МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет імені Петра Могили

Факультет комп'ютерних наук

Кафедра інженерії програмного забезпечення

ДОПУЩЕНО ДО ЗАХИСТУ

Завідувач кафедри, канд. техн. наук,
доцент Є. О. Давиденко
«___»____2023 р.

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

ВЕБЗАСТОСУНОК ПОШУКУ КОВОРКІНГУ

Спеціальність «Інженерія програмного забезпечення»

121 – КРБ.1 – 408. 21910825

Студент _____ Г. Л. Чернигін
«___»_____2023 р.

Керівник канд. техн. наук, доцент _____ Є. О. Давиденко
«___»_____2023 р.

Консультант канд. техн. наук, доцент _____ А. О. Алексєєва
«___»_____2023 р.

Миколаїв 2023

Завдання на виконання кваліфікаційної роботи

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Чорноморський національний університет імені Петра Могили
Факультет комп'ютерних наук
Кафедра інженерії програмного забезпечення

ЗАТВЕРДЖУЮ

Зав. кафедри _____ Є. О. Давиденко

«___» _____ 2023 р.

ЗАВДАННЯ
на виконання кваліфікаційної роботи бакалавра

Видано студенту групи 408 факультету комп'ютерних наук

Чернигіну_Глібу_Леонідовичу_____
(прізвище, ім'я, по батькові студента)

1. Тема кваліфікаційної роботи
Вебзастосунок_пошуку_коворкінгу_____
_____

Затверджена наказом по ЧНУ від «17» _березня____ 2023 р. № _60_____

2. Строк представлення кваліфікаційної роботи «___» _____ 2023 р.

3. Очікуваний результат роботи та початкові дані, якщо такі потрібні
Очікуваним_результатом_є_вебзастосунок_пошуку_коворкінгу._____
_____

4. Перелік питань, що підлягають розробці
- дослідження предметної області та аналіз існуючих аналогів ПЗ;
- формування специфікації вимог до ПЗ;
- визначення архітектури ПЗ;
- моделювання та проєктування ПЗ;

- розробка ПЗ;
- здійснення тестування роботи ПЗ;
- проведення аналізу результатів розробки;

5. Перелік графічних матеріалів

Презентація_____
_____

6. Завдання до спеціальної частини

Аналіз охорони праці на робочих місцях фахівців з інформаційних технологій_____
_____

7. Консультанти:

Консультант	Кафедра (організація)	Частина роботи
Алексєєва А. О.	Кафедра екології	Спеціальна частина з охорони праці

Керівник роботи канд. техн. наук, доцент Давиденко Євген Олександрович___

(посада, прізвище, ім'я, по батькові)

_____

(підпис)

Завдання прийнято до виконання

___Чернигін_Гліб_Леонідович_____

(прізвище, ім'я, по батькові студента)

_____

(підпис)

Дата видачі завдання «17» _березня_____ 2023 р.

КАЛЕНДАРНИЙ ПЛАН
виконання кваліфікаційної роботи

Тема:_____Вебзастосунок_пошуку_коворкінгу_____
_____

№	Найменування роботи	Початок	Закінчення	Примітки
1.	Розробка та затвердження завдання на виконання КРБ	17.03.2023 р	17.03.2023 р	виконано
2.	Огляд літератури за темою роботи	20.03.2023 р.	20.03.2023 р.	виконано
3.	Складання календарного плану КРБ	21.03.2023 р.	21.03.2023 р.	виконано
4.	Аналіз предметної області	22.03.2023 р.	22.03.2023 р.	виконано
5.	Розробка проєктних рішень	23.03.2023 р.	23.03.2023 р.	виконано
6.	Моделювання та конструювання ПЗ	24.03.2023	27.03.2023 р.	виконано
7.	Кодування, тестування та апробація розробленого ПЗ, аналіз результатів тестування, розробка керівництва користувача	28.03.2023 р.	28.04.2023 р.	виконано
8.	Розробка спеціальної частини з охорони праці	1.05.2023 р.	25.05.2023 р.	виконано
9.	Відгук керівника КРБ	26.05.2023 р.	26.05.2023 р.	виконано
10.	Оформлення КРБ та презентації	29.05.2023 р.	2.06.2023 р.	виконано
11.	Попередній захист	06.06.2023 р.	06.06.2023 р.	виконано
12.	Рецензування	20.06.2023 р.	21.06.2023 р.	виконано
13.	Завершення оформлення КРБ та презентації	22.06.2023 р.	23.06.2023 р.	виконано
14.	Захист кваліфікаційної роботи	26.06.2023 р.	26.06.2023 р.	виконано

Розробив студент _____Чернигін Гліб Леонідович_____ _____

(прізвище, ім'я, по батькові) *(підпис)*

«__» _____ 2023 р.

Керівник роботи __канд. техн. наук, доцент Давиденко Є. О.__ _____

(*посада, прізвище, ім'я, по батькові)* *(підпис)*

«__» _____ 2023 р.

АНОТАЦІЯ

до кваліфікаційної роботи бакалавра

«Вебзастосунок пошуку коворкінгу»

Студент 408 гр.: Чернигін Гліб Леонідович

Керівник: зав. кафедри ІПЗ, канд. техн. наук, доцент Давиденко Є.О.

Дана робота присвячена розробці програмного забезпечення для пошуку коворкінгу базуючись на критеріях оснащення приміщення та використовуючи можливості Google Maps.

Об’єкт: процес пошуку закладу з необхідним переліком умов для роботи.

Предмет: програмні засоби створення застосунку для пошуку коворкінгу.

Метою роботи є підвищення зручності пошуку місця для дистанційної роботи в умовах нестабільного електропостачання шляхом розробки інформаційного агрегатору з переліком таких місць.

Пояснювальна записка кваліфікаційної роботи бакалавра складається з вступу, трьох розділів, висновків та додатків.

У вступі визначається актуальність теми, що досліджується, описується поставлена задача, а також мета, предмет та об’єкт дослідження.

У першому розділі описується огляд існуючих застосунків, які мають функції створеного програмного забезпечення, а саме: відображення коворкінгів на мапі, можливість перегляду детальної інформації про коворкінг, пошук та фільтр коворкінгів за критеріями та можливість додати власний коворкінг. Крім цього описуються головні проблеми, що вирішує розроблений програмний застосунок.

У другому розділі визначаються основні функції та можливості застосунку: відображення та кластеризація коворкінгів на мапі, інтеграція з

Google Maps для отримання детальної інформації про місце, підтвердження власності певного місця та можливість додати додатку інформацію. Розділ визначає архітектуру та структуру застосунку, наводить блок-схеми роботи та UML діаграми варіантів використання готового програмного продукту.

У третьому розділі описується програмний код, бібліотеки та фреймворки, що були використані під час розробки вебзастосунку пошуку коворкінгу.

У четвертому розділі проводиться тестування розробленого програмного забезпечення, аналіз роботи та отриманих даних. Також розділ наводить скріншоти фінального інтерфейсу застосунку та демонстрації роботи.

У висновках проводиться аналіз виконаного об’єму робіт та отриманих у ході виконання результатів.

Кваліфікаційна робота містить **61** сторінку основної частини, **4** розділи, **37** рисунків, **6** таблиць, **21** джерело в переліку посилань

Ключові слова: *коворкінг, пошук коворкінгів, агрегація даних, геопросторові дані, створення вебзастосунку.*

ABSTRACT

of the Bachelor’s Thesis

«Webapp for coworking search»

Student of group 408: Hlib Chernyhin Leonidovych

Supervisor: Deputy Dean, Associate Professor of the Department of software engineering Davydenko Ye.O

This work is devoted to the development of software for finding a co-working space based on the criteria of building equipment and using the capabilities of Google Maps.

Object: the process of finding an institution with the necessary list of conditions for work.

Subject: software tools for creating an application for finding a co-working space.

The purpose of the work is to increase the convenience of finding a place for remote work in conditions of unstable electricity supply by developing an information aggregator with a list of such places.

The explanatory note of the bachelor's thesis consists of an introduction, three sections, conclusions and appendices.

The introduction determines the relevance of the researched topic, describes the task, as well as the goal, subject and object of the research.

The first section describes an overview of existing applications that have the functions of the created software, namely: displaying coworking spaces on a map, the ability to view detailed information about a coworking space, search and filter coworking spaces by criteria, and the ability to add your own coworking space. In addition, the main problems solved by the developed software application are described.

The second section defines the main functions and capabilities of the application: display and clustering of coworking spaces on the map, integration with Google Maps to obtain detailed information about the place, confirmation of ownership of a certain place and the ability to add information to the application. The section defines the architecture and structure of the application, provides flowcharts and UML diagrams of the use cases of the finished software product.

The third chapter describes the software code, libraries and frameworks that were used during the development of the coworking search web application.

In the fourth section, testing of the developed software, analysis of work and received data is carried out. The section also provides screenshots of the final application interface and work demonstrations.

In the conclusions, an analysis of the volume of work performed and the results obtained in the course of implementation is carried out.

The qualification paper contains **61** pages of the main part, **4** sections, **37** figures, **6** tables, **21** sources in the list of references,

Keywords: *coworking, coworking search, data aggregation, geospatial data, software development.*

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

БД	–	база даних
ОС	–	операційна система
ПЗ	–	програмне забезпечення
ПК	–	персональний комп’ютер
API	–	application programming interface
JS	–	JavaScript
ORM	–	Object-relational mapping
PHP	–	hypertext preprocessor
REST	–	Representational state transfer
UI	–	user interface
UML	–	unified modeling language
UX	–	user experience

ВСТУП

Робота є невід'ємною частиною життя кожної людини. Середньостатистична людина витрачає майже 1 рік свого життя тільки на те, щоб добратися до місця роботи. Світова пандемія COVID-19 показала усьому світові, що для того, щоб працювати – не обов'язково ходити в офіс і навіть більше – робітник не повинен знаходитися у тому самому місці чи навіть країні. Але, підхід до дистанційної роботи має і свої недоліки – уся відповідальність на забезпечені стабільності роботи лежить на робітнику.

В Україні ця проблема розвинулась ще більше після початку війни – через масовані обстріли кричної інфраструктури мільйони українців залишаються без стабільного електропостачання, а як наслідок – без можливості працювати. Рятувальним кругом у боротьбі з темрявою стали генератори, акумулятори, павербанки та старлінки. Багато ІТ-компаній знову відкрили офіси для тих, у кого є проблеми з електроенергією, але більшість людей залишається працювати віддалено та вимушені шукати місце для роботи у кафе, коворкінгах та інших закладах

Актуальність теми кваліфікаційної роботи бакалавра зумовлена тенденцією пошуку місця для роботи зі стабільним доступом до мережі інтернету та можливістю підзарядити гаджети.

Об'єкт роботи: процес пошуку закладу з необхідним переліком умов для роботи.

Предмет роботи: програмні засоби створення застосунку для пошуку коворкінгу.

Мета: підвищення зручності пошуку місця для дистанційної роботи в умовах нестабільного електропостачання шляхом розробки інформаційного агрегатору з переліком таких місць.

Для досягнення визначеної мети необхідно вирішити наступні **завдання:**

- аналіз існуючих аналогів ПЗ;
- визначення функціоналу застосунку;
- створення блок-схем та діаграм роботи застосунку;
- збір та аналіз методичних рекомендацій;
- розробка frontend-частини вебзастосунку на базі фреймворку Angular;
- розробка backend-частини вебзастосунку на базі фреймворку ASP.NET.

Сфера застосування: вебзастосунок пошуку коворкінгу можна використовувати під час процесу пошуку місця для роботи. Цільова аудиторія застосунку – люди, що працюють віддалено: програмісти, дизайнери, вчителі тощо.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Огляд застосунків-аналогів для пошуку коворкінгу

Аналіз ринку та аналогів ПЗ – це важливий етап перед створенням програмного рішення для пошуку коворкінгів. Він допомагає зрозуміти, які основні функції та можливості повинен мати додаток, а також які особливості повинен мати коворкінг для залучення користувачів. На українському ринку існує кілька додатків для пошуку коворкінгів, серед яких можна виділити: SvitloSpot, MatchOffice та Асоціацію коворкінгів України.

SviltoSpot

Застосунок створений для швидкого пошуку місця для праці зі світлом та інтернетом поряд. Основними недоліками застосунку є невеликий функціонал фільтрації закладів, а також ручне додавання та оновлення місць праці.

Таблиця 1.1 – Опис системи SvitloSpot

Назва	SvitloSpot
Розробник	CODY Ukraine
Архітектура	3tier application
Мова реалізації	JavaScript
Функції	1. Пошук місця праці за локацією. 2. Можливість зв’язку з власником закладу. 3. Можливість додати локацію. 4. Віджет з кількістю локації за областями.
Переваги	1. Зручний та зрозумілий інтерфейс. 2. Наявність додаткових характеристик місця.
Недоліки	1. Невеликий набір Фільтрів. 2. Управління місцями роботи через google форму.
Вебсайт	https://svitlospot.com.ua

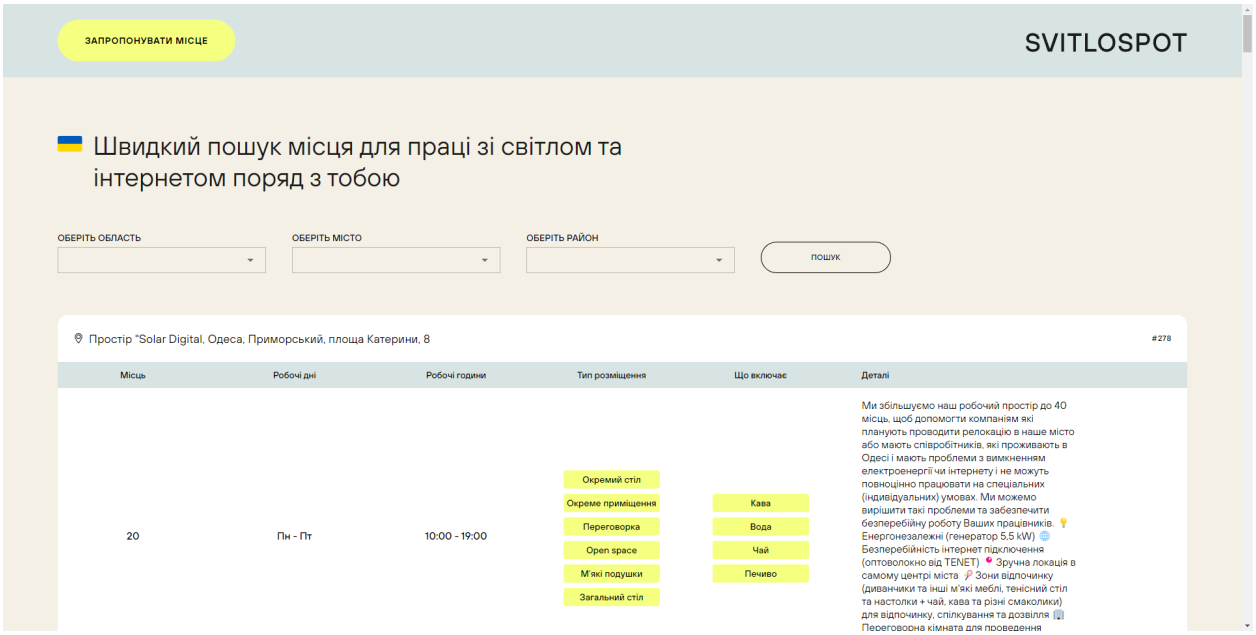

Рисунок 1.1 – Вигляд застосунку SvitloSpot

MatchOffice

Онлайн-платформа для оренди офісних приміщень в Україні. Цей сервіс надає можливість користувачам швидко та зручно знайти приміщення для своєї компанії в будь-якому місті України.

MatchOffice працює як посередник між орендодавцями та орендарями. На сайті можна знайти різноманітні офісні приміщення, такі як окремі кабінети, віртуальні офіси, коворкінги, орендувати конференц-зали та багато іншого.

Користувачі можуть шукати офіси за містами, вказувати параметри, такі як площа, кількість робочих місць, ціна тощо. На сайті також є фільтри для відсіювання непідходящих варіантів.

Також застосунок надає орендодавцям можливість безкоштовно розмістити свої оголошення на сайті та привернути нових клієнтів. Крім того, сервіс допомагає орендодавцям з підготовки договорів та іншої документації. Сервіс спрощує процес пошуку та оренди, забезпечуючи користувачам швидкий та надійний спосіб знайти оптимальний варіант.

Таблиця 1.2 – Опис системи MatchOffice

Назва	MatchOffice
Розробник	Match Office
Архітектура	3tier application
Мова реалізації	Ruby On Rails, JavaScript
Функції	1. Пошук коворкінгів за типом і локацією. 2. Управління власними коворкінгами через web-ui. 3. Перегляд місць на мапі. 4. Можливість детального перегляду інформації про місце.
Переваги	1. Наявність можливості порівняння локацій. 2. Голосовий пошук.
Недоліки	1. Фільтри доступні лише у режимі порівняння. 2. Незручний UI/UX.
Вебсайт	https://www.matchoffice.com.ua

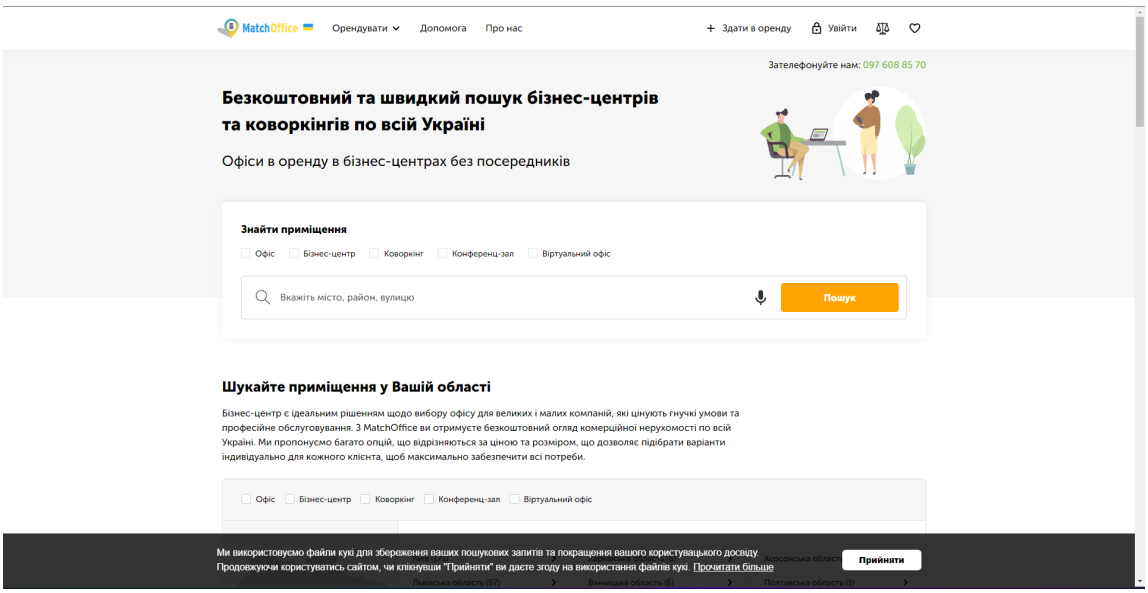

Рисунок 1.2 – Вигляд застосунку MatchOffice

Асоціація коворкінгів України

Онлайн-ресурс, створений з метою підтримки та розвитку коворкінг-спільноти в Україні. Цей вебсайт є інформаційним порталом, де можна знайти різноманітні корисні матеріали для власників та учасників коворкінгів.

Таблиця 1.3 – Опис системи Асоціація Коворкінгів України

Назва	Асоціація Коворкінгів України
Розробник	АКУ
Архітектура	3tier application
Мова реалізації	PHP, JS, WordPress
Функції	1. Мапа з коворкінгами. 2. Можливість обрати локацію. 3. Можливість переглянути інформацію про місце.
Переваги	1. Зручний інтерфейс. 2. Наявність карти з локаціями.
Недоліки	1. Відсутність фільтрів за критеріями. 2. Відсутність детальної інформації про місце.
Вебсайт	https://coworkingassociation.org.ua/map

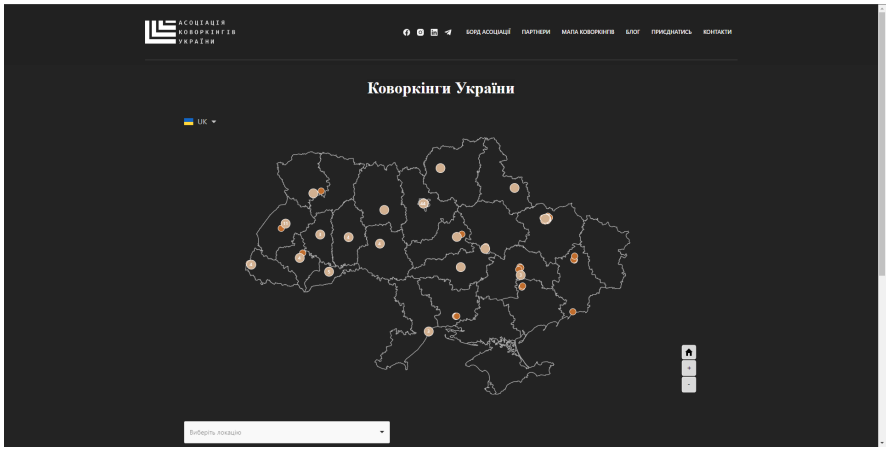

Рисунок 1.3 – Вигляд застосунку Асоціація коворкінгів України

1.2 Аналіз системи, що розробляється

Вебзастосунку для пошуку коворкінгу призначений для пошуку підходящого місця для роботи. Користувач має потребу у знаходженні тимчасової або постійної локації для роботи, фільтрує потрібні заклади у застосунку, та обирає необхідний.

Система повинна забезпечувати багатокористувацький режим роботи та високу доступність. Інтерфейс має бути зрозумілим для пересічного користувача. Система повинна бути у працездатному стані як мінімум 12 годин на добу, 7 днів на тиждень.

Таблиця 1.4 – Опис системи, що розробляється

Функції	1. Реєстрація користувачів. 2. Пошук та перегляд закладів коворкінгу. 3. Фільтрація закладів за критеріями. 4. Додавання нових закладівх. 5. Перегляд інформації про заклад. 6. Підтвердження власності закладом. 7. Адміністрування власників закладів.
Користувачі	1. Анонімний користувач. 2. Власник закладу. 3. Адміністратор.

Кінець таблиці 1.4

Сценарії роботи	1. Користувач фільтрує та переглядає заклади. 2. Користувач шукає необхідну локацію на мапі за адресою. 3. Користувач переглядає детальну інформацію про заклад та відкриває його у Google Maps для прокладання маршруту. 4. Власник додає новий заклад. 5. Власник закладу підтверджує власність закладом. 6. Адміністратор підтверджує власність закладом власника. 7. Адміністратор блокує власника.

1.3 Специфікація вимог до програмного забезпечення

ПРИЗНАЧЕННЯ ТА МЕЖІ ПРОЄКТУ

Призначення системи (застосунку), для якої розробляється програмне забезпечення

Призначенням застосунку є автоматизація процесу пошуку коворкінгу за рахунок розробки ПЗ вебзастосунку пошуку коворкінгу.

Погодження, що ухвалені в програмній документації

Було погоджено, що для створення загального ПЗ та його злагодженої роботи будуть використовуватися фреймворки ASP.NET та Angular.

Межі проєкту ПЗ

Крайня дата завершення роботи над ПЗ – 28.04.2023р.

ЗАГАЛЬНИЙ ОПИС

Сфера застосування

Дане ПЗ не має обмежень у сферах його застосування, за виключенням регіонального обмеження, оскільки використання поза Україною не передбачене.

Характеристики користувачів

Основні характеристики користувачів: наявність будь-якого девайсу, що підтримує сучасні браузери та доступу до мережі Інтернет.

Загальна структура і склад системи

Основні частини для створення програмного забезпечення: проксі-сервер, БД, вебзастосунок, API.

Загальні обмеження

Обмеження для роботи з ПЗ – наявність доступу до мережі Інтернет та наявність відповідної версії браузера.

ФУНКЦІЇ ВЕБЗАСТОСУНКУ ПОШУКУ КОВОРКІНГУ

Функція пошуку та перегляду коворкінгів

Опис функції

Функція пошуку та перегляду коворкінгів дозволяє користувачу знайти відповідний заклад для роботи.

Вхідна і вихідна інформація

Вхідна інформація – назва міста, області, населеного пункту, у якому проводиться пошук, а також критерії фільтрації. Уся інформація є опціональною.

Вихідна інформація – набір локацій, що відповідають критеріям користувача.

Функціональні вимоги

База даних з підтримкою геопросторових функцій маніпуляції.

Функція додання нового закладу

Опис функції

Функція додання нового закладу допомагає додати власнику закладу свій заклад у систему з метою поширення інформації про нього та підвищення відвідуємості.

Вхідна і вихідна інформація

Вхідна інформація – фізична адреса закладу, додаткова інформація про обладнання закладу і цінову політику.

Вихідна інформація – заклад, доданий у базу даних.

Функціональні вимоги

База даних, доступ до Google Maps і Google Places API та доступ до мережі інтернет.

ВИМОГИ ДО ІНФОРМАЦІЙНОГО ЗАБЕЗПЕЧЕННЯ

Джерела і зміст вхідної інформації (даних)

В даному ПЗ є два основних джерела інформації – користувач і Google Places API. Користувач надає додаткову і уточнюючу інформацію до тої, що надається Google Places API, а саме: інформацію про цінову політику закладу і наявність додаткових характеристик закладу.

Нормативно-довідкова інформація (класифікатори, довідники тощо)

Вимоги до даного пункту відсутні.

Вимоги до способів організації, збереження та ведення інформації

Обмін даними між клієнтською та серверними частинами відбувається за допомогою RESTful API, а сервер зберігає дані у БД – PostgreSQL.

ВИМОГИ ДО ТЕХНІЧНОГО ЗАБЕЗПЕЧЕННЯ

Жорстких технічних вимог немає. Комп’ютер чи ноутбук з процесором не старше ніж кілька поколінь, а також з оперативною пам’яттю не менше 8 ГБ.

ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Архітектура програмної системи

Архітектура складається з серверної та клієнтської частин та БД.

Системне програмне забезпечення

Для написання front-end частини повинні бути використані Angular та Angular Material, для back-end частини – ASP.NET та PostgreSQL у якості БД. Процес обміну даними має відбуватися за допомогою REST.

Мережне програмне забезпечення

Для створення ПЗ використано ОС Windows 10, у якості інтегрованого середовища розробки – Webstorm та Rider та Google Chrome у якості браузеру.

Програмне забезпечення ведення інформаційної бази

Усе введення інформаційної бази відбувається лише через REST-інтерфейс, який надає back-end частина, яка у свою чергу взаємодіє за допомогою Entity Framework з PostgreSQL.

Мова і технологія розробки ПЗ

Програмне забезпечення має розроблюватися з використанням мов C# і Typescript та фреймворків ASP.NET і Angular.

ВИМОГИ ДО ЗОВНІШНІХ ІНТЕРФЕЙСІВ

Інтерфейс користувача

Інтерфейс має бути зручним та задовольняти усі потреби користувача і бути легким для розуміння. Шаблон вебзастосунка складається з трьох частин – тулбару, бокового меню та нижньої панелі фільтрів.

Апаратний інтерфейс

Апаратним інтерфейсом користувача є інтерфейс девайсу – ПК, смартфона, ноутбука тощо, який буде використовуватися для взаємодії з застосунком.

Програмний інтерфейс

Angular – сучасний фреймворк для побудови реактивних вебзастосунків. ASP.NET – у свою чергу використовується для серверної частини застосунків.

Комунікаційний протокол

Застосунок передбачає використання протоколів HTTP, HTTPS для REST та TCP/IP для взаємодії з БД.

ВЛАСТИВОСТІ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Доступність

ПЗ має бути доступним для звичайного користувача за умови наявності у нього доступу в інтернет та відповідного девайсу з підтримкою сучасних браузерів.

Супроводжуваність

Застосунок має бути легко супроводжуємим і відкритим для розширення у майбутньому.

Переносимість

Програмне забезпечення може працювати на будь-яких операційних системах, що підтримують сучасні браузери.

Продуктивність

Продуктивність роботи ПЗ залежить від мережевого з’єднання та навантаження на БД. Середній час на виконання запиту не має перевищувати 3 секунди.

Надійність

ПЗ має бути надійним і виключати можливості зловживанням рівнем доступу. Користувачі, за виключенням адміністратора, мають бачити та редагувати лише свої ресурси.

Безпека

Процедура авторизації та автентифікації має відбуватися згідно з найкращих практик та останніх стандартів. Згідно з вимогами, реєструватися та взаємодіяти з застосунком можна лише на території України.

Висновки до розділу 1

В першому розділі кваліфікаційної роботи бакалавра розглянуто та проведено аналіз застосунків-аналогів вебзастосунку пошуку коворкінгу, завдяки чому було виділено основні недоліки існуючих систем та їх переваги. Крім того було проведено аналіз системи, що розробляється, де було визначено основний функціонал системи та користувачів застосунку.

Важливою частиною розділу є специфікація вимог до програмного забезпечення. Специфікація містить у собі перелік вимог, які повинні бути задоволені в розробці програмного забезпечення та описує функціональні та нефункціональні вимоги до системи, а також вимоги до документації, тестування та підтримки. Специфікація вимог є дуже важливою частиною на етапі проектування системи, оскільки вона дозволяє забезпечити зрозумілість та однозначність вимог до програмного забезпечення.

2 МОДЕЛЮВАННЯ ВЕБЗАСТОСУНКУ ПОШУКУ КОВОРКІНГУ

2.1 Алгоритм роботи вебзастосунку пошуку коворкінгу

Вебзастосунок пошуку коворкінгу має бути зручним та простим у використанні, оскільки розрахований на широку користувацьку аудиторію. Користувач повинен мати змогу користуватися застосунком використовуючи ПК або смартфон. Застосунок призначений для пошуку коворкінгів у обраній користувачем області.

Для графічного відображення будь-яких алгоритмів та процесів часто використовують блок-схеми та діаграми діяльності.

Блок-схема – це графічне зображення алгоритму, що складається з різних блоків, які вказують на кроки або дії, що потрібно виконати для досягнення певної мети. У розробці програмного забезпечення, блок-схема може бути використана для відображення алгоритмів роботи застосунку.

Діаграма діяльності – це графічне зображення процесу або дії, яка складається з різних етапів та дій. Вона описує, як об'єкти взаємодіють між собою, які дії потрібно виконати та в якому порядку, щоб досягнути певної мети. Діаграма діяльності зазвичай використовується для моделювання бізнес-процесів, але може також бути використана для моделювання операцій та алгоритмів в програмному забезпеченні.

Розробка діаграм діяльності та блок-схем дозволить детальніше продемонструвати процес роботи вебзастосунку для пошуку коворкінгу та краще зрозуміти алгоритми взаємодії користувача із застосунком. Усього створено три діаграми діяльності, що демонструють основні функції різних користувачів у системі:

1) *Перегляд місць для роботи* (рис. 2.1). Діаграма містить містить такі кроки як встановлення фільтрів, переміщення мапи, а також клік на маркер локації.

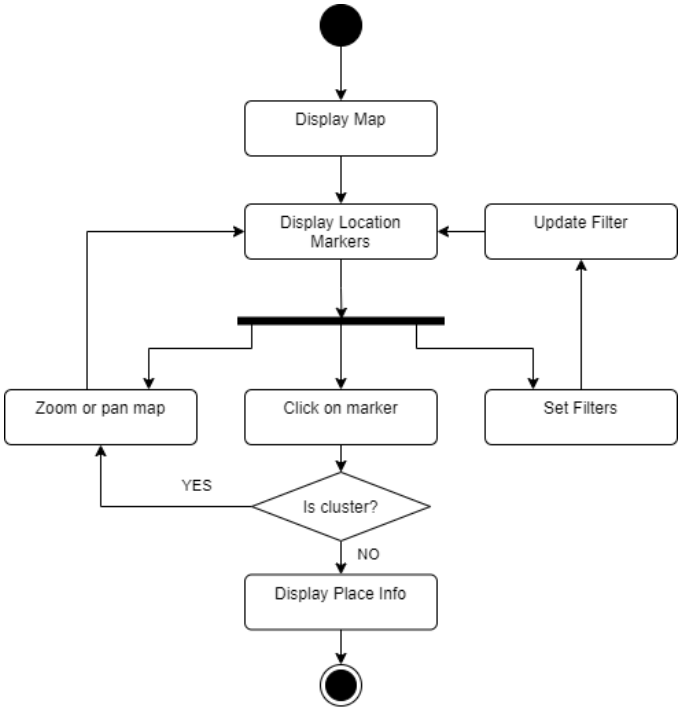

Рисунок 2.1 – Діаграма діяльності перегляду місць для роботи

2) *Підтвердження володіння місцем* (рис. 2.2). Діаграма містить алгоритм підтвердження володіння місцем з двома варіантами підтвердження власності.

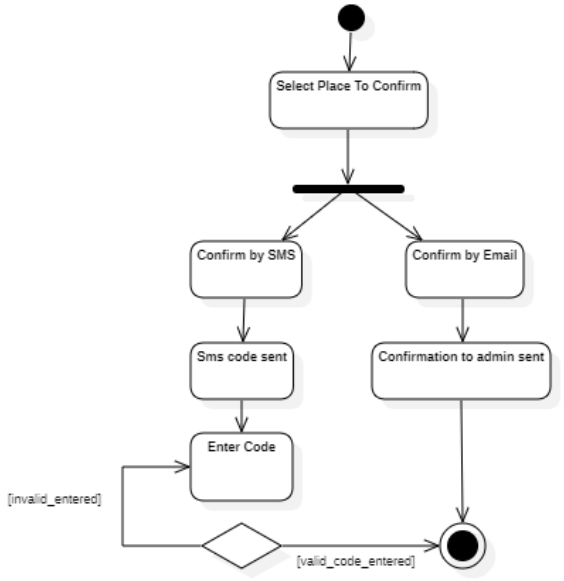

Рисунок 2.2 – Діаграма діяльності підтвердження володіння місцем

3) *Додавання місця для роботи* (рис. 2.3). Діаграма містить такі введеня адреси, вибір локації, заповнення деталей та перевірка на унікальність доданої локації.

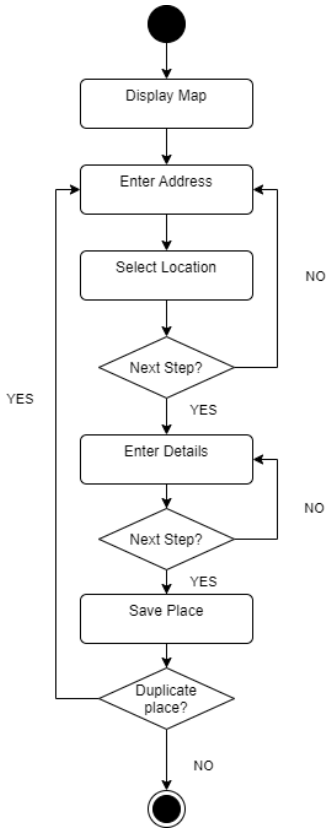

Рисунок 2.3 – Діаграма діяльності додавання місця роботи

2.2 Створення Use Case

Текстові сценарії use case-ів – це описи послідовностей дій, які здійснюють користувачі системи, щоб виконати певні задачі з використанням системи. Такі сценарії можуть бути використані для детального опису вимог до системи, для забезпечення спільного розуміння між розробниками та зацікавленими сторонами, а також для тестування системи.

Текстові сценарії use case-ів можуть бути представлені у трьох формах:

1) Коротка форма: це найбільш стислий опис сценарію, який містить тільки найбільш важливі елементи. Цей опис має формат «Користувач робить щось, система робить щось відповідне»;

2) Поверхнева форма: це дещо більш детальний опис, який включає в себе більше інформації про дії користувача та системи. Цей опис може містити кроки сценарію, а також умови, за яких вони відбуваються;

3) Повна форма: це найбільш детальний опис сценарію, який включає в себе всі можливі кроки сценарію, а також всі умови, які можуть вплинути на виконання цих кроків. Цей опис зазвичай містить більше інформації, ніж необхідно для розуміння сценарію, і може включати подробиці щодо виконання дій та взаємодії користувача та системи.

Кожна з цих форм може бути використана в залежності від потреб проєкту. Наприклад, коротка форма може бути використана для швидкого опису вимог до системи, а повна форма може бути використана для детального опису поведінки системи відповідно до вимог.

Короткий usecase

Користувач має потребу у пошуку місця для роботи, заходить на вебсайт, шукає свою поточну локацію та переглядає локації, що знаходяться поруч. Після чого обирає необхідне місце для роботи.

Поверхневий usecase

Головний сценарій (успішний)

Користувач має потребу у пошуку місця роботи, заходить на вебсайт, налаштовує перелік фільтрів з умовами та переглядає місця, що знаходяться поруч. Знайшовши необхідне місце користувач обирає перегляд інформації про локацію і переглядає детальні свідчення про місце, після чого прокладає маршрут до місця, скориставшись кнопкою «Відкрити у Google Maps»

Альтернативний сценарій:

1) Користувач не знаходить поблизу місць для роботи і змінює масштаб мапи.
2) Користувач не знаходить місць для роботи з потрібними умовами і змінює фільтри.

Таблиця 2.1 – Повний usecase

Primary Actor	Користувач
Scope	System
Level	User-goal
Preconditions	Відсутні
Stakeholders and interests	1. Адміністратор: зацікавлений у підтримці застосунку. 2. Власник закладу: зацікавлений у просуванні свого закладу. 3. Користувач: зацікавлений у пошуку закладу для роботи.
Main Success Scenario	1. Користувач відкриває застосунок. 2. Користувач обирає область на мапі. 3. Користувач встановлює необхідні фільтри. 4. Користувач бачить оновлені маркери локацій на мапі. 5. Користувач обирає підходящу йому локацію за місцем розташування. 6. Користувач бачить детальну інформацію на боковій панелі і за необхідності продовжує пошуки. 7. Користувач натискає на кнопку «Прокласти маршрут» або «Відкрити в Google Maps». 8. Користувач переходить у застосунок Google Maps.
Result	Користувач знайшов коворкінг

Кінець таблиці 2.1

Extensions	1. Користувач не бачить маркерів на мапі. 1.1. Користувач змінює фільтри. 1.2. Користувач віддаляє мапу. 1.3. Користувач бачить оновлені маркери.
Special Requirements	Система має швидкий відгук та зручний інтерфейс
Frequency of Occurrence	Система може працювати майже безперервно.

Use case діаграми (також відомі як діаграми використання) – це інструмент моделювання вимог для розробки програмного забезпечення та систем, який дозволяє зобразити функціональність системи з точки зору користувача. Вони дозволяють визначити, які дії повинні виконувати користувачі, які взаємодіють з системою, та як система повинна відповідати на ці дії.

Use case діаграми включають в себе акторів (користувачів) та їхні взаємодії з системою. Вони демонструють, як користувачі використовують систему та як система реагує на їх дії. Use case діаграми можуть бути використані для спілкування зі стейкхолдерами та забезпечення того, щоб всі зацікавлені сторони мали чітке уявлення про те, що повинна робити система.

Основні переваги використання use case діаграм полягають у тому, що вони:

- дозволяють розуміти вимоги до системи з точки зору користувача;
- допомагають ідентифікувати можливість взаємодії з системою;
- забезпечують чітке уявлення про те, як система повинна працювати з точки зору користувача;
- допомагають виявляти проблеми та потенційні ризики в системі на ранніх стадіях розробки.

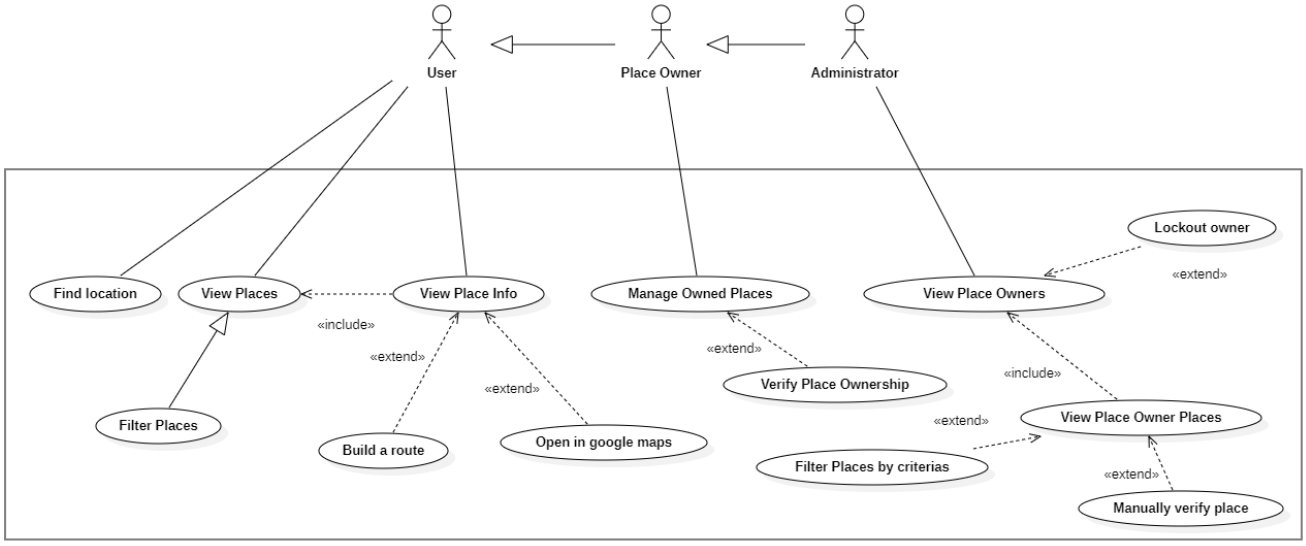

Рисунок 2.4 – Діаграма використання вебзастосунку пошуку коворкінгу

На цьому розгляд use case діаграм заверешений.

2.3 Створення діаграми розгортання

Діаграми розгортання (deployment diagrams) – це один з видів діаграм UML, які використовуються для моделювання архітектури системи та показу розміщення її компонентів на різних фізичних пристроях (наприклад, на серверах або комп'ютерах).

Діаграми розгортання дозволяють описати технічну структуру системи, що включає в себе апаратне забезпечення, програмне забезпечення та мережеві пристрої. Вони використовуються для візуалізації інфраструктури системи, де можна показати фізичне розташування компонентів, що забезпечують роботу програмного забезпечення, та зв'язки між ними.

Діаграми розгортання складаються з наступних елементів:

1) Вузол (node) – фізичний пристрій або сервіс, на якому розгортовується програмне забезпечення;
2) Компонент (component) – програмний модуль, що виконує певну функцію системи;

3) Взаємодія між вузлами (communication) – з'єднання між вузлами, які виконують різні функції системи;
4) Взаємодія між компонентами та вузлами (deployment) – зв'язки між компонентами та вузлами, що дозволяють запустити та виконувати програмний код на певному пристрої;
5) Артефакт (artifact) – фізичний об'єкт, що використовується для зберігання програмних компонентів та їх виконання.

Розглянемо діаграму розгортання для вебзастосунку пошуку коворкінгу (рис. 2.5). Оскільки вебзастосунок має звичайну трьохланкову архітектуру (3tier application), то основні компоненти системи це:

1) Клієнтська сторона (client side): веббраузер, який взаємодіє з додатком;
2) Серверна сторона (server side): вебсервер, який обслуговує запити від клієнта та надсилає відповіді;
3) Система керування базами даних (DBMS): PostgreSQL яка забезпечує зберігання та доступ до даних.

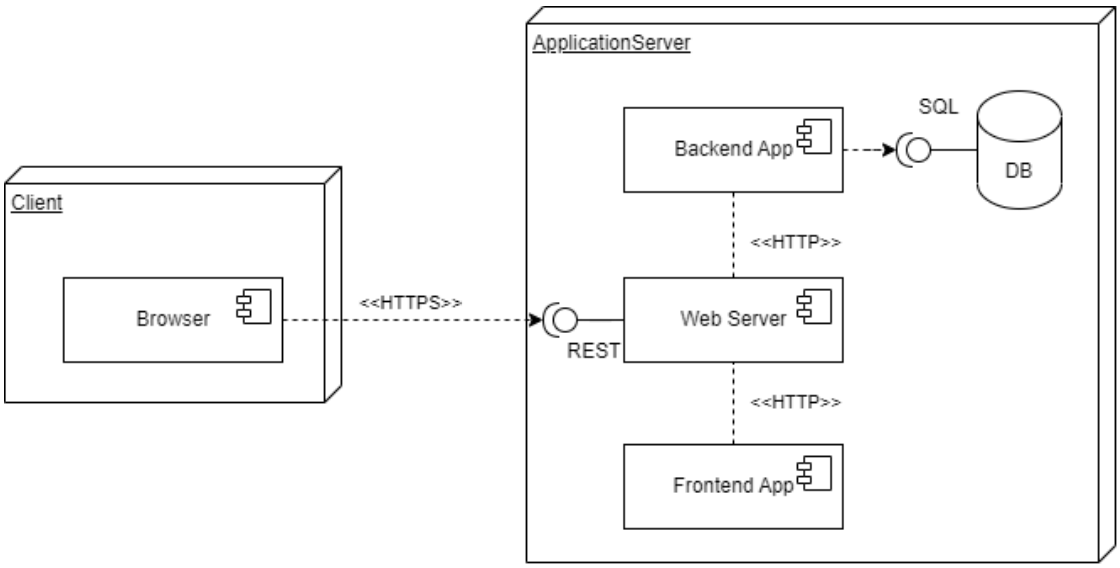

Рисунок 2.5 – Діаграма розгортання вебзастосунку пошуку коворкінгу

На цьому розгляд діаграм розгортання завершений.

2.4 Створення діаграми взаємодій

Діаграма взаємодій – це графічне зображення взаємодії між об'єктами в системі, що включає послідовність повідомлень, що передаються між об'єктами. Вона показує порядок виконання повідомлень та зв'язки між об'єктами. Діаграма взаємодій є одним з видів діаграм UML і використовується для моделювання складних систем, де об'єкти взаємодіють один з одним. Ця діаграма допомагає візуалізувати послідовність повідомлень між об'єктами, які взаємодіють в системі, і дозволяє зрозуміти, як система працює у різних сценаріях взаємодії.

Діаграма №1 (рис. 2.6) демонструє взаємодію користувача (власника закладу) із застосунком при підтвердженні володінням місцем. Дана діаграма демонструє послідовність викликів методів та об’єкти в системі, що їх обробляють.

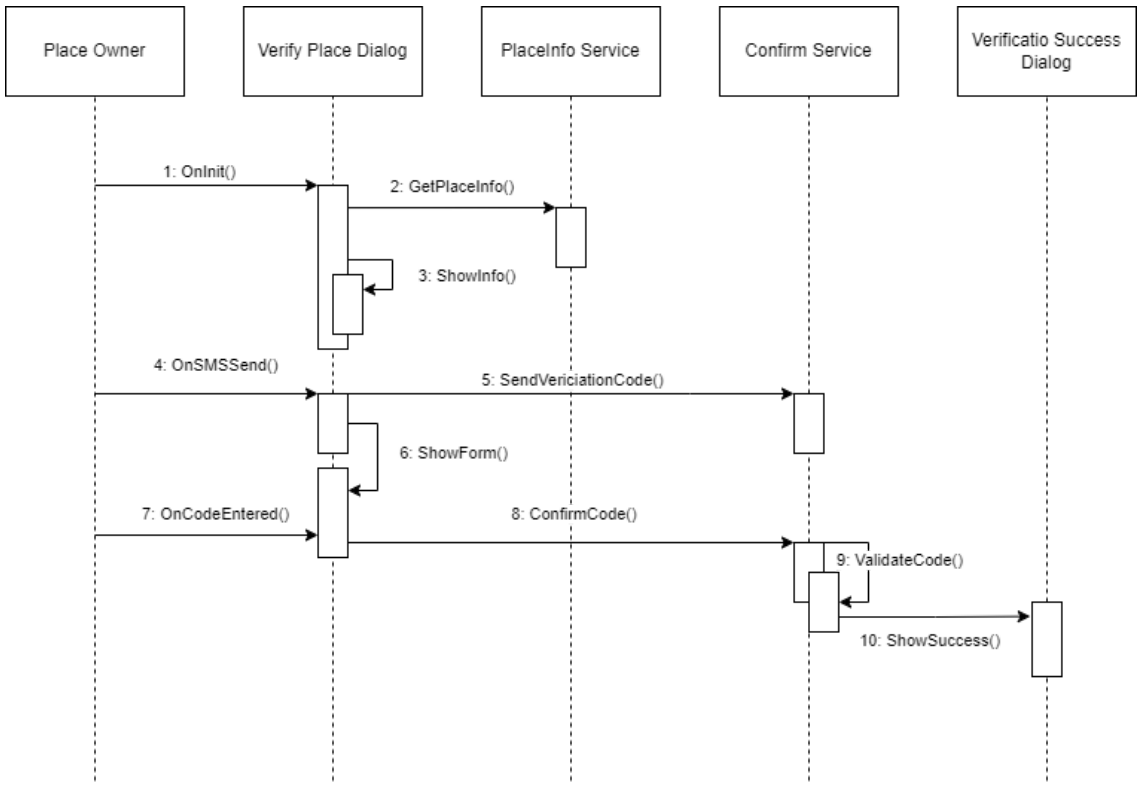

Рисунок 2.6 – Діаграма взаємодії підтвердження володіння місцем

Діаграма №2 (рис. 2.7) демонструє взаємодію користувача з застосунком під час перегляду детальної інформації про місце. На діаграмі зображено взаємодію з різними компонентами, як наприклад, сервісом фотографій, який спілкується з Google Maps API.

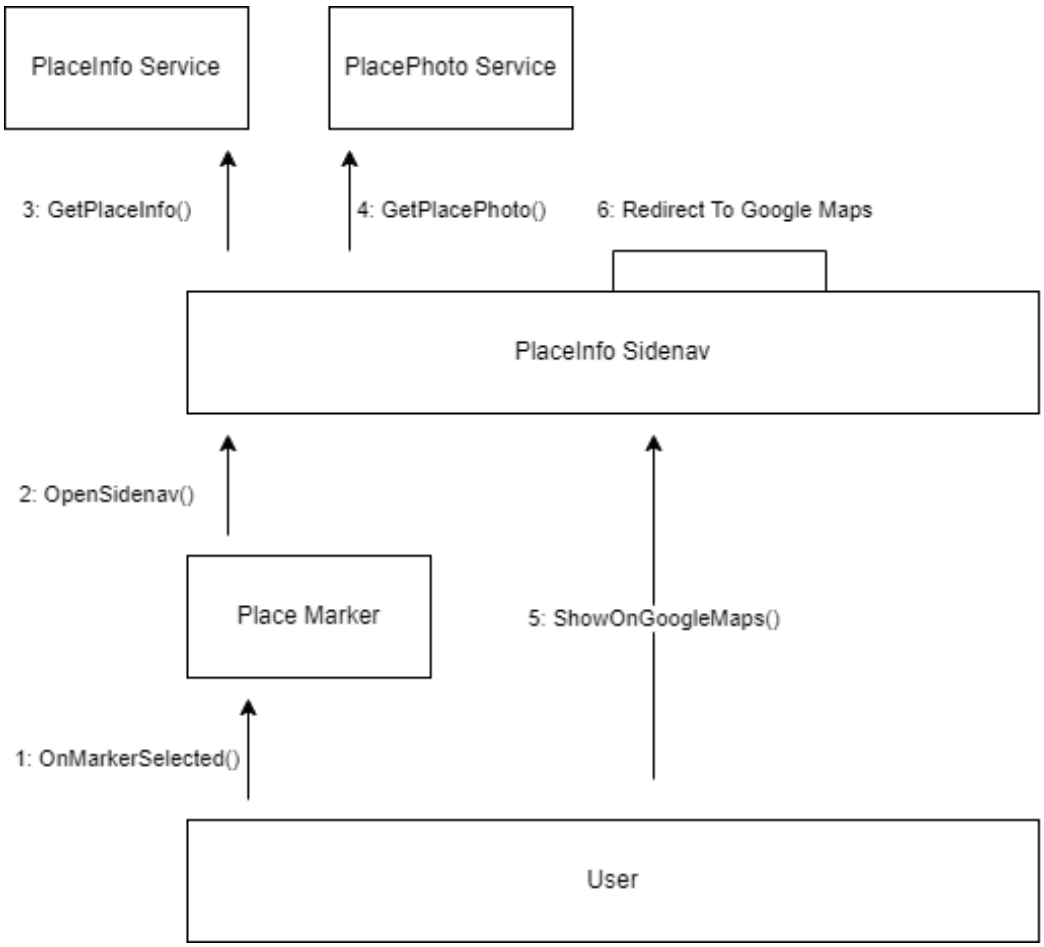

Рисунок 2.7 – Діаграма взаємодії перегляду місця у Google Maps

Діаграма №3 (рис. 2.8) демонструє взаємодію користувача під час процесу додавання нової локації у застосунок. На ній зображено послідовність дій та виклики методів, що відбуваються під час проходження даного процесу, зокрема і етапів, які користувач не бачить напряму, як наприклад, звернення до функціоналу API.

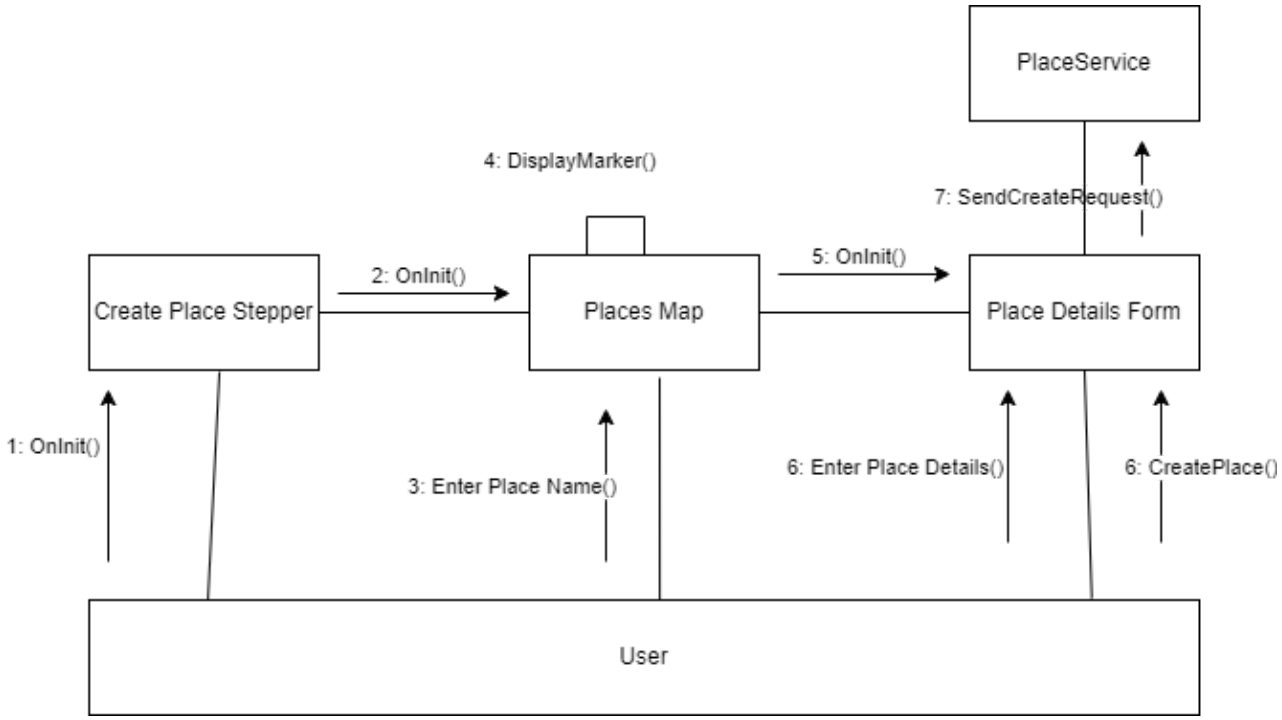

Рисунок 2.8 – Діаграма взаємодії додавання нової локації

На цьому розгляд діаграм взаємодій заверешно.

Висновки до розділу 2

У другому розділі кваліфікаційної роботи виконано моделювання програмного забезпечення, а саме:

1) розроблені діаграми діяльності;
2) створено три текстових сценарії використання: короткий, поверхневний і повний;
3) створено діаграму використання;
4) створено діаграму взаємодії;
5) створено діаграму розгортання.

Створені діаграми та описи допомогли чіткіше сформувати уявлення про майбутню систему, а також виступають чудовим засобом документації функціоналу застосунку.

3 ПРОЄКТУВАННЯ ВЕБЗАСТОСУНКУ ПОШУКУ КОВОРКІНГУ ТА ОГЛЯД ТЕХНОЛОГІЙ

3.1 Створення UML-діаграм

UML-діаграми є стандартом для моделювання програмного забезпечення [6] та дозволяють розглядати систему на різних рівнях деталей.

Діаграма класів описує структуру системи, включаючи класи, їх атрибути та методи, спадкування та асоціації між класами.

Діаграма компонентів дозволяє відображати фізичну структуру системи та відносини між компонентами, такі як залежності та інтерфейси.

Діаграма пакетів допомагає відображати організацію компонентів системи на рівні пакетів, що дозволяє структурувати та управляти складністю системи.

Усі ці діаграми (табл. 3.1) є важливими для розуміння та проєктування системи та дозволяють комунікувати з розробниками, бізнес-аналітиками та іншими учасниками процесу розробки програмного забезпечення.

Таблиця 3.1 – Опис використання UML-діаграм

Назва	Використання
Діаграма класів	Використовується для відображення класів, інтерфейсів, спадкування, асоціацій, методів та властивостей системи.
Діаграма компонентів	Використовується для відображення компонентів системи та зв'язків між ними.
Діаграма пакетів	Використовується для відображення пакетів та залежностей між ними.

Перед початком розробки вебзастосунку для пошуку коворкінгу на етапі проєктування створено діаграми класів, діаграму компонентів та пакетів, для

кращого уявлення про майбутню структуру коду створюваного ПЗ. Для створеня діаграм використано застосунок Diagrams.net.

3.1.1 Діаграма класів

Діаграма класів (рис. 3.1) є однією з ключових моделей, що використовуються під час проектування програмного забезпечення. Це схематичне зображення, що показує класи програмного забезпечення, їх взаємозв'язки та методи, які вони мають.

Перед початком розробки програмного забезпечення важливо створити діаграму класів, оскільки це допомагає:

1) Визначити структуру програмного забезпечення: діаграма класів допомагає визначити, які класи потрібні для програмного забезпечення та як вони пов'язані між собою;
2) Розробити кращі рішення: створення діаграми класів дозволяє розробникам відповідати на питання, які класи потрібні для реалізації певної функції, які методи вони повинні містити та як вони пов'язані між собою;
3) Зменшити ризик помилок: діаграма класів дозволяє зрозуміти, як класи взаємодіють між собою та що очікується від кожного класу;
4) Покращити комунікацію: діаграма класів може бути використана для комунікації між розробниками та іншими учасниками проекту;
5) Спростити тестування та супровід: діаграма класів може допомогти визначити, які класи повинні бути тестовані та які методи повинні бути покриті тестами;
6) Зберегти час та кошти: створення діаграми класів може зберегти час та кошти, оскільки воно дозволяє розробникам зрозуміти вимоги до програмного забезпечення та визначити оптимальну структуру.

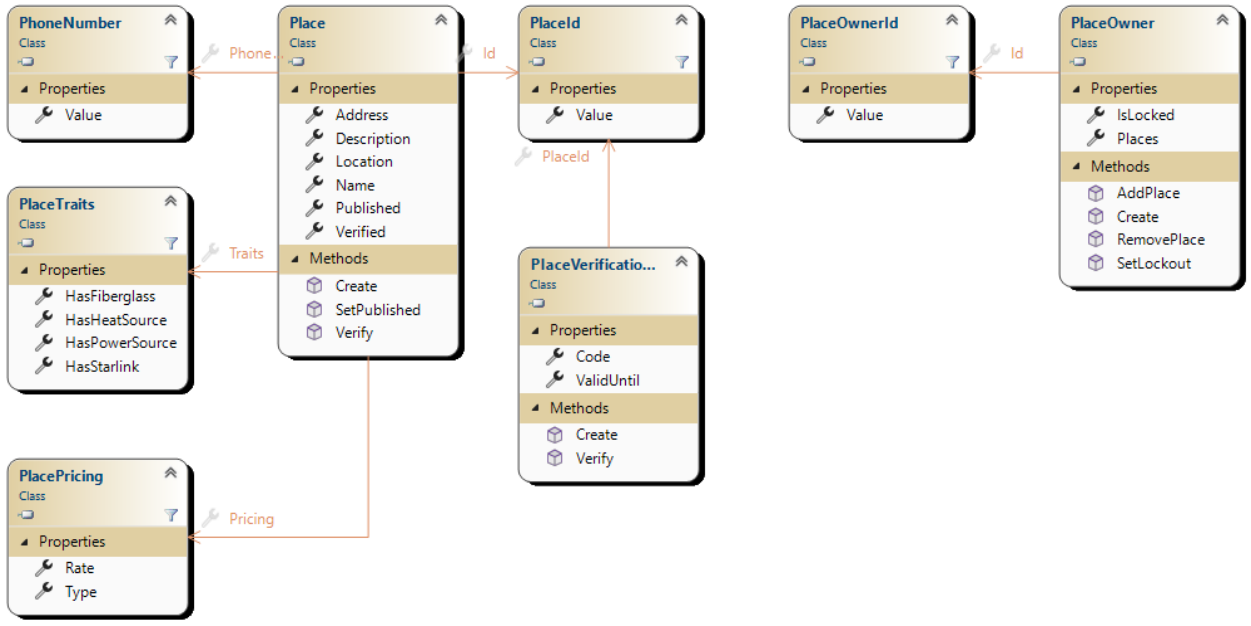

Рисунок 3.1 – Діаграма домених класів

Додатково для діаграми класів наведена таблиця (табл. 3.2), у якій представлений опис компонентів методів та призначення кожного класу.

Таблиця 3.2 – Класи та їх характеристика

Клас	**Компоненти**	**Методи**	**Призначення класу**
PlaceVerificationCode	Code	Create()	Клас коду підтвердження
	ValidUntil	Verify()	
PlacePricing	Rate		Інкапсуляція цінової політики
	Type		
PlaceOwner	IsLocked	AddPlace()	Клас власника закладу
		Create()	
	Places	RemovePlace()	
		SetLockout()	
PhoneNumber	Value		Інкапсуляція логіки номеру телефону

Кінець таблиці 3.2

Клас	Компоненти	Методи	Призначення класу
Place	Address	Create()	Клас закладу
	Description		
	Location		
	Name	SetPublished()	
	Published		
	Verified		
	PhoneNumber	Verify()	
	PlacePricing		
	PlaceTraits		
PlaceTraits	HasFiberglass		Інкапсуляція характеристик закладу
	HasHeatSource		
	HasPowerSource		
	HasStarlink		

Діаграма класів відображає структуру класів у системі та має наступні особливості:

1) Певні типи зв’язків не представлені на діаграмі через особливості ORM-бібліотеки;
2) Діаграма відображає взаємодію між доменними класами. Класи інших пакетів, фреймворку та стандартної бібліотеки не представлені;
3) Діаграма єдина для усіх класів і класи не можуть повноцінно існувати в розриві один від одного через композицію [7] .

3.1.2 Діаграма компонентів

Оскільки вебзастосунок для пошуку коворкінгу має трьохланкову архітектуру на діаграмі компонентів (рис. 3.2) складатиметься з таких шарів:

1) Шар презентації (Presentation layer) – цей шар відповідає за відображення інформації користувачу. Він містить компоненти, які взаємодіють з користувачем, такі як форми, кнопки, меню і т.д;

2) Шар логіки додатку (Business layer) – цей шар відповідає за обробку даних і здійснення логіки додатку. Він містить компоненти, які забезпечують взаємодію з базою даних, валідацію даних, обчислення і т.д;

3) Шар даних (Persistence layer) – цей шар відповідає за зберігання та доступ до даних. Він містить компоненти, які забезпечують доступ до бази даних, файлову систему, мережу і т.д;

4) Шар доменої логіки додатку (Domain layer) – цей шар відповідає за логіку додатку, що відображає бізнес-логіку конкретної предметної галузі або домену, в якому працює додаток.

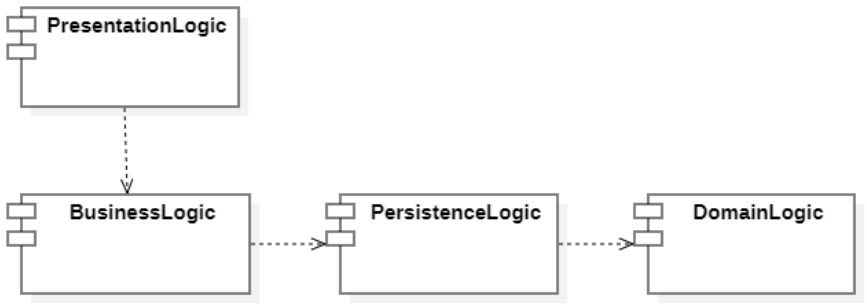

Рисунок 3.2 – Діаграма компонентів

На цьому розгляд діаграми компонентів завершено.

3.1.3 Діаграма пакетів

Усі класи в системі розподілені на пакети (рис 3.3), що у свою чергу формують систему. Пакети можуть посилатися на інші пакети у системі таким чином формуючи дерево залежностей [19].

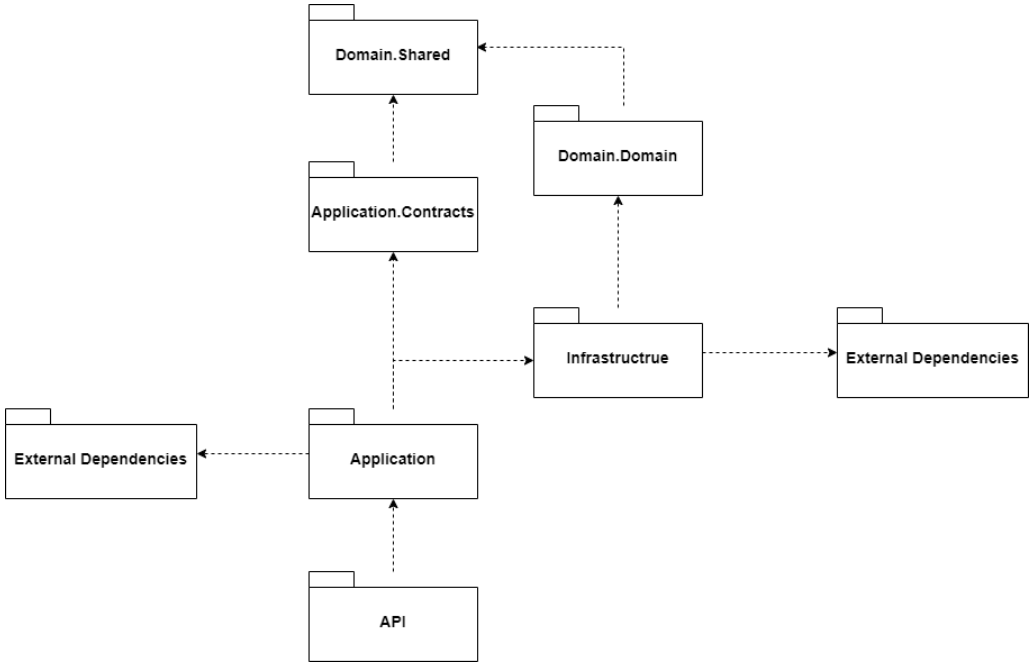

Рисунок 3.3 – Діаграма пакетів

Коренем дерева залежностей пакетів системи є пакет API, який акумулює у собі всі можливі пакети системи разом з їх залежностями. До діаграми пакетів були також включені сторонні залежності, які представлені у вигляді єдиного пакету під назвою External Dependencies.

3.2 Створення ERD-діаграми

ERD (Entity Relationship Diagram) – це графічне зображення, яке показує взаємозв'язки між об'єктами (сутностями) в базі даних. Це популярний інструмент проєктування баз даних, який дозволяє описати структуру даних і зв'язки між ними.

ERD складається з наступних елементів:

1) Сутності (Entity) – це об'єкти або поняття, що містять дані, які потрібно зберігати в базі даних. Наприклад, «Клієнт», «Замовлення», «Товар»;
2) Атрибути (Attribute) – це характеристики сутності. Наприклад, для сутності «Клієнт» атрибутами можуть бути «Ім'я», «Прізвище», «Адреса»;

3) Відношення (Relationship) – це зв'язки між сутностями, які показують, як одна сутність пов'язана з іншою. Відношення може бути один-до-одного, один-до-багатьох або багато-до-багатьох.

ERD-діаграма (рис. 3.4) дуже важлива для проєктування баз даних. Вона допомагає розуміти структуру даних і забезпечує однозначне і чітке визначення сутностей, атрибутів та їх відношень. Створення ERD діаграми на початку проєктування дозволяє уникнути помилок і проблем з базою даних в майбутньому.

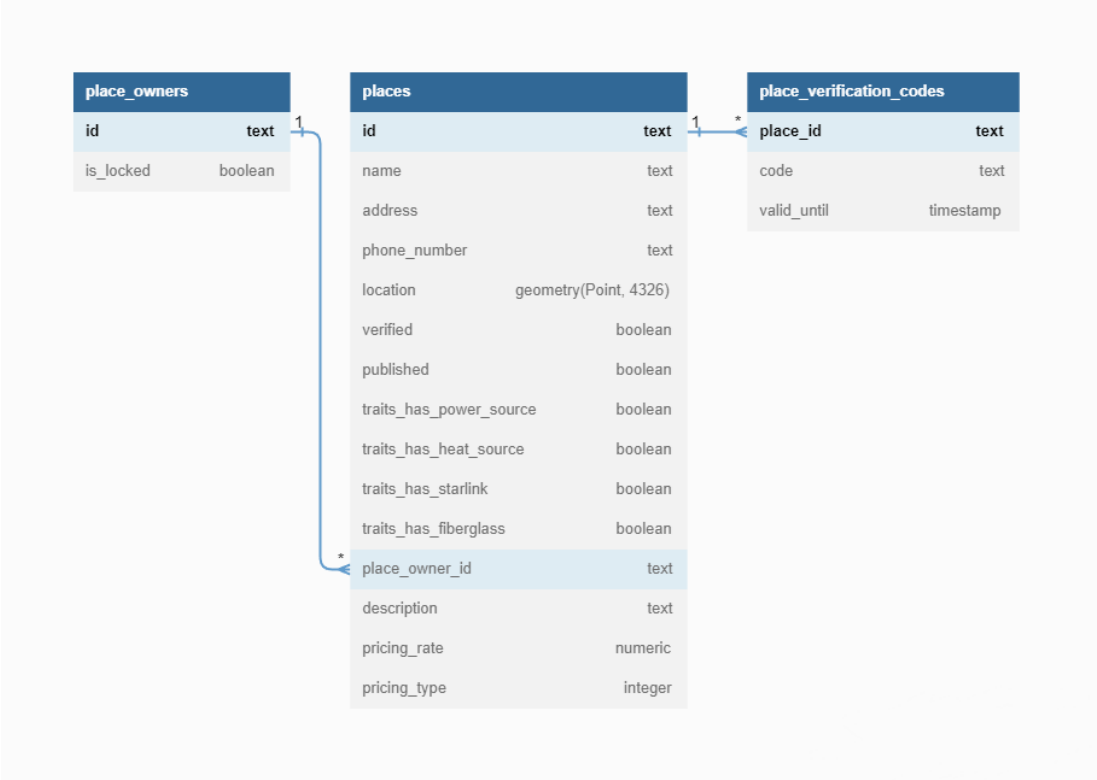

Рисунок 3.4 – ERD-діаграма

Центральною сутністю у системі є власник закладу, який одночасно є і користувачем системи. Другорядною, але не менш важливою сутнітю є заклад, що містить у собі всю необхідну інформацію, яка відображається на вебінтерфейсі застосунка.

3.3 Огляд стеку технологій

При розробці вебзастосунку пошуку коворкінгу використано наступний стек технологій (рис. 3.5):

- мови програмування: C# та TypeScript;
- front-end фреймворк Angular;
- back-end фреймворк ASP.NET Core;
- сервіси Google Maps та Auth0;
- БД postgres.

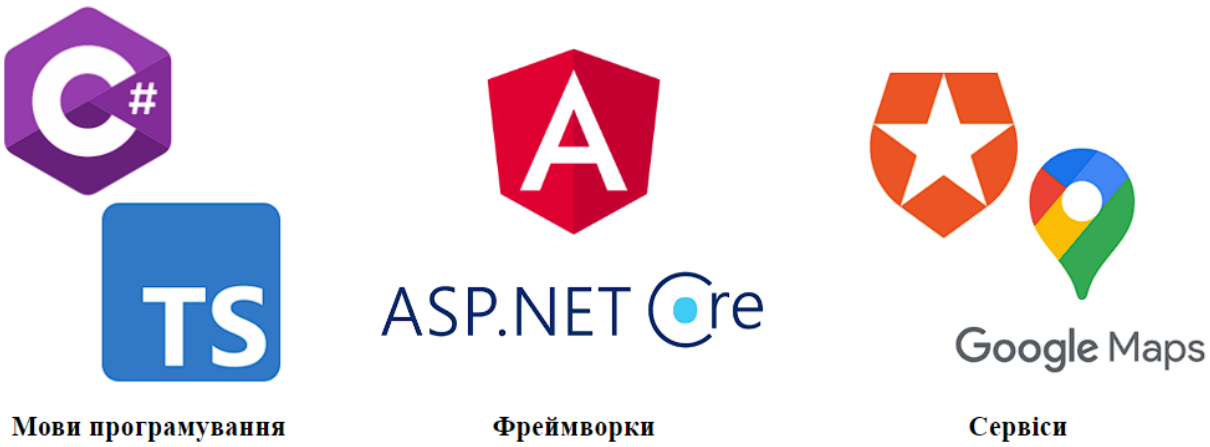

Рисунок 3.5 – Логотипи мов програмуваня та фреймворків

Вибір стеку технологій є важливим на кожному етапі розробки програмного забезпечення, оскільки це визначає майбутню архітектуру системи, продуктивність та масштабованість, зручність розробки та підтримки, доступність розробних інструментів, а також безпеку та стійкість.

3.3.1 Мови програмування

C# – високорівнева мова програмування та одна з найпопулярніших мов (рис. 3.6) для розробки бекенду застосунків. Дана мова є хорошим вибором для написання бекенду, оскільки вона кросплатформенна, що означає, що

програми, написані на ній, можуть запускатися на різних операційних системах, таких як Windows, Linux та MacOS. Мова має велику кількість потужних фреймворків, таких як .NET Core, ASP.NET, Entity Framework, та багато інших.

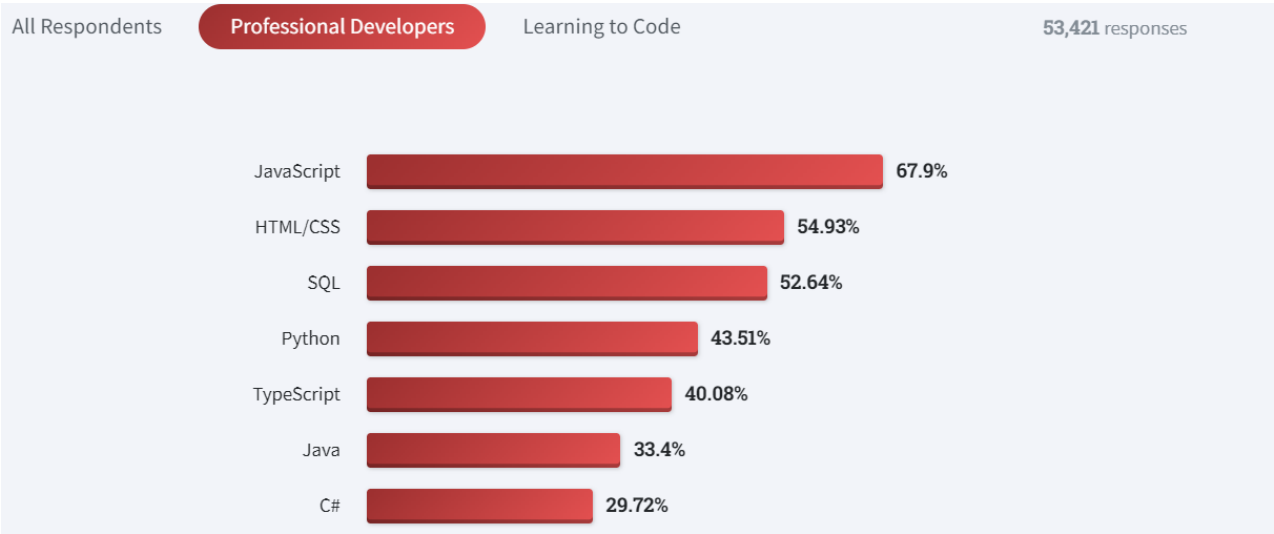

Рисунок 3.6 – Статистика популярності мов програмування згідно з StackOverflow Survey 2022

Фреймворки ASP.NET дозволяє розробляти швидкі та масштабовані бекенди, які можуть оброблювати великі обсяги даних. Крім цього, вбудована підтримка паралельної обробки даних дозволяє програмістам ефективно використовувати потужність сучасних процесорів. Загалом, C# є потужною та ефективною мовою програмування, яка дозволяє розробляти високопродуктивні та масштабовані бекенди.

TypeScript – це мова програмування, яка є підмножиною JavaScript. Одним з головних переваг TypeScript є те, що вона дозволяє програмістам використовувати типи даних при написанні коду, що зменшує кількість помилок та робить код більш прогнозованим та легше зрозумілим. TypeScript також підтримує останні версії стандарту ECMAScript, тому розробники можуть використовувати нові функції та можливості JavaScript безпосередньо

в своєму коді. Однією з основних переваг TypeScript для фронтенду є можливість розробки масштабованих та складних застосунків. TypeScript дозволяє розробникам зберігати структуру та прогнозованість свого коду, що спрощує роботу з великими проектами. Крім того, TypeScript підтримує концепцію «інтерфейсів», що дозволяє визначити типи даних та взаємодії між різними елементами програми. Це дозволяє покращити співпрацю між розробниками та зменшує кількість помилок при розробці.Ще однією перевагою TypeScript є наявність різноманітних інструментів, фреймворків та бібліотек, що дозволяють розробникам швидко та ефективно розробляти фронтенд застосунки. Наприклад, Angular – один з найпопулярніших фреймворків (рис. 3.7) для розробки фронтенду, який підтримує TypeScript та дозволяє створювати багатофункціональні та динамічні вебзастосунки. Також існують інші фреймворки, такі як React та Vue, які підтримують TypeScript та дозволяють розробникам створювати ефективні та зручні інтерфейси користувача.

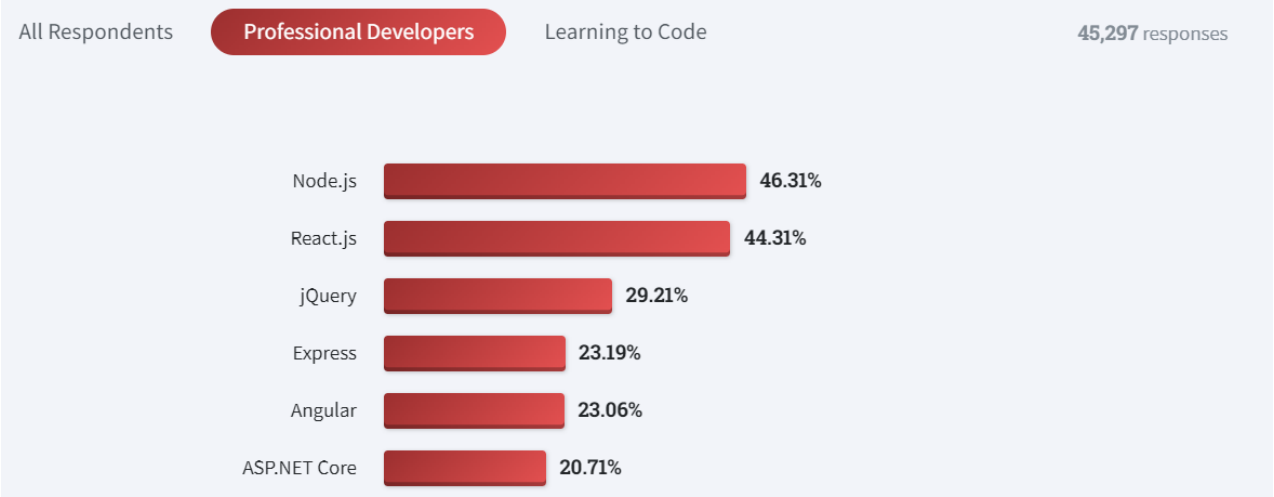

Рисунок 3.7 – Статистика популярності вебфрейворків згідно з StackOverflow Survey 2022

Загалом, TypeScript є потужною та гнучкою мовою програмування, яка забезпечує безпечну та продуктивну розробку фронтенду. Її здатність до

підтримки статичної типізації, розширення JavaScript та інструментів розробки дозволяє розробникам будувати більш безпечні та ефективні вебзастосунки.

3.3.2 Front-end технології

Angular – це JavaScript-фреймворк, призначений для розробки вебзастосунків з односторінковою архітектурою (SPA). Angular пропонує ряд інструментів для розробки високоякісних, масштабованих та ефективних вебзастосунків [21], таких як підтримка двостороннього зв'язку даних, компонентний підхід до розробки, залежність від впровадження, валідація даних та багато іншого.

Angular Material – це набір готових до використання компонентів, які базуються на матеріальному дизайні [11], розроблені для Angular. Він включає в себе компоненти, такі як форми, таблиці, кнопки, меню та інші, які можна використовувати для швидкої та легкої розробки вебінтерфейсів з привабливим дизайном.

Вибір Angular для розробки вебзастосунків має декілька переваг. Зокрема, Angular має велику та активну спільноту розробників, яка надає регулярні оновлення та підтримку фреймворку. Крім того, Angular пропонує компонентний підхід до розробки, що полегшує роботу з розробкою та підтримкою коду. Нарешті, статистика з опитувань StackOverflow Survey показує, що Angular є одним з найпопулярніших фреймворків для розробки вебзастосунків.

Отже, вибір Angular та Angular Material може бути доцільним для розробки вебзастосунків, якщо потрібен швидкий та ефективний спосіб створення вебінтерфейсу з привабливим дизайном, який буде підтримуватись і розвиватись протягом тривалого часу.

3.3.3 Back-end технології

ASP.NET Core – це відкритий фреймворк для розробки вебзастосунків, який забезпечує кросплатформенність, високу продуктивність, безпеку та простоту розробки [8]. Він має модульну архітектуру, яка дозволяє використовувати тільки необхідні компоненти [12], що сприяє швидкому розгортанню додатків та зменшенню обсягу коду.

PostgreSQL – це потужна об'єктно-реляційна база даних з відкритим кодом, яка забезпечує стандарти відмовостійкості, безпеки та високої продуктивності. Вона підтримує розширення, JSON та геодані, що робить її дуже гнучкою та можливою для використання в різних проектах [20].

ASP.NET Core є ідеальним вибором для створення кросплатформенних вебзастосунків, які мають бути швидкими та безпечними, а PostgreSQL надає досить високий рівень безпеки, тому її можна використовувати для зберігання конфіденційної інформації, такої як паролі, фінансові дані тощо. Крім того, PostgreSQL є дуже гнучкою та можливою для використання в різних проектах, які можуть потребувати складних запитів до бази даних.

Таким чином, використання ASP.NET Core та PostgreSQL є доцільним вибором для створення вебзастосунку пошуку коворкінгу.

3.3.4 Онлайн-сервіси

API сервісу Google Maps – це інтерфейс програмування додатків, який дозволяє розробникам використовувати дані та функції картографічної платформи Google Maps у своїх вебзастосунках.

Google Maps API має наступні переваги, які роблять його кращим вибором на ринку для створення застосунків, що взаємодіють з мапами:

- надійність та широка функціональність: API Google Maps забезпечує доступ до найбільш повної та актуальної бази даних мап, включаючи

інформацію про геолокацію, транспортні маршрути, місця відпочинку та інші корисні дані;

- простота використання: Google Maps API забезпечує зручний та інтуїтивний інтерфейс програмування, який дозволяє розробникам швидко та ефективно використовувати дані та функції платформи без необхідності в глибокому знанні картографії та геодезії;
- підтримка різних мов програмування та платформ: Google Maps API підтримує широкий спектр мов програмування, включаючи JavaScript, Python, Java, Ruby та інші, а також може бути використаний на різних платформах, включаючи веб-сайти, мобільні додатки та настільні додатки.

Ключові методи та функції API Google Maps включають:

1) Відображення мапи на вебсторінці;
2) Пошук місць на мапі;
3) Розрахунок маршрутів та навігація;
4) Отримання інформації про місця відпочинку та інші об'єкти на мапі;
5) Використання вбудованих засобів для малювання маршрутів, маркерів та інших об'єктів на мапі.

Додаткові функції та можливості API Google Maps включають:

- Візуалізація даних на мапі: Google Maps API дозволяє відображати різні типи даних на мапі, такі як графіки, діаграми, теплові карти та інші. Це дозволяє розробникам створювати високоякісні та інтерактивні візуалізації даних на мапі;
- Інтеграція з іншими сервісами Google: Google Maps API може бути легко інтегрований з іншими сервісами Google, такими як Google Places, Google Street View, Google Earth та інші. Це дозволяє розробникам

створювати розширені та багатофункціональні застосунки на основі даних мап Google;

- Геодезичні операції: Google Maps API дозволяє виконувати різні геодезичні операції, такі як визначення відстаней між точками, розрахунок координат місць на мапі та інші. Це дозволяє розробникам створювати застосунки, що пов'язані з геодезією та навігацією;
- Мобільна підтримка: Google Maps API має підтримку мобільних пристроїв та може бути використаний в мобільних застосунках. Це дозволяє розробникам створювати мобільні застосунки, які використовують дані мап Google та їх функції.

В загальному, API Google Maps є потужним та розширюваним інструментом для розробки вебзастосунків, що взаємодіють з мапами. Він надає доступ до повного набору функцій та даних мап Google, а також має легкий інтерфейс програмування та широку підтримку мов та платформ програмування.

Auth0 – це ідентифікаційний сервіс, який надає механізми аутентифікації та авторизації для вебзастосунків та мобільних додатків. Нижче подано короткий опис API Auth0:

- authentication API: цей API надає механізми аутентифікації, такі як аутентифікація користувача з використанням електронної пошти та пароля, аутентифікація з використанням соціальних мереж та ін;
- management API: цей API надає можливість здійснювати управління користувачами, клієнтами та іншими об'єктами Auth0;
- authorization API: цей API надає можливість здійснювати управління правами доступу до ресурсів відповідно до ролей користувачів;
- MFA API: цей API надає можливість налаштування багатофакторної автентифікації для додаткової безпеки.

Auth0 є потужним інструментом для забезпечення безпеки вебзастосунків та мобільних додатків, особливо коли йдеться про складні системи авторизації та управління доступом до ресурсів. Використання Auth0 дозволяє зосередитися на розробці функціоналу застосунку, не займаючись проблемами безпеки, тому що Auth0 надає готові механізми авторизації та аутентифікації, що дозволяє економити час та зусилля. Крім того, Auth0 підтримує різні мови програмування та платформи, що дозволяє використовувати його у різних проектах.

Висновки до розділу 3

В третьому розділі кваліфікаційної роботи бакалавра виконано проєктування вебзастосунку пошуку коворкінгу, а саме: створення UML та ERD діаграм. Під час проєктування вебзастосунку закріплено навички із розробки діаграм класів, компонентів та пакетів. Створені діаграми спрощують подальшу розробку системи та допомагають краще її зрозуміти.

Також було розглянуто обраний технологічний стек, аргументовано чому були обрані зазначені технології, їх особливості та переваги у порівнянні з іншими технологіями на ринку.

4 ПРОГРАМНА РЕАЛІЗАЦІЯ ВЕБЗАСТОСУНКУ ПОШУКУ КОВОРКІНГУ

4.1 Огляд дизайну вебзастосунку пошуку коворкінгу

Для створенні макетів дизайну вебзастосунку використано застосунок «Moqups». Усього створено 5 макетів основних екранів застосунку, а саме:

- головна сторінка застосунку;
- сторінка власника закладів;
- сторінка перегляду власників;
- сторінка створення закладу;
- сторінка оновлення закладу.

Головна сторінка застосунку (рис. 4.1) – те з чим будуть взаємодіяти більшість цільової авдиторії [2]. Основним елементом взаємодії користувача є мапа, на якій відображено маркери з локаціями. При кліку на маркер з лівої сторони користувачу показується детальна інформація про локацію. У правому кутку екрану наявний рядок для пошуку адрес, щоб зручніше і точніше переміщатися по мапі. Для зручності пошуку по критеріям знизу розташована панель з фільтрами. На тулбарі розміщена кнопка спливаючого меню, що дозволяє переходити по різним секціям вебзатосунку.

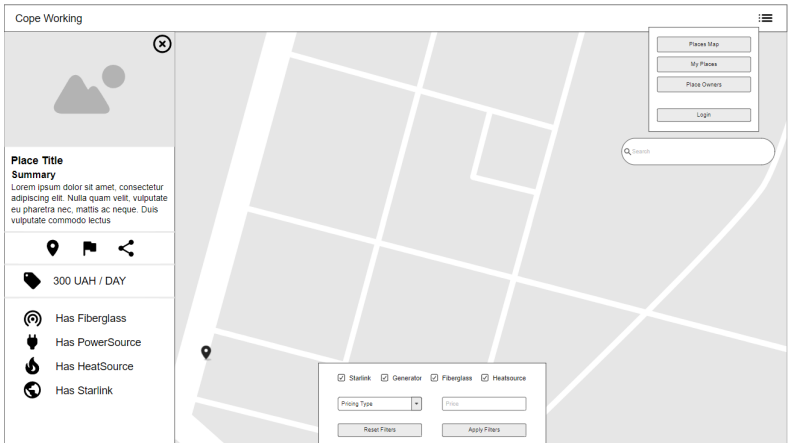

Рисунок 4.1 – Головна сторінка застосунку

Наступна за популярністю і важливістю є сторінка для відображення доданих місць власника закладу (рис. 4.2). Основними елементами сторінки є таблиця, дозволяє швидко переглянути усі додані користувачем заклади та бокова панель з мапою. Також для зручності користувача також була додані кнопка показу мапи, додання нового місця, а також кнопка, що підсвічує обране місце на мапі.

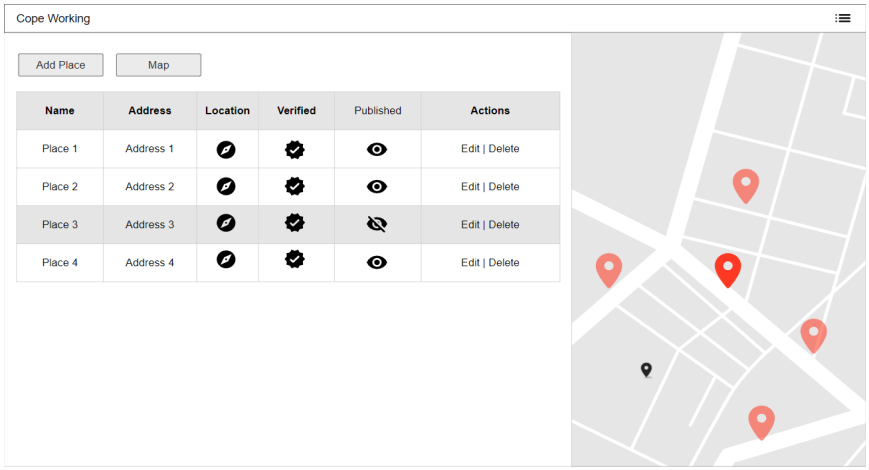

Рисунок 4.2 – Сторінка власника закладів

Сторінка адміністрування (рис. 4.3) дає можливість адміністратору застосунку переглянути зареєстрованих власників у системі та дані про них. Окрім перегляду для зручності адміністратора також розміщені кнопки швидкої взаємодії з користувачем – перегляд детальної інформації та блокування.

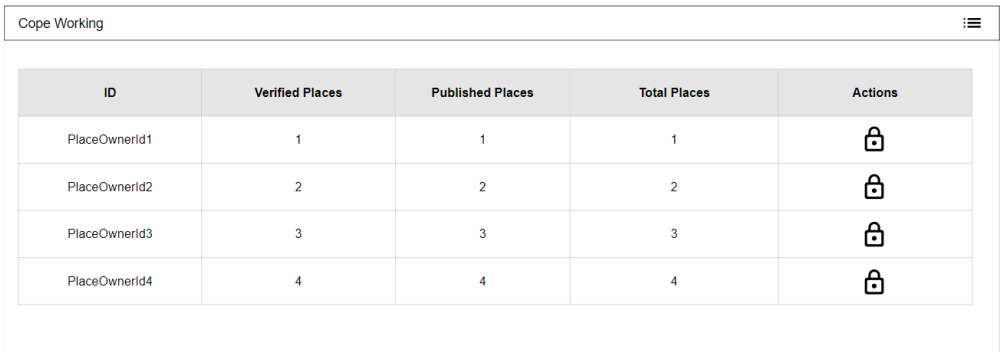

Рисунок 4.3 – Сторінка перегляду власників

Сторінка створення нового закладу (рис. 4.4) використовує пошагову форму, оскільки усі елементи форми не можливо розмістити на екрані і при цьому зберегти позитивний користувацький досвід. На першому кроці створення закладу власник взаємодіє з мапою, а на другому уточнює деталі інформації щодо закладу.

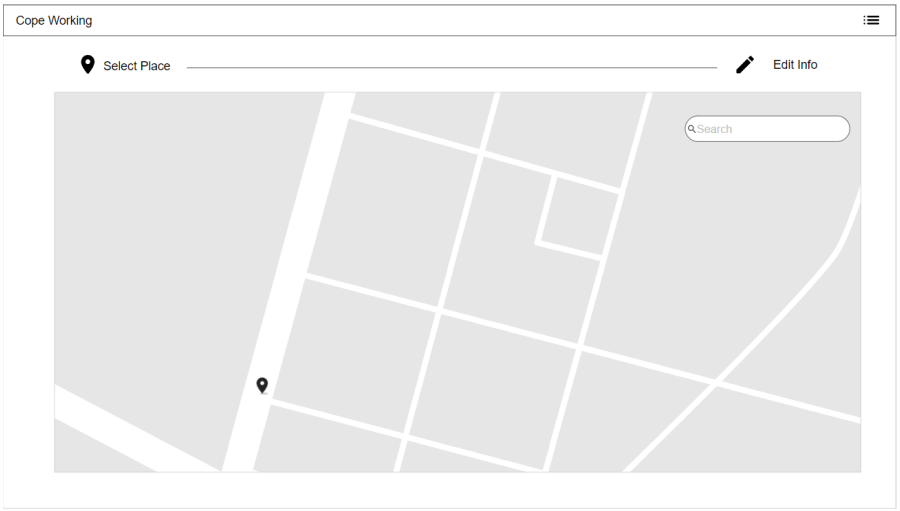

Рисунок 4.4 – Сторінка створення закладу

Парною сторінкою для сторінки створення закладу є сторінки оновлення інформації про заклад (рис. 4.5). На ній власник може змінити деяку інформацію про заклад, наприклад оновити цінову політику чи змінити його обладнання.

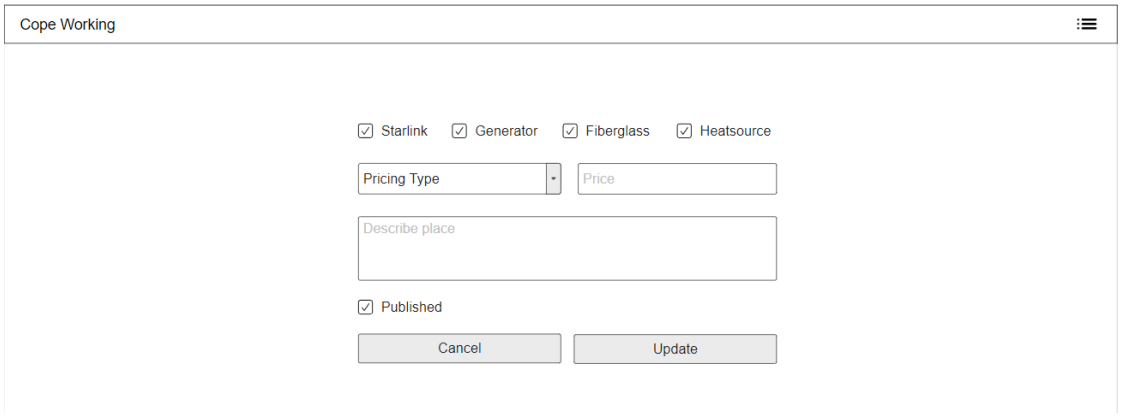

Рисунок 4.5 – Сторінка оновлення закладу

На цьому розгляд макетів дизайну завершено.

4.2 Кодування програмних компонентів back-end частини

4.2.1 Налаштування сервісу Auth0

Авторизація та автентифікація є важливими компонентами будь-якого веб-застосунку, оскільки вони дозволяють забезпечити безпеку даних користувачів та контролювати доступ до ресурсів [1].

Використання сторонніх сервісів для автентифікації та авторизації має декілька переваг [13]. По-перше, це дозволяє значно зменшити кількість коду, який потрібно написати для реалізації цих функцій власноруч. По-друге, сторонні сервіси зазвичай мають розгорнуту інфраструктуру для забезпечення безпеки, яка може бути складною для самостійної реалізації. По-третє, використання стороннього сервісу дозволяє стандартизувати процес автентифікації та авторизації в застосунку, що полегшує його розробку та підтримку.

Auth0 – це один зі сторонніх сервісів, який дозволяє розробникам додавати автентифікацію та авторизацію до своїх застосунків. Auth0 надає безпечну інфраструктуру для автентифікації та авторизації, підтримує багато різних методів автентифікації, включаючи логін та пароль, соціальні мережі, біометрію та інші.

Auth0 також має інтерфейс (рис. 4.6), що дозволяє додатково настроювати права доступу користувачів, створювати ролі та дозволи для кожного користувача чи груп користувачів, що дозволяє точно налаштувати рівні доступу до різних функцій та даних в застосунку.

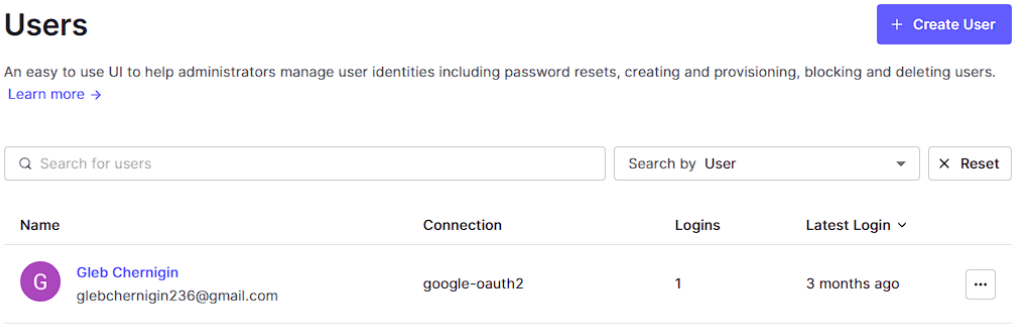

Рисунок 4.6 – Приклад інтерфейсу Auth0 для взаємодії з користувачами

Функціонал обмеження реєстрації за геолокацією дозволяє обмежити доступ до застосунку тільки користувачам з певних географічних регіонів. Наприклад, якщо застосунок створений для користувачів з певної країни, можна обмежити реєстрацію для інших країн, що забезпечить додатковий рівень безпеки та підвищить ефективність застосунку. Для цього Auth0 надає можливість налаштувати правила для кожного клієнта (рис. 4.7), що дозволяє точно налаштувати цей функціонал.

Рисунок 4.7 – Імплементація обмеження реєстрацій за регіоном

Для того, щоб налаштувати Auth0 для роботи з ASP.NET Core та Angular, для початку створимо об’єкти «Application» (рис. 4.8) та «API» (рис. 4.9), використовуючи вебінтерфейс Auth0.

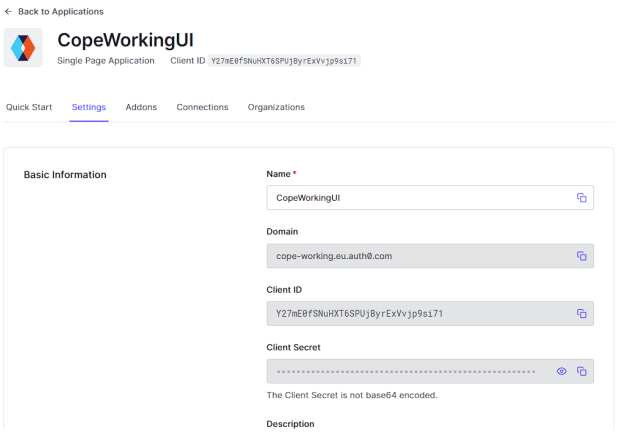

Рисунок 4.8 – Створений об’єкт «Application»

Application в Auth0 – це клієнтський додаток, який використовується для автентифікації та авторизації користувачів. Application отримує токен доступу (access token) та/або токен оновлення (refresh token) для доступу до захищених ресурсів, які надаються за допомогою Auth0.

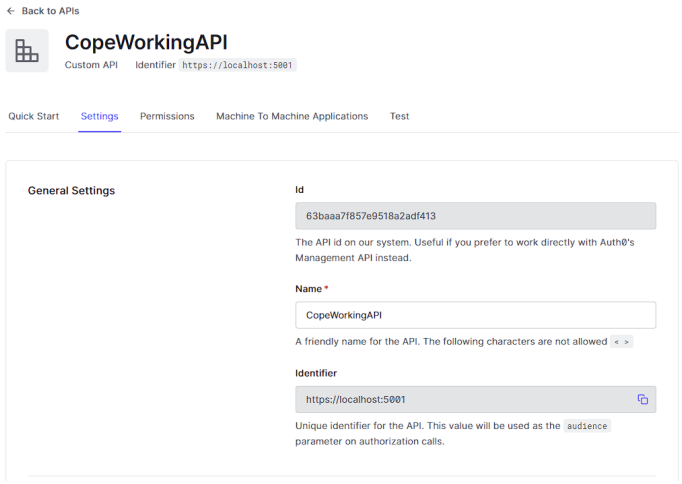

Рисунок 4.9 – Створений об’єкт «API»

API в Auth0 – це захищений веб-сервіс, до якого можна отримати доступ за допомогою токенів доступу, які видані клієнтськими додатками. API може бути використаний для захисту доступу до ресурсів, які надаються клієнтам. Auth0 може використовуватися для реалізації захисту API з допомогою токенів доступу, які надаються за допомогою Auth0.

4.2.2 Налаштування сервісу Google Maps

Для налаштування доступу застосунку до Google Maps API потрібно отримати API-ключ. Для цього створено новий проєкт (рис. 4.10) у консолі розробника Google та зареєстровано новий ключ API.

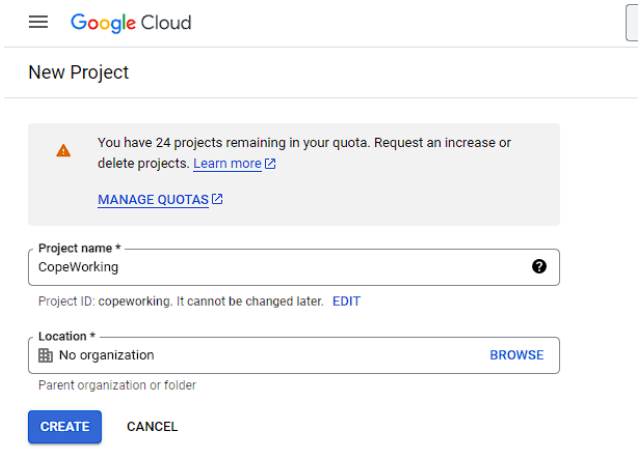

Рисунок 4.10 – Створення проєкту

Для ключа налаштовано з обмеженнями на використання лише від певних доменів (рис. 4.11). Крім того, враховано обмеження та ліміти, пов'язані з використанням Google Maps API, а саме була обмежена кількість запитів до API, щоб уникнути перевищення лімітів використання та забезпечити стабільну роботу API на сторінці.

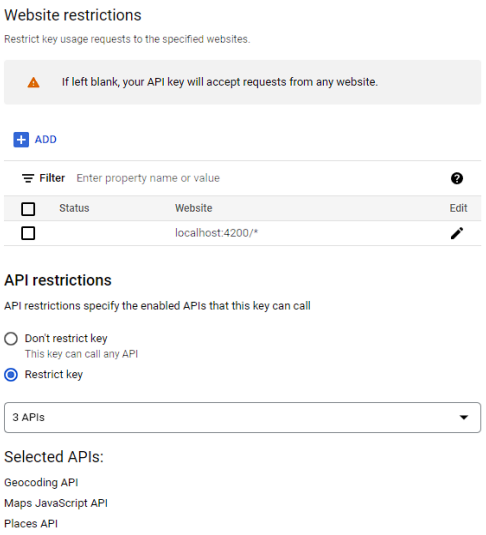

Рисунок 4.11 – Налаштування обмежень для ключа

4.2.3 Отримання кластеризованих даних про заклади

Запит для отримання інформації про заклади дозволяє отримати дані для відображення їх на мапі, але оскільки закладів може бути тисячі, то для ефективності у застосунку використовується кластеризація – усі заклади у певному радіусі об’єднують і відображаться [10], як один маркер на мапі. Код на рисунку 4.12 відповідає за логіку формування даного запиту.

```
public async Task<PlaceClusterDto[]> Handle(GetPlaceClustersQuery request, CancellationToken cancellationToken)
{
    var (west:double, south:double, east:double, north:double) = request.Bounds;
    var parameters:{minX,minY,...} = new
    {...};
    var (hasPowerSource:bool?,
        hasHeatSource:bool?,
        hasStarlink:bool?,
        hasFiberglass:bool?,
        pricingType:PricingType?,
        pricingRate:decimal?) = request.PlaceFilter;
    var displayAsClusters:bool = request.Zoom < 15;
    var builder = new SqlBuilder()
        .Select(sql: "id")
        .Select(sql: "location")
        .Select(displayAsClusters
            ? "st_clusterdbscan(location, @minDistance, @minPlacesInCluster) over () as cluster_id"
            : "null as cluster_id")
        .Where(sql: "published = true")
        .WhereIf(pricingType.HasValue, sql: $"pricing_type = @{nameof(pricingType)}", parameters: new { pricingType })
        .WhereIf(pricingRate.HasValue, sql: $"pricing_rate = @{nameof(pricingRate)}", parameters: new { pricingRate })
        .WhereIf(hasPowerSource.HasValue, sql: $"traits_has_power_source = @{nameof(hasPowerSource)}", parameters: new { hasPowerSource })
        .WhereIf(hasHeatSource.HasValue, sql: $"traits_has_heat_source = @{nameof(hasHeatSource)}", parameters: new { hasHeatSource })
        .WhereIf(hasFiberglass.HasValue, sql: $"traits_has_fiberglass = @{nameof(hasFiberglass)}", parameters: new { hasFiberglass })
        .WhereIf(hasStarlink.HasValue, sql: $"traits_has_starlink = @{nameof(hasStarlink)}", parameters: new { hasStarlink });
    var clusteredPlacesTpl:Template? = builder.AddTemplate(ClusteredPlacesQuery, parameters);
    using var connection = _connectionFactory.CreateConnection();
    var places:IEnumerable<dynamic>? = await connection.QueryAsync(
        clusteredPlacesTpl.RawSql,
        clusteredPlacesTpl.Parameters); // Task<IEnumerable<...>>
    return _mapper.Map<PlaceClusterDto[]>(places);
}
```

Рисунок 4.12 – Логіка формування запиту

Спочатку з запиту отримуються межі, на основі яких визначаються параметри кластерування (найменша відстань між місцями, щоб їх згрупувати в один кластер). Після цього, за допомогою SqlBuilder [15] формується запит до бази даних, який містить умови відбору місць за різними критеріями (наприклад, наявність джерел енергії, тип цінування тощо).

Запит (рис. 4.13) використовує змінні, які були визначені в попередньому фрагменті коду, тобто minX, minY, maxX, maxY, minDistance, minPlacesInCluster і expandFactor. Запит складається з двох частин, які об'єднані за допомогою оператора «UNION».

Перша частина запиту формує кластери на основі певних критеріїв. Вона використовує функції ST_* [14], що дозволяють працювати з геоданими. У фрагменті «select» визначаються поля, які повертає запит. У фрагменті «from» вказується таблиця, над якою виконується запит. У фрагменті «where» визначаються умови відбору місць за певними критеріями (наприклад, наявність джерел енергії, тип цінування тощо).

```
private static string ClusteredPlacesQuery => """
        with place_clusters as (
        select
            /**select**/
            from places
            /**where**/ and st_contains(st_expand(st_makeenvelope(@minX, @minY, @maxX, @maxY, 4326), @expandFactor), location)
        )
        select
            array_length(array_agg(location), 1) "TotalPlaces",
            st_centroid(st_collect(array_agg(location))) as "Position",
            null as "PlaceId"
        from place_clusters
        where cluster_id is not null
        group by cluster_id
        union
        select 1 as "TotalPlaces", location as "Position", id as "PlaceId"
        from place_clusters
        where cluster_id is null
    """;
```

Рисунок 4.13 – Логіка формування запиту

У другій частині запиту формуються окремі місця, які не належать до кластерів [16]. Після виконання запиту результати обробляються в попередньому фрагменті коду, де вони перетворюються в масив об'єктів PlaceClusterDto.

4.2.4 Отримання даних з Google Place API

Для отримання більш детальної інформації про заклад [9] при його створенні застосунок використовує Google Places API.

Логіка взаємодії з Google Places API інкапсульована у PlaceInfoService, що дозволяє легко змінити логіку у разі чого та полегшує тестування застосунка. На рисунках 4.14-4.15 наведені фрагмент коду, що відповідають за отримання даних про заклад.

```
public async Task<PlaceDetails?> GetPlaceDetails(PlaceId id)
{
    var requestUri:string = BuildPlaceRequestUri(id.Value);
    var placeDetailResult = await _httpClient.GetFromJsonAsync<PlaceDetail>(requestUri);
    if (placeDetailResult == null) return null;

    var placeDetail = placeDetailResult.Result;
    if (!placeDetail.Types.Contains("establishment"))
    {
        throw new ArgumentException(message:"Place is not an establishment");
    }

    return new PlaceDetails
    {
        Id = placeDetail.Id,
        Address = placeDetail.Address,
        Name = placeDetail.Name,
        Location = new Point(x:placeDetail.Geometry.Location.Lng, y:placeDetail.Geometry.Location.Lat),
        PhoneNumber = placeDetail.PhoneNumber?.Replace(oldValue:" ", newValue:"")
    };
}
```

Рисунок 4.14 – Логіка отримання інформації про заклад

Перший фрагмент коду описує метод GetPlaceDetails, який отримує інформацію про конкретне місце. Цей метод приймає об'єкт PlaceId як параметр, який містить унікальний ідентифікатор місця [18].

Якщо результат запиту дорівнює null, то метод повертає null. Інакше перевіряється, чи місце є закладом (властивість Types містить список категорій, які описують місце) і викликається виняток ArgumentException у разі невідповідності.

```
private string BuildPlaceRequestUri(string placeId)
{
    const string endpoint =
        "place/details/json?fields=formatted_address,name,geometry,place_id,international_phone_number,types";
    return $"{_settings.Url}/{endpoint}&place_id={placeId}&key={_settings.ApiKey}";
}
```

Рисунок 4.15 – Логіка формування посилання

Другий фрагмент коду містить приватний метод BuildPlaceRequestUri, який формує URL для HTTP-запиту до API за допомогою заданого ідентифікатора місця. Цей метод будує URL, що включає необхідну інформацію про запит до API, включаючи список полів, що повинні бути повернені, та ключ API, який використовується для автентифікації запиту. Це дозволяє автентифікуватися та отримувати доступ до послуг сервісу.

4.3 Кодування програмних компонентів front-end частини

4.3.1 Підключення та використання бібліотеки google-maps

Для підключення Google Maps до Angular проєкту створено клас GoogleMapsLoaderGuard (рис. 4.16), який забезпечує завантаження Google Maps API з підключеним ключем API. У даному класі відправляється запит на отримання бібліотеки JavaScript API для Google Maps [3].

Якщо Google Maps API ще не був завантажений, то сервіс HttpClient відправляє запит за допомогою методу jsonp. Після завантаження API, GoogleMapsLoaderGuard зберігає флаг googleMapsLoaded, щоб запобігти повторному завантаженню Google Maps API.

```
@Injectable({
  providedIn: 'root'
})
export class GoogleMapsLoaderGuard implements CanActivate {

  private googleMapsLoaded = false;

  private readonly loadUrl: string;

  constructor(private http: HttpClient) {
    this.loadUrl = `https://maps.googleapis.com/maps/api/js?key=${environment.mapKey}&libraries=places`
  }

  public canActivate(
    route: ActivatedRouteSnapshot,
    state: RouterStateSnapshot): Observable<boolean | UrlTree> | Promise<boolean | UrlTree> | boolean | UrlTree {
    if (this.googleMapsLoaded) {
      return true;
    } else {
      return this.http.jsonp(this.loadUrl, 'callback')
        .pipe(
          map(() => true),
          tap(() => this.googleMapsLoaded = true),
        );
    }
  }
}
```

Рисунок 4.16 – Логіка класу GoogleMapsLoaderGuard

Далі у місцях застосунку, де потрібна робота з мапою, у конфігурації маршрутизації зазначається, що GoogleMapsLoaderGuard був активним, перед тим, як можна буде завантажити компонент.

Це забезпечує, що Google Maps API буде завантажено перед тим, як буде показано компонент, що використовує мапу. На рисунку 4.17 наведено приклад конфігурації маршрутизації для модуля PlacesManagement.

```
const routes: Routes = [
  { path: '', component: PlacesViewComponent, canActivate: [GoogleMapsLoaderGuard] },
  { path: 'new', component: CreatePlaceViewComponent, canActivate: [GoogleMapsLoaderGuard] },
  { path: ':placeId/update', component: UpdatePlaceViewComponent }
];

@NgModule({
  imports: [RouterModule.forChild(routes)],
  exports: [RouterModule]
})
export class PlacesManagementRoutingModule {
}
```

Рисунок 4.17 – Приклад конфігурації маршруштизації

Конфігурація включає в себе декларування NgModule та маршрутів для модулю затосунку, які складаються з маршрутів, компоненту, що буде відобратажатися для маршруту та опціональне вказання guards, що контролюють активацію маршруту.

Відображення карти (рис. 4.18) здійснюється за допомогою тегу <google-map>. Цей тег приймає різні параметри, що визначають вигляд та поведінку карти. Наприклад, параметри height та width встановлюють розміри карти у відсотках від розміру батьківського контейнера. Параметр center встановлює початкове положення карти за координатами (широта та довгота). Параметр options дозволяє встановити додаткові опції для карти, такі як тип карти, масштабування тощо. Параметр zoom встановлює масштаб карти на початку.

```
Go to component
<google-map
  height="100%"
  width="100%"
  [center]="{lat: 50.4, lng: 30.5}"
  [options]="mapOptions"
  [zoom]="currentZoom"
>
  <map-marker
    *ngFor="let marker of markers$ | async; trackBy: trackPlace"
    [position]="marker.position"
    [label]="marker.labelOptions!"
    (mapClick)="onMarkerClick(marker)">
  </map-marker>
</google-map>
```

Рисунок 4.18 – Приклад конфігурації маршруштизації

Крім того, компонент мапи містить тег <map-marker>, який відповідає за відображення маркерів на карті [4]. Для кожного маркера встановлюються його координати та параметри відображення (наприклад, напис на маркері).

4.3.2 Відображення інформації про заклад

PlaceInfoComponent (рис. 4.19) відповідає за відображення інформації про заклад. Він містить в собі ряд елементів, таких як фото закладу, назву, опис, адресу та інші характеристики, які можуть бути корисні для користувача. Крім того, компонент також містить кнопки [5], які дозволяють користувачам взаємодіяти з закладом, наприклад, перейти до нього на мапі або поділитися посиланням на нього.

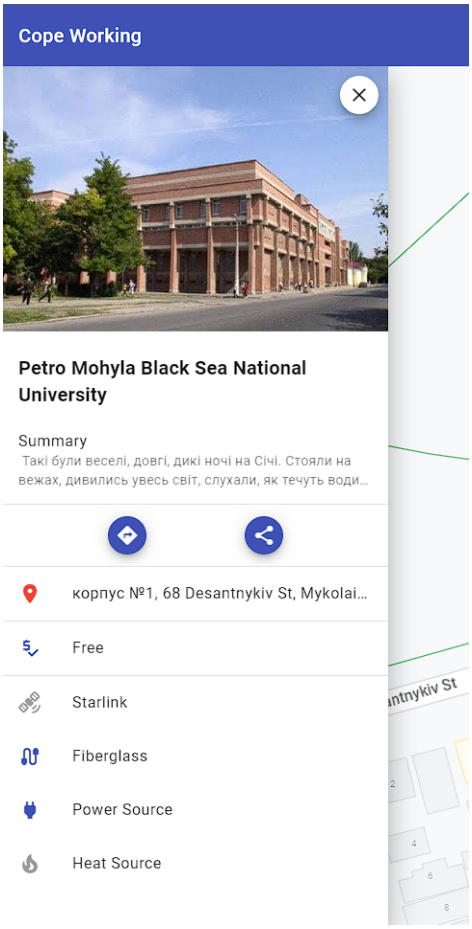

Рисунок 4.19 – Вигляд PlaceInfoComponent

Також компонент використовує декілька директив Angular, таких як ngIf, ngFor та ngSwitch, для того, щоб динамічно відображати різні елементи в залежності від наявності деяких характеристик і містить кнопки дій, такі як

«навігація» та «поділитися», які перенаправляють користувача на відповідні сторінки у застосунку Google Maps.

Висновки до розділу 4

У четвертому розділі кваліфакаційної роботи розлянуто кодування програмних компонентів back-end та front-end частин вебзастосунку пошуку коворкінгу. В розділі розглянуто складні частини застосунку, а саме:

- налаштування обмеження реєстрації за регіоном;
- отримання інформації з Google Places API;
- підключення та використання Google Maps;
- кластеризацію точок у просторі.

Під час кодування програмних компонентів було закріплено навички роботи з мовами програмування C# та TypeScript, знання фреймворків ASP.NET Core та Angular і сервісів Auth0 та Google Maps.

ВИСНОВКИ

В результаті виконання кваліфакаційної роботи бакалавра автоматизовано і покращено процес пошуку закладу для роботи за рахунок розробки програмного забезпечення – вебзастосунок пошуку коворкінгу.

Для досягнення поставленої мети вирішено поставлені завдання:

- проаналізовано предметну область і аналоги ПЗ;
- визначено функціоналу застосунку;
- створено блок-схеми та діаграми роботи застосунку;
- зібрано та проаналізовано методичні рекомендації;
- розроблена frontend-частини вебзастосунку на базі фреймворку Angular;
- розроблена backend-частини вебзастосунку на базі фреймворку ASP.NET.

Завдяки ретельному дослідженню предметної області та глибокому аналізу аналогів ПЗ була сформована детальна специфікація вебзастосунку пошуку коворкінгу, яка враховує та виправляє недоліки існуючих програмних рішень, при цьому пропонуючи нові способи та підходи до вирішення поставленої задачі.

Використовуючи сформовану специфікацію обрано найбільш доцільні шляхи вирішення задач та актуальні технології. Розроблені UML-діаграми демонструють стуктуру та алгоритми роботи системи.

Результатом проведеної роботи є вебзастосунок пошуку коворкінгу.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Anugrah I. G., Fakhruddin M. A. R. I. Development Authentication and Authorization Systems of Multi Information Systems Based REst API and Auth Token. *Innovation Research Journal*. Vol. 1, Issue 2. 2011. P. 127–132.
2. Billsus D., Brunk C. A., Evans C. et al. Adaptive interfaces for ubiquitous web access. *Communications of the ACM*. Vol. 45, Issue 5. 2018. P. 34–38.
3. Clow M., Clow M. RxJS with Angular. *Angular 5 Projects: Learn to Build Single Page Web Applications Using 70+ Projects*. 2018. P. 309–313.
4. Daniels P., Atencio L. RxJS in Action. Simon and Schuster, 2017. P. 200.
5. Desai R. Google Material Design–The User Interface Development Notion. *National Conference on Advances in Computer Science Engineering & Technology*. 2017. P. 150.
6. Eriksson H.-E., Penker M. Business modeling with UML. *New York*. Vol. 12, 2000. P. 444.
7. Evans E., Evans E. J. Domain-driven design: tackling complexity in the heart of software. Addison-Wesley Professional, 2004. P. 500–525.
8. Freeman A. Working with Visual Studio Code. 2020. P. 300.
9. Hu S., Dai T. Online map application development using Google Maps API, SQL database, and ASP .NET. *International Journal of Information and Communication Technology Research*. Vol. 3, Issue 3. 2016. P. 343.
10. Juba S., Vannahme A., Volkov A. Learning PostgreSQL. Packt Publishing Ltd, 2015. P. 210.
11. Kotaru V. K. Angular for Material Design. Springer. 2020. P 110.
12. Lock A. ASP. NET core in Action. Simon and Schuster. 2021. P. 333.
13. Madden N. API security in action. Simon and Schuster. 2020. P. 95.
14. Obe R., Hsu L. S. PostGIS in action. Simon and Schuster. 2021. P. 197.
15. Price M. J. C# 9 and. NET 5–Modern Cross-Platform Development: Build

intelligent apps, websites, and services with Blazor, ASP. NET Core, and Entity Framework Core using Visual Studio Code. Packt Publishing Ltd. 2020. p. 560.

16. Russell C. T. Geophysical coordinate transformations. *Cosmic electrodynamics*. Vol. 2, Issue 2. P. 184–196.
17. Seemann M. Dependency injection in. NET. Manning New York, 2012. p. 299.
18. Svennerberg G. Beginning google maps API 3. Apress, 2010. p. 135.
19. Tonella P., Potrich A. Package Diagram. *Reverse Engineering of Object Oriented Code*. 2005. P. 133–154.
20. Wang X., Wang J. Using clustering methods in geospatial information systems. *Geomatica*. Vol. 64, Issue 3. P. 347–361.
21. Wilken J. Angular in Action. Simon and Schuster, 2018. p. 232.