УДК 004.7

Лупашина А.А., здобувач,
Фант М.О., к.філол.н., доцент,
Громський О.О., асистент,
Нерода С.І., асистент
Державний університет «Житомирська політехніка»

АРХІТЕКТУРА ТА ТЕХНІЧНА РЕАЛІЗАЦІЯ ВЕБ-СИСТЕМИ УПРАВЛІННЯ СТУДЕНТСЬКИМ ГУРТОЖИТКОМ

Розроблення сучасних веб-систем потребує поєднання формальних методів моделювання, гнучких архітектурних рішень та технологій, орієнтованих на масштабованість і безпеку. Система управління студентським гуртожитком є прикладом складної предметної області, де необхідно координувати численні взаємодіючі сутності: студентів, кімнати, гуртожитки, адміністративний персонал, заявки, бронювання та технічне обслуговування. Коректне математичне моделювання таких процесів дозволяє створити структуровану та передбачувану архітектуру програмного забезпечення.

На етапі проєктування виконано декомпозицію предметної області на підсистеми та побудовано UML-діаграми відповідно до стандартів OMG. Діаграма прецедентів описує ролі та сценарії взаємодії (реєстрація студента, бронювання кімнати, управління поселеннями, обробка заявок). Діаграма класів формалізує структуру сутностей із чітким виділенням атрибутів, асоціацій та обмежень цілісності. Діаграми активностей та послідовності дозволили сформувати математичну модель потоків даних і логічних станів системи, що мінімізує можливість виникнення суперечностей у бізнес-логіці.

На рисунку 1 наведено приклад UML діаграми прецедентів.

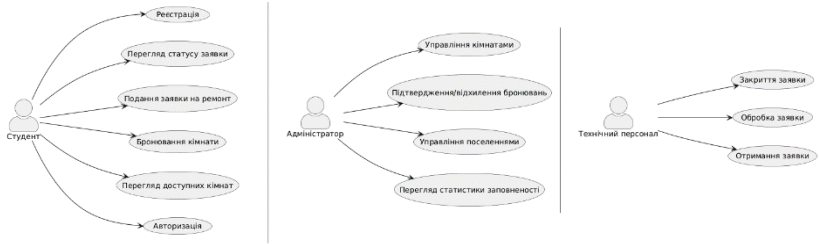

Рисунок 1 – Діаграма прецедентів системи управління гуртожитком

Архітектура системи реалізована у відповідності до принципів багаторівневого (Layered) та модульного підходу. Серверна частина побудована на NestJS, який забезпечує інверсію залежностей, модульність та суворе структурування бізнес-логіки. Система використовує RESTful API, а в перспективі передбачена можливість інтеграції GraphQL для оптимізації вибірок даних. Для зберігання даних обрано PostgreSQL, яке поєднує ACID-властивості, підтримку транзакцій, складні індекси та JSONB-поля для змішаних моделей даних. Взаємодія із СУБД реалізована через TypeORM, що забезпечує автоматичні міграції, lazy/eager loading та репозиторний підхід.

Для логування та моніторингу застосовано Winston і Prometheus, що дозволяє відслідковувати продуктивність, помилки та поведінку системи під навантаженням. У межах математичного аналізу продуктивності виконано оцінку часової складності основних операцій, зокрема пошуку доступних кімнат та оптимізації розподілу студентів за критеріями (курс, факультет, статус пільги).

Фронтенд реалізовано з використанням React, що дозволяє впроваджувати компонентний підхід та декларативну логіку побудови інтерфейсу. Для керування станом можуть бути використані Redux Toolkit або Zustand, залежно від складності глобальної логіки. Для оптимізації рендерингу застосовуються React.memo та динамічне завантаження модулів. Комунікація з сервером реалізована через Axios, а роутинг — за допомогою React Router 6.

Інтеграція формальних методів моделювання, сучасних JavaScript-технологій та архітектурних патернів дозволяє створити масштабовану, продуктивну й безпечну систему. Запропонована архітектура може бути розширена за рахунок мікросервісного підходу, контейнеризації у Docker та оркестрації через Kubernetes, що відкриває можливість адаптації системи для великих освітніх комплексів.

Список використаних джерел:

1. TypeORM Official Documentation. TypeORM Team [Електронний ресурс] – Режим доступу до ресурсу: https://typeorm.io

2. Kubernetes Documentation. Cloud Native Computing Foundation [Електронний ресурс] – Режим доступу до ресурсу: https://kubernetes.io/docs/

3. Design patterns: elements of reusable object-oriented software. Gamma E., Helm R., Johnson R., Vlissides J. [Електронний ресурс] – Режим доступу до ресурсу: https://www.oreilly.com/library/view/design-patterns/0201633612/