ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра інженерії програмного забезпечення

Пояснювальна записка

до бакалаврської роботи
на ступінь вищої освіти бакалавр

на тему: «**РОЗРОБКА СЕРВІСУ "OPENPOST" ПОШУКУ ПОПУТНОГО ТРАНСПОРТУ ДЛЯ ПЕРЕДАЧІ ПОСИЛОК МОВОЮ C#**»

Виконав: студент 5 курсу, групи ППЗ-51
спеціальності 121 «Інженерія програмного забезпечення»
Горбань А.М.
Керівник Золотухіна О.А.
(прізвище, ініціали)
Рецензент _____
(прізвище, ініціали)
Нормоконтроль _____
(прізвище, ініціали)

Київ – 2023

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут Інформаційних технологій
Кафедра Інженерії програмного забезпечення
Ступінь вищої освіти - «Бакалавр»
Спеціальність – 121, інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри
інженерії програмного забезпечення
_____О.В. Негоденко
“____” _____2023 року

З А В Д А Н Н Я
НА БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТУ

Горань Андрій Миколайович
(прізвище, ім’я, по батькові)

1. Тема роботи: «Розробка сервісу "OpenPost" пошуку попутного транспорту для передачі посилок мовою c#»
Керівник роботи _____Золотухіна О.А. к.т.н, доцент_____,
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від “24 лютого” 2023 року №26.
2. Строк подання студентом роботи____1 червня 2023 р._____
3. Вихідні дані до роботи:
   3.1 JetBrains Rider;
   3.2 VS Code;
   3.3 Diagrams.net;
   3.4 Офіційна документація мови програмування C#;
   3.5 Офіційна документація платформи .Net;
   3.6 Офіційна документація фреймворку ASP.Net;
   3.7 Офіційна документація фреймворку EntityFramework;
   3.8 Офіційна документація мови програмування TypeScript;
   3.9 Офіційна документація фреймворку Angular;
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
   4.1 Аналіз предметної області та визначення особливостей процесу пошуку попутного транспорту для передачі посилок;
   4.2 Постановка задач та вимог до сервісу пошуку попутного транспорту для передачі посилок

4.3 Проектування системи
4.4 Розробка програмного забезпечення
4.5 Тестування програмного забезпечення
4.6 Опис реалізації програмного забезпечення
5. Перелік графічного матеріалу
5.1 Об’єкт, предмет, мета дослідження
5.2 Задачі дипломної роботи
5.3 Аналіз аналогів
5.4 Вимоги до веб сервісу
5.5 Засоби розробки
5.6 Узагальнена діаграма використання
5.7 Діаграма компонентів додатку
5.8 Діаграма моделі даних географічних об'єктів
5.9 Схема розгортання даних до бд з відкритих ресурсів GeoNames
5.10 Діаграма моделі даних заявок на доставку та маршрутів
5.11 Діаграма моделі даних користувачів
5.12 Діаграма послідовності авторизації на основі jwt токену
5.13 Екранні форми
5.14 Апробація результатів дослідження
5.15 Висновки
6. Дата видачі завдання 25 лютого 2023 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської роботи	Строк виконання етапів роботи	Примітка
1	Аналіз предметної області	15.02.2023	виконано
2	Дослідження існуючих інструментів	22.02.2023	виконано
3	Постановка задачі	01.03.2023	виконано
4	Вибір та розробка архітектури додатку	03.03.2023	виконано
5	Програмна реалізація серверної частини	10.03.2023	виконано
6	Програмна реалізація клієнтської частини	07.04.2023	виконано
7	Тестування та оптимізація розробленого ПЗ	05.05.2023	виконано
8	Оформлення пояснювальної записки	12.05.2023	виконано
9	Розробка обов’язкових демонстраційних матеріалів	16.05.2023	виконано
10	Попередній захист роботи	19.05.2023	виконано
11	Подання роботи в деканат	01.06.2023	виконано

Студент _____ Горбань А.М.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Золотухіна О.А.
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Текстова частина бакалаврської роботи 80 с., 42 рис., 4 таблиці, 17 джерел.

Об'єкт дослідження – процес пошуку попутного транспорту для передачі посилок.

Предмет дослідження – програмне забезпечення для автоматизації пошуку попутного транспорту для передачі посилок.

Мета роботи – спростити пошук попутного транспорту для передачі посилок за рахунок автоматизації процесу з використанням веб-сервісу, розробленого мовою C#.

Методи дослідження – моделювання програмного забезпечення, методи об'єктно-орієнтованого програмування, методи тестування.

В роботі проведено аналіз особливостей процесу пошуку попутного транспорту для передачі посилок. Визначено основні поняття, об'єкти та етапи процесу. Розглянуто існуючі засоби та інструменти пошуку попутного транспорту для передачі посилок, визначено їх переваги та недоліки. На сонові аналізу визначені основні функціональні та нефункціональні вимоги до сервісу. Виконано проектування сервісу з використанням UML діаграми. Розроблено програмне забезпечення серверного та клієнтського додатку мовами C# та TypeScript.

Здійснено тестування розробленого продукту. Результати тестування підтвердили відповідність роботи основного функціоналу додатку поставленим вимогам. Користувачі можуть створювати заявки та маршрути транспорту і здійснювати по ним пошук. Забезпечено канал зв'язку між учасниками процесу у вигляді обміну текстовими повідомленнями.

Галузь використання – як окремі користувачі у яких є потреба в передачі посилок всередині країни або за кордон, так і малі підприємства без власного автопарку та логістики.

ПОШУК ПОПУТНОГО ТРАНСПОРТУ, ЛОГІСТИКА, ВЕБ-СЕРВІС, ТРАНСПОРТ, СПІЛЬНЕ ВИКОРИСТАННЯ ТРАНСПОРТУ, C#, TYPESCRIPT.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

UML	-	Unified Modeling Language
БД	-	База даних
SPA	-	Односторінковий додаток
ПЗ	-	Програмне забезпечення
API	-	Прикладни́й програмний інтерфейс
CRUD	-	створення, читання, оновлення і вилучення

ВСТУП

Сучасний світ що характеризується стрімким розвитком технологій, платформ електронної комерції, де спостерігається збільшення попиту на експрес-доставку протягом одного дня, та зростанням рівня міграції стає все більш залежним від ефективної логістики та способів доставки посилок. Традиційні способи доставки стикаються з рядом проблем, таких як високі витрати, повільність та негативний вплив на навколишнє середовище. Також такий чинник як війна в країні призводить до серйозних перешкод у забезпеченні стабільної та ефективної роботи традиційної системи доставки посилок, в таких умовах також постає питання способів доставки гуманітарних вантажів. Все це зумовлює необхідність пошуку більш швидких та ефективних альтернатив традиційним способам доставки посилок. Одним з таких способів є спільне використання ресурсів шляхом передачі посилок попутним транспортом. Передача посилок попутним транспортом – це краудсорсингова модель використання транспорту, з наступними перевагами:

– швидкість обробки запиту на доставку;

– низька вартість доставки для замовника;

– зниження вартості поїздки або доставки основного вантажу за рахунок використання вільного місця;

– зниження ризиків відсутності наявного транспорту;

– зменшений вплив на екологію за рахунок спільного та ефективного використання транспорту.

Метою роботи є спрощення процесу пошуку попутного транспорту для передачі посилок за рахунок автоматизації процесу з використанням веб-сервісу.

Об’єкт дослідження – процес пошуку попутного транспорту для передачі посилок.

Предмет дослідження – програмне забезпечення для автоматизації пошуку попутного транспорту для передачі посилок.

Розглянуті існуючі аналоги інструментів пошуку попутного транспорту для передачі посилок мають як свої переваги так і недоліки. У частини з них відсутні спеціалізовані інструменти та фільтри пошуку маршрутів попутного транспорту, що потребує додаткових зусиль та часу, мають низький рівень безпеки а відсутність пошуку по проміжним точкам маршруту знижує ефективність використання попутного транспорту. Інша частина має високий рівень автоматизації та прийнятний рівень безпеки, але також мають значні обмеження по типу транспорту та посилок, географічні та мовні обмеження. Для пошуку по містам необхідні повні збіги певною мовою, це не враховує назв на нативній мові країна або місцевих назв, що може ускладнити пошук для користувачів з певних регіонів. Тож функціональні вимоги в даній бакалаврській роботі було сформовано з врахуванням особливостей та недоліків існуючих аналогів. При проектуванні та побудові були використані сучасні методи проектування та архітектурні патерни розробки веб додатку.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ РОЗРОБКИ СЕРВІСУ ПОШУКУ ПОПУТНОГО ТРАНСПОРТУ ДЛЯ ПЕРЕДАЧІ ПОСИЛОК

1.1 Особливості процесу пошуку попутного транспорту для передачі посилок

Передача посилок попутним транспортом являє собою краудсорсингову модель використання транспорту. Такий процес як альтернатива традиційним способам доставки покликаний вирішити наступні завдання [1][2]:

– Зменшення часу очікування на обробку заявки.

– Зниження вартості доставки за разунок спільного використання транспорту.

– Зменшення ризиків відстуності доступного транспорту у власному автопарку.

Учасників процесу можна поділити на дві основні групи:

– Відправник — учасник процесу який має необхідність передати посилку, для цього відправник здійснює пошук попутного транспорту що маршрут якого проходить через точку відправлення та точку прибуття посилки.

– Водій попутного транспорту — учасник з наявним транспортом який слідує певному маршруту і має можливість взяти посилку у відправкика проїжджаючи місто відправки та передати отримувачу проїжджаючи місто прибуття посилки. Також замість водія може виступати учасником пасажир транспорту що готовий взяти невелику посилку в якості свого багажу та доставити його в пункт прибуття.

Виділено основні напрямки застосування попутного транспорту для передачі посилок[1]:

– Експрес-передача відправлень в межах країни. Для цієї мети доцільно використовувати спільний вантажний транспорт для великих відправлень,

особистий транспорт для відправлень середнього обсягу, пасажирський транспорт та вільне місце в багажі пасажирів для невеликих відправлень.

– Міжнародна передача відправлень. До вищезазначених видів транспорту можна додати вільне місце в багажі пасажирів повітряного транспорту. Важливо враховувати законодавство та митні правила країн, кордони яких перетинаються.

– Експрес-передача відправлень в межах міських агломерацій. До доступних видів транспорту доцільно додати велосипедистів та операторів інших видів малого транспорту. В межах міської території можуть бути актуальними послуги доставки для невеликих торгових підприємств та електронних комерційних платформ, які не мають власного автопарку для безпосередньої доставки до споживачів, від пунктів видачі до місця призначення.

Розв'язання даної проблеми може бути досягнуте шляхом застосування різних підходів, включаючи інформаційні технології. На даний час найбільш поширеними методами вирішення проблеми в рамках інформаційних технологій є розміщення оголошень у соціальних мережах або оголошень на спеціалізованих веб-сайтах дошках оголошень (наприклад, OLX[5]) з інформацією про необхідність передачі вантажу, місце відправлення та прибуття, або публікація даних щодо маршруту подорожі та можливості доставки вантажу водієм транспортного засобу. Такі сійти мають велику аудиторія але зазвичай на них відсутні спеціалізовані інструменти пошуку попутного транспорту що робить пошук мало ефективним. Також існують спеціалізовані сервіси для пошуку попутного транспорту, де водій та відправник можуть розмістити свої заявки, а сервіс автоматично забезпечує їх взаємозв'язок. Крім того, поширеним способом вирішення проблеми без застосування інформаційних технологій є відвідування відправником місць відправлення транспортних засобів (наприклад, автобусний вокзал) з метою знаходження водія, який прямує до потрібного пункту доставки та здатний передати вантаж, та такий спосіб не можливо автоматизувати і має вкрай низьку ефективність так як потребує особистої присутності в точках збору і

відправлення транспорту і не дає ніяких гарантій наявності доступного транспорту.

Було визначено що основний приклад послідовності пошуку попутного транспорту для передачі посилок за допомогою інформаційних технологій виглядає наступним чином. Спочатку відправник на одній з веб платформ здійснює пошук оголошень про перевезення по маршруту що його цікавить, якщо є співпадіння то відправник зв’язується з водієм попутного транспорту для того що б домовитися про відправку. Якщо співпадінь немає або з знайденими водіями не вдалося дійти домовленостей публікую власне оголошення про потребу в передачі посилок вказуючи тип посилки, місто відправлення та прибуття і далі чекає коли на оголошення відгукнеться водій попутного транспорту, здійснюючи час від часу повторний погук. Водії попутного транспорту за можливості можуть брати одразу декілька посилок для ефективного використання доступності вільного місця в транспорті. Так, один водій може покрити потреби одразу декількох відправників (рис.1.1).

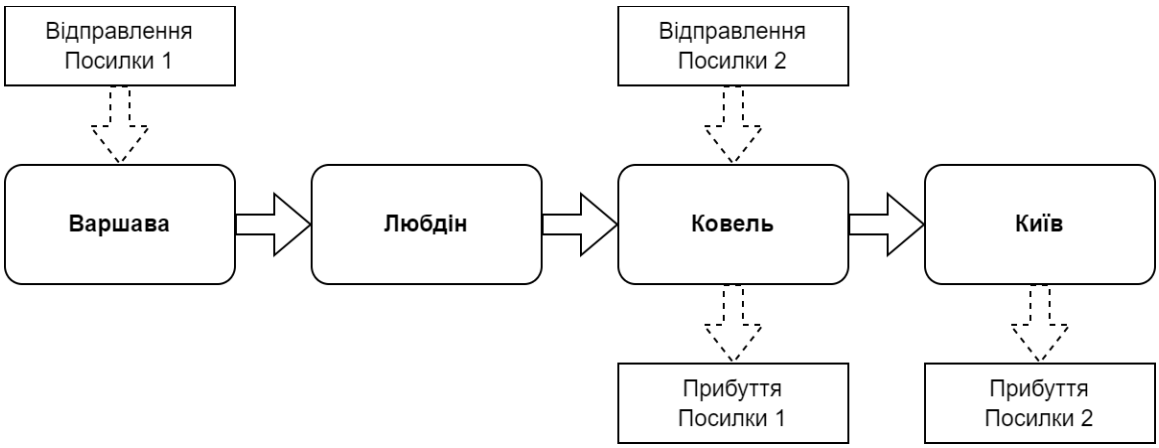


Рисунок 1.1 – Попутна доставка декількох посилок одночасно

На основі аналізу було визначено наступні ключові поняття та об’єкти процесу передачі посилок попутним транспортом які закладено в модель даних та використані в додатку.

Учасник процесу, користувач — може виступати як в ролі відправника так і водія попутного транспорту в залежності від обставин і потреб в певний момент часу.

– Маршрут — впорядкований по черзі слідування список міст через який їде попутний транспорт.

– Заявка на доставку — оголошення з інформацією про місто відправлення, місто прибуття, тип посилки, опис, та бажане вікно часу в яке треба здійснити відправку.

Процес поділено на ключові етапи (рис.1.2):

– Відправник здійснює пошук серед наявних варіантів доступного попутного транспорту.

– В разі відсутності доступного транспорту користувач створює заявку .

– Система формує список попутного транспорту згідно пошукового запиту або сформованої заявки.

– Користувач обирає серед варіантів та з’вязується з водієм попутного транспорту.

– Водій приймає та доставляє посилку

– Система дає змогу оцінити результат та виставити оцінку в внутрішній системі рейтингу відправнику та водію.

Рисунок 1.2 – Послідовність ключових етапів процесу пошуку та доставки посилок попутним транспортом

Таким чином визначено що розроблений додаток має забезпечити функціональні можливості для здійснення вказаних кроків.

Також на основі дослідження аналогів та наявних оголошень про доставку в мережі інтернет відправлення і типи транспорту були поділені на типи і визначені з’язки між ними за принципом транспорт і рекомендований для нього набір доступних типів посилок (рис.1.3):

– Конверт — Посилка, яка може бути поміщена в конверт, підходить для листів, документів або невеликих предметів.

– Рюкзак — Посилка середнього розміру, яка може бути поміщена в рюкзак, підходить для книг, одягу, аксесуарів.

– Валіза — Посилка великого розміру, яка може бути поміщена в валізу, підходить для електроніки, побутових товарів, іграшок.

– Багажник легкового автомобіля — Посилка дуже великого розміру, яка може бути поміщена в багажник легкового автомобіля, підходить для великої електроніки, спорядження, меблів.

– Багажник легкого вантажного автомобіля — Посилка великого обсягу, яка може бути поміщена в багажник легкого вантажного автомобіля, підходить для меблів, будівельних матеріалів, спорядження.

– Вантажний автомобіль — Посилка величезного розміру та обсягу, яка вимагає вантажного автомобіля для перевезення, підходить для перевезення великих меблів, промислового обладнання, будівельних матеріалів або великої кількості товарів.

Та транспорт відповідно:

– Пасажир — Окремі пасажири, які подорожують зі своїми речами, можуть взяти на себе відповідальність за перевезення посилок у своєму багажі (наприклад, в рюкзаках, валізах тощо).

– Особистий автомобіль — Транспортні засоби, призначені для перевезення власника та його пасажирів, зазвичай невеликого розміру, можуть мати багажник для перевезення невеликих посилок.

– Пасажирський транспорт — Транспортні засоби, які курсують за встановленими маршрутами та графіками, призначені для перевезення пасажирів та їхніх ручних поклаж.

– Легка вантажівка (легкий комерційний транспорт) — Транспортні засоби, призначені для перевезення вантажів середнього обсягу та ваги, зазвичай з більшим вантажним відсіком, ніж у персональних автомобілях.

– Вантажівка — Транспортні засоби, призначені для перевезення великих вантажів, з великим вантажним відсіком та високою вантажопідйомністю.

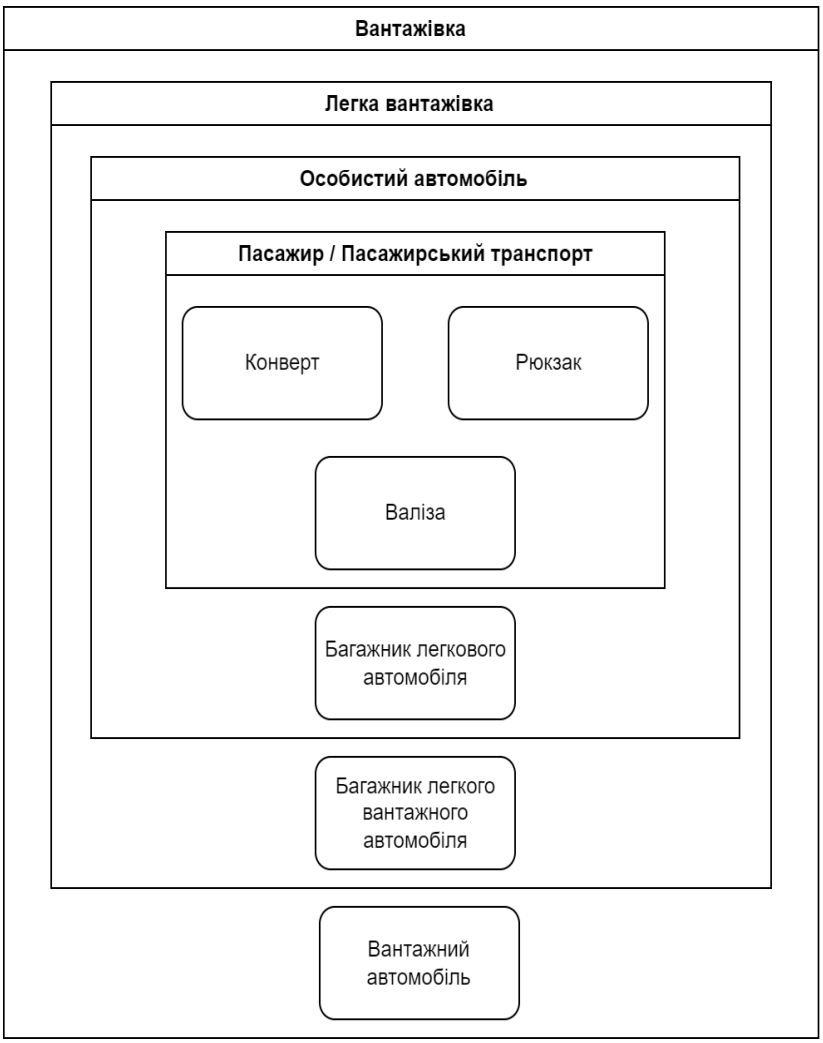


Рисунок 1.3 – Типи транспорту та типи посилок

Визначені типи транспорту та посилок біло включено в об’єктну модель додатку.

1.2 Аналіз інструментальних засобів пошуку попутного транспорту для передачі посилок

Існує кілька типів засобів пошуку попутного транспорту для передачі посилок, серед них такі як:

– Оголошення - можна розмістити оголошення на різних веб-сайтах та соціальних мережах, де шукаються попутні водії;

– Форуми та групи в соціальних мережах - можна приєднатися до груп, присвячених автоподорожам, і поставити запит на перевезення посилки;

– Сайти та мобільні додатки для пошуку транспорту такі як 112kilo[3] та sharry.cc[4];

– Безпосередній пошук попутного транспорту серед кола знайомих та в місцях стоянки та відправлення транспорту;

Ці тири засобів було розглянуто на прикладі поданих нижче платформ.

Прикладом платформи для розміщення оголошень є сайт Olx.ua[5].

Olx.ua - це онлайн-платформа, яка дозволяє публікувати різноманітні оголошення, такі як продаж товарів, продаж нерухомості, пропозицій оренди, оголошення про надання різноманітних послуг, в тому числі і оголошення про послуги перевезення пасажирів та транспортні перевезення.

Що б скористатися платформою для пошуку попутного транспорту треба відкрити головну сторінку і ввести ключові слова такі як “поїздка”, “транспорт”, “посилка”, “Передача”, тощо (рис.1.4), та вказати в запиті місце відправки та прибуття, додатково варто додати фільтр обравши пошук по місту або регіону відправлення.

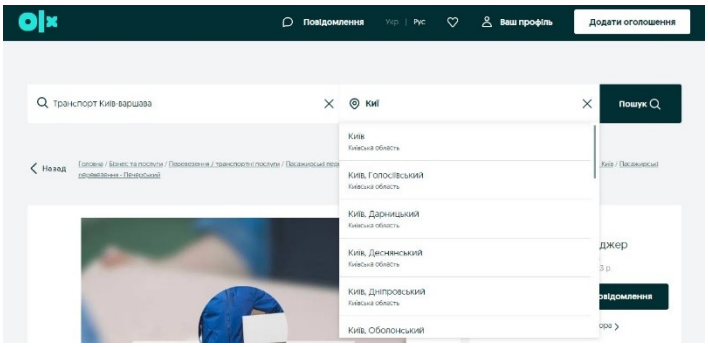


Рисунок 1.4 – Приклад форми пошуку Olx.ua

Приклад пошуку підходящого оголошення з потрібним маршрутом та можливість скористатися формою зв'язку з власником оголошення, домовитися про передачу посилки наведено на рис.1.5, 1.6.

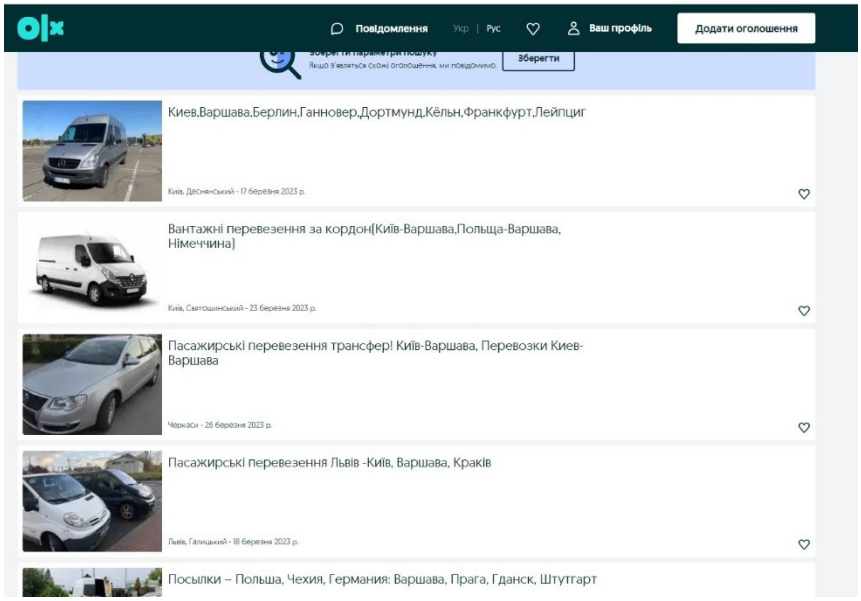


Рисунок 1.5 – Приклад результатів пошуку Olx.ua

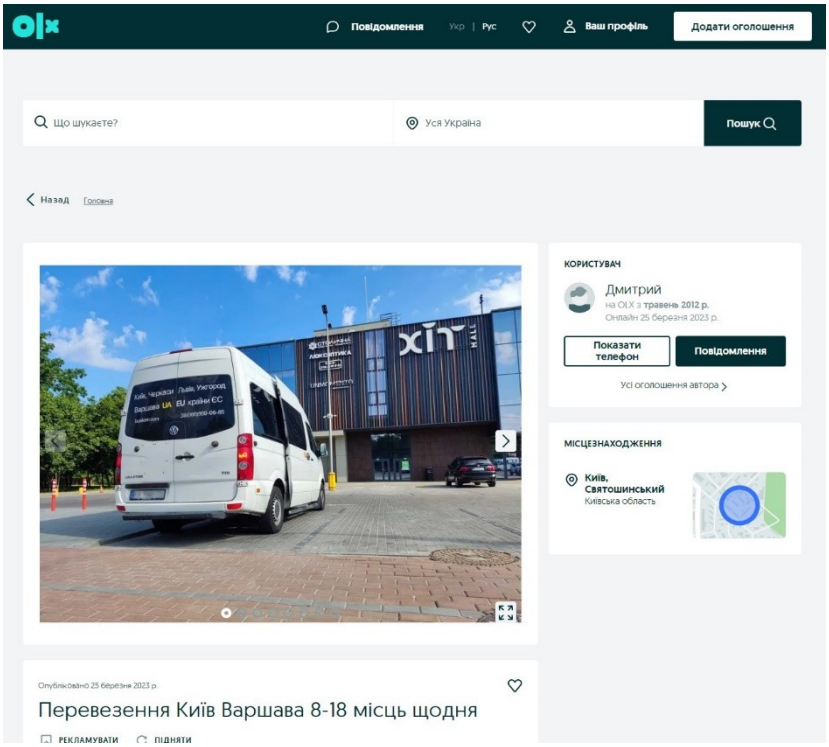


Рисунок 1.6 – Приклад сторінки оголошення Olx.ua

Використання вказаної платформи для пошуку попутного транспорту дозволяє вирішувати такі задачі та надає такі переваги як:

– Публікувати оголошення з інформацією про поїздки та їх маршрут;

– Публікувати оголошення з заявкою на перевезення посилки;

– Забезпечує канал зв'язку між замовником та виконавцем

– Надає можливість використовувати систему рейтингу яка дозволяє оцінити надійність виконавця;

До мінусів використання платформи можна віднести наступні пункти:

– Відсутність функції автоматичного підбору попутного транспорту, основна функція платформи - продаж та купівля товарів, то ж пошук попутного транспорту може бути складним та вимагати багато часу;

– В зв’язку з не спеціалізованістю сервісу та відсутністю відповідного розділу, доступна обмежена кількість варіантів попутного транспорту. Це може стати проблемою, якщо користувач шукає попутника для передачі посилки в рідкісному напрямку або в далекому місці;

– Пошук попутного транспорту може бути складним в зв'язку з відсутністю специфікації та стандартизації оголошень - оголошення публікуються в довільній формі і тому доводиться складати декілька пошукових запитів щоб знайти можливі варіанти;

– Неможливість перевірки достовірності інформації про попутника на olx.ua. Користувач не можете бути впевненими в тому, що інформація про водія, який відгукнувся на оголошення, є правдивою;

– Обмеженість сервісу за географією, оголошення публікуються користувачами україни і це робить його мало придатним для передачі посилок за кордон;

– Відсутність локалізації інформації про маршрут - інформація про маршрут зазвичай заповнюється користувачами однією мовою і в разі пошуку попутного транспортного засобу користувачу з іншої країни можуть бути не відомі географічні назви іншою мовою, або при пошуку однією мовою в вибірку не потраплять оголошення заповнені іншою;

– В разі якщо оголошення не знайдено то є необхідність регулярно повторювати пошук що б не пропустити оголошення;

Таким чином, платформа дає змогу вирішувати задачі пошуку попутного транспорту для передачі посилок. Але при цьому відсутня будь яка автоматизація

цього процесу. Мало підходить для пошуку транспорту для передачі посилок за кордон. Також оголошення мають обмежене охоплення аудиторії, яке пов’язане з складнощами пошуку та загальною спеціалізацією сервісу на оголошеннях продажу товарів а не надання транспортних послуг.

Розповсюдженим прикладом груп в соціальних мережах є велика кількість груп присвяченим подорожам, групи присвячені волонтерству, групи комьюніті мігрантів в різних країнах та містах та групи присвячені транспортним перевезенням розміщені в соціальній мережі, наприклад група мережі Facebook[6].

Facebook як соціальна мережа, яка може бути корисною для пошуку попутного транспорту для передачі посилок. Групи в Facebook[6] можуть бути особливо корисними для вирішення завдання пошуку попутного транспорту.

Що б виконати пошуку попутного транспорту для передачі посилок з використанням груп в соціальній мережі Facebook[6] треба виконати наступні дії:

– Відкрити соціальну мережу Facebook[6] та виконати пошук груп пов’язаних з подорожами (рис.1.7), пасажирськими та транспортними перевезеннями, наприклад “Попутка Україна”, “Маршрутка Київ” “Ukrainian Travelers”, та інші. Для передачі посилки за кордон можна шукати місцеві групи, наприклад “Українці у Варшаві”, “Наші люди Gorzow”, тощо. З результатів пошуку вибрати ті групи які найбільше відповідають потребам;

– Приєднатися до груп, якщо група закрита то надіслати запит на вступ в групу.

– Переглянути вміст груп та знайти оголошення з можливими попутними транспортними засобами (рис.1.8),(рис.1.9).

– Зв’язатися з автором оголошення та домовитися про передачу.

– Якщо не знайдено підходящої пропозиції, слід розмістити власне оголошення з описом посилки, зазначенням маршруту, розмірами, вагою та будь-якою іншою корисною інформацією, також варто не забути додати контактну інформацію.

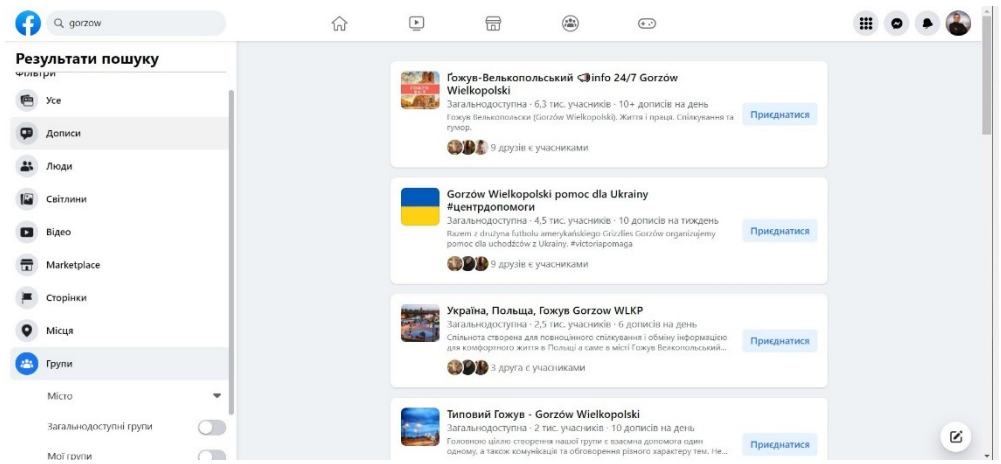


Рисунок 1.7 – Форма пошуку групи в Facebook

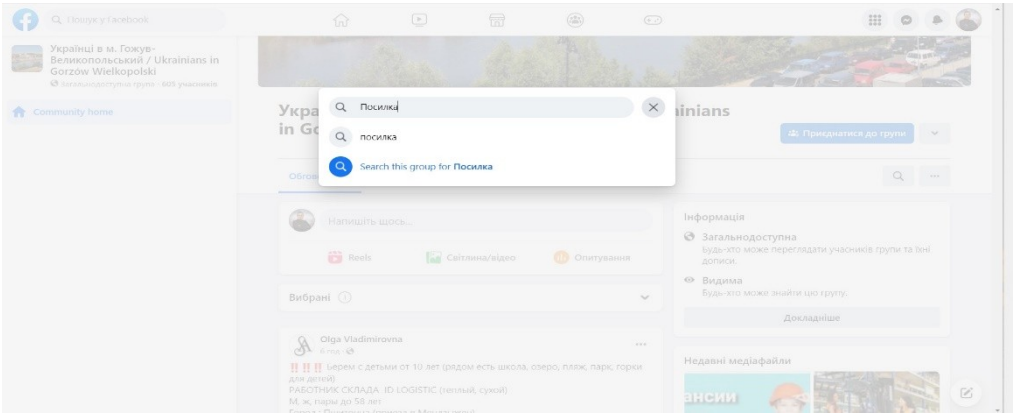


Рисунок 1.8 – Форма пошуку оголошень по групі в Facebook

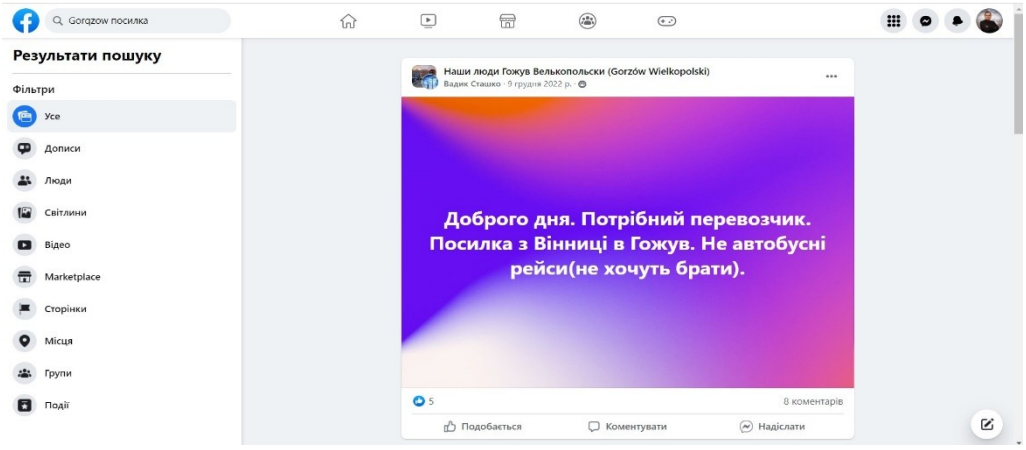


Рисунок 1.9 – Форма виводу результату пошуку в Facebook

Можливості та переваги які надає використання груп в соціальній мережі Facebook[6] для пошуку попутного транспорту для передачі посилкок:

– Можливість публікувати оголошення з інформацією про поїздки та їх маршрут;

– Можливість публікувати оголошення з потребою передачі посилки;

– Забезпечує потенційно велику аудиторію яка побачить оголошення в стрічці новин, Facebook[6] має більше 2,8 мільярдів активних користувачів по всьому світу;

– Канал зв’язку між замовником та виконавцем доставки;

– Наявність спеціалізованих груп що полегшує пошук попутного транспорту;

– Facebook дає змогу не обмежувати пошук однією країною, що є перевагою при пошуку попутного транспорту для передачі посилок за кордон;

До мінусів використання Facebook[6] можна віднести наступні пункти:

– Обмежені можливості фільтрації. Незважаючи на те, що Facebook надає можливість використовувати пошукові запити, щоб знайти потрібну групу або оголошення, обмежені можливості фільтрації можуть зробити процес пошуку менш ефективним;

– Користувачу може потрапити багато неактуальних оголошень або вони можуть бути в групах з низькою активністю;

– Відсутність стандартизації оголошень - оголошення публікуються в довільній формі і тому доводиться складати декілька пошукових запитів щоб знайти можливі варіанти;

– Глобальний пошук менш ефективний, тому користувач спочатку має знайти та вступити в відповідні групи, ситуацію ускладнюють закриті групи;

– Пошук вимагає від користувача виконати багато кроків від пошуку групи до пошуку оголошення в середині групи;

– Низька довіреність до користувачів. Не всі користувачі Facebook є надійними, і можливо, що деякі із них можуть бути шахраями. Відсутня система рейтингу. Користувач не можете бути впевненими в тому, що інформація про водія, який відгукнувся на оголошення, є правдивою;

– Відсутність локалізації інформації про маршрут - інформація про маршрут зазвичай заповнюється користувачами однією мовою і в разі пошуку попутного транспортного засобу користувачу з іншої країни можуть бути не відомі географічні назви іншою мовою, або при пошуку однією мовою в вибірку не потраплять оголошення заповнені іншою;

– Необхідність регулярно переглядати стрічку новин групи що б не пропустити нове оголошення якщо не було підходящих;

Отже як і в варіанті з сервісами оголошень форуми та групи в соціальних мережах дають базові можливості для пошуку попутного транспорту. Як і в варіанті з сервісами оголошень теж відсутні інструменти які б дозволили автоматизувати та прискорити процес пошуку. Інколи щоб знайти підходящу пропозицію треба регулярно переглядати групи на наявність нових оголошень. Більша аудиторія бачить оголошення завдяки його показу в стрічці новин. В соціальних мережах відсутній будь який рейтинг надійності користувачів, тому процес передачі посилки несе в собі більше ризиків. В той же час велика аудиторія соціальних мереж дозволяє знайти більше варіантів транспорту. Добре підходять для передачі посилок за кордон завдяки тому що користувачі не обмежені однією країною.

Sharry.cc[4] є платформою для пошуку та бронювання пасажирського транспорту між Європою та Україною. Також ця платформа дозволяє користувачам забронювати попутний транспорт для перевезення різних посилок між країнами. Крім того, сервіс надає інформацію про різні види транспорту та маршрути, які дозволяють вибрати найбільш зручний та ефективний варіант для перевезення посилки. З Sharry.cc[4] можна ефективно та безпечно передавати різні види вантажів між країнами.

Для того щоб виконати пошук попутного транспорту за допомогою платформи Sharry.cc[4] треба перейти на головну сторінку, над полем пошуку вибрати варіант запиту “Я відправник”, та обрати варіант “Знайти перевізника”, в поля пошуку ввести місто відправлення, місто прибуття, та бажану дату відправлення (рис.1.10).

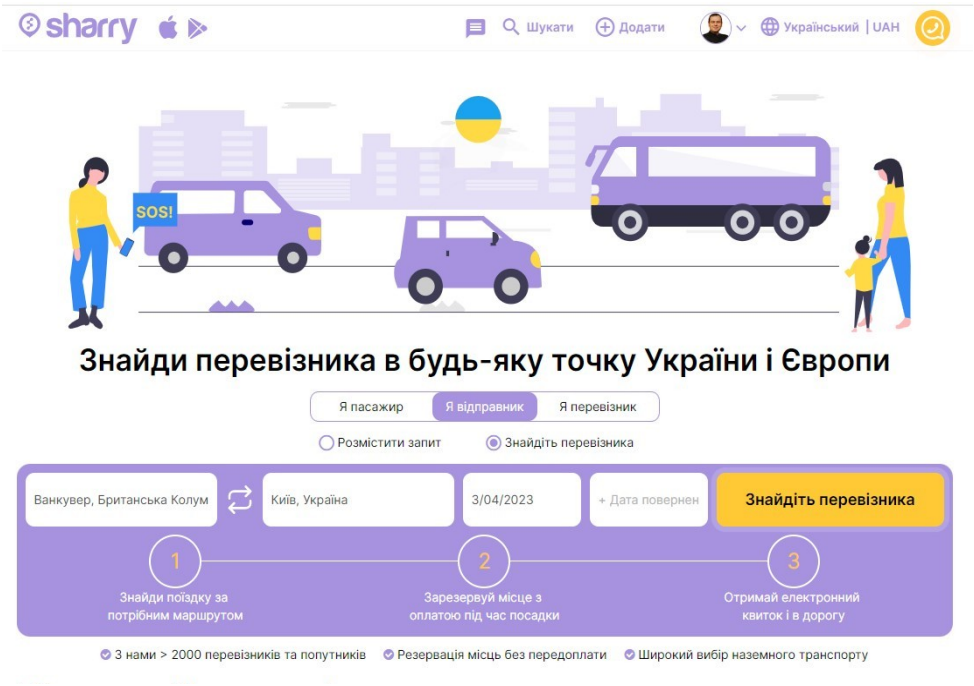


Рисунок 1.10 – Форма пошуку попутного транспорту Sharry.cc

Система за наявності автоматично підбере та запропонує підходящі варіанти транспорту (рис.1.11).

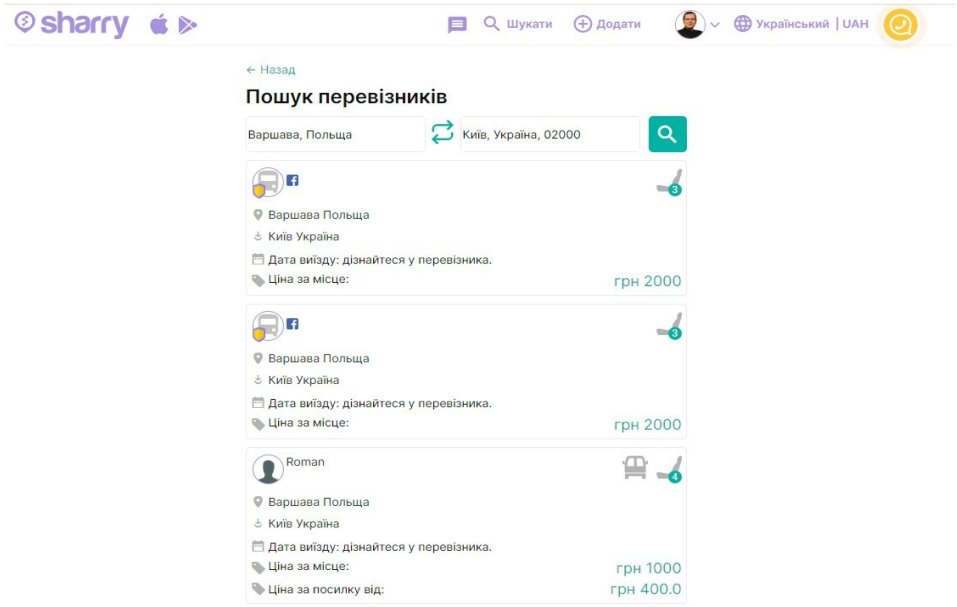


Рисунок 1.11 – Форма виводу результатів пошуку Sharry.cc

З запропонованих варіантів треба вибрати той який відповідає розміру посилки та натиснути кнопку “Відправити повідомлення” щоб зв'язатися з

перевізником. Якщо ж підходящих варіантів не знайдено то варто переключити перемикач з “Знайти перевізника” на “Залишити заявку”, і далі система пропонує обрати бажаний діапазон дат відправки, тип посилки опис і бажану ціну доставки (рис.1.12).

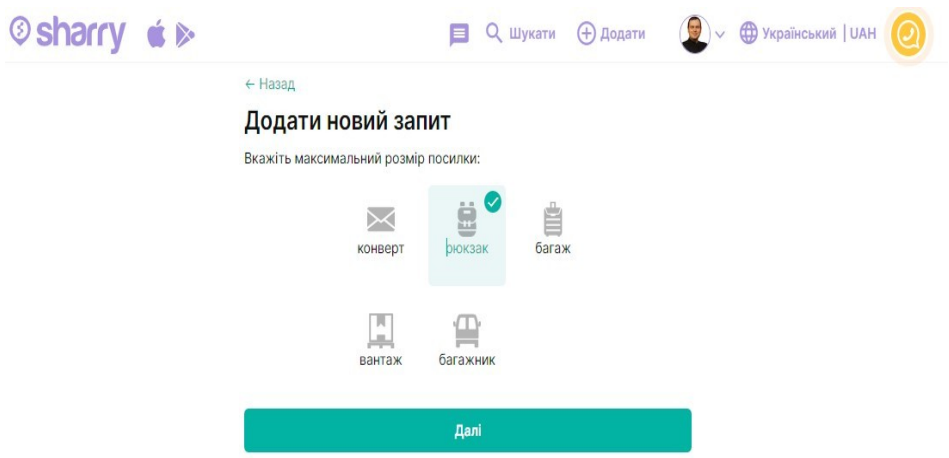


Рисунок 1.12 – Форма створення заявки Sharry.cc

Можливості та переваги які надає використання платформи Sharry.cc[4] для пошуку попутного транспорту для передачі посилкок:

– Автоматичний пошук та підбір оголошень про перевезення;

– Можливість публікувати заявку на перевезення з подальшим автоматичним підбором та рекомендацією перевізника;

– Можливість публікувати оголошення з інформацією про маршрут поїздки та подальшим автоматичним підбором відправників;

– Можливість додавати до маршруту проміжні пункти, що дозволяє маршруту потрапляти в пошук по більшій кількості запитів і оптимальніше підбирати маршрути та посилки;

– Надає систему рейтингу для оцінки рівня довіри користувачів;

– Підвищення безпеки шляхом верифікації перевізників;

– Підтримується локалізація на декілька мов (Українська, Англійська, Польська) що потенційно збільшує аудиторію а тому і кількість варіантів транспорту та посилок;

До мінусів можна віднести наступні пункти:

– Типи транспорту в основному обмежені пасажирськими, так як основний напрямок сервісу це пасажироперевезення;

– Маршрути в основному обмежені країнами європейського союзу та Україною;

Таким чином платформа Sharry.cc[4] надає розширені інструменти та можливості по пошуку попутного транспорту для передачі посилок. Також на платформі добре автоматизований процес підбору маршрутів та посилок. Основний напрям платформи це пасажироперевезення, тому для передачі посилок по основним напрямам постійно є транспорт який перевозить пасажирів і має місце для додаткового багажу, але це також обмежує типи транспорту, наприклад на ній важко знайти транспорт для великого вантажу, також неможливо знайти транспорт на інший континент (наприклад передати документи мандрівником який летить літаком).

112kilo[3] сервіс термінової доставки вантажних відправлень попутним транспортом. Являє собою електронну платформу, яка дозволяє відправнику вантажу знайти перевізника, що їде в попутному напрямку і готового виконати доставку. Сервіс спеціалізується на вантажних перевезеннях і дозволяє знайти перевізника під велику кількість типів специфічних вантажів, таких як тварини, рослини, крихкі вантажі, охолоджені та заморожені продукти. У користувачів сервісу є 2 ролі - “водій” та “Клієнт”. Сервіс надає страхову підтримку супроводжує посилку під час доставки, дозволяє створювати електронні накладні та підписувати їх електронними ключами. Дозволяє відслідковувати геолокацію вантажу.

Для пошуку попутного транспорту за допомогою сервісу 112kilo[3] треба: Зареєструватися на сайті та заповнити профіль. Створити замовлення у особистому кабінеті, вказати що, як, куди і коли треба перевезти. Отримати

список відповідних маршрутів та переглянути карточки водіїв. Обрати підходящого водія та зв’язатися з ним. Перед відправкою перевірити правильність заповнення накладної та зробити фото вантажу. При передачі вантажу підписати накладну електронним ключем. При отриманні посилки отримувач теж має підписати її електронним ключем в додатку на телефоні водія.

До можливостей і переваг сервісу можна віднести:

– Можливість публікувати заявки на перевезення посилок;

– Можливість публікувати інформацію про маршрути;

– Автоматизований пошук підходящих маршрутитів;

– Забезпечення каналом зв'язку між замовником і виконавцем;

– Наявність системи рейтингу надійності користувачів;

– Верифікація перевізників;

– Можливість відслідковувати геолокацію посилок;

– Локалізація на декілька мов (Українська, Англійська, Російська);

– Страхування вантажів (опціональна, потрібно поводитися на огляд вантажу);

– Дозволяє шукати транспорт придатний для передачі вантажів що потребують певних умов перевезення;

До мінусів можна віднести наступні пункти:

– Типи транспорту що можуть приймати участь в доставці обмежини вантажним;

– Обмеження по країнам. Доступний дуже обмежений перелік країн в яких можуть реєструватися користувачі;

– Не можна просто шукати транспорт по вказаному маршруту, спочатку треба обов’язково створити заявку;

Таким чином сервіс 112kilo[3] надає всі необхідні інструменти для пошуку попутного транспорту. Процес пошуку автоматизований, але вимагає додаткових дій у вигляді обов’язкового створення заявки. Сервіс забезпечує найбільший рівень безпеки посилки серед розглянутих засобів, та допомагає з генерацією

документів. Суттєвим мінусом є його географічні обмеження - в переліку країн майже немає країн європейського союзу. Також специфіка сервісу накладає значні обмеження на тип транспорту - доступний тільки вантажний транспорт. Сервіс більше підходить для передачі великих вантажів та вантажів які потребують особливих умов транспортування.

На рисунках 1.13 і 1.14 наведено приклади форми пошуку та фоми створення заявки.

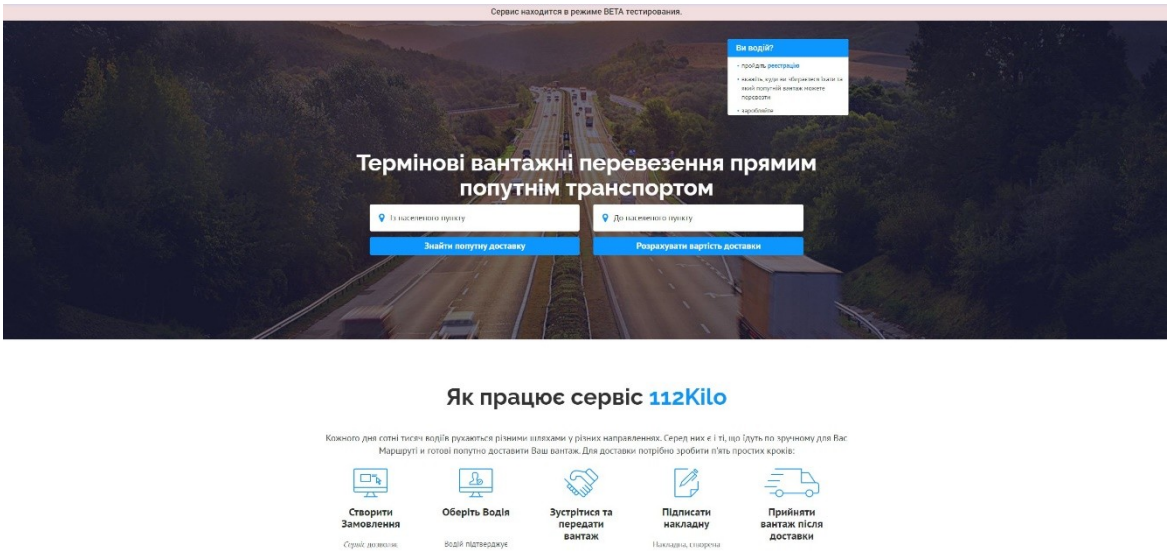


Рисунок 1.13 – Форма створення заявки 112kilo

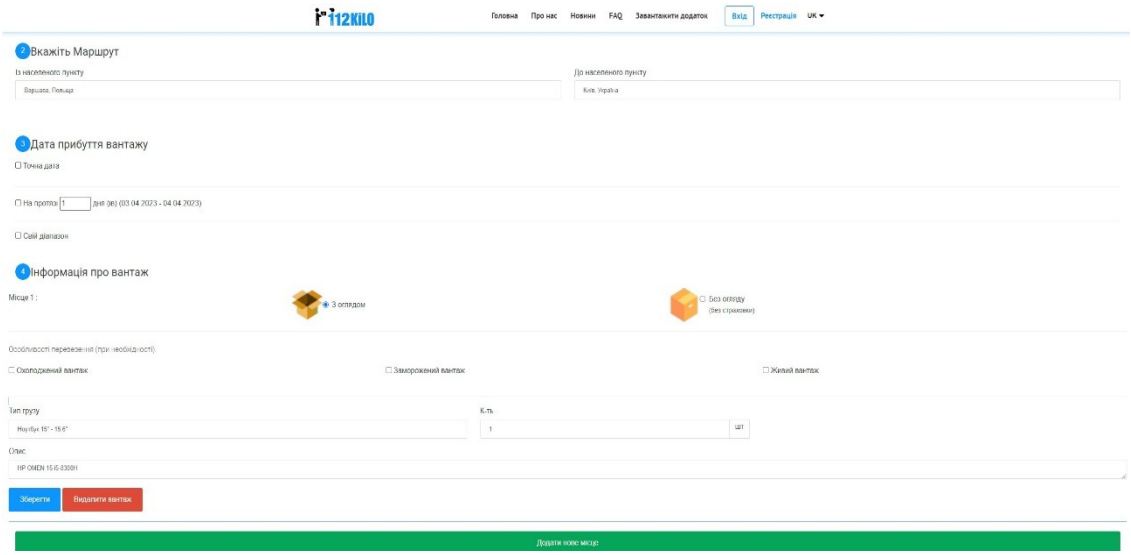


Рисунок 1.14 – Форма створення заявки 112kilo

Як видно з таблиці 1.1, кожен інструмент має як свої переваги так і недоліки.

Таблиця 1.1 – Узагальнені результати аналізу особливостей інструментів пошуку попутного транспорту.

Характеристика	Olx.ua	Facebook	Sharry	112kilo
Типи транспорту і посилок	Обмеження відсутні	Обмеження відсутні	Тільки маршрутний транспорт	Тільки вантажний комерційний транспорт
Спосіб пошуку попутного транспорту	Відсутні спеціальні фільтри	Відсутні спеціальні фільтри	Є спеціальні фільтри	Є спеціальні фільтри
Область пошуку попутного транспорту	Україна	Україна та за кордоном	За кордоном - декілька країн (Європа)	За кордоном - декілька країн
Типовий розмір посилок	Обмеження відсутні	Обмеження відсутні	Посилки середнього та малого розміру	Крупногабаритні посилки
Рівень безпеки пересилки	Низький	Низький	Прийнятний	Високий
Пошук по проміжним точкам	Відсутній	Відсутній	Присутній	Присутній

1.3 Вибір засобів програмної реалізації додатку

Сьогодні вимоги користувачів до сучасних додатків виросли як ніколи раніше. Користувачі очікують що додаток буде доступний з будь якого пристрою, з будь якої точки світу цілодобово[12]. Таку доступність додатку може забезпечити його розробка у вигляді веб додатку з клієнт-серверною моделлю у вигляді REST Api[8].

Клієнт-серверна модель передбачає взаємодію в якій клієнтській додаток посилає запит на сервер, сервер обробляє запит та присилає відповідь на клієнтський додаток, клієнтський додаток відображає отриману інформацію користувачеві. В такій схемі відповідальність розподілена між двома частинами: сервер відповідає за обробку та збереження інформації, клієнтський додаток відповідає за взаємодію з користувачем. Причому клієнтський додаток може мати будь яку реалізацію, як веб сторінка, Android додаток, IOs додаток, Windows додаток, IOT пристрій, чат бот в месенджері… один сервер можуть використовувати одночасно різні реалізації клієнтських додатків. Безпека та доступність даних забезпечується тим що фізично вони зберігаються на захищеному сервері, вони доступні з будь якої точки світу в будь який час за наявності зв’язку з сервером і наприклад втрата пристрою клієнта не тягне за собою втрати даних[12].

На рисунку 1.15 зображено приклад взаємодії компонентів і користувачів в клієнт серверній архітектурі додатку.

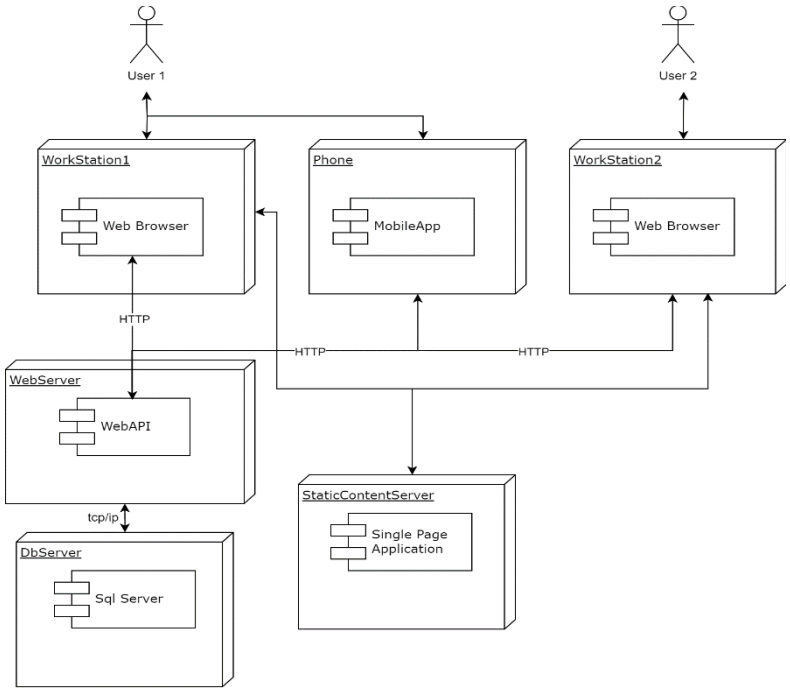


Рисунок 1.15 – Схема компонентів клієнт серверної архітектури

REST - абревіатура від Representational State Transfer, яка перекладається як «передача репрезентативного стану». Це архітектурний стиль в основі якого лежать обмеження[7]:

1. Клієнт-серверна архітектура системи;

2. Взаємодія без збереження сервером стану. Сервер не має зберігати інформацію про стан. Кожен запит має виконуватися незалежно, не впливаючи один на один і не покладаючись на стан іншого. Клієнт має передати всю необхідну інформацію для виконання запиту. При цьому сервер не зберігає інформацію про сесію користувача, замість звичайної автентифікації зазвичай використовують метод сервісу який передає ключ. Далі клієнт має повертати цей ключ в кожному запиті який цього потребує[8];

3. API має мати єдиний логічний інтерфейс, запити до ресурсів мають будуватися за єдиним стандартом. Відповіді теж мають бути стандартизованими. Інформація має подаватися в єдиній стандартизованій формі а не залежати від потреб додатку. Для CRUD операцій використовуються стандартні HTTP методи:

– GET: отримати ресурс, наприклад GET \user\1 - Отримати дані про користувача з номером 1;

– POST: створити ресурс, наприклад POST \user - створити нового користувача;

– PUT: замінити один ресурс на інший, наприклад PUT \user\1 - Змінити дані користувача 1 новими даними;

– PATCH: частково замінити дані в ресурсі, наприклад PATCH \user\1 - частково оновити дані користувача 1;

– DELETE: видалити ресурс, наприклад DELETE\user\1 - видалити запис користувача 1;

Однією з найбільш перспективни платформ для розробки Web Api додатку сьогодні є платформа .Net та основна мова програмування для цієї платформи — C#.

C# - це сучасна високорівнева загальна мова програмування, що бере свій початок як вдосконалення мов сімейства C. Об’єктно орієнтована та компонентно

орієнтована, і надає всі необхідні мовні конструкції для підтримки цих концепцій. Має особливості які допомагають писати швидко безпечні та надійні програми. Ось деякі з них:

Збирач сміття (Garbage collector), керує об’єктами та автоматично очищає пам’ять від невикористовуваних об’єктів. Це дозволяє звільнити розробника від необхідності за виділенням та звільненням пам’яті, що виключає більшість помилок що призводять до витоків пам’яті;

– Nullable типи, захищають від змінних які не посилаються на об’єкти в пам’яті;

– Лямбда методи, надають засоби для функціонального програмування;

– Language Integrated Query (LINQ) - синтаксис що надає єдиний шаблон для роботи з даними з будь якого джерела;

– Асинхронні та паралельні операції - надає широкий набір безпечних методів та мовних конструкцій для роботи з паралельністю та асинхронністю;

– Уніфікована система типів - всі типи наслідуються від типу Object і мають набір спільних методів і властивостей що забезпечує підвищену безпеку типів. Хоча й C# насправді не має власних типів, всі типи є частиною самої платформи .Net що дозволяє використовувати в одному додатку бібліотеки написані різними мовами що підтримує ця платформа;

– C# підтримує користувацькі типи посилань та типи значень;

– Дозволяє безпечне перевантаження методів за допомогою ключових слів abstract, virtual, override та sealed;

.Net це сучасна уніфікована, безкоштовна, кросплатформна платформа з відкритим вихідним кодом створена компанією Microsoft та підтримується незалежною організацією .NET Foundation. Призначена для розробки багатьох типів програм і містить широкий стек технологій[10]:

– Хмарні технології - Azure;

– Розробка веб додатків - Asp.Net та Blazor;

– Розробка програм під операційну систему Windows - .Net Maui, WPF та WindowsForms;

– Розробка мобільних додатків - .Net Maui, Xamarin;

– Розробка ігор - ігровий рушій Unity;

– IOT - підтримка архітектур ARM32 та ARM64;

– ШІ та машинне навчання - ML.Net та .Net for Apache spark.

Платформа .Net має реалізацію декількох основних мов програмування (Visual Basic, C#, F#), та дозволяє додавати реалізацію підтримки інших мов.

Підтримка кросплатформності та різних мов забезпечує стандарт CLI (common language infrastructure), який гарантує для всіх підтримуваних мов такі основні можливості як: управління пам'яттю, завантаження збірок, безпека, обробка виключень, синхронізація. При цьому він дозволяє використовувати переваги різних мов для виконання різних задач в одному додатку. Всі підтримувані мови компілюються відповідними компіляторами в проміжний код IL(проміжний код), який потім може виконуватися середовищем CLR(Common Language Runtime - реалізація стандарту CLI від Microsoft) незалежно від середовища виконання та від мови якою він був написаний (рис.1.16)[11].

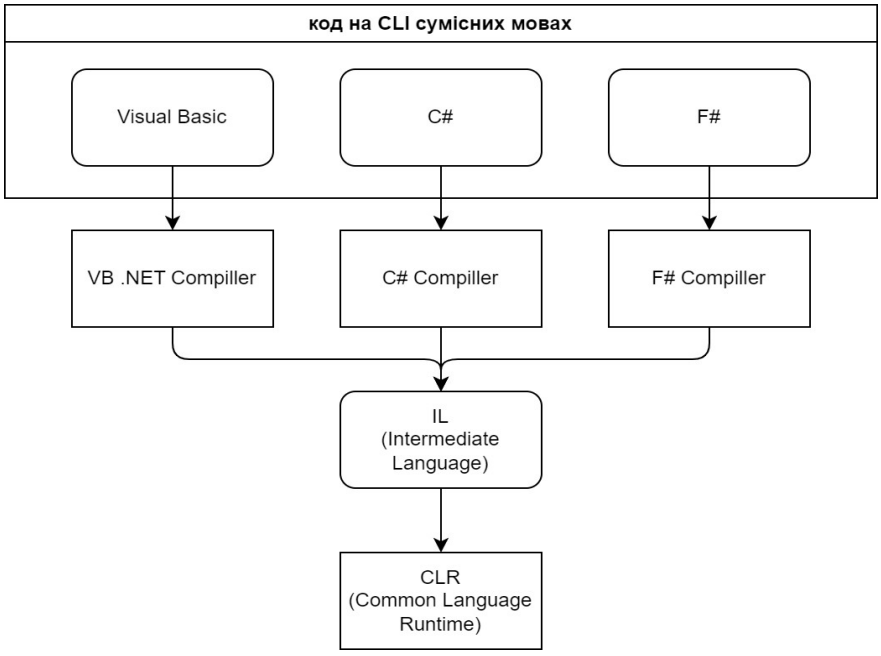


Рисунок 1.16 – Схема компіляції та виконання різних CLI сумісних мов.

Отже платформа .Net добре підходить для написання надійних, продуктивних додатків які не будуть залежати від платформи і використ. При цьому використовуючи комбінацію будь яких підтримуваних CLI сумісних мов в різних модулях додатку. Платформа має всі необхідні інструменти для збірки, компіляції, розгортання додатку, NuGet менеджер пакетів, IDE та інші...

Саме для написання веб додатків платформа .Net пропонує фреймворк ASP.Net. Цей фреймворк містить всі необхідні бібліотеки та інструменти для створення веб додатків. Які роблять процес розробки швидшим і легшим.

– Має вбудовані шаблони та бібліотеки для створення REST Api що добре підходить для написання нашого додатку що розробляється;

– Забезпечує можливість розгортання на різних серверах та в різних операційних системах, в тому числі і в хмарних сервісах Azure;

– Має вбудовані механізми безпеки, захист від атак таких як впровадження SQL-ін'єкцій та XSS, механізми авторизації та автентифікації;

– Механізми для роботи з БД та іншими джерелами даних такі як ADO.Net, Entity Framework - який дозволяє працювати з записами в БД як об’єктами додатку і дозволяє застосовувати підхід розробки БД Code first - спочатку ми описуємо сутності як звичайні класи в коді а потім на основі них будується структура БД;

– Механізми конфігурації серверу;

– Механізми для роботи з файловим контентом;

– Засоби для обробки HTTP запитів, засоби для роботи з Сокетами та інші…;

– Вбудований механізм Інверсії залежностей Dependency Injection.

– Засоби для створення динамічних сторінок;

– Засоби створення мікро сервісів;

Та багато інших бібліотек та засобів для створення надійного та якісного веб додатку.

Тож як бачимо мова C# та платформа .Net можуть забезпечити нас всіма необхідними засобами для створення серверу web додатку.

Також в для веб додатку описаного в цій роботі розроблено клієнт у вигляді SPA – Single Page Application (односторінковий веб додаток). SPA це підхід до розробки веб-додатку в якому контент сторінки завантажується один раз а далі взаємодія з користувачем відбувається без перезавантаження і переходу по сторінках. Це забезпечує кращий користувацький досвід наближений до нативних додатків, більшу швидкість і реактивність, допомагає відокремити фронтенд від бекенду.

Для розробки клієнтського додатку обрано феріймворк Angular та мову TypeScrypt. Це надає такі переваги як[14]:

– Строга типізація та об’єктно орієнтовний підхід, Спрощує розуміння та дебагінг коду, строга типізація забезпечує виявлення помилок статичним аналізатором та на етапі компіляції а не під час виконання[13].

– Компонентний підхід Angular сприяє легкому повторному використанню коду та його чистоті за рахунок його розбиття на логічні та функціональні частини.

– Має такі вбудовані сервіси як HTTPClientModule що забезпечує легку та надійну взаємодію з WebApi

– Має вбудовану підтримку тестування.

– Angular CLI — допомагає пришвидшити розробку за рахунок можливості генерації шаблонів компонентів та класів, дає інструмент автоматизації завдань та інтеграцію з середовищем розробки.

Отже враховуючи ці переваги можна сказати що вибір мови TypeScript та фреймворку Angular є доцільним вибором для розробки клієнтського додатку для WebApi сервісу.

1.4 Постановка технічного завдання

Провівши аналіз особливостей процесу пошуку попутного транспорту для передачі посилок та дослідивши існуючі програмні анологи було сформовано наступні функціональні вимоги:

– Пошук попутного транспорту по містам відправлення та прибуття.

– Створення заявки на доставку.

– Створення маршруту доставки.

– Обмін повідомленнями між замовником та перевізником.

– Пошук підходящого для заявки попутного транспорту.

– Пошук підходящих заявок на доставку по маршруту.

А також сформовані такі нефункціональні вимоги:

– Доступність клієнтського додатку на різних платформах (Windows, iOS, Android, linux)

– Відсутність обмежень за країною при підключенні.

– Дані авторизації користувачів не повинні зберігатися в відкритому вигляді.

– Дані не повинні зберігатися на пристроях користувачів.

– Можливість підключатися до API з інших, в тому числі сторонніх клієнтських додатків.

Висновки за розділом 1.

1. Проведено аналіз процесу пошуку попутного транспорту для передачі посилок. Розглянуті особливості процесу. Визначено основні етапи процесу, поняття та об’єкти процесу. На основі аналізу сформовані основні вимоги до об’єктної моделі додатку.

2. Зроблено огляд наявних засобів та способів вирішення проблеми пошуку попутного транспорту. Було визначено ключові особливості, переваги та недоліки розглянутих інструментів пошуку попутного транспорту. На основі

результатів аналізу сформовано ключові вимоги до додатку що представлено в даній дипломній роботі.

3. Проведено огляд сучасних засобів розробки Web додатку. Визначено переваги використання платформи .Net та мови C# для розробки WebApi частини додатку та фреймворку Angular з використанням TypeScript для клієнської частини.

4. На основі проведеного аналізу формульовано функціональні та нефункціональні вимоги до веб сервісу пошуку попутного транспорту для передачі посилок.

2 ПРОЕКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Використання сучасних методологій проектування акцентує увагу на створенні різного рівня абстракцій моделей предметної області [15] та системи. Кожна модель описує певний компонент системи, використовуючи різні типи діаграм та документів що б представити певну перспективу яка відповідає певним потребам різних учасниками розробки, властиву їх ролям та обов’язкам. Створення моделей є невід'ємною частиною в процесі розробки чи оновлення будь якої системи. Вони дозволяють більш чітко бачити завдання на різних рівнях деталізації. У цьому розділі здійснено проектування додатку з використанням UML діаграм, описаний створений додаток та його архітектура.

2.1 Визначення архітектури додатку

Є декілька основних популярних архітектурних підходів в розробці веб додатку на платформі .Net. Одразу було відхилена мікросервісна архітектура як збиткова при даній складності і розміру додатку. Вона б потягнула за собою не обґрунтоване збільшення складності і підтрики. Отже було розглянуто наступні 3 архітектури:

– Монолітна архітектура — це архітектурний підхід при якому всі компоненти знаходяться в одному проекті. Добре підходить для швидкого створення прототипів і невеликих додатків без складної бізнес логіки, які містять здебільшого тільки CRUD операції. Має великі ризики при розростанні проекту та при роботі великих команд, так як не забезпечує достатній рівень абстракції та незалежності компонентів.

– N-Layer архітектура (рис.2.1) — це архітектура в якій компоненти зазвичай діляться на 3-4 шари. Розділяючи рівень доступу до даних, рівень бізнес логіки і рівень презентації, яким може бути як в нашому WebApi або інтерфейс користувача на основі View контролерів. При цьому компонент вищого рівня може звертатися до компонента на рівень нижче. Така архітектура порівняно з

монолітом хоч і потребує трохи більше зусиль в проектуванні та розробці — забезпечує певний рівень абстракції та нижчий рівень зв’язаності компонентів. Що допомагає незалежно тестувати модулі цих рівнів, наприклад бізнес логіку не залежно від даних та представлення незалежно від логіки. Все ж має один недолік при проектуванні, основа додатку це рівень доступу до даних і вся архітектура будується на його основі, цей підхід вважається застарілим.

– Clean або Onion архітектура (рис.2.2) — заміна N-Layer архітектури. На відміну від попередньої ділить додаток на Домен (або ядро) в якому описуються об’єктна модель доменної області, події доменної області та основні бізнес правил і має бути незалежним від технологій, бібліотек та фреймворків, Application – рівень логіки який описує основну бізнес логіку додатку з мінімальною прив’язкою до технологій, без прив’язки до даних та сервісів, рівень інфраструктури та шар представлення, які містять різні сервіси для забезпечення функціонування додатку і надають інтерфейс взаємодії з ним. При цьому представлення і інфраструктура не взаємодіють між собою. Такий підхід є більш сучасним і дозволяє при проектуванні додатку відштовхуватися не від технологій а від предметної області, приймаючи рішення по реалізації інтерфейсів та використання технологій на пізні етапи, коли необхідність в них стає більш очевидною [9].

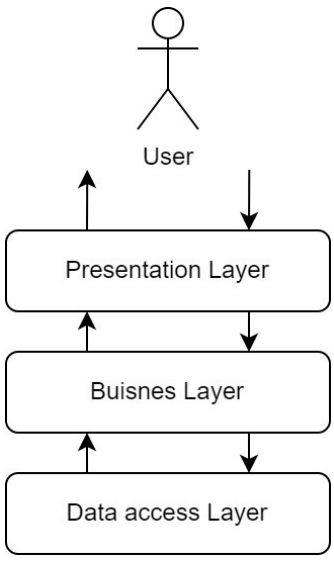


Рисунок 2.1 – Принцип взаємодії компонентів N-Layer архітектури

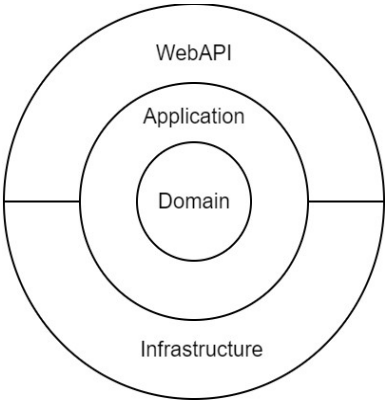


Рисунок 2.2 – Схема принципу побудови компонентів в Clean архітектури

Оскільки при розробці архітектури додатку було вирішено відштовхуватися від визначеної моделі предметної області з врахуванням переваг що він надає та закладених принципів — очевидним вибором є використання Clean архітектури.

2.2 Моделювання додатку засобами UML

2.2.1 Деталізація вимог сервісу з використанням Use Case діаграми.

Use Case (випадки використання) діаграми є ключовим компонентом моделювання вимог і допомагають формалізувати поведінку системи з точки зору її користувачів. Вони допомагають зосередитися на взаємодії між користувачами та системою, виявляючи основні функціональні вимоги до програмного продукту.

Кожний Use Case представляє окрему дію або послідовність дій, які система може виконувати для досягнення конкретної цілі користувача. Вони дозволяють команді розробників і стейкхолдерів бачити, як система повинна працювати у відповідь на певні вхідні дії користувача.

Use Case діаграми сприяють виявленню потенційних проблем у вимогах до системи на ранніх стадіях розробки, дозволяючи їх виправити ще до початку розробки коду.

На основі аналізу предметної області та існуючих інструментів вже були сформовані основні функціональні вимоги до сервісу:

– Пошук попутного транспорту по містам відправлення та прибуття.

– Створення заявки на доставку.

– Створення маршруту доставки.

– Обмін повідомленнями між замовником та перевізником.

– Пошук підходящого для заявки попутного транспорту.

– Пошук підходящих заявок на доставку по маршруту.

На основі аналізу та отриманих вимог побудовано діаграму основних варіантів використання додатку, зображену на рисунку 2.3.

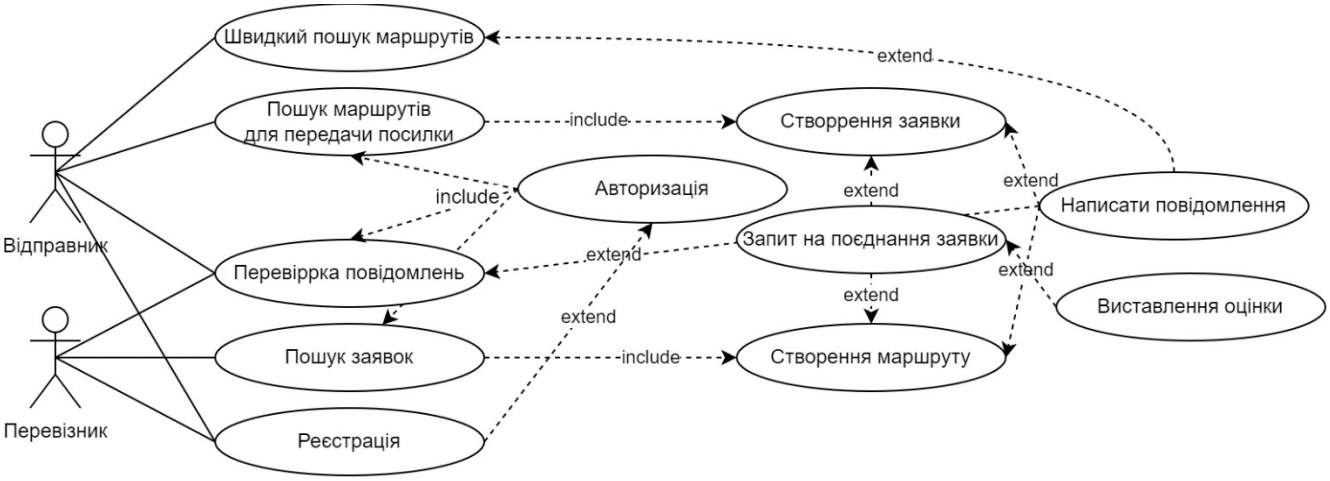


Рисунок 2.3 – Схема основних варіантів використання додатку

На основі діаграми використання було деталізовано та доповнено варіанти використання:

Варіант використання: Швидкий пошук маршруту.

Актор: Відправник, зареєстрований користувач.

Передумова: Користувач не авторизований в системі.

1. Користувач переходить на форму швидкого пошуку.

2. Користувач вводить місто відправлення та місто прибуття посилки.

3. Система надає список маршрутів що збігаються з параметрами запиту.

Extend:

1. Користувач обирає варіант та пише повідомлення водію попутного транспорту.

Include: Авторизація в системі.

Варіант використання: Пошук маршруту для передачі посилки

Актор: Відправник, зареєстрований користувач

Передумова: Користувач не авторизований в системі.

1. Користувач переходить на форму швидкого пошуку.

2. Користувач вводить місто відправлення та місто прибуття посилки.

3. Система не знаходить маршрутів за вказаним запитом.

Include: Авторизація в системі.

Extend:

1. Користувач створює заявку на доставку.

2. Користувач через час проводить пошук нових маршрутів підходящих під заявку.

3. Користувач зв’язується з водієм попутного транспорту шляхом написання повідомлення

Extend:

1. Користувач натискає на обраному маршруту кнопку запиту на поєднання заявки та маршруту.

Варіант використання: Пошук маршруту для передачі посилки

Актор: Відправник, зареєстрований користувач

Передумова: Користувач не авторизований в системі.

1. Користувач переходить на форму швидкого пошуку.

2. Користувач вводить місто відправлення та місто прибуття посилки.

3. Система не знаходить маршрутів за вказаним запитом.

Include: Авторизація в системі.

Extend:

1. Користувач створює заявку на доставку.

2. Користувач отримує повідомлення від водія попутного транспорту

3. Користувач відповідає на повідомлення

Варіант використання: Перевірка повідомлень.

Актор: Зареєстрований користувач.

Передумова: Користувач не авторизований в системі.

1. Користувач переходить до списку повідомлень.

2. Система надає список повідомлень та помічає

3. Обирає та читає непрочитане повідомлення.

Include: Авторизація в системі.

Extend:

1. Користувач відповідає на повідомлення.

Варіант використання: Пошук заявок що зібгаються з маршрутом.

Актор: Водій попутного транспорту, Зареєстрований користувач.

Передумова: Користувач не авторизований в системі.

1. Користувач переходить до списку власних маршрутів.

2. Система надає список маршрутів.

3. Користувач обирає маршрут.

4. Користувач натискає кнопку пошуку заявок для маршруту.

5. Система відображає список заявок що збігаються з маршрутом.

Include:

1. Авторизація в системі.

2. Створення маршруту поїздки.

Extend:

1. Користувач пише повідомлення автору заявки.

2. Користувач натискає кнопку поєднання заявки та маршруту.

Варіант використання: Реєстрація користувача.

Актор: Не зареєстрований користувач.

Передумова: Користувач не авторизований в системі.

1. Користувач відкриває додаток.

2. Система відображає вікно входу.

3. Користувач натискає кнопку реєстрації.

4. Користувач вводить реєстраційні дані.

5. В системі реєструється новий користувач

Extend:

1. Користувач авторизується в системі.

2.2.2 Опис доменної моделі даних додатку

Скільки в основі архітектури сервісу лежить доменна модель, було створено абстрактний верхнерівневий опис доменної моделі даних та описано за допомогою UML діаграми класів (рис.2.4). Зображена діаграма є абстрактною і показує мінімальні необхідні об’єкти які задіяні в процесі пошуку попутного транспорту для передачаі посилки.

Діаграми класів UML дозволяють представити структуру системи, показуючи класи, їх атрибути та методи, а також відносини між класами. Вони використовуються для визначення статичної структури системи та відображення доменної моделі даних [16].

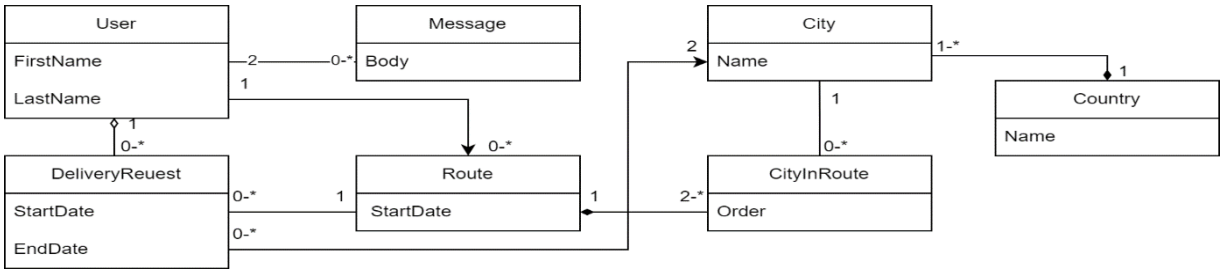


Рисунок 2.4 – Верхньорівневий опис об’єктної моделі процесу пошуку попутного транспорту для передачі посилок

Отже як видно з діаграми основними об’єктами в системі є:

- User – профіль користувача системи, як відправника так і водія попутного транспорту, зберігає данні про особу користувача.
- DeliveryRequest – заявка на доставку, зберігає інформацію про власника заявки, бажаний інтервал часу в який треба доставити вантаж, місто відправлення та місто прибуття помилки.
- Route – маршрут попутного транспорту, містить інформацію про власника маршруту, час відправки та колекцію міст через які буде проходити маршрут попутного транспорту.

- CityInRoute — представляє місто на шляху маршруту. Містить інформацію про маршрут, місто та порядок (позиція) цього місця на маршруті, так як при пошуку важливо враховувати що б місто відправлення ішло до міста прибуття.
- Message – слугує для представлення повідомленнями якими обмінюються користувачі, містить інформацію про відправника, отримувача та тіло повідомлення.
- City – місто або населений пункт, має інформацію про країну до якої належить.
- Country – країна.

Таким чином було сформовано базове уявлення про об’єктну модель додатку що розроблено в рамках цієї дипломної роботи.

2.2.3 Вибір джерела даних для списку та географічних об’єктів планети

Оскільки серед вимог до додатку є відсутність територіальних та мовних обмежень постало питання про збір даних всіх країн та міст планети, їх альтернативних назв різними мовами та наповнення ними бази даних додатку. Також потрібно було адаптувати БД для збереження потрібної інформації що б забезпечити зворотній зв’язок для періодичного оновлення даних.

Розглянутим варіантом в якості джерела географічних назв був сервіс геокодінгу GeiNames[17] який надає у відкритому доступі дані про більш ніж 12 мільйонів географічних одиниць серед яких і міста та країни. Данні надаються у вигляді окремих текстових файлів і містять різну інформацію таку як назви різними мовами, альтернативні назви, коди країн, адміністративні одиниці, координати, населеність і та інше...

В додатку було вирішено використати такі дані:

- Назви географічних одиниць — їх за спеціальним текстовим кодом було розбито на типи, такі як міста, країни, адміністративні одиниці та інше.
- Коди країн — необхідні для зв’язування міст з країною

- Коди адміністративних одиниць, так як під час аналізу даних стало очевидним що в одній країні може бути декілька населених пунктів з однаковою назвою, тому треба додаткова інформація.
- Населеність — для сортування результатів пошуку вирішено використовувати показник населеності, було зроблено припущення що чим більше населення в місті тим вірогідніше що шукають саме його.
- Альтернативні назви різними мовами для пошуку та локалізації маршрутів різними мовами.
- Ідентифікаційний номер запису GeoNameId, для збереження сумісності з джерелом даних і подальшої можливості періодичного оновлення.
- Широта та довгота — закладено для подальшого розвитку проекту і можливості в окремих клієнтських додатках відображати точки на карті.

Для забезпечення роботи системи з вказаними даними схема об’єктної моделі була розширена наступним чином як вказано на схемі 2.5.

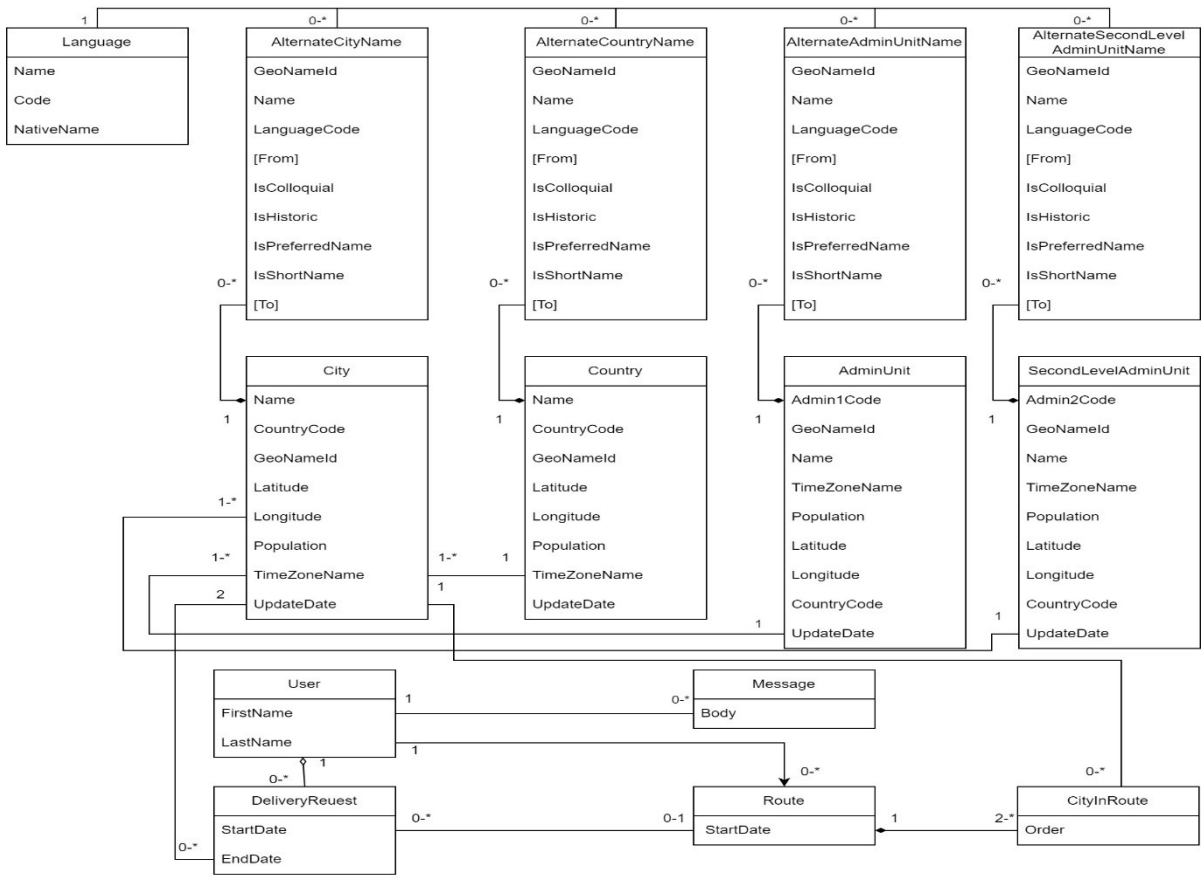


Рисунок 2.5 – Розширений прототип об’єктної моделі у вигляді діаграми класів

2.2.4 Діаграма компонентів додатку

Діаграма компоненту відображає високорівневу структуру системи, розбитої на взаємодіючі компоненти, та схему розгортання додатку. В контексті додатку що розроблено в рамках цієї роботи, система розбита на наступні компоненти (рис.2.6):

– Сервер бази даних.

– Клієнський додаток у вигляді SPA.

– Сервер веб сервісу, який в свою чергу розбитий на такі компоненти як webApi, Domain, Application та Infrastructure.

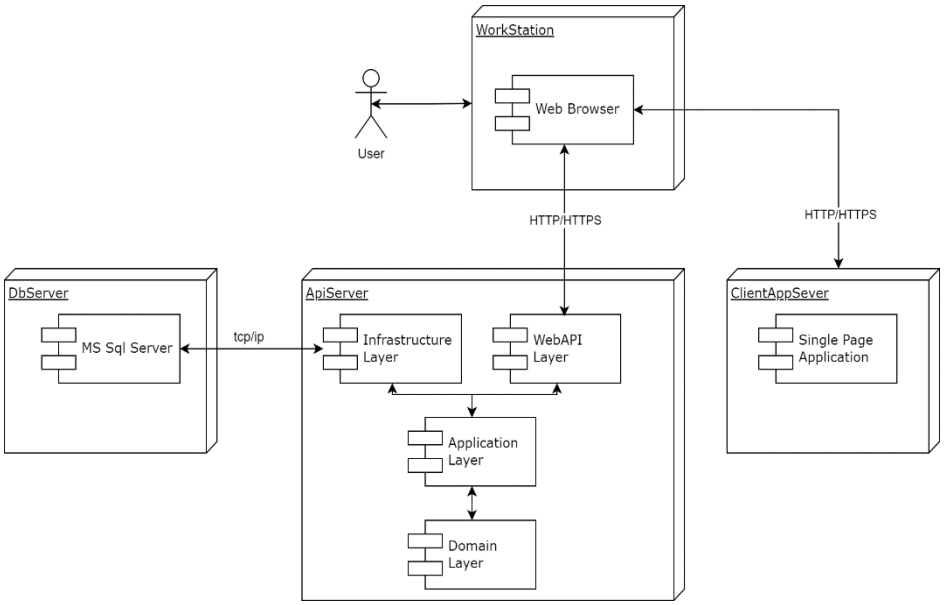


Рисунок 2.6 – Схема компонентів додатку

Як видно зі схеми серверна частина складається з чотирьох компонентів:

– Рівень домену — описує основні сутності події та бізнес правила доменної області. В ньому немає посилань на інші пакети та він є незалежним від конкретних бібліотек чи технологій.

– Application рівень — містить бізнес логіку без прив’язки до кінцевих технологій та інтерфейси необхідних йому сервісів, таких як джерело даних система авторизації чи інше. Тільки логіка предметної області.

- Рівень інфраструктури реалізовує інтерфейси різних сервісів, відповідає за доступ до даних, авторизацію та автентифікацію користувачів та інше. Тобто забезпечує все необхідне для функціонування Application рівня.
- Рівень презентації — в нашому випадку це WebApi, його зона відповідальності це забезпечення інтерфейсу взаємодії з зовнішніми клієнтами через протокол HTTP.

Для зменшення залежностей між компонентами веб додатку вирішено використовувати архітектурний шаблон Mediator (посередник), що сприяє створенню більш незалежних модулів з низькою зв’язаністю, підвищенню гнучкості, організованості та тестованості додатку. Цей шаблон реалізує замість прямої взаємодії модулів, взаємодію через посередника через команди та запити. Модуль публікує команду а медіатор вирішує хто і як її виконає.

2.2.5 Діаграма схеми бази даних.

Діаграма схеми бази даних показує структуру зв’язки між таблицями бази даних. В випадку розробленого додатку вона також ілюструє модель даних додатку. Так як при розробці використовувався підхід Code First при якому база даних генерується на основі класів моделі описаного кодом C#. таку генерацію таблиць забезпечує фреймворк EntityFramework.

Оскільки схема даних виявилася достатньо комплексною для кращого розуміння схеми поділені на логічні зони відповідальності (рис.2.7 -2.9).

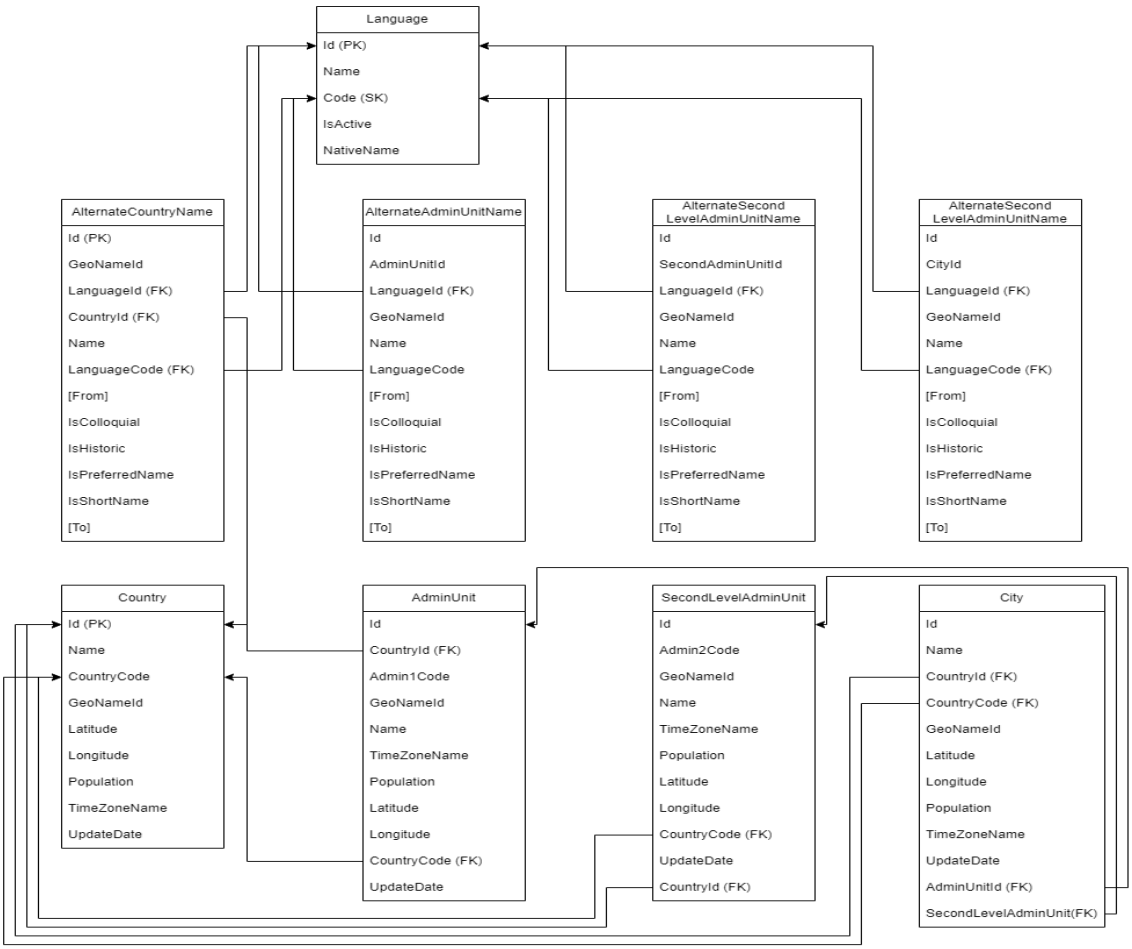


Рисунок 2.7 – Схема моделі даних географічних об’єктів та пошуку по ним

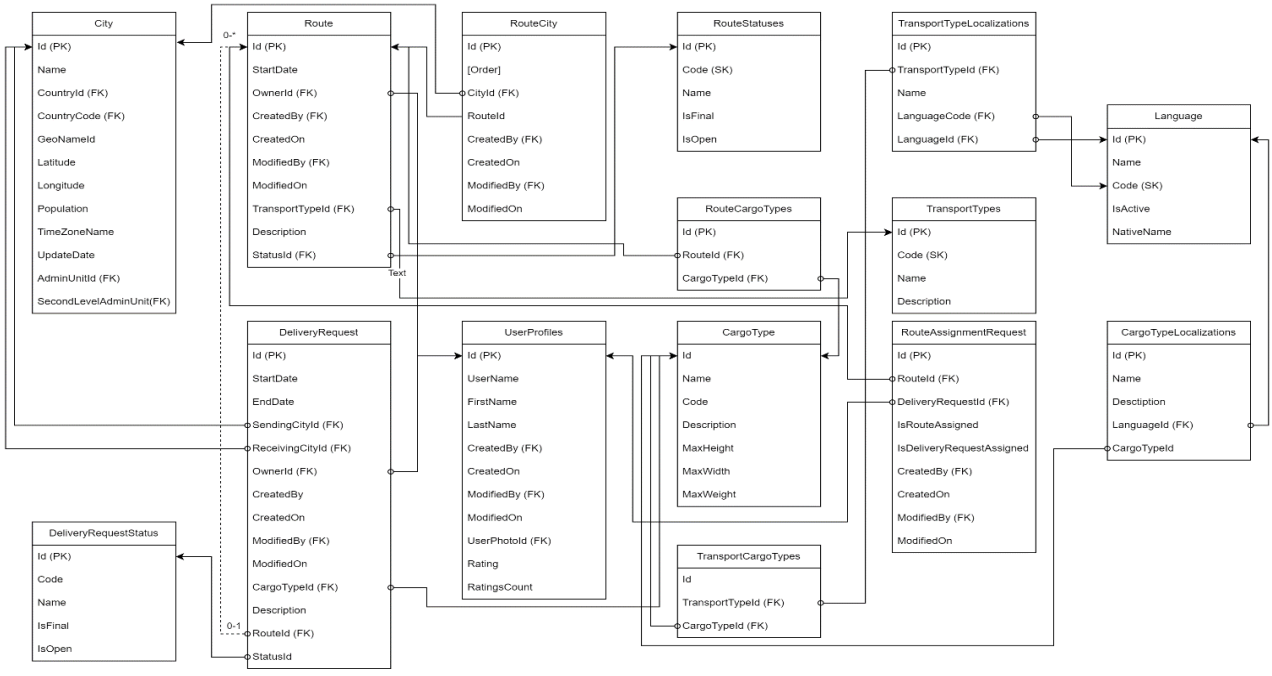


Рисунок 2.8 – Схема моделі даних побудови та пошуку маршрутів і заявок на доставку

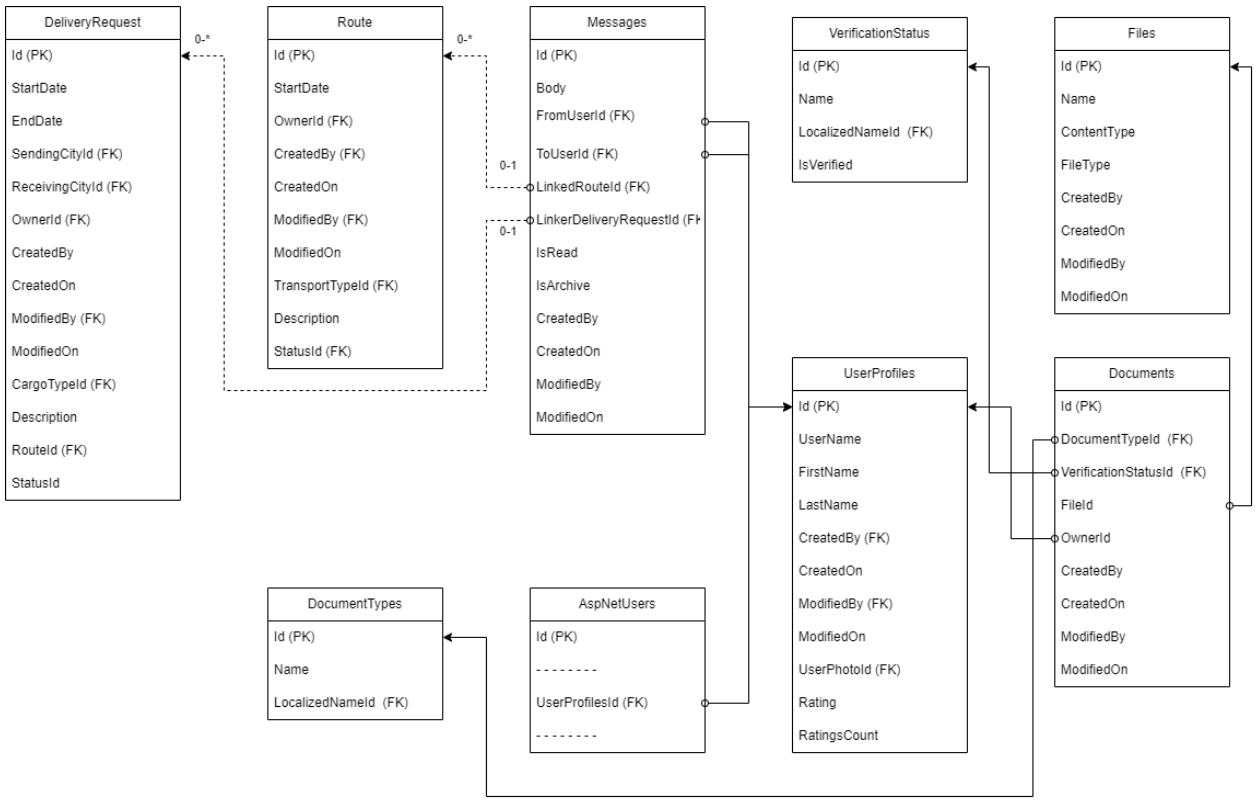


Рисунок 2.9 – Схема моделі даних користувачів та взаємодії між ними

2.2.6 Діаграма пакетів проекту серверної частини.

Для структуризації коду та модулів проекту створено діаграму пакетів, яка покликана спростити організацію та структуризацію процесу розробки серверної частини додатку (рис2.10).

Згідно схеми в пакеті Domain класи поділені на чотири групи:

– Common — Загальні класи які містять загальні властивості для класів групи Entity.

– Entity – описує об’єктну модель даних додатку.

– Events – описує події об’єктної моделі додатку.

– Enums – перечислиння які використовує об’єктна модель додатку.

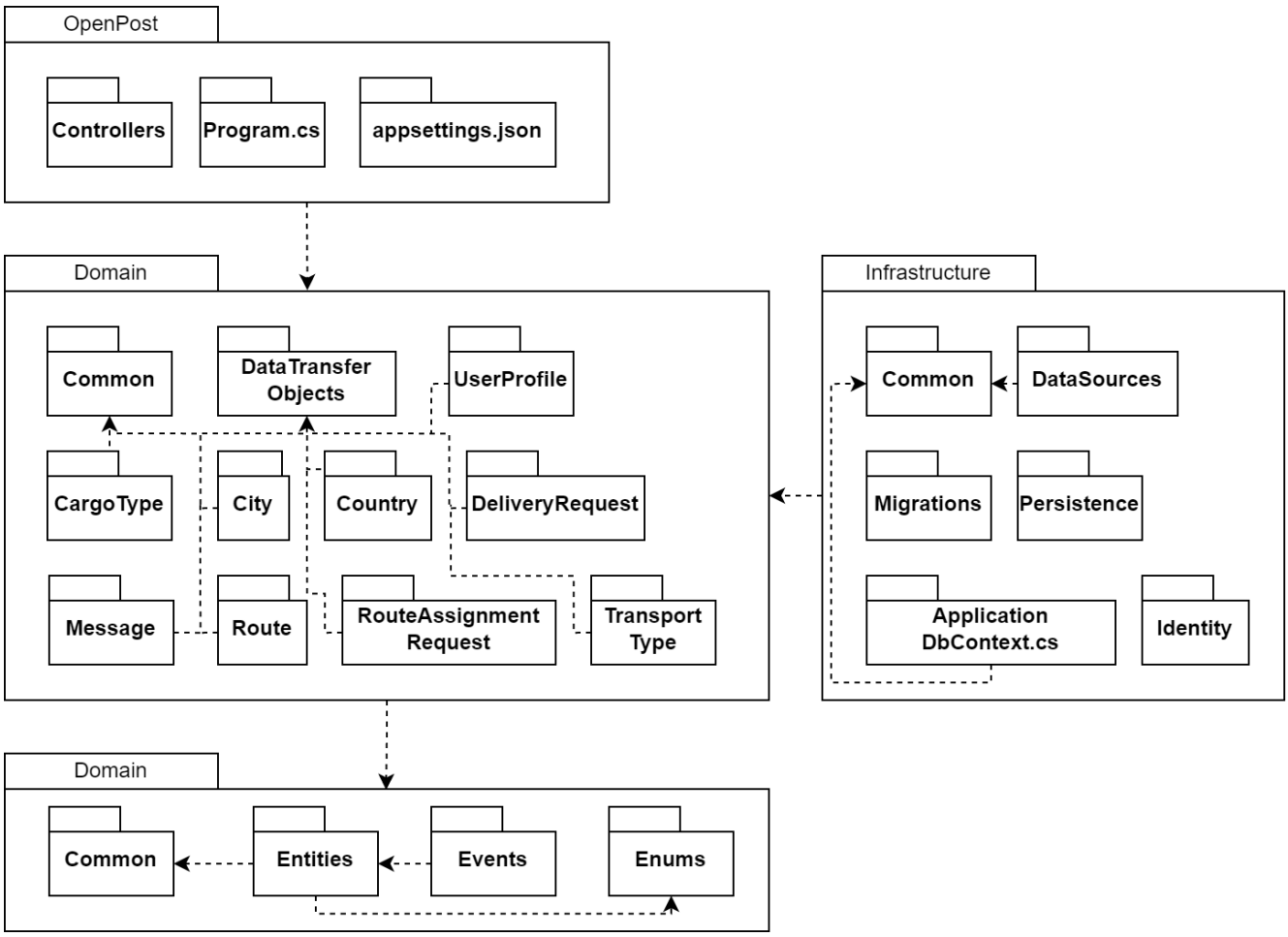


Рисунок 2.10 – Схема компонентів серверної частини додатку

В пакеті Application класи здебільшого згруповані по об’єктам, як видно на прикладі групи Country (рис.2.11) в групі об’єкта присутній другий рівень групування на команди та запити а також DTO – об’єкти транспортування даних. Окремо можна виділити групу класів Common в ньому згруповані інтерфейси сервісів потрібні для роботи всього пакету, класи налаштування маппінгу та інтерфейс для DTO класів.

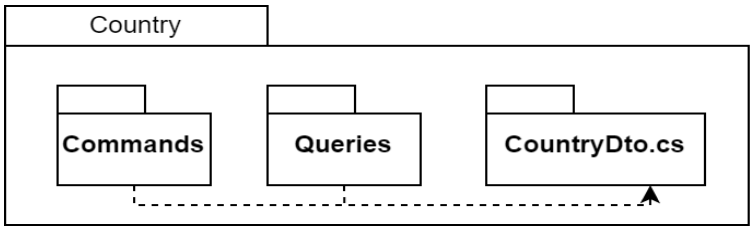


Рисунок 2.11 – Компоненти групи Country

В пакеті Application група Common – відповідає за загальні інтерфейси цього компоненту. В групі DataSources — знаходяться реалізації інтерфейсів доступу до даних які потрібні компоненту Application для роботи з даними і побудовані за шаблонами «репозиторій» та «одиниця роботи». Група Identity відповідає за сервіси і механізми ідентифікації та авторизації користувачів. Persistence містить класи налаштування для конвертації моделі даних в таблиці бази даних. ApplicationDbContext.cs описує контекст бази даних для забезпечення роботи Entity Framework.

Пакет OpenPost є шаром презентації і являє собою WebApi проект. Містить контролери з API методами та файли конфігурації, є точкою входу в додаток.

Всі чотири вказані пакети компілюються в окремі dll файли.

2.2.6 Схема збору та розгортання файлів GeoNames з текстових файлів до бази даних

Для розгортання даних географічних одиниць було розгорнуто окрему БД більш наближену до структури файлів (рис.2.12). Текстові файли експортовані до окремої бази даних та конвертовані за розробленою схемою (рис.2.13) до основної бази додатку зі створенням зв’язків між сутностями.

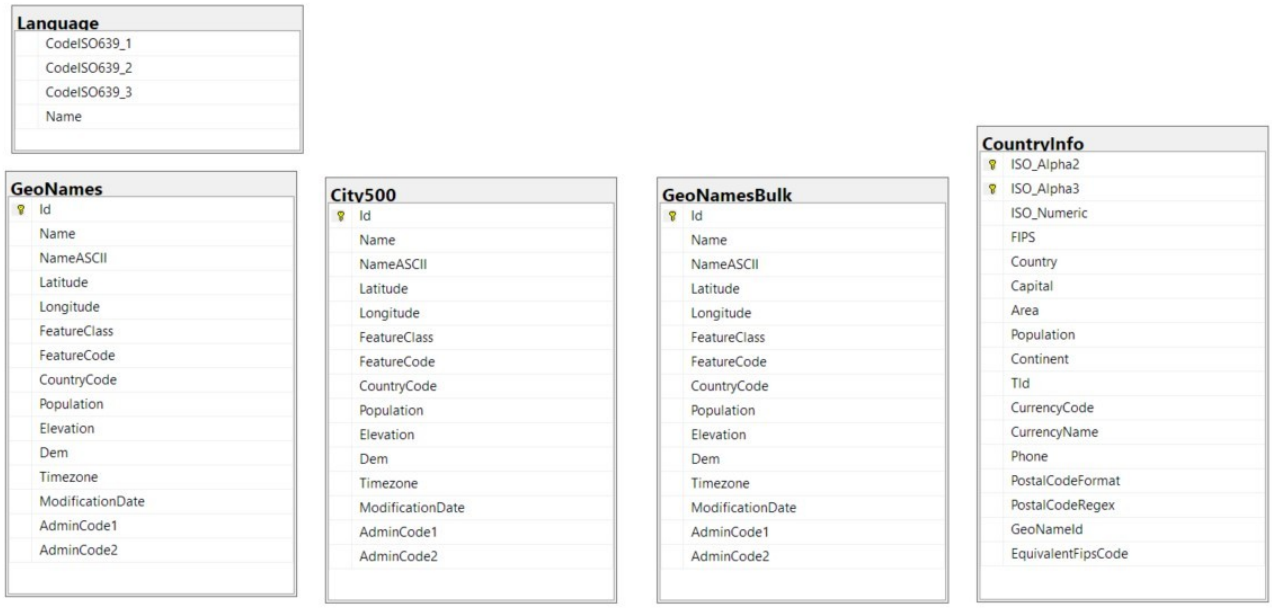


Рисунок 2.12 – Структура проміжної бази GeoNames

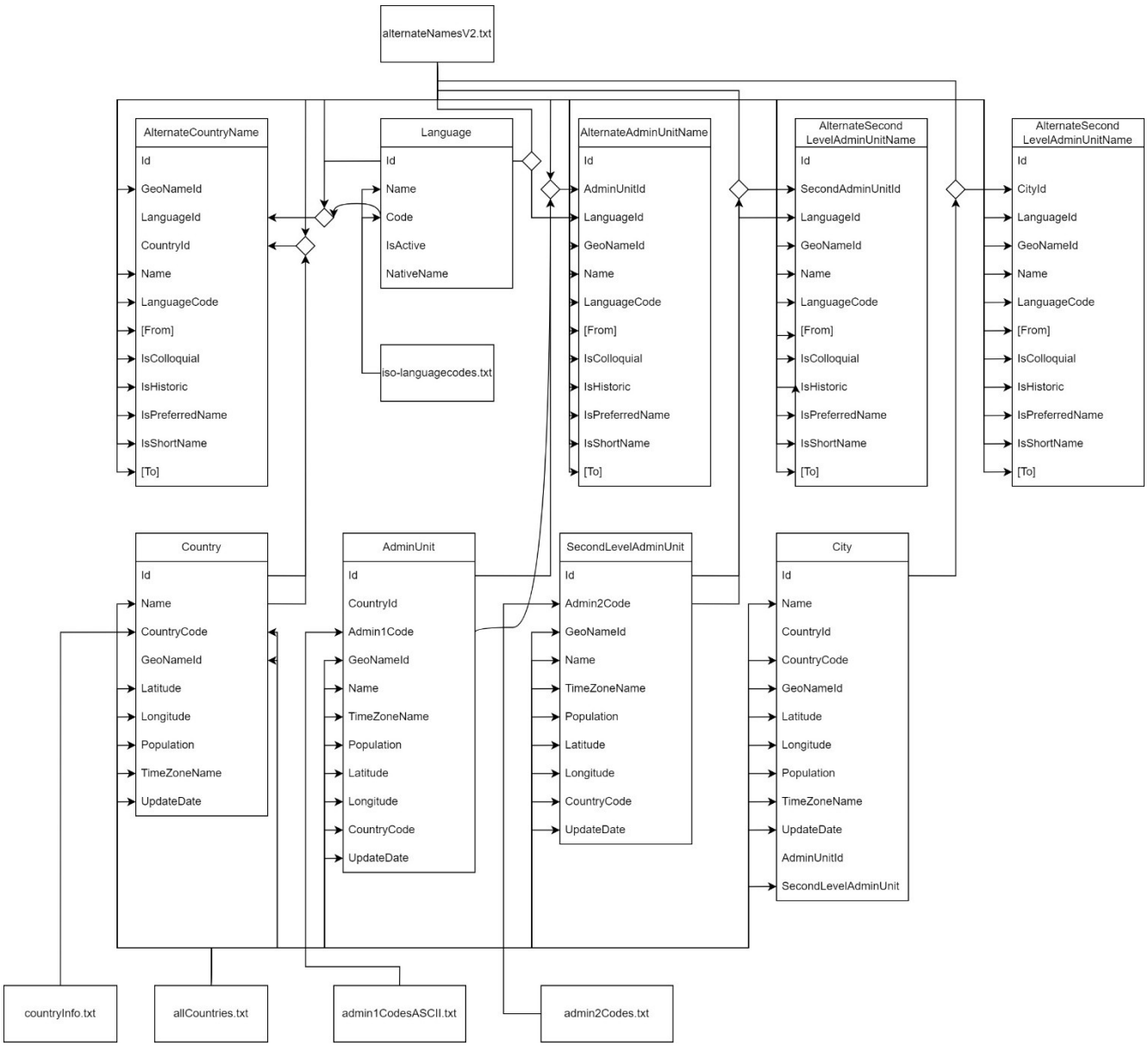


Рисунок 2.13 – Схема розгортання даних з вайлів GeoNames.

2.3 Опис класів додатку

2.3.1 Класи Domain рівня

Під час розробки серверної частини додатку були розроблені класи в компоненті Domain, частина з них була вже описана в схемах даних. Далі наведене класи які не відображені на попередніх схемах:

– Абстрактний клас BaseEntity — є базовим класом для всіх сутностей доменної моделі. В ному визначене поле Id типу Guid яке є унікальним ідентифікатором об’єкту.

– Абстрактний клас BaseAuditableEntity — наслідує клас BaseEntity та розширяє його додаючи поля з інформацією про дату створення та змінення і ідентифікатор користувача який зробив ці зміни. Використовується як базовий клас для тих сутностей які створюють і редагують користувачі, такі як повідомлення, маршрути, заявки.

– Абстрактний клас BaseAlternateName — використовується як базовий для класів локалізації міст, країн та адміністративних одиниць, в нього винесені спільні для всіх класів локалізації поля, такі як альтернативна назва, мова, період коли використовувалася історична назва та інші.

– бстрактний клас BaseGeoNameEntity є базовим для класів City, Country, AdminUnit та SecondLevelAdmonUnit, в нього винесені спільні для цих класів властивості такі як координати, назва, ідентифікатор запису в базі GeoNames та інші.

– Абстрактний клас BaseEvent наслідується від інтерфейсу INotification і представляє абстрактну подію доменної моделі додатку

– Клас DeliveryRequestAddedEvent наслідується від BaseEvent та представляє собою подію додавання заявки на доставку.

– Клас DeliveryRequestChangedEvent представляє подію зміни заявки на доставку.

– Клас RouteAddedEvent представляє подію додавання маршруту.

– RouteChangedEvent — подія зміни маршруту. Далі викликаючи ці події на Application рівні через Mediator (посередник) можна на них реагувати, наприклад висилати сповіщення.

2.3.2 Класи Application рівня

– Для рівня Application було створено такі класи та інтерфейси:

– Інтерфейс ICurrentUserService описує сервіс отримання поточного користувача. Описує властивість UserId, та методи GetUserProfileIdAsync і GetUserProfileAsync.

– Інтерфейс IRepository<T> описує шаблонний клас патерну репозиторій, який надає доступ до роботи з однією з сутностей в базі даних або іншому джерелі. Описує методи отримання запису, отримання колекції по фільтру, додавання, видалення та модифікацію даних.

– Інтерфейс IDataSource описує клас що реалізує шаблон одиниця роботи, агрегує в собі набір репозиторіїв та полегшує роботу з ними через, дає змогу одночасно зберегти дані в усіх репозиторіях та одночасно вивільнити їх ресурси по закінченню роботи.

– Інтерфейс ImapFrom<T> описує можливість сутності DTO до автоматичного маппінгу. Додаток за допомогою бібліотеки Automapper налаштований шукати всі класи в збірці помічені цим інтерфейсом і автоматично створювати для них конфігурацію маппінгу.

– Клас MappingProfile автоматично виявляє збірці класи помічені інтерфейсом ImapFrom та конфігурує Atomapper для них.

– Клас ConfigureServices містить метод розширення який викликається в точці старту додатку, додає та налаштовує сервіси рівня Application такі як Automapper.

Далі для кожної сутності є перелік класів DTO, команд та запитів які наслідуються від інтерфейсу Irequest<T> і також мають відповідний обробник наслідуваний від інтерфейсу IrequestHandler. Вони слугують для реалізації шаблону Mediator (посередник) відповідно інші компоненти системи через бібліотеку MediatR публікують команду або запит а посередник знаходить обробник виконує його та повертає результат. Таким чином були створені команди та запити для роботи з усіма доступними користувачам сутностями.

Також на великому наборі даних локалізації виявилося не ефективним використання вбудованого LazyLoad механізму. Тому замість отримання і перебору колекції перекладів розроблено ряд класів з методами розширення які отримують код мови та дістають з БД строку з перекладом замість колекції сутностей перекладу.

2.3.3 Класи Infrastructure рівня

Перелік та опис класів розроблених для рівня Infrastructure наступний:

– Інтерфейс IApplicationDbContext наслідується від IDisposable та описує контекст бази даних додатку.

– Клас ApplicationDbContext наслідує IdentityDbContext <ApplicationUser> та IApplicationDbContext і визначає контекст бази даних для роботи Entity Framework.

– Клас ApplicationDbContextInitializer має метод InitialiseAsync та слугує для ініціалізації і початкового наповнення БД при першому запуску додатку.

– Клас EfRepository<T> наслідник інтерфейсу Irepository<T> і реалізує його методи для використовуючи в якості джерела даних Entity Framework.

– Клас EfDataSource наслідник IDataSource, реалізує його використовуючи EfRepository.

– Клас ApplicationUser наслідник IdentityUser, слугує для представлення даних авторизації та ідентифікації користувачів, має навігаційну властивість для зв’язку з UserProfile.

– CurrentUserService реалізує ICurrentUserService з використанням бібліотеки Identity.

– JwtGenerator має метод CreateToken який генерує JWT токен для авторизації клієнтського додатку.

– AuthOption містить константи з параметрами для генерації токену.

– CreateUserCommand — команда реєстрації нового користувача.

– LogInQuery — запит на авторизацію клієнтського додатку.

– UserSession — описує відповідь на запит авторизації, містить токена, час дії токена та дані про поточного користувача.

– Група класів Persistence.Configurations відповідає за налаштування збереження сутностей в EntityFramework. Через ці класи конфігурації було

виконано оптимізацію бази даних, додані ключі, налаштовані відносини, індекси та обмеження. Що позитивно вплинуло на швидкість обробки запитів.

– ConfigureServices має шаблонний метод який конфігурує сервіси Infrastructure рівня.

2.3.4 Класи OpenPost (представлення) рівня

На рівні OpenPost було розроблено класи контролери з методами WebApi та базовий клас контролерів ApiControllerBase який надає властивість Mediator для виклику команд з контролерів. Також на цьому рівні викликаються методи конфігурації сервісів інших рівнів.

2.4 Розробка клієнтського додатку

Для можливості взаємодії з сервісом було створено клієнтський додаток у вигляді SPA – односторінкової програми на фреймворку Angular.
Для забезпечення базових функцій на основі сформованих вимог розроблено схему прототипу інтерфейсу користувача (рис.2.14).

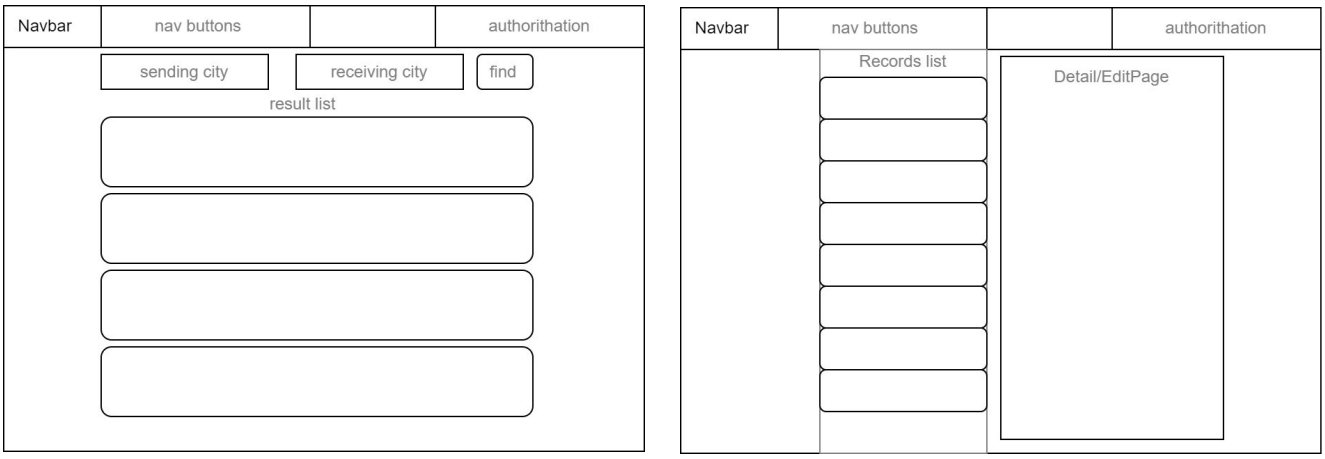


Рисунок 2.14 – Шаблон інтерфейсу клієнтського додатку

Шаблон містить прототип сторінки швидкого пошуку та роботи зі списками маршрутів та заявок. Сторінка роботи зі списком маршрутів складається з двох

частин, зліва список записів а справа вікно деталізації та редагування, при кліку по запису зі списку він підвантажується в вікно деталізації та редагування. Аналогічно працює сторінка списку заявок на доставку та повідомлень.

Адаптивність сторінки забезпечує використання стилів Bootstrap 5. Це надає змогу зручно працювати з сервісом як з персонального комп'ютера так і з планшету чи телефону.

Діаграма пакетів показує структуру компонентів клієнтського додатку (рис2.15).

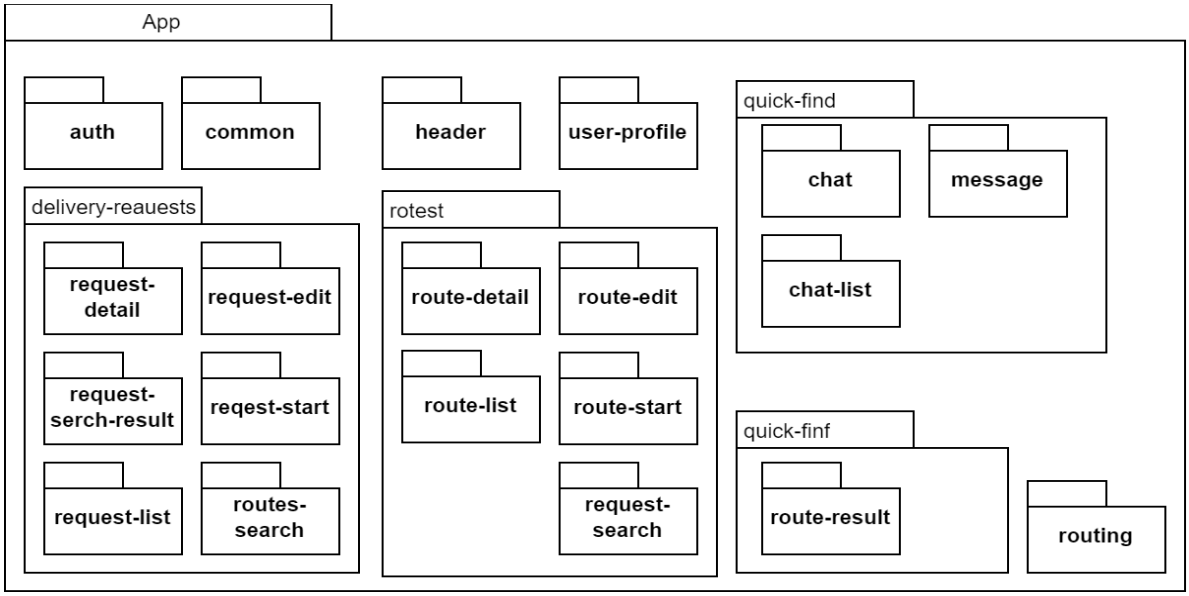


Рисунок 2.15 – Діаграма клієнтського додатку

Односторінкові додатки на Angular не перезавантажують та не переходять по сторінкам. Замість цього використовується внутрішня система маршрутизації яка завантажує певні компоненти в інші в залежності від маршруту. При цьому така маршрутизація може бути багаторівнева і рівні вкладеності будуть обробляти свою частину маршруту. На рисунку 2.16 зображено маршрутизацію клієнтського додатку що розроблено в рамках цієї бакалаврської роботи. Вона вказує на те який компонент в який батьківський завантажиться при певному маршруті. Наприклад при машруті localhost:4200/routes/create – в модуль root завантажиться RoutesComponent а в нього RouteEditComponent.

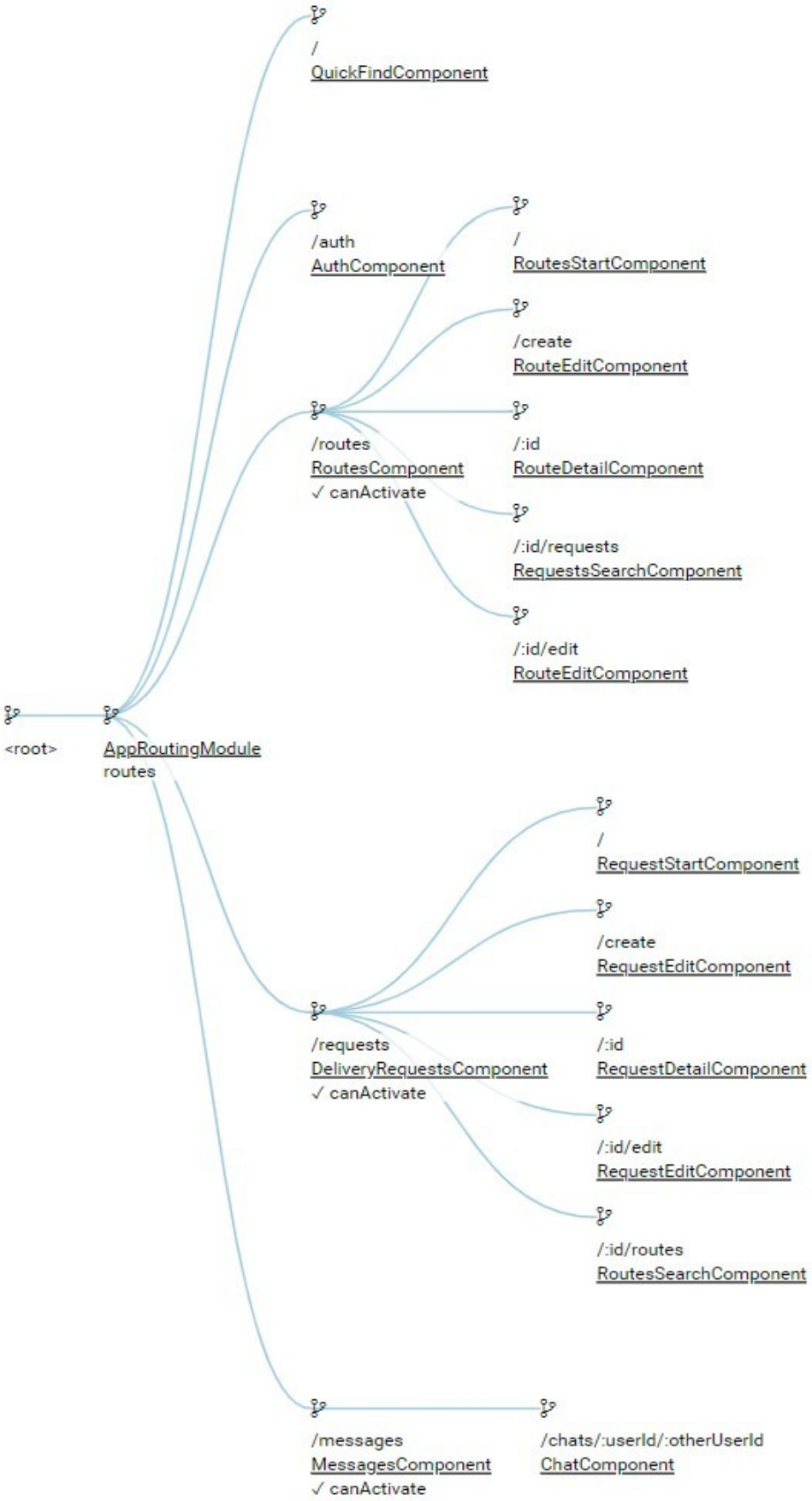


Рисунок 2.16 – Діаграма маршрутизації.

3 ОПИС ФУНКЦІОНУВАННЯ ПРОГРАМНОГО ДОДАТКУ

3.1 Форма входу та реєстрації

Для початку роботи з додатком користувачу сервісу достатньо ввести адресу сервісу в адресний рядок браузеру. При першому вході користувачу пропонується пройти процес реєстрації або авторизації. Якщо користувач вперше заходить до системи то він має натиснути кнопку «Switch to Sign Up» та перейти до форми реєстрації. Після вводу необхідних даних та успішного проходження процесу система переходить на сторінку швидкого пошуку маршрутів (рис.3.1).

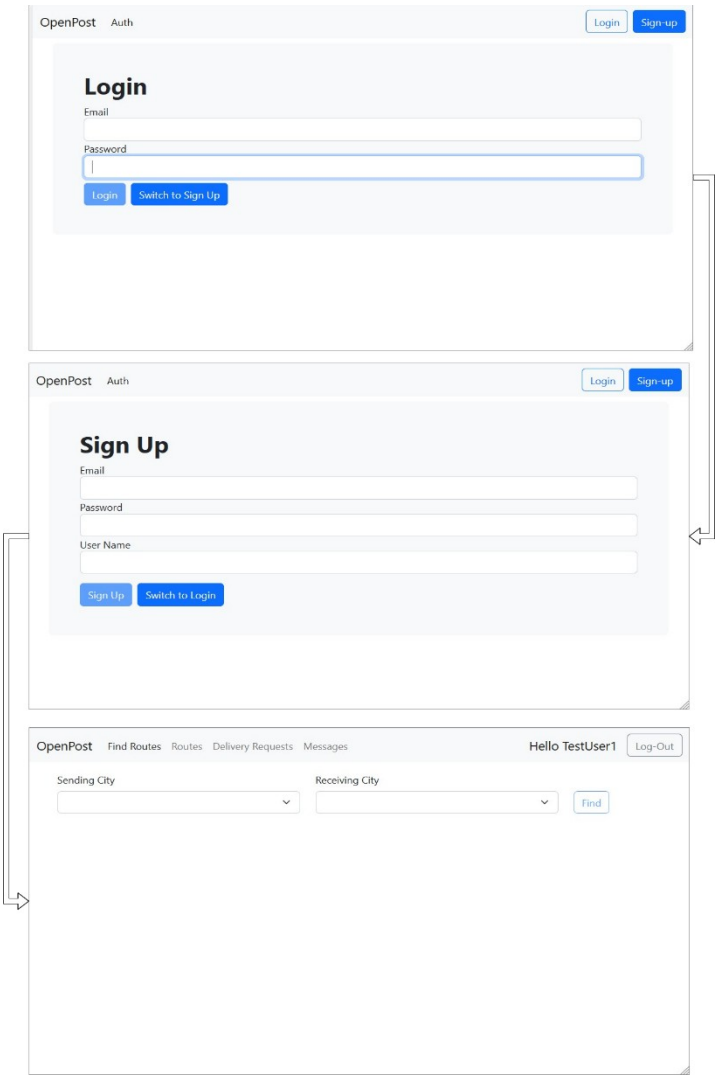


Рисунок 3.1 – Форма входу та реєстрації користувача

Форма авторизації та форма входу в систему при заповненні даних використовують певні обмеження, так поле пошти перевіряє валідність за певним шаблоном який вже в будований в фреймворк Angular. І поки в поля не будуть введені коректні значення, кнопка відправки буде заблокована. Поле паролю перевіряється на мінімальну довжину. Ці значення тако ж повторно валідуються на серверній частині де перевірку безпечності паролю забезпечує бібліотека Identity.

3.2 Елемент пошуку та вибору міст

Невід'ємною частиною роботи з сервісом є вибір міст. Оскільки в базі міститься інформація 192201 населеному пункту планети і включаючи 276798 альтернативних назв для них, для зручного пошуку та вибору розроблено компонент пошуку та вибору міст (рис.3.2). Цей компонент використовується для вибору міст при створенні маршрутів, заявок на доставку а також в формі пошуку.

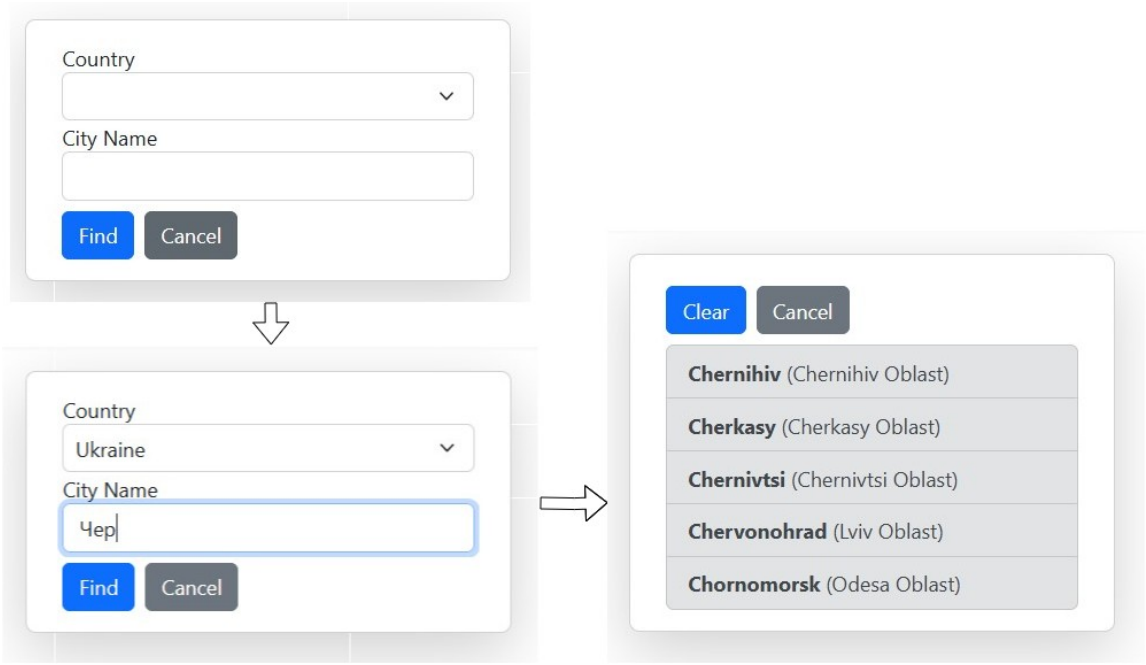


Рисунок 3.2 – Компонент пошуку та вибору міст

Робота з компонентом проходить в три етапи.

1. Вибір країни — цей крок додано для оптимізації за рахунок зменшення області пошуку однією країною.

2. Ввід частково або повністю назви міста однією з мов що використовується в країні.

3. Вибір міста з запропонованих варіантів.

3.3 Швидкий пошук маршрутів

Перше вікно, на яке потрапляє користувач після реєстрації це вікно швидкого пошуку маршрутів попутного транспорту (рис.3.3). В цьому вікні користувач може ввести місто відправлення та місто прибуття посилки і побачити наявні маршрути попутного транспорту що проходять через вказані міста.

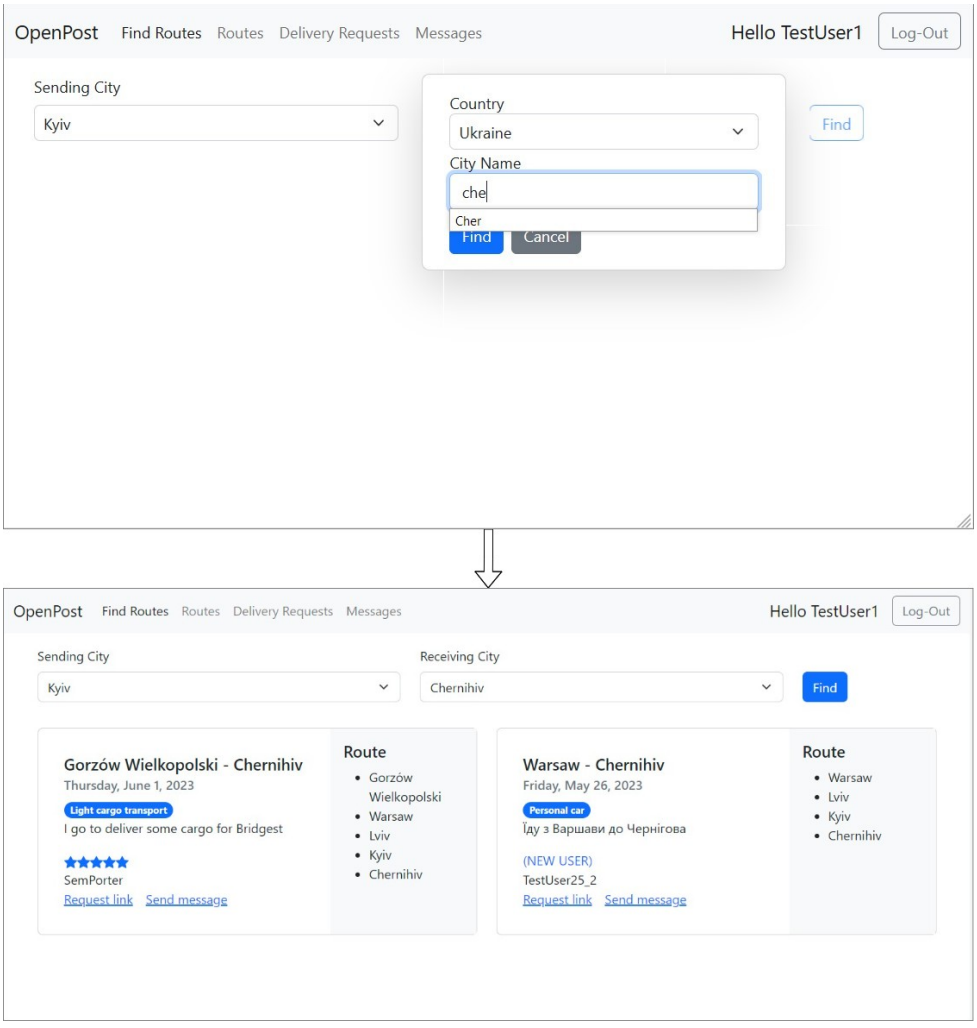


Рисунок 3.3 – Форма швидкого пошуку маршрутів попутного транспорту

Для відображення результатів створено компонент карточки результату (рис3.4). В ній відображається вся необхідна інформація, така як тип транспорту, дата відправки, міста через які він проходить, рейтинг користувача та опис маршруту. Ця карточка дозволяє при кліку на кнопку «Send message» одразу відправити повідомлення автору маршруту.

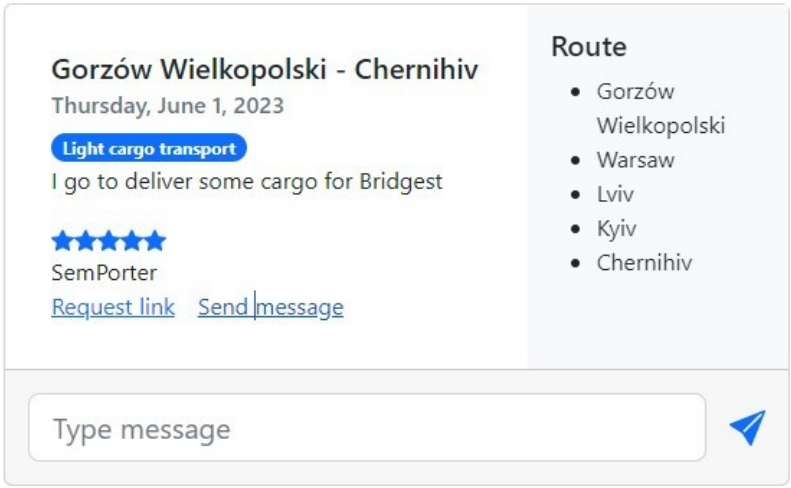


Рисунок 3.4 – Карточка результату пошуку

3.4 Робота з записами маршрутів користувача

Вікно Routes слугує для роботи з записами маршрутів користувача (рис.3.5), воно умовно ділиться на 2 частини — список та область деталізації і редагування. В лівій частині користувач обирає один із своїх маршрутів а в правій відображається його деталізація з повним описом маршруту та кнопками що надають можливість редагувати маршрут та здійснювати пошук заявок на доставку що збігаються з маршрутом.

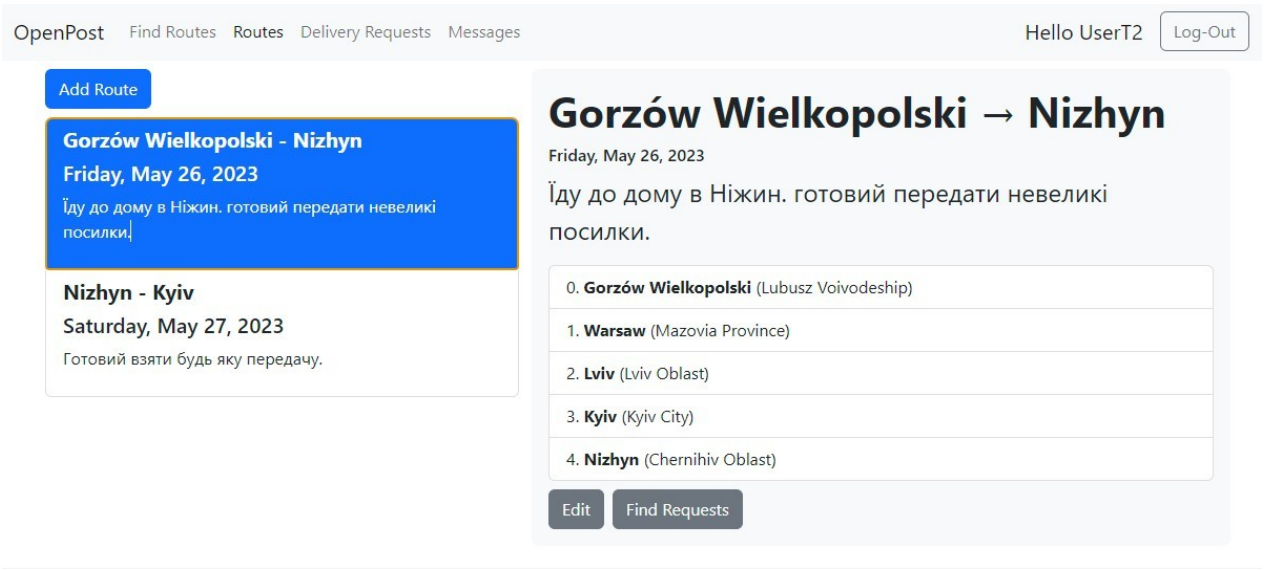


Рисунок 3.5 – Сторінка роботи з маршрутами

Для створення маршруту користувач натискає кнопку «Add Route», ця дія відкриває в лівій частині вікно редагування та створення маршруту (рис.3.6). В цьому вікні користувач вказує опис маршруту, обирає дату відправки, тип транспорту та задає та видаляє міста через які проходить маршрут.

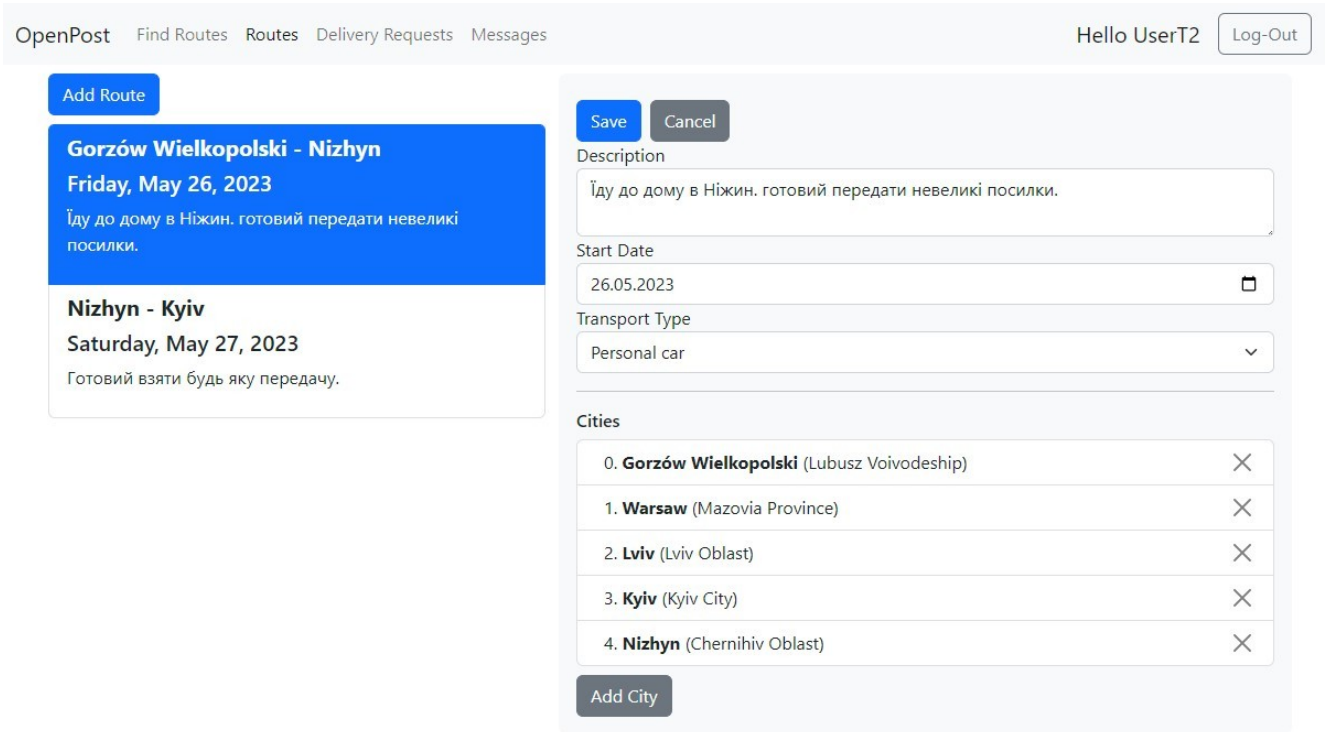


Рисунок 3.6 – Форма створення та редагування маршруту

При натисканні кнопки Find в вікні деталізації відбувається пошук заявок що збігаються з маршрутом (рис.3.7). В цьому вікні користувачу відображається список що складається з карточок результатів пошуку заявок. Вони дозволяють швидко відправити повідомлення автору заявок, або надіслати запит на поєднання заявки та маршруту.

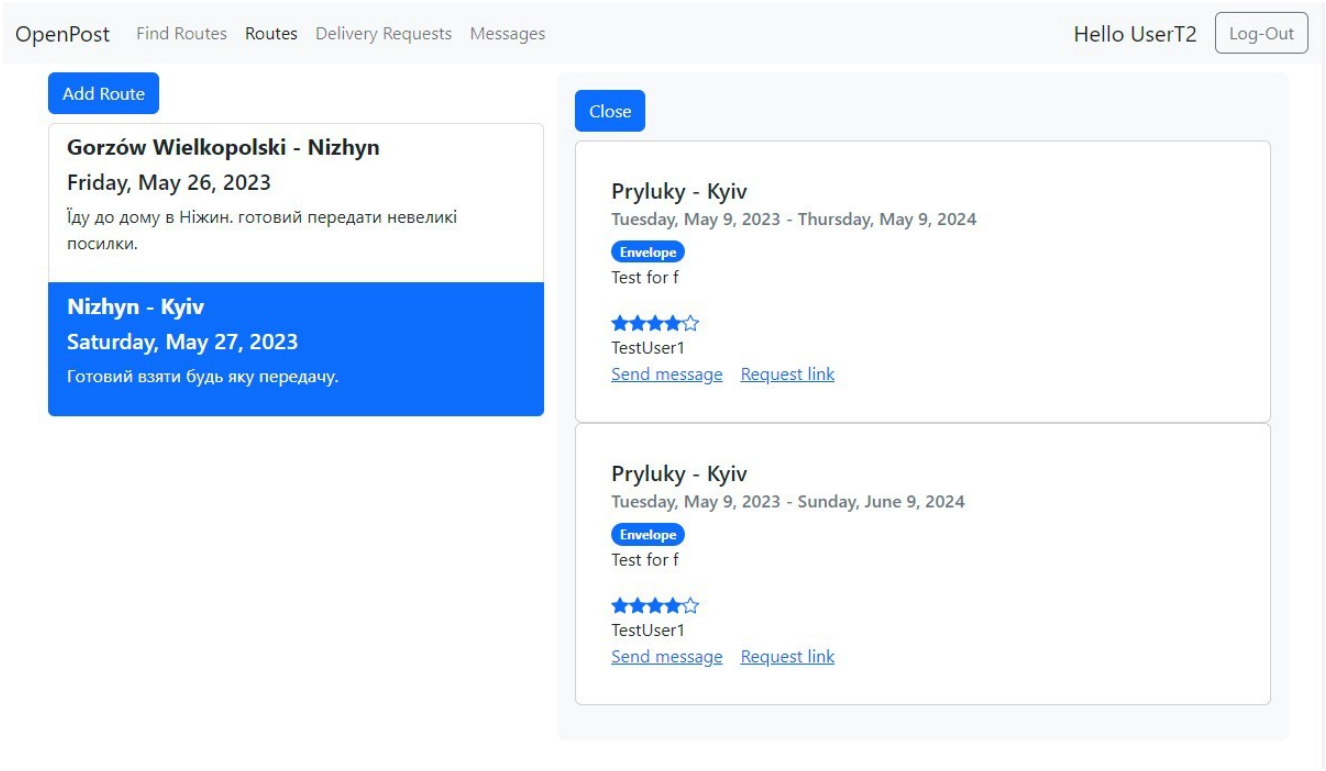


Рисунок 3.7 – Результати пошуку заявок по маршруту

3.4 Робота з записами заявок на доставку

Аналогічно вікну роботи з маршрутами, вікно роботи з записами заявок на доставку (рис.3.8) надає інструменти роботи з записами заявок. І дозволяє виконувати такі дії як редагування, створення заявок та пошук маршрутів попутного транспорту що збігаються з параметрами заявки та надсилати повідомлення авторам маршрутів.

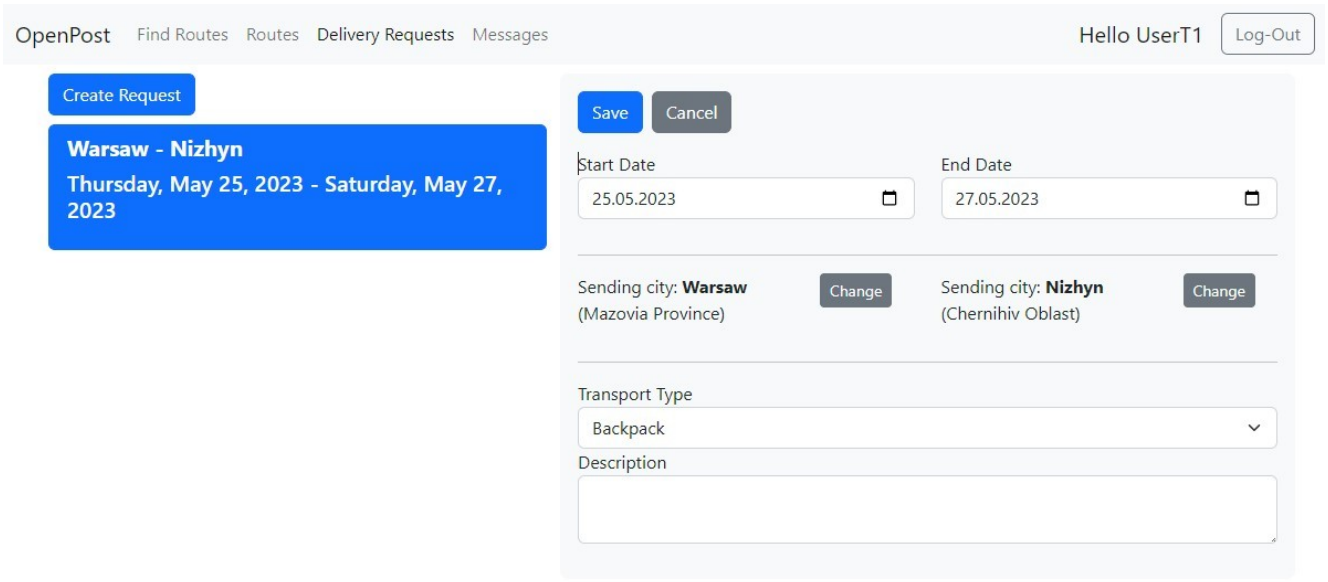


Рисунок 3.8 – Вікно роботи з записами заявок на доставку

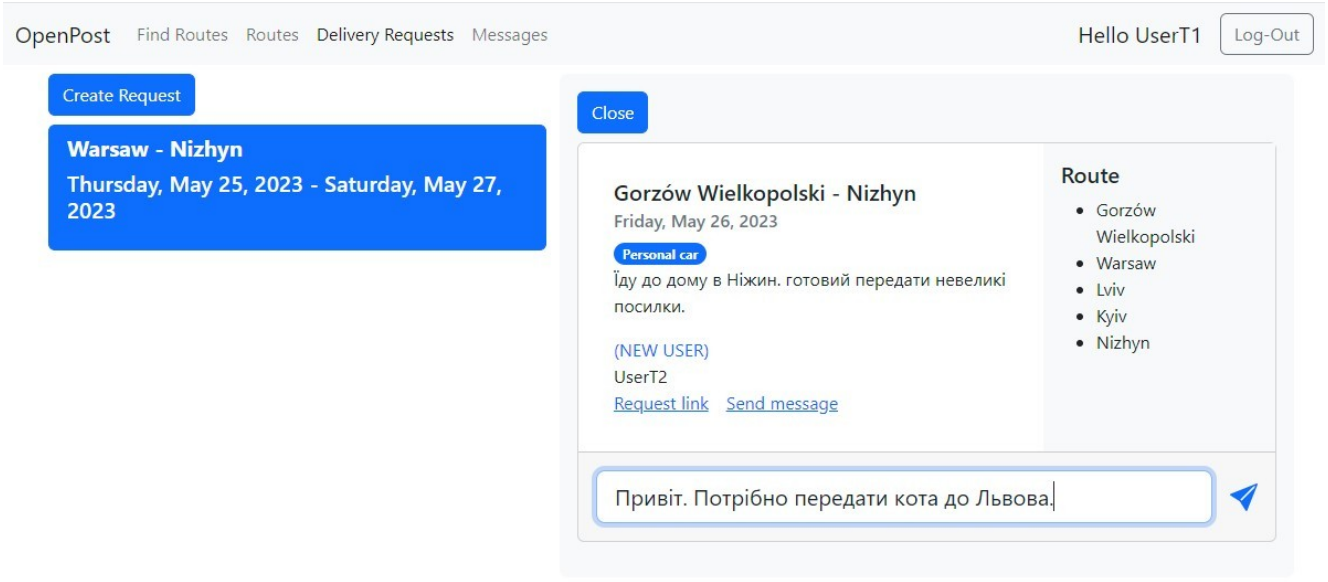


Рисунок 3.9 – Пошук маршруту попутного транспорту та надсилання повідомлень автору

3.5 Робота повідомленнями користувача

Для доступу до повідомлень слугує вікно повідомлень (рис.3.10). В лівій частині відображається список чатів в карточці запису якого вказано нікнейм співрозмовника, загальна кількість повідомлень та кількість непрочитаних повідомлень. По кліку на карточку в лівій частині завантажується чат з повідомленнями користувача та співрозмовника який дозволяє читати та відправляти повідомлення.

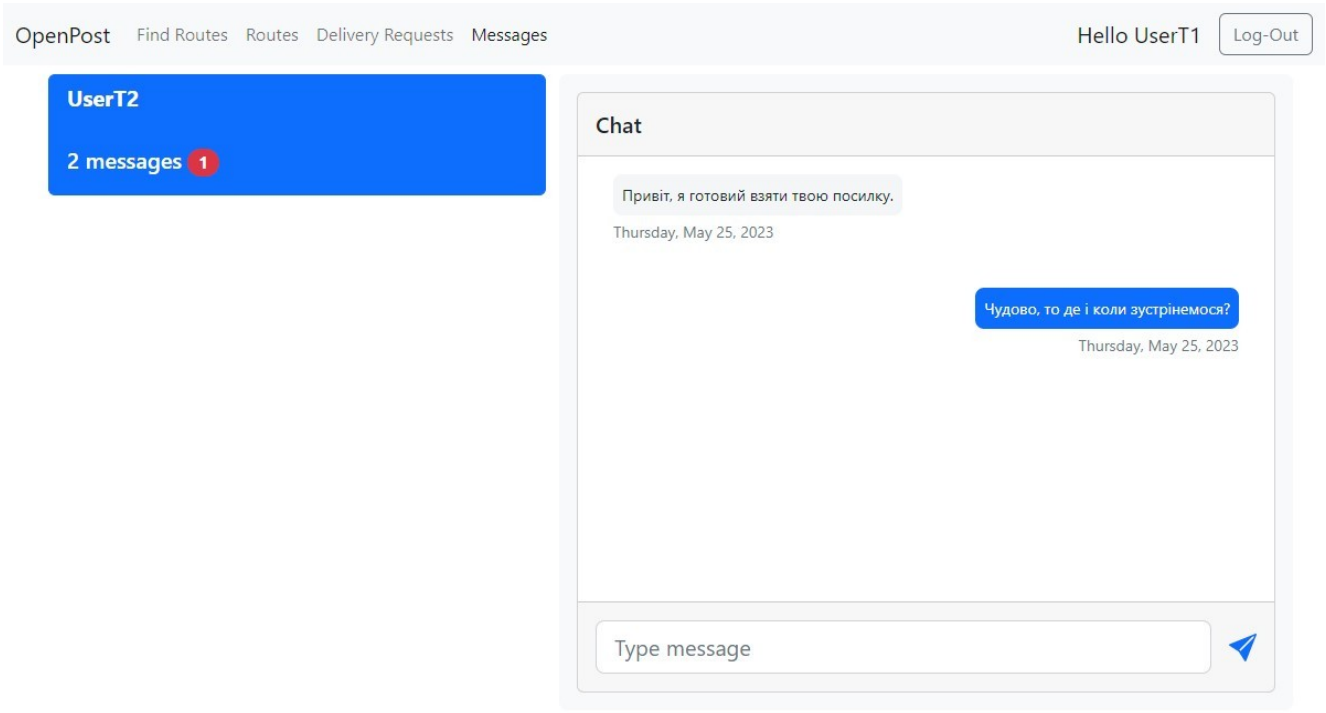


Рисунок 3.10 – вікно роботи з повідомленнями

4 ТЕСТУВАННЯ ПРОГРАМНОГО ДОДАТКУ

Тестування програмного забезпечення - це процес оцінки взаємодії програми з використанням вибраного набору тестів для переконання в тому, що її реальна робота відповідає очікуваній поведінці та властивостям. Це ключова частина контролю якості програмного забезпечення. У цій секції розглянуто основні аспекти, що вимагають тестування в нашому додатку, сформовано тестові сценарії для перевірки функціоналу додатку та проведено тестування програми згідно з розробленим планом тестування.

4.1 Модульне тестування додатку

Модульне тестування є ключовим етапом в процесі розробки програмного забезпечення, оскільки воно спрямоване на перевірку окремих модулів (або компонентів) додатку. Такі тести швидко дають інформацію про стан модулів системи і знижують ризики регресії при внесенні змін. Кожен модуль перевіряється ізольовано від інших, що допомагає швидко виявляти і вирішувати проблеми. Для тестування серверної частини використано тестовий фреймворк NUnit, структура тестів відповідає шаблону AAA – Arrange, Act, Assert. Для забезпечення ізоляції в тестах, для створення імітацій сервісів та класів інших пов’язаних модулів використано бібліотеки Moq та NSubstitute, Які дають змогу швидко та зручно створювати імітацію стану та поведінки елементів системи.

4.2 Тестування швидкодії

Для тестування швидкодії основних запитів використано лог додатку з часом виконання запитів та профайлер «ms sql Server Management Studio». Знайдено повільні запити та проведено аналіз плану виконання запитів, згідно рекомендацій профайлера додано індекси та обмеження до таблиць що в деяких випадках пришвидшило виконання запитів до 10 разів.

4.3 Розробка тест-кейсів для тестування методом чорного ящика

Метод чорного ящика, відомий також як тестування на рівні функціональності, - це вид тестування, коли тестер не володіє інформацією про внутрішню структуру програмного продукту. Цей метод тестування базується на вимогах та специфікаціях системи.

Процес тестування методом чорного ящика можна розбити на декілька основних кроків:

– Розуміння вимог: Тестер повинен зрозуміти, що очікується від системи, в тому числі її функціональності, обмеження, взаємодії з іншими системами, тощо.

– Розробка тестових сценаріїв: На основі вимог до системи тестер розробляє тестові сценарії, що охоплюють можливі використання та умови системи.

– Виконання тестів: Тестер запускає тестові сценарії на системі, надаючи вхідні дані і відслідковуючи вихідні результати.

– Порівняння вихідних даних з очікуваними: Після отримання результатів тестер порівнює їх з очікуваними результатами.

– Запис результатів: Тестер фіксує результати тестування, включаючи використані вхідні дані, отримані вихідні дані та статус проходження тесту (пройшов/не пройшов).

Цей метод є досить ефективним для тестування функціональності програмного продукту.

Для тестування додатку було спроектовано тестові випадки з певною послідовністю дій, покликано перевірити відповідність функцій системи очікуваній поведінці.

Тести поділені на негативні OK які перевіряють очікувану поведінку при введенні коректних даних та негативні NOK які перевіряють реакцію системи на виключення при введенні некоректних даних (таблиці 4.1-4.3).

Таблиця 4.1 – Тестові випадки вікна входу в систему

№ тесту	ОК/ NOK	Кроки тесту	Очікуваний результат
1	ОК	1. Натиснута кнопка входу в систему 2. Введено валідну адресу електронної пошти (UserT1@gmail.com) 3. Введено валідній пароль (UserT1-) 4. Натиснута кнопка Login	-Система успішно пройшла процес авторизації. -На навігаційні панелі з’явилося ім’я користувача та кнопка Logout -Система перейшла на сторінку швидкого пошуку
2	NOK	1. Натиснута кнопка входу в систему 2. Введено не валідну дресу електронної пошти (UserT1..gmail.com) 3. Введено валідній пароль (UserT1-)	-Кнопка Login заблокована
3	NOK	1. Натиснута кнопка входу в систему 2. Введено валідну адресу електронної пошти (UserT1@gmail.com) 3. Введено не валідній пароль ([пробіл])	-Кнопка Login заблокована

Таблиця 4.2 – Тестові випадки пошуку маршрутів попутного транспорту з врахуванням порядку слідування транспорту

№ тесту	ОК/ NOK	Кроки тесту	Очікуваний результат
1	ОК	1. Попередньо створено маршрут що проходить через міста Варшава-Київ-Чернігів. 2. Користувач входить в систему. 3. Користувач переходить до форми пошуку маршрутів. 4. Поле міста відправлення заповнює значенням -Варшава 5. Поле міста прибуття заповнює значенням — Чернігів. 6. Натиснута кнопка Find	-Знайдено маршрут створений на кроці 1.
2	ОК	1. Попередньо створено маршрут що проходить через міста Варшава-Київ-Чернігів. 2. Користувач входить в систему. 3. Користувач переходить до форми пошуку маршрутів. 4. Поле міста відправлення заповнює значенням - Київ 5. Поле міста прибуття заповнює значенням — Чернігів. 6. Натиснута кнопка Find	-Знайдено маршрут створений на кроці 1.

Продовження таблиці 4.2 – Тестові випадки пошуку маршрутів попутного транспорту з врахуванням порядку слідування транспорту

№ тесту	OK/ NOK	Кроки тесту	Очікуваний результат
3	NOK	1. Попередньо створено маршрут що проходить через міста Варшава-Київ-Чернігів. 2. Користувач входить в систему. 3. Користувач переходить до форми пошуку маршрутів. 4. Поле міста відправлення заповнює значенням - Чернігів 5. Поле міста прибуття заповнює значенням - Варшава. 6. Натиснута кнопка Find	-Маршрут створений на кроці 1 не відображається в списку знайдених.

Таблиця 4.3 – Тестові випадки створення запису заявки на доставку

№ тесту	OK/ NOK	Кроки тесту	Очікуваний результат
1	OK	1. Користувач входить в систему. 2. Користувач переходить до вікна роботи з заявками. 3. натиснута кнопка Create Request 4. В лівій частині відображено вікно створення заявки 5. Поле Start Date заповнюється значенням поточної дати. 6. Поле End Date заповнюється значенням поточної дати + 1 день. 7. Обране місто відправлення.	-В списку заявок з’явився новий запис. -Вікно редагування і деталізації перейшло в режим деталізації. -Вікні деталізації відображається інформація про новий запис.

Продовження таблиці 4.3 – Тестові випадки створення запису заявки на доставку

№ тесту	OK/ NOK	Кроки тесту	Очікуваний результат
		8. Оране місто прибуття. 9. Обрано тип посилки. 10. Поле Description заповнюється довільним текстом. 11. Натиснута кнопка Save	-
2	NOK	1. Користувач входить в систему. 2. Користувач переходить до вікна роботи з заявками. 3. натиснута кнопка Create Request 4. В лівій частині відображено вікно створення заявки 5. Поле Start Date заповнюється значенням поточної дати + 1 день. 6. Поле End Date заповнюється значенням поточної дати. 7. Обране місто відправлення. 8. Оране місто прибуття. 9. Обрано тип посилки. 10. Поле Description заповнюється довільним текстом.	-Кнопка Save заблокована. -Відображається валідаційне повідомлення про не валідність дат.
3	NOK	1. Користувач входить в систему. 2. Користувач переходить до вікна роботи з заявками. 3. натиснута кнопка Create Request 4. В лівій частині відображено вікно створення заявки	-Кнопка Save заблокована.

Продовження таблиці 4.3 – Тестові випадки створення запису заявки на доставку

№ тесту	OK/ NOK	Кроки тесту	Очікуваний результат
		5. Поле Start Date заповнюється значенням поточної дати. 6. Поле End Date заповнюється значенням поточної дати + 1 день. 7. Обране місто відправлення. 9. Обрано тип посилки. 10. Поле Description заповнюється довільним текстом.	

ВИСНОВКИ

Всі задачі, поставлені під час виконання бакалаврської роботи, було виконано в повному обсязі.

1. Проведено аналіз процесу пошуку попутного транспорту для передачі посилок. Розглянуті особливості процесу. Визначено основні етапи процесу, поняття та об’єкти процесу. На основі аналізу сформовані основні вимоги до об’єктної моделі додатку. Визначено переваги використання платформи .Net та мови C# для реалізації веб сервісу пошуку попутного транспорту для передачі посилок.

2. Визначено основні функціональні та нефункціональні вимоги до сервісу. Обрано та спроектовано архітектуру для реалізації сервісу пошуку попутного транспорту. В проектування використано UML діаграми.

3. Розроблено програмне забезпечення серверної частини сервісу з використанням платформи .Net та мови C#. Та клієнтський додаток з використанням фреймвоку Angular та мовиTypeScript

4. Сервіс забезпечує всі необхідні функціональні можливості:

– Створення маршрутів транспорту.

– Пошук маршрутів попутного транспорту з врахування напрямку слідування.

– Створення заявок на доставку.

– Пошук попутного транспорту маршрут якого збігається з заявкою.

– Пошук заявок що збігаються з записом маршруту попутного транспорту.

– Обмін повідомленнями між користувачами.

5. Проведено тестування розробленого сервісу пошуку попутного транспорту для передачі посилок. Сервіс відповідає визначеним функціональним та нефункціональним вимогам. Проведено оптимізацію.

6. Перспектива розвитку додатку полягає в додаванні верифікації користувачів з перевіркою документів для підвищення безпеки, відслідковування

статусу посилки, впровадження системи оцінки доставки та рекомендації ціни, додавання сповіщень про появу нових маршрутів що відповідають створеній заявці.

ПЕРЕЛІК ПОСИЛАНЬ

1. Qi, Wei & Li, Lefei & Liu, Sheng & Shen, Max. (2018). Shared Mobility for Last-Mile Delivery: Design, Operational Prescriptions, and Environmental Impact. Manufacturing & Service Operations Management. 20. 10.1287/msom.2017.0683.
2. Bellos, Ioannis & Ferguson, Mark & Toktay, Beril. (2017). The Car Sharing Economy: Interaction of Business Model Choice and Product Line Design. Manufacturing & Service Operations Management. 19. 10.1287/msom.2016.0605.
3. 112kilo [Електронний ресурс] Режим доступу до ресурсу: https://112kilo.com/.
4. sharry.cc [Електронний ресурс] Режим доступу до ресурсу: https://sharry.cc/.
5. Olx.ua [Електронний ресурс] Режим доступу до ресурсу: https://www.olx.ua/.
6. Facebook [Електронний ресурс] Режим доступу до ресурсу: https://www.facebook.com/.
7. Що таке RESTful API? [Електронний ресурс] // codeguida. – 2016. – Режим доступу до ресурсу: https://codeguida.com/post/601.
8. Richardson L. RESTful Web APIs / L. Richardson, M. Amundsen, S. Ruby., 2013.
9. Smith S. Architect Modern Web Applications with ASP.NET Core and Azure / Steve Smith. – Redmond, Washington 98052-6399: Microsoft Developer Division, .NET, and Visual Studio product teams, 2023.
10. A tour of the C# language [Електронний ресурс] // Microsoft Learn. – 2023. – Режим доступу до ресурсу: https://learn.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/.
11. Jeffrey R. CLR via C# / Richter Jeffrey. – Redmond, Washington 98052-6399: Microsoft press, 2012.
12. Price M. C# 11 and .NET 7 – Modern Cross-Platform Development Fundamentals / Mark J. Price., 2022.

13. Steve F. Pro TypeScript: Application-Scale JavaScript Development / Fenton Steve., 2014.
14. Introduction to the Angular [Електронний ресурс] – Режим доступу до ресурсу: https://angular.io/docs.
15. Eric E. Domain-Driven Design: Tackling Complexity in the Heart of Software / Evans Eric., 2003.
16. Martin F. UML Distilled: A Brief Guide to the Standard Object Modeling Language / Fowler Martin., 2003.
17. GeoNames [Електронний ресурс] – Режим доступу до ресурсу: https://www.geonames.org/.

ДОДАТОК А.
ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ (ПРЕЗЕНТАЦІЯ)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕСТИТЕТ ТЕЛЕКОМУНІКАЦІЙ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Розробка сервісу "OpenPost" пошуку попутного транспорту для передачі посилок мовою C#

Виконав
студент групи ППЗ-51
Горбань Андрій Миколайович
Керівник
к.т.н., доцент, доцент кафедри ІПЗ Золотухіна Оксана Анатоліївна

Київ - 2023

ОБ’ЄКТ, ПРЕДМЕТ, МЕТА ДОСЛІДЖЕННЯ

Мета дослідження: спростити пошук попутного транспорту для передачі посилок за рахунок автоматизації процесу з використанням веб-сервісу, розробленого мовою C#.

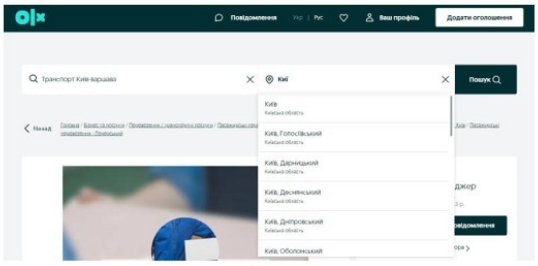
Об'єкт дослідження: процес пошуку попутного транспорту для передачі посилок.

Предмет дослідження: програмне забезпечення для автоматизації пошуку попутного транспорту для передачі посилок.

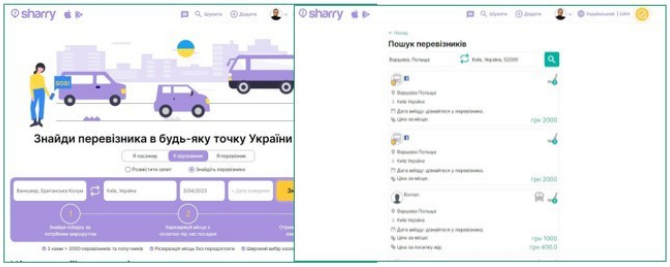
ЗАДАЧІ ДИПЛОМНОЇ РОБОТИ

1. Дослідити процес пошуку попутного транспорту для передачі посилок.
2. Проаналізувати способи та програмні рішення що використовуються для вирішення задач пошуку попутного транспорту для передачі посилок.
3. Сформулювати вимоги до програмного забезпечення з урахуванням недоліків існуючих засобів
4. Провести аналіз засобів та технологій розробки програмного забезпечення для виконання кваліфікаційної роботи.
5. Спроектувати та розробити сервіс пошуку попутного транспорту для передачі посилок.

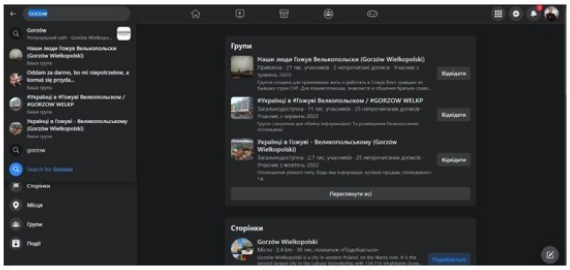
АНАЛІЗ АНАЛОГІВ



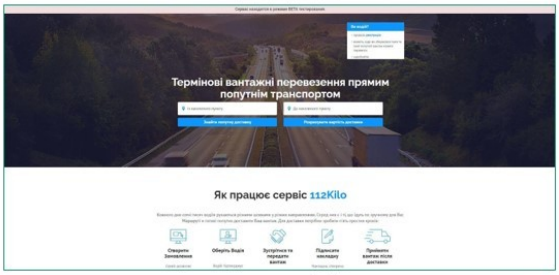
Olx.ua



Sharry



Facebook



112kilo

АНАЛІЗ АНАЛОГІВ

Характеристика	Olx.ua	Facebook	Sharry	112kilo	OpenPost
Типи транспорту і посилок	Обмеження ві дсутні	Обмеження відсутні	Тільки маршрутний транспорт	Тільки вантажний комерційний транспорт	Обмеження відсутні
Спосіб пошуку попутного транспорту	Відсутні спеціальні фільтри	Відсутні спеціальні фільтри	Є спеціальні фільтри	Є спеціальні фільтри	Є спеціальні фільтри
Область пошуку попутного транспорту	Україна	Україна та за кордоном	За кордоном - декілька країн (Європа)	За кордоном - декілька країн	Обмеження відсутні
Типовий розмір посилок	Обмеження відсутні	Обмеження відсутні	Посилки середнього та малого розміру	Крупногабаритні посилки	Обмеження відсутні
Рівень безпеки пересилки	Низький	Низький	Прийнятний	Високий	Прийнятний
Особливості пошуку назв міст	Точний збіг назви певною мовою	Точний збіг назви певною мовою	Точний збіг назви певною мовою	Точний збіг назви певною мовою	Пошук назв міст різними мовами

ВИМОГИ ДО ВЕБ-СЕРВІСУ

Функціональні вимоги:

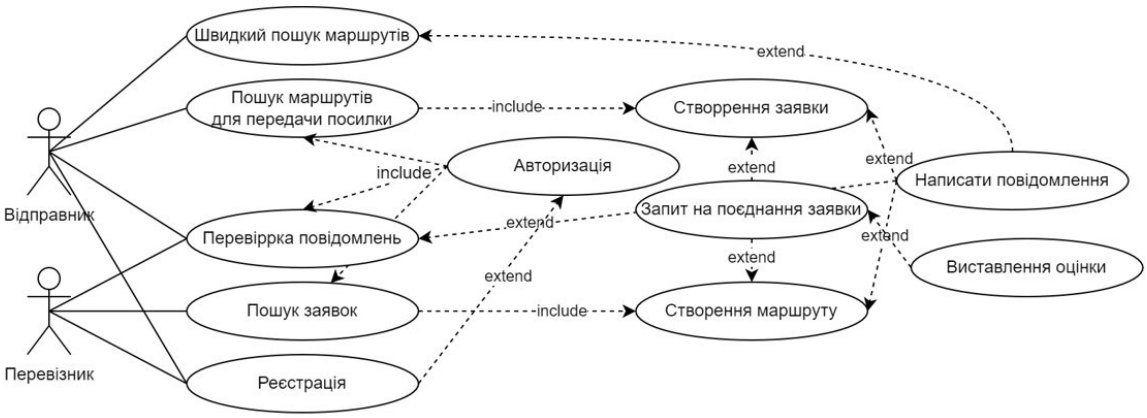
1. Пошук попутного транспорту по містам відправлення та прибуття.
2. Створення заявки на доставку.
3. Створення маршруту доставки.
4. Обмін повідомленнями між замовником та перевізником.
5. Пошук підходящого для заявки попутного транспорту.
6. Пошук підходящих заявок на доставку по маршруту.

Нефункціональні вимоги:

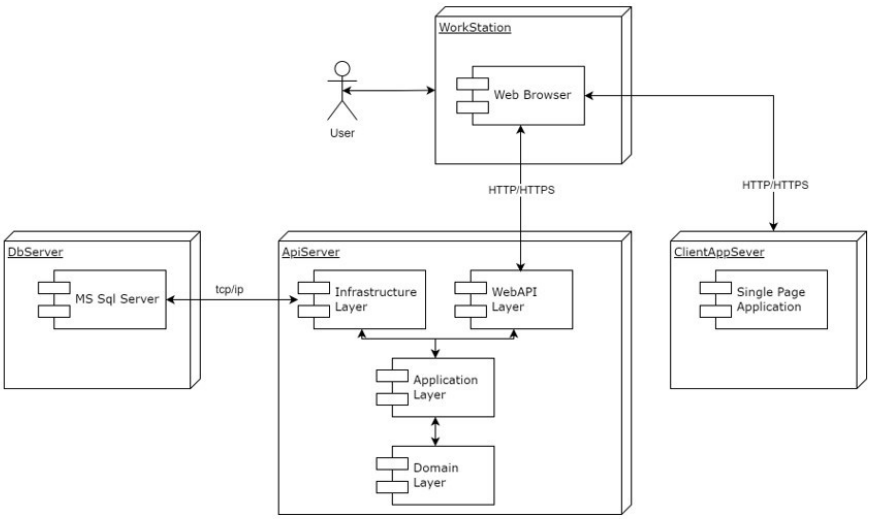
1. Доступність клієнтського додатку на різних платформах (Windows, iOS, Android, linux)
2. Відсутність обмежень за країною при підключенні.
3. Дані авторизації користувачів не повинні зберігатися в відкритому вигляді.
4. Дані не повинні зберігатися на пристроях користувачів.
5. Можливість підключатися до API з інших, в тому числі сторонніх клієнтських додатків.

ЗАСОБИ РОЗРОБКИ

УЗАГАЛЬНЕНА ДІАГРАМА ВАРІАНТІВ ВИКОРИСТАННЯ



ДІАГРАМА КОМПОНЕНТІВ ДОДАТКУ



ДІАГРАМА МОДЕЛІ ДАНИХ ГЕОГРАФІЧНИХ ОБ'ЄКТІВ

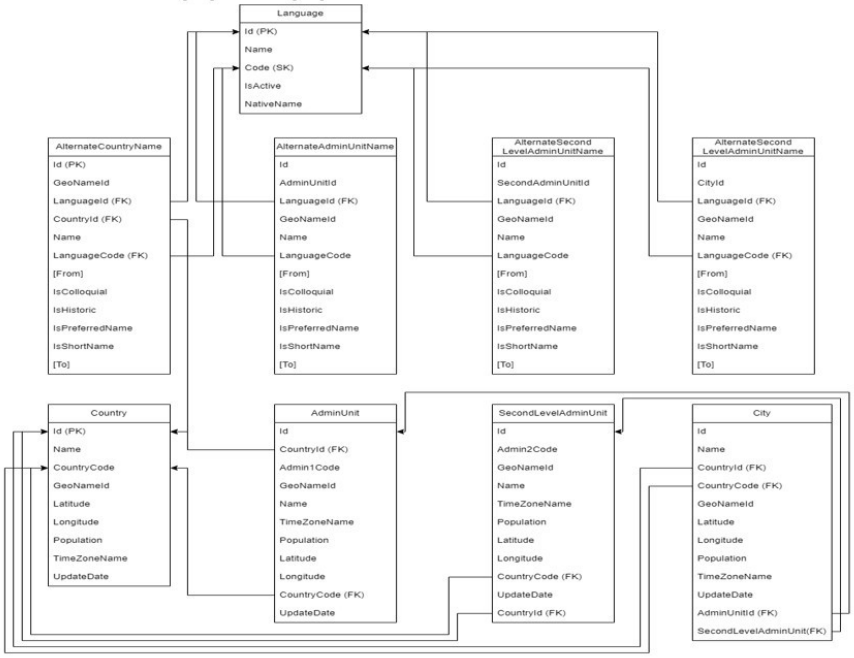
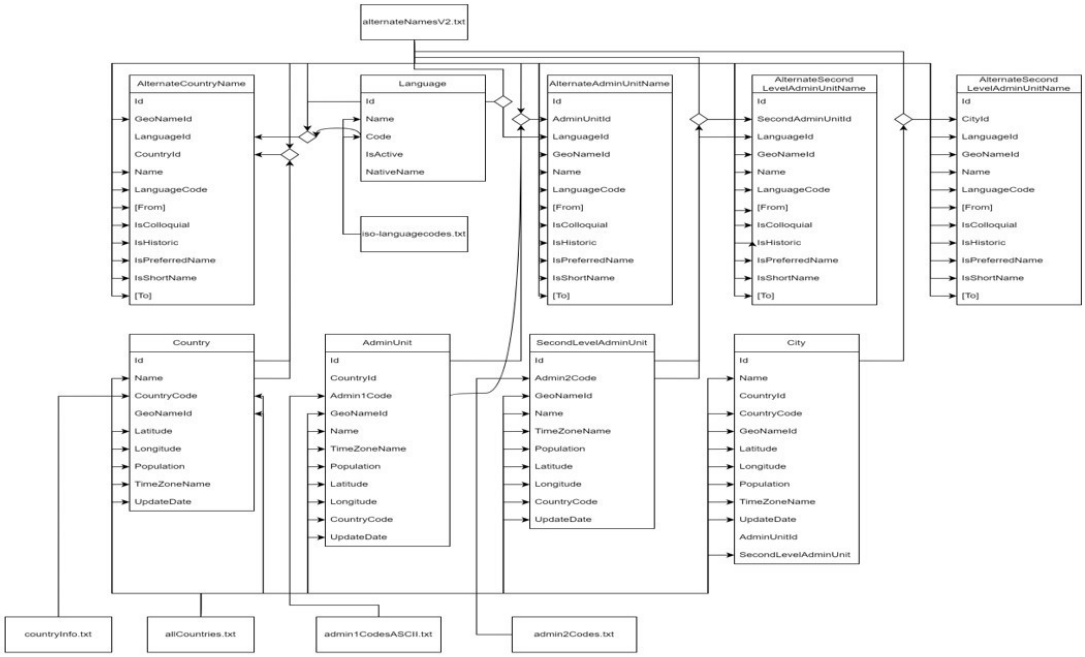
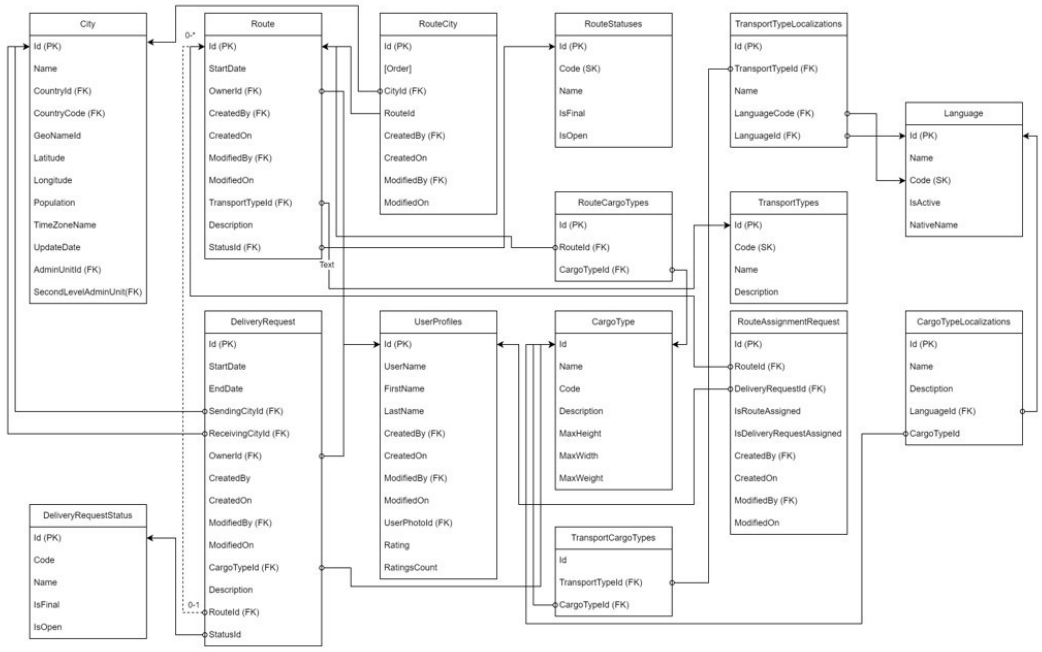


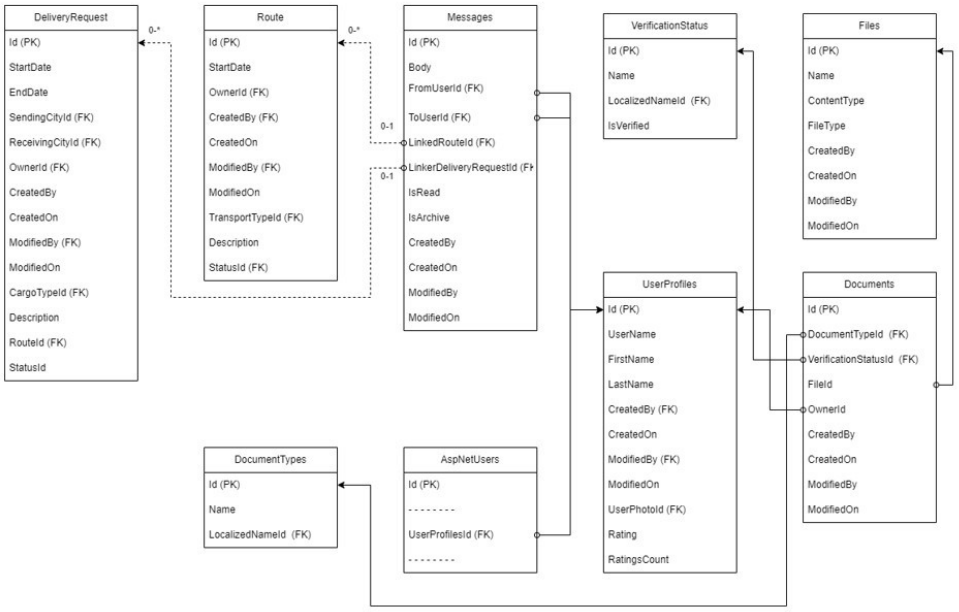
СХЕМА РОЗГОРТАННЯ ДАНИХ ДО БД З ВІДКРИТИХ РЕСУРСІВ GEONAMES



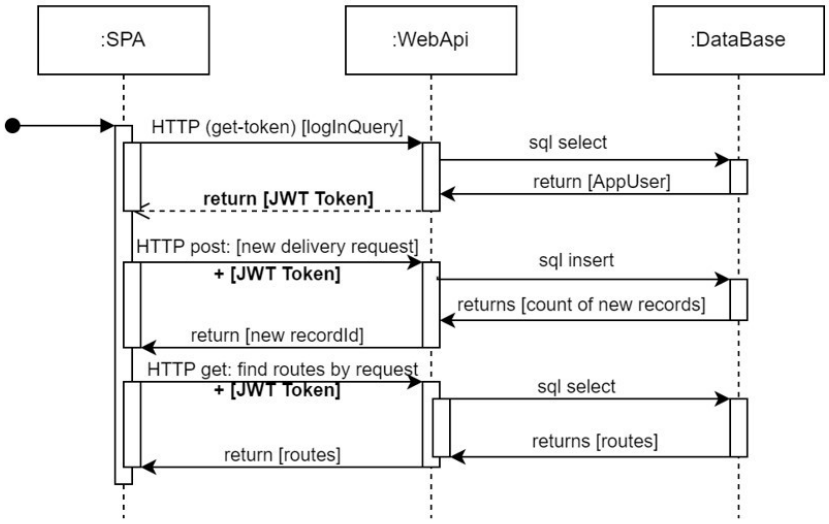
ДІАГРАМА МОДЕЛІ ДАНИХ ЗАЯВОК НА ДОСТАВКУ ТА МАРШРУТІВ



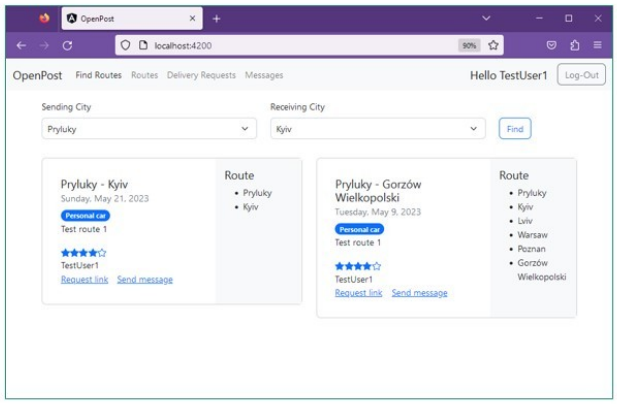
ДІАГРАМА МОДЕЛІ ДАНИХ КОРИСТУВАЧІВ



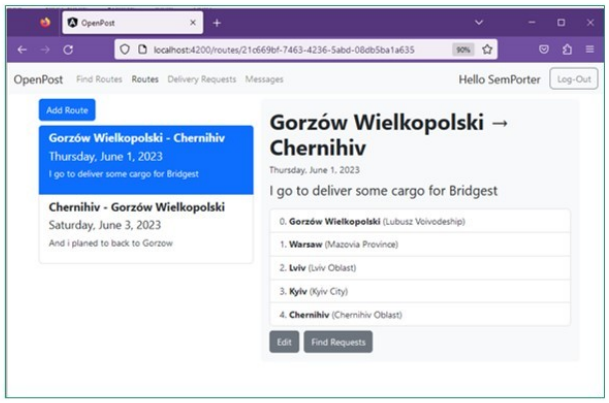
ДІАГРАМА ПОСЛІДОВНОСТІ АВТОРИЗАЦІЇ НА ОСНОВІ JWT ТОКЕНУ



ЕКРАННІ ФОРМИ

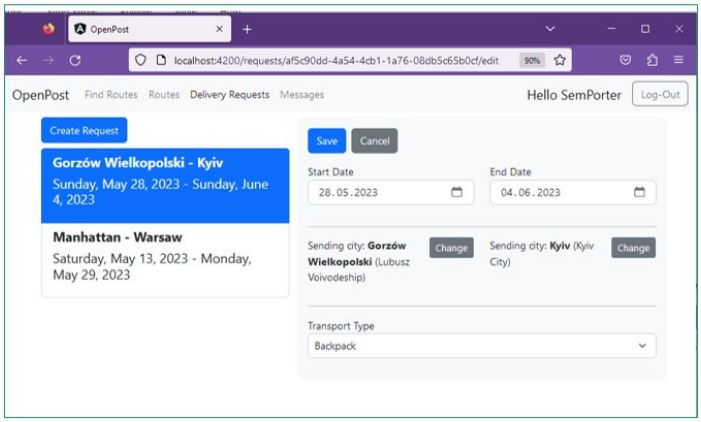


Вікно пошуку попутного транспорту



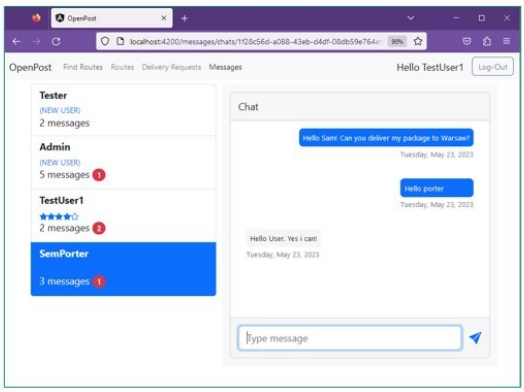
Вікно списку маршрутів користувача

ЕКРАННІ ФОРМИ

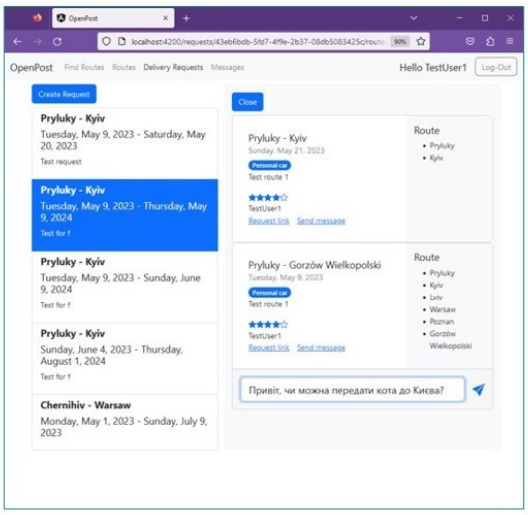


Вікно списку та вікно редагування заявок на доставку

ЕКРАННІ ФОРМИ

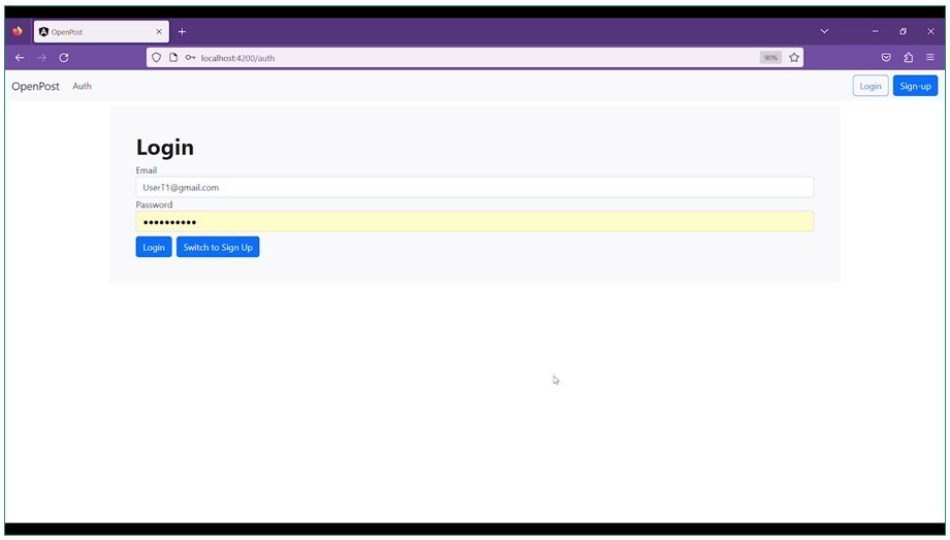


Вікно обміну повідомленнями



Пошук попутного транспорту по заявці

ПРИКЛАД РОБОТИ ПРОГРАМИ



АПРОБАЦІЯ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

- АКТУАЛЬНІСТЬ АВТОМАТИЗАЦІЇ ПРОЦЕСУ ПОШУКУ ПОПУТНОГО ТРАНСПОРТУ ДЛЯ ПЕРЕДАЧІ ПОСИЛОК. Горбань Андрій Миколайович, Державний університет телекомунікацій / Золотухіна Оксана Анатоліївна, к.т.н, доцент, Державний університет телекомунікацій - Всеукраїнська науково-технічна конференція «Застосування програмного забезпечення в інфокомунікаційних технологіях». Збірник тез. – К.: ДУТ, 2023.
- ПЕРЕВАГИ ВИКОРИСТАННЯ ПЛАТФОРМИ .NET ДЛЯ РОЗРОБКИ СЕРВІСУ ПОШУКУ ПОПУТНОГО ТРАНСПОРТУ ДЛЯ ПЕРЕДАЧІ ПОСИЛОК. Горбань Андрій Миколайович, Державний університет телекомунікацій / Золотухіна Оксана Анатоліївна, к.т.н, доцент, Державний університет телекомунікацій - Міжнародна науково-практична конференція «Сучасні аспекти діджиталізації та інформатизації в програмній та комп'ютерній інженерії»

ВИСНОВКИ

1. Досліджено процес пошуку попутного транспорту для передачі посилок. Визначено основні поняття, учасників процесу, та проблеми які вирішуються під час погуку попутного транспорту.
2. Проведено аналіз способів та програмних рішень що використовуються для вирішення задач пошуку попутного транспорту для передачі посилок. Аналіз показав наявність низького рівня автоматизації в одних рішеннях та обмежень по типам транспорту, типам посилок, мовні обмеження та територіальні обмеження в інших.
3. Сформульовано вимоги до програмного забезпечення з урахуванням недоліків та обмежень існуючих засобів.
4. Проведено аналіз технологій та засобів розробки. Обрано платформу .net та мову C# для розробки бекенд частини додатку та фреймоврк Angular з використанням мови TypeScript для клієнтського додатку.
5. Спроектовано та розроблено веб-сервіс та клієнський додаток пошуку попутного транспорту для передачі посилок. Перевагами додатку є доступність різних видів транспорту та посилок, автоматизація процесу пошуку, відсутність обмежень по географії та можливість пошуку великою кількістю мов завдяки використанню відкритої бази географічних назв GeoNames.