Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

(повна назва факультету)

(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

(назва освітнього ступеня)

на тему:

Виконав: студент _____ курсу, групи _____

спеціальності

(шифр і назва спеціальності)

(підпис) (прізвище та ініціали)

Керівник (підпис) (прізвище та ініціали)

Нормоконтроль (підпис) (прізвище та ініціали)

Завідувач кафедри (підпис) (прізвище та ініціали)

Рецензент (підпис) (прізвище та ініціали)

Тернопіль
20___

РЕФЕРАТ

Атестаційна робота магістра. Тернопільський національний технічний університет імені Івана Пулюя, кафедра програмної інженерії, спеціальність 121 «Інженерія програмного забезпечення». ТНТУ, 2021. Сторінок 54, рисунків 25, презентація

Тема: Розробка веб-платформи для обміну будинками агенції нерухомості "Еліт Дім" на базі фреймворку Spring

В атестаційній роботі магістра була розроблена веб-платформа для обміну будинками. За допомогою Spring була розроблена бекенд частина сервісу. Фронтенд частина була написана за допомогою фреймворка Vue.js. Був здійснений опис предметної області, створена діаграма варіантів використання. Перед написанням програми було здійснено проектування архітектури. Розробив UML – діаграму класів, оаисав її методи та поля. В ході роботи ознайомився з Criteria API для створення пошуку по будинках.

Головна мета даної роботи є створення доступного сервісу для обміну будинка між власниками.

Результатом роботи є створена веб-платформа для обміну будинками на базі фреймворка Spring.

Ключові слова: веб-платформа, модель предметної області, UML - діаграма, EER - діаграма, Criteria API

ABSTACT

Master's certification work. Ternopil National Technical University named after Ivan Pulyuy, Department of Software Engineering, specialty 121 "Software Engineering". TNTU, 2021. Pages 66, figures 25, presentation

Topic: Development of a web platform for the exchange of houses of real estate agency "Elite House" based on the Spring framework

A web platform for home exchange was developed in the master's certification work. With the help of Spring, the backend part of the service was developed. The front end was written using the Vue.js framework. A description of the subject area was made, a diagram of use cases was created.

Before writing the program, the architecture was designed. Developed UML - class diagram, its methods and fields. During the work I got acquainted with the Criteria API for creating a home search.

The main purpose of this work is to create an affordable service for the exchange of the house between the owners.

The result is a web platform for exchanging houses based on the Spring framework.

Keywords: web platform, subject area model, UML diagram, EER diagram, Criteria API

ЗМІСТ

ВСТУП

Протягом останніх кількох років IT сфера дуже швидко розвивається. Це вплинуло і на оточуючий нас світ. На даний момент все більше і більше біснес починає переходити в Інтернет. Створюються багато різноманітних сайтів, CRM – систем та інших додатків, які допомагають бізнесу розвиватись.

Не є винятком і сфера нерухомості. Зараз покупка або оренда житла є досить затратною і не кожен зможе собі дозволити купити нерухомість. На даний момент агенція нерухомості тільки продають або здають в оренду будівлі. Людям не завжди такі варіанти підходять. Тому виникло питання як можна зекономити на житлі та можливо навіть переїхати на інше місце проживання на певний час.

Розібравшись з проблематикою даного питання постала ціль створити веб-сервіс, за допомогою якого можна обмінюватись будинками на певний термін. Ця ідея дуже підходить для людей, які люблять подорожувати, оскільки при вдалому знаходженні людини, яка б хотіла помінятись будинками, є можливість економно відпочити в іншій країні або місті, не переживаючи за оренду готелю.

На базі фреймворка Spring можна реалізувати клієнт-серверну архітектуру веб-сервіса для обміну будинками. Використовуючи весь функціонал фреймоворка не буде проблемо реалізувати функціонал даного сервісу. З використанням Java 8 і вище можна використовувати Stream API для полегшення роботи з даними. Крім того для роботи з базою даних можна використати фреймворк Hibernate, який дає легкий каркас для відображення між реляційною базою даних та обєктно-орієнтованою моделлю. Також для полегшення написання запитів в базу даних можна використати Criteria API для побудови пошукової системи будинків, яка буде одним з основних функціоналів веб-платформи.

ОГЛЯД ПРЕДМЕТНОЇ ОБЛАСТІ

1.1. Загальні відомості про веб-сайти

Кожен веб-сайт складається з багатьох частин і основних елементів. Саме всі ці елементи, об’єднані разом, створюють веб-сайт [1].

Веб-сторінка або веб-сторінка — це документ, зазвичай написаний у HTML, який переглядається в Інтернет-браузері. Доступ до веб-сторінки можна отримати, ввівши URL-адресу в адресний рядок браузера. Веб-сторінка може містити текст, графіку та гіперпосилання на інші веб-сторінки та файли. Веб-сторінка часто використовується для надання інформації глядачам, включаючи зображення або відео, які допомагають проілюструвати важливі теми. Веб-сторінка також може використовуватися як спосіб продажу продуктів або послуг користувачам [2].

Перше, що повинен мати веб-сайт, це правильне представлення бізнесу та цілей, які стоять за створенням веб-сайту. Сьогодні ми бачимо багато типів веб-сайтів. Отже, перше, на що потрібно звернути увагу, це мати правильний тип веб-сайту. Якщо ви бізнес, вам потрібен бізнес-сайт, але якщо ви працюєте в організації, але не для прибутку, ваш веб-сайт буде мало відрізнятися від бізнесу. Правильний тип веб-сайту найкраще відображатиме вашу мету. Це створить правильну платформу для правильної структури та правильної інформації, яку ви повинні мати на своєму веб-сайті. Чудовий веб-сайт – це той, який більше цінують його користувачі або більшість людей, які його відвідують.

Веб-сайт хороший, коли він створений в основному з урахуванням його аудиторії та того, як найкраще він може їх обслуговувати. Хороший і чудовий веб-сайт повинен мати багато задоволеної аудиторії. Є ще одне не менш важливе призначення вашого сайту. Це потрібно для того, щоб охопити та задовольнити цільову аудиторію.

Кожен сайт повинен мати:

1) Актуальний контент

2) Проста та ефективна презентація через дизайн

3) Сторінки та вміст, щоб допомогти користувачам краще зрозуміти вміст

4) Корисні блоки або інформація для користувачів

5) Посилання або кнопки для виконання дії

6) Легкий для читання та навігації веб-сайт

7) Добре визначена структура

1.2 Веб-сервіси. Переваги та Недоліки

Веб-сервіси мають погані назви та нечітко визначені. Як наслідок, на ринку існує велика плутанина щодо того, що саме таке веб-сервіси, яке їх значення та де вони вписуються в загальну картину IT. Тим не менш, фраза «веб-послуги» стала де-факто терміном для невизначеного набору можливостей. По суті, веб-сервіси — це функціональні можливості додатків, що використовуються в системах, які приймають запити від інших систем локально або через Інтернет за допомогою легких, нейтральних для постачальників комунікаційних технологій [4]. Розбиття визначення має зробити його зрозумілішим:

1) Інкапсульований означає, що реалізація кожної веб-служби невидима ззовні веб-сервіси. Його функціональність відома лише за інтерфейсом, який він надає. По суті, веб-сервіси абстрагують реалізацію від інтерфейсу, подібно до того, як XML відокремлює вміст від обробки. Веб-сервіси успадковують властивість інкапсуляції від об'єктно-орієнтованих архітектур, наприклад, на основі Java, C++ і COM.

2) Веб-сервіси слабо пов'язані. Слабо пов’язані означає, що веб-сервіси та програми, які їх викликають (відомі як споживачі веб-служб), можна змінювати

незалежно один від одного, замість того, щоб вимагати переробки залучених компонентів. Споживач веб-сервісу — це будь-яка частина програмного забезпечення, яка знаходить і викликає (або прив’язується до) веб-сервісу . Споживачі самі можуть бути або не бути веб-сервісами.

3) Контракт означає, що поведінка веб-сервісу, а також те, як підключитися або зв’язатися з нею, а також її вхідні та вихідні параметри, доступні для тих споживачів, які мають до неї доступ.

4) У контексті цього звіту, який все більше сприймається індустрією програмного забезпечення в цілому, веб-сервіси побудовані на стандартних протоколах XML (розширювана мова розмітки) і HTTP (протокол передачі гіпертексту), які є відкритими та вільно доступними. Крім того, веб-сервіси використовують SOAP (простий протокол доступу до об’єктів), WSDL (мову опису веб-сервісів) і UDDI (універсальний опис, виявлення та інтеграцію), які є стандартними протоколами, заснованими на XML. Як ми побачимо пізніше, визначення веб-сервісів вимагає стандартів, але ці специфічні стандарти (SOAP, HTTP, WSDL) не є обов’язковими для того, щоб рішення розглядалося як «веб-служба», якщо прийняті інші стандарти.

Веб-сервіси самовизначаються. Самовизначення — це особливість XML, яку в повній мірі використовують основні технології веб-сервісів SOAP і WSDL. Оскільки файли WSDL під час виконання, а також під час розробки описують, як обмінюватися інформацією з веб-сервісами, такі описи можна змінювати на льоту, не порушуючи користувачів веб-служб.

Веб-сервіси підтримують динамічне виявлення та виклик. Користуючись перевагами реєстру UDDI, споживач веб-сервісу може знайти та викликати веб-службу під час виконання — без необхідності завчасно програмувати цю можливість доступу до певного веб-сервіса. Ця можливість називається інтеграцією Just-in-time (JIT) [4].

Перевагами веб-сервісів є:

1) Сумісність – це найважливіша перевага веб-сервісів. Веб-сервіси зазвичай працюють за межами приватних мереж, пропонуючи розробникам

непроприетарний шлях до їхніх рішень. Тому розроблені послуги, швидше за все, матимуть більший термін служби, забезпечуючи кращу окупність інвестицій у розроблену послугу. Веб-служби також дозволяють розробникам використовувати вподобані мови програмування. Крім того, завдяки використанню методів зв'язку на основі стандартів, веб-сервіси практично не залежать від платформи.

2) Зручність використання – веб-сервіси дозволяють розкривати бізнес-логіку багатьох різних систем через Інтернет. Це дає вашим програмам свободу вибору веб-служб, які їм потрібні. Замість того, щоб заново винаходити колесо для кожного клієнта, вам потрібно лише включити додаткову специфічну для програми бізнес-логіку на стороні клієнта. Це дозволяє вам розробляти служби та/або клієнтський код, використовуючи потрібні мови та інструменти.

3) Повторне використання – веб-сервіси надають не компонентну модель розробки додатків, а найбільш близьку до розгортання таких сервісів із нульовим кодуванням. Це полегшує повторне використання компонентів веб-служби відповідно до інших служб. Це також полегшує розгортання застарілого коду як веб-служби.

4) Можливість розгортання – веб-сервіси розгортаються за допомогою стандартних Інтернет-технологій. Це дає змогу розгортати веб-служби навіть через брандмауну на серверах, що працюють в Інтернеті на іншому кінці земної кулі. Крім того, завдяки використанню перевірених стандартів спільноти, базова безпека (наприклад, SSL) вже вбудована [3].

До недоліків можна віднести:

1) Хоча простота веб-сервісів є перевагою в деяких аспектах, вона також може бути перешкодою. Веб-сервіси використовують протоколи простого тексту, які використовують досить докладний метод для ідентифікації даних. Це означає, що запити веб-служби більші, ніж запити, закодовані за допомогою двійкового протоколу. Додатковий розмір насправді є проблемою лише для низькошвидкісних з'єднань або дуже зайнятих з'єднань.

2) Хоча HTTP і HTTPS (основні веб-протоколи) прості, вони насправді не призначені для довготривалих сеансів. Як правило, браузер встановлює з'єднання HTTP, запитує веб-сторінку і, можливо, деякі зображення, а потім розриває з'єднання. У типовому середовищі CORBA або RMI клієнт підключається до сервера і може залишатися на зв'язку протягом тривалого періоду часу. Сервер може періодично надсилати дані клієнту. Така взаємодія є складною з веб-сервісами, і вам потрібно зробити трохи додаткової роботи, щоб компенсувати те, чого HTTP не робить для вас.

3) Проблема з HTTP і HTTPS, коли справа доходить до веб-сервісів, полягає в тому, що ці протоколи «без стану» — взаємодія між сервером і клієнтом, як правило, є короткою, і коли немає обміну даними, сервер і клієнт не знають один одного. Точніше, якщо клієнт робить запит до сервера, отримує деяку інформацію, а потім негайно виходить з ладу через відключення електроенергії, сервер ніколи не дізнається, що клієнт більше не активний. Серверу потрібен спосіб стежити за тим, що робить клієнт, а також визначати, коли клієнт більше не активний.

4) Як правило, сервер надсилає клієнту якусь ідентифікацію сеансу, коли клієнт вперше звертається до сервера. Потім клієнт використовує цю ідентифікацію, коли робить додаткові запити до сервера. Це дозволяє серверу відкликати будь яку інформацію про клієнта. Сервер зазвичай повинен покладатися на механізм тайм-ауту, щоб визначити, що клієнт більше не активний. Якщо сервер не отримує запит від клієнта через заздалегідь визначений проміжок часу, він припускає, що клієнт неактивний, і видаляє будь-яку інформацію про клієнта, яку він зберігав. Ці додаткові витрати означають більше роботи для розробників веб-сервісів [3].

1.3 Огляд конкурентів

В наш час обмін будинками не є дуже популярним оскільки інформації по даному виду діяльності дуже мало і тому люди не спішать обмінювати валасні будинки. Проте подібні ідеї спостерігаються на сайті OLX та низки інших сайтів оренди житла. Проте відсутність веб-платформи робить їх старання марними оскільки немає чіткої інструкції як можна зробити цей обмін та обмежений вибір будинків для обміну. Також не завжди присутня інформація про власника і будинок. Знаючи всю проблематику можна створити свою веб-платформу, яка буде нівелювати всі недоліки, які спостерігаються на цих сайтах відносно процесу обміну будинків.

1.4. Обгрунтування вибору середовища розробки програмної системи та мови програмування.

Spring Framework — це модульна структура з безліччю модулів для різних завдань, а spring core — це основний модуль, на якому працюють інші модулі системи. Модульність є однією з важливих особливостей фреймворку Spring, що робить його простим у розробці додатків.

1. Вирішення труднощів розробки додатків Enterprise

Spring вирішує труднощі розробки складних додатків, він надає Spring Core, Spring IoC і Spring AOP для інтеграції різних компонентів бізнес-додатків.

2. Підтримка розробки корпоративних додатків за допомогою POJO

Spring підтримує розробку корпоративних додатків за допомогою класів POJO, що усуває необхідність імпортувати важкий контейнер Enterprise під час розробки. Це значно полегшує тестування програми.

3. Легка інтеграція інших фреймворків

Spring розроблений для використання з усіма іншими фреймворками Java, ви можете використовувати ORM, Struts, Hibernate та інші фреймворки Java разом. Spring Framework не накладає жодних обмежень на фреймворки, які будуть використовуватися разом.

4. Тестування додатків

Spring Container можна використовувати для розробки та запуску тестових випадків за межами корпоративного контейнера, що значно полегшує тестування.

5. Модульність

Spring Framework є модульною структурою, і вона постачається з багатьма модулями, такими як Spring MVC, Spring ORM, Spring JDBC, Spring Transactions тощо, які можна використовувати відповідно до вимог програми в модульній формі.

6. Spring Transaction Management

Інтерфейс Spring Transaction Management дуже гнучкий, його можна налаштувати на використання локальних транзакцій у невеликих програмах, які можна масштабувати до JTA для глобальних транзакцій [5].

1.5. Технічний аспект проблеми

Веб-платформа для обміну будинками буде зберігати великі обєми інформації про користувача та його будинок. Під час додавання будинку або інформації про користувача буде можливість завантажувати документи, які підтверджують власність будинку, поспорт або іншу конфіденційну інформацію Звідси слідую що потрібно зробити секюрну клієнт-серверну архітектуру з використанням методу JWT(JSON Web Token), з JWT фільтром для запобігання взламу даного сервіса.

2. РОЗРОБКА МОДЕЛІ ТА ПРОГРАМНОГО КОЛМПЛЕКСУ

2.1 Проєктування веб-платформи для обміну будинками

В процесі розробки веб-платформи для обміну будинками була розроблена модель предменої області та бізнес модель. Здійснено проєктування архітектури. Перед написанням коду було створено діаграму класів та EER діаграму.

2.1.1. Розробка моделі предметної області

Предметна область: обмін будинками агенції нерухомості.

При аналізі предметної області було враховано фактори, які безпосередньо стосуються особливостей системи:

- Наскільки часто інформація оновлюється;
- Яка додаткова інформація має бути відображена;
- Яким чином виконується обмін інформацією між акторами та системою;
- Як зберігається інформація.

Зробивши аналіз всіх критеріїв процесу було виявлено список проблем, з якими можуть зіштовхнутися цільові актори, серед них такі:

1) відсутність списку будинків та їх дати для обміну
2) не оптимізований процес обміну будинками
3) відсутність даних по власникам будинків
4) відсутність комунікації між власниками будинків
5) відсутність зручного пошуку будинків по їх параметрах
6) відсутність чіткої адреси будинків
7) відсутність сповіщень по нових запитах на обмін будинку

8) відсутність оцінки будинків

Головна причина побудови цієї системи – забезпечення цілісності даних, запобігання ймовірної втрати даних, спрощення обміну будинками, автоматизація обробки, збереження даних. Звідси випливають основні цілі даної платформи:

1) Зробити сервіс, де можна знайти інформацію по будинках та час їх обміну;

2) Зробити алгоритм обміну будинками для пришвидшення цього процесу;

3) Зробити базу даних по користувачам для структуризації всієї інформації по власникам будинків;

4) Зробити функціонал з чатами для полегшення комунікації між користувачами;

5) Зробити пошукову систему для будинків по різним параметрах для полегшення пошуку будинків;

6) Додати гугл карту для полегшення знаходження будинків;

7) Створити API для сповіщень, щоб користувач не пропустив важливу інформацію по обміну;

8) Додати форму для коментарів та рейтингу.

Основні задачі, що буде вирішувати сервіс:

1) Авторизація відповідно до прав доступу(User, Admin);

2) Особистий кабінет, де буде профіль юзера та інформація по його будинках та запитах;

3) Зручний пошук будинків по їх параметрах;

4) Великий об’єм інформації по будинках;

5) Виводити інформацію по запитах для обміну;

6) Можливість обмінюватись на певний час будинками у разі згоди обох сторін;

7) Можливість отримувати сповіщення на телефон або імейл про нові будинки або запити на обмін.

2.1.2. Розробка бізнес моделі

В ході проектування була розроблена діаграма варіантів використання (див. рис 2.1.2.1).

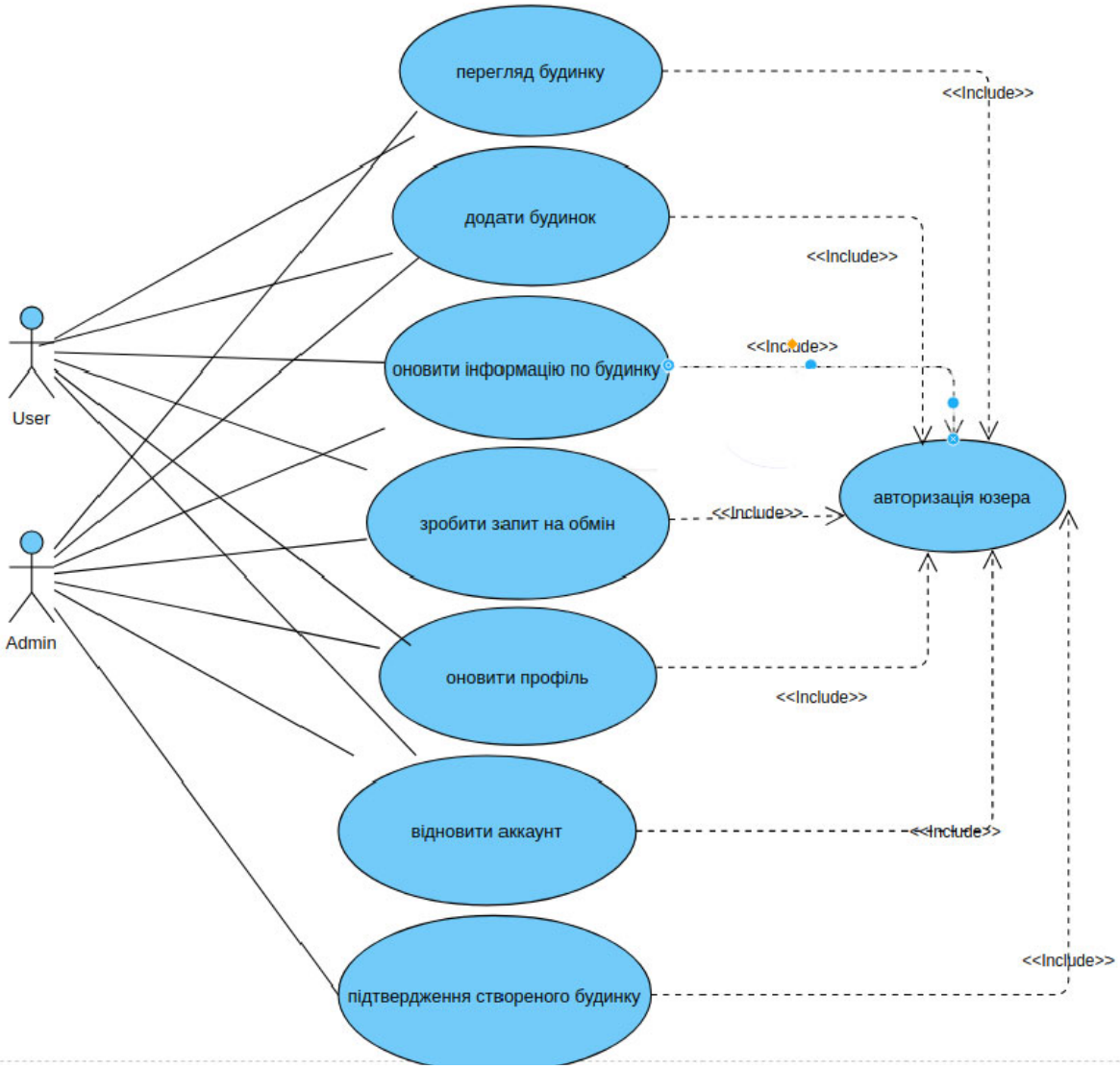

Рисунок 2.1.2.1 – Діаграма варіантів використання

На веб-платформі будуть 2 ролі: User та Admin.

Користувач з роллю User зможе робити такі дії:

- перегляд будинків;

- додавання будинку;
- оновлення інформації по будинку;
- зробити запит на обмін;
- оновити профіль;
- відновити аккаунт.

Користувач з роллю Admin матиме доступ до функціоналу:

- перегляд будинків;
- додавання будинку;
- оновлення інформації по будинку;
- зробити запит на обмін;
- оновити профіль;
- відновити аккаунт;
- підтвердження створеного будинку.

Адмін зможе перевіряти чи всі дані по будинку заповнені. У разі якщо інформвція по будинку не повна будинок не буде відображатись в пошуку. Тому користувачу потрібно буде добавляти інформацію по будинку і чекати поки адмін підтвердить зміни.Після того будинок буде доступний в пошуку і інші користувачі зможуть дивитись інформацію по ньому.

2.1.3. Проєктування архітектури

Перед написанням функціоналу була створена UML-діаграма класів з полями та методами, які будуть реалізовані на веб-платформі для обміну будинками (див. рис.2.1.3.1).

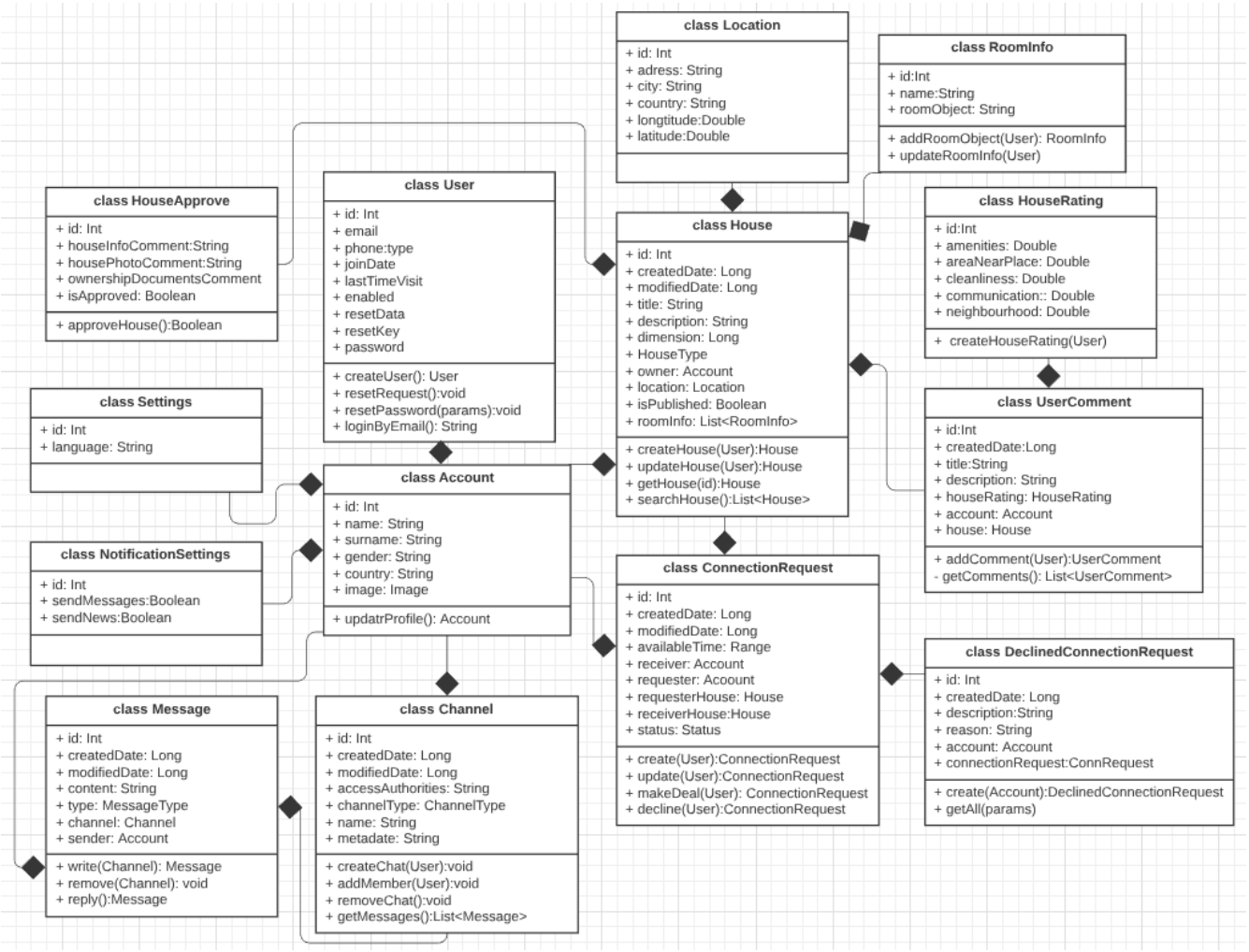

Рисунок 2.1.3.1 – UML- діаграма класів

На рисунку 2.1.3.1 можна побачити основні класи з яких складається діаграма.

Клас User (рис. 2.1.3.2) відповідає за всіх користувачів в системі. В ньому є інформація по юзеру: імейл, телефон, пароль, дата реєстрації та дата останнього входу. Даний клас матиме такі методи: створення нового юзера, авторизація юзера, відправка запиту на зміну пароля та зміну пароля за допомогою верифікаційного ключа, який буде приходити на пошту у разі запиту на зміну пароля.

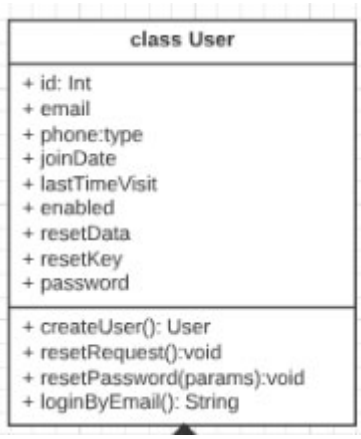

Рисунок 2.1.3.2 – Клас User

Клас Account(рис. 2.1.3.3) відповідає за профіль корискувача. В ньому є інформація про користувача: його імя, фамілія, стать, країна та аватарка. Користувач зможе також оновлявати інформацію про себе.

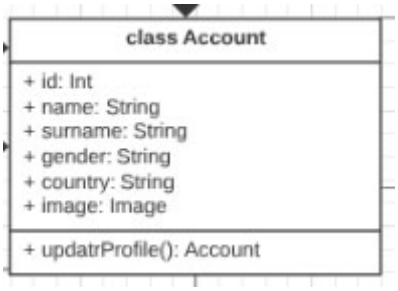

Рисунок 2.1.3.3 – Клас Account

Клас House(рис. 2.1.3.4) відповідає за будинок. Даний клас містить загальну інформацію по будинку: тип будинку, власника, місце знаходження та інформацію про кімнати. Даний клас буде включати пошук будинку по айді, створення та оновлення інфомації по будинку та пошук будинків по різних параметрах.

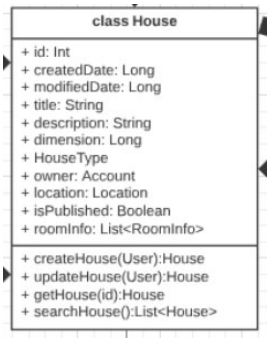

Рисунок 2.1.3.4 – Клас House

Клас Location(рис. 2.1.3.5) містить дані, які показують в якій локації занходиться будинок. Даний клас містить не тільки адресу, місто і країну. Також є довгота і широта, яка передається у вигляді параметрів на фронтенд і відображається на гугл карті.

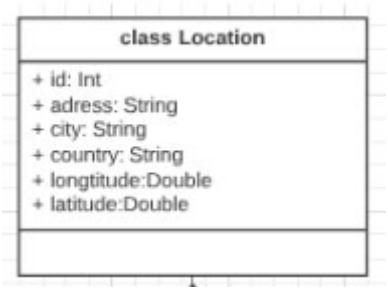

Рисунок 2.1.3.5 – Клас Location

Клас RoomInfo(рис. 2.1.3.6) відповідає за інформацію про кімнату, яка знаходиться в будинку. Можна буде описати характеристики даної кімнати, що зробить інформацію про будинок більш детальною.

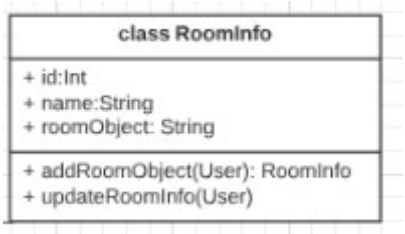

Рисунок 2.1.3.6 – Клас RoomInfo

Клас UserComment(рис. 2.1.3.7) відповідає за коментар, який користувач зможе залишити після перегляду будинку. Буде зроблена спеціальна форма для коментарів, де користувач може поділитись своєю думкою про будинок та поставити рейтинг. Даний клас міститиме методи на пошук коментарів по будинку та створення нових коментарів.

Клас HouseRating(рис. 2.1.3.7) відповідає за рейтинг будинку. Даний рейтинг містить в собі такі параметри: зручність, район навколо будівлі(наявність різної інфраструктури), чистота та комунікація з власником на рахунок.

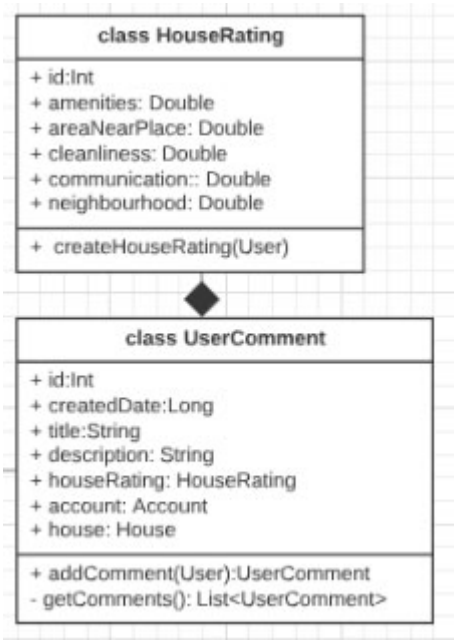

Рисунок 2.1.3.7 – Клас UserComment та HouseRating

Клас ConnectionRequest(рис. 2.1.3.8) відповідає за запит по обміну будинками. Тут буде міститись інфомація по будинках, їх власниках, даті створення запиту. Також даний клас матиме статуси: ACCEPTED, DECLINED, NOT_SET, DEAL. При сторенні запиту статус буде NOT_SET. Якщо погодився один користувач на обмін тоді буде статус ACCEPTED, в іншому випадку DECLINED. Якщо обидва користувач згодні з обміном то статус буде DEAL. Відповідно методи в даному класі будуть на створення, оновлення, відхилення та погодження запиту.

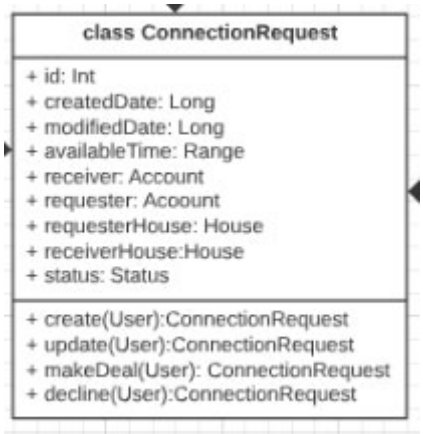

Рисунок 2.1.3.8 – Клас ConnectionRequest

Клас DeclinedConnectionRequest(рис. 2.1.3.9) відповідає за відхилений запит по будинку. Даний клас буде використовуватись якщо статус запита на будинок буде DECLINED. Він містить такі поля: причину відхилення та опис проблеми або скарги.

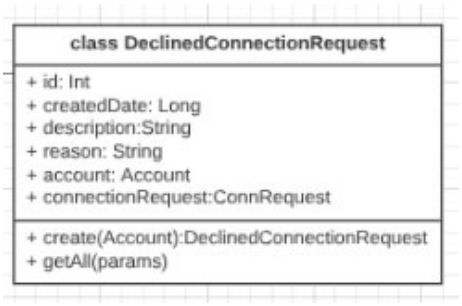

Рисунок 2.1.3.9 – Клас DeclinedConnectionRequest

Клас Channel(рис. 2.1.3.10) відповідає за чати в ситемі. Містить такі поля: тип чату(DIRECT, SUPPORT), дату створення та модифікації, назв, метадату та права доступу. В даному класі буде можливість створювати нові чати, додавати нових учасників чату, видаляти чат та виводити всі повідомлення відповідно до чату.

Клас Message(рис. 2.1.3.10) відповідає за повідомлення. Містить тип повідомлення(текст або файл), контент, дату написання та чат в якому буде написане.

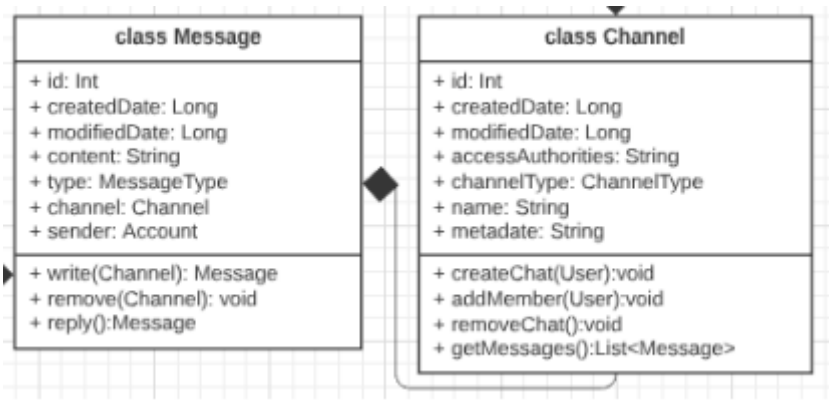

Рисунок 2.1.3.10 – Клас Message та Channel

Клас Settings(рис. 2.1.3.11) відповідає за мову яка буде відображатись на сайті.

Клас NotificationSettings(рис. 2.1.3.11) відповідає за пересилання сповіщень.

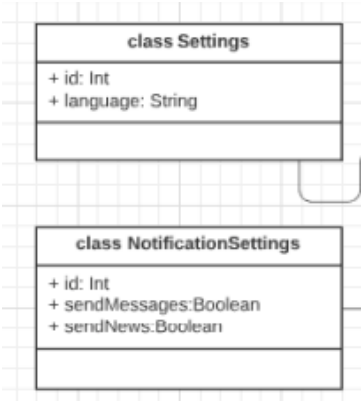

Рисунок 2.1.3.11 – Клас Settings та NotificationSettings

Клас HouseApprove(рис. 2.1.3.12) відповідає за підтвердження створеного будинку. Може бути використане тільки адміном для перевірки правильності заповнених даних під час додавання нового будинку.

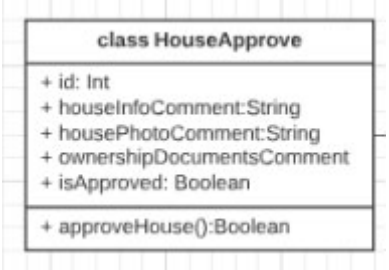

Рисунок 2.1.3.12 – Клас HouseApprove

В ході проектування архітектури була розроблена EER діаграма для веб-платформи для обміну будинками(рис. 2.1.3.13).

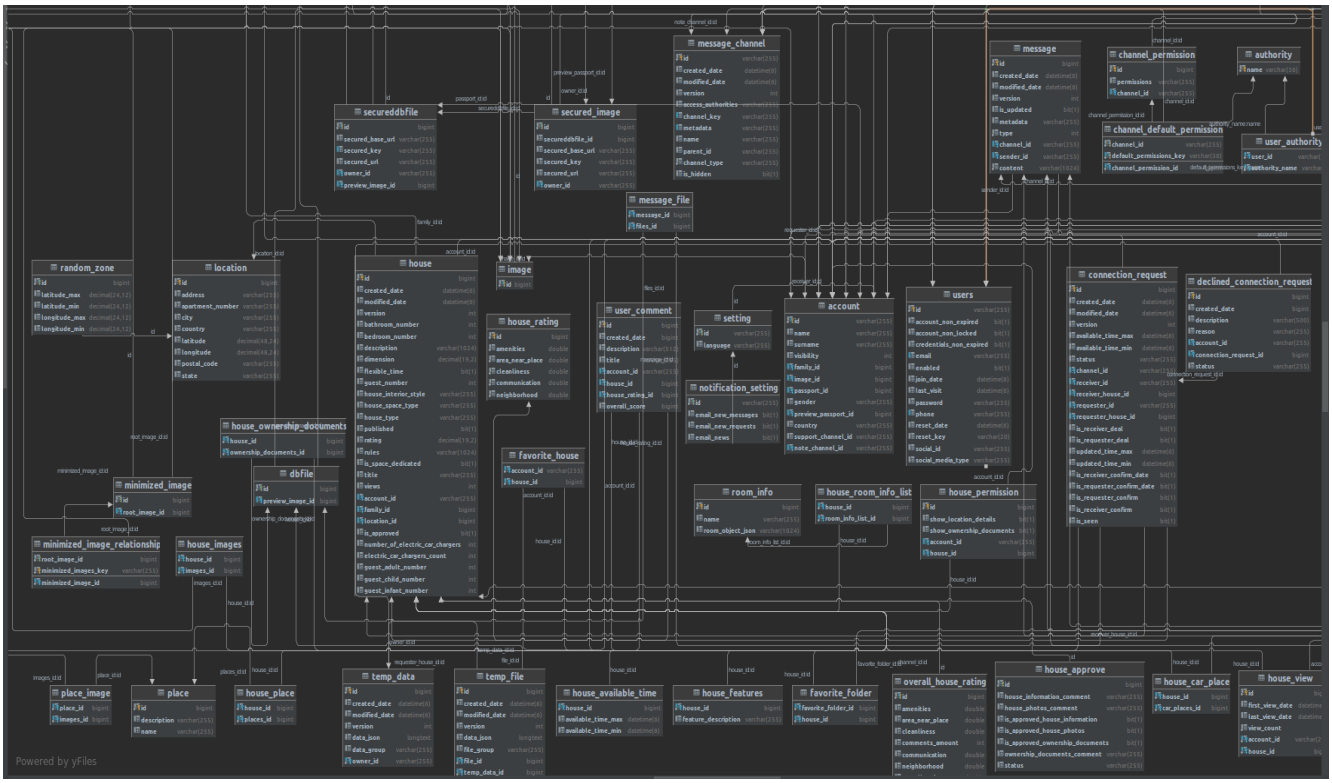

Рисунок 2.1.3.13 – EER діаграма

Можна виділити основні сутності даної діаграми: house, connection_request, user, account, channel, message, room_info. Опис даних сутностей був проведений вище під час опису UML – діаграми(див. 2.1.3.1).

2.2 Конструювання веб-платформи для обміну будинками

??????????????????????????????????

2.2.1. Реалізація ключових класів

Створюємо проект на бекенді та добавляємо мавен залежності(рис. 2.2.1.2)

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-jpa</artifactId>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-jdbc</artifactId>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-mail</artifactId>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-security</artifactId>
</dependency>
<dependency>
    <groupId>org.springframework.security</groupId>
    <artifactId>spring-security-messaging</artifactId>
</dependency>
```

Рисунок 2.2.1.2 – Мавен залежності для проекта

Метод додавання будинку приймає параметри користувача та обєкт будинка(рис. 2.2.1.3). Перевіряється чи власник може додати ще один будинок. Якщо є обмеження на створення нового будинку то буде викинута помилка в консолі.

```
public HouseDTO createHouse(User creator, String houseOwnerId, CreateHouseDTO createHouseDTO) {
    if (creator.getId().equals(houseOwnerId)) {
        return createHouse(creator, createHouseDTO);
    }

    Account houseOwnerAccount = accountRepository.getOne(houseOwnerId);

    GlobalAccountPermission permission = globalAccountPermissionRepository.findByAccounts(houseOwnerAccount, creator.getAccount());

    if (permission == null || !permission.getPermissionList().contains(GlobalAccountPermission.Permission.CREATE_HOUSE)) {
        throw new IllegalStateException("No permissions to create a house for this user");
    }

    return createHouse(houseOwnerAccount, createHouseDTO);
}
```

Рисунок 2.2.1.3 – Метод додавання будинку

Для пошуку будинків вмкористав Criteria API(рис. 2.2.1.4). У Hibernate API Criteria допомагає нам динамічно створювати об’єкти запиту критеріїв. Критерії — це ще одна техніка пошуку даних, крім HQL та власних запитів SQL. Основна перевага Criteria API полягає в тому, що він інтуїтивно розроблений для маніпулювання даними без використання жорстко закодованих операторів SQL. Всі параметри зарисуються у обєкт SearchHouse. Якщо один з параметрів null тоді по цьому параметру не йде пошук. Якщо SearchHouse дорівнює null то виводяться всі будинки.

```
public List<HouseDTO> search(User user, SearchHouseDTO searchHouseDTO, Pageable pageable) {
    CriteriaBuilder criteriaBuilder = entityManager.getCriteriaBuilder();
    CriteriaQuery<House> criteriaQuery = criteriaBuilder.createQuery(House.class);
    Root<House> houseRoot = criteriaQuery.from(House.class);

    Join<Object, Object> timeRange = houseRoot.join( s: "availableTimeRanges", JoinType.LEFT);

    List<Predicate> predicates = proxy.initializePredicates(user, searchHouseDTO, criteriaBuilder, houseRoot, timeRange);

    if (user != null && Boolean.TRUE.equals(searchHouseDTO.getMatchesMyHouseTime())) {
        predicates = proxy.searchMatchesMyHouseTime(user, criteriaBuilder, houseRoot, timeRange);
    }

    CriteriaQuery<House> result = criteriaQuery.select(houseRoot).where(predicates.toArray(new Predicate[0])).distinct(true);
    result = sorting(result, criteriaBuilder, houseRoot, pageable.getSort().toString());


    return houseMapper.toDto(proxy.pageableResolver(result, pageable).getContent(), Arrays.asList(Mapper.SET_PARENT, HouseMapper
}
```

Рисунок 2.2.1.4 – Метод пошуку будинку

Для обрахунку рейтингу був написаний окремий метод (рис 2.2.1.5). Суть його полягає в тому що під час додавання нового коментаря користувач може

дати оцінку даному будинку. При кожному додаванні йде переобрахунок загального рейтинга. Звідси слідує що по продуктивності метод працює добре, оскільки не потрбно брати масив рейтингів з кожного коментаря.

```
private void calculateHouseRating(House house) {
    OverallHouseRating overallHouseRating = overallHouseRatingRepository.getOne(house.getId());

    List<UserComment> userComments = userCommentRepository.findAllByHouseId(house.getId());

    List<HouseRating> houseRatings = userComments.stream().map(UserComment::getHouseRating).collect(Collectors.toList());

    int commentsAmount = houseRatings.size();
    double cleanliness = houseRatings.stream().map(HouseRating::getCleanliness).mapToDouble(Double::doubleValue).sum() / commentsAmount;
    double communication = houseRatings.stream().map(HouseRating::getCommunication).mapToDouble(Double::doubleValue).sum() / commentsAmoun
    double areaNearPlace = houseRatings.stream().map(HouseRating::getAreaNearPlace).mapToDouble(Double::doubleValue).sum() / commentsAmoun
    double neighborhood = houseRatings.stream().map(HouseRating::getNeighborhood).mapToDouble(Double::doubleValue).sum() / commentsAmount;
    double amenities = houseRatings.stream().map(HouseRating::getAmenities).mapToDouble(Double::doubleValue).sum() / commentsAmount;
    double overallRating = (cleanliness + communication + areaNearPlace + neighborhood + amenities) / 5;

    overallHouseRating.setCommentsAmount(commentsAmount);
    overallHouseRating.setCleanliness(cleanliness);
    overallHouseRating.setCommunication(communication);
    overallHouseRating.setAreaNearPlace(areaNearPlace);
    overallHouseRating.setNeighborhood(neighborhood);
    overallHouseRating.setAmenities(amenities);
    overallHouseRating.setOverallRating(overallRating);

}
```

Рисунок 2.2.1.5 – Метод обрахунку рейтингу будинку

2.2.2. Розробка GUI

На початку був створений проект для фронтенд частини (рис. 2.2.1.1)

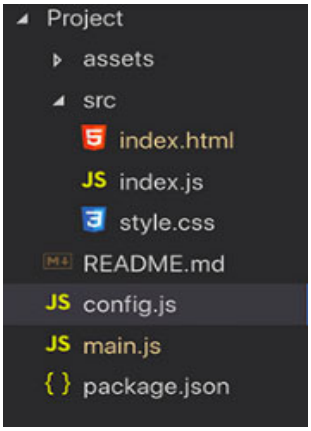

Рисунок 2.2.1.1 – Структура фронтенд частини

Для написання платформи був використаний фреймворк VueJS. Перевагою є компактність з крихітним розміром 18-21 КБ, що дозволяє користувачам завантажувати фреймворк в найкоротші терміни. Оскільки швидкість завантаження сторінки є одним з основних факторів то це робить даний фреймворк популярним.

Багато сторонніх бібліотек і компонентів підтримуються Vue JS, що дозволяє розробникам інтегрувати нові технології з існуючими додатками і заощаджує величезну кількість часу для програмістів на підтримці програм у курсі нових тенденцій на ринку.

Під час візуалізації веб-сторінок буде діяти DOM (Document Object Model). Простими словами, DOM містить представлення різних сторінок HTML разом зі стилями, елементами та вмістом як об’єкти. Ці об’єкти зберігаються у вигляді деревоподібної структури.

2.2.3. Тестування програмного забезпечення та оцінка якості

Тестування спрямоване на виявлення дефектів у програмному забезпеченні. Але незалежно від того, наскільки ретельно продукт був перевірений, ми ніколи не можемо бути на 100 відсотків впевнені, що немає дефектів. Ми можемо використовувати лише тестування, щоб зменшити кількість невиявлених проблем.

Вартість помилки зростає в геометричній прогресії на всіх етапах SDLC. Тому важливо розпочати тестування програмного забезпечення якомога швидше, щоб виявлені проблеми були вирішені, а не «сніжний ком».

Повна відсутність у вашому продукті помилок не обов’язково означає його успіх. Незалежно від того, скільки часу ви витратили на полірування свого коду або покращення функціональності, якщо ваш продукт не є корисним або не відповідає очікуванням користувачів, він не буде прийнято цільовою аудиторією.

Крім того, потрібно дотримуватись таких принципів:

1) Тестування має бути незалежним процесом, яким займаються неупереджені професіонали;

2) Перевірте на наявність недійсних і неочікуваних введених значень, а також дійсних і очікуваних значень;

3) Тестування має виконуватися лише на статичній програмі (в процесі тестування не слід вносити жодних змін);

4) Використовуйте вичерпну та вичерпну документацію, щоб визначити очікувані результати випробувань.

В даній роботі будуть протестовані наступні методи:

- Авторизація користувача;
- Реєстрація користувача;
- Перегляд будинків;
- Пошук будинків;
- Відправка запиту на обмін будинками.

2.2.4 Результат розробки

Для входу в систему створюємо аккаунт (див. рис. 2.2.4.1). Потрібно ввести ім'я, фамілію, емейл і телефон. Далі приходить сповіщення на емейл з валідаційним кодом, який вводиться на 2 кроці разом з паролем.

??????????????????????????????????

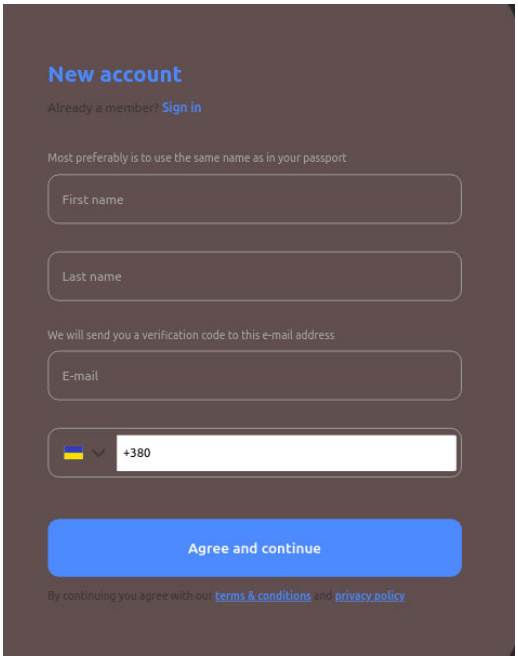

Рисунок 2.2.4.1 – Сторінка реєстрації користувача

Якщо аккаунт створений, то можна залогінитись, ввівши імейл та пароль і перейти на головну сторінку (див. рис. 2.2.4.2).

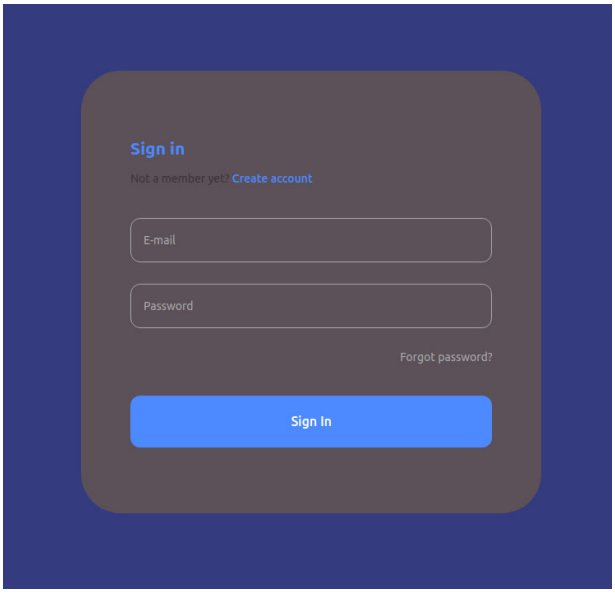

Рисунок 2.2.4.2 – Сторінка входу

У разі якщо пароль користувач не памятає його можна відновити (див.рис. 2.2.4.3). На першому кроці вводимо емейл і на пошту має прийти верифікаційний код.

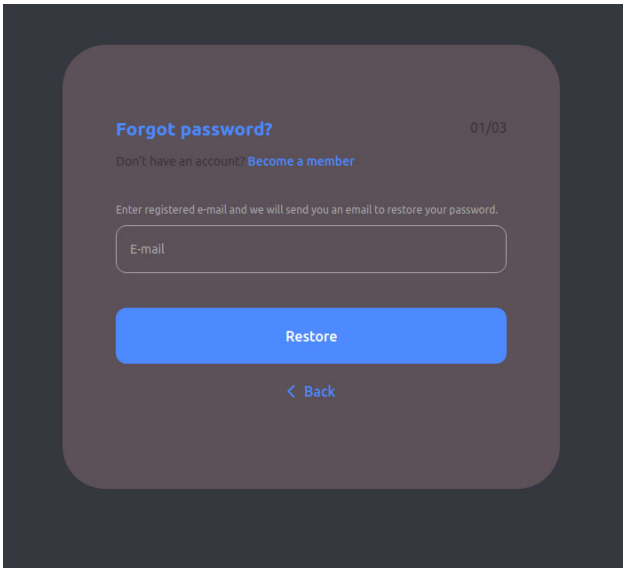

Рисунок 2.2.4.3 – Сторінка відновлення паролю(1 крок)

На другому кроці вводимо верифікаційний код і переходимо на 3 крок(див.рис. 2.2.4.4). В разі якщо код не прийшов можна відправити запит ще раз.

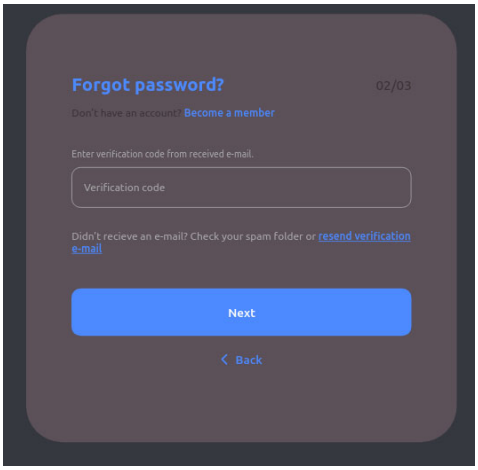

Рисунок 2.2.4.4 – Сторінка відновлення паролю(2 крок)

У разі успішного вводу верифікаційного коду переходимо до 3 кроку(див.рис. 2.2.4.5). На цій вкладці вводимо пароль і підтверджуємо введений пароль. Нажимаємо кнопку Save і пароль відновлено.

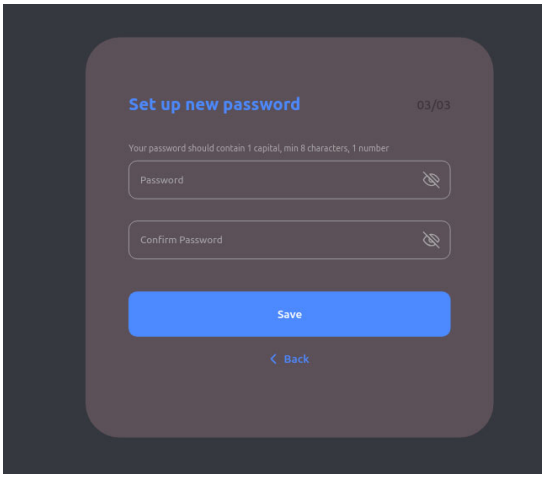

Рисунок 2.2.4.5 – Сторінка відновлення паролю(3 крок)

Після входу на веб-платформу ми переходи на головну сторінку, де є випадаючий список (див.рис. 2.2.4.6).

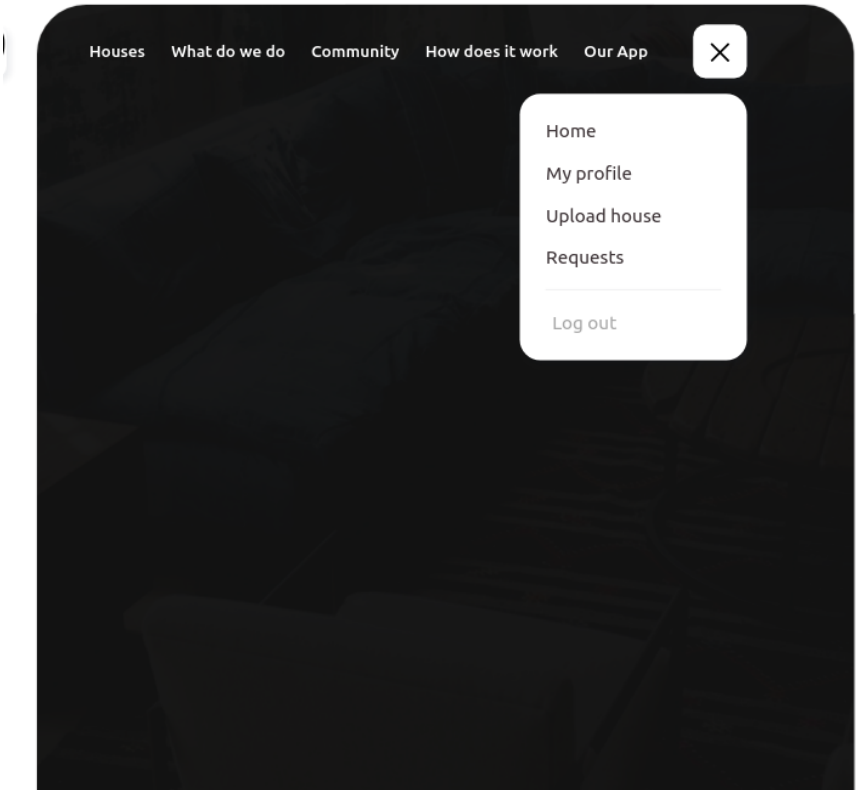

Рисунок 2.2.4.6 – Головна сторінка

Нажимаємо на посилання Houses. Переходимо на сторінку зі всіма будинками (див. рис 2.2.4.7). На даній сторінці є пошукова система, випадаючий

список, де можна переходити на інші сторінки та верифіковані будинки, які можна переглядати і робити обмін. Також у кожного будинка є свою картинка, власник, рейтинг та місце дислокації будинку.

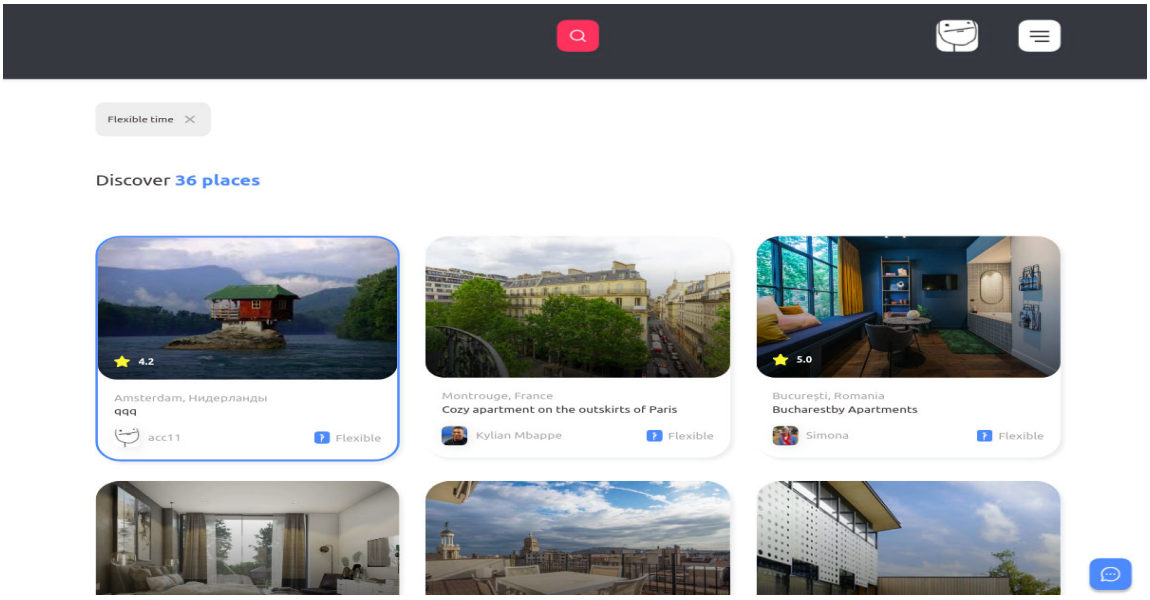

Рисунок 2.2.4.7 – Сторінка з будинками

Нажавши на червону кнопку пошуку відкривається форма з параметрами, по яких можна знайти будинок. На рис. 2.2.4.8 можна побачити пошук по місцю знаходження, даті, кількосьі місць, кількості кімнат та типу будинка.

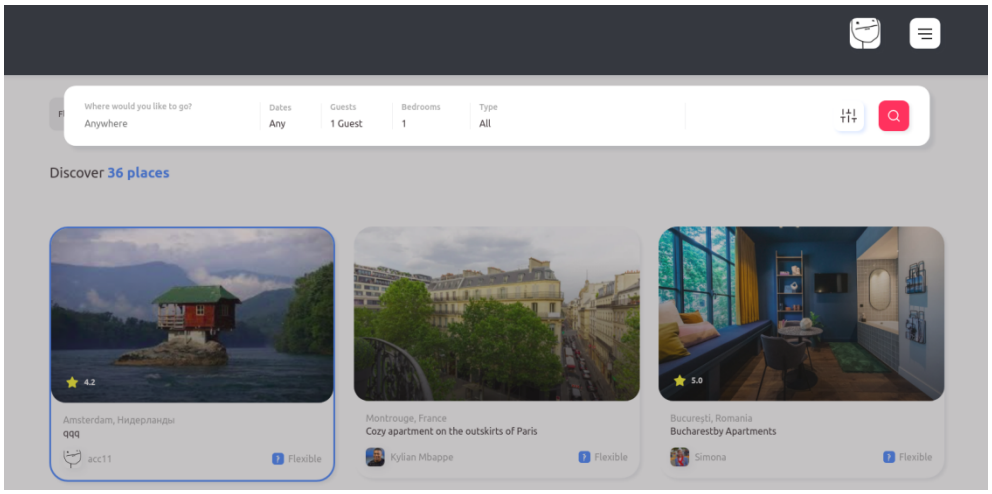

Рисунок 2.2.4.8 – Вкладка для пошуку будинків

Натискаємо на будинок і переходимо на його сторінку (див. рис. 2.2.4.9). На даній сторінці є загальна інфорацій по будинку: картинка з інтерєром, кількість кімнат, на скільки розраховано людей, власник, кількість обмінів, рейтинг та дата обміну.

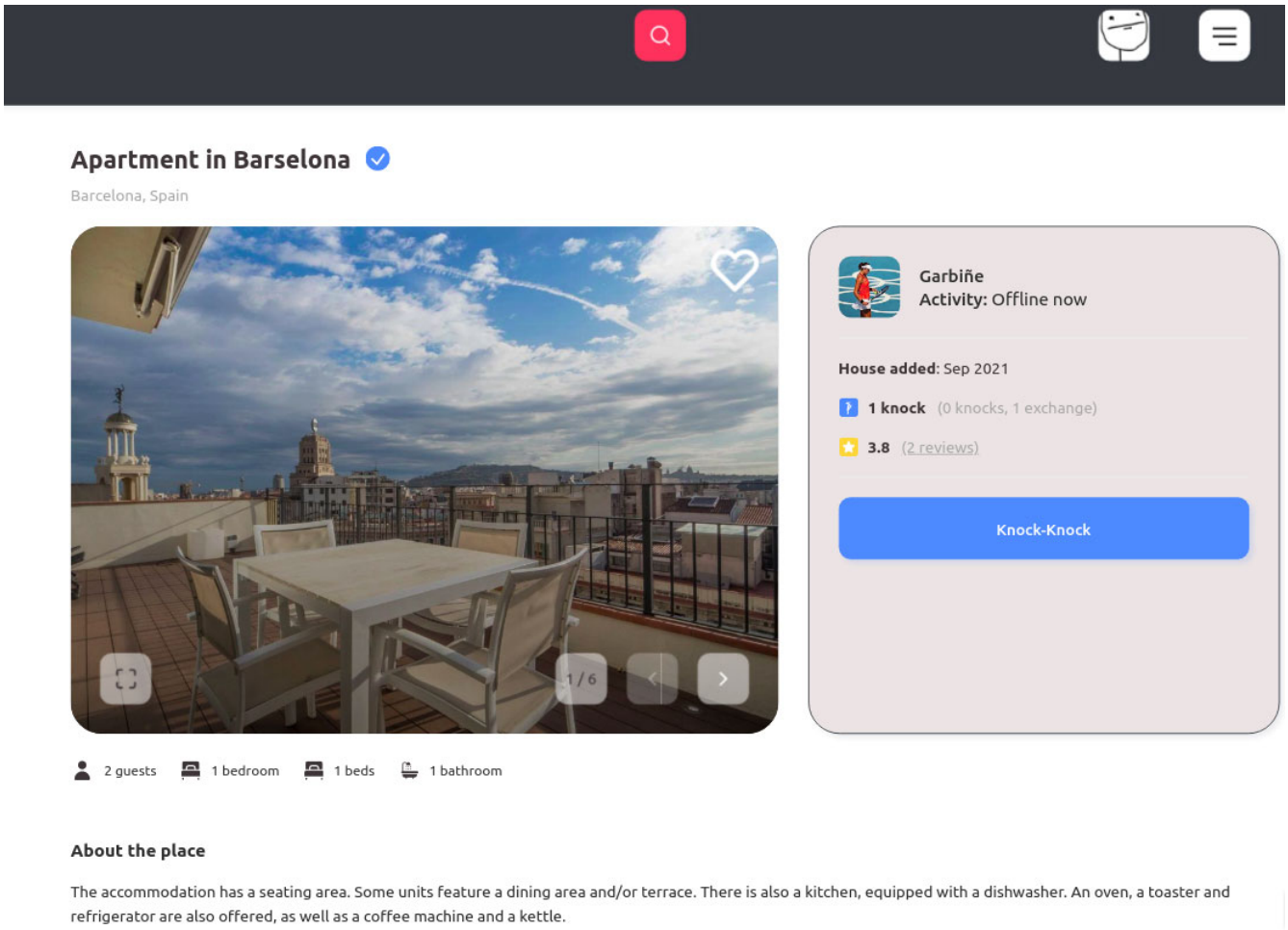

Рисунок 2.2.4.9 – Сторінка з інформацією по будинку

На даній сторінці також є його місце знаходження, яке відображене на Google Maps (див. рис.2.2.4.10). Також можна подивитись додаткові переваги цього будинку.

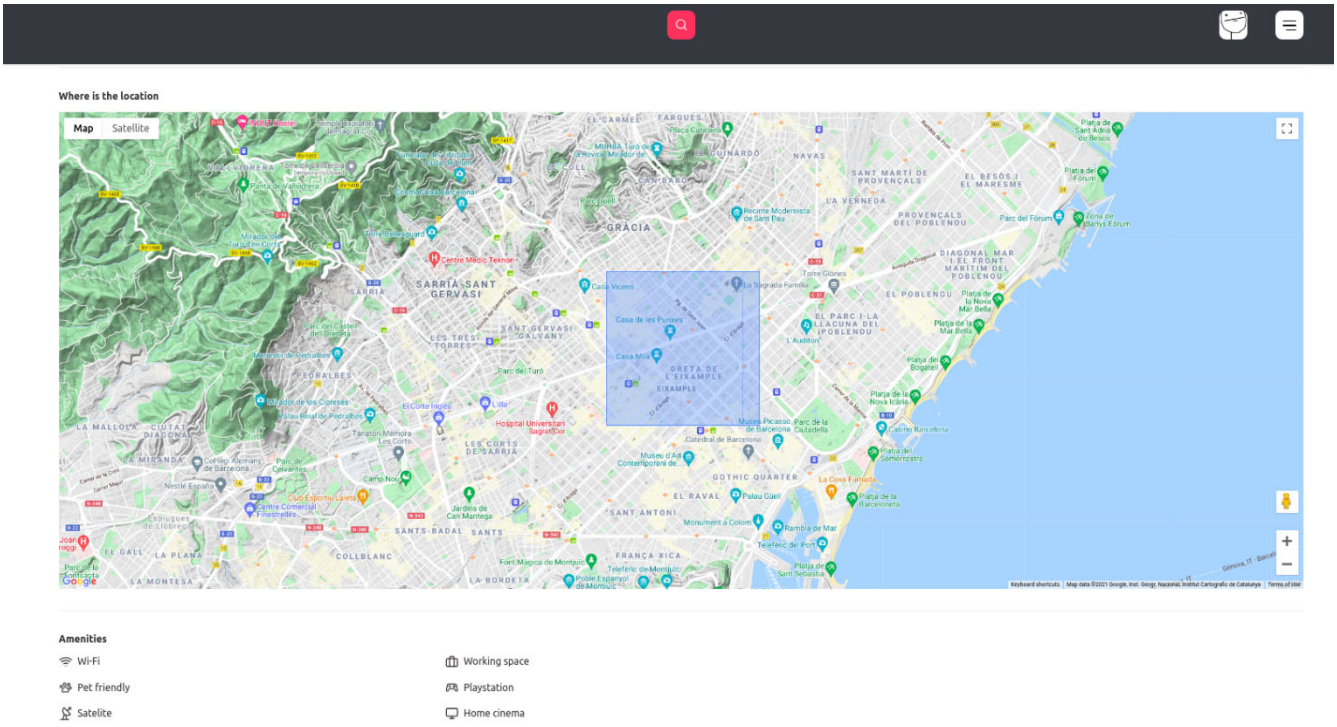

Рисунок 2.2.4.10 – Локація та переваги на сторінці будинку

В кінці даної сторінці можна побачити коментарі та рейтинг даного будинку.(див. рис. 2.2.4.11). Також можна залишити свій коментар до цієї нерухомості.

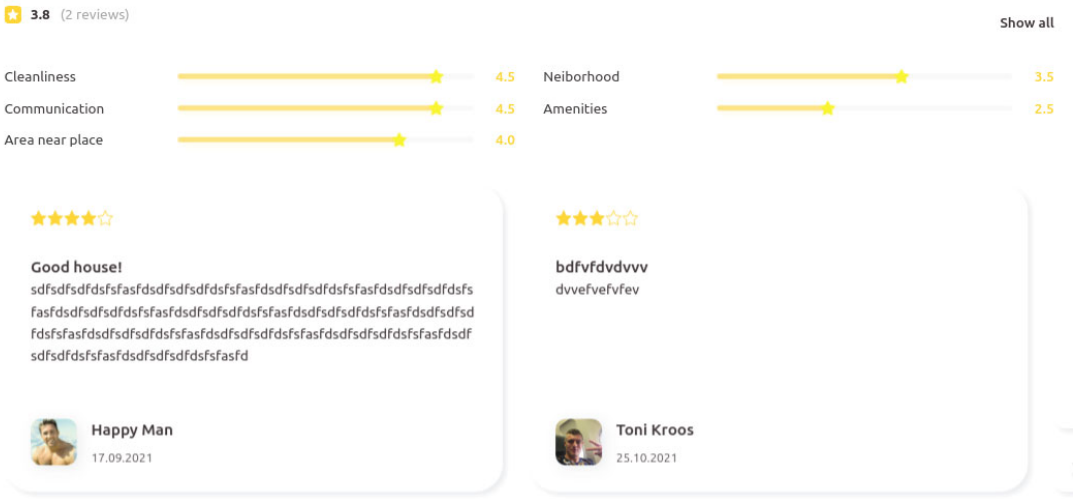

Рисунок 2.2.4.11 – Коментарі та рейтинг на сторінці будинку

В даному сервісі є сторінка з профілем корисувача (див. рис. 2.2.4.12). Тут користувач може бачити свої дані, які були введені при реєстрації. Можна

побачити ім'я, та фамілію користувача, дату створеня аккаунта, мобільний телефон та електронну пошту. Також знаходиться список всіх коментарів, які користувач писав та які залишили інші користувачі. Крім того, на даній сторінці є кнопку для оновлення профілю.

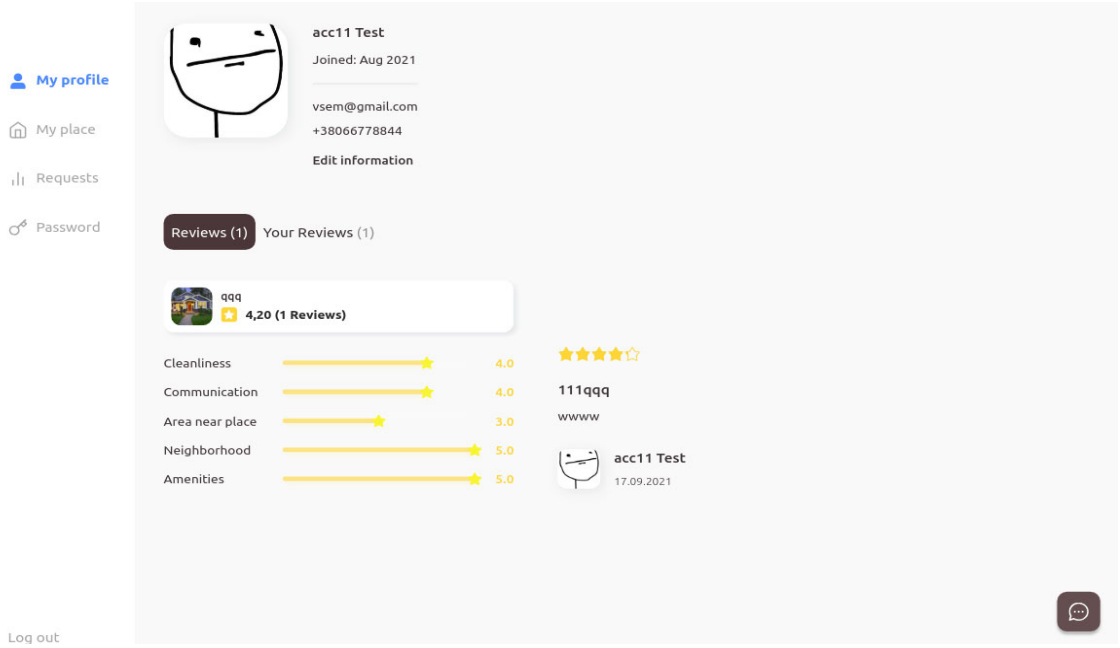

Рисунок 2.2.4.12 – Сторінка з профілем користувача

Нажавши на вкладку з чатами можна буде побачити список чатів (див. рис.2.2.4.13). Чат створюється при створенні запиту на обмін будинками. В чатах є інформація та статус запитів по обміну будинками. Крім того, користувач може переписуватись з іншим користувачем, що полегшує комунікацію між власниками будинків.

??????????????????

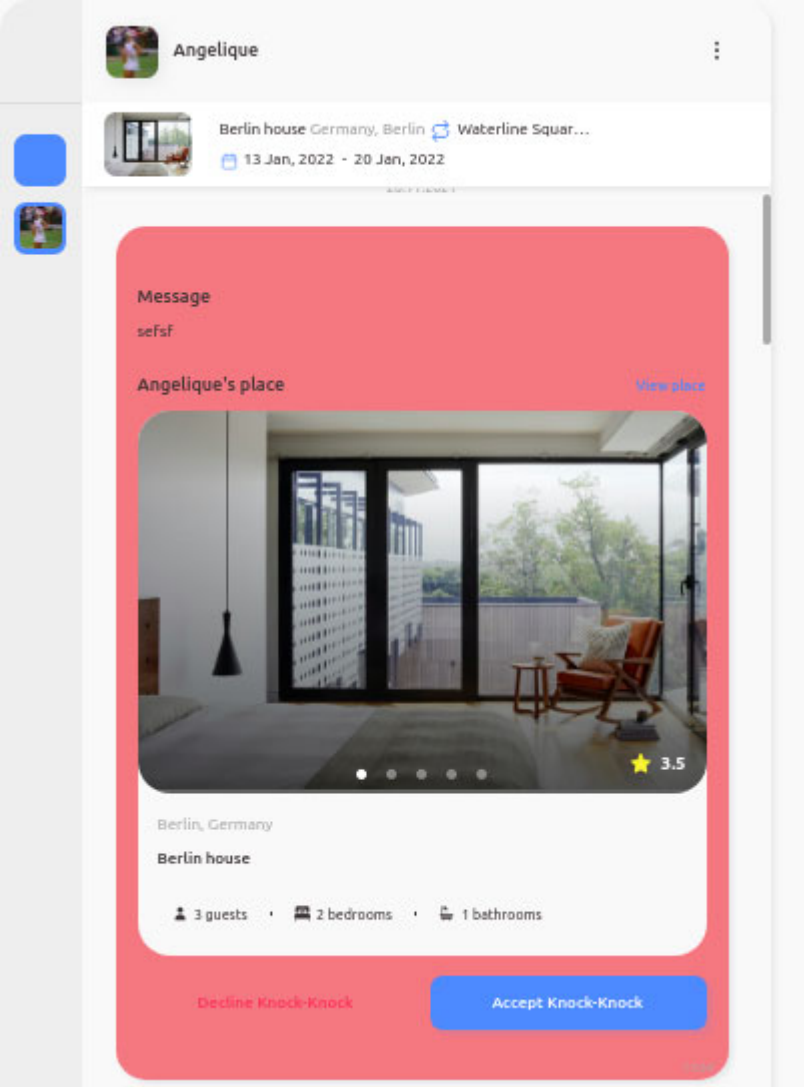

Рисунок 2.2.4.13 – Вкладка з список чатів корисувача

Перейшовши на сторінку My Place отримаємо інформацію по будинку користувача (див. рис. 2.2.4.14). На даній сторіці можна створювати новий будинок для обміну, переглянути існуючий будинок та оновити його інформацію.

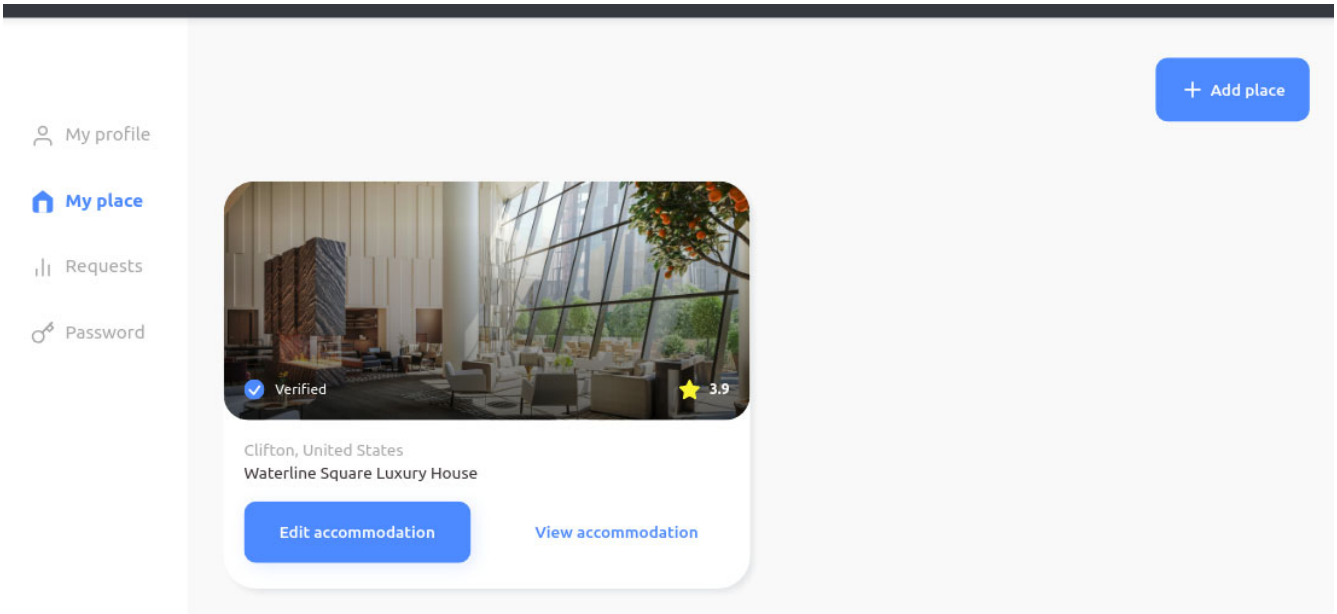

Рисунок 2.2.4.14 – Сторінка для перегляду всіх будинків користувача

Нажавши на кнопку Edit accomodation переходимо на сторінку, де можна оновити інформацію по будинку(див. рис. 2.2.4.15). Тут можно поміняти загальну інформацію по будинку: його вид, тип та площу. Крім того переходячи по вкладках можна місце знаходження, фотографії, документи на будинок та час обміну даного будинку.

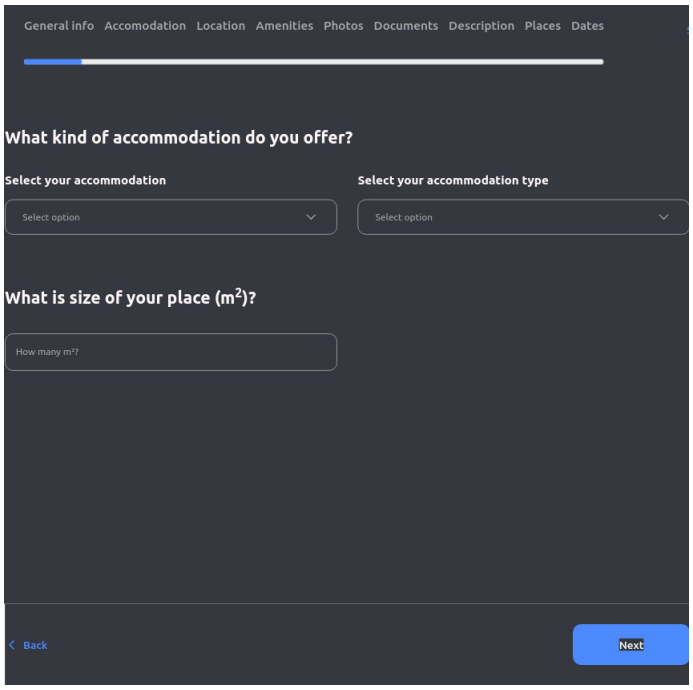

Рисунок 2.2.4.15 – Сторінка для оновлення інформації по будинку

На вкладці Requests знаходяться всі запити на обмін будинками(див.рис.2.2.4.16). Тут можна виділити запити, які зроблені користувачем або отримані запити від інших користувачів по будинку. Також можна переглянути відхилені, успішні або в стані обробки запити. Дана сторінка також містить статистику по загальній кількось запитів в якмх брав участь юзер і загальну кількість переглядів будинку. Нажавши на запит також можно прийняти або відхилити його.

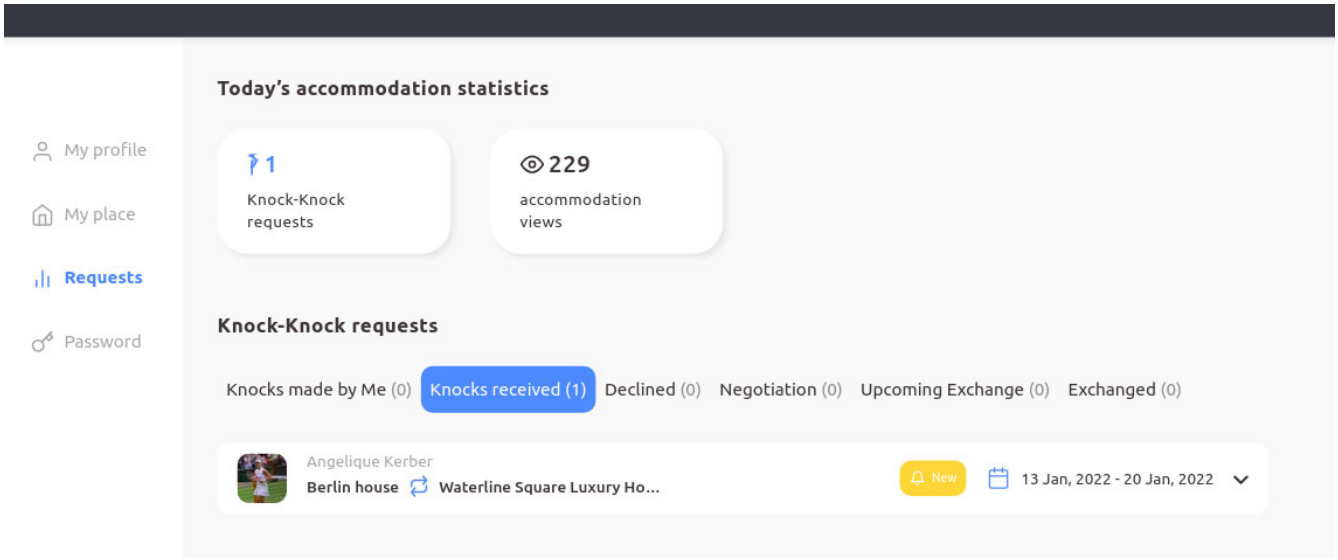

Рисунок 2.2.4.16 – Сторінка всіх запитів, які пов'язані з користувачем

Перейшовши на вкладку Password можна поміняти пароль на новий (див рис.2.2.4.17). Для цього потрібно ввести поточний пароль, змінений пароль та підтвердити змінений пароль. Після цього користувач успішно зможе заходи на сторінку під іншим паролем.

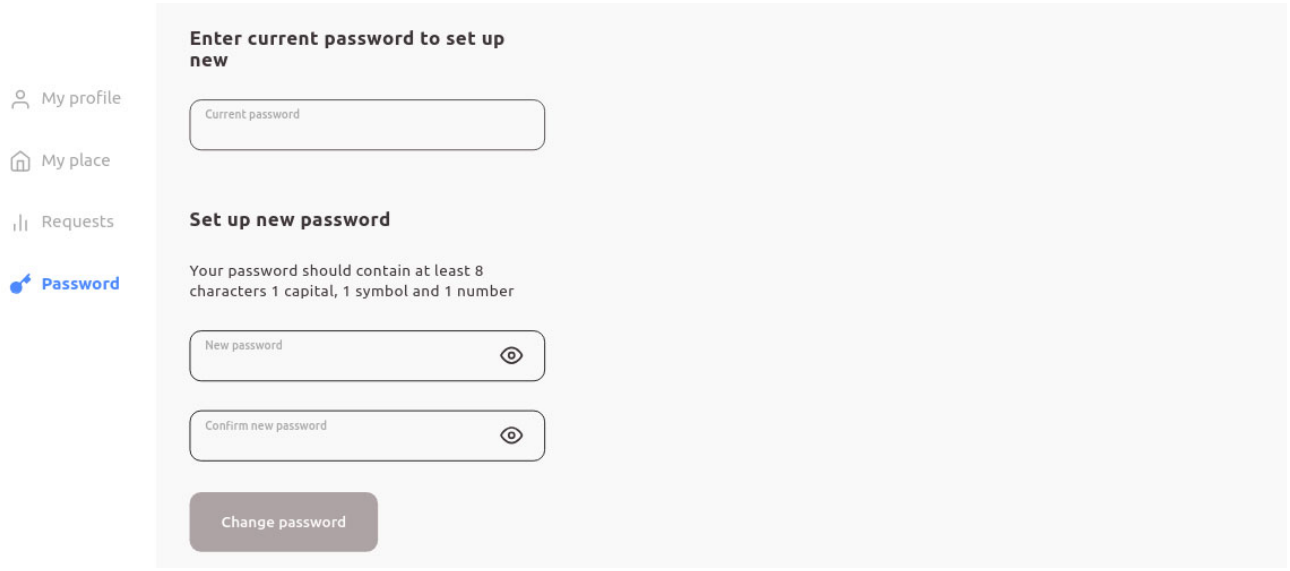

Рисунок 2.2.4.17 – Сторінка для оновлення пароля

На даній платформі є функціонал, який допоможе оновити дані по користувачу(див. рис. 2.2.4.18). На даній вкладці можна оновити ім'я, фамілію, аватарку та мобільний телефон.

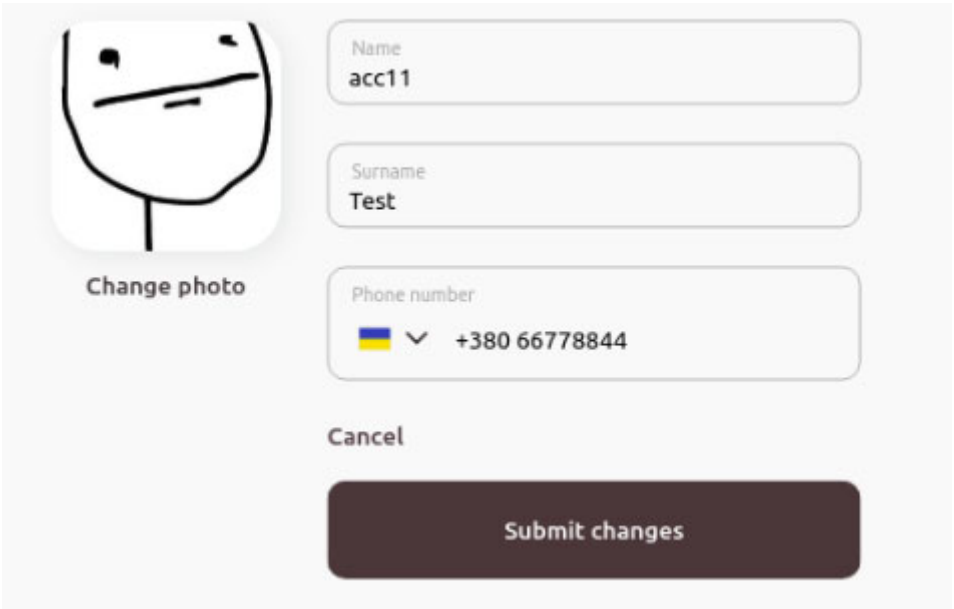

Рисунок 2.2.4.18 – Сторінка для оновлення профіля

3. ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

3.1 Охорона праці

Сучасний розвиток технічного та технологічного стану виробництва передбачає постійну автоматизацію та оптимізацію виробничих процесів. Сьогодні, напевно, важко уявити компанію, господарська діяльність в якій здійснювалась би без використання комп'ютерної техніки. Через масовий характер робіт, що виконуються працівниками за допомогою комп'ютера, законодавством України чітко врегульовано норми та вимоги до працівників, робочих приміщень та безпосередньо охорони праці при роботі з комп'ютером.

Тема дипломного проекту – створення веб-платформи, тобто розробка автономного програмного продукту, виконаного в офісі. Тому потрібно розглянути заходи безпеки у офісному приміщенні. Розмір приміщення становить 200м2, кількість осіб, які працюють у даному приміщенні становить 7 чоловік. На робочому місці кожного працівника є ноутбук Ноутбук Lenovo B570.

Приміщення, в яких проводиться робота з комп'ютером, відповідають проектній документації будинку, погоджений з уповноваженими державними органами. Крім того, були враховані санітарні нормативи освітлення, вимоги до параметрів мікроклімату (температура, відносна вологість), ступеня і сили вібрації, звукового шуму і вогнестійкості приміщення, а також характеристики електромагнітного, ультрафіолетового та інфрачервоного полів. Відповідне приміщення укомплектоване системами центрального опалення, кондиціонування та вентиляції повітря. При установці зазначених систем враховано , що батареї опалення, водопровідні труби, вентиляційні кабелі тощо, надійно сховані під захисними щитками, які перешкоджатимуть можливому потраплянню робітника під напругу.

У кожній кімнаті, де є робочі місця співробітників, що працюють на комп'ютері, є елементи природного та штучного освітлення. На вікнах

встановлено легко регульовані жалюзі чи штори, які дозволять працівникам коригувати рівень освітлення в приміщенні. Розміщено робочі місця таким чином, щоб світло потрапляло на екрани моніторів з півдня чи північного сходу. Природне освітлення в будівлі відповідає будівельним нормам і правилам (ДБН В2.5-28-2018 «Природне і штучне освітлення»). Завдяки цьому запобігаються такі проблеми як: недостатнє освітлення, підвищена яскравість світла та аварійна робота джерел штучного світла. Як наслідок відсутність перенапруження і порушення зору працівників, підвищення працездатності працездатності.

Рівень шуму регулюється санітарними нормами виробничого шуму, ультразвуку та інфразвуку ДСН 3.3.6.037-99. Шум є одним з факторів, що впливає на роботу працівників. У разі надмірного шуму чи вібрації технічного обладнання, є можливість забезпечити працівників анти-вібраційними килимками.

Приміщення перебуває на значній відстані від гучної вулиці та поблизу немає гучних об’єктів. Основними джерелами шуму є:

1) вентилятори в блоках живлення ноутбука та серверу;

2) вентилятори, що охолоджують процесори;

3) робота жорстких дисків

4) шум вулиць та внутрішній шум;

В Україні норми електромагнітної безпеки регламентуються «Державними санітарними нормами і правилами захисту населення від впливу електромагнітного випромінювання», які затверджені МОЗ України (№ 26/35648 від 30.11.2020). Дані норми виконуються. Електромагнітне випромінювання комп’ютера на рівні норми. Звідси слідує відсутність головної болі, негативний вплив на сітківку ока та підвищеного тиску у працівників.

Пожежна безпека регулюється кодексом цивільного захисту України. Джерела пожежної небезпеки офісу є: несправність електропроводки, перегрузка проводки, недотримання заходів пожежної безпеки та легкозаймисті речовини. Факторами даних джерел виступає: коротке замикання, оплавлення ізоляції та загорання. Як наслідок – виникнення пожежі, травмування людей, погіршення або

втрата здоров’я, знищення цінного майна. Дані фактори були враховані відповідно до норм.

Для попередження пожежі використовуються технічні, організаційні та засоби індивідуального захисту. До технічних засобів належить суворе дотримання правил і норм, визначених чинними нормативними документами для приміщень, будівель та об'єктів, технічному оснащенні виробництва, експлуатації чи можливому переобладнанні електромереж, опалення, вентиляції, освітлення. До організаційних заходів можна віднести організацію пожежної охорони на об'єкті, проведення навчань з питань пожежної безпеки (включаючи інструктажі та пожежно-технічні мінімуми), застосування наочних засобів протипожежної пропаганди та агітації, проведення перевірок, оглядів стану пожежної безпеки приміщень. До засобів індивідуально захисту – порошковий вогнегасник.

У робочому приміщенні забороняється: проводити ремонт та технічне обслуговування комп’ютера за робочим місцем працівника; самочинно ремонтувати або намагатись здійснити технічне налагодження комп’ютера без залучення компетентних спеціалістів; складати на робочому місці зайві документи, деталі та предмети, що не потрібні для роботи; використовувати монітори з не чітким зображенням та монітори, у яких наявні поломки екрану; працювати з матричним принтером без антивібраційного покриття та зі знятою кришкою. Допускати до роботи осіб, які не пройшли затверджений на підприємстві курс охорони праці для роботи з комп’ютером, не дозволяється. Для уникнення можливих аварій та замикань, поряд з приміщеннями, де вестиметься робота з комп’ютером (над чи під ними), також не дозволяється проведення робіт, що потребують здійснення надмірно вологих технологічних процесів.

З метою досягнення максимального рівня безпечності і охорони праці при роботі з комп’ютером, приміщення обладнано аптечками першої медичної допомоги, системами автоматичної пожежної сигналізації і вогнегасниками.

Отже, мікрокліматичні параметри офісу відповідають встановленим нормам – оптимальна температура, дія шумів та електромагнітного поля на організм людини.

3.2 Підвищення стійкості обєкта господарської діяльності у воєнний час

Під стійкістю роботи промислового об’єкта (об’єкта господарювання любої форми власності) розуміють здатність його в умовах надзвичайних ситуацій мирного і воєнного часу випускати продукцію в запланованому обсязі й номенклатурі, а при одержанні слабких і середніх руйнувань, порушенні зв’язків по кооперації і постачанням відновлювати виробництво в мінімальний термін. Здатність об’єкта народного господарства випускати продукцію залежить від захисту і нормального функціонування чотирьох основних елементів сучасного виробництва, якими є:

– виробничий персонал (робітники та службовці);

– будинки і споруди з технологічним устаткуванням;

– система постачання енергією, водою, паливом, устаткуванням і ремонтною базою;

– система виробничих і кооперативних зв’язків з іншими об’єктами.

Тому стійкість роботи об’єктів і галузі народного господарства в цілому в умовах надзвичайних ситуацій визначається наступними факторами:

– надійністю захисту робітників та службовців від усіх вражаючих факторів зброї масового ураження;

– здатністю інженерно-технічного комплексу (ІТК) об’єкта протистояти вражаючим факторам ядерного вибуху;

– надійністю системи постачання об’єкта всім необхідним для виробництва продукції (сировиною, паливом, що комплектують виробами, електроенергією, водою, газом тощо.);

– захищеності об’єкта від вторинних вражаючих факторів (пожеж, вибухів, затоплень, зараження місцевості отруйними і сильнодіючими отруйними речовинами);

– стійкістю і безперервністю керування виробництвом і цивільною обороною;

– підготовленість об’єкта до проведення рятувальних та інших невідкладних робіт і робіт з відновленням порушеного виробництва. Перераховані фактори визначають собою й основні, загальні для всіх об’єктів господарювання, шляхи підвищення стійкості роботи в надзвичайних ситуаціях, а саме:

– забезпечення надійного захисту робітників та службовців від вражаючих факторів зброї масового ураження;

– захист основних виробничих фондів від вражаючих факторів, у тому числі й від вторинних;

– підвищення надійності й оперативності керування виробництвом;

– забезпечення стійкості постачання всім необхідним для випуску запланованої на час надзвичайних ситуацій продукцією;

– підготовка до відновлення порушеного виробництва.

Захист робітників та службовців в умовах воєнного часу це найголовніша задача по підвищенню стійкості роботи об’єкта господарювання. Робітники й службовці – головна продуктивна сила і тому стійкість економіки визначається, насамперед, здатністю захистити і зберегти цю силу. Військові конфлікти супроводжуються руйнуванням будинків, споруджень і знищенням основної продуктивної сили – працюючого населення. Тому серед усіх задач по підвищенню стійкості роботи об’єктів народного господарства основною є задача завчасного вживання заходів по забезпеченню захисту робітників та службовців і членів їхніх родин. Захист робітників та службовців від зброї масової поразки в сучасних умовах здійснюється трьома основними способами:

– укриття людей у захисних спорудженнях (сховищах, протирадіаційних укриттях);

– проведення евакуації робітників, службовців і членів їхніх родин;

– використання засобів індивідуального захисту, а також проведенням заходів щодо протирадіаційного, протихімічного і протибактеріологічного захисту з урахуванням конкретних обставин.

Варто також підкреслити, що найважливішою умовою успішного вирішення задачі захисту людей є навчання їх правилам дії по сигналах оповіщення

цивільного захисту, застосуванню способів і засобів захисту, наданню самодопомоги і взаємодопомоги, діям у складі формувань ЦЗ(цивільного захисту).

Основу діяльності керівника виробництва – начальника ЦЗ, а також його штабу складає якісне та професійне керування підлеглими йому структурами в організації їхньої дії і напрямку зусиль на своєчасне й успішне виконання виробничих завдань. Тому, забезпечення надійності й оперативності керування є важливою ланкою в підвищенні стійкості роботи об’єкта, в умовах швидко мінливої обстановки воєнного часу і надзвичайних ситуацій. Надійність і оперативність керування досягається створенням на об’єкті стійкої системи керування, високої підготовки керівного і командноначальницького складу ЦЗ до виконання покладених функціональних обов’язків, своєчасним прийняттям рішень і постановкою задач підлеглим відповідно до обставин, що складаються.

Забезпечення стійкого постачання підприємств є також одним з пунктів підвищення стійкості. Для виробництва продукції необхідні: електроенергія, вода, паливо, сировина, матеріали й інші матеріально-технічні засоби. Забезпечення підприємств цими ресурсами багато в чому визначає можливість нормального їхнього функціонування в умовах воєнного часу. Це досягається проведенням таких заходів, що сприяють підвищенню не ураженості комунально-енергетичних мереж, транспортних комунікацій і джерел постачання, надійному захисту необхідних запасів палива, сировини, напівфабрикатів, що комплектують, виробів тощо.

Далі йде підготовка до відновлення порушеного виробництва. Можливості вражаючою дії сучасних видів зброї такі, що забезпечити абсолютний захист від нього об’єктів і споруд практично неможливо. Вони можуть одержати той чи інший ступінь руйнування. У цих умовах задача зводиться до того, щоб у випадку слабких і середніх руйнувань на об’єкті відбудувати об’єкт і відновити випуск необхідної продукції в мінімальний термін. Підвищення, стійкості роботи об’єкта народного господарства у воєнний час і в умовах надзвичайних ситуацій досягається завчасним проведенням комплексу інженерно-технологічних, технологічних і організаційних заходів, спрямованих на максимальне зниження

впливу вражаючих факторів зброї масового ураження і створення умов для швидкої ліквідації наслідків. Підготовка до відновлення порушеного виробництва здійснюється завчасно і передбачає планування відбудовних робіт по декількох варіантах: підготовку ремонтних бригад, створення необхідного запасу матеріалів і устаткування, надійний його захист. Потрібно проводити інженерно-технічні, технологічні та органцізаційні заходи.

Інженерно-технічні заходи, як правило, включають комплекс робіт, що забезпечують підвищення стійкості виробничих будинків і споруджень, верстатного і технологічного устаткування, комунально-енергетичних систем.

Технологічні заходи забезпечують підвищення стійкості роботи об’єкта шляхом зміни технологічного процесу, що сприяє прискоренню виробництва продукції і виключає можливість утворення вторинних вражаючих факторів.

Організаційні заходи передбачають розробку і планування дій керівного, командно-начальницького складу, штабу, служб і формувань ЦЗ при захисті робітників та службовців підприємства й інших невідкладних робіт, відновленні виробництва, а також по випуску продукції на збережених потужностях.

Крім того потрібно під час проектування потрібно враховувати вимоги до планування міста, в якому знаходяться обєкти господарської діяльності. Розміщення об’єктів господарювання повинне здійснюватися з урахуванням зон можливих руйнувань. Нові важливі промислові підприємства, основні склади і бази повинні розміщатися за межами зони можливих руйнувань. Поза зонами можливих сильних руйнувань повинні розміщатися: бази і склади з продовольчими і промисловими товарами першої необхідності; базисні склади легкозаймистих і паливних матеріалів; головні спорудження системи водопостачання; насосні і компресорні станції магістральних трубопроводів; міжміські кабельні магістральні лінії зв’язку й інші важливі об’єкти. У зоні можливих сильних руйнувань допускається розміщати: комунальні гаражі, тролейбусні депо, трамвайні парки, склади поточного постачання, підземні магістральні трубопроводи, одну з груп головних споруджень системи водопостачання й інших підприємств по обслуговуванню населення міста.

Магістральні вулиці повинні мати перетинання з іншими магістралями, автомобільними і залізничними дорогами в різних рівнях. Внутріміська транспортна мережа повинна забезпечити надійне сполучення між житловими і промисловими районами, вільний вихід до магістралей, що ведуть за межі міста, а також найбільш короткий і зручний зв'язок центра міста, міських промислових і житлових районів із залізничними й автобусними вокзалами й аеропортами. По території міста та прилеглими районами повинні бути дублюючі шляхи сполучення.

Міжміські автомобільні дороги повинні прокладатися в обхід міста. Навколо великих міст доцільно будувати кільцеві дороги і сполучні обхідні шляхи. Це зменшить забруднення повітряного басейну в границях міста від автомобільного транспорту і не порушить транспортних зв'язків у випадку поразки міста ядерною зброєю.

Створення лісопаркового поясу навколо міста має важливе значення для організації масового відпочинку населення, а у воєнний час для розміщення робітників і службовців підприємств і евакуйованого населення. З цією метою в лісопарковому поясі за межами зони можливих руйнуванні повинне вестися будівництво туристичних і спортивних баз, пансіонатів, будинків відпочинку, санаторіїв, що сприяє розширенню житлового фонду в заміській зоні. Тут також варто розвивати дорожню мережу з твердим покриттям, електропостачання, водопостачання і зв'язок. Ці міри добре сполучаються з розвитком зон відпочинку для населення і забезпечення нормальних умов життя в районах заміської зони у воєнний час.

ВИСНОВКИ

В ході магістерської роботи було розроблено веб-платформу для обміну будинками агенції нерухомості "Еліт Дім" на базі фреймворку Spring. Перед початком написання було вибрано середовище розробки, мова програмування та фреймворки. Були враховані всі технічні аспекти проблеми даного проєкту.

Під час написання роботи було проведено проектування веб-платформи для обміну будинками. Розроблена модель предметної області. Була описана проблематика даного проєкту. Виходячи із проблематики були складені цілі та задачі. Крім того, була проведена розробка бізнес моделі, а саме була створена діаграма варіантів використання. Ключовим в даній роботі було проєктування архітектури. Були описані основні класи, методи, компоненти та створено EER діаграму.

Після проектування був написаний функціонал та GUI для веб-платформи. Особливістю даної роботи є використання Criteria API для написання пошукової системи будинків. Завдяки цьомуможна писати запити без виконання необробленого SQL. Крім того, дає нам деякий об’єктно-орієнтований контроль над запитами, що є однією з головних функцій Hibernate. Також API Criteria дозволила програмно створити об’єкт запиту критеріїв, де пізніше можна застосовувати різні види фільтрації та логічних умов.

Захищеність API була забезпечена за рахунок метода JWT(Json Web Token). Були прописані конфігурації та створений JWT фільтер для того щоб API з ролями працювали коректно і були захищеними.

В ході тестування веб-платформи були перевірені методи авторизації, рєстрація користувача, перегляд будинків, пошук будинків, відправка запиту на обмін будинками.

В подальшому планується створити CRM систему для даної веб-платформи.

Планується добавити додаткову інформацію для будинку та графіки по різних параметрах. Також крім чатів буде добавлено відео дзвінок для покращення комунікації між користувачами даної веб-платформи.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Spring Framework – [Електронний ресурс] – Режим доступу до ресурсу: http://spring-projects.ru/projects/spring-framework/.0
2. Spring Framework и Spring Boot– [Електронний ресурс] – Режим доступу до ресурсу: https://topjava.ru/blog/spring-framework-vs-spring-boot-differences.
3. MSDN – Основи HTML – [Електронний ресурс] – Режим доступу до ресурсу: https://msdn.microsoft.com/ru-ru/library/wx4hctt4.aspx.
4. MSDN – Working with CSS Overview – [Електронний ресурс] – Режим доступу до ресурсу: https://msdn.microsoft.com/en-us/library/bb398931(v=vs.110).
5. Про проєкт | Vue.js – [Електронний ресурс] – Режим доступу до ресурсу: https://habr.com/ru/post/329452/.
6. Берназ-Лукавецька, Олена Михайлівна, and Елена Михайловна Берназ-Лукавецкая. "Обмін житла (жилих приміщень): цивілістичні аспекти." (2010).
7. Микросервисы со Spring Boot – [Електронний ресурс] – Режим доступу до ресурсу: https://habr.com/ru/post/484130/
8. Spring Security – [Електронний ресурс] – Режим доступу до ресурсу: http://spring-projects.ru/projects/spring-security/
9. Java Persistence API (JPA) – [Електронний ресурс] – Режим доступу до ресурсу: https://javastudy.ru/frameworks/spring/spring-data/
10. Spring Data JPA — пример приложения – [Електронний ресурс] – Режим доступу до ресурсу: https://javastudy.ru/spring-data-jpa/spring-data-jpa-helloworld/
11. Building REST services with Spring – [Електронний ресурс] – Режим доступу до ресурсу: https://spring.io/guides/tutorials/rest/

12. Обзор REST – [Електронний ресурс] – Режим доступу до ресурсу: https://javarush.ru/groups/posts/2488-obzor-rest-chastjh-3-sozdanie-restful-servisa-na-spring-boot
13. Документація Spring Framework. 1. Introduction to Spring Framework [Електронний ресурс] — Режим доступу: https://docs.spring.io/spring/docs/3.0.0.M4/reference/html/ch01s02.html
14. Документація Spring Framework. 6. The Web [Електронний ресурс] — Режим доступу: https://docs.spring.io/spring/ docs/current/ spring-frameworkreference/html/mvc.html
15. Документація Spring Data Framework [Електронний ресурс] — Режим доступу:https://docs.spring.io/spring-data/jpa/docs/current /reference/html/
16. Hibernate Framework [Електронний ресурс] — Режим доступу: http://hibernate.org/orm/
17. Документація Maven Framework [Електронний ресурс] — Режим доступу: http://maven.apache. org/guides/introduction /introduction-to-thelifecycle.html#Lifecycle_Reference
18. Пауэлл Т.А. Полное руководство по HTML / Т.А. Пауэлл. — М.: Мир, 2001. — 912 с. 13. Кисленко Н.П. HTML. Самое необходимое. / Н.П. Кисленко — СПб.: БХВПетербург, 2008. — 352 с.
19. Справочник CSS [Електронний ресурс] — Режим доступу: http://htmlbook.ru/css

ДОДАТКИ