МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра комп'ютерних наук

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему: **«РОЗРОБКА ЗАСТОСУНКУ ДЛЯ РОЗРАХУНКУ ВАРТОСТІ ВІКОННИХ КОНСТРУКЦІЙ ТА ВАРТОСТІ РОБОТИ З МОНТАЖУ»**

Виконав: студент 4 курсу, групи 6.1261
спеціальності 126 Інформаційні системи та технології
(шифр і назва спеціальності)
освітньої програми інформаційні системи та технології
(назва освітньої програми)
К.С. Набока
(ініціали та прізвище)

Керівник доцент кафедри комп'ютерних наук, PhD, Калюжняк А.В.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Рецензент доцент кафедри фундаментальної та прикладної математики, доцент, к.ф.-м.н., Д'яченко Н.М.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя – 2025

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет математичний

Кафедра комп'ютерних наук

Рівень вищої освіти бакалавр

Спеціальність 126 Інформаційні системи та технології
(шифр і назва)

Освітня програма інформаційні системи та технології

ЗАТВЕРДЖУЮ
Завідувачка кафедри комп'ютерних наук, д.т.н., професор

_____ Шило Г.М.
(підпис)

" ____ " _____ 2024 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Набока Кіріллу Сергійовичу
(прізвище, ім'я та по-батькові)

1. Тема роботи Розробка застосунку для розрахунку вартості віконних конструкцій та вартості роботи з монтажу

керівник роботи Калюжняк Анастасія Вікторівна, PhD
(прізвище, ім'я та по-батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від « 20 » грудня 2024 року № 2001-с

2. Строк подання студентом роботи 26.05.2025 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 за темою доповіді

6. Консультанти розділів роботи

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

7. Дата видачі завдання 26.12.2024 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 роботи.	02.01.2025	
2.	Збір вихідних даних.	11.01.2025	
3.	Обробка методичних та теоретичних	03.02.2025	
	джерел.		
4.	Розробка першого та другого розділу.	24.02.2025	
5.	Розробка третього розділу.	15.05.2025	
6.	Оформлення та нормоконтроль	26.05.2025	
	кваліфікаційної роботи бакалавра.		
7.	Захист кваліфікаційної роботи.	21.06.2025	

Студент _____ К.С. Набока
(підпис) (ініціали та прізвище)

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

Нормоконтроль пройдено

Нормоконтролер _____ О.Г. Спиця
(підпис) (ініціали та прізвище)

РЕФЕРАТ

Кваліфікаційна робота бакалавра «Розробка застосунку для розрахунку вартості віконних конструкцій та вартості роботи з монтажу»: 56 с., 27 рис., 1 табл., 21 джерело, 4 додатки.

ANGULAR, EXPRESS, NODE.JS, MVC, MYSQL, REST API, RXJS, UML.

Об’єкт дослідження – системи для розрахунку вартості віконних конструкцій та вартості роботи з монтажу.

Мета роботи – розробити застосунок для розрахунку вартості віконних конструкцій та вартості роботи з монтажу.

Методи дослідження – аналітичний, моделювання, проєктування, програмний.

У роботі розглянуто проєктування та реалізацію вебзастосунку для формування замовлення на віконну конструкцію з урахуванням її параметрів, вибраних матеріалів, склопакетів, способу монтажу та додаткових послуг. Розробка виконана з використанням Angular для клієнтської частини, Node.js з Express для серверної логіки та MySQL як основної бази даних. Для взаємодії між клієнтом і сервером реалізовано REST API.

Використано UML-діаграми варіантів використання, діяльності, послідовності та розгортання. Форма замовлення реалізована у вигляді покрокового інтерфейсу з візуальним попереднім переглядом обраної конструкції та динамічним обчисленням вартості. Передбачено валідацію даних, автоматичне оновлення вартості при зміні параметрів, та збереження результатів на сервері.

Таким чином, розроблений застосунок забезпечує автоматизований механізм попереднього розрахунку вартості віконних конструкцій, полегшуючи процес оформлення замовлень як для кінцевого користувача, так і для менеджера компанії. Система є гнучкою, та масштабованою.

SUMMARY

Bachelor’s qualifying paper «Development of an Application for Calculating the Cost of Window Structures and the Cost of Installation Work»: 56 pages, 27 figures, 1 table, 21 references, 4 supplements.

ANGULAR, EXPRESS, NODE.JS, MVC, MYSQL, REST API, RXJS, UML.

The object of the study is systems for calculating the cost of window structures and the cost of installation work.

The aim of the study is to develop an application for calculating the cost of window structures and the cost of installation work.

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

The paper considers the design and implementation of a web application for placing orders for window structures, taking into account their parameters, selected materials, glass units, installation method, and additional services. The development was carried out using Angular for the client side, Node.js with Express for the server logic, and MySQL as the main database. A REST API was implemented for interaction between the client and the server.

UML diagrams of use cases, activities, sequences, and deployment were used. The order form is implemented as a step-by-step interface with a visual preview of the selected design and dynamic cost calculation. Data validation, automatic cost updates when parameters change, and saving results on the server are provided.

Thus, the developed application provides an automated mechanism for preliminary calculation of the cost of window designs, facilitating the ordering process for both the end user and the company manager. The system is flexible and scalable.

ЗМІСТ

Завдання на кваліфікаційну роботу 2
Реферат 4
Summary 5
Перелік умовних позначень, скорочень, термінів 8
Вступ 9
1 Технічне завдання та функціональні вимоги системи 11
1.1 Функціональні вимоги 11
1.1.1 Призначення і цілі створення системи 11
1.1.2 Загальні функціональні можливості системи 11
1.2 Нефункціональні вимоги 12
1.3 Опис предметної області 12
1.4 Опис системи 13
1.5 Огляд та порівняння аналогів 13
2 Проєктування інформаційної системи 16
2.1 Використання UML під час розробки системи 16
2.2 Діаграма варіантів використання 17
2.2.1 Опис варіантів використання 21
2.3 Діаграма діяльності 28
2.4 Діаграма послідовності 31
2.5 Діаграма розгортання 33
3 Програмна реалізація та тестування 35
3.1 Опис інструментів розробки 35
3.2 Основні класи системи 35
3.3 Встановлення та налаштування оточення 37
3.4 Опис API 40
3.5 Тестування проєкту 41
3.6 Керівництво користувача 43

3.6.1 Відкриття застосунку та заповнення першої частини форми 43
3.6.2 Заповнення другої частини форми 44
3.6.3 Ознайомлення з деталями замовлення та оформлення 45
3.6.4 Типові помилки валідації при заповненні форми 46
Висновки 47
Перелік посилань 48
ДОДАТОК А Unit-тест OrderFormComponent 50
ДОДАТОК Б OrderFormComponent 52
ДОДАТОК В OrderService 54
ДОДАТОК Г Epress OrderController 55

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ, ТЕРМІНІВ

Система – вебзастосунок для розрахунку вартості віконних конструкцій та вартості роботи з монтажу.

Angular – TypeScript-фреймворк для розробки масштабованих односторінкових вебзастосунків із реактивним інтерфейсом користувача [1].

Node.js – середовище виконання JavaScript-коду на сервері, що використовується для створення масштабованих вебсервісів [2].

MySQL – система керування реляційними базами даних з відкритим вихідним кодом, яка широко використовується для збереження структурованих даних [9].

Користувач – зареєстрована особа (менеджер або адміністратор), яка взаємодіє із системою згідно наданих прав.

Менеджер – користувач, який може керувати замовленнями.

Адміністратор – користувач, який має повні права.

Клієнт – незареєстрована особа, яка може тільки робити замовлення.

ДВВ – діаграма варіантів використання.

ДД – діаграма діяльності.

ДП – діаграма послідовності.

ДР – діаграма розгортання.

API – інтерфейс взаємодії між клієнтом і сервером, який реалізує обмін структурованими запитами та відповідями у форматі JSON [3].

MVC – архітектурний шаблон, що поділяє логіку додатка на три компоненти: модель, представлення й контролер [20].

JWT – відкритий стандарт для передачі інформації у вигляді токенів між сторонами для забезпечення аутентифікації та авторизації [12].

ВСТУП

У сучасному світі такі галузі як будівництво, проєктування та ремонт не стоять ні місці та активно впроваджують автоматизацію базових процесів. Це призводить до зростання попиту на цифрові інструменти, які дозволяють швидко, просто та точно виконувати попередні розрахунки вартості матеріалів, конструкцій та супутніх робіт. Одним із важливих етапів у процесі встановлення вікон є розрахунок загальної вартості віконної конструкції з урахуванням її параметрів, обраних матеріалів, склопакетів, варіантів монтажу та додаткових послуг.

На ринку існує багато різних інструментів для подібних розрахунків, однак більшість із них не мають зручного та інтуїтивно зрозумілого інтерфейсу користувача, не розраховують автоматично фінальну вартісь або не мають розширеного вибору параметрів та послуг.

Враховуючи вищесказане, а також з огляду на зростання попиту на інтерактивні вебзастосунки, виникає потреба у створенні відкритої, адаптивної та масштабованої системи, що дозволяє користувачу розраховувати вартість віконної конструкції в режимі реального часу та залишати свої контактні дані для подальшого оформлення замовлення і при цьому не потребуватиме авторизації чи спеціальних навичок.

Актуальність роботи полягає у створенні сучасного інтерактивного застосунку, що дозволяє користувачу швидко, просто та зручно сформувати замовлення на віконну конструкцію з точним підрахунком вартості, не звертаючись до менеджера на ранніх етапах.

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

Мета: розробити застосунок для розрахунку вартості віконних конструкцій та монтажу з інтерактивною формою для користувача.

Досягнення поставленої мети передбачає реалізацію наступних завдань:

– сформулювати функціональні та нефункціональні вимоги до

застосунку;

- проаналізувати та порівняти аналоги;
- спроєктувати та побудувати архітектуру застосунку;
- реалізувати застосунок для розрахунку вартості віконних конструкцій та вартості роботи з монтажу;
- протестувати застосунок та надати керівництво користувача.

Об’єкт дослідження: системи для розрахунку вартості віконних конструкцій та вартості роботи з монтажу.

Предмет дослідження: Angular, Node.js та взаємодія з базою даних MySQL.

Методи дослідження: аналітичний, моделювання, проєктування, програмний.

У першому розділі надано аналіз предметної області, описано функціональні й нефункціональні вимоги, сформульовано технічне завдання, розглянуто існуючі аналоги та побудовано таблицю порівняння.

У другому розділі спроєктовано архітектуру застосунку, описано доречність використання UML, надані основні діаграми, а також описані варіанти використання.

У третьому розділі описано інструменти розробки, розгортання серверної та клієнтської частини, структуру API, приклади модульного тестування, а також наведено повноцінне керівництво користувача та типові помилки валідації.

1 ТЕХНІЧНЕ ЗАВДАННЯ ТА ФУНКЦІОНАЛЬНІ ВИМОГИ СИСТЕМИ

1.1 Функціональні вимоги

1.1.1 Призначення і цілі створення системи

Функціональне призначення системи – надати можливість клієнтам оформлювати замовлення на виготовлення вікон, вводячи необхідні параметри та відразу отримуючи розраховану вартість з урахуванням монтажу та додаткових послуг.

Експлуатаційне призначення – система експлуатується клієнтами на рівні оформлення замовлення, а також менеджерами та адміністраторами на рівні керування замовленнями та системою відповідно.

Мета створення системи – розробити сучасний вебзастосунок, що дозволяє швидко формувати замовлення на встановлення вікон та точно оцінювати його вартість.

1.1.2 Загальні функціональні можливості системи

Під час процесу оформлення замовлення, система має надавати клієнтам наступні можливості:

- задавати розміри вікна (висота, ширина);
- обирати тип віконної конструкції;
- обирати матеріал профілю;
- обирати тип склопакету;
- визначати монтажну опцію;
- додавати додаткові послуги;
- переглядати загальну вартість замовлення;

– вводити персональні дані.

Для керування замовленнями система має надавати менеджерам наступні можливості:

– переглядати/редагувати/видаляти замовлення.

Система має надавати адміністраторам такі додаткові можливості:

– створення/перегляд/редагування/видалення користувачів;

– оновлення ролі користувача.

Для всіх користувачів система має надавати такі спільні можливості:

– вхід/вихід до/з системи.

1.2 Нефункціональні вимоги

Інтерфейс користувача. Інтерфейс користувача повинен бути повинен бути інтуїтивно зрозумілим та адаптивним для всіх типів пристроїв.

Підтримка браузерів. Система має підтримувати актуальні версії розповсюджених браузерів: Chrome, Firefox, Edge, Safari.

Вимоги до продуктивності. Розрахунок вартості має виконуватись не довше 1 секунди. Список замовлень повинен завантажуватися не довше 2 секунд при 100+ записах.

Вимоги до безпеки. Пароль зберігається у вигляді хешу. Доступ до API обмежується токеном. Сервер захищає внутрішню логіку від прямого доступу користувача. Незареєстровані користувачі не повинні мати доступу до адмін-панелі.

1.3 Опис предметної області

Сфера виготовлення та встановлення віконних конструкцій передбачає постійну необхідність у швидкому формуванні кошторисів. Класичні підходи,

де менеджери вручну розраховують суму залежно від параметрів, є неефективними та часозатратними. Тому автоматизація розрахунку вартості замовлення є важливим та актуальним завдання для успішного ведення бізнесу.

Вебзастосунок враховує всі важливі параметри, а саме: розмір конструкції, тип профілю, тип склопакету, складність монтажу, додаткові послуги. Це дозволяє пришвидшити процес розрахунку фінальної вартості, а також максимально скоротити можливість помилки.

Застосунок повинен зберігати всі замовлення у базі даних, підтримувати фільтрацію та сортування, а також мати захист від несанкціонованого доступу.

1.4 Опис системи

Система побудована за архітектурою «клієнт-сервер»:

- Клієнт – написаний на Angular 19, відповідає за введення параметрів і відображення результатів;
- Сервер – реалізовано на Node.js з використанням Express та Sequelize;
- База даних – реалізована на MySQL, що забезпечує структуру таблиць для типів вікон, матеріалів, замовлень, додаткових послуг тощо.

При створенні замовлення користувач заповнює форму, і система автоматично виконує розрахунок за допомогою формул, враховуючи базову ціну + множники (матеріал, склопакет, монтаж) + фіксовані доплати.

1.5 Огляд та порівняння аналогів

Ринок України у сфері встановлення віконних конструкцій представлений низкою глобальних та регіональних компаній. Багато з них мають власні онлайн-сервіси з калькуляторами розрахунку вартості робіт. Нижче наведено огляд деяких з них.

OKNA.ua має калькулятор, який дозволяє визначити вартість вікна залежно від його розмірів, типу та енергоефективних характеристик. Розрахунок проводиться за пропозиціями від понад 100 компаній із різних регіонів України. Крім металопластикових вікон, калькулятор дозволяє розрахувати ціну на балконне скління, балконні або офісні двері, вікна та офісні двері з фрамугою, а також розсувний балкон. Калькулятор дозволяє не тільки обрати, а і замовити пластикові вікна онлайн [13].

VEKA.ua надає калькулятор для розрахунку вартості замовлення на основі типу вибраного вікна чи двері, розмірів і властивостей конструкції, а також додаткових характеристик. Після введення необхідних параметрів, таких як розміри вікна, тип склопакета, колір та додаткові опції, замовлення надсилається на опрацювання менеджеру, який повідомляє клієнту фінальну ціну [15].

WDS не має стандартного калькулятора розрахунку цін, а пропонує калькулятор вітрових навантажень на вікна, який дозволяє користувачам вибрати вітрову зону, ввести розміри будівлі та розрахувати значення вітрових навантажень [19].

Калькулятор пластикових вікон компанії «Лінія Вікон» дозволяє розрахувати вартість вікна за розмірами, типу конструкції та додатковим параметрам. Калькулятор також дозволяє врахувати вартість демонтажу/монтажу. При зміні параметру калькулятора, він автоматично перераховує ціну, що дозволяє відразу отримати фінальну вартість замовлення. Також варто зазаначити, що це перший в Україні віконний калькулятор, який використовує штучний інтелект [21].

Vikna.ua надає калькулятор вікон, який допомагає користувачам обрати розмір і комплектацію вікна, але як і у випадку з VEKA.ua необхідно залишити заявку на розрахунок точної вартості [16].

У таблиці 1.1 наведено порівняння описаних вище систем з системою, що розробляється.

Таблиця 1.1 – Порівняння аналогів

Система	**Онлайн розрахунок**	**Врахування монтажу**	**Адаптивний дизайн**	**Додаткові послуги**	**Особливості**
OKNA.ua	Так	Частково	Так	Так	Представлено 100+ компаній [13]
VEKA.ua	Ні	Ні	Так	Так	Для рохрахунку вартості потрібен контакт з менеджером [15]
WDS	Ні	Ні	Так	Ні	Розрахунок вітрових навантажень [19]
Лінія Вікон	Так	Так	Так	Так	Використання штучного інтелекту [21]
Vikna.ua	Ні	Ні	Так	Так	Для розрахунку вартості потрібен контакт з менеджером [16]
Дана система	Так	Так	Так	Так	Повна автоматизація

Аналіз існуючих систем показує, що багато з них надають можливість онлайн-розрахунку вартості віконних конструкцій з урахуванням різних параметрів. Проте, не всі враховують вартість монтажу та додаткових послуг, що може бути важливим для користувачів. Також не всі системи відразу відображають користувачу фінальну вартість, що може відштовхнути при дослідженні ним різних параметрів, так як для уточнення необхідна консультація з менеджером. Наша система відрізняється повною автоматизацією процесу розрахунку, врахуванням всіх необхідних параметрів, адаптивним дизайном та зручністю використання, що робить її конкурентоспроможною на ринку.

2 ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1 Використання UML під час розробки системи

Уніфікована мова моделювання (Unified Modeling Language, UML) – це стандартизована мова моделювання, яка використовується з метою візуалізації, специфікації, проектування та документування компонентів програмних систем. UML забезпечує стандартизований підхід до формулювання структурних і функціональних характеристик систем, незалежно від обраної платформи або мови програмування [7].

UML є мовою проєктування та не має прямого відношення до програмування. Основна функція UML полягає в тому, щоб дозволити розробникам сконцентруватися на логіці системи до початку процесу реалізації. Це важливо в контексті розробки складних інформаційних систем, таких як веб-додатки з клієнт-серверною архітектурою, де взаємодіють численні компоненти, сутності та ролі користувачів [8, 17].

Основними перевагами використання UML є:

- стандартизованість – UML це стандарт, розроблений та підтримуваний консорціумом Object Management Group (OMG);
- візуалізація системи – UML-діаграми дозволяють розробникам та замовникам спільно пропрацьовувати архітектуру майбутньої системи;
- підтримка об’єктно-орієнтованого підходу – UML добре інтегрується з принципами інкапсуляції, наслідування та поліморфізму;
- зменшення витрат на розробку завдяки формалізації вимог ще на ранньому етапі [7, 17].

UML забезпечує підтримку різноманітних діаграм, які умовно поділяються на дві категорії: структурні (відображають статичні аспекти системи) та поведінкові (моделюють динаміку процесів) [7, 8]. Для моделювання архітектури та функціонування додатку в цій дипломній роботі

були використані наступні діаграми:

- Діаграма варіантів використання – для відображення функціональних вимог та взаємодії користувачів із системою. Будується з точки зору саме користувача, а не розробника;
- Діаграма діяльності – для опису алгоритмів обробки замовлення;
- Діаграма послідовності – для моделювання сценаріїв взаємодії між клієнтом, сервером та базою даних;
- Діаграма розгортання – для демонстрації розміщення компонентів системи на фізичному рівні.

Використання цих діаграм дозволяє забезпечити узгодженість логіки роботи, структур даних та реалізації інтерфейсів.

Практичні результати показують, що використання UML при розробці IT-систем призводить до суттєвого підвищення якості проектної документації, сприяє підвищенню ефективності командної роботи та зменшенню кількості помилок під час впровадження [17].

2.2 Діаграма варіантів використання

Діаграма варіантів використання – це основний тип діаграм у мові UML, який полегшує зображення функціональності системи з точки зору її користувачів (акторів). Вона застосовується для моделювання взаємодії між зовнішніми сутностями (акторами) та системою, що розробляється [18].

Основне призначення ДВВ полягає в тому, щоб формально відобразити вимоги до системи, проілюструвати дії (сценарії), які може виконувати кожен тип користувачів, і продемонструвати, яким чином ці дії пов'язані між собою. Ця система забезпечує виконання наступних функцій:

- формалізувати функціональну специфікацію;
- узгодити бачення системи між розробниками, аналітиками та замовниками;

– забезпечити основу для створення тестів, діаграм діяльності, послідовності, класів [17, 18].

Основним поняттям ДВВ є актор, який представляє зовнішній суб’єкт, який взаємодіє із системою. Це може бути реальна особа, наприклад користувач чи адміністратор, а також інша система або пристрій, що потребують доступу до функціоналу системи [18].

Другим основним поняттям є варіант використання, який являє собою опис функціональної можливості системи з точки зору її взаємодії з актором. Варіанти використання концентруються на тому, що робить система, а не на тому, як саме вона це виконує, що дозволяє зосередитись на бізнес-логіці та потребах користувачів [18].

Зв’язки в діаграмах представлені лініями між акторами і варіантами використання, які позначають взаємодію між ними. Крім базових зв1язків, використовуються спеціальні типи: include для повторного включення одного сценарію в інші, extend для умовного розширення сценарію додатковою функціональністю, та generalization для успадкування дій актора чи варіанта використання [18].

Діаграми варіантів використання надають зрозумілий спосіб фіксації вимог до системи, що робить їх доступними для всіх учасників проєкту незалежно від їхнього технічного рівня. Вони забезпечують ефективне спілкування між зацікавленими сторонами, дозволяючи бізнес-аналітикам, розробникам та замовникам спілкуватися однією мовою [7, 18].

Підтримка ієрархічної декомпозиції функціоналу дає можливість структурувати складні системи від загальних сценаріїв до детальних варіантів використання. Крім того, діаграми варіантів використання зручні для подальшої трансформації в інші види діаграм, що полегшує перехід від аналізу вимог до проектування та реалізації системи [7, 8].

На рисунку 2.1 зображена ДВВ вебзастосуноку для розрахунку вартості віконних конструкцій та вартості роботи з монтажу.

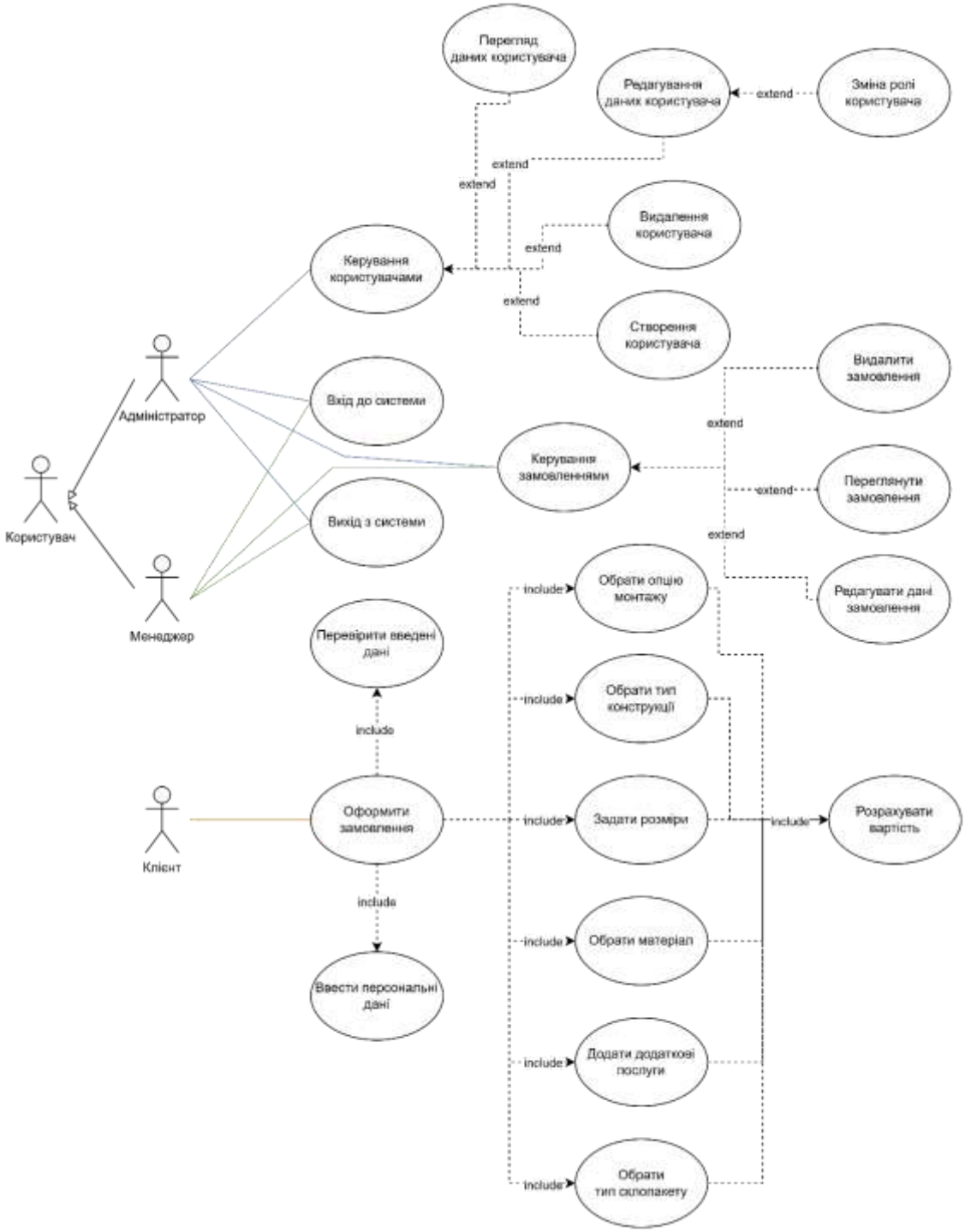


Рисунок 2.1 – Діаграма варіантів використання

Актори на діаграми діляться на дві умовні групи: користувачі системи та зовнішні користувачі. Користувачі системи представлені акторами «Адміністратор» та «Менеджер», які взаємодіють з нею на різних рівнях, починаючи від керування замовленнями і закінчуючі повним керуванням системою (рис. 2.2). Ці актори об’єднані у одного спільного – «Користувач». Зі

сторони зовнішніх користувачів представлено актора «Клієнт», який взаємодіє із зовнішнім інтерфейсом для розрахунку вартості віконної конструкції з подальшою можливістю оформлення замовлення (рис. 2.3).

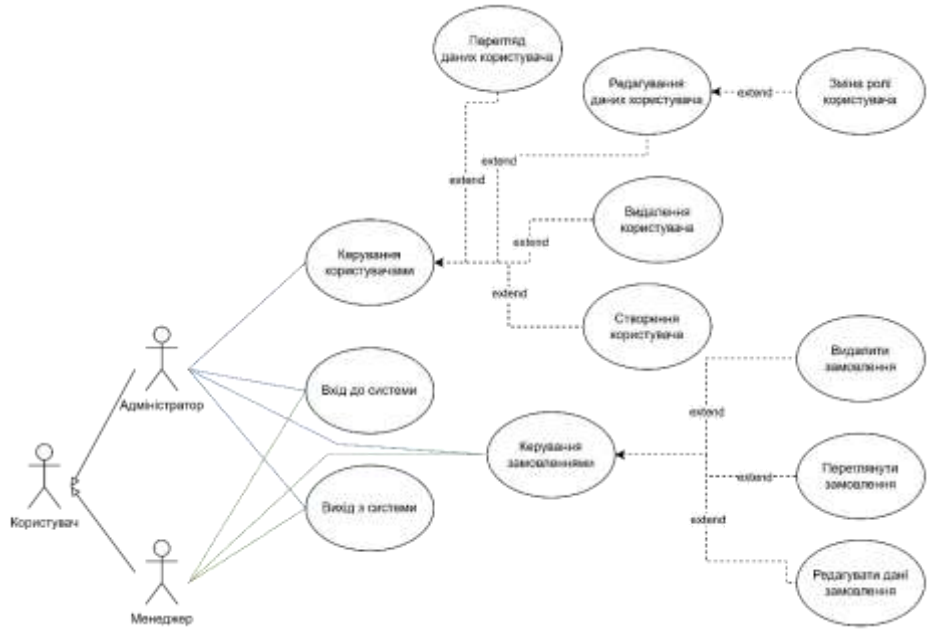


Рисунок 2.2 – ДВВ акторів «Менеджер» та «Адміністратор»

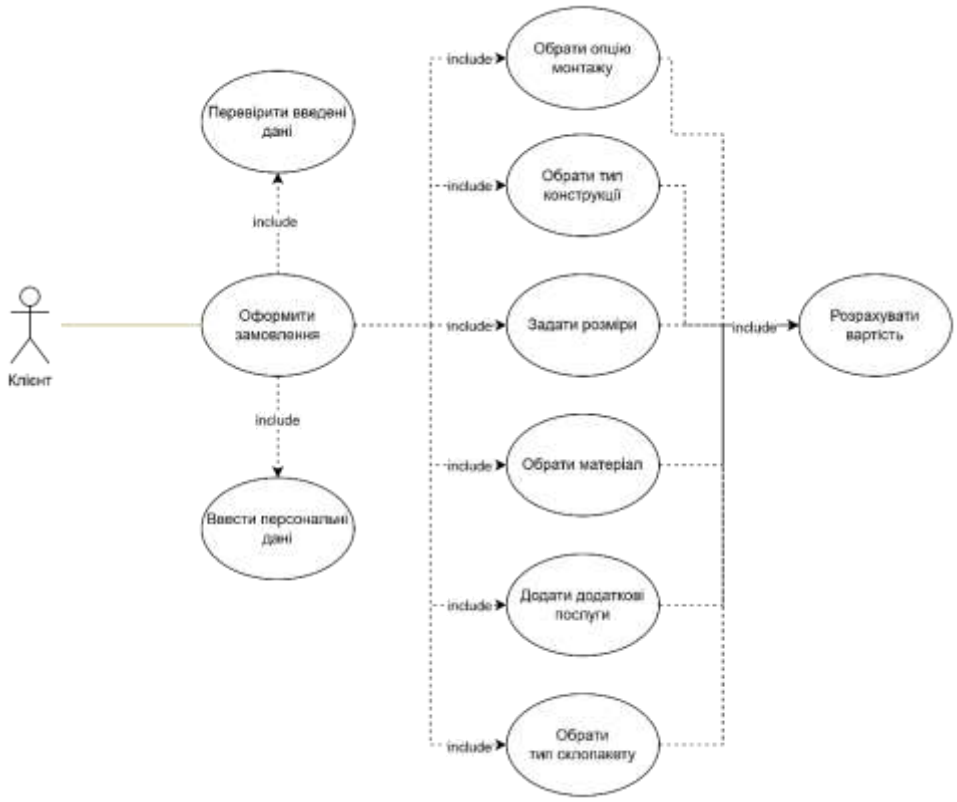


Рисунок 2.3 – ДВВ актора «Клієнт»

Виділено 1 основний варіант використання – «Оформити замовлення». Даний сценарій є основним бізнес-процесом всієї системи. Після того, як клієнт відкриває головну сторінку системи, відображається форма введення параметрів відповідним зображенням типу вікна. Ця форма не вимагає входу до системи чи попередньої реєстрації. Клієнт повинен заповнити необхідні поля, при зміні будь-якого з них, система автоматично перераховує та відображає актуальну суму. Для продовження оформлення замовлення, клієнту необхідно натиснути на кнопку «Ввести персональні дані» та заповнити відповідні поля. Після цього він натискає на кнопку «Попередній перегляд», система відображає обрані клієнтом параметри, суму замовлення та персональні дані. Для завершення процесу оформлення замовлення клієнт натискає на кнопку «Оформити замовлення».

Цей варіант використання не потребує автентифікації, що робить взаємодію зі сторони клієнта максимально простою та швидкою. Усі розрахунки виконуються динамічно на стороні клієнта, а серверна частина обробляє лише фінальний запит на створення замовлення.

2.2.1 Опис варіантів використання

Прецедент «Оформити замовлення»

Призначення: даний варіант використання надає можливість клієнту оформити замовлення віконної конструкції.

Основний потік подій: даний варіант використання починає виконуватися одразу після відкриття головної сторінки системи. Клієнту відображається покрокова форма, яка складається з трьох етапів. На першому кроці користувач обирає параметри вікна – вводить ширину й висоту, обирає тип конструкції, матеріал профілю, тип склопакету, монтажну опцію та додаткові послуги. Після кожної дії система автоматично розраховує загальну вартість. На другому кроці користувач вводить особисті дані: ім’я та номер телефону. На третьому кроці

система відображає всі введені дані для перевірки. Користувач натискає на кнопку «Оформити замовлення», система зберігає замовлення до бази даних і виводить повідомлення про успішне завершення оформлення.

Виняткова ситуація 1: не заповнені обов’язкові поля форми– система підсвічує відповідні елементи, відображає помилку і не дозволяє перейти до наступного кроку.

Виняткова ситуація 2: введено невалідні дані – система підсвічує відповідні елементи, відображає помилку валідації і не дозволяє перейти до наступного кроку.

Виняткова ситуація 3: помилка зв’язку з сервером – система повідомляє про неможливість збереження замовлення та пропонує повторити спробу пізніше.

Прецедент «Вхід у систему»

Призначення: даний варіант використання надає можливість користувачам увійти до системи.

Основний потік подій: даний варіант використання починає виконуватися, коли користувач відкриває сторінку входу, вводить email і пароль. Система перевіряє правильність введених облікових даних, і, у разі успішної перевірки, надає доступ до інтерфейсу керування замовленнями або користувачами відповідно до ролі.

Передумова: користувач повинен бути зареєстрованим у системі.

Виняткова ситуація 1: введено неправильні облікові дані – система повідомляє про помилку авторизації.

Виняткова ситуація 2: сервер недоступний – система повідомляє користувача про технічну помилку й рекомендує спробувати пізніше.

Прецедент «Вихід з системи»

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

Основний потік подій: даний варіант використання починає виконуватися, коли користувач натискає на кнопку «Вийти з системи». Система

видаляє токен авторизації та відображає головну сторінку.

Передумова: користувач повинен бути автризованим у системі.

Прецедент «Обрати тип конструкції»

Призначення: даний варіант використання надає можливість клієнту обрати тип конструкції вікна (глухе, поворотне, поворотно-відкидне, розсувне тощо) як частину процесу створення замовлення.

Основний потік подій: даний варіант використання починає виконуватися, коли клієнт обирає один із доступних типів конструкцій у випадаючому списку. Після цього система відображає схематичне зображення вибраного типу, а також використовує відповідну базову вартість у розрахунках.

Виняткова ситуація 1: список типів недоступний або порожній – система повідомляє про помилку завантаження даних.

Прецедент «Задати розміри»

Призначення: даний варіант використання надає можливість клієнту вказати ширину та висоту вікна.

Основний потік подій: даний варіант використання починає виконуватися, коли клієнт заповнює поля «Ширина» і «Висота». Система приймає дані у міліметрах, обраховує площу конструкції та використовує її в подальших обчисленнях вартості.

Виняткова ситуація 1: введено недопустиме або некоректне значення — система блокує подальший перехід і підсвічує помилки.

Прецедент «Обрати матеріал профілю»

Призначення: даний варіант використання надає можливість клієнту обрати матеріал, з якого виготовлений профіль вікна (ПВХ, алюміній, дерево).

Основний потік подій: даний варіант використання починає виконуватися, коли клієнт обирає один із доступних матеріалів у випадаючому списку. Кожен матеріал має власний коефіцієнт вартості, який система використовує при розрахунках.

Виняткова ситуація 1: список матеріалів недоступний – система

повідомляє користувача про технічну проблему.

Прецедент «Обрати тип склопакету»

Призначення: даний варіант використання надає можливість клієнту обрати варіант склопакету (однокамерний, двокамерний, енергозберігаючий).

Основний потік подій: даний варіант використання починає виконуватися, коли клієнт обирає тип склопакету з випадаючого списку, система враховує відповідний множник вартості у фінальному розрахунку.

Виняткова ситуація 1: список недоступний – система повідомляє про помилку завантаження склопакетів.

Прецедент «Обрати монтажну опцію»

Призначення: даний варіант використання надає можливість клієнту вказати поверховість або наявність ліфта для врахування складності монтажу.

Основний потік подій: даний варіант використання починає виконуватися, коли клієнт обирає один із варіантів (1 поверх, 2–4 поверх, 5+ з ліфтом або без). Кожен варіант має власний коефіцієнт вартості, який додається до розрахунку.

Виняткова ситуація 1: помилка завантаження опцій монтажу – система повідомляє користувача.

Прецедент «Додати додаткові послуги»

Призначення: даний варіант використання надає можливість клієнту вибрати додаткові послуги (встановлення підвіконня, відлив, откоси, демонтаж старого вікна).

Основний потік подій: даний варіант використання починає виконуватися, коли клієнт відмічає потрібні послуги у списку. Система підсумовує вартість обраних послуг до загальної вартості.

Виняткова ситуація 1: перелік послуг недоступний – система повідомляє про неможливість завантаження даних.

Прецедент «Розрахувати вартість»

Призначення: даний варіант використання забезпечує автоматичний динамічний розрахунок вартості замовлення після кожної зміни параметра.

Основний потік подій: даний варіант використання починає виконуватися після кожної зміни у формі (тип, розмір, матеріал, послуги) система автоматично виконує перерахунок вартості в реальному часі. Клієнту відображається актуальна ціна.

Виняткова ситуація 1: неможливо виконати обчислення через відсутність ключових даних – система не виводить результат і пропонує завершити заповнення форми.

Прецедент «Ввести персональні дані»

Призначення: даний варіант використання надає можливість клієнту ввести персональну інформацію, необхідну для обробки замовлення.

Основний потік подій: даний варіант використання починає виконуватися на другому кроці форми, клієнт вводить ім’я та номер телефону. Ці дані зберігаються разом із замовленням і використовуються менеджером для зв’язку з клієнтом.

Виняткова ситуація 1: не заповнені обов’язкові поля форми– система підсвічує відповідні елементи, відображає помилку і не дозволяє перейти до наступного кроку.

Виняткова ситуація 2: введено невалідні дані – система підсвічує відповідні елементи, відображає помилку валідації і не дозволяє перейти до наступного кроку.

Прецедент «Перевірити введені дані»

Призначення: даний варіант використання надає можливість клієнту переглянути всю введену інформацію перед оформленням замовлення.

Основний потік подій: даний варіант використання починає виконуватися на третьому кроці покрокової форми, система відображає підсумкову таблицю з усіма введеними параметрами, включаючи обрані опції та розраховану вартість.

Прецедент «Переглянути замовлення»

Призначення: даний варіант використання надає можливість менеджеру переглядати всі замовлення, оформлені клієнтами через систему.

Основний потік подій: даний варіант використання починає

виконуватися, коли після входу до системи менеджер відкриває розділ «Замовлення». Система надсилає запит до серверної частини й повертає список усіх замовлень. Дані відображаються у вигляді таблиці, де кожен запис містить інформацію про клієнта, параметри вікна, обрані послуги та загальну вартість.

Виняткова ситуація 1: помилка при отриманні даних – система повідомляє про проблему та пропонує оновити сторінку.

Прецедент «Редагувати замовлення»

Призначення: даний варіант використання надає можливість менеджеру вносити зміни до існуючих замовлень.

Основний потік подій: даний варіант використання починає виконуватися, коли менеджер обирає зі списку потрібне замовлення та натискає на піктограму «Редагувати». Система відображає форму редагування у яку завантажує поточні дані, які можна змінити. Після редагування менеджер натискає кнопку «Оновити замовлення», і система зберігає зміни до бази даних.

Виняткова ситуація 1: введено невалідні дані – система виводить повідомлення про помилку та не виконує оновлення.

Виняткова ситуація 2: помилка збереження – система відображає повідомлення про тимчасову недоступність сервера.

Прецедент «Видалити замовлення»

Призначення: даний варіант використання надає можливість менеджеру видалити помилкове або неактуальне замовлення.

Основний потік подій: даний варіант використання починає виконуватися, коли у списку замовлень менеджер натискає на піктограму «Видалити» навпроти обраного запису. Система запитує підтвердження дії. Після підтвердження замовлення видаляється з бази даних.

Виняткова ситуація 1: помилка при видаленні – система повідомляє про неможливість виконання операції.

Виняткова ситуація 2: сервер не відповідає – система відображає повідомлення про тимчасову недоступність сервера.

Прецедент «Створити користувача»

Призначення: даний варіант використання надає можливість адміністратору створити нового користувача з роллю менеджера.

Основний потік подій: даний варіант використання починає виконуватися, коли адміністратор відкриває розділ «Користувачі», натискає кнопку «Створити» та вводить email і пароль. Система перевіряє валідність введених даних і створює обліковий запис нового менеджера.

Виняткова ситуація 1: email вже існує – система повідомляє про те, що користувач вже існує.

Виняткова ситуація 2: невалідні дані – система блокує створення користувача та виводить повідомлення про помилку.

Прецедент «Переглянути користувачів»

Призначення: даний варіант використання надає можливість адміністратору переглядати список усіх користувачів, створених у системі.

Основний потік подій: даний варіант використання починає виконуватися, коли адміністратор переходить у розділ «Користувачі». Система виконує запит до бази даних і повертає перелік користувачів у вигляді таблиці з контактними даними та роллю.

Виняткова ситуація 1: серверна помилка – система повідомляє про тимчасову недоступність сервера.

Прецедент «Редагувати користувача»

Призначення: даний варіант використання надає можливість адміністратору змінювати дані існуючих користувачів.

Основний потік подій: даний варіант використання починає виконуватися, коли адміністратор обирає зі списку користувача, відкриває форму редагування та змінює дані (email, роль). Після натискання кнопки «Зберегти» система оновлює інформацію в базі даних.

Виняткова ситуація 1: серверна помилка – зміни не зберігаються, система відображає повідомлення про збій.

Прецедент «Видалити користувача»

Призначення: даний варіант використання надає можливість

адміністратору видалити обліковий запис користувача.

Основний потік подій: даний варіант використання починає виконуватися, коли адміністратор натискає на кнопку «Видалити» навпроти користувача. Система запитує підтвердження, після чого видаляє обліковий запис.

Виняткова ситуація 1: помилка видалення – система повідомляє про проблему та пропонує повторити спробу.

2.3 Діаграма діяльності

Діаграма активності є поведінковим елементом мови UML, що полегшує моделювання послідовності дій або логіки окремого бізнес-процесу. Ця діаграма пропонує практичний підхід для роз'яснення основної послідовності подій у варіанті використання, особливо в випадках, коли є розгалуження або альтернативні сценарії [7, 8, 18].

Діаграми активності використовуються для представлення алгоритмів, процедур, робочих процесів, а також логіки взаємодії в межах інформаційної системи. Вони особливо ефективні у візуалізації бізнес-процесів, оскільки дозволяють зображати дії, умови переходів, паралельні процеси та точки прийняття рішень. Кожна дія в діаграмі активності позначається прямокутником із закругленими кутами, а потоки управління – стрілками, що вказують напрямок виконання [7, 8, 18].

Однією з ключових переваг діаграм активності є можливість відображення як послідовного, так і паралельного виконання операцій. Для цього використовуються спеціальні конструкції: вузли розгалуження, вузли об’єднання, точки синхронізації та умовні переходи. Такий підхід дозволяє моделювати навіть складні сценарії бізнес-логіки, включаючи повторення, залежності між діями, і обробку виняткових ситуацій. Також, діаграми активності пов’язані з варіантами використання: на їх основі часто будуються

діаграми, які деталізують сценарії взаємодії користувача з системою [7, 8, 18].

Основним бізнес-процесом данної роботи є оформлення замовлення клієнтом, тому на рисунку 2.4 зображено ДД прецеденту «Оформити замовлення».



Рисунок 2.4 – ДД основного бізнес-процесу

Діаграма активності для цього варіанту використання відображає логіку основного бізнес-процесу системи, який полягає у формуванні замовлення клієнтом. Вона дозволяє зрозуміти взаємозв'язки між діями та визначити моменти, в яких можуть виникнути виняткові ситуації.

2.4 Діаграма послідовності

Діаграма послідовності є поведінковим елементом мови UML, який використовується для моделювання динамічних взаємодій між об’єктами в часовому проміжку. Основна задача цієї діаграми – проілюструвати об’єкти, що беруть участь у процесі, та повідомлення, якими вони обмінюються, враховуючи їх послідовність [7, 8, 18].

На відміну від діаграми активності, яка акцентує увагу на потоці дій, діаграма послідовності фокусується на міжоб’єктній комунікації – хто з ким і в якому порядку взаємодіє. Головною віссю є вертикальний часовий масштаб, де кожен об’єкт представлений як «життєва лінія», а горизонтальні стрілки між ними показують передавання повідомлень або виклики методів [7, 8, 18].

Діаграми послідовності зручні для моделювання логіки складних бізнес-процесів, у яких важливо простежити послідовність звернень до системних компонентів, включно з асинхронними операціями, відкладеними відповідями або паралельною взаємодією [7, 8, 18].

У процесі проєктування вебзастосунку така діаграма дозволяє візуально відобразити етапи передачі даних від клієнта до сервера та бази даних і назад, що є особливо корисним для розуміння механізмів REST API, контролерів, сервісів і ORM-шарів [7, 8, 18].

Основним варіантом використання, що розглядається, є оформлення замовлення клієнтом. Надана діаграма (рис. 2.5) відображає взаємодію між клієнтом, клієнтською частиною, сервером, ORM та базою даних у контексті цього варіанта використання.

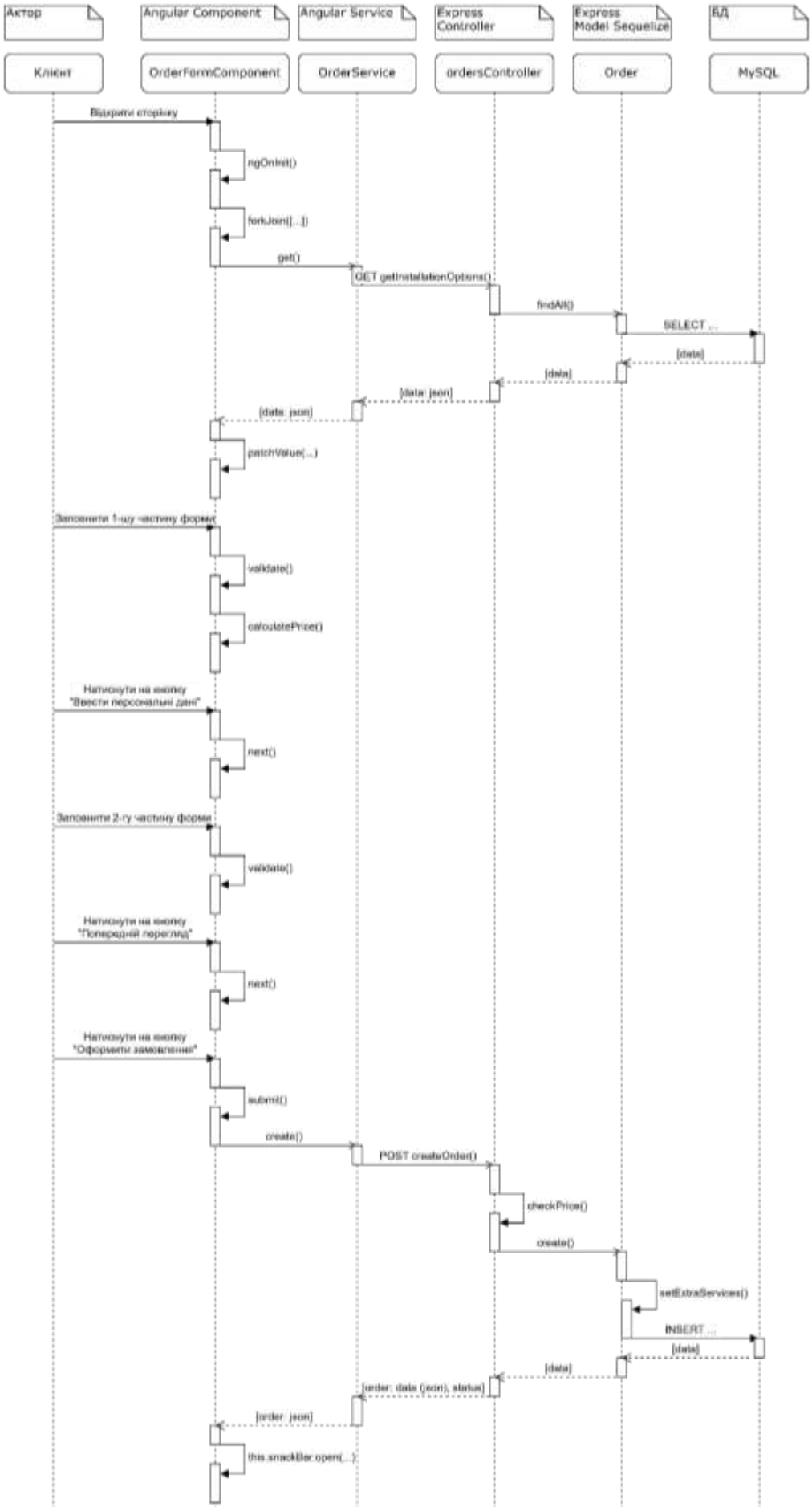


Рисунок 2.5 – ДП основного бізнес-процесу

Цей тип діаграми надає можливість побачити всю логіку обробки одного замовлення як лінійну взаємодію між компонентами. Вона показує не лише дії, що відбуваються, але і послідовність, і відповідального за кожну дію [17].

2.5 Діаграма розгортання

Діаграма розгортання відноситься до структурних діаграм UML, яка призначена для моделювання фізичного розміщення програмних компонентів системи на апаратних пристроях. Вона ілюструє, які вузли (сервери, клієнти, бази даних тощо) беруть участь у роботі системи та які програмні артефакти (модулі, служби, клієнтські додатки) розміщені на них [7, 8, 18].

Розглянута система використовує архітектуру «клієнт-сервер», яка поділяється на три основні рівні: клієнтський, серверний та рівень даних.

На рисунку 2.6 зображена діаграма розгортання системи.

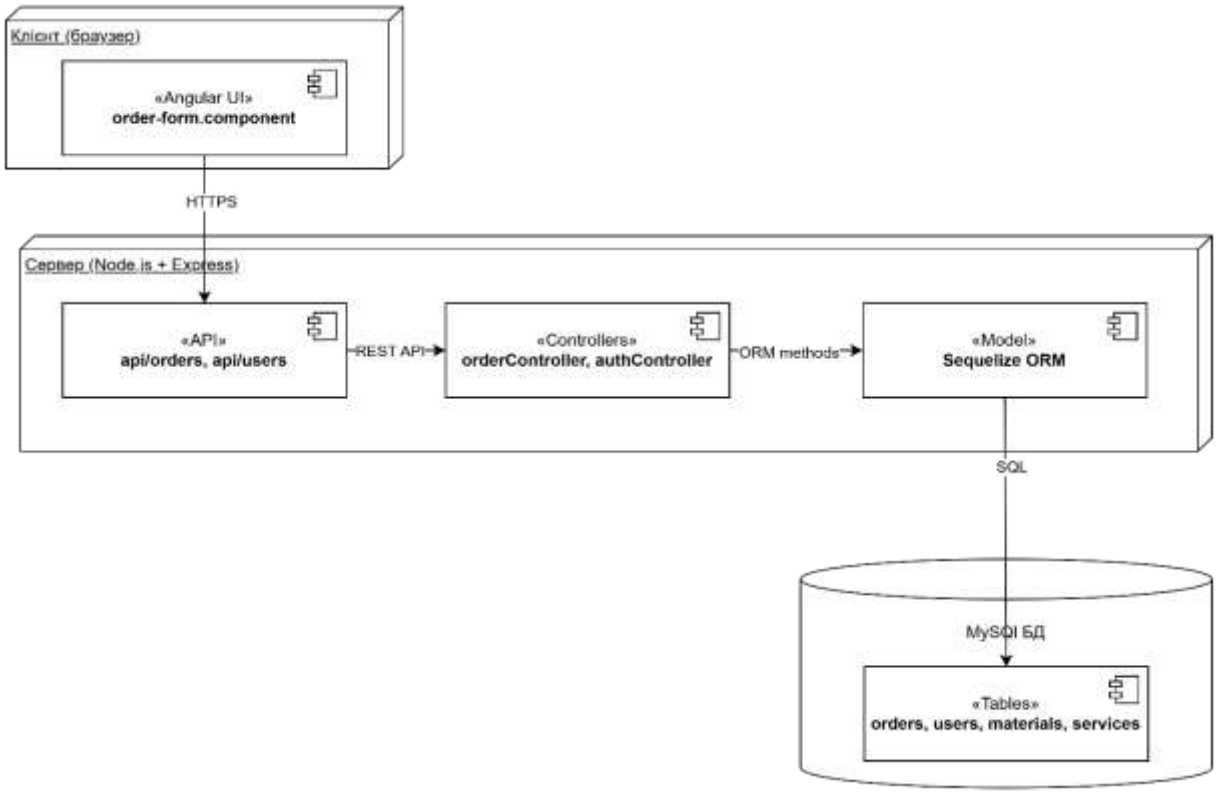


Рисунок 2.6 – Діаграма розгортання

У підсумку, в розділі описано методологію проєктування інформаційної системи для розрахунку вартості віконних конструкцій та їх монтажу. Обґрунтовано використання UML як засобу для зображення функціональності, структури та взаємодії компонентів системи. Створено діаграми варіантів використання, активності, послідовності та розгортання, що відображають логіку розміщення замовлень, ролі користувачів, етапи обробки запитів та архітектуру розгортання. Також надається детальний опис прецедентів, їх потоку подій, виключних ситуацій та призначення, що надає комплексне уявлення про функціонування системи з точки зору як клієнтів, так і користувачів (менеджерів та адміністраторів). Цей розділ забезпечує основу для розуміння подальшої реалізації функціональності на практиці.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ

3.1 Опис інструментів розробки

Для реалізації системи обрано стек, який дозволяє створювати сучасні веб-застосунки з клієнт-серверною архітектурою, гнучкістю, свисокою продуктивністю та масштабованістю.

Angular – TS-фреймворк для розробки SPA-додатків, який виділяється модульним підходом, декларативним програмуванням, реактивністю, підтримкою інкапсуляції та інтеграцією з TypeScript [1, 4].

RxJS – бібліотека для реактивного програмування. Надає можливість асинхроно оброблювати дані за допомогою Observable і операторів для трансформації потоків [5, 14].

Node.js – серверна платформа, яка дозволяє створювати масштабовані та високопродуктивні серверні додатки. Завдяки асинхронній обробці запитів вона добре підходить для створення REST API [10].

Express.js – мінімалістичний вебфреймворк для Node.js, який забезпечує маршрутизацію, обробку запитів і підключення middleware [6].

MySQL – реляційна СКБД, яка зберігає основну інформацію про замовлення, клієнтів і довідники параметрів конструкцій [9].

Angular Material – бібліотека, що забезпечує адаптивний UI згідно з принципами Material Design [2].

3.2 Основні класи системи

Основним варіантом використання системи є «Оформити замовлення», тому основними класами є ті, які забезпечують повний життєвий цикл даного прецедента.

Основними класами системи зі сторони клієнту є:

– OrderFormComponent – основний Angular-компонент, який відповідає за відображення та обробку покрокової форми оформлення замовлення;
– OrderService – Angular-сервіс, який реалізує обмін даними між клієнтом і сервером через REST API;
– ReferenceService – допоміжний Angular-сервіс, який відповідає за завантаження довідкових даних з REST API.

У свою чергу зі сторони серверу основними класами є:

– orderController.js – контролер Express.js, який приймає HTTP-запити, валідує дані та перенаправляє до відповідних сервісів;
– models/ — каталог, який містить Sequelize-моделі бази даних. Модель Order є основною, у ній описані поля замовлення, включаючи габарити конструкції, вибрані опції та контактні дані клієнта. Інші моделі каталога є допоміжними;
– referencesController.js – контролер Express.js, який надає API-endpoints для отримання всіх довідкових параметрів.

Нижче наведено функцію зі сторони серверу, яка відповідає за збереження замовлення до бази даних (рис. 3.1).



```
exports.createOrder = async (req, res) => {
    try {
        const {
            customer_name, customer_phone, user_id, width, height,
            window_type_id, material_id, glazing_id, install_id, extra_services
        } = req.body;
        const window = await WindowType.findByPk(window_type_id);
        const material = await Material.findByPk(material_id);
        const glazing = await GlazingType.findByPk(glazing_id);
        const install = await InstallationOption.findByPk(install_id);

        const baseArea = (width / 1000) * (height / 1000);
        let price = window.base_price * baseArea;
        price *= material.multiplier;
        price *= glazing.multiplier;
        price *= install.multiplier;
        const order = await Order.create({
            user_id, width, height, window_type_id, material_id, glazing_id, install_id,
            customer_name, customer_phone, total_price: price
        });
        if (Array.isArray(extra_services)) {
            await order.setExtraServices(extra_services);
        }
        res.status(201).json(order);
```

Рисунок 3.1 – Функція створення замовлення

Ознайомитися з основним компонентом можна в додатку Б, з сервісом – додаток В, а з контролером в додатку Г. З кодом всього застосунку можна ознайомитися в додатку Д.

3.3 Встановлення та налаштування оточення

Перед запуском системи необхідно підготувати та налаштувати оточення. Так як використовується архітектура типу «клієнт-сервер», то оточення необхідно розгорнути як на стороні сервера, так і на стороні клієнта.

Перш ніж почати, необхідно встановити актуальну версію Node.js та npm. Встановлення у ОС Linux Ubuntu відбувається за допомогою команд sudo apt install nodejs та sudo apt install npm [10, 11].

Після встановлення, розгорнемо клієнтську частину, для створення базової структури необхідно встановити Angular CLI за допомогою команди npm install -g @angular/cli [1], далі необхідно перейти у корінь директорії проєкту та виконати команду ng new name_project –key (рис. 3.2) [1, 4].

```
svarog@svarog:~/projects/window-calc-app$ ng new window-calc-client --no-standalone
✔ Which stylesheet format would you like to use? Sass (SCSS)     [ https://sass-lang.com/documentation/syntax#scss                ]
✔ Do you want to enable Server-Side Rendering (SSR) and Static Site Generation (SSG/Prerendering)? No
CREATE window-calc-client/README.md (1479 bytes)
```

Рисунок 3.2 – Розгортання клієнтської частини

Далі створюємо теку для серверу у корні проєкту та ініціалізуємо новий проєкт за допомогою команди npm init -y (рис. 3.3) [11].

```
svarog@svarog:~/projects/window-calc-app/window-calc-server$ npm init -y
Wrote to /home/svarog/projects/window-calc-app/window-calc-server/package.json:

{
  "name": "window-calc-server",
  "version": "1.0.0",
  "main": "index.js",
```

Рисунок 3.3 – Ініціалізація сервера

Встановлюємо необхідні для роботи додатку залежності (рис. 3.4) [11].

```
svarog@svarog:~/projects/window-calc-app/window-calc-server$ npm install express sequelize mysql2 dotenv jsonwebtoken bcryptjs

added 110 packages, and audited 111 packages in 4s

17 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

Рисунок 3.4 – Встановлення залежностей

Також варто звернути увагу на встановлення та підключення пакету CORS, так як без нього запити зі сторони клієнта будуть блокуватися (рис. 3.5) [11].

```
svarog@svarog:~/projects/window-calc-app/window-calc-server$ npm install cors

added 2 packages, and audited 139 packages in 966ms

21 packages are looking for funding
```

Рисунок 3.5 – Встановлення CORS

Структурно серверна частина відповідає архітектурному шаблону MVC з урахуванням специфіки Express (рис. 3.6) [6, 20].



Рисунок 3.6 – Структура

Далі необхідно прописати базові налаштування для коректної роботи серверу. У файлі оточення .env вказуємо параметри підключення до бази даних та порт, на якому буде запущено сервер (рис. 3.7) [6, 10].

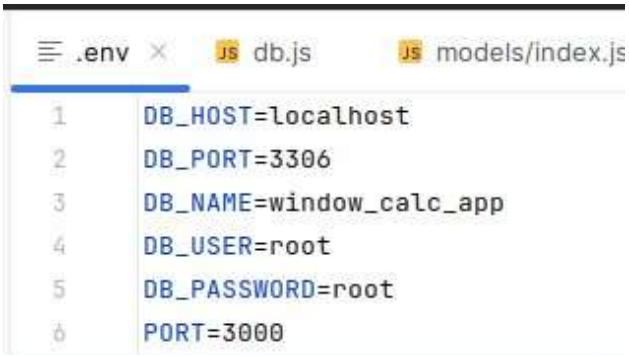


Рисунок 3.7 – Файл .env

У файлі config/db.js створюємо змінну для підключення до бази даних, використовуючи Sequelize (рис. 3.8) [6, 10].

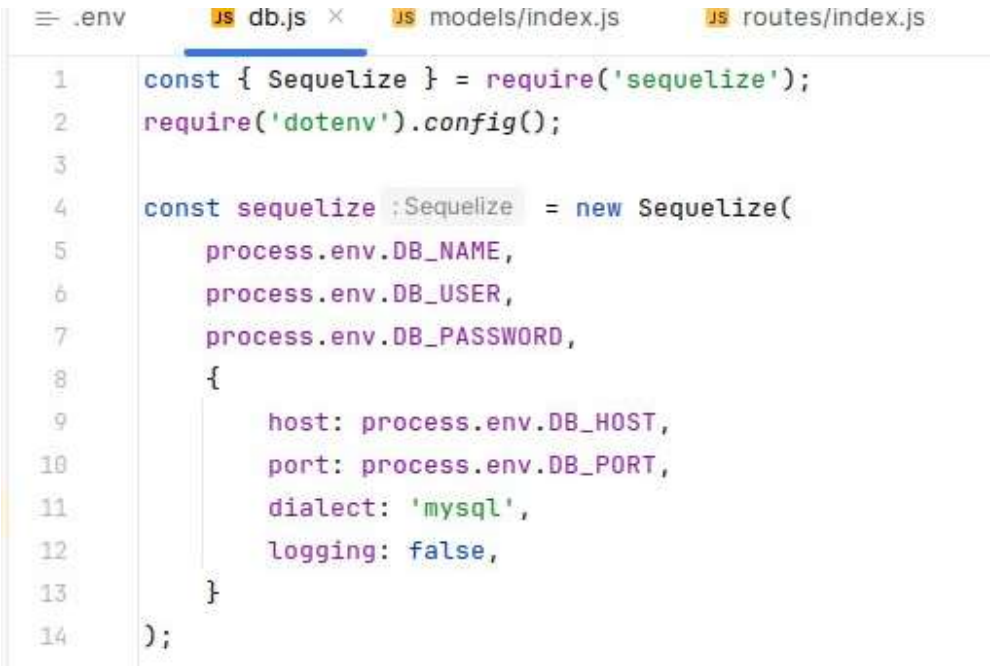


Рисунок 3.8 – Файл db.js

Файл app.js відповідає за підключення middleware, routers та інших частин проєкту, які викликаються один раз на застосунок (рис. 3.9) [6, 10].

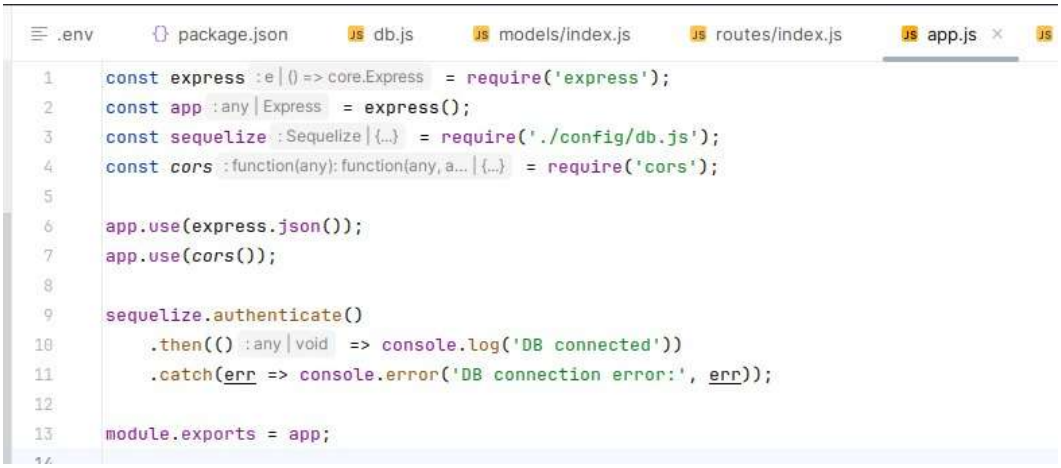


Рисунок 3.9 – Файл app.js

Головним файлом є server.js, саме він надає точку доступу та підключає app.js (рис. 3.10) [6, 10].



Рисунок 3.10 – Файл server.js

3.4 Опис API

У системі реалізовано REST API, що полегшує взаємодію між клієнтськими та серверними частинами. Усі запити до сервера обробляються через фреймворк Express.js [6], а дані зберігаються в базі даних MySQL. API сумісний з форматами JSON і використовує методи HTTP POST, GET, PUT та DELETE відповідно до CRUD- операцій [3].

Нижче наведено опис основних API-endpoints.

Запит POST /api/orders використовується для створення нового замовлення. На вхід приймає об’єкт JSON, що містить параметри конструкції, а також контактні дані користувача. У відповідь повертається підтвердження або

повідомлення про помилку.

Запит GET /api/orders повертає список усіх замовлень (тільки для користувачів (менеджер та адміністратор)).

Запити GET /api/references/window-types, /materials, /glazing-types, /installation-options, /extra-services є допоміжними та повертають довідкову інформацію, необхідну для створення форми замовлення.

Структура запитів та взаємодія з API описана за допомогою Swagger UI, що дозволяє переглядати структуру маршрутів, тестувати запити та перевіряти валідацію вхідних даних.

На рисунку 3.11 наведено приклад опису API у Swagger.

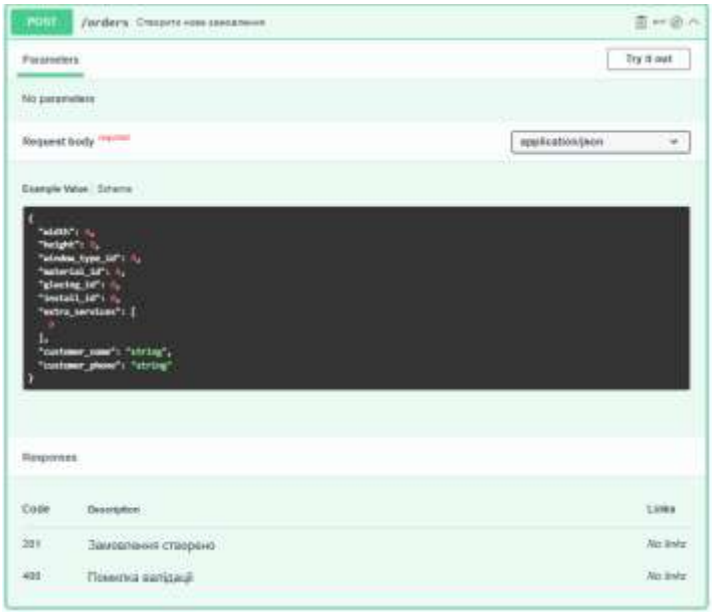


Рисунок 3.11 – Приклад опису API у Swagger

Детально можна ознайомитися за посиланням https://app.swaggerhub.com/apis/zaporizhzhianational/window-cost_estimator_api/1.0.0.

3.5 Тестування проєкту

Під час розробки системи було проведено модульне тестування, щоб переконатися в правильності основної бізнес-логіки, надійності взаємодії з API та зручності використання інтерфейсу користувача [1, 4].

Тестування проводилося за допомогою фреймворків Jasmine і Karma, які входять в Angular CLI [1]. Написано тести для перевірки валідності форми, роботи методу calculatePrice() у компоненті OrderFormComponent, а також взаємодії з OrderService.

На рисунку 3.12 наведено приклад тестування чи є валідною форма після введення даних.

```
it('форма повинна бути валідною після заповнення', () => {
  component.form.patchValue({
    width: 1000,
    height: 1500,
    window_type_id: 1,
    material_id: 1,
    glazing_id: 1,
    install_id: 1
  });

  component.contactForm.patchValue({
    customer_name: 'Тестовий Клієнт',
    customer_phone: '+380991234567'
  });

  expect(component.form.valid).toBeTrue();
  expect(component.contactForm.valid).toBeTrue();
});
```

Рисунок 3.12 – Unit-тест OrderFormComponent

Для запуску тестів використовується команда ng test, результати якої можна подивитися у браузері за шлюзом 9876 (рис. 3.13) [1, 4].



Рисунок 3.13 – Результат тестування

Значення тестування є критичним для розробки стабільних Angular-додатків. Вони дозволяють виявити як помилки у бізнес-логіці, так і проблеми у валідації, візуальному відображенні чи взаємодії з сервером [1, 4].

Детально ознайомитися з усіма тестами OrderFormComponent можна в додатку А.

3.6 Керівництво користувача

3.6.1 Відкриття застосунку та заповнення першої частини форми

Необхідно перейти за наданим посиланням на вебдодаток у браузері. Після завантаження сторінки автоматично відкривається перша частина форми.

На цьому кроцікклієнт взаємодіє з полями для введення ширини та висоти вікна, вибору типу конструкції, матеріалу профілю, склопакету, способу монтажу, а також додаткових послуг за необхідності. Праворуч від форми автоматично відображається зображення обраного типу конструкції (рис. 3.14), що допомагає візуально зрозуміти конфігурацію майбутнього виробу.

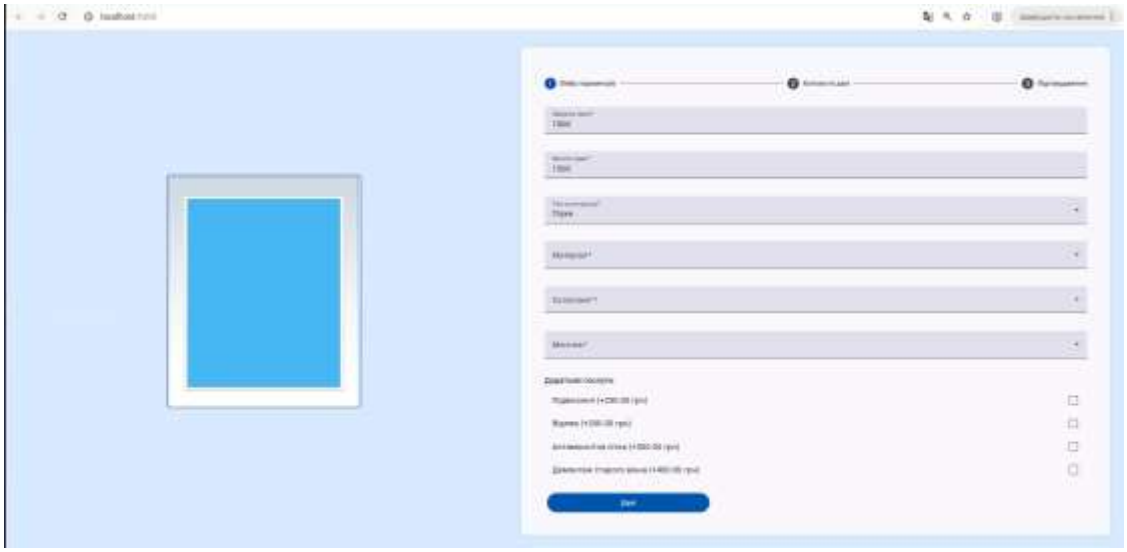


Рисунок 3.14 – Перша частина форми

Після заповнення всіх обов’язкових полів, клієнт бачить приблизну

вартість виробу, що розраховується в реальному часі залежно від обраних параметрів (рис. 3.15 – 3.16).

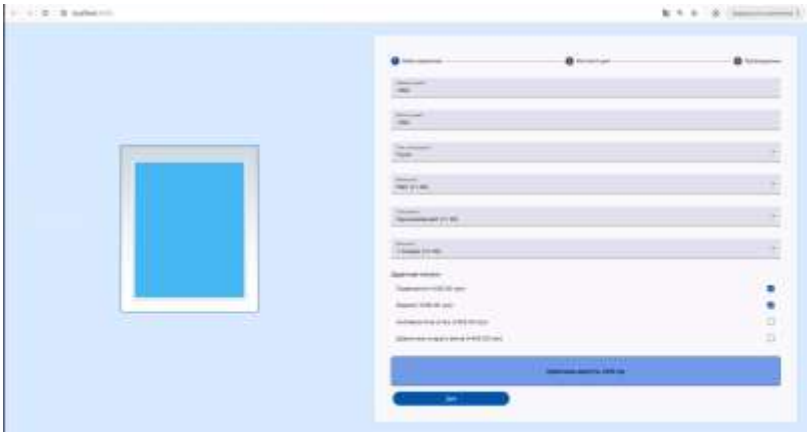


Рисунок 3.15 – Розрахунок вартості



Рисунок 3.16 – Автоматичний перерахунок при зміні параметрів

Якщо клієнта влаштовує ціна, то необхідно натиснути на кнопку «Далі».

3.6.2 Заповнення другої частини форми

Після того, як клієнт натискає на кнопку «Далі», система відображає другу частину форми.

На цьому кроці клієнт вводить персональні дані, необхідні для зв’язку з ним для узгодження деталей оплати, доставки та відповіді на можливі питання (рис. 3.17).



Рисунок 3.17 – Введення персональних даних

Натиснувши кнопку «Далі», клієнт переходить до фінального кроку.

3.6.3 Ознайомлення з деталями замовлення та оформлення

На останньому кроці, клієнт може ознайомитися з всією інформацією стосовно замовлення (параметри, персональні дані, вартість) (рис. 3.18).



Рисунок 3.18 – Перегляд деталей замовлення

Якщо клієнта все влаштовує, то для завершення процесу замовлення, необхідно натиснути на кнопку «Оформити замовлення». Система надсилає запит до серверу і якщо не відбулося виняткових ситуацій – відображає повідомлення, що замовлення прийнято, і що з клієнтом зв’яжеться менеджер.(рис. 3.19).

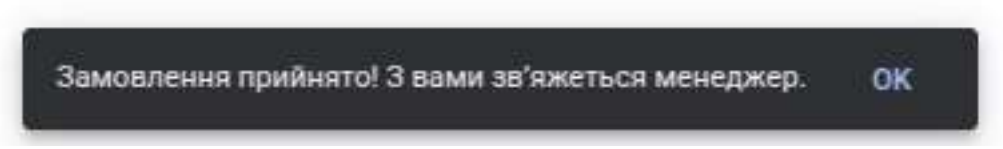


Рисунок 3.19 – Повідомлення

3.6.4 Типові помилки валідації при заповненні форми

Під час роботи з формою система автоматично перевіряє введені клієнтом значення. У випадках виявлення помилки відповідне поле підсвічується червоним кольором та з’являється текст пояснення.

Деякі поля форми мають вбудований захист від введенення невалідних даних, як наприклад числові (ширина і довжина), або вибір зі списку/чекбокси (параметри). Цей захист унеможливлює допущення деяких видів помилок, але не захищає повністю.

Першим видом помилки є незаповнення обов’язкового поля. Вона виникає, коли клієнт взаємодіє з будь-яким текстовим або числовим полем поміченим червоною зірочкою, і при цьому залишає його порожнім. Система виділяє таке поле кольором та відображає повідомлення під ним (рис. 3.20).

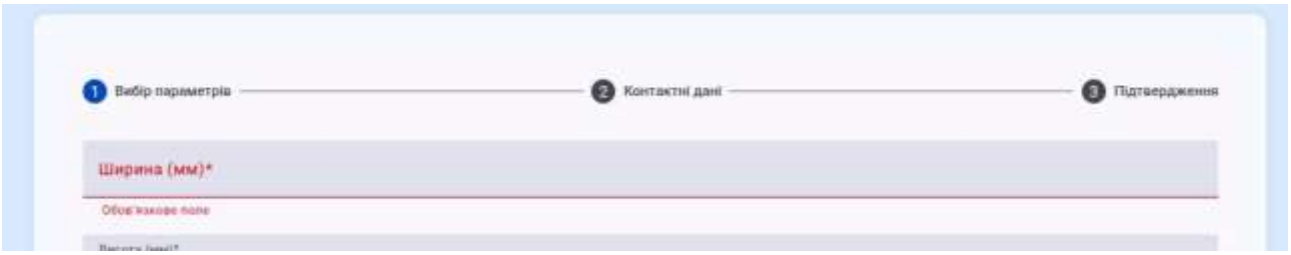


Рисунок 3.20 – Незаповнено обов’язкове поле

Другим видом помилки є введення невалідних даних у текстові поля, які присутні на 2 кроці форми. Ця помилки виникає, коли клієнт заповнює поле даними, які та не очікує прийняти. Клієнт отримує сповіщення про помилку за аналогією з першим видом помилки (рис. 3.21).



Рисунок 3.21 – Невалідні дані

ВИСНОВКИ

У ході виконання даної роботи було розроблено застосунок для розрахунку вартості віконних конструкцій та вартості роботи з монтажу. Застосунок надає можливість автоматично отримати розрахунок вартості встановлення віконних конструкцій в залежності від вибраних параметрів, а також легко та швидко оформити замовлення без необхідності авторизуватися. Дотримано архітектури «клієнт-сервер», що дозволило побудувати масштабований, гнучкий та продуктивний застосунок.

У відповідності з метою кваліфікаційної роботи було розроблено застосунок для розрахунку вартості віконних конструкцій та вартості роботи з монтажу з використанням таких технологій:

- Angular для реалізації клієнтської частини;
- Node.js та Express для реалізації серверної частини та REST API;
- MySQL для зберігання даних.

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

- сформовані функціональні та нефункціональні вимоги до застосунку, а також наведено опис предметної області та системи;
- проаналізовані та порівняні аналоги, надано висновки;
- спроєктувана та побудована архітектура застосунку (створені діаграми варіантів використання, діяльності, послідовності, розгортання; надано опис прецедентів);
- реалізовано застосунок для розрахунку вартості віконних конструкцій та вартості роботи з монтажу (надано керівництво користувача, опис API, інструкція по розгортанню та налаштуванню оточнення, описані інструменти розробки);
- протестована робота системи.

ПЕРЕЛІК ПОСИЛАНЬ

1. Angular Developer Documentation. URL: https://angular.io/docs/ (дата звернення: 21.03.2025).

2. Angular Material Guide. URL: https://material.angular.dev/guides (дата звернення: 02.04.2025).

3. Postman. API documentation. URL: https://www.postman.com/api-platform/api-documentation/ (дата звернення: 14.03.2025).

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. Express.js Guide. URL: https://expressjs.com/ (дата звернення: 14.03.2025).

7. Fowler M. UML Distilled: A Brief Guide to the Standard Object Modeling Language. Addison-Wesley Professional, 2003. 208 p.

8. Mrzyglocki F. UML Summarized: Key Concepts and Diagrams for Software Engineers, Architects, and Designers. Phillip Mrzyglocki, 2023. 59 p.

9. MySQL Documentation. URL: https://dev.mysql.com/doc/ (дата звернення: 15.02.2025).

10. Node.js Documentation. URL: https://nodejs.org/docs/latest/api/ (дата звернення: 14.03.2025).

11. npm Docs. URL: https://docs.npmjs.com/ (дата звернення: 14.03.2025).

12. npm jsonwebtoken. URL: https://www.npmjs.com/package/jsonwebtoken (дата звернення: 14.03.2025).

13. OKNA.ua. Калькулятор вікон. URL: https://okna.ua/ua/calc.php (дата звернення: 11.01.2025).

14. RxJS Documentation. URL: https://rxjs.dev/api (дата звернення: 04.03.2025).

15. VEKA.ua. Калькулятор вікон. URL: https://veka.ua/ua/calc/ (дата звернення: 11.01.2025).

16. Vikna.ua. Калькулятор вартості вікон. URL: https://vikna.ua/uk/calculator/ (дата звернення: 22.01.2025).

17. Visual Paradigm. What is Unified Modeling Language (UML)?. URL: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-uml/ (дата звернення: 07.02.2025).

18. Visual Paradigm. Use Case Diagram Guide. URL: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-use-case-diagram/ (дата звернення: 07.02.2025).

19. WDS. Калькулятор вітрових навантажень. URL: https://calc.wds.ua/ (дата звернення: 13.01.2025).

20. Understanding the MVC Pattern with Node.js. URL: https://dev.to/abhishekjaiswal_4896/understanding-the-mvc-pattern-with-nodejs-1fpg (дата звернення: 14.03.2025).

21. Лінія Вікон. Калькулятор вікон. URL: https://calc.wds.ua/ (дата звернення: 11.01.2025).

ДОДАТОК А

Unit-тест OrderFormComponent

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

    fixture = TestBed.createComponent(OrderFormComponent);
    component = fixture.componentInstance;
    fixture.detectChanges();
  });

  it('повинен створити компонент', () => {
    expect(component).toBeTruthy();
  });

  it('форма має бути невалідною за замовчуванням', () => {
    expect(component.form.valid).toBeFalse();
    expect(component.contactForm.valid).toBeFalse();
  });

  it('форма повинна бути валідною після заповнення', () => {
    component.form.patchValue({
      width: 1000,
      height: 1500,
      window_type_id: 1,
      material_id: 1,
      glazing_id: 1,
      install_id: 1
    });

    component.contactForm.patchValue({
      customer_name: 'Тестовий Клієнт',
      customer_phone: '+380991234567'
    });
```

```
    expect(component.form.valid).toBeTrue();
    expect(component.contactForm.valid).toBeTrue();
  });

  it('calculatePrice повертає null, якщо форма не повністю заповнена', () => {
    component.form.patchValue({
      width: 1000,
      height: null, // пропущено поле
      window_type_id: 1,
      material_id: 1,
      glazing_id: 1,
      install_id: 1
    });
    expect(component.calculatePrice()).toBeNull();
  });

  it('calculatePrice повертає значення, якщо всі дані є (з mock даними)', () => {
    component.windowTypes = [{ id: 1, name: 'Глухе', base_price: 1000, image_url: '/' }];
    component.materials = [{ id: 1, name: 'ПВХ', multiplier: 1 }];
    component.glazingTypes = [{ id: 1, name: 'Однокамерний', multiplier: 1 }];
    component.installationOptions = [{ id: 1, name: '1 поверх', multiplier: 1 }];
    component.extraServices = [];

    component.form.patchValue({
      width: 1000,
      height: 1000,
      window_type_id: 1,
      material_id: 1,
      glazing_id: 1,
      install_id: 1,
      extra_services: []
    });

    const price = component.calculatePrice();
    expect(price).toBeGreaterThan(0);
  });
});
```

ДОДАТОК Б

OrderFormComponent

```
export class OrderFormComponent implements OnInit {

  form!: FormGroup;
  contactForm!: FormGroup;

  windowTypes: WindowType[] = [];
  materials: Material[] = [];
  glazingTypes: GlazingType[] = [];
  extraServices: ExtraService[] = [];
  installationOptions: InstallationOption[] = [];

  selectedWindowType?: WindowType;

  constructor(
    private fb: FormBuilder,
    private http: HttpClient,
    private orderService: OrderService,
    private refService: ReferencesService,
    private snackBar: MatSnackBar
  ) {}

  ngOnInit(): void {

    this.form = this.fb.group({
      width: [1000, [Validators.required, Validators.min(300)]],
      height: [1500, [Validators.required, Validators.min(300)]],
      window_type_id: [null, Validators.required],
      material_id: [null, Validators.required],
      glazing_id: [null, Validators.required],
      install_id: [null, Validators.required],
      extra_services: [[]]
    });
```

```
    this.contactForm = this.fb.group({
      customer_name: ['', Validators.required],
      customer_phone: ['', [Validators.required, Validators.pattern('^[0-9\\-\\+\\s()]+$')]]
    });

    forkJoin({
      windowTypes: this.refService.getWindowTypes(),
      materials: this.refService.getMaterials(),
      glazingTypes: this.refService.getGlazingTypes(),
      extraServices: this.refService.getExtraServices(),
      installationOptions: this.refService.getInstallationOptions(),

    }).subscribe(data => {
      this.windowTypes = data.windowTypes;
      this.materials = data.materials;
      this.glazingTypes = data.glazingTypes;
      this.extraServices = data.extraServices;
      this.installationOptions = data.installationOptions;
      if (this.windowTypes.length > 0) {
        const defaultType = this.windowTypes.find(t => t.name.toLowerCase().includes('глух'));
        if (defaultType) {
          this.form.get('window_type_id')?.setValue(defaultType.id);
          this.onWindowTypeChange(defaultType.id);
        }
      }

    });


}


}
```

ДОДАТОК В

OrderService

```
export class OrderService {
  private api = 'http://localhost:3000/api/orders';

  constructor(private http: HttpClient) {}

  getAll(): Observable<Order[]> {
    return this.http.get<Order[]>(this.api);
  }

  getById(id: number): Observable<Order> {
    return this.http.get<Order>(`${this.api}/${id}`);
  }

  create(order: Order): Observable<Order> {
    return this.http.post<Order>(this.api, order);
  }

  update(id: number, order: Order): Observable<Order> {
    return this.http.put<Order>(`${this.api}/${id}`, order);
  }

  delete(id: number): Observable<any> {
    return this.http.delete(`${this.api}/${id}`);
  }
}
```

ДОДАТОК Г

Express OrderController

```
exports.getAllOrders = async (req, res) => {
  try {
    const orders = await Order.findAll({
      include: [WindowType, Material, GlazingType, InstallationOption, ExtraService, User]
    });
    res.json(orders);
  } catch (err) {
    res.status(500).json({ error: 'Помилка при отриманні замовлень', details: err });
  }
};

exports.createOrder = async (req, res) => {
  try {
    const {
      customer_name, customer_phone, user_id, width, height,
      window_type_id, material_id, glazing_id, install_id, extra_services
    } = req.body;
    const window = await WindowType.findByPk(window_type_id);
    const material = await Material.findByPk(material_id);
    const glazing = await GlazingType.findByPk(glazing_id);
    const install = await InstallationOption.findByPk(install_id);

    const baseArea = (width / 1000) * (height / 1000);
    let price = window.base_price * baseArea;
    price *= material.multiplier;
    price *= glazing.multiplier;
    price *= install.multiplier;
    const order = await Order.create({
      user_id, width, height, window_type_id, material_id, glazing_id, install_id,
```

```
            customer_name, customer_phone, total_price: price
        });
        if (Array.isArray(extra_services)) {
            await order.setExtraServices(extra_services);
        }
        res.status(201).json(order);
    } catch (err) {
        res.status(500).json({ error: 'Помилка при створенні замовлення', details: err });
    }
};

exports.getOrderById = async (req, res) => {
    try {
        const order = await Order.findByPk(req.params.id, {
            include: [WindowType, Material, GlazingType, InstallationOption, ExtraService, User]
        });
        if (!order) return res.status(404).json({ error: 'Замовлення не знайдено' });
        res.json(order);
    } catch (err) {
        res.status(500).json({ error: 'Помилка при отриманні замовлення', details: err });
    }
};
};
```