. Підтримка RTP/RTCP
   - Використовує RTP для передачі даних і RTCP для контролю якості, що забезпечує стабільність зв’язку.

Проте головним недоліком є те, що браузери не підтримують rtsp-поток, тому треба реалізувати перетворення у hls-поток та ретранслювати вже його. Це все реалізовано у backend частині та приведено у додатках Г (модель камери та сервіс) та Ґ (сервіс по перетворенню потоку та контролер для ретрансляції).

Для перетворення rtsp-потоку у hls було використано FFmpeg.

FFmpeg — це потужний набір бібліотек і інструментів з відкритим кодом (LGPL/GPL) для обробки мультимедіа. Він дозволяє записувати, конвертувати, транслювати, редагувати та аналізувати аудіо- та відеофайли майже у будь-якому форматі.

Основні можливості FFmpeg

1. Конвертація форматів

- Перетворення відео/аудіо між різними кодеками (наприклад, MP4 → MKV, H.264 → H.265, AAC → MP3).
- Підтримка сотень форматів (AVI, MP4, FLV, WebM, MOV тощо).

2. Транскодування (перекодування)

- Зміна параметрів відео (бітрейт, роздільна здатність, частота кадрів).

3. Запис і трансляція потоків

- Захоплення відео з вебкамер, ТВ-тюнерів, RTSP-камер.
- Трансляція через RTMP, HLS, SRT тощо (використовується на Twitch, YouTube).

4. Обробка мультимедіа

- Обрізання, склеювання, накладання ефектів (текст, водяні знаки).
- Видалення аудіодоріжки, додавання субтитрів.

5. Аналіз та відладка

- Вивчення технічних параметрів файлів (кодеки, бітрейт, метадані).

4. РЕАЛІЗАЦІЯ FRONTEND

Для фронтенд-частини платформи було обрано React.js з використанням сучасного стеку технологій:

- Create React App (CRA) як стартовий шаблон
- Функціональні компоненти з хуками
- Context API + useReducer для глобального стану (на початковому етапі)
- React Router v6 для навігації

4.1 Додання залежностей у react

Для комфортної роботи з камерами і чатом було додано такі бібліотеки і залежності, як:

- StompJS
- Router DOM
- Axios
- Hls.js
- SockJS client

На рис. 4.1.1 показано всі додані заленості

Рис. 4.1.1. – залежності підключені до frontend частини

4.2 Наповнення додатку

Додаток включає в себе 4 сторінки:

- Home.js (головна сторінка на якій є список камер та можливість перейти на інші сторінки)
- CameraPage.jsx (сторінка з камерою де можна взаємодіяти з нею)
- Chat.js (сторінка з чатом, при першому заході проходить псевдо авторизація для визначення ім’я користувача)
- AdminPage.js (сторінка адміністратора для додавання та видалення камер, також адмін може вимкнути або увімкнути камеру. Перехід на цю сторінку можливий лише якщо дописати у посиланні /admin)

На рис. 4.2.1 – 4.2.3 надані скріншоти сторінок та у додатку Д надано лістинг сторінок CameraPage.jsx та AdminPage.js.

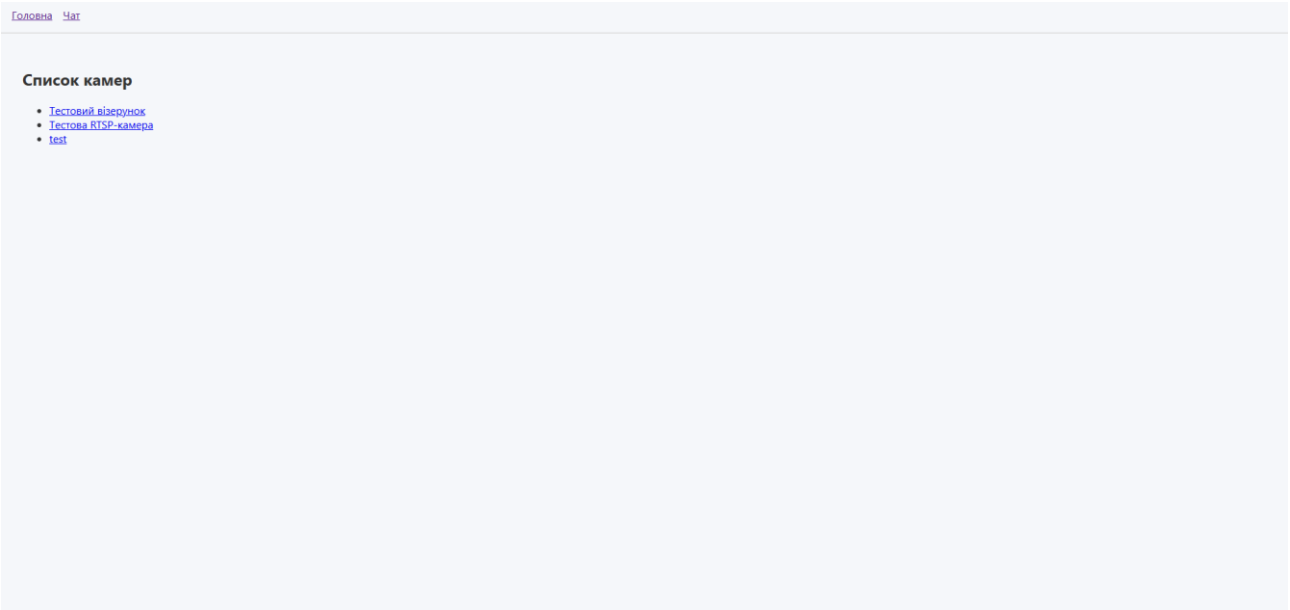

Рис. 4.2.1 – скріншот сторінки Home.js

Рис. 4.2.2 – скріншот сторінки AdminPage.js

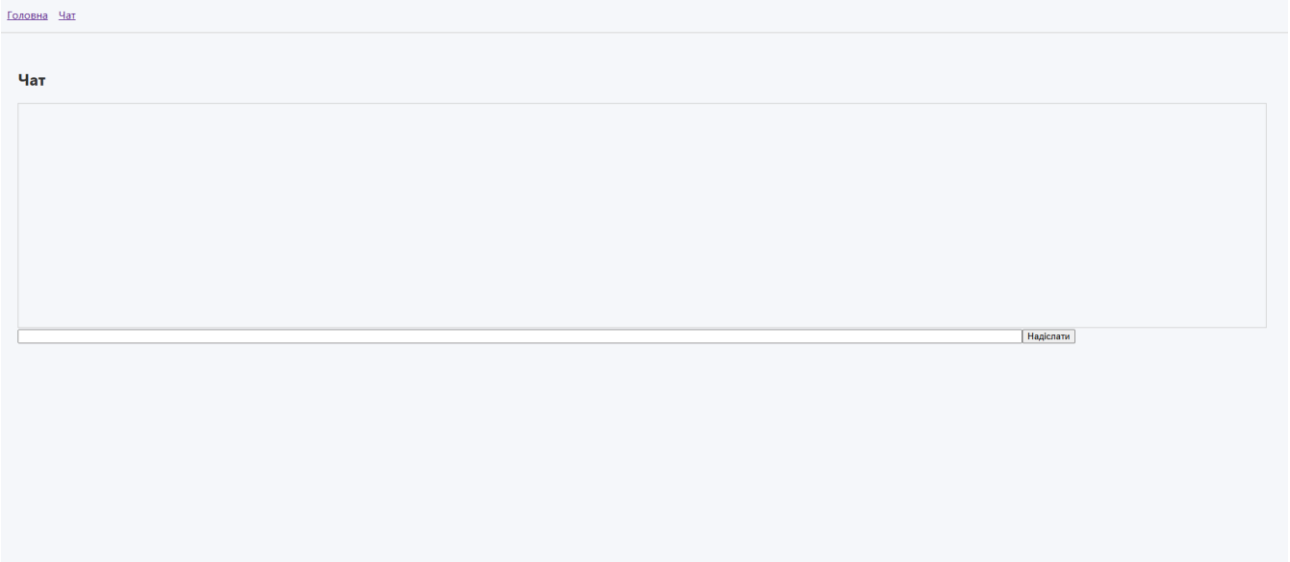

Рис. 4.2.3 – скріншот сторінки Chat.js

4.3 Створення плеєру hls-потоків

Для коректного відображення записів з камери було реалізовано плеєр, шо пов’язаний з backend. Після відкриття сторінки з певною камерою відправляється запит на сервер з якого у відповідь приходить уже перетворений hls-потік, реалізація класу додана до додатку Е.

ВИСНОВКИ

У даній науковій роботі було розроблено універсальну платформу для управління ОСББ та ОТГ, яка інтегрує функції відеоспостереження, комунікації між жильцями та адміністрацією, а також інструменти для ефективного управління комунальною інфраструктурою. Використання сучасних технологій, таких як Java Spring Boot для бекенду та React.js для фронтенду, дозволило створити масштабоване та безпечне рішення, адаптоване до потреб українських громад.

Основні досягнення роботи:

- Реалізація модульної архітектури з чітким розділенням компонентів (Maven для бекенду, Create React App для фронтенду).
- Інтеграція відеоспостереження з можливістю перегляду потоків у реальному часі та архівуванням записів (JavaCV, WebSocket).
- Розробка інтуїтивного інтерфейсу на базі React з використанням Material-UI, що забезпечує зручну взаємодію для різних груп користувачів (адміністраторів, жильців).
- Налаштування безпеки (JWT-аутентифікація, Spring Security, рольовий доступ).
- Використання PostgreSQL як надійної СУБД з підтримкою геоданих для мапування камер.

Переваги запропонованого рішення:

- Економія часу та ресурсів завдяки автоматизації рутинних процесів (документообіг, збори, сповіщення).
- Прозорість управління через централізований доступ до інформації (протоколи, відеоархіви).
- Гнучкість — платформа легко адаптується під потреби конкретних ОСББ/ОТГ.

- Скорочення витрат у порівнянні з іноземними аналогами (локалізація, open-source технології).

Перспективи розвитку:

- Додавання AI-аналітики для автоматичного виявлення підозрілих подій на відео.
- Інтеграція з державними е-сервісами (наприклад, "Дія", "Е-майно").
- Розширення для "розумних будинків" (датчики води, світла, температури).
- Впровадження мобільного додатку (React Native) для зручності жильців.

Заключна думка

Робота довела ефективність обраного підходу до створення цифрових рішень для місцевих громад. Запропонована платформа не лише вирішує актуальні проблеми ОСББ та ОТГ, але й стає основою для побудови сучасних "smart city" екосистем в Україні. Подальші дослідження можуть бути спрямовані на вдосконалення штучного інтелекту для аналізу даних та розширення функціоналу для державно-комунального сектору.

Ключове значення роботи полягає в тому, що вона демонструє, як технології можуть спростити життя людей та підвищити ефективність управління місцевими ресурсами — від окремого будинку до цілої територіальної громади.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Sommerville I. *Software Engineering*. Boston: Pearson, 2015. 816 с.
2. Нові теоретичні засади технології виробництва сімейств програмних систем у контексті генерувального програмування: кол. монографія / К.М. Лаврищева та ін. Київ: Ін-т програм. систем НАН України, 2011. 277 с. URL: http://cyb.univ.kiev.ua/library/books/lavrishcheva-5.pdf (дата звернення: 12.03.2025).
3. Tkachuk N., Sokol V., Glukhovtsova K. *An Integrated Development Framework for Advanced IT-Service Management: Proof-of-Concept Project in Universities Domain* // V. Ermolaev et al. (Eds.): ICTERI 2013: Revised Selected Papers. Berlin: Springer-Verlag Heidelberg, 2013. Vol. 412. P. 50–69.
4. *Про об'єднання співвласників багатоквартирного будинку*: Закон України від 14.05.2015 № 417-VIII. Київ: ВР України, 2015. 12 с.
5. *ДСТУ 8302:2015. Інформація та документація. Бібліографічне посилання. Загальні положення та правила складання*. Київ: ДП «УкрНДНЦ», 2016. 17 с. URL: http://lib.pu.if.ua/files/dstu-8302-2015.pdf (дата звернення: 20.03.2025).
6. *Правила технічної експлуатації систем відеоспостереження*. Затв. Держспецзв'язку України 15.03.2020. Київ: Техніка, 2020. 89 с.
7. *ДСТУ ISO 9001:2015. Системи управління якістю. Вимоги*. Київ, 2015. 30 с.
8. Мартінкус І.О. *Інформаційна технологія розробки лінійок програмних продуктів на основі методів та засобів доменного моделювання*: автореф. дис. канд. техн. наук. Харків, 2017. 20 с.
9. *Наукові публікації НАН України в галузі інформаційних технологій*. Київ, 2023. URL: http://www.nas.gov.ua/publications (дата звернення: 22.03.2025).
10. *Object Management Group (OMG)*. Needham, Massachusetts, USA, 2024. URL: https://www.omg.org/ (дата звернення: 25.03.2025).
11. *Conferences on Software Engineering and Smart Cities*. URL: http://www.conference-service.com/conferences/software-engineering.html (дата звернення: 25.03.2025).

12. Лаврищева К.М. *Основи програмної інженерії* // Факультет комп'ютерних наук та кібернетики КНУ ім. Т. Шевченка. URL: http://cyb.univ.kiev.ua/library/books/lavrishcheva-6.pdf (дата звернення: 26.03.2025).
13. *Розумні міста: досвід впровадження IoT-рішень* // Хабрахабр. URL: https://habr.com/ru/post/215117 (дата звернення: 27.03.2025).
14. *Hypertext Transfer Protocol – HTTP/2* / M. Belshe та ін. // IETF Documents. URL: https://tools.ietf.org/html/rfc7540 (дата звернення: 28.03.2025).
15. *Spring Framework Official Documentation*. URL: https://spring.io/projects/spring-framework (дата звернення: 12.04.2025).
16. *FFmpeg Official Website*. URL: https://ffmpeg.org/ (дата звернення: 20.04.2025).
17. *React.js Documentation*. URL: https://reactjs.org/docs/getting-started.html (дата звернення: 29.04.2025).
18. *PostgreSQL 15 Documentation*. URL: https://www.postgresql.org/docs/15/ (дата звернення: 29.04.2025).
19. *WebSocket API Specification*. URL: https://html.spec.whatwg.org/multipage/web-sockets.html (дата звернення: 03.05.2025).
20. *Дія.Місто: цифрові сервіси для громад*. URL: https://city.diia.gov.ua/ (дата звернення: 04.05.2025).
21. *Smart City Hub: IoT Solutions for Urban Infrastructure*. URL: https://www.smartcityhub.com/ (дата звернення: 05.05.2025).
22. *BuildingLink: Property Management Software*. URL: https://www.buildinglink.com/ (дата звернення: 06.05.2025).

ДОДАТОК А

Лістинг А.1 – конфігураційний файл pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>3.4.2</version>
        <relativePath/> <!-- lookup parent from repository -->
    </parent>
    <groupId>com.hw</groupId>
    <artifactId>Cameras</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <name>Cameras</name>
    <description>Cameras</description>
    <url/>
    <licenses>
        <license/>
    </licenses>
    <developers>
        <developer/>
    </developers>
    <scm>
        <connection/>
        <developerConnection/>
        <tag/>
        <url/>
    </scm>
    <properties>
        <java.version>17</java.version>
    </properties>
    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-data-
jpa</artifactId>
        </dependency>
        <dependency>
            <groupId>org.postgresql</groupId>
            <artifactId>postgresql</artifactId>
        </dependency>
        <dependency>
            <groupId>org.projectlombok</groupId>
```

```
            <artifactId>lombok</artifactId>
            <scope>provided</scope>
        </dependency>
        <dependency>
            <groupId>org.apache.velocity</groupId>
            <artifactId>velocity</artifactId>
            <version>1.7</version>
        </dependency>
        <dependency>
            <groupId>org.openjfx</groupId>
            <artifactId>javafx-graphics</artifactId>
            <version>20.0.1</version>
        </dependency>
        <dependency>
            <groupId>jakarta.validation</groupId>
            <artifactId>jakarta.validation-api</artifactId>
            <version>3.0.2</version>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-
websocket</artifactId>
        </dependency>
        <dependency>
            <groupId>org.junit.jupiter</groupId>
            <artifactId>junit-jupiter</artifactId>
            <scope>test</scope>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-test</artifactId>
            <version>3.4.2</version>
            <scope>test</scope>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-
websocket</artifactId>
        </dependency>
        <dependency>
            <groupId>com.h2database</groupId>
            <artifactId>h2</artifactId>
            <scope>runtime</scope>
        </dependency>
    </dependencies>

    <build>
        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-compiler-plugin</artifactId>
                <configuration>
                    <annotationProcessorPaths>
```

```
                        <path>
                            <groupId>org.projectlombok</groupId>
                            <artifactId>lombok</artifactId>
                            <version>1.18.30</version>
                        </path>
                    </annotationProcessorPaths>
                </configuration>
            </plugin>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-
plugin</artifactId>
                <configuration>
                    <excludes>
                        <exclude>
                            <groupId>org.projectlombok</groupId>
                            <artifactId>lombok</artifactId>
                        </exclude>
                    </excludes>
                </configuration>
            </plugin>
        </plugins>
    </build>

</project>
```

ДОДАТОК Б

Лістинг Б.1 – код конфігураційного файлу CorsConfig

```
@Configuration
public class CorsConfig {

    @Bean
    public CorsFilter corsFilter() {
        CorsConfiguration config = new CorsConfiguration();

config.setAllowedOrigins(Arrays.asList("http://localhost:3000"))
; // точне походження
        config.setAllowCredentials(true);
        config.addAllowedHeader("*");
        config.addAllowedMethod("*");

        UrlBasedCorsConfigurationSource source = new
UrlBasedCorsConfigurationSource();
        source.registerCorsConfiguration("/**", config); //
глобально

        return new CorsFilter(source);
    }
}
```

Лістинг Б.2 – код конфігураційного файлу WebSocketConfig

```
@Configuration
@EnableWebSocketMessageBroker
public class WebSocketConfig implements
WebSocketMessageBrokerConfigurer {

    @Override
    public void registerStompEndpoints(StompEndpointRegistry
registry) {
        registry.addEndpoint("/ws")
                .setAllowedOriginPatterns("*")
                .withSockJS();
    }

    @Override
    public void configureMessageBroker(MessageBrokerRegistry
config) {
        config.enableSimpleBroker("/topic");
        config.setApplicationDestinationPrefixes("/app");
    }
}
```

ДОДАТОК В

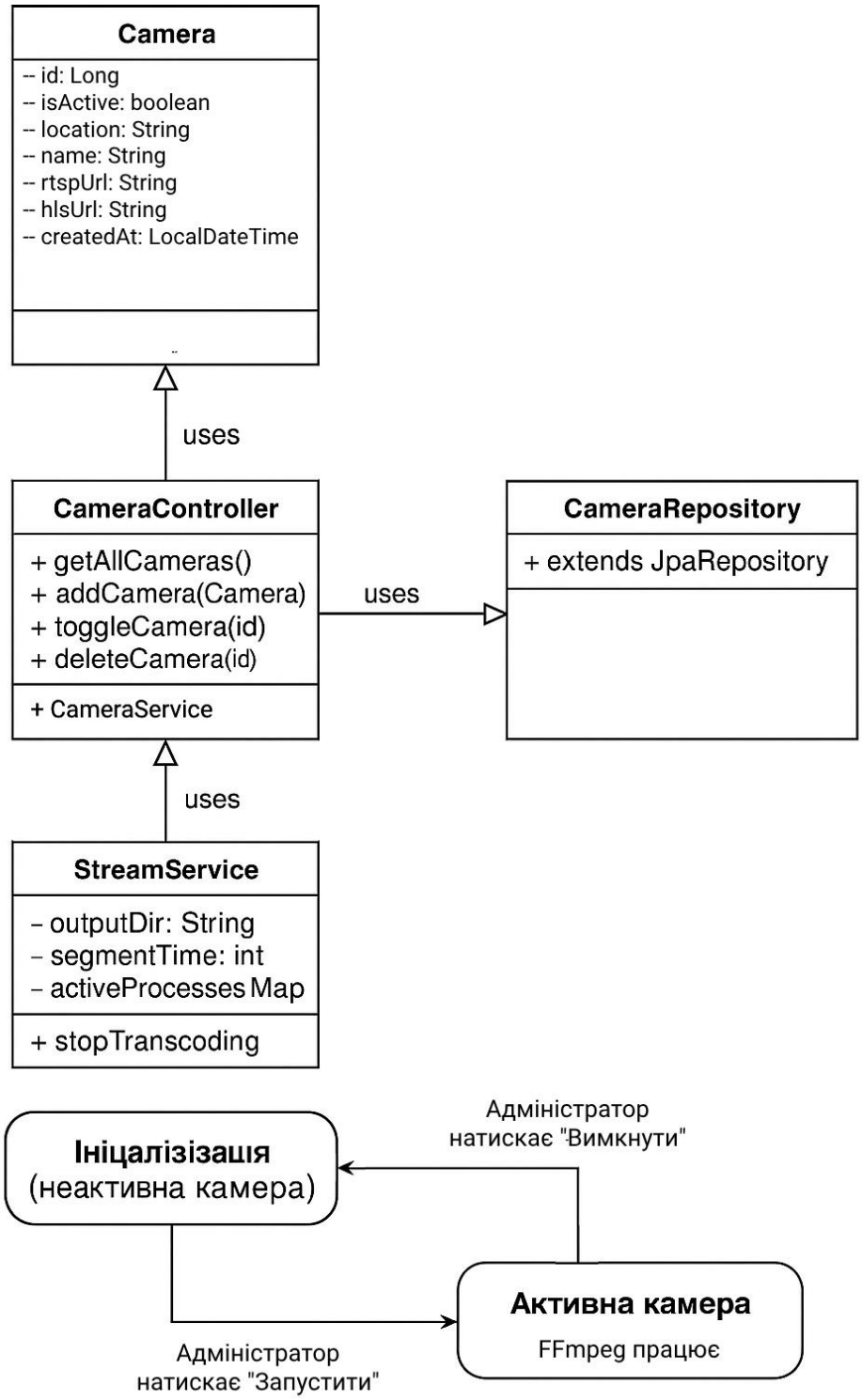

Рис. В.1 – діаграма класів елементу камер

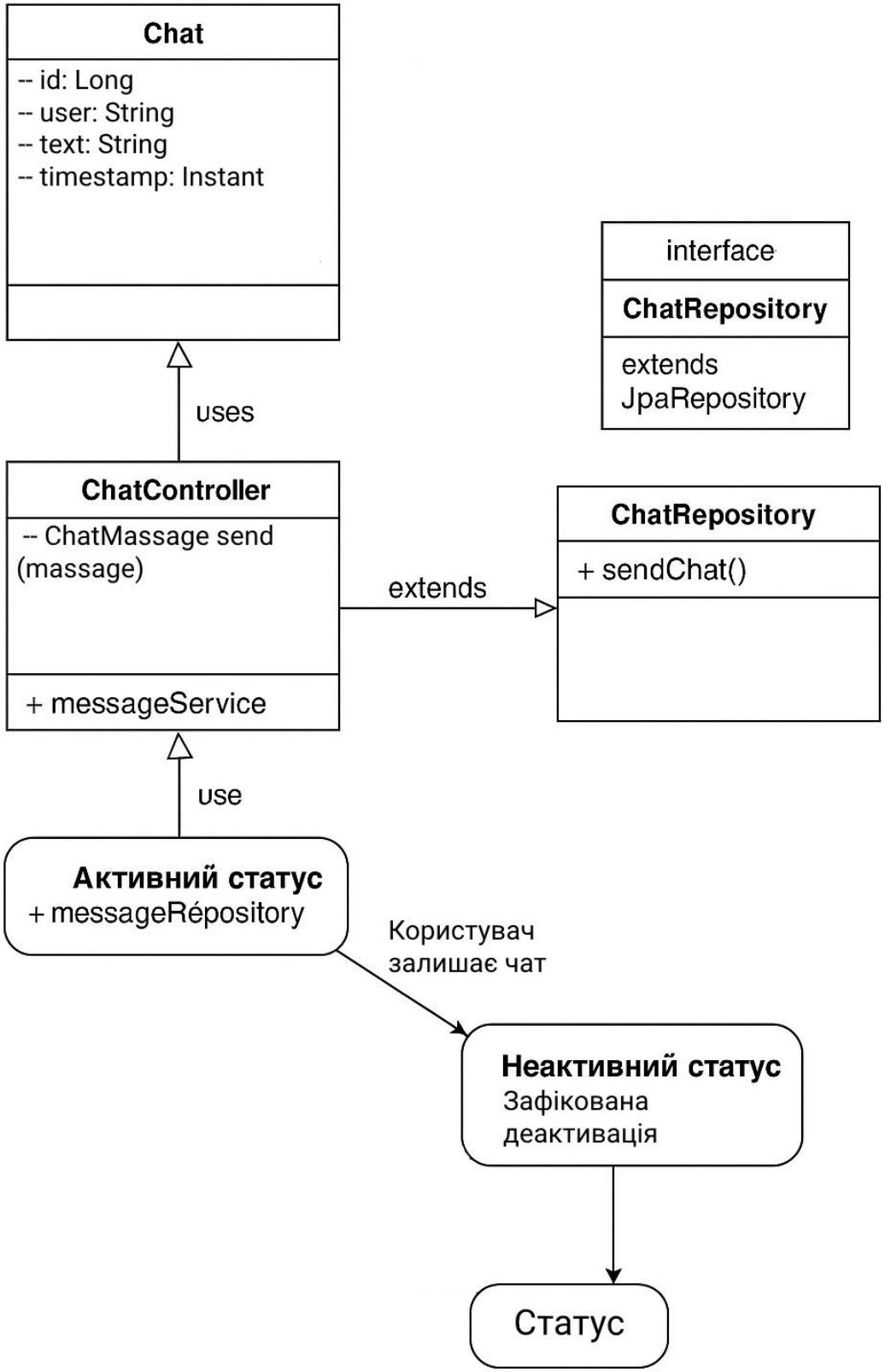

Рис. В.2 – діаграма класів елементу чату

ДОДАТОК Г

Лістинг Г.1 – код класу Camera

```
@Entity
@Data
public class Camera {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private boolean isActive;
    private String location;

    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getRtspUrl() {
        return rtspUrl;
    }

    public void setRtspUrl(String rtspUrl) {
        this.rtspUrl = rtspUrl;
    }

    public String getHlsUrl() {
        return hlsUrl;
    }

    public void setHlsUrl(String hlsUrl) {
        this.hlsUrl = hlsUrl;
    }

    public boolean isActive() {
        return isActive;
    }

    public void setActive(boolean active) {
        isActive = active;
    }
```

```
    @NotBlank(message = "Назва обов'язкова")
    private String name;

    @NotBlank(message = "RTSP URL обов'язковий")
    private String rtspUrl;

    private String hlsUrl;
    @Column(name = "created_at")
    private LocalDateTime createdAt;


    @PrePersist
    protected void onCreate() {
        createdAt = LocalDateTime.now();
    }
    public String getLocation() {
        return location;
    }

    public void setLocation(String location) {
        this.location = location;
    }
}
```

Лістинг Г.2 – код сервісу CameraService

```
@Service
public class CameraService {
    private final CameraRepository cameraRepository;

    public CameraService(CameraRepository cameraRepository) {
        this.cameraRepository = cameraRepository;
    }

    public List<Camera> getAllCameras() {
        return cameraRepository.findAll();
    }

    public Camera addCamera(Camera camera) {
        return cameraRepository.save(camera);
    }

    @Transactional
    public Camera toggleCamera(Long id) {
        Camera camera = cameraRepository.findById(id)
                .orElseThrow(() -> new
EntityNotFoundException("Камера не знайдена"));

        camera.setActive(!camera.isActive());
        return cameraRepository.save(camera);
    }
```

```
    public void deleteCamera(Long id) {
        cameraRepository.deleteById(id);
    }
}
```

ДОДАТОК Ґ

Лістинг Ґ.1 – код сервісу для перетворення потоків SreamService

```
@Service
public class StreamService {
    private static final Logger log =
LoggerFactory.getLogger(StreamService.class);

    @Value("${hls.output-directory}")
    private String outputDir;

    @Value("${hls.segment-time}")
    private int segmentTime;

    private final Map<Long, Process> activeProcesses = new
ConcurrentHashMap<>();

    public String startTranscoding(Camera camera) throws
IOException {
        String streamKey = "stream_" + camera.getId();
        String hlsPath = Paths.get(outputDir,
streamKey).toString();

        Files.createDirectories(Paths.get(hlsPath));

        String command = String.format(
                "ffmpeg -i %s -c:v libx264 -c:a aac -hls_time %d
-hls_list_size 5 -hls_segment_filename %s/%%03d.ts -f hls
%s/playlist.m3u8",
                camera.getRtspUrl(),
                segmentTime,
                hlsPath,
                hlsPath
        );

        log.info("Виконується команда: {}", command);

        Process process = Runtime.getRuntime().exec(command);
        activeProcesses.put(camera.getId(), process);

        new Thread(() -> {
            try (BufferedReader reader = new BufferedReader(new
InputStreamReader(process.getErrorStream()))) {
                String line;
                while ((line = reader.readLine()) != null) {
                    log.error("[FFMPEG {}] {}", camera.getId(),
line);
                }
            } catch (IOException e) {
                log.error("Помилка читання stderr ffmpeg", e);
            }
        }).start();
```

```
        String hlsUrl = "/hls/" + streamKey + "/playlist.m3u8";
        log.info("Started transcoding for camera {}: {}",
camera.getId(), hlsUrl);

        return hlsUrl;
    }

    public void stopTranscoding(Long cameraId) {
        Process process = activeProcesses.get(cameraId);
        if (process != null) {
            process.destroy();
            activeProcesses.remove(cameraId);
            log.info("Stopped transcoding for camera {}",
cameraId);
        }
    }
}
```

Лістинг Ґ.2 – код контролеру hls-потоків HlsResourceController

```
@RestController
@RequestMapping("/hls")
public class HlsResourceController {

    @Value("${hls.output-directory}")
    private String outputDir;

    @GetMapping("/{streamKey}/playlist.m3u8")
    public ResponseEntity<Resource> getPlaylist(@PathVariable
String streamKey) {
        Path path = Paths.get(outputDir, streamKey,
"playlist.m3u8");
        return serveFile(path, "application/vnd.apple.mpegurl");
    }

    @GetMapping("/{streamKey}/{segment}.ts")
    public ResponseEntity<Resource> getSegment(
            @PathVariable String streamKey,
            @PathVariable String segment
    ) {
        Path path = Paths.get(outputDir, streamKey, segment +
".ts");
        return serveFile(path, "video/MP2T");
    }

    private ResponseEntity<Resource> serveFile(Path path, String
contentType) {
        try {
            Resource resource = (Resource) new
UrlResource(path.toUri());
            return ResponseEntity.ok()
```

```
.contentType(MediaType.parseMediaType(contentType))
                    .body(resource);
        } catch (Exception e) {
            return ResponseEntity.notFound().build();}}}
```

ДОДАТОК Д

Лістинг Д.1 – код класу CameraPage

```
const CameraPage = () => {
  const { id } = useParams();
  const [camera, setCamera] = useState(null);

  useEffect(() => {
    fetch(`/api/cameras/${id}`)
      .then((res) => res.json())
      .then(setCamera);
  }, [id]);

  if (!camera) return <div>Завантаження...</div>;

  return (
    <div style={{ padding: '2rem' }}>
      <h2>{camera.name}</h2>
      <video src={camera.streamUrl} controls autoPlay />
    </div>
  );
};
export default CameraPage;
```

Лістинг Д.2 – код класу AdminPage

```
function AdminPage() {
  const [cameras, setCameras] = useState([]);
  const [newCamera, setNewCamera] = useState({ name: '',
location: '', rtspUrl: '' });

  useEffect(() => {
    fetchCameras();
  }, []);
```

```
const fetchCameras = async () => {
  try {
    const response = await axios.get('/api/cameras');
    setCameras(response.data);
  } catch (error) {
    console.error('Помилка при отриманні камер:', error);
  }
};

const handleAddCamera = async () => {
  try {
    await axios.post('/api/cameras', newCamera);
    setNewCamera({ name: '', location: '', rtspUrl: '' });
    fetchCameras();
  } catch (error) {
    console.error('Помилка при додаванні камери:', error);
  }
};

const handleDeleteCamera = async (id) => {
  try {
    await axios.delete(`/api/cameras/${id}`);
    fetchCameras();
  } catch (error) {
    console.error('Помилка при видаленні камери:', error);
  }
};

const handleToggleCamera = async (id) => {
  try {
    await axios.patch(`/api/cameras/${id}/toggle`);
    fetchCameras();
  } catch (error) {
```

```
      console.error('Помилка при перемиканні камери:', error);
    }
  };

  return (
    <div style={{ padding: '2rem' }}>
      <h2>Адмін-панель камер</h2>

      <div style={{ marginBottom: '2rem' }}>
        <h3>Додати камеру</h3>
        <input
          type="text"
          placeholder="Назва"
          value={newCamera.name}
          onChange={(e) => setNewCamera({ ...newCamera, name:
e.target.value })}
          style={{ marginRight: '1rem' }}
        />
        <input
          type="text"
          placeholder="Локація"
          value={newCamera.location}
          onChange={(e) => setNewCamera({ ...newCamera, location:
e.target.value })}
          style={{ marginRight: '1rem' }}
        />
        <input
          type="text"
          placeholder="RTSP URL"
          value={newCamera.rtspUrl}
          onChange={(e) => setNewCamera({ ...newCamera, rtspUrl:
e.target.value })}
          style={{ marginRight: '1rem' }}
        />
```

```
        <button onClick={handleAddCamera}>Додати</button>
      </div>
      <h3>Список камер</h3>
      {cameras.map((camera) => (
        <div key={camera.id} style={{ borderBottom: '1px solid
#ccc', padding: '1rem 0' }}>
          <strong>{camera.name}</strong> ({camera.location})
          <div>
            Статус: {camera.active ? 'Активна' : 'Неактивна'}
          </div>
          <button onClick={() => handleToggleCamera(camera.id)}
style={{ marginRight: '1rem' }}>
            {camera.active ? 'Зупинити' : 'Запустити'}
          </button>
          <button onClick={() =>
handleDeleteCamera(camera.id)}>Видалити</button>
        </div>
      ))}
    </div>
  );
}
export default AdminPage;
```

ДОДАТОК Е

Лістинг Е.1 – код класу RTSPPlayer

```
const RTSPPlayer = ({ streamUrl, isActive }) => {
  const videoRef = useRef(null);
  const [error, setError] = useState(null);
  const [loading, setLoading] = useState(true);

  useEffect(() => {
    if (!isActive || !streamUrl) return;

    let hls;
    const initPlayer = () => {
      if (Hls.isSupported()) {
        hls = new Hls({
          maxMaxBufferLength: 30,
          maxBufferSize: 60 * 1000 * 1000,
          maxBufferHole: 0.5,
          enableWorker: true
        });

        hls.loadSource(streamUrl);
        hls.attachMedia(videoRef.current);

        hls.on(Hls.Events.MANIFEST_PARSED, () => {
          setLoading(false);
          videoRef.current.play().catch(e => {
            setError('Помилка відтворення: ' + e.message);
          });
        });

        hls.on(Hls.Events.ERROR, (event, data) => {
          if (data.fatal) {
            setError('Помилка потоку: ' + data.details);
```

```
            hls.destroy();
          }
        });
      } else if
(videoRef.current.canPlayType('application/vnd.apple.mpegurl'))
{
        videoRef.current.src = streamUrl;
        videoRef.current.addEventListener('loadedmetadata', () =>
{
          setLoading(false);
          videoRef.current.play();
        });
      }
    };

    initPlayer();

    return () => {
      if (hls) {
        hls.destroy();
      }
    };
  }, [streamUrl, isActive]);

  if (!isActive) return null;

  return (
    <div className="rtsp-player-container">
      {loading && <Loader />}
      {error ? (
        <div className="stream-error">
          <p>{error}</p>
          <button onClick={() => setError(null)}>🔄 Спробувати
знову</button>
        </div>
```

```
      ) : (
        <video
          ref={videoRef}
          controls
          autoPlay
          muted
          playsInline
          className="rtsp-video"
        />
      )}
    </div>
  );
};
export default RTSPPlayer;
```