

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ  
Національний університет «Запорізька політехніка»

Інститут інформатики та радіoeлектроніки,  
Факультет комп'ютерних наук і технологій  
(повне найменування інституту, факультету)

Кафедра програмних засобів  
(повне найменування кафедри)

## Пояснювальна записка

до дипломного проєкту (роботи)

магістра

(ступінь вищої освіти)

на тему ДОСЛІДЖЕННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ МЕТОДІВ  
ПРОГНОЗУВАННЯ ЧАСУ ДОСТАВКИ ВИРОБІВ ПЕКАРНІ  
RESEARCH AND SOFTWARE IMPLEMENTATION OF METHODS FOR  
PREDICTING THE BAKERY DELIVERY TIME

Виконав: студент(ка) 2 курсу, групи КНТ-220м  
Спеціальності 122 Комп'ютерні науки  
(код і найменування спеціальності)

Освітня програма (спеціалізація)  
Системи штучного інтелекту

Федорчук Д.О.

(прізвище та ініціали)

Керівник Льовкін В.М.

(прізвище та ініціали)

Рецензент Терещенко Е.В.

(прізвище та ініціали)

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

Інститут, факультет ІРЕ, 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  
“      ”                                  2021 року

**З А В Д А Н Н Я**

**НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)**

Федорчука Данила Олександровича  
(прізвище, ім'я, по батькові)

1. Тема проєкту (роботи) Дослідження та програмна реалізація методів прогнозування часу доставки виробів пекарні  
Research and Software Implementation of Methods for Predicting the Bakery Delivery Time

керівник проєкту (роботи) Льовкін Валерій Миколайович, к.т.н., доцент,  
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу вищої освіти від “5” листопада 2021 року № 435

2. Строк подання студентом проєкту (роботи)            грудень 2021 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 | Льовкін В.М., доцент                      |                |                           |
| Нормоконтролер      | Камінська Ж.К., асистент                  |                |                           |
|                     |                                           |                |                           |
|                     |                                           |                |                           |

7. Дата видачі завдання “3” вересня 2021 року.

## КАЛЕНДАРНИЙ ПЛАН

| № з/п | Назва етапів дипломного проєкту (роботи)             | Строк виконання етапів проєкту (роботи) | Примітка             |
|-------|------------------------------------------------------|-----------------------------------------|----------------------|
| 1     | Аналіз та дослідження предметної області             | 1-2 тиждень                             |                      |
| 2     | Розробка та затвердження технічного завдання         | 3 тиждень                               | Технічне завдання    |
| 3     | Дослідження методів розв’язання проблеми             | 4 тиждень                               |                      |
| 4     | Проектування програмного забезпечення                | 5 тиждень                               |                      |
| 5     | Розроблення модулів програми                         | 6-7 тиждень                             |                      |
| 6     | Розроблення програмного коду                         | 8 тиждень                               | Текст програми       |
| 7     | Тестування та відлагодження програмного забезпечення | 9 тиждень                               | Програма             |
| 8     | Розробка розділів пояснювальної записки              | 10-11 тиждень                           | Пояснювальна записка |
| 9     | Розробка графічного матеріалу                        | 12 тиждень                              | Графічний матеріал   |
| 10    | Захист магістерської роботи                          | 13 тиждень                              |                      |
|       |                                                      |                                         |                      |
|       |                                                      |                                         |                      |
|       |                                                      |                                         |                      |

Студент(ка)

\_\_\_\_\_  
( підпис )      Федорчук Д.О.  
(прізвище та ініціали)

Керівник проєкту (роботи)

\_\_\_\_\_  
( підпис )      Льовкін В.М.  
(прізвище та ініціали)

## РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи магістра:  
91 с., 29 рис., 3 табл., 3 дод., 18 джерел.

ПЕКАРНЯ, ЧАС ДОСТАВКИ, ПРОГНОЗУВАННЯ, ДЕРЕВА  
РІШЕНЬ, ГРАДІЄНТНИЙ БУСТИНГ, XGBOOST.

Об'єкт дослідження – процес замовлення доставки виробів пекарні.  
Предмет дослідження – методи прогнозування часу доставки виробів пекарні. Мета роботи – розроблення програмного забезпечення замовлення доставки виробів пекарні для підвищення зручності роботи на основі прогнозування часу доставки.

Матеріали, методи та технічні засоби: об'єктно-орієнтоване програмування, мова програмування C#, бібліотека XGBoost.

Результати. Результатами роботи є виконане проєктування програмного забезпечення, включаючи моделювання сценаріїв роботи, вибір засобів розробки, визначення структури бази даних, програмна реалізація та створений метод прогнозування часу доставки.

Наукова новизна роботи полягає в запропонованому методі прогнозування часу доставки виробів пекарні, який використовує ансамбль дерев рішень з навчанням на основі градієнтного бустинга, що дозволяє визначати час доставки замовлень виробів пекарні за набором їх характеристик.

Висновки. Розроблений вебзастосунок може використовуватися клієнтами пекарні, кур'єрами, менеджерами, касирами та адміністраторами для замовлення виробів пекарні, керування цими замовленнями, перегляду доступних виробів з визначенням різної вартості виробів на різні періоди, створенням пропозицій на придбання наборів виробів.

Галузь використання – пекарні зі службою доставки та наявністю додаткових пунктів.

## ABSTRACT

Explanatory note to the diploma qualifying work of the master: 91 pages, 29 figures, 3 tables, 3 appendixes, 18 sources.

BAKERY, DELIVERY TIME, FORECASTING, DECISION TREE, GRADIENT BOOSTING, XGBOOST.

The object of research is the process of ordering the delivery of bakery products. The subject of research is methods of predicting the delivery time of bakery products. The purpose of work is development of the software of ordering the delivery of products of a bakery to increase the convenience of work on the basis of predicting delivery time.

Materials, methods and technical means: object-oriented programming, C# programming language, XGBoost library.

Results. The results of the work are software design, including modeling of use case scenarios, selection of development tools, determination of the database and the software structure, software implementation and the created method of predicting delivery time.

The scientific novelty of the work lies in the proposed method of predicting the delivery time of bakery products, which uses an ensemble of decision trees with training based on gradient boosting, which enables to predict the delivery time of bakery orders by a set of its characteristics.

Conclusions. The developed web application should be used by bakery customers, couriers, managers, cashiers and administrators to order bakery products, manage these orders, view available products, to determine different product prices for different periods, create offers to purchase product sets.

Area of use is bakeries with delivery service and additional departments.

## ЗМІСТ

|                                                                                             |    |
|---------------------------------------------------------------------------------------------|----|
|                                                                                             | С. |
| Перелік скорочень та умовних познач.....                                                    | 8  |
| Вступ.....                                                                                  | 9  |
| 1 Аналіз проблеми та постановка завдань дослідження .....                                   | 10 |
| 1.1 Огляд існуючих програмних засобів підтримки замовлень у пекарні ..                      | 10 |
| 1.2 Визначення функціональних вимог до програми .....                                       | 13 |
| 1.3 Висновки за розділом 1 та постановка завдань дослідження .....                          | 14 |
| 2 Матеріали і методи.....                                                                   | 16 |
| 2.1 Використання дерев рішень для розв’язання проблеми прогнозування .....                  | 16 |
| 2.2 Метод прогнозування часу доставки виробів пекарні .....                                 | 20 |
| 2.3 Висновки за розділом 2 .....                                                            | 22 |
| 3 Проєктування програмного забезпечення .....                                               | 24 |
| 3.1 Вибір засобів веброзробки.....                                                          | 24 |
| 3.2 Моделювання роботи користувачів з програмою .....                                       | 25 |
| 3.3 Проєктування бази даних.....                                                            | 30 |
| 3.4 Висновки за розділом 3 .....                                                            | 38 |
| 4 Основні рішення щодо реалізації компонентів системи .....                                 | 39 |
| 4.1 Опис послідовності роботи з замовленнями на доставку .....                              | 39 |
| 4.2 Опис модулів реалізованої програмної системи.....                                       | 41 |
| 4.3 Висновки за розділом 4 .....                                                            | 46 |
| 5 Експлуатація, тестування та експериментальне дослідження програми .....                   | 48 |
| 5.1 Призначення програми.....                                                               | 48 |
| 5.2 Умови виконання програми .....                                                          | 48 |
| 5.3 Виконання програми .....                                                                | 48 |
| 5.4 Експериментальне дослідження проблеми прогнозування часу доставки виробів пекарні ..... | 54 |
| 5.5 Висновки за розділом 5 .....                                                            | 56 |
| Висновки.....                                                                               | 57 |

|                                    |    |
|------------------------------------|----|
| Перелік джерел посилання .....     | 59 |
| Додаток А Технічне завдання.....   | 61 |
| Додаток Б Текст програми.....      | 65 |
| Додаток В Слайди презентації ..... | 83 |

## ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

|         |                                       |
|---------|---------------------------------------|
| CART    | – classification and regression tree; |
| ID3     | – iterative dichotomiser 3;           |
| XGBoost | – extreme gradient boosting;          |
| БД      | – база даних;                         |
| ПЗ      | – програмне забезпечення.             |

## ВСТУП

Пекарні є частиною традицій в багатьох країнах. Свіжий хліб, круасани – все це зокрема є обов’язковою складовою частиною сніданків у Франції. Пекарня поруч з будинком присутня в житті багатьох громадян країн Європейського союзу. У той же час в Україні така традиція тільки починає розповсюджуватися, а тому знайти пекарню поруч з будинком, а тим більше з якісними виробами ще може бути достатньо складно. Вирішити це питання може наявність доставки.

Забезпечення доставки виробів пекарень може зробити життя людей зручнішим. При цьому важливо не тільки розробити програмне забезпечення (ПЗ) підтримки даного процесу, але і виконати прогнозування тривалості доставки, адже розуміння того, як буде працювати доставка для конкретного замовлення, є важливим для підтримки доставки виробів пекарні. Така інформація може бути корисною одразу для двох сторін:

- клієнти будуть розуміти, скільки фактично часу може зайняти доставка свіжих виробів для них і спланувати свій час, що зокрема дуже важливо, якщо замовлення доставляється для організації сніданку, обіду, а не тільки як певний десерт;
- менеджери пекарні будуть краще розуміти, як відпрацює служба доставки компанії у конкретному випадку і, якщо це потрібно, будуть завчасно реагувати на потенційні затримки, щоб зробити цей процес, за можливості, зручнішим для всіх клієнтів.

У результаті метою роботи є розроблення програмного забезпечення замовлення доставки виробів пекарні для підвищення зручності роботи на основі прогнозування часу доставки [1].

## 1 АНАЛІЗ ПРОБЛЕМИ ТА ПОСТАНОВКА ЗАВДАНЬ ДОСЛІДЖЕННЯ

### 1.1 Огляд існуючих програмних засобів підтримки замовлень у пекарні

Аналізуючи ринок пекарень в Україні і світі необхідно зазначити, що багато з них не мають своїх власних вебсайтів, а ті, що мають, не завжди мають засоби замовлення їх виробів. При чому частіше замовлення підтримуються для пекарень, які виготовляють святкові вироби.

Першим серед програмних аналогів розглянуто вебсайт пекарні Baker Street Bakery (рис. 1.1) [2].

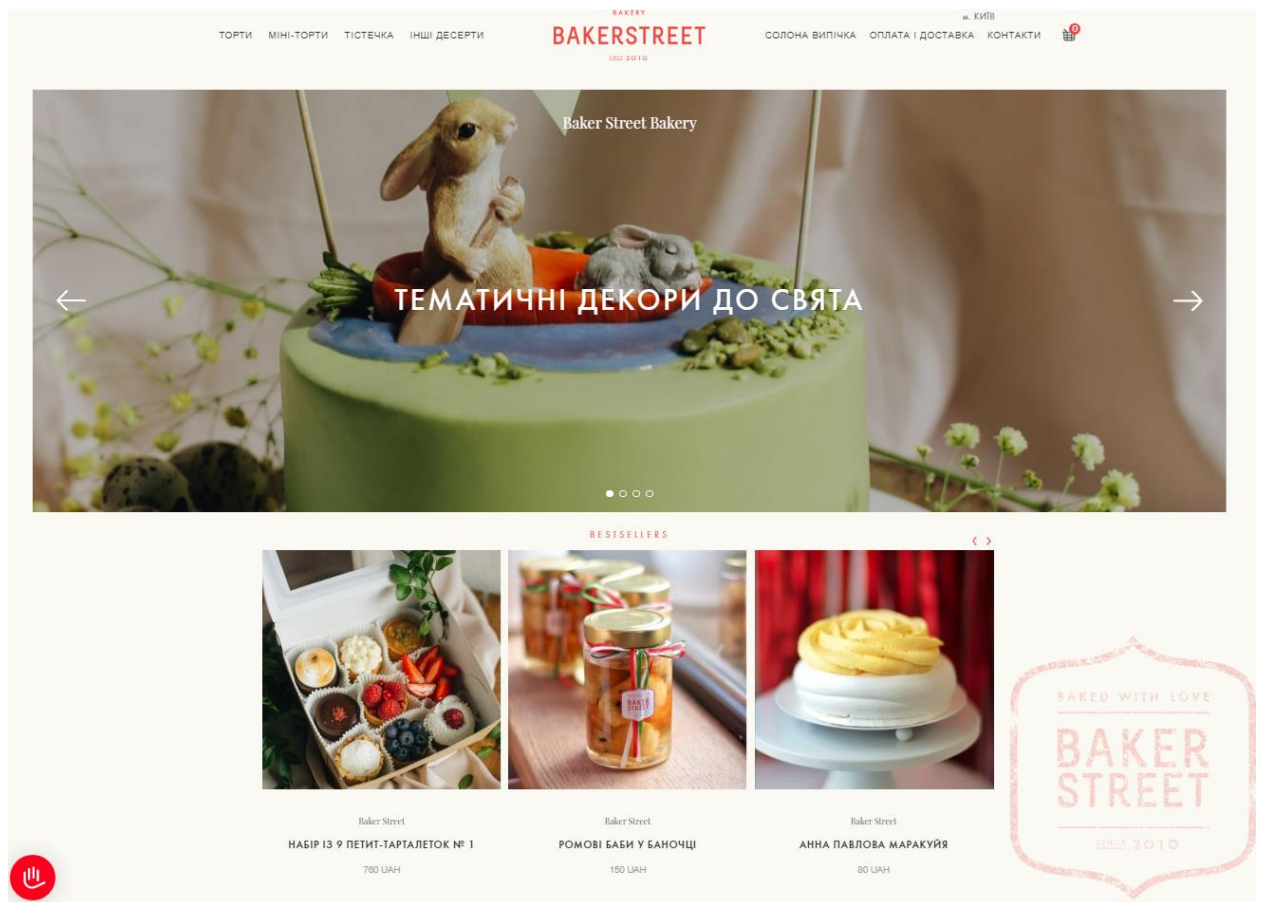


Рисунок 1.1 – Вебсайт пекарні Baker Street Bakery

На основній вебсторінці пекарні відображається перелік з 6 виробів з зазначенням їх вартості. Окрім того вироби відображаються за групами, які

доступні з меню вебсайту. Також можна замовити індивідуальний виріб. Замовлення доступне за допомогою стандартної контактної форми.

Звичайне замовлення формується шляхом додавання виробів у потрібній кількості в кошик і визначення способу доставки.

Вебсайт пекарні Shoso (рис. 1.2) дозволяє переглядати вироби як за окремими групами (до них окрім десертів належать, наприклад, і сніданки), так і безпосередньо інформацію про самі вироби, а також замовляти їх з можливістю доставки [3].

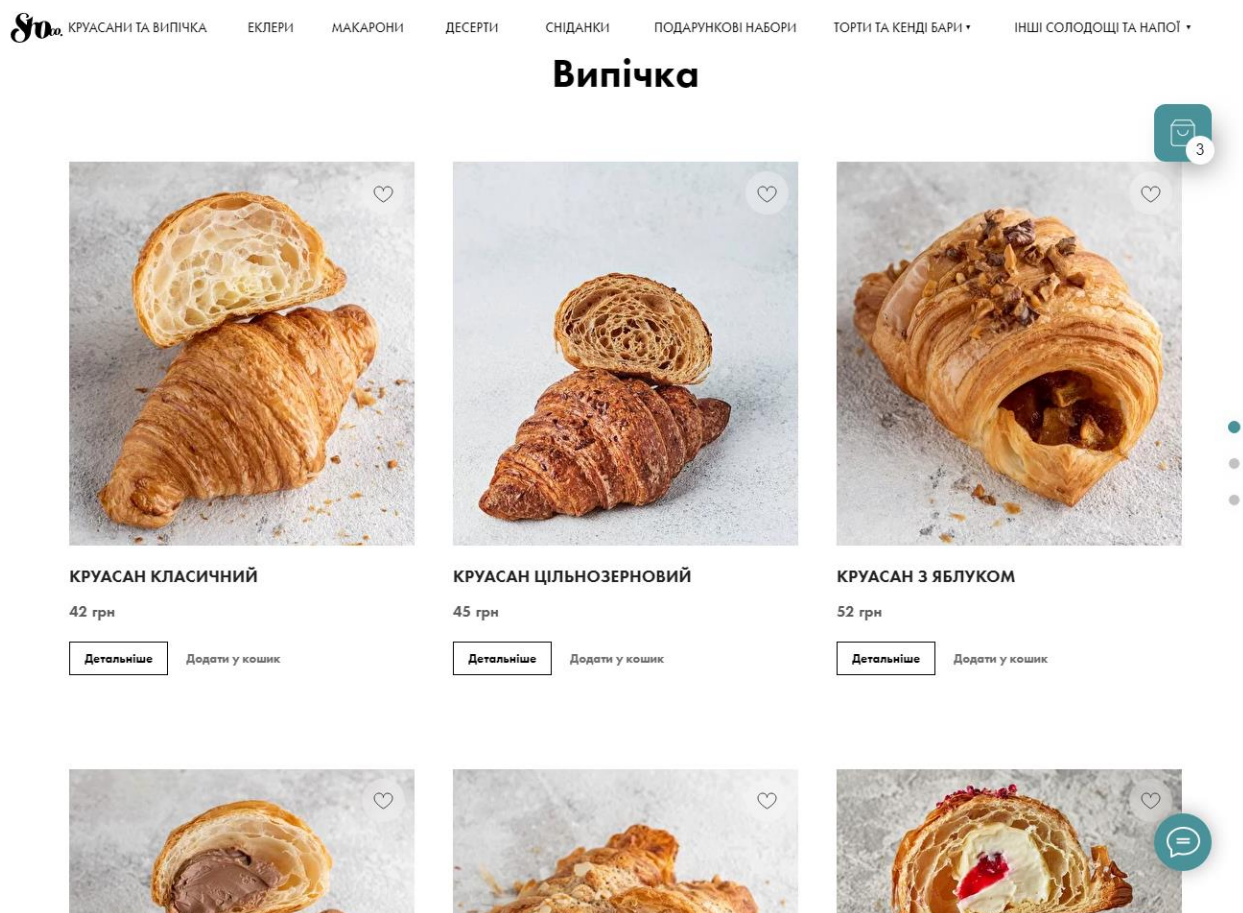


Рисунок 1.2 – Вебсайт пекарні Shoso

Під час оформлення замовлення (рис. 1.3) потрібно вказати дані про себе, місто і час доставки, а також надається можливість залишити коментар до замовлення.

Ваші замовлення



Круасан класичний

⊖

1

⊕

42 грн

⊗



Круасан з яблуком

⊖

2

⊕

104 грн

⊗

Сума: 146 грн

Залиште свої дані нижче

Ми зателефонуємо вам протягом 15 хвилин для підтвердження замовлення

Ваше ім'я

Максим Максименко

Ваш телефон

Перевірте, будь ласка, правильність номеру

Ваш email

your-mail@site.com

Instagram чи Facebook, якщо бажаєте

Бажаєте каву з собою?

Рисунок 1.3 – Оформлення замовлення через вебсайт пекарні Shoco

У якості прикладу вебсайту іноземної компанії було використано вебсайт лондонської пекарні Comptoirgourmand (рис. 1.4) [4]. Даний програмний засіб дозволяє замовляти, а також передзамовляти вироби пекарні. Доступні і десерти, і закуски. До замовлення можна залишати коментарі, що дозволяє вказати на особливі потреби.

Окрім того до виробу може пропонуватися спеціальна пропозиція, що дозволяє купити набір виробів за зниженою вартістю.

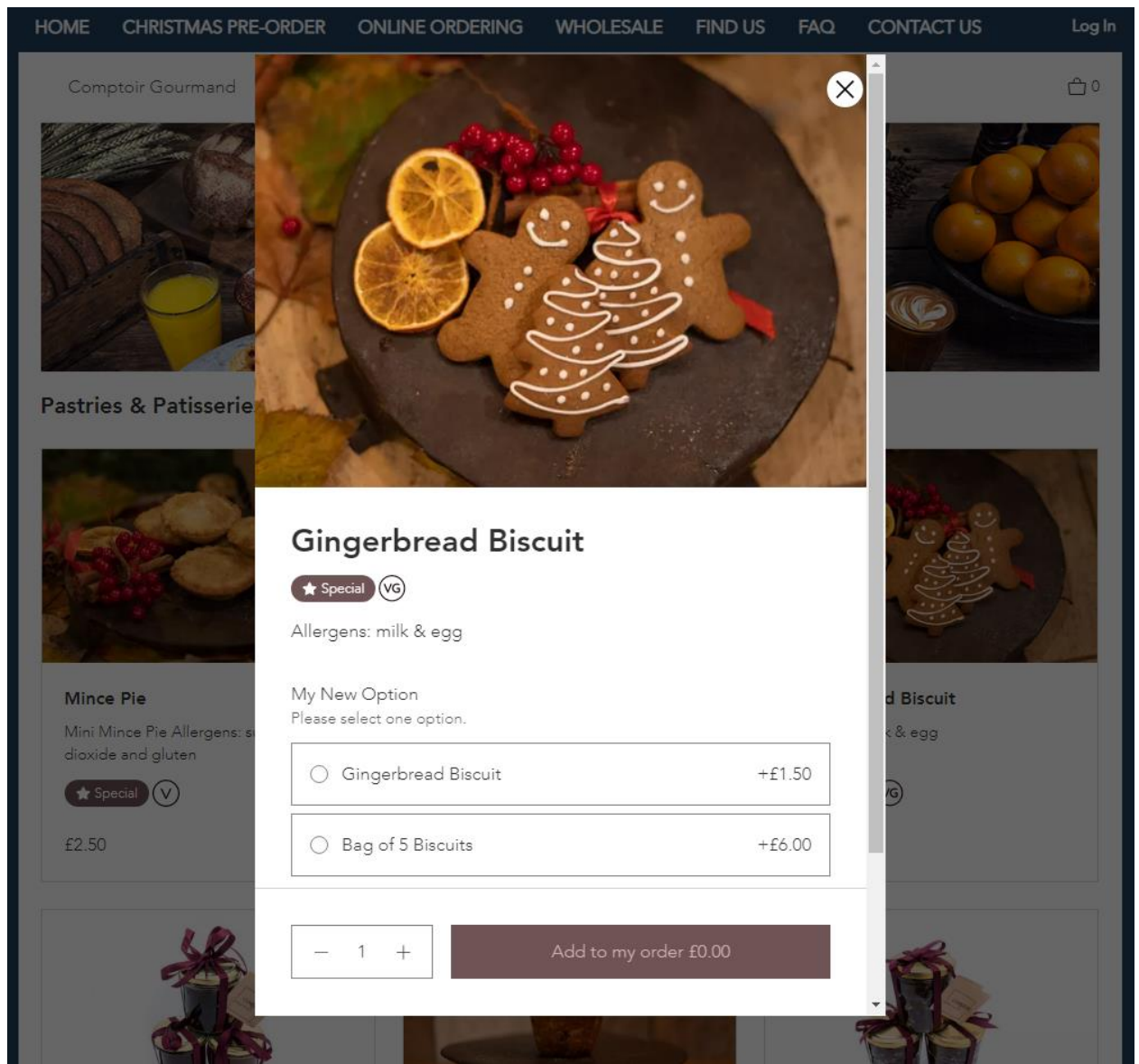


Рисунок 1.4 – Вебсайт пекарні Comptoirgourmand

## 1.2 Визначення функціональних вимог до програми

ПЗ, що розробляється в роботі, для реалізації мети роботи повинно забезпечувати виконання функцій:

- замовлення доставки виробів пекарні;
- відображення виробів пекарні за групами;
- відображення груп виробів пекарні;
- відображення замовлень, які потребують відвантаження;
- керування замовленнями;

- скасування замовлення;
- відображення прострочених замовлень;
- відображення замовлень за період;
- вибір транспортного засобу для доставки;
- відображення замовлень для обслуговування кур'єру;
- відображення власних замовлень;
- обчислення вартості замовлення;
- внесення випічених виробів;
- внесення запланованих до випікання обсягів виробів;
- визначення доставки замовлення;
- визначення кур'єра для доставки;
- прогнозування тривалості доставки;
- внесення нових виробів пекарні;
- визначення стану транспортного засобу служби доставки;
- створення нових груп виробів пекарні;
- додавання виробів пекарні до груп;
- редагування існуючих записів про вироби пекарні, групи виробів;
- внесення виробників транспортних засобів служби доставки, самих транспортних засобів, їх типів і моделей, відображення та редагування існуючих;
- зв'язування кур'єра з транспортним засобом;
- створення спеціальних пропозицій щодо купівлі наборів виробів;
- визначення вартості виробів пекарні на конкретний період часу;
- визначення транспорту служби доставки;
- відображення даних про виріб.

### **1.3 Висновки за розділом 1 та постановка завдань дослідження**

За результатами огляду програмних засобів підтримки замовлень у пекарні було виділено основні функціональні можливості даних програмних

засобів і визначено функціональні вимоги до програми. Особливістю програми є надання можливості прогнозування часу доставки виробів пекарні.

Завданнями магістерської роботи є:

- дослідження методів прогнозування часу доставки виробів пекарні;
- розробка технічного завдання і вибір засобів розробки;
- проєктування програмної системи;
- розробка програмної системи замовлення доставки виробів пекарні.

## 2 МАТЕРІАЛИ І МЕТОДИ

### 2.1 Використання дерев рішень для розв'язання проблеми прогнозування

Дерева рішень – це моделі класифікації або регресії, які виконують визначення цільової змінної (передбачення її значення) на основі інших змінних:

- використання деревоподібної структури для прийняття рішення є дуже прозорим і зрозумілим, особливо якщо дерево не надто велике, як і інші типи моделей, що використовуються в машинному навчанні, такі як штучні нейронні мережі;

- модель будується шляхом навчання її за допомогою навчальних даних, а потім здійснюється перевірка його на тестових даних, щоб налаштувати модель на переобладнання та покращити узагальнення моделі;

- для оцінки якості кінцевої прогнозної моделі використовуються різні критерії, наприклад, помилка класифікації або прогнозування;

- основна ідея полягає в тому, щоб передбачити майбутні дані на основі минулих даних, припускаючи, що базова система достатньо стабільна, щоб зробити такі прогнози дійсними [4]-[5].

Алгоритм дерева рішень використовує екземпляр даних і проводить його через все дерево, задаючи запитання чи це правда, чи не неправда, починаючи з кореневого вузла, ставлячи запитання, і для кожної відповіді створюються окремі гілки, і це триває до тих пір, поки не буде досягнуто листкового вузла, для побудови дерева використовується рекурсивне розбиття [6].

Переваги дерев рішень:

- модель дерева рішень може використовуватися як для задач класифікації, так і для задач регресії, і її легко інтерпретувати, розуміти та візуалізувати;

- вихід дерева рішень можна легко зрозуміти: у порівнянні з іншими алгоритмами підготовка даних під час попередньої обробки в дереві рішень вимагає менше зусиль і не вимагає нормалізації даних;

- реалізація алгоритму може бути виконана без масштабування даних: дерево рішень є одним із найшвидших способів ідентифікувати зв'язки між змінними та найбільш значимою змінною;

- на дерева рішень не дуже впливають викиди або відсутні значення;

- дерева рішень можуть обробляти як числові, так і категоріальні змінні;

- оскільки це непараметричний метод, він не має припущень щодо розподілу простору та структури моделі [6]-[7].

Структура дерева рішень характеризується наступним:

- дерево має гілки, вузли, листки тощо;

- кореневий вузол – це початковий вузол, що представляє всю вибірку або сукупність, і його можна далі розділити на інші вузли або однорідні набори: вузол рішення складається з двох або більше вузлів, які представляють окремі значення перевіреного атрибута;

- листовий/кінцевий вузол не розбивається на інші вузли, і він представляє рішення;

- гілка або піддерево – це частина цілого дерева;

- розбиття – це процес поділу вузла на два або більше підвузлів, протилежність розділенню називається усіканням, тобто видаленням підвузлів вузла рішення;

- батьківський вузол – це вузол, який розбивається на підвузли, а підвузол – дочірній вузол [6].

Градiєнтний бустинг є одним з найпотужніших методів для побудови прогнозних моделей, він дозволяє мінімізувати функцію втрат шляхом додавання слабких учнів за допомогою алгоритму оптимізації градієнтного спуску, при цьому можуть використовуватися довільні диференційовані функції втрат, розширюючи техніку за межі проблем бінарної класифікації

для підтримки регресії, класифікації за багатьма класами тощо, алгоритм складається з трьох основних компонентів:

- функція втрат: роль функції втрат полягає в тому, щоб оцінити, наскільки кращою є модель для прогнозування з наданими даними;
- слабкий учень – це класифікатор, що погано класифікує дані в порівнянні з випадковими припущеннями;
- адитивна модель: це ітераційний і послідовний процес додавання дерев рішень по одному кроку, кожна ітерація повинна зменшувати значення функції втрат, додається фіксована кількість дерев, або навчання припиняється, коли втрата досягає прийняттого рівня або більше не покращується набір даних зовнішньої перевірки [8]-[9].

Особливості алгоритму навчання дерев рішень extreme gradient boosting (XGBoost):

- регулярне навчання: упорядкування допомагає згладити остаточні засвоєні ваги, щоб уникнути надмірного налаштування, регуляризована ціль буде мати тенденцію до вибору моделі, яка використовує прості та передбачувані функції;
- градієнтний бустинг дерев рішень: модель ансамблю дерев не може бути оптимізована за допомогою традиційних методів оптимізації в євклідовому просторі, натомість модель тренується адитивним чином;
- усадка та підвибірка стовпців: окрім регуляризованої мети, використовуються дві додаткові методики для подальшого запобігання перенавчанню, перша техніка – усадка масштабує нещодавно додані ваги на коефіцієнт після кожного кроку бустингу дерева, подібно до швидкості навчання в стохастичній оптимізації, усадка зменшує вплив кожного дерева і залишає простір для майбутніх дерев для покращення моделі, друга техніка – підвибірка стовпців, що запобігає перенавчанню і прискорює обчислення паралельного алгоритму [8].

XGBoost – це ансамблевий метод навчання, адже не завжди достатньо покладатися на результати лише однієї моделі машинного навчання, у свою

чергу ансамблеве навчання надає систематичне рішення для поєднання сили прогнозування декількох моделей прогнозування, результатом є одна модель, яка дає сукупний результат декількох моделей:

- моделі, які утворюють ансамбль, також відомі як базові учні, можуть використовувати як один алгоритм навчання, так і різні, так беггінг і бустинг – два широко використовувані ансамблеві методи навчання, хоча ці дві методики можна використовувати з декількома статистичними моделями, найбільш ефективно їх використання було досягнуто з деревами рішень;

- беггінг: хоча дерева рішень є однією з найбільш легко інтерпретованих моделей, вони демонструють дуже мінливу поведінку, тому якщо один навчальний набір даних випадковим чином розділили на дві частини, то кожен частину можна використати для навчання дерев рішень, щоб отримати дві моделі, якщо застосувати обидві ці моделі, вони дадуть різні результати, тому кажуть, що дерева рішень пов'язані з високою дисперсією через таку поведінку, бустинг або беггінг допомагають зменшити дисперсію у будь-якого учня, декілька дерев рішень, які генеруються паралельно, становлять основу для вивчення техніки беггінга, дані, відібрані із заміною, подаються цим учням для навчання, остаточний прогноз – усереднений результат усіх учнів;

- бустинг: під час бустингу дерева будуються послідовно так, що кожне наступне дерево має на меті зменшити помилки попереднього дерева, кожне дерево вчиться у своїх попередників і оновлює залишкові помилки, отже дерево, яке росте наступним у послідовності, буде навчатися на оновленій версії залишків даних, основні учні, які навчаються у бустингу, – це слабкі учні, у яких поріг є високим, а передбачення лише трохи краще, ніж випадкові припущення, кожен із цих слабких учнів надає певну життєво важливу інформацію для прогнозування, що дозволяє підвищити ефективність сильного учня шляхом ефективного поєднання слабких учнів, останній сильний учень зменшує дисперсію [10].

Техніка бустингу складається з трьох простих кроків:

- початкова модель визначена для прогнозування цільової змінної;
- нова модель підлаштовується до залишків з попереднього кроку;
- моделі з попередніх кроків об'єднуються, щоб отримати нову модель, розширену версію першої, середня квадратична помилка від другої моделі буде нижчою, ніж від першої [4]-[10].

Серед дерев рішень було виділено наступні алгоритми навчання для порівняння:

- iterative dichotomiser 3 (ID3);
- classification and regression tree (CART);
- C4.5;
- XGBoost.

Порівняння існуючих алгоритмів навчання дерев рішень зведено в табл. 2.1.

Таблиця 2.1 – Порівняння алгоритмів навчання дерев рішень

| Критерій                              | ID3 | C4.5 | CART | XGBoost |
|---------------------------------------|-----|------|------|---------|
| Задача класифікації                   | +   | +    | +    | +       |
| Задача прогнозування                  | –   | –    | +    | +       |
| Підтримка бустингу                    | –   | –    | +    | +       |
| Швидкість роботи                      | –   | –+   | +    | ++      |
| Підтримка неповної навчальної вибірки | –   | +    | +    | +       |

## 2.2 Метод прогнозування часу доставки виробів пекарні

Для роботи метода прогнозування часу доставки виробів пекарні спочатку потрібно сформулювати набір даних.

Набір даних формується на основі використання наступних ознак зі зведенням до описаних значень:

- час початку доставки: момент, коли замовлення почало формуватися на видачу (відвантаження виробів);
- тип транспортного засобу: автомобіль, мотоцикл, велосипед безпосередньо впливають на час доставки;
- кількість одиниць виробів у замовленні: більший обсяг замовлення призводить до збільшення часу на його оброблення, може призводити до необхідності пошуку відповідного транспортного засобу;
- район, в якому розташовано початковий пункт доставки: визначає маршрут, який впливає на час доставки;
- найкоротша відстань від точки відправлення до точки доставки: чим більш далеко розташовані пункти, тим довше займає доставка;
- напрямок руху від початкової точки до кінцевої ( $90^\circ$  – схід,  $-90^\circ$  – захід,  $-180^\circ$  або  $+180^\circ$  – північ або південь): напрямок руху визначає, які райони доводиться перетинати, щоб дістатися потрібної точки, тому у випадку підвищеного трафіку в цей час в районі перетину може відбуватися затримка;
- температура повітря ( $^\circ\text{C}$ ): погодні умови також впливають на час доставки, адже впливають на сам транспортний засіб, необхідність його обслуговування тощо;
- вологість повітря (%);
- напрямок вітру: даний фактор по-різному впливає на різні типи транспортних засобів, але впливає на будь-який з них, при цьому вплив залежить і від напрямку руху транспортного засобу;
- швидкість вітру (км/г): чим більша швидкість вітру, тим більшими можуть бути затримки, особливо при доставці велосипедом;
- видимість (км): за невеликої видимості доставка також може тривати довше, при цьому в даному факторі агреговано і фактор наявності туману, адже таким чином визначається його сила;

- наявність снігу: сніг сам по собі впливає на умови, в яких працює кур'єр, окрім того може у великій кількості впливати на можливість взагалі дістатися до певного місця;
- наявність дощу: дощ ускладнює умови руху транспортного засобу;
- час доставки – час, за який було віддано замовлення повністю клієнту, починаючи з часу початку його формування.

У випадку відсутності даних про погоду вони мають бути взяті з зовнішнього джерела за статистикою погоди в даному місті у вказаний період часу. У випадку наявності інших пропущених значень ознак у наборі даних, вони мають бути заповнені значеннями за замовчуванням (наприклад, кількість одиниць виробів у замовленні вважається рівною 1, якщо не вказано іншого).

Після цього реалізується навчання моделі прогнозування на основі дерев рішень та алгоритму навчання XGBoost за допомогою сформованого набору даних. Даний алгоритм обрано за результатами порівняння з іншими алгоритмами навчання за сукупністю критеріїв (табл. 2.1), адже з одного боку він придатний для розв'язання задачі прогнозування, яка як раз і розглядається в роботі, а з іншого боку він, знаходячись на верхівці ієрархії алгоритмів навчання дерев рішень, відзначається високою швидкістю роботи (в тому числі за рахунок здатності до паралелізації), підтримкою бустингу, що дозволяє досягти кращої точності прогнозування за рахунок різноманітності наявних моделей.

## **2.3 Висновки за розділом 2**

Виконано аналіз підходу до прогнозування на основі дерев рішень з виділенням основних використовуваних на даний момент технік, що використовуються для навчання дерев рішень. Виконано порівняння алгоритмів навчання дерев рішень.

На основі отриманих результатів у роботі запропоновано метод прогнозування часу доставки виробів пекарні, який використовує ансамбль дерев рішень з навчанням на основі градієнтного бустинга, що дозволяє визначати час доставки замовлень виробів пекарні за набором їх характеристик.

### 3 ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Зважаючи на актуальні на даний момент засоби забезпечення віддаленого доступу, ПЗ потрібно реалізувати як вебзастосунок.

#### 3.1 Вибір засобів веброзробки

Для веброзробки важливим фактором є навіть не вибір самої мови програмування, а вибір загальної концепції веброзробки, що включає до свого складу і певну мову програмування, тому вибір засобів веброзробки виконувався саме на такому рівні (табл. 3.1).

Таблиця 3.1 – Вибір вебфреймворка

| Критерій                                    | Django | ASP.NET Core |
|---------------------------------------------|--------|--------------|
| Мова програмування                          | Python | C#           |
| Наявність об'єктно-реляційного відображення | +      | +            |
| Використання шаблону проєктування           | +      | +            |
| Загальна надійність                         | +/-    | +            |
| Швидкісне виконання                         | -/+    | +            |
| Кросплатформність                           | +      | +            |

Порівнюючи між собою вебфреймворки Django і ASP.NET Core, можна зазначити, що обидва є достатньо актуальними засобами, проте в результаті було вирішено виконувати розробку на основі вебфреймворку ASP.NET Core [11]-[15]. Це в результаті було обумовлено вищою загальною надійністю створених рішень, оскільки загалом фреймворк ASP.NET існує достатньо давно, добре відлагоджений, забезпечується підтримкою компанії Microsoft, а ASP.NET Core є його подальшим кросплатформним розвитком.

Також ASP.NET Core дозволяє створювати дещо кращі рішення за швидкістю виконання. Тож зважаючи на ці два критерія, а також те, що в результаті має бути створене рішення для практичного використання в системах середнього або великого масштабу (де ці два критерія є дуже важливими), було обрано саме ASP.NET Core для розробки.

### 3.2 Моделювання роботи користувачів з програмою

За результатами моделювання роботи клієнта з програмою створено відповідну діаграму прецедентів (рис. 3.1).

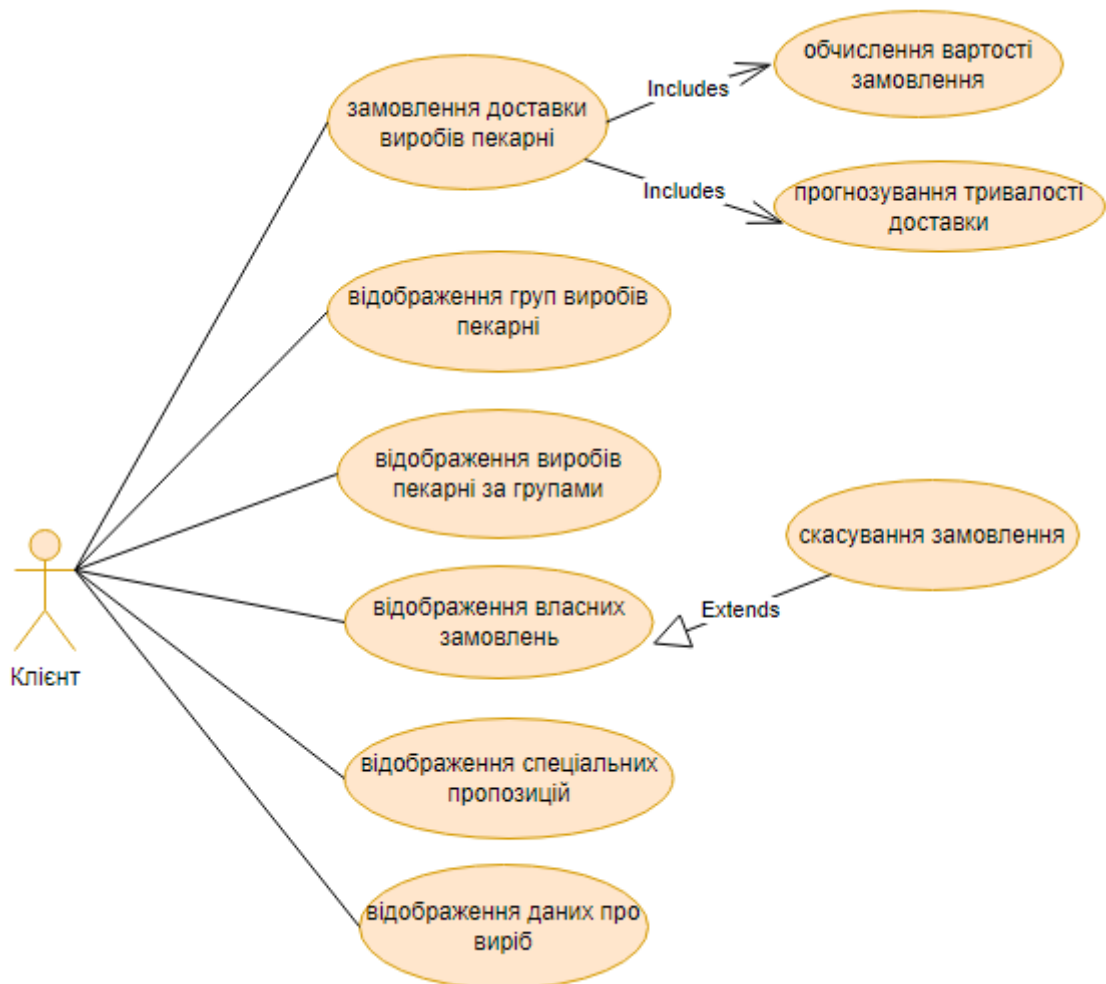


Рисунок 3.1 – Діаграма прецедентів клієнта

Потік прецедентів клієнта:

- замовлення доставки виробів пекарні;
- відображення виробів пекарні за групами;
- відображення груп виробів пекарні;
- відображення власних замовлень;
- відображення спеціальних пропозицій;
- відображення даних про виріб.

Менеджер у ПЗ відповідальний за керування замовленнями і підтримку всіх процесів стосовно цих замовлень.

Потік прецедентів менеджера (рис. 3.2):

- замовлення доставки виробів пекарні;
- відображення виробів пекарні за групами;
- відображення груп виробів пекарні;
- керування замовленнями;
- відображення замовлень;
- внесення випічених виробів;
- внесення запланованих до випікання обсягів виробів;
- керування транспортними засобами служби доставки;
- створення спеціальних пропозицій щодо купівлі наборів виробів;
- відображення спеціальних пропозицій;
- визначення вартості виробів пекарні на конкретний період часу;
- визначення транспорту служби доставки;
- відображення даних про виріб (рис. 3.2).

Кур'єр відповідальний за доставку замовлень клієнтам.

Потік прецедентів кур'єра:

- відображення замовлень для обслуговування кур'єру;
- визначення стану транспортного засобу служби доставки;
- відображення складу замовлення;
- керування замовленнями.

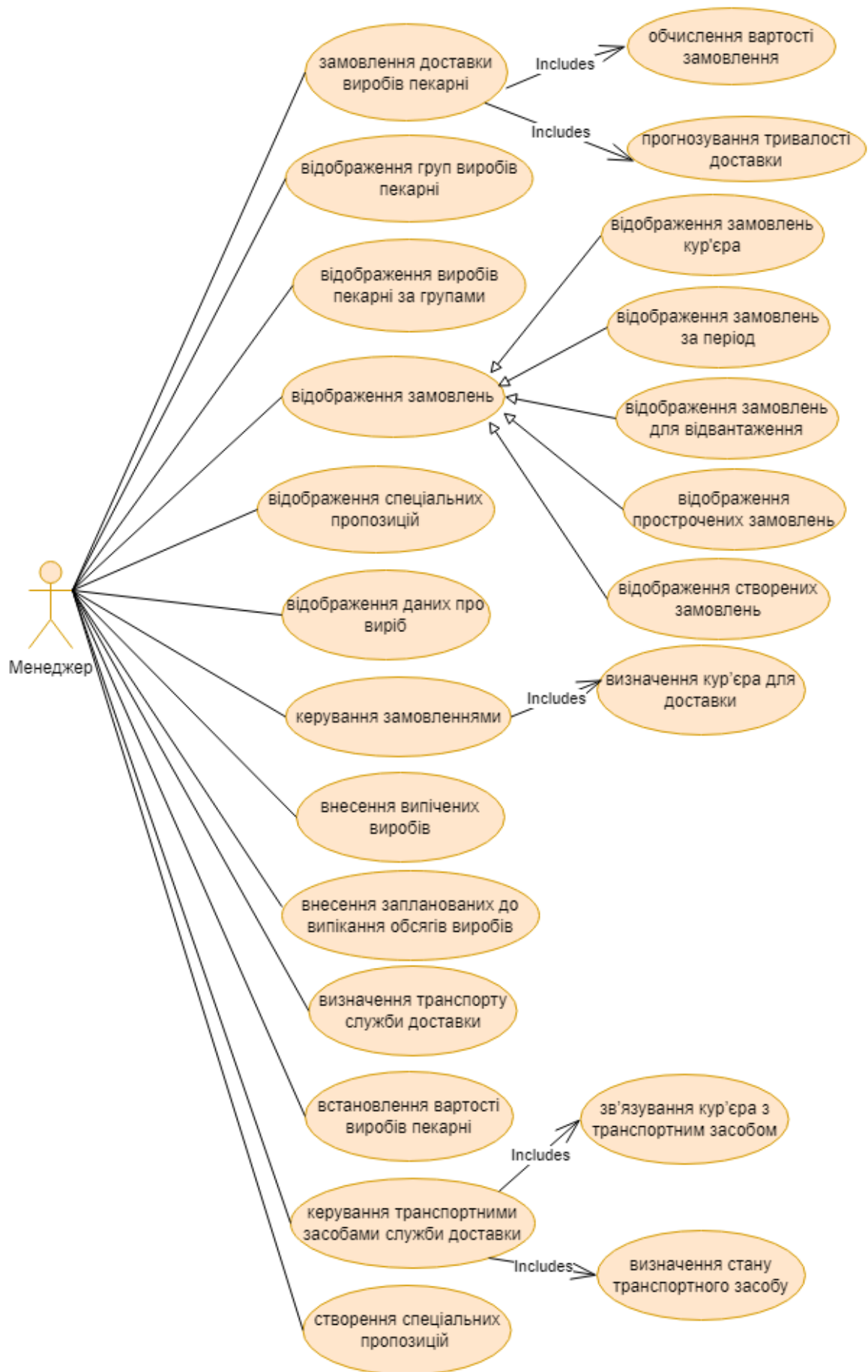


Рисунок 3.2 – Діаграма прецедентів менеджера

На рис. 3.3 представлено діаграму прецедентів кур'єра.

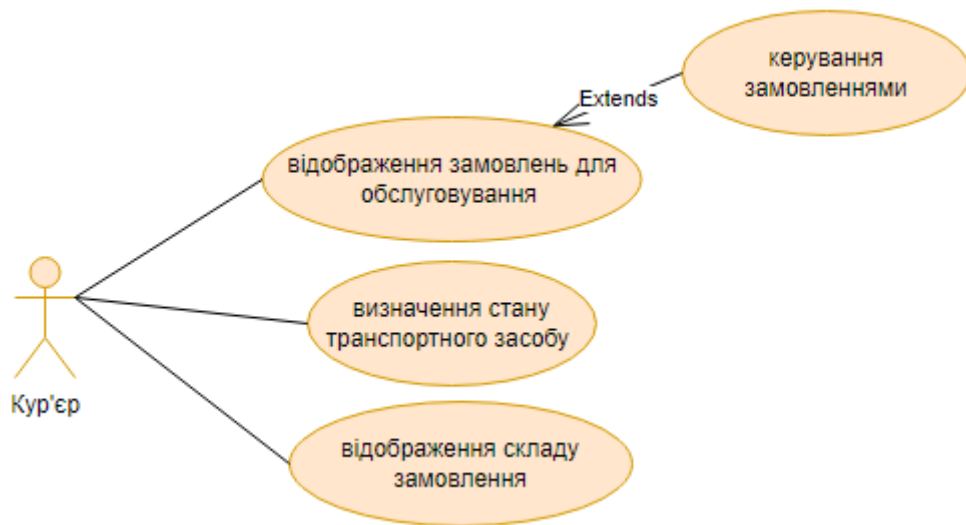


Рисунок 3.3 – Діаграма прецедентів кур'єра

Касира (рис. 3.4) було виділено окремо серед користувачів для підтримки внесення даних про виконані замовлення на придбання виробів пекарні без необхідності доставки.



Рисунок 3.4 – Діаграма прецедентів касира

Потік прецедентів касира:

- оформлення купівлі виробів пекарні;
- відображення виробів пекарні за групами;
- відображення груп виробів пекарні;
- обчислення вартості замовлення;
- відображенні спеціальних пропозицій щодо купівлі наборів виробів;
- відображення даних про виріб.

Адміністратор (рис. 3.5) відповідальний за внесення основних даних, які використовуються в роботі інших користувачів.

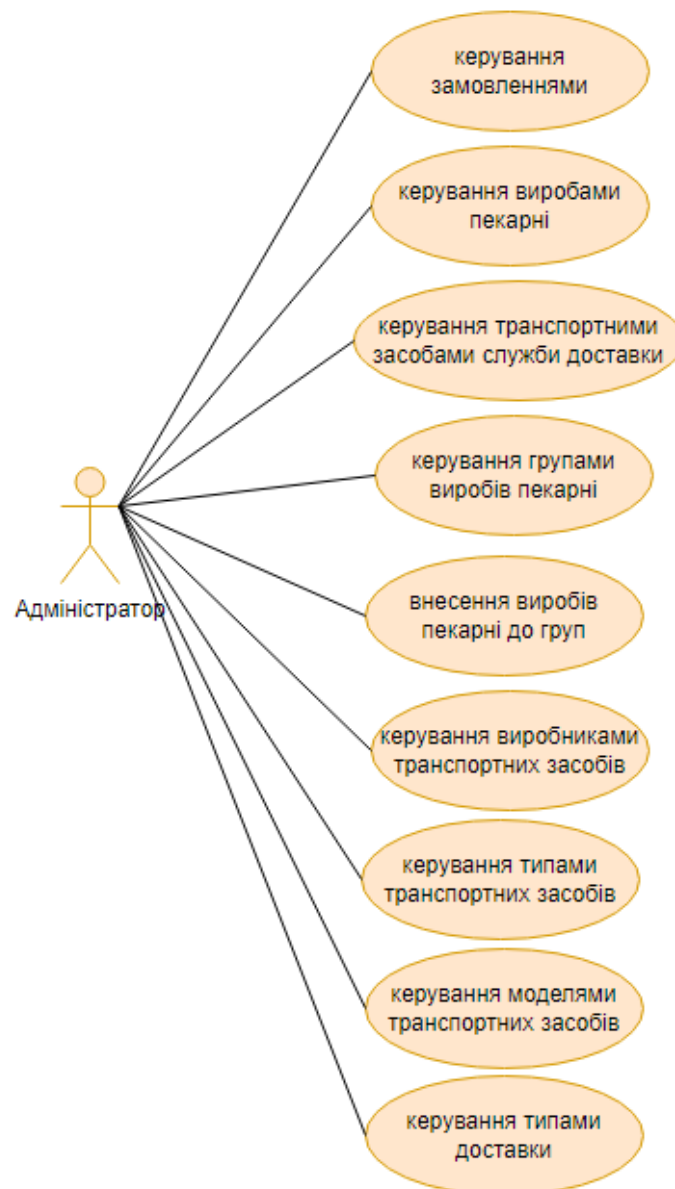


Рисунок 3.5 – Діаграма прецедентів адміністратора

Потік прецедентів адміністратора:

- керування замовленнями;
- керування виробами пекарні;
- керування транспортними засобами служби доставки;
- керування групами виробів пекарні;
- внесення виробів пекарні до груп;
- керування виробниками транспортних засобів;
- керування типами транспортних засобів;
- керування моделями транспортних засобів;
- керування типами доставки.

### **3.3 Проєктування бази даних**

Для підтримки виконання описаних функцій програма повинна забезпечувати відповідну роботу з даними. Для цього було визначено структуру бази даних (БД), що складається з наступних таблиць.

Таблиця `VehicleType` визначає тип транспортного засобу (автомобіль, велосипед тощо), який використовується для доставки службою доставки пекарні, включає такий набір стовпців:

- `VehicleTypeId` – ідентифікатор типу транспортного засобу, що є первинним ключем таблиці;
- `Title` – назва типу транспортного засобу.

Таблиця `VehicleManufacturer` визначає виробника транспортного засобу і включає такий набір стовпців:

- `VehicleManufacturerId` – ідентифікатор виробника транспортного засобу, що є первинним ключем таблиці;
- `Title` – назва виробника транспортного засобу.

Таблиця `VehicleModel` визначає модель транспортного засобу і включає такий набір стовпців:

- VehicleModelId – ідентифікатор моделі транспортного засобу, що є первинним ключем таблиці;
- VehicleTypeId – тип транспортного засобу, до якого належить дана модель, є зовнішнім ключем таблиці VehicleType;
- VehicleManufacturerId – виробник транспортного засобу, що є зовнішнім ключем таблиці VehicleManufacturer;
- Title – назва моделі транспортного засобу;
- Quantity – кількість транспортних засобів даної моделі.

Між таблицями VehicleType і VehicleModel зв'язок один до багатьох: один тип транспортного засобу представлений тимчасово нулем, однією або декількома моделями. Між таблицями VehicleManufacturerId і VehicleModel зв'язок один до багатьох: один виробник транспортного засобу представлений тимчасово нулем, однією або декількома моделями.

Таблиця Vehicle визначає конкретний транспортний засіб служби доставки і включає такий набір стовпців:

- VehicleId – ідентифікатор транспортного засобу, що є первинним ключем таблиці;
- VehicleModelId – модель транспортного засобу, що є зовнішнім ключем таблиці VehicleModel;
- Number – номер транспортного засобу (може бути відсутній фактично: наприклад, номер має автомобіль, але він відсутній у велосипеда);
- Status – стан транспортного засобу (необхідний, щоб відзначити недоступність транспортного засобу).

Між таблицями VehicleModel і Vehicle зв'язок один до багатьох: одна модель транспортного засобу представлена тимчасово нулем, одним або декількома конкретними засобами.

Таблиця CourierVehicle визначає використання транспортного засобу кур'єром для виконання його службових функцій протягом деякого періоду часу і включає такий набір стовпців:

- CourierVehicleId – ідентифікатор використання транспортного засобу кур'єром, що є первинним ключем таблиці;
- UserId – користувач програми, кур'єр, є зовнішнім ключем таблиці User;
- VehicleId – транспортний засіб, який використовує кур'єр, є зовнішнім ключем таблиці Vehicle;
- StartDate – дата початку використання транспортного засобу кур'єром;
- EndDate – дата завершення використання транспортного засобу кур'єром;
- Status – стан даного використання (завершено – кур'єр тепер користується іншим транспортним засобом, продовжується).

Фактично дана таблиця реалізовує зв'язок багато до багатьох між сутностями User і Vehicle. Між таблицями User і CourierVehicle зв'язок один до багатьох: один користувач (кур'єр) може користуватися різними транспортними засобами протягом всієї своєї роботи в службі доставки, тому записів може бути тимчасово нуль, один або декілька. Між таблицями Vehicle і CourierVehicle зв'язок один до багатьох: один транспортний засіб може використовуватися різними кур'єрами, тому записів може бути тимчасово нуль, один або декілька.

Таблиця Order визначає замовлення на придбання виробів пекарні, при цьому в основному розглядаються замовлення з доставкою, але підтримується і внесення замовлень касирами. Таблиця включає такий набір стовпців:

- OrderId – ідентифікатор замовлення, що є первинним ключем таблиці Order;
- CourierVehicleId – транспортний засіб, яким виконується доставка, є зовнішнім ключем таблиці;
- UserId – користувач, який замовив вироби пекарні, є зовнішнім ключем таблиці User;

– DeliveryTypeId – тип доставки, що є зовнішнім ключем таблиці DeliveryType;

– TotalPrice – сумарна вартість замовлення;

– DelivAddress – адреса доставки замовлення;

– OrderTime – час замовлення;

– Status – стан замовлення;

– StartTime – час, коли було відвантажено замовлення;

– DeliveryTime – час, коли має бути доставлено замовлення;

– DeliveredTime – час, коли було фактично доставлено замовлення;

– PredictedTime – час, до якого за прогнозом має бути доставлено замовлення;

– ClientComment – коментар щодо замовлення від клієнта;

– ManagerComment – коментар щодо замовлення менеджера.

Між таблицями User і Order зв'язок один до багатьох: один користувач може створити нуль, одне або декілька замовлень. Таким користувачем може виступати клієнт або касир.

Між таблицями CourierVehicle і Order зв'язок один до багатьох: один транспортний засіб з конкретним кур'єром може доставляти тимчасово нуль, одне або декілька замовлень.

Між таблицями DeliveryType і Order зв'язок один до багатьох: один тип доставки може використовуватися різними замовленнями, одне замовлення відповідно може доставлятися тільки одним типом.

Таблиця DeliveryType визначає тип доставки і включає такий набір стовпців:

– DeliveryTypeId – ідентифікатор типу доставки, що є первинним ключем таблиці;

– VehicleTypeId – тип транспортного засобу, що є зовнішнім ключем таблиці VehicleType;

– Title – назва типу доставки.

Між таблицями DeliveryType і VehicleType зв'язок один до багатьох: один тип доставки обслуговується тільки одним типом транспортних засобів, але один тип транспортного засобу може використовуватися для різних типів доставки.

Таблиця User визначає користувача ПЗ (кур'єра, клієнта, менеджера, касира) і включає такий набір стовпців:

- UserId – ідентифікатор облікового запису користувача, що є первинним ключем таблиці;
- FullName – повне ім'я користувача;
- Phone – номер телефону;
- Address – адреса, що має використовуватися як стандартна адреса доставки замовлення;
- Password – пароль;
- UserType – тип користувача (менеджер, кур'єр, клієнт).

Таблиця OrderProducts необхідна для реалізації зв'язку багато до багатьох між сутностями Order і BakeryProduct, тобто по суті визначає склад корзини замовлення і включає такий набір стовпців:

- OrderProductId – ідентифікатор складової замовлення, що є первинним ключем таблиці;
- OrderId – ідентифікатор замовлення, що є зовнішнім ключем таблиці Order;
- BakeryProductId – виріб пекарні, що є зовнішнім ключем таблиці BakeryProduct;
- Quantity – кількість даних виробів пекарні в замовленні;
- Price – вартість одного виробу.

Таблиця ProductPrice включає такий набір стовпців:

- ProductPriceId – ідентифікатор засобу про вартість виробу, що є первинним ключем таблиці;
- BakeryProductId – виріб пекарні, для якого встановлюється вартість, є зовнішнім ключем таблиці BakeryProduct;

- Price – вартість одного виробу;
- StartDate – дата, з якої встановлено дану вартість виробу;
- EndDate – дата, до якої встановлено дану вартість виробу.

Між таблицями BakeryProduct і ProductPrice зв'язок один до багатьох: одне найменування виробу пекарні може входити до складу різних замовлень, при цьому кожне замовлення може тільки один раз включати конкретне найменування виробу пекарні (для декількох екземплярів для відповідного найменування лише зазначається кількість).

Таблиця BakeryProduct включає такий набір стовпців:

- BakeryProductId – ідентифікатор виробу пекарні, що є первинним ключем таблиці;
- BakeryGroupId – група виробів, до якої належить даний, є зовнішнім ключем таблиці BakeryGroup;
- Title – назва виробу;
- Photo – фотографія виробу;
- Description – опис виробу.

Між таблицями BakeryGroup і BakeryProduct зв'язок один до багатьох: одна група виробів пекарні може включати від 0 до декількох конкретних виробів, а кожен виріб може входити тільки до однієї групи.

Оскільки одна пропозиція щодо придбання виробів пекарні може включати різні вироби, а кожне найменування виробів може входити до різних пропозицій, зв'язок між таблицями багато до багатьох. Для реалізації зв'язку використовується таблиця BakeryProductOffer з набором стовпців:

- OfferId – пропозиція про покупку виробів пекарні, що є зовнішнім ключем таблиці Offer;
- BakeryProductId – виріб пекарні, який входить до пропозиції, є зовнішнім ключем таблиці BakeryProduct;
- Quantity – кількість виробів, яку потрібно придбати для виконання пропозиції.

Схема БД приведена на рис. 3.6.

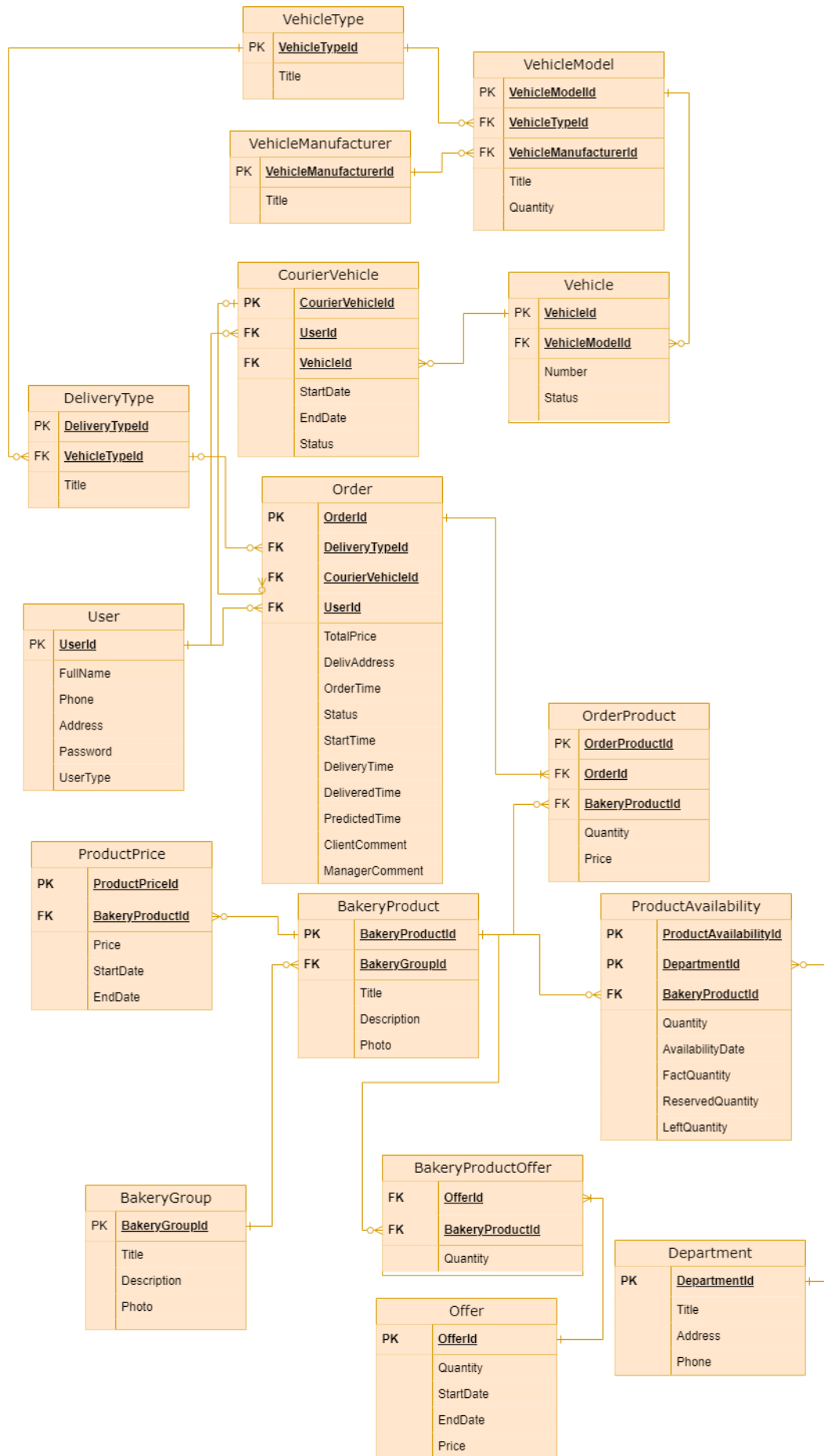


Рисунок 3.6 – Схема БД

Таблиця ProductAvailability включає такий набір стовпців:

- ProductAvailabilityId – ідентифікатор запису, що є первинним ключем таблиці;
- DepartmentId – пункт пекарні, що є зовнішнім ключем таблиці Department;
- BakeryProductId – виріб пекарні, що є зовнішнім ключем таблиці BakeryProduct;
- Quantity – кількість даних виробів, які має бути випечено на задане число в відділенні;
- AvailabilityDate – дата, якої стосується інформація про доступність виробу;
- FactQuantity – фактична випічена кількість виробів;
- ReservedQuantity – зарезервована кількість виробів (якщо вироби тільки заплановані, але ще не випічені);
- LeftQuantity – кількість виробів, яку ще не було продано.

Таблиця Department включає такий набір стовпців:

- DepartmentId – ідентифікатор пункту пекарні, що є первинним ключем таблиці;
- Title – назва пункту пекарні;
- Phone – номер телефону пункту пекарні;
- Address – адреса пункту пекарні.

Таблиця Offer включає такий набір стовпців:

- OfferId – ідентифікатор пропозиції про купівлю виробів пекарні, що є первинним ключем таблиці;
- Quantity – кількість доступних комплектів;
- Price – загальна вартість пропозиції;
- StartDate – дата початку дії пропозиції;
- EndDate – дата завершення дії пропозиції.

Таблиця BakeryGroup включає такий набір стовпців:

- BakeryGroupId – група виробів пекарні, що є первинним ключем таблиці;
- Title – назва групи виробів пекарні;
- Description – опис групи виробів пекарні;
- Photo – зображення групи виробів.

### **3.4 Висновки за розділом 3**

Обрано фреймворк веброзробки ASP.NET Core та мову програмування C# для реалізації програмного забезпечення. Створено структурну схему БД. Виконано моделювання сценаріїв роботи кур'єра, клієнта, адміністратора і касира з програмою.

## **4 ОСНОВНІ РІШЕННЯ ЩОДО РЕАЛІЗАЦІЇ КОМПОНЕНТІВ СИСТЕМИ**

### **4.1 Опис послідовності роботи з замовленнями на доставку**

Замовлення на доставку виробів пекарні підтримують організацію всіх процесів у межах роботи ПЗ окрім безпосередньо внесення базових даних, що необхідні для підтримки роботи з іншими сутностями БД, що входить до зони відповідальності адміністратора.

Спочатку будь-яке замовлення знаходиться в стані «створене» після того, як воно з'являється в системі. Після цього може бути одразу реалізовано покупку виробів з замовлення. Це відбувається у випадку, коли покупка виконується одразу в пункті пекарні. Тоді клієнт сам забирає замовлення і доставка фактично не потрібна. Касир сам вносить дані замовлення в систему. Замовлення після цього переходить до стану «придбано» і робота з ним на цьому завершується.

Одразу після створення (або без переходу в інший стан) замовлення може бути переведено до стану «скасоване». Перехід до такого стану відбувається, якщо замовлення не очікується клієнтом (він відмовляється від нього) або якщо замовлення не може бути виконано. Останній варіант може реалізуватися в тому випадку, якщо, наприклад, виробів було випічено менше ніж заплановано.

Якщо ж замовлення реалізувалось звичайним чином, то замовлення переходить до стану «відвантажено». У такому випадку вироби вже випічені і підготовлені кур'єру.

Зі стану «відвантажено» замовлення може стандартно перейти до стану «в доставці», якщо кур'єр забрав замовлення. Якщо ж замовлення було зіпсоване і не може бути виконано, то воно переводиться до стану «скасоване».

Якщо замовлення перебуває в стані «в доставці», то у випадку, коли клієнт забрав замовлення, воно переходить до стану «доставлене».

Якщо клієнт відмовився від замовлення, воно переходить до стану «відмовлено».

Якщо клієнт не зустрівся з кур'єром, не зважаючи на всі його спроби, то замовлення переходить до стану «не доставлене».

У результаті було створено діаграму станів замовлення (рис. 4.1).

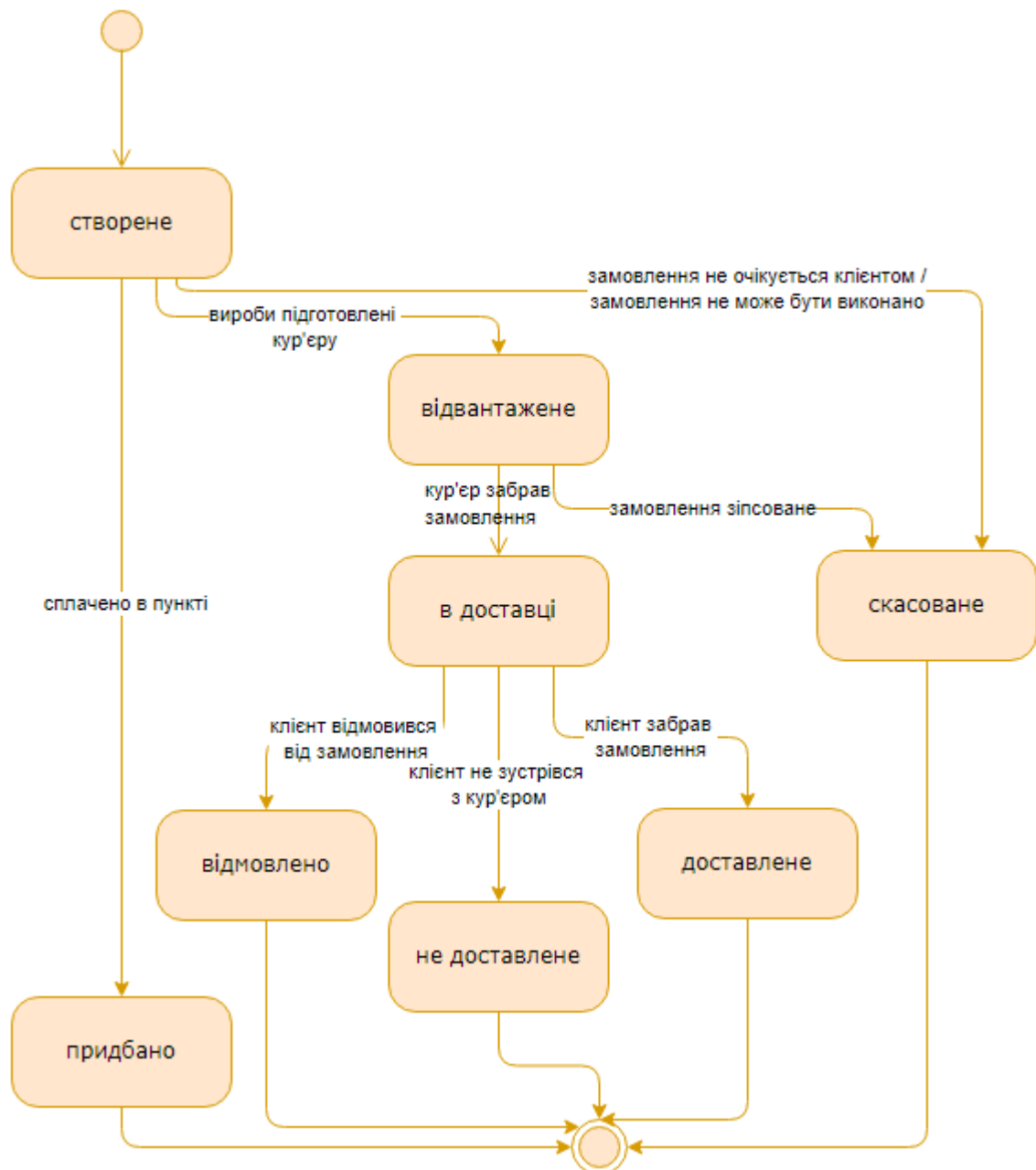


Рисунок 4.1 – Діаграма станів замовлення на доставку виробів пекарні

## 4.2 Опис модулів реалізованої програмної системи

Модуль BakeryProduct містить клас BakeryProduct, який представляє виріб пекарні і включає атрибути:

- public int BakeryProductId – ідентифікатор виробу пекарні;
- public string Title – назва виробу;
- public string Description – опис виробу;
- public string Photo – шлях до фотографії, яка представляє виріб;
- public int BakeryGroupId – номер групи виробів пекарні;
- public float Price – вартість виробу пекарні на поточний момент, яка визначається шляхом пошуку в БД і виділення вартості за період, який включає поточну дату;
- public Offer[] Offers – перелік поточних пропозицій про купівлю виробів, який містить даний виріб (повна інформація даних пропозицій);
- public virtual BakeryGroup BakeryGroup – група, до якої належить даний виріб (містить повну інформацію про групу з номером BakeryGroupId, об'єкт класу BakeryGroup), який заповнюється автоматично;
- public virtual ICollection<ProductPrice> ProductPrices – вартість виробу пкекарні за всі наявні дні (перелік, який заповнюється автоматично);
- public virtual ICollection<BakeryProductOffer> BakeryProductOffers – пропозиції про купівлю даного виробу пекарні за зменшеною ціною шляхом придбання набору виробів (перелік, який заповнюється автоматично);
- public virtual ICollection<ProductAvailability> ProductAvailabilities – інформація про доступність даного виробу за всі наявні дні (перелік, який заповнюється автоматично);
- public string PriceStr – текстове представлення вартості виробу пекарні, округленої до 2 знаків після коми.

Модуль BakeryProductController містить клас контролера виробів пекарні BakeryProductController, що реалізує засоби роботи, які стосуються виробів пекарні.

Метод `Index()` даного класу повертає представлення, заповнене даними 5 найдешевших виробів пекарні. Він використовується для підтримки базового засобу перенаправлення під час роботи різних інших методів.

Метод `Details(int? bprod)` повертає представлення, заповнене повними даними виробу пекарні з номером `bprod`.

Для реалізації даного методу спочатку реалізується виділення об'єкту виробу пекарні з заданим номером `bprod` з заповненням його ідентифікатора, повної інформації про групу виробу, назви виробу, опису виробу, фотографії виробу:

```
var bakprod = bpcontext.BakeryProducts.Include(bp =>
bp.BakeryProductId).Include(bp => bp.BakeryGroup).Include(bp =>
bp.Title).Include(bp => bp.Description).Include(bp =>
bp.Photo).FirstOrDefault(bp => bp.BakeryProductId == bprod);
```

Після цього реалізується визначення вартості даного виробу. Для цього аналізується весь набір записів про вартість саме даного виробу з ідентифікатором `bprod` з виділенням запису, проміжок часу в якому містить поточну дату:

```
var prices = bpcontext.ProductPrices.Include(pp =>
pp.BakeryProductId).Include(pp => Price).Include(pp => pp.StartDate).Include(pp
=> pp.EndDate).Where(pp => pp.StartDate <= nowdate && pp.EndDate >=
nowdate && pp.BakeryProductId == prod.BakeryProductId);
```

Після цього в методі реалізується пошук пропозицій про купівлю набору виробів, який містить даний виріб. Окрім того така пропозиція повинна включати в своєму часовому проміжку поточну дату, тобто прострочені пропозиції або пропозиції, строк дії яких ще не настав, не

враховуються. Кількість доступних комплектів за цією пропозицією повинна бути більше 0, інакше пропозиція вважається неактивною:

```
var offers = bpcontext.Offers.Include(bp => bp.StartDate).Include(bp => bp.EndDate).Include(bp => bp.Price).Include(bp => bp.Quantity).Where(bp => bp.Quantity > 0 && bp.StartDate <= nowdate && bp.EndDate >= nowdate);
```

У свою чергу при замовленні виробів, які формують відповідну пропозицію, дана кількість доступних комплектів зменшується на відповідне значення.

Саме в цьому методі на основі виділених пропозицій заповнюється атрибут класу BakeryProduct з назвою Offers, відповідно в результаті він містить набір активних пропозицій для даного виробу.

Метод CheckAvailability(int? prodid, int cdday, int cdmonth, int cdeyear) повертає кількість виробу з номером prodid, доступну на дату, що представлена днем cdday, місяцем cdmonth, роком cdeyear. Для цього формується інформація про даний виріб пекарні, заповнюється автоматично інформація про його доступність в атрибуті ProductAvailabilities. За даним атрибутом виділяється об'єкт, який відповідає заданій в якості параметра даті.

Далі, якщо фактична кількість в атрибуті FactQuantity нульова, тобто вироби ще не випічені, то відбувається перевірка того, що загальна кількість виробів Quantity більша за зарезервовану кількість ReservedQuantity. У тому випадку, якщо перевірка здійснена успішно, тобто вся запланована кількість не зарезервована, то різниця між цими показниками повертається в результуючому представленні.

У випадку, якщо фактична кількість виготовлених виробів заповнена, то береться значення з атрибута LeftQuantity, тобто доступна кількість виробів. Якщо кількість нульова, то повертається відповідне представлення.

Метод `AddAvailability(int bprod, int q, int depid, int cdday, int cdmonth, int cdyear)` вносить інформацію про доступність виробу з номером `bprod` у пункті `depid` на дату з номером дня місяця `cdday`, номером місяця `cdmonth`, номером року `cdyear`. Загальна кількість виробів встановлюється у значення `q`. Тобто спочатку вноситься інформація про доступність виробу на дату, а потім в цю позицію заноситься інформація про випічені вироби.

Метод `ChangeAvailability(int bprod, int q, int fq, int depid, int cdday, int cdmonth, int cdyear)` виконує редагування інформації про доступність виробу на задану дату у відділенні, також підтримуючи внесення даних про фактичну кількість випічених виробів. Якщо здійснюється тільки редагування з метою виправлення даних, то фактична кількість може задаватися нульовою.

Для реалізації даного методу спочатку виконується пошук запису про виріб з номером `bprod` у відділенні `depid` на дату з номером дня місяця `cdday`, номером місяця `cdmonth`, номером року `cdyear`, у результаті встановлюється загальна кількість виробів у `q`, а фактична кількість – у `fq`, а змінені дані фіксуються в БД.

Метод `GroupBakeryProducts(int? grid)` повертає представлення, заповнене виробами пекарні, що належать групі виробів з номером `grid`. При цьому за кожним виробом з заданої групи визначається актуальне значення вартості на поточну дату.

Метод `AllProducts()` повертає представлення, заповнене даними всіх виробів пекарні. Даний метод необхідний для створення представлення для підтримки подальшого редагування виробів у системі.

Метод `BakeryProductOffers(int? prodid)` виконує пошук пропозицій про придбання наборів виробів, до яких входить виріб з номером `prodid`. Цей перелік пропозицій повинен бути активним, тобто в строк дії пропозиції повинна входити поточна дата.

Метод `Create()` реалізує підтримку базового представлення для створення нового запису про виріб пекарні.

Метод `Create([Bind("BakeryGroupId, Title, Description, Photo")] BakeryProduct bprod)` реалізує представлення створення нового запису про виріб пекарні з передачею значень через форму методом POST: назва виробу `Title`, опис виробу `Description`, фотографія `Photo`, номер групи виробів `BakeryGroupId`.

Метод `Edit(int? bprod)` підтримує ініціалізацію представлення редагування даних виробу пекарні `bprod`. Для цього спочатку виконується перевірка наявності виробу з заданим номером, а далі заповнюється перелік груп виробів.

Метод `Edit(int bprod, [Bind("BakeryProductId, BakeryGroupId, Title, Description, Photo")] BakeryProduct bprod)` виконує редагування виробу пекарні з номером `bprod`, заповнюючи в якості значень виробу отриманих через форму користувацького інтерфейсу і передачу методом POST значень: назви виробу `Title`, опису виробу `Description`, фотографії `Photo`, номеру групи виробів `BakeryGroupId`.

Метод `AddPrice(int bprod, int price, int sdday, int sdmonth, int sdyear, int edday, int edmonth, int edyear)` вносить інформацію про вартість виробу з номером `bprod` протягом періоду, починаючи з дати з номером дня місяця `sdday`, номером місяця `sdmonth`, номером року `sdyear` і завершуючи датою з номером дня місяця `edday`, номером місяця `edmonth` і номером року `edyear`. Вартість встановлюється у значення `price`.

Для реалізації даного методу спочатку реалізується перевірка, чи є в БД запис про вартість даного виробу в період, що включає частково або повністю заданий період часу.

Метод `ChangePrice(int bprod, int price, int sdday, int sdmonth, int sdyear, int edday, int edmonth, int edyear)` виконує редагування інформації про вартість виробу з номером `bprod` протягом періоду, починаючи з дати з номером дня місяця `sdday`, номером місяця `sdmonth`, номером року `sdyear` і завершуючи датою з номером дня місяця `edday`, номером місяця `edmonth` і номером року `edyear`. Вартість встановлюється у значення `price`.

Даний метод забезпечує змінювання вартості, яке реалізується не за номером запису про вартість, а шляхом пошуку запису, що включає заданий період.

Аналогічно організовано роботу інших моделей та контролерів, які підтримують взаємодію між моделями та представленнями.

Реалізацію дерев рішень виконано на основі застосування бібліотеки XGBoost з використанням обгортки [16].

Представлення відображення виробів за обраною групою реалізовано за допомогою шаблону [17] наступним чином:

```
@foreach (var item in Model) {
<div class="post-slide">
    <div class="post-img">
        <div class="post-abs"><p>@Html.DisplayFor(mitem =>
item.Title)</p></div>
        
    </div>
    <h3 class="post-title"><a href="#">@Html.DisplayFor(mitem =>
item.Title)</a></h3>
    <p class="post-description">
        @Html.DisplayFor(mitem => item.Price)
    </p>
</div>
}
```

### 4.3 Висновки за розділом 4

Розглянуто послідовність роботи програми над замовленнями з доставки виробів пекарні, в результаті чого створено діаграму станів замовлення.

Реалізацію вебзастосунку виконано шляхом використання модуля об'єктно-реляційного відображення Entity Framework Core, що дозволило, визначаючи класи моделей вебзастосунку, одночасно визначити таблиці БД і відношеннями між ними. Описано перелік і логіку реалізацію методів класів контролерів. Представлено логіку організації представлень вебзастосунку.

## **5 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ**

### **5.1 Призначення програми**

Програма призначена для підтримки доставки виробів пекарні містом на основі забезпечення інформаційної і методологічної підтримки даного процесу. Програма може використовуватися як клієнтами, так і працівниками пекарні і її служби доставки, включаючи кур'єрів, менеджерів та, за необхідності, касирів.

### **5.2 Умови виконання програми**

Для виконання програми потрібно забезпечити наступний рівень апаратних ресурсів сервера:

- процесор: щонайменше два ядра, мінімальна частота – 2 ГГц;
- оперативний запам'ятовуючий пристрій – не менше 4 Гб;
- жорсткий диск: вільний простір не менше 1 Гб.

### **5.3 Виконання програми**

Взаємодія користувачів всіх типів відбувається через інтерфейс вебзастосунку з доступом через мережу інтернет. Спочатку взаємодія відбувається в режимі клієнта, при цьому клієнту доступні всі функціональні можливості, але для того, щоб ними скористатися, клієнта буде направлено на сторінку авторизації.

Клієнту доступні наступні варіанти роботи через меню.

Пункт меню «Смачненьке» дозволяє переглянути перелік груп виробів пекарні. Після цього користувач може обрати потрібну йому групу і перейти до перегляду виробів, що належать до даної групи (рис. 5.1).

ВН ОБРАЛИ ГРУПУ ВИРОБІВ

## Круасани



Класичний круасан  
**20₴**



М'ясний круасан  
**35₴**



Абрикосовий круасан  
**25₴**



Шоколадний круасан  
**27₴**



Горіховий круасан  
**31₴**



Полуничний круасан  
**25₴**

Рисунок 5.1 – Перегляд виробів за групою виробів пекарні

За кожним виробом виводиться його фотографія, назва та вартість. Для перегляду повної інформації про виріб або його замовлення необхідно натиснути на назву або зображення відповідного виробу.

На сторінці виводиться по 6 виробів. У випадку, якщо виробів у групі більше, то нижче виводяться номери сторінок, за якими можна перейти до перегляду наступного набору виробів пекарні цієї групи.

На сторінці перегляду інформації про конкретний виріб пекарні (рис. 5.2) виводиться безпосередньо фотографія виробу, його назва, помаранчевим кольором над назвою зазначається група виробу, а нижче назви – вартість за одиницю. Ще нижче розташовано опис виробу.

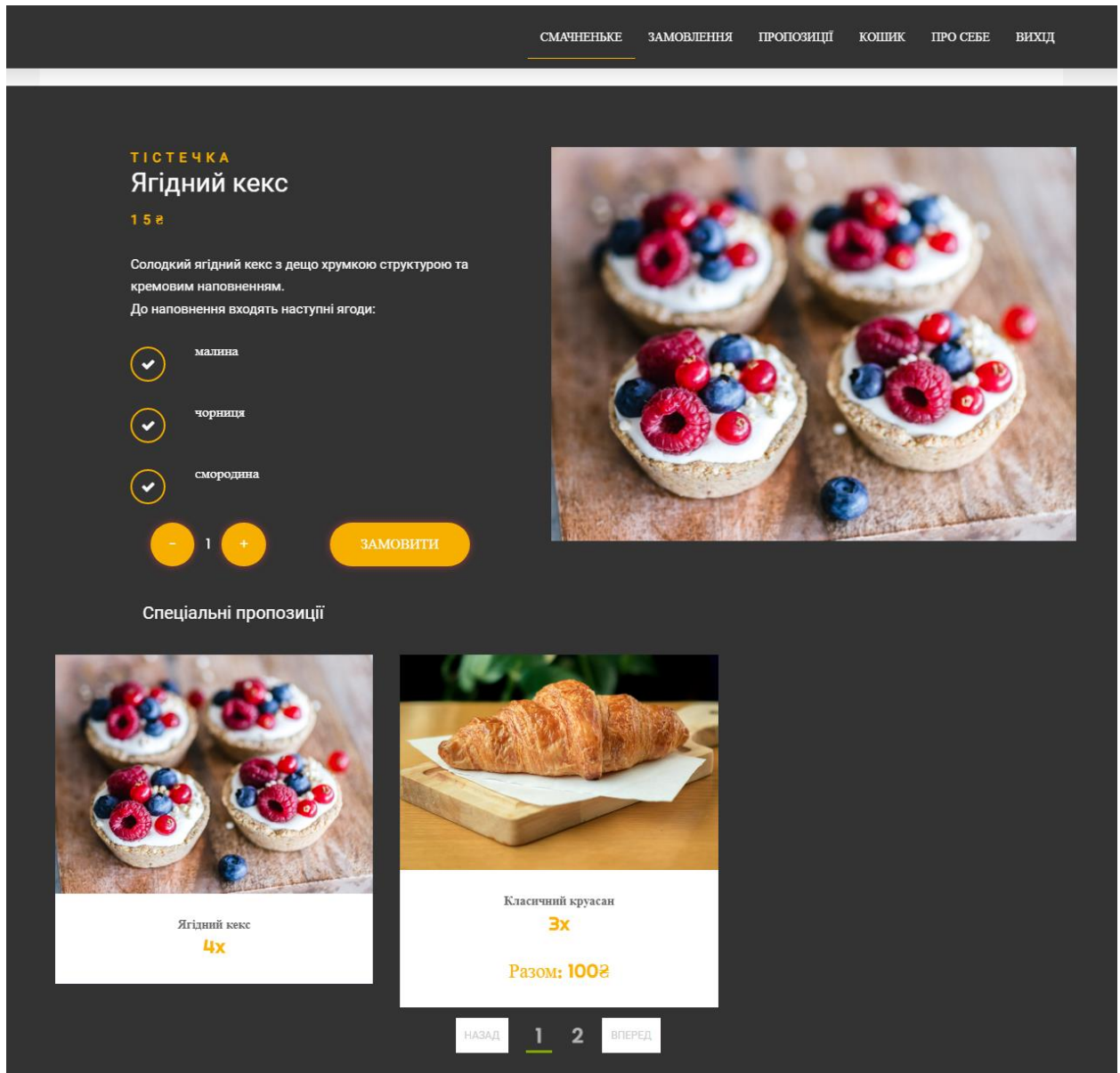


Рисунок 5.2 – Перегляд інформації про виріб клієнтом

Під описом доступна можливість визначення кількості виробів, які необхідно замовити. Для цього потрібно кнопками «-» і «+» визначити відповідну кількість виробів. Після цього необхідно натиснути на кнопку «Замовити», що дозволить помістити обраний виріб у кошик.

Нижче на сторінці розташовані спеціальні пропозиції, до яких входить виріб, розташований на даній сторінці. Кожна пропозиція виводиться переліком виробів з зазначенням кількості виробів, яку потрібно придбати для купівлі замовлення за зниженою ціною. В останній позиції зазначається загальна вартість пропозиції у такому випадку.

Якщо пропозицій декілька, то нижче виводяться номери пропозицій, за допомогою яких можна перемикатися між ними. Якщо в пропозиції виробів більше 3, то між ними також можна перемикатися за допомогою кнопок «<» та «>», які з'являються в такому випадку.

Усі вироби додаються в кошик, формуючи поточне замовлення. Щоб перейти до кошику, потрібно обрати пункт меню «Кошик». Саме з кошика потрібно задати пункт, з якого клієнт бажає виконати доставку. Також звідси доступний вибір дати, на яку має бути виконане замовлення. Після вибору пункту та дати актуалізується доступна кількість виробів. Якщо обрана кількість не доступна, то виводиться коротке повідомлення нижче відповідної позиції в замовленні.

З пункту меню «Замовлення» клієнту доступне керування замовленнями клієнта. У випадку якщо користувач не авторизований, він буде запрошений до авторизації. Звідси клієнт зокрема може скасувати замовлення, якщо воно було тільки створене.

З пункту меню «Пропозиції» доступний перелік всіх наявних пропозицій про придбання наборів виробів за зниженою вартістю. Тобто на цій сторінці пропозиції вже не будуть зв'язуватися з конкретним виробом.

З пункту меню «Про себе» можна змінити дані облікового запису або переглянути їх.

Якщо в застосунку авторизувався менеджер, то йому доступний варіант меню з великою кількістю пунктів.

Пункт меню «Групи» дозволяє переглядати групи виробів менеджеру, переходити до конкретних виробів. Менеджер не може змінювати вироби, ця функція доступна тільки адміністратору.

Пункт меню «Вартість» дозволяє встановлювати вартість виробів на конкретні дати.

Пункт меню «План» дозволяє вносити дані про заплановану до продажу на наступні дні кількість виробів. Окрім того звідси можна вносити також фактично виготовлену кількість виробів.

Пункт меню «Замовлення» (рис. 5.3) дозволяє переглядати і після того керувати замовленнями менеджеру.

ГРУПИ ВАРТІСТЬ ПЛАН **ЗАМОВЛЕННЯ** ТРАНСПОРТ ПРОПОЗИЦІЇ КОШИК ПРО СЕБЕ ВИХІД

СТВОРИТИ НОВЕ ЗАМОВЛЕННЯ →

**ОФОРМЛЕНІ ЗАМОВЛЕННЯ**

Номер

Пошук за номером замовлення

Параметр пошуку

Дати

Початкова дата

03.12.2021

Кінцева дата

04.12.2021

ШУКАТИ

| Номер | Вартість | Стан         | Створення            | Прогноз              |
|-------|----------|--------------|----------------------|----------------------|
| 31    | 300₴     | Створене     | 04 грудня 2021 11:15 | 04 грудня 2021 11:43 |
| 30    | 215₴     | Створене     | 04 грудня 2021 11:13 | 04 грудня 2021 11:57 |
| 29    | 35₴      | Створене     | 04 грудня 2021 11:12 | 04 грудня 2021 12:03 |
| 28    | 210.2₴   | Відвантажене | 04 грудня 2021 11:06 | 04 грудня 2021 11:53 |
| 27    | 77₴      | Відвантажене | 04 грудня 2021 11:03 | 04 грудня 2021 11:51 |
| 26    | 143.5₴   | В доставці   | 04 грудня 2021 10:58 | 04 грудня 2021 11:39 |
| 25    | 89₴      | В доставці   | 04 грудня 2021 10:56 | 04 грудня 2021 11:42 |
| 24    | 124₴     | В доставці   | 04 грудня 2021 10:53 | 04 грудня 2021 11:32 |

Рисунок 5.3 – Перегляд замовлень клієнтів менеджером

При перегляді замовлень доступні одразу засоби пошуку цих замовлень. Менеджер може ввести конкретний номер замовлення. Окрім того менеджер може обрати варіант параметрів пошуку.

Доступний варіант пошуку «Дати», який дозволяє задати проміжок дат створення замовлень, які будуть відображені нижче в таблиці. При цьому з'являються два відповідні поля для визначення дат.

Також доступні варіанти «Усі» для відображення всіх замовлень, «Кур'єр» для визначення кур'єра (обирається з переліку з можливістю

введення), який виконує доставку замовлення, «Відвантажені» для відображення відвантажених замовлень, «Прострочені» для відображення замовлень, дата постачання яких прострочена, «Створені» для відображення ще не оброблених замовлень.

Після того, як менеджер визначив параметри пошуку, потрібно натиснути на кнопку «Шукати». Внизу буде відображено перелік замовлень, що відповідають заданим параметрам.

Усі замовлення відображаються з визначенням номеру замовлення, його вартості, стану, дати і часу створення та прогнозованої дати і часу доставки. При натисканні на будь-яку позицію в рядку замовлення відкривається сторінка роботи з цим замовленням. Звідси доступна повна інформація про замовлення. Окрім того менеджер з цієї сторінки може визначати кур'єра для замовлення, змінювати стан замовлення.

Менеджеру через пункт меню «Транспорт» доступний перелік транспортних засобів служби доставки. Звідси можна їх переглядати, прив'язувати до конкретного кур'єра.

Через пункт меню «Пропозиції» менеджер може не тільки переглянути пропозиції про придбання наборів товарів, але і безпосередньо створити такі пропозиції.

Менеджеру також надано можливість створення замовлення. Вона доступна через кнопку «Створити замовлення» зі сторінки перегляду замовлень, а також шляхом внесення виробів зі сторінок конкретних виробів у кошик.

Кур'єру доступне обмежене меню, що складається з пункту «Замовлення», звідки кур'єр може переглянути всі замовлення, які він має доставити, а також визначити результат доставки та отриманні замовлення на доставку, та пункту меню «Транспорт», звідки кур'єр може змінювати стан власного транспортного засобу.

Касиру доступне меню з пунктів «Замовлення», звідки можна тільки внести замовлення, обираючи з переліку потрібну групу виробу, а потім

конкретний виріб, а також задаючи потрібну кількість виробів, «Групи» (аналогічно до роботи менеджера), «Пропозиції», «Про себе».

Адміністратору для керування, тобто відображення відповідного переліку з можливістю редагування даних і створення нових записів, доступні пункти меню: «Вироби», «Групи», «Замовлення», «Транспорт», звідки можна зокрема визначати і виробників, і моделі транспортних засобів, «Доставки», звідки можна визначити типи доставки.

#### **5.4 Експериментальне дослідження проблеми прогнозування часу доставки виробів пекарні**

Для експериментального дослідження було використано вибірку [18], що характеризує доставку на основі сукупності наступних параметрів:

- номер служби доставки;
- номер міста;
- час початку доставки;
- модель транспортного засобу;
- кількість одиниць виробів у замовленні;
- абсциса точки відправлення замовлення;
- ордината точки відправлення замовлення;
- найкоротша відстань між точками початку доставки і цільовою точкою, зазначена в метрах (визначається двома окремими способами і зазначається у вибірці двічі);
  - напрямок руху від початкової точки до кінцевої ( $90^\circ$  – схід,  $-90^\circ$  – захід,  $-180^\circ$  або  $+180^\circ$  – північ або південь);
  - температура повітря ( $^\circ\text{C}$ );
  - температура точки роси;
  - вологість повітря;
  - напрямок вітру;
  - швидкість вітру (км/г);

- видимість (км);
- тиск атмосферного повітря;
- погодні умови;
- наявність снігу;
- наявність туману;
- наявність дощу;
- наявність граду;
- час доставки.

З цієї вибірки було сформовано вибірку, необхідну для роботи методу прогнозування часу доставки виробів пекарні.

Для виконання прогнозування окрім методу на основі XGBoost було використано інші відомі методи прогнозування:

- метод опорних векторів;
- логістична регресія.

Отримані на даній вибірці результати представлено в табл. 5.1.

Таблиця 5.1 – Порівняння точності прогнозування за результатами експериментів

| Модель                                               | Точність прогнозування |
|------------------------------------------------------|------------------------|
| Регресія за методом опорних векторів                 | 0,372                  |
| Логістична регресія                                  | 0,317                  |
| Дерева рішень XGBoost (представлений у роботі метод) | 0,198                  |

Для оцінювання точності прогнозування використовувалась середньоквадратична похибка. Отримані результати показують, що метод, який представлений в роботі та який ґрунтується на деревах рішень і методі навчання XGBoost, призвів до отримання найкращої точності класифікації.

Точність даного методу виявилась на 37,5 % вищою за точність логістичної регресії.

### **5.5 Висновки за розділом 5**

Робота з програмою реалізується 4 основними типами користувачів: менеджером, клієнтом, кур'єром, адміністратором, а також касиром, для якого забезпечено підтримку внесення даних про виконане замовлення без доставки. Для кожного з цих типів користувачів створено відповідні графічні інтерфейси.

Проведено експериментальне дослідження прогнозування часу доставки. Представлений у роботі методі підтвердив свою здатність до вирішення проблем такого типу.

## ВИСНОВКИ

За результатами огляду програмних засобів підтримки замовлень у пекарні було виділено основні функціональні можливості даних програмних засобів і визначено функціональні вимоги до програмної розробки. Особливістю програми є надання можливості прогнозування часу доставки виробів пекарні.

Виконано аналіз підходу до прогнозування на основі дерев рішень з виділенням основних використовуваних на даний момент технік, що застосовуються для навчання дерев рішень. Виконано порівняння алгоритмів навчання дерев рішень. Представлено власний метод прогнозування часу доставки виробів пекарні.

Обрано фреймворк веброзробки ASP.NET Core та мову програмування C# для реалізації програмного забезпечення. Створено структурну схему БД.

Виконано програмну реалізацію вебзастосунку замовлення доставки виробів пекарні. Реалізацію вебзастосунку виконано шляхом використання модуля об'єктно-реляційного відображення Entity Framework Core, що дозволило, визначаючи класи моделей вебзастосунку, одночасно визначити таблиці БД і відношеннями між ними. Описано перелік і логіку реалізацію методів класів контролерів. Представлено логіку організації представлень вебзастосунку. Розглянуто послідовність роботи програми над замовленнями з доставки виробів пекарні, в результаті чого створено діаграму станів замовлення.

Робота з програмою реалізується 4 основними типами користувачів: менеджером, клієнтом, кур'єром, адміністратором, а також касиром, для якого забезпечено підтримку внесення даних про виконане замовлення без доставки. Для кожного з цих типів користувачів створено відповідні графічні інтерфейси.

Проведено експериментальне дослідження прогнозування часу доставки. Представлений у роботі методі підтвердив свою здатність до вирішення проблеми.

Наукова новизна роботи полягає в запропонованому методі прогнозування часу доставки виробів пекарні, який використовує ансамбль дерев рішень з навчанням на основі градієнтного бустинга, що дозволяє визначати час доставки замовлень виробів пекарні за набором їх характеристик.

Завдання роботи виконані повністю.

## ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Кондитерська-пекарня Baker Street Bakery [Електрон. ресурс]. – Режим доступу : <https://www.bakerstreetbakery.com.ua/>.
2. ШОСо. – пекарня-кондитерська у Львові [Електрон. ресурс]. – Режим доступу : <https://shoco.ua/>.
3. Comptoir Gourmand. Bakery. Cafe. London [Electronic resource]. – Access mode : <https://www.comptoirgourmand.co.uk/>.
4. Zaimon, O. Z. Data Mining with Decision Trees [Text] : Theory and Applications / O. Z. Zaimon, L. Rokach. – London : WSPC, 2014. – 330 p.
5. Biehler, R. Introducing students to machine learning with decision trees using CODAP and Jupyter Notebooks [Text] / Rolf Biehler, Yannik Fleischer // Teaching Statistics. – 2021. – Volume 43, Issue S1. – Pp. 133-142.
6. Vadapalli, P. Pros and Cons of Decision Tree Regression in Machine Learning [Electronic resource] / P. Vapadalli. – Access mode : <https://www.upgrad.com/blog/pros-and-cons-of-decision-tree-regression-in-machine-learning/>
7. Sheppard, C. Tree-based Machine Learning Algorithms [Text] : Decision Trees, Random Forests, and Boosting / C. Sheppard. – Scotts Valley, California : CreateSpace Independent Publishing Platform, 2017. – 110 p.
8. Understanding XGBoost Algorithm. What is XGBoost Algorithm? [Electronic resource]. – Access mode : <https://www.mygreatlearning.com/blog/xgboost-algorithm/>.
9. Molnar, Ch. Interpretable Machine Learning : A Guide for Making Black Box Models Explainable [Electronic resource] / Ch. Molnar. – Access mode : <https://christophm.github.io/interpretable-ml-book/>.
10. An End-to-End Guide to Understand the Math behind XGBoost [Electronic resource]. – Access mode : <https://www.analyticsvidhya.com/blog/2018/09/an-end-to-end-guide-to-understand-the-math-behind-xgboost/>.

11. The web framework for perfectionists with deadlines. Django [Electronic resource]. – Access mode : <https://www.djangoproject.com/>.
12. ASP.NET Core. Полное руководство [Электрон. ресурс]. – Режим доступа : <https://metanit.com/sharp/aspnet5/>.
13. Документация по ASP.NET. Microsoft Docs [Электрон. ресурс]. – Режим доступа : <https://docs.microsoft.com/ru-ru/aspnet/core/?view=aspnetcore-6.0>.
14. Lock, A. ASP.NET Core in Action [Text] / A. Lock. – New York : Manning, 2021. – 832 p.
15. De Sanctis, V. ASP.NET Core 5 and Angular [Text] : Full-stack web development with .NET 5 and Angular 11 / V. De Sanctis. – Birmingham : Packt Publishing, 2021. – 746 p.
16. PicNet/XGBoost.Net [Electronic resource] : .Net wrappers for the awesome XGBoost library. – Access mode : <https://github.com/PicNet/XGBoost.Net>.
17. Bakery Free Website Template. Free CSS Templates. Free CSS [Electronic resource]. – Access mode : <https://www.free-css.com/free-css-templates/page273/bakery>.
18. Package Delivery Time. Kaggle [Electronic resource]. – Access mode : <https://www.kaggle.com/c/package-delivery-time/data>.

**ДОДАТОК А**  
**Технічне завдання**

## **Вступ**

Забезпечення доставки виробів пекарень направлене на те, щоб зробити життя людей зручнішим. Важливо не тільки розробити ПЗ підтримки даного процесу, але і виконати прогнозування тривалості доставки, адже розуміння того, як буде працювати доставка для конкретного замовлення, є важливим для підтримки доставки виробів пекарні.

### **A.1 Підстави для розробки**

Завдання на дипломну кваліфікаційну роботу магістра визначене за темою “Дослідження та програмна реалізація методів прогнозування часу доставки виробів пекарні” і затверджене наказом № 435 від 5 листопада 2021 р. за Національним університетом «Запорізька політехніка».

### **A.2 Призначення розробки**

Розробка призначена для підтримки доставки виробів пекарні містом на основі забезпечення інформаційної і методологічної підтримки даного процесу. Розробка може використовуватися як клієнтами, так і працівниками пекарні і її служби доставки, включаючи кур’єрів, менеджерів та, за необхідності, касирів.

### **A.3 Основні вимоги до програми**

#### **A.3.1 Вимоги до функціональних характеристик**

Розробка повинна забезпечувати виконання:

- замовлення доставки виробів пекарні;
- відображення виробів пекарні за групами;
- відображення груп виробів пекарні;

- відображення замовлень, які потребують відвантаження;
- керування замовленнями;
- скасування замовлення;
- відображення прострочених замовлень;
- відображення замовлень за період;
- вибір транспортного засобу для доставки;
- відображення замовлень для обслуговування кур'єру;
- відображення власних замовлень;
- обчислення вартості замовлення;
- внесення випічених виробів;
- внесення запланованих до випікання обсягів виробів;
- визначення доставки замовлення;
- визначення кур'єра для доставки;
- прогнозування тривалості доставки;
- внесення нових виробів пекарні;
- визначення стану транспортного засобу служби доставки;
- створення нових груп виробів пекарні;
- додавання виробів пекарні до груп;
- редагування існуючих записів про вироби пекарні, групи виробів;
- внесення виробників транспортних засобів служби доставки, самих транспортних засобів, їх типів і моделей, відображення та редагування існуючих;
- зв'язування кур'єра з транспортним засобом;
- створення спеціальних пропозицій щодо купівлі наборів виробів;
- визначення вартості виробів пекарні на конкретний період часу;
- визначення транспорту служби доставки;
- відображення даних про виріб.

### **A.3.2 Вимоги до надійності**

Для забезпечення надійного функціонування програмного забезпечення необхідна постійна підтримка віддалених запитів користувачів як через локальну мережу, так і через мережу Інтернет. Для цього програма має розташовуватися на виділеному для цього сервері, який повинен функціонувати постійно і обробляти запити до вебсервера та сервера БД.

### **A.3.3 Вимоги до складу та параметрів технічних засобів**

Технічні засоби, необхідні для підтримки роботи програми, повинні відповідати наступним вимогам:

- процесор: щонайменше два ядра, мінімальна частота – 2 ГГц;
- оперативний запам'ятовуючий пристрій – не менше 4 Гб;
- жорсткий диск: вільний простір не менше 1 Гб.

### **A.4 Порядок контролю та приймання**

Контроль виконання роботи здійснюється за календарним планом роботи. Під час приймання роботи здійснюється перевірка відповідності визначеним функціональним характеристикам програми.

**ДОДАТОК Б**  
**Текст програми**

## Б.1 Текст файла Order.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;

namespace FedoBakery.Models
{
    public class Order
    {
        [Key]
        public int OrderId
        {get; set;}
        public string DelivAddress
        {get; set;}
        [Required]
        [DataType(DataType.Date)]
        public DateTime OrderTime
        {get; set;}
        [Required]
        public byte Status
        {get; set;}
        [DataType(DataType.Date)]
        public DateTime DeliveryTime
        {get; set;}
        [DataType(DataType.Date)]
        public DateTime DeliveredTime
        {get; set;}
        [DataType(DataType.Date)]
        public DateTime PredictedTime
        {get; set;}
        public string ClientComment
        {get; set;}
        public string ManagerComment
        {get; set;}
        [Required]
        public float TotalPrice
        {get; set;}
        public int DeliveryTypeId
        {get; set;}
        [Required]
        public int UserId
```

```

    {get; set;}
    public int CourierVehicleId
    {get; set;}
    public virtual DeliveryType DeliveryType
    {get; set;}
    public virtual User User
    {get; set;}
    public virtual CourierVehicle CourierVehicle
    {get; set;}
    public virtual ICollection<OrderProduct> OrderProducts
    {get; set;}
    [NotMapped]
    public string DeliveryTimeStr
    {
        get
        {
            return this.DeliveryTime.ToString("dd MMMM yyyy
H:mm");
        }
    }
    [NotMapped]
    public string OrderTimeStr
    {
        get
        {
            return this.OrderTime.ToString("dd MMMM yyyy
H:mm");
        }
    }
    [NotMapped]
    public string DeliveredTimeStr
    {
        get
        {
            return this.DeliveredTime.ToString("dd MMMM yyyy
H:mm");
        }
    }
    [NotMapped]
    public string PredictedTimeStr
    {
        get
        {

```

```

        return this.PredictedTime.ToString("dd MMMM yyyy
H:mm");
    }
}
[NotMapped]
public string StatusStr
{
    get
    {
        switch (this.Status)
        {
            case 0:
                return "створене";
            case 1:
                return "відвантажене";
            case 2:
                return "в достаці";
            case 3:
                return "доставлено";
            case 4:
                return "скасовано";
            case 5:
                return "відмова";
            case 6:
                return "не доставлено";
            case 7:
                return "сплачено";
            default:
                return "не встановлено";
        }
    }
}
[NotMapped]
public string TotalPriceStr
{
    get
    {
        if (this.TotalPrice - Math.Floor(this.TotalPrice)
< 0.01 )
        {
            return Math.Floor(this.TotalPrice).ToString();
        }
        else

```

```

        {
            if (this.TotalPrice -
Math.Round(this.TotalPrice, 1) < 0.01 )
            {
                return Math.Round(this.TotalPrice,
1).ToString();
            }
            else
            {
                return Math.Round(this.TotalPrice,
2).ToString();
            }
        }
    }
}
}
}
}

```

## Б.2 Текст файла BakeryProduct.cs

```

using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;

namespace FedoBakery.Models
{
    public class BakeryProduct
    {
        [Key]
        public int BakeryProductId
        {get; set;}
        [Required]
        public string Title
        {get; set;}
        public string Description
        {get; set;}
        public string Photo
        {get; set;}
        [Required]
        public int BakeryGroupId
        {get; set;}
        [NotMapped]

```

```

        public float Price
        {get; set;}
        [NotMapped]
        public Offer[] Offers
        {get; set;}
        public virtual BakeryGroup BakeryGroup
        {get; set;}
        public virtual ICollection<ProductPrice> ProductPrices
        {get; set;}
        public virtual ICollection<BakeryProductOffer>
BakeryProductOffers
        {get; set;}
        public virtual ICollection<ProductAvailability>
ProductAvailabilities
        {get; set;}
        [NotMapped]
        public string PriceStr
        {
            get
            {
                if (this.TotalPrice - Math.Floor(this.TotalPrice)
< 0.01 )
                {
                    return Math.Floor(this.TotalPrice).ToString();
                }
                else
                {
                    if (this.TotalPrice -
Math.Round(this.TotalPrice, 1) < 0.01 )
                    {
                        return Math.Round(this.TotalPrice,
1).ToString();
                    }
                    else
                    {
                        return Math.Round(this.TotalPrice,
2).ToString();
                    }
                }
            }
        }
    }
}

```

### Б.3 Текст файла ProductAvailability.cs

```

using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
namespace FedoBakery.Models
{
    public class ProductAvailability
    {
        [Key]
        public int ProductAvailabilityId
        {get; set;}
        public int Quantity
        {get; set;}
        public int FactQuantity
        {get; set;}
        public int LeftQuantity
        {get; set;}
        public int ReservedQuantity
        {get; set;}
        [DataType(DataType.Date)]
        public DateTime AvailabilityDate
        {get; set;}
        public int DepartmentId
        {get; set;}
        public int BakeryProductId
        {get; set;}
        [NotMapped]
        public string AvailabilityDateStr
        {
            get
            {
                return this.AvailabilityDate.ToString("dd MMMM yyyy
H:mm");
            }
        }
        public virtual Department Department
        {get; set;}
        public virtual BakeryProduct BakeryProduct
        {get; set;}
    }
}

```

#### Б.4 Текст файла Offer.cs

```

using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
namespace FedoBakery.Models
{
    public class Offer
    {
        [Key]
        public int OfferId
        {get; set;}
        public int Quantity
        {get; set;}
        public float Price
        {get; set;}
        [DataType(DataType.Date)]
        public DateTime StartDate
        {get; set;}
        [DataType(DataType.Date)]
        public DateTime EndDate
        {get; set;}
        public virtual ICollection<BakeryProductOffer>
BakeryProductOffers
        {get; set;}

        [NotMapped]
        public string StartDateStr
        {
            get
            {
                return this.StartDate.ToString("dd MMMM yyyy
H:mm");
            }
        }
        [NotMapped]
        public string EndDateStr
        {
            get
            {
                return this.EndDate.ToString("dd MMMM yyyy H:mm");
            }
        }
    }
}

```

```

    }
    [NotMapped]
    public string PriceStr
    {
        get
        {
            if (this.Price - Math.Floor(this.Price) < 0.01 )
            {
                return Math.Floor(this.Price).ToString();
            }
            else
            {
                if (this.Price - Math.Round(this.Price, 1) <
0.01 )
                {
                    return Math.Round(this.Price,
1).ToString();
                }
                else
                {
                    return Math.Round(this.Price,
2).ToString();
                }
            }
        }
    }
}

```

## Б.5 Текст файла BakeryProductController.cs

```

using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.Rendering;
using Microsoft.EntityFrameworkCore;
using FedoBakery.Data;
using FedoBakery.Models;
using System;
using System.Collections.Generic;
using System.Globalization;
using System.Linq;

```

```

namespace FedoBakery.Controllers
{
    public class BakeryProductController : Controller
    {
        private readonly FedoBakeryDbContext bpcontext;

        public AdvertisementController (FedoBakeryDbContext bpc)
        {
            bpcontext = bpc;
        }

        public ActionResult Index()
        {
            var products = bpcontext.BakeryProducts.Include(bp =>
bp.BakeryProductId).Include(bp => bp.BakeryGroup).Include(bp =>
bp.Title).Include(bp => bp.Description).Include(bp => bp.Photo).Where(bp
=> bp.BakeryGroup.GroupId == grid);
            foreach (var prod in products)
            {
                var prices = bpcontext.ProductPrices.Include(pp =>
pp.BakeryProductId).Include(pp => Price).Include(pp =>
pp.StartDate).Include(pp => pp.EndDate).Where(pp => pp.StartDate <=
nowdate && pp.EndDate >= nowdate && pp.BakeryProductId ==
prod.BakeryProductId);
                var price = prices.FirstOrDefault();
                prod.Price = price;
            }
            products.OrderBy(bp => bp.Price);
            var newarr = products;
            Array.Copy(products, 0, newarr, 0, 5);
            return View(newarr);
        }

        public ActionResult Details(int? bprod)
        {
            DateTime nowdate = DateTime.Now;
            var bakprod = bpcontext.BakeryProducts.Include(bp =>
bp.BakeryProductId).Include(bp => bp.BakeryGroup).Include(bp =>
bp.Title).Include(bp => bp.Description).Include(bp =>
bp.Photo).FirstOrDefault(bp => bp.BakeryProductId == bprod);
            if (bakprod == null)
            {
                return NotFound();
            }
        }
    }
}

```

```

    }
    var prices = bpcontext.ProductPrices.Include(pp =>
pp.BakeryProductId).Include(pp => Price).Include(pp =>
pp.StartDate).Include(pp => pp.EndDate).Where(pp => pp.StartDate <=
nowdate && pp.EndDate >= nowdate && pp.BakeryProductId ==
prod.BakeryProductId);
    var price = prices.FirstOrDefault();
    bakprod.Price = price;
    var offers = bpcontext.Offers.Include(bp =>
bp.StartDate).Include(bp => bp.EndDate).Include(bp =>
bp.Price).Include(bp => bp.Quantity).Where(bp => bp.Quantity > 0 &&
bp.StartDate <= nowdate && bp.EndDate >= nowdate);
    foreach (var off in offers)
    {
        bool bpo = false;
        foreach (var of in off.BakeryProducOffers)
        {
            if (of.BakeryProductId == bprod)
            {
                bpo = true;
            }
        }
        if (!bpo)
        {
            offers.Remove(off);
        }
    }
    int offsize = offers.Count();
    bakprod.Offers = new Offer[offsize];
    int i = 0;
    foreach (var off in offers)
    {
        bakprod.Offers[i] = off;
        i++;
    }
    return View(bakprod);
}

```

```

[Authorize]
public ActionResult CheckAvailability(int? prodid, int
cdday, int cdmonth, int cdyear)
{
    DateTime cd = DateTime(cdday, cdmonth, cdyear);

```

```

        var prod = bpcontext.BakeryProducts.Include(bp =>
bp.BakeryProductId).Include(bp => bp.ProductAvailabilities).Include(bp =>
bp.BakeryGroup).Include(bp => bp.Title).Include(bp =>
bp.Description).Include(bp => bp.Photo).FirstOrDefault(bp =>
bp.BakeryProductId == prodid);
        var avail = prod.ProductAvailabilities.Include(pa =>
pa.Quantity).Include(pa => pa.AvailabilityDate).Include(pa =>
pa.FactQuantity).Include(pa => pa.ReservedQuantity) Include(pa =>
pa.LeftQuantity).FirstOrDefaultAsync(bp => bp.AvailabilityDate == cd);
        if (avail.FactQuantity == 0)
        {
            if (avail.Quantity > avail.ReservedQuality)
            {
                int q = avail.Quantity - avail.ReservedQuality;
                return View(q);
            }
            else
            {
                return View(-1);
            }
        }
        else
        {
            if (avail.LeftQuantity > 0)
            {
                return View(avail.LeftQuantity);
            }
            else
            {
                return View(-1);
            }
        }
    }

[Authorize]
public ActionResult AddAvailability(int bprod, int q, int
depid, int cdday, int cdmonth, int cdyear)
{
    DateTime cd = DateTime(cdday, cdmonth, cdyear);
    ProductAvailability pa = new ProductAvailability();
    pa.DepartmentId = depid;
    pa.BakeryProductId = bprod;
    pa.Quantity = q;

```

```

        pa.AvailabilityDate = cd;
        pa.FactQuantity = 0;
        pa.ReservedQuantity = 0;
        pa.LeftQuantity = 0;
        bpcontext.Add(pa);
        bpcontext.SaveChanges();
        return RedirectToAction("Index");
    }

    [Authorize]
    public ActionResult ChangeAvailability(int bprod, int q,
int fq, int depid, int cdday, int cdmonth, int cdyear)
    {
        DateTime cd = DateTime(cdday, cdmonth, cdyear);
        var avail = bpcontext.ProductAvailabilities.Include(pa =>
pa.Quantity).Include(pa => pa.AvailabilityDate).Include(pa =>
pa.FactQuantity).Include(pa => pa.ReservedQuantity).Include(pa =>
pa.LeftQuantity).FirstOrDefaultAsync(bp => bp.AvailabilityDate == cd &&
bp.DepartmentId == depid && bp.BakeryProductId == bprod);
        avail.Quantity = q;
        avail.FactQuantity = fq;
        bpcontext.Entry(avail).State = EntityState.Modified;
        bpcontext.SaveChanges();
        return RedirectToAction("Index");
    }

    public ActionResult GroupBakeryProducts(int? grid)
    {
        var products = bpcontext.BakeryProducts.Include(bp =>
bp.BakeryProductId).Include(bp => bp.BakeryGroup).Include(bp =>
bp.Title).Include(bp => bp.Description).Include(bp => bp.Photo).Where(bp
=> bp.BakeryGroup.GroupId == grid);
        foreach (var prod in products)
        {
            var prices = bpcontext.ProductPrices.Include(pp =>
pp.BakeryProductId).Include(pp => Price).Include(pp =>
pp.StartDate).Include(pp => pp.EndDate).Where(pp => pp.StartDate <=
nowdate && pp.EndDate >= nowdate && pp.BakeryProductId ==
prod.BakeryProductId);
            var price = prices.FirstOrDefault();
            prod.Price = price;
        }
        return View(products);
    }

```

```

    }

    public ActionResult AllProducts()
    {
        var products = bpcontext.BakeryProducts.Include(bp =>
bp.BakeryProductId).Include(bp => bp.BakeryGroup).Include(bp =>
bp.Title).Include(bp => bp.Description).Include(bp => bp.Photo);
        foreach (var prod in products)
        {
            var prices = bpcontext.ProductPrices.Include(pp =>
pp.BakeryProductId).Include(pp => Price).Include(pp =>
pp.StartDate).Include(pp => pp.EndDate).Where(pp => pp.StartDate <=
nowdate && pp.EndDate >= nowdate && pp.BakeryProductId ==
prod.BakeryProductId);
            var price = prices.FirstOrDefault();
            prod.Price = price;
        }
        return View(products);
    }
    public ActionResult BakeryProductOffers(int? prodid)
    {
        DateTime nowdate = DateTime.Now;
        var offers = bpcontext.Offers.Include(bp =>
bp.StartDate).Include(bp => bp.EndDate).Include(bp =>
bp.StartDateStr).Include(bp => bp.EndDateStr).Include(bp =>
bp.Price).Include(bp => bp.Quantity).Where(bp => bp.Quantity > 0 &&
bp.StartDate <= nowdate && bp.EndDate >= nowdate);
        foreach (var off in offers)
        {
            bool bpo = false;
            foreach (var of in off.BakeryProducOffers)
            {
                if (of.BakeryProductId == prodid)
                {
                    bpo = true;
                }
            }
            if (!bpo)
            {
                offers.Remove(off);
            }
        }
        return View(offers);
    }

```

```

    }

    [Authorize]
    public ActionResult Create()
    {
        ViewData["Groups"] = new
SelectList(bpcontext.BakeryGroups, "BakeryGroupId", "Title");
        return View();
    }

    [HttpPost]
    [ValidateAntiForgeryToken]
    public ActionResult Create([Bind("BakeryGroupId, Title,
Description, Photo")] BakeryProduct bprod)
    {
        if (ModelState.IsValid)
        {
            bpcontext.Add(bprod);
            bpcontext.SaveChanges();
            return RedirectToAction("Index");
        }
        return View(bprod);
    }

    [Authorize]
    public ActionResult Edit(int? bprod)
    {
        BakeryProduct bp =
bpcontext.BakeryProducts.Find(bprod);
        if (bp == null)
        {
            return NotFound();
        }
        ViewData["Groups"] = new
SelectList(bpcontext.BakeryProducts, "BakeryProductId", "Title");
        return View(bp);
    }

    [HttpPost]
    [ValidateAntiForgeryToken]
    public ActionResult Edit(int bprod, [Bind("BakeryProductId,
BakeryGroupId, Title, Description, Photo")] BakeryProduct bprod)
    {

```

```

        if (ModelState.IsValid)
        {
            bpcontext.Entry(bprod).State =
EntityState.Modified;
            bpcontext.SaveChanges();
        }
        ViewData["Groups"] = new
SelectList(bpcontext.BakeryProducts, "BakeryProductId", "Title");
        return View(bprod);
    }

    [Authorize]
    public ActionResult AddPrice(int bprod, int price, int
sdday, int sdmonth, int sdyear, int edday, int edmonth, int edyear)
    {
        bool noprice = false;
        DateTime sd = DateTime(sdday, sdmonth, sdyear);
        DateTime ed = DateTime(edday, edmonth, edyear);
        var pp = bpcontext.ProductPrices.Include(pa =>
pa.BakeryProductId).Include(pa => pa.Price).Include(pa =>
pa.StartDate).Include(pa => pa.EndDate).FirstOrDefaultAsync(pa =>
pa.BakeryProductId == bprod && pa.StartDate >= sd && pa.EndDate <= ed );
        if (pp != null)
        {
            noprice = true;
        }
        pp = bpcontext.ProductPrice.Include(pa =>
pa.BakeryProductId).Include(pa => pa.Price).Include(pa =>
pa.StartDate).Include(pa => pa.EndDate).FirstOrDefaultAsync(pa =>
pa.BakeryProductId == bprod && pa.StartDate <= sd && pa.EndDate >= ed );
        if (pp != null)
        {
            noprice = true;
        }
        pp = bpcontext.ProductPrice.Include(pa =>
pa.BakeryProductId).Include(pa => pa.Price).Include(pa =>
pa.StartDate).Include(pa => pa.EndDate).FirstOrDefaultAsync(pa =>
pa.BakeryProductId == bprod && pa.StartDate <= sd && pa.EndDate >= sd &&
pa.EndDate <= ed);
        if (pp != null)
        {
            noprice = true;
        }
    }

```

```

        var pp = bpcontext.ProductPrice.Include(pa =>
pa.BakeryProductId).Include(pa => pa.Price).Include(pa =>
pa.StartDate).Include(pa => pa.EndDate).FirstOrDefaultAsync(pa =>
pa.BakeryProductId == bprod && pa.StartDate >= sd && pa.StartDate <= sd
&& pa.EndDate >= ed);
        if (pp != null)
        {
            noprice = true;
        }
        if (noprice)
        {
            return NotFound();
        }
        ProductPrice pa = new ProductPrice();
        pa.BakeryProductId = bprod;
        pa.Price = price;
        pa.StartDate = sd;
        pa.EndDate = ed;
        bpcontext.Add(pa);
        bpcontext.SaveChanges();
        return RedirectToAction("Index");
    }

```

```

[Authorize]
public ActionResult ChangePrice(int bprod, int price, int
sdday, int sdmonth, int sdyear, int edday, int edmonth, int edyear)
{
    DateTime sd = DateTime(sdday, sdmonth, sdyear);
    DateTime ed = DateTime(edday, edmonth, edyear);
    var pp = bpcontext.ProductPrices.Include(pa =>
pa.BakeryProductId).Include(pa => pa.Price).Include(pa =>
pa.StartDate).Include(pa => pa.EndDate).FirstOrDefaultAsync(pa =>
pa.BakeryProductId == bprod && pa.StartDate >= sd && pa.EndDate <= ed );
    if (pp == null)
    {
        pp = bpcontext.ProductPrice.Include(pa =>
pa.BakeryProductId).Include(pa => pa.Price).Include(pa =>
pa.StartDate).Include(pa => pa.EndDate).FirstOrDefaultAsync(pa =>
pa.BakeryProductId == bprod && pa.StartDate <= sd && pa.EndDate >= ed );
        if (pp == null)
        {
            pp = bpcontext.ProductPrice.Include(pa =>
pa.BakeryProductId).Include(pa => pa.Price).Include(pa =>

```

```

pa.StartDate).Include(pa => pa.EndDate).FirstOrDefaultAsync(pa =>
pa.BakeryProductId == bprod && pa.StartDate <= sd && pa.EndDate >= sd &&
pa.EndDate <= ed);

        if (pp == null)
        {
            pp = bpcontext.ProductPrice.Include(pa =>
pa.BakeryProductId).Include(pa => pa.Price).Include(pa =>
pa.StartDate).Include(pa => pa.EndDate).FirstOrDefaultAsync(pa =>
pa.BakeryProductId == bprod && pa.StartDate >= sd && pa.StartDate <= sd
&& pa.EndDate >= ed);
        }
    }
    if (pp == null)
    {
        return NotFound();
    }
    pp.Price = price;
    pa.StartDate = sd;
    pa.EndDate = ed;
    bpcontext.Entry(pa).State = EntityState.Modified;
    bpcontext.SaveChanges();
    return RedirectToAction("Index");
}
}
}

```

**ДОДАТОК В**  
**Слайди презентації**

Національний університет «Запорізька політехніка»  
Кафедра програмних засобів

**Дослідження та програмна реалізація методів  
прогнозування часу доставки виробів пекарні**

**Research and Software Implementation of Methods for  
Predicting the Bakery Delivery Time**

|                             |               |
|-----------------------------|---------------|
| Виконав<br>ст. гр. КНТ-220м | Федорчук Д.О. |
| Керівник<br>к.т.н., доцент  | Льовкін В.М.  |

2021

Рисунок В.1 – Слайд 1

Об'єкт дослідження – процес замовлення доставки виробів пекарні.

Предмет дослідження – методи прогнозування часу доставки виробів пекарні.

Мета роботи – розроблення програмного забезпечення замовлення доставки виробів пекарні для підвищення зручності роботи на основі прогнозування часу доставки.

---

▶ 2

Рисунок В.2 – Слайд 2

## Завдання дипломної кваліфікаційної роботи

- дослідження методів прогнозування часу доставки виробів пекарні;
- розробка технічного завдання і вибір засобів розробки;
- проєктування програмної системи;
- розробка програмної системи замовлення доставки виробів пекарні.

► 3

Рисунок В.3 – Слайд 3

## Порівняння алгоритмів навчання дерев рішень

| Критерій                              | ID3 | C4.5 | CART | XGBoost |
|---------------------------------------|-----|------|------|---------|
| Задача класифікації                   | +   | +    | +    | +       |
| Задача прогнозування                  | –   | –    | +    | +       |
| Підтримка бустингу                    | –   | –    | +    | +       |
| Швидкість роботи                      | –   | –+   | +    | ++      |
| Підтримка неповної навчальної вибірки | –   | +    | +    | +       |

► 4

Рисунок В.4 – Слайд 4

## Метод прогнозування часу доставки виробів пекарні

Набір даних сформувати на основі використання наступних ознак:

- ▶ час початку доставки;
- ▶ тип транспортного засобу;
- ▶ кількість одиниць виробів у замовленні;
- ▶ район, в якому розташовано початковий пункт доставки;
- ▶ найкоротша відстань від точки відправлення до точки доставки;
- ▶ напрямок руху від початкової точки до кінцевої;
- ▶ температура повітря;
- ▶ вологість повітря;
- ▶ напрямок вітру;
- ▶ швидкість вітру;
- ▶ видимість;
- ▶ наявність снігу;
- ▶ наявність дощу;
- ▶ час доставки.

Виконати передбороблення даних.

Виконати навчання моделі прогнозування на основі дерев рішень та алгоритму навчання XGBoost за допомогою сформованого набору даних.

▶ 5

Рисунок В.5 – Слайд 5

## Вибір вебфреймворка

| Критерій                                    | Django | ASP.NET Core |
|---------------------------------------------|--------|--------------|
| Мова програмування                          | Python | C#           |
| Наявність об'єктно-реляційного відображення | +      | +            |
| Використання шаблону проєктування           | +      | +            |
| Загальна надійність                         | +/-    | +            |
| Швидкісне виконання                         | -/+    | +            |
| Кросплатформність                           | +      | +            |

▶ 6

Рисунок В.6 – Слайд 6

## Діаграми прецедентів клієнта, касира, кур'єра

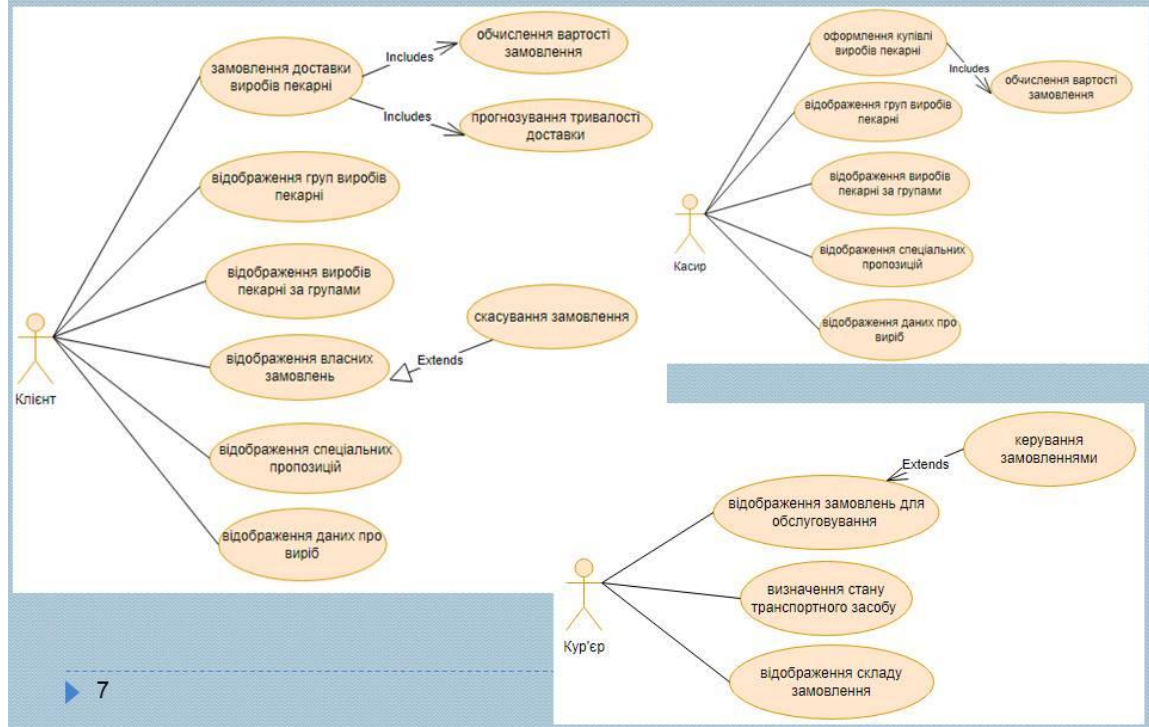


Рисунок В.7 – Слайд 7

## Діаграми прецедентів менеджера та адміністратора

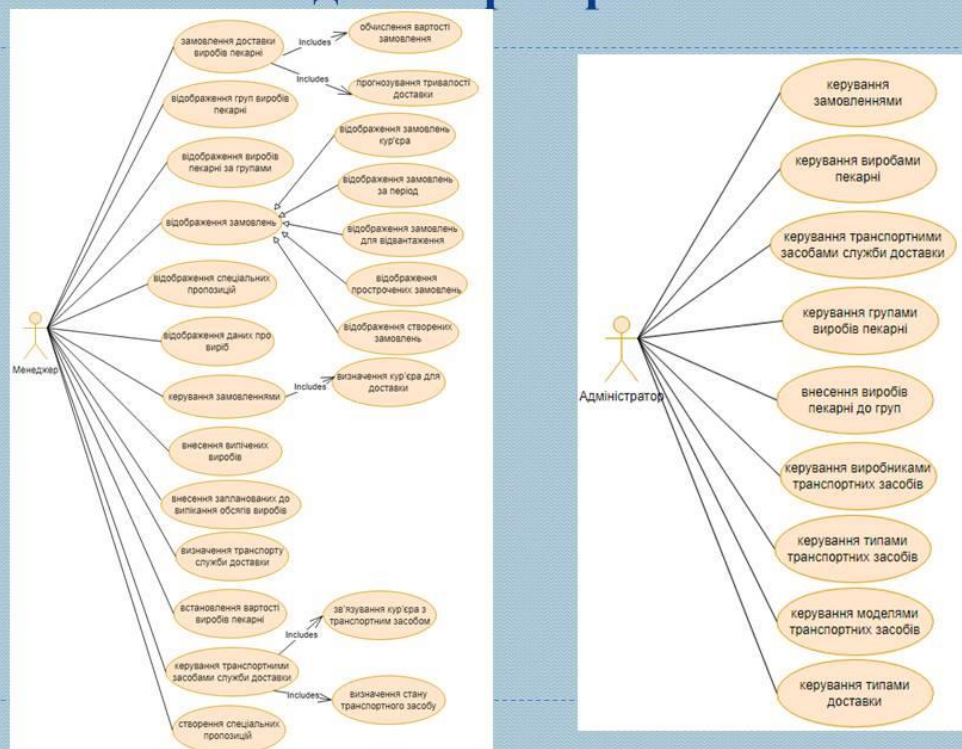
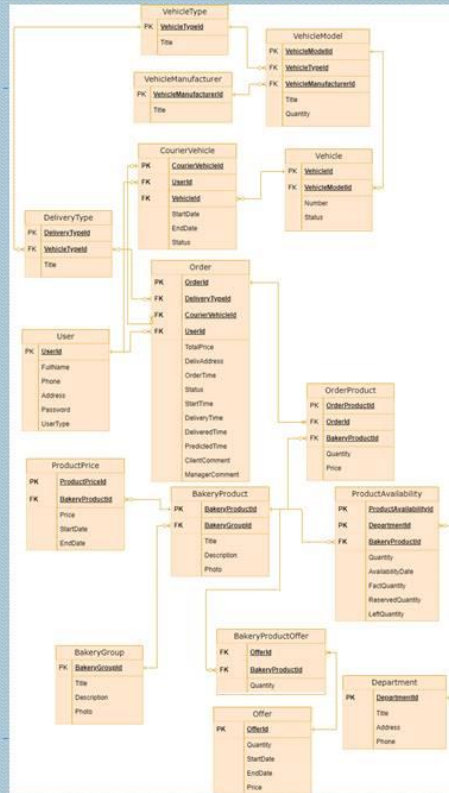


Рисунок В.8 – Слайд 8

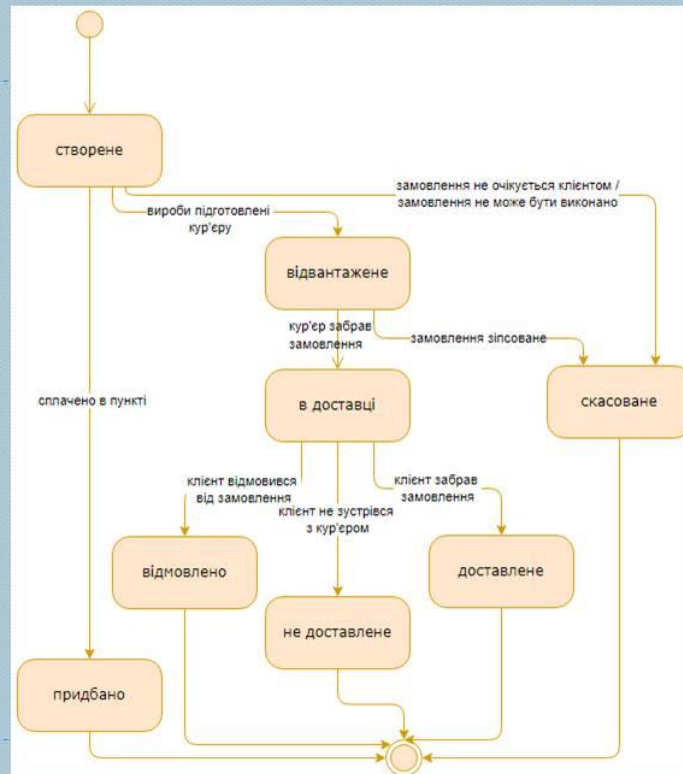
## Схема бази даних



► 9

Рисунок В.9 – Слайд 9

## Діаграма станів замовлення на доставку



► 10

Рисунок В.10 – Слайд 10



Рисунок В.11 – Слайд 11

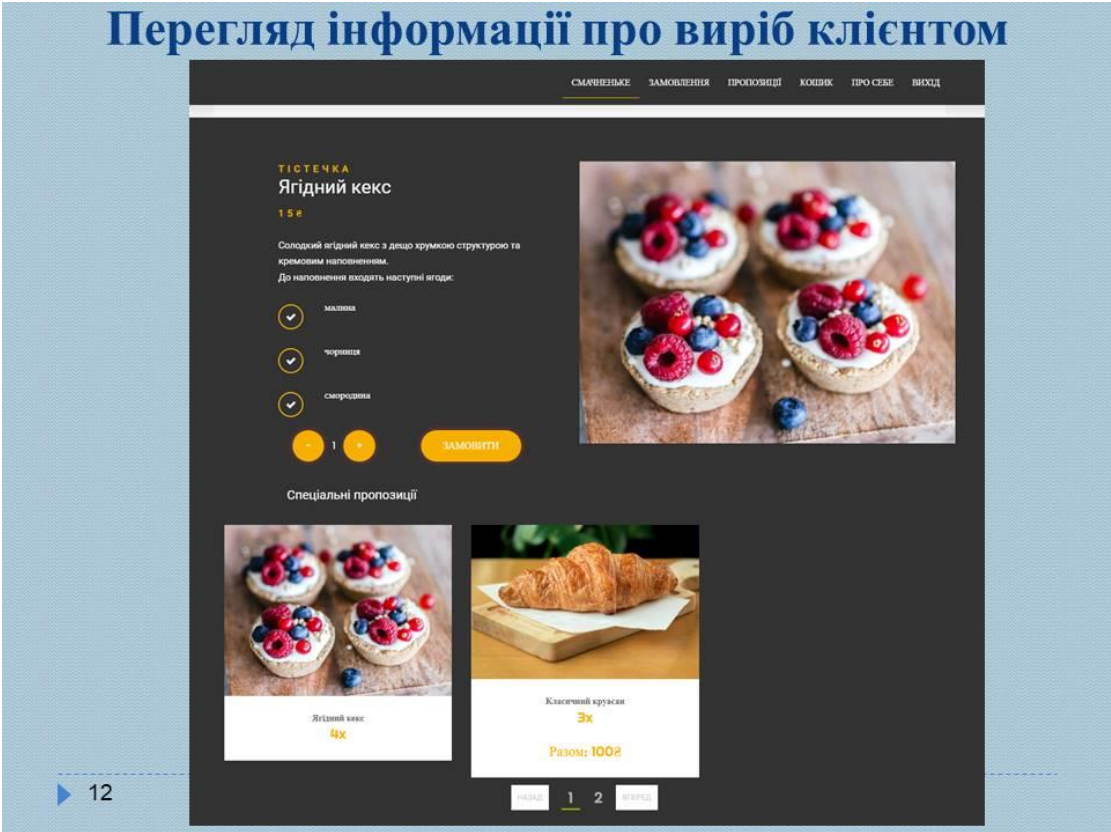


Рисунок В.12 – Слайд 12

## Перегляд замовлень клієнтів менеджером

ГРУПИ
ВАРТІСТЬ
ПЛАН
**ЗАМОВЛЕННЯ**
ТРАНСПОРТ
ПРОПОВІДІ
КОШТІВ
ПРО СЕБЕ
ВИХІД

СТВОРИТИ НОВЕ ЗАМОВЛЕННЯ →

**ОФОРМЛЕНІ ЗАМОВЛЕННЯ**  
 Номер  
 Пошук за номером замовлення  
 \_\_\_\_\_  
 Параметр пошуку  
 Дати  
 Початкова дата  
 03.12.2021  
 Кінцева дата  
 04.12.2021  
 ШУКАТИ

| Номер | Вартість | Стан         | Створення            | Прогноз              |
|-------|----------|--------------|----------------------|----------------------|
| 31    | 3008     | Створене     | 04 грудня 2021 11:15 | 04 грудня 2021 11:43 |
| 30    | 2158     | Створене     | 04 грудня 2021 11:13 | 04 грудня 2021 11:57 |
| 29    | 358      | Створене     | 04 грудня 2021 11:12 | 04 грудня 2021 12:03 |
| 28    | 210.28   | Відвантажено | 04 грудня 2021 11:06 | 04 грудня 2021 11:53 |
| 27    | 778      | Відвантажено | 04 грудня 2021 11:03 | 04 грудня 2021 11:51 |
| 26    | 143.58   | В доставці   | 04 грудня 2021 10:58 | 04 грудня 2021 11:39 |
| 25    | 898      | В доставці   | 04 грудня 2021 10:56 | 04 грудня 2021 11:42 |
| 24    | 1248     | В доставці   | 04 грудня 2021 10:53 | 04 грудня 2021 11:32 |

13

Рисунок В.13 – Слайд 13

## Порівняння точності прогнозування за результатами експериментів

| Модель                                               | Точність прогнозування |
|------------------------------------------------------|------------------------|
| Регресія за методом опорних векторів                 | 0,372                  |
| Логістична регресія                                  | 0,317                  |
| Дерева рішень XGBoost (представлений у роботі метод) | 0,198                  |

14

Рисунок В.14 – Слайд 14

## Висновки

За результатами огляду програмних засобів підтримки замовлень у пекарні було виділено основні функціональні можливості даних програмних засобів і визначено функціональні вимоги до програмної розробки. Особливістю програми є надання можливості прогнозування часу доставки виробів пекарні.

Виконано порівняння алгоритмів навчання дерев рішень. Представлено власний метод прогнозування часу доставки виробів пекарні.

Обрано фреймворк веброзробки ASP.NET Core та мову програмування C# для реалізації програмного забезпечення. Створено структурну схему БД.

Виконано програмну реалізацію вебзастосунку замовлення доставки виробів пекарні.

Проведено експериментальне дослідження прогнозування часу доставки. Представлений у роботі методі підтвердив свою здатність до вирішення проблеми.

Наукова новизна роботи полягає в запропонованому методі прогнозування часу доставки виробів пекарні, який використовує ансамбль дерев рішень з навчанням на основі градієнтного бустинга, що дозволяє визначати час доставки замовлень виробів пекарні за набором їх характеристик.

► 15 Завдання роботи виконані повністю.

Рисунок В.15 – Слайд 15