

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування інституту, назва факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

магістра

(ступінь вищої освіти)

на тему ДОСЛІДЖЕННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ МЕТОДІВ
ПІДТРИМКИ РОБОТИ СЕРВІСУ ДОСТАВКИ ЇЖІ
RESEARCH AND SOFTWARE IMPLEMENTATION OF METHODS FOR
SUPPORTING FOOD DELIVERY SERVICE

Виконав: студент(ка) 2 курсу, групи КНТ-211м

Спеціальності 122 Комп'ютерні науки

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Системи штучного інтелекту

Воложанін С.О.

(прізвище та ініціали)

Керівник Степаненко О.О.

(прізвище та ініціали)

Рецензент Голуб Т.В.

(прізвище та ініціали)

2022

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
(повне найменування закладу вищої освіти)

Інститут, факультет ФКНТ
Кафедра програмних засобів
Ступінь вищої освіти магістр
Спеціальність 122 Комп'ютерні науки
(код і найменування)

Освітня програма (спеціалізація) Системи штучного інтелекту
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
С.О. Субботін
“ ” 2022 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

Воложаніна Станіслава Олександровича

(прізвище, ім'я, по батькові)

1. Тема проєкту (роботи) Дослідження та програмна реалізація методів підтримки роботи сервісу доставки їжі
Research and Software Implementation of Methods for Supporting Food Delivery Service

керівник проєкту (роботи) Степаненко Олександр Олексійович, к.т.н., доцент,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу вищої освіти від “4” листопада 2022 року № 363

2. Строк подання студентом проєкту (роботи) грудень 2022 року

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз проблеми та постановка завдань дослідження. 2. Матеріали і методи. 3. Проєктування програмного забезпечення. 4. Основні рішення щодо реалізації компонентів системи. 5. Експлуатація, тестування та експериментальне дослідження програми.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-5 Основна частина	Степаненко О.О., доцент		
Нормоконтролер	Дейнега Л.Ю., ст. викладач		

7. Дата видачі завдання “1” вересня 2022 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи	1 тиждень	Завдання, ТЗ
2	Аналіз та дослідження предметної області	2-5 тижні	Розділи 1, 2
3	Розробка архітектури програми	6 тиждень	Розділ 3
4	Розробка програми	7-9 тижні	Розділ 4
5	Тестування та експериментальне дослідження програми	10-11 тижні	Розділ 5
6	Оформлення пояснювальної записки та документів до неї. Нормоконтроль та рецензування	12 тиждень	Додатки
7	Захист роботи	13 тиждень	

Студент(ка)

(підпис) Воложанін С.О.
(прізвище та ініціали)

Керівник проєкту (роботи)

(підпис) Степаненко О.О.
(прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи магістра:
101 с., 36 рис., 4 табл., 3 дод., 13 джерел.

ДОСТАВКА ЇЖІ, ЗАКЛАД ХАРЧУВАННЯ, КУР'ЄР,
ВЕБЗАСТОСУНОК, ЗАДАЧА КЛАСИФІКАЦІЇ, ДЕРЕВА РІШЕНЬ.

Об'єкт роботи – процес підтримки роботи сервісу доставки їжі.

Предмет роботи – методи підтримки роботи сервісу доставки їжі.

Мета роботи – розробити програмну систему підтримки роботи сервісу доставки їжі для забезпечення задоволеності замовника шляхом керування доставкою замовлення задля досягнення вчасної доставки.

Матеріали, методи та технічні засоби: дерева рішень, алгоритм FSCART, мова Python, пакет scikit-learn, фреймворк створення вебзастосунків Django, система керування базами даних MySQL.

Результати. Наукова новизна роботи виражена в методі керування доставкою їжі, який було запропоновано в роботі і який характеризується виконанням класифікації замовлень за тривалістю доставки за допомогою побудови дерева рішення на основі алгоритму FSCART, що дозволяє обирати такий керуючий вплив на реалізацію доставки замовлення, щоб воно з більшою вірогідністю завершувалося успішно.

Висновки. Реалізовано програмну систему підтримки роботи сервісу доставки їжі як вебзастосунок, що надає інтерфейси для клієнтів, представників мереж закладів харчування, окремих закладів харчування та сервісу доставки їжі загалом. У реалізовану програмну систему імплементовано представлений в роботі метод керування замовленням доставки їжі, який підтримує роботу представників сервісу доставки щодо досягнення задоволеності замовника.

Галузь використання – сервіси з доставки їжі з закладів харчування.

ABSTRACT

Explanatory note to the diploma qualifying work of the master: 101 pages, 36 figures, 4 tables, 3 appendixes, 13 sources.

FOOD DELIVERY, FOOD ESTABLISHMENT, COURIER, WEB APPLICATION, CLASSIFICATION PROBLEM, DECISION TREE.

The object of the work is the process of supporting food delivery service.

The subject of the work is methods of supporting food delivery service.

The purpose of the work is to develop a software system to support product delivery service to ensure customer satisfaction by managing order delivering to achieve on-time delivery.

Materials, methods and technical means: decision tree, FSCART algorithm, Python language, scikit-learn package, Django web application creation framework, MySQL database management system.

The results. The scientific novelty of the work is based on the method of managing food delivery, which was proposed in the work and which is characterized by performing the classification of orders by the duration of delivery using the construction of a decision tree based on the FSCART algorithm, which allows to choose such a controlling influence on the implementation of an order delivery that it is more likely completed successfully.

Conclusions. A software system for supporting the food delivery service as a web application has been implemented. It provides interfaces for customers, representatives of food establishment network, individual food establishments and food delivery service. The developed software system implements the method of food delivery order management presented in the work, which supports the work of representatives of the delivery service in order to achieve customer satisfaction.

The field of use is food delivery services from food establishments.

ЗМІСТ

С.

Перелік скорочень та умовних познач.....	7
Вступ.....	8
1 Аналіз проблеми та постановка завдань дослідження	10
1.1 Основні поняття і актуальність сервісів доставки їжі.....	10
1.2 Програмні аналоги сервісу доставки їжі	16
1.3 Висновки за розділом 1	21
2 Матеріали і методи.....	22
2.1 Задача керування доставкою їжі.....	22
2.2 Дерева рішень	24
2.3 Метод керування доставкою їжі	26
2.4 Висновки за розділом 2	28
3 Проєктування програмного забезпечення	29
3.1 Вибір засобів розробки	29
3.2 Моделювання роботи з програмою	30
3.3 Проєктування бази даних	35
3.4 Висновки за розділом 3	44
4 Основні рішення щодо реалізації компонентів системи.....	46
4.1 Структура програмної системи.....	46
4.2 Опис розроблених класів.....	47
4.3 Висновки за розділом 4	51
5 Експлуатація, тестування та експериментальне дослідження програми	52
5.1 Опис роботи з програмою	52
5.2 Експериментальне дослідження	57
5.3 Висновки за розділом 5	63
Висновки	64
Перелік джерел посилення	65
Додаток А Технічне завдання	67
Додаток Б Текст програми	72
Додаток В Слайди презентації.....	93

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

CART – Classification And Regression Tree;

FSCART – Feature Selection Classification And Regression Tree;

GPS – Global Positioning System;

ID3 – Iterative Dichotomiser 3.

ВСТУП

На даний момент однією зі сфер, що найдинамічніше розвивається у всьому світі, є доставка їжі, що реалізується відповідними сервісами. Загалом сам спосіб харчування у світі кардинально змінюється: трохи менше двох десятиліть тому доставка їжі ресторанної якості все ще здебільшого обмежувалася такими продуктами, як піца та китайська кухня, нині доставка їжі стала глобальним ринком, вартість якого перевищує 150 мільярдів доларів США, а ця сума зросла більш ніж утричі з 2017 року, так у Сполучених Штатах ринок зріс більш ніж удвічі під час пандемії COVID-19 після історичного зростання до того на 8 % [1].

Звичайно, що такі великі абсолютні цифри та темпи збільшення обсягів надання послуг додатково привертають увагу до самої концепції, яка вкладається в загальну логіку розподіленої економіки, коли послуги надаються для клієнтів, що входять у більш широке ніж зазвичай коло. При цьому у випадку доставки їжі не можна говорити про продаж їжі між країнами (якщо це не розташовані дуже близько один одного міста на кордоні країн, які входять до єдиного простору, як наприклад, до Європейського Союзу), але доставка їжі може надати можливість користуватися послугами закладів у всьому місті загалом, а також можливо у містах, містечках, селах, які розташовані на певній, невеликій відстані від міста або місця, де розташований заклад, в якому було виготовлено їжу. Сама ця концепція загалом впливає на зміну концепції споживання (харчування), адже за рахунок доступності послуги кожна людина отримує можливість замість приготування їжі власноруч, походу до відповідного закладу, скористатися послугами таких сервісів у той момент, коли не має часу на приготування їжі та відвідування закладу. Відповідно це сприяє загальній концепції життя сучасної людини, а значить сприяє розвитку галузі.

Сама по собі дана галузь розвивається не тільки за рахунок описаних вище ідей, але і за рахунок простоти і зручності замовлення таких послуг.

Для забезпечення підтримки послуг використовується програмне забезпечення, яке повинно надавати можливість швидко і зручно замовити потрібну їжу, визначити параметри замовлення (для клієнта), а для менеджерів сервісу – відповідно керувати цими замовленнями таким чином, щоб забезпечити потреби клієнтів. Проте варто відзначити, що також на загальне задоволення послугами впливає те, на скільки швидко і якісно доставку було виконано. І у цьому відношенні для програмного забезпечення важливо забезпечити підтримку визначення того інтервалу часу, за який вдасться доставити їжу.

У підсумку метою роботи є розробка програмного забезпечення підтримки роботи сервісу доставки їжі для забезпечення керування замовленнями на основі класифікації інтервалів часу доставки замовлень їжі з закладів харчування.

1 АНАЛІЗ ПРОБЛЕМИ ТА ПОСТАНОВКА ЗАВДАНЬ ДОСЛІДЖЕННЯ

1.1 Основні поняття і актуальність сервісів доставки їжі

Сервіси доставки їжі передбачають найчастіше доставку їжі з закладів харчування (ресторанів) клієнтам додому, на роботу або загалом за потрібною адресою без необхідності самотійного відвідування даного закладу харчування.

Однак існують також, наприклад, так звані «кухні-привиди», які надають можливість замовлення на доставку продукції, але не мають фізичного ресторану чи вітрини, таким чином вони дозволяють ресторанам розширюватися та експериментувати з мінімальними інвестиційними ризиками [1].

Відповідно до останнього звіту Emergen Research, очікується, що обсяг світового ринку послуг доставки їжі досягне 311,43 мільярдів доларів США у 2028 році за незмінного середньорічного зростання доходу в 11,4 %, а сприяти всьому цьому за прогнозом мають:

- збільшення використання смартфонів;
- збільшення кількості домогосподарств з подвійним доходом;
- збільшення використання Інтернету;
- зростання бажання швидкого доступу до доступної їжі [2].

Важливим є також порівняння темпів зростання різних галузей. До пандемії ресторанна індустрія в Сполучених Штатах Америки зростала на 3-4 % на рік, а в свою чергу продажі їжі з доставкою зростали приблизно вдвічі вищими темпами (7-8 %) [1].

Збільшення кількості користувачів смартфонів дало поштовх онлайн-сервісам доставки їжі по всьому світу: користувачі смартфонів є основними онлайн-покупцями для харчової та безалкогольної індустрії, і збільшення кількості користувачів смартфонів відображає потенційне збільшення онлайн-покупок їжі та напоїв, у 2019 році кількість користувачів електронної комерції у світі досягла 1,5 мільярди, і очікується, що до 2024 року вона зросте на 800 мільйонів, у середньому на 25 % порівняно з аналогічним

періодом минулого року, так індійська платформа доставки їжі Zomato має 80 мільйонів активних користувачів щомісяця та поставила ціль досягти додаткових 20 мільйонів протягом наступних кількох років, отже збільшення кількості користувачів смартфонів і проникнення Інтернету стимулюють зростання онлайн-служб доставки їжі [3].

Організація сервісу доставки їжі визначається не тільки створенням меню, а потребує вирішення одразу цілого набору питань, кожне з яких визначає спосіб роботи з відповідними сутностями предметної області, процесами, а також безпосередньо формує ці сутності і процеси.

До питань, які потрібно розв'язати під час визначення сервісу доставки їжі, відносяться:

- вибір аудиторії: незважаючи на більше охоплення клієнтської бази порівняно з традиційними ресторанами, важливо враховувати потреби цільової аудиторії, щоб забезпечити постійний приплив замовлень, для цього потрібно визначити, які клієнти відвідують ресторан, яку вартість вони готові платити і які їхні характеристики, в результаті мають бути визначені найпоширеніші демографічні групи, окрім того потрібно враховувати доступ до самої доставки, відповідно визначивши місцезнаходження і доступність самої доставки (це студентський район або місто, передмістя) та відповідні характеристики конкурентів (краще мати аудиторію, відмінну від конкурентів);

- вибір постачальників послуг доставки: наявність власної служби доставки може дати більше контролю над усім процесом, проте для багатьох ресторанів це може бути занадто великою інвестицією, з іншого боку, доставка третьою стороною дає змогу ресторанам скористатися випробуваним підходом, який включає величезний досвід і ресурси, в обмін на це треті сторони стягують щомісячну комісію, при цьому якщо обирається власна служба доставки, доведеться визначити свою бізнес-модель, найняти додатковий персонал доставки, інвестувати в парк доставки та розробити хороший додаток для замовлення;

– вибір технології доставки для максимальної ефективності: програмне забезпечення для керування доставкою може автоматизувати потік замовлень на доставку, підвищуючи ефективність і допомагаючи зосередитися на основному бізнесі, автоматизація доставки має вирішальне значення для швидкого та ефективного розширення бізнесу, ефективно використовуючи час і ресурси;

– визначення меню: одним із наріжних каменів успішного бізнесу доставки є меню доставки, потрібно ретельно вибирати страви, які продаватимуться онлайн, аналізуючи транспортабельність кожної страви, розмір прибутку, потреби в упаковці, час приготування тощо, при цьому спочатку краще за все провести випробування, перевіривши, як виглядає їжа після транспортування, після того, як обрано приблизно 15-20 страв, можна почати створювати привабливу онлайн-пропозицію, включаючи чіткі описи та якісні зображення, додаючи модифікатори продукту, які дають змогу персоналізувати страви, наприклад, макарони без глютену, без бекону, додатковий сир;

– підготовка логістики: додавання доставки до параметрів замовлення ресторану створить додаткове навантаження, особливо на кухні, тому дуже важливо передбачити та нарощувати потужність, якщо це необхідно, кухня часто є вузьким місцем доставки, тому потрібно спочатку переконатися, що вона завжди може підтримувати обробку великих обсягів, іншою важливою частиною доставки є упаковка, бо відсутність належної упаковки їжі призведе до неправильної температури, зіпсованих смаків, поганих відгуків і незадоволених клієнтів, тому не можна економити на упаковці та використовувати ізольовані пакети, екологічно чисті та фірмові контейнери та матеріали, що підлягають переробці, до того ж потрібно кур'єрам спростити доступ до кухні, забрати замовлення та доставити їх клієнтам якомога швидше та простіше, тому потрібно призначити одного зі своїх співробітників спеціалістом із доставки та контактною особою для водіїв у разі виникнення незвичайних ситуацій, як-от затримки, проблеми з погодою, неочікувано великих обсягів;

– реклама служби доставки: коли сервіс почав працювати, потрібно не забути належним чином продати своє нове підприємство, розміщуючи рекламу в соціальних мережах, місцевих газетах і у ресторані або роздаючи рекламні листівки, важливо пропонувати спеціальні пропозиції, щоб відсвяткувати відкриття в Інтернеті, особливо для лояльної клієнтської бази, подумати про акції та знижки, які клієнти можуть використати для своєї першої покупки, і винагородити їх за рекомендацію другу [4].

Клієнти в сучасних умовах підвищення активності доставки, зокрема їжі, очікують загалом більше від служб доставки їжі, віддаючи пріоритет таким характеристикам:

- швидкість доставки, при чому ціль у менше ніж 30 хвилин є роздільником між існуючими платформами, визначаючи те, на скільки вони задовольняють сучасним умовам;
- якість їжі з очікуванням ресторанної якості їжі навіть після доставки;
- 100-відсоткова точність і виконуваність як для звичайних товарів, так і для спеціальних запитів;
- різноманітність кухонь [1].

Проте тільки відкриття і доступності сервісу доставки їжі в сучасному світі звичайно недостатньо, при цьому поява сервісів, які враховують тренди, призводить до того, що вони займають важливу нішу в підсумку на ринку.

Основними трендами, які визначають розвиток галузі в найближчому майбутньому, є:

- міленіали як ключові драйвери розвитку: міленіали є пріоритетною аудиторією сервісів доставки їжі, оскільки вони витрачають найбільшу частку свого бюджету на готову їжу порівняно з іншими поколіннями, так зі стрімким темпом життя це покоління більше схильне їсти в ресторанах швидкого обслуговування та замовляти їжу з доставкою (у звіті Technomic Generation зазначено, що в 3 із 5 випадків міленіали замовляють доставку їжі або відвідують ресторани, беручи їжу на винос), однак потрібно враховувати, що з більшою платіжною спроможністю міленіали висувають вищі вимоги

(легке замовлення, миттєва доставка, оплата за один крок тощо), тому щоб задовольнити вимогливих міленіалів, компанії з доставки їжі винайшли нові канали замовлення та варіанти доставки;

- нові канали доставки: 63 % американців відмовляються від електронних замовлень на винос, якщо у них поганий клієнтський досвід, а 55 % дорослих американців потребують простішої системи онлайн-замовлень, тому враховуючи це, платформи доставки їжі мають значно спростити замовлення та дозволити клієнтам робити замовлення через зручні для них канали;

- сучасні варіанти доставки: компанії з доставки їжі більше не обмежені лише робочою силою, вони можуть значно пришвидшити доставку та знизити експлуатаційні витрати за допомогою роботів, дронів і навіть парашутів;

- охоплення нових ніш, адже багато служб доставки їжі зосередилися на особливій ніші: існують сервіси доставки їжі, які подобаються веганам і компаніям, які доставляють суто коробки для гамбургерів або вирощену на фермі їжу чи здорові смузі, перебування в ніші може допомогти стартапам з доставки їжі подолати бар'єри входу та боротися з гострою конкуренцією, тому власник бізнесу з доставки їжі повинен стежити за новими нішами, які набирають обертів, зокрема боротьбою з харчовими відходами, кормом для домашніх тварин і віртуальними кухнями;

- замовлення їжі за допомогою криптовалюти: розвиток технології блокчейн породив нову нішу послуг – криптографічні замовлення їжі, клієнти по всьому світу можуть замовити будь-який вид виносу або доставки за допомогою криптовалют, використовуючи криптовалюту для замовлень, компанії можуть усунути шахрайські дії та покращити загальний досвід замовлення їжі, клієнти не обмежені певним варіантом оплати та можуть оплачувати свої замовлення навіть відносно невідомими криптовалютами;

- великі дані допомагають революціонізувати служби доставки їжі та зробити їх максимально ефективними, так постачальники продуктів харчування збирають і аналізують величезний обсяг даних: Grubhub,

наприклад, аналізує дані про мільйони замовлень, щоб розробити найбільш відповідні рекомендації щодо їжі для кожного клієнта, Doordash використовує великі дані для керування запасами та прогнозування кількості кур'єрів, необхідних на певний день, очікуваного попиту на конкретну кухню та часу приготування їжі, Just Eat вивчає сотні тисяч записів клієнтів, щоб зрозуміти, чого клієнти хочуть від їхніх послуг: яку їжу вони замовлятимуть, коли та як часто, а в підсумку можна відзначити, що саме великі дані сприяли зростанню в 50 % компанії на ринку Великобританії, що звичайно є дуже значним показником [5].

Загалом відносно роботи з великими даними можна вказати, що сервіси доставки їжі на даний момент збирають дані про:

- дорожній рух;
- вплив температури на їжу;
- вплив ринкових тенденцій на споживання запасів;
- історію покупок клієнтів;
- складові поточних замовлень;
- відгуки та коментарі в соціальних мережах, що в результаті допомагає компаніям з доставки їжі точно розраховувати час доставки, стежити за емоціями клієнтів у соціальних мережах, миттєво реагувати на скарги, персоналізувати харчові пропозиції тощо [5].

Згідно зі звітом Cargemini, роздрібні торговці можуть втратити до 26 % свого прибутку, якщо вони не зможуть оновити свою логістичну систему, щоб забезпечити своєчасну доставку, незважаючи на розширену онлайн-систему продуктових магазинів, таким чином, витрати, які несе постачальник, можуть стримувати зростання ринку послуг доставки їжі онлайн, але важливим результуючим показником для замовників щодо оцінки роботи сервісів доставки їжі є забезпечення виконання вчасної доставки [3].

1.2 Програмні аналоги сервісу доставки їжі

Найвідомішим в Україні сервісом доставки їжі є Glovo (рис. 1.1), що був заснований в Іспанії, а на даний момент веде діяльність у багатьох країнах світу [6].

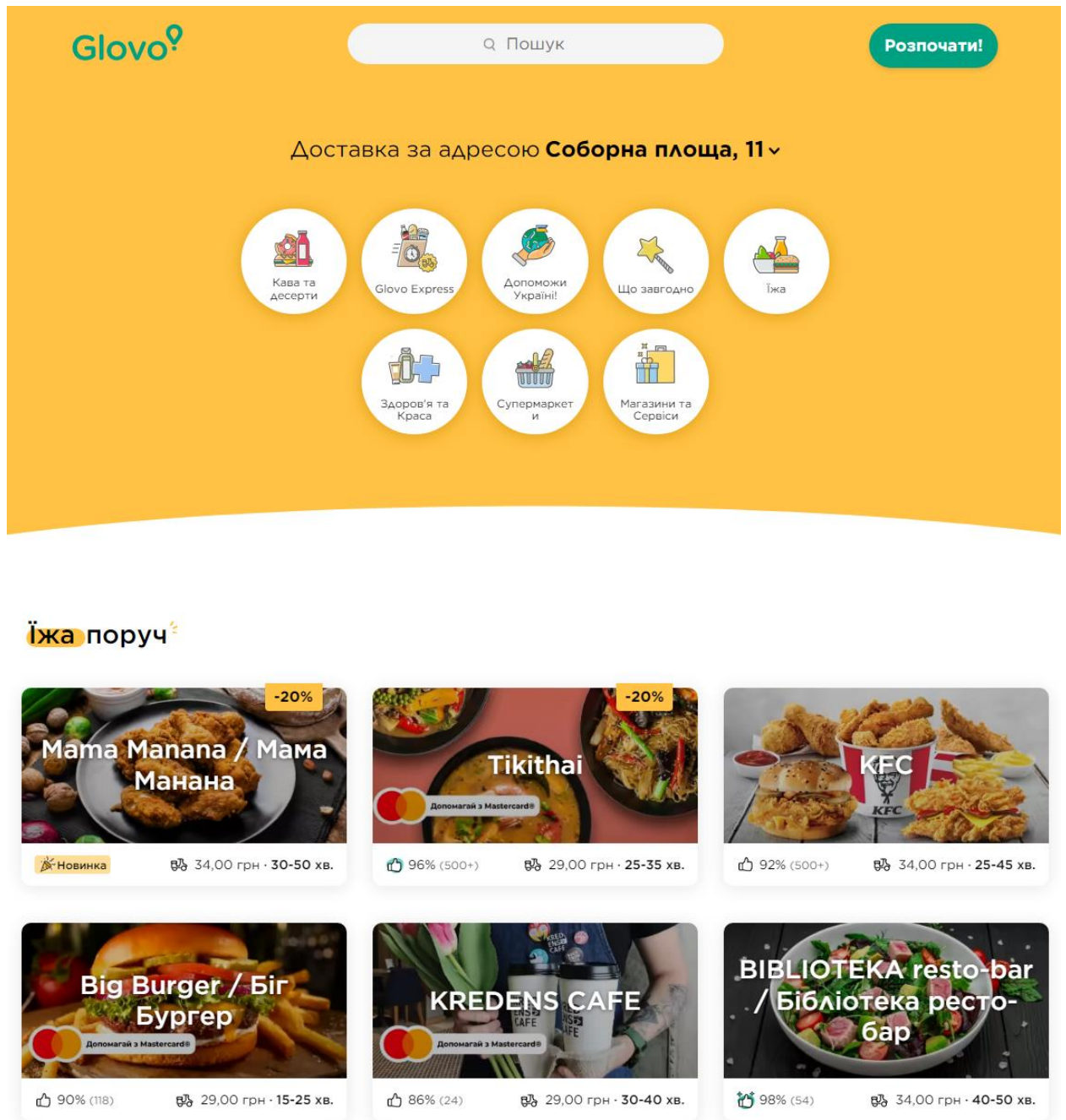


Рисунок 1.1 – Інтерфейс вебсайту сервісу доставки їжі Glovo

Glovo дозволяє працювати з сервісом, використовуючи вебсайт, де доступна вся основна функціональність, та мобільні застосунки для операційних систем Android, iOS. Тобто охоплення аудиторії достатньо широке, при цьому будь-який новий користувач може достатньо швидко і зручно знайти потрібну йому інформацію, а далі виконати замовлення через вебсайт без необхідності обов'язкового встановлення окремого застосунку. Авторизація вимагається саме під час замовлення доставки, хоча може бути виконана за допомогою кнопки «Розпочати» в будь-який необхідний для цього момент.

Робота з вебсайтом одразу класифікована за містами, де відбувається доставка, відповідно початкова сторінка формується у відповідності з початковим заданим містом. Тоді з такої сторінки можна виконати пошук закладів харчування, супермаркетів, засобів здоров'я та краси, магазинів або всього одразу. Тобто фактично початковий вибір реалізується для заданого міста щодо вибору конкретного закладу. Вебсторінка, яка відкривається в результаті, відображається як результат пошуку, на якій можна обрати фільтр, що відповідає категорії їжі (конкретна кухня або категорія їжі), визначити варіант сортування (за віддаленістю, за рейтингом або за комісією за доставку). У будь-якому випадку після цього перехід відбувається фактично на сторінку конкретного закладу. Ця сторінка представляє собою меню закладу, де потрібно спочатку обрати категорію страв, відкриваючи перелік страв за категорією. З останньої вебсторінки можна додати конкретну страву до замовлення. Під час додавання конкретної страви можна задати її кількість та обрати потрібні добавки до страви у випадку, якщо добавки доступні. У підсумку можна сформулювати замовлення.

З вебсайту також можна подати заявку на те, щоб долучитися до роботи сервісу, включаючи заявку працювати кур'єром, заявку на партнерство, переглянути доступні вакансії.

В якості аналогу, який є широковідомим американським сервісом, розглянуто DoorDash (рис. 1.2) [7].

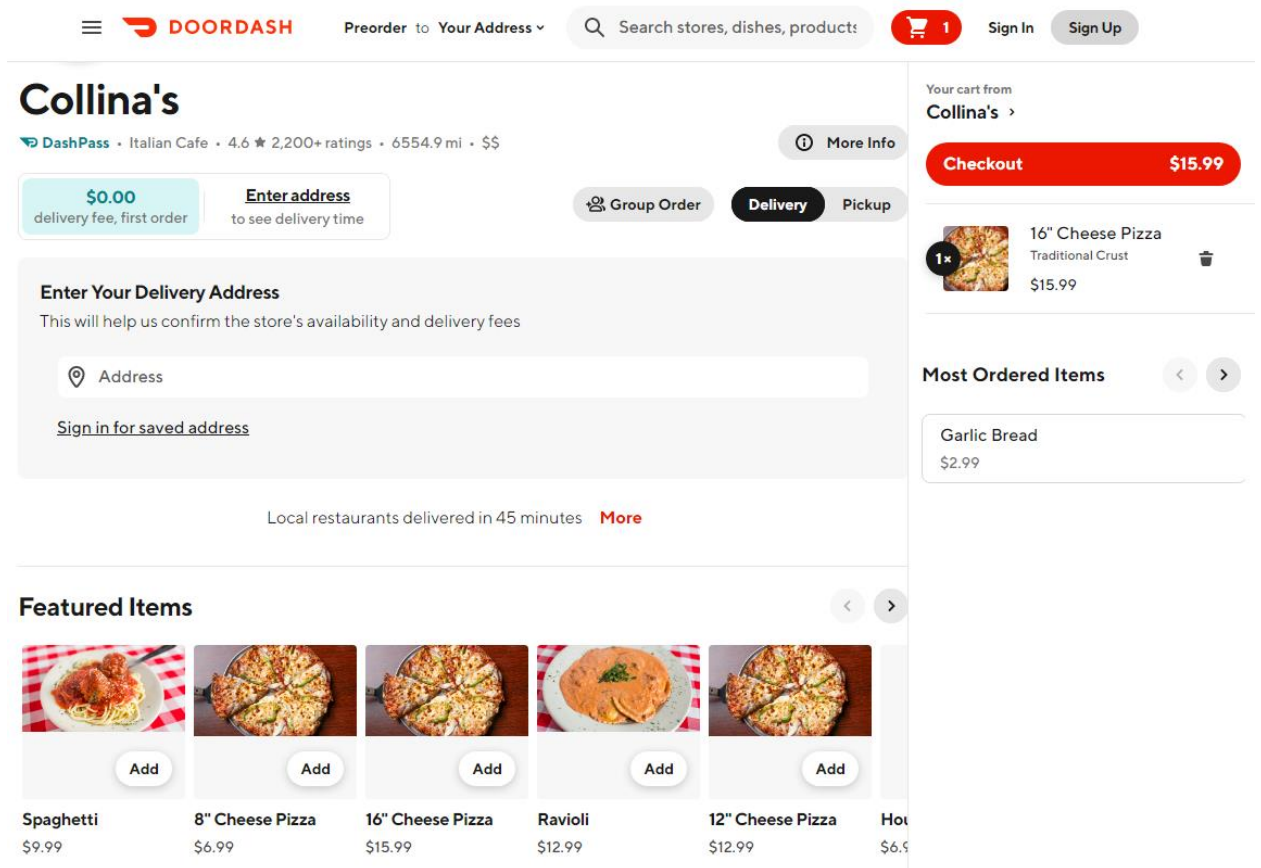


Рисунок 1.2 – Інтерфейс вебсайту сервісу доставки їжі DoorDash

DoorDash також має саме вебсайт, а також надає мобільні застосунки для операційних систем Android, iOS. Слід вказати, що загалом в даній сфері найпопулярніші сервіси мають саме вебсайт в якості універсального інструменту доступу, а вже до нього надається вебзастосунок.

DoorDash не має початкового розподілу за містом, а виконує одразу доступ через пошук, тобто саме пошук є початковою точкою входу для роботи з вебсайтом. З відповідної сторінки пошук відбувається за замовчуванням у Нью-Йорку, але можна задати необхідне місто і адресу, фактично замінивши значення за замовчуванням у параметрах пошуку. Серед параметрів пошуку можна також обрати категорію їжі (бургери, випічка, салати, здорова їжа, сніданки, кава тощо), визначити можливість забрати замовлення самостійно, визначити, що доставка має відбутися за строк до 30 хвилин, обмежити вартість. Окрім того можна задати дату і час бажаної доставки, тобто замовлення можна оформлювати наперед.

У результатах пошуку також перелічуються саме заклади харчування, після чого потрібно відкрити вебсторінку з відповідним закладом, де доступна інформація про тип кухні в закладі, виставлену клієнтами оцінку, кількість таких оцінок, час роботи закладу, середній час доставки, адресу, номер телефону, меню закладу за категоріями з можливістю пошуку серед страв за всім меню, а також перелік відгуків клієнтів. Кожну страву можна додати до замовлення. За кожною стравою надається її назва, опис, вартість, зображення, відсоток позитивних відгуків та кількість таких відгуків. Також можна додати до страви за необхідності спеціальні інструкції та вибрати варіант реагування на відсутність цієї страви для замовлення. При пошуку можна обрати варіант сніданку або повного меню (для варіантів, коли страва недоступна для замовлення в певний період).

Під час оформлення замовлення можна авторизуватися, визначити деталі щодо доставки та внести інформацію для здійснення платежу.

Проаналізовані і описані вище можливості стосуються роботи клієнтів, однак доступний також інтерфейс для роботи самих торговців, які представляють заклади харчування, що надає можливість роботи з наступними функціями:

- відслідковування замовлень і керування ними: для керування можна прийняти замовлення, переглянути перелік замовлень, для яких прострочено час доставки через кур'єра або через клієнта, отримати зворотній зв'язок від кур'єра;

- аналізування результатів роботи бізнесу, під час якого надається можливість переглянути популярні страви, отримані кошти за заданий період, рейтинги закладів, узагальнену інформацію про продажі, відгуки клієнтів (широкий інструментар відображення потрібних відгуків, включаючи виставлені оцінки, останні відгуки тощо);

- керування майбутніми замовленнями, визначаючи паузи в роботі закладу або завершуючи роботу закладу дочасно.

Для даного інтерфейсу надається рівень доступу адміністратора, менеджера та відповідно кур'єра.

За результатами проведеного аналізу визначено ідею програми, яка розробляється (табл. 1.1).

Таблиця 1.1 – Таблиця результатів аналізу програмних аналогів

Характеристика	Glovo	DoorDash	Програма-розробка
Пошук за ресторанами	+	+	+
Вибір ресторану за категорією страви	+	+	+
Текстові відгуки клієнтів	—	+	+
Оцінки ресторанів клієнтами	+	+	+
Оцінки страв клієнтами	—	+	+
Визначення спеціальних інструкцій	—	+	+
Визначення варіанта скасування клієнтом	—	+	+
Керування замовленнями службою доставки	+	+	+
Керування замовленнями і інформацією зі сторони закладу харчування	+	+	+
Класифікація часу майбутньої доставки	—	—	+

Програма-розробка має ґрунтуватись на забезпеченні функціональності для клієнта, для закладів харчування та зокрема забезпечення класифікації часу майбутньої доставки, щоб можна було мати фактично прогнозований час доставки, щоб завчасно орієнтувати клієнта і

представників сервісу, закладу харчування на те, як керувати замовленням найкраще. Основним аналогом, який став прототипом розробки, є DoorDash, який має широку функціональність.

1.3 Висновки за розділом 1

У відповідності з проблемою, що розглядається, за результатами виконання магістерської роботи необхідно забезпечити виконання наступних завдань:

- проаналізувати принципи роботи сервісів доставки їжі та програмне забезпечення, що забезпечує підтримку їх роботи;
- дослідити методи підтримки роботи сервісу доставки їжі;
- виконати проєктування програмної системи підтримки роботи сервісу доставки їжі;
- реалізувати програмну систему підтримки роботи сервісу доставки їжі, що повинна надавати інтерфейси для клієнтів, представників мереж закладів харчування, окремих закладів харчування та сервісу доставки їжі загалом.

У першому розділі саме перше завдання було розв’язано. Для цього спочатку було вивчено основні поняття, напрямки розвитку, питання популярності сервісів доставки їжі, а потім виконано огляд і аналіз програмних аналогів, серед яких для більш детального розгляду було виділено Glovo і DoorDash, останній яких визначений в якості основного прототипу програмної системи, що розробляється.

2 МАТЕРІАЛИ І МЕТОДИ

За результатами, представленими в розділі 1, можна стверджувати, що для сервісів доставки їжі критичною є доставка за 30 хвилин замовлення клієнту. У такому випадку можна говорити про успішне виконання замовлення і задоволеність клієнта. Тому важливим стає визначення фактичного інтервалу часу доставки замовлення одразу під час роботи з ним, тобто фактично класифікація такого замовлення. Це є важливим, адже у випадку, якщо буде визначено, що замовлення не вдасться доставити вчасно, то у менеджера залишається можливість відреагувати на це і все-таки виконати доставку вчасно, використовуючи можливо додаткові ресурси або більш ефективно використовуючи наявні ресурси для реалізації доставки порівняно з першим прийнятим рішенням.

При цьому варто враховувати, що фактично результуючий показник витраченого часу на доставку, а значить і клас тривалості доставки, визначається за достатньо складною залежністю, оскільки на нього впливає багато факторів.

У даному розділі необхідно виконати дослідження даної проблеми, визначивши задачу керування доставкою їжі, дослідивши методи, які можуть бути застосовані для її розв'язання і в підсумку визначивши конкретний метод, який буде використовуватися в роботі загалом і в програмному забезпеченні зокрема для розв'язання цієї проблеми.

2.1 Задача керування доставкою їжі

Нехай відомі параметри замовлення доставки їжі, що входять до множини X , що складається з елементів x_i , кожен з яких є значенням, яке характеризує замовлення, при цьому $i = \overline{1, K}$, де K – кількість таких параметрів.

У свою чергу множина Y складається з елементів y_i , кожен з яких представляє певний керуючий вплив на замовлення. Кожен елемент y_i може мати значення з деякого набору V_i . Кожен елемент набору V_i має значення v_i^q , що визначає відповідний конкретний вплив на замовлення, тобто фактично варіант обслуговування даного замовлення. Усього передбачається наявність Q варіантів для такого впливу, відповідно $q = \overline{1, Q}$.

Відомо, що вся сукупність значень параметрів замовлення X та відповідних керуючих впливів Y призводить до доставки замовлення за деякий час. Цей час належить одному з класів тривалості доставки замовлення з множини Z , де кожен відповідний клас z_i , визначає конкретний інтервал часу, за який вдалось доставити замовлення.

Задача керування доставкою їжі полягає у визначенні таких значень керуючого впливу Y , які разом зі значеннями параметрів замовлення X призводять до отримання потрібного класу тривалості доставки замовлення z_0 (успішної доставки). При цьому необхідно встановити відповідну функціональну залежність f між класом тривалості доставки замовлення та параметрами замовлення і керуючим впливом, яка б дозволила визначати клас тривалості доставки для відповідного замовлення на основі залежності (2.1):

$$f(X, Y) = Z. \quad (2.1)$$

Значення даної залежності f відомі для деякої вибірки T , що називається тренувальною, і визначається виразом (2.2):

$$T = (X^r, Y^r, Z^r), \quad (2.2)$$

де r – порядковий номер замовлення в тренувальній вибірці T , $r = \overline{1, R}$;

R – кількість замовлень, дані про які представлені в тренувальній вибірці;

X^r – множина значень, що характеризують параметри замовлення з номером r ;

Y^r – множина значень, що характеризують керуючий вплив, заподіяний на замовлення з номером r ;

Z^r – клас тривалості доставки замовлення з номером r .

2.2 Древа рішень

Алгоритм дерев рішень належить до сімейства алгоритмів керованого навчання (навчання з учителем), при цьому даний алгоритм на відміну від інших алгоритмів навчання з учителем можна використовувати як для вирішення проблем регресії, так і проблем класифікації, метою використання дерев рішень є створення навчальної моделі, яка може використовуватися для прогнозування класу або значення цільової змінної шляхом вивчення простих правил прийняття рішень, виведених із попередніх даних (навчальних даних): прогнозування мітки класу для запису починається з кореня дерева, при цьому порівнюється значення кореневого атрибута з атрибутом запису та на основі результатів порівняння слідують за гілкою, що відповідає цьому значенню, і переходять до наступного вузла [8].

Враховуючи, що основна задача у складі задачі керування доставкою їжі полягає в класифікації, то було виділено наступні дерева рішень, які дозволяють вирішити дану задачу:

- Iterative Dichotomiser 3 (ID3);
- Classification And Regression Tree (CART);
- Feature Selection CART (FSCART).

Виконуючи порівняння даних алгоритмів (табл. 2.1), було з'ясовано, що всі вони дозволяють вирішити задачу класифікації даних, проте ID3 не дозволяє працювати з набором ознак, а у випадку даної задачі клас тривалості доставки залежить не тільки від однієї ознаки, до того ж не можна бути точно одразу впевненим у тому, яка потенційна ознака, який саме вплив

має, а тому наявність можливості вибору ознак з множини у FSCART є позитивною додатковою характеристикою, що визначає подальший вибір саме даного алгоритму роботи з деревами рішень для реалізації методу керування доставкою їжі.

Таблиця 2.1 – Порівняння алгоритмів дерев рішень

Критерій	ID3	CART	FSCART
Класифікація даних	+	+	+
Множина ознак	–	+	+
Вибір ознак	–	–	+

Індекс Джині є показником, який використовується для завдань класифікації в CART, по суті визначаючи те, яким чином в результаті відбувається розподіл класів за певним критерієм:

– зберігає суму квадратів ймовірностей для кожного класу, обчислюючи ступінь ймовірності конкретної змінної, яка неправильно класифікується, коли її вибирають випадковим чином, і зміну при цьому індексу Джині, працює з категоріальними змінними, забезпечує результати класифікації як «успішно» або «невдало» і, отже, проводить лише двійкове розбиття;

– ступінь індексу Джині змінюється від 0 до 1, де 0 означає, що всі елементи пов’язані з певним класом або існує лише один клас, відповідно значення 1 означає, що всі елементи випадковим чином розподілені між різними класами, і значення 0,5 означає, що всі елементи рівномірно розподілені в деяких класах [8]-[10].

Математично можна представити індекс Джині на основі виразу (2.3):

$$Gini = 1 - \sum_{i=1}^n (p_i)^2, \quad (2.3)$$

де p_i – ймовірність того, що об'єкт буде віднесений до відповідного i -го класу.

Алгоритм вибору ознак, який використовує CART, або FSCART, реалізується для відкидання тих ознак, які можуть бути неважливими для прийняття рішень у підсумку, наступним чином:

- спочатку алгоритм CART визначає коефіцієнти всіх ознак, тоді неоднорідність дорівнює величині I_0 ;
- відкидається перша ознака x_1 , встановлюючи $w_1 = 0$, і обчислюється нова неоднорідність I_1 , тоді збільшення (зміну) неоднорідності можна представити як $I_1 - I_0$;
- обчислюється збільшення (зміна) неоднорідностей для інших ознак як $I_2 - I_0, I_3 - I_0$ для наступних ознак;
- якщо видалення однієї ознаки значно збільшує неоднорідність, це означає, що ця ознака є важливою, тому отримані значення збільшення неоднорідності сортуються та виявляються найбільш і найменш важливі ознаки;
- обчислюється різниця D між збільшенням неоднорідності для найменш важливої ознаки та найбільш важливої ознаки, а тоді визначається, чи ця різниця менше ніж деяка константа c (зазвичай 0,1 або 0,2), і якщо різниця менша, то вважається, що це непотрібна ознака, і вона відкидається;
- заново алгоритм CART обчислює нові коефіцієнти для всіх ознак, але цього разу без вилученої ознаки, тобто алгоритм починає свою роботу заново і повторює її до того моменту, поки обчислена на попередньому кроці різниця D перестане задовольняти умові вилучення ознаки [11].

2.3 Метод керування доставкою їжі

Метод керування доставкою їжі дозволяє розв'язати задачу, представлену в розділі 2.1, на основі послідовності наступних етапів.

На першому етапі має бути визначено набір даних T , що охоплює всі виконані замовлення з доставки їжі, для кожного з яких відомі значення з набору параметрів X , завданого керуючого впливу Y та результуючий клас тривалості доставки Z . При цьому будь-які текстові характеристики мають бути під час визначення вибірки перетворені до числових представлень.

На другому етапі має бути побудовано дерево рішень шляхом навчання на основі вибірки T за допомогою використання алгоритму FSCART. Підсумком виконання даного етапу є не тільки побудоване дерево, що буде використовуватися для класифікації замовлень, але і узгоджені множини параметрів $X' \subset X$ та керуючих впливів $Y' \subset Y$, що включають тільки ті параметри і керуючі впливи, які завдають значний вплив на результуючий клас тривалості доставки Z .

На третьому етапі визначається порядок застосування керуючих впливів: встановлюється пріоритет v_i^q для кожного значення керуючого впливу V_i , що допустиме для будь-якого керуючого впливу $y'_i \in Y'$.

На четвертому, п'ятому та шостому етапах відбувається циклічний процес керування замовленнями.

На четвертому етапі для кожного нового замовлення визначаються і перетворюються значення його параметрів X' .

На п'ятому етапі на основі створеного на другому етапі дерева рішень визначається для всіх допустимих комбінацій керуючих впливів, що входять до підсумкової множини Y'_A , та зафіксованих значень параметрів X' значення відповідних класів тривалості доставки замовлення. При цьому множина Y'_A складається з тих же керуючих впливів $y'_i \in Y'$, але кожен такий керуючий вплив може приймати меншу кількість значень $v_i^p, p = \overline{1, P}, P < Q$.

На шостому етапі відбираються для замовлення тільки ті комбінації керуючих впливів, які призвели до отримання класу тривалості доставки, який відповідає успішній доставці. Після цього всі комбінації впорядковуються за правилами, які були встановлені на третьому етапі.

Таким чином впорядковані прийнятні рішення надаються менеджеру як результат роботи методу.

З наданих варіантів менеджер має обрати найбільш прийнятне рішення і таким чином визначити обслуговування даного замовлення.

2.4 Висновки за розділом 2

У другому розділі виконано дослідження методів підтримки роботи сервісу доставки їжі, для цього було виконано постановку задачі керування доставкою їжі, порівняно існуючі алгоритми навчання дерев рішень, які можуть використовуватися для вирішення даної задачі, та розв'язано дану задачі за допомогою запропонованого методу керування доставкою їжі. Наукова новизна роботи виражена в цьому методі, який характеризується виконанням класифікації замовлень за тривалістю доставки за допомогою побудови дерева рішення на основі алгоритму FSCART, що дозволяє обирати такий керуючий вплив на реалізацію доставки замовлення, щоб воно з більшою вірогідністю завершувалося успішно.

3 ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Вибір засобів розробки

Проаналізувавши існуючі програмні засоби, які забезпечують підтримку сервісів доставки їжі, було вирішено в межах даної роботи розробити вебзастосунок як універсальне рішення для підтримки роботи з програмною системою. Оскільки враховуючи задачі, сучасний стан веброзробки, розробка вебзастосунків відбувається на основі використання вебфреймворків. Тому вибір мови програмування повинен відбуватися спільно з вибором вебфреймворків. Одними з найбільш активно використовуваних вебфреймворків на даний момент є Laravel і Django (табл. 3.1).

Таблиця 3.1 – Порівняння вебфреймворків Laravel і Django

Критерій	Laravel	Django
Архітектура	Model-View-Controller	Model-View-Template
Мова програмування	PHP	Python
Поширеність і зручність засобів роботи з даними і деревами рішень	Менша	Більша
Продуктивність роботи	Менша	Значно більша
Безпека роботи	Надає тільки базові засоби	Застосовує найкращі сучасні практики

Враховуючи, що вебфреймворк Django краще підтримує роботу з даними щодо їх аналізу загалом, оброблення, а також зокрема надає потужні засоби для роботи з деревами рішень, відзначається кращою продуктивністю роботи, адже за одну секунду обробляє значно більше даних порівняно з альтернативним рішенням, та окрім того ще й надає кращі вбудовані засоби підтримки безпеки роботи, тому його було обрано для розробки.

Даний вибір фреймворку Django і мови програмування Python визначає і відповідну архітектуру вебзастосунку, адже в підсумку він побудований на основі використання моделі Model-View-Template, тобто основними одиницями є моделі, представлення, шаблони та менеджер адрес.

3.2 Моделювання роботи з програмою

Програмна система, що розробляється, в підсумку через всі інтерфейси повинна надавати такі функціональні можливості:

- створення і ведення каталогів страв і добавок для мережі закладів;
- визначення особливостей страв і добавок для закладу харчування;
- створення і ведення загального каталогу добавок до страв;
- створення і ведення загального каталогу категорій страв;
- визначення вартості варіантів доставки сервісом;
- замовлення страв у закладах харчування;
- керування замовленнями;
- скасування страви або добавки з замовлення;
- вподобання або невподобання страви;
- оцінювання закладу харчування;
- перегляд відгуків про заклад харчування;
- перегляд відгуків про страву;
- класифікація замовлення за часом доставки;
- визначення переліку замовлень для доставки кур'єром;
- визначення переліку замовлень для опрацювання закладом харчування;
- тимчасове зняття страви або добавки з можливості доставки;
- звітування щодо продажу страв закладом;
- звітування щодо задоволеності клієнтів закладом і стравами;
- пошук закладів харчування;

- відображення доступного меню для замовлення у закладі харчування;
- перегляд власних замовлень;
- авторизація користувачів;
- реєстрація мереж закладів харчування і їх представників;
- реєстрація закладів харчування у мережі і їх представників;
- реєстрація кур'єрів;
- перегляд власних відгуків.

Прецеденти для роботи різних користувачів з програмною системою представлені у вигляді відповідних діаграм для клієнта (рис. 3.1), кур'єра (рис. 3.2), адміністратора мережі закладів (рис. 3.3).

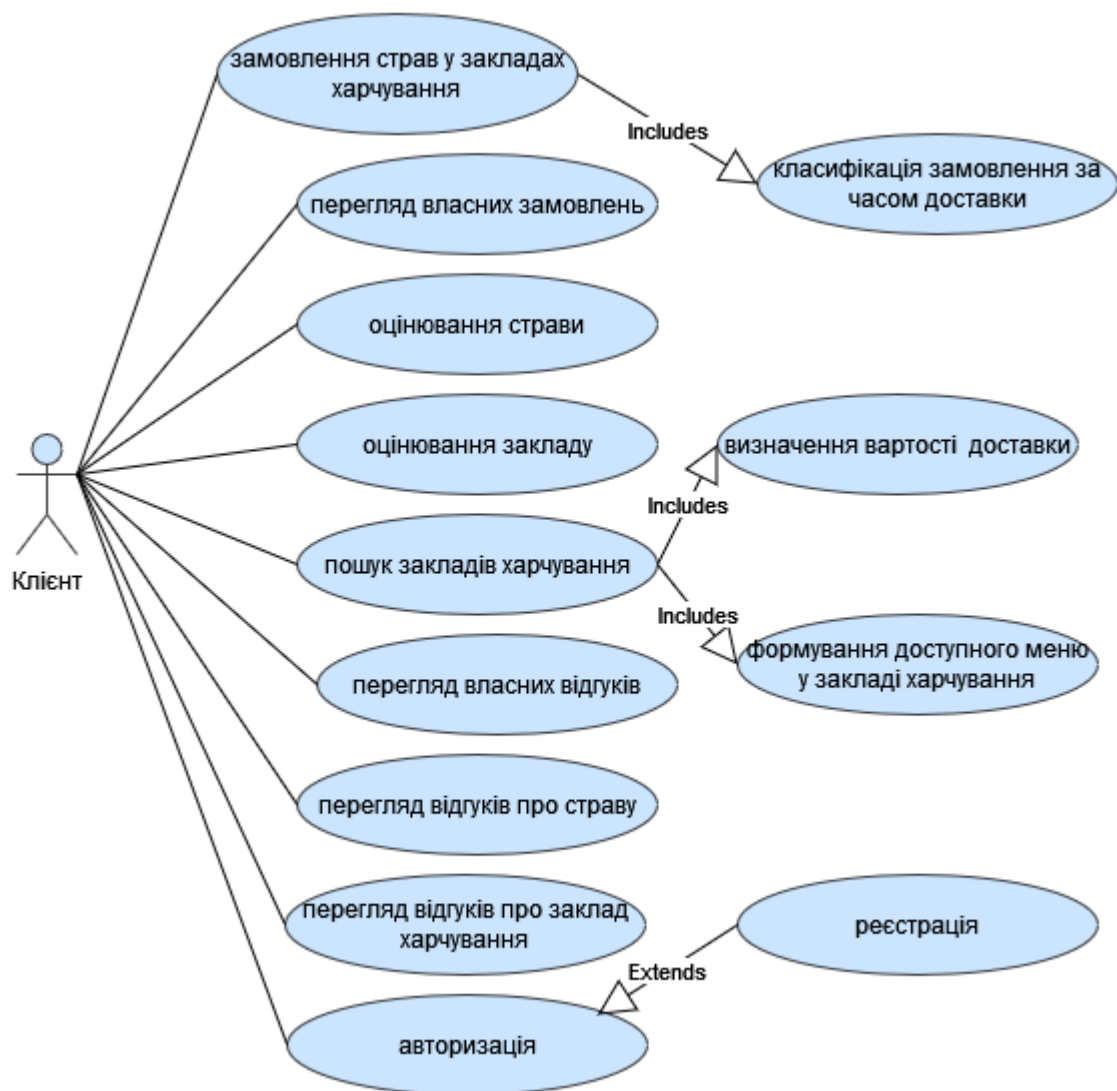


Рисунок 3.1 – Діаграма прецедентів для клієнта



Рисунок 3.2 – Діаграма прецедентів для кур'єра

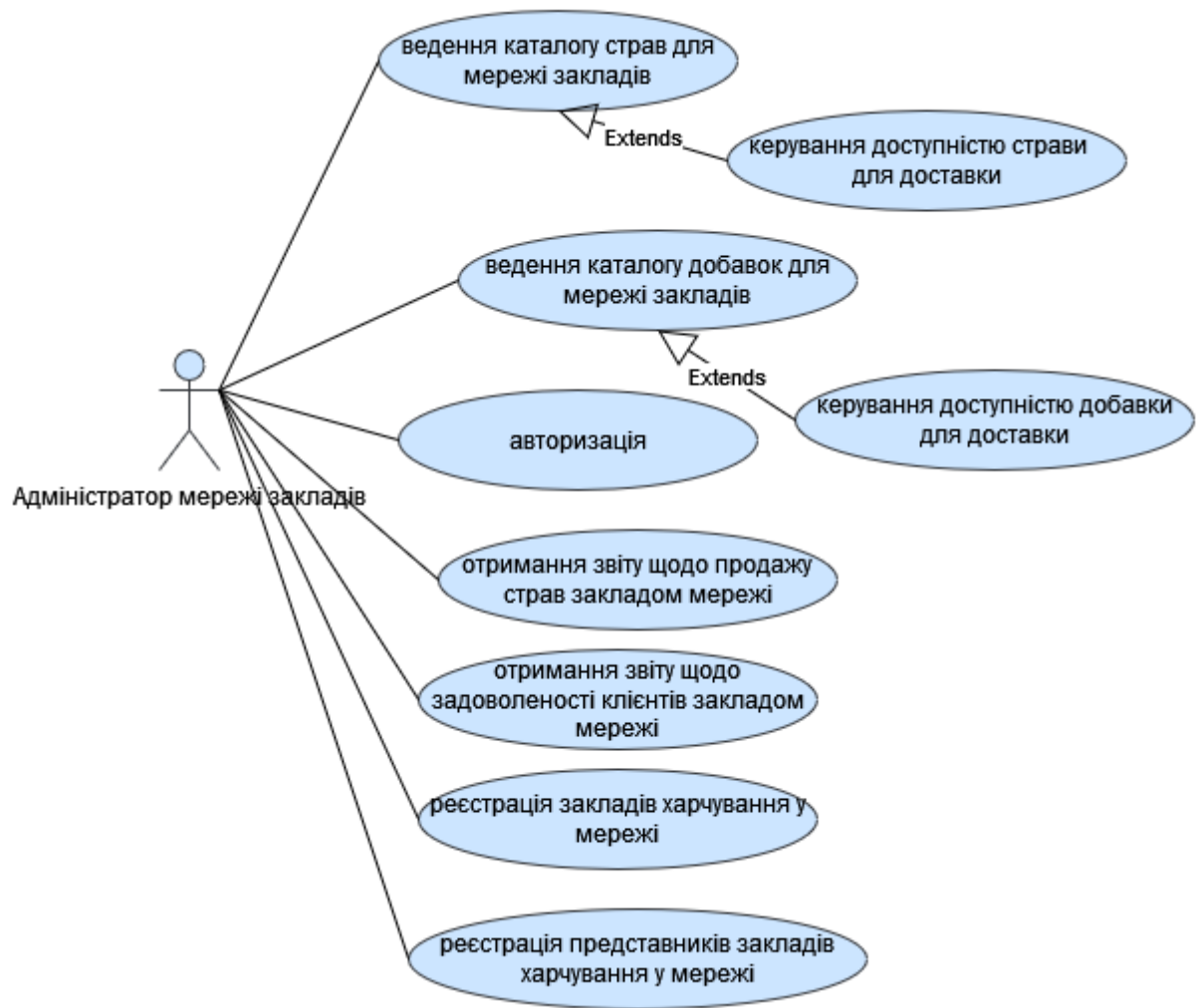


Рисунок 3.3 – Діаграма прецедентів для адміністратора мережі закладів

Для менеджера (рис. 3.4) та адміністратора (рис. 3.5) закладу харчування визначено відповідні прецеденти для їх роботи з системою у вигляді відповідних діаграм.

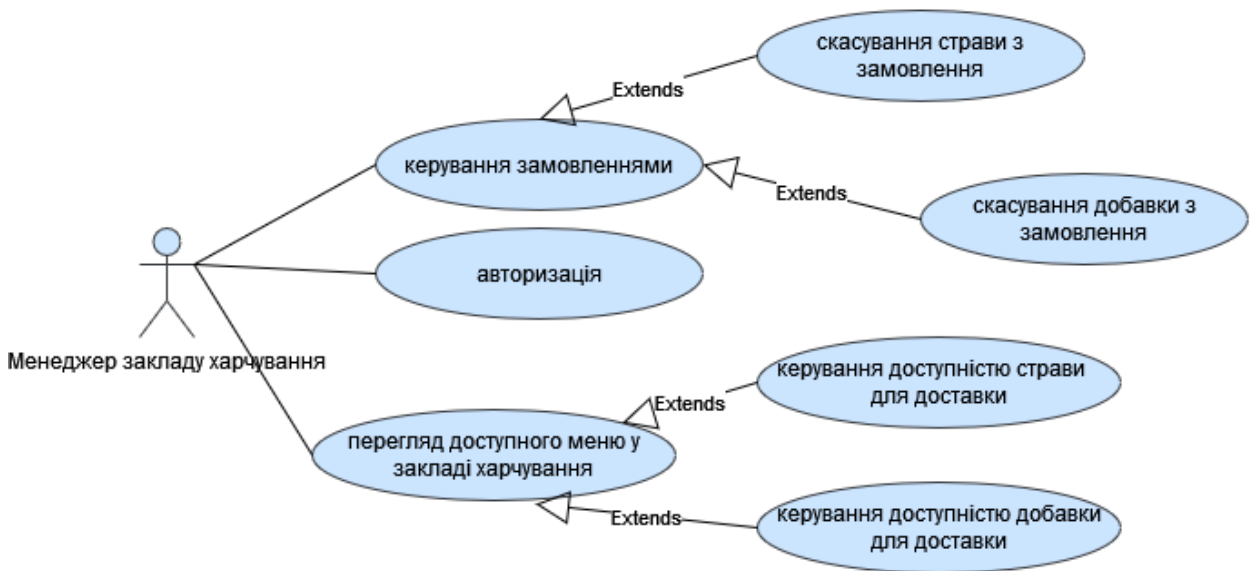


Рисунок 3.4 – Діаграма прецедентів для менеджера закладу харчування

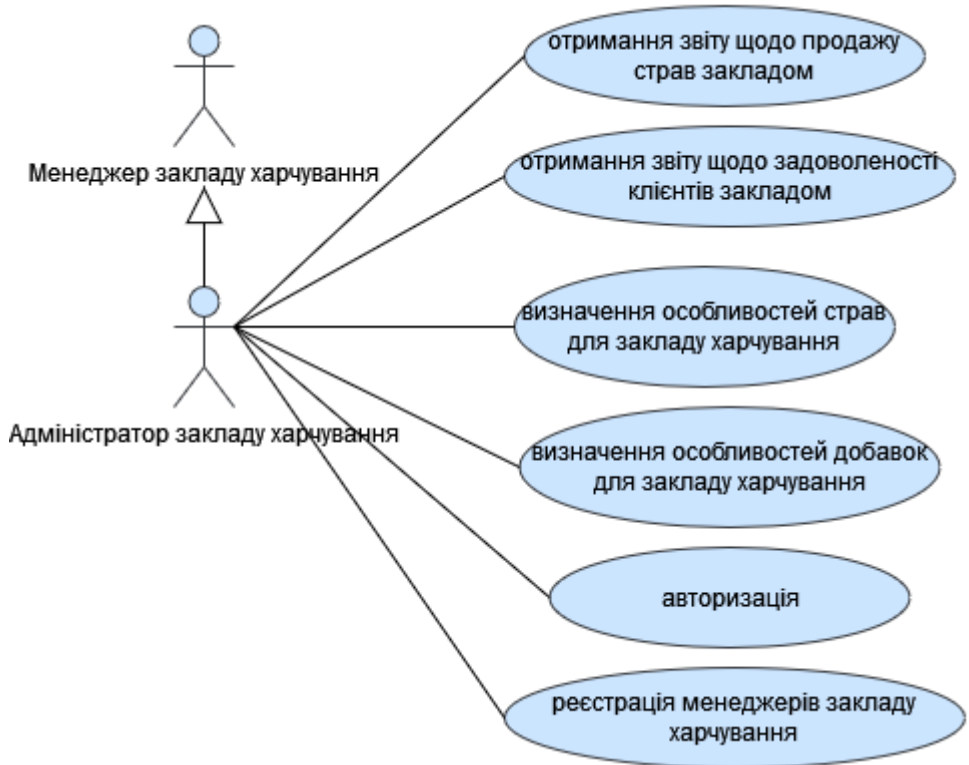


Рисунок 3.5 – Діаграма прецедентів для адміністратора закладу харчування

Для адміністратора системи (рис. 3.6) та менеджера сервісу доставки їжі (рис. 3.7) закладу харчування визначено відповідні прецеденти для їх роботи з системою у вигляді відповідних діаграм.

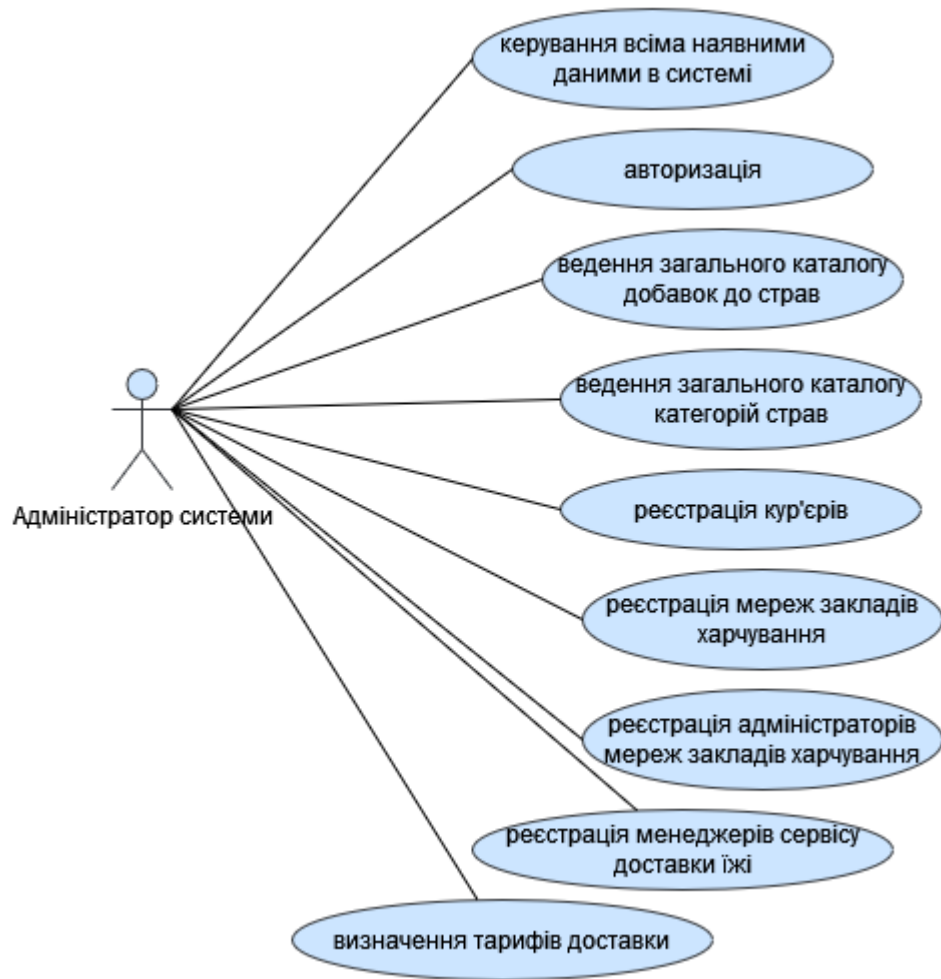


Рисунок 3.6 – Діаграма прецедентів для адміністратора системи

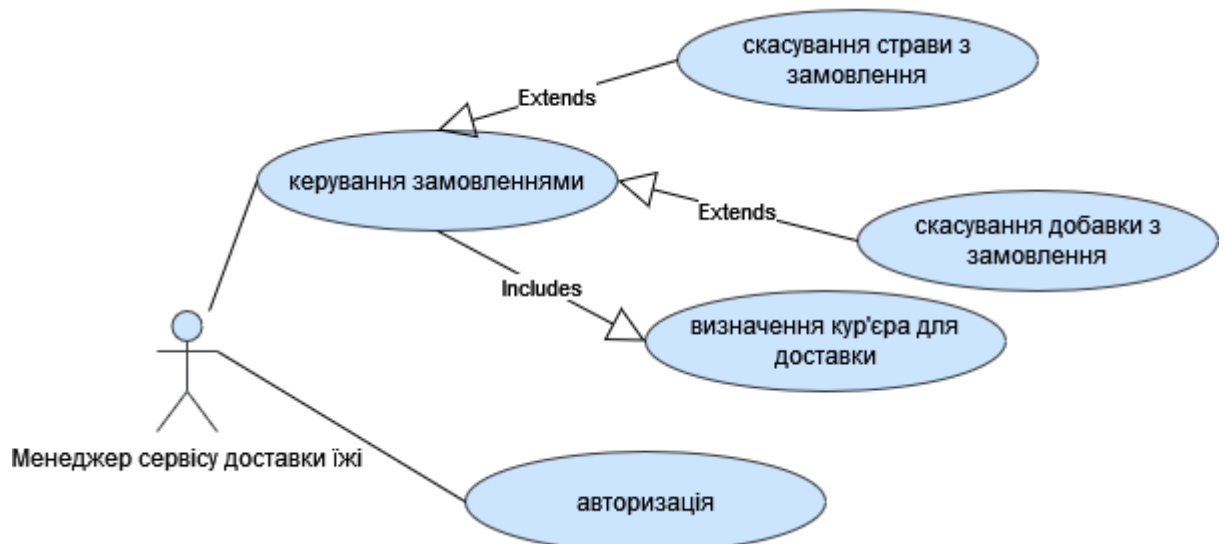


Рисунок 3.7 – Діаграма прецедентів для менеджера сервісу доставки їжі

3.3 Проєктування бази даних

Проєктування бази даних було реалізовано у відповідності з тими функціональними вимогами, які було описано, з результатами аналізу аналогів загалом і обраного прототипу зокрема та враховуючи особливості програмної системи, що розробляється. Для реалізації самої бази даних і логіки роботи з нею було використано систему керування базами даних MySQL, що за результатами багаторічного використання для програмних систем, що витримують великі користувацькі навантаження, продемонструвала свою надійність.

Інформація про працівників включає інформацію про працівників сервісу доставки їжі, працівників закладів харчування і представляється в таблиці `employee`, де виділено поля:

- `id` – номер акаунта працівника;
- `name` – повне ім'я працівника;
- `username` – логін, що відповідає акаунту працівника;
- `password` – пароль, що відповідає акаунту працівника;
- `email` – електронна пошта працівника;
- `phone` – номер телефону працівника;
- `job` – позиція, на якій працює працівник: адміністратор закладу харчування, менеджер закладу харчування, менеджер служби доставки, адміністратор служби доставки;
- `establishmentdepartment` – заклад харчування, в якому працює працівник, якщо він працює на позиції, що пов'язана з конкретним закладом харчування (тобто не працівник служби доставки), є зовнішнім ключем.

Інформація про кур'єрів служби доставки виділена окремо і представляється в таблиці `courier`, де виділено поля:

- `id` – номер акаунта кур'єра;
- `name` – повне ім'я кур'єра;
- `username` – логін, що відповідає акаунту кур'єра;

- password – пароль, що відповідає акаунту кур'єра;
- email – електронна пошта кур'єра;
- phone – номер телефону кур'єра;
- vehicle – транспортний засіб кур'єра;
- region_x – абсциса центральної точки Global Positioning System (GPS) ділянки, на якій працює кур'єр;
- region_y – ордината центральної точки GPS ділянки, на якій працює кур'єр;
- region_distance – відстань від центральної точки ділянки, на якій працює кур'єр;
- jobstart – час початку роботи кур'єра;
- jobend – час завершення роботи кур'єра.

Інформація про категорії доставки їжі представляється в таблиці `deliverycategory`, де виділено поля:

- id – номер категорії доставки їжі;
- title – назва категорії доставки їжі;
- presentation – опис категорії доставки їжі;
- price – вартість доставки для цієї категорії;
- maxdistance – максимальна відстань, на яку здійснюється доставка.

Інформація про мережі закладів харчування представляється в таблиці `foodestablishment`, де виділено поля:

- id – номер мережі закладів харчування;
- title – назва мережі закладів харчування;
- presentation – опис мережі закладів харчування;
- picture – зображення (логотип) мережі закладів харчування;
- availability – відмітка про те, чи працює зараз дана мережа закладів харчування;
- phone – номер телефону мережі закладів харчування.

Інформація про заклади харчування в мережі (або заклад у мережі, якщо мережа відсутня, а це тільки окремий заклад харчування) представляється в таблиці `establishmentdepartment`, де виділено поля:

- `id` – номер закладу харчування;
- `foodestablishment` – мережа закладів харчування, до якої відноситься даний заклад (або фактично конкретна адреса закладу, якщо відсутня мережа як така, тобто це окремий заклад), є зовнішнім ключем;
- `title` – назва закладу харчування;
- `presentation` – опис закладу харчування;
- `picture` – фотографія закладу харчування;
- `region` – регіон, в якому знаходиться заклад харчування;
- `address` – адреса закладу харчування (словесне вираження);
- `address_x` – перша GPS-координата закладу харчування;
- `address_y` – друга GPS-координата закладу харчування;
- `stars` – середня оцінка закладу харчування;
- `votes` – кількість оцінок закладу харчування;
- `availability` – відмітка про те, чи працює зараз даний заклад харчування;
- `phone` – номер телефону закладу харчування;
- `jobstart` – час початку роботи закладу харчування;
- `jobend` – час завершення роботи закладу харчування.

Інформація про страви в каталогу мережі представляється в таблиці `dishtype`, де виділено поля:

- `id` – номер страви в каталозі мережі;
- `foodestablishment` – мережа закладів харчування, в якій подається дана страва (зовнішній ключ);
- `dishcategory` – категорія страви (зовнішній ключ);
- `title` – назва страви в каталозі мережі закладів харчування;
- `presentation` – опис страви з каталогу мережі закладів харчування;
- `picture` – фотографія страви;

– price – вартість страви (може бути не зазначена, тоді передбачається, що вартість може бути різною в різних закладах мережі, а якщо зазначена, то тоді вартість встановлюється однаковою);

– availability – примітка, яка визначає, чи доступна на даний момент дана страв з каталогу в мережі закладів харчування (якщо вказано, що ні, то в будь-якому з закладів мережі її не можна буде замовити).

Інформація про страви в конкретних закладах (на основі уточнення даних з каталогу) представляється в таблиці dish, де виділено поля:

– id – номер страви;

– establishmentdepartment – заклад харчування, в якому виготовляється ця страв (зовнішній ключ);

– dishtype – страв в каталозі мережі закладів харчування, якій відповідає дане уточнення (зовнішній ключ);

– presentation – уточнюючий опис страви (необхідний для ситуацій, коли страв має певні особливості, а якщо не задано опис, то передбачається використання опису з запису про страву в каталозі);

– picture – фотографія страви (необхідна для ситуацій, коли використовується окрема фотографія, а якщо не задано, то використовується загальна для мережі фотографія);

– posvotes – кількість позитивних відгуків про страву, виготовлену в даному закладі;

– negvotes – кількість негативних відгуків про страву, виготовлену в даному закладі;

– price – вартість страви (якщо не задано, то вартість дорівнює вартості страви з каталогу, яка тоді рівна для всіх закладів мережі);

– availability – примітка, яка визначає, чи доступна на даний момент дана страв в цьому закладі харчування;

– starttime – час, з якого готується страв;

– endtime – час, до якого готується страв.

Останні два поля необхідні для того, щоб визначити доступність страви тільки в певний момент кожного дня. Це необхідно, якщо, наприклад, страва готується тільки на сніданок і не доступна в інший час.

Інформація про замовлення представляється в таблиці `order`, де виділено поля:

- `id` – номер замовлення;
- `client` – клієнт, який створив замовлення (зовнішній ключ);
- `instruction` – інструкції клієнта щодо особливостей замовлення;
- `created` – час створення замовлення;
- `state` – поточний стан замовлення;
- `updated` – час останнього оновлення замовлення;
- `deliveryprice` – вартість доставки замовлення;
- `price` – сумарна вартість замовлення;
- `address` – адреса доставки (словесне вираження);
- `address_x` – перша GPS-координата адреси доставки;
- `address_y` – друга GPS-координата адреси доставки;
- `phone` – номер телефону, за яким можна зв'язатися з клієнтом за цим замовленням;
- `cancel` – варіант політики скасування для замовлення, який обраний клієнтом (за відсутності страви скасувати все замовлення, за відсутності страви все інше замовлення доставити, за відсутності страви з'єднатися з клієнтом для прямого уточнення);
- `delivered` – час, коли було фактично доставлено замовлення;
- `deliveryclassified` – інтервал часу доставки, до якого початкового за прогнозом було віднесено замовлення;
- `courier` – кур'єр, який призначений для доставки замовлення (зовнішній ключ).

Інформація про конкретні страви в замовленні представляється в таблиці `orderdish`, де виділено поля:

- `id` – номер страви, яка була замовлення;

- order – замовлення, до якого входить страва (зовнішній ключ);
- dish – страва, яка входить до замовлення (зовнішній ключ);
- instruction – спеціальні інструкції клієнта щодо особливостей страви;
- number – кількість одиниць страви у замовленні;
- price – вартість страви у замовленні;
- availability – примітка про наявність страви з замовлення (необхідна для визначення ситуацій, коли за замовленням не вдалось зготувати страву);
- state – стан приготування страви з замовлення.

Інформація про категорії добавок представляється в таблиці `additioncategory`, де виділено поля:

- id – номер категорії добавок;
- title – назва категорії добавок.

Інформація про конкретні добавки представляється в таблиці `addition`, де виділено поля:

- id – номер добавки;
- additioncategory – категорія добавок, до якої відноситься дана (зовнішній ключ);
- name – назва добавки;
- picture – фотографія добавки.

Інформація про добавки до страв мережі закладів харчування представляється в таблиці `dishtypeaddition`, де виділено поля:

- id – номер добавки до страв мережі закладів харчування;
- dishtype – страва з каталогу мережі, до якої подається ця добавка (зовнішній ключ);
- addition – добавка, яка до цієї страви подається (зовнішній ключ);
- title – назва добавки (якщо не задано, використовується стандартна назва цієї добавки з таблиці `addition`);
- picture – фотографія добавки до цієї страви;
- price – вартість добавки до цієї страви;

– availability – примітка, яка визначає, чи доступна в даний момент ця добавка для цієї страви в мережі закладів.

Інформація про конкретні добавки до страв у тих варіантах, які подаються в конкретних закладах харчування мережі (фактично уточнення), представляється в таблиці dishaddition, де виділено поля:

- id – номер уточнення добавки до конкретної страви в конкретному закладі харчування;
- dishtypeaddition – добавка до страви мережі закладів харчування, до якої відноситься уточнення (зовнішній ключ);
- dish – конкретна страв з конкретного закладу харчування мережі (зовнішній ключ), до якої подається ця добавка;
- title – назва добавки в даному закладі;
- price – вартість добавки;
- availability – примітка про наявність добавки в даний момент.

Інформація про добавки в складі замовлень представляється в таблиці orderaddition, де виділено поля:

- id – номер добавки з замовлення;
- order – замовлення, до якого внесено цю добавку (зовнішній ключ);
- dishaddition – конкретна добавка, яку було внесено до замовлення (зовнішній ключ);
- number – кількість одиниць цієї добавки у замовленні;
- price – вартість цих добавок у замовленні;
- state – стан обслуговування добавки з замовлення (необхідно на випадок фактичної відсутності цієї добавки).

Інформація про категорії страв представляється в таблиці dishcategory, де виділено поля:

- id – номер категорії страв;
- title – назва категорії страв.

Інформація про відгуки про страви в закладах представляється в таблиці dishreview, де виділено поля:

- id – номер відгуку;
- dish – страва з конкретного закладу, про яку сформовано відгук (зовнішній ключ);
- client – клієнт, який створив відгук (зовнішній ключ);
- vote – позитивне або негативне загалом оцінювання страви (уподобано або ні);
- created – час створення відгуку;
- comment – коментар до відгуку.

Інформація про відгуки про заклади представляється в таблиці `establishmentreview`, де виділено поля:

- id – номер відгуку;
- establishmentdepartment – конкретний заклад мережі, про який створено відгук (зовнішній ключ);
- client – клієнт, який створив відгук (зовнішній ключ);
- stars – оцінка відгуку;
- created – час створення відгуку.

Інформація про клієнтів представляється в таблиці `client`, де виділено поля:

- id – номер акаунту клієнта;
- name – повне ім'я клієнта;
- username – логін, що відповідає акаунту клієнта;
- password – пароль, що відповідає акаунту клієнта;
- email – електронна пошта клієнта;
- phone – номер телефону клієнта.

Інформація про співвідношення між категоріями страв та категоріями доставки представляється в таблиці `dishdeliverycategory`, де виділено поля:

- id – номер співвідношення;
- dishcategory – категорія страви (зовнішній ключ);
- deliverycategory – категорія доставки (зовнішній ключ).

Схема бази даних у загальному вигляді є на рис. 3.8.

Інформація про регіони представляється в таблиці `region`, де виділено поля:

- `id` – номер регіону;
- `name` – назва регіону;
- `distance` – відстань, на якій в регіоні діє доставка.

Інформація про працівників мережі закладів харчування представляється в таблиці `networkemployee`, де виділено поля:

- `id` – номер акаунта працівника мережі закладів харчування;
- `name` – повне ім'я працівника;
- `username` – логін, що відповідає акаунту працівника;
- `password` – пароль, що відповідає акаунту працівника;
- `email` – електронна пошта працівника;
- `phone` – номер телефону працівника;
- `job` – позиція, на якій працює працівник, адміністратор мережі

закладів харчування;

- `foodeestablishment` – мережа закладів харчування, в якій працює працівник.

Інформація про класифікацію замовлень за класом тривалості доставки представляється в таблиці `orderclassification`, де виділено поля:

- `id` – номер запису;
- `order` – замовлення;
- `courrier` – кур'єр, який може виконувати доставку;
- `classified` – результат класифікації.

3.4 Висновки за розділом 3

У третьому розділі розглянуто засоби, які можуть використовуватися для розробки програмної системи, а в підсумку обрано мову програмування Python, фреймворк розробки вебзастосунків Django для реалізації і відповідно шаблон Model-View-Template для побудови архітектури

вебзастосунку, виконано моделювання роботи з програмною системою на основі відповідних діаграм прецедентів для різних акторів та проєктування бази даних.

4 ОСНОВНІ РІШЕННЯ ЩОДО РЕАЛІЗАЦІЇ КОМПОНЕНТІВ СИСТЕМИ

4.1 Структура програмної системи

Програмну систему побудовано на основі використання моделі Model-View-Template зі створенням цілого ряду інтерфейсів, які в залежності від ролі користувача визначають варіант його роботи з програмною системою. У той же час вся інша структура системи побудована за єдиним принципом, тобто розподіл відбувається тільки на рівні шаблонів і системи керування переходами між цими шаблонами. Система переходів між шаблонами визначається самими шаблонами та менеджером адрес, який знаходиться в файлі `urls.py`. Самі шаблони знаходяться в папці `template` і визначають основу динамічного формування вебсторінок, які відповідають функціональним вимогам до програми. Для визначення дизайну вебсторінок використано шаблон [12]. На цих вебсторінках в залежності від ролі користувача відображається відповідний склад меню, який визначає варіанти переходу між вебсторінками.

Усі моделі вебзастосунку знаходяться в файлі `models.py` і відповідають схемі бази даних, в чіткій відповідності з якою вони і були створені.

Усі основні форми, які використовуються на вебсторінках, створені в файлі `forms.py`. До таких форм зокрема належать `EDCreateForm`, що визначає заповнення даних щодо внесення і оновлення даних мережі закладів харчування, `SearchForm`, що визначає завдання параметрів пошуку закладів харчування, які цікавлять клієнта, `FECreateForm`, що визначає заповнення даних щодо внесення і оновлення даних закладів харчування, `EReviewCreateForm`, що визначає створення відгуку про заклад харчування тощо.

Усі представлення вебзастосунку розділені одразу на декілька файлів, що включають:

- представлення, пов’язані з закладами і мережами харчування – файл `views_establishments.py`;
- представлення, пов’язані зі стравами і добавками – файл `views_dishes.py`;
- представлення, пов’язані з роботою з замовленнями – файл `views_orders.py`;
- представлення, пов’язані з аутентифікацією, реєстрацією користувачів і керування їх ролями – файл `views_users.py`.

Деякі з цих класів описані в підрозділі 4.2.

4.2 Опис розроблених класів

Принципи розроблення і структуризації класів усередині вебзастосунку було описано в попередньому підрозділі. Нижче буде описано принципи створення деяких основних класів представлень вебзастосунку, які визначають наповнення даними відповідних шаблонів вебсторінок. Для реалізації безпосередньої роботи з деревами рішень використано пакет `scikit-learn`.

Клас `EstablishmentDepartmentMenuView` визначає підготовку даних щодо роботи з поточним меню закладу харчування. Відповідна вебсторінка має адресу `ed_menu.html` і задається атрибутом `template_name` класу. Даний клас створено на основі моделі `EstablishmentDepartment`, тобто базові дані містяться в класі моделі самого закладу харчування. Даний клас визначається шляхом наслідування класу `DetailView`, тобто визначає деталізоване представлення в даному випадку закладу харчування на сторінці.

У методі `get` даного класу відбувається наповнення даними самого об’єкту класу `EstablishmentDepartment` через визначення його номеру, який передається через останній параметр методу `get`, що має назву `pk`. Результат роботи даного методу, який передається у вигляді контексту до шаблону вебсторінки, включає окрім даних самого закладу ще набір даних стосовно

меню страв. Якщо поточний час не входить до часу роботи закладу, то в контексті даних для шаблону повертається параметр `paid` зі значенням `False`, який вказує на те, що замовлення страв у поточний момент неможливо, якщо ж повертається значення `True`, то це визначає, що зараз страви можна і замовити. Усі доступні в меню цього закладу харчування категорії страв повертаються через параметр `dish_categories` контексту даних. Останні 5 відгуків про заклад харчування повертаються через параметр `reviews` контексту даних. Параметр `dishes` містить перелік страв для кожної категорії страв, номер якої визначає відповідний індекс.

Клас `EstablishmentDepartmentReportView` визначає підготовку даних щодо формування звіту про роботу закладу харчування. Відповідна вебсторінка має адресу `ed_report.html` і задається атрибутом `template_name` класу. Даний клас створено на основі моделі `EstablishmentDepartment`, тобто базові дані містяться в класі моделі самого закладу харчування, тому клас і належить до файлу `views_establishments.py`. Даний клас визначається шляхом наслідування класу `DetailView`, тобто визначає деталізоване представлення.

Для визначення роботи методу `get` спочатку встановлюються дані закладу харчування, після цього визначаються страви, які в даному закладі замовлялись протягом останнього року (якщо поточний місяць – січень, то звіт надається за минулий рік), далі до інформації про кожну страву додається інформація про статистику за кожен місяць поточного (або відповідно попереднього року), включаючи: кількість разів, яку було замовлено цю страву, сумарний дохід, який було отримано від продажу даної страви за місяць, та середню вартість страви за цей період. Відповідно в контексті, що повертається в підсумку, за ключем `dishes` зберігаються дані про кожну страву у вигляді переліку. Кожен елемент даного переліку включає перелік, що складається зі словників, кожен з яких містить відповідні статистичні показники.

Подібний клас `EstablishmentDepartmentYearReportView` визначає звітування за цілий рік роботи закладу харчування сумарно.

Реєстрація закладів харчування реалізується через клас `EstablishmentDepartmentCreateView`, який реалізовано наслідуванням класів `LoginRequiredMixin`, що визначає потребу в авторизації, та базового класу представлення `View`. В основі побудови даного класу використано форму `EDCreateForm` для внесення відповідних даних про заклад, який додається до мережі. Додатково при передаванні структури форми до вебсторінки в методі `get(self, request, pk=None)` перевіряється, чи є авторизований в поточний момент користувач працівником мережі закладів. У випадку, якщо це не так, повертається вебсторінка з помилкою 403.

У методі `post(self, request, pk=None)` даного класу відбувається перевірка коректності введених у формі даних через виклик методу форми `is_valid()`, після чого, у випадку коректності, реалізується обробка даних. Оскільки користувач задає тільки адресу закладу, то на основі цієї адреси з використанням клієнта вебсерверів геокодування геору за допомогою виклику геокодера `OpenStreetMap`, `Nominatim`, з викликом методу `geocode` реалізується створення об'єкта, з якого в якості складової виділяється назва регіону, широта та довгота GPS-координати закладу. Цей підхід дозволяє уникнути необхідності визначення адреси користувачем декількома способами одночасно або необхідності визначення адреси за допомогою точки на карті або визначення заданих координат взагалі вручну. Окрім того вносяться нульові значення кількості голосів, оцінки. У відповідності з даними про те, до якої мережі закладів харчування прив'язаний працівник мережі закладів харчування, визначається, до якої мережі закладів відноситься новостворений заклад харчування. Усе це в підсумку дозволяє повністю створити об'єкт, що відповідає новому закладу харчування.

На основі тієї ж форми організовано підтримку роботи класу `EstablishmentDepartmentUpdateView`, який відповідає за створення представлення оновлення даних існуючого закладу харчування. Даний клас також побудовано на основі двох методів та параметру `template_name`, який

визначає той же самий шаблон `stdepartment_form.html`, що використовується і для класу `EstablishmentDepartmentCreateView`.

Метод `get(self, request, pk)` реалізує на основі GET-запиту початкове заповнення даними форми `EDCreateForm` у відповідності з поточними характеристиками закладу харчування, для того щоб потім користувач міг потрібним чином змінити ці значення через форму. Для цього запитуються дані самого об'єкту моделі `EstablishmentDepartment`, що відповідає номеру `pk`, визначеного параметром методу. Також перевіряється, чи є авторизований користувач працівником закладу харчування, чи мережі закладів харчування. Якщо це так, то повертається шаблон, заповнений відповідними даними. Якщо ні, то повертається вебсторінка з кодом помилки 403.

Метод `post(self, request, pk=None)` у свою чергу реалізує підтримку роботи з формою через метод POST. Для цього спочатку виконується перевірка коректності заповнення форми даними на основі того ж самого способу, що було описано вище, потім у випадку некоректності відкривається сторінка повторно, а у випадку коректності – реалізується уточнення даних за новою адресою закладу харчування. Для цього аналогічно описаному вище способу встановлюються GPS-координати за вказаною адресою, визначається регіон, де знаходиться дана адреса. Після цього також відбувається автоматичне внесення даних про мережу закладів харчування. У підсумку відкривається вебсторінка з інформацією про мережу закладів харчування, до якої належить оновлений заклад харчування.

Відповідний клас представлення деталізації мережі закладів харчування `FoodEstablishmentView` створений шляхом наслідування класу `DetailView`. В якості моделі класу визначено клас `FoodEstablishment`, а в якості шаблону задано `fe.html`, що визначає виведення інформації про мережу закладів. Метод `get(self, request, pk)` реалізує заповнення контексту даних для сторінки шаблону. Для цього спочатку відбувається витягання даних про мережу закладів на основі звернення до відповідного масиву об'єктів

моделей класу `FoodEstablishment` з поверненням відповідної мережі під номером, що відповідає параметру `pk` методу. Після цього реалізується заповнення даними закладів харчування, що відповідають даній мережі: заклади відфільтровуються з відповідного масиву об'єктів моделей класу `EstablishmentDepartment` за заданою мережею закладів. Увесь список таких закладів зберігається в список `establishments`. У подальшому всі заклади зі списку `establishments` розподіляються за регіонами, в яких ці заклади знаходяться. У підсумку формується список словників `regioned_establishments`. Кожен елемент даного списку містить словник, що містить ключі:

- `info`, за яким зберігається інформація про об'єкт регіону;
- `establishments`, за яким зберігається перелік закладів з відфільтрованого переліку `establishments`, що відносяться до регіону, який міститься за ключем `info`.

У результаті дані про мережу закладів та список `regioned_establishments` формують контекст даних для сторінки вебшаблону і разом повертаються за результатами роботи цієї функції.

4.3 Висновки за розділом 4

У четвертому розділі виконано реалізацію програмної системи і підведено підсумки виконання даного етапу. В результаті було створено програмну систему, що надає інтерфейси для клієнтів, представників мереж закладів харчування, окремих закладів харчування та сервісу доставки їжі загалом. За результатами реалізації описано принципи створення деяких класів у складі вебзастосунку.

5 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ

5.1 Опис роботи з програмою

Розроблена програмна система передбачає доступ до неї через адресу вебсервера. Загальна адреса вебсервера за замовчуванням визначає доступ до інтерфейса клієнта. Інтерфейс клієнта дозволяє виконати пошук закладів харчування, визначаючи відповідні параметри (назва закладу, назва мережі, адреса, мінімальна і максимальна оцінки, час доставки, максимальна вартість страви). Після цього формується сторінка з переліком закладів, що відповідають результатам пошуку.

Обираючи потрібний заклад, можна перейти на його сторінку (рис. 5.1).

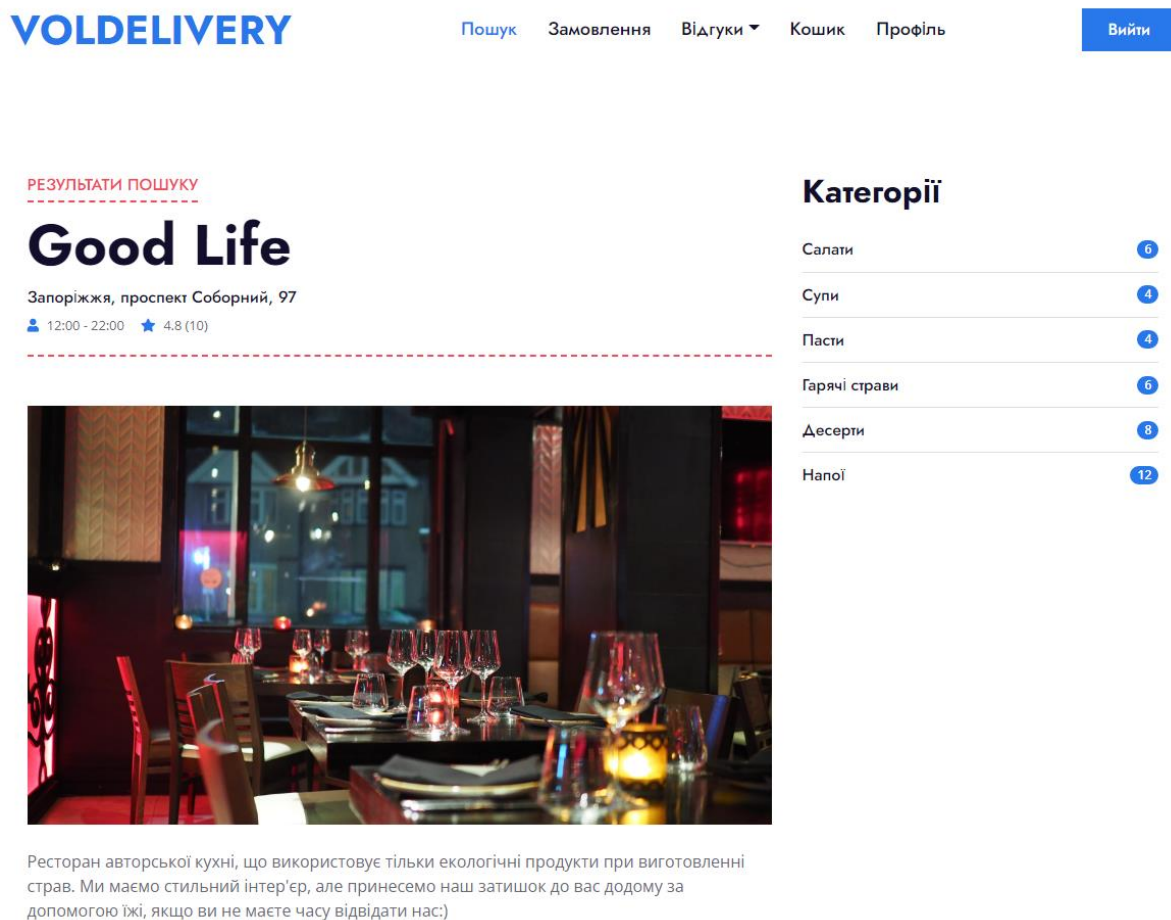


Рисунок 5.1 – Відображення вебсторінки закладу харчування

Якщо під час пошуку адреса не задається, то пошук відбувається за автоматично визначеною адресою клієнта.

На сторінці закладу відображається його фотографія, опис, назва, адреса, номер телефону (в контактах внизу сторінки), час роботи, середня оцінка та кількість встановлених оцінок різними клієнтами.

У правій частині сторінки відображається перелік категорій страв. Обираючи потрібну категорію, на сторінці внизу відображається перелік страв цієї категорії.

У переліку страв кожна відображається окремим блоком, що складається з фотографії страви та нанесеної на неї назви з приведенням в рядку внизу кількості позитивних оцінок, негативних оцінок, вартості однієї порції страви та посилання для замовлення страви (рис. 5.2).

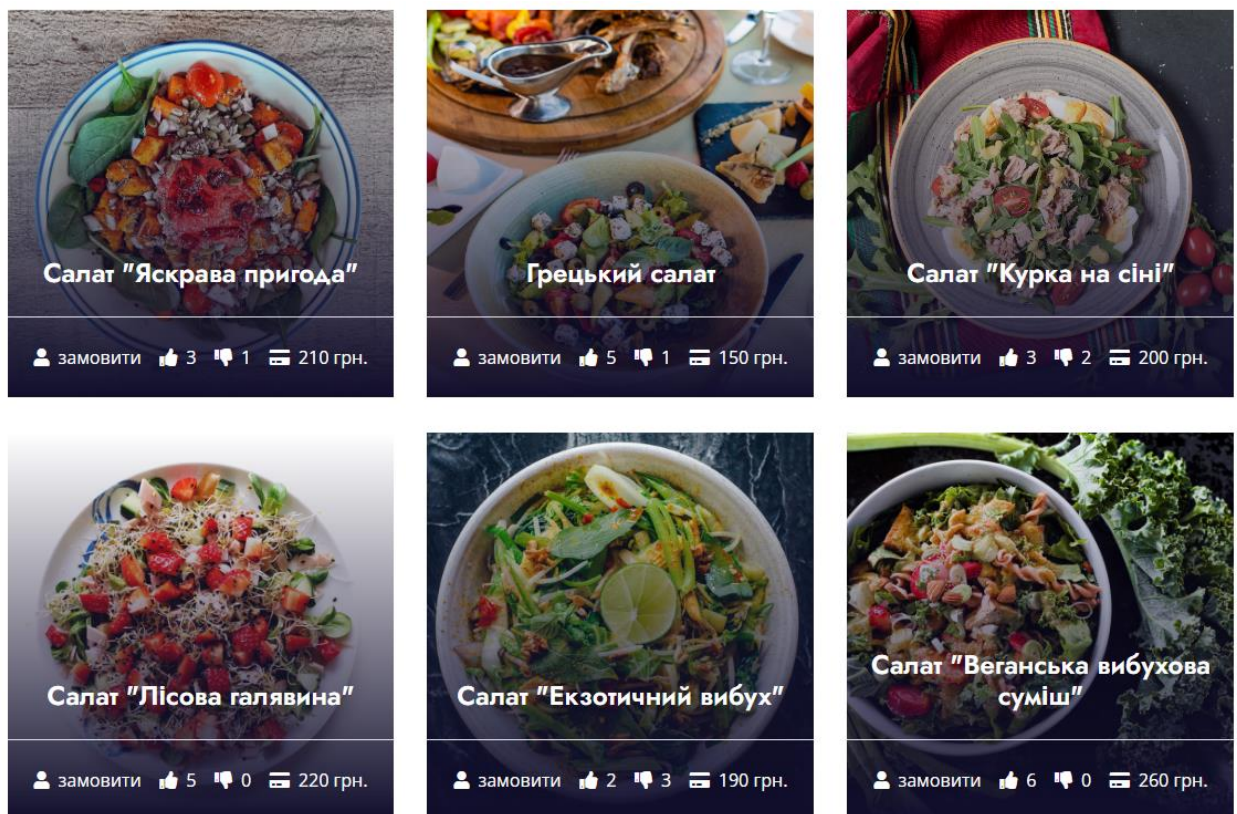


Рисунок 5.2 – Перегляд страв у категорії салатів на вебсторінці закладу харчування

Після вибору замовлення страви вона додається до кошика. Перейти до кошика можна, натиснувши на пункт меню «Кошик». З відповідної сторінки можна визначити кількість страв, обрати потрібні добавки для страви. Також можна натиснути на назву на страві на сторінці закладу і тоді відкриється сторінка з усіма доступними добавками, звідки можна сформулювати позицію по страві в замовленні повністю.

Відповідні пункти меню дозволяють також переглядати власні відгуки про заклади і страви, керувати власним профілем та отримувати доступ до переліку власних замовлень.

Стандартно при додаванні до адреси сервера розділу service відкривається інтерфейс менеджера сервісу доставки їжі (рис. 5.3).

VOLDELIVERY: CEPBIC

Нові
Оброблені
Старі
Профіль

Вийти

ЗАМОВЛЕННЯ #47 ВІД 18.11.2022 19:43

Сафроненко Сергій Вікторович
099-314-20-62
safronenkosv@gmail.com

Доставка: Запоріжжя, вулиця Олексія Поради, 45

Заклад: Python Cafe
Запоріжжя, проспект Соборний, 33

Вартість замовлення: 790 грн.
Додаткових інструкцій немає



Грибна юшка
Кількість: 2
Вартість: 500 грн.
Сметана: 2
Вартість: 20 грн.



Картопля фрі
Кількість: 2
Вартість: 200 грн.
Кетчуп: 2
Вартість: 20 грн.

Підтвердити замовлення

Скасувати все замовлення

Доставка

Кур'єр	Прогноз
Стойчев Андрій Автомобіль, добрий стан Поточних замовлень: 1	Успіх
Зоренко Віктор Автомобіль, відмінний стан Поточних замовлень: 1	Успіх
Тяжиков Петро Велосипед, відмінний стан Поточних замовлень: 3	Затримка

Вартість доставки: 50 грн.

Оберіть кур'єра:

Стойчев Андрій ▾

Рисунок 5.3 – Керування замовленням менеджером сервісу доставки їжі

Даний інтерфейс дозволяє менеджеру керувати замовленнями, визначаючи кур'єрів для доставки та відслідковуючи стан замовлень у виконанні, тому основними пунктами меню є «Нові» для визначення кур'єрів для новостворених замовлень, «Оброблені» для перегляду стану замовлень, які віддані в роботу, «Старі» для перегляду замовлень, які вже були доставлені клієнту.

На сторінці керування новими замовленнями менеджер отримує інформацію про перелік всіх замовлень і може перейти до конкретного замовлення, на сторінці якого йому в правій частині надається перелік доступних на цій ділянці кур'єрів, а за кожним з них інформація про транспортний засіб та прогнозований клас за тривалістю доставки замовлення. Внизу менеджер може обрати з випадального переліку кур'єра, якого він вважає за потрібне, а потім натиснути на кнопку «Підтвердити замовлення», щоб почати працювати з замовленням. Якщо натиснути на кнопку «Скасувати все замовлення», то замовлення не буде виконано.

Аналогічний інтерфейс для кур'єра, де він може отримувати інформацію про замовлення, які він в поточний момент повинен доставити, змінювати стан цих замовлень, доступний через розділ courier. Адміністратор системи може отримати доступ до свого інтерфейсу через розділ admin.

Інтерфейс для роботи з мережею закладів харчування доступний через розділ network і в лівому куті має в частині назви слово «мережа» («кур'єр» відповідно для інтерфейса кур'єра). Таким варіантом визначається робота і адміністратора, і менеджера мережі. Вони відрізняються доступною кількістю пунктів меню.

Інтерфейс для роботи з закладом харчування його адміністратора або менеджера доступний через розділ restaurant. У лівому куті сторінок даний інтерфейс визначається словом «заклад» (рис. 5.4). Варіанти роботи для адміністратора і менеджера відрізняються доступною кількістю пунктів меню.

Помісячний звіт

Poletto, мережа ItaRest

Pizza Carbonara ▾	Сформувати		
Місяць	Середня вартість	Продано, кількість	Продано, сумарно грн.
01.2022	-	-	-
02.2022	-	-	-
03.2022	-	-	-
04.2022	-	-	-
05.2022	-	-	-
06.2022	-	-	-
07.2022	-	-	-
08.2022	-	-	-
09.2022	270	1	270
10.2022	299	1	299
11.2022	299	10	2990

Рисунок 5.4 – Відображення звіту про роботу закладу

Адміністратору закладу доступний пункт меню «Звіти», який містить звіти за відгуками та звіти за продажем страв. Зіти за продажем страв бувають помісячні, а бувають річними, включаючи таким чином повну історію.

У випадку помісячного звіту обирається потрібна страва з меню закладу, а після натискання на кнопку «Сформувати» відображається таблиця з інформацією про середню вартість цієї страви, продану кількість та сумарний дохід за стравою протягом кожного відповідного місяця.

5.2 Експериментальне дослідження

Для виконання експериментального дослідження використовувалась вибірка даних [13], яка характеризує доставку їжі. Вибірка містить 56992 записів. Кожен запис у вибірці містить наступні характеристики:

- номер доставки;
- номер кур'єра;
- вік кур'єра;
- рейтинг кур'єра;
- широта координати закладу харчування;
- довгота координати закладу харчування;
- широта координати місця доставки;
- довгота координати місця доставки;
- дата замовлення;
- час замовлення;
- час, коли було отримано замовлення кур'єром;
- час доставки;
- погодні умови;
- щільність дорожнього трафіку;
- тип транспортного засобу;
- стан транспортного засобу;
- тип замовлення (тип страви);
- кількість одночасних замовлень;
- тип розташування.

За результатами проведеного аналізу вибірки було виявлено, що фактичний час доставки в результаті лежить в інтервалі від 10 до 54 хвилин, що звичайно є достатньо розпорошеним простором. Детальніше аналізуючи даний розподіл, було встановлено відсотковий розподіл для всіх цих значень тривалості доставки від 10 до 54 хвилин (рис. 5.5). Даний розподіл підтверджує, що сервіси доставки загалом прагнуть того, щоб доставка

виконувалась за 30 хвилин: у загальній масі замовлень тривалість доставки від 15 до 29 хвилин має вищі відсотки ніж доставка, яка триває більше 30 хвилин.

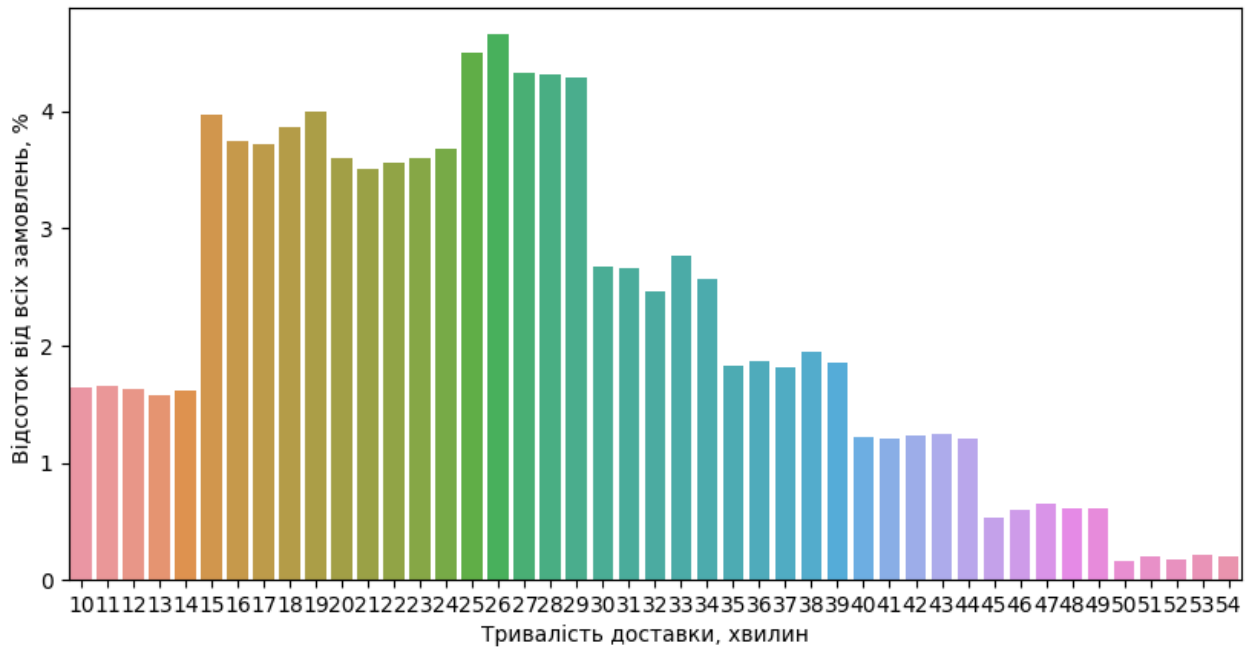


Рисунок 5.5 – Відсотковий розподіл тривалості доставки

Зважаючи на дану логіку, було виділено групи замовлень за тривалістю (інтервалом часу) доставки:

- від 10 до 19 хвилин (синій колір);
- від 20 до 29 хвилин (помаранчевий колір);
- від 30 до 39 хвилин (зелений колір);
- від 40 до 49 хвилин (червоний колір);
- від 50 до 59 хвилин (фіолетовий колір).

Тоді було проаналізовано відсотковий розподіл між цими групами (рис. 5.6). Переважний обсяг замовлень доставляється за час до 30 хвилин, при чому найбільший відсоток замовлень доставляється за час від 20 до 29 хвилин.

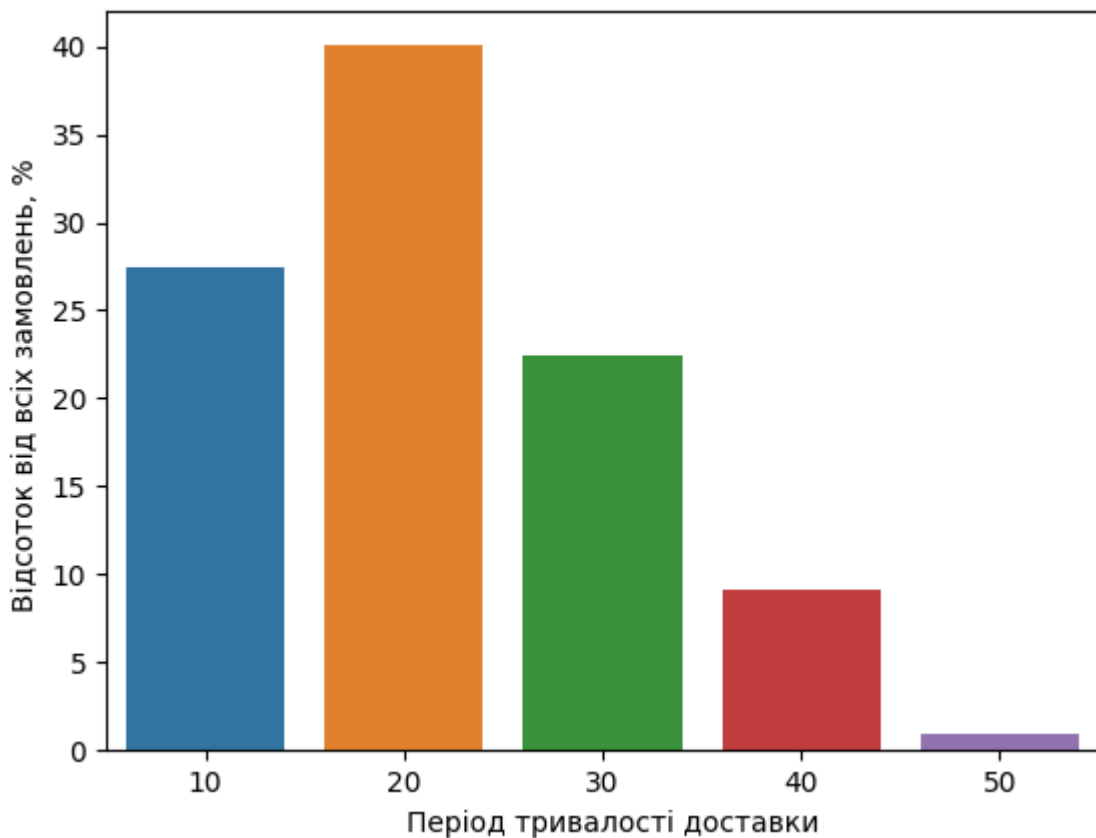


Рисунок 5.6 – Відсотковий розподіл інтервальної тривалості доставки для всіх замовлень

У подальшому було проаналізовано на основі наявної вибірки фактори, які фактори впливають на те, до якого класу за тривалістю доставки належить замовлення. Одним з основних факторів, які визначають в підсумку на тривалість доставки замовлення, є трафік на шляху доставки. Тому було окремо візуалізовано залежність інтервалу часу доставки від типу трафіку. Трафік характеризувався чотирма рівнями:

- низький;
- середній;
- високий;
- затори.

Порівнюючи результати між низьким (рис. 5.7) і середнім (рис. 5.8) рівнем трафіка, помітно, що відсоток замовлень з тривалістю доставки до 20 хвилин знижується, з тривалістю доставки більше 30 хвилин – збільшується.

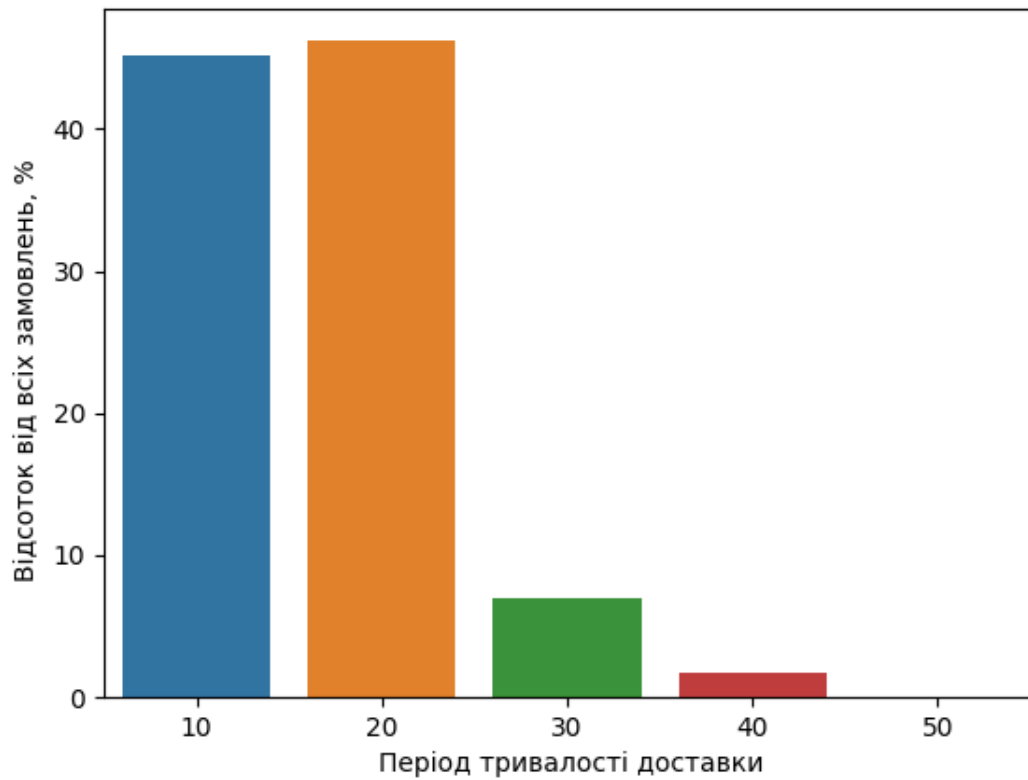


Рисунок 5.7 – Відсотковий розподіл інтервальної тривалості доставки для замовлень за низького рівня трафіка

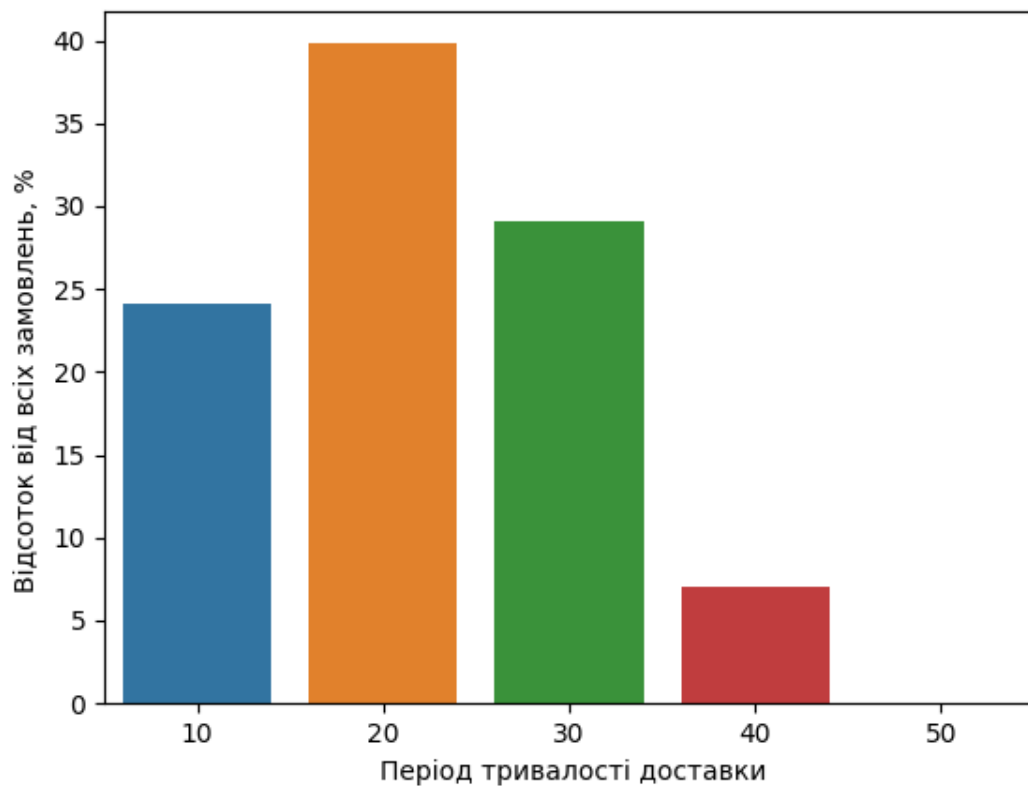


Рисунок 5.8 – Відсотковий розподіл інтервальної тривалості доставки для замовлень за середнього рівня трафіка

У подальшому, за високого рівня трафіка (рис. 5.9), не відбувається такої значної зміни відсоткового співвідношення між інтервалами тривалості доставки, проте відсоток замовлень з часом доставки до 20 хвилин зменшується, відсотки ж замовлень з часом доставки більше 20 хвилин дещо збільшуються.

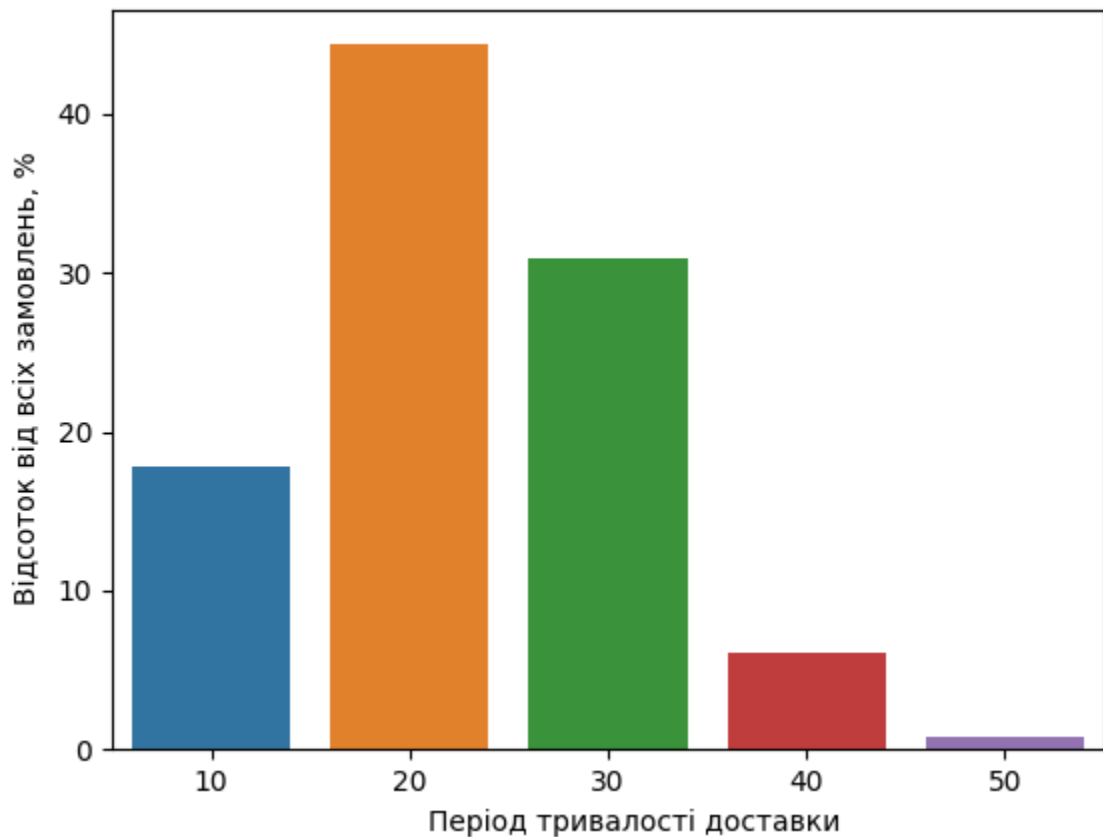


Рисунок 5.9 – Відсотковий розподіл інтервальної тривалості доставки для замовлень за високого рівня трафіка

За наявності заторів (рис. 5.10) відсоток замовлень з часом доставки загалом до 30 хвилин значно зменшується і відповідно більша частина замовлень доставляється за час більший за 30 хвилин. Значно збільшується при цьому відсоток замовлень, які доставляються за більше ніж 40 хвилин: 6 % за високого рівня трафіка і майже 20 % за наявності заторів.

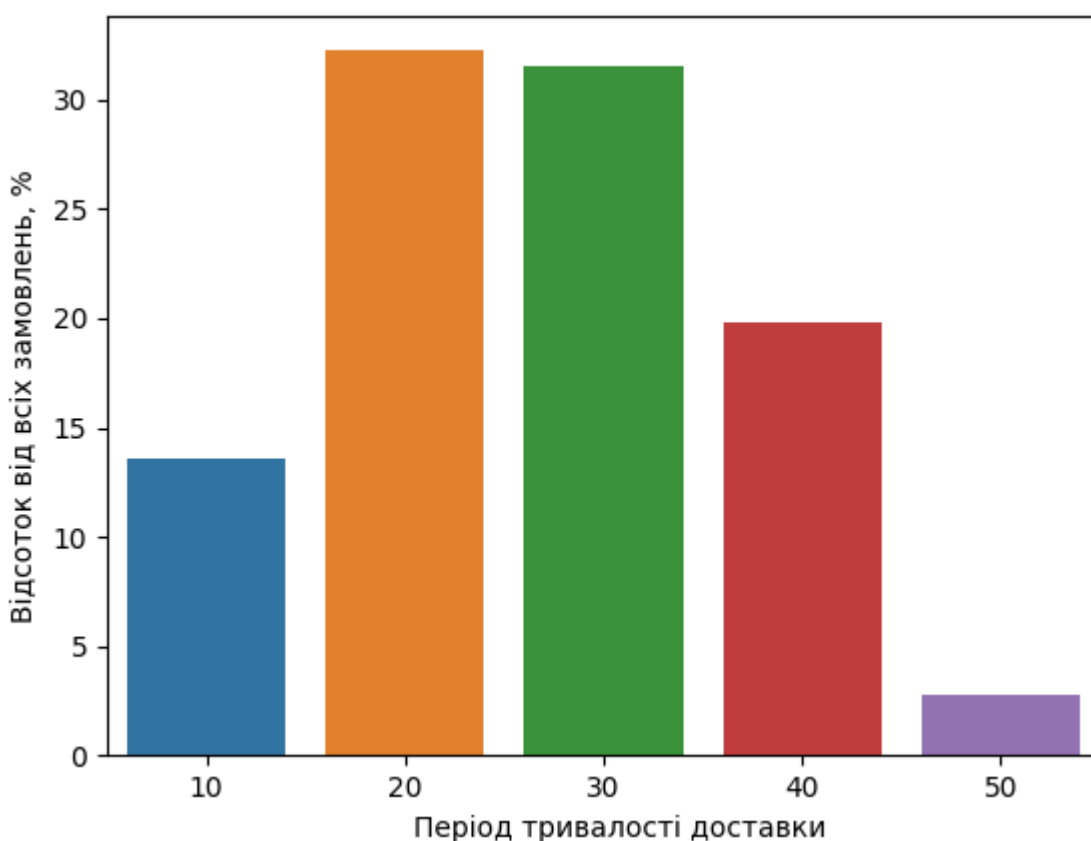


Рисунок 5.10 – Відсотковий розподіл інтервальної тривалості доставки для замовлень за наявності заторів

У підсумку під час дослідження використовувались класи тривалості доставки: до 30 хвилин і більше 30 хвилин. Фактично всі замовлення, які було доставлено за час менше 30 хвилин, слід вважати повністю успішно виконаними. Замовлення, які доставлено за час більше ніж 30 хвилин, – проблемні замовлення.

За результатами дослідження ступеня кореляції між розподілом за класами тривалості доставки замовлень і різними факторами на даній вибірці було виділено наступні фактори, які використовувались для класифікації замовлень:

- розташування закладу харчування (фактично було використано поштовий індекс, за яким розташовано заклад харчування: це впливає зокрема на те, на скільки швидко зможе запаркуватися кур'єр, на скільки проблемно пересуватися взагалі в даному окрузі);

- відстань між закладом харчування і місцем доставки: обчислюється за GPS-координатами;
- розташування місця доставки (фактично було використано поштовий індекс, за яким розташовано заклад харчування);
- рівень трафіку: обчислюється у відповідності з даними Google Maps, які дозволяють встановити рівень трафіку для пересування за маршрутом;
- стан і тип транспортного засобу: кодується відповідним значенням, що враховує стан і тип;
- наявність множинних замовлень;
- заклад харчування, який обслуговує замовлення.

Під час експериментального дослідження, використовуючи наявну вибірку, відбувалось визначення класу успішності за тривалістю доставки. Для цього використовувався представлений у роботі метод та метод CART.

Для кожного методу оцінювалась похибка класифікації, яка визначалася часткою замовлень, які були віднесені до невірної класу в підсумку. Результати продемонстрували перевагу представленого у роботі методу (табл. 5.1).

Таблиця 5.1 – Результати експериментального дослідження

Метод	Похибка
CART	0,28
Метод, представлений у роботі	0,23

5.3 Висновки за розділом 5

У п'ятому розділі описано розроблені інтерфейси програмної системи за результатами проведеного тестування програми та виконано експериментальне дослідження з використанням представленого в роботі методу, що підтвердив ефективність власного застосування на основі вибірки даних.

ВИСНОВКИ

За підсумками виконання роботи розроблено програмну систему підтримки роботи сервісу доставки їжі для забезпечення задоволеності замовника шляхом керування доставкою замовлення задля досягнення вчасної доставки.

Вивчено основні поняття, напрямки розвитку сервісів доставки їжі, виконано огляд і аналіз програмних аналогів, серед яких для більш детального розгляду було виділено Glovo і DoorDash. Виконано дослідження методів підтримки роботи сервісу доставки їжі, для цього було виконано постановку задачі керування доставкою їжі, порівняно існуючі алгоритми навчання дерев рішень, які можуть використовуватися для вирішення даної задачі, та розв'язано дану задачі за допомогою запропонованого методу керування доставкою їжі. Наукова новизна роботи виражена в методі керування доставкою їжі, який було запропоновано в роботі і який характеризується виконанням класифікації замовлень за тривалістю доставки за допомогою побудови дерева рішення на основі алгоритму FSCART, що дозволяє обирати такий керуючий вплив на реалізацію доставки замовлення, щоб воно з більшою вірогідністю завершувалося успішно.

Реалізовано програмну систему підтримки роботи сервісу доставки їжі як вебзастосунок, що надає інтерфейси для клієнтів, представників мереж закладів харчування, окремих закладів харчування та сервісу доставки їжі загалом. Для реалізації використано мову програмування Python, пакет scikit-learn, фреймворк створення вебзастосунків Django, систему керування базами даних MySQL. У реалізовану програмну систему імплементовано представлений в роботі метод керування замовленням доставки їжі, який підтримує роботу представників сервісу доставки щодо досягнення задоволеності замовника.

Увесь набір завдань, які необхідно було забезпечити для реалізації в роботі, було виконано.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Ordering in [Electronic resource] : The rapid evolution of food delivery / K. Ahuja, V. Chandra, V. Lord, C. Peens. – Access mode : <https://www.mckinsey.com/industries/technology-media-and-telecommunications/our-insights/ordering-in-the-rapid-evolution-of-food-delivery>.
2. Food Delivery Services Market To Grow At Rate Of 11.4% Through 2028 [Electronic resource]. – Access mode : https://www.einnews.com/pr_news/585013933/food-delivery-services-market-to-grow-at-rate-of-11-4-through-2028.
3. Global Online Food Delivery Services Market Report 2021 [Electronic resource] : Market is Expected to Reach \$192.16 Billion in 2025, from \$126.91 Billion in 2021 – Long-term Forecast to 2030. – Access mode : <https://www.prnewswire.com/news-releases/global-online-food-delivery-services-market-report-2021-market-is-expected-to-reach-192-16-billion-in-2025--from-126-91-billion-in-2021---long-term-forecast-to-2030--301285677.html>.
4. Checklist for launching a successful delivery business [Electronic resource]. – Access mode : <https://www.deliverect.com/en/blog/online-food-delivery/checklist-for-launching-a-successful-delivery-business>.
5. 7 Hot Trends That Will Change the Future of the Food Delivery Industry [Electronic resource]. – Access mode : <https://rubygarage.org/blog/food-delivery-trends-2019>.
6. Доставка Glovo в місті Запоріжжя. Замовляйте онлайн їжу, продукти, парафармацевтику та багато чого іншого [Електрон. ресурс]. – Режим доступу : <https://glovoapp.com/ua/uk/zaporizhzhya/>.
7. DoorDash Food Delivery – Delivering Now, From Restaurants Near You [Electronic resource]. – Access mode : <https://www.doordash.com/>.
8. Chauhan, N. S. Decision Tree Algorithm, Explained [Electronic resource] / N. S. Chauhan. – Access mode : <https://www.kdnuggets.com/2020/01/decision-tree-algorithm-explained.html>.

9. A Step by Step CART Decision Tree Example [Electronic resource]. – Access mode : <https://sefiks.com/2018/08/27/a-step-by-step-cart-decision-tree-example/>.
10. Brownlee, J. Classification And Regression Trees for Machine Learning [Electronic resource] / J. Brownlee. – Access mode : <https://machinelearningmastery.com/classification-and-regression-trees-for-machine-learning/>.
11. Yildiz, O. T. Comparing Univariate and Multivariate Decision Trees [Electronic resource] / Olcay Taner Yildiz, Ethem Alpaydin. – Access mode : https://web.njit.edu/~usman/courses/cs675_fall15/Comparing_Univariate_and_Multivariate_De.pdf.
12. Edukate Free Website Template [Electronic resource]. – Access mode : <https://www.free-css.com/free-css-templates/page282/edukate>.
13. Food Delivery Dataset. Kaggle [Electronic resource]. – Access mode : <https://www.kaggle.com/datasets/gauravmalik26/food-delivery-dataset>.

ДОДАТОК А
Технічне завдання

Вступ

Сервіси доставки їжі знаходяться в форватері сучасних тенденцій щодо розвитку бізнесу, надаючи можливість закладам харчування розширити доступ до клієнтів до власної продукції, а клієнтам відповідно розширити споживання продукції закладів харчування на випадки, коли вони не мають часу, можливості, бажання на приготування їжі власноруч і одночасно на похід до закладу харчування.

Для будь-якого сервісу доставки їжі критичною є підтримка з використанням відповідного програмного забезпечення. Якщо таке програмне забезпечення буде здатне автоматично аналізувати наявні дані і допомагати менеджерам сервісу під час керування замовленнями щодо визначення орієнтованого часу доставки в конкретних умовах, то це буде мати великий позитивний вплив на задоволеність клієнтами сервісом доставки їжі.

A.1 Підстави для розробки

Розробка програмного забезпечення і дипломної кваліфікаційної роботи магістра загалом виконується на основі завдання за темою «Дослідження та програмна реалізація методів підтримки роботи сервісу доставки їжі», затвердженою наказом № 363 від 4 листопада 2022 р. за Національним університетом «Запорізька політехніка».

A.2 Призначення розробки

Розроблювана програма загалом призначена для підтримки роботи сервісу доставки їжі. Для цього вона повинна надавати інтерфейси для роботи різних учасників даного процесу:

- адміністратора системи, який може виконувати керування всіма даними і зокрема забезпечувати підтримку роботи з базовими каталогами;
- менеджерів сервісу доставки їжі;
- кур'єрів;
- адміністратора і менеджерів мережі закладів харчування;
- адміністратора і менеджерів закладів харчування;
- клієнтів.

А.3 Основні вимоги до програми

А.3.1 Вимоги до функціональних характеристик

Функціональні характеристики, якими має володіти програма, що буде розроблена в підсумку, включають:

- створення і ведення каталогів страв і добавок для мережі закладів;
- визначення особливостей страв і добавок для закладу харчування;
- створення і ведення загального каталогу добавок до страв;
- створення і ведення загального каталогу категорій страв;
- визначення вартості варіантів доставки сервісом;
- замовлення страв у закладах харчування;
- керування замовленнями;
- скасування страви або добавки з замовлення;
- вподобання або невподобання страви;
- оцінювання закладу харчування;
- перегляд відгуків про заклад харчування;
- перегляд відгуків про страву;
- класифікація замовлення за часом доставки;
- визначення переліку замовлень для доставки кур'єром;
- визначення переліку замовлень для опрацювання закладом харчування;
- тимчасове зняття страви або добавки з можливості доставки;

- звітування щодо продажу страв закладом;
- звітування щодо задоволеності клієнтів закладом і стравами;
- пошук закладів харчування;
- відображення доступного меню для замовлення у закладі харчування;
- авторизація користувачів;
- перегляд власних замовлень;
- реєстрація мереж закладів харчування і їх представників;
- реєстрація закладів харчування у мережі і їх представників;
- реєстрація кур'єрів;
- перегляд власних відгуків.

А.3.2 Вимоги до складу та параметрів технічних засобів

Для забезпечення доступу різних користувачів за допомогою відповідних інтерфейсів не висувається додаткових вимог: вимоги відповідають вимогам до веббраузерів (стандартно останніх версій веббраузерів Chrome, Mozilla Firefox тощо).

У свою чергу для роботи вебсервера визначено більш широкі вимоги, що включають наступні. Процесор з мінімальними параметрами: 4 ядра, частота від 2 ГГц. Оперативна пам'ять з мінімальним обсягом від 4 Гб для можливості доступу декількох закладів і підтримки мінімально 10 запитів одночасно до вебсервера. Жорсткий диск, на якому має бути доступно хоча би 2 Гб простору.

А.3.3 Умови експлуатації

Для експлуатації програми потрібно розгорнути вебсервер, до складу якого мають входити сучасний вебсервер, роль якого може виконувати,

наприклад, Apache, система керування базами даних MySQL, яка використовується як базова.

A.3.4 Вимоги до програмної документації

Базова програмна документація включає саме технічне завдання, текст програми, що є додатками до пояснювальної записки.

A.4 Порядок контролю та приймання

Основний порядок контролю та приймання визначається календарним планом, який входить до складу завдання на дипломну кваліфікаційну роботу магістра.

ДОДАТОК Б
Текст програми

Б.1 Текст файла `views_establishments.py`

```

from restdel.models import FoodEstablishment,
EstablishmentDepartment, Dish, DishCategory, DishType,
NetworkEmployee, Employee, Region, EstablishmentReview, Client,
OrderDish
    from django.db.models import Q
    from django.urls import reverse_lazy
    from django.views import View
    from django.shortcuts import render, redirect, get_object_or_404
    from django.views.generic import DetailView, ListView
    import datetime
    from django.contrib.auth.mixins import LoginRequiredMixin
    from restdel.forms import EDCreateForm, SearchForm,
FECreateForm, EReviewCreateForm, DCreateForm
    from geopy.geocoders import Nominatim
    import geocoder
    from django.http import HttpResponseRedirect
    from django.urls import reverse

class FoodEstablishmentCreateView(LoginRequiredMixin, View):
    template_name = 'restdel/foodest_form.html'

    def get(self, request, pk=None):
        fef = FECreateForm()
        formcontext = {'form': fef }
        return render(request, self.template_name, formcontext)

    def post(self, request, pk=None):
        fef = FECreateForm (request.POST, request.FILES or None)
        if not fef.is_valid():
            formcontext = {'form': fef }
            return render(request, self.template_name,
formcontext)
        fe = fef.save(commit=False)
        fe.save()
        return redirect(reverse_lazy('restdel:fes'))

class FoodEstablishmentUpdateView (LoginRequiredMixin, View):
    template_name = 'restdel/estdepartment_form.html'

    def get(self, request, pk):
        fe = get_object_or_404(FoodEstablishment, id=pk)
        fef = FECreateForm (instance=fe)

```

```

        formcontext = {'form': fef }
        return render(request, self.template_name, formcontext)

    def post(self, request, pk=None):
        fe = get_object_or_404(FoodEstablishment, id=pk)
        fef = FECreateForm (request.POST, request.FILES or None,
instance=fe)
        if not fef.is_valid():
            formcontext = {'form': fef }
            return render(request, self.template_name,
formcontext)
        fe = fef.save(commit=False)
        fe.save()
        return redirect(reverse_lazy('restdel:fes'))

class FoodEstablishmentView(DetailView):
    model = FoodEstablishment
    template_name = "restdel/fe.html"

    def get(self, request, pk) :
        fe = FoodEstablishment.objects.get(id=pk, availability =
True)
        establishments =
EstablishmentDepartment.objects.filter(foodestablishment=fe,
availability=True)
        regioned_establishments = []
        regions = []
        for est in establishments:
            if est.region not in regions:
                regions.append(est.region)
                region_all = {}
                region_all['info'] = est.region
                region_all['establishments'] =
establishments.filter(region=est.region).values()
                regioned_establishments.append(region_all)
        full_context = { 'food_establishment' : fe,
                        'establishments':
regioned_establishments}
        return render(request, self.template_name, full_context)

class EstablishmentDepartmentMenuView(DetailView):
    model = EstablishmentDepartment
    template_name = "restdel/ed_menu.html"

    def get(self, request, pk) :

```

```

        ed = EstablishmentDepartment.objects.get(id=pk,
availability = True)
        ct=datetime.datetime.now()
        curr_hour = ct.hour
        curr_minute = ct.minute
        curr_time=datetime.time(hour=curr_hour,
minute=curr_minute)
        paid = False
        if ed.jobstart >= curr_time and ed.jobend <= curr_time:
            paid = True
        dishes = Dish.objects.filter(establishmentdepartment=ed,
availability=True, dishtype__availability=True,
starttime__lte=curr_time, endtime__gte=curr_time)
        dishtypes = DishType.objects.filter(dishes__in=dishes)
        categories =
DishCategory.objects.filter(dishtypes__in=dishtypes)
        reviews =
EstablishmentReview.objects.filter(establishmentdepartment =
ed).order_by('-created')[:5]
        categorized_dishes = {}
        for categ in categories:
            cid = categ.id
            categorized_dishes[cid] =
dishes.filter(dishtype__dishcategory=categ).values()
            full_context = { 'estab_department' : ed,
                            'dish_categories': categories,
                            'dishes': categorized_dishes,
                            'reviews': reviews,
                            'paid': True }
        return render(request, self.template_name, full_context)

class EstablishmentDepartmentReportView(DetailView):
    model = EstablishmentDepartment
    template_name = "restdel/ed_report.html"

    def get(self, request, pk) :
        ed = EstablishmentDepartment.objects.get(id=pk)
        dishes = Dish.objects.filter(establishmentdepartment =
ed)

        ct=datetime.datetime.now()
        curr_year = ct.year
        curr_month = ct.month
        if curr_month == 1:
            curr_year = curr_year - 1

```

```

        curr_time_start=datetime.time(year=curr_year, month=1,
day=1, hour=0, minute=0)
        ordered_dishes =
OrderDish.objects.filter(dish__in=dishes, order__delivered__gte =
curr_time_start, order__delivered__lte=ct)
        last_year_dishes = []
        for curr_dish in dishes:
            ordered_last_year =
ordered_dishes.filter(dish=curr_dish)
            if ordered_last_year:
                dish = {}
                dish['info'] = curr_dish
                dish_month = []
                for one_month in range(1, curr_month+1):
                    month_datetime_start =
datetime.time(year=curr_year, month=one_month, day=1, hour=0,
minute=0)
                    month_datetime_end =
datetime.time(year=curr_year, month=one_month+1, day=1, hour=0,
minute=0)
                    ordered_last_year_month =
ordered_last_year.filter(order__delivered__gte=month_datetime_start,
order__delivered__lt=month_datetime_end)
                    order_counts =
ordered_last_year_month.count()
                    order_price = 0
                    order_number = 0
                    for order in ordered_last_year_month:
                        order_price = order_price + order.price
* order.number
                        order_number = order_number +
order.number
                    dish_month_stat = {}
                    dish_month_stat['revenue'] = order_price
                    dish_month_stat['number'] = order_number
                    dish_month_stat['price'] = order_price /
order_number
                    dish_month.append(dish_month_stat)
                    dish['month_stat'] = dish_month
                    last_year_dishes.append(dish)
                full_context = { 'estab_department' : ed,
                                'dishes': last_year_dishes }
            return render(request, self.template_name, full_context)

class EstablishmentDepartmentYearReportView(DetailView):

```

```

model = EstablishmentDepartment
template_name = "restdel/ed_year_report.html"
def get(self, request, pk) :
    ed = EstablishmentDepartment.objects.get(id=pk)
    dishes = Dish.objects.filter(establishmentdepartment =
ed)

    ct=datetime.datetime.now()
    curr_year = ct.year
    start_year = 2022
    all_dishes = []
    for curr_dish in dishes:
        ordered_last_year =
OrderDish.objects.filter(dish=curr_dish)
        dish = {}
        dish['info'] = curr_dish
        dish_year = []
        for one_year in range(start_year, curr_year+1):
            year_datetime_start =
datetime.time(year=one_year, month=1, day=1, hour=0, minute=0)
            year_datetime_end =
datetime.time(year=one_year+1, month=1, day=1, hour=0, minute=0)
            ordered_year =
OrderDish.objects.filter(dish=curr_dish,
order__delivered__gte=year_datetime_start,
order__delivered__lt=year_datetime_end)
            order_counts = ordered_year.count()
            order_price = 0
            order_number = 0
            for order in ordered_year:
                order_price = order_price + order.price *
order.number

                order_number = order_number + order.number
            dish_year_stat = {}
            dish_year_stat['revenue'] = order_price
            dish_year_stat['number'] = order_number
            dish_year_stat['price'] = order_price /
order_number

            dish_year.append(dish_year_stat)
        dish['year_stat'] = dish_year
        all_dishes.append(dish)
    full_context = { 'estab_department' : ed,
                    'dishes': all_dishes }
    return render(request, self.template_name, full_context)

```

```

class EstablishmentDepartmentCreateView(LoginRequiredMixin,
View):
    template_name = 'restdel/estdepartment_form.html'
    def get(self, request, pk=None):
        edf = EDCreateForm()
        formctx = {'form': edf }
        fempl = get_object_or_404(NetworkEmployee,
user=self.request.user)
        if not fempl:
            return render(request, '403.html', formctx,
status=403)
        return render(request, self.template_name, formctx)

    def post(self, request, pk=None):
        edf = EDCreateForm(request.POST, request.FILES or None)
        if not edf.is_valid():
            formctx = {'form': edf }
            return render(request, self.template_name,
formctx)

        ed = edf.save(commit=False)
        nomloc = Nominatim(user_agent='restdel')
        location = nomloc.geocode(ed.address)
        reg = get_object_or_404(Region,
name=location.raw['address']['state'])
        ed.region = reg
        ed.address_x = location.latitude
        ed.address_y = location.longitude
        ed.stars = 0
        ed.votes = 0
        fempl = get_object_or_404(NetworkEmployee,
user=self.request.user)
        ed.foodestablishment = fempl.foodestablishment
        ed.save()
        fe = ed.foodestablishment
        return redirect(reverse('restdel:fe', args=[fe.id]))

class EstablishmentDepartmentUpdateView(LoginRequiredMixin,
View):
    template_name = 'restdel/estdepartment_form.html'

    def get(self, request, pk):
        ed = get_object_or_404(EstablishmentDepartment, id=pk)
        empl = get_object_or_404(Employee,
user=self.request.user)
        fempl = None

```

```

        res = False
        if empl:
            res = empl.establishmentdepartment.id == pk and
empl.job == 'admin'
        else:
            fempl = get_object_or_404(NetworkEmployee,
user=self.request.user)
            res = fempl.foodestablishment ==
ed.foodestablishment and fempl.job == 'admin'
        if res:
            edf = EDCreateForm(instance=ed)
            formctx = {'form': edf }
            return render(request, self.template_name,
formctx)
        else:
            formctx = {'form': edf }
            return render(request, '403.html', formctx,
status=403)

    def post(self, request, pk=None):
        ed = get_object_or_404(EstablishmentDepartment, id=pk)
        edf = EDCreateForm(request.POST, request.FILES or None,
instance=ed)
        if not edf.is_valid():
            formctx = {'form': edf }
            return render(request, self.template_name,
formctx)

        ed = edf.save(commit=False)
        nomloc = Nominatim(user_agent='restdel')
        location = nomloc.geocode(ed.address)
        ed.address_x = location.latitude
        ed.address_y = location.longitude
        reg = get_object_or_404(Region,
name=location.raw['address']['state'])
        ed.region = reg
        ed.save()
        return redirect(reverse('restdel:fe', args=[fe.id]))

class EstablishmentDepartmentSearchView(ListView):
    model = EstablishmentDepartment
    template_name = "restdel/search.html"

    def get_queryset(self, *args, **kwargs):
        eds = super().get_queryset(*args, **kwargs)
        eds = eds.filter(availability = True)

```

```

address = str(self.request.GET.get('address'))
if address:
    nomloc = Nominatim(user_agent='restdel')
    location = nomloc.geocode(address)
    reg = get_object_or_404(Region,
name=location.raw['address']['state'])
    ed.region = reg
    ed.address_x = location.latitude
    ed.address_y = location.longitude
    eds = eds.filter(region = reg)
else:
    g = geocoder.ip('me')
    gps = g.latlng
    nomloc = Nominatim(user_agent='restdel')
    location = nomloc.reverse(gps[0]+", "+gps[1])
    reg = get_object_or_404(Region,
name=location.raw['address']['state'])
    eds = eds.filter(region = reg)
    depart_title =
str(self.request.GET.get('department_title'))
    depart_title = depart_title.strip()
    network_title =
str(self.request.GET.get('network_title'))
    network_title = network_title.strip()
    if depart_title:
        eds = eds.filter(title__icontains = depart_title)
    if network_title:
        eds = eds.filter(foodestablishment__title__icontains
= network_title)
    stars_min = float(self.request.GET.get('stars1'))
    if stars_min:
        eds = eds.filter(stars__gte = stars_min)
    stars_max = float(self.request.GET.get('stars2'))
    if stars_max:
        eds = eds.filter(stars__lte = stars_max)
    category_id = float(self.request.GET.get('category'))
    dish_category = None
    if category_id:
        dish_category = DishCategory.objects.get(pk =
category_id)
    curr_hour = int(self.request.GET.get('time1'))
    curr_minute = int(self.request.GET.get('time2'))
    if curr_hour < 0 and curr_minute < 0:
        ct=datetime.datetime.now()
        curr_hour = ct.hour

```

```

        curr_minute = ct.minute
        curr_time=datetime.time(hour=curr_hour,
minute=curr_minute)
        price = float(self.request.GET.get('price'))
        left_eds = []
        for ed in eds:
            ed_x, ed_y, gps_x, gps_y = map(radians,
[ed.address_x, ed.address_y, gps[0], gps[1]])
            diff_x = gps_x - ed_x
            diff_y = gps_y - ed_y
            q = sin(diff_y/2)**2 + cos(ed_y) * cos(gps_y) *
sin(diff_x/2)**2
            v = 2 * asin(sqrt(q))
            dishes = Dish.objects.filter(establishmentdepartment
= ed, availability = True, starttime__lte=curr_time,
endtime__gte=curr_time)
            if dish_category:
                dishes = dishes.filter(dishtype__dishcategory =
dish_category)
            if price > 0:
                dishes = dishes.filter(Q(price__lte=price) &
Q(price__gt=0) | Q(dishtype__price__lte=price) &
Q(dishtype__price__gt=0)).count()
            if v * 6371 < reg.distance and dishes > 0:
                left_eds.append(ed)
        return left_eds

    def get_context_data(self, **kwargs):
        context = super(EstablishmentDepartmentSearchView,
self).get_context_data(**kwargs)
        context['search_form'] = SearchForm(initial={
            'department_title':
self.request.GET.get('department_title', ''),
            'address': self.request.GET.get('address', ''),
            'network_title':
self.request.GET.get('network_title', ''),
            'price': self.request.GET.get('price', '0'),
            'time1': self.request.GET.get('time1', '-1'),
            'time2': self.request.GET.get('time2', '-1'),
            'stars2': self.request.GET.get('stars2', '5'),
            'stars1': self.request.GET.get('stars1', '0')
        })
        return context

```

```

class EstablishmentReviewCreateView(LoginRequiredMixin, View):

```

```

template_name = 'restdel/est_review_form.html'
def get(self, request, pk=None):
    erf = EReviewCreateForm()
    formctx = {'form': erf }
    return render(request, self.template_name, formctx)

def post(self, request, pk=None):
    erf = EReviewCreateForm ()
    if not erf.is_valid():
        formctx = {'form': erf }
        return render(request, self.template_name,
formctx)

    er = erf.save(commit=False)
    er.client = Client.objects.get(user=self.request.user)
    er.save()
    ed = er.establishmentdepartment
    ers =
EstablishmentReview.objects.filter(establishmentdepartment=ed)
    score = 0
    for er in ers:
        score = score + er.stars
    ed.votes = ers.count()
    ed.stars = score / ed.votes
    ed.save()
    return redirect(reverse('restdel:fe',
args=[er.establishmentdepartment]))

class EstablishmentReviewsView(ListView):
    model = EstablishmentReview
    template_name = 'restdel/ed_reviews.html'

    def get_queryset(self, *args, **kwargs):
        ers = super().get_queryset(*args, **kwargs)
        department = int(self.request.GET.get('department'))
        ers = ers.filter(establishmentdepartment__id =
department)

        ers = ers.order_by('-created')
        return ers.values()

class ClientEstablishmentReviewsView(ListView):
    model = EstablishmentReview
    template_name = 'restdel/client_reviews.html'

    def get_queryset(self, *args, **kwargs):
        ers = super().get_queryset(*args, **kwargs)

```

```

        iclient = Client.objects.get(user=self.request.user)
        ers = ers.filter(client = iclient)
        ers = ers.order_by('-created')
        return ers.values()

class EstablishmentDepartmentReviewsReportView(ListView):
    model = EstablishmentReview
    template_name = "restdel/ed_reviews_report.html"

    def get_queryset(self, *args, **kwargs):
        ers = super().get_queryset(*args, **kwargs)
        department = self.request.GET.get('department')
        if not department:
            empl = get_object_or_404(Employee,
user=self.request.user)
            department = empl.establishmentdepartment.id
        else:
            department = int(department)
        ers = ers.filter(establishmentdepartment__id =
department)
        level = int(self.request.GET.get('level'))
        last = self.request.GET.get('last')
        if last:
            curr_date = datetime.datetime.now()
            start = curr_date - datetime.timedelta(days=28)
            ers = ers.filter(created__gte=start)
        if level and level2:
            ers = ers.filter(stars__gte=level,
stars__lte=level2)
        elif level:
            ers = ers.filter(stars=level)
        return ers.order_by('-created')

    def get_context_data(self, **kwargs):
        context = super(EstablishmentDepartmentSearchView,
self).get_context_data(**kwargs)
        context['search_form'] = SearchForm(initial={
            'level': self.request.GET.get('level', '0'),
            'level2': self.request.GET.get('level2', '0'),
            'last': self.request.GET.get('last', '0'),
            'department': self.request.GET.get('department', '-
1')
        })
        return context

```

```

class EstablishmentDepartmentReviewReportView(DetailView):
    model = EstablishmentDepartment
    template_name = "restdel/ed_review_report.html"

    def get(self, request, pk) :
        ed = EstablishmentDepartment.objects.get(id=pk)
        reviews =
EstablishmentReview.objects.filter(establishmentdepartment = ed)
        curr_date = datetime.datetime.now()
        start = curr_date - datetime.timedelta(days=28)
        reviews_last = reviews.filter(created__gte=start)
        stat = {}
        stat['number'] = reviews.count()
        stat_last = {}
        stat_last['number'] = reviews_last.count()
        for i in range(1,6):
            reviews_level = reviews.filter(stars=i)
            reviews_last_level = reviews_last.filter(stars=i)
            stat[i] = reviews_level.count()
            stat_last[i] = reviews_last_level.count()
        full_context = { 'estab_department' : ed,
                        'stat': stat,
                        'stat_last': stat_last }
        return render(request, self.template_name, full_context)

def show_ed_picture(request, pk):
    ed = get_object_or_404(EstablishmentDepartment, id=pk)
    httpresp = HttpResponse()
    httpresp ['Content-Type'] = 'image/jpg'
    httpresp ['Content-Length'] = len(ed.picture)
    httpresp.write(ed.picture)
    return httpresp

class FoodEstablishmentDishTypesView(DetailView):
    model = FoodEstablishment
    template_name = "restdel/fe_dishtypes.html"

    def get(self, request, pk) :
        fe = FoodEstablishment.objects.get(id=pk)
        dishtypes =
DishType.objects.filter(foodestablishment=fe)
        full_context = { 'food_establishment' : fe,
                        'dish_types': dishtypes }
        return render(request, self.template_name, full_context)

```

```

class EstablishmentDepartmentDishesView(DetailView):
    model = EstablishmentDepartment
    template_name = "restdel/ed_dishes.html"

    def get(self, request, pk) :
        ed = EstablishmentDepartment.objects.get(id=pk)
        dishes = Dish.objects.filter(establishmentdepartment=ed)
        full_context = { 'establishment' : ed,
                        'dishes': dishes }
        return render(request, self.template_name, full_context)

```

Б.2 Текст файла views_dishes.py

```

from restdel.models import Dish, DishType, NetworkEmployee,
Client, DishReview, DishTypeAddition, Addition, DishAddition
from django.urls import reverse_lazy
from django.views import View
from django.shortcuts import render, redirect, get_object_or_404
from django.views.generic import DetailView, ListView
from django.contrib.auth.mixins import LoginRequiredMixin
from restdel.forms import DTCreateForm, DCreateForm,
DTACreateForm, DACreateForm, AddCreateForm, DishReviewCreateForm
from django.urls import reverse

class DishTypeCreateView(LoginRequiredMixin, View):
    template_name = 'restdel/dishtype_form.html'

    def get(self, request, pk=None):
        dtf = DTCreateForm()
        formcontext = {'form': dtf }
        return render(request, self.template_name, formcontext)

    def post(self, request, pk=None):
        dtf = DTCreateForm (request.POST, request.FILES or None)
        if not dtf.is_valid():
            formcontext = {'form': dtf }
            return render(request, self.template_name,
formcontext)

        dt = dtf.save(commit=False)
        fempl = get_object_or_404(NetworkEmployee,
user=self.request.user)
        dt.foodestablishment = fempl.foodestablishment
        dt.save()

```

```
        return redirect(reverse('restdel:fe_dts', args=[
dt.foodestablishment])))
```

```
class DishTypeUpdateView (LoginRequiredMixin, View):
    template_name = 'restdel/dishtype_form.html'

    def get(self, request, pk):
        dt = get_object_or_404(DishType, id=pk)
        dtf = DTCreateForm (instance=dt)
        formctx = {'form': dtf }
        return render(request, self.template_name, formctx)

    def post(self, request, pk=None):
        dt = get_object_or_404(DishType, id=pk)
        dtf = DTCreateForm (request.POST, None, instance=dt)
        if not dtf.is_valid():
            formctx = {'form': dtf }
            return render(request, self.template_name,
formctx)

        dt = dtf.save(commit=False)
        dt.save()
        return redirect(reverse('restdel:fe_dts', args=[
dt.foodestablishment])))
```

```
class DishCreateView(LoginRequiredMixin, View):
    template_name = 'restdel/dish_form.html'

    def get(self, request, pk=None):
        df = DCreateForm()
        formctx = {'form': df }
        return render(request, self.template_name, formctx)

    def post(self, request, pk=None):
        df = DCreateForm (request.POST, request.FILES or None)
        if not df.is_valid():
            formctx = {'form': df }
            return render(request, self.template_name,
formctx)

        d = df.save(commit=False)
        d.posvotes = 0
        d.negvotes = 0
        d.save()
        return redirect(reverse('restdel:ed_ds', args=[
d.establishmentdepartment])))
```

```

class DishUpdateView (LoginRequiredMixin, View):
    template_name = 'restdel/dish_form.html'

    def get(self, request, pk):
        d = get_object_or_404(Dish, id=pk)
        df = DTCreateForm (instance=d)
        formctx = {'form': df }
        return render(request, self.template_name, formctx)

    def post(self, request, pk=None):
        d = get_object_or_404(Dish, id=pk)
        df = DTCreateForm (request.POST, None, instance=d)
        if not df.is_valid():
            formctx = {'form': df }
            return render(request, self.template_name,
formctx)

        d = df.save(commit=False)
        d.save()
        return redirect(reverse('restdel:ed_ds', args=[
d.establishmentdepartment]))

class DishTypeDishesView(DetailView):
    model = DishType
    template_name = "restdel/dt_dishes.html"

    def get(self, request, pk) :
        dt = DishType.objects.get(id=pk)
        dishes = Dish.objects.filter(dishtype=dt)
        additions = DishTypeAddition.objects.filter(dishtype=dt)
        full_context = { 'dish_type' : dt,
                        'dishes': dishes,
                        'additions': additions }
        return render(request, self.template_name, full_context)

class DishReviewCreateView(LoginRequiredMixin, View):
    template_name = 'restdel/dish_review_form.html'

    def get(self, request, pk=None):
        drf = DishReviewCreateForm()
        formctx = {'form': drf }
        return render(request, self.template_name, formctx)

    def post(self, request, pk=None):
        drf = DishReviewCreateForm()
        if not drf.is_valid():

```

```

        formcontext = {'form': drf }
        return render(request, self.template_name,
formcontext)

    dr = drf.save(commit=False)
    dr.client = Client.objects.get(user=self.request.user)
    dr.save()
    drs = DishReview.objects.filter(dish=dr)
    posvotes = 0
    negvotes = 0
    for dr in drs:
        if dr.vote:
            posvotes = posvotes + 1
        else:
            negvotes = negvotes + 1
    d = Dish.objects.get(id = dr.dish)
    d.posvotes = posvotes
    d.negvotes = negvotes
    d.save()
    return redirect(reverse('restdel:odrs', args=[
dr.client])))

```

```

class DishReviews(ListView):
    model = DishReview
    template_name = "restdel/dish_reviews.html"

    def get_queryset(self, *args, **kwargs):
        drs = super().get_queryset(*args, **kwargs)
        drs = drs.filter(dish__id = self.kwargs['pk'])
        drs = drs.order_by('-created')
        return drs.values()

```

```

class DishReviewReportView(DetailView):
    model = Dish
    template_name = "restdel/dish_review_report.html"

    def get(self, request, pk) :
        d = Dish.objects.get(id=pk)
        reviews = DishReview.objects.filter(dish = d)
        curr_date = datetime.datetime.now()
        start = curr_date - datetime.timedelta(days=28)
        reviews_last = reviews.filter(created__gte=start)
        stat = {}
        stat['number'] = reviews.count()
        stat['last_number'] = reviews_last.count()
        stat['posvotes'] = reviews.filter(vote=True).count()

```

```

        stat['negvotes'] = reviews.filter(vote=False).count()
        stat['last_posvotes'] =
reviews_last.filter(vote=True).count()
        stat['last_negvotes'] =
reviews_last.filter(vote=False).count()
        full_context = { 'dish' : d,
                        'stat': stat }
        return render(request, self.template_name, full_context)

class DishView(DetailView):
    model = Dish
    template_name = "restdel/dish.html"

    def get(self, request, pk) :
        d = Dish.objects.get(id=pk)
        reviews = DishReview.objects.filter(dish=d).order_by('-
created')[0:10]
        additions = DishAddition.objects.filter(dish=d)
        full_context = { 'dish' : d,
                        'reviews': reviews,
                        'additions': additions }
        return render(request, self.template_name, full_context)

class OrderingDishView(DetailView):
    model = Dish
    template_name = "restdel/dish_for_order.html"

    def get(self, request, pk) :
        d = Dish.objects.get(id=pk)
        reviews = DishReview.objects.filter(dish=d).order_by('-
created')[0:10]
        additions = DishAddition.objects.filter(dish=d,
availability = True)
        categories = AdditionCategory.objects.all()
        add_categories = []
        for cat in categories:
            adds =
additions.filter(dishtypeaddition__addition__additioncategory=cat)
            if adds:
                ac = {}
                ac['category'] = cat
                ac['additions'] = adds
                add_categories.append(ac)
        full_context = { 'dish' : d,
                        'reviews': reviews,

```

```

        'additions': add_categories }
    return render(request, self.template_name, full_context)

class DishTypeAdditionCreateView(LoginRequiredMixin, View):
    template_name = 'restdel/dishtype_addition_form.html'

    def get(self, request, pk=None):
        dtaf = DTACreateForm()
        formcontext = {'form': dtaf }
        return render(request, self.template_name, formcontext)

    def post(self, request, pk=None):
        dtaf = DTACreateForm(request.POST, request.FILES or
None)
        if not dtaf.is_valid():
            formcontext = {'form': dtaf }
            return render(request, self.template_name,
formcontext)
        dta = dtaf.save(commit=False)
        dta.save()
        return redirect(reverse('restdel:dt', args=[
dta.dishtype]))

class DishTypeAdditionUpdateView (LoginRequiredMixin, View):
    template_name = 'restdel/dishtype_addition_form.html'

    def get(self, request, pk):
        dta = get_object_or_404(DishTypeAddition, id=pk)
        dtaf = DTACreateForm (instance=dta)
        formcontext = {'form': dtaf }
        return render(request, self.template_name, formcontext)

    def post(self, request, pk=None):
        dta = get_object_or_404(DishTypeAddition, id=pk)
        dtaf = DTACreateForm (request.POST, None, instance=dta)
        if not dtaf.is_valid():
            formcontext = {'form': dtaf }
            return render(request, self.template_name,
formcontext)
        dta = dtaf.save(commit=False)
        dta.save()
        redirect(reverse('restdel:dt', args=[ dta.dishtype]))

class DishAdditionCreateView(LoginRequiredMixin, View):
    template_name = 'restdel/dish_addition_form.html'

```

```

def get(self, request, pk=None):
    daf = DACreateForm()
    formcontext = {'form': daf }
    return render(request, self.template_name, formcontext)

def post(self, request, pk=None):
    daf = DACreateForm (request.POST)
    if not daf.is_valid():
        formcontext = {'form': daf }
        return render(request, self.template_name,
formcontext)

    da = daf.save(commit=False)
    da.save()
    return redirect(reverse('restdel:dish', args=[da.dish]))

class DishAdditionUpdateView (LoginRequiredMixin, View):
    template_name = 'restdel/dish_addition_form.html'

    def get(self, request, pk):
        da = get_object_or_404(DishAddition, id=pk)
        daf = DACreateForm (instance=da)
        formcontext = {'form': daf }
        return render(request, self.template_name, formcontext)

    def post(self, request, pk=None):
        da = get_object_or_404(DishAddition, id=pk)
        daf = DACreateForm (request.POST, None, instance=da)
        if not daf.is_valid():
            formcontext = {'form': daf }
            return render(request, self.template_name,
formcontext)

        da = daf.save(commit=False)
        da.save()
        return redirect(reverse('restdel:dish', args=[da.dish]))

class AdditionCreateView(LoginRequiredMixin, View):
    template_name = 'restdel/addition_form.html'

    def get(self, request, pk=None):
        af = AddCreateForm()
        formcontext = {'form': af }
        return render(request, self.template_name, formcontext)

    def post(self, request, pk=None):
        af = AddCreateForm(request.POST, request.FILES or None)

```

```

        if not af.is_valid():
            formctx = {'form': af }
            return render(request, self.template_name,
formctx)

        a = af.save(commit=False)
        a.save()
        return redirect(reverse_lazy('restdel:adds'))

class AdditionUpdateView (LoginRequiredMixin, View):
    template_name = 'restdel/addition_form.html'

    def get(self, request, pk):
        a = get_object_or_404(Addition, id=pk)
        af = AddCreateForm (instance=a)
        formctx = {'form': af }
        return render(request, self.template_name, formctx)

    def post(self, request, pk=None):
        a = get_object_or_404(Addition, id=pk)
        af = AddCreateForm (request.POST, None, instance=a)
        if not af.is_valid():
            formctx = {'form': af }
            return render(request, self.template_name,
formctx)

        a = af.save(commit=False)
        a.save()
        return redirect(reverse_lazy('restdel:adds'))

```

ДОДАТОК В
Слайди презентації

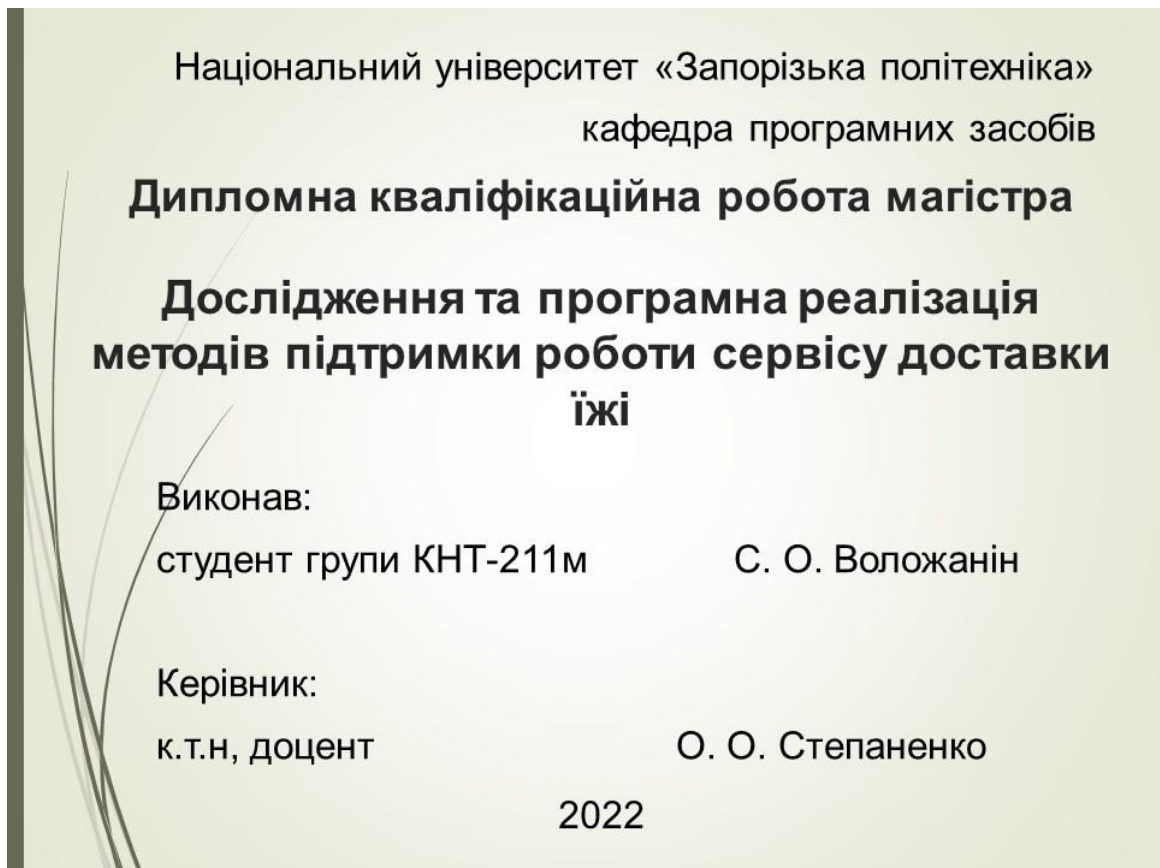


Рисунок В.1 – Титульний слайд

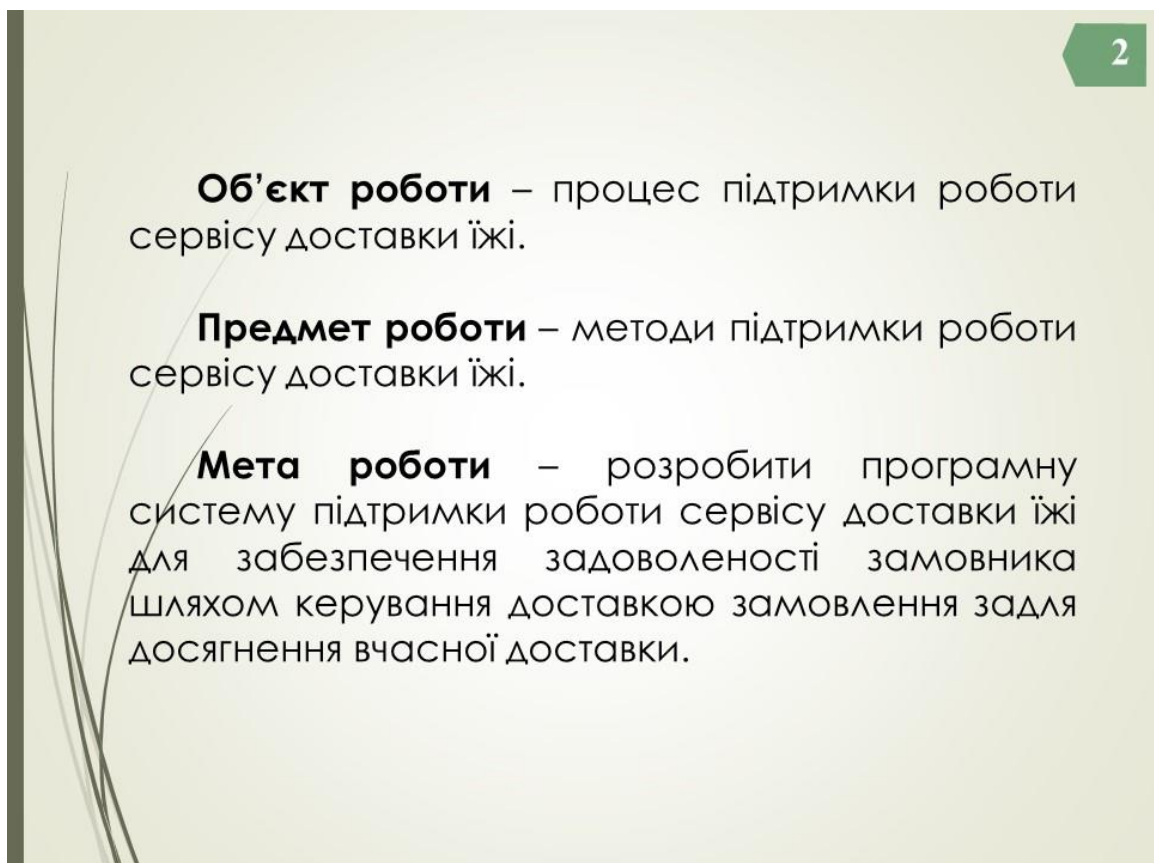


Рисунок В.2 – Об'єкт, предмет, мета роботи

Завдання роботи

3

- проаналізувати принципи роботи сервісів доставки їжі та програмне забезпечення, що забезпечує підтримку їх роботи;
- дослідити методи підтримки роботи сервісу доставки їжі;
- виконати проєктування програмної системи підтримки роботи сервісу доставки їжі;
- реалізувати програмну систему підтримки роботи сервісу доставки їжі, що повинна надавати інтерфейси для клієнтів, представників мереж закладів харчування, окремих закладів харчування та сервісу доставки їжі загалом.

Рисунок В.3 – Завдання роботи

Аналіз програмних аналогів

4

Характеристика	Glovo	DoorDash	Програма-розробка
Пошук за ресторанами	+	+	+
Вибір ресторану за категорією страви	+	+	+
Текстові відгуки клієнтів	–	+	+
Оцінки ресторанів клієнтами	+	+	+
Оцінки страв клієнтами	–	+	+
Визначення спеціальних інструкцій	–	+	+
Визначення варіанта скасування клієнтом	–	+	+
Керування замовленнями службою доставки	+	+	+
Керування замовленнями і інформацією зі сторони закладу харчування	+	+	+
Класифікація часу майбутньої доставки	–	–	+

Рисунок В.4 – Аналіз програмних аналогів

Задача керування доставкою їжі

5

Задача керування доставкою їжі полягає у визначенні таких значень керуючого впливу Y , які разом зі значеннями параметрів замовлення X призводять до отримання потрібного класу тривалості доставки замовлення $z_0 \in Z$ (успішної доставки). При цьому необхідно встановити відповідну функціональну залежність f між класом тривалості доставки замовлення та параметрами замовлення і керуючим впливом, яка б дозволила визначати клас тривалості доставки для відповідного замовлення на основі залежності:

$$f(X, Y) = Z.$$

Значення даної залежності f відомі для деякої вибірки T , що називається тренувальною, і визначається виразом:

$$T = (X^r, Y^r, Z^r),$$

де r – порядковий номер замовлення в тренувальній вибірці T , $r = \overline{1, R}$;

R – кількість замовлень, дані про які представлені в тренувальній вибірці;

X^r – множина значень, що характеризують параметри замовлення з номером r ;

Y^r – множина значень, що характеризують керуючий вплив, заподіяний на замовлення з номером r ;

Z^r – клас тривалості доставки замовлення з номером r .

Рисунок В.5 – Задача керування доставкою їжі

Порівняння алгоритмів дерев рішень

6

Критерій	ID3	CART	FSCART
Класифікація даних	+	+	+
Множина ознак	–	+	+
Вибір ознак	–	–	+

Рисунок В.6 – Порівняння алгоритмів дерев рішень

Метод керування доставкою їжі

7

Етап 1. Визначення набору даних T , що охоплює всі виконані замовлення з доставки їжі, для кожного з яких відомі значення з набору параметрів X , завданого керуючого впливу Y та результуючий клас тривалості доставки Z .

Етап 2. Побудова дерева рішень шляхом навчання на основі вибірки T за допомогою використання алгоритму FSCART з узгодженням множин параметрів $X' \subset X$ та керуючих впливів $Y' \subset Y$, що включають тільки ті параметри і керуючі впливи, які завдають значний вплив на результуючий клас тривалості доставки Z .

Етап 3. Визначення порядку застосування керуючих впливів.

Етап 4. Для кожного нового замовлення визначення і перетворення значення його параметрів X' .

Етап 5. Визначення на основі дерева рішень для всіх допустимих комбінацій керуючих впливів, що входять до підсумкової множини Y'_D , та зафіксованих значень параметрів X' значення відповідних класів тривалості доставки замовлення.

Етап 6. Вибір для замовлення тих комбінацій керуючих впливів, які призвели до отримання класу тривалості доставки, який відповідає успішній доставці. Після цього всі комбінації впорядковуються за правилами, які були встановлені на третьому етапі. Таким чином впорядковані прийнятні рішення надаються менеджеру як результат роботи методу.

Рисунок В.7 – Метод керування доставкою їжі

Порівняння вебфреймворків Laravel і Django

8

Критерій	Laravel	Django
Архітектура	Model-View-Controller	Model-View-Template
Мова програмування	PHP	Python
Поширеність і зручність засобів роботи з даними і деревами рішень	Менша	Більша
Продуктивність роботи	Менша	Значно більша
Безпека роботи	Надає тільки базові засоби	Застосовує найкращі сучасні практики

Рисунок В.8 – Порівняння вебфреймворків Laravel і Django

Діаграми прецедентів

9

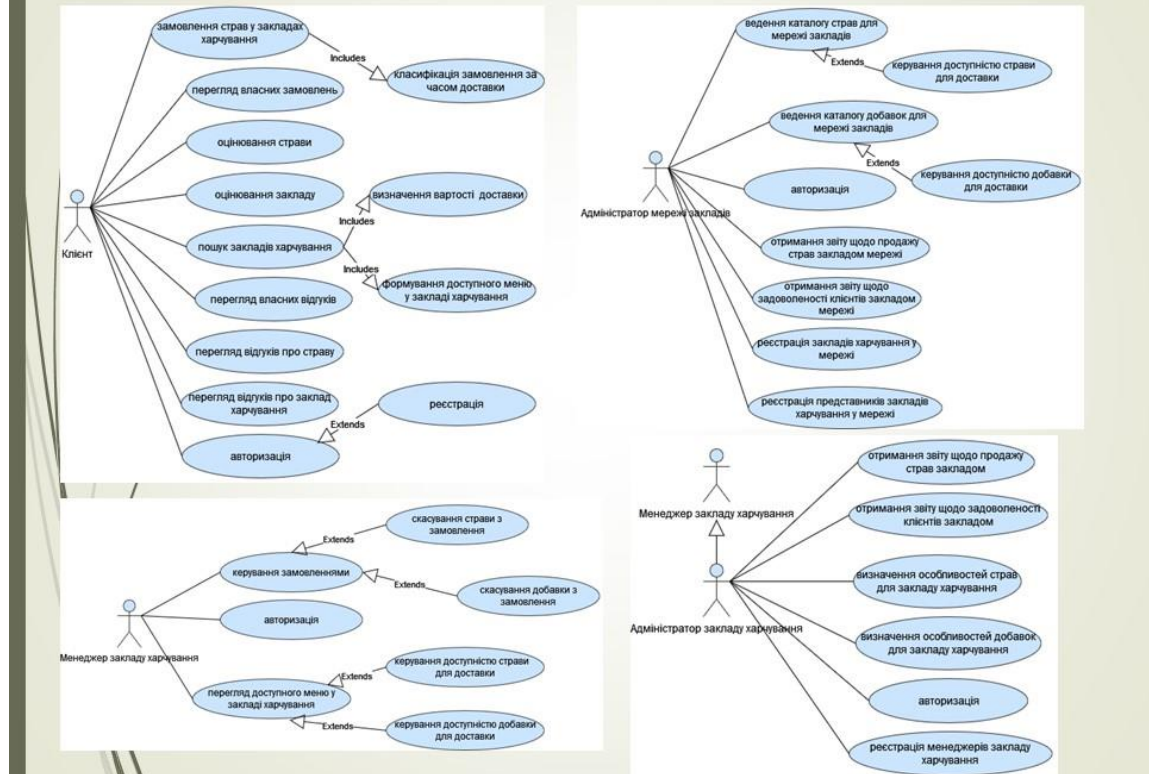


Рисунок В.9 – Діаграми прецедентів

Схема бази даних

10

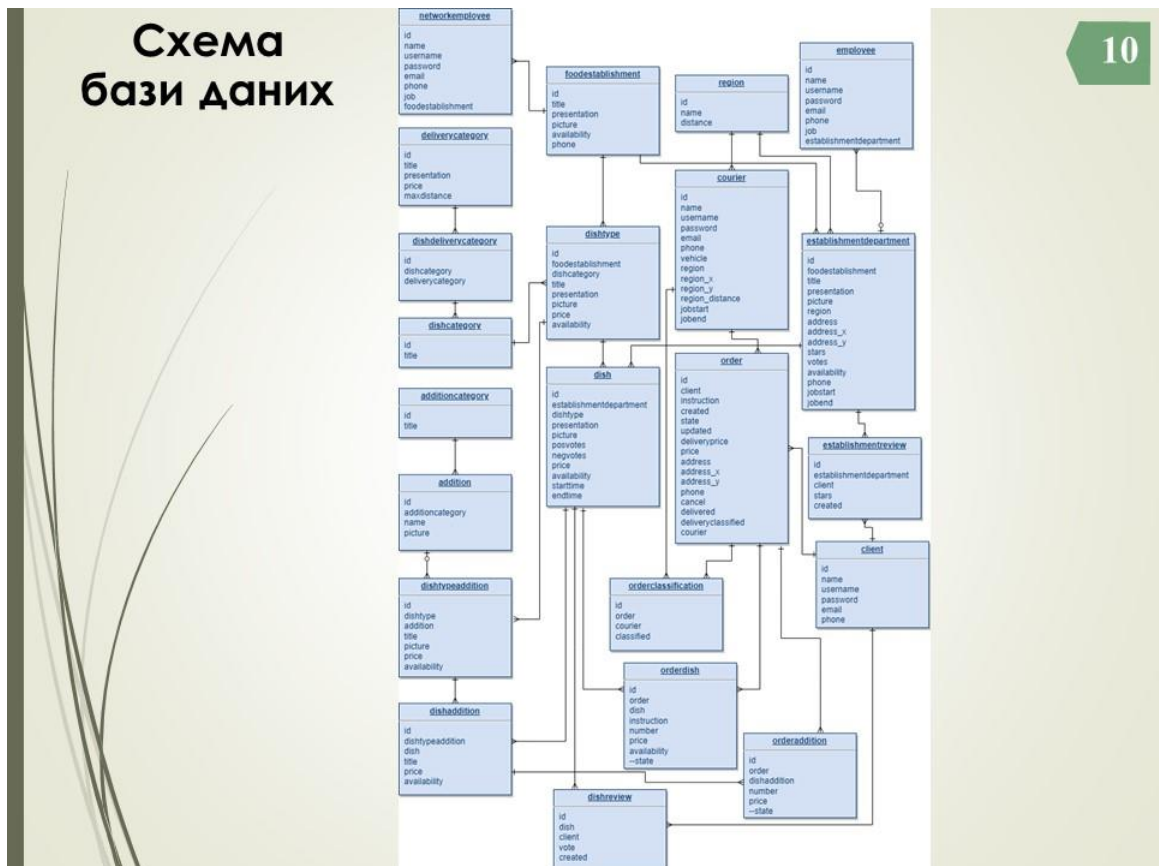


Рисунок В.10 – Схема бази даних

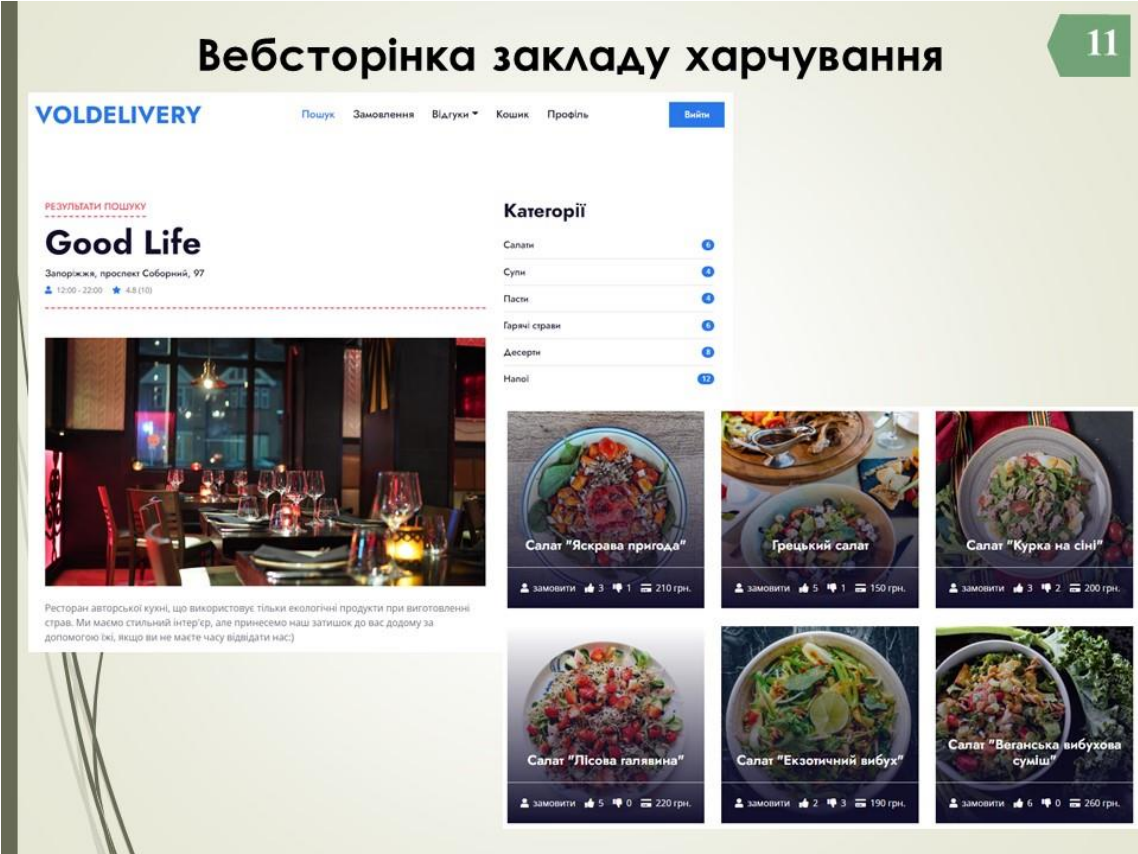


Рисунок В.11 – Вебсторінка закладу харчування

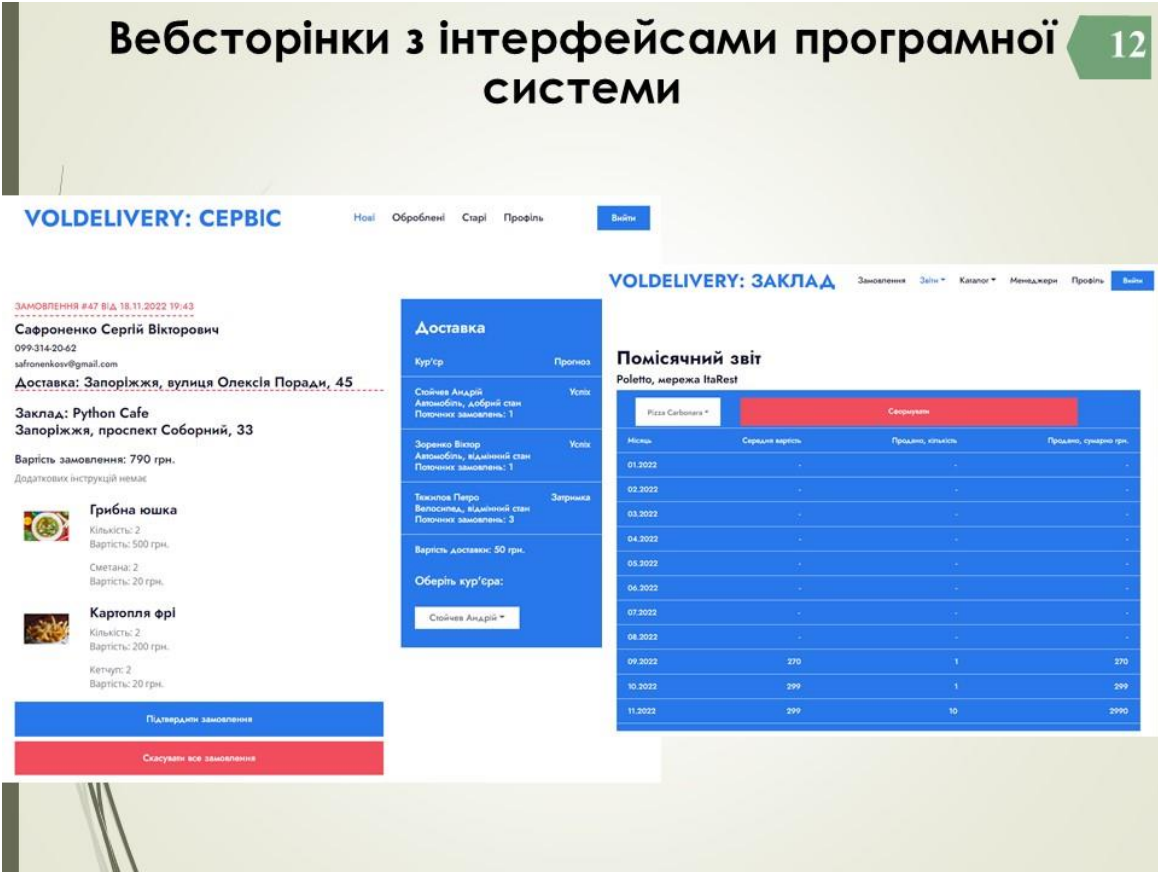


Рисунок В.12 – Вебсторінки з інтерфейсами програмної системи

Експериментальне дослідження

13

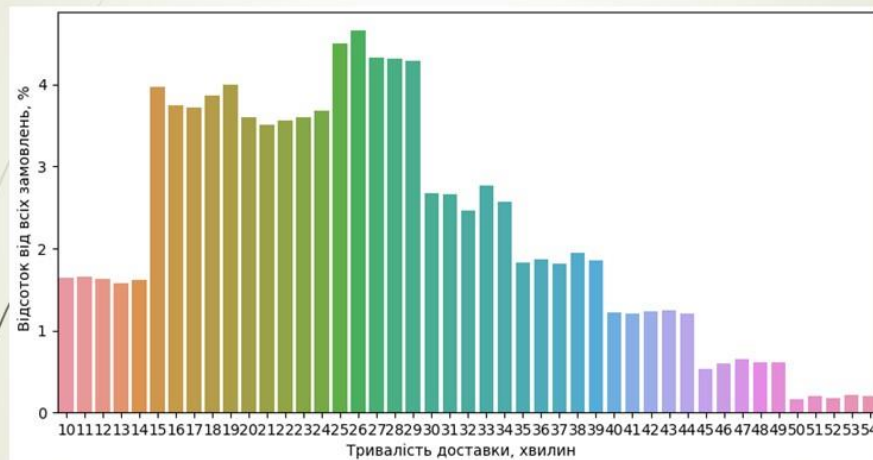


Рисунок 1 – Відсотковий розподіл тривалості доставки

Рисунок В.13 Експериментальне дослідження (загальна вибірка)

Експериментальне дослідження

14

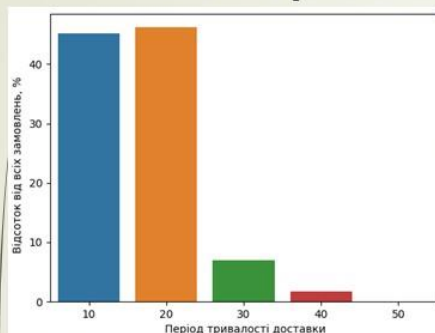


Рисунок 2 – Відсотковий розподіл інтервальної тривалості доставки для замовлень за низького рівня трафіка

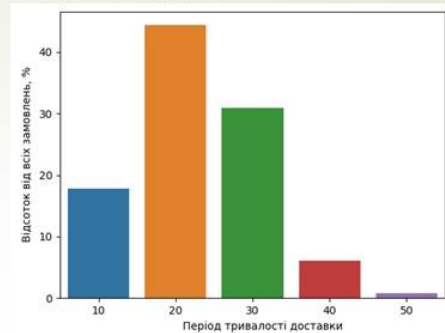


Рисунок 4 – Відсотковий розподіл інтервальної тривалості доставки для замовлень за високого рівня трафіка

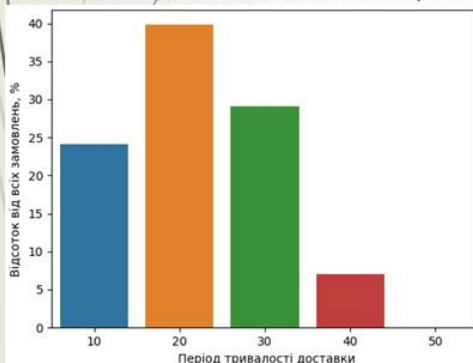


Рисунок 3 – Відсотковий розподіл інтервальної тривалості доставки для замовлень за середнього рівня трафіка

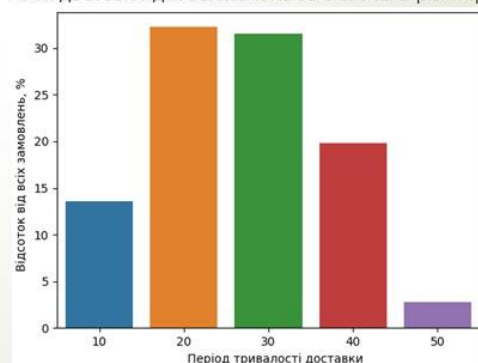


Рисунок 5 – Відсотковий розподіл інтервальної тривалості доставки для замовлень за наявності заторів

Рисунок В.14 – Експертментальне дослідження (аналіз вибірки)

Результати експериментального дослідження

15

Метод	Похибка
CART	0,25
Метод, представлений у роботі	0,19

Рисунок В.15 – Результати експериментального дослідження

Висновки

16

Вивчено основні поняття, напрямки розвитку сервісів доставки їжі, виконано огляд і аналіз програмних аналогів, серед яких для більш детального розгляду було виділено Glovo і DoorDash. Реалізовано програмну систему підтримки роботи сервісу доставки їжі як вебзастосунок. Виконано дослідження методів підтримки роботи сервісу доставки їжі, для цього було виконано постановку задачі керування доставкою їжі, порівняно існуючі алгоритми навчання дерев рішень, які можуть використовуватися для вирішення даної задачі, та розв'язано дану задачі за допомогою запропонованого методу керування доставкою їжі.

Наукова новизна роботи виражена в методі керування доставкою їжі, який було запропоновано в роботі і який характеризується виконанням класифікації замовлень за тривалістю доставки за допомогою побудови дерева рішення на основі алгоритму FSCART, що дозволяє обирати такий керуючий вплив на реалізацію доставки замовлення, щоб воно з більшою вірогідністю завершувалося успішно.

Рисунок В.16 – Висновки