МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет інформаційних технологій
Кафедра інтелектуальних технологій

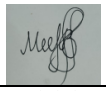
ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

НА ТЕМУ

Система моніторингу наявності пального на базі скрейпінг-технологій

Галузь знань **12 «Інформаційні технології»**
Спеціальність **122 «Комп'ютерні науки»**
Освітня програма «**Комп'ютерні науки»**
Освітній рівень: бакалавр

Виконав: студент 4 курсу, групи КН- 42

Волошина М.О. 
(прізвище та ініціали)

Керівник *Доманецька І.М.* 
(прізвище та ініціали)
к.т.н., доцент
(науковий ступінь, звання)

Випускна кваліфікаційна робота бакалавра допущена до захисту
рішенням кафедри *інтелектуальних технологій*
Протокол №___від_____р.
зав. кафедри_____ доц. Іларіонов О.Є.

Київ – 2023

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ТАРАСА ШЕВЧЕНКА

Факультет інформаційних технологій

Кафедра інтелектуальних технологій

Спеціальність 122 «Комп'ютерні науки»

ЗАТВЕРДЖУЮ
Завідувач кафедри
інтелектуальних технологій
Іларіонов О.Є.

_____

"___"_____2023 р.

ЗАВДАННЯ

НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

_____Волошиній Марині Олегівні

_____

(прізвище, ім'я, по батькові)

1. Тема проекту (роботи)
   «Система моніторингу наявності пального на базі скрейпінг-технологій»
затверджена протоколом засідання кафедри від « 23 » грудня 2022 р. №4
2. Термін здачі студентом закінченого проекту (роботи) 29 травня 2023 року
3. Вихідні дані до проекту (роботи)
   Інформація про доступне пальне в Україні
4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити)
   1. Аналіз особливостей задачі збору інформації про пальне в Україні. 2. Проектування архітектури застосунку. 3. Програмна реалізація застосунку.
5. Перелік презентаційного матеріалу (з точним зазначенням обов'язкових презентацій)
   1. Об'єкт та предмет дослідження, мета роботи (1 слайд)
   2. Актуальність розробки застосунку та аналіз наявних систем (3 слайди)
   3. Функціональний аналіз та визначення основних вимог до застосунку (2 слайди)
   4. Проектні рішення та архітектура застосунку (3 слайди)
   5. Програмна реалізація (3 слайди)
   6. Висновки (1 слайд)

6. Консультанти з випускної кваліфікаційної роботи із зазначенням розділів випускної кваліфікаційної роботи, що їх стосуються

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання 15 лютого 2023 року

Керівник _____ / Доманецька І. М. /
(підпис) (ПІБ)

Завдання прийняв до виконання _____ / Волошина М. О. /
(підпис) (ПІБ)

КАЛЕНДАРНИЙ ПЛАН

Пор. №	Назва етапів випускної кваліфікаційної роботи	Термін виконання етапів випускної кваліфікаційної роботи	Примітка
1	Опрацювання літератури	15.02 – 27.02	
2	Робота над розділом 1. Аналіз особливостей задачі збору інформації про пальне в Україні	27.02 – 08.03	
3	Робота над розділом 2. Проектування архітектури застосунку	08.03 – 08.04	
4	Робота над розділом 3. Програмна реалізація застосунку	08.04 – 13.05	
5	Робота над оформленням пояснювальної записки	13.05 – 25.05	
6	Робота над презентацією	25.05 – 29.05	

Студент-дипломник _____ / Волошина М. О. /
(підпис) (ПІБ)

Керівник випускної кваліфікаційної роботи _____ / Доманецька І. М. /

АНОТАЦІЯ

Дипломна робота викладена на 79 сторінках, містить 3 розділи, 60 ілюстрацій, 1 таблицю, 14 джерел в переліку посилань.

Об’єктом дослідження роботи є процес збору інформації про наявність пального, його кількість, тип, вартість по всіх містах та найбільших мережах АЗС України.

Предметом дослідження роботи є підхід до видобутку даних про пальне на базі використання технології скрейпінгу.

Метою роботи є розробка програмного застосунку для збору та узагальнення інформації щодо наявності пального по найбільшим мережам АЗС України задля підвищення зручності та спрощення доступу до інформації про наявне пальне.

У першому розділі проведено аналіз особливостей задачі збору інформації про пальне в Україні, описано технологію скрейпінгу, проаналізовано існуючі рішення на ринку. У другому розділі спроектовано функціональну структуру системи, надано проектні рішення щодо схем функціонування застосунку, розроблено архітектуру застосунку, а також універсальну інформаційну модель для збору інформації про пальне. У третьому розділі обґрунтувано вибір інструментальних засобів для програмної реалізації застосунку, описано структуру класів системи, протестовано основні функції готового застосунку та наведено контрольний приклад роботи застосунку.

Ключові слова: скрейпінг, адаптер, мапер, web-застосунок, Spring, Java.

ANNOTATION

The thesis is laid out on 79 pages, contains 3 chapters, 60 illustrations, 1 table, 14 sources in the list of references.

The object of the work research is the process of collecting information about the availability of fuel, its quantity, type, cost in all cities and the largest networks of gas stations in Ukraine.

The subject of the research work is an approach to fuel data extraction based on the use of scraping technology.

The purpose of the work is to develop a software application for collecting and summarizing information on the availability of fuel at the largest networks of gas stations in Ukraine in order to increase convenience and simplify access to information about available fuel.

In the first chapter, an analysis of the specifics of the task of collecting fuel information in Ukraine was carried out, scraping technology was described, and existing solutions on the market were analyzed. In the second section, the functional structure of the system is designed, design solutions are provided for the application's functioning schemes, the application's architecture is developed, as well as a universal information model for collecting fuel information. The third section substantiates the choice of tools for the software implementation of the application, describes the structure of the system classes, tests the main functions of the ready-made application, and provides a control example of the application's operation.

Keywords: scraping, adapter, mapper, web application, Spring, Java.

ЗМІСТ

СПИСОК ТЕРМІНІВ, СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ

ІС – інформаційна система.

АЗС – автозаправна станція.

СТО – станція технічного обслуговування.

ДП – дизельне паливо.

TIR – міжнародні дорожні перевезення.

API – прикладний програмний інтерфейс.

MVC – модель-відображення-контролер, архітектурний шаблон, який використовується під час проектування та розробки програмного забезпечення.

БД – база даних.

СУБД – система управління (керування) базами даних.

ВСТУП

На сьогоднішній день події в Україні стрімко змінюють умови життя людей та змушують шукати способи адаптації до непередбачуваних ситуацій. Сфера інформаційних технологій має свою значну частку у задоволенні потреб населення та продовжує норощувати свій вплив на сьогодення будь-якої людини.

Після початку повномасштабної війни на АЗС усієї країни утворились кілометрові черги, ситуація також погіршилась у травні 2022 року після ракетного удару по нафтопереробній галузі. Багато людей потерпали від неможливості знайти пальне, годинами стояли у чергах та витрачали купу часу на пошук інформації про наявність пального у мережі інтернет.

Наразі ситуація з пальним покращилась, але війна триває і спрогнозувати який стан справ буде далі майже неможливо.

Метою дипломної роботи є розробка програмного застосунку для збору та узагальнення інформації щодо наявності пального по найбільшим мережам АЗС України задля підвищення зручності та спрощення доступу до інформації про наявне пальне.

Об’єктом дослідження є процес збору інформації про наявність пального, його кількість, тип, вартість по всіх містах та найбільших мережах АЗС України.

Предметом дослідження є підхід до видобутку даних про пальне на базі використання технології скрейпінгу.

Завданням дослідження є:

- Узагальнення даних, що є доступними по найбільшим автозаправним компаніям України.
- Побудова архітектури та структуризації даних на базі скрейпінгу.
- Розробка гнучкої моделі для узагальнення інформації по різним мережам АЗС, що дасть змогу масштабувати, змінювати, додавати нові модулі та АЗС до застосунку.

- Перевірка роботоспроможності, актуальності та доцільності реалізованого підходу.

РОЗДІЛ 1. АНАЛІЗ ОСОБЛИВОСТЕЙ ЗАДАЧІ ЗБОРУ ІНФОРМАЦІЇ ПРО ПАЛЬНЕ В УКРАЇНІ

1.1 Аналіз особливостей роботи паливного бізнесу в Україні

Предметна область, що розглядається - доступне пальне в Україні. Наразі на ринку нашої країни досить багато паливних компаній, майже кожна з яких має власні ресурси для інформування споживачів про наявність товару, сам товар, послуги. Це можуть бути сайти, застосунки, телеграм-боти тощо. Загалом, інформації в мережі досить багато, але вона різнорідна та неструктурована. Наприклад, на сайті одних компаній ми можемо бачити карту АЗС з переліком доступного пального, на інших таблицю з відмітками, десь ціна на різні види пального вказана, а десь відсутня, в одних додана інформація про способи оплати, у інших її немає. Це все робить пошук пального в умовах дефіциту досить складним та довгим процесом.

До того ж, споживачі мають потребу у наочному порівнянні послуг та продуктів, які надаються паливними компаніями. Пошук конкретного специфічного типу пального, найдешевшої позиції заданого типу палива, пального у конкретному регіоні, конкретного сервісу, що надається паливною компанією тощо - це все інформація, яка цікавить покупців і є у відкритому доступі, але для її отримання необхідно проаналізувати досить масивний об’єм даних.

Окрім цього, користувачі зацікавлені не лише в інформації щодо пального, але і зручності та функціональності ресурсів, які вони використовують для його пошуку. Наприклад, у наявності функцій прокладання маршруту до вибраної АЗС, доступу до історії пошуку АЗС, визначення чи є черги на тих чи інших заправках. Усе це економить час користувача та дозволяє робити все і одразу в одному застосунку і не вдаватись до додаткових пошуків. Але наявні застосунки паливних компаній не задовольняють або лише частково задовольняють ці потреби користувачів.

Звідси випливає наступна задача - розробка застосунку, що збирає інформацію про доступне паливо по різних мережах АЗС України та приводить її до структурованої моделі, а також відображає користувачеві у простому для розуміння, аналізу і порівняння вигляді, дає змогу фільтрувати інформацію за базовими параметрами.

1.2 Аналіз і структурування інформації, що подається найбільшими АЗС України

Майже кожна українська паливна компанія має власні онлайн-ресурси для інформування клієнтів про послуги, товари, сервіси і т.д. Серед найпопулярніших - сайти та мобільні застосунки. Для того, аби з'ясувати, яка інформація та функціонал є важливими для потенційних клієнтів, щоб обрати, де заправитись були проаналізовані ресурси найпопулярніших АЗС України. В процесі аналізу були виділені основні дані, що цікавлять користувача, корисні функції, що полегшують пошук пального та графічні інтерфейси кожного з онлайн-ресурсів.

В рамках проектування дипломної роботи розглядаються наступні АЗС:

1. WOG
2. UPG
3. SOCAR
4. БРСМ-Нафта

WOG

Одна з найбільших мереж АЗК в Україні. WOG представлена 13 нафтобазами та понад 400 АЗК на всій території підконтрольній Україні. Має також власні лабораторії, де здійснюється контроль якості пального, а також сервісну компанію, що займається техобслуговуванням АЗК.

Види палива, що реалізуються АЗС WOG:

1. 95 Євро-5
2. 92 Євро-5
3. Mustang 95

4. Mustang 100
5. ДП Євро-5
6. ДП Mustang+
7. Газ нафтовий скраплений

Серед доступних функцій сайту та застосунку мережі WOG:

1. Перегляд усіх доступних АЗС на мапі України
2. Перегляд усіх доступних АЗС у вигляді списку
3. Фільтрація АЗС по наявності певного типу палива чи сервісу
4. Перегляд детальної інформації по АЗС включаючи адресу, доступні типи палива, доступні сервіси, графік роботи та доступні типи оплати
5. Побудова маршруту до обраної АЗС
6. Перегляд цін на паливо в інтернет-магазині WOG

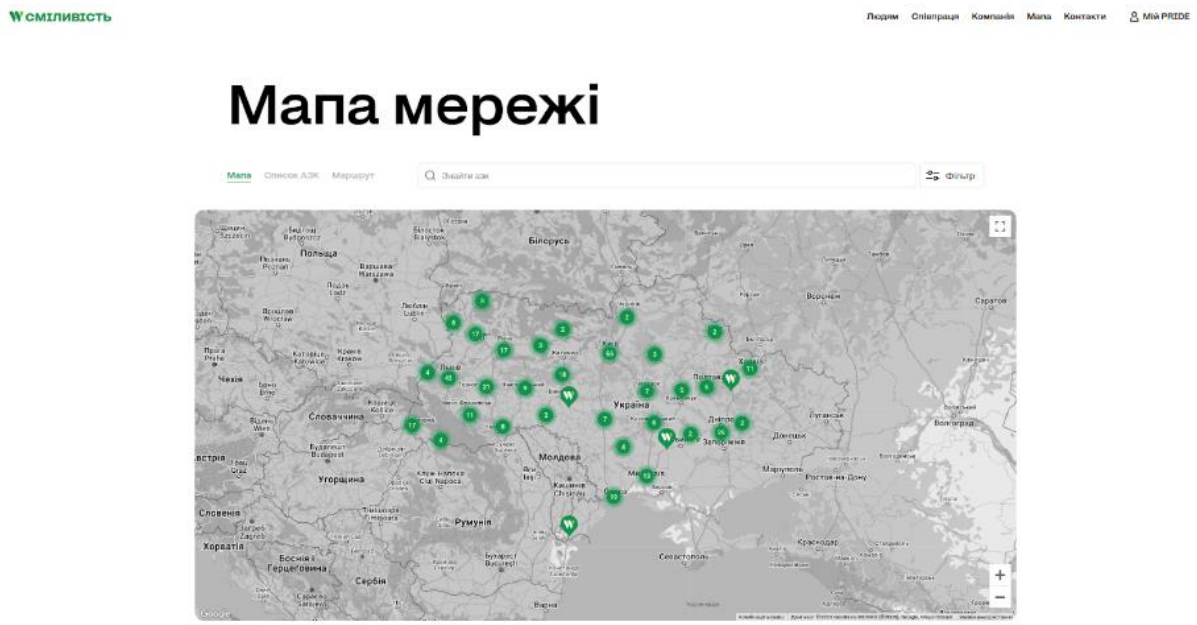


Рис. 1.1. Мапа мережі АЗС WOG

UPG

Українська мережа АЗС, що працює на ринку паливно-мастильних матеріалів України з 2003 року. До мережі UPG входить 73 автозаправки у 17 областях - за цим показником компанія у десятці найбільших операторів українського паливного ринку.

Види палива, що реалізуються АЗС UPG:

1. А-95
2. ГАЗ
3. EURO DIESEL
4. Arctic diesel
5. upg100
6. upg95

Серед доступних функцій сайту та застосунку мережі UPG:

1. Перегляд усіх доступних АЗС на мапі України
2. Фільтрація АЗС по області розташування
3. Перегляд детальної інформації по АЗС включаючи адресу, доступні типи палива та доступні сервіси
4. Перегляд загальноукраїнських цін на паливо

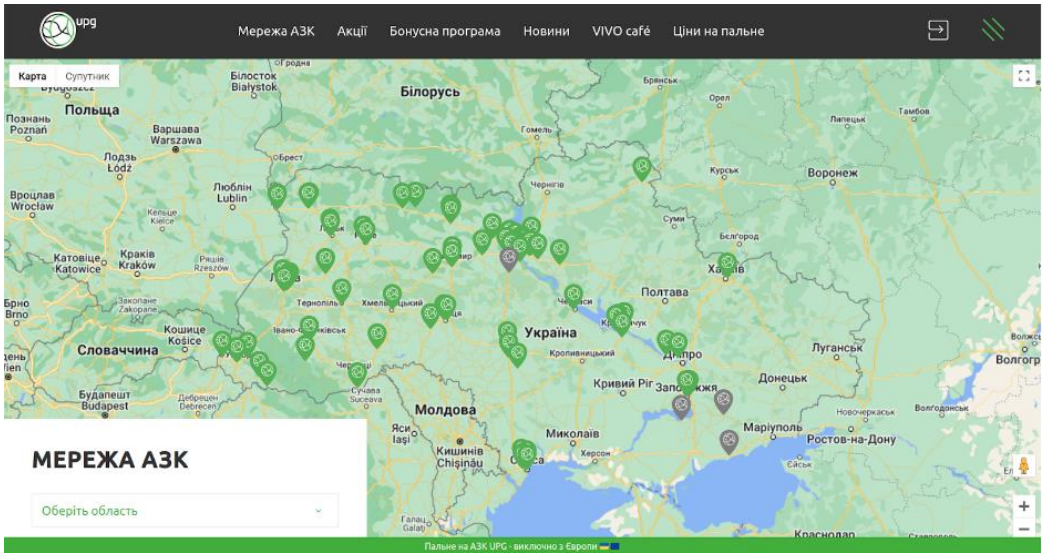


Рис. 1.2. Мапа мережі АЗС UPG

SOCAR

Дочірня компанія Державної нафтової компанії Азербайджанської республіки. В Україні SOCAR почав свою діяльність в 2008 році. У 2010 році компанія приступила до розвитку власної мережі автозаправних комплексів преміум-класу. Мережа SOCAR Energy Ukraine налічує 57 заправних станцій.

Види палива, що реалізуються АЗС SOCAR:

1. Diesel Nano Extro
2. NANO ДП
3. A 95
4. NANO 95
5. LPG
6. A 92
7. NANO 98
8. AdBlue

Серед доступних функцій сайту та застосунку мережі SOCAR:

1. Перегляд усіх доступних АЗС на мапі України
2. Перегляд усіх доступних АЗС у вигляді списку
3. Фільтрація АЗС по назві регіону, наявності певного типу палива чи сервісу
4. Перегляд детальної інформації по АЗС включаючи адресу, доступні типи палива, ціни, доступні сервіси, координати, графік роботи АЗС.

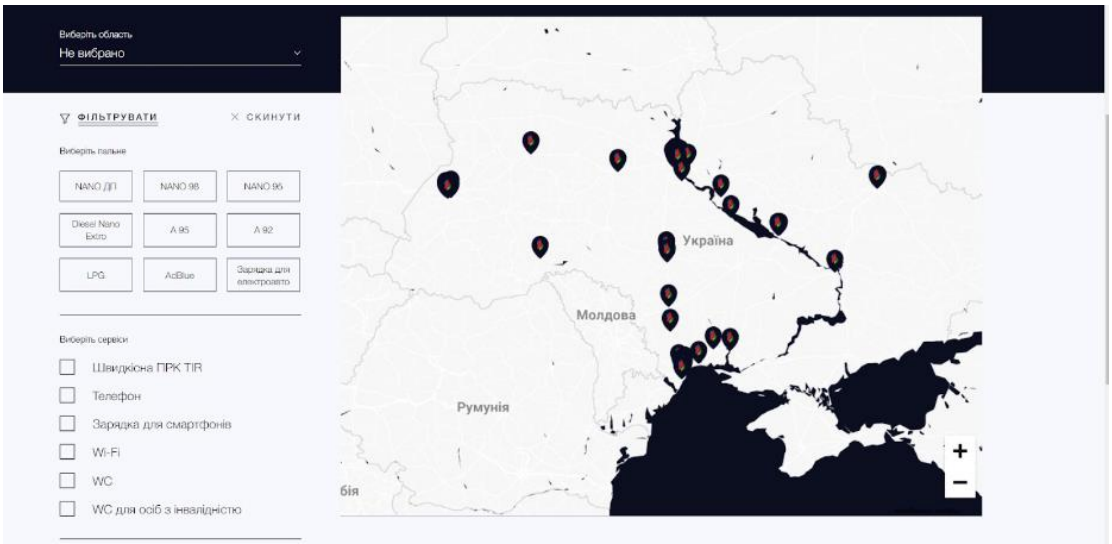


Рис. 1.3. Мапа мережі АЗС SOCAR

БРСМ-Нафта

Торгова марка мережі автозаправних комплексів під управлінням керівної компанії - ТОВ «ЄВРО СМАРТ ПАУЕР». Розвивається з 1992 року. Станом на 2022 рік мережа налічує 217 АЗК. Мережа автозаправних комплексів «БРСМ-

Нафта» входить в трійку найбільших імпортерів зрідженого автомобільного газу в Україну.

Види палива, що реалізуються АЗС БРСМ-Нафта:

1. Газ+ PLUS
2. А95 PREMIUM PLUS
3. А95 EURO PLUS
4. ДП EURO
5. ДП EURO PLUS
6. А92 EURO
7. А95 E PREMIUM+
8. Метан

Серед доступних функцій сайту та застосунку мережі БРСМ-Нафта:

1. Перегляд усіх доступних АЗС на мапі України
2. Фільтрація АЗС по назві регіону, наявності певного типу палива чи сервісу
3. Перегляд детальної інформації по АЗС включаючи адресу, доступні типи палива, доступні сервіси, координати, ім’я менеджера, телефон АЗС.
4. Побудова маршруту до обраної АЗС
5. Перегляд загальноукраїнських цін на паливо

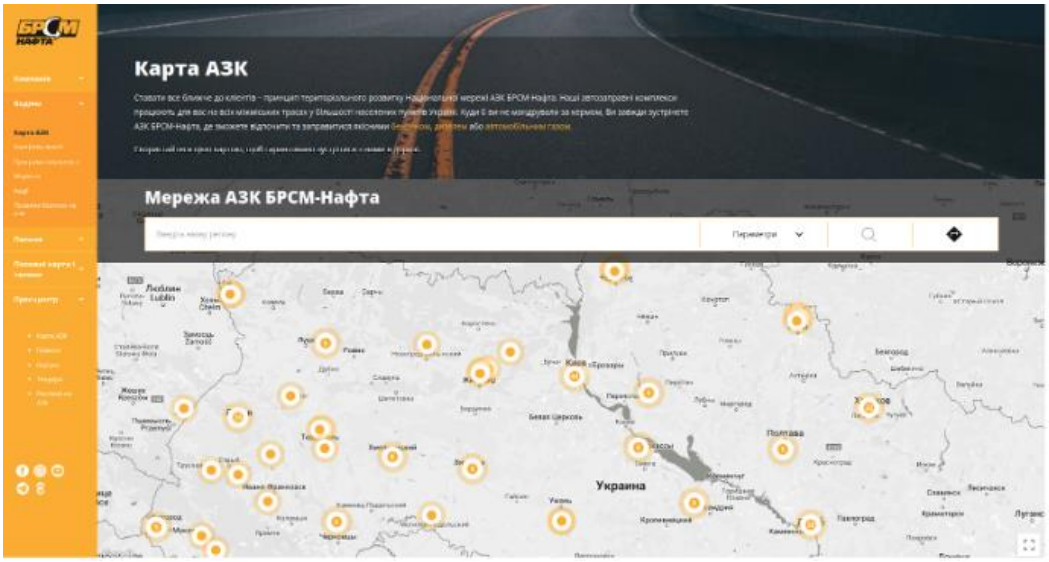


Рис. 1.4. Мапа мережі АЗС БРСМ-Нафта

Загалом сайти і застосунки усіх розглянутих мереж мають схоже наповнення та базовий функціонал, що робить можливим приведення усієї корисної користувачу інформації до базової моделі. Наприклад, можемо підсумувати, що види палива, що реалізуються представленими мережами АЗС можна привести до 4 основних типів:

1. А-92
2. А-95
3. Дизель
4. Газ

Сервіси, що надаються можна привести до 4 основних типів:

1. Їжа
2. Продаж товарів
3. Автообслуговування
4. Інше

Функції, що доступні користувачам можна розбити на 4 основні:

1. Відображення карти з АЗС
2. Побудова маршруту до АЗС
3. Фільтрація АЗС за заданими параметрами
4. Відображення детальної інформації по кожній АЗС

В рамках дипломного проекту будуть використані приведені вище узагальнення для того аби користувач мав змогу бачити корисну для себе інформацію в одному місці та обирати АЗС по тим параметрам, що дійсно можуть бути для нього цікавими та важливими, не вдаючись до додаткових пошуків та уточнень на інших інтернет-ресурсах.

1.3 Порівняльний аналіз вже існуючих програмних систем, які призначені для розв’язку задачі, що вирішується в рамках дипломної роботи

На сьогодні технологія скрейпінгу набула досить широкого розповсюдження. Її реалізації присутні у різних доменах. Таких як порівняння

цін на товари, продаж та оренда нерухомості, бізнес-аналіз ринку тощо. Після загострення проблем із дефіцитом пального в Україні скрейпінг став хорошим програмним рішенням для реалізації застосунків для аналізу інформації про пальне у веб-середовищі. Наразі на ринку є декілька телеграм-ботів, що призначені для розв'язку тієї ж задачі, що вирішується в рамках дипломної роботи.

Серед них:

1. телеграм-бот "Kyiv_AZS" - Моніторить інформацію про 95 пальне по заправкам Avias (ANP), WOG, OKKO, UPG, Socar в місті Київ. Працює по принципу повідомлень, тобто якщо на заправці з'явився 95 бензин, то бот публікує сповіщення про це з адресою заправки та її точкою на карті [9].

   Приклад сповіщення:

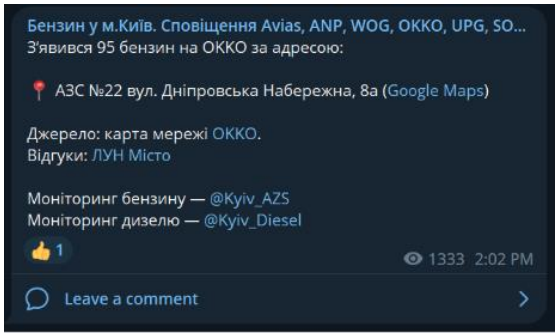


Рис. 1.5. Сповіщення про появу пального у телеграм-боті "Kyiv_AZS"

Перевірка даних відбувається через кожну годину, повідомлення приходить якщо з моменту попередньої перевірки змінились дані про наявність бензину.

Приклад точки на карті:

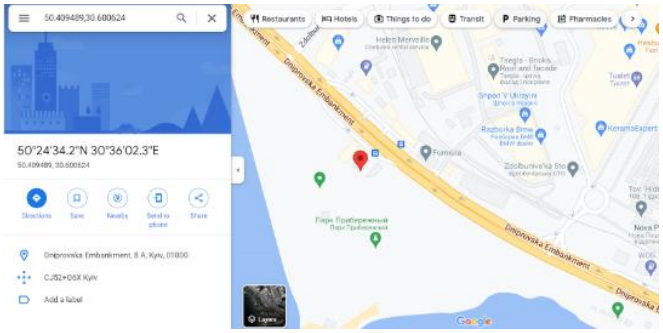


Рис. 1.6. Точка АЗС на карті в телеграм-боті "Kyiv_AZS"

2. Телеграм-бот “Kyiv_Diesel” - Моніторить інформацію про дизельне пальне по заправкам Avias (ANP), WOG, OKKO, UPG, Socar в місті Київ. Працює по принципу повідомлень, тобто якщо на заправці з’явився дизель, то бот публікує сповіщення про це з адресою заправки та її точкою на карті [7]. Приклад сповіщення:

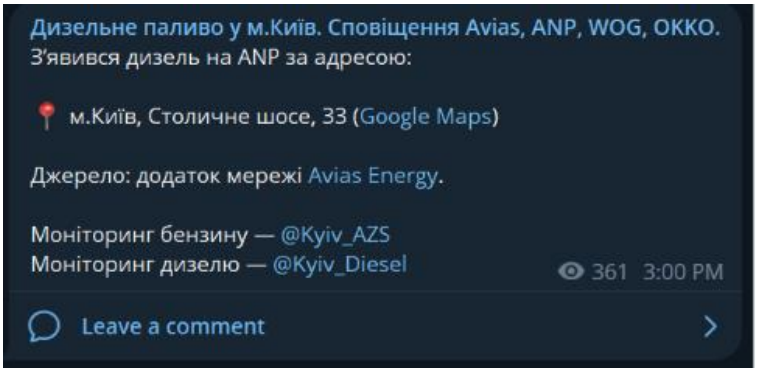


Рис. 1.7. Сповіщення про появу пального у телеграм-боті “Kyiv_Diesel”

Аналогічно як і в боті “Kyiv_AZS” перевірка даних відбувається через кожну годину, повідомлення приходить якщо з моменту попередньої перевірки змінились дані про наявність бензину.

Приклад точки на карті:

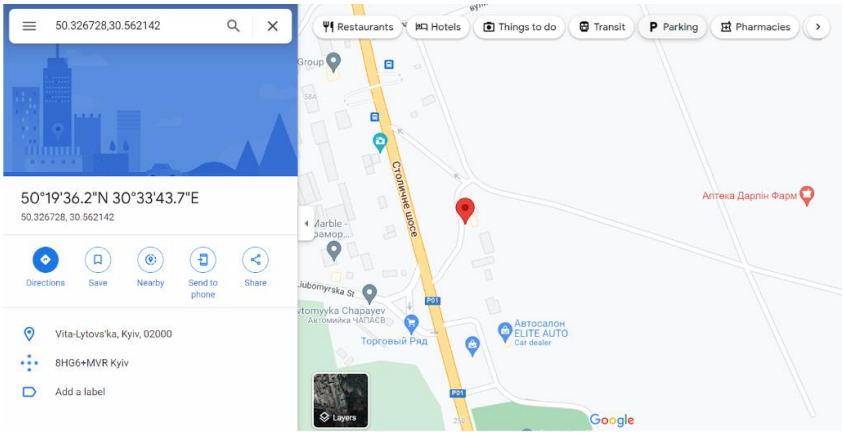


Рис. 1.8. Точка АЗС на карті в телеграм-боті “Kyiv_Diesel”

3. Телеграм-бот “petrol_alert_bot” - Моніторить інформацію за основними типами пального по всій Україні. Бот дає можливість обрати заправки

поруч або ті які цікавлять і по запиту виводить у повідомлення інформацію про доступне паливо [8].

Бот пропонує налаштувати індивідуальні параметри профілю: обрати тип пального, який цікавить користувача, а також область, район і місто по яким буде здійснюватись пошук.

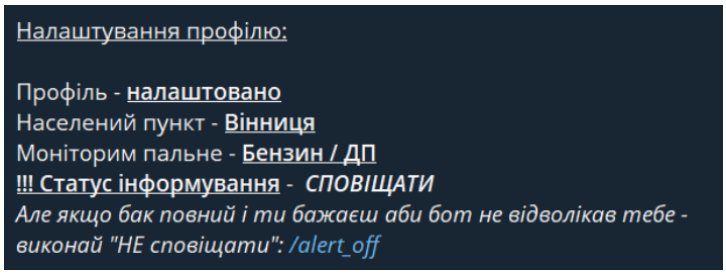


Рис. 1.9. Панель налаштування профілю в телеграм-боті “petrol_alert_bot”

Основні запити, які виконує бот:

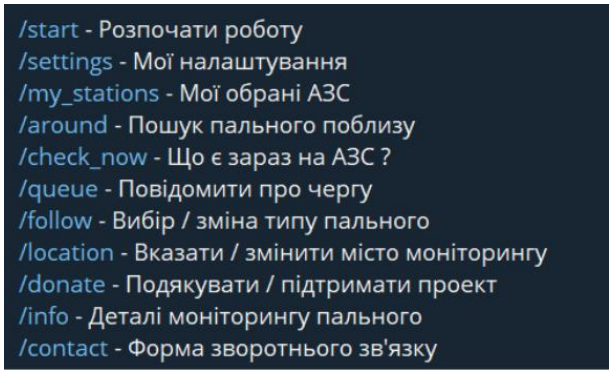


Рис. 1.10. Меню в телеграм-боті “petrol_alert_bot”

Приклад опису яка надається по станції:

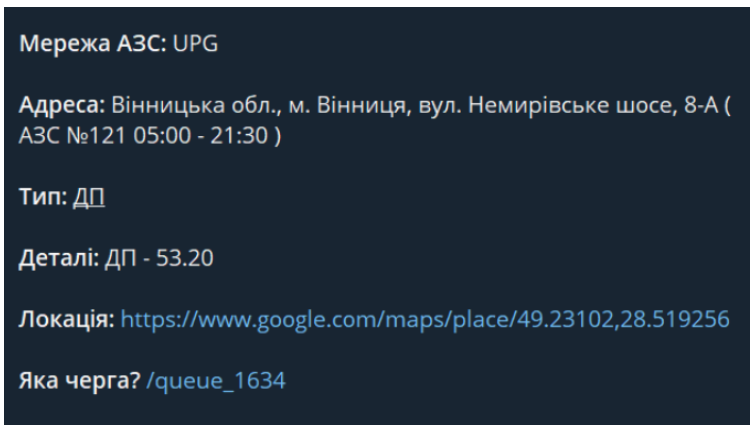


Рис. 1.11. Опис АЗС в телеграм-боті “petrol_alert_bot”

Для наочного порівняння вже реалізованих програмних рішень (Бот “Kyiv_AZS”, Бот “Kyiv_Diesel”, Бот “petrol_alert_bot”) та того, що описується в даній роботі (“Fuel_App”) було розроблено порівняльну таблицю, яка оцінює системи за наступними функціональними критеріями:

- Локація - яку територію збору інформації покриває застосунок.
- Компанії - по яким компаніям підтримується пошук інформації.
- Тип пального - по яким типам пального підтримується пошук інформації.
- Графік роботи - чи надається інформація про графік роботи АЗС.
- Ціни - чи надається інформація про ціни на конкретне пальне.
- Можливість фільтрації АЗС по місту - чи можливий користувацький пошук по містах.
- Можливість фільтрації АЗС по типу палива - чи можливий користувацький пошук по типу палива.
- Можливість фільтрації АЗС по ціні.
- Можливість фільтрації АЗС по компанії - чи можливий користувацький пошук по компанії.
- Інформація про черги на АЗС - чи надається інформація про чергу на конкретній АЗС.
- Контакти та додаткова інформація - чи надається додаткова інформація (номер гарячої лінії, посилання на сайт, посилання на додаток).
- Координати АЗС - чи надається точка на карті з точними координатами АЗС.
- Побудова маршруту до АЗС.

Таблиця 1.1. Порівняльна таблиця функціоналу різних програмних рішень

Назва	Бот Kyiv_AZS	Бот Kyiv_Diesel	Бот petrol_alert_bot	Fuel_App
Локації	Київ	Київ	Вся Україна	Вся Україна

Команії	Avias, WOG, OKKO, UPG, Socar	Avias, WOG, OKKO, UPG, Socar	WOG, UPG, OKKO, ANP	WOG, UPG, OKKO, Socar
Тип пального	A95	ДП	A95, ДП, Газ	A92, A95, ДП, Газ
Графік роботи	-	-	+	+
Ціни	-	-	-	+
Можливість фільтрації АЗС по місту	-	-	+	+
Можливість фільтрації АЗС по типу палива	-	-	+	+
Можливість фільтрації АЗС по ціні	-	-	-	+
Можливість фільтрації АЗС по компанії	-	-	-	+
Інформація про черги на АЗС	-	-	+	-
Контакти та додаткова інформація	+	+	+	+
Координати АЗС	+	+	+	+
Побудова маршруту до АЗС	-	-	-	+

Підбиваючи підсумки порівняння існуючих програмних рішень, можна зробити висновок, що наразі ринок програмних рішень у цьому домені не задовольняє повністю потреби користувачів. Телеграм-боти, що представленні в

аналізі не мають повної інтеграції АЗС з картами та відповідно функціоналу, який вони надають, не дають можливості обирати АЗС за ціною палива та збирають інформацію по невеликій кількості автозаправних станцій. Деякі надають інформацію лише в рамках міста або певного типу палива. Згідно наведених порівнянь, можна зробити висновок, що найфункціональнішим та найпристосованішим до сьогоднішнього ринку застосунків-скрейперів палива в Україні є телеграм-бот petrol_alert_bot. Він підтримує досить велику кількість фільтрів, видів палив та мереж АЗС, по яким шукається інформація, проте на відміну від розроблюваного в рамках дипломного проекту застосунку, він не дає можливості будувати маршрут до обраної АЗС, не має фільтрації по ціні та доступним сервісам, не має зручного інтерфейсу та доступний лише користувачам Telegram, що робить Fuel_App унікальним продуктом, що має найширший перелік застосувань та розрахований на широку аудиторію.

1.4 Технологія скрейпінгу

Оскільки для вирішення задачі дипломного проекту була обрана технологія скрейпінгу, необхідно розкрити її сутність та доцільність використання.

Скрейпінг - це процес збору конкретних загальнодоступних даних з багатьох різних веб-сайтів і складання їх в єдиний формат, щоб їх можна було оцінювати або використовувати різними способами [12].

Як правило, виконується за допомогою комп'ютерних програм, що імітують поведінку людини в інтернеті, або з'єднуються з веб сервером напряму по протоколу HTTP, або управляють повноцінним веб браузером.

Застосування:

- Порівняння та моніторинг цін
- Ціноутворення
- Збагачення технології машинного навчання
- Агрегація фінансових даних

- Моніторинг настроїв споживачів
- Відстеження новин.
- Аналіз даних
- Наукові дослідження

Як працює

Збір даних у скрейпінгу автоматизований, і для кожного завдання потрібен бот або програма з певними налаштуваннями. Вона називається скрейпер. Спочатку користувач визначає набір необхідних даних, список інтернет-ресурсів для роботи скрейпера, особливості отримання інформації [1].

Потрібні дані з ресурсу, який обробляється, можуть бути в:

- API веб-сервісу.
- Вихідному HTML-коді.
- Всередині файлу, куди веде посилання з ресурсу (наприклад, у javascript-файлі).
- Відповіді на запит мережі на сервер

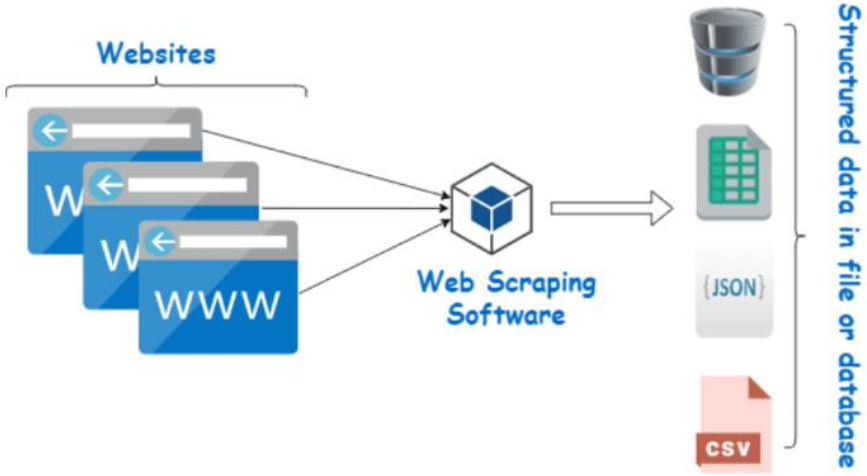


Рис. 1.12. Класична модель роботи скрейпера

Найчастіше для реалізації цієї технології використовуються вже готові програмні рішення: ScrapingBot, Scraper API, Xtract.io, Octoparse, headless-браузери Puppeteer та Playwright. Вони здатні отримувати потрібний вміст HTML, працювати з JavaScript, фільтрувати отримані відомості та виводити їх у формі готових баз даних, таблиць Excel, файлів CSV або окремих API - а також

оминати встановлені сайтами обмеження. У випадку даного застосунку реалізація скрейперу має бути кастомною та працювати на базі стандартних Java бібліотек (Springframework Web Client та Jsoup) [11].

Переваги застосування технології скрейпінгу:

1. Швидкість - на відміну від мануального збирання даних, готові програмні рішення (скрейпери) дають можливість автоматизувати такі процеси як отримання доступу до цільового ресурсу, формування датасету та його структурування, що значно економить час від збору до аналізу інформації для широкого класу задач.
2. Гнучкість - програмне забезпечення для скрейпінгу дає можливість збільшувати або зменшувати кількість операцій по збору даних по мірі необхідності, позбавляючи необхідності його підтримки чи обслуговування, налаштування конвертерів даних.
3. Багатофункціональність - наразі існує дуже багато доменів та бізнес-задач для яких можна використати технологію скрейпінгу і їх кількість тільки росте, від аналізу ринку криптовалют до пошуку найдешевших товарів в мережі - під різнорідні задачі можна писати скрейпери, що будуть виконувати роботу користувача в мережі.

Недоліки застосування технології скрейпінгу:

1. Незастосовність - не для всіх випадків скрейпінг є доцільним або навіть можливим у використанні, часом ресурси можуть бути унікальними і складність розробки алгоритму для їх структурування робить використання технології занадто трудомісткою задачею.
2. Складність - наразі досить багато ресурсів розробляється із захистом від скрейпінгу, наприклад, виявленням та забороною ботів для сканування (перегляду) своїх сторінок. Це змушує шукати обхідні шляхи для збору інформації, які роблять застосунок на базі скрейпінгу громіздким та важким для подальшої підтрики.

1.5 Постановка задачі на створення веб-застосунку для збору інформації про пальне в Україні

Завдання дипломної роботи полягає в тому, щоб реалізувати інформаційну систему пошуку та обробки інформації про доступне пальне в Україні, використовуючи технологію скрейпінгу.

Метою дослідження є розробка програмного застосунку для збору та узагальнення інформації щодо наявності пального по найбільшим мережам АЗС України задля підвищення зручності та спрощення доступу до інформації про наявне пальне.

Об’єктом дослідження є процес збору інформації про наявність пального, його кількість, тип, вартість по всіх містах та найбільших мережах АЗС України.

Предметом дослідження є підхід до видобутку даних про пальне на базі використання технології скрейпінгу.

Задачі, що вирішуються в рамках дипломного проекту:

1. З’ясування форматів даних, що надходять з ресурсів найбільших мереж АЗС України “WOG”, “UPG”, “SOCAR”, “БРСМ-Нафта”.
2. Формування інформаційної моделі, що є універсальною для даних із кожної компанії та узагальнює корисну для користувача інформацію.
3. Проектування бази даних для збереження інформації
4. Розробка серверної частини для отримання, обробки та збереження даних предметної області.
5. Розробка клієнтської частини, що реалізує зручний користувацький інтерфейс для роботи із застосунком.

1.6 Функціональний аналіз та визначення основних вимог до програмного забезпечення

Наведемо загальну картину функціонування застосунку через його представлення у вигляді “чорної скрині”. (див. рис. 1.13)

На вході система прийматиме наступні вхідні дані:

- Задані параметри фільтрації користувача (ціна пального, тип пального, сервіс, місце розташування АЗС).
- Перелік АЗС, отриманий з ресурсів кожної з мереж АЗС, що розглядаються у роботі.
- Доступні типи палива кожної з АЗС на даний момент.
- Доступні сервіси кожної з АЗС.
- Ціни на пальне по кожній АЗС.

На виході система буде подавати наступні дані:

- Відфільтровані за заданими параметрами користувача АЗС.
- Структурована інформація по кожній АЗС.
- Карта України з вказаними на ній відповідно точками розташування АЗС

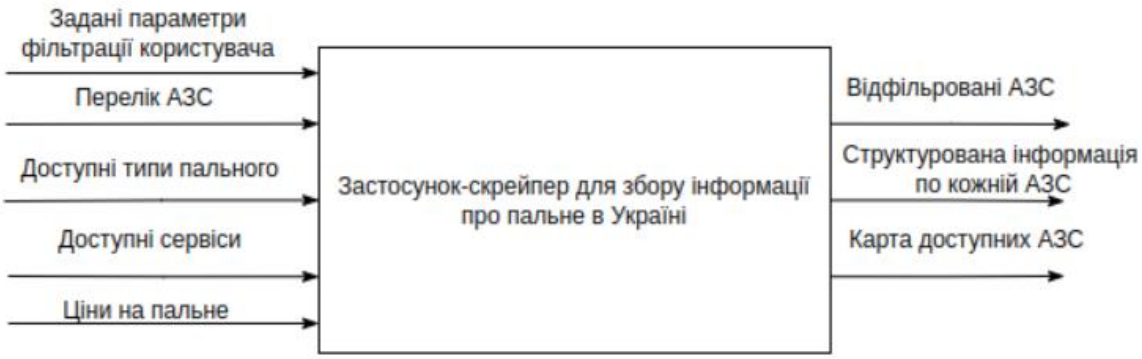


Рис. 1.13 Функціонування застосунку у вигляді “чорної скрині”

Функціональні вимоги:

- Можливість перегляду усіх доступних АЗС на карті.
- Можливість перегляду детальної інформації по кожній АЗС.

- Клієнтська частина повинна робити запити на серверну для отримання всіх необхідних даних.
- Застосунок повинен робити запити на API “WOG”, “UPG”, “SOCAR” та “БРСМ-Нафта”.
- Серверна частина повинна мапити дані отримані від веб-сторінок у єдину інформаційну модель.
- Можливість фільтрування даних по місту, типу палива, компанії та сервісу.
- Можливість сортування АЗС по ціні палива
- Можливість побудови маршруту до обраної АЗС.
- При відмові однієї зі сторінок компаній, що є джерелами інформації сервер повинен залишати доступною попередню змаплену інформацію.

Нефункціональні вимоги:

- Серверна частина повинна взаємодіяти з клієнтською шляхом REST API.
- Оновлення даних в базі даних повинно відбуватись кожні 10 хвилин.
- Завантаження всіх даних у БД повинно займати на більше 3 хвилин.
- Продуктивність (програма не повинна перевантажувати сервер).
- Зберігання даних у реляційній СУБД.
- Інтерфейс користувача повинен бути інтуїтивно простим у використанні.
- Інформація по кожній АЗС повинна бути представлена по узагальненому шаблону (містити загальні для кожної мережі АЗС дані).
- АЗС повинні бути розміщені на карті України відносно їхніх реальних координат.
- Для побудови маршруту до обраної АЗС та відображення їх на карті застосунок повинен інтегруватись з Google API.

Висновок до першого розділу

В результаті виконання першого розділу було проведено аналітичний огляд дипломної роботи. В рамках нього було розкрито тему проблем предметної області доступного пального в межах України, що породжують задачу, яка

вирішується в рамках дипломного проекту. Також було проаналізовано онлайн-ресурси 5 найбільших мереж АЗС України з метою узагальнення потреб користувачів при виборі заправної станції, пального, виокремлено базову модель інформації, що подається цими ресурсами. Окрім того, був проведений порівняльний аналіз вже існуючих програмних систем, які призначені для розв’язку задачі, що вирішується в рамках дипломної роботи та описаний алгоритм її розв’язку. До того ж, було описано технологію скрейпінгу, висвітлено її переваги та недоліки, а також був проведений функціональний аналіз застосунку, що включав створення функціональних та нефункціональних вимог до застосунку, аналіз функціональної частини застосунку у вигляді “чорної скрині”.

РОЗДІЛ 2. ПРОЕКТУВАННЯ АРХІТЕКТУРИ ЗАСТОСУНКУ

2.1. Функціональне проектування інформаційної системи моніторингу пального

Опишемо функціонування інформаційної системи пошуку моніторингу пального за допомогою структурних діаграм IDEF0. На рисунку 2.1 подана контекстна діаграма “ЯК БУДЕ”.

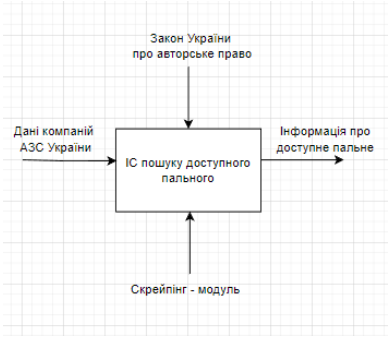


Рис. 2.1. Контекстна діаграма “ЯК БУДЕ”

Декомпозуємо бізнес-процеси з контекстної діаграми “ЯК БУДЕ” (Рис. 2.1). Після отримання сирих даних з ресурсів компаній АЗС система переходить до процесу завантаження цієї інформації. Далі система переходить до обробки і перетворення інформації, а далі - до фільтрації структурованих даних. В результаті, користувач отримує відсортовані дані за заданими параметрами.

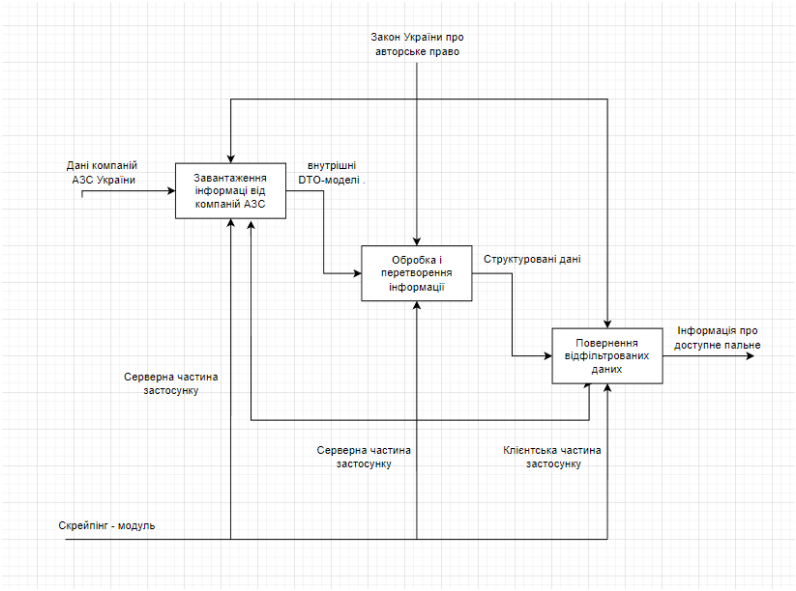


Рис. 2.2. Декомпозиція основних функцій діаграми ЯК БУДЕ

2.2 Проектні рішення щодо схем функціонування застосунку

Для кращого розуміння загального представлення функціональної складової системи, яку ми намагаємося реалізувати, побудуємо діаграму варіантів використання даної програми.

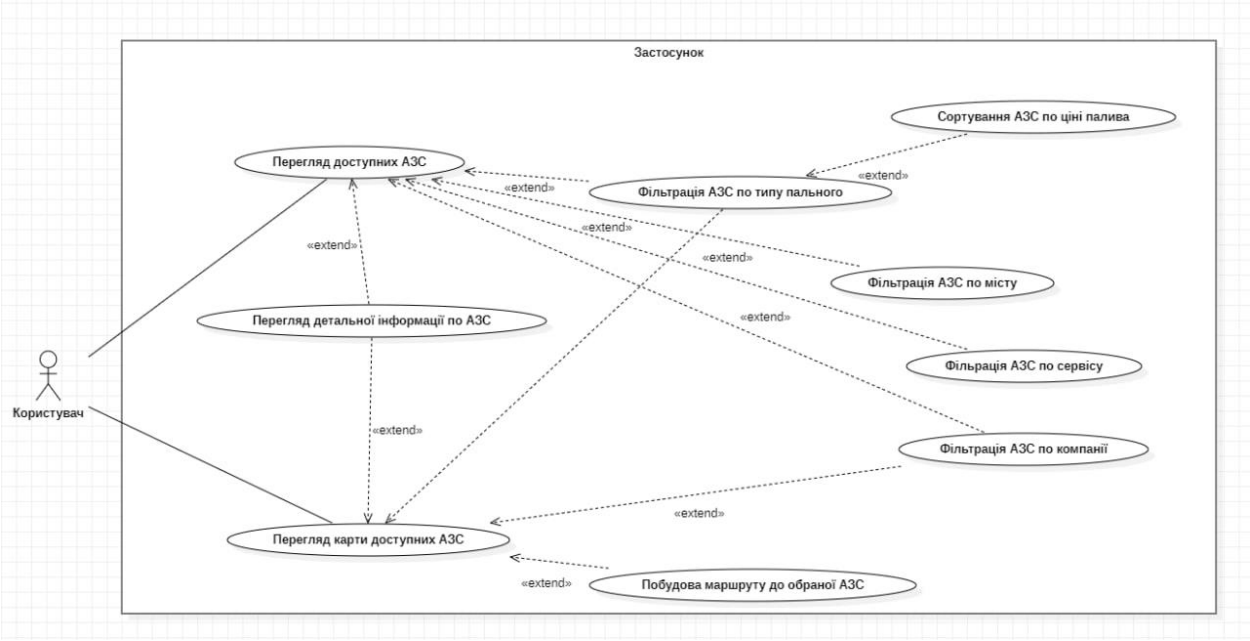


Рис. 2.3. Діаграма варіантів використання

Опис моделі варіантів використання

Дійові особи:

- Користувач – користувач програмного продукту.

Варіанти використання:

- Перегляд карти доступних АЗС.
- Фільтрація АЗС.
- Фільтрація АЗС по місту.
- Фільтрація АЗС по типу пального.
- Фільтрація АЗС по компаніям.
- Фільтрація АЗС по сервісу.
- Перегляд детальної інформації по АЗС.
- Побудова маршруту до обраної АЗС
- Перегляд історії запитів

- Сортування АЗС по ціні

Оформлення сценаріїв

Основний сценарій:

1. Користувач переглядає карту з доступними АЗС.
2. Користувач сортує АЗС по ціні.
3. Користувач обирає параметри фільтрації, що його цікавлять.
4. Користувач фільтрує АЗС.
5. Користувач обирає цікаву йому з відфільтрованих АЗС.
6. Користувач натискає на цю АЗС у списку та переглядає детальну інформацію по ній.
7. Користувач переглядає маршрут до обраної АЗС попередньо задавши пункт відправлення.

Альтернативний сценарій:

1. Якщо сервер однієї з компаній не відповідає, користувачу доступна попередньо змаплена інформація.
2. Якщо користувач не знайде необхідної інформації, він може перейти на ресурс тієї мережі АЗС яка його цікавить і вказана для кожної заправної станції у вигляді посилання.

Опишемо діаграму послідовностей для застосунку, що проектується. Діаграма складається із наступних компонентів взаємодії:

1. Користувач
2. Застосунок
3. Ресурси мереж АЗС
4. БД

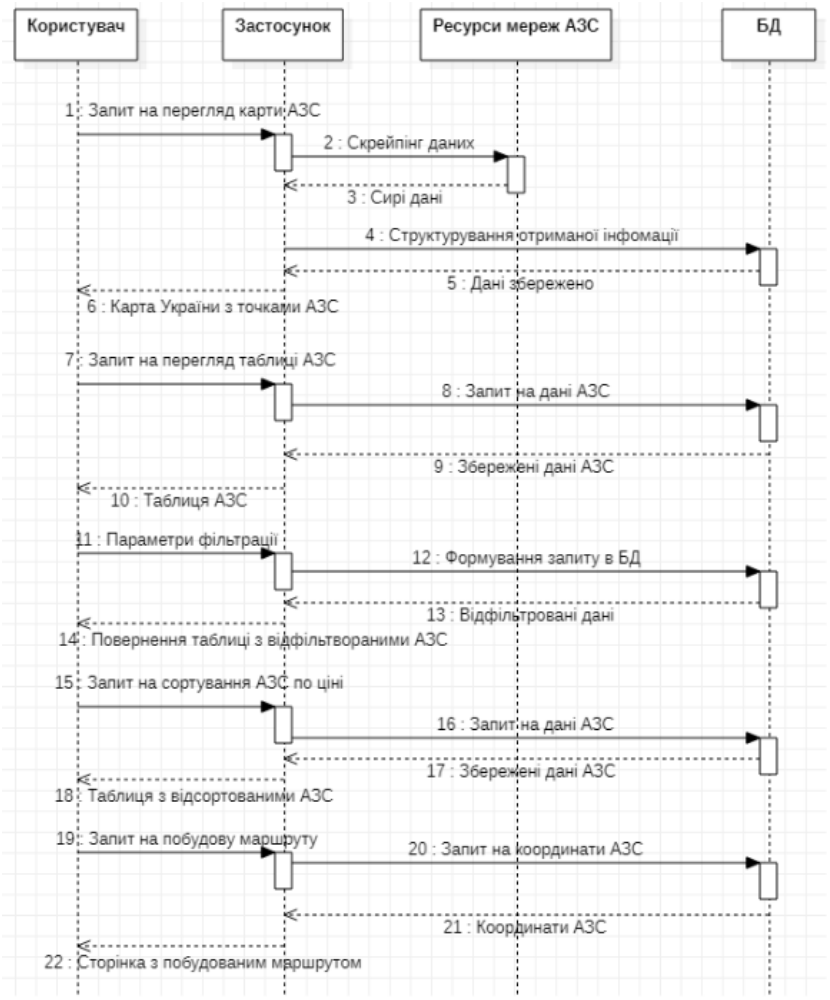


Рис. 2.4. Діаграма послідовностей

Під час першої взаємодії клієнта з системою відпрацьовує скрейпер-модуль, що збирає та структурує усі необхідні для відображення дані. Інформація зберігається в базі даних та використовується для опрацювання подальших запитів. Після того як система накопичила дані із сайтів мереж АЗС користувач має змогу переглядати карту України з розташованими на ній доступними АЗС, фільтрувати АЗС, прокладати маршрути та переглядати таблицю з доступними видами палива.

2.3 Розробка архітектури застосунку

Система моніторингу наявності пального на базі скрейпінг-технологій по найбільших мережах АЗС України складається з 3 підсистем:

Підсистема керування даними

1. Модуль збереження інформації в базу даних (модуль, що мапить об’єкти узагальненої інформаційної моделі у об’єкти фізичної моделі БД та зберігає їх).
2. Модуль повернення інформації з бази даних (модуль, що мапить об’єкти фізичної моделі БД у об’єкти узагальненої інформаційної моделі та віддає їх верхнім модулям на подальшу обробку).

Підсистема пошуку

1. Модуль виконання запитів (модуль, що здійснює пошук інформації, шляхом виконання запитів до першоджерел).
2. Модуль менеджменту запитів (модуль, що підтримує стабільність та надійність роботи модуля виконання запитів шляхом підтримки перевиконання запитів та сценаріїв обробки помилок).

Підсистема мапінгу

1. Модуль конвертації (модуль, що здійснює конвертацію отриманих сирих даних до програмних, що можуть в подальшому оброблятись іншими модулями).
2. Модуль приведення (модуль, що здійснює приведення даних до єдиної інформаційної моделі).

Підсистема фільтрації

1. Модуль користувацької фільтрації (модуль, що здійснює пошук інформації за заданими параметрами користувача).
2. Модуль відображення (модуль, що перетворює дані у модель відображення - карту або таблицю з АЗС).

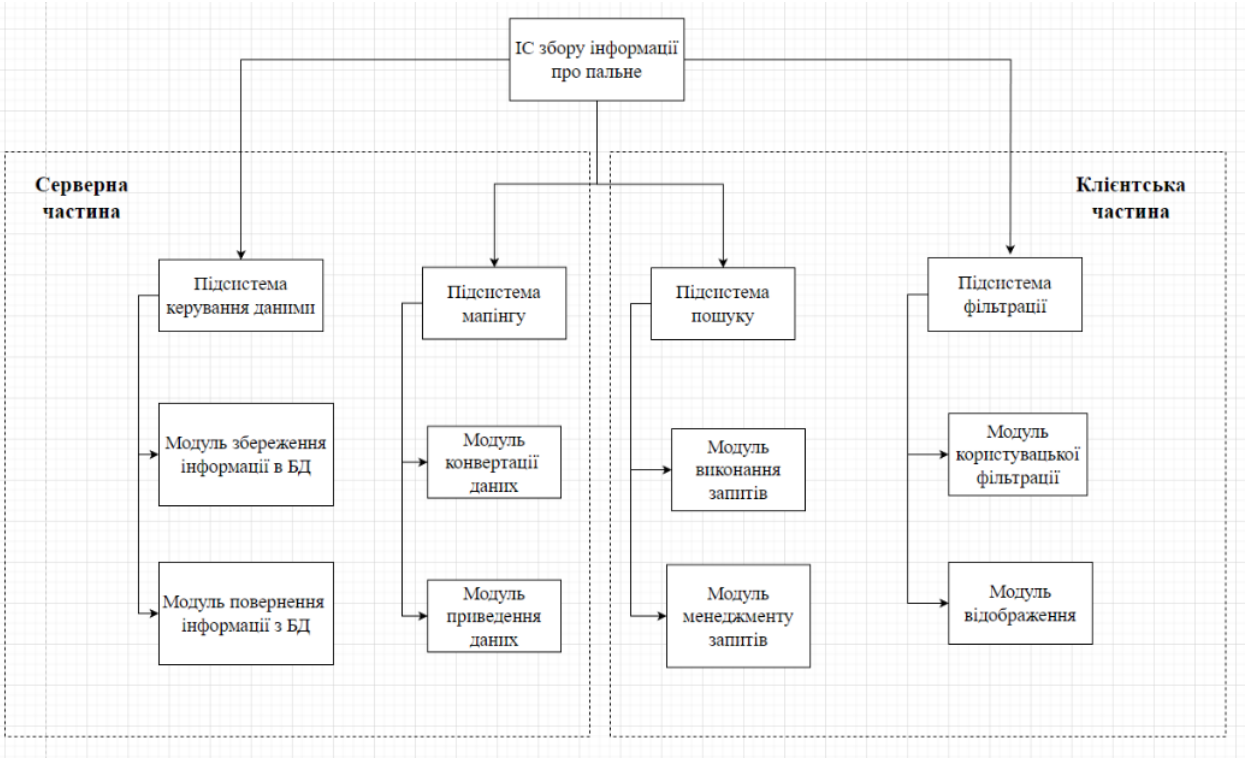


Рис. 2.5. Структурно-функціональна діаграма системи моніторингу наявності пального

Окрім того, під час проектування архітектури серверної частини застосунку був використаний GoF-паттерн “Адаптер” [2]. Це структурний патерн проектування, що дозволяє об'єктам з несумісними інтерфейсами працювати разом.

Загалом, компоненти застосунку можна умовно розділити на 2 категоріїї за функціями, що вони виконують:

Пошук-запис інформації

Scrapper - компоненти із заданим інтерфейсом виконують функцію пошуку інформації, роблять запити на відповідні ресурси і дістають з них сирі дані у першоджерельному вигляді.

Adapter - компоненти із заданим інтерфейсом виконують функцію “перетворювачів”, вони парсять необроблені дані, отримані від Scrapper-компонента у класи-переносники - що по суті є унікальними відображеннями корисних даних кожної заправки у java-класи.

Mapper - компоненти із заданим інтерфейсом виконують найважливішу функцію - перетворюють класи-переносники, отримані від Adapter-компонента в узагальненні моделі, які в подальшому зберігаються у базу даних.

Виконання користувацьких запитів

Matcher - компоненти із заданим інтерфейсом виконують функції фільтрування даних за заданими користувацькими параметрами.

Repository - компоненти із заданим інтерфейсом виконують функції взаємодії застосунку з базою даних (crud-операції).

Service - компоненти із заданим інтерфейсом виконують функції логічної обробки даних із бази даних відповідно до користувацького запиту.

Controller - компоненти із заданим інтерфейсом виконують функції надання REST API для UI-частини розробленої системи та проксювання запитів на відповідні логічні компоненти програми.

Відповідно до компонентів застосунку можемо побудувати узагальнену архітектуру користувацької та серверної частин:

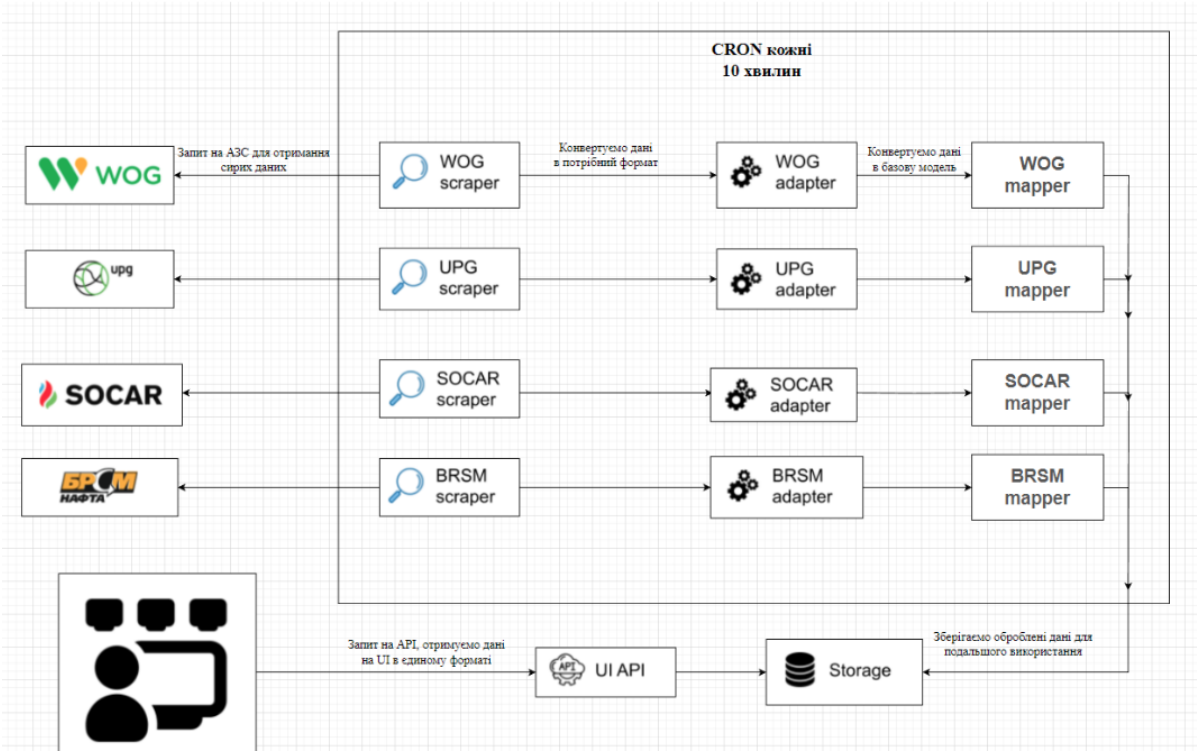


Рис. 2.6. Узагальнена архітектура роботи користувацької та серверної частин застосунку

Опис узагальненої взаємодії компонентів

Scrapper-компонент робить виклик до ресурсу мережі, за збір даних з якої він відповідає. Під кожну мережу спроектований свій scrapper-компонент так як способи видобутку інформації для кожної мережі індивідуальні. Роботу scrapper-компоненту ініціює adapter-компонент, передаючи отримані необроблені дані відповідному mapper-компоненту, який в свою чергу структурує їх у програмні об’єкти, що відповідають спроектованій інформаційній моделі. Отримані від mapper-компонента об’єкти повертаються назад на adapter-компонент, який ініціює їх збереження до реляційної бази даних. Вищеописаний процес є повторюваним, і відбувається кожних 10 хвилин. Тому дані, що буде бачити користувач завжди будуть актуальними. Зі сторони клієнтської частини ми маємо запит від користувача, що проходить низку перетворень та опрацьовується користувацькою частиною застосунку за рахунок видобутку необхідних даних безпосередньо з бази даних. Тобто клінтська і серверна частина працюють незалежно.

2.4 Розробка універсальної інформаційної моделі

Важливою частиною проектування додатку є розробка універсальної інформаційної моделі, що узагальнює корисні для користувача дані по всім АЗС усіх заправок, що відповідно дає змогу здійснювати пошук, фільтрацію та сортування по заданим параметрам. Основними об’єктами, що узагальнюють дані із ресурсів мереж АЗС є:

1. Компанія - містить все, що стосується конкретної мережі АЗС (контактні дані, посилання на першоджерело)
2. Станція АЗС - містить інформацію про конкретну заправку (координати, адреса, послуги що надаються, доступне в даний момент паливо, графік роботи)
3. Сервіс - містить інформацію про послугу що надається конкретною мережею АЗС, її наявністю на тій чи іншій станції

4. Паливо - містить інформацію, що цікавить користувача відносно конкретного палива (ціна, тип, наявність, спосіб оплати)

Для кращого розуміння структури бази даних побудуємо модель “сутність-зв'язок” у нотації Пітера Чена (Рис. 2.7).

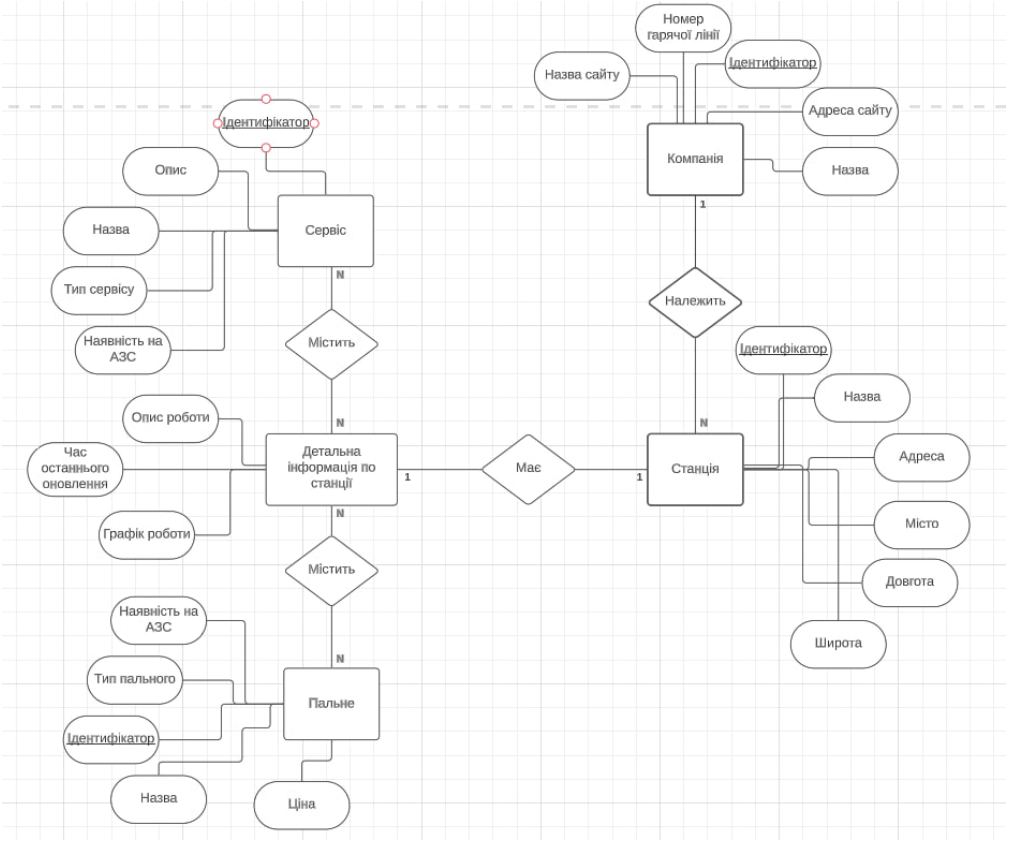


Рис. 2.7. Схема бази даних застосунку у нотації Чена

Так як інформація, що скрейпиться застосунком має зберігатись у базі даних була розроблена її фізична модель (Рис. 2.8). Опишемо атрибути таблиць баз даних.

Таблиця “Company” - зберігає інформацію про компанію.

- id - унікальний ідентифікатор компанії.
- hot_line - номер телефону гарячої лінії компанії.
- link_name - коротка назва сайту компанії.
- map_link - посилання на сайт компанії.
- name - назва компанії.

Таблиця “Station” - зберігає загальну інформацію по кожній станції.

- id - унікальний ідентифікатор станції.
- address - фізична адреса станції.
- city - місто, у якому розташована станція.
- lat - географічна широта місцерозташування АЗС.
- lng - географічна довгота місцерозташування АЗС.
- name - назва станції.
- company_id - посилання на компанію, до якої належить станція.
- station_info_id - посилання на об’єкт повної інформації про станцію.

Таблиця “Station_Info” - зберігає детальну інформацію по станції.

- id - унікальний ідентифікатор моделі station info.
- last_update - дата та час останнього оновлення інформації про станцію.
- schedule - графік роботи станції.
- work_description - опис доступних палив та їх вартості (способів оплати).

Таблиця “Fuel” - зберігає інформацію про паливо, що реалізується на станціях.

- id - унікальний ідентифікатор палива.
- fuel_type - тип палива (A92, A95, DIESEL, GAS, UNKNOWN).
- is_available - інформація про наявність палива.
- name - назва палива.
- price - ціна палива.

Таблиця “Station_Info_Fuels” - зберігає мапінг палив на станції, які їх реалізовують.

- station_info_id - унікальний ідентифікатор моделі station info.
- fuels_id - унікальний ідентифікатор палива.

Таблиця “Service” - зберігає інформацію про сервіси, що надаються станціями.

- id - унікальний ідентифікатор сервісу.
- service_type - тип сервісу(FOOD, GOODS, CAR_SERVICE, OTHER).

- is_available - інформація про доступність сервісу.
- name - назва сервісу.
- description - опис сервісу.

Таблиця “Station_Info_Services” - зберігає мапінг сервісів на станції, які їх реалізовують.

- station_info_id - унікальний ідентифікатор моделі station info.
- services_id - унікальний ідентифікатор сервісу.

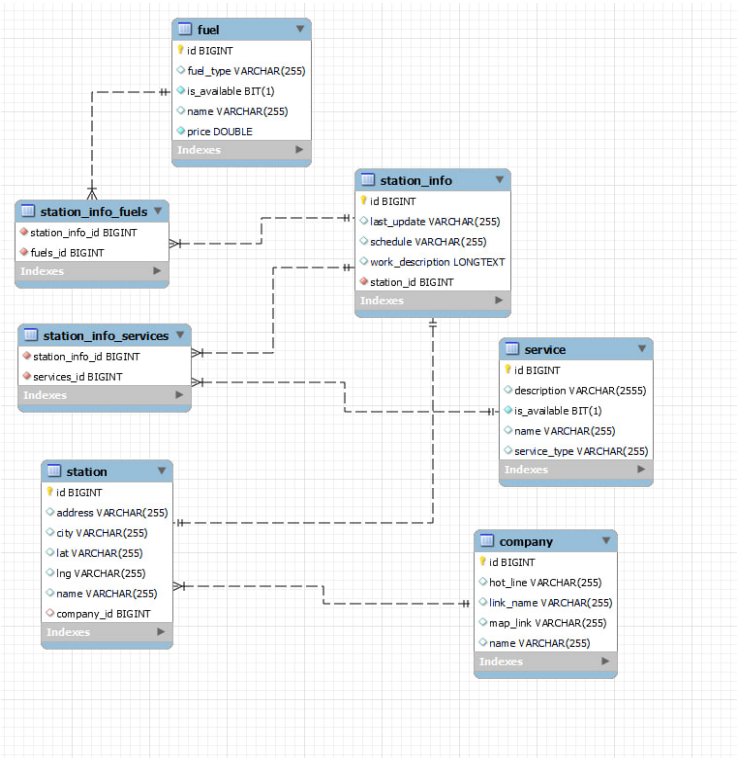


Рис. 2.8. Фізична модель бази даних застосунку

2.5 Розробка прототипів сторінок користувацького інтерфейсу

В рамках проектування необхідним кроком є розробка прототипів сторінок користувацького інтерфейсу для того аби сформувати бачення візуальної частини застосунку, яка інформація в ній повинна бути присутня, що відповідно впливає на розробку і серверної частини. Для користувача майбутнього додатку важливими є:

1. Лаконічність у зовнішньому вигляді застосунку
2. Інтуїтивність у пошуку необхідних функцій

3. Щонайменша кількість тексту
4. Щонайменша кількість часу на навігацію
5. Маленький час відгуку

Відповідно до вищезазначених вимог були розроблені прототипи сторінок інтерфейсу майбутнього додатку:

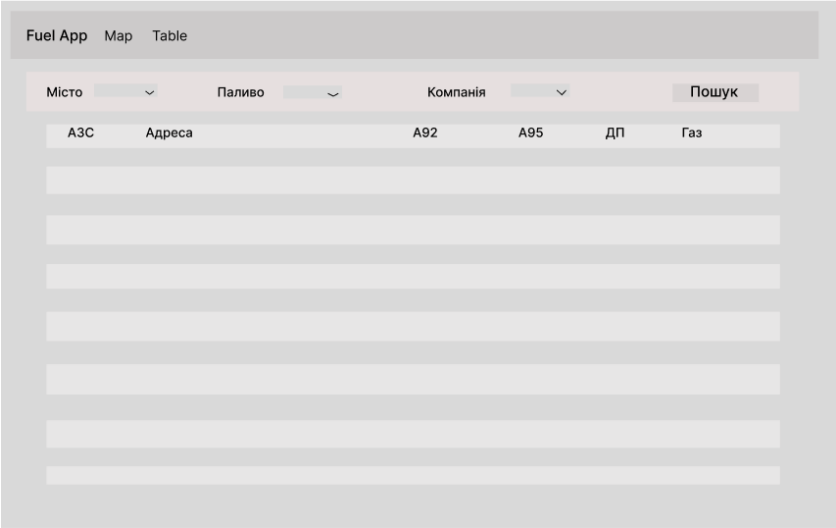


Рис. 2.9. Прототип таблиці головної сторінки застосунку

Головна сторінка має одразу демонструвати весь функціонал застосунку і показувати таблицю з АЗС та доступним на них паливом. До того ж має бути присутній легкий у використанні та помітний фільтр по місту, типу палива та назві компанії, що цікавлять користувача.

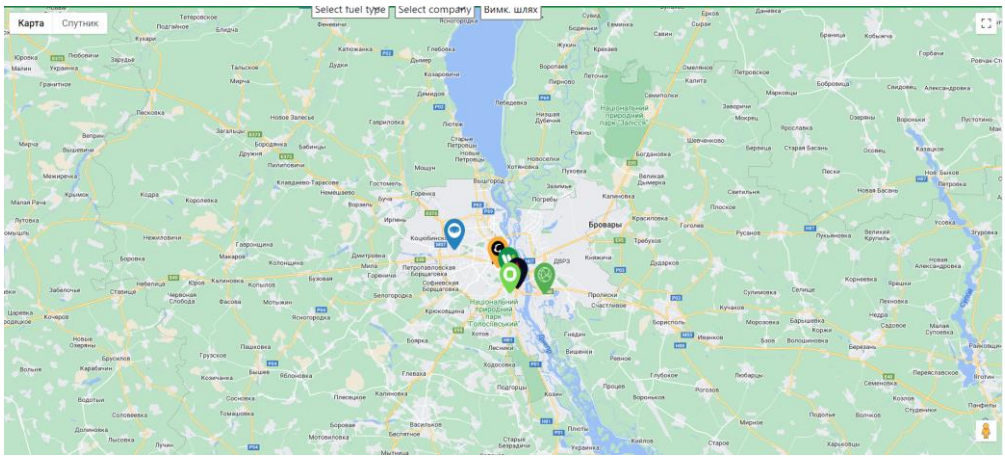


Рис. 2.10. Прототип карти

Відповідно до функціональних вимог застосунку, він має містити карту інтегровану з Google API. Тому прототипом є google-карта з відмітками у вигляді лого відповідної автозаправної компанії для швидшої орієнтації користувача. Окрім того, на карті мають бути аналогічні функції фільтрації АЗС.

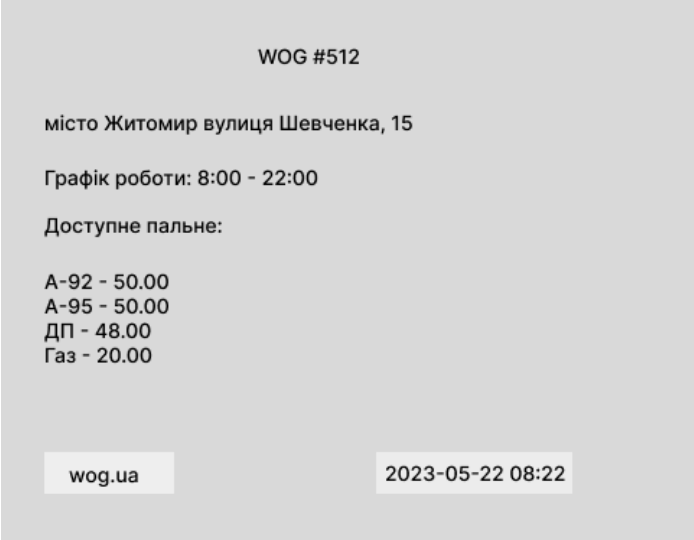


Рис. 2.11. Прототип вікна інформації про АЗС

Для того, аби користувач мав змогу дізнатись більше про обрану ним АЗС, застосунок має містити вікно з детальною інформацією про станцію. Тобто, користувач має бачити все, що може бути йому цікаво, а саме – адреса АЗС, її графік роботи, яке паливо в даний момент доступне на станції та за якою ціною, посилання на джерело інформації, тобто офіційний сайт компанії, а також дату та час останнього оновлення інформації.

Висновок до другого розділу

Під час написання другого розділу було спроектовано архітектуру додатку, що задовольняє поставленій задачі даної дипломної роботи. Також було описано архітектуру застосунку з точки зору серверу-клієнта і модулів інформаційної системи, описано основні компоненти системи та їх взаємодію. В цьому розділі також проведено функціональний аналіз застосунку, описано бізнес процеси предметної області, спроектовано прототипи сторінок інтерфейсу користувача, а

також розроблену інформаційну модель майбутньої бази даних, яка демонструє спосіб структурування інформації, що дістається із джерел мереж АЗС.

РОЗДІЛ 3. ПРОГРАМНА РЕАЛІЗАЦІЯ ЗАСТОСУНКУ

3.1. Обґрунтування вибору інструментальних засобів для програмної реалізації застосунку

Розроблений застосунок складається з серверної та клієнтської частин для розділення задач по колеціюванню даних та їх відображення. Відповідно для кожної програмної частини були вибрані свої технології, що більше підходять для реалізації її функціоналу.

Серверна частина

Основна задача серверної частини - збирання необхідної інформації, її мапінг, приведення до єдиної гнучкої моделі даних та збереження її у базу даних. Цей процес виконується з деякою періодичністю, тому основний акцент у виборі технологій для серверної частини був поставлений на СУБД та відповідно мову, яка б мала вибір бібліотек, що дозволяли б реалізувати парсинг html та була б швидкою та зручною для веб-розробки.

Мова

Серверна частина написана на мові java. Так як java одна з найпопулярніших мов програмування, що використовується у веб-середовищі, має широкий спектр бібліотек для веб-розробки та багато надійних і практичних веб-фреймворків. До того ж java незалежна від платформи на якій запускається, що дозволяє запускати код на будь-якому сервері, що підтримує середовище для виконання байт-коду. Також java базована на об’єктно-орієнтованому підході, що дало змогу спроектувати досить гнучну, модульну архітектуру застосунку та без проблем інтегруватись з базою даних.

СУБД

В якості СУБД була обрана MySQL як одна з найпопулярніших для інтеграції з додатками, написаними на java. До того ж MySQL має достатньо велике різноманіття доступних інструментів для створення майже будь-якого застосунку.

Бібліотеки

Spring Boot - модуль фреймворку Spring, що дозволяє швидко створювати та запускати веб-застосунки на базі Spring. Spring Boot має досить великий функціонал, але його найбільш значущими особливостями є: управління залежностями, автоматична конфігурація та вбудовані контейнери сервлетів [4].

Jsoup - це бібліотека Java з відкритим вихідним кодом, призначена для аналізу, вилучення та обробки даних, що зберігаються в HTML-документах [6].

Hibernate - це бібліотека, призначена для вирішення задач об'єктно-реляційного відображення, найпопулярніша реалізація специфікації JPA [3].

Клієнтська частина

Основна задача клієнтської частини - побудова інтерфейсів користувача та спілкування з сервером через REST API.

Мова

Клієнтська частина написана на TypeScript. TypeScript - мова програмування, представлена Microsoft у 2012 році і позиціонована як засіб розробки веб-додатків, що розширює можливості JavaScript. Саме ця мова була обрана через наявність типізації, підтримку найновіших функцій JavaScript, сумісність із браузером.

Бібліотеки

React - JavaScript-бібліотека з відкритим вихідним кодом для розробки інтерфейсів користувача. Його мета — надати високу швидкість розробки, простоту та масштабованість [5] [14].

Next.js - відкритий JavaScript фреймворк, створений поверх React.js для створення веб-додатків. Він дозволяє створювати веб-програми з покращеною продуктивністю та покращеним користувальницьким досвідом за допомогою додаткових функцій попереднього рендерингу, таких як повноцінний рендеринг на стороні сервера (SSR) та статична генерація сторінок (SSG) [10].

3.2 Структура класів програмної реалізації

Усі класи розробленого застосунку можна умовно розділити на 3 модулі за функцією, що в них закладена:

1. MVC-модуль - є кінцевою точкою застосунку, створений для взаємодії інших модулів з користувацькою частиною
2. Модуль керування даними - є ключовим механізмом збору інформації, створений для взаємодії з БД
3. Модуль мапінгу - є бізнес-логікою застосунку, містить послідовність класів, що приводять сирі дані до вигляду універсальної інформаційної моделі

Наведемо діаграму класів для кожного модуля:

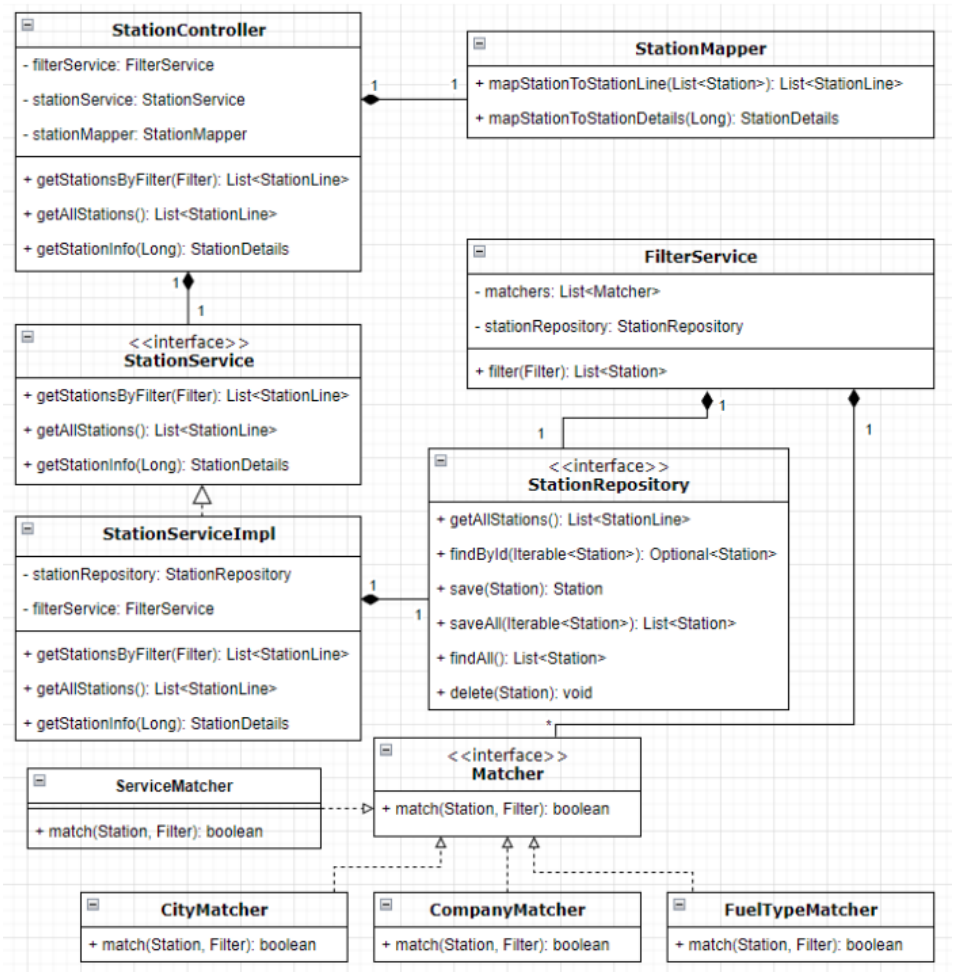


Рис. 3.1. Діаграма класів програмного модулю MVC

Кінцевою точкою застосунку є клас StationController. Він є API серверної частини, яку використовує клієнтська для побудови відображень. Controller клас дістає запитувані дані безпосередньо із БД, але перед тим запит проходить ще низку шарів MVC-патерну, а саме StationService та StationRepository класи. Окрім того, отримані дані із БД мапляться у моделі, що використовуються UI-частиною перед поверненням за допомогою StationMapper класу. Також клієнтська частина робить запити на фільтрування даних. Цей функціонал реалізує FilterService, що використовує Matcher-класи для застосування фільтрування на вибірці усіх АЗС за кожним можливим параметром.

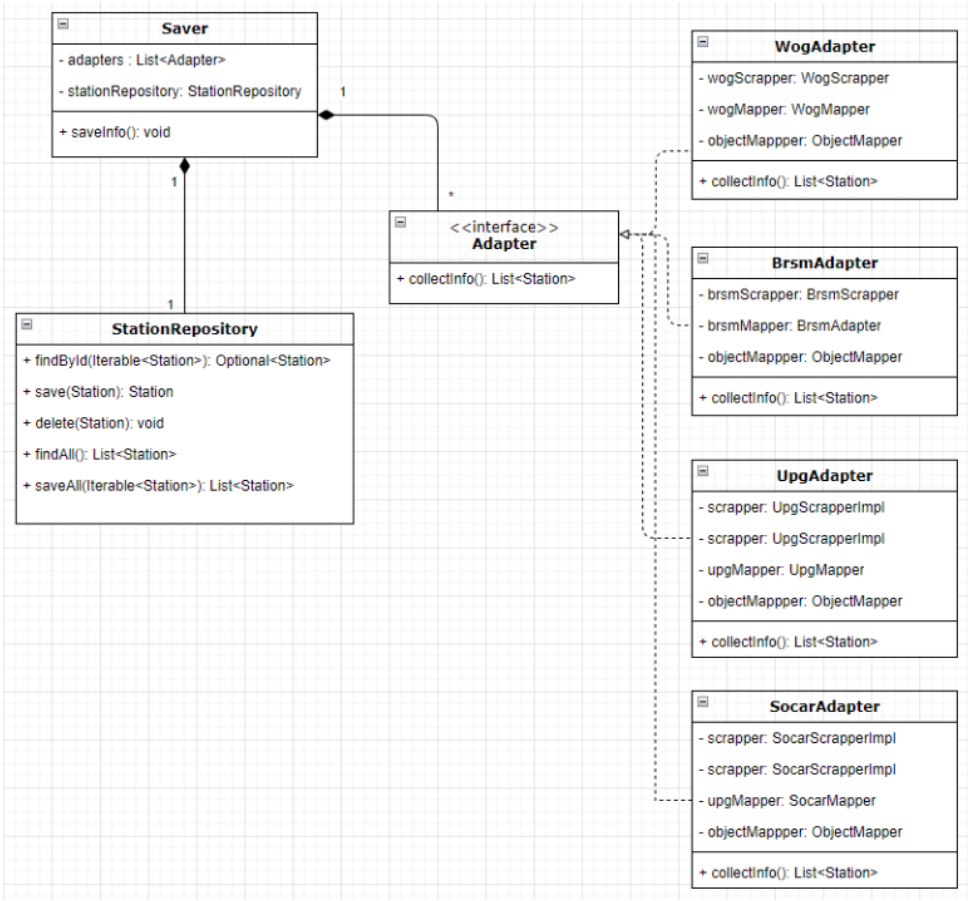


Рис. 3.2. Діаграма класів програмного модулю керування даними

Ключовим елементом даного модулю є клас Saver, він керує збереженням видобутих та приведених даних, а саме використовуючи класи-адаптери (WogAdapter, UpgAdapter, BrsmAdapter, SocarAdapter) кожні 10 хвилин робить запити до модулю мапінгу та зберігає отримані дані у БД за допомогою інтерфейсу StationRepository.

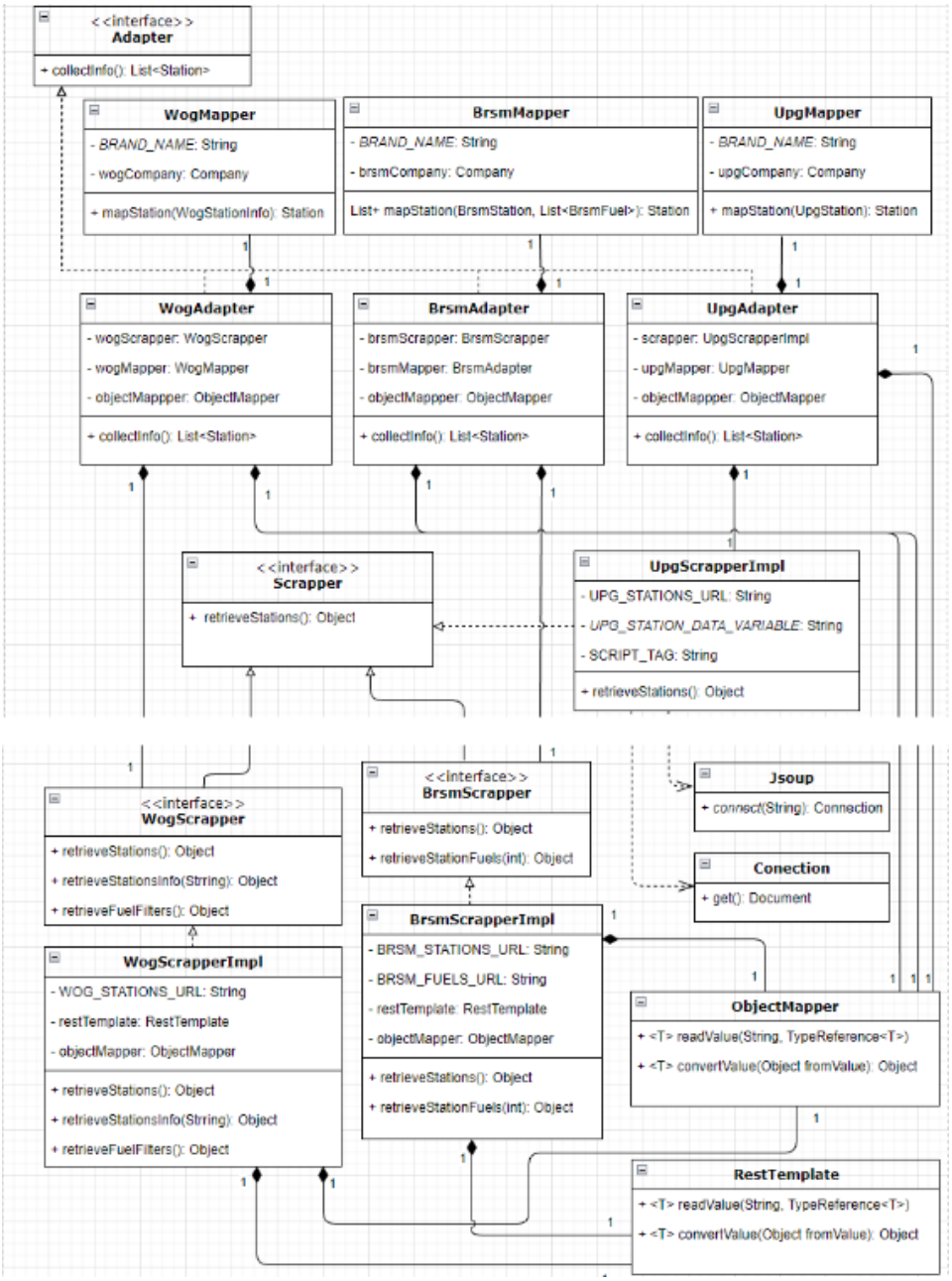


Рис. 3.3. Діаграма класів програмного модулю мапінгу

Найоб’ємнішим та найскладнішим з точки зору реалізації є модуль мапінгу. Він включає в себе як реалізацію самого скрейпінгу так і всієї бізнес-логіки приведення даних від різних мереж у єдину інформаційну модель. Класи-скрейпери (WogScrapperImpl, UpgScrapperImpl, BrsmScrapperImpl, SocarScrapperImpl) реалізують логіку програмного видобутку даних з ресурсів мереж АЗС [13]. Залежно від мережі кожен скрейпер використовує свою технологію пошуку. UpgScrapperImpl використовує класи бібліотеки Jsoup, інші - клієнта для виконання HTTP-запитів. На рівень вище над класами-скрейперами

знаходяться класи-адаптери. Вони реалізують загальні методи, що використовуються в подальшому іншими модулями для оперування видобутими даними. Класи-адаптери (WogAdapter, UpgAdapter, BrsmAdapter, SocarAdapter) поєднують логіку скрейперів та класів-маперів. Класи-мапери (WogMapper, UpgMapper, BrsmMapper, SocarMapper) в свою чергу приводять отримані від скрейперів кастомні об’єкти, що повністю відповідають моделі даних конкретної мережі АЗС у об’єкти узагальненої моделі, що потім зберігаються у БД.

3.3 Контрольний приклад роботи застосунку

Робота застосунку починається із запуску серверної частини
Під час запуску застосунок створює необхідні таблиці в базі даних.

```
Hibernate: drop table if exists company
Hibernate: drop table if exists fuel
Hibernate: drop table if exists station
Hibernate: drop table if exists station_info
Hibernate: drop table if exists station_info_fuels
Hibernate: create table company (id bigint not null auto_increment, hot_line varchar(255), link_name varchar(255), map_link varchar(255), name var
Hibernate: create table fuel (id bigint not null auto_increment, fuel_type varchar(255), is_available bit not null, name varchar(255), price doubl
Hibernate: create table station (id bigint not null auto_increment, address varchar(255), city varchar(255), name varchar(255), company_id bigint,
Hibernate: create table station_info (id bigint not null auto_increment, last_update varchar(255), schedule varchar(255), work_description longtex
Hibernate: create table station_info_fuels (station_info_id bigint not null, fuels_id bigint not null) engine=InnoDB
Hibernate: alter table company add constraint UK_niu8sfil2gxywcru9ah3r4ec5 unique (name)
Hibernate: alter table station add constraint FKilcxplfp9ux78uqas46pckb47 foreign key (company_id) references company (id)
Hibernate: alter table station add constraint FK6m5k1329tws3u0xhglic8sk3t foreign key (station_info_id) references station_info (id)
Hibernate: alter table station_info_fuels add constraint FK6m3cbfipoqdg0egv2tju9ue0p foreign key (fuels_id) references fuel (id)
Hibernate: alter table station_info_fuels add constraint FK363v9sfgw11te9sqbimelhbnn foreign key (station_info_id) references station_info (id)
```

Рис. 3.4. Логи створення таблиць у бд

```
2022-12-11 11:10:04.474  WARN 14456 --- [  restartedMain] JpaBaseConfiguration$JpaWebConfiguration : spring.jpa.open-in-view is enabled by defaul
2022-12-11 11:10:04.659  WARN 14456 --- [  restartedMain] ion$DefaultTemplateResolverConfiguration : Cannot find template location: classpath:/te
2022-12-11 11:10:04.761  INFO 14456 --- [  restartedMain] o.s.b.d.a.OptionalLiveReloadServer       : LiveReload server is running on port 35729
2022-12-11 11:10:04.791  INFO 14456 --- [  restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port(s): 8080 (http) with
2022-12-11 11:10:04.801  INFO 14456 --- [  restartedMain] com.stranger.gas.GasApplication          : Started GasApplication in 4.306 seconds (JVM
```

Рис. 3.5. Логи успішного запуску серверної частини застосунку

Далі компонент Saver починає збір інформації по мережах АЗС “WOG”, “BRSM”, “SOCAR”, “UPG”. Відповідно бачимо логи запису інформації у таблиці.

```
Hibernate: insert into fuel (fuel_type, is_available, name, price) values (?, ?, ?, ?)
Hibernate: insert into fuel (fuel_type, is_available, name, price) values (?, ?, ?, ?)
Hibernate: insert into fuel (fuel_type, is_available, name, price) values (?, ?, ?, ?)
Hibernate: insert into fuel (fuel_type, is_available, name, price) values (?, ?, ?, ?)
Hibernate: insert into station (address, city, company_id, name, station_info_id) values (?, ?, ?, ?, ?)
Hibernate: insert into station_info (last_update, schedule, work_description) values (?, ?, ?)
Hibernate: insert into fuel (fuel_type, is_available, name, price) values (?, ?, ?, ?)
Hibernate: insert into fuel (fuel_type, is_available, name, price) values (?, ?, ?, ?)
Hibernate: insert into fuel (fuel_type, is_available, name, price) values (?, ?, ?, ?)
Hibernate: insert into fuel (fuel_type, is_available, name, price) values (?, ?, ?, ?)
Hibernate: insert into fuel (fuel_type, is_available, name, price) values (?, ?, ?, ?)
Hibernate: insert into station (address, city, company_id, name, station_info_id) values (?, ?, ?, ?, ?)
Hibernate: insert into station_info (last_update, schedule, work_description) values (?, ?, ?)
Hibernate: insert into fuel (fuel_type, is_available, name, price) values (?, ?, ?, ?)
Hibernate: insert into fuel (fuel_type, is_available, name, price) values (?, ?, ?, ?)
Hibernate: insert into fuel (fuel_type, is_available, name, price) values (?, ?, ?, ?)
```

Рис. 3.6. Логи запису інформації у таблиці бд

Після того як модуль збору інформації відпрацював бачимо відповідні записи у таблицях бази даних.

id	address	city	name	company_id	station_info_id
1	Рівненська обл., Рокитнівський р-н, с/рада ...	Рівне	АЗС №2	2	1
2	м. Львів, вул. Шевченка, 200	Львів	АЗС №5	2	2
3	Житомирська обл., Чуднівський р-н, с. Дуби...	Житомир	АЗС №9	2	3
4	Сумська обл., м. Глухів, вул. Путивльська, 6...	Суми	АЗС №16	2	4
5	Житомирська обл., Брусилівський р-н, с. Йос...	Житомир	АЗС №17	2	5
6	Київська обл., Бородянський р-н, с/рада Заг...	Київ	АЗС №18	2	6
7	Київська обл., Переяслав-Хмельницький р-н,...	Київ	АЗС №19	2	7
8	Запорізька обл., м. Запоріжжя, вул. Аврамен...	Запоріжжя	АЗС №43	2	8
9	Запорізька обл., м. Запоріжжя, вул. Адмірал...	Запоріжжя	АЗС №44	2	9
10	Закарпатська обл., Ужгородський р-н, с. Кін...	Ужгород	АЗС №54	2	10
11	Закарпатська обл, м. Тячів, вул. Армійська, ...	Ужгород	АЗС №55	2	11
12	Львівська обл., м. Броди, а/д Київ - Чоп 441 ...	Львів	АЗС №56	2	12
13	Львівська обл., м. Броди, а/д Київ - Чоп 441 ...	Львів	АЗС №57	2	13
14	Кіровоградська обл., Голованівський р-н, ав...	Кропивни...	АЗС №58	2	14

Рис. 3.7. Таблиця станцій

id	hot_line	link_name	map_link	name
1	0800300525	wog.ua	https://wog.ua/ua/map/	WOG
2	0800500064	upg.ua	https://upg.ua/merezha_azk/	UPG
3	0800303404	brsm-nafta.com	https://brsm-nafta.com/map	BRSM
4	0800508585	socar.ua	https://socar.ua/map	SOCAR

Рис. 3.8. Таблиця компаній

id	last_update	schedule	work_description
1	2023-05-16 01:41	Відчинено	Номер телефону АЗС - 38(067)217-66-84
2	2023-05-16 01:41	Відчинено	Номер телефону АЗС - 38(067)404-23-09
3	2023-05-16 01:41	Зачинено	Номер телефону АЗС - 38(067)208-63-70
4	2023-05-16 01:41	Зачинено	Номер телефону АЗС - 38(067)405-07-16
5	2023-05-16 01:41	Відчинено	Номер телефону АЗС - 38(067)405-06-91
6	2023-05-16 01:41	Зачинено	Номер телефону АЗС - 38(067)404-81-59
7	2023-05-16 01:41	Відчинено	Номер телефону АЗС - 38(067)404-81-77
8	2023-05-16 01:41	Зачинено	Номер телефону АЗС - 38(067)404-81-49
9	2023-05-16 01:41	Відчинено	Номер телефону АЗС - 38(067)404-81-63
10	2023-05-16 01:41	Зачинено	Номер телефону АЗС - 38(067)209-05-12
11	2023-05-16 01:41	Відчинено	Номер телефону АЗС - 38(067)217-84-00
12	2023-05-16 01:41	Відчинено	Номер телефону АЗС - 38(067)209-26-71
13	2023-05-16 01:41	Відчинено	Номер телефону АЗС - 38(067)404-57-58
14	2023-05-16 01:41	Відчинено	Номер телефону АЗС - 38(067)404-57-62
15	2023-05-16 01:41	Зачинено	Номер телефону АЗС - 38(067)404-57-72
16	2023-05-16 01:41	Відчинено	Номер телефону АЗС - 38(067)556-53-04

Рис. 3.9. Таблиця детальної інформації про станції

id	description	is_available	name	service_type
4	Можливість зробити безкоштовний телефон...	1	Телефон	OTHER
5	Безкоштовний швидкісний бездротовий інте...	1	Wi-Fi	OTHER
6	Ми пропонуємо широкий асортимент кавових...	1	Кава/напої	FOOD
7	Buta Market - це можливість придбати автот...	1	Buta Market	GOODS
8	Сервіс з використанням електронного гаман...	1	SOCAR Pay	OTHER
9	Смарт-карта sCard – це універсальна паливн...	1	Картки sCard	OTHER
10	Зручно та швидко - поповнити рахунки в пл...	1	Платіжний термінал	OTHER
11	Якщо за якоюсь з причин ви не маєте можли...	1	Виніс термінала	OTHER
12	Наявність колонки для швидкої видачі дизел...	1	Швидкісна ПРК TIR	CAR_SERVICE
13	Можливість зробити безкоштовний телефон...	1	Телефон	OTHER
14	Безкоштовна зарядка для мобільних пристр...	1	Зарядка для смартфо...	OTHER
15	Безкоштовний швидкісний бездротовий інте...	1	Wi-Fi	OTHER
16	Чиста і безкоштовна вбиральня знаходиться...	1	WC	OTHER
17	Ми пропонуємо широкий асортимент кавових...	1	Кава/напої	FOOD
18	Buta Market - це можливість придбати автот...	1	Buta Market	GOODS
19	Наявність місць для паркування для автомоб...	1	Парковка	CAR_SERVICE

Рис. 3.10. Таблиця сервісів

id	fuel_type	is_available	name	price
1	A95	1	A 95 PREMIUM PLUS	33.99
2	DIESEL	1	ДП	41.49
3	A95	1	A 95 EURO PLUS	42.49
4	GAS	1	Газ+ PLUS	20.59
5	A95	1	A 95 EURO PLUS	42.49
6	A95	1	A 95 PREMIUM PLUS	33.99
7	DIESEL	1	ДП	41.49
8	GAS	1	Газ+ PLUS	20.59
9	GAS	1	Газ+ PLUS	20.59
10	DIESEL	1	ДП	40.99
11	A95	1	A 95 EURO PLUS	41.99
12	A95	1	A 95 PREMIUM PLUS	33.99
13	DIESEL	1	ДП	40.99

Рис. 3.11. Таблиця пального.

Наступним кроком є запуск клієнтської частини застосунку.

```
PS C:\Users\maryna.voloshyna\Desktop\study_4\diploma\azsclient> npm run dev

> azsclient@0.1.0 dev
> next dev

ready - started server on 0.0.0.0:3000, url: http://localhost:3000
warn  - You have enabled experimental feature (appDir) in next.config.js.
warn  - Experimental features are not covered by semver, and may cause unexpected or broken application behavior. Use at your ow
n risk.
info  - Thank you for testing `appDir` please leave your feedback at https://nextjs.link/app-feedback
```

Рис. 3.12. Логи успішного запуску клієнтської частини застосунку.

Після цього переходимо на головну сторінку застосунку. Тут бачимо список усіх автозаправних станцій, що є доступними по мережах “WOG”, “BRSM”, “SOCAR”, “UPG”.

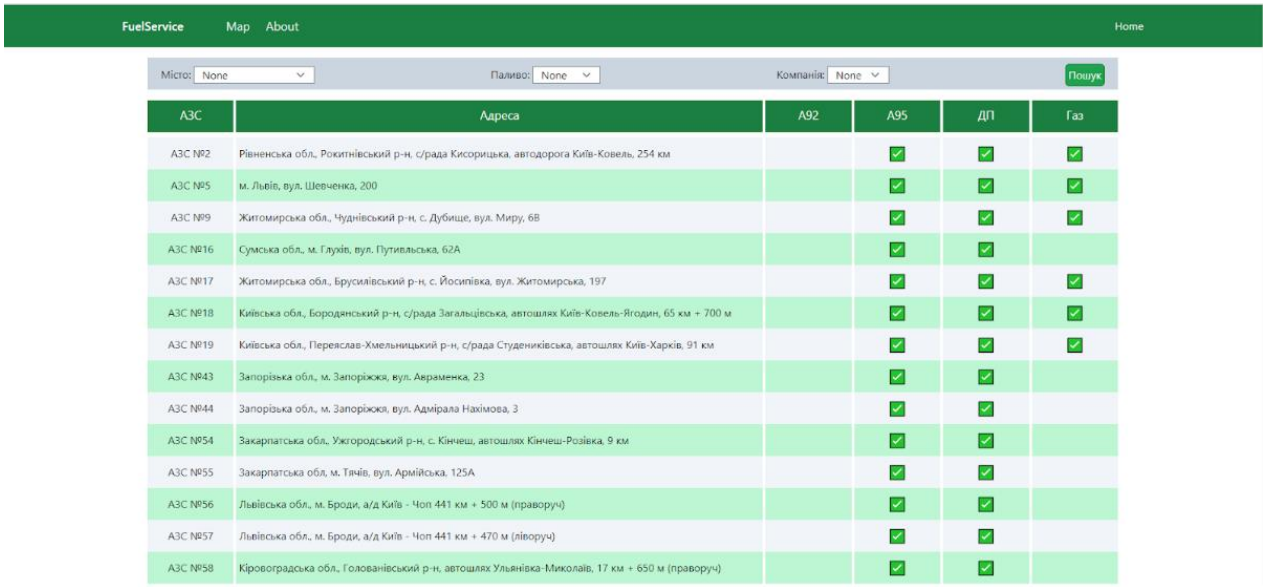


Рис. 3.13. Головна сторінка застосунку.

Також у застосунку доступна карта України з точками АЗС з доступним пальним (Рис. 3.14).

Таблиця складається з наступних колонок:

- АЗС - назва автозаправної станції.
- Адреса - відповідно адреса автозаправної станції.
- A92, A95, ДП, Газ - основні типи пального, показують наявність даного типу пального на автозаправній станції.

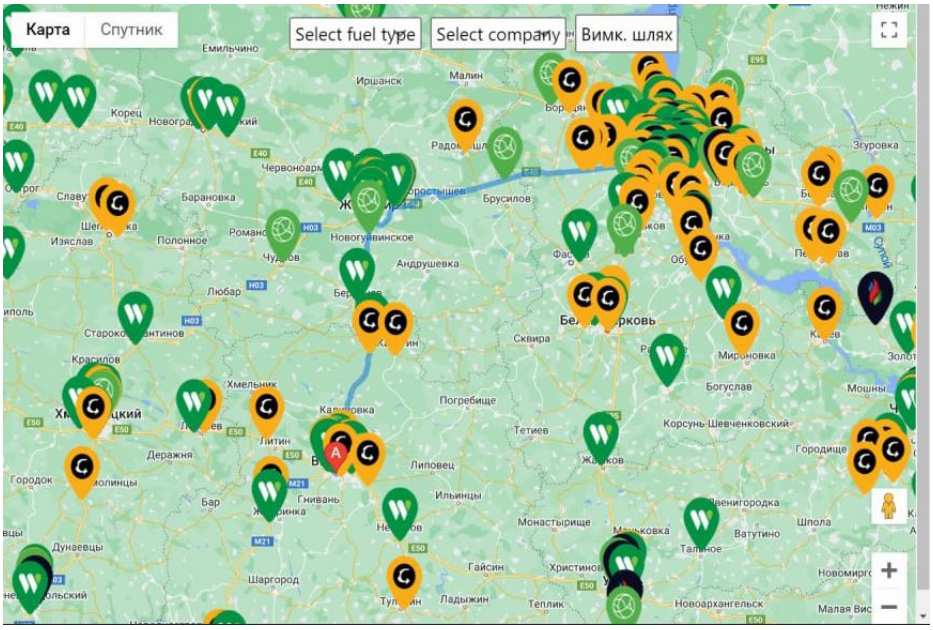


Рис. 3.14. Карта з АЗС.

Також для користувача доступна функція фільтрації. Основні параметри фільтрації:

Місто

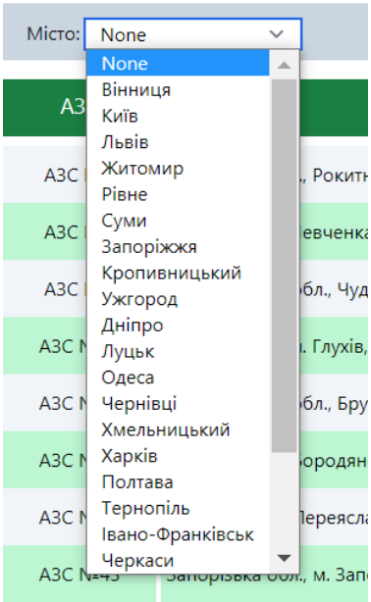


Рис. 3.15. Фільтр по місту.

Для фільтрації по місту доступні всі обласні центри України. Фільтрація по місту не включає в себе фільтрацію по містам області, лише безпосередньо по обласному центру.

Паливо

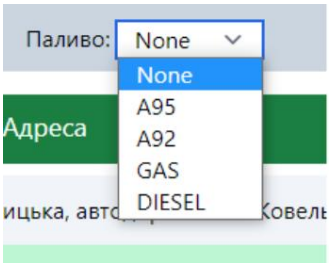


Рис. 3.16. Фільтр по типу палива

Доступні типи палива можна побачити на Рис. 3.16. У кожної паливної компанії можуть бути власні додаткові типи палива, вони логічно співвідносяться вищезгаданим чотирьом типам та можуть бути доступними до перегляду у вікні додаткової інформації АЗС.

Компанія

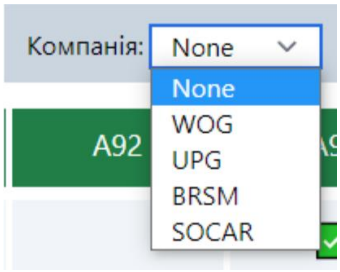


Рис. 3.17. Фільтр по компанії

Серед фільтрів по компанії – усі мережі АЗС, що розглядаються в дипломній роботі. Компанія включається в назву АЗС у відфільтрованій таблиці.

Сервіс

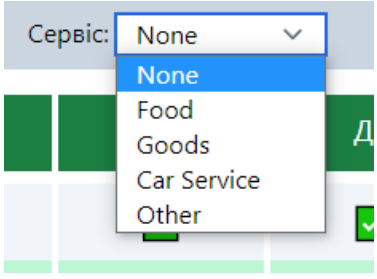


Рис. 3.18. Фільтр по сервісу

Параметри фільтрації по сервісу включають в себе пункти вказані на Рис. 3.18. Мережі АЗС мають безліч різних сервісів, але усі вони були структуровані у 4 типи.

Приклад користувацької фільтрації у таблиці.

Місто: Львів Паливо: DIESEL Компанія: WOG Пошук

АЗС	Адреса	А92	А95	ДП	Газ
WOG 870	м. Львів, вул. Богданівська, 11		✓	✓	✓
WOG 871	м.Львів, вул.Пасічна. 144		✓	✓	
WOG 873	Пустомитівський р-н, с. Сокільники, вул. Стрийська,12		✓	✓	✓
WOG 959	м. Львів, пр. В.Чорновола,10		✓	✓	
WOG 961	м.Львів, вул.Луганська - Стрийська		✓	✓	✓
WOG 962	м.Львів, пер. вул.Б.Хмельницького,192а		✓	✓	
WOG 964	м. Львів, вул. Городоцька, 288		✓	✓	✓
WOG 965	м. Львів, пр. В.Чорновола,107 "Гама-Хім"		✓	✓	
WOG 968	м. Львів, вул. Кузневича, 5		✓	✓	✓
WOG 969	501 км траси Київ-Чоп (мотель "Катерина")		✓	✓	
WOG 1014	м. Львів, вул. В. Стуса, 57-а		✓	✓	
WOG 1097	м. Львів, вул. Мазепи, 7		✓	✓	✓

Рис. 3.19. АЗС відфільтровані по місту “Львів”, типу палива “ДП” та компанії “WOG”

Приклад користувацької фільтрації на карті.

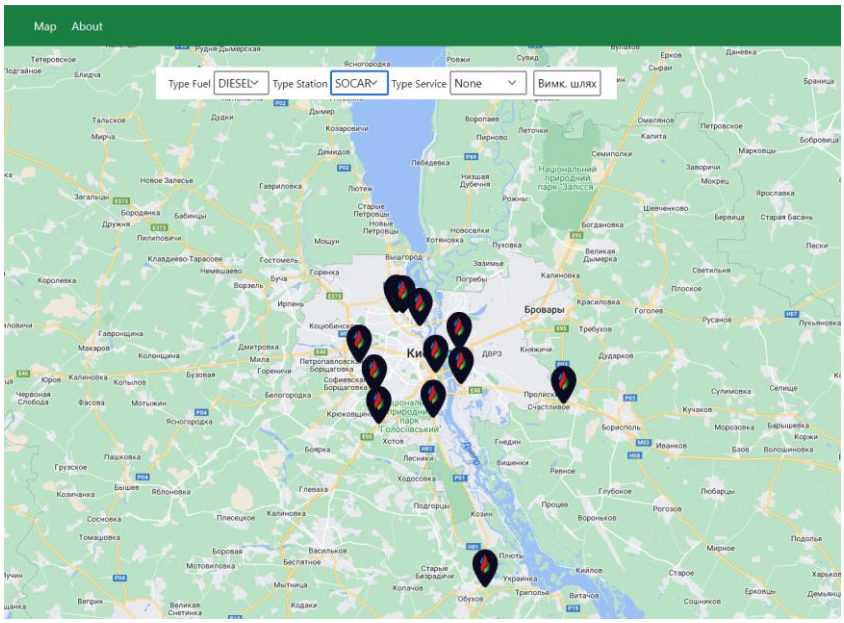


Рис. 3.20. АЗС відфільтровані по типу палива “ДП” та компанії “SOCAR”

Окрім того, користувачу доступна функція фільтрації АЗС по ціні. Для її застосування необхідно лише вказати тип пального, який цікавить.

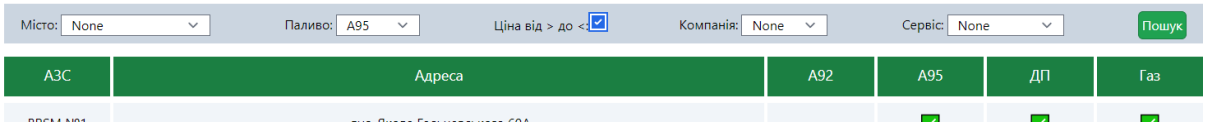


Рис. 3.21. Кнопка фільтрації АЗС по ціні

Також користувачу доступна детальна інформація по кожній автозаправній станції як на карті, так і в таблиці.

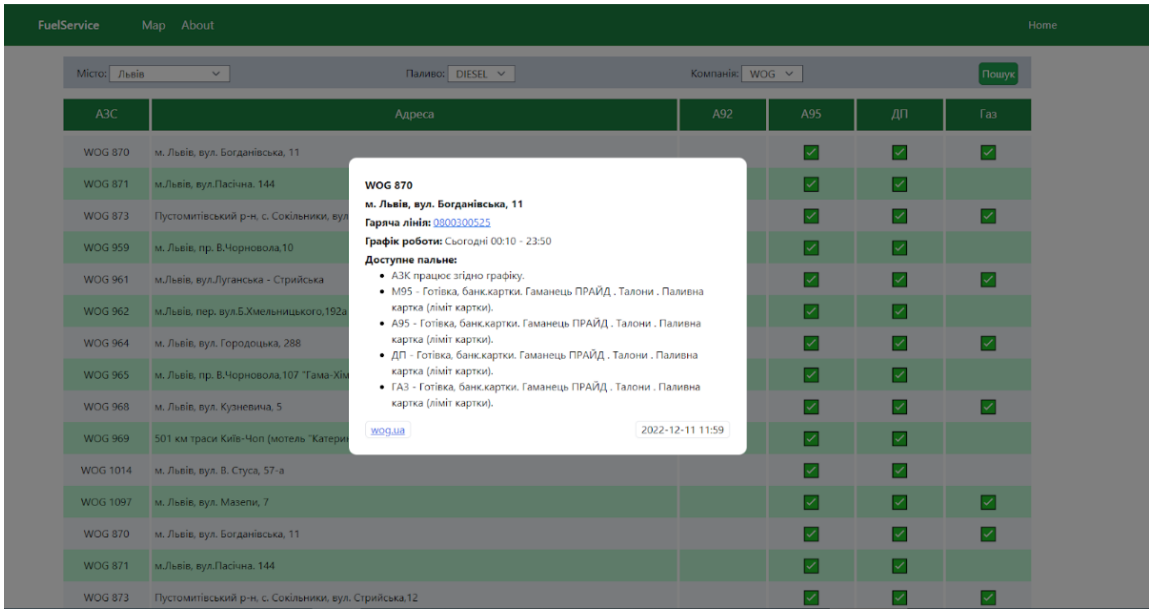


Рис. 3.22. Детальна інформація по обраній АЗС в таблиці

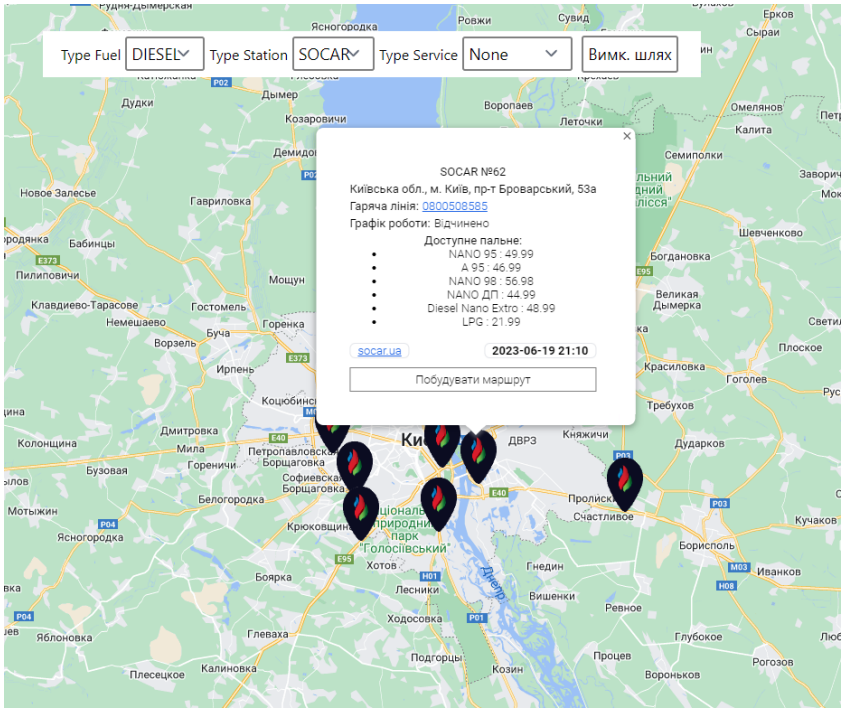


Рис. 3.23. Детальна інформація по обраній АЗС на карті

У вікні детальної інформації про станцію користувачу доступні наступні дані:

- Назва АЗС.
- Адреса.
- Гаряча лінія - номер телефону мережі, на яку можна звернутись за додатковими питаннями.
- Графік роботи станції - якщо дана інформація доступна по АЗС, показує робочі години станції або ж чи доступна вона зараз.
- Доступне пальне - інформація про доступне пальне на даній АЗС, яка включає перелік всіх типів пального, що є в наявності, а також, в залежності від мережі станції, ціну або ж можливі способи оплати.
- Додаткова інформація - якщо така доступна по АЗС, описує особливості роботи станції.
- Сайт компанії.
- Дата та час останнього оновлення інформації по цій станції.

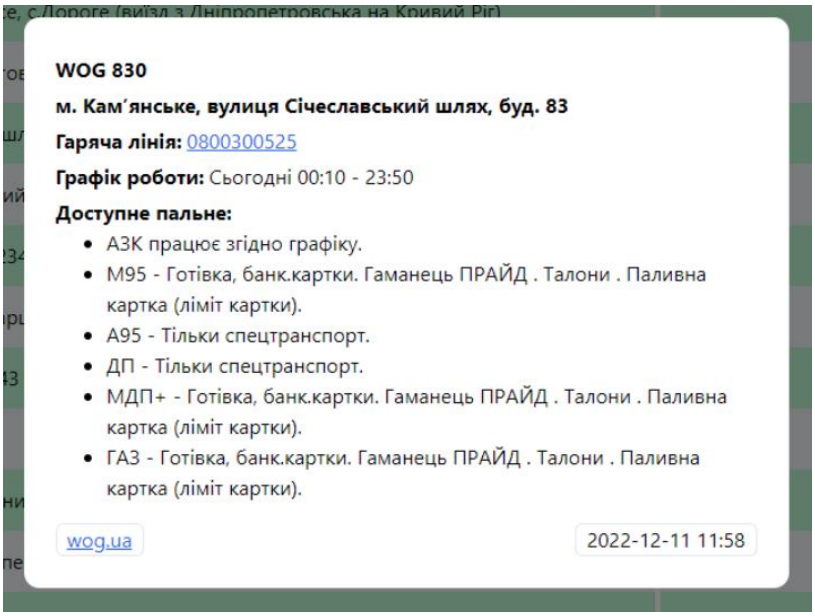


Рис. 3.24. Детальна інформація по станції WOG

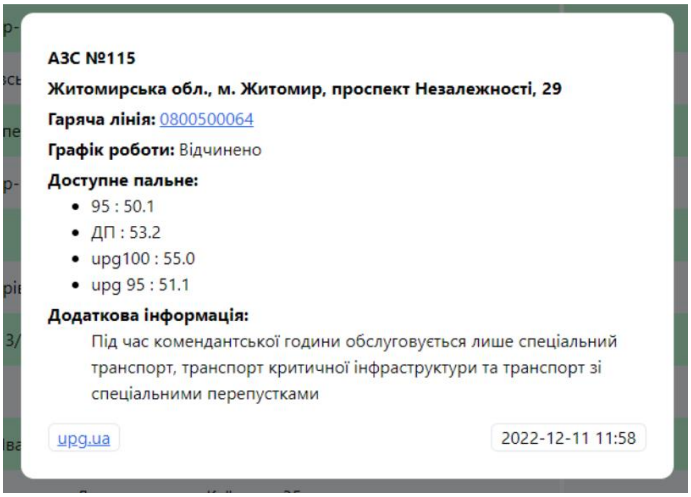


Рис. 3.25. Детальна інформація по станції UPG

Рис. 3.26. Детальна інформація по станції SOCAR

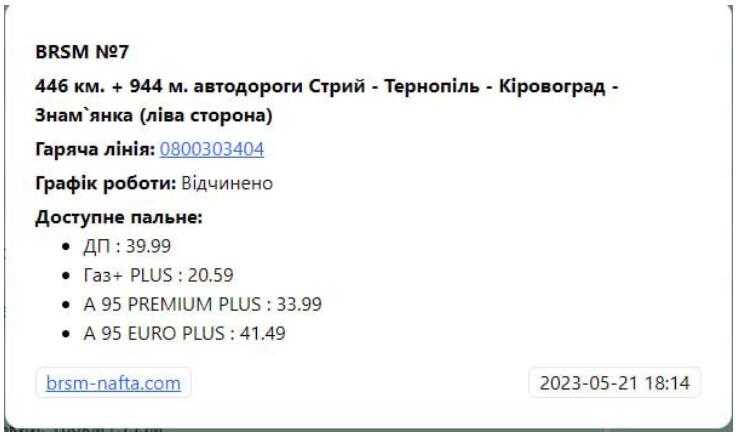


Рис. 3.27. Детальна інформація по станції BRSM

Також у додатку є можливість побудови маршруту до обраної АЗС від поточного місцезнаходження, що визначається геолокацією. За бажанням, його можна вимкнути, нажавши кнопку “Вимк. шлях”.

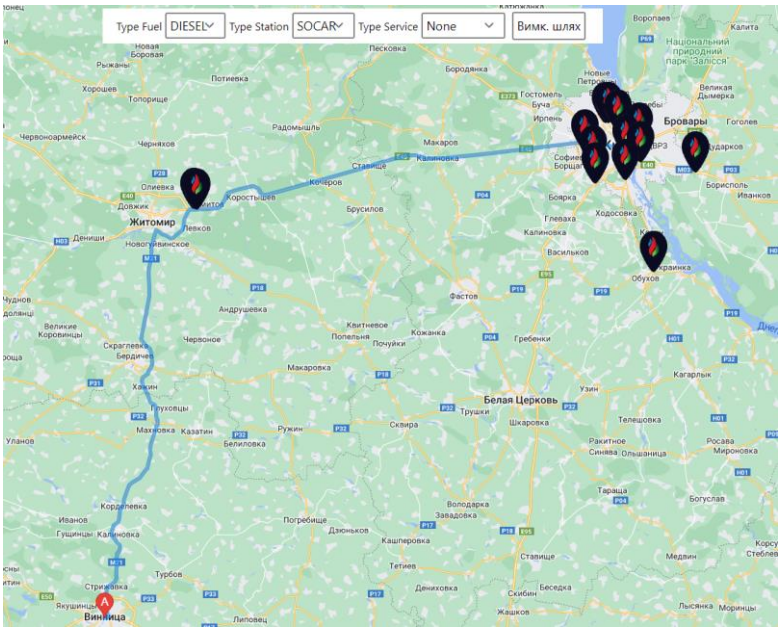


Рис. 3.28. Відбудований маршрут до обраної АЗС

3.4. Опис та аналіз результатів тестових прикладів роботи застосунку

Для опису та аналізу функціональних та нефункціональних особливостей системи, було обрано тестування чорного ящика.

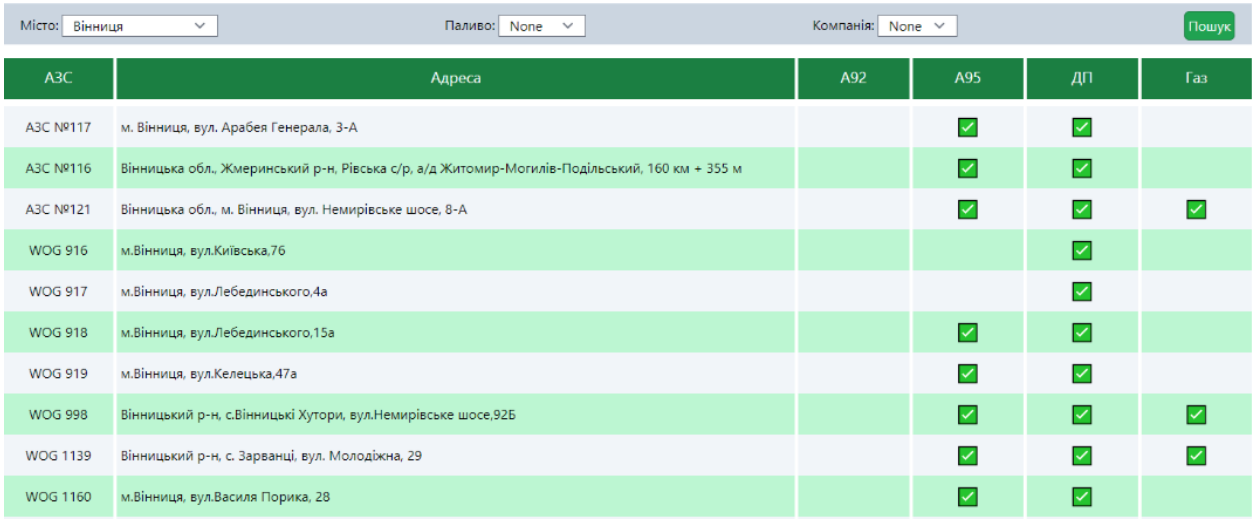
Тест 1. Прегляд АЗС без фільрації.

АЗС	Адреса	А92	А95	ДП	Газ
АЗС №2	Рівненська обл., Рокитнівський р-н, с/рада Кисорицька, автодорога Київ-Ковель, 254 км		✓	✓	✓
АЗС №5	м. Львів, вул. Шевченка, 200		✓	✓	✓
АЗС №9	Житомирська обл., Чуднівський р-н, с. Дубище, вул. Миру, 6В		✓	✓	✓
АЗС №16	Сумська обл., м. Глухів, вул. Путивльська, 62А		✓	✓	
АЗС №17	Житомирська обл., Брусилівський р-н, с. Йосипівка, вул. Житомирська, 197		✓	✓	✓
АЗС №18	Київська обл., Бородянський р-н, с/рада Загальцівська, автошлях Київ-Ковель-Ягодин, 65 км + 700 м		✓	✓	✓
АЗС №19	Київська обл., Переяслав-Хмельницький р-н, с/рада Студенниківська, автошлях Київ-Харків, 91 км		✓	✓	✓
АЗС №43	Запорізька обл., м. Запоріжжя, вул. Авраменка, 23		✓	✓	
АЗС №44	Запорізька обл., м. Запоріжжя, вул. Адмірала Нахімова, 3		✓	✓	
АЗС №54	Закарпатська обл., Ужгородський р-н, с. Кінчеш, автошлях Кінчеш-Розівка, 9 км		✓	✓	
АЗС №55	Закарпатська обл, м. Тячів, вул. Армійська, 125А		✓	✓	
АЗС №56	Львівська обл., м. Броди, а/д Київ - Чоп 441 км + 500 м (праворуч)		✓	✓	
АЗС №57	Львівська обл., м. Броди, а/д Київ - Чоп 441 км + 470 м (ліворуч)		✓	✓	
АЗС №58	Кіровоградська обл., Голованівський р-н, автошлях Ульянівка-Миколаїв, 17 км + 650 м (праворуч)		✓	✓	
АЗС №59	Кіровоградська обл., Голованівський р-н, автошлях Ульянівка-Миколаїв, 17 км + 735 м (ліворуч)		✓	✓	

Рис. 3.29. Результат виконання Тесту 1

Після відпрацювання фільтрації на сторінці доступна інформація по всім АЗС усіх чотирьох мереж, що розглядаються в дипломній роботі, що мають хоча б один доступний тип палива.

Тест 2. АЗС відфільтровані по місту “Вінниця”.



АЗС	Адреса	А92	А95	ДП	Газ
АЗС №117	м. Вінниця, вул. Арабея Генерала, 3-А		✓	✓	
АЗС №116	Вінницька обл., Жмеринський р-н, Рівська с/р, а/д Житомир-Могилів-Подільський, 160 км + 355 м		✓	✓	
АЗС №121	Вінницька обл., м. Вінниця, вул. Немирівське шосе, 8-А		✓	✓	✓
WOG 916	м.Вінниця, вул.Київська,76			✓	
WOG 917	м.Вінниця, вул.Лебединського,4а			✓	
WOG 918	м.Вінниця, вул.Лебединського,15а		✓	✓	
WOG 919	м.Вінниця, вул.Келецька,47а		✓	✓	
WOG 998	Вінницький р-н, с.Вінницькі Хутори, вул.Немирівське шосе,92Б		✓	✓	✓
WOG 1139	Вінницький р-н, с. Зарванці, вул. Молодіжна, 29		✓	✓	✓
WOG 1160	м.Вінниця, вул.Василя Порика, 28		✓	✓	

Рис. 3.30. Результат виконання Тесту 2

Після відпрацювання фільтрації на сторінці доступна інформація лише про АЗС, що знаходяться у Вінниці та мають в наявності хоча б один доступний тип палива.

Тест 3. АЗС відфільтровані по наявності типу палива “Газ”.

АЗС	Адреса	А92	А95	ДП	Газ
АЗС №2	Рівненська обл., Рокитнівський р-н, с/рада Кисорицька, автодорога Київ-Ковель, 254 км		✓	✓	✓
АЗС №5	м. Львів, вул. Шевченка, 200		✓	✓	✓
АЗС №9	Житомирська обл., Чуднівський р-н, с. Дубище, вул. Миру, 6В		✓	✓	✓
АЗС №17	Житомирська обл., Брусилівський р-н, с. Йосипівка, вул. Житомирська, 197		✓	✓	✓
АЗС №18	Київська обл., Бородянський р-н, с/рада Загальцівська, автошлях Київ-Ковель-Ягодин, 65 км + 700 м		✓	✓	✓
АЗС №19	Київська обл., Переяслав-Хмельницький р-н, с/рада Студениківська, автошлях Київ-Харків, 91 км		✓	✓	✓
АЗС №60	Дніпропетровська обл., м. Кам'янське, Єлизаветівське шосе, 27		✓	✓	✓
АЗС №73	Житомирська обл., м. Коростень, вул. Гастелло, 3		✓	✓	✓
АЗС №74	Житомирська обл., м. Коростень, вул. Жовтнева, 11А		✓	✓	✓
АЗС №4	Львівська обл., Пустомитівський р-н, с. Підберізці, вул. Галицька, 1		✓	✓	✓
АЗС №81	Одеська обл., Овідіопольский р-н, смт Авангард, вул. Ангарська, 4		✓	✓	✓
АЗС №89	Київська обл., м. Буча, вул. Шевченка, 4А		✓	✓	✓
АЗС №91	Житомирська обл., Коростенський р-н, Кожухівська с/р, а/д М-07 Київ-Ковель-Ягодин, км154-540 ліворуч		✓	✓	✓
АЗС №90	Київська обл., Києво-Святошинський р-н, а/д Т 10-27 Київське півкільце, 12+363-12+552 км (праворуч)		✓	✓	✓
АЗС №92	Житомирська обл., с. Сінгури, вул. Житомирська (вздовж а/д М-21 Житомир-Могилів-Подільський, км. 10)		✓	✓	✓

Рис. 3.31. Результат виконання Тесту 3

Після відпрацювання фільтрації на сторінці доступна інформація про усім АЗС, що мають в наявності тип палива “Газ”.

Тест 4. АЗС відфільтровані по компанії “WOG”.

АЗС	Адреса	А92	А95	ДП	Газ
WOG 807	Вінницька обл., м.Могилів-Подільський, вул.Пушкіна, 74			☑	
WOG 808	м.Хмельницький, вул.Вінницька, 2/2		☑	☑	☑
WOG 809	м.Харків, вл.Шевченка, 41		☑	☑	
WOG 811	м.Київ, Кільцева дорога 9		☑	☑	☑
WOG 812	Полтавський р-н, с.Говтвянчик, вул.Київське шосе, 1 + Restorio		☑	☑	☑
WOG 813	м.Київ, вул. Дніпровська Набережна, 17В		☑	☑	☑
WOG 814	м.Київ, вул. Стеценка, 21		☑	☑	☑
WOG 816	м. Миколаїв, вул. Одеське шосе, 116 , а/д М-14 Одеса-Миколаїв-Новоазовськ , 127 км , ліворуч		☑	☑	☑
WOG 817	м. Запоріжжя, вул. Донецьке шосе, 7		☑	☑	☑
WOG 818	м. Запоріжжя, вул. Магістральна, 100б, біля заводу "АЙС"		☑	☑	☑
WOG 823	м. Запоріжжя, пр. Ювілейний, 36		☑	☑	☑
WOG 824	Запорізький р-н, с. Сонячне, вул. Паркова, 1		☑	☑	☑
WOG 825	м. Запоріжжя, вул. Дослідна станція, 2в		☑	☑	☑
WOG 826	м. Дніпропетровськ, вул.Панікахи (в районі буд. №91)		☑	☑	☑
WOG 827	м. Дніпропетровськ, Донецьке шосе (територія Ювілейної с/р, виїзд на Підгородне)		☑	☑	☑

Рис. 3.32. Результат виконання Тесту 4

Після відпрацювання фільтрації на сторінці доступна інформація про усім АЗС, що мають, що належать компанії “WOG”.

Тест 5. АЗС відфільтровані по місту “Житомир” та типу палива “ДП”.

АЗС	Адреса	А92	А95	ДП	Газ
АЗС №9	Житомирська обл., Чуднівський р-н, с. Дубище, вул. Миру, 6В		☑	☑	☑
АЗС №17	Житомирська обл., Брусилівський р-н, с. Йосипівка, вул. Житомирська, 197		☑	☑	☑
АЗС №53	Житомирська обл., м. Олевськ, вул. Київська, 7А		☑	☑	
АЗС №73	Житомирська обл., м. Коростень, вул. Гастелло, 3		☑	☑	☑
АЗС №74	Житомирська обл., м. Коростень, вул. Жовтнева, 11А		☑	☑	☑
АЗС №91	Житомирська обл., Коростенський р-н, Кожухівська с/р, а/д М-07 Київ-Ковель-Ягодин, км154-540 ліворуч		☑	☑	☑
АЗС №92	Житомирська обл., с. Сінгури, вул. Житомирська (вздовж а/д М-21 Житомир-Могилів-Подільський, км. 10)		☑	☑	☑
АЗС №115	Житомирська обл., м. Житомир, проспект Незалежності, 29		☑	☑	
WOG 947	Житомирський р-н, с. Зарічани, вул. Майдан Бердичевський, 8		☑	☑	☑
WOG 948	Житомирський р-н, с. Зарічани, вул. Бердичевське шосе, 3а		☑	☑	☑
WOG 950	м. Житомир, проспект Незалежності, буд. 63		☑	☑	☑
WOG 951	м. Житомир, вул. Вітрука, 1			☑	☑
WOG 972	м. Житомир, вулиця Покровська, буд. 269 (с.Оліївка)		☑	☑	☑
WOG 1010	м. Житомир, м-н Визволення 11		☑	☑	☑
WOG 1188	м.Житомир, вул. Чуднівська, 104		☑	☑	☑

Рис. 3.33. Результат виконання Тесту 5

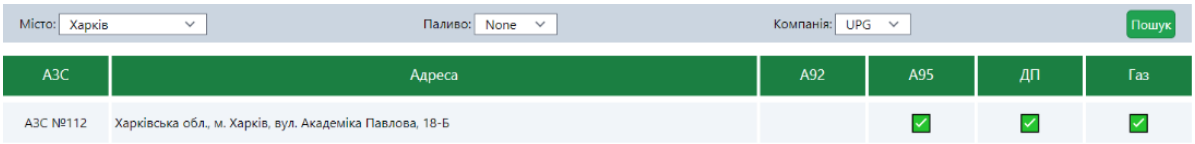
Після відпрацювання фільтрації на сторінці доступна інформація по всім АЗС, що мають в наявності дизель-паливо та розсташовані вв Житомирі.

Тест 6. АЗС відфільтровані по типу палива “A95” та компанією “UPG”.

АЗС	Адреса	A92	A95	ДП	Газ
АЗС №2	Рівненська обл., Рокитнівський р-н, с/рада Кисорицька, автодорога Київ-Ковель, 254 км		✓	✓	✓
АЗС №5	м. Львів, вул. Шевченка, 200		✓	✓	✓
АЗС №9	Житомирська обл., Чуднівський р-н, с. Дубище, вул. Миру, 6В		✓	✓	✓
АЗС №16	Сумська обл., м. Глухів, вул. Путивльська, 62А		✓	✓	
АЗС №17	Житомирська обл., Брусилівський р-н, с. Йосипівка, вул. Житомирська, 197		✓	✓	✓
АЗС №18	Київська обл., Бородянський р-н, с/рада Загальцівська, автошлях Київ-Ковель-Ягодин, 65 км + 700 м		✓	✓	✓
АЗС №19	Київська обл., Переяслав-Хмельницький р-н, с/рада Студениківська, автошлях Київ-Харків, 91 км		✓	✓	✓
АЗС №43	Запорізька обл., м. Запоріжжя, вул. Авраменка, 23		✓	✓	
АЗС №44	Запорізька обл., м. Запоріжжя, вул. Адмірала Нахімова, 3		✓	✓	
АЗС №54	Закарпатська обл., Ужгородський р-н, с. Кінчеш, автошлях Кінчеш-Розівка, 9 км		✓	✓	
АЗС №55	Закарпатська обл, м. Тячів, вул. Армійська, 125А		✓	✓	
АЗС №56	Львівська обл., м. Броди, а/д Київ - Чоп 441 км + 500 м (праворуч)		✓	✓	
АЗС №57	Львівська обл., м. Броди, а/д Київ - Чоп 441 км + 470 м (ліворуч)		✓	✓	
АЗС №58	Кіровоградська обл., Голованівський р-н, автошлях Ульянівка-Миколаїв, 17 км + 650 м (праворуч)		✓	✓	
АЗС №59	Кіровоградська обл., Голованівський р-н, автошлях Ульянівка-Миколаїв, 17 км + 735 м (ліворуч)		✓	✓	

Рис. 3.34. Результат виконання Тесту 6

Тест 7. АЗС відфільтровані по місту “Харків” та компанії “UPG”.



АЗС	Адреса	A92	A95	ДП	Газ
АЗС №112	Харківська обл., м. Харків, вул. Академіка Павлова, 18-Б		✓	✓	✓

Рис. 3.35. Результат виконання Тесту 7

Після відпрацювання фільтрації на сторінці доступна інформація по всім АЗС мережі “UPG”, що розсташовані в Харкові, що мають хоча б один доступний тип палива.

Тест 8. АЗС відфільтровані по місту “Львів”, типу палива “ДП” та компанії “WOG”.

Рис. 3.36. Результат виконання Тесту 8

Після відпрацювання фільтрації на сторінці доступна інформація по всім АЗС мережі “WOG”, що розсташовані у Львові, що мають в наявності дизель-паливо.

Висновок до третього розділу

Під час написання програмної реалізації проекту було проаналізовано та підібрано стек технологій, що забезпечують виконання функціональних та нефункціональних вимог висунутих до застосунку, проведено аналітичний огляд доступних ресурсів мереж АЗС та пошук способів програмного добуття їх даних. Також було розроблено серверну та клієнтську частини застосунку, протестовано їх взаємодію та коректність роботи. Для відображення логічної структури коду серверної частини було створено діаграму класів, описано практичне призначення основних класів та їх взаємодію. Для демонстрації роботи готового програмного продукту було створено керівництво користувача з детальним описом усіх можливостей застосунку. А також було роз’яснено як працює кожна програмна частина застосунку та як вони взаємодіють між собою.

ВИСНОВКИ

В рамках виконання дипломної роботи було розроблено додаток для пошуку пального. Відповідно до завдань на дипломну роботу, за допомогою застосунку було узагальнено дані, що є доступними по 2 найбільших мережах АЗС України “WOG”, “UPG”, “SOCAR” та “BRSM”. Під час розробки програми було спроектовано гнучку, легкомасштабовану архітектуру серверної частини, що дає змогу додавати нові модулі та функціонал без значних зусиль, розроблено гнучку модель даних, що узагальнює інформацію за різними доменними моделями компаній. Також під час розробки було застосовано технологію скрейпінгу. І на фінальному етапі було перевірено та доведено роботоспроможність актуальність та доцільність реалізованого підходу.

Отриманий застосунок може набути широкого вжиття у повсякденному житті звичайних українців. Це практично дозволить людям скоротити свій витрачений час на пошуки потрібного пального в умовах паніки або дефіциту.

Застосунок можна розширити низкою додаткового функціоналу, зокрема можна розширити список компаній, чиї дані обробляються, додати мапу із розміщеними на ній точками АЗС, додати нові фільтри або ж підтримку зворотнього зв’язку із системою відгуків клієнтів про конкретні АЗС та інформацією про наявні там черги.

Підсумовуючи, результатом виконання дипломної роботи є її втілена мета - роботоспроможний та актуальний програмний застосунку для збору та узагальнення інформації щодо наявності пального по найбільшим мережам АЗС України.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Jay M. Patel. Getting Structured Data from the Internet – 2020. – 420 с.
2. Eric Freeman, Elisabeth Robson. Head of First Design Patterns – 2021. – 672 с.
3. Christian Bauer and Gavin King. Java Persistence API with Hibernate – 2017. - 632 с.
4. Craig Walls. Spring Boot in Action - 2016 - 264 с.
5. Artemij Fedosejev. React.js Essentials: A fast-paced guide to designing and building scalable and maintainable web apps with React.js – 2015 – 208 с.
6. Documentation of Jsoup [Електронний ресурс] // Jonathan Hedley. – 2023. – Режим доступу до ресурсу: https://jsoup.org/
7. Телеграм-бот “Kyiv_Diesel” [Електронний ресурс] // 2023. – Режим доступу до ресурсу: https://t.me/Kyiv_Diesel
8. Телеграм-бот “petrol_alert_bot” [Електронний ресурс] // 2023. – Режим доступу до ресурсу: https://t.me/petrol_alert_bot
9. Телеграм-бот “Kyiv_AZS” [Електронний ресурс] // 2023. – Режим доступу до ресурсу: https://t.me/Kyiv_AZS
10. Nextjs documentation [Електронний ресурс] // Vercel. – 2023. – Режим доступу до ресурсу: https://nextjs.org/docs/
11. Web Scraping [Електронний ресурс] // SysNucleus. – 2023. – Режим доступу до ресурсу: https://www.webharvy.com/articles/what-is-web-scraping.html.
12. Web Scraping Guide [Електронний ресурс] // ParseHub. – 2023. – Режим доступу до ресурсу: parsehub.com/blog/what-is-web-scraping/
13. Ryan Mitchell. Instant Web Scraping with Java - Packt Publishing. – 2013. – 72 с. – ISBN 978-1849696883
14. React documentation [Електронний ресурс] // Meta Platforms. – 2023. – Режим доступу до ресурсу: https://uk.legacy.reactjs.org/docs/getting-started.html

ДОДАТОК

Програмний код основних класів серверної частини

```
@RestController
@CrossOrigin
public class StationController {
    private FilterService filterService;
    private StationService stationService;
    private StationMapper stationMapper;

    public StationController(FilterService filterService, StationService stationService, StationMapper stationMapper) {
        this.filterService = filterService;
        this.stationService = stationService;
        this.stationMapper = stationMapper;
    }

    @SneakyThrows
    @PostMapping(value = "/station/filter")
    public List<StationLine> getStationsByFilter(@RequestBody Filter filters) {
        return stationMapper.mapStationToStationLine(filterService.filter(filters));
    }

    @GetMapping(value = "/station")
    public List<StationLine> getAllStations() {
        return stationMapper.mapStationToStationLine(stationService.getAllStations());
    }

    @GetMapping(value = "/station/details/{stationId}")
    public StationDetails getStationInfo(@PathVariable Long stationId) {
        Station station = stationService.getStation(stationId);

        StationDetails stationDetails = stationMapper.mapStationToStationDetails(station);

        return stationDetails;
    }
}
@Component
```

```
public class StationMapper {
    public List<StationLine> mapStationToStationLine(List<Station> stations) {
        return stations.stream()
                .map(station -> {
                    return StationLine.builder()
                            .id(station.getId())
                            .lat(station.getLat())
                            .lng(station.getLng())
                            .city(station.getCity())
                            .address(station.getAddress())
                            .companyName(station.getCompany().getName())
                            .stationInfoId(station.getStationInfo().getId())
                            .name(station.getName())
                            .fuelTypesAvailableInfo(mapFuelTypesAvailableInfo(station))
                            .servicesNames(mapServicesNames(station))
                            .build();
                }).collect(Collectors.toList());
    }

    public StationDetails mapStationToStationDetails(Station station) {
        return StationDetails.builder()
                .id(station.getId())
                .name(station.getName())
                .address(station.getAddress())
                .hotLine(station.getCompany().getHotLine())
                .schedule(station.getStationInfo().getSchedule())
                .availableFuelsDescription(mapDescription(station))
                .additionalInfo(mapAdditionalInfo(station))
                .link(station.getCompany().getMapLink())
                .linkName(station.getCompany().getLinkName())
                .lastUpdate(station.getStationInfo().getLastUpdate())
                .build();
    }

    private String mapDescription(Station station) {
        if (station.getCompany().getName().equals("WOG")) {
            return station.getStationInfo().getWorkDescription();
```

```
    }

    StringBuilder stringBuilder = new StringBuilder();

    station.getStationInfo().getFuels()
            .stream()
            .filter(Fuel::isAvailable)
            .forEach(fuel -> {
                stringBuilder.append(fuel.getName() + " : " + fuel.getPrice());
                stringBuilder.append("\n");
            });
    return stringBuilder.toString();
}

private String mapAdditionalInfo(Station station) {
    if(station.getCompany().getName().equals("UPG")) {
        return station.getStationInfo().getWorkDescription();
    }

    return "";
}

private Map<String, Boolean> mapFuelTypesAvailableInfo(Station station) {
    List<Fuel> fuels = station.getStationInfo().getFuels();

    Map<String, Boolean> fuelTypesAvailableInfo = Arrays.stream(Fuel.FuelType.values())
            .filter(fuelType -> fuelType != Fuel.FuelType.UNKNOWN)
            .collect(Collectors.toMap(Enum::name, fuelType -> {
                return fuels.stream()
                        .filter(fuel -> fuel.getFuelType() == fuelType).anyMatch(Fuel::isAvailable);
            }));

    return fuelTypesAvailableInfo;
}

private List<String> mapServicesNames(Station station) {
    return station.getStationInfo()
```

```
                .getServices()
                .stream()
                .map(Service::getName)
                .collect(Collectors.toList());
    }
}


@Component
public class Saver {
    private final List<Adapter> adapters;
    private final StationRepository stationRepository;

    public Saver(List<Adapter> adapters, StationRepository stationRepository) {
        this.adapters = adapters;
        this.stationRepository = stationRepository;
    }
    @Scheduled(fixedDelay = 1000000)
    void saveInfo() {
        adapters.forEach(adapter -> {
            List<Station> stations = adapter.collectInfo();
            if (!stations.isEmpty()) {
                stationRepository.saveAll(stations);
            }
        });
    }
}

@Slf4j
@EnableCaching
@AllArgsConstructor
@Component("wogAdapter")
public class WogAdapter implements Adapter {

    private WogScrapper wogScrapper;
    private ObjectMapper objectMapper;
    private WogMapper wogMapper;
```

```
@Override
public List<Station> collectInfo() {
    List<WogFuelFilter> allWogFuelFilters = getAllWogFuelFilters();
    wogMapper.setFuelFilters(allWogFuelFilters);

    List<WogStation> allStations = getAllWogStations();

    return allStations.stream()
        .map(wogStation -> getWogStationInfo(wogStation.getLink()))
        .map(wogStationInfo -> wogMapper.mapStation(wogStationInfo))
        .collect(Collectors.toList());
}

@Override
public List<Station> recoverCollectInfo(Exception e, String sql) {
    log.error("Wog adapter wasn't process data correctly");
    return Adapter.super.recoverCollectInfo(e, sql);
}

@SneakyThrows
private List<WogStation> getAllWogStations() {
    Object allStationsInfo = wogScrapper.retrieveStations();

    return objectMapper.convertValue(allStationsInfo, new TypeReference<>() {});
}

@SneakyThrows
private List<WogFuelFilter> getAllWogFuelFilters() {
    Object allFuelFiltersInfo = wogScrapper.retrieveFuelFilters();

    return objectMapper.convertValue(allFuelFiltersInfo, new TypeReference<>() {});
}

@SneakyThrows
public WogStationInfo getWogStationInfo(String stationLink) {
    Object stationInfo = wogScrapper.retrieveStationInfo(stationLink);
```

```
        return objectMapper.convertValue(stationInfo, new TypeReference<>() {});
    }
}

@Component("wogScrapper")
public class WogScrapperImpl implements WogScrapper {

    @Value("${station.links.wog}")
    private String WOG_STATIONS_URL;
    @Autowired
    private RestTemplate restTemplate;
    @Autowired
    private ObjectMapper objectMapper;

    @Override
    public Object retrieveStations() {
        ResponseEntity<Object> rawResponse = restTemplate.getForEntity(WOG_STATIONS_URL, Object.class);

        return parseRowResponseToAllStationsResponse(rawResponse.getBody());
    }

    @Override
    public Object retrieveStationInfo(final String station_url) {
        Object rawResponse = restTemplate.getForObject(station_url, Object.class);

        return parseRowResponseToStationInfoResponse(rawResponse);
    }

    @Override
    public Object retrieveFuelFilters() {
        Object rawResponse = restTemplate.getForObject(WOG_STATIONS_URL, Object.class);

        return parseRowResponseToAllFuelFiltersResponse(rawResponse);
    }

    @SneakyThrows
```

```
    private Object parseRowResponseToAllStationsResponse(Object rawResponse) {
        Object dataResponse = getObject(rawResponse, "data");

        return getObject(dataResponse, "stations");
    }

    @SneakyThrows
    private Object parseRowResponseToStationInfoResponse(Object rawResponse) {

        return getObject(rawResponse, "data");
    }

    @SneakyThrows
    private Object parseRowResponseToAllFuelFiltersResponse(Object rawResponse) {
        Object dataResponse = getObject(rawResponse, "data");

        Object allFuelFiltersResponse = getObject(dataResponse, "fuel_filters");

        return allFuelFiltersResponse;
    }

    @SneakyThrows
    private Object getObject(Object objFromJson, String fieldInJson) {
        String json = objectMapper.writeValueAsString(objFromJson);

        Map<String, Object> objectMap = objectMapper.readValue(json, new TypeReference<>(){});

        return objectMap.get(fieldInJson);
    }
}

@Component
@Data
public class WogMapper {
    private static final String UNAVAILABLE_FUEL_PATTERN = "Пальне відсутнє";
    private static final DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm");
```

```
    private Company wogCompany;
    private CompanyRepository companyRepository;
    private List<WogFuelFilter> fuelFilters;
    private WogShortNamesMap wogShortNamesMap;
    private WogServiceMap wogServiceMap;
    public static final String BRAND_NAME = "WOG";

    public WogMapper(CompanyRepository companyRepository, WogShortNamesMap wogShortNamesMap,
WogServiceMap wogServiceMap) {
        this.companyRepository = companyRepository;
        this.wogShortNamesMap = wogShortNamesMap;
        this.wogServiceMap = wogServiceMap;
    }

    public Station mapStation(WogStationInfo wogStationInfo) {
        return Station.builder()
                .city(wogStationInfo.getCity())
                .name(mapStationName(wogStationInfo.getId()))
                .address(wogStationInfo.getName())
                .company(getCompany())
                .lat(String.valueOf(wogStationInfo.getCoordinates().getLatitude()))
                .lng(String.valueOf(wogStationInfo.getCoordinates().getLongitude()))
                .stationInfo(mapStationInfo(wogStationInfo))
                .build();
    }
    StationInfo mapStationInfo(WogStationInfo wogStationInfo) {
        return StationInfo.builder()
                .lastUpdate(LocalDateTime.now().format(formatter))
                .workDescription(wogStationInfo.getWorkDescription())
                .schedule(mapSchedule(wogStationInfo.getSchedule()))
                .fuels(mapFuelList(wogStationInfo.getFuels(), wogStationInfo.getWorkDescription()))
                .services(mapServiceList(wogStationInfo.getServices()))
                .build();
    }

    private List<Service> mapServiceList(List<WogService> services) {
        return services.stream()
```

```
            .map(this::mapService)
            .collect(Collectors.toList());
    }

    private List<Fuel> mapFuelList(List<WogFuel> fuels, String workDescription) {
        return fuels.stream()
            .map(wogFuel -> mapFuel(wogFuel, workDescription))
            .collect(Collectors.toList());
    }



    private String mapSchedule(List<WogSchedule> schedule) {
        return schedule.stream()
            .map(scheduleObj -> scheduleObj.getDay() + " " + scheduleObj.getInterval())
            .findFirst()
            .orElse("");
    }

    private String mapStationName(int id) {
        return BRAND_NAME + " " + id;
    }

    private Service mapService(WogService wogService) {
        return Service.builder()
            .name(wogService.getName())
            .serviceType(mapServiceType(wogService.getIcon()))
            .isAvailable(true)
            .build();
    }

    private Fuel mapFuel(WogFuel wogFuel, String workDescription) {
        return Fuel.builder()
            .name(mapFuelName(wogFuel.getName(), wogFuel.getBrand()))
            .price(mapFuelPrice(getFuelPrice(wogFuel.getId())))
            .fuelType(mapFuelType(wogFuel.getName()))
            .isAvailable(checkIfFuelIsAvailable(workDescription, mapFuelName(wogFuel.getName(),
wogFuel.getBrand())))
```

```
            .build();
    }

    private boolean checkIfFuelIsAvailable(String workDescription, String fuelName) {
        String shortFuelName = this.wogShortNamesMap.getShortNames().get(fuelName);

        int fuelDescriptionStartIndex = workDescription.indexOf(shortFuelName);

        if (fuelDescriptionStartIndex < 0) {
            return false;
        }

        String fuelDescription = workDescription.substring(fuelDescriptionStartIndex);

        int fuelDescriptionEndIndex = fuelDescription.indexOf(".");

        if (fuelDescriptionEndIndex <= shortFuelName.length()) {
            return false;
        }
        String resultDescription = fuelDescription.substring(0, fuelDescriptionEndIndex).trim();
        return !resultDescription.contains(UNAVAILABLE_FUEL_PATTERN);
    }

    private String mapFuelName(String name, String brand) {
        return (brand != null && !brand.isEmpty()) ? name + " " + brand : name;
    }

    private int getFuelPrice(int fuelId) {
        return fuelFilters.stream()
                .filter(fuelFilter -> (fuelId == 11) ? fuelFilter.getId() == 3 : fuelFilter.getId() == fuelId)
                .findFirst()
                .orElseGet(WogFuelFilter::new)
                .getPrice();
    }
    private double mapFuelPrice(int price) {
        if (price == 0) {
            return 0.0;
```

```
    }

    String stringPrice = String.valueOf(price).trim();

    double drobne = Double.parseDouble(stringPrice.substring(stringPrice.length() - 2)) / 100;
    double zile = Double.parseDouble(stringPrice.substring(0, stringPrice.length() - 2));

    double result = zile + drobne;

    return result;
}

FuelType mapFuelType(String name) {
    FuelType fuelType = FuelType.UNKNOWN;

    switch (name) {
        case "92":
            fuelType = FuelType.A92;
            break;
        case "95":
            fuelType = FuelType.A95;
            break;
        case "ДП":
            fuelType = FuelType.DIESEL;
            break;
        case "ГАЗ":
            fuelType = FuelType.GAS;
            break;
        default:
            break;
    }

    return fuelType;
}

private Service.ServiceType mapServiceType(String name) {
    return wogServiceMap.getServiceNameToServiceType().get(name);
```

```
    }

    private Company getCompany() {
        if (this.wogCompany == null) {
            this.wogCompany = this.companyRepository.getByName(BRAND_NAME);
        }

        return this.wogCompany;
    }
}

@Service
public class FilterService {

    @Autowired
    private List<Matcher> matchers;

    @Autowired
    StationRepository stationRepository;

    public List<Station> filter(Filter filter) {

        List<Station> stations = stationRepository.findAll();

        return filter(stations, filter);
    }

    private List<Station> filter(List<Station> stations, Filter filter) {
        return stations.stream().filter(station -> isMatch(station, filter))
            .collect(Collectors.toList());
    }

    private boolean isMatch(Station station, Filter filter) {
        return matchers.stream().allMatch(matcher -> matcher.match(station, filter));
    }
}
@Component
```

```
public class FuelTypeMatcher implements Matcher {

    @Override
    public boolean match(Station station, Filter filter) {
        return filter.getFuelType() == null
                || station.getStationInfo()
                .getFuels()
                .stream()
                .filter(Fuel::isAvailable)
                .map(Fuel::getFuelType)
                .anyMatch(fuelType -> fuelType.equals(filter.getFuelType()));
    }
}

@Data
@Component
public class WogServiceMap {
    private Map<String, Service.ServiceType> serviceNameToServiceType = new HashMap<>();

    {
        serviceNameToServiceType.putAll(Map.of(
                "vilka", Service.ServiceType.OTHER,
                "wogpride", Service.ServiceType.OTHER,
                "carwash", Service.ServiceType.CAR_SERVICE,
                "tirefitting", Service.ServiceType.CAR_SERVICE,
                "parking", Service.ServiceType.CAR_SERVICE,
                "wc", Service.ServiceType.OTHER,
                "wifi", Service.ServiceType.OTHER,
                "wogcafe", Service.ServiceType.FOOD,
                "wogmarket", Service.ServiceType.GOODS));

        serviceNameToServiceType.putAll(Map.of(
                "charge", Service.ServiceType.OTHER,
                "baby", Service.ServiceType.OTHER,
                "rozetka2", Service.ServiceType.GOODS,
                "ibox", Service.ServiceType.OTHER,
                "tire", Service.ServiceType.CAR_SERVICE,
```

```
            "kso", Service.ServiceType.OTHER,
            "logo", Service.ServiceType.CAR_SERVICE,
            "diesel", Service.ServiceType.OTHER
        ));
    }
}
```