Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Кафедра мережних технологій факультету інформатики

ВЕБ-САЙТ ДЛЯ ПРОДАЖУ КВАРТИР ТА НЕЖИТЛОВИХ ПРИМІЩЕНЬ У НОВОБУДОВІ

Текстова частина до курсової роботи за спеціальністю «Інженерія програмного забезпечення» 121

Керівник курсової роботи
Кандидат фізико-математичних наук,
Гречко А. В.

Виконав студент
Ксьондзик М.Ю.

Київ – 2021

Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Кафедра мережних технологій факультету інформатики

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Г. І. Малашонок

(підпис)

«___» _____ 2021р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

на курсову роботу

студенту Ксьондзик М.Ю. факультету інформатики 3-го курсу

ТЕМА: Веб-сайт для продажу квартири та нежитлових приміщень у новобудові

Зміст ТЧ до курсової роботи:

- Індивідуальне завдання
- Календарний план
- Вступ
    - Аналіз предметної області. Постановка завдання курсової роботи
    - Теоретичні відомості
    - Опис реалізації програмного продукту
- Висновки
- Перелік використаних джерел
- Додатки

Дата видачі «___» _____ 2021р. Керівник _____

(підпис)

Завдання отримав _____ (підпис)

Календарний план виконання курсової роботи

Тема: Веб-сайт для продажу квартири та нежитлових приміщень у новобудові

Календарний план виконання роботи:

№ п/п	Назва етапу курсової роботи	Термін виконання етапу	Примітка
1	Отримання завдання на курсову роботу.	12.10.2020	
2	Аналіз сучасного стану питання	20.12.2021	
3	Початок написання текстової частини	01.01.2021	
4	Огляд систем аналогів	05.01.2021	
5	Дослідження технологій для веб-розробки	05.02.2021	
6	Складання плану роботи та формулювання технічний вимог до системи	15.02.2021	
7	Початок написання практичної частини	20.02.2021	
8	Надання проміжної версії текстової частини	09.05.2021	
9	Надання проміжної версії практичної частини	12.05.2021	
10	Тестування практичної частини	14.05.2021	
11	Остаточне завершення написання практичної частини	15.05.2021	
12	Підготування презентації	16.05.2021	
13	Остаточне завершення написання текстової частини	17.05.2021	
14	Захист курсової роботи	Травень 2021	

Студент Ксьондзик М. Ю.

Керівник Гречко А. В.

«____» _____

Зміст

Календарний план виконання курсової роботи **3**
Зміст **4**
Перелік термінів та умовних позначень **5**
Вступ **6**
РОЗДІЛ 1 Аналіз предметної області. Постановка завдання курсової роботи **8**
1.1 Аналіз сучасного стану питання та обґрунтування теми 8
1.2 Огляд існуючих аналогів розробки 8
1.3 Постановка завдання 12
РОЗДІЛ 2 Теоретичні відомості **13**
РОЗДІЛ 3 Опис реалізації програмного продукту **17**
3.1 Аналіз технічного завдання 17
3.2 Обґрунтування алгоритму й структури програми 18
3.3 Обґрунтування вибору засобів розробки 19
3.4 Опис розробки програми 20
3.5 Створення об’єктів і розробка головної програми 21
3.6 Опис файлів даних та інтерфейсу програми 23
3.7 Тестування програми і результати її виконання 30
Висновки **33**
Список використаних джерел **35**
Додатки **37**
Додаток А – додаткові зображення 37
Додаток Б – список ілюстрацій 40
Додаток В – програмний код 41

Перелік термінів та умовних позначень

- Стек технологій – список, набір технологій, що використовуються в сукупності для розробки.
- Скролл – гортання.
- Скроллбар – панель для гортання.
- IDE – Integrated Development Environment (середовище розробки).
- Рефакторинг – процес реструктурування коду.
- Роутер – направляє вхідний запит до певного методу.
- Футер – компонент, що розташовується у самому низу веб-сторінки.
- Рендер – зображення, зроблене за допомогою створення тривимірної моделі.
- Стилус – електронна ручка для використання на сенсорному екрані.
- Тачпад – сенсорна панель керування, що, здебільшого, розташована на ноутбуках.
- DOM – Document Object Model (інтерфейс програмування для HTML та XML документів, представляє документ у виглядів вузлів та об’єктів).

Вступ

Щороку в Україні будуються сотні новобудов, для кожної з яких необхідно розробити унікальний веб-сайт, що надавав би користувачам усю необхідну інформацію та стимулював до купівлі квартири. Сайт повинен мати гарний дизайн, високу швидкодію та інтуїтивно зрозумілий інтерфейс. На жаль, більшість сучасних веб-сторінок для продажу квартир не мають деяких з перелічених вище характеристик. Особливо часто страждає швидкодія. Як наслідок, більшість подібних систем орієнтовані, в першу чергу, на гарний вигляд веб-сторінки, зовсім ігноруючи інші аспекти, при тому, що за статистикою [1], у 40% випадків, сторінку, яка завантажується довше трьох секунд, користувачі лишають і у 80% випадків більше не повертаються на неї.

Завданням курсової роботи є створення веб-сайту для продажу квартир та нежитлових приміщень у новобудові. На сайті повинні бути розташовані сторінки «Про комплекс», «Розташування», «Контакти», «Хід будівництва» – як галерея фотографій в історичній ретроспективі, а також необхідно забезпечити можливість вибору поверху і планування квартири на загальному плані будинку.

Метою є покращення показнику швидкодії для подібних сайтів зі збереженням функціональності та забезпеченням приємного та зрозумілого інтерфейсу зі зручною навігацією веб-сторінкою.

Об’єктом дослідження є процес розробки веб-сайту для продажу квартир та нежитлових приміщень у новобудові, що мав би високу швидкодію, гарний дизайн і зручну навігацію

Предметом дослідження є застосування різноманітних технологій веб-програмування для створення сайту для продажу квартир у новобудові.

Дослідження, що були проведені у процесі роботи дозволили отримати уявлення про те, як повинен виглядати сайт для продажу квартир і нежитлових приміщень у новобудові, які мають бути структурні елементи, де краще розмістити певні компоненти. Розроблену систему можна використовувати будівельним компаніям для ознайомлення клієнтів із інформацією про новобудови або як приклад чи шаблон для подібних сайтів. Для використання у складніших проектах рекомендується замінити базу даних на реляційну, виділивши сутності та зв'язки між ними. Це забезпечить стабільнішу архітектуру. У даному випадку у нас є фактично лише одна сутність – unit, що представляє і квартири і нежитлові приміщення, вирізняючи їх типом планування. Така структура задовольняє потреби у цьому конкретному випадку. Проте у більшому та розгалуженішому проекті документо-орієнтованої бази даних із подібним підходом може бути недостатньо.

Для розробки було використано наступне програмне забезпечення:

- Середовище розробки – WebStorm
- Мова програмування – JavaScript
- Мова розмітки HTML
- Стилі CSS
- Стек технологій MERN: база даних – MongoDB, фреймворки – Express.js, React, сервер – Node.js.

Загальна структура роботи складається зі вступу, трьох розділів: аналіз предметної області та постановка завдання курсової роботи, теоретичні відомості, опис реалізації програмного продукту. А також висновків, списку використаних джерел і додатків.

РОЗДІЛ 1 Аналіз предметної області. Постановка завдання курсової роботи

1.1 Аналіз сучасного стану питання та обґрунтування теми

Щороку в Україні, зокрема у місті Києві та на його околицях, будуються сотні нових будинків та житлових комплексів, кожен із яких потребує зручного способу презентації та продажу квартир. Одним із найкращих варіантів є сайт. Існує купа різноманітних веб-сторінок розроблених для цих потреб, кожна з яких створювалася під специфічні потреби замовника. Як правило, замовники бажають представити проект якнайкраще, тому на сайтах можна бачити багато різноманітних фотографій, відео та анімацій, що мали би справити враження на потенційного покупця (див. Рисунок 1, Рисунок 3). Проте, часто, надлишковість медіа-файлів та засилля анімацій призводить до низької швидкодії та неприємного досвіду використання сайту. Даний факт не може не розчаровувати, адже на такі веб-сторінки витрачається купа часу та ресурсів, а результатом часто є повільний сайт, яким незручно користуватися та з якого хочеться піти якомога швидше. На жаль, серед сайтів новобудов вкрай рідко зустрічається збалансований продукт, що би поєднував красивий дизайн, швидкодію та інтуїтивно зрозумілий інтерфейс.

1.2 Огляд існуючих аналогів розробки

WestHouse – сайт має необхідну функціональність, зручну навігацію та зрозумілий інтерфейс. При потраплянні на основну сторінку (див. Рисунок 1) відбувається анімація та програється відео, в результаті на завантаження сторінки витрачається багато часу. Також основна сторінка надає можливість взаємодіяти з фоном за допомогою пересування курсора мишки, проте це тільки засмічує інтерфейс і не надає жодних переваг у користуванні. При скроллі мишкою

відбувається перехід на наступний елемент сторінки, проте це відбувається дуже повільно і спонукає використовувати радше скроллбар для навігації, що не дуже зручно. Основними недоліками сайту виступають: погана читабельність окремих текстів (див. Рисунок 2), завелика кількість медіа файлів, зокрема зображень, що виступають фоном для сторінки, тим самим сповільнюючи завантаження. Також серед недоліків варто зазначити можливість обрати поверх, навіть якщо на ньому всі квартири недоступні. Загалом сайт переповнений контентом і відчувається дуже повільним і важким [2].

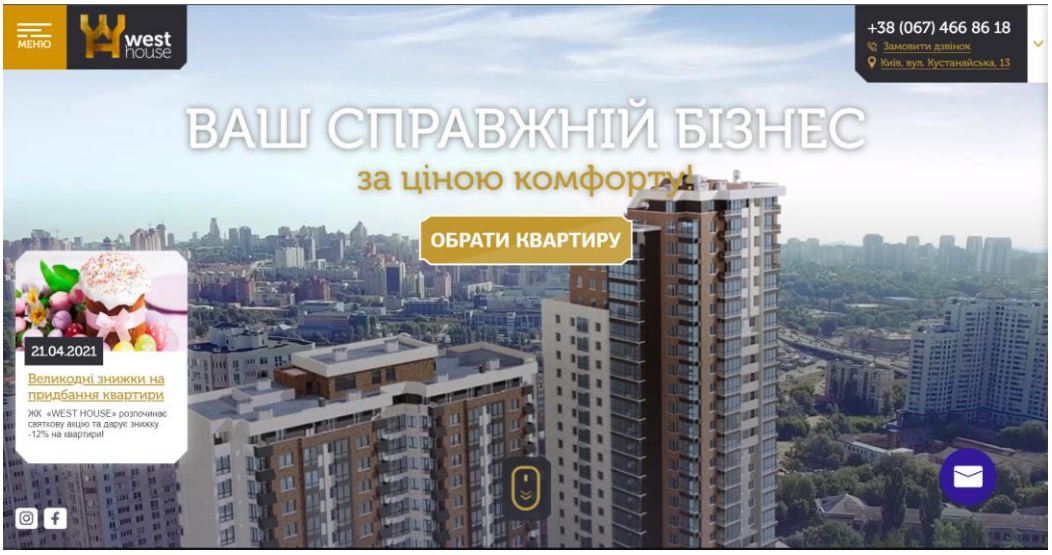

Рисунок 1 - WestHouse, головна сторінка

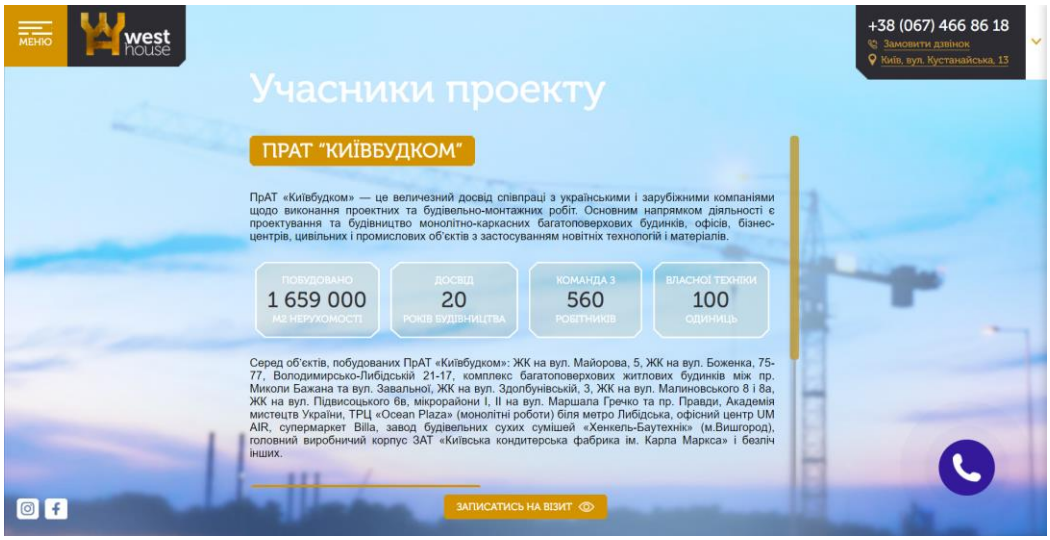

Рисунок 2 - WestHouse, сторінка учасників проекту

Taryan Towers - сайт також має необхідну функціональність, проте страждає від тих самих проблем, що і WestHouse. Головна сторінка (див. Рисунок 3) більш вдала за рахунок безперервного скроллу, анімації виконуються швидше і більш плавно. Загалом дизайн дуже гарний, інтерфейс зрозумілий. Проте увесь сайт перезавантажується при зміні розміру вікна браузеру, що є незрозумілим рішенням і дуже сильно дратує при використанні. Проте ще більше проблеми сайту проявляються при виборі квартири. Візуальний вибір поверху реалізований скроллом (див. Рисунок 4) і буває важко обрати потрібний. Через засилля різноманітною анімацією увесь процес дуже повільний і на кожному кроці боїшся натиснути не туди, адже при поверненні назад уся сторінка знову перезавантажиться [3].

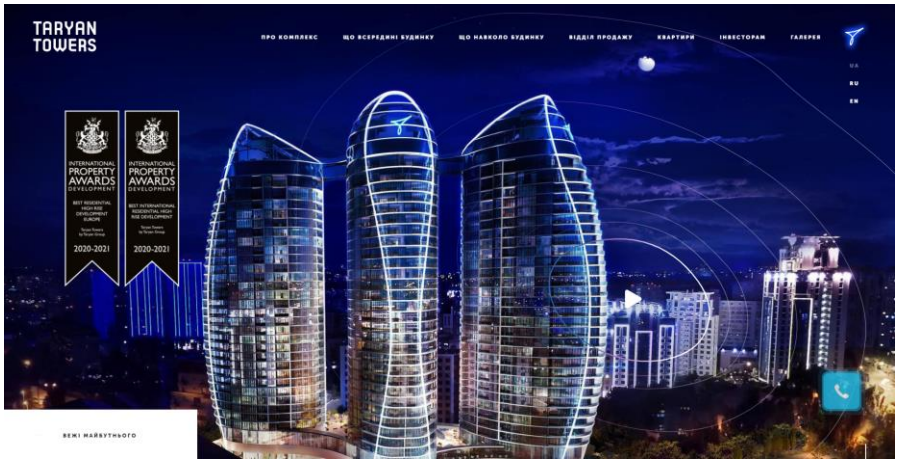

Рисунок 3 - Taryan Towers, головна сторінка

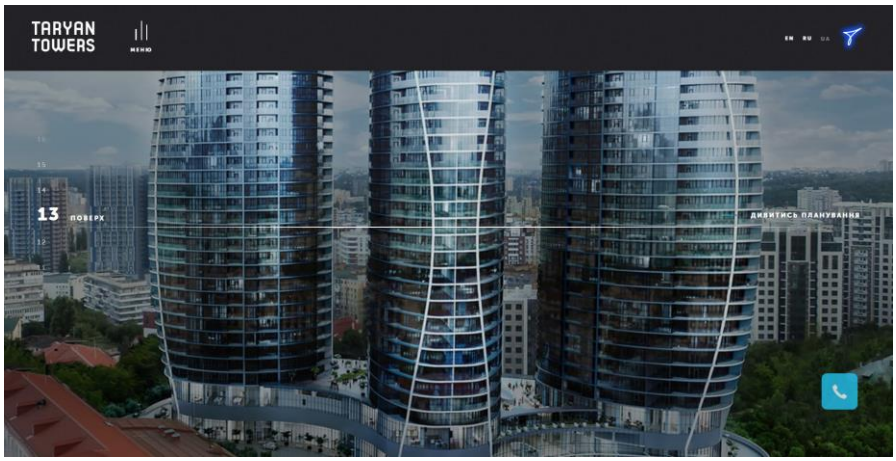

Рисунок 4 - Taryan Towers, вибір поверху

CHICAGO Central House – Сайт позитивно відрізняється швидкодією. Мінімальна кількість анімацій та прості елементи сайту забезпечують приємний досвід використання. Головна сторінка (див. Рисунок 5), все-таки, не залишилася без проблем, адже на ній розташовано багато ресурсномістких елементів, зокрема кастомний компонент Google Maps із розташуванням, що є невдалим рішенням, адже при скроллі вниз карта не встигає завантажитися і сайт ненадовго зависає. Проте основним мінусом даного сайту є дизайн (див. Рисунок 6). Яскраво червоний колір автоматично викликає бажання якомога швидше піти. Навігація незрозуміла через відсутність акцентування на важливіших елементах. Вибір квартир знаходиться десь посеред меню із рештою сторінок із непотрібними підменю [4].

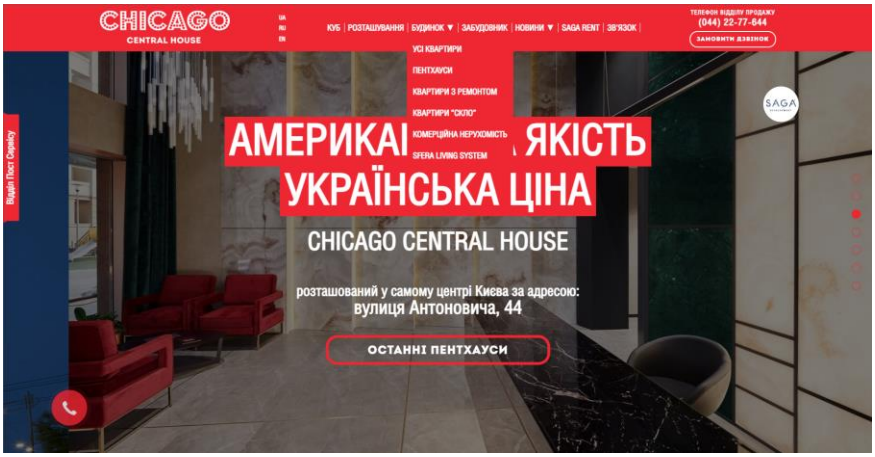

Рисунок 5 - Chicago Central House, головна сторінка

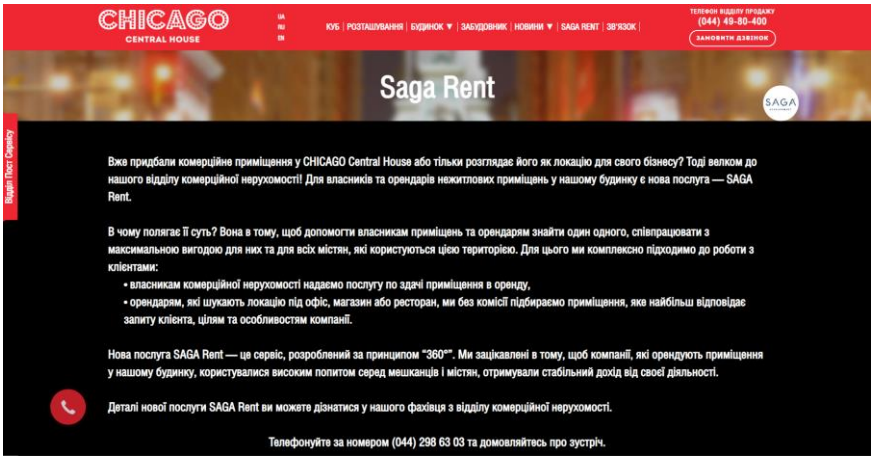

Рисунок 6 - Chicago Central House, сторінка Saga Rent

1.3 Постановка завдання

В першу чергу, необхідно дослідити предметну область задля розуміння того, які технології існують та активно використовуються, вивчити їхні особливості та сценарії використання. Також важливо проаналізувати існуючі системи-аналоги, щоб отримати загальне враження поточного стану веб-сайтів для продажу квартир та сформулювати основні вимоги до системи:

- надати сайту приємний та зручний інтерфейс зі зрозумілою навігацією та забезпечити високу швидкодію;
- провадити компоненти «Про комплекс», «Розташування», «Контакти», «Хід будівництва» – у вигляді галереї фотографій в історичній ретроспективі;
- забезпечити можливість вибору поверху і планування квартири на загальному плані будинку.

Використати знання, отримані при аналізі сайтів-аналогів, врахувати їхні плюси та мінуси, щоб в результаті отримати сайт з продажу квартир, що поєднував би все позитивне та відкидав негативний досвід систем-аналогів.

РОЗДІЛ 2 Теоретичні відомості.

Існує безліч методів та підходів для розв’язання даної задачі. Найкращим підходом буде обрати певний стек технологій, адже вони складалися на основі досвіду багатьох розробок та довели свою ефективність. Стек технологій - це набір технологій, які зазвичай використовуються для розробки програм. Зазвичай у стеці наявні три складові, а саме: база даних, рішення для back end, рішення для front end. Для Front end частини фактично використовується єдина мова – JavaScript, тому тут вибір доволі очевидний. Надалі можна обрати фреймворк. [5]

Фреймворк – це платформа для розробки програм. Наприклад, фреймворк може включати заздалегідь визначені класи та функції, які можна використовувати для різних цілей. Оскільки їх найчастіше розроблюють досвідчені програмісти, фреймворки являють собою надійну, універсальну та ефективну основу для написання програм. Це економить час та зусилля, оскільки не потрібно думати про низькорівневу функціональність і можна зосередитися на інших аспектах проекту. Одними з найпопулярніших фреймворків на сьогоднішній день є Angular, React та Vue. Вибір серед них часто зводиться до персональних вподобань, адже кожен із них майже однаково добре впорається із поставленим завданням [5,6,7].

Для Back end частини часто використовуються такі мови програмування, як JavaScript, Python, Java, Ruby та інші. Тут вибір сильно залежить від проекту. Гарним вибором буде JavaScript, адже нею пишеться і front end частина. Також для цієї мови існує багато різноманітних бібліотек та фреймворків, що можуть значно спростити розробку. Мова Python також має велику кількість бібліотек та фреймворків, хоча мінусом є те, що вона дуже сильно залежить від них. Java потребує доволі багато ресурсів для роботи і тому краще працює на дорогих системах. Ruby має плюси у вигляді мета-програмування та швидкості розробки,

проте є не дуже швидкою і має невелику кількість бібліотек у порівнянні з іншими мовами. [8]

Вибір конкретної бази даних не є дуже важливим. Головне – розуміти, потрібна реляційна чи документо-орієнтована база даних. У реляційних базах даних дані дуже структуровані, а у документо-орієнтованих – більш вільна структура. Реляційні бази даних зберігають дані у таблицях, у той час як документо-орієнтовані – у файлах або, власне, документах. Серед реляційних баз даних існують такі популярні рішення: PostgreSQL, MySQL, SQLite, Oracle Database, Microsoft SQL та багато інших. Серед документо-орієнтованих популярними варіантами для розробки виступають: MongoDB, MarkLogic, BaseX, Apache CouchDB та інші. Використовуючи MongoDB, дані будуть збережені у форматі, схожому на JSON, що зручно в контексті представлення даних, а за умови використання JavaScript для back end і front end частин проекту, дозволяє утримувати дані в одному форматі та представленні на кожному етапі розробки [5,9,10].

Наразі існує декілька популярних стеків для розробки:

MEAN – складається з бази даних MongoDB, веб-фреймворку для back end – Express.js, front end фреймворку Angular і серверу Node.js. Кожен елемент стеку використовує єдину мову – JavaScript, а також розуміє формат JSON, Даний стек дозволяє розробляти високоефективні та швидкі програми.

MERN – складається з бази даних MongoDB, веб-фреймворку для back end – Express.js, front end фреймворку React і серверу Node.js. Ідентичний до MEAN, проте використовує React як front end фреймворк.

LAMP – складається з бази даних MySQL, операційної системи Linux, мови програмування PHP та HTTP серверу Apache. Це доволі простий стек, проте

потужний стек для розробки, проте його компоненти достатньо складно пов’язати разом.

Ruby on Rails – стек, що використовує мову Ruby для back end розробки із фреймворком Rails. Для front end частини використовується HTML, CSS та JavaScript. Використовує XML або JSON для передачі даних. Недоліком даного рішення є низька швидкодія [11].

Для розробки існує багато різних середовищ розробки. Вибір конкретного залежить від обраної мови програмування, загалом стеку технологій та власних вподобань. Вибір IDE може здатися не дуже важливим, проте це насправді не так, адже це програма, в якій потрібно буде працювати над найбільшою частиною проекту і потрібно щоби цей процес був приємний, тому варто приділити особливу увагу тому, чи подобається IDE та чи підходить вона особисто Вам. Серед найпопулярніших IDE на сьогоднішній день є такі варіанти:

Visual Studio Code – вважається однією з найкращих IDE для мови JavaScript. Має вбудовану підтримку JavaScript, TypeScript та Node.js. Також є безліч розширень для інших мов, таких як Ruby, C#, Python та PHP. Visual Studio Code є досить простою і гарною IDE для людей із невеликим досвідом розробки, оскільки пояснює теги HTML, синтаксис, обробку помилок та багато іншого. Не зважаючи на величезну кількість переваг, це open-source проект, тож дана IDE абсолютно безкоштовна.

Sublime Text 3 – гнучка та безкоштовна IDE, що підтримує багато різних мов, зокрема Python, C, HTML, JavaScript та CSS. Має швидкий та простий інтерфейс. Підтримує множинні виділення, редагування у режимі розділеного екрану та автозавершення коду.

PyCharm – IDE для розробки, що підтримує багато різних мов, зокрема Python, CSS, HTML, JavaScript, Node.js. Хоча програма безкоштовна, існує платна

версія, що більш надійна та стабільна. Безкоштовна версія PyCharm може мати багато багів, особливо це стосується автозавершення коду.

IntelliJ IDEA – java-центрична IDE, що є однією з найкращих у своєму сегменті. Має можливість автоматично додавати інструменти та бібліотеки залежно від контексту. Окрім Java, це середовище розробки також підтримує HTML, CSS, JavaScript, PHP, Python, Ruby та багато інших. Загалом є безкоштовною, проте існує платна версія з покращеним функціоналом та більшою кількістю інструментів для розробки.

WebStorm – IDE, що була розроблена спеціально для створення веб-застосунків. Підтримує такі мови як HTML, JavaScript, CSS та інші. Також має підтримку різноманітних бібліотек та фреймворків, наприклад React та Angular.

PHPStorm – чудово підходить для розробки із PHP-фреймворками, такими як WordPress, Drupal, Magento та іншими. Підтримує багато front end мов, наприклад HTML, CSS, JavaScript та інші.

Загалом усі середовища розробки, створені IntelliJ, до таких відносяться: PyCharm, IntelliJ IDEA, WebStorm та PHPStorm, мають ряд спільних переваг, таких як розумне автозавершення коду, автоматичний рефакторинг, інтеграція Unit-тестів, інтеграція із системою керування версіями Git, зручна навігація та приємний дизайн [12].

РОЗДІЛ 3 Опис реалізації програмного продукту

3.1 Аналіз технічного завдання

Для розробки сайту використаємо середовище розробки WebStorm та стек технологій MERN. Дані про квартири зберігатимуться у базі даних MongoDB. Клієнтська частина з інтерфейсом буде реалізована на React, а серверна на Node.js. Для полегшення зв’язку з базою даних використаємо фреймворк Express.

Оскільки основний функціонал буде розкиданий по різних сторінках, знадобиться бібліотека React-Router, яка дозволить імітувати ці самі сторінки, таким чином буде логічно розподілена функціональність та інформація на сайті.

Для забезпечення гарного відображення використаємо стилі CSS, фреймворк React-Bootstrap та бібліотеку React-icons.

Швидкодію забезпечуватимуть React-Router та Node.js

Впровадження можливостей «Про комплекс» і «Контакти» відбудеться шляхом створення відповідних React-компонентів та заповнення їх даними.

Для впровадження можливості «Хід будівництва» буде створено React-компонент. У компонента буде вікно із фотографією, підписом та можливістю гортати(пагінацією). Для цього буде використано бібліотеку Swiper.

Щоб реалізувати вибір поверху та вибір планування квартири на плані поверху будуть створені React-компонент для будинку та React-компонент для плану поверху. Необхідно знайти зображення будинку та плану будинку. За допомогою програми Inkscape потрібно поділити зображення на відповідні зони (відповідно поверхи та квартири) та експортувати файл у форматі svg. Надалі отриманий файл відкриємо у блокноті, видалимо зайві дані та скопіюємо

елементи <path/>. Дані елементи вставимо у відповідний React-компонент за допомогою тегів svg, g, path. Кожен path буде обгорнуто у тег <Link> який забезпечуватиме перехід на потрібну сторінку при натисканні на відповідні зони. Позаду потрібно буде додати наші початкові зображення тегом <img>. Для коректного накладання зон на зображення необхідно буде помістити svg та img в один div та застосувати стилі CSS.

3.2 Обґрунтування алгоритму й структури програми

Основний алгоритм програми проілюстровано на рисунку 7. Клієнт на front end частині посилає запит на back end, де його приймає та оброблює Express і відправляє на Mongoose, який надсилає запит до бази даних MongoDB. Точно так само відбувається зворотній процес.

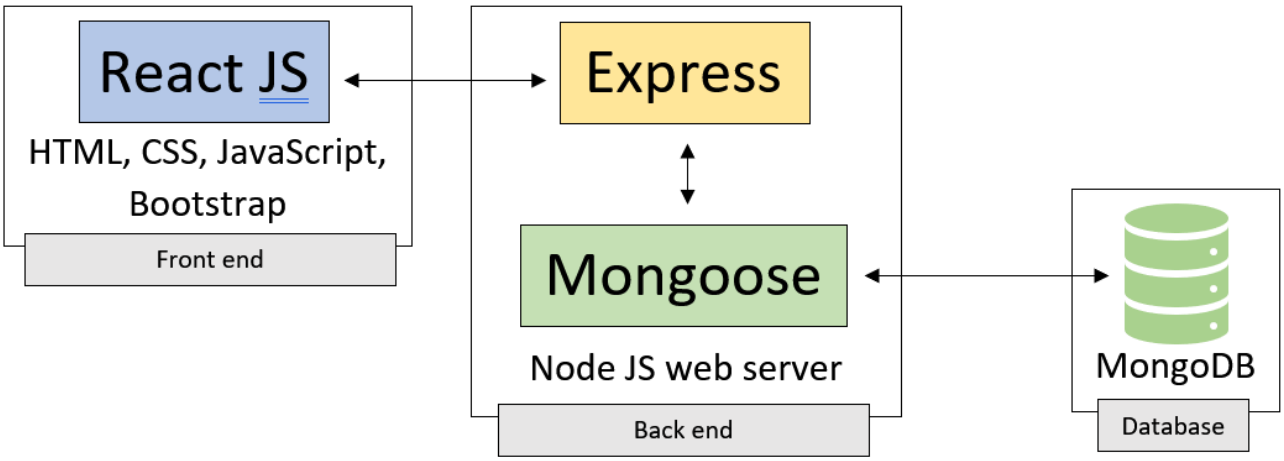

Рисунок 7 - алгоритм програми

Відповідно до даного алгоритму, програму можна розбити на декілька логічних частин: front end – React компоненти зі стилями CSS; back end – Node.js із фреймворком Express для серверу та фреймворком mongoose для керування даними, перевірки схем та переведення об’єктів із коду до їхньої репрезентації у MongoDB; база даних MongoDB.

3.3 Обґрунтування вибору засобів розробки

Для розробки було обрано мову JavaScript. Ця мова була розроблена спеціально для браузерів і є однією з найпопулярніших на даний момент для розробки веб-додатків. Оскільки вона виконується у браузері, збільшується швидкість завантаження сторінок. Це асинхронна мова, отже комунікація з сервером відбувається на задньому плані, не впливаючи на користування сайтом. Це доволі легка для вивчення мова, а завдяки її популярності, можна знайти надзвичайно велику кількість матеріалів та відповідей на будь-які питання і таким чином швидко вирішити можливі проблеми [13].

Для розробки було обрано стек-технологій MERN через ряд переваг що він надає. По-перше, кожен елемент стеку використовує єдину мову програмування – JavaScript. Це означає, що для розробки даного сайту необхідно знати лише одну мову програмування. Оскільки React генерує DOM елементи використовуючи JavaScript, не потрібно навіть використовувати шаблонізатори. По-друге, дані представлені у форматі json і в базі даних, і на серверній і клієнтській частинах – фактично всюди. Це надає можливість завжди знати в якому представленні знаходяться дані на кожному етапі. Також Node.js це дуже швидкий веб-сервер, що є безумовним плюсом у даній розробці. Менеджер пакетів npm надає можливість просто та швидко отримати бібліотеки на будь-який смак, які були створені користувачами по всьому світу. Це значно економить час розробки, адже не потрібно думати як краще реалізувати певні елементи сайту [14]. Безперечною перевагою React є Virtual DOM. Оскільки читання та запис у DOM відбувається набагато повільніше за читання та запис у об'єкт JavaScript, React використовує віртуальний DOM. Virtual DOM - JavaScript об'єкт, що виступає репрезетацією реального DOM. Таким чином React ніколи не взаємодіє напряму із реальним DOM. React робить дві копії DOMу - це і є Virtual DOM. Коли щось змінюється на сайті, React рендерить зміни у Virtual DOM, а тоді порівнює із іншою копією. Потім він виділяє компонент, що зазнав зміни, та

змінює тільки його у реальному DOM. Таким чином досягається високий показник швидкодії.

Для поточної розробки краще пасуватиме документо-орієнтована модель бази даних, у якій квартири зберігатимуться як окремі файли із певними характеристиками об’єкта (номер, поверх, кількість кімнат, тощо).

3.4 Опис розробки програми

Розробка програми поділяється на 3 етапи: розробка back end частини, заповнення бази даних, розробка front end частини.

На back end необхідно розробити модель для квартири шляхом визначення схеми зі всіма необхідними полями та обмеженнями накладеними на них. Також потрібно написати роутер для отримання інформації з бази даних за допомогою різних префіксів та методів запитів. І зрештою потрібно написати власне сервер який спілкуватиметься з клієнтом та базою даних.

Заповнити базу даних можна декількома способами, зокрема написанням файлу та імпортуванням у MongoDB або скористатися інтерфейсом керування базою даних та занести дані безпосередньо.

На front end потрібно створити html-файл сторінки, основний index js-файл та React-компоненти для кожного елементу/сторінки сайту. До кожного компоненту бажано виокремити файли для їх власних стилів задля кращої навігації кодом.

3.5 Створення об’єктів і розробка головної програми

Файлова структура проекту зображена на рисунку 8. У кореневій папці проекту знаходиться файл .eslintcache, який зберігає файли які були змінені, щоб інструмент eslint, який перевіряє код на стилістичні помилки не перевіряв кожного разу увесь проект [15], файл .gitignore, в якому вказані файли та папки які не будуть завантажуватися та контролюватися системою контролю версій Git, файл package.json, з усіма залежностями front end частини застосунку для встановлення за допомогою менеджера пакетів npm і файл package-lock.json, який зберігає конкретні версії залежностей задля уникнення небажаного їх оновлення при виконанні команди npm install, що встановлює залежності, стандартні папки build та node_modules, а також папки backend, public та src. Папка build містить у собі збудований застосунок. Папка node_modules містить модулі та бібліотеки проекту, що були встановлені за допомогою менеджера пакетів npm.

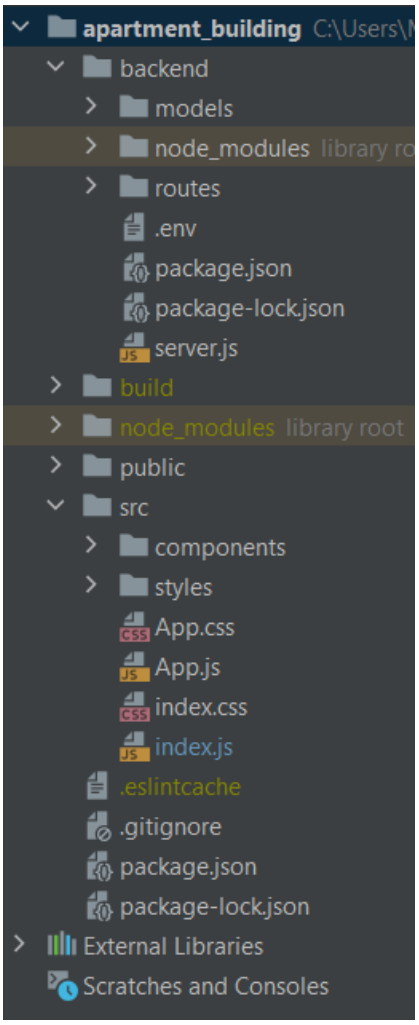

Рисунок 8 - файлова структура проекту

Папка backend представляє собою back end частину застосунку, в ній містяться моделі бази даних у папці modules, шляхи для доступу до бази даних у папці routes, .env файл, де прописане посилання для підключення до бази даних, файл server.js, що містить сервер застосунку, а також файл package.json, де прописані всі залежності back end частини проекту для встановлення за допомогою менеджера пакетів npm і файл package-lock.json, який зберігає конкретні версії залежностей щоб уникнути небажаного їх оновлення при виконанні команди npm install, що встановлює залежності.

Папка public містить іконки логотипу застосунку у різних розмірах, файл index.html, який використовується як шаблон для застосунку та файл manifest.json, що описує застосунок – у ньому зазначені назва, іконки логотипу для різних розмірів, адреса яка буде завантажена при запуску проекту, тип відображення та основні кольори.

Папка src містить папку components, у якій знаходяться всі React-компоненти (див. Рисунок 9):

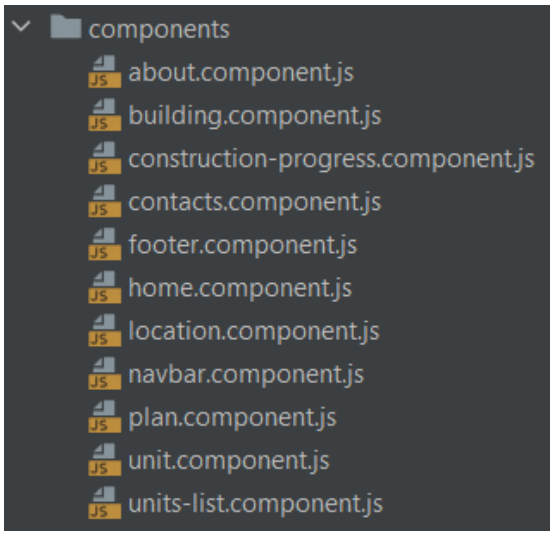

Рисунок 9 - вміст папки components

- about.component – елемент «Про нас»
- building.component – елемент зображення будинку із вибором поверху
- construction-progress.component – елемент «Хід будівництва»
- contacts.component – елемент «Контакти»
- footer.component – футер сторінки
- home.component – елемент початкової сторінки сайту, на якій розташовуються деякі інші компоненти.
- location.component – елемент «Розташування комплексу»
- navbar.component – елемент навігаційної панелі
- plan.component – елемент плану поверху із вибором приміщення
- unit.component – елемент приміщення з інформацією про нього
- unit-list.component – елемент списку усіх доступних приміщень із можливістю фільтрації

Також у папці src знаходяться: папка styles, у яку винесені стилі для кожного компоненту із відповідною назвою, файл App.css, із загальними стилями сайту, файл App.js, в який доданий компоненти Footer, а також об'єкт Router, всередині якого доданий компонент Navbar задля підтримки зміни сторінок через панель навігації та перелічені маршрути Route із атрибутами: path – адреса запиту та component – конкретний компонент, що відповідає заданій адресі. Тут прописані основні компоненти та адреси до них, що дозволяє робити імітацію зміни сторінок. Також у директорії src розташований файл index.js, що відповідає файлу index.html у папці public, в ньому розташований даний шматок коду: ReactDOM.render(<React.StrictMode> <App /> </React.StrictMode>, document.getElementById('root')); він надає вказівку щоб компонент App завантажився у html елемент з id root. Також у src розташований файл зі стилями index.css.

3.6 Опис файлів даних та інтерфейсу програми

Для розробки використовувалася документо-орієнтована база даних MongoDB, на рисунку 10 можна побачити приклад документу.

```
1   _id: 31                                                    Int32
2   planning: "apt21 "                                         String
3   floor: 3                                                   Int32
4   rooms: 0                                                   Int32
5   total_area: 636                                            Int32
6   availability: true                                         Boolean
7   price: 70000                                               Int32
8   createdAt: 2021-01-11T14:06:15.143+00:00                   Date
9   updatedAt: 2021-01-11T14:06:15.143+00:00                   Date
10  __v: 0                                                     Int32
```

Рисунок 10 - приклад документу в базі даних

- _id – стандартне поле для унікального ідентифікатора. У даному випадку використовуємо в його якості номеру приміщення, а тип даних – Int32;
- planning – тип планування приміщення: для квартир – apt[номер планування], для комор – storage, тип даних – String;

- floor – номер поверху, тип даних – Int32;
- rooms – кількість спалень для квартир або загальна кількість кімнат для комор, тип даних – Int32;
- total_area – загальна площа приміщення, тип даних – Int32;
- availability – визначає чи вільне приміщення, тип даних – Boolean;
- price – вартість приміщення, тип даних – Int32;
- createdAt – стандартне поле: дата, коли було створено запис у базі даних, тип даних – Date;
- updatedAt – стандартне поле: дата, коли було оновлено запис у базі даних, тип даних Date;
- __v – стандартне поле, що визначає версію документу, тип даних – Int32.

Для кращого візуального сприйняття можна переглянути Рисунок 11 із реляційною версією даної бази даних.

Рисунок 11 - структура бази даних у реляційному форматі

Інтерфейс програми складається із трьох основних компонентів: navbar – навігаційна панель, основна частина, де розташовані основна функціональність та інформація і footer – футер сторінки, на якому розташований компонент Contacts.

На навігаційній панелі (див. Рисунок 12) присутня назва житлового комплексу – Forward. При натисканні на цей елемент здійснюється перехід на головну сторінку. Також присутні елементи навігації для окремих сторінок: Вибір квартири – відкриває сторінку із вибором поверху на плані будинку, усі квартири – відкриває сторінку з усіма квартирами та можливістю фільтрації. Також, для швидкого доступу, на навігаційній панелі знаходяться посилання на компоненти, що розташовані на головній сторінці: про нас, розташування та хід будівництва і на контакти, що розташовані у футері сайту.

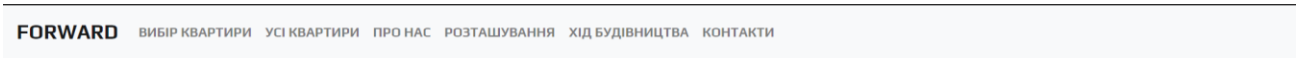

Рисунок 12 - навігаційна панель

При зменшенні ширини вікна браузеру, із навігаційної панелі усі посилання, окрім основного, із назвою житлового комплексу, переносяться у список, що випадає при натисканні на зображення трьох горизонтальних смуг. (див. Рисунок 13, Рисунок 14).

Рисунок 14 - навігаційна панель у зменшеному масштабі

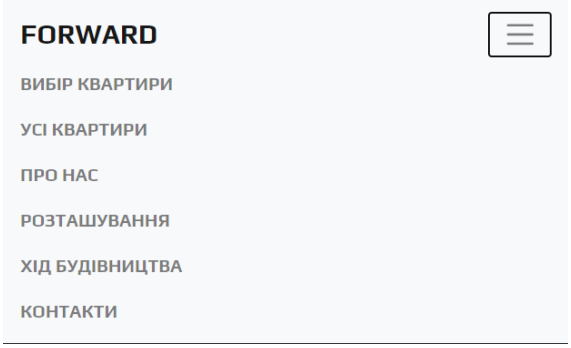

Рисунок 13 - навігаційна панель у зменшеному масштабі (розгорнута)

Елемент галерея для показу рендерів будинку показаний на Рисунок 15. Наверху зображення розташований індикатор, який візуально показує скільки зображень вже перегорнуто, та скільки залишилося в галереї. Для гортання зображень використовуються стрілочки по обидві сторони компоненту: при натисканні на ліву відкриється попереднє зображення, а при натисканні на праву – наступне, також це можна робити провівши затиснутим курсором (або пальцем чи стилусом за наявності сенсорного екрану) у бажану сторону (вліво для показу попереднього зображення та вправо для показу наступного відповідно). Якщо гортати у певну сторону вже не можна, стрілочка стає неактивною та напівпрозорою.

Рисунок 15 - галерея зображень

Компонент Хід будівництва ідентичний, проте знизу ще наявний підпис для індикація місяця та року, коли було зроблене фото. (див. Рисунок 16).

Рисунок 16 - хід будівництва

Компонент розташування (див. Рисунок 17) містить у собі адресу будинку, найближчі станції метро, приблизний час, за який можна дістатися автівкою до центра міста. Інтерактивну мапу сервісу Google Maps, де можна побачити околиці житлового комплексу із можливістю переміщення мапи за допомогою затиснення і пересування курсору. Також можна переглянути рейтинг житлового комплексу, якщо натиснути на reviews – можна подивитися відгуки. При натисненні на Directions, можна прокласти маршрут до заданого місцезнаходження. View larger map відкриє сторінку сервісу Google Maps із тою самою локацією. Для збільшення або зменшення масштабу можна використати + та – відповідно, що знаходяться у правому нижньому кутку мапи або за допомогою розведення або зведення двох пальців на тачпаді (або прямо на карті за наявності сенсорного екрану). Також є можливість перемкнути вигляд карти на супутниковий та назад за допомогою елементу у нижньому лівому кутку мапи.

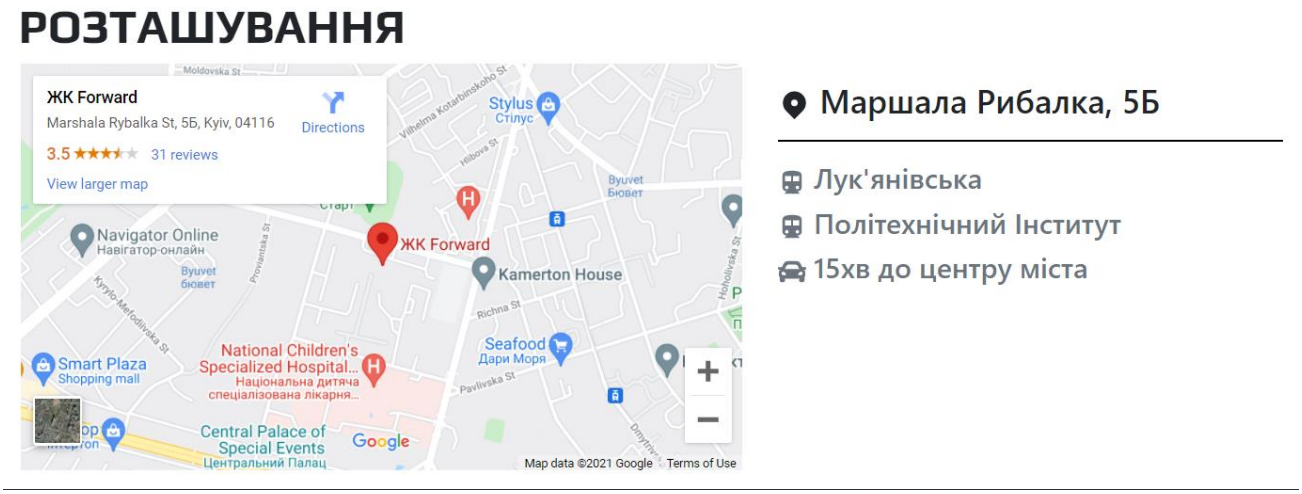

Рисунок 17 - розташування

У футері знаходиться компонент Контакти із інформацією про контактні дані (див. Рисунок 18). При натисканні на пошту, відкриється застосунок, що заданий за замовчуванням у системі, на сторінці нового листа зі вже заданою адресою отримувача із тексту посилання. При натисканні на телефонні номери відкриється стандартний застосунок системи чи браузеру для здійснення дзвінків

та буде введений номер із посилання. При натисканні на адреси відділів продажу, відкриється сервіс Google Maps із заданою адресою із посилання. При натисканні на зображення соціальної мереж, відкриється сторінка житлового комплексу у відповідній мережі. У даному проекті ці посилання просто ведуть на стандартні сторінки відповідних соціальних мереж.

Рисунок 18 - футер

На сторінці будинку, для вибору поверху необхідно обрати за допомогою курсора потрібний поверх на зображенні будинку. При наведенні, поверхи виділяються чорним кольором. Для обрання поверху потрібно натиснути на виділення. Поверхи, на яких відсутні вільні квартири виділені світло-сірим кольором та на них не можна натискати (див. Рисунок 19).

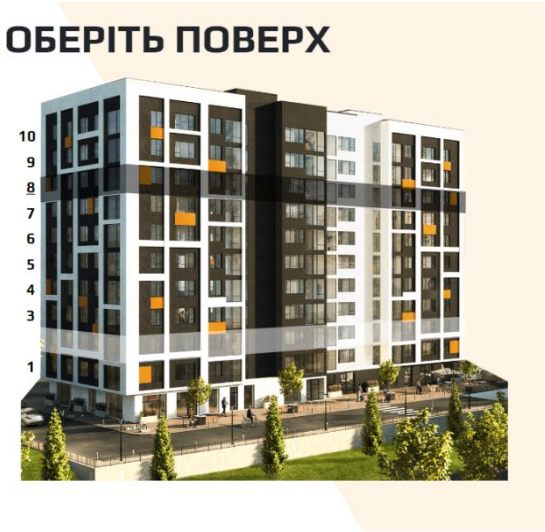

Рисунок 19 - вибір поверху на плані будинку

На сторінці поверху, для вибору приміщення необхідно обрати за допомогою курсора потрібне планування на зображенні плану поверху. При наведенні, приміщення виділяються чорним кольором. Для обрання приміщення потрібно натиснути на

Рисунок 20 - вибір приміщення на плані поверху

виділення. Недоступні приміщення виділені світло-сірим кольором та на них не можна натискати (див. Рисунок 20).

На сторінці з усіма квартирами, щоб обрати квартиру, необхідно навести курсор на картку з потрібною квартирою та натиснути. При наведенні картка трохи збільшується у розмірі. У лівому блоці розташовані фільтри для налаштування відображення. Введення ціни відбувається текстом у два поля: «Від» і «До», також при наведенні на поле з’являються стрілочки, якими можна збільшувати або зменшувати значення на 5000. Мінімальне значення встановлене на 10000, а максимальне – на 70000, що відповідає найдешевшому та найдорожчому приміщенням у будинку відповідно. Значення «Від» не може перевищувати значення «До» і значення «До» не може бути меншим за значення «Від». Галочками можна фільтрувати кімнати, поверхи, та тип приміщення (див. Рисунок 21).

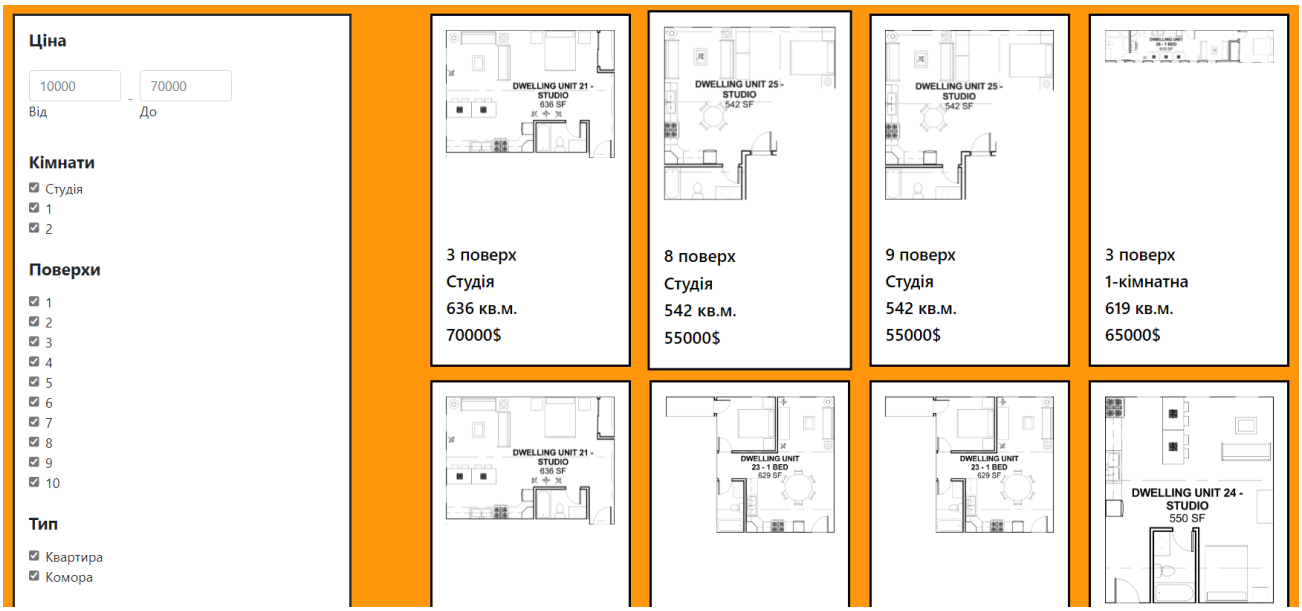

Рисунок 21 - сторінка Усі приміщення

3.7 Тестування програми і результати її виконання

Інструкція типового сценарію використання.

Заходимо на сайт (див. Рисунок 22).

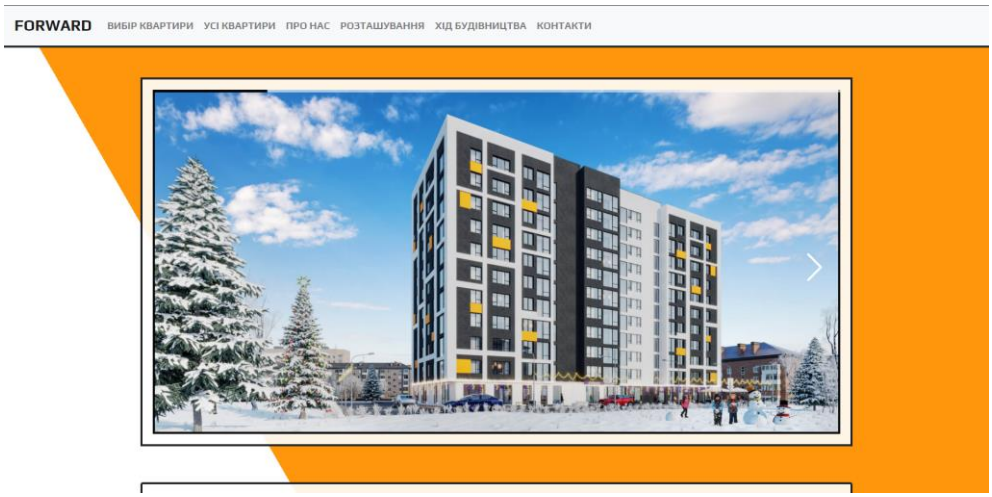

Рисунок 22 - головна сторінка

Прогортуємо декілька картинок за допомогою стрілочок (Додаток А, Рис. 1) Гортаємо сторінку вниз, читаємо інформацію у блоці «Про нас». Догортуємо до блоку «Розташування» (Додаток А, Рис. 2) та переглядаємо карту, змінюючи масштаб за допомогою кнопок + та – (Додаток А, Рис. 3) і пересуваємо карту, щоби подивитися околиці. Натискаємо на панелі навігації на «Вибір квартири» (див. Рисунок 23).

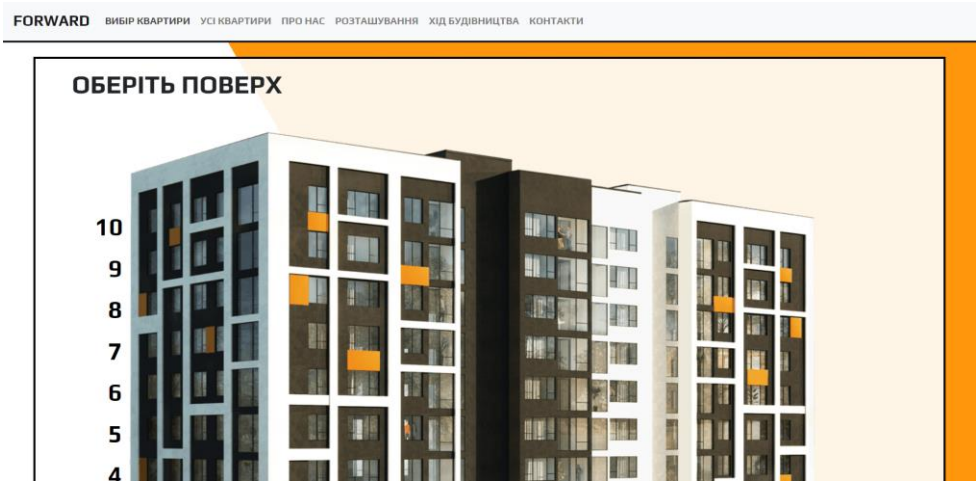

Рисунок 23 - сторінка вибору поверху

Наводимо курсор на необхідний поверх та натискаємо (Додаток А, Рис. 4). Потрапляємо на сторінку вибору приміщення (див. Рисунок 24).

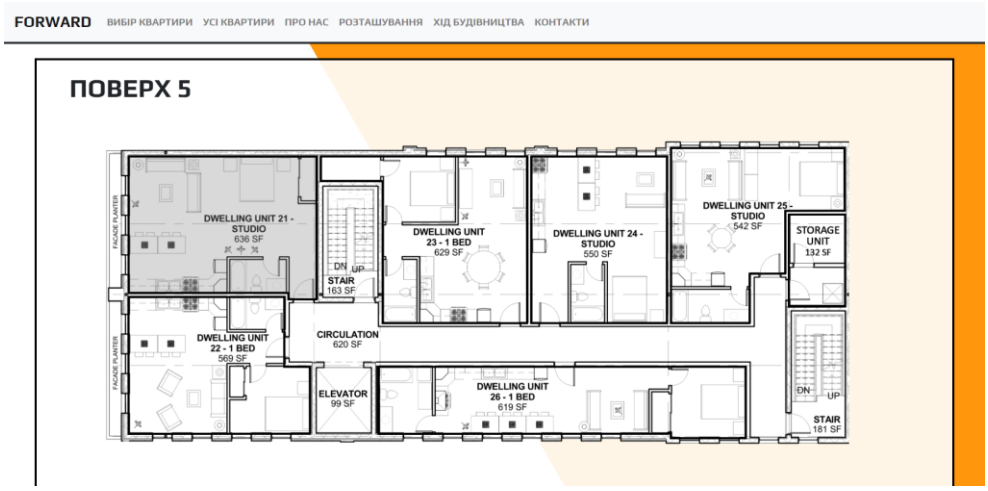

Рисунок 24 - сторінка вибору приміщення на плані поверху

Наводимо курсор на вподобане приміщення та натискаємо. (Додаток А, Рис. 5). Переглядаємо план та інформацію про приміщення (Додаток А, Рис. 6). Натискаємо на «Хід будівництва» на навігаційній панелі. (див. Рисунок 25).

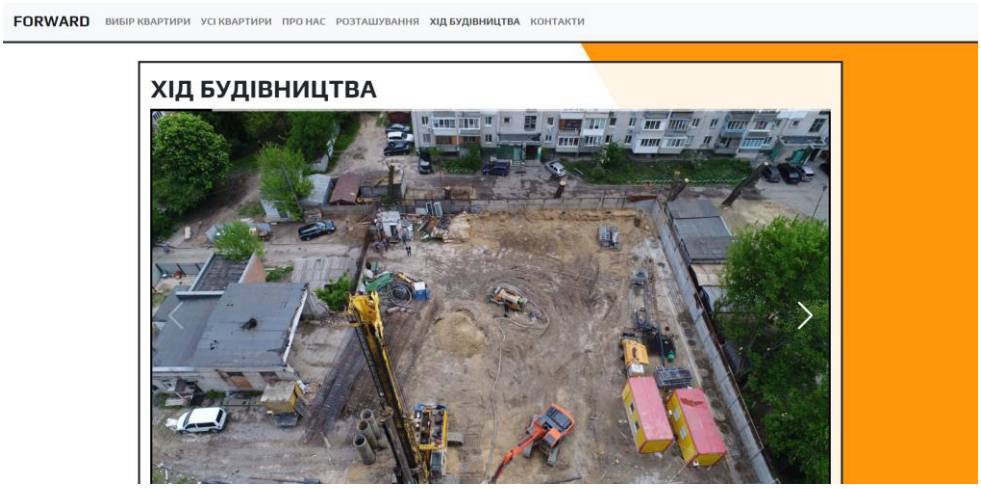

Рисунок 25 - сторінка із ходом будівництва

Переглядаємо фотографії за допомогою стрілочок (Додаток А, Рис. 1). Натискаємо на «Усі квартири» на панелі навігації. У блоці фільтрації вписуємо в поле «До» значення 60000, обираємо тип «Квартира», у розділі «Кімнати»

залишаємо відміченим тільки «Студія», у розділі «Поверхи» залишаємо відміченими тільки 7, 8, 9 та 10 (див. Рисунок 26).

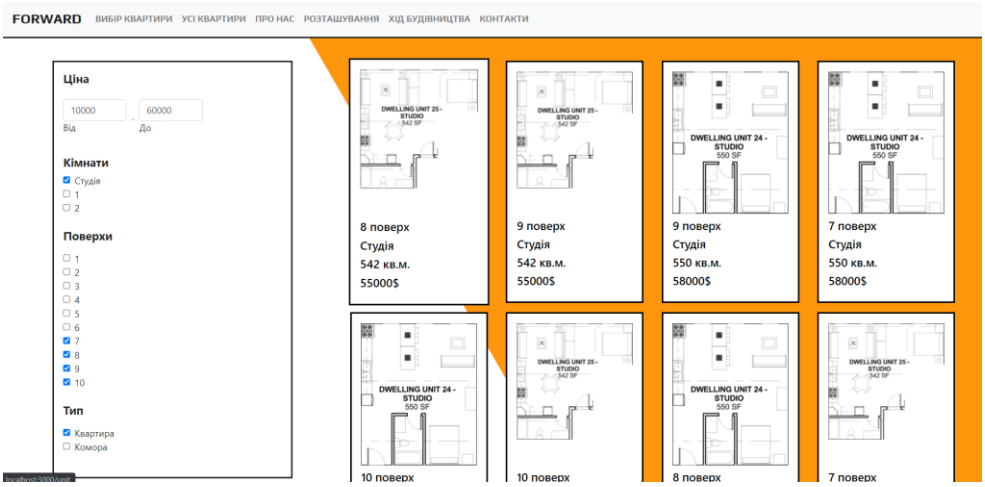

Рисунок 26 - сторінка з усіма квартирами

Натискаємо на обрану квартиру, дивимося, повертаємося назад за допомогою кнопки браузеру «Назад», обираємо іншу квартиру за даними фільтрами, дивимося і натискаємо на «Контакти» на навігаційній панелі. (Додаток А, Рис. 7).

Натискаємо на посилання на пошту, надсилаємо листа і зрештою закриваємо сайт.

Висновки

При розробці програми було:

1. Проведено аналіз сучасного стану питання та проведено огляд існуючих аналогів розробки, завдяки чому вдалося створити чітке розуміння того, які функціональні можливості має надавати розроблювана система.
2. Знайдено переваги та недоліки розроблюваних систем, що дозволило покращити різні аспекти проекту та уникнути повторювання помилок аналогічних веб-сайтів для продажу квартир.
3. Досліджено багато різних технологій для веб-розробки, у тому числі мови програмування, фреймворки, бази даних, стеки технологій та середовища розробки, завдяки чому було обрано оптимальні рішення для розроблюваного проекту згідно з визначеними вимогами до сайту.
4. Складено план роботи та сформульовані технічні вимоги до системи, відповідно до яких проводилася розробка.
5. Для оптимізації процесу активно використовувалася система контролю версій Git, на якій були створені окремі гілки для впровадження різної функціональності. Інтеграція із обраним середовищем розробки WebStorm надавала можливість одразу бачити, які файли були змінені з моменту останнього збереження, що допомогло швидко повертати зміни та спростило навігацію між задіяними у розробці на даний момент часу компонентами програми.
6. У процесі тестування, шляхом імітації користування сайтом потенційного покупця на кожному з етапів розробки, було знайдено помилки та проблеми, які було успішно усунуто. Окрім того, було виявлено певні недоліки дизайну та навігації, завдяки чому були внесені деякі зміни у

структуру сайту, чим було досягнуто кращого загального враження від користування системою та високого показнику швидкодії.

Розроблений веб-сайт має декілька сторінок, на головній розташовані компоненти: «Хід будівництва», «Контакти», «Про комплекс», «Розташування», «Хід будівництва» та «Галерея». Також містяться сторінки із вибором поверху на плані будинку, вибором приміщення на плані поверху та переліком усіх доступних приміщень із можливістю фільтрації.

Отже, розроблений веб-сайт для продажу квартир та нежитлових приміщень може бути використаний за своїм призначенням, надаючи клієнтам необхідну інформацію про певну новобудову. Сайт має високу швидкодію, зручну навігацію та приємний інтерфейс і може бути використаний як приклад або як шаблон для розробки подібних сайтів за допомогою стеку технологій MERN. За умови збільшення масштабу проекту, наприклад продажу квартир у житловому комплексі з великою кількістю будинків, пропонується використовувати реляційну базу даних задля додаткового забезпечення цілісності даних. Беручи до уваги результати, отримані у процесі роботи над проектом, можна стверджувати, що завдання виконане та кінцева мета була досягнена.

Список використаних джерел

[1] (17.03.2021) Інтернет-тенденції 2019. Статистика та факти в усьому світі https://uk.vpnmentor.com/blog/%d1%96%d0%bd%d1%82%d0%b5%d1%80%d0%bd%d0%b5%d1%82-%d1%82%d0%b5%d0%bd%d0%b4%d0%b5%d0%bd%d1%86%d1%96%d1%97-%d1%80%d0%be%d0%ba%d1%83-%d1%81%d1%82%d0%b0%d1%82%d0%b8%d1%81%d1%82%d0%b8%d0%ba%d0%b0/

[2] (11.05.2021) WestHouse https://westhouse.kiev.ua/

[3] (11.05.2021) Taryan Towers https://taryantowers.com/

[4] (11.05.2021) CHICAGO Central House https://chicago.kiev.ua/en/

[5] (09.07.2021) Web development stacks – what stacks (should) we use in 2021? https://tsh.io/blog/web-development-stacks/

[6] (15.05.2021) What is a Framework? [Definition] Types of Frameworks https://hackr.io/blog/what-is-frameworks

[7] (07.03.2013) Framework Definition https://techterms.com/definition/framework

[8] (09.05.2021) The Best Ten Backend Languages https://blog.back4app.com/best-backend-language/

[9] (09.05.2021) Top 12 NoSQL Document Databases https://www.predictiveanalyticstoday.com/top-nosql-document-databases/

[10] (09.05.2021) Best Relational Databases Software https://www.g2.com/categories/relational-databases]

[11] (16.02.2021) Top 7 Web Development Technology Stacks for 2021 https://dzone.com/articles/7-top-web-development-technology-stacks-for-2021

[12] (23.02.2021) Best Web Development IDE https://hackr.io/blog/web-development-ide

[13] (11.04.2019) Reasons Why JavaScript is Omnipresent in Modern Development https://snipcart.com/blog/why-javascript-benefits

[14] (09.02.2017) Why MERN? https://www.apress.com/gp/blog/all-blog-posts/why-mern/12056000

[15] (11.05.2021) Command Line Interface https://eslint.org/docs/user-guide/command-line-interface

Додатки

Додаток А – додаткові зображення

Рис. 1 - стрілочка галереї

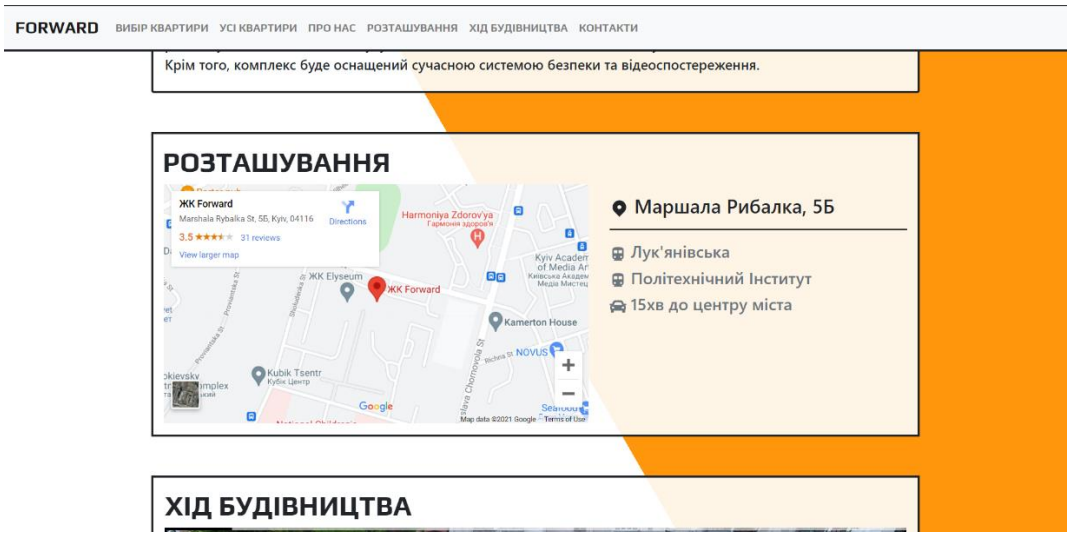

Рис. 2 - сторінка із розташуванням

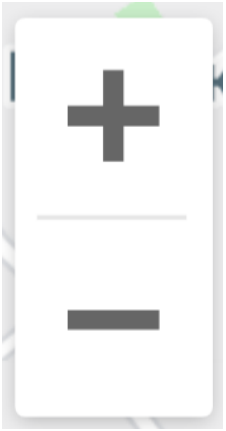

Рис. 3 - компонент збільшення та зменшення масштабу

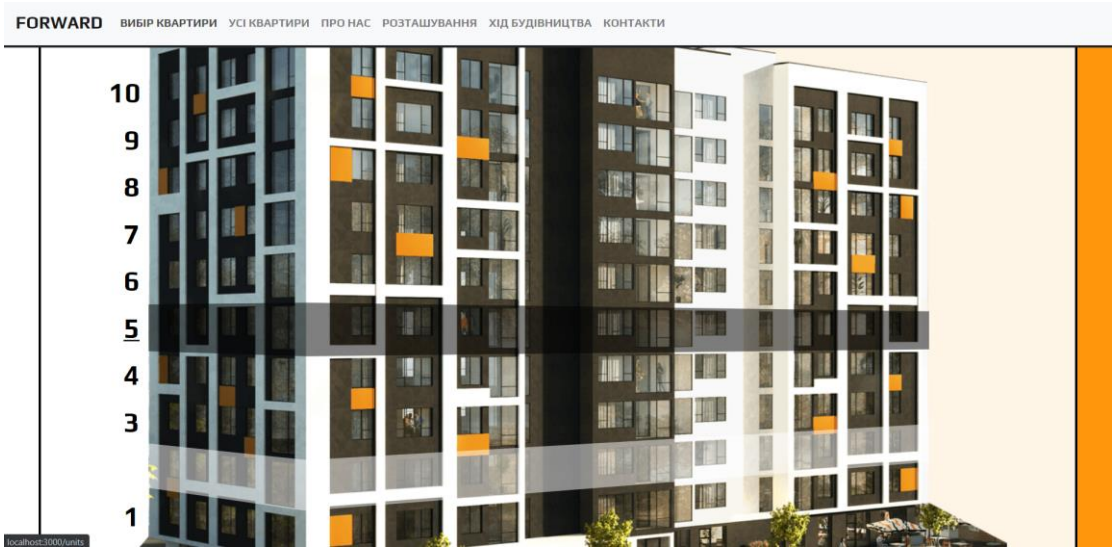

Рис. 4 - обраний поверх

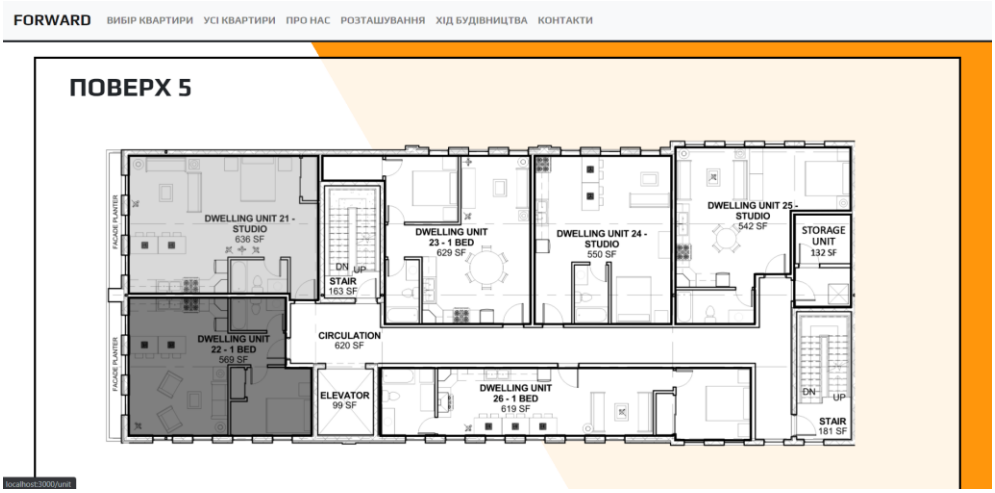

Рис. 5 - обране приміщення

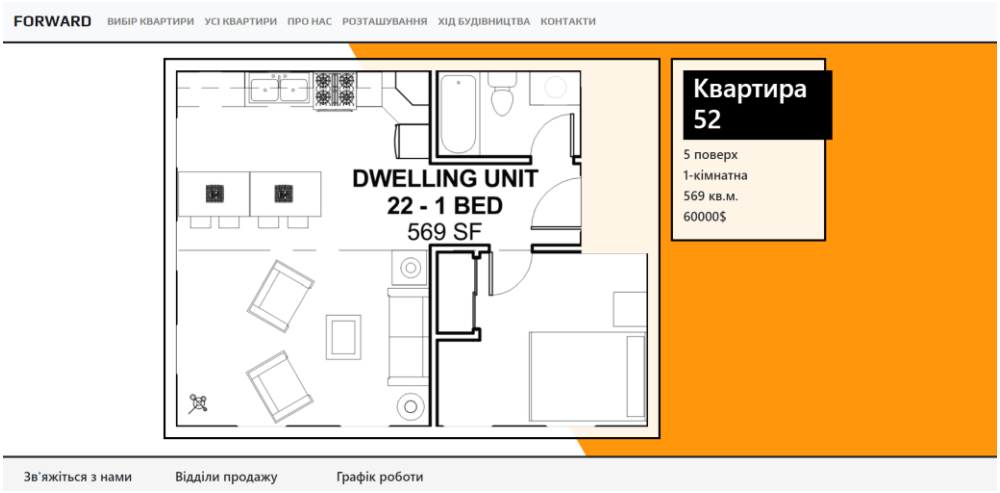

Рис. 6 - сторінка приміщення

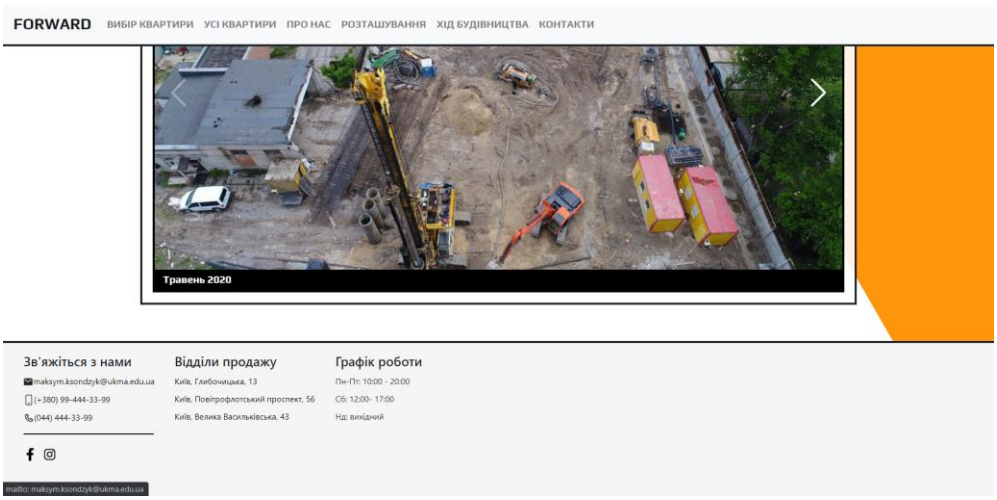

Рис. 7 - контакти у футері на сторінці

Додаток Б – список ілюстрацій

Рисунок 1 - WestHouse, головна сторінка 9
Рисунок 2 - WestHouse, сторінка учасників проекту 9
Рисунок 3 - Taryan Towers, головна сторінка 10
Рисунок 4 - Taryan Towers, вибір поверху 10
Рисунок 5 - Chicago Central House, головна сторінка 11
Рисунок 6 - Chicago Central House, сторінка Saga Rent 11
Рисунок 7 - алгоритм програми 18
Рисунок 8 - файлова структура проекту 21
Рисунок 9 - вміст папки components 22
Рисунок 10 - приклад документу в базі даних 23
Рисунок 11 - структура бази даних у реляційному форматі 24
Рисунок 12 - навігаційна панель 25
Рисунок 13 - навігаційна панель у зменшеному масштабі (розгорнута) 25
Рисунок 14 - навігаційна панель у зменшеному масштабі 25
Рисунок 15 - галерея зображень 26
Рисунок 16 - хід будівництва 26
Рисунок 17 - розташування 27
Рисунок 18 - футер 28
Рисунок 19 - вибір поверху на плані будинку 28
Рисунок 20 - вибір приміщення на плані поверху 28
Рисунок 21 - сторінка Усі приміщення 29
Рисунок 22 - головна сторінка 30
Рисунок 23 - сторінка вибору поверху 30
Рисунок 24 - сторінка вибору приміщення на плані поверху 31
Рисунок 25 - сторінка із ходом будівництва 31
Рисунок 26 - сторінка з усіма квартирами 32

Додаток В – програмний код

server.js

```
const express = require('express');
const cors = require('cors');
const mongoose = require('mongoose');

require('dotenv').config();

const app = express();
const port = process.env.PORT || 5000;

app.use(cors());
app.use(express.json());

const uri = process.env.ATLAS_URI;
mongoose.connect(uri, { useNewUrlParser: true, useCreateIndex: true }
);

mongoose.connection.once('open', () => {
  console.log("MongoDB database connection established successfully");
});

const unitsRouter = require('./routes/units');

app.use('/units', unitsRouter);

app.listen(port, () => {
  console.log(`Server is running on port: ${port}`);
});
```

units.js

```
const router = require('express').Router();
let Unit = require('../models/unit');

router.route('/').get((req, res) => {
  Unit.find()
    .then(units => res.json(units))
    .catch(err => res.status(400).json('Error: ' + err));
});

router.route('/add').post((req, res) => {
  const planning = String(req.body.planning)
  const _id = req.body._id;
  const floor = req.body.floor;
  const rooms = req.body.rooms;
  const total_area = req.body.total_area;
  const availability = req.body.availability;
  const price = req.body.price;

  const newApartment = new Unit({
    planning,
    _id,
```

```
        floor,
        rooms,
        total_area,
        availability,
        price,
    });

    newApartment.save()
        .then(() => res.json('Unit added!'))
        .catch(err => res.status(400).json('Error: ' + err));
});

router.route('/:id').get((req, res) => {
    Unit.findById(req.params.id)
        .then(apartment => res.json(apartment))
        .catch(err => res.status(400).json('Error: ' + err));
});

router.route('/:id').delete((req, res) => {
    Unit.findByIdAndDelete(req.params.id)
        .then(() => res.json('Unit deleted.'))
        .catch(err => res.status(400).json('Error: ' + err));
});

router.route('/update/:id').post((req, res) => {
    Unit.findById(req.params.id)
        .then(unit => {
            unit.planning = String(req.body.planning)
            unit._id = Number(req.body._id);
            unit.floor = Number(req.body.floor);
            unit.rooms = Number(req.body.rooms);
            unit.total_area = Number(req.body.total_area);
            unit.availability = Boolean(req.body.availability);
            unit.price = Number(req.body.price);

            unit.save()
                .then(() => res.json('Unit updated!'))
                .catch(err => res.status(400).json('Error: ' + err));
        })
        .catch(err => res.status(400).json('Error: ' + err));
})

module.exports = router;
```

unit.js

```
const mongoose = require('mongoose');

const Schema = mongoose.Schema;

const unitSchema = new Schema({
    planning:{type: String, required: true},
    _id: {
        type: Number,
        required: true,
```

```
      unique: true,
      trim: true,
    },
    floor: { type: Number, required: true },
    rooms: { type: Number, required: true },
    total_area: { type: Number, required: true },
    availability: { type: Boolean, required: true },
    price: { type: Number, required: true },
}, {
    timestamps: true,
});

const Unit = mongoose.model('Unit', unitSchema);

module.exports = Unit;
```

index.js

```
import React from 'react';
import ReactDOM from 'react-dom';
import './index.css';
import App from './App';

ReactDOM.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>,
  document.getElementById('root')
);
```

App.js

```
import React from 'react'
import { BrowserRouter as Router, Route} from "react-router-dom";
import "jquery"
import "bootstrap/dist/css/bootstrap.min.css"
import "./styles/app.styles.css"

import Navbar from "./components/navbar.component"
import Footer from "./components/footer.component";
import Plan from "./components/plan.component";
import Unit from "./components/unit.component";
import Building from "./components/building.component";
import UnitsList from "./components/units-list.component";
import Home from "./components/home.component";

function App() {
  return (
     <div className="main-window">
        <div className={"content"}>
          <Router>
             <Navbar />
             <br/>
             <Route path="/" exact component = {Building}/>
             <Route path="/units" exact component={Plan} />
```

```
            <Route path="/unit" exact component={Unit} />
            <Route path="/building" exact component={Building} />
            <Route path="/units-list" exact component={UnitsList} />
            <Route path="/home" exact component={Home} />
          </Router>
        </div>
        <div className={"footer-content"}>
          <Footer />
        </div>
      </div>

  );
}

export default App;
```

home.component.js

```
import React, {Component} from 'react';
import "../styles/about.styles.css"
import About from "./about.component";
import BuildingLocation from "./location.component";
import ConstructionProgress from "./construction-progress.component";
import {Swiper, SwiperSlide} from 'swiper/react';

import 'swiper/swiper.min.css';
import "swiper/components/pagination/pagination.min.css"
import "swiper/components/navigation/navigation.min.css"
import SwiperCore, {
  Pagination, Navigation
} from 'swiper/core';

SwiperCore.use([Pagination, Navigation]);

export default class Home extends Component {
  render() {
    return (
      <div className="container" style={{marginTop: "4em"}}>
        <br/>
        <div className="container gallery-container">
          <Swiper pagination={{
            "type": "progressbar"
          }} navigation={true} className="mySwiper">
            <SwiperSlide>
              <img className={"gallery-image"}
                src="https://forward.kyiv.ua/wp-content/uploads/2020/12/forward_007-crop.jpg"
                alt={"May 2020"}/>
            </SwiperSlide>
            <SwiperSlide>
              <img className={"gallery-image"}
                src="https://forward.kyiv.ua/wp-content/uploads/2020/12/forward_008-crop.jpg"
                alt={"June 2020"}/>
            </SwiperSlide>
            <SwiperSlide>
              <img className={"gallery-image"}
```

```
                        src="https://forward.kyiv.ua/wp-content/uploads/2020/01/gallery-img-1.jpg"
                        alt={"July 2020"}/>
                </SwiperSlide>
                <SwiperSlide>
                    <img className={"gallery-image"}
                        src="https://forward.kyiv.ua/wp-content/uploads/2020/01/gallery-img-2.jpg"
                        alt={"August 2020"}/>
                </SwiperSlide>
                <SwiperSlide>
                    <img className={"gallery-image"}
                        src="https://forward.kyiv.ua/wp-content/uploads/2020/02/img-13.jpg"
                        alt={"September 2020"}/>
                </SwiperSlide>
                <SwiperSlide>
                    <img className={"gallery-image"}
                        src="https://forward.kyiv.ua/wp-content/uploads/2020/02/img-12.jpg"
                        alt={"October 2020"}/>
                </SwiperSlide>
            </Swiper>
        </div>
        <br/>
        <About/>
        <br/>
        <BuildingLocation/>
        <br/>
        <ConstructionProgress/>
        <br/>
    </div>
  );
 }
}
```

navbar.component.js

```
import React, {Component} from 'react';
import {HashLink as Link} from 'react-router-hash-link';
import "bootstrap/js/src/collapse.js";
import "../styles/navbar.styles.css"

export default class Navbar extends Component {

 render() {
  return (
   <nav className="navigation navbar navbar-expand-lg navbar-light bg-light">
    <a className="navbar-brand" href="/home">FORWARD</a>
    <button className="navbar-toggler" type="button" data-toggle="collapse"
        data-target="#navbarSupportedContent" aria-controls="navbarSupportedContent"
        aria-expanded="false" aria-label="Toggle navigation">
     <span className="navbar-toggler-icon"/>
    </button>

    <div className="collapse navbar-collapse" id="navbarSupportedContent">
     <ul className="navbar-nav mr-auto">
      <li className="nav-item">
       <Link to="/building" className="nav-link">ВИБІР КВАРТИРИ</Link>
```

```
                    </li>
                    <li className="nav-item">
                        <Link to="/units-list" className="nav-link">УСІ КВАРТИРИ</Link>
                    </li>
                    <li className="nav-item">
                        <Link to="/home#about" className="nav-link">ПРО НАС</Link>
                    </li>
                    <li className="nav-item">
                        <Link to="/home#location" className="nav-link">РОЗТАШУВАННЯ</Link>
                    </li>
                    <li className="nav-item">
                        <Link to="/home#construction-progress" className="nav-link">ХІД БУДІВНИЦТВА</Link>
                    </li>
                    <li className="nav-item">
                        <Link to="/home#contacts" className="nav-link">КОНТАКТИ</Link>
                    </li>
                </ul>
            </div>
        </nav>
    );
  }
}
```

footer.component.js

```
import React, { Component } from 'react';

import "bootstrap/js/src/collapse.js";
import "../styles/footer.styles.css"
import Contacts from "./contacts.component";

export default class Footer extends Component {
  render() {
    return (
        <footer className="footer">
            <Contacts/>
        </footer>
    );
  }
}
```

building.component.js

```
import React from 'react';

import "../styles/building.styles.css"
import {Link} from "react-router-dom";
import axios from "axios";

class Building extends React.Component {

  componentDidMount() {
    window.scrollTo(0, 0);
    axios.get('http://localhost:5000/units/')
        .then(response => {
```

```
            this.setState({ units: response.data })
          })
          .catch((error) => {
            console.log(error);
          })

    }

    getUnits(floor){
        return this.state.units.filter(currentUnit =>
            currentUnit.floor === floor && currentUnit.availability === true
        )
    }

    render() {
        return(
            <div className={"building-main-container"}>
                <div className={"pageLabel"} style={{marginLeft: "4%"}}>
                    <h1>ОБЕРІТЬ ПОВЕРХ</h1>
                </div>
            <div className={"building-container"}>
                <img className="building-plan-img " src={"https://i.ibb.co/KsgvPHt/building.png"} alt={"Building
plan"}/>
                <svg viewBox="-19 0 370 280" className={"building-svg"}>
                    <g transform="scale(0.292)">
                        {this.state && this.getUnits(2).length ?
                            <Link to={
                                {
                                    pathname: '/units',
                                    state: this.state ? this.getUnits(2) : null
                                }
                            }>
                                <path
                                    d="m46.28 696.57 199.58 29.09 796.10 -62.27 0.35 -48.47L247.84 660.74 46.44 639.80Z"
                                    data-code="floor2"
                                    className="floor"
                                />
                                <text x="14" y="685" fontSize={35} fill="black">2</text>
                            </Link>
                            : <path
                                d="m46.28 696.57 199.58 29.09 796.10 -62.27 0.35 -48.47L247.84 660.74 46.44 639.80Z"
                                data-code="floor2"
                                className="floor_taken"
                            />
                        }

                        {this.state && this.getUnits(3).length ?
                            <Link to={
                                {
                                    pathname: '/units',
                                    state: this.state ? this.getUnits(3) : null
                                }
                            }>
                                <path
                                    d="m46.44 639.80 201.40 20.93 794.47 -45.82 -0.03 -46.06 -794.97 25.81 -201.19 -10.61z"
                                    data-code="floor3"
```

```
                    className="floor"
                  />
                  <text x="14" y="625" fontSize={35} fill="black">3</text>
                </Link>
                : <path
                  d="m46.44 639.80 201.40 20.93 794.47 -45.82 -0.03 -46.06 -794.97 25.81 -201.19 -10.61z"
                  data-code="floor3"
                  className="floor_taken"
                />
              }

              {this.state && this.getUnits(4).length ?
                <Link to={
                  {
                    pathname: '/units',
                    state: this.state ? this.getUnits(4) : null
                  }
                }>
                  <path
                    d="m46.12 584.06 201.19 10.61 794.97 -25.81 -0.05 -49.23 -793.77 8.00 -202.63 -5.55z"
                    data-code="floor4"
                    className="floor"
                  />
                  <text x="14" y="565" fontSize={35} fill="black">4</text>
                </Link>
                :  <path
                  d="m46.12 584.06 201.19 10.61 794.97 -25.81 -0.05 -49.23 -793.77 8.00 -202.63 -5.55z"
                  data-code="floor4"
                  className="floor_taken"
                />
              }

              {this.state && this.getUnits(5).length ?
                <Link to={
                  {
                    pathname: '/units',
                    state: this.state ? this.getUnits(5) : null
                  }
                }>
                  <path
                    d="m45.83 522.08 201.91 6.52 794.49 -8.96 -0.05 -48.47 -794.97 -11.04 -201.43 2.35z"
                    data-code="floor5"
                    className="floor"
                  />
                  <text x="14" y="505" fontSize={35} fill="black">5</text>
                </Link>
                : <path
                  d="m45.83 522.08 201.91 6.52 794.49 -8.96 -0.05 -48.47 -794.97 -11.04 -201.43 2.35z"
                  data-code="floor5"
                  className="floor_taken"
                />
              }

              {this.state && this.getUnits(6).length ?
                <Link to={
                  {
```

```
                pathname: '/units',
                state: this.state ? this.getUnits(6) : null
              }
            }>
              <path
                d="m45.78 462.48 202.59 -1.99 793.81 10.68 -0.22 -49.06 -794.63 -29.33 -200.92 13.50z"
                data-code="floor6"
                className="floor"
              />
              <text x="14" y="445" fontSize={35} fill="black">6</text>
            </Link>
            : <path
              d="m45.78 462.48 202.59 -1.99 793.81 10.68 -0.22 -49.06 -794.63 -29.33 -200.92 13.50z"
              data-code="floor6"
              className="floor_taken"
            />
          }

          {this.state && this.getUnits(7).length ?
            <Link to={
              {
                pathname: '/units',
                state: this.state ? this.getUnits(7) : null
              }
            }>
              <path
                d="m46.41 406.28 200.92 -13.50 794.63 29.33 0.63 -49.06 -794.63 -47.37 -200.92 22.34z"
                data-code="floor7"
                className="floor"
              />
              <text x="14" y="385" fontSize={35} fill="black">7</text>
            </Link>
            : <path
              d="m46.41 406.28 200.92 -13.50 794.63 29.33 0.63 -49.06 -794.63 -47.37 -200.92 22.34z"
              data-code="floor7"
              className="floor_taken"
            />
          }
          {this.state && this.getUnits(8).length ?
            <Link to={
              {
                pathname: '/units',
                state: this.state ? this.getUnits(8) : null
              }
            }>
              <path
                d="m47.03 348.03 200.92 -22.34 794.63 47.37 0.29 -50.42 -795.57 -65.12 -200.66 30.91z"
                data-code="floor8"
                className="floor"
              />
              <text x="14" y="325" fontSize={35} fill="black">8</text>
            </Link>
            : <path
              d="m47.03 348.03 200.92 -22.34 794.63 47.37 0.29 -50.42 -795.57 -65.12 -200.66 30.91z"
              data-code="floor8"
              className="floor_taken"
```

```
    />
}

{this.state && this.getUnits(9).length ?
    <Link to={
        {
            pathname: '/units',
            state: this.state ? this.getUnits(9) : null
        }
    }>
        <path
            d="m46.64 288.43 200.66 -30.91 795.57 65.12 -0.65 -48.20 -795.57 -85.09 -200.66 38.13z"
            data-code="floor9"
            className="floor"
        />
        <text x="14" y="265" fontSize={35} fill="black">9</text>
    </Link>
    :  <path
        d="m46.64 288.43 200.66 -30.91 795.57 65.12 -0.65 -48.20 -795.57 -85.09 -200.66 38.13z"
        data-code="floor9"
        className="floor_taken"
    />
}

{this.state && this.getUnits(10).length ?
    <Link to={
        {
            pathname: '/units',
            state: this.state ? this.getUnits(10) : null
        }
    }>
        <path
            d="m45.99 227.48 200.66 -38.13 795.57 85.09 -0.65 -52.52L246.00 121.18 47.05 167.14Z"
            data-code="floor10"
            className="floor"
        />
        <text x="-5" y="205" fontSize={35} fill="black">10</text>
    </Link>
    : <path
        d="m45.99 227.48 200.66 -38.13 795.57 85.09 -0.65 -52.52L246.00 121.18 47.05 167.14Z"
        data-code="floor10"
        className="floor_taken"
    />
}

{this.state && this.getUnits(1).length ?
    <Link to={
        {
            pathname: '/units',
            state: this.state ? this.getUnits(1) : null
        }
    }>
        <path
            d="m46.34 761.31 200.26 39.75 795.42 -85.00 -0.06 -52.67 -796.10 62.27 -199.58 -29.09z"
            data-code="floor1"
            className="floor"
```

```
                        />
                        <text x="14" y="745" fontSize={35} fill="black">1</text>
                    </Link>
                    : <path
                        d="m46.34 761.31 200.26 39.75 795.42 -85.00 -0.06 -52.67 -796.10 62.27 -199.58 -29.09z"
                        data-code="floor1"
                        className="floor_taken"
                    />
                }
            </g>
        </svg>
    </div>
        </div>
    )
  }
}

export default Building;
```

plan.component.js

```
import React from 'react';

import "../styles/plan.styles.css"
import {Link} from 'react-router-dom';

class Plan extends React.Component {
  constructor(props) {
    super(props);
    if (props.location.state) {
      this.state = {units: props.location.state};
    }

  }

  componentDidMount() {
    window.scrollTo(0, 0);
  }

  getUnit(planning) {
    let result;
    this.state.units.forEach(unit => {
      if (unit.planning === planning) {
        result = unit
      }
    })
    return result;

  }

  render() {
    return (
      <div className="plan-main-container">
        <div className={"floor-number"}>
          <h1>ПОВЕРХ {this.state.units[0].floor}</h1>
```

```
            </div>
            <div className="plan-container">
               <img className="plan-image" src={"https://i.ibb.co/gmbcd2Q/Main-plan1.png"} alt={"Floor
plan"}/>
               <svg viewBox="0 50 900 400" className={"plan-svg"}>

                  <g transform="scale(0.16)">
                     {this.getUnit("apt22") ?
                        <Link to={
                           {
                              pathname: '/unit',
                              state: this.getUnit("apt22"),
                           }
                        }>
                           <path
                              d="m587.75 2380.85 -0.65 -873.78 997.33 -0.85v0.00 445.88l163.65 0.62 0.31 427.47z"
                              data-code="apt22"
                              className="unit"/>
                        </Link>
                        : <path
                           d="m587.75 2380.85 -0.65 -873.78 997.33 -0.85v0.00 445.88l163.65 0.62 0.31 427.47z"
                           data-code="apt22"
                           className="unit-taken"/>
                     }
                     {this.getUnit("apt23") ?
                        <Link to={
                           {
                              pathname: '/unit',
                              state: this.getUnit("apt23"),
                           }
                        }>
                           <path
                              d="m1816.74 607.25 1321.52 0.20 0.20 1066.01 -924.48 0.19 -0.08 -908.79 -396.95 -
0.15z"

                              data-code="apt23"
                              className="unit"/>
                        </Link>
                        : <path
                           d="m1816.74 607.25 1321.52 0.20 0.20 1066.01 -924.48 0.19 -0.08 -908.79 -396.95 -0.15z"
                           data-code="apt23"
                           className="unit-taken"/>
                     }
                     {this.getUnit("apt24") ?
                        <Link to={
                           {
                              pathname: '/unit',
                              state: this.getUnit("apt24"),
                           }
                        }>
                           <path d="m3164.73 1673.24 0.43 -1065.17 868.49 -0.31 -0.31 1066.35z"
                              data-code="apt24"
                              className="unit"/>
                        </Link>
                        : <path d="m3164.73 1673.24 0.43 -1065.17 868.49 -0.31 -0.31 1066.35z"
                           data-code="apt24"
                           className="unit-taken"/>
```

```
                    }
                    {this.getUnit("apt21") ?
                        <Link to={
                            {
                                pathname: '/unit',
                                state: this.getUnit("apt21"),
                            }
                        }>
                            <path
                                d="m587.75 1480.81 -0.22 -873.23 1203.17 0.44v0.00 913.55l-179.79 -0.44 -0.22 -
41.41z"
                                data-code="apt21"
                                className="unit"/>
                        </Link>
                        : <path
                            d="m587.75 1480.81 -0.22 -873.23 1203.17 0.44v0.00 913.55l-179.79 -0.44 -0.22 -41.41z"
                            data-code="apt21"
                            className="unit-taken"/>
                    }
                    {this.getUnit("apt26") ?
                        <Link to={
                            {
                                pathname: '/unit',
                                state: this.getUnit("apt26"),
                            }
                        }>
                            <path
                                d="m2187.57 2380.96 -0.02 -428.55 2349.08 0.01 0.77 456.44h-478.63 0.00l-0.31 -34.21
-16.95 0.31 -0.15 5.70z"
                                data-code="apt26"
                                className="unit"/>
                        </Link>
                        : <path
                            d="m2187.57 2380.96 -0.02 -428.55 2349.08 0.01 0.77 456.44h-478.63 0.00l-0.31 -34.21 -
16.95 0.31 -0.15 5.70z"
                            data-code="apt26"
                            className="unit-taken"/>
                    }
                    {this.getUnit("apt25") ?
                        <Link to={
                            {
                                pathname: '/unit',
                                state: this.getUnit("apt25"),
                            }
                        }>
                            <path
                                d="m4059.32 1673.46 -0.22 -1119.71 1111.21 0.65 0.87 400.55 -380.06 0.87v0.00
400.77l-228.14 1.16 -0.03 315.71z"
                                data-code="apt25"
                                className="unit"/>
                        </Link>
                        : <path
                            d="m4059.32 1673.46 -0.22 -1119.71 1111.21 0.65 0.87 400.55 -380.06 0.87v0.00 400.77l-
228.14 1.16 -0.03 315.71z"
                            data-code="apt25"
                            className="unit-taken"/>
```

```
                    }
                    {this.getUnit("storage") ?
                        <Link to={
                            {
                                pathname: '/unit',
                                state: this.getUnit("storage"),
                            }
                        }>
                            <path d="m5173.84 1567.09 -360.14 0.79 0.23 -589.04 360.48 -0.23z"
                                data-code="storage"
                                className="unit"/>
                        </Link>
                        : <path d="m5173.84 1567.09 -360.14 0.79 0.23 -589.04 360.48 -0.23z"
                                data-code="storage"
                                className="unit-taken"/>
                    }
                </g>
            </svg>
        </div>
    </div>
    );
  }
}

export default Plan;
```

units-list.component

```
import React, {Component} from 'react';
import "../styles/units-list.styles.css"
import axios from "axios";
import {Link} from "react-router-dom";

export default class UnitsList extends Component {
  componentDidMount() {
    axios.get('http://localhost:5000/units/')
      .then(response => {
        this.setState({
          units: response.data,
          min: 10000,
          max: 70000,
          rooms: ['0', '1', '2'],
          floors: ['1', '2', '3', '4', '5', '6', '7', '8', '9', '10'],
          living: ['apt', 'storage']
        });
        if (this.state) {
          let state = this.state
          if (localStorage.getItem('min'))
            state.min = localStorage.getItem('min')
          if (localStorage.getItem('max'))
            state.max = localStorage.getItem('max')
          if (localStorage.getItem('rooms'))
            state.rooms = JSON.parse(localStorage.getItem('rooms'))
          if (localStorage.getItem('floors'))
```

```
                state.floors = JSON.parse(localStorage.getItem('floors'))
            if (localStorage.getItem('living'))
                state.living = JSON.parse(localStorage.getItem('living'))
            this.setState(state)
        }
    })
    .catch((error) => {
        console.log(error);
    })
window.scrollTo(0, 0);

}

setMin(min) {
    this.setState({min: min});
    localStorage.setItem('min', min);
}

setMax(max) {
    this.setState({max: max});
    localStorage.setItem('max', max);
}

setRooms(val) {
    const checked = val.checked;
    const num = val.value;
    let rooms = this.state.rooms;
    if (checked) {
        rooms.push(num)
        this.setState({rooms: rooms})
        localStorage.setItem('rooms', JSON.stringify(rooms));
    } else {
        const index = rooms.indexOf(num);
        rooms.splice(index, 1)
        this.setState({rooms: rooms})
        localStorage.setItem('rooms', JSON.stringify(rooms));
    }

}

setFloors(val) {
    const checked = val.checked;
    const num = val.value;
    let floors = this.state.floors;
    if (checked) {
        floors.push(num)
        this.setState({floors: floors})
        localStorage.setItem('floors', JSON.stringify(floors));
    } else {
        const index = floors.indexOf(num);
        floors.splice(index, 1)
        this.setState({floors: floors})
        localStorage.setItem('floors', JSON.stringify(floors));
    }
}
```

```
  setLiving(val) {
    const checked = val.checked;
    const type = val.value;
    let living = this.state.living;
    if (checked) {
      living.push(type)
      this.setState({living: living})
      localStorage.setItem('living', JSON.stringify(living));
    } else {
      const index = living.indexOf(type);
      living.splice(index, 1)
      this.setState({living: living})
      localStorage.setItem('living', JSON.stringify(living));
    }
  }

  getUnits() {
    let cards = [];
    let image;
    if (this.state) {
      this.state.units.forEach(unit => {
        if (this.state) {
          switch (unit.planning) {
            case "apt21":
              image = "https://i.ibb.co/ZxP9DcS/apt21.png";
              break;
            case "apt22":
              image = "https://i.ibb.co/n60rPFs/apt22.png";
              break;
            case "apt23":
              image = "https://i.ibb.co/xzLzCqK/apt23.png";
              break;
            case "apt24":
              image = "https://i.ibb.co/9nQ7080/apt24.png";
              break;
            case "apt25":
              image = "https://i.ibb.co/bXWrByx/apt25.png";
              break;
            case "apt26":
              image = "https://i.ibb.co/XbNXjsg/apt26.png";
              break;
            default:
              image = "https://i.ibb.co/qJCnsNP/storage-unit.png";
          }
        }
        unit.availability === true &&
        unit.price >= this.state.min &&
        unit.price <= this.state.max &&
        this.state.rooms.includes(unit.rooms.toString()) &&
        this.state.floors.includes(unit.floor.toString()) &&
        this.state.living.includes(unit.planning.toString().includes('apt') ? unit.planning.slice(0, 3) :
unit.planning.toString()) &&
        cards.push(
          <Link to={
            {
              pathname: '/unit',
```

```
                    state: unit,
                }
            } className={"unit-link"}>
                <div className={"unit-list-info unit-card"}>
                    <div className={"unit-list-info-container"}>
                        {/*{unit.planning === "storage" ?*/}
                        {/*    <h1 className={"h1-num"}>Кладівна {unit._id}</h1>*/}
                        {/*    : <h1 className={"h1-num"}>Квартира {unit.  id}</h1>*/}
                        {/*}*/}
                        <img className="unit-info-plan" src={image} alt={"Floor plan"}/>
                    </div>
                    <h5>{unit.floor} поверх</h5>
                    {
                        unit.rooms === 1 ? <h5>1-кімнатна</h5> : unit.rooms === 0 ? <h5>Студія</h5> :
                            <h5>{unit.rooms} кімнати</h5>
                    }
                    <h5>{unit.total_area} кв.м.</h5>
                    <h5>{unit.price}$</h5>
                </div>
            </Link>
        );
    });
    return cards
  }
}

render() {
  let cards = [];
  if (this.state)
    cards = this.getUnits()

  return (
    <div className="row units-list-container">
      <div className={"units-filters col-md-3 container-fluid"}>
        <section>

          <section>
            <section className="mb-4">

              <h5 className="font-weight-bold mb-3">Ціна</h5>
              <form>
                <div className="d-flex align-items-center mt-4 pb-1">
                  <div className="md-form md-outline my-0">
                    {this.state &&
                    <input onChange={event => this.setMin(event.target.value)} type="number"
                           min="10000" max={this.state.max} step="5000" value={this.state.min}
                           className="form-control mb-0"/>}
                    <label>Від</label>
                  </div>
                  <p className="px-2 mb-0 text-muted"> - </p>
                  <div className="md-form md-outline my-0">
                    {this.state &&
                    <input onChange={event => this.setMax(event.target.value)} type="number"
                           min={this.state.min} max="70000" step="5000" value={this.state.max}
                           className="form-control mb-0"/>}
                    <label>До</label>
```

```
                </div>
            </div>
        </form>

    </section>

    <section className="mb-4">

        <h5 className="font-weight-bold">Кімнати</h5>

        <div className="form-check">
            {this.state && <input onChange={event => this.setRooms(event.target)} value={'0'}
                                  checked={this.state.rooms.includes('0')} type="checkbox"
                                  className="form-check-input filled-in"/>}
            <label className="form-check-label card-link-secondary">Студія</label>
        </div>
        <div className="form-check">
            {this.state &&
            <input onChange={event => this.setRooms(event.target)} type="checkbox" value={'1'}
                   checked={this.state.rooms.includes('1')}
                   className="form-check-input filled-in"/>}
            <label className="form-check-label card-link-secondary">1</label>
        </div>
        <div className="form-check">
            {this.state &&
            <input onChange={event => this.setRooms(event.target)} type="checkbox" value={'2'}
                   checked={this.state.rooms.includes('2')}
                   className="form-check-input filled-in"/>}
            <label className="form-check-label card-link-secondary">2</label>
        </div>
    </section>

    <section className="mb-4">

        <h5 className="font-weight-bold mb-3">Поверхи</h5>

        <div className="form-check">
            {this.state && <input onChange={event => this.setFloors(event.target)} value={'1'}
                                  checked={this.state.floors.includes('1')} type="checkbox"
                                  className="form-check-input filled-in"/>}
            <label className="form-check-label card-link-secondary">1</label>
        </div>
        <div className="form-check">
            {this.state &&
            <input onChange={event => this.setFloors(event.target)} type="checkbox" value={'2'}
                   checked={this.state.floors.includes('2')}
                   className="form-check-input filled-in"/>}
            <label className="form-check-label card-link-secondary">2</label>
        </div>
        <div className="form-check">
            {this.state &&
            <input onChange={event => this.setFloors(event.target)} type="checkbox" value={'3'}
                   checked={this.state.floors.includes('3')}
                   className="form-check-input filled-in"/>}
            <label className="form-check-label card-link-secondary">3</label>
        </div>
```

```
                <div className="form-check">
                    {this.state && <input onChange={event => this.setFloors(event.target)} value={'4'}
                                          checked={this.state.floors.includes('4')} type="checkbox"
                                          className="form-check-input filled-in"/>}
                    <label className="form-check-label card-link-secondary">4</label>
                </div>
                <div className="form-check">
                    {this.state &&
                    <input onChange={event => this.setFloors(event.target)} type="checkbox" value={'5'}
                           checked={this.state.floors.includes('5')}
                           className="form-check-input filled-in"/>}
                    <label className="form-check-label card-link-secondary">5</label>
                </div>
                <div className="form-check">
                    {this.state &&
                    <input onChange={event => this.setFloors(event.target)} type="checkbox" value={'6'}
                           checked={this.state.floors.includes('6')}
                           className="form-check-input filled-in"/>}
                    <label className="form-check-label card-link-secondary">6</label>
                </div>

                <div className="form-check">
                    {this.state && <input onChange={event => this.setFloors(event.target)} value={'7'}
                                          checked={this.state.floors.includes('7')} type="checkbox"
                                          className="form-check-input filled-in"/>}
                    <label className="form-check-label card-link-secondary">7</label>
                </div>
                <div className="form-check">
                    {this.state &&
                    <input onChange={event => this.setFloors(event.target)} type="checkbox" value={'8'}
                           checked={this.state.floors.includes('8')}
                           className="form-check-input filled-in"/>}
                    <label className="form-check-label card-link-secondary">8</label>
                </div>
                <div className="form-check">
                    {this.state &&
                    <input onChange={event => this.setFloors(event.target)} type="checkbox" value={'9'}
                           checked={this.state.floors.includes('9')}
                           className="form-check-input filled-in"/>}
                    <label className="form-check-label card-link-secondary">9</label>
                </div>
                <div className="form-check">
                    {this.state &&
                    <input onChange={event => this.setFloors(event.target)} type="checkbox" value={'10'}
                           checked={this.state.floors.includes('10')}
                           className="form-check-input filled-in"/>}
                    <label className="form-check-label card-link-secondary">10</label>
                </div>
            </section>
            <section className="mb-4">

                <h5 className="font-weight-bold mb-3">Тип</h5>

                <div className="form-check">
                    {this.state && <input onChange={event => this.setLiving(event.target)} value={'apt'}
```

```
                                    checked={this.state.living.includes('apt')} type="checkbox"
                                    className="form-check-input filled-in"/>}
                        <label className="form-check-label card-link-secondary">Квартира</label>
                    </div>
                    <div className="form-check">
                        {this.state &&
                        <input onChange={event => this.setLiving(event.target)} type="checkbox"
                               value={'storage'} checked={this.state.living.includes('storage')}
                               className="form-check-input filled-in"/>}
                        <label className="form-check-label card-link-secondary">Комора</label>
                    </div>
                </section>

            </section>

        </section>
      </div>
      <div className={"row col-md-8"}>
        {cards}
      </div>
    </div>
  );
 }
}
```