МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра програмної інженерії

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

на тему: **«РОЗРОБКА СИСТЕМИ ОНЛАЙН ОГОЛОШЕНЬ ЗАСОБАМИ ANGULAR ТА FIREBASE STORAGE»**

Виконав: студент 2 курсу, групи 8.1212-з
спеціальності 121 інженерія програмного забезпечення
(шифр і назва спеціальності)
освітньої програми інженерія програмного забезпечення
(назва освітньої програми)
Є.В. Возняк
(ініціали та прізвище)

Керівник доцент кафедри програмної інженерії, PhD Столярова А.В.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Рецензент доцент кафедри комп'ютерних наук, доцент, к.т.н. Матвіїшина Н.В.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя – 2023

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет математичний

Кафедра програмної інженерії

Рівень вищої освіти магістр

Спеціальність 121 інженерія програмного забезпечення
(шифр і назва)

Освітня програма інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри програмної інженерії, к.ф.-м.н., доцент

_____ Лісняк А.О.
(підпис)

“ ____ ” _____ 2023 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Возняку Євгену Валентиновичу
(прізвище, ім’я та по-батькові)

1. Тема роботи Розробка системи онлайн оголошень засобами Angular та Firebase Storage

керівник роботи Столярова Анастасія Валеріївна, PhD
(прізвище, ім’я та по-батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від « 01 » травня 2023 року № 643-с

2. Строк подання студентом роботи 27.11.2023 р.

3. Вихідні дані до роботи 1. Постановка задачі.
2. Перелік літератури.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Постановка задачі, аналіз предметної області.
2. Проєктування.
3. Реалізація та тестування.

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)
презентація

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання 03.05.2023 р.

К А Л Е Н Д А Р Н И Й П Л А Н

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Розробка плану виконання кваліфікаційної	17.05.2023	
	роботи магістра.		
2.	Збір вихідних даних та аналіз предметної	07.06.2023	
	області.		
3.	Обробка методичних та теоретичних	28.06.2023	
	джерел.		
4.	Специфікація вимог до системи. Робота над	30.08.2023	
	першим розділом.		
5.	Моделювання та проєктування системи.	11.10.2023	
	Робота над другим розділом.		
6.	Реалізація та тестування системи. Робота над	08.11.2023	
	третім розділом.		
7.	Оформлення та нормоконтроль	20.11.2023	
	кваліфікаційної роботи магістра.		
8.	Захист кваліфікаційної роботи.	15.12.2023	

Студент _____ Є.В. Возняк
(підпис) (ініціали та прізвище)

Керівник роботи _____ А.В. Столярова
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

Нормоконтролер _____ А.В. Столярова
(підпис) (ініціали та прізвище)

РЕФЕРАТ

Кваліфікаційна робота магістра «Розробка системи онлайн оголошень засобами Angular та Firebase Storage»: 52 с., 24 рис., 13 джерел, 4 додатка.

ANGULAR, API, FIREBASE, FRAMEWORK, RXJS, TYPESCRIPT, UML.

Об’єкт дослідження – система, інструменти для взаємодії Angular та Firebase, засоби взаємодії системи з користувачем.

Мета роботи – розробити систему онлайн оголошень.

Методи дослідження – моделювання, проєктування, програмний, аналітичний.

Вплив технологічного розвитку на сучасне суспільство важко переоцінити. Цей вплив охоплює всі сфери нашого життя, включаючи економіку, освіту, комунікацію, і розваги. Однією з найяскравіших проявів цього технологічного революційного періоду є системи онлайн оголошень.

Як правило, подібні системи мають різний функціонал, який охоплює різні аспекти взаємодії з формуванням оголошень. Але відсутність або обмеженість потрібного функціоналу не є виключенням. Одним з прикладів такої проблеми є обмеженість функціоналу подібних сервісів.

Таким чином, за результатами роботи створено зручну та ефективну систему онлайн оголошень засобами Angular та Firebase Storage.

SUMMARY

Master’s qualifying paper «Development of The Online Announcement Systems Using Angular and Firebase Storage»: 52 p., 24 figures, 13 references, 4 supplements.

ANGULAR, API, FIREBASE, FRAMEWORK, RXJS, TYPESCRIPT, UML.

The object of the study is system, tools for interaction between Angular and Firebase, means of interaction between the system and the user.

The aim of the study is to develop an online classifieds system.

The methods of research are modeling, design, software, analytical.

The impact of technological development on modern society is difficult to overestimate. This influence covers all areas of our lives, including the economy, education, communication, and entertainment. One of the most prominent manifestations of this technological revolution is online classifieds.

As a rule, such systems have different functionalities that cover different aspects of interaction with the formation of ads. But the absence or limited functionality is not an exception. One example of this problem is the limited functionality of such services.

Thus, based on the results of our work, we created a convenient and efficient online ad system using Angular and Firebase Storage.

ЗМІСТ

ВСТУП

Раніше, пошук товарів, послуг, або роботи вимагав багато часу та зусиль. Тепер же, завдяки системам онлайн оголошень, цей процес став значно ефективнішим і зручнішим. Користувачі можуть швидко переглядати доступні пропозиції, використовуючи фільтри та ключові слова для пошуку, що дозволяє знайти потрібну інформацію за лічені секунди.

Однією з ключових особливостей систем онлайн оголошень є їхній вплив на економіку. Завдяки цим платформам, малі підприємства та індивідуальні підприємці мають можливість ефективно рекламувати свої продукти та послуги. Це допомагає збільшити їхню видимість та досягти нових клієнтів, що стимулює розвиток малого бізнесу та збільшує конкуренцію на ринку. З іншого боку, споживачі отримують доступ до більшого вибору товарів та послуг, що може призвести до зниження цін та покращення якості обслуговування.

Виходячи з цього, було вирішено створити систему, котра би мала зрозумілий інтерфейс та надавала гнучкий функціонал користувачам.

Актуальність дослідження: актуальність теми зумовлена необхідністю спрощення процесу взаємодії користувача та подібних систем.

З огляду на це, можна виділити наступні цілі і задачі нашого дослідження:

Мета: розробити систему онлайн оголошень.

Задачі:

– сформувати вимоги до системи;
– спроєктувати та побудувати архітектуру системи;
– реалізувати систему онлайн оголошень;
– протестувати роботу системи.

Об’єкт дослідження: процес створення оголошення користувачем, інструменти для реалізації системи онлайн оголошень, функціонал системи,

необхідний користувачам.

Предмет дослідження: створення системи, що дозволяє користувачам розміщувати оголошення.

Методи дослідження: моделювання, проєктування, програмний, аналітичний.

Перший розділ присвячено збору та аналізуванню вимог до системи, огляду інструментів розробки та опису системи.

У другому розділі розглянуто етапи проєктування системи, наведено детальний опис прецедентів.

Третій розділ присвячено реалізації та тестуванню роботи системи, наведено детальне керівництво користувача, яке описує процес роботи з системою онлайн оголошень.

1 ТЕХНІЧНЕ ЗАВДАННЯ

1.1 Терміни та визначення

1.1.1 Загальні терміни

Система – система онлайн оголошень.

Angular – це фреймворк для створення вебдодатків, який базується на TypeScript. Angular дозволяє створювати потужні та масштабовані додатки з використанням компонентної архітектури та забезпечує розробникам ряд інструментів для зручної роботи.

Firebase — це платформа, розроблена компанією Google, яка надає набір інструментів і послуг для розробки веб та мобільних додатків.

ДВІ – Діаграма Варіантів Використання чи Use Case Diagram.

ДД – Діаграма Діяльності.

ДП – Діаграма Послідовності.

Користувач – людина, яка зареєстрована у системі та може взаємодіяти з функціоналом.

Адміністратор – користувач, який має повні права на взаємодію з системою.

Гість – людина, яка не зареєстрована у системі та має обмежені права на взаємодію з нею.

1.1.2 Технічні терміни

Realtime Database – база даних, яка дозволяє синхронізувати дані в реальному часі.

1.2 Функціональні вимоги

1.2.1 Призначення і цілі створення системи

Функціональне призначення системи – реалізувати можливість розміщувати електронні оголошення.

Експлуатаційне призначення системи: система може експлуатуватися користувачами та гостями системи.

Мета створення системи – розробка системи онлайн оголошень.

1.2.2 Загальні функціональні можливості системи

Система має надавати користувачам такі можливості:

– створення/редагування/перегляд оголошення;
– перегляд списку оголошень;
– пошук оголошення;
– вхід/вихід до/з системи.

Система має надавати адміністраторам такі можливості:

– створення/редагування/перегляд/видалення користувача;
– видалення оголошення.

Система має надавати гостям такі можливості:

– перегляд списку оголошень;
– перегляд оголошення;

1.3 Нефункціональні вимоги

Інтерфейс користувача. Система повинна відображати коректно інтерфейс користувача на будь-якому пристрої.

Підтримка браузерів. Система повинна працювати для наступних браузерів останніх версій:

– Mozilla Firefox;
– Google Chrome;
– Safari;
– Microsoft Edge;
– Opera.

Вимоги до продуктивності. Система повинна відображати будь-яку сторінку не довше, ніж за 2 секунди.

Система повинна генерувати кожен сертифікат не довше, ніж 10 секунд.

Вимоги до безпеки. Система не повинна надавати доступ неавторизованим користувачам доступ до функціоналу системи.

1.4 Опис предметної області

Предметною областю є розробка сервісу генерації сертифікатів. Дана система повинна надавати користувачам можливість згенерувати сертифікати, обравши шаблон та дані. Процес взаємодії з системою проходить наступним чином. Користувач, повинен ввести адресу сайту в браузері та ввести дані в форму входу до системи. Якщо користувач вперше на сайті, він повинен спочатку зареєструватися.

Після входу до системи користувач має можливість взаємодіяти з такими розділами як: шаблони, сертифікати. Перед початком генерації сертифікату/ів, користувач повинен створити перший шаблон та завантажити дані.

Процес генерації сертифікату/ів проходить наступним чином. Користувач обирає розділ «Шаблони», система відображає інтерфейс взаємодії з даним розділом. Для створення нового шаблону користувач натискає кнопку «Створити», система відображає форму побудови шаблону. Після створення, користувач натискає на кнопку «Зберегти», система зберігає шаблон. Далі користувач обирає розділ «Сертифікати», система відображає інтерфейс взаємодії з даним розділом. Користувач обирає шаблон, завантажує

дані та натискає на кнопку «Генерація», система завантажує сертифікати на пристрій користувача.

1.5 Опис системи

Сучасний світ, в якому ми живемо, вимагає від нас постійного вдосконалення та розвитку. В освіті та бізнесі сертифікати стали невід'ємною частиною документації та визнання компетенції. Сертифікати доводять нашу кваліфікацію та можуть відкривати двері до нових можливостей.

Ручна генерація сертифікатів може бути часовою та ресурсозатратною задачею. Особливо це стосується організацій, які регулярно видають сертифікати, такі як навчальні заклади або компанії зі стажуванням. Розробка сервісу генерації сертифікатів дозволить автоматизувати цей процес і зекономити час та ресурси.

У зв'язку з цим актуальність розробки сервісу генерації сертифікатів надзвичайно висока.

Можливості системи:

– керування шаблонами;
– використання набору даних користувача;
– генерація сертифікатів.

1.6 Огляд інструментів розробки

1.6.1 Angular

Angular – це потужний та високопродуктивний фреймворк для розробки вебдодатків з використанням HTML, CSS та TypeScript. Angular заснований на принципах компонентної архітектури, де кожен елемент вебдодатку

представлений компонентом, який містить свою логіку та представлення. Компоненти Angular є самодостатніми, що дозволяє їх використовувати в різних частинах додатку, що підвищує його масштабованість та підтримку.

Основними перевагами фреймворку Angular є [1, 9]:

– декларативність: Angular використовує декларативні шаблони для відображення даних, що дозволяє розробникам зосередитися на функціональному програмуванні та забезпечити легку та читабельну кодову базу;
– широкі можливості: Angular надає вбудовані директиви, такі як *ngIf та *ngFor, які дозволяють керувати відображенням даних на сторінці, крім того, Angular надає розробникам можливість використовувати підключені бібліотеки, такі як RxJS, для роботи з асинхронними операціями та роботи з потоками даних;
– ін’єкція залежностей: Angular має потужну систему ін’єкції залежностей, яка дозволяє забезпечити легку тестовість додатку та збільшити його модульність – розробникам не потрібно прямо включати залежності в клас, вони можуть бути включені за допомогою ін’єкції;
– маршрутизація: Angular має потужну систему маршрутизації, яка дозволяє розробникам змінювати сторінки без перезавантаження сторінки – це дозволяє забезпечити користувачам більш зручний та швидкий досвід взаємодії з додатком;
– підтримка мобільних пристроїв: Angular дозволяє розробникам створювати мобільні додатки з використанням іонів (Ionic), який є фреймворком для розробки гібридних мобільних додатків на базі Angular;
– узагальненість: Angular дозволяє розробникам створювати універсальні додатки, які можуть працювати як на клієнтському боці, так і на серверному – це дозволяє розробникам забезпечувати більшу кількість можливостей для оптимізації додатку та його підтримки.

Angular є популярним вибором для розробки вебдодатків, особливо для

більш складних та масштабних проєктів. За допомогою Angular розробники можуть створювати високопродуктивні додатки, які мають легку та читабельну кодову базу, а також підтримку мобільних пристроїв та інші переваги, які забезпечують більш зручний та ефективний досвід взаємодії з додатком [3].

1.6.2 Firebase

Firebase – це облачна платформа, розроблена компанією Google, яка надає набір інструментів і послуг для розробки та хостингу мобільних та вебдодатків.

Firebase надає різні можливості для розробників, включаючи [13]:

– зберігання даних: Firebase пропонує рішення для зберігання реального часу та об’єктного сховища даних, що дозволяє легко зберігати і синхронізувати дані між різними платформами та пристроями;
– аутентифікація користувачів: Firebase дозволяє налаштовувати аутентифікацію користувачів через різні методи, такі як електронна пошта, соціальні мережі та інші;
– реальний час: Firebase Realtime Database надає можливість синхронізації даних в реальному часі між клієнтами, що дозволяє створювати додатки, які миттєво відображають зміни;
– хостинг: Firebase також надає послугу хостингу для вебдодатків, що дозволяє розгортати статичні та динамічні сайти;
– аналітика та звітність: Firebase пропонує інструменти для відстеження користувацької активності в додатку та отримання звітів про його продуктивність;
– інші сервіси: Firebase містить багато інших сервісів, такі як Cloud Functions, Cloud Storage, Firebase Cloud Messaging (FCM), Authentication, і багато інших.

Firebase дозволяє розробникам швидко створювати, розгортати та

масштабувати додатки з використанням інфраструктури хмарних послуг, що робить його популярним рішенням для мобільної та веброзробки.

2 ПРОЄКТУВАННЯ

2.1 Використання UML під час розробки системи

UML (Unified Modeling Language) є стандартною мовою для візуального моделювання програмних систем. Використання UML під час розробки системи допомагає розробникам і командам розробки краще розуміти, проєктувати і документувати програмні системи.

Розглянемо аспекти використання UML під час розробки системи.

Моделювання структури системи: UML дозволяє розробникам моделювати структуру системи за допомогою діаграм класів, діаграм об’єктів та інших структурних діаграм. Це допомагає визначити класи, їх атрибути і взаємозв’язки.

Моделювання поведінки системи: UML також надає засоби для моделювання поведінки системи. Діаграми активностей, діаграми станів і діаграми послідовностей дозволяють описати, як система реагує на події і взаємодіє з іншими системами або користувачами.

Документування системи: UML може служити для створення документації системи. Кожна діаграма UML може бути супроводжена пояснювальними текстами та анотаціями, що полегшує розуміння системи для розробників, тестувальників і інших учасників проєкту.

Спільна мова комунікації: UML створює спільну мову для комунікації між учасниками проєкту. Це допомагає уникнути недорозумінь і сприяє кращому спілкуванню в команді.

Аналіз і проєктування: UML може бути використаний для аналізу потреб системи і проєктування її архітектури. Візуалізація системи допомагає ідентифікувати проблеми та вирішувати їх на ранніх стадіях розробки.

Реалізація і тестування: UML може бути використаний для підтримки реалізації коду та проведення тестування. Наприклад, діаграми

послідовностей можуть бути використані для генерації коду або тестових сценаріїв.

Моделювання архітектури: UML може бути використаний для моделювання загальної архітектури системи, включаючи рівні компонентів, пакети і взаємозв'язки між ними.

Використання UML допомагає створювати більш ясну та структуровану систему, полегшує спілкування в команді та допомагає вдосконалити процес розробки програмного забезпечення. У той же час, важливо зауважити, що використання UML повинно бути адаптовано до конкретних потреб та розміру проєкту, оскільки надмірна деталізація може призвести до зайвої складності.

2.2 Діаграма варіантів використання

Діаграма варіантів використання (Use Case Diagram) – це один із видів діаграм UML, який використовується для моделювання функціональності системи та взаємодій між системою та її користувачами або іншими зовнішніми сутностями. Діаграма варіантів використання допомагає визначити, як система реагує на зовнішні події або взаємодіє з користувачами та іншими системами.

Розглянемо основні елементами діаграми варіантів використання.

Актори: актори представляють ролі, які взаємодіють з системою. Це можуть бути користувачі, інші системи, зовнішні програми або будь-які інші сутності, які мають певні вимоги до системи.

Варіанти використання (Use Cases): варіанти використання описують конкретні функціональність системи або дії, які система може виконувати для задоволення потреб акторів. Кожен варіант використання представляє собою одну конкретну функцію або можливість.

Зв'язки між акторами та варіантами використання: зв'язки показують, які актори взаємодіють з якими варіантами використання.

Наприклад, актор може викликати певний варіант використання.

Система: зазвичай на діаграмі є один об’єкт «Система», який представляє саму систему, що розробляється.

Діаграма варіантів використання використовується для:

– опису функціональних вимог до системи;

– визначення взаємодій користувачів із системою;

– уточнення, як система реагує на певні події або сценарії;

– встановлення основних варіантів використання системи.

Варіанти використання можуть бути узагальнені до більш детальних сценаріїв, наприклад, за допомогою діаграм секвенцій або діаграм діяльності.

Ця діаграма допомагає команді розробників і стейкхолдерам краще зрозуміти, як система взаємодіє зі своїми користувачами та іншими системами, і є корисним інструментом для аналізу та специфікації вимог до програмного забезпечення.

На рисунку 2.1 представлена діаграма варіантів використання.

На діаграмі представлено акторів «Користувач», «Клієнт» та «Адміністратор», кожен з яких може взаємодіяти з системою згідно на різних рівнях.

Виділено 1 основний варіант використання – «Створення оголошення». Після того, як користувач авторизується, система відображає інтерфейс взаємодії з оголошеннями.

Для створення нового оголошення користувач натискає кнопку «Створити», система відображає форму створення нового оголошення. Після заповнення необхідних полів, користувач натискає на кнопку «Зберегти», система зберігає оголошення. На кожну з цих дій система посилає запит до Firebase і повідомляє користувача про результат дії.

Варіанти використання визначають функціональні можливості. Кожен з них представляє певний спосіб використання. Таким чином, кожен варіант використання відповідає послідовності дій для того, щоб користувач міг отримати певний результат (див. рис. 2.2, 2.3).

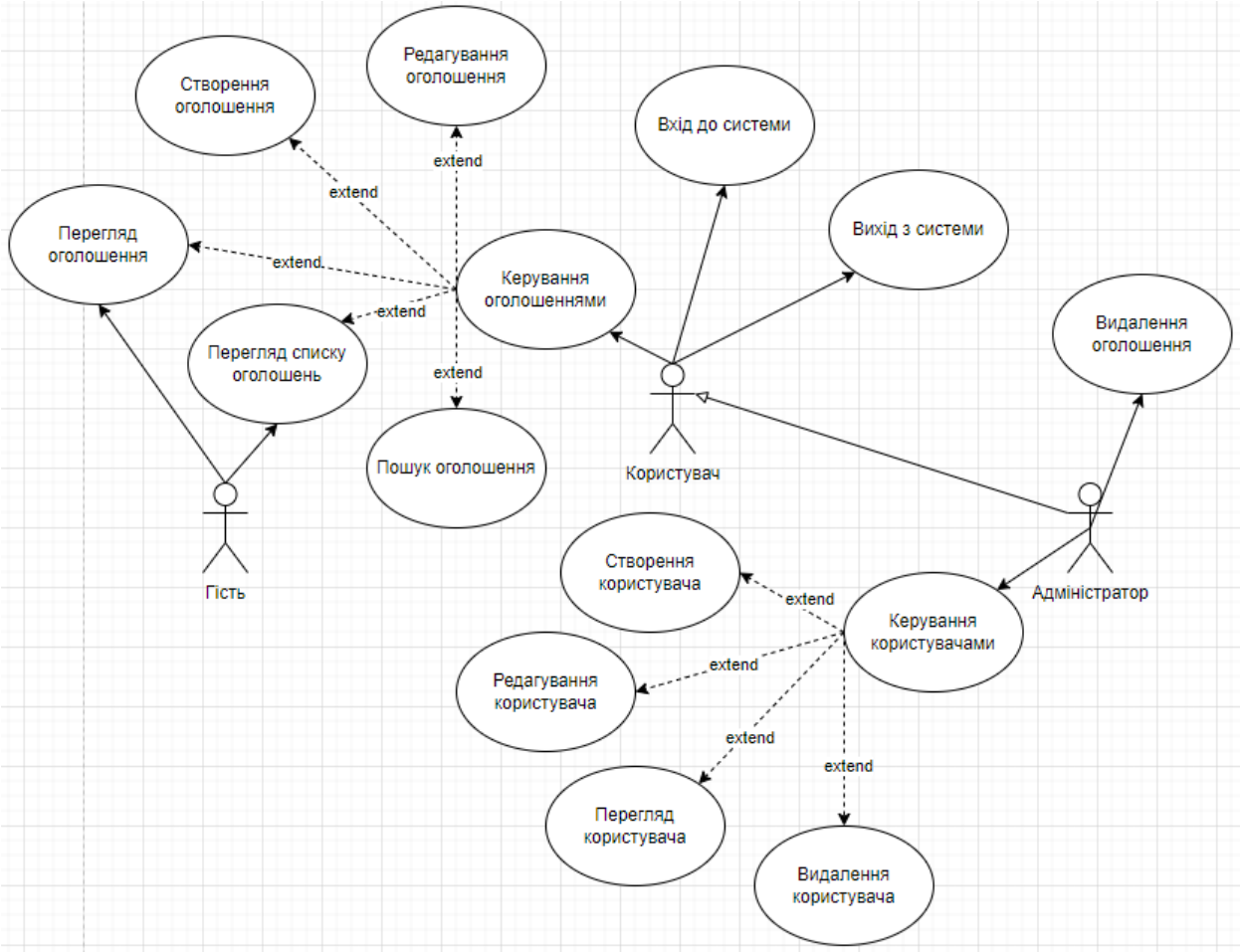

Рисунок 2.1 – Діаграма варіантів використання

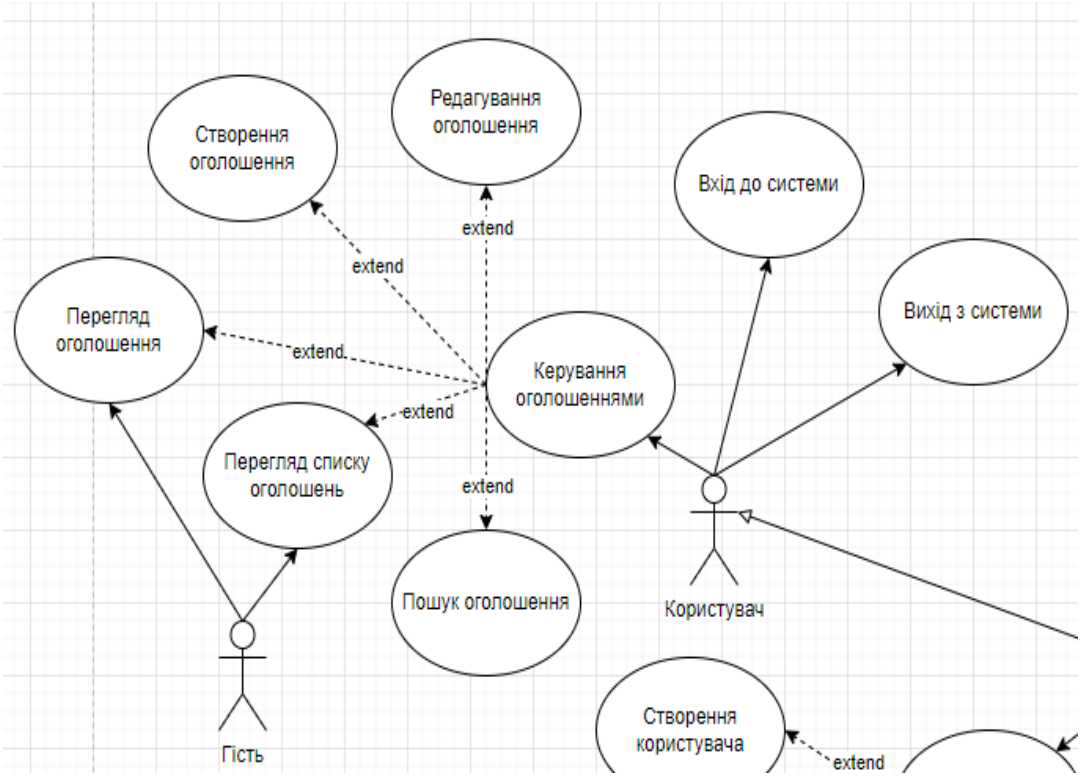

Рисунок 2.2 – Діаграма варіантів використання «Користувач» і «Клієнт»

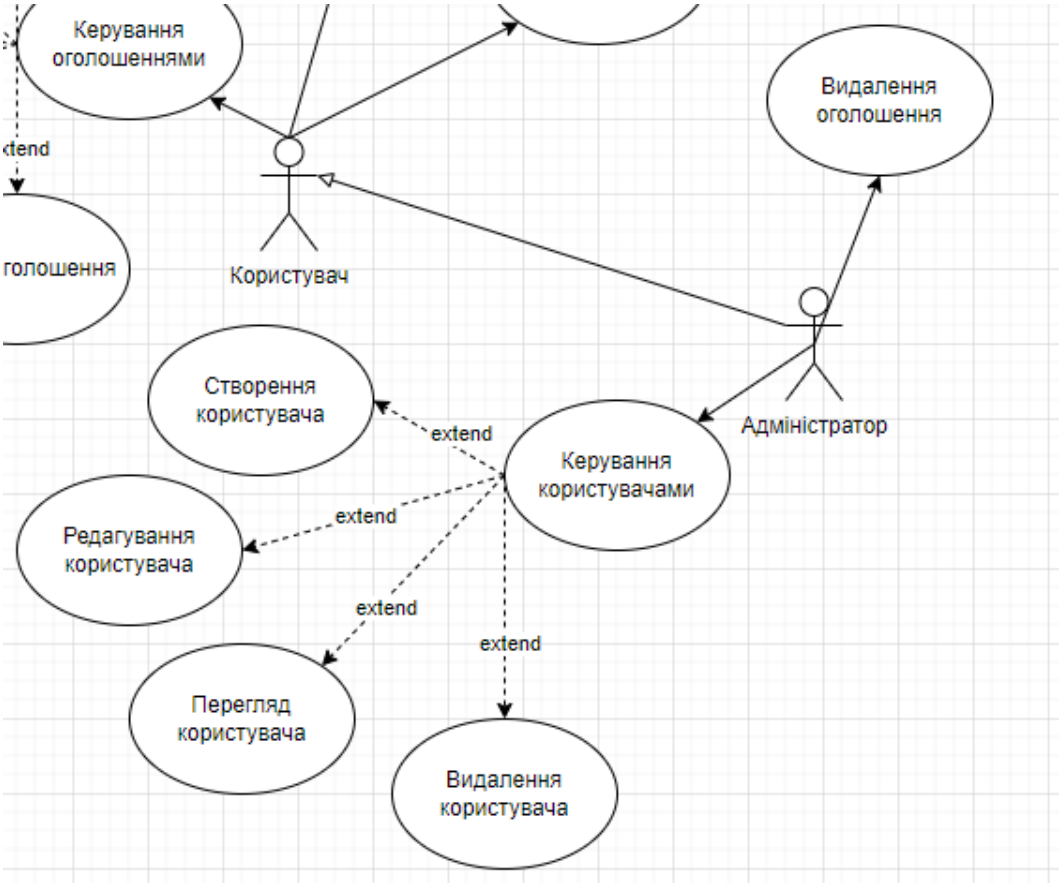

Рисунок 2.3 – Діаграма варіантів використання «Адміністратор»

2.2.1 Опис варіантів використання

Прецедент «Вхід до системи»

Призначення: даний варіант використання надає можливість зареєстрованому користувачу увійти у систему для використання.

Основний потік подій: даний варіант використання починає виконуватися, коли зареєстрованому користувачу потрібно авторизуватися. Система пропонує ввести логін та пароль. Після того, як користувач ввів їх, система перевіряє правильність введених даних, і у випадку вдачі надає йому функціонал.

Альтернативний потік: якщо логін чи пароль невірні – система сповіщає про це користувача. Користувач може спробувати ще раз пройти авторизацію.

Передумова: перед початком виконання даного варіанта використання користувач повинен бути зареєстрований у системі.

Прецедент «Вихід з системи»

Призначення: даний варіант використання надає можливість користувачу вийти з системи.

Основний потік подій: даний варіант використання починає виконуватися, коли користувачу потрібно вийти з системи. Користувач натискає на значок профіля та у списку дій обирає «Вихід». Після того, як користувач вийшов, система відображає головну сторінку.

Передумова: перед початком виконання даного варіанта використання користувач повинен авторизуватися в системі.

Прецедент «Керування оголошеннями»

Призначення: даний варіант використання надає можливість користувачу взаємодіяти з оголошеннями відповідно до наданих прав.

Основний потік подій: даний варіант використання починає виконуватися, коли користувач системи входить до особистого кабінету. Система відображає список оголошень та інструменти взаємодії з ними.

Альтернативний потік подій: якщо жодного оголошення ще не було додано, система повідомляє про це у вигляді напису «Немає жодного оголошення.

Передумова: перед початком виконання даного варіанта використання користувач повинен виконати вхід до системи.

Прецедент «Створення оголошення»

Призначення: даний варіант використання надає можливість користувачу створювати нові оголошення.

Основний потік подій: даний варіант використання починає виконуватися, коли користувач системи натискає кнопку «Створити», система відображає форму створення нового оголошення. Після заповнення необхідних полів, користувач натискає кнопку «Зберегти», система зберігає нове оголошення та відображає головну сторінку особистого кабінету.

Передумова: перед перед початком виконання даного варіанта використання користувач повинен виконати вхід до системи.

Виняткова ситуація 1: користувач натиснув кнопку «Скасувати» – система відображає головну сторінку особистого кабінету.

Виняткова ситуація 2: користувач ввів невалідні дані – система сповіщає про це, і користувач має можливість ввести дані знов.

Виняткова ситуація 3: сервер повернув помилку – система сповіщає про це, і дані не зберігаються.

Прецедент «Видалення оголошення»

Призначення: даний варіант використання надає можливість користувачу видалити оголошення відповідно до наданих прав.

Основний потік подій: даний варіант використання починає виконуватися, коли користувач системи натискає на кнопку «Видалити» біля відповідного оголошення, система видаляє його.

Передумова: перед перед початком виконання даного варіанта використання користувач повинен виконати вхід до системи та створити принаймні одне оголошення.

Виняткова ситуація 1: сервер повернув помилку – система сповіщає про це, і оголошення не видаляється.

Прецедент «Редагування оголошення»

Призначення: даний варіант використання надає можливість користувачу редагувати оголошення відповідно до наданих прав.

Основний потік подій: даний варіант використання починає виконуватися, коли користувач системи натискає на кнопку «Редагувати» біля відповідного оголошення, система відображає інтерфейс редагування оголошення. Після внесення змін, користувач натискає кнопку «Зберегти», система зберігає оновлені дані та відображає головну сторінку особистого кабінету.

Передумова: перед перед початком виконання даного варіанта використання користувач повинен виконати вхід до системи та створити

принаймні одне оголошення.

Виняткова ситуація 1: користувач натиснув кнопку «Скасувати» – система відображає головну сторінку особистого кабінету.

Виняткова ситуація 2: користувач ввів невалідні дані – система сповіщає про це, і користувач має можливість ввести дані знов.

Виняткова ситуація 3: сервер повернув помилку – система сповіщає про це, і дані не зберігаються.

Прецедент «Перегляд оголошення»

Призначення: даний варіант використання надає можливість користувачу переглядати оголошення відповідно до наданих прав.

Основний потік подій: даний варіант використання починає виконуватися, коли користувач системи натискає на кнопку «Переглянути» біля відповідного оголошення, система відображає оголошення.

Прецедент «Перегляд списку оголошень»

Призначення: даний варіант використання надає можливість користувачу переглядати список оголошень.

Основний потік подій: даний варіант використання починає виконуватися, коли користувач входить до системи або переходить на головну сторінку.

Виняткова ситуація 1: жодного оголошення не додано – система відображає напис «Оголошення відсутні».

Прецедент «Пошук оголошення»

Призначення: даний варіант використання надає можливість користувачу шукати оголошення.

Основний потік подій: даний варіант використання починає виконуватися, коли користувач системи вводить дані в поле «Пошук», система відображає відповідні оголошення .

Виняткова ситуація 1: жодного оголошення не додано – система відображає напис «Оголошення відсутні».

Виняткова ситуація 2: жодного оголошення не було знайдено – система

відображає напис «Не знайдено оголошень за такими параметрами пошуку».

Прецедент «Керування користувачами»

Призначення: даний варіант використання надає можливість адміністратору взаємодіяти з користувачами.

Основний потік подій: даний варіант використання починає виконуватися, коли адміністратор системи натискає на пункт меню «Користувачі» в особистому кабінеті. Система відображає перелік користувачів та інструментарій взаємодії з ними.

Передумова: перед початком виконання даного варіанта використання користувач повинен виконати вхід до системи з правами «Адміністратор».

Прецедент «Створення користувача»

Призначення: даний варіант використання надає можливість адміністратору створювати нових користувачів.

Основний потік подій: даний варіант використання починає виконуватися, коли адміністратор системи натискає кнопку «Створити», система відображає форму створення нового користувача. Після заповнення необхідних полів, адміністратор натискає кнопку «Зберегти», система зберігає нового користувача та відображає сторінку керування користувачами.

Передумова: перед початком виконання даного варіанта використання користувач повинен виконати вхід до системи з правами «Адміністратор».

Виняткова ситуація 1: адміністратор натиснув кнопку «Скасувати» – система відображає сторінку керування користувачами.

Виняткова ситуація 2: адміністратор ввів невалідні дані – система сповіщає про це, і адміністратор має можливість ввести дані знов.

Виняткова ситуація 3: сервер повернув помилку – система сповіщає про це, і дані не зберігаються.

Прецедент «Видалення користувача»

Призначення: даний варіант використання надає можливість адміністратору видаляти користувачів.

Основний потік подій: даний варіант використання починає

виконуватися, коли адміністратор системи натискає на кнопку «Видалити» біля відповідного користувача, система видаляє його.

Передумова: перед початком виконання даного варіанта використання користувач повинен виконати вхід до системи з правами «Адміністратор» та створити принаймні одного користувача.

Виняткова ситуація 1: сервер повернув помилку – система сповіщає про це, і користувач не видаляється.

Прецедент «Редагування користувача»

Призначення: даний варіант використання надає можливість адміністратору редагувати дані користувачів.

Основний потік подій: даний варіант використання починає виконуватися, коли адміністратор системи натискає на кнопку «Редагувати» біля відповідного користувача, система відображає інтерфейс редагування користувача. Після внесення змін, адміністратор натискає кнопку «Зберегти», система зберігає оновлені дані та відображає сторінку керування користувачами.

Передумова: перед початком виконання даного варіанта використання користувач повинен виконати вхід до системи з правами «Адміністратор» та створити принаймні одного користувача.

Виняткова ситуація 1: адміністратор натиснув кнопку «Скасувати» – система відображає сторінку керування користувачами.

Виняткова ситуація 2: адміністратор ввів невалідні дані – система сповіщає про це, і адміністратор має можливість ввести дані знов.

Виняткова ситуація 3: сервер повернув помилку – система сповіщає про це, і дані не зберігаються.

Прецедент «Перегляд користувача»

Призначення: даний варіант використання надає можливість адміністратору переглядати дані користувача.

Основний потік подій: даний варіант використання починає виконуватися, коли адміністратор системи натискає на кнопку «Переглянути»

біля відповідного користувача, система відображає інформацію.

Передумова: перед початком виконання даного варіанта використання користувач повинен виконати вхід до системи з правами «Адміністратор» та створити принаймні одного користувача.

2.3 Діаграма діяльності

Діаграма діяльності (Activity Diagram) – це один з видів діаграм UML (Unified Modeling Language), який використовується для моделювання послідовності дій або процесів в системі або програмі. Діаграми діяльності візуально представляють кроки або дії, які виконуються в системі та відображають послідовність цих дій, умови і варіанти переходів між ними.

Розглянемо основні елементами діаграми діяльності.

Дії (Actions): дії представляють конкретні кроки або дії, які виконуються в процесі. Це може бути все, від простих операцій до складних обчислень або виконання інших послідовних дій.

Рішення (Decisions): рішення або відгалуження вказують на різні шляхи або альтернативи в процесі, що залежать від певних умов. Вони використовуються для визначення умовних переходів в діаграмі діяльності.

Контрольні вузли (Control Nodes): вузли, такі як «початок» (Initial Node) і «кінець» (Final Node), вказують на початок і завершення діаграми діяльності. Вони визначають початок та кінець процесу або підпроцесу.

Потоки (Flow): взаємодія між діями визначається за допомогою стрілок або ліній потоків, які показують послідовність виконання дій. Стрілки можуть також вказувати на умовні або безумовні переходи.

Діаграми діяльності використовуються для:

– моделювання процесів бізнесу і послідовностей дій в системі;
– визначення і аналізу алгоритмів та процесів;
– визначення паралельних виконань та синхронізації дій;

– уточнення і візуалізації послідовностей дій в рамках конкретних сценаріїв.

Діаграми діяльності допомагають команді розробників та стейкхолдерам краще розуміти логіку системи або процесу і визначати оптимальні шляхи виконання дій для досягнення певної мети.

Наведемо діаграму діяльності, що описує модель поведінки варіанта використання «Створити оголошення». Діаграма представлена на рисунках 2.4 – 2.5.

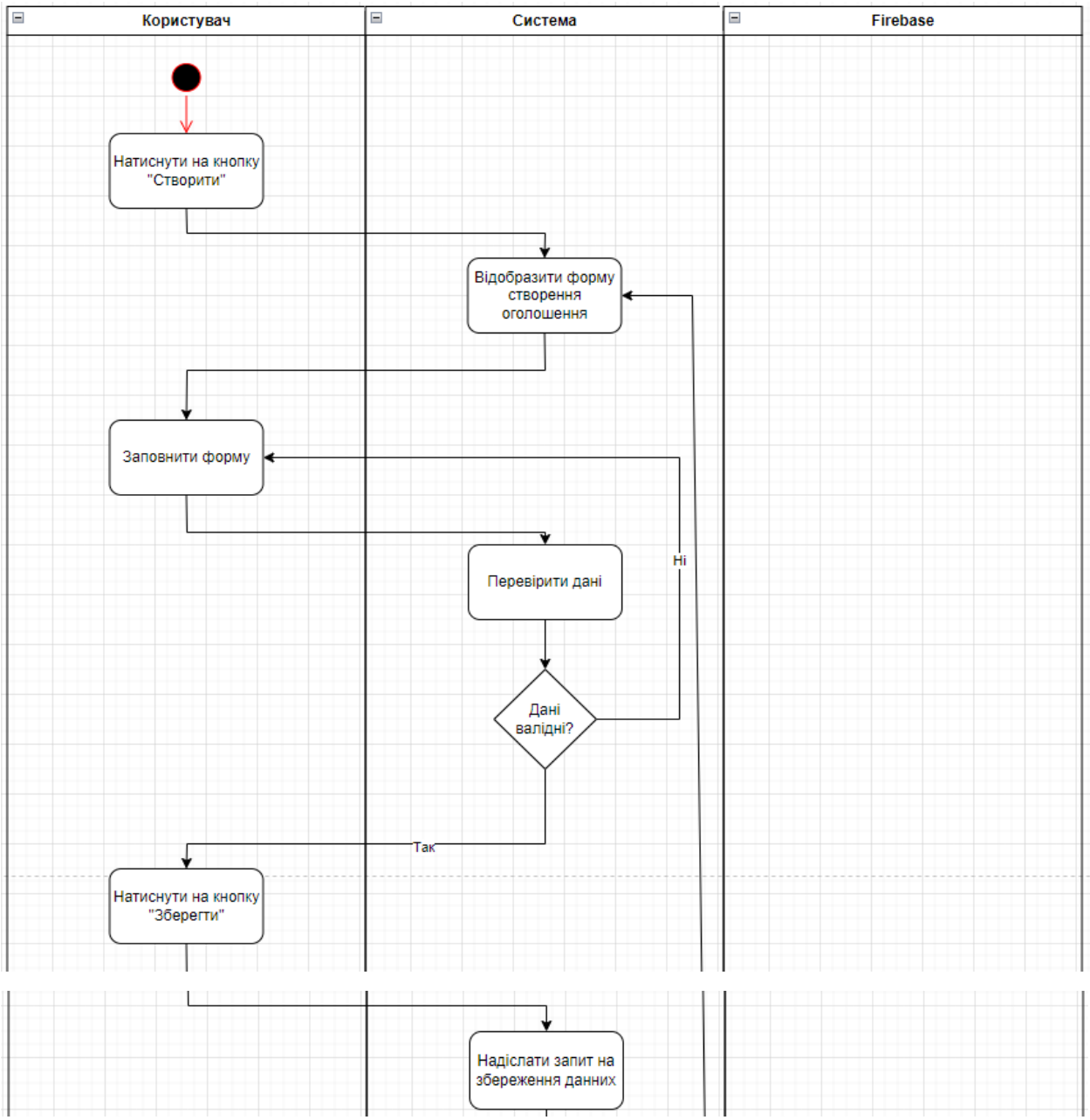

Рисунок 2.4 – Діаграма діяльності

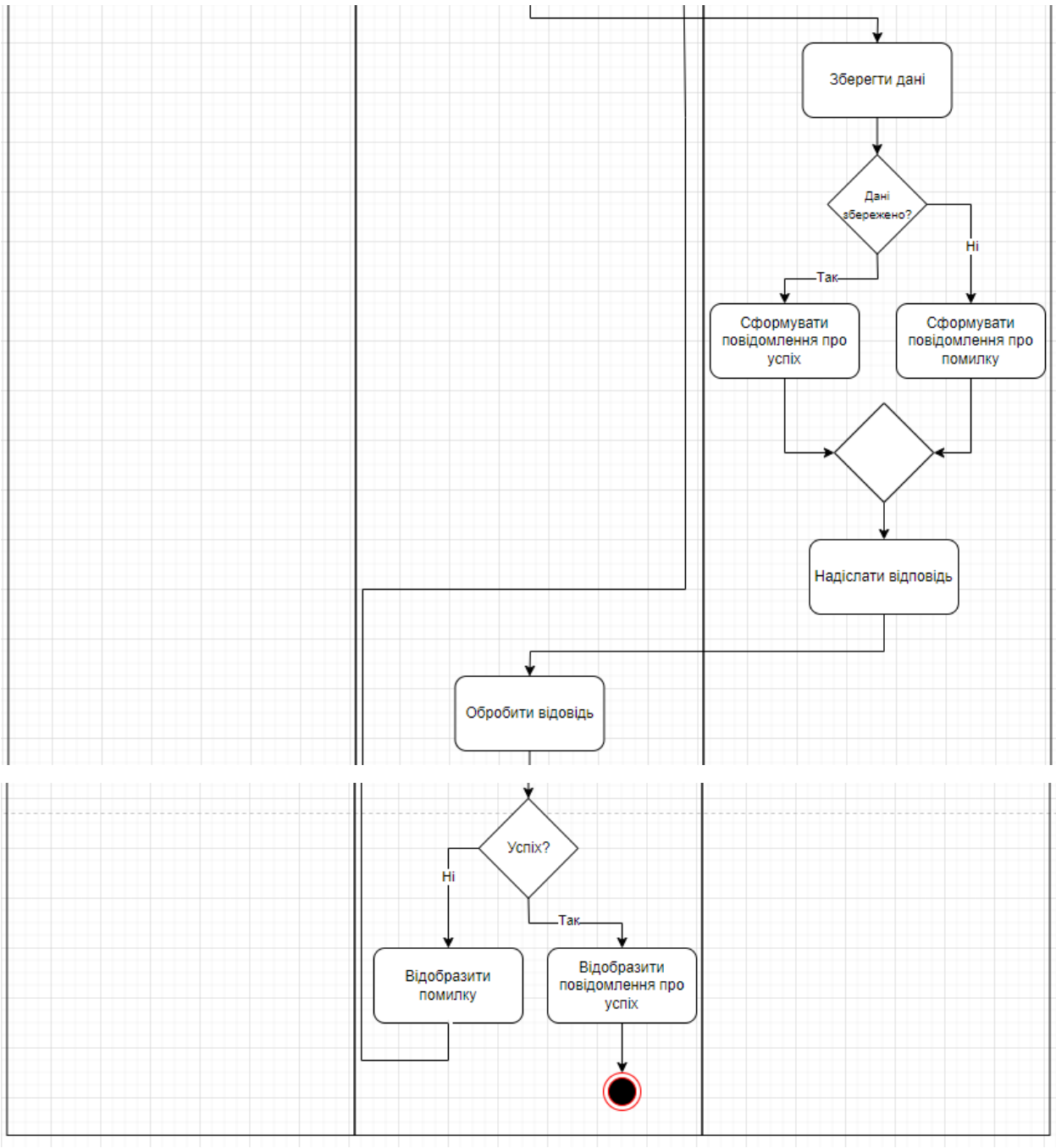

Рисунок 2.5 – Діаграма діяльності

2.4 Діаграма послідовності

Діаграма послідовності (Sequence Diagram) – це один із видів діаграм UML (Unified Modeling Language), який використовується для моделювання взаємодій між різними об’єктами або компонентами в системі в часовій послідовності. Діаграма послідовності допомагає візуалізувати, як об’єкти

взаємодіють один з одним у певному сценарії або взаємодій.

Основними елементами діаграми послідовності представлені нижче.

Об'єкти (Objects): об'єкти або екземпляри класів, які беруть участь у взаємодії, представляються на діаграмі. Кожен об'єкт має ім'я та опціонально – тип (клас або компонент, до якого він належить).

Лінії життя (Lifelines): лінії життя відображають «життя» об'єкта на діаграмі та вказують на той час, коли об'єкт існує та бере участь у взаємодії. Лінії життя зв'язані з об'єктами та показують, коли об'єкт активний або пасивний.

Послідовності повідомлень (Message Sequences): послідовності повідомлень показують взаємодію між об'єктами. Це може бути виклик методу, повідомлення або сповіщення від одного об'єкта іншому. Вони представляються стрілками, які зєднують об'єкти та мають імена та вміст, що описують, що сталося в процесі взаємодії.

Опціональні фрагменти (Optional Fragments): діаграми послідовності можуть також включати опціональні фрагменти, такі як умовні або циклічні фрагменти, які показують альтернативи або повторювані дії в процесі взаємодії.

Діаграми послідовності використовуються для:

– моделювання взаємодій між об'єктами в системі;
– визначення послідовності дій в рамках конкретного сценарію;
– відображення викликів методів, обміну повідомленнями та інших видів взаємодій між об'єктами;
– аналіз та специфікація функціональності системи.

Діаграми послідовності є потужним інструментом для розуміння та моделювання взаємодій в системі та можуть бути використані для аналізу, проєктування та документування програмних систем.

На рисунку 2.6 описана діаграма послідовності прецедента «Створити сертифікат».

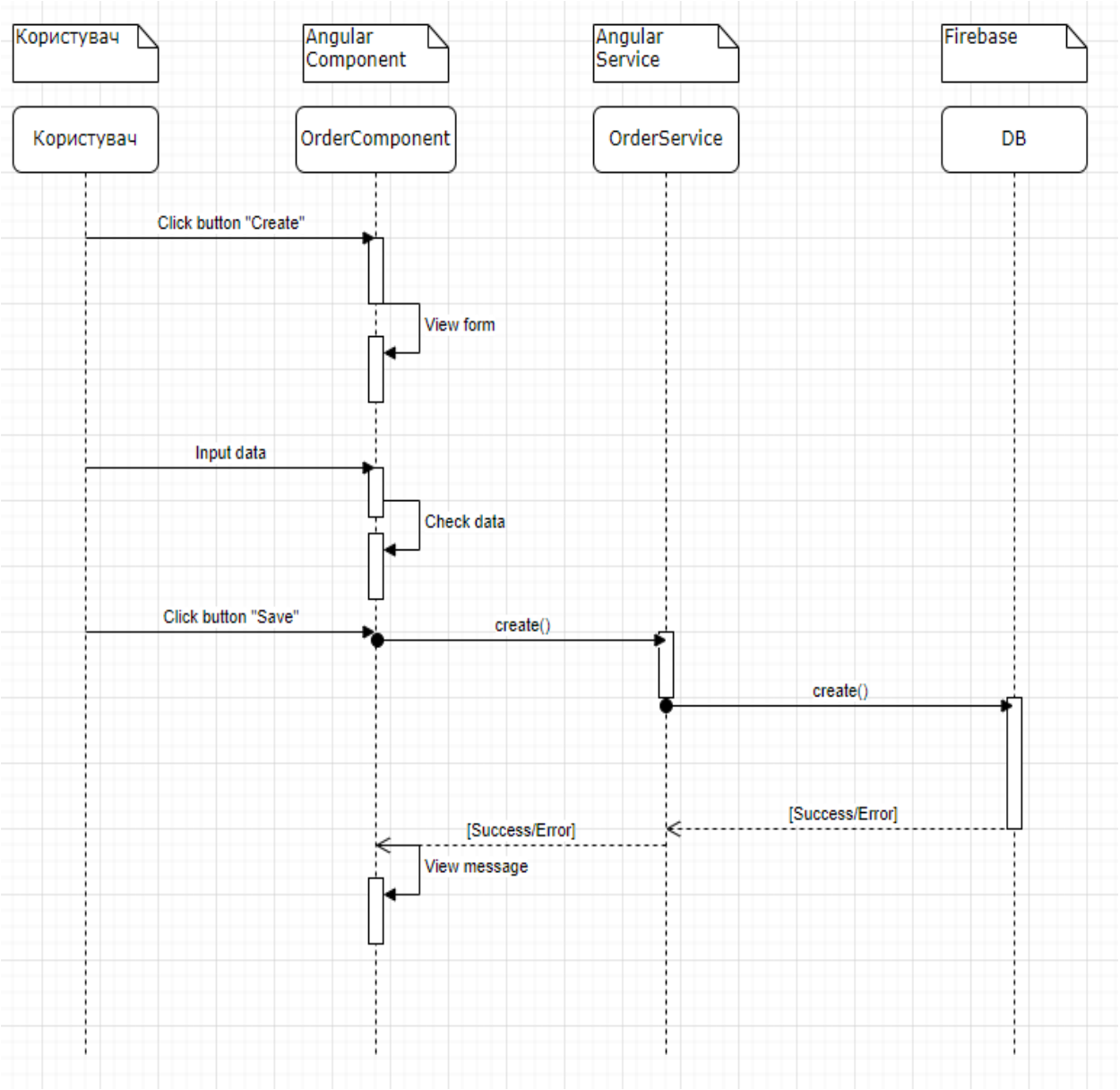

Рисунок 2.6 – Діаграма послідовності

2.5 Діаграма розгортання

Діаграма розгортання (Deployment Diagram) – це один з видів діаграм UML (Unified Modeling Language), який використовується для моделювання фізичної архітектури системи та розташування компонентів програмного забезпечення на апаратному обладнанні або інфраструктурі.

Діаграми розгортання використовуються для:

– моделювання фізичної інфраструктури системи та взаємодії

компонентів з апаратним обладнанням та іншими середовищами;

– специфікації розташування та конфігурації компонентів системи;
– аналізу і документування архітектури системи;
– визначення вимог до інфраструктури та розгортання системи на реальних вузлах.

Ці діаграми допомагають командам розробників та системним архітекторам краще розуміти, як компоненти системи взаємодіють з фізичною архітектурою та інфраструктурою, що допомагає в ефективному проєктуванні та розгортанні програмних систем.

На рисунку 2.7 наведено діаграму розгортання системи, що розробляється.

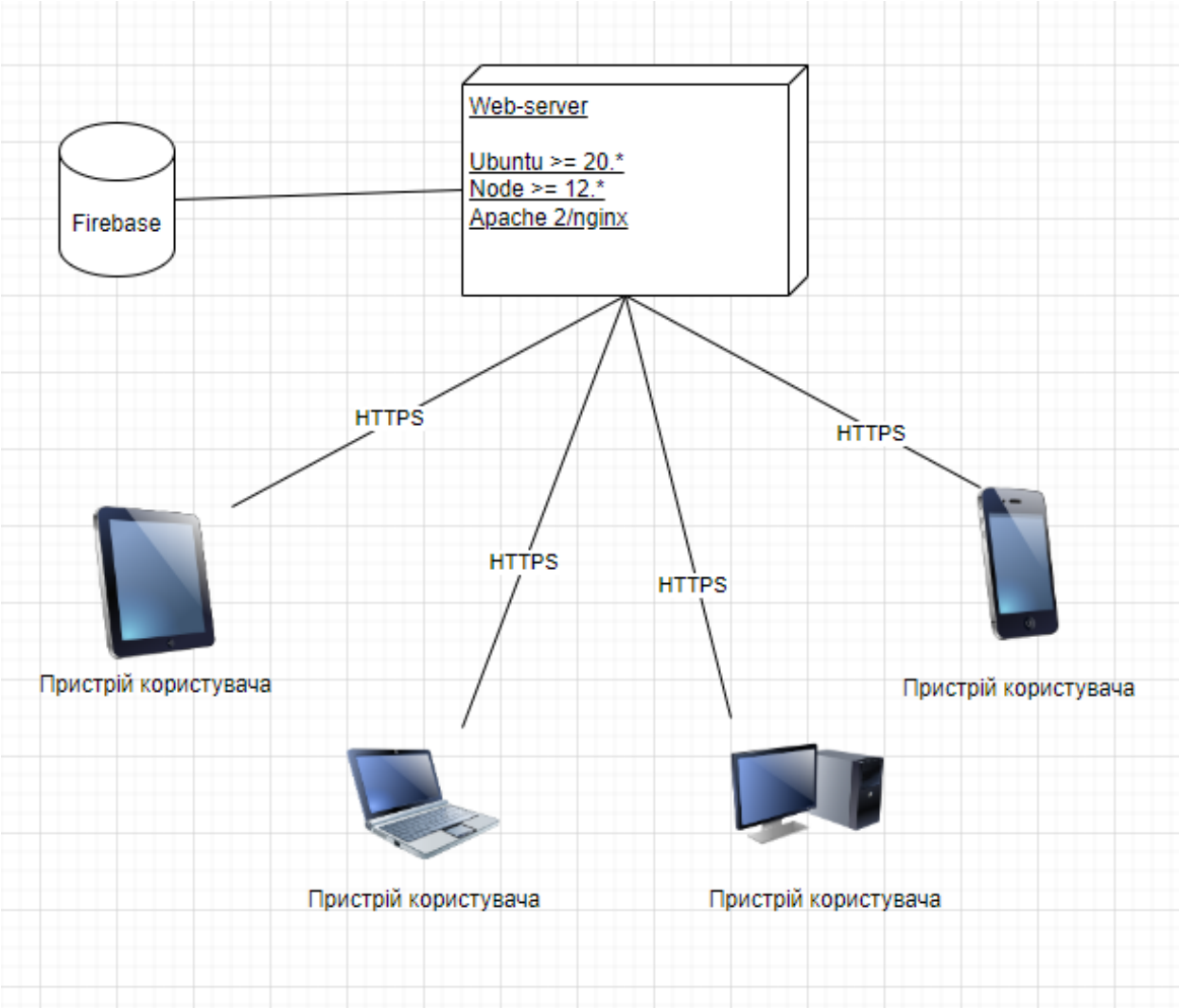

Рисунок 2.7 – Діаграма розгортання

Основні елементи діаграми розгортання розглянемо нижче.

Вузли (Nodes): вузли представляють фізичні або віртуальні пристрої або середовища, на яких будуть розгортані компоненти системи. Вони можуть включати сервери, робочі станції, хмарні середовища, мережеве обладнання тощо.

Артефакти (Artifacts): артефакти представляють програмні компоненти, такі як файли, бінарні об’єкти, бібліотеки або модулі, які розгортатимуться на вузлах. Ці компоненти можуть бути фізично розташовані на вузлах або надавати послуги через мережу.

Зв’язки (Connections): зв’язки вказують на способи комунікації між вузлами і компонентами. Вони показують, як дані або повідомлення пересилаються між різними частинами системи.

Артефакти розгортання (Deployment Artifacts): артефакти розгортання вказують, які саме компоненти (артефакти) будуть розташовані на конкретних вузлах. Це може включати виконувані файли, конфігураційні файли, бази даних і т.д.

3 РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ

3.1 Опис інструментів розробки

Для реалізації було використано front-end фреймворк Angular та Firebase Storage.

RxJS – це бібліотека, що реалізує принципи реактивного програмування JavaScript. Заснована на об’єктах типу Observable, вона спрощує написання та контроль асинхронного та подвійного коду [5].

Firebase Storage – це облачна платформа, розроблена компанією Google, яка надає набір інструментів і послуг для розробки та хостингу мобільних та вебдодатків [13].

3.2 Angular-компонент

Компоненти в Angular використовуються для організації ієрархії вебсторінки або додатка. Кожен компонент має свій власний набір логіки, шаблону та стилів, що дозволяє створювати частини коду, які легко масштабуються та вбудовуються в інші компоненти [4, 8].

Приклад коду компонента наведено у Додатку А.

3.3 Angular-сервіс

Клас, який надає певний функціонал та може бути використаний для обробки бізнес-логіки, обміну даними між компонентами та інших сервісів, а також для реалізації загальних функцій, доступних у всьому додатку [6, 7].

Приклад коду сервісу наведено у Додатку Б.

3.4 Firebase Realtime Database

База даних в реальному часі, що надається Google Firebase, облачною платформою для розробки мобільних та вебдодатків. Firebase Realtime Database надає зручний спосіб зберігання та синхронізації даних між різними клієнтами в реальному часі.

Використання Firebase Realtime Database забезпечує простий спосіб організації та синхронізації даних в реальному часі для веб та мобільних додатків без необхідності налаштування та управління власним сервером бази даних [12].

Приклад бази наведено у Додатку В.

3.5 Основні класи системи

Так як основним прецедентом системи є «Створити оголошення», то основними класами системи будуть ті, що надають можливість створювати, змінювати, видаляти та переглядати дані.

Головними класами основного бізнес-процесу є:

– PostService – відповідає за дії пов’язані з Firebase API, а саме містить запити на створення, видалення, зміну та відображення даних про оголошення;
– PostPageComponent – обробляє та відображає дані, отримані від методів PostService. Містить методи для управління відображенням прив’язаного до компоненту шаблона. Отримує дані від шаблону та передає їх у сервіс (наприклад видалення).

Детально ознайомитися з кодом проєкту можна у додатку Г.

3.6 Тестування проєкту

Unit-тест. Вид тестування програмного забезпечення, спрямований на перевірку коректності роботи окремого «юнітарного» компонента програми.

Unit-тестування сприяє виявленню помилок та покращує структуру коду, допомагаючи забезпечити його надійність та легкість розуміння. Також воно сприяє підтримці коду під час його подальшого розвитку [2, 11].

Приклади такого тестування наведено на рисунку 3.1–3.2.

```
describe('PostPageComponent', () => {
    let component: PostPageComponent;
    let fixture: ComponentFixture<PostPageComponent>;
    let mockActivatedRoute;
    let mockPostsService;

    beforeEach(() => {
        mockActivatedRoute = {
            params: of({ id: '1' }), // Simulate route parameters
        };

        mockPostsService = jasmine.createSpyObj(['getById']);
        mockPostsService.getById.and.returnValue(of({
            id: '1',
            title: 'Sample Post',
            text: 'This is a sample post content.',
            author: '',
            date: new Date('01-01-2023')
        } as Post));

        TestBed.configureTestingModule({
            declarations: [PostPageComponent],
            providers: [
                { provide: ActivatedRoute, useValue: mockActivatedRoute },
```

Рисунок 3.1 – Тестування PostPageComponent

```
describe('LoginPageComponent', () => {
    let component: LoginPageComponent;
    let fixture: ComponentFixture<LoginPageComponent>;
    let mockAuthService;
    let mockRouter;
    let mockActivatedRoute;

    beforeEach(() => {
        mockAuthService = jasmine.createSpyObj(['login']);
        mockRouter = jasmine.createSpyObj(['navigate']);
        mockActivatedRoute = { queryParams: of({}) };

        TestBed.configureTestingModule({
            declarations: [LoginPageComponent],
            imports: [ReactiveFormsModule],
            providers: [
                { provide: AuthService, useValue: mockAuthService },
                { provide: Router, useValue: mockRouter },
                { provide: ActivatedRoute, useValue: mockActivatedRoute },
            ],
        });

        fixture = TestBed.createComponent(LoginPageComponent);
        component = fixture.componentInstance;
    });

    it('should create', () => {
        expect(component).toBeTruthy();
    });
```

Рисунок 3.2 – Тестування LoginPageComponent

Integration-тест. Вид тестування програмного забезпечення, спрямований на визначення взаємодії між різними частинами системи для перевірки, чи працюють ці частини коректно як єдина сукупність. У контексті веброзробки, також і в інших областях, це може включати в себе перевірку взаємодії між компонентами, модулями, сервісами, базою даних і т. д.

Integration-тестування допомагає виявити проблеми, які можуть виникнути при взаємодії різних компонентів програми, і забезпечує впевненість у тому, що програма працює як цілісна система [2, 10].

На рисунку 3.3 зображено тестування на створення сервісу.

```
afterEach(() => {
    httpMock.verify();
});

it('should create a post and retrieve it from the server', (done: DoneFn) => {
    const post: Post = {
        id: null,
        title: 'Test Post',
        text: 'This is a test post.',
        author: 'Test',
        date: new Date()
    };

    service.create(post).subscribe(createdPost => {
        expect(createdPost.title).toEqual(post.title);
        expect(createdPost.text).toEqual(post.text);

        service.getById(createdPost.id).subscribe(retrievedPost => {
            expect(retrievedPost).toEqual(createdPost);
            done();
        });
    });

    const createReq = httpMock.expectOne(`${environment.fbDbUrl}/posts.json`);
    expect(createReq.request.method).toBe('POST');
    createReq.flush({ name: 'generated-post-id' } as FbCreateResponse);
});
});
```

Рисунок 3.3 – Тестування сервісу

3.7 Керівництво користувача

3.7.1 Рівень підготовки користувача

Користувач сайту повинен володіти певною кваліфікацією. Навички користувача для роботи з ПК, та навички роботи з web-браузером. Знайомство з Керівництвом користувача.

3.7.2 Підготовка до роботи

Запуск системи. Доступ до сайту здійснюється через мережу Інтернет за допомогою звичайного web-браузера. Адреса сайту в мережі Інтернет: https://board.loc. Для коректної роботи клієнтської частини повинен використовуватися браузер Google Chrome, Mozilla Firefox, Opera, Safari. При вході на Сайт користувач потрапляє на сторінку входу до системи.

На Сайті розрізняються наступні групи користувачів: гість (не увійшли або не зареєстровані, мають доступ до сторінки входу та перегляду оголошень); користувач – приватна особа, авторизований на сайті (далі Користувач), що має доступ до функцій користувача у особистому кабінеті; адміністратор системи (далі Адміністратор) відповідає за підтримку та конфігурування системи.

3.7.3 Вхід до системи та головна сторінка

При вході на сайт, система відображає головну сторінку зі списком всіх створених оголошень (рис. 3.4).

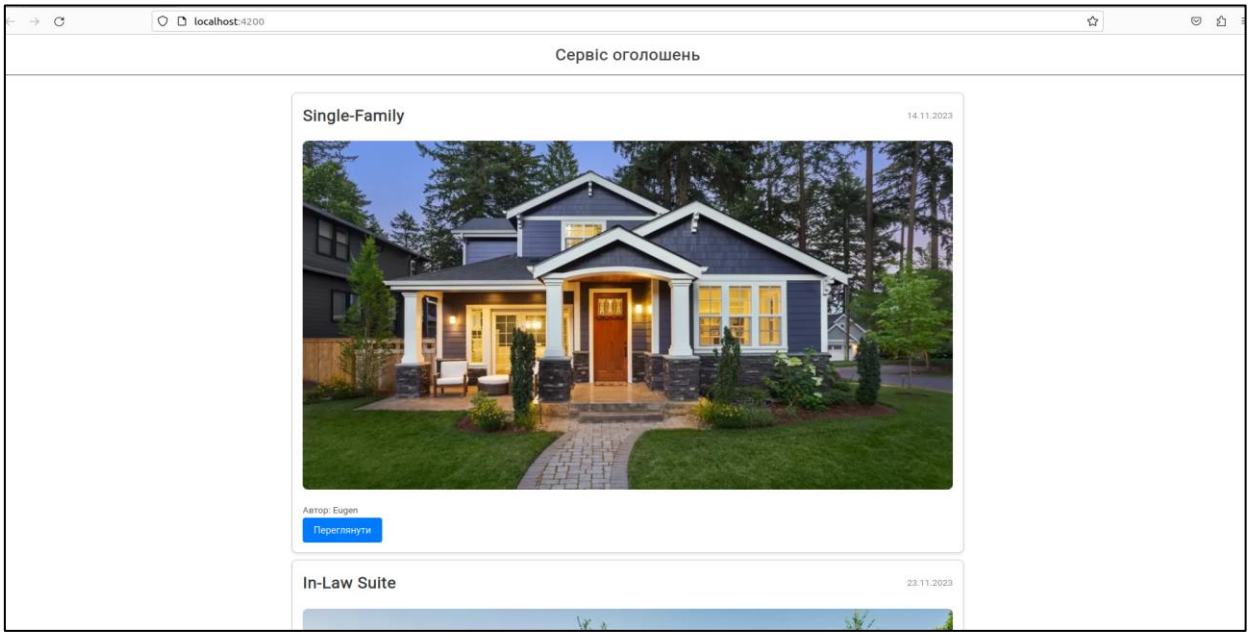

Рисунок 3.4 – Список оголошень

На цьому етапі користувач може тільки переглядати загальний список, а також переглянути детальний вміст кожного з них (рис. 3.5).

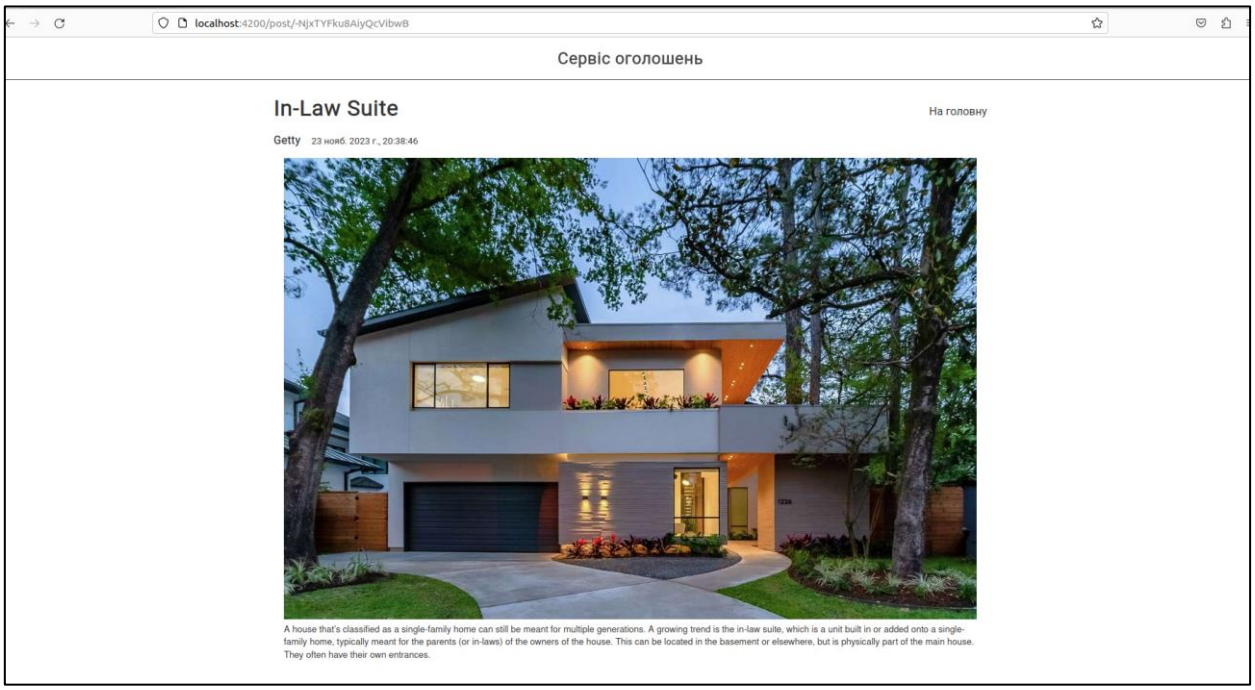

Рисунок 3.5 – Перегляд оголошення

Для створення власного оголошення, а також подальшої взаємодії з ними (редагування, видалення, тощо), користувач повинен авторизуватися в системі (рис. 3.6).

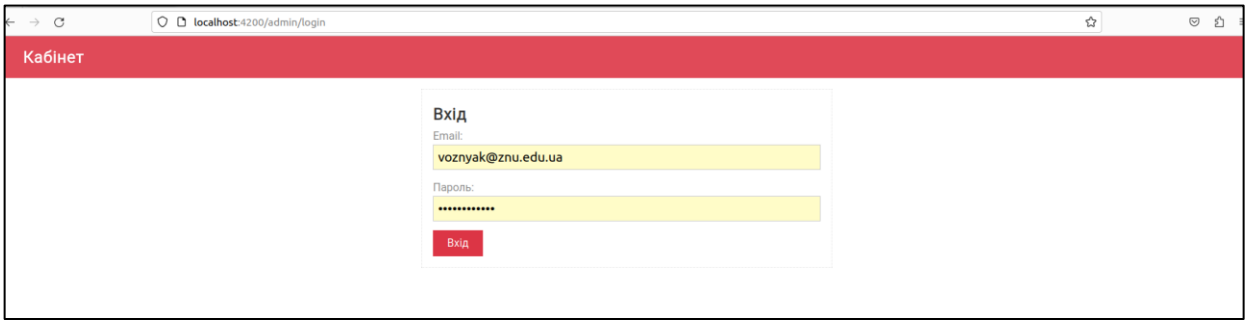

Рисунок 3.6 – Форма входу

Якщо введені дані невалідні (помилка в логіні чи паролі, неіснуючий користувач), то система сповіщає про це відповідним повідомленням (рис. 3.7).

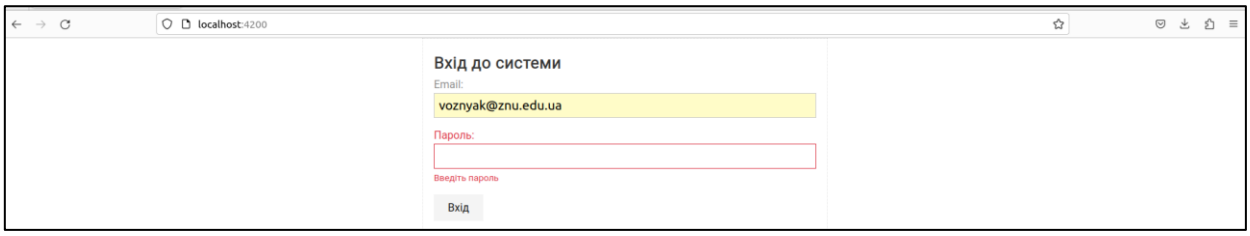

Рисунок 3.7 – Невалідні дані

Якщо дані авторизації валідні, то система відображає особистий кабінет, де користувач може взаємодіяти з існуючими оголошеннями або створювати нові (рис. 3.8).

Рисунок 3.8 – Особистий кабінет

3.7.4 Створення нового оголошення

Для створення нового оголошення, користувач повинен натиснути на кнопку «Створити», яка знаходиться в лівому куті навігаційного меню (рис. 3.9).

Рисунок 3.9 – Створення оголошення

Система відображає форму створення нового оголошення (рис. 3.10), користувач заповнює відповідні поля (назва, контент) та натискає на кнопку «Створити» (рис. 3.11).

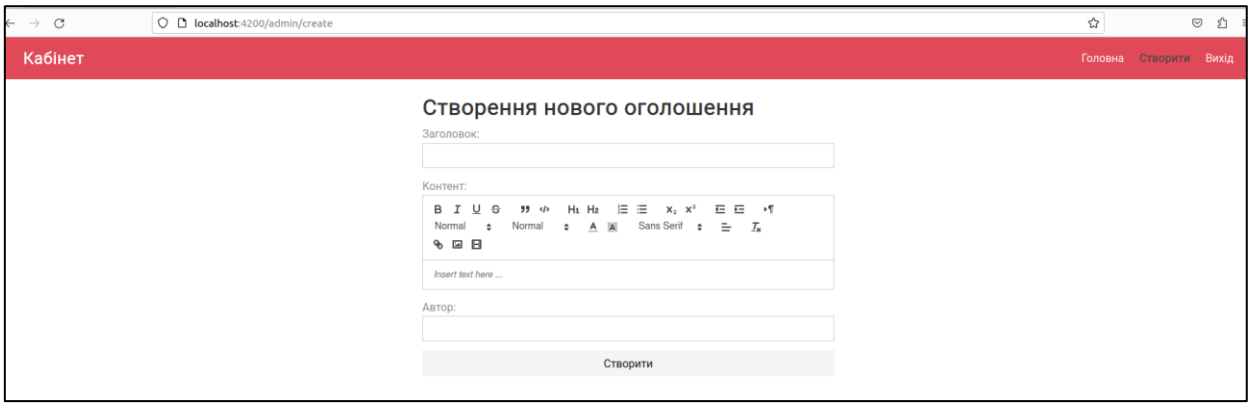

Рисунок 3.10 – Форма створення оголошення

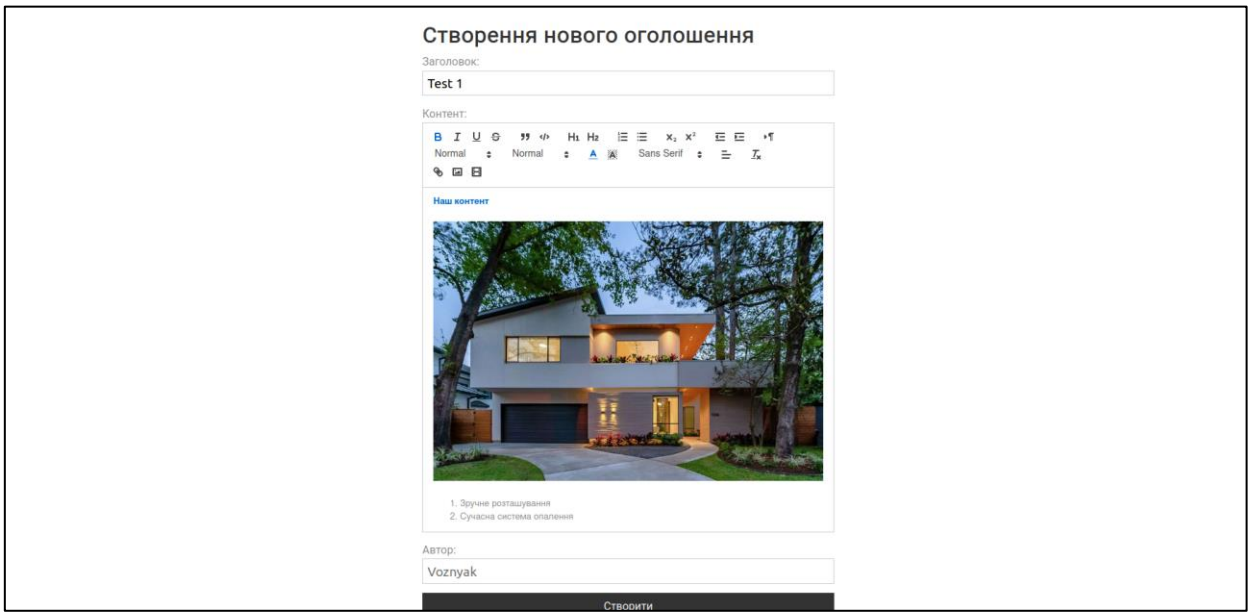

Рисунок 3.11 – Приклад заповнення форми

Система зберігає оголошення та відображає користувачу сторінку зі списком створених оголошень (у випадку адміністратора – всіх користувачів) (рис. 3.12).

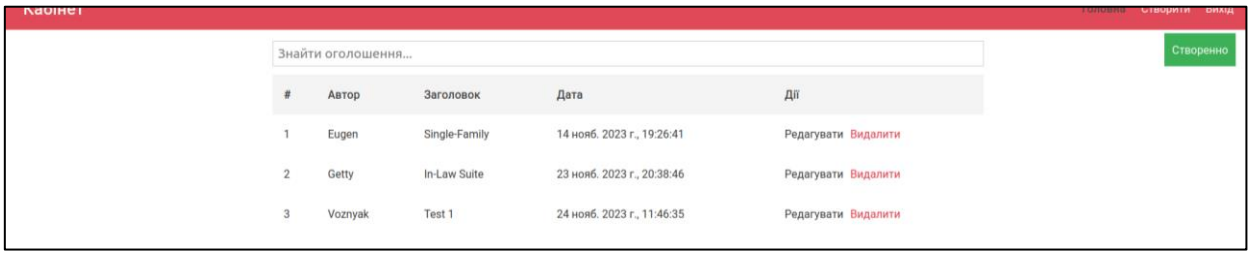

Рисунок 3.12 – Список оголошень

3.7.5 Редагування та видалення оголошення

Після створення оголошення, користувач має можливість відредагувати вміст та назву. Для цього потрібно натиснути на кнопку «Редагувати» у рядку відповідного оголошення, система відображає форму зі збереженими раніше даними (рис. 3.13). Користувач змінює дані та натискає на кнопку «Оновити», система відображає користувачу сторінку зі списком створених оголошень (рис. 3.14).

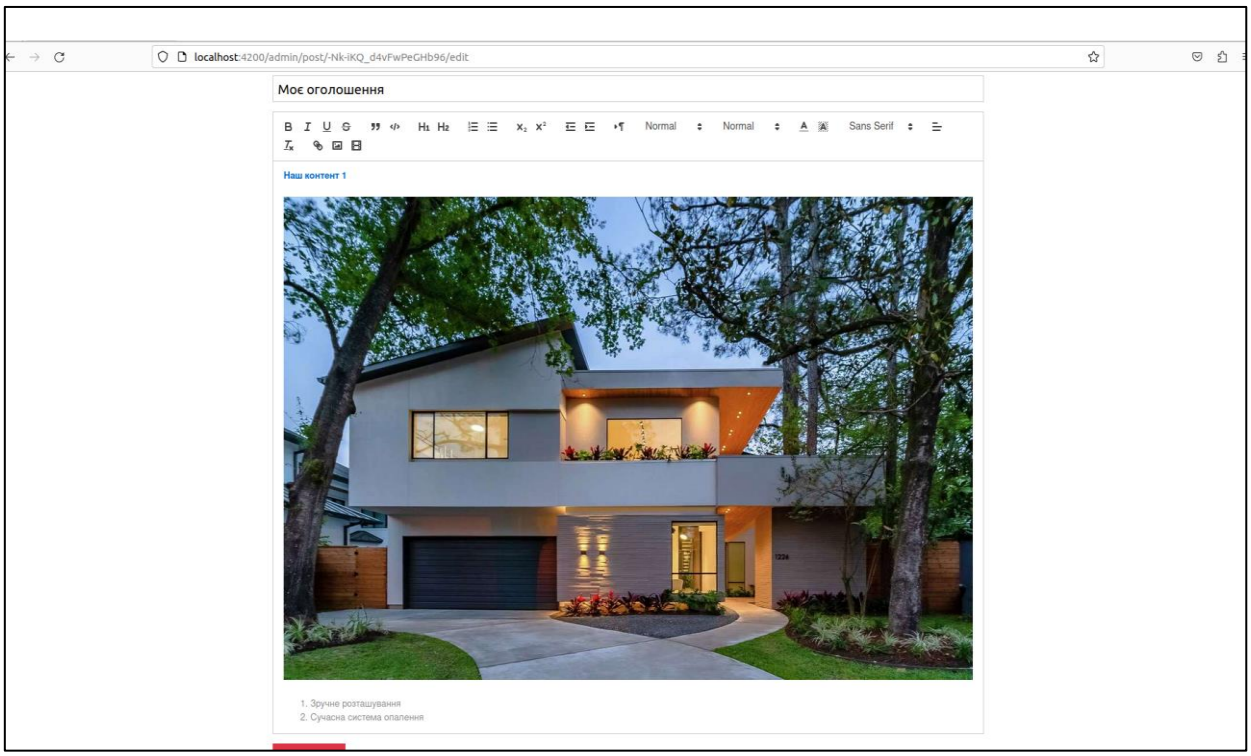

Рисунок 3.13 – Редагування оголошення

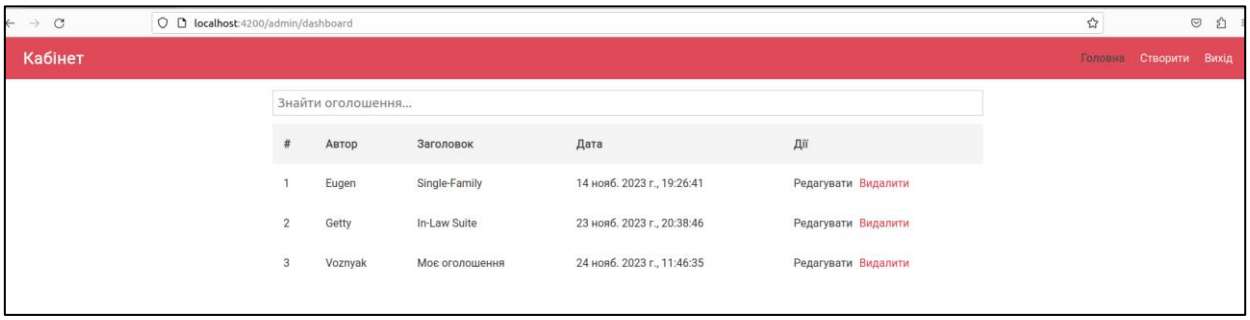

Рисунок 3.14 – Перегляд відредагованого оголошення

Для видалення оголошення, користувач повинен натиснути на кнопку «Видалити» у рядку відповідного оголошення, система видаляє оголошення та відображає відповідне повідомлення (рис. 3.15).

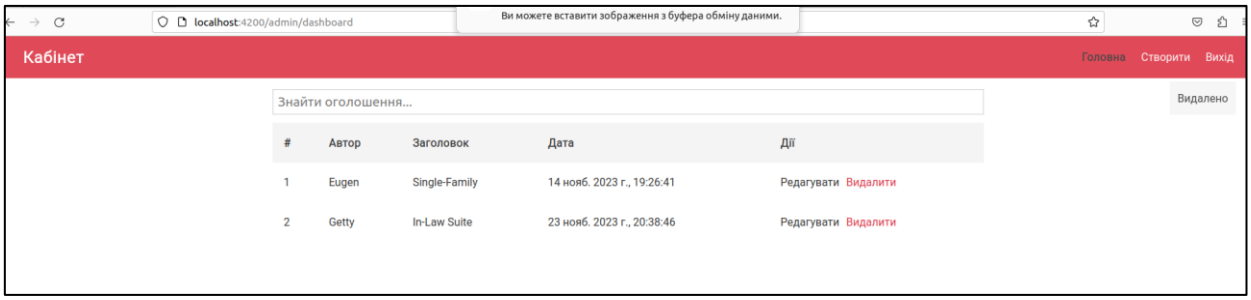

Рисунок 3.15 – Видалення оголошення

3.7.6 Пошук оголошення

Користувач має можливість шукати (фільтрувати) оголошення за повною або частковою назвою. Для цього потрібно ввести назву оголошення в поле пошуку, що знаходиться над списком оголошень. Система відображає результат пошуку (рис. 3.16).

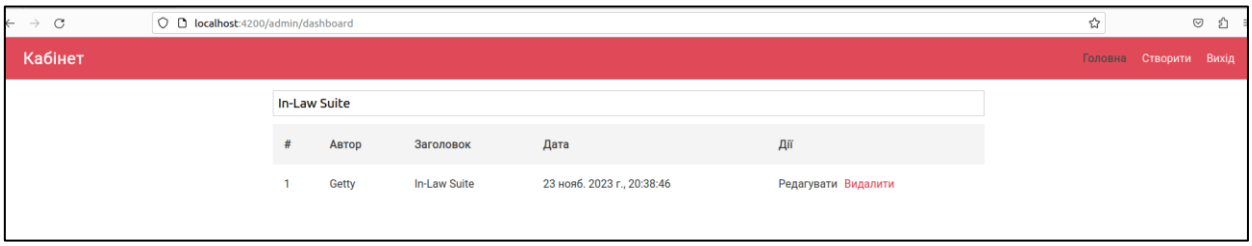

Рисунок 3.16 – Пошук оголошення

ВИСНОВКИ

В результаті роботи було написано технічне завдання на розробку системи онлайн оголошень. Для створення цієї системи було обрано фреймворк Angular та облачну БД Firebase Storage.

У відповідності з метою кваліфікаційної роботи була розроблена система онлайн оголошень із застосуванням наступних технологій:

- Firebase для зберігання даних та авторизації;
- Angular для реалізації front end частини, а також відправки запитів до БД.

У відповідності з поставленими задачами були виконані наступні етапи створення системи:

- сформовані вимоги до системи (функціональні та нефункціональні (інтерфейс, кросбраузерність, безпека, продуктивність)). Також проведено огляд предметної області та інструментів розробки;
- спроєктована та побудована структура системи (побудовані діаграми прецедентів, діяльності, послідовності та розгортання; надано детальний опис прецедентів);
- реалізовано систему онлайн оголошень (наведена інструкція по створенню компонентів системи, надано керівництво користувача та структура проєкту);
- протестована робота системи.

ПЕРЕЛІК ПОСИЛАНЬ

1. Alam A. Angular End-to-End: Master Angular from the basics to building an advanced application with Firebase’s Firestore as well as authentication. Independently published, 2023. 319 p.

2. Angular Developer Documentation. URL: https://angular.io/docs/ (дата звернення: 01.10.2023).

3. Bampakos A. Angular Projects: Build modern web apps by exploring Angular 12 with 10 different projects and cutting-edge technologies. Birmingham : Packt Publishing, 2021. 344 p.

4. Bampakos A., Deeleman P. Learning Angular: A no-nonsense guide to building web applications with Angular 15. Birmingham : Packt Publishing, 2023. 446 p.

5. Chebbi L. Reactive Patterns with RxJS for Angular: A practical guide to managing your Angular application’s data reactively and efficiently using RxJS 7. Birmingham : Packt Publishing, 2022. 224 p.

6. De Sanctis V. ASP.NET Core 6 and Angular: Full-stack web development with ASP.NET 6 and Angular 13. Packt Publishing, 2022. 780 p.

7. De Sanctis V. Building Web APIs with ASP.NET Core. Manning, 2023. 472 p.

8. Fain Y., Moiseev A. Angular Development with TypeScript. New York : Manning Publications, 2019. 560 p.

9. Freeman A. Pro Angular: Build Powerful and Dynamic Web Apps. London : Apress, 2022. 905 p.

10. Gaitatzis T. Learn REST APIs: Your guide to how to find, learn, and connect to the REST APIs that powers the Internet of Things revolution. BackupBrain Press, 2019. 109 p.

11. Kocer J. Angular 12 & ASP.NET Core 5.0 Web API. Kindle Edition, 2021. 446 p.

12. Machado K. Angular 11 e Firebase: Construindo uma aplicação integrada com a plataforma do Google. Casa do Código, 2021. 193 p.

13. Wieruch R. The Road to Firebase: Your journey to master Firebase in JavaScript. Independently published, 2021. 199 p.

ДОДАТОК А

Angular-компонент

```
import {Component, OnInit} from '@angular/core';
import {FormControl, FormGroup, Validators} from '@angular/forms';
import {Post} from '../../shared/interfaces';
import {PostsService} from '../../shared/posts.service';
import {AlertService} from '../shared/services/alert.service';

@Component({
  selector: 'app-create-page',
  templateUrl: './create-page.component.html',
  styleUrls: ['./create-page.component.scss']
})
export class CreatePageComponent implements OnInit {

  form: FormGroup;

  constructor(
    private postsService: PostsService,
    private alert: AlertService
  ) {
  }

  ngOnInit() {
    this.form = new FormGroup({
      title: new FormControl(null, Validators.required),
      text: new FormControl(null, Validators.required),
      author: new FormControl(null, Validators.required)
```

```
    })
  }

  submit() {
    if (this.form.invalid) {
      return
    }

    const post: Post = {
      title: this.form.value.title,
      author: this.form.value.author,
      text: this.form.value.text,
      date: new Date()
    }

    this.postsService.create(post).subscribe(() => {
      this.form.reset()
      this.alert.success('Створенно')
    })
  }

}
```

ДОДАТОК Б

Angular-сервіс

```
import {Injectable} from '@angular/core';
import {HttpClient} from '@angular/common/http';
import {Observable} from 'rxjs';
import {FbCreateResponse, Post} from './interfaces';
import {environment} from '../../environments/environment';
import {map} from 'rxjs/operators';

@Injectable({providedIn: 'root'})
export class PostsService {
  constructor(private http: HttpClient) {}

  create(post: Post): Observable<Post> {
    return this.http.post(`${environment.fbDbUrl}/posts.json`, post)
      .pipe(map((response: FbCreateResponse) => {
        return {
          ...post,
          id: response.name,
          date: new Date(post.date)
        }
      }))
  }

  getAll(): Observable<Post[]> {
    return this.http.get(`${environment.fbDbUrl}/posts.json`)
      .pipe(map((response: {[key: string]: any}) => {
        return Object
```

```
          .keys(response)
          .map(key => ({
            ...response[key],
            id: key,
            date: new Date(response[key].date)
          }))
      }))
  }

  getById(id: string): Observable<Post> {
    return this.http.get<Post>(`${environment.fbDbUrl}/posts/${id}.json`)
      .pipe(map((post: Post) => {
        return {
          ...post, id,
          date: new Date(post.date)
        }
      }))
  }

  remove(id: string): Observable<void> {
    return this.http.delete<void>(`${environment.fbDbUrl}/posts/${id}.json`)
  }

  update(post: Post): Observable<Post> {
    return this.http.patch<Post>(`${environment.fbDbUrl}/posts/${post.id}.json`,
post)
  }
}
```

ДОДАТОК В

Firebase Realtime Database

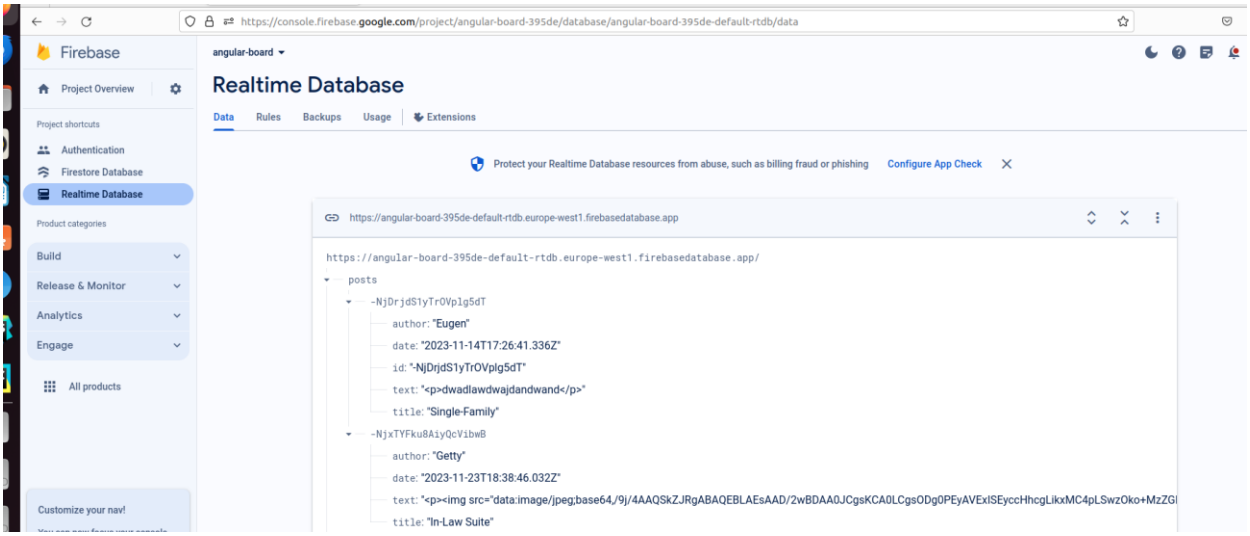

Рисунок В.1 – Колекція оголошень

ДОДАТОК Г

Посилання на GIT

https://bitbucket.org/s_var_og/board/src/master/