Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)
Кафедра Інформатики
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

РОЗРОБКА ЗАСТОСУНКУ ДЛЯ ОБЛІКУ ТА СПИСАННЯ ПРОДУКТІВ У СИСТЕМІ РЕСТОРАНУ
(тема)

Виконав:
студент 4 курсу, групи ІТІНФ-19-2
Артьомов Д.О.
(прізвище, ініціали)

Спеціальності 122 Комп'ютерні науки
(код і повна назва спеціальності)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

Керівник доц. Руденко Д.О.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ Кобилін О.А.
(підпис) (прізвище, ініціали)

2023 р.

Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)

Кафедра Інформатики
(повна назва)

Рівень вищої освіти перший (бакалаврський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 2023 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Артьомову Данилу Олексійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка застосунку для обліку та списання продуктів у системі ресторану

затверджена наказом університету від 15 травня 2023 року № 474 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 22 травня 2023 р.

3. Вихідні дані до роботи науково-методична та науково-технічна література, матеріали конференцій, дані інтернет-мережі, бібліотека для розробки графічного інтерфейсу користувача з відкритим кодом JavaFX, бібліотека Apache POI для роботи із файлами в форматах Microsoft Office, бібліотека для накладення анімації на елементи робочого вікна AnimateFX, бібліотека Apache Commons Codec для отримання геш-значення змінних.

4. Перелік питань, що потрібно опрацювати в роботі
1. Реалізація двофакторної автентифікації засобами інтернет-сервісів.
2. Розробка стабільної структури бази даних.
3. Розробка «дружнього» інтерфейсу користувача.
4. Гешування паролів з метою безпеки.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) Розробка концептуальної моделі, розробка повної структури БД, програмна реалізація, тестування застосунку, тестові зображення.

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата
Консультант з дотримання діючих стандартів та норм	Доцент Творошенко І.С.		

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	10.04.2023	
2	Аналіз завдання, підбір літератури	11.04.23-13.04.23	
3	Аналіз літератури з досліджуваної проблеми	14.04.23-16.04.23	
4	Аналіз технічних засобів	17.04.23-18.04.23	
5	Розробка структури бази даних	19.04.23-01.05.23	
6	Програмна реалізація	02.05.23-18.05.23	
7	Оформлення пояснювальної записки	18.05.23-23.05.23	
8	Перевірка на плагіат	24.05.23-26.05.23	
9	Рецензування	27.05.23-28.05.23	
10	Підготовка презентації та доповіді	29.05.23-30.05.23	
11	Занесення роботи в електронний архів	31.05.23	
12	Попередній захист кваліфікаційної роботи	31.05.23	

Дата видачі завдання 10 квітня 2023 р.

Студент _____
(підпис)

Керівник роботи _____ доц. Руденко Д.О.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ/ABSTRACT

Пояснювальна записка до кваліфікаційної роботи: 75 с., 60 рис., 1 дод., 31 джерело.

БАЗА ДАНИХ, ТОВАРИ, ЗАМОВЛЕННЯ, СПИСАННЯ, ЗАСТОСУНОК, РЕСТОРАН, ВІДНОШЕННЯ, МОДЕЛЬ, КОРИСТУВАЧ, ПОСТАЧАЛЬНИК, ОБЛІК.

Об’єктом роботи є низка товарів, які використовуються у ресторані для приготування страв та реалізації.

Метою цієї роботи є створення програмного забезпечення, яке дозволить ресторану ефективно вести облік та списання продуктів.

У рамках роботи проведено дослідження різних методів пошуку та списання продуктів у системі ресторану, а також аналіз методів управління базою даних. Використовуючи методи числового моделювання та аналітичного обґрунтування, було розроблено ефективний алгоритм для деталізованого обліку продуктів у системі ресторану.

Основними компонентами розробленого застосунку є база даних Oracle, яка використовується для зберігання інформації про продукти та графічний інтерфейс користувача, реалізований з використанням JavaFX, який надає зручний інтерфейс для взаємодії користувача з системою.

DATABASE, PRODUCTS, ORDER, WRITE-OFF, APP, RESTAURANT, RELATION, MODEL, USER, SUPPLIER, ACCOUNTING.

The purpose of this work is to create software that will allow the restaurant to efficiently keep records and write off products.

The object of the work is a number of products that are used in the restaurant for cooking and sales.

As part of the work, a study of various methods of searching and writing off products in the restaurant system was carried out, as well as an analysis of database management methods. Using the methods of numerical modeling and analytical reasoning, an effective algorithm was developed for detailed accounting of products in the restaurant system.

The main components of the developed application are an Oracle database, which is used to store information about products and a graphical user interface, implemented using JavaFX, which provides a convenient interface for user interaction with the system.

ЗМІСТ

Перелік умовних позначень, символів, одиниць, скорочень і термінів 6
Вступ 7
1 Аналіз поточного становища завдань обліку та списання товарів у системах ресторанів 9
1.1 Стан проблеми обліку та списання товарів у системі ресторану 9
1.2 Загальна характеристика об’єкта дослідження 10
1.3 Характеристика існуючих систем обліку 12
1.3.1 Система ULTRA 13
1.3.2 Система IIKO 15
1.3.3 Система Poster 17
1.4 Постановка задачі 19
2 Алгоритми та засоби реалізації задачі обліку та списання товарів у системі ресторану 20
2.1 Описання засобів підтримки бази даних, що використовуються .. 20
2.1.1 Модель даних 20
2.1.2 Концептуальна модель 26
2.2 Розробка концептуальної моделі 28
2.3 Розробка повної структури БД 29
3 Створення програмного продукту 36
3.1 Розробка структури БД 36
3.2 Програмна реалізація 39
3.3 Тестування застосунку 45
3.4 Виявлені недоліки 60
Висновки 61
Перелік джерел посилання 63
Додаток А Тестові зображення 66

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

БД – база даних

SQL – Structured Query Language

GUI – Graphical User Interface

PK(ПК) – Primary Key (первинний ключ)

FK(ЗК) – Foreign Key (зовнішній ключ)

РБД – реляційна база даних

ІМ – ієрархічна модель

ТЗ – технічне завдання

ВСТУП

У сучасному глобалізованому світі ресторанна галузь постійно зростає та розвивається. Однак, в умовах жорсткої конкуренції та швидкого темпу життя, ефективне управління ресторанним бізнесом стає складним завданням. Одним із важливих аспектів ефективного управління рестораном є точний та оперативний облік руху товарів, таких як продукти харчування, напої та інші матеріали, які використовуються в ресторанному бізнесі. Відсутність або недостатня ефективність такого обліку може призвести до фінансових втрат, збитків та незадоволення з боку клієнтів. Для вирішення цієї проблеми в кваліфікаційній роботі розглядається розробка застосунку для обліку та списання товарів в системі ресторану, з використанням мови програмування Java, технології JavaFX для розробки графічного інтерфейсу користувача, та OracleDB для зберігання даних.

Актуальність даної кваліфікаційної роботи обумовлена високою конкуренцією в ресторанній галузі, де ефективне управління запасами та облік руху товарів має велике значення для забезпечення стабільності бізнесу. Ручний облік товарів може бути часо-, праце- та ресурсномістким, а також викликати помилки, що може впливати на точність та швидкість обліку.

Використання мови програмування Java та технології JavaFX дозволить розробити зручний та інтуїтивно зрозумілий графічний інтерфейс користувача, що сприятиме зручній взаємодії операторів з застосунком. Використання OracleDB для зберігання даних дозволить забезпечити безпеку, надійність та швидкий доступ до даних.

Дана кваліфікаційна робота має практичне значення, оскільки розроблений застосунок може бути використаний в ресторанному бізнесі для ефективного управління товарами, забезпечення точного обліку та списання товарів, а також покращення ефективності та прибутковості ресторанного бізнесу. Застосунок може бути впроваджений в різних типах ресторанів, від

малих закладів до великих мереж ресторанів, що дозволяє розширити його застосування в різних галузях ресторанної індустрії.

У ході розробки кваліфікаційної роботи будуть використані сучасні методи та технології розробки програмного забезпечення, такі як архітектура клієнт-сервер, обробка подій, робота з базами даних та вебсервісами, що дозволить отримати практичні навички розробки програмного забезпечення на об’єктно-орієнтованій мові Java .

Java стала широко розповсюджуватися ще з 1995 р. Вона орієнтована на мережі комп’ютерів і, перш за все, серверні застосунку, які тісно працюють із базами даних [1].

Основні наукові цілі даної роботи включають вивчення та аналіз сучасних методів та технологій розробки застосунків на мові Java, проєктування та реалізація функціоналу застосунку для управління товарами в ресторанному бізнесі, проведення тестування та валідації розробленого застосунку, а також формування висновків щодо його ефективності та можливостей подальшого вдосконалення.

Очікується, що результати цієї роботи допоможуть в розробці функціонального застосунку для управління товарами в ресторанному бізнесі, що має потенціал бути використаним в реальних умовах роботи ресторанів.

1 АНАЛІЗ ПОТОЧНОГО СТАНОВИЩА ЗАВДАНЬ ОБЛІКУ ТА СПИСАННЯ ТОВАРІВ У СИСТЕМАХ РЕСТОРАНІВ

1.1 Стан проблеми обліку та списання товарів у системі ресторану

Управління запасами є однією з ключових функцій ефективного ведення ресторанного бізнесу. Один з важливих аспектів управління запасами – це облік та списання товарів. Цей процес має вирішальне значення для забезпечення ефективного функціонування ресторану, оптимізації витрат і підтримки високої якості продукції.

Проте, стан проблеми обліку та списання товарів у системі ресторану може бути складним через кілька факторів. Ось кілька основних проблем, які можуть виникнути в цьому процесі:

– недостатній облік та відсутність системи контролю. Однією з найпоширеніших проблем є відсутність точної системи обліку запасів та механізмів контролю за ними. Це може призводити до неправильного обліку кількості товарів, втрати контролю над термінами придатності та непередбачуваних списань товарів, що може вплинути на прибутковість ресторану;

– не співпадіння фактичного та облікового обліку. Інша проблема, яка може виникнути, – це відсутність відповідності між фактичним обсягом товарів на складі та обліковими записами. Це може бути викликано різними причинами, такими як крадіжки, помилки в обліку або недостатня точність вимірювання. Це може призвести до недостачі або переобліку товарів, що може вплинути ефективність управління запасами;

– несистемний процес списання товарів. Ще одна проблема, яка може виникнути, – це відсутність системного підходу до процесу списання товарів. Відсутність чітко визначених правил та процедур може призвести до неправильного списання товарів або навіть до можливості зловживань.

Наприклад, списання товарів може бути здійснюватися без належного документування, без відповідної авторизації або без перевірки фактичного стану товарів на складі. Це може призвести до втрати товарів, втрати контролю над запасами та можливих фінансових втрат;

– неправильне врахування різних факторів. При обліку та списанні товарів в ресторані необхідно враховувати різні фактори, такі як витрати на доставку, знижки, бонуси, повернення товарів, а також врахування можливих різниць в якості товарів. Неправильне врахування цих факторів може призвести до неточностей у фінансовому обліку та вплинути на прибутковість ресторану;

– відсутність автоматизованої системи обліку та списання товарів. В деяких ресторанах використовуються ручні методи обліку та списання товарів, що може бути неефективним та призводити до помилок. Відсутність автоматизованої системи обліку та списання товарів може призвести до збільшення ручної праці, помилок в обліку та втрати часу.

Резюмуючи, стан проблеми обліку та списання товарів у системі ресторану може бути викликом для ефективного управління запасами. Недостатній облік, відсутність системного підходу, неправильне врахування різних факторів та відсутність автоматизованої системи можуть призводити до помилок, втрати контролю над запасами та впливати на фінансові показники ресторану.

1.2 Загальна характеристика об’єкта дослідження

Застосунок, що знаходиться у розробці, призначений для автоматизації процесу обліку та списання товарів в ресторанному бізнесі з метою підвищення ефективності управління запасами та контролю над рухом товарів.

Java – мова програмування високого рівня, що надає розробнику набір засобів для створення застосунків різного напряму. Однією із важливих

функцій Java є модульність. Модульне програмування є розвитком і вдосконалюванням процедурного програмування й бібліотек спеціальних програм. Основна риса модульного програмування – стандартизація інтерфейсу між окремими програмними одиницями [2].

Центральною компонентою більшості інформаційних систем є централізовані та розподілені бази даних, які розробляються на сучасних СУБД, таких як ORACLE та інших, з використанням технології клієнт-сервер [3].

Окрім бази даних застосунок використовує JavaFX – графічний фреймворк для розробки «десктопних» застосунків, що надає можливості створення інтуїтивно зрозумілого та зручного GUI. JavaFX дозволяє розробникам створювати візуально привабливі та функціональні застосунки, які можуть взаємодіяти з користувачем за допомогою різних компонентів, таких як кнопки, тексти, таблиці, форми та інші.

Одним з ключових елементів застосунку є використання OracleDB в якості бази даних для зберігання і обробки даних про товари, їх кількість, вартість, рух та інші аспекти обліку. OracleDB є однією з найпоширеніших реляційних баз даних, відомою своєю надійністю, масштабованістю та можливостями розширення. Використання бази даних дозволяє забезпечити цілісність, консистентність та безпеку даних, а також забезпечити ефективну роботу зі списками товарів, їх рухом та аналітикою [4].

СУБД Oracle було обрано, оскільки вона дає можливість скористатися якісним ПО для розробки системи [5].

Проте, незважаючи на усі переваги, якими б не були засоби розробки і підхід до створення програмного забезпечення, у більшості випадків кінцева програма має бути ефективною, продуктивною та стійкою до помилок у разі відхилення від «ідеальних дій користувача», і залежить це саме від розробника [6].

Застосунок для обліку та списання товарів у системі ресторану має наступні основні функції:

– додавання та відображення товарів: застосунок дозволяє додавати та відображати товари, включаючи їх назву, категорію, кількість, ціну та інші важливі атрибути;

– облік руху товарів: застосунок дозволяє реєструвати рух товарів, такий як закупівля, продаж, списання та інші операції. Рух товарів реєструється з використанням дати, кількості, ціни та інших деталей, що дозволяє відслідковувати деталі переміщення товарів у системі ресторану;

– списання товарів: застосунок автоматично відслідковує кількість товарів та їх рух у системі, що дозволяє визначати необхідність списання товарів. Списання може бути автоматичним або виконуватися вручну згідно з встановленими правилами та процедурами;

– аналітика та звітність: застосунок надає можливості для аналізу даних про товари, їх рух, вартість та інші аспекти обліку. Завдяки цьому, керівництво ресторану може отримувати звіти та аналітичні дані, необхідні для прийняття рішень щодо управління запасами, виявлення ризиків та оптимізації процесів;

– безпека та авторизація: застосунок забезпечує захист даних, використовуючи механізми авторизації та аутентифікації користувачів. Це дозволяє забезпечити обмежений доступ до даних, забезпечити цілісність та конфіденційність даних.

1.3 Характеристика існуючих систем обліку

Основні функції застосунку для обліку та списання товарів у системі ресторану мають включати:

– додавання, оновлення та видалення записів про товари;

– відстеження руху товарів, таких як закупівля, продаж, тощо;

– автоматичне списання товарів на основі заданих правил;

– контроль рівня запасів та відслідковування мінімального рівня запасів;

– генерація звітів про залишки товарів, рух товарів, витрати, прибуток тощо;

– забезпечення безпеки даних, включаючи авторизацію, аутентифікацію та контроль доступу до даних;

– можливість імпорту/експорту даних для обміну з іншими системами;

– моніторинг вартості товарів та витрат на їх закупівлю та реалізацію;

– підтримка різних одиниць виміру товарів та конвертація між ними;

– інтеграція з іншими системами, такими як бухгалтерський облік, касовий апарат тощо.

Основні та найпопулярніші аналоги, що вже існують на ринку перелічені у підпунктах 1.3.1, 1.3.2 та 1.3.3.

1.3.1 Система ULTRA

Ultra – це програма для автоматизації обліку, розрахунків і управління бізнесом в галузях ресторанного бізнесу, торгівлі, готельного господарства і фітнес-центрів. Вона дозволяє ефективно вести бізнес-процеси, оптимізувати витрати, підвищувати ефективність роботи та забезпечувати кращий рівень обслуговування клієнтів [7].

Основні функції Ultra включають:

– облік продажів і розрахунків: Ultra дозволяє вести детальний облік продажів, розрахунків та виручки. Можна налаштувати різні види оплати, включаючи готівку, кредитні карти, купони, сертифікати та інші способи оплати;

– управління складським обліком: Програма дозволяє вести облік товарів на складі, керувати постачанням товарів, контролювати рівень запасів та оптимізувати процеси замовлення;

– управління персоналом: Ultra допомагає контролювати робочий час співробітників, розраховувати заробітну плату, налаштовувати рівні доступу для різних категорій персоналу та відстежувати виконання робочих завдань;

– аналітика та звітність: Програма надає можливість генерувати різноманітні звіти та аналітичні дані про продажі, рентабельність, оборотність товарів, а також роботу персоналу. Це дозволяє керівникам приймати обґрунтовані рішення для розвитку бізнесу;

– інтеграція з іншими системами: Ultra може бути інтегрована з різними системами, такими як системи електронного оформлення замовлень, програми лояльності клієнтів, системи обліку фінансів, системи керування готельними номерами та інші системи, що використовуються в ресторанному бізнесі, торгівлі, готельному господарстві та фітнес-центрах;

– управління замовленнями: Ultra дозволяє зручно приймати та обробляти замовлення в режимі реального часу, включаючи можливість додавання знижок, опцій та додаткових послуг. Замовлення можуть бути прийняті з різних джерел, таких як внутрішній сайт, мобільний застосунок, залізничні станції та інші зовнішні джерела;

– керування клієнтами: Ultra дозволяє зберігати інформацію про клієнтів, керувати програмами лояльності, створювати спеціальні пропозиції та знижки для постійних клієнтів, а також відстежувати історію замовлень клієнтів;

– мобільний доступ: Ultra надає можливість віддаленого доступу до системи з використанням мобільних пристроїв, що дозволяє керувати бізнесом з будь-якого місця і в будь-який час;

– підтримка різних мов та валют: Програма підтримує різні мови та валюти, що дозволяє використовувати її в різних країнах та регіонах;

– безпека та захист даних: Ultra забезпечує високий рівень захисту даних клієнтів, включаючи захист від несанкціонованого доступу та резервне копіювання даних.

За допомогою проведеного дослідження, вдалося встановити, що програма автоматизації Ultra має низку недоліків, а саме:

– відсутня класифікація причини списання товару, що унеможливлює створення окремих звітів;

– звіти зберігають не повну інформацію стосовно позицій;

– формування звітів відбувається задовго;

– процедура збереження недостатньо автоматизована (після зміни даних, що були введені у минулому – усі майбутні дані мають бути збережені в ручному режимі).

Також, шляхом дослідження, було виявлено декілька неявних переваг, а саме:

– велика різноманітність звітів;

– клієнт-орієнтована підтримка, що за умови сплаченої підписки виконує персональне налаштування, створення окремого шаблону звіту, тощо.

1.3.2 Система IIKO

IIKO – це комплексна система автоматизації управління ресторанним бізнесом, яка включає в себе різноманітні модулі та функції для оптимізації роботи в гастрономічному секторі. Основна мета IIKO – це покращення ефективності роботи ресторану, зниження витрат та підвищення рівня обслуговування клієнтів [8].

Основні функції та можливості системи IIKO:

– управління замовленнями: IIKO дозволяє приймати та обробляти замовлення в режимі реального часу, включаючи можливість додавання знижок, опцій та додаткових послуг. Замовлення можуть бути прийняті з різних джерел, таких як каса в ресторані, мобільні пристрої, онлайн-замовлення та інші зовнішні джерела;

– управління складом та обліком товарів: IIKO дозволяє вести облік товарів на складі, включаючи приймання товарів, відправлення на кухню,

контроль за залишками, автоматичне оновлення меню на основі наявності товарів та інші функції, що допомагають знизити втрати і оптимізувати процес управління товаро-запасами;

– управління персоналом: IIKO дозволяє керувати роботою персоналу, включаючи розклад роботи, контроль доступу, облік робочого часу, розрахунок заробітної плати та інші функції, що допомагають підвищити ефективність роботи колективу;

– аналітика та звітність: IIKO надає різноманітні звіти та аналітику, що дозволяє власникам ресторанів аналізувати фінансові показники, відстежувати динаміку продажів;

– інтерфейс: Інтерфейс IIKO є зручним та інтуїтивно зрозумілим, що дозволяє оптимізувати роботу в ресторані та забезпечити високий рівень обслуговування для клієнтів;

– інтеграція з різними платформами: IIKO може бути інтегрована з різними зовнішніми системами, такими як онлайн-замовлення, облікові системи, електронні меню та інші, що дозволяє автоматизувати процеси та забезпечує швидку та ефективну роботу;

– мобільний застосунок: IIKO має мобільний застосунок, який дозволяє керувати рестораном та відстежувати важливі показники діяльності в режимі реального часу, навіть коли ви не знаходитесь в ресторані;

– підтримка клієнтів: IIKO надає високий рівень технічної підтримки та навчання персоналу, включаючи «вебінари», онлайн-допомогу та консультації з фахівцями, що допомагає власникам ресторанів отримувати максимальну користь від використання системи.

IIKO використовується в ресторанному бізнесі, торговельних мережах, готелях, фітнес-центрах та інших гастрономічних закладах для автоматизації процесів управління, підвищення ефективності роботи, контролю над фінансами та забезпечення високого рівня обслуговування клієнтів.

Проведене дослідження дало розуміння справжніх переваг та недоліків, які стають помітні під час роботи із системою. Із переваг необхідно виділити:

– інтерактивні звіти, що містять посилання на вікна у програмі;

– User-Friendly інтерфейс;

– оптимізований процес списання товарів;

– можливість вести облік та нарахування заробітної платні працівників.

Основний недолік – система розроблена російською компанією «Айко». Вона приведена як приклад, та наразі заборонена до використання в Україні, через терористичні дії «російської федерації» на території нашої держави.

1.3.3 Система Poster

Українська компанія Poster є розробником та постачальником програмного забезпечення для автоматизації бізнесу в гастрономічній і роздрібній сферах, таких як ресторани, кафе, бари, магазини та інші заклади. Poster пропонує комплексний підхід до автоматизації обліку, керування персоналом, контролю за продажами, управлінням складом та іншими операційними процесами в гастрономічному бізнесі [9].

Основні особливості та переваги системи Poster:

– простий та зручний інтерфейс: Poster має легкий у використанні інтерфейс, що дозволяє швидко освоїти систему та ефективно використовувати її в роботі;

– функціональність: Poster пропонує широкий спектр функціональних можливостей, таких як керування замовленнями, облік продажів, контроль за складом, управління знижками та акціями, розрахунок зарплати персоналу, аналітика та звіти, інтеграція з онлайн-замовленнями та багато іншого;

– гнучкість та налаштування: Poster дозволяє налаштовувати систему під потреби конкретного бізнесу, включаючи налаштування меню, знижок, акцій, робочого графіка персоналу та інших параметрів;

– хмарна технологія: Poster працює в хмарному середовищі, що дозволяє отримувати доступ до системи з будь-якого пристрою з Інтернет-

підключенням, забезпечує резервне копіювання даних та забезпечує високий рівень безпеки;

– мобільні застосунки: Poster має мобільні застосунки для iOS та Android, що дозволяють керувати бізнесом з мобільних пристроїв, зокрема приймати замовлення, відстежувати стан замовлень, контролювати роботу персоналу, отримувати звіти та аналітику, та багато іншого;

– інтеграція з платіжними системами: Poster має можливість інтеграції з різними платіжними системами, що дозволяє приймати платежі в різних форматах, таких як безготівкові платежі, онлайн-платежі, карткові платежі тощо;

– підтримка клієнтського сервісу: Poster надає високий рівень підтримки клієнтів, включаючи технічну підтримку, навчання та консультування щодо оптимального використання системи;

– розширення функціональності: Poster має можливість розширення функціональності за допомогою додаткових модулів, таких як програма лояльності, онлайн-замовлення, управління доставкою, електронний документообіг та інші, що дозволяє відповідати розширеним потребам бізнесу;

– досвід на ринку: Компанія Poster має багаторічний досвід на ринку автоматизації гастрономічного бізнесу, що дозволяє їй розуміти потреби та вимоги клієнтів і розробляти відповідні рішення;

– партнерська мережа: Poster має розвинуту партнерську мережу, що дозволяє клієнтам отримати підтримку та обслуговування на місці в різних регіонах.

Загалом, компанія Poster пропонує потужну систему автоматизації бізнесу в гастрономічній та роздрібній сферах з багатофункціональним програмним забезпеченням, зручним інтерфейсом та додатковими можливостями, такими як інтеграція з платіжними системами, програмами лояльності, онлайн-замовленнями та іншими. Компанія Poster також надає високий рівень підтримки клієнтів та має широку партнерську мережу для

забезпечення задоволення потреб клієнтів в різних регіонах. Виходячи із результатів дослідження – програма Poster ідеальна для новачків, оскільки має максимально простий та інтуїтивно зрозумілий інтерфейс, проте простота робить програму недостатньо потужною з точки зору аналізу даних.

1.4 Постановка задачі

Розв’язання будь-якої задачі починається з її постановки, викладеної мовою чітко визначених математичних понять. При цьому слід добре уявити суть поставленої задачі, необхідні початкові дані та інформацію, що вважається результатами розв’язання [10].

Будь-який ресторан проводить облік товарів та продуктів на складі, отже у тому чи іншому випадку відбувається рух товарів у ресторані та їх списання.

Застосунок, що розробляється має дозволити користувачу (відповідальному за облік товарів та їх списання) вносити в систему дані, щодо поставок нових товарів та списувати зіпсовані, прострочені товари, тощо.

Об’єктом роботи є низка товарів, які використовуються у ресторані для приготування страв та реалізації.

Метою цієї роботи є створення програмного забезпечення, яке дозволить ресторану ефективно вести облік та списання продуктів.

Для досягнення мети необхідно вирішити наступні цілі:

– реалізувати GUI;

– створити модель бази даних;

– поєднати БД із GUI;

– реалізувати функції списання, додавання товарів;

– реалізувати автоматичне створення звітів.

2 АЛГОРИТМИ ТА ЗАСОБИ РЕАЛІЗАЦІЇ ЗАДАЧІ ОБЛІКУ ТА СПИСАННЯ ТОВАРІВ У СИСТЕМІ РЕСТОРАНУ

2.1 Описання засобів підтримки бази даних, що використовуються

2.1.1 Модель даних

Реляційна модель даних – це модель даних, де текстова чи числова інформація зображується за допомогою таблиць [11].

Однією з основних переваг реляційної моделі є її простота та універсальність. Це означає, що відношення можуть бути використані для зберігання даних з різних джерел та для вирішення різних завдань. Також реляційна модель забезпечує легкий доступ до даних та можливість проведення складних запитів.

Основними елементами реляційної БД є атрибути, кортежі та відношення. Відношенням називається деяка сукупність об’єктів, яка характеризується однаковим набором атрибутів. Зручно подавати відношення як таблицю, де кожний рядок є кортеж і кожний стовпець є атрибут. Стовпці таблиці – це елементи даних а рядки – записи [12].

Потенційний ключ – мінімальна множина атрибутів, що є підмножиною заголовка даного відношення, складене значення яких унікально, визначає кортеж відношення [13].

Первинний ключ – потенційний ключ, який обраний для унікальної ідентифікації кортежів усередині відношення (PK). Зв’язок між таблицями встановлюється шляхом створення відношення між стовпцями в таблиці та ключовими полями в інших таблицях [13].

Інші моделі даних, такі як ієрархічна модель та мережева модель, базуються на інших підходах до зберігання та організації даних. У ієрархічній моделі дані організовані у вигляді дерева, де кожен вузол може мати багато дочірніх вузлів. У мережевій моделі дані організовані у вигляді графа, де кожен вузол може мати багато пов’язаних з ним вузлів.

Незважаючи на те, що ієрархічна модель та мережева модель можуть бути корисними для певних завдань, реляційна модель є більш універсальною та більш популярною у світі баз даних.

Кожен атрибут визначає тип даних, що разом з областю його значень називається доменом. Домен також визначає обмеження, які можуть бути застосовані до значень цього стовпця. Наприклад, домен «цифри» може мати значення, які є цифрами від 0 до 9, а домен «рядки» може обмежуватися максимальною довжиною рядка [14].

Домени дозволяють декларувати типи даних, які можуть бути збережені в таблиці, та обмеження, які можуть бути застосовані до цих значень. Використання доменів дозволяє забезпечити правильність даних у таблицях в базі даних. Крім того, використання доменів робить розробку таблиць більш простою та зрозумілою, тому що вони дозволяють чітко визначити типи даних та їх обмеження.

Відношення R, на n множин $D_1, \dots D_n$ є множиною впорядкованих кортежів $d_1, \dots, d_n$ таких, що $d_1 \in D_1, d_2 \in D_2, \dots, d_n \in D_n$. Множини $D_1, \dots D_n$ називаються доменами відношення R.

Операція декартового добутку визначає нове відношення, яке є результатом конкатенації (зчеплення) кожного кортежу відношення R_1 з кожним кортежем відношення R_2. Підмножина декартового добутку – це множина, яка складається з певних кортежів, вибраних з декартового добутку, згідно з певними умовами [15].

Декартовим добутком доменів $D_1, D_2, \dots, D_n$ називається множина усіх кортежів довжини k, таких, що $d_1 \in D_1, d_2 \in D_2, \dots, d_n \in D_n$:

$$D_1 \times D_2 \times \dots \times D_n = \{< d_1, d_2, \dots, d_n > | d_i \in D_i, i = 1 \dots n\}. \qquad (2.1)$$

Наприклад, якщо ми маємо два домени: домен «кольори» (червоний, зелений, синій) та домен «форми» (круг, квадрат, трикутник), то декартовим добутком цих доменів буде множина { (червоний, круг), (червоний, квадрат),

(червоний, трикутник), (зелений, круг), (зелений, квадрат), (зелений, трикутник), (синій, круг), (синій, квадрат), (синій, трикутник) }.

Підмножина декартового добутку списку доменів – це множина усіх кортежів, які можна сформувати з елементів цих доменів, відповідно до певної умови.

Кожен домен має своє ім'я. Нехай множина Na – множина імен атрибутів, множина Nr – множина імен відношень.

Тоді, для будь-якого $A \in Na$ існує множина значень цього атрибута, яка має співпадати з одним із доменів $D_1, D_2, \ldots, D_k$ і вважається заданою функція присвоєння множини значень:

$$\text{Domen: Na} \rightarrow \{D_1, \ldots, D_k\}. \qquad (2.2)$$

Отже, пара $< A, DomenA >$, де $A \in$ Na, називається атрибутом A з областю значень $DomenA$. Елемент $d \in DomenA$ є значенням атрибута A.

Прикладом підмножини декартового добутку може бути множина {(червоний, круг), (зелений, квадрат)}, що складається з двох кортежів, вибраних з декартового добутку доменів «кольори» та «форми», і задовольняє умову, наприклад, що кортеж повинен містити один елемент з кожного з цих доменів та бути певної форми або кольору. Це і є реляційна модель.

Список імен атрибутів відношення $R(A_1, \ldots, A_n)$, де $R \in Nr,\ A_1, \ldots, A_n \in Na$ називається схемою відношення з ім'ям R. Сукупність схем відношень називається системою бази даних, а їх поточне значення — базою даних.

РБД є найбільш популярним типом баз даних, який використовується в сучасних інформаційних системах.

Вибір РБД для проєкту обґрунтовується наступними аспектами:

– перш за все, РБД забезпечує надійність та цілісність даних. Вона дозволяє зберігати дані у вигляді нормалізованих відношень, що зменшує ризик втрати даних та дублювання інформації. Крім того, РБД забезпечує

можливість контролювати доступ до даних, встановлювати права доступу для різних користувачів та груп користувачів;

– другим аспектом, що обґрунтовує вибір РБД, є її масштабованість. РБД може працювати з великою кількістю даних та користувачів, що забезпечує підтримку динамічних проєктів, що продовжують розвиватися;

– третім аспектом є підтримка мови SQL, яка дозволяє легко взаємодіяти з базою даних, створювати запити та отримувати результати.

Крім того, РБД забезпечує можливість резервного копіювання даних, що зменшує ризик втрати інформації у випадку аварії.

Отже, використання РБД для проєкту забезпечує надійність, масштабованість та легку взаємодію з базою даних.

Реляційна база даних – це набір нормалізованих відношень *r*, які мають різні назви [16].

Реляційна база даних працює за правилами реляційної алгебри, яка являє собою сукупність операцій високого рівня над відношеннями.

Реляційна алгебра є алгеброю в строгому класичному розумінні її визначення. Елементами основної множини є реляційні відношення. У зв'язку з цим, операції алгебри можуть вкладатися одна в одну, тобто аргументом певної операції може бути результат виконання іншої операції. Це дає можливість записувати запити довільного рівня складності у вигляді виразів, що містять вкладені одна в одну операції [17].

Об'єднанням сумісних реляційних відношень R_1 і R_2 (позначається як $R_1 \cup R_2$) називається таке реляційне відношення R, що містить кортежі обох поєднуваних відношень, але без повторень:

$$R = R_1 \cup R_2 = \{r | r \in R_1 \vee r \in R_2\}. \quad (2.3)$$

Перетином сумісних реляційних відношень R_1 і R_2 (позначається як $R_1 \cap R_2$) називається таке реляційне відношення R, яке містить кортежі, що входять до складу обох операндів:

$$R = R_1 \cap R_2 = \{r | r \in R_1 \wedge r \in R_2\}. \quad (2.4)$$

Різницею сумісних реляційних відношень R_1 і R_2 (позначається як R_1-R_2) називається реляційне відношення R, що містить ті кортежі з першого операнду R_1, яких немає у другому операнді R_2:

$$R = R_1 - R_2 = \{r | r \in R_1 \wedge r \notin R_2\}. \quad (2.5)$$

Декартовим добутком двох відношень R_1 і R_2 є множина усіх кортежів n таких, що конкатенація r_1, що належить R_1, та кортежу r_2, що належить R_2:

$$R_1 \oplus R_2 = \{(r_1, r_2) | (r_1 \in R_1 \setminus r_2 \in R_2)\}. \quad (2.6)$$

Реляційні бази даних (РБД) є однією з найбільш поширених та важливих технологій зберігання та обробки даних. Основні плюси та недоліки РБД описані нижче.

Плюси РБД:

– простота використання та зручність управління: РБД забезпечують простоту та зручність використання баз даних завдяки стандартизованому SQL і майже універсальній можливості звернення до даних у всіх мовах програмування;

– спрощення структури даних: у РБД дані зберігаються в структурах відношень, що спрощує структуру даних і зменшує кількість даних, що зберігаються;

– збереження цілісності даних: РБД використовують механізми контролю цілісності даних для забезпечення того, що дані в відношеннях відповідають заданим правилам та обмеженням;

– забезпечення безпеки даних: РБД забезпечують можливість встановлення різних рівнів доступу до даних та захисту від несанкціонованого доступу;

– підтримка транзакцій: РБД забезпечують механізми підтримки транзакцій, що дозволяє забезпечити атомарність, стійкість, ізольованість та цілісність даних.

Недоліки РБД:

– складність структури бази даних: Структура РБД може стати досить складною зі зростанням розміру бази даних, що може привести до складнощів у збереженні та управлінні даними;

– відносно повільна швидкість: РБД можуть працювати дещо повільніше в порівнянні з іншими технологіями зберігання даних, такими як NoSQL;

– високі вимоги до обладнання: РБД можуть вимагати високопродуктивних серверів та іншого обладнання для забезпечення ефективної роботи з великими обсягами даних;

– обмежена гнучкість: РБД можуть бути обмежені у своїй гнучкості, зокрема у відношенні до додавання або зміни структури даних, що може вимагати значних зусиль для їх модифікації;

– обмежена масштабованість: РБД можуть мати обмежену масштабованість у випадку, якщо база даних зростає дуже швидко або коли вимагається велика кількість звернень до бази даних;

– технологія баз даних NoSQL зберігає інформацію в документах JSON замість стовпців і рядків, які використовуються реляційними базами даних [18].

Отже, РБД мають свої переваги та недоліки, і їх вибір для конкретного проєкту залежить від його вимог та потреб. Оскільки, поточна робота не вимагає обробки великої кількості нерівномірних даних, то використання РБД є оптимальним варіантом.

2.1.2 Концептуальна модель

Існує кілька типів концептуальних моделей, які використовуються для проєктування баз даних.

Модель сутність-зв’язок (Entity-Relationship Model, ERM) – це найбільш поширена концептуальна модель, яка використовується для проєктування реляційних баз даних. У моделі сутність-зв’язок сутності представляються у вигляді відношень, а зв’язки – у вигляді зовнішніх ключів. Модель сутність-зв’язок дозволяє зображувати складні структури даних та зв’язки між ними.

Модель об’єктів (Object-Oriented Model, OOM) – ця модель використовується для проєктування об’єктно-орієнтованих баз даних. У моделі об’єктів дані представляються у вигляді об’єктів, а зв’язки між об’єктами представляються у вигляді методів, які можуть виконувати дії з даними.

Модель ієрархічної бази даних (Hierarchical Model, HM) – ця модель використовується для проєктування ієрархічних баз даних, де кожен запис має посилання на свого батька. Модель ієрархічної бази даних дозволяє ефективно зберігати дані, що мають ієрархічну структуру, наприклад, дерева.

Модель мережевої бази даних (Network Model, NM) – ця модель використовується для проєктування мережевих баз даних, де кожен запис може мати кілька посилань на інші записи. Модель мережевої бази даних дозволяє зберігати складні зв’язки між даними, такі як багато-до-багатьох зв’язки.

Один з головних недоліків моделі сутність-зв’язок (ERM) полягає в тому, що вона не може детально описати дії, що відбуваються в системі. Ця модель призначена лише для відображення сутностей та зв’язків між ними. Тому для опису дій може бути потрібна додаткова модель, наприклад, діаграма послідовності або діаграма діяльності.

Головний недолік суб’єкт-орієнтованої моделі полягає в тому, що вона може вимагати великої уваги до визначення класів суб’єктів. Тобто, для того щоб правильно визначити класи суб’єктів, потрібно проводити глибокий аналіз даних та досліджувати бізнес-процеси організації. Якщо цей аналіз не проводиться достатньо ретельно, може статися так, що класи суб’єктів не будуть відповідати реальному стану речей, що в свою чергу призведе до проблем з функціонуванням бази даних.

Ієрархічна модель даних, має головний недолік: Обмежені можливості виразності даних. Зазвичай у ІМ відсутній механізм зв’язків багато-до-багатьох і вона підтримує тільки ієрархічні зв’язки. Це робить її менш гнучкою, ніж інші моделі даних, зокрема реляційну модель, що може використовувати зв’язки багато-до-багатьох для зберігання більш складних відносин між даними.

Головним недоліком мережевої моделі є її складність та високий рівень абстракції. Мережева модель може бути складною для розуміння та використання, особливо для користувачів без технічної освіти. Крім того, мережева модель може бути важко підтримувати та змінювати через необхідність редагування великої кількості записів, пов’язаних зі зв’язками між елементами. Це може призвести до складнощів при додаванні, видаленні та оновленні даних у мережевій моделі. Крім того, мережева модель не підтримує легке додавання нових зв’язків між елементами, оскільки це може призвести до зміни всієї структури моделі.

Модель сутність-зв’язок (ER-модель) є однією з найбільш популярних концептуальних моделей для проєктування баз даних і була обрана для розробки поточного застосунку. Основна перевага ER-моделі полягає в тому, що вона дозволяє легко визначити структуру бази даних та відношення між сутностями, що спрощує процес проєктування та розробки. Крім того, ER-модель дозволяє легко вносити зміни у базу даних та додавати нові відношення, що є важливим для вдосконалення та змінення застосунку в процесі розробки.

2.2 Розробка концептуальної моделі

Перш за все, необхідно розробити модель зв’язку між сутностями інформаційної системи «Ресторан».

Діаграма ER – це блок-схеми, які ілюструють, як «сутності» (люди, об’єкти або концепції) ставляться один до одного в системі. ER-діаграма – це та модель, яка найчастіше використовуються для розробки або налагодження реляційних баз даних [19].

У діаграмі використовується набір геометричних символів, таких як прямокутник, ромб, овал і лінії, для відображення взаємозв’язку об’єктів, відносин і їх атрибутів [20].

Для великих баз даних побудова ER-моделі дозволяє уникнути помилок проєктування, які надзвичайно важко виправляти, особливо, якщо база даних вже експлуатується чи на стадії тестування. Помилки в розробці структури бази даних може призвести до перебудови коду програмного забезпечення, що керує цією базою даних. У результаті час, кошти та людські ресурси будуть використані неефективно [21].

В розробленій структурі БД існує 5 сутностей, а саме:

– товар. Товар є однією з головних сутностей, оскільки усі операції відбуваються з товаром. Товар представляє собою абстрактне поняття якоїсь продукції, яку можна замовити;

– склад. Склад є сутністю, яка містить перелік товарів, що на даний момент знаходяться на складі. Вона дозволяє контролювати наявність товару і його кількість на складі;

– списання. Списання є сутністю, яка містить список товарів, що було списано. Ця сутність є важливою для обліку продукції, що втратила свою споживчу властивість або не відповідає якісним стандартам. Користувачем може бути налаштоване автоматичне видалення даних;

– постачальник. Постачальник є сутністю, що містить усі постачальників, які знаходяться в базі. У кожного постачальника є список

товарів, які він постачає. Це дозволяє контролювати наявність постачальників та їхніх продуктів;

– замовлення. Замовлення є сутністю, що зберігає усі замовлення. Інформація щодо складу замовлення знаходиться в окремій таблиці та з часом видаляється з метою швидкодії БД. Ця сутність є важливою для контролювання продажу товарів та їхньої кількості, а також для створення звітів. Користувачем може бути налаштоване автоматичне видалення даних;

– використання. Сутність аналогічна списанню, але вона містить ті товари, які було використано на виробництві.

На рисунку 2.1 зображено модель зв’язку між сутностями інформаційної системи.

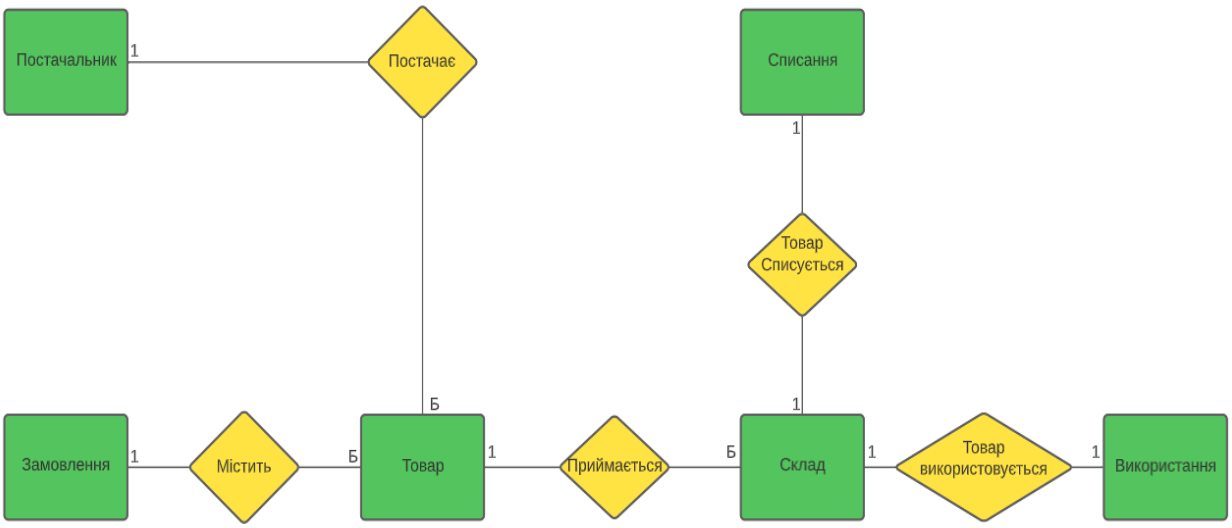

Рисунок 2.1 – ER-модель сутностей

2.3 Розробка повної структури БД

Повна схема БД обліку товарів у ресторані зображає усі відношення та атрибути табличного простору «Ресторан» (рис. 2.2).

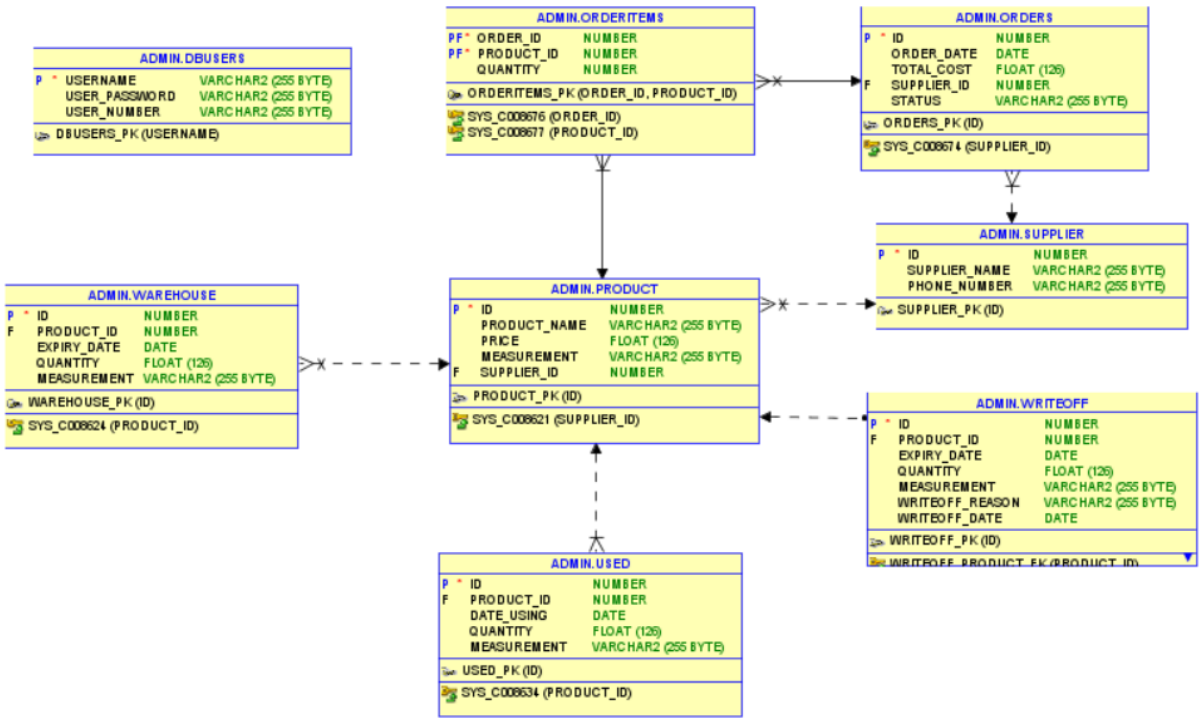

Рисунок 2.2 – Схема БД

Розглянемо кожне відношення окремо.

DBUSERS – відношення, що містить інформацію про усіх користувачів:

– Username – унікальний ідентифікатор користувача, varchar (255 byte), PK;

– Password – пароль користувача, що зберігається у вигляді хеш-значення, яке є результатом роботи хеш-функції SHA-256, varchar (255 byte).

Хешування (англ. Hashing) – перетворення вхідного масиву даних довільної довжини в вихідний бітовий рядок фіксованої довжини. Такі перетворення також називаються хеш-функціями, а їх результати називають хешем [22].

Хеш-функції є основним компонентом цифрового підпису, який є важливим аспектом перевірки справжності та цілісності даних [23]. Окрім цього, хеш застосовують для безпечного зберігання паролів у випадку зловмисного доступу до баз даних.

У якості хеш-функції було обрано SHA-256, оскільки вона є основною алгоритму proof of work у «Bitcoin», тобто надійною.

Надійність функції полягає у стійкості до колізій. Колізія – це ситуація, при якій $H(k_1) = H(k_2)$, тобто хеш однієї послідовності бітів дорівнює хешу іншої послідовності. Для поточної хеш-функції колізія ще не була знайдена.

Такий принцип використання хешу означає, що пароль відомий виключно користувачу. У випадку втрати, єдина можливість – це відновлення.

PRODUCT – відношення, що містить інформацію, щодо усіх товарів, які раніше були замовлені у постачальників:

– ID – унікальний ідентифікатор товару, який видається автоматично при додаванні рядка, number, PK;

– Product_Name – назва продукту, varchar (255 byte);

– Price – ціна, за 1 kg/pcs продукту, float;

– Measurement – одиниці вимірювання, «kg» або «pcs»;

– Supplier_id – унікальний ідентифікатор постачальника, зв’язок із відношенням SUPPLIER (FK).

SUPPLIER – відношення усіх постачальників:

– Id – унікальний ідентифікатор постачальника, який видається автоматично при додаванні рядка, number, PK;

– Supplier_Name – ім’я постачальника, varchar (255 byte);

– Phone_Number – номер телефону постачальника, varchar (255 byte).

ORDERITEM – відношення усіх товарів з кожного замовлення, що коли-небудь були зроблені (вбудовано функцію автоматичного видалення записів із метою оптимізації):

– Order_Id – унікальний ідентифікатор замовлення, зв’язок із відношенням ORDERS (PK);

– Product_Id – унікальний ідентифікатор продукту, зв’язок із відношенням PRODUCTS (PK);

– Quantity – кількість продукту, що було замовлено, number.

Відношення має складений первинний ключ (PK), що зумовлює унікальність пари замовлення-товар, тобто в одному замовленні не може бути дві позиції(два товари) з однаковим id.

ORDERS – усі замовлення, що були зроблені (аналогічна відношенню ORDERITEM функція автоматичного видалення):

– Id – унікальний ідентифікатор замовлення, який видається автоматично при додаванні рядка, number, PK;

– Order_Date – дата створення замовлення (після встановлення значення ‘Completed’ у статус – дата автоматично змінюється на дату виконання замовлення), date;

– Total_Cost – вартість усього замовлення, float;

– Status – поточний статус замовлення, pending або completed.

WAREHOUSE – усі товари, що на даний момент знаходяться на складі:

– Id – унікальний ідентифікатор товару на складі, який видається автоматично при додаванні рядка, number, PK;

– Product_id – унікальний ідентифікатор товару, зв’язок із відношенням PRODUCT (FK);

– Expiry_date – строк придатності товару, date;

– Quantity – кількість товару, що знаходиться на складі, float;

– Measurement – одиниця вимірювання товару, («kg» або «pcs»).

WRITEOFF – відношення списаних товарів, користувачем налаштовується видалення даних в автоматичному режимі:

– Id – унікальний ідентифікатор товару на складі, який видається автоматично при додаванні рядка, number, PK;

– Product_ID – унікальний ідентифікатор товару, зв’язок із відношенням PRODUCT (FK);

– Expiry_date – строк придатності товару, date;

– Quantity – кількість товару, що знаходиться на складі, float;

– Measurement – одиниця вимірювання товару, («kg» або «pcs»);

– Writeoff_date – дата списання товару, date.

USED – відношення використаних товарів, користувач у ручному режимі виносить товари зі складу до використаних:

– Id – унікальний ідентифікатор товару в відношенні використання, який видається автоматично при додаванні рядка, number, PK;

– Product_ID – унікальний ідентифікатор товару, зв’язок із відношенням PRODUCT (FK);

– Date_Using – дата використання товару, date;

– Quantity – кількість товару, яка була використана, number;

– Measurement – одиниця вимірювання, («kg» або «pcs»).

База даних, окрім збереження інформації про товари в автоматичному режимі виконує ряд операцій.

Автоматичне внесення значення одиниці вимірювання товару, при додаванні його на склад.

Лістинг 2.1 Тригер внесення значення одиниці вимірювання:

```
create or replace TRIGGER trg_set_warehouse_measurement
BEFORE INSERT ON Warehouse
FOR EACH ROW
BEGIN
SELECT measurement INTO :new.measurement
FROM Product
WHERE id = :new.product_id;
END;
```

Автоматична зміна дати замовлення при змінені значення атрибуту ‘status’ на ‘completed’.

Лістинг 2.2 Зміна дати замовлення:

```
create or replace TRIGGER trg_set_order_date
```

BEFORE UPDATE OF status ON Orders

FOR EACH ROW

BEGIN

IF :NEW.status = ‘Completed’ THEN

:NEW.order_date := SYSDATE;

END IF;

END;

Автоматичне внесення даних до відношення «WriteOff» при видаленні їх зі складу, SQL.

Лістинг 2.3 Автоматичне внесення даних до списання:

create or replace TRIGGER TRG_DELETE_WAREHOUSE

AFTER DELETE ON Warehouse

FOR EACH ROW

BEGIN

INSERT INTO Writeoff (product_id, expiry_date, quantity, measurement, writeoff_date)

VALUES (:OLD.product_id, :OLD.expiry_date, :OLD.quantity, :OLD.measurement, SYSDATE);

END;

Дані неможливо просто видалити із складу, вони у будь-якому разі потрапляють до відношення «WriteOff». Докладніше про тригери у підрозділі 3.1.

Розроблена структура бази даних є ключовим елементом для оптимального керування обліком та списанням товарів у системі ресторану. БД відіграє важливу роль у зборі, збереженні та обробці інформації про продукти, замовлення та клієнтів. Правильно спроєктована БД дозволяє

швидко та ефективно отримувати потрібну інформацію та створювати різноманітні звіти для аналізу роботи ресторану.

Хоча застосунок є прототипом та може поступатись відомим аналогам, його функціонал є повноцінним для малого ресторанного бізнесу. За допомогою розробленої структури БД, ресторан може вести детальний облік продуктів та замовлень, відстежувати рух товарів та швидко реагувати на зміни в попиті.

Крім того, за допомогою звітів, ресторан може аналізувати фінансову ефективність та ефективність роботи персоналу, що дозволяє оптимізувати роботу ресторану та забезпечити його успішність на ринку.

3 СТВОРЕННЯ ПРОГРАМНОГО ПРОДУКТУ

3.1 Розробка структури БД

Першим етапом розробки застосунку є розробка конфігурації, структури та наповнення БД. Для початку, використовуючи мову PL/SQL, створюємо табличний простір для розміщення відношень. Наступним кроком є створення користувача-адміністратора бази даних, який буде керувати даними у відношеннях. Після початкової конфігурації можна виконати під’єднання до табличного простору через інтегроване середовище розробки SQLDeveloper та початки створювати та наповнювати відношення.

Конфігурація відношень відбувається виключно із використанням мови SQL. У лістингу 3.1 наведено приклад створення відношень.

Лістинг 3.1 Приклад створення відношень:

```
CREATE TABLE Product (
id NUMBER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
product_name VARCHAR2(255),
price FLOAT,
measurement VARCHAR2(255),
supplier_id NUMBER,
FOREIGN KEY (supplier_id) REFERENCES Supplier(id)ON DELETE CASCADE);
CREATE TABLE Warehouse (
id NUMBER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
product_id NUMBER,
expiry_date DATE,
quantity FLOAT,
measurement VARCHAR2(255),
```

FOREIGN KEY (product_id) REFERENCES Product(id)ON DELETE CASCADE);

У цьому фрагменті програмного коду створюються два відношення: Product та Warehouse. В обох відношеннях PK – це атрибут «id», який встановлюється автоматично засобами СУБД під час додавання у БД кортежу. Відношення Product пов’язана із відношенням Supplier, через атрибут supplier_id (у кожного товару є постачальник). Аналогічно, відношення «Warehouse» пов’язано із відношенням «Product» через атрибут product_id, адже на складі може буде низка однакових товарів у різній кількості та із різним терміном придатності, а у таблиці «Product» товар класифікується за постійними атрибутами, як: назва, одиниця вимірювання, постачальник та ціна у постачальника. За логікою інформаційної системи, якщо вартість товару змінено – створюється новий товар (існує можливість змінити ціну товару в базі даних, проте це загрожує цілісності даних, оскільки система отримує вартість товару із БД у момент створення звіту, і може виникнути ситуація, коли товар замовлено і отримано за однією вартістю, а в звіті вказано іншу вартість).

Після створення відношення необхідно заповнити відношення тестовими даними.

Лістинг 3.2 Фрагмент коду заповнення відношень тестовими кортежами:

INSERT INTO Product (product_name, price, measurement, supplier_id) VALUES (‘Цукор ваг.’, 29, ‘kg’, 1);

INSERT INTO Warehouse (product_id, expiry_date, quantity) VALUES (1 , TO_DATE(‘2023-07-30’, ‘YYYY-MM-DD’), 100);

INSERT INTO DBUSERS (username, user_password, user_number) values (‘admin’,’8c6976e5b5410415bde908bd4dee15dfb167a9c873fc4bb8a81f6f2ab448a918’, ‘XXXXXXXXXXX’);

Замість XXXXXXXXXXXX вказано персональний номер телефону для реалізації автентифікації (детальніше у підрозділі 3.3).

Пароль у БД зберігається у вигляді хеш-значення, що генерує функція SHA-256. Це зроблено з метою безпеки даних. У випадку отримання даних із БД злочинниками – вони не зможуть отримати пароль, оскільки хешування – невідворотний процес. Під час авторизації у застосунку – користувач вводить дані для входу, після чого система автоматичне генерує хеш-значення паролю і порівнює із записом у БД. Це можливо, оскільки однією із головних властивостей хеш-функцій є генерація ідентичних послідовностей у випадку, якщо на вхід подаються однакові дані.

Окрім керування даними, розроблена структура БД дозволяє виконувати ще низку функцій. У лістингах 2.1 – 2.3 наведено код тригерів із БД.

Тригер TRG_SET_WAREHOUSE_MEASUREMENT виконується BEFORE INSERT (перед додаванням) нового кортежу до відношення. Цей тригер автоматично витягає із таблиці «Product» одиницю вимірювання відповідного товару і встановлює це значення кортежу, що додається. Таке дублювання даних не є необхідним і це можна, навіть, вважати за помилку, проте це було зроблено для візуального спрощення роботи із SQLDeveloper, а також для зменшення кількості програмного коду під час розробки застосунку.

Тригер TRG_DELETE_WAREHOUSE виконується AFTER DELETE (після видалення) товару із складу. Цей тригер додає товар, який видалено зі складу до списання, при чому, виставляє датою списання SYSDATE (поточна дата системи), а причиною списання виставляється значення за замовчуванням «Expired». Це необхідно, оскільки застосунком, що розробляється (далі – Fast&Easy) передбачено автоматичне видалення товарів, в яких закінчився термін придатності, а під час видалення вони автоматично вносяться до відношення списання. Такий підхід значно економить час розробки та кількість програмного коду, який реалізує процес списання.

Тригер TRG_SET_ORDER_DATE виконується BEFORE UPDATE (перед оновленням) даних у відношенні. У випадку, якщо оновлюється

атрибут «status» відношення «Orders» і встановлюється значення «Completed» – дату замовлення буде змінено на поточну. Такий функціонал реалізовано з метою зменшення кількості атрибутів, адже після виконання замовлення відсутня необхідності мати дату замовлення, необхідно мати лише дату отримання. Тим самим, значення атрибуту «order_date» завжди актуальне (для замовлень, що знаходяться в очікуванні – це дата замовлення, а для виконаних замовлень – дата отримання). Для атрибуту «status» існує третій можливий статус «Canceled». Цей статус означає, що замовлення було скасовано однією із сторін. Поточна реалізація не передбачає відображення скасованих замовлень у звітах або у застосунку, доступ до цих даних можливо отримати лише через напряму із БД.

3.2 Програмна реалізація

Другим етапом розробки є реалізація графічного інтерфейсу та функціональності застосунку. Для цього необхідно, спочатку, виконати підключення до бази даних. Відповідає за це клас DBUtil, який використовує бібліотеку Oracle JDBC для встановлення з’єднання із БД.

Лістинг 3.3 Встановлення з’єднання із БД:

```
private final static String url="jdbc:oracle:thin:@localhost:1521 :ORCL";
private final static String user="admin";
private final static String password="admin";
connection = DriverManager.getConnection(url,user,password);
```

Після встановлення з’єднання, об’єкт «connection» зберігається і використовується кожного разу, коли застосунок виконує операцію із БД.

Графічний інтерфейс розроблено засобами CSS та Scene Builder.

CSS – це формальна мова опису зовнішнього виду документа написана з використанням мови розмітки [24].

JavaFX Scene Builder – це інструмент візуального макета, який дозволяє користувачам швидко розробляти користувацькі інтерфейси застосунків JavaFX без кодування. Користувачі можуть перетягувати компоненти інтерфейсу користувача в робочу область, змінювати їхні властивості, застосовувати таблиці стилів, а код FXML для макета, який вони створюють, автоматично генерується у фоновому режимі [25].

Процес розробки застосунків на JavaFX схожий на процес розробки сайтів. Під час створення розмітки сайтів використовується HTML, мова розмітки, а під час створення JavaFX-застосунків – FXML, скриптова мова, що поєднує у собі текстову розмітку, візуальне відображення контейнерів та шаблонні стилі. Як і у стилізації сайтів, для стилізації елементів у JavaFX використовується CSS із схожими правилами, які починаються з «-fx».

Для застосунку Fast&Easy було розроблено унікальний логотип (рис. 3.1), що відображено у вікні авторизації та у лівій частині головного вікна застосунку.

Рисунок 3.1 – Логотип застосунку

Під час розробки було використано бібліотеку AnimateFX, яка дозволяє застосовувати різні візуальні ефекти до елементів. До кнопки «Login» було додано ефект поштовхування, що активується під час неправильного введення даних авторизації і відтворює переміщення кнопки по осі X у різні сторони декілька разів.

Також, у вікні авторизації було реалізовано слухача клавіши «Enter». Натискання клавіши «Enter» викликає ту саму функцію, що і кнопка «Login».

Лістинг 3.4 Слухач натискання клавіши «Enter»:

```
scene.getRoot().addEventHandler(KeyEvent.KEY_PRESSED, event -> {
        if(event.getCode() == KeyCode.ENTER){
           LoginController.controller.login();
        }
});
```

Під час автентифікації користувач може натиснути кнопку «Forgot password». Система запропонує йому відновити пароль, для чого необхідно буде ввести свій логін. Якщо введений логін існує – на номер телефону, що вказано в БД буде відправлено повідомлення із тимчасовим паролем, який користувач може змінити у вікні налаштувань. Такий функціонал реалізовано через сервіс розсилки СМС-повідомлень «SMS-flu.ua».

Для реалізації необхідно створити кабінет користувача, поповнити баланс та активувати HTTP-протокол у налаштуваннях, після чого зберегти свій API-ключ.

Для використання сервісу, за посиланням «https://sms-fly.ua/api/v2/api.php» необхідно відправити POST-запит у вигляді JSON-файлу із даними до відправки.

Лістинг 3.5 Зміст JSON-файлу, який відправляється на сервер:

```
{
 "auth": {
     "key": "specialsecretapikey"
 },
 "action": "SENDMESSAGE",
 "data": {
```

"recipient": "380501234567",

"channels": ["sms"],

"sms": {

"source": "MySMSSource",

"ttl": 5,

"text": "SMS text"

}

}

}

Опис полів запиту:

– *specialsecretapikey* – це API-ключ, який необхідно було скопіювати у налаштуваннях;

– *action* – дія, яку необхідно виконати;

– *data* – дані до відправки;

– *recipient* – номер телефону отримувача;

– *channels* – канали відправки (існує можливість відправки через Viber);

– *sms* – описання інформації до відправки у СМС-повідомленні;

– *source* – ім’я відправника, можливо вказати лише зареєстроване ім’я, а реєстрація відбувається лише для компаній, отже ім’я залишається за замовчуванням;

– *ttl* – час життя повідомлення (час, який буде відбуватись спроба відправити повідомлення, якщо користувач поза зоною);

– *text* – текст повідомлення.

Реалізований функціонал дозволяє користувачу отримувати тимчасовий 8-знаковий пароль у випадку, якщо він не пам’ятає свій та змінювати його у налаштуваннях після авторизації.

При розробці системи обміну інформацією з БД рекомендується розробити окремі класи, які будуть забезпечувати доступ до БД всіх інших компонент системи [26].

Для взаємодії застосунку Fast&Easy із БД було створено низку класів, що відповідають об’єктам БД і зменшують кількість необхідних запитів до БД.

Клас – це опис об’єктів певного типу. На основі класів створюють об’єкти. Може існувати множина об’єктів, які належать одному класу. З іншого боку, може існувати клас без об’єктів, реалізованих на його основі [27].

Клас, який виконує спадкування, називається підкласом. Він успадковує всі змінні екземпляра і методи, визначені суперкласом, і додає свої власні унікальні елементи [28].

Для розуміння логіки застосунку необхідно розглянути кожен клас окремо.

Клас AbstractProduct – клас, який є абстрактним, існує для зменшення кількості програмного коду і уникання дублювання, містить у собі поля, що спільні для класів, які успадковуються від нього.

Клас WarehouseProduct – клас, який успадковується від AbstractProduct. Зберігає у собі інформацію продукту на складі (відношення «Warehouse»).

Клас WriteOffProduct – клас, який успадковується від AbstractProduct. Зберігає у собі інформацію списаного продукту (відношення «WriteOff»).

Клас Order – клас, який зберігає інформацію стосовно відповідного замовлення (відношення «Orders»).

Клас OrderProduct – клас, який зберігає інформацію стосовно відповідного товару замовлення (відношення «OrderItems»). У класі Order зберігається список об’єктів OrderProduct.

Клас Supplier – клас, що зберігає у собі інформацію про постачальника (відношення «Supplier»).

Клас Product – клас, який зберігає інформацію про продукти, які внесені в базу даних (відношення «Product»).

Клас CartProduct – клас, який необхідний для роботи програми. Немає прямого відношення до таблиці у БД. Успадковується від класу «Product». Єдина відмінність від класу «Product» – це поле «quantity». CartProduct – це

товар, який відображаються у списку замовлення, отже необхідно, щоб він мав кількість, що потребувало створення нового класу.

У більшості випадків, дані до вищевказаних класів вносяться під час ініціалізації програми, а під час, наприклад, видалення – видаляються із БД та із відповідного списку, без повторної ініціалізації через SQL-запит, що економить пам’ять та час процесору.

На відміну від інформації про товари, що зберігаються в БД, налаштування зберігаються на машині. Користувач має можливість власноруч обрати через який час видаляти дані із бази даних, а також політику видалення товарів, в яких закінчився термін придатності. Функціонал автоматичного очищення реалізовано з метою підтримки швидкодії БД. На поточний момент реалізовано 4 варіанти видалення даних: «Do Not Delete», «3 Months», «6 Months», «1 Year». Налаштувати видалення даних можливо окремо для товарів замовлення (OrderItems), замовлень (Orders) та історії використання (Used).

Вказані користувачем налаштування застосовуються при наступній ініціалізації застосунку та видаляють дані, у випадку відповідності критеріям видалення.

Окрім видалення застарілих даних, існує можливість списати товари зі складу, якщо в них вийшов термін придатності, автоматично. Реалізовано 4 варіанти налаштувань для видалення товарів з терміном придатності, що закінчився, а саме: «Do not write-off», «Notify after write-off», «Ask before write-off», «Write-off without notifying». Під час наступної ініціалізації застосунку буде списано (або не списано) товар за тим алгоритмом, який встановив користувач.

Fast&Easy дає користувачу можливість створювати звіти в залежності від його потреб. На поточний момент реалізовано 5 варіантів звітів:

– звіт списаних товарів за обраний період;

– звіт списаних з конкретної причини товарів за обраний період;

– звіт з інформацією про усіх постачальників;

– звіт виконаних замовлень за окремий період;

– звіт замовлень, що на поточний момент в очікуванні (Pending).

Усі звіти зберігаються у форматі .xlsx, що дозволяє відкривати їх програмою Microsoft Excel. За генерацію звітів відповідають методи утилітарного класу ReportGenerator, які витягають із БД необхідні дані відповідно до створюємого звіту та передають набір цих даних в вигляді списку списків до класу ExcelWorker. Клас використовує бібліотеку Apache POI Common, яка дозволяє йому створювати файли у форматі .xlsx.

Списком називається впорядкована множина, що складається зі змінної кількості елементів, до яких застосовні операції включення та виключення (вдосконалена версія масиву) [29].

Під час створення, назвою звіту вказується «Report» + поточні дата та час, а назвою аркушу таблиці – тип звіту, який генерується + обраний діапазон дат. Формат унікальної назви в залежності від поточного часу з точністю до секунд виключає можливість перезаписування старого звіту новим. Усі згенеровані звіти завжди зберігаються у кореневу папку проєкт, що робить їх пошук доволі прости завданням

3.3 Тестування застосунку

Під час ініціалізації відкривається вікно авторизації (рис. 3.2). Користувач вводить логін у поле «Username» та пароль у поле «Password». Написи «Username» та «Password» анімовані. При початку введення тексту до відповідного поля – поступово зміщуються догори (рис. А.1).

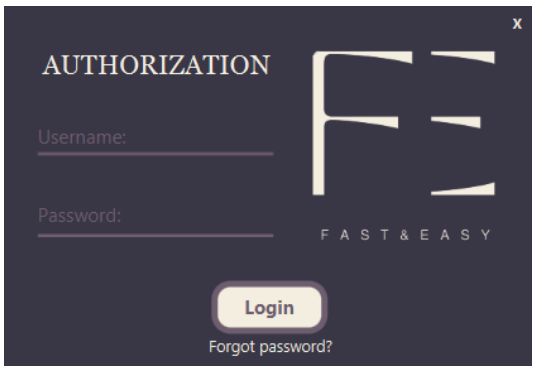

Рисунок 3.2 – Вікно авторизації

Припустимо, що користувач загубив пароль і намагається його відновити. Після натискання кнопки «Forgot password» вводимо логін у відповідне поле (рис. А.2). Після підтвердження (рис. 3.3) протягом 10 секунд на номер телефону із БД приходить СМС-повідомлення із тимчасовим паролем (рис. 3.4).

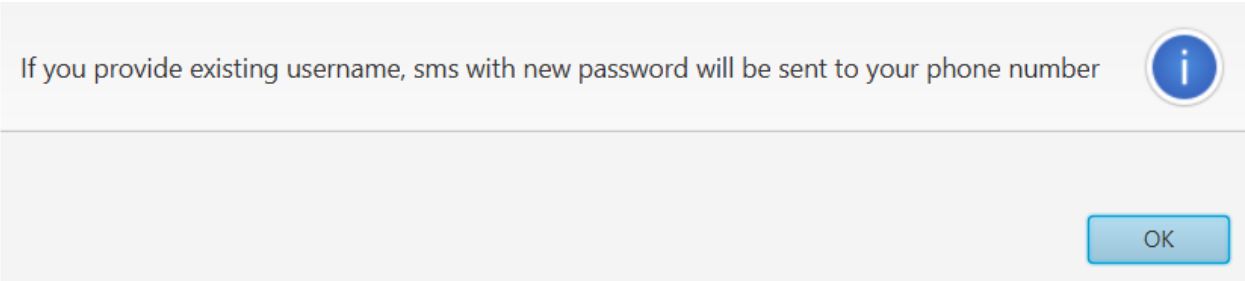

Рисунок 3.3 – Підтвердження відправки СМС-повідомлення

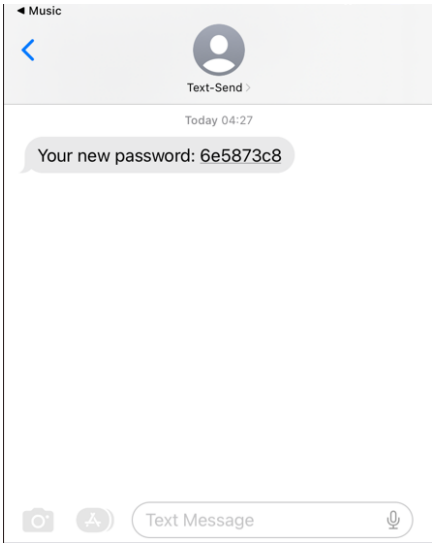

Рисунок 3.4 – СМС-повідомлення із тимчасовим паролем

Після введення тимчасового паролю із СМС-повідомлення авторизація відбувається успішно. Відтепер у БД зберігається нове хеш-значення, яке відповідає паролю «6e5873c8».

Оскільки авторизація відбулась успішно – на екрані відображається головний екран (рис. 3.5).

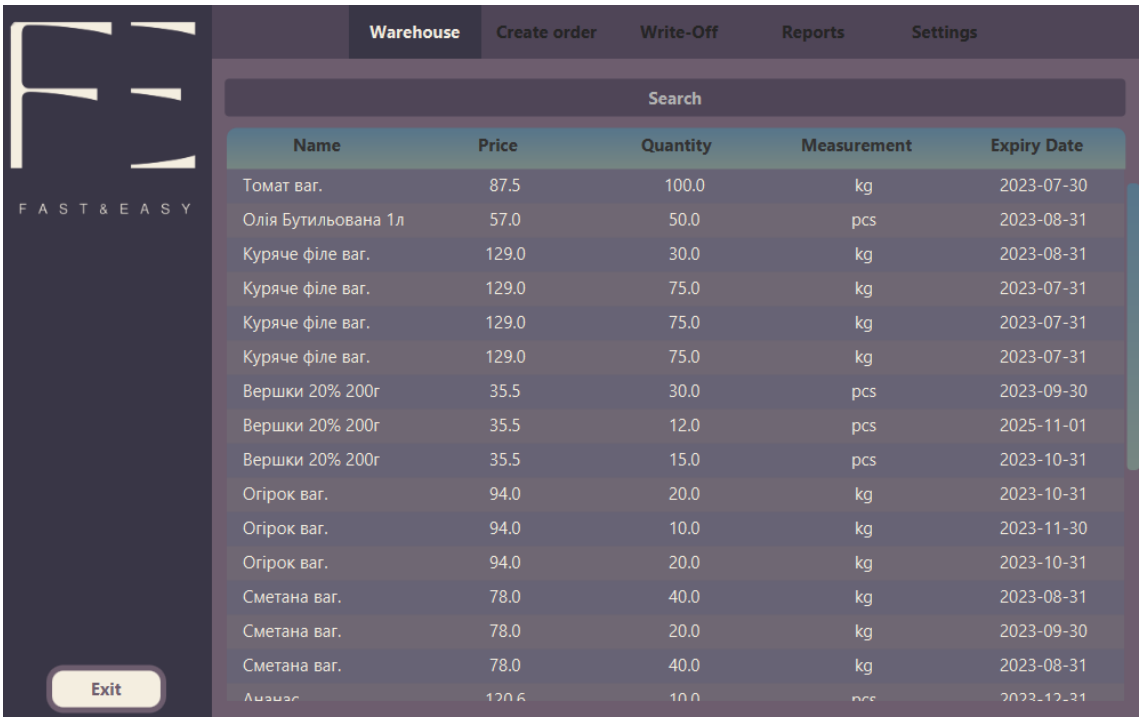

Рисунок 3.5 – Головний екран застосунку

На головному екрані відображається список поточних товарів які знаходяться на складі, їх ціна, одиниця вимірювання, кількість та термін придатності. Реалізовано можливість пошуку по списку, для цього необхідно ввести текст у поле «Search». Пошук до регістру не чутливий та відбувається за назвою. Введені символи у поле пошуку необов’язково мають бути початком назви товару. Наприклад, одним із товарів, які з’являться у пошуку при введені «ома» буде «Томат ваг.» (рис. А.3).

Кожен об’єкт у списку дозволяє із ним взаємодіяти. При натисканні правою кнопкою миші на елемент відкривається контекстне меню (рис. 3.6) із кнопками «Write-off» – списати товар або його частину, та «Use» – використати товар, або його частину.

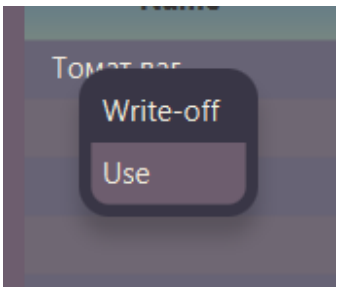

Рисунок 3.6 – Контекстне меню

При натисканні «Write-off» відкривається вікно із об’єктом «slider» який дозволяє обрати кількість, яку необхідно списати (рис. 3.7).

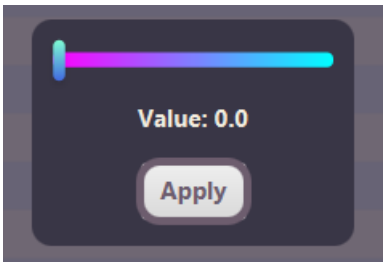

Рисунок 3.7 – Визначення кількості

При натисканні «Apply» на екрані з’явиться вікно причини списання (рис. А.4). У випадку, якщо користувач обирає частину, а не весь товар – його кількість зменшується, а кількість, що була списана – з’являється в таблиці списання. З метою тестування списуються усі томати.

При натисканні «Use» відкривається вікно із об’єктом «slider» який дозволяє обрати кількість, яку необхідно використати (рис. 3.7). Аналогічно списанню, кількість товару, яка була використана відобразиться у відношенні «Used». Важливо зазначити, що відношення «Used» використовується лише для збереження використаних товарі. На поточний момент у застосунку не реалізовано можливість передивлятись використані товари.

Оскільки «Томат ваг.» у кількості 100«kg» було списано – на вкладці «Write-Off» вводимо пошук «ома». Відображаються співпадаючі товари, а серед них – списаний товар із датою 17.05.2023 (дата тестування) та тестовою причиною списання (рис. 3.8).

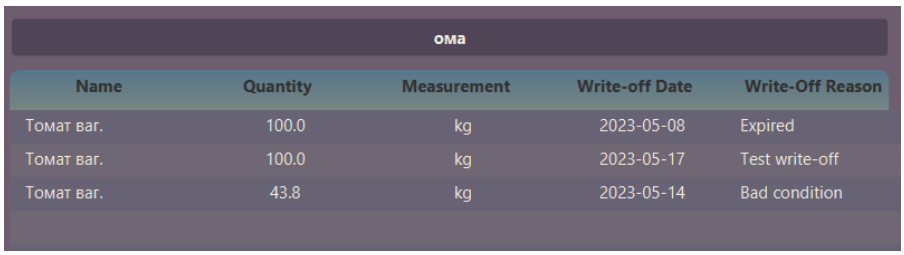

Рисунок 3.8 – Списаний товар

У вкладці «Create Order» (рис. 3.9) користувач може додавати нові замовлення, підтверджувати замовлення, які знаходяться в очікуванні, додавати нових постачальників та нові товари.

Для того, щоб створити замовлення необхідно спершу обрати постачальника. Кожен постачальник має свій список товарів, які постачає.

Одне замовлення можна створити лише для конкретного постачальника. У випадку, якщо користувач додасть низку товарів одного постачальника, а потім обере іншого – товари автоматично видаляться. Для додавання товарів можна розгорнути відповідний ComboBox, або почати вводити в нього назву товару – він автоматично розгорнеться та зобразить усі товари, які співпадають (рис. 3.10).

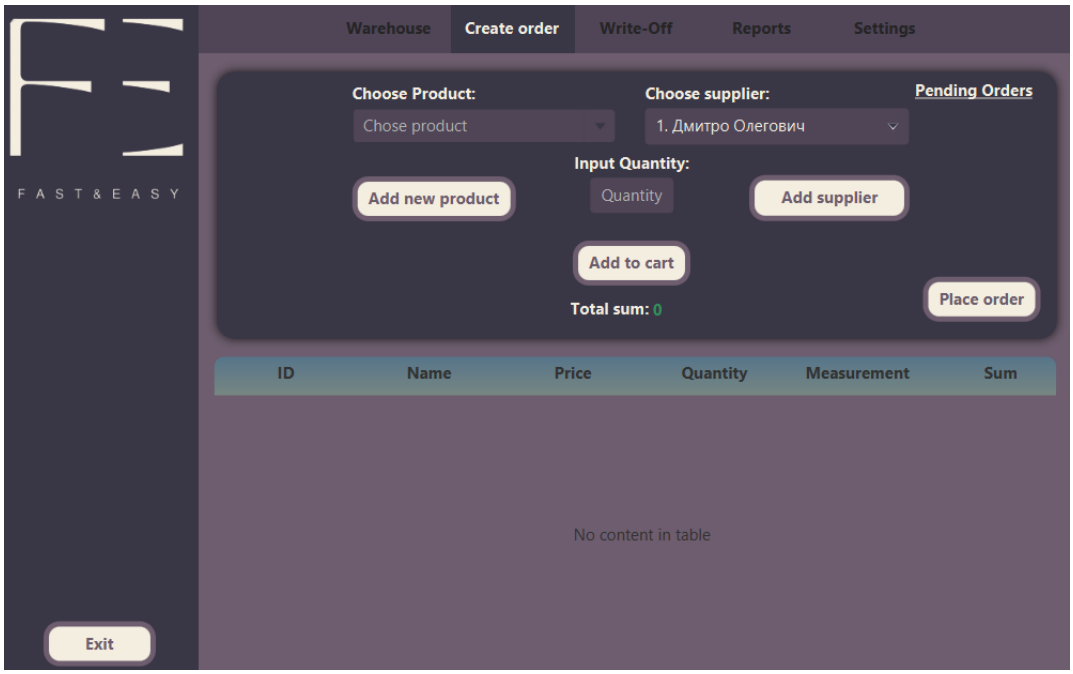

Рисунок 3.9 – Вкладка «Create Order»

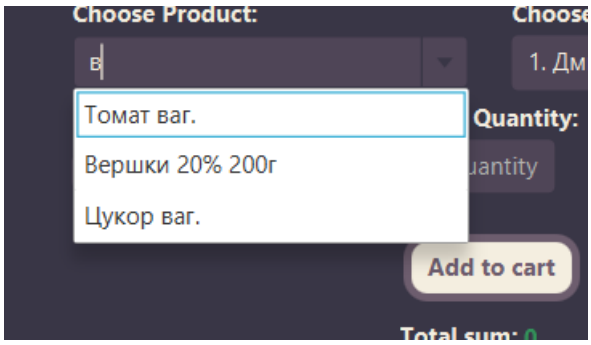

Рисунок 3.10 – Фільтрація товарів за назвою

Обравши товар, необхідно ввести кількість товару в відповідне поле. Після того, як товар було обрано, праворуч від поля для введення кількості буде відображатись одиниця вимірювання цього товару («pcs» або «kg») (рис. 3.11).

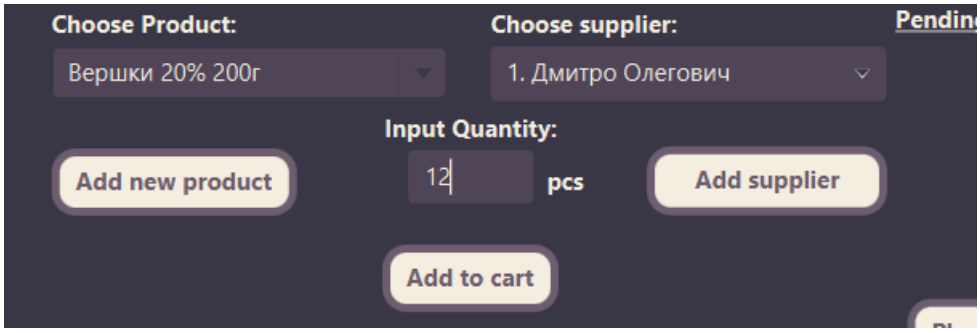

Рисунок 3.11 – Введення кількості товару

Якщо користувач не введе кількість і натисне «Add to cart» – з’явиться помилка (рис. А.5). Якщо значення кількості введене – товар, який було обрано для додавання з’явиться в списку товарів (рис. 3.12).

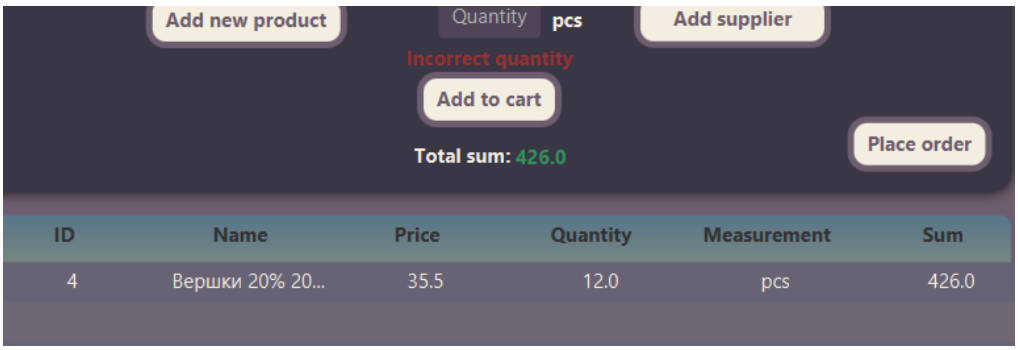

Рисунок 3.12 – Список доданих товарів

У списку, окрім назви, одиниці вимірювання і кількості, відображається ціна цього товару, та вартість обраної кількості цього товару.

Окрім вартості товару в таблиці, в полі «Total sum» відображається загальна сума замовлення. Додаємо ще низку товарів та дивимось на результат (рис. А.6).

Тепер, натиснувши кнопку «Place order» буде створено нове замовлення із статусом «Pending», про що повідомить повідомлення на екрані (рис. А.7) а таблиця і усі поля на вкладці очистяться (рис. А.8).

Тепер, натиснувши на кнопку «Pending Orders» побачимо на екрані вікно із замовленнями, статус яких «Pending» (рис. А.9). Натиснемо на випадаючий список «Products» щоб побачити зміст замовлення (рис. 3.13).

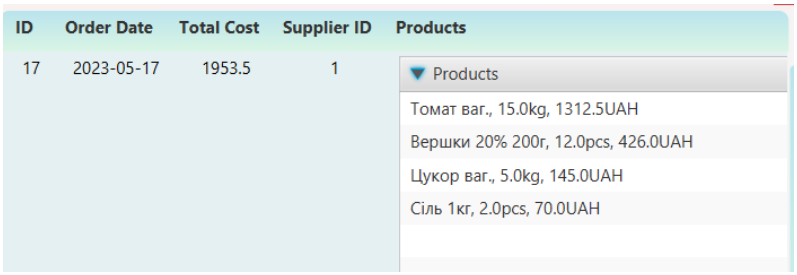

Рисунок 3.13 – Замовлення в очікуванні та його зміст

При натисканні на замовлення правою кнопкою миші відкривається контекстне меню із двома опціями: «Cancel» – скасувати замовлення, та «Accept» – прийняти товари до складу (рис. А.10).

Для тестування натиснемо «Accept». Після натискання на екрані будуть поступово з’являтися вікна з запитом терміну придатності для відповідного товару (рис. 3.14).

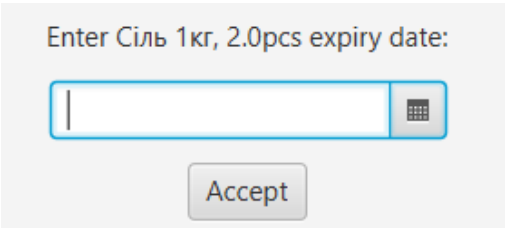

Рисунок 3.14 – Запит терміну придатності

У випадку, якщо користувач ввів термін придатності – товар буде додано до бази даних та видалено із списку замовлення у випадаючому списку «Products», АЛЕ він не буде відображений у таблиці на вкладці «Warehouse», оскільки не буде виконано операцію «COMMIT». Якщо користувач натисне «Accept» без введення дати – хоча б для одного товару і цей товар залишиться в випадаючому списку «Products» – жоден товар не потрапить на склад, адже замовлення не вважається прийнятим. Користувач може закрити вікно очікуючих замовлень та продовжити вводити термін придатності пізніше, але якщо застосунок буде закрито – усі введені дати будуть загублені. Для того, щоб було виконано операцію «COMMIT» необхідно, щоб у списку «Products» не залишилось жодного товару. Після додавання дати останньому товару замовлення буде видалено зі списку очікуючих замовлень (його статус змінюється на «completed», а дата на дату здійснення прийняття) (рис. А.11).

Для тестування введемо датою для усіх полів 01.05.2025. Після введення дати та автоматичного видалення замовлення зі списку очікуючих замовлень повертаємось до списку товарів на складі. Сортуємо кортежі за датою, натискаючи на назву стовпця і бачимо 4 записи із датою «01.05.2025» (рис. 3.15).

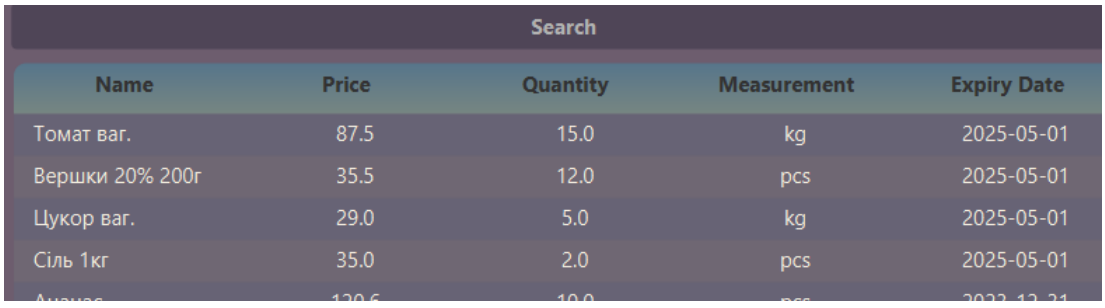

Рисунок 3.15 – Товари, що додано до складу

Якщо необхідно додати нового постачальника – необхідно натиснути на кнопку «Add supplier» на вкладці «Order create». Після натискання на екрані відобразиться вікно додавання нового постачальника (рис. 3.16).

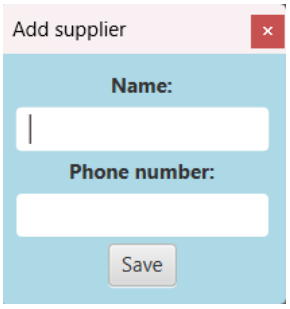

Рисунок 3.16 – Вікно додавання нового постачальника

Вводимо ім’я «Семен Семенович» та номер «0991234567». Натискаємо «Save». На екрані відобразиться повідомлення про успішне додавання постачальника (рис. А.12).

Після того, як було створено нового постачальника, він відобразиться у списку постачальників, а його список товарів буде пустим (рис. А.13).

Для додавання постачальнику товарів, необхідно натиснути кнопку «Add new product». На екрані відобразиться вікно додавання нового товару (рис. 3.17). Одразу заповнимо усі поля тестовим значенням та натиснемо кнопку «Save».

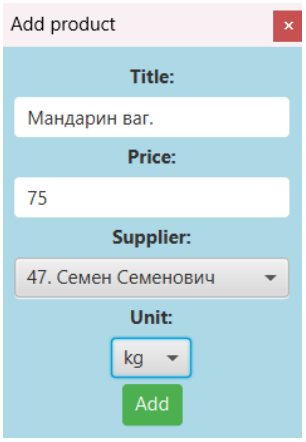

Рисунок 3.17 – Вікно додавання нового товару постачальнику

Після натискання кнопки «Save» на екрані відобразиться повідомлення про успішне додавання продукту (рис. А.14). В тестових цілях додамо ще декілька товарів. На рисунку 3.18 зображено товари, які було додано новому постачальнику.

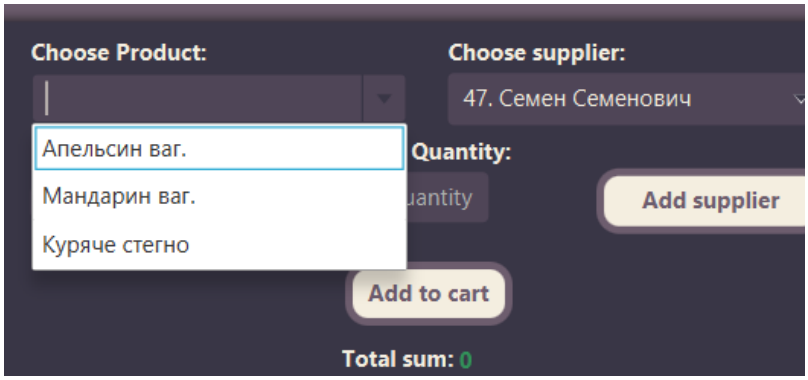

Рисунок 3.18 – Товари, додані новому постачальнику

Товари, що було додано можна додавати до корзини та розташовувати замовлення. Додаємо по 1 «kg» кожного товару до замовлення. Результат можна побачити на рисунку А.15.

Функція додавання товарів реалізує перевірку на дублікати. Додамо ще по одному товару. Товари не дублюються, а збільшується їх кількість у списку (рис. 3.19).

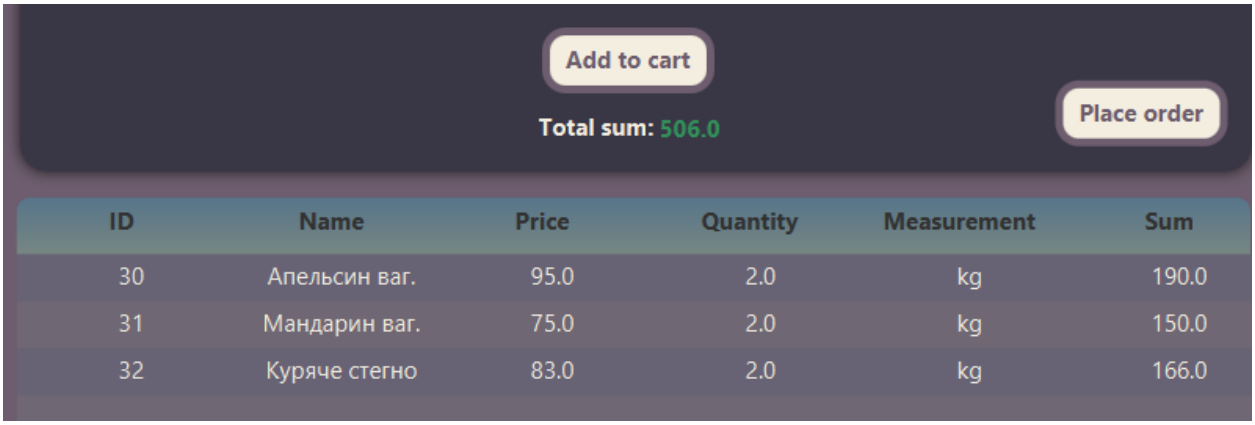

Рисунок 3.19 – Робота функції запобігання дублікатів

Створимо це замовлення для майбутнього тестування.

На вкладці «Reports» користувач може створювати різні звіти (рис. 3.20).

Рисунок 3.20 – Вкладка «Reports»

Обираємо великий діапазон дат для отримання усіх даних з відношень. Натискаємо кнопку «Write-offed Products by reason Report». На екрані з'являється поле, в яке необхідно ввести причину списання, за якою створювати звіт (рис. А.16). Вводимо в поле «Expired» та натискаємо «Ok». Отримуємо повідомлення (рис. А.17) про успішне створення звіту. У папці застосунку створено звіт із назвою «Report + дата та час створення звіту». Для інших звітів вводити нічого не потрібно. Натискаємо кнопку кожного звіту повідомлення успішної генерації звіту після кожного натискання (рис. А.17). По завершенню генерації звітів в папці застосунку буде знаходитись 5 різних звітів. Зміст звіту списаних товарів за причиною зображено на рисунку 3.21. Зміст звітів списані товари, постачальники, виконані замовлення, замовлення в очікуванні зображено на рисунках А.18 – А.21 відповідно.

	A	B	C	D	E	F	G
1	Назва товару	Кількість	Одиниця вимірювання	Дата списання	Причина списання	Вартість товару	
2							
3	Томат ваг.	100	kg	2023-05-08 22:57:44	Expired	8750	
4	Куряче філе ваг.	75	kg	2023-05-08 22:57:44	Expired	9675	
5	Куряче філе ваг.	75	kg	2023-05-08 22:57:44	Expired	9675	
6	Вершки 20% 200г	2	pcs	2023-05-16 22:10:52	Expired	71	
7	Вершки 20% 200г	2	pcs	2023-05-16 22:10:52	Expired	71	
8	Вершки 20% 200г	2	pcs	2023-05-16 22:10:52	Expired	71	
9	Вершки 20% 200г	2	pcs	2023-05-16 21:52:57	Expired	71	
10	Вершки 20% 200г	2	pcs	2023-05-16 22:18:38	Expired	71	
11	Огірок ваг.	1	kg	2023-05-16 22:10:52	Expired	94	
12	Огірок ваг.	1	kg	2023-05-16 22:10:52	Expired	94	
13	Огірок ваг.	1	kg	2023-05-16 22:10:52	Expired	94	
14	Огірок ваг.	10	kg	2023-05-08 22:54:41	Expired	940	
15	Огірок ваг.	1	kg	2023-05-16 21:52:48	Expired	94	
16	Огірок ваг.	1	kg	2023-05-16 22:18:37	Expired	94	
17	Сметана ваг.	20	kg	2023-05-08 22:54:41	Expired	1560	
18	Ананас	5	pcs	2023-05-08 22:54:41	Expired	603	
19	Ананас	123	pcs	2023-05-16 22:18:37	Expired	14833,8	
20	Картопля ваг.	5	kg	2023-05-08 22:54:41	Expired	49,25	
21	Лимон ваг.	8	kg	2023-05-08 22:54:41	Expired	396	
22	Цукор ваг.	3	kg	2023-05-08 22:54:41	Expired	87	
23	Цукор ваг.	123	kg	2023-05-16 22:18:36	Expired	3567	
24	Цукор ваг.	1,5	kg	2023-05-16 22:25:49	Expired	43,5	
25	Сіль 1кг	5	pcs	2023-05-16 22:25:48	Expired	175	
26							
27					Загальна сума:	51179,55	

Рисунок 3.21 – Списані товарів за причиною

Остання для тестування вкладка – вкладка «Settings». Вкладка ділиться на 4 плитки: плитка зміни паролю, плитка видалення даних, плитка автоматичного списання, плитка очищення БД (рис. 3.22).

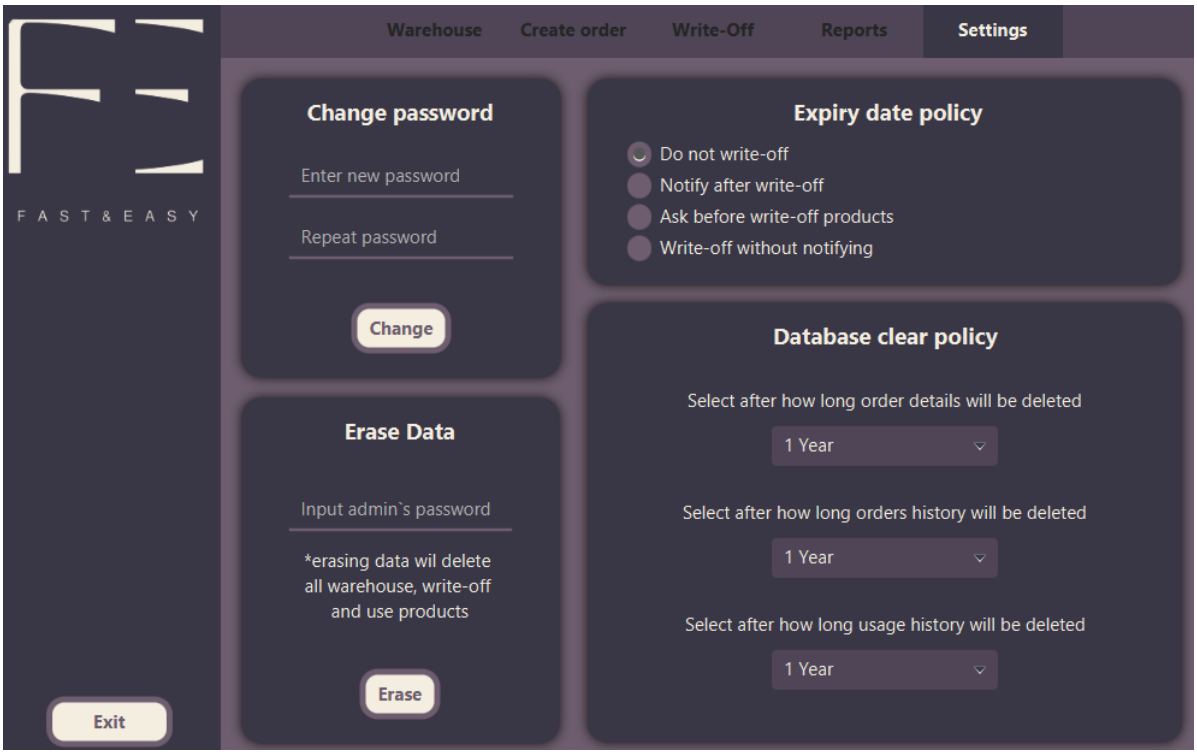

Рисунок 3.22 – Вкладка «Settings»

Плитка змінення паролю дозволяє користувачу змінити пароль. Для тестування змінимо пароль, який було створено автоматично і відправлено в СМС-повідомлені. При введенні паролю, у реальному часі оновлюється напис, що інформує користувача, чи правильні данні введені (рис. 3.23).

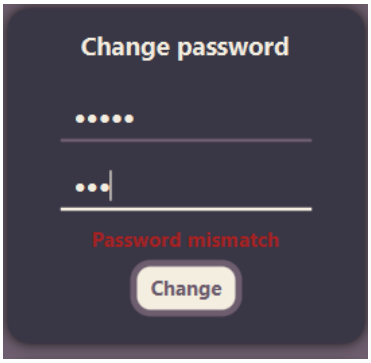

Рисунок 3.23 – Валідація введених даних у реальному часі

Якщо паролі збігаються, проте кількість символів менше 5 або більше 16 – користувач побачить відповідне повідомлення (рис. А.22). У випадку, коли довжина у діапазоні 5-16 символів та паролі співпадають, при натисканні кнопки «Change» користувач побачить вікно для введення коду (рис. А.23), а на його номер телефону із БД буде відправлено СМС-повідомлення з кодом (рис. А.24). Якщо ввести правильний код у поле – пароль буде змінено, а користувач побачить відповідне повідомлення (рис. А.25), а якщо код введено невірно, відобразиться повідомлення із помилкою (рис. А.26).

Плитка автоматично списання дозволяє користувачу встановити політику видалення товарів із терміном придатності, що закінчився. Якщо обрано «Do not write-off» списання можна зробити лише в ручному режимі. Якщо обрано «Notify after write-off» – користувач отримає повідомлення при наступній ініціалізації, що усі прострочені позиції було списано (рис. А.27). Якщо обрано «Ask before write-off products» – під час наступної ініціалізації система запитає користувача, чи потрібно списувати прострочений

товар (рис. А.28). Якщо обрано «Write-off without notifying» – видалення товарів відбудеться автоматично без будь-якого повідомлення.

Плитка очищення БД відповідає за параметри видалення старих даних із БД. Виставимо значення першого поля «3 months», другого поля 6 «months» і третього – «Do mot delete». Тепер створимо два замовлення, із тестовим статусом «Test» (рис. А.29) і в кожне замовлення додаємо по 2 товари (рис. А.30). Зімітуємо, що замовлення з id=20, створено місяць тому, а замовлення з id=21 було створено 1 рік і 4 місяці тому.

Оскільки ми вказали не видаляти дані із таблиці «Used» – вони залишилися (рис. А.31). Дані замовлення(20) не видалено, оскільки воно було створено місяць тому, а замовлення(21) було повністю видалено із БД (рис. А.32).

Останнім етапом тестування є функція очищення бази даних. Для знищення даних необхідно ввести пароль користувача логін якого=admin (необхідно ввести пароль адміністратора). Якщо пароль введено вірно – після натискання кнопки «Erase» на екрані з’явиться вікно підтвердження (рис. 3.24), де синім кольором перекрито персональний номер телефону, а в СМС-повідомленні на вказаний номер буде відправлено код підтвердження (рис. 3.25).

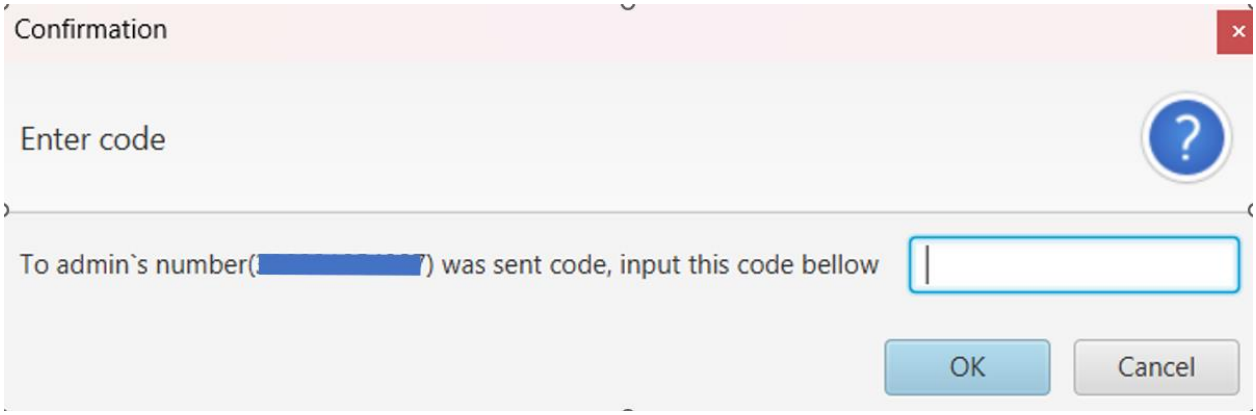

Рисунок 3.24 – Вікно підтвердження

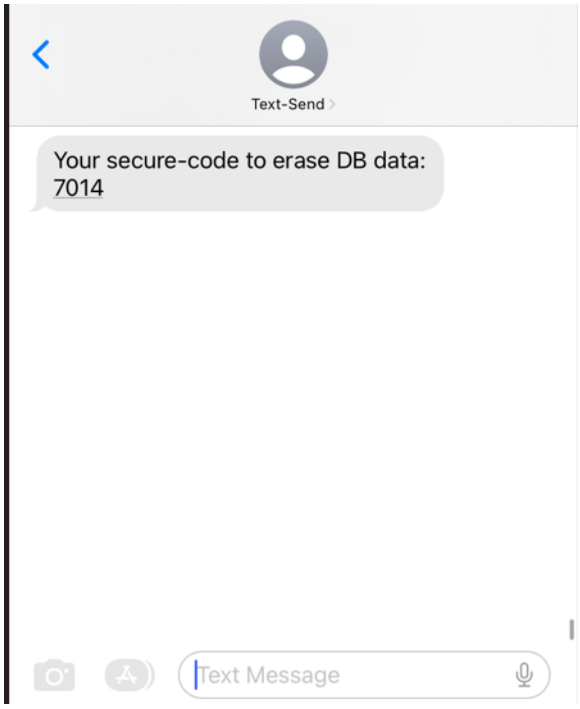

Рисунок 3.25 – Код підтвердження видалення даних

Якщо код із СМС-повідомлення введено вірно користувач побачить вікно підтвердження намірів (рис. 3.26).

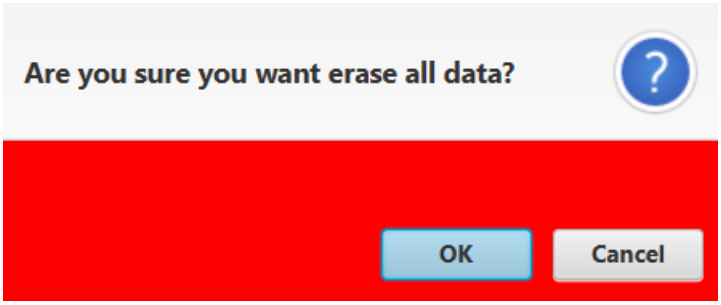

Рисунок 3.26 – Вікно підтвердження намірів видалення інформації

У випадку підтвердження – дані будуть видалені, а застосунок автоматично закриється. Видалення не стосується відношення «DBUsers», дані із цієї таблиці не видаляються під час очищення.

3.4 Виявлені недоліки

Під час тестування було виявлено, що при виборі товару у ComboBox для додавання у замовлення (рис. 3.9) його можна видалити лише виділивши цілком і натиснувши «BackSpace». При спробі стерти по одному символу з’являється Exception. Це зумовлено тим, що на об’єкт ComboBox було додано слухач введення тексту, який після кожного введення оновлює дані у випадаючому списку і під час стирання обраного товару відбувається нескінченна рекурсія, що призводить до переповнення стеку.

Рекурсія – спосіб організації обчислювального процесу, при якому підпрограма під час виконання складових її операторів звертається сама до себе [30].

В слухач було вбудовано перевірку, що обмежує рекурсивний виклик у випадку, коли видаляється увесь товар цілком, проте досягнути безпомилкового видалення товару по символу не вдалося. Була протестована велика вибірка слухачів із різними перевірками, що не призвело до вирішення питання. Залучення штучного інтелекту «ChatGPT» до вирішення проблеми не принесло результатів.

На поточний момент на відомих інтернет-ресурсах та форумах не вдалось знайти рішення. На момент написання кваліфікаційної роботи продовжуються спроби досягти безпомилкового видалення товару.

ВИСНОВКИ

У даній кваліфікаційній роботі був розроблений застосунок на мові Java з використанням JavaFX та OracleDB для обліку товарів на складі ресторану з назвою «Fast&Easy». Застосунок надає широкий функціонал для керування товарами, постачальниками, замовленнями та створенням звітів.

У першу чергу, застосунок забезпечує можливість списування або використання товарів. Ця функція є важливою для забезпечення актуальності даних про кількість товарів на складі. Користувачі можуть вибрати конкретний товар, вказати кількість, яку необхідно списати або використати, і оновити інформацію у базі даних.

Застосунок також дозволяє додавати нові товари у систему. Користувачі можуть ввести всю необхідну інформацію про товар, таку як назва, опис, ціна тощо, і зберегти ці дані у базі даних. Це дозволяє зручно відстежувати наявність різних товарів на складі та контролювати їх доступність.

Ще одна важлива функція застосунку – це можливість додавати нових постачальників. Записи про постачальників зберігаються в базі даних і включають інформацію про назву, контактну особу, електронну пошту, телефон тощо. Це дозволяє управляти списком постачальників та швидко знаходити необхідну інформацію про них.

Застосунок також має можливість створювати замовлення та приймати замовлення. Користувачі можуть вибрати потрібний товар, вказати кількість та вибрати постачальника. Дані про замовлення зберігаються у базі даних і можуть бути легко відстеженні та оновлені. Замовлення дозволяють ефективно керувати постачанням товарів та забезпечити своєчасну доставку.

Одна з важливих особливостей застосунку – це реалізація підтвердження особистості через SMS-повідомлення. Ця функція забезпечує високий рівень безпеки для доступу до системи. Користувачі отримують SMS-повідомлення з унікальними кодами, які потрібно ввести для підтвердження своєї

ідентичності. Це додає додатковий шар захисту від несанкціонованого доступу.

При цьому, застосунок «Fast&Easy» не позбавлений недоліків. Серед них можна виділити невелику кількість звітів і мінімальний рух товарів в системі.

Ці недоліки не складно усунути при масштабуванні проєкту, але для цього необхідно розглянути конкретні вимоги та розробити відповідне ТЗ.

Результати роботи апробовано у вигляді тез доповідей під час Міжнародного молодіжного форуму «РАДІОЕЛЕКТРОНІКА І МОЛОДЬ У XXI СТОЛІТТІ» [31].

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Любченко В.А., (2019) Об’єктно-орієнтоване програмування: КНМЗ. *Харків: ХНУРЕ, 423 с.*

2. Кобилін О. А., (2017) Технологія програмування: КНМЗ. *Харків: ХНУРЕ.*

3. Тітова О. В., (2018) Системи управління базами даних: КНМЗ. *Харків: ХНУРЕ, 311 с.*

4. Oracle Database. URL: https://www.oracle.com/cis/database/ (дата звернення 14.04.2023).

5. Мизніков Р. І., (2020) Дослідження та розробка методів аналізу даних для класифікації заявок на отримання банківських кредитів: кваліфікаційна робота здобувача вищої освіти, пояснювальна записка. *Харків: ХНУРЕ, 77 с.*

6. Руденко Д. О., (2017) Програмування: КНМЗ. *Харків: ХНУРЕ, 307 с.*

7. Ultra Company. URL: https://ultra-company.com/ua/ (дата звернення 14.04.2023).

8. Порівняння переваг та недоліків систем автоматизації бізнесу IIKO і Poster. URL: https://konotop.city/articles/213176/porivnyannya-perevagi-ta-nedoliki-sistem-avtomatizacii-biznesu-iiko-i-poster (дата звернення 14.04.2023).

9. Poseter. URL: https://joinposter.com/ua/business/restaurant (дата звернення 15.04.2023).

10. Любченко В. А., (2017) Алгоритмізація та програмування: КНМЗ. *Харків: ХНУРЕ, 405 с.*

11. Реляційна модель даних. URL: http://xn--r1a3b.xn--b1amgblet.xn--j1amh/index.php/%D0%A0%D0%B5%D0%BB%D1%8F%D1%86%D1%96%D0%B9%D0%BD%D0%B0_%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C_%D0%B4%D0%B0%D0%BD%D0%B8%D1%85 (дата звернення 03.05.2023).

12. Бази даних і знань як інформаційні мережі. URL: https://nmetau.edu.ua/file/04_4.2_lbr_gr_rbr_.pdf (дата звернення 03.05.2023).

13. Яковлева О. В., (2017) Організація баз даних і знань: КНМЗ. *Харків: ХНУРЕ, 386 с.*

14. Поняття відношення. URL: https://elearning.sumdu.edu.ua/free_content/lectured:89b3d175c06a6b137e410cb14821d0e94549ad5a/20151030211833/44448/index.html (дата звернення 03.05.2023).

15. Реляційна алгебра. URL: https://rdb.dp.ua/uk/chapter_05 (дата звернення 03.05.2023).

16. Яковлева О. В., (2017) Бази даних та інформаційні системи: КНМЗ. *Харків: ХНУРЕ, 445 с.*

17. Булатецька Л. В. , Булатецький В. В. (2020) Методичні вказівки для підготовки до контрольної роботи з нормативних навчальних дисциплін «Бази даних та розподілені інформаційно-аналітичні системи» та «Організація баз даних та знань»: навч.посібник. *Луцьк: ВНУ імені Лесі Українки*.

18. Висоцький Д. О., (2021) Дослідження та розробка медичної інформаційної системи для підтримки паліативних пацієнтів: кваліфікаційна робота здобувача вищої освіти, пояснювальна записка. *Харків: ХНУРЕ*.

19. ER-діаграма – це. URL: https://hi-news.pp.ua/kompyuteri/14668-er-dagrama-ce-opis-vidi-pravila-pobudovi.html (дата звернення 03.05.2023).

20. ER-діаграма – Опис, види, правила. URL: https://uaeu.top/digital-online/er-diagrama-tse-opis-vidi-pravila-pobudovi.html (дата звернення 03.05.2023).

21. Поняття ER-моделі. Поняття сутності (entity). Атрибути. Види атрибутів. URL: https://www.bestprog.net/uk/2019/01/24/the-concept-of-er-model-the-concept-of-essence-and-communication-attributes-attribute-types-ua/ (дата звернення 03.05.2023).

22. Хешування. URL: https://wiki.tntu.edu.ua/Хешування (дата звернення 03.05.2023).

23. Шифрування, хешування, засолювання: у чому різниця та коли застосовувати. URL: https://instagalleryapp.com/informacijnabezpeka/shifruvannjaheshuvannja-zasoljuvannja-u-chomu/ (дата звернення 03.05.2023).

24. Бондар Д. Ю., (2020) Дослідження методів і розробка сайту для підприємств телекомунікаційних послуг : атестаційна робота здобувача вищої освіти, пояснювальна записка. *Харків: ХНУРЕ, 100 с.*

25. JavaFX Scene Builder. URL: https://www.oracle.com/java/technologies/javase/javafxscenebuilder-info.html (дата звернення 03.05.2023).

26. Піщухіна О. О., (2017) Проектування програмних систем: КНМЗ. *Харків: ХНУРЕ, 76 с.*

27. Руденко Д. О., (2017) Об’єктно-орієнтоване програмування з використанням Pyton: -. *Харків: ХНУРЕ, 121с.*

28. Куліков Ю. О., (2017) Програмне забезпечення обчислювальних систем: КНМЗ. *Харків: ХНУРЕ, 102 с.*

29. Гороховатський В. О., (2017) Алгоритми та структури даних: КНМЗ. *Харків: ХНУРЕ, 50 с.*

30. Шафроненко А. Ю., Шергін В.Л., (2019) Теорія алгоритмів: методичні вказівки до лабораторних робіт. *Харків: ХНУРЕ, 24с.*

31. Артьомов Д. О., (2023) Огляд застосунку ULTRA для обліку та списання товарів у ресторанному бізнесі: 27-й Міжнародний молодіжний форум «Радіоелектроніка і молодь у XXI столітті». Зб. матеріалів форуму. Т. 7. Харків: ХНУРЕ. С. 82-83.