МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ВАСИЛЯ СТУСА
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ І ПРИКЛАДНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

УМАНСЬКА АНАСТАСІЯ ВОЛОДИМИРІВНА

Допускається до захисту:
в.о. завідувача кафедри
інформаційних технологій,
д-р. техн. наук, професор
_____Наталія ВЕСЕЛОВСЬКА
«____»_____ 2026 р.

РОЗРОБКА ПРОГРАМНОГО МОДУЛЯ ДЛЯ УПРАВЛІННЯ ЛОГІСТИЧНИМ ОБСЛУГОВУВАННЯМ ПІДПРИЄМСТВА

Спеціальність 122 Комп’ютерні науки

Кваліфікаційна (бакалаврська) робота

Керівник:
Надія ПОТАПОВА, доцент кафедри
інформаційних технологій,
к. е. н., доцент
_____

Оцінка: _____ / _____ / _____
(бали за шкалою ЄКТС / за національною шкалою)

Голова ЕК: _____
(підпис)

Вінниця - 2026

АНОТАЦІЯ

Уманська А. В. Розробка програмного модуля для управління логістичним обслуговуванням підприємства. Спеціальність 122 «Комп'ютерні науки», освітня програма «Комп'ютерні науки». Донецький національний університет імені Василя Стуса, Вінниця 2026.

У кваліфікаційній (бакалаврській) роботі досліджено теоретичні основи логістичного обслуговування підприємства, особливості побудови інформаційних систем у сфері логістики та існуючі програмні рішення для автоматизації логістичних процесів. Розроблено веборієнтований програмний модуль для виконання логістичних розрахунків і підтримки прийняття рішень. Система дозволяє розраховувати оптимальний розмір замовлення, кількість одиниць вантажу, тривалість маршруту, необхідну кількість палива, транспортний тариф і собівартість перевезення, а також забезпечує авторизацію користувачів і збереження історії розрахунків.

Ключові слова: логістичне обслуговування, програмний модуль, логістичні розрахунки, Python, Django, PostgreSQL, вебзастосунок.

64 ст., 29 рис., 3 дод., 60 джерел.

ABSTRACT

Umanska A. Development of a Software Module for Managing the Logistics Service of an Enterprise. Specialty 122 «Computer Science», Educational Program «Computer Science». Vasyl Stus Donetsk National University, Vinnytsia, 2026.

The qualification paper examines the theoretical foundations of enterprise logistics service, the features of building logistics information systems, and existing software solutions for automating logistics processes. A web-based software module has been developed to perform logistics calculations and support decision-making.

The system calculates the optimal order quantity according to the Wilson model, the number of cargo units, route duration, required fuel amount, transport tariff, and transportation cost. User authorization, database interaction, and calculation history storage were also implemented.

Keywords: logistics service, software module, web application, logistics calculations, Python, Django, PostgreSQL.

ЗМІСТ

ВСТУП

Розвиток інформаційних технологій є важливим етапом цифрової трансформації сучасних підприємств, зокрема у сфері логістичного обслуговування. В умовах зростання обсягів перевезень, підвищення вимог до швидкості доставки, необхідності оптимізації витрат і забезпечення якісного сервісу логістика потребує використання програмних засобів, які здатні автоматизувати обробку даних і підтримувати прийняття управлінських рішень. Особливої актуальності набуває створення веборієнтованих програмних модулів, які дозволяють користувачам швидко виконувати логістичні розрахунки, підбирати транспорт, зберігати результати операцій та аналізувати попередні дані.

Незважаючи на наявність значної кількості корпоративних інформаційних систем, таких як ERP-, TMS-, WMS- та CRM-рішення, їх використання не завжди є доцільним для підприємств, яким необхідно автоматизувати лише окремі логістичні задачі. Такі системи часто мають високу вартість впровадження, потребують складного налаштування, навчання персоналу та технічного супроводу. У той час як великі підприємства можуть використовувати комплексні рішення для управління всіма бізнес-процесами, для виконання базових логістичних розрахунків більш доцільним може бути створення окремого програмного модуля, орієнтованого на конкретні потреби користувача.

Актуальність теми роботи зумовлена необхідністю розробки доступного та зручного програмного інструменту для управління логістичним обслуговуванням підприємства. Такий модуль має забезпечувати автоматизоване виконання розрахунків, пов’язаних із плануванням запасів, розміщенням вантажу, визначенням тривалості маршруту, розрахунком кількості палива, транспортного тарифу та собівартості перевезень. Крім того, важливим є збереження історії розрахунків, що дозволяє користувачеві аналізувати попередні результати та використовувати їх для ухвалення подальших управлінських рішень.

Метою бакалаврської роботи є проєктування та розробка програмного модуля для управління логістичним обслуговуванням підприємства, який забезпечує виконання основних логістичних розрахунків, підбір транспорту, роботу з базою даних, авторизацію користувачів і збереження історії виконаних операцій.

Об'єктом дослідження є процес розробки програмного модуля для управління логістичним обслуговуванням підприємства.

Предметом дослідження є методи, математичні моделі та програмні засоби автоматизації логістичних розрахунків і підтримки прийняття рішень у сфері логістичного обслуговування підприємства.

Для реалізації поставленої мети у роботі визначено такі *завдання*:

1. Дослідити особливості побудови інформаційних систем у сфері логістики.
2. Проаналізувати існуючі програмні рішення, що використовуються для автоматизації логістичних процесів.
3. Визначити засоби та інструменти розробки програмного модуля.
4. Сформувати функціональні вимоги до програмного модуля.
5. Формалізувати основні логістичні операції та подати їх у вигляді математичних моделей.
6. Спроєктувати, реалізувати та перевірити працездатність веборієнтованого програмного модуля із використанням Python, Django, PostgreSQL, HTML та CSS.

У процесі виконання роботи *використано методи* аналізу та узагальнення теоретичних джерел, порівняльного аналізу існуючих програмних рішень, математичного моделювання логістичних операцій, проєктування інформаційних систем, програмної реалізації та тестування вебзастосунку.

Основна частина роботи складається з трьох основних розділів. У першому розділі досліджено теоретичні основи управління логістичним обслуговуванням підприємства, розглянуто особливості побудови інформаційних систем у сфері логістики та проаналізовано існуючі програмні рішення. У другому розділі

описано засоби та інструменти розробки, сформовано функціональні вимоги, розглянуто математичні моделі логістичних операцій і визначено проєктні рішення для створення програмного модуля. У третьому розділі наведено практичну реалізацію веборієнтованого програмного модуля, описано його інтерфейс, основні функції, роботу з базою даних і результати тестування.

Практична цінність даної роботи полягає у розробці веборієнтованого програмного модуля, який може використовуватися для автоматизації базових логістичних розрахунків. Розроблена система дозволяє визначати оптимальний розмір замовлення за моделлю Уїлсона, розраховувати кількість одиниць вантажу для розміщення у транспортному засобі, тривалість маршруту, необхідну кількість палива, транспортний тариф і собівартість перевезення. Наявність функції збереження історії розрахунків підвищує зручність роботи користувача та дає змогу використовувати попередні результати для подальшого аналізу й прийняття управлінських рішень.

Апробація результатів даного дослідження була здійснена на VII Всеукраїнській науково-практичній конференції здобувачів вищої освіти та молодих вчених «Прикладні інформаційні технології» (м. Вінниця, 15 травня 2026 року).

РОЗДІЛ 1
ТЕОРЕТИЧНІ ОСНОВИ ПОБУДОВИ ІНФОРМАЦІЙНИХ СИСТЕМ В ЛОГІСТИЧНОМУ ОБСЛУГОВУВАННІ

1.1. Сутність та роль логістичного обслуговування підприємства

У сучасних умовах розвитку ринкової економіки логістична діяльність є однією з ключових складових ефективного функціонування підприємства [1]. Вона охоплює процеси планування, організації, контролю та регулювання руху матеріальних, інформаційних і фінансових потоків від постачальника до кінцевого споживача [3; 27]. Водночас сучасна логістика не обмежується лише транспортуванням продукції або управлінням запасами. Її значення значно ширше, оскільки вона забезпечує узгоджену роботу всіх ланок підприємства, сприяє зниженню витрат, підвищенню якості обслуговування клієнтів і формуванню конкурентних переваг.

Особливе місце в системі логістичної діяльності займає логістичне обслуговування. Його можна розглядати як сукупність операцій, процесів і сервісних дій, спрямованих на забезпечення своєчасного, якісного та економічно обґрунтованого виконання замовлень споживачів [6]. Логістичне обслуговування охоплює не лише безпосередню доставку товарів, а й приймання та обробку замовлень, підготовку вантажу до транспортування, вибір транспортного засобу, планування маршруту, контроль руху вантажу, інформування клієнта та післяпродажну підтримку.

Сутність логістичного обслуговування полягає у забезпеченні такого рівня сервісу, за якого продукція надходить до споживача у потрібний час, у необхідній кількості, відповідної якості та з оптимальними витратами. Саме тому логістичне обслуговування є важливим інструментом задоволення потреб клієнтів і підвищення ефективності діяльності підприємства. Воно поєднує організаційні, економічні, технологічні та інформаційні аспекти управління логістичними процесами.

У науковій і практичній логістиці для характеристики якісного

логістичного обслуговування часто використовується концепція «7R». Вона визначає основні умови ефективного виконання логістичних операцій:

1. Right product – потрібний товар, тобто відповідність продукції замовленню клієнта.

2. Right quality – належна якість, що передбачає збереження властивостей продукції під час транспортування та зберігання.

3. Right quantity – необхідна кількість, тобто точність комплектації замовлення.

4. Right time – доставка у встановлений час відповідно до узгодженого графіка.

5. Right place – доставка у визначене місце, зазначене споживачем.

6. Right customer – передання товару конкретному споживачеві або одержувачу.

7. Right cost – забезпечення логістичного сервісу з оптимальними витратами.

Зазначена концепція відображає комплексний характер логістичного обслуговування, оскільки якість сервісу визначається не одним окремим показником, а сукупністю взаємопов'язаних умов [7]. Наприклад, своєчасна доставка не може вважатися ефективною, якщо товар прибув у пошкодженому стані або в неправильній кількості. Так само мінімізація витрат не повинна призводити до зниження якості обслуговування клієнта. Отже, завдання підприємства полягає у досягненні балансу між рівнем сервісу та витратами на його забезпечення.

Логістичне обслуговування доцільно розглядати як безперервний процес, що охоплює декілька етапів взаємодії підприємства з клієнтом. Першим етапом є допродажне обслуговування. Воно передбачає формування умов майбутньої логістичної операції, визначення вимог клієнта, узгодження параметрів замовлення, вибір способу доставки, попереднє планування витрат і строків виконання. На цьому етапі підприємство формує очікування клієнта та створює передумови для якісного виконання замовлення.

Другим етапом є обслуговування у процесі виконання замовлення. Саме цей етап є найбільш активним, оскільки охоплює практичну реалізацію логістичної операції. До нього належать підготовка вантажу до перевезення, пакування, маркування, завантаження, вибір транспортного засобу, складання маршруту, контроль руху вантажу та інформування клієнта про стан виконання замовлення. Ефективність цього етапу безпосередньо впливає на швидкість, надійність і якість логістичного обслуговування.

Третім етапом є післяпродажне обслуговування. Воно включає роботу з рекламаціями, поверненням товарів, обміном продукції, зворотною логістикою, аналізом якості виконання замовлення та підтриманням довгострокових відносин із клієнтом. Післяпродажне обслуговування має важливе значення для формування довіри до підприємства, оскільки саме на цьому етапі клієнт оцінює не лише факт доставки, а й готовність підприємства реагувати на можливі проблеми.

Роль логістичного обслуговування в діяльності підприємства проявляється у декількох напрямах [19]. Насамперед воно сприяє підвищенню конкурентоспроможності. В умовах, коли товари різних виробників можуть мати подібні характеристики, саме рівень сервісу часто стає вирішальним чинником вибору постачальника. Підприємство, яке здатне забезпечити швидку, точну та надійну доставку, отримує перевагу над конкурентами.

Важливим напрямом є також оптимізація витрат. Раціональна організація транспортування, складських операцій, управління запасами та обробки замовлень дозволяє зменшити непродуктивні витрати [8; 9], уникнути простоїв, надлишкових запасів і помилок під час комплектації. У результаті підприємство може знижувати собівартість логістичних операцій без втрати якості обслуговування.

Логістичне обслуговування забезпечує стабільність бізнес-процесів [18]. Узгоджена робота постачальників, складів, транспортних підрозділів і споживачів дає змогу мінімізувати затримки, забезпечити ритмічність постачання та підвищити надійність функціонування підприємства [28]. Це

особливо важливо для виробничих і торговельних підприємств, де несвоєчасна доставка може призвести до зупинки виробництва, втрати клієнтів або додаткових фінансових витрат.

Окремого значення логістичне обслуговування набуває в умовах нестабільного зовнішнього середовища. Під час економічних криз, порушення транспортної інфраструктури, зміни попиту або форс-мажорних обставин підприємство повинно швидко адаптувати свої логістичні процеси [21; 24]. У таких умовах ефективна логістика стає не лише засобом підвищення прибутковості, а й інструментом забезпечення безперервності діяльності підприємства.

Важливе місце в системі логістичного обслуговування займає автомобільний транспорт. Його перевагами є мобільність, гнучкість, можливість доставки вантажів за схемою «від дверей до дверей» [5], оперативна зміна маршруту та придатність для перевезення малих і середніх партій вантажів. Саме тому автотранспортна логістика широко використовується у торгівлі, виробництві, дистрибуції та сфері електронної комерції [22].

Разом із тим, автомобільний транспорт має певні обмеження, зокрема залежність від стану доріг, погодних умов, цін на паливо та дорожньої інфраструктури. Проте його гнучкість і доступність роблять його одним із найважливіших елементів логістичного обслуговування підприємства. Для підвищення ефективності автомобільних перевезень доцільно використовувати інформаційні системи, які дозволяють виконувати розрахунки витрат, планувати маршрути, підбирати транспорт і контролювати виконання логістичних операцій [16].

Отже, логістичне обслуговування підприємства є комплексною системою організаційних, економічних, транспортних та інформаційних процесів, спрямованих на своєчасне й якісне задоволення потреб споживачів [23]. Його роль полягає у забезпеченні ефективного руху матеріальних потоків, зниженні витрат, підвищенні рівня сервісу та формуванні конкурентних переваг підприємства. Подальше підвищення ефективності логістичного обслуговування

пов’язане з використанням інформаційних систем, які автоматизують розрахунки та підтримують прийняття управлінських рішень.

1.2. Особливості побудови інформаційних систем у сфері логістики

У сучасних умовах цифровізації економіки ефективне управління логістичними процесами неможливе без використання інформаційних систем [10; 12; 58]. Зростання обсягів даних, ускладнення ланцюгів постачання, підвищення вимог клієнтів до швидкості та якості обслуговування зумовлюють необхідність автоматизації процесів планування, обліку, контролю та аналізу логістичних операцій [11; 25]. Інформаційні системи у сфері логістики дозволяють підприємствам швидше обробляти дані, зменшувати кількість помилок, підвищувати прозорість процесів і приймати більш обґрунтовані управлінські рішення [13; 29].

Інформаційну систему у сфері логістики можна визначити як сукупність програмних, технічних, інформаційних та організаційних засобів, призначених для збору, збереження, обробки, передавання та аналізу даних, пов’язаних із рухом матеріальних, інформаційних і фінансових потоків [17]. Основним призначенням такої системи є забезпечення узгодженої роботи всіх учасників логістичного процесу: постачальників, складів, транспортних підрозділів, менеджерів, клієнтів і керівництва підприємства.

На відміну від традиційного ручного обліку, логістична інформаційна система забезпечує оперативність доступу до даних і можливість їх швидкого оновлення. Це особливо важливо для підприємств, які працюють із великою кількістю замовлень, транспортних засобів, складських залишків і маршрутів. За допомогою інформаційних систем можна контролювати стан виконання замовлень, аналізувати витрати, визначати потребу в транспорті, планувати запаси та оцінювати ефективність логістичного обслуговування [11; 14].

Особливість побудови інформаційних систем у сфері логістики полягає в тому, що вони повинні забезпечувати взаємозв’язок між різними видами даних [15]. У логістиці важливо одночасно враховувати інформацію про товари,

вантажі, клієнтів, транспортні засоби, маршрути, склади, витрати, терміни доставки та результати виконаних операцій. Тому така система має бути не ізольованим програмним продуктом, а цілісним інструментом управління, який поєднує облік, розрахунки, контроль і підтримку прийняття рішень.

Побудова логістичної інформаційної системи передбачає визначення її основних компонентів. До них належать база даних, програмна логіка, користувацький інтерфейс, засоби обробки інформації та механізми захисту даних [31; 26]. База даних забезпечує збереження інформації про користувачів, транспорт, замовлення, розрахунки та інші об’єкти системи [32]. Програмна логіка відповідає за обробку введених даних, виконання розрахунків і формування результатів. Користувацький інтерфейс забезпечує взаємодію користувача із системою через вебсторінки, форми, таблиці та навігаційні елементи.

Важливою вимогою до інформаційних систем у логістиці є оперативність обробки даних. Логістичні процеси часто потребують швидкого реагування на зміни: затримки транспорту, зміну маршруту, нестачу товару, збільшення попиту або зміну витрат. Тому система повинна забезпечувати швидке введення, обробку та відображення інформації. Чим менше часу витрачається на отримання результатів, тим ефективніше підприємство може реагувати на поточну ситуацію.

Наступною важливою особливістю є точність і достовірність інформації. Помилки у логістичних даних можуть призвести до неправильного вибору транспорту, неточного розрахунку витрат, затримки доставки або нераціонального використання ресурсів. Тому під час побудови інформаційної системи необхідно передбачити перевірку введених даних, обмеження некоректних значень і логічний контроль результатів. Наприклад, система не повинна дозволяти вводити від’ємні значення відстані, швидкості, витрат або площі вантажу.

Однією з ключових характеристик логістичних інформаційних систем є інтегрованість. Це означає, що система повинна забезпечувати взаємодію між різними функціональними блоками підприємства. Дані, введені на одному етапі,

мають використовуватися на інших етапах без дублювання. Наприклад, інформація про транспортний засіб може використовуватися під час підбору транспорту, розрахунку витрат, формування історії перевезень і подальшого аналізу ефективності. Такий підхід підвищує узгодженість роботи системи та зменшує ризик помилок.

Для логістичних систем важливим є також збереження історії операцій. Наявність історії розрахунків або перевезень дозволяє користувачеві аналізувати попередні результати, порівнювати варіанти, оцінювати витрати та приймати більш обґрунтовані рішення. Історія даних є особливо корисною для підприємств, які регулярно виконують однотипні перевезення або мають потребу у контролі витрат за певний період.

Окрему увагу під час побудови інформаційних систем у сфері логістики слід приділяти зручності користувацького інтерфейсу. Оскільки з такими системами можуть працювати менеджери, логісти, адміністратори або інші працівники підприємства, інтерфейс має бути зрозумілим, логічно структурованим і простим у використанні. Користувач повинен швидко знаходити потрібні функції, вводити дані у зрозумілі форми, отримувати результати розрахунків і переглядати збережену інформацію.

Важливими вимогами є також безпека та розмежування доступу. Оскільки логістична інформаційна система може містити дані про користувачів, маршрути, витрати, транспортні засоби та історію розрахунків, необхідно забезпечити захист цих даних від несанкціонованого доступу [34]. Для цього можуть використовуватися механізми авторизації користувачів, обмеження прав доступу, збереження персоналізованої історії та контроль дій у системі.

Сучасні логістичні інформаційні системи можуть мати різний рівень складності. На великих підприємствах часто використовуються комплексні системи [31]:

- ERP-системи забезпечують загальне управління ресурсами підприємства;
- WMS-системи використовуються для автоматизації складських

процесів;

- TMS-системи призначені для управління транспортом і маршрутами;
- CRM-системи допомагають організовувати взаємодію з клієнтами.

Процес побудови інформаційної системи у сфері логістики можна подати як послідовність етапів. На першому етапі здійснюється аналіз потреб підприємства та визначення проблем, які повинна вирішувати система [33]. На другому етапі формуються функціональні вимоги, тобто визначається, які задачі система має виконувати. Далі проєктуються структура бази даних, логіка роботи програмного модуля та користувацький інтерфейс. Після цього відбуваються програмна реалізація, тестування, впровадження та подальше вдосконалення системи.

Для розроблюваного програмного модуля особливо важливими є такі принципи побудови інформаційної системи: простота використання, надійне збереження даних, можливість виконання розрахунків, персоналізація історії для кожного користувача та доступність через веб-інтерфейс. Саме тому доцільним є створення веборієнтованого модуля, у якому користувач може працювати через браузер, вводити логістичні дані, отримувати результати розрахунків і переглядати попередні операції.

Веборієнтований підхід має низку переваг для логістичних систем. По-перше, користувачеві не потрібно встановлювати окрему програму на комп’ютер, оскільки доступ до системи здійснюється через браузер. По-друге, централізоване збереження даних у базі даних спрощує контроль інформації та її подальшу обробку. По-третє, вебсистема може бути розширена новими функціями, наприклад, додаванням нових розрахунків, модулів звітності або засобів адміністрування.

Таким чином, інформаційні системи у сфері логістики є важливим інструментом автоматизації та підвищення ефективності управлінських процесів [26; 31; 58]. Їх побудова має враховувати специфіку логістичної діяльності, необхідність швидкої обробки даних, точність розрахунків, інтеграцію різних інформаційних потоків, безпеку інформації та зручність роботи користувача. У

межах даної дипломної роботи ці положення покладено в основу розробки веборієнтованого програмного модуля для управління логістичним обслуговуванням підприємства.

1.3. Аналіз існуючих програмних рішень в логістиці

Сучасний ринок логістичного програмного забезпечення представлений як комплексними корпоративними системами, так і спеціалізованими програмними продуктами для окремих напрямів логістики. До таких напрямів належать управління ресурсами підприємства, транспортними перевезеннями, складськими процесами, взаємодією з клієнтами, документообігом, маршрутами та аналітикою витрат. Розглянемо найбільш поширені програмні рішення, які застосовуються у сфері логістики.

Одним із найвідоміших комплексних рішень є SAP S/4HANA. Це корпоративна система, яка використовується великими підприємствами для управління фінансами, закупівлями, виробництвом, складськими процесами, продажами та логістикою [49]. У сфері логістики SAP S/4HANA дає змогу об’єднати в єдиному інформаційному середовищі дані про постачання, запаси, переміщення товарів, транспортні операції та фінансові показники. Перевагою такого рішення є високий рівень інтеграції бізнес-процесів, можливість роботи з великими обсягами даних і підтримка складної організаційної структури підприємства.

Водночас SAP S/4HANA має низку обмежень для невеликих або середніх підприємств. Насамперед це висока вартість впровадження, потреба у тривалому налаштуванні, складність адаптації під конкретні процеси та необхідність залучення кваліфікованих спеціалістів для супроводу системи. Для підприємств, яким потрібен не повний комплекс управління, а лише окремий інструмент для виконання логістичних розрахунків, таке рішення може бути надмірно складним.

Іншим поширеним рішенням є Microsoft Dynamics 365 Business Central [51]. Ця система орієнтована на управління фінансами, запасами, закупівлями, продажами та операційною діяльністю підприємства. Її перевагою є інтеграція з

іншими сервісами Microsoft, зокрема Excel, Outlook і Power BI, що полегшує роботу з аналітикою та звітністю. У логістичній діяльності Microsoft Dynamics 365 може використовуватися для контролю залишків, обліку замовлень, аналізу закупівель і планування поставок.

Проте для задач, пов’язаних із вузькоспеціалізованими логістичними розрахунками, така система також може бути надлишковою. Вона потребує налаштування, адаптації бізнес-процесів і певного рівня підготовки персоналу. Якщо підприємству необхідно швидко отримувати результати окремих розрахунків, наприклад, тривалості маршруту, витрат палива, транспортного тарифу або собівартості перевезення, використання повноцінної ERP-системи не завжди є раціональним.

Серед більш гнучких рішень можна виділити Odoo [52]. Це модульна система з відкритим кодом, яка дозволяє підприємству впроваджувати лише ті компоненти, які йому необхідні. Наприклад, можна використовувати окремі модулі для складу, продажів, закупівель, обліку або роботи з клієнтами. Перевагою Odoo є модульність, відносна гнучкість і можливість адаптації під конкретні потреби підприємства.

Рисунок 1.1 – Приклад модульної структури ERP-системи Odoo

На рисунку 1.1 наведено приклад модульної структури ERP-системи Odoo. Система об’єднує різні напрями діяльності підприємства, зокрема продажі, закупівлі, складський облік, електронну комерцію, CRM та бухгалтерський облік

[52]. Така структура є зручною для комплексного управління бізнес-процесами, однак для виконання окремих логістичних розрахунків її функціонал може бути надлишковим. Саме тому у межах даної роботи доцільним є створення окремого програмного модуля, орієнтованого на конкретні задачі логістичного обслуговування.

Разом із тим, навіть використання Odoo потребує налаштування, технічної підтримки та розуміння структури системи. Крім того, для реалізації специфічних логістичних розрахунків може виникнути потреба у додатковій розробці або налаштуванні окремих модулів. Тому для невеликої системи підтримки прийняття рішень, орієнтованої саме на виконання базових логістичних обчислень, доцільною може бути розробка окремого програмного модуля.

Для управління транспортними процесами використовуються спеціалізовані TMS-рішення. До таких систем можна віднести програмні продукти, що забезпечують планування маршрутів, вибір транспортних засобів, контроль витрат, моніторинг перевезень і формування транспортної документації [50]. Наприклад, рішення класу Oracle Transportation Management орієнтовані на комплексне управління перевезеннями, планування логістичних ланцюгів, оптимізацію маршрутів і контроль витрат на транспортування [53].

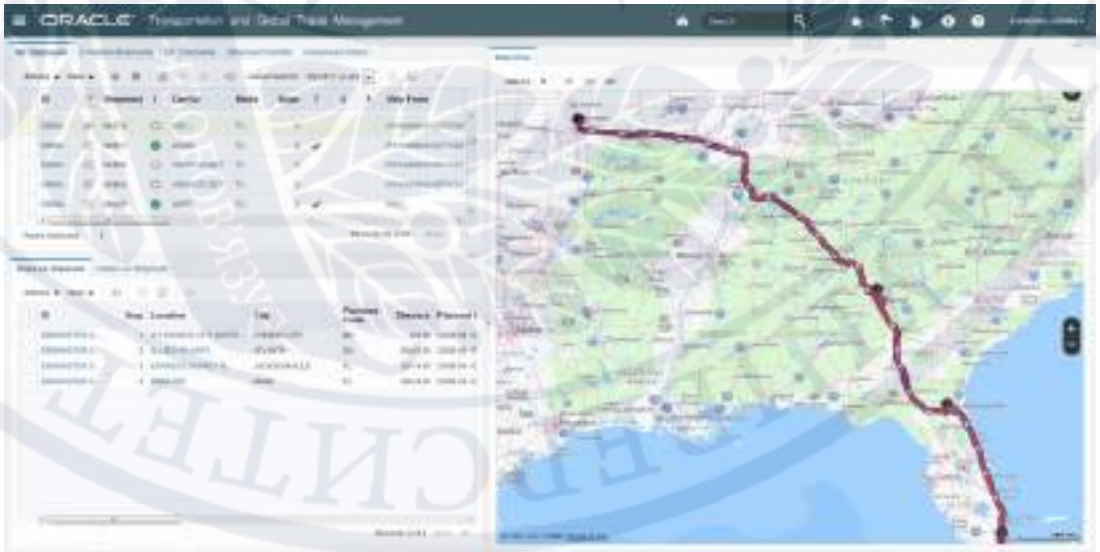

Рисунок 1.2 – Інтерфейс TMS-системи для управління транспортними перевезеннями

На рисунку 1.2 наведено приклад інтерфейсу TMS-системи, призначеної для управління транспортними перевезеннями. Такі системи дозволяють контролювати відправлення, планувати маршрути, відстежувати зупинки, аналізувати роботу перевізників і візуалізувати рух транспорту на карті. Їх перевагою є широкий функціонал для організації транспортної логістики, однак для виконання окремих базових логістичних розрахунків така система може бути надмірно складною.

Перевагою TMS-систем є їхня орієнтація саме на транспортну логістику. Вони дозволяють автоматизувати значну частину процесів, пов'язаних із доставкою вантажів, зменшити витрати на перевезення, підвищити прозорість транспортних операцій і контролювати якість виконання замовлень. Однак такі системи часто розраховані на підприємства з великим обсягом перевезень і складною транспортною інфраструктурою. Для навчального або прикладного програмного модуля, який має виконувати конкретні розрахунки, повноцінна TMS-система є занадто складною.

Окрему групу становлять системи управління складом [59]. Одним із прикладів є Manhattan Active Warehouse Management. Це рішення орієнтоване на великі складські комплекси та дозволяє автоматизувати процеси приймання товарів, розміщення, зберігання, комплектації, відвантаження та контролю складських залишків. Його перевагою є можливість роботи з великими потоками товарів і підтримка складних складських процесів [54].

Ще одним прикладом є Infor WMS, який застосовується для управління складськими операціями, контролю запасів, оптимізації використання складського простору та підвищення точності обробки замовлень [55]. Такі системи є ефективними для підприємств, де складська логістика має велике значення. Проте вони не орієнтовані безпосередньо на виконання простих транспортно-логістичних розрахунків, таких як розрахунок витрат палива, тарифу, тривалості маршруту або собівартості перевезення.

На рисунку 1.3 наведено приклад інтерфейсу WMS-системи, призначеної для управління складськими процесами. Такі системи дозволяють контролювати

розміщення товарів, стан виконання складських операцій, комплектацію замовлень і використання складського простору. Водночас WMS-рішення орієнтовані переважно на складську логістику, тому не є повноцінною заміною окремого програмного модуля для виконання транспортно-логістичних розрахунків.

Рисунок 1.3 – Інтерфейс WMS-системи для управління складськими процесами

У сфері взаємодії з клієнтами широко застосовуються CRM-системи. Наприклад, Salesforce Service Cloud дозволяє організовувати роботу із зверненнями клієнтів, контролювати комунікації, зберігати історію взаємодії та аналізувати якість обслуговування [56]. У логістиці такі системи можуть використовуватися для інформування клієнтів про статус замовлення, обробки запитів, фіксації претензій і підтримки довгострокових відносин із замовниками.

Серед доступніших CRM-рішень можна виділити Bitrix24. Він поєднує функції CRM, управління завданнями, комунікації та документообігу [57]. Для логістичного підприємства така система може бути корисною для обліку клієнтів, контролю заявок, зберігання історії комунікацій і розподілу завдань між працівниками. Проте CRM-системи не призначені для повноцінного математичного моделювання логістичних операцій і не можуть замінити

спеціалізований калькулятор або модуль підтримки прийняття рішень.

Проведений аналіз показує, що існуючі програмні рішення мають значний функціонал і можуть ефективно використовуватися для автоматизації різних аспектів логістичної діяльності. Комплексні ERP-системи доцільні для великих підприємств, які потребують інтегрованого управління всіма ресурсами. TMS-системи ефективні для компаній із розвиненою транспортною інфраструктурою та великими обсягами перевезень. WMS-рішення є необхідними для підприємств зі складними складськими процесами, а CRM-системи корисні для організації взаємодії з клієнтами.

Водночас більшість розглянутих систем має спільні обмеження: високу вартість впровадження, складність налаштування, потребу у навчанні персоналу та наявність надлишкового функціоналу для підприємств, яким необхідні лише окремі логістичні розрахунки. Крім того, готові системи не завжди повністю відповідають специфічним вимогам конкретного підприємства або навчального проєкту, тому їх адаптація може потребувати додаткових ресурсів.

У межах даної дипломної роботи розроблюваний програмний модуль не має на меті замінити повноцінні корпоративні системи. Його призначення полягає у створенні доступного та вузькоорієнтованого інструменту для виконання базових логістичних розрахунків і підтримки прийняття рішень. На відміну від великих інформаційних систем, такий модуль має простішу структуру, може бути реалізований як вебзастосунок і адаптований до конкретних потреб користувача.

Таким чином, аналіз існуючих програмних рішень у логістиці підтверджує доцільність розробки власного веборієнтованого програмного модуля. Готові системи мають широкий функціонал у межах великих підприємств, проте для задач автоматизації окремих логістичних розрахунків вони можуть бути надмірно складними [20]. Розроблюваний програмний модуль є більш придатним для вирішення конкретних логістичних задач, пов’язаних з управлінням логістичним обслуговуванням підприємства.

РОЗДІЛ 2
ПРОЕКТУВАННЯ ПРОГРАМНОГО МОДУЛЯ ДЛЯ УПРАВЛІННЯ ЛОГІСТИЧНИМ ОБСЛУГОВУВАННЯМ ПІДПРИЄМСТВА

2.1. Засоби та інструменти розробки програмного модуля

Для розробки програмного модуля управління логістичним обслуговуванням підприємства я використала комплекс сучасних програмних засобів та інструментів, які забезпечують створення веборієнтованої системи з можливістю виконання логістичних розрахунків, збереження результатів у базі даних, перегляду історії операцій та адміністрування основних даних системи.

Програмний модуль реалізовано як вебзастосунок, що передбачає поділ системи на серверну частину, клієнтську частину та базу даних. Серверна частина відповідає за обробку запитів користувача, виконання розрахунків, авторизацію, збереження та отримання даних. Клієнтська частина забезпечує відображення сторінок у браузері та взаємодію користувача з функціями системи. База даних використовується для збереження інформації про користувачів, доступні транспортні засоби та історію виконаних логістичних розрахунків.

Основні засоби та інструменти, використані під час розробки програмного модуля:

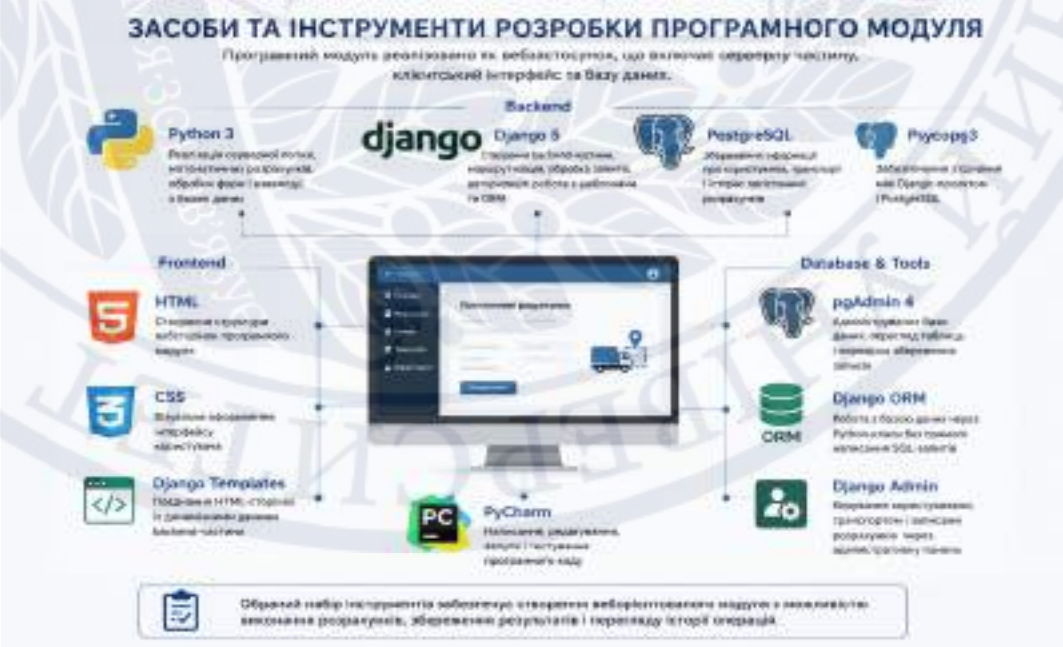

Рисунок 2.1 – Засоби та інструменти розробки программного модуля

Python 3 є основною мовою програмування, на якій реалізовано серверну логіку програмного модуля [36]. За допомогою Python виконується обробка даних, які користувач вводить у вебформах, реалізуються математичні розрахунки, формується логіка роботи окремих функцій системи та забезпечується взаємодія між backend-частиною і базою даних.

У межах розробленого програмного модуля Python використовується для приймання вхідних параметрів логістичних операцій, їх перевірки, виконання обчислень і підготовки результатів для подальшого відображення на вебсторінках. Оскільки система орієнтована на підтримку прийняття рішень у сфері логістичного обслуговування, важливим завданням є швидке та коректне виконання розрахунків, пов’язаних із транспортуванням, витратами та вибором оптимального варіанта перевезення. Саме Python забезпечує реалізацію цієї розрахункової частини.

Перевагою Python є зрозумілий синтаксис, велика кількість бібліотек, зручність роботи з даними та сумісність із сучасними вебфреймворками. Це робить його доцільним вибором для створення програмного модуля, який поєднує вебінтерфейс, базу даних і алгоритми логістичних розрахунків.

Django 5 використовується як основний вебфреймворк для створення backend-частини системи [37]. Він забезпечує структуру програмного модуля, обробку HTTP-запитів, маршрутизацію між сторінками, роботу з HTML-шаблонами, авторизацію користувачів, взаємодію з базою даних через ORM та доступ до адміністративної панелі.

Розроблений програмний модуль Django відповідає за зв’язок між діями користувача у браузері та внутрішньою логікою системи. Наприклад, коли користувач заповнює форму для логістичного розрахунку, Django приймає ці дані, передає їх до відповідної функції обробки, формує результат, зберігає його в базі даних і повертає на сторінку для відображення.

Важливою особливістю Django є використання архітектурного підходу MVT, тобто Model – View – Template. У цьому підході моделі відповідають за структуру даних і зв’язок із базою даних, представлення обробляють запити

користувача та визначають логіку роботи сторінок, а шаблони забезпечують відображення інформації у браузері. Такий підхід дозволяє логічно розділити частини програмного модуля, що спрощує його розробку, тестування та подальше вдосконалення.

PostgreSQL є реляційною системою керування базами даних, яка використовується для збереження основної інформації програмного модуля [40]. У базі даних зберігаються відомості про зареєстрованих користувачів, список доступного транспорту, параметри логістичних операцій, результати розрахунків та історія виконаних дій.

Використання PostgreSQL є доцільним, оскільки дані програмного модуля мають чітку структуру та взаємозв'язки [32]. Наприклад, кожен користувач може мати власну історію розрахунків, а кожен запис історії може містити дату створення, введені параметри, обраний транспорт і отриманий результат. Реляційна модель бази даних дозволяє впорядковано зберігати такі дані та забезпечувати їх цілісність[35; 41].

PostgreSQL також забезпечує надійність збереження інформації, підтримку зв'язків між таблицями, можливість швидкого пошуку й фільтрації записів [60]. Це є важливим для програмного модуля, оскільки результати логістичних розрахунків повинні бути доступними користувачеві не лише під час поточного сеансу роботи, а й після повторного входу в систему.

Psycopg3 використовується як драйвер для з'єднання Django-проєкту з базою даних PostgreSQL [42]. Його основне призначення полягає в тому, щоб забезпечити технічну можливість передавання запитів від Django до PostgreSQL і отримання результатів у зворотному напрямку.

У процесі роботи програмного модуля користувач вводить дані на вебсторінці, після чого backend-частина обробляє ці дані та за необхідності зберігає їх у базі даних. Саме Psycopg3 забезпечує взаємодію між програмним кодом Django та PostgreSQL. Завдяки цьому система може створювати нові записи, отримувати наявні дані, оновлювати інформацію та зберігати результати логістичних обчислень.

Хоча користувач безпосередньо не взаємодіє з Psycopg3, цей компонент є важливою частиною технічної реалізації системи, оскільки без нього неможлива повноцінна робота Django з PostgreSQL.

pgAdmin 4 є графічним інструментом для адміністрування PostgreSQL [43]. Він використовується розробником для перегляду бази даних, перевірки створених таблиць, аналізу збережених записів, контролю структури бази даних та виконання SQL-запитів.

У процесі розробки програмного модуля pgAdmin 4 застосовується для перевірки правильності збереження інформації. Наприклад, після створення нового користувача, додавання транспортного засобу або виконання логістичного розрахунку можна перевірити, чи відповідні записи були додані до таблиць бази даних. Це дає змогу виявляти помилки на етапі розробки та забезпечувати коректну роботу системи.

pgAdmin 4 також використовується для контролю таблиць, пов’язаних із користувачами, транспортом та історією розрахунків. Таким чином, цей інструмент виконує допоміжну, але важливу роль у процесі розробки й тестування програмного модуля.

HTML використовується для створення структури вебсторінок програмного модуля [44]. За допомогою HTML формуються основні елементи інтерфейсу: заголовки, текстові блоки, форми введення даних, кнопки, таблиці, посилання, меню навігації та блоки для відображення результатів.

У межах програмного модуля HTML застосовується для створення головної сторінки, сторінки з інструкцією, сторінок калькулятора, підбору транспорту, авторизації користувача та перегляду історії розрахунків. Кожна з цих сторінок має власну структуру та набір елементів, які забезпечують взаємодію користувача із системою.

HTML є основою клієнтської частини програмного модуля, оскільки саме він визначає, які елементи буде бачити користувач у браузері та як буде організовано зміст сторінок.

CSS використовується для стилізації та візуального оформлення

вебсторінок [45]. Якщо HTML визначає структуру сторінки, то CSS відповідає за її зовнішній вигляд: кольори, шрифти, розташування блоків, розміри кнопок, відступи, межі, оформлення таблиць, форм і навігаційного меню.

Для програмного модуля управління логістичним обслуговуванням підприємства важливо, щоб інтерфейс був не лише функціональним, а й зручним для користувача. Завдяки CSS сторінки системи мають єдиний стиль оформлення, зрозумілу структуру та візуально виділені елементи керування. Це спрощує роботу користувача з модулем і робить процес введення даних та перегляду результатів більш зручним.

CSS використовується для оформлення кнопок, форм, таблиць історії розрахунків, меню навігації, інформаційних блоків і сторінок з результатами. Таким чином, CSS забезпечує візуальну цілісність програмного модуля.

Django Templates – це шаблонізатор Django, який дозволяє поєднувати HTML-сторінки з динамічними даними, що надходять із backend-частини. Завдяки шаблонам на сторінках можуть відображатися результати логістичних розрахунків, інформація про поточного користувача, список доступного транспорту та записи історії.

У розробленій системі Django Templates використовуються для того, щоб одна й та сама HTML-сторінка могла відображати різні дані залежно від дій користувача. Наприклад, після виконання розрахунку шаблон отримує результат із backend-частини та виводить його на сторінці. Аналогічно, на сторінці історії шаблон відображає записи, які належать конкретному авторизованому користувачеві.

Використання шаблонів дозволяє зробити сторінки програмного модуля динамічними та пов’язаними з даними, що зберігаються у базі даних.

Django ORM є механізмом роботи з базою даних через Python-класи. ORM дозволяє описувати таблиці бази даних у вигляді моделей, а потім виконувати операції створення, читання, оновлення, видалення та фільтрації записів без прямого написання SQL-запитів.

У межах програмного модуля Django ORM використовується для

створення моделей, які відповідають основним сутностям системи: користувачам, транспортним засобам та історії логістичних розрахунків. Наприклад, запис історії може створюватися безпосередньо в коді Python після виконання розрахунку, а ORM забезпечує його збереження у відповідній таблиці PostgreSQL.

Перевагою Django ORM є те, що він спрощує роботу з базою даних, зменшує кількість помилок у SQL-запитах і робить код більш зрозумілим та структурованим.

Django Admin – це вбудована адміністративна панель Django, яка використовується для керування даними програмного модуля. За допомогою адміністративної панелі можна переглядати, додавати, редагувати та видаляти записи, пов’язані з користувачами, транспортом та історією логістичних розрахунків.

У даній системі Django Admin є зручним інструментом для адміністратора або розробника. Наприклад, через адміністративну панель можна додати новий вид транспорту, змінити його характеристики, переглянути список користувачів або перевірити збережені розрахунки. Це дозволяє керувати основними даними системи без необхідності безпосередньо працювати з базою даних через SQL-запити.

Django Admin значно спрощує процес адміністрування програмного модуля та забезпечує швидкий доступ до основних об’єктів системи.

PyCharm використовується як інтегроване середовище розробки, у якому створюється, редагується та тестується програмний код [46]. У PyCharm виконується написання Python-коду, створення файлів Django-проєкту, редагування HTML- і CSS-файлів, запуск локального сервера, виконання міграцій бази даних та перевірка роботи програмного модуля.

Перевагою PyCharm є зручне середовище для роботи з проєктами Python і Django. Воно дозволяє швидко переходити між файлами, перевіряти синтаксис, запускати команди в терміналі, працювати з віртуальним середовищем та контролювати структуру проєкту. У процесі розробки це допомагає зменшити

кількість технічних помилок і прискорити створення програмного модуля.

Таким чином, PyCharm є основним середовищем, у якому здійснюється практична реалізація програмного модуля.

Отже, обраний набір засобів та інструментів дозволяє реалізувати повноцінний веборієнтований програмний модуль для управління логістичним обслуговуванням підприємства. Python 3 і Django 5 забезпечують серверну логіку, маршрутизацію, обробку запитів і виконання розрахунків. PostgreSQL відповідає за надійне збереження даних, Psycopg3 забезпечує з'єднання між Django та базою даних, pgAdmin 4 використовується для адміністрування і контролю інформації. HTML, CSS та Django Templates формують користувацький інтерфейс, а Django ORM і Django Admin спрощують роботу з даними та адміністрування системи. Використання PyCharm забезпечує зручну розробку, тестування та супровід програмного модуля.

2.2. Формування функціональних вимог до програмного модуля

Формування функціональних вимог є важливим етапом проєктування програмного модуля, оскільки саме на цьому етапі визначається, які задачі повинна виконувати система, які дані вона має обробляти та які можливості повинні бути доступні користувачеві [47]. Для програмного модуля управління логістичним обслуговуванням підприємства функціональні вимоги формуються з урахуванням основної мети розробки – забезпечення швидкого виконання логістичних розрахунків, підбору відповідного виду транспорту, збереження результатів і надання користувачеві зручного доступу до історії виконаних операцій.

Розроблений програмний модуль призначений для використання як веборієнтована система підтримки прийняття рішень у сфері логістичного обслуговування. Користувач отримує можливість працювати із системою через браузер, вводити необхідні параметри перевезення, виконувати розрахунки, переглядати результати та аналізувати попередні обчислення. Такий підхід дозволяє автоматизувати частину логістичних процесів і зменшити кількість

ручних розрахунків.

Функціональні вимоги до програмного модуля можна поділити на декілька основних груп: вимоги до користувацької взаємодії, вимоги до авторизації, вимоги до виконання логістичних розрахунків, вимоги до підбору транспорту, вимоги до збереження даних, а також вимоги до адміністрування системи [48].

Основні функціональні вимоги до програмного модуля:

1) Перегляд головної сторінки. Користувач має можливість ознайомитися з призначенням програмного модуля та перейти до основних розділів системи.

2) Перегляд інструкції. Система повинна надавати користувачеві коротке пояснення щодо використання модуля.

3) Реєстрація користувача. Новий користувач має можливість створити обліковий запис.

4) Авторизація користувача. Зареєстрований користувач має можливість увійти до системи.

5) Вихід із системи. Користувач має можливість завершити поточний сеанс роботи.

6) Введення параметрів логістичного розрахунку. Система повинна приймати дані, необхідні для виконання обчислень.

7) Виконання логістичних розрахунків. Програмний модуль має автоматично обчислювати результат на основі введених даних.

8) Підбір транспорту. Система повинна надавати можливість вибору або рекомендації транспортного засобу.

9) Відображення результату. Після виконання розрахунку користувач має бачити отриманий результат на сторінці.

10) Збереження історії розрахунків. Результати виконаних розрахунків повинні зберігатися у базі даних.

11) Перегляд історії розрахунків. Авторизований користувач має можливість переглядати власні попередні розрахунки.

12) Адміністрування даних. Адміністратор має можливість керувати користувачами, транспортом і записами розрахунків.

Опишемо більш детально кожен з пунктів. Першою з основних функціональних вимог є наявність головної сторінки, яка виконує інформаційну та навігаційну функцію. На ній користувач повинен мати можливість ознайомитися з призначенням програмного модуля та перейти до основних сторінок системи. Головна сторінка є початковою точкою взаємодії користувача із системою, тому вона повинна бути зрозумілою, логічно структурованою та містити посилання на ключові функції.

Важливою вимогою є наявність сторінки з інструкцією, яка пояснює користувачеві порядок роботи з програмним модулем. Така сторінка дозволяє зменшити ймовірність помилок під час введення даних і допомагає користувачеві правильно виконувати логістичні розрахунки. Інструкція може містити короткий опис призначення системи, порядок заповнення форми, пояснення основних параметрів і спосіб перегляду результатів.

Оскільки програмний модуль передбачає збереження історії розрахунків для окремих користувачів, необхідними функціональними вимогами є реєстрація та авторизація користувачів. Реєстрація дозволяє створити новий обліковий запис, а авторизація – отримати доступ до персоналізованих можливостей системи. Після входу до системи користувач може виконувати розрахунки та переглядати саме власну історію, що забезпечує розмежування даних між різними користувачами.

Функція виходу із системи також є необхідною, оскільки вона дозволяє користувачеві безпечно завершити роботу з програмним модулем. Це особливо важливо у випадках, коли система використовується на спільному комп’ютері або в робочому середовищі підприємства.

Центральною функцією програмного модуля є введення параметрів логістичного розрахунку. Користувач повинен мати можливість заповнити форму, у якій зазначаються необхідні дані для виконання обчислень. До таких параметрів можуть належати відстань перевезення, маса або обсяг вантажу, тип транспорту, тариф, витрати на паливо, додаткові витрати або інші показники, які впливають на кінцевий результат. Дані, введені користувачем, мають

передаватися до backend-частини системи для подальшої обробки.

Наступною важливою вимогою є автоматичне виконання логістичних розрахунків. Програмний модуль повинен обробляти введені користувачем дані та формувати результат без необхідності ручного обчислення. Це підвищує швидкість роботи, зменшує ризик арифметичних помилок і забезпечує єдиний підхід до розрахунків. Результати можуть включати орієнтовну вартість перевезення, загальні логістичні витрати, ефективність обраного виду транспорту або інші показники, передбачені логікою системи.

Окремою функціональною вимогою є підбір транспортного засобу. Система повинна надавати користувачеві можливість переглядати доступні види транспорту або отримувати рекомендацію щодо вибору транспорту залежно від умов перевезення. Наприклад, вибір транспорту може залежати від ваги вантажу, відстані, вантажопідйомності, швидкості доставки або вартості використання транспортного засобу. Це дозволяє використовувати програмний модуль не лише як калькулятор, а й як інструмент підтримки прийняття рішень.

Після виконання розрахунку система повинна забезпечувати відображення результату на відповідній сторінці. Результат має бути поданий у зрозумілому вигляді, щоб користувач міг швидко оцінити отримані дані та використати їх для прийняття управлінського рішення. Доцільно, щоб разом із результатом відображалися основні вхідні параметри, які були використані під час обчислення, оскільки це дозволяє перевірити правильність введених даних.

Важливою функціональною вимогою є збереження історії логістичних розрахунків. Після виконання обчислень система повинна зберігати результат у базі даних. Це дозволяє користувачеві повернутися до попередніх розрахунків, порівняти різні варіанти перевезення та проаналізувати раніше отримані результати. Збереження історії також підвищує практичну цінність програмного модуля, оскільки система стає не лише інструментом одноразового розрахунку, а й засобом накопичення інформації.

Функція перегляду історії розрахунків повинна бути доступна авторизованому користувачеві. При цьому кожен користувач має бачити лише

власні записи. Такий підхід забезпечує персоналізацію роботи із системою та захист даних інших користувачів. На сторінці історії можуть відображатися дата виконання розрахунку, введені параметри, обраний транспорт і отриманий результат.

Також програмний модуль повинен передбачати можливість адміністрування даних. Адміністратор системи має мати доступ до керування користувачами, списком транспортних засобів і записами історії розрахунків. Для цього використовується вбудована адміністративна панель Django Admin. Через неї можна додавати нові транспортні засоби, редагувати їх характеристики, переглядати користувачів і контролювати збережені результати обчислень.

Загальну логіку взаємодії користувача з програмним модулем можна описати таким чином. Спочатку користувач відкриває головну сторінку системи, за потреби переглядає інструкцію, проходить реєстрацію або авторизацію. Після входу до системи він переходить до сторінки логістичного розрахунку або підбору транспорту, вводить необхідні параметри та запускає обчислення. Система обробляє дані, виконує розрахунок, відображає результат і зберігає його в базі даних. Надалі користувач може перейти до сторінки історії та переглянути попередні результати.

Таким чином, сформовані функціональні вимоги визначають основні можливості програмного модуля та забезпечують його відповідність поставленій меті. Розроблена система повинна забезпечувати введення даних, виконання логістичних розрахунків, підбір транспорту, збереження результатів, перегляд історії та адміністрування інформації. Реалізація зазначених вимог дозволяє створити програмний модуль, який може бути використаний як інструмент підтримки прийняття рішень у процесі управління логістичним обслуговуванням підприємства.

2.3 Формалізація та математичне моделювання логістичних операцій

Формалізація логістичних операцій є одним із ключових етапів

проєктування програмного модуля, оскільки вона дозволяє подати логістичні процеси у вигляді математичних залежностей, які можуть бути реалізовані засобами програмування [2]. У межах розробленого програмного модуля основна увага приділяється автоматизації типових логістичних розрахунків, що використовуються під час планування запасів, розміщення вантажу, оцінювання тривалості маршруту, визначення витрат палива, розрахунку транспортного тарифу та собівартості перевезення.

Математичне моделювання в даному випадку виконує роль основи для побудови логіки програмного модуля. Користувач вводить необхідні початкові дані через веб-інтерфейс, після чого система передає ці дані до серверної частини, виконує відповідний розрахунок і повертає результат на сторінку. Такий підхід дозволяє скоротити час виконання обчислень, зменшити ймовірність помилок і забезпечити єдиний підхід до розрахунку логістичних показників.

У програмному модулі реалізовано декілька основних математичних моделей, кожна з яких відповідає окремій логістичній задачі. Загальна структура розрахунків передбачає введення вхідних параметрів, обробку цих параметрів за відповідною формулою, округлення результату та його подальше відображення користувачеві.

Першою реалізованою моделлю є модель оптимального розміру замовлення Роберта Уїлсона. Вона використовується для визначення такого обсягу замовлення, за якого витрати, пов’язані із замовленням і зберіганням запасів, є раціональними [1] [2]. У програмному модулі цей розрахунок реалізовано за формулою:

$$Q_w = \sqrt{\frac{2 \cdot K \cdot v}{s}}, \qquad (2.1)$$

де Q_w – оптимальний розмір замовлення;

K – витрати на здійснення одного замовлення;

v – інтенсивність використання запасів або попит;

s – витрати на зберігання одиниці запасу.

У програмному коді ця модель реалізована у функції

calculate_wilson_model (рис. 2.2). Користувач вводить витрати на здійснення замовлення, інтенсивність використання запасів і витрати на зберігання. Після цього система обчислює оптимальний розмір замовлення за допомогою квадратного кореня.

```
def calculate_wilson_model(order_cost, demand_intensity, storage_cost):

    result = math.sqrt((2 * order_cost * demand_intensity) / storage_cost)
    return round(result, 2)
```

Рисунок 2.2 – Реалізація у коді формули Роберта Уїлсона

Застосування моделі Уїлсона в програмному модулі дозволяє оцінити раціональний обсяг замовлення та може бути використане підприємством для планування запасів. Це особливо важливо для логістичного обслуговування, оскільки надмірні запаси призводять до збільшення витрат на зберігання, а недостатні – до ризику переривання постачання.

Наступною логістичною операцією є розрахунок кількості одиниць вантажу, які можна розмістити у транспортному засобі [4]. Цей розрахунок використовується для оцінювання ефективності використання вантажного простору. Його суть полягає у визначенні кількості вантажних одиниць, які можуть бути розміщені на доступній площі транспортного засобу.

Кількість одиниць вантажу визначається за формулою:

$$N = \frac{S_t}{S_c}, \tag{2.2}$$

де N – кількість одиниць вантажу;

S_t – доступна площа вантажного місця;

S_c – площа одного вантажу.

У програмному модулі ця формула реалізована функцією calculate_cargo_distribution (рис. 2.3). Вона приймає площу одного вантажного місця та доступну площу транспорту, після чого визначає кількість вантажних одиниць, які можна розмістити. Оскільки фактично неможливо розмістити

частину окремої одиниці вантажу, результат перетворюється на ціле число.

```
def calculate_cargo_distribution(cargo_area, available_area):

    result = available_area / cargo_area
    return int(result)
```

Рисунок 2.3 – Реалізація у коді формули Розподілу вантажу

Такий розрахунок дозволяє користувачеві швидко оцінити, чи достатньо вантажного простору для перевезення певного вантажу, а також визначити доцільність використання конкретного транспортного засобу.

Важливим показником під час планування перевезення є тривалість маршруту. Вона залежить від довжини маршруту та середньої швидкості руху транспортного засобу [5]. Цей розрахунок дає змогу оцінити орієнтовний час доставки вантажу та може використовуватися під час планування графіка перевезень.

Тривалість маршруту визначається за формулою:

$$T = \frac{L}{V}, \tag{2.3}$$

де T – тривалість маршруту;

L – шлях, за яким рухається транспорт;

V – швидкість транспорту.

У програмному модулі розрахунок тривалості маршруту реалізовано функцією calculate_route_time (рис. 2.4). Особливістю цієї функції є те, що результат подається не лише у вигляді десяткового числа, а у зручному для користувача форматі – у годинах і хвилинах. Наприклад, якщо результат розрахунку становить 3,5 години, система відображає його як 3 години 30 хвилин. Така форма подання є зручною для практичного використання, оскільки дозволяє швидко оцінити час виконання маршруту.

Наступним розрахунком є визначення кількості палива (рис. 2.5), необхідного для проходження маршруту. У транспортній логістиці цей показник

є важливим, оскільки витрати на паливо становлять значну частину загальних витрат на перевезення [5], [16].

```
def calculate_route_time(distance, speed):

    total_hours = distance / speed

    hours = int(total_hours)
    minutes = round((total_hours - hours) * 60)

    if minutes == 60:
        hours += 1
        minutes = 0
    return hours, minutes
```

Рисунок 2.4 – Реалізація у коді формули Тривалості маршруту

Для виконання цього розрахунку враховується довжина маршруту та середня витрата палива транспортного засобу на 100 км.Кількість палива визначається за формулою:

$$F = \frac{L \cdot V_f}{100}, \tag{2.4}$$

де F – кількість палива, необхідна для проходження маршруту;

L – шлях, за яким рухається транспорт;

V_f – витрата палива транспортного засобу на 100 км.

```
def calculate_fuel_amount(distance, fuel_consumption):

    result = (distance * fuel_consumption) / 100
    return round(result, 2)
```

Рисунок 2.5 – Реалізація у коді формули Витрат палива

Результат округлюється до двох знаків після коми, що робить його зручним для подальшого використання під час розрахунку транспортних витрат. Застосування цієї формули дозволяє визначити, скільки палива потрібно для

виконання конкретного перевезення. Отриманий результат може бути використаний для подальшого планування маршруту, оцінювання ресурсних потреб і визначення економічної доцільності використання певного транспортного засобу.

Окремою логістичною операцією є розрахунок транспортного тарифу. Транспортний тариф є важливим економічним показником у сфері перевезень [5]. Він відображає вартість переміщення вантажу на певну відстань і використовується для оцінювання витрат на транспортне обслуговування. Транспортні тарифи можуть включати плату за безпосереднє перевезення вантажу, а також додаткові витрати, пов'язані з виконанням транспортної операції.

У логістичній діяльності транспортний тариф дає змогу визначити вартість перевезення в розрахунку на одиницю відстані. Це дозволяє порівнювати різні маршрути, транспортні засоби або варіанти організації доставки. Транспортний тариф розраховується за формулою:

$$R = \frac{C}{L}, \tag{2.5}$$

де R – транспортний, тариф;

C – витрати на перевезення вантажу;

L – шлях, за яким рухається транспорт.

У програмному модулі цей розрахунок реалізовано функцією calculate_transport_tariff (рис. 2.6). Функція приймає загальні витрати на перевезення та відстань, після чого визначає тариф на одиницю відстані.

```
def calculate_fuel_amount(distance, fuel_consumption):

    result = (distance * fuel_consumption) / 100
    return round(result, 2)
```

Рисунок 2.6 – Реалізація у коді формули Транспортного тарифу

Отримане значення тарифу показує, яка частина витрат припадає на один

кілометр перевезення. Цей показник може бути використаний як окремо для оцінювання ефективності транспортного процесу, так і як складова подальшого розрахунку собівартості перевезення.

Завершальним розрахунком у програмному модулі є визначення собівартості перевезення. Собівартість перевезення відображає загальні витрати, пов'язані з виконанням транспортної операції [5] [8]. Вона враховує операційні витрати та витрати, що залежать від тарифу і відстані перевезення.

Процес перевезення вантажів складається з декількох етапів, серед яких можна виділити початковий, рухомий та кінцевий. Початковий етап охоплює підготовку транспортного засобу, завантаження вантажу, оформлення необхідних операцій і підготовку до відправлення. Рухомий етап пов'язаний із безпосереднім переміщенням вантажу за маршрутом. Кінцевий етап передбачає прибуття транспорту, розвантаження та завершення транспортної операції.

З урахуванням цих етапів витрати на перевезення можна поділити на початково-кінцеві та рухомі. До початково-кінцевих витрат належать витрати, які не залежать безпосередньо від довжини маршруту: підготовка транспорту, завантаження, розвантаження, простої, маневрові роботи та інші організаційні операції. Рухомі витрати, навпаки, залежать від відстані перевезення та включають витрати, пов'язані з рухом транспортного засобу.

У загальному вигляді формула собівартості перевезення має такий вигляд:

$$\mathrm{C_{тт}} = \mathrm{B_{п.к}} + R \cdot L, \qquad (2.6)$$

де $C_{\text{тт}}$ – собівартість перевезення;

$B_{\text{п.к}}$ – операційні витрати, пов'язані з початково-кінцевими операціями;

R – транспортний тариф або шляхові витрати на 1 км;

L – відстань перевезення вантажу.

У програмному коді цей розрахунок реалізовано функцією calculate_transport_cost (рис. 2.7). Функція приймає операційні витрати, тариф і відстань, після чого визначає підсумкову собівартість перевезення.

```
def calculate_transport_cost(operation_cost, tariff, distance):

    result = operation_cost + tariff * distance
    return round(result, 2)
```

Рисунок 2.7 – Реалізація у коді формули Собівартості перевезення

Цей розрахунок має практичне значення для підприємства, оскільки дозволяє оцінити загальну вартість виконання логістичної операції. Він враховує як постійні витрати, пов’язані з підготовкою та завершенням перевезення, так і змінні витрати, які залежать від довжини маршруту. Чим більшою є відстань перевезення, тим більший вплив на загальну собівартість мають рухомі витрати. Водночас відносна частка початково-кінцевих витрат у собівартості одного кілометра може зменшуватися зі збільшенням відстані. На основі отриманого результату можна приймати рішення щодо доцільності перевезення, вибору маршруту або транспортного засобу.

Загальний алгоритм виконання логістичного розрахунку у програмному модулі можна описати таким чином. Спочатку користувач переходить на сторінку калькулятора та обирає потрібний вид розрахунку. Після цього він вводить початкові дані у відповідну форму. Система передає ці дані до backend-частини, де вони обробляються засобами Python і Django. Далі викликається відповідна функція розрахунку, яка виконує математичні обчислення та повертає результат. Після цього результат відображається на вебсторінці та, за потреби, зберігається у базі даних PostgreSQL як запис історії розрахунків користувача.

У спрощеному вигляді цей процес можна подати так:

Вхідні дані → Обробка даних → Математичний розрахунок → Результат → Збереження в базі даних.

Таким чином, математичні моделі виконують не лише розрахункову функцію, а й забезпечують накопичення інформації для підтримки прийняття управлінських рішень.

РОЗДІЛ 3
РЕАЛІЗАЦІЯ ПРОГРАМНОГО МОДУЛЯ ДЛЯ УПРАВЛІННЯ ЛОГІСТИЧНИМ ОБСЛУГОВУВАННЯМ ПІДПРИЄМСТВА

3.1. Розробка інтерфейсу користувача

Інтерфейс користувача програмного модуля розроблено з урахуванням простоти, зрозумілості та зручності використання. Оскільки система призначена для виконання логістичних розрахунків, основну увагу приділено чіткому розміщенню форм введення даних, блоків із формулами, кнопок керування та результатів обчислень.

Користувацький інтерфейс реалізовано за допомогою HTML, CSS та шаблонізатора Django Templates. HTML-шаблони відповідають за структуру сторінок, CSS-файли – за зовнішнє оформлення, а Django Templates забезпечують відображення динамічних даних, які надходять із backend-частини системи.

У структурі проєкту створено набір HTML-шаблонів:

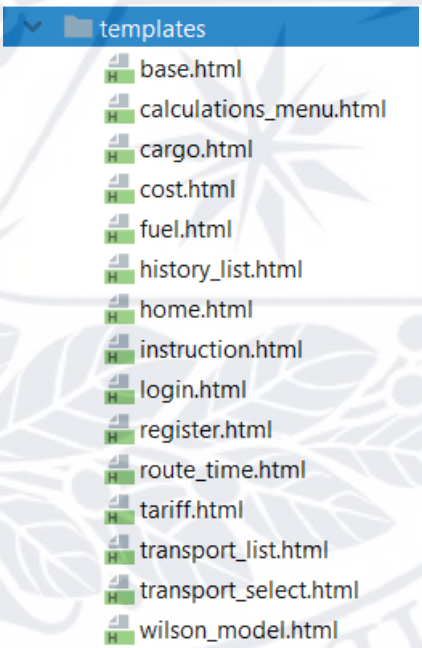

Рисунок 3.1 – HTML-шаблони

Такий поділ дозволяє розділити сторінки за їх функціональним призначенням і спростити підтримку програмного коду.

Основним шаблоном є base.html, який використовується як базова структура для інших сторінок. У ньому розміщено загальні елементи інтерфейсу: назву програмного застосунку, навігаційне меню та область для відображення основного вмісту сторінки. Меню містить переходи до головної сторінки, інструкції користувача, логістичного калькулятора, підбору транспорту та історії розрахунків. Завдяки цьому користувач може швидко переходити між основними функціональними частинами системи.

Головна сторінка (рис. 3.2) містить короткий опис призначення програмного застосунку та перелік основних функцій системи. На ній зазначено, що програмний модуль призначений для полегшення управління логістичним обслуговуванням підприємства. Також подано перелік доступних можливостей.

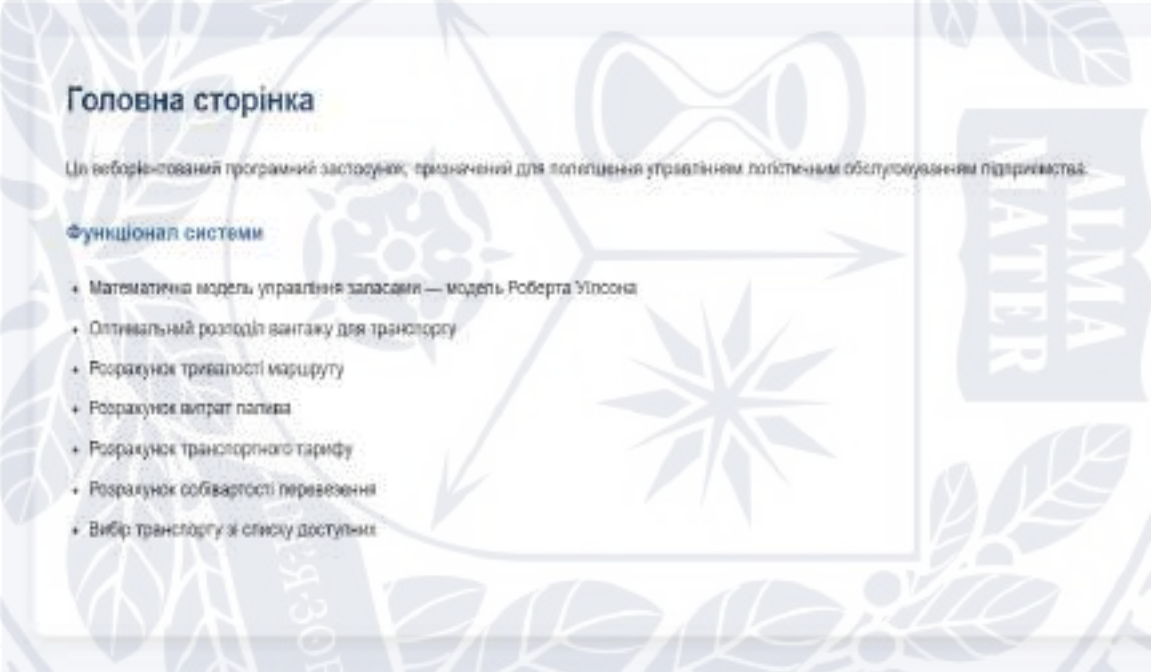

Рисунок 3.2 – Головна сторінка сайту

Окремо реалізовано сторінку інструкції користувача (рис. 3.3). Вона пояснює режими роботи системи та порядок використання програмного модуля. У системі передбачено два режими роботи: гість і авторизований користувач. Гість може виконувати логістичні розрахунки, але результати не зберігаються в історії. Авторизований користувач має можливість виконувати всі розрахунки та зберігати їх результати у власній історії. Такий підхід дозволяє використовувати

систему як для швидких одноразових розрахунків, так і для повноцінної роботи з накопиченням результатів.

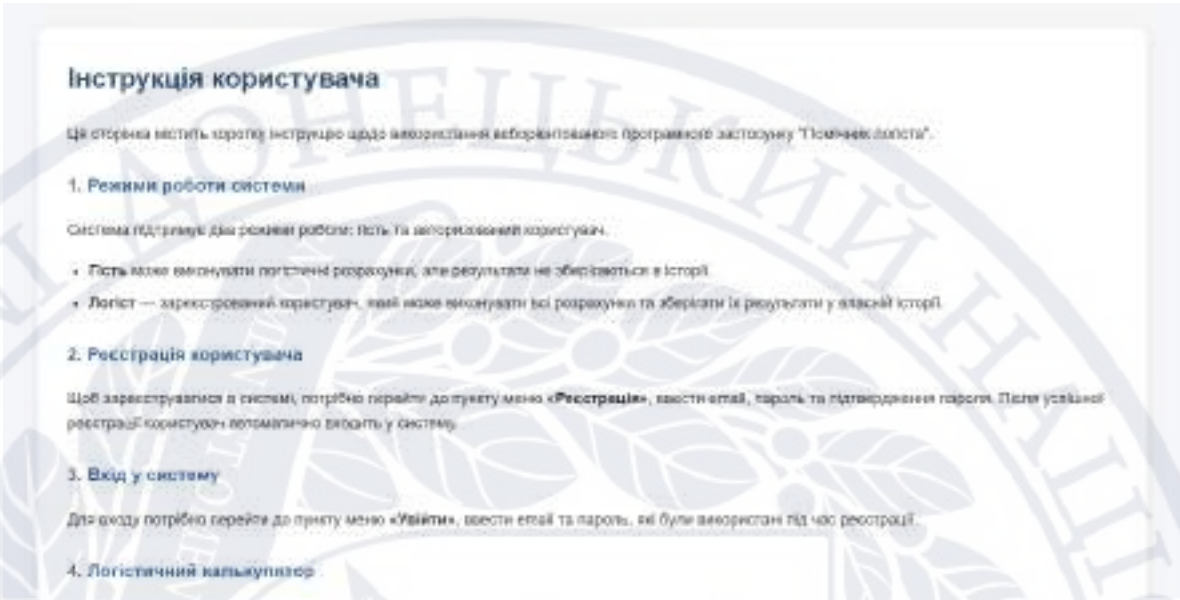

Рисунок 3.3 – Інструкція користування сайтом

Центральним елементом інтерфейсу є сторінка логістичного калькулятора (рис. 3.4). Вона містить меню вибору розрахунку, у якому користувач може обрати потрібний модуль: модель Уїлсона, розподіл вантажу, тривалість маршруту, витрати палива, транспортний тариф або собівартість перевезення. Кожен пункт меню оформлено у вигляді окремого блоку з назвою та коротким поясненням призначення відповідного розрахунку.

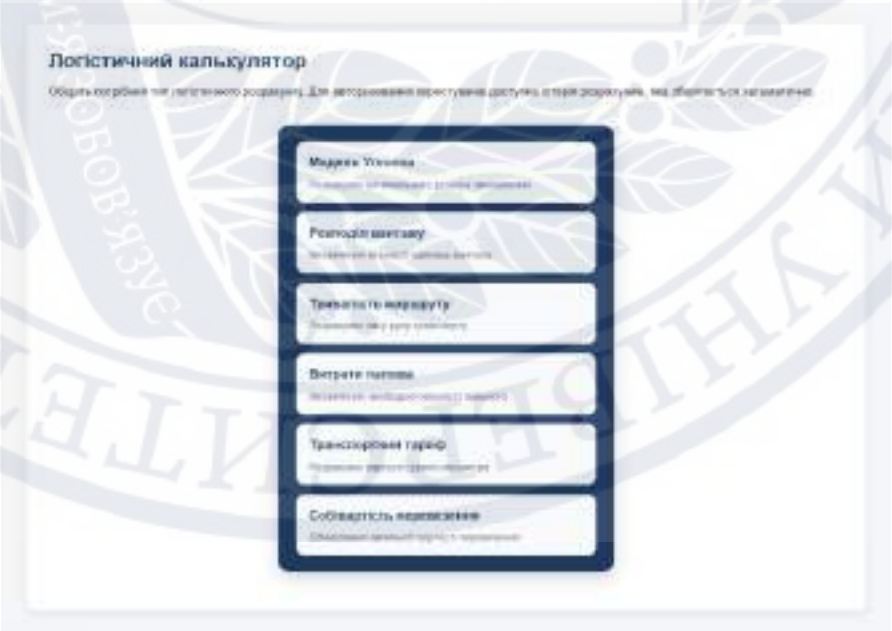

Рисунок 3.4 – Меню вибору логістичних розрахунків

Сторінки окремих розрахунків мають уніфіковану структуру. У верхній частині розміщено назву розрахунку та короткий опис його призначення. Основна частина сторінки поділена на два блоки: ліворуч розташовано форму введення даних, праворуч – формулу розрахунку з поясненням позначень. Така структура є зручною для користувача, оскільки він одночасно бачить, які дані потрібно ввести, та розуміє, за якою формулою буде виконано обчислення.

Наприклад, на сторінці моделі Роберта Уїлсона користувач вводить витрати на здійснення замовлення, інтенсивність використання запасів і витрати на зберігання одиниці запасу. Поруч відображається формула розрахунку оптимального розміру замовлення. Аналогічно побудовано всі інші розрахункові сторінки.

Окремим елементом інтерфейсу є сторінка підбору транспорту (рис. 3.5). На ній користувач може ввести параметри вантажу: вагу, об’єм та бажані витрати палива. Після цього система виконує автоматичний підбір транспортних засобів із бази даних. Також користувач має можливість переглянути повний список доступного транспорту (рис. 3.6). У таблиці транспорту відображаються модель транспортного засобу, вантажопідйомність, об’єм кузова та витрати палива на 100 км.

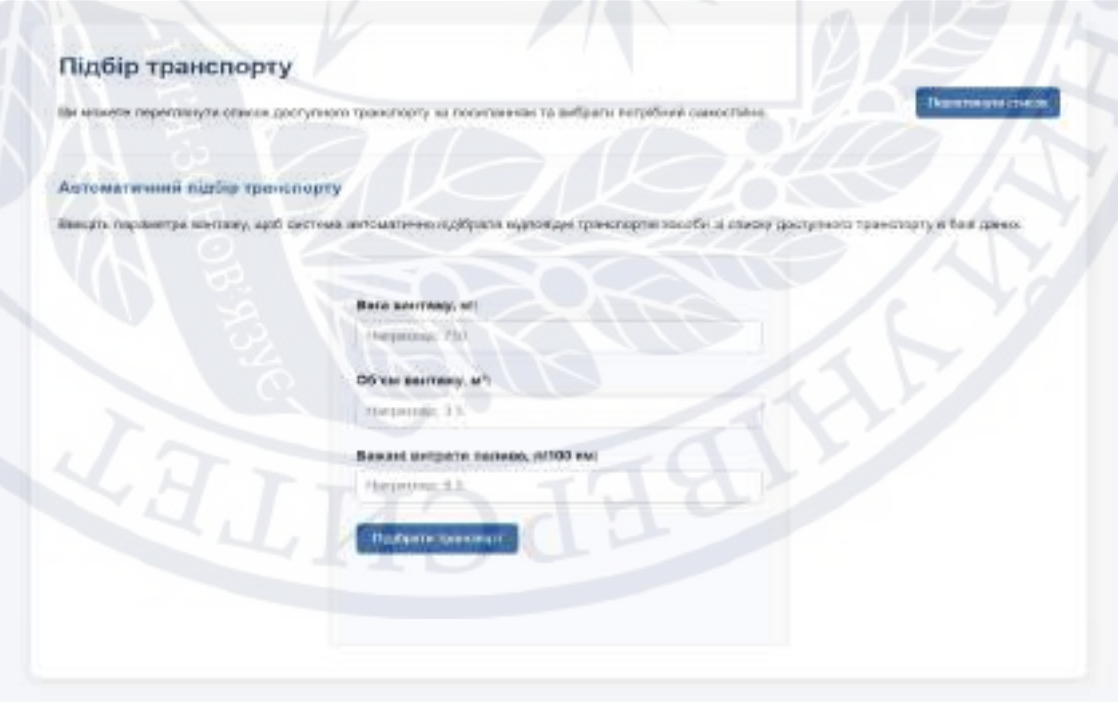

Рисунок 3.5 – Сторінка підбору транспорту

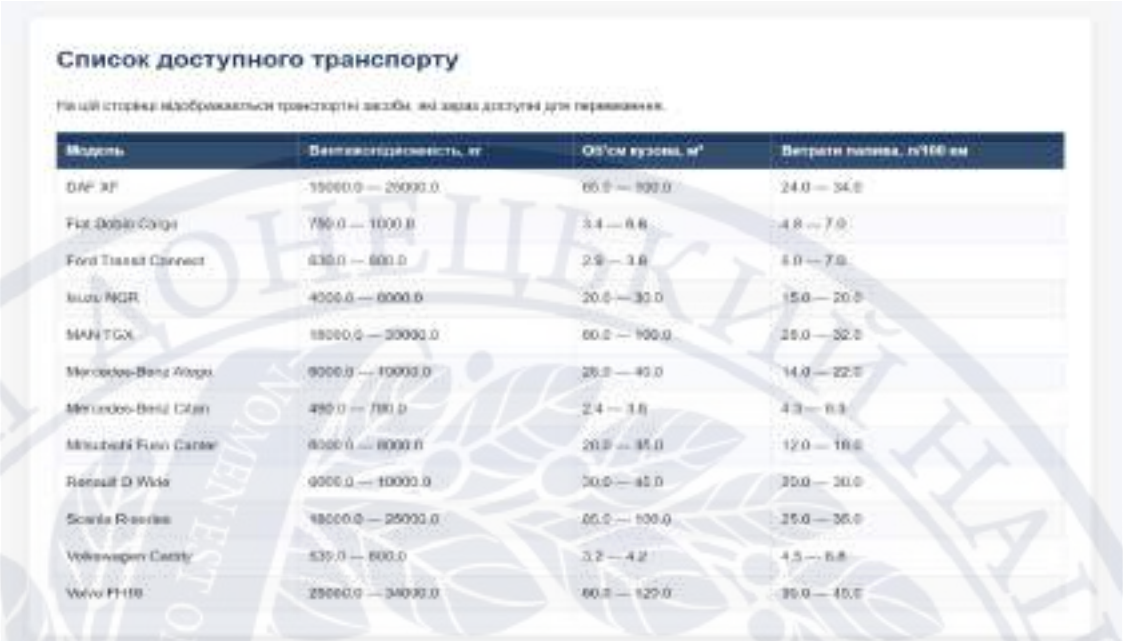

Рисунок 3.6 - Список доступних транспортних засобів

Для роботи з користувачами створено сторінки входу та реєстрації. Сторінка входу містить поля для введення email і пароля, а також кнопку авторизації. Сторінка реєстрації дозволяє створити обліковий запис логіста, ввівши ім’я користувача, email, пароль і підтвердження пароля. У разі помилок система відображає повідомлення, наприклад, про те, що користувач із таким іменем або email уже існує. Це підвищує зручність роботи та допомагає користувачеві правильно заповнити форму.

У нижній частині сторінок реалізовано інформаційний блок із відомостями про програмний застосунок і розробника. Це надає інтерфейсу завершеного вигляду та дозволяє ідентифікувати призначення програмного модуля.

Отже, інтерфейс користувача реалізовано у вигляді логічно пов’язаних сторінок, які забезпечують доступ до всіх основних функцій системи. Єдиний стиль оформлення, наявність навігаційного меню, зрозумілі форми введення даних і відображення формул роблять програмний модуль зручним для практичного використання.

3.2. Реалізація модулів логістичних розрахунків

Модуль логістичних розрахунків реалізовано як окремий функціональний блок системи. Доступ до нього здійснюється через сторінку «Логістичний калькулятор», на якій користувач бачить перелік доступних операцій. Кожен розрахунок представлений окремим елементом меню з короткою назвою та поясненням його призначення. Такий підхід дозволяє користувачеві швидко обрати потрібний тип обчислення без складної навігації. Вигляд розрахункових модулів наведений на рисунку 3.4.

До складу логістичного калькулятора входять сторінки для розрахунку моделі Роберта Уїлсона (рис. 3.7), розподілу вантажу, тривалості маршруту, витрат палива, транспортного тарифу та собівартості перевезення. Кожна сторінка має однакову логіку побудови: назва розрахунку, короткий опис, форма для введення початкових даних, кнопка запуску обчислення, блок із формулою та область для відображення результату.

Для кожного виду обчислення створено окремий HTML-шаблон. Це дозволяє розділити інтерфейсні елементи різних калькуляторів і спростити подальше редагування сторінок. Наприклад, одна сторінка містить поля для введення параметрів запасів, інша – поля для відстані та швидкості, наступна – поля для витрат і тарифу. Завдяки цьому кожна форма відповідає конкретній логістичній задачі.

Рисунок 3.7 – Реалізація сторінки розрахунку за моделлю Уїлсона

Передача даних у розрахункових модулях здійснюється через HTML-форми. Користувач вводить початкові значення у відповідні поля, після чого натискає кнопку «Розрахувати». Після цього дані передаються до серверної частини вебзастосунку, де їх обробляє відповідне представлення Django. Серверна частина отримує значення з форми, перевіряє їх, виконує необхідне обчислення та повертає результат назад до шаблону сторінки.

Загальна схема роботи розрахункового модуля має такий вигляд:

Сторінка калькулятора → Форма введення даних → Обробка запиту → Формування результату → Відображення на сторінці.

Після виконання обчислення результат не відкривається на окремій сторінці, а виводиться безпосередньо в межах тієї сторінки, де користувач вводив дані (рис. 3.13). Це робить роботу з калькулятором більш зручною, оскільки користувач одразу бачить підсумок розрахунку та за потреби може змінити введені параметри й повторити обчислення.

Крім форми введення даних, на сторінках розрахунків розміщено блок із формулою. Це дає змогу користувачеві бачити, за якою логікою виконується обчислення.

Важливою особливістю реалізації є зв’язок розрахункового модуля з механізмом збереження історії. Якщо користувач працює в гостьовому режимі, він може виконувати розрахунки, але їхні результати не зберігаються в базі даних. Якщо користувач авторизований, після виконання розрахунку система може зберегти результат у таблиці історії. Завдяки цьому користувач має можливість повернутися до попередніх обчислень і переглянути їх у відповідному розділі.

Окрім стандартних розрахункових сторінок, до функціоналу програмного модуля включено підбір транспорту. Цей блок відрізняється від звичайних калькуляторів тим, що він працює не лише з введеними користувачем параметрами, а й із записами, які зберігаються у базі даних. Користувач вводить характеристики вантажу, після чого система порівнює їх із характеристиками наявних транспортних засобів. У результаті користувач отримує перелік транспортних засобів, які відповідають заданим умовам.

Таким чином, реалізація модулів логістичних розрахунків полягає у створенні набору взаємопов’язаних вебсторінок, форм введення даних, серверної обробки запитів, механізму відображення результатів і збереження історії. На відміну від математичного моделювання, яке описує самі формули, програмна реалізація забезпечує їх практичне використання у вебсистемі.

3.3. Реалізація модуля управління процесом взаємодії з користувачами

Модуль управління процесом взаємодії з користувачами забезпечує персоналізовану роботу програмного модуля. Його основне призначення полягає в організації реєстрації, авторизації, виходу із системи, розмежуванні режимів роботи та збереженні результатів розрахунків для конкретного користувача.

У програмному модулі передбачено два режими роботи: гостьовий режим і режим авторизованого користувача. Такий підхід дозволяє використовувати систему як для швидких одноразових розрахунків, так і для повноцінної роботи із збереженням історії. Гостьовий режим не потребує створення облікового запису. Користувач може переглядати сторінки системи, відкривати інструкцію, користуватися калькулятором і виконувати підбір транспорту. Проте результати розрахунків у цьому режимі не зберігаються. Повідомлення про гостьовий режим має наступний вигляд:

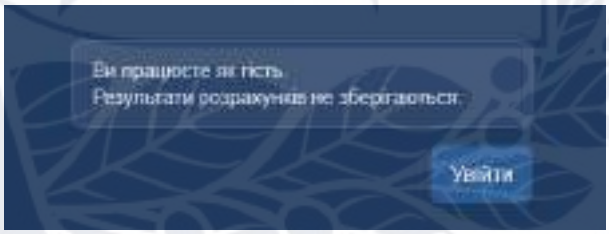

Рисунок 3.8 – Повідомлення про роботу користувача в гостьовому режимі

Як показано на рисунку 3.8, система повідомляє користувачу про роботу в гостьовому режимі та попереджає, що результати розрахунків не будуть збережені.

Авторизований користувач, зображений на рисунку 3.9, має розширені можливості. Після входу до системи він може не лише виконувати розрахунки, а

й зберігати їхні результати в базі даних. Також для такого користувача доступна історія розрахунків, у якій відображаються попередньо виконані операції. Це дозволяє аналізувати результати, порівнювати різні варіанти перевезень і використовувати збережені дані у подальшій роботі.

Рисунок 3.9 – Відображення авторизованого користувача в системі

Реєстрація користувача реалізована через окрему вебформу. Користувач вводить ім’я, email, пароль і підтвердження пароля. Після надсилання форми система перевіряє введені дані. Якщо користувач із таким іменем або email уже існує, система відображає відповідне повідомлення про помилку (рис. 3.17). Це дозволяє уникнути дублювання облікових записів і забезпечує коректну роботу механізму авторизації.

Авторизація виконується через сторінку входу (рис. 3.10). Користувач вводить email і пароль, після чого система перевіряє наявність відповідного облікового запису. У разі успішної авторизації користувач переходить у персоналізований режим роботи. В інтерфейсі відображається інформація про поточного користувача та доступна можливість виходу із системи.

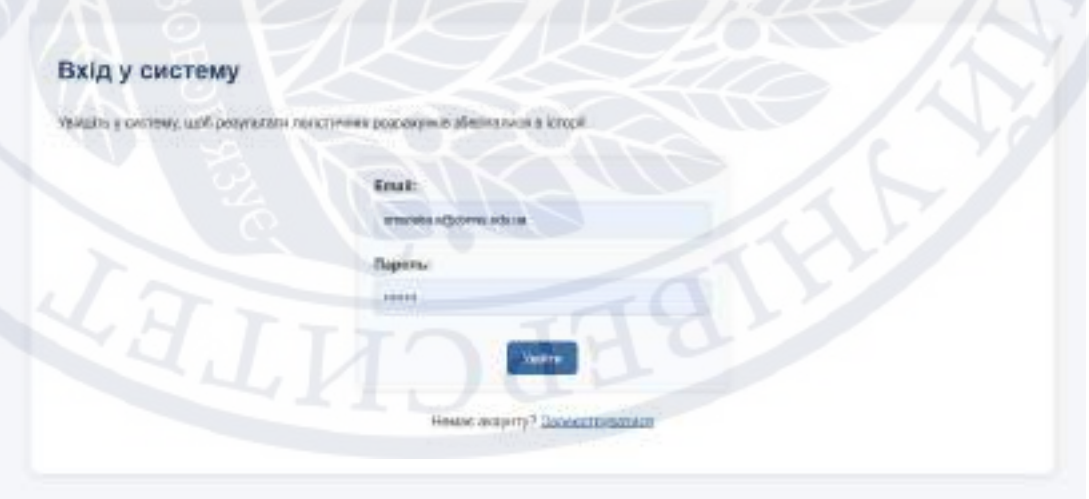

Рисунок 3.10 – Форма авторизації користувача

Вихід із системи є важливою функцією з точки зору безпеки й розмежування доступу. Після завершення роботи користувач може вийти з облікового запису, після чого система знову працюватиме для нього у гостьовому режимі. Це особливо важливо у випадках, коли програмний модуль використовується на спільному комп’ютері.

Історія розрахунків реалізована як персоналізований розділ системи та подана у вигляді таблиці (рис. 3.11). У ньому відображаються лише ті записи, які належать поточному авторизованому користувачеві. Це означає, що результати різних користувачів не змішуються між собою, а кожен користувач має доступ лише до власних розрахунків.

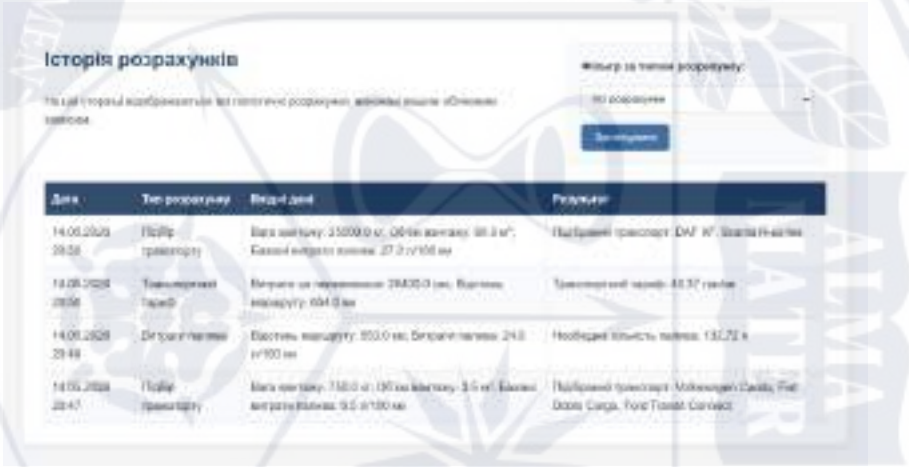

Рисунок 3.11 – Сторінка історії розрахунків

Для зручності роботи з великою кількістю записів у системі реалізовано фільтрацію історії за типом розрахунку. Користувач може обрати потрібну категорію і не шукати потрібні дані з великого списку власноруч:

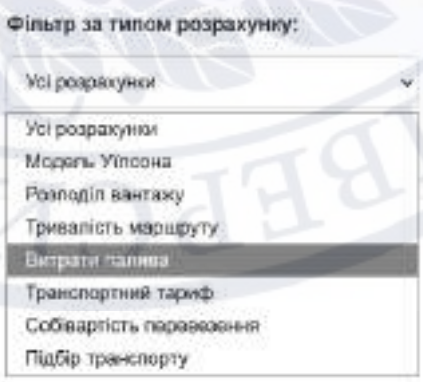

Рисунок 3.12 – Фільтрація історії за типом розрахунку

Збереження історії виконується через базу даних PostgreSQL. Під час створення запису система фіксує тип розрахунку, введені параметри, отриманий результат, дату виконання операції та користувача, якому належить цей запис. Завдяки цьому історія розрахунків стає не просто списком числових результатів, а структурованим набором даних, придатним для подальшого аналізу.

Робота з користувачами та історією здійснюється за допомогою можливостей Django. Для обробки реєстрації, авторизації та виходу використовуються відповідні механізми backend-частини. Взаємодія з базою даних здійснюється через Django ORM, що дозволяє працювати із записами як із Python-об'єктами без необхідності вручну писати SQL-запити [38; 39].

Окреме значення має адміністративна частина системи. Через Django Admin або pgAdmin 4 можна контролювати користувачів, записи історії та транспортні засоби. Це дозволяє адміністратору перевіряти правильність збереження даних, редагувати записи та контролювати роботу системи.

Таким чином, модуль управління взаємодією з користувачами функціонує не лише як набір окремих калькуляторів, а й як повноцінна вебсистема з персоналізованою роботою користувачів.

3.4. Тестування функціонування програмного модуля

Тестування програмного модуля виконувалося з метою перевірки коректності роботи основних функцій системи, правильності відображення інтерфейсу, виконання логістичних розрахунків, роботи з користувачами, підбору транспорту та збереження результатів у базі даних [30].

На першому етапі було перевірено структуру програмного проєкту. Перевірялося, чи правильно розміщено файли Django-проєкту, HTML-шаблони, CSS-файли, маршрути, моделі та представлення. Наявність окремих сторінок для головного меню, інструкції, калькулятора, транспорту, авторизації, реєстрації та історії підтверджує логічну організацію програмного модуля.

На другому етапі було протестовано інтерфейс користувача. Перевірялися відкриття головної сторінки, сторінки інструкції, меню логістичного

калькулятора, сторінок окремих розрахунків, сторінки підбору транспорту, списку транспорту, сторінок входу та реєстрації. У процесі тестування було встановлено, що сторінки відкриваються коректно, мають єдиний стиль оформлення та забезпечують доступ до основних функцій системи.

На третьому етапі перевірялася робота розрахункових модулів. Для кожної сторінки калькулятора вводилися тестові значення, після чого перевірялося, чи приймає система введені дані, чи виконує обчислення та чи відображає результат на сторінці. Окремо перевірялася зручність повторного виконання розрахунку після зміни вхідних параметрів:

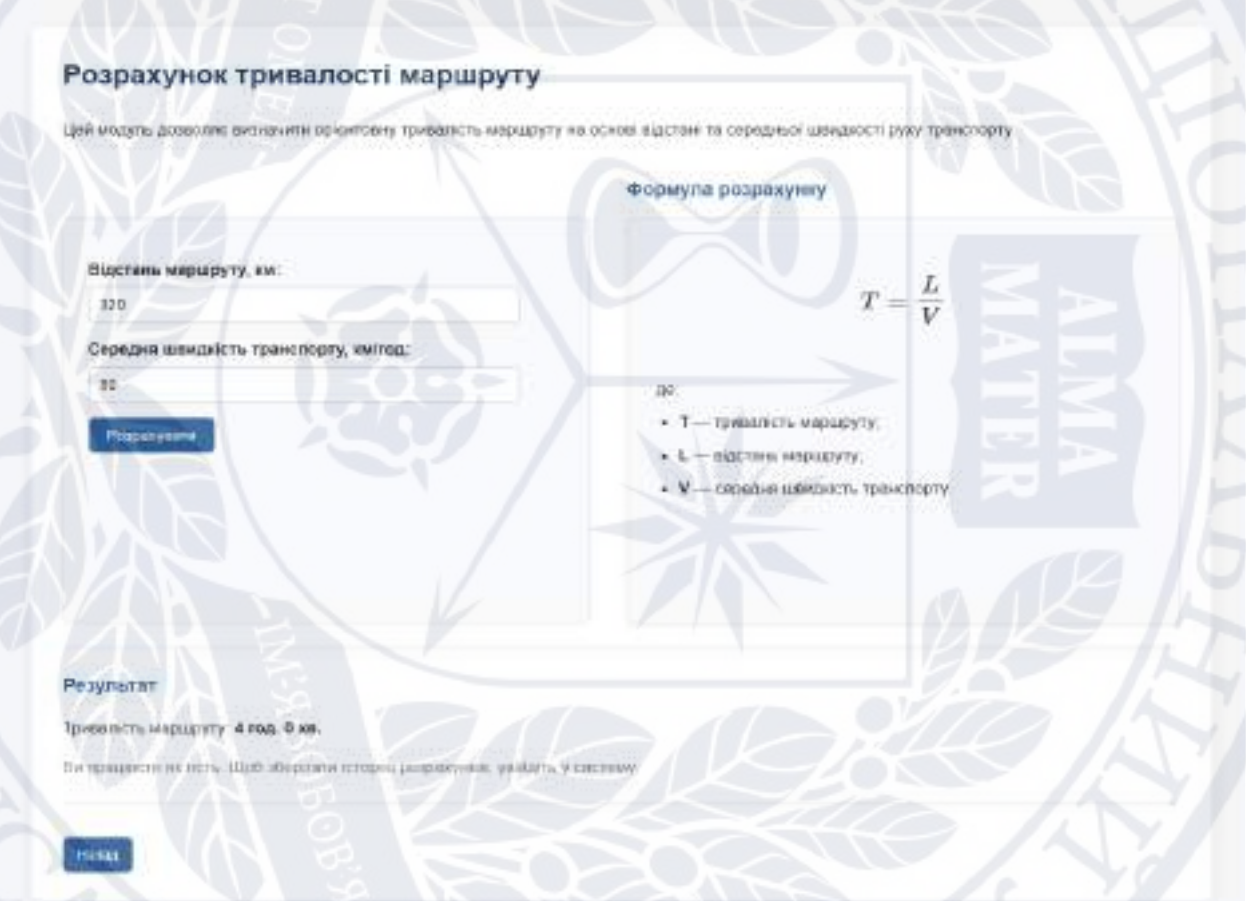

Рисунок 3.13 – Відображення результату розрахунку в гостьовому режимі

На четвертому етапі було протестовано підбір транспорту (рис. 3.14). Користувач вводив параметри вантажу, після чого система повинна була знайти транспортні засоби, які відповідають заданим умовам. Також перевірялася сторінка списку транспорту, на якій відображаються транспортні засоби з їх

характеристиками. Це дозволило переконатися, що дані з бази PostgreSQL коректно передаються в інтерфейс:

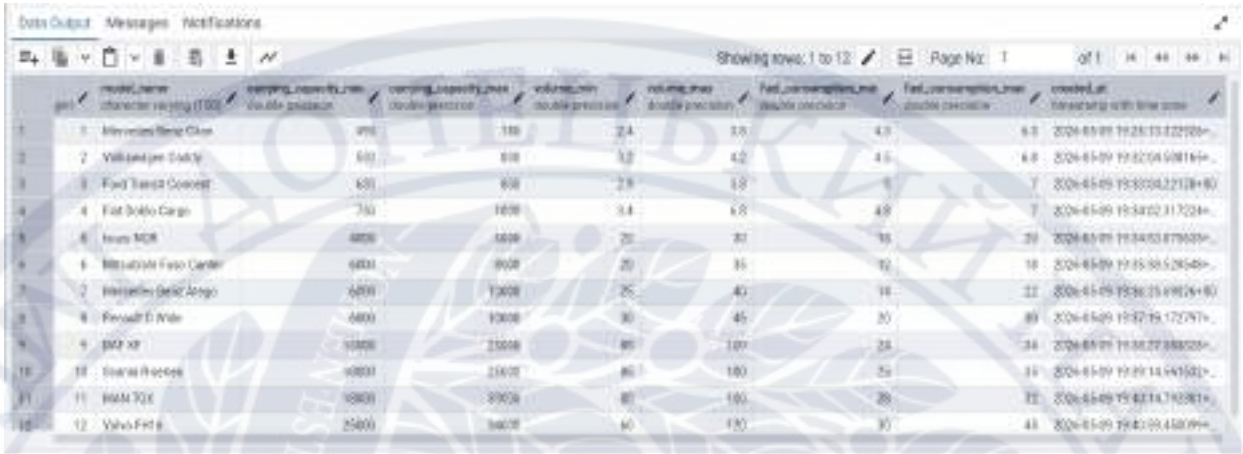

Рисунок 3.14 – Перевірка даних про транспортні засоби у базі PostgreSQL

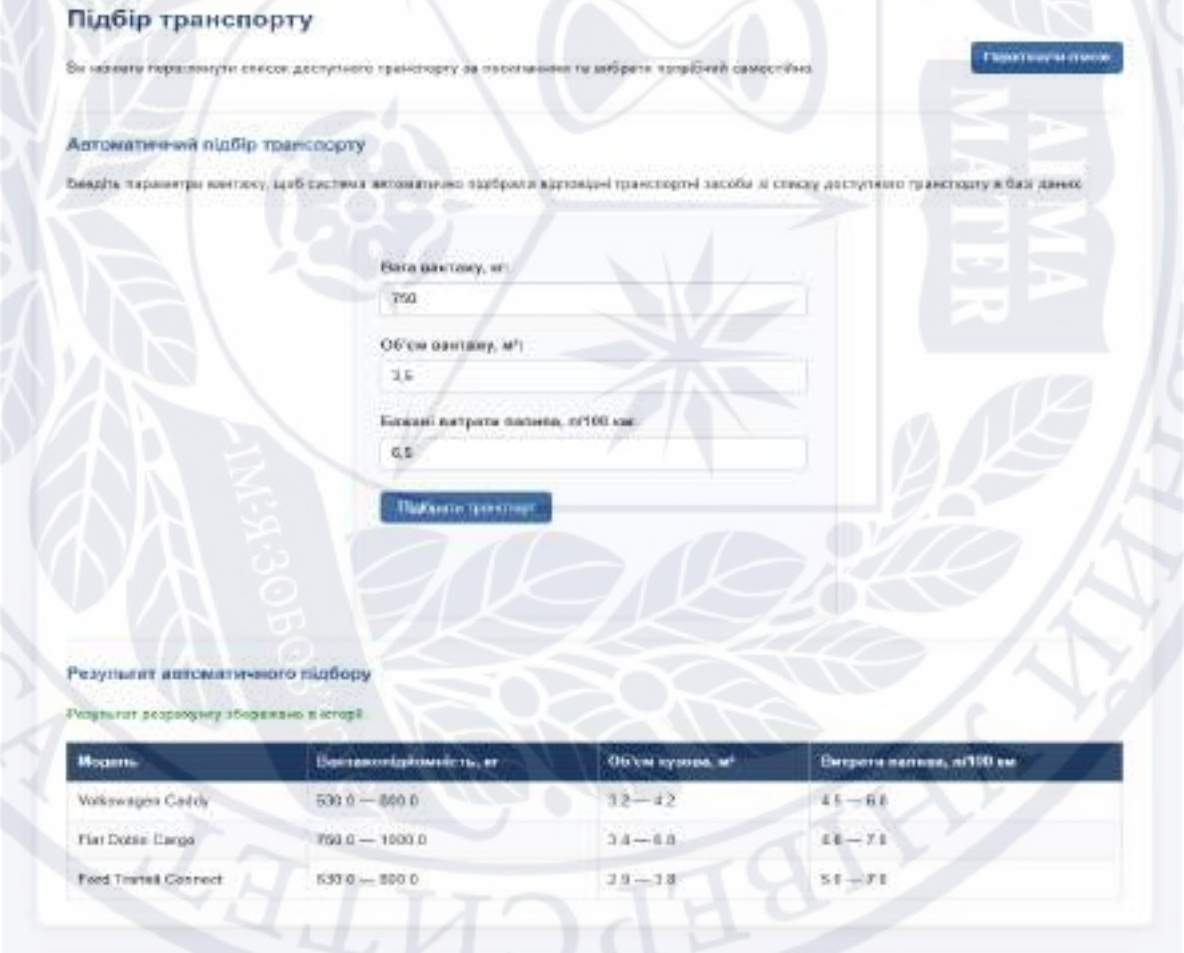

Рисунок 3.15 – Результат автопідбору транспорту

На п’ятому етапі було перевірено реєстрацію користувача. Тестування

включало створення нового облікового запису, введення email, пароля та підтвердження пароля. Після реєстрації користувача відповідний обліковий запис зберігається у таблиці користувачів бази даних. Це підтверджує коректну роботу механізму реєстрації та авторизації.

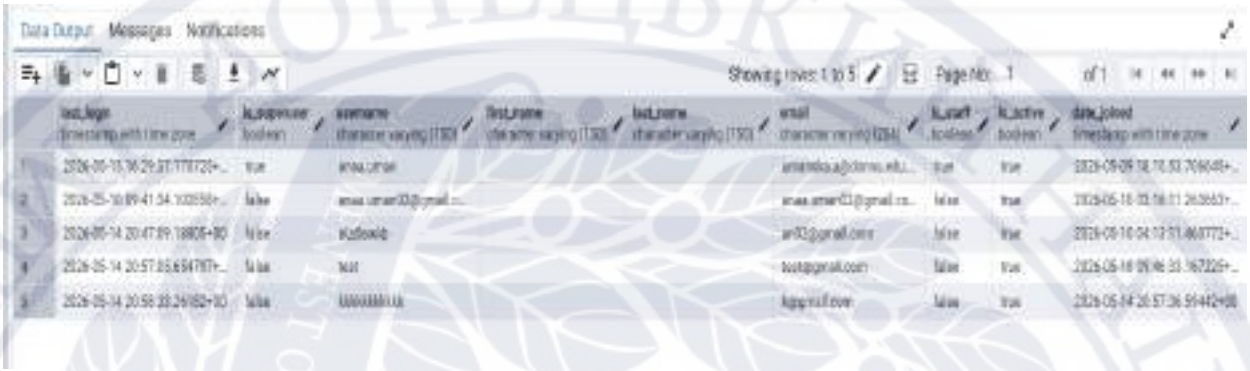

Рисунок 3.16 – Перевірка збереження облікових записів користувачів у PostgreSQL

Також перевірялася ситуація повторної реєстрації з уже наявними даними. У такому випадку система повинна повідомити користувача про помилку:

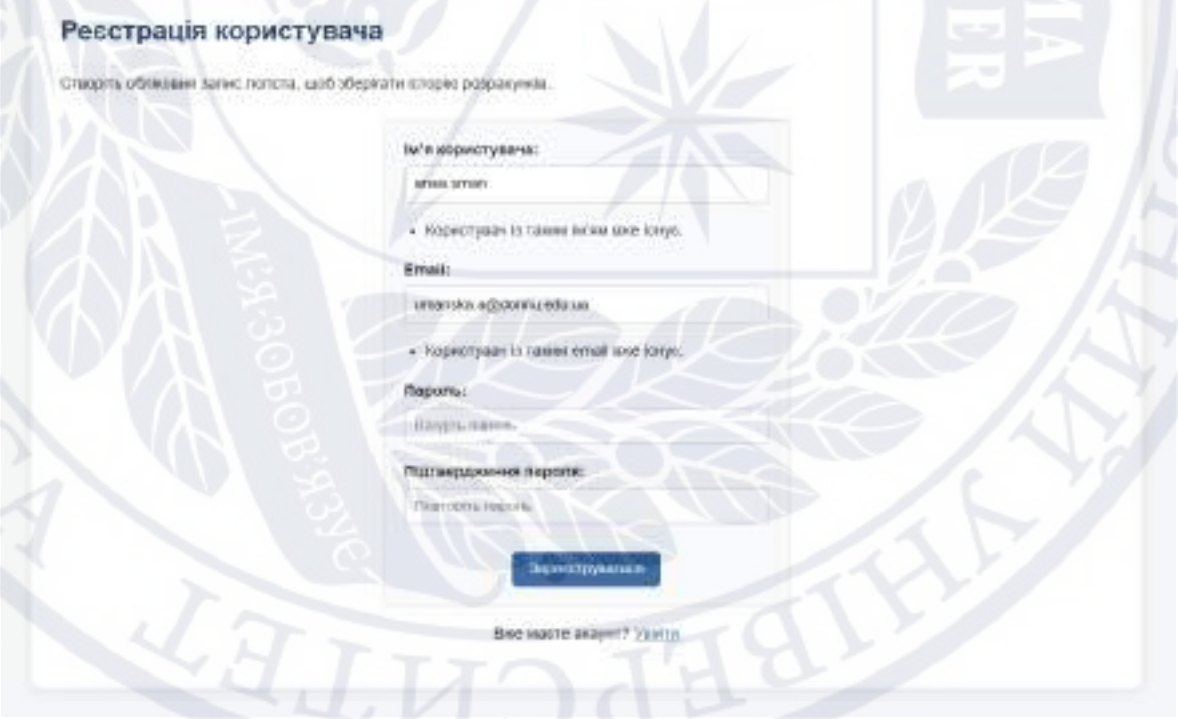

Рисунок 3.17 – Повідомлення про помилку під час повторної реєстрації користувача

На шостому етапі було протестовано авторизацію та вихід із системи. Перевірялося, чи може зареєстрований користувач увійти до системи, чи відображається його статус в інтерфейсі та чи працює кнопка виходу. Також перевірялося, чи змінюється режим роботи після виходу з облікового запису.

На сьомому етапі було перевірено збереження історії розрахунків. Для цього авторизований користувач виконував розрахунок, після чого перевірялося, чи з’являється відповідний запис у базі даних (рис. 3.19) і чи відображається він на сторінці історії (рис. 3.18). Додатково перевірялося, що користувач бачить лише власні записи, а не результати інших користувачів.

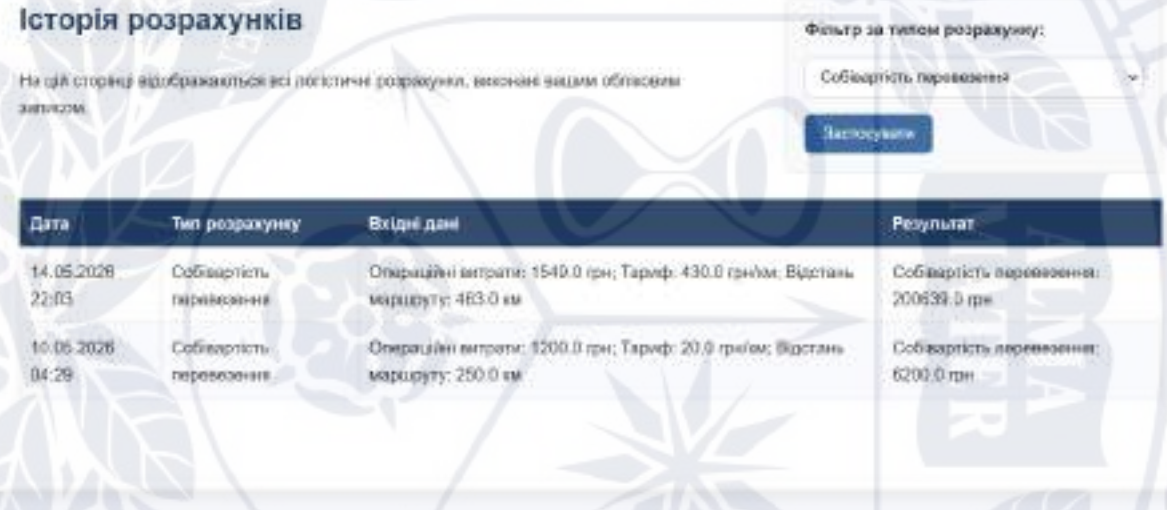

Рисунок 3.18 – Фільтрація історії за типом Собівартість перевезення

Рисунок 3.19 – Перевірка збереження історії логістичних розрахунків у PostgreSQL

Як видно з рисунка 3.19, у таблиці історії зберігаються результати виконаних логістичних розрахунків, дата створення запису та ідентифікатор користувача, якому належить відповідний розрахунок.

На восьмому етапі було перевірено роботу бази даних у pgAdmin 4. Переглядалися таблиці користувачів, транспорту та історії розрахунків. Це дозволило переконатися, що дані коректно зберігаються, оновлюються та використовуються програмним модулем.

Проведене тестування показало, що програмний модуль виконує основні функції відповідно до поставлених вимог. Сторінки відкриваються коректно, форми приймають дані користувача, розрахункові модулі формують результати, підбір транспорту працює на основі даних із бази, користувачі можуть реєструватися й авторизуватися, а результати розрахунків зберігаються в історії.

Отже, реалізований програмний модуль є працездатним і може бути використаний для автоматизації окремих задач управління логістичним обслуговуванням підприємства. Він забезпечує зручний інтерфейс, підтримує логістичні розрахунки, дозволяє виконувати підбір транспорту та зберігати результати роботи користувача у базі даних.

ВИСНОВКИ

У результаті виконання роботи було розроблено веборієнтований програмний модуль для управління логістичним обслуговуванням підприємства. Проведене дослідження підтвердило, що логістичне обслуговування є важливою складовою діяльності підприємства, оскільки воно впливає на своєчасність виконання замовлень, рівень витрат, якість сервісу та конкурентоспроможність підприємства. За результатами роботи зроблено висновки:

1. Дослідження сутності і ролі логістичного обслуговування встановило, що ці категорії охоплюють не лише транспортування вантажів, а й планування, обробку замовлень, вибір транспорту, контроль руху вантажу, Проведений аналіз особливостей побудови інформаційних систем у сфері логістики показав, що такі системи повинні забезпечувати оперативність обробки інформації, точність розрахунків, збереження даних, зручність користувацького інтерфейсу, безпеку та можливість розмежування доступу. Веборієнтований підхід є доцільним для розробки програмного модуля, оскільки забезпечує доступ до системи через браузер і не потребує встановлення окремого програмного забезпечення на комп’ютер користувача. інформування користувачів та післяпродажну підтримку. Визначено, що в сучасних умовах значну роль у підвищенні ефективності логістичних процесів відіграють інформаційні системи, які забезпечують швидку обробку даних, автоматизацію розрахунків і підтримку прийняття управлінських рішень.

2. Під час аналізу існуючих програмних рішень у сфері логістики було розглянуто корпоративні та спеціалізовані системи, зокрема ERP-, TMS-, WMS- та CRM-рішення. Визначено, що такі системи мають широкий функціонал і можуть ефективно використовуватися великими підприємствами, однак часто є складними, дорогими у впровадженні та надлишковими для виконання окремих логістичних розрахунків. Це підтвердило доцільність створення власного програмного модуля, орієнтованого на конкретні задачі логістичного обслуговування.

3. Під час визначення засобів та інструментів розробки програмного модуля, було обгрунтувано доцільність використанння Python 3, Django 5, PostgreSQL, Psycopg3, pgAdmin 4, HTML, CSS, Django Templates, Django ORM, Django Admin та середовища розробки PyCharm. Обраний набір технологій дозволив реалізувати серверну логіку, користувацький інтерфейс, роботу з базою даних, авторизацію користувачів, адміністрування даних і збереження історії виконаних розрахунків.

4. Були сформовані функціональні вимоги до програмного модуля. Визначено, що система повинна забезпечувати перегляд головної сторінки та інструкції, реєстрацію й авторизацію користувачів, введення параметрів логістичних розрахунків, виконання обчислень, підбір транспорту, відображення результатів, збереження історії розрахунків і адміністрування основних даних.

5. Формалізація основних логістичних операцій, які реалізовано у програмному модулі, охоплювала розрахунки за моделлю оптимального розміру замовлення Уїлсона, визначення одиниць вантажу для розміщення у транспортному засобі, тривалості маршруту, необхідного ресурса палива, транспортного тарифу та собівартості перевезення. Це дозволило перетворити логістичні задачі на чіткі алгоритми, придатні для програмної реалізації.

6. У процесі проєктування та реалізації програмного модуля було визначено основні складові системи: клієнтську частину, серверну частину та базу даних. Клієнтська частина забезпечує взаємодію користувача із системою через вебсторінки, серверна частина відповідає за обробку запитів, авторизацію користувачів і виконання розрахунків, а база даних використовується для збереження інформації про користувачів, транспортні засоби та історію логістичних розрахунків.

Отже, розроблений програмний модуль відповідає визначеним вимогам, забезпечує автоматизацію основних логістичних розрахунків і може бути використана для підвищення ефективності управління логістичним обслуговуванням підприємства.

СПИСОК ВИКОРИСТАНИХ ПОСИЛАНЬ

1. Зрібнєва І. П. Логістика: навчальний посібник для студентів ЗВО. Одеса: Олді+, 2025. 156 с. URL:https://archer.chnu.edu.ua/bitstream/handle/123456789/12107/Зрибнєва%20І.П._Логістика_навчальний%20посібник_лабораторні%20роботи_2025.pdf?sequence=1&isAllowed=y (дата звернення: 22.05.2026)

2. Піддубна Н. М. Конспект лекцій з навчальної дисципліни «Логістика» для здобувачів першого рівня вищої освіти. Одеса: Одеський національний морський університет, 2025. URL: 4_лекции з дисциплини логістика пм.pdf (дата звернення: 22.05.2026)

3. Балабанова Л. В., Германчук А. Н. Логістика: підручник. 2-ге вид., стереотип. Львів: «Магнолія 2006», 2025. 365 с. URL: https://magnolia.lviv.ua/wp-content/uploads/2024/12/Lohistyka_zmist2025.pdf (дата звернення: 22.05.2026)

4. Боковець В. В., Безсмертна О. В. Функціональна логістика: навчальний посібник. Вінниця: ВНТУ, 2025. (дата звернення: 23.05.2026)

5. Гринів Н. Т., Наконечна Т. В., Балик У. О., Костюк О. С., Литвиненко С. Л. Транспортна логістика: навчальний посібник. Київ, 2025. (дата звернення: 21.05.2026)

6. Ільченко Т. Управління якістю логістичних послуг підприємства. Економіка та суспільство. 2025. URL: Перегляд управління якістю логістичних послуг підприємства (дата звернення: 23.05.2026)

7. Служенко А. Теоретичні, стратегічні та операційні аспекти управління якістю логістичного сервісу. Економіка та суспільство. 2025. URL: https://economyandsociety.in.ua/index.php/journal/article/view/6246/6189 (дата звернення: 22.05.2026)

8. Мельникова К. В. Логістичний менеджмент як основа управління ефективністю логістичних процесів. Бізнес Інформ. 2024. № 11. С. 317–322. URL: https://www.business-inform.net/export_pdf/business-inform-2024-11_0-pages-317_322.pdf (дата звернення: 22.05.2026)

9. Боковець В., Шварц І., Верьовкін П. Сучасні управлінські рішення в логістичному менеджменті. Вісник Хмельницького національного університету. Серія: Економічні науки. 2024. URL: https://heraldes.khmnu.edu.ua/index.php/heraldes/article/view/1190/1211 (дата звернення: 21.05.2026)

10. Штельмашук М. Цифровізація та автоматизація логістичних процесів: сучасний стан та перспективи. *Економіка та суспільство*. 2024. № 68. URL: Перегляд цифровізація та автоматизація логістичних процесів: сучасний стан та перспективи (дата звернення: 20.05.2026)

11. Канцедал Н. Цифровізація логістики: нові технології для оптимізації процесів постачання, зберігання та транспортування товарів. *Економічний простір*. 2025. URL: https://economic-prostir.com.ua/wp-content/uploads/2025/03/199-45-51-kanczedal.pdf (дата звернення: 21.05.2026)

12. Глущенко С. Д. Трансформація логістики під впливом цифровізації. Причорноморські економічні студії. 2025. URL: https://bses.in.ua/journals/2025/92_2025/26.pdf (дата звернення: 22.05.2026)

13. Гриценко С. І., Миколаєнко Д. С. Цифрова трансформація логістичних послуг в Україні. Економічний вісник Донбасу. 2025. URL: https://www.evd-journal.org/download/2025/2/06-Hrytsenko.pdf (дата звернення: 20.05.2026)

14. Шарко В. Цифрова логістика як ключовий елемент сталого розвитку бізнесу. Київ: Київський національний економічний університет імені Вадима Гетьмана, 2025. URL: https://ir.kneu.edu.ua/server/api/core/bitstreams/54896e94-e5f9-4806-a71c-fa13200b87fb/content (дата звернення: 20.05.2026)

15. Литовченко О. Сучасні методи логістики як інструмент цифрової трансформації бізнесу. Економіка та суспільство. 2024. URL: https://dees.iei.od.ua/index.php/journal/article/view/859/828 (дата звернення: 22.05.2026)

16. Пелех К., Демидчук Л. Інтернет речей в сучасній транспортній логістиці. Вісник Хмельницького національного університету. Серія: Економічні науки. 2024. URL: https://heraldes.khmnu.edu.ua/index.php/heraldes/article/

view/527/531 (дата звернення: 23.05.2026)

17. Чобіток В., Літвінчик С. Системи інформаційного забезпечення транспортної логістики в підприємницькій діяльності. Вісник Хмельницького національного університету. Серія: Економічні науки. 2024. URL: перегляд системи інформаційного забезпечення транспортної логістики в підприємницькій діяльності (дата звернення: 23.05.2026)

18. Ільченко Т. Формування ефективної системи логістичного менеджменту у сфері реалізації продукції підприємства. Економіка та суспільство. 2024. URL: https://economyandsociety.in.ua/index.php/journal/article/view/4720/4661 (дата звернення: 21.05.2026)

19. Воронько-Невіднича Т. Логістичні аспекти в системі управління бізнес-процесами підприємства. Вісник Хмельницького національного університету. Серія: Економічні науки. 2024. URL: перегляд логістичні аспекти в системі управління бізнес-процесами підприємства (дата звернення: 21.05.2026)

20. Алєксєєв М. Оптимізація основних управлінських підходів у сфері логістики в умовах цифрової трансформації. *Економіка та суспільство*. 2025. URL: перегляд оптимізація основних управлінських підходів в логістиці в умовах цифрової трансформації (дата звернення: 21.05.2026)

21. Самойленко Б. Механізми та перспективи імплементації міжнародних практик регулювання ринку логістичних послуг в Україні. Вісник Хмельницького національного університету. Серія: Економічні науки. 2026. URL: https://heraldes.khmnu.edu.ua/index.php/heraldes/article/view/2701/2775 (дата звернення: 22.05.2026)

22. Тимчик А. М. Особливості управління логістичними процесами електронної комерції підприємства. Бізнес Інформ. 2024. № 2. С. 272–278. URL: https://www.business-inform.net/export_pdf/business-inform-2024-2_0-pages-272_278.pdf (дата звернення: 22.05.2026)

23. Губарєв Р. В. Методичні засади формування логістичного механізму діяльності підприємств. Бізнес Інформ. 2024. № 5. С. 220–226. URL: https://www.business-inform.net/export_pdf/business-inform-2024-5_0-pages-

220_226.pdf (дата звернення: 20.05.2026)

24. Харун О. Управління логістичними ризиками на підприємстві. Вісник Хмельницького національного університету. Серія: Економічні науки. 2024. URL: https://heraldes.khmnu.edu.ua/index.php/heraldes/article/view/1408/1437 (дата звернення: 20.05.2026)

25. Корман І. І., Семенда О. В., Мазур Ю. П. Вплив цифрових технологій на управління каналами розподілу та логістику в умовах глобальної економіки. Економіка та суспільство. 2025. № 71. С. 213–221. URL: https://economyandsociety.in.ua/index.php/journal/article/view/5480/5419 (дата звернення: 20.05.2026)

26. Азарова А. О., Юрчук Н. П., Муращенко О. Г. Інформаційні системи і технології. Частина 1 : навчальний посібник. Вінниця: ВНТУ, 2024. 152 с. URL: https://pdf.lib.vntu.edu.ua/books/2025/Azarova_P1_2024_152.pdf (дата звернення: 22.05.2026)

27. Трофименко О. Г., Прокоп Ю. В., Логінова Н. І., Задерейко О. В. Тестування та забезпечення якості програмних систем: навчально-методичний посібник. Одеса: Фенікс, 2024. URL: Тестування та забезпечення якості програмних систем. Навчально-методичний посібник (дата звернення: 22.05.2026)

28. Іваненко Л. М. Інтегральний підхід до логістики постачання, виробництва та збуту на промисловому підприємстві. Бізнес Інформ. 2024. № 4. С. 315–325. URL: https://www.business-inform.net/export_pdf/business-inform-2024-4_0-pages-315_325.pdf (дата звернення: 20.05.2026)

29. Македон В. Інтеграція цифрових інструментів у міжнародні логістичні процеси. Економіка та суспільство. 2024. URL: https://economyandsociety.in.ua/index.php/journal/article/view/4486/4425 (дата звернення: 22.05.2026)

30. Ануфрієва Т. Г. Логістичні інновації в холодових ланцюгах постачання. Бізнес Інформ. 2024. № 6. С. 235–241. URL: https://www.business-inform.net/export_pdf/business-inform-2024-6_0-pages-235_241.pdf (дата

звернення: 22.05.2026)

31. Саченко І. А. Проєктування інформаційних систем: конспект лекцій. Київ: КНУБА, 2024. 96 с. URL: https://library.knuba.edu.ua/books/22_1_24.pdf (дата звернення: 20.05.2026)

32. Мулеса О. Ю. Інформаційні системи та реляційні бази даних: навчальний посібник. Ужгород: ДВНЗ «Ужгородський національний університет», 2023. URL: https://dspace.uzhnu.edu.ua/server/api/core/bitstreams/4df72779-9c93-4422-81bb-c4d8589fcb53/content (дата звернення: 22.05.2026)

33. Зінов'єва О. Г., Шаров С. В. Проектування інформаційних систем: конспект лекцій. Запоріжжя: Таврійський державний агротехнологічний університет імені Дмитра Моторного, 2024. 148 с. URL: https://www.tsatu.edu.ua/kn/wp-content/uploads/sites/16/pis.pdf (дата звернення: 21.05.2026)

34. Паршина О. А., Паршин Ю. І., Косарєв В. М. Інформаційні системи і технології в управлінні об'єктами критичних інфраструктур: Навч. посібник. Дніпро: УМФС, 2024. С. 260. URL: https://files.znu.edu.ua/files/Bibliobooks/Inshi84/0064143.pdf (дата звернення: 22.05.2026)

35. Берко А. Ю., Верес О. М., Пасічник В. В. Моделі баз даних та знань: підручник. Львів: Магнолія, 2024. URL: https://magnolia.lviv.ua/wp-content/uploads/2024/04/Modeli-baz-danykh-ta-znan.pdf (дата звернення: 23.05.2026)

36. Python Software Foundation. What's New In Python 3.12. 2023. URL: https://docs.python.org/3/whatsnew/3.12.html (дата звернення: 23.05.2026)

37. Django Software Foundation. Django 5.0 release notes. 2023. URL: https://docs.djangoproject.com/en/5.0/releases/5.0/ (дата звернення: 20.05.2026)

38. Django Software Foundation. Using the Django authentication system. 2026. URL: https://docs.djangoproject.com/en/6.0/topics/auth/default/ (дата звернення: 23.05.2026)

39. Django Software Foundation. User authentication in Django. 2026. URL:

https://docs.djangoproject.com/en/6.0/topics/auth/ (дата звернення: 23.05.2026)

40. PostgreSQL Global Development Group. PostgreSQL 16 Release Notes. 2023. URL: https://www.postgresql.org/docs/release/16.0/ (дата звернення: 23.05.2026)

41. PostgreSQL Global Development Group. PostgreSQL Documentation: Constraints. 2026. URL: https://www.postgresql.org/docs/current/ddl-constraints.html (дата звернення: 23.05.2026)

42. Psycopg Project. Psycopg 3 Documentation. 2026. URL: https://www.psycopg.org/psycopg3/docs/ (дата звернення: 23.05.2026)

43. pgAdmin Development Team. pgAdmin 4 Documentation. 2026. URL: https://www.pgadmin.org/docs/pgadmin4/latest/ (дата звернення: 22.05.2026)

44. WHATWG. HTML Living Standard. 2026. URL: https://html.spec.whatwg.org/multipage/ (дата звернення: 21.05.2026)

45. MDN Web Docs. CSS Reference. 2025. URL: https://developer.mozilla.org/en-US/docs/Web/CSS/Reference (дата звернення: 20.05.2026)

46. JetBrains. Getting started with PyCharm. 2026. URL: https://www.jetbrains.com/help/pycharm/getting-started.html (дата звернення: 20.05.2026)

47. ISO/IEC 25010:2023. Systems and software engineering – Systems and software Quality Requirements and Evaluation – Product quality model. Geneva: International Organization for Standardization, 2023. URL: https://www.iso.org/standard/78176.html (дата звернення: 20.05.2026)

48. ISO/IEC 25002:2024. Systems and software engineering – Quality model overview and usage. Geneva: International Organization for Standardization, 2024. URL: https://www.iso.org/standard/78175.html (дата звернення: 20.05.2026)

49. SAP. SAP S/4HANA Cloud Public Edition: Supply Chain Product Tour. 2026. URL: https://www.sap.com/ukraine/products/erp/s4hana/features/supply-chain/product-tour.html (дата звернення: 21.05.2026)

50. SAP. Transportation Management and Logistics System. 2026. URL:

https://www.sap.com/products/scm/transportation-logistics.html (дата звернення: 21.05.2026)

51. Microsoft. Managing Inventory in Microsoft Dynamics 365 Business Central. 2026. URL: https://learn.microsoft.com/en-us/dynamics365/business-central/inventory-manage-inventory (дата звернення: 20.05.2026)

52. Odoo. Inventory Management – Odoo 18.0 Documentation. 2025. URL: https://www.odoo.com/documentation/18.0/applications/inventory_and_mrp/inventory/warehouses_storage/inventory_management.html (дата звернення: 20.05.2026)

53. Oracle. Oracle Transportation Management. 2026. URL: https://www.oracle.com/scm/logistics/transportation-management/ (дата звернення: 20.05.2026)

54. Manhattan Associates. Manhattan Active Warehouse Management. 2026. URL: https://www.manh.com/solutions/supply-chain-management-software/warehouse-management (дата звернення: 21.05.2026)

55. Infor. Infor Warehouse Management System (WMS). 2026. URL: https://www.infor.com/solutions/scm/warehouse-management-system (дата звернення: 21.05.2026)

56. Salesforce. Service Cloud: AI-powered Customer Service Agent Console. 2025. URL: https://www.salesforce.com/service/cloud/ (дата звернення: 21.05.2026)

57. Bitrix24. Free online workspace for your business: CRM, tasks, sites, online store. 2025. URL: https://www.bitrix24.com/ (дата звернення: 20.05.2026)

58. Reference Global. Digital Transformation of Logistics: How Modern Technologies Are Transforming Logistics and Supply Chains. 2024. URL: https://reference-global.com/article/10.2478/czoto-2024-0042 (дата звернення: 20.05.2026)

59. Gartner. Magic Quadrant for Warehouse Management Systems. 2025. URL: https://www.gartner.com/en/documents/6411575 (дата звернення: 22.05.2026)

60. PostgreSQL Global Development Group. PostgreSQL Documentation: Indexes. 2026. URL: https://www.postgresql.org/docs/current/indexes.html (дата звернення: 23.05.2026)

ДОДАТКИ

ДОДАТОК А

```
from django.db import models
from django.contrib.auth.models import User

# Модель транспортного засобу
class Transport(models.Model):
    model_name = models.CharField(
        max_length=150,
        verbose_name='Модель автомобіля'
    )

    carrying_capacity_min = models.FloatField(
        verbose_name='Мінімальна вантажопідйомність, кг'
    )

    carrying_capacity_max = models.FloatField(
        verbose_name='Максимальна вантажопідйомність, кг'
    )

    volume_min = models.FloatField(
        verbose_name='Мінімальний об'єм кузова, м³'
    )

    volume_max = models.FloatField(
        verbose_name='Максимальний об'єм кузова, м³'
    )

    fuel_consumption_min = models.FloatField(
        verbose_name='Мінімальні витрати палива, л/100 км'
    )

    fuel_consumption_max = models.FloatField(
        verbose_name='Максимальні витрати палива, л/100 км'
    )

    created_at = models.DateTimeField(
        auto_now_add=True,
        verbose_name='Дата додавання'
    )

    class Meta:
        verbose_name = 'Транспорт'
        verbose_name_plural = 'Транспорт'

    def __str__(self):
        return self.model_name
```

```
# Модель історії логістичних розрахунків
class CalculationHistory(models.Model):
    CALCULATION_TYPES = [
        ('wilson', 'Модель Уїлсона'),
        ('cargo', 'Розподіл вантажу'),
        ('time', 'Тривалість маршруту'),
        ('fuel', 'Витрати палива'),
        ('tariff', 'Транспортний тариф'),
        ('cost', 'Собівартість перевезення'),
        ('transport', 'Підбір транспорту'),
    ]

    user = models.ForeignKey(
        User,
        on_delete=models.CASCADE,
        verbose_name='Користувач'
    )

    calculation_type = models.CharField(
        max_length=30,
        choices=CALCULATION_TYPES,
        verbose_name='Тип розрахунку'
    )

    input_data = models.TextField(
        verbose_name='Вхідні дані'
    )

    result = models.TextField(
        verbose_name='Результат'
    )

    created_at = models.DateTimeField(
        auto_now_add=True,
        verbose_name='Дата розрахунку'
    )

    class Meta:
        verbose_name = 'Історія розрахунку'
        verbose_name_plural = 'Історія розрахунків'
        ordering = ['-created_at']

    def __str__(self):
        return f'{self.user.username} — {self.get_calculation_type_display()}'
```

ДОДАТОК Б

```
import math
from django.shortcuts import render
from django.contrib.auth.decorators import login_required
from history.models import CalculationHistory
from transport.models import Transport
from calculations.forms import WilsonForm
from transport.forms import TransportSelectionForm

# Функції виконання логістичних розрахунків
def calculate_wilson_model(order_cost, demand_intensity, storage_cost):

    result = math.sqrt((2 * order_cost * demand_intensity) / storage_cost)
    return round(result, 2)

def calculate_cargo_distribution(cargo_area, available_area):

    result = available_area / cargo_area
    return int(result)

def calculate_route_time(distance, speed):

    total_hours = distance / speed

    hours = int(total_hours)
    minutes = round((total_hours - hours) * 60)

    if minutes == 60:
        hours += 1
        minutes = 0

    return hours, minutes

def calculate_fuel_amount(distance, fuel_consumption):

    result = (distance * fuel_consumption) / 100
    return round(result, 2)

def calculate_transport_tariff(transportation_cost, distance):

    result = transportation_cost / distance
    return round(result, 2)

def calculate_transport_cost(operation_cost, tariff, distance):

    result = operation_cost + tariff * distance
    return round(result, 2)

# Обробка розрахунку за моделлю Роберта Уілсона
def wilson_calculation(request):

    form = WilsonForm()
    result = None
    is_saved = False

    if request.method == "POST":
        form = WilsonForm(request.POST)

        if form.is_valid():
            order_cost = form.cleaned_data['order_cost']
            demand_intensity = form.cleaned_data['demand_intensity']
            storage_cost = form.cleaned_data['storage_cost']

            result = calculate_wilson_model(
                order_cost,
                demand_intensity,
                storage_cost
            )

            input_data = (
                f'Витрати на замовлення: {order_cost} грн; '
                f'Інтенсивність використання запасів: {demand_intensity}; '
                f'Витрати на зберігання: {storage_cost} грн'
            )

            result_text = f'Оптимальний розмір замовлення: {result}'

            if request.user.is_authenticated:
                CalculationHistory.objects.create(
                    user=request.user,
                    calculation_type='wilson',
                    input_data=input_data,
                    result=result_text
                )
                is_saved = True

    return render(request, 'wilson.html', {
        'form': form,
        'result': result,
        'is_saved': is_saved
    })
```

```
# Автоматичний підбір транспорту
def transport_select(request):

    form = TransportSelectionForm()
    suitable_transports = None
    is_saved = False

    if request.method == 'POST':
        form = TransportSelectionForm(request.POST)

        if form.is_valid():
            cargo_weight = form.cleaned_data['cargo_weight']
            cargo_volume = form.cleaned_data['cargo_volume']
            fuel_consumption = form.cleaned_data['fuel_consumption']

            suitable_transports = Transport.objects.filter(
                carrying_capacity_min__lte=cargo_weight,
                carrying_capacity_max__gte=cargo_weight,
                volume_min__lte=cargo_volume,
                volume_max__gte=cargo_volume,
                fuel_consumption_min__lte=fuel_consumption,
                fuel_consumption_max__gte=fuel_consumption,
            ).order_by('fuel_consumption_min')

            transport_names = [
                transport.model_name for transport in suitable_transports
            ]

            input_data = (
                f'Вага вантажу: {cargo_weight} кг; '
                f'Об'єм вантажу: {cargo_volume} м³; '
                f'Бажані витрати палива: {fuel_consumption} л/100 км'
            )

            if transport_names:
                result = 'Підібраний транспорт: ' + ', '.join(transport_names)
            else:
                result = (
                    "За введеними параметрами відповідного транспорту "
                    'не знайдено.'
                )

            if request.user.is_authenticated:
                CalculationHistory.objects.create(
                    user=request.user,
                    calculation_type='transport',
                    input_data=input_data,
                    result=result
                )
                is_saved = True

    return render(request, 'transport_select.html', {
        'form': form,
        'suitable_transports': suitable_transports,
        'is_saved': is_saved
    })

# Виведення та фільтрація історії розрахунків
@login_required
def history_list(request):

    calculation_type = request.GET.get('calculation_type')

    calculations = CalculationHistory.objects.filter(
        user=request.user
    ).order_by('-created_at')

    if calculation_type:
        calculations = calculations.filter(
            calculation_type=calculation_type
        )

    return render(request, 'history_list.html', {
        'calculations': calculations,
        'calculation_type': calculation_type,
        'calculation_types': CalculationHistory.CALCULATION_TYPES,
    })
```

ДОДАТОК В